Kapitel 22

Einführung in jQuery

In diesem Kapitel:

Steve Suehring: JavaScript - Schritt für Schritt, Microsoft Press 2011 (ISBN 978-3-86645-551-1)

Nachdem Sie dieses Kapitel gelesen haben, sind Sie in der Lage:

- jQuery in Ihr HTML einzubinden
- wichtige Konzepte und die Syntax von jQuery zu verstehen
- jQuery mit Ihren Webseiten einzusetzen

Grundlagen zu jQuery

jQuery ist ein bekanntes und leicht einzusetzendes JavaScript-Framework. Es erleichtert schwierige JavaScript-Aufgaben und beseitigt oftmals auch Probleme bei browserübergreifendem JavaScript.

Die gesamte jQuery-Bibliothek besteht aus nur einer einzigen JavaScript-Datei und lässt sich somit ganz einfach in Ihr JavaScript einbinden. Die jQuery-Syntax ist ebenfalls leicht zu erlernen; sie verwendet einen einfachen Namespace und einheitliche Funktionalität. Zusammen mit dem Add-On jQuery User Interface (mit dem sich Kapitel 23 befasst) können Sie leistungsfähige und höchst interaktive Webanwendungen herstellen.

Dieses Kapitel erläutert im Rahmen einer Einführung in jQuery unter anderem, wie Sie die Bibliothek herunterladen und in Ihrem JavaScript verwenden.

jQuery verwenden

jQuery ist unter *http://www.jquery.com/* erhältlich. In diesem Abschnitt erfahren Sie, wie Sie jQuery herunterladen und in eine Webseite integrieren.

Die beiden jQuery-Downloads

Auf der Homepage von jQuery sind zwei Downloads verfügbar: eine Produktionsversion und eine Entwicklungsversion. Sofern Sie nicht vorhaben, jQuery-Plug-Ins zu entwickeln oder sich die Interna von jQuery anzusehen, sollten Sie die verkleinerte Produktionsversion verwenden.

Insbesondere für das Durcharbeiten dieses Kapitels können Sie alternativ auch über ein Netzwerk für Inhaltsübermittlung (Content Delivery Network, CDN) auf eine gehostete Version von jQuery zugreifen. Unter anderem fungiert die API-Website von Google als Host für jQuery und andere Bibliotheken. Das heißt, dass Sie jQuery in Ihre Webseiten und JavaScript-Programme einbinden können, ohne die Datei lokal auf Ihrem Server hosten zu müssen. Weitere Informationen hierzu finden Sie unter *http://code.google.com/apis/libraries/devguide.html*.

HINWEIS Für fast alle Szenarios, in denen Sie mit jQuery arbeiten, empfehle ich, die jQuery-Datei herunterzuladen und lokal zu hosten. Mit der lokalen Version lässt sich möglicherweise schneller und zuverlässiger arbeiten als mit der CDN-Version. Wenn Sie zum Beispiel eine CDN-gehostete Version verwenden und der CDN-Server ausfällt, funktioniert auf Ihrer Site nichts mehr, was sich auf diese Bibliothek stützt! Allerdings ist eine CDN-gehostete Datei für die Entwicklungsaufgaben in diesem Kapitel durchaus akzeptabel.

Für die Übungen in diesem Kapitel ist es erforderlich, dass Sie jQuery auf Ihren lokalen Entwicklungscomputer heruntergeladen haben oder über ein CDN damit verbunden sind.

jQuery einbinden

In eine Webseite binden Sie jQuery auf die gleiche Weise ein wie jede andere externe JavaScript-Datei – mit einem <*script*>-Tag, das auf die Quelldatei verweist. Sehen Sie sich dazu den Code in Listing 22.1 an.

```
<!DOCTYPE HTML PUBLIC  "-//W3C//DTD HTML 4.01//EN"  "http://www.w3.org/TR/html4/
strict.dtd">
<html>
<head>
<title>jQuery einbinden</title>
<script type="text/javascript" src="jquery-1.5.1.min.js"></script>
</head>
<body>
</body>
</html>
```

Listing 22.1 jQuery in eine Webseite einbinden

Nachdem Sie jQuery heruntergeladen oder über eine CDN-Site referenziert haben und anhand des obigen Beispielcodes wissen, wie Sie jQuery in eine Datei einbinden, ist es an der Zeit, sich mit der jQuery-Syntax zu befassen.

WICHTIG Zum Zeitpunkt der Übersetzung war die Version 1.5.1 aktuell. Es ist davon auszugehen, dass sich die Version geändert hat, wenn Sie dieses Kapitel lesen, sodass Sie das *src*-Attribut entsprechend an die Version Ihres heruntergeladenen jQuery-Skripts anpassen müssen.

Grundlegende jQuery-Syntax

Wenn Sie die jQuery-Bibliothek in eine Seite einbinden, fügt jQuery eine Funktion *jquery()* hinzu. Man könnte nun annehmen, dass sämtliche Aufrufe von jQuery-Funktionen über diese *jquery()*-Funktionsschnittstelle erfolgen müssen, doch es gibt einen Shortcut für die *jquery()*-Funktion: *$()*. Anstatt jedes Mal **jquery** eintippen zu müssen, greifen Sie auf die jQuery-Bibliothek mit einem Dollarzeichen gefolgt von Klammern zu, wie es die Beispiele in Tabelle 22.1 zeigen.

Syntax	Beschreibung
$("a")	Alle <*a*>-Elemente im Dokument
$(document)	Das gesamte Dokument; häufig verwendet, um auf die später in diesem Kapitel gezeigte Funktion *ready()* zuzugreifen
$("#elementID")	Das Element, das durch die ID *elementID* identifiziert wird
$(".className")	Die Elemente, die zur Klasse *className* gehören

Tabelle 22.1 Einige jQuery-Selektoren

Weitere Selektoren und einschlägige Funktionen lernen Sie später in diesem Kapitel kennen.

Wie in JavaScript-Code sollten jQuery-Anweisungen mit einem Semikolon enden. Außerdem sei darauf hingewiesen, dass man entweder einfache oder doppelte Anführungszeichen als Selektoren in jQuery verwenden kann. Zum Beispiel sind die beiden folgenden Anweisungen gleichermaßen gültig:

```
$("a")

$('a')
```

Wenn Sie sich Beispiele der jQuery-Verwendung in der Praxis ansehen (was nicht heißen soll, dass dieses Buch praxisfremd ist), finden Sie sowohl einfache als auch doppelte Anführungszeichen. Die Beispiele in diesem Kapitel verwenden ebenfalls eine Mischung dieser Varianten, um Sie mit beiden Fällen vertraut zu machen. Allerdings ist es am besten, in der praktischen Programmierung einen Stil zu wählen und diesen konsequent beizubehalten.

jQuery mit dem load-Ereignis verbinden

Üblicherweise arbeitet man mit jQuery, indem man Elemente während des *load* (oder *onload*)-Ereignisses der Seite verbindet. (Dieses Kapitel geht später ausführlicher auf Ereignisse und Funktionen ein.) In jQuery bewerkstelligen Sie das über die Hilfsfunktion *.ready()* des *document*-Elements.

Wie Sie aus dem kurzen Beispiel im vorherigen Abschnitt wissen, greift jQuery auf Elemente mit der *$()*-Syntax zu. Somit können Sie auf das *document*-Element wie folgt zugreifen.

```
$(document)
```

Die Funktion *ready()* können Sie dann folgendermaßen aufrufen:

```
$(document).ready()
```

Die folgende Übung setzt voraus, dass Sie entweder jQuery auf Ihren lokalen Entwicklungscomputer heruntergeladen haben oder ein CDN verwenden. Das Beispiel verwendet die Version 1.5.1 von jQuery, wobei sich aber diese Versionsnummer möglicherweise bereits geändert hat, wenn Sie die Übung ausführen.

