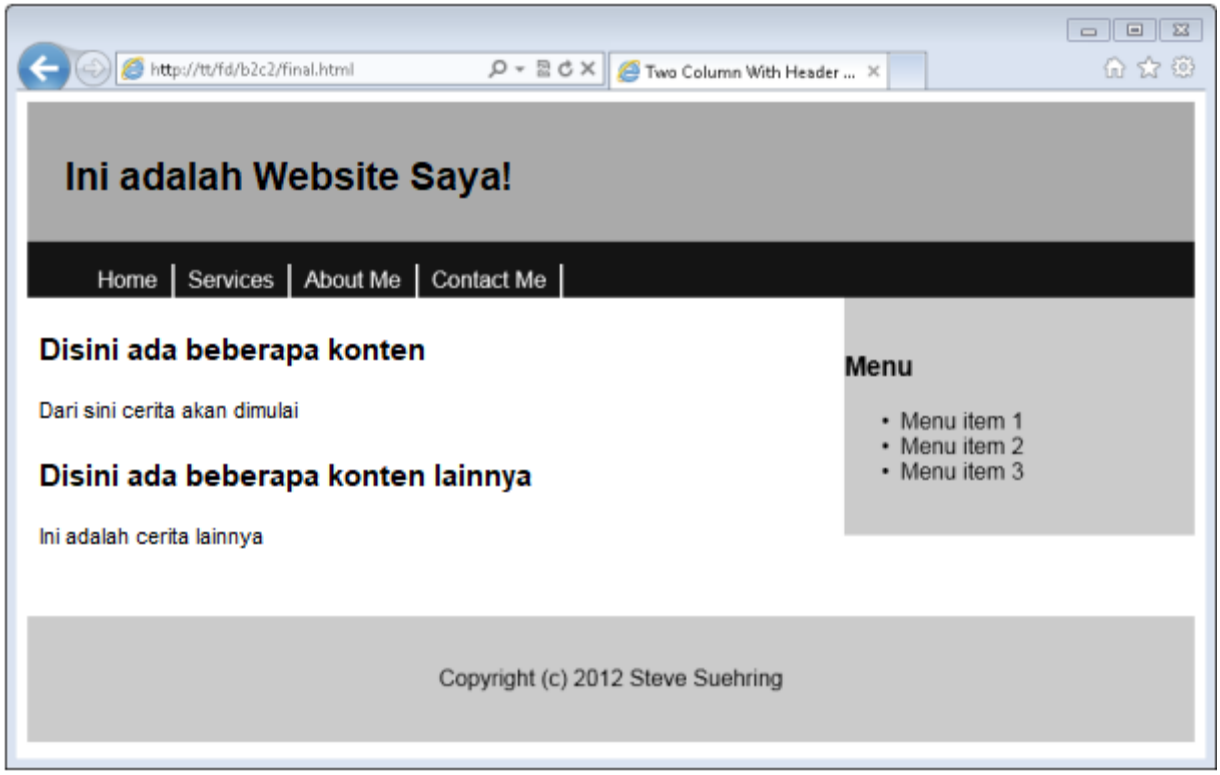
```
<li>Menu item 3</li>
</ul>
</div> <!-- end sidebar -->
<div id="footer">
<p>Copyright (c) 2022Agus Wibowo</p>
</div> <!-- end footer -->
</div> <!-- end mainContainer -->
</div> <!-- end container -->
</body>
```

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;
}
```

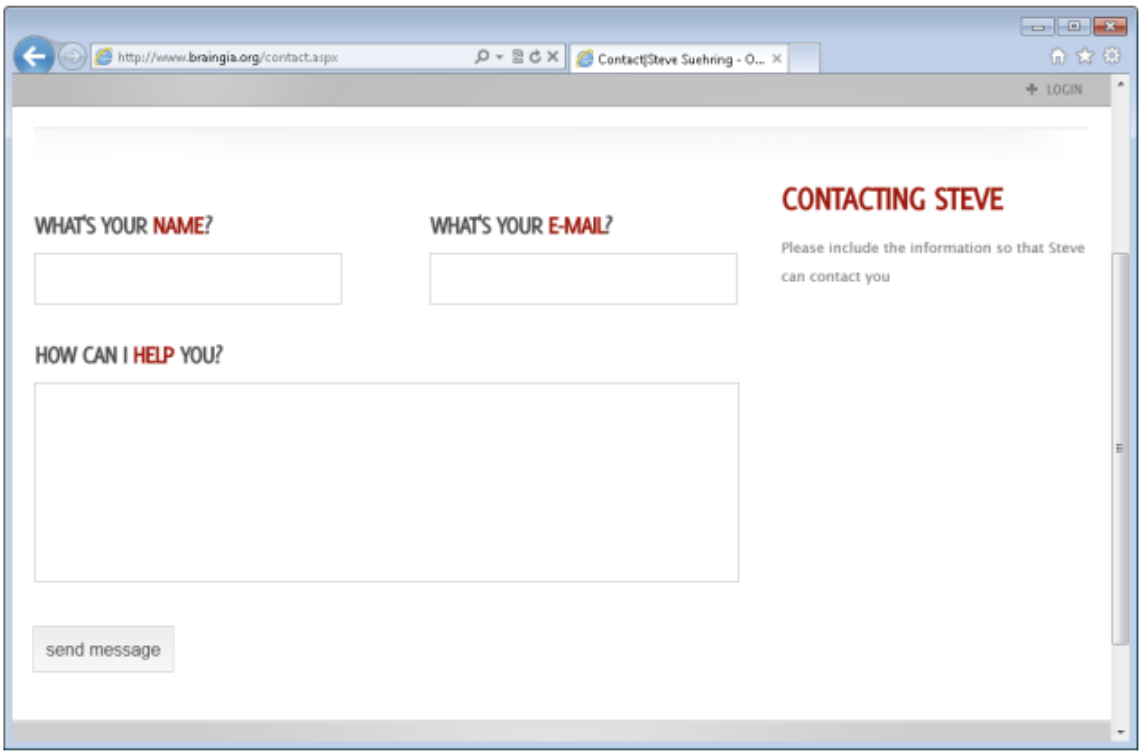
Aturan CSS ini menargetkan perilaku mengarahkan kursor ke elemen <a>. Saat pengunjung mengarahkan kursor ke tautan, itu akan berubah warna, dalam hal ini menjadi #CCCCCC, yang merupakan bayangan abu-abu.

# 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>
```

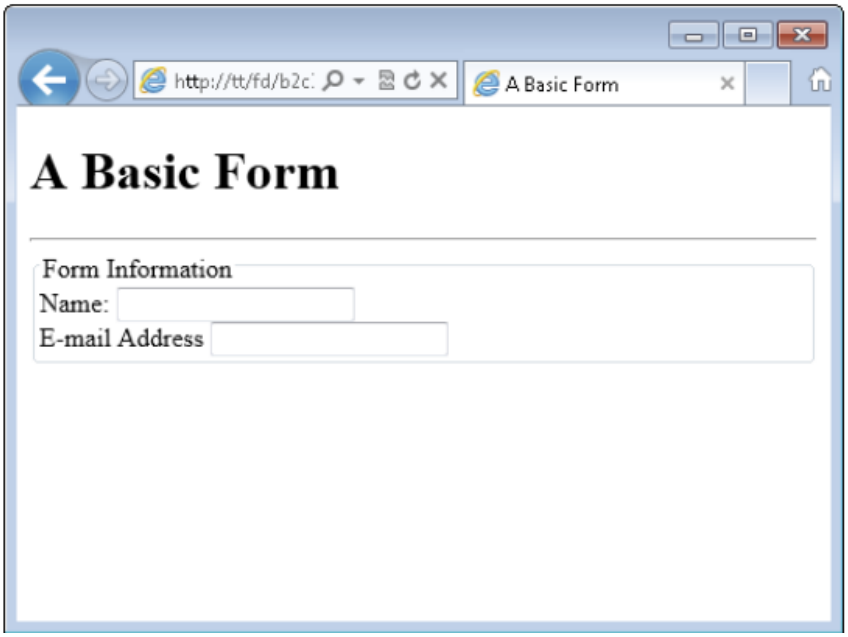
Anda melihat cara membuat formulir kita sendiri di bagian selanjutnya.

