YAYASAN PRIMA AGUS TEKNIK

HTML CSS dan Javascript

Muhammad Sholikhan, S.Kom., M.Kom

HTML, CSS dan Javascript

Penulis :
Muhammad Sholikhan, S.Kom., M.Kom

ISBN :

Editor :
Dr. Joseph Teguh Santoso, S.Kom., M.Kom.

Penyunting :
Dr. Mars Caroline Wibowo. S.T., M.Mm.Tech

Desain Sampul dan Tata Letak :
Irdha Yunianto, S.Ds.,M.Kom.

Penebit :
Yayasan Prima Agus Teknik Bekerja sama dengan
Universitas Sains & Teknologi Komputer (Universitas STEKOM)

Redaksi :
Jl. Majapahit no 605 Semarang
Telp. (024) 6723456
Fax. 024-6710144
Email : penerbit_ypat@stekom.ac.id

Distributor Tunggal :
Universitas STEKOM
Jl. Majapahit no 605 Semarang
Telp. (024) 6723456
Fax. 024-6710144
Email : info@stekom.ac.id

KATA PENGANTAR

Puji syukur pada Tuhan Yang Maha Esa bahwa buku yang berjudul "**HTML, CSS & JavaScript**" ini dapat diselesaikan dengan baik. Pernahkah kalian berpikir bagaimana sebuah website bisa dibuat dengan tampilan yang menarik dan user friendly serta dapat berjalan secara interaktif terhadap pengunjung/pengguna website itu sendiri? Sebuah website agar mempunyai UI dan UX yang baik tidak hanya menggunakan satu bahasa pemrograman saja, kita harus mengombinasikan beberapa bahasa pemrograman agar menghasilkan sebuah website yang baik. Sesuai dengan judul buku, buku ini memuat tentang HTML, CSS, dan Javascript. HTML (Hypertext Markup Language) merupakan bahasa pemrograman standar untuk dokumen yang dirancang sebagai kerangka atau susunan sebuah website yang kemudian akan diatur beberapa komponen supaya lebih terstruktur dan mempercantik kerangka HTML menggunakan CSS (Cascading Style Sheet). Terakhir akan ditambahkan sebuah bahasa pemrograman javascript yang mempunyai peran sebagai logika dan cara berpikirnya, seperti saat tombol ditekan akan muncul sesuatu. Javascript dapat disisipkan ke sebuah halaman web dengan tag script.

Buku ini cocok bagi anda yang tertarik untuk memulai membuat sebuah website karena berisikan pemahaman dasar mengenai bahasa pemrograman HTML, CSS, dan Javascript mulai dari teori, aturan dan tata cara penulisan coding yang baik, dan cara menghubungkan antara HTML, CSS, dan Javascript. Semoga buku ini bermanfaat bagi pembaca, akhir kata penulis ucapkkan terima kasih.

Penulis, September 2022

Muhammad Sholikhan, S.Kom., M.Kom.

DAFTAR ISI

Halaman Judul ..... **i**
Kata Pengantar ..... **iii**
Daftar Isi ..... **iv**
BAB 1 HTML **1**
1.1 Membuat Halaman Dasar dengan HTML 1
1.2 Memahami Block Bangunan HTML 1
1.3 Jenis Dokumen 1
1.4 Membuat HTML yang Baik 4
1.5 Perbedaan Antara Elemen Div dan Span 5
1.6 Berlatih Membuat Tabel 14
1.7 Menyertakan Tautan dan Gambar di Halaman Web Anda 16
1.8 Menulis HTML Valid 23
BAB 2 CSS **27**
2.1 Menambahkan Style dengan CSS 27
2.2 Keterbatasan CSS 28
2.3 Tipe Gaya 29
2.4 Menyambungkan CSS ke Halaman 30
2.5 Cascade CSS 32
2.6 Mengimpor CSS dari Dalam HTML 38
2.7 Comment CSS 40
2.8 Menggunakan Kelas 42
2.9 Aturan CSS 43
2.10 Font dan Tipografi 44
2.11 Menambahkan Border 52
2.12 Mengelola Gaya Teks 55
2.13 Menyesuaikan Padding 58
2.14 Selektor CSS 60
2.15 Memilih Berdasarkan Grup 65
2.16 Menambahkan Background 69
2.17 Warna CSS 71
2.18 Elemen Pemosisian 74
2.19 PseudeClass 76
2.20 Menambahkan Gambar Berulang 81
2.21 Membuat Layout Halaman 82
2.22 Menambahkan Header dan Footer ke Halaman 92
BAB 3 MEMBUAT DAN MENATA FORMULIR WEB MENGGUNAKAN CSS **98**
3.1 Menggunakan Formulir Web untuk Mendapatkan Informasi 98
3.2 Membuat Formulir 100

3.3 Mengetahui Perbedaan Metode GET dan POST 101
3.4 Menggunakan CSS untuk Menyejajarkan Bidang Formulir 109
BAB 4 HTML TINGKAT LANJUT DENGAN CSS3 113
4.1 Selektor Atribut 113
4.2 Properti Box-sizing 114
4.3 Background CSS3 115
4.4 Border CSS3 119
4.5 Bayangan Kotak 123
4.6 Elemen Overflow 124
4.7 Warna dan Opacity 126
4.8 Efek Teks 128
4.9 Font Web 129
4.10 Transformasi 131
4.11 Transisi 133
BAB 5 HTML5 DAN FITURNYA 137
5.1 Pengenalan HTML5 137
5.2 Geolokasi dan Layanan GPS 142
5.3 Penyimpanan Lokal 146
5.4 Pekerja Web 149
5.5 Aplikasi Web Offline 151
5.6 Drop-down 152
5.7 Pesan Lintas Dokumen 154
5.8 Mikrodata 158
5.9 Tag HTML5 Lainnya 160
BAB 6 MEMAHAMI DASAR JAVASCRIPT 162
6.1 Melihat Dunia JavaScriptld dari JavaScript 162
6.2 Memeriksa Cara Menambahkan JavaScript ke Halaman 163
6.3 Menjelajahi JavaScript 165
6.4 JavaScript dan Teks HTML 165
6.5 Termasuk File JavaScript 168
6.6 Men-debug Kesalahan JavaScript 168
6.7 Menggunakan Komentar 170
6.8 Titik Koma 170
6.9 Variabel 170
6.10 Array 171
6.11 Mengembalikan Nilai 177
6.12 Operator 178
BAB 7 MEMBANGUN PROGRAM JAVASCRIPT 186
7.1 Memulai dengan Pemrograman JavaScript 186
7.2 Fungsi javaScript, Objek dan Array 198
7.3 Ekspresi dan Control Flow di JavaScript 199
7.4 Objek JavaScript 208

7.5 Menggunakan Fungsi untuk Menghindari Perulangan 212
7.6 Bekerja dengan Dokumen HTML 218
7.7 Model Objek Dokumen 221
BAB 8 MENAMBAHKAN JQUERY 228
8.1 Pengenalan jQuery 228
8.2 Menginstal jQuery 229
8.3 Menggunakan jQuery yang Dihosting CDN 229
8.4 Menggabungkan Fungsi jQuert ready() 231
8.5 Memilih Elemen dengan jQuery 232
8.6 Bekerja dengan HTML Menggunakan jQuerry 234
8.7 Mengubah Atribut dan Gaya 236
BAB 9 REAKSI EVENT DENGAN JAVASCRIPT DAN JQUERY 243
9.1 Memahami Event 243
9.2 Bekerja dengan Formulir 243
9.3 Memantau Mouse Event 249
9.4 Reaksi Peristiwa Keyboard 255
BAB 10 TROUBLESHOOT PROGRAM JAVASCRIPT..... 261
10.1 Mempekerjakan Teknik Pemecahan Masalah JavaScript Dasar 261
10.2 Mengidentifikasi Masalah JavaScript dengan Firebug 263
BAB 11 VALIDASI JAVASCRIPT DAN PHP DAN PENANGANAN KESALAHAN 267
11.1 Memvalidasi Input Pengguna dengan JavaScript 267
11.2 Ekspresi Reguler 272
11.3 Beberapa Contoh yang Lebih Rumit 275
11.4 Ringkasan Karakter Meta 276
11.5 Modifier Umum 278
11.6 Menggunakan Ekspresi Reguler dalam JavaScript 278
11.7 Menggunakan Ekspresi Reguler di PHP 278
11.8 Menampilkan Kembali Formulir Setelah Validasi PHP 279
BAB 12 MENGGUNAKAN AJAX 285
12.1 Apa itu Ajax? 285
12.2 Program Ajax Pertama Anda 287
12.3 Properti readyState 289
12.4 Menggunakan GET Alih-alih POST 291
12.5 Tentang XML 296
12.6 Menggunakan Kerangka kerja untuk Ajax 298
BAB 13 MENGAKSES CSS DARI JAVASCRIPT..... 299
13.1 Meninjau Kembali Fungsi getElementById 299
13.2 Memasukkan Fungsi 302
13.3 Mengakses Properti CSS dari JavaScript 303
13.4 JavaScript Sebaris 306
13.5 Melampirkan Acara ke Objek dalam Script 307
13.6 Menambahkan Elemen Baru 308

13.7 Menggunakan Interupsi 311
DAFTAR PUSTAKA 316

BAB 1
HTML

1.1 MEMBUAT HALAMAN DASAR DENGAN HTML

HyperText Markup Language (HTML) adalah bahasa web. Saat kita membuka halaman web di browser web seperti Internet Explorer, Firefox, atau Safari, browser akan mengunduh dan menampilkan HTML. Pada intinya, HTML hanyalah sebuah dokumen, sama seperti dokumen yang dibuat di word. Program seperti Microsoft Word digunakan untuk melihat dokumen pengolah kata karena microsoft word tahu cara membaca dan menampilkan teks. Demikian juga dengan web, browser web adalah program yang mengetahui cara membaca dan menampilkan dokumen yang dibuat dengan HTML.

Dokumen Word Process dapat dibuat dan dibaca dengan satu program. Di sisi lain, dokumen HTML membutuhkan program yang berbeda untuk pembuatan dan pembacaan; Kita tidak dapat membuat dokumen HTML menggunakan browser. Untuk membuat dokumen HTML kita membutuhkan program yang disebut editor. Editor ini bisa sesederhana program Notepad yang disertakan dengan Microsoft Windows atau serumit Eclipse dan Microsoft Visual Studio. Umumnya kita dapat menggunakan program yang sama untuk membuat dokumen HTML yang digunakan untuk membuat program PHP.

1.2 MEMAHAMI BLOK BANGUNAN HTML

Dokumen HTML dapat disimpan di folder *Documents* dalam komputer. Namun jika dibuka, hanya kita saja yang bisa melihat isinya. Untuk menyiasati hal ini, kita memerlukan lebih banyak sumber daya, sebut saja sebagai server web. Menyimpan dokumen di server web memungkinkan orang lain untuk dapat melihat dokumen kita. Server web adalah komputer yang menjalankan perangkat lunak khusus yang mengetahui cara mengirim (atau menyajikan) halaman web ke banyak orang pada saat yang bersamaan. Dokumen HTML diatur dalam urutan tertentu, dengan bagian-bagian tertentu. Mereka terstruktur sedemikian rupa agar browser web dapat membaca dan menampilkannya. Karena hal tersebut, ketika kita membuat dokumen HTML, kita harus mengikuti strukturnya dan menyiapkan dokumen agar browser dapat membacanya.

1.3 JENIS DOKUMEN

Browser web dapat menampilkan beberapa jenis dokumen, tidak hanya HTML, jadi saat membuat dokumen web, hal pertama yang harus lakukan adalah memberi tahu browser tentang jenis dokumen apa yang akan digunakan. Kita karus menetapkan jenis dokumen dengan baris khusus HTML di bagian atas dokumen. Dalam istilah teknis, tipe dokumen disebut *Document Text Declaration*, atau disingkat DTD. Dalam versi HTML sebelumnya, developer terus-menerus menyalin dan menempelkan jenis dokumen ke dalam dokumen karena panjang dan rumit. Dengan dirilisnya versi terbaru HTML, yang disebut HTML5, jenis dokumen menjadi sangat disederhanakan. Jenis dokumen untuk HTML5 adalah:

```
<!doctype html>
```

Ini akan menjadi baris pertama dari setiap dokumen HTML yang kita buat, sebelum membuat dokumen lainnya. Setiap kali, kita perlu menampilkan HTML dan menyertakan jenis dokumennya yang kadang-kadang disebut *doctype*.

Bisa jadi kita akan tergoda untuk menggunakan <!doctype html5>, tetapi tidak ada nomor versi yang terkait dengan jenis dokumen HTML5. Karena ketika versi HTML berikutnya

keluar, kita tidak perlu lalgu memperbarui atau mengubah semua jenis dokumen yang telah kita buat pada HTML sebelumnya ke versi HTML6. Ada berbagai jenis dokumen lain yang lebih lama, diantaranya adalah:

HTML 4.01 Strict <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">

HTML 4.01 Transitional<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">

XHTML 1.0 Strict <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

XHTML 1.1 DTD <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

Jenis dokumen lain selain diatas juga ada, dan kebanyakan sama-sama rumit dan sulit untuk diingat. Jika kita melihat jenis dokumen ini di halaman web, kita akan tahu bahwa halaman tersebut mungkin menggunakan sintaks yang sedikit berbeda untuk membuat dokumen HTML-nya. Dokumen HTML terdiri dari huruf dan kata yang diapit tanda kurung siku, terkadang disebut tanda kurang dari atau lebih besar dari:

< >

Ini adalah elemen utama dalam dokumen HTML dan disebut sebagai elemen root:

<html>

Biasanya, elemen HTML memiliki tag pembuka < dan penutup >. Elemen ditutup dengan garis miring di depan nama elemen. Melihat <html> dalam dokumen berarti nantinya dokumen tersebut akan memiliki </html> untuk menutup elemen tersebut. Fungsi pembukaan <html> dan penutup </html> ini untuk membentuk dokumen yang dibungkus di dalam elemen-elemen tersebut.

Bagian dari Dokumen HTML

Selanjutnya, kita akan melihat tiga bagian utama dari dokumen HTML, kita juga akan melihat perbedaan antara dokumen HTML dan halaman HTML?. Kedua istilah tersebut dapat dipertukarkan. Daftar dibawah ini menunjukkan seluruh dokumen HTML.

Contoh 1 Daftar Halaman Web Dasar

```
<!doctype html>
<html>
<head>
<title>My First Document</title>
</head>
<body>
<div>My Web Page</div>
</body>
</html>
```

Teks HTML ini akan muncul di browser web seperti berikut ini:

My Web Page

Halaman memiiki judul di title bar atau tab bar browser dan teks yang menyatakan isi dari dokumen HTML tersebut. Bagian selanjutnya dalam bab ini menjelaskan cara menyempurnakan halaman ini dengan lebih banyak elemen HTML dan teks.

Elemen root

Meskipun bukan bagian dari dokumen HTML, elemen root adalah yang membungkus seluruh dokumen, muncul sebagai hal pertama setelah *doctype* dan hal terakhir dalam dokumen.

Elemen root dibuka dengan:

<html>

Elemen root ditutup dengan:

</html>

Bagian head dan elemen judul

Bagian head dokumen berisi informasi tentang dokumen itu sendiri. Bagian head dibuka dengan:

<head>

Bagian head ditutup dengan:

</head>

Bagian head tidak boleh disamakan dengan menu pada halaman itu sendiri. Bagian head adalah elemen di balik layar halaman, ini dapat berisi banyak informasi tentang halaman. Informasi tersebut mencakup hal-hal seperti judul halaman, bahasa halaman (Inggris, Indonesia, dan sebagainya), jenis informasi yang disajkan, dan hal-hal lain yang umum pada halaman tersebut. Elemen deskriptif di bagian head ini terkadang disebut elemen meta. Disetiap bagian head harus selalu memiki elemen judul. Elemen judul adalah elemen yang akan muncul di title bar browser web atau sebagai judul tab di browser web, ini ditunjukkan oleh gambar dibawah ini.

Gambar 1.1 Elemen judul muncul di tab atau title bar browser web.

Bagian body

Bagian body merupakan jantung dari halaman web. Ini adalah tempat untuk menempatkan semua teks dan gambar sebuah halaman, sebut saja ini adalah isi dari web.

Bagian body dibuka dengan:

<body>

Bagian body ditutup dengan:

</body>

Sama seperti bagian head yang berisi elemen lain seperti judul dan informasi meta, bagian body juga dapat berisi beberapa elemen HTML. Misalnya, di dalam bagian body ini dapat ditemukan semua tautan dan elemen gambar bersama dengan paragraf, tabel, dan apa pun yang diperlukan untuk menampilkan halaman.

1.4 MEMBUAT HTML YANG BAIK

Halaman web yang baik disusun dalam urutan yang logis. Ini berarti elemen-elemen harus ditempatkan diberbagai urutan tertentu agar dapat dibaca dengan benar oleh browser web. Ketika aturan, ketika membuka setiap elemen maka harus menutup elemen tersbeut juga menggunakan tag tertentu yang digunakan dnegan tanda kurung siku dan garis miring..

Menggunakan elemen yang sesuai

Halaman web menggunakan beberapa elemen halaman yang juga disebut sebagai tag. Beberapa elemen tersebut disajikan pada tabel dibawah ini:

Tabel 1.1 berbagai jenis elemen dan penggunaanya

Elemen	***Deskripsi***	***Jenis penggunaan***
<a>	Anchor	Membuat link ke halaman lain atau bagian dari dokumen yang sama.
 	Jeda baris	Memasuki jeda baris atau karakter kembali.
	Bagian dari halaman	Membuat area keseluruhan atau pembagian logis pada halaman, seperti bagian heading/menu, area konten, atau footer.
<form>	Form web	Membuat formulir web untuk menerima input pengguna.
<h1> hingga <h6>	Heading	Membuat wadah untuk judul, seperti teks judul.
<hr>	Hard rule	Membuat garis horizontal.
<img>	Image	Wadah untuk gambar.
<input>	Input	Elemen untuk menerima input pengguna.
<link>	Link sumber	Tautan ke sumber daya untuk halaman tersebut; jangan bingung dengan elemen Anchor
<p>	Paragraf dalam halaman	Membuat paragraf tekstual atau area dan wadah lain untuk teks.
<script>	Tag skrip	Menunjukkan skrip web atau program. Juga sering ditemukan di bagian kepala.
	Span	Membuat wadah untuk elemen. Sering digunakan bersama dengan informasi gaya.

Markup semantik merupakan istilah dalam penggunaan berbagai jenis elemen pada waktu yang tepat, ini berkaitan dengan konsep struktur dan layout elemen. Pertimbangkan manfaat markup semantik ini:

- **Meningkatkan hasil pencarian.** Manfaat utama markup semantik adalah pengunjung dan search engine sama-sama dapat menemukan informasi yang mereka butuhkan.
- **Menyederhanakan pemeliharaan**. Manfaat sekunder dari markup semantik adalah memudahkan pemeliharaan di kemudian hari.

Ketika sebuah halaman secara semantik benar dan HTML yang valid, maka akan terbentuk dengan baik.

1.5 PERBEDAAN ANTARA ELEMEN DIV DAN SPAN

Elemen dan keduanya adalah jenis wadah, tetapi dengan beberapa kualitas yang berbeda. Secara default, elemen memiliki lebar tak terbatas (setidaknya ke tepi browser), yang dapat kita lihat dengan menerapkan batas ke salah satunya, seperti ini:

```
<div style="border:1px solid green;">Hello</div>
```

Namun, elemen hanya selebar teks yang dikandungnya. Oleh karena itu, baris HTML berikut membuat batas hanya di sekitar kata Hello, yang tidak meluas ke tepi kanan browser

```
<span style="border:1px solid green;">Hello</span>
```

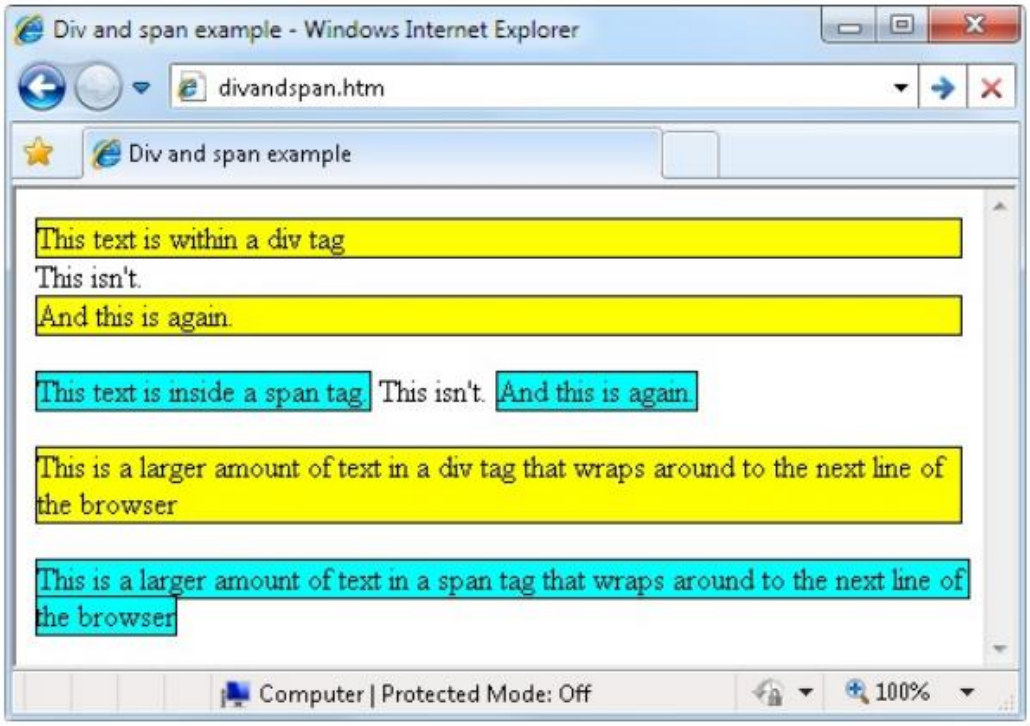
Juga, elemen mengikuti teks atau objek lain saat mereka membungkusnya, dan karena itu dapat memiliki batas yang rumit. Misalnya, pada contoh koding html dibawah, saya menggunakan CSS untuk membuat latar belakang semua elemen menjadi yellow, untuk membuat semua elemen menjadi cyan, dan untuk menambahkan batas pada keduanya, sebelum kemudian membuat beberapa contoh dan bagian.

Contoh 2 div dan span

```
<!DOCTYPE html>
<html>
<head>
<title>Div and span example</title>
<style>
div, span { border :1px solid black; }
div { background-color:yellow; }
span { background-color:cyan; }
</style>
</head>
<body>
<div>This text is within a div tag</div>
This isn't. <div>And this is again.</div><br>
<span>This text is inside a span tag.</span>
This isn't. <span>And this is again.</span><br><br>
<div>This is a larger amount of text in a div that wraps around
to the next line of the browser</div><br>
<span>This is a larger amount of text in a span that wraps around
to the next line of the browser</span>
</body>
</html>
```

Gambar 1.2 menunjukkan seperti apa contoh ini di browser web. Meskipun hanya dicetak dalam nuansa abu-abu dalam buku ini, gambar tersebut dengan jelas menunjukkan

bagaimana elemen meluas ke tepi kanan browser, dan memaksa konten berikut untuk muncul di awal posisi pertama yang tersedia di bawahnya.



Gambar 1.2 Berbagai elemen dengan lebar berbeda

Gambar tersebut juga menunjukkan bagaimana elemen menjaga dirinya sendiri dan hanya menggunakan ruang yang diperlukan untuk menyimpan kontennya, tanpa memaksa konten berikutnya muncul di bawahnya. Misalnya, di dua contoh gambar di bawah, kita juga dapat melihat bahwa ketika elemen membungkus tepi layar, elemen tersebut mempertahankan bentuk persegi panjang, sedangkan elemen hanya mengikuti alur teks (atau konten lainnya) mereka mengandung.

Catatan: Karena tag hanya bisa berbentuk persegi panjang, tag ini lebih cocok untuk memuat objek seperti gambar, kotak, kutipan, dan sebagainya, sedangkan tag paling baik digunakan untuk menyimpan teks atau atribut lain yang ditempatkan satu demi satu inline lain, dan yang harus mengalir dari kiri ke kanan (atau kanan ke kiri dalam beberapa bahasa)

Menempatkan teks pada halaman

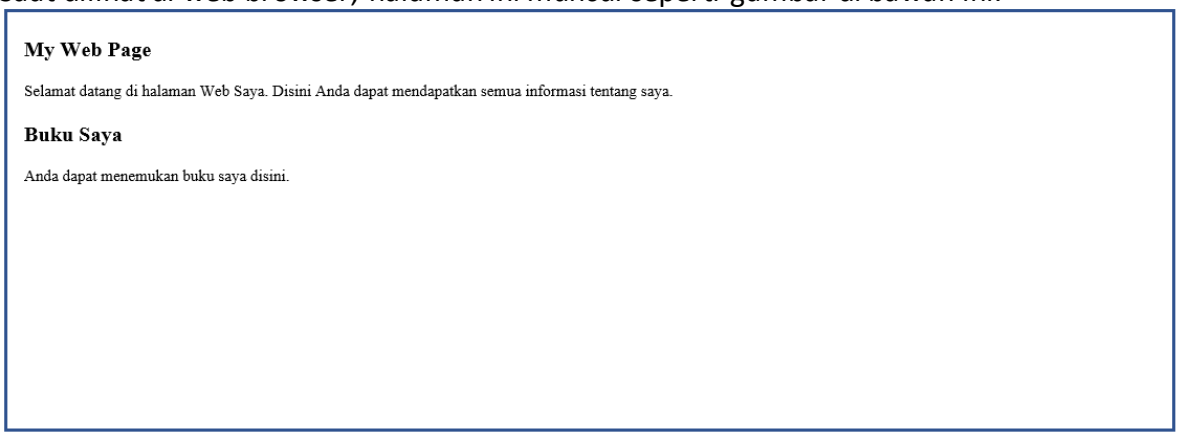
Ada banyak cara untuk menyisipkan teks ke dalam halaman web dan banyak elemen yang sesuai untuk menyimpan teks. Elemen heading seperti <h1>, <h2>, hingga <h6>, adalah tempat yang tepat untuk meletakkan heading, sedangkan <p>, , dan adalah wadah yang sesuai untuk teks bentuk yang lebih panjang, seperti paragraf. Daftar kode HTML dibawah ini menunjukkan halaman web sederhana dengan dua judul dan beberapa paragraf.

Contoh 3 Daftar Halaman Web dengan Judul dan Paragraf

```
<!doctype html>
<html>
<head>
<title>My First Document</title>
</head>
<body>
```

```
<h1>My Web Page</h1>
<p>Selamat datang dihalaman Web Saya. Disini kita dapat mendapatkan semua informasi tentang saya.</p>
<h2>Buku Saya</h2>
<p>Anda dapat menemukan buku saya disini.</p>
</body>
</html>
```

Saat dilihat di web browser, halaman ini muncul seperti gambar di bawah ini.



Gambar 1.3 Halaman web sederhana dengan judul dan paragraf.

Seperti yang kita lihat dari gambar diatas, informasi pada halaman menyertakan elemen <h1> yang diikuti dengan paragraf, elemennya adalah . Paragraf ditutup dengan elemen dan heading menggunakan elemenen <h2>. Judul keuda ditutup dengan elemen </h2>.

Buat Halaman Pertama Anda

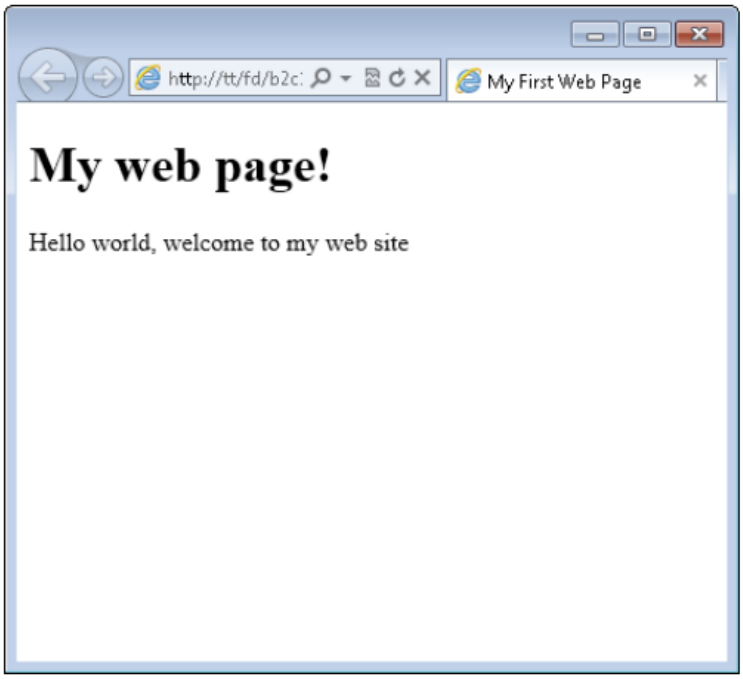
Untuk membuat HTML kita harus menggunakan text editor bukan pengolah kata seperti Microsoft Word. Microsoft Word atau program serupa seperti Pages di Mac menambahkan segala macam informasi pemformatan tambahan yang menghalangi pembuatan HTML, gunakan program seperti Notepad. Semakin sederhana bentuk HTML maka akan semakin baik. Mac menyertakan program bernama TextEdit yang dapat digunakan untuk membuat dokumen teks biasa — tetapi hati-hati: Program TextEdit akan mencoba menyimpan file dalam *Rich Text Forma*t (RTF) secara default. Saat membuat atau menyimpan file dengan **TextEdit**, pilih Usual Text dari menu **Format File drop-down**.

Untuk membuat halaman pertama Anda, ikuti langkah-langkah berikut ini:

1. Buka Text editor
2. Di text editor, masukkan HTML berikut.

```
<!doctype html>
<html>
<head>
<title>My First Web Page</title>
</head>
<body>
<h1>My web page!</h1>
<p>Hello world, welcome to my web site</p>
</body>
</html>
```

3. Simpan file dengan nama firstpage.html
   Simpan file persis seperti namanya, menggunakan huruf kecil di seluruh nama. Apache, server web yang digunakan untuk mengirim file ke browser Anda, peka huruf besar/kecil untuk nama file, jadi tetap menggunakan huruf kecil akan menghemat banyak masalah. Pastikan ekstensinya adalah .html dan bukan .txt atau ekstensi lainnya. Simpan file ke root dokumen Anda. Lokasi root dokumen tergantung pada bagaimana kita menginstal Apache dan pada jenis sistem yang kita gunakan. Jika kita menggunakan penyedia hosting, maka ini adalah titik di mana kita mengunggah file ke sistem mereka.
4. Buka browser web untuk load page
   Di browser web, arahkan ke http://localhost/firstpage.html. Setelah itu, kita akan melihat halaman seperti gambar dibawah ini.



Gambar 1.4 halaman pertama kita yang ditampilkan oleh browser

Memilih elemen level blok atau sebaris

Ketika mempertimbangkan jenis elemen mana yang akan ditambahkan ke halaman, pikirkan terlebih dahulu, apakah kita menginginkan elemen tersebut diperluas ke seluruh lebar halaman atau tidak.

- **Elemen level blok**: Elemen dan <p> keduanya dikenal sebagai elemen level blok. Elemen tingkat blok ditampilkan di seluruh lebar halaman; tidak ada yang bisa muncul di samping atau di samping elemen level blok. Pada dasarnya, anggap elemen level blok memiliki carriage return setelahnya.
- **Elemen sebaris**: Elemen tertentu, terutama elemen , dianggap sebagai elemen sebaris, yang berarti elemen lain dapat muncul di sebelahnya. Dengan kata lain, elemen sebaris tidak memiliki carriage return setelahnya

Memasukkan jeda baris dan spasi

Dalam pembuatan halaman, ada kalanya untuk menyisipkan jeda baris. Untuk melakukan tindakan ini dalam pengolah kata, cukup menekan tombol **Enter** atau **Return** pada keyboard. Hal-hal yang tidak begitu sederhana dalam HTML. Tidak peduli berapa kali tombol Enter ditekan dalam dokumen HTML, teks akan tetap ditampilkan pada baris yang sama di browser web. Perhatikan kode pada daftar dibawah ini.

Contoh 4 DaftarMencoba Memasukkan Carriage Returns ke dalam HTML

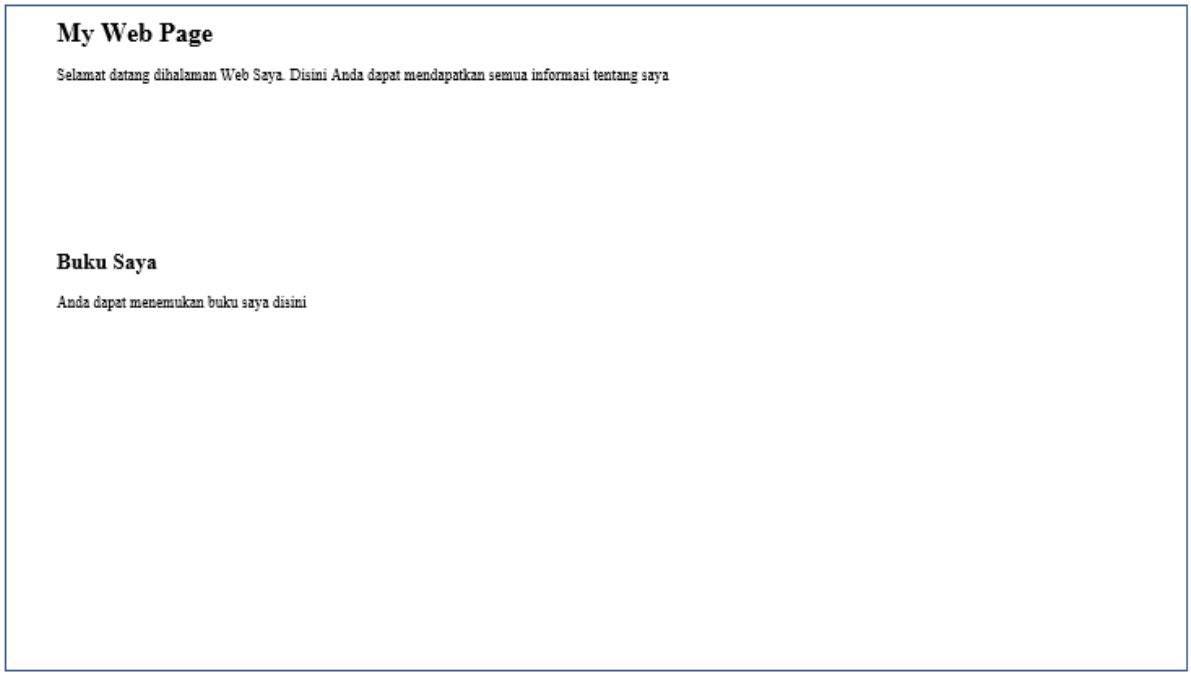
```
<!doctype html>
<html>
<head>
<title>My First Document</title>
</head>
<body>
<h1>My Web Page</h1>
<p>Selamat datang dihalaman Web Saya. Disini kita dapat mendapatkan semua informasi tentang saya.</p>


<h2>Buku Saya</h2>
<p>Anda dapat menemukan buku saya disini.</p>
</body>
</html>
```

Jika dilihat melalui web browser, outputnya sama seperti Gambar 1.5 pada bab sebelumnya. kita tidak melihat baris kosong antara paragraf pertama dan judul kedua. Hal yang sama terjadi pada spasi tambahan di HTML. Tidak peduli berapa kali spasi pada keyboard di dokumen web ditekan, maka outputnya pada html web adalah 1 spasi. Tag
 digunakan untuk menyisipkan jeda baris ke halaman web. Lihat kode daftar dibawah ini. Daripada menggunakan tombol **Enter** (atau **Return** di Mac) akan jauh lebih baik untuk menggunakan tag
 untuk menambahkan carriage return:

*Contoh 5 Menggunakan
 untuk Line Breaks*

```
<!doctype html>
<html>
<head>
<title>My First Document</title>
</head>
<body>
<h1>My Web Page</h1>
<p>Selamat datang dihalaman Web Saya. Disini kita dapat mendapatkan semua informasi tentang saya.</p>
<br>
<br>
<br>
<br>
<br>
<h2>Buku Saya</h2>
<p>Anda dapat menemukan buku saya disini.</p>
</body>
</html>
```

Saat dilihat di browser, efek yang diinginkan akan ditampilkan, seperti yang diilustrasikan pada gambar di bawah ini.



Gambar 1.5 menggunakan tag
 untuk memasukkan carriage returns.

Terkadang kita akan melihat garis miring tambahan di beberapa tag seperti
 sehingga akan ditulis sebagai
. Ini adalah peninggalan dari XHTML tetapi tidak diperlukan untuk HTML5. Kita harus melihat bahwa
 tidak memiliki mitra penutup, seperti </br>. Kita dapat menggunakan
 apa adanya, tanpa khawatir harus menutupnya. Menambahkan spasi ke HTML dilakukan dengan   entitas terkadang ditulis sebagai . Namun, ada cara yang lebih baik untuk mencapai spasi dalam HTML, terutama melalui penggunaan *Cascading Style Sheets* (CSS). Oleh karena itu, penggunaan   entitas tidak akan dibahas demi metode yang lebih umum dan didukung lebih luas melalui CSS.

Membuat dokumen lebih mudah dirawat

Developer sering menggunakan comment untuk mencatat informasi di balik layar tentang halaman atau tentang kode mereka, dan comment tidak ditampilkan di halaman web. Misalnya, komentar di halaman web mungkin seperti "Saya menambahkan ini pada 19/10/2012" atau "Ditambahkan untuk mendukung inisiatif penjualan kami." Jika kita mengunjungi halaman web, kita dapat melihat komentar tersebut hanya dengan melihat file HTML halaman tersebut.

Comment HTML dibuka dengan sintaks ini:

<!--

Comment HTML ditutup dengan sintaks ini:

-->

Segala sesuatu yang muncul dari awal <!-- hingga bagian pertama --> dianggap sebagai bagian dari komentar. Daftar dibawah ini berisi contoh dokumen HTML dengan komentar.

Contoh 6 Menambahkan Komentar HTML

```
<!doctype html>
<html>
```

```
<head>
<title>My First Document</title>
</head>
<body>
<h1>My Web Page</h1>
<p>Selamat datang dihalaman Web Saya. Disini kita dapat mendapatkan semua informasi tentang saya.</p>
<!—Menambahkan informasi tentang bukuk saya 10/1/2022 -->
<h2>Buku Saya</h2>
<p>Anda dapat menemukan buku saya disini.</p>
</body>
</html>
```

Contoh 6a Multi-line comment

```
<!doctype html>
<html>
<head>
<title>My First Document</title>
</head>
<body>
<h1>My Web Page</h1>
<p>Selamat datang dihalaman Web Saya. Disini kita dapat mendapatkan semua informasi tentang saya.</p>
<!—
Menambahkan informasi tentang bukuk saya
10/1/2012
-->
<h2>Buku Saya</h2>
<p>Anda dapat menemukan buku saya disini.</p>
</body>
</html>
```

Dalam komentar ini, kita dapat melihat bahwa teks sebenarnya dari komentar tersebut diindentasi, yang memunculkan poin penting lainnya: Menggunakan lekukan saat membuat dokumen akan sangat membantu. Dokumen lebih mudah dibaca dan dipelihara nanti ketika elemen diindentasi, sehingga kita dapat melihat dengan jelas secara visual elemen mana yang "di dalam" elemen lainnya.

Menambahkan daftar dan tabel

Daftar dan tabel membantu untuk mewakili jenis informasi tertentu. Misalnya, daftar pohon di halaman kampus paling baik diwakili dengan daftar seperti ini:

Pinus
Beringin
Palm

Tetapi jika ingin memasukkan lebih banyak informasi tentang pohon, tabel adalah format yang lebih baik:

Jenis Pohon	Deskripsi
Pinus	Pohon umum di lapangan
Beringin	Ada beberapa pohon beringin halaman kampus
Palm	Ada 5 pohon palm di halaman kampus dekat gazebo

HTML memiliki tag untuk membuat daftar dan tabel. Tabel 1.2 menjelaskan berbagai elemen tersebut.

Tabel 1.2 Daftar Umum dan Elemen Tabel dalam HTML

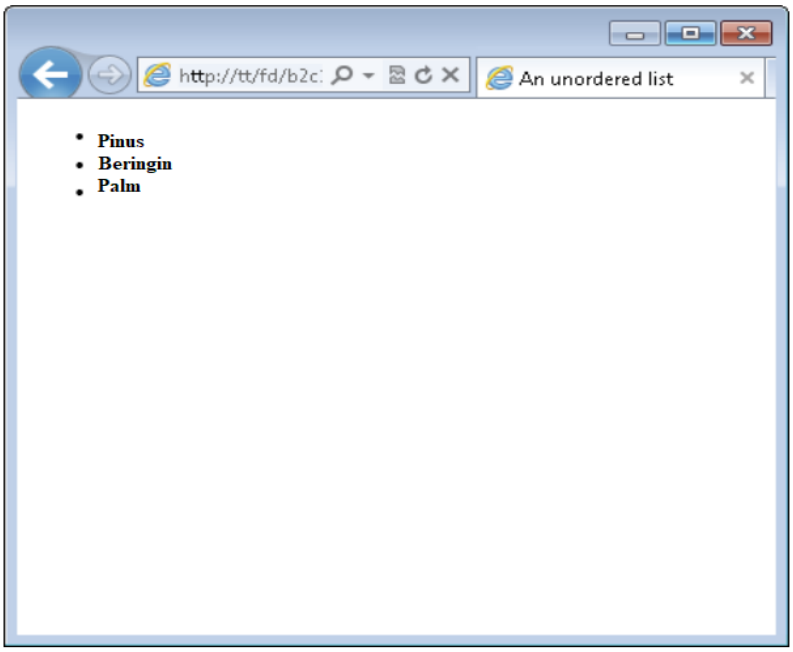
Eleemn	***Jenis***	***Deskripsi***
	List Item	Digunakan bersama dengan atau untuk membuat daftar informasi.
	Order List	Daftar informasi yang diurutkan, digunakan bersama dengan .
<table>	Tabel	Digunakan dengan <tr>, <td>, dan elemen lain untuk membuat tabel untuk menyajikan informasi.
<td>	Sel Tabel	Membuat sel di baris tabel
<th>	Header Tabel	Sel tabel yang merupakan heading.
<tr>	Row Tabel	Membuat deretan tabel.
	Unorder List	Terkait dengan dan untuk membuat daftar informasi

Saat membuat daftar, kita memiliki dua pilihan jenis daftar yang akan dibuat: daftar yang diurutkan atau daftar yang tidak diurutkan. Daftar berurut digunakan untuk hal-hal seperti membuat garis besar, sementara daftar yang tidak berurutan membuat hampir semua jenis daftar lainnya. Daftar berikut menunjukkan HTML yang digunakan untuk membuat daftar *unordered* standar.

Contoh 7 Membuat Daftar Unordered

```
<!doctype html>
<html>
<head>
<title>An unordered list</title>
</head>
<body>
<ul>
 <li>Pine</li>
 <li>Oak</li>
 <li>Elm</li>
</ul>
</body>
</html>
```

Jika dilihat di browser, HTML ini menghasilkan halaman seperti pada gambar dibawah ini.

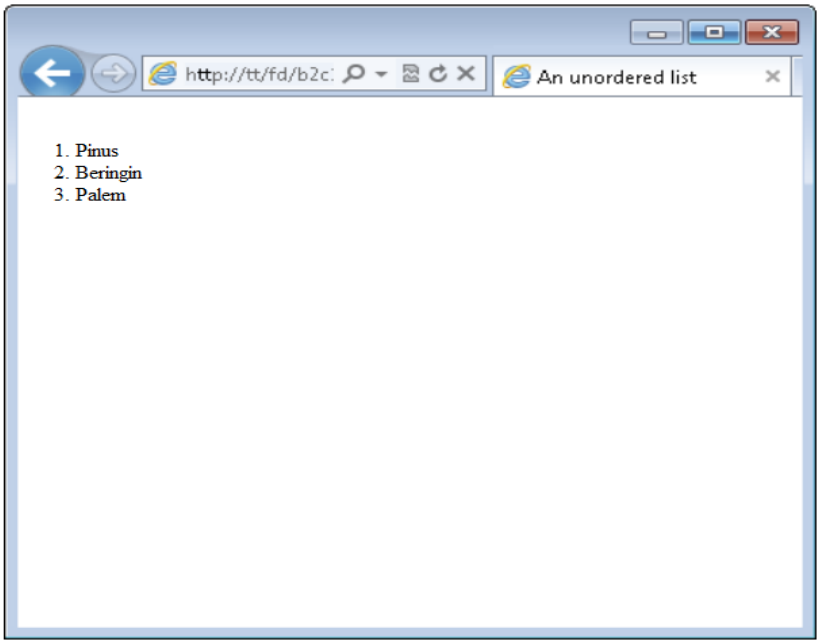


Gambar 1.6 Daftar tidak berurutan

Daftar tidak berurutan menggunakan gaya default untuk daftar, yang menambahkan poin di samping setiap item. Gaya peluru ini dapat diubah, atau bisa juga tidak di ikut sertakan dalam CSS. Membuat daftar terurut berarti cukup mengubah elemen <ul> menjadi <ol>. Prosesnya akan terlihat seperti ini:

```
<ol>
 <li>Pine</li>
 <li>Oak</li>
 <li>Elm</li>
</ol>
```

Saat dilihat di browser, poin dari contoh sebelumnya diganti dengan angka, seperti pada gambar dibawah ini.



Gambar 1.7 Daftar yang dipesan

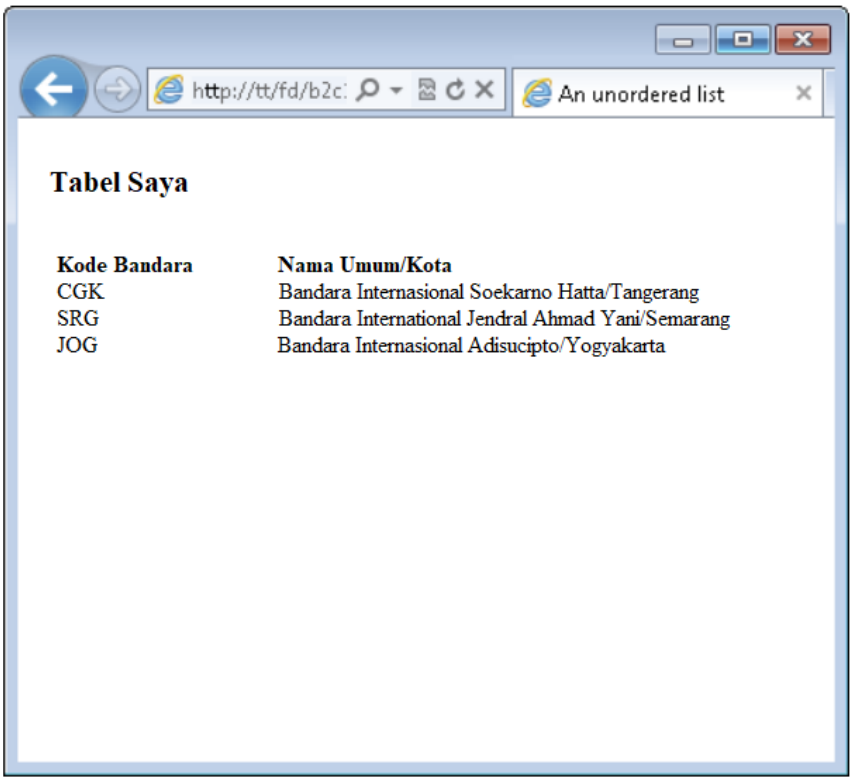
1.6 BERLATIH MEMBUAT TABEL

Untuk membuat tabel ikuti langkah berikut ini:

1. Buka text editor
2. Buat new text document pada text editor.
   Sebagian besar text editor akan terbuka dengan dokumen kosong. Jika terdapat sesuatu dalam dokumen, maka hapus terlebih dahulu sebelum melanjutkan.
3. Masukkan HTML berikut:

```
<!doctype html>
<html>
<head>
<title>My First Web Page</title>
</head>
<body>
<h1>My Table</h1>
<table>
        <tr>
        <th>Airport Code</th>
        <th>Common Name/City</th>
        </tr>
<tr>
        <td>CWA</td>
        <td>Central Semarang Airport</td>
        </tr>
        <tr>
        <td>ORD</td>
        <td>Chicago O'Hare</td>
        </tr>
        <tr>
        <td>LHR</td>
        <td>London Heathrow</td>
        </tr>
</table>
</body>
</html>
```

4. Simpan file sebagai table.html.
   Simpan file dengan ekstensi .html. File harus disimpan di root dokumen.
5. Lihat file di browser Anda.
   Buka browser web kita dan ketik http://localhost/table.html ke dalam bilah alamat. Prosesnya akan terlihat seperti gambar dibawah ini.



Gambar 1.8 Tabel latihan.

Perhatikan bahwa tabel tidak memiliki batas di sekitarnya. Jika ingin menambahkan batas, terus lakukan latihan ini. Jika tidak, lanjutkan ke bagian berikutnya.

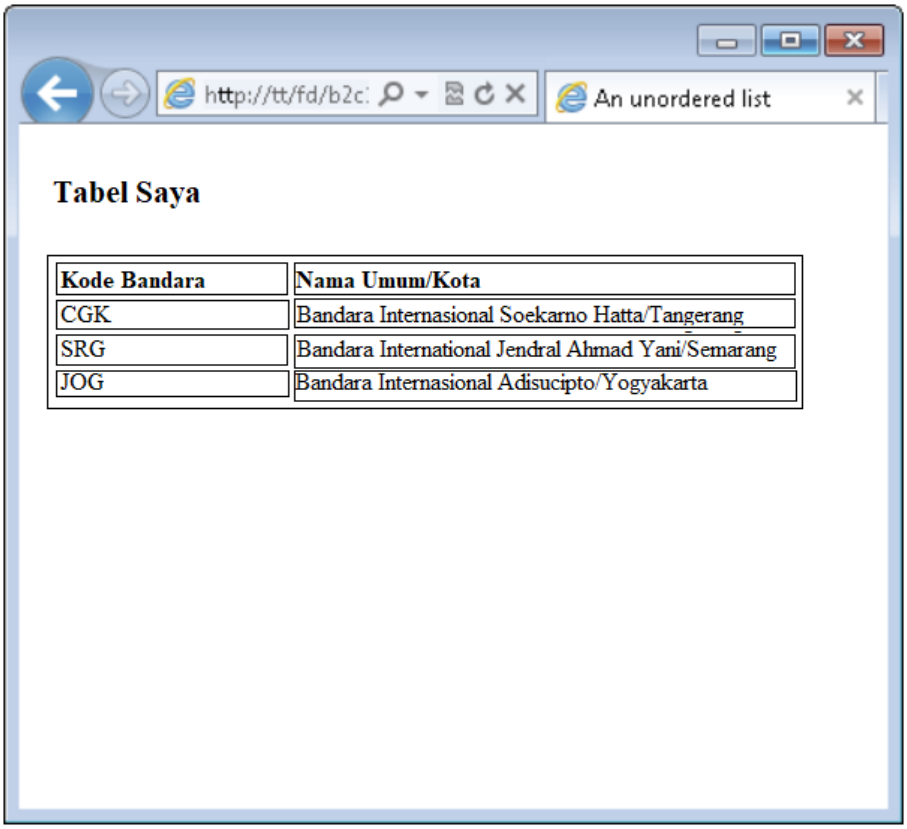
6. Buka table.html di text editor.
   Jika kita menutup text editor, buka lagi dan muat table.html.
7. Ubah kode di table.html menjadi berikut:

```
<!doctype html>
<html>
<head>
<title>My First Web Page</title>
</head>
<body>
<h1>My Table</h1>
<table border=”1”>
        <tr>
                <th>Airport Code</th>
                <th>Common Name/City</th>
        </tr>
        <tr>
                <td>CGK</td>
                <td>Bandara International Soekarno-Hatta/Tangerang</td>
        </tr>
        <tr>
                <td>SRG</td>
                <td>Bandara Internasional Jendral Ahmad Yani/Semarang</td>
        </tr>
        <tr>
```

```
        <td>JOG</td>
        <td>Bandara Internasional Adi Sucipto/Yogyakarta</td>
        </tr>
</table>
</body>
</html>
```

Perhatikan bahwa satu-satunya perubahan adalah menambahkan spasi dan kemudian border="1" di dalam elemen <table>.

8. Refresh table.html di browser Anda.
   Jika kita menutup browser, buka kembali dan buka http://localhost/table.html. Jika browser kita masih terbuka, tekan Ctrl+R untuk merefresh halaman (Command+R di Mac). Selanjutnya, hasil dari tabel adalah sebagai berikut.



Gambar 1.9 Tabel dengan batas di sekitar setiap sel

Saat menambahkan border="1" ke elemen <table>, ini disebut atribut. Atribut membantu untuk mendeskripsikan atau mendefinisikan elemen lebih lanjut atau memberikan detail tambahan tentang bagaimana elemen tersebut harus berperilaku.

1.7 MENYERTAKAN TAUTAN DAN GAMBAR DI HALAMAN WEB ANDA

Apa jadinya web tanpa tautan — dan juga gambar? Tidak banyak web sama sekali. Tautan adalah item yang kita klik di dalam halaman web untuk terhubung atau memuat halaman lain, dan ketika kita berbicara tentang gambar, yang kita maksud adalah ilustrasi dan foto. Bagian ini membahas cara menambahkan tautan dan juga gambar ke halaman web Anda

Menambahkan tautan

Tautan ditambahkan dengan <a>, atau elemen anchor. Atribut href memberi tahu elemen anchor tujuan tautan. Tautan itu sendiri dapat ditambahkan ke apa saja di halaman.

Misalnya, kita dapat menautkan setiap pohon yang disebutkan di bagian sebelumnya ke artikel tentang masing-masing jenis pohon tersebut. Ketika sesuatu ditautkan, browser akan memberikan umpan balik visual bahwa ada tautan dengan menyorot dan menggarisbawahi area yang ditautkan. Seperti elemen HTML lainnya, elemen <a> memiliki tag penutup yang sesuai yang digunakan untuk memberi tahu browser kapan harus berhenti menyorot dan menggarisbawahi tautan.

Menautkan ke halaman lain

Menautkan ke halaman lain, baik di situs yang sama atau di situs yang berbeda, dilakukan dengan cara yang sama. Sebagai contoh, lihat HTML berikut:

Here's a link to web universitas stekom

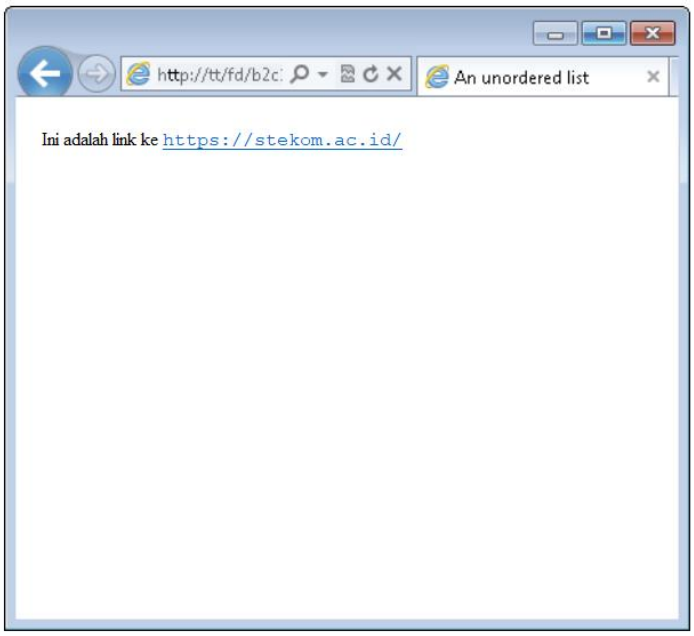
Baris ini menggunakan elemen paragraf untuk membuat kalimat, "Ini tautan ke web universitas stekom." Ini adalah web, kita memutuskan untuk benar-benar memberikan tautan sehingga pengunjung dapat mengklik kata-kata tertentu dan dibawa ke halaman itu. kita melakukannya dengan elemen <a> bersama dengan atribut href. Dalam hal ini, elemen <a> terlihat seperti ini:

Atribut href menunjuk ke URL http://www.braingia.org dan diapit oleh tanda kutip. Teks yang akan disorot kemudian muncul, diikuti dengan tag penutup . Berikut adalah latihan untuk menerapkan tautan ini.

1. Buka text editor.
2. Tempatkan HTML berikut pada text editor Anda:

```
<!doctype html>
<html>
<head>
<title>Link</title>
</head>
<body>
<p>Here's a link to <a href="http://www.braingia.
org">Steve Suehring's site</a></p>
</body>
</html>
```

3. Simpan file sebagai link.html.
   File harus disimpan ke root dokumen kita dengan nama link.html.
4. Buka browser kita dan lihat halamannya.
   Buka browser web kita dan arahkan ke http://localhost/link.html dengan memasukkan URL tersebut ke bilah alamat. kita akan melihat halaman seperti itu pada gambar dibawah ini.



Gambar 1.10 Halaman dengan tautan

Selalu tutup elemen <a> dengan tag penutup yang sesuai. Kesalahan yang sering terjadi adalah membiarkan elemen <a> terbuka, sehingga semua teks berikutnya disorot sebagai tautan. Contoh dan latihan menunjukkan cara menautkan ke halaman di situs web yang berbeda. Membuat tautan ke halaman di situs yang sama dilakukan dengan cara yang sama, tetapi alih-alih menyertakan skema *Uniform Resource Identifier* (URI) dan nama host (bagian http://www.stekom.ac.id dari contoh ini), kita hanya dapat menautkan ke halaman itu sendiri. Jika kita telah mengikuti latihan sebelumnya, maka kita harus memiliki halaman bernama table.html. Berikut HTML untuk membuat tautan ke table.html. HTML latihan sebelumnya disertakan sehingga kita dapat melihat keseluruhan konteks untuk tautan:

```
<!doctype html>
<html>
<head>
<title>Link</title>
</head>
<body>
<p>link ke <a href=”table.html”>contoh tabel</a></p>
</body>
</html>
```

Seperti sebelumnya, tautan terdapat di dalam elemen <p> tetapi perhatikan bahwa atribut href sekarang hanya menunjuk ke table.html.

Memahami tautan absolut versus relatif

Tautan yang ditunjukkan pada contoh sebelumnya disebut tautan relatif karena tidak dimulai dengan skema *Uniform Resource Identifier* (URI) (http://) atau garis miring awal (/). Tautan relatif mengasumsikan bahwa target (table.html dalam contoh) berada di direktori atau folder yang sama dengan dokumen atau halaman yang ditautkan. Dalam kasus contoh, tautan relatif berfungsi karena halaman saat ini, link.html, dan halaman yang ditautkan, table.html, keduanya ada di root dokumen.

Jika kedua halaman tidak berada di direktori yang sama (dengan kata lain, jika table.html berada di folder bernama tabel di root dokumen dan file link.html ada di folder

bernama link di root dokumen), maka kita perlu untuk membuat tautan absolut. Tautan absolut memberi tahu server di mana tepatnya mencari untuk menemukan target. Misalnya, tautan absolut mungkin terlihat seperti /tables/table.html. Tautan ini memberi tahu server bahwa ia perlu mulai mencari di root dokumennya untuk direktori yang disebut tabel dan kemudian harus menemukan file bernama tables.html di direktori tabel.

Gunakan tautan absolut ketika kita perlu memberikan referensi yang tepat atau absolut ke target yang ditautkan. Gunakan tautan relatif ketika sumber daya yang ditautkan akan selalu ditemukan di tempat yang sama dengan halaman yang menautkannya. Jika lokasi halaman atau target berubah, maka tautan relatif akan berhenti berfungsi.

Menautkan dalam satu halaman

Terkadang kita ingin menautkan dalam halaman yang sama. kita dapat melakukan ini pada halaman yang sangat panjang, di mana kita memiliki daftar isi di bagian atas dan kemudian artikel lengkap di bagian bawah halaman. Membuat tautan dalam halaman menggunakan elemen <a> yang sama dengan yang kita lihat, kali ini dengan atribut nama. Daftar dibawah menunjukkan HTML untuk membuat anchor dalam halaman.

Contoh 8 Anchor Dalam Halaman

```
<!doctype html>
<html>
<head>
<title>Link</title>
</head>
<body>
<ul>
 <li><a href=”#pine”>Pine</a></li>
 <li><a href=”#oak”>Oak</a></li>
 <li><a href=”#elm”>Elm</a></li>
</ul>
<p><a name=”pine”> Ada pohon pinus di halaman.</a><p>
<p><a name=”oak”> Ada beberapa pohon beringin halaman kampus.</a><p>
<p><a name=”elm”> Ada 5 pohon palm di halaman kampus dekat gazebo.</a><p>
</body>
</html>
```

Pada daftar kode diatas, tag href yang ditambahkan ke setiap item daftar menggunakan tanda pound atau hash (#). Ini adalah kunci yang digunakan untuk memberi tahu browser bahwa sumber daya akan ditemukan di halaman yang sama. Kemudian di HTML kita melihat elemen <a>lainnya, kali ini menggunakan atribut name. Atribut name itu sesuai dengan masing-masing atribut href dari halaman sebelumnya. Jadi, tidak ada lagi yang perlu ditambahkan untuk menambahkan tautan dalam halaman. Hanya perlu menggunakan tanda pound untuk menunjukkan bahwa sumber daya ditemukan kemudian di halaman dan kemudian menggunakan atribut name untuk membuat elemen lain cocok dengan itu.

Membuka tautan di jendela baru

Terkadang kita ingin membuat tautan terbuka di tab baru atau jendela baru. Saat pengunjung mengeklik tautan yang ditentukan sedemikian rupa, browser akan membuka tab baru dan memuat sumber daya tertaut di tab baru itu. Situs yang ada akan tetap terbuka di browser pengunjung juga. Jangan membuat setiap tautan terbuka di jendela baru. kita harus melakukannya hanya jika masuk akal, seperti yang mungkin terjadi ketika pengunjung berada di tengah-tengah proses yang panjang di situs web kita dan perlu menautkan ke referensi

sumber daya atau situs lain, seperti direktori kode ZIP atau persyaratan perjanjian layanan. Juga, apakah tautan terbuka di tab baru atau jendela baru tergantung pada browser; kita tidak dapat mengontrolnya.

Ini dapat dilakukan dengan menambahkan atribut target ke elemen <a> kita dengan nilai khusus, _blank. Misalnya, contoh sebelumnya menunjukkan cara membuat tautan ke situs web universitas stekom, www.stekom.ac.id. Ingatlah bahwa tautannya terlihat seperti ini:

web universitas stekom

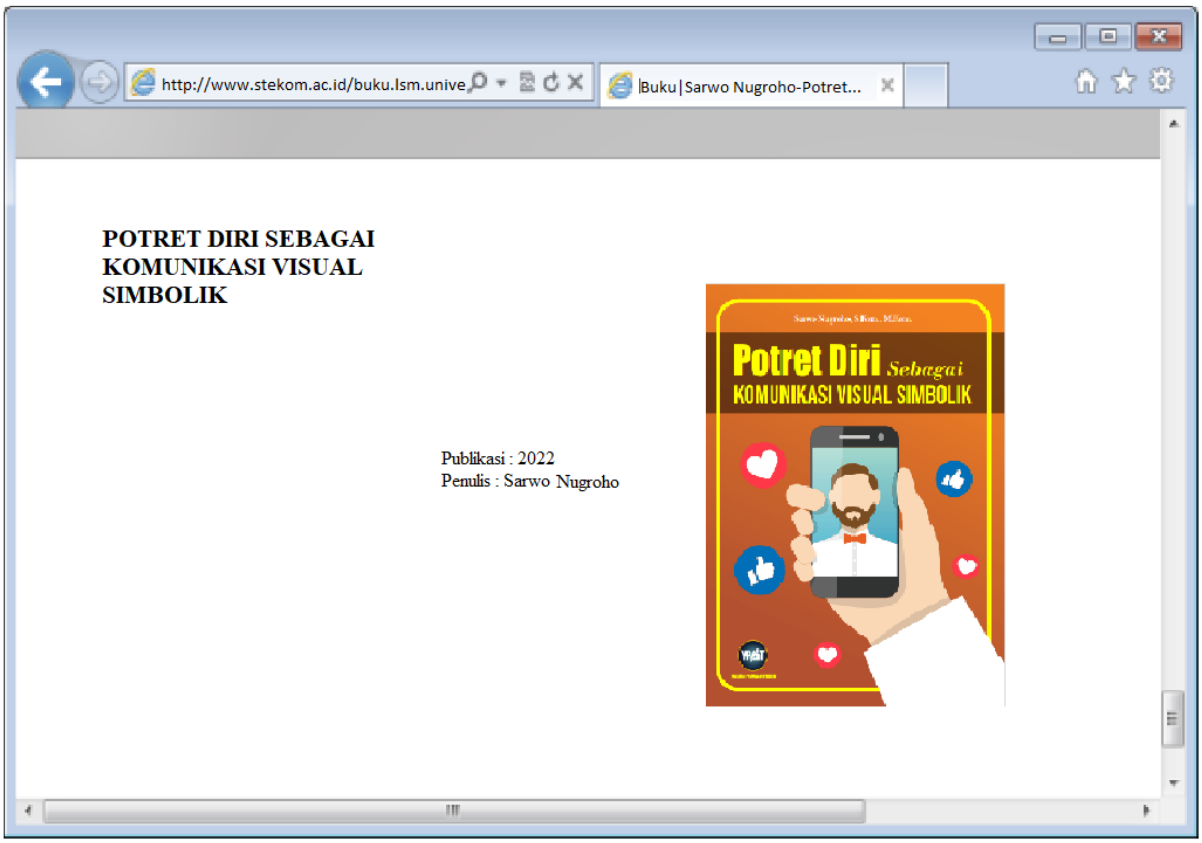
Untuk membuat tautan ini terbuka di jendela baru, kita menambahkan target="_blank" atribut/nilai pasangan ke elemen, sehingga terlihat seperti ini:

web Universitas stekom

Anda dapat mencobanya dengan membuka file link.html dari latihan sebelumnya dan menambahkan target="_blank" seperti yang ditunjukkan. Perhatikan penggunaan garis bawah sebelum kata kosong. Saat kita menyimpan file dan memuat ulang halaman itu (Ctrl+R atau Command+R), tautannya tidak akan terlihat berbeda. Namun, mengklik tautan akan membuka tab baru (atau jendela baru, tergantung pada browser dan konfigurasi Anda).

Menambahkan gambar

Gambar, seperti foto atau grafik, meningkatkan daya tarik visual halaman web. Gambar biasanya disematkan di halaman, seperti yang ditunjukkan pada contoh gambar dibawah ini, di mana foto sampul buku Sarwo Nugroho, ditampilkan.



Gambar 1.11 pada halaman web.

Dalam penyematan gambar, ada hal yang perlu di perhatikan ketika menyertakan gambar. Kita dapat menyematkan gambar appaun dalam website milik kita, dengan sayarat itu adalah hak milik sendiri, atau jka gambar tersebut memiliki hak cipta maka kita harus mendapatkan izin dari pencipta gambar tersebut sebelum menguploadnya ke website kita. Dalam pembuatan website, gambar biasanya dijadikan sebegai pelengkap, atau bahkan objek utama, ada juga yang digunakan sebagai latar belakang, ini tergantung pada kebutuhan kita dalam membuat website.

Referensi lokasi gambar

Gambar ditambahkan dengan elemen <img>. Sama seperti elemen <a>, elemen <img> menggunakan atribut untuk memberi tahu browser lebih banyak informasi tentang dirinya sendiri. Atribut src digunakan untuk memberi tahu browser di mana menemukan gambar. Sebelumnya, pada gambar dibawah atas. HTML membawa gambar tersebut ke halaman website dengan bentuk coding seperti ini:

```
<img src=”images/books_sarwonugroho.gif”>
```

Seperti yang kita lihat, elemen <img> menambahkan atribut src, yang kemudian merujuk ke lokasi gambar di server web. Elemen <img> tidak memiliki tag penutup </img> karena elemen ini tidak memiliki kontennya sendiri, berbeda dengan elemen <p>dan <a> — yang keduanya membutuhkan konten untuk masuk ke dalamnya dan oleh karena itu harus ditutup. Terkadang kita mungkin melihat elemen seperti <img> ditutup dengan /> alih-alih hanya >, seperti pada contoh. Keduanya adalah cara yang dapat diterima dan valid untuk menutup elemen jenis ini.

Elemen <img> harus selalu memiliki atribut alt. Atribut alt memberi tahu seach engine dan teknologi bantu tentang gambar yang digunakan. Saat digunakan dengan elemen <img>, atribut alt terlihat seperti ini:

```
<img src=” books_sarwonugroho” alt=”potret diri”>
```

Anda harus menggunakan deskripsi singkat sebagai isi dari atribut alt. Menggunakan sesuatu seperti "MySQL Bible adalah buku yang bagus dan semua orang harus membelinya" tidak menggambarkan gambarnya, tetapi "MySQL Bible" menggambarkannya.

Memilih gambar web yang bagus

Ketika memilih gambar untuk website, Kita perlu memastikan bahwa foto tidak blur saat foto tersebut diambil. Kita juga harus mempertimbangkan ukuran dari foto dan format foto. Web browser dapat melihat gambar berbagai format termasuk JPG, GIF, PNG dan beberapa lainnya. Ukuran file merupakan salah satu aspek terpenting untuk dipertimbangkan ketika memilih gambar. Saat menyertakan gambar dengan ukuran besar, seperti yang diambil pada pengaturan raw yang berukuran besat dengan kamera digital Anda, pengunjung harus mengunduh file tersebut terlebih dahulu sebelum dapat melihat gambar tersebut, dan tentunya ini lumayan memakan waktu, tergantung pada kecepatan koneksi pengunjung.

Untuk menyiasati hal ini, ketika menggunggah gambar kita harus mengubah ukuran gambar gambar terlebih dahulu agar gambar hanya berukuran kurang dari 100 KB. Aspek penting lainnya yang perlu dipertimbangkan adalah jumlah semua gambar pada halaman. Misalnya, dalam website kita membutuhkan sekitar 10 atau 15 gambar, jika ukurannya masing-masing adalah 100kb maka total semuanya adalah 1.5mb dan ini masih lumayan berat, saran saya akan lebih baik jika gambar yang akan digunakan dalam website memiki ukuran lebih kecil dari 100kb agar pemuatan gambar pada web yang diakses pengunjung lebih cepat.

Ketika memilih gambar, Kita juga perlu menyamakan semua format gambar agar lebih mudah diakses oleh pengunjung. Setidaknya, pastikan agar para pengunjung web kita tak perlu berpindah menggunakan aplikasi lainnya hanya untuk melihat gambar tersebut. Kenyamanan penjgunjung website kita sangat perlu untuk difikirkan. Ingatlah jumlah semua gambar saat menentukan ukuran gambar untuk halaman halaman web kita agar halaman yang bergambar tersebut lebih cepat diunduh oleh pengunjung.

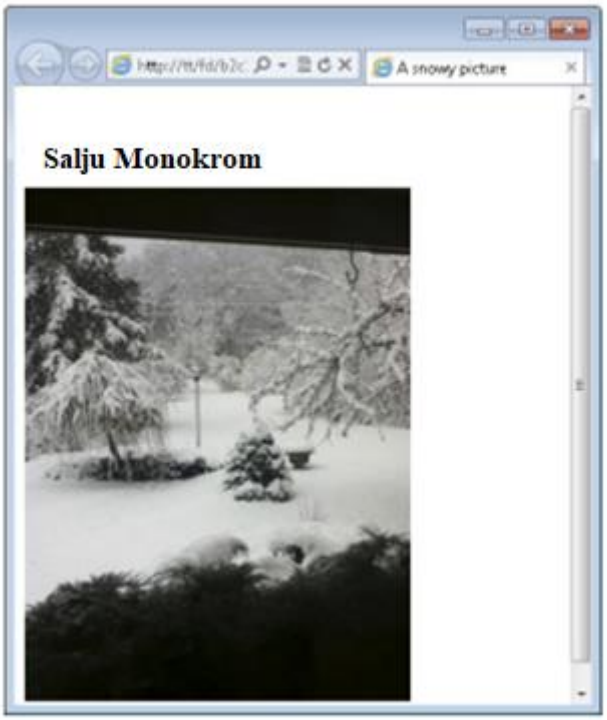
Membuat halaman dengan gambar

Saatnya membuat halaman dengan gambar sehingga kita dapat melihat bagaimana dan di mana gambar cocok dengan keseluruhan halaman HTML yang lebih besar. Ikuti langkah ini.

1. Buka Text editor
   Lihat pembahasan sebelumnya tentang text editor.
2. Masukkan HTML berikut:

   ```
   <!doctype html>
   <html>
   <head>
   <title>Gambar matahari terbit diakhir bulan agustus</title>
   </head>
   <body>
   <h1>SALJU MONOKROM</h1>
   <p><img src=”IMG01.jpg” alt=”Akhir DESEMBER 2022”></p>
   </body>
   </html>
   ```

   Saat kita membuat HTML ini, kita perlu menggunakan foto atau gambar lain milik kita sendiri atau kita dapat menggunakan file IMG01.jpg. Terlepas dari gambar yang kita pilih, kita perlu menempatkan file di root dokumen server web. Pastikan bahwa huruf besar dan kecil untuk nama file cocok dengan apa yang kita masukkan ke dalam atribut src. Dengan kata lain, jika gambar kita bernama Akhir DESEMBER, harus beratribut src harus “Akhir DESEMBER”.
3. Simpan file sebagai image.html.
   Simpan file persis seperti namanya, menggunakan huruf kecil di seluruh nama. File harus disimpan ke root dokumen Anda. Lokasi root dokumen tergantung pada bagaimana kita menginstal Apache dan pada jenis sistem yang kita gunakan. Jika kita menggunakan hostinger, maka kita hanya perlu mengunggah file ke sistem hosting tersebut.
4. Buka browser web kita untuk memuat halaman.
   Di browser web, arahkan ke http://localhost/image.html.

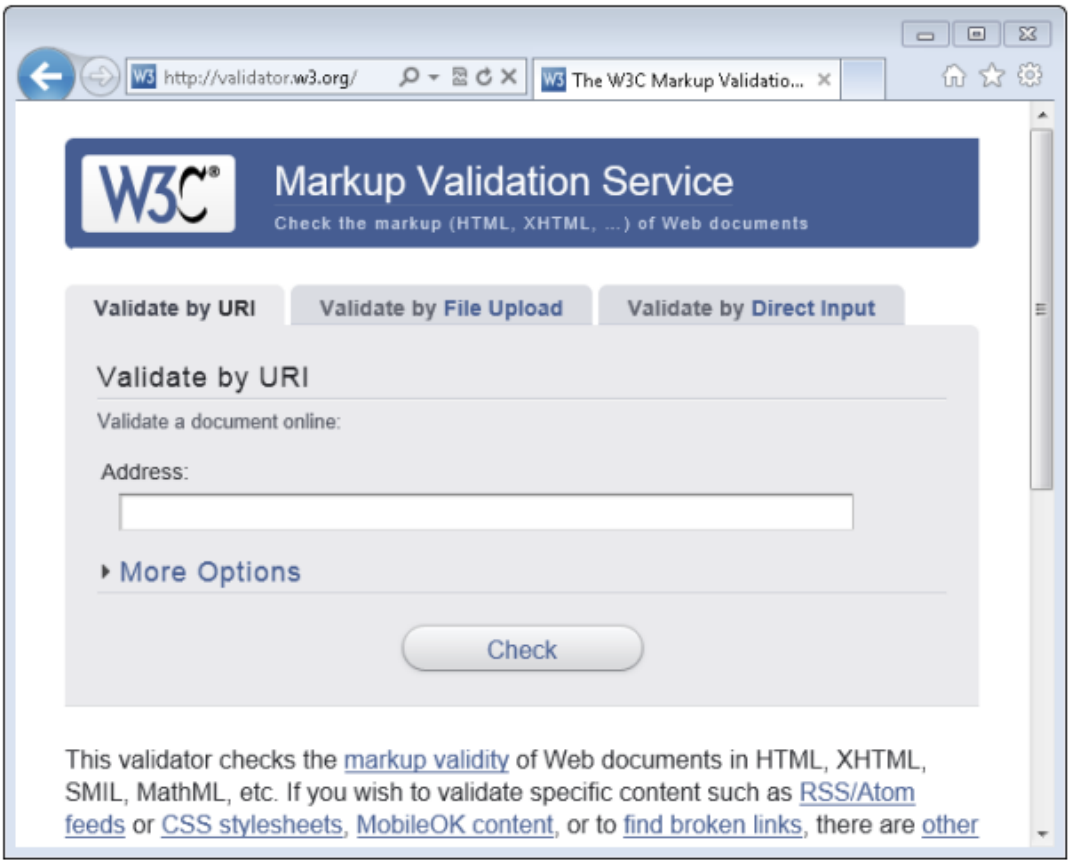


Gambar 1.12 Menambahkan gambar ke halaman.

HTML ini menggunakan elemen <img> untuk memuat foto bernama IMG01.jpg dari direktori saat ini. Dengan kata lain, IMG01.jpg berada di direktori yang sama dengan halaman image.html di server web. Hindari spasi dalam nama file gambar. Ingat juga bahwa URL, file, dan gambar peka huruf besar/kecil.

1.8 MENULIS HTML VALID

Saat kita membuat halaman web dengan HTML, ada aturan tertentu yang harus diikuti untuk memastikan bahwa browser web dapat membaca dan menampilkan halaman dengan benar. Versi saat ini dari spesifikasi HTML adalah HTML versi 5, yang hanya dikenal sebagai HTML5. Proses memvalidasi halaman berarti bahwa situs web khusus memeriksa kode HTML yang kita tulis dan bandingkan dengan spesifikasi untuk versi HTML tersebut. Situs web yang digunakan untuk memvalidasi HTML disebut *W3C Markup Validation Service* (sering disebut Validator W3C) dan dioperasikan oleh *World Wide Web Consortium* (W3C). Validator W3C dapat ditemukan di http://validator.w3.org dan gratis untuk digunakan.



Gambar 1.13 W3C

Layanan Validasi Markup, terkadang hanya disebut Validator.
Validasi HTML kita dengan salah satu dari tiga cara:

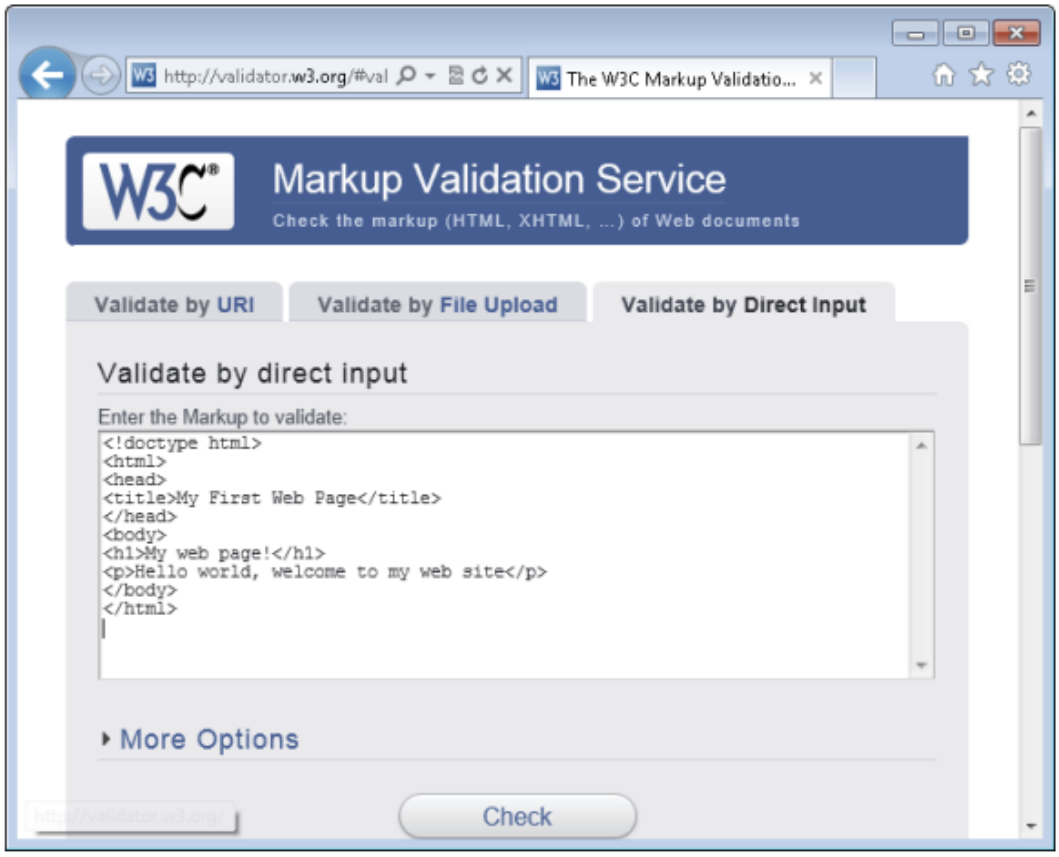
- **Menyediakan URL:** kita dapat memasukkan URL ke Validator dan secara otomatis akan mengambil HTML di URL tersebut dan mencoba memvalidasinya. Agar Validator dapat mengambil HTML kita menggunakan metode ini, halaman harus tersedia untuk umum. Ini biasanya tidak terjadi ketika kita menginstal server web di komputer Anda. Jika kita menggunakan hostinger eksternal, maka situs dan halaman kita mungkin tersedia di Internet. Dalam hal ini, kita dapat memasukkan URL di kotak alamat "Validate by URI".
- **Mengunggah file**: kita dapat mengunggah file menggunakan opsi "Validate by File Upload". Dengan menggunakan metode ini, kita memilih file di komputer Anda. File itu kemudian diunggah ke Validator.
- **Menempelkan HTML ke dalam Validator:** Ini berarti menyalin HTML dari text editor dan menempelkannya ke tab "Validate by Direct Input" di Validator. Opsi ini biasanya merupakan metode tercepat dan termudah dan itulah yang kami tunjukkan di bagian ini.

Validasi HTML

Jika kita telah mengikuti latihan di bab ini, maka kita telah membuat beberapa HTML. Latihan selanjutnya menggunakan W3C Validator untuk memastikan bahwa HTML yang kita tulis valid sesuai dengan spesifikasi HTML5. Ikuti langkah ini:

1. Buka firstpage.html menggunakan text editor
   Halaman ini adalah halaman pertama yang kita buat di bab ini. Namun, jika kita melewatkan latihan itu, buka salah satu file HTML yang kita buat di bab ini.
2. Sorot/pilih semua HTML di file yang terbuka.
   Gunakan mouse atau perangkat penunjuk untuk menyorot semua HTML atau tekan Ctrl+A di Windows atau Command+A di Mac.

3. Salin HTML ke clipboard Anda.
   Pilih Salin (ditemukan di menu Edit di sebagian besar text editor) atau tekan Ctrl+C di Windows atau Command+C di Mac untuk menyalin HTML yang disorot ke clipboard.
4. Buka browser web kita dan navigasikan ke W3C Validator.
   Dengan browser terbuka, ketik http://validator.w3.org di bilah alamat atau lokasi di browser dan tekan Enter untuk membuka Validator.
5. Pilih Validate Direct Input
   Tab Validasi dengan Input Langsung akan digunakan untuk menempelkan kode di papan klip Anda.
6. Tempelkan HTML ke dalam Validator.
   Tekan Ctrl+V di Windows atau Command+V di Mac untuk menempelkan HTML dari clipboard ke kotak Enter the Markup to Validate di halaman Validator. Jika kita menggunakan HTML dari firstpage.html, layar kita akan terlihat seperti pada Gambar dibawah ini.



Gambar 1.14 Menempelkan HTML ke dalam Validator

7. Klik Check
   Klik tombol Check pada halaman Validator untuk menjalankan validasi HTML Anda

Gambar 1.15 Dokumen HTML yang valid melewati Validator.

Perhatikan tiga peringatan dalam validasi ini. Menggulir ke bawah mengungkapkan bahwa salah satu peringatannya adalah bahwa validator HTML5 sebenarnya eksperimental saat ini, meskipun itu dapat berubah pada saat kita membaca ini. Dua peringatan lainnya terkait dengan pengaturan bahasa. Sebaiknya sertakan pengkodean karakter, yang membantu browser menentukan cara membaca dokumen, termasuk bahasa apa yang digunakan untuk HTML dan halaman. Lihat http://www.w3.org/International/tutorials/tutorial-char-enc/#Slide0250 untuk informasi lebih lanjut tentang pengkodean karakter.

BAB 2
CSS

2.1 MENAMBAHKAN STYLE DENGAN CSS

Dalam bab ini, kita mempelajari apa itu *Cascading Style Sheets* (CSS) dan bagaimana menggunakannya untuk berbagai tujuan layout dan gaya. Kami menyarankan kita untuk mengerjakan bab ini dari awal sampai akhir, karena beberapa latihan dibangun di atas latihan sebelumnya.

Menemukan Apa yang Dapat dan Tidak Dapat Dilakukan CSS untuk Halaman Web Anda

Bagian ini melihat CSS dari tingkat tinggi untuk memberi kita dasar di mana kita akan belajar cara menggunakan CSS di situs web Anda.

Apa itu CSS?

CSS melengkapi HTML dengan memberikan tampilan dan nuansa ke halaman web. Halaman HTML yang kita buat di bab sebelumnya tampak cukup sederhana, dengan font dan ukuran font default. Dengan menggunakan CSS, kita dapat membumbui tampilan itu, menambahkan warna dan gambar latar belakang, mengubah font dan ukuran font, menggambar batas di sekitar area, dan bahkan mengubah layout halaman itu sendiri CSS memiliki bahasanya sendiri, terpisah dari HTML, tetapi kita tidak akan melakukannya. t menggunakan CSS tanpa halaman HTML. Dengan kata lain, meskipun HTML dapat berdiri sendiri dan menampilkan halaman ke browser, CSS tidak bisa. kita tidak akan menulis halaman CSS. Sebaliknya, kita menulis HTML dan kemudian menggunakan CSS untuk membantu menata halaman tersebut agar terlihat seperti yang kita inginkan. Seperti HTML, CSS ditentukan oleh spesifikasi, dengan versi terbaru adalah CSS 3, yang dikenal sebagai CSS3.

Menggunakan CSS (*Cascading Style Sheets*), kita dapat menerapkan gaya ke halaman web kita agar terlihat persis seperti yang kita inginkan. Ini berfungsi karena CSS terhubung ke DOM (Document Object Model). Dengan CSS dan integrasinya dengan DOM, kita dapat dengan cepat dan mudah menata ulang elemen apa pun. Misalnya, jika kita tidak menyukai tampilan default <h1>, <h2>, dan tag heading lainnya, kita dapat menetapkan gaya baru untuk mengesampingkan pengaturan default untuk jenis dan ukuran font yang digunakan, atau apakah huruf tebal atau miring harus diatur, dan banyak lagi properti juga. Salah satu cara kita dapat menambahkan gaya ke halaman web adalah dengan menyisipkan pernyataan yang diperlukan ke kepala halaman web di antara tag <head> dan </head>. Jadi, untuk mengubah gaya tag <h1>, kita dapat menggunakan kode berikut (saya akan menjelaskan sintaksnya nanti):

```
<style>
h1 { color:red; font-size:3em; font-family:Arial; }
</style>
```

Dalam halaman HTML ini mungkin terlihat seperti Contoh 1 (lihat Gambar 2.1), yang, seperti semua contoh dalam bab ini, menggunakan deklarasi DOCTYPE HTML5 standar.

Contoh 1. Halaman HTML sederhana

```
<!DOCTYPE html>
<html>
<head>
```

```
<title>Hello World</title>
<style>
h1 { color:red; font-size:3em; font-family:Arial; }
</style>
</head>
<body>
<h1>Hello there</h1>
</body>
</html>
```

Gambar 2.1 Menata tag, dengan gaya asli yang ditampilkan di sisipan

Mengapa Menggunakan CSS?

Sebelum adanya CSS, seorang developer HTML mengubah font dan warna dengan cara mengubah atribut pada setiap elemen. Jika developer HTML ingin semua judul terlihat dengan cara tertentu, maka ia harus mengubah setiap judul tersebut, dan, bayangkan saja untuk melakukan ini pada lebih dari 10 halaman atau bhakan 50 halaman, sangat melelahkan, membosankan dan memakan waktu bukan. CSS meringankan beban pembaruan elemen satu per satu ini dan membuatnya sehingga kita dapat menerapkan satu gaya tunggal di satu atau beberapa elemen. Kita dapat menerapkan beberapa gaya ke elemen yang sama, dan kita dapat menargetkan gaya tertentu hingga ke elemen individual. Misalnya, jika ingin semua judul menjadi font tebal tetapi judul tertentu harus miring, kita dapat melakukannya dengan CSS. Gunakan CSS untuk membuat perubahan pada layout, tampilan, dan nuansa halaman web. CSS mempermudah pengelolaan perubahan ini.

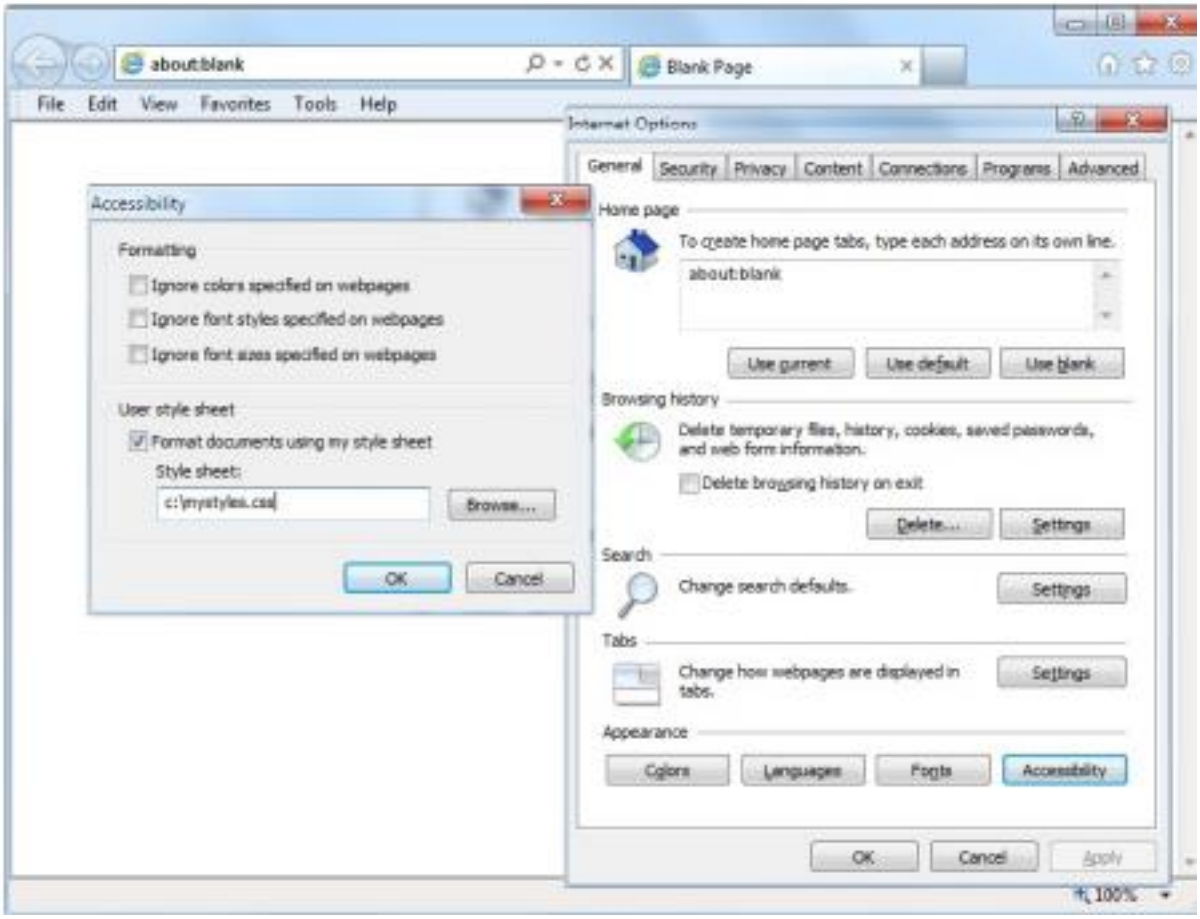
2.2 KETERBATASAN CSS

CSS memiliki batasan, dan keterbatasan utama CSS adalah tidak semua browser web mendukung CSS dengan cara yang persis sama. Satu browser mungkin menafsirkan layout dengan cara yang sedikit berbeda, menempatkan item lebih tinggi atau lebih rendah atau di tempat yang sama sekali berbeda. Selain itu, browser lama tidak mendukung versi CSS yang lebih baru, khususnya spesifikasi CSS3. Ini berarti bahwa browser tersebut tidak dapat menggunakan beberapa fitur dari spesifikasi CSS3. Untuk menyiasatinya, kita dapat menggunakan versi lama dari spesifikasi yang lebih banyak didukung oleh browser lama tersebut.

Kuncinya adalah menguji di beberapa browser. Browser web seperti Firefox, Chrome, dan Safari semuanya dapat diunduh gratis, dan Microsoft menawarkan perangkat lunak yang disebut PC Virtual untuk Kompatibilitas Aplikasi, yang merupakan versi Windows gratis, terbatas waktu, yang menyertakan versi Internet Explorer yang lebih lama. kita dapat menjalankannya di dalam perangkat lunak emulasi Virtual PC gratis Microsoft. Dengan menguji di browser lain, kita dapat melihat bagaimana situs akan terlihat di browser tersebut dan memperbaiki masalah layout sebelum menyebarkan situs ke Internet.

2.3 TIPE GAYA

Ada sejumlah jenis gaya yang berbeda, mulai dari gaya default yang diatur oleh browser (dan user style apa pun yang mungkin telah diterapkan di browser untuk mengganti defaultnya), melalui gaya sebaris atau yang disematkan, hingga style sheet eksternal. Gaya yang didefinisikan di setiap jenis memiliki hierarki prioritas, dari rendah hingga tinggi.



Gambar 2.2 Menerapkan user style ke Internet Explorer

Gaya Default

Tingkat prioritas gaya terendah adalah gaya default yang diterapkan oleh browser web. Gaya ini dibuat sebagai fallback ketika halaman web tidak memiliki gaya apa pun, dan gaya ini dimaksudkan sebagai kumpulan gaya umum yang akan ditampilkan dengan cukup baik di sebagian besar kasus. Pra-CSS, ini adalah satu-satunya gaya yang diterapkan pada dokumen, dan hanya sedikit dari mereka yang dapat diubah oleh halaman web (seperti tampilan font, warna, dan ukuran, dan beberapa argumen ukuran elemen).

User style

Ini adalah prioritas gaya tertinggi berikutnya, dan didukung oleh sebagian besar browser modern tetapi diterapkan secara berbeda oleh masing-masing browser. Jika kita ingin mempelajari cara membuat gaya default kita sendiri untuk menjelajah, gunakan mesin pencari untuk memasukkan nama browser kita diikuti dengan "user style" (mis., "User style Firefox" atau "User style Opera") untuk mengetahui caranya . Gambar 2.3 menunjukkan style sheet pengguna yang diterapkan ke Microsoft Internet Explorer. Jika user style ditetapkan yang telah ditetapkan sebagai default browser, maka pengaturan default browser akan ditimpa. Gaya apa pun yang tidak ditentukan dalam style sheet pengguna akan mempertahankan nilai defaultnya seperti yang diatur di browser.

2.4 MENYAMBUNGKAN CSS KE HALAMAN

Kita dapat menambahkan CSS ke halaman dengan beberapa cara berbeda:

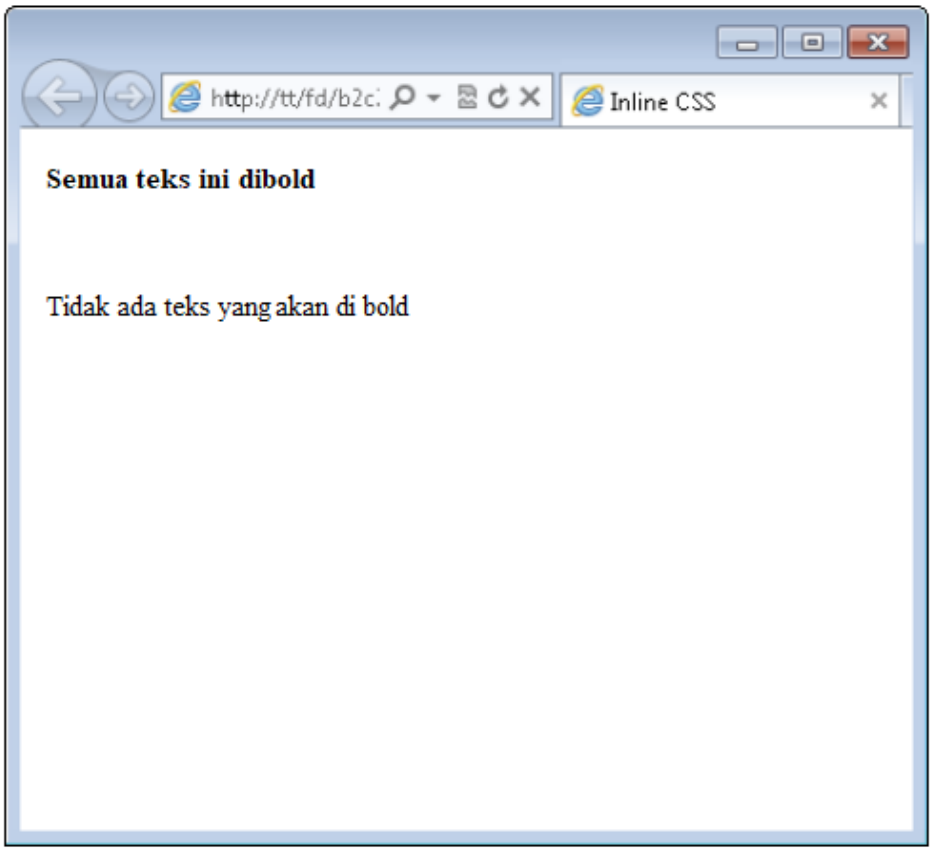
- Langsung ke elemen HTML
- Dengan style sheet internal
- Dengan style sheet eksternal

Cara yang paling dapat digunakan kembali untuk menambahkan CSS ke halaman adalah dengan menggunakan style sheet eksternal, tetapi yang paling sederhana adalah menambahkan informasi gaya secara langsung pada sebuah elemen.

Menambahkan style ke elemen HTML

Menambahkan gaya ke hampir semua elemen HTML dengan atribut gaya, seperti dalam contoh ini yang membuat semua teks di paragraf pertama menjadi font tebal:

```
<p style=”font-weight: bold;”>All of this text will be bold.</p>
```



Gambar 2.3 Teks tebal diatur dengan CSS

Menghitung Spesifisitas

Kami menghitung kekhususan aturan dengan membuat nomor tiga bagian berdasarkan jenis selektor dalam daftar bernomor di bagian sebelumnya. Bilangan majemuk ini mulai terlihat seperti [0,0,0]. Saat memproses aturan, setiap selektor yang mereferensikan ID menambah angka pertama sebanyak 1, sehingga angka majemuk menjadi [1,0,0]. Mari kita lihat aturan berikut, yang memiliki tujuh referensi, dengan tiga di antaranya ke ID #heading, #main, dan #menu. Sehingga bilangan majemuk menjadi [3,0,0].

```
#heading #main #menu .text .quote p span {
// Rules go here;
}
```

Kemudian jumlah kelas dalam selektor ditempatkan di bagian kedua dari bilangan majemuk. Dalam contoh ini ada dua (.text dan .quote), sehingga bilangan majemuk menjadi [3,2,0]. Akhirnya, semua selektor yang tag elemen referensi dihitung, dan nomor ini ditempatkan di bagian terakhir dari nomor gabungan. Dalam contoh, ada dua (p dan span), sehingga bilangan majemuk akhir menjadi [3,2,2], itu saja yang diperlukan untuk membandingkan kekhususan aturan ini dengan yang lain, seperti berikut ini:

```
#heading #main .text .quote .news p span {
// Rules go here;
}
```

Di sini, meskipun tujuh elemen juga direferensikan, sekarang hanya ada dua referensi ID, tetapi ada tiga referensi kelas, yang menghasilkan bilangan majemuk [2,3,2]. Karena 322 lebih besar dari 232, contoh pertama lebih diutamakan daripada yang terakhir.

