```
break
case "About":
document.write("You selected About")
break
case "News":
document.write("You selected News")
break
case "Login":
document.write("You selected Login")
break
case "Links":
document.write("You selected Links")
break
}
</script>
```

Halaman variabel hanya disebutkan satu kali di awal pernyataan switch. Setelah itu, perintah case akan memeriksa kecocokan. Ketika satu terjadi, pernyataan kondisional yang cocok dijalankan. Tentu saja, program nyata akan memiliki kode di sini untuk ditampilkan atau melompat ke halaman, daripada hanya memberi tahu pengguna apa yang dipilih.

Breaking out

Seperti yang kita lihat pada Contoh 7, seperti halnya PHP, perintah break memungkinkan kode kita untuk keluar dari pernyataan switch setelah kondisi terpenuhi. Ingatlah untuk menyertakan jeda kecuali jika kita ingin melanjutkan mengeksekusi pernyataan di bawah kasus berikutnya.

Tindakan default

Ketika tidak ada kondisi yang terpenuhi, kita dapat menentukan tindakan default untuk pernyataan switch menggunakan kata kunci default. Contoh 8 menunjukkan potongan kode yang dapat dimasukkan ke dalam Contoh 7.

Contoh 7 Pernyataan default untuk ditambahkan ke Contoh 6

```
default:
document.write("Unrecognized selection")
break
```

? Operator

Operator ternary (?), dikombinasikan dengan karakter :, menyediakan cara cepat untuk melakukan tes if ... else. Dengannya kita dapat menulis ekspresi untuk dievaluasi, lalu ikuti dengan ? simbol dan kode untuk dieksekusi jika ekspresi benar. Setelah itu, tempatkan a : dan kode yang akan dieksekusi jika ekspresi bernilai false. Pernyataan telah dipecah menjadi beberapa baris untuk kejelasan, tetapi kita akan lebih cenderung menggunakan pernyataan seperti itu pada satu baris.

Looping

Sekali lagi, kita akan menemukan banyak kesamaan yang dekat antara JavaScript dan PHP dalam hal perulangan. Kedua bahasa mendukung while, do ... while, dan for loop.

While Loop

While loop JavaScript pertama-tama memeriksa nilai ekspresi dan mulai mengeksekusi pernyataan di dalam perulangan hanya jika ekspresi itu benar. Jika salah, eksekusi melompat ke pernyataan JavaScript berikutnya (jika ada).

Setelah menyelesaikan iterasi dari loop, ekspresi diuji lagi untuk melihat apakah itu benar dan proses berlanjut sampai ekspresi bernilai salah, atau sampai eksekusi dihentikan. Contoh 9 menunjukkan lingkaran.

Contoh 9 While loop

```
<script>
counter=0
while (counter < 5)
{
document.write("Counter:  " + counter + "<br>")
++counter
}
</script>
```

Skrip ini menghasilkan yang berikut:

Counter: 0
Counter: 1
Counter: 2
Counter: 3
Counter: 4

Jika penghitung variabel tidak bertambah dalam loop, sangat mungkin beberapa browser menjadi tidak responsif karena loop yang tidak pernah berakhir, dan halaman mungkin bahkan tidak mudah dihentikan dengan tombol Escape atau Stop. Jadi berhati-hatilah dengan loop JavaScript Anda.

do ... while Loop

Bila kita memerlukan perulangan untuk melakukan iterasi setidaknya sekali sebelum pengujian dilakukan, gunakan perulangan do ... while, yang mirip dengan while loop, kecuali bahwa ekspresi uji diperiksa hanya setelah setiap iterasi perulangan. Jadi, untuk menampilkan tujuh hasil pertama dalam tabel perkalian tujuh, kita dapat menggunakan kode seperti pada Contoh 10.

Contoh 10. do ... while loop

```
<script>
count = 1
do
{
document.write(count + " times 7 is " + count * 7 + "<br>")
} while (++count <= 7)
</script>
```

Seperti yang kita harapkan, loop ini menghasilkan yang berikut:

1 times 7 is 7
2 times 7 is 14

3 times 7 is 21
4 times 7 is 28
5 times 7 is 35
6 times 7 is 42
7 times 7 is 49

For Loop

For loop menggabungkan yang terbaik dari semua dunia ke dalam satu konstruksi looping yang memungkinkan kita untuk melewatkan tiga parameter untuk setiap pernyataan:

- Ekspresi inisialisasi
- Ekspresi kondisi
- Ekspresi modifikasi

Ini dipisahkan oleh titik koma, seperti ini: for (expr1 ; expr2 ; expr3). Pada awal iterasi pertama dari loop, ekspresi inisialisasi dijalankan. Dalam kasus kode untuk tabel perkalian untuk 7, count akan diinisialisasi ke nilai 1. Kemudian, setiap kali di sekitar loop, ekspresi kondisi (dalam hal ini, count <= 7) diuji, dan loop adalah dimasukkan hanya jika kondisinya benar. Akhirnya, pada akhir setiap iterasi, ekspresi modifikasi dieksekusi. Dalam kasus tabel perkalian untuk 7, jumlah variabel bertambah. Contoh 11 menunjukkan seperti apa kode itu nantinya.

Contoh 11. Menggunakan for loop

```
<script>
for (count = 1 ; count <= 7 ; ++count)
{
document.write(count + "times 7 is " + count * 7 + "<br>");
}
</script>
```

Seperti di PHP, kita dapat menetapkan beberapa variabel dalam parameter pertama untuk loop dengan memisahkannya dengan koma, seperti ini:

```
for (i = 1, j = 1 ; i < 10 ; i++)
```

Demikian juga, kita dapat melakukan beberapa modifikasi pada parameter terakhir, seperti ini:

```
for (i = 1 ; i < 10 ; i++, --j)
```

Atau kita dapat melakukan keduanya secara bersamaan:

```
for (i = 1, j = 1 ; i < 10 ; i++, --j)
```

Keluar dari Loop

Perintah break, yang akan kita ingat penting di dalam pernyataan switch, juga tersedia di dalam for loop. kita mungkin perlu menggunakan ini, misalnya, saat mencari kecocokan. Setelah kecocokan ditemukan, kita tahu bahwa melanjutkan pencarian hanya akan membuang waktu dan membuat pengunjung kita menunggu. Contoh 12 menunjukkan bagaimana menggunakan perintah break.

Contoh 12 Menggunakan perintah break dalam for loop

```
<script>
haystack = new Array()
haystack[17] = "Needle"
for (j = 0 ; j < 20 ; ++j)
{
if (haystack[j] == "Needle")
{
document.write("<br>- Found at location " + j)
break
}
else document.write(j + ", ")
}
</script>
```

Skrip ini menghasilkan yang berikut:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16,
- Found at location 17

Pernyataan lanjutan

Terkadang kita tidak ingin sepenuhnya keluar dari loop, tetapi ingin melewatkan pernyataan yang tersisa hanya untuk iterasi loop ini. Dalam kasus seperti itu, kita dapat menggunakan perintah continue. Contoh 13 menunjukkan ini sedang digunakan.

Contoh 13 Menggunakan perintah continue dalam for loop

```
<script>
haystack = new Array()
haystack[4] = "Needle"
haystack[11] = "Needle"
haystack[17] = "Needle"
for (j = 0 ; j < 20 ; ++j)
{
if (haystack[j] == "Needle")
{
document.write("<br>- Found at location " + j + "<br>")
continue
}
document.write(j + ", ")
}
</script>
```

Perhatikan bagaimana panggilan document.write kedua tidak harus diapit oleh pernyataan else (seperti yang terjadi sebelumnya), karena perintah continue akan melewatkannya jika kecocokan telah ditemukan. Output dari skrip ini adalah sebagai berikut:

0, 1, 2, 3,
- Found at location 4
5, 6, 7, 8, 9, 10,

- Found at location 11
12, 13, 14, 15, 16,
- Found at location 17
18, 19,

Eksplisit casting

Tidak seperti PHP, JavaScript tidak memiliki casting tipe eksplisit seperti (int) atau (float). Sebagai gantinya, ketika kita membutuhkan nilai untuk tipe tertentu, gunakan salah satu fungsi bawaan JavaScript, yang ditunjukkan pada Tabel 7.1.

Tabel 7.1 Fungsi pengubah tipe JavaScript

Perubahan ke tipe	**Fungsi yang akan digunakan**
Int, Integer	**parseInt()**
Bool, Boolean	**Boolean()**
Float, Double, Real	**parseFloat()**
String	**String()**
Array	**split()**

Jadi, misalnya, untuk mengubah angka floating-point menjadi integer, kita bisa menggunakan kode seperti berikut (yang menampilkan nilai 3):
Atau kita dapat menggunakan bentuk majemuk:

```
n = 3.1415927
i = parseInt(n)
document.write(i)
```

Itu saja untuk aliran dan ekspresi kontrol. Bab berikutnya berfokus pada penggunaan fungsi, objek, dan array di JavaScript:

```
document.write(parseInt(3.1415927))
```

7.2 FUNGSI JAVASCRIPT, OBJEK DAN ARRAY

Sama seperti PHP, JavaScript menawarkan akses ke fungsi dan objek. Faktanya, JavaScript sebenarnya didasarkan pada objek, karena—seperti yang telah kita lihat—ia harus mengakses DOM, yang membuat setiap elemen dokumen HTML tersedia untuk dimanipulasi sebagai objek. Penggunaan dan sintaksnya juga sangat mirip dengan PHP, jadi kita akan merasa seperti di rumah sendiri saat saya membawa kita menggunakan fungsi dan objek dalam JavaScript, serta melalui eksplorasi mendalam tentang penanganan array.

Fungsi JavaScript

Selain memiliki akses ke lusinan fungsi (atau metode) bawaan seperti menulis, yang telah kita lihat digunakan di document.write, kita dapat dengan mudah membuat fungsi kita sendiri. Setiap kali kita memiliki bagian kode yang lebih kompleks yang kemungkinan akan digunakan kembali, kita memiliki kandidat untuk suatu fungsi.

Mendefinisikan Fungsi

Sama seperti PHP, JavaScript menawarkan akses ke fungsi dan objek. Faktanya, JavaScript sebenarnya didasarkan pada objek, karena—seperti yang telah kita lihat—ia harus mengakses DOM, yang membuat setiap elemen dokumen HTML tersedia untuk dimanipulasi sebagai objek. Penggunaan dan sintaksnya juga sangat mirip dengan PHP, jadi kita akan merasa seperti di rumah sendiri saat saya membawa kita menggunakan fungsi dan objek dalam JavaScript, serta melalui eksplorasi mendalam tentang penanganan array.
Baris pertama sintaks menunjukkan bahwa:

- Definisi dimulai dengan kata fungsi.
- Nama mengikuti yang harus dimulai dengan huruf atau garis bawah, diikuti dengan sejumlah huruf, angka, simbol dolar, atau garis bawah.
- Tanda kurung diperlukan.
- Satu atau beberapa parameter, dipisahkan dengan koma, bersifat opsional (ditunjukkan dengan kurung siku, yang bukan merupakan bagian dari sintaks fungsi).

Nama fungsi peka huruf besar/kecil, jadi semua string berikut merujuk ke fungsi yang berbeda: getInput, GETINPUT, dan getinput. Dalam JavaScript ada konvensi penamaan umum untuk fungsi: huruf pertama dari setiap kata dalam nama dikapitalisasi kecuali untuk huruf pertama, yaitu huruf kecil. Oleh karena itu, dari contoh sebelumnya, getInput akan menjadi nama pilihan yang digunakan oleh sebagian besar programmer. Konvensi ini biasanya disebut sebagai bumpyCaps, bumpyCase, atau camelCase. Kurung kurawal pembuka memulai pernyataan yang akan dijalankan saat kita memanggil fungsi; kurung kurawal yang cocok harus menutupnya. Pernyataan ini dapat mencakup satu atau lebih pernyataan pengembalian, yang memaksa fungsi untuk menghentikan eksekusi dan kembali ke kode panggilan. Jika suatu nilai dilampirkan ke pernyataan pengembalian, kode panggilan dapat mengambilnya.

7.3 EKSPRESI DAN CONTROL FLOW DI JAVASCRIPT

Di bab sebelumnya, saya memperkenalkan dasar-dasar JavaScript dan DOM. Sekarang saatnya untuk melihat bagaimana membangun ekspresi kompleks dalam JavaScript dan bagaimana mengontrol aliran program skrip kita menggunakan pernyataan bersyarat.

Ekspresi

Ekspresi JavaScript sangat mirip dengan yang ada di PHP. Ekspresi adalah kombinasi nilai, variabel, operator, dan fungsi yang menghasilkan nilai; hasilnya bisa berupa angka, string, atau nilai Boolean (yang dievaluasi menjadi benar atau salah). Contoh 14 menunjukkan beberapa ekspresi sederhana. Untuk setiap baris, itu mencetak huruf antara a dan d, diikuti oleh titik dua dan hasil dari ekspresi. Tag
 ada untuk membuat jeda baris dan memisahkan output menjadi empat baris (ingat bahwa
 dan
 dapat diterima di HTML5, jadi saya memilih untuk menggunakan gaya sebelumnya untuk singkatnya).

Contoh 14. Empat ekspresi Boolean sederhana

```
<script>
document.write("a: " + (42 > 3) + "<br>")
document.write("b: " + (91 < 4) + "<br>")
document.write("c: " + (8 == 2) + "<br>")
document.write("d: " + (4 < 17) + "<br>")
</script>
```

Output dari kode ini adalah sebagai berikut:

a: true
b: false
c: false
d: true

Perhatikan bahwa kedua ekspresi a: dan d: bernilai true. Tapi b: dan c: evaluasi ke false. Tidak seperti PHP (yang masing-masing akan mencetak angka 1 dan tidak ada apa-apa), string true dan false yang sebenarnya ditampilkan. Dalam JavaScript, saat kita memeriksa apakah suatu nilai benar atau salah, semua nilai dievaluasi menjadi true dengan pengecualian berikut ini, yang bernilai false: string false itu sendiri, 0, 0, string kosong, null, undefined, dan NaN (Not a Number, konsep teknik komputer untuk operasi floating-point ilegal seperti pembagian dengan nol). Perhatikan bagaimana saya mengacu pada true dan false dalam huruf kecil. Ini karena, tidak seperti di PHP, nilai-nilai ini harus dalam huruf kecil di JavaScript. Oleh karena itu, hanya yang pertama dari dua pernyataan berikut yang akan ditampilkan, mencetak kata dengan huruf kecil benar, karena yang kedua akan menyebabkan kesalahan 'BENAR' tidak ditentukan:

```
if (1 == true) document.write('true') // True
if (1 == TRUE) document.write('TRUE') // Will cause an error
```

Literal dan Variabel

Bentuk ekspresi yang paling sederhana adalah literal, yang berarti sesuatu yang mengevaluasi dirinya sendiri, seperti angka 22 atau string Tekan Enter. Ekspresi juga bisa berupa variabel, yang mengevaluasi nilai yang telah ditetapkan padanya. Keduanya adalah jenis ekspresi, karena mereka mengembalikan nilai. Contoh 15menunjukkan tiga literal yang berbeda dan dua variabel, yang semuanya mengembalikan nilai, meskipun dari tipe yang berbeda.

Contoh 15. Lima jenis literal

```
<script>
myname = "Peter"
myage = 24
document.write("a: " + 42 + "<br>") // Numeric literal
document.write("b: " + "Hi" + "<br>") // String literal
document.write("c: " + true + "<br>") // Constant literal
document.write("d: " + myname + "<br>") // String variable
document.write("e: " + myage + "<br>") // Numeric variable
</script>
```

Dan, seperti yang kita harapkan, kita melihat nilai pengembalian dari semua ini di output berikut:

```
a: 42
b: Hi
c: true
d: Peter
e: 24
```

Operator memungkinkan kita membuat ekspresi yang lebih kompleks yang mengevaluasi hasil yang bermanfaat. Saat kita menggabungkan konstruksi tugas atau aliran kontrol dengan ekspresi, hasilnya adalah pernyataan. Contoh 16 menunjukkan satu dari masing-masing, pertama memberikan hasil dari ekspresi 366 - day_number ke variabel days_to_new_year, dan yang kedua mengeluarkan pesan ramah hanya jika ekspresi days_to_new_year < 30 bernilai true.

Contoh 16 Dua pernyataan JavaScript sederhana

```
<script>
days_to_new_year = 366 - day_number;
if (days_to_new_year < 30) document.write("It's nearly New Year")
</script>
```

Memegang beberapa nilai dalam array

Variabel memegang satu hal dan mereka melakukannya dengan baik, tetapi ada kalanya kita ingin menyimpan banyak hal. Tentu, kita bisa membuat beberapa variabel, satu untuk setiap hal. kita juga bisa membuat array. Array adalah jenis variabel khusus yang digunakan untuk menampung banyak nilai. Berikut ini contohnya:

```
var myArray = [“Steve”,”Jakob”,”Rebecca”,”Owen”];
```

Array ini berisi empat hal, yang dikenal sebagai elemen. kita akan melihat lebih banyak tentang array nanti, ketika kami memberi tahu kita tentang loop.

Membuat string untuk melacak kata-kata

Saat kita menempatkan kata dalam tanda kutip di JavaScript, kita membuat apa yang disebut string. Biasanya untuk menempatkan konten string ke dalam variabel, seperti ini:

```
var aString = “This is a string.”;
```

String dapat berisi angka, dan ketika kita memasukkan angka dalam tanda kutip, itu akan menjadi string. Kuncinya adalah kutipan, seperti yang ditunjukkan di sini:

```
var anotherString = “This is more than 5 letters long!”;
```

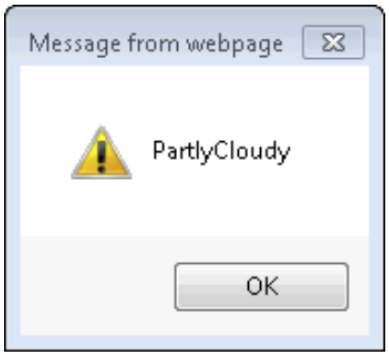
String dapat diapit dalam tanda kutip tunggal atau tanda kutip ganda.

Senar dapat disatukan menggunakan tanda tambah (+), seperti dalam latihan yang akan kita kerjakan. Untuk berlatih membuat string bersambung, mulailah dengan membuka text editor dengan dokumen baru atau kosong. Di text editor, tempatkan HTML dan JavaScript berikut:

```
<!doctype html>
<html>
<head>
<title>JavaScript Chapter 2</title>
</head>
<body>
```

```
<h1>Here's another basic page</h1>
<script type="text/javascript">
 var myString = "Partly" + "Cloudy";
 alert(myString);
</script>
</body>
</html>
```

Simpan file sebagai string.html di root dokumen Anda. Buka browser kita dan lihat halaman dengan membuka http://localhost/string.html. kita akan melihat peringatan seperti pada gambar berikut.



Gambar 7.4 String bersambung.

Perhatikan baik-baik gambar diatas. Perhatikan bahwa tidak ada spasi antara kata Partly dan Cloudy. Untuk memiliki spasi di sana perlu ditambahkan baik di akhir kata Sebagian atau di awal kata Mendung

Bekerja dengan angka

Anda sudah melihat bahwa variabel JavaScript dapat menyimpan angka. kita juga dapat melakukan matematika dengan JavaScript, baik secara langsung pada angka atau melalui variabel. Misalnya, menambahkan dua angka:

```
var myNumber = 4 +
```

Pengurangan dilakukan dengan tanda minus (-), pembagian dengan garis miring (/), dan perkalian dengan tanda bintang (*):

```
//Subtraction
var subtraction = 5 - 3;
//Division
var division = 20 / 5;
//Multiplication
var multiply = 2 * 2;
```

Menguji dengan Persyaratan

Dengan beberapa halaman JavaScript primer selesai, saatnya untuk melihat cara membuat keputusan dengan JavaScript. Keputusan ini disebut kondisional. Cara yang baik untuk menjelaskannya adalah dengan menjelaskan proses pemikiran Steve seputar

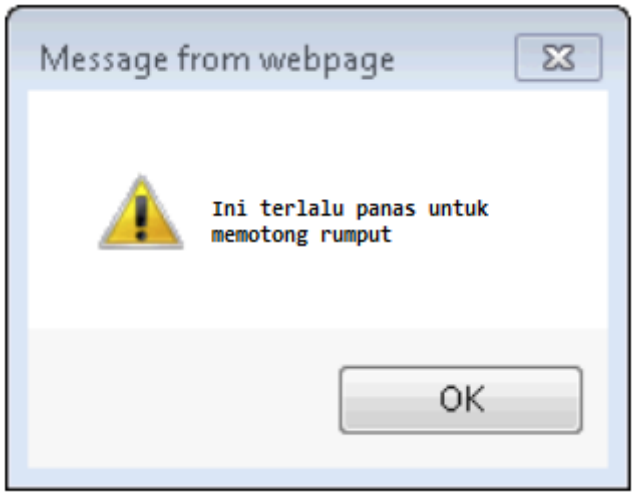
memotong rumput: Jika lebih besar dari 75 derajat, maka terlalu panas untuk memotong. Jika hujan, maka dia tidak bisa memotong rumput. Kalau tidak, dia bisa memotong rumput. Ini dapat diatur dalam JavaScript seperti ini:

```
if (temperatur > 75) {
 alert(“terlalu panas untuk memotong rumput”);
} else if (weather == “hujan”) {
 alert(“Ini hujan, tidak dapat memotong rumb=put”);
} else {
 alert(“Memotong rumput”);
}
```

Sedikit kode itu mengungkapkan semua yang perlu kita ketahui tentang conditional dalam JavaScript! kita menguji suatu kondisi dan kemudian melakukan sesuatu berdasarkan hasil dari kondisi tersebut. Saat kita menyiapkan kondisi, kita menggunakan tanda kurung untuk memuat pengujian dan semua yang kita inginkan terjadi kemudian muncul di antara kurung kurawal buka dan tutup. Kondisional adalah salah satu kasus di mana kita tidak mengakhiri setiap baris dengan titik koma. Inilah latihan yang dapat kita coba untuk bekerja dengan kondisional. Mulailah dengan membuka text editor dengan dokumen baru atau kosong. Di text editor, tempatkan HTML dan JavaScript berikut:

```
<!doctype html>
<html>
<head>
<title>JavaScript Chapter 2</title>
</head>
<body>
<h1>Here’s another basic page</h1>
<script type=”text/javascript”>
 var temperature = 76;
 var weather = “raining”;
 if (temperature > 75) {
 alert(“It’s too hot to mow”);
 } else if (weather == “raining”) {
 alert(“It’s raining, can’t mow”);
 } else {
 alert(“Gotta mow”);
 }
</script>
</body>
</html>
```

Simpan file sebagai cond.html di root dokumen Anda, dan lihat halaman di browser dengan membuka http://localhost/cond.html. kita akan melihat peringatan seperti pada gambar berikut ini.



Gambar 7.5 Peringatan berdasarkan pengujian bersyarat.

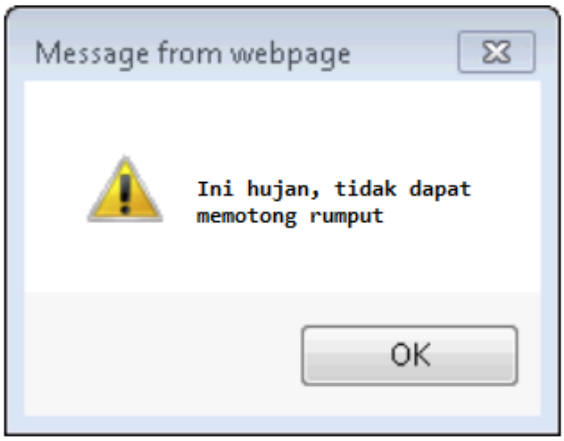
Klik OK untuk mengabaikan peringatan. Untuk melihat bagaimana program merespons saat kita mengubah nilai, di dalam editor, ubah nilai suhu menjadi 70.

Contoh 17 baris yang diubah dicetak tebal:

```
<!doctype html>
<html>
<head>
<title>JavaScript Chapter 2</title>
</head>
<body>
<h1>Here's another basic page</h1>
<script type="text/javascript">
 var temperature = 70;
 var weather = "raining";
 if (temperature > 75) {
 alert("It's too hot to mow");
 } else if (weather == "raining") {
 alert("It's raining, can't mow");
} else {
 alert("Gotta mow");
 }
</script>
</body>
</html>
```

Simpan sebagai cond.html.

Muat ulang halaman di browser kita dengan menekan Ctrl+R atau Command+R. kita akan melihat peringatan seperti pada gambar dibawah ini.



Gambar 7.6 Masuk ke kondisi else if.

Klik OK untuk mengabaikan peringatan. Jika tes gagal, kondisional dapat diatur untuk menjalankan tes lain. Dalam kasus contoh, tes kedua diatur untuk melihat cuaca untuk melihat apakah hujan. Perhatikan penggunaan tanda sama dengan ganda dalam kondisi else if. Terakhir, jika semua pengujian gagal, maka blok kode dapat diatur agar dapat berjalan ketika semuanya gagal. Ini dicatat oleh kata kunci lain dalam contoh kode. Penting untuk dicatat bahwa setelah suatu kondisi benar, dalam contoh setelah suhu lebih besar dari 75, kode di blok itu akan dijalankan tetapi tidak ada kondisi lain yang akan dievaluasi . Ini berarti bahwa tidak ada kode lain di blok lain mana pun yang akan dijalankan.

Melakukan Tindakan Beberapa Kali dengan Loop

Terkadang kita ingin mengulang kode yang sama berulang kali. Ini disebut perulangan, dan JavaScript menyertakan beberapa cara untuk melakukannya, termasuk for dan while.

Untuk apa nilainya

Jika kita ingin melakukan sesuatu beberapa kali dalam JavaScript, cara umum untuk melakukannya adalah dengan for loop. Sebuah for loop memiliki sintaks yang cukup spesifik, seperti yang kita lihat di sini:

```
for (var i = 0; i < 10; i++) {
 // Do something here
}
```

Struktur itu mencakup tiga hal spesifik di dalam tanda kurung.

- Variabel: Pertama, sebuah variabel diatur, dalam hal ini disebut i. Variabel itu diatur ke angka 0.
- Kondisi: Selanjutnya adalah kondisi yang akan diuji. Dalam hal ini, loop menguji apakah variabel i kurang dari 10. Jika i kurang dari 10, kode di dalam kurung kurawal berjalan.
- Operator Postfix: Bagian terakhir dari konstruksi loop for menambah variabel i menggunakan sesuatu yang disebut operator postfix (i++), yang pada dasarnya menaikkan nilai sebesar 1.

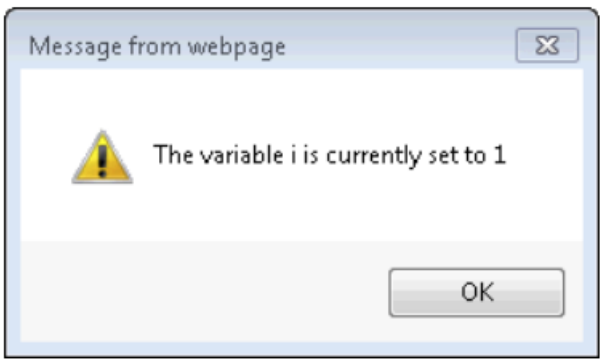
Dalam bahasa sederhana, loop ini membuat variabel dan menyetelnya ke 0, lalu menguji untuk melihat apakah variabel masih kurang dari 10. Jika ya, maka kode di dalam blok akan dieksekusi. Untuk saat ini, kode itu hanyalah sebuah komentar sehingga tidak ada yang terjadi. Jika tidak, maka variabel bertambah 1 dan semuanya dimulai dari awal lagi. Dua bagian pertama dari for loop menggunakan titik koma; bagian terakhir tidak. Pertama kali melalui,

variabel i adalah 0, yang (jelas) kurang dari 10 — dan kode di dalam blok dijalankan sehingga i bertambah 1. Kali berikutnya, nilai i adalah 1, yang masih kurang dari 10, sehingga kode di blok dieksekusi lagi. Ini terus berlanjut sampai nilai I adalah 10. Cobalah dengan latihan. Mulailah dengan membuka text editor dengan dokumen baru atau kosong. Di text editor, tempatkan HTML dan JavaScript berikut:

Contoh 18

```
<!doctype html>
<html>
<head>
<title>JavaScript Chapter 2</title>
</head>
<body>
<h1>Here's another basic page</h1>
<script type="text/javascript">
 for (i = 0; i < 10; i++) {
 alert("The variable i is currently set to " + i);
 }
</script>
</body>
</html>
```

Simpan file sebagai for.html di root dokumen Anda. Lihat file di browser dengan membuka http://localhost/for.html. kita akan melihat serangkaian peringatan, salah satunya ditunjukkan pada gambar berikut:



Gambar 7.7 hasil for loop

Sekarang lihat cara menentukan panjang array. Sebelumnya di bab ini kita melihat array seperti ini:

```
var myArray = ["Steve","Jakob","Rebecca","Owen"]
```

Penggunaan umum for loop adalah untuk memutar array dan melakukan sesuatu dengan setiap nilai. Kondisi dalam contoh for loop yang kita lihat sebelumnya menetapkan nilai pada 10. Tapi bagaimana kita tahu berapa banyak nilai dalam array? Ya, kita dapat dengan mudah menghitung jumlah variabel dalam larik yang ditampilkan, tetapi terkadang kita tidak tahu

berapa banyak elemen dalam larik. kita dapat meminta larik itu sendiri untuk memberi tahu kita berapa lama dengan menanyakannya. kita bertanya melalui properti length, seperti ini:

myArray.length;

Contoh 19 dibawah ini menunjukkan contoh yang mengulang melalui larik yang ditampilkan dan menampilkan setiap elemen.

Contoh 19 Menggunakan for Loop pada Array

```
<!doctype html>
<html>
<head>
<title>JavaScript Chapter 2</title>
</head>
<body>
<h1>Here's another basic page</h1>
<script type="text/javascript">
 var myArray = ["Steve","Jakob","Rebecca","Owen"];
 for (i = 0; i < myArray.length; i++) {
 alert("Hello " + myArray[i]);
 }
</script>
</body>
</html>
```

Kode ini menggunakan standar for loop, kode ini menggunakan properti length untuk mengetahui berapa lama myArrayreally. Setiap kali melalui loop, peringatan ditampilkan, seperti yang ada pada gambar dibawah ini.

Gambar 7.8 penampilan peringatan dari array

Variabel i digunakan tepat di dalam loop, untuk mengakses setiap elemen dari variabel myArray. kita lihat, array adalah daftar hal-hal yang berurutan, dengan urutan yang disediakan oleh angka-angka yang biasanya tidak kita lihat. Angka-angka tersembunyi ini disebut indeks. Indeks elemen pertama dalam array adalah 0 (bukan 1 seperti yang kita harapkan). Dalam contoh ini, karena i adalah 0 pertama kali melalui loop, ia dapat mengakses elemen pertama. Kali kedua melalui loop, seperti yang ditunjukkan pada Gambar 7.5, i sama dengan 1 dan

elemen kedua ditampilkan. Sintaks yang kita lihat di sana, myArray[i], adalah sintaks yang sangat umum yang kita lihat di for loop.

Saat kita di sini

Jenis perulangan lainnya disebut while loop, dan tampilannya seperti ini:

```
while (i < 10) {
 // Do something interesting
 // Don't forget to increment the counter!
}
```

While loop mirip dengan for loop sejauh kode di dalam kurung kurawal dieksekusi selama kondisinya benar. Namun, tidak seperti for loop, kita perlu melakukan sesuatu secara eksplisit di dalam perulangan untuk keluar dari perulangan. Jika kita lupa, kita akan terjebak dalam lingkaran tanpa akhir!

Fungsi

Seperti halnya PHP, fungsi JavaScript digunakan untuk memisahkan bagian kode yang melakukan tugas tertentu. Untuk membuat fungsi, deklarasikan dengan cara yang ditunjukkan pada Contoh 20.

Contoh 20 Deklarasi fungsi sederhana

```
<script>
function product(a, b)
{
return a * b
}
</script>
```

Fungsi ini mengambil dua parameter yang dilewati, mengalikannya, dan mengembalikan produk.

7.4 OBJEK JAVASCRIPT

Objek JavaScript adalah peningkatan dari variabel, yang hanya dapat berisi satu nilai pada satu waktu, di mana objek dapat berisi beberapa nilai dan bahkan fungsi. Sebuah objek mengelompokkan data bersama dengan fungsi yang diperlukan untuk memanipulasinya.

Mendeklarasikan Kelas

Saat membuat skrip untuk menggunakan objek, kita perlu mendesain gabungan data dan kode yang disebut kelas. Setiap objek baru berdasarkan kelas ini disebut instance (atau kejadian) dari kelas itu. Seperti yang telah kita lihat, data yang terkait dengan objek disebut propertinya, sedangkan fungsi yang digunakannya disebut metode. Mari kita lihat cara mendeklarasikan kelas untuk objek bernama Pengguna yang akan berisi detail tentang pengguna saat ini. Untuk membuat kelas, cukup tulis fungsi yang dinamai berdasarkan kelas. Fungsi ini dapat menerima argumen (saya akan menunjukkan nanti bagaimana itu dipanggil) dan dapat membuat properti dan metode untuk objek di kelas itu. Fungsi tersebut disebut konstruktor. Contoh 21 menunjukkan konstruktor untuk kelas Pengguna dengan tiga properti: nama depan, nama pengguna, dan kata sandi. Kelas juga mendefinisikan metode showUser.

Contoh 21 Mendeklarasikan kelas Pengguna dan metodenya

```
<script>
function User(forename, username, password)
{
this.forename = forename
this.username = username
this.password = password
this.showUser = function()
{
document.write("Forename: " + this.forename + "<br>")
document.write("Username: " + this.username + "<br>")
document.write("Password: " + this.password + "<br>")
}
}
</script>
```

Fungsi ini berbeda dari fungsi lain yang telah kita lihat sejauh ini dalam dua cara:

- Ini mengacu pada objek bernama ini. Saat program membuat instance Pengguna dengan menjalankan fungsi ini, ini mengacu pada instance yang sedang dibuat. Fungsi yang sama dapat dipanggil berulang kali dengan argumen yang berbeda, dan akan membuat Pengguna baru setiap kali dengan nilai yang berbeda untuk nama depan properti, dan seterusnya.
- Fungsi baru bernama showUser dibuat di dalam fungsi. Sintaks yang ditampilkan di sini baru dan agak rumit, tetapi tujuannya adalah untuk mengikat showUser ke kelas Pengguna. Dengan demikian, showUser muncul sebagai metode kelas Pengguna.

Konvensi penamaan yang saya gunakan adalah untuk menyimpan semua properti dalam huruf kecil dan menggunakan setidaknya satu karakter huruf besar dalam nama metode, mengikuti konvensi camelCase yang disebutkan sebelumnya dalam bab ini. Contoh 16-5 mengikuti cara yang disarankan untuk menulis konstruktor kelas, yaitu dengan menyertakan metode dalam fungsi konstruktor. Namun, kita juga dapat merujuk ke fungsi yang didefinisikan di luar konstruktor, seperti pada Contoh 22.

Contoh 22 Secara terpisah mendefinisikan kelas dan metode

```
<script>
function User(forename, username, password)
{
this.forename = forename
this.username = username
this.password = password
this.showUser = showUser
}
function showUser()
{
document.write("Forename: " + this.forename + "<br>")
document.write("Username: " + this.username + "<br>")
document.write("Password: " + this.password + "<br>")
```

```
}
</script>
```

Saya tunjukkan formulir ini karena kita pasti akan menemukannya saat membaca kode programmer lain

Membuat Objek

Untuk membuat instance kelas User, kita dapat menggunakan pernyataan seperti berikut:

```
details = new User("Wolfgang", "w.a.mozart", "composer")
```

Atau kita dapat membuat objek kosong, seperti ini:

```
details = new User() and then populat
```

dan kemudian mengisinya nanti, seperti ini:

```
details.forename = "Wolfgang"
details.username = "w.a.mozart"
details.password = "composer"
```

Anda juga dapat menambahkan properti baru ke objek, seperti ini:

```
details.greeting = "Hello"
```

Anda dapat memverifikasi bahwa menambahkan properti baru tersebut berfungsi dengan yang penyataan berikut:

```
document.write(details.greeting)
```

Mengakses Objek

Untuk mengakses suatu objek, kita dapat merujuk ke propertinya, seperti dalam dua contoh pernyataan berikut yang tidak terkait:

```
name = details.forename
if (details.username == "Admin") loginAsAdmin()
```

Jadi, untuk mengakses metode showUser dari objek kelas Pengguna, kita akan menggunakan sintaks berikut, di mana detail objek telah dibuat dan diisi dengan data:

```
details.showUser()
```

Dengan asumsi data yang diberikan sebelumnya, kode ini akan ditampilkan:

```
Forename: Wolfgang
Username: w.a.mozart
```

Password: composer

Kata kunci prototipe

Kata kunci prototipe dapat menghemat banyak memori. Di kelas Pengguna, setiap instance akan berisi tiga properti dan metode. Oleh karena itu, jika kita memiliki 1.000 objek ini di memori, metode showUser juga akan direplikasi 1.000 kali. Namun, karena metodenya identik dalam setiap kasus, kita dapat menentukan bahwa objek baru harus merujuk ke satu instance metode alih-alih membuat salinannya. Jadi, alih-alih menggunakan yang berikut ini di konstruktor kelas:

this.showUser = function()

anda bisa menggantinya dengan ini:

User.prototype.showUser = function()

Contoh 23 menunjukkan seperti apa konstruktor baru itu.