***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***

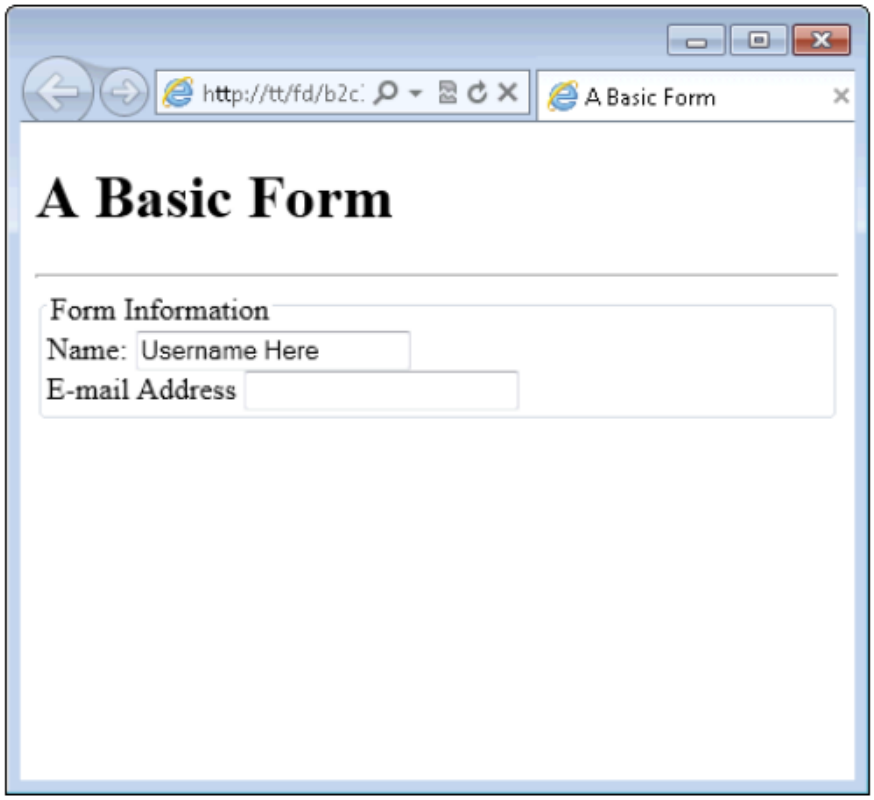
Anda telah melihat input teks dalam bab ini. Menambahkan satu semudah menggunakan jenis "teks". kita juga dapat menambahkan beberapa atribut, ukuran, dan ukuran maksimal yang lebih praktis, yang memberi tahu browser seberapa besar untuk membuat kotak teks di layar dan jumlah karakter maksimum yang diizinkan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

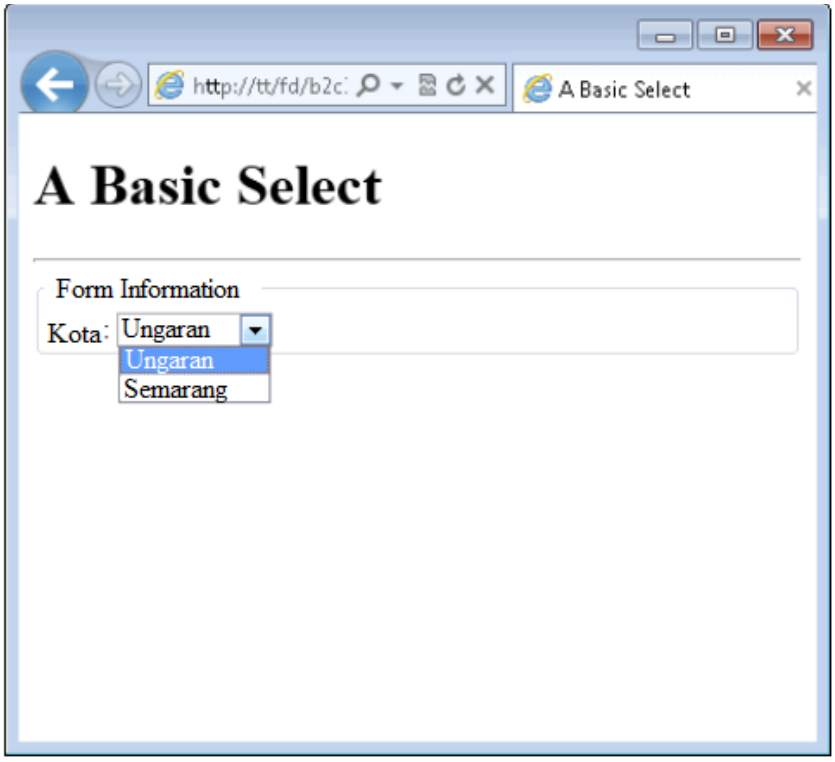
```
<select name=”state”>
 <option value=”CA”>Ungaran</option>
 <option value=”WI”>Semarang</option>
</select>
```

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.