Dalam kasus di mana ada sembilan atau kurang dari setiap jenis dalam bilangan majemuk, Kita dapat mengonversinya langsung ke bilangan desimal, yang dalam hal ini adalah 352. Aturan dengan angka lebih rendah dari ini akan memiliki prioritas lebih rendah, dan aturan dengan angka lebih tinggi nomor akan lebih diutamakan. Di mana dua aturan berbagi nilai yang sama, aturan yang paling baru diterapkan menang

Style sheet Eksternal

Jenis gaya berikutnya adalah yang ditetapkan dalam style sheet eksternal. Pengaturan ini akan menimpa semua yang ditetapkan baik oleh pengguna atau oleh browser. Style sheet eksternal adalah cara yang disarankan untuk membuat gaya karena Kita dapat menghasilkan style sheet yang berbeda untuk tujuan yang berbeda seperti penataan gaya untuk penggunaan web umum, untuk dilihat pada browser seluler dengan layar yang lebih kecil, untuk tujuan pencetakan, dan sebagainya. Cukup terapkan yang dibutuhkan untuk setiap jenis media saat membuat halaman web.

Style Internal

Lalu ada gaya internal, yang kita buat dalam tag <style> ... </style>, dan yang lebih diutamakan daripada semua jenis gaya sebelumnya. Namun, pada titik ini, kita harus mulai memecah pemisahan antara gaya dan konten, karena setiap style sheet eksternal yang dimuat pada saat yang sama akan memiliki prioritas yang lebih rendah.

Stykle Inline

Terakhir, gaya sebaris adalah tempat Kita menetapkan properti langsung ke elemen. Mereka memiliki prioritas tertinggi dari semua jenis gaya, dan digunakan seperti ini:

```
<a href="http://google.com" style="color:green;">Visit Google</a>
```

Dalam contoh ini, tautan yang ditentukan akan ditampilkan dalam warna hijau, terlepas dari pengaturan warna default atau lainnya yang diterapkan oleh jenis style sheet lainnya, baik secara langsung ke tautan ini atau secara umum untuk semua tautan.

2.5 CASCADE CSS

Salah satu hal yang paling mendasar tentang properti CSS adalah bahwa mereka mengalir, itulah sebabnya mereka disebut Cascading Style Sheets. Tapi apa artinya ini? Cascading adalah metode yang digunakan untuk menyelesaikan potensi konflik antara berbagai jenis style sheet yang didukung browser, dan menerapkannya dalam urutan prioritas menurut siapa yang membuatnya, metode yang digunakan untuk membuat gaya, dan jenis properti yang dipilih.

Creator Style Sheet

Ada tiga jenis utama style sheet yang didukung oleh semua browser modern. Dalam urutan prioritas dari tinggi ke rendah, mereka adalah:

1. dibuat oleh penulis dokumen
2. dibuat oleh pengguna
3. dibuat oleh browser

Ketiga set style sheet ini diproses dalam urutan terbalik. Pertama, default di browser web diterapkan ke dokumen. Tanpa default ini, halaman web yang tidak menggunakan style sheet akan terlihat buruk. Mereka termasuk wajah font, ukuran, dan warna; jarak elemen; batas dan spasi tabel; dan semua standar wajar lainnya yang diharapkan pengguna. Selanjutnya, jika pengguna telah membuat gaya apa pun untuk digunakan alih-alih gaya standar, gaya ini diterapkan, menggantikan gaya default browser apa pun yang mungkin bertentangan. Terakhir, gaya apa pun yang dibuat oleh penulis dokumen saat ini kemudian diterapkan, menggantikan gaya apa pun yang telah dibuat baik sebagai default browser atau oleh pengguna.

Metode Style sheet

Style sheet dapat dibuat melalui tiga metode berbeda. Dalam urutan prioritas dari tinggi ke rendah, mereka adalah:

1. Sebagai gaya sebaris
2. Dalam style sheet yang disematkan
3. Sebagai style sheet eksternal

Sekali lagi, metode pembuatan style sheet ini diterapkan dalam urutan prioritas yang terbalik. Oleh karena itu, semua style sheet eksternal diproses terlebih dahulu, dan gayanya diterapkan ke dokumen. Selanjutnya, setiap gaya yang disematkan (dalam tag <style> ... </style>) diproses, dan semua yang bertentangan dengan aturan eksternal akan diprioritaskan dan akan menimpanya. Terakhir, gaya apa pun yang diterapkan langsung ke elemen sebagai gaya sebaris (seperti ...) diberi prioritas tertinggi, dan menimpa semua properti yang ditetapkan sebelumnya.

Mengimpor Style Sheet

Saat kita ingin menata seluruh situs, daripada satu halaman, cara yang lebih baik untuk mengelola style sheet adalah dengan memindahkannya sepenuhnya dari halaman web ke file yang terpisah, lalu mengimpor yang kita butuhkan. Ini memungkinkan menerapkan style sheet yang berbeda untuk layout yang berbeda (seperti web dan cetak), tanpa mengubah HTML. Ada beberapa cara berbeda untuk mencapai ini, bagian pertama adalah dengan menggunakan arahan CSS @import seperti ini:

```
<style>
@import url('styles.css');
</style>
```

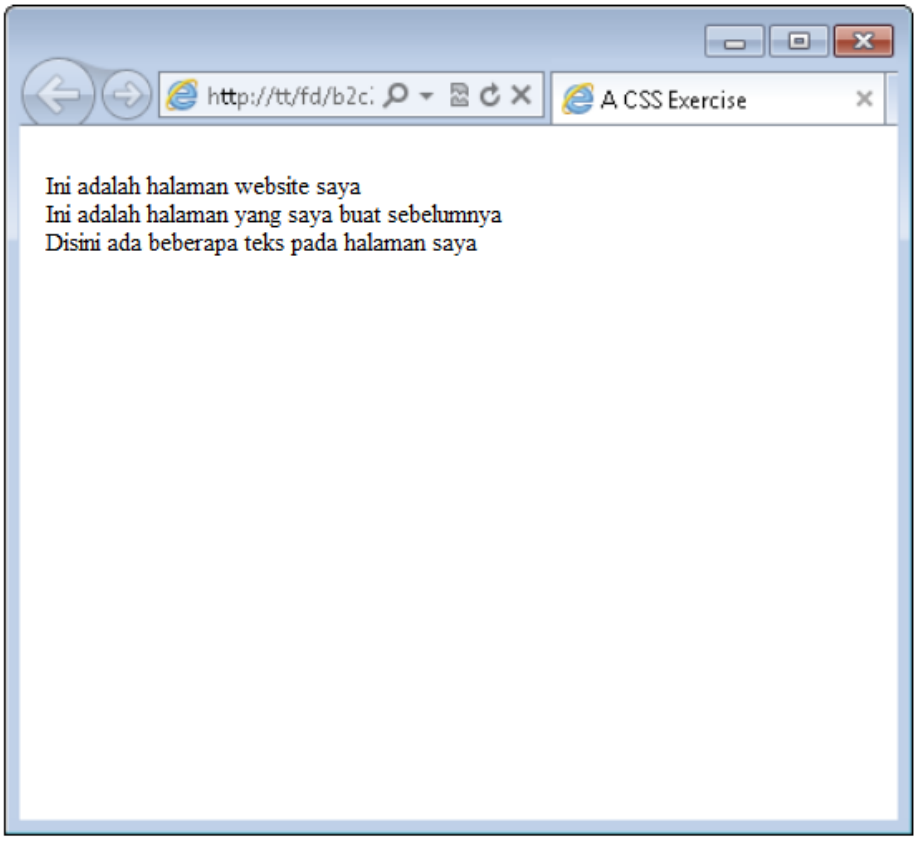
Pernyataan ini memberitahu browser untuk mengambil style sheet dengan nama styles.css. Perintah @import cukup fleksibel karena kita dapat membuat style sheet yang menarik daripada style sheet lain, dan seterusnya. kita hanya perlu memastikan bahwa tidak ada tag <style> atau </style> di salah satu style sheet eksternal Anda, atau mereka tidak akan berfungsi.

Berikut contoh yang bisa kita coba. kita membuat beberapa HTML terlebih dahulu dan kemudian mulai menambahkan gaya ke dalamnya.

1. Buka text editor
2. Di dalam dokumen teks kosong, tempatkan HTML berikut:

```
<!doctype html>
<html>
<head>
<title>A CSS Exercise</title>
</head>
<body>
<div>This is my web page.</div>
<div>
 This is the <span>nicest</span> page I've made yet.
</div>
<div>Here is some text on my page.</div>
</body>
</html>
```

3. Simpan file sebagai css.html.
   Di dalam text editor, simpan file menggunakan nama css.html, pastikan tidak ada spasi atau karakter lain dalam nama file. File harus disimpan di dalam root dokumen Anda.
4. Buka browser web kita dan lihat halamannya.
   Di dalam bilah alamat browser web, ketik http://localhost/css.htmldan kita akan melihat halaman yang mirip dengan yang ditunjukkan pada gambar berikut.

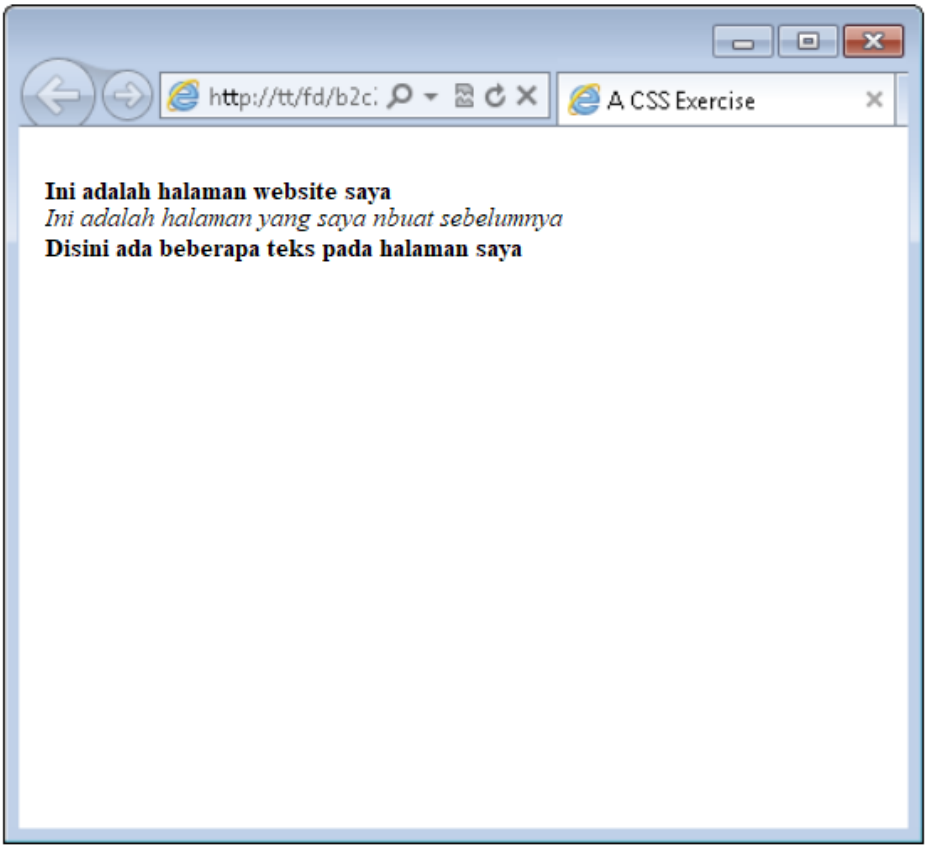


Gambar 2.4 Membuat halaman web sederhana.

5. Tutup browser.
   Sekarang setelah kita memverifikasi bahwa halaman tersebut berfungsi, tutup browser.
6. Beralih ke text editor untuk mengedit HTML.
   Di dalam text editor, edit HTML dari Langkah 2 untuk menambahkan CSS. Jika kita menutup file, buka kembali di text editor.
7. Ubah HTML untuk menambahkan dua atribut gaya yang berbeda, seperti yang ditunjukkan di sini:

```
<!doctype html>
<html>
<head>
<title>A CSS Exercise</title>
</head>
<body>
<div style="font-weight: bold;"> Ini adalah halaman website saya.</
div>
<div>
Ini adalah <span style="font-style:
italic;">nicest</span> halaman yang saya buat sebelumnya.
</div>
<div style="font-weight: bold;"> Disini ada beberapa teks pada
halaman saya.</div>
</body>
</html>
```

8. Simpan file.
   Anda dapat menyimpannya sebagai css.html atau menyimpannya sebagai css2.html jika kita tidak ingin menimpa file css.html asli Anda. File harus disimpan di root dokumen Anda.
9. Buka browser web kita dan lihat halamannya.
   Mengetik di http://localhost/css.html (atau css2.html jika kita menyimpannya sebagai css2.html) mengungkapkan file, sekarang dengan gaya sebaris diterapkan ke dua area. Ini diilustrasikan seperti gambar berikut ini.



Gambar 2.5 Menambahkan gaya sebaris ke HTML.

Latihan ini membuat file HTML yang menggunakan elemen dan . HTML kemudian ditata menggunakan gaya sebaris. Gaya sebaris menyesuaikan bobot font dan gaya font untuk membuat teks tebal untuk dua elemen dan teks miring untuk satu elemen. Ketika digunakan dengan CSS, font-weight dan font-style dikenal sebagai properti. Properti ini kemudian diberi nilai, seperti tebal dan miring. Saat kita melihat terminologi bahwa properti CSS telah diubah, kita tahu bahwa properti itu adalah nama dan nilainya adalah untuk mengubah properti itu.

Selektor Style sheet

Ada tiga cara berbeda untuk memilih elemen yang akan ditata. Mulai dari urutan prioritas tertinggi ke terendah, mereka adalah:

- Referensi oleh ID individu atau selektor atribut
- Referensi dalam kelompok berdasarkan kelas
- Referensi oleh tag elemen (seperti <p> atau <b>)

Selektor diproses sesuai dengan jumlah dan jenis elemen yang dipengaruhi oleh aturan, yang sedikit berbeda dari dua metode sebelumnya untuk menyelesaikan konflik. Ini karena aturan tidak harus berlaku hanya untuk satu jenis selektor pada satu waktu, dan mungkin merujuk ke

banyak selektor yang berbeda. Oleh karena itu, diperlukan suatu metode untuk menentukan prioritas aturan yang dapat memuat kombinasi selektor apa saja. Ini dilakukan dengan menghitung kekhususan setiap aturan dengan mengurutkannya dari lingkup tindakan terluas hingga tersempit.

Menggunakan style sheet internal

Menerapkan gaya ke elemen individual dengan cepat akan menjadi rumit ketika kita memiliki halaman web yang besar. Seperti yang kita lihat di latihan sebelumnya, untuk membuat teks dari dua elemen menjadi tebal, kita perlu menambahkan atribut gaya ke setiap elemen . Untungnya, ada cara yang lebih baik. kita dapat membuat area khusus halaman web untuk menyimpan informasi gaya. Informasi gaya ini kemudian diterapkan ke elemen yang sesuai dalam HTML. Ini mengurangi kebutuhan untuk menambahkan atribut gaya ke setiap elemen. kita menambahkan gaya internal di dalam bagian <head> halaman web menggunakan elemen <style>. Daftar dibawah ini menunjukkan HTML dengan elemen <style>.

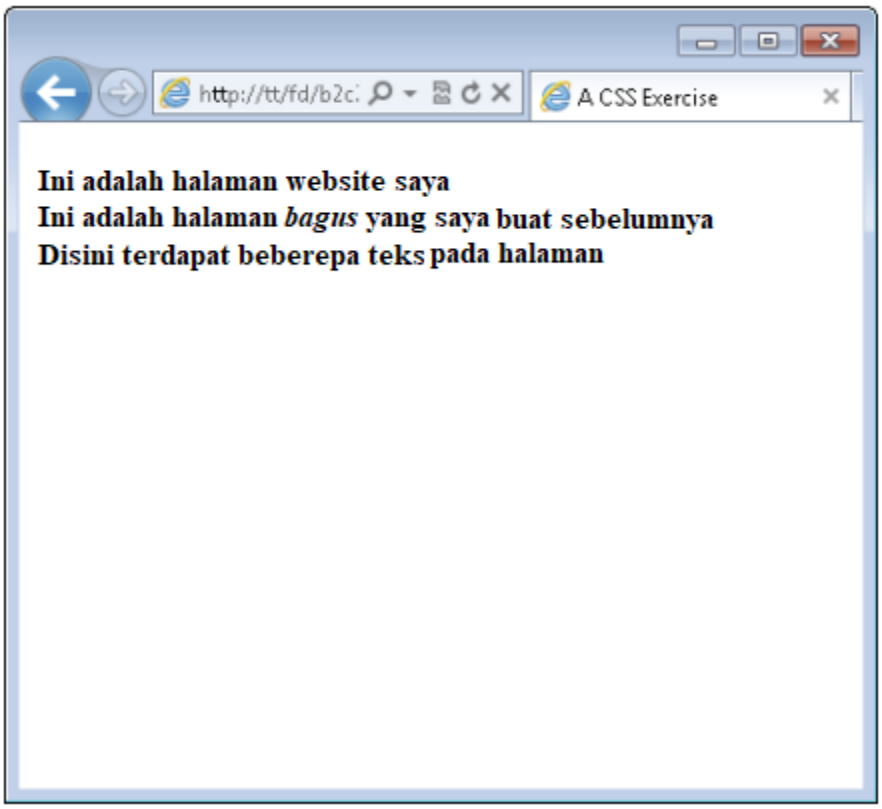
Contoh 2 Menggunakan Style sheet Internal

```
<!doctype html>
<html>
<head>
 <title>A CSS Exercise</title>
 <style type=”text/css”>
 div {
 font-weight: bold;
 }
 span {
 font-style: italic;
 }
 </style>
</head>
<body>
<div>Ini adalah halaman website saya.</div>
<div>
Ini adalah <span>nicest</span> halaman yang saya buat sebelumnya.
</div>
<div>Disini ada beberapa teks pada halaman saya.</div>
</body>
</html>
```

Halaman menambahkan style sheet internal untuk menambahkan font tebal ke elemen dan font gaya miring ke semua elemen di halaman.

```
<style type=”text/css”>
 div {
 font-weight: bold;
 }
 span {
 font-style: italic;
 }
 </style>
```

Elemen <style> menggunakan atribut type untuk memberi tahu browser jenis informasi gaya apa yang diharapkan. Dalam hal ini, kami menggunakan gaya tipe teks/css. Perhatikan juga tag penutup, yang diperlukan.



Gambar 2.6 Menggunakan style sheet internal

Perhatikan baik-baik diatas; terdapat sedikit perbedaan dengan tampilan dari gambar sebelumnya, bagian baris kedua dari gambar sebelumnya tidak dicetak tebal, tetapi baris muncul dalam huruf tebal pada gambar selanjutnya. Perbedaan ini muncul karena gaya internal sheet menargetkan semua elemen di halaman, bukan hanya yang spesifik yang diubah dengan metode gaya sebaris yang ditunjukkan sebelumnya.

Menggunakan style sheet eksternal

Sampai sini kita telah melihat bagaimana gaya sebaris menambahkan informasi gaya ke setiap elemen satu per satu, ini bisa menjadi membosankan. Dilanjutkan dengan cara menggunakan style sheet internal untuk membuat gaya informasi untuk halaman secara keseluruhan. Browser membaca style sheet eksternal ini seperti halnya membaca gaya yang diterapkan di dalam halaman itu sendiri, dan menerapkan gaya tersebut sesuai dengan itu. kita menambahkan atau menyertakan style sheet eksternal dengan elemen <link>, yang berada di area <head> sebuah halaman HTML.

Elemen <link> khas untuk menambahkan CSS terlihat seperti ini:

```
<link rel=”stylesheet” type=”text/css” href=”style.css”>
```

Baris itu menyertakan file bernama style.css di direktori saat ini dan memasukkannya ke dalam halaman. Semua informasi <style> dan gaya sebaris dapat dihapus menggantikan satu baris di bagian <head> halaman. Di dalam style sheet eksternal terdapat aturan yang harus diterapkan — dan hanya aturan yang harus diterapkan. kita tidak perlu menyertakan atribut gaya atau bahkan elemen pembuka atau penutup <style> dalam style sheet eksternal.

```
div {
 font-weight: bold;
 }
 span {
 font-style: italic;
 }
```

Sekarang style sheet eksternal dapat dibagikan di antara beberapa file HTML. Jika kita perlu membuat perubahan gaya, kita hanya perlu mengedit satu file CSS, dan itu secara otomatis menerapkan gaya ke semua halaman yang menggunakan file CSS itu. Seperti yang kita lihat, file CSS eksternal membuat pemeliharaan situs web menjadi lebih mudah. Style sheet eksternal adalah metode yang disarankan untuk menggunakan CSS, dan dengan hanya beberapa pengecualian, sisa buku ini menggunakan CSS yang disertakan dari style sheet eksternal.

Penargetan Style

Ingat masalah yang diidentifikasi sebelumnya, di mana font tebal diterapkan ke semua elemen pada halaman, ketika kita mungkin tidak ingin menerapkannya ke semua elemen tersebut. kita dapat memperbaiki masalah tersebut dengan menargetkan atau mempersempit cakupan aturan CSS menggunakan selector yang lebih spesifik. CSS menggunakan selector untuk menentukan elemen atau menerapkan aturan. Dalam contoh style sheet internal sebelumnya selector adalah elemen , atau semua elemen pada halaman.

2.6 MENGIMPOR CSS DARI DALAM HTML

Anda juga dapat menyertakan style sheet dengan tag <link> HTML seperti ini:

```
<link rel='stylesheet' type='text/css' href='styles.css'>
```

Ini memiliki efek yang sama persis dengan direktif @import, kecuali bahwa <link> adalah tag khusus HTML dan bukan direktif gaya yang valid, sehingga tidak dapat digunakan dari dalam satu style sheet untuk menarik yang lain, dan juga tidak dapat ditempatkan dalam sepasang tag <style> ... </style>. Sama seperti kita dapat menggunakan beberapa @import directives dalam CSS kita untuk menyertakan beberapa style sheet eksternal, kita juga dapat menggunakan elemen <link> sebanyak yang kita suka dalam HTML Anda.

Pengaturan Gaya Tertanam

Juga tidak ada yang menghentikan kita untuk mengatur atau mengganti gaya tertentu secara individual untuk halaman saat ini berdasarkan kasus per kasus dengan menyisipkan deklarasi gaya langsung di dalam HTML, seperti ini (yang menghasilkan teks biru miring di dalam tag):

```
<div style='font-style:italic; color:blue;'>Hello there</div>
```

Tapi ini harus dicadangkan hanya untuk keadaan yang paling luar biasa, karena melanggar pemisahan konten dan presentasi.

Memilih elemen HTML

Hampir semua elemen HTML dapat menjadi target selector, bahkan hal-hal seperti elemen <body>. Faktanya, elemen <body> sering digunakan sebagai selector untuk

menargetkan gaya di seluruh halaman, seperti kumpulan font apa yang akan digunakan untuk halaman. kita melihat contohnya di bagian berikutnya, "Changing font." kita telah melihat contoh menggunakan elemen HTML sebagai penyeleksi. kita cukup menggunakan nama elemen, tanpa tanda kurung di sekitarnya. Alih-alih seperti dalam HTML, kita menggunakan div saat menggunakannya sebagai selector CSS. Berikut tampilannya:

```
div {
 font-berat: tebal;
 }
```

Seperti yang kita lihat, nama elemen, div, diikuti oleh kurung kurawal. Ini menunjukkan bahwa aturan dimulai. Di dalam kurung kurawal pembuka dan penutup yang sesuai, properti, font-weight, dipilih, diikuti oleh titik dua (:). Nilai tersebut kemudian disetel menjadi tebal. Baris diakhiri dengan titik koma (;). Titik koma ini memberi tahu browser bahwa baris sudah selesai; dengan kata lain, pasangan properti/nilai ditutup. Beberapa properti dapat diatur dalam selector yang sama. Mengambil contoh sebelumnya, kita dapat mengubah gaya font menjadi tebal dan miring, seperti ini:

```
div {
 font-weight: bold;
 font-style: italic;
 }
```

Setiap baris diakhiri dengan titik koma, dan seluruh aturan diapit oleh kurung kurawal buka dan tutup.

Memilih elemen individu

Id (singkatan dari identifier) memungkinkan kita untuk memilih satu dan hanya satu elemen dalam sebuah halaman. Untuk melakukannya, kita perlu memodifikasi HTML untuk menambahkan atribut id dan memberikan nama untuk elemen tersebut. Misalnya, pertimbangkan HTML seperti ini:

```
<div>Agus Widodo</div>
```

Jika ingin menerapkan font tebal ke elemen itu, kita bisa memilih semua elemen tetapi itu juga akan menerapkan font tebal ke elemen lain di halaman. Sebagai gantinya, solusinya adalah menambahkan id ke tertentu, seperti:

```
<div id=”myName”> Agus Widodo</div>
```

Nilai id disetel ke myName. Perhatikan kasus yang digunakan dalam contoh ini, dengan huruf besar N. Huruf besar ini harus dicocokkan dengan CSS. Untuk memilih id ini di dalam CSS, kita menggunakan karakter hash (#), seperti:

```
#myName
```

Dengan mengingat hal itu, membuat #myName id tebal terlihat persis seperti contoh yang telah kita lihat, cukup ganti #myName untuk div:

```
#myName {
```

```
font-weight: bold;
}
```

Selalu cocokkan case yang kita gunakan di HTML dengan case yang kita gunakan di CSS. Jika kita menggunakan huruf besar semua dalam memberi nama ID di HTML, maka gunakan huruf besar semua di CSS. Jika kita menggunakan semua huruf kecil di HTML, gunakan huruf kecil di CSS. Jika kita menggunakan kombinasi, seperti contoh, maka cocokkan kombinasi tersebut di CSS. Saat menggunakan ID dalam HTML, penting untuk menyadari bahwa ID harus digunakan sekali di seluruh halaman. Tak masalah menggunakan ID yang sama di halaman yang berbeda, tetapi ID tersebut seharusnya hanya muncul sekali dalam satu halaman.

Solusi yang lebih baik untuk menyetel gaya elemen adalah dengan menetapkan ID ke dalamnya dalam HTML, seperti ini:

```
<div id='welcome'>Hello there</div>
```

Ini menyatakan bahwa konten dengan sambutan ID harus menerapkan gaya yang ditentukan dalam pengaturan gaya sambutan. Pernyataan CSS yang cocok untuk ini mungkin terlihat seperti berikut:

```
#welcome { font-style:italic; color:blue; }
```

Catatan: Perhatikan penggunaan simbol #, yang menyatakan bahwa hanya ID dengan nama selamat datang yang harus diberi gaya dengan pernyataan ini.

Memilih sekelompok elemen

Kita telah mempelajari cara menargetkan elemen HTML di seluruh halaman dan cara menargetkan hanya satu elemen individual. Kelas CSS digunakan untuk memilih beberapa elemen dan menerapkan gaya ke dalamnya. Berbeda dengan selectoran yang terjadi saat kita memilih semua elemen , kelas CSS hanya diterapkan pada elemen tertentu yang kita pilih. Elemen HTML bahkan tidak perlu dari jenis yang sama; kita dapat menerapkan kelas CSS yang sama ke , ke tag <img>, dan ke elemen <p> sama.

2.7 COMMENT CSS

Di dalam aturan CSS yang ditampilkan di dekatnya, ada komentar: /* CSS Goes Here */. Sama seperti di HTML di mana kita dapat menggunakan komentar untuk membantu menjelaskan bagian kode tertentu, demikian juga kita dapat menggunakan komentar di CSS untuk membantu menjelaskan CSS. Seperti komentar HTML, komentar dalam CSS tidak terlihat di output halaman tetapi, juga seperti komentar HTML, komentar CSS dapat dilihat dengan melihat sumber dari dokumen HTML atau CSS itu sendiri. Ini berarti pengunjung juga dapat melihatcomment.

Komentar dalam CSS dibuka dengan /* dan ditutup dengan */. Segala sesuatu yang muncul di antara /* dan */ diperlakukan sebagai komentar.

Seperti id, kelas diterapkan terlebih dahulu ke elemen HTML dengan atribut. Atributnya adalah kelas dengan judul yang tepat, seperti dalam contoh ini:

```
<div class=”boldText”>This text has a class.<div>
```

Seperti pada contoh id, kelas juga peka huruf besar/kecil. Kasing yang digunakan dalam HTML harus cocok dengan yang ada di CSS. Sedangkan selector ID menggunakan tanda pound (#) di

CSS, kelas menggunakan satu titik atau titik. Dalam contoh sebelumnya, di mana kelas diberi nama boldText di HTML, itu akan direferensikan seperti ini di CSS:

```
.boldText {
/* CSS Goes Here */
}
```

Dalam contoh ini, kelas boldText dipilih. Kelas dapat digunakan untuk memecahkan masalah yang ditemukan sebelumnya (di bagian "Menggunakan style sheet internal"), di mana font tebal diterapkan ke semua elemen karena CSS menggunakan selector div. kita dapat menggunakan kelas dalam HTML untuk menargetkan hanya elemen-elemen yang ingin kita targetkan.

Merupakan ide yang baik untuk mengomentari aturan CSS Anda, bahkan jika kita hanya menjelaskan kelompok utama pernyataan daripada semua atau sebagian besar dari mereka. kita dapat melakukan ini dengan dua cara berbeda. Pertama, kita dapat menempatkan komentar di dalam sepasang tag /* ... */, seperti ini:

```
/* This is a CSS comment */
```

Atau kita dapat memperpanjang komentar melalui banyak baris, seperti ini:

```
/*
A Multi
line
comment
*/
```

Saatnya menguji teori itu. Ikuti langkah ini.

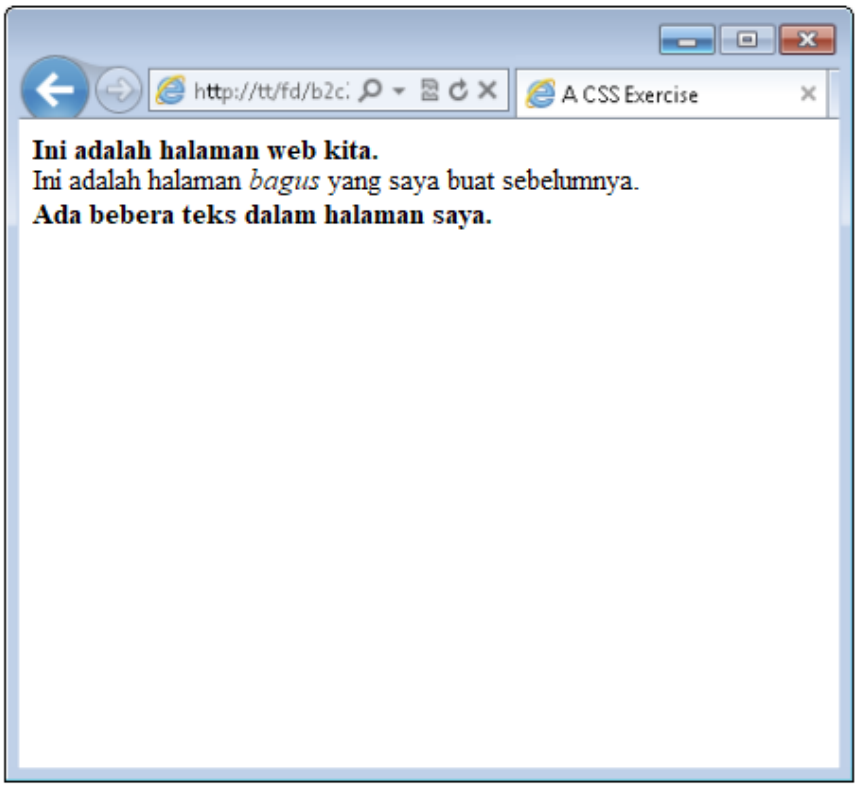
1. Buka text editor
2. Buka css.html
3. Ubah css.html untuk mengubah CSS

```
<!doctype html>
<html>
<head>
<title>A CSS Exercise</title>
<link rel=”stylesheet” type=”text/css” href=”style.
css”>
</head>
<body>
<div class=”boldText”>This is my web page.</div>
<div>
 Ini adalah <span>nicest</span> halaman yang saya buat sebelumnya.
</div>
<div class=”boldText”>Here is some text on my page.</
div>
</body>
</html>
```

4. Simpan file
   Anda dapat menyimpannya sebagai css.html atau mengganti namanya menjadi css3.html. Simpan file di root dokumen Anda.
5. Buat file teks kosong baru.
   Menggunakan text editor, buat file kosong baru.
6. Tempatkan CSS berikut dalam file.

```
.boldText {
 font-weight: bold;
 }
 span {
 font-style: italic;
 }
```

7. Simpan file.
   Simpan file sebagai style.css di dalam root dokumen Anda. Perhatikan bahwa kita harus memastikan bahwa file diberi nama dengan huruf kecil semua dan memiliki ekstensi file yang benar, .css.
8. Buka browser kita dan lihat file css.html.
   Ketik http://localhost/css.html di bilah alamat browser. Jika kita menyimpan file sebagai css3.html, gunakan itu sebagai ganti css.html. Outputnya akan terlihat seperti pada gambar dibawah ini.



Gambar 2.7 Halaman dengan eksternal style sheet.

2.8 MENGGUNAKAN KELAS

Jika kita ingin menerapkan gaya yang sama ke banyak elemen, kita tidak perlu memberikan masing-masing ID yang berbeda karena kita dapat menentukan kelas untuk mengelola semuanya, seperti ini:

Hello

Ini menyatakan bahwa konten elemen ini (dan elemen lain yang menggunakan kelas) harus menerapkan gaya yang didefinisikan di kelas selamat datang. Setelah kelas diterapkan, kita dapat menggunakan aturan berikut, baik di header halaman atau di dalam style sheet eksternal untuk menyetel gaya kelas:

.welcome { font-style:italic; color:blue; }

Alih-alih simbol #, yang dicadangkan untuk ID, pernyataan kelas diawali dengan . (Titik).

Menggunakan Titik Koma

Dalam CSS, titik koma digunakan untuk memisahkan beberapa pernyataan CSS pada baris yang sama. Tetapi jika hanya ada satu pernyataan dalam aturan (atau dalam pengaturan gaya sebaris dalam tag HTML), kita dapat menghilangkan titik koma, seperti yang kita bisa untuk pernyataan terakhir dalam grup. Namun, untuk menghindari kesalahan CSS yang sulit ditemukan, kita dapat memilih untuk selalu menggunakan titik koma setelah setiap pengaturan CSS. kita kemudian dapat menyalin dan menempelkannya, dan sebaliknya memodifikasi properti, tanpa khawatir menghapus titik koma di tempat yang tidak terlalu diperlukan atau harus menambahkannya jika diperlukan.

2.9 ATURAN CSS

Setiap pernyataan dalam aturan CSS dimulai dengan pemilih, yang merupakan item yang akan diterapkan aturan. Misalnya, dalam tugas ini, h1 adalah selektor yang diberi ukuran font 240% lebih besar dari default:

h1 { font-size:240%; }

font-size adalah properti. Memberikan nilai 240% ke properti font-size dari selektor memastikan bahwa konten dari semua <h1> ... </h1> pasangan tag akan ditampilkan pada ukuran font yang 240% dari ukuran default. Semua perubahan aturan harus berada dalam simbol { dan } yang mengikuti pemilih. Dalam ukuran font: 240%; bagian sebelum : (titik dua) adalah properti, sedangkan sisanya adalah nilai yang diterapkan padanya. Terakhir datang ; (titik koma) untuk mengakhiri pernyataan. Dalam contoh ini, karena ukuran font adalah properti terakhir dalam aturan, titik koma tidak diperlukan (tetapi akan diperlukan jika tugas lain mengikuti).

Anda dapat membuat beberapa deklarasi gaya dalam beberapa cara berbeda. Pertama, kita dapat menggabungkannya pada baris yang sama, seperti ini:

h1 { font-size:240%; color:blue; }

Ini menambahkan tugas kedua yang mengubah warna semua judul <h1> menjadi biru. kita juga dapat menempatkan tugas satu per baris, seperti berikut:

```
h1 { font-size:240%;
color:blue; }
```

Atau kita dapat memberi ruang lebih sedikit pada tugas, sehingga mereka berbaris di bawah satu sama lain dalam kolom di titik dua, seperti ini:

```
h1 {
font-size:240%;
color :blue;
}
```

Dengan cara ini, kita dapat dengan mudah melihat di mana setiap kumpulan aturan baru dimulai, karena selektor selalu berada di kolom pertama, dan penetapan berikutnya berbaris rapi dengan semua nilai properti yang dimulai pada offset horizontal yang sama. Dalam contoh sebelumnya, titik koma terakhir tidak diperlukan, tetapi jika kita ingin menggabungkan kelompok pernyataan seperti itu menjadi satu baris, ini sangat cepat dilakukan dengan semua titik koma yang sudah ada. kita dapat menentukan selektor yang sama sebanyak yang kita inginkan, dan CSS menggabungkan semua properti. Jadi contoh sebelumnya juga dapat ditentukan sebagai:

```
h1 { font-size: 240%; }
h1 { color : blue; }
```

Bagaimana jika kita menentukan properti yang sama ke selektor yang sama dua kali?

```
h1 { color : red; }
h1 { color : blue; }
```

Nilai terakhir yang ditentukan—dalam hal ini, biru—akan berlaku. Dalam satu file, mengulangi properti yang sama untuk selektor yang sama tidak akan ada gunanya, tetapi pengulangan seperti itu sering terjadi di halaman web kehidupan nyata ketika beberapa style sheet diterapkan. Ini adalah salah satu fitur berharga dari CSS, dan dari mana istilah cascading berasal.

2.10 FONT DAN TIPOGRAFI

Ada empat properti font utama yang dapat kita gaya menggunakan CSS: keluarga, gaya, ukuran, dan berat. Di antara mereka, kita dapat menyempurnakan cara teks ditampilkan di halaman web kita dan/atau saat dicetak.

Mengubah Font

Sejauh ini kita telah melihat banyak perubahan berat font untuk membuat font tampak tebal dan sedikit tentang gaya font untuk membuat font muncul miring. Namun, kita dapat melakukan lebih banyak hal dengan font di web menggunakan CSS, termasuk memilih jenis font dan memilih ukuran dan warna font.

font-family

Properti font-family menetapkan font yang akan digunakan. Ini juga mendukung daftar berbagai font dalam urutan preferensi dari kiri ke kanan, sehingga gaya dapat mundur dengan anggun saat pengguna tidak menginstal font pilihan. Misalnya, untuk menyetel font default untuk paragraf, kita dapat menggunakan aturan CSS seperti ini:

```
p { font-family:Verdana, Arial, Helvetica, sans-serif; }
```

Jika nama font terdiri dari dua kata atau lebih, kita harus menyertakan nama dalam tanda kutip, seperti ini:

```
p { font-family:"Times New Roman", Georgia, serif; }
```

Istilah *font family* menggambarkan jenis huruf atau tampilan font yang digunakan untuk teks. Font family dapat diubah menggunakan CSS tetapi ada batasan besar: Font yang akan kita gunakan juga harus tersedia di komputer pengunjung. Secara praktis, ini berarti kita harus menggunakan font "ramah web" tertentu yang muncul di sebagian besar komputer pengunjung. Ini juga berarti bahwa kita tidak selalu dapat menjamin font apa yang akan dilihat pengunjung. Jika pengunjung tidak memiliki font yang kita tentukan, browser pengunjung tersebut akan otomatis mengganti jenis font tersebut ke font umum. Properti CSS untuk font disebut font-family.

Catatan: Karena harus tersedia di hampir semua browser web dan sistem operasi, jenis font paling aman untuk digunakan di halaman web adalah Arial, Helvetica, Times New Roman, Times, Courier New, dan Courier. Font Verdana, Georgia, Comic Sans MS, Trebuchet MS, Arial Black, dan Impact aman untuk penggunaan Mac dan PC, tetapi tidak dapat diinstal pada sistem operasi lain seperti Linux. Font umum lainnya tetapi kurang aman adalah Palatino, Garamond, Bookman, dan Avant Garde. Jika kita menggunakan salah satu font yang kurang aman, pastikan kita menawarkan fallback dari satu atau lebih font yang lebih aman di CSS kita sehingga halaman web kita akan terdegradasi dengan anggun di browser tanpa font pilihan Anda. Gambar 2.8 menunjukkan dua set aturan CSS yang diterapkan.

Gambar 2.8 Memilih font-family

Saat menyetel font, praktik terbaik adalah memberikan daftar font yang dapat dipilih browser, seperti dalam contoh ini:

```
font-family: arial, helvetica, sans-serif;
```

Anda dapat mengatur font yang disarankan untuk seluruh halaman HTML dengan menggunakan selector untuk elemen <body>, seperti dalam contoh ini:

```
body {
 font-family: arial, helvetica, sans-serif;
}
```

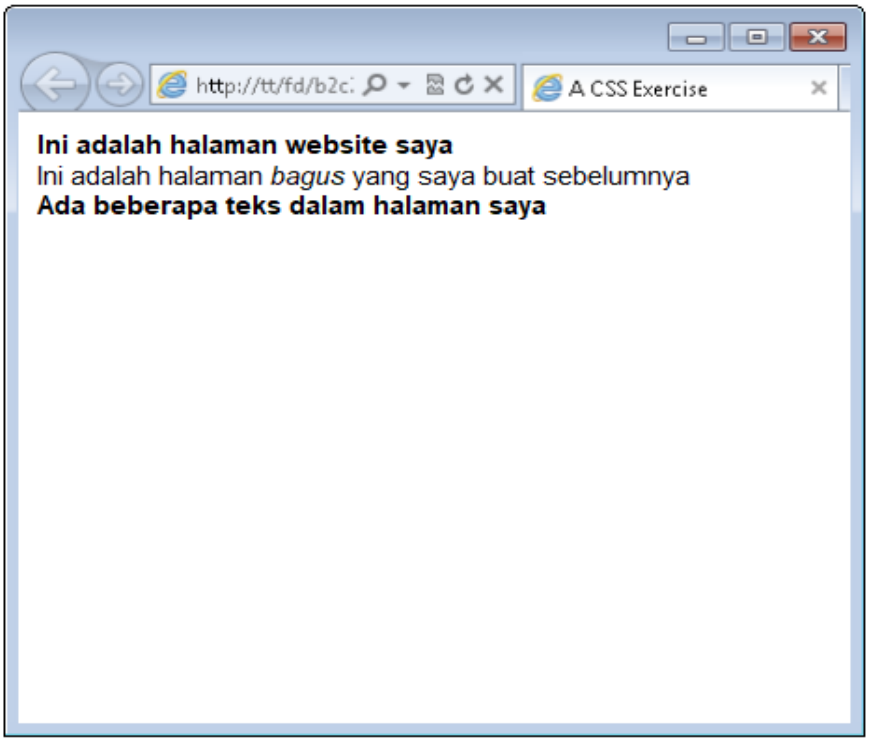
Setiap halaman yang menggunakan aturan CSS itu akan mencoba menampilkan teksnya terlebih dahulu dengan font Arial. Jika font tersebut tidak tersedia, font arial digunakan selanjutnya. Jika font tersebut tidak tersedia, maka font sans-serif digunakan. Jika tidak ada yang tersedia, maka browser memilih font untuk digunakan sendiri. Nilai umum untuk font-family adalah: arial, helvetica, sans-serif, “Arial Black”, Gadget, sans-serif, Georgia, serif, “Times New Roman”, Times, serif.

Daftar kode dibawah menunjukkan CSS yang kita lihat pada contoh sebelumnya. Cantuman ini menambahkan properti CSS font-family ke isi halaman, yang berarti bahwa pengaturan font-family ini akan diterapkan di seluruh halaman.

Daftar : Mengatur Nilai Font-Family dengan CSS

```
body {
 font-family: arial ,sans-serif;
}
.boldText {
 font-weight: bold;
}
span {
 font-style: italic;
}
```

Jika dilihat di browser menggunakan HTML yang sama dari latihan sebelumnya, hasilnya terlihat seperti gambar dibawah ini:



Gambar 2.9 Mengubah font family dengan CSS.

font-style

Dengan properti font-style kita dapat memilih untuk menampilkan font secara normal, miring, atau miring. Aturan berikut membuat tiga kelas (normal, miring, dan miring) yang dapat diterapkan ke elemen untuk membuat efek ini:

```
.normal { font-style:normal; }
.italic { font-style:italic; }
.oblique { font-style:oblique; }
```

Pengukuran

CSS mendukung berbagai unit pengukuran yang mengesankan, memungkinkan kita untuk menyesuaikan halaman web kita secara tepat dengan nilai tertentu, atau menurut dimensi relatif.

Font size

Mengatur ukuran huruf

Seberapa besar teks yang muncul di halaman web adalah ukuran fontnya. Kita dapat mengatur ukuran font menggunakan properti font-size CSS.

Piksel

Ukuran piksel bervariasi sesuai dengan dimensi dan kedalaman piksel monitor pengguna. Satu piksel sama dengan lebar/tinggi satu titik di layar, sehingga pengukuran ini paling cocok untuk monitor. Sebagai contoh:

```
.classname { margin:5px; }
```

Poin

Sebuah titik setara dengan ukuran 1/72 inci. Pengukuran berasal dari latar belakang desain cetak dan paling cocok untuk media itu, tetapi juga biasa digunakan pada monitor. Sebagai contoh:

```
.classname { font-size:14pt; }
```

Inci

Satu inci setara dengan 72 poin dan juga merupakan jenis pengukuran yang paling cocok untuk dicetak. Sebagai contoh:

```
.classname { width:3in; }
```

Sentimeter

Sentimeter adalah unit pengukuran lain yang paling cocok untuk dicetak. Satu sentimeter sedikit di atas 28 poin. Sebagai contoh:

```
.classname { height:2cm; }
```

Milimeter

Satu milimeter adalah 1/10 sentimeter (atau hampir 3 poin). Milimeter adalah ukuran lain yang paling cocok untuk dicetak. Sebagai contoh:

.classname { font-size:5mm; }

Foto

Pica adalah ukuran tipografi cetak lainnya, yang setara dengan 12 poin. Sebagai contoh:

.classname { font-size:1pc; }

Em

Em sama dengan ukuran font saat ini dan oleh karena itu merupakan salah satu pengukuran yang lebih berguna untuk CSS karena digunakan untuk menggambarkan dimensi relatif. Sebagai contoh:

.classname { font-size:2em; }

Ex

Mantan juga terkait dengan ukuran font saat ini; itu setara dengan tinggi huruf kecil x. Ini adalah unit pengukuran yang kurang populer yang paling sering digunakan sebagai perkiraan yang baik untuk membantu mengatur lebar kotak yang akan berisi beberapa teks. Sebagai contoh:

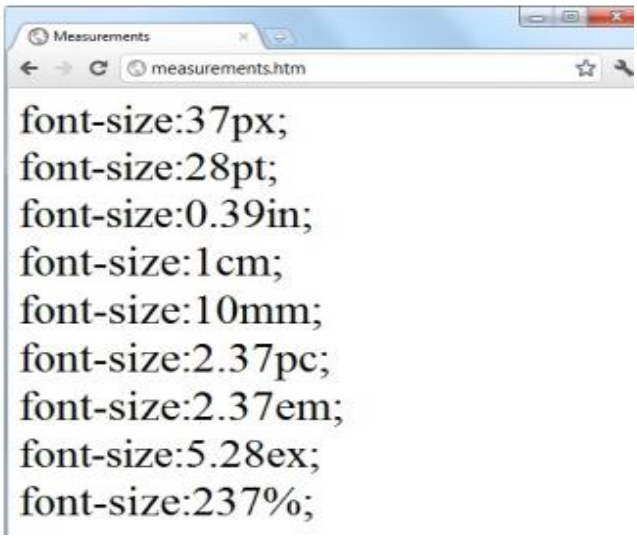
.classname { width:20ex; }

Persen

Unit ini terkait dengan em karena ukurannya tepat 100 kali lebih besar (bila digunakan pada font). Sedangkan 1 em sama dengan ukuran font saat ini, ukuran yang sama adalah 100 dalam persen. Saat tidak terkait dengan font, unit ini relatif terhadap ukuran wadah properti yang sedang diakses. Sebagai contoh:

.classname { height:120%; }

Gambar 2.10 menunjukkan masing-masing jenis pengukuran ini pada gilirannya digunakan untuk menampilkan teks dalam ukuran yang hampir sama.



Gambar 2.10 Pengukuran berbeda yang menampilkan hampir sama

Seperti yang dijelaskan di bagian sebelumnya tentang pengukuran, ada banyak cara kita dapat mengubah ukuran font. Tapi ini semua bermuara pada dua jenis utama: tetap dan relatif. Pengaturan tetap terlihat seperti aturan berikut, yang mengatur ukuran font paragraf default menjadi 14 poin:

```
p { font-size:14pt; }
```

Atau, kita mungkin ingin bekerja dengan ukuran font default saat ini, menggunakannya untuk menata berbagai jenis teks seperti heading. Dalam aturan berikut, ukuran relatif dari beberapa header ditentukan, dengan tag <h4> mulai 20% lebih besar dari default, dan dengan setiap ukuran lebih besar, 40% lebih besar dari yang sebelumnya:

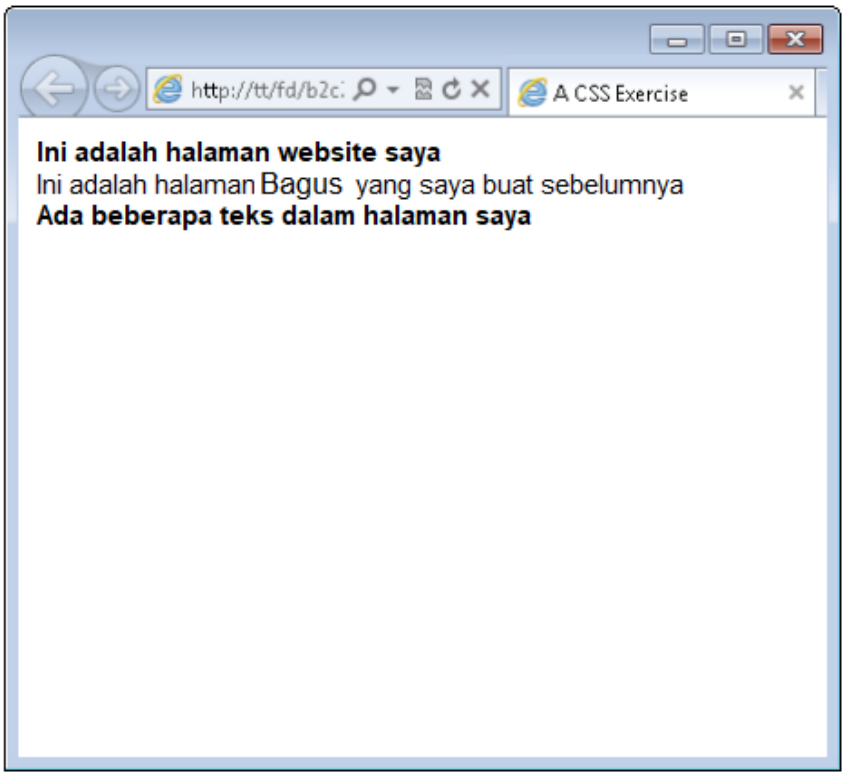
```
h1 { font-size:240%; }
h2 { font-size:200%; }
h3 { font-size:160%; }
h4 { font-size:120%; }
```

Gambar 2.11 menunjukkan pilihan ukuran font yang digunakan.

Gambar 2.11 Mengatur empat ukuran heading dan ukuran paragraf default

Metode untuk memilih ukuran font

Saat memilih metode ukuran font, kita dapat menggunakan persentase, em, poin, dan piksel. Poin dan piksel adalah ukuran tetap dan beberapa browser dapat mengalami kesulitan mengubah ukurannya, atau lebih tepatnya, browser tidak mengizinkan pengunjung untuk mengubah ukuran teks tanpa menggunakan alat zoom. Persentase dan em memungkinkan pengubahan ukuran.



Gambar 2.12 Mengubah ukuran font dengan CSS.

Ketika digabungkan dengan HTML dari latihan sebelumnya, kita mendapatkan halaman seperti itu pada Gambar dibawah ini. Perhatikan peningkatan ukuran font untuk kata yang paling bagus, berkat peningkatan ukuran yang disetel dengan em.

Saat menggunakan em untuk ukuran font, nilai em 1,0 sesuai dengan 100%. Oleh karena itu, 0.9em akan menjadi sekitar 90%, sedangkan 1.7em (seperti dalam contoh) pada dasarnya adalah 170%. Font yang diatur dengan piksel atau titik menggunakan singkatannya, seperti dalam contoh berikut:

```
font-size: 12px;
font-size: 12pt;
```

font-weight

Menggunakan properti font-weight kita dapat memilih seberapa berani untuk menampilkan font. Ini mendukung sejumlah nilai, tetapi nilai utama yang akan kita gunakan cenderung normal dan tebal, seperti ini:

```
.bold { font-weight:bold; }
```

Mengatur warna font

Sama seperti ukuran font yang dapat diatur, warna font juga dapat diatur. Kita harus berhati-hati dalam memilih warna font karena ini akan mempengaruhi kenyamanan mata dalam melihat font berwarna. Dalam hal ini, kita dapat menggunakan nama yang ramah untuk warna umum, seperti merah, biru, hijau, dan sebagainya, atau kita dapat menggunakan kode heksadesimal, atau singkatnya kode heksa. Kode hex adalah kode tiga sampai enam karakter yang sesuai dengan campuran warna *Red, Green dan Blue* (RGB) yang sesuai untuk

mendapatkan warna yang diinginkan. Tabel dibawah ini menunjukkan beberapa kode hex umum dan warna yang sesuai.

Tabel 2.1 kode warna untuk Hex

Kode	***Warna***
#FF0000	Merah
#00FF00	Hijau
#0000FF	Biru
#666666	Abu-abu gelap
#000000	Hitam
#FFFFFF	Putih
#EEEE00	Kuning
#FFA500	Oranye

Kode hex adalah cara yang lebih akurat dan disukai untuk mengatur warna dalam HTML tetapi sulit untuk diingat. Alat seperti Lab Warna Visibone di www.visibone.com/colorlab sangat penting untuk mendapatkan kode hex yang sesuai dengan warna yang ingin kita gunakan. Warna font diatur menggunakan properti CSS warna, seperti dalam contoh ini (yang merupakan kode untuk merah):

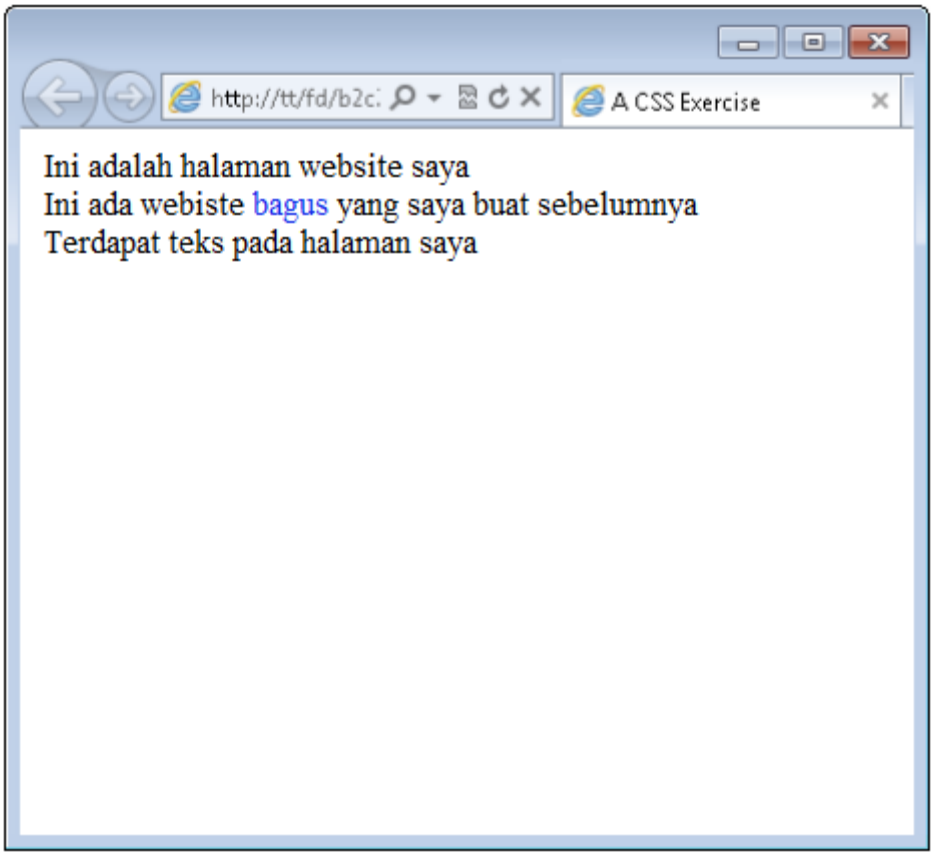
color: #FF0000;

Daftar dibawah ini menunjukkan CSS untuk mengubah warna elemen menjadi biru menggunakan kode hex:

Contoh Mewarnai Font Menggunakan CSS

```
span {
 color: #0000FF;
}
```

Jika dilihat di browser dengan HTML yang dibuat sebelumnya di bab ini, outputnya terlihat seperti Gambar dibawah ini. Perhatikan warna biru untuk kata “bagus” di halaman.



Gambar 2.13 Mengubah warna font menjadi biru

2.11 MENAMBAHKAN BORDER

Batas dapat membantu memberikan pemisahan visual antar elemen pada halaman. kita dapat menambahkan border di sekitar apa saja di HTML dan ada beberapa gaya border untuk dipilih. Border ditambahkan dengan properti CSS border. Saat membuat border dengan CSS, kita mengatur tiga hal:

- Ketebalan border
- Gaya Border
- Warna border

Ketiga item ini diatur dalam daftar, dipisahkan oleh spasi, seperti dalam contoh ini:

```
border: 1px solid black;
```

Dalam contoh ini, border akan dibuat dan tebalnya 1 piksel. Border akan padat dan akan berwarna hitam. Beberapa gaya batas umum ditunjukkan pada tabel berikut.

Tabel 2.2 border style di CSS

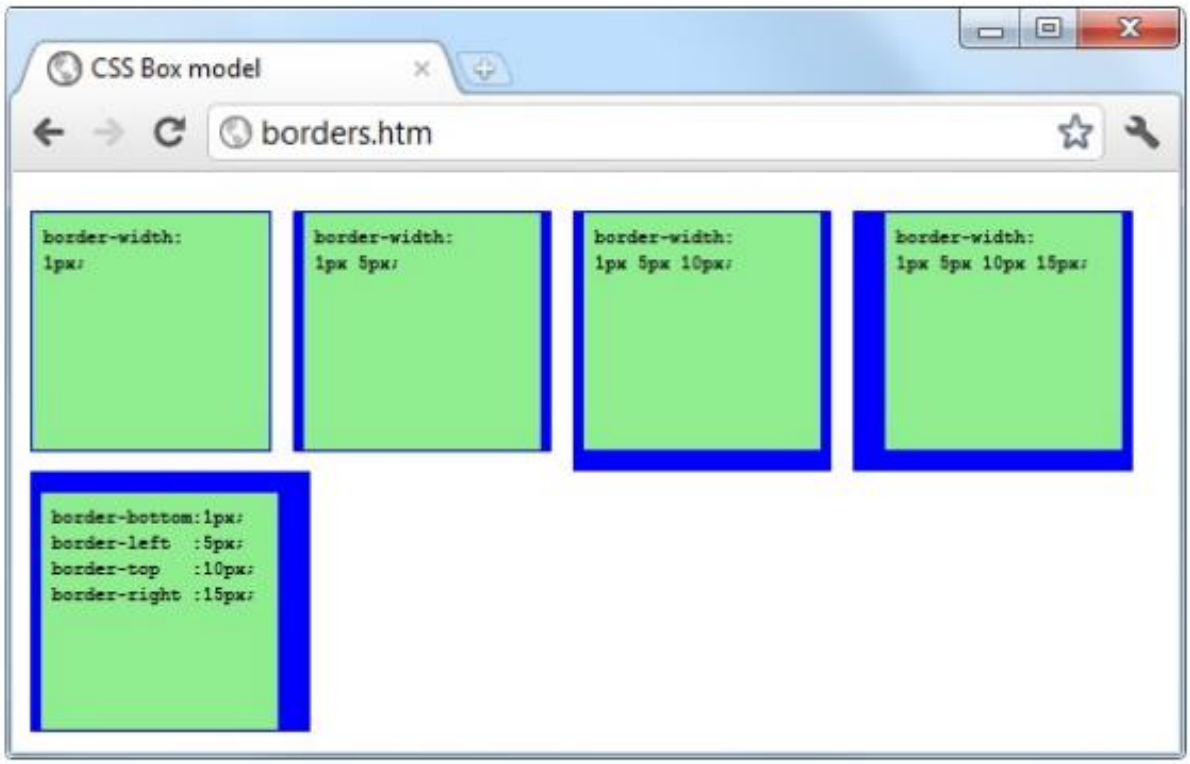
Style	***Deskripsi***
Solid	Garis padat
Titik-titik	Garis titik-titik
Tanda hubung	Garis dengan efek tanda hubung
Double	Dua garis padat

Menerapkan Border

Tingkat batas model kotak mirip dengan margin kecuali bahwa tidak ada keruntuhan. Ini adalah level berikutnya saat kita beralih ke model kotak. Properti utama yang digunakan untuk mengubah batas adalah batas, batas kiri, batas atas, batas kanan, dan batas bawah, dan masing-masing dapat memiliki subproperti lain yang ditambahkan sebagai sufiks, seperti -warna, -gaya, dan -lebar . Empat cara untuk mengakses pengaturan properti individual yang digunakan untuk properti margin juga berlaku dengan properti border-width, jadi semua yang berikut ini adalah aturan yang valid:

```
/* All borders
border-width:1px;
Top/bottom left/right
border-width:1px 5px;
Top left/right bottom
border-width:1px 5px 10px;
Top right bottom left */
border-width:1px 5px 10px 15px;
```

Gambar 2.14 menunjukkan masing-masing aturan ini diterapkan pada sekelompok elemen persegi. Bagian pertama, kita dapat dengan jelas melihat bahwa semua batas memiliki lebar 1 piksel. Elemen kedua, bagaimanapun, memiliki lebar batas atas dan bawah 1 piksel, sedangkan lebar sisinya masing-masing 5 piksel.



Gambar 2.14 Menerapkan nilai aturan batas panjang dan singkat

Elemen ketiga memiliki lebar atas 1 piksel, sisi-sisinya lebar 5 piksel, dan bagian bawahnya lebar 10 piksel. Elemen keempat memiliki lebar batas atas 1 piksel, lebar batas kanan 5 piksel, lebar batas bawah 10 piksel, dan lebar batas kiri 15 piksel. Elemen terakhir, di bawah yang sebelumnya, tidak menggunakan aturan singkatan; sebagai gantinya, masing-

masing lebar border diatur secara terpisah. Seperti yang kita lihat, dibutuhkan lebih banyak mengetik untuk mencapai hasil yang sama.

Saatnya latihan untuk membuat batas di sekitar beberapa elemen. Mengikuti langkah-langkah ini.

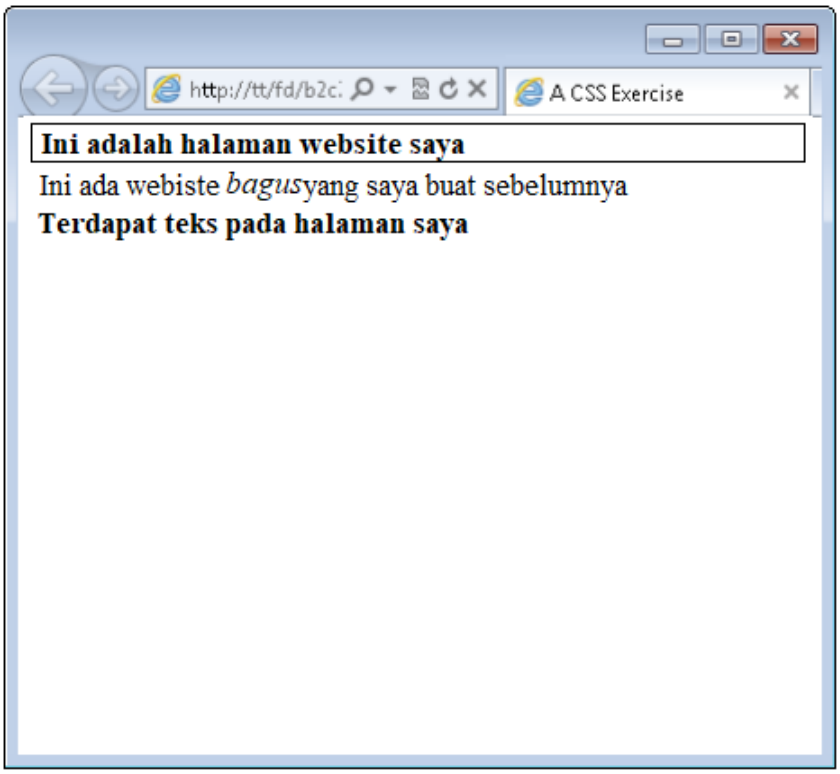
1. Buka text edirot
2. Verifikasi file HTML dari latihan sebelumnya.
   HTML dari latihan sebelumnya adalah titik awal untuk latihan ini. Jika milik kita tidak terlihat seperti ini, ubah agar terlihat seperti HTML ini. Bagi kita yang memiliki file ini persis seperti pada latihan sebelumnya, satu-satunya hal yang perlu kita lakukan adalah menambahkan kelas bernama addBorder di elemen pertama.

```
<!doctype html>
<html>
<head>
<title>A CSS Exercise</title>
<link rel=”stylesheet” type=”text/css” href=”style.
css”>
</head>
<body>
<div class=”boldText addBorder”>This is my web page.</
div>
<div>
 This is the <span>nicest</span> page I’ve made yet.
</div>
<div class=”boldText”>Here is some text on my page.</
div>
</body>
</html>
```

3. Save file HTML
   Simpan ini dengan format css-border.html dan tempatkan ini pada dokumen root Anda
4. Open file CSS
   File CSS harus berisi kelas yang disebut boldText dan aturan CSS yang mengubah semua elemen menjadi miring. Di dalam file CSS, tambahkan dan ubah CSS kita sehingga terlihat seperti berikut:

```
.boldText {
 font-weight: bold;
}
span {
 font-style: italic;
}
.addBorder {
 border: 3px double black;
}
```

5. Simpan file CSS
   Simpan file sebagai style.css di root dokumen Anda.
6. Lihat halaman di browser.
   Buka browser web kita dan arahkan ke http://localhost/css-border.html untuk melihat halaman. kita akan melihat halaman seperti gambar dibawah ini.



Gambar 2.15 Menambahkan batas ke elemen div

2.12 MENGELOLA GAYA TEKS

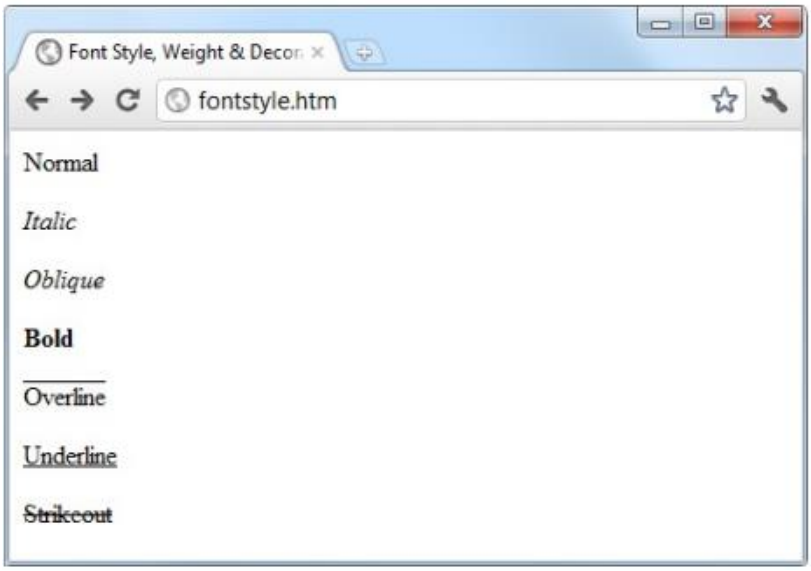
Terlepas dari font yang digunakan, kita dapat memodifikasi lebih lanjut cara teks ditampilkan dengan mengubah dekorasi, spasi, dan perataannya. Ada persilangan antara properti teks dan font, dalam efek seperti miring atau teks tebal dicapai melalui properti *font-style* dan font-weight, sementara yang lain seperti menggarisbawahi memerlukan properti teks-dekorasi.

Dekorasi

Dengan properti dekorasi teks, kita dapat menerapkan efek ke teks seperti garis bawah, garis tembus, garis atas, dan kedipan. Aturan berikut membuat kelas baru yang dipanggil over yang menerapkan overline ke teks (bobot garis over, under, dan through akan cocok dengan font):

```
.over { text-decoration:overline; }
```

Pada Gambar 2.16 Anda dapat melihat pilihan gaya font, bobot, dan dekorasi.



Gambar 2.16 Contoh gaya dan aturan dekorasi yang tersedia

Spasi

Sejumlah properti yang berbeda memungkinkan kita untuk mengubah spasi baris, kata, dan huruf. Misalnya, aturan berikut mengubah spasi baris untuk paragraf dengan memodifikasi properti tinggi baris menjadi 25% lebih besar, sedangkan properti spasi kata diatur ke 30 piksel, dan spasi huruf diatur ke 3 piksel:

```
p {
line-height :125%;
word-spacing :30px;
letter-spacing:3px;
}
```

Alignment

Empat jenis perataan teks tersedia di CSS: kiri, kanan, tengah, dan ratakan. Dalam aturan berikut, teks paragraf default diatur ke justifikasi penuh:

```
p { text-align:justify; }
```

Transformasi

Ada empat properti yang tersedia untuk mengubah teks: tidak ada, huruf besar, huruf besar, dan huruf kecil. Aturan berikut membuat kelas yang disebut atas yang akan memastikan bahwa semua teks ditampilkan dalam huruf besar saat digunakan:

```
.upper { text-transform:uppercase; }
```

Indentasi

Menggunakan properti indentasi teks, kita dapat membuat indentasi baris pertama dari blok teks dengan jumlah tertentu. Aturan berikut mengindentasi baris pertama setiap paragraf sebanyak 20 piksel, meskipun unit pengukuran yang berbeda atau peningkatan persen juga dapat diterapkan:

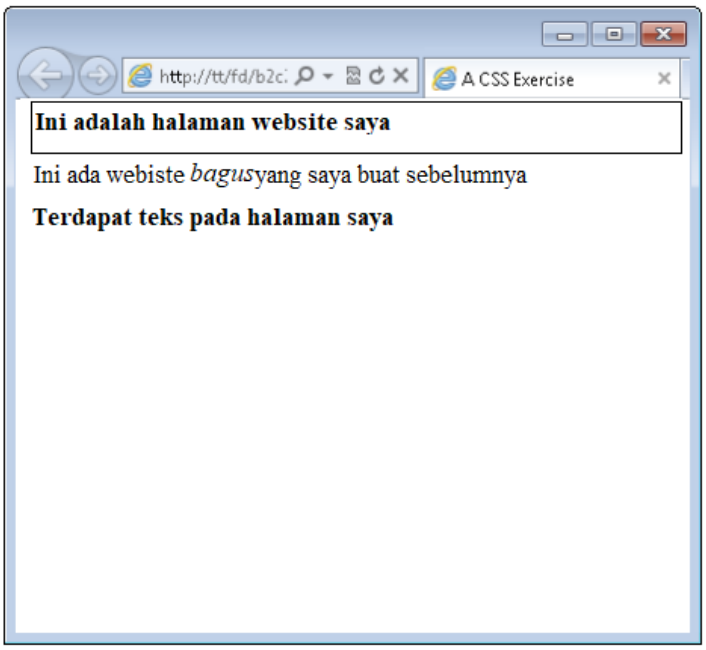
p { text-indent:20px; }
Pada Gambar 19-9, aturan berikut telah diterapkan pada bagian teks:
p { line-height :150%;
word-spacing :10px;
letter-spacing:1px;
}
.justify { text-align :justify; }
.uppercase { text-transform:uppercase; }
.indent { text-indent :20px; }

Gambar 2.17 Aturan indentasi, huruf besar, dan spasi diterapkan

Anda mungkin telah memperhatikan dalam latihan ini bahwa kita sekarang memiliki dua kelas di pertama di halaman. Itu adalah fitur hebat dari kelas karena kita dapat menggunakan lebih dari satu elemen untuk menggabungkannya. kita dapat bereksperimen dengan CSS dari latihan ini untuk menambahkan gaya batas yang berbeda ke elemen yang berbeda di halaman. kita mungkin tidak menyukai seberapa dekat teks dengan border pada Gambar dibawah. Kami yakin tidak. kita dapat mengubah ini dengan CSS. Properti padding CSS mengubah seberapa dekat teks akan sampai ke tepi bagian dalam border. Misalnya, kita dapat mengubah CSS untuk kelas addBorder agar terlihat seperti ini:

```
.addBorder {
 border: 3px double black;
 padding: 5px;
}
```

Ketika kita melakukannya, halaman yang dihasilkan akan terlihat seperti digambar dibawah ini.



Gambar 2.18 Menambahkan padding dalam kelas addBorder.

Padding dapat ditambahkan untuk memindahkan teks lebih jauh dari bordernya. Padding dapat diterapkan ke elemen apa pun, terlepas dari apakah elemen tersebut memiliki batas, untuk memindahkan konten elemen tersebut.

Saat kita menambahkan padding, konten elemen menjauh dari semua tepi. Namun, kita juga dapat menambahkan padding agar konten menjauh dari atas, bawah, kanan, atau kiri, atau kombinasi apa pun di dalamnya. Ini dilakukan dengan properti *padding-top, padding-bottom*, *padding-right,* dan *padding-left*, masing-masing. Ada metode pintasan untuk menyetel padding yang melihat semua padding didefinisikan pada satu baris. Pintasan itu tidak digunakan di sini, tetapi kita akan melihatnya di CSS orang lain. Di mana padding memindahkan elemen dari dalam, ada juga properti untuk memindahkan atau menggeser elemen dari luar. Elemen ini disebut margin, dan kita akan membahasnya nanti di bab ini ketika kita berbicara tentang membuat layout halaman.

2.13 MENYESUAIKAN PADDING

Level model kotak terdalam (selain konten elemen) adalah padding, yang diterapkan di dalam batas dan/atau margin apa pun. Properti utama yang digunakan untuk memodifikasi padding adalah padding, padding-left, padding-top, padding-right, dan padding-bottom. Empat cara mengakses pengaturan properti individual yang digunakan untuk properti margin dan border juga berlaku dengan properti padding, jadi semua berikut ini adalah aturan yang valid:

```
/* All padding
padding:1px;
Top/bottom and left/right
padding:1px 2px;
Top, left/right and bottom
padding:1px 2px 3px;
Top, right, bottom and left */
```

padding:1px 2px 3px 4px;

Gambar 2.19 menunjukkan aturan padding (ditampilkan dalam huruf tebal) pada Contoh 3 yang diterapkan ke beberapa teks dalam sel tabel (seperti yang didefinisikan oleh aturan display:table-cell;, yang membuat elemen enkapsulasi ditampilkan seperti a table cell), yang tidak diberi dimensi sehingga hanya akan membungkus teks sedekat mungkin. Akibatnya ada padding 10 piksel di atas elemen dalam, 20 piksel di sebelah kanan, 30 piksel di bawahnya, dan 40 piksel di sebelah kirinya.

Contoh 3. Menerapkan bantalan

```
<!DOCTYPE html>
<html>
<head>
<title>Padding</title>
<style>
#object1 {
border-style:solid;
border-width:1px;
background :orange;
color :darkred;
font-family :Arial;
font-size :12px;
text-align :justify;
display :table-cell;
width :148px;
padding :10px 20px 30px 40px; }
</style>
</head>
<body>
<div id='object1'>To be, or not to be that is the question:
Whether 'tis Nobler in the mind to suffer
The Slings and Arrows of outrageous Fortune,
Or to take Arms against a Sea of troubles,
And by opposing end them.</div>
</body>
</html>
```

Gambar 2.19 Menerapkan nilai padding yang berbeda ke objek

Isi Objek

Jauh di dalam level model kotak, di tengahnya, terletak sebuah elemen yang dapat ditata dengan semua cara yang dibahas dalam bab ini, dan yang dapat (dan biasanya akan) berisi sub-elemen lebih lanjut, yang pada gilirannya dapat berisi sub-sub- elemen, dan seterusnya, masing-masing dengan pengaturan gaya dan model kotaknya sendiri.

2.14 SELEKTOR CSS

Cara kita mengakses satu atau lebih elemen disebut seleksi, dan bagian dari aturan CSS yang melakukan ini dikenal sebagai pemilih. Seperti yang kita duga, ada banyak jenis pemilih.

Selektor Jenis

Selektor jenis bekerja pada jenis elemen HTML seperti atau <i>. Misalnya, aturan berikut akan memastikan bahwa semua teks dalam tag ... sepenuhnya dibenarkan:

```
p { text-align:justify; }
```

Selektor Keturunan

Selektor turunan memungkinkan kita menerapkan gaya ke elemen yang ada di dalam elemen lain. Misalnya, aturan berikut menyetel semua teks dalam tag ... menjadi merah, tetapi hanya jika teks tersebut muncul di dalam tag ... (seperti ini: Hello di sana):

```
p b { color:red; }
```

Selektor turunan dapat terus bersarang tanpa batas, jadi berikut ini adalah aturan yang benar-benar valid untuk membuat teks menjadi biru di dalam teks tebal, di dalam elemen daftar dari daftar yang tidak diurutkan:

```
ul li b { color:blue; }
```

Sebagai contoh praktis, misalkan kita ingin menggunakan sistem penomoran yang berbeda untuk daftar terurut yang bersarang di dalam daftar terurut lainnya. kita dapat mencapai ini dengan cara berikut, yang akan menggantikan penomoran numerik default (mulai dari 1) dengan huruf kecil (mulai dari a):

```
<!DOCTYPE html>
<html>
<head>
<style>
ol ol { list-style-type:lower-alpha; }
</style>
</head>
<body>
<ol>
li>One</li>
<li>Two</li>
<li>Three
<ol>
<li>One</li>
<li>Two</li>
<li>Three</li>
</ol>
</li>
</ol>
</body>
</html>
```

Hasil pemuatan HTML ini ke browser web adalah sebagai berikut, di mana kita dapat melihat bahwa elemen daftar kedua ditampilkan secara berbeda:

1. One
2. Two
3. Three
   a. One
   b. Two
   c. Three

Selector Child

Selektor turunan mirip dengan selektor turunan tetapi lebih membatasi kapan gaya akan diterapkan, dengan memilih hanya elemen yang merupakan anak langsung dari elemen lain. Misalnya, kode berikut menggunakan selektor turunan yang akan mengubah teks tebal apa pun dalam paragraf menjadi merah, meskipun teks tebal itu sendiri berada di dalam huruf miring (seperti ini <i>Hello di sana< /i>):

```
p b { color:red; }
```

Dalam contoh ini, kata Hello ditampilkan dengan warna merah. Namun, ketika jenis perilaku yang lebih umum ini tidak diperlukan, selektor anak dapat digunakan untuk mempersempit

cakupan pemilih. Misalnya, selektor anak berikut akan menyetel teks tebal menjadi merah hanya jika elemen tersebut adalah anak langsung dari sebuah paragraf, dan tidak berisi elemen lain:

```
p > b { color:red; }
```

Sekarang kata Hello tidak akan berubah warna karena bukan turunan langsung dari paragraf. Sebagai contoh praktis, misalkan kita hanya ingin memberanikan elemen <li> yang merupakan turunan langsung dari elemen <ol>. kita dapat mencapai ini sebagai berikut, di mana elemen <li> yang merupakan anak langsung dari elemen <ul> tidak dikuatkan:

```
<!DOCTYPE html>
<html>
<head>
<style>
ol > li { font-weight:bold; }
</style>
</head>
<body>
<ol>
<li>One</li>
<li>Two</li>
<li>Three</li>
</ol>
<ul>
<li>One</li>
<li>Two</li>
<li>Three</li>
</ul>
</body>
</html>
```

Hasil loading HTML ini ke dalam browser adalah sebagai berikut:

1. One
2. Two
3. Three

- One
- Two
- Three

Selektor ID

Jika kita memberi elemen nama ID (seperti ini:) maka kita dapat langsung mengaksesnya dari CSS dengan cara berikut, yang mengubah semua teks dalam elemen menjadi miring:

```
#mydiv { font-style:italic; }
```

ID hanya dapat digunakan sekali dalam dokumen, jadi hanya kemunculan pertama yang ditemukan yang akan menerima nilai properti baru yang ditetapkan oleh aturan CSS. Tetapi di CSS kita dapat langsung mereferensikan ID apa pun yang memiliki nama yang sama, selama ID tersebut muncul dalam tipe elemen yang berbeda, seperti ini:

Hello Hello

Karena ID biasanya hanya berlaku untuk elemen unik, aturan berikut akan menerapkan garis bawah hanya untuk kemunculan pertama myid:

#myid { text-decoration:underline; }

Namun, kita dapat memastikan bahwa CSS menerapkan aturan untuk kedua kejadian seperti ini:

span#myid { text-decoration:underline; }
Atau lebih ringkasnya seperti ini (lihat Memilih berdasarkan Grup):
div#myid { text-decoration:underline; }
span#myid, div#myid { text-decoration:underline; }

Catatan: Saya tidak menyarankan menggunakan bentuk pemilihan ini karena JavaScript apa pun yang juga harus mengakses elemen-elemen ini tidak dapat dengan mudah melakukannya karena fungsi getElementByID() yang umum digunakan hanya akan mengembalikan kemunculan pertama. Untuk mereferensikan contoh lain, sebuah program harus mencari seluruh daftar elemen dalam dokumen—tugas yang lebih sulit untuk dilakukan. Jadi umumnya lebih baik untuk selalu menggunakan nama ID yang unik.

Selektor Kelas

Bila ada sejumlah elemen di halaman yang ingin kita bagikan gaya yang sama, kita dapat menetapkan semua nama kelas yang sama (seperti ini:); kemudian, buat aturan tunggal untuk memodifikasi semua elemen tersebut sekaligus, seperti pada aturan berikut, yang membuat offset margin kiri 10 piksel untuk semua elemen menggunakan kelas:

.myclass { margin-left:10px; }

Di browser modern, kita dapat membuat elemen HTML menggunakan lebih dari satu kelas dengan memisahkan nama kelas dengan spasi, seperti ini: . Namun, ingat bahwa beberapa browser yang sangat lama hanya mengizinkan satu nama kelas dalam argumen kelas. kita dapat mempersempit cakupan tindakan kelas dengan menentukan jenis elemen yang harus diterapkan. Misalnya, aturan berikut ini hanya berlaku untuk paragraf yang menggunakan kelas utama:

p.main { text-indent:30px; }

Dalam contoh ini, hanya paragraf yang menggunakan kelas utama (seperti ini: <p class="main">) yang akan menerima nilai properti baru. Jenis elemen lain yang mungkin

mencoba menggunakan kelas (seperti) tidak akan terpengaruh oleh aturan ini.

Atribut Selektor

Banyak tag HTML mendukung atribut, dan menggunakan jenis selektor ini dapat menyelamatkan kita dari keharusan menggunakan ID dan kelas untuk merujuknya. Misalnya, kita dapat langsung mereferensikan atribut dengan cara berikut, yang menyetel semua elemen dengan atribut type="submit" ke lebar 100 piksel:

```
[type="submit"] { width:100px; }
```

Jika kita ingin mempersempit cakupan pemilih, misalnya, hanya membentuk elemen input dengan tipe atribut tersebut, kita dapat menggunakan aturan berikut sebagai gantinya:

```
form input[type="submit"] { width:100px; }
```

Catatan: Selektor atribut juga berfungsi pada ID dan kelas sehingga, misalnya, [class~="classname"] berfungsi persis seperti selektor kelas .classname (kecuali bahwa yang terakhir memiliki prioritas lebih tinggi). Demikian juga, [id="idname"] sama dengan menggunakan selektor ID #idname. Selektor kelas dan ID diawali dengan # dan . karena itu dapat dilihat sebagai singkatan untuk penyeleksi atribut, tetapi dengan prioritas yang lebih tinggi. Operator ~= cocok dengan atribut meskipun itu adalah salah satu dari grup atribut yang dipisahkan spasi.

Selektor Universal

Wildcard * atau selektor universal cocok dengan elemen apa pun, jadi aturan berikut akan membuat dokumen berantakan total dengan memberikan batas hijau ke semua elemen -elemennya:

```
* { border:1px solid green; }
```

Oleh karena itu, kecil kemungkinan kita akan menggunakan * sendiri, tetapi sebagai bagian dari aturan majemuk, itu bisa sangat kuat. Misalnya, aturan berikut akan menerapkan gaya yang sama seperti sebelumnya, tetapi hanya untuk semua paragraf yang merupakan sub-elemen dari elemen dengan kotak ID, dan hanya selama mereka bukan turunan langsung:

```
#boxout * p {border:1px solid green; }
```

Mari kita lihat apa yang terjadi di sini. Selektor pertama setelah #boxout adalah simbol *, jadi ini merujuk ke elemen apa pun di dalam objek boxout. Selektor p berikut kemudian mempersempit fokus pemilihan dengan mengubah selektor untuk diterapkan hanya pada paragraf (sebagaimana didefinisikan oleh p) yang merupakan sub-elemen dari elemen yang dikembalikan oleh selektor *. Oleh karena itu, aturan CSS ini melakukan tindakan berikut (di mana saya menggunakan istilah objek dan elemen secara bergantian):

1. Temukan objek dengan ID boxout.
2. Temukan semua sub-elemen objek yang dikembalikan pada langkah 1.

3. Temukan semua p sub-elemen dari objek yang dikembalikan pada langkah 2 dan, karena ini adalah selektor terakhir dalam grup, temukan juga semua p sub-dan sub-elemen (dan seterusnya) dari objek yang dikembalikan pada langkah 2.
4. Terapkan gaya dalam karakter {and } ke objek yang dikembalikan pada langkah 3.

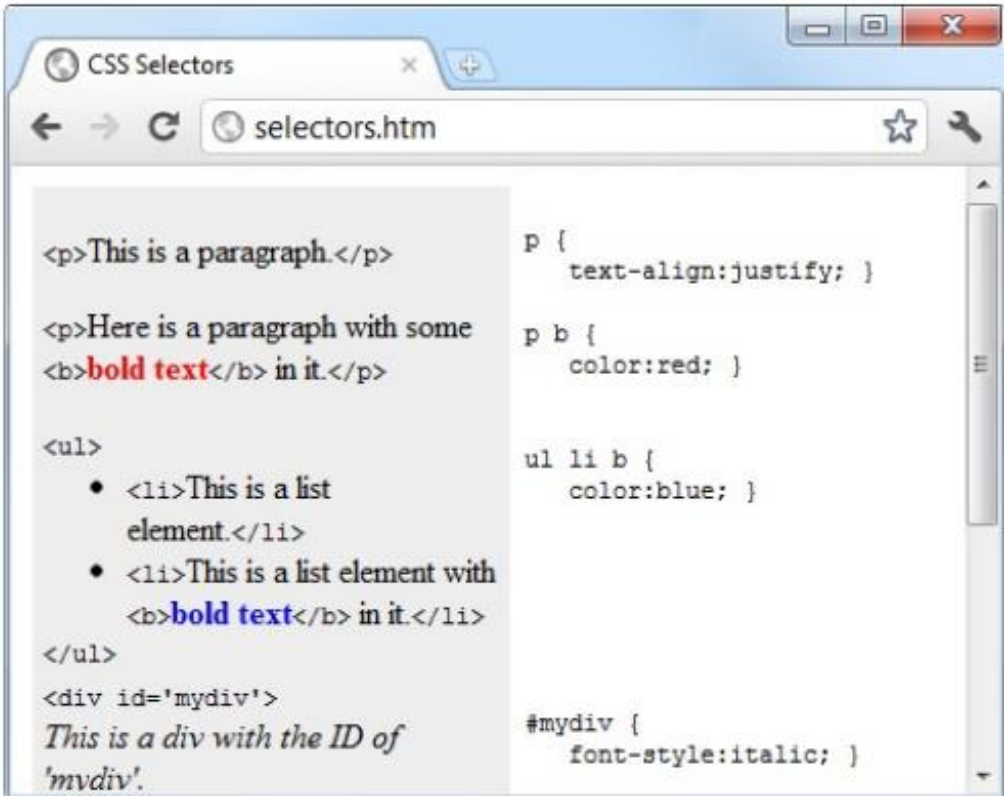
Hasil bersih dari ini adalah bahwa batas hijau hanya diterapkan pada paragraf yang adalah cucu (atau cicit, dll.) dari elemen utama.

2.15 MEMILIH BERDASARKAN GRUP

Menggunakan CSS kita dapat menerapkan aturan ke lebih dari satu elemen, kelas, atau jenis selektor lainnya secara bersamaan dengan memisahkan selektor dengan koma. Jadi, misalnya, aturan berikut akan menempatkan garis oranye putus-putus di bawah semua paragraf, elemen dengan ID idname, dan semua elemen yang menggunakan class classname:

```
p, #idname, .classname { border-bottom:1px  dotted orange; }
```

Gambar 2.20 menunjukkan berbagai selektor yang digunakan, dengan aturan yang diterapkan bersamanya.



Gambar 2.20 Beberapa aturan HTML dan CSS yang digunakan olehnya

Mengubah List style

Kita bisa. Gaya peluru untuk daftar ditentukan oleh properti CSS list-style-type. Ada banyak nilai untuk properti list-style-type. Tabel 2-3 menunjukkan beberapa yang umum.

Tabel 2.3 List Style

Style	***Deskripsi***
circle	Menyediakan peluru tipe lingkaran
decimal	Gaya default untuk daftar , angka sederhana.

disc	Gaya default untuk daftar , gaya lingkaran yang diisi.
none	Menghapus gaya sepenuhnya untuk daftar.
square	Sebuah peluru persegi.
upper-roman	Angka Romawi huruf besar, seperti dalam garis besar

Menggunakan basis angka yang berbeda

Di mana ada lebih dari sembilan tipe dalam suatu angka, kita harus bekerja di basis angka yang lebih tinggi. Misalnya, kita tidak dapat mengubah bilangan majemuk [11,7,19] menjadi desimal hanya dengan menggabungkan ketiga bagian tersebut. Sebagai gantinya, kita dapat mengonversi angka ke basis yang lebih tinggi seperti basis 20 (atau lebih tinggi jika ada lebih dari 19 jenis apa pun). Untuk melakukan ini, kalikan tiga bagian dan tambahkan hasilnya seperti ini, dimulai dengan angka paling kanan dan bekerja di kiri:

20 × 19 = 380
20×20 × 7 = 2800
20×20×20 × 11 = 88000
Total in decimal = 91180

Di sebelah kiri, ganti nilai 20 dengan basis yang kita gunakan. Setelah semua bilangan majemuk dari seperangkat aturan dikonversi dari basis ini ke desimal, mudah untuk menentukan spesifisitas, dan oleh karena itu didahulukan, dari masing-masing. Untungnya, prosesor CSS menangani semua ini untuk Anda, tetapi mengetahui cara kerjanya membantu kita membuat aturan dengan benar dan memahami prioritas apa yang akan mereka miliki.

Mengubah gaya poin

Cara terbaik untuk melihat gaya ini beraksi adalah dengan mencobanya.

1. Buka teks Editor
2. Ubah atau buat ul.html.
   Di dalam file, gunakan HTML berikut. Jika menggunakan ul.html, kita hanya perlu menambahkan elemen <link> untuk memasukkan file CSS.

```
<!doctype html>
<html>
<head>
<title>An unordered list</title>
<link rel=”stylesheet” type=”text/css” href=”ul.css”>
</head>
<body>
<ul>
 <li>Pine</li>
 <li>Oak</li>
 <li>Elm</li>
</ul>
</body>
</html>
```

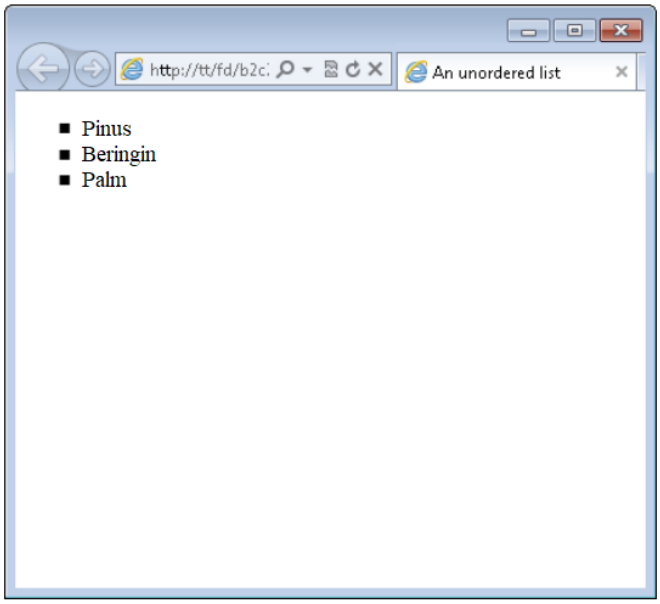
3. Simpan file.

Simpan file sebagai ul.html di root dokumen Anda.

4. Buat file baru.
   Buat dokumen teks kosong baru menggunakan text editor.
5. Tempatkan CSS berikut di dokumen baru:

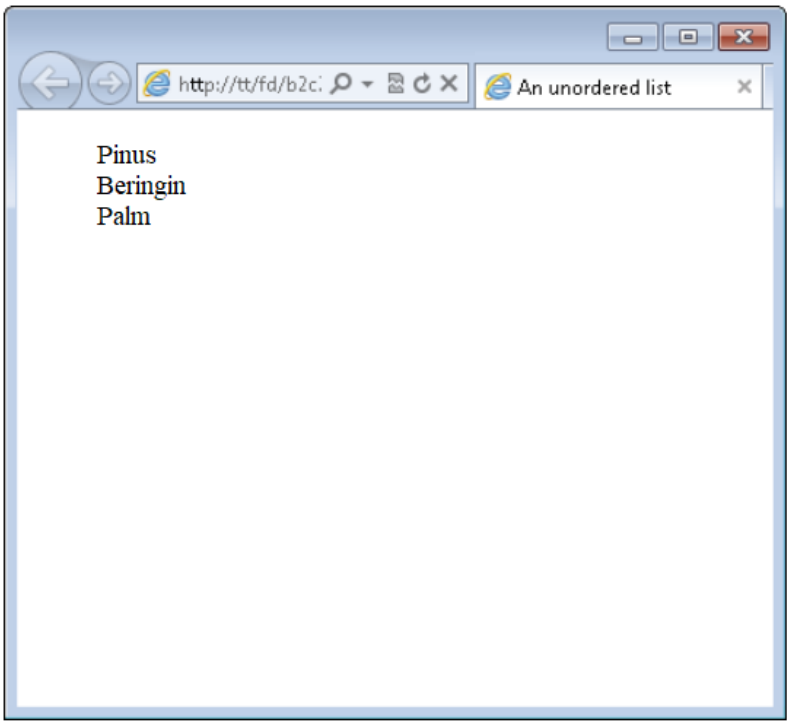
```
ul {
 list-style-type: square;
}
```

6. Simpan file CSSnya.
   Simpan file sebagai ul.css di dokumen root Anda.
7. Buka browser web kita dan lihat halamannya.
   Di browser web, ketik http://localhost/ul.html ke bilah alamat dan tekan Enter. Selanjutnya, kita akan melihat halaman seperti pada gambar dibawah ini.



Gambar 2.21 Mengubah list style.

Dengan properti list style kita dapat bereksperimen untuk menambah atau mengubah gaya peluru.



Gambar 2.22 Menghapus poin dari daftar HTML.

Menghapus poin

Tampilan umum untuk daftar di halaman web tidak menggunakan poin sama sekali. Efek ini dibuat dengan menyetel nilai tipe list style ke none, seperti dalam contoh ini, yang dapat digunakan dalam file ul.css yang baru saja di buat.

```
ul {
 list-style-type: none;
}
```

Ketika diterapkan ke halaman yang dibuat pada latihan sebelumnya, hasilnya terlihat seperti gabar dibawah ini. kita menerapkan properti tipe-daftar ke <ul> atau <ol> dan bukan ke item daftar individual (elemen).

Beberapa aturan lebih setara dari yang lain

Jika dua atau lebih aturan gaya sama persis, hanya aturan yang paling baru diproses yang akan diprioritaskan. Namun, kita dapat memaksakan aturan ke prioritas yang lebih tinggi daripada aturan lain yang setara menggunakan deklarasi !important, seperti ini:

```
p { color:#ff0000 !important; }
```

Saat kita melakukan ini, semua pengaturan sebelumnya yang setara akan ditimpa (termasuk yang menggunakan !important) dan semua aturan yang setara yang diproses nanti akan diabaikan. Jadi, misalnya, yang kedua dari dua aturan berikut biasanya akan didahulukan, tetapi karena penggunaan !important dalam penetapan sebelumnya, yang kedua diabaikan:

```
p { color:#ff0000 !important; }
p { color:#ffff00 }
```

Catatan: Style sheet pengguna dapat dibuat untuk menentukan gaya browser default, dan mereka dapat menggunakan deklarasi !important, dalam hal ini pengaturan user style akan didahulukan dari properti yang sama yang ditentukan di halaman web saat ini. Namun, pada browser yang sangat lama menggunakan CSS 1, fitur ini tidak didukung

2.16 MENAMBAHKAN BACKGROUND

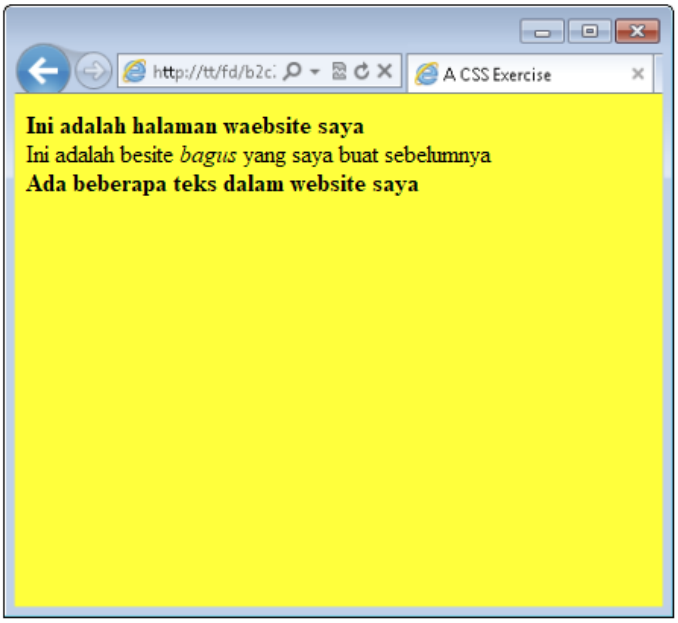
Halaman yang telah kita buat sejauh ini memiliki latar belakang putih, atau lebih tepatnya, mereka memiliki latar belakang default yang dipilih oleh browser. Di browser web versi lama, warna latar belakang itu abu-abu. kita dapat mengubah warna latar belakang menggunakan CSS, atau menggunakan gambar latar belakang. Warna latar belakang dan gambar latar belakang dapat diterapkan ke seluruh halaman atau ke elemen individual. Mengubah warna latar belakang pada elemen individual membantu menambahkan sorotan dan warna ke area halaman tertentu.

Mengubah warna background

Warna latar belakang elemen HTML diubah dengan properti CSS warna latar belakang. Warna latar belakang menggunakan sintaks yang sama (kode hex) sebagai warna font; lihat pembahasan warna font sebelumnya dalam bab ini untuk melihat kode hex untuk warna umum. Berikut adalah contoh yang mengubah warna latar belakang seluruh halaman:

```
body {
 background-color: #FFFF00;
}
```

Gambar dibawah ini menunjukkan halaman yang dihasilkan. Perhatikan bahwa warna kuning tidak akan muncul dengan baik di dalam buku, tetapi ada di sana!