Contoh 23 Mendeklarasikan kelas menggunakan kata kunci prototipe untuk suatu metode

```
<script>
function User(forename, username, password)
{
this.forename = forename
this.username = username
this.password = password
User.prototype.showUser = function()
{
document.write("Forename: " + this.forename + "<br>")
document.write("Username: " + this.username + "<br>")
document.write("Password: " + this.password + "<br>")
}
}
</script>
```

Ini berfungsi karena semua fungsi memiliki properti prototipe, yang dirancang untuk menampung properti dan metode yang tidak direplikasi dalam objek apa pun yang dibuat dari kelas. Sebaliknya, mereka diteruskan ke objeknya dengan referensi. Ini berarti kita dapat menambahkan properti atau metode prototipe kapan saja dan semua objek (bahkan yang sudah dibuat) akan mewarisinya, seperti yang diilustrasikan oleh pernyataan berikut:

```
User.prototype.greeting = "Hello"
document.write(details.greeting)
```

Pernyataan pertama menambahkan properti prototipe salam dengan nilai Hello ke kelas Pengguna. Di baris kedua, detail objek, yang telah dibuat, menampilkan properti baru ini

dengan benar. kita juga dapat menambahkan atau memodifikasi metode di kelas, seperti yang diilustrasikan oleh pernyataan berikut:

```
User.prototype.showUser  = function()
{
document.write("Name " + this.forename +
" User " + this.username +
" Pass " + this.password)
}
details.showUser()
```

Anda dapat menambahkan baris ini ke skrip kita dalam pernyataan bersyarat (seperti jika), sehingga baris tersebut berjalan jika aktivitas pengguna menyebabkan kita memutuskan bahwa kita memerlukan metode showUser yang berbeda. Setelah baris ini dijalankan, meskipun detail objek telah dibuat, panggilan lebih lanjut ke detail.showUser akan menjalankan fungsi baru. Definisi lama showUser telah dihapus.

7.5 MENGGUNAKAN FUNGSI UNTUK MENGHINDARI PENGULANGAN

Praktik pemrograman yang baik adalah menggunakan kembali kode bila memungkinkan. Ini tidak hanya mengurangi jumlah kemungkinan kesalahan dalam kode Anda, tetapi juga mengurangi pekerjaan, yang selalu bagus dalam hal pengkodean. Bagian ini membahas cara utama untuk menerapkan penggunaan kembali kode: fungsi JavaScript menyertakan sejumlah fungsi bawaan. kita telah menggunakan satu di sepanjang bab ini: alert(). Fungsi alert() membuat dialog di browse

Membuat fungsi

Fungsi dibuat dengan kata kunci fungsi diikuti dengan nama fungsi, tanda kurung, dan kurung kurawal buka dan tutup, seperti ini:

```
function myFunction() {
 // Function code goes here
}
```

Apa yang kita lakukan di dalam fungsi terserah Anda. Apa pun yang dapat kita lakukan di luar fungsi dapat kita lakukan di dalamnya. Jika kita menemukan bahwa halaman kita perlu memperbarui banyak HTML, kita dapat menggunakan fungsi sehingga kita tidak perlu terus mengulangi kode yang sama berulang kali.

Menambahkan argumen fungsi

Kekuatan fungsi hadir dengan kemampuannya untuk menerima input, yang disebut argumen, dan kemudian melakukan sesuatu dengan input itu. Misalnya, inilah fungsi sederhana untuk menambahkan dua angka:

```
function addNumbers(num1,num2) {
 alert(num1+num2);
}
```

Fungsi ini menerima dua argumen yang disebut num1 dan num2. Argumen tersebut kemudian digunakan dalam fungsi alert(). kita telah melihat fungsi alert() di sepanjang bab ini dan sekarang kita memahami lebih banyak tentang apa yang terjadi! Fungsi alert() menerima satu argumen, teks untuk ditampilkan dalam dialog alert. Dalam hal ini, karena kita menambahkan dua angka, peringatan menampilkan angka yang dihasilkan. kita mengerjakan latihan untuk ini di bagian berikut.

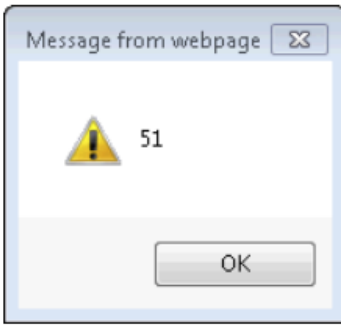
Memanggil fungsi

Membuat fungsi saja tidak cukup; kita perlu menyebutnya juga. Memanggil fungsi berarti kita menjalankannya, sama seperti saat kita memanggil fungsi alert() di awal bab ini. Sampai kita memanggil suatu fungsi, itu tidak benar-benar melakukan apa-apa, seperti fungsi alert() tidak melakukan apa-apa sampai kita memanggilnya.

Memanggil salah satu fungsi kita sendiri terlihat seperti panggilan ke fungsi alert(). Inilah latihan yang kami janjikan. Mulailah dengan membuka text editor dengan dokumen baru atau kosong. Di text editor, tempatkan HTML dan JavaScript berikut:

```
<!doctype html>
<html>
<head>
<title>JavaScript Chapter 2</title>
</head>
<body>
<h1>Here's another basic page</h1>
<script type="text/javascript">
 // Define the function
 function addNumbers(num1,num2) {
 alert(num1+num2);
 }
 // Call the function
 addNumbers(49,2);
</script>
</body>
</html>
```

Simpan file sebagai func.html di root dokumen Anda. Buka browser kita dan arahkan ke http://localhost/func.html. kita akan melihat peringatan seperti yang ditunjukkan pada gambar berikut.



Gambar 7.9 Menjalankan fungsi

Meningkatkan fungsi addNumbers

Fungsi yang telah kita buat, addNumbers(), menerima dua argumen dan menambahkannya. Tetapi bagaimana jika kita mengirim sesuatu yang bukan angka? Fungsi tidak memiliki cara untuk mengujinya dan dengan senang hati mencoba menambahkannya. Untuk bereksperimen dengan ini, ubah 2 dalam panggilan ke addNumbers menjadi "two", seperti ini:

```
addNumbers(49,"two");
```

Saat kita memuat ulang halaman, kita akan melihat peringatan seperti pada Gambar 7.10.

Gambar 7.10 Mencoba menambahkan sesuatu yang bukan angka.

JavaScript menyertakan fungsi untuk menguji apakah sesuatu itu angka. Fungsi tersebut disebut isNaN(), yang merupakan singkatan dari bukan angka. Ini dapat ditambahkan di mana saja yang perlu kita uji untuk memastikan sesuatu adalah angka sebelum bekerja dengannya, seperti dalam kasus fungsi addNumbers() . kita menggunakan fungsi isNaN() dalam kondisi if dan kemudian bereaksi sesuai jika itu bukan angka. Jika kita menyebutnya dengan salah satu dari dua argumen sebagai sesuatu selain angka, kita akan menerima peringatan yang menyatakan itu. Misalnya, jika kita mengubah angka 2 menjadi "two", kita akan menerima peringatan yang ditunjukkan oleh gambar berikut:

Gambar 7.11 Peringatan yang dibuat menggunakan isNaN().

Mengembalikan hasil dari fungsi

Fungsi yang kita buat di bagian ini mengirimkan peringatan. Tetapi ada kalanya kita ingin agar fungsi mengirim sesuatu kembali kepada kita — untuk melakukan sesuatu dan kemudian mengembalikan hasilnya. Kata kunci kembali digunakan untuk mengembalikan hasil dari suatu fungsi. Dalam contoh fungsi addNumbers() yang ditampilkan, alih-alih menggunakan alert() langsung di dalam fungsi, kita bisa mengembalikan hasilnya. Berikut ini contohnya:

```
function addNumbers(num1,num2) {
 var result = num1+num2;
 return result;
 }
```

Anda dapat memanggil fungsi seperti sebelumnya, tetapi sekarang kita perlu menangkap hasilnya, biasanya ke variabel lain, seperti ini:

```
var myResult = addNumbers(49,2);
```

Daftar Mengembalikan Nilai dari Fungsi

```
<!doctype html>
<html>
<head>
<title>JavaScript Chapter 2</title>
</head>
<body>
<h1>Here's another basic page</h1>
<script type="text/javascript">
 // Define the function
 function addNumbers(num1,num2) {
 var result = num1+num2;
 return result;
 }
 // Call the function
 var myResult = addNumbers(49,2);
</script>
</body>
</html>
```

Obyek Secara Singkat

Anda telah melihat array dan bagaimana mereka dapat digunakan untuk menyimpan banyak nilai. Array menyimpan data itu menggunakan indeks bernomor yang tidak terlihat. Di sisi lain, objek dapat menyimpan banyak nilai tetapi nilai tersebut diakses melalui indeks bernama, yang disebut properti. Objek sebenarnya dapat melakukan lebih dari ini, seperti fungsi penahanan (disebut metode) tetapi untuk contoh ini, pertimbangkan fokus sempit ini, menggunakan objek untuk menampung banyak nilai.

Membuat objek

Berikut adalah contoh objek untuk bola:

```
var ball = {
"color": "white",
"type": "baseball"
};
```

Sedangkan array dibuat dengan tanda kurung siku, objek dibuat dengan kurung kurawal, seperti yang ditunjukkan pada contoh. Saat kita mendefinisikan sebuah objek, kita dapat mendefinisikan satu atau lebih properti (mirip dengan elemen dalam array). Dalam contoh ini, kita membuat dua properti, satu disebut warna dan satu lagi disebut tipe. Nilai-nilai tersebut kemudian diatur ke putih dan baseball, masing-masing. kita dapat mengakses objek menggunakan satu titik, seperti:

ball.color

Titik tunggal dikenal sebagai notasi titik bila digunakan dengan cara ini. Daftar dibawah ini menunjukkan HTML untuk membuat objek bola dan menampilkan properti warnanya.

```
<!doctype html>
<html>
<head>
<title>JavaScript Chapter 2</title>
</head>
<body>
<h1>Here's another basic page</h1>
<script type="text/javascript">
 var ball = {
 "color": "white",
 "type": "baseball"
};
 alert(ball.color);
</script>
</body>
</html>
```

Membuat Objek dan Menampilkan Properti

Gambar 7.12 Menampilkan properti objek

Menambahkan properti ke objek

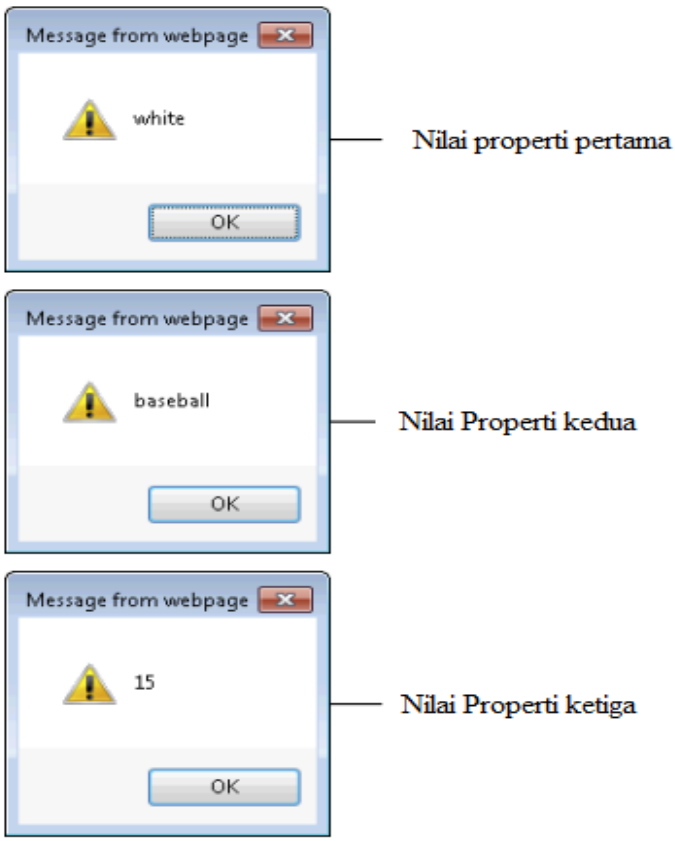
Terkadang kita ingin menambahkan properti ke objek setelah dibuat. Ini juga dapat dilakukan dengan menggunakan notasi titik, seperti:

```
<!doctype html>
```

```
<html>
<head>
<title>JavaScript Chapter 2</title>
</head>
<body>
<h1>Here's another basic page</h1>
<script type="text/javascript">
 var ball = {
 "color": "white",
 "type": "baseball"
 };

 ball.weight = 15;
 for (var prop in ball) {
 alert(ball[prop]);
 }
</script>
</body>
</html>
```

Berikut adalah latihan untuk membuat objek, menambahkannya, dan kemudian mengulangnya menggunakan konstruktor loop jenis baru. Mulailah dengan membuka text editor dengan dokumen baru atau kosong. Simpan ini sebagai obj.html di root dokumen Anda. Lihat halaman di browser dengan membuka http://localhost/obj.html. kita akan melihat tiga peringatan, seperti gambar berikut:



Gambar 7.13 Tiga peringatan properti objek.

Seperti yang kita lihat dari peringatan, properti yang dibuat bersama dengan objek ditampilkan, seperti properti bobot yang ditambahkan kemudian menggunakan notasi titik. Nanti di bab ini, kita akan melihat lebih banyak tentang objek, jadi sedikit latar belakang ini akan membantu.

7.6 BEKERJA DENGAN DOKUMEN HTML

Semua pemrograman JavaScript ini dapat digunakan secara praktis saat kita mulai menambahkannya ke halaman web. JavaScript terintegrasi ke dalam HTML dan memiliki akses ke semua yang ada di halaman web. Ini berarti kita dapat menambahkan HTML ke halaman, menghapusnya, atau mengubahnya, semuanya dengan cepat, dalam waktu nyata. Agar dapat bekerja sama, JavaScript dan HTML membutuhkan bahasa yang sama agar JavaScript dapat mengetahui apa yang harus dilakukan pada suatu halaman. JavaScript dan HTML bekerja sama melalui sesuatu yang disebut *Document Object Model* (DOM). DOM memberikan akses JavaScript ke halaman web sehingga dapat memanipulasi halaman.

Hubungan antara JavaScript dan HTML adalah melalui documentobject. kita melihat objek dokumen yang digunakan di bagian ini bersama dengan fungsi lain untuk mengakses bagian halaman menggunakan JavaScript. Objek dokumen sebenarnya adalah anak dari objek jendela dan ada anak-anak lain yang menarik juga, seperti yang kita lihat nanti di bab ini.

Mengakses HTML dengan JavaScript

Anda mengakses bagian halaman web, objek dokumen, menggunakan fungsi JavaScript. kita dapat melihat bagian halaman untuk melihat teks apa yang ada di dalamnya atau mengubah teks di halaman. kita juga dapat menambahkan ke halaman dengan JavaScript dan banyak lagi.

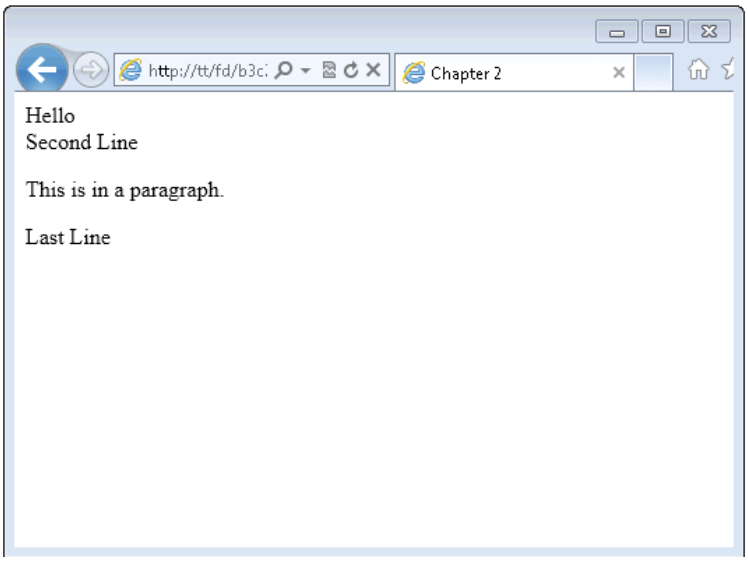
Menggunakan GetElementById untuk mengakses elemen tertentu

Cara paling spesifik agar kita dapat mengakses elemen pada halaman adalah dengan menggunakan ID-nya. Ingat bahwa atribut ID dapat ditempatkan pada elemen apa pun, dan atribut tersebut (seharusnya) unik di seluruh halaman. Dengan cara ini, setiap elemen yang sudah menjadi bagian dari DOM dapat diakses secara langsung daripada dengan melintasi pohon dokumen.

Daftar 24 HTML Dasar untuk Mendemonstrasikan DOM

```
<!doctype html>
<html>
<head>
<title>Chapter 2</title>
</head>
<body>
<div id=”myDiv”>Hello</div>
<div class=”divClass”>Second Line</div>
<div class=”divClass”>
 <p class=”pClass”>This is in a paragraph.</p>
</div>
<div class=”divClass”>Last Line</div>
</body>
</html>
```

HTML itu, ketika dilihat di browser, membuat halaman seperti yang ditunjukkan gambar berikut:



Gambar 7.14 Membuat halaman dasar untuk mendemonstrasikan DOM.

Melihat HTML lagi, kita dapat melihat atribut ID pada pertama pada halaman. kita dapat menggunakan fungsi getElementById untuk mengakses elemen tersebut. kita berkata, "Bagus, saya dapat mengakses elemen tetapi apa yang dapat saya lakukan dengannya?" Senang kita bertanya. Saat mengakses elemen, kita melihat HTML saat ini atau membuat perubahan seperti gaya CSS atau konten sebenarnya dari elemen itu sendiri. Cobalah dalam latihan.

1. Buka text editor dengan dokumen baru atau kosong.
2. Di text editor, tempatkan HTML dan JavaScript berikut:

```
<!doctype html>
<html>
<head>
<title>Chapter 2</title>
</head>
<body>
<div id=”myDiv”>Hello</div>
<div class=”divClass”>Second Line</div>
<div class=”divClass”>
 <p class=”pClass”>This is in a paragraph.</p>
</div>
<div class=”divClass”>Last Line</div>
<script type=”text/javascript”>
 var theDiv = document.getElementById(“myDiv”);
 alert(“The content is “ + theDiv.innerHTML);
</script>
</body>
</html>
```

3. Simpan file sebagai getbyid.html di root dokumen Anda.

4. Buka browser web kita dan lihat halaman di http://localhost/getbyid.html.
   Anda akan melihat peringatan seperti yang ditunjukkan oleh gambar berikut ini:

Gambar 7.15 Peringatan yang dihasilkan oleh fungsi getElementById.

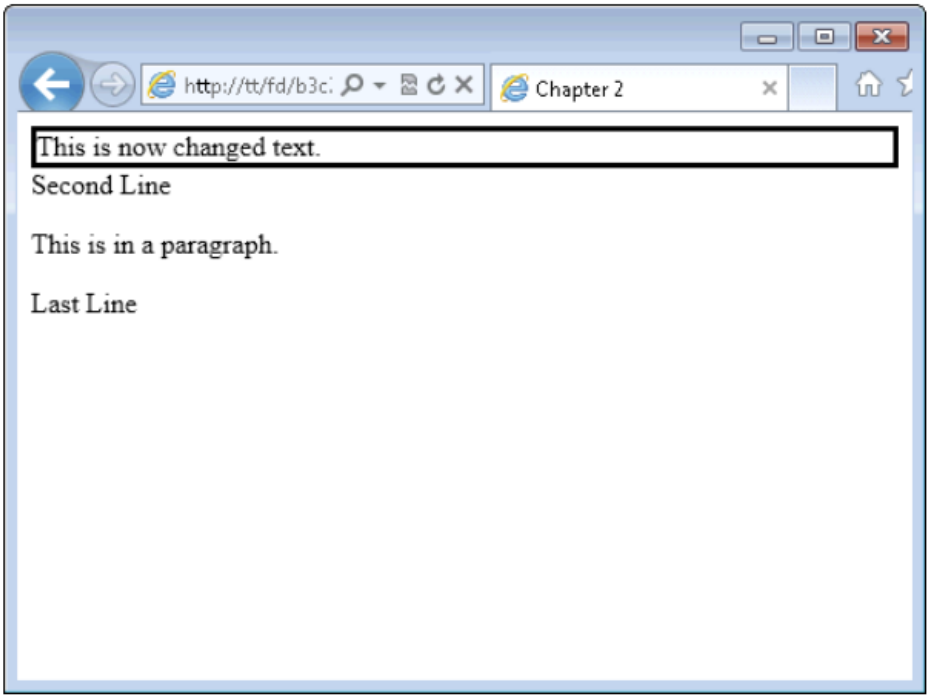
5. Klik OK untuk mengabaikan peringatan dan menutup browser Anda.
6. Kembali ke getbyid.html di editor Anda, hapus baris JavaScript yang dimulai dengan alert(. Ganti baris itu dengan dua ini:

```
theDiv.style.border = “3px solid black”;
theDiv.innerHTML = “This is now changed text.”;
```

Seluruh blok skrip sekarang akan terlihat seperti ini:

```
<script type=”text/javascript”>
 var theDiv = document.getElementById(“myDiv”);
 theDiv.style.border = “3px solid black”;
 theDiv.innerHTML = “This is now changed text.”;
</script>
```

Jika dilihat di browser, halaman tersebut sekarang terlihat seperti pada Gambar 2-16. Perhatikan secara khusus bahwa teks dari baris atas telah diubah dan sekarang memiliki batas. Dalam latihan ini, kita membuat HTML dan JavaScript. Dalam JavaScript, kita mengakses elemen HTML menggunakan fungsi getElementById. Dari sana kita menampilkannya menggunakan alert(). Bagian kedua dari latihan ini melihat kita mengubah konten elemen menggunakan innerHTML dan juga mengubah gaya CSS elemen menggunakan properti style.border.



Gambar 7.16 Halaman setelah diubah dengan getElementById.

Anda sekarang telah melihat cara menggunakan getElementById sebagai bagian dari DOM di JavaScript, jadi periksa salah satu dari daftar ember Anda. Ada baiknya untuk memahami bahwa getElementById ada jika kita perlu bekerja dengan JavaScript orang lain. Namun, ada cara yang lebih baik untuk bekerja dengan halaman web melalui JavaScript dan itu disebut jQuery. kita belajar tentang jQuery di bab selanjutnya. Untuk saat ini, nikmati kemenangan kita atas DOM.

7.7 MODEL OBJEK DOKUMEN

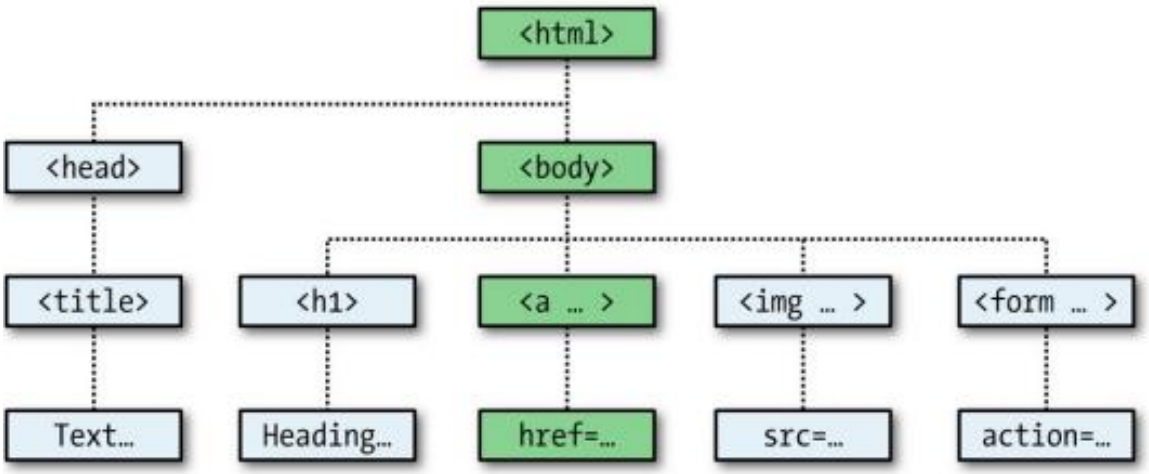
Para desainer JavaScript sangat cerdas. Daripada hanya membuat bahasa skrip lain (yang masih merupakan peningkatan yang cukup bagus pada saat itu), mereka memiliki visi untuk membangunnya di sekitar Model Objek Dokumen, atau DOM. Ini memecah bagian dari dokumen HTML menjadi objek diskrit, masing-masing dengan properti dan metodenya sendiri dan masing-masing tunduk pada kontrol JavaScript. JavaScript memisahkan objek, properti, dan metode menggunakan titik (salah satu alasan bagus mengapa + adalah operator rangkaian string dalam JavaScript, bukan titik). Misalnya, mari kita pertimbangkan kartu nama sebagai objek yang akan kita sebut kartu. Objek ini berisi properti seperti nama, alamat, nomor telepon, dan sebagainya. Dalam sintaks JavaScript, properti ini akan terlihat seperti ini:

```
card.name
card.phone
card.address
```

Metodenya adalah fungsi yang mengambil, mengubah, dan bertindak berdasarkan properti. Misalnya, untuk memanggil metode yang menampilkan properti kartu objek, kita dapat menggunakan sintaks seperti:

```
card.display()
```

Lihat beberapa contoh sebelumnya dalam bab ini dan lihat di mana pernyataan document.write digunakan. Sekarang setelah kita memahami bagaimana JavaScript didasarkan pada objek, kita akan melihat bahwa menulis sebenarnya adalah metode objek dokumen. Di dalam JavaScript, ada hierarki objek induk dan anak, yang dikenal sebagai Model Objek Dokumen.



Gambar 7.17 Contoh hierarki objek DOM

Gambar tersebut menggunakan tag HTML yang sudah kita kenal untuk menggambarkan hubungan induk/anak antara berbagai objek dalam dokumen. Misalnya, URL di dalam tautan adalah bagian dari badan dokumen HTML. Dalam JavaScript, direferensikan seperti ini:

```
url = document.links.linkname.href
```

Perhatikan bagaimana ini mengikuti kolom tengah ke bawah. Bagian pertama, dokumen, mengacu pada tag <html> dan <body>; link.linkname ke tag <a>; dan href ke atribut href. Mari kita ubah ini menjadi beberapa HTML dan skrip untuk membaca properti tautan. Simpan Contoh 25 sebagai linktest.html, lalu panggil di browser Anda.

Jika kita menggunakan Microsoft Internet Explorer sebagai browser pengembangan utama Anda, baca bagian ini terlebih dahulu (tanpa mencoba contohnya), lalu baca bagian yang berjudul Tapi Tidak Sesederhana itu, dan terakhir kembali ke sini dan coba contoh dengan modifikasi getElementById yang dibahas di sana. Tanpa itu, contoh ini tidak akan bekerja untuk Anda.

Contoh 25 Membaca URL tautan dengan JavaScript

```
<html>
<head>
<title>Link Test</title>
</head>
<body>
<a id="mylink" href="http://mysite.com">Click me</a><br>
<script>
url = document.links.mylink.href
document.write('The URL is ' + url)
```

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

Perhatikan bentuk pendek dari tag <script> di mana saya telah menghilangkan parameter type="text/JavaScript" untuk menghemat pengetikan. Jika diinginkan, hanya untuk tujuan pengujian ini (dan contoh lainnya), kita juga dapat menghilangkan semua yang ada di luar tag <script> dan </script>. Output dari contoh ini adalah:

Click me

The URL is http://mysite.com

Baris keluaran kedua berasal dari metode document.write. Perhatikan bagaimana kode mengikuti pohon dokumen turun dari dokumen ke tautan ke mylink (id yang diberikan ke tautan) ke href (nilai tujuan URL). Ada juga formulir pendek yang berfungsi sama baiknya, yang dimulai dengan nilai dalam atribut id: mylink.href. Jadi kita bisa mengganti ini:

```
url = document.links.mylink.href
```

dengan berikut ini:

```
url = mylink.href
```

Tapi Tidak Sesederhana Itu

Jika kita mencoba Contoh 25 di Safari, Firefox, Opera, atau Chrome, itu akan bekerja dengan sangat baik. Tetapi di Internet Explorer itu akan gagal, karena implementasi JavaScript Microsoft, yang disebut JScript, memiliki banyak perbedaan halus dari standar yang diakui. Selamat datang di dunia pengembangan web tingkat lanjut! Jadi apa yang bisa kita lakukan tentang ini? Nah, dalam kasus ini, alih-alih menggunakan tautan objek anak dari objek dokumen induk, yang ditolak oleh Internet Explorer, kita harus menggantinya dengan metode untuk mengambil elemen dengan id-nya. Oleh karena itu, baris berikut:

```
url = document.links.mylink.href
```

bisa diganti dengan yang ini :

```
url = document.getElementById('mylink').href
```

Dan sekarang skrip akan berfungsi di semua browser utama. Kebetulan, ketika kita tidak perlu mencari elemen berdasarkan id, formulir singkat berikut akan tetap berfungsi di Internet Explorer, serta browser lainnya:

```
url = mylink.href
```

Penggunaan Lain untuk Simbol $

Seperti disebutkan sebelumnya, simbol $ diperbolehkan dalam variabel JavaScript dan nama fungsi. Karena itu, terkadang kita mungkin menemukan kode yang tampak aneh seperti ini:

```
url = $('mylink').href
```

Beberapa programmer yang giat telah memutuskan bahwa fungsi getElementById begitu lazim dalam JavaScript sehingga mereka telah menulis sebuah fungsi untuk menggantikannya yang disebut $, yang ditunjukkan pada Contoh 26.

Contoh 26 Fungsi pengganti untuk metode getElementById

```
<script>
function $(id)
{
return document.getElementById(id)
}
</script>
```

Oleh karena itu, selama kita telah memasukkan fungsi $ dalam kode Anda, sintaks seperti:

```
$('mylink').href
```

dapat mengganti kode seperti:

```
document.getElementById('mylink').hre
```

Menggunakan DOM

Objek link sebenarnya adalah larik URL, jadi URL mylink pada Contoh 25 juga dapat dirujuk dengan aman di semua browser dengan cara berikut (karena ini adalah link pertama dan satu-satunya):

```
url = document.links[0].href
```

Jika kita ingin mengetahui berapa banyak tautan yang ada di seluruh dokumen, kita dapat menanyakan properti panjang dari objek tautan seperti ini:

```
numlinks = document.links.length
```

Karena itu kita dapat mengekstrak dan menampilkan semua tautan dalam dokumen seperti ini:

```
for (j=0 ; j < document.links.length ; ++j)
document.write(document.links[j].href + '<br>')
```

Panjang sesuatu adalah properti dari setiap array, dan banyak objek juga. Misalnya, jumlah item dalam riwayat web browser kita dapat ditanyakan seperti ini:

```
document.write(history.length)
```

Namun, untuk menghentikan situs web mengintip riwayat penjelajahan Anda, objek riwayat hanya menyimpan jumlah situs dalam larik: kita tidak dapat membaca atau menulis ke nilai ini. Tetapi kita dapat mengganti halaman saat ini dengan halaman dari riwayat, jika kita tahu posisinya di dalam riwayat. Ini bisa sangat berguna dalam kasus di mana kita tahu bahwa halaman tertentu dalam riwayat berasal dari situs Anda, atau kita hanya ingin mengirim browser kembali satu atau beberapa halaman, yang kita lakukan dengan metode go dari objek history. Misalnya, untuk mengirim browser kembali tiga halaman, jalankan perintah berikut:

```
history.go(-3)
```

Anda juga dapat menggunakan metode berikut untuk memundurkan atau meneruskan halaman sekaligus:

```
history.back()
history.forward()
```

Dengan cara yang sama, kita dapat mengganti URL yang sedang dimuat dengan salah satu pilihan Anda, seperti ini:

```
document.location.href = 'http://google.com'
```

Tentu saja, ada lebih banyak hal di DOM daripada membaca dan memodifikasi tautan. Saat kita maju melalui bab-bab berikut tentang JavaScript, kita akan menjadi cukup akrab dengan DOM dan cara mengaksesnya.

Bekerja dengan Browser Web

Primer JavaScript ini diakhiri dengan tampilan cepat pada tampilan JavaScript dari browser web. Seperti yang baru saja kita lihat, saat halaman dimuat, objek dokumen memberikan tampilan halaman ke JavaScript. Demikian juga, beberapa objek lain memberikan JavaScript tampilan browser web itu sendiri. Dengan menggunakan objek ini, yang merupakan anak dari objek jendela, kita dapat melakukan hal-hal seperti mendeteksi jenis browser yang digunakan pengunjung dan juga mengarahkan pengguna ke browser lain. halaman web sepenuhnya.

Mendeteksi browser

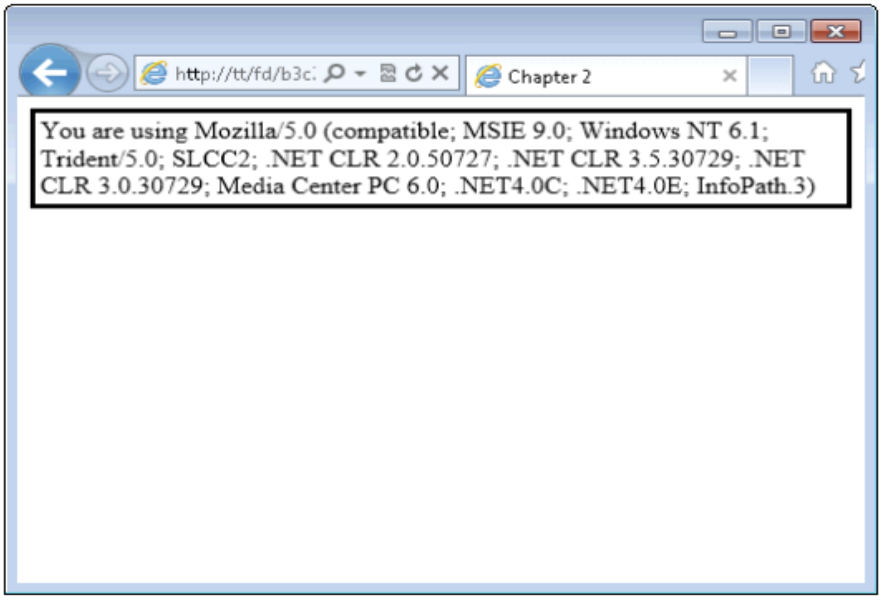
Objek navigator digunakan untuk mendeteksi hal-hal tentang browser pengunjung, seperti versi apa itu. Informasi ini dapat digunakan untuk menyajikan halaman atau layout tertentu kepada pengguna.

Contoh 27 Menampilkan User agent

```
<!doctype html>
<html>
<head>
<title>Chapter 2</title>
</head>
<body>
```

```
<div id=”output”></div>
<script type=”text/javascript”>
 var outputDiv = document.getElementById(“output”);
 outputDiv.style.border  = “3px solid black”;
 outputDiv.style.padding  = “3px”;
 var userAgent = navigator.userAgent;
 outputDiv.innerHTML = “You are using “ + userAgent;
</script>
</body>
</html>
```

Jika dilihat di browser, outputnya terlihat seperti pada gambar berikut.



Gambar 7.18 Melihat properti userAgent

Perhatikan bahwa jika kita menjalankan kode ini, versi browser kita kemungkinan akan berbeda dari ini.

Keterbatasan deteksi browser

Saat kita menggunakan metode seperti yang ditunjukkan di sini, kita perlu menyadari bahwa itu tidak selalu akurat. Mendeteksi browser dengan cara ini hanya bergantung pada apa yang diklaim browser dan informasi ini dapat dipalsukan oleh pengguna. Oleh karena itu, ketika kita menggunakan objek navigator (atau metode “User Agent Sniffer" lainnya), kita harus menyadari bahwa ada batasan pada keakuratannya dan ini jelas tidak 100% sangat mudah.

