Open Source Library

Ralph Steyer

jQuery

Update-Service
auf
jquery.safety-first-rock.de

Das JavaScript-Framework für interaktives Design

ADDISON-WESLEY

CD

3 Grundlagenwissen

Wir haben im letzten Kapitel erste Beispiele mit jQuery durchgespielt. Nun ist es an der Zeit, uns etwas den Grundlagen zu widmen. Noch nicht im Detail von jQuery selbst, aber in der Welt, in der jQuery eingesetzt und wie das Framework darin verankert wird. Und diese Welt ist das WWW mit HTML bzw. XHTML, CSS, JavaScript, XML, JSON und AJAX. Denn wie viele andere Frameworks und Toolkits in diesem Bereich ist ja jQuery eine JavaScript-Erweiterung für Webseiten, die zudem neben einer eigenständigen Syntax gewisse CSS-Features (und im Fall von jQuery UI sogar eigene CSS-Themen) und Unterstützung für AJAX liefert. Sie können jQuery ohne elementare Grundlagenkenntnisse der Basistechnologien kaum sinnvoll und effektiv anwenden. Wir werden bei allen grundlegenden Ausführungen natürlich den jQuery-Aspekt auch in dem Kapitel im Fokus haben und unter anderem an dieser Stelle zeigen, wie Sie die jQuery-Bibliothek in Ihre Webseite einbinden und was Sie dabei zu beachten haben. Deshalb geht es jetzt ran an den Speck, mit dem wir dann jQuery anbraten wollen.

HINWEIS

Was dieses Kapitel nicht leisten kann, soll und will, ist, eine vollständige Einführung in die jeweiligen Techniken zu geben. Sie finden hier nur die grundlegenden Informationen, die aus Sicht von jQuery unabdingbar sind. Bei Bedarf sollten Sie sich mit zusätzlichen Quellen einarbeiten. Zu JavaScript als Kerntechnologie von jQuery finden Sie im Anhang eine Abhandlung über die wichtigsten Grundlagen.

3.1 Das WWW, das Web 2.0 und das Client-Server-Prinzip im Internet

Zu Beginn nur eine ganz kurze Einleitung zum World Wide Web – die grundlegenden Fakten zum WWW werden bekannt sein. Das WWW basiert im Kern auf dem Dienstprotokoll HTTP (Hypertext Transfer Protocol) zur Kommunikation, der Dokumentenbeschreibungssprache HTML (Hypertext Markup Language) respektive dem strengen XML-basierenden Zwilling XHTML (Extensible Hypertext Markup Language) sowie den Programmtypen Webserver und Webbrowser.

Das WWW ist als Dienst im Internet – wie alle dort verfügbaren Dienste – ein klassisches Client-Server-System, bei dem jede Aktion aus einem Zyklus »Anfordern einer Leistung – Bereitstellen der Leistung« besteht. Konkret bedeutet das, dass im Web so gut wie immer ein Browser (der Client) eine neue Datei (in der Regel eine Webseite oder Inhalt, der in eine Web-

seite eingebaut werden soll) anfordert und eventuell darin referenzierte externe Ressourcen wie Grafiken, Videos, Flash-Animationen oder externe JavaScript- oder CSS-Dateien nachgefordert werden. Diese werden dann zusammen mit der Webseite im Browser dargestellt oder sonst verwendet. Die zentrale Steuerungstechnik ist immer die (X)HTML-Datei.

3.1.1 Programmierung im WWW

Über die Jahre hat sich das WWW nun zu einem System entwickelt, in dem man als Anbieter von Content sowohl auf dem Server als auch Client programmieren kann. Insbesondere ist in den letzten Jahren die zeitweilige Hysterie bezüglich der Gefahren clientseitiger Programmierung abgeklungen, und nahezu alle modernen Webseiten verwenden clientseitige Programmierung für den Teil der Geschäftslogik, der sinnvollerweise schon im Client erledigt werden sollte. Zwar ist über die Jahre als einziger relevanter Vertreter der clientseitigen Zunft JavaScript übrig geblieben, aber das wird mittlerweile wieder flächendeckend eingesetzt und akzeptiert. Schauen Sie sich heute einmal populäre Webseiten im Internet an. Keine einzige der heutzutage angesagten Webseiten kommt ohne JavaScript daher. Und fast alle Anwender im Web haben JavaScript im Browser aktiviert. Denn kaum ein Anwender möchte auf den uneingeschränkten Nutzen populärer Webangebote wie Google, Amazon, eBay, Facebook, Twitter, Wikipedia oder Yahoo verzichten. Als Ersteller von Webangeboten können Sie im Umkehrschluss JavaScript samt zusammenhängender Techniken heutzutage wieder bei den meisten Clients voraussetzen[1]. Und damit fallen bezüglich JavaScript-Frameworks wie jQuery oder auch Dojo Toolkit, Prototype, YUI etc. die meisten Argumente gegen einen Einsatz weg.

HINWEIS

Wie schon vorher im Buch erwähnt, versuchen gerade verschiedene Hersteller, proprietäre Techniken wie JavaFX oder Silverlight im Clientsystem zu etablieren, um den Einschränkungen von JavaScript und Webbrowsern als Clients zu entfliehen. Aber es wird noch dauern, bevor diese flächendeckend unterstützt und akzeptiert werden. Teilweise erscheint es gar fraglich, ob diese proprietären Ansätze sich überhaupt durchsetzen.

3.1.2 Das Web 2.0

So genial und im Grunde einfach das Konzept des WWW ist – es gibt ein grundsätzliches und gravierendes Problem mit HTTP, HTML und dem Konzept klassischer Webbrowser. Bei der Anfrage des Browsers an einen Webserver nach neuen Daten muss dieser immer eine vollständige Webseite als Antwort senden. Oder genauer – der Browser versteht die Antwort so, dass er die bisher im Browser angezeigte Seite durch diesen neuen Inhalt vollständig ersetzt. Offensichtlich ist das sehr ineffektiv. Und hier kommt AJAX ins Spiel, auf dessen technischen Hintergrund wir gleich noch genauer eingehen.