Gambar 2.23 Menambahkan warna latar belakang kuning ke halaman

Seperti yang dinyatakan sebelumnya, elemen individual juga dapat diubah dan kita dapat menggunakan semua selector CSS yang berbeda untuk memfokuskan perubahan warna itu ke kelas, ke elemen individual (menggunakan id), atau ke semua elemen dengan

menggunakan nama elemen. Misalnya, mengubah semua elemen menjadi kuning terlihat seperti ini:

```
div {
 background-color: #FFFF00;
}
```

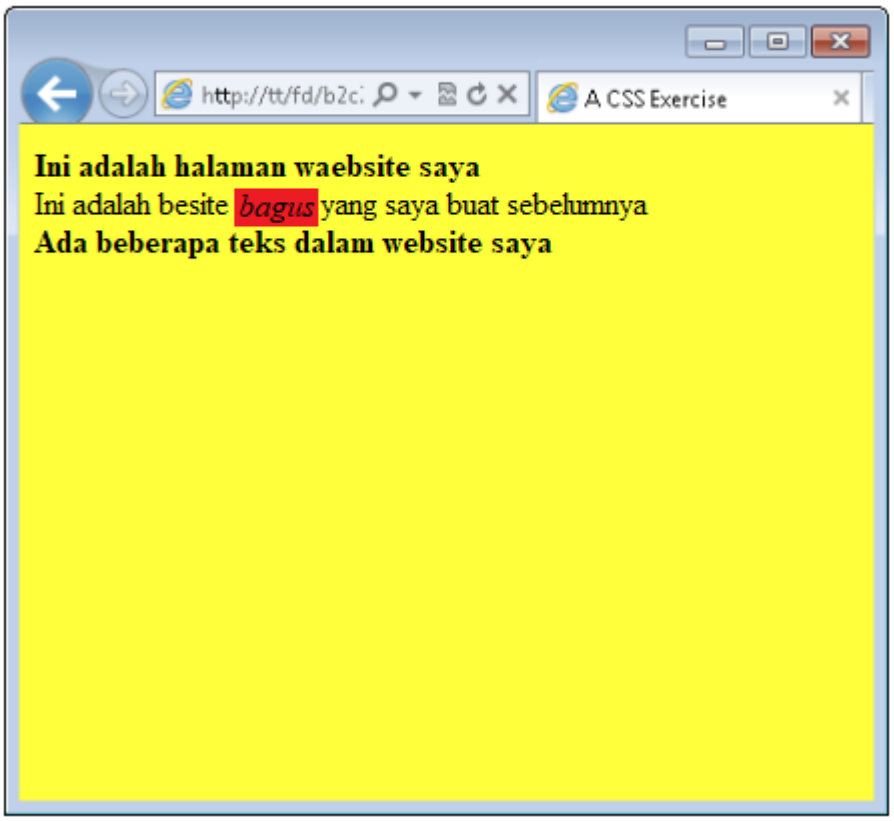
Anda juga dapat menggunakan CSS untuk menargetkan elemen berdasarkan hierarkinya; dengan kata lain, kita dapat menargetkan elemen ketika mereka muncul sebagai anak dari elemen lain. Ini membutuhkan contoh. Banyak contoh dalam buku ini menggunakan HTML yang mirip dengan yang ditunjukkan pada contoh.

Contoh 4 HTML Digunakan dalam Beberapa Contoh

```
<!doctype html>
<html>
<head>
<title>A CSS Exercise</title>
<link rel="stylesheet" type="text/css" href="style8.css">
</head>
<body>
<div class="boldText">This is my web page.</div>
<div>
 This is the <span>nicest</span> page I've made yet.
</div>
<div class="boldText">Here is some text on my page.</div>
</body>
</html>
```

Fokus pada elemen di dalam kedua dalam HTML ini. Bisa dibilang elemen adalah anak dari . Dengan menggunakan CSS, kita dapat menargetkan span ini berdasarkan posisinya sebagai anak dari . Ini berguna jika kita ingin menerapkan gaya tertentu ke semua elemen dari tipe tertentu tetapi kita tidak (atau tidak bisa) menambahkan kelas ke elemen tersebut. Misalnya, jika kita ingin membuat semua elemen yang muncul dalam memiliki latar belakang merah, kita dapat melakukannya dengan CSS ini:

```
div span {
 background-color: #FF0000;
}
```



Gambar 2.24 Menargetkan elemen untuk menerapkan aturan CSS.

Penargetan CSS ini dapat diterapkan dengan cara apa pun yang kita inginkan, apakah itu menargetkan ID tertentu, kelas tertentu, atau elemen tertentu, seperti contoh. kita dapat membuat kombinasi hierarki CSS yang kuat (dan terkadang membingungkan) untuk menerapkan aturan CSS. kita dapat menggunakan penargetan CSS ini untuk menerapkan aturan CSS apa pun, bukan hanya warna latar belakang.

2.17 WARNA CSS

Anda dapat menerapkan warna ke latar depan dan latar belakang teks dan objek menggunakan properti warna dan warna latar (atau dengan menyediakan satu argumen ke properti latar belakang). Warna yang ditentukan dapat berupa salah satu warna bernama (seperti merah atau biru), warna yang dibuat dari triplet RGB heksadesimal (seperti #ff0000 atau #0000ff), atau warna yang dibuat menggunakan fungsi CSS rgb. 16 nama warna standar seperti yang didefinisikan oleh organisasi standar W3C adalah: aqua, hitam, biru, fuchsia, abu-abu, hijau, kapur, merah marun, biru tua, zaitun, ungu, merah, perak, teal, putih, dan kuning. Aturan berikut menggunakan salah satu dari nama ini untuk mengatur warna latar belakang untuk objek dengan ID objek:

```
#object { background-color:silver; }
```

Dalam aturan ini, warna latar depan teks di semua elemen diatur ke kuning (karena pada tampilan komputer, tingkat heksadesimal ff merah, ditambah ff hijau, ditambah 00 biru menghasilkan warna kuning):

```
div { color:#ffff00; }
```

Atau, jika kita tidak ingin bekerja dalam heksadesimal, kita dapat menentukan triplet warna kita menggunakan fungsi rgb, seperti dalam aturan berikut, yang mengubah warna latar belakang dokumen saat ini menjadi aqua:

```
body { background-color:rgb(0, 255, 255); }
```

Catatan: Jika kita memilih untuk tidak bekerja dalam rentang 256 level per warna, kita dapat menggunakan persentase dalam fungsi rgb sebagai gantinya, dengan nilai dari 0 hingga 100 mulai dari terendah (0) hingga tertinggi (100) jumlah primer warna, seperti ini: rgb (58%, 95%, 74%). kita juga dapat menggunakan nilai floating point untuk kontrol warna yang lebih halus, seperti ini: rgb(23,4%, 67,6%, 15,5%).

String Warna Pendek

Ada juga bentuk pendek dari string digit hex di mana hanya yang pertama dari setiap pasangan 2-byte yang digunakan untuk setiap warna. Misalnya, alih-alih menetapkan warna #fe4692, kita malah menggunakan #f49, menghilangkan digit heksagonal kedua dari setiap pasangan, yang setara dengan nilai warna #ff4499. Ini menghasilkan warna yang hampir sama dan berguna di mana warna yang tepat tidak diperlukan. Perbedaan antara string enam digit dan tiga digit adalah bahwa yang pertama mendukung 16 juta warna berbeda, sedangkan yang kedua mendukung empat ribu. Di mana pun kita ingin menggunakan warna seperti #883366, ini adalah padanan langsung dari #836 (karena digit berulang tersirat oleh versi yang lebih pendek), dan kita dapat menggunakan salah satu string untuk membuat warna yang sama persis.

Menambahkan gambar background

Gambar background adalah cara yang baik untuk membuat halaman HTML agar terlihat bagus. Dalam penggunaan gambar background, kita dapat membuat efek gradien, di mana satu bagian halaman berwarna solid dan warnanya memudar atau menjadi lebih terang saat membentang ke sisi lain. Gambar background muncul di belakang elemen lainnya. Ini berarti kita dapat melapisi semua HTML Anda, termasuk gambar lain, di atas gambar latar belakang. kita dapat menemukan banyak gambar gratis melalui *Creative Commons*. Lihat http://search.creativecommons.org untuk informasi lebih lanjut. Pastikan untuk memilih gambar yang masih memungkinkan teks dapat dibaca di halaman; teks hitam pada gambar gelap tidak cocok. Gambar latar belakang ditambahkan dengan properti CSS gambar latar, seperti yang dijelaskan di sini dan di bagian berikut.

```
background-image:url(“myImage.jpg”);
```

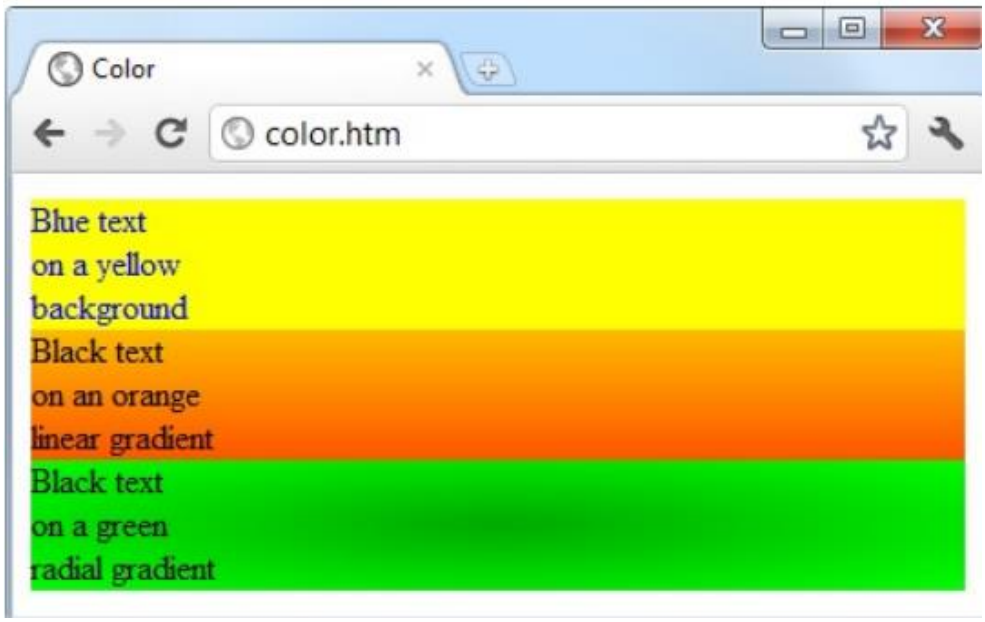
Gradien

Di tempat menggunakan warna latar belakang yang solid, kita dapat memilih untuk menerapkan gradien, yang kemudian akan secara otomatis mengalir dari warna awal yang diberikan ke warna akhir pilihan Anda. Paling baik digunakan bersama dengan aturan warna sederhana sehingga browser yang tidak mendukung gradien setidaknya akan menampilkan warna solid. Contoh 5 menggunakan aturan untuk menampilkan gradien oranye (atau hanya oranye polos pada browser yang tidak mendukung) seperti yang ditunjukkan di bagian tengah Gambar 2.25.

Contoh 5. Membuat gradien linier

```
<!DOCTYPE html>
<html>
<head>
<title>Creating a linear gradient</title>
<style>
.orangegrad {
background:orange;
background:linear-gradient(top, #fb0, #f50);
background:-moz-linear-gradient(top, #fb0, #f50);
background:-webkit-linear-gradient(top, #fb0, #f50);
background:-o-linear-gradient(top, #fb0, #f50);
background:-ms-linear-gradient(top, #fb0, #f50); }
</style>
</head>
<body>
<div class='orangegrad'>Black text<br>
on an orange<br>linear gradient</div>
</body>
</html>
```

Catatan: Seperti yang ditunjukkan pada contoh sebelumnya, banyak aturan CSS memerlukan awalan khusus browser seperti -moz-, -webkit-, -o-, dan -ms- (untuk browser berbasis Mozilla seperti Firefox; browser berbasis WebKit seperti seperti Apple Safari, Google Chrome, dan browser iOS dan Android; dan browser Opera dan Microsoft). http://caniuse.com mencantumkan aturan dan atribut CSS utama, dan apakah versi khusus browser diperlukan.



Gambar 2.25 Warna latar belakang yang solid, gradien linier, dan gradien radial

Untuk membuat gradien, pilih di mana itu akan dimulai dari atas, bawah, kiri, kanan, dan tengah (atau kombinasi apa pun, seperti kiri atas atau kanan tengah), masukkan warna awal dan akhir yang kita butuhkan, lalu terapkan salah satu gradien linier atau aturan gradien

radial, pastikan kita juga menyediakan aturan untuk semua browser yang kita targetkan. kita juga dapat menggunakan lebih dari sekadar warna awal dan akhir dengan juga menyediakan apa yang disebut warna berhenti di antaranya sebagai argumen tambahan. Dalam kasus ini, misalnya, jika lima argumen diberikan, setiap argumen akan mengontrol perubahan warna pada seperlima area yang diwakili oleh lokasinya dalam daftar argumen.

2.18 ELEMEN PEMOSISIAN

Elemen dalam halaman web berada di tempatnya ditempatkan dalam dokumen, tetapi kita dapat memindahkannya dengan mengubah properti posisi elemen dari default statis menjadi mutlak, relatif, atau tetap.

Pemosisian Absolut

Elemen dengan pemosisian absolut dihapus dari dokumen, dan elemen lain apa pun yang mampu akan mengalir ke ruang bebasnya. kita kemudian dapat memposisikan objek di mana pun kita suka di dalam dokumen menggunakan properti atas, kanan, bawah, dan kiri. Kemudian akan beristirahat di atas (atau di belakang) elemen lainnya. Jadi, misalnya, untuk memindahkan objek dengan ID objek ke lokasi absolut 100 piksel ke bawah dari awal dokumen dan 200 piksel ke dalam dari kiri, kita akan menerapkan aturan berikut (Anda juga dapat menggunakan salah satu dari unit pengukuran lain yang didukung oleh CSS):

```
#object {
position:absolute;
top :100px;
left :200px;
}
```

Pemosisian Relatif

Demikian juga, kita dapat memindahkan objek relatif ke lokasi yang akan ditempatinya dalam aliran dokumen normal. Jadi, misalnya, untuk memindahkan objek 10 piksel ke bawah dan 10 piksel ke kanan dari lokasi normalnya, kita akan menggunakan aturan berikut:

```
#object {
position:relative;
top :10px;
left :10px;
}
```

Pemosisian Tetap

Pengaturan properti pemosisian akhir memungkinkan kita memindahkan objek ke lokasi absolut, tetapi hanya di dalam viewport browser saat ini. Kemudian, saat dokumen di-scroll, objek tetap berada di tempatnya, dengan dokumen utama di-scroll di bawahnya—cara yang bagus untuk membuat dock bar dan perangkat serupa lainnya. Untuk memperbaiki objek ke sudut kiri atas jendela browser, kita akan menggunakan aturan berikut:

```
#object {
position:fixed;
top :0px;
```

```
left :0px;
}
```

Pada Gambar 2.26, Contoh 6 telah dimuat ke dalam browser, dan lebar dan tinggi browser telah dikurangi sehingga kita harus menggulir ke bawah untuk melihat semua halaman web.

Gambar 2.26 Menggunakan nilai pemosisian yang berbeda

Ketika ini selesai, segera terlihat bahwa elemen dengan posisi tetap tetap di tempatnya bahkan melalui pengguliran. kita juga dapat melihat bahwa elemen dengan pemosisian absolut terletak tepat pada 100 piksel ke bawah, dengan 0 offset horizontal, sedangkan elemen dengan pemosisian relatif sebenarnya dipindahkan ke atas sebesar 8 piksel dan kemudian diimbangi dari margin kiri sebesar 110 piksel untuk membuat garis di samping elemen pertama.

Contoh 6 Menerapkan nilai pemosisian yang berbeda

```
<!DOCTYPE html>
<html>
<head>
<title>Positioning</title>
<style>
#object1 {
position :absolute;
background:pink;
width :100px;
height :100px;
top :100px;
left :0px;
}
#object2 {
position :relative;
background:lightgreen;
width :100px;
height :100px;
top :-8px;
left :110px;
```

```
}
#object3 {
position :fixed;
background:yellow;
width :100px;
height :100px;
top :100px;
left :236px;
}
</style>
</head>
<body>
<br><br><br><br><br>
<div id='object1'>Absolute Positioning</div>
<div id='object2'>Relative Positioning</div>
<div id='object3'>Fixed Positioning</div>
</body>
</html>
```

Pada gambar, elemen dengan posisi tetap awalnya sejajar dengan dua elemen lainnya, tetapi tetap berada di tempatnya sementara yang lain telah digulir ke atas halaman, dan sekarang muncul offset di bawahnya.

2.19 PSEUDECLASS

Ada sejumlah selektor dan kelas yang digunakan hanya dalam style sheet dan tidak memiliki tag atau atribut yang cocok dalam HTML apa pun. Tugas mereka adalah mengklasifikasikan elemen menggunakan karakteristik selain nama, atribut, atau kontennya — yaitu, karakteristik yang tidak dapat disimpulkan dari pohon dokumen. Ini termasuk pseudoclasses seperti link dan dikunjungi. Ada juga Pseudo-element yang membuat pilihan, yang mungkin terdiri dari elemen parsial seperti baris pertama atau huruf pertama. *Pseudeclass* dan *Pseudo-element* dipisahkan oleh karakter : (titik dua). Misalnya, untuk membuat kelas yang disebut bigfirst untuk menekankan huruf pertama dari suatu elemen, kita akan menggunakan aturan seperti berikut:

```
.bigfirst:first-letter {
font-size:400%;
float :left;
}
```

Ketika kelas bigfirst diterapkan pada suatu elemen, huruf pertama akan ditampilkan jauh lebih besar, dengan teks yang tersisa ditampilkan pada ukuran normal, mengalir dengan rapi di sekitarnya (karena properti float) seolah-olah huruf pertama adalah gambar atau objek lain . Pseudoclasses termasuk hover, link, aktif, dan dikunjungi, yang semuanya sebagian besar berguna untuk menerapkan elemen jangkar, seperti dalam aturan berikut, yang mengatur warna default semua link ke biru, dan link yang telah dikunjungi ke biru muda:

```
a:link { color:blue; }
a:visited { color:lightblue; }
```

Aturan berikut ini menarik karena mereka menggunakan pseudoclass hover sehingga hanya diterapkan ketika mouse ditempatkan di atas elemen. Dalam contoh ini, mereka mengubah tautan menjadi teks putih dengan latar belakang merah, memberikan efek dinamis yang biasanya hanya kita harapkan dari penggunaan kode JavaScript:

```
a:hover {
color :white;
background:red;
}
```

Di sini saya telah menggunakan properti background dengan satu argumen, alih-alih properti background-color yang lebih panjang. Pseudoclass aktif juga dinamis karena mempengaruhi perubahan pada tautan selama waktu antara tombol mouse diklik dan dilepaskan, seperti aturan ini, yang mengubah warna tautan menjadi biru tua:

```
a:active { color:darkblue; }
```

Pseudoclass dinamis lainnya yang menarik adalah fokus, yang diterapkan hanya ketika sebuah elemen diberikan fokus oleh pengguna yang memilihnya dengan keyboard atau mouse. Aturan berikut menggunakan selektor universal untuk selalu menempatkan batas 2-piksel pertengahan abu-abu, putus-putus di sekitar objek yang saat ini difokuskan:

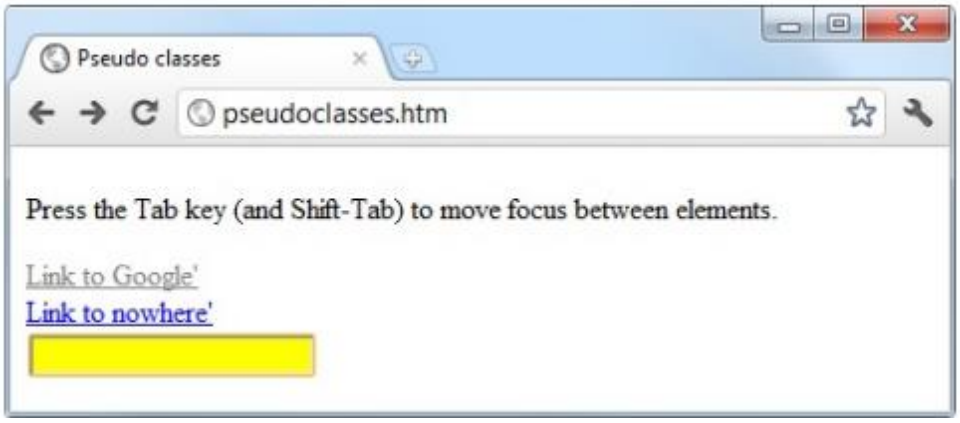
```
*:focus { border:2px dotted #888888; }
```

Contoh 7 menampilkan dua link dan sebuah field input, seperti yang ditunjukkan pada Gambar 2.27. Tautan pertama muncul sebagai abu-abu karena telah dikunjungi di browser ini, tetapi tautan kedua belum dan ditampilkan dengan warna biru. Tombol Tab telah ditekan dan fokus input sekarang menjadi bidang input, sehingga latar belakangnya berubah menjadi kuning. Ketika salah satu tautan diklik, itu akan ditampilkan dalam warna ungu, dan ketika diarahkan ke atasnya akan tampak merah.

Contoh 7 Tautkan dan fokuskan pseudoclass

```
<!DOCTYPE html>
<html>
<head>
<title>Pseudoclasses</title>
<style>
a:link { color:blue; }
a:visited { color:gray; }
a:hover { color:red; }
a:active { color:purple; }
*:focus { background:yellow; }
</style>
</head>
```

```
<body>
<a href='http://google.com'>Link to Google'</a><br>
<a href='nowhere'>Link to nowhere'</a><br>
<input type='text'>
</body>
</html>
```



Gambar 2.27 Pseudoclass diterapkan pada pilihan elemen

Pseudeclass lainnya juga tersedia, dan kita bisa mendapatkan informasi lebih lanjut tentangnya di http://tinyurl.com/pseudoclasses.

Catatan: Hati-hati dalam menerapkan pseudoclass fokus ke selektor universal, *, seperti yang ditunjukkan dalam contoh ini; Internet Explorer menganggap dokumen yang tidak fokus sebagai benar-benar memiliki fokus yang diterapkan ke seluruh halaman web, dan (dalam hal ini) seluruh halaman akan berubah menjadi kuning hingga Tab ditekan atau fokus diterapkan ke salah satu elemen usia.

Menambahkan satu gambar background

Salah satu fitur gambar background adalah kita dapat memasang atau mengulanginya dalam satu halaman. Artinya, seberapa besar ukuran layar pengunjung website, gambar backgrounf akan selalu muncul. Sebalikny, kita juga dapat memilih untuk tidak mengulangi gambar background. Bagian ini menunjukkan cara menambahkan gambar tunggal yang tidak berulang. Untuk menyelesaikan latihan ini, kita memerlukan gambar. Gambar sebaiknya paling sedikit 800 piksel kali 600 piksel. kita dapat mengetahui resolusi gambar dengan mengklik kanan gambar dan memilih Properties di Windows atau memilih Get Info dari jendela Finder di Mac.

1. Buka text editor
   Buat dokumen teks kosong baru di text editor.
2. Mmasukkan HTML berikut

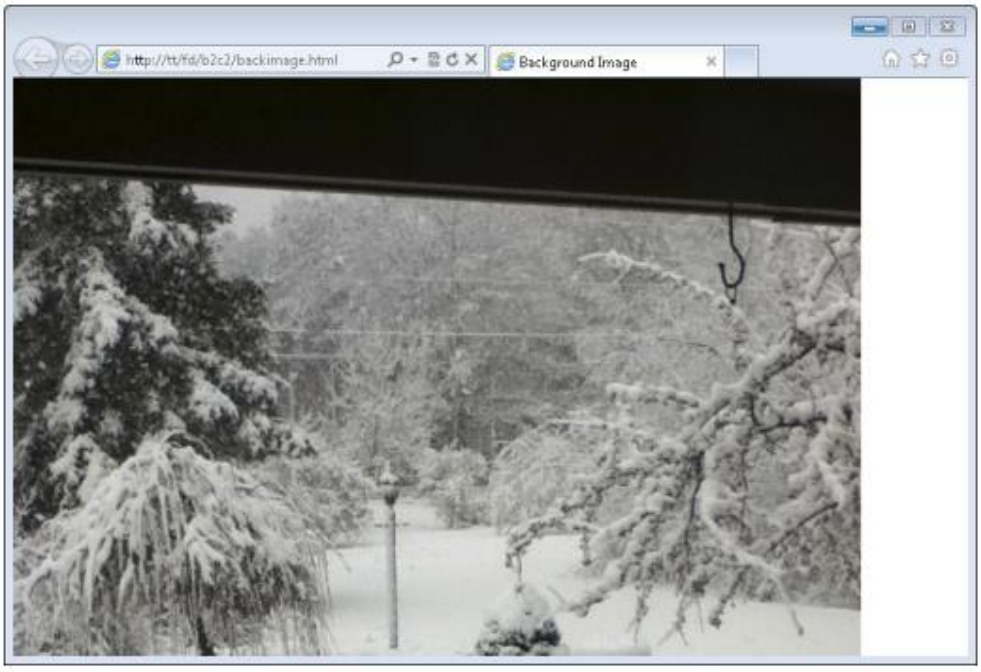
```
<!doctype html>
<html>
<head>
<title>Background Image</title>
<link rel=”stylesheet” type=”text/css”
href=”image-style.css”>
</head>
```

```
<body>
</body>
</html>
```

3. Simpan file.
   Simpan file sebagai backimage.html di root dokumen Anda.
4. Buat dokumen teks baru.
   Buat dokumen teks kosong baru dengan text editor.
5. Tempatkan CSS berikut di dokumen baru.
   Pastikan untuk menggunakan nama gambar Anda. Dalam contoh ini, kami menggunakan gambar yang disebut large-snow.jpg. Gambar harus disimpan di dalam root dokumen Anda.

```
body {
 background-image:url("large-snow.jpg");
 background-repeat: no-repeat;
}
```

6. Simpan file CSS.
   Simpan file sebagai image-style.css dan pastikan itu disimpan di dalam root dokumen Anda.
7. Buka browser web kita dan lihat halamannya.
   Buka browser web kita dan navigasikan ke halaman di http://localhost/backimage.html. nda akan melihat halaman dengan gambar latar belakang. kita dapat melihat tangkapan layar halaman kami, dengan gambar large-snow.jpg, pada gambar berikut.



Gambar 2.28 Sebuah gambar latar belakang tunggal.

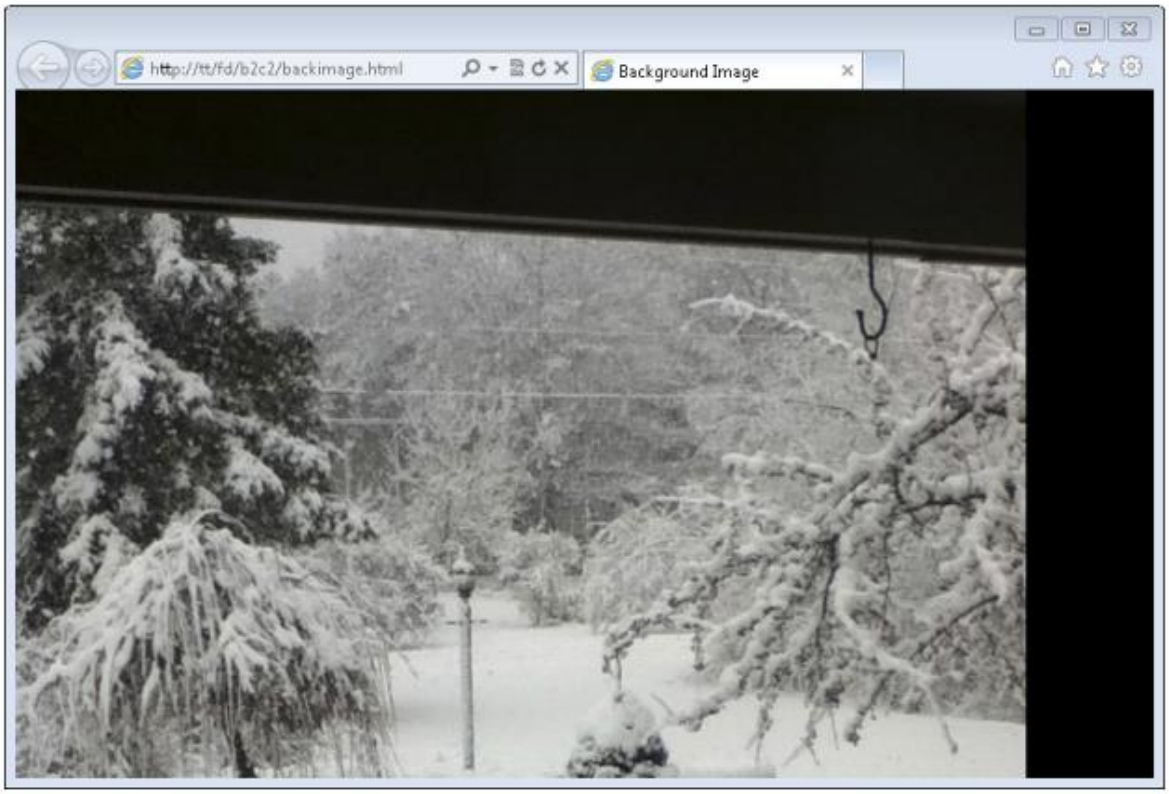
Tergantung pada ukuran gambar dan layar Anda, kita mungkin memperhatikan bahwa gambar berakhir, seperti halnya di sepanjang sisi kanan diatas. Selain itu, kita mungkin memperhatikan bahwa gambar tidak berada di tengah. Baca terus untuk solusinya.

Meningkatkan halaman gambar latar belakang tunggal

Pendekatan umum yang digunakan untuk membuat halaman yang tampak lebih baik adalah dengan menambahkan warna latar belakang yang cocok dengan tepi gambar. Dalam kasus gambar kami, bagian atas dan bawah berwarna hitam. Oleh karena itu, kita dapat menambahkan aturan ke CSS untuk membuat warna latar belakang default menjadi hitam. Ini tidak akan berpengaruh apa pun di mana gambar berada — gambar didahulukan — tetapi itu akan menjadi masalah di sepanjang bagian bawah di mana gambar berakhir. CSS untuk tampilan ini adalah sebagai berikut:

```
body {
 background-image:url(“large-snow.jpg”);
 background-repeat: no-repeat;
 background-color: #000000;
}
```

Dengan aturan itu, gambar akan tetap berakhir tetapi penampilannya tidak akan terlalu mengejutkan atau mencolok karena cocok dengan warna tepi gambar, seperti yang ditunjukkan pada gmbar berikut.



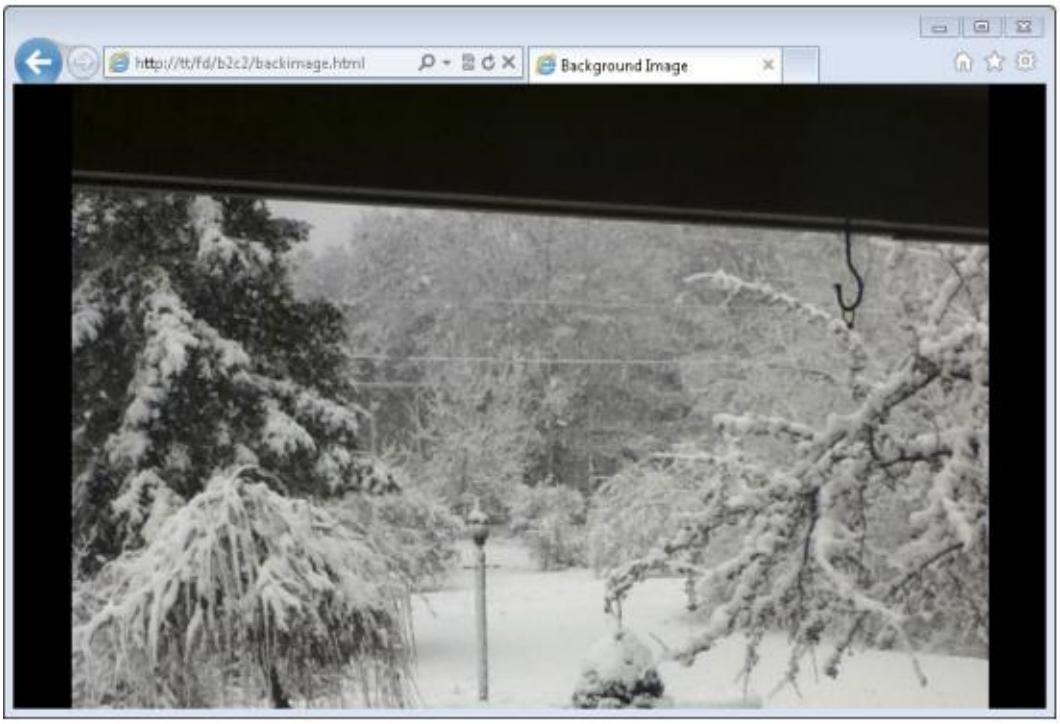
Gambar 2.29 Menambahkan warna latar belakang dan gambar background.

Sementara trik warna latar belakang memecahkan masalah dengan tepi gambar, itu tidak menyelesaikan masalah pemusatan. Gambar latar belakang saat ini diterapkan ke badan

— dengan kata lain, seluruh halaman. Untuk memusatkan gambar latar belakang, properti CSS lain perlu ditambahkan, seperti yang ditunjukkan dalam CSS ini:

```
body {
 background-image:url(“large-snow.jpg”);
 background-repeat: no-repeat;
 background-color: #000000;
 background-position: center top;
}
```

CSS ini menambahkan aturan posisi latar belakang dan menempatkannya di tengah di bagian atas halaman. Nilai lainnya termasuk kiri, kanan, dan bawah, dan kita dapat menggabungkannya sehingga gambar latar belakang akan muncul di kanan bawah, misalnya. CSS yang ditampilkan di sini menempatkan gambar di tengah halaman dan di atas. Ini menghasilkan halaman yang ditunjukkan pada Gambar berikut.



Gambar 2.30 Gambar latar tengah di bagian atas, dengan warna latar belakang.

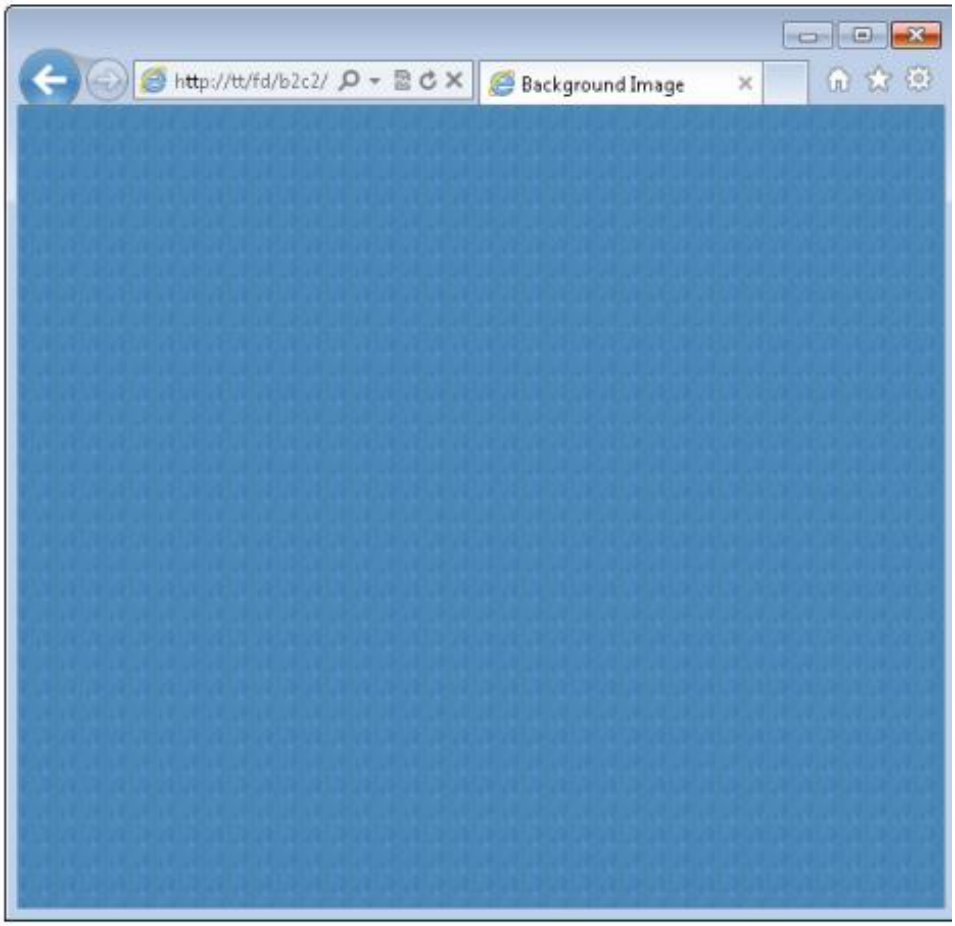
Dengan gambar itu di tempatnya, kita kemudian dapat menambahkan HTML apa pun ke halaman yang kita inginkan. Perhatikan dengan gambar seperti ini (bagian atas gelap dan tengah terang) kita perlu menyesuaikan warna font agar teks terlihat di halaman.

2.20 MENAMBAHKAN GAMBAR BERULANG

Anda dapat menambahkan gambar yang berulang. Ini adalah skenario umum untuk halaman web karena gambar tidak berakhir di sepanjang sisi, tidak peduli seberapa besar resolusi Anda. Ini juga mengurangi kebutuhan akan posisi latar belakang karena gambar latar berlaku untuk seluruh elemen. Saat diterapkan ke seluruh halaman, seperti pada contoh yang ditampilkan, kita juga dapat mengabaikan aturan pengulangan latar belakang dan warna latar belakang karena gambar berlanjut di seluruh halaman. Gambar berulang yang ideal adalah

gambar yang tidak memiliki batas yang terlihat karena batas tersebut akan muncul saat gambar dipetakan atau diulang pada halaman. Gambar 2.31 menunjukkan gambar kecil (15 piksel x 15 piksel) yang digunakan sebagai gambar berulang dengan CSS berikut:

```
body {
 background-image:url(“back.jpg”);
}
```



Gambar 2.31 Gambar latar berulang.

Seperti pada contoh untuk latar belakang gambar tunggal, kita sekarang dapat menambahkan HTML di atas latar belakang, sekali lagi memilih warna font yang mengimbangi gambar sehingga pengunjung dapat dengan mudah membaca teks.

2.21 MEMBUAT LAYOUT HALAMAN

Anda sekarang telah mempelajari banyak CSS untuk mengubah perilaku dan tampilan setiap item, menambahkan warna latar belakang, daftar gaya, dan sebagainya. Semua ini mengarah pada pembuatan halaman dengan menggunakan CSS. CSS digunakan untuk membuat tampilan halaman web yang lebih kompleks daripada yang kita lihat sejauh ini. Misalnya, kita dapat membuat efek kolom, di mana ada menu di sisi kiri atau kanan dan konten di kolom lain, dan kami memberi tahu kita cara melakukannya di sini.

Saat bekerja dengan perataan dan layout kolom, terkadang berguna untuk menambahkan batas ke elemen untuk melihat di mana itu dimulai dan berakhir sehingga kita bisa melihat tampilan layout.

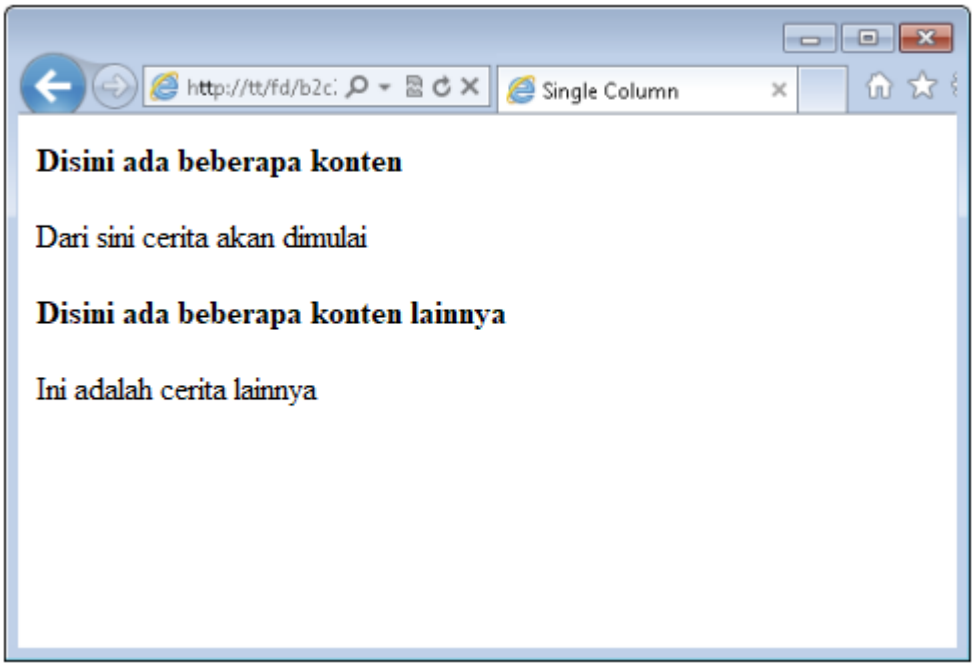
Membuat layout satu kolom

Semua yang kita lihat sejauh ini adalah layout satu kolom. Hanya ada satu kolom, sejajar di sebelah kiri halaman. Namun, kita dapat mengontrol penyelarasan itu dengan CSS. Melakukannya berarti membuat HTML yang lebih kompleks daripada yang kita lihat sejauh ini, tetapi tidak ada yang baru di HTML; hanya akan ada lebih banyak HTML daripada sebelumnya.

1. Buka text editor
2. Di dalam dokumen kosong, masukkan HTML berikut:

```
<!doctype html>
<html>
<head>
<title>Single Column</title>
<link rel="stylesheet" type="text/css" href="single.
css">
</head>
<body>
<div id="container">
 <div id="content">
 <h2>Disini ada beberapa konten</h2>
 <p>Dari sini cerita akan dimulai go</p>
 <h2>Disini ada beberapa konten lainnya</h2>
 <p>Ini adalah cerita lainnya</p>
 </div> <!-- end content -->
</div> <!-- end container -->
</body>
</html>
```

3. Simpan file
   Simpan file sebagai single.html pada dokumen root.
4. Buka browser untuk menampilkan halaman
   Lihat halaman dengan membuka http://localhost/single.html di browser Anda. kita akan melihat halaman seperti gambar berikut.

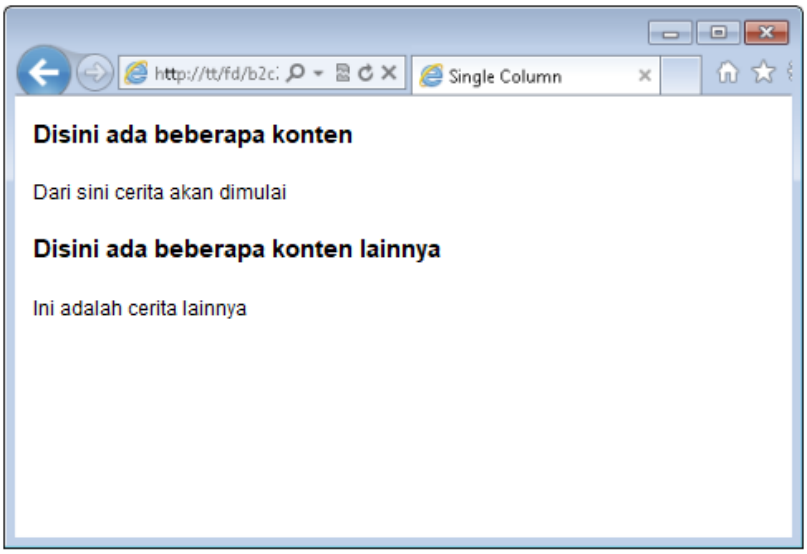


Gambar 2.32 Halaman dasar sebelum menambahkan CSS

5. Buat dokumen teks baru.
   Buat dokumen teks kosong baru di text editor.
6. Dalam dokumen, tempatkan CSS berikut

```
body {
 font-family: arial,helvetica,sans-serif;
}
#container {
 margin: 0 auto;
 width: 600px;
 background: #FFFFFF;
}
#content {
 clear: left;
 padding: 20px;
}
```

7. Simpan file CSS.
   Simpan file sebagai single.css di root dokumen Anda.
8. Buka browser web Anda.
   Arahkan ke http://localhost/single.html di browser Anda. Jika browser kita masih terbuka, muat ulang halaman dengan Ctrl+R di Windows atau Command+R di Mac. kita akan melihat halaman seperti itu pada Gambar dibawah ini. Lihat paragraf berikut untuk informasi lebih lanjut tentang modifikasi spesifik apa yang kita buat di Langkah 6.



Gambar 2.33 Satu layout kolom.

Untuk layout HTML ini menggunakan elemen sebagai wadah. Wadah membantu membuat layout dan tidak menyimpan konten teks apa pun sendiri. CSS untuk latihan ini menggunakan tiga properti CSS width, margin, dan clear. Cara kerjanya adalah sebagai berikut:

- **width**: Mengatur lebar horizontal dari sebuah elemen. Dalam hal ini, wadah diatur ke lebar 600px (piksel). Tidak peduli seberapa kecil jendela browser, HTML kita tidak akan pernah lebih kecil dari 600px.
- **margin**: Ini adalah pelengkap dari properti padding yang ditunjukkan sebelumnya dalam bab ini, di bagian "Adding Border". Properti margin mendefinisikan jarak di luar elemen. Dalam kasus yang ditunjukkan di sini (margin: 0 auto;), metode pintasan digunakan. Lihat bilah sisi untuk informasi lebih lanjut. Nilai "auto" berarti browser akan memilih nilainya.
- **clear**: Membuatnya agar tidak ada elemen yang muncul di sisi elemen yang aturannya berlaku. Dalam contoh, clear left digunakan pada dengan id #content. Ini berarti tidak ada yang bisa muncul di sisi kiri elemen itu. Nilai lain untuk clear termasuk "both", "none", "right", dan "inherent".

Dari sini kita dapat bereksperimen dengan margin jendela browser untuk melihat bagaimana layout yang dibuat dapat bereaksi atau bergerak seiring dengan pergerakan browser. Layout yang dibuat di bagian ini disebut *layout single-column fixed-width*. Pilihan lainnya adalah layout *single-column liquid*. Layout *single-column liquid* dapat bekerja lebih baik di perangkat tertentu. Layout lebar tetap yang ditampilkan terkadang dapat menghasilkan bilah gulir horizontal di bagian bawah halaman. Untuk mengubah layout menjadi layout *single-column liquid*, hanya perlu mengubah sedikit CSS di #container, seperti yang ditunjukkan di sini:

```
body {
 font-family: arial,helvetica,sans-serif;
}
#container {
 margin: 0 30px;
 background: #FFFFFF;
```

```
}
#content {
 clear: left;
 padding: 20px;
}
```

Perhatikan satu-satunya perubahan adalah menghapus properti lebar di dalam #container dan juga mengubah margin dari "0 auto" menjadi "0 30px." Dengan itu, layout menjadi liquid layout dan berfungsi lebih baik, terutama di perangkat seluler.

Aturan Singkatan

Untuk menghemat ruang, grup properti CSS terkait dapat digabungkan menjadi satu tugas singkatan. Misalnya, saya telah menggunakan singkatan untuk membuat batas beberapa kali, seperti dalam aturan fokus di bagian sebelumnya:

```
*:focus { border:2px dotted #ff8800; }
```

Ini sebenarnya adalah rangkaian singkatan dari kumpulan aturan berikut:

```
*:focus {
border-width:2px;
border-style:dotted;
border-color:#ff8800;
}
```

Saat menggunakan aturan singkatan, kita hanya perlu menerapkan properti hingga titik di mana kita ingin mengubah nilai. Jadi kita dapat menggunakan yang berikut ini untuk hanya mengatur lebar dan gaya batas, memilih untuk tidak mengatur warnanya:

```
*:focus { border:2px dotted; }
```

Model dan Layout Kotak

Properti CSS yang memengaruhi layout halaman didasarkan pada model kotak, kumpulan properti bersarang yang mengelilingi elemen. Hampir semua elemen memiliki (atau dapat memiliki) properti ini, termasuk badan dokumen, yang marginnya dapat Anda, misalnya, hapus dengan aturan berikut:

```
body { margin:0px; }
```

Model kotak suatu objek dimulai dari luar, dengan margin objek. Di dalam ini adalah border, lalu ada padding antara border dan isi bagian dalam, dan akhirnya ada isi objek. Setelah kita memahami model kotak, kita akan siap untuk membuat halaman yang ditata secara profesional, karena properti ini saja yang akan membuat banyak gaya halaman Anda.

Pintasan untuk margin dan padding

Daripada menentukan aturan untuk setiap elemen margin atau padding atas, bawah, kanan, dan kiri, kita dapat menggunakan metode pintasan yang mendefinisikan semuanya dalam satu baris. Sebagai contoh:

```
margin: 0px 50px 200px 300px;
setara dengan ini:
margin-top: 0px;
margin-right: 50px;
margin-bottom: 200px;
margin-left: 300px;
```

Ketika empat angka muncul dalam aturan, urutannya adalah atas, kanan, bawah, dan kiri. Untuk membantu mengingat urutannya, gunakan mnemonic "TrouBLe," yang mengambil huruf pertama dari masing-masing Atas, Kanan, Bawah, Kiri, dan membuatnya menjadi sebuah kata untuk mengingatkan kita betapa sulitnya mengingat urutannya. Alih-alih keempat nilai, terkadang kita melihat satu, dua, atau tiga nilai yang ada untuk margin atau padding, seperti pada contoh yang ditunjukkan sebelumnya:

```
margin: 0 auto;
```

Ketika dua nilai digunakan, nilai pertama sesuai dengan atas dan bawah dan nilai kedua sesuai dengan kanan dan kiri. Ketika tiga nilai digunakan, yang pertama adalah bagian atas, yang kedua adalah kiri dan kanan, dan yang terakhir adalah bagian bawah. Akhirnya, ketika satu nilai digunakan, itu berlaku sama untuk atas, kanan, bawah, dan kiri.

Mengatur Margin

Margin adalah tingkat terluar dari model kotak. Ini memisahkan elemen satu sama lain dan penggunaannya cukup cerdas. Misalnya, asumsikan kita telah memilih untuk memberi sejumlah elemen margin default 10 piksel di sekitar masing-masing elemen. Ketika mereka ditempatkan di atas satu sama lain, ini akan menciptakan celah 20 piksel (total lebar border yang berdekatan). CSS mengatasi masalah potensial ini, namun: ketika dua elemen dengan batas diposisikan langsung satu di atas yang lain, hanya yang lebih besar dari dua margin yang digunakan untuk memisahkannya. Jika kedua margin memiliki lebar yang sama, hanya salah satu lebar yang digunakan. Dengan cara ini, kita jauh lebih mungkin untuk mendapatkan hasil yang kita inginkan. Tetapi kita harus memperhatikan bahwa margin elemen yang diposisikan secara mutlak atau sebaris tidak runtuh.

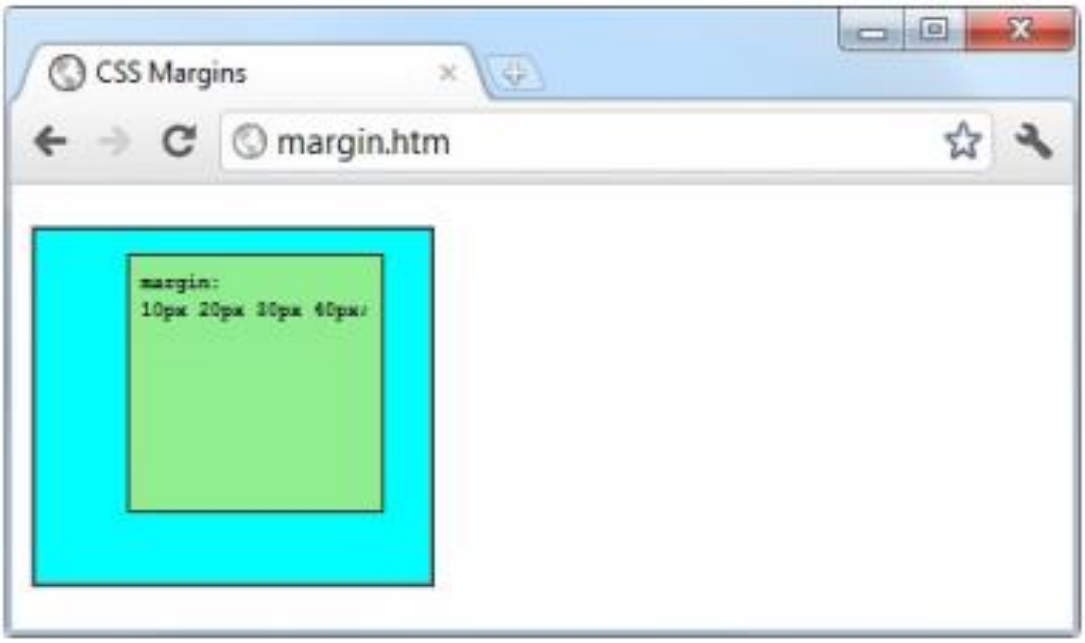
Margin suatu elemen dapat diubah secara massal dengan properti margin, atau secara individual dengan margin-kiri, margin-atas, margin-kanan, dan margin bawah. Saat mengatur properti margin, kita bisa menyediakan satu, dua, tiga, atau empat argumen, yang memiliki efek yang dikomentari dalam aturan berikut:

```
/* Set all margins to 1 pixel
margin:1px;
Set top and bottom to 1 pixel, and left and right to 2
margin:1px 2px;
Set top to 1 pixel, left and right to 2, and bottom to 3
margin:1px 2px 3px;
Set top to 1 pixel, right to 2, bottom to 3, and left to 4 */
margin:1px 2px 3px 4px;
```

Gambar 2.34 menunjukkan Contoh 8 dimuat ke dalam browser, dengan aturan properti margin (disorot dalam huruf tebal) diterapkan pada elemen persegi yang telah ditempatkan di dalam elemen tabel. Tabel tidak diberi dimensi, sehingga hanya akan membungkus elemen bagian dalam sedekat mungkin. Akibatnya, ada margin 10 piksel di atasnya, 20 piksel di sebelah kanan, 30 piksel di bawahnya, dan 40 piksel di sebelah kirinya.

Contoh 8. Bagaimana margin diterapkan

```
<!DOCTYPE html>
<html>
<head>
<title>Margins</title>
<style>
#object1 {
background :lightgreen;
border-style:solid;
border-width:1px;
font-family :"Courier New";
font-size :9px;
width :100px;
height :100px;
padding :5px;
margin :10px 20px 30px 40px;
}
table {
padding :0;
border :1px solid black;
background :cyan;
}
</style>
</head>
<body>
<table>
<tr>
<td>
<div id='object1'>margin:<br>10px 20px 30px 40px;</div>
</td>
</tr>
</table>
</body>
</html>
```



Gambar 2.34 Tabel luar mengembang sesuai dengan lebar margin

Membuat layout dua kolom

Layout dua kolom menggunakan sedikit lebih banyak HTML untuk mencapai efek beberapa kolom. Ini sering dilakukan untuk menambahkan menu di sepanjang sisi halaman atau tautan ke cerita atau konten lain. Daftar berikut menunjukkan HTML yang terlibat untuk layout lebar tetap dua kolom.

Contoh 9 Layout Lebar Tetap Dua Kolom

```
<!doctype html>
<html>
<head>
<title>Two Column</title>
<link rel=”stylesheet” type=”text/css” href=”double.css”>
</head>
<body>
<div id=”container”>
 <div id=”mainContainer”>
 <div id=”content”>
 <h2>Disini ada beberapa konten</h2>
 <p>Dari sini cerita akan dimulai go</p>
 <h2>Disini ada beberapa konten lainnya</h2>
 <p>Ini adalah cerita lainnya</p>
 </div> <!-- end content -->
 <div id=”sidebar”>
 <h3>Menu</h3>
 <ul>
 <li>Menu item 1</li>
 <li>Menu item 2</li>
 <li>Menu item 3</li>
```

```
 </ul>
 </div> <!-- end sidebar -->
 </div> <!-- end mainContainer -->
</div> <!-- end container -->
</body>
</html>
```

HTML ini menggunakan wadah dari layout kolom tunggal dan menambahkan wadah lain untuk menampung konten. ini disebut sebagai mainContainer, menampung konten dan bilah sisi. Tambahan lainnya adalah sidebar itu sendiri, tepat berjudul sidebar. Bilah sisi itu menyimpan menu dengan daftar tidak berurutan (<ul>) di dalamnya. CSS untuk layout dua kolom ditunjukkan pada daftar berikut.

Contoh 10 CSS untuk Layout Lebar Tetap Dua Kolom

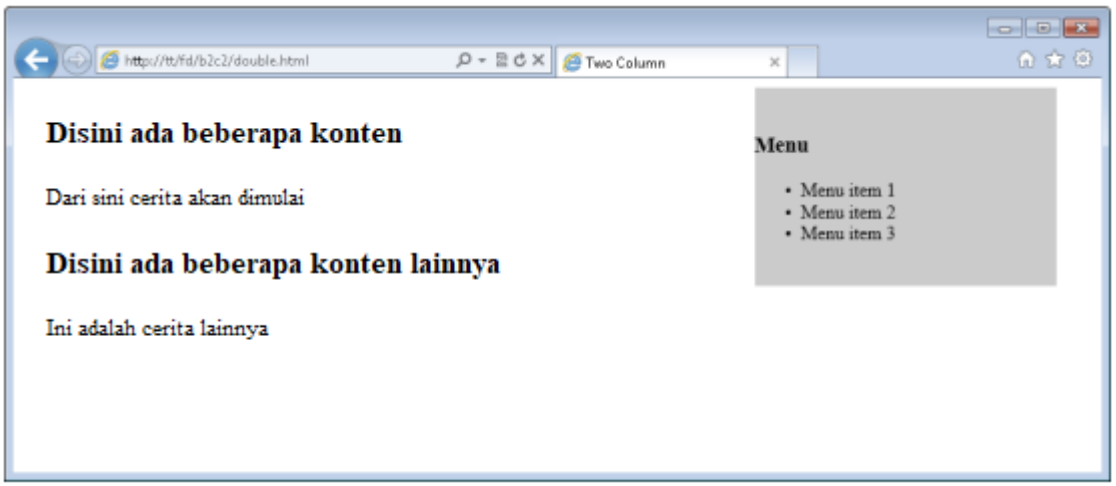
```
#container {
 margin: 0 auto;
 width: 900px;
}
#mainContainer {
 float: left;
 width: 900px;
}
#content {
 clear: left;
 float: left;
 width: 500px;
 padding: 20px 0;
 margin: 0 0 0 30px;
 display: inline;
}
#sidebar {
 float: right;
 width: 260px;
 padding: 20px 0;
 margin: 0 20px 0 0;
 display: inline;
 background-color: #CCCCCC;
}
```

CSS ini menggunakan beberapa item yang telah kita lihat, antara lain margin, padding, clear, dan background-color. Di CSS ini hal yang baru adalah float dan properti display. Properti float menentukan apakah sebuah elemen akan bergerak atau mengambang di dalam sebuah layout, baik ke kiri atau ke kanan atau apakah elemen tersebut tidak akan mengambang sama sekali (tidak ada), seperti defaultnya. Namun, karena kita ingin membuat dua kolom bersebelahan, kita perlu melayangkan wadah konten ke kiri dan bilah sisi ke kanan. Oleh

karena itu, jika kita ingin sidebar muncul di sebelah kanan, kita hanya perlu menukar float: left di #content CSS dengan float: right yang terdapat di CSS #sidebar.

Properti display mengatur bagaimana elemen harus ditampilkan. Elemen tertentu dikenal sebagai elemen level blok dan menampilkan seluruh lebar halaman. Elemen adalah contoh yang bagus untuk ini. Karena kita ingin membuat kolom muncul bersebelahan, kita perlu mengubah perilaku tampilan blok ini menjadi sebaris (kami memperkenalkan elemen sebaris di bab sebelumnya), sehingga elemen tidak memperpanjang lebar penuh halaman.

Tiga nilai yang sering digunakan untuk properti tampilan adalah blok (untuk memperluas elemen dengan lebar penuh), inline (untuk membuat elemen hanya menggunakan lebarnya sendiri untuk tampilan), dan none (yang menghilangkan elemen dari tampilan seluruhnya).



Gambar 2.35 Layout lebar tetap dua kolom.

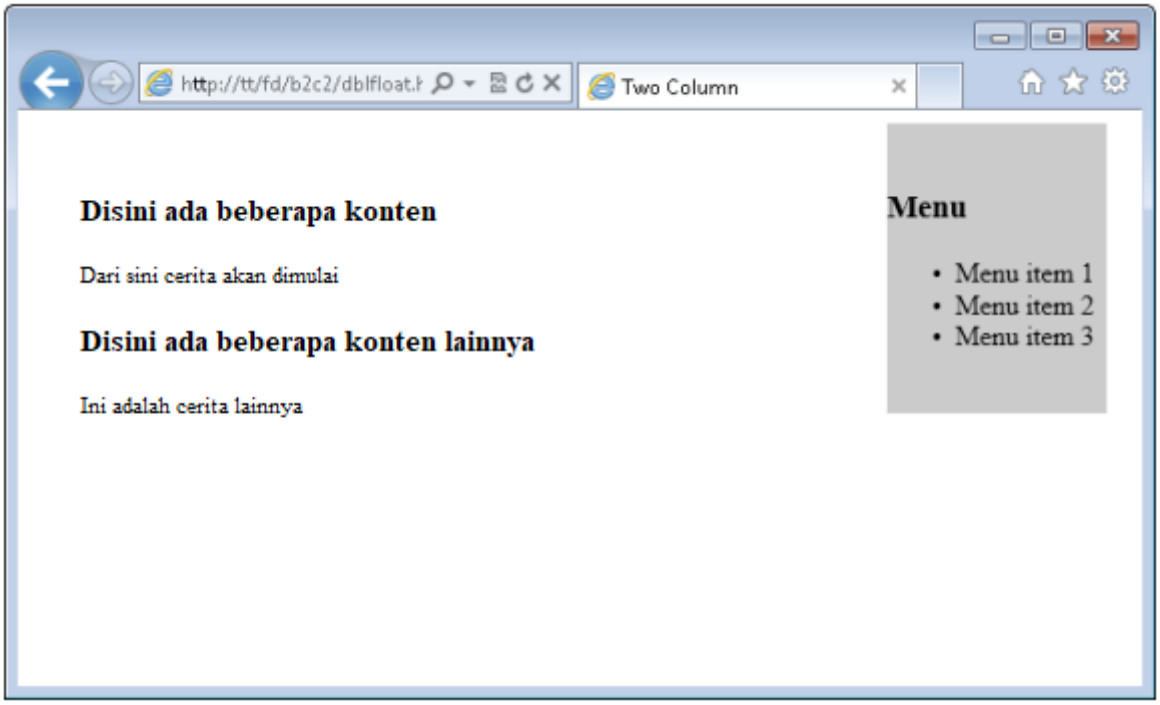
Layout yang ditunjukkan pada gambar diatas adalah layout lebar tetap. Mengonversi ini ke liquid layout berarti mengubah nilai width dan margin dalam CSS dari piksel (px) menjadi persentase (%). CSS untuk diubah menjadi liquid layout ditunjukkan pada daftar berikut.

Contoh 11 Mengonversi ke Liquid layoutan Dua Kolom.

```
#container {
 margin: 0 auto;
width: 100%;
}
#mainContainer {
 float: left;
 width: 100%;
}
#content {
 clear: left;
 float: left;
 width: 65%;
 padding: 20px 0;
 margin: 0 0 0 5%;
```

```
 display: inline;
}
#sidebar {
 float: right;
 width: 20%;
 padding: 20px 0;
 margin: 0 2% 0 0;
 display: inline;
 background-color: #CCCCCC;
}
```

Perubahan terjadi di bagian #container, #mainContainer, #content, dan #sidebar, untuk mengubah nilai sebelumnya yang menggunakan piksel menjadi persentase. Layout ini sekarang berubah dengan lebar browser, seperti yang ditunjukkan pada Gambar 2.36, di mana kita akan melihat bahwa lebar browser jauh lebih kecil.



Gambar 2.36 Membuat liquid layout dengan dua kolom.

Menyembunyikan elemen

Menyetel properti tampilan CSS ke none akan menyembunyikan elemen dari halaman. Saat kita melakukannya, elemen tersebut akan dihapus seluruhnya dari halaman. kita juga dapat menggunakan properti CSS lain, visibility, untuk menyembunyikan elemen. Saat menyembunyikan elemen dengan properti visibility (visibility: hidden;), kotak atau area pada halaman tetap berada di tempatnya tetapi elemen tersebut menjadi tidak terlihat. Membuat elemen terlihat lagi (visibility: terlihat;) menunjukkan elemen tersebut.

2.22 MENAMBAHKAN HEADER DAN FOOTER KE HALAMAN

Header biasanya digunakan untuk menyampaikan informasi seperti nama situs atau untuk menyediakan menu; footer digunakan untuk memberikan informasi tambahan seperti

hak cipta dan juga digunakan untuk menyediakan peta tautan dalam sebuah situs, yang dikenal sebagai *site map*. Selain itu, kami memberi tahu kita cara membuat menu di dalam header.

Membuat header, menu header, dan footer

Anda telah melihat cara membuat layout multi-kolom dengan area konten utama dan bilah sisi. Untuk membuat layout ini, kita menambahkan elemen untuk menahan konten bilah sisi. kita kemudian menerapkan aturan CSS ke untuk mengatur lebar dan posisinya. Membuat header dan footer sebagian besar dilakukan dengan cara yang sama. tambahan dibuat untuk menampung konten untuk masing-masing dan kemudian aturan diterapkan ke elemen tersebut untuk memposisikannya. Ini menjadi contoh terakhir dalam bab ini, ini berfungsi sebagai latihan batu penjuru.

1. Buka text editor
2. Masukkan HTML ini pada text dokumen :

```
<!doctype html>
<html>
<head>
<title>Two Column With Header and Footer</title>
<link rel=”stylesheet” type=”text/css” href=”final.
css”>
</head>
<body>
<div id=”container”>
 <div id=”header”>
 <h1>This is my site!</h1>
 </div> <!-- end header -->
 <div id=”menu”>
 <ul>
 <li><a href=”#”>Home</a></li>
 <li><A href=”#”>Services</a></li
<li><a href=”#”>About Me</a></li>
 <li><a href=”#”>Contact Me</a></li>
 </ul>
 </div> <!-- end menu -->
 <div id=”mainContainer”>
 <div id=”content”>
 <h2>Disini ada beberapa konten</h2>
 <p>Dari sini cerita akan dimulai go</p>
 <h2>Disini ada beberapa konten lainnya</h2>
 <p>Ini adalah cerita lainnya</p>
 </div> <!-- end content -->
 <div id=”sidebar”>
 <h3>Menu</h3>
 <ul>
 <li>Menu item 1</li>
 <li>Menu item 2</li>
```

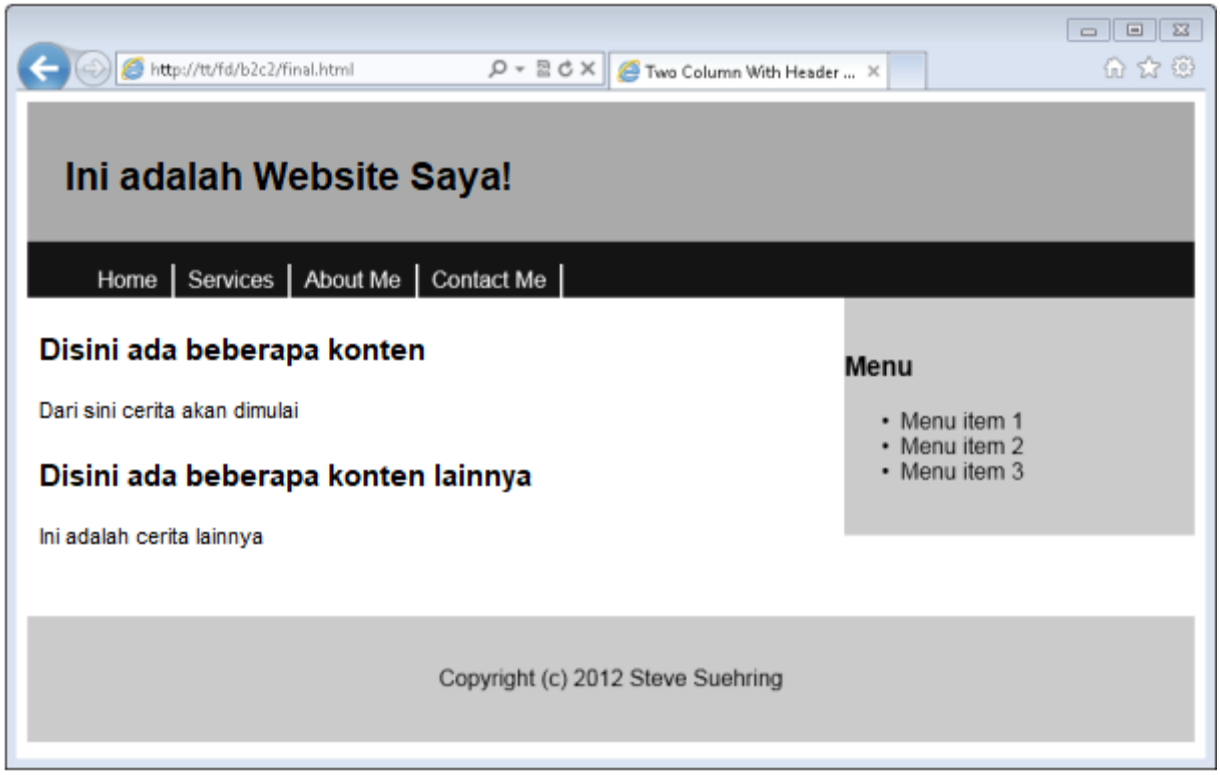
```
<li>Menu item 3</li>
</ul>
</div> <!-- end sidebar -->
<div id="footer">
<p>Copyright (c) 2022Agus Wibowo</p>
</div> <!-- end footer -->
</div> <!-- end mainContainer -->
</div> <!-- end container -->
</body>
```

3. Simpan file
4. Buat dokumen teks baru
   Buat dokumen menggunakan CSS berikut:

```
body {
 font-family: arial,helvetica,sans-serif;
}
#container {
 margin: 0 auto;
 width: 100%;
}
#header {
 background-color: #abacab;
 padding: 10px;
}
#menu {
 float: left;
 width: 100%;
background-color: #0c0c0c;
}
#menu ul li {
 list-style-type: none;
 display: inline;
}
#menu li a {
 display: block;
 text-decoration: none;
 border-right: 2px solid #FFFFFF;
 padding: 3px 10px;
 float: left;
 color: #FFFFFF;
}
#menu li a:hover {
 background-color: #CCCCCC;
}
#mainContainer {
```

```
 float: left;
 width: 100%;
}
#content {
 clear: left;
 float: left;
 width: 65%;
 padding: 20px 0;
 margin: 0 0 0 5%;
 display: inline;
}
#sidebar {
 float: right;
 width: 30%;
 padding: 20px 0;
 margin: 0;
 display: inline;
 background-color: #CCCCCC;
}
#footer {
 clear: left;
 background-color: #CCCCCC;
 text-align: center;
 padding: 20px;
 height: 1%;
}
```

5. Simpan File
   Simpan file CSS sebagai final.css pada dokumen root.
6. Buka browser untuk melihat halaman
   Buka browser web Anda, navigasikan ke http://localhost/final.html, dan kita akan melihat halaman, seperti yang ditunjukkan oleh gambar berikut.



Gambar 2.37 Layout liquid dua kolom dengan header dan footer

Memeriksa file HTML dan CSS

Untuk membuat layout ini, kita menggunakan file HTML yang lebih kompleks daripada yang pernah kita gunakan sebelumnya, tetapi tidak ada apa pun di file itu yang belum kita lihat. Hanya saja lebih lama untuk membuat HTML dan konten tambahan untuk halaman! CSS memang menggunakan beberapa item tambahan, khususnya untuk membuat menu atau tautan di bagian atas. Perhatikan bahwa ini terpisah dari menu kontekstual yang muncul di sebelah kanan. Menu yang dibuat untuk halaman ini muncul di header dan menyediakan tautan ke area situs, seperti **Home, Service, About Me**, dan **Contact Me.** CSS untuk bagian itu terlihat seperti ini:

```
#menu ul li {
 list-style-type: none;
 display: inline;
}
```

Bagian tersebut menggunakan struktur hierarki untuk menargetkan hanya elemen <li> dalam area #menu. Tipe-gaya-daftar disetel ke tidak ada, yang kita lihat sebelumnya di bab ini. Namun, tampilan diatur ke inline. Saat digunakan dengan daftar, itu membuat daftar mengalir secara horizontal daripada vertikal, sehingga kita mendapatkan efek yang diinginkan di sini. Bagian CSS selanjutnya mengubah perilaku elemen <a> dalam menu itu dan kembali ditargetkan menggunakan #menu li a sehingga aturan CSS hanya diterapkan pada elemen <a> tertentu tersebut.

```
#menu li a {
 display: block;
```

```
 text-decoration: none;
 border-right: 2px solid #FFFFFF;
 padding: 3px 10px;
 float: left;
 color: #FFFFFF;
}
```

Aturan CSS ini menggunakan properti float, display, dan border standar yang dijelaskan sebelumnya dalam bab ini. Ditambahkan di sini adalah properti CSS text-decoration, yang mengubah perilaku default tautan <a>. Alih-alih digarisbawahi dan diwarnai, mengubah text-decoration menjadi menghilangkan none dari efek tersebut, memberikan menu tampilan yang lebih bersih. Bagian terakhir dari CSS menu adalah ini:

```
#menu li a:hover {
 background-color: #CCCCCC;
}
```

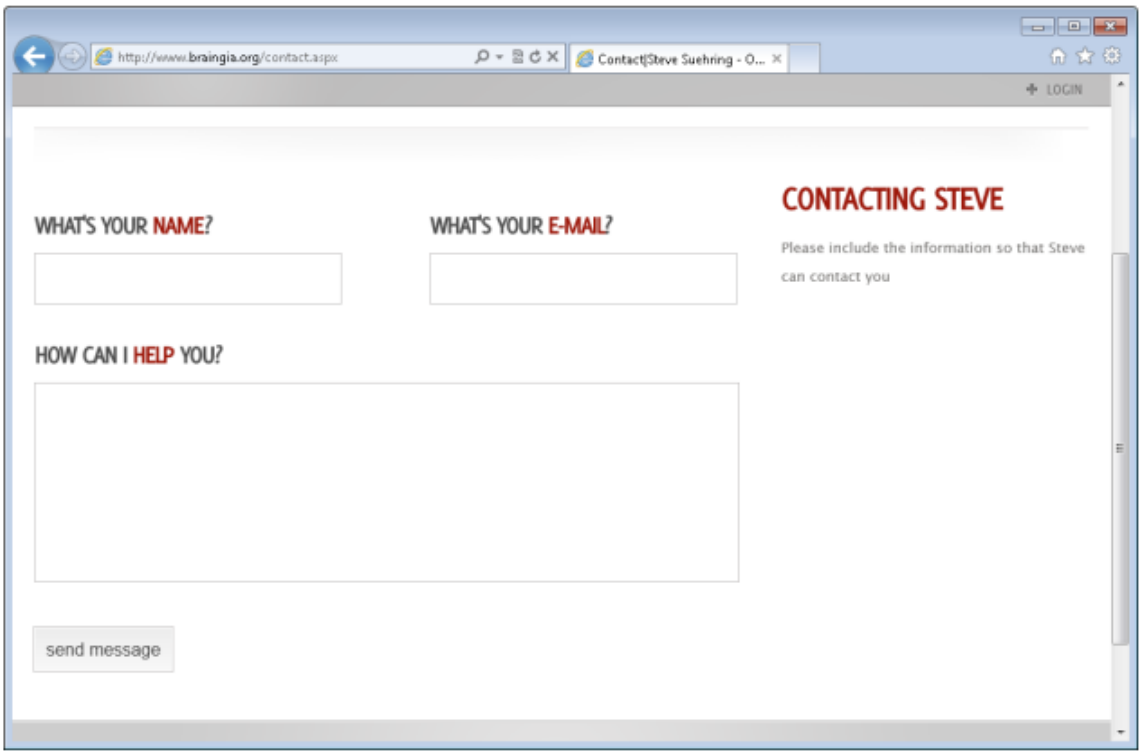
Aturan CSS ini menargetkan perilaku mengarahkan kursor ke elemen <a>. Saat pengunjung mengarahkan kursor ke tautan, itu akan berubah warna, dalam hal ini menjadi #CCCCCC, yang merupakan bayangan abu-abu.

BAB 3
MEMBUAT DAN MENATA FORMULIR WEB MENGGUNAKAN CSS

Formulir web memungkinkan situs kita mengumpulkan informasi dari pengguna. Bab ini membahas formulir web dengan segala kemegahannya dan menunjukkan kepada kita cara membuat formulir dan cara menatanya dengan CSS.

3.1 MENGGUNAKAN FORMULIR WEB UNTUK MENDAPATKAN INFORMASI

Dengan formulir web, seperti yang ditunjukkan pada gambar dibawah ini, kita dapat mengumpulkan informasi dari pengguna.



Gambar 3.1 Formulir web dasar

Formulir web dapat mengumpulkan apa saja mulai dari nama dan alamat email dan pesan, seperti yang ditunjukkan pada gambar diatas, hingga gambar dan file dari komputer Anda. Misalnya, ketika kita masuk ke akun email berbasis web seperti Gmail, kita mengisi formulir dengan nama pengguna dan kata sandi Anda. Berikut adalah tampilan bagaimana kita dapat menggunakan HTML untuk membuat formulir web.

Memahami formulir web

Saat kita mengisi formulir, informasi dikirim ke server web. Apa yang server web lakukan pada informasi yang diinput tergantung pada program yang berjalan di server. Misalnya, ketika kita mengisi formulir kontak di situs web, server mengirimkan e-mail informasi yang dikirimkan, tetapi ketika kita mengisi formulir untuk menemukan kamar hotel di situs web hotel, server mencari di databasenya untuk kamar yang cocok berdasarkan tanggal yang diisi. Sekarang, mari fokus pada formulir. Dalam istilah HTML, formulir dibuat dengan elemen <form>. Formulir dibuka dengan <form> dan ditutup dengan </form>, seperti pada contoh ini:

```
<form action=”#”>
<input type=”text” name=”emailaddress”>
<input type=”submit” name=”submit”>
</form>
```

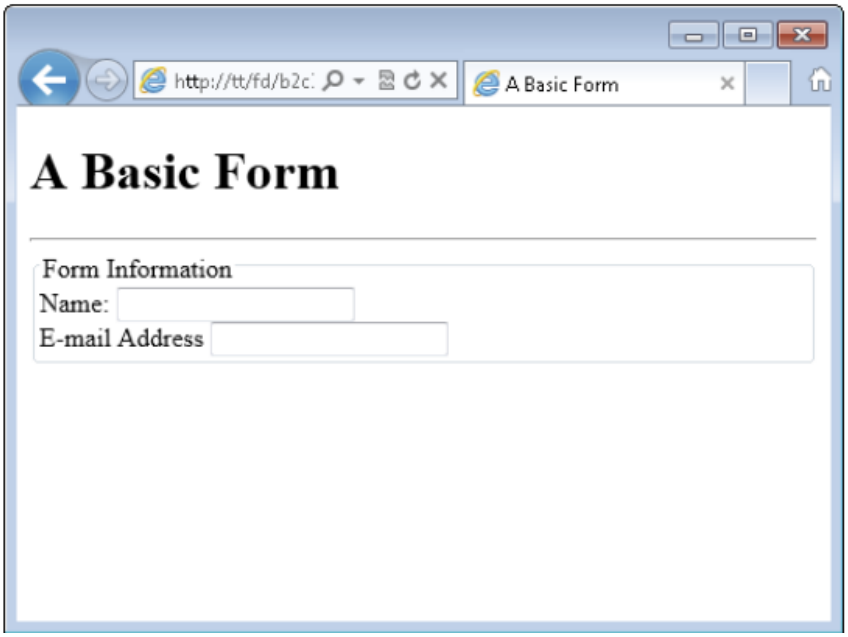
Anda melihat cara membuat formulir kita sendiri di bagian selanjutnya.

Melihat elemen formulir

Ada banyak cara untuk mendapatkan input melalui formulir, masing-masing dengan nama atau jenis inputnya sendiri. Contoh kode di bagian sebelumnya mencakup dua input types: text type dan submit text. Type text membuat kotak di mana pengguna dapat memasukkan informasi. Submit type membuat tombol yang digunakan pengguna untuk mengirim informasi ke server. Ada banyak jenis elemen input lainnya dalam formulir, termasuk ini:

- **Drop-down atau select**: Membuat kotak drop-down dengan beberapa pilihan yang dapat dipilih pengguna.
- **Check boxes:** Membuat satu atau lebih kotak yang dapat dipilih pengguna.
- **Tombol radio:** Membuat satu atau lebih tombol kecil, di mana pengguna hanya dapat memilih satu.
- **Lainnya**: Ada jenis khusus lainnya — termasuk kata sandi, area teks, dan file — yang memungkinkan kita mengumpulkan jenis input lain dari pengguna.

Kita telah melihat elemen formulir dasar, tetapi ada lebih banyak hal untuk membuat formulir daripada hanya menambahkan elemen. Formulir perlu diintegrasikan dengan HTML lain agar dapat ditampilkan seperti yang kita inginkan. Di luar itu, seperti yang kita lihat nanti di bab ini, kita juga dapat menata formulir dengan *Cascading Style Sheets* (CSS). Tetapi untuk saat ini, mari kita kerjakan formulir sederhana. Gambar berikut menunjukkan formulir web menggunakan dua jenis input teks.



Gambar 3.2 Formulir web dasar dengan dua input.

HTML yang digunakan untuk membuat formulir ini ditampilkan di sini.

```
<!doctype html>
<html>
<head>
<title>A Basic Form</title>
</head>
<body>
<h1>A Basic Form</h1>
<hr>
<form action="#">
<fieldset>
 <legend>Form Information</legend>
 <div>
 <label for="username">Name:</label>
 <input id="username" type="text" name="username">
 </div>
 <div>
 <label for="email">E-mail Address:</label>
 <input id="email" type="text" name="email">
 </div>
</fieldset>
</form>
</body>
</html>
```

Formulir dimulai dengan tag <form> pembuka. Saat kita membuat formulir, kita cukup sering menggunakan dua atribut, salah satunya adalah atribut tindakan. Atribut action memberi tahu formulir ke mana harus pergi atau apa yang harus dilakukan ketika pengguna mengklik Kirim. kita melihat atribut lain, metode, beberapa saat kemudian.

Elemen berikutnya yang ditemukan dalam formulir adalah <fieldset>, yang merupakan opsional untuk formulir. Elemen <fieldset> digunakan terutama untuk layout dan aksesibilitas. Elemen berikutnya yang ditemukan adalah elemen <legend>. Elemen ini membuat legenda Informasi Formulir dan kotak yang (meskipun sulit dilihat di tangkapan layar) mengelilingi input dalam formulir. Seperti <fieldset>, elemen <legend> sepenuhnya opsional. Selanjutnya dalam formulir adalah elemen yang digunakan untuk membuat setiap baris input. Elemen <label> mengikat nama ramah — apa yang kita lihat di layar, dalam hal ini, Nama — ke input yang sebenarnya. Elemen <label> bersifat opsional tetapi direkomendasikan karena membantu dengan teknologi bantu. Di bawah elemen <label> kita melihat elemen <input>. Struktur , <label>, <input> ini diulang untuk bidang Alamat E-mail.

3.2 MEMBUAT FORMULIR

Dengan beberapa pemahaman tentang bagaimana formulir disusun, saatnya untuk melihat pembuatannya dengan beberapa elemen yang sudah dibahas. Di bagian ini, kita mengetahui lebih lanjut tentang elemen <form> dan cara membuat kotak teks, kotak drop-down, kotak centang, dan tombol radio yang dapat digunakan pengunjung situs web kita untuk memasukkan informasi. kita juga mengetahui cara membuat tombol Kirim, yang

memungkinkan pengunjung menunjukkan bahwa mereka siap mengirimkan informasi itu kepada Anda.

Elemen bentuk

Anda telah melihat bahwa elemen <form> biasanya menggunakan beberapa atribut, tindakan, dan metode yang berbeda. Tindakan formulir biasanya menunjuk ke program server yang akan menangani input dari formulir. Di situlah formulir mengirimkan datanya.

Jika tindakan memberi tahu formulir ke mana harus mengirim data, maka atribut metode memberi tahu formulir cara mengirim data ke server. Ada dua metode utama yang akan kita temui: GET dan POST. Metode GET cocok untuk formulir kecil, sedangkan metode POST cocok untuk formulir besar atau formulir yang perlu mengirim banyak informasi. Ketika tindakan diatur, seperti yang kita lihat, ke tanda pagar atau tanda pagar (#), formulir pada dasarnya tidak ada apa-apanya dan tidak melakukan apa-apa, yang untuk saat ini persis seperti yang kita inginkan karena kita belum membangun program server untuk bekerja dengan data yang masuk belum.

3.3 MENGETAHUI PERBEDAAN METODE GET DAN POST

Saat kita menggunakan metode GET, konten formulir dikirim sebagai bagian dari URL. Dalam contoh formulir yang kita lihat sebelumnya, URL akan menjadi seperti:

http://localhost/form1.html?username=Agus&email=wibowo@example.com

Hal pertama yang kita perhatikan adalah bahwa pengguna dapat dengan mudah melihat semua elemen formulir, termasuk nama dan nilainya, langsung di bilah alamat browser mereka. Namun, di luar itu, ada batasan praktis dalam berapa lama URL itu bisa didapat. Banyak browser, seperti Internet Explorer, hanya mengizinkan sejumlah karakter tertentu di URL, jadi jika formulir kita atau data yang dikirim terlalu panjang, maka itu tidak akan berfungsi. Saat kita menggunakan POST, tidak ada batasan panjang yang ditetapkan oleh browser. Namun, penting untuk dicatat bahwa pengguna masih dapat melihat data formulir dan bagaimana data itu akan dikirim ke server; kita tidak dapat menyembunyikannya dari pengguna, apa pun metode yang kita gunakan. Untuk sebagian besar formulir, saya menggunakan POST kecuali ada alasan khusus untuk menggunakan metode GET.

Menambahkan input teks

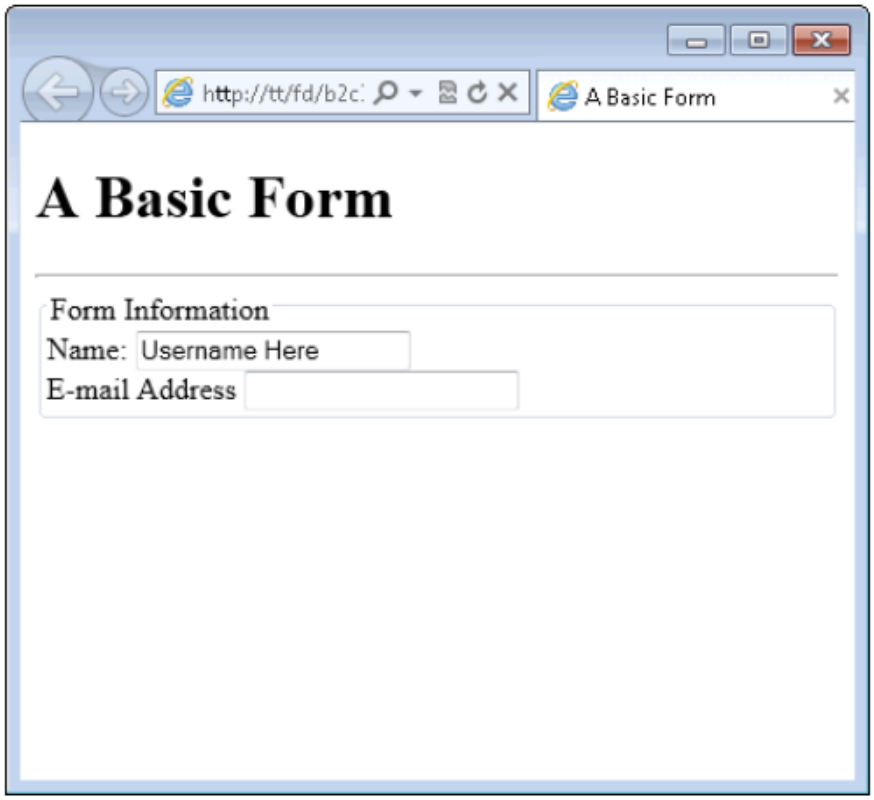
Anda telah melihat input teks dalam bab ini. Menambahkan satu semudah menggunakan jenis "teks". kita juga dapat menambahkan beberapa atribut, ukuran, dan ukuran maksimal yang lebih praktis, yang memberi tahu browser seberapa besar untuk membuat kotak teks di layar dan jumlah karakter maksimum yang diizinkan di bidang.
Sebagai contoh:

<input type=”text” name=”username” size=”20” maxsize=”30”>

HTML ini membuat kotak input selebar 20 karakter, dan paling banyak yang bisa dimasukkan seseorang ke dalam kotak adalah 30 karakter. Atribut lain yang mungkin kita lihat adalah atribut value, yang mengisi kolom dengan nilai yang kita berikan. Perhatikan contoh HTML ini:

<input type=”text” name=”username” value=”Username Here”>

Menambahkannya ke formulir menghasilkan formulir seperti yang ditunjukkan pada gambar dibawah. Perhatikan nilai di bidang Nama sekarang diatur sesuai dengan properti nilai dalam definisi <input>.



Gambar 3.3 Menambahkan nilai ke bidang.

Menambahkan kotak drop-down

Sebuah kotak drop-down, juga dikenal sebagai kotak pilih, menyajikan banyak pilihan, dari mana pengguna dapat memilih satu. Contohnya adalah daftar negara bagian, seperti Salatiga, Ungaran, Semarang, dan seterusnya, di mana pengguna biasanya memilih salah satu dari daftar tersebut. Kotak drop-down menyediakan cara yang baik untuk menampilkan informasi tersebut. kita membuat drop-down menggunakan elemen <select> bersama dengan elemen <option>, seperti ini:

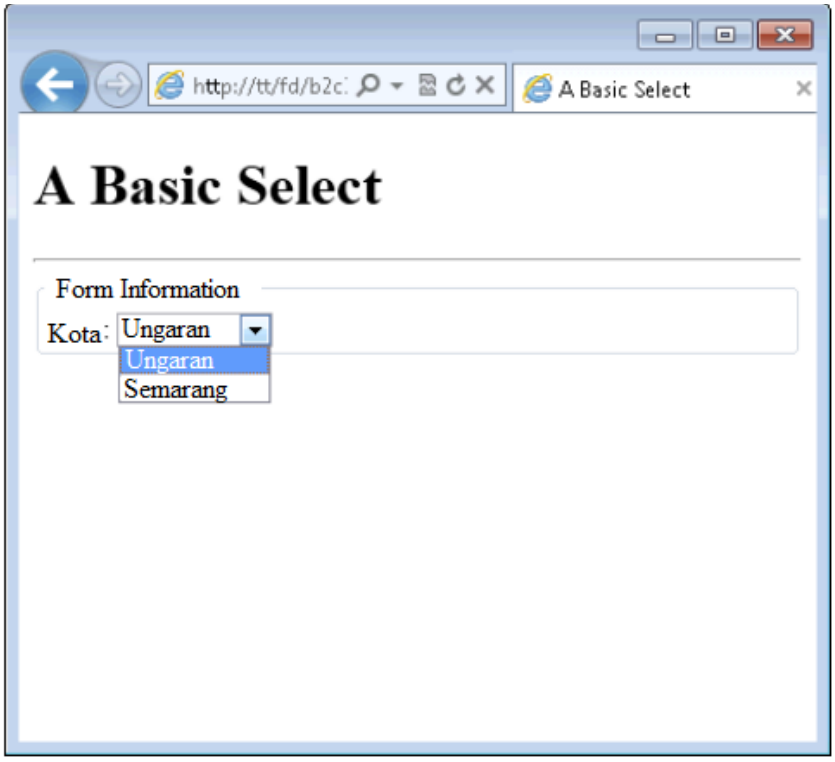
```
<select name=”state”>
 <option value=”CA”>Ungaran</option>
 <option value=”WI”>Semarang</option>
</select>
```

Berikut formulir lengkap dengan drop-down yang ditambahkan ke dalamnya:

```
<!doctype html>
<html>
<head>
<title>A Basic Select</title>
</head>
<body>
<h1>A Basic Select</h1>
<hr>
```

```
<form action=”#”>
<fieldset>
 <legend>Form Information</legend>
<div>
 <label for=”state”>State:</label>
 <select id=”state” name=”state”>
 <option value=”CA”>Ungaran</option>
 <option value=”WI”>Semarang</option>
 </select>
 </div>
</fieldset>
</form>
</body>
</html>
```

Ketika dilihat di browser, maka akan terlihat seperti gambar berikut:



Gambar 3.4 Membuat kotak drop-down pilih

Saat kotak drop-down ditampilkan, elemen pertama adalah elemen yang muncul sebagai default. Dalam contoh yang ditunjukkan pada gambar diatas, Ungaran ditampilkan sebagai opsi default. Namun, kita dapat mengubah nilai default dengan dua cara berbeda, seperti yang dibahas di sini.

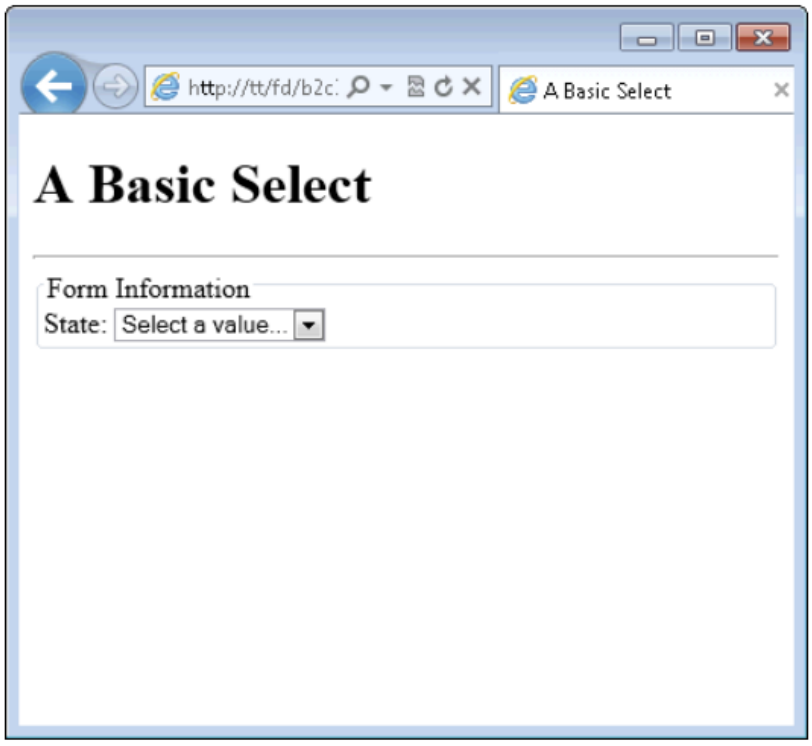
Seperti kotak teks, kita dapat menetapkan nilai default untuk kotak drop-down. Ini dicapai dengan menggunakan atribut yang dipilih. Meskipun tidak selalu diperlukan, sebaiknya tetapkan nilai untuk atribut yang dipilih, seperti dalam contoh ini yang akan mengubah nilai default ke Semarang:

```
<select name=”state”>
```

```
 <option value=”UN”>Ungaran</option>
 <option selected=”selected” value=”SM”>Semarang</option>
</select>
```

Cara lain untuk menetapkan jenis nilai default adalah dengan menetapkan opsi kosong sebagai opsi pertama dalam daftar. Meskipun secara teknis ini bukan nilai default, ini muncul pertama kali dalam daftar sehingga akan ditampilkan sebagai opsi default saat pengguna memuat halaman. Cara umum kita akan melihat ini adalah dengan menggunakan "Pilih nilai" atau kata-kata serupa sebagai opsi pertama, yang menunjukkan kepada pengguna bahwa ada beberapa tindakan yang diperlukan, seperti yang ditunjukkan di sini dan gambar berikut:

```
<select name=”state”>
 <option value=””>Select a value...</option>
 <option value=”CA”>Ungaran</option>
 <option value=”WI”>Semarang</option>
 </select>
```



Gambar 3.5 Mengatur nilai pertama untuk drop-down.

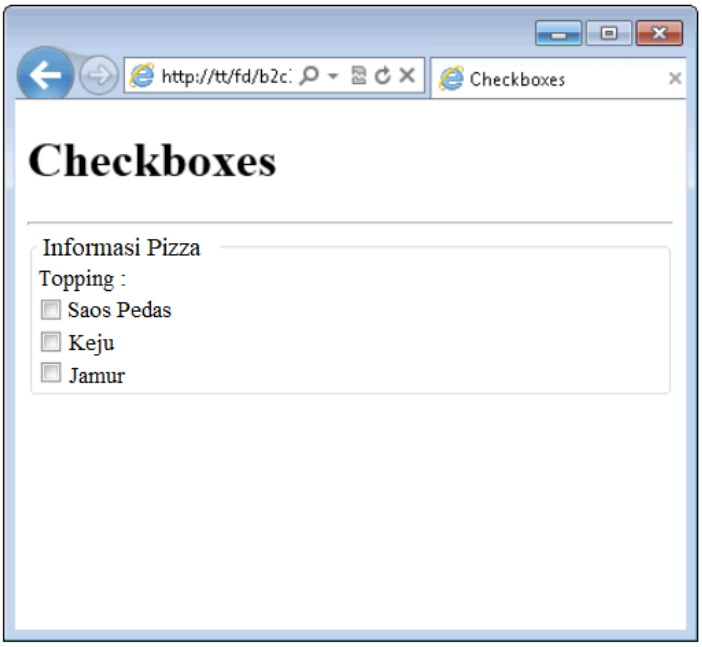
Menggunakan atribut yang dipilih mengesampingkan trik nilai pertama yang ditunjukkan dalam contoh ini.

Membuat check box

Cara lain untuk mewakili beberapa nilai adalah dengan menggunakan kotak centang. Di mana *drop-down* bagus untuk mewakili beberapa nilai saat ada banyak opsi, kotak centang bagus untuk mewakili beberapa nilai saat hanya ada beberapa opsi, seperti yang mungkin terjadi saat membuat formulir untuk memilih topping pizza. Ketika seseorang menambahkan topping pizza, dia dapat memilih lebih dari satu pada pizzanya, tetapi biasanya toppingnya tidak terlalu banyak.

```
<!doctype html>
<html>
<head>
<title>Checkboxes</title>
</head>
<body>
<h1>Checkboxes</h1>
<hr>
<form action=”#”>
<fieldset>
 <legend>Pizza Information</legend>
 <div>Toppings: <br />
 <input type=”checkbox” id=”sausage”
 name=”toppings” value=”sausage”>
 <label for=”sausage”>Sausage</label><br />
 <input type=”checkbox” id=”pep”
 name=”toppings” value=”pep”>
 <label for=”pep”>Pepperoni</label><br />
 <input type=”checkbox” id=”mush”
 name=”toppings” value=”mush”>
 <label for=”mush”>Mushrooms</label><br />
 </div>
</fieldset>
</form>
</body>
</html>
```

HTML ini membuat tiga kotak centang dalam grup yang disebut "topping". Halaman yang dihasilkan akan terlihat seperti gambar berikut:



Gambar 3.6 Menggunakan kotak centang untuk input.