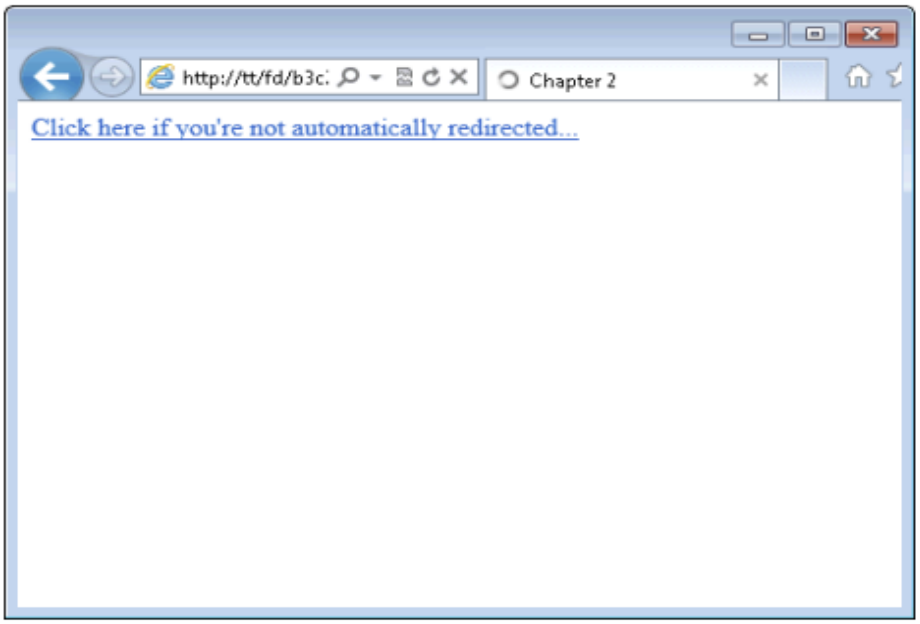
Mengarahkan ke halaman lain

Anda mungkin pernah menemukan satu di suatu tempat di sepanjang jalan, halaman yang mengatakan "Click here if you’re not automatically redirected" dan kemudian secara otomatis mengalihkan Anda. Pernahkah kita bertanya-tanya mengapa mereka repot-repot dengan bagian "Click Here"? Itu dilakukan jika browser kita tidak mengaktifkan JavaScript. Bagian ini menunjukkan kode untuk halaman tersebut. Objek lokasi menyediakan kemampuan untuk mengarahkan ulang ke halaman lain, dan ini adalah salah satu halaman JavaScript paling sederhana yang pernah kita tulis. Daftar dibawah ini menunjukkan HTML dan JavaScript.

Contoh 28 Daftar Halaman Pengalihan

```
<!doctype html>
<html>
<head>
<title>Chapter 2</title>
</head>
<body>
<div>
 <a href=”http://www.braingia.org”> Click here if
you’re not automatically redirected...
 </a>
</div>
<script type=”text/javascript”>
 window.location.replace(“http://www.braingia.org”);
</script>
</body>
</html>
```

Halaman yang dilihat di browser terlihat seperti pada gambar diabwah, tetapi hanya untuk waktu yang singkat. Bahkan, kita mungkin tidak melihatnya sebelum dialihkan!



Gambar 7.19 Halaman pengalihan sesaat sebelum dialihkan.

HTML untuk halaman ini cukup menyiapkan elemen <a> dasar dengan tautan, tidak perlu JavaScript. Kemudian JavaScript menggunakan location.replaceobject untuk mengirim pengguna ke halaman yang berbeda. Hampir tidak ada apa-apanya! Ada lebih banyak objek navigator dan lokasi di JavaScript. Untuk informasi lebih lanjut tentang objek navigator, buka halaman ini di Jaringan Pengembang Mozilla:
https://developer.mozilla.org/en-US/docs/DOM/window.navigator
Untuk informasi lebih lanjut tentang objek lokasi, lihat halaman ini:
https://developer.mozilla.org/en-US/docs/DOM/window.location

BAB 8
MENAMBAHKAN JQUERY

jQuery adalah perpustakaan JavaScript. Oke, itu mungkin tidak masuk akal. Apa itu perpustakaan JavaScript? Pustaka JavaScript adalah kumpulan kode yang kita gunakan saat kita ingin mendapatkan akses ke fungsionalitas tambahan atau membuat hidup lebih mudah. jQuery melakukan keduanya. jQuery hanyalah JavaScript yang kita tambahkan ke halaman web kita untuk membuat penulisan JavaScript lebih mudah. kita masih menggunakan JavaScript dengan jQuery, jadi semua yang kita pelajari di Bab 2 tidak sia-sia. Namun, ada hal-hal tertentu yang dilakukan jQuery jauh lebih baik daripada JavaScript lama biasa. Bekerja dengan halaman web adalah salah satunya. Misalnya, di mana kita mungkin menggunakan getElementById, jQuery memiliki hal-hal yang disebut selector yang memungkinkan cara yang jauh lebih kuat untuk mengakses hal-hal di halaman web untuk digunakan JavaScript. Bab ini menjelaskan cara memulai dengan jQuery dan kemudian menunjukkan beberapa contoh menggunakan jQuery.

8.1 PENGENALAN JQUERY

jQuery cukup populer. Meskipun tidak ada statistik akurat untuk menunjukkan seberapa sering jQuery digunakan, pandangan sepintas ke situs populer menunjukkan bahwa jQuery ada di seluruh web. jQuery juga membuat pengembangan lintas-browser lebih mudah. Meskipun kita belum banyak melihatnya sejauh ini (terutama jika kita telah membaca buku ini dalam urutan linier), dukungan untuk JavaScript sangat berbeda dari browser ke browser dan dari versi ke versi. Apa yang berfungsi di Firefox mungkin tidak berfungsi sama sekali di Internet Explorer atau mungkin bekerja sebaliknya.

Contoh favorit tentang bagaimana dukungan JavaScript berbeda dari browser ke browser melibatkan penanganan tanggal. Ada fungsi JavaScript tertentu yang mengembalikan tahun. Misalnya, dengan asumsi itu tahun 2008 ketika kita memanggil fungsi, JavaScript seharusnya mengembalikan 2008 — tetapi itu tidak selalu terjadi, tergantung pada browser yang kita gunakan. Ketika fungsi itu digunakan di Firefox atau Safari, kita menerima tahun penuh 2008, seperti yang kita harapkan. Saat kita menggunakan JavaScript di Internet Explorer, kita menerima jumlah tahun yang telah berlalu sejak 1900. Saat tahun 2008, kita akan menerima 108 kembali dari Internet Explorer.

Jelas jika kita mencoba melakukan perhitungan tanggal apa pun dengan nilai itu, itu akan menjadi sangat miring. Browser mana yang benar? Itu tidak masalah. Yang penting adalah bahwa produsen browser membaca spesifikasi JavaScript secara berbeda dan pada akhirnya mengembalikan hal yang berbeda untuk fungsi yang sama. Sayangnya, contoh tanggal hanyalah salah satu dari banyak contoh (beberapa jauh lebih serius dari itu) di mana browser berbeda dalam cara mereka mengimplementasikan JavaScript. Kabar baiknya adalah bahwa jQuery menghilangkan komplikasi itu. Fungsi jQuery mencari tahu browser apa yang digunakan dengan cara yang akurat dan kemudian memperhitungkannya untuk membuat browser berperilaku secara konsisten.

8.2 MENGINSTAL JQUERY

Ada dua cara untuk menggunakan jQuery, baik diunduh secara lokal atau di Jaringan Pengiriman Konten (CDN). Salinan lokal hanya itu, file yang berada di dalam root dokumen kita di hard drive server Anda. Versi yang dihosting CDN berarti bahwa file jQuery berada di server orang lain dan kita hanya mereferensikannya dalam kode Anda. Apakah kita menggunakan salinan CDN lokal terserah Anda. Untuk situs web produksi, kami sangat menyarankan untuk menggunakan salinan lokal jQuery untuk kecepatan dan keandalan. Namun, dalam pengembangan, seperti saat kita mengikuti buku ini, boleh saja menggunakan jQuery versi CDN. Contoh buku ini menggunakan jQuery yang dihosting CDN, tetapi bagian ini menunjukkan cara menggunakan lokal dan CDN.

Menginstal jQuery secara lokal

jQuery tersedia sebagai unduhan dari www.jquery.com. Sesampai di sana, pilih versi Produksi dan klik Unduh. Bergantung pada pengaturan browser Anda, kita mungkin berakhir dengan halaman yang penuh dengan kode JavaScript. Jika demikian, pilih Save As dari menu File. Dalam kasus lain, kita hanya akan diminta untuk mengunduh file. Pada akhirnya, kita ingin mendapatkan file bernama seperti jquery-1.8.1.min.js, terlepas dari apakah kita menyimpan file atau mengunduhnya. File harus ditempatkan ke root dokumen Anda. Ingat nama filenya; kita akan membutuhkannya nanti. Itu saja untuk menginstal jQuery — unduh dan masukkan file ke root dokumen Anda.

8.3 MENGGUNAKAN JQUERY YANG DIHOSTING CDN

Opsi yang dihosting CDN untuk jQuery sangat bagus untuk pengembangan. kita tidak perlu khawatir mengunduh file atau meletakkannya di tempat yang tepat; itu selalu tersedia (selama CDN habis). Versi CDN-host tersedia dari banyak pemain besar di web, seperti Google dan Microsoft. kita tidak perlu mengunduh apa pun untuk menggunakan jQuery yang dihosting CDN, jadi bagian ini singkat dan manis. kita dapat menemukan tautan untuk versi yang dihosting CDN di www.jquery.com/download. Bagian selanjutnya menunjukkan cara menambahkan jQuery yang dihosting CDN ke halaman Anda.

Menambahkan jQuery ke Halaman

Sekarang setelah kita mengunduh jQuery atau tahu di mana menemukan versi yang di-host CDN, kita perlu merujuknya di halaman Anda. jQuery sama seperti file JavaScript eksternal. Bagian ini menunjukkan cara menambahkan jQuery ke halaman kita baik untuk jQuery yang dihosting secara lokal maupun jQuery yang dihosting CDN.

Menambahkan jQuery lokal ke halaman

Di bagian sebelumnya, kami menginstruksikan kita untuk mengunduh jQuery dan menempatkannya di root dokumen server web. Jika kita tidak ingat nama filenya, cari di root dokumen Anda. Ini akan dinamai seperti jquery-1.8.1.min.js. (Perhatikan bahwa nomor versi hampir pasti akan berbeda saat kita membaca ini.) Menambahkan jQuery ke halaman berarti menambahkan referensi skrip eksternal, seperti ini:

```
<script type=”text/javascript” src=”jquery-1.8.1-min.js”></
script>
```

Referensi itu biasanya ditambahkan di bagian <head> halaman. Daftar kode html selanjutnya menunjukkan halaman dengan skrip jQuery yang direferensikan di bagian <head>.

Contoh 1 Daftar Menambahkan jQuery ke Halaman

```
<!doctype html>
<html>
<head>
<title>jQuery</title>
<script type=”text/javascript”
 src=”jquery-1.8.1.min.js”></script>
</head>
<body>
<h1>Adding jQuery</h1>
</body>
</html>
```

Menambahkan CDN jQuery ke halaman

Memuat jQuery yang dihosting CDN sama seperti memuatnya secara lokal, tanpa bagian di mana kita menyimpan jQuery di hard drive Anda, tentu saja. Selain detail itu, kita cukup menambahkan jQuery seperti file JavaScript eksternal lainnya. Berikut ini contohnya:

```
<script type=”text/javascript”
src=” http://ajax.aspnetcdn.com/ajax/jQuery/jquery-1.8.1.min.
js”>
</script>
```

Tetapi bagaimana kita mengetahui lokasi rahasia di mana jQuery di-host untuk penggunaan umum? Buka http://jquery.com/download dan kita dapat menemukan jQuery CDNhosted. Di dalam halaman Unduh, kita melihat bagian untuk jQuery yang dihosting CDN. Saat kita menemukan yang ingin kita gunakan, klik kanan dan pilih opsi Salin Lokasi Tautan atau serupa dari menu konteks di browser Anda. Itu akan menyalin URL ke clipboard kita untuk digunakan nanti. Contoh halaman penuh dengan jQuery yang dihosting CDN terlihat sangat mirip dengan halaman untuk salinan yang dihosting secara lokal, hanya atribut src yang berubah. Daftar 3-2 menunjukkan HTML dan JavaScript; perhatikan secara khusus tag <script> di bagian <head>.

Contoh 2 Daftar jQuery yang dihosting CDN

```
<!doctype html>
<html>
<head>
<title>jQuery</title>
<script type=”text/javascript”
 src=”http://ajax.aspnetcdn.com/ajax/jQuery/jquery1.8.1.min.js”>
</script>
</head>
<body>
<h1>Adding jQuery</h1>
</body>
</html>
```

8.4 MENGGABUNGKAN FUNGSI JQUERY READY()

Masalah umum saat memprogram JavaScript adalah program JavaScript akan berjalan sebelum halaman dimuat. Bab sebelumnya menjelaskan bahwa kita dapat mengakses elemen HTML pada halaman. Ini berarti kita juga dapat mengakses hal-hal seperti gambar, formulir, dan apa pun yang kita inginkan, di halaman web. Masalahnya muncul ketika kita mencoba mengakses sesuatu di halaman sebelum browser memuatnya. jQuery menawarkan cara untuk mengatasinya, yang kita lihat di bagian ini.

jQuery memiliki fungsi yang disebut ready() yang menunggu halaman siap. Tidak semuanya tersedia (misalnya, beberapa gambar mungkin masih dimuat), tetapi elemen HTML siap untuk digunakan saat fungsi ready() dipanggil.

Saat kita memprogram dengan jQuery, biasanya menempatkan kode kita di dalam fungsi ready() sehingga kita dapat memastikan bahwa semua hal di halaman siap untuk kita gunakan dalam program Anda. Sungguh, tidak ada banyak untuk ini, jadi cobalah untuk tidak terlalu memikirkannya. Sebuah contoh akan membantu menggambarkan! Contoh Daftar 3 menunjukkan halaman HTML dengan JavaScript di dalam fungsi jQuery ready().

Contoh Daftar Menggunakan Fungsi jQuery ready()

```
<!doctype html>
<html>
<head>
<title>jQuery</title>
<script type=”text/javascript”
 src=”http://ajax.aspnetcdn.com/ajax/jQuery/jquery1.8.1.min.js”>
</script>
</head>
<body>
<h1>Adding jQuery</h1>
<script type=”text/javascript”>
$(document).ready(function() {
 alert(“hi”);
});
</script>
</body>
</html>
```

Jika dilihat melalui browser, hasilnya adalah peringatan seperti gambar berikut ini:

Gambar 8.1 Peringatan yang dihasilkan oleh fungsi jQuery ready().

Kode ini memiliki dua bidang minat. Bagian pertama adalah elemen <script> itu sendiri:

```
<script type=”text/javascript”
 src=”http://ajax.aspnetcdn.com/ajax/jQuery/jquery1.8.1.min.js”>
</script>
```

Ini termasuk jQuery dari CDN Microsoft ke dalam halaman. Area minat berikutnya ada di dalam <body>, khususnya, <script> di dalam body:

```
<script type=”text/javascript”>
$(document).ready(function() {
 alert(“hi”);
});
</script>
```

Kode ini memanggil fungsi jQuery ready(), bagian dari objek dokumen. Perhatikan sintaks khusus dengan tanda dolar dan tanda kurung. Inilah yang memberi tahu browser dan JavaScript bahwa yang berikut akan menjadi jQuery, jadi pemrosesan diserahkan ke jQuery. Dan karena jQuery memiliki fungsi yang disebut ready(), ia tahu apa yang harus dilakukan. kita menggunakan $() di semua tempat dengan jQuery; itulah yang memberi tahu jQuery bahwa ia harus memperhatikan. Di dalam fungsi jQuery ready() ada kode ini:

```
function() {
 alert(“hi”);
}
```

Anda sudah tahu semua tentang fungsi jadi ini bukan sesuatu yang baru. Atau itu? Jika ini adalah fungsi, di mana nama fungsinya? Untuk sebagian besar penggunaan jQuery, kita akan melihat sintaks yang mirip dengan apa yang kita lihat di sini, dengan fungsi tanpa nama seperti ini dan kemudian kode di dalamnya. Saat kita melihat sintaks ini, function(), tanpa nama, itu disebut fungsi anonim. Untuk sebagian besar, kita tidak perlu tahu banyak tentang fungsi anonim sampai kita masuk lebih dalam ke pemrograman JavaScript. Untuk apa yang kita lakukan di sini, ketahuilah bahwa ini adalah sintaks khas yang kita gunakan saat menggunakan jQuery. Di dalam fungsi, peringatan ditampilkan. Tidak mengherankan di sini — ini adalah fungsi alert() standar yang telah kita gunakan di sepanjang buku ini. Tapi apa yang terjadi di sini penting: kita menggunakan jQuery bersama dengan JavaScript di dalam skrip yang sama.

8.5 MEMILIH ELEMEN DENGAN JQUERY

Bagian sebelumnya menjelaskan cara memilih objek dokumen. Ini juga menyediakan banyak cara kerja jQuery. Saat kita menggunakan kode $(document), kita menggunakan sesuatu yang disebut selector. Sebagian besar dari apa yang akan kita lakukan di jQuery terjadi melalui penyeleksi. Misalnya, kita akan sering memilih bagian halaman web dan kemudian meminta jQuery melakukan tindakan pada bagian halaman itu. Tindakan itu bisa apa saja dari menambahkan teks, mengubah HTML, mengubah CSS atau, yah, apa saja yang dapat kita pikirkan! Alur dasar untuk pemrograman JavaScript dengan jQuery adalah ini:

1. Pilih elemen pada halaman web (atau seluruh halaman itu sendiri).
2. Lakukan sesuatu yang menarik dengan elemen itu

Oke, apa yang kita lakukan dengan elemen tidak harus menarik, tetapi kita akan melakukan sesuatu dengan elemen yang dipilih. Sesuatu itu bisa berupa apa saja mulai dari menghapus elemen hingga menambahkan atau mengubahnya atau sekadar mendapatkan informasi dari elemen, seperti teksnya atau gaya CSS saat ini.

Selector jQuery dari dekat

Ada tiga selector utama atau dasar di jQuery. Kami menyebutnya primer atau dasar karena merekalah yang paling sering kita gunakan. kita dapat mengatur penyeleksi yang sangat kompleks berdasarkan struktur halaman, tetapi paling sering kita akan menggunakan salah satu dari tiga penyeleksi ini:

- Berdasarkan kelas
- Dengan ID
- Berdasarkan elemen

Jika kita memiliki beberapa HTML yang terlihat seperti ini:

My Book

Anda dapat mengaksesnya dengan jQuery seperti ini:

$(“#bookTitle”)

Penting untuk dicatat bahwa hal-hal di jQuery (dan JavaScript) peka terhadap huruf besar-kecil. Jadi booktitle tidak sama dengan BOOKTITLE dan tidak sama dengan bookTitle. Tidak masalah case apa yang kita gunakan, asalkan cocok antara HTML dan JavaScript dan jQuery. Sekarang lihat sedikit HTML ini:

This is some text

Selector jQuery terlihat seperti ini:

$(“.titleClass”)

Jika kita berpikir bahwa penyeleksi ini terlihat seperti rekan CSS mereka, kita benar. Jangan khawatir jika kita tidak memikirkannya; tidak akan ada kuis. Di CSS, kita mengakses item dengan ID mereka dengan tanda pound (#) dan kita mengakses kelas dengan satu titik (.):

#bookTitle
.titleClass

Semua yang kita lakukan untuk jQuery adalah membungkusnya dalam konstruksi $() dan menggunakan beberapa tanda kutip juga. Jadi kita mendapatkan ini:

$(“#bookTitle”)
$(“.titleClass

Selektor lain yang sering digunakan mengambil semua elemen dari tipe tertentu. Selector berikut memilih semua elemen di seluruh halaman:

$(“div”)

Ada selector yang lebih maju. Misalnya, kita dapat memilih berdasarkan posisi elemen pada halaman dan, yah, hampir semua kombinasi yang dapat kita pikirkan. Tetapi kita akan menggunakan ketiganya paling sering dan di mana kita membutuhkan lebih banyak, kami akan menunjukkannya kepada Anda.

Filtering

Satu hal tambahan yang harus kita ketahui tentang penyeleksi jQuery adalah kita dapat memfilternya. Ini sangat berguna ketika bekerja dengan formulir dan event.

8.6 BEKERJA DENGAN HTML MENGGUNAKAN JQUERY

Anda dapat menggunakan jQuery untuk melakukan segala macam hal menyenangkan dengan HTML pada halaman dan kami memberikan petunjuk tentang beberapa hal tersebut, seperti menambahkan HTML ke halaman atau mengubah teks, dan sebagainya. Saatnya untuk belajar bagaimana melakukannya!

Menambahkan HTML ke halaman

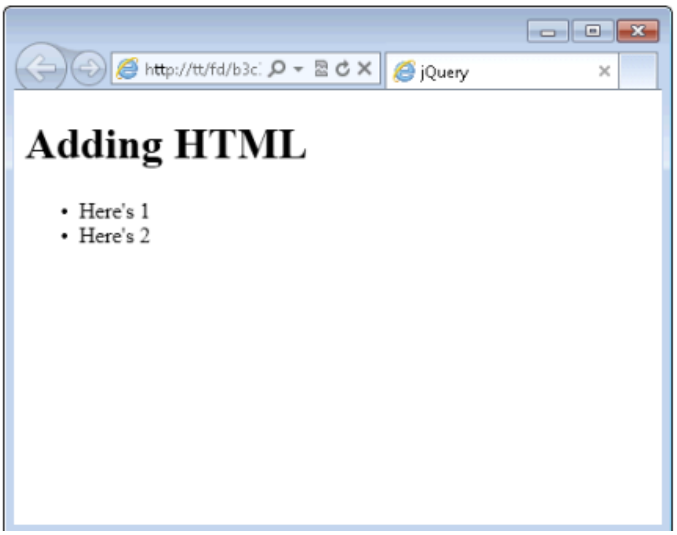
jQuery dapat digunakan untuk menambahkan HTML ke halaman. kita dapat menambahkan segala macam HTML, gambar, apa saja, dan sepenuhnya mengubah layout halaman menggunakan jQuery. Namun, melakukan hal itu bukanlah ide yang bagus, karena bisa sangat, sangat membingungkan untuk mencari tahu dari mana datangnya dan juga bisa lebih sulit untuk dipertahankan di masa mendatang ketika kita perlu mengubah halaman.

Bagaimanapun, menambahkan HTML untuk hal-hal seperti pesan kesalahan atau untuk menambahkan data ke halaman cukup umum. Pikirkan tentang situs perjalanan yang mencari informasi penerbangan. kita mengklik tombol dan itu membangun hasil secara dinamis, tepat di halaman yang sama. Situs-situs tersebut menggunakan JavaScript, berkali-kali jQuery, untuk mencapai prestasi ini. Tetapi sebelum kita mengubah HTML, kita harus mempelajari cara menambahkan HTML ke halaman.

Contoh 4 Daftar HTML dengan Daftar

```
<!doctype html>
<html>
<head>
<title>jQuery</title>
</head>
<body>
<h1>Adding HTML</h1>
<ul id=”theList”>
 <li>Here’s 1</li>
 <li>Here’s 2</li>
</ul>
</body>
</html>
```

Halaman yang dilihat di browser web terlihat seperti pada gambar berikut ini:



Gambar 8.2 Halaman sederhana dengan daftar.

Halaman ini menggunakan daftar tidak berurutan dengan dua item. kita dapat menambahkan item lain ke daftar itu dengan fungsi jQuery append(). Melakukannya berarti memilih elemen <ul>, yang sudah kita ketahui caranya, lalu memanggil fungsi append(). Berikut ini contoh untuk menambahkan item ketiga ke daftar:

$(“#theList”).append(“
Here’s 3”);

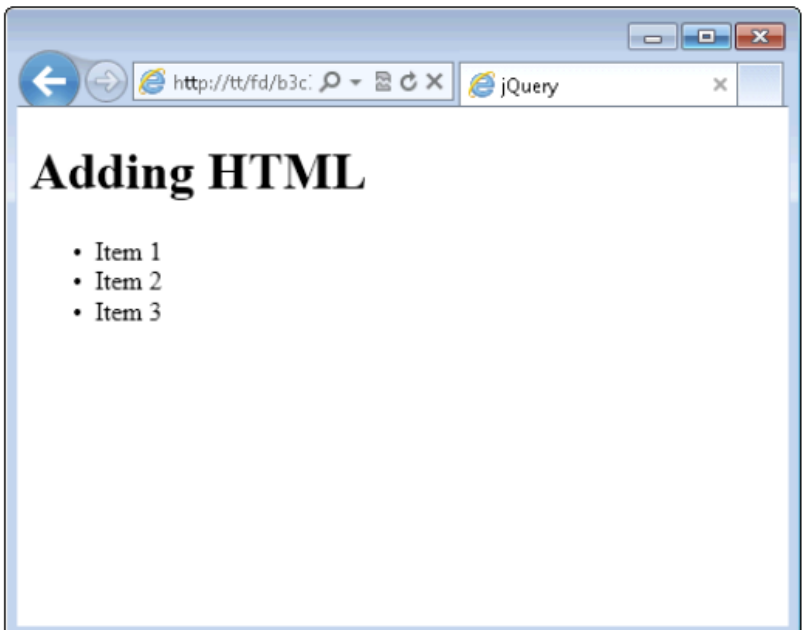
Seperti yang kita lihat, kita memilih elemen <ul> menggunakan selector ID dan kemudian memanggil fungsi append() dengan HTML untuk ditambahkan. Tidak jauh lebih sederhana dari itu. Daftar kode selanjutnya menunjukkan kode akhir. Perhatikan bahwa jQuery telah ditambahkan ke dalamnya di bagian <head> dan fungsi append() berada di dalam fungsi ready(), seperti yang dibahas sebelumnya.

Contoh 5 Daftar Menambahkan Item dengan jQuery

```
<!doctype html>
<html>
<head>
<title>jQuery</title>
<script type=”text/javascript”
 src=”http://ajax.aspnetcdn.com/ajax/jQuery/jquery1.8.1.min.js”>
</script>
</head>
<body>
<h1>Adding HTML</h1>
<ul id=”theList”>
 <li>Here’s 1</li>
 <li>Here’s 2</li>
</ul>
<script type=”text/javascript”>
$(document).ready(function() {
 $(“#theList”).append(“<li>Here’s 3</li>”);
```

```
});
</script>
</body>
</html>
```

Jika dilihat di browser, hasilnya seperti gambar berikut ini:



Gambar 8.3 Menambahkan HTML dengan fungsi append() jQuery

8.7 MENGUBAH ATRIBUT DAN GAYA

jQuery membuat pengambilan dan pengaturan atribut HTML dan gaya CSS menjadi mudah. Ini berarti kita dapat mengubah hal-hal seperti sumber gambar atau kelas CSS atau bahkan gaya CSS itu sendiri. Bagian ini melihat bagaimana melakukan hal itu.

Membaca atribut

Ingat dari cara, jauh sebelumnya dalam buku ini. kita belajar bahwa hal-hal deskriptif yang terkandung di dalam elemen HTML disebut atribut. Sebagai contoh:

```
<a id=”exLink” class=”link” href=”http://www.example.com”>Website</a>
```

Bagian id, class, dan href yang kita lihat di elemen <a> itu semuanya adalah atribut. Menggunakan jQuery, kita dapat mengetahui nilai untuk semua atribut tersebut, dan seperti yang kita lihat nanti, kita juga dapat mengaturnya. Membaca atribut dengan jQuery berarti menggunakan fungsi attr(). Daftar 3-7 menunjukkan kode menggunakan attr untuk membaca atribut href dari tautan yang baru saja kita lihat.

Contoh 6 Menggunakan Fungsi attr()

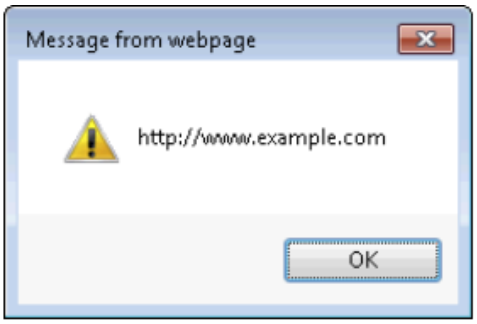
```
<!doctype html>
<html>
<head>
<title>jQuery</title>
<script type=”text/javascript”
 src=”http://ajax.aspnetcdn.com/ajax/jQuery/jquery1.8.1.min.js”>
```

```
</script>
</head>
<body>
<h1>Attributes</h1>
<a id=”exLink” class=”link” href=”http://www.example.com”>Web
site</a>
<script type=”text/javascript”>
$(document).ready(function() {
 alert($(“#exLink”).attr(“href”));
});
</script>
</body>
</html>
```

Sebagian besar pekerjaan dilakukan pada satu baris:

```
alert($(“#exLink”).attr(“href”));
```

Baris itu menggunakan selector untuk memilih elemen dengan ID exLink dan kemudian memanggil fungsi attr() dengan "href" sebagai argumennya. Hasilnya dikembalikan dan ditempatkan di alert(), yang ditunjukkan pada gambar berikut ini:



Gambar 8.4 Mengakses atribut href.

Menulis atribut

Sama seperti fungsi text() dan html(), kita juga dapat mengatur nilai atribut menggunakan fungsi attr(). Misalnya, untuk mengubah nilai atribut href dari kode di sebelumny, dan lakukan ini:

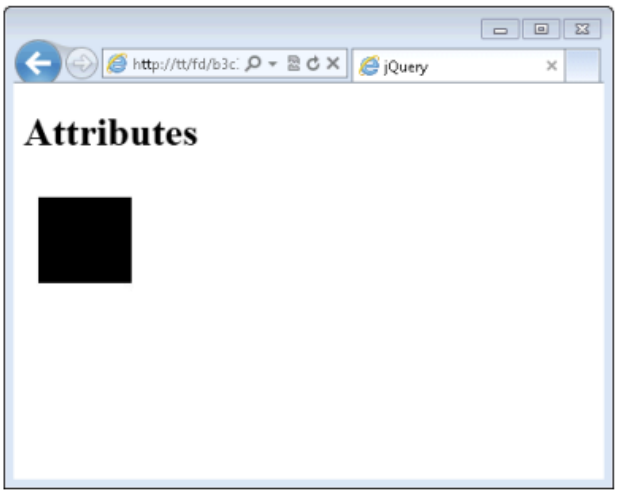
```
$(“#exLink”).attr(“href”, “http://www.braingia.org”);
```

Gambar ditambahkan ke halaman dengan menggunakan atribut src. Ini berarti kita dapat mengubah atribut src untuk mengubah gambar, dengan cepat, melalui JavaScript. Daftar selanjutnya berisi HTML untuk sebuah halaman. HTML berisi gambar yang memuat square.jpg.

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

```
<title>jQuery</title>
</head>
<body>
<h1>Attributes</h1>
<img alt=”square” id=”theImage” src=”square.jpg” />
</body>
</html>
```

Saat dilihat di browser, halaman terlihat seperti Gambar dibawah ini:



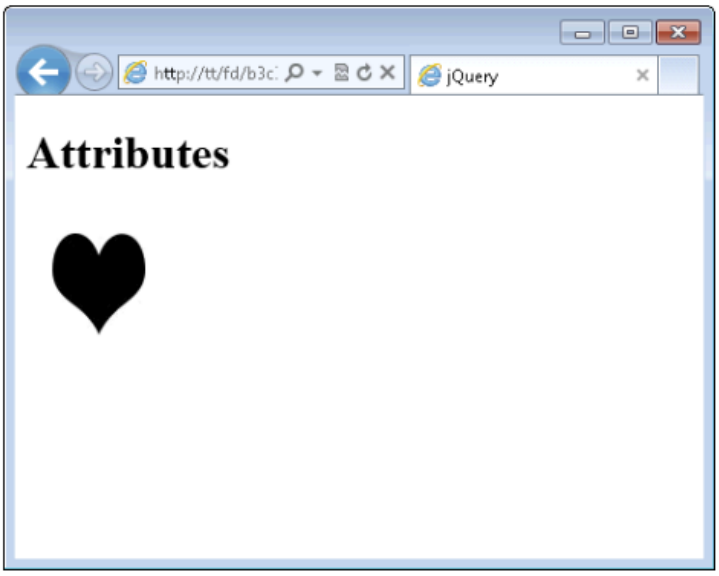
Gambar 8.5 Halaman dengan gambar persegi.

Kita dapat mengubah gambar diatas menjadi gambar lain menggunakan fungsi attr().

Contoh 7 Daftar Mengubah Sumber Gambar

```
<!doctype html>
<html>
<head>
<title>jQuery</title>
<script type=”text/javascript”
 src=”http://ajax.aspnetcdn.com/ajax/jQuery/jquery1.8.1.min.js”>
</script>
</head>
<body>
<h1>Attributes</h1>
<img alt=”square” id=”theImage” src=”square.jpg” />
<script type=”text/javascript”>
$(document).ready(function() {
 $(“#theImage”).attr(“src”,”heart.jpg”);
});
</script>
</body>
</html>
```

Gambar dibawah ini menunjukkan hasil dari koding diatas.



Gambar 8.6 Mengubah gambar melalui jQuery.

Perhatikan baik-baik HTML dan kita akan melihat masalah. kita telah berhasil mengubah atribut src, tetapi atribut alt masih mengatakan bahwa gambar adalah persegi. kita harus mengubah atribut alt agar sesuai dengan gambar. Melakukannya semudah memanggil attr() lagi, kali ini untuk menyetel atribut alt.

$(“#theImage”).attr(“alt”,”heart”);

Karena kita tidak pernah melihat perubahannya, kode pada daftar sebelumnya mungkin tidak terlalu menarik. Lakukan sesuatu untuk mengubah itu. Tambahkan pengatur waktu untuk menunda sakelar. Untuk timer ini, gunakan fungsi JavaScript asli yang disebut setTimeout. Fungsi setTimeout() membutuhkan dua argumen, fungsi untuk memanggil saat timer berakhir dan berapa lama menunggu. Nilai waktu yang kita gunakan dalam milidetik, jadi 2 detik adalah 2000 milidetik. Daftar dibawah ini menunjukkan kode baru.

Contoh 8 Daftar Menunda Perubahan Gambar

```
<!doctype html>
<html>
<head>
<title>jQuery</title>
<script type=”text/javascript”
 src=”http://ajax.aspnetcdn.com/ajax/jQuery/jquery1.8.1.min.js”>
</script>
</head>
<body>
<h1>Attributes</h1>
<img alt=”square” id=”theImage” src=”square.jpg” />
<script type=”text/javascript”>
function changeImage() {
 $(“#theImage”).attr(“src”,”heart.jpg”);
 $(“#theImage”).attr(“alt”,”heart”);
```

```
}
$(document).ready(function() {
 setTimeout(changeImage,2000);
});
</script>
</body>
</html>
```

Kode ini membangun fungsi yang disebut changeImage(). Di dalam fungsi itu ada baris jQuery yang sama dengan yang kita miliki pada contoh sebelumnya. Di dalam fungsi ready(), sekarang ada panggilan ke setTimeout dengan dua argumen fungsi yang telah kami sebutkan, fungsi changeImage, dan 2000, untuk penundaan 2 detik.

Mengubah CSS

Anda juga dapat mengubah informasi gaya pada halaman, baik dengan menyetel gaya secara langsung atau dengan mengubah kelas CSS yang diterapkan ke elemen. Kelas hanyalah atribut lain pada suatu elemen, jadi mengubah CSS berarti menggunakan fungsi attr() lagi.

Menambahkan kelas

Daftar berikut berisi HTML dasar dengan beberapa informasi gaya di bagian <head>.

Contoh 9 Daftar HTML dengan CSS

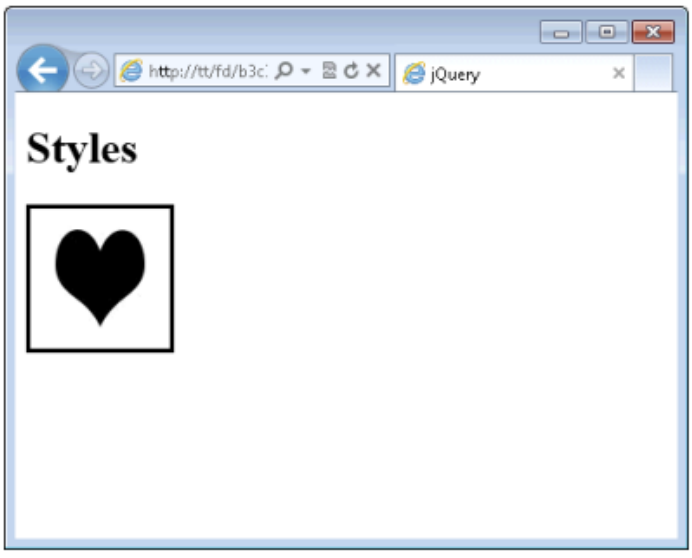
```
<!doctype html>
<html>
<head>
<title>jQuery</title>
<script type=”text/javascript”
 src=”http://ajax.aspnetcdn.com/ajax/jQuery/jquery1.8.1.min.js”>
</script>
<style type=”text/css”>
.borderClass {
 border: 3px solid black;
}
</style>
</head>
<body>
<h1>Styles</h1>
<img alt=”heart” id=”theImage” src=”heart.jpg” />
</body>
</html>
```

Daftar Menambahkan Kelas Menggunakan attr()

```
<!doctype html>
<html>
<head>
<title>jQuery</title>
<script type=”text/javascript”
 src=”http://ajax.aspnetcdn.com/ajax/jQuery/jquery1.8.1.min.js”>
```

```
</script>
<style type=”text/css”>
.borderClass {
 border: 3px solid black;
}
</style>
</head>
<body>
<h1>Styles</h1>
<img alt=”heart” id=”theImage” src=”heart.jpg” />
<script type=”text/javascript”>
$(document).ready(function() {
 $(“#theImage”).attr(“class”,”borderClass”);
});
</script>
</body>
</html>
```

Kode hanya memanggil attr() untuk mengubah atribut class menjadi borderClass. Dalam hal ini, sebenarnya belum ada atribut class pada elemen tersebut, jadi jQuery cukup pintar untuk menambahkan satu untuk Anda.