Document Ready verwenden

1. Bearbeiten Sie die Datei *docready.html* im Ordner der Begleitdateien von Kapitel 22 mit Visual Studio, Eclipse oder einem anderen Editor.
2. Ersetzen Sie in dieser Datei den TODO-Kommentar durch den folgenden fettgedruckten Code:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Document Ready</title>
<script type="text/javascript" src="jquery-1.5.1.min.js"></script>
</head>
<body>
<script type="text/javascript">
$(document).ready(alert('hi'));
</script>
</body>
</html>
```

3. Speichern Sie die Datei und öffnen Sie sie in einem Webbrowser. Es erscheint ein *alert()*-Dialogfeld wie in Abbildung 22.1 gezeigt.

Abbildung 22.1 Meldungsdialogfeld für das jQuery-Beispiel

Der Code in dieser Schritt-für-Schritt-Übung kombiniert jQuery über die Funktion *$(document).ready()* mit normalem JavaScript-Code, der im Beispiel durch die *alert()*-Funktion verkörpert wird. Diese Mischung aus jQuery und JavaScript stellt ein wichtiges Konzept dar: Mit jQuery ergänzen Sie normales JavaScript. Viele schwierige und manchmal auch profane Aufgaben lassen sich mit jQuery leichter realisieren – letztlich so leicht, dass Sie Ihre Zeit sinnvoller in das Erstellen von Features investieren können, anstatt sich den Kopf über browserübergreifende Nuancen zu zerbrechen.

Die Funktion *$(document).ready()* macht es überflüssig, das *load*-Ereignis des Browsers zu verwenden oder einen Funktionsaufruf in das *load*-Ereignis einzufügen. Mit *$(document).ready()* stehen alle Elemente des DOM (Document Object Model) vor der Funktion *.ready()* zur Verfügung.

TIPP Die Funktion *$(document).ready()* spielt in vielen Programmieraufgaben mit jQuery eine zentrale Rolle.

Selektoren verwenden

Selektoren sind der Schlüssel für das Arbeiten mit jQuery und dem DOM. Mithilfe von Selektoren identifizieren und gruppieren Sie die Elemente, auf denen eine jQuery-Funktion ausgeführt wird. Wie aus Tabelle 22.1 hervorgeht, sammeln Sie mit Selektoren alle Elemente eines bestimmten Tags, einer bestimmten ID oder mit einer bestimmten Klasse, die ihnen zugeordnet ist. Außerdem können Sie Selektoren noch wirkungsvoller einsetzen, etwa um eine festgelegte Anzahl von Elementen oder nur Elemente mit einer bestimmten Herkunft – beispielsweise nur diejenigen <*p*>-Tags, die auf ein <*div*>-Tag folgen – auszuwählen. Dieser Abschnitt beschäftigt sich ausführlich mit Selektoren.

TIPP Selektoren und ihre Arbeitsweise in jQuery basieren auf Selektoren in CSS. Wenn Sie mit deren Verwendung in CSS vertraut sind (wie in Kapitel 15 erläutert), werden Sie sich in diesem Modell schnell zuhause fühlen.

Elemente nach ID auswählen

Das Beispiel in Tabelle 22.1 hat die allgemeine Syntax für die Auswahl eines Elements nach seinem ID-Attribut gezeigt:

```
$("#elementID")
```

Sehen Sie sich zum Beispiel folgenden HTML-Ausschnitt an:

```
<a href="#" id="linkOne">Link</a>
```

Mit normalem JavaScript greifen Sie folgendermaßen auf dieses Element zu:

```
getElementById("linkOne")
```

Mit jQuery sieht der Zugriff auf das Element so aus:

```
$("#linkOne")
```

Elemente nach Klasse auswählen

Um Elemente nach Klasse auszuwählen, setzen Sie einen Punkt (.) vor den Klassennamen. Die Syntax lautet:

```
$(".className")
```

Zum Beispiel weist der folgende Code einem *div*-Element die Klasse *specialClass* zu:

```
<div class="specialClass">
```

Auf dieses Element greifen Sie über jQuery folgendermaßen zu:

```
$(".specialClass")
```

Beachten Sie, dass Sie möglicherweise nicht nur auf ein einzelnes Element zugreifen; der Klassenselektor greift auf *alle* Elemente zu, denen die angegebene Klasse zugeordnet ist. Mit anderen Worten: Wenn mehreren Elementen in der Seite die Klasse *"specialClass"* zugeordnet ist, greift jQuery über den Selektor *$(".specialClass")* auf alle diese Elemente zu. Mehr zu diesem Thema lernen Sie später kennen, wenn es um Funktionen geht, die jedes mit einem Selektor abgerufene Element durchlaufen.

Elemente nach Typ auswählen

Selektoren erlauben es auch, auf Elemente nach Typ zuzugreifen, beispielsweise auf alle <*div*>-Elemente, alle <*a*>-Elemente usw. So zeigt der folgende Code, wie Sie auf alle <*div*>-Elemente in einem Dokument zugreifen:

```
$('div')
```

Um auf alle <*a*>-Elemente zuzugreifen, schreiben Sie dementsprechend:

```
$('a')
```

Ein Typselektor bietet Zugriff auf alle Elemente des angegebenen Typs auf einer Seite. Wie der Klassenselektor können Typselektoren mehrere Elemente zurückgeben.

Elemente nach Hierarchie auswählen

Wie bereits erwähnt, können Sie Elemente nach ihrer Position in Bezug auf andere Elemente auf der Seite auswählen. Um zum Beispiel alle <*a*>-Elemente auszuwählen, die in <*div*>-Elementen enthalten sind, verwenden Sie diese Syntax:

```
$("div a")
```

Die Auswahl lässt sich auch spezifischer festlegen. Brauchen Sie zum Beispiel alle Ankerelemente, die nur auf ein bestimmtes *div*-Element folgen, kombinieren Sie den Typselektor mit der Syntax des ID-Selektors. Als Beispiel sei folgendes HTML gegeben:

```
<div id="leftNav">
<a href="link1.html">Link 1</a>
<a href="link2.html">Link 2</a>
</div>
```

Mit der folgenden jQuery-Selektorsyntax rufen Sie die beiden Ankerelemente innerhalb des *div*-Elements *leftNav* ab:

```
$("#leftNav a")
```

Möchten Sie lediglich die direkten Nachfahren eines Elements haben, können Sie dies allgemeiner mit dem Größer-als-Zeichen formulieren:

```
$("div > p")
```

Diese Syntax liefert alle <*p*>-Elemente, die direkte Nachfolger eines *div*-Elements sind, jedoch keine <*p*>-Elemente, die in den ausgewählten <*p*>-Elementen enthalten sind.

Mit dem Selektor *:nth-child()* lässt sich das *n*-te untergeordnete Elemente in einer Menge auswählen. Das folgende Beispiel wählt das dritte untergeordnete Element aus:

```
$("p:nth-child(3)")
```

Daneben gibt es noch andere hierarchische Selektoren. Weitere Informationen finden Sie in der Referenzdokumentation für jQuery-Selektoren unter *http://api.jquery.com/category/selectors/* (in englischer Sprache).

Elemente nach Position auswählen

Wie die letzten Abschnitte gezeigt haben, sind die Selektoren in jQuery »gierig« (greedy). Zum Beispiel wählt die Syntax $('a') alle Ankertags aus. In jQuery gibt es aber auch mehrere Möglichkeiten, um Elemente in einer Gruppe gezielter auszuwählen. Dazu gehören die Selektoren *first* und *last*. Der folgende Code wählt das erste <*p*>-Element in der Seite aus:

```
$("p:first")
```

Analog dazu lässt sich das letzte Element wie folgt auswählen:

```
$("p:last")
```

Außerdem können Sie Elemente durch ihre direkte Position auswählen. Nehmen Sie als Beispiel folgendes HTML:

```
<p>Erstes P</p>
<p>Zweites P</p>
<p>Drittes P</p>
```

Um das zweite <*p*>-Element auszuwählen, verwenden Sie diese Syntax:

```
$("p")[1]
```

Zu beachten ist, dass der Arrayindex für einen derartigen Selektor bei *0* beginnt, sodass das erste Element den Index *0*, das zweite den Index *1* hat usw. Diese Syntax ist etwas gefährlich, da sie sich auf die strikte Positionierung der Elemente innerhalb der Hierarchie verlässt. Wenn jemand ein anderes <*p*>-Tag in die Seite vor dem Element, das Sie auswählen möchten, einfügt, verschiebt sich die Indexposition und der Selektor liefert das falsche Element zurück. Deshalb ist es besser, einen ID-Selektor zu verwenden, um ein einzelnes oder spezifisches Element auszuwählen, anstatt sich auf die Position eines Elements zu verlassen.

Eine Alternative zur Auswahl nach Index ist die *:eq*-Syntax. Um zum Beispiel den dritten Absatz auszuwählen, können Sie schreiben:

```
$("p:eq(3)")
```

Schließlich sind manchmal auch die Positionsselektoren *even* (gerade) und *odd* (ungerade) willkommen, die jedes zweite Element in einer Menge auswählen:

```
$("p:even")
```

Die Selektoren *even* und *odd* sind insbesondere für tabellarische Daten zweckmäßig, wenn die Zeilen alternierend gefärbt werden sollen. Listing 22.2 zeigt ein Beispiel mit dem *odd*-Selektor, um den ungeraden Zeilen einer Tabelle eine andere Hintergrundfarbe als den geraden Zeilen zuzuweisen.

HINWEIS Der Code in Listing 22.2 verwendet zwei Elemente, die bisher noch nicht formal eingeführt wurden: eine benutzerdefinierte Funktion und die Funktion *.css()*. Machen Sie sich keine großen Gedanken darum, mehr zu diesen Elementen erfahren Sie später im Kapitel.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"  "http://www.w3.org/TR/html4/strict.
dtd">
<html>
<head>
<title>Tabellentest</title>
<script type="text/javascript" src="jquery-1.5.1.min.js"></script>
</head>
<body>
<table>
  <tr>
    <td>Zeile 1 Spalte 1 der Tabelle</td>
    <td>Zeile 1 Spalte 2 der Tabelle</td>
  </tr>
  <tr>
    <td>Zeile 2 Spalte 1 der Tabelle</td>
    <td>Zeile 2 Spalte 2 der Tabelle</td>
```

```
  </tr>
  <tr>
    <td>Zeile 3 Spalte 1 der Tabelle</td>
    <td>Zeile 3 Spalte 2 der Tabelle</td>
  </tr>
  <tr>
    <td>Zeile 4 Spalte 1 der Tabelle</td>
    <td>Zeile 4 Spalte 2 der Tabelle</td>
  </tr>
  <tr>
    <td>Zeile 5 Spalte 1 der Tabelle</td>
    <td>Zeile 5 Spalte 2 der Tabelle</td>
  </tr>
  <tr>
    <td>Zeile 6 Spalte 1 der Tabelle</td>
    <td>Zeile 6 Spalte 2 der Tabelle</td>
  </tr>
</table>
<script type="text/javascript">
$(document).ready(function() {
    $('tr:odd').css("background-color", "#abacab");
});
</script>

</body>
</html>
```

Listing 22.2 Tabellendaten und jQuery

Der entscheidende Teil dieses Codes befindet sich im JavaScript-Abschnitt im <*body*>-Teil des HTML-Codes:

```
$(document).ready(function() {
    $('tr:odd').css("background-color", "#abacab");
});
```

Der Code verwendet die Funktion *$(document).ready()* zusammen mit dem *:odd*-Selektor, um die Hintergrundfarbe auf den hexadezimalen Wert *#abacab* – eine hellgraue Färbung – zu setzen. Abbildung 22.2 zeigt ein Beispiel für die Ausgabe.

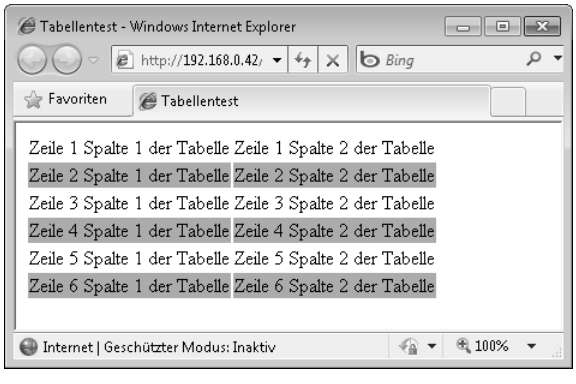

Abbildung 22.2 Eine mithilfe von jQuery eingefärbte Tabelle

HINWEIS Die gebräuchlichsten Positionsselektoren kennen Sie nun. Darüber hinaus stehen viele andere Positionsselektoren zur Verfügung. Weitere Informationen erhalten Sie unter *http://api.jquery.com/category/selectors/*.

Elemente nach Attribut auswählen

Wie die Erläuterungen zum Klassenselektor nahe legen, lassen sich mit jQuery auch Elemente auswählen, die lediglich ein Attribut oder ein Attribut mit einem bestimmten Wert enthalten. So wählt der folgende Code alle Bilder aus, denen ein *alt*-Attribut zugeordnet ist:

```
$("img[alt]")
```

Um nur Bilder auszuwählen, bei denen ein *alt*-Attribut auf einen bestimmten Wert gesetzt ist, können Sie schreiben:

```
$("img[alt='alternate text']")
```

Der obige Code wählt ein Bild nur aus, wenn dem *alt*-Attribut der Text *alternate text* zugewiesen ist. Beachten Sie in diesem Beispiel, wie einzelne und doppelte Anführungszeichen alternierend verwendet werden. Der *img*-Selektor ist in doppelte Anführungszeichen eingeschlossen, während der innere Selektor für das *alt*-Attribut von einfachen Anführungszeichen umgeben ist. Genauso gut hätte man umgekehrt die einfachen Anführungszeichen für den *img*-Selektor und die doppelten Anführungszeichen für den *alt*-Attributselektor verwenden können:

```
$('img[alt="alternate text"]')
```

Prinzipiell ist es auch möglich, die gleichen Anführungszeichen für beide Selektoren zu verwenden, doch ist es dazu erforderlich, die inneren Anführungszeichen mit Escapezeichen zu versehen:

```
$("img[alt=\"alternate text\"]")
```

Wichtig ist, dass ein derartiger Selektor eine genaue Übereinstimmung erwartet. Im Beispiel muss das *alt*-Attribut die Zeichenfolge *"alternate text"* aufweisen. Jede Variation wie etwa *"alternate text 2"* oder *" alternate text "* ergibt keine Übereinstimmung. In jQuery existieren aber auch Platzhalterselektoren, die keine exakte Übereinstimmung bei Attributen verlangen. Sehen Sie sich dazu die Beispiele in Tabelle 22.2 an.

Syntax	Beschreibung
attribute=value*	Wählt Elemente aus, die das Attribut (*attribute*) enthalten, dessen Attributwert den angegebenen Wert (*value*) als Teilzeichenfolge enthält
attribute~=value	Wählt Elemente aus, die das Attribut enthalten, dessen Attributwert den angegebenen Wert als ganzes Wort enthält
attribute!=value	Wählt Elemente aus, die entweder das Attribut nicht enthalten oder für die der Attributwert nicht mit dem angegebenen Wert übereinstimmt
attribute$=value	Wählt Elemente aus, die das angegebene Attribut enthalten, dessen Attributwert mit der angegebenen Zeichenfolge endet
attribute^=value	Wählt Elemente aus, die das Attribut enthalten, dessen Attributwert mit der angegebenen Zeichenfolge beginnt

Tabelle 22.2 Übereinstimmungen bei Attributselektoren

Formularelemente auswählen

In jQuery gibt es auch native Selektoren, die sich auf Webformulare beziehen. Tabelle 22.3 listet ausgewählte Selektoren auf, wobei einige davon im Rest dieses Kapitels verwendet werden.