Perhatikan di HTML bahwa setiap kotak centang memiliki atribut nama yang sama tetapi menggunakan atribut value yang berbeda dan atribut id yang berbeda. Atribut id harus unik agar HTML valid (dan agar label berfungsi dengan benar). Namanya sama karena kotak centang sebenarnya dikelompokkan bersama; mereka mewakili satu jenis informasi: topping pizza.

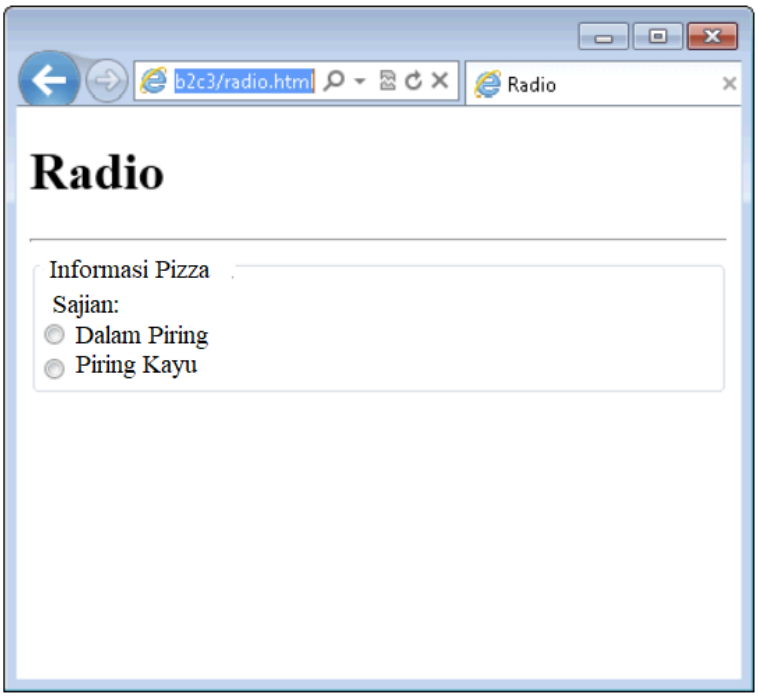
Dalam praktiknya, kita mungkin melihat kotak centang tanpa atribut nama atau dengan atribut nama yang berbeda untuk setiap kotak centang. Contoh yang kita lihat di sini adalah salah satu yang membuat informasi dikelompokkan secara logis, yang membuatnya lebih mudah untuk dipelihara nanti dan juga memudahkan untuk bekerja dengan program server.

Menggunakan tombol radio

Tombol radio digunakan di mana ada beberapa nilai tetapi pengguna hanya dapat memilih satu dari opsi tersebut, seperti halnya dengan jenis kerak untuk pizza. Keraknya bisa berupa piring tipis atau dalam - tetapi tidak keduanya atau pizza akan benar-benar berantakan. Berikut HTML untuk membuat tombol radio. Perhatikan bahwa HTML tidak terlalu jauh berbeda dari contoh kotak centang:

```
<!doctype html>
<html>
<head>
<title>Radio</title>
</head>
<body>
<h1>Radio</h1>
<hr>
<form action="#">
<fieldset>
 <legend>Pizza Information</legend>
 <div>Crust: <br />
 <input type="radio" id="deep"
 name="crust" value="deep">
 <label for="deep">Deep Dish</label><br />
 <input type="radio" id="thin"
 name="crust" value="thin">
 <label for="thin">Thin</label><br />
 </div>
</fieldset>
</form>
</body>
</html>
```

Jika dilihat di browser, hasilnya seperti gambar berikut:



Gambar 3.7 Tombol radio pada halaman web.

Seperti kotak centang, tombol radio memiliki nama yang sama tetapi menggunakan atribut nilai dan id yang berbeda. Seperti kotak centang, tombol radio menggunakan nilai ini untuk alasan yang sama. Dengan tombol radio, atribut nama bahkan lebih penting. Tombol radio yang memiliki atribut nama yang sama berada dalam grup yang sama, artinya pengguna hanya dapat memilih salah satu opsi dalam grup tersebut. Jika kita ingin pengguna dapat memilih lebih dari satu opsi, kita mungkin harus menggunakan kotak centang. Namun, kita dapat menggunakan lebih dari satu grup tombol radio pada satu halaman. Cukup gunakan nama yang berbeda untuk grup tombol radio baru dan pengguna akan dapat memilih dari grup itu juga.

Menyerahkan dan membersihkan formulir

Kita telah melihat beberapa jenis input yang dapat kita gunakan di halaman web untuk mengumpulkan informasi.

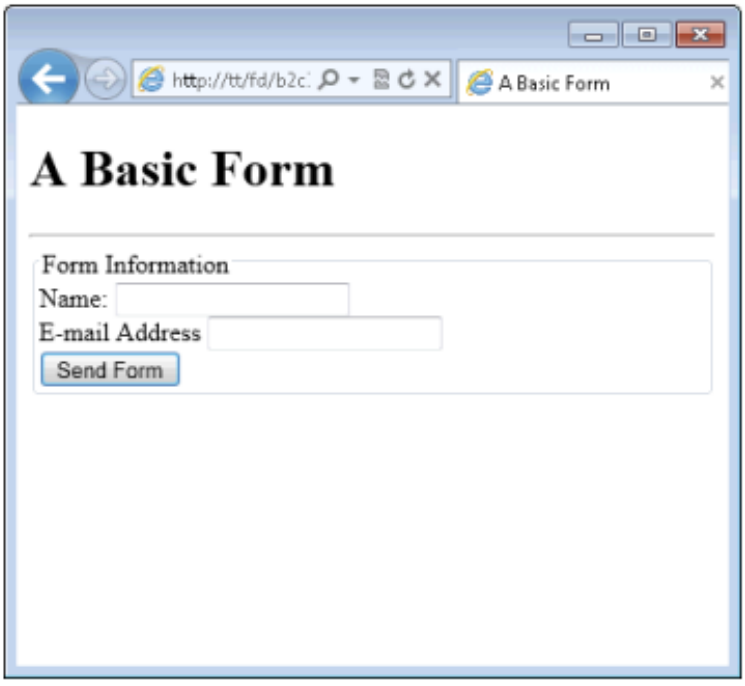
```
<input type=”submit” name=”submit”
 value=”Process Request”>
```

Misalnya, pertimbangkan contoh ini, di mana tombol Kirim ditambahkan ke formulir yang kita lihat sebelumnya di bab ini:

```
<!doctype html>
<html>
<head>
<title>A Basic Form</title>
</head>
<body>
<h1>A Basic Form</h1>
<hr>
```

```
<form action=”#”>
<fieldset>
 <legend>Form Information</legend>
 <div>
 <label for=”username”>Name:</label>
 <input type=”text” id=”username” name=”username”>
 </div>
 <div>
<label for=”email”>E-mail Address:</label>
 <input type=”text” id=”username” name=”email”>
 </div>
 <div>
 <input type=”submit” name=”submit”
 value=”Send Form”>
 </div>
</fieldset>
</form>
</body>
</html>
```

Hasil dari HTML ini akan terlihat seperti gambar berikut:



Gambar 3.8 menambahkan tombil submit

Tombol lain yang kita lihat di formulir adalah tombol Deleted atau Reset. Tombol Reset menghapus input dan mengatur ulang formulir, menghapus apa pun yang telah dimasukkan pengguna ke dalam formulir. Menambahkan tombol Reset semudah menambahkan jenis input "reset":

```
<input type=”reset” name=”reset” value=”Clear Form”>
```

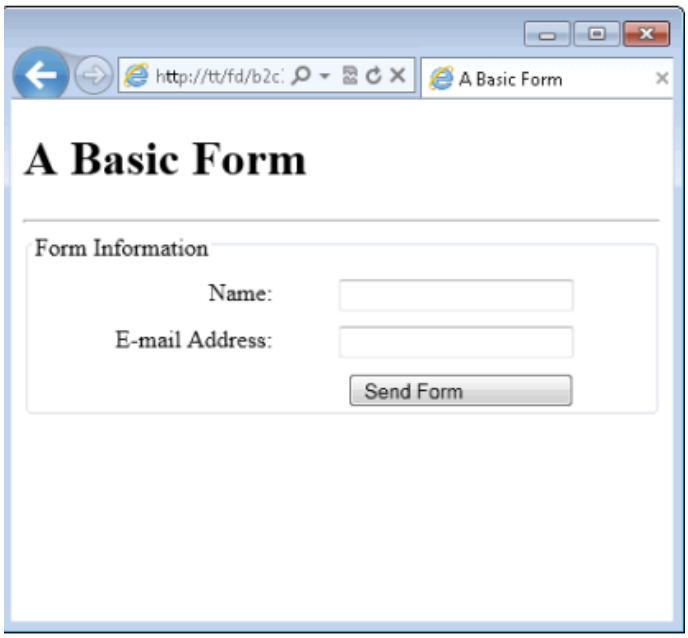
3.4 MENGGUNAKAN CSS UNTUK MENYEJAJARKAN BIDANG FORMULIR

Contoh formulir yang kita lihat sejauh ini cukup membosankan, hanya HTML biasa tanpa keselarasan atau daya tarik visual. Formulir hanyalah HTML standar, sehingga dapat ditata menggunakan CSS. Bagian ini melihat bagaimana melakukan hal itu. Contoh yang akan kita lihat di bagian ini menggunakan CSS tepat di dalam file HTML. Ini dilakukan untuk kesederhanaan.

Saat menyelaraskan bidang formulir, kuncinya adalah menggunakan HTML yang terstruktur dengan baik. HTML yang telah kita lihat sejauh ini dalam bab ini sesuai dengan tagihan dan menyelaraskan bidang formulir akan lebih mudah. Sebenarnya, menggunakan HTML dari contoh terakhir sebagai panduan, cukup menambahkan informasi gaya ini ke bagian <head> untuk menyelaraskan bidang:

```
<style type=”text/css”>
.form-field {
 clear: both;
 padding: 10px;
 width: 350px;
}
.form-field label {
 float: left;
 width: 150px;
 text-align: right;
}
.form-field input {
 float: right;
 width: 150px;
 text-align: left;
}
</style>
```

Hasilnya dari HTML ini ditunjukkan pada dibawah. Setiap aturan gaya cocok menggunakan kelas CSS dan, dalam hal label dan input, selector anak selanjutnya digunakan untuk mempersempit penerapan aturan CSS.



Gambar 3.9 Menyelaraskan bidang formulir dengan CSS

Tapi tunggu! Tombol Kirim Formulir sekarang direntangkan hingga lebar 150px dan teks (“Send Form”) disejajarkan dengan sisi kiri tombol:

```
.form-field input {
 float: right;
 width: 150px;
 text-align: left;
}
```

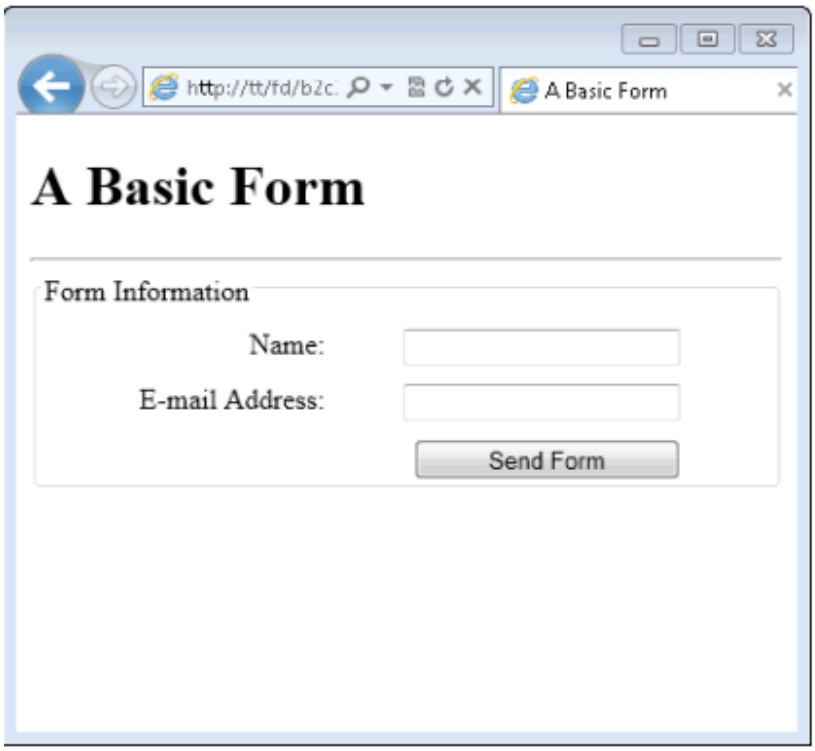
Anda memerlukan cara untuk membuat tombol itu lebih kecil atau setidaknya menyelaraskan teks di tengahnya. Steve secara pribadi menyukai tombol yang lebih besar. Mereka memudahkan pengguna untuk mengklik atau mengetuk, jika mereka menggunakan perangkat seluler. Jadi kami memilih untuk menyelaraskan teks di tengah tetapi membiarkan tombol dengan ukuran yang sama. Menyejajarkannya di tengah berarti menambahkan sesuatu ke HTML tombol Kirim agar dapat mengaksesnya di dalam CSS. Cara termudah untuk melakukannya adalah dengan menambahkan atribut id ke tombol Kirim, seperti:

```
<input id=”submit” type=”submit” name=”submit”
 value=”Send Form”>
```

Berikut CSS untuk ditambahkan:

```
#submit {
 text-align: center;
}
```

Hasilnya ditunjukkan seperti gambar berikut ini:



Gambar 3.10 Menyejajarkan teks dari tombol Kirim.

```
<!doctype html>
<html>
<head>
<title>A Basic Form</title>
<style type=”text/css”>
.form-field {
 clear: both;
 padding: 10px;
 width: 350px;
}
.form-field label {
 float: left;
 width: 150px;
 text-align: right;
}
.form-field input {
 float: right;
 width: 150px;
 text-align: left;
}
#submit {
 text-align: center;
}
</style>
</head>
<body>
```

```
<h1>A Basic Form</h1>
<hr>
<form action=”#”>
<fieldset>
 <legend>Form Information</legend>
 <div class=”form-field”>
 <label for=”username”>Name:</label>
 <input type=”text” id=”username” name=”username”>
 </div>
 <div class=”form-field”>
 <label for=”email”>E-mail Address:</label>
 <input type=”text” id=”username” name=”email”>
 </div>
 <div class=”form-field”>
 <input id=”submit” type=”submit” name=”submit”
value=”Send Form”>
 </div>
</fieldset>
</form>
</body>
</html>
```

BAB 4
CSS TINGKAT LANJUT DENGAN CSS3

Implementasi pertama dari CSS dibuat pada tahun 1996, dirilis pada tahun 1999, dan telah didukung oleh semua rilis browser sejak tahun 2001. Standar untuk versi ini, CSS1, direvisi pada tahun 2008. Mulai tahun 1998, pengembang mulai menyusun yang kedua spesifikasi, CSS2; standarnya selesai pada tahun 2007 dan direvisi pada tahun 2009. Pengembangan spesifikasi CSS3 dimulai pada tahun 2001, dengan beberapa fitur yang diusulkan baru-baru ini pada tahun 2009. Oleh karena itu, proses pengembangan kemungkinan akan berlanjut selama beberapa waktu sebelum rekomendasi akhir untuk CSS3 disetujui. Dan meskipun CSS3 belum selesai, orang-orang sudah mulai mengajukan saran untuk CSS4. Dalam bab ini, saya akan membawa kita melalui fitur CSS3 yang secara umum telah diadopsi oleh browser utama. Beberapa fitur ini menyediakan fungsionalitas yang sampai sekarang hanya dapat disediakan dengan JavaScript. Saya sarankan menggunakan CSS3 untuk menerapkan fitur dinamis di mana kita bisa, bukan JavaScript. Fitur-fitur yang disediakan oleh CSS membuat atribut dokumen menjadi bagian dari dokumen itu sendiri, alih-alih ditempelkan melalui JavaScript. Menjadikannya bagian dari dokumen adalah desain yang lebih bersih.

4.1 SELEKTOR ATRIBUT

Pada bab sebelumnya, saya merinci berbagai selektor atribut CSS, yang sekarang akan saya rangkum dengan cepat. Selector digunakan dalam CSS untuk mencocokkan elemen HTML, dan ada 10 jenis yang berbeda, seperti yang dijelaskan pada Tabel 4.1.

Tabel 4.1 Selektor CSS, pseudeclass, dan Pseudo-element

Tipe Selektor	Contoh
Universal selector	* { color:#555; }
Type selectors	b { color:red; }
Class selectors	.classname { color:blue; }
ID selectors	#idname { background:cyan; }
Descendant selectors	span em { color:green; }
Child selectors	div > em { background:lime; }
Adjacent sibling selectors	i + b { color:gray; }
Attribute selectors	a[href='info.htm'] { color:red; }
Pseudo-classes	a:hover { font-weight:bold; }
Pseudo-elements	P::first-letter { font-size:300%; }

Perancang CSS3 memutuskan bahwa sebagian besar penyeleksi ini berfungsi dengan baik sebagaimana adanya, tetapi mereka membuat tiga peningkatan sehingga kita dapat lebih mudah mencocokkan elemen berdasarkan konten atributnya. Misalnya, di CSS2, kita dapat menggunakan selektor seperti a[href='info.htm'] untuk mencocokkan string info.htm saat ditemukan di atribut href, tetapi tidak ada cara untuk mencocokkan hanya sebagian dari string. Di situlah tiga operator baru CSS3—^, $, dan *—datang untuk menyelamatkan. Jika salah satu secara langsung mendahului simbol = , kita dapat mencocokkan awal, akhir, atau bagian mana pun dari string, masing-masing.

Operator ^

Operator ini cocok di awal string jadi, misalnya, berikut ini akan cocok dengan atribut href yang nilainya dimulai dengan string http://website:

```
a[href^='http://website']
```

Oleh karena itu, elemen berikut akan cocok:

```
<a href='http://website.com'>
```

Tapi ini tidak akan:

```
<a href='http://mywebsite.com'>
```

Operator $

Untuk mencocokkan hanya di akhir string, kita dapat menggunakan selektor seperti berikut ini, yang akan cocok dengan semua tag img yang atribut src diakhiri dengan .png:

```
img[src$='.png']
```

Misalnya, berikut ini akan cocok:

```
<img src='photo.png'>
```

Tapi ini tidak akan:

```
<img src='snapshot.jpg'>
```

Operator *

Untuk mencocokkan substring mana pun dalam atribut, kita dapat menggunakan selektor seperti berikut ini untuk menemukan tautan apa pun di halaman yang memiliki string google di mana saja di dalamnya:

```
a[href*='google']
```

Misalnya, segmen HTML <a href='http://google.com'> akan cocok, sedangkan segmen <a href='http://gmail.com'> tidak akan cocok.

4.2 PROPERTI BOX-SIZING

Model kotak W3C menetapkan bahwa lebar dan tinggi suatu objek harus merujuk hanya ke dimensi konten elemen, mengabaikan padding atau batas apa pun. Tetapi beberapa desainer web telah menyatakan keinginan untuk menentukan dimensi yang merujuk ke seluruh elemen, termasuk padding dan batas. Untuk menyediakan fitur ini, CSS3 memungkinkan kita memilih model kotak yang ingin kita gunakan dengan properti ukuran kotak. Misalnya, untuk menggunakan lebar dan tinggi total suatu objek termasuk padding dan border, kita akan menggunakan deklarasi ini:

```
box-sizing:border-box;
```

Atau, agar lebar dan tinggi objek hanya merujuk ke kontennya, kita akan menggunakan deklarasi ini (default):

box-sizing:content-box;

Catatan: Browser berbasis Safari dan Mozilla (seperti Firefox) memerlukan awalan tersendiri untuk deklarasi ini (-webkit- dan -moz-). Untuk detail lebih lanjut, lihat http://caniuse.com.

4.3 BACKGROUND CSS3

CSS3 menyediakan dua properti baru: background-clip dan background-origin. Di antara mereka, kita dapat menentukan di mana latar belakang harus dimulai dalam sebuah elemen, dan cara memotong latar belakang sehingga tidak muncul di bagian model kotak yang tidak kita inginkan. Untuk mencapai ini, kedua properti mendukung nilai-nilai berikut: border-box Mengacu ke tepi luar dari border padding-box Mengacu ke tepi luar dari kotak isi area padding Mengacu pada tepi luar dari area konten.

Properti Background-clip

Properti background-clip menentukan apakah latar belakang harus diabaikan (terpotong) jika muncul di dalam batas atau area padding elemen. Misalnya, deklarasi berikut menyatakan bahwa latar belakang dapat ditampilkan di semua bagian elemen, sampai ke tepi luar batas:

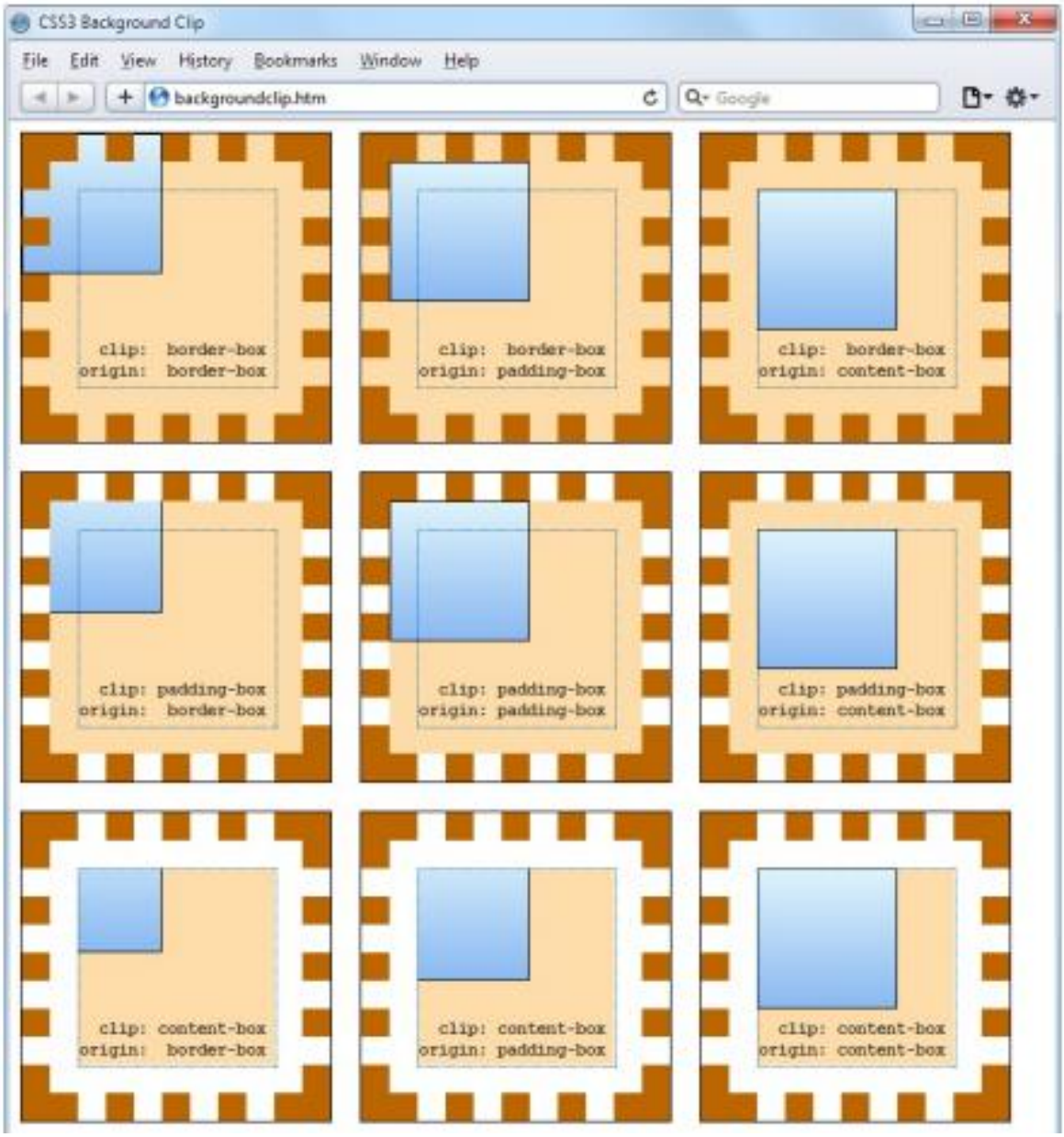
background-clip:border-box;

Agar latar belakang tidak muncul di dalam area batas elemen, kita dapat membatasinya hanya pada bagian elemen di dalam tepi luar area paddingnya, seperti ini:
background-clip:padding-box;
Atau untuk membatasi latar belakang agar hanya ditampilkan di dalam area konten suatu elemen, kita akan menggunakan deklarasi ini:

background-clip:content-box;

Gambar 4.1 menunjukkan tiga baris elemen yang ditampilkan di browser web Safari, di mana baris pertama menggunakan kotak batas untuk properti klip latar, baris kedua menggunakan kotak pengisi, dan baris ketiga menggunakan kotak konten. Di baris pertama, kotak dalam (file gambar yang telah dimuat ke kiri atas elemen, dengan pengulangan dinonaktifkan) diizinkan untuk ditampilkan di mana saja di elemen. kita juga dapat melihatnya dengan jelas ditampilkan di area border kotak pertama karena border telah disetel ke titik-titik.

Pada baris kedua, baik gambar latar belakang maupun bayangan latar belakang tidak ditampilkan di area border, karena mereka telah dijepitkan ke area padding dengan nilai properti background-clip dari padding-box. Kemudian, di baris ketiga, bayangan latar belakang dan gambar telah dipotong untuk ditampilkan hanya di dalam area konten dalam setiap elemen (ditampilkan di dalam kotak putus-putus berwarna terang), menggunakan properti klip latar belakang kotak konten.



Gambar 4-1 Berbagai cara menggabungkan properti latar belakang CSS3

Properti Background-origin

Dengan properti background-origin, kita dapat mengontrol lokasi gambar latar belakang dengan menentukan di mana kiri atas gambar harus dimulai. Misalnya, deklarasi berikut menyatakan bahwa asal gambar latar belakang harus berada di sudut kiri atas tepi luar batas:

background-origin:border-box;

Untuk mengatur asal gambar ke pojok kiri atas area padding, kita akan menggunakan deklarasi ini:

background-origin:padding-box;

Atau untuk menyetel asal gambar ke sudut kiri atas bagian konten dalam elemen, kita akan menggunakan deklarasi ini:

background-origin:content-box;

Melihat lagi pada Gambar 20-1, kita dapat melihat di setiap baris kotak pertama menggunakan properti background-origin dari border-box, yang kedua menggunakan padding-box, dan yang ketiga menggunakan content-box. Akibatnya, di setiap baris kotak dalam yang lebih kecil ditampilkan di kiri atas border di kotak pertama, kiri atas padding di kotak kedua, dan kiri atas konten di kotak ketiga.

Properti ukuran latar belakang

Dengan cara yang sama seperti kita dapat menentukan lebar dan tinggi gambar saat digunakan dalam tag <img>, kini kita juga dapat melakukannya untuk gambar latar di versi terbaru semua browser. kita menerapkan properti sebagai berikut (di mana ww adalah lebar dan hh adalah tinggi):

background-size:wwpx hhpx;

Jika mau, kita hanya dapat menggunakan satu argumen, lalu kedua dimensi akan disetel ke nilai tersebut. Juga, jika kita menerapkan properti ini ke elemen level blok seperti (bukan yang sebaris seperti), kita dapat menentukan lebar dan/atau tinggi sebagai persentase, alih-alih a nilai tetap. Jika kita ingin menskalakan hanya satu dimensi dari gambar latar belakang, dan kemudian membuat yang lain menskalakan secara otomatis untuk mempertahankan proporsi yang sama, kita dapat menggunakan nilai otomatis untuk dimensi lain, seperti ini:

background-size:100px auto;

Ini menetapkan lebar ke 100 piksel, dan tinggi ke nilai yang sebanding dengan penambahan atau pengurangan lebar.

Beberapa Latar Belakang

Dengan CSS3 kita sekarang dapat melampirkan beberapa latar belakang ke sebuah elemen, yang masing-masing dapat menggunakan properti latar belakang CSS3 yang telah dibahas sebelumnya. Gambar 20-2 menunjukkan contohnya; delapan gambar berbeda telah ditetapkan ke latar belakang, untuk membuat empat sudut dan empat tepi batas sertifikat.

Gambar 4.2 Latar belakang dibuat dengan banyak gambar

Untuk menampilkan beberapa gambar latar belakang dalam satu deklarasi CSS, pisahkan dengan koma. Contoh 1 menunjukkan HTML dan CSS yang digunakan untuk membuat latar belakang pada Gambar 4.2.

Contoh 1 Menggunakan banyak gambar di latar belakang

```
<!DOCTYPE html>
<html> <!-- backgroundimages.html -->
<head>
<title>CSS3 Multiple Backgrounds Example</title>
<style>
.border {
font-family:'Times New Roman';
font-style :italic;
```

```
font-size :170%;
text-align :center;
padding :60px;
width :350px;
height :500px;
background :url('b1.gif') top left no-repeat,
url('b2.gif') top right no-repeat,
url('b3.gif') bottom left no-repeat,
url('b4.gif') bottom right no-repeat,
url('ba.gif') top repeat-x,
url('bb.gif') left repeaty,
url('bc.gif') right repeaty,
url('bd.gif') bottom repeat-x
}
</style>
</head>
<body>
<div class='border'>
<h1>Employee of the month</h1>
<h2>Awarded To:</h2>
<h3>_____</h3>
<h2>Date:</h2>
<h3>___/___/_____</h3>
</div>
</body>
</html>
```

Melihat bagian CSS, kita melihat bahwa empat baris pertama dari deklarasi latar belakang menempatkan gambar sudut ke dalam empat sudut elemen, dan empat terakhir menempatkan gambar tepi, yang ditangani terakhir karena urutan prioritas untuk gambar latar pergi dari tinggi ke rendah. Dengan kata lain, di mana mereka tumpang tindih, gambar latar belakang tambahan akan muncul di belakang gambar yang sudah ditempatkan. Jika GIF berada dalam urutan terbalik, gambar tepi berulang akan ditampilkan di atas sudut, yang akan salah.

4.4 BORDER CSS3

CSS3 juga memberikan lebih banyak fleksibilitas dalam cara menyajikan batas, dengan memungkinkan kita mengubah warna keempat tepi tepi secara mandiri, menampilkan gambar untuk tepi dan sudut, memberikan nilai radius untuk menerapkan sudut membulat ke tepi, dan untuk menempatkan bayangan kotak di bawah elemen.

Properti warna border

Ada dua cara kita dapat menerapkan warna ke border. Pertama, kita dapat meneruskan satu warna ke properti, sebagai berikut:

```
border-color:#888;
```

Properti ini menetapkan semua batas elemen menjadi abu-abu tengah. kita juga dapat mengatur warna batas satu per satu, seperti ini (yang mengatur warna batas ke berbagai warna abu-abu):

```
border-top-color :#000;
border-left-color :#444;
border-right-color :#888;
border-bottom-color:#ccc;
```

Anda juga dapat mengatur semua warna satu per satu dengan satu deklarasi, sebagai berikut:

```
border-color:#f00 #0f0 #880 #00f;
```

Deklarasi ini menetapkan warna batas atas menjadi #f00, yang kanan menjadi #0f0, yang bawah menjadi #880, dan yang kiri menjadi #00f (masing-masing merah, hijau, oranye, dan biru). kita juga dapat menggunakan nama warna untuk argumen.

Properti radius border

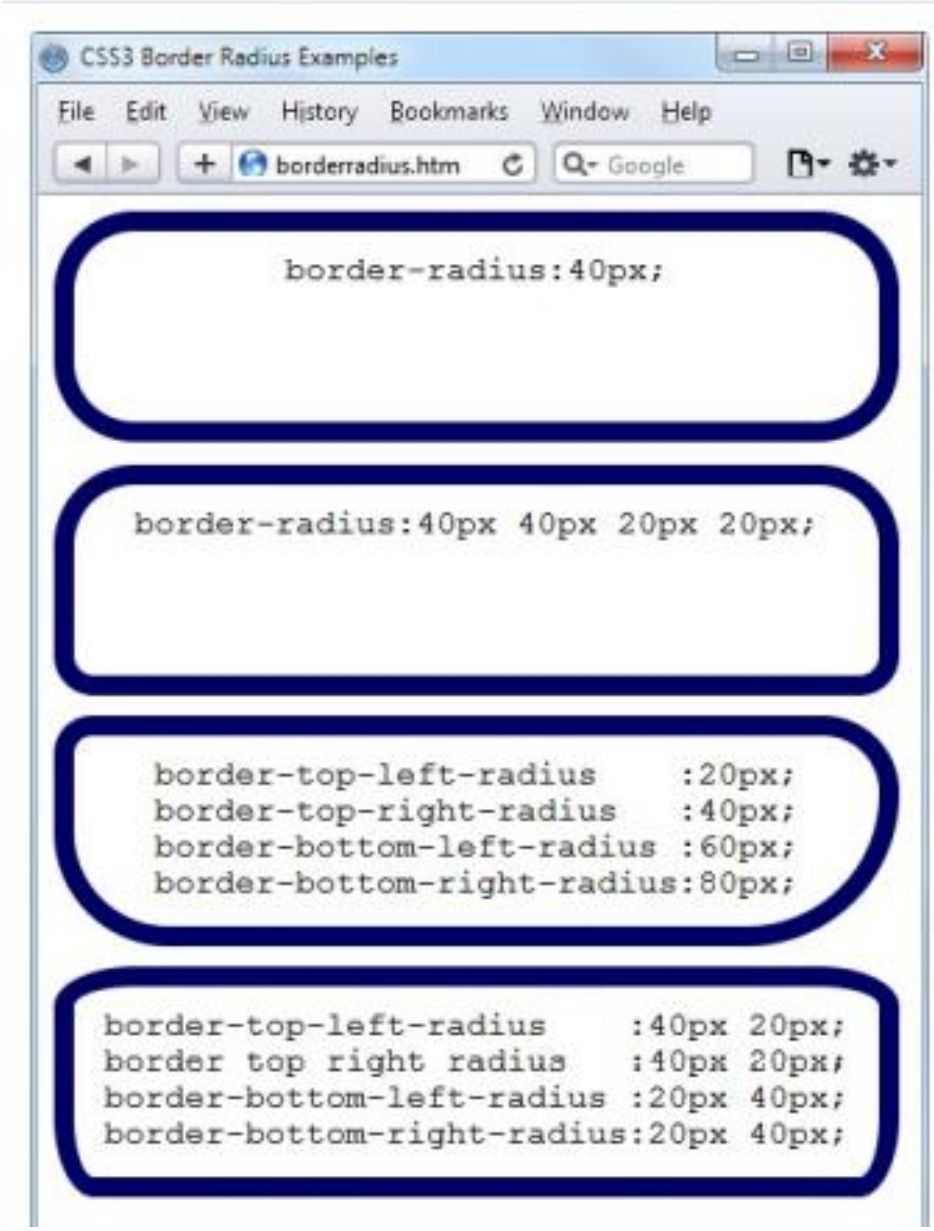
Sebelum CSS3, pengembang web berbakat datang dengan banyak penyesuaian dan perbaikan yang berbeda untuk mencapai batas bulat, umumnya menggunakan tag <table> atau . Tapi sekarang menambahkan batas bulat ke elemen sangat sederhana, dan ini berfungsi pada versi terbaru dari semua browser utama, seperti yang ditunjukkan pada Gambar 4.3, di mana batas 10-piksel ditampilkan dengan cara yang berbeda. Contoh 2 menunjukkan HTML untuk ini.

Contoh 2 Properti borderradius

```
<!DOCTYPE html>
<html> <!-- borderradius.html -->
<head>
<title>CSS3 Border Radius Examples</title>
<style>
.box {
margin-bottom:10px;
font-family :'Courier New', monospace;
font-size :12pt;
text-align :center;
padding :10px;
width :380px;
height :75px;
border :10px solid #006;
}
.b1 {
-moz-border-radius :40px;
-webkit-border-radius:40px;
borderradius :40px;
}
.b2 {
```

```
-moz-border-radius :40px 40px 20px 20px;
-webkit-border-radius:40px 40px 20px 20px;
borderradius :40px 40px 20px 20px;
}
.b3 {
-moz-border-radius-topleft :20px;
-moz-border-radius-topright :40px;
-moz-border-radius-bottomleft :60px;
-moz-border-radius-bottomright :80px;
-webkit-border-topleft-radius :20px;
-webkit-border-topright-radius :40px;
-webkit-border-bottomleft-radius :60px;
-webkit-border-bottomright-radius:80px;
border-top-left-radius :20px;
border-top-right-radius :40px;
border-bottom-left-radius :60px;
border-bottom-right-radius :80px;
}
.b4 {
-moz-border-radius-topleft :40px 20px;
-moz-border-radius-topright :40px 20px;
-moz-border-radius-bottomleft :20px 40px;
-moz-border-radius-bottomright :20px 40px;
-webkit-border-topleft-radius :40px 20px;
-webkit-border-topright-radius :40px 20px;
-webkit-border-bottomleft-radius :20px 40px;
-webkit-border-bottomright-radius:20px 40px;
border-top-left-radius :40px 20px;
border-top-right-radius :40px 20px;
border-bottom-left-radius :20px 40px;
border-bottom-right-radius :20px 40px;
}
</style>
</head>
<body>
<div class='box b1'>
borderradius:40px;
</div>
<div class='box b2'>
borderradius:40px 40px 20px 20px;
</div>
<div class='box b3'>
border-top-left-radius    :20px;<br>
border-top-right-radius   :40px;<br>
border-bottom-left-radius :60px;<br>
border-bottom-right-radius:80px;
```

```
</div>
<div class='box b4'>
border-top-left-radius    :40px  20px;<br>
border-top-right-radius   :40px  20px;<br>
border-bottom-left-radius :20px 40px;<br>
border-bottom-right-radius:20px  40px;
</div>
</body>
</html>
```



Gambar 4.3 Mencampur dan mencocokkan berbagai properti radius border

Jadi, misalnya, untuk membuat batas bulat dengan radius 20 piksel, kita cukup menggunakan deklarasi berikut:

```
borderradius:20
```

Anda dapat menentukan radius terpisah untuk masing-masing dari empat sudut, seperti ini (diterapkan searah jarum jam mulai dari sudut kiri atas):

borderradius:10px 20px 30px 40px;

Jika mau, kita juga dapat menangani setiap sudut elemen satu per satu, seperti ini:

border-top-left-radius :20px;
border-top-right-radius :40px;
border-bottom-left-radius :60px;
border-bottom-right-radius:80px;

Dan, saat mereferensikan setiap sudut, kita dapat memberikan dua argumen untuk memilih radius vertikal dan horizontal yang berbeda (memberikan batas yang lebih menarik dan halus) seperti ini:

border-top-left-radius :40px 20px;
border-top-right-radius :40px 20px;
border-bottom-left-radius :20px 40px;
border-bottom-right-radius:20px 40px;

Argumen pertama adalah horizontal, dan yang kedua adalah radius vertikal.

4.5 BAYANGAN KOTAK

Untuk menerapkan bayangan kotak, tentukan offset horizontal dan vertikal dari objek, jumlah pengaburan untuk ditambahkan ke bayangan, dan warna yang akan digunakan, seperti ini:

box-shadow:15px 15px 10px #888;

Dua contoh 15px menentukan offset vertikal dan horizontal dari elemen, dan nilai ini bisa negatif, nol, atau positif. 10px menentukan jumlah keburaman, dengan nilai yang lebih kecil menghasilkan keburaman yang lebih sedikit. Dan #888 adalah warna untuk bayangan, yang dapat berupa nilai warna apa pun yang valid. Hasil dari deklarasi ini dapat dilihat pada Gambar 4.4.

Gambar 4.4 Bayangan kotak ditampilkan di bawah elemen

4.6 ELEMEN OVERFLOW

Di CSS2, kita dapat menunjukkan apa yang harus dilakukan ketika satu elemen terlalu besar untuk sepenuhnya ditampung oleh induknya dengan menyetel properti overflow ke hidden, visible, scroll, atau auto. Tetapi dengan CSS3, kita sekarang dapat menerapkan nilai-nilai ini secara terpisah dalam arah horizontal atau vertikal juga, seperti pada contoh deklarasi berikut:

```
overflow-x:hidden;
overflow-x:visible;
overflow-y:auto;
overflow-y:scroll;
```

Layout Multikolom

Salah satu fitur yang paling banyak diminta oleh pengembang web adalah beberapa kolom, dan ini akhirnya direalisasikan di CSS3, dengan Internet Explorer 10 menjadi browser utama terakhir yang mengadopsinya. Sekarang, mengalirkan teks ke beberapa kolom semudah menentukan jumlah kolom, lalu (secara opsional) memilih jarak antara kolom dan jenis garis pemisah (jika ada).

Gambar 4.5 Teks mengalir dalam beberapa kolom

Contoh Menggunakan CSS untuk membuat banyak kolom

```
<!DOCTYPE html>
<html> <!-- multiplecolumns.html -->
<head>
<title>Multiple Columns</title>
<style>
.columns {
text-align :justify;
font-size :16pt;
-moz-column-count :3;
-moz-column-gap :1em;
-moz-column-rule :1px solid black;
-webkit-column-count:3;
-webkit-column-gap :1em;
-webkit-column-rule :1px solid black;
column-count :3;
column-gap :1em;
column-rule :1px solid black;
}
</style>
</head>
<body>
<div class='columns'>
Now is the winter of our discontent
Made glorious summer by this sun of York;
And all the clouds that lour'd upon our house
In the deep bosom of the ocean buried.
Now are our brows bound with victorious wreaths;
Our bruised arms hung up for monuments;
Our stern alarums changed to merry meetings,
Our dreadful marches to delightful measures.
Grim-visaged war hath smooth'd his wrinkled front;
And now, instead of mounting barded steeds
```

```
To fright the souls of fearful adversaries,
He capers nimbly in a lady's chamber
To the lascivious pleasing of a lute.
</div>
</body>
</html>
```

Di dalam kelas .columns, dua baris pertama hanya memberi tahu browser untuk membenarkan teks dengan benar dan mengaturnya ke ukuran font 16pt. Deklarasi ini tidak diperlukan untuk beberapa kolom, tetapi mereka meningkatkan tampilan teks. Baris yang tersisa mengatur elemen sehingga, di dalamnya, teks akan mengalir di atas tiga kolom, dengan jarak 1em di antara kolom, dan dengan batas piksel tunggal di tengah setiap celah.

4.7 WARNA DAN OPACITY

Cara kita dapat menentukan warna telah sangat diperluas dengan CSS3, dan sekarang kita juga dapat menggunakan fungsi CSS untuk menerapkan warna dalam format umum RGB (Merah, Hijau, dan Biru), RGBA (Merah, Hijau, Biru, dan Alfa).), HSL (Hue, Saturation, dan Luminance), dan HSLA (Hue, Saturation, Luminance, dan Alpha). Nilai Alpha menentukan transparansi warna, yang memungkinkan elemen yang mendasarinya terlihat.

Warna HSL

Untuk menentukan warna dengan fungsi hsl, kita harus terlebih dahulu memilih nilai rona antara 0 dan 359 dari roda warna. Setiap angka warna yang lebih tinggi cukup membungkus ke awal lagi, sehingga nilai 0 adalah merah, dan begitu juga nilai 360 dan 720. Dalam roda warna, warna utama merah, hijau, dan biru dipisahkan oleh 120 derajat, sehingga merah murni adalah 0, hijau adalah 120, dan biru adalah 240. Angka-angka di antara nilai-nilai ini mewakili nuansa yang terdiri dari proporsi berbeda dari warna primer di kedua sisi.

Selanjutnya kita membutuhkan tingkat saturasi, yaitu nilai antara 0% dan 100%. Ini menentukan bagaimana warna yang pudar atau cerah akan muncul. Nilai saturasi dimulai di bagian tengah roda dengan warna abu-abu pertengahan (saturasi 0%) dan kemudian menjadi semakin jelas saat berlanjut ke tepi luar (saturasi 100%).

Yang tersisa adalah kita memutuskan seberapa cerah warna yang kita inginkan, dengan memilih nilai luminance antara 0% dan 100%. Nilai 50% untuk luminance memberikan warna paling terang dan penuh; menurunkan nilainya (hingga minimum 0%) menggelapkan warna hingga ditampilkan sebagai hitam; dan menaikkan nilainya (maksimal 100%) mencerahkan warna hingga terlihat putih. kita dapat memvisualisasikan ini seolah-olah kita sedang mencampur tingkat hitam atau putih ke dalam warna. Oleh karena itu, misalnya, untuk memilih warna kuning jenuh penuh dengan standar persen kecerahan, kita akan menggunakan deklarasi seperti ini:

```
color:hsl(60, 100%, 50%);
```

Atau, untuk warna biru yang lebih gelap, kita dapat menggunakan deklarasi seperti:

```
color:hsl(240, 100%, 40%);
```

Anda juga dapat menggunakan ini (dan semua fungsi warna CSS lainnya) dengan properti apa pun yang mengharapkan warna, seperti warna latar, dan sebagainya.

Warna HSLA

Untuk memberikan kontrol lebih jauh tentang bagaimana warna akan muncul, kita dapat menggunakan fungsi hsla, menyediakannya dengan tingkat keempat (atau alfa) untuk warna, yang merupakan nilai floating-point antara 0 dan 1. Nilai 0 menentukan bahwa warnanya benar-benar transparan, sedangkan 1 berarti sepenuhnya buram. Inilah cara kita memilih warna kuning jenuh penuh dengan standar kecerahan dan 30% opacity:

```
color:hsla(60, 100%, 50%, 0.3);
```

Atau, untuk warna biru yang sepenuhnya jenuh tetapi lebih terang dengan opacity 82%, kita dapat menggunakan deklarasi ini:

```
color:hsla(240, 100%, 60%, 0.82);
```

Warna RGB

Anda mungkin akan lebih terbiasa menggunakan sistem RGB dalam memilih warna, karena mirip dengan menggunakan format warna #nnnnnn dan #nnn. Misalnya, untuk menerapkan warna kuning ke properti, kita dapat menggunakan salah satu dari deklarasi berikut (yang pertama mendukung 16 juta warna, dan yang kedua empat ribu):

```
color:#ffff00;
color:#ff0;
```

Anda juga dapat menggunakan fungsi CSS rgb untuk mencapai hasil yang sama, tetapi kita menggunakan angka desimal alih-alih heksadesimal (di mana 255 desimal adalah heksadesimal ff):

```
color:rgb(255, 255, 0);
```

Namun lebih baik dari itu, kita bahkan tidak perlu memikirkan jumlah hingga 256 lagi, karena kita dapat menentukan nilai persentase, seperti ini:

```
color:rgb(100%, 100%, 0);
```

Bahkan, kita sekarang bisa sangat dekat dengan warna yang diinginkan hanya dengan memikirkan warna primernya. Misalnya, hijau dan biru menghasilkan cyan, jadi untuk membuat warna yang mendekati cyan, tetapi dengan lebih banyak biru daripada hijau, kita bisa menebak dengan baik pada 0% merah, 40% hijau, dan 60% biru, dan coba deklarasi seperti ini:

```
color:rgb(0%, 40%, 60%);
```

Warna RGBA

Seperti fungsi hsla, fungsi rgba mendukung argumen alfa keempat, jadi kita dapat, misalnya, menerapkan warna seperti cyan sebelumnya dengan opacity 40% dengan menggunakan deklarasi seperti ini:

```
color:rgba(0%, 40%, 60%, 0.4);
```

Properti opacity

Properti opacity menyediakan kontrol alfa yang sama dengan fungsi hsla dan rgba, tetapi memungkinkan kita memodifikasi opacity objek (atau transparansi jika kita mau) secara terpisah dari warnanya. Untuk menggunakannya, terapkan deklarasi seperti berikut ke elemen (yang dalam contoh ini menetapkan opacity menjadi 25%, atau 75% transparan):

```
opacity:0.25;
```

Catatan: Browser berbasis WebKit dan Mozilla memerlukan awalan khusus browser untuk properti ini. Dan untuk kompatibilitas mundur dengan rilis Internet Explorer sebelum versi 9, kita harus menambahkan deklarasi berikut (di mana nilai opacity dikalikan dengan 100):

```
filter:alpha(opacity='25');
```

4.8 EFEK TEKS

Sejumlah efek baru sekarang dapat diterapkan pada teks dengan bantuan CSS3, termasuk bayangan teks, teks yang tumpang tindih, dan pembungkusan kata.

Properti teks-bayangan

Properti text-shadow mirip dengan properti box-shadow dan menggunakan kumpulan argumen yang sama: offset horizontal dan vertikal, jumlah pengaburan, dan warna yang akan digunakan. Misalnya, deklarasi berikut mengimbangi bayangan dengan 3 piksel baik secara horizontal maupun vertikal, dan menampilkan bayangan dalam abu-abu gelap, dengan pengaburan 4 piksel:

```
text-shadow:3px  3px 4px #444;
```

Hasil dari deklarasi ini terlihat seperti Gambar 4.6, dan berfungsi di semua yang terbaru versi semua browser utama (tetapi bukan IE9 atau lebih rendah).

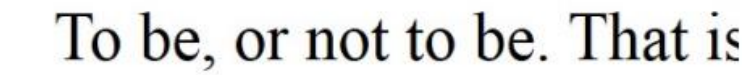
Gambar 4.6 Menerapkan bayangan ke teks

Properti text-overflow

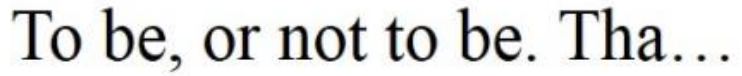
Saat menggunakan salah satu properti CSS overflow dengan nilai tersembunyi, kita juga dapat menggunakan properti text-overflow untuk menempatkan elipsis (tiga titik) tepat sebelum cutoff untuk menunjukkan bahwa beberapa teks telah terpotong, seperti ini:

```
text-overflow:ellipsis;
```

Tanpa properti ini, ketika teks “To be, or not to be. Itulah pertanyaannya.” terpotong, hasilnya akan terlihat seperti Gambar 4.7; dengan deklarasi yang diterapkan, hasilnya seperti Gambar 20-8



Gambar 4.7 Teks secara otomatis terpotong



Gambar 4.8 Alih-alih terpotong, teks menghilang menggunakan elipsis

Agar ini berhasil, tiga hal diperlukan:

- Elemen harus memiliki properti overflow yang tidak terlihat, seperti melimpah: tersembunyi.
- Elemen harus memiliki properti white-space:nowrap yang disetel ke constrain teks.
- Lebar elemen harus lebih kecil dari teks yang akan dipotong.

Properti pembungkus kata

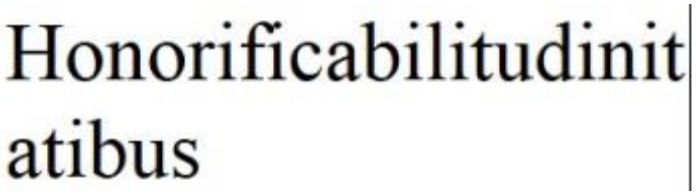
Ketika kita memiliki kata yang sangat panjang yang lebih lebar dari elemen yang memuatnya, kata itu akan meluap atau terpotong. Tetapi sebagai alternatif untuk menggunakan properti text-overflow dan memotong teks, kita dapat menggunakan properti word-wrap dengan nilai break-word untuk membungkus garis panjang, seperti ini:

word-wrap:break-word;

Misalnya, pada Gambar 4.9 kata Honorificabilitudinitatibus terlalu lebar untuk kotak yang berisi (yang tepi kanannya ditampilkan sebagai garis vertikal padat antara huruf t dan a) dan, karena tidak ada sifat luapan yang diterapkan, ia telah melampaui batasnya.

Gambar 4.9 Kata itu terlalu lebar untuk wadahnya dan telah meluap

Tetapi pada Gambar 4.10 properti word-wrap dari elemen telah diberi nilai break-word, sehingga kata tersebut terbungkus rapi ke baris berikutnya.



Gambar 4.10. Kata sekarang membungkus di tepi kanan

4.9 FONT WEB

Penggunaan font web CSS3 sangat meningkatkan tipografi yang tersedia untuk desainer web dengan memungkinkan font dimuat dan ditampilkan dari seluruh Web, bukan

hanya dari komputer pengguna. Untuk mencapai ini, deklarasikan font web menggunakan @font-face, seperti ini:

```
@font-face
{
font-family:FontName;
src:url('FontName.otf');
}
```

Fungsi url memerlukan nilai yang berisi jalur atau URL font. Di sebagian besar browser, kita dapat menggunakan font TrueType (.ttf) atau OpenType (.otf), tetapi Internet Explorer membatasi kita untuk font TrueType yang telah dikonversi ke EOT (.eot). Untuk memberi tahu browser jenis font, kita dapat menggunakan fungsi format, seperti ini (untuk font OpenType):

```
@font-face
{
font-family:FontName;
src:url('FontName.otf') format('opentype');
}
```

Atau ini untuk font TrueType:

```
@font-face
{
font-family:FontName;
src:url('FontName.ttf') format('truetype');
}
```

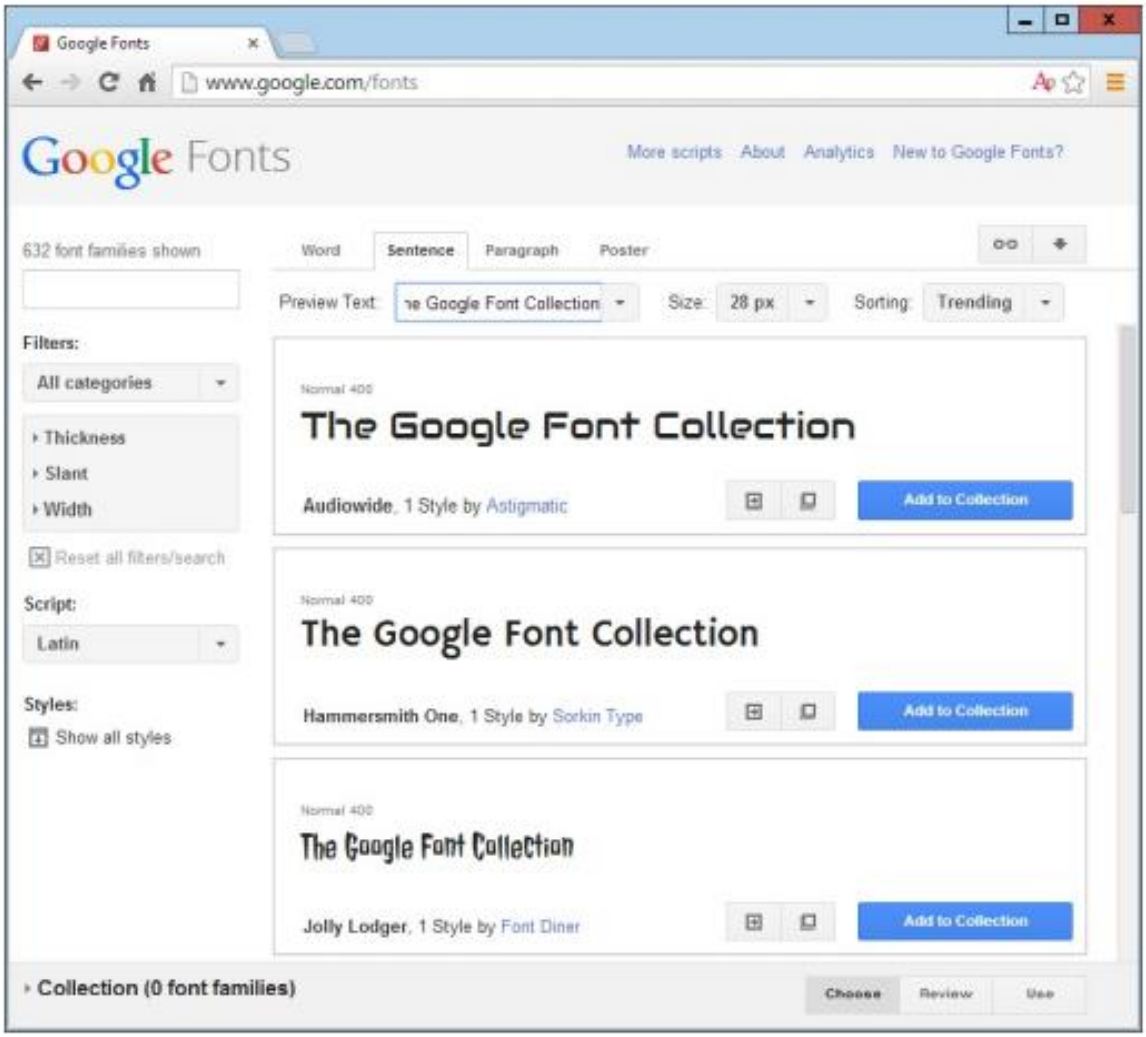
Namun, karena Microsoft Internet Explorer hanya menerima font EOT, itu mengabaikan deklarasi @font-face yang berisi fungsi format.

Font Web Google

Salah satu cara terbaik untuk menggunakan font web adalah memuatnya secara gratis dari server Google. Untuk mengetahui lebih lanjut tentang ini, periksa situs web Google Fonts (lihat Gambar 4.11), di mana kita bisa mendapatkan akses ke lebih dari 630 font-family, dan terus bertambah! Untuk menunjukkan betapa mudahnya menggunakan salah satu font ini, berikut adalah cara kita memuat font Google (dalam hal ini, Lobster) ke dalam HTML kita untuk digunakan dalam heading <h1>:

```
<!DOCTYPE html>
<html>
<head>
<style>
h1 { font-family:'Lobster', arial, serif; }
</style>
<link href='http://fonts.googleapis.com/css?family=Lobster'
rel='stylesheet' type='text/css'>
</head>
<body>
```

```
<h1>Hello</h1>
</body>
</html>
```



Gambar 4.11 Sangat mudah untuk menyertakan font web Google

4.10 TRANSFORMASI

Dengan menggunakan transformasi, kita dapat memiringkan, memutar, meregangkan, dan menekan elemen di salah satu dari hingga tiga dimensi (ya, 3D didukung, tetapi hanya di browser berbasis WebKit untuk saat ini). Hal ini memudahkan untuk membuat efek hebat dengan keluar dari layout persegi panjang seragam dan elemen lainnya, karena sekarang mereka dapat ditampilkan pada berbagai sudut dan dalam berbagai bentuk. Untuk melakukan transformasi, gunakan properti transform (sayangnya memiliki awalan khusus browser untuk browser Mozilla, WebKit, Opera, dan Microsoft, jadi sekali lagi kita harus merujuk ke http://caniuse.com). kita dapat menerapkan berbagai properti ke properti transformasi, dimulai dengan nilai none, yang mereset objek ke status nontransformasi:

```
transform:none;
```

Anda dapat menyediakan satu atau beberapa fungsi berikut ke properti transformasi:

matrix

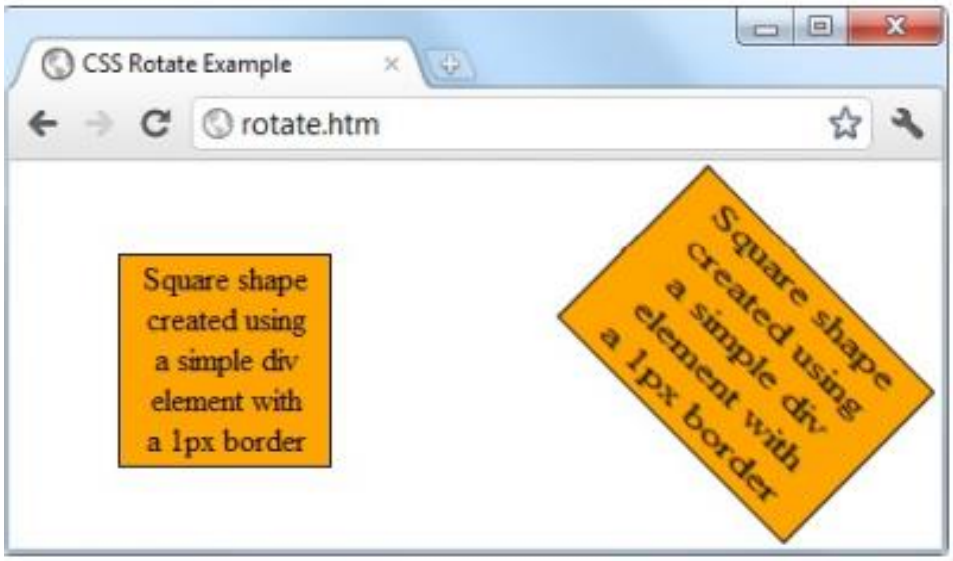
Mengubah objek dengan menerapkan matriks nilai padanya

translate
Memindahkan asal elemen
scale
Menskalakan suatu objek
rotate
Memutar objek
skew
Mencondongkan objek
Ada juga versi tunggal dari banyak fungsi ini, seperti translateX, scaleY, dan sebagainya. Jadi, misalnya, untuk memutar elemen searah jarum jam sebesar 45 derajat, kita dapat menerapkan deklarasi ini padanya:

```
transform:rotate(45deg);
```

Pada saat yang sama, kita dapat memperbesar objek ini, seperti dalam deklarasi berikut, yang memperbesar lebarnya 1,5 kali dan tingginya 2 kali, dan kemudian melakukan rotasi (Gambar 4.12 menunjukkan objek sebelum transformasi diterapkan, dan kemudian sesudahnya):

```
transform:scale(1.5, 2) rotate(45deg);
```

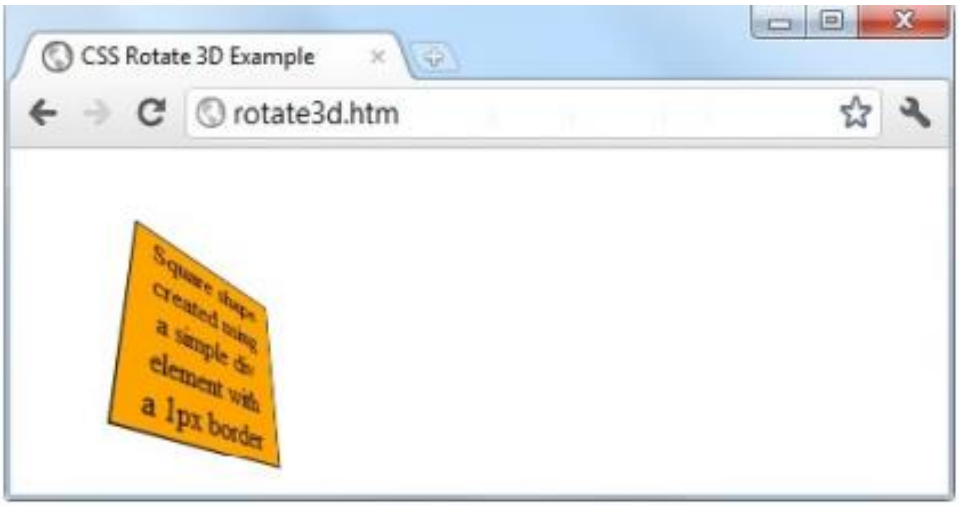


Gambar 4.12 Sebuah objek sebelum dan sesudah transformasi

Transformasi 3D
Anda juga dapat mengubah objek dalam tiga dimensi menggunakan fitur transformasi 3D CSS3 berikut:
perspective
Melepaskan elemen dari ruang 2D dan menciptakan dimensi ketiga di dalamnya yang bisa bergerak
Transform-origin
Menetapkan lokasi di mana semua garis bertemu ke satu titik
Translate3d
Memindahkan elemen ke lokasi lain dalam ruang 3D

Scale3d
Mengubah skala satu atau lebih dimensi
Rotate3d
Memutar elemen di sekitar salah satu sumbu X, Y, dan Z.
Gambar 4.13 menunjukkan objek 2D yang telah diputar dalam ruang 3D dengan CSS aturan seperti berikut ini:

transform:perspective(200px) rotateX(10deg) rotateY(20deg) rotateZ(30deg);



Gambar 4.13 Sosok yang diputar dalam ruang 3D

Untuk informasi lebih lanjut, lihat tutorial di http://tinyurl.com/3dcsstransforms.

4.11 TRANSISI

Juga muncul di semua versi terbaru dari browser utama (termasuk Internet Explorer 10, tetapi bukan versi yang lebih rendah) adalah fitur baru yang dinamis yang disebut transisi. Ini menentukan efek animasi yang kita inginkan terjadi saat elemen diubah, dan browser akan secara otomatis menangani semua bingkai di antara Anda. Ada empat properti yang harus kita sediakan untuk menyiapkan transisi, sebagai berikut:

transition-property :property;
transition-duration :time;
transition-delay :time;
transition-timing-function:type;

Properti untuk Transisi

Transisi memiliki properti seperti tinggi dan warna batas. Tentukan properti yang ingin kita ubah di properti CSS bernama properti transisi (di sini kata properti digunakan oleh alat yang berbeda untuk mengartikan hal yang berbeda). kita dapat menyertakan beberapa properti dengan memisahkannya dengan koma, seperti ini:

transition-property:width, height, opacity;

Atau, jika kita benar-benar menginginkan segala sesuatu tentang suatu elemen untuk ditransisikan (termasuk warna), gunakan nilai all, seperti ini:

```
transition-property:all;
```

Durasi Transisi

Properti durasi transisi memerlukan nilai 0 detik atau lebih, seperti berikut ini, yang menetapkan bahwa transisi harus membutuhkan waktu 1,25 detik untuk diselesaikan:

```
transition-duration:1.25s;
```

Delay Transisi

Jika properti transisi-delay diberi nilai lebih besar dari 0 detik (default), ini memperkenalkan penundaan antara tampilan awal elemen dan awal transisi. Berikut ini memulai transisi setelah penundaan 0,1 detik:

```
transition-delay:0.1s;
```

Jika properti transisi-delay diberi nilai kurang dari 0 detik (dengan kata lain, nilai negatif), transisi akan dijalankan saat properti diubah, tetapi akan tampak telah memulai eksekusi pada offset yang ditentukan, di tengah jalan siklusnya.

Waktu Transisi

Properti fungsi waktu transisi memerlukan salah satu dari nilai berikut:

ease

Mulai perlahan, dapatkan lebih cepat, lalu akhiri perlahan.

linear

Transisi dengan kecepatan konstan.

ease-in

Mulailah dengan perlahan, lalu lanjutkan dengan cepat hingga selesai.

easy-out

Mulai dengan cepat, tetap cepat sampai mendekati akhir, lalu akhiri perlahan.

ease-in-out

Mulai perlahan, cepat, lalu akhiri perlahan.

Menggunakan salah satu nilai yang mengandung kata kemudahan memastikan bahwa transisi terlihat ekstra cair dan alami, tidak seperti transisi linier yang entah bagaimana tampak lebih mekanis. Dan jika ini tidak cukup bervariasi untuk Anda, kita juga dapat membuat transisi kita sendiri menggunakan fungsi kubik bezier. Misalnya, berikut adalah deklarasi yang digunakan untuk membuat lima jenis transisi sebelumnya, yang menggambarkan bagaimana kita dapat dengan mudah membuat sendiri:

```
transition-timing-function:cubic-bezier(0.25, 0.1, 0.25, 1);
transition-timing-function:cubic-bezier(0, 0, 1, 1);
transition-timing-function:cubic-bezier(0.42, 0, 1, 1);
transition-timing-function:cubic-bezier(0, 0, 0.58, 1);
transition-timing-function:cubic-bezier(0.42, 0, 0.58, 1);
```

Sintaks Singkatan

Anda mungkin merasa lebih mudah menggunakan versi singkatan dari properti ini dan menyertakan semua nilai dalam satu deklarasi seperti berikut, yang akan mentransisikan semua properti secara linier, selama .3 detik, setelah awal (opsional) penundaan 0,2 detik:

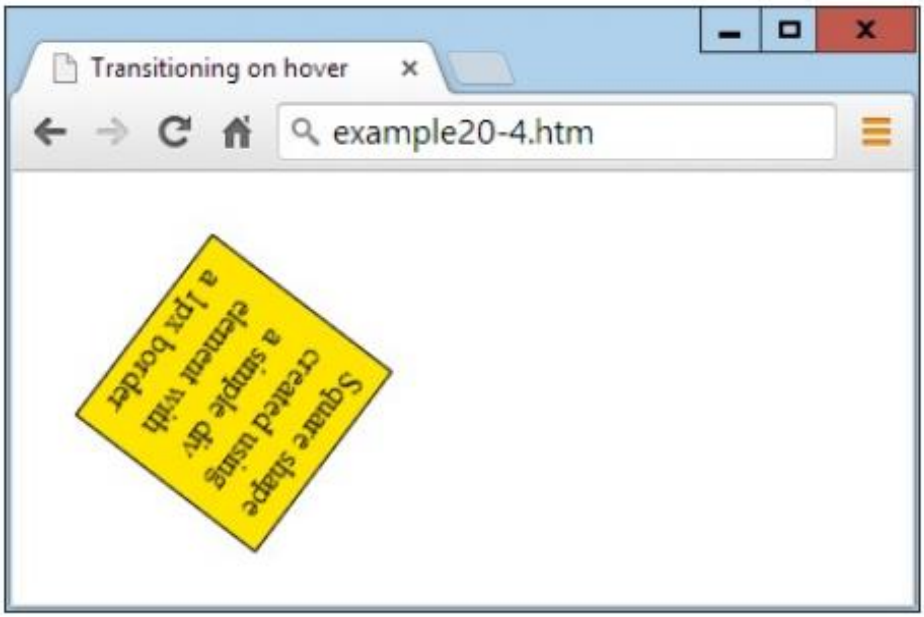
```
transition:all .3s linear .2s;
```

Melakukannya akan menyelamatkan kita dari kesulitan memasukkan banyak deklarasi yang sangat mirip, terutama jika kita mendukung semua awalan browser utama. Contoh koding selanjutnya mengilustrasikan bagaimana kita dapat menggunakan transisi dan transformasi bersama-sama. CSS membuat elemen persegi, oranye dengan beberapa teks di dalamnya, dan pseudeclass hover yang menentukan bahwa ketika mouse melewati objek itu harus berputar 180 derajat dan berubah dari oranye menjadi kuning (lihat Gambar 4.14).

Contoh Transisi pada efek hover

```
<!DOCTYPE html>
<html>
<head>
<title>Transitioning on hover</title>
<style>
#square {
position :absolute;
top :50px;
left :50px;
width :100px;
height :100px;
padding :2px;
text-align :center;
border-width :1px;
border-style :solid;
background :orange;
transition :all .8s ease-in-out;
-moz-transition :all .8s ease-in-out;
-webkit-transition:all .8s ease-in-out;
-o-transition :all .8s ease-in-out;
-ms-transition :all .8s ease-in-out;
}
#square:hover {
background :yellow;
-moz-transform :rotate(180deg);
-webkit-transform :rotate(180deg);
-o-transform :rotate(180deg);
-ms-transform :rotate(180deg);
transform :rotate(180deg);
}
</style>
```

```
</head>
<body>
<div id='square'>
Square shape<br>
created using<br>
a simple div<br>
element with<br>
a 1px border
</div>
</body>
</html>
```



Gambar 4.14 Objek berputar dan berubah warna saat dilayangkan

Kode contoh melayani semua browser yang berbeda dengan menyediakan versi khusus browser dari deklarasi. Pada semua browser terbaru (termasuk IE10 atau lebih tinggi), objek akan berputar searah jarum jam saat dilayangkan, sementara perlahan berubah dari oranye menjadi kuning. Transisi CSS cerdas karena ketika dibatalkan, mereka dengan lancar kembali ke nilai aslinya. Jadi, jika kita memindahkan mouse sebelum transisi selesai, mouse akan langsung membalik dan memulai transisi kembali ke keadaan awal.

BAB 5
HTML5 DAN FITURNYA

5.1 PENGENALAN HTML5

HTML5 mewakili lompatan besar ke depan dalam desain web, layout, dan kegunaan. Ini menyediakan cara sederhana untuk memanipulasi grafik di browser web tanpa menggunakan plug-in seperti Flash, menawarkan metode untuk memasukkan audio dan video ke halaman web (sekali lagi tanpa plug-in), dan mengatasi beberapa inkonsistensi yang mengganggu yang merayap ke HTML selama evolusinya. Selain itu, HTML5 menyertakan banyak peningkatan lain seperti penanganan geolokasi, pekerja web untuk mengelola tugas latar belakang, penanganan formulir yang ditingkatkan, akses ke bundel penyimpanan lokal (jauh melebihi kemampuan cookie yang terbatas), dan bahkan fasilitas untuk membalik halaman web ke dalam aplikasi web untuk browser seluler.

Apa yang membuat penasaran tentang HTML5, bagaimanapun, adalah bahwa ini telah menjadi evolusi yang berkelanjutan, di mana browser yang berbeda telah mengadopsi fitur yang berbeda pada waktu yang berbeda. Untungnya, semua tambahan HTML5 terbesar dan terpopuler akhirnya kini didukung oleh semua browser utama (yang menguasai lebih dari 1% atau lebih pasar, seperti Chrome, Internet Explorer, Firefox, Safari, dan Opera, serta Android dan iOS. browser).

Tetapi dengan HTML5 yang baru secara resmi diserahkan ke W3C pada awal 2013, masih ada sejumlah fitur yang luar biasa di beberapa browser. Namun demikian, kita sekarang sepenuhnya memasuki gelombang besar kedua menuju interaktivitas web dinamis (yang pertama adalah adopsi dari apa yang kemudian dikenal sebagai Web 2.0). Saya ragu untuk menyebutnya Web 3.0, karena istilah HTML5 mengatakan itu semua untuk kebanyakan orang, dan dalam pandangan saya itu dapat dianggap sebagai versi Web 2.0 yang lebih baru (mungkin seperti Web 2.7).

Sebenarnya, saya pikir akan sangat menarik untuk melihat seperti apa Web 3.0 nantinya. Namun, jika saya berani memprediksi, saya akan mengatakan itu akan dihasilkan dari penerapan kecerdasan buatan (AI) dalam bentuk versi perangkat lunak yang jauh lebih mampu seperti Apple Siri, Microsoft Cortana, dan IBM Watson, dikombinasikan dengan perangkat yang dapat dikenakan. teknologi yang menggunakan input visual dan suara — seperti Google Glass dan jam tangan Galaxy Gear — bukan keyboard. Untuk saat ini, setelah menulis tentang apa yang akan datang di HTML5 selama beberapa tahun, dan sekarang begitu banyak bagian dari spesifikasi dapat digunakan di hampir semua perangkat dan browser. Jadi izinkan saya memberi kita gambaran umum tentang apa yang tersedia untuk kita di HTML5 saat ini.

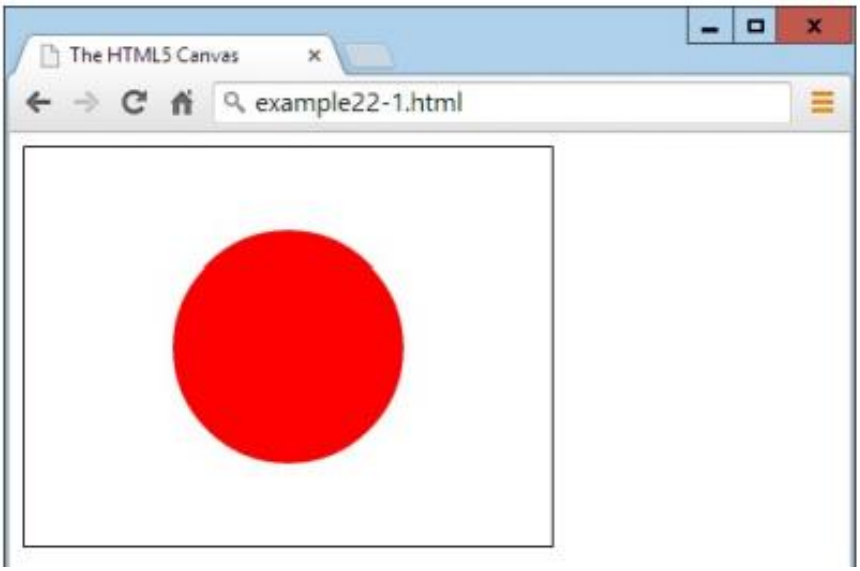
Kanvas

Awalnya diperkenalkan oleh Apple untuk mesin rendering WebKit (yang berasal dari mesin layout HTML KDE) untuk browser Safari-nya (dan sekarang juga diimplementasikan di iOS, Android, Kindle, Chrome, BlackBerry, Opera, dan Tizen), elemen kanvas memungkinkan kita untuk menggambar grafik di halaman web tanpa harus bergantung pada plug-in seperti Java atau Flash. Setelah distandarisasi, kanvas diadopsi oleh semua browser lain dan sekarang menjadi andalan pengembangan web modern. Seperti elemen HTML lainnya, kanvas hanyalah elemen dalam halaman web dengan dimensi yang ditentukan, dan di dalamnya kita dapat

menggunakan JavaScript untuk menggambar grafik. kita membuat kanvas menggunakan tag <canvas>, yang juga harus kita tetapkan ID sehingga JavaScript akan mengetahui kanvas mana yang diaksesnya (karena kita dapat memiliki lebih dari satu kanvas pada satu halaman). Dalam Contoh 1, saya telah membuat elemen <canvas>, dengan ID mycanvas, yang berisi beberapa teks yang hanya ditampilkan di browser yang tidak mendukung kanvas. Di bawah ini ada bagian JavaScript, yang menggambar bendera Jepang di kanvas (seperti yang ditunjukkan pada Gambar 5.1).

Contoh 1. Menggunakan elemen kanvas HTML5

```
<!DOCTYPE html>
<html>
<head>
<title>The HTML5 Canvas</title>
<script src='OSC.js'></script>
</head>
<body>
<canvas id='mycanvas' width='320' height='240'>
This is a canvas element given the ID <i>mycanvas</i>
This text is only visible in non-HTML5 browsers
</canvas>
<script>
canvas = O('mycanvas')
context = canvas.getContext('2d')
context.fillStyle = 'red'
S(canvas).border = '1px solid black'
context.beginPath()
context.moveTo(160, 120)
context.arc(160, 120, 70, 0, Math.PI * 2, false)
context.closePath()
context.fill()
</script>
</body>
</html>
```



Gambar 5.1 Menggambar bendera Jepang menggunakan kanvas HTML5

Pada titik ini, tidak perlu merinci apa yang sedang terjadi, tetapi kita seharusnya sudah melihat bagaimana menggunakan kanvas tidak sulit, tetapi perlu mempelajari beberapa fungsi JavaScript baru. Perhatikan bahwa contoh ini mengacu pada rangkaian fungsi OSC.js dari bab sebelumnya untuk membantu menjaga kode tetap rapi dan kompak.

Geolokasi

Menggunakan geolokasi, browser dapat mengembalikan informasi ke server web tentang lokasi Anda. Informasi ini dapat berasal dari chip GPS di komputer atau perangkat seluler yang kita gunakan, dari alamat IP Anda, atau dari analisis hotspot WiFi terdekat. Untuk tujuan keamanan, pengguna selalu memegang kendali dan dapat menolak untuk memberikan informasi ini sekali saja, atau dapat mengaktifkan pengaturan untuk memblokir secara permanen atau mengizinkan akses ke data ini dari satu atau semua situs web. Ada banyak kegunaan untuk teknologi ini, termasuk memberi kita navigasi belokan demi belokan; menyediakan peta lokal; memberi tahu kita tentang restoran terdekat, hotspot WiFi, atau tempat lain; memberi tahu kita teman mana yang ada di dekat Anda; mengarahkan kita ke pompa bensin terdekat; dan banyak lagi. Contoh 2 akan menampilkan peta Google dari lokasi pengguna, selama browser mendukung geolokasi dan pengguna memberikan akses ke lokasinya (seperti yang ditunjukkan pada Gambar 5.2). Jika tidak, itu akan menampilkan kesalahan.

Contoh 2. Menampilkan peta di lokasi pengguna

```
<!DOCTYPE html>
<html>
<head>
<title>Geolocation Example</title>
<script src='OSC.js'></script>
<script src="https://maps.googleapis.com/maps/api/js?sensor=false"></script>
</head>
<body>
<div id='status'></div>
<div id='map'></div>
```

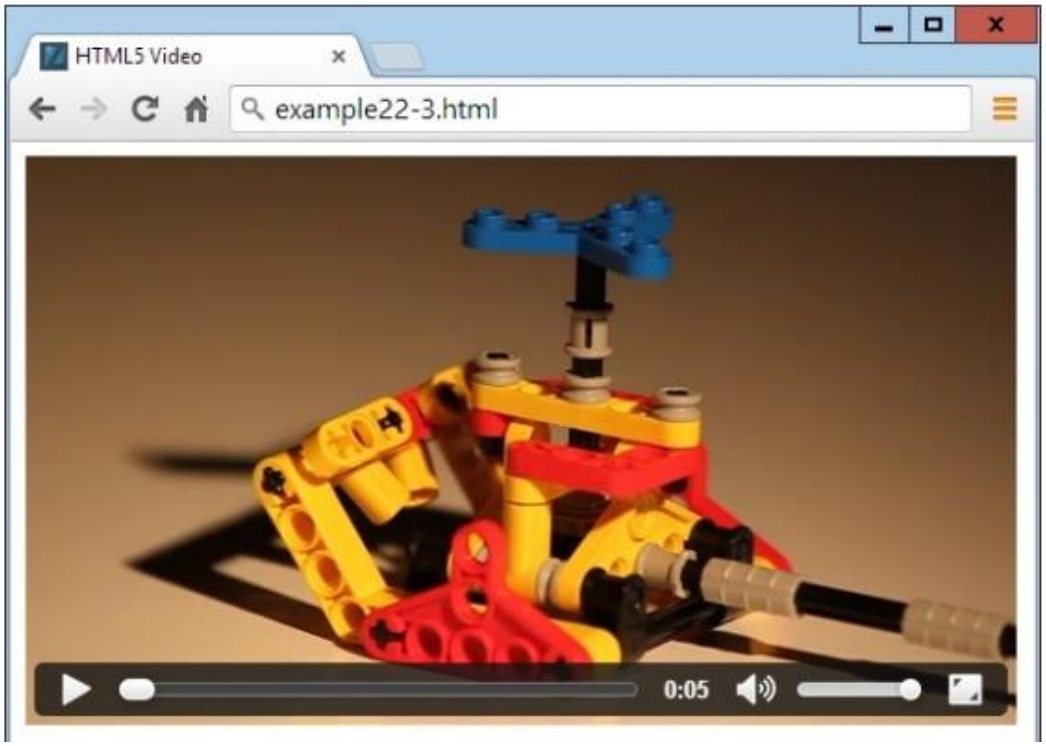
```
<script>
if (typeof navigator.geolocation == 'undefined')
alert("Geolocation not supported.")
else
navigator.geolocation.getCurrentPosition(granted, denied)
function granted(position)
{
O('status').innerHTML = 'Permission Granted'
S('map').border = '1px solid black'
S('map').width = '640px'
S('map').height = '320px'
var lat = position.coords.latitude
var long = position.coords.longitude
var gmap = O('map')
var gopts =
{
center: new google.maps.LatLng(lat, long),
zoom: 9, mapTypeId: google.maps.MapTypeId.ROADMAP
}
var map = new google.maps.Map(gmap, gopts)
}
function denied(error)
{
var message
switch(error.code)
{
case 1: message = 'Permission Denied'; break;
case 2: message = 'Position Unavailable'; break;
case 3: message = 'Operation Timed Out'; break;
case 4: message = 'Unknown Error'; break;
}
O('status').innerHTML = message
}
</script>
</body>
</html>
```

Audio dan Video

Tambahan hebat lainnya untuk HTML5 adalah dukungan untuk audio dan video dalam browser. Meskipun memutar jenis media ini bisa menjadi sedikit rumit karena variasi jenis dan lisensi penyandian, elemen <audio> dan <video> memberikan fleksibilitas yang kita perlukan untuk menampilkan jenis media yang kita miliki. Dalam Contoh 3, file video yang sama telah dikodekan dalam format yang berbeda untuk memastikan bahwa semua browser utama diperhitungkan. Browser hanya akan memilih jenis pertama yang mereka kenali dan memainkannya.

Contoh 3. Memutar video dengan HTML5

```
<!DOCTYPE html>
<html>
<head>
<title>HTML5 Video</title>
</head>
<body>
<video width='560' height='320' controls>
<source src='movie.mp4' type='video/mp4'>
<source src='movie.webm' type='video/webm'>
<source src='movie.ogv' type='video/ogg'>
</video>
</body>
</html>
```



Gambar 5.2 Menampilkan video menggunakan HTML5

Formulir

Formulir HTML5 sedang dalam proses penyempurnaan, tetapi dukungan di semua browser tetap tidak merata. Sementara itu, kita dapat terus mengikuti perkembangan terbaru tentang formulir HTML5 di http://tinyurl.com/h5forms.

Penyimpanan lokal

Dengan penyimpanan lokal, kemampuan kita untuk menyimpan data pada perangkat lokal meningkat secara substansial dari sedikit ruang yang disediakan oleh cookie. Ini membuka kemungkinan kita menggunakan aplikasi web untuk mengerjakan dokumen secara offline dan kemudian hanya menyinkronkannya dengan server web saat koneksi internet tersedia. Ini juga meningkatkan prospek penyimpanan database kecil secara lokal untuk akses dengan WebSQL, mungkin untuk menyimpan salinan detail koleksi musik Anda, atau semua statistik pribadi kita sebagai bagian dari diet atau rencana penurunan berat badan, misalnya.

Pekerja Web

Telah dimungkinkan untuk menjalankan aplikasi yang digerakkan oleh interupsi di latar belakang menggunakan JavaScript selama bertahun-tahun, tetapi ini adalah proses yang kikuk dan tidak efisien. Jauh lebih masuk akal untuk membiarkan teknologi browser yang mendasari menjalankan tugas latar belakang atas nama Anda, yang dapat dilakukan jauh lebih cepat daripada yang kita bisa dengan terus mengganggu browser untuk memeriksa bagaimana keadaannya. Sebagai gantinya, dengan pekerja web kita mengatur semuanya dan meneruskan kode kita ke browser web, yang kemudian menjalankannya. Ketika sesuatu yang signifikan terjadi, kode kita hanya perlu memberi tahu browser, yang kemudian melaporkan kembali ke kode utama Anda. Sementara itu, halaman web kita tidak dapat melakukan apa pun atau sejumlah tugas lain, dan dapat melupakan tugas latar belakang hingga tugas tersebut diketahui sendiri.

Aplikasi Web

Semakin banyak hari ini, halaman web mulai menyerupai aplikasi, dan dengan HTML5 mereka dapat menjadi aplikasi web dengan sangat mudah. Yang harus kita lakukan adalah memberi tahu browser web tentang sumber daya yang digunakan dalam aplikasi Anda, dan itu akan mengunduhnya ke tempat yang dapat dijalankan dan diakses secara lokal, offline, dan tanpa koneksi internet jika perlu. Bab 25 menunjukkan bagaimana kita dapat melakukan ini untuk mengubah contoh jam di bagian pekerja web menjadi aplikasi web.

Mikrodata

Saya menunjukkan bagaimana kita dapat menandai kode kita dengan microdata agar benar-benar dapat dimengerti oleh browser atau teknologi lain yang perlu mengaksesnya. Microdata pasti menjadi semakin penting untuk pengoptimalan mesin pencari juga, jadi penting bagi kita untuk mulai memasukkannya atau setidaknya memahami informasi apa yang dapat diberikannya tentang situs web Anda.

Ringkasan

Seperti yang kita lihat, ada cukup banyak hal untuk HTML5, dan itu semua adalah kebaikan yang telah lama ditunggu banyak orang—tetapi akhirnya mereka ada di sini. Dimulai dengan kanvas, beberapa bab berikut akan menjelaskan fitur-fitur ini kepada kita dengan sangat rinci, sehingga kita dapat menggunakannya, dan menyempurnakan situs web Anda, dalam waktu singkat.

Fitur HML5 lainnya

Sebenarnya, sebagian besar fitur ini (seperti kebanyakan HTML5) sebenarnya bukan ekstensi ke HTML, karena kita mengaksesnya dengan JavaScript daripada dengan markup HTML. Mereka hanyalah teknologi yang dianut oleh pengembang browser, dan telah diberi nama payung HTML5 yang praktis. Namun, ini berarti kita harus memahami sepenuhnya tutorial JavaScript dalam buku ini agar dapat menggunakannya dengan benar. Yang mengatakan, begitu kita memahaminya, kita akan bertanya-tanya bagaimana kita bisa melakukannya tanpa fitur-fitur baru yang kuat ini.

5.2 GEOLOKASI DAN LAYANAN GPS

Layanan GPS (Global Positioning System) terdiri dari beberapa satelit yang mengorbit bumi yang posisinya diketahui dengan sangat tepat. Saat perangkat berkemampuan GPS menyetelnya, waktu yang berbeda saat sinyal dari berbagai satelit ini tiba memungkinkan perangkat untuk mengetahui lokasinya dengan cukup akurat; karena kecepatan cahaya (dan karena itu gelombang radio) adalah konstanta yang diketahui, waktu yang dibutuhkan sinyal

untuk sampai dari satelit ke perangkat GPS menunjukkan jarak satelit. Dengan mencatat waktu yang berbeda di mana sinyal datang dari satelit yang berbeda, yang berada di lokasi orbit yang diketahui secara tepat pada satu waktu, perhitungan triangulasi sederhana memberikan perangkat posisinya relatif terhadap satelit dalam beberapa meter atau kurang. Banyak perangkat seluler, seperti ponsel dan tablet, memiliki chip GPS dan dapat memberikan informasi ini. Tetapi beberapa tidak, yang lain menyetelnya, dan yang lain mungkin digunakan di dalam ruangan di mana mereka terlindung dari satelit GPS dan karenanya tidak dapat menerima sinyal apa pun. Dalam kasus ini, teknik tambahan dapat digunakan untuk mencoba menentukan lokasi Anda.

Metode Lokasi Lainnya

Jika perangkat kita memiliki perangkat keras ponsel tetapi bukan GPS, perangkat mungkin mencoba melakukan triangulasi lokasi dengan memeriksa waktu sinyal yang diterima dari berbagai menara komunikasi yang dapat digunakan untuk berkomunikasi (dan posisinya diketahui dengan sangat tepat). Jika ada beberapa menara, ini bisa mendekati lokasi kita seperti GPS. Tetapi jika hanya ada satu menara, kekuatan sinyal dapat digunakan untuk menentukan radius kasar di sekitar menara, dan lingkaran yang dibuatnya mewakili area di mana kita mungkin berada. Ini bisa menempatkan kita di mana saja dalam jarak satu atau dua mil dari lokasi kita yang sebenarnya, hingga beberapa puluh meter.

Jika tidak, mungkin ada titik akses WiFi yang diketahui posisinya dalam jangkauan perangkat Anda, dan karena semua titik akses memiliki alamat pengidentifikasi unik yang disebut alamat MAC (Kontrol Akses Media), perkiraan lokasi yang cukup baik dapat diperoleh, mungkin dalam satu atau dua jalan. Ini adalah jenis informasi yang dikumpulkan oleh kendaraan Google Street View. Dan jika gagal, alamat IP (Protokol Internet) yang digunakan oleh perangkat kita dapat ditanyakan dan digunakan sebagai indikator kasar lokasi Anda. Namun, seringkali, ini hanya menyediakan lokasi sakelar utama milik penyedia Internet Anda, yang jaraknya bisa puluhan atau bahkan ratusan mil. Tetapi paling tidak, alamat IP kita dapat (biasanya) mempersempit negara dan terkadang wilayah tempat kita berada.

Catatan: Alamat IP biasanya digunakan oleh perusahaan media untuk membatasi pemutaran konten mereka berdasarkan wilayah. Namun, ini masalah sederhana untuk menyiapkan server proxy yang menggunakan alamat IP penerusan (di wilayah yang memblokir akses luar) untuk mengambil dan meneruskan konten melalui blokade langsung ke browser "asing". Server proxy juga sering digunakan untuk menyamarkan alamat IP asli pengguna atau melewati batasan sensor, dan dapat dibagikan ke banyak pengguna di hotspot WiFi. Oleh karena itu, jika kita menemukan seseorang berdasarkan alamat IP, kita tidak dapat sepenuhnya yakin bahwa kita telah mengidentifikasi lokasi yang tepat, atau bahkan negara, dan harus memperlakukan informasi ini hanya sebagai tebakan terbaik.

Geolokasi dan HTML5

Sekarang saatnya untuk melihatnya secara mendalam, dimulai dengan contoh yang saya berikan sebelumnya, ditunjukkan lagi pada Contoh 3.

Contoh 3. Menampilkan peta lokasi kita saat ini

```
<!DOCTYPE html>
<html>
<head>
<title>Geolocation Example</title>
<script src='OSC.js'></script>
```

```
<script src="https://maps.googleapis.com/maps/api/js?sensor=false"></script>
</head>
<body>
<div id='status'></div>
<div id='map'></div>
<script>
if (typeof navigator.geolocation == 'undefined')
alert("Geolocation not supported.")
else
navigator.geolocation.getCurrentPosition(granted, denied)
function granted(position)
{
O('status').innerHTML = 'Permission Granted'
S('map').border = '1px solid black'
S('map').width = '640px'
S('map').height = '320px'
var lat = position.coords.latitude
var long = position.coords.longitude
var gmap = O('map')
var gopts =
{
center: new google.maps.LatLng(lat, long),
zoom: 9, mapTypeId: google.maps.MapTypeId.ROADMAP
}
var map = new google.maps.Map(gmap, gopts)
}
function denied(error)
{
var message
switch(error.code)
{
case 1: message = 'Permission Denied'; break;
case 2: message = 'Position Unavailable'; break;
case 3: message = 'Operation Timed Out'; break;
case 4: message = 'Unknown Error'; break;
}
O('status').innerHTML = message
}
</script>
</body>
</html>
```

Mari menelusuri kode ini dan melihat cara kerjanya, dimulai dengan bagian <head>, yang menampilkan judul; memuat dalam file OSC.js yang berisi fungsi O, S, dan C yang saya sediakan untuk mempermudah mengakses elemen HTML dari JavaScript; dan kemudian juga menarik

kode JavaScript untuk layanan Google Maps, yang digambar nanti dalam program. Setelah ini, dua elemen div dibuat—satu untuk menampilkan status koneksi, dan yang lainnya untuk peta:

```
<div id='status'></div>
<div id='map'></div>
```

Sisa dokumen adalah JavaScript, yang segera dimulai dengan menginterogasi properti navigator.geolocation. Jika nilai yang dikembalikan tidak ditentukan, maka geolokasi tidak didukung oleh browser dan jendela peringatan kesalahan akan muncul. Jika tidak, metode getCurrentPosition dari properti akan dipanggil, meneruskannya dengan nama dua fungsi: diberikan dan ditolak (ingat bahwa dengan meneruskan nama fungsi, kita meneruskan kode fungsi yang sebenarnya, bukan hasil pemanggilan fungsi, yang akan terjadi jika nama fungsi memiliki tanda kurung):

```
navigator.geolocation.getCurrentPosition(granted, denied)
```

Fungsi-fungsi ini muncul kemudian dalam skrip dan untuk menangani dua kemungkinan izin untuk menyediakan data lokasi pengguna: diberikan atau ditolak. Fungsi yang diberikan didahulukan dan dimasukkan hanya jika data dapat diakses. Dalam fungsi ini, properti innerHTML dari elemen div dengan ID status diatur ke string Izin Diberikan untuk menunjukkan keberhasilan selama penundaan saat peta diambil. Kemudian div peta memiliki beberapa gaya CSS yang diterapkan untuk memberinya batas dan mengatur dimensinya:

```
O('status').innerHTML = 'Permission Granted'
S('map').border = '1px solid black'
S('map').width = '640px'
S('map').height = '320px'
```

Selanjutnya, variabel lat dan long diberi nilai yang dikembalikan oleh rutinitas geolokasi di browser, dan objek gmap dibuat untuk mengakses elemen div peta:

```
var lat = position.coords.latitude
var long = position.coords.longitude
var gmap = O('map')
```

Setelah ini, objek gopts diisi dengan nilai dalam lat dan long, tingkat zoom diatur (dalam hal ini ke 9), dan jenis peta ROADMAP dipilih:

```
var gopts =
{
center: new google.maps.LatLng(lat, long),
zoom: 9, mapTypeId: google.maps.MapTypeId.ROADMAP
}
```

Terakhir, dalam fungsi ini, kita membuat objek peta baru dengan meneruskan gmap dan gopts ke metode Peta dari objek google.maps (kode yang akan kita ingat dimuat tepat setelah file OSC.js).

```
var map = new google.maps.Map(gmap, gopts)
```

Jika izin ditolak atau ada masalah lain, pesan kesalahan adalah satu-satunya yang ditampilkan, sebagai output ke properti innerHTML dari div status oleh fungsi yang ditolak, sesuai dengan masalah yang dihadapi:

```
switch(error.code)
{
case 1: message = 'Permission Denied'; break;
case 2: message = 'Position Unavailable'; break;
case 3: message = 'Operation Timed Out'; break;
case 4: message = 'Unknown Error'; break;
}
O('status').innerHTML = message
```

Peta Google akan sepenuhnya interaktif dan dapat diperbesar oleh pengguna, yang juga dapat mengubah jenis peta menjadi citra satelit. kita dapat mengatur tingkat zoom atau jenis citra yang berbeda dengan memberikan nilai yang berbeda ke objek gopts. Misalnya, nilai 1 untuk zoom akan memperkecil paling jauh, dan 20 akan memperbesar paling banyak. Nilai SATELIT untuk properti google.maps.MapTypeId akan beralih ke citra satelit, atau HYBRID akan menggabungkan data peta dan satelit.

Catatan: Pengaturan sensor=false dari bagian ekor URL tempat skrip dimuat (dekat awal dokumen) harus disetel ke true jika kita mengetahui bahwa perangkat pengguna memiliki sensor GPS; jika tidak, biarkan apa adanya. Jika kita hanya ingin menampilkan peta Google untuk lokasi tertentu, dan tidak mengakses data lokasi pengguna, kita dapat menggunakan kode inti dalam fungsi yang diberikan, mengganti nilai lat dan long (dan lainnya) dengan yang kita pilih. Juga, jika kita lebih suka menggunakan peta Bing daripada Google, lihat http://tinyurl.com/bingmapsapi.

5.3 PENYIMPANAN LOKAL

Cookie adalah bagian penting dari Internet modern karena memungkinkan situs web untuk menyimpan potongan kecil informasi di mesin setiap pengguna yang dapat digunakan untuk tujuan pelacakan. Sekarang ini tidak seburuk kedengarannya, karena sebagian besar pelacakan yang terjadi membantu peselancar web dengan menyimpan nama pengguna dan kata sandi, membuat mereka tetap masuk ke jejaring sosial seperti Twitter atau Facebook, dan banyak lagi. Cookie juga dapat menyimpan preferensi kita secara lokal untuk cara kita mengakses situs web (daripada menyimpan pengaturan tersebut di server situs web) atau dapat digunakan untuk melacak keranjang belanja saat kita membuat pesanan di situs web e-niaga. Tapi ya, mereka juga dapat digunakan lebih agresif untuk melacak situs web yang sering kita kunjungi dan mendapatkan gambaran tentang minat kita untuk mencoba menargetkan iklan dengan lebih efektif. Itulah mengapa Uni Eropa sekarang mengharuskan semua situs web di dalam bordernya untuk memperingatkan kita tentang hal ini, dan membiarkan kita menonaktifkan cookie jika kita mau. Namun, sebagai pengembang web, pikirkan betapa bermanfaatnya menyimpan data di perangkat pengguna, terutama jika kita memiliki anggaran kecil untuk server komputer dan ruang disk. Misalnya, kita dapat membuat aplikasi dan layanan web dalam browser untuk mengedit dokumen pengolah kata, spreadsheet, dan

gambar grafik, menyimpan semua data ini di luar situs di komputer pengguna dan menjaga anggaran pembelian server kita serendah mungkin.

Dari sudut pandang pengguna, pikirkan tentang seberapa cepat dokumen dapat dimuat secara lokal daripada dari seluruh Web, terutama pada koneksi yang lambat. Plus, ada lebih banyak keamanan jika kita tahu bahwa situs web tidak menyimpan salinan dokumen Anda. Tentu saja, kita tidak pernah dapat menjamin bahwa situs web atau aplikasi web benar-benar aman, dan tidak boleh bekerja pada dokumen yang sangat sensitif menggunakan perangkat lunak (atau perangkat keras) yang dapat online. Tetapi untuk dokumen pribadi minimal seperti foto keluarga, kita mungkin merasa lebih nyaman menggunakan aplikasi web yang menyimpan secara lokal daripada yang menyimpan file ke server eksternal.