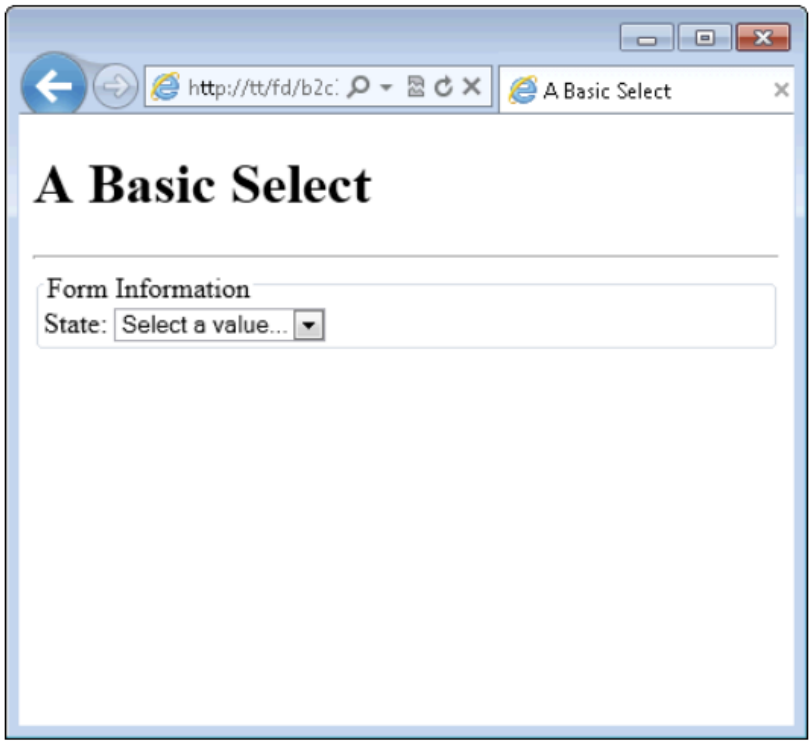
Seperti kotak teks, kita dapat menetapkan nilai default untuk kotak drop-down. Ini dicapai dengan menggunakan atribut yang dipilih. Meskipun tidak selalu diperlukan, sebaiknya tetapkan nilai untuk atribut yang dipilih, seperti dalam contoh ini yang akan mengubah nilai default ke Semarang:

```
<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.

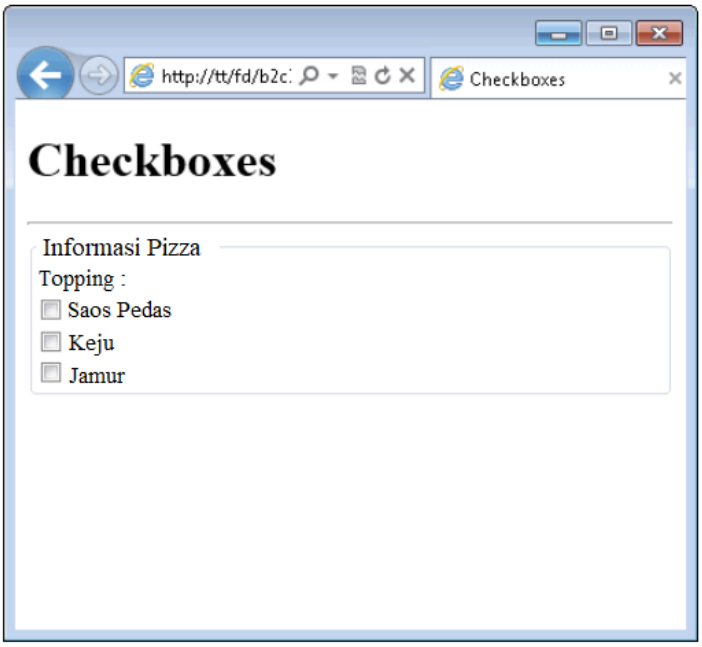
Menggunakan atribut yang dipilih mengesampingkan trik nilai pertama yang ditunjukkan dalam contoh ini.

***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.

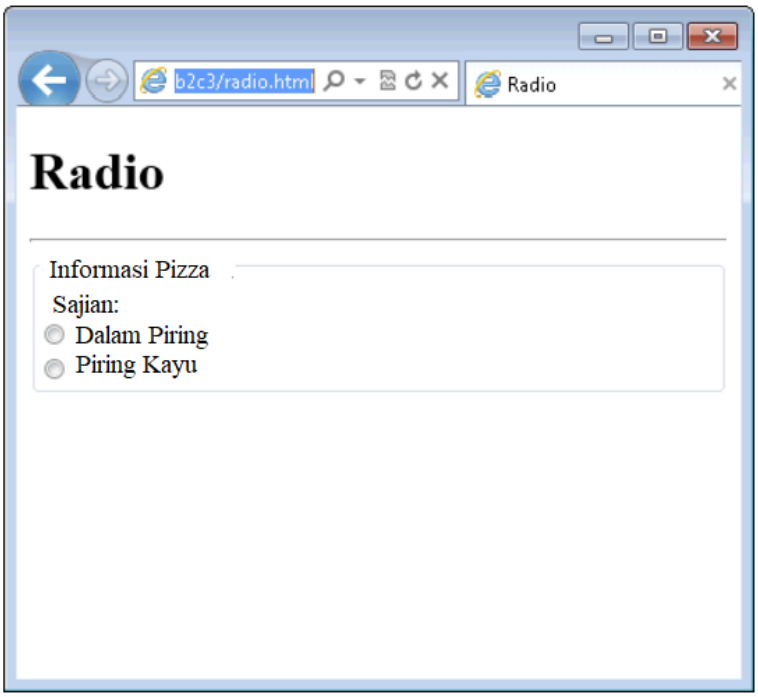
Dalam praktiknya, kita mungkin melihat kotak centang tanpa atribut nama atau dengan atribut nama yang berbeda untuk setiap kotak centang. Contoh yang kita lihat di sini adalah salah satu yang membuat informasi dikelompokkan secara logis, yang membuatnya lebih mudah untuk dipelihara nanti dan juga memudahkan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

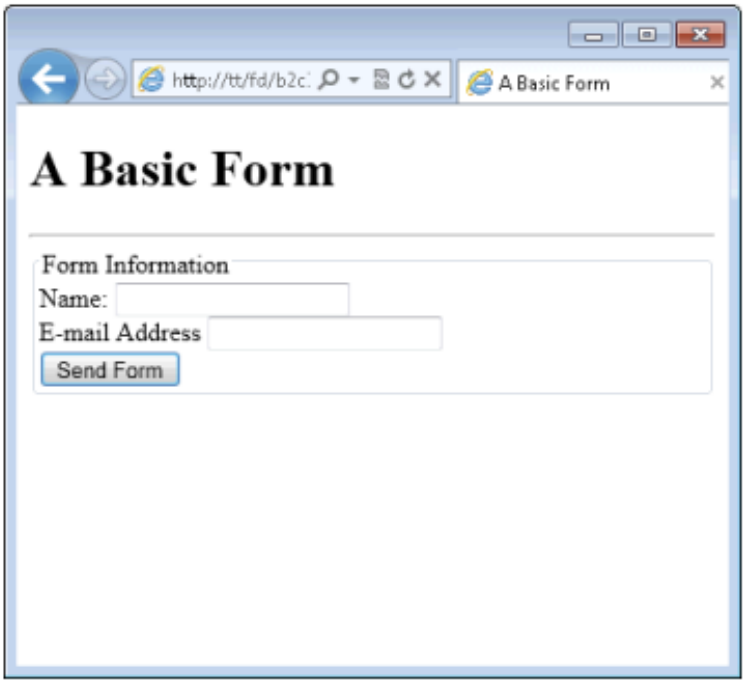
```
<input type=”submit” name=”submit”
 value=”Process Request”>
```

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”>
```

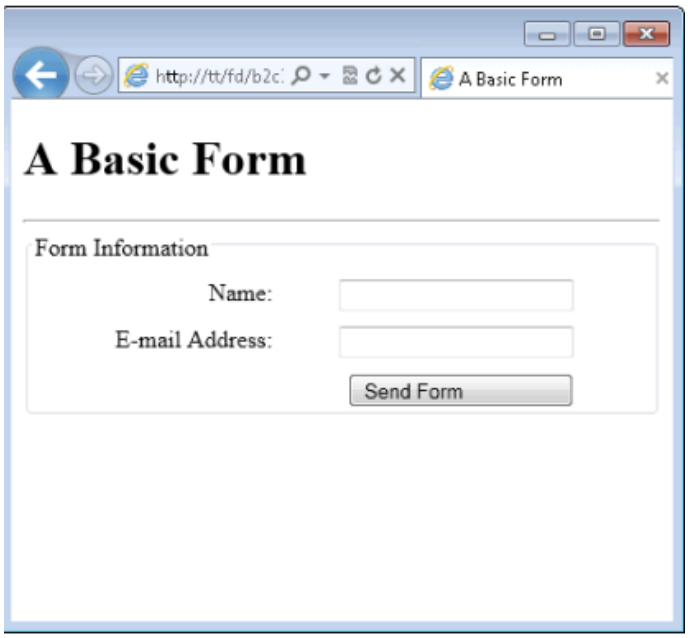
## 3.4 MENGGUNAKAN CSS UNTUK MENYEJAJARKAN BIDANG FORMULIR