1 Nach den mir bekannten Statistiken schwankt die Unterstützung für JavaScript massiv. Manche Statistiken gehen von 99,9 % Verfügbarkeit aus, während andere »nur« 99,1% Verfügbarkeit sehen ;-). Ich hoffe, Sie bemerken die Ironie – man kann die Verfügbarkeit aktuell fast uneingeschränkt voraussetzen.

Allgemein geht es um ein Verfahren, wie man eine Reaktion einer Webapplikation in (nahezu) Echtzeit gewährleisten kann, obwohl neue Daten vom Webserver angefordert werden. Statt zu einer vollständigen Webseite mit im Prinzip schon im Browser vorhandenen Daten wird eine AJAX-Datenanfrage dazu führen, dass nur die wirklich neuen Daten vom Webserver geschickt und diese dann mit DHTML-Mitteln in die bereits beim Client geladene Webseite »eingebaut« werden. Dabei wird in der Regel nicht einmal die normale Interaktion des Benutzers mit der Webanwendung durch das Laden neuer Daten unterbrochen. Dies erlaubt nun auch im Web die Erstellung von Angeboten, die sehr stark auf Interaktion mit dem Anwender setzen. Insbesondere Google hat dafür gesorgt, dass sich AJAX mittlerweile als Standardverfahren solcher interaktiver Webseiten etabliert hat. Und das Schlagwort Web 2.0 hat sich so ab dem Jahr 2005 als Oberbegriff der meisten interaktiven Webangebote breitgemacht. Oft nennt man das Web 2.0 auch »Mitmach-Web«, weil Anwender nicht nur reine Konsumenten sind, sondern selbst Content beisteuern. Denken Sie beispielsweise an Blogs, Twitter, Wikis oder Communities wie Xing, Facebook, MySpace etc. Aber auch wenn Anwender etwa Daten in einen Online-Kalender eintragen, wird das in eine veränderte Darstellung der Webseite führen (der Termin wird beispielsweise angezeigt – und natürlich auch auf dem Server gespeichert). So gesehen ist auch dies eine Form des Mitmachens im Web 2.0.

3.2 JavaScript und das Verhältnis zu jQuery

Da jQuery im Wesentlichen eine JavaScript-Bibliothek darstellt, müssen wir uns natürlich genauer mit dieser Sprache beschäftigen. Auch wenn es zu den Marketingaussagen von jQuery als auch diverser anderer Frameworks gehört, dass man bei dessen Verwendung viele Dinge in Hinsicht auf JavaScript abgenommen bekommt. Sie kommen für den Einsatz von jQuery um JavaScript-Grundkenntnisse nicht umhin. Und wenn Sie jQuery richtig effektiv nutzen wollen (etwa zur Erstellung von Plug-Ins), sollten Sie JavaScript sogar richtig gut beherrschen.

HINWEIS

Tiefer in JavaScript steigen wir in den nächsten Kapiteln ein, wenn es sich aus Sicht von jQuery jeweils anbietet. Hier soll nur etwas Grundsätzliches zu JavaScript ausgeführt werden. Grundkonzepte und Grundlagen zu JavaScript finden Sie zudem im Anhang.

Allgemein sind Skriptsprachen im WWW eine der wichtigsten Erweiterungen von HTML bzw. XHTML und realisieren die clientseitige Logik einer Webapplikation. Bei diesen Skriptsprachen handelt es sich um Interpretersprachen, die im Rahmen eines umgebenden Programms (des Browsers) zur Laufzeit übersetzt und ausgeführt werden. Das gilt insbesondere für JavaScript.

HINWEIS

Neue Browser verfügen über einen Just-in-time-Compiler für JavaScript. Damit wird das Interpreterprinzip für JavaScript erweitert um die Möglichkeit, bereits einmal übersetzten Code im Speicher vorzuhalten und bei einer erneuten Ausführung performanter bereitzustellen. Das ist ein Schlüssel für RIAs, die sich von der Performance her wie Desktop-Applikationen verhalten sollen.

Nun ist jQuery im Kern wie gesagt eine JavaScript-Bibliothek. Das heißt mit anderen Worten, dass jQuery nur auf einer Funktionalität beruht, die jeder moderner Browser zur Verfügung stellt. Es wird kein Plug-In oder sonst eine Erweiterung des Browsers benötigt, damit die Funktionen in dieser Bibliothek verfügbar sind. Es wird also dem Browser oder gar dem Betriebssystem (wie es bei Konkurrenztechnologien der Fall sein kann) nicht hinzugefügt. Das ist einerseits von erheblichem Vorteil, andererseits kann man in der Bibliothek nur das realisieren, was man mit JavaScript beziehungsweise DHTML und CSS sowie der DOM-Manipulation auch machen kann. Um nun aber diese spannenden und mächtigen Effekte und Leistungen von jQuery mithilfe dieser einfachen Basistechnologien bereitstellen zu können, mussten diese bis an die Grenze ausgereizt werden. Und das führt dazu, dass nicht jeder Browser mehr vollständig unterstützt werden kann (insbesondere keine älteren Browser).

Wenn Sie richtig gut JavaScript programmieren können, könnten Sie sämtliche Funktionalität von jQuery im Grunde selbst nachbauen. Nur wäre das halt ein »gewisser« Aufwand. Das Team von jQuery hat entsprechend viele Jahre Arbeit bereits in die Entwicklung dieser Bibliothek investiert, und diese Arbeit können Sie für sich nutzen.

3.2.1 Die allgemeine Einbindung von JavaScript in Webseiten

JavaScript ist als eine unmittelbare Ergänzung und Erweiterung zu HTML bzw. XHTML zu sehen und nur als eingebundener Bestandteil eines entsprechenden Gerüsts einer Webseite zu verwenden. Webskripte werden als Klartext in eine Webseite direkt notiert oder eingebunden und zur Laufzeit interpretiert. Es gibt verschiedene Techniken, wie Skripte mit einer Webseite verbunden werden können. Zwei wollen wir kurz ansprechen.

Der <script>-Container in der Webseite

Die Verbindung von einem JavaScript mit einer Webseite kann etwa über eine direkte Notation eines JavaScripts in der Webseite erfolgen. Dazu werden JavaScript-Anweisungen einfach in die entsprechende (X)HTML-Datei als Klartext hineingeschrieben. Der Beginn eines Skripts wird dazu durch eine eigene Steueranweisung gekennzeichnet, die noch zu HTML gehört und die mit ihrem zugehörigen Abschluss-Tag einen Container für die Skriptanweisungen bildet. Über das `<script>`-Element wird angegeben, dass all das, was in dem eingeschlossenen Container steht, ein Skript ist. Im Inneren eines solchen Containers führen Sie also JavaScript aus.