Selektor	Beschreibung
:checkbox	Wählt alle Kontrollkästchen aus
:checked	Wählt alle aktivierten Elemente aus, beispielsweise aktivierte Kontrollkästchen
:input	Wählt alle Eingabeelemente auf einer Seite aus
:password	Wählt alle Kennworteingaben aus
:radio	Wählt alle Optionsfeldeingaben aus
:reset	Wählt alle Eingaben vom Typ *reset* aus
:selected	Wählt alle Elemente aus, die momentan markiert sind
:submit	Wählt alle Eingaben vom Typ *submit* aus
:text	Wählt alle Eingaben vom Typ *text* aus

Tabelle 22.3 Formularbezogene Selektoren

Weitere Selektoren

In jQuery gibt es noch mehr Selektoren, wie zum Beispiel Selektoren, die sämtliche ausgeblendeten Elemente (*:hidden*) oder alle sichtbaren Elemente (*:visible*) sowie aktivierte, deaktivierte und andere Elemente auswählen. Eine vollständige und aktualisierte Liste der Selektoren in jQuery finden Sie unter *http://api.jquery.com/category/selectors/*.

TIPP Anstatt sich eine komplexe und anfällige Syntax auszudenken, um zu einem bestimmten Element zu gelangen, sollten Sie in der Referenz der jQuery-Selektoren (*http://api.jquery.com/category/selectors/*) nachsehen, ob bereits jemand das betreffende Selektorproblem gelöst hat.

Funktionen

Inzwischen kennen Sie jede Menge Beispiele, die Elemente mit jQuery auswählen, jedoch nur wenige Beispiele, die zeigen, was Sie mit diesen Elementen anstellen können, nachdem Sie sie ausgewählt haben. Um Aktionen auf ausgewählten Elementen auszuführen, verwendet jQuery Funktionen. Dabei ist zwischen integrierten Funktionen von jQuery und benutzerdefinierten Funktionen zu unterscheiden. Fast immer werden Sie beide Arten gleichzeitig verwenden.

Das DOM durchsuchen

Das Wesen der Programmierung im Web mit JavaScript und jetzt auch jQuery verlangt häufig, mehrere Elemente zu durchsuchen oder zu durchlaufen – zum Beispiel übernimmt die Funktion *.each()* eine Liste von ausgewählten Elementen, geht die Elemente nacheinander durch und führt beim Schleifendurchlauf

durch die Liste auf jedem Element eine Aktion (oder nichts) aus. In jQuery sind zahlreiche Funktionen für die Schleifenbearbeitung und das schrittweise Durchlaufen vorhanden. Man bezeichnet diese Vorgänge auch als *Traversieren* oder *Durchsuchen*. Weitere Informationen zu Funktionen, die für das Durchsuchen relevant sind, finden Sie unter *http://api.jquery.com/category/traversing/*.

Wenn Sie Funktionen zum Durchsuchen verwenden, werden Sie dabei fast immer auf eine benutzerdefinierte Wrapperfunktion zusammen mit dem Selektor *$(this)* zurückgreifen. Wie das Schlüsselwort *this* in der objektorientierten Programmierung verweist der Selektor *$(this)* auf das aktuelle Objekt – in diesem Fall auf das momentan traversierte Element.

Hier dürfte ein Beispiel zweckmäßig sein. Das folgende HTML erstellt eine Spielstandsseite für eine fiktive Volleyball-Meisterschaft:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Iterationstest</title>
<script type="text/javascript" src="jquery-1.5.1.min.js"></script>
</head>
<body>
<table>
  <th>Team-Name</th>
  <th>GV-Satz</th>
  <th>Gewinn-%</th>
  <tr>
    <td>Team 1</td>
    <td>12-8</td>
    <td class="percentage">.600</td>
  </tr>
  <tr>
    <td>Team 5</td>
    <td>11-9</td>
    <td class="percentage">.550</td>
  </tr>
  <tr>
    <td>Team 4</td>
    <td>10-10</td>
    <td class="percentage">.500</td>
  </tr>
  <tr>
    <td>Team 2</td>
    <td>9-11</td>
    <td class="percentage">.450</td>
  </tr>
  <tr>
    <td>Team 6</td>
    <td>6-14</td>
    <td class="percentage">.300</td>
  </tr>
  <tr>
    <td>Team 3</td>
    <td>2-18</td>
    <td class="percentage">.100</td>
  </tr>
</table>
<script type="text/javascript">
$(document).ready(function() {
    $('tr:odd').css("background-color", "#abacab");
```

```
});
</script>

</body>
</html>
```

Abbildung 22.3 zeigt die in einem Webbrowser geöffnete Seite.

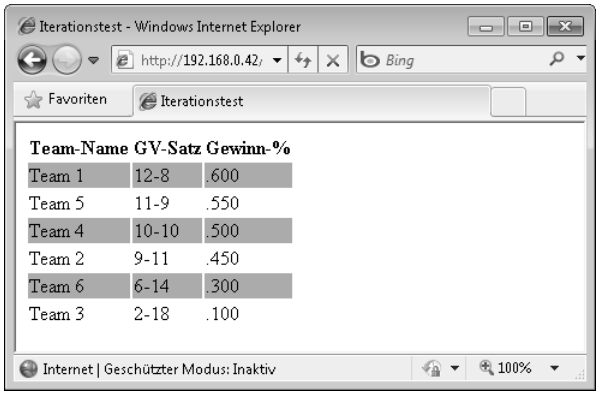

Abbildung 22.3 Spielstandsseite für eine fiktive Volleyball-Meisterschaft

Dieses Beispiel durchläuft alle Elemente, deren *class*-Attribut mit *percentage* übereinstimmt – d.h. mit einer Klasse, die den Zellen in der Spalte *Gewinn-%* der Tabelle zugeordnet ist. Für jedes Team, dessen Gewinn/Verlust-Verhältnis größer oder gleich 0,500 ist (das also mindestens die Hälfte seiner Spiele gewonnen hat), weist der Beispielcode dem entsprechenden Feld eine Fettschrift zu. Dies lässt sich mit dem folgenden jQuery-Code realisieren, der unmittelbar unter dem bereits in der Seite enthaltenen jQuery-Code einzufügen ist:

```
$('.percentage').each(function() {
    if ($(this).text() >= .5) {
        $(this).css('font-weight', 'bold');
    }
});
```

Dieser Code verwendet einen Selektor, um alle Elemente zu erfassen, denen die Klasse *percentage* zugeordnet ist. Dann greift er auf jedes dieser Elemente mithilfe der Funktion *.each()* in jQuery zu. In der Funktion *.each()* ermittelt eine benutzerdefinierte Funktion mit einer Bedingungsanweisung, ob der Wert in der Spalte *Gewinn-%* größer oder gleich 0,5 ist. Trifft dies zu, ruft der Code die Funktion *.css()* auf, um diesem Element eine *font-weight*-Eigenschaft hinzuzufügen, die auf *bold* (fett) gesetzt ist. Listing 22.3 zeigt das Ergebnis, nachdem Sie diesen Code in die Seite eingefügt haben.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict. dtd">
<html>
<head>
<title>Iterationstest</title>
<script type="text/javascript" src="jquery-1.5.1.min.js"></script>
</head>
<body>
<table>
  <th>Team-Name</th>
```

```
  <th>GV-Satz</th>
  <th>Gewinn-%</th>
  <tr>
    <td>Team 1</td>
    <td>12-8</td>
    <td class="percentage">.600</td>
  </tr>
  <tr>
    <td>Team 5</td>
    <td>11-9</td>
    <td class="percentage">.550</td>
  </tr>
  <tr>
    <td>Team 4</td>
    <td>10-10</td>
    <td class="percentage">.500</td>
  </tr>
  <tr>
    <td>Team 2</td>
    <td>9-11</td>
    <td class="percentage">.450</td>
  </tr>
  <tr>
    <td>Team 6</td>
    <td>6-14</td>
    <td class="percentage">.300</td>
  </tr>
  <tr>
    <td>Team 3</td>
    <td>2-18</td>
    <td class="percentage">.100</td>
  </tr>
</table>
<script type="text/javascript">
$(document).ready(function() {
    $('tr:odd').css("background-color", "#abacab");
    $('.percentage').each(function() {
        if ($(this).text() >= .5) {
            $(this).css('font-weight', 'bold');
        }
    });
});
</script>

</body>
</html>
```

Listing 22.3 Die Seite für die Volleyball-Meisterschaft mit zusätzlichem jQuery-Code

Wenn Sie diese Seite in einem Browser öffnen, ist nun die Spalte für diejenigen Teams, die mindestens die Hälfte der Spiele gewonnen haben, fett formatiert, wie Abbildung 22.4 zeigt.