Gambar 8.7 Menambahkan kelas melalui fungsi attr().

Tetapi apa yang harus dilakukan jika sudah ada kelas pada elemen tersebut? kita dapat mengambil kelas terlebih dahulu menggunakan attr dan kemudian menambahkan yang lain. Atau kita bisa menggunakan fungsi jQuery untuk menambahkan kelas, yang disebut addClass(). Fungsi addClass() tidak mengganggu kelas lain yang sudah diterapkan ke elemen; itu hanya menambahkan kelas lain untuk itu.

```
$(“#theImage”).attr(“class”,”borderClass”);
```

ke:

$(“#theImage”).addClass(“borderClass”);

Dengan perubahan sederhana itu, class borderClass akan ditambahkan dan kita tidak perlu khawatir untuk menghapus kelas lain yang diterapkan ke elemen tersebut.

Menghapus kelas

Pendamping untuk fungsi addClass(), yang disebut removeClass, mengambil kelas dari sebuah elemen. Seperti addClass(), removeClass() tidak memengaruhi kelas lain yang ada di elemen; itu hanya menghapus kelas yang ditentukan. Sintaks untuk removeClass seperti sintaks addClass:

$(“#theImage”).removeClass(“borderClass”);

Di bab berikutnya, kita akan melihat fungsi terkait lainnya yang disebut toggleClass yang menambahkan atau menghapus kelas, bergantung pada apakah itu sudah diterapkan ke elemen.

BAB 9
REAKSI EVENT DENGAN JAVASCRIPT DAN JQUERY

Event (peristiwa) adalah hal-hal yang terjadi. Misalnya, matahari terbit adalah suatu peristiwa. Matahari terbenam adalah sebuah peristiwa. kita dapat memilih untuk bereaksi terhadap peristiwa tersebut. Misalnya, saat matahari terbit, kita mungkin bangun dari tempat tidur — atau mungkin tidak. Saat matahari terbenam, kita mungkin menyalakan lampu atau mungkin pergi tidur. Ketika datang ke pemrograman web, peristiwa adalah hal-hal yang terjadi di halaman web. Misalnya, pengguna mungkin menggerakkan mouse di atas tombol, mengklik tombol, atau mengirimkan formulir. Seperti contoh matahari terbit, kita dapat memilih untuk bereaksi terhadap peristiwa tersebut atau kita dapat mengabaikannya. Jika semua yang ingin kita lakukan adalah mengabaikan peristiwa, maka ini akan menjadi bab yang sangat, sangat singkat. Tetapi kita mungkin ingin bereaksi terhadap peristiwa, dan kami menunjukkan cara melakukannya untuk beberapa skenario umum.

9.1 MEMAHAMI EVENT

Secara umum, ada empat jenis acara yang kita khawatirkan:

Persitiwa formulir: Mencakup hal-hal seperti memilih kotak centang atau mengirimkan formulir itu sendiri. Bereaksi untuk membentuk acara adalah salah satu hal paling umum yang dilakukan oleh programmer JavaScript.

Peristiwa mouse: Ini bisa berupa apa saja, mulai dari klik mouse hingga gerakan mouse di halaman. Betul sekali; kita benar-benar dapat melacak di mana mouse berada pada halaman dan bereaksi terhadapnya.

Peristiwa keyboard: Hal-hal yang terjadi melalui keyboard, seperti penekanan tombol, dianggap peristiwa keyboard.

Peristiwa halaman: Hal-hal seperti pemuatan atau pembongkaran halaman dianggap sebagai peristiwa halaman. kita akan senang mengetahui bahwa kita telah bereaksi terhadap peristiwa pemuatan halaman melalui fungsi jQuery ready().

Berbicara tentang jQuery, bab ini sebagian besar berkonsentrasi pada penggunaan jQuery untuk bekerja dengan acara. Menggunakan jQuery menghemat banyak masalah kompatibilitas yang muncul saat kita mulai mencoba membuat kode JavaScript kita berfungsi di seluruh browser.

9.2 BEKERJA DENGAN FORMULIR

Bab-bab sebelumnya dari buku ini menunjukkan formulir yang sedang dibuat dan gaya bentuk itu dengan CSS. Sekarang saatnya mempelajari cara bekerja dengan formulir itu menggunakan JavaScript. Cara yang sering digunakan JavaScript dengan formulir adalah dengan memberikan validasi formulir saat pengguna mengisinya atau sebelum dikirimkan ke server.

Menambahkan Handler Kirim

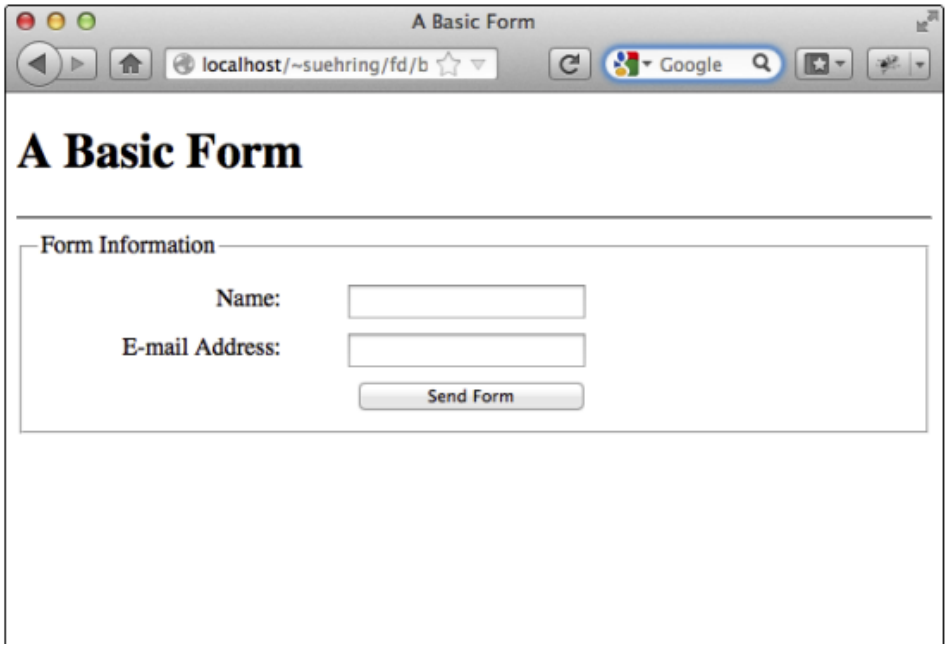
jQuery menyertakan fungsi yang secara otomatis mengawasi formulir yang akan dikirimkan. Contoh 1 menunjukkan HTML untuk formulir yang digunakan sebelumnya dalam buku ini. Kami telah mengambil kebebasan untuk menambahkan jQuery ke bagian <head> formulir.

Contoh 1 Daftar Bentuk Sederhana

```
<!doctype html>
<html>
<head>
<title>A Basic Form</title>
<script type=”text/javascript”
src=”http://ajax.aspnetcdn.com/ajax/jQuery/jquery1.8.1.min.js”>
</script>
<style type=”text/css”>
.form-field {
 clear: both;
 padding: 10px;
 width: 350px;
}
.form-field label {
 float: left;
 width: 150px;
 text-align: right;
}
.form-field input {
 float: right;
 width: 150px;
 text-align: left;
}
#submit {
text-align: center;
}
</style>
</head>
<body>
<h1>A Basic Form</h1>
<hr>
<form action=”#”>
<fieldset>
 <legend>Form Information</legend>
 <div class=”form-field”>
 <label for=”username”>Name:</label>
 <input type=”text” id=”username”
name=”username”>
 </div>
 <div class=”form-field”>
 <label for=”email”>E-mail Address:</label>
 <input type=”text” id=”username”
name=”email”>
 </div>
 <div class=”form-field”>
```

```
 <input id=”submit” type=”submit”
name=”submit” value=”Send Form”>
 </div>
</fieldset>
</form>
</body>
</html>
```

Halaman yang dilihat di browser terlihat seperti gambar berikut ini:



Gambar 9.1 Formulir web dasar.

Saat ini, ketika kita mengklik Kirim Formulir, tidak ada yang terjadi. Ubah itu dengan mengikuti langkah-langkah ini.

1. Buka text editor.
2. Di editor, tempatkan kode berikut:

```
<!doctype html>
<html>
<head>
<title>A Basic Form</title>
<script type=”text/javascript”
 src=”http://ajax.aspnetcdn.com/ajax/jQuery/jquery-1.8.1.min.js”>
</script>
<style type=”text/css”>
.form-field {
 clear: both;
 padding: 10px;
 width: 350px;
}
.form-field label {
```

```
 float: left;
 width: 150px;
 text-align: right;
}
.form-field input {
 float: right;
 width: 150px;
 text-align: left;
}
#submit {
 text-align: center;
}
</style>
</head>
<body>
<h1>A Basic Form</h1>
<hr>
<form action="#">
<fieldset>
 <legend>Form Information</legend>
 <div class="form-field">
 <label for="username">Name:</label>
 <input type="text" id="username"
name="username">
 </div>
 <div class="form-field">
 <label for="email">E-mail Address:</
label>
 <input type="text" id="username"
name="email">
 </div>
<div class="form-field">
 <input id="submit" type="submit"
name="submit" value="Send Form">
 </div>
</fieldset>
</form>
</body>
</html>
```

3. Simpan formulir sebagai form1.html di root dokumen Anda.
4. Lihat halaman di browser web di http://localhost/form1.html.
5. Sekarang tambahkan kode berikut, tepat setelah tag penutup </form> dan sebelum tag </body> penutup.

```
<script type="text/javascript">
```

```
$(document).ready(function() {
 $("form").submit(function() {
 alert("You submitted the form");
 return false;
 });
});
</script>
```

6. Simpan file, lagi sebagai form1.html.
7. Lihat halaman di browser; kita juga dapat memuat ulang halaman dengan Ctrl+R atau Command+R jika kita membiarkannya terbuka dari langkah sebelumnya.
8. Klik Kirim Formulir.
   Anda akan menerima peringatan seperti yang ditunjukkan pada Gambar 9.2.

Gambar 9.2 Peringatan ini mengonfirmasikan bahwa input telah dikirimkan.

```
$(document).ready(function() {
 $("form").submit(function() {
 alert("You submitted the form");
 return false;
 });
});
```

Anda telah menambahkan pengendali acara kirim. Lihat kodenya dimulai dengan fungsi ready(), yang telah kita lihat sebelumnya. Selanjutnya, kita memilih formulir dengan memilih semua elemen <form> pada halaman. Jika ada lebih dari satu formulir, kita mungkin ingin memberi nama atau ID formulir tersebut sehingga kita dapat memilih yang tepat, tetapi untuk contoh ini, cukup memilih berdasarkan elemen berfungsi. Selanjutnya, fungsi submit() dipanggil dan fungsi lain dibuat di dalamnya. Tugas utama fungsi ini adalah menampilkan peringatan, yang kita lihat. Baris kedua dalam fungsi, return false, menarik untuk form. Saat kita menggunakan return false dalam acara pengiriman formulir, kita pada dasarnya mencegah pengiriman formulir ke server. Oleh karena itu, kita hanya ingin melakukan ini untuk alasan tertentu, seperti saat formulir tidak valid seperti saat pengguna belum mengisi semua bidang yang diperlukan. Saat kita menambahkan return false, kita mencegah tindakan default. Karena tindakan default formulir adalah mengirimkan ke server, menambahkan return false mencegah terjadinya tindakan default tersebut. Cara lain untuk mencegah tindakan default adalah dengan fungsi jQuery preventDefault(). Kita menggunakan preventDefault dalam keadaan tertentu di mana return false tidak melakukan apa yang kita

inginkan. Mengubah JavaScript dari contoh sebelumnya untuk menggunakan preventDefault dan return false akan terlihat seperti ini:

```
$(document).ready(function() {
 $(“form”).submit(function(event) {
 alert(“You submitted the form”);
 event.preventDefault();
 return false;
 });
});
```

Memeriksa bidang kosong

Meskipun kami menyertakan seluruh bab tentang validasi di sini kami memberikan sekilas tentang validasi formulir, setidaknya memeriksa bidang kosong. Perhatikan bentuk dari bab sebelumnya dan ditunjukkan pada Gambar 9.1. Katakan bahwa bidang Nama diperlukan; itu berarti pengguna perlu mengisi sesuatu agar formulir dikirim ke server. kita dapat mengubah JavaScript untuk memastikan bahwa bidang telah diisi. Ingat bahwa ID bidang Nama adalah "nama pengguna". jQuery dapat memilihnya dengan cukup mudah, dan kemudian masalah mendapatkan nilai untuk bidang tersebut. Untuk itu, kita dapat menggunakan fungsi val() jQuery. Terakhir, yang perlu kita lakukan adalah memasukkan semuanya ke dalam pernyataan if untuk memastikan nilainya tidak kosong. Kodenya akan terlihat seperti ini:

```
$(document).ready(function() {
 $(“form”).submit(function(event) {
 if ($(“#username”).val() == “”) {
 alert(“Name can’t be blank”);
 event.preventDefault();
 return false;
 }
 });
});
```

Anda dapat mencoba ini dengan mengganti JavaScript dari contoh sebelumnya dengan kode itu. Jika kita mencoba mengirimkan formulir tanpa mengisi apa pun di bidang Nama, kita akan menerima peringatan seperti gambar dibawah ini:

Gambar 9.3 Validasi dasar pada formulir.

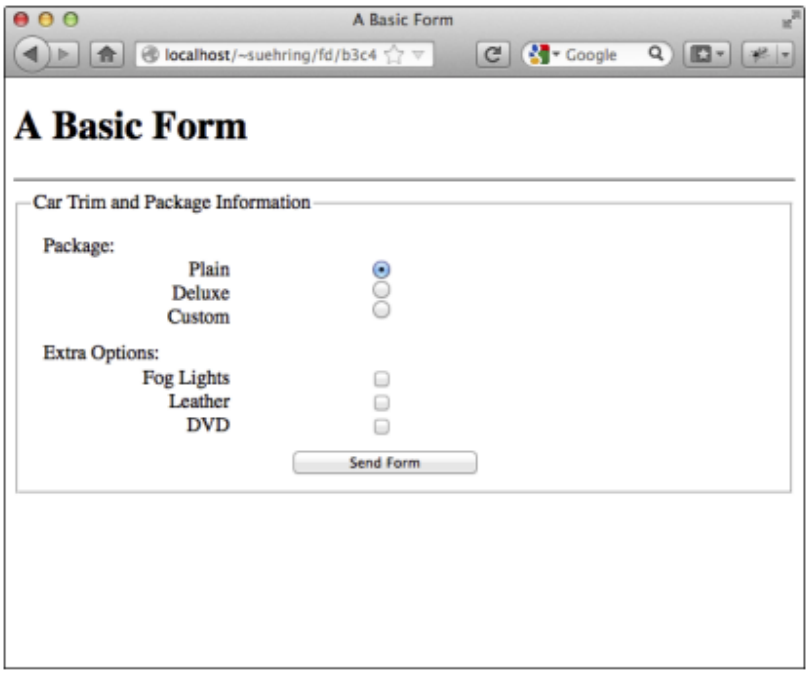
Validasi formulir ini sangat mendasar. Misalnya, kita bisa menempatkan satu ruang kosong di bidang itu dan itu akan valid.

9.3 MEMANTAU MOUSE EVENT

Anda dapat melihat dan bereaksi terhadap peristiwa mouse dengan JavaScript. Bagian ini membahas bagaimana melakukan keduanya.

Menangkap klik mouse

Hal yang umum dilakukan adalah merekam peristiwa klik mouse dengan JavaScript. Misalnya, ketika seseorang mengklik gambar atau mengklik elemen formulir, kita dapat bereaksi untuk memuat gambar yang berbeda atau memilih elemen formulir lainnya. Bayangkan kita telah menyiapkan toko mobil tempat orang dapat menyesuaikan mobil mereka dengan beberapa peningkatan. kita mengkhususkan diri dalam menambahkan lampu kabut, trim kulit, dan pemutar DVD. Orang-orang dapat mengunjungi situs web kita dan memilih level trim. Gambar dibawah ini menunjukkan halaman contoh tempat pengguna memilih opsi.



Gambar 9.4 Selectoran opsi trim

Jika orang memilih paket Deluxe, formulir harus mencentang semua kotak centang Opsi Ekstra. Jika orang memilih paket Biasa, semua opsi harus dihapus centangnya. Terakhir, jika orang memilih opsi individual dalam daftar Opsi Ekstra, maka paket Kustom harus dicentang. Ini semua dapat dicapai dengan beberapa baris JavaScript dan jQuery. Daftar 2 menunjukkan HTML, CSS, dan JavaScript untuk membuat perilaku yang diinginkan.

Contoh 2 Daftar membuat formulir custom

```
<!doctype html>
<html>
<head>
<title>A Basic Form</title>
<script type=”text/javascript”
 src=”http://ajax.aspnetcdn.com/ajax/jQuery/jquery1.8.1.min.js”>
</script>
<style type=”text/css”>
```

```
.form-field {
 clear: both;
 padding: 10px;
 width: 350px;
}
.form-field label {
 float: left;
 width: 150px
text-align: right;
}
.form-field input {
 float: right;
 width: 150px;
 text-align: left;
}
#submit {
 text-align: center;
}
</style>
</head>
<body>
<h1>A Basic Form</h1>
<hr>
<form action="#">
<fieldset>
 <legend>Car Trim and Package Information</legend>
 <div class="form-field">
 <div>Package: </div>
 <input id="plain" type="radio" name="trim"
 value="plain">
 <label for="plain">Plain</label>
 <input id="deluxe" type="radio" name="trim"
 value="deluxe">
 <label for="deluxe">Deluxe</label>
 <input id="custom" type="radio" name="trim"
 value="custom">
 <label for="custom">Custom</label>
 </div>
 <div class="form-field">
 <div>Extra Options:</div>
 <input type="checkbox" id="foglights"
name="option"
 value="foglights">
 <label for="foglights">Fog Lights</label>
 <input type="checkbox" id="leather" name="option"
 value="leather">
```

```
 <label for=”leather”>Leather</label>
 <input type=”checkbox” id=”dvd” name=”option”
 value=”dvd”>
 <label for=”dvd”>DVD</label>
 </div>
 <div class=”form-field”>
 <input id=”submit” type=”submit”
name=”submit” value=”Send Form”>
 </div>
</fieldset>
</form>
<script type=”text/javascript”>
$(document).ready(function() {
 $(“input[name=’trim’]”).click(function(event) {
 if ($(this).val() == “deluxe”) {
$(“input[name=’option’]”).attr(“checked”,true);
 } else if ($(this).val() == “plain”) {
 $(“input[name=’option’]”).attr(“checked”,false);
 }
 });
 $(“input[name=’option’]”).click(function(event) {
 $(“#custom”).attr(“checked”,”checked”);
 });
});
</script>
</body>
</html>
```

Perhatikan bahwa *Cascading Style Sheet* (CSS) daftar sebelumya. HTML sangat mudah, sejauh ia mengatur bentuk yang kita lihat pada Gambar 4.4. JavaScript adalah tempat hal-hal menjadi menarik. Hal pertama yang dilakukan JavaScript adalah memasukkan semuanya ke dalam fungsi ready(). kita telah melihatnya berkali-kali; tidak perlu penjelasan lebih lanjut di sana. Hal kedua dalam JavaScript adalah selector yang dilampirkan ke event click(). Selector ini sedikit lebih kompleks daripada yang kita lihat sebelumnya:

```
$(“input[name=’trim’]”)
```

Selector itu mencari semua elemen <input> pada halaman tetapi kemudian menggunakan filter untuk mendapatkan hanya elemen input yang memiliki nama "trim". Dalam hal ini, elemen tersebut sesuai dengan tombol radio pada formulir. Perhatikan di sini penggunaan dua tanda kutip yang berbeda. Selektor keseluruhan diapit oleh tanda kutip ganda sedangkan kata trim diapit oleh tanda kutip tunggal. kita melakukan ini karena jika tidak, jQuery akan menjadi sangat bingung dan berpikir kita bermaksud menutup selector. Selector dirantai ke fungsi click() yang menangani kejadian klik. Dalam fungsi click(), nilai item yang diklik diperiksa:

```
if ($(this).val() == “deluxe”) {
```

Itu dilakukan melalui selektor $(this) dan fungsi val(). Jika nilainya "deluxe", maka semua kotak centang dicentang dengan baris ini:
$(“input[name=’option’]”).attr(“checked”,true);

Baris itu lagi menggunakan selector dan filter, tetapi kali ini mendapatkan semua kotak centang dengan nama "opsi". Kotak centang yang dipilih kemudian dicentang, berkat fungsi attr() yang dijelaskan di bab sebelumnya. Lain jika kemudian digunakan untuk melihat apakah tombol radio Biasa dipilih:

else if ($(this).val() == “plain”) {

Jika tombol radio itu dipilih, maka kotak centang 'opsi' tidak dicentang.

$(“input[name=’option’]”).attr(“checked”,false);

Setelah event handler klik ditutup, yang lain dibuat. Kali ini, pawang terhubung ke kotak centang:

$(“input[name=’option’]”).click(function(event) {

Jika seseorang mengklik salah satu kotak centang, apakah akan mencentangnya atau menghapus centang, kita harus mengaktifkan tombol radio "custom". Itu dicapai dengan daftar ini:

$(“#custom”).attr(“checked”,”checked”);

Saat memilih Deluxe, kotak centang akan secara otomatis dicentang dan jika kita memilih Biasa, kotak tersebut tidak akan dicentang. Mengklik salah satu kotak centang satu per satu menghasilkan tombol radio Kustom menjadi dipilih. Meskipun contoh ini menunjukkan cara bekerja dengan formulir dan peristiwa klik, kita sebenarnya dapat melampirkan pengendali peristiwa klik ke elemen apa pun di halaman. Lihat http://api.jquery.com/click untuk informasi lebih lanjut tentang event handler klik di jQuery.

Menonton gerakan mouse

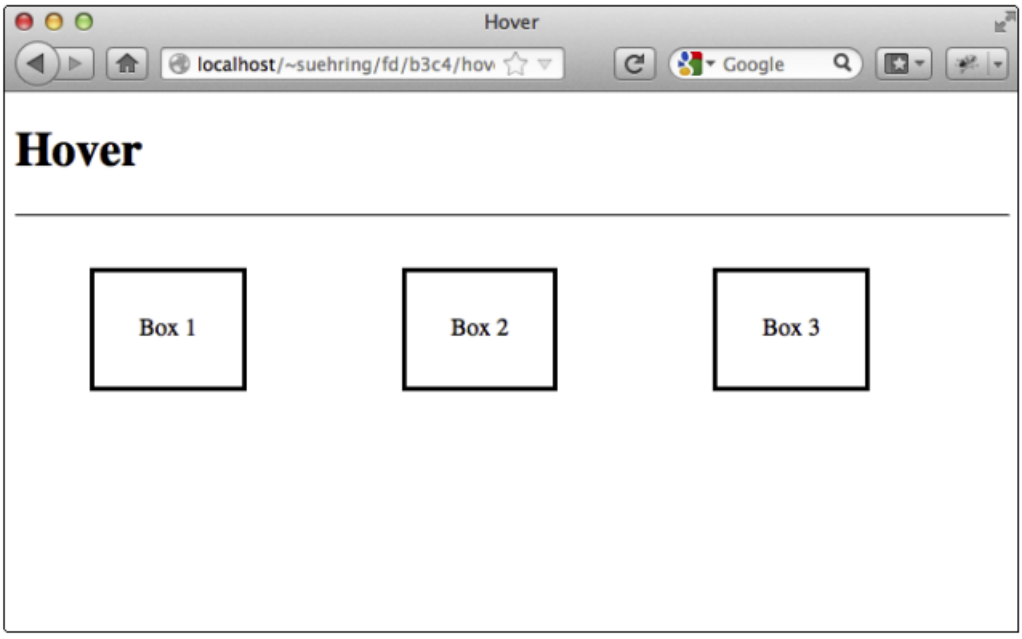
Ada beberapa item menarik seputar pergerakan mouse atau alat penunjuk. Misalnya, kita dapat mengubah elemen saat mouse melayang di atasnya atau saat mouse meninggalkan elemen. Bagian ini menunjukkan event handler hover, yang menangani hover over dan saat mouse meninggalkan elemen. kita dapat meniru pengendali acara hover melalui CSS meskipun dukungan untuk pseudoclass CSS :hover tidak tersedia di browser lama.

Contoh 3 Daftar Membuat Tiga Kotak dalam HTML

```
<!doctype html>
<html>
<head>
<title>Hover</title>
<script type=”text/javascript”
```

```
 src=”http://ajax.aspnetcdn.com/ajax/jQuery/jquery1.8.1.min.js”>
</script>
<style type=”text/css”>
.box {
 margin: 50px;
 padding: 30px;
 width: 50px;
 border: 3px solid black;
}
.colorBox {
 background-color: #abacab;
}
</style>
</head>
<body>
<h1>Hover</h1>
<hr>
<br />
<br />
<br />
<span class=”box”>Box 1</span>
<span class=”box”>Box 2</span>
<span class=”box”>Box 3</span>
</body>
</html>
```

Halaman yang dilihat di browser akan terlihat seperti gambar dibawah ini:



Gambar 9.5 HTML untuk efek hover.

Untuk efek hover, kita menambahkan JavaScript agar ketika sebuah kotak diarahkan dengan mouse, warna latar belakang berubah. Untuk membuat efek ini, JavaScript berikut digunakan di dalam halaman, di lokasi biasanya tepat di atas tag penutup </body>.

```
<script type=”text/javascript”>
$(document).ready(function() {
 $(“.box”).hover(
 //Hover over
 function() {
 $(this).addClass(“colorBox”);
 },
 //Hover out
 function() {
 $(this).removeClass(“colorBox”);
 }
 );
});
```

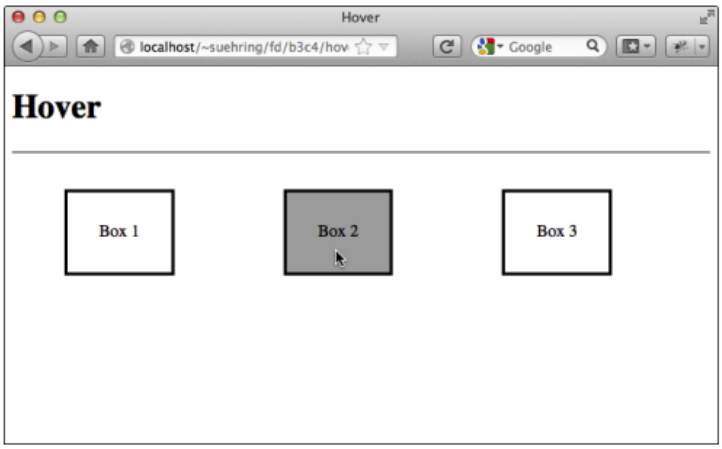
Kode ini menempatkan semuanya di fungsi jQuery ready(), seperti biasa. Selanjutnya, semua item dengan kelas "box" dipilih dan fungsi hover() dirantai ke mereka:

```
$(“.box”).hover(
```

Fungsi hover() membutuhkan dua argumen: apa yang harus dilakukan saat hover berlaku dan apa yang harus dilakukan saat hover selesai. Jadi masing-masing fungsi tersebut dibuat. Bagian pertama menambahkan kelas yang disebut "colorBox", yang hanya mengubah warna latar belakang menjadi abu-abu. Fungsi kedua, diterapkan saat mouse bergerak keluar dari elemen yang dipilih, menghapus kelas “colorBox”, seperti yang ditunjukkan di sini.

```
//Hover over
 function() {
 $(this).addClass(“colorBox”);
 },
 //Hover out
 function() {
 $(this).removeClass(“colorBox”);
 }
 );
```

Hasilnya, dengan mouse melayang di atas elemen berlabel Kotak 2, ditunjukkan pada Gambar dibawah ini. Kotak berubah menjadi abu-abu (#abacab).



Gambar 9.6 Melayang di atas elemen.

9.4 REAKSI PERISTIWA KEYBOARD

Sama seperti kita dapat bereaksi terhadap peristiwa mouse, kita juga dapat bereaksi terhadap peristiwa keyboard. Hal-hal seperti menonton ketika tombol tertentu ditekan, atau lebih umum, hanya menghitung berapa kali tombol ditekan, semuanya dapat dilakukan dengan JavaScript. Bagian ini melihat dua contoh JavaScript untuk bereaksi terhadap kejadian keyboard.

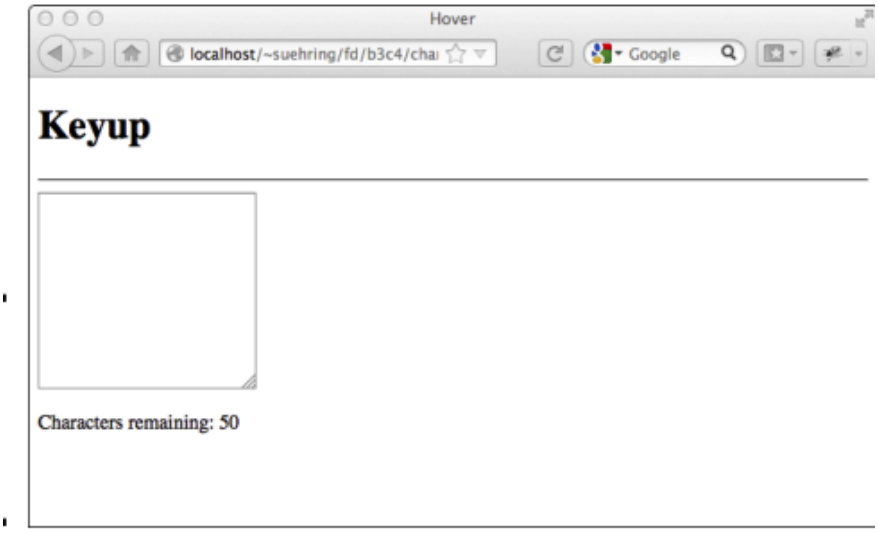
Menghitung karakter

Jika kita pernah menggunakan Twitter, kita telah melihat contoh kotak teks yang menghitung mundur saat kita mengetik. Banyak formulir kontak juga menyertakan fungsi serupa, di mana kita dapat mengetik pesan hanya hingga sejumlah karakter tertentu. Daftar 4 menunjukkan beberapa contoh HTML untuk membuat kotak teks kecil, yang disebut area teks dalam bahasa HTML.

Contoh 4 Daftar Membuat textarea HTML

```
<!doctype html>
<html>
<head>
<title>Hover</title>
<script type=”text/javascript”
 src=”http://ajax.aspnetcdn.com/ajax/jQuery/jquery1.8.1.min.js”>
</script>
</head>
<body>
<h1>Keyup</h1>
<hr>
<form action=”#”>
<textarea rows=”10” cols=”20” id=”message” name=”message”>
</textarea>
<p>Characters remaining: <span id=”remaining”>50</span></p>
</form>
</body>
</html>
```

Halaman yang dilihat di browser terlihat seperti gambar berikut ini.



Gambar 9.7 Area teks kecil untuk input

Menambahkan JavaScript untuk menghitung karakter terlihat seperti ini:

```
<script type=”text/javascript”>
$(document).ready(function() {
 var maxCharacters = 50;
 $(“#message”).on(“keyup”,function() {
 var currentVal = $(“#message”).val().length;
 var totalRemaining = maxCharacters - currentVal;
 $(“#remaining”).text(totalRemaining);
 });
});
</script>
```

JavaScript ini pertama-tama menetapkan variabel dengan jumlah karakter maksimum yang diizinkan, 50. Kemudian elemen dengan ID "message" dipilih. Sebuah event handler dilampirkan ke elemen yang dipilih. Event handler dilampirkan dengan fungsi on(), yang merupakan event handler generik. Penangan acara yang kita lihat sejauh ini sangat umum sehingga orang-orang di jQuery membuat fungsi khusus untuk menanganinya. Jadi hal-hal seperti submit(), click(), dan hover() semuanya memiliki acaranya sendiri. Namun, semua peristiwa lain dilampirkan menggunakan fungsi on(). Argumen pertama untuk fungsi on() adalah nama acara yang harus diperhatikan. Dalam hal ini, kita ingin menonton acara "keyup" untuk mengetahui kapan tombol ditekan dan kemudian dilepaskan.

Hal pertama yang kita lakukan setelah acara keyup diaktifkan adalah menghitung jumlah karakter di bidang teks. Itu dicapai dengan baris kode ini:

```
var currentVal = $(“#message”).val().length;
```

Sekarang kita tahu karakter maksimum yang diizinkan, 50, dan kita tahu jumlah karakter saat ini di bidang. Selanjutnya: matematika. kita perlu mengurangi jumlah karakter saat ini dalam

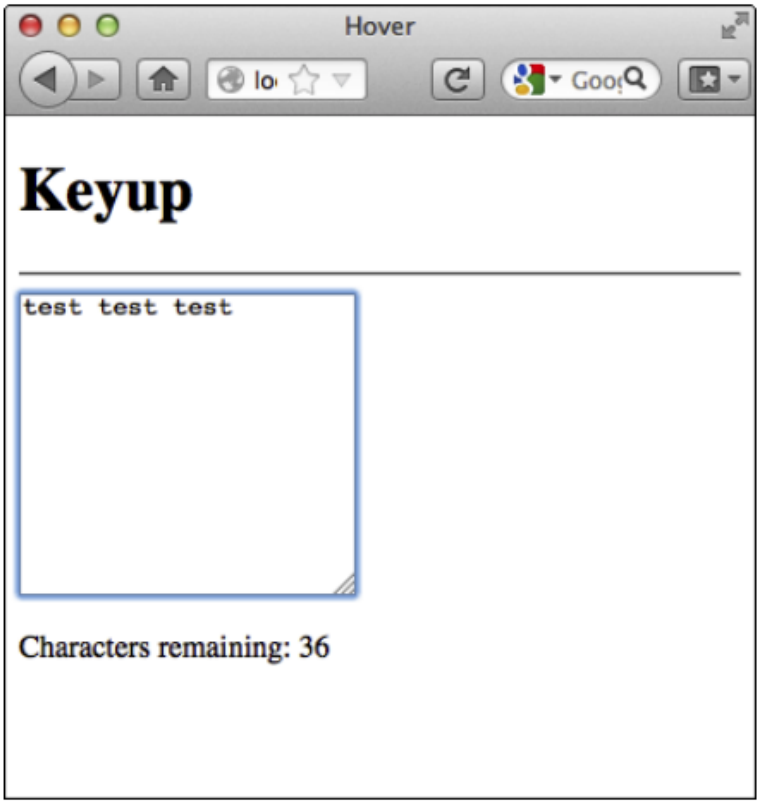
variabel currentVal dari karakter maksimum yang diizinkan dalam variabel maxCharacters. Hasilnya akan ditempatkan dalam variabel baru yang disebut totalRemaining.

var totalRemaining = maxCharacters - currentVal;

Hal terakhir yang harus dilakukan adalah menempatkan nilai tersebut ke dalam elemen yang menunjukkan karakter yang tersisa:

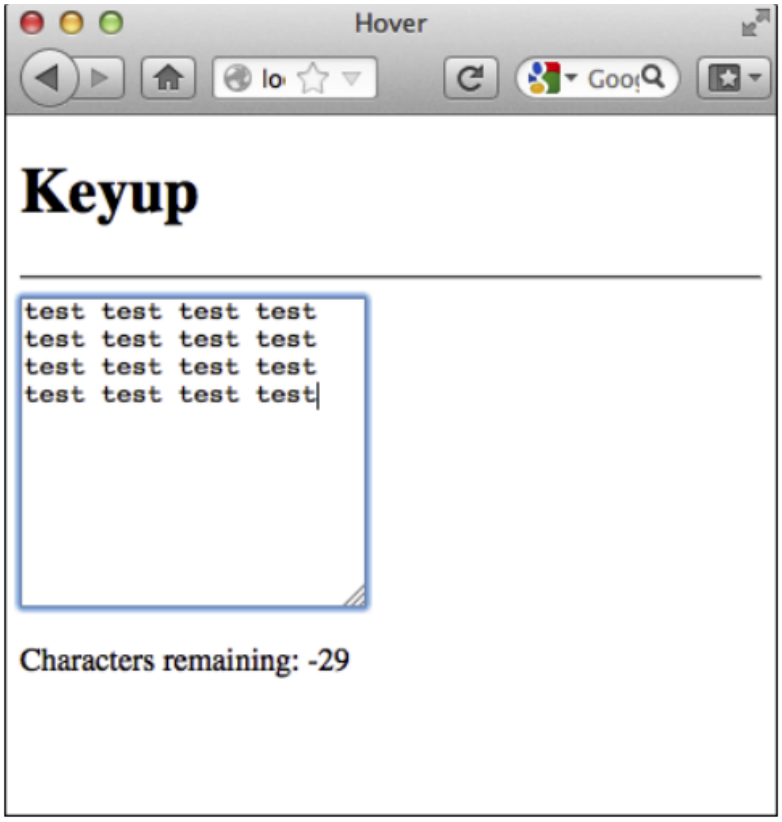
$("#remaining").text(totalRemaining);

Sekarang ketika kita mengetik ke dalam textarea, jumlah karakter yang tersisa dihitung mundur. Gambar dibawah ini menunjukkan sebuah contoh.