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;
}
```

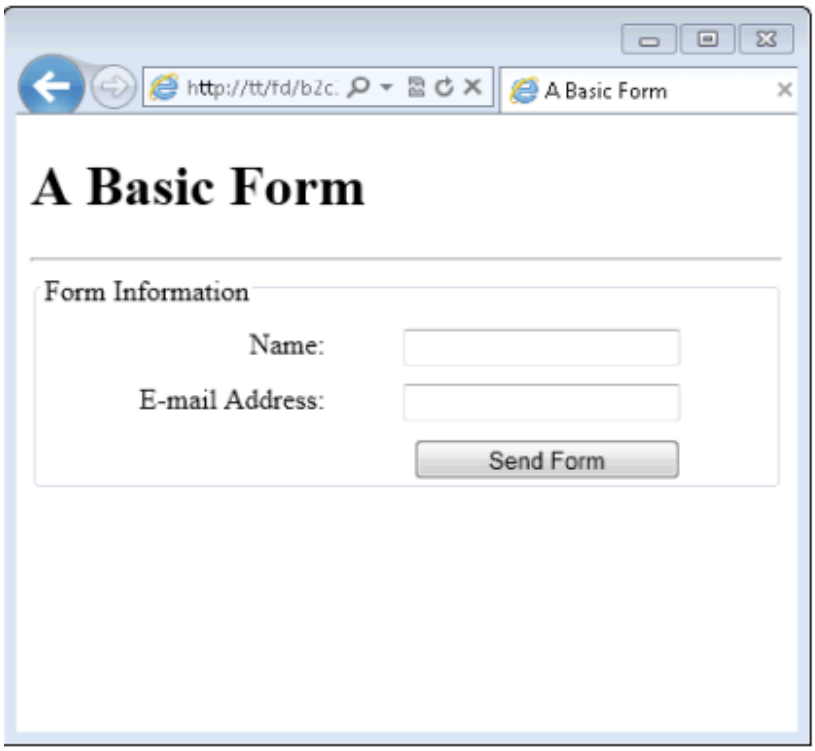
Anda memerlukan cara untuk membuat tombol itu lebih kecil atau setidaknya menyelaraskan teks di tengahnya. Steve secara pribadi menyukai tombol yang lebih besar. Mereka memudahkan pengguna untuk mengklik atau mengetuk, jika mereka menggunakan perangkat seluler. Jadi kami memilih untuk menyelaraskan teks 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:

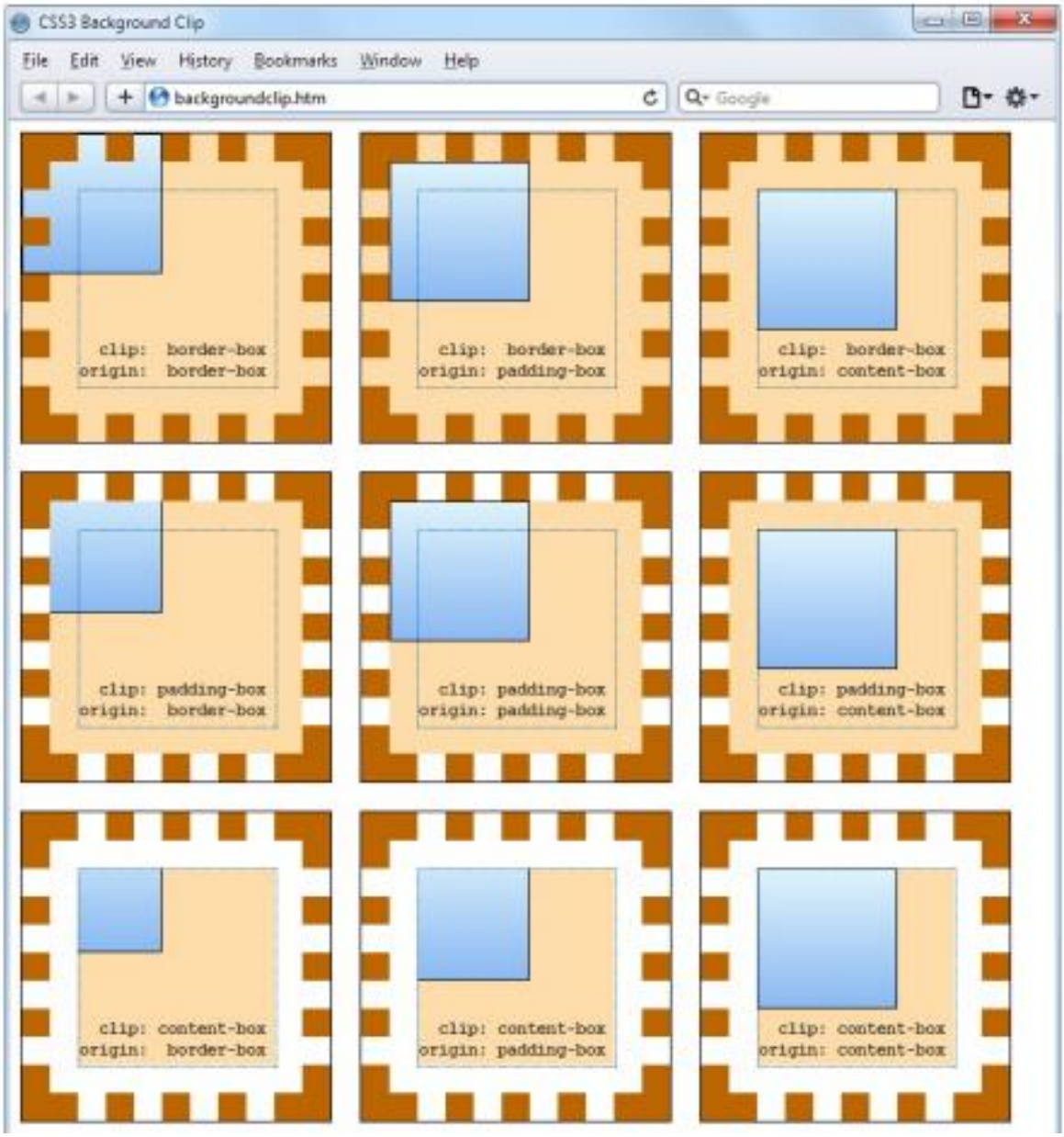
background-clip:border-box;

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**

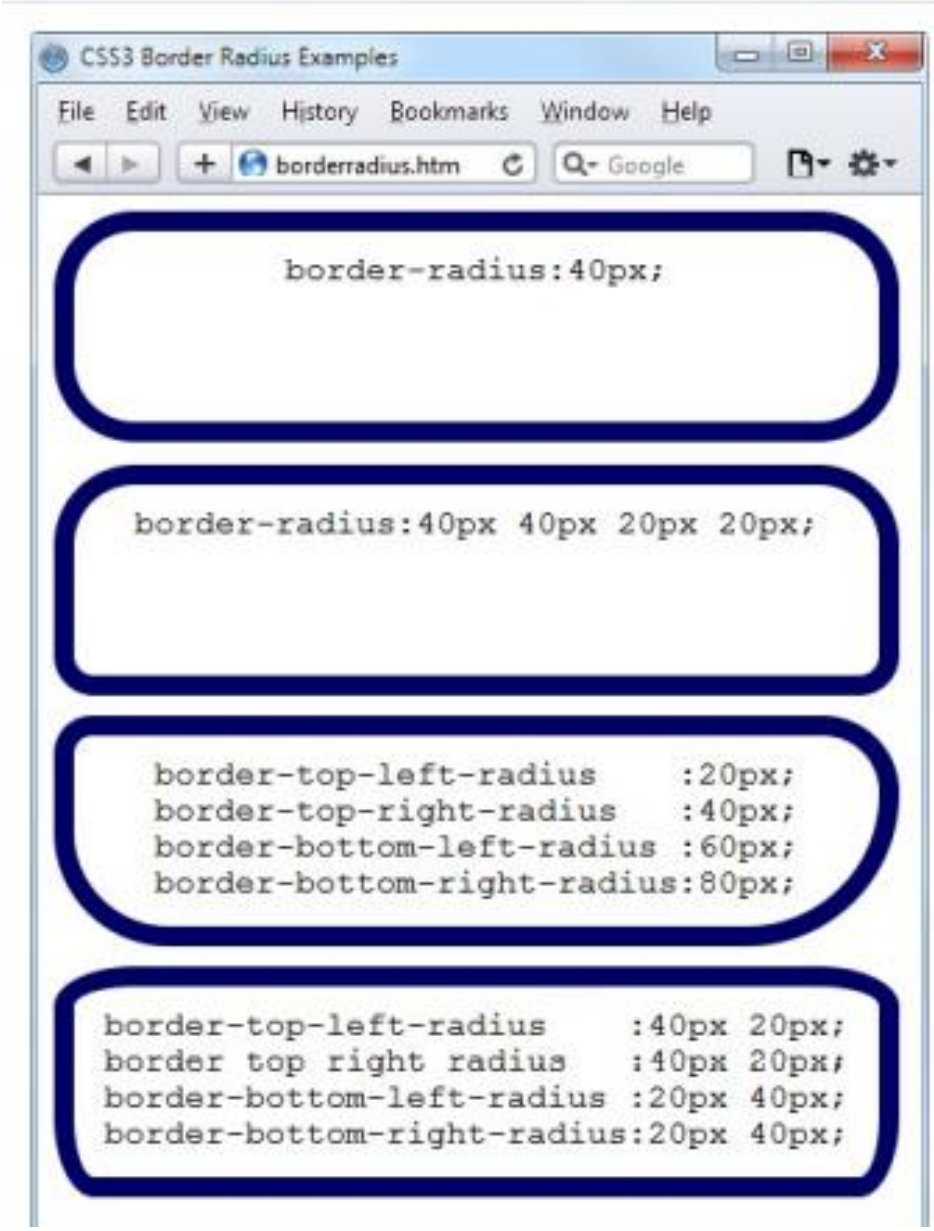
Sebelum CSS3, pengembang web berbakat datang dengan banyak penyesuaian dan perbaikan yang berbeda untuk mencapai batas bulat, umumnya menggunakan tag <table> atau . Tapi sekarang menambahkan batas bulat ke elemen sangat sederhana, dan ini berfungsi pada versi terbaru dari semua browser utama, seperti 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).

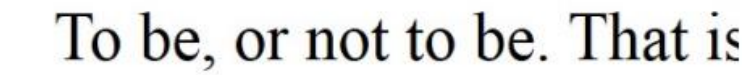

**Gambar 4.6** Menerapkan bayangan ke teks

**Properti text-overflow**

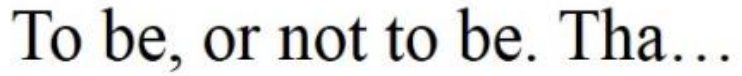
Saat menggunakan salah satu properti CSS overflow dengan nilai tersembunyi, kita juga dapat menggunakan properti text-overflow untuk menempatkan elipsis (tiga titik) tepat sebelum cutoff untuk menunjukkan bahwa beberapa teks telah terpotong, seperti ini:

```
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**

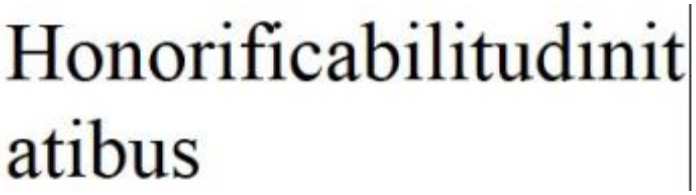
Ketika kita memiliki kata yang sangat panjang yang lebih lebar dari elemen yang memuatnya, kata itu akan meluap atau terpotong. Tetapi sebagai alternatif untuk menggunakan properti text-overflow dan memotong teks, kita dapat menggunakan properti word-wrap dengan nilai break-word untuk membungkus garis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

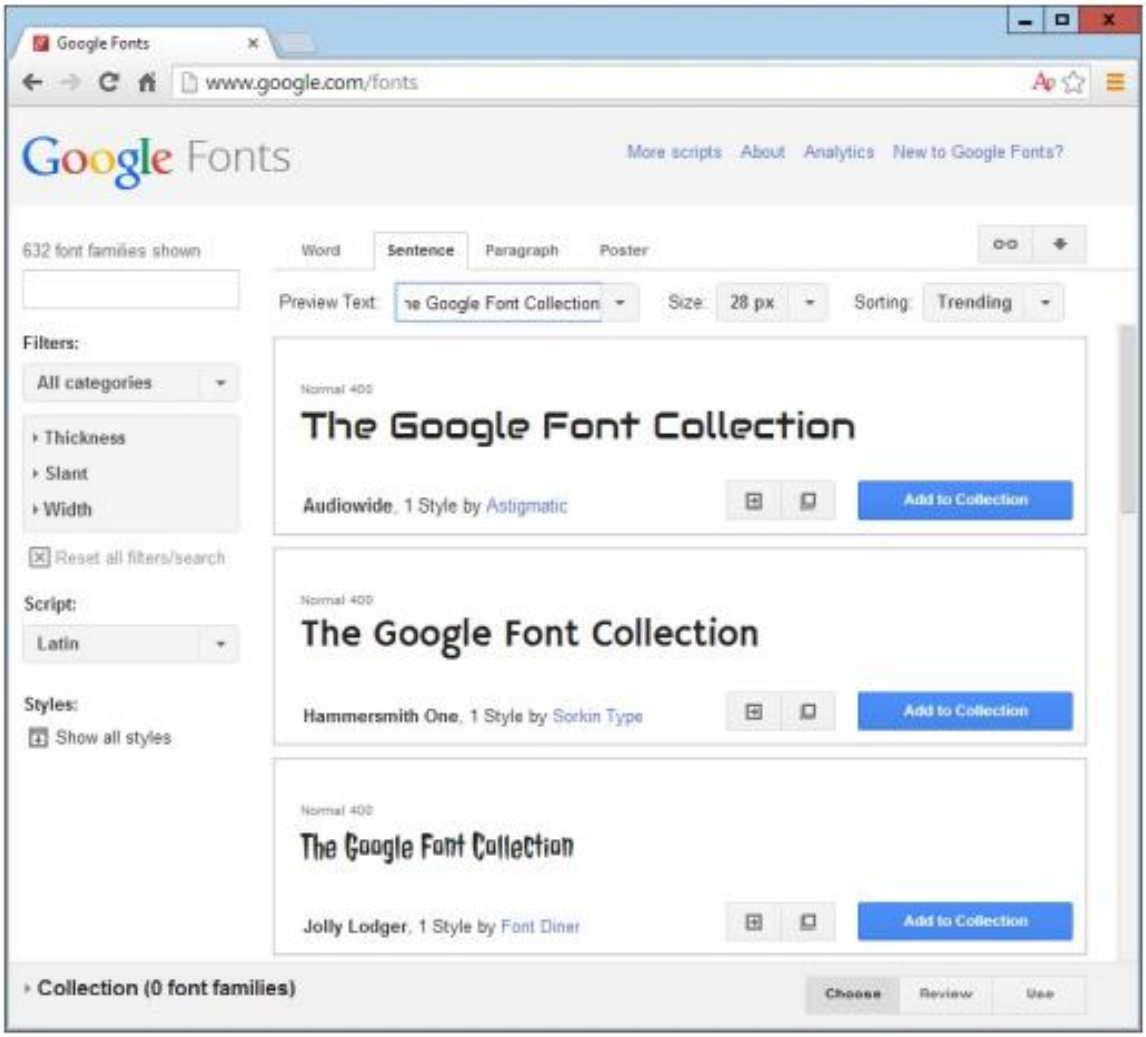
Namun, karena Microsoft Internet Explorer hanya menerima font EOT, itu mengabaikan deklarasi @font-face yang berisi fungsi format.