Betrachten Sie das nachfolgende Codefragment:

Listing 3.1: **Ein Codefragment mit der direkten Notation von einem JavaScript in eine Webseite**

```
<body>
...
 <script>
 ... Skriptanweisungen
 </script>
...
</body>
```

Wenn Sie im Skript-Tag keine Sprache angeben, wird von allen bekannten Browsern als Standard JavaScript bzw. – im Fall des Internet Explorers – dessen Klon JScript verwendet. Allerdings nutzen Sie dabei das sehr tolerante Verhalten von Browsern aus, denn eigentlich müssen Sie die verwendete Skriptsprache spezifizieren. Über den Parameter `language` oder `type` können Sie festlegen, um welche Skriptsprache es sich handelt[2].

Hier sehen Sie zwei alternative Beispiele:

Listing 3.2: **Alternative Spezifizierungen der Skriptsprache**

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

Während nun bei der Angabe des Werts von `language` Groß- und Kleinschreibung irrelevant ist, müssen Sie bei der Festlegung des MIME-Typs[3] über `type` die Kleinschreibung verwenden. Bei `language` können Sie auch die Version von JavaScript angeben – beispielsweise über die Angabe `JavaScript1.3` oder `JavaScript1.5`. Sie stellen also dem Token *JavaScript* einfach – ohne Leerzeichen – die entsprechende Versionsnummer nach. Das führt dazu, dass Browser, die diese Version noch nicht beherrschen, den Skriptblock ignorieren.

ACHTUNG

Die parallele Verwendung von `type` und `language` sollte unterbleiben. Die Angabe `language` wird grundsätzlich ignoriert, wenn zusätzlich `type` angegeben wird. Damit würden eventuell gewünschte Versionsangaben über `language` wirkungslos.

Früher hatte man im Inneren des Skript-Containers HTML-Kommentare (`<!-- ... -->` oder gar `<!-- ... //-->`) notiert, um Browser, die kein JavaScript kennen, davon abzuhalten, die Anweisungen einfach anzuzeigen. Das ist heutzutage vollkommen unnötig, da solche Browser zuletzt etwa 1996 auf den Markt kamen und damit in der Praxis nicht mehr existent sind. Wohlgemerkt – es geht bei dieser Anmerkung nicht um Browser, bei denen JavaScript nur deaktiviert ist.

Bei der Einbindung von einem `<script>`-Element in eine Webseite stellt sich die Frage, wo es notiert werden darf. Diese Frage ist nicht eindeutig zu beantworten. Im Grunde gehört das Element in den Header einer Webseite.

In der Praxis wird man `<script>`-Elemente jedoch an jeder beliebigen Stelle innerhalb einer Webseite finden. Im Prinzip kann so ein Element sogar hinter der eigentlichen Webseite stehen. Die Skriptanweisungen werden von einem Browser einfach verarbeitet, wenn die Webseite in den Browser geladen und von oben nach unten geparst wird. Entsprechend ist klar, dass unten in einer Webseite notierte Skriptanweisungen (Funktionen) erst dann zur Verfügung stehen, wenn die Seite bis dahin geladen ist. Eine übliche Vorgehensweise ist es deshalb, einen `<script>`-Container vor den Body zu platzieren und darin die Deklarationen von wichtigen Funktionen einzubetten. Diese Funktionen sind dann für das ganze Dokument verfügbar.

2 Offiziell wird die Angabe des MIME-Typs gefordert.

3 Ein MIME-Typ legt die Art eines Inhalts fest. Dabei wird zuerst eine Hauptkategorie wie *text* oder *image* angegeben und dann – durch einen Slash abgetrennt – eine Unterkategorie wie *html*, *css* oder *javascript*.

Nun hat sich in der Praxis gezeigt, dass es in der Programmierung mit JavaScript zahlreiche Situationen gibt, in denen die Position des `<script>`-Elements die Ausführung massiv beeinflusst. Viele Webseiten erzwingen beispielsweise, dass `<script>`-Container nicht im Header, sondern in anderen Abschnitten der Webseite platziert werden. Selbst hochprofessionelle und moderne Konzepte wie Projekte von Google oder Microsoft oder auch AJAX-Frameworks müssen dies ausnutzen, um bestimmte Verhaltensweisen von einigen Browsern zu kompensieren.

HINWEIS

Erinnern Sie sich an die Methode `ready()` aus den letzten Beispielen? Diese kompensiert genau diese Probleme und macht so eine Trickserei nicht mehr notwendig.

Grundsätzliches zu externen JavaScript-Dateien

Mit der eben beschriebenen Vorgehensweise notieren Sie die Skripte direkt in die (X)HTML-Datei. Für die Verwendung von JavaScript ist in der Regel jedoch eine Auslagerung in eine externe Datei vorziehen. Zumindest dann, wenn es sich um Unterprogramme (Funktionen) und Variablendeklarationen handelt. Dann bildet die externe Datei mit der Funktions- und Variablensammlung eine Bibliothek, die Sie in beliebige Webseiten einbinden können. Das sehen wir ja auch explizit bei der Referenz auf die jQuery-Bibliothek. Vor allem wird über diese Vorgehensweise die Trennung der Struktur und Funktionalität erreicht, was für die Praxis gar nicht hoch genug bewertet werden kann. Die externe JavaScript-Datei (in der Regel mit der Erweiterung *.js*) enthält im optimalen Fall Ihren gesamten JavaScript-Code mit Ausnahme der expliziten Aufrufe von JavaScript-Funktionen und wenigen Anweisungen.

Um eine externe JavaScript-Datei in eine Webseite einzubinden, müssen Sie den `<script>`-Tag nur um das Attribut `src` erweitern. Damit geben Sie die URL der externen JavaScript-Daten an. Dabei gelten beim Referenzieren von separaten JavaScript-Dateien die üblichen Regeln für URLs.