Menggunakan Penyimpanan Lokal

Masalah terbesar dengan menggunakan cookie untuk penyimpanan lokal adalah kita dapat menyimpan maksimal 4KB data di masing-masing. Cookie juga harus diteruskan bolak-balik pada setiap halaman yang dimuat ulang. Dan, kecuali server kita menggunakan enkripsi SSL (Secure Sockets Layer), setiap kali cookie dikirimkan, cookie akan berjalan dengan jelas. Tetapi dengan HTML5 kita memiliki akses ke ruang penyimpanan lokal yang jauh lebih besar (biasanya antara 5MB dan 10MB per domain tergantung pada browser) yang tersisa selama pemuatan halaman, dan di antara kunjungan situs web (dan bahkan setelah mematikan komputer dan mencadangkan kembali). Juga, data penyimpanan lokal tidak dikirim ke server pada setiap pemuatan halaman. kita menangani data penyimpanan lokal dalam pasangan kunci/nilai. Kuncinya adalah nama yang ditetapkan untuk mereferensikan data, dan nilainya dapat menampung semua jenis data, tetapi disimpan sebagai string. Semua data unik untuk domain saat ini, dan untuk alasan keamanan, penyimpanan lokal apa pun yang dibuat oleh situs web dengan domain berbeda terpisah dari penyimpanan lokal saat ini, dan tidak dapat diakses oleh domain apa pun selain domain yang menyimpan data tersebut.

Obyek Penyimpanan lokal

Anda mendapatkan akses ke penyimpanan lokal melalui objek localStorage. Untuk menguji apakah objek ini tersedia, kita menanyakan jenisnya untuk memeriksa apakah objek telah ditentukan atau belum, seperti ini:

```
if (typeof localStorage == 'undefined')
{
// Local storage is not available, tell the user and quit.
// Or maybe offer to save data on the web server instead?
}
```

Bagaimana kita menangani kurangnya penyimpanan lokal yang tersedia akan tergantung pada tujuan kita menggunakannya, jadi kode yang kita tempatkan di dalam pernyataan if akan terserah Anda. Setelah kita memastikan bahwa penyimpanan lokal tersedia, kita dapat mulai menggunakannya dengan metode setItem dan getItem dari objek localStorage, seperti ini:

```
localStorage.setItem('username', 'ceastwood')
localStorage.setItem('password', 'makemyday')
```

Untuk mengambil data ini nanti, berikan kunci ke metode getItem, seperti ini:

```
username = localStorage.getItem('username')
password = localStorage.getItem('password')
```

Tidak seperti menyimpan dan membaca cookie, kita dapat memanggil metode ini kapan saja kita suka, tidak hanya sebelum header apa pun dikirim oleh server web. Nilai yang disimpan akan tetap berada di penyimpanan lokal hingga dihapus dengan cara berikut:

```
localStorage.removeItem('username')
localStorage.removeItem('password')
```

Atau, kita dapat menghapus total penyimpanan lokal untuk domain saat ini dengan memanggil metode clear, seperti ini:

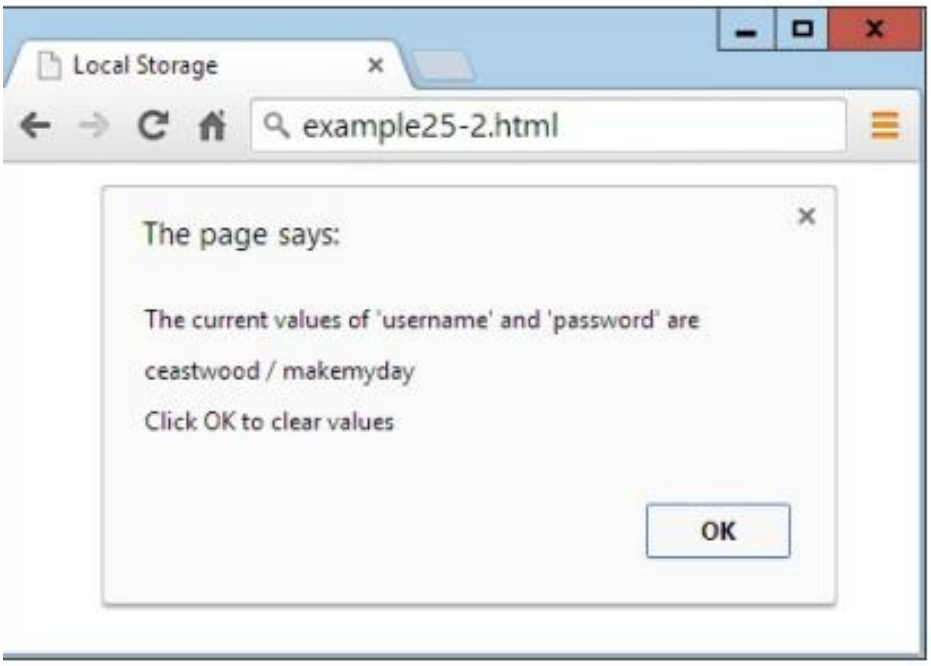
```
localStorage.clear()
```

Contoh 4 menggabungkan contoh sebelumnya ke dalam satu dokumen yang menampilkan nilai saat ini dari dua kunci dalam pesan peringatan pop-up, yang awalnya akan nol. Kemudian kunci dan nilai disimpan ke penyimpanan lokal, diambil, dan ditampilkan kembali, kali ini memiliki nilai yang ditetapkan. Akhirnya, kunci dihapus dan kemudian upaya untuk mengambil nilai-nilai ini dilakukan lagi, tetapi nilai yang dikembalikan sekali lagi nol. Gambar 5.5 menunjukkan yang kedua dari tiga pesan peringatan ini.

Contoh 4. Mendapatkan, mengatur, dan menghapus data penyimpanan lokal.

```
if (typeof localStorage == 'undefined')
{
alert("Local storage is not available")
}
else
{
username = localStorage.getItem('username')
password = localStorage.getItem('password')
alert("The current values of 'username' and 'password' are\n\n" +
username + " " + password + "
Click OK to assign values")
localStorage.setItem('username', 'ceastwood')
localStorage.setItem('password', 'makemyday')
username = localStorage.getItem('username')
password = localStorage.getItem('password')
alert("The current values of 'username' and 'password' are
" +
username + " " + password + "\n\nClick OK to clear values")
localStorage.removeItem('username')
localStorage.removeItem('password')
username = localStorage.getItem('username')
password = localStorage.getItem('password')
alert("The current values of 'username' and 'password' are\n\n" +
username + " / " + password)
```

}



Gambar 5.3 Dua kunci dan nilainya dibaca dari penyimpanan lokal

5.4 PEKERJA WEB

Dengan pekerja web, kita dapat membuat bagian kode JavaScript yang akan berjalan di latar belakang, tanpa harus menyiapkan dan memantau interupsi. Sebaliknya, setiap kali memiliki sesuatu untuk dilaporkan, proses latar belakang kita berkomunikasi dengan JavaScript utama melalui penggunaan suatu peristiwa. Ini berarti penerjemah JavaScript dapat memutuskan bagaimana mengalokasikan irisan waktu dengan paling efisien, dan kode kita hanya perlu khawatir tentang berkomunikasi dengan tugas latar belakang setiap kali ada informasi untuk disampaikan. Contoh 5 menunjukkan bagaimana kita dapat menyiapkan pekerja web untuk menghitung tugas berulang di latar belakang—dalam hal ini, menghitung bilangan prima.

Contoh 5 Menyiapkan dan berkomunikasi dengan pekerja web

```
<!DOCTYPE html>
<html>
<head>
<title>Web Workers</title>
<script src='OSC.js'></script>
</head>
<body>
Current highest prime number:
<span id='result'>0</span>
<script>
if (!!window.Worker)
{
var worker = new Worker('worker.js')
worker.onmessage = function (event)
```

```
{
O('result').innerHTML = event.data;
}
}
else
{
alert("Web workers not supported")
}
</script>
</body>
</html>
```

Contoh ini pertama-tama membuat elemen dengan ID hasil di mana output dari web worker akan ditempatkan. Kemudian, di bagian <script>, window.Worker diuji melalui !! pasangan bukan operator. Ini memiliki efek mengembalikan nilai Boolean true jika metode Worker ada, dan false sebaliknya. Jika tidak benar, pesan akan ditampilkan di bagian lain yang memperingatkan kami bahwa pekerja web tidak tersedia. Jika tidak, objek pekerja baru dibuat dengan memanggil Pekerja, meneruskannya dengan nama file pekerja.js (ditampilkan segera). Kemudian acara onmessage dari objek pekerja baru dilampirkan ke fungsi anonim yang menempatkan pesan apa pun yang diteruskan oleh pekerja.js ke dalam properti innerHTML dari elemen yang dibuat sebelumnya. Pekerja web itu sendiri disimpan dalam file pekerja.js, dalam Contoh 6.

Contoh 6 Pekerja web pekerja.js

```
var n = 1
search: while (true)
{
n += 1
for (var i = 2; i <= Math.sqrt(n); i += 1)
{
if (n % i == 0) continue search
}
postMessage(n)
}
```

File ini memberikan nilai 1 ke variabel n. Kemudian loop terus menerus, menambah n dan memeriksanya untuk primality dengan metode brute force menguji semua nilai dari 1 ke akar kuadrat dari n untuk melihat apakah mereka membagi tepat menjadi n, tanpa sisa. Jika faktor ditemukan, perintah continue menghentikan serangan brute force dengan segera karena angkanya bukan bilangan prima, dan mulai memproses lagi pada nilai n berikutnya yang lebih tinggi. Namun, jika semua faktor yang mungkin diuji dan tidak ada yang menghasilkan sisa nol, maka n harus prima, sehingga nilainya diteruskan ke postMessage, yang mengirimkan pesan kembali ke acara onmessage dari objek yang menyiapkan pekerja web ini. Hasilnya terlihat seperti berikut:

Current highest prime number: 30477191

Untuk menghentikan pekerja web agar tidak berjalan, lakukan panggilan ke metode penghentian objek pekerja, seperti ini:

worker.terminate()

5.5 APLIKASI WEB OFFLINE

Dengan memberikan informasi yang tepat ke browser, kita dapat memberi tahu browser cara mengunduh semua komponen halaman web agar dapat dimuat dan dijalankan secara offline. File utama yang kita butuhkan adalah file manifes dengan ekstensi file .appcache. Untuk mengilustrasikan aplikasi web sederhana, saya memilih untuk membuat jam, sehingga file manifes diberi nama file clock.appcache, dan terlihat seperti Contoh 7.

Contoh 7. File clock.appcache

```
CACHE MANIFEST
clock.html
OSC.js
clock.css
clock.js
```

Baris pertama dalam file ini mendeklarasikannya sebagai file manifes. Baris berikut mencantumkan file yang perlu diunduh dan disimpan oleh browser, dimulai dengan Contoh 8 file clock.html, dan diikuti oleh file OSC.js, yang sama dengan yang digunakan oleh banyak contoh dalam buku ini.

Contoh 8. File clock.html

```
<!DOCTYPE html>
<html manifest='clock.appcache'>
<head>
<title>Offline Web Apps</title>
<script src='OSC.js'></script>
<script src='clock.js'></script>
<link rel='stylesheet' href='clock.css'>
</head>
<body>
<p>The time is: <output id='clock'></output></p>
</body>
</html>
```

File ini menyatakan bahwa ia memiliki file manifes yang tersedia dari dalam tag <html>, seperti ini:

```
<html manifest='clock.appcache'>
```

Catatan: Untuk mendukung aplikasi web offline, kita perlu menambahkan teks/manifest cache jenis MIME untuk ekstensi file .appcache ke server Anda, agar dapat mengirim file manifes menggunakan jenis yang benar. Ada jalan pintas yang rapi yang dapat kita gunakan

untuk ini, yaitu membuat file bernama .htaccess di folder yang sama dengan file yang akan tersedia offline, dengan konten berikut:

AddType text/cache-manifest .appcache
File OSC.js, clock.js, dan clock.css kemudian diimpor dan digunakan oleh dokumen. JavaScript di clock.js
File clock.js

```
setInterval(function()
{
O('clock').innerHTML = new Date()
}, 1000)
```

Ini adalah fungsi anonim yang sangat sederhana yang dilampirkan pada interval yang berulang sekali setiap detik untuk menyimpan tanggal dan waktu saat ini ke dalam properti innerHTML dari elemen <output> yang memiliki ID jam. File terakhir adalah file clock.css, yang hanya menerapkan gaya tebal ke elemen <output>.

Contoh 9 File jam.css

```
output { font-weight:bold; }
```

Selama file clock.appcache mencantumkan semuanya, keempat file ini (clock.html, OSC.js, clock.css, dan clock.js) bersama-sama membentuk aplikasi web offline yang berfungsi, yang akan diunduh dan tersedia secara lokal oleh browser web apa pun yang memahami aplikasi web offline. Saat dijalankan, outputnya terlihat seperti ini:

The time is: Thu Jul 19 2018 15:24:26 GMT+0000 (GMT Standard Time)

5.6 DROP-DOWN

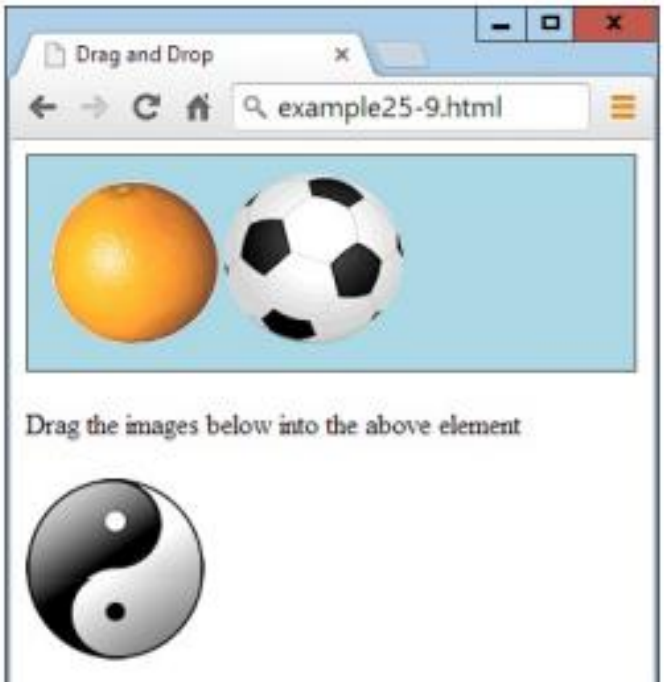
Anda dapat dengan mudah mendukung menyeret dan menjatuhkan objek pada halaman web dengan menyiapkan event handler untuk event ondragstart, ondragover, dan ondrop, seperti pada Contoh 10.

Contoh 10 Menyeret dan menjatuhkan objek

```
<!DOCTYPE HTML>
<html>
<head>
<title>Drag and Drop</title>
<script src='OSC.js'></script>
<style>
#dest {
background:lightblue;
border :1px solid #444;
width :320px;
height :100px;
padding :10px;
}
```

```
</style>
</head>
<body>
<div id='dest' ondrop='drop(event)' ondragover='allow(event)'></div><br>
Drag the image below into the above element<br><br>
<img id='source1' src='image1.png' draggable='true' ondragstart='drag(event)'>
<img id='source2' src='image2.png' draggable='true' ondragstart='drag(event)'>
<img id='source3' src='image3.png' draggable='true' ondragstart='drag(event)'>
<script>
function allow(event)
{
event.preventDefault()
}
function drag(event)
{
event.dataTransfer.setData('image/png', event.target.id)
}
function drop(event)
{
event.preventDefault()
var data=event.dataTransfer.getData('image/png')
event.target.appendChild(O(data))
}
</script>
</body>
</html>
```

Setelah menyiapkan HTML, judul, dan memuat dalam file OSC.js, dokumen ini menata elemen div dengan ID tujuan, memberinya warna latar belakang, batas, dimensi yang ditetapkan, dan bantalan. Kemudian, di bagian <body>, elemen div dibuat, dan event ondrop dan ondragover memiliki fungsi event handler drop dan allow dilampirkan padanya. Setelah ini ada beberapa teks, dan kemudian tiga gambar ditampilkan dengan properti draggable yang disetel ke true, dan fungsi drag dilampirkan ke event ondragstart masing-masing. Di bagian <script>, fungsi allow event handler hanya mencegah tindakan default untuk menyeret (yaitu melarangnya), sedangkan fungsi drag event handler memanggil metode setData dari objek dataTransfer event, meneruskannya dengan tipe MIME image/png dan target.id dari acara (yang merupakan objek yang diseret). Objek dataTransfer menyimpan data yang sedang diseret selama operasi drag-and-drop. Terakhir, fungsi drop event handler juga memotong tindakan defaultnya sehingga drop diperbolehkan, dan kemudian mengambil konten objek yang diseret dari objek dataTransfer, meneruskannya ke tipe MIME objek. Kemudian data yang dijatuhkan ditambahkan ke target (yang merupakan div tujuan) menggunakan metode appendChild-nya.



Gambar 5.4 Dua gambar telah diseret dan dijatuhkan

Peristiwa lain yang dapat kita lampirkan untuk menyertakan ondragenter ketika operasi seret memasuki elemen, ondragleave ketika seseorang meninggalkan elemen, dan ondragend ketika operasi menyeret berakhir, yang dapat kita gunakan, misalnya, untuk memodifikasi kursor selama operasi ini.

5.7 PESAN LINTAS DOKUMEN

Anda telah melihat perpesanan yang digunakan sedikit lebih awal, di bagian pekerja web. Namun, saya tidak merinci apa pun, karena itu bukan topik inti yang sedang dibahas, dan pesan itu hanya diposting ke dokumen yang sama. Tetapi untuk alasan keamanan yang jelas, pengiriman pesan lintas dokumen perlu diterapkan dengan hati-hati, jadi kita perlu memahami sepenuhnya cara kerjanya jika kita berencana untuk menggunakannya. Sebelum HTML5, pengembang browser melarang pembuatan skrip lintas situs, tetapi selain memblokir situs serangan potensial, ini mencegah komunikasi antara halaman yang sah—artinya interaksi dokumen dalam bentuk apa pun umumnya harus terjadi melalui Ajax dan server web pihak ketiga, yang tidak praktis. dan fiddly untuk membangun dan memelihara. Tetapi pesan web sekarang memungkinkan skrip untuk berinteraksi melintasi batas-batas ini dengan menggunakan beberapa batasan keamanan yang masuk akal untuk mencegah upaya peretasan yang berbahaya. Hal ini dicapai melalui penggunaan metode postMessage, yang memungkinkan pesan teks biasa dikirim dari satu domain ke domain lainnya. Ini mengharuskan JavaScript terlebih dahulu mendapatkan objek Window dari dokumen penerima, membiarkan pesan dikirim ke berbagai jendela, bingkai, atau iframe lain yang terkait langsung dengan dokumen pengirim. Acara pesan yang diterima memiliki atribut berikut:

data

Pesan yang masuk

origin

Asal dokumen pengirim, termasuk skema, nama host, dan port

source
Jendela sumber dokumen pengirim
Kode untuk mengirim pesan hanyalah satu instruksi, di mana kita meneruskan pesan tema yang akan dikirim dan domain tempat kode tersebut diterapkan, seperti pada Contoh 11.

Contoh 11 Mengirim pesan web ke iframe

```
<!DOCTYPE HTML>
<html>
<head>
<title>Web Messaging (a)</title>
<script src='OSC.js'></script>
</head>
<body>
<iframe id='frame' src='example25-11.html' width='360' height='75'></iframe>
<script>
count = 1
setInterval(function()
{
O('frame').contentWindow.postMessage('Message ' + count++, '*')
}, 1000)
</script>
</body>
</html>
```

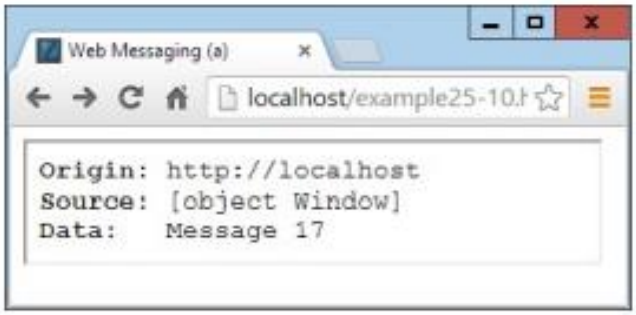
Di sini, file OSC.js biasanya digunakan untuk menarik fungsi O, dan kemudian elemen iframe dengan ID bingkai dibuat, yang dimuat dalam Contoh 12. Kemudian, di dalam bagian <script>, jumlah variabel diinisialisasi ke 1 dan interval berulang diatur agar terjadi setiap detik untuk memposting string 'Pesan' (menggunakan metode postMessage) bersama dengan nilai hitungan saat ini, yaitu kemudian bertambah. Panggilan postMessage dilampirkan ke properti contentWindow dari objek iframe, bukan objek iframe itu sendiri. Ini penting karena pesan web mengharuskan posting dibuat ke jendela, bukan ke objek di dalam jendela.

Contoh 12. Menerima pesan dari dokumen lain

```
<!DOCTYPE HTML>
<html>
<head>
<title>Web Messaging (b)</title>
<style>
#output {
font-family:"Courier New";
white-space:pre;
}
</style>
<script src='OSC.js'></script>
</head>
<body>
```

```
<div id='output'>Received messages will display here</div>
<script>
window.onmessage = function(event)
{
O('output').innerHTML =
'<b>Origin:</b> ' + event.origin + '<br>' +
'<b>Source:</b> ' + event.source + '<br>' +
'<b>Data:</b> ' + event.data
}
</script>
</body>
</html> <!DOCTYPE HTML>
<html>
<head>
<title>Web Messaging (b)</title>
<style>
#output {
font-family:"Courier New";
white-space:pre;
}
</style>
<script src='OSC.js'></script>
</head>
<body>
<div id='output'>Received messages will display here</div>
<script>
window.onmessage = function(event)
{
O('output').innerHTML =
'<b>Origin:</b> ' + event.origin + '<br>' +
'<b>Source:</b> ' + event.source + '<br>' +
'<b>Data:</b> ' + event.data
}
</script>
</body>
</html>
```

Contoh ini mengatur sedikit gaya untuk membuat output lebih jelas, kemudian membuat elemen div dengan output ID, di mana konten pesan yang diterima akan ditempatkan. Di bagian <script> ada satu fungsi anonim yang dilampirkan ke acara pesan di jendela. Dalam fungsi ini, nilai properti event.origin, event.source, dan event.data kemudian ditampilkan, seperti yang ditunjukkan pada Gambar 5.7.



Gambar 5.5 iframe sejauh ini telah menerima 17 pesan

Pesan web hanya berfungsi di seluruh domain, jadi kita tidak dapat mengujinya dengan memuat file dari sistem file; kita harus menggunakan server web. Seperti yang kita lihat dari Gambar 5.5, asalnya adalah http://localhost karena contoh-contoh ini berjalan di server pengembangan lokal. Sumbernya adalah objek Window, dan nilai pesan saat ini adalah Pesan 17. Saat ini, Contoh 13 sama sekali tidak aman karena nilai domain yang diteruskan ke postMessage adalah wildcard *:

```
O('frame').contentWindow.postMessage('Message ' + count++, '*')
```

Untuk mengarahkan pesan hanya ke dokumen yang berasal dari domain tertentu, kita dapat mengubah parameter ini. Dalam kasus saat ini, nilai http://localhost akan memastikan bahwa hanya dokumen yang dimuat dari server lokal yang akan dikirimi pesan apa pun, seperti ini:

```
O('frame').contentWindow.postMessage('Message ' + count++, 'http://localhost')
```

Demikian juga, sebagaimana adanya, program pendengar menampilkan setiap dan semua pesan yang diterimanya. Ini juga bukan keadaan yang sangat aman, karena dokumen berbahaya yang juga ada di browser dapat mencoba mengirim pesan yang dapat diakses oleh kode pendengar yang tidak waspada di dokumen lain. Oleh karena itu, kita dapat membatasi pesan yang akan ditanggapi oleh pendengar kita menggunakan pernyataan if, seperti ini:

```
window.onmessage = function(event)
{
if (event.origin) == 'http://localhost')
{
O('output').innerHTML =
'<b>Origin:</b> ' + event.origin + '<br>' +
'<b>Source:</b> ' + event.source + '<br>' +
'<b>Data:</b> ' + event.data
}
}
```

Perhatikan bahwa, jika kita selalu menggunakan domain yang tepat untuk situs tempat kita bekerja, komunikasi pesan web kita akan lebih aman. Namun, perlu diketahui bahwa karena pesan dikirim dengan jelas, mungkin ada ketidakamanan di beberapa browser atau plugin browser yang mungkin membuat komunikasi semacam ini tidak aman. Maka, salah satu cara

untuk meningkatkan keamanan kita adalah dengan membuat skema penyamaran atau enkripsi kita sendiri untuk semua pesan web Anda, dan juga pertimbangkan untuk memperkenalkan protokol komunikasi dua arah kita sendiri untuk memverifikasi setiap pesan sebagai asli.

Biasanya, kita tidak akan memberi tahu pengguna tentang nilai asal atau sumber, dan hanya akan menggunakannya untuk pemeriksaan keamanan. Namun, contoh-contoh ini menampilkan nilai-nilai tersebut untuk membantu kita bereksperimen dengan pesan web dan melihat apa yang terjadi. Selain iframe, dokumen di jendela pop-up dan tab lain juga dapat saling berbicara menggunakan metode ini.

5.8 MIKRODATA

Microdata adalah subset dari HTML yang dirancang untuk menyediakan metadata ke dokumen agar memiliki arti bagi perangkat lunak, sama seperti memiliki arti bagi pembaca dokumen. Microdata menyediakan atribut tag baru berikut: itemscope, itemtype, itemid, itemref, dan itemprop. Dengan menggunakan ini, kita dapat dengan jelas menentukan properti suatu item seperti buku, menyediakan berbagai informasi yang dapat digunakan komputer untuk memahami, misalnya, penulisnya, penerbit, kontennya, dan sebagainya. Atau, lebih sering akhir-akhir ini, microdata penting untuk mesin pencari dan situs jejaring sosial. Contoh 14 membuat bio singkat untuk George Washington seolah-olah itu adalah profil di situs jejaring sosial, dengan mikrodata ditambahkan ke berbagai elemen (ditampilkan disorot dalam huruf tebal). Hasilnya terlihat seperti Gambar 5.8, yang akan terlihat sama dengan atau tanpa microdata, karena tidak pernah terlihat oleh pengguna.

Contoh 14 Menambahkan mikrodata ke HTML

```
<!DOCTYPE html>
<html>
<head>
<title>Microdata</title>
</head>
<body>
<section itemscope itemtype='http://schema.org/Person'>
<img itemprop='image' src='gw.jpg' alt='George Washington'
align='left' style='margin-right:10px'>
<h2 itemprop='name'>George Washington</h2>
<p>I am the first <span itemprop='jobTitle'>US President</span>.
My website is: <a itemprop='url'
href='http://georgewashington.si.edu'>georgewashington.si.edu</a>.
My address is:</p>
<address itemscope itemtype='http://schema.org/PostalAddress'
itemprop='address'>
<span itemprop='streetAddress'>1600 Pennsylvania Avenue</span>,<br>
<span itemprop='addressLocality'>Washington</span>,<br>
<span itemprop='addressRegion'>DC</span>,<br>
<span itemprop='postalCode'>20500</span>,<br>
<span itemprop='addressCountry'>United States</span>.
</address>
```

```
</section>
</body>
</html>
```

Gambar 5.6 Dokumen ini berisi microdata,yang tidak memiliki visi.

Peramban belum benar-benar melakukan apa pun dengan mikrodata, tetapi masih sangat berharga untuk mengetahuinya. Menggunakan microdata yang tepat memberikan banyak informasi ke mesin telusur seperti Google atau Bing, dan dapat membantu mempromosikan halaman beranotasi yang jelas di peringkat dibandingkan dengan situs yang tidak menerapkan microdata. Namun, di beberapa titik browser juga dapat menemukan kegunaan untuk informasi ini, dan kita akan dapat menentukan apakah mereka mendukungnya atau tidak dengan memeriksa apakah metode getItems ada, seperti ini:

```
if (!!document.getItems)
{
// Microdata is supported
}
else
{
// Microdata is not supported
}
```

!! pair of not operator adalah cara singkat untuk mengembalikan nilai Boolean yang mewakili keberadaan (atau ketiadaan) metode getItems. Jika ada, maka true dikembalikan dan microdata didukung; jika tidak, false dikembalikan. Saat ini, hanya browser Mozilla Firefox dan Opera yang mendukung akses microdata, tetapi browser lain pasti akan segera menyusul. Ketika mereka melakukannya, kita akan dapat mengekstrak data ini dengan cara berikut, di mana (setelah halaman dimuat) objek data diambil dari panggilan ke getItems, dan nilai untuk kunci 'jobTitle' (seperti contoh) diambil dengan mengakses objek properti objek data, dan kemudian mengambil textContent objek Properti terakhir:

```
window.onload  = function()
{
if (!!document.getItems)
{
```

```
data = document.getItems('http://schema.org/Person')[0]
alert(data.properties['jobTitle'][0].textContent)
}
}
```

Browser yang mendukung fitur ini akan ditampilkan seperti Gambar 5.9, tetapi browser lain tidak akan memicu jendela pop-up

```
window.onload  = function()
{
if (!!document.getItems)
{
data = document.getItems('http://schema.org/Person')[0]
alert(data.properties['jobTitle'][0].textContent)
}
}
```

Gambar 5.7 Menampilkan nilai untuk kunci mikrodata 'jobTitle'

Google telah menyatakan bahwa ia pasti menggunakan microdata ketika menemukannya, dan bahwa microdata juga merupakan format cuplikan pilihan untuk Google+, jadi ada baiknya mulai menambahkannya ke HTML kita jika berlaku. Untuk perincian lengkap dari segudang properti microdata yang tersedia, lihat http://schema.org, yang juga merupakan referensi untuk skema microdata seperti yang dideklarasikan dalam properti itemType.

5.9 TAG HTML5 LAINNYA

Ada sejumlah tag HTML5 baru lainnya yang belum diterapkan di banyak browser, dan oleh karena itu saya tidak akan membahasnya (terutama karena spesifikasinya dapat berubah). Tapi, demi kelengkapan, tag tersebut adalah <article>, <aside>, <details>, <figcaption>, <figure>, <footer>, <header>, <hgroup>, <keygen>, <mark> , <menuitem>, <meter>, <nav>, <output>, <progress>, <rp>, <rt>, <ruby>, <section>, <summary>, <time>,

dan <wbr>. kita dapat memperoleh informasi lebih lanjut tentang ini dan semua tag HTML5 lainnya di http://tinyurl.com/h5markup (lihat elemen yang menampilkan ikon BARU).

Ringkasan

Ini menyimpulkan pengantar kita ke HTML5. kita sekarang memiliki sejumlah fitur baru yang kuat untuk membuat situs web yang lebih dinamis dan menarik. Di bab terakhir, saya akan menunjukkan kepada kita bagaimana kita dapat menyatukan semua teknologi yang berbeda dalam buku ini untuk membuat situs jejaring sosial mini.

BAB 6
MEMAHAMI DASAR JAVASCRIPT

JavaScript adalah bahasa yang sangat kuat, dan kita dapat menggunakannya untuk menambahkan fitur hebat guna meningkatkan pengalaman pengguna. Dalam bab ini, kami memberi tahu kita sedikit tentang jenis interaktivitas yang dapat kita tambahkan ke halaman web dengan JavaScript dan kemudian menunjukkan cara menambahkan JavaScript ke halaman.

6.1 MELIHAT DUNIA JAVASCRIPTLD DARI JAVASCRIPT

JavaScript digunakan untuk pemrograman web untuk meningkatkan atau menambah pengalaman pengguna saat menggunakan halaman web. Bagian ini melihat beberapa aspek JavaScript yang akan membantu kita memahami bahasa dan memberi kita dasar yang baik di mana kita akan benar-benar dapat membuat halaman web kita menonjol.

JavaScript bukan Java

Jangan bingung dengan namanya. JavaScript sama sekali tidak ada hubungannya dengan Java — kopi atau bahasa pemrograman. Nama JavaScript muncul karena orang-orang pemasaran ingin menggunakan faktor "keren" ketika bahasa pemrograman Java masih baru dan mengkilap. Java adalah bahasa berat yang tidak selalu berjalan di komputer semua orang; orang harus menginstal perangkat lunak tambahan untuk menjalankannya. Meskipun kuat, Java tidak dimaksudkan untuk jenis pemrograman web yang biasanya perlu kita lakukan. JavaScript, di sisi lain, disertakan dengan hampir semua browser web dan tidak perlu menginstal apa pun. kita menggunakan JavaScript untuk membuat halaman menjadi hidup, dengan kolom formulir yang terisi otomatis, dan semua jenis lonceng dan peluit yang meningkatkan pengalaman pengguna.

Salah satu hal paling umum yang kami dengar dari orang-orang nonteknis adalah membingungkan atau menyebut JavaScript, "Java." Sekarang setelah kita tahu bahwa keduanya benar-benar berbeda, kita tidak akan melakukan hal yang sama! Namun, kita harus menahan keinginan untuk mengoreksi orang ketika kita mendengar mereka mengacaukan kedua bahasa tersebut. JavaScript didefinisikan oleh spesifikasi yang dikenal sebagai ECMA-262. Browser web memiliki berbagai tingkat dukungan untuk spesifikasi ECMA-262, sehingga versi persis JavaScript yang tersedia di browser bervariasi sesuai dengan versi browser yang digunakan.

Mengetahui apa yang dapat dilakukan JavaScript

JavaScript adalah bagian integral dari halaman web saat ini. Saat kita melihat sesuatu seperti Google Maps, di mana kita dapat menggulir ke kiri dan kanan hanya dengan menyeret peta, itulah JavaScript di balik layar. Saat kita membuka situs untuk mencari detail penerbangan, dan situs tersebut secara otomatis menyarankan bandara saat kita mengetik di bidang, itulah JavaScript. Widget yang tak terhitung jumlahnya dan peningkatan kegunaan yang kita anggap remeh saat menggunakan web sebenarnya adalah program JavaScript. Program JavaScript berjalan di browser web pengguna. Ini adalah berkah sekaligus kutukan. Di satu sisi, dengan berjalan di browser web pengguna berarti server kita tidak perlu menjalankan program tersebut. Di sisi lain, dengan berjalan di browser pengguna itu berarti

program kita berjalan sedikit berbeda tergantung pada versi browser yang digunakan pengguna di situs Anda. Faktanya, pengguna mungkin menonaktifkan JavaScript sepenuhnya! Meskipun secara teoritis semua JavaScript harus berjalan sama, dalam praktiknya tidak. Internet Explorer, terutama versi lama seperti 6 dan 7, menafsirkan JavaScript dengan cara yang sangat berbeda dari browser lain seperti Firefox dan Chrome. Ini berarti kita perlu membuat dua program berbeda atau dua cara berbeda untuk membuat hal yang sama berfungsi di halaman web Anda.

6.2 MEMERIKSA CARA MENAMBAHKAN JAVASCRIPT KE HALAMAN

Meskipun JavaScript disertakan di browser web semua orang, kita masih perlu memprogram tindakan yang kita inginkan terjadi di halaman Anda. kita dapat menata halaman kita dengan *Cascading Style Sheets* (CSS) yang ditambahkan langsung ke HTML atau merujuk file CSS terpisah. Demikian pula, bagian ini menunjukkan berbagai cara untuk memasukkan JavaScript ke dalam halaman. kita dapat menambahkan JavaScript langsung ke file HTML, mereferensikan file JavaScript terpisah, atau melakukan keduanya — dan kami membantu kita memahami kapan setiap opsi sesuai.

Menambahkan tag JavaScript

Anda menambahkan JavaScript ke halaman dengan tag <script>, seperti ini:

```
<script type=”text/javascript”>
// JavaScript goes here
</script>
```

Anda mungkin melihat berbagai cara untuk memasukkan JavaScript dalam sebuah halaman, seperti “text/ecmascript” atau tanpa atribut type sama sekali, cukup dengan tag <script> kosong. Metode ini bekerja, semacam, dan beberapa di antaranya secara teknis benar. Tapi salah satu yang paling sering kita lihat dan yang paling beruntung adalah yang ditampilkan, dengan jenis "teks/javascript".Jika kita bertanya-tanya, set garis miring ganda yang kita lihat di sini contoh memulai komentar, yang akan kami ceritakan lebih lanjut di bab berikutnya.

Menambahkan JavaScript ke HTML halaman

Selalu posisikan kode JavaScript setelah tag pembuka <script type="text/javascript"> dan sebelum tag </script> penutup. kita dapat menyertakan tag tersebut di bagian <head> dan <body> pada halaman. Berikut adalah contoh yang menampilkan JavaScript di dua lokasi berbeda dalam satu halaman:

```
<!doctype html>
<html>
<head>
<title>Another Basic Page</title>
<script type=”text/javascript”>
 // JavaScript goes here
</script>
</head>
<body>
<h1>Here’s another basic page</h1>
<script type=”text/javascript”>
```

```
// JavaScript can also go here
</script>
</body>
</html>
```

Anda sebenarnya dapat menempatkan elemen skrip terpisah sebanyak yang kita inginkan di halaman, tetapi biasanya tidak ada alasan untuk melakukannya.

Menggunakan JavaScript eksternal

Contoh yang baru saja kita lihat menunjukkan JavaScript di dalam halaman, dengan cara yang sama seperti kita dapat menambahkan CSS di dalam halaman. Meskipun metode itu bekerja untuk skrip kecil dan tentu saja berguna untuk menunjukkan contoh dalam buku ini, cara yang lebih baik untuk menambahkan JavaScript adalah dengan menggunakan file JavaScript eksternal. Menggunakan file JavaScript eksternal adalah konsep yang sama dengan menggunakan file eksternal untuk CSS. Melakukannya akan mendorong penggunaan kembali dan membuat pemecahan masalah dan perubahan menjadi lebih mudah. kita dapat menambahkan JavaScript eksternal dengan menggunakan atribut src, seperti ini:

```
<script type=”text/javascript”
 src=”externalfile.js”></script>
```

Contoh ini memuat file “externalfile.js” dari direktori yang sama di server web. Isi file itu diharapkan berupa JavaScript. Perhatikan dalam contoh ini bahwa tidak ada apa pun di antara tag pembuka <script>dan penutup </script>. Saat menggunakan file JavaScript eksternal, kita tidak dapat menempatkan JavaScript di dalam kumpulan tag yang sama. kita dapat menambahkan referensi, seperti yang ditunjukkan, di mana saja di halaman, tetapi tempat tradisional untuk itu ada di bagian <head> halaman. Perhatikan juga tidak ada yang mencegah kita menggunakan file JavaScript eksternal bersama dengan JavaScript dalam halaman, jadi ini benar-benar valid:

```
<!doctype html>
<html>
<head>
<title>Another Basic Page</title>
<script type=”text/javascript”>
 // JavaScript goes here
</script>
</head>
<body>
<h1>Here’s another basic page</h1>
<script type=”text/javascript”>
 // JavaScript can also go here
</script>
</body>
</html>
```

Contoh ini memuat file JavaScript eksternal dan kemudian menjalankan beberapa JavaScript tepat di dalam halaman.

6.3 MENJELAJAHI JAVASCRIPT

JavaScript menghadirkan fungsionalitas dinamis ke situs web Anda. Setiap kali kita melihat sesuatu muncul saat kita mengarahkan mouse ke item di browser, atau melihat teks, warna, atau gambar baru muncul di halaman di depan mata Anda, atau mengambil objek di halaman dan menyeretnya ke lokasi baru —semua hal itu dilakukan melalui JavaScript. Ini menawarkan efek yang tidak mungkin, karena berjalan di dalam browser dan memiliki akses langsung ke semua elemen dalam dokumen web. JavaScript pertama kali muncul pada browser Netscape Navigator pada tahun 1995, bertepatan dengan penambahan dukungan teknologi Java pada browser tersebut. Karena kesan awal yang salah bahwa JavaScript adalah spin-off dari Java, ada beberapa kebingungan jangka panjang atas hubungan mereka. Namun, penamaan itu hanya taktik pemasaran untuk membantu bahasa scripting baru mendapatkan keuntungan dari popularitas bahasa pemrograman Java.

JavaScript memperoleh kekuatan baru ketika elemen HTML halaman web mendapat definisi yang lebih formal dan terstruktur dalam apa yang disebut Model Objek Dokumen, atau DOM. DOM membuatnya relatif mudah untuk menambahkan paragraf baru atau fokus pada sepotong teks dan mengubahnya. Karena JavaScript dan PHP mendukung banyak sintaks pemrograman terstruktur yang digunakan oleh bahasa pemrograman C, mereka terlihat sangat mirip satu sama lain. Keduanya juga bahasa tingkat tinggi; misalnya, mereka diketik dengan lemah, jadi mudah untuk mengubah variabel ke tipe baru hanya dengan menggunakannya dalam konteks baru.

Sekarang setelah kita mempelajari PHP, kita seharusnya menemukan JavaScript dengan lebih mudah. Dan kita akan senang melakukannya, karena itu adalah inti dari teknologi Web 2.0 Ajax yang menyediakan antarmuka web yang lancar yang (bersama dengan fitur HTML5) yang diharapkan oleh pengguna web cerdas akhir-akhir ini.

6.4 JAVASCRIPT DAN TEKS HTML

JavaScript adalah bahasa skrip sisi klien yang berjalan sepenuhnya di dalam browser web. Untuk memanggilnya, kita menempatkannya di antara tag HTML pembuka <script> dan penutup </script>. Dokumen HTML 4.01 "Hello World" yang khas menggunakan JavaScript mungkin terlihat seperti Contoh 1.

Contoh 1 "Hello World" ditampilkan menggunakan JavaScript

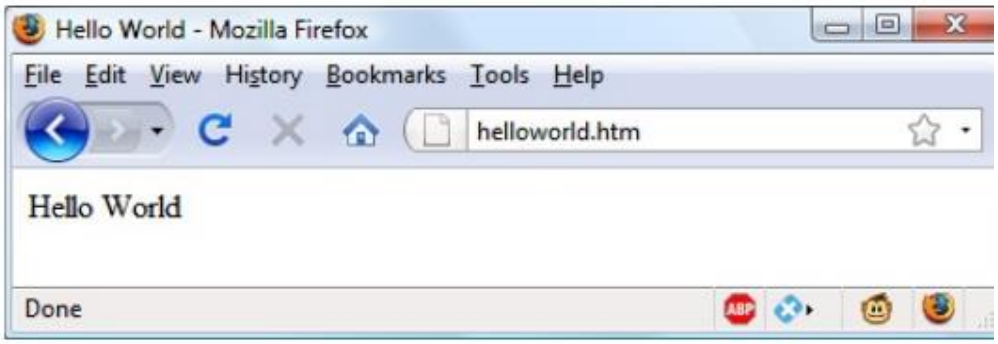
```
<html>
<head><title>Hello World</title></head>
<body>
<script type="text/javascript">
document.write("Hello World")
</script>
<noscript>
Your browser doesn't support or has disabled JavaScript
</noscript>
</body>
</html>
```

Anda mungkin pernah melihat halaman web yang menggunakan tag HTML <script language="javascript">, tetapi penggunaan itu sekarang sudah tidak digunakan lagi. Contoh ini menggunakan <script type="text/javascript"> yang lebih baru dan lebih disukai, atau kita bisa menggunakan <script> sendiri jika kita mau.

Di dalam tag <script> terdapat satu baris kode JavaScript yang menggunakan setara dengan perintah PHP echo atau print, document.write. Seperti yang kita harapkan, itu hanya menampilkan string yang disediakan ke dokumen saat ini, di mana ia ditampilkan. kita mungkin juga memperhatikan bahwa, tidak seperti PHP, tidak ada tanda titik koma (;). Ini karena baris baru memiliki tujuan yang sama dengan titik koma di JavaScript. Namun, jika kita ingin memiliki lebih dari satu pernyataan dalam satu baris, kita perlu menempatkan titik koma setelah setiap perintah kecuali yang terakhir. Tentu saja, jika diinginkan, kita dapat menambahkan titik koma di akhir setiap pernyataan dan JavaScript kita akan berfungsi dengan baik.

Hal lain yang perlu diperhatikan dalam contoh ini adalah pasangan tag <noscript> dan </noscript>. Ini digunakan ketika kita ingin menawarkan HTML alternatif kepada pengguna yang browsernya tidak mendukung JavaScript atau yang menonaktifkannya. Menggunakan tag ini terserah Anda, karena tidak diperlukan, tetapi kita benar-benar harus menggunakannya karena biasanya tidak terlalu sulit untuk menyediakan alternatif HTML statis untuk operasi yang kita berikan menggunakan JavaScript. Namun, contoh lainnya dalam buku ini akan menghilangkan tag <noscript>, karena kami berfokus pada apa yang dapat kita lakukan dengan JavaScript, bukan apa yang dapat kita lakukan tanpanya. Ketika Contoh html sebelumnya di dimuat, browser web dengan JavaScript diaktifkan akan menampilkan yang berikut:

Hello World



Gambar 6.1 JavaScript, diaktifkan dan berfungsi

Sebuah browser dengan JavaScript dinonaktifkan akan menampilkan pesan pada gambar berikut ini:

Gambar 6.2 JavaScript telah dinonaktifkan

Menggunakan Script Dalam Document Head

Selain menempatkan skrip di dalam badan dokumen, kita dapat meletakkannya di bagian <head>, yang merupakan tempat ideal jika kita ingin menjalankan skrip saat halaman dimuat. Jika kita menempatkan kode dan fungsi penting di sana, kita juga dapat memastikan bahwa kode dan fungsi tersebut siap digunakan segera oleh bagian skrip lain dalam dokumen yang bergantung padanya. Alasan lain untuk menempatkan skrip di kepala dokumen adalah untuk mengaktifkan JavaScript untuk menulis hal-hal seperti tag meta ke dalam bagian <head>, karena lokasi skrip kita adalah bagian dari dokumen yang ditulisnya secara default.

Browser Lama dan Tidak Standar

Jika kita perlu mendukung browser yang tidak menawarkan skrip, kita perlu menggunakan tag komentar HTML (<!-- dan -->) untuk mencegah mereka menemukan kode skrip yang seharusnya tidak mereka lihat. Contoh dibawah ini menunjukkan bagaimana jika kita menambahkannya ke kode skrip Anda.

Contoh 2 "Hello World" dimodifikasi untuk browser non-JavaScript

```
<html>
<head><title>Hello World</title></head>
<body>
<script type="text/javascript"><!--
document.write("Hello World")
// --></script>
</body>
</html>
```

Di sini tag komentar HTML pembuka (<!--) telah ditambahkan langsung setelah pernyataan <script> pembuka dan tag komentar penutup (// -->) tepat sebelum skrip ditutup dengan </script>. Garis miring ganda (//) digunakan oleh JavaScript untuk menunjukkan bahwa sisa baris adalah komentar. Itu ada sehingga browser yang mendukung JavaScript akan mengabaikan yang berikut -->, tetapi browser non-JavaScript akan mengabaikan // sebelumnya, dan bertindak atas --> dengan menutup komentar HTML. Meskipun solusinya sedikit berbelit-belit, yang perlu kita ingat adalah menggunakan dua baris berikut untuk menyertakan JavaScript kita ketika kita ingin mendukung browser yang sangat lama atau tidak standar:

```
<script type="text/javascript"><!-
(Your JavaScript goes here...)
// --></script>
```

Namun, penggunaan komentar ini tidak diperlukan untuk browser apa pun yang dirilis selama beberapa tahun terakhir.

Ada beberapa bahasa skrip lain yang harus kita ketahui. Ini termasuk VBScript Microsoft, yang didasarkan pada bahasa pemrograman Visual Basic, dan Tcl, bahasa prototipe cepat. Mereka dipanggil dengan cara yang mirip dengan JavaScript, kecuali mereka menggunakan jenis teks/vbscript dan teks/tcl, masing-masing. VBScript hanya berfungsi di Internet Explorer; penggunaannya di browser lain membutuhkan plugin. Tcl selalu

membutuhkan plug-in. Jadi keduanya harus dianggap tidak standar, dan keduanya tidak tercakup dalam buku ini.

6.5 TERMASUK FILE JAVASCRIPT

Selain menulis kode JavaScript secara langsung dalam dokumen HTML, kita dapat memasukkan file kode JavaScript baik dari situs web kita atau dari mana saja di Internet. Sintaks untuk ini adalah:

```
<script type="text/javascript" src="script.js"></script>
```

Atau, untuk menarik file dari Internet, gunakan:

```
<script type="text/javascript" src="http://someserver.com/script.js">
</script>
```

Untuk file skrip itu sendiri, mereka tidak boleh menyertakan tag <script> atau </script>, karena tidak diperlukan: browser sudah mengetahui bahwa file JavaScript sedang dimuat. Menempatkannya di file JavaScript akan menyebabkan kesalahan. Menyertakan file skrip adalah cara yang lebih disukai untuk menggunakan file JavaScript pihak ketiga di situs web Anda. Dimungkinkan untuk mengabaikan parameter type="text/javascript"; semua browser modern default untuk mengasumsikan bahwa skrip berisi JavaScript.

6.6 MEN-DEBUG KESALAHAN JAVASCRIPT

Saat kita mempelajari JavaScript, penting untuk dapat melacak kesalahan pengetikan atau pengkodean lainnya. Tidak seperti PHP yang menampilkan pesan error di browser, JavaScript menangani pesan error dengan cara yang berubah-ubah sesuai browser yang digunakan. Tabel diatas mencantumkan cara mengakses pesan kesalahan JavaScript di masing-masing dari lima browser yang paling umum digunakan. Tabel tersebut menyajikan cara akses pesan kesalahan JavaScript di browser yang berbeda.

Tabel 6.1 Mengakses pesan ksalahan JavaScript pada browser yang berbeda

Browser	**Bagaimana cara mengakses pesan kesalahan JavaScript**
Safari Apple	Safari tidak mengaktifkan Konsol Kesalahan secara default, tetapi kita dapat mengaktifkannya dengan memilih Safari→Preferences→Advanced→"Show Develop menu in menu bar." Namun, kita mungkin lebih suka menggunakan modul JavaScript Firebug Lite, yang menurut banyak orang lebih mudah digunakan.
Google Chrome	Klik menu ikon yang terlihat dihalaman dengan sudut berbelok, kemudia pilih Developer→JavaScript Console. kita jga dapat menekan shortcut Ctrl+Shift+J pada dPC atau Command+Shift+J pada Mac
Microsoft Internet Explorer	Pilih Tools→Internet Options→Advanced, kemudian hapus centang pada kotak dialog Disable Script Debugging dan centang kotak "Display a Notification about Every Script Error"
Mozilla Firefox	Pilih Tools→Error Console atau gunakan shortcut Ctrl+Shift+J pada PC, atau command Shift-J pada Mac.
Opera	Pilih Tools→Advanced→Error Console.

Pengguna OS X: meskipun saya telah menunjukkan kepada kita cara membuat Konsol Kesalahan untuk JavaScript, kita mungkin lebih suka menggunakan Google Chrome (untuk Intel OS X 10.5 atau lebih tinggi).

Untuk mencoba Konsol Kesalahan mana pun yang kita gunakan, mari buat skrip dengan kesalahan kecil. Contoh 2 hampir sama dengan Contoh 3 tetapi tanda kutip ganda terakhir telah ditinggalkan di akhir string "Hello World"—kesalahan sintaks yang umum.

Contoh 3 Skrip JavaScript "Hello world" dengan kesalahan

```
<html>
<head><title>Hello World</title></head>
<body>
<script type="text/javascript">
document.write("Hello World)
</script>
</body>
</html>
```

Masukkan contoh dan simpan sebagai test.html, lalu panggil di browser Anda. Seharusnya hanya berhasil menampilkan judul, bukan apa pun di jendela browser utama. Sekarang panggil Konsol Kesalahan di browser Anda, dan kita akan melihat pesan seperti yang ada di Contoh 4. Di sebelah kanan akan ada tautan ke sumber, yang, ketika diklik, menunjukkan garis kesalahan yang disorot (tetapi tidak menunjukkan posisi di mana kesalahan ditemukan).

Contoh 4. Pesan Konsol Kesalahan Mozilla Firefox

```
SyntaxError: unterminated string literal
```

Di Microsoft Internet Explorer, pesan kesalahan akan terlihat seperti Contoh 14-5, dan tidak ada panah yang membantu, tetapi kita diberi garis dan posisi.

Contoh 5. Pesan Konsol Kesalahan Microsoft Internet Explorer

```
Unterminated string constan
```

Google Chrome dan Opera akan memberikan pesan pada Contoh 6. Sekali lagi, kita akan diberikan nomor kesalahan baris tetapi bukan lokasi yang tepat.

Contoh 6. Pesan Konsol Kesalahan Google Chrome/Opera

```
Uncaught SyntaxError: Unexpected token ILLEGAL
```

Dan Apple Safari memberikan pesan di Contoh 7, dengan tautan ke sumber di sebelah kanan yang menyatakan nomor baris kesalahan. kita dapat mengklik tautan untuk menyorot garis, tetapi itu tidak akan menunjukkan di mana kesalahan itu terjadi

Contoh 7. Pesan Konsol Kesalahan Opera

```
SyntaxError: Unexpected EOF
```

Jika kita merasa dukungan ini sedikit mengecewakan, plug-in Firebug untuk Firefox (dan sekarang Chrome juga) sangat populer di kalangan pengembang JavaScript untuk men-debug kode, dan pasti layak untuk dilihat.

6.7 MENGGUNAKAN KOMENTAR

Karena warisan bersama dari bahasa pemrograman C, PHP dan JavaScript memiliki banyak kesamaan, salah satunya adalah berkomentar. Pertama, ada komentar satu baris, seperti ini:

```
// This is a comment
```

Gaya ini menggunakan sepasang karakter garis miring (//) untuk memberi tahu JavaScript bahwa semua yang berikut harus diabaikan. Dan kemudian kita juga memiliki komentar multiline, seperti ini:

```
/* This is a section
of multiline comments
that will not be
interpreted */
```

Di sini kita memulai komentar multiline dengan urutan /* dan mengakhirinya dengan */. Ingatlah bahwa kita tidak dapat menyarangkan komentar multibaris, jadi pastikan kita tidak mengomentari sebagian besar kode yang sudah berisi komentar multibaris.

6.8 TITIK KOMA

Tidak seperti PHP, JavaScript umumnya tidak memerlukan titik koma jika kita hanya memiliki satu pernyataan dalam satu baris. Oleh karena itu, berikut ini valid:

```
x += 10
```

Namun, bila kita ingin menempatkan lebih dari satu pernyataan dalam satu baris, kita harus memisahkannya dengan titik koma, seperti ini:

```
x += 10; y −= 5; z = 0
```

Anda biasanya dapat membiarkan titik koma terakhir mati, karena baris baru mengakhiri pernyataan terakhir.

6.9 VARIABEL

Tidak ada karakter tertentu yang mengidentifikasi variabel dalam JavaScript seperti yang dilakukan tanda dolar di PHP. Sebagai gantinya, variabel menggunakan aturan penamaan berikut:

- Variabel hanya boleh menyertakan huruf a–z, A–Z, 0–9, simbol $, dan garis bawah (_).
- Tidak ada karakter lain, seperti spasi atau tanda baca, yang diperbolehkan dalam nama variabel.
- Karakter pertama dari nama variabel hanya boleh a–z, A–Z, $, atau _ (tanpa angka).

- Nama peka huruf besar-kecil. Hitung, hitung, dan COUNT semuanya adalah variabel yang berbeda.
- Tidak ada batasan yang ditetapkan pada panjang nama variabel.

Dan ya, kita benar, itu adalah $ yang ada di daftar itu. Hal ini diperbolehkan oleh JavaScript dan mungkin karakter pertama dari variabel atau nama fungsi. Meskipun saya tidak menyarankan untuk menyimpan simbol $, itu berarti kita dapat mem-porting banyak kode PHP lebih cepat ke JavaScript dengan cara itu.

Variabel String

Variabel string JavaScript harus diapit dengan tanda kutip tunggal atau ganda, seperti ini:

```
greeting = "Hello there"
warning = 'Be careful'
```

Anda dapat menyertakan kutipan tunggal dalam string yang dikutip ganda atau kutipan ganda dalam string yang dikutip tunggal. Tetapi kita harus menghindari kutipan dari jenis yang sama menggunakan karakter garis miring terbalik, seperti ini:

```
greeting = "\"Hello there\" is a greeting"
warning = '\'Be careful\' is a warning'
```

Untuk membaca dari variabel string, kita dapat menetapkannya ke variabel lain, seperti ini:

```
newstring = oldstring
```

atau kita dapat menggunakannya dalam suatu fungsi, seperti ini:

```
status = "All systems are working"
document.write(status)
```

Variabel Numerik

Membuat variabel numerik semudah menetapkan nilai, seperti contoh berikut:

```
count = 42
temperature = 98.4
```

Seperti string, variabel numerik dapat dibaca dan digunakan dalam ekspresi dan fungsi.

6.10 ARRAY

Array JavaScript juga sangat mirip dengan yang ada di PHP, di mana array dapat berisi data string atau numerik, serta array lainnya. Untuk menetapkan nilai ke array, gunakan sintaks berikut (yang dalam hal ini membuat array string):

```
toys = ['bat', 'ball', 'whistle', 'puzzle', 'doll'
```

Untuk membuat array multidimensi, susun array yang lebih kecil di dalam array yang lebih besar. Jadi, untuk membuat larik dua dimensi yang berisi warna wajah tunggal dari Kubus

Rubik yang diacak (di mana warna merah, hijau, oranye, kuning, biru, dan putih diwakili oleh huruf awal kapital), kita dapat menggunakan kode berikut:

```
face =
[
['R', 'G', 'Y'],
['W', 'R', 'O'],
['Y', 'W', 'G']
]
```

Contoh sebelumnya telah diformat untuk memperjelas apa yang sedang terjadi, tetapi dapat juga ditulis seperti ini:

```
face = [['R', 'G', 'Y'], ['W', 'R', 'O'], ['Y', 'W', 'G']]
```

atau bahkan seperti ini:

```
top = ['R', 'G', 'Y']
mid = ['W', 'R', 'O']
bot = ['Y', 'W', 'G']
face = [top, mid, bot]
```

Untuk mengakses elemen dua ke bawah dan tiga bersama dalam matriks ini, kita akan menggunakan yang berikut (karena elemen array dimulai dari posisi 0)

```
document.write(face[1][2])
```

Pernyataan ini akan menampilkan huruf O untuk oranye

Array argumen

Array argumen adalah anggota dari setiap fungsi. Dengan itu, kita dapat menentukan jumlah variabel yang diteruskan ke suatu fungsi dan apa itu. Ambil contoh fungsi yang disebut displayItems. Contoh 8 menunjukkan salah satu cara penulisannya.

Contoh 8. Mendefinisikan sebuah fungsi

```
<script>
displayItems("Dog", "Cat", "Pony", "Hamster", "Tortoise")
function displayItems(v1, v2, v3, v4, v5)
{
document.write(v1 + "<br>")
document.write(v2 + "<br>")
document.write(v3 + "<br>")
document.write(v4 + "<br>")
document.write(v5 + "<br>")
}
</script>
```

Saat kita memanggil skrip ini di browser Anda, itu akan menampilkan yang berikut:
Dog
Cat
Pony
Hamster
Tortoise
Semua ini baik-baik saja, tetapi bagaimana jika kita ingin meneruskan lebih dari lima item ke fungsi? Juga menggunakan kembali panggilan document.write beberapa kali alih-alih menggunakan loop adalah pemrograman yang sia-sia. Untungnya, array argumen memberi kita fleksibilitas untuk menangani sejumlah argumen yang bervariasi. Contoh 9 menunjukkan bagaimana kita dapat menggunakannya untuk menulis ulang contoh dengan cara yang jauh lebih efisien.

Contoh 9. Memodifikasi fungsi untuk menggunakan array argumen

```
<script>
function displayItems()
{
for (j = 0 ; j < displayItems.arguments.length ; ++j)
document.write(displayItems.arguments[j] + "<br>")
}
</script>
```

Perhatikan penggunaan properti length, yang telah kita temui di bab sebelumnya, dan juga bagaimana array displayItems.arguments direferensikan menggunakan variabel j sebagai offset ke dalamnya. Saya juga memilih untuk menjaga agar fungsi tetap pendek dan manis dengan tidak mengelilingi konten for loop dalam kurung kurawal, karena hanya berisi satu pernyataan. Dengan menggunakan teknik ini, kita sekarang memiliki fungsi yang dapat mengambil sebanyak (atau sesedikit) argumen yang kita suka dan bertindak pada setiap argumen sesuai keinginan Anda.

Array JavaScript

Penanganan array dalam JavaScript sangat mirip dengan PHP, meskipun sintaksnya sedikit berbeda. Namun demikian, mengingat semua yang telah kita pelajari tentang array, bagian ini seharusnya relatif mudah bagi Anda.

Array Numerik

Untuk membuat array baru, gunakan sintaks berikut:

```
arrayname = new Array()
```

Atau kita dapat menggunakan formulir singkatan, sebagai berikut:

```
arrayname
```

Menetapkan nilai elemen

Di PHP, kita bisa menambahkan elemen baru ke array hanya dengan menetapkannya tanpa menentukan elemen offset, seperti ini:

```
$arrayname[] = "Element 1";
$arrayname[] = "Element 2";
```

Tetapi dalam JavaScript kita menggunakan metode Push untuk mencapai hal yang sama, seperti ini:

```
arrayname.push("Element 1")
arrayname.push("Element 2")
```

Ini memungkinkan kita untuk terus menambahkan item ke array tanpa harus melacak jumlah item. Saat kita perlu mengetahui berapa banyak elemen dalam array, kita dapat menggunakan properti length, seperti ini:

```
document.write(arrayname.length)
```

Atau, jika kita ingin melacak sendiri lokasi elemen dan menempatkannya di lokasi tertentu, kita dapat menggunakan sintaks seperti ini:

```
arrayname[0] = "Element 1"
arrayname[1] = "Element 2"
```

Contoh 10 menunjukkan skrip sederhana yang membuat larik, memuatnya dengan beberapa nilai, lalu menampilkannya.

Contoh 10 Membuat, membangun, dan mencetak array

```
<script>
numbers = []
numbers.push("One")
numbers.push("Two")
numbers.push("Three")
for (j = 0 ; j < numbers.length ; ++j)
document.write("Element " + j + " = " + numbers[j] + "<br>")
</script>
```

Output dari skrip ini adalah:

```
Element 0 = One
Element 1 = Two
Element 2 = Three
```

Tugas menggunakan kata kunci array

Anda juga dapat membuat array bersama dengan beberapa elemen awal menggunakan kata kunci Array, seperti ini:

```
numbers = Array("One", "Two", "Three")
```

Tidak ada yang menghentikan kita untuk menambahkan lebih banyak elemen sesudahnya. Jadi sekarang kita memiliki beberapa cara untuk menambahkan item ke array, dan satu cara

untuk mereferensikannya, tetapi JavaScript menawarkan lebih banyak lagi, yang akan segera saya bahas. Tapi pertama-tama kita akan melihat tipe array yang lain.

Array Asosiatif

Array asosiatif adalah array di mana elemen-elemennya direferensikan dengan nama daripada dengan offset numerik. Untuk membuat array asosiatif, tentukan blok elemen di dalam kurung kurawal. Untuk setiap elemen, tempatkan kunci di sebelah kiri dan konten di sebelah kanan titik dua (:). Contoh 11 menunjukkan bagaimana kita dapat membuat larik asosiatif untuk menampung konten bagian "bola" dari pengecer peralatan olahraga online.

Contoh 11. Membuat dan menampilkan array asosiatif.

```
<script>
balls = {"golf": "Golf balls, 6",
"tennis": "Tennis balls, 3",
"soccer": "Soccer ball, 1",
"ping": "Ping Pong balls, 1 doz"}
for (ball in balls)
document.write(ball + " = " + balls[ball] + "<br>")
</script>
```

Untuk memverifikasi bahwa array telah dibuat dan diisi dengan benar, saya telah menggunakan for loop jenis lain menggunakan kata kunci in. Ini membuat variabel baru untuk digunakan hanya di dalam larik (bola, dalam contoh ini) dan mengulangi semua elemen larik di sebelah kanan kata kunci in (bola, dalam contoh ini). Loop bekerja pada setiap elemen bola, menempatkan nilai kunci ke dalam bola. Dengan menggunakan nilai kunci yang disimpan dalam bola ini, kita juga bisa mendapatkan nilai elemen bola saat ini. Hasil pemanggilan script contoh di browser adalah sebagai berikut:

```
golf = Golf balls, 6
tennis = Tennis balls, 3
soccer = Soccer ball, 1
ping = Ping Pong balls, 1 doz
```

Untuk mendapatkan elemen tertentu dari array asosiatif, kita dapat menentukan kunci secara eksplisit, dengan cara berikut (dalam hal ini, mengeluarkan nilai Soccer ball, 1):

```
document.write(balls['soccer'])
```

Array Multidimensi

Untuk membuat array multidimensi dalam JavaScript, cukup tempatkan array di dalam array lain. Misalnya, untuk membuat larik untuk menampung detail kotak-kotak dua dimensi (8x8 bujur sangkar), kita dapat menggunakan kode dalam Contoh 12.

Contoh 12. Membuat array numerik multidimensi

```
<script>
checkerboard = Array(
Array(' ', 'o', ' ', 'o', ' ', 'o', ' ', 'o'),
Array('o', ' ', 'o', ' ', 'o', ' ', 'o', ' '),
Array(' ', 'o', ' ', 'o', ' ', 'o', ' ', 'o'),
```

```
Array(' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '),
Array(' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '),
Array('O', ' ', 'O', ' ', 'O', ' ', 'O', ' '),
Array(' ', 'O', ' ', 'O', ' ', 'O', ' ', 'O'),
Array('O', ' ', 'O', ' ', 'O', ' ', 'O', ' '))
document.write("<pre>")
for (j = 0 ; j < 8 ; ++j)
{
for (k = 0 ; k < 8 ; ++k)
document.write(checkerboard[j][k] + " ")
document.write("<br>")
}
document.write("</pre>")
</script>
```

Dalam contoh ini, huruf kecil mewakili potongan hitam, dan huruf besar putih. Sepasang loop for bersarang berjalan melalui array dan menampilkan isinya. Loop luar berisi dua pernyataan, jadi kurung kurawal mengapitnya. Loop dalam kemudian memproses setiap kotak dalam satu baris, mengeluarkan karakter di lokasi [j] [k], diikuti dengan spasi (untuk mengkuadratkan hasil cetak). Loop ini berisi satu pernyataan, jadi kurung kurawal tidak diperlukan untuk mengapitnya. Tag <pre> dan </pre> memastikan bahwa output ditampilkan dengan benar, seperti ini:

```
o       o       o       o
o       o       o       o
  o       o       o       o

o       o       o       o
  o       o       o       o
o       o       o       o
```

Anda juga dapat langsung mengakses elemen apa pun dalam array ini menggunakan tanda kurung siku, sebagai berikut:

```
document.write(checkerboard[7][2])
```

Pernyataan ini menampilkan huruf besar O, elemen kedelapan ke bawah dan elemen ketiga sepanjang—ingat bahwa indeks array dimulai dari 0, bukan 1.

Mengembalikan Array

Dalam Contoh 12, fungsi hanya mengembalikan satu parameter, tetapi bagaimana jika kita perlu mengembalikan beberapa parameter? kita dapat melakukan ini dengan mengembalikan sebuah array, seperti pada Contoh 13.

Contoh 13. Mengembalikan array nilai

```
<script>
words = fixNames("the", "DALLAS", "CowBoys")
for (j = 0 ; j < words.length ; ++j)
document.write(words[j] + "<br>")
```

```
function fixNames()
{
var s = new Array()
for (j = 0 ; j < fixNames.arguments.length ; ++j)
s[j] = fixNames.arguments[j].charAt(0).toUpperCase() +
fixNames.arguments[j].substr(1).toLowerCase()
return s
}
</script>
```

Di sini kata-kata variabel secara otomatis didefinisikan sebagai array dan diisi dengan hasil panggilan yang dikembalikan ke fungsi fixNames. Kemudian for loop iterasi melalui array dan menampilkan setiap anggota. Adapun fungsi fixNames, hampir identik dengan Contoh 13, kecuali bahwa variabel s sekarang menjadi array, dan setelah setiap kata diproses, disimpan sebagai elemen array ini, yang dikembalikan oleh pernyataan return. Fungsi ini memungkinkan ekstraksi parameter individu dari nilai yang dikembalikan, seperti berikut ini (output dari yang hanya The Cowboys):

```
words = fixNames("the", "DALLAS", "CowBoys")
document.write(words[0] + " " + words[2])
```

6.11 MENGEMBALIKAN NILAI

Fungsi tidak digunakan hanya untuk menampilkan sesuatu. Bahkan, mereka sebagian besar digunakan untuk melakukan perhitungan atau manipulasi data dan kemudian mengembalikan hasil. Fungsi fixNames dalam Contoh 14 menggunakan array argumen (dibahas di bab sebelumnya) untuk mengambil serangkaian string yang diteruskan ke sana dan mengembalikannya sebagai string tunggal. "Perbaikan" yang dilakukannya adalah mengonversi setiap karakter dalam argumen menjadi huruf kecil kecuali untuk karakter pertama dari setiap argumen, yang diatur ke huruf kapital.

Contoh 14 Membersihkan n . penuh

```
<script>
document.write(fixNames("the", "DALLAS", "CowBoys"))
function fixNames()
{
var s = ""
for (j = 0 ; j < fixNames.arguments.length ; ++j)
s += fixNames.arguments[j].charAt(0).toUpperCase() +
fixNames.arguments[j].substr(1).toLowerCase() + " "
return s.substr(0, s.length-1)
}
</script>
```

Ketika dipanggil dengan parameter the, DALLAS, dan CowBoys, misalnya, fungsi mengembalikan string The Dallas Cowboys. Mari kita telusuri fungsinya. Fungsi pertama menginisialisasi variabel sementara (dan lokal) s ke string kosong. Kemudian for loop

mengiterasi setiap parameter yang diteruskan, mengisolasi karakter pertama parameter menggunakan metode charAt dan mengonversinya menjadi huruf besar dengan metode toUpperCase. Berbagai metode yang ditunjukkan dalam contoh ini semuanya dibangun ke dalam JavaScript dan tersedia secara default. Kemudian metode substr digunakan untuk mengambil sisa setiap string, yang diubah menjadi huruf kecil menggunakan metode toLowerCase. Versi yang lebih lengkap dari metode substr di sini akan menentukan berapa banyak karakter yang merupakan bagian dari substring sebagai argumen kedua:

substr(1, (arguments[j].length) - 1)

Dengan kata lain, metode substr ini mengatakan, "Mulai dengan karakter di posisi 1 (karakter kedua) dan kembalikan sisa string (panjang dikurangi satu)." Namun, sebagai sentuhan yang bagus, metode substr mengasumsikan bahwa kita menginginkan sisanya.

6.12 OPERATOR

JavaScript menawarkan banyak operator hebat yang berkisar dari operator aritmatika, string, dan logika hingga penugasan, perbandingan, dan banyak lagi.

Tabel 6.2 Jenis operator JavaScript

Assignment	Penetapan Nilai	a=b+23
Bitwise	Memanipulasi bit dengan byte	12^9
Perbandingan	MEmbandingkan dua nilai	a < b
Increment/Decrement	Menambahkan atau mengurangi satu	a ++
Logika	Boolean	a && b
String	Rangkaian	a + ‘string’

Setiap operator mengambil jumlah operan yang berbeda:

- Operator unary, seperti incrementing (a++) atau negasi (-a), mengambil satu operan.
- Operator biner, yang mewakili sebagian besar operator JavaScript—termasuk penambahan, pengurangan, perkalian, dan pembagian—mengambil dua operan.
- Satu operator ternary, yang berbentuk ? x : y. Ini adalah pernyataan if satu baris singkat yang memilih antara dua ekspresi tergantung pada ekspresi ketiga.

Prioritas Operator

Seperti halnya PHP, JavaScript menggunakan prioritas operator, di mana beberapa operator dalam ekspresi dianggap lebih penting daripada yang lain dan oleh karena itu dievaluasi terlebih dahulu. Tabel 6.3mencantumkan operator JavaScript dan prioritasnya.

Tabel 6.3 Prioritas operator JavaScript (tinggi ke rendah)

Operator	**Jenis**
() [] .	Parentheses, call dan member
++ ––	Increment/decrement
+ – ~ !	Unary, bitwise, dan logika
* / %	Aritmetika
+ –	String dan aritmetika

<< >> >>>	Bitwise
< > <= >=	Perbandingan
== != === !==	Perbandingan
& ^ |	Botwise
&&	Logika
||	Logika
? :	Ternary
= += –= *= /= %=	Assignment
<<= >>= >>>= &= ^= |=	Assignment
,	Separator

Keterkaitan

Sebagian besar operator JavaScript diproses secara berurutan dari kiri ke kanan dalam sebuah persamaan. Tetapi beberapa operator memerlukan pemrosesan dari kanan ke kiri sebagai gantinya. Arah pemrosesan disebut asosiatifitas operator. Keterkaitan ini menjadi penting dalam kasus di mana kita tidak secara eksplisit memaksakan prioritas. Misalnya, lihat operator penugasan berikut, di mana tiga variabel semuanya disetel ke nilai 0:

```
level = score = time = 0
```

Penetapan ganda ini hanya dimungkinkan karena bagian paling kanan dari ekspresi dievaluasi terlebih dahulu dan kemudian pemrosesan berlanjut ke arah kanan-ke-kiri. Tabel 6.4 mencantumkan operator dan asosiasinya.

Operator dalam JavaScript, seperti dalam PHP, dapat melibatkan matematika, perubahan string, dan perbandingan dan operasi logis (dan, atau, dll.). Operator matematika JavaScript sangat mirip dengan aritmatika biasa; misalnya, pernyataan berikut menghasilkan 15:

```
document.write(13 + 2)
```

Bagian berikut mengajarkan kita tentang berbagai operator.

Operator Aritmatika

Operator aritmatika digunakan untuk melakukan matematika. kita dapat menggunakannya untuk empat operasi utama (penjumlahan, pengurangan, perkalian, dan pembagian) serta untuk menemukan modulus (sisa setelah pembagian) dan untuk menambah atau mengurangi nilai.

Tabel 6.4 Operator aritmatika

Operator	***Deskripsi***	***Contoh***
+	Penjumlahan	J+2
-	Pengurangan	j-22
*	Perkalian	J*7
/	Pembagian	j/3.13
%	Modulus	J%6
++	Increment	++j
--	Decrement	--j

Operator Penugasan

Operator penugasan digunakan untuk memberikan nilai ke variabel. Mereka mulai dengan yang sangat sederhana =, dan berlanjut ke +=, =, dan seterusnya. Operator += menambahkan nilai di sisi kanan ke variabel di sebelah kiri, alih-alih mengganti nilai di sebelah kiri sepenuhnya. Jadi, jika hitungan dimulai dengan nilai 6, pernyataan:

count += 1

set count hingga 7, seperti pernyataan penugasan yang lebih familiar:

count = count + 1

Tabel 6.5mencantumkan berbagai operator penugasan yang tersedia.

Tabel 6.5 Operator penugasan

Operator	***Contoh***	***Ekuivalen ke***
=	j=99	j=99
+=	j+=2	j=j+2
+=	j+='string'	j=j+'string'
-+	j-=12	j=j-12
*=	j*=2	j=j*2
/=	j/=6	j=j/6
%=	j%=7	j=j%7

Operator Perbandingan

Operator perbandingan umumnya digunakan di dalam konstruk seperti pernyataan if di mana kita perlu membandingkan dua item. Misalnya, kita mungkin ingin mengetahui apakah variabel yang telah kita tingkatkan telah mencapai nilai tertentu, atau apakah variabel lain kurang dari nilai yang ditetapkan, dan seterusnya.

Tabel 6.6 perbandingan operator

Operator	***Deskripsi***	***Contoh***
==	Sama dengan	**j == 42**
!=	Tidak sama dnegan	**j != 17**
>	Lebeih besar dari	**j > 0**
<	Kurang dari	**j < 100**
>=	Lebih besar atau sama dengan	**j >= 23**
<=	Lebih kecil atau sama dengan	**j <= 13**
===	Sama dengan (dan pada jenis yang sama)	**j ===56**
!==	Tidak sama dengan (dan pada jenis yang sama)	**j !== '1'**

Operator Logika

Tidak seperti PHP, operator logika JavaScript tidak menyertakan dan dan atau setara ke && dan ||, dan tidak ada operator xor

Tabel 6.7 Operator Logika

Operator	Deskripsi	Contoh
&&	Dan	J==1 **&&** ==2
| |	Atau	J < 100 **| |** >0
!	Tidak	**! (j==k)**

Variabel Incrementing dan Decrementing

Bentuk post-and pre-incrementing dan decrementing berikut yang kita pelajari untuk digunakan di PHP juga didukung oleh JavaScript:

++x
−−y
x += 22
y −= 3

Penggabungan String

JavaScript menangani penggabungan string sedikit berbeda dari PHP. Alih-alih. (titik) operator, menggunakan tanda plus (+), seperti ini:

document.write("You have " + messages + " messages.")

Dengan asumsi bahwa pesan variabel diatur ke nilai 3, output dari baris kode ini adalah:

You have 3 messages.

Sama seperti kita dapat menambahkan nilai ke variabel numerik dengan operator +=, kita juga dapat menambahkan satu string ke string lain dengan cara yang sama:

name = "James"
name += " Dean"

Escaping character

Karakter escape, yang kita lihat digunakan untuk menyisipkan tanda kutip dalam string, juga dapat menyisipkan berbagai karakter khusus seperti tab, baris baru, dan carriage return. Berikut adalah contoh penggunaan tab untuk meletakkan heading; itu disertakan di sini hanya untuk mengilustrasikan pelarian, karena di halaman web, ada cara yang lebih baik untuk melakukan layout:

heading = "Name\tAge\tLocation"

Tabel 6.8 merinci karakter pelarian yang tersedia.

Tabel 6.8 Karakter pelarian JavaScript

Karakter	Maksud
\b	Menghapus
\f	Feed formulir
\n	Baris baru
\r	Carriage return
\t	Tab
\’	Petik satu (apostrophe)
\”	Petik dua
\\	Garis miirng terbalik
\XXX	Angka oktal antara 000 dan 377 yang mewakili karakter Latin-1 setara (seperti \251 untuk simbol ©)
\xXX	Angka heksadesimal antara 00 dan FF yang mewakili karakter Latin-1 setara (seperti \xA9 untuk simbol ©)
\uXXXX	Angka heksadesimal antara 0000 dan FFFF yang mewakili Unicode setara karakter (seperti \u00A9 untuk simbol ©)

Pengetikan Variabel

Seperti PHP, JavaScript adalah bahasa yang diketik dengan sangat longgar; jenis variabel ditentukan hanya ketika nilai ditetapkan dan dapat berubah saat variabel muncul dalam konteks yang berbeda. Biasanya, kita tidak perlu khawatir tentang jenisnya; JavaScript mencari tahu apa yang kita inginkan dan melakukannya. Lihatlah Contoh 15, di mana:

1. Variabel n diberi nilai string 838102050, baris berikutnya mencetak nilainya, dan operator typeof digunakan untuk mencari jenisnya.
2. n diberikan nilai yang dikembalikan ketika angka 12345 dan 67890 dikalikan bersama. Nilai ini juga 838102050, tetapi ini adalah angka, bukan string. Jenis variabel kemudian dicari dan ditampilkan.
3. Beberapa teks ditambahkan ke angka n dan hasilnya ditampilkan.

Contoh 15. Menyetel tipe variabel berdasarkan penugasan

```
<script>
n = '838102050' // Set 'n' to a string
document.write('n = ' + n + ', and is a ' + typeof n + '<br>')
n = 12345 * 67890; // Set 'n' to a number
document.write('n = ' + n + ', and is a ' + typeof n + '<br>')
n += ' plus some text' // Change 'n' from a number to a string
document.write('n = ' + n + ', and is a ' + typeof n + '<br>')
</script>
```

Output dari skrip ini terlihat seperti:

```
n = 838102050, and is a string
n = 838102050, and is a number
n = 838102050 plus some text, and is a string
```

Jika ada keraguan tentang tipe variabel, atau kita perlu memastikan bahwa variabel memiliki tipe tertentu, kita dapat memaksanya ke tipe tersebut menggunakan pernyataan seperti berikut (yang masing-masing mengubah string menjadi angka dan angka menjadi string):

```
n = "123"
n *= 1 // Convert 'n' into a number
n = 123
n += "" // Convert 'n' into a string
```

Atau, tentu saja, kita selalu dapat mencari tipe variabel menggunakan operator typeof.

Tabel 6.9 aktivitas operator

Operator	Deskripsi	Asosiativitas
++ ––	Increment dan Decrement	Tidak ada
new	Membuat objek baru	Benar
+ – ~ !	Unary dan Bitwise	Benar
?	Ternary	Benar
= *= /= %= += –=	Assignment	Benar
<<= >>= >>>= &= ^= |=	Assignment	Benar
,	Separator	Kiri
+ – * / %	Aritmetika	Kiri
<< >> >>>	Bitwise	Kiri
< <= > >= == != === !==	Aritmetika	Kiri

Operator Relasional

Operator relasional menguji dua operan dan mengembalikan hasil Boolean baik benar atau salah. Ada tiga jenis operator relasional: kesetaraan, perbandingan, dan logika.

Operator kesetaraan

Operator kesetaraan adalah == (yang tidak boleh disamakan dengan operator penugasan =). Dalam Contoh 16, pernyataan pertama memberikan nilai dan yang kedua mengujinya untuk kesetaraan. Seperti berdiri, tidak ada yang akan dicetak, karena bulan diberi nilai string Juli, dan karena itu cek untuk itu memiliki nilai Oktober akan gagal.

Contoh 16 Menetapkan nilai dan menguji kesetaraan.

```
<script>
month = "July"
if (month == "October") document.write("It's the Fall")
</script>
```

Jika dua operan dari ekspresi kesetaraan memiliki tipe yang berbeda, JavaScript akan mengonversinya ke tipe apa pun yang paling masuk akal. Misalnya, string apa pun yang seluruhnya terdiri dari angka akan dikonversi menjadi angka setiap kali dibandingkan dengan angka. Dalam Contoh 17, a dan b adalah dua nilai yang berbeda (satu adalah angka dan yang lainnya adalah string), dan karena itu kita biasanya tidak mengharapkan pernyataan if untuk menghasilkan hasil.

Contoh 17 Operator kesetaraan dan identitas

```
<script>
a = 3.1415927
b = "3.1415927"
if (a == b) document.write("1")
if (a === b) document.write("2")
</script>
```

Namun, jika kita menjalankan contoh, kita akan melihat bahwa itu menghasilkan angka 1, yang berarti bahwa pernyataan if pertama dievaluasi menjadi benar. Ini karena nilai string b pertama kali diubah sementara menjadi angka, dan oleh karena itu kedua bagian persamaan memiliki nilai numerik 3,1415927. Sebaliknya, pernyataan if kedua menggunakan operator identitas, tiga tanda sama dengan berturut-turut, yang mencegah JavaScript mengonversi tipe secara otomatis. Ini berarti bahwa a dan b karena itu ditemukan berbeda, jadi tidak ada output. Seperti memaksa operator didahulukan, setiap kali kita ragu tentang bagaimana JavaScript akan mengonversi jenis operan, kita dapat menggunakan operator identitas untuk menonaktifkan perilaku ini.

Operator perbandingan

Dengan menggunakan operator perbandingan, kita dapat menguji lebih dari sekadar kesetaraan dan ketidaksetaraan. JavaScript juga memberi kita > (lebih besar dari), < (lebih kecil dari), >= (lebih besar dari atau sama dengan), dan <= (kurang dari atau sama dengan) untuk dimainkan.

Contoh 18 menunjukkan operator ini digunakan.

Contoh 18 Empat operator pembanding

```
<script>
a = 7; b = 11
if (a > b) document.write("a is greater than b<br>")
if (a < b) document.write("a is less than b<br>")
if (a >= b) document.write("a is greater than or equal to b<br>")
if (a <= b) document.write("a is less than or equal to b<br>")
</script>
```

Dalam contoh ini, di mana a adalah 7 dan b adalah 11, berikut adalah outputnya (karena 7 lebih kecil dari 11, dan juga lebih kecil dari atau sama dengan 11):

a is less than b
a is less than or equal to b

Operator logika

Operator logika menghasilkan hasil benar atau salah, dan juga dikenal sebagai operator Boolean. Ada tiga di antaranya dalam JavaScript (lihat Tabel 6.10).

Tabel 6.10 Operator logika JavaScript

Operator Logika	Deskripsi
&& (and)	true jika operan true

|| (or)	true jika operan lainnya true
! (not)	true jika operan adalah false, atau false jika operan adalah true

Anda dapat melihat bagaimana ini dapat digunakan dalam Contoh 19, yang menghasilkan 0, 1, dan true.

Contoh 19 Operator logika yang digunakan

```
<script>
a = 1; b = 0
document.write((a && b) + "<br>")
document.write((a || b) + "<br>")
document.write(( !b ) + "<br>")
</script>
```

Pernyataan && mengharuskan kedua operan bernilai true jika akan mengembalikan nilai true, yaitu || pernyataan akan benar jika salah satu nilainya benar, dan pernyataan ketiga melakukan NOT pada nilai b, mengubahnya dari 0 menjadi nilai benar. || operator dapat menyebabkan masalah yang tidak disengaja, karena operan kedua tidak akan dievaluasi jika yang pertama dievaluasi sebagai benar. Pada Contoh 20, fungsi getnext tidak akan pernah dipanggil jika selesai memiliki nilai 1.

Contoh 20 Pernyataan menggunakan || operator

```
<script>
if (finished == 1 || getnext() == 1) done = 1
</script>
```

Jika kita membutuhkan getnext untuk dipanggil pada setiap pernyataan if, kita harus menulis ulang kode seperti yang ditunjukkan pada Contoh 21.

Contoh 21 Pernyataan if...or dimodifikasi untuk memastikan pemanggilan get.

```
<script>
gn = getnext()
if (finished == 1 OR gn == 1) done = 1;
</script>
```

Dalam hal ini, kode dalam fungsi getnext akan dieksekusi dan nilai kembaliannya disimpan di gn sebelum pernyataan if. Tabel 6.11 menunjukkan semua kemungkinan variasi penggunaan operator logika. kita juga harus mencatat bahwa !true sama dengan salah dan !false sama dengan benar.

Tabel 6.11 Semua kemungkinan ekspresi logis

Input		**Operator dan hasil**	
a	b	&&	||
true	true	true	true
true	false	false	true
false	true	false	true
flase	false	false	false

BAB 7
MEMBANGUN PROGRAM JAVASCRIPT

Membangun Bab sebelumnya menunjukkan cara menambahkan tag JavaScript ke halaman, dan bab ini berkonsentrasi pada apa yang dapat kita lakukan setelah itu. Kunci untuk memahami bahasa pemrograman adalah mempelajari sintaksnya. Sama seperti ketika kita belajar bahasa asing dan kita perlu mempelajari kata-kata dan tata bahasa bahasa tersebut, sintaks bahasa pemrograman hanya itu: kata-kata dan tata bahasa yang membentuk bahasa tersebut. JavaScript dilihat melalui browser web dan diprogram dalam text editor, seperti HTML dan CSS. Contoh yang kita lihat di seluruh bab ini dapat diprogram seperti contoh lain yang kita lihat di seluruh buku. Dalam bab ini, kita akan melihat bagaimana membangun program JavaScript, termasuk beberapa seluk beluk pemrograman dalam JavaScript.

7.1 MEMULAI DENGAN PEMROGRAMAN JAVASCRIPT

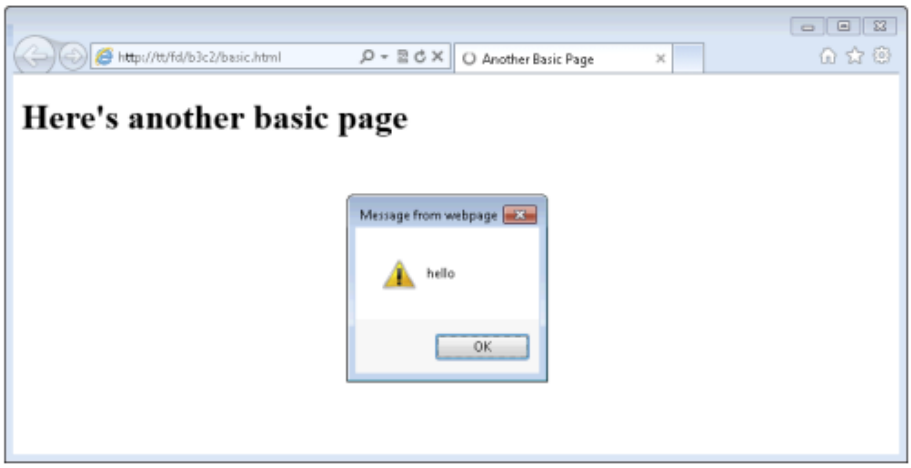
Karena ini mungkin pertama kalinya kita mengenal pemrograman dalam bentuk apa pun, bagian ini dimulai dengan beberapa informasi dasar untuk mempercepat Anda.

Mengirim peringatan ke layar

Anda dapat menggunakan JavaScript untuk mengirim peringatan ke layar. Meskipun ini tidak banyak digunakan di halaman web, kita menggunakannya untuk memecahkan masalah program Anda, dan ini juga merupakan cara cepat untuk melihat JavaScript beraksi. Mulailah dengan membuka text editor dengan dokumen baru atau kosong. Di text editor, tempatkan HTML dan JavaScrip berikut:

```
<!doctype html>
<html>
<head>
<title>Another Basic Page</title>
</head>
<body>
<h1>Here's another basic page</h1>
<script type="text/javascript">
 alert("hello");
</script>
</body>
</html>
```

Simpan file sebagai basic.html di root dokumen. Lihat halaman dengan membuka browser web dan navigasi ke http://localhost/basic.html. Kita akan melihat halaman ini seperti gambar berikut.



Gambar 7.1 Memuat halaman dengan sebuah peringatan. Klik **OK** untuk mengabaikan peringatan.

Melihat program itu, yang terdapat dalam satu baris antara tag pembuka dan penutup <script>, hanya ada kata alert dengan kata "hello" yang diapit tanda kutip dan tanda kurung. Kata alert sebenarnya adalah fungsi bawaan. Baris JavaScript diakhiri dengan titik koma. Itu konsep penting dan harus menjadi pelajaran utama dari latihan ini: kita mengakhiri hampir setiap baris JavaScript dengan titik koma.

Menambahkan komentar

Sama seperti fungsi peringatan yang berguna, demikian juga komentar, yang seperti catatan tempel untuk kode Anda. Komentar dapat digunakan agar kita mengingat apa yang seharusnya dilakukan oleh bagian kode tertentu atau dapat digunakan untuk melewati bagian kode yang tidak ingin kita jalankan. Bentuk komentar yang umum dimulai dengan dua garis miring, seperti ini:

```
// This is a comment
```

Anda melihat bentuk komentar itu di bab sebelumnya. Kata-kata yang mengikuti dua garis miring tidak akan dibaca oleh browser web, tetapi dapat dibaca oleh orang-orang yang melihat JavaScript Anda, jadi jaga kebersihannya!

Jenis komentar lainnya dimulai dengan garis miring dan tanda bintang, seperti ini /*, dan ditutup dengan tanda bintang dan garis miring, seperti ini */. Dengan gaya komentar seperti itu, semua yang ada di antara komentar pembuka dan penutup tidak terbaca.

```
/*
This won't be read, it's in a comment
*/
```

Menyimpan data untuk nanti dalam variabel

Saat kita bekerja dengan bahasa pemrograman seperti JavaScript, kita sering kali perlu menyimpan data untuk digunakan nanti. kita akan mendapatkan nilai, seperti input dari formulir yang diisi pengguna, dan kemudian kita harus menggunakannya nanti. Untuk menyimpan data, kita menggunakan variabel, yang melacak data yang kita perintahkan untuk disimpan selama masa pakai program Anda. Itu konsep penting: Isi variabel hanya hidup

selama program Anda. Tidak seperti data dalam database, tidak ada persistensi untuk data variabel. Variabel didefinisikan dalam JavaScript dengan kata kunci var, kependekan dari variabel.

var myVariable;

JavaScript peka huruf besar-kecil. kita melihat dalam contoh bahwa kata kunci var adalah huruf kecil dan variabel myVariable menggunakan huruf besar campuran. Penting untuk menggunakan kasus yang sama untuk nama variabel dan selalu perhatikan bahwa, misalnya, MYVARIABLE tidak sama dengan myVariable atau myvariable. Selalu ikuti sensitivitas huruf besar-kecil untuk JavaScript, dan kita tidak akan pernah memiliki masalah dengan itu! Saat kita membuat variabel, biasanya memberikan beberapa data untuk disimpan. kita melakukan ini dengan tanda sama dengan:

var myVariable = 4;

Sedikit kode tersebut menyetel variabel bernama myVariable sama dengan angka 4. Jika kita tidak menyetel variabel dengan benar saat kita membuatnya, seperti pada contoh itu, kita dapat menyetelnya kapan saja hanya dengan menyetelnya sama dengan nilai yang kita ingin. Berikut ini contohnya:

```
var myVariable;
myVariable = 4;
```

Anda dapat memutar variabel dengan memodifikasi JavaScript yang kita buat di latihan sebelumnya agar terlihat seperti daftara berikut ini.

Contoh 1 Daftar Mencoba Variabel

```
<!doctype html>
<html>
<head>
<title>JavaScript Chapter 2</title>
</head>
<body>
<h1>Here's another basic page</h1>
<script type="text/javascript">
 var myVariable = 4;
 alert(myVariable);
</script>
</body>
</html>
```

Jika kita melihat kode itu di browser, kita akan melihat peringatan seperti yang ditunjukkan pada gambar dibawah ini.

Gambar 7.2 Menampilkan isi variabel.

Variabel JavaScript dapat menampung string, yang pada dasarnya adalah kata-kata yang diapit tanda kutip, atau angka, seperti yang kita lihat dalam contoh. Variabel perlu diberi nama dengan cara tertentu. Variabel harus dimulai dengan huruf dan tidak boleh dimulai dengan angka. Meskipun karakter khusus tertentu baik-baik saja, secara umum variabel hanya boleh berisi huruf dan angka. Nama variabel harus deskriptif tentang apa yang dikandungnya atau apa yang dilakukannya.

Pernyataan with

Pernyataan with bukanlah yang kita lihat di bab-bab sebelumnya tentang PHP, karena itu eksklusif untuk JavaScript. Dengan itu (jika kita mengerti maksud saya), kita dapat menyederhanakan beberapa jenis pernyataan JavaScript dengan mengurangi banyak referensi ke suatu objek menjadi hanya satu referensi. Referensi ke properti dan metode dalam blok with diasumsikan berlaku untuk objek itu. Misalnya, ambil kode dalam Contoh 2, di mana fungsi document.write tidak pernah mereferensikan string variabel berdasarkan nama.

Contoh 2 Menggunakan pernyataan with

```
<script>
string = "The quick brown fox jumps over the lazy dog"
with (string)
{
document.write("The string is " + length + " characters<br>")
document.write("In uppercase it's: " + toUpperCase())
}
</script>
```

Meskipun string tidak pernah direferensikan secara langsung oleh document.write, kode ini masih berhasil menampilkan yang berikut:

```
The string is 43 characters
In uppercase it's: THE QUICK BROWN FOX JUMPS OVER THE LAZY D
```

Beginilah cara kerja kode: penerjemah JavaScript mengenali bahwa properti length dan metode toUpperCase() harus diterapkan ke beberapa objek. Karena mereka berdiri sendiri,

interpreter menganggap mereka berlaku untuk objek string yang kita tentukan dalam pernyataan with.

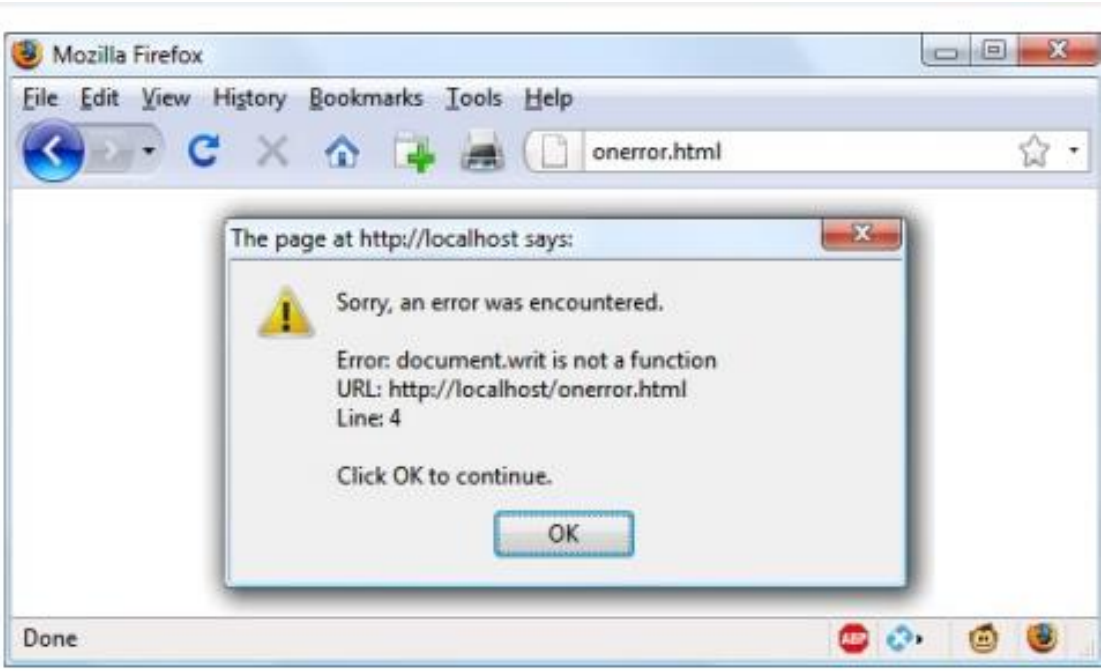
Menggunakan onerror

Ada lebih banyak konstruksi yang tidak tersedia di PHP. Menggunakan event onerror, atau kombinasi dari kata kunci try and catch, kita dapat menangkap kesalahan JavaScript dan menanganinya sendiri. Peristiwa adalah tindakan yang dapat dideteksi oleh JavaScript. Setiap elemen di halaman web memiliki peristiwa tertentu yang dapat memicu fungsi JavaScript. Misalnya, acara onclick dari elemen tombol dapat diatur untuk memanggil fungsi dan membuatnya berjalan setiap kali pengguna mengklik tombol. Contoh 3 mengilustrasikan bagaimana menggunakan event onerror.

Contoh 3 Sebuah skrip yang menggunakan acara onerror

```
<script>
onerror = errorHandler
document.writ("Welcome to this website") // Deliberate error
function errorHandler(message, url, line)
{
out = "Sorry, an error was encountered.\n\n";
out += "Error: " + message + "\n";
out += "URL: " + url + "\n";
out += "Line: " + line + "\n\n";
out += "Click OK to continue.\n\n";
alert(out);
return true;
}
</script>
```

Baris pertama skrip ini memberi tahu peristiwa kesalahan untuk menggunakan fungsi errorHandler baru mulai sekarang dan seterusnya. Fungsi ini membutuhkan tiga parameter—pesan, url, dan nomor baris—jadi mudah untuk menampilkan semua ini dalam pop up peringatan. Kemudian, untuk menguji fungsi baru, kami sengaja menempatkan kesalahan sintaksis dalam kode dengan panggilan ke document.writ alih-alih document.write (e terakhir tidak ada). Gambar 7.3 menunjukkan hasil menjalankan skrip ini di browser. Menggunakan onerror dengan cara ini juga bisa sangat berguna selama proses debugging.