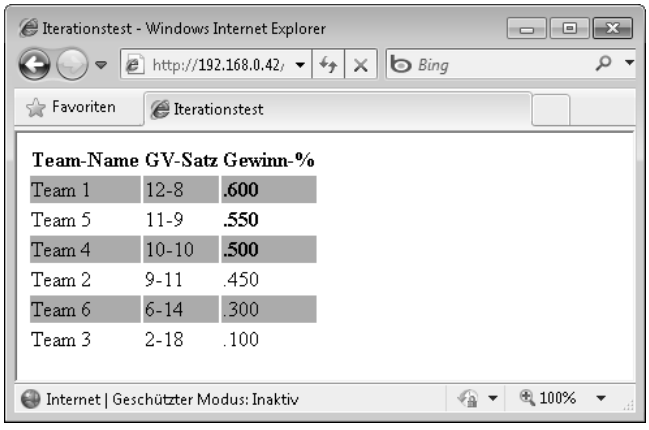

Abbildung 22.4 Die Spalte Gewinn-% ist jetzt mithilfe von jQuery fett formatiert

Wie die Ausgabe in Abbildung 22.4 zeigt, würde es sicherlich besser aussehen, die Fettschrift auf die gesamte Tabellenzeile und nicht nur auf die Spalte *Gewinn-%* anzuwenden. Dies scheint schwierig zu sein, da der Code logisch bereits hinter dieser HTML-Tabellenzeile liegt, wenn der Test stattfindet, um den Gewinnanteil zu ermitteln. Hier kommt uns jQuery mit der Funktion *.parent()* zu Hilfe. (Prinzipiell gibt es mehrere Möglichkeiten, diese Aufgabe zu realisieren. Die Funktion *.parent()* ist lediglich eine davon.)

Mit der Funktion *parent()* geht man praktisch in der DOM-Struktur nach oben, um das <*tr*>-Tag zu finden, das das jeweilige <*td*>-Element umschließt. Indem man die CSS-Formatänderung auf das <*tr*>-Element anwendet, kann man die gesamte Zeile fett formatieren. Der neue Code sieht wie folgt aus (Änderung fettgedruckt):

```
$('.percentage').each(function() {
    if ($(this).text() >= .5) {
        $(this).parent().css('font-weight', 'bold');
    }
});
```

Wenn Sie diese Änderung in den Code von Listing 22.3 übernehmen, sieht die Ausgabe wie in Abbildung 22.5 gezeigt aus.

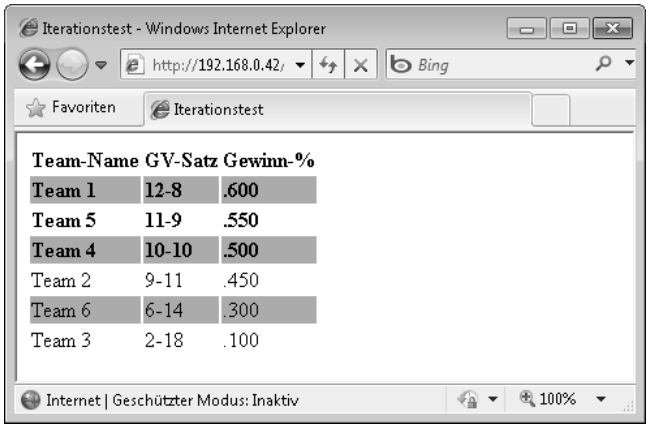

Abbildung 22.5 Eine CSS-Formatierung auf der Ebene der Tabellenzeile zuweisen

HINWEIS Dieser überarbeitete Code ist in der Begleitdatei *listing22-4.html* enthalten.

Mit der Funktion *.parent()* kommen wir zu einem neuen Konzept, das als *Verkettung* bezeichnet wird. Verkettung ist ein leistungsfähiges Konstrukt in jQuery, da es zusätzliche Auswahlebenen sowie die Anwendung von Funktionen über mehrere Ebenen hinweg ermöglicht. Im Beispiel wird der Selektor *$(this)* mit der Funktion *.parent()* verkettet, wodurch das übergeordnete Element von *$(this)* ausgewählt wird. Nur dann führt der Code die Funktion *.css()* aus.

Die Leistungsfähigkeit der Verkettung birgt aber auch gewisse Risiken. Es ist durchaus möglich, dass Sie auf diesem Wege zu schwer verständlichem und schwer zu wartendem Code gelangen. Außerdem kann durch Verkettung schwer kontrollierbarer Code entstehen, wenn sich die Elemente in einem verketteten Selektor ändern. Verkettung ist leistungsfähig – und Sie sollten sie wenn möglich auch verwenden – doch sollte dies nicht zu Lasten der Verständlichkeit oder Wartbarkeit gehen.

Die bisher in diesem Kapitel gezeigten Beispiele haben das CSS per JavaScript direkt angesprochen und geändert. Wie Kapitel 15 erläutert hat, empfiehlt es sich nicht, das Format oder die Darstellungsaspekte einer Webseite über JavaScript zu ändern. Besser ist es, Formate über CSS anzuwenden oder zu entfernen, anstatt Attribute direkt zu ändern. Es existieren mehrere Methoden, um mit CSS-Formatklassen in jQuery zu arbeiten. Dazu gehören *.hasClass()*, *.addClass()*, *.removeClass()* und *.toggleClass()*. Weitere Informationen zum Arbeiten mit Klassen über diese und andere Funktionen finden Sie unter *http://api.jquery.com/category/css/*.

Mit Attributen arbeiten

Neben den klassenbezogenen Attributfunktionen bietet jQuery auch Funktionen, um mit Attributen des DOM zu arbeiten. Die generischste davon ist die Funktion *.attr()*, obwohl auch andere Funktionen wie *.html()* und *.val()* nützlich sind. Dieser Abschnitt beschäftigt sich mit der Funktion *.attr()*, die Funktionen *.html()*, *.val()* und andere kommen in einem späteren Abschnitt zur Sprache. Mit der Funktion *.attr()* lassen sich Attribute sowohl abrufen als auch festlegen. Zum Beispiel können Sie das *alt*-Attribut mit der folgenden Syntax abrufen bzw. setzen:

```
// Das alt-Attribut abrufen:
$("#myImageID").attr("alt")
// Das alt-Attribut festlegen:
$("#myImageID").attr("alt", "new text")
```

HINWEIS Es ist nicht erforderlich, den Wert des Elements abzurufen, bevor man ihn festlegt.

Text und HTML ändern

Mit Funktionen wie *.text()* und *.val()* können Sie eine Seite vollkommen neu schreiben. Aber nur weil man es kann, heißt das nicht, dass es eine gute Idee ist. Dennoch kann es manchmal notwendig sein, Teile von HTML in einer Seite neu zu schreiben oder Text bzw. Werte zu ändern. Die Funktion *.html()* ruft das gesamte HTML in einem ausgewählten Element ab oder legt es fest. Sehen Sie sich zum Beispiel folgendes HTML an:

```
<div id="myDiv">Hier ist ein div-Element</div>
```

Der Code mit jQuery sieht folgendermaßen aus:

```
$("#myDiv").html('<span class="redStyle">Dies ist der neue Inhalt des div-Elements</span>');
```

Im Ergebnis dieses jQuery-Fragments enthält jetzt das <*div*>-Element mit der ID *myDiv* ein <*span*>-Element mit neuem Text. Dieses Beispiel ist zwar stark vereinfacht, doch stellen Sie sich vor, wenn <*div*> einen ganzen Inhaltsabschnitt enthalten würde. Mit jQuery können Sie praktisch diesen gesamten Abschnitt, HTML usw. neu schreiben.