**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**

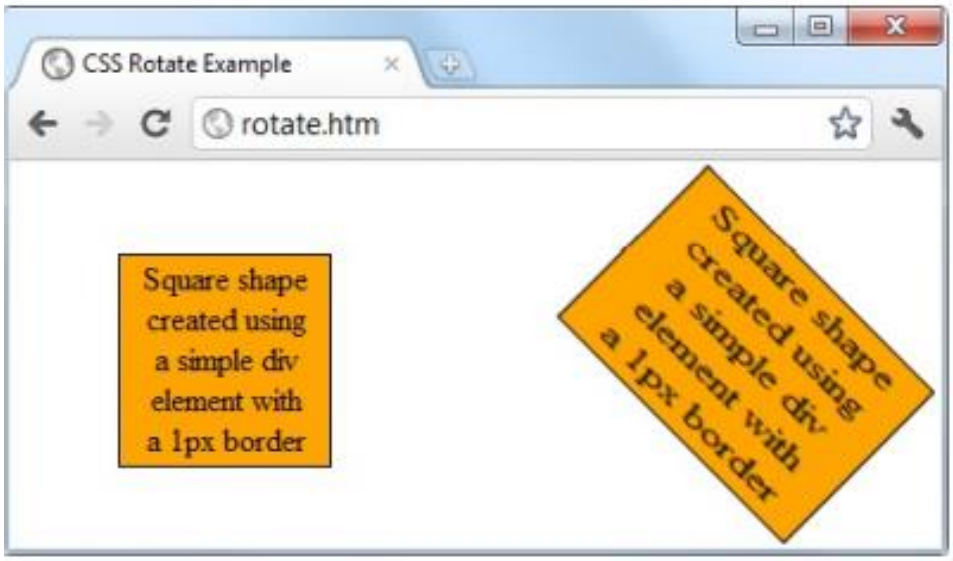
Mengubah objek dengan menerapkan matriks nilai padanya

**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);
```

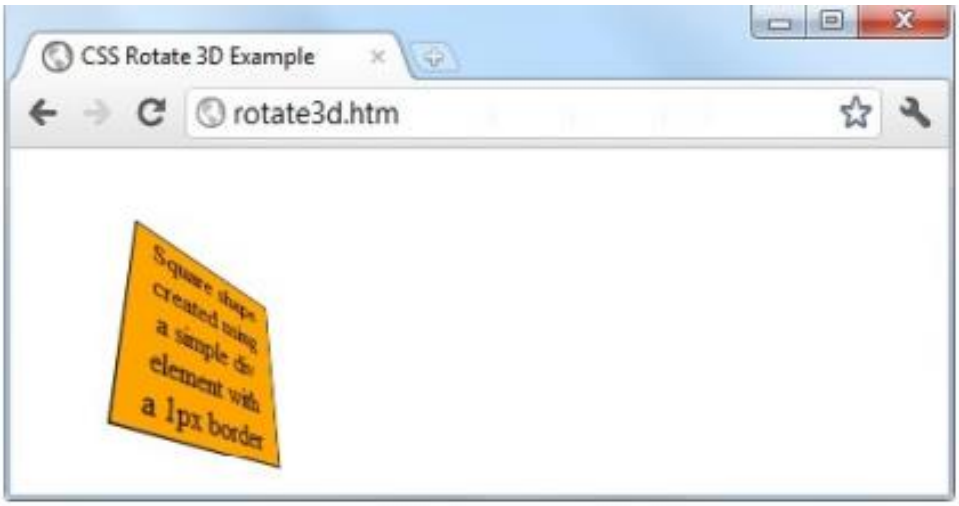



**Gambar 4.12** Sebuah objek sebelum dan sesudah transformasi

**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:

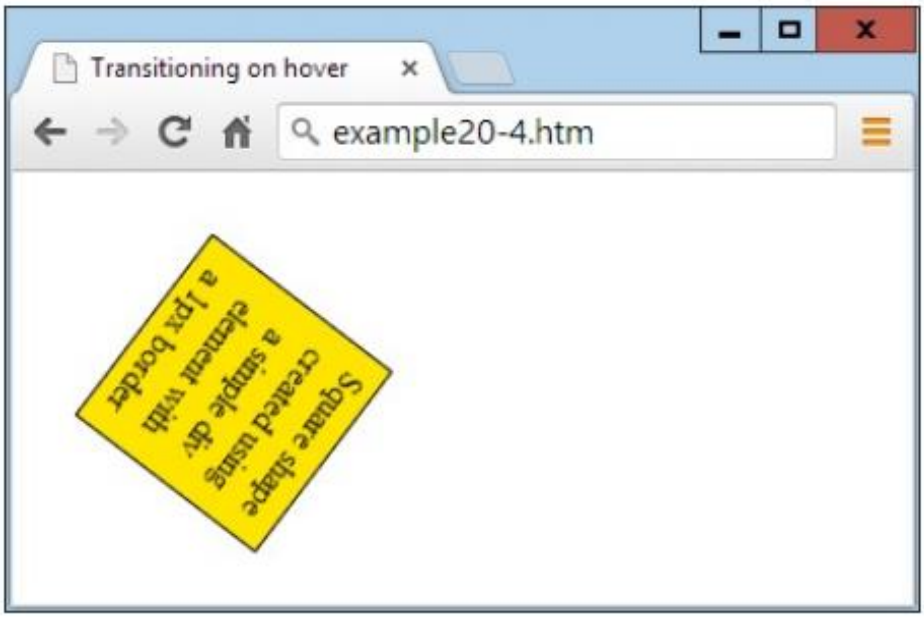
```
transition:all .3s linear .2s;
```

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.

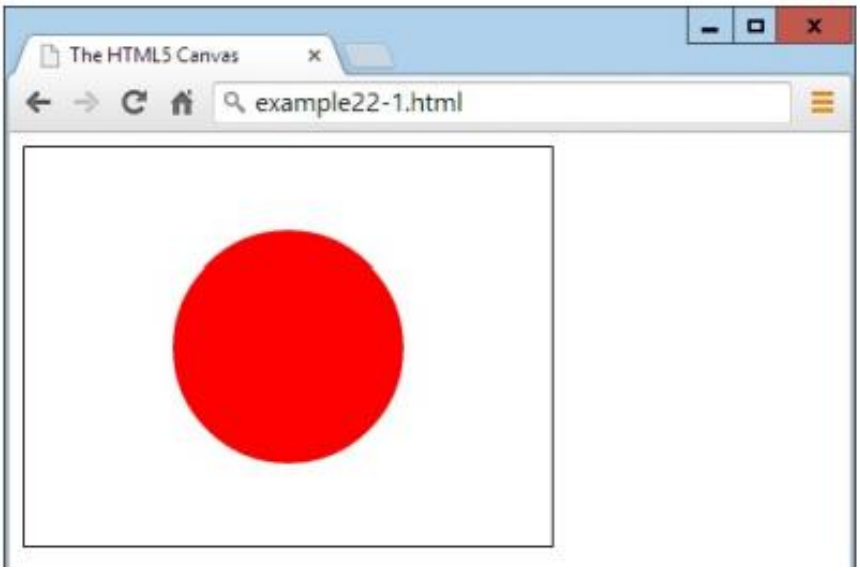
**Kanvas**

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>
```