Gambar 7.3 Menggunakan acara onerror dengan metode peringatan muncul

Menggunakan try…catch

Kata kunci try and catch lebih standar dan lebih fleksibel daripada teknik onerror yang ditunjukkan pada bagian sebelumnya. Kata kunci ini memungkinkan kita menjebak kesalahan untuk bagian kode yang dipilih, bukan semua skrip dalam dokumen. Namun, mereka tidak menangkap kesalahan sintaks, yang kita butuhkan untuk kesalahan. Konstruksi try ... catch didukung oleh semua browser utama dan berguna saat kita ingin mengetahui kondisi tertentu yang kita ketahui dapat terjadi di bagian tertentu dari kode Anda. Misalnya, di Bab 12 kita akan menjelajahi teknik Ajax yang menggunakan objek XMLHttpRequest. Sayangnya, ini tidak tersedia di browser Internet Explorer (meskipun ada di semua browser utama lainnya). Oleh karena itu, kita dapat menggunakan try and catch untuk menjebak kasus ini dan melakukan sesuatu yang lain jika fungsinya tidak tersedia. Contoh 4 menunjukkan caranya.

Contoh 4 Menjebak kesalahan dengan coba dan tangkap

```
<script>
try
{
request = new XMLHTTPRequest()
}
catch(err)
{
// Use a different method to create an XML HTTP Request object
}
</script>
```

Saya tidak akan membahas bagaimana kami mengimplementasikan objek yang hilang di Internet Explorer di sini, tetapi kita dapat melihat bagaimana sistem bekerja. Ada juga kata kunci lain yang terkait dengan coba dan tangkap yang disebut akhirnya yang selalu dieksekusi,

terlepas dari apakah terjadi kesalahan dalam klausa try. Untuk menggunakannya, cukup tambahkan sesuatu seperti pernyataan berikut setelah pernyataan catch:

```
finally
{
alert("The 'try' clause was encountered")
}
```

Bersyarat

Kondisional mengubah aliran program. Mereka memungkinkan kita untuk mengajukan pertanyaan tentang hal-hal tertentu dan menanggapi jawaban yang kita dapatkan dengan cara yang berbeda. Ada tiga jenis kondisional non-perulangan: pernyataan if, pernyataan switch, dan operator ?.

Pernyataan jika

Beberapa contoh dalam bab ini telah menggunakan pernyataan if. Kode dalam pernyataan seperti itu dijalankan hanya jika ekspresi yang diberikan bernilai benar. Pernyataan multiline if memerlukan kurung kurawal di sekelilingnya, tetapi seperti pada PHP, kita dapat menghilangkan kurung kurawal untuk pernyataan tunggal. Oleh karena itu, pernyataan berikut ini valid:

```
if (a > 100)
{
b=2
document.write("a is greater than 100")
}
if (b == 10) document.write("b is equal to 10")
```

Pernyataan lain

Ketika suatu kondisi belum terpenuhi, kita dapat menjalankan alternatif menggunakan pernyataan lain, seperti ini:

```
if (a > 100)
{
document.write("a is greater than 100")
}
else
{
document.write("a is less than or equal to 100")
}
```

Tidak seperti PHP, JavaScript tidak memiliki pernyataan elseif, tetapi itu tidak masalah, karena kita dapat menggunakan else diikuti oleh if lainnya untuk membentuk ekuivalen dengan pernyataan elseif, seperti ini:

```
if (a > 100)
{
```

```
document.write("a is greater than 100")
}
else if(a < 100)
{
document.write("a is less than 100")
}
else
{
document.write("a is equal to 100")
}
```

Seperti yang kita lihat, kita dapat menggunakan else lain setelah if baru, yang bisa juga diikuti oleh pernyataan if lainnya, dan seterusnya. Meskipun saya telah menunjukkan tanda kurung kurawal pada pernyataan, karena masing-masing adalah satu baris, seluruh contoh sebelumnya dapat ditulis sebagai berikut:

```
if (a > 100) document.write("a is greater than 100")
else if(a < 100) document.write("a is less than 100")
else document.write("a is equal to 100")
```

Pernyataan switch

Pernyataan switch berguna ketika satu variabel atau hasil ekspresi dapat memiliki beberapa nilai, yang masing-masingnya ingin kita jalankan fungsi yang berbeda. Sebagai contoh, kode berikut mengambil sistem menu PHP dan mengubahnya menjadi JavaScript. Ia bekerja dengan melewatkan satu string ke kode menu utama sesuai dengan permintaan pengguna. Katakanlah opsinya adalah Beranda, Tentang, Berita, Masuk, dan Tautan, dan kami mengatur halaman variabel ke salah satunya sesuai dengan input pengguna. Kode untuk ini ditulis menggunakan if ... else if ... mungkin terlihat seperti Contoh 5.

Contoh 5 Pernyataan multiline if...else if...

```
<script>
if (page == "Home") document.write("You selected Home")
else if (page == "About") document.write("You selected About")
else if (page == "News") document.write("You selected News")
else if (page == "Login") document.write("You selected Login")
else if (page == "Links") document.write("You selected Links")
</script>
```

Tetapi menggunakan konstruksi sakelar, kodenya bisa terlihat seperti contoh 6.

Contoh 6 Konstruksi sakelar

```
<script>
switch (page)
{
case "Home":
document.write("You selected Home")
```

```
break
case "About":
document.write("You selected About")
break
case "News":
document.write("You selected News")
break
case "Login":
document.write("You selected Login")
break
case "Links":
document.write("You selected Links")
break
}
</script>
```

Halaman variabel hanya disebutkan satu kali di awal pernyataan switch. Setelah itu, perintah case akan memeriksa kecocokan. Ketika satu terjadi, pernyataan kondisional yang cocok dijalankan. Tentu saja, program nyata akan memiliki kode di sini untuk ditampilkan atau melompat ke halaman, daripada hanya memberi tahu pengguna apa yang dipilih.

Breaking out

Seperti yang kita lihat pada Contoh 7, seperti halnya PHP, perintah break memungkinkan kode kita untuk keluar dari pernyataan switch setelah kondisi terpenuhi. Ingatlah untuk menyertakan jeda kecuali jika kita ingin melanjutkan mengeksekusi pernyataan di bawah kasus berikutnya.

Tindakan default

Ketika tidak ada kondisi yang terpenuhi, kita dapat menentukan tindakan default untuk pernyataan switch menggunakan kata kunci default. Contoh 8 menunjukkan potongan kode yang dapat dimasukkan ke dalam Contoh 7.

Contoh 7 Pernyataan default untuk ditambahkan ke Contoh 6

```
default:
document.write("Unrecognized selection")
break
```

? Operator

Operator ternary (?), dikombinasikan dengan karakter :, menyediakan cara cepat untuk melakukan tes if ... else. Dengannya kita dapat menulis ekspresi untuk dievaluasi, lalu ikuti dengan ? simbol dan kode untuk dieksekusi jika ekspresi benar. Setelah itu, tempatkan a : dan kode yang akan dieksekusi jika ekspresi bernilai false. Pernyataan telah dipecah menjadi beberapa baris untuk kejelasan, tetapi kita akan lebih cenderung menggunakan pernyataan seperti itu pada satu baris.

Looping

Sekali lagi, kita akan menemukan banyak kesamaan yang dekat antara JavaScript dan PHP dalam hal perulangan. Kedua bahasa mendukung while, do ... while, dan for loop.

While Loop

While loop JavaScript pertama-tama memeriksa nilai ekspresi dan mulai mengeksekusi pernyataan di dalam perulangan hanya jika ekspresi itu benar. Jika salah, eksekusi melompat ke pernyataan JavaScript berikutnya (jika ada).

Setelah menyelesaikan iterasi dari loop, ekspresi diuji lagi untuk melihat apakah itu benar dan proses berlanjut sampai ekspresi bernilai salah, atau sampai eksekusi dihentikan. Contoh 9 menunjukkan lingkaran.

Contoh 9 While loop

```
<script>
counter=0
while (counter < 5)
{
document.write("Counter:  " + counter + "<br>")
++counter
}
</script>
```

Skrip ini menghasilkan yang berikut:

Counter: 0
Counter: 1
Counter: 2
Counter: 3
Counter: 4

Jika penghitung variabel tidak bertambah dalam loop, sangat mungkin beberapa browser menjadi tidak responsif karena loop yang tidak pernah berakhir, dan halaman mungkin bahkan tidak mudah dihentikan dengan tombol Escape atau Stop. Jadi berhati-hatilah dengan loop JavaScript Anda.

do ... while Loop

Bila kita memerlukan perulangan untuk melakukan iterasi setidaknya sekali sebelum pengujian dilakukan, gunakan perulangan do ... while, yang mirip dengan while loop, kecuali bahwa ekspresi uji diperiksa hanya setelah setiap iterasi perulangan. Jadi, untuk menampilkan tujuh hasil pertama dalam tabel perkalian tujuh, kita dapat menggunakan kode seperti pada Contoh 10.

Contoh 10. do ... while loop

```
<script>
count = 1
do
{
document.write(count + " times 7 is " + count * 7 + "<br>")
} while (++count <= 7)
</script>
```

Seperti yang kita harapkan, loop ini menghasilkan yang berikut:

1 times 7 is 7
2 times 7 is 14

3 times 7 is 21
4 times 7 is 28
5 times 7 is 35
6 times 7 is 42
7 times 7 is 49

For Loop

For loop menggabungkan yang terbaik dari semua dunia ke dalam satu konstruksi looping yang memungkinkan kita untuk melewatkan tiga parameter untuk setiap pernyataan:

- Ekspresi inisialisasi
- Ekspresi kondisi
- Ekspresi modifikasi

Ini dipisahkan oleh titik koma, seperti ini: for (expr1 ; expr2 ; expr3). Pada awal iterasi pertama dari loop, ekspresi inisialisasi dijalankan. Dalam kasus kode untuk tabel perkalian untuk 7, count akan diinisialisasi ke nilai 1. Kemudian, setiap kali di sekitar loop, ekspresi kondisi (dalam hal ini, count <= 7) diuji, dan loop adalah dimasukkan hanya jika kondisinya benar. Akhirnya, pada akhir setiap iterasi, ekspresi modifikasi dieksekusi. Dalam kasus tabel perkalian untuk 7, jumlah variabel bertambah. Contoh 11 menunjukkan seperti apa kode itu nantinya.

Contoh 11. Menggunakan for loop

```
<script>
for (count = 1 ; count <= 7 ; ++count)
{
document.write(count + "times 7 is " + count * 7 + "<br>");
}
</script>
```

Seperti di PHP, kita dapat menetapkan beberapa variabel dalam parameter pertama untuk loop dengan memisahkannya dengan koma, seperti ini:

```
for (i = 1, j = 1 ; i < 10 ; i++)
```

Demikian juga, kita dapat melakukan beberapa modifikasi pada parameter terakhir, seperti ini:

```
for (i = 1 ; i < 10 ; i++, --j)
```

Atau kita dapat melakukan keduanya secara bersamaan:

```
for (i = 1, j = 1 ; i < 10 ; i++, --j)
```

Keluar dari Loop

Perintah break, yang akan kita ingat penting di dalam pernyataan switch, juga tersedia di dalam for loop. kita mungkin perlu menggunakan ini, misalnya, saat mencari kecocokan. Setelah kecocokan ditemukan, kita tahu bahwa melanjutkan pencarian hanya akan membuang waktu dan membuat pengunjung kita menunggu. Contoh 12 menunjukkan bagaimana menggunakan perintah break.

Contoh 12 Menggunakan perintah break dalam for loop

```
<script>
haystack = new Array()
haystack[17] = "Needle"
for (j = 0 ; j < 20 ; ++j)
{
if (haystack[j] == "Needle")
{
document.write("<br>- Found at location " + j)
break
}
else document.write(j + ", ")
}
</script>
```

Skrip ini menghasilkan yang berikut:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16,
- Found at location 17

Pernyataan lanjutan

Terkadang kita tidak ingin sepenuhnya keluar dari loop, tetapi ingin melewatkan pernyataan yang tersisa hanya untuk iterasi loop ini. Dalam kasus seperti itu, kita dapat menggunakan perintah continue. Contoh 13 menunjukkan ini sedang digunakan.

Contoh 13 Menggunakan perintah continue dalam for loop

```
<script>
haystack = new Array()
haystack[4] = "Needle"
haystack[11] = "Needle"
haystack[17] = "Needle"
for (j = 0 ; j < 20 ; ++j)
{
if (haystack[j] == "Needle")
{
document.write("<br>- Found at location " + j + "<br>")
continue
}
document.write(j + ", ")
}
</script>
```

Perhatikan bagaimana panggilan document.write kedua tidak harus diapit oleh pernyataan else (seperti yang terjadi sebelumnya), karena perintah continue akan melewatkannya jika kecocokan telah ditemukan. Output dari skrip ini adalah sebagai berikut:

0, 1, 2, 3,
- Found at location 4
5, 6, 7, 8, 9, 10,

- Found at location 11
12, 13, 14, 15, 16,
- Found at location 17
18, 19,

Eksplisit casting

Tidak seperti PHP, JavaScript tidak memiliki casting tipe eksplisit seperti (int) atau (float). Sebagai gantinya, ketika kita membutuhkan nilai untuk tipe tertentu, gunakan salah satu fungsi bawaan JavaScript, yang ditunjukkan pada Tabel 7.1.

Tabel 7.1 Fungsi pengubah tipe JavaScript

Perubahan ke tipe	**Fungsi yang akan digunakan**
Int, Integer	**parseInt()**
Bool, Boolean	**Boolean()**
Float, Double, Real	**parseFloat()**
String	**String()**
Array	**split()**

Jadi, misalnya, untuk mengubah angka floating-point menjadi integer, kita bisa menggunakan kode seperti berikut (yang menampilkan nilai 3):
Atau kita dapat menggunakan bentuk majemuk:

```
n = 3.1415927
i = parseInt(n)
document.write(i)
```

Itu saja untuk aliran dan ekspresi kontrol. Bab berikutnya berfokus pada penggunaan fungsi, objek, dan array di JavaScript:

```
document.write(parseInt(3.1415927))
```

7.2 FUNGSI JAVASCRIPT, OBJEK DAN ARRAY

Sama seperti PHP, JavaScript menawarkan akses ke fungsi dan objek. Faktanya, JavaScript sebenarnya didasarkan pada objek, karena—seperti yang telah kita lihat—ia harus mengakses DOM, yang membuat setiap elemen dokumen HTML tersedia untuk dimanipulasi sebagai objek. Penggunaan dan sintaksnya juga sangat mirip dengan PHP, jadi kita akan merasa seperti di rumah sendiri saat saya membawa kita menggunakan fungsi dan objek dalam JavaScript, serta melalui eksplorasi mendalam tentang penanganan array.

Fungsi JavaScript

Selain memiliki akses ke lusinan fungsi (atau metode) bawaan seperti menulis, yang telah kita lihat digunakan di document.write, kita dapat dengan mudah membuat fungsi kita sendiri. Setiap kali kita memiliki bagian kode yang lebih kompleks yang kemungkinan akan digunakan kembali, kita memiliki kandidat untuk suatu fungsi.

Mendefinisikan Fungsi

Sama seperti PHP, JavaScript menawarkan akses ke fungsi dan objek. Faktanya, JavaScript sebenarnya didasarkan pada objek, karena—seperti yang telah kita lihat—ia harus mengakses DOM, yang membuat setiap elemen dokumen HTML tersedia untuk dimanipulasi sebagai objek. Penggunaan dan sintaksnya juga sangat mirip dengan PHP, jadi kita akan merasa seperti di rumah sendiri saat saya membawa kita menggunakan fungsi dan objek dalam JavaScript, serta melalui eksplorasi mendalam tentang penanganan array.
Baris pertama sintaks menunjukkan bahwa:

- Definisi dimulai dengan kata fungsi.
- Nama mengikuti yang harus dimulai dengan huruf atau garis bawah, diikuti dengan sejumlah huruf, angka, simbol dolar, atau garis bawah.
- Tanda kurung diperlukan.
- Satu atau beberapa parameter, dipisahkan dengan koma, bersifat opsional (ditunjukkan dengan kurung siku, yang bukan merupakan bagian dari sintaks fungsi).

Nama fungsi peka huruf besar/kecil, jadi semua string berikut merujuk ke fungsi yang berbeda: getInput, GETINPUT, dan getinput. Dalam JavaScript ada konvensi penamaan umum untuk fungsi: huruf pertama dari setiap kata dalam nama dikapitalisasi kecuali untuk huruf pertama, yaitu huruf kecil. Oleh karena itu, dari contoh sebelumnya, getInput akan menjadi nama pilihan yang digunakan oleh sebagian besar programmer. Konvensi ini biasanya disebut sebagai bumpyCaps, bumpyCase, atau camelCase. Kurung kurawal pembuka memulai pernyataan yang akan dijalankan saat kita memanggil fungsi; kurung kurawal yang cocok harus menutupnya. Pernyataan ini dapat mencakup satu atau lebih pernyataan pengembalian, yang memaksa fungsi untuk menghentikan eksekusi dan kembali ke kode panggilan. Jika suatu nilai dilampirkan ke pernyataan pengembalian, kode panggilan dapat mengambilnya.

7.3 EKSPRESI DAN CONTROL FLOW DI JAVASCRIPT

Di bab sebelumnya, saya memperkenalkan dasar-dasar JavaScript dan DOM. Sekarang saatnya untuk melihat bagaimana membangun ekspresi kompleks dalam JavaScript dan bagaimana mengontrol aliran program skrip kita menggunakan pernyataan bersyarat.

Ekspresi

Ekspresi JavaScript sangat mirip dengan yang ada di PHP. Ekspresi adalah kombinasi nilai, variabel, operator, dan fungsi yang menghasilkan nilai; hasilnya bisa berupa angka, string, atau nilai Boolean (yang dievaluasi menjadi benar atau salah). Contoh 14 menunjukkan beberapa ekspresi sederhana. Untuk setiap baris, itu mencetak huruf antara a dan d, diikuti oleh titik dua dan hasil dari ekspresi. Tag
 ada untuk membuat jeda baris dan memisahkan output menjadi empat baris (ingat bahwa
 dan
 dapat diterima di HTML5, jadi saya memilih untuk menggunakan gaya sebelumnya untuk singkatnya).

Contoh 14. Empat ekspresi Boolean sederhana

```
<script>
document.write("a: " + (42 > 3) + "<br>")
document.write("b: " + (91 < 4) + "<br>")
document.write("c: " + (8 == 2) + "<br>")
document.write("d: " + (4 < 17) + "<br>")
</script>
```

Output dari kode ini adalah sebagai berikut:

a: true
b: false
c: false
d: true

Perhatikan bahwa kedua ekspresi a: dan d: bernilai true. Tapi b: dan c: evaluasi ke false. Tidak seperti PHP (yang masing-masing akan mencetak angka 1 dan tidak ada apa-apa), string true dan false yang sebenarnya ditampilkan. Dalam JavaScript, saat kita memeriksa apakah suatu nilai benar atau salah, semua nilai dievaluasi menjadi true dengan pengecualian berikut ini, yang bernilai false: string false itu sendiri, 0, 0, string kosong, null, undefined, dan NaN (Not a Number, konsep teknik komputer untuk operasi floating-point ilegal seperti pembagian dengan nol). Perhatikan bagaimana saya mengacu pada true dan false dalam huruf kecil. Ini karena, tidak seperti di PHP, nilai-nilai ini harus dalam huruf kecil di JavaScript. Oleh karena itu, hanya yang pertama dari dua pernyataan berikut yang akan ditampilkan, mencetak kata dengan huruf kecil benar, karena yang kedua akan menyebabkan kesalahan 'BENAR' tidak ditentukan:

```
if (1 == true) document.write('true') // True
if (1 == TRUE) document.write('TRUE') // Will cause an error
```

Literal dan Variabel

Bentuk ekspresi yang paling sederhana adalah literal, yang berarti sesuatu yang mengevaluasi dirinya sendiri, seperti angka 22 atau string Tekan Enter. Ekspresi juga bisa berupa variabel, yang mengevaluasi nilai yang telah ditetapkan padanya. Keduanya adalah jenis ekspresi, karena mereka mengembalikan nilai. Contoh 15menunjukkan tiga literal yang berbeda dan dua variabel, yang semuanya mengembalikan nilai, meskipun dari tipe yang berbeda.

Contoh 15. Lima jenis literal

```
<script>
myname = "Peter"
myage = 24
document.write("a: " + 42 + "<br>") // Numeric literal
document.write("b: " + "Hi" + "<br>") // String literal
document.write("c: " + true + "<br>") // Constant literal
document.write("d: " + myname + "<br>") // String variable
document.write("e: " + myage + "<br>") // Numeric variable
</script>
```

Dan, seperti yang kita harapkan, kita melihat nilai pengembalian dari semua ini di output berikut:

```
a: 42
b: Hi
c: true
d: Peter
e: 24
```

Operator memungkinkan kita membuat ekspresi yang lebih kompleks yang mengevaluasi hasil yang bermanfaat. Saat kita menggabungkan konstruksi tugas atau aliran kontrol dengan ekspresi, hasilnya adalah pernyataan. Contoh 16 menunjukkan satu dari masing-masing, pertama memberikan hasil dari ekspresi 366 - day_number ke variabel days_to_new_year, dan yang kedua mengeluarkan pesan ramah hanya jika ekspresi days_to_new_year < 30 bernilai true.

Contoh 16 Dua pernyataan JavaScript sederhana

```
<script>
days_to_new_year = 366 - day_number;
if (days_to_new_year < 30) document.write("It's nearly New Year")
</script>
```

Memegang beberapa nilai dalam array

Variabel memegang satu hal dan mereka melakukannya dengan baik, tetapi ada kalanya kita ingin menyimpan banyak hal. Tentu, kita bisa membuat beberapa variabel, satu untuk setiap hal. kita juga bisa membuat array. Array adalah jenis variabel khusus yang digunakan untuk menampung banyak nilai. Berikut ini contohnya:

```
var myArray = [“Steve”,”Jakob”,”Rebecca”,”Owen”];
```

Array ini berisi empat hal, yang dikenal sebagai elemen. kita akan melihat lebih banyak tentang array nanti, ketika kami memberi tahu kita tentang loop.

Membuat string untuk melacak kata-kata

Saat kita menempatkan kata dalam tanda kutip di JavaScript, kita membuat apa yang disebut string. Biasanya untuk menempatkan konten string ke dalam variabel, seperti ini:

```
var aString = “This is a string.”;
```

String dapat berisi angka, dan ketika kita memasukkan angka dalam tanda kutip, itu akan menjadi string. Kuncinya adalah kutipan, seperti yang ditunjukkan di sini:

```
var anotherString = “This is more than 5 letters long!”;
```

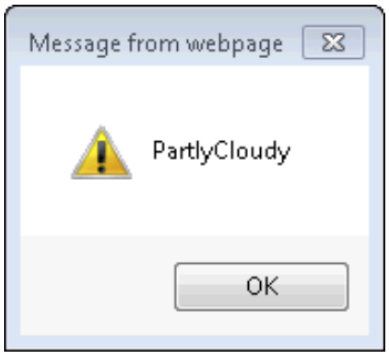
String dapat diapit dalam tanda kutip tunggal atau tanda kutip ganda.

Senar dapat disatukan menggunakan tanda tambah (+), seperti dalam latihan yang akan kita kerjakan. Untuk berlatih membuat string bersambung, mulailah dengan membuka text editor dengan dokumen baru atau kosong. Di text editor, tempatkan HTML dan JavaScript berikut:

```
<!doctype html>
<html>
<head>
<title>JavaScript Chapter 2</title>
</head>
<body>
```

```
<h1>Here's another basic page</h1>
<script type="text/javascript">
 var myString = "Partly" + "Cloudy";
 alert(myString);
</script>
</body>
</html>
```

Simpan file sebagai string.html di root dokumen Anda. Buka browser kita dan lihat halaman dengan membuka http://localhost/string.html. kita akan melihat peringatan seperti pada gambar berikut.



Gambar 7.4 String bersambung.

Perhatikan baik-baik gambar diatas. Perhatikan bahwa tidak ada spasi antara kata Partly dan Cloudy. Untuk memiliki spasi di sana perlu ditambahkan baik di akhir kata Sebagian atau di awal kata Mendung

Bekerja dengan angka

Anda sudah melihat bahwa variabel JavaScript dapat menyimpan angka. kita juga dapat melakukan matematika dengan JavaScript, baik secara langsung pada angka atau melalui variabel. Misalnya, menambahkan dua angka:

```
var myNumber = 4 +
```

Pengurangan dilakukan dengan tanda minus (-), pembagian dengan garis miring (/), dan perkalian dengan tanda bintang (*):

```
//Subtraction
var subtraction = 5 - 3;
//Division
var division = 20 / 5;
//Multiplication
var multiply = 2 * 2;
```

Menguji dengan Persyaratan

Dengan beberapa halaman JavaScript primer selesai, saatnya untuk melihat cara membuat keputusan dengan JavaScript. Keputusan ini disebut kondisional. Cara yang baik untuk menjelaskannya adalah dengan menjelaskan proses pemikiran Steve seputar

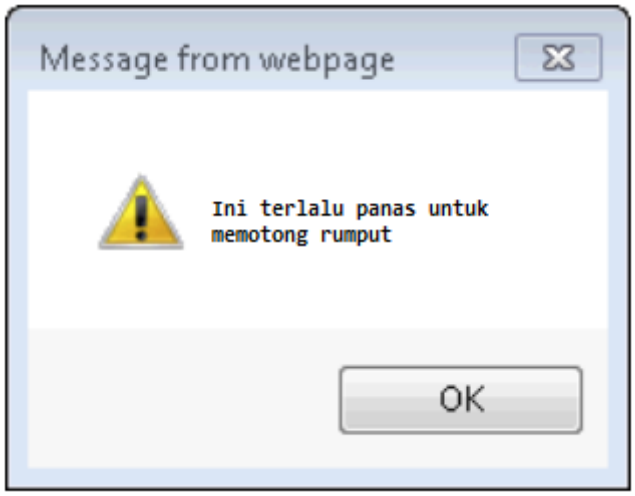
memotong rumput: Jika lebih besar dari 75 derajat, maka terlalu panas untuk memotong. Jika hujan, maka dia tidak bisa memotong rumput. Kalau tidak, dia bisa memotong rumput. Ini dapat diatur dalam JavaScript seperti ini:

```
if (temperatur > 75) {
 alert(“terlalu panas untuk memotong rumput”);
} else if (weather == “hujan”) {
 alert(“Ini hujan, tidak dapat memotong rumb=put”);
} else {
 alert(“Memotong rumput”);
}
```

Sedikit kode itu mengungkapkan semua yang perlu kita ketahui tentang conditional dalam JavaScript! kita menguji suatu kondisi dan kemudian melakukan sesuatu berdasarkan hasil dari kondisi tersebut. Saat kita menyiapkan kondisi, kita menggunakan tanda kurung untuk memuat pengujian dan semua yang kita inginkan terjadi kemudian muncul di antara kurung kurawal buka dan tutup. Kondisional adalah salah satu kasus di mana kita tidak mengakhiri setiap baris dengan titik koma. Inilah latihan yang dapat kita coba untuk bekerja dengan kondisional. Mulailah dengan membuka text editor dengan dokumen baru atau kosong. Di text editor, tempatkan HTML dan JavaScript berikut:

```
<!doctype html>
<html>
<head>
<title>JavaScript Chapter 2</title>
</head>
<body>
<h1>Here’s another basic page</h1>
<script type=”text/javascript”>
 var temperature = 76;
 var weather = “raining”;
 if (temperature > 75) {
 alert(“It’s too hot to mow”);
 } else if (weather == “raining”) {
 alert(“It’s raining, can’t mow”);
 } else {
 alert(“Gotta mow”);
 }
</script>
</body>
</html>
```

Simpan file sebagai cond.html di root dokumen Anda, dan lihat halaman di browser dengan membuka http://localhost/cond.html. kita akan melihat peringatan seperti pada gambar berikut ini.



Gambar 7.5 Peringatan berdasarkan pengujian bersyarat.

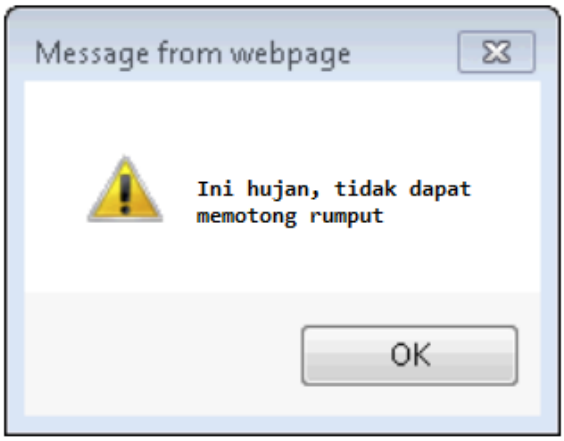
Klik OK untuk mengabaikan peringatan. Untuk melihat bagaimana program merespons saat kita mengubah nilai, di dalam editor, ubah nilai suhu menjadi 70.

Contoh 17 baris yang diubah dicetak tebal:

```
<!doctype html>
<html>
<head>
<title>JavaScript Chapter 2</title>
</head>
<body>
<h1>Here's another basic page</h1>
<script type="text/javascript">
 var temperature = 70;
 var weather = "raining";
 if (temperature > 75) {
 alert("It's too hot to mow");
 } else if (weather == "raining") {
 alert("It's raining, can't mow");
} else {
 alert("Gotta mow");
 }
</script>
</body>
</html>
```

Simpan sebagai cond.html.

Muat ulang halaman di browser kita dengan menekan Ctrl+R atau Command+R. kita akan melihat peringatan seperti pada gambar dibawah ini.



Gambar 7.6 Masuk ke kondisi else if.

Klik OK untuk mengabaikan peringatan. Jika tes gagal, kondisional dapat diatur untuk menjalankan tes lain. Dalam kasus contoh, tes kedua diatur untuk melihat cuaca untuk melihat apakah hujan. Perhatikan penggunaan tanda sama dengan ganda dalam kondisi else if. Terakhir, jika semua pengujian gagal, maka blok kode dapat diatur agar dapat berjalan ketika semuanya gagal. Ini dicatat oleh kata kunci lain dalam contoh kode. Penting untuk dicatat bahwa setelah suatu kondisi benar, dalam contoh setelah suhu lebih besar dari 75, kode di blok itu akan dijalankan tetapi tidak ada kondisi lain yang akan dievaluasi . Ini berarti bahwa tidak ada kode lain di blok lain mana pun yang akan dijalankan.

Melakukan Tindakan Beberapa Kali dengan Loop

Terkadang kita ingin mengulang kode yang sama berulang kali. Ini disebut perulangan, dan JavaScript menyertakan beberapa cara untuk melakukannya, termasuk for dan while.

Untuk apa nilainya

Jika kita ingin melakukan sesuatu beberapa kali dalam JavaScript, cara umum untuk melakukannya adalah dengan for loop. Sebuah for loop memiliki sintaks yang cukup spesifik, seperti yang kita lihat di sini:

```
for (var i = 0; i < 10; i++) {
 // Do something here
}
```

Struktur itu mencakup tiga hal spesifik di dalam tanda kurung.

- Variabel: Pertama, sebuah variabel diatur, dalam hal ini disebut i. Variabel itu diatur ke angka 0.
- Kondisi: Selanjutnya adalah kondisi yang akan diuji. Dalam hal ini, loop menguji apakah variabel i kurang dari 10. Jika i kurang dari 10, kode di dalam kurung kurawal berjalan.
- Operator Postfix: Bagian terakhir dari konstruksi loop for menambah variabel i menggunakan sesuatu yang disebut operator postfix (i++), yang pada dasarnya menaikkan nilai sebesar 1.

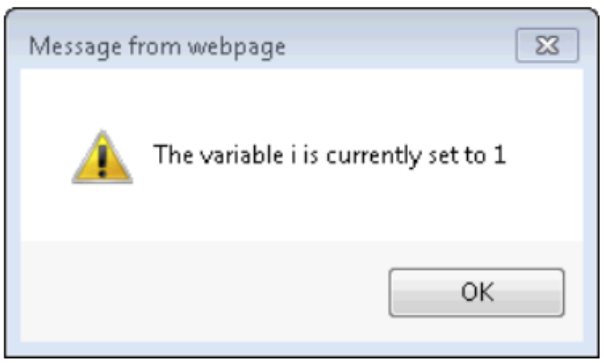
Dalam bahasa sederhana, loop ini membuat variabel dan menyetelnya ke 0, lalu menguji untuk melihat apakah variabel masih kurang dari 10. Jika ya, maka kode di dalam blok akan dieksekusi. Untuk saat ini, kode itu hanyalah sebuah komentar sehingga tidak ada yang terjadi. Jika tidak, maka variabel bertambah 1 dan semuanya dimulai dari awal lagi. Dua bagian pertama dari for loop menggunakan titik koma; bagian terakhir tidak. Pertama kali melalui,

variabel i adalah 0, yang (jelas) kurang dari 10 — dan kode di dalam blok dijalankan sehingga i bertambah 1. Kali berikutnya, nilai i adalah 1, yang masih kurang dari 10, sehingga kode di blok dieksekusi lagi. Ini terus berlanjut sampai nilai I adalah 10. Cobalah dengan latihan. Mulailah dengan membuka text editor dengan dokumen baru atau kosong. Di text editor, tempatkan HTML dan JavaScript berikut:

Contoh 18

```
<!doctype html>
<html>
<head>
<title>JavaScript Chapter 2</title>
</head>
<body>
<h1>Here's another basic page</h1>
<script type="text/javascript">
 for (i = 0; i < 10; i++) {
 alert("The variable i is currently set to " + i);
 }
</script>
</body>
</html>
```

Simpan file sebagai for.html di root dokumen Anda. Lihat file di browser dengan membuka http://localhost/for.html. kita akan melihat serangkaian peringatan, salah satunya ditunjukkan pada gambar berikut:



Gambar 7.7 hasil for loop

Sekarang lihat cara menentukan panjang array. Sebelumnya di bab ini kita melihat array seperti ini:

```
var myArray = ["Steve","Jakob","Rebecca","Owen"]
```

Penggunaan umum for loop adalah untuk memutar array dan melakukan sesuatu dengan setiap nilai. Kondisi dalam contoh for loop yang kita lihat sebelumnya menetapkan nilai pada 10. Tapi bagaimana kita tahu berapa banyak nilai dalam array? Ya, kita dapat dengan mudah menghitung jumlah variabel dalam larik yang ditampilkan, tetapi terkadang kita tidak tahu

berapa banyak elemen dalam larik. kita dapat meminta larik itu sendiri untuk memberi tahu kita berapa lama dengan menanyakannya. kita bertanya melalui properti length, seperti ini:

myArray.length;

Contoh 19 dibawah ini menunjukkan contoh yang mengulang melalui larik yang ditampilkan dan menampilkan setiap elemen.

Contoh 19 Menggunakan for Loop pada Array

```
<!doctype html>
<html>
<head>
<title>JavaScript Chapter 2</title>
</head>
<body>
<h1>Here's another basic page</h1>
<script type="text/javascript">
 var myArray = ["Steve","Jakob","Rebecca","Owen"];
 for (i = 0; i < myArray.length; i++) {
 alert("Hello " + myArray[i]);
 }
</script>
</body>
</html>
```

Kode ini menggunakan standar for loop, kode ini menggunakan properti length untuk mengetahui berapa lama myArrayreally. Setiap kali melalui loop, peringatan ditampilkan, seperti yang ada pada gambar dibawah ini.

Gambar 7.8 penampilan peringatan dari array

Variabel i digunakan tepat di dalam loop, untuk mengakses setiap elemen dari variabel myArray. kita lihat, array adalah daftar hal-hal yang berurutan, dengan urutan yang disediakan oleh angka-angka yang biasanya tidak kita lihat. Angka-angka tersembunyi ini disebut indeks. Indeks elemen pertama dalam array adalah 0 (bukan 1 seperti yang kita harapkan). Dalam contoh ini, karena i adalah 0 pertama kali melalui loop, ia dapat mengakses elemen pertama. Kali kedua melalui loop, seperti yang ditunjukkan pada Gambar 7.5, i sama dengan 1 dan

elemen kedua ditampilkan. Sintaks yang kita lihat di sana, myArray[i], adalah sintaks yang sangat umum yang kita lihat di for loop.

Saat kita di sini

Jenis perulangan lainnya disebut while loop, dan tampilannya seperti ini:

```
while (i < 10) {
 // Do something interesting
 // Don't forget to increment the counter!
}
```

While loop mirip dengan for loop sejauh kode di dalam kurung kurawal dieksekusi selama kondisinya benar. Namun, tidak seperti for loop, kita perlu melakukan sesuatu secara eksplisit di dalam perulangan untuk keluar dari perulangan. Jika kita lupa, kita akan terjebak dalam lingkaran tanpa akhir!

Fungsi

Seperti halnya PHP, fungsi JavaScript digunakan untuk memisahkan bagian kode yang melakukan tugas tertentu. Untuk membuat fungsi, deklarasikan dengan cara yang ditunjukkan pada Contoh 20.

Contoh 20 Deklarasi fungsi sederhana

```
<script>
function product(a, b)
{
return a * b
}
</script>
```

Fungsi ini mengambil dua parameter yang dilewati, mengalikannya, dan mengembalikan produk.

7.4 OBJEK JAVASCRIPT

Objek JavaScript adalah peningkatan dari variabel, yang hanya dapat berisi satu nilai pada satu waktu, di mana objek dapat berisi beberapa nilai dan bahkan fungsi. Sebuah objek mengelompokkan data bersama dengan fungsi yang diperlukan untuk memanipulasinya.

Mendeklarasikan Kelas

Saat membuat skrip untuk menggunakan objek, kita perlu mendesain gabungan data dan kode yang disebut kelas. Setiap objek baru berdasarkan kelas ini disebut instance (atau kejadian) dari kelas itu. Seperti yang telah kita lihat, data yang terkait dengan objek disebut propertinya, sedangkan fungsi yang digunakannya disebut metode. Mari kita lihat cara mendeklarasikan kelas untuk objek bernama Pengguna yang akan berisi detail tentang pengguna saat ini. Untuk membuat kelas, cukup tulis fungsi yang dinamai berdasarkan kelas. Fungsi ini dapat menerima argumen (saya akan menunjukkan nanti bagaimana itu dipanggil) dan dapat membuat properti dan metode untuk objek di kelas itu. Fungsi tersebut disebut konstruktor. Contoh 21 menunjukkan konstruktor untuk kelas Pengguna dengan tiga properti: nama depan, nama pengguna, dan kata sandi. Kelas juga mendefinisikan metode showUser.

Contoh 21 Mendeklarasikan kelas Pengguna dan metodenya

```
<script>
function User(forename, username, password)
{
this.forename = forename
this.username = username
this.password = password
this.showUser = function()
{
document.write("Forename: " + this.forename + "<br>")
document.write("Username: " + this.username + "<br>")
document.write("Password: " + this.password + "<br>")
}
}
</script>
```

Fungsi ini berbeda dari fungsi lain yang telah kita lihat sejauh ini dalam dua cara:

- Ini mengacu pada objek bernama ini. Saat program membuat instance Pengguna dengan menjalankan fungsi ini, ini mengacu pada instance yang sedang dibuat. Fungsi yang sama dapat dipanggil berulang kali dengan argumen yang berbeda, dan akan membuat Pengguna baru setiap kali dengan nilai yang berbeda untuk nama depan properti, dan seterusnya.
- Fungsi baru bernama showUser dibuat di dalam fungsi. Sintaks yang ditampilkan di sini baru dan agak rumit, tetapi tujuannya adalah untuk mengikat showUser ke kelas Pengguna. Dengan demikian, showUser muncul sebagai metode kelas Pengguna.

Konvensi penamaan yang saya gunakan adalah untuk menyimpan semua properti dalam huruf kecil dan menggunakan setidaknya satu karakter huruf besar dalam nama metode, mengikuti konvensi camelCase yang disebutkan sebelumnya dalam bab ini. Contoh 16-5 mengikuti cara yang disarankan untuk menulis konstruktor kelas, yaitu dengan menyertakan metode dalam fungsi konstruktor. Namun, kita juga dapat merujuk ke fungsi yang didefinisikan di luar konstruktor, seperti pada Contoh 22.

Contoh 22 Secara terpisah mendefinisikan kelas dan metode

```
<script>
function User(forename, username, password)
{
this.forename = forename
this.username = username
this.password = password
this.showUser = showUser
}
function showUser()
{
document.write("Forename: " + this.forename + "<br>")
document.write("Username: " + this.username + "<br>")
document.write("Password: " + this.password + "<br>")
```

```
}
</script>
```

Saya tunjukkan formulir ini karena kita pasti akan menemukannya saat membaca kode programmer lain

Membuat Objek

Untuk membuat instance kelas User, kita dapat menggunakan pernyataan seperti berikut:

```
details = new User("Wolfgang", "w.a.mozart", "composer")
```

Atau kita dapat membuat objek kosong, seperti ini:

```
details = new User() and then populat
```

dan kemudian mengisinya nanti, seperti ini:

```
details.forename = "Wolfgang"
details.username = "w.a.mozart"
details.password = "composer"
```

Anda juga dapat menambahkan properti baru ke objek, seperti ini:

```
details.greeting = "Hello"
```

Anda dapat memverifikasi bahwa menambahkan properti baru tersebut berfungsi dengan yang penyataan berikut:

```
document.write(details.greeting)
```

Mengakses Objek

Untuk mengakses suatu objek, kita dapat merujuk ke propertinya, seperti dalam dua contoh pernyataan berikut yang tidak terkait:

```
name = details.forename
if (details.username == "Admin") loginAsAdmin()
```

Jadi, untuk mengakses metode showUser dari objek kelas Pengguna, kita akan menggunakan sintaks berikut, di mana detail objek telah dibuat dan diisi dengan data:

```
details.showUser()
```

Dengan asumsi data yang diberikan sebelumnya, kode ini akan ditampilkan:

```
Forename: Wolfgang
Username: w.a.mozart
```

Password: composer

Kata kunci prototipe

Kata kunci prototipe dapat menghemat banyak memori. Di kelas Pengguna, setiap instance akan berisi tiga properti dan metode. Oleh karena itu, jika kita memiliki 1.000 objek ini di memori, metode showUser juga akan direplikasi 1.000 kali. Namun, karena metodenya identik dalam setiap kasus, kita dapat menentukan bahwa objek baru harus merujuk ke satu instance metode alih-alih membuat salinannya. Jadi, alih-alih menggunakan yang berikut ini di konstruktor kelas:

this.showUser = function()

anda bisa menggantinya dengan ini:

User.prototype.showUser = function()

Contoh 23 menunjukkan seperti apa konstruktor baru itu.

Contoh 23 Mendeklarasikan kelas menggunakan kata kunci prototipe untuk suatu metode

```
<script>
function User(forename, username, password)
{
this.forename = forename
this.username = username
this.password = password
User.prototype.showUser = function()
{
document.write("Forename: " + this.forename + "<br>")
document.write("Username: " + this.username + "<br>")
document.write("Password: " + this.password + "<br>")
}
}
</script>
```

Ini berfungsi karena semua fungsi memiliki properti prototipe, yang dirancang untuk menampung properti dan metode yang tidak direplikasi dalam objek apa pun yang dibuat dari kelas. Sebaliknya, mereka diteruskan ke objeknya dengan referensi. Ini berarti kita dapat menambahkan properti atau metode prototipe kapan saja dan semua objek (bahkan yang sudah dibuat) akan mewarisinya, seperti yang diilustrasikan oleh pernyataan berikut:

```
User.prototype.greeting = "Hello"
document.write(details.greeting)
```

Pernyataan pertama menambahkan properti prototipe salam dengan nilai Hello ke kelas Pengguna. Di baris kedua, detail objek, yang telah dibuat, menampilkan properti baru ini

dengan benar. kita juga dapat menambahkan atau memodifikasi metode di kelas, seperti yang diilustrasikan oleh pernyataan berikut:

```
User.prototype.showUser  = function()
{
document.write("Name " + this.forename +
" User " + this.username +
" Pass " + this.password)
}
details.showUser()
```

Anda dapat menambahkan baris ini ke skrip kita dalam pernyataan bersyarat (seperti jika), sehingga baris tersebut berjalan jika aktivitas pengguna menyebabkan kita memutuskan bahwa kita memerlukan metode showUser yang berbeda. Setelah baris ini dijalankan, meskipun detail objek telah dibuat, panggilan lebih lanjut ke detail.showUser akan menjalankan fungsi baru. Definisi lama showUser telah dihapus.

7.5 MENGGUNAKAN FUNGSI UNTUK MENGHINDARI PENGULANGAN

Praktik pemrograman yang baik adalah menggunakan kembali kode bila memungkinkan. Ini tidak hanya mengurangi jumlah kemungkinan kesalahan dalam kode Anda, tetapi juga mengurangi pekerjaan, yang selalu bagus dalam hal pengkodean. Bagian ini membahas cara utama untuk menerapkan penggunaan kembali kode: fungsi JavaScript menyertakan sejumlah fungsi bawaan. kita telah menggunakan satu di sepanjang bab ini: alert(). Fungsi alert() membuat dialog di browse

Membuat fungsi

Fungsi dibuat dengan kata kunci fungsi diikuti dengan nama fungsi, tanda kurung, dan kurung kurawal buka dan tutup, seperti ini:

```
function myFunction() {
 // Function code goes here
}
```

Apa yang kita lakukan di dalam fungsi terserah Anda. Apa pun yang dapat kita lakukan di luar fungsi dapat kita lakukan di dalamnya. Jika kita menemukan bahwa halaman kita perlu memperbarui banyak HTML, kita dapat menggunakan fungsi sehingga kita tidak perlu terus mengulangi kode yang sama berulang kali.

Menambahkan argumen fungsi

Kekuatan fungsi hadir dengan kemampuannya untuk menerima input, yang disebut argumen, dan kemudian melakukan sesuatu dengan input itu. Misalnya, inilah fungsi sederhana untuk menambahkan dua angka:

```
function addNumbers(num1,num2) {
 alert(num1+num2);
}
```

Fungsi ini menerima dua argumen yang disebut num1 dan num2. Argumen tersebut kemudian digunakan dalam fungsi alert(). kita telah melihat fungsi alert() di sepanjang bab ini dan sekarang kita memahami lebih banyak tentang apa yang terjadi! Fungsi alert() menerima satu argumen, teks untuk ditampilkan dalam dialog alert. Dalam hal ini, karena kita menambahkan dua angka, peringatan menampilkan angka yang dihasilkan. kita mengerjakan latihan untuk ini di bagian berikut.

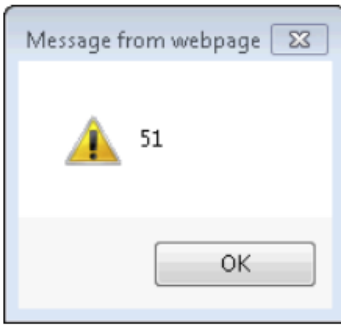
Memanggil fungsi

Membuat fungsi saja tidak cukup; kita perlu menyebutnya juga. Memanggil fungsi berarti kita menjalankannya, sama seperti saat kita memanggil fungsi alert() di awal bab ini. Sampai kita memanggil suatu fungsi, itu tidak benar-benar melakukan apa-apa, seperti fungsi alert() tidak melakukan apa-apa sampai kita memanggilnya.

Memanggil salah satu fungsi kita sendiri terlihat seperti panggilan ke fungsi alert(). Inilah latihan yang kami janjikan. Mulailah dengan membuka text editor dengan dokumen baru atau kosong. Di text editor, tempatkan HTML dan JavaScript berikut:

```
<!doctype html>
<html>
<head>
<title>JavaScript Chapter 2</title>
</head>
<body>
<h1>Here's another basic page</h1>
<script type="text/javascript">
 // Define the function
 function addNumbers(num1,num2) {
 alert(num1+num2);
 }
 // Call the function
 addNumbers(49,2);
</script>
</body>
</html>
```

Simpan file sebagai func.html di root dokumen Anda. Buka browser kita dan arahkan ke http://localhost/func.html. kita akan melihat peringatan seperti yang ditunjukkan pada gambar berikut.



Gambar 7.9 Menjalankan fungsi

Meningkatkan fungsi addNumbers

Fungsi yang telah kita buat, addNumbers(), menerima dua argumen dan menambahkannya. Tetapi bagaimana jika kita mengirim sesuatu yang bukan angka? Fungsi tidak memiliki cara untuk mengujinya dan dengan senang hati mencoba menambahkannya. Untuk bereksperimen dengan ini, ubah 2 dalam panggilan ke addNumbers menjadi "two", seperti ini:

```
addNumbers(49,"two");
```

Saat kita memuat ulang halaman, kita akan melihat peringatan seperti pada Gambar 7.10.

Gambar 7.10 Mencoba menambahkan sesuatu yang bukan angka.

JavaScript menyertakan fungsi untuk menguji apakah sesuatu itu angka. Fungsi tersebut disebut isNaN(), yang merupakan singkatan dari bukan angka. Ini dapat ditambahkan di mana saja yang perlu kita uji untuk memastikan sesuatu adalah angka sebelum bekerja dengannya, seperti dalam kasus fungsi addNumbers() . kita menggunakan fungsi isNaN() dalam kondisi if dan kemudian bereaksi sesuai jika itu bukan angka. Jika kita menyebutnya dengan salah satu dari dua argumen sebagai sesuatu selain angka, kita akan menerima peringatan yang menyatakan itu. Misalnya, jika kita mengubah angka 2 menjadi "two", kita akan menerima peringatan yang ditunjukkan oleh gambar berikut:

Gambar 7.11 Peringatan yang dibuat menggunakan isNaN().

Mengembalikan hasil dari fungsi

Fungsi yang kita buat di bagian ini mengirimkan peringatan. Tetapi ada kalanya kita ingin agar fungsi mengirim sesuatu kembali kepada kita — untuk melakukan sesuatu dan kemudian mengembalikan hasilnya. Kata kunci kembali digunakan untuk mengembalikan hasil dari suatu fungsi. Dalam contoh fungsi addNumbers() yang ditampilkan, alih-alih menggunakan alert() langsung di dalam fungsi, kita bisa mengembalikan hasilnya. Berikut ini contohnya:

```
function addNumbers(num1,num2) {
 var result = num1+num2;
 return result;
 }
```

Anda dapat memanggil fungsi seperti sebelumnya, tetapi sekarang kita perlu menangkap hasilnya, biasanya ke variabel lain, seperti ini:

```
var myResult = addNumbers(49,2);
```

Daftar Mengembalikan Nilai dari Fungsi

```
<!doctype html>
<html>
<head>
<title>JavaScript Chapter 2</title>
</head>
<body>
<h1>Here's another basic page</h1>
<script type="text/javascript">
 // Define the function
 function addNumbers(num1,num2) {
 var result = num1+num2;
 return result;
 }
 // Call the function
 var myResult = addNumbers(49,2);
</script>
</body>
</html>
```

Obyek Secara Singkat

Anda telah melihat array dan bagaimana mereka dapat digunakan untuk menyimpan banyak nilai. Array menyimpan data itu menggunakan indeks bernomor yang tidak terlihat. Di sisi lain, objek dapat menyimpan banyak nilai tetapi nilai tersebut diakses melalui indeks bernama, yang disebut properti. Objek sebenarnya dapat melakukan lebih dari ini, seperti fungsi penahanan (disebut metode) tetapi untuk contoh ini, pertimbangkan fokus sempit ini, menggunakan objek untuk menampung banyak nilai.

Membuat objek

Berikut adalah contoh objek untuk bola:

```
var ball = {
"color": "white",
"type": "baseball"
};
```

Sedangkan array dibuat dengan tanda kurung siku, objek dibuat dengan kurung kurawal, seperti yang ditunjukkan pada contoh. Saat kita mendefinisikan sebuah objek, kita dapat mendefinisikan satu atau lebih properti (mirip dengan elemen dalam array). Dalam contoh ini, kita membuat dua properti, satu disebut warna dan satu lagi disebut tipe. Nilai-nilai tersebut kemudian diatur ke putih dan baseball, masing-masing. kita dapat mengakses objek menggunakan satu titik, seperti:

ball.color

Titik tunggal dikenal sebagai notasi titik bila digunakan dengan cara ini. Daftar dibawah ini menunjukkan HTML untuk membuat objek bola dan menampilkan properti warnanya.

```
<!doctype html>
<html>
<head>
<title>JavaScript Chapter 2</title>
</head>
<body>
<h1>Here's another basic page</h1>
<script type="text/javascript">
 var ball = {
 "color": "white",
 "type": "baseball"
};
 alert(ball.color);
</script>
</body>
</html>
```

Membuat Objek dan Menampilkan Properti

Gambar 7.12 Menampilkan properti objek

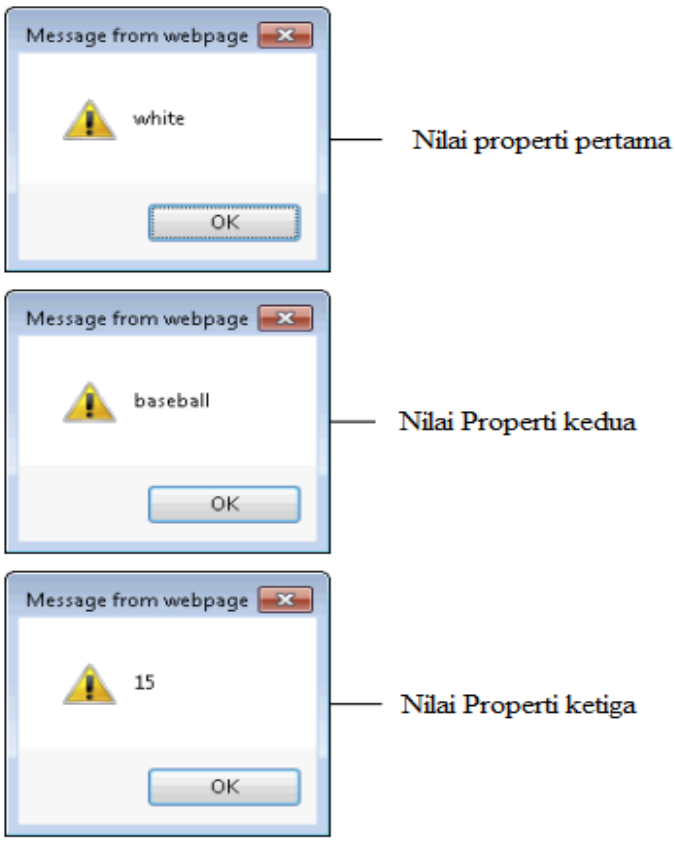
Menambahkan properti ke objek

Terkadang kita ingin menambahkan properti ke objek setelah dibuat. Ini juga dapat dilakukan dengan menggunakan notasi titik, seperti:

```
<!doctype html>
```

```
<html>
<head>
<title>JavaScript Chapter 2</title>
</head>
<body>
<h1>Here's another basic page</h1>
<script type="text/javascript">
 var ball = {
 "color": "white",
 "type": "baseball"
 };

 ball.weight = 15;
 for (var prop in ball) {
 alert(ball[prop]);
 }
</script>
</body>
</html>
```

Berikut adalah latihan untuk membuat objek, menambahkannya, dan kemudian mengulangnya menggunakan konstruktor loop jenis baru. Mulailah dengan membuka text editor dengan dokumen baru atau kosong. Simpan ini sebagai obj.html di root dokumen Anda. Lihat halaman di browser dengan membuka http://localhost/obj.html. kita akan melihat tiga peringatan, seperti gambar berikut:



Gambar 7.13 Tiga peringatan properti objek.

Seperti yang kita lihat dari peringatan, properti yang dibuat bersama dengan objek ditampilkan, seperti properti bobot yang ditambahkan kemudian menggunakan notasi titik. Nanti di bab ini, kita akan melihat lebih banyak tentang objek, jadi sedikit latar belakang ini akan membantu.

7.6 BEKERJA DENGAN DOKUMEN HTML

Semua pemrograman JavaScript ini dapat digunakan secara praktis saat kita mulai menambahkannya ke halaman web. JavaScript terintegrasi ke dalam HTML dan memiliki akses ke semua yang ada di halaman web. Ini berarti kita dapat menambahkan HTML ke halaman, menghapusnya, atau mengubahnya, semuanya dengan cepat, dalam waktu nyata. Agar dapat bekerja sama, JavaScript dan HTML membutuhkan bahasa yang sama agar JavaScript dapat mengetahui apa yang harus dilakukan pada suatu halaman. JavaScript dan HTML bekerja sama melalui sesuatu yang disebut *Document Object Model* (DOM). DOM memberikan akses JavaScript ke halaman web sehingga dapat memanipulasi halaman.

Hubungan antara JavaScript dan HTML adalah melalui documentobject. kita melihat objek dokumen yang digunakan di bagian ini bersama dengan fungsi lain untuk mengakses bagian halaman menggunakan JavaScript. Objek dokumen sebenarnya adalah anak dari objek jendela dan ada anak-anak lain yang menarik juga, seperti yang kita lihat nanti di bab ini.

Mengakses HTML dengan JavaScript

Anda mengakses bagian halaman web, objek dokumen, menggunakan fungsi JavaScript. kita dapat melihat bagian halaman untuk melihat teks apa yang ada di dalamnya atau mengubah teks di halaman. kita juga dapat menambahkan ke halaman dengan JavaScript dan banyak lagi.

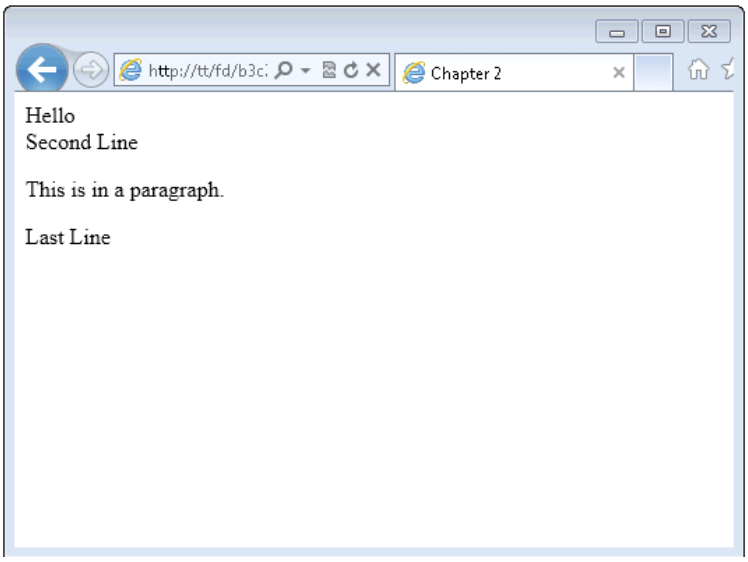
Menggunakan GetElementById untuk mengakses elemen tertentu

Cara paling spesifik agar kita dapat mengakses elemen pada halaman adalah dengan menggunakan ID-nya. Ingat bahwa atribut ID dapat ditempatkan pada elemen apa pun, dan atribut tersebut (seharusnya) unik di seluruh halaman. Dengan cara ini, setiap elemen yang sudah menjadi bagian dari DOM dapat diakses secara langsung daripada dengan melintasi pohon dokumen.

Daftar 24 HTML Dasar untuk Mendemonstrasikan DOM

```
<!doctype html>
<html>
<head>
<title>Chapter 2</title>
</head>
<body>
<div id=”myDiv”>Hello</div>
<div class=”divClass”>Second Line</div>
<div class=”divClass”>
 <p class=”pClass”>This is in a paragraph.</p>
</div>
<div class=”divClass”>Last Line</div>
</body>
</html>
```

HTML itu, ketika dilihat di browser, membuat halaman seperti yang ditunjukkan gambar berikut:



Gambar 7.14 Membuat halaman dasar untuk mendemonstrasikan DOM.

Melihat HTML lagi, kita dapat melihat atribut ID pada pertama pada halaman. kita dapat menggunakan fungsi getElementById untuk mengakses elemen tersebut. kita berkata, "Bagus, saya dapat mengakses elemen tetapi apa yang dapat saya lakukan dengannya?" Senang kita bertanya. Saat mengakses elemen, kita melihat HTML saat ini atau membuat perubahan seperti gaya CSS atau konten sebenarnya dari elemen itu sendiri. Cobalah dalam latihan.

1. Buka text editor dengan dokumen baru atau kosong.
2. Di text editor, tempatkan HTML dan JavaScript berikut:

```
<!doctype html>
<html>
<head>
<title>Chapter 2</title>
</head>
<body>
<div id=”myDiv”>Hello</div>
<div class=”divClass”>Second Line</div>
<div class=”divClass”>
 <p class=”pClass”>This is in a paragraph.</p>
</div>
<div class=”divClass”>Last Line</div>
<script type=”text/javascript”>
 var theDiv = document.getElementById(“myDiv”);
 alert(“The content is “ + theDiv.innerHTML);
</script>
</body>
</html>
```

3. Simpan file sebagai getbyid.html di root dokumen Anda.

4. Buka browser web kita dan lihat halaman di http://localhost/getbyid.html.
   Anda akan melihat peringatan seperti yang ditunjukkan oleh gambar berikut ini:

Gambar 7.15 Peringatan yang dihasilkan oleh fungsi getElementById.

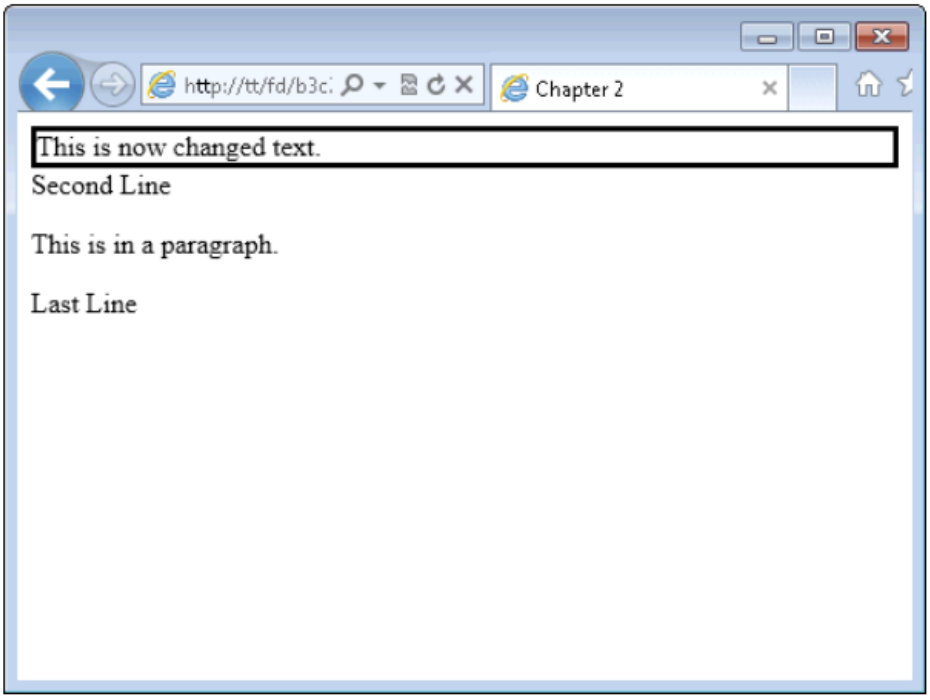
5. Klik OK untuk mengabaikan peringatan dan menutup browser Anda.
6. Kembali ke getbyid.html di editor Anda, hapus baris JavaScript yang dimulai dengan alert(. Ganti baris itu dengan dua ini:

```
theDiv.style.border = “3px solid black”;
theDiv.innerHTML = “This is now changed text.”;
```

Seluruh blok skrip sekarang akan terlihat seperti ini:

```
<script type=”text/javascript”>
 var theDiv = document.getElementById(“myDiv”);
 theDiv.style.border = “3px solid black”;
 theDiv.innerHTML = “This is now changed text.”;
</script>
```

Jika dilihat di browser, halaman tersebut sekarang terlihat seperti pada Gambar 2-16. Perhatikan secara khusus bahwa teks dari baris atas telah diubah dan sekarang memiliki batas. Dalam latihan ini, kita membuat HTML dan JavaScript. Dalam JavaScript, kita mengakses elemen HTML menggunakan fungsi getElementById. Dari sana kita menampilkannya menggunakan alert(). Bagian kedua dari latihan ini melihat kita mengubah konten elemen menggunakan innerHTML dan juga mengubah gaya CSS elemen menggunakan properti style.border.



Gambar 7.16 Halaman setelah diubah dengan getElementById.

Anda sekarang telah melihat cara menggunakan getElementById sebagai bagian dari DOM di JavaScript, jadi periksa salah satu dari daftar ember Anda. Ada baiknya untuk memahami bahwa getElementById ada jika kita perlu bekerja dengan JavaScript orang lain. Namun, ada cara yang lebih baik untuk bekerja dengan halaman web melalui JavaScript dan itu disebut jQuery. kita belajar tentang jQuery di bab selanjutnya. Untuk saat ini, nikmati kemenangan kita atas DOM.

7.7 MODEL OBJEK DOKUMEN

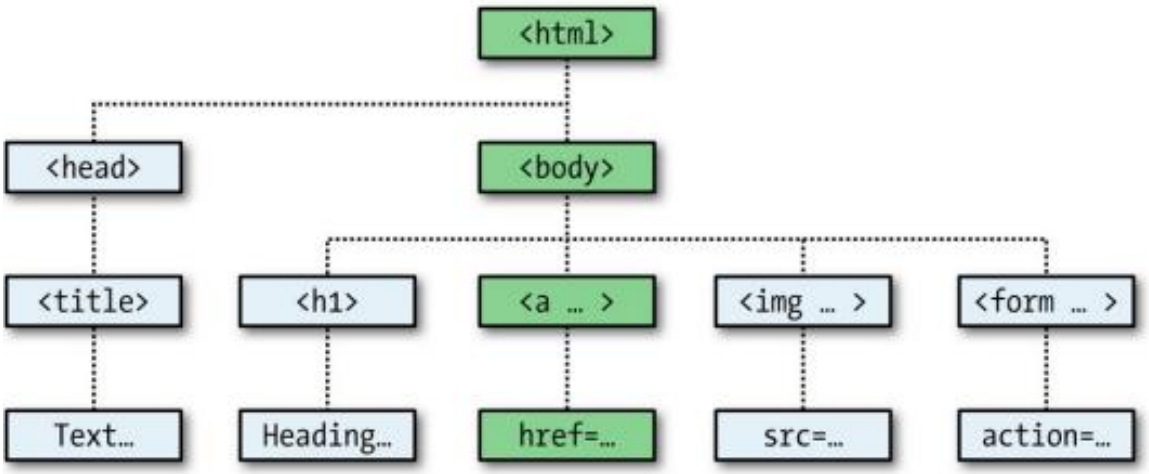
Para desainer JavaScript sangat cerdas. Daripada hanya membuat bahasa skrip lain (yang masih merupakan peningkatan yang cukup bagus pada saat itu), mereka memiliki visi untuk membangunnya di sekitar Model Objek Dokumen, atau DOM. Ini memecah bagian dari dokumen HTML menjadi objek diskrit, masing-masing dengan properti dan metodenya sendiri dan masing-masing tunduk pada kontrol JavaScript. JavaScript memisahkan objek, properti, dan metode menggunakan titik (salah satu alasan bagus mengapa + adalah operator rangkaian string dalam JavaScript, bukan titik). Misalnya, mari kita pertimbangkan kartu nama sebagai objek yang akan kita sebut kartu. Objek ini berisi properti seperti nama, alamat, nomor telepon, dan sebagainya. Dalam sintaks JavaScript, properti ini akan terlihat seperti ini:

```
card.name
card.phone
card.address
```

Metodenya adalah fungsi yang mengambil, mengubah, dan bertindak berdasarkan properti. Misalnya, untuk memanggil metode yang menampilkan properti kartu objek, kita dapat menggunakan sintaks seperti:

```
card.display()
```

Lihat beberapa contoh sebelumnya dalam bab ini dan lihat di mana pernyataan document.write digunakan. Sekarang setelah kita memahami bagaimana JavaScript didasarkan pada objek, kita akan melihat bahwa menulis sebenarnya adalah metode objek dokumen. Di dalam JavaScript, ada hierarki objek induk dan anak, yang dikenal sebagai Model Objek Dokumen.



Gambar 7.17 Contoh hierarki objek DOM

Gambar tersebut menggunakan tag HTML yang sudah kita kenal untuk menggambarkan hubungan induk/anak antara berbagai objek dalam dokumen. Misalnya, URL di dalam tautan adalah bagian dari badan dokumen HTML. Dalam JavaScript, direferensikan seperti ini:

```
url = document.links.linkname.href
```

Perhatikan bagaimana ini mengikuti kolom tengah ke bawah. Bagian pertama, dokumen, mengacu pada tag <html> dan <body>; link.linkname ke tag <a>; dan href ke atribut href. Mari kita ubah ini menjadi beberapa HTML dan skrip untuk membaca properti tautan. Simpan Contoh 25 sebagai linktest.html, lalu panggil di browser Anda.

Jika kita menggunakan Microsoft Internet Explorer sebagai browser pengembangan utama Anda, baca bagian ini terlebih dahulu (tanpa mencoba contohnya), lalu baca bagian yang berjudul Tapi Tidak Sesederhana itu, dan terakhir kembali ke sini dan coba contoh dengan modifikasi getElementById yang dibahas di sana. Tanpa itu, contoh ini tidak akan bekerja untuk Anda.

Contoh 25 Membaca URL tautan dengan JavaScript

```
<html>
<head>
<title>Link Test</title>
</head>
<body>
<a id="mylink" href="http://mysite.com">Click me</a><br>
<script>
url = document.links.mylink.href
document.write('The URL is ' + url)
```

```
</script>
</body>
</html>
```

Perhatikan bentuk pendek dari tag <script> di mana saya telah menghilangkan parameter type="text/JavaScript" untuk menghemat pengetikan. Jika diinginkan, hanya untuk tujuan pengujian ini (dan contoh lainnya), kita juga dapat menghilangkan semua yang ada di luar tag <script> dan </script>. Output dari contoh ini adalah:

Click me

The URL is http://mysite.com

Baris keluaran kedua berasal dari metode document.write. Perhatikan bagaimana kode mengikuti pohon dokumen turun dari dokumen ke tautan ke mylink (id yang diberikan ke tautan) ke href (nilai tujuan URL). Ada juga formulir pendek yang berfungsi sama baiknya, yang dimulai dengan nilai dalam atribut id: mylink.href. Jadi kita bisa mengganti ini:

```
url = document.links.mylink.href
```

dengan berikut ini:

```
url = mylink.href
```

Tapi Tidak Sesederhana Itu

Jika kita mencoba Contoh 25 di Safari, Firefox, Opera, atau Chrome, itu akan bekerja dengan sangat baik. Tetapi di Internet Explorer itu akan gagal, karena implementasi JavaScript Microsoft, yang disebut JScript, memiliki banyak perbedaan halus dari standar yang diakui. Selamat datang di dunia pengembangan web tingkat lanjut! Jadi apa yang bisa kita lakukan tentang ini? Nah, dalam kasus ini, alih-alih menggunakan tautan objek anak dari objek dokumen induk, yang ditolak oleh Internet Explorer, kita harus menggantinya dengan metode untuk mengambil elemen dengan id-nya. Oleh karena itu, baris berikut:

```
url = document.links.mylink.href
```

bisa diganti dengan yang ini :

```
url = document.getElementById('mylink').href
```

Dan sekarang skrip akan berfungsi di semua browser utama. Kebetulan, ketika kita tidak perlu mencari elemen berdasarkan id, formulir singkat berikut akan tetap berfungsi di Internet Explorer, serta browser lainnya:

```
url = mylink.href
```

Penggunaan Lain untuk Simbol $

Seperti disebutkan sebelumnya, simbol $ diperbolehkan dalam variabel JavaScript dan nama fungsi. Karena itu, terkadang kita mungkin menemukan kode yang tampak aneh seperti ini:

```
url = $('mylink').href
```

Beberapa programmer yang giat telah memutuskan bahwa fungsi getElementById begitu lazim dalam JavaScript sehingga mereka telah menulis sebuah fungsi untuk menggantikannya yang disebut $, yang ditunjukkan pada Contoh 26.

Contoh 26 Fungsi pengganti untuk metode getElementById

```
<script>
function $(id)
{
return document.getElementById(id)
}
</script>
```

Oleh karena itu, selama kita telah memasukkan fungsi $ dalam kode Anda, sintaks seperti:

```
$('mylink').href
```

dapat mengganti kode seperti:

```
document.getElementById('mylink').hre
```

Menggunakan DOM

Objek link sebenarnya adalah larik URL, jadi URL mylink pada Contoh 25 juga dapat dirujuk dengan aman di semua browser dengan cara berikut (karena ini adalah link pertama dan satu-satunya):

```
url = document.links[0].href
```

Jika kita ingin mengetahui berapa banyak tautan yang ada di seluruh dokumen, kita dapat menanyakan properti panjang dari objek tautan seperti ini:

```
numlinks = document.links.length
```

Karena itu kita dapat mengekstrak dan menampilkan semua tautan dalam dokumen seperti ini:

```
for (j=0 ; j < document.links.length ; ++j)
document.write(document.links[j].href + '<br>')
```

Panjang sesuatu adalah properti dari setiap array, dan banyak objek juga. Misalnya, jumlah item dalam riwayat web browser kita dapat ditanyakan seperti ini:

```
document.write(history.length)
```

Namun, untuk menghentikan situs web mengintip riwayat penjelajahan Anda, objek riwayat hanya menyimpan jumlah situs dalam larik: kita tidak dapat membaca atau menulis ke nilai ini. Tetapi kita dapat mengganti halaman saat ini dengan halaman dari riwayat, jika kita tahu posisinya di dalam riwayat. Ini bisa sangat berguna dalam kasus di mana kita tahu bahwa halaman tertentu dalam riwayat berasal dari situs Anda, atau kita hanya ingin mengirim browser kembali satu atau beberapa halaman, yang kita lakukan dengan metode go dari objek history. Misalnya, untuk mengirim browser kembali tiga halaman, jalankan perintah berikut:

```
history.go(-3)
```

Anda juga dapat menggunakan metode berikut untuk memundurkan atau meneruskan halaman sekaligus:

```
history.back()
history.forward()
```

Dengan cara yang sama, kita dapat mengganti URL yang sedang dimuat dengan salah satu pilihan Anda, seperti ini:

```
document.location.href = 'http://google.com'
```

Tentu saja, ada lebih banyak hal di DOM daripada membaca dan memodifikasi tautan. Saat kita maju melalui bab-bab berikut tentang JavaScript, kita akan menjadi cukup akrab dengan DOM dan cara mengaksesnya.

Bekerja dengan Browser Web

Primer JavaScript ini diakhiri dengan tampilan cepat pada tampilan JavaScript dari browser web. Seperti yang baru saja kita lihat, saat halaman dimuat, objek dokumen memberikan tampilan halaman ke JavaScript. Demikian juga, beberapa objek lain memberikan JavaScript tampilan browser web itu sendiri. Dengan menggunakan objek ini, yang merupakan anak dari objek jendela, kita dapat melakukan hal-hal seperti mendeteksi jenis browser yang digunakan pengunjung dan juga mengarahkan pengguna ke browser lain. halaman web sepenuhnya.

Mendeteksi browser

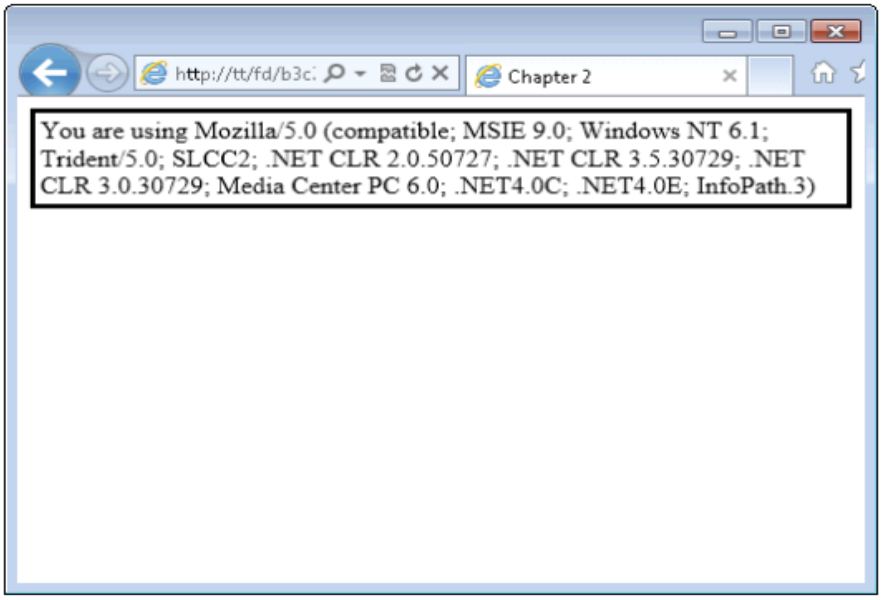
Objek navigator digunakan untuk mendeteksi hal-hal tentang browser pengunjung, seperti versi apa itu. Informasi ini dapat digunakan untuk menyajikan halaman atau layout tertentu kepada pengguna.

Contoh 27 Menampilkan User agent

```
<!doctype html>
<html>
<head>
<title>Chapter 2</title>
</head>
<body>
```

```
<div id=”output”></div>
<script type=”text/javascript”>
 var outputDiv = document.getElementById(“output”);
 outputDiv.style.border  = “3px solid black”;
 outputDiv.style.padding  = “3px”;
 var userAgent = navigator.userAgent;
 outputDiv.innerHTML = “You are using “ + userAgent;
</script>
</body>
</html>
```

Jika dilihat di browser, outputnya terlihat seperti pada gambar berikut.



Gambar 7.18 Melihat properti userAgent

Perhatikan bahwa jika kita menjalankan kode ini, versi browser kita kemungkinan akan berbeda dari ini.

Keterbatasan deteksi browser

Saat kita menggunakan metode seperti yang ditunjukkan di sini, kita perlu menyadari bahwa itu tidak selalu akurat. Mendeteksi browser dengan cara ini hanya bergantung pada apa yang diklaim browser dan informasi ini dapat dipalsukan oleh pengguna. Oleh karena itu, ketika kita menggunakan objek navigator (atau metode “User Agent Sniffer" lainnya), kita harus menyadari bahwa ada batasan pada keakuratannya dan ini jelas tidak 100% sangat mudah.

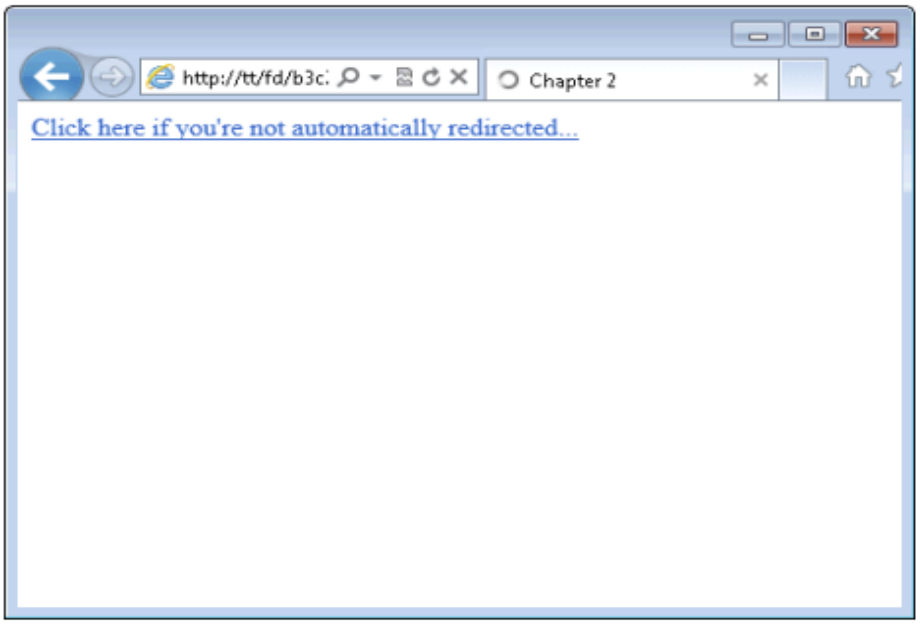
Mengarahkan ke halaman lain

Anda mungkin pernah menemukan satu di suatu tempat di sepanjang jalan, halaman yang mengatakan "Click here if you’re not automatically redirected" dan kemudian secara otomatis mengalihkan Anda. Pernahkah kita bertanya-tanya mengapa mereka repot-repot dengan bagian "Click Here"? Itu dilakukan jika browser kita tidak mengaktifkan JavaScript. Bagian ini menunjukkan kode untuk halaman tersebut. Objek lokasi menyediakan kemampuan untuk mengarahkan ulang ke halaman lain, dan ini adalah salah satu halaman JavaScript paling sederhana yang pernah kita tulis. Daftar dibawah ini menunjukkan HTML dan JavaScript.

Contoh 28 Daftar Halaman Pengalihan

```
<!doctype html>
<html>
<head>
<title>Chapter 2</title>
</head>
<body>
<div>
 <a href=”http://www.braingia.org”> Click here if
you’re not automatically redirected...
 </a>
</div>
<script type=”text/javascript”>
 window.location.replace(“http://www.braingia.org”);
</script>
</body>
</html>
```

Halaman yang dilihat di browser terlihat seperti pada gambar diabwah, tetapi hanya untuk waktu yang singkat. Bahkan, kita mungkin tidak melihatnya sebelum dialihkan!



Gambar 7.19 Halaman pengalihan sesaat sebelum dialihkan.

HTML untuk halaman ini cukup menyiapkan elemen <a> dasar dengan tautan, tidak perlu JavaScript. Kemudian JavaScript menggunakan location.replaceobject untuk mengirim pengguna ke halaman yang berbeda. Hampir tidak ada apa-apanya! Ada lebih banyak objek navigator dan lokasi di JavaScript. Untuk informasi lebih lanjut tentang objek navigator, buka halaman ini di Jaringan Pengembang Mozilla:
https://developer.mozilla.org/en-US/docs/DOM/window.navigator
Untuk informasi lebih lanjut tentang objek lokasi, lihat halaman ini:
https://developer.mozilla.org/en-US/docs/DOM/window.location

BAB 8
MENAMBAHKAN JQUERY

jQuery adalah perpustakaan JavaScript. Oke, itu mungkin tidak masuk akal. Apa itu perpustakaan JavaScript? Pustaka JavaScript adalah kumpulan kode yang kita gunakan saat kita ingin mendapatkan akses ke fungsionalitas tambahan atau membuat hidup lebih mudah. jQuery melakukan keduanya. jQuery hanyalah JavaScript yang kita tambahkan ke halaman web kita untuk membuat penulisan JavaScript lebih mudah. kita masih menggunakan JavaScript dengan jQuery, jadi semua yang kita pelajari di Bab 2 tidak sia-sia. Namun, ada hal-hal tertentu yang dilakukan jQuery jauh lebih baik daripada JavaScript lama biasa. Bekerja dengan halaman web adalah salah satunya. Misalnya, di mana kita mungkin menggunakan getElementById, jQuery memiliki hal-hal yang disebut selector yang memungkinkan cara yang jauh lebih kuat untuk mengakses hal-hal di halaman web untuk digunakan JavaScript. Bab ini menjelaskan cara memulai dengan jQuery dan kemudian menunjukkan beberapa contoh menggunakan jQuery.

8.1 PENGENALAN JQUERY

jQuery cukup populer. Meskipun tidak ada statistik akurat untuk menunjukkan seberapa sering jQuery digunakan, pandangan sepintas ke situs populer menunjukkan bahwa jQuery ada di seluruh web. jQuery juga membuat pengembangan lintas-browser lebih mudah. Meskipun kita belum banyak melihatnya sejauh ini (terutama jika kita telah membaca buku ini dalam urutan linier), dukungan untuk JavaScript sangat berbeda dari browser ke browser dan dari versi ke versi. Apa yang berfungsi di Firefox mungkin tidak berfungsi sama sekali di Internet Explorer atau mungkin bekerja sebaliknya.

Contoh favorit tentang bagaimana dukungan JavaScript berbeda dari browser ke browser melibatkan penanganan tanggal. Ada fungsi JavaScript tertentu yang mengembalikan tahun. Misalnya, dengan asumsi itu tahun 2008 ketika kita memanggil fungsi, JavaScript seharusnya mengembalikan 2008 — tetapi itu tidak selalu terjadi, tergantung pada browser yang kita gunakan. Ketika fungsi itu digunakan di Firefox atau Safari, kita menerima tahun penuh 2008, seperti yang kita harapkan. Saat kita menggunakan JavaScript di Internet Explorer, kita menerima jumlah tahun yang telah berlalu sejak 1900. Saat tahun 2008, kita akan menerima 108 kembali dari Internet Explorer.

Jelas jika kita mencoba melakukan perhitungan tanggal apa pun dengan nilai itu, itu akan menjadi sangat miring. Browser mana yang benar? Itu tidak masalah. Yang penting adalah bahwa produsen browser membaca spesifikasi JavaScript secara berbeda dan pada akhirnya mengembalikan hal yang berbeda untuk fungsi yang sama. Sayangnya, contoh tanggal hanyalah salah satu dari banyak contoh (beberapa jauh lebih serius dari itu) di mana browser berbeda dalam cara mereka mengimplementasikan JavaScript. Kabar baiknya adalah bahwa jQuery menghilangkan komplikasi itu. Fungsi jQuery mencari tahu browser apa yang digunakan dengan cara yang akurat dan kemudian memperhitungkannya untuk membuat browser berperilaku secara konsisten.

8.2 MENGINSTAL JQUERY

Ada dua cara untuk menggunakan jQuery, baik diunduh secara lokal atau di Jaringan Pengiriman Konten (CDN). Salinan lokal hanya itu, file yang berada di dalam root dokumen kita di hard drive server Anda. Versi yang dihosting CDN berarti bahwa file jQuery berada di server orang lain dan kita hanya mereferensikannya dalam kode Anda. Apakah kita menggunakan salinan CDN lokal terserah Anda. Untuk situs web produksi, kami sangat menyarankan untuk menggunakan salinan lokal jQuery untuk kecepatan dan keandalan. Namun, dalam pengembangan, seperti saat kita mengikuti buku ini, boleh saja menggunakan jQuery versi CDN. Contoh buku ini menggunakan jQuery yang dihosting CDN, tetapi bagian ini menunjukkan cara menggunakan lokal dan CDN.

Menginstal jQuery secara lokal

jQuery tersedia sebagai unduhan dari www.jquery.com. Sesampai di sana, pilih versi Produksi dan klik Unduh. Bergantung pada pengaturan browser Anda, kita mungkin berakhir dengan halaman yang penuh dengan kode JavaScript. Jika demikian, pilih Save As dari menu File. Dalam kasus lain, kita hanya akan diminta untuk mengunduh file. Pada akhirnya, kita ingin mendapatkan file bernama seperti jquery-1.8.1.min.js, terlepas dari apakah kita menyimpan file atau mengunduhnya. File harus ditempatkan ke root dokumen Anda. Ingat nama filenya; kita akan membutuhkannya nanti. Itu saja untuk menginstal jQuery — unduh dan masukkan file ke root dokumen Anda.

8.3 MENGGUNAKAN JQUERY YANG DIHOSTING CDN

Opsi yang dihosting CDN untuk jQuery sangat bagus untuk pengembangan. kita tidak perlu khawatir mengunduh file atau meletakkannya di tempat yang tepat; itu selalu tersedia (selama CDN habis). Versi CDN-host tersedia dari banyak pemain besar di web, seperti Google dan Microsoft. kita tidak perlu mengunduh apa pun untuk menggunakan jQuery yang dihosting CDN, jadi bagian ini singkat dan manis. kita dapat menemukan tautan untuk versi yang dihosting CDN di www.jquery.com/download. Bagian selanjutnya menunjukkan cara menambahkan jQuery yang dihosting CDN ke halaman Anda.

Menambahkan jQuery ke Halaman

Sekarang setelah kita mengunduh jQuery atau tahu di mana menemukan versi yang di-host CDN, kita perlu merujuknya di halaman Anda. jQuery sama seperti file JavaScript eksternal. Bagian ini menunjukkan cara menambahkan jQuery ke halaman kita baik untuk jQuery yang dihosting secara lokal maupun jQuery yang dihosting CDN.

Menambahkan jQuery lokal ke halaman

Di bagian sebelumnya, kami menginstruksikan kita untuk mengunduh jQuery dan menempatkannya di root dokumen server web. Jika kita tidak ingat nama filenya, cari di root dokumen Anda. Ini akan dinamai seperti jquery-1.8.1.min.js. (Perhatikan bahwa nomor versi hampir pasti akan berbeda saat kita membaca ini.) Menambahkan jQuery ke halaman berarti menambahkan referensi skrip eksternal, seperti ini:

```
<script type=”text/javascript” src=”jquery-1.8.1-min.js”></
script>
```

Referensi itu biasanya ditambahkan di bagian <head> halaman. Daftar kode html selanjutnya menunjukkan halaman dengan skrip jQuery yang direferensikan di bagian <head>.

Contoh 1 Daftar Menambahkan jQuery ke Halaman

```
<!doctype html>
<html>
<head>
<title>jQuery</title>
<script type=”text/javascript”
 src=”jquery-1.8.1.min.js”></script>
</head>
<body>
<h1>Adding jQuery</h1>
</body>
</html>
```

Menambahkan CDN jQuery ke halaman

Memuat jQuery yang dihosting CDN sama seperti memuatnya secara lokal, tanpa bagian di mana kita menyimpan jQuery di hard drive Anda, tentu saja. Selain detail itu, kita cukup menambahkan jQuery seperti file JavaScript eksternal lainnya. Berikut ini contohnya:

```
<script type=”text/javascript”
src=” http://ajax.aspnetcdn.com/ajax/jQuery/jquery-1.8.1.min.
js”>
</script>
```

Tetapi bagaimana kita mengetahui lokasi rahasia di mana jQuery di-host untuk penggunaan umum? Buka http://jquery.com/download dan kita dapat menemukan jQuery CDNhosted. Di dalam halaman Unduh, kita melihat bagian untuk jQuery yang dihosting CDN. Saat kita menemukan yang ingin kita gunakan, klik kanan dan pilih opsi Salin Lokasi Tautan atau serupa dari menu konteks di browser Anda. Itu akan menyalin URL ke clipboard kita untuk digunakan nanti. Contoh halaman penuh dengan jQuery yang dihosting CDN terlihat sangat mirip dengan halaman untuk salinan yang dihosting secara lokal, hanya atribut src yang berubah. Daftar 3-2 menunjukkan HTML dan JavaScript; perhatikan secara khusus tag <script> di bagian <head>.

Contoh 2 Daftar jQuery yang dihosting CDN

```
<!doctype html>
<html>
<head>
<title>jQuery</title>
<script type=”text/javascript”
 src=”http://ajax.aspnetcdn.com/ajax/jQuery/jquery1.8.1.min.js”>
</script>
</head>
<body>
<h1>Adding jQuery</h1>
</body>
</html>
```

8.4 MENGGABUNGKAN FUNGSI JQUERY READY()

Masalah umum saat memprogram JavaScript adalah program JavaScript akan berjalan sebelum halaman dimuat. Bab sebelumnya menjelaskan bahwa kita dapat mengakses elemen HTML pada halaman. Ini berarti kita juga dapat mengakses hal-hal seperti gambar, formulir, dan apa pun yang kita inginkan, di halaman web. Masalahnya muncul ketika kita mencoba mengakses sesuatu di halaman sebelum browser memuatnya. jQuery menawarkan cara untuk mengatasinya, yang kita lihat di bagian ini.

jQuery memiliki fungsi yang disebut ready() yang menunggu halaman siap. Tidak semuanya tersedia (misalnya, beberapa gambar mungkin masih dimuat), tetapi elemen HTML siap untuk digunakan saat fungsi ready() dipanggil.

Saat kita memprogram dengan jQuery, biasanya menempatkan kode kita di dalam fungsi ready() sehingga kita dapat memastikan bahwa semua hal di halaman siap untuk kita gunakan dalam program Anda. Sungguh, tidak ada banyak untuk ini, jadi cobalah untuk tidak terlalu memikirkannya. Sebuah contoh akan membantu menggambarkan! Contoh Daftar 3 menunjukkan halaman HTML dengan JavaScript di dalam fungsi jQuery ready().

Contoh Daftar Menggunakan Fungsi jQuery ready()

```
<!doctype html>
<html>
<head>
<title>jQuery</title>
<script type=”text/javascript”
 src=”http://ajax.aspnetcdn.com/ajax/jQuery/jquery1.8.1.min.js”>
</script>
</head>
<body>
<h1>Adding jQuery</h1>
<script type=”text/javascript”>
$(document).ready(function() {
 alert(“hi”);
});
</script>
</body>
</html>
```

Jika dilihat melalui browser, hasilnya adalah peringatan seperti gambar berikut ini:

Gambar 8.1 Peringatan yang dihasilkan oleh fungsi jQuery ready().

Kode ini memiliki dua bidang minat. Bagian pertama adalah elemen <script> itu sendiri:

```
<script type=”text/javascript”
 src=”http://ajax.aspnetcdn.com/ajax/jQuery/jquery1.8.1.min.js”>
</script>
```

Ini termasuk jQuery dari CDN Microsoft ke dalam halaman. Area minat berikutnya ada di dalam <body>, khususnya, <script> di dalam body:

```
<script type=”text/javascript”>
$(document).ready(function() {
 alert(“hi”);
});
</script>
```

Kode ini memanggil fungsi jQuery ready(), bagian dari objek dokumen. Perhatikan sintaks khusus dengan tanda dolar dan tanda kurung. Inilah yang memberi tahu browser dan JavaScript bahwa yang berikut akan menjadi jQuery, jadi pemrosesan diserahkan ke jQuery. Dan karena jQuery memiliki fungsi yang disebut ready(), ia tahu apa yang harus dilakukan. kita menggunakan $() di semua tempat dengan jQuery; itulah yang memberi tahu jQuery bahwa ia harus memperhatikan. Di dalam fungsi jQuery ready() ada kode ini:

```
function() {
 alert(“hi”);
}
```

Anda sudah tahu semua tentang fungsi jadi ini bukan sesuatu yang baru. Atau itu? Jika ini adalah fungsi, di mana nama fungsinya? Untuk sebagian besar penggunaan jQuery, kita akan melihat sintaks yang mirip dengan apa yang kita lihat di sini, dengan fungsi tanpa nama seperti ini dan kemudian kode di dalamnya. Saat kita melihat sintaks ini, function(), tanpa nama, itu disebut fungsi anonim. Untuk sebagian besar, kita tidak perlu tahu banyak tentang fungsi anonim sampai kita masuk lebih dalam ke pemrograman JavaScript. Untuk apa yang kita lakukan di sini, ketahuilah bahwa ini adalah sintaks khas yang kita gunakan saat menggunakan jQuery. Di dalam fungsi, peringatan ditampilkan. Tidak mengherankan di sini — ini adalah fungsi alert() standar yang telah kita gunakan di sepanjang buku ini. Tapi apa yang terjadi di sini penting: kita menggunakan jQuery bersama dengan JavaScript di dalam skrip yang sama.

8.5 MEMILIH ELEMEN DENGAN JQUERY

Bagian sebelumnya menjelaskan cara memilih objek dokumen. Ini juga menyediakan banyak cara kerja jQuery. Saat kita menggunakan kode $(document), kita menggunakan sesuatu yang disebut selector. Sebagian besar dari apa yang akan kita lakukan di jQuery terjadi melalui penyeleksi. Misalnya, kita akan sering memilih bagian halaman web dan kemudian meminta jQuery melakukan tindakan pada bagian halaman itu. Tindakan itu bisa apa saja dari menambahkan teks, mengubah HTML, mengubah CSS atau, yah, apa saja yang dapat kita pikirkan! Alur dasar untuk pemrograman JavaScript dengan jQuery adalah ini:

1. Pilih elemen pada halaman web (atau seluruh halaman itu sendiri).
2. Lakukan sesuatu yang menarik dengan elemen itu

Oke, apa yang kita lakukan dengan elemen tidak harus menarik, tetapi kita akan melakukan sesuatu dengan elemen yang dipilih. Sesuatu itu bisa berupa apa saja mulai dari menghapus elemen hingga menambahkan atau mengubahnya atau sekadar mendapatkan informasi dari elemen, seperti teksnya atau gaya CSS saat ini.

Selector jQuery dari dekat

Ada tiga selector utama atau dasar di jQuery. Kami menyebutnya primer atau dasar karena merekalah yang paling sering kita gunakan. kita dapat mengatur penyeleksi yang sangat kompleks berdasarkan struktur halaman, tetapi paling sering kita akan menggunakan salah satu dari tiga penyeleksi ini:

- Berdasarkan kelas
- Dengan ID
- Berdasarkan elemen

Jika kita memiliki beberapa HTML yang terlihat seperti ini:

My Book

Anda dapat mengaksesnya dengan jQuery seperti ini:

$(“#bookTitle”)

Penting untuk dicatat bahwa hal-hal di jQuery (dan JavaScript) peka terhadap huruf besar-kecil. Jadi booktitle tidak sama dengan BOOKTITLE dan tidak sama dengan bookTitle. Tidak masalah case apa yang kita gunakan, asalkan cocok antara HTML dan JavaScript dan jQuery. Sekarang lihat sedikit HTML ini:

This is some text

Selector jQuery terlihat seperti ini:

$(“.titleClass”)

Jika kita berpikir bahwa penyeleksi ini terlihat seperti rekan CSS mereka, kita benar. Jangan khawatir jika kita tidak memikirkannya; tidak akan ada kuis. Di CSS, kita mengakses item dengan ID mereka dengan tanda pound (#) dan kita mengakses kelas dengan satu titik (.):

#bookTitle
.titleClass

Semua yang kita lakukan untuk jQuery adalah membungkusnya dalam konstruksi $() dan menggunakan beberapa tanda kutip juga. Jadi kita mendapatkan ini:

$(“#bookTitle”)
$(“.titleClass

Selektor lain yang sering digunakan mengambil semua elemen dari tipe tertentu. Selector berikut memilih semua elemen di seluruh halaman:

$(“div”)

Ada selector yang lebih maju. Misalnya, kita dapat memilih berdasarkan posisi elemen pada halaman dan, yah, hampir semua kombinasi yang dapat kita pikirkan. Tetapi kita akan menggunakan ketiganya paling sering dan di mana kita membutuhkan lebih banyak, kami akan menunjukkannya kepada Anda.

Filtering

Satu hal tambahan yang harus kita ketahui tentang penyeleksi jQuery adalah kita dapat memfilternya. Ini sangat berguna ketika bekerja dengan formulir dan event.

8.6 BEKERJA DENGAN HTML MENGGUNAKAN JQUERY

Anda dapat menggunakan jQuery untuk melakukan segala macam hal menyenangkan dengan HTML pada halaman dan kami memberikan petunjuk tentang beberapa hal tersebut, seperti menambahkan HTML ke halaman atau mengubah teks, dan sebagainya. Saatnya untuk belajar bagaimana melakukannya!

Menambahkan HTML ke halaman

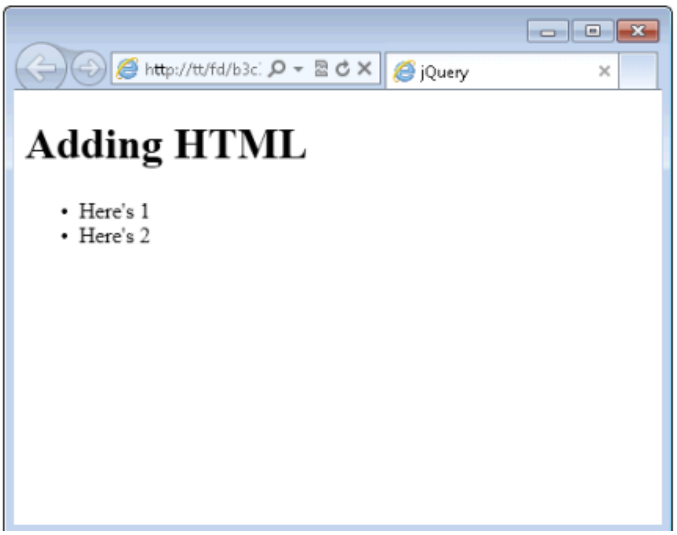
jQuery dapat digunakan untuk menambahkan HTML ke halaman. kita dapat menambahkan segala macam HTML, gambar, apa saja, dan sepenuhnya mengubah layout halaman menggunakan jQuery. Namun, melakukan hal itu bukanlah ide yang bagus, karena bisa sangat, sangat membingungkan untuk mencari tahu dari mana datangnya dan juga bisa lebih sulit untuk dipertahankan di masa mendatang ketika kita perlu mengubah halaman.