Wie die Funktion *.html()* unterstützt die Funktion *.text()* sowohl das Abrufen als auch Festlegen von Text innerhalb eines ausgewählten Elements. Im Unterschied zu *.html()* ändert die Funktion *.text()* nur den Text. Es ist also nicht möglich, das HTML innerhalb des ausgewählten Elements zu ändern.

```
<div id="myDiv">Hier ist ein div-Element</div>
$("#myDiv").text('Dies ist der neue Inhalt des div-Elements');
```

Dieser Beispielcode hat nur den Text geändert; er hat weder ein *span*-Tag hinzugefügt noch Formate angewandt.

Elemente einfügen

Mit jQuery lassen sich auch leicht Elemente zu einer Seite hinzufügen. Hierfür sind vor allem die beiden Funktionen *:after()* und *:before()* vorgesehen. Wie aus den Namen hervorgeht, fügen diese Funktionen Elemente entweder nach oder vor einem ausgewählten Element ein.

Nehmen Sie als Beispiel noch einmal das *div*-Element:

```
<div id="myDiv">Hier ist ein div-Element</div>
```

Das folgende jQuery-Fragment fügt vor diesem *div*-Element ein weiteres *div*-Element ein:

```
$("#myDiv").before("<div>Dies ist ein neues div-Element</div>");
```

Die Funktion *:after()* arbeit in ähnlicher Weise:

```
$("#myDiv").after("<div>Dies ist ein neues div-Element, es erscheint nach myDiv</div>");
```

Nach der Ausführung gibt es in der Seite, die diesen Code enthält, drei <*div*>-Elemente:

```
<div>Dies ist ein neues div-Element</div>
<div id="myDiv">Hier ist ein div-Element</div>
<div>Dies ist ein neues div-Element, es erscheint nach myDiv</div>
```

Die gezeigten Beispiele fügen zusätzliche <*div*>-Elemente ein – natürlich ist es aber möglich, jedes gültige Element in diesen Funktionen einzufügen.

Callback-Funktionen

Manchmal müssen Sie eine Funktion ausführen, wenn eine andere Funktion oder der Teil einer Funktion abgeschlossen ist. Ein derartiges Konstrukt bezeichnet man als *Callback-Funktion* (Rückruffunktion). Eine Callback-Funktion wird ausgeführt, nachdem ihre übergeordnete Funktion beendet ist. jQuery macht ausgiebig Gebrauch von Callback-Funktionen, insbesondere in AJAX. Ein Beispiel für eine Callback-Funktion haben Sie bereits beim Durchsuchen einer Liste mit der Funktion *.each()* kennengelernt.

Weitere Informationen zu Callback-Funktionen finden Sie unter *http://docs.jquery.com/Tutorials:How_jQuery_Works* (in englischer Sprache).

Die Codebeispiele im Rest dieses Kapitels verwenden ebenfalls Callback-Funktionen. Wenn Sie Einsteiger oder Fortgeschrittener in der JavaScript-Programmierung sind, sollten Sie sich nicht zu viele Gedanken über Callback-Funktionen machen. Es handelt sich lediglich um einen Codeblock, der innerhalb einer anderen Funktion aufgerufen wird.

Ereignisse

Inzwischen haben Sie mehrere Beispiele von Selektoren kennengelernt und an der Oberfläche von Funktionen in jQuery gekratzt. Der letzte Teil dieser Einführung zu jQuery dreht sich nun um Ereignisse. Genau wie die Ereignisbehandlung von JavaScript versetzt Sie jQuery in die Lage, dass Ihre Programme auf Mausklicks, Formularübertragungen, Tastenbetätigen usw. reagieren. Im Unterschied zu JavaScript ist die browserübergreifende Ereignisbehandlung in jQuery recht einfach. Für derartige Umgebungen ist jQuery geradezu prädestiniert und erspart es Ihnen insbesondere bei der Ereignisbehandlung herauszufinden, wie jeder Browser auf bestimmte Funktionen reagiert.

Binden und Bindung aufheben

Die Funktion *.bind()* verbindet einen Ereignishandler mit einem Ereignis, beispielsweise einem Mausklick:

```
.bind(event, data, handler)
```

In diesem Beispiel ist *event* das Ereignis, auf das Sie reagieren möchten, *data* ein optionales Objekt mit zusätzlichen Daten, die an den Ereignishandler zu übergeben sind, und *handler* die Funktion, die als Reaktion auf dieses Ereignis auszuführen ist.

Zum Beispiel:

```
<a href="/link1.html" id="myLink">Ein Link</a>
$("#myLink").bind("click", function() {
alert("Link wurde angeklickt");
});
```

Im Ergebnis dieses Codes zeigt die Seite eine *alert()*-Meldung an, nachdem das *click*-Ereignis für das Ankertag abgefangen wurde. Beachten Sie, dass dieses Beispiel nicht den optionalen Parameter im Aufruf von *.bind()* verwendet.

Mit der Funktion *.bind()* lassen sich die folgenden Ereignisse binden:

- *beforeunload*
- *blur*
- *change*
- *click*
- *dblclick*
- *hover*
- *keydown*
- *keypress*
- *keyup*
- *load*
- *mouseout*
- *mouseover*
- *mouseup*
- *resize*
- *scroll*

- *error*
- *focus*
- *focusin*
- *focusout*
- *mousedown*
- *mouseenter*
- *mouseleave*
- *mousemove*
- *select*
- *submit*
- *toggle*
- *unload*

In vorherigen Kapiteln haben Sie Ereignisse mit dem in Kapitel 11 entwickelten Skript *ehandler.js* abgefangen. Das Skript *ehandler.js* realisiert einen generischen Ereignishandler, der bis zu einem gewissen Grad browserübergreifend arbeitet. Praktisch setzt das Skript das um, was die Funktion *.bind()* von jQuery verkörpert. Der Unterschied besteht darin, dass die Funktion *.bind()* in Bezug auf die browserübergreifende Ereignisbehandlung wesentlich besser abschneidet und leistungsfähiger ist als das Skript *ehandler.js*.

Es ist zwar durchaus richtig, die Funktion *.bind()* für die Ereignisbehandlung zu verwenden, doch bietet jQuery auch Shortcut-Funktionen, die in der gleichen Weise wie *.bind()* arbeiten. So können Sie anstelle von *.bind("click", function())...* einfach *.click(function()...* schreiben. Zum Beispiel lässt sich das weiter oben angegebene *.bind()*-Beispiel wie folgt neu formulieren:

```
$("#myLink").click(function() {
alert("Link wurde angeklickt");
});
```

Es ist nicht nur möglich, auf Ereignisse wie zum Beispiel das Anklicken eines Links zu reagieren, Ereignisse lassen sich auch auslösen. Sehen Sie sich als Beispiel den Code in Listing 22.5 an.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Trigger-Test</title>
<script type="text/javascript" src="jquery-1.5.1.min.js"></script>
</head>
<body>
<div id="myDiv">
Hier steht etwas Text.<br>
Er steht in diesem div-Element<br>
</div>
<p>
<a id="braingiaLink" href="http://www.braingia.org">Steve Suehring</a>
</p>
<script type="text/javascript">
$(document).ready(function() {

$('#braingiaLink').click(function() {
    alert("Hallo");
    return true;
});

$('#myDiv').click(function() {
    $('#braingiaLink').click();
});

});
</script>

</body>
</html>
```

Listing 22.5 Auf Ereignisse reagieren

Wenn diese Seite in einen Webbrowser geladen wird, löst ein Klicken an beliebiger Stelle innerhalb des <*div*>-Elements das *click*-Ereignis für den Anker aus, als hätten Sie auf den Anker selbst geklickt.

Um das Reagieren auf Ereignisse zu stoppen, können Sie mit der Funktion *.unbind()* die Bindung aufheben. Die Funktion *.unbind()* übernimmt zwei Argumente:

```
.unbind(event, function)
```

Das Argument *event* ist das Ereignis, auf das nicht mehr reagiert werden soll, und das Argument *function* bezeichnet die Funktion, die momentan an das Ereignis gebunden ist.

HINWEIS Um mehrere Ereignishandler mit demselben Ereignis zu verbinden, rufen Sie *.bind()* entsprechend mehrere Male für dieses Ereignis auf.

Mausereignisse und Hover

Die vorhergehenden Beispiele haben gezeigt, wie sich das *click*-Ereignis binden und behandeln lässt. Darüber hinaus ist es auch möglich, Mausereignisse wie *mouseover* und *mouseout* zu verarbeiten. So ist denkbar, Elemente verschwinden zu lassen, wenn der Benutzer den Mauszeiger daraufsetzt (was möglicherweise zur Frustration des Benutzers führt und auf einer realen Site nicht verwendet werden sollte). Der Code in Listing 22.6 lässt einen Anker verschwinden, wenn der Benutzer den Mauszeiger über den darin enthaltenen Absatz führt.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<title>Trigger-Test</title>
<script type="text/javascript" src="jquery-1.5.1.min.js"></script>
<style type="text/css">
#braingiaLink {
    border: solid 1px black;
    padding: 3px;
}
#wrapperP {
    padding: 50px;
}
</style>
</head>
<body>
<div id="myDiv">
Hier steht etwas Text.<br>
Er steht in diesem div-Element<br>
</div>
<p id="wrapperP">
<a id="braingiaLink" href="http://www.braingia.org">Steve Suehring</a>
</p>
<script type="text/javascript">
$(document).ready(function() {
$('#braingiaLink').click(function() {
    alert("Hallo");
    return true;
});
```

```
$('#myDiv').click(function() {
    $('#braingiaLink').click();
});

$('#wrapperP').mouseover(function() {
    $('#braingiaLink').hide();
});

$('#wrapperP').mouseout(function() {
    $('#braingiaLink').show();
});

});
</script>

</body>
</html>
```

Listing 22.6 Mit Mausereignissen arbeiten

Im Kern besteht dieser Code aus den *.mouseover()*- und *.mouseout()*-Ereignishandlern, die ihrerseits die jQuery-Funktionen *.hide()* und *.show()* aufrufen. Die Ereignisse *.mouseover()* und *.mouseout()* sind mit dem Absatz mit der ID *wrapperP* verknüpft. Wenn sich der Mauszeiger über diesem Absatz befindet, verschwindet der Anker mit der ID *braingiaLink* und erscheint erst wieder, wenn der Mauszeiger den Absatzbereich verlässt. Beachten Sie, dass sich der Link trotzdem noch mithilfe der Tastaturnavigation aktivieren lässt. Denken Sie immer daran, dass viele Wege nach Rom führen – d.h. eine Webseite lässt sich auf mehreren Wegen umgehen.

Außerdem gibt es in jQuery eine Funktion *.hover()*, die fast das Gleiche wie die Ereignisse *.mouseover()* und *.mouseout()* leistet. Weitere Informationen zur Funktion *.hover()* finden Sie unter *http://api.jquery.com/hover/* (in englischer Sprache).

Viele weitere Ereignishandler

Wie aus der weiter vorn gezeigten Liste hervorgeht, gibt es in jQuery zahlreiche andere Ereignishandler – zu viele, als dass sie sich in einem Einführungskapitel zu jQuery besprechen lassen. Deshalb empfehle ich die ausgezeichnete Dokumentation zu jQuery-Ereignissen unter *http://api.jquery.com/category/events/* (in englischer Sprache).

AJAX und jQuery

Die beiden letzten Kapitel haben gezeigt, wie Sie AJAX schreiben und verwenden. Nachdem Sie die beiden vorherigen Abschnitte dieses Kapitels gelesen haben, werden Sie sicherlich schon vermuten, das jQuery eigene Methoden besitzt, um mit AJAX zusammenzuarbeiten. Und wie bei vielen anderen JavaScript-bezogenen Aufgaben, erleichtert jQuery auch das Arbeiten mit AJAX. Dieser Abschnitt zeigt, wie Sie AJAX mit jQuery verwenden.

In jQuery existieren mehrere Funktionen, um Daten von einem Server zu verarbeiten und an einen Server zu senden. Dazu gehören die Funktionen *.load()*, *.post()* und *.get()*. Außerdem gibt es eine spezielle AJAX-Funktion, die passend mit *.ajax()* benannt ist.

Mit der Funktion *.ajax()* lassen sich mehrere Parameter festlegen, unter anderem die HTTP-Methode, die der Aufruf verwenden soll (*GET* oder *POST*), das Time-out und die auszuführende Aktion, wenn ein Fehler auftritt (und natürlich auch, wenn der Code erfolgreich ausgeführt wird).

HINWEIS Eine vollständige Liste der verfügbaren Parameter für die Funktion *.ajax()* finden Sie unter *http://api.jquery.com/jQuery.ajax/*.

Die grundlegende Syntax der Funktion *.ajax()* lautet:

```
$.ajax({
    parameter: value
});
```

An die Funktion *.ajax()* können Sie eine Reihe von *parameter*: *value*-Paaren übergeben, wobei Sie normalerweise die Methode, die URL und eine Callback-Funktion spezifizieren. Üblicherweise legt man auch fest, welcher Datentyp zurückzugeben ist, ob die Antwort zwischenzuspeichern ist, welche Daten an den Server übergeben werden und was bei einem Fehler passieren soll.

HINWEIS Mit der Funktion *.ajaxSetup()* können Sie Standardwerte für AJAX-bezogene Parameter festlegen, unter anderem für Caching, Methoden und Fehlerbehandlung.

Der folgende Code zeigt ein praxisnahes Beispiel für die Funktion *.ajax()*:

```
$.ajax({
    url:  "testajax.aspx",
    success:  function(data) {
        alert("Erfolgreich geladen");
    }
});
```

Außerdem gibt es in jQuery die Funktion *.getJSON()*, die praktisch die gleichen Aufgaben wie die anderen AJAX-bezogenen Funktionen realisiert, aber speziell mit JSON-kodierten Daten vom Server arbeitet. Die Funktion *.getJSON()* ist das Äquivalent zum Aufruf der Funktion *.ajax()* mit dem zusätzlichen Parameter *dataType: 'json'*.

Sehen Sie sich als Beispiel die folgende JSON-kodierte Liste von US-Bundesstaaten an:

```
["Wisconsin","California","Colorado","Illinois","Minnesota","Oregon","Washington","New
York","New Jersey","Nevada","Alabama","Tennessee","Iowa","Michigan"]
```

Für dieses Beispiel sei angenommen, dass die JSON-kodierten Daten zurückgegeben werden, wenn die Datei *json.php* auf dem lokalen Server aufgerufen wird. Der folgende Code holt mit der Funktion *.ajax()* die Daten ab und ruft eine Funktion *showStates* auf, wenn die Übertragung erfolgreich war:

```
$.ajax({
    type: "GET",
    url: "json.php",
    dataType: "json",
    success: showStates
});
```

Die Funktion *showStates* erzeugt eine Liste und fügt sie zum <*select*>-Dropdownfeld eines Formulars hinzu.

AJAX mit jQuery verwenden

Um diese Schritt-für-Schritt-Übung durchzuführen, ist die Datei *json.php* erforderlich, die sich im selben Verzeichnis wie die Datei für diese Übung befinden muss (als Begleitdatei *json.php* vorhanden). Wie bei den Beispielen der vorherigen Kapitel zu AJAX muss sich die Datei in derselben Domäne wie die Datei, die die AJAX-Anforderung ausführt, befinden.

1. Bearbeiten Sie die Datei *ajax.html* im Ordner der Begleitdateien von Kapitel 22 mit einem Editor Ihrer Wahl.
2. Ersetzen Sie in der Datei den TODO-Kommentar durch den folgenden fettgedruckten Code (der auch in der Begleitdatei *ajax.txt* enthalten ist):

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"  "http://www.w3.org/TR/html4/strict. dtd">
<html>
<head>
<title>AJAX-Test</title>
<script type="text/javascript" src="jquery-1.5.1.min.js"></script>
</head>
<body>
<div id="states">
</div>
<script type="text/javascript">
$(document).ready(function() {

$.ajax({
    type: "GET",
    url: "json.php",
    dataType: "json",
    success: showStates
});

function showStates(data,status) {
    $.each(data, function(item) {;
        $("#states").append("<div>" + data[item] + "</div>");
    });
}


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

3. Speichern Sie die Datei und öffnen Sie sie in einem Webbrowser. Es erscheint eine Liste der US-Bundesstaaten, wie Abbildung 22.6 zeigt.