Gambar 9.8 Menampilkan karakter yang tersisa

Sekarang kita tahu bagaimana mereka melakukan ini di Twitter! Namun, jika kita mengetik banyak ke dalam formulir, kita akan melihat bahwa kita benar-benar dapat terus mengetik melewati 50 karakter. Penghitung akan masuk ke angka negatif.



Gambar 9.9 Memasukkan karakter yang melewati batas

Mencegah input karakter

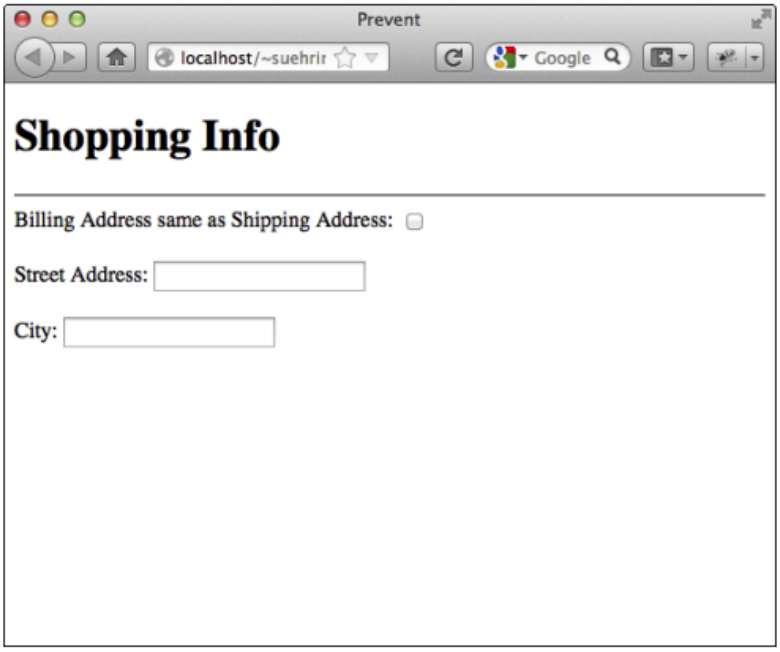
Anda dapat menggunakan JavaScript untuk menonaktifkan bidang formulir. Misalnya, banyak situs menggunakan kombinasi alamat pengiriman dan penagihan di mana pengguna mengklik kotak centang untuk menunjukkan bahwa alamat penagihan sama dengan alamat pengiriman. Daftar dibawah ini menunjukkan cuplikan HTML untuk halaman tersebut.

Contoh 5 Daftar kode untuk Halaman Info Pengiriman

```
<!doctype html>
<html>
<head>
<title>Prevent</title>
<script type=”text/javascript”
 src=”http://ajax.aspnetcdn.com/ajax/jQuery/jquery1.8.1.min.js”>
</script>
</head>
<body>
<h1>Shopping Info</h1>
<hr>
<form action=”#”>
Billing Address same as Shipping Address:
<input type=”checkbox” name=”billingAddress”
id=”billingAddress”>
<br />
<br />
```

```
Street Address:
<input class=”baddr” type=”text” name=”street” id=”street”>
<br />
<br />
City:
<input class=”baddr” type=”text” name=”city” id=”city”>
</form>
</body>
</html>
```

Halaman yang dilihat di browser terlihat seperti gambar dibawah ini.



Gambar 9.10 Cuplikan untuk halaman info penagihan.

Pada kenyataannya, halaman info penagihan akan menangkap lebih banyak informasi, seperti negara bagian dan kode pos sebagai permulaan, tetapi ini memberi kita sedikit gambaran tentang jenis halaman yang kita siapkan untuk contoh ini. Menambahkan JavaScript untuk menonaktifkan kotak teks tersebut terlihat seperti ini:

```
$(“#billingAddress”).click(function() {
 if ($(“#billingAddress”).attr(“checked”) ==
“checked”) {
 $(“.baddr”).val(“”);
 $(“.baddr”).attr(“disabled”,”disabled”);
 } else if ($(“#billingAddress”).attr(“checked”) ==
undefined) {
 $(“.baddr”).removeAttr(“disabled”);
 }
});
})
```

Kode dimulai dengan fungsi ready(), tentu saja. Setelah itu, clickevent handler ditambahkan ke kotak centang. Jika kotak centang billingAddress dicentang, maka nilai dari bidang formulir akan dihapus dan bidang formulir tersebut dinonaktifkan. Jika kotak centang billingAddress tidak dicentang, maka atribut yang dinonaktifkan akan dihapus, berkat fungsi removeAttr(). Perhatikan dalam contoh ini penggunaan kelas pada bidang teks untuk dinonaktifkan. Menggunakan kelas, yang disebut "baddr" untuk contoh ini, memudahkan pengelompokan mereka untuk selector jQuery.

BAB 10
TROUBLESHOOT PROGRAM JAVASCRIPT

10.1 MEMPEKERJAKAN TEKNIK PEMECAHAN MASALAH JAVASCRIPT DASAR

Teknik utama untuk mengatasi masalah teknis adalah berhenti. Hentikan apa yang kita lakukan dan tetap tenang. Kami telah melihat banyak orang yang sangat pintar goyah ketika ada yang salah — dan ada yang salah. Jadi kami ulangi: Saran terbaik yang dapat kami berikan untuk pemecahan masalah adalah berhenti dan tetap tenang. Setelah kita selesai melakukannya, lihat masalah dari sudut yang berbeda dan kurangi menjadi bagian-bagian kecil. Misalnya, kita akan mengalami masalah dengan halaman web yang kita programkan. Masalahnya bisa jadi halaman tidak dimuat, halaman tidak terlihat benar, atau yang lainnya. Pertimbangkan apakah masalahnya ada pada database, dengan PHP, dengan server, dengan JavaScript, atau tidak satu pun dari itu — atau lebih dari satu.

Jika menurut kita masalahnya ada pada JavaScript, keluarkan JavaScript dari halaman sepenuhnya. Kemudian tambahkan kembali secara perlahan. Cara lain untuk memecahkan masalah JavaScript adalah dengan menambahkan fungsi alert() di berbagai tempat. Saat kita melakukan pemecahan masalah, kita dapat menambahkan komentar dalam kode untuk membantu upaya pemecahan masalah Anda. Kemudian di bab ini, kami menunjukkan kepada kita sebuah plug-in untuk Firefox yang sangat membantu dalam memecahkan masalah JavaScript.

Menambahkan peringatan

Anda telah melihat dan menggunakan fungsi alert(). Cara yang baik untuk memecahkan masalah JavaScript adalah dengan menggunakan fungsi alert() dalam kode untuk menunjukkan nilai variabel atau hanya untuk menunjukkan di mana kita berada dalam kode.

Kejadian umum dengan JavaScript adalah kita akan memiliki program yang panjang dan kita tidak akan dapat memahami mengapa program tersebut tidak berhasil. Menambahkan peringatan pada baris 1 program dapat menunjukkan kepada kita apakah itu dipanggil atau tidak. Kemudian menambahkan peringatan pada baris 50 akan menunjukkan jika program sudah sejauh itu. Jika tidak, maka kita tahu bahwa pasti ada masalah di antara baris 1 dan 50. Menambahkan peringatan adalah cara yang mudah dan efisien untuk membantu memecahkan masalah yang rumit. kita cukup menambahkan kode seperti ini:

```
alert(“Just executed this code!”)
```

Atau, jika kita perlu menunjukkan nilai variabel bernama myVariable, kita akan melakukan ini:

```
alert(myVariable);
```

Perhatikan kurangnya tanda kutip di sekitar nama variabel. Jika kita memberi tanda kutip di sekitar nama variabel, JavaScript akan menafsirkannya sebagai string lama biasa sehingga kita hanya akan melihat nama variabel dan bukan isinya. kita juga bisa menjadi mewah dan menggabungkannya:

```
alert(“The value of the variable is: “ + myVariable);
```

Kode itu tidak hanya akan menampilkan teks ramah, tetapi juga nilai dari variabel itu sendiri. Berhati-hatilah dalam menggunakan lansiran dalam struktur loop karena kita harus mengabaikan setiap peringatan secara manual di browser. Pastikan juga untuk menghapus peringatan apa pun sebelum merilis kode untuk penggunaan produksi di situs web kita yang sebenarnya. Jika tidak, pengunjung situs web akan menganggap situs tersebut sangat mengganggu.

Menggunakan komentar dalam JavaScript

Komentar membantu mendokumentasikan kode, yang dapat sangat membantu saat kita perlu memperbarui atau mengubah kode nanti. kita juga dapat menggunakan komentar untuk membantu pemecahan masalah. Komentar tidak hanya berguna untuk mendokumentasikan kode, tetapi juga dapat membantu pemecahan masalah. Misalnya, jika kita telah mengidentifikasi area kode yang bermasalah, kita dapat mengomentari bagian itu sementara untuk mengatasi masalah tersebut.

Dalam JavaScript, komentar datang dalam dua bentuk.

- //: kita dapat menggunakan garis miring ganda sebagai komentar satu baris.
- /* dan */: kita dapat menggunakan format slash-asterisk untuk komentar yang mencakup beberapa baris.

Berikut ini contoh komentar satu baris:

```
// This is a single line comment.
var myVariable = 77;
```

Dalam contoh ini, baris pertama yang dimulai dengan dua garis miring akan diabaikan oleh interpreter JavaScript tetapi itu akan membuat dan menetapkan variabel di baris berikutnya karena itu tidak dikomentari. kita dapat mengomentari baris kode. Jadi pada contoh sebelumnya, jika kita ingin mengomentari baris var myVariable = 77, kita cukup menambahkan dua garis miring di depan baris, seperti ini:

```
// var myVariable = 77;
```

Apa pun yang muncul setelah dua garis miring hingga akhir baris dianggap sebagai komentar dan diabaikan. Jika kita ingin mengomentari beberapa baris atau membuat komentar multi-baris, kita menggunakan garis miring tunggal dengan tanda bintang untuk membuka blok komentar dan kemudian tanda bintang dan garis miring untuk menutup blok. Apa pun yang muncul di antara ini akan diabaikan. Ini contohnya:

```
/*
Everything within this area is a comment and will be ignored.
This includes normal lines like this and also code lines,
like:
if (myVariable = “something”) {
 return false;
}
*/
```

Dalam contoh itu, semua kode akan diabaikan karena muncul di blok komentar. Penting untuk diperhatikan saat menggunakan komentar multi-baris bahwa kita tidak dapat membuat sarangnya. Misalnya, ini tidak valid:

```
/*
Another multi-line comment
/* A comment in a comment */
Ending the comment
*/
```

Setelah juru bahasa JavaScript menemukan kode */ akhir, ia akan menganggap bahwa komentar telah berakhir dan karena itu tidak akan tahu apa yang harus dilakukan dengan */ berikutnya yang ditemuinya. kita masih dapat menggunakan konstruksi garis miring ganda, seperti ini:

```
/*
Another multi-line comment
// A comment within a comment
Ending the comment
*/
```

JavaScript dapat dilihat oleh semua orang yang melihat sumber halaman web Anda, jadi berhati-hatilah dengan apa yang kita tampilkan di komentar. Menempatkan informasi sensitif (atau menyinggung) di komentar kode dapat membuat kita mendapat masalah!

10.2 MENGIDENTIFIKASI MASALAH JAVASCRIPT DENGAN FIREBUG

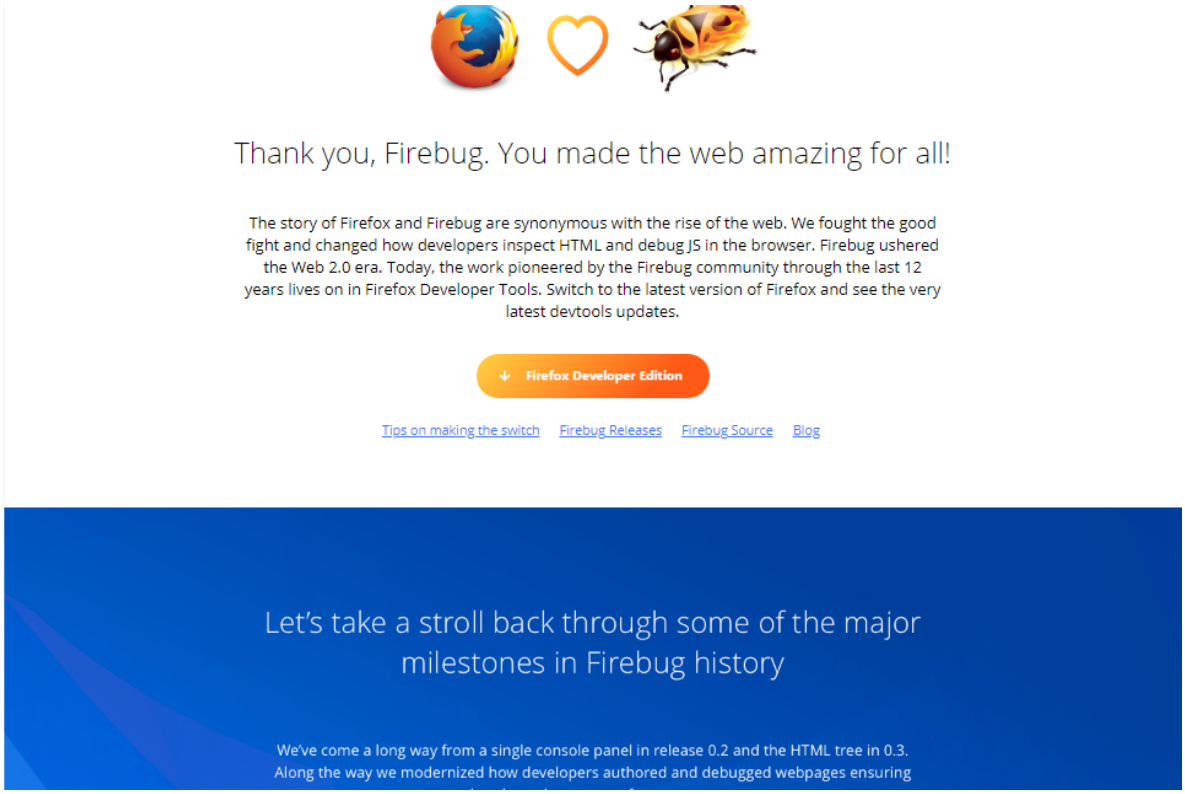
Peringatan dan komentar berfungsi dengan baik sebagai alat pemecahan masalah dalam JavaScript. Namun, alat yang sangat diperlukan untuk programmer JavaScript adalah alat yang disebut Firebug, yang merupakan add-on untuk browser web Firefox. Ini berisi alat debugging JavaScript canggih serta beberapa alat lain yang terkait dengan pengembangan web. Firebug mengidentifikasi masalah dengan kode JavaScript saat dijalankan dan membantu menemukan solusi dengan cepat. Bagian ini membahas cara menginstal Firebug dan kemudian cara menggunakannya. Kami berasumsi bahwa kita telah menginstal Firefox. Jika tidak, dapatkan di www.mozilla.org sebelum melanjutkan ke bagian berikutnya. Firebug bukan satu-satunya alat yang dapat digunakan untuk debugging. Internet Explorer memiliki alat yang disebut Alat Pengembang F12, dan Chrome juga memiliki seperangkat alat pengembangnya sendiri. Namun, Firebug cukup kuat dan mudah digunakan, jadi itulah yang kami bahas di sini.

Memasang Firebug

Anda menginstal Firebug sebagai add-on untuk Firefox. Instalasi mudah tetapi membutuhkan restart Firefox sesudahnya. Ikuti prosedur ini untuk menginstal Firebug. Meskipun, prosedur ini mungkin sedikit berubah saat kita membaca ini, keseluruhan prosesnya sama: Gunakan Firefox untuk mengunduh dan menginstal add-on Firebug. Namun, lokasi dan nama tautan dapat berubah.

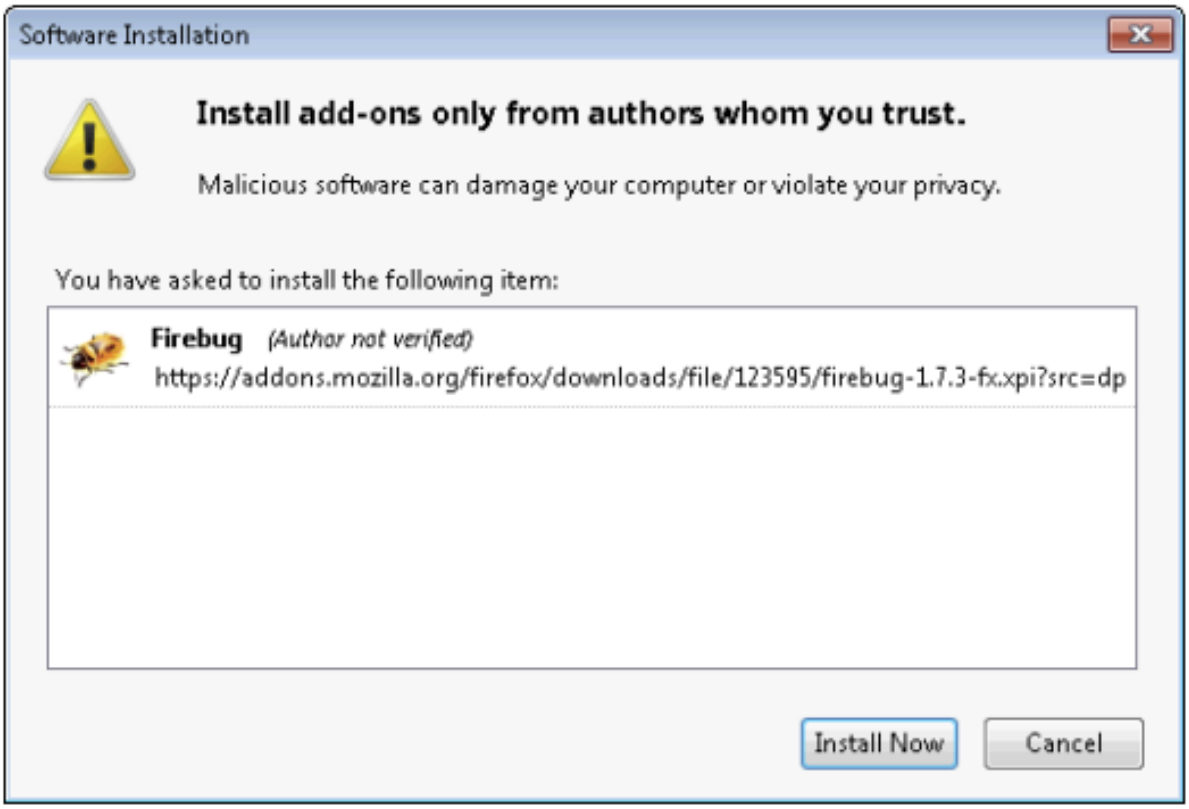
1. Buka Firefox.
2. Arahkan ke http://getfirebug.com.

3. Di halaman beranda Firebug, klik Instal Firebug (atau tautan serupa/ tombol, jika verbiage berubah pada saat kita membaca ini).
4. Pada halaman Unduhan, klik untuk mengunduh versi Firebug yang direkomendasikan. Ini biasanya akan menjadi tautan teratas untuk versi Firefox yang lebih baru. kita memulai proses pengunduhan dengan mengeklik tautan Unduh, yang akan membawa kita ke halaman Pengaya.
5. Pada halaman Pengaya Mozilla untuk Firebug, yang ditunjukkan pada Gambar dibawah ini, klik tombol Tambahkan ke Firefox.
   Firebug akan mengunduh dan menginstal.



Gambar 10.1 Halaman Pengaya Mozilla untuk Firebug.

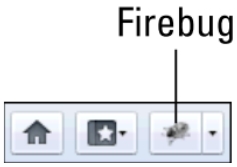
6. Pada dialog Instalasi Perangkat Lunak yang ditunjukkan pada Gambar 10.2, klik Instal Sekarang.
7. Saat kita diminta untuk me-restart Firefox, klik Restart Now.
   Firefox akan restart dan kita akan diperlihatkan halaman beranda Firebug lagi.



Gambar 10.2 install add-on Firebug

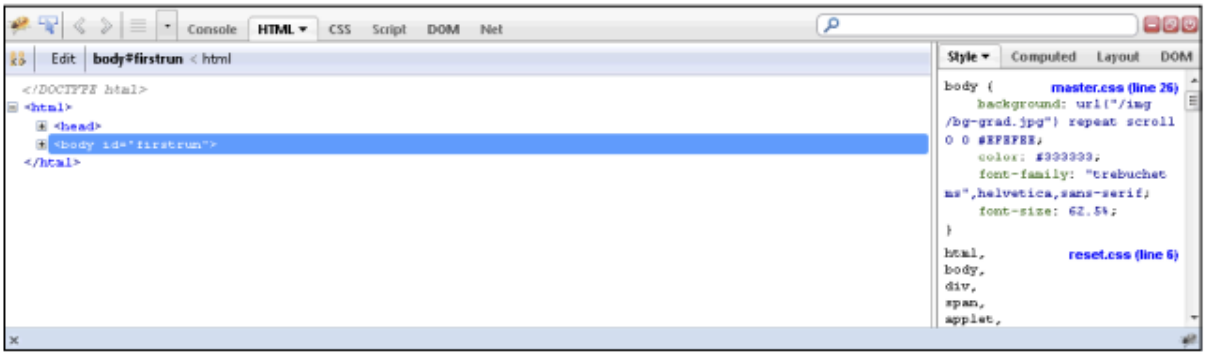
Menggunakan Firebug

Ketika Firebug dimuat, itu akan dimasukkan ke dalam bilah alat di Firefox. Ikon Firebug biasanya ditemukan di sudut kanan atas Firefox. Lihat dibawah ini untuk contoh tampilan ikon Firebug.



Gambar 10.3 Ikon Firebug di Firefox.

Mengklik ikon Firebug menampilkan konsol Firebug, yang ditunjukkan pada gambar dibawah ini, untuk halaman apa pun yang sedang kita buka.

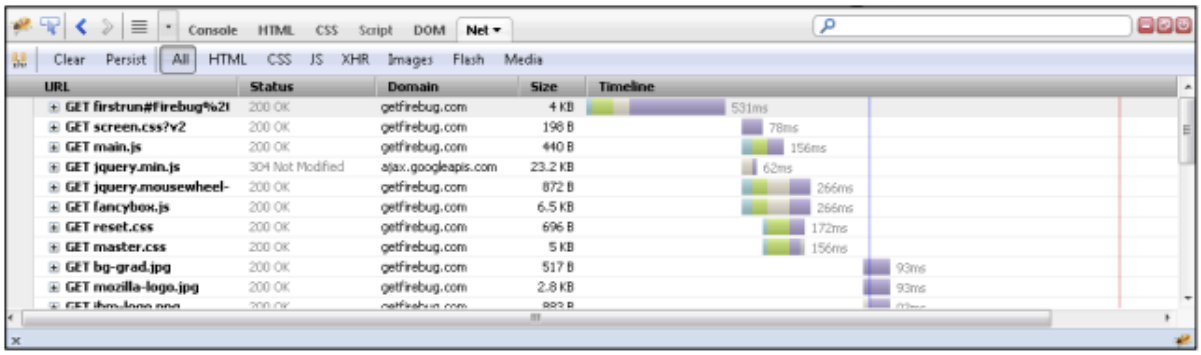


Gambar 10.4 Konsol Firebug.

Anda dapat mengklik berbagai tab dalam antarmuka Firebug untuk melihat beberapa opsi. Saat men-debug JavaScript, kita akan sering menggunakan Panel Konsol. Namun, Panel Konsol mungkin dinonaktifkan secara default, seperti yang ada pada gambar dibawah ini:

Gambar 10.5 Panel Konsol dinonaktifkan secara default di Firebug

Mengaktifkan Panel Konsol melibatkan mengklik panah bawah di sebelah kata Konsol dan memilih Diaktifkan. Saat kita melakukannya, Panel Konsol akan diaktifkan. Namun, kita perlu memuat ulang halaman agar kesalahan atau informasi lain muncul di Panel Konsol. Menekan kombinasi tombol Ctrl+R atau Command+R akan memuat ulang halaman di Firefox. Proses yang sama diperlukan untuk mengaktifkan panel lain di Firebug, seperti Panel Net. Panel Net menunjukkan pengambilan elemen pada halaman, termasuk semua JavaScript, CSS, gambar, dan elemen lainnya. Ini juga menunjukkan kode respons, yang terkadang berguna untuk menunjukkan bahwa file tertentu tidak dimuat. Panel Net juga menampilkan informasi waktu sehingga kita dapat melihat berapa lama waktu yang dibutuhkan browser untuk mengunduh berbagai elemen halaman juga. Panel Net ditunjukkan pada gambar berikut:



Gambar 10.6 Panel Net di Firebug

Jika kita menggunakan Firebug atau browser Chrome, kita juga dapat memanfaatkan cara lain untuk pemecahan masalah, yang disebut console.log. Menggunakan console.log, hasil dari apa pun yang kita debug akan ditampilkan di dalam tab Konsol area alat pengembang. Fitur console.log digunakan seperti peringatan:

```
console.log("hello world");
```

Luangkan waktu dengan Firebug untuk mengetahui kegunaannya dan bagaimana Firebug dapat membantu pemrograman JavaScript Anda. Setelah kita terbiasa dengan alat ini, itu akan menjadi sangat diperlukan bagi Anda.

BAB 11
VALIDASI JAVASCRIPT DAN PHP DAN PENANGANAN KESALAHAN

Dengan dasar yang kuat baik dalam PHP dan JavaScript, inilah saatnya untuk menyatukan teknologi ini untuk membuat formulir web yang seramah mungkin. Kami akan menggunakan PHP untuk membuat formulir dan JavaScript untuk melakukan validasi sisi klien untuk memastikan bahwa data selengkap dan sebenar mungkin sebelum dikirimkan. Validasi akhir dari input kemudian akan dilakukan oleh PHP, yang jika perlu, akan menampilkan kembali formulir tersebut kepada pengguna untuk modifikasi lebih lanjut. Dalam prosesnya, bab ini akan mencakup validasi dan ekspresi reguler di JavaScript dan PHP.

11.1 MEMVALIDASI INPUT PENGGUNA DENGAN JAVASCRIPT

Validasi JavaScript harus dianggap sebagai bantuan lebih kepada pengguna kita daripada ke situs web kita karena, seperti yang telah saya tekankan berkali-kali, kita tidak dapat mempercayai data apa pun yang dikirimkan ke server Anda, bahkan jika itu seharusnya telah divalidasi dengan JavaScript. Ini karena peretas dapat dengan mudah mensimulasikan formulir web kita dan mengirimkan data apa pun yang mereka pilih. Alasan lain kita tidak dapat mengandalkan JavaScript untuk melakukan semua validasi input kita adalah karena beberapa pengguna menonaktifkan JavaScript, atau menggunakan browser yang tidak mendukungnya. Jadi jenis validasi terbaik yang harus dilakukan dalam JavaScript adalah memeriksa bahwa bidang memiliki konten jika tidak dibiarkan kosong, memastikan bahwa alamat email sesuai dengan format yang tepat, dan memastikan bahwa nilai yang dimasukkan berada dalam batas yang diharapkan.

Dokumen Validasi.html (Bagian Satu)

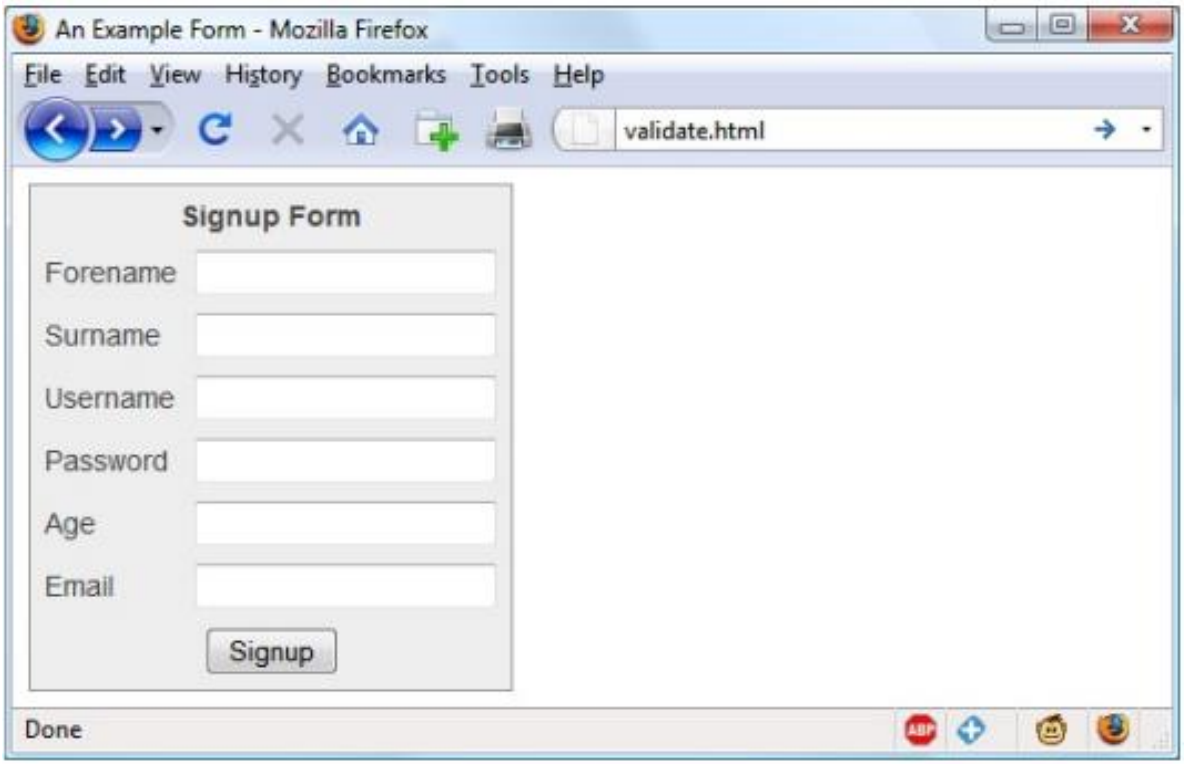
Mari kita mulai dengan formulir pendaftaran umum, umum di sebagian besar situs yang menawarkan keanggotaan atau pengguna terdaftar. Input yang diminta adalah nama depan, nama belakang, nama pengguna, kata sandi, usia, dan alamat email. Contoh 1 menyediakan template yang baik untuk formulir seperti itu.

Contoh 1. Formulir dengan validasi JavaScript (bagian satu)

```
+= validateEmail(form.email.value)
if (fail == "") return true
else { alert(fail); return false }
}
</script>
</head>
<body>
<table border="0" cellpadding="2" cellspacing="5" bgcolor="#eeeeee">
<th colspan="2" align="center">Signup Form</th>
<form method="post" action="adduser.php" onsubmit="return validate(this)">
<tr><td>Forename</td>
<td><input type="text" maxlength="32" name="forename"></td></tr>
<tr><td>Surname</td>
<td><input type="text" maxlength="32" name="surname"></td></tr>
```

```
<tr><td>Username</td>
<td><input type="text" maxlength="16" name="username"></td></tr>
<tr><td>Password</td>
<td><input type="text" maxlength="12" name="password"></td></tr>
<tr><td>Age</td>
<td><input type="text" maxlength="3" name="age"></td></tr>
<tr><td>Email</td>
<td><input type="text" maxlength="64" name="email"></td></tr>
<tr><td colspan="2" align="center"><input type="submit"
value="Signup"></td></tr>
</form>
</table>
</body>
</html>
```

Seperti berdiri, formulir ini akan ditampilkan dengan benar tetapi tidak akan memvalidasi sendiri, karena fungsi validasi utama belum ditambahkan. Meskipun demikian, simpan sebagai validasi.html, dan ketika kita memanggilnya di browser Anda, itu akan terlihat seperti Gambar 11.1.



Gambar 11.1 Output dari Contoh 1

Mari kita lihat bagaimana dokumen ini dibuat. Beberapa baris pertama mengatur dokumen dan menggunakan sedikit CSS untuk membuat formulir terlihat sedikit kurang polos. Bagian-bagian dari dokumen yang terkait dengan JavaScript datang berikutnya dan ditampilkan dalam huruf tebal. Di antara tag <script> dan </script> terdapat satu fungsi yang disebut validasi yang dengan sendirinya memanggil enam fungsi lain untuk memvalidasi setiap bidang input formulir. Kami akan segera membahas fungsi-fungsi ini. Untuk saat ini saya hanya

akan menjelaskan bahwa mereka mengembalikan string kosong jika bidang divalidasi, atau pesan kesalahan jika gagal. Jika ada kesalahan, baris terakhir skrip akan memunculkan kotak peringatan untuk menampilkannya.

Setelah melewati validasi, fungsi validasi mengembalikan nilai true; jika tidak, ia mengembalikan false. Nilai kembalian dari validasi penting, karena jika mengembalikan salah, formulir dicegah untuk dikirimkan. Ini memungkinkan pengguna untuk menutup pop up peringatan dan membuat perubahan. Jika benar dikembalikan, tidak ada kesalahan yang ditemukan di bidang formulir sehingga formulir diizinkan untuk dikirimkan.

Bagian kedua dari contoh ini menampilkan HTML untuk formulir dengan setiap bidang dan namanya ditempatkan di dalam baris tabelnya sendiri. Ini adalah HTML yang cukup sederhana, dengan pengecualian pernyataan onSubmit="return memvalidasi(ini)" di dalam tag <form> pembuka. Menggunakan onSubmit, kita dapat menyebabkan fungsi pilihan kita dipanggil saat formulir dikirimkan. Fungsi itu dapat melakukan beberapa pemeriksaan dan mengembalikan nilai benar atau salah untuk menandakan apakah formulir harus diizinkan untuk dikirimkan. Parameter this adalah objek saat ini (yaitu, formulir ini) dan diteruskan ke fungsi validasi yang baru saja dibahas. Fungsi validasi menerima parameter ini sebagai bentuk objek. Seperti yang kita lihat, satu-satunya JavaScript yang digunakan dalam HTML formulir adalah panggilan untuk kembali terkubur di atribut onSubmit. Browser dengan JavaScript yang dinonaktifkan atau tidak tersedia akan mengabaikan atribut onSubmit, dan HTML akan ditampilkan dengan baik.

Dokumen Validasi.html (Bagian Kedua

Sekarang kita sampai pada Contoh 2, satu set enam fungsi yang melakukan validasi bidang formulir yang sebenarnya. Saya sarankan kita mengetik semua bagian kedua ini dan menyimpannya di bagian <script> ... </script> pada Contoh 1, yang seharusnya sudah kita simpan sebagai validasi.html.

Contoh 2. Formulir dengan validasi JavaScript (bagian dua)

```
function validateForename(field)
{
return (field == "") ? "No Forename was entered.\n" : ""
}
function validateSurname(field)
{
return (field == "") ? "No Surname was entered.\n" : ""
}
function validateUsername(field)
{
if (field == "") return "No Username was entered.\n"
else if (field.length < 5)
return "Usernames must be at least 5 characters.\n"
else if ([^a-zA-Z0-9_-].test(field))
return "Only a-z, A-Z, 0-9, - and _ allowed in Usernames.\n"
return ""
}
function validatePassword(field)
{
```

```
if (field == "") return "No Password was entered.\n"
else if (field.length < 6)
return "Passwords must be at least 6 characters.\n"
else if (![a-z].test(field) || ! [A-Z].test(field) ||
![0-9].test(field))
return "Passwords require one each of a-z, A-Z and 0-9.\n"
return ""
}
function validateAge(field)
{
if (isNaN(field)) return "No Age was entered.\n"
else if (field < 18 || field > 110)
return "Age must be between 18 and 110.\n"
return ""
}
function validateEmail(field)
{
if (field == "") return "No Email was entered.\n"
else if (!((field.indexOf(".") > 0) &&
(field.indexOf("@") > 0)) ||
[^a-zA-Z0-9.@_-].test(field))
return "The Email address is invalid.\n"
return ""
}
```

Memvalidasi nama depan

validasiForename adalah fungsi yang cukup singkat yang menerima bidang parameter, yang merupakan nilai nama depan yang diteruskan ke sana oleh fungsi validasi. Jika nilai ini adalah string kosong, pesan kesalahan dikembalikan; jika tidak, string kosong dikembalikan untuk menandakan bahwa tidak ada kesalahan yang ditemukan. Jika pengguna memasukkan spasi di bidang ini, itu akan diterima oleh validasiForename, meskipun kosong untuk semua maksud dan tujuan. kita dapat memperbaikinya dengan menambahkan pernyataan tambahan untuk memangkas spasi dari bidang sebelum memeriksa apakah kosong, gunakan ekspresi reguler untuk memastikan ada sesuatu selain spasi di bidang, atau—seperti yang saya lakukan di sini—biarkan pengguna membuat kesalahan dan biarkan program PHP menangkapnya di server.