Bagaimanapun, menambahkan HTML untuk hal-hal seperti pesan kesalahan atau untuk menambahkan data ke halaman cukup umum. Pikirkan tentang situs perjalanan yang mencari informasi penerbangan. kita mengklik tombol dan itu membangun hasil secara dinamis, tepat di halaman yang sama. Situs-situs tersebut menggunakan JavaScript, berkali-kali jQuery, untuk mencapai prestasi ini. Tetapi sebelum kita mengubah HTML, kita harus mempelajari cara menambahkan HTML ke halaman.

Contoh 4 Daftar HTML dengan Daftar

```
<!doctype html>
<html>
<head>
<title>jQuery</title>
</head>
<body>
<h1>Adding HTML</h1>
<ul id=”theList”>
 <li>Here’s 1</li>
 <li>Here’s 2</li>
</ul>
</body>
</html>
```

Halaman yang dilihat di browser web terlihat seperti pada gambar berikut ini:



Gambar 8.2 Halaman sederhana dengan daftar.

Halaman ini menggunakan daftar tidak berurutan dengan dua item. kita dapat menambahkan item lain ke daftar itu dengan fungsi jQuery append(). Melakukannya berarti memilih elemen <ul>, yang sudah kita ketahui caranya, lalu memanggil fungsi append(). Berikut ini contoh untuk menambahkan item ketiga ke daftar:

$(“#theList”).append(“
Here’s 3”);

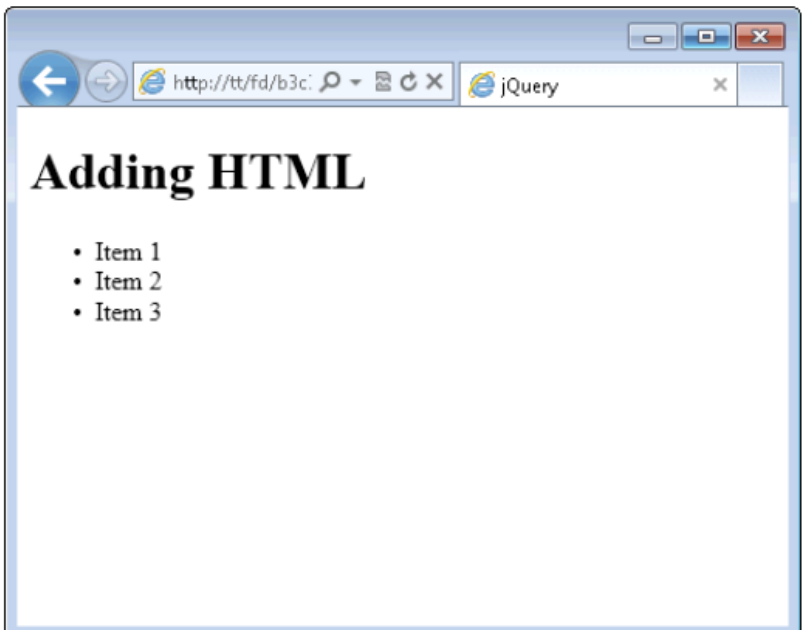
Seperti yang kita lihat, kita memilih elemen <ul> menggunakan selector ID dan kemudian memanggil fungsi append() dengan HTML untuk ditambahkan. Tidak jauh lebih sederhana dari itu. Daftar kode selanjutnya menunjukkan kode akhir. Perhatikan bahwa jQuery telah ditambahkan ke dalamnya di bagian <head> dan fungsi append() berada di dalam fungsi ready(), seperti yang dibahas sebelumnya.

Contoh 5 Daftar Menambahkan Item dengan jQuery

```
<!doctype html>
<html>
<head>
<title>jQuery</title>
<script type=”text/javascript”
 src=”http://ajax.aspnetcdn.com/ajax/jQuery/jquery1.8.1.min.js”>
</script>
</head>
<body>
<h1>Adding HTML</h1>
<ul id=”theList”>
 <li>Here’s 1</li>
 <li>Here’s 2</li>
</ul>
<script type=”text/javascript”>
$(document).ready(function() {
 $(“#theList”).append(“<li>Here’s 3</li>”);
```

```
});
</script>
</body>
</html>
```

Jika dilihat di browser, hasilnya seperti gambar berikut ini:



Gambar 8.3 Menambahkan HTML dengan fungsi append() jQuery

8.7 MENGUBAH ATRIBUT DAN GAYA

jQuery membuat pengambilan dan pengaturan atribut HTML dan gaya CSS menjadi mudah. Ini berarti kita dapat mengubah hal-hal seperti sumber gambar atau kelas CSS atau bahkan gaya CSS itu sendiri. Bagian ini melihat bagaimana melakukan hal itu.

Membaca atribut

Ingat dari cara, jauh sebelumnya dalam buku ini. kita belajar bahwa hal-hal deskriptif yang terkandung di dalam elemen HTML disebut atribut. Sebagai contoh:

```
<a id=”exLink” class=”link” href=”http://www.example.com”>Website</a>
```

Bagian id, class, dan href yang kita lihat di elemen <a> itu semuanya adalah atribut. Menggunakan jQuery, kita dapat mengetahui nilai untuk semua atribut tersebut, dan seperti yang kita lihat nanti, kita juga dapat mengaturnya. Membaca atribut dengan jQuery berarti menggunakan fungsi attr(). Daftar 3-7 menunjukkan kode menggunakan attr untuk membaca atribut href dari tautan yang baru saja kita lihat.

Contoh 6 Menggunakan Fungsi attr()

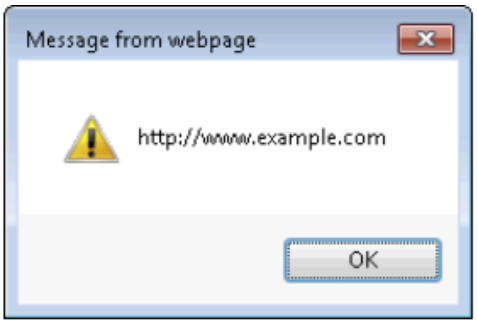
```
<!doctype html>
<html>
<head>
<title>jQuery</title>
<script type=”text/javascript”
 src=”http://ajax.aspnetcdn.com/ajax/jQuery/jquery1.8.1.min.js”>
```

```
</script>
</head>
<body>
<h1>Attributes</h1>
<a id=”exLink” class=”link” href=”http://www.example.com”>Web
site</a>
<script type=”text/javascript”>
$(document).ready(function() {
 alert($(“#exLink”).attr(“href”));
});
</script>
</body>
</html>
```

Sebagian besar pekerjaan dilakukan pada satu baris:

```
alert($(“#exLink”).attr(“href”));
```

Baris itu menggunakan selector untuk memilih elemen dengan ID exLink dan kemudian memanggil fungsi attr() dengan "href" sebagai argumennya. Hasilnya dikembalikan dan ditempatkan di alert(), yang ditunjukkan pada gambar berikut ini:



Gambar 8.4 Mengakses atribut href.

Menulis atribut

Sama seperti fungsi text() dan html(), kita juga dapat mengatur nilai atribut menggunakan fungsi attr(). Misalnya, untuk mengubah nilai atribut href dari kode di sebelumny, dan lakukan ini:

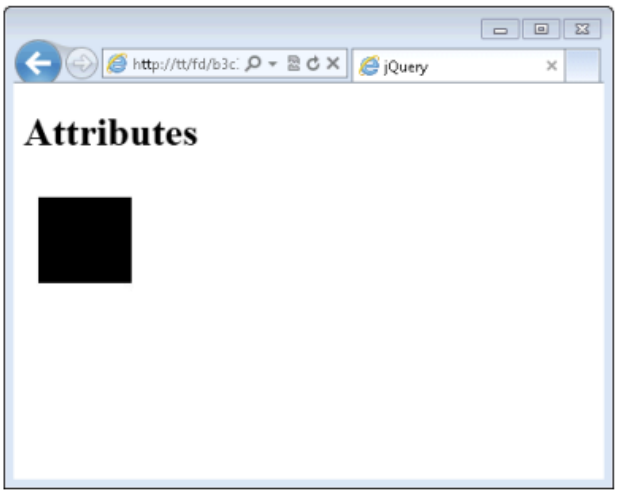
```
$(“#exLink”).attr(“href”, “http://www.braingia.org”);
```

Gambar ditambahkan ke halaman dengan menggunakan atribut src. Ini berarti kita dapat mengubah atribut src untuk mengubah gambar, dengan cepat, melalui JavaScript. Daftar selanjutnya berisi HTML untuk sebuah halaman. HTML berisi gambar yang memuat square.jpg.

```
<!doctype html>
<html>
<head>
```

```
<title>jQuery</title>
</head>
<body>
<h1>Attributes</h1>
<img alt=”square” id=”theImage” src=”square.jpg” />
</body>
</html>
```

Saat dilihat di browser, halaman terlihat seperti Gambar dibawah ini:



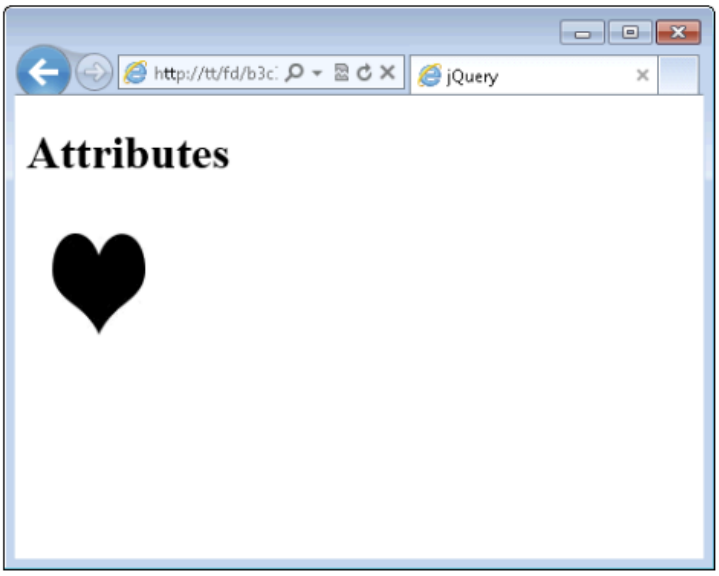
Gambar 8.5 Halaman dengan gambar persegi.

Kita dapat mengubah gambar diatas menjadi gambar lain menggunakan fungsi attr().

Contoh 7 Daftar Mengubah Sumber Gambar

```
<!doctype html>
<html>
<head>
<title>jQuery</title>
<script type=”text/javascript”
 src=”http://ajax.aspnetcdn.com/ajax/jQuery/jquery1.8.1.min.js”>
</script>
</head>
<body>
<h1>Attributes</h1>
<img alt=”square” id=”theImage” src=”square.jpg” />
<script type=”text/javascript”>
$(document).ready(function() {
 $(“#theImage”).attr(“src”,”heart.jpg”);
});
</script>
</body>
</html>
```

Gambar dibawah ini menunjukkan hasil dari koding diatas.



Gambar 8.6 Mengubah gambar melalui jQuery.

Perhatikan baik-baik HTML dan kita akan melihat masalah. kita telah berhasil mengubah atribut src, tetapi atribut alt masih mengatakan bahwa gambar adalah persegi. kita harus mengubah atribut alt agar sesuai dengan gambar. Melakukannya semudah memanggil attr() lagi, kali ini untuk menyetel atribut alt.

$(“#theImage”).attr(“alt”,”heart”);

Karena kita tidak pernah melihat perubahannya, kode pada daftar sebelumnya mungkin tidak terlalu menarik. Lakukan sesuatu untuk mengubah itu. Tambahkan pengatur waktu untuk menunda sakelar. Untuk timer ini, gunakan fungsi JavaScript asli yang disebut setTimeout. Fungsi setTimeout() membutuhkan dua argumen, fungsi untuk memanggil saat timer berakhir dan berapa lama menunggu. Nilai waktu yang kita gunakan dalam milidetik, jadi 2 detik adalah 2000 milidetik. Daftar dibawah ini menunjukkan kode baru.

Contoh 8 Daftar Menunda Perubahan Gambar

```
<!doctype html>
<html>
<head>
<title>jQuery</title>
<script type=”text/javascript”
 src=”http://ajax.aspnetcdn.com/ajax/jQuery/jquery1.8.1.min.js”>
</script>
</head>
<body>
<h1>Attributes</h1>
<img alt=”square” id=”theImage” src=”square.jpg” />
<script type=”text/javascript”>
function changeImage() {
 $(“#theImage”).attr(“src”,”heart.jpg”);
 $(“#theImage”).attr(“alt”,”heart”);
```

```
}
$(document).ready(function() {
 setTimeout(changeImage,2000);
});
</script>
</body>
</html>
```

Kode ini membangun fungsi yang disebut changeImage(). Di dalam fungsi itu ada baris jQuery yang sama dengan yang kita miliki pada contoh sebelumnya. Di dalam fungsi ready(), sekarang ada panggilan ke setTimeout dengan dua argumen fungsi yang telah kami sebutkan, fungsi changeImage, dan 2000, untuk penundaan 2 detik.

Mengubah CSS

Anda juga dapat mengubah informasi gaya pada halaman, baik dengan menyetel gaya secara langsung atau dengan mengubah kelas CSS yang diterapkan ke elemen. Kelas hanyalah atribut lain pada suatu elemen, jadi mengubah CSS berarti menggunakan fungsi attr() lagi.

Menambahkan kelas

Daftar berikut berisi HTML dasar dengan beberapa informasi gaya di bagian <head>.

Contoh 9 Daftar HTML dengan CSS

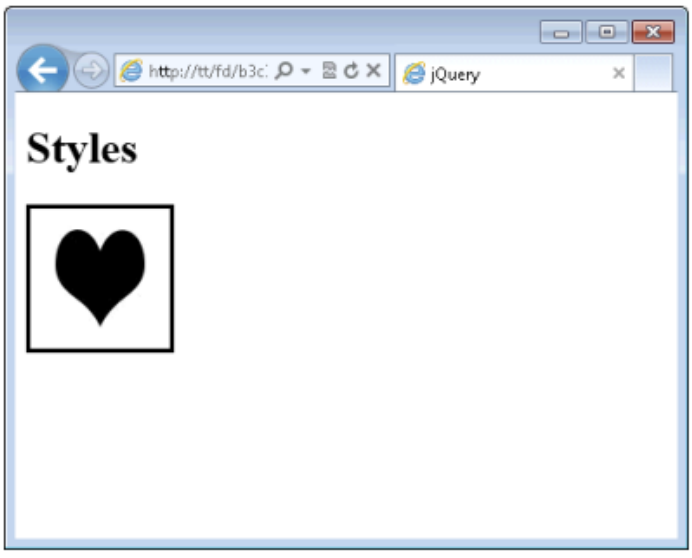
```
<!doctype html>
<html>
<head>
<title>jQuery</title>
<script type=”text/javascript”
 src=”http://ajax.aspnetcdn.com/ajax/jQuery/jquery1.8.1.min.js”>
</script>
<style type=”text/css”>
.borderClass {
 border: 3px solid black;
}
</style>
</head>
<body>
<h1>Styles</h1>
<img alt=”heart” id=”theImage” src=”heart.jpg” />
</body>
</html>
```

Daftar Menambahkan Kelas Menggunakan attr()

```
<!doctype html>
<html>
<head>
<title>jQuery</title>
<script type=”text/javascript”
 src=”http://ajax.aspnetcdn.com/ajax/jQuery/jquery1.8.1.min.js”>
```

```
</script>
<style type=”text/css”>
.borderClass {
 border: 3px solid black;
}
</style>
</head>
<body>
<h1>Styles</h1>
<img alt=”heart” id=”theImage” src=”heart.jpg” />
<script type=”text/javascript”>
$(document).ready(function() {
 $(“#theImage”).attr(“class”,”borderClass”);
});
</script>
</body>
</html>
```

Kode hanya memanggil attr() untuk mengubah atribut class menjadi borderClass. Dalam hal ini, sebenarnya belum ada atribut class pada elemen tersebut, jadi jQuery cukup pintar untuk menambahkan satu untuk Anda.



Gambar 8.7 Menambahkan kelas melalui fungsi attr().

Tetapi apa yang harus dilakukan jika sudah ada kelas pada elemen tersebut? kita dapat mengambil kelas terlebih dahulu menggunakan attr dan kemudian menambahkan yang lain. Atau kita bisa menggunakan fungsi jQuery untuk menambahkan kelas, yang disebut addClass(). Fungsi addClass() tidak mengganggu kelas lain yang sudah diterapkan ke elemen; itu hanya menambahkan kelas lain untuk itu.

```
$(“#theImage”).attr(“class”,”borderClass”);
```

ke:

$(“#theImage”).addClass(“borderClass”);

Dengan perubahan sederhana itu, class borderClass akan ditambahkan dan kita tidak perlu khawatir untuk menghapus kelas lain yang diterapkan ke elemen tersebut.

Menghapus kelas

Pendamping untuk fungsi addClass(), yang disebut removeClass, mengambil kelas dari sebuah elemen. Seperti addClass(), removeClass() tidak memengaruhi kelas lain yang ada di elemen; itu hanya menghapus kelas yang ditentukan. Sintaks untuk removeClass seperti sintaks addClass:

$(“#theImage”).removeClass(“borderClass”);

Di bab berikutnya, kita akan melihat fungsi terkait lainnya yang disebut toggleClass yang menambahkan atau menghapus kelas, bergantung pada apakah itu sudah diterapkan ke elemen.

BAB 9
REAKSI EVENT DENGAN JAVASCRIPT DAN JQUERY

Event (peristiwa) adalah hal-hal yang terjadi. Misalnya, matahari terbit adalah suatu peristiwa. Matahari terbenam adalah sebuah peristiwa. kita dapat memilih untuk bereaksi terhadap peristiwa tersebut. Misalnya, saat matahari terbit, kita mungkin bangun dari tempat tidur — atau mungkin tidak. Saat matahari terbenam, kita mungkin menyalakan lampu atau mungkin pergi tidur. Ketika datang ke pemrograman web, peristiwa adalah hal-hal yang terjadi di halaman web. Misalnya, pengguna mungkin menggerakkan mouse di atas tombol, mengklik tombol, atau mengirimkan formulir. Seperti contoh matahari terbit, kita dapat memilih untuk bereaksi terhadap peristiwa tersebut atau kita dapat mengabaikannya. Jika semua yang ingin kita lakukan adalah mengabaikan peristiwa, maka ini akan menjadi bab yang sangat, sangat singkat. Tetapi kita mungkin ingin bereaksi terhadap peristiwa, dan kami menunjukkan cara melakukannya untuk beberapa skenario umum.

9.1 MEMAHAMI EVENT

Secara umum, ada empat jenis acara yang kita khawatirkan:

Persitiwa formulir: Mencakup hal-hal seperti memilih kotak centang atau mengirimkan formulir itu sendiri. Bereaksi untuk membentuk acara adalah salah satu hal paling umum yang dilakukan oleh programmer JavaScript.

Peristiwa mouse: Ini bisa berupa apa saja, mulai dari klik mouse hingga gerakan mouse di halaman. Betul sekali; kita benar-benar dapat melacak di mana mouse berada pada halaman dan bereaksi terhadapnya.

Peristiwa keyboard: Hal-hal yang terjadi melalui keyboard, seperti penekanan tombol, dianggap peristiwa keyboard.

Peristiwa halaman: Hal-hal seperti pemuatan atau pembongkaran halaman dianggap sebagai peristiwa halaman. kita akan senang mengetahui bahwa kita telah bereaksi terhadap peristiwa pemuatan halaman melalui fungsi jQuery ready().

Berbicara tentang jQuery, bab ini sebagian besar berkonsentrasi pada penggunaan jQuery untuk bekerja dengan acara. Menggunakan jQuery menghemat banyak masalah kompatibilitas yang muncul saat kita mulai mencoba membuat kode JavaScript kita berfungsi di seluruh browser.

9.2 BEKERJA DENGAN FORMULIR

Bab-bab sebelumnya dari buku ini menunjukkan formulir yang sedang dibuat dan gaya bentuk itu dengan CSS. Sekarang saatnya mempelajari cara bekerja dengan formulir itu menggunakan JavaScript. Cara yang sering digunakan JavaScript dengan formulir adalah dengan memberikan validasi formulir saat pengguna mengisinya atau sebelum dikirimkan ke server.

Menambahkan Handler Kirim

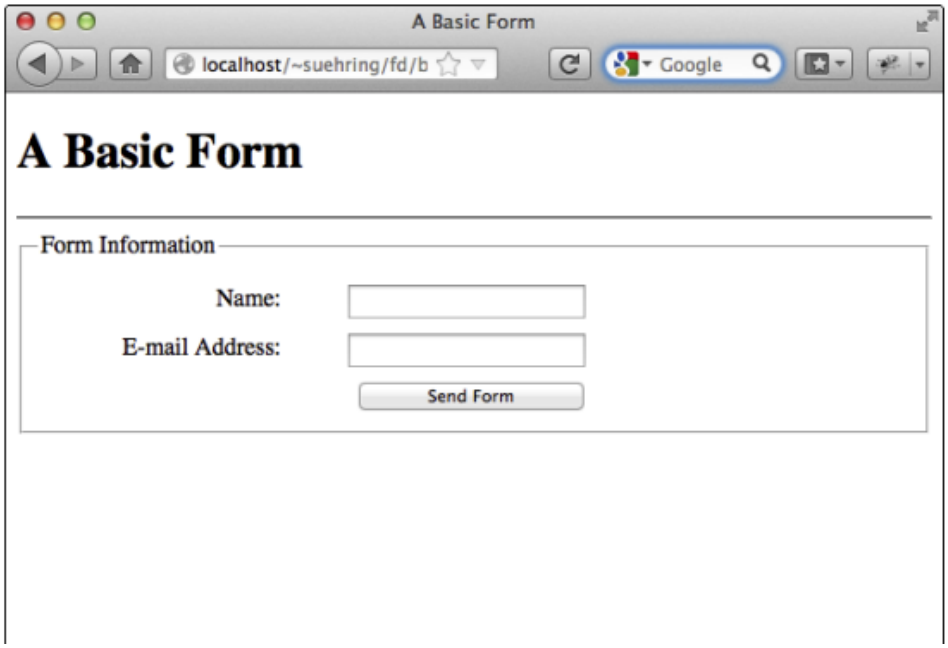
jQuery menyertakan fungsi yang secara otomatis mengawasi formulir yang akan dikirimkan. Contoh 1 menunjukkan HTML untuk formulir yang digunakan sebelumnya dalam buku ini. Kami telah mengambil kebebasan untuk menambahkan jQuery ke bagian <head> formulir.

Contoh 1 Daftar Bentuk Sederhana

```
<!doctype html>
<html>
<head>
<title>A Basic Form</title>
<script type=”text/javascript”
src=”http://ajax.aspnetcdn.com/ajax/jQuery/jquery1.8.1.min.js”>
</script>
<style type=”text/css”>
.form-field {
 clear: both;
 padding: 10px;
 width: 350px;
}
.form-field label {
 float: left;
 width: 150px;
 text-align: right;
}
.form-field input {
 float: right;
 width: 150px;
 text-align: left;
}
#submit {
text-align: center;
}
</style>
</head>
<body>
<h1>A Basic Form</h1>
<hr>
<form action=”#”>
<fieldset>
 <legend>Form Information</legend>
 <div class=”form-field”>
 <label for=”username”>Name:</label>
 <input type=”text” id=”username”
name=”username”>
 </div>
 <div class=”form-field”>
 <label for=”email”>E-mail Address:</label>
 <input type=”text” id=”username”
name=”email”>
 </div>
 <div class=”form-field”>
```

```
 <input id=”submit” type=”submit”
name=”submit” value=”Send Form”>
 </div>
</fieldset>
</form>
</body>
</html>
```

Halaman yang dilihat di browser terlihat seperti gambar berikut ini:



Gambar 9.1 Formulir web dasar.

Saat ini, ketika kita mengklik Kirim Formulir, tidak ada yang terjadi. Ubah itu dengan mengikuti langkah-langkah ini.

1. Buka text editor.
2. Di editor, tempatkan kode berikut:

```
<!doctype html>
<html>
<head>
<title>A Basic Form</title>
<script type=”text/javascript”
 src=”http://ajax.aspnetcdn.com/ajax/jQuery/jquery-1.8.1.min.js”>
</script>
<style type=”text/css”>
.form-field {
 clear: both;
 padding: 10px;
 width: 350px;
}
.form-field label {
```

```
 float: left;
 width: 150px;
 text-align: right;
}
.form-field input {
 float: right;
 width: 150px;
 text-align: left;
}
#submit {
 text-align: center;
}
</style>
</head>
<body>
<h1>A Basic Form</h1>
<hr>
<form action="#">
<fieldset>
 <legend>Form Information</legend>
 <div class="form-field">
 <label for="username">Name:</label>
 <input type="text" id="username"
name="username">
 </div>
 <div class="form-field">
 <label for="email">E-mail Address:</
label>
 <input type="text" id="username"
name="email">
 </div>
<div class="form-field">
 <input id="submit" type="submit"
name="submit" value="Send Form">
 </div>
</fieldset>
</form>
</body>
</html>
```

3. Simpan formulir sebagai form1.html di root dokumen Anda.
4. Lihat halaman di browser web di http://localhost/form1.html.
5. Sekarang tambahkan kode berikut, tepat setelah tag penutup </form> dan sebelum tag </body> penutup.

```
<script type="text/javascript">
```

```
$(document).ready(function() {
 $("form").submit(function() {
 alert("You submitted the form");
 return false;
 });
});
</script>
```

6. Simpan file, lagi sebagai form1.html.
7. Lihat halaman di browser; kita juga dapat memuat ulang halaman dengan Ctrl+R atau Command+R jika kita membiarkannya terbuka dari langkah sebelumnya.
8. Klik Kirim Formulir.
   Anda akan menerima peringatan seperti yang ditunjukkan pada Gambar 9.2.

Gambar 9.2 Peringatan ini mengonfirmasikan bahwa input telah dikirimkan.

```
$(document).ready(function() {
 $("form").submit(function() {
 alert("You submitted the form");
 return false;
 });
});
```

Anda telah menambahkan pengendali acara kirim. Lihat kodenya dimulai dengan fungsi ready(), yang telah kita lihat sebelumnya. Selanjutnya, kita memilih formulir dengan memilih semua elemen <form> pada halaman. Jika ada lebih dari satu formulir, kita mungkin ingin memberi nama atau ID formulir tersebut sehingga kita dapat memilih yang tepat, tetapi untuk contoh ini, cukup memilih berdasarkan elemen berfungsi. Selanjutnya, fungsi submit() dipanggil dan fungsi lain dibuat di dalamnya. Tugas utama fungsi ini adalah menampilkan peringatan, yang kita lihat. Baris kedua dalam fungsi, return false, menarik untuk form. Saat kita menggunakan return false dalam acara pengiriman formulir, kita pada dasarnya mencegah pengiriman formulir ke server. Oleh karena itu, kita hanya ingin melakukan ini untuk alasan tertentu, seperti saat formulir tidak valid seperti saat pengguna belum mengisi semua bidang yang diperlukan. Saat kita menambahkan return false, kita mencegah tindakan default. Karena tindakan default formulir adalah mengirimkan ke server, menambahkan return false mencegah terjadinya tindakan default tersebut. Cara lain untuk mencegah tindakan default adalah dengan fungsi jQuery preventDefault(). Kita menggunakan preventDefault dalam keadaan tertentu di mana return false tidak melakukan apa yang kita

inginkan. Mengubah JavaScript dari contoh sebelumnya untuk menggunakan preventDefault dan return false akan terlihat seperti ini:

```
$(document).ready(function() {
 $(“form”).submit(function(event) {
 alert(“You submitted the form”);
 event.preventDefault();
 return false;
 });
});
```

Memeriksa bidang kosong

Meskipun kami menyertakan seluruh bab tentang validasi di sini kami memberikan sekilas tentang validasi formulir, setidaknya memeriksa bidang kosong. Perhatikan bentuk dari bab sebelumnya dan ditunjukkan pada Gambar 9.1. Katakan bahwa bidang Nama diperlukan; itu berarti pengguna perlu mengisi sesuatu agar formulir dikirim ke server. kita dapat mengubah JavaScript untuk memastikan bahwa bidang telah diisi. Ingat bahwa ID bidang Nama adalah "nama pengguna". jQuery dapat memilihnya dengan cukup mudah, dan kemudian masalah mendapatkan nilai untuk bidang tersebut. Untuk itu, kita dapat menggunakan fungsi val() jQuery. Terakhir, yang perlu kita lakukan adalah memasukkan semuanya ke dalam pernyataan if untuk memastikan nilainya tidak kosong. Kodenya akan terlihat seperti ini:

```
$(document).ready(function() {
 $(“form”).submit(function(event) {
 if ($(“#username”).val() == “”) {
 alert(“Name can’t be blank”);
 event.preventDefault();
 return false;
 }
 });
});
```

Anda dapat mencoba ini dengan mengganti JavaScript dari contoh sebelumnya dengan kode itu. Jika kita mencoba mengirimkan formulir tanpa mengisi apa pun di bidang Nama, kita akan menerima peringatan seperti gambar dibawah ini:

Gambar 9.3 Validasi dasar pada formulir.

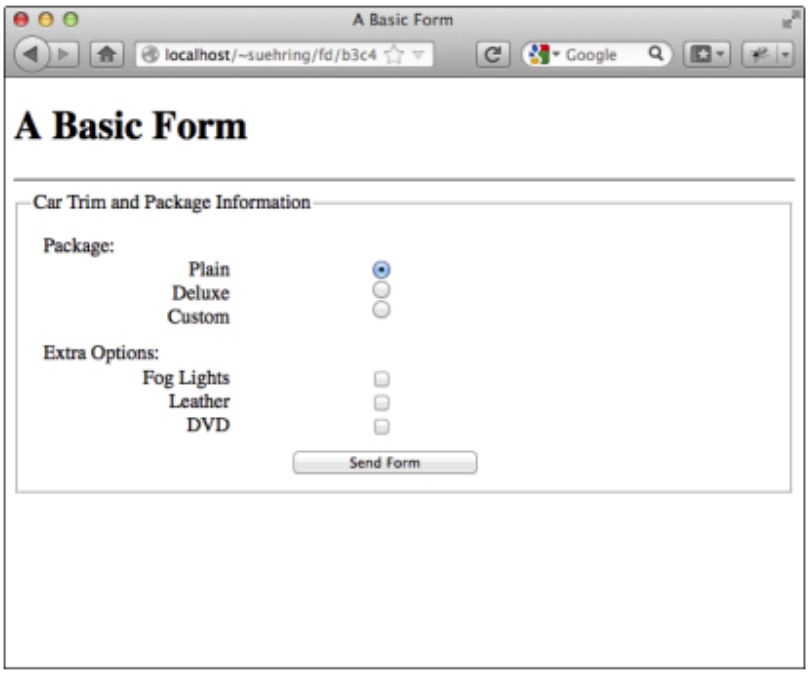
Validasi formulir ini sangat mendasar. Misalnya, kita bisa menempatkan satu ruang kosong di bidang itu dan itu akan valid.

9.3 MEMANTAU MOUSE EVENT

Anda dapat melihat dan bereaksi terhadap peristiwa mouse dengan JavaScript. Bagian ini membahas bagaimana melakukan keduanya.

Menangkap klik mouse

Hal yang umum dilakukan adalah merekam peristiwa klik mouse dengan JavaScript. Misalnya, ketika seseorang mengklik gambar atau mengklik elemen formulir, kita dapat bereaksi untuk memuat gambar yang berbeda atau memilih elemen formulir lainnya. Bayangkan kita telah menyiapkan toko mobil tempat orang dapat menyesuaikan mobil mereka dengan beberapa peningkatan. kita mengkhususkan diri dalam menambahkan lampu kabut, trim kulit, dan pemutar DVD. Orang-orang dapat mengunjungi situs web kita dan memilih level trim. Gambar dibawah ini menunjukkan halaman contoh tempat pengguna memilih opsi.



Gambar 9.4 Selectoran opsi trim

Jika orang memilih paket Deluxe, formulir harus mencentang semua kotak centang Opsi Ekstra. Jika orang memilih paket Biasa, semua opsi harus dihapus centangnya. Terakhir, jika orang memilih opsi individual dalam daftar Opsi Ekstra, maka paket Kustom harus dicentang. Ini semua dapat dicapai dengan beberapa baris JavaScript dan jQuery. Daftar 2 menunjukkan HTML, CSS, dan JavaScript untuk membuat perilaku yang diinginkan.

Contoh 2 Daftar membuat formulir custom

```
<!doctype html>
<html>
<head>
<title>A Basic Form</title>
<script type=”text/javascript”
 src=”http://ajax.aspnetcdn.com/ajax/jQuery/jquery1.8.1.min.js”>
</script>
<style type=”text/css”>
```

```
.form-field {
 clear: both;
 padding: 10px;
 width: 350px;
}
.form-field label {
 float: left;
 width: 150px
text-align: right;
}
.form-field input {
 float: right;
 width: 150px;
 text-align: left;
}
#submit {
 text-align: center;
}
</style>
</head>
<body>
<h1>A Basic Form</h1>
<hr>
<form action="#">
<fieldset>
 <legend>Car Trim and Package Information</legend>
 <div class="form-field">
 <div>Package: </div>
 <input id="plain" type="radio" name="trim"
 value="plain">
 <label for="plain">Plain</label>
 <input id="deluxe" type="radio" name="trim"
 value="deluxe">
 <label for="deluxe">Deluxe</label>
 <input id="custom" type="radio" name="trim"
 value="custom">
 <label for="custom">Custom</label>
 </div>
 <div class="form-field">
 <div>Extra Options:</div>
 <input type="checkbox" id="foglights"
name="option"
 value="foglights">
 <label for="foglights">Fog Lights</label>
 <input type="checkbox" id="leather" name="option"
 value="leather">
```

```
 <label for=”leather”>Leather</label>
 <input type=”checkbox” id=”dvd” name=”option”
 value=”dvd”>
 <label for=”dvd”>DVD</label>
 </div>
 <div class=”form-field”>
 <input id=”submit” type=”submit”
name=”submit” value=”Send Form”>
 </div>
</fieldset>
</form>
<script type=”text/javascript”>
$(document).ready(function() {
 $(“input[name=’trim’]”).click(function(event) {
 if ($(this).val() == “deluxe”) {
$(“input[name=’option’]”).attr(“checked”,true);
 } else if ($(this).val() == “plain”) {
 $(“input[name=’option’]”).attr(“checked”,false);
 }
 });
 $(“input[name=’option’]”).click(function(event) {
 $(“#custom”).attr(“checked”,”checked”);
 });
});
</script>
</body>
</html>
```

Perhatikan bahwa *Cascading Style Sheet* (CSS) daftar sebelumya. HTML sangat mudah, sejauh ia mengatur bentuk yang kita lihat pada Gambar 4.4. JavaScript adalah tempat hal-hal menjadi menarik. Hal pertama yang dilakukan JavaScript adalah memasukkan semuanya ke dalam fungsi ready(). kita telah melihatnya berkali-kali; tidak perlu penjelasan lebih lanjut di sana. Hal kedua dalam JavaScript adalah selector yang dilampirkan ke event click(). Selector ini sedikit lebih kompleks daripada yang kita lihat sebelumnya:

```
$(“input[name=’trim’]”)
```

Selector itu mencari semua elemen <input> pada halaman tetapi kemudian menggunakan filter untuk mendapatkan hanya elemen input yang memiliki nama "trim". Dalam hal ini, elemen tersebut sesuai dengan tombol radio pada formulir. Perhatikan di sini penggunaan dua tanda kutip yang berbeda. Selektor keseluruhan diapit oleh tanda kutip ganda sedangkan kata trim diapit oleh tanda kutip tunggal. kita melakukan ini karena jika tidak, jQuery akan menjadi sangat bingung dan berpikir kita bermaksud menutup selector. Selector dirantai ke fungsi click() yang menangani kejadian klik. Dalam fungsi click(), nilai item yang diklik diperiksa:

```
if ($(this).val() == “deluxe”) {
```

Itu dilakukan melalui selektor $(this) dan fungsi val(). Jika nilainya "deluxe", maka semua kotak centang dicentang dengan baris ini:
$(“input[name=’option’]”).attr(“checked”,true);

Baris itu lagi menggunakan selector dan filter, tetapi kali ini mendapatkan semua kotak centang dengan nama "opsi". Kotak centang yang dipilih kemudian dicentang, berkat fungsi attr() yang dijelaskan di bab sebelumnya. Lain jika kemudian digunakan untuk melihat apakah tombol radio Biasa dipilih:

else if ($(this).val() == “plain”) {

Jika tombol radio itu dipilih, maka kotak centang 'opsi' tidak dicentang.

$(“input[name=’option’]”).attr(“checked”,false);

Setelah event handler klik ditutup, yang lain dibuat. Kali ini, pawang terhubung ke kotak centang:

$(“input[name=’option’]”).click(function(event) {

Jika seseorang mengklik salah satu kotak centang, apakah akan mencentangnya atau menghapus centang, kita harus mengaktifkan tombol radio "custom". Itu dicapai dengan daftar ini:

$(“#custom”).attr(“checked”,”checked”);

Saat memilih Deluxe, kotak centang akan secara otomatis dicentang dan jika kita memilih Biasa, kotak tersebut tidak akan dicentang. Mengklik salah satu kotak centang satu per satu menghasilkan tombol radio Kustom menjadi dipilih. Meskipun contoh ini menunjukkan cara bekerja dengan formulir dan peristiwa klik, kita sebenarnya dapat melampirkan pengendali peristiwa klik ke elemen apa pun di halaman. Lihat http://api.jquery.com/click untuk informasi lebih lanjut tentang event handler klik di jQuery.

Menonton gerakan mouse

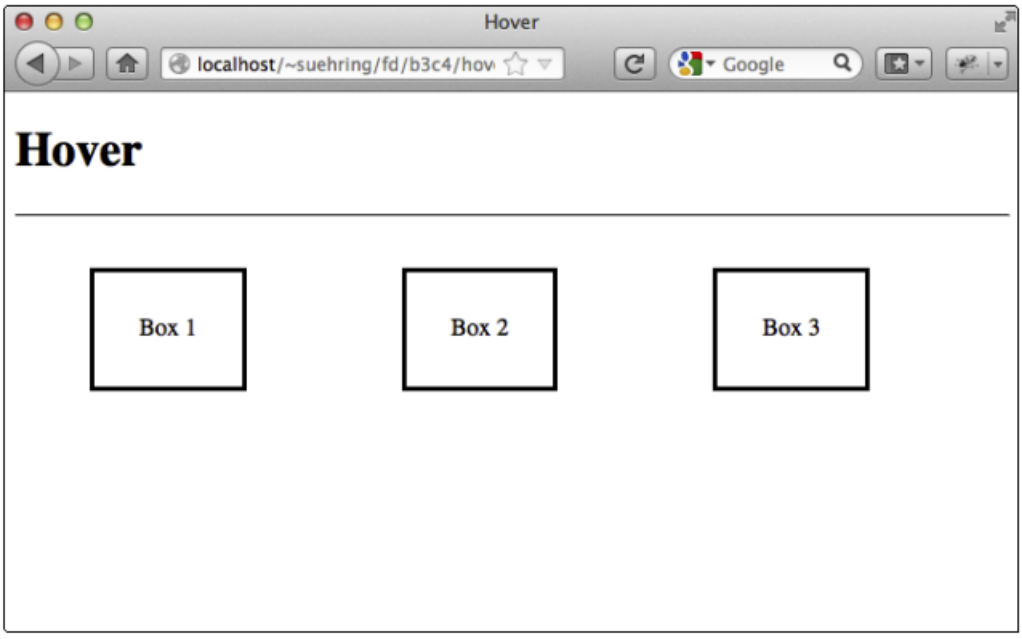
Ada beberapa item menarik seputar pergerakan mouse atau alat penunjuk. Misalnya, kita dapat mengubah elemen saat mouse melayang di atasnya atau saat mouse meninggalkan elemen. Bagian ini menunjukkan event handler hover, yang menangani hover over dan saat mouse meninggalkan elemen. kita dapat meniru pengendali acara hover melalui CSS meskipun dukungan untuk pseudoclass CSS :hover tidak tersedia di browser lama.

Contoh 3 Daftar Membuat Tiga Kotak dalam HTML

```
<!doctype html>
<html>
<head>
<title>Hover</title>
<script type=”text/javascript”
```

```
 src=”http://ajax.aspnetcdn.com/ajax/jQuery/jquery1.8.1.min.js”>
</script>
<style type=”text/css”>
.box {
 margin: 50px;
 padding: 30px;
 width: 50px;
 border: 3px solid black;
}
.colorBox {
 background-color: #abacab;
}
</style>
</head>
<body>
<h1>Hover</h1>
<hr>
<br />
<br />
<br />
<span class=”box”>Box 1</span>
<span class=”box”>Box 2</span>
<span class=”box”>Box 3</span>
</body>
</html>
```

Halaman yang dilihat di browser akan terlihat seperti gambar dibawah ini:



Gambar 9.5 HTML untuk efek hover.

Untuk efek hover, kita menambahkan JavaScript agar ketika sebuah kotak diarahkan dengan mouse, warna latar belakang berubah. Untuk membuat efek ini, JavaScript berikut digunakan di dalam halaman, di lokasi biasanya tepat di atas tag penutup </body>.

```
<script type=”text/javascript”>
$(document).ready(function() {
 $(“.box”).hover(
 //Hover over
 function() {
 $(this).addClass(“colorBox”);
 },
 //Hover out
 function() {
 $(this).removeClass(“colorBox”);
 }
 );
});
```

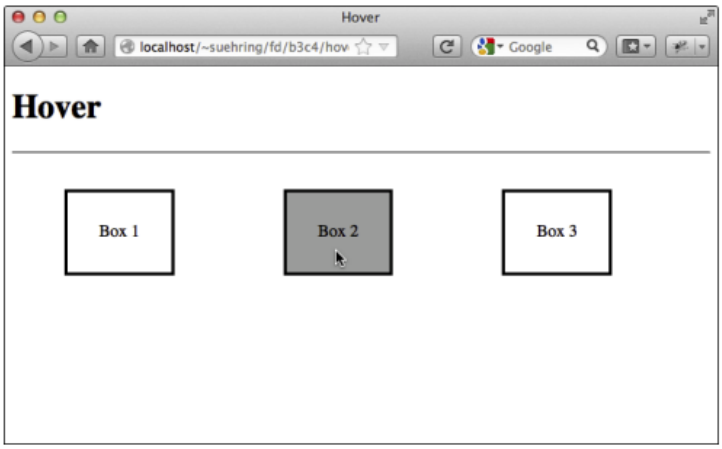
Kode ini menempatkan semuanya di fungsi jQuery ready(), seperti biasa. Selanjutnya, semua item dengan kelas "box" dipilih dan fungsi hover() dirantai ke mereka:

```
$(“.box”).hover(
```

Fungsi hover() membutuhkan dua argumen: apa yang harus dilakukan saat hover berlaku dan apa yang harus dilakukan saat hover selesai. Jadi masing-masing fungsi tersebut dibuat. Bagian pertama menambahkan kelas yang disebut "colorBox", yang hanya mengubah warna latar belakang menjadi abu-abu. Fungsi kedua, diterapkan saat mouse bergerak keluar dari elemen yang dipilih, menghapus kelas “colorBox”, seperti yang ditunjukkan di sini.

```
//Hover over
 function() {
 $(this).addClass(“colorBox”);
 },
 //Hover out
 function() {
 $(this).removeClass(“colorBox”);
 }
 );
```

Hasilnya, dengan mouse melayang di atas elemen berlabel Kotak 2, ditunjukkan pada Gambar dibawah ini. Kotak berubah menjadi abu-abu (#abacab).



Gambar 9.6 Melayang di atas elemen.

9.4 REAKSI PERISTIWA KEYBOARD

Sama seperti kita dapat bereaksi terhadap peristiwa mouse, kita juga dapat bereaksi terhadap peristiwa keyboard. Hal-hal seperti menonton ketika tombol tertentu ditekan, atau lebih umum, hanya menghitung berapa kali tombol ditekan, semuanya dapat dilakukan dengan JavaScript. Bagian ini melihat dua contoh JavaScript untuk bereaksi terhadap kejadian keyboard.

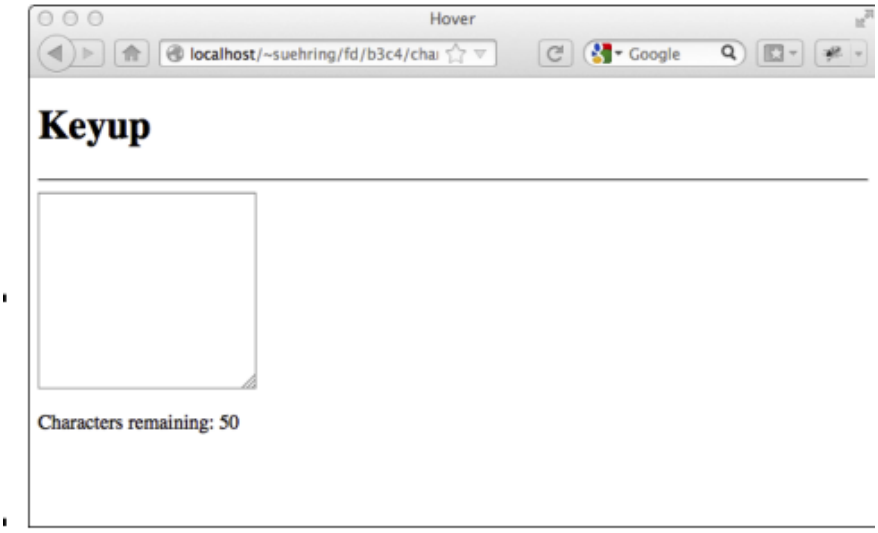
Menghitung karakter

Jika kita pernah menggunakan Twitter, kita telah melihat contoh kotak teks yang menghitung mundur saat kita mengetik. Banyak formulir kontak juga menyertakan fungsi serupa, di mana kita dapat mengetik pesan hanya hingga sejumlah karakter tertentu. Daftar 4 menunjukkan beberapa contoh HTML untuk membuat kotak teks kecil, yang disebut area teks dalam bahasa HTML.

Contoh 4 Daftar Membuat textarea HTML

```
<!doctype html>
<html>
<head>
<title>Hover</title>
<script type=”text/javascript”
 src=”http://ajax.aspnetcdn.com/ajax/jQuery/jquery1.8.1.min.js”>
</script>
</head>
<body>
<h1>Keyup</h1>
<hr>
<form action=”#”>
<textarea rows=”10” cols=”20” id=”message” name=”message”>
</textarea>
<p>Characters remaining: <span id=”remaining”>50</span></p>
</form>
</body>
</html>
```

Halaman yang dilihat di browser terlihat seperti gambar berikut ini.



Gambar 9.7 Area teks kecil untuk input

Menambahkan JavaScript untuk menghitung karakter terlihat seperti ini:

```
<script type=”text/javascript”>
$(document).ready(function() {
 var maxCharacters = 50;
 $(“#message”).on(“keyup”,function() {
 var currentVal = $(“#message”).val().length;
 var totalRemaining = maxCharacters - currentVal;
 $(“#remaining”).text(totalRemaining);
 });
});
</script>
```

JavaScript ini pertama-tama menetapkan variabel dengan jumlah karakter maksimum yang diizinkan, 50. Kemudian elemen dengan ID "message" dipilih. Sebuah event handler dilampirkan ke elemen yang dipilih. Event handler dilampirkan dengan fungsi on(), yang merupakan event handler generik. Penangan acara yang kita lihat sejauh ini sangat umum sehingga orang-orang di jQuery membuat fungsi khusus untuk menanganinya. Jadi hal-hal seperti submit(), click(), dan hover() semuanya memiliki acaranya sendiri. Namun, semua peristiwa lain dilampirkan menggunakan fungsi on(). Argumen pertama untuk fungsi on() adalah nama acara yang harus diperhatikan. Dalam hal ini, kita ingin menonton acara "keyup" untuk mengetahui kapan tombol ditekan dan kemudian dilepaskan.

Hal pertama yang kita lakukan setelah acara keyup diaktifkan adalah menghitung jumlah karakter di bidang teks. Itu dicapai dengan baris kode ini:

```
var currentVal = $(“#message”).val().length;
```

Sekarang kita tahu karakter maksimum yang diizinkan, 50, dan kita tahu jumlah karakter saat ini di bidang. Selanjutnya: matematika. kita perlu mengurangi jumlah karakter saat ini dalam

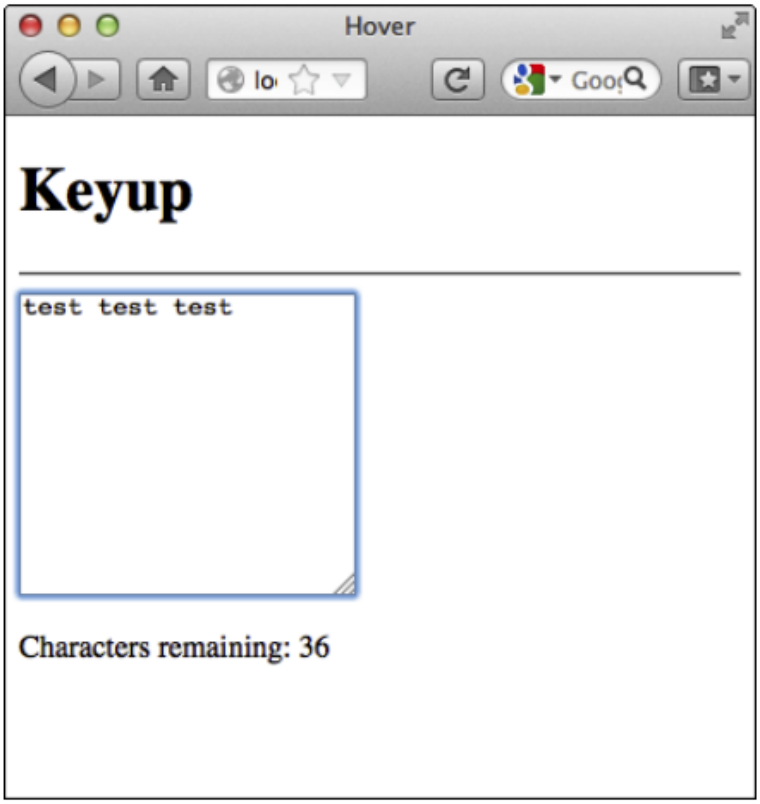
variabel currentVal dari karakter maksimum yang diizinkan dalam variabel maxCharacters. Hasilnya akan ditempatkan dalam variabel baru yang disebut totalRemaining.

var totalRemaining = maxCharacters - currentVal;

Hal terakhir yang harus dilakukan adalah menempatkan nilai tersebut ke dalam elemen yang menunjukkan karakter yang tersisa:

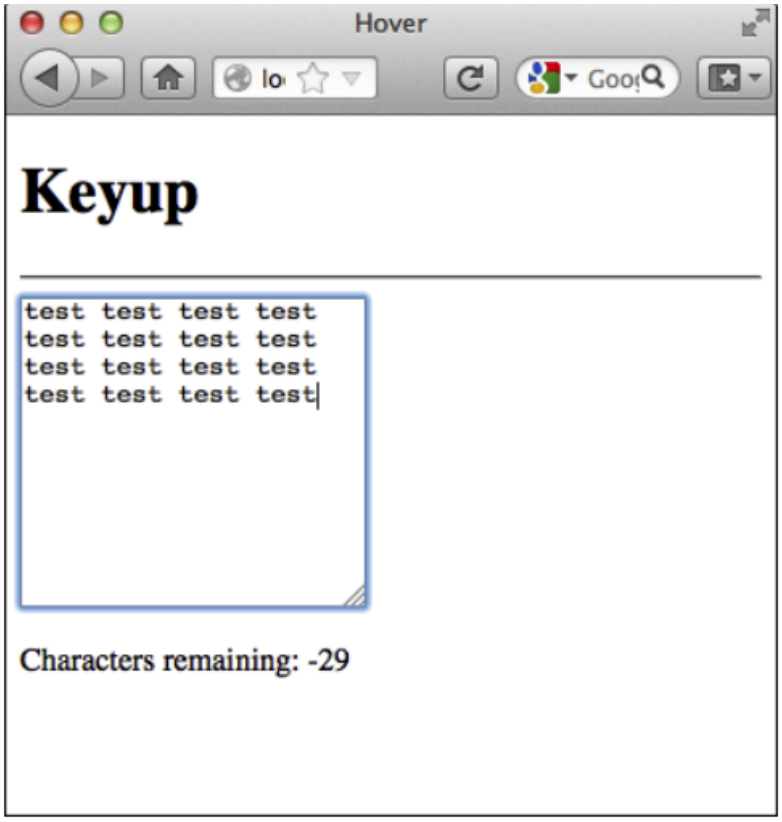
$("#remaining").text(totalRemaining);

Sekarang ketika kita mengetik ke dalam textarea, jumlah karakter yang tersisa dihitung mundur. Gambar dibawah ini menunjukkan sebuah contoh.



Gambar 9.8 Menampilkan karakter yang tersisa

Sekarang kita tahu bagaimana mereka melakukan ini di Twitter! Namun, jika kita mengetik banyak ke dalam formulir, kita akan melihat bahwa kita benar-benar dapat terus mengetik melewati 50 karakter. Penghitung akan masuk ke angka negatif.



Gambar 9.9 Memasukkan karakter yang melewati batas

Mencegah input karakter

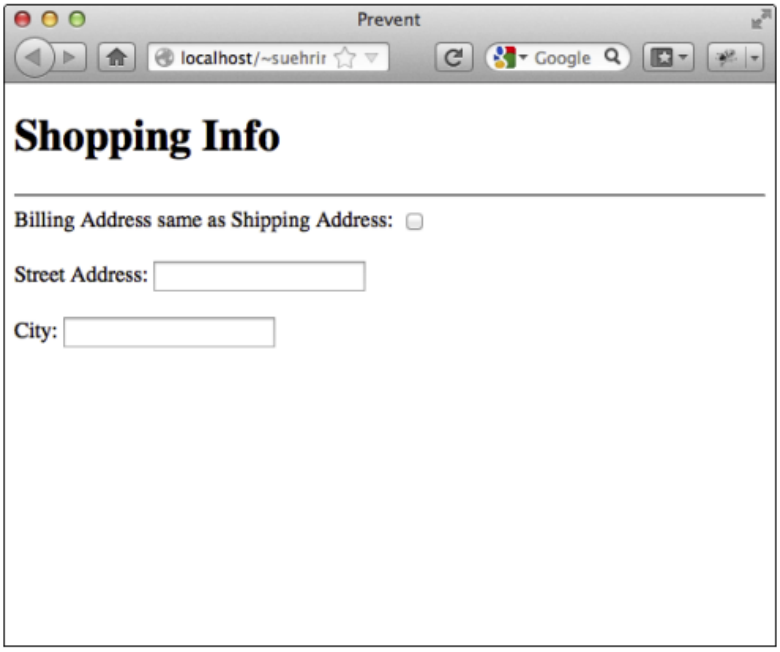
Anda dapat menggunakan JavaScript untuk menonaktifkan bidang formulir. Misalnya, banyak situs menggunakan kombinasi alamat pengiriman dan penagihan di mana pengguna mengklik kotak centang untuk menunjukkan bahwa alamat penagihan sama dengan alamat pengiriman. Daftar dibawah ini menunjukkan cuplikan HTML untuk halaman tersebut.

Contoh 5 Daftar kode untuk Halaman Info Pengiriman

```
<!doctype html>
<html>
<head>
<title>Prevent</title>
<script type=”text/javascript”
 src=”http://ajax.aspnetcdn.com/ajax/jQuery/jquery1.8.1.min.js”>
</script>
</head>
<body>
<h1>Shopping Info</h1>
<hr>
<form action=”#”>
Billing Address same as Shipping Address:
<input type=”checkbox” name=”billingAddress”
id=”billingAddress”>
<br />
<br />
```

```
Street Address:
<input class=”baddr” type=”text” name=”street” id=”street”>
<br />
<br />
City:
<input class=”baddr” type=”text” name=”city” id=”city”>
</form>
</body>
</html>
```

Halaman yang dilihat di browser terlihat seperti gambar dibawah ini.



Gambar 9.10 Cuplikan untuk halaman info penagihan.

Pada kenyataannya, halaman info penagihan akan menangkap lebih banyak informasi, seperti negara bagian dan kode pos sebagai permulaan, tetapi ini memberi kita sedikit gambaran tentang jenis halaman yang kita siapkan untuk contoh ini. Menambahkan JavaScript untuk menonaktifkan kotak teks tersebut terlihat seperti ini:

```
$(“#billingAddress”).click(function() {
 if ($(“#billingAddress”).attr(“checked”) ==
“checked”) {
 $(“.baddr”).val(“”);
 $(“.baddr”).attr(“disabled”,”disabled”);
 } else if ($(“#billingAddress”).attr(“checked”) ==
undefined) {
 $(“.baddr”).removeAttr(“disabled”);
 }
});
})
```

Kode dimulai dengan fungsi ready(), tentu saja. Setelah itu, clickevent handler ditambahkan ke kotak centang. Jika kotak centang billingAddress dicentang, maka nilai dari bidang formulir akan dihapus dan bidang formulir tersebut dinonaktifkan. Jika kotak centang billingAddress tidak dicentang, maka atribut yang dinonaktifkan akan dihapus, berkat fungsi removeAttr(). Perhatikan dalam contoh ini penggunaan kelas pada bidang teks untuk dinonaktifkan. Menggunakan kelas, yang disebut "baddr" untuk contoh ini, memudahkan pengelompokan mereka untuk selector jQuery.

BAB 10
TROUBLESHOOT PROGRAM JAVASCRIPT

10.1 MEMPEKERJAKAN TEKNIK PEMECAHAN MASALAH JAVASCRIPT DASAR

Teknik utama untuk mengatasi masalah teknis adalah berhenti. Hentikan apa yang kita lakukan dan tetap tenang. Kami telah melihat banyak orang yang sangat pintar goyah ketika ada yang salah — dan ada yang salah. Jadi kami ulangi: Saran terbaik yang dapat kami berikan untuk pemecahan masalah adalah berhenti dan tetap tenang. Setelah kita selesai melakukannya, lihat masalah dari sudut yang berbeda dan kurangi menjadi bagian-bagian kecil. Misalnya, kita akan mengalami masalah dengan halaman web yang kita programkan. Masalahnya bisa jadi halaman tidak dimuat, halaman tidak terlihat benar, atau yang lainnya. Pertimbangkan apakah masalahnya ada pada database, dengan PHP, dengan server, dengan JavaScript, atau tidak satu pun dari itu — atau lebih dari satu.

Jika menurut kita masalahnya ada pada JavaScript, keluarkan JavaScript dari halaman sepenuhnya. Kemudian tambahkan kembali secara perlahan. Cara lain untuk memecahkan masalah JavaScript adalah dengan menambahkan fungsi alert() di berbagai tempat. Saat kita melakukan pemecahan masalah, kita dapat menambahkan komentar dalam kode untuk membantu upaya pemecahan masalah Anda. Kemudian di bab ini, kami menunjukkan kepada kita sebuah plug-in untuk Firefox yang sangat membantu dalam memecahkan masalah JavaScript.

Menambahkan peringatan

Anda telah melihat dan menggunakan fungsi alert(). Cara yang baik untuk memecahkan masalah JavaScript adalah dengan menggunakan fungsi alert() dalam kode untuk menunjukkan nilai variabel atau hanya untuk menunjukkan di mana kita berada dalam kode.

Kejadian umum dengan JavaScript adalah kita akan memiliki program yang panjang dan kita tidak akan dapat memahami mengapa program tersebut tidak berhasil. Menambahkan peringatan pada baris 1 program dapat menunjukkan kepada kita apakah itu dipanggil atau tidak. Kemudian menambahkan peringatan pada baris 50 akan menunjukkan jika program sudah sejauh itu. Jika tidak, maka kita tahu bahwa pasti ada masalah di antara baris 1 dan 50. Menambahkan peringatan adalah cara yang mudah dan efisien untuk membantu memecahkan masalah yang rumit. kita cukup menambahkan kode seperti ini:

```
alert(“Just executed this code!”)
```

Atau, jika kita perlu menunjukkan nilai variabel bernama myVariable, kita akan melakukan ini:

```
alert(myVariable);
```

Perhatikan kurangnya tanda kutip di sekitar nama variabel. Jika kita memberi tanda kutip di sekitar nama variabel, JavaScript akan menafsirkannya sebagai string lama biasa sehingga kita hanya akan melihat nama variabel dan bukan isinya. kita juga bisa menjadi mewah dan menggabungkannya:

```
alert(“The value of the variable is: “ + myVariable);
```

Kode itu tidak hanya akan menampilkan teks ramah, tetapi juga nilai dari variabel itu sendiri. Berhati-hatilah dalam menggunakan lansiran dalam struktur loop karena kita harus mengabaikan setiap peringatan secara manual di browser. Pastikan juga untuk menghapus peringatan apa pun sebelum merilis kode untuk penggunaan produksi di situs web kita yang sebenarnya. Jika tidak, pengunjung situs web akan menganggap situs tersebut sangat mengganggu.

Menggunakan komentar dalam JavaScript

Komentar membantu mendokumentasikan kode, yang dapat sangat membantu saat kita perlu memperbarui atau mengubah kode nanti. kita juga dapat menggunakan komentar untuk membantu pemecahan masalah. Komentar tidak hanya berguna untuk mendokumentasikan kode, tetapi juga dapat membantu pemecahan masalah. Misalnya, jika kita telah mengidentifikasi area kode yang bermasalah, kita dapat mengomentari bagian itu sementara untuk mengatasi masalah tersebut.

Dalam JavaScript, komentar datang dalam dua bentuk.

- //: kita dapat menggunakan garis miring ganda sebagai komentar satu baris.
- /* dan */: kita dapat menggunakan format slash-asterisk untuk komentar yang mencakup beberapa baris.

Berikut ini contoh komentar satu baris:

```
// This is a single line comment.
var myVariable = 77;
```

Dalam contoh ini, baris pertama yang dimulai dengan dua garis miring akan diabaikan oleh interpreter JavaScript tetapi itu akan membuat dan menetapkan variabel di baris berikutnya karena itu tidak dikomentari. kita dapat mengomentari baris kode. Jadi pada contoh sebelumnya, jika kita ingin mengomentari baris var myVariable = 77, kita cukup menambahkan dua garis miring di depan baris, seperti ini:

```
// var myVariable = 77;
```

Apa pun yang muncul setelah dua garis miring hingga akhir baris dianggap sebagai komentar dan diabaikan. Jika kita ingin mengomentari beberapa baris atau membuat komentar multi-baris, kita menggunakan garis miring tunggal dengan tanda bintang untuk membuka blok komentar dan kemudian tanda bintang dan garis miring untuk menutup blok. Apa pun yang muncul di antara ini akan diabaikan. Ini contohnya:

```
/*
Everything within this area is a comment and will be ignored.
This includes normal lines like this and also code lines,
like:
if (myVariable = “something”) {
 return false;
}
*/
```

Dalam contoh itu, semua kode akan diabaikan karena muncul di blok komentar. Penting untuk diperhatikan saat menggunakan komentar multi-baris bahwa kita tidak dapat membuat sarangnya. Misalnya, ini tidak valid:

```
/*
Another multi-line comment
/* A comment in a comment */
Ending the comment
*/
```

Setelah juru bahasa JavaScript menemukan kode */ akhir, ia akan menganggap bahwa komentar telah berakhir dan karena itu tidak akan tahu apa yang harus dilakukan dengan */ berikutnya yang ditemuinya. kita masih dapat menggunakan konstruksi garis miring ganda, seperti ini:

```
/*
Another multi-line comment
// A comment within a comment
Ending the comment
*/
```

JavaScript dapat dilihat oleh semua orang yang melihat sumber halaman web Anda, jadi berhati-hatilah dengan apa yang kita tampilkan di komentar. Menempatkan informasi sensitif (atau menyinggung) di komentar kode dapat membuat kita mendapat masalah!

10.2 MENGIDENTIFIKASI MASALAH JAVASCRIPT DENGAN FIREBUG

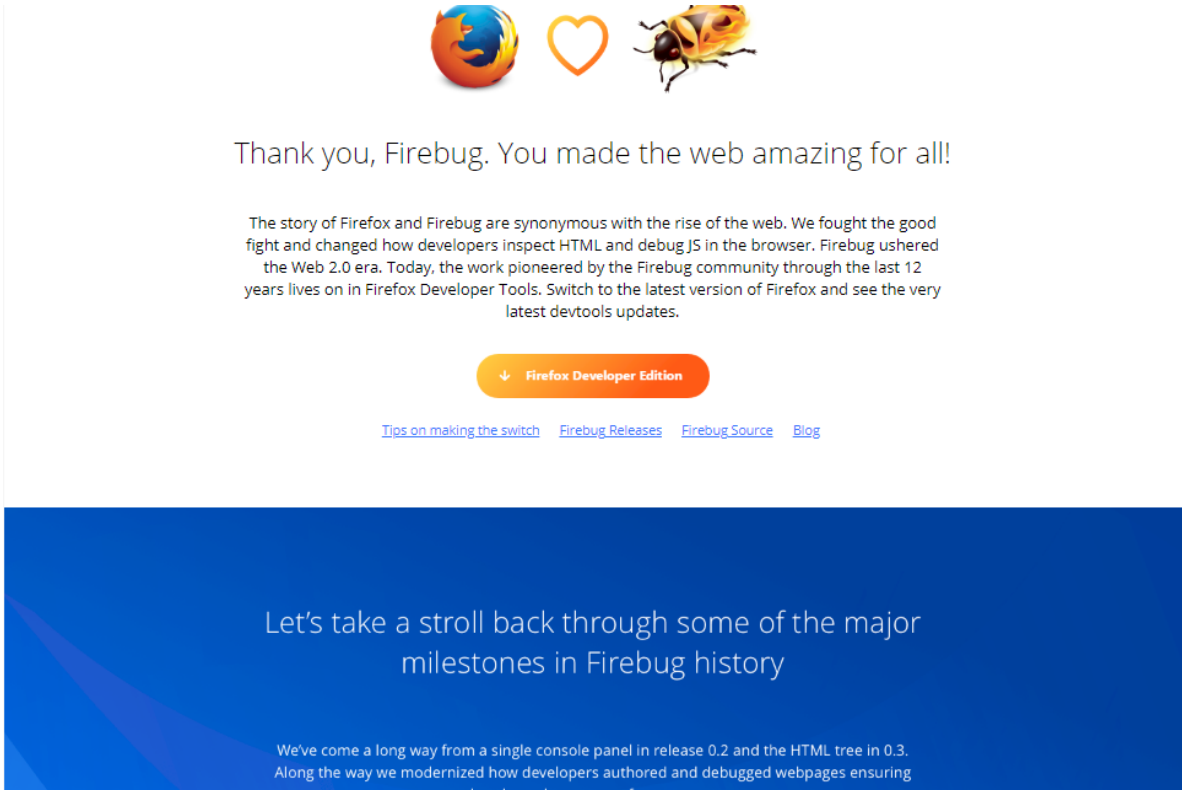
Peringatan dan komentar berfungsi dengan baik sebagai alat pemecahan masalah dalam JavaScript. Namun, alat yang sangat diperlukan untuk programmer JavaScript adalah alat yang disebut Firebug, yang merupakan add-on untuk browser web Firefox. Ini berisi alat debugging JavaScript canggih serta beberapa alat lain yang terkait dengan pengembangan web. Firebug mengidentifikasi masalah dengan kode JavaScript saat dijalankan dan membantu menemukan solusi dengan cepat. Bagian ini membahas cara menginstal Firebug dan kemudian cara menggunakannya. Kami berasumsi bahwa kita telah menginstal Firefox. Jika tidak, dapatkan di www.mozilla.org sebelum melanjutkan ke bagian berikutnya. Firebug bukan satu-satunya alat yang dapat digunakan untuk debugging. Internet Explorer memiliki alat yang disebut Alat Pengembang F12, dan Chrome juga memiliki seperangkat alat pengembangnya sendiri. Namun, Firebug cukup kuat dan mudah digunakan, jadi itulah yang kami bahas di sini.

Memasang Firebug

Anda menginstal Firebug sebagai add-on untuk Firefox. Instalasi mudah tetapi membutuhkan restart Firefox sesudahnya. Ikuti prosedur ini untuk menginstal Firebug. Meskipun, prosedur ini mungkin sedikit berubah saat kita membaca ini, keseluruhan prosesnya sama: Gunakan Firefox untuk mengunduh dan menginstal add-on Firebug. Namun, lokasi dan nama tautan dapat berubah.

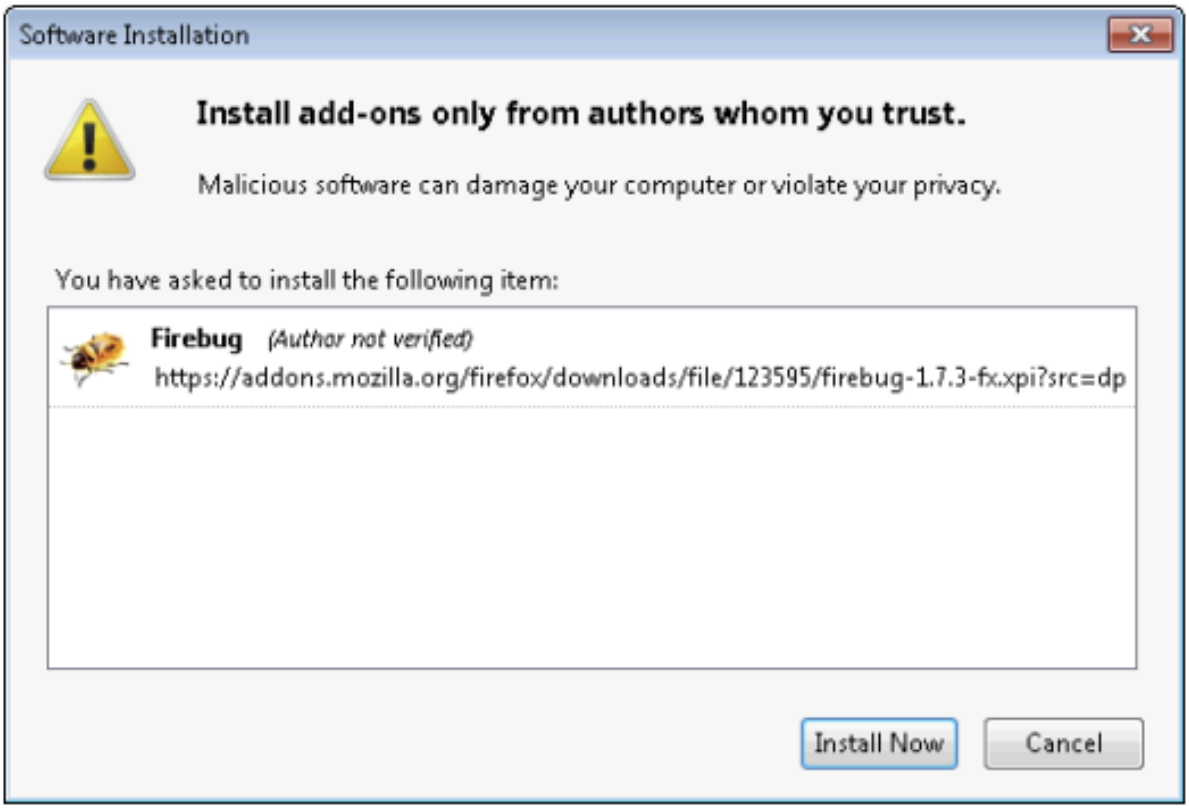
1. Buka Firefox.
2. Arahkan ke http://getfirebug.com.

3. Di halaman beranda Firebug, klik Instal Firebug (atau tautan serupa/ tombol, jika verbiage berubah pada saat kita membaca ini).
4. Pada halaman Unduhan, klik untuk mengunduh versi Firebug yang direkomendasikan. Ini biasanya akan menjadi tautan teratas untuk versi Firefox yang lebih baru. kita memulai proses pengunduhan dengan mengeklik tautan Unduh, yang akan membawa kita ke halaman Pengaya.
5. Pada halaman Pengaya Mozilla untuk Firebug, yang ditunjukkan pada Gambar dibawah ini, klik tombol Tambahkan ke Firefox.
   Firebug akan mengunduh dan menginstal.



Gambar 10.1 Halaman Pengaya Mozilla untuk Firebug.

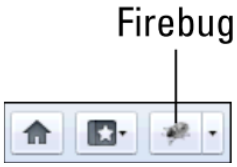
6. Pada dialog Instalasi Perangkat Lunak yang ditunjukkan pada Gambar 10.2, klik Instal Sekarang.
7. Saat kita diminta untuk me-restart Firefox, klik Restart Now.
   Firefox akan restart dan kita akan diperlihatkan halaman beranda Firebug lagi.



Gambar 10.2 install add-on Firebug

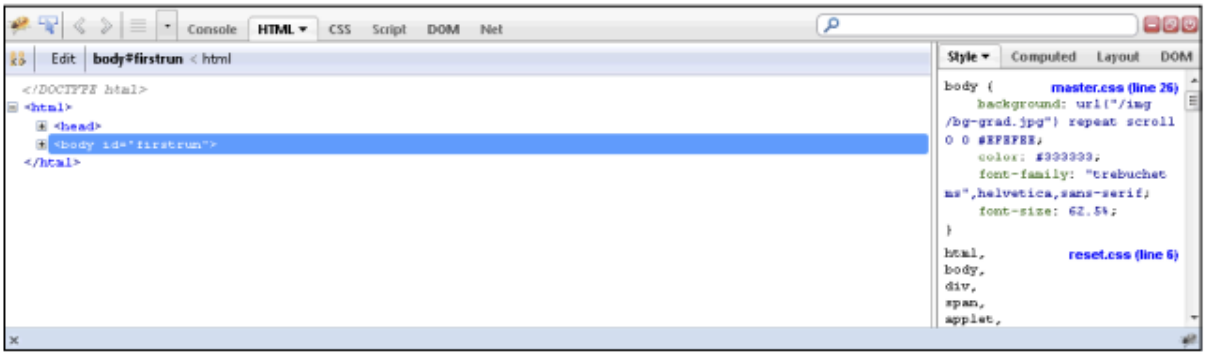
Menggunakan Firebug

Ketika Firebug dimuat, itu akan dimasukkan ke dalam bilah alat di Firefox. Ikon Firebug biasanya ditemukan di sudut kanan atas Firefox. Lihat dibawah ini untuk contoh tampilan ikon Firebug.



Gambar 10.3 Ikon Firebug di Firefox.

Mengklik ikon Firebug menampilkan konsol Firebug, yang ditunjukkan pada gambar dibawah ini, untuk halaman apa pun yang sedang kita buka.

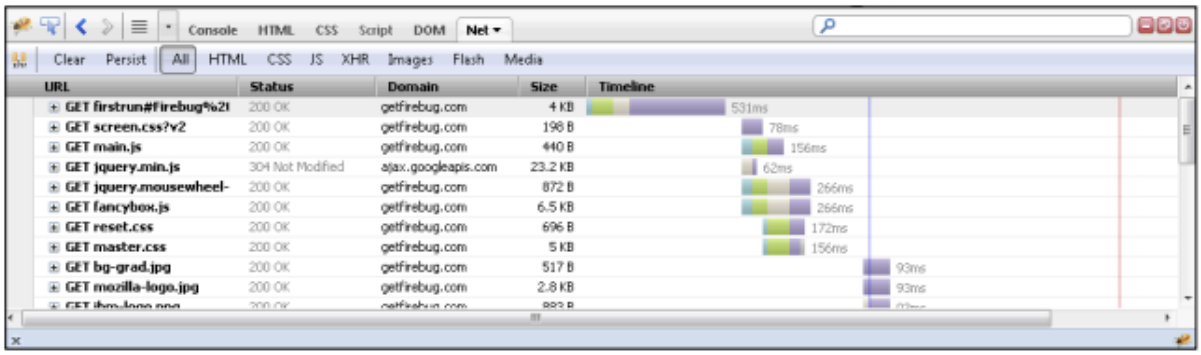


Gambar 10.4 Konsol Firebug.

Anda dapat mengklik berbagai tab dalam antarmuka Firebug untuk melihat beberapa opsi. Saat men-debug JavaScript, kita akan sering menggunakan Panel Konsol. Namun, Panel Konsol mungkin dinonaktifkan secara default, seperti yang ada pada gambar dibawah ini:

Gambar 10.5 Panel Konsol dinonaktifkan secara default di Firebug

Mengaktifkan Panel Konsol melibatkan mengklik panah bawah di sebelah kata Konsol dan memilih Diaktifkan. Saat kita melakukannya, Panel Konsol akan diaktifkan. Namun, kita perlu memuat ulang halaman agar kesalahan atau informasi lain muncul di Panel Konsol. Menekan kombinasi tombol Ctrl+R atau Command+R akan memuat ulang halaman di Firefox. Proses yang sama diperlukan untuk mengaktifkan panel lain di Firebug, seperti Panel Net. Panel Net menunjukkan pengambilan elemen pada halaman, termasuk semua JavaScript, CSS, gambar, dan elemen lainnya. Ini juga menunjukkan kode respons, yang terkadang berguna untuk menunjukkan bahwa file tertentu tidak dimuat. Panel Net juga menampilkan informasi waktu sehingga kita dapat melihat berapa lama waktu yang dibutuhkan browser untuk mengunduh berbagai elemen halaman juga. Panel Net ditunjukkan pada gambar berikut:



Gambar 10.6 Panel Net di Firebug

Jika kita menggunakan Firebug atau browser Chrome, kita juga dapat memanfaatkan cara lain untuk pemecahan masalah, yang disebut console.log. Menggunakan console.log, hasil dari apa pun yang kita debug akan ditampilkan di dalam tab Konsol area alat pengembang. Fitur console.log digunakan seperti peringatan:

```
console.log("hello world");
```

Luangkan waktu dengan Firebug untuk mengetahui kegunaannya dan bagaimana Firebug dapat membantu pemrograman JavaScript Anda. Setelah kita terbiasa dengan alat ini, itu akan menjadi sangat diperlukan bagi Anda.

BAB 11
VALIDASI JAVASCRIPT DAN PHP DAN PENANGANAN KESALAHAN

Dengan dasar yang kuat baik dalam PHP dan JavaScript, inilah saatnya untuk menyatukan teknologi ini untuk membuat formulir web yang seramah mungkin. Kami akan menggunakan PHP untuk membuat formulir dan JavaScript untuk melakukan validasi sisi klien untuk memastikan bahwa data selengkap dan sebenar mungkin sebelum dikirimkan. Validasi akhir dari input kemudian akan dilakukan oleh PHP, yang jika perlu, akan menampilkan kembali formulir tersebut kepada pengguna untuk modifikasi lebih lanjut. Dalam prosesnya, bab ini akan mencakup validasi dan ekspresi reguler di JavaScript dan PHP.

11.1 MEMVALIDASI INPUT PENGGUNA DENGAN JAVASCRIPT

Validasi JavaScript harus dianggap sebagai bantuan lebih kepada pengguna kita daripada ke situs web kita karena, seperti yang telah saya tekankan berkali-kali, kita tidak dapat mempercayai data apa pun yang dikirimkan ke server Anda, bahkan jika itu seharusnya telah divalidasi dengan JavaScript. Ini karena peretas dapat dengan mudah mensimulasikan formulir web kita dan mengirimkan data apa pun yang mereka pilih. Alasan lain kita tidak dapat mengandalkan JavaScript untuk melakukan semua validasi input kita adalah karena beberapa pengguna menonaktifkan JavaScript, atau menggunakan browser yang tidak mendukungnya. Jadi jenis validasi terbaik yang harus dilakukan dalam JavaScript adalah memeriksa bahwa bidang memiliki konten jika tidak dibiarkan kosong, memastikan bahwa alamat email sesuai dengan format yang tepat, dan memastikan bahwa nilai yang dimasukkan berada dalam batas yang diharapkan.

Dokumen Validasi.html (Bagian Satu)

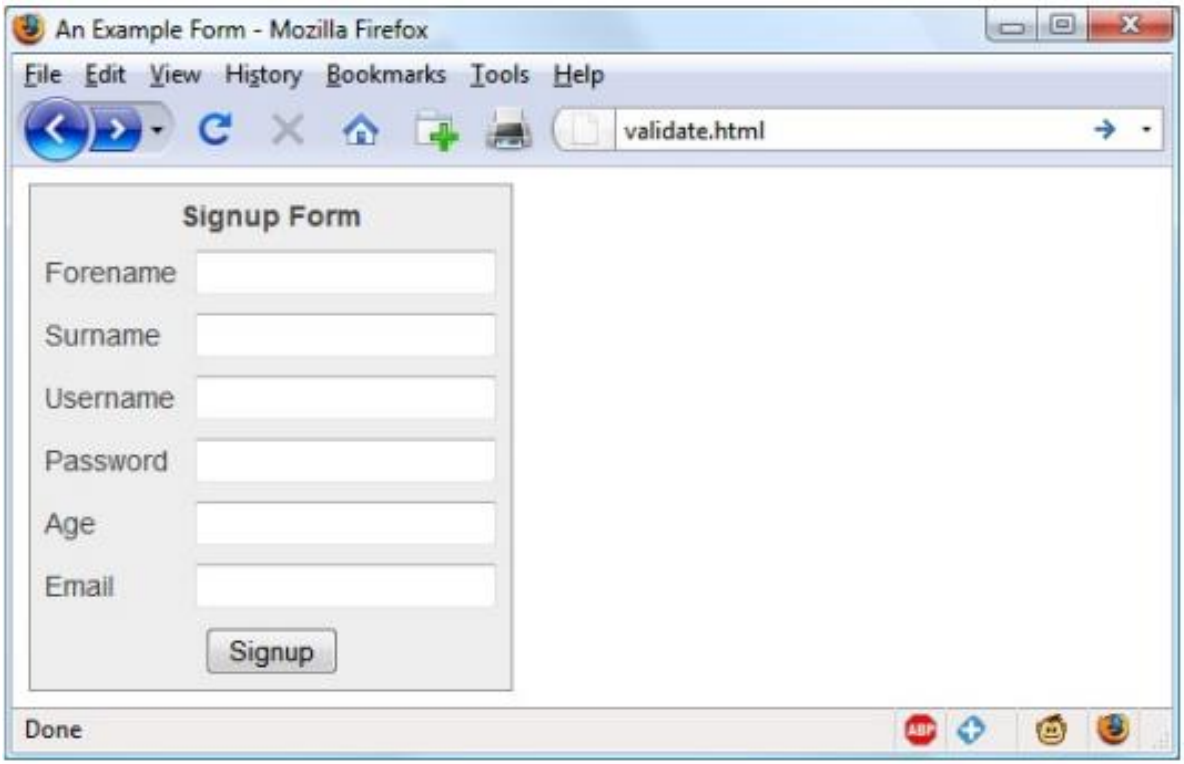
Mari kita mulai dengan formulir pendaftaran umum, umum di sebagian besar situs yang menawarkan keanggotaan atau pengguna terdaftar. Input yang diminta adalah nama depan, nama belakang, nama pengguna, kata sandi, usia, dan alamat email. Contoh 1 menyediakan template yang baik untuk formulir seperti itu.

Contoh 1. Formulir dengan validasi JavaScript (bagian satu)

```
+= validateEmail(form.email.value)
if (fail == "") return true
else { alert(fail); return false }
}
</script>
</head>
<body>
<table border="0" cellpadding="2" cellspacing="5" bgcolor="#eeeeee">
<th colspan="2" align="center">Signup Form</th>
<form method="post" action="adduser.php" onsubmit="return validate(this)">
<tr><td>Forename</td>
<td><input type="text" maxlength="32" name="forename"></td></tr>
<tr><td>Surname</td>
<td><input type="text" maxlength="32" name="surname"></td></tr>
```

```
<tr><td>Username</td>
<td><input type="text" maxlength="16" name="username"></td></tr>
<tr><td>Password</td>
<td><input type="text" maxlength="12" name="password"></td></tr>
<tr><td>Age</td>
<td><input type="text" maxlength="3" name="age"></td></tr>
<tr><td>Email</td>
<td><input type="text" maxlength="64" name="email"></td></tr>
<tr><td colspan="2" align="center"><input type="submit"
value="Signup"></td></tr>
</form>
</table>
</body>
</html>
```

Seperti berdiri, formulir ini akan ditampilkan dengan benar tetapi tidak akan memvalidasi sendiri, karena fungsi validasi utama belum ditambahkan. Meskipun demikian, simpan sebagai validasi.html, dan ketika kita memanggilnya di browser Anda, itu akan terlihat seperti Gambar 11.1.



Gambar 11.1 Output dari Contoh 1

Mari kita lihat bagaimana dokumen ini dibuat. Beberapa baris pertama mengatur dokumen dan menggunakan sedikit CSS untuk membuat formulir terlihat sedikit kurang polos. Bagian-bagian dari dokumen yang terkait dengan JavaScript datang berikutnya dan ditampilkan dalam huruf tebal. Di antara tag <script> dan </script> terdapat satu fungsi yang disebut validasi yang dengan sendirinya memanggil enam fungsi lain untuk memvalidasi setiap bidang input formulir. Kami akan segera membahas fungsi-fungsi ini. Untuk saat ini saya hanya

akan menjelaskan bahwa mereka mengembalikan string kosong jika bidang divalidasi, atau pesan kesalahan jika gagal. Jika ada kesalahan, baris terakhir skrip akan memunculkan kotak peringatan untuk menampilkannya.

Setelah melewati validasi, fungsi validasi mengembalikan nilai true; jika tidak, ia mengembalikan false. Nilai kembalian dari validasi penting, karena jika mengembalikan salah, formulir dicegah untuk dikirimkan. Ini memungkinkan pengguna untuk menutup pop up peringatan dan membuat perubahan. Jika benar dikembalikan, tidak ada kesalahan yang ditemukan di bidang formulir sehingga formulir diizinkan untuk dikirimkan.

Bagian kedua dari contoh ini menampilkan HTML untuk formulir dengan setiap bidang dan namanya ditempatkan di dalam baris tabelnya sendiri. Ini adalah HTML yang cukup sederhana, dengan pengecualian pernyataan onSubmit="return memvalidasi(ini)" di dalam tag <form> pembuka. Menggunakan onSubmit, kita dapat menyebabkan fungsi pilihan kita dipanggil saat formulir dikirimkan. Fungsi itu dapat melakukan beberapa pemeriksaan dan mengembalikan nilai benar atau salah untuk menandakan apakah formulir harus diizinkan untuk dikirimkan. Parameter this adalah objek saat ini (yaitu, formulir ini) dan diteruskan ke fungsi validasi yang baru saja dibahas. Fungsi validasi menerima parameter ini sebagai bentuk objek. Seperti yang kita lihat, satu-satunya JavaScript yang digunakan dalam HTML formulir adalah panggilan untuk kembali terkubur di atribut onSubmit. Browser dengan JavaScript yang dinonaktifkan atau tidak tersedia akan mengabaikan atribut onSubmit, dan HTML akan ditampilkan dengan baik.

Dokumen Validasi.html (Bagian Kedua

Sekarang kita sampai pada Contoh 2, satu set enam fungsi yang melakukan validasi bidang formulir yang sebenarnya. Saya sarankan kita mengetik semua bagian kedua ini dan menyimpannya di bagian <script> ... </script> pada Contoh 1, yang seharusnya sudah kita simpan sebagai validasi.html.

Contoh 2. Formulir dengan validasi JavaScript (bagian dua)

```
function validateForename(field)
{
return (field == "") ? "No Forename was entered.\n" : ""
}
function validateSurname(field)
{
return (field == "") ? "No Surname was entered.\n" : ""
}
function validateUsername(field)
{
if (field == "") return "No Username was entered.\n"
else if (field.length < 5)
return "Usernames must be at least 5 characters.\n"
else if ([^a-zA-Z0-9_-].test(field))
return "Only a-z, A-Z, 0-9, - and _ allowed in Usernames.\n"
return ""
}
function validatePassword(field)
{
```

```
if (field == "") return "No Password was entered.\n"
else if (field.length < 6)
return "Passwords must be at least 6 characters.\n"
else if (![a-z].test(field) || ! [A-Z].test(field) ||
![0-9].test(field))
return "Passwords require one each of a-z, A-Z and 0-9.\n"
return ""
}
function validateAge(field)
{
if (isNaN(field)) return "No Age was entered.\n"
else if (field < 18 || field > 110)
return "Age must be between 18 and 110.\n"
return ""
}
function validateEmail(field)
{
if (field == "") return "No Email was entered.\n"
else if (!((field.indexOf(".") > 0) &&
(field.indexOf("@") > 0)) ||
[^a-zA-Z0-9.@_-].test(field))
return "The Email address is invalid.\n"
return ""
}
```

Memvalidasi nama depan

validasiForename adalah fungsi yang cukup singkat yang menerima bidang parameter, yang merupakan nilai nama depan yang diteruskan ke sana oleh fungsi validasi. Jika nilai ini adalah string kosong, pesan kesalahan dikembalikan; jika tidak, string kosong dikembalikan untuk menandakan bahwa tidak ada kesalahan yang ditemukan. Jika pengguna memasukkan spasi di bidang ini, itu akan diterima oleh validasiForename, meskipun kosong untuk semua maksud dan tujuan. kita dapat memperbaikinya dengan menambahkan pernyataan tambahan untuk memangkas spasi dari bidang sebelum memeriksa apakah kosong, gunakan ekspresi reguler untuk memastikan ada sesuatu selain spasi di bidang, atau—seperti yang saya lakukan di sini—biarkan pengguna membuat kesalahan dan biarkan program PHP menangkapnya di server.

Memvalidasi nama keluarga

Fungsi ValidasiSurname hampir identik dengan ValidasiForename dalam kesalahan yang dikembalikan hanya jika nama keluarga yang diberikan adalah string kosong. Saya memilih untuk tidak membatasi karakter yang diizinkan di salah satu bidang nama untuk memungkinkan kemungkinan seperti karakter non-Inggris dan beraksen.

Memvalidasi nama pengguna

Fungsi ValidasiUsername sedikit lebih menarik, karena memiliki pekerjaan yang lebih rumit. Itu harus memungkinkan hanya melalui karakter a–z, A–Z, 0–9, _ dan -, dan memastikan bahwa nama pengguna setidaknya memiliki panjang lima karakter. Pernyataan if ... else dimulai dengan mengembalikan kesalahan jika bidang belum diisi. Jika bukan string kosong, tetapi panjangnya kurang dari lima karakter, pesan kesalahan lain akan ditampilkan. Kemudian

fungsi pengujian JavaScript dipanggil, meneruskan ekspresi reguler (yang cocok dengan karakter apa pun yang bukan salah satu yang diizinkan) untuk dicocokkan dengan bidang (lihat bagian Ekspresi Reguler). Jika bahkan satu karakter yang bukan salah satu karakter yang dapat diterima ditemukan, maka fungsi pengujian mengembalikan nilai true, dan selanjutnya validasiUser mengembalikan string kesalahan.

Memvalidasi kata sandi

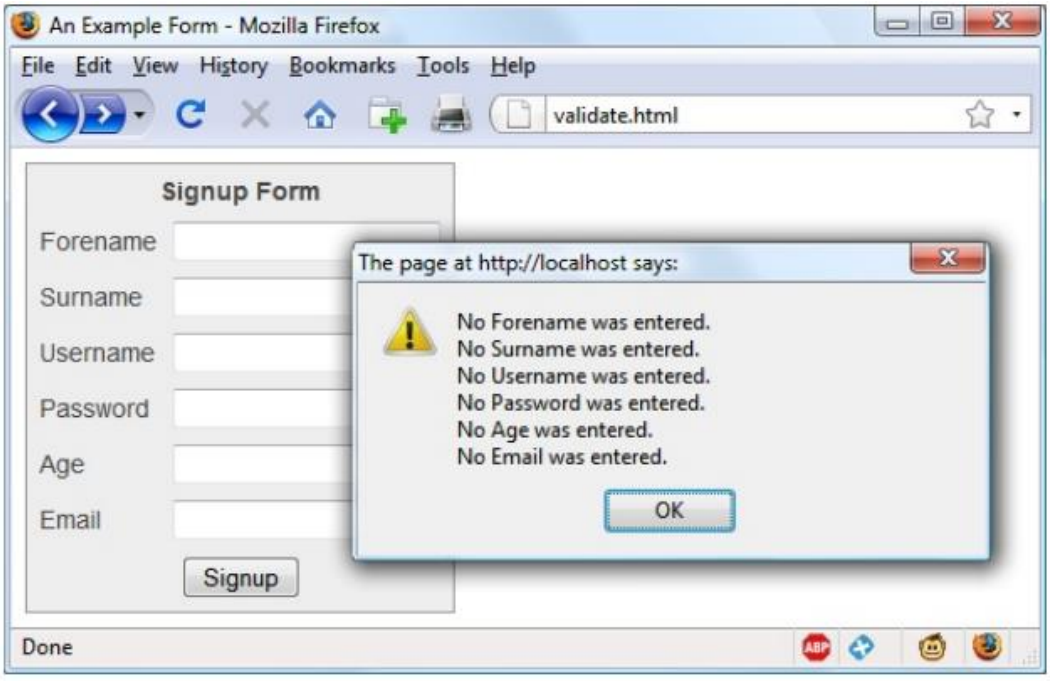
Teknik serupa digunakan dalam fungsi validasiPassword. Pertama, fungsi memeriksa apakah bidang kosong, dan jika ya, mengembalikan kesalahan. Selanjutnya, pesan kesalahan ditampilkan jika kata sandi lebih pendek dari enam karakter. Salah satu persyaratan yang kami terapkan pada kata sandi adalah kata sandi harus memiliki setidaknya satu karakter huruf kecil, huruf besar, dan numerik, sehingga fungsi pengujian dipanggil tiga kali, sekali untuk setiap kasus ini. Jika salah satu dari mereka mengembalikan false, salah satu persyaratan tidak terpenuhi dan pesan kesalahan dikembalikan. Jika tidak, string kosong dikembalikan untuk menandakan bahwa kata sandinya OK.

Memvalidasi usia

validasiAge mengembalikan pesan kesalahan jika bidang bukan angka (ditentukan oleh panggilan ke fungsi isNaN), atau jika usia yang dimasukkan lebih rendah dari 18 atau lebih besar dari 110. Aplikasi kita mungkin memiliki persyaratan usia yang berbeda atau tidak sama sekali. Sekali lagi, setelah validasi berhasil, string kosong dikembalikan.

Memvalidasi email

Dalam contoh terakhir dan paling rumit, alamat email divalidasi dengan validasiEmail. Setelah memeriksa apakah ada sesuatu yang benar-benar dimasukkan, dan mengembalikan pesan kesalahan jika tidak, fungsi tersebut memanggil fungsi JavaScript indexOf dua kali. Pertama kali pemeriksaan dilakukan untuk memastikan ada titik (.) di suatu tempat dari setidaknya karakter kedua bidang, dan pemeriksaan kedua bahwa simbol @ muncul di suatu tempat pada atau setelah karakter kedua. Jika kedua pemeriksaan tersebut terpenuhi, fungsi pengujian dipanggil untuk melihat apakah ada karakter yang tidak diizinkan muncul di bidang.



Gambar 11.2 Validasi formulir JavaScript dalam tindakan

Jika salah satu tes ini gagal, pesan kesalahan dikembalikan. Karakter yang diizinkan dalam alamat email adalah huruf besar dan huruf kecil, angka, dan karakter _, -, titik, dan @, sebagaimana dirinci dalam ekspresi reguler yang diteruskan ke metode pengujian. Jika tidak ada kesalahan yang ditemukan, string kosong akan dikembalikan untuk menunjukkan validasi yang berhasil. Pada baris terakhir, skrip dan dokumen ditutup. Gambar 11.2 menunjukkan hasil pengguna mengklik tombol Signup tanpa mengisi kolom apapun.

Menggunakan file JavaScript terpisah

Tentu saja, karena konstruksinya generik dan dapat diterapkan ke banyak jenis validasi yang mungkin kita perlukan, enam fungsi ini membuat kandidat ideal untuk pindah ke file JavaScript terpisah. Misalnya, kita dapat memberi nama file tersebut seperti validasi_fungsi.js dan memasukkannya tepat setelah bagian skrip awal pada Contoh 1, menggunakan pernyataan berikut:

```
<script src="validate_functions.js"></script>
```

11.2 EKSPRESI REGULER

Mari kita lihat sedikit lebih dekat pada pencocokan pola yang telah kita lakukan. Kami telah mencapainya menggunakan ekspresi reguler, yang didukung oleh JavaScript dan PHP. Mereka memungkinkan untuk membangun algoritma pencocokan pola yang paling kuat dalam satu ekspresi.

Mencocokkan melalui metakarakter

Setiap ekspresi reguler harus diapit oleh garis miring. Dalam garis miring ini, karakter tertentu memiliki arti khusus; mereka disebut metakarakter. Misalnya, tanda bintang (*) memiliki arti yang mirip dengan apa yang kita lihat jika kita menggunakan shell atau Command Prompt Windows (tetapi tidak persis sama). Tanda bintang berarti, "teks yang kita coba cocokkan mungkin memiliki sejumlah karakter sebelumnya—atau tidak sama sekali." Misalnya, katakanlah kita sedang mencari nama "Le Guin" dan tahu bahwa seseorang mungkin mengejanya dengan atau tanpa spasi. Karena teks ditata dengan aneh (misalnya, seseorang mungkin telah memasukkan spasi ekstra untuk membenarkan garis), kita mungkin harus mencari baris seperti:

```
The difficulty of classifying Le Guin's works
```

Jadi, kita harus mencocokkan "LeGuin", serta "Le" dan "Guin" yang dipisahkan oleh sejumlah spasi. Solusinya adalah mengikuti spasi dengan tanda bintang:

```
/Le *Guin/
```

Ada lebih banyak daripada nama "Le Guin" di baris, tapi tidak apa-apa. Selama ekspresi reguler cocok dengan beberapa bagian baris, fungsi pengujian mengembalikan nilai sebenarnya. Bagaimana jika penting untuk memastikan bahwa baris tidak berisi apa pun selain "Le Guin"? Saya akan menunjukkan cara memastikannya nanti. Misalkan kita tahu selalu ada setidaknya satu ruang. Dalam hal ini, kita dapat menggunakan tanda tambah (+), karena memerlukan setidaknya satu dari karakter sebelumnya:

```
Le +Guin
```

Pencocokan karakter kabur

Titik (.) sangat berguna, karena bisa cocok dengan apa pun kecuali baris baru. Misalkan kita mencari tag HTML, yang dimulai dengan < dan diakhiri dengan >. Cara sederhana untuk melakukannya adalah:

```
/<. *>/
```

Titik cocok dengan karakter apa pun dan * memperluasnya agar cocok dengan nol atau lebih karakter, jadi ini mengatakan, "cocokkan apa pun yang terletak di antara < dan >, bahkan jika tidak ada apa-apa." kita akan mencocokkan <>, ,
, dan seterusnya. Tetapi jika kita tidak ingin mencocokkan kasing kosong, <>, kita harus menggunakan + daripada *, seperti ini:

```
< . +>
```

Tanda plus memperluas titik untuk mencocokkan satu atau lebih karakter, dengan mengatakan, "cocokkan apa pun yang terletak di antara < dan > selama setidaknya ada satu karakter di antara mereka." kita akan mencocokkan dan , <h1> dan </h1>, dan tag dengan atribut seperti:

```
<a href="www.mozilla.org">
```

Sayangnya, tanda plus terus cocok hingga > terakhir di telepon, jadi kita mungkin akan mendapatkan:

```
<h1><b>Introduction</b></h1
```

Lebih dari satu tag! Saya akan menunjukkan solusi yang lebih baik nanti di bagian ini.

Jika kita ingin mencocokkan karakter titik itu sendiri (.), kita harus menghindarinya dengan menempatkan garis miring terbalik (\) di depannya, karena jika tidak, itu adalah metakarakter dan cocok dengan apa pun. Sebagai contoh, misalkan kita ingin mencocokkan angka floating-point 5.0. Ekspresi regulernya adalah:

```
5\.0
```

Garis miring terbalik dapat menghindari karakter meta apa pun, termasuk garis miring terbalik lainnya (jika kita mencoba mencocokkan garis miring terbalik dalam teks). Namun, untuk membuat segalanya sedikit membingungkan, kita akan melihat nanti bagaimana garis miring terbalik terkadang memberi arti khusus pada karakter berikut. Kami baru saja mencocokkan angka floating-point. Tapi mungkin kita ingin mencocokkan 5. dan juga 5.0, karena keduanya memiliki arti yang sama dengan angka floating-point. kita juga ingin mencocokkan 5,00, 5.000, dan seterusnya—sejumlah nol diperbolehkan. kita dapat melakukan ini dengan menambahkan tanda bintang, seperti yang kita lihat:

```
/5\.0*/
```

Pengelompokan melalui tanda kurung

Misalkan kita ingin mencocokkan kekuatan kenaikan unit, seperti kilo, mega, giga, dan tera. Dengan kata lain, kita ingin semua yang berikut ini cocok:

1,000
1,000,000
1,000,000,000
1,000,000,000,000
...