Die Referenz auf die jQuery-Bibliothek

Die jQuery-Bibliothek ist nun eine externe JavaScript-Datei. Nachfolgend sehen Sie drei alternative Codefragmente mit zwei Varianten zur Referenz auf eine externe JavaScript-Datei, die die jQuery-Bibliothek referenzieren:

Listing 3.3: **Einbindung der externen jQuery-JavaScript-Dateien**

```
01 <script language="JavaScript"
     src="jquery-1.5.min.js"></script>
02 <script language="JavaScript"
     src="lib/jquery-1.5.min.js"></script>
03 <script type="text/javascript"
     src="lib/jquery-1.5.min.js"></script>
```

ACHTUNG

Der `<script>`-Container sollte auf jeden Fall leer sein (auch keine Leerzeichen oder Zeilenumbrüche enthalten), denn einige Browser machen Probleme, wenn darin Inhalt vorhanden ist (rein formal darf darin auch gar nichts stehen – es handelt sich um ein leeres Element). Andere Browser ignorieren Inhalt darin nach dem Prinzip der Fehlertoleranz. Aber die XML-Syntax mit der Angabe eines leeren Elements `<script ... />` dürfen Sie nicht verwenden, da beispielsweise der Internet Explorer hier aussteigt (obwohl die Syntax vollkommen legitim wäre).

Im ersten Fall befindet sich die JavaScript-Datei im gleichen Verzeichnis wie die referenzierende Webseite, im zweiten Fall in einem Unterverzeichnis *lib* vom Verzeichnis der Webseite aus gesehen. In beiden Fällen spezifiziert `language` den Typ des referenzierten Skripts.

In Fall drei wird wie in Fall zwei die jQuery-Bibliothek aus dem Unterverzeichnis *lib* eingebunden, nur der Typ des referenzierten Skripts wird mit `type` angegeben.

Die Referenz auf die jQuery-Bibliothek sollten Sie normalerweise in den Header der Webseite schreiben. Nur wird in den offiziellen Dokumentationen zu jQuery darauf hingewiesen, dass man das unter manchen Umständen vermeiden muss! Sie sehen auch hier, dass – wie oben schon beschrieben – der Bruch der theoretischen Regeln manchmal notwendig ist.

Konkret geht es um den Fall, wenn die gesamte Seite mit der Referenz auf jQuery via der PHP-Funktionen `include()` oder `require()` in ein PHP-Skript eingebunden wird. In der Dokumentation heißt es ganz lapidar: »because calling jQuery script file from inside the <head> tag doesn´t work for some reason«. Mit anderen Worten – auch im jQuery-Team haben sie keine Ahnung, warum das manchmal nicht geht. Aber das ist einfach Webprogrammierung. Durch die extrem unterschiedlichen Verhaltensweisen von Browsern oder auch bei serverseitigen Prozessen wie der Arbeit eines PHP-Interpreters muss man einfach immer wieder tricksen, Regeln brechen, Workarounds einsetzen etc. Google ist ein Meister beim Finden solcher Praxislösungen, die sämtlichen offiziellen Lehrmeinungen spotten. Aber eben auch andere Projekte stellen einfach die praktische Funktionalität in den Fokus. Natürlich kann man nicht vorhersehen, ob dieser Trick zur Verlagerung der Referenz aus dem Header heraus in allen Versionen von jQuery und/oder PHP beziehungsweise in jedem Browser notwendig ist. Aber wenn ein Problem auftritt, sollten Sie die Einbindung einfach in den Körper der Webseite verlagern und es dann noch einmal neu probieren. Webprogrammierung ist einfach Try and Error. Davor schützt auch die Verwendung eines ausgereiften Frameworks nicht vollständig.

3.3 AJAX und XMLHttpRequest (XHR)

Wie oben schon erwähnt, beschreibt man mit AJAX ein Verfahren, wie man eine Reaktion einer Webapplikation in (nahezu) Echtzeit gewährleisten kann, obwohl neue Daten vom Webserver angefordert werden. Nun bezeichnet AJAX ganz konkret aber nur das Zusammenspiel von lange etablierten Technologien, beginnend bei HTML bzw. XHTML und HTTP über JavaScript und CSS bis hin zu XML oder JSON. Auch die asynchrone Nachforderung von

Daten, die in eine Webseite integriert werden sollen, gibt es im Prinzip schon seit ungefähr 1998. Nur der Begriff AJAX ist relativ neu und hat sich etwa um das Jahr 2005 zusammen mit dem Buzzword Web 2.0 eingebürgert.

AJAX hätte im Grunde als eine Erweiterung des JavaScript-Objektmodells eingeführt werden können. Mit so einer banalen Marketingstrategie wäre nach meiner Meinung das Konzept zwar niemals so erfolgreich wie mit dem Buzzword AJAX geworden. Aber rein vom Konzept her könnte man AJAX so beschreiben.

Moderne Browser bieten nun zur Unterstützung dieser asynchronen Kommunikation mit dem als Erweiterung des JavaScript-Objektmodells aufgenommenen Objekt `XMLHttpRequest` eine eingebaute Schnittstelle zur Kontrolle von HTTP-Transaktionen aus clientseitigen Programmiersprachen (hauptsächlich JavaScript), die unabhängig von der »normalen« Datenanforderung des Webbrowsers ablaufen. Diese XHR- bzw. `XMLHttpRequest`-Objekte sind damit unmittelbar an dem internen Aufbau von HTTP orientiert und das Rückgrat jeder AJAX-Anfrage. Und sie werden in allen Frameworks und Toolkits, die sich mit dem Titel AJAX schmücken, in irgendeiner Form zur Verfügung gestellt; natürlich auch jQuery mit einigen sehr komfortablen Methoden und sogar seit jQuery 1.5 einer Erweiterung des Objekts selbst!

Für die asynchrone Kommunikation zwischen Browser und Webserver erlaubt das Objekt mit Funktionsreferenzen die Registrierung von sogenannten Callback-Funktionen, die bei jeder Änderung des Transaktionszustands ausgewertet werden.

Zudem kann über ein `XMLHttpRequest`-Objekt auf alle HTTP-Header-Felder von einer AJAX-Anfrage als auch -Antwort zugegriffen werden.

Nun stehen im Fokus der Datenanforderung von AJAX neben Klartext XML- und JSON-Daten. Ich möchte nicht einfach voraussetzen, dass Sie sich damit bereits auskennen, und zumindest die wichtigsten Details kurz erläutern.