Memvalidasi nama keluarga

Fungsi ValidasiSurname hampir identik dengan ValidasiForename dalam kesalahan yang dikembalikan hanya jika nama keluarga yang diberikan adalah string kosong. Saya memilih untuk tidak membatasi karakter yang diizinkan di salah satu bidang nama untuk memungkinkan kemungkinan seperti karakter non-Inggris dan beraksen.

Memvalidasi nama pengguna

Fungsi ValidasiUsername sedikit lebih menarik, karena memiliki pekerjaan yang lebih rumit. Itu harus memungkinkan hanya melalui karakter a–z, A–Z, 0–9, _ dan -, dan memastikan bahwa nama pengguna setidaknya memiliki panjang lima karakter. Pernyataan if ... else dimulai dengan mengembalikan kesalahan jika bidang belum diisi. Jika bukan string kosong, tetapi panjangnya kurang dari lima karakter, pesan kesalahan lain akan ditampilkan. Kemudian

fungsi pengujian JavaScript dipanggil, meneruskan ekspresi reguler (yang cocok dengan karakter apa pun yang bukan salah satu yang diizinkan) untuk dicocokkan dengan bidang (lihat bagian Ekspresi Reguler). Jika bahkan satu karakter yang bukan salah satu karakter yang dapat diterima ditemukan, maka fungsi pengujian mengembalikan nilai true, dan selanjutnya validasiUser mengembalikan string kesalahan.

Memvalidasi kata sandi

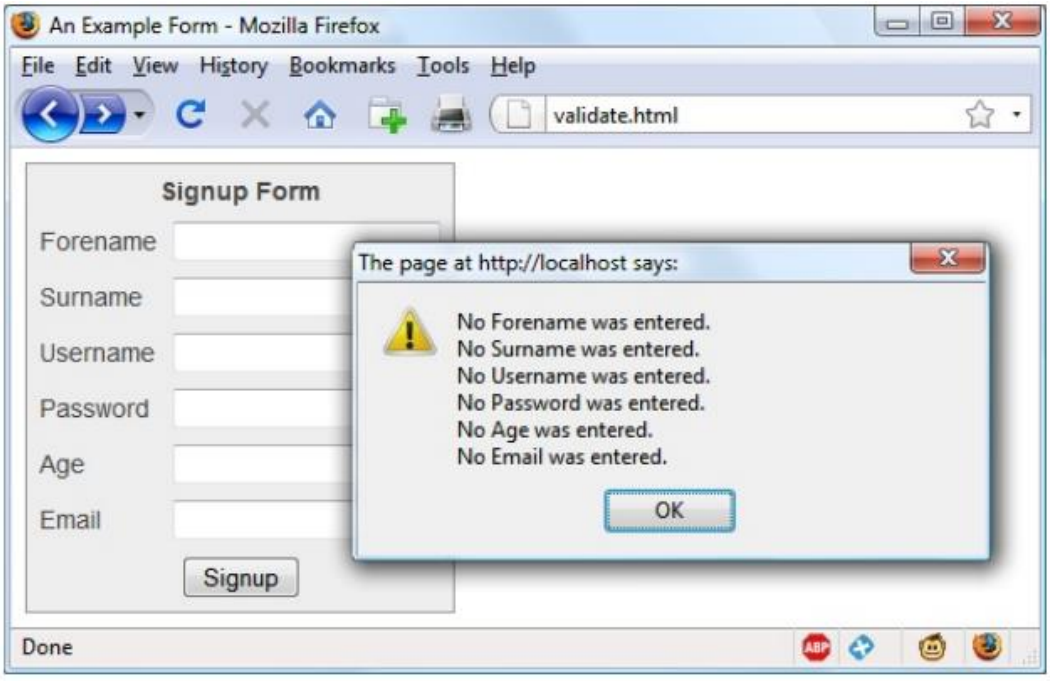
Teknik serupa digunakan dalam fungsi validasiPassword. Pertama, fungsi memeriksa apakah bidang kosong, dan jika ya, mengembalikan kesalahan. Selanjutnya, pesan kesalahan ditampilkan jika kata sandi lebih pendek dari enam karakter. Salah satu persyaratan yang kami terapkan pada kata sandi adalah kata sandi harus memiliki setidaknya satu karakter huruf kecil, huruf besar, dan numerik, sehingga fungsi pengujian dipanggil tiga kali, sekali untuk setiap kasus ini. Jika salah satu dari mereka mengembalikan false, salah satu persyaratan tidak terpenuhi dan pesan kesalahan dikembalikan. Jika tidak, string kosong dikembalikan untuk menandakan bahwa kata sandinya OK.

Memvalidasi usia

validasiAge mengembalikan pesan kesalahan jika bidang bukan angka (ditentukan oleh panggilan ke fungsi isNaN), atau jika usia yang dimasukkan lebih rendah dari 18 atau lebih besar dari 110. Aplikasi kita mungkin memiliki persyaratan usia yang berbeda atau tidak sama sekali. Sekali lagi, setelah validasi berhasil, string kosong dikembalikan.

Memvalidasi email

Dalam contoh terakhir dan paling rumit, alamat email divalidasi dengan validasiEmail. Setelah memeriksa apakah ada sesuatu yang benar-benar dimasukkan, dan mengembalikan pesan kesalahan jika tidak, fungsi tersebut memanggil fungsi JavaScript indexOf dua kali. Pertama kali pemeriksaan dilakukan untuk memastikan ada titik (.) di suatu tempat dari setidaknya karakter kedua bidang, dan pemeriksaan kedua bahwa simbol @ muncul di suatu tempat pada atau setelah karakter kedua. Jika kedua pemeriksaan tersebut terpenuhi, fungsi pengujian dipanggil untuk melihat apakah ada karakter yang tidak diizinkan muncul di bidang.



Gambar 11.2 Validasi formulir JavaScript dalam tindakan

Jika salah satu tes ini gagal, pesan kesalahan dikembalikan. Karakter yang diizinkan dalam alamat email adalah huruf besar dan huruf kecil, angka, dan karakter _, -, titik, dan @, sebagaimana dirinci dalam ekspresi reguler yang diteruskan ke metode pengujian. Jika tidak ada kesalahan yang ditemukan, string kosong akan dikembalikan untuk menunjukkan validasi yang berhasil. Pada baris terakhir, skrip dan dokumen ditutup. Gambar 11.2 menunjukkan hasil pengguna mengklik tombol Signup tanpa mengisi kolom apapun.

Menggunakan file JavaScript terpisah

Tentu saja, karena konstruksinya generik dan dapat diterapkan ke banyak jenis validasi yang mungkin kita perlukan, enam fungsi ini membuat kandidat ideal untuk pindah ke file JavaScript terpisah. Misalnya, kita dapat memberi nama file tersebut seperti validasi_fungsi.js dan memasukkannya tepat setelah bagian skrip awal pada Contoh 1, menggunakan pernyataan berikut:

```
<script src="validate_functions.js"></script>
```

11.2 EKSPRESI REGULER

Mari kita lihat sedikit lebih dekat pada pencocokan pola yang telah kita lakukan. Kami telah mencapainya menggunakan ekspresi reguler, yang didukung oleh JavaScript dan PHP. Mereka memungkinkan untuk membangun algoritma pencocokan pola yang paling kuat dalam satu ekspresi.

Mencocokkan melalui metakarakter

Setiap ekspresi reguler harus diapit oleh garis miring. Dalam garis miring ini, karakter tertentu memiliki arti khusus; mereka disebut metakarakter. Misalnya, tanda bintang (*) memiliki arti yang mirip dengan apa yang kita lihat jika kita menggunakan shell atau Command Prompt Windows (tetapi tidak persis sama). Tanda bintang berarti, "teks yang kita coba cocokkan mungkin memiliki sejumlah karakter sebelumnya—atau tidak sama sekali." Misalnya, katakanlah kita sedang mencari nama "Le Guin" dan tahu bahwa seseorang mungkin mengejanya dengan atau tanpa spasi. Karena teks ditata dengan aneh (misalnya, seseorang mungkin telah memasukkan spasi ekstra untuk membenarkan garis), kita mungkin harus mencari baris seperti:

```
The difficulty of classifying Le Guin's works
```

Jadi, kita harus mencocokkan "LeGuin", serta "Le" dan "Guin" yang dipisahkan oleh sejumlah spasi. Solusinya adalah mengikuti spasi dengan tanda bintang:

```
/Le *Guin/
```

Ada lebih banyak daripada nama "Le Guin" di baris, tapi tidak apa-apa. Selama ekspresi reguler cocok dengan beberapa bagian baris, fungsi pengujian mengembalikan nilai sebenarnya. Bagaimana jika penting untuk memastikan bahwa baris tidak berisi apa pun selain "Le Guin"? Saya akan menunjukkan cara memastikannya nanti. Misalkan kita tahu selalu ada setidaknya satu ruang. Dalam hal ini, kita dapat menggunakan tanda tambah (+), karena memerlukan setidaknya satu dari karakter sebelumnya:

```
Le +Guin
```

Pencocokan karakter kabur

Titik (.) sangat berguna, karena bisa cocok dengan apa pun kecuali baris baru. Misalkan kita mencari tag HTML, yang dimulai dengan < dan diakhiri dengan >. Cara sederhana untuk melakukannya adalah:

```
/<. *>/
```

Titik cocok dengan karakter apa pun dan * memperluasnya agar cocok dengan nol atau lebih karakter, jadi ini mengatakan, "cocokkan apa pun yang terletak di antara < dan >, bahkan jika tidak ada apa-apa." kita akan mencocokkan <>, ,
, dan seterusnya. Tetapi jika kita tidak ingin mencocokkan kasing kosong, <>, kita harus menggunakan + daripada *, seperti ini:

```
< . +>
```

Tanda plus memperluas titik untuk mencocokkan satu atau lebih karakter, dengan mengatakan, "cocokkan apa pun yang terletak di antara < dan > selama setidaknya ada satu karakter di antara mereka." kita akan mencocokkan dan , <h1> dan </h1>, dan tag dengan atribut seperti:

```
<a href="www.mozilla.org">
```

Sayangnya, tanda plus terus cocok hingga > terakhir di telepon, jadi kita mungkin akan mendapatkan:

```
<h1><b>Introduction</b></h1
```

Lebih dari satu tag! Saya akan menunjukkan solusi yang lebih baik nanti di bagian ini.

Jika kita ingin mencocokkan karakter titik itu sendiri (.), kita harus menghindarinya dengan menempatkan garis miring terbalik (\) di depannya, karena jika tidak, itu adalah metakarakter dan cocok dengan apa pun. Sebagai contoh, misalkan kita ingin mencocokkan angka floating-point 5.0. Ekspresi regulernya adalah:

```
5\.0
```

Garis miring terbalik dapat menghindari karakter meta apa pun, termasuk garis miring terbalik lainnya (jika kita mencoba mencocokkan garis miring terbalik dalam teks). Namun, untuk membuat segalanya sedikit membingungkan, kita akan melihat nanti bagaimana garis miring terbalik terkadang memberi arti khusus pada karakter berikut. Kami baru saja mencocokkan angka floating-point. Tapi mungkin kita ingin mencocokkan 5. dan juga 5.0, karena keduanya memiliki arti yang sama dengan angka floating-point. kita juga ingin mencocokkan 5,00, 5.000, dan seterusnya—sejumlah nol diperbolehkan. kita dapat melakukan ini dengan menambahkan tanda bintang, seperti yang kita lihat:

```
/5\.0*/
```

Pengelompokan melalui tanda kurung

Misalkan kita ingin mencocokkan kekuatan kenaikan unit, seperti kilo, mega, giga, dan tera. Dengan kata lain, kita ingin semua yang berikut ini cocok:

1,000
1,000,000
1,000,000,000
1,000,000,000,000
...

Tanda plus juga berfungsi di sini, tetapi kita perlu mengelompokkan string ,000 sehingga tanda plus cocok dengan semuanya. Ekspresi regulernya adalah:

```
1(,000)+
```

Tanda kurung berarti "perlakukan ini sebagai grup ketika kita menerapkan sesuatu seperti tanda tambah." 1,00,000 dan 1,000,00 tidak akan cocok karena teks harus memiliki 1 diikuti oleh satu atau lebih kelompok koma yang diikuti oleh tiga nol. Spasi setelah karakter + menunjukkan bahwa kecocokan harus berakhir saat ditemukan spasi. Tanpa itu, 1,000,00 akan salah cocok karena hanya 1.000 pertama yang akan diperhitungkan, dan sisanya ,00 akan diabaikan. Memerlukan spasi sesudahnya memastikan bahwa pencocokan akan berlanjut hingga akhir nomor.

Kelas karakter

Terkadang kita ingin mencocokkan sesuatu yang kabur, tetapi tidak terlalu luas sehingga kita ingin menggunakan titik. Kekaburan adalah kekuatan besar dari ekspresi reguler: mereka memungkinkan kita untuk menjadi tepat atau tidak jelas seperti yang kita inginkan. Salah satu fitur utama yang mendukung pencocokan fuzzy adalah pasangan tanda kurung siku, []. Itu cocok dengan satu karakter, seperti titik, tetapi di dalam tanda kurung kita memasukkan daftar hal-hal yang bisa cocok. Jika salah satu karakter tersebut muncul, teksnya cocok. Misalnya, jika kita ingin mencocokkan ejaan Amerika "abu-abu" dan ejaan Inggris "abu-abu", kita dapat menentukan:

```
gr[ae]y
```

Setelah gr dalam teks yang kita cocokkan, bisa ada a atau e. Tetapi harus ada hanya satu dari mereka: apa pun yang kita masukkan ke dalam tanda kurung cocok dengan tepat satu karakter. Kelompok karakter di dalam tanda kurung disebut kelas karakter.

Menunjukkan rentang

Di dalam tanda kurung, kita dapat menggunakan tanda hubung (-) untuk menunjukkan rentang. Salah satu tugas yang sangat umum adalah mencocokkan satu digit, yang dapat kita lakukan dengan rentang sebagai berikut:

```
[0-9]
```

Digit adalah item yang umum dalam ekspresi reguler sehingga satu karakter adalah disediakan untuk mewakili mereka: \d. kita dapat menggunakannya sebagai pengganti kurung ekspresi biasa untuk mencocokkan angka:

```
\d
```

Negosiasi

Salah satu fitur penting lainnya dari tanda kurung siku adalah negasi dari kelas karakter. kita dapat mengubah seluruh kelas karakter dengan menempatkan tanda sisipan (^) setelah tanda kurung buka. Ini artinya, “Cocokkan semua karakter kecuali yang berikut ini.” Jadi katakanlah kita ingin menemukan contoh "Yahoo" yang tidak memiliki tanda seru berikut. (Nama perusahaan secara resmi mengandung tanda seru!) kita dapat melakukannya sebagai berikut:

Yahoo[^!]

Kelas karakter terdiri dari satu karakter—tanda seru—tetapi dibalikkan oleh ^ sebelumnya. Ini sebenarnya bukan solusi yang bagus untuk masalah—misalnya, gagal jika "Yahoo" berada di akhir baris, karena itu tidak diikuti oleh apa pun, sedangkan tanda kurung harus cocok dengan karakter. Solusi yang lebih baik melibatkan pandangan ke depan yang negatif (mencocokkan sesuatu yang tidak diikuti oleh hal lain).

11.3 BEBERAPA CONTOH YANG LEBIH RUMIT

Dengan pemahaman tentang kelas karakter dan negasi, kita sekarang siap untuk melihat solusi yang lebih baik untuk masalah pencocokan tag HTML. Solusi ini menghindari melewati akhir satu tag, tetapi masih mencocokkan tag seperti dan serta tag dengan atribut seperti:

<a href="www.mozilla.org">

Salah satu solusinya adalah:

<[^>]+>

Ekspresi reguler itu mungkin terlihat seperti saya baru saja menjatuhkan cangkir teh saya di keyboard, tetapi itu benar-benar valid dan sangat berguna. Mari kita pecahkan.
Elemen-elemennya adalah:
/
Garis miring pembuka yang menunjukkan ini adalah ekspresi reguler.
<
Braket pembuka dari tag HTML. Ini sangat cocok; itu bukan metakarakter.
[^>]
Kelas karakter. ^> yang disematkan berarti "cocok dengan apa pun kecuali braket sudut penutup."
+
Mengizinkan sejumlah karakter untuk mencocokkan [^>] sebelumnya, selama setidaknya ada salah satunya.
>
Tanda kurung penutup dari tag HTML. Ini cocok persis.
/
Garis miring penutup yang menunjukkan akhir dari ekspresi reguler.

Catatan: Solusi lain untuk masalah pencocokan tag HTML adalah dengan menggunakan operasi yang tidak serakah. Secara default, pencocokan pola adalah serakah, mengembalikan kecocokan terpanjang yang mungkin. Pencocokan nongreedy menemukan kecocokan sesingkat mungkin, , tetapi ada detail lebih lanjut di http://oreilly.com/catalog/regex/chapter/ch04. Sekarang kita akan melihat salah satu ekspresi dimana fungsi validasiUsername digunakan:
[^a-zA-Z0-9_-]

_
Garis bawah.
-
Sebuah tanda hubung.
]
Braket penutup yang mengakhiri kelas karakter.
/
Garis miring penutup yang menunjukkan akhir dari ekspresi reguler.

Ada dua metakarakter penting lainnya. Mereka "menjangkar" ekspresi reguler dengan mengharuskannya muncul di tempat tertentu. Jika tanda sisipan (^) muncul di awal ekspresi reguler, ekspresi harus muncul di awal baris teks; jika tidak, itu tidak cocok. Demikian pula, jika tanda dolar ($) muncul di akhir ekspresi reguler, ekspresi harus muncul di akhir baris teks. Kami akan menyelesaikan eksplorasi dasar ekspresi reguler kami dengan menjawab pertanyaan yang diajukan sebelumnya: misalkan kita ingin memastikan tidak ada tambahan pada baris selain ekspresi reguler? Bagaimana jika kita menginginkan garis yang memiliki "Le Guin" dan tidak ada yang lain? Kita dapat melakukannya dengan mengubah ekspresi reguler sebelumnya untuk mengaitkan kedua ujungnya:

/^Le *Guin$/

11.4 RINGKASAN KARAKTER META

Tabel 11.1 menunjukkan metakarakter yang tersedia dalam ekspresi reguler.

Metakarakter	Deskripsi
/	Ekspresi reguler awal dan akhir
.	Mencocokkan karakter tunggal kecuali bari baru
*Element**	Mencocokkan *element* nol atau lebih
Element+	Mencocokkan *element* satu atau lebih
Element?	Mencocokkan *element* nol atau satu kali
[character]	Mencocokkan karakter dari karakter yang ada di dalam kurung dengan karakter yang ada di dalam kurung
[^character]	Mencocokkan satu karakter yang tidak termasuk dalam tanda kurung
(regex)	Memperlakukan regex sebagai grup untuk menghitung atau mengikuti *, +, atau ?
Left|right	Cocok dengan *left* atau *kanan*
[l-r]	Mencocokkan rentang karakter antara l dan r
^	Membutuhkan kecocokan untuk berada di awal string
$	Membutuhkan kecocokan untuk berada di ujung string

\b	Mencocokkan batas kata
\B	Cocok di mana tidak ada batas kata
\d	Cocok dengan satu digit
\D	Cocok dengan satu nondigit
\n	Cocok dengan karakter baris baru
\s	Cocok dengan karakter spasi putih
\S	Cocok dengan karakter bukan spasi
\t	Cocok dengan karakter tab
\w	Mencocokkan karakter kata (a-z, A-Z, 0-9, dan _)
\W	Mencocokkan karakter bukan kata (apa pun kecuali a-z, A-Z, 0-9, dan _)
\x	x (berguna jika x adalah metakarakter, tetapi kita benar-benar menginginkan x)
{n}	Cocok persis n kali
{n, }	Cocok n kali atau lebih
{min, max}	Cocok setidaknya *min* dan paling banyak *max*

Dengan tabel ini, dan melihat kembali ekspresi [^a-zA-Z0-9_], kita dapat melihat bahwa itu dapat dengan mudah disingkat menjadi [^\w] karena metakarakter tunggal \w (dengan huruf kecil w) menentukan karakter a-z, A-Z, 0-9, dan_.

Sebenarnya, kita bisa lebih pintar dari itu, karena metakarakter \W (dengan huruf besar W) menentukan semua karakter kecuali untuk a-z, A-Z, 0-9, dan _. Oleh karena itu, kita juga dapat menghapus karakter meta ^ dan cukup menggunakan [\W] untuk ekspresinya. Untuk memberi kita lebih banyak ide tentang bagaimana semua ini bekerja, Tabel 11.2 menunjukkan berbagai ekspresi dan pola yang cocok.

Tabel 11.2 Beberapa contoh ekspresi reguler

Metakarakter	**Deskripsi**
r	r pertama di The quick brown
rec[ei][ei]ve	Baik menerima atau menerima (tetapi juga menerima atau menerima)
rec[ei]{2}ve	Baik menerima atau menerima (tetapi juga menerima atau menerima)
rec(ei|ie)ve	Baik menerima atau menerima (tetapi tidak menerima atau menerima
cat	Kata kucing di aku suka kucing dan anjing
cat|dog	Salah satu dari kata-kata kucing atau anjing di saya suka kucing dan anjing
\.	. (\ diperlukan karena . adalah metakarakter)
5\.0*	5., 5.0, 5.00, 5.000, dst.
[a-f]	Setiap karakter a, b, c, d, e atau f
cats$	Hanya kucing terakhir di Kucing saya yang kucing ramah
^my	Hanya kucing pertama saya yang menjadi hewan peliharaan saya
\d{2,3}	Angka dua atau tiga digit (00 hingga 999)
7(,000)+	7,000; 7,000,000; 7,000,000,000; 7,000,000,000,000; dan lainnya
[\w]+	Kata apa pun dari satu atau lebih karakter
[\w]{5}	Kata lima huruf apa saja

11.5 MODIFIER UMUM

Beberapa pengubah tambahan tersedia untuk ekspresi reguler:

- /g memungkinkan pencocokan "global". Saat menggunakan fungsi ganti, tentukan pengubah ini untuk mengganti semua kecocokan, bukan hanya yang pertama.
- /i membuat ekspresi reguler tidak peka huruf besar/kecil. Jadi, alih-alih [a zA-Z], kita dapat menentukan [a-z]i atau [A-Z]i.
- /m mengaktifkan mode multiline, di mana tanda sisipan (^) dan dolar ($) cocok sebelum dan sesudah setiap baris baru dalam string subjek. Biasanya, dalam string multiline, ^ hanya cocok di awal string dan $ hanya cocok di akhir string.

Misalnya, ekspresi catg akan cocok dengan kedua kemunculan kata kucing dalam kalimat saya suka kucing dan kucing seperti saya. Demikian pula, doggi akan mencocokkan kedua kemunculan kata anjing (Anjing dan anjing) dalam kalimat Anjing seperti anjing lainnya, karena kita dapat menggunakan penentu ini bersama-sama.

11.6 MENGGUNAKAN EKSPRESI REGULER DALAM JAVASCRIPT

Dalam JavaScript, kita akan menggunakan ekspresi reguler sebagian besar dalam dua metode: uji (yang telah kita lihat) dan ganti. Sedangkan test hanya memberi tahu kita apakah argumennya cocok dengan ekspresi reguler, replace mengambil parameter kedua: string untuk mengganti teks yang cocok. Seperti kebanyakan fungsi, replace menghasilkan string baru sebagai nilai kembalian; itu tidak mengubah masukan. Untuk membandingkan kedua metode tersebut, pernyataan berikut hanya mengembalikan true untuk memberi tahu kami bahwa kata cat muncul setidaknya sekali di suatu tempat di dalam string:

```
document.write(catsi.test("Cats are funny. I like cats."))
```

Tetapi pernyataan berikut menggantikan kedua kemunculan kata kucing dengan kata anjing, mencetak hasilnya. Pencarian harus global (/g) untuk menemukan semua kemunculan, dan case-insensitive (/i) untuk menemukan Cat dengan huruf kapital:

```
document.write("Cats are friendly. I like cats.".replace(catsgi,"dogs"))
```

Jika kita mencoba pernyataan tersebut, kita akan melihat batasan penggantian: karena ini menggantikan teks dengan string yang kita perintahkan untuk digunakan, kata pertama Cats diganti dengan dog, bukan Dogs.

11.7 MENGGUNAKAN EKSPRESI REGULER DI PHP

Fungsi ekspresi reguler paling umum yang mungkin kita gunakan di PHP adalah preg_match, preg_match_all, dan preg_replace. Untuk menguji apakah kata cat muncul di mana saja dalam string, dalam kombinasi huruf besar dan kecil apa pun, kita dapat menggunakan preg_match seperti ini:

```
$n = preg_match("catsi", "Cats are crazy. I like cats.")
```

Karena PHP menggunakan 1 untuk TRUE dan 0 untuk FALSE, pernyataan sebelumnya menetapkan $n ke 1. Argumen pertama adalah ekspresi reguler dan yang kedua adalah teks

yang akan dicocokkan. Tapi preg_match sebenarnya jauh lebih kuat dan rumit, karena dibutuhkan argumen ketiga yang menunjukkan teks apa yang cocok:

```
$n = preg_match("catsi", "Cats are curious. I like cats.", $match);
echo "$n Matches: $match[0]";
```

Argumen ketiga adalah array (di sini, diberi nama $match). Fungsi ini menempatkan teks yang cocok ke dalam elemen pertama, jadi jika pencocokan berhasil, kita dapat menemukan teks yang cocok di $match[0]. Dalam contoh ini, output memberi tahu kami bahwa teks yang cocok dikapitalisasi:

```
1 Matches: Cats
```

Jika kita ingin menemukan semua kecocokan, kita menggunakan fungsi preg_match_all, seperti ini:

```
$n = preg_match_all("catsi", "Cats are strange. I like cats.", $match);
echo "$n Matches: ";
for ($j=0 ; $j < $n ; ++$j) echo $match[0][$j]." ";
```

Seperti sebelumnya, $match diteruskan ke fungsi dan elemen $match[0] diberikan kecocokan yang dibuat, tetapi kali ini sebagai subarray. Untuk menampilkan subarray, contoh ini mengulanginya dengan for loop. Saat kita ingin mengganti bagian dari string, kita dapat menggunakan preg_replace seperti yang ditunjukkan di sini. Contoh ini menggantikan semua kemunculan kata kucing dengan kata anjing, apa pun kasusnya:

```
echo preg_replace("catsi", "dogs", "Cats are furry. I like cats.");
```

11.8 MENAMPILKAN KEMBALI FORMULIR SETELAH VALIDASI PHP

Oke, kembali ke validasi formulir. Sejauh ini kita telah membuat dokumen HTML validasi.html, yang akan dikirim ke program PHP adduser.php, tetapi hanya jika JavaScript memvalidasi bidang atau jika JavaScript dinonaktifkan atau tidak tersedia. Jadi sekarang saatnya untuk membuat adduser.php untuk menerima formulir yang diposting, melakukan validasinya sendiri, dan kemudian menyajikan kembali formulir tersebut kepada pengunjung jika validasi gagal. Contoh 3 berisi kode yang harus kita ketik dan simpan (atau unduh dari situs web pendamping).

Contoh 3. Program adduser.php

```
<?php // adduser.php
// The PHP code
$forename = $surname = $username = $password = $age = $email = "";
if (isset($_POST['forename']))
$forename = fix_string($_POST['forename']);
if (isset($_POST['surname']))
$surname = fix_string($_POST['surname']);
if (isset($_POST['username']))
```

```
$username = fix_string($_POST['username']);
if (isset($_POST['password']))
$password = fix_string($_POST['password']);
if (isset($_POST['age']))
$age = fix_string($_POST['age']);
if (isset($_POST['email']))
$email = fix_string($_POST['email']);
$fail = validate_forename($forename);
$fail .= validate_surname($surname);
$fail .= validate_username($username);
$fail .= validate_password($password);
$fail .= validate_age($age);
$fail .= validate_email($email);
echo "<!DOCTYPE html>\n<html><head><title>An Example Form</title>";
if ($fail == "")
{
echo "</head><body>Form data successfully validated:
$forename, $surname, $username, $password, $age, $email.</body></html>";
// This is where you would enter the posted fields into a database,
// preferably using hash encryption for the password.
exit;
}
echo <<<_END
<!-- The HTML/JavaScript section -->
<style>
.signup {
border: 1px solid #999999;
font: normal 14px helvetica; color:#444444;
}
</style>
<script>
function validate(form)
{
fail = validateForename(form.forename.value)
fail += validateSurname(form.surname.value)
fail += validateUsername(form.username.value)
fail += validatePassword(form.password.value)
fail += validateAge(form.age.value)
fail += validateEmail(form.email.value)
if (fail == "") return true
else { alert(fail); return false }
}
function validateForename(field)
{
return (field == "") ? "No Forename was entered.\n" : ""
}
```

```
function validateSurname(field)
{
return (field == "") ? "No Surname was entered.\n" : ""
}
function validateUsername(field)
{
if (field == "") return "No Username was entered.\n"
else if (field.length < 5)
return "Usernames must be at least 5 characters.\n"
else if ([^a-zA-Z0-9_-].test(field))
return "Only a-z, A-Z, 0-9, - and _ allowed in Usernames.\n"
return ""
}
function validatePassword(field)
{
if (field == "") return "No Password was entered.\n"
else if (field.length < 6)
return "Passwords must be at least 6 characters.\n"
else if (![a-z].test(field) || ! [A-Z].test(field) ||
![0-9].test(field))
return "Passwords require one each of a-z, A-Z and 0-9.\n"
return ""
}
function validateAge(field)
{
if (isNaN(field)) return "No Age was entered.\n"
else if (field < 18 || field > 110)
return "Age must be between 18 and 110.\n"
return ""
}
ign="center">Signup Form</th>
<tr><td colspan="2">Sorry, the following errors were found<br>
in your form: <p><font color=red size=1><i>$fail</i></font></p>
</td></tr>
<form method="post" action="adduser.php" onSubmit="return validate(this)">
<tr><td>Forename</td>
<td><input type="text" maxlength="32" name="forename" value="forename">
</td></tr><tr><td>Surname</td>
<td><input type="text" maxlength="32" name="surname" value="surname">
</td></tr><tr><td>Username</td>
<td><input type="text" maxlength="16" name="username" value="username">
</td></tr><tr><td>Password</td>
<td><input type="text" maxlength="12" name="password" value="password">
</td></tr><tr><td>Age</td>
<td><input type="text" maxlength="3" name="age" value="age">
</td></tr><tr><td>Email</td>
```

```
<td><input type="text" maxlength="64" name="email" value="email">
</td></tr><tr><td colspan="2" align="center"><input type="submit"
value="Signup"></td></tr>
</form>
</table>
</body>
</html>
_END;
// The PHP functions
function validate_forename($field)
{
return ($field == "") ? "No Forename was entered<br>": "";
}
function validate_surname($field)
{
return($field == "") ? "No Surname was entered<br>" : "";
}
function validate_username($field)
{
if ($field == "") return "No Username was entered<br>";
else if (strlen($field) < 5)
return "Usernames must be at least 5 characters<br>";
else if (preg_match("[^a-zA-Z0-9_-]", $field))
return "Only letters, numbers, - and _ in usernames<br>";
return "";
}
function validate_password($field)
{
if ($field == "") return "No Password was entered<br>";
else if (strlen($field) < 6)
return "Passwords must be at least 6 characters<br>";
else if (!preg_match("[a-z]", $field) ||
!preg_match("[A-Z]", $field) ||
!preg_match("[0-9]", $field))
return "Passwords require 1 each of a-z, A-Z and 0-9<br>";
return "";
}
function validate_age($field)
{
if ($field == "") return "No Age was entered<br>";
else if ($field < 18 || $field > 110)
return "Age must be between 18 and 110<br>";
return "";
}
function validate_email($field)
{
```

```
if ($field == "") return "No Email was entered<br>";
else if (!((strpos($field, ".") > 0) &&
(strpos($field, "@") > 0)) ||
preg_match("[^a-zA-Z0-9.@_-]", $field))
return "The Email address is invalid<br>";
return "";
}
function fix_string($string)
{
if (get_magic_quotes_gpc()) $string = stripslashes($string);
return htmlentities ($string);
}
?>
```

Hasil pengiriman formulir dengan JavaScript dinonaktifkan (dan dua kolom salah diisi) ditunjukkan pada Gambar 11.3. Saya telah menempatkan bagian PHP dari kode ini (dan mengubah bagian HTML) dalam jenis huruf tebal sehingga kita dapat lebih jelas melihat perbedaan antara ini dan Contoh 1 dan 2. Jika kita menelusuri contoh ini (atau mengetiknya atau mengunduhnya dari situs web pendamping), kita akan melihat bahwa kode PHP hampir merupakan tiruan dari kode JavaScript; ekspresi reguler yang sama digunakan untuk memvalidasi setiap bidang dalam fungsi yang sangat mirip. Tapi ada beberapa hal yang perlu diperhatikan. Pertama, fungsi fix_string (tepat di akhir) digunakan untuk membersihkan setiap bidang dan mencegah upaya injeksi kode agar tidak berhasil. Juga, kita akan melihat bahwa HTML dari Contoh 1 telah diulang dalam kode PHP dalam <<<_END ... _END; struktur, menampilkan formulir dengan nilai yang dimasukkan pengunjung sebelumnya. kita melakukan ini hanya dengan menambahkan parameter nilai ekstra ke setiap tag <input> (seperti value="$forename"). Kesopanan ini sangat disarankan sehingga pengguna hanya perlu mengedit nilai yang dimasukkan sebelumnya, dan tidak perlu mengetikkan bidang lagi.

Gambar 11.3 Formulir yang direpresentasikan setelah validasi PHP gagal

Sekarang setelah kita melihat cara menyatukan semua PHP, HTML, dan JavaScript, bab berikutnya akan memperkenalkan Ajax (JavaScript dan XML Asinkron), yang menggunakan panggilan JavaScript ke server di latar belakang untuk memperbarui bagian halaman web dengan mulus. , tanpa harus mengirim ulang seluruh halaman ke server web.

BAB 12
MENGGUNAKAN AJAX

Istilah Ajax pertama kali diciptakan pada tahun 2005. Ini adalah singkatan dari Asynchronous JavaScript and XML, yang, dalam istilah sederhana, berarti menggunakan seperangkat metode yang dibangun ke dalam JavaScript untuk mentransfer data antara browser dan server di latar belakang. Contoh yang sangat baik dari teknologi ini adalah Google Maps, di mana bagian baru dari peta diunduh dari server saat dibutuhkan, tanpa memerlukan penyegaran halaman. Menggunakan Ajax tidak hanya secara substansial mengurangi jumlah data yang harus dikirim bolak-balik, tetapi juga membuat halaman web menjadi dinamis—memungkinkan mereka untuk berperilaku lebih seperti aplikasi mandiri. Hasilnya adalah antarmuka pengguna yang jauh lebih baik dan responsivitas yang lebih baik.

12.1 APA ITU AJAX?

Awal dari Ajax seperti yang digunakan saat ini dimulai dengan rilis Internet Explorer 5 pada tahun 1999, yang memperkenalkan objek ActiveX baru, XMLHttpRequest. ActiveX adalah teknologi Microsoft untuk menandatangani plug-in yang menginstal perangkat lunak tambahan ke komputer Anda. Pengembang browser lain kemudian mengikutinya, tetapi alih-alih menggunakan ActiveX, mereka semua mengimplementasikan fitur tersebut sebagai bagian asli dari penerjemah JavaScript. Namun, bahkan sebelum itu, bentuk awal Ajax telah muncul yang menggunakan bingkai tersembunyi pada halaman yang berinteraksi dengan server di latar belakang. Ruang obrolan adalah pengadopsi awal teknologi ini, menggunakannya untuk polling dan menampilkan posting pesan baru tanpa memerlukan pemuatan ulang halaman. Jadi mari kita lihat bagaimana mengimplementasikan Ajax menggunakan JavaScript.

Menggunakan XMLHttpRequest

Karena perbedaan antara implementasi browser XMLHttpRequest, kita harus membuat fungsi khusus untuk memastikan bahwa kode kita akan berfungsi di semua browser utama.

Untuk melakukan ini, kita harus memahami tiga cara membuat XMLHttpRequest obyek:

- IE 5: request = new ActiveXObject("Microsoft.XMLHTTP")
- IE 6+: request = new ActiveXObject("Msxml2.XMLHTTP")
- All others: request = new XMLHttpRequest ()

Hal ini terjadi karena Microsoft memilih untuk menerapkan perubahan dengan merilis Internet Explorer 6, sementara semua browser lain menggunakan metode yang sedikit berbeda. Oleh karena itu, kode dalam Contoh 1 akan berfungsi untuk semua browser utama yang dirilis selama beberapa tahun terakhir.