Tanda plus juga berfungsi di sini, tetapi kita perlu mengelompokkan string ,000 sehingga tanda plus cocok dengan semuanya. Ekspresi regulernya adalah:

```
1(,000)+
```

Tanda kurung berarti "perlakukan ini sebagai grup ketika kita menerapkan sesuatu seperti tanda tambah." 1,00,000 dan 1,000,00 tidak akan cocok karena teks harus memiliki 1 diikuti oleh satu atau lebih kelompok koma yang diikuti oleh tiga nol. Spasi setelah karakter + menunjukkan bahwa kecocokan harus berakhir saat ditemukan spasi. Tanpa itu, 1,000,00 akan salah cocok karena hanya 1.000 pertama yang akan diperhitungkan, dan sisanya ,00 akan diabaikan. Memerlukan spasi sesudahnya memastikan bahwa pencocokan akan berlanjut hingga akhir nomor.

Kelas karakter

Terkadang kita ingin mencocokkan sesuatu yang kabur, tetapi tidak terlalu luas sehingga kita ingin menggunakan titik. Kekaburan adalah kekuatan besar dari ekspresi reguler: mereka memungkinkan kita untuk menjadi tepat atau tidak jelas seperti yang kita inginkan. Salah satu fitur utama yang mendukung pencocokan fuzzy adalah pasangan tanda kurung siku, []. Itu cocok dengan satu karakter, seperti titik, tetapi di dalam tanda kurung kita memasukkan daftar hal-hal yang bisa cocok. Jika salah satu karakter tersebut muncul, teksnya cocok. Misalnya, jika kita ingin mencocokkan ejaan Amerika "abu-abu" dan ejaan Inggris "abu-abu", kita dapat menentukan:

```
gr[ae]y
```

Setelah gr dalam teks yang kita cocokkan, bisa ada a atau e. Tetapi harus ada hanya satu dari mereka: apa pun yang kita masukkan ke dalam tanda kurung cocok dengan tepat satu karakter. Kelompok karakter di dalam tanda kurung disebut kelas karakter.

Menunjukkan rentang

Di dalam tanda kurung, kita dapat menggunakan tanda hubung (-) untuk menunjukkan rentang. Salah satu tugas yang sangat umum adalah mencocokkan satu digit, yang dapat kita lakukan dengan rentang sebagai berikut:

```
[0-9]
```

Digit adalah item yang umum dalam ekspresi reguler sehingga satu karakter adalah disediakan untuk mewakili mereka: \d. kita dapat menggunakannya sebagai pengganti kurung ekspresi biasa untuk mencocokkan angka:

```
\d
```

Negosiasi

Salah satu fitur penting lainnya dari tanda kurung siku adalah negasi dari kelas karakter. kita dapat mengubah seluruh kelas karakter dengan menempatkan tanda sisipan (^) setelah tanda kurung buka. Ini artinya, “Cocokkan semua karakter kecuali yang berikut ini.” Jadi katakanlah kita ingin menemukan contoh "Yahoo" yang tidak memiliki tanda seru berikut. (Nama perusahaan secara resmi mengandung tanda seru!) kita dapat melakukannya sebagai berikut:

Yahoo[^!]

Kelas karakter terdiri dari satu karakter—tanda seru—tetapi dibalikkan oleh ^ sebelumnya. Ini sebenarnya bukan solusi yang bagus untuk masalah—misalnya, gagal jika "Yahoo" berada di akhir baris, karena itu tidak diikuti oleh apa pun, sedangkan tanda kurung harus cocok dengan karakter. Solusi yang lebih baik melibatkan pandangan ke depan yang negatif (mencocokkan sesuatu yang tidak diikuti oleh hal lain).

11.3 BEBERAPA CONTOH YANG LEBIH RUMIT

Dengan pemahaman tentang kelas karakter dan negasi, kita sekarang siap untuk melihat solusi yang lebih baik untuk masalah pencocokan tag HTML. Solusi ini menghindari melewati akhir satu tag, tetapi masih mencocokkan tag seperti dan serta tag dengan atribut seperti:

<a href="www.mozilla.org">

Salah satu solusinya adalah:

<[^>]+>

Ekspresi reguler itu mungkin terlihat seperti saya baru saja menjatuhkan cangkir teh saya di keyboard, tetapi itu benar-benar valid dan sangat berguna. Mari kita pecahkan.
Elemen-elemennya adalah:
/
Garis miring pembuka yang menunjukkan ini adalah ekspresi reguler.
<
Braket pembuka dari tag HTML. Ini sangat cocok; itu bukan metakarakter.
[^>]
Kelas karakter. ^> yang disematkan berarti "cocok dengan apa pun kecuali braket sudut penutup."
+
Mengizinkan sejumlah karakter untuk mencocokkan [^>] sebelumnya, selama setidaknya ada salah satunya.
>
Tanda kurung penutup dari tag HTML. Ini cocok persis.
/
Garis miring penutup yang menunjukkan akhir dari ekspresi reguler.

Catatan: Solusi lain untuk masalah pencocokan tag HTML adalah dengan menggunakan operasi yang tidak serakah. Secara default, pencocokan pola adalah serakah, mengembalikan kecocokan terpanjang yang mungkin. Pencocokan nongreedy menemukan kecocokan sesingkat mungkin, , tetapi ada detail lebih lanjut di http://oreilly.com/catalog/regex/chapter/ch04. Sekarang kita akan melihat salah satu ekspresi dimana fungsi validasiUsername digunakan:
[^a-zA-Z0-9_-]

_
Garis bawah.
-
Sebuah tanda hubung.
]
Braket penutup yang mengakhiri kelas karakter.
/
Garis miring penutup yang menunjukkan akhir dari ekspresi reguler.

Ada dua metakarakter penting lainnya. Mereka "menjangkar" ekspresi reguler dengan mengharuskannya muncul di tempat tertentu. Jika tanda sisipan (^) muncul di awal ekspresi reguler, ekspresi harus muncul di awal baris teks; jika tidak, itu tidak cocok. Demikian pula, jika tanda dolar ($) muncul di akhir ekspresi reguler, ekspresi harus muncul di akhir baris teks. Kami akan menyelesaikan eksplorasi dasar ekspresi reguler kami dengan menjawab pertanyaan yang diajukan sebelumnya: misalkan kita ingin memastikan tidak ada tambahan pada baris selain ekspresi reguler? Bagaimana jika kita menginginkan garis yang memiliki "Le Guin" dan tidak ada yang lain? Kita dapat melakukannya dengan mengubah ekspresi reguler sebelumnya untuk mengaitkan kedua ujungnya:

/^Le *Guin$/

11.4 RINGKASAN KARAKTER META

Tabel 11.1 menunjukkan metakarakter yang tersedia dalam ekspresi reguler.

Metakarakter	Deskripsi
/	Ekspresi reguler awal dan akhir
.	Mencocokkan karakter tunggal kecuali bari baru
*Element**	Mencocokkan *element* nol atau lebih
Element+	Mencocokkan *element* satu atau lebih
Element?	Mencocokkan *element* nol atau satu kali
[character]	Mencocokkan karakter dari karakter yang ada di dalam kurung dengan karakter yang ada di dalam kurung
[^character]	Mencocokkan satu karakter yang tidak termasuk dalam tanda kurung
(regex)	Memperlakukan regex sebagai grup untuk menghitung atau mengikuti *, +, atau ?
Left|right	Cocok dengan *left* atau *kanan*
[l-r]	Mencocokkan rentang karakter antara l dan r
^	Membutuhkan kecocokan untuk berada di awal string
$	Membutuhkan kecocokan untuk berada di ujung string

\b	Mencocokkan batas kata
\B	Cocok di mana tidak ada batas kata
\d	Cocok dengan satu digit
\D	Cocok dengan satu nondigit
\n	Cocok dengan karakter baris baru
\s	Cocok dengan karakter spasi putih
\S	Cocok dengan karakter bukan spasi
\t	Cocok dengan karakter tab
\w	Mencocokkan karakter kata (a-z, A-Z, 0-9, dan _)
\W	Mencocokkan karakter bukan kata (apa pun kecuali a-z, A-Z, 0-9, dan _)
\x	x (berguna jika x adalah metakarakter, tetapi kita benar-benar menginginkan x)
{n}	Cocok persis n kali
{n, }	Cocok n kali atau lebih
{min, max}	Cocok setidaknya *min* dan paling banyak *max*

Dengan tabel ini, dan melihat kembali ekspresi [^a-zA-Z0-9_], kita dapat melihat bahwa itu dapat dengan mudah disingkat menjadi [^\w] karena metakarakter tunggal \w (dengan huruf kecil w) menentukan karakter a-z, A-Z, 0-9, dan_.

Sebenarnya, kita bisa lebih pintar dari itu, karena metakarakter \W (dengan huruf besar W) menentukan semua karakter kecuali untuk a-z, A-Z, 0-9, dan _. Oleh karena itu, kita juga dapat menghapus karakter meta ^ dan cukup menggunakan [\W] untuk ekspresinya. Untuk memberi kita lebih banyak ide tentang bagaimana semua ini bekerja, Tabel 11.2 menunjukkan berbagai ekspresi dan pola yang cocok.

Tabel 11.2 Beberapa contoh ekspresi reguler

Metakarakter	**Deskripsi**
r	r pertama di The quick brown
rec[ei][ei]ve	Baik menerima atau menerima (tetapi juga menerima atau menerima)
rec[ei]{2}ve	Baik menerima atau menerima (tetapi juga menerima atau menerima)
rec(ei|ie)ve	Baik menerima atau menerima (tetapi tidak menerima atau menerima
cat	Kata kucing di aku suka kucing dan anjing
cat|dog	Salah satu dari kata-kata kucing atau anjing di saya suka kucing dan anjing
\.	. (\ diperlukan karena . adalah metakarakter)
5\.0*	5., 5.0, 5.00, 5.000, dst.
[a-f]	Setiap karakter a, b, c, d, e atau f
cats$	Hanya kucing terakhir di Kucing saya yang kucing ramah
^my	Hanya kucing pertama saya yang menjadi hewan peliharaan saya
\d{2,3}	Angka dua atau tiga digit (00 hingga 999)
7(,000)+	7,000; 7,000,000; 7,000,000,000; 7,000,000,000,000; dan lainnya
[\w]+	Kata apa pun dari satu atau lebih karakter
[\w]{5}	Kata lima huruf apa saja

11.5 MODIFIER UMUM

Beberapa pengubah tambahan tersedia untuk ekspresi reguler:

- /g memungkinkan pencocokan "global". Saat menggunakan fungsi ganti, tentukan pengubah ini untuk mengganti semua kecocokan, bukan hanya yang pertama.
- /i membuat ekspresi reguler tidak peka huruf besar/kecil. Jadi, alih-alih [a zA-Z], kita dapat menentukan [a-z]i atau [A-Z]i.
- /m mengaktifkan mode multiline, di mana tanda sisipan (^) dan dolar ($) cocok sebelum dan sesudah setiap baris baru dalam string subjek. Biasanya, dalam string multiline, ^ hanya cocok di awal string dan $ hanya cocok di akhir string.

Misalnya, ekspresi catg akan cocok dengan kedua kemunculan kata kucing dalam kalimat saya suka kucing dan kucing seperti saya. Demikian pula, doggi akan mencocokkan kedua kemunculan kata anjing (Anjing dan anjing) dalam kalimat Anjing seperti anjing lainnya, karena kita dapat menggunakan penentu ini bersama-sama.

11.6 MENGGUNAKAN EKSPRESI REGULER DALAM JAVASCRIPT

Dalam JavaScript, kita akan menggunakan ekspresi reguler sebagian besar dalam dua metode: uji (yang telah kita lihat) dan ganti. Sedangkan test hanya memberi tahu kita apakah argumennya cocok dengan ekspresi reguler, replace mengambil parameter kedua: string untuk mengganti teks yang cocok. Seperti kebanyakan fungsi, replace menghasilkan string baru sebagai nilai kembalian; itu tidak mengubah masukan. Untuk membandingkan kedua metode tersebut, pernyataan berikut hanya mengembalikan true untuk memberi tahu kami bahwa kata cat muncul setidaknya sekali di suatu tempat di dalam string:

```
document.write(catsi.test("Cats are funny. I like cats."))
```

Tetapi pernyataan berikut menggantikan kedua kemunculan kata kucing dengan kata anjing, mencetak hasilnya. Pencarian harus global (/g) untuk menemukan semua kemunculan, dan case-insensitive (/i) untuk menemukan Cat dengan huruf kapital:

```
document.write("Cats are friendly. I like cats.".replace(catsgi,"dogs"))
```

Jika kita mencoba pernyataan tersebut, kita akan melihat batasan penggantian: karena ini menggantikan teks dengan string yang kita perintahkan untuk digunakan, kata pertama Cats diganti dengan dog, bukan Dogs.

11.7 MENGGUNAKAN EKSPRESI REGULER DI PHP

Fungsi ekspresi reguler paling umum yang mungkin kita gunakan di PHP adalah preg_match, preg_match_all, dan preg_replace. Untuk menguji apakah kata cat muncul di mana saja dalam string, dalam kombinasi huruf besar dan kecil apa pun, kita dapat menggunakan preg_match seperti ini:

```
$n = preg_match("catsi", "Cats are crazy. I like cats.")
```

Karena PHP menggunakan 1 untuk TRUE dan 0 untuk FALSE, pernyataan sebelumnya menetapkan $n ke 1. Argumen pertama adalah ekspresi reguler dan yang kedua adalah teks

yang akan dicocokkan. Tapi preg_match sebenarnya jauh lebih kuat dan rumit, karena dibutuhkan argumen ketiga yang menunjukkan teks apa yang cocok:

```
$n = preg_match("catsi", "Cats are curious. I like cats.", $match);
echo "$n Matches: $match[0]";
```

Argumen ketiga adalah array (di sini, diberi nama $match). Fungsi ini menempatkan teks yang cocok ke dalam elemen pertama, jadi jika pencocokan berhasil, kita dapat menemukan teks yang cocok di $match[0]. Dalam contoh ini, output memberi tahu kami bahwa teks yang cocok dikapitalisasi:

```
1 Matches: Cats
```

Jika kita ingin menemukan semua kecocokan, kita menggunakan fungsi preg_match_all, seperti ini:

```
$n = preg_match_all("catsi", "Cats are strange. I like cats.", $match);
echo "$n Matches: ";
for ($j=0 ; $j < $n ; ++$j) echo $match[0][$j]." ";
```

Seperti sebelumnya, $match diteruskan ke fungsi dan elemen $match[0] diberikan kecocokan yang dibuat, tetapi kali ini sebagai subarray. Untuk menampilkan subarray, contoh ini mengulanginya dengan for loop. Saat kita ingin mengganti bagian dari string, kita dapat menggunakan preg_replace seperti yang ditunjukkan di sini. Contoh ini menggantikan semua kemunculan kata kucing dengan kata anjing, apa pun kasusnya:

```
echo preg_replace("catsi", "dogs", "Cats are furry. I like cats.");
```

11.8 MENAMPILKAN KEMBALI FORMULIR SETELAH VALIDASI PHP

Oke, kembali ke validasi formulir. Sejauh ini kita telah membuat dokumen HTML validasi.html, yang akan dikirim ke program PHP adduser.php, tetapi hanya jika JavaScript memvalidasi bidang atau jika JavaScript dinonaktifkan atau tidak tersedia. Jadi sekarang saatnya untuk membuat adduser.php untuk menerima formulir yang diposting, melakukan validasinya sendiri, dan kemudian menyajikan kembali formulir tersebut kepada pengunjung jika validasi gagal. Contoh 3 berisi kode yang harus kita ketik dan simpan (atau unduh dari situs web pendamping).

Contoh 3. Program adduser.php

```
<?php // adduser.php
// The PHP code
$forename = $surname = $username = $password = $age = $email = "";
if (isset($_POST['forename']))
$forename = fix_string($_POST['forename']);
if (isset($_POST['surname']))
$surname = fix_string($_POST['surname']);
if (isset($_POST['username']))
```

```
$username = fix_string($_POST['username']);
if (isset($_POST['password']))
$password = fix_string($_POST['password']);
if (isset($_POST['age']))
$age = fix_string($_POST['age']);
if (isset($_POST['email']))
$email = fix_string($_POST['email']);
$fail = validate_forename($forename);
$fail .= validate_surname($surname);
$fail .= validate_username($username);
$fail .= validate_password($password);
$fail .= validate_age($age);
$fail .= validate_email($email);
echo "<!DOCTYPE html>\n<html><head><title>An Example Form</title>";
if ($fail == "")
{
echo "</head><body>Form data successfully validated:
$forename, $surname, $username, $password, $age, $email.</body></html>";
// This is where you would enter the posted fields into a database,
// preferably using hash encryption for the password.
exit;
}
echo <<<_END
<!-- The HTML/JavaScript section -->
<style>
.signup {
border: 1px solid #999999;
font: normal 14px helvetica; color:#444444;
}
</style>
<script>
function validate(form)
{
fail = validateForename(form.forename.value)
fail += validateSurname(form.surname.value)
fail += validateUsername(form.username.value)
fail += validatePassword(form.password.value)
fail += validateAge(form.age.value)
fail += validateEmail(form.email.value)
if (fail == "") return true
else { alert(fail); return false }
}
function validateForename(field)
{
return (field == "") ? "No Forename was entered.\n" : ""
}
```

```
function validateSurname(field)
{
return (field == "") ? "No Surname was entered.\n" : ""
}
function validateUsername(field)
{
if (field == "") return "No Username was entered.\n"
else if (field.length < 5)
return "Usernames must be at least 5 characters.\n"
else if ([^a-zA-Z0-9_-].test(field))
return "Only a-z, A-Z, 0-9, - and _ allowed in Usernames.\n"
return ""
}
function validatePassword(field)
{
if (field == "") return "No Password was entered.\n"
else if (field.length < 6)
return "Passwords must be at least 6 characters.\n"
else if (![a-z].test(field) || ! [A-Z].test(field) ||
![0-9].test(field))
return "Passwords require one each of a-z, A-Z and 0-9.\n"
return ""
}
function validateAge(field)
{
if (isNaN(field)) return "No Age was entered.\n"
else if (field < 18 || field > 110)
return "Age must be between 18 and 110.\n"
return ""
}
ign="center">Signup Form</th>
<tr><td colspan="2">Sorry, the following errors were found<br>
in your form: <p><font color=red size=1><i>$fail</i></font></p>
</td></tr>
<form method="post" action="adduser.php" onSubmit="return validate(this)">
<tr><td>Forename</td>
<td><input type="text" maxlength="32" name="forename" value="forename">
</td></tr><tr><td>Surname</td>
<td><input type="text" maxlength="32" name="surname" value="surname">
</td></tr><tr><td>Username</td>
<td><input type="text" maxlength="16" name="username" value="username">
</td></tr><tr><td>Password</td>
<td><input type="text" maxlength="12" name="password" value="password">
</td></tr><tr><td>Age</td>
<td><input type="text" maxlength="3" name="age" value="age">
</td></tr><tr><td>Email</td>
```

```
<td><input type="text" maxlength="64" name="email" value="email">
</td></tr><tr><td colspan="2" align="center"><input type="submit"
value="Signup"></td></tr>
</form>
</table>
</body>
</html>
_END;
// The PHP functions
function validate_forename($field)
{
return ($field == "") ? "No Forename was entered<br>": "";
}
function validate_surname($field)
{
return($field == "") ? "No Surname was entered<br>" : "";
}
function validate_username($field)
{
if ($field == "") return "No Username was entered<br>";
else if (strlen($field) < 5)
return "Usernames must be at least 5 characters<br>";
else if (preg_match("[^a-zA-Z0-9_-]", $field))
return "Only letters, numbers, - and _ in usernames<br>";
return "";
}
function validate_password($field)
{
if ($field == "") return "No Password was entered<br>";
else if (strlen($field) < 6)
return "Passwords must be at least 6 characters<br>";
else if (!preg_match("[a-z]", $field) ||
!preg_match("[A-Z]", $field) ||
!preg_match("[0-9]", $field))
return "Passwords require 1 each of a-z, A-Z and 0-9<br>";
return "";
}
function validate_age($field)
{
if ($field == "") return "No Age was entered<br>";
else if ($field < 18 || $field > 110)
return "Age must be between 18 and 110<br>";
return "";
}
function validate_email($field)
{
```

```
if ($field == "") return "No Email was entered<br>";
else if (!((strpos($field, ".") > 0) &&
(strpos($field, "@") > 0)) ||
preg_match("[^a-zA-Z0-9.@_-]", $field))
return "The Email address is invalid<br>";
return "";
}
function fix_string($string)
{
if (get_magic_quotes_gpc()) $string = stripslashes($string);
return htmlentities ($string);
}
?>
```

Hasil pengiriman formulir dengan JavaScript dinonaktifkan (dan dua kolom salah diisi) ditunjukkan pada Gambar 11.3. Saya telah menempatkan bagian PHP dari kode ini (dan mengubah bagian HTML) dalam jenis huruf tebal sehingga kita dapat lebih jelas melihat perbedaan antara ini dan Contoh 1 dan 2. Jika kita menelusuri contoh ini (atau mengetiknya atau mengunduhnya dari situs web pendamping), kita akan melihat bahwa kode PHP hampir merupakan tiruan dari kode JavaScript; ekspresi reguler yang sama digunakan untuk memvalidasi setiap bidang dalam fungsi yang sangat mirip. Tapi ada beberapa hal yang perlu diperhatikan. Pertama, fungsi fix_string (tepat di akhir) digunakan untuk membersihkan setiap bidang dan mencegah upaya injeksi kode agar tidak berhasil. Juga, kita akan melihat bahwa HTML dari Contoh 1 telah diulang dalam kode PHP dalam <<<_END ... _END; struktur, menampilkan formulir dengan nilai yang dimasukkan pengunjung sebelumnya. kita melakukan ini hanya dengan menambahkan parameter nilai ekstra ke setiap tag <input> (seperti value="$forename"). Kesopanan ini sangat disarankan sehingga pengguna hanya perlu mengedit nilai yang dimasukkan sebelumnya, dan tidak perlu mengetikkan bidang lagi.

Gambar 11.3 Formulir yang direpresentasikan setelah validasi PHP gagal

Sekarang setelah kita melihat cara menyatukan semua PHP, HTML, dan JavaScript, bab berikutnya akan memperkenalkan Ajax (JavaScript dan XML Asinkron), yang menggunakan panggilan JavaScript ke server di latar belakang untuk memperbarui bagian halaman web dengan mulus. , tanpa harus mengirim ulang seluruh halaman ke server web.

BAB 12
MENGGUNAKAN AJAX

Istilah Ajax pertama kali diciptakan pada tahun 2005. Ini adalah singkatan dari Asynchronous JavaScript and XML, yang, dalam istilah sederhana, berarti menggunakan seperangkat metode yang dibangun ke dalam JavaScript untuk mentransfer data antara browser dan server di latar belakang. Contoh yang sangat baik dari teknologi ini adalah Google Maps, di mana bagian baru dari peta diunduh dari server saat dibutuhkan, tanpa memerlukan penyegaran halaman. Menggunakan Ajax tidak hanya secara substansial mengurangi jumlah data yang harus dikirim bolak-balik, tetapi juga membuat halaman web menjadi dinamis—memungkinkan mereka untuk berperilaku lebih seperti aplikasi mandiri. Hasilnya adalah antarmuka pengguna yang jauh lebih baik dan responsivitas yang lebih baik.

12.1 APA ITU AJAX?

Awal dari Ajax seperti yang digunakan saat ini dimulai dengan rilis Internet Explorer 5 pada tahun 1999, yang memperkenalkan objek ActiveX baru, XMLHttpRequest. ActiveX adalah teknologi Microsoft untuk menandatangani plug-in yang menginstal perangkat lunak tambahan ke komputer Anda. Pengembang browser lain kemudian mengikutinya, tetapi alih-alih menggunakan ActiveX, mereka semua mengimplementasikan fitur tersebut sebagai bagian asli dari penerjemah JavaScript. Namun, bahkan sebelum itu, bentuk awal Ajax telah muncul yang menggunakan bingkai tersembunyi pada halaman yang berinteraksi dengan server di latar belakang. Ruang obrolan adalah pengadopsi awal teknologi ini, menggunakannya untuk polling dan menampilkan posting pesan baru tanpa memerlukan pemuatan ulang halaman. Jadi mari kita lihat bagaimana mengimplementasikan Ajax menggunakan JavaScript.

Menggunakan XMLHttpRequest

Karena perbedaan antara implementasi browser XMLHttpRequest, kita harus membuat fungsi khusus untuk memastikan bahwa kode kita akan berfungsi di semua browser utama.

Untuk melakukan ini, kita harus memahami tiga cara membuat XMLHttpRequest obyek:

- IE 5: request = new ActiveXObject("Microsoft.XMLHTTP")
- IE 6+: request = new ActiveXObject("Msxml2.XMLHTTP")
- All others: request = new XMLHttpRequest ()

Hal ini terjadi karena Microsoft memilih untuk menerapkan perubahan dengan merilis Internet Explorer 6, sementara semua browser lain menggunakan metode yang sedikit berbeda. Oleh karena itu, kode dalam Contoh 1 akan berfungsi untuk semua browser utama yang dirilis selama beberapa tahun terakhir.

Contoh 1. Fungsi Ajax lintas-browser.

```
<script>
function ajaxRequest()
{
try // Non IE Browser?
```

```
{ // Yes
var request = new XMLHttpRequest()
}
catch(e1)
{
try // IE 6+?
{ // Yes
request = new ActiveXObject("Msxml2.XMLHTTP")
}
catch(e2)
{
try // IE 5?
{ // Yes
request = new ActiveXObject("Microsoft.XMLHTTP")
}
catch(e3) // There is no AJAX Support
{
request = false
}
}
}
return request
}
</script>
```

Anda mungkin ingat pengantar penanganan kesalahan di bab sebelumnya, menggunakan konstruksi try ... catch. Contoh 1 adalah ilustrasi sempurna dari utilitasnya, karena menggunakan kata kunci try untuk mengeksekusi perintah non-IE Ajax, dan setelah berhasil, melompat ke pernyataan pengembalian akhir, di mana objek baru dikembalikan. Jika tidak, tangkapan menjebak kesalahan dan perintah berikutnya dijalankan. Sekali lagi, setelah berhasil, objek baru dikembalikan; jika tidak, yang terakhir dari tiga perintah dicoba. Jika upaya itu gagal, maka browser tidak mendukung Ajax dan objek permintaan disetel ke false; jika tidak, objek dikembalikan. Jadi begitulah: fungsi permintaan Ajax lintas-browser yang mungkin ingin kita tambahkan ke perpustakaan fungsi JavaScript yang berguna.

Oke, jadi sekarang kita memiliki cara untuk membuat objek XMLHttpRequest, tetapi apa yang dapat kita lakukan dengan objek ini? Nah, masing-masing dilengkapi dengan seperangkat properti (variabel) dan metode (fungsi), yang dirinci dalam Tabel 12.1 dan 12.2.

Tabel 12.1 Properti objek XMLHttpRequest

Properti	**Deskripsi**
onreadystatechange	Menentukan fungsi penanganan peristiwa yang akan dipanggil setiap kali properti readyState dari suatu objek berubah.
readyState	Properti integer yang melaporkan status permintaan. Itu dapat memiliki salah satu dari nilai-nilai ini: 0 = Tidak diinisialisasi, 1 = Memuat, 2 = Dimuat, 3 = Interaktif, dan 4 = Selesai.

responseText	Data yang dikembalikan oleh server dalam format teks.
responseXML	Data yang dikembalikan oleh server dalam format XML
status	Kode status HTTP dikembalikan oleh server.
statusText	Teks status HTTP dikembalikan oleh server.

Tabel 12.2 metode objek XMLHttpRequest

Metode	**Deskripsi**
abort()	Membatalkan permintaan saat ini.
getAllResponseHeaders	Mengembalikan semua header sebagai string.
getResponseHeader(param)	Mengembalikan nilai param sebagai string.
open('method', 'url', 'asynch')	Menentukan metode HTTP yang akan digunakan (GET atau POST), URL target, dan apakah permintaan harus ditangani secara asinkron (benar atau salah).
send(data)	Mengirim data ke server target menggunakan metode HTTP yang ditentukan.
setRequestHeader('param', 'value')	Menetapkan header dengan pasangan parameter/nilai.

Properti dan metode ini memberi kita kendali atas data apa yang kita kirim ke server dan terima kembali, serta pilihan metode kirim dan terima. Misalnya, kita dapat memilih apakah akan meminta data dalam teks biasa (yang dapat menyertakan HTML dan tag lainnya) atau dalam format XML. kita juga dapat memutuskan apakah kita ingin menggunakan metode POST atau GET untuk mengirim ke server. Mari kita lihat metode POST terlebih dahulu dengan membuat sepasang dokumen yang sangat sederhana: kombinasi HTML dan JavaScript, dan program PHP untuk berinteraksi melalui Ajax dengan yang pertama. Mudah-mudahan kita akan menikmati contoh-contoh ini, karena contoh-contoh tersebut mengilustrasikan apa itu Web 2.0 dan Ajax. Dengan beberapa baris JavaScript, mereka meminta dokumen web dari server web pihak ketiga, yang kemudian dikembalikan ke browser oleh server kita dan ditempatkan di dalam bagian dokumen saat ini.

12.2 PROGRAM AJAX PERTAMA ANDA

Ketik dan simpan kode di Contoh 2 sebagai urlpost.html, tetapi jangan memuatnya ke browser Anda.

Contoh 2. urlpost.html

```
<!DOCTYPE html>
<html>
<head>
<title>AJAX Example</title>
</head>
<body style='text-align:center'>
<h1>Loading a web page into a DIV</h1>
<div id='info'>This sentence will be replaced</div>
<script>
params = "url=amazon.com/gp/aw"
request = new ajaxRequest()
```

```
request.open("POST", "urlpost.php", true)
request.setRequestHeader("Content-type",
"application/x-www-form-urlencoded")
request.setRequestHeader("Content-length", params.length)
request.setRequestHeader("Connection", "close")
request.onreadystatechange = function()
{
if (this.readyState == 4)
{
if (this.status == 200)
{
if (this.responseText != null)
{
document.getElementById('info').innerHTML =
this.responseText
}
else alert("Ajax error: No data received")
}
else alert( "Ajax error: " + this.statusText)
}
}
request.send(params)
function ajaxRequest()
{
try
{
var request = new XMLHttpRequest()
}
catch(e1)
{
try
{
request = new ActiveXObject("Msxml2.XMLHTTP")
}
catch(e2)
{
try
{
request = new ActiveXObject("Microsoft.XMLHTTP")
}
catch(e3)
{
request = false
}
}
}
```

```
return request
}
</script>
</body>
</html>
```

Mari kita lihat dokumen ini dan lihat apa yang dilakukannya, dimulai dengan enam baris pertama, yang hanya mengatur dokumen HTML dan menampilkan heading. Baris berikutnya membuat DIV dengan info ID, berisi teks Kalimat ini akan diganti secara default. Nantinya, teks yang dikembalikan dari panggilan Ajax akan disisipkan di sini. Enam baris berikutnya diperlukan untuk membuat permintaan HTTP POST Ajax. Bagian pertama menetapkan parameter variabel ke pasangan parameter=nilai, yang akan kami kirim ke server. Kemudian permintaan objek Ajax dibuat. Setelah ini, metode open dipanggil untuk mengatur objek agar membuat permintaan POST ke urlpost.php dalam mode asinkron. Tiga baris terakhir dalam grup ini mengatur header yang diperlukan server penerima untuk mengetahui bahwa permintaan POST akan datang.

12.3 PROPERTI READYSTATE

Sekarang kita sampai pada seluk beluk panggilan Ajax, yang semuanya tergantung pada properti readyState. Aspek "asynchronous" dari Ajax memungkinkan browser untuk tetap menerima input pengguna dan mengubah layar, sementara program kami menetapkan properti onreadystatechange untuk memanggil fungsi pilihan kami setiap kali readyState berubah. Dalam hal ini, fungsi inline tanpa nama (atau anonim) telah digunakan, sebagai lawan dari fungsi bernama yang terpisah. Jenis fungsi ini dikenal sebagai fungsi panggilan balik, karena dipanggil kembali setiap kali readyState berubah. Sintaks untuk mengatur fungsi panggilan balik menggunakan fungsi anonim sebaris adalah sebagai berikut:

```
request.onreadystatechange = function()
{
if (this.readyState == 4)
{
// do something
}
}
```

Jika kita ingin menggunakan fungsi bernama terpisah, sintaksnya sedikit berbeda:

```
request.onreadystatechange = ajaxCallback
function ajaxCallback()
{
if (this.readyState == 4)
{
// do something
}
}
```

Melihat Tabel 12.1, kita akan melihat bahwa readyState dapat memiliki lima nilai berbeda. Tetapi hanya satu dari mereka yang menjadi perhatian kami: nilai 4, yang mewakili panggilan Ajax yang selesai. Oleh karena itu, setiap kali fungsi baru dipanggil, ia kembali tanpa melakukan apa pun hingga readyState memiliki nilai 4. Ketika fungsi kami mendeteksi nilai itu, selanjutnya memeriksa status panggilan untuk memastikannya memiliki nilai 200, yang berarti bahwa panggilan berhasil. Jika bukan 200, pop up peringatan menampilkan pesan kesalahan yang terdapat dalam statusText.

Catatan: kita akan melihat bahwa semua properti objek ini direferensikan menggunakan this.readyState, this.status, dan seterusnya, daripada nama objek saat ini, permintaan, seperti dalam request.readyState atau request.status. Ini agar kita dapat dengan mudah menyalin dan menempelkan kode dan itu akan berfungsi dengan nama objek apa pun, karena kata kunci this selalu merujuk ke objek saat ini.

Jadi, setelah memastikan bahwa readyState adalah 4 dan statusnya adalah 200, kami menguji nilai responseText untuk melihat apakah itu berisi nilai. Jika tidak, pesan kesalahan ditampilkan di kotak peringatan. Jika tidak, HTML bagian dalam DIV diberi nilai responseText, seperti ini:

```
document.getElementById('info').innerHTML = this.responseText
```

Di baris ini, info elemen direferensikan melalui metode getElementByID, dan kemudian properti innerHTML-nya diberi nilai yang dikembalikan oleh panggilan Ajax. Setelah semua pengaturan dan persiapan ini, permintaan Ajax akhirnya dikirim ke server melalui perintah berikut, yang melewati parameter yang sudah ditentukan dalam parameter variabel:

```
request.send(params)
```

Setelah itu, semua kode sebelumnya diaktifkan setiap kali readyState berubah. Sisa dokumen adalah fungsi ajaxRequest dari Contoh 1, dan skrip penutup dan tag HTML.

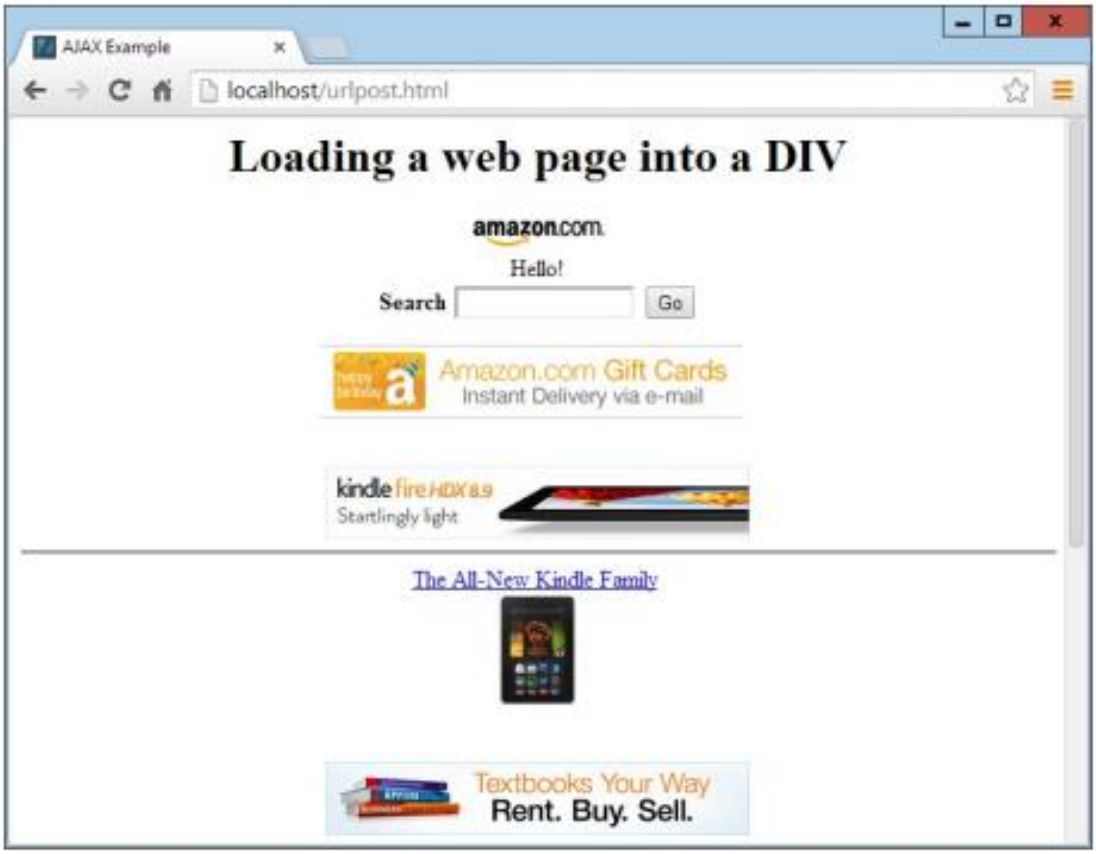
Server setengah dari proses Ajax

Sekarang kita sampai pada setengah PHP dari persamaan, yang dapat kita lihat pada Contoh 3. Ketik dan simpan sebagai urlpost.php.

Contoh 3. urlpost.php

```
<?php // urlpost.php
if (isset($_POST['url']))
{
echo file_get_contents('http://' . SanitizeString($_POST['url']));
}
function SanitizeString($var)
{
$var = strip_tags($var);
$var = htmlentities($var);
return stripslashes($var);
}
?>
```

Seperti yang kita lihat, ini singkat dan manis, dan seperti yang harus dilakukan dengan semua data yang diposting, ini juga menggunakan fungsi SanitizeString yang selalu penting. Dalam hal ini, data yang tidak bersih dapat menyebabkan pengguna mendapatkan keuntungan dari kode Anda. Program ini menggunakan fungsi PHP file_get_contents untuk memuat di halaman web pada URL yang diberikan padanya dalam variabel POST $_POST['url']. Fungsi file_get_contents serbaguna karena memuat seluruh konten file atau halaman web dari server lokal atau jauh; bahkan memperhitungkan halaman yang dipindahkan dan pengalihan lainnya. Setelah kita mengetik program, kita siap untuk memanggil urlpost.html ke browser web kita dan, setelah beberapa detik, kita akan melihat konten halaman depan ponsel Amazon dimuat ke DIV yang kami buat untuk tujuan itu. Ini tidak akan secepat langsung memuat halaman web, karena ditransfer dua kali: sekali ke server dan lagi dari server ke browser Anda. Hasilnya akan terlihat seperti Gambar 12.1. Kami tidak hanya berhasil membuat panggilan Ajax dan mendapatkan respons yang dikembalikan ke JavaScript, kami juga memanfaatkan kekuatan PHP untuk memungkinkan penggabungan objek web yang sama sekali tidak terkait. Kebetulan, jika kami mencoba menemukan cara untuk mengambil halaman web seluler Amazon secara langsung melalui Ajax (tanpa bantuan modul sisi server PHP), kami tidak akan berhasil, karena ada blok keamanan yang mencegah Ajax lintas-domain. Jadi contoh kecil ini juga menggambarkan solusi praktis untuk masalah yang sangat praktis.



Gambar 12.1 Situs web seluler Amazon telah dimuat ke dalam DIV

12.4 MENGGUNAKAN GET ALIH-ALIH POST

Seperti halnya mengirimkan data formulir apa pun, kita memiliki opsi untuk mengirimkan data kita dalam bentuk permintaan GET, dan kita akan menyimpan beberapa baris kode jika kita melakukannya. Namun, ada kelemahannya: beberapa browser mungkin menyimpan permintaan GET, sedangkan permintaan POST tidak akan pernah di-cache. kita

tidak ingin men-cache permintaan, karena browser hanya akan menampilkan ulang apa yang didapat terakhir kali alih-alih pergi ke server untuk input baru. Solusi untuk ini adalah dengan menggunakan solusi yang menambahkan parameter acak untuk setiap permintaan, memastikan bahwa setiap URL yang diminta adalah unik. Contoh 4 menunjukkan bagaimana kita akan mencapai hasil yang sama seperti dengan Contoh 2, tetapi menggunakan permintaan GET Ajax alih-alih POST.

Contoh 4. urlget.html

```
<!DOCTYPE html>
<html>
<head>
<title>AJAX Example</title>
</head>
<body style='text-align:center'>
<h1>Loading a web page into a DIV</h1>
<div id='info'>This sentence will be replaced</div>
<script>
nocache = "&nocache=" + Math.random() * 1000000
request = new ajaxRequest()
request.open("GET", "urlget.php?url=amazon.com/gp/aw" + nocache, true)
request.onreadystatechange = function()
{
if (this.readyState == 4)
{
if (this.status == 200)
{
if (this.responseText != null)
{
document.getElementById('info').innerHTML =
this.responseText
}
else alert("Ajax error: No data received")
}
else alert( "Ajax error: " + this.statusText)
}
}
request.send(null)
function ajaxRequest()
{
try
{
var request = new XMLHttpRequest()
}
catch(e1)
{
try
```

```
{
request = new ActiveXObject("Msxml2.XMLHTTP")
}
catch(e2)
{
try
{
request = new ActiveXObject("Microsoft.XMLHTTP")
}
catch(e3)
{
request = false
}
}
}
return request
}
</script>
</body>
</html>
```

Perbedaan yang perlu diperhatikan antara kedua dokumen disorot dalam huruf tebal, dan dijelaskan sebagai berikut: Tidak perlu mengirim header untuk permintaan GET. Kami memanggil metode terbuka menggunakan permintaan GET, menyediakan URL dengan string yang terdiri dari ? simbol diikuti oleh pasangan parameter/nilai url=amazon.com/gp/aw. Kami memulai pasangan parameter/nilai kedua menggunakan simbol &, kemudian menetapkan nilai parameter nocache ke nilai acak antara 0 dan satu juta. Ini digunakan untuk memastikan bahwa setiap URL yang diminta berbeda, dan oleh karena itu tidak ada permintaan yang akan di-cache.

Panggilan untuk mengirim sekarang hanya berisi parameter null, karena tidak ada parameter yang diteruskan melalui permintaan POST. Perhatikan bahwa membiarkan parameter keluar bukanlah pilihan, karena akan mengakibatkan kesalahan.

Untuk melengkapi dokumen baru ini, program PHP harus dimodifikasi untuk menanggapi permintaan GET, seperti pada Contoh 5, urlget.php

Contoh 5. urlget.php

```
<?php
if (isset($_GET['url']))
{
echo file_get_contents("http://".sanitizeString($_GET['url']));
}
function sanitizeString($var)
{
$var = strip_tags($var);
$var = htmlentities($var);
return stripslashes($var);
```

```
}
?>
```

Yang membedakan antara ini dan Contoh 3 adalah bahwa referensi ke $_POST telah diganti dengan $_GET. Hasil akhir dari pemanggilan urlget.html di browser kita sama dengan loading di urlpost.html.

Mengirim Permintaan XML

Meskipun objek yang telah kita buat disebut objek XMLHttpRequest, sejauh ini kita sama sekali tidak menggunakan XML. Di sinilah istilah Ajax sedikit keliru, karena teknologi sebenarnya memungkinkan kita untuk meminta semua jenis data tekstual, hanya salah satunya adalah XML. Seperti yang kita lihat, kami telah meminta seluruh dokumen HTML melalui Ajax, tetapi kami juga dapat meminta halaman teks, string atau angka, atau bahkan data spreadsheet. Jadi mari kita ubah dokumen contoh sebelumnya dan program PHP untuk mengambil beberapa data XML. Untuk melakukannya, pertama-tama lihat program PHP, xmlget.php, yang ditunjukkan pada Contoh 6.

Contoh 6 xmlget.php

```
<?php
if (isset($_GET['url']))
{
header('Content-Type: text/xml');
echo file_get_contents("http://".sanitizeString($_GET['url']));
}
function sanitizeString($var)
{
$var = strip_tags($var);
$var = htmlentities($var);
return stripslashes($var);
}
?>
```

Program ini telah sedikit dimodifikasi (ditampilkan dalam penyorotan tebal) untuk menampilkan header XML yang benar sebelum mengembalikan dokumen yang diambil. Tidak ada pemeriksaan yang dilakukan di sini, karena diasumsikan bahwa pemanggilan Ajax akan meminta dokumen XML yang sebenarnya. Sekarang ke dokumen HTML, xmlget.html, yang ditunjukkan pada Contoh 7.

Contoh 7. xmlget.html

```
<!DOCTYPE html>
<html>
<head>
<title>AJAX Example</title>
</head>
<body>
<h1>Loading a web page into a DIV</h1>
<div id='info'>This sentence will be replaced</div>
```

```
<script>
nocache = "&nocache=" + Math.random() * 1000000
url = "rss.news.yahoo.com/rss/topstories"
out = "";
request = new ajaxRequest()
request.open("GET", "xmlget.php?url=" + url + nocache, true)
request.onreadystatechange = function()
{
if (this.readyState == 4)
{
if (this.status == 200)
{
if (this.responseText != null
{
titles = this.responseXML.getElementsByTagName('title')
for (j = 0 ; j < titles.length ; ++j)
{
out += titles[j].childNodes[0].nodeValue + '<br>'
}
document.getElementById('info').innerHTML = out
}
else alert("Ajax error: No data received")
}
else alert( "Ajax error: " + this.statusText)
}
}
request.send(null)
function ajaxRequest()
{
try
{
var request = new XMLHttpRequest()
}
catch(e1)
{
try
{
request = new ActiveXObject("Msxml2.XMLHTTP")
}
catch(e2)
{
try
{
request = new ActiveXObject("Microsoft.XMLHTTP")
}
catch(e3)
```

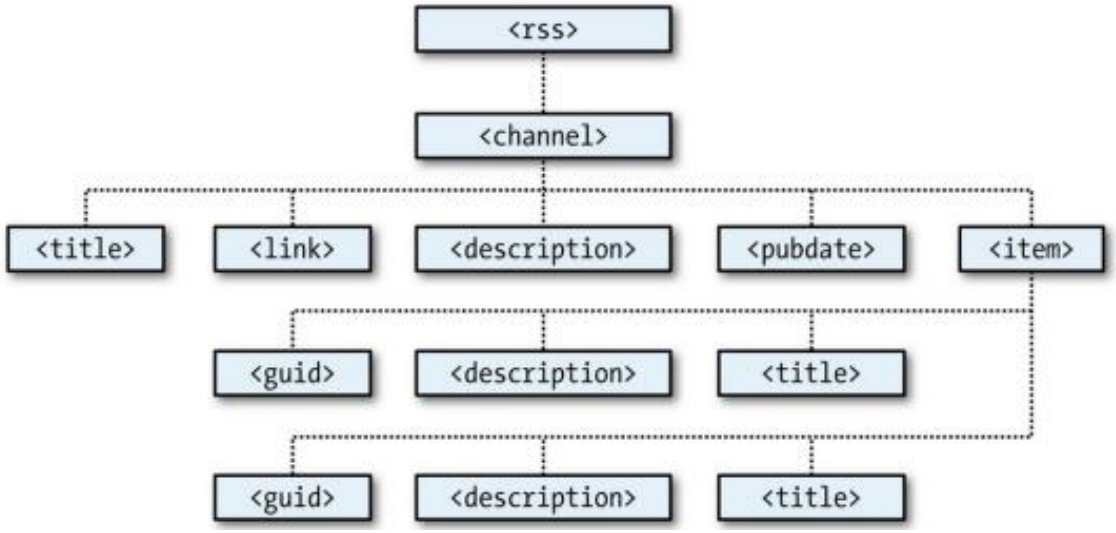
```
{
request = false
}
}
}
return request
}
</script>
</body>
</html>
```

Sekali lagi, perubahan telah disorot dalam huruf tebal, sehingga kita dapat melihat bahwa kode ini secara substansial mirip dengan versi sebelumnya, kecuali bahwa URL yang sekarang diminta, rss.news.yahoo.com/rss/topstories, berisi dokumen XML, Yahoo! Umpan Berita Teratas.

Perbedaan besar lainnya adalah penggunaan properti responseXML, yang menggantikan properti responseText. Setiap kali server mengembalikan data XML, responseXML akan berisi XML yang dikembalikan. Namun, responseXML tidak hanya berisi string teks XML: sebenarnya ini adalah objek dokumen XML lengkap yang dapat kita periksa dan urai menggunakan metode dan properti pohon DOM. Ini berarti dapat diakses, misalnya, dengan metode JavaScript getElementsByTagName.

12.5 TENTANG XML

Sebuah dokumen XML umumnya akan mengambil bentuk RSS feed yang ditunjukkan pada Contoh 18. Namun, keindahan XML adalah kita dapat menyimpan jenis struktur ini secara internal di pohon DOM (lihat Gambar 12.2) agar dapat dicari dengan cepat.



Gambar 12.2 Pohon DOM dari Contoh 18

Contoh 18. Sebuah dokumen XML

```
<?xml version="1.0" encoding="UTF-8"?>
<rss version="2.0">
<channel>
```

```
<title>RSS Feed</title>
<link>http://website.com</link>
<description>website.com's RSS Feed</description>
<pubDate>Mon, 11 May 2020 00:00:00 GMT</pubDate>
<item>
<title>Headline</title>
<guid>http://website.com/headline</guid>
<description>This is a headline</description>
</item>
<item>
<title>Headline 2</title>
<guid>http://website.com/headline2</guid>
<description>The 2nd headline</description>
</item>
</channel>
</rss>
```

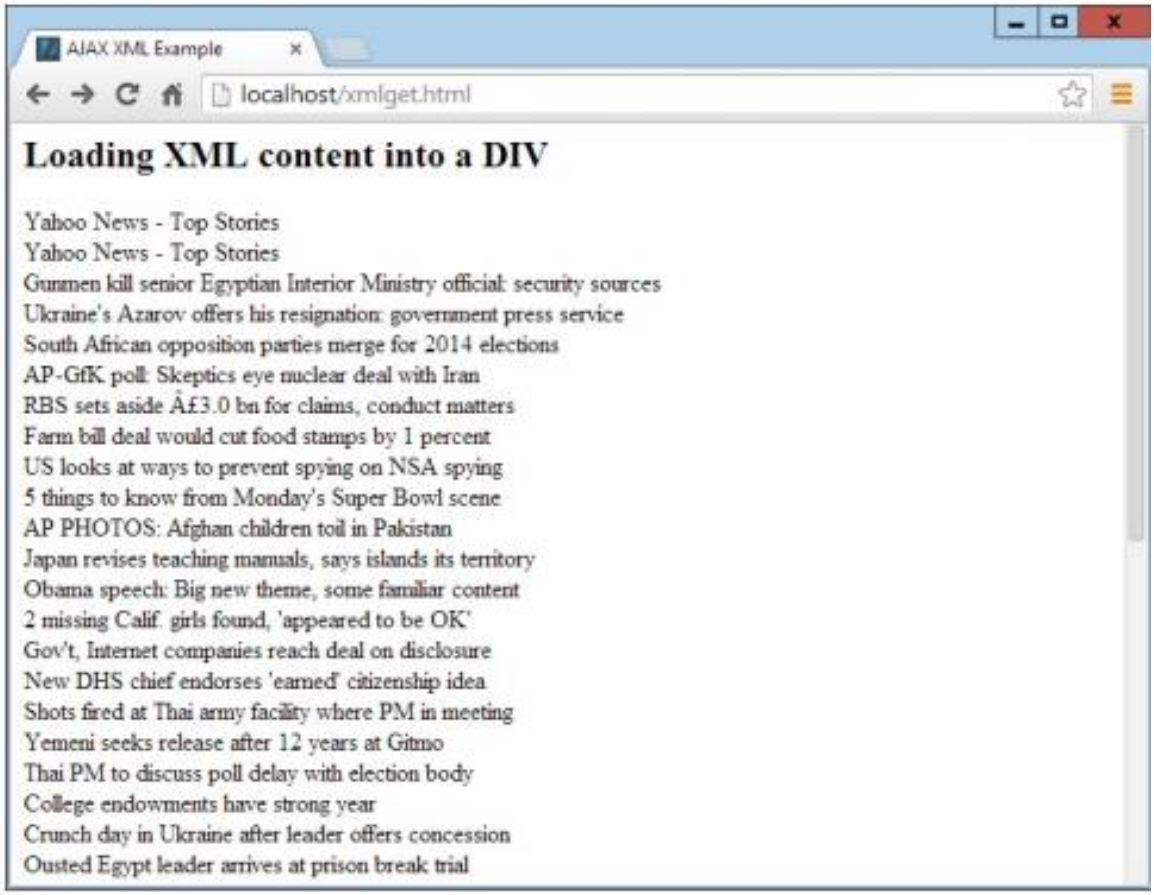
Oleh karena itu, dengan menggunakan metode getElementsByTagName, kita dapat dengan cepat mengekstrak nilai yang terkait dengan berbagai tag tanpa banyak pencarian string. Ini persis seperti yang kita lakukan pada Contoh 7, di mana perintah berikut dikeluarkan: judul = this.responseXML.getElementsByTagName('title') Perintah tunggal ini memiliki efek menempatkan semua nilai elemen judul ke dalam judul larik . Dari sana, mudah untuk mengekstraknya dengan ekspresi berikut (di mana j adalah judul untuk diakses)

```
titles[j].childNodes[0].nodeValue
```

Semua judul kemudian ditambahkan ke variabel string keluar dan, setelah semua diproses, hasilnya dimasukkan ke dalam DIV kosong di awal dokumen. Saat kita memanggil xmlget.html di browser Anda, hasilnya akan seperti Gambar 12.3.

Mengapa menggunakan XML?

Anda mungkin bertanya mengapa kita menggunakan XML selain untuk mengambil dokumen XML seperti umpan RSS. Jawaban sederhananya adalah kita tidak harus melakukannya, tetapi jika kita ingin meneruskan data terstruktur kembali ke aplikasi Ajax Anda, akan sangat merepotkan untuk mengirim teks yang sederhana dan tidak terorganisir yang memerlukan pemrosesan rumit dalam JavaScript. . Sebagai gantinya, kita dapat membuat dokumen XML dan meneruskannya kembali ke fungsi Ajax, yang secara otomatis akan menempatkannya ke dalam pohon DOM, semudah diakses seperti objek DOM HTML yang sekarang kita kenal.



Gambar 12.3 Mengambil Yahoo! Umpan berita XML melalui Ajax

12.6 MENGGUNAKAN KERANGKA KERJA UNTUK AJAX

Sekarang setelah kita mengetahui cara membuat kode rutin Ajax kita sendiri, kita mungkin ingin menyelidiki beberapa kerangka kerja gratis yang tersedia untuk membuatnya lebih mudah, karena mereka menawarkan lebih banyak fitur lanjutan. Secara khusus, saya sarankan kita memeriksa jQuery, yang mungkin merupakan kerangka kerja yang paling umum digunakan. kita dapat mengunduhnya (dan mendapatkan dokumentasi lengkap) dari http://jquery.com, tetapi perlu diketahui bahwa pada awalnya ada kurva belajar yang curam, karena kita harus membiasakan diri dengan fungsi $ yang disediakannya, yang digunakan secara luas untuk mengakses jQuery's fitur. Yang mengatakan, setelah kita memahami cara kerja jQuery, kita akan merasa bahwa itu dapat membuat pengembangan web kita lebih mudah dan lebih cepat karena banyaknya fitur siap pakai yang ditawarkannya.

BAB 13
MENGAKSES CSS DARI JAVASCRIPT

Dengan pemahaman yang baik tentang DOM dan CSS sekarang di bawah ikat pinggang Anda, kita akan belajar di bab ini cara mengakses DOM dan CSS langsung dari JavaScript, memungkinkan kita membuat situs web yang sangat dinamis dan responsif. Saya juga akan menunjukkan cara menggunakan interupsi sehingga kita dapat membuat animasi atau memberikan kode apa pun yang harus terus berjalan (seperti jam). Terakhir, saya akan menjelaskan bagaimana kita dapat menambahkan elemen baru atau menghapus elemen yang sudah ada dari DOM sehingga kita tidak perlu membuat elemen sebelumnya dalam HTML untuk berjaga-jaga jika JavaScript mungkin perlu mengaksesnya nanti.

13.1 MENINJAU KEMBALI FUNGSI GETELEMENTBYID

Dalam Bab 8, saya menyebutkan penggunaan umum karakter $ sebagai nama fungsi untuk memberikan akses yang lebih mudah ke fungsi getElementById. Faktanya, kerangka kerja utama seperti jQuery menggunakan fungsi $ baru ini, dan secara substansial memperluas fungsinya juga.

Saya juga ingin memberi kita versi yang disempurnakan dari fungsi ini, sehingga kita dapat menangani elemen DOM dan gaya CSS dengan cepat dan efisien. Namun, untuk menghindari konflik dengan framework yang menggunakan karakter $, saya cukup menggunakan huruf besar O, karena itu adalah huruf pertama dari kata Object, yang akan dikembalikan ketika fungsi dipanggil (objek yang diwakili oleh ID diteruskan ke fungsi).

fungsi O

Inilah yang terlihat seperti fungsi O tanpa tulang:

```
function O(obj)
{
return document.getElementById(obj)
}
```

Ini saja menghemat 22 karakter pengetikan setiap kali dipanggil. Tetapi saya memilih untuk memperluas fungsi sedikit dengan mengizinkan nama ID atau objek untuk diteruskan ke fungsi ini, seperti yang ditunjukkan dalam versi lengkap fungsi di Contoh 1.

Contoh 1. Fungsi O()

```
function O(obj)
{
if (typeof obj == 'object') return obj
else return document.getElementById(obj)
}
```

Jika suatu objek diteruskan ke fungsi, itu hanya mengembalikan objek itu kembali. Jika tidak, ia mengasumsikan bahwa ID dilewatkan dan mengembalikan objek yang dirujuk ID.

Tetapi mengapa saya ingin menambahkan pernyataan pertama ini, yang hanya mengembalikan objek yang diteruskan ke sana?

Fungsi S

Jawaban atas pertanyaan ini menjadi jelas ketika kita melihat fungsi mitra yang disebut S, yang memberi kita akses mudah ke properti gaya (atau CSS) suatu objek, seperti yang ditunjukkan pada Contoh 2.

Contoh 2. Fungsi S()

```
function S(obj)
{
return O(obj).style
}
```

S dalam nama fungsi ini adalah huruf pertama dari Gaya, dan fungsi melakukan tugas mengembalikan properti gaya (atau subobjek) dari elemen yang dirujuk. Karena fungsi O yang disematkan menerima ID atau objek, kita juga dapat meneruskan ID atau objek ke S. Mari kita lihat apa yang terjadi di sini dengan mengambil elemen dengan ID myobj dan mengatur warna teksnya menjadi hijau, seperti ini:

```
<div id='myobj'>Some text</div>
<script>
O('myobj').style.color = 'green'
</script>
```

Kode sebelumnya akan melakukan pekerjaan itu, tetapi jauh lebih mudah untuk memanggil fungsi S baru, seperti ini:

```
S('myobj').color = 'green
```

Sekarang perhatikan kasus di mana objek yang dikembalikan dengan memanggil O disimpan di, misalnya, objek bernama fred, seperti ini:

```
fred = O('myobj')
```

Karena cara kerja fungsi S, kita masih bisa memanggilnya untuk mengubah warna teks menjadi hijau, seperti ini:

```
S(fred).color = 'green'
```

Ini berarti apakah kita ingin mengakses objek secara langsung atau melalui ID-nya, kita dapat melakukannya dengan meneruskannya ke fungsi O atau S sesuai kebutuhan. Ingatlah bahwa ketika kita melewati sebuah objek (bukan ID), kita tidak boleh menempatkannya dalam tanda kutip.

Fungsi C

Sejauh ini saya telah memberi kita dua fungsi sederhana yang memudahkan kita mengakses elemen apa pun di halaman web, dan properti gaya apa pun dari suatu elemen.

Namun, terkadang kita ingin mengakses lebih dari satu elemen sekaligus, dan kita dapat melakukannya dengan menetapkan nama kelas CSS untuk setiap elemen tersebut, seperti contoh berikut, yang keduanya menggunakan kelas myclass:

```
<div class='myclass'>Div contents</fiv>
<p class='myclass'>Paragraph contents</p>
```

Jika kita ingin mengakses semua elemen pada halaman yang menggunakan kelas tertentu, kita dapat menggunakan fungsi C (untuk huruf pertama Kelas), yang ditunjukkan pada Contoh 3, untuk mengembalikan array yang berisi semua objek yang cocok dengan kelas nama yang disediakan.

Contoh 3 Fungsi C()

```
function C(name)
{
var elements = document.getElementsByTagName('*')
var objects = []
for (var i = 0 ; i < elements.length ; ++i)
if (elements[i].className == name)
objects.push(elements[i])
return objects
}
```

Mari kita uraikan contoh ini. Pertama, nama argumen berisi nama kelas yang kita coba ambil objeknya. Kemudian, di dalam fungsi, yang baru objek yang disebut elemen dibuat yang berisi semua elemen dalam dokumen, seperti yang dikembalikan oleh panggilan ke getElementsByTagName dengan argumen '*', yang berarti "temukan semua elemen":

```
var elements = document.getElementsByTagName('*')
```

Kemudian array baru yang disebut objek dibuat, di mana semua objek yang cocok ditemukan akan ditempatkan:

```
var objects = []
```

Selanjutnya, for loop iterasi melalui semua elemen dalam objek elemen menggunakan variabel i sebagai indeks:

```
for (var i = 0 ; i < elements.length ; ++i)
```

Setiap kali di sekitar loop, jika properti className elemen sama dengan nilai string yang diteruskan dalam nama argumen, objek didorong ke array objek:

```
if (elements[i].className == name)
objects.push(elements[i])
```

Akhirnya, setelah loop selesai, array objek akan berisi semua elemen dalam dokumen yang menggunakan nama kelas dalam nama, sehingga dikembalikan oleh fungsi:

return objects

Untuk menggunakan fungsi ini, panggil saja sebagai berikut, simpan array yang dikembalikan sehingga kita dapat mengakses setiap elemen satu per satu sesuai kebutuhan atau (lebih mungkin demikian) secara massal melalui satu lingkaran:

myarray = C('myclass')

Sekarang kita dapat melakukan apa pun yang kita suka dengan objek yang dikembalikan, seperti, misalnya, menyetel properti gaya textDecoration ke 'garis bawah', sebagai berikut:

for (i = 0 ; i < myarray.length ; ++i)
S(myarray[i]).textDecoration = 'underline

Kode ini berulang melalui objek di myarray[] dan kemudian menggunakan fungsi S untuk mereferensikan masing-masing properti gaya, menyetel properti textDecoration ke 'garis bawah'.

13.2 MEMASUKKAN FUNGSI

Saya menggunakan fungsi O dan S dalam contoh untuk sisa bab ini, karena mereka membuat kode lebih pendek dan lebih mudah diikuti. Oleh karena itu, saya telah menyimpannya dalam file OSC.js (bersama dengan fungsi C, karena saya pikir kita akan merasa sangat berguna) dapat diunduh secara bebas dari situs web pendamping. kita dapat menyertakan fungsi-fungsi ini di halaman web mana pun menggunakan pernyataan berikut — sebaiknya di bagian <head>-nya, di mana pun sebelum skrip apa pun yang bergantung pada pemanggilannya:

<script src='OSC.js'></script>

Isi OSC.js ditunjukkan pada Contoh 4.

Contoh 4. File OSC.js

```
function O(obj)
{
if (typeof obj == 'object') return obj
else return document.getElementById(obj)
}
function S(obj)
{
return O(obj).style
}
function C(name)
{
```

```
var elements = document.getElementsByTagName('*')
var objects = []
for (var i = 0 ; i < elements.length ; ++i)
if (elements[i].className == name)
objects.push(elements[i])
return objects
}
```

13.3 MENGAKSES PROPERTI CSS DARI JAVASCRIPT

Properti textDecoration yang saya gunakan dalam contoh sebelumnya mewakili properti CSS yang biasanya diberi tanda hubung seperti ini: text-decoration. Tetapi karena JavaScript menyimpan karakter tanda hubung untuk digunakan sebagai operator matematika, setiap kali kita mengakses properti CSS yang diberi tanda hubung, kita harus menghilangkan tanda hubung dan mengatur karakter segera setelahnya menjadi huruf besar. Contoh lain dari ini adalah properti font-size, yang direferensikan dalam JavaScript sebagai fontSize ketika ditempatkan setelah operator titik, seperti ini:

```
myobject.fontSize = '16pt'
```

Alternatif untuk ini adalah menjadi lebih bertele-tele dan menggunakan fungsi setAttribute, yang mendukung (dan sebenarnya membutuhkan) nama properti CSS standar, seperti ini:

```
myobject.setAttribute('style', 'font-size:16pt')
```

Perhatikan bahwa, beberapa versi Microsoft Internet Explorer yang lebih lama pilih-pilih dalam kasus tertentu tentang penggunaan nama properti CSS gaya JavaScript saat menerapkan versi aturan yang diawali -ms- khusus browser. Jika kita mengalami ini, gunakan fungsi setAttribute dan kita akan baik-baik saja.

Beberapa Properti Umum

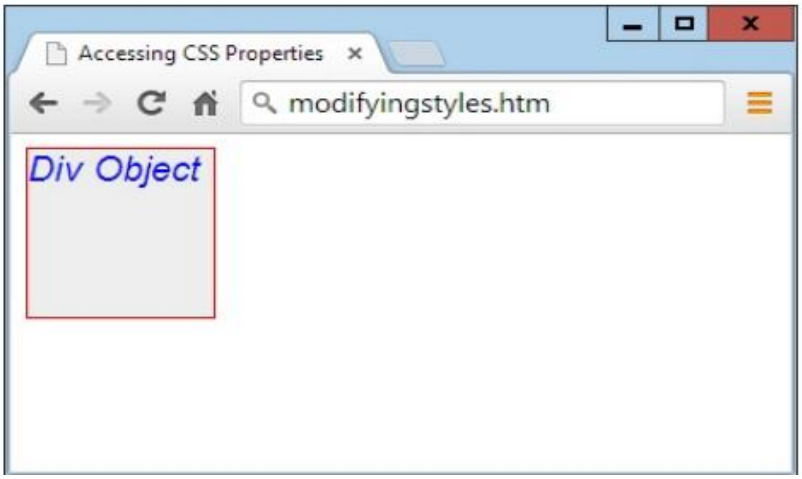
Menggunakan JavaScript, kita dapat memodifikasi properti apa pun dari elemen apa pun dalam dokumen web, dengan cara yang mirip dengan menggunakan CSS. Saya telah menunjukkan kepada kita cara mengakses properti CSS menggunakan bentuk pendek JavaScript atau fungsi setAttribute untuk menggunakan nama properti CSS yang tepat, jadi saya tidak akan membuat kita bosan dengan merinci semua ratusan properti ini. Sebaliknya, saya ingin menunjukkan kepada kita cara mengakses hanya beberapa properti CSS sebagai ikhtisar dari beberapa hal yang dapat kita lakukan. Pertama, mari kita lihat memodifikasi beberapa properti CSS dari JavaScript menggunakan Contoh 5, yang memuat tiga fungsi sebelumnya, membuat elemen , dan kemudian mengeluarkan pernyataan JavaScript dalam bagian <script> HTML, untuk memodifikasi berbagai atributnya (lihat Gambar 13.1).

Contoh 5. Mengakses properti CSS dari JavaScript

```
<!DOCTYPE html>
<html>
<head>
<title>Accessing CSS Properties</title>
<script src='OSC.js'></script>
```

```
</head>
<body>
<div id='object'>Div Object</div>
<script>
S('object').border = 'solid 1px red'
S('object').width = '100px'
S('object').height = '100px'
S('object').background = '#eee'
S('object').color = 'blue'
S('object').fontSize = '15pt'
S('object').fontFamily = 'Helvetica'
S('object').fontStyle = 'italic'
</script>
</body>
</html>
```

Anda tidak mendapatkan apa-apa dengan memodifikasi properti seperti ini, karena kita dapat dengan mudah memasukkan beberapa CSS secara langsung, tetapi segera kami akan memodifikasi properti sebagai respons terhadap interaksi pengguna—dan kemudian kita akan melihat kekuatan sebenarnya dari menggabungkan JavaScript dan CSS.



Gambar 13.1 Memodifikasi gaya dari JavaScript

Properti lainnya

JavaScript juga membuka akses ke berbagai properti lain yang sangat luas, seperti lebar dan tinggi browser dan jendela atau bingkai pop-up atau dalam browser, informasi berguna seperti jendela induk (jika ada) , dan riwayat URL yang mengunjungi sesi ini. Semua properti ini diakses dari objek jendela melalui operator periode (misalnya, nama jendela), dan Tabel 13.1 mencantumkan semuanya, bersama dengan deskripsi masing-masing.

Tabel 13.1 Properti jendela umum

Properti	**Set dan/atau kembali**
closed	Mengembalikan nilai Boolean yang menunjukkan apakah jendela telah ditutup atau tidak

closed	Menyetel atau mengembalikan teks default di bilah status jendela
document	Mengembalikan objek dokumen untuk jendela
frames	Mengembalikan array dari semua frame dan iframe di jendela
history	Mengembalikan objek sejarah untuk jendela
innerHeight	Mengatur atau mengembalikan ketinggian bagian dalam area konten jendela
innerWidth	Mengatur atau mengembalikan lebar bagian dalam area konten jendela
length	Mengembalikan jumlah bingkai dan iframe di jendela
location	Mengembalikan objek lokasi untuk jendela
name	Menetapkan atau mengembalikan nama jendela
navigator	Mengembalikan objek navigator untuk jendela
opener	Mengembalikan referensi ke jendela yang membuat jendela
outerHeight	Mengatur atau mengembalikan ketinggian luar jendela, termasuk alat dan bilah gulir
outerWidth	Mengatur atau mengembalikan lebar luar jendela, termasuk alat dan bilah gulir
pageXOffset	Mengembalikan piksel dokumen yang telah digulir secara horizontal dari kiri jendela
pageYOffset	Mengembalikan piksel dokumen yang telah digulir secara vertikal dari atas jendela
parent	Mengembalikan jendela induk dari sebuah jendela
screen	Mengembalikan objek layar untuk jendela
screenLeft	Mengembalikan koordinat x dari jendela relatif terhadap layar di semua browser terbaru kecuali Mozilla Firefox (untuk itu kita harus menggunakan screenX)
screenTop	Mengembalikan koordinat y dari jendela relatif terhadap layar di semua browser terbaru kecuali Mozilla Firefox (untuk itu kita harus menggunakan screenY)
screenX	Mengembalikan koordinat x jendela relatif terhadap layar di semua browser terbaru kecuali Opera, yang mengembalikan nilai yang salah; tidak didukung di versi IE sebelum 9
screenY	Mengembalikan koordinat y dari jendela relatif terhadap layar di semua browser terbaru kecuali Opera, yang mengembalikan nilai yang salah; tidak didukung di versi IE sebelum 9
self	Mengembalikan jendela saat ini
status	Menyetel atau mengembalikan teks di bilah status jendela
top	Mengembalikan jendela browser teratas

Ada beberapa poin yang perlu diperhatikan tentang beberapa properti ini:

- Status default dan properti status dapat disetel hanya jika pengguna telah memodifikasi browser mereka untuk mengizinkannya (sangat tidak mungkin).
- Objek sejarah tidak dapat dibaca (sehingga kita tidak dapat melihat dari mana pengunjung kita berselancar). Tapi itu mendukung properti length untuk menentukan berapa lama sejarah, dan metode mundur, maju, dan pergi untuk menavigasi ke halaman tertentu dalam sejarah.

- Saat kita perlu mengetahui berapa banyak ruang yang tersedia di jendela browser web saat ini, cukup baca nilai di window.innerHeight dan window.innerWidth. Saya sering menggunakan nilai-nilai ini untuk memusatkan peringatan pop-up dalam browser atau jendela "konfirmasi dialog".
- Objek layar mendukung properti baca availHeight, availWidth, colorDepth, height, pixelDepth, dan lebar, dan oleh karena itu bagus untuk menentukan informasi tentang tampilan pengguna.

Beberapa item informasi ini akan membantu kita memulai dan sudah memberi kita banyak hal baru dan menarik yang dapat kita lakukan dengan JavaScript. Namun, pada kenyataannya, ada jauh lebih banyak properti dan metode yang tersedia daripada yang dapat dibahas dalam bab ini. Namun, sekarang setelah kita mengetahui cara mengakses dan menggunakan properti, yang kita butuhkan hanyalah sumber daya yang mencantumkan semuanya, jadi saya sarankan kita memeriksa http://tinyurl.com/domproperties sebagai titik awal yang baik.

13.4 JAVASCRIPT SEBARIS

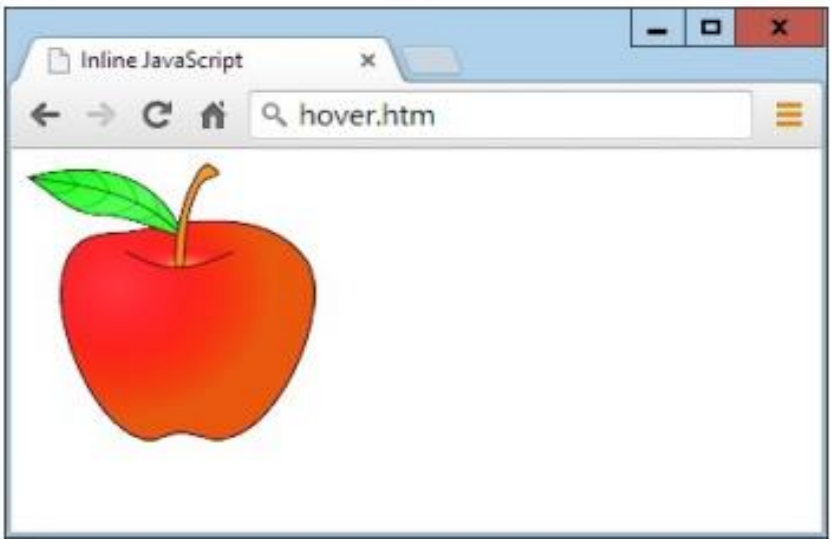
Menggunakan tag <script> bukan satu-satunya cara kita dapat mengeksekusi pernyataan JavaScript; kita juga dapat mengakses JavaScript dari dalam tag HTML, yang menghasilkan interaktivitas dinamis yang hebat. Misalnya, untuk menambahkan efek cepat saat mouse melewati suatu objek, kita dapat menggunakan kode seperti pada tag <img> pada Contoh 6, yang menampilkan apel secara default, tetapi menggantinya dengan oranye saat mouse melewati, dan mengembalikan apel lagi ketika mouse pergi.

Contoh 6. Menggunakan JavaScript sebaris

```
<!DOCTYPE html>
<html>
<head>
<title>Inline JavaScript</title>
</head>
<body>
<img src='apple.png'
onmouseover="this.src='orange.png'"
onmouseout="this.src='apple.png'">
</body>
</html>
```

Kata Kunci This

Pada contoh sebelumnya, kita melihat kata kunci this sedang digunakan. Ini memberitahu JavaScript untuk beroperasi pada objek panggilan, yaitu tag <img>. kita dapat melihat hasilnya pada Gambar 13.2, di mana mouse belum melewati apel



Gambar 13.2 Contoh JavaScript hover mouse sebaris

13.5 MELAMPIRKAN ACARA KE OBJEK DALAM SCRIPT

Kode sebelumnya sama dengan memberikan ID ke tag <img>, lalu melampirkan tindakan ke peristiwa mouse tag, seperti Contoh 7.

Contoh 7 JavaScript non-sebaris

```
<!DOCTYPE html>
<html>
<head>
<title>Non-inline JavaScript</title>
<script src='OSC.js'></script>
</head>
<body>
<img id='object' src='apple.png'>
<script>
O('object').onmouseover = function() { this.src = 'orange.png' }
O('object').onmouseout = function() { this.src = 'apple.png' }
</script>
</body>
</html>
```

Di bagian HTML, contoh ini memberikan elemen <img> sebuah ID objek, kemudian mulai memanipulasinya secara terpisah di bagian JavaScript, dengan melampirkan fungsi anonim ke setiap peristiwa.

Melampirkan ke Acara Lain

Baik kita menggunakan JavaScript sebaris atau terpisah, ada beberapa peristiwa yang dapat kita lampirkan tindakannya, memberikan banyak fitur tambahan yang dapat kita tawarkan kepada pengguna Anda. Tabel 13.2 mencantumkan peristiwa ini dan detailnya kapan akan dipicu.

Tabel 13.2 Peristiwa dan kapan dipicu

onabort	Saat pemuatan gambar dihentikan sebelum selesai

onblur	Ketika sebuah elemen kehilangan fokus
onchange	Ketika ada bagian dari formulir yang berubah
onclick	Ketika suatu objek diklik
ondblclick	Ketika sebuah objek diklik dua kali
onerror	Ketika kesalahan JavaScript ditemukan
onfocus	Ketika sebuah elemen mendapat fokus
onkeydown	Saat tombol ditekan (termasuk Shift, Alt, Ctrl, dan Esc)
onkeypress	Saat tombol ditekan (tidak termasuk Shift, Alt, Ctrl, dan Esc)
onkeyup	Saat kunci dilepaskan
onload	Ketika sebuah objek telah dimuat
onmousedown	Saat tombol mouse ditekan di atas elemen
onmousemove	Saat mouse digerakkan di atas elemen
onmouseout	Saat mouse meninggalkan elemen
onmouseover	Saat mouse melewati elemen dari luarnya
onmouseup	Saat tombol mouse dilepaskan
onsubmit	Saat formulir dikirimkan
onreset	Saat formulir diatur ulang
onresize	Saat browser diubah ukurannya
onscroll	Saat dokumen digulir
onselect	Ketika beberapa teks dipilih
onunload	Saat dokumen dihapus

13.6 MENAMBAHKAN ELEMEN BARU

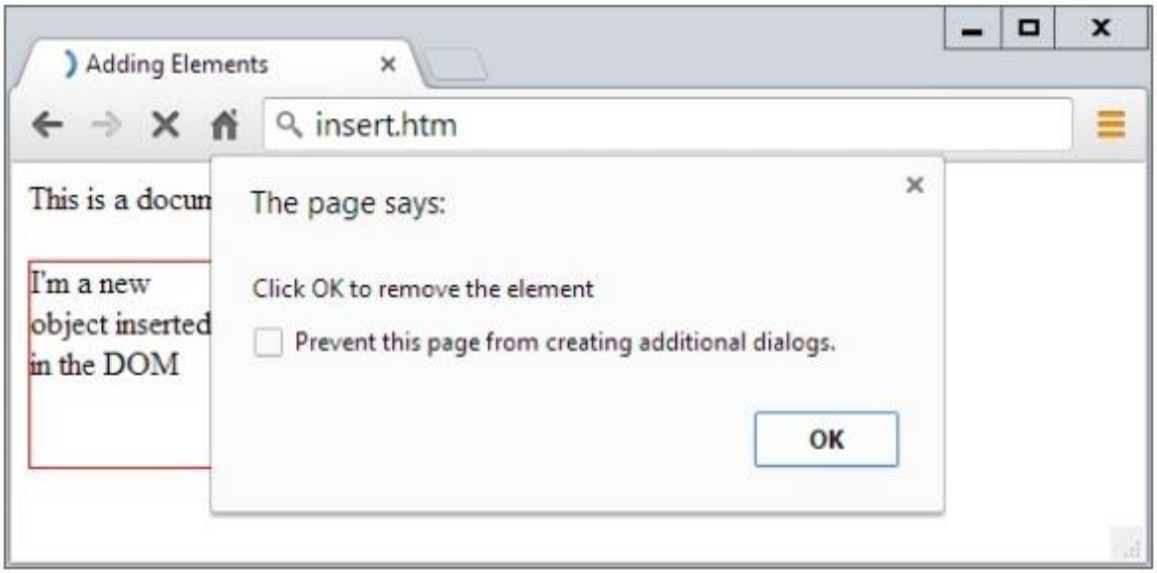
Dengan JavaScript kita tidak dibatasi untuk memanipulasi elemen dan objek yang diberikan ke dokumen dalam HTML-nya. Bahkan, kita dapat membuat objek sesuka hati dengan memasukkannya ke dalam DOM. Misalnya, kita memerlukan elemen baru. Contoh 8 menunjukkan satu cara kita dapat menambahkannya ke halaman web.

Contoh 8 Memasukkan elemen ke dalam DOM

```
<!DOCTYPE html>
<html>
<head>
<title>Adding Elements</title>
<script src='OSC.js'></script>
</head>
<body>
This is a document with only this text in it.<br><br>
<script>
alert('Click OK to add an element')
newdiv = document.createElement('div')
newdiv.id = 'NewDiv'
document.body.appendChild(newdiv)
S(newdiv).border = 'solid 1px red'
S(newdiv).width = '100px'
S(newdiv).height = '100px'
newdiv.innerHTML = "I'm a new object inserted in the DOM"
tmp = newdiv.offsetTop
```

```
alert('Click OK to remove the element')
pnode = newdiv.parentNode
pnode.removeChild(newdiv)
tmp = pnode.offsetTop
</script>
</body>
</html>
```

Gambar 13.3 menunjukkan kode diatas yang digunakan untuk menambahkan elemen baru ke dokumen web. Pertama, elemen baru dibuat dengan createElement, kemudian fungsi appendChild dipanggil dan elemen dimasukkan ke dalam DOM.



Gambar 13.3 Memasukkan elemen baru ke dalam DOM

Setelah ini, berbagai properti ditetapkan ke elemen, termasuk beberapa teks untuk HTML bagian dalamnya. Dan kemudian, untuk memastikan elemen baru langsung terungkap, properti offsetTop-nya dibacakan ke dalam variabel sekali pakai tmp. Ini memaksa penyegaran DOM dan membuat elemen ditampilkan di browser apa pun yang mungkin tertunda sebelum melakukannya—khususnya Internet Explorer. Elemen baru ini persis sama seperti jika telah disertakan dalam HTML asli, dan memiliki semua properti dan metode yang sama yang tersedia.

Menghapus Elemen

Anda juga dapat menghapus elemen dari DOM, termasuk yang tidak kita sisipkan menggunakan JavaScript; bahkan lebih mudah daripada menambahkan elemen. Ini berfungsi seperti ini, dengan asumsi elemen yang akan dihapus ada di elemen objek:

```
element.parentNode.removeChild(element)
```

Kode ini mengakses objek parentNode elemen sehingga dapat menghapus elemen dari node tersebut. Kemudian ia memanggil metode removeChild pada objek itu, meneruskan objek yang akan dihapus. Namun, untuk memastikan DOM langsung disegarkan di semua browser, kita mungkin lebih suka mengganti pernyataan tunggal sebelumnya dengan sesuatu seperti berikut:

```
pnode = element.parentNode
pnode.removeChild(element)
tmp = pnode.offsetTop
```

Alternatif untuk Menambah dan Menghapus Elemen

Memasukkan elemen dimaksudkan untuk menambahkan objek yang sama sekali baru ke dalam halaman web. Tetapi jika semua yang ingin kita lakukan adalah menyembunyikan dan mengungkapkan objek menurut onmouseover atau peristiwa lainnya, jangan lupa bahwa selalu ada beberapa properti CSS yang dapat kita gunakan untuk tujuan ini, tanpa mengambil tindakan drastis seperti membuat dan menghapus DOM elemen. Misalnya, ketika kita ingin membuat elemen tidak terlihat tetapi membiarkannya di tempatnya (dan dengan semua elemen di sekitarnya tetap berada di posisinya), kita cukup menyetel properti visibilitas objek ke 'tersembunyi', seperti ini:

```
myobject.visibility = 'hidden'
```

Dan untuk menampilkan kembali objek, kita dapat menggunakan yang berikut ini:

```
myobject.visibility = 'visible'
```

Anda juga dapat menciutkan elemen ke bawah untuk menempati lebar dan tinggi nol (dengan semua objek di sekitarnya mengisi ruang kosong), seperti ini:

```
myobject.display = 'none'
```

Untuk kemudian mengembalikan elemen ke dimensi aslinya, kita akan menggunakan yang berikut ini:

```
myobject.display = 'block'
```

Dan, tentu saja, selalu ada properti innerHTML, yang dengannya kita dapat mengubah HTML yang diterapkan ke elemen, seperti ini misalnya:

```
mylement.innerHTML = '<b>Replacement HTML</b>'
```

Atau kita dapat menggunakan fungsi O yang saya uraikan sebelumnya, seperti ini:

```
O('someid').innerHTML = 'New contents'
```

Atau kita dapat membuat elemen tampak menghilang, seperti ini:

```
O('someid').innerHTML = ''
```

Catatan: Jangan lupa properti CSS berguna lainnya yang dapat kita akses dari JavaScript, seperti opacity untuk mengatur visibilitas objek ke suatu tempat antara terlihat dan tidak

terlihat, atau lebar dan tinggi untuk mengubah ukuran objek. Dan, tentu saja, dengan menggunakan properti posisi dengan nilai 'absolut', 'statis', atau 'relatif', kita bahkan dapat menemukan objek di mana saja di (atau di luar) jendela browser yang kita suka.

13.7 MENGGUNAKAN INTERUPSI

JavaScript menyediakan akses ke interupsi, sebuah metode di mana kita dapat meminta browser untuk memanggil kode kita setelah jangka waktu tertentu, atau bahkan untuk terus memanggilnya pada interval tertentu. Ini memberi kita cara untuk menangani tugas latar belakang seperti komunikasi Ajax, atau bahkan hal-hal seperti animasi elemen web. Untuk mencapai ini, kita memiliki dua jenis interupsi: setTimeout dan setInterval, yang memiliki fungsi clearTimeout dan clearInterval yang menyertai untuk mematikannya lagi.

Menggunakan setTimeout

Saat kita memanggil setTimeout, kita meneruskannya beberapa kode JavaScript atau nama fungsi, dan nilai dalam milidetik yang menunjukkan berapa lama menunggu sebelum kode harus dieksekusi, seperti ini:

```
setTimeout(dothis, 5000)
```

Dan fungsi dothis kita mungkin terlihat seperti ini:

```
function dothis()
{
alert('This is your wakeup alert!');
}
```

Catatan: Jika kita bertanya-tanya, kita tidak bisa begitu saja menentukan alert() (dengan tanda kurung) sebagai fungsi yang akan dipanggil oleh setTimeout, karena fungsi tersebut akan langsung dieksekusi. Hanya ketika kita memberikan nama fungsi tanpa tanda kurung argumen (mis., Peringatan) kita dapat dengan aman meneruskan nama fungsi sehingga kodenya akan dieksekusi hanya ketika batas waktu terjadi.

Melewati string

Saat kita perlu memberikan argumen ke suatu fungsi, kita juga bisa meneruskan nilai string ke fungsi setTimeout, yang tidak akan dieksekusi hingga waktu yang tepat, seperti ini:

```
setTimeout("alert('Hello!')", 5000)
```

Sebenarnya, kita dapat memberikan baris kode JavaScript sebanyak yang kita suka, jika kita menempatkan titik koma setelah setiap pernyataan, seperti ini:

```
setTimeout("document.write('Starting'); alert('Hello!')", 5000)
```

Timeout berulang

Salah satu teknik yang digunakan beberapa programmer untuk menyediakan interupsi berulang dengan setTimeout adalah dengan memanggil fungsi setTimeout dari kode yang dipanggil olehnya, seperti berikut ini, yang akan memulai loop jendela peringatan yang tidak pernah berakhir:

```
setTimeout(dothis, 5000)
function dothis()
{
setTimeout(dothis, 5000)
alert('I am annoying!')
}
```

Sekarang peringatan akan muncul setiap lima detik.

Membatalkan Timeout

Setelah batas waktu diatur, kita dapat membatalkannya jika sebelumnya kita menyimpan nilai yang dikembalikan dari panggilan awal ke setTimeout, seperti ini:

```
handle = setTimeout(dothis, 5000)
```

Berbekal nilai dalam pegangan, kita sekarang dapat membatalkan interupsi kapan saja hingga waktunya, seperti ini:

```
clearTimeout(handle)
```

Ketika kita melakukan ini, interupsi benar-benar dilupakan, dan kode yang diberikan padanya tidak akan dieksekusi.

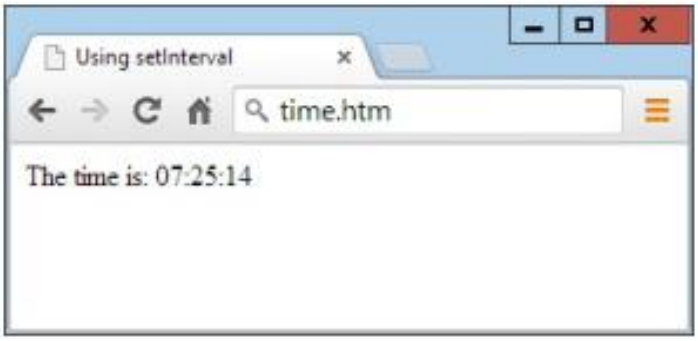
Menggunakan setInterval

Cara yang lebih mudah untuk mengatur interupsi reguler adalah dengan menggunakan fungsi setInterval. Ini bekerja dengan cara yang sama, kecuali bahwa setelah muncul setelah interval yang kita tentukan dalam milidetik, ia akan melakukannya lagi setelah interval itu berlalu lagi, dan seterusnya selamanya, kecuali jika kita membatalkannya. Contoh 9 menggunakan fungsi ini untuk menampilkan jam sederhana di browser, seperti yang ditunjukkan pada Gambar 13.4.

Contoh 9. Jam yang dibuat menggunakan interupsi

```
<!DOCTYPE html>
<html>
<head>
<title>Using setInterval</title>
<script src='OSC.js'></script>
</head>
<body>
The time is: <span id='time'>00:00:00</span><br>
<script>
setInterval("showtime(O('time'))", 1000)
function showtime(object)
{
var date = new Date()
object.innerHTML = date.toTimeString().substr(0,8)
}
</script>
```

```
</body>
</html>
```



Gambar 13.4 Mempertahankan waktu yang tepat dengan interupsi

Setiap kali ShowTime dipanggil, ia menetapkan tanggal objek ke tanggal dan waktu saat ini dengan panggilan ke Date:

```
var date = new Date()
```

Kemudian properti innerHTML dari objek yang diteruskan ke showtime (yaitu, objek) diatur ke waktu saat ini dalam jam, menit, dan detik, sebagaimana ditentukan oleh panggilan ke toTimeString. Ini mengembalikan string seperti 09:57:17 UTC+0530, yang kemudian dipotong menjadi hanya delapan karakter pertama dengan panggilan ke fungsi substr:

```
object.innerHTML = date.toTimeString().substr(0,8)
```

Menggunakan fungsi

Untuk menggunakan fungsi ini, pertama-tama kita harus membuat objek yang properti innerHTML-nya akan digunakan untuk menampilkan waktu, seperti HTML ini:

```
The time is: 00:00:00
```

Kemudian, dari bagian kode <script>, panggilan dilakukan ke fungsi setInterval, seperti ini:

```
setInterval("showtime(O('time'))", 1000)
```

Kemudian meneruskan string ke setInterval, yang berisi pernyataan berikut, yang diatur untu k dieksekusi sekali per detik (setiap 1.000 milidetik):

```
showtime(O('time'))
```

Dalam situasi yang jarang terjadi di mana seseorang telah menonaktifkan JavaScript (yang terkadang dilakukan orang untuk alasan keamanan), JavaScript kita tidak akan berjalan dan pengguna akan melihat 00:00:00 yang asli.

Membatalkan interval

Untuk menghentikan interval berulang, saat pertama kali mengatur interval dengan panggilan ke setInterval, kita harus membuat catatan tentang pegangan interval, seperti ini:

handle = setInterval("showtime(O('time'))", 1000)

Sekarang kita dapat menghentikan jam kapan saja dengan mengeluarkan panggilan berikut:

clearInterval(handle)

Anda bahkan dapat mengatur timer untuk menghentikan jam setelah jangka waktu tertentu, seperti ini:

setTimeout("clearInterval(handle)", 10000)

Pernyataan ini akan mengeluarkan interupsi dalam 10 detik yang akan menghapus interval berulang.

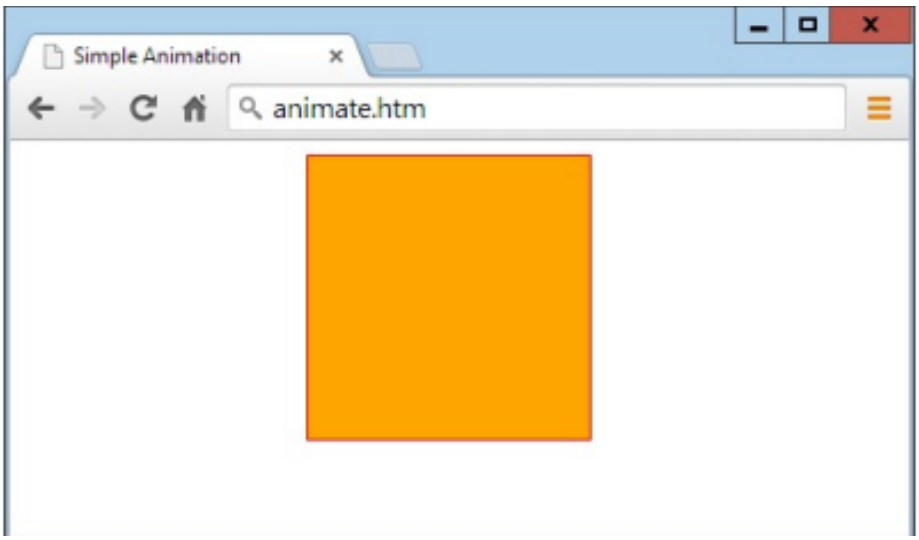
Menggunakan Interupsi untuk Animasi

Dengan menggabungkan beberapa properti CSS dengan interupsi berulang, kita dapat menghasilkan segala macam animasi dan efek. Misalnya, kode pada Contoh 10 memindahkan bentuk persegi di bagian atas browser, sepanjang waktu menggelembung dalam ukuran, seperti yang ditunjukkan pada Gambar 13.5 sebelum memulai dari awal lagi saat LEFT direset ke 0.

Contoh 10 Animasi sederhana

```
<!DOCTYPE html>
<html>
<head>
<title>Simple Animation</title>
<script src='OSC.js'></script>
<style>
#box {
position :absolute;
background:orange;
border :1px solid red;
}
</style>
</head>
<body>
<div id='box'></div>
<script>
SIZE = LEFT = 0
setInterval(animate, 30)
function animate()
{
SIZE += 10
LEFT += 3
if (SIZE == 200) SIZE = 0
if (LEFT == 600) LEFT = 0
```

```
S('box').width = SIZE + 'px'
S('box').height = SIZE + 'px'
S('box').left = LEFT + 'px'
}
</script>
</body>
</html>
```



Gambar 13.5 Objek ini meluncur dari kiri sambil mengubah ukuran

Dalam dokumen <head>, objek kotak diatur ke warna latar belakang 'oranye' dengan nilai batas '1px merah solid', dan properti posisinya disetel ke absolut sehingga diizinkan untuk dipindahkan dalam peramban. Kemudian, dalam fungsi animasi, variabel global SIZE dan LEFT terus diperbarui dan kemudian diterapkan ke atribut lebar, tinggi, dan gaya kiri objek kotak (dengan 'px' ditambahkan setelah masing-masing untuk menentukan bahwa nilainya dalam piksel) , sehingga menjiwainya pada frekuensi sekali setiap 30 milidetik— memberikan kecepatan 33,33 frame per detik (1.000/30 milidetik).

DAFTAR PUSTAKA

Abdulloh, Rohi. 2016. Easy & Simple Web Programing. PT Elex Media Komputindo. Jakarta.

Arief, M.R., 2011, Pemrograman Web Dinamis Menggunakan PHP Dan MySQL, Andi Offset, Yogyakarta.

Burnette. Ed. 2009. Brilliant HTML and CSS. United Kingdom: Pearson.

Djayali, A. D. 2020. Analisa Serangan SQL Injection pada Server pengisian Kartu Rencana Studi (KRS) Online. 1(1), 1–9.

Kadir, A., 2001, Pemrograman WEB Mencakup: HTML, CSS, JAVA SCRIPT, dan PHP, Andi Offset, Yogyakarta.

Kadir, A., 2008, Belajar Database Mengunakan MySQL, Andi Offset, Yogyakarta.

Kurniawan, Rulianto. 2010. PHP & MySQL untuk orang awam. Palembang: Maxikom.

Robon Nixon, 2014. Lerning PHP, MySQL, JavaScript, CSS & HTML5, A step by step guide to creating dynamic Websites. Third Edition. O 'Reilly Media, Inc. Canada.

Steve Suehring and Janet Valade, 2013. PHP, MySQL, Javascript & HTML5 All In One for DUMMIES. Published by John Wiley & Sons, Inc., Hoboken, New Jersey.

Styawantoro, I., & Komarudin, A. (2021). PEMROGRAMAN BERBASIS WEB HTML, PHP 7, MySQLi, Dan Bootstrap 4. Penerbit Lakeisha.

Sugiri. (2007). Desain Web Menggunakan HTML + CSS. Yogyakarta: Andi Offset.

Suryana, Taryana dan Koesheryatin. 2014. Aplikasi Internet Menggunakan HTML, CSS, & JavaScript. Elex Media Komputindo. Jakarta.

Sutarman., 2003, Membangun Aplikasi Web Dengan PHP Dan MySQL, Graha Ilmu, Yogyakarta.

Winarno, Edy, ST, M.Eng. Zaki, Ali. SmitDev Community. 2015. Desain Web Responsif dengan HTML5 dan CSS3. PT Elex Media Komputindo. Jakarta.

HTML CSS dan Javascript

Muhammad Sholikhan, S.Kom., M.Kom

BIODATA PENULIS

Penulis buku ini adalah dosen Universitas Sains dan Teknologi Komputer bernama Muhammad Sholikhan, S.Kom., M.Kom, lahir di kota Kudus, 14 Juni 1982. Penulis memiliki riwayat pendidikan S1 Jurusan Sistem Komputer Grafis di Sekolah Tinggi Elektronika dan Komputer (STEKOM) Semarang dan S2 Magister Sistem Informasi Universitas Kristen Satya Wacana (UKSW) Salatiga. Saat ini penulis adalah dosen tetap di Universitas Sains dan Teknologi Komputer (Universitas STEKOM) pada program studi S1 Desain Grafis dalam bidang ilmu Sistem Informasi dengan jabatan fungsional Asisten Ahli. Penulis mengampu mata kuliah antara lain Desain Web, Pengantar Teknologi Informasi serta Pengolahan dan Publikasi Konten Digital.

YAYASAN PRIMA AGUS TEKNIK

PENERBIT :

YAYASAN PRIMA AGUS TEKNIK
JL. Majapahit No. 605 Semarang
Telp. (024) 6723456. Fax. 024-6710144
Email : penerbit_ypat@stekom.ac.id

Muhammad Sholikhan, S.Kom., M.Kom

HTML CSS dan Javascript

YAYASAN PRIMA AGUS TEKNIK

PENERBIT :
YAYASAN PRIMA AGUS TEKNIK
JL. Majapahit No. 605 Semarang
Telp. (024) 6723456. Fax. 024-6710144
Email : penerbit_ypat@stekom.ac.id