XML

XML beschreibt einen plattformneutralen Klartextstandard auf Unicode-Basis zur Erstellung maschinen- und menschenlesbarer Dokumente, um darüber beliebige Informationen auszutauschen. XML-Dokumente liegen in Form einer Baumstruktur vor, die eine Navigation zu den einzelnen Zweigen des Baums gestattet. Dabei ist XML wie HTML eine Auszeichnungssprache (Markup Language), um über die Textinformation hinaus eine Struktur der Information zu bieten. Die in einem Dokument enthaltenen Informationen werden dazu durch Tags strukturiert, die sowohl in HTML als auch XML in spitze Klammen notiert werden. Die Elemente in XML sind im Gegensatz zu HTML aber nicht vorgegeben. Es gibt keinen vorgefertigten, beschränkten Sprachschatz an Elementen. Es gibt keine vorgegebenen XML-Tags in dem Sinne, wie die meisten Anwender HTML-Tags kennen. XML besteht lediglich aus einer Beschreibung der Syntax für Elemente und Strukturen. Damit ist XML beliebig erweiterbar. Die XML-Spezifikation beschreibt lediglich, nach welchen Regeln Tags zu definieren sind.

Im Gegensatz zu HTML ist XML syntaktisch eine sehr strenge Sprache, bei der es kein Prinzip der Fehlertoleranz gibt (das ja bei der Verwendung von HTML ein zentraler Aspekt ist). Die XML-Spezifikation ist streng formal und lässt keine Ausnahmen und unklaren Strukturen zu. Dadurch ist XML jedoch einfach und automatisiert zu validieren und auszuwerten. Das prädestiniert XML für einen Datenaustausch.

XML beschreibt nur wenige, einfache, aber eben sehr strenge und absolut eindeutige Regeln, nach denen ein Dokument zusammengesetzt sein kann.

Sogenannte Komponenten bilden die Bausteine eines XML-Dokuments. Die Grundstruktur eines XML-Dokuments besteht dabei aus Elementen, die – sofern sie nicht eingeschränkt werden – selbst Unterelemente enthalten können und die wichtigste Form von Komponenten darstellen. Elemente selbst sind so aufgebaut, wie man es im Wesentlichen von HTML kennt. Es gibt einen Anfangs-Tag, der beispielsweise so aussieht:

Listing 3.4: **Ein Anfangs-Tag in XML**

```
<rjs>
```

Der Anfangs-Tag muss in XML immer mit einem Ende-Tag abgeschlossen werden, der hinter einem Slash den Bezeichner wiederholt – es sei denn, ein Tag wird als leeres Element gekennzeichnet. Eventuell im Anfangs-Tag notierte Attribute werden auf keinen Fall im Ende-Tag wiederholt. In unserem Fall sieht der Ende-Tag so aus:

Listing 3.5: **Ein Ende-Tag in XML, der zu dem Anfangs-Tag passt**

```
</rjs>
```

Sie sollten das alles von HTML kennen, nur sind da die Elemente nicht frei wählbar.

Im Inneren des Elements kann – sofern nicht besondere Regeln vorgegeben sind – beliebiger Inhalt notiert werden. Das können weitere Elemente oder Text sein. Auch gemischte Formen sind erlaubt.

XML erlaubt die Deklaration von leeren Elementen. Diese werden meist in Verbindung mit Attributen verwendet. Ein leeres Element wird wie folgt gekennzeichnet:

Listing 3.6: **Ein leeres Element mit Attribut**

```
<rjs url="www.rjs.de" />
```

Alternativ geht auch diese Form:

Listing 3.7: **Eine alternative Schreibweise für ein leeres Element**

```
<rjs url="www.rjs.de"></rjs>
```

Dabei darf nicht einmal ein Leerzeichen im Container notiert werden. Leerraum wird in XML in der Regel als echter Inhalt und damit als ein Knoten im Baum angesehen. Leider beachten das bestimmte Browser bzw. die darunter liegenden XML-Parser nicht.

Ein XML-Dokument muss wenige, aber strenge syntaktische Regeln einhalten. Wenn ein Dokument diese Regeln einhält, nennt man es wohlgeformt bzw. well-formed.

XML-Dokumente bestehen grundsätzlich aus Unicode-Zeichen (16-Bit-Zeichensatz). Allerdings können Sie bei der Erstellung Ihres XML-Dokuments in einem Editor ANSI-Code speichern. Dann interpretiert der Parser das als Unicode.

In XML wird grundsätzlich zwischen Groß- und Kleinschreibung streng unterschieden.

Ein XML-Dokument beginnt immer mit einem sogenannten Prolog. Der Prolog muss am Beginn eines XML-Dokuments stehen und sieht derzeit in der einfachsten Form so aus:

Listing 3.8: **Ein einfacher XML-Prolog**

```
<?xml version="1.0" ?>
```

Ein XML-Dokument darf nur genau ein Wurzelelement (auch Root-Tag oder Dokumentelement genannt) besitzen. So ein Wurzelelement muss auch auf jeden Fall vorhanden sein und folgt, abgesehen von Kommentaren, unmittelbar dem Prolog.

Elemente müssen sauber verschachtelt werden.

Jedes Element muss einen Ende-Tag haben oder als leeres Element notiert werden.

Attributen muss immer ein Wert zugewiesen werden, und der Attributwert muss zwingend in Anführungszeichen stehen.

Beispiel einer XML-Datei (*rjs.xml*):

Listing 3.9: **Eine typische XML-Datei**

```
01 <?xml version="1.0" encoding="utf-8"?>
02 <rjs>
03   <geschaeftlich>
04     <name beruf="Dipl Math" firma="RJS EDV-KnowHow">
05     Ralph Steyer</name>
06     <ort>Eppstein</ort>
07     <ort>Bodenheim</ort>
08   </geschaeftlich>
09   <webseiten>
10     <url>www.rjs.de</url>
11     <url>blog.rjs.de</url>
12     <url>www.ajax-net.de</url>
13   </webseiten>
14 </rjs>
```

XML-Dateien können Sie sich von jedem Browser anzeigen lassen. Meist stellt er die Datei in einem Baum dar. Dessen Zweige können Sie in der Regel durch das Anklicken von sensitiven Elementen des Baums expandieren und kollabieren.