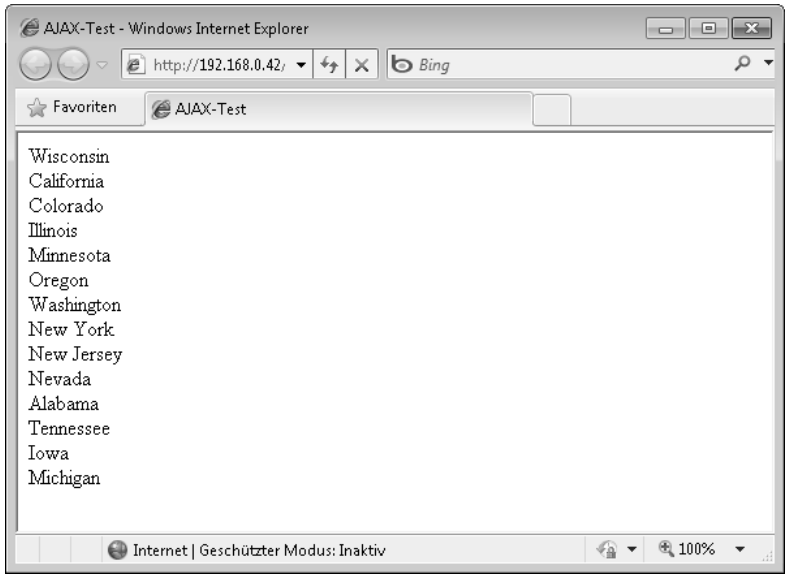

Abbildung 22.6 Seite mit den US-Bundesstaaten

AJAX-Fehler und Time-outs

Mit der Funktion *.ajax()* lassen sich Fehler und Time-outs (Zeitüberschreitungen) ordnungsgemäß behandeln. Neben dem Handler für erfolgreiche Verarbeitung (*success*) können Sie einen Handler mit dem Parameter *error* spezifizieren. Der Wert des *error*-Parameters ist gewöhnlich eine Callback-Funktion. Im folgenden Beispiel ist der neu hinzugefügte *error*-Parameter fettgedruckt:

```
$.ajax({
    type: "GET",
    url: "json.php",
    dataType: "json",
    success: successFunction,
    error:  errorFunction
});
```

Die für die Fehlerbehandlung verwendete Callback-Funktion (*errorFunction* im vorherigen Codebeispiel) übernimmt drei Argumente: das *XMLHTTPRequest*-Objekt, eine Zeichenfolge, die den aufgetretenen Fehler darstellt, und ein Ausnahmeobjekt, falls eine Ausnahme aufgetreten ist. Demzufolge sollte eine Funktion zur Fehlerbehandlung diese drei Argumente übernehmen und die Ergebnisse in geeigneter Weise verarbeiten. Das folgende Beispiel zeigt eine Warnungsmeldung:

```
function errorFunction(xhr, statusMessage, exceptionObj) {
    alert("Folgender Fehler ist aufgetreten: " + statusMessage);
}
```

Manchmal ist es erforderlich, einen Time-out für eine AJAX-Anforderung festzulegen. Das ist zum einen möglich mit einem generischen AJAX-Time-out-Wert, den Sie über die Standardfunktion *$.ajaxSetup()* spezifizieren, zum anderen aber auch für jeden einzelnen Aufruf mithilfe des Parameters *timeout*. Zum Beispiel:

```
$.ajax({
    type: "GET",
    url: "json.php",
    dataType: "json",
    success: successFunction,
    error:  errorFunction,
    timeout: 5000
});
```

Achten Sie darauf, die Time-outs in Millisekunden anzugeben. Das obige Beispiel setzt also den Time-out auf 5 Sekunden.

Daten an den Server senden

In einem AJAX-Aufruf müssen Sie nicht nur Daten von einem Server empfangen, sondern auch Daten an einen Server senden und eine Antwort entgegennehmen. Hierfür verwenden Sie in der Funktion *.ajax()* den Parameter *data* und senden die Daten entweder mit *GET* oder *POST*.

Die Daten können Sie als Schlüssel/Wert-Paare (*key=value*) formatieren und jeweils durch ein kaufmännisches Und-Zeichen trennen *(key1=value1&key2=value2)* oder als zugeordnete Paare schreiben *{key1: value1, key2: value2}*. Das nachfolgende Beispiel verwendet die Option *key=value*, die auch als *Abfragezeichenfolge* bezeichnet wird.

Der Code ruft ein serverseitiges Programm namens *statefull.php* auf, das für ein zweibuchstabiges Staatenkürzel den vollständigen Staatennamen zurückgibt:

```
 $.ajax({
    type: "POST",
    url: "statefull.php",
    dataType: "json",
    success: successFunction,
    data: "state=WI"
});
```

Andere wichtige Optionen

Für die Funktion *.ajax()* gibt es zahlreiche Optionen, von denen Sie einige bereits kennengelernt haben. In diesem Abschnitt möchte ich zwei Optionen erläutern:

- *async*
- *cache*

Die Option *async* informiert das Skript darüber, ob es warten (und weitere Eingaben im Browser blockieren) soll, während die AJAX-Transaktion gesendet, empfangen und verarbeitet wird. Ist die Option auf *true* gesetzt (Standardeinstellung), läuft die AJAX-Transaktion asynchron ab, d.h. blockiert nicht.

Die *cache*-Einstellung, die in den meisten Fällen auf den Standardwert *true* gesetzt wird, steuert, ob jQuery die AJAX-Transaktion zwischenspeichert. Dies ist zweckmäßig, wenn sich die empfangenen Daten nicht oft ändern, da Caching die Transaktion beschleunigt. Allerdings kann Caching (Zwischenspeicherung) auch zu Problemen führen, wenn die Anwendung mit veralteten Daten aus dem Cache arbeitet, obwohl sich die Daten auf dem Server bereits geändert haben. Ich erachte es als hilfreich, diese Option auf *false* zu setzen, sodass die Antwort nicht zwischengespeichert wird. Das gilt vor allem dann, wenn Probleme auftreten, da die Daten augenscheinlich nicht aktualisiert werden.

Mehr jQuery

Dieses Kapitel konnte nur einen kleinen Teil dessen vorstellen, was jQuery zu leisten vermag. Wenn Sie mehr über JavaScript lernen möchten und wie Sie damit Ihre Websites aktiver gestalten, sollten Sie auf jQuery oder eine andere JavaScript-Bibliothek zurückgreifen, die Sie bei Ihren Entwicklungsaufgaben unterstützt.

Weiterführende Lern- und Referenzmaterialien zu jQuery finden Sie unter *http://www.jquery.com*.

Übungen

1. Verwenden Sie den Code in *ajax.html* (aus dem Beispiel »Ajax mit jQuery verwenden«) als Basis, fügen Sie einen CSS-Stil hinzu, um die Hintergrundfarbe des jeweiligen <*div*>-Elements eines Staates in Blau zu ändern, wenn sich der Mauszeiger über einem der Staaten in der Liste befindet. Hinweis: Diese Aufgabe lässt sich auf mehreren Wegen realisieren.
2. Erstellen Sie ein serverseitiges Programm, das Daten zurückgibt, wenn Sie Parameter mithilfe der Funktion *$.ajax()* übergeben. Verarbeiten Sie die Daten, indem Sie etwa eine Warnungsmeldung ausgeben oder Text in die Seite schreiben. Zum Beispiel könnten Sie ein serverseitiges Programm implementieren, um die Summe zweier Zahlen zurückzugeben oder wie im gezeigten Beispiel den vollständigen Staatennamen zu liefern, wenn dem Programm die Abkürzung des Staats übergeben wird.