Contoh 1. Fungsi Ajax lintas-browser.

```
<script>
function ajaxRequest()
{
try // Non IE Browser?
```

```
{ // Yes
var request = new XMLHttpRequest()
}
catch(e1)
{
try // IE 6+?
{ // Yes
request = new ActiveXObject("Msxml2.XMLHTTP")
}
catch(e2)
{
try // IE 5?
{ // Yes
request = new ActiveXObject("Microsoft.XMLHTTP")
}
catch(e3) // There is no AJAX Support
{
request = false
}
}
}
return request
}
</script>
```

Anda mungkin ingat pengantar penanganan kesalahan di bab sebelumnya, menggunakan konstruksi try ... catch. Contoh 1 adalah ilustrasi sempurna dari utilitasnya, karena menggunakan kata kunci try untuk mengeksekusi perintah non-IE Ajax, dan setelah berhasil, melompat ke pernyataan pengembalian akhir, di mana objek baru dikembalikan. Jika tidak, tangkapan menjebak kesalahan dan perintah berikutnya dijalankan. Sekali lagi, setelah berhasil, objek baru dikembalikan; jika tidak, yang terakhir dari tiga perintah dicoba. Jika upaya itu gagal, maka browser tidak mendukung Ajax dan objek permintaan disetel ke false; jika tidak, objek dikembalikan. Jadi begitulah: fungsi permintaan Ajax lintas-browser yang mungkin ingin kita tambahkan ke perpustakaan fungsi JavaScript yang berguna.

Oke, jadi sekarang kita memiliki cara untuk membuat objek XMLHttpRequest, tetapi apa yang dapat kita lakukan dengan objek ini? Nah, masing-masing dilengkapi dengan seperangkat properti (variabel) dan metode (fungsi), yang dirinci dalam Tabel 12.1 dan 12.2.

Tabel 12.1 Properti objek XMLHttpRequest

Properti	**Deskripsi**
onreadystatechange	Menentukan fungsi penanganan peristiwa yang akan dipanggil setiap kali properti readyState dari suatu objek berubah.
readyState	Properti integer yang melaporkan status permintaan. Itu dapat memiliki salah satu dari nilai-nilai ini: 0 = Tidak diinisialisasi, 1 = Memuat, 2 = Dimuat, 3 = Interaktif, dan 4 = Selesai.

responseText	Data yang dikembalikan oleh server dalam format teks.
responseXML	Data yang dikembalikan oleh server dalam format XML
status	Kode status HTTP dikembalikan oleh server.
statusText	Teks status HTTP dikembalikan oleh server.

Tabel 12.2 metode objek XMLHttpRequest

Metode	**Deskripsi**
abort()	Membatalkan permintaan saat ini.
getAllResponseHeaders	Mengembalikan semua header sebagai string.
getResponseHeader(param)	Mengembalikan nilai param sebagai string.
open('method', 'url', 'asynch')	Menentukan metode HTTP yang akan digunakan (GET atau POST), URL target, dan apakah permintaan harus ditangani secara asinkron (benar atau salah).
send(data)	Mengirim data ke server target menggunakan metode HTTP yang ditentukan.
setRequestHeader('param', 'value')	Menetapkan header dengan pasangan parameter/nilai.

Properti dan metode ini memberi kita kendali atas data apa yang kita kirim ke server dan terima kembali, serta pilihan metode kirim dan terima. Misalnya, kita dapat memilih apakah akan meminta data dalam teks biasa (yang dapat menyertakan HTML dan tag lainnya) atau dalam format XML. kita juga dapat memutuskan apakah kita ingin menggunakan metode POST atau GET untuk mengirim ke server. Mari kita lihat metode POST terlebih dahulu dengan membuat sepasang dokumen yang sangat sederhana: kombinasi HTML dan JavaScript, dan program PHP untuk berinteraksi melalui Ajax dengan yang pertama. Mudah-mudahan kita akan menikmati contoh-contoh ini, karena contoh-contoh tersebut mengilustrasikan apa itu Web 2.0 dan Ajax. Dengan beberapa baris JavaScript, mereka meminta dokumen web dari server web pihak ketiga, yang kemudian dikembalikan ke browser oleh server kita dan ditempatkan di dalam bagian dokumen saat ini.

12.2 PROGRAM AJAX PERTAMA ANDA

Ketik dan simpan kode di Contoh 2 sebagai urlpost.html, tetapi jangan memuatnya ke browser Anda.

Contoh 2. urlpost.html

```
<!DOCTYPE html>
<html>
<head>
<title>AJAX Example</title>
</head>
<body style='text-align:center'>
<h1>Loading a web page into a DIV</h1>
<div id='info'>This sentence will be replaced</div>
<script>
params = "url=amazon.com/gp/aw"
request = new ajaxRequest()
```

```
request.open("POST", "urlpost.php", true)
request.setRequestHeader("Content-type",
"application/x-www-form-urlencoded")
request.setRequestHeader("Content-length", params.length)
request.setRequestHeader("Connection", "close")
request.onreadystatechange = function()
{
if (this.readyState == 4)
{
if (this.status == 200)
{
if (this.responseText != null)
{
document.getElementById('info').innerHTML =
this.responseText
}
else alert("Ajax error: No data received")
}
else alert( "Ajax error: " + this.statusText)
}
}
request.send(params)
function ajaxRequest()
{
try
{
var request = new XMLHttpRequest()
}
catch(e1)
{
try
{
request = new ActiveXObject("Msxml2.XMLHTTP")
}
catch(e2)
{
try
{
request = new ActiveXObject("Microsoft.XMLHTTP")
}
catch(e3)
{
request = false
}
}
}
```

```
return request
}
</script>
</body>
</html>
```

Mari kita lihat dokumen ini dan lihat apa yang dilakukannya, dimulai dengan enam baris pertama, yang hanya mengatur dokumen HTML dan menampilkan heading. Baris berikutnya membuat DIV dengan info ID, berisi teks Kalimat ini akan diganti secara default. Nantinya, teks yang dikembalikan dari panggilan Ajax akan disisipkan di sini. Enam baris berikutnya diperlukan untuk membuat permintaan HTTP POST Ajax. Bagian pertama menetapkan parameter variabel ke pasangan parameter=nilai, yang akan kami kirim ke server. Kemudian permintaan objek Ajax dibuat. Setelah ini, metode open dipanggil untuk mengatur objek agar membuat permintaan POST ke urlpost.php dalam mode asinkron. Tiga baris terakhir dalam grup ini mengatur header yang diperlukan server penerima untuk mengetahui bahwa permintaan POST akan datang.

12.3 PROPERTI READYSTATE

Sekarang kita sampai pada seluk beluk panggilan Ajax, yang semuanya tergantung pada properti readyState. Aspek "asynchronous" dari Ajax memungkinkan browser untuk tetap menerima input pengguna dan mengubah layar, sementara program kami menetapkan properti onreadystatechange untuk memanggil fungsi pilihan kami setiap kali readyState berubah. Dalam hal ini, fungsi inline tanpa nama (atau anonim) telah digunakan, sebagai lawan dari fungsi bernama yang terpisah. Jenis fungsi ini dikenal sebagai fungsi panggilan balik, karena dipanggil kembali setiap kali readyState berubah. Sintaks untuk mengatur fungsi panggilan balik menggunakan fungsi anonim sebaris adalah sebagai berikut:

```
request.onreadystatechange = function()
{
if (this.readyState == 4)
{
// do something
}
}
```

Jika kita ingin menggunakan fungsi bernama terpisah, sintaksnya sedikit berbeda:

```
request.onreadystatechange = ajaxCallback
function ajaxCallback()
{
if (this.readyState == 4)
{
// do something
}
}
```

Melihat Tabel 12.1, kita akan melihat bahwa readyState dapat memiliki lima nilai berbeda. Tetapi hanya satu dari mereka yang menjadi perhatian kami: nilai 4, yang mewakili panggilan Ajax yang selesai. Oleh karena itu, setiap kali fungsi baru dipanggil, ia kembali tanpa melakukan apa pun hingga readyState memiliki nilai 4. Ketika fungsi kami mendeteksi nilai itu, selanjutnya memeriksa status panggilan untuk memastikannya memiliki nilai 200, yang berarti bahwa panggilan berhasil. Jika bukan 200, pop up peringatan menampilkan pesan kesalahan yang terdapat dalam statusText.

Catatan: kita akan melihat bahwa semua properti objek ini direferensikan menggunakan this.readyState, this.status, dan seterusnya, daripada nama objek saat ini, permintaan, seperti dalam request.readyState atau request.status. Ini agar kita dapat dengan mudah menyalin dan menempelkan kode dan itu akan berfungsi dengan nama objek apa pun, karena kata kunci this selalu merujuk ke objek saat ini.

Jadi, setelah memastikan bahwa readyState adalah 4 dan statusnya adalah 200, kami menguji nilai responseText untuk melihat apakah itu berisi nilai. Jika tidak, pesan kesalahan ditampilkan di kotak peringatan. Jika tidak, HTML bagian dalam DIV diberi nilai responseText, seperti ini:

```
document.getElementById('info').innerHTML = this.responseText
```

Di baris ini, info elemen direferensikan melalui metode getElementByID, dan kemudian properti innerHTML-nya diberi nilai yang dikembalikan oleh panggilan Ajax. Setelah semua pengaturan dan persiapan ini, permintaan Ajax akhirnya dikirim ke server melalui perintah berikut, yang melewati parameter yang sudah ditentukan dalam parameter variabel:

```
request.send(params)
```

Setelah itu, semua kode sebelumnya diaktifkan setiap kali readyState berubah. Sisa dokumen adalah fungsi ajaxRequest dari Contoh 1, dan skrip penutup dan tag HTML.

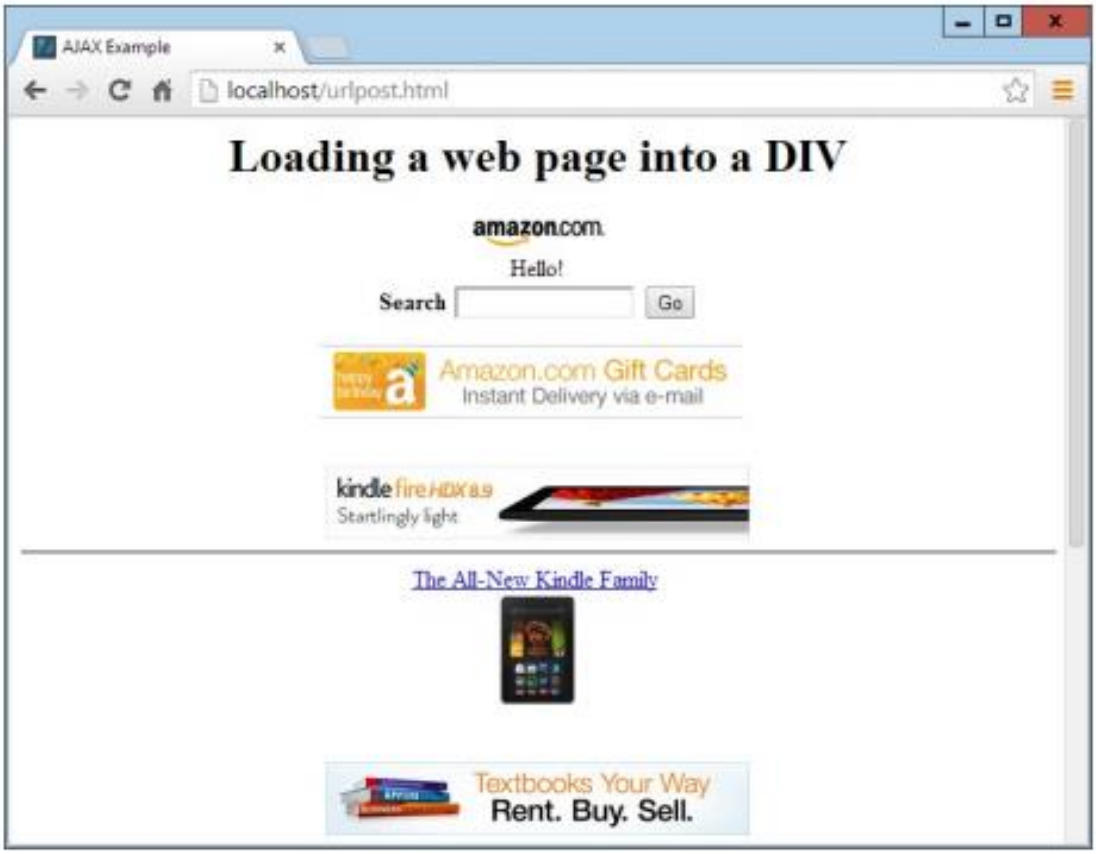
Server setengah dari proses Ajax

Sekarang kita sampai pada setengah PHP dari persamaan, yang dapat kita lihat pada Contoh 3. Ketik dan simpan sebagai urlpost.php.

Contoh 3. urlpost.php

```
<?php // urlpost.php
if (isset($_POST['url']))
{
echo file_get_contents('http://' . SanitizeString($_POST['url']));
}
function SanitizeString($var)
{
$var = strip_tags($var);
$var = htmlentities($var);
return stripslashes($var);
}
?>
```

Seperti yang kita lihat, ini singkat dan manis, dan seperti yang harus dilakukan dengan semua data yang diposting, ini juga menggunakan fungsi SanitizeString yang selalu penting. Dalam hal ini, data yang tidak bersih dapat menyebabkan pengguna mendapatkan keuntungan dari kode Anda. Program ini menggunakan fungsi PHP file_get_contents untuk memuat di halaman web pada URL yang diberikan padanya dalam variabel POST $_POST['url']. Fungsi file_get_contents serbaguna karena memuat seluruh konten file atau halaman web dari server lokal atau jauh; bahkan memperhitungkan halaman yang dipindahkan dan pengalihan lainnya. Setelah kita mengetik program, kita siap untuk memanggil urlpost.html ke browser web kita dan, setelah beberapa detik, kita akan melihat konten halaman depan ponsel Amazon dimuat ke DIV yang kami buat untuk tujuan itu. Ini tidak akan secepat langsung memuat halaman web, karena ditransfer dua kali: sekali ke server dan lagi dari server ke browser Anda. Hasilnya akan terlihat seperti Gambar 12.1. Kami tidak hanya berhasil membuat panggilan Ajax dan mendapatkan respons yang dikembalikan ke JavaScript, kami juga memanfaatkan kekuatan PHP untuk memungkinkan penggabungan objek web yang sama sekali tidak terkait. Kebetulan, jika kami mencoba menemukan cara untuk mengambil halaman web seluler Amazon secara langsung melalui Ajax (tanpa bantuan modul sisi server PHP), kami tidak akan berhasil, karena ada blok keamanan yang mencegah Ajax lintas-domain. Jadi contoh kecil ini juga menggambarkan solusi praktis untuk masalah yang sangat praktis.



Gambar 12.1 Situs web seluler Amazon telah dimuat ke dalam DIV

12.4 MENGGUNAKAN GET ALIH-ALIH POST

Seperti halnya mengirimkan data formulir apa pun, kita memiliki opsi untuk mengirimkan data kita dalam bentuk permintaan GET, dan kita akan menyimpan beberapa baris kode jika kita melakukannya. Namun, ada kelemahannya: beberapa browser mungkin menyimpan permintaan GET, sedangkan permintaan POST tidak akan pernah di-cache. kita

tidak ingin men-cache permintaan, karena browser hanya akan menampilkan ulang apa yang didapat terakhir kali alih-alih pergi ke server untuk input baru. Solusi untuk ini adalah dengan menggunakan solusi yang menambahkan parameter acak untuk setiap permintaan, memastikan bahwa setiap URL yang diminta adalah unik. Contoh 4 menunjukkan bagaimana kita akan mencapai hasil yang sama seperti dengan Contoh 2, tetapi menggunakan permintaan GET Ajax alih-alih POST.

Contoh 4. urlget.html

```
<!DOCTYPE html>
<html>
<head>
<title>AJAX Example</title>
</head>
<body style='text-align:center'>
<h1>Loading a web page into a DIV</h1>
<div id='info'>This sentence will be replaced</div>
<script>
nocache = "&nocache=" + Math.random() * 1000000
request = new ajaxRequest()
request.open("GET", "urlget.php?url=amazon.com/gp/aw" + nocache, true)
request.onreadystatechange = function()
{
if (this.readyState == 4)
{
if (this.status == 200)
{
if (this.responseText != null)
{
document.getElementById('info').innerHTML =
this.responseText
}
else alert("Ajax error: No data received")
}
else alert( "Ajax error: " + this.statusText)
}
}
request.send(null)
function ajaxRequest()
{
try
{
var request = new XMLHttpRequest()
}
catch(e1)
{
try
```

```
{
request = new ActiveXObject("Msxml2.XMLHTTP")
}
catch(e2)
{
try
{
request = new ActiveXObject("Microsoft.XMLHTTP")
}
catch(e3)
{
request = false
}
}
}
return request
}
</script>
</body>
</html>
```

Perbedaan yang perlu diperhatikan antara kedua dokumen disorot dalam huruf tebal, dan dijelaskan sebagai berikut: Tidak perlu mengirim header untuk permintaan GET. Kami memanggil metode terbuka menggunakan permintaan GET, menyediakan URL dengan string yang terdiri dari ? simbol diikuti oleh pasangan parameter/nilai url=amazon.com/gp/aw. Kami memulai pasangan parameter/nilai kedua menggunakan simbol &, kemudian menetapkan nilai parameter nocache ke nilai acak antara 0 dan satu juta. Ini digunakan untuk memastikan bahwa setiap URL yang diminta berbeda, dan oleh karena itu tidak ada permintaan yang akan di-cache.

Panggilan untuk mengirim sekarang hanya berisi parameter null, karena tidak ada parameter yang diteruskan melalui permintaan POST. Perhatikan bahwa membiarkan parameter keluar bukanlah pilihan, karena akan mengakibatkan kesalahan.

Untuk melengkapi dokumen baru ini, program PHP harus dimodifikasi untuk menanggapi permintaan GET, seperti pada Contoh 5, urlget.php

Contoh 5. urlget.php

```
<?php
if (isset($_GET['url']))
{
echo file_get_contents("http://".sanitizeString($_GET['url']));
}
function sanitizeString($var)
{
$var = strip_tags($var);
$var = htmlentities($var);
return stripslashes($var);
```

```
}
?>
```

Yang membedakan antara ini dan Contoh 3 adalah bahwa referensi ke $_POST telah diganti dengan $_GET. Hasil akhir dari pemanggilan urlget.html di browser kita sama dengan loading di urlpost.html.

Mengirim Permintaan XML

Meskipun objek yang telah kita buat disebut objek XMLHttpRequest, sejauh ini kita sama sekali tidak menggunakan XML. Di sinilah istilah Ajax sedikit keliru, karena teknologi sebenarnya memungkinkan kita untuk meminta semua jenis data tekstual, hanya salah satunya adalah XML. Seperti yang kita lihat, kami telah meminta seluruh dokumen HTML melalui Ajax, tetapi kami juga dapat meminta halaman teks, string atau angka, atau bahkan data spreadsheet. Jadi mari kita ubah dokumen contoh sebelumnya dan program PHP untuk mengambil beberapa data XML. Untuk melakukannya, pertama-tama lihat program PHP, xmlget.php, yang ditunjukkan pada Contoh 6.

Contoh 6 xmlget.php

```
<?php
if (isset($_GET['url']))
{
header('Content-Type: text/xml');
echo file_get_contents("http://".sanitizeString($_GET['url']));
}
function sanitizeString($var)
{
$var = strip_tags($var);
$var = htmlentities($var);
return stripslashes($var);
}
?>
```

Program ini telah sedikit dimodifikasi (ditampilkan dalam penyorotan tebal) untuk menampilkan header XML yang benar sebelum mengembalikan dokumen yang diambil. Tidak ada pemeriksaan yang dilakukan di sini, karena diasumsikan bahwa pemanggilan Ajax akan meminta dokumen XML yang sebenarnya. Sekarang ke dokumen HTML, xmlget.html, yang ditunjukkan pada Contoh 7.

Contoh 7. xmlget.html

```
<!DOCTYPE html>
<html>
<head>
<title>AJAX Example</title>
</head>
<body>
<h1>Loading a web page into a DIV</h1>
<div id='info'>This sentence will be replaced</div>
```

```
<script>
nocache = "&nocache=" + Math.random() * 1000000
url = "rss.news.yahoo.com/rss/topstories"
out = "";
request = new ajaxRequest()
request.open("GET", "xmlget.php?url=" + url + nocache, true)
request.onreadystatechange = function()
{
if (this.readyState == 4)
{
if (this.status == 200)
{
if (this.responseText != null
{
titles = this.responseXML.getElementsByTagName('title')
for (j = 0 ; j < titles.length ; ++j)
{
out += titles[j].childNodes[0].nodeValue + '<br>'
}
document.getElementById('info').innerHTML = out
}
else alert("Ajax error: No data received")
}
else alert( "Ajax error: " + this.statusText)
}
}
request.send(null)
function ajaxRequest()
{
try
{
var request = new XMLHttpRequest()
}
catch(e1)
{
try
{
request = new ActiveXObject("Msxml2.XMLHTTP")
}
catch(e2)
{
try
{
request = new ActiveXObject("Microsoft.XMLHTTP")
}
catch(e3)
```

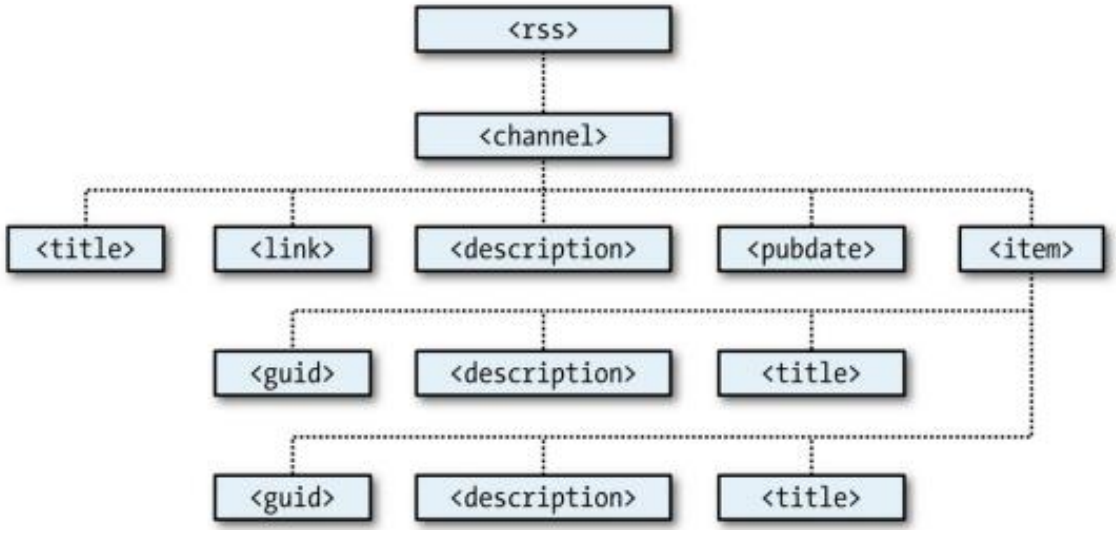
```
{
request = false
}
}
}
return request
}
</script>
</body>
</html>
```

Sekali lagi, perubahan telah disorot dalam huruf tebal, sehingga kita dapat melihat bahwa kode ini secara substansial mirip dengan versi sebelumnya, kecuali bahwa URL yang sekarang diminta, rss.news.yahoo.com/rss/topstories, berisi dokumen XML, Yahoo! Umpan Berita Teratas.

Perbedaan besar lainnya adalah penggunaan properti responseXML, yang menggantikan properti responseText. Setiap kali server mengembalikan data XML, responseXML akan berisi XML yang dikembalikan. Namun, responseXML tidak hanya berisi string teks XML: sebenarnya ini adalah objek dokumen XML lengkap yang dapat kita periksa dan urai menggunakan metode dan properti pohon DOM. Ini berarti dapat diakses, misalnya, dengan metode JavaScript getElementsByTagName.

12.5 TENTANG XML

Sebuah dokumen XML umumnya akan mengambil bentuk RSS feed yang ditunjukkan pada Contoh 18. Namun, keindahan XML adalah kita dapat menyimpan jenis struktur ini secara internal di pohon DOM (lihat Gambar 12.2) agar dapat dicari dengan cepat.



Gambar 12.2 Pohon DOM dari Contoh 18

Contoh 18. Sebuah dokumen XML

```
<?xml version="1.0" encoding="UTF-8"?>
<rss version="2.0">
<channel>
```

```
<title>RSS Feed</title>
<link>http://website.com</link>
<description>website.com's RSS Feed</description>
<pubDate>Mon, 11 May 2020 00:00:00 GMT</pubDate>
<item>
<title>Headline</title>
<guid>http://website.com/headline</guid>
<description>This is a headline</description>
</item>
<item>
<title>Headline 2</title>
<guid>http://website.com/headline2</guid>
<description>The 2nd headline</description>
</item>
</channel>
</rss>
```

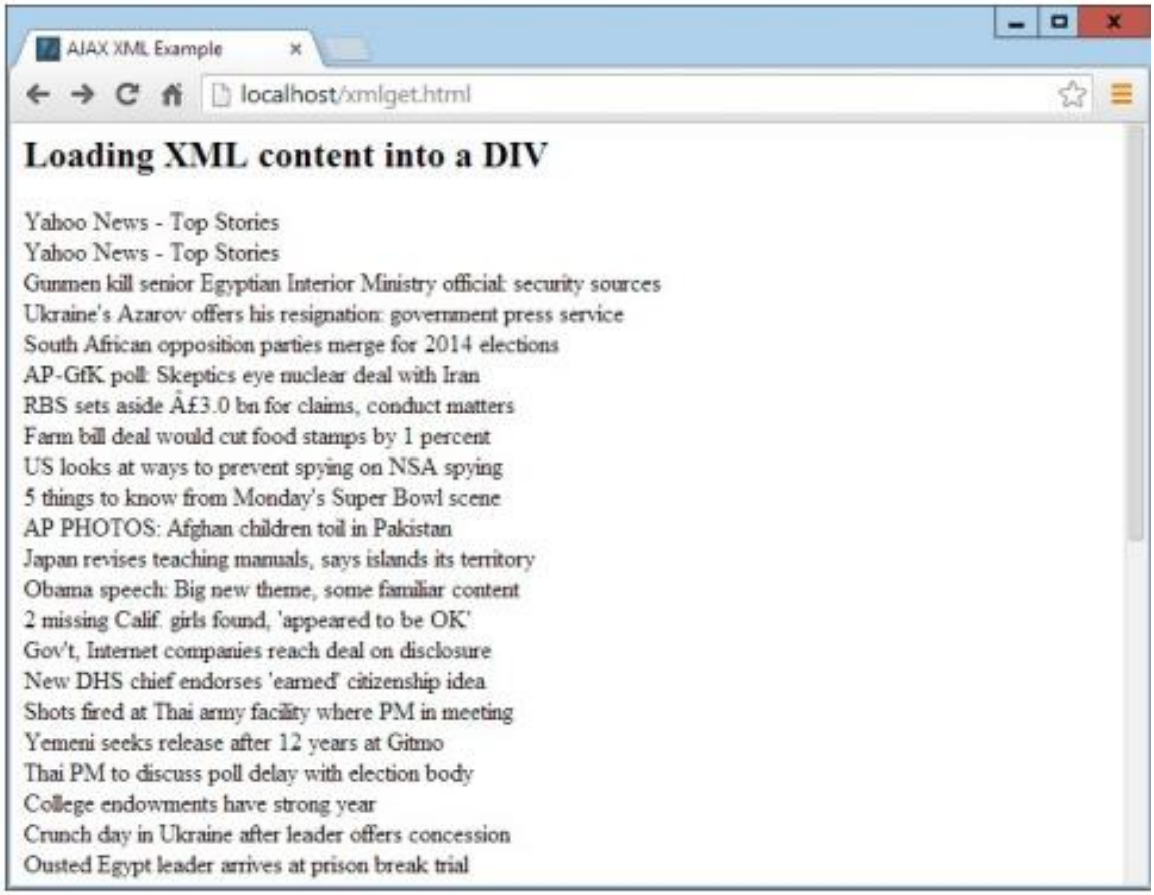
Oleh karena itu, dengan menggunakan metode getElementsByTagName, kita dapat dengan cepat mengekstrak nilai yang terkait dengan berbagai tag tanpa banyak pencarian string. Ini persis seperti yang kita lakukan pada Contoh 7, di mana perintah berikut dikeluarkan: judul = this.responseXML.getElementsByTagName('title') Perintah tunggal ini memiliki efek menempatkan semua nilai elemen judul ke dalam judul larik . Dari sana, mudah untuk mengekstraknya dengan ekspresi berikut (di mana j adalah judul untuk diakses)

```
titles[j].childNodes[0].nodeValue
```

Semua judul kemudian ditambahkan ke variabel string keluar dan, setelah semua diproses, hasilnya dimasukkan ke dalam DIV kosong di awal dokumen. Saat kita memanggil xmlget.html di browser Anda, hasilnya akan seperti Gambar 12.3.

Mengapa menggunakan XML?

Anda mungkin bertanya mengapa kita menggunakan XML selain untuk mengambil dokumen XML seperti umpan RSS. Jawaban sederhananya adalah kita tidak harus melakukannya, tetapi jika kita ingin meneruskan data terstruktur kembali ke aplikasi Ajax Anda, akan sangat merepotkan untuk mengirim teks yang sederhana dan tidak terorganisir yang memerlukan pemrosesan rumit dalam JavaScript. . Sebagai gantinya, kita dapat membuat dokumen XML dan meneruskannya kembali ke fungsi Ajax, yang secara otomatis akan menempatkannya ke dalam pohon DOM, semudah diakses seperti objek DOM HTML yang sekarang kita kenal.



Gambar 12.3 Mengambil Yahoo! Umpan berita XML melalui Ajax

12.6 MENGGUNAKAN KERANGKA KERJA UNTUK AJAX

Sekarang setelah kita mengetahui cara membuat kode rutin Ajax kita sendiri, kita mungkin ingin menyelidiki beberapa kerangka kerja gratis yang tersedia untuk membuatnya lebih mudah, karena mereka menawarkan lebih banyak fitur lanjutan. Secara khusus, saya sarankan kita memeriksa jQuery, yang mungkin merupakan kerangka kerja yang paling umum digunakan. kita dapat mengunduhnya (dan mendapatkan dokumentasi lengkap) dari http://jquery.com, tetapi perlu diketahui bahwa pada awalnya ada kurva belajar yang curam, karena kita harus membiasakan diri dengan fungsi $ yang disediakannya, yang digunakan secara luas untuk mengakses jQuery's fitur. Yang mengatakan, setelah kita memahami cara kerja jQuery, kita akan merasa bahwa itu dapat membuat pengembangan web kita lebih mudah dan lebih cepat karena banyaknya fitur siap pakai yang ditawarkannya.

BAB 13
MENGAKSES CSS DARI JAVASCRIPT

Dengan pemahaman yang baik tentang DOM dan CSS sekarang di bawah ikat pinggang Anda, kita akan belajar di bab ini cara mengakses DOM dan CSS langsung dari JavaScript, memungkinkan kita membuat situs web yang sangat dinamis dan responsif. Saya juga akan menunjukkan cara menggunakan interupsi sehingga kita dapat membuat animasi atau memberikan kode apa pun yang harus terus berjalan (seperti jam). Terakhir, saya akan menjelaskan bagaimana kita dapat menambahkan elemen baru atau menghapus elemen yang sudah ada dari DOM sehingga kita tidak perlu membuat elemen sebelumnya dalam HTML untuk berjaga-jaga jika JavaScript mungkin perlu mengaksesnya nanti.

13.1 MENINJAU KEMBALI FUNGSI GETELEMENTBYID

Dalam Bab 8, saya menyebutkan penggunaan umum karakter $ sebagai nama fungsi untuk memberikan akses yang lebih mudah ke fungsi getElementById. Faktanya, kerangka kerja utama seperti jQuery menggunakan fungsi $ baru ini, dan secara substansial memperluas fungsinya juga.

Saya juga ingin memberi kita versi yang disempurnakan dari fungsi ini, sehingga kita dapat menangani elemen DOM dan gaya CSS dengan cepat dan efisien. Namun, untuk menghindari konflik dengan framework yang menggunakan karakter $, saya cukup menggunakan huruf besar O, karena itu adalah huruf pertama dari kata Object, yang akan dikembalikan ketika fungsi dipanggil (objek yang diwakili oleh ID diteruskan ke fungsi).

fungsi O

Inilah yang terlihat seperti fungsi O tanpa tulang:

```
function O(obj)
{
return document.getElementById(obj)
}
```

Ini saja menghemat 22 karakter pengetikan setiap kali dipanggil. Tetapi saya memilih untuk memperluas fungsi sedikit dengan mengizinkan nama ID atau objek untuk diteruskan ke fungsi ini, seperti yang ditunjukkan dalam versi lengkap fungsi di Contoh 1.

Contoh 1. Fungsi O()

```
function O(obj)
{
if (typeof obj == 'object') return obj
else return document.getElementById(obj)
}
```

Jika suatu objek diteruskan ke fungsi, itu hanya mengembalikan objek itu kembali. Jika tidak, ia mengasumsikan bahwa ID dilewatkan dan mengembalikan objek yang dirujuk ID.

Tetapi mengapa saya ingin menambahkan pernyataan pertama ini, yang hanya mengembalikan objek yang diteruskan ke sana?

Fungsi S

Jawaban atas pertanyaan ini menjadi jelas ketika kita melihat fungsi mitra yang disebut S, yang memberi kita akses mudah ke properti gaya (atau CSS) suatu objek, seperti yang ditunjukkan pada Contoh 2.

Contoh 2. Fungsi S()

```
function S(obj)
{
return O(obj).style
}
```

S dalam nama fungsi ini adalah huruf pertama dari Gaya, dan fungsi melakukan tugas mengembalikan properti gaya (atau subobjek) dari elemen yang dirujuk. Karena fungsi O yang disematkan menerima ID atau objek, kita juga dapat meneruskan ID atau objek ke S. Mari kita lihat apa yang terjadi di sini dengan mengambil elemen dengan ID myobj dan mengatur warna teksnya menjadi hijau, seperti ini:

```
<div id='myobj'>Some text</div>
<script>
O('myobj').style.color = 'green'
</script>
```

Kode sebelumnya akan melakukan pekerjaan itu, tetapi jauh lebih mudah untuk memanggil fungsi S baru, seperti ini:

```
S('myobj').color = 'green
```

Sekarang perhatikan kasus di mana objek yang dikembalikan dengan memanggil O disimpan di, misalnya, objek bernama fred, seperti ini:

```
fred = O('myobj')
```

Karena cara kerja fungsi S, kita masih bisa memanggilnya untuk mengubah warna teks menjadi hijau, seperti ini:

```
S(fred).color = 'green'
```

Ini berarti apakah kita ingin mengakses objek secara langsung atau melalui ID-nya, kita dapat melakukannya dengan meneruskannya ke fungsi O atau S sesuai kebutuhan. Ingatlah bahwa ketika kita melewati sebuah objek (bukan ID), kita tidak boleh menempatkannya dalam tanda kutip.

Fungsi C

Sejauh ini saya telah memberi kita dua fungsi sederhana yang memudahkan kita mengakses elemen apa pun di halaman web, dan properti gaya apa pun dari suatu elemen.

Namun, terkadang kita ingin mengakses lebih dari satu elemen sekaligus, dan kita dapat melakukannya dengan menetapkan nama kelas CSS untuk setiap elemen tersebut, seperti contoh berikut, yang keduanya menggunakan kelas myclass:

```
<div class='myclass'>Div contents</fiv>
<p class='myclass'>Paragraph contents</p>
```

Jika kita ingin mengakses semua elemen pada halaman yang menggunakan kelas tertentu, kita dapat menggunakan fungsi C (untuk huruf pertama Kelas), yang ditunjukkan pada Contoh 3, untuk mengembalikan array yang berisi semua objek yang cocok dengan kelas nama yang disediakan.

Contoh 3 Fungsi C()

```
function C(name)
{
var elements = document.getElementsByTagName('*')
var objects = []
for (var i = 0 ; i < elements.length ; ++i)
if (elements[i].className == name)
objects.push(elements[i])
return objects
}
```

Mari kita uraikan contoh ini. Pertama, nama argumen berisi nama kelas yang kita coba ambil objeknya. Kemudian, di dalam fungsi, yang baru objek yang disebut elemen dibuat yang berisi semua elemen dalam dokumen, seperti yang dikembalikan oleh panggilan ke getElementsByTagName dengan argumen '*', yang berarti "temukan semua elemen":

```
var elements = document.getElementsByTagName('*')
```

Kemudian array baru yang disebut objek dibuat, di mana semua objek yang cocok ditemukan akan ditempatkan:

```
var objects = []
```

Selanjutnya, for loop iterasi melalui semua elemen dalam objek elemen menggunakan variabel i sebagai indeks:

```
for (var i = 0 ; i < elements.length ; ++i)
```

Setiap kali di sekitar loop, jika properti className elemen sama dengan nilai string yang diteruskan dalam nama argumen, objek didorong ke array objek:

```
if (elements[i].className == name)
objects.push(elements[i])
```

Akhirnya, setelah loop selesai, array objek akan berisi semua elemen dalam dokumen yang menggunakan nama kelas dalam nama, sehingga dikembalikan oleh fungsi:

return objects

Untuk menggunakan fungsi ini, panggil saja sebagai berikut, simpan array yang dikembalikan sehingga kita dapat mengakses setiap elemen satu per satu sesuai kebutuhan atau (lebih mungkin demikian) secara massal melalui satu lingkaran:

myarray = C('myclass')

Sekarang kita dapat melakukan apa pun yang kita suka dengan objek yang dikembalikan, seperti, misalnya, menyetel properti gaya textDecoration ke 'garis bawah', sebagai berikut:

for (i = 0 ; i < myarray.length ; ++i)
S(myarray[i]).textDecoration = 'underline

Kode ini berulang melalui objek di myarray[] dan kemudian menggunakan fungsi S untuk mereferensikan masing-masing properti gaya, menyetel properti textDecoration ke 'garis bawah'.