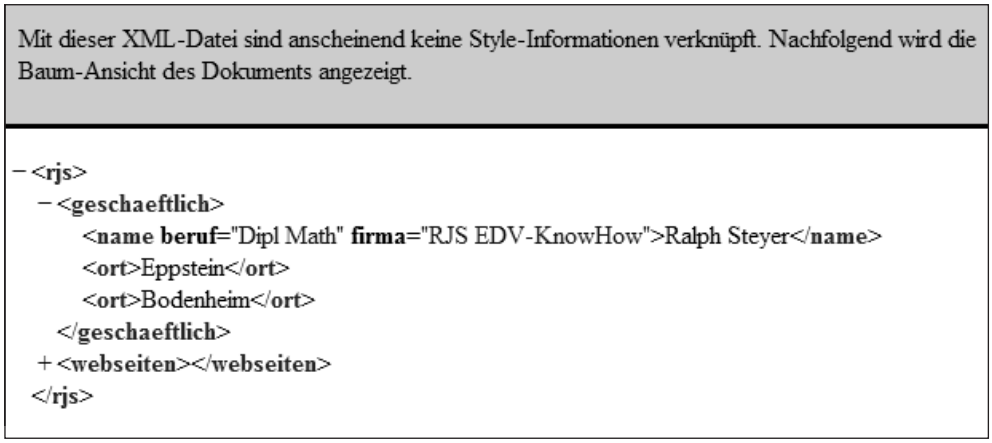

Abbildung 3.1: **Die meisten Browser stellen eine XML-Datei als einen interaktiven Baum dar – der untere Zweig des Baums ist kollabiert.**

HINWEIS

Um die Sache prägnant zu fassen – jede XML-Datei lässt sich als DOM auffassen. Und damit haben Sie alle DOM-Methoden und -Eigenschaften samt jQuery-Methoden zur Verfügung. Diese Aussage ist Ihnen entweder klar, wenn Sie das DOM-Konzept schon kennen, oder aber wird gleich beim Abschnitt zum DOM-Konzept vertieft.

JSON

XML erlaubt im Fall einer Datennachforderung per AJAX die Verlagerung von sehr viel Logik in das Übertragungsformat, ist aber sehr sperrig und oft für AJAX-Anfragen überdimensioniert. Zudem gibt es mit der Verwertung in verschiedenen Browsern diverse Probleme, obgleich jQuery diese Probleme weitgehend kompensiert. Dennoch – wenn nun aber reiner Klartext zu wenig Logik bietet und XML zu komplex und schwergewichtig ist, gibt es eine Alternative zwischen den beiden Extremen. Mit JSON steht Ihnen eine Struktur für das Übertragungsformat zur Verfügung, die einfacher als XML ist und vor allen Dingen in verschiedenen Webbrowsern viel konsistenter verarbeitet wird.

JSON ist wie XML ein maschinenlesbares Klartextformat zum Datenaustausch, das zwar bedeutend weniger flexibel als etwa XML ist, aber viel kompakter. Und JSON beinhaltet direkt an JavaScript orientierte Datenstrukturen bzw. Datentypen (Objekte, Arrays, Zeichenketten, Zahlen, die booleschen Werte `true` und `false` sowie den Wert `null`), die verschachtelt und beliebig durch Whitespace-Zeichen strukturiert werden können.

Das Konzept von JSON baut auf zwei zentralen Strukturen auf:

» Name-Wert-Paare. Solche Kombinationen findet man in fast allen modernen Programmiersprachen. Sie werden dann meist als Objekt (Object), Satz (Record), Struktur (Struct), Wörterbuch bzw. Verzeichnis (Dictionary), Hash-Tabelle (Hash Table), Schlüsselliste (Keyed List) oder als ein assoziatives Array (Associative Array) implementiert.

» Eine geordnete Werteliste, wie sie in den meisten Sprachen vorkommt. Die Implementierung erfolgt je nach Programmiersprache über Arrays (Array), Vektoren (Vector), Listen (List) oder Sequenzen (Sequence).

Beim Grundkonzept von JSON handelt es sich also um die Beschreibung von universellen Datenstrukturen, die von so gut wie allen modernen Programmiersprachen in der einen oder anderen Form unterstützt werden. Mit JSON steht damit ein Datenformat zum Austausch zwischen Programmiersprachen zur Verfügung, das auf diesen Strukturen aufbaut. Insbesondere kann JSON somit von diversen Programmiersprachen unmittelbar verarbeitet werden; in JavaScript etwa durch die Funktion `eval()`.

Die konkrete Syntax von Formatdefinitionen ist in JSON – wie der Name schon deutlich macht – an JavaScript angelehnt, wobei es keine Bezeichner für Objekte oder Datenfelder gibt:

» Eine Objektdefinition beginnt mit der geschweiften Klammer `{` und endet mit der geschweiften Klammer `}`. So ein Objektblock kann eine durch Kommata separierte, ungeordnete Liste von Eigenschaften enthalten.

» Eine Eigenschaft wiederum besteht aus einem Schlüssel und einem Wert. Beide werden durch einen Doppelpunkt getrennt. Der Schlüssel ist dabei eine Zeichenkette und ein zugehöriger Wert ein beliebiger Datentyp (ein Objekt, ein Array, eine Zeichenkette, eine Zahl oder einer der Ausdrücke `true`, `false` oder `null`).

» Ein Array in JSON beginnt einfach mit der eckigen Klammer `[` und endet mit der eckigen Klammer `]`. Auch ein Array kann eine durch Kommata separierte, hier aber geordnete Liste von Werten enthalten.

» Eine Zeichenkette in JSON beginnt und endet wie üblich mit den doppelten Anführungszeichen `"`. Sie kann beliebige Unicode-Zeichen und Escape-Sequenzen enthalten. Eine Zeichenkette in JSON ist einer Zeichenkette in C oder Java sehr ähnlich.

» Eine Zahl ist einfach eine Folge der Ziffern 0–9. Dabei kann man Vorzeichen und Dezimalpunkte sowie einen Exponenten über e oder E verwenden. Auch eine Zahl in JSON ist einer Zahl in C oder Java sehr ähnlich. Es gibt aber einen wesentlichen Unterschied. In JSON kann keine oktale oder hexadezimale Darstellung einer Zahl verwendet werden.

» Ein boolescher Wert wird wie üblich durch die Ausdrücke `true` bzw. `false` repräsentiert.

Nachfolgend finden Sie ein Beispiel für ein einfaches JSON-Dokument, dessen Informationsgehalt und strukturelle Basis[4] der vorangehenden XML-Datei entspricht (*rjs.json*):

Listing 3.10: **Eine JSON-Struktur**

```
01 {
02 "geschaeftlich" : {
03   "name":"Ralph Steyer",
```

4 Die Struktur ist nicht zu 100% identisch, aber weitgehend. Die Attribute in der XML-Datei hätte man etwas anders abbilden müssen, um die Struktur zu 100% deckungsgleich zu bekommen. Aber diese Feinheiten sollen uns nicht interessieren.

```
04    "firma":"RJS EDV-KnowHow",
05    "beruf":"Dipl Math",
06    "ort":["Eppstein","Bodenheim"]
07   } ,
08   "webseiten" : {
09     "url":["www.rjs.de","blog.rjs.de","www.ajax-net.de"]
10   }
11 }
```

Das Schöne an JSON ist, dass die Sprache wirklich einfach und mit diesen knappen Erklärungen bereits fast vollständig erklärt ist. Weitere Informationen finden Sie bei Bedarf aber unter `http://json.org/json-de.html`.

Vertiefung zur Verwertung von JSON für JavaScript-Profis

Für diejenigen Leser, die sich mit JavaScript gut auskennen, soll mit dem folgenden Beispiel die grundsätzliche Verwertung von JSON unter JavaScript demonstriert werden. Dazu werden wir einmal den Fall betrachten, dass wir bereits in JavaScript ein JSON-Objekt vorliegen haben, und zudem den Fall, dass wir einen String erhalten, der zwar eine JSON-Struktur enthält, den wir aber erst in ein JSON-Objekt umwandeln müssen. Letzteres ist beispielsweise der Fall, wenn wir per AJAX JSON-Daten anfordern (was wir hier aber nicht machen).

ACHTUNG

Ohne entsprechende Grundlagen in JavaScript werden Sie das Beispiel nur schwer nachvollziehen können. Aber lassen Sie sich in dem Fall nicht abschrecken – es ist ja gerade ein Vorteil von einem Framework wie jQuery, dass Sie damit nicht so tief auf JavaScript-Ebene arbeiten müssen. Wenn Ihnen also die JavaScript-Grundlagen fehlen oder Ihnen das Listing eher Sorgen macht, ignorieren Sie die Passage.

Beispiel (*kap3_1.html*):

Listing 3.11: **Die HTML-Seite, in der JSON-Daten angezeigt werden sollen**

```
...
06    <script type="text/javascript"
07       src="lib/kap3_1.js">
08    </script>
09  </head>
10  <body>
11    <h1>Verwerte JSON</h1>
12    <script type="text/javascript">
13      verwerte_json_text();
14      verwerte_json_objekt();
15    </script>
16  </body>
17 </html>
```

Die HTML-Seite ist harmlos. Außer der Referenz auf die externe JavaScript-Datei und die Aufrufe der dort deklarierten Funktionen `verwerte_json_text()` und `verwerte_json_objekt()` in den Zeilen 13 und 14 passiert da nichts von Interesse. Die beiden Funktionen werden wir einfach direkt in die Webseite schreiben.

Schauen wir uns die referenzierte JavaScript-Datei *kap3_1.js* an:

Listing 3.12: **Die referenzierte JavaScript-Datei**

```
01 function verwerte_json_text(){
02   var JSONText = '{ "geschaeftlich" : { "name" :
     "Ralph Steyer", "firma" : "RJS EDV-KnowHow", "beruf" :
     "Dipl Math", "ort" : ["Eppstein","Bodenheim"] } ,
     "webseiten" :  { "url" :["www.rjs.de","blog.rjs.de",
     "www.ajax-net.de"]  }}';
03  var JSONObjekt = eval('(' + JSONText + ')');
04  ausgabe_alles(JSONObjekt);
05  ausgabe_gezielt(JSONObjekt, name);
06 }
07 function verwerte_json_objekt(){
08  var JSONObjekt = {
09    "geschaeftlich": {
10      "name": "Ralph Steyer",
11      "firma": "RJS EDV-KnowHow",
12      "beruf": "Dipl Math",
13      "ort": ["Eppstein", "Bodenheim"]
14    },
15    "webseiten": {
16      "url":
17       ["www.rjs.de", "blog.rjs.de", "www.ajax-net.de"]
18    }
19  };
20  ausgabe_alles(JSONObjekt);
21  ausgabe_gezielt(JSONObjekt);
22 }
23 function ausgabe_alles(JSONObjekt){
24  document.write(
25     "<table><tr><th>Schlüssel</th><th>Wert</th></tr>");
26  for (i in JSONObjekt.geschaeftlich)
27    document.write("<tr><td>" + i + "</td><td>" +
28       JSONObjekt.geschaeftlich[i] + "</td></tr>");
29  for (i in JSONObjekt.webseiten)
30    document.write("<tr><td>" + i + "</td><td>" +
31       JSONObjekt.webseiten[i] + "</td></tr>");
32  document.write("</table>");
33 }
34 function ausgabe_gezielt(JSONObjekt){
35  document.write(
36     "<table><tr><th>Schlüssel</th><th>Wert</th></tr>");
37  document.write("<tr><td>Name</td><td>" +
38     JSONObjekt.geschaeftlich.name + "</td></tr>");
39  document.write("<tr><td>Firmen-Webseite</td><td>" +
40     JSONObjekt.webseiten.url[0] + "</td></tr>");
41  document.write("<tr><td>Blog</td><td>" +
42     JSONObjekt.webseiten.url[1] + "</td></tr>");
43  document.write("</table>");
44 }
```

In den Funktionen in dem Beispiel wird einmal aus einem String, der eine JSON-Struktur entsprechend der Datei *rjs.json* enthält (Zeile 2 – beachten Sie, dass der String im Editor

unbedingt in einer Zeile stehen muss, anderenfalls muss er über eine Stringverkettung aufgeteilt und verbunden werden), über die eingebaute JavaScript-Funktion `eval()` ein JSON-Objekt erzeugt (Zeile 3 – `var JSONObjekt = eval('(' + JSONText + ')');`). In der zweiten Funktion finden Sie die direkte Deklaration des JSON-Objekts in den Zeilen 8 bis 19 über ein Array-Literal. Beide Funktionen verhalten sich in der Folge gleich – es werden zwei Ausgabefunktionen aufgerufen, die auf dem JSON-Objekt operieren. Das per `eval()` aus dem String erzeugte JSON-Objekt ist vollkommen identisch mit dem nativ per Array-Literal angelegten Objekt.

Verwerte JSON

Schlüssel	Wert
name	Ralph Steyer
firma	RJS EDV-KnowHow
beruf	Dipl Math
ort	Eppstein,Bodenheim
url	www.rjs.de,blog.rjs.de,www.ajax-net.de

Schlüssel	Wert
Name	Ralph Steyer
Firmen-Webseite	www.rjs.de
Blog	blog.rjs.de

Schlüssel	Wert
name	Ralph Steyer
firma	RJS EDV-KnowHow
beruf	Dipl Math
ort	Eppstein,Bodenheim
url	www.rjs.de,blog.rjs.de,www.ajax-net.de

Schlüssel	Wert
Name	Ralph Steyer
Firmen-Webseite	www.rjs.de
Blog	blog.rjs.de

Abbildung 3.2: **Die Verwertung von JSON-Daten per JavaScript**

Mit den Funktionen wird in beiden Fällen dynamisch eine HTML-Tabelle geschrieben. In der Funktion `ausgabe_alles()` wird mit einer Schleife über das JSON-Objekt iteriert, und via einer Mischung aus Punktnotation und Arraynotation werden der Schlüssel und der enthaltene Wert ausgegeben. Das Schöne an der Verwertung von JSON-Objekten unter JavaScript ist, dass die JSON-Strukturen als verschachtelte Objekte abgebildet werden. Gibt es auf

einer Ebene der Objekthierarchie mehrere Objekte eines Typs, werden diese als Objektfelder zur Verfügung gestellt. Einfacher und logischer kann man kaum solche Strukturen verfügbar machen.

In der Funktion `ausgabe_gezielt()` werden gezielt Werte aus der Struktur angesprochen.

3.4 DOM und Objekte

Die bisherigen Beispiele mit JavaScript haben Sie bereits mit Objekten konfrontiert. JavaScript ist eine teilweise objektorientierte Sprache. Genau genommen nennt man sie objektbasierend, denn es fehlen einige Fähigkeiten, die für eine echte objektorientierte Sprache unabdingbar sind.

Unter einem Objekt stellt man sich in der Programmierung ein Softwaremodell vor, das ein Ding aus der realen Welt mit all seinen Eigenschaften und Verhaltensweisen beschreiben soll. Zum Beispiel sind ein Drucker, ein Bildschirm oder die Tastatur ein Objekt. Auch Teile der Software selbst können ein Objekt sein. Der Browser beispielsweise oder ein Teil davon – zum Beispiel ein Frame, die Statuszeile des Browsers oder das Browserfenster oder nur ein Teil eines Dokumentes –, eine Überschrift in einer Webseite, ein Absatz, eine Grafik usw. Eigentlich ist in dem objektorientierten Denkansatz alles als Objekt zu verstehen, was sich eigenständig erfassen, beschreiben und ansprechen lässt.

Objektorientierte Programmierung lässt sich darüber definieren, dass zusammengehörende Anweisungen und Daten eine zusammengehörende, abgeschlossene und eigenständige Einheit bilden (ein Objekt), die Eigenschaften und Methoden (Fähigkeiten) bereitstellt und sich in einem gewissen Zustand befinden kann. Eigenschaften und Methoden bilden die sogenannten Member eines Objekts.

Wesentlich an der objektorientierten Programmierung ist, dass die Methoden und die Eigenschaften (Attribute oder Daten) immer gemeinsam einem Objekt zugeordnet sind. Dies bedeutet weiterhin, dass es keine Methoden oder Eigenschaften ohne ein zugehöriges Objekt gibt.

Klassenelemente sind nun spezielle Eigenschaften und Methoden, die nicht über ein spezifisches Objekt, sondern bereits über die Klasse verfügbar sind. Man kann sie verwenden, ohne vorher ein Objekt zu erzeugen.

3.4.1 DOM und der Zugriff auf Elemente einer Webseite

Es wird nun **oft** vorkommen, dass Sie in JavaScript selbst ein Objekt erzeugen oder Klassenelemente verwenden. So gut wie **immer** werden Sie in Ihren JavaScript-Codes Objekte nutzen, die Ihnen automatisch vom Browser zu Verfügung gestellt werden. Und diese Objekte basieren auf einem Objektmodell, das nicht zu JavaScript oder auch (X)HTML zählt, sondern die Strukturen von weitgehend beliebigen baumartig aufgebauten Dokumenten beschreibt – das schon mehrfach erwähnte DOM-Konzept (Document Object Model). Oder um es noch drastischer zu sagen – ohne die Verwendung des DOM ist JavaScript-Programmierung nutzlos!

Diese Objektbibliothek kann mittels diverser Techniken genutzt werden, sowohl aus Programmier- und Skriptsprachen als auch aus Anwendungen heraus. Dem Zugriff auf eine Webseite unter diesem Objektgesichtspunkt liegt mit DOM ein Konzept zugrunde, das eine plattform- als auch programmiersprachenübergreifende Schnittstelle bezeichnet.

In diesem Konzept wird eine (X)HTML-Seite (oder allgemein ein baumartig aufgebautes Dokument – z.B. auch ein XML-Dokument) nicht als statisch aufgebaute, fertige und nicht unterscheidbare Einheit, sondern als differenzierbare Struktur betrachtet, deren einzelne Bestandteile Programmen und Skripten dynamisch zugänglich sind. Dieser Ansatz ermöglicht die individuelle Behandlung von Bestandteilen einer Webseite auch dann, wenn die Webseite bereits in den Browser geladen ist. Und zwar eine Behandlung, die weit über die einfache Interpretation durch den Browser beim Laden eines Dokuments von oben nach unten hinausgeht.