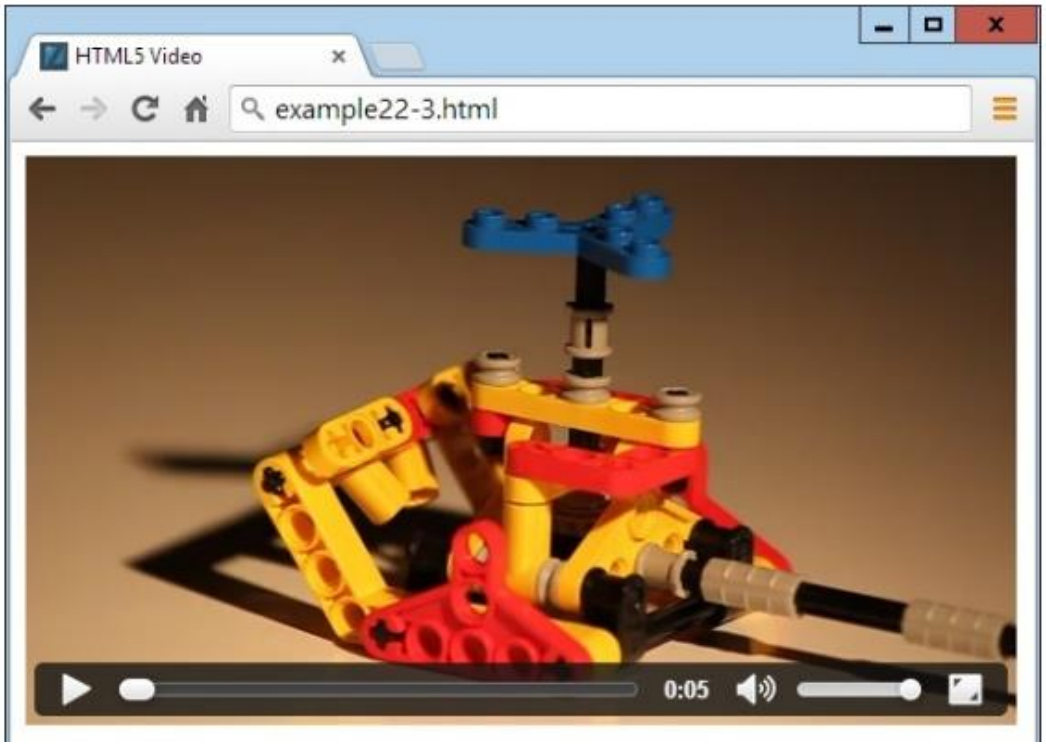
```
<script>
if (typeof navigator.geolocation == 'undefined')
alert("Geolocation not supported.")
else
navigator.geolocation.getCurrentPosition(granted, denied)
function granted(position)
{
O('status').innerHTML = 'Permission Granted'
S('map').border = '1px solid black'
S('map').width = '640px'
S('map').height = '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:

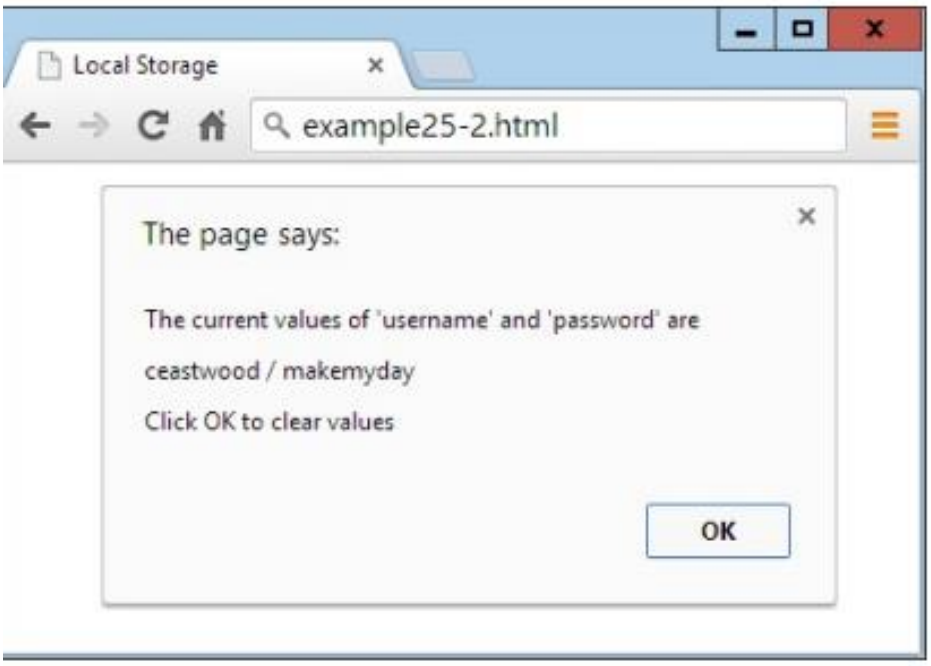
```
localStorage.clear()
```

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

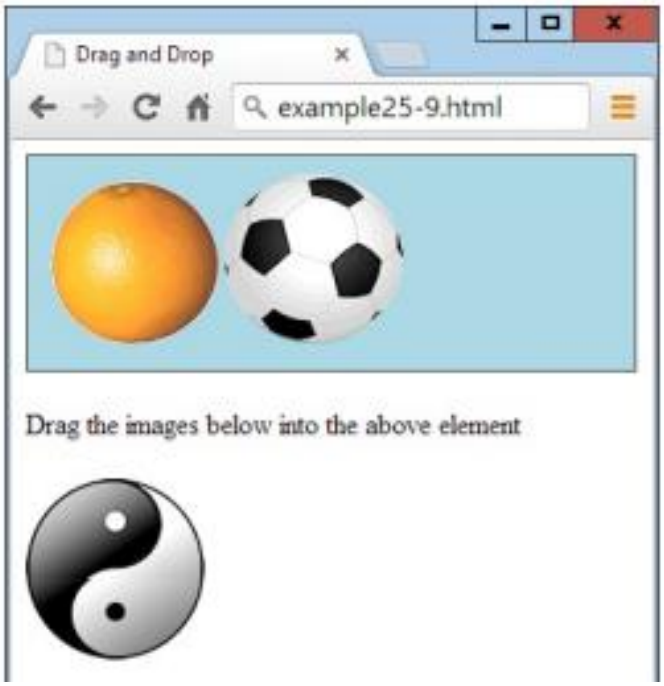
Anda dapat dengan mudah mendukung menyeret dan menjatuhkan objek pada halaman web dengan menyiapkan event handler untuk event ondragstart, ondragover, dan ondrop, seperti pada Contoh 10.

*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>
```