13.2 MEMASUKKAN FUNGSI

Saya menggunakan fungsi O dan S dalam contoh untuk sisa bab ini, karena mereka membuat kode lebih pendek dan lebih mudah diikuti. Oleh karena itu, saya telah menyimpannya dalam file OSC.js (bersama dengan fungsi C, karena saya pikir kita akan merasa sangat berguna) dapat diunduh secara bebas dari situs web pendamping. kita dapat menyertakan fungsi-fungsi ini di halaman web mana pun menggunakan pernyataan berikut — sebaiknya di bagian <head>-nya, di mana pun sebelum skrip apa pun yang bergantung pada pemanggilannya:

<script src='OSC.js'></script>

Isi OSC.js ditunjukkan pada Contoh 4.

Contoh 4. File OSC.js

```
function O(obj)
{
if (typeof obj == 'object') return obj
else return document.getElementById(obj)
}
function S(obj)
{
return O(obj).style
}
function C(name)
{
```

```
var elements = document.getElementsByTagName('*')
var objects = []
for (var i = 0 ; i < elements.length ; ++i)
if (elements[i].className == name)
objects.push(elements[i])
return objects
}
```

13.3 MENGAKSES PROPERTI CSS DARI JAVASCRIPT

Properti textDecoration yang saya gunakan dalam contoh sebelumnya mewakili properti CSS yang biasanya diberi tanda hubung seperti ini: text-decoration. Tetapi karena JavaScript menyimpan karakter tanda hubung untuk digunakan sebagai operator matematika, setiap kali kita mengakses properti CSS yang diberi tanda hubung, kita harus menghilangkan tanda hubung dan mengatur karakter segera setelahnya menjadi huruf besar. Contoh lain dari ini adalah properti font-size, yang direferensikan dalam JavaScript sebagai fontSize ketika ditempatkan setelah operator titik, seperti ini:

```
myobject.fontSize = '16pt'
```

Alternatif untuk ini adalah menjadi lebih bertele-tele dan menggunakan fungsi setAttribute, yang mendukung (dan sebenarnya membutuhkan) nama properti CSS standar, seperti ini:

```
myobject.setAttribute('style', 'font-size:16pt')
```

Perhatikan bahwa, beberapa versi Microsoft Internet Explorer yang lebih lama pilih-pilih dalam kasus tertentu tentang penggunaan nama properti CSS gaya JavaScript saat menerapkan versi aturan yang diawali -ms- khusus browser. Jika kita mengalami ini, gunakan fungsi setAttribute dan kita akan baik-baik saja.

Beberapa Properti Umum

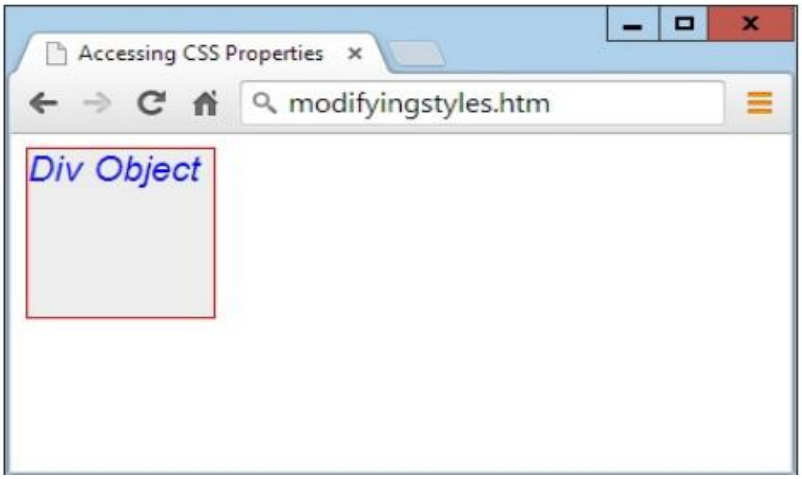
Menggunakan JavaScript, kita dapat memodifikasi properti apa pun dari elemen apa pun dalam dokumen web, dengan cara yang mirip dengan menggunakan CSS. Saya telah menunjukkan kepada kita cara mengakses properti CSS menggunakan bentuk pendek JavaScript atau fungsi setAttribute untuk menggunakan nama properti CSS yang tepat, jadi saya tidak akan membuat kita bosan dengan merinci semua ratusan properti ini. Sebaliknya, saya ingin menunjukkan kepada kita cara mengakses hanya beberapa properti CSS sebagai ikhtisar dari beberapa hal yang dapat kita lakukan. Pertama, mari kita lihat memodifikasi beberapa properti CSS dari JavaScript menggunakan Contoh 5, yang memuat tiga fungsi sebelumnya, membuat elemen , dan kemudian mengeluarkan pernyataan JavaScript dalam bagian <script> HTML, untuk memodifikasi berbagai atributnya (lihat Gambar 13.1).

Contoh 5. Mengakses properti CSS dari JavaScript

```
<!DOCTYPE html>
<html>
<head>
<title>Accessing CSS Properties</title>
<script src='OSC.js'></script>
```

```
</head>
<body>
<div id='object'>Div Object</div>
<script>
S('object').border = 'solid 1px red'
S('object').width = '100px'
S('object').height = '100px'
S('object').background = '#eee'
S('object').color = 'blue'
S('object').fontSize = '15pt'
S('object').fontFamily = 'Helvetica'
S('object').fontStyle = 'italic'
</script>
</body>
</html>
```

Anda tidak mendapatkan apa-apa dengan memodifikasi properti seperti ini, karena kita dapat dengan mudah memasukkan beberapa CSS secara langsung, tetapi segera kami akan memodifikasi properti sebagai respons terhadap interaksi pengguna—dan kemudian kita akan melihat kekuatan sebenarnya dari menggabungkan JavaScript dan CSS.



Gambar 13.1 Memodifikasi gaya dari JavaScript

Properti lainnya

JavaScript juga membuka akses ke berbagai properti lain yang sangat luas, seperti lebar dan tinggi browser dan jendela atau bingkai pop-up atau dalam browser, informasi berguna seperti jendela induk (jika ada) , dan riwayat URL yang mengunjungi sesi ini. Semua properti ini diakses dari objek jendela melalui operator periode (misalnya, nama jendela), dan Tabel 13.1 mencantumkan semuanya, bersama dengan deskripsi masing-masing.

Tabel 13.1 Properti jendela umum

Properti	**Set dan/atau kembali**
closed	Mengembalikan nilai Boolean yang menunjukkan apakah jendela telah ditutup atau tidak

closed	Menyetel atau mengembalikan teks default di bilah status jendela
document	Mengembalikan objek dokumen untuk jendela
frames	Mengembalikan array dari semua frame dan iframe di jendela
history	Mengembalikan objek sejarah untuk jendela
innerHeight	Mengatur atau mengembalikan ketinggian bagian dalam area konten jendela
innerWidth	Mengatur atau mengembalikan lebar bagian dalam area konten jendela
length	Mengembalikan jumlah bingkai dan iframe di jendela
location	Mengembalikan objek lokasi untuk jendela
name	Menetapkan atau mengembalikan nama jendela
navigator	Mengembalikan objek navigator untuk jendela
opener	Mengembalikan referensi ke jendela yang membuat jendela
outerHeight	Mengatur atau mengembalikan ketinggian luar jendela, termasuk alat dan bilah gulir
outerWidth	Mengatur atau mengembalikan lebar luar jendela, termasuk alat dan bilah gulir
pageXOffset	Mengembalikan piksel dokumen yang telah digulir secara horizontal dari kiri jendela
pageYOffset	Mengembalikan piksel dokumen yang telah digulir secara vertikal dari atas jendela
parent	Mengembalikan jendela induk dari sebuah jendela
screen	Mengembalikan objek layar untuk jendela
screenLeft	Mengembalikan koordinat x dari jendela relatif terhadap layar di semua browser terbaru kecuali Mozilla Firefox (untuk itu kita harus menggunakan screenX)
screenTop	Mengembalikan koordinat y dari jendela relatif terhadap layar di semua browser terbaru kecuali Mozilla Firefox (untuk itu kita harus menggunakan screenY)
screenX	Mengembalikan koordinat x jendela relatif terhadap layar di semua browser terbaru kecuali Opera, yang mengembalikan nilai yang salah; tidak didukung di versi IE sebelum 9
screenY	Mengembalikan koordinat y dari jendela relatif terhadap layar di semua browser terbaru kecuali Opera, yang mengembalikan nilai yang salah; tidak didukung di versi IE sebelum 9
self	Mengembalikan jendela saat ini
status	Menyetel atau mengembalikan teks di bilah status jendela
top	Mengembalikan jendela browser teratas

Ada beberapa poin yang perlu diperhatikan tentang beberapa properti ini:

- Status default dan properti status dapat disetel hanya jika pengguna telah memodifikasi browser mereka untuk mengizinkannya (sangat tidak mungkin).
- Objek sejarah tidak dapat dibaca (sehingga kita tidak dapat melihat dari mana pengunjung kita berselancar). Tapi itu mendukung properti length untuk menentukan berapa lama sejarah, dan metode mundur, maju, dan pergi untuk menavigasi ke halaman tertentu dalam sejarah.

- Saat kita perlu mengetahui berapa banyak ruang yang tersedia di jendela browser web saat ini, cukup baca nilai di window.innerHeight dan window.innerWidth. Saya sering menggunakan nilai-nilai ini untuk memusatkan peringatan pop-up dalam browser atau jendela "konfirmasi dialog".
- Objek layar mendukung properti baca availHeight, availWidth, colorDepth, height, pixelDepth, dan lebar, dan oleh karena itu bagus untuk menentukan informasi tentang tampilan pengguna.

Beberapa item informasi ini akan membantu kita memulai dan sudah memberi kita banyak hal baru dan menarik yang dapat kita lakukan dengan JavaScript. Namun, pada kenyataannya, ada jauh lebih banyak properti dan metode yang tersedia daripada yang dapat dibahas dalam bab ini. Namun, sekarang setelah kita mengetahui cara mengakses dan menggunakan properti, yang kita butuhkan hanyalah sumber daya yang mencantumkan semuanya, jadi saya sarankan kita memeriksa http://tinyurl.com/domproperties sebagai titik awal yang baik.

13.4 JAVASCRIPT SEBARIS

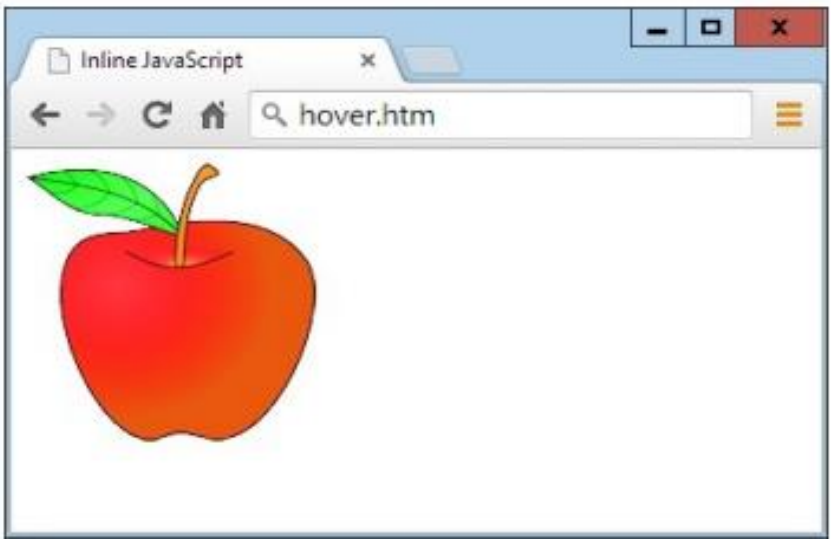
Menggunakan tag <script> bukan satu-satunya cara kita dapat mengeksekusi pernyataan JavaScript; kita juga dapat mengakses JavaScript dari dalam tag HTML, yang menghasilkan interaktivitas dinamis yang hebat. Misalnya, untuk menambahkan efek cepat saat mouse melewati suatu objek, kita dapat menggunakan kode seperti pada tag <img> pada Contoh 6, yang menampilkan apel secara default, tetapi menggantinya dengan oranye saat mouse melewati, dan mengembalikan apel lagi ketika mouse pergi.

Contoh 6. Menggunakan JavaScript sebaris

```
<!DOCTYPE html>
<html>
<head>
<title>Inline JavaScript</title>
</head>
<body>
<img src='apple.png'
onmouseover="this.src='orange.png'"
onmouseout="this.src='apple.png'">
</body>
</html>
```

Kata Kunci This

Pada contoh sebelumnya, kita melihat kata kunci this sedang digunakan. Ini memberitahu JavaScript untuk beroperasi pada objek panggilan, yaitu tag <img>. kita dapat melihat hasilnya pada Gambar 13.2, di mana mouse belum melewati apel



Gambar 13.2 Contoh JavaScript hover mouse sebaris

13.5 MELAMPIRKAN ACARA KE OBJEK DALAM SCRIPT

Kode sebelumnya sama dengan memberikan ID ke tag <img>, lalu melampirkan tindakan ke peristiwa mouse tag, seperti Contoh 7.

Contoh 7 JavaScript non-sebaris

```
<!DOCTYPE html>
<html>
<head>
<title>Non-inline JavaScript</title>
<script src='OSC.js'></script>
</head>
<body>
<img id='object' src='apple.png'>
<script>
O('object').onmouseover = function() { this.src = 'orange.png' }
O('object').onmouseout = function() { this.src = 'apple.png' }
</script>
</body>
</html>
```

Di bagian HTML, contoh ini memberikan elemen <img> sebuah ID objek, kemudian mulai memanipulasinya secara terpisah di bagian JavaScript, dengan melampirkan fungsi anonim ke setiap peristiwa.

Melampirkan ke Acara Lain

Baik kita menggunakan JavaScript sebaris atau terpisah, ada beberapa peristiwa yang dapat kita lampirkan tindakannya, memberikan banyak fitur tambahan yang dapat kita tawarkan kepada pengguna Anda. Tabel 13.2 mencantumkan peristiwa ini dan detailnya kapan akan dipicu.

Tabel 13.2 Peristiwa dan kapan dipicu

onabort	Saat pemuatan gambar dihentikan sebelum selesai

onblur	Ketika sebuah elemen kehilangan fokus
onchange	Ketika ada bagian dari formulir yang berubah
onclick	Ketika suatu objek diklik
ondblclick	Ketika sebuah objek diklik dua kali
onerror	Ketika kesalahan JavaScript ditemukan
onfocus	Ketika sebuah elemen mendapat fokus
onkeydown	Saat tombol ditekan (termasuk Shift, Alt, Ctrl, dan Esc)
onkeypress	Saat tombol ditekan (tidak termasuk Shift, Alt, Ctrl, dan Esc)
onkeyup	Saat kunci dilepaskan
onload	Ketika sebuah objek telah dimuat
onmousedown	Saat tombol mouse ditekan di atas elemen
onmousemove	Saat mouse digerakkan di atas elemen
onmouseout	Saat mouse meninggalkan elemen
onmouseover	Saat mouse melewati elemen dari luarnya
onmouseup	Saat tombol mouse dilepaskan
onsubmit	Saat formulir dikirimkan
onreset	Saat formulir diatur ulang
onresize	Saat browser diubah ukurannya
onscroll	Saat dokumen digulir
onselect	Ketika beberapa teks dipilih
onunload	Saat dokumen dihapus

13.6 MENAMBAHKAN ELEMEN BARU

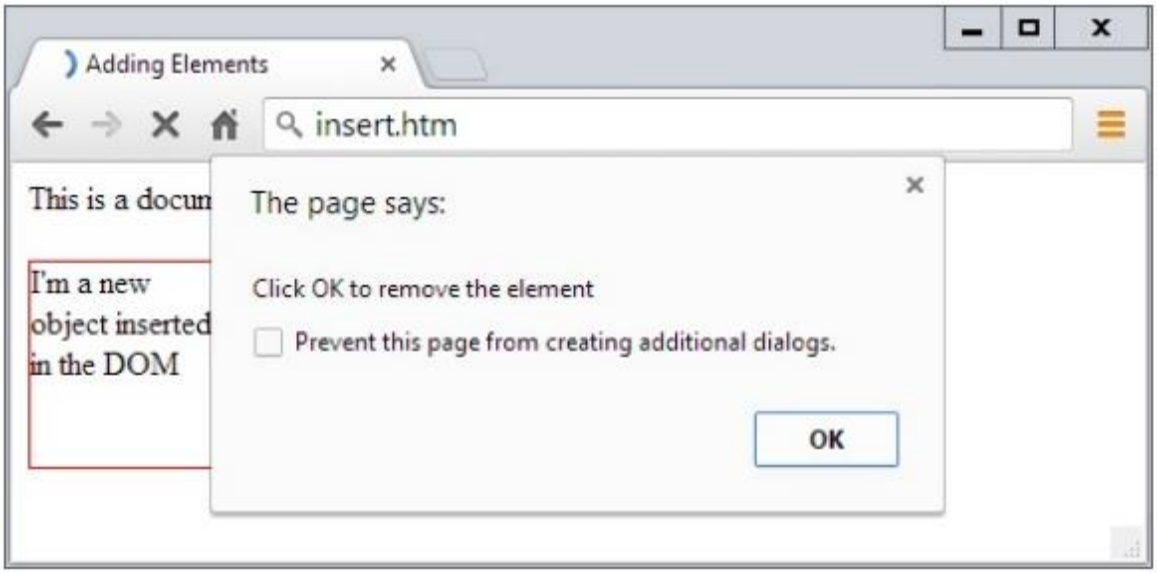
Dengan JavaScript kita tidak dibatasi untuk memanipulasi elemen dan objek yang diberikan ke dokumen dalam HTML-nya. Bahkan, kita dapat membuat objek sesuka hati dengan memasukkannya ke dalam DOM. Misalnya, kita memerlukan elemen baru. Contoh 8 menunjukkan satu cara kita dapat menambahkannya ke halaman web.

Contoh 8 Memasukkan elemen ke dalam DOM

```
<!DOCTYPE html>
<html>
<head>
<title>Adding Elements</title>
<script src='OSC.js'></script>
</head>
<body>
This is a document with only this text in it.<br><br>
<script>
alert('Click OK to add an element')
newdiv = document.createElement('div')
newdiv.id = 'NewDiv'
document.body.appendChild(newdiv)
S(newdiv).border = 'solid 1px red'
S(newdiv).width = '100px'
S(newdiv).height = '100px'
newdiv.innerHTML = "I'm a new object inserted in the DOM"
tmp = newdiv.offsetTop
```

```
alert('Click OK to remove the element')
pnode = newdiv.parentNode
pnode.removeChild(newdiv)
tmp = pnode.offsetTop
</script>
</body>
</html>
```

Gambar 13.3 menunjukkan kode diatas yang digunakan untuk menambahkan elemen baru ke dokumen web. Pertama, elemen baru dibuat dengan createElement, kemudian fungsi appendChild dipanggil dan elemen dimasukkan ke dalam DOM.



Gambar 13.3 Memasukkan elemen baru ke dalam DOM

Setelah ini, berbagai properti ditetapkan ke elemen, termasuk beberapa teks untuk HTML bagian dalamnya. Dan kemudian, untuk memastikan elemen baru langsung terungkap, properti offsetTop-nya dibacakan ke dalam variabel sekali pakai tmp. Ini memaksa penyegaran DOM dan membuat elemen ditampilkan di browser apa pun yang mungkin tertunda sebelum melakukannya—khususnya Internet Explorer. Elemen baru ini persis sama seperti jika telah disertakan dalam HTML asli, dan memiliki semua properti dan metode yang sama yang tersedia.

Menghapus Elemen

Anda juga dapat menghapus elemen dari DOM, termasuk yang tidak kita sisipkan menggunakan JavaScript; bahkan lebih mudah daripada menambahkan elemen. Ini berfungsi seperti ini, dengan asumsi elemen yang akan dihapus ada di elemen objek:

```
element.parentNode.removeChild(element)
```

Kode ini mengakses objek parentNode elemen sehingga dapat menghapus elemen dari node tersebut. Kemudian ia memanggil metode removeChild pada objek itu, meneruskan objek yang akan dihapus. Namun, untuk memastikan DOM langsung disegarkan di semua browser, kita mungkin lebih suka mengganti pernyataan tunggal sebelumnya dengan sesuatu seperti berikut:

```
pnode = element.parentNode
pnode.removeChild(element)
tmp = pnode.offsetTop
```

Alternatif untuk Menambah dan Menghapus Elemen

Memasukkan elemen dimaksudkan untuk menambahkan objek yang sama sekali baru ke dalam halaman web. Tetapi jika semua yang ingin kita lakukan adalah menyembunyikan dan mengungkapkan objek menurut onmouseover atau peristiwa lainnya, jangan lupa bahwa selalu ada beberapa properti CSS yang dapat kita gunakan untuk tujuan ini, tanpa mengambil tindakan drastis seperti membuat dan menghapus DOM elemen. Misalnya, ketika kita ingin membuat elemen tidak terlihat tetapi membiarkannya di tempatnya (dan dengan semua elemen di sekitarnya tetap berada di posisinya), kita cukup menyetel properti visibilitas objek ke 'tersembunyi', seperti ini:

```
myobject.visibility = 'hidden'
```

Dan untuk menampilkan kembali objek, kita dapat menggunakan yang berikut ini:

```
myobject.visibility = 'visible'
```

Anda juga dapat menciutkan elemen ke bawah untuk menempati lebar dan tinggi nol (dengan semua objek di sekitarnya mengisi ruang kosong), seperti ini:

```
myobject.display = 'none'
```

Untuk kemudian mengembalikan elemen ke dimensi aslinya, kita akan menggunakan yang berikut ini:

```
myobject.display = 'block'
```

Dan, tentu saja, selalu ada properti innerHTML, yang dengannya kita dapat mengubah HTML yang diterapkan ke elemen, seperti ini misalnya:

```
mylement.innerHTML = '<b>Replacement HTML</b>'
```

Atau kita dapat menggunakan fungsi O yang saya uraikan sebelumnya, seperti ini:

```
O('someid').innerHTML = 'New contents'
```

Atau kita dapat membuat elemen tampak menghilang, seperti ini:

```
O('someid').innerHTML = ''
```

Catatan: Jangan lupa properti CSS berguna lainnya yang dapat kita akses dari JavaScript, seperti opacity untuk mengatur visibilitas objek ke suatu tempat antara terlihat dan tidak

terlihat, atau lebar dan tinggi untuk mengubah ukuran objek. Dan, tentu saja, dengan menggunakan properti posisi dengan nilai 'absolut', 'statis', atau 'relatif', kita bahkan dapat menemukan objek di mana saja di (atau di luar) jendela browser yang kita suka.

13.7 MENGGUNAKAN INTERUPSI

JavaScript menyediakan akses ke interupsi, sebuah metode di mana kita dapat meminta browser untuk memanggil kode kita setelah jangka waktu tertentu, atau bahkan untuk terus memanggilnya pada interval tertentu. Ini memberi kita cara untuk menangani tugas latar belakang seperti komunikasi Ajax, atau bahkan hal-hal seperti animasi elemen web. Untuk mencapai ini, kita memiliki dua jenis interupsi: setTimeout dan setInterval, yang memiliki fungsi clearTimeout dan clearInterval yang menyertai untuk mematikannya lagi.

Menggunakan setTimeout

Saat kita memanggil setTimeout, kita meneruskannya beberapa kode JavaScript atau nama fungsi, dan nilai dalam milidetik yang menunjukkan berapa lama menunggu sebelum kode harus dieksekusi, seperti ini:

```
setTimeout(dothis, 5000)
```

Dan fungsi dothis kita mungkin terlihat seperti ini:

```
function dothis()
{
alert('This is your wakeup alert!');
}
```

Catatan: Jika kita bertanya-tanya, kita tidak bisa begitu saja menentukan alert() (dengan tanda kurung) sebagai fungsi yang akan dipanggil oleh setTimeout, karena fungsi tersebut akan langsung dieksekusi. Hanya ketika kita memberikan nama fungsi tanpa tanda kurung argumen (mis., Peringatan) kita dapat dengan aman meneruskan nama fungsi sehingga kodenya akan dieksekusi hanya ketika batas waktu terjadi.

Melewati string

Saat kita perlu memberikan argumen ke suatu fungsi, kita juga bisa meneruskan nilai string ke fungsi setTimeout, yang tidak akan dieksekusi hingga waktu yang tepat, seperti ini:

```
setTimeout("alert('Hello!')", 5000)
```

Sebenarnya, kita dapat memberikan baris kode JavaScript sebanyak yang kita suka, jika kita menempatkan titik koma setelah setiap pernyataan, seperti ini:

```
setTimeout("document.write('Starting'); alert('Hello!')", 5000)
```

Timeout berulang

Salah satu teknik yang digunakan beberapa programmer untuk menyediakan interupsi berulang dengan setTimeout adalah dengan memanggil fungsi setTimeout dari kode yang dipanggil olehnya, seperti berikut ini, yang akan memulai loop jendela peringatan yang tidak pernah berakhir:

```
setTimeout(dothis, 5000)
function dothis()
{
setTimeout(dothis, 5000)
alert('I am annoying!')
}
```

Sekarang peringatan akan muncul setiap lima detik.

Membatalkan Timeout

Setelah batas waktu diatur, kita dapat membatalkannya jika sebelumnya kita menyimpan nilai yang dikembalikan dari panggilan awal ke setTimeout, seperti ini:

```
handle = setTimeout(dothis, 5000)
```

Berbekal nilai dalam pegangan, kita sekarang dapat membatalkan interupsi kapan saja hingga waktunya, seperti ini:

```
clearTimeout(handle)
```

Ketika kita melakukan ini, interupsi benar-benar dilupakan, dan kode yang diberikan padanya tidak akan dieksekusi.

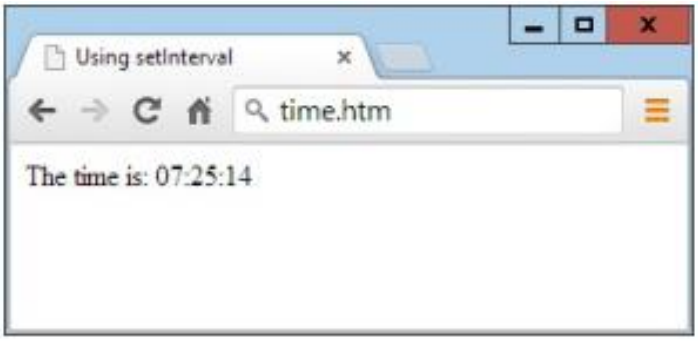
Menggunakan setInterval

Cara yang lebih mudah untuk mengatur interupsi reguler adalah dengan menggunakan fungsi setInterval. Ini bekerja dengan cara yang sama, kecuali bahwa setelah muncul setelah interval yang kita tentukan dalam milidetik, ia akan melakukannya lagi setelah interval itu berlalu lagi, dan seterusnya selamanya, kecuali jika kita membatalkannya. Contoh 9 menggunakan fungsi ini untuk menampilkan jam sederhana di browser, seperti yang ditunjukkan pada Gambar 13.4.

Contoh 9. Jam yang dibuat menggunakan interupsi

```
<!DOCTYPE html>
<html>
<head>
<title>Using setInterval</title>
<script src='OSC.js'></script>
</head>
<body>
The time is: <span id='time'>00:00:00</span><br>
<script>
setInterval("showtime(O('time'))", 1000)
function showtime(object)
{
var date = new Date()
object.innerHTML = date.toTimeString().substr(0,8)
}
</script>
```

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



Gambar 13.4 Mempertahankan waktu yang tepat dengan interupsi

Setiap kali ShowTime dipanggil, ia menetapkan tanggal objek ke tanggal dan waktu saat ini dengan panggilan ke Date:

```
var date = new Date()
```

Kemudian properti innerHTML dari objek yang diteruskan ke showtime (yaitu, objek) diatur ke waktu saat ini dalam jam, menit, dan detik, sebagaimana ditentukan oleh panggilan ke toTimeString. Ini mengembalikan string seperti 09:57:17 UTC+0530, yang kemudian dipotong menjadi hanya delapan karakter pertama dengan panggilan ke fungsi substr:

```
object.innerHTML = date.toTimeString().substr(0,8)
```

Menggunakan fungsi

Untuk menggunakan fungsi ini, pertama-tama kita harus membuat objek yang properti innerHTML-nya akan digunakan untuk menampilkan waktu, seperti HTML ini:

```
The time is: 00:00:00
```

Kemudian, dari bagian kode <script>, panggilan dilakukan ke fungsi setInterval, seperti ini:

```
setInterval("showtime(O('time'))", 1000)
```

Kemudian meneruskan string ke setInterval, yang berisi pernyataan berikut, yang diatur untu k dieksekusi sekali per detik (setiap 1.000 milidetik):

```
showtime(O('time'))
```

Dalam situasi yang jarang terjadi di mana seseorang telah menonaktifkan JavaScript (yang terkadang dilakukan orang untuk alasan keamanan), JavaScript kita tidak akan berjalan dan pengguna akan melihat 00:00:00 yang asli.

Membatalkan interval

Untuk menghentikan interval berulang, saat pertama kali mengatur interval dengan panggilan ke setInterval, kita harus membuat catatan tentang pegangan interval, seperti ini:

handle = setInterval("showtime(O('time'))", 1000)

Sekarang kita dapat menghentikan jam kapan saja dengan mengeluarkan panggilan berikut:

clearInterval(handle)

Anda bahkan dapat mengatur timer untuk menghentikan jam setelah jangka waktu tertentu, seperti ini:

setTimeout("clearInterval(handle)", 10000)

Pernyataan ini akan mengeluarkan interupsi dalam 10 detik yang akan menghapus interval berulang.

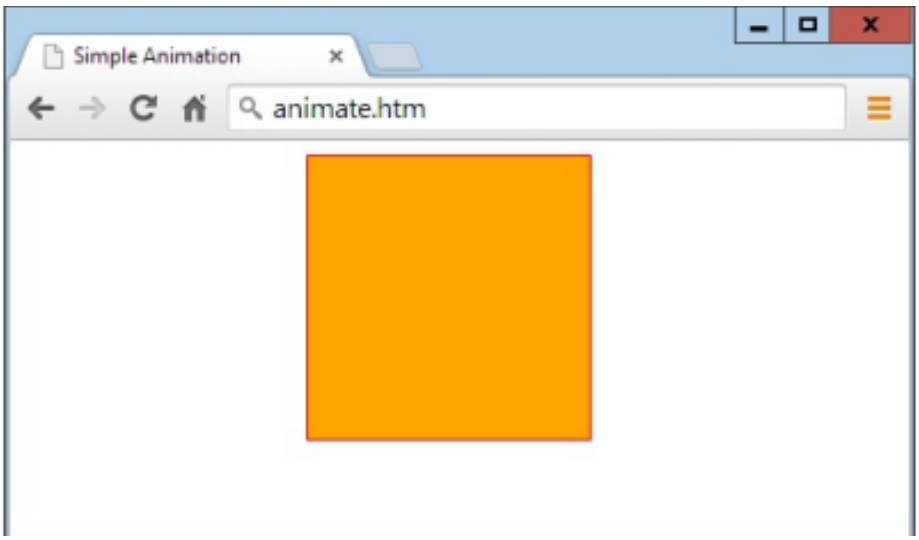
Menggunakan Interupsi untuk Animasi

Dengan menggabungkan beberapa properti CSS dengan interupsi berulang, kita dapat menghasilkan segala macam animasi dan efek. Misalnya, kode pada Contoh 10 memindahkan bentuk persegi di bagian atas browser, sepanjang waktu menggelembung dalam ukuran, seperti yang ditunjukkan pada Gambar 13.5 sebelum memulai dari awal lagi saat LEFT direset ke 0.

Contoh 10 Animasi sederhana

```
<!DOCTYPE html>
<html>
<head>
<title>Simple Animation</title>
<script src='OSC.js'></script>
<style>
#box {
position :absolute;
background:orange;
border :1px solid red;
}
</style>
</head>
<body>
<div id='box'></div>
<script>
SIZE = LEFT = 0
setInterval(animate, 30)
function animate()
{
SIZE += 10
LEFT += 3
if (SIZE == 200) SIZE = 0
if (LEFT == 600) LEFT = 0
```

```
S('box').width = SIZE + 'px'
S('box').height = SIZE + 'px'
S('box').left = LEFT + 'px'
}
</script>
</body>
</html>
```



Gambar 13.5 Objek ini meluncur dari kiri sambil mengubah ukuran

Dalam dokumen <head>, objek kotak diatur ke warna latar belakang 'oranye' dengan nilai batas '1px merah solid', dan properti posisinya disetel ke absolut sehingga diizinkan untuk dipindahkan dalam peramban. Kemudian, dalam fungsi animasi, variabel global SIZE dan LEFT terus diperbarui dan kemudian diterapkan ke atribut lebar, tinggi, dan gaya kiri objek kotak (dengan 'px' ditambahkan setelah masing-masing untuk menentukan bahwa nilainya dalam piksel) , sehingga menjiwainya pada frekuensi sekali setiap 30 milidetik— memberikan kecepatan 33,33 frame per detik (1.000/30 milidetik).

DAFTAR PUSTAKA

Abdulloh, Rohi. 2016. Easy & Simple Web Programing. PT Elex Media Komputindo. Jakarta.

Arief, M.R., 2011, Pemrograman Web Dinamis Menggunakan PHP Dan MySQL, Andi Offset, Yogyakarta.

Burnette. Ed. 2009. Brilliant HTML and CSS. United Kingdom: Pearson.

Djayali, A. D. 2020. Analisa Serangan SQL Injection pada Server pengisian Kartu Rencana Studi (KRS) Online. 1(1), 1–9.

Kadir, A., 2001, Pemrograman WEB Mencakup: HTML, CSS, JAVA SCRIPT, dan PHP, Andi Offset, Yogyakarta.

Kadir, A., 2008, Belajar Database Mengunakan MySQL, Andi Offset, Yogyakarta.

Kurniawan, Rulianto. 2010. PHP & MySQL untuk orang awam. Palembang: Maxikom.

Robon Nixon, 2014. Lerning PHP, MySQL, JavaScript, CSS & HTML5, A step by step guide to creating dynamic Websites. Third Edition. O 'Reilly Media, Inc. Canada.

Steve Suehring and Janet Valade, 2013. PHP, MySQL, Javascript & HTML5 All In One for DUMMIES. Published by John Wiley & Sons, Inc., Hoboken, New Jersey.

Styawantoro, I., & Komarudin, A. (2021). PEMROGRAMAN BERBASIS WEB HTML, PHP 7, MySQLi, Dan Bootstrap 4. Penerbit Lakeisha.

Sugiri. (2007). Desain Web Menggunakan HTML + CSS. Yogyakarta: Andi Offset.

Suryana, Taryana dan Koesheryatin. 2014. Aplikasi Internet Menggunakan HTML, CSS, & JavaScript. Elex Media Komputindo. Jakarta.

Sutarman., 2003, Membangun Aplikasi Web Dengan PHP Dan MySQL, Graha Ilmu, Yogyakarta.

Winarno, Edy, ST, M.Eng. Zaki, Ali. SmitDev Community. 2015. Desain Web Responsif dengan HTML5 dan CSS3. PT Elex Media Komputindo. Jakarta.

HTML CSS dan Javascript

Muhammad Sholikhan, S.Kom., M.Kom

BIODATA PENULIS

Penulis buku ini adalah dosen Universitas Sains dan Teknologi Komputer bernama Muhammad Sholikhan, S.Kom., M.Kom, lahir di kota Kudus, 14 Juni 1982. Penulis memiliki riwayat pendidikan S1 Jurusan Sistem Komputer Grafis di Sekolah Tinggi Elektronika dan Komputer (STEKOM) Semarang dan S2 Magister Sistem Informasi Universitas Kristen Satya Wacana (UKSW) Salatiga. Saat ini penulis adalah dosen tetap di Universitas Sains dan Teknologi Komputer (Universitas STEKOM) pada program studi S1 Desain Grafis dalam bidang ilmu Sistem Informasi dengan jabatan fungsional Asisten Ahli. Penulis mengampu mata kuliah antara lain Desain Web, Pengantar Teknologi Informasi serta Pengolahan dan Publikasi Konten Digital.

YAYASAN PRIMA AGUS TEKNIK

PENERBIT :

YAYASAN PRIMA AGUS TEKNIK
JL. Majapahit No. 605 Semarang
Telp. (024) 6723456. Fax. 024-6710144
Email : penerbit_ypat@stekom.ac.id

Muhammad Sholikhan, S.Kom., M.Kom

HTML CSS dan Javascript

YAYASAN PRIMA AGUS TEKNIK

PENERBIT :
YAYASAN PRIMA AGUS TEKNIK
JL. Majapahit No. 605 Semarang
Telp. (024) 6723456. Fax. 024-6710144
Email : penerbit_ypat@stekom.ac.id