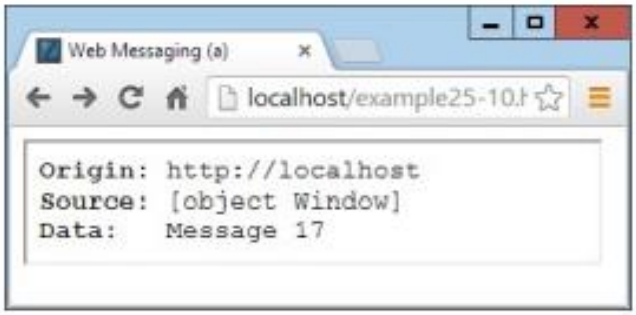
Di sini, file OSC.js biasanya digunakan untuk menarik fungsi O, dan kemudian elemen iframe dengan ID bingkai dibuat, yang dimuat dalam Contoh 12. Kemudian, di dalam bagian <script>, jumlah variabel diinisialisasi ke 1 dan interval berulang diatur agar terjadi setiap detik untuk memposting string 'Pesan' (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*

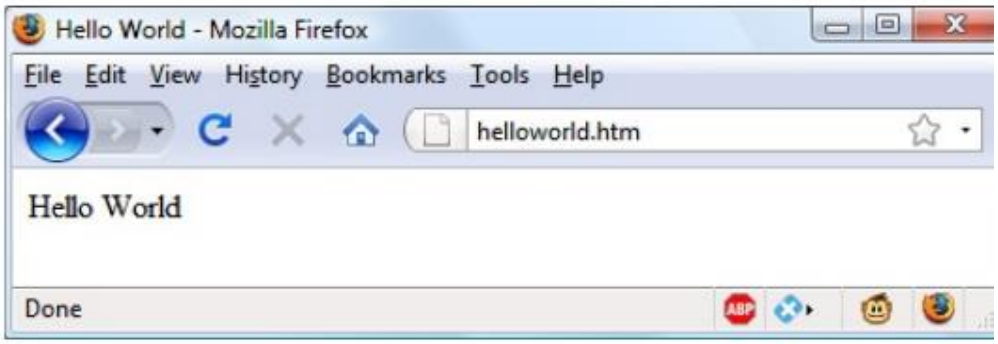
```
<html>
<head><title>Hello World</title></head>
<body>
<script type="text/javascript">
document.write("Hello World")
</script>
<noscript>
Your browser doesn't support or has disabled JavaScript
</noscript>
</body>
</html>
```

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<br>setara (seperti \xA9 untuk simbol ©) |
| \uXXXX | Angka heksadesimal antara 0000 dan FFFF yang mewakili Unicode<br>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

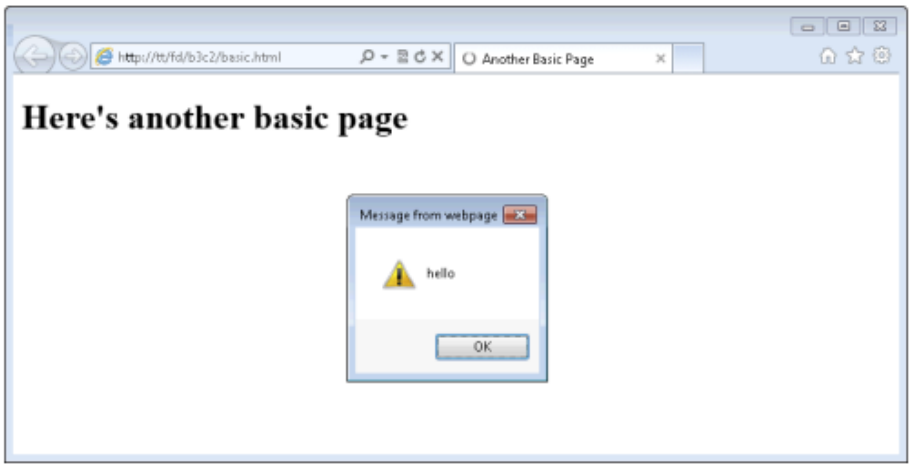
Karena ini mungkin pertama kalinya kita mengenal pemrograman dalam bentuk apa pun, bagian ini dimulai dengan beberapa informasi dasar untuk mempercepat Anda.

**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.

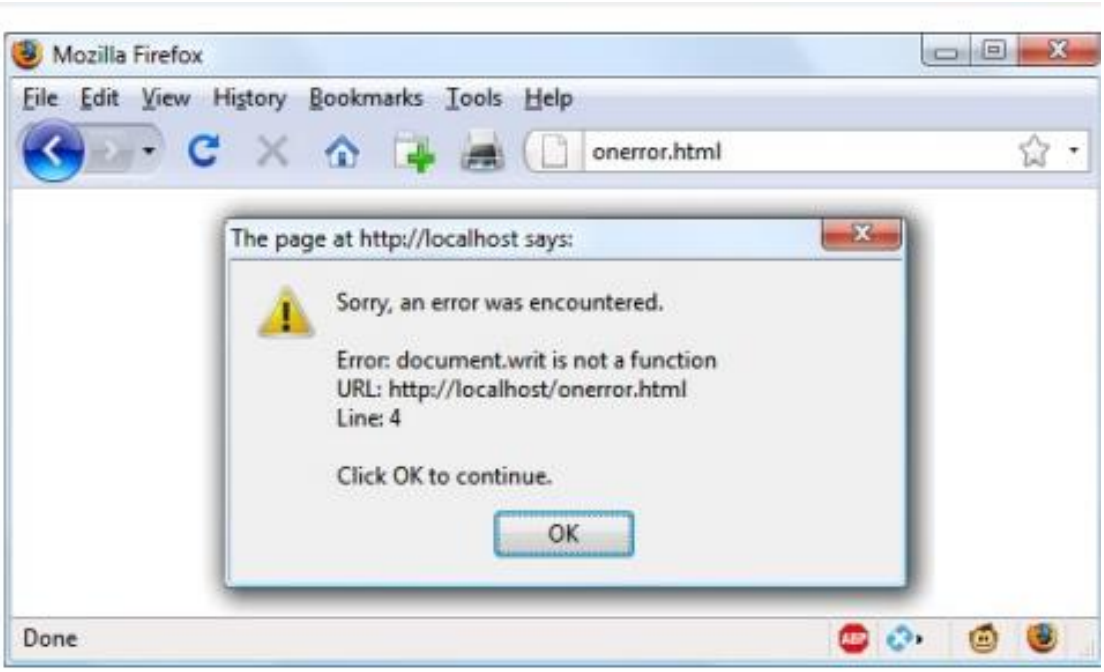
**Menggunakan onerror**

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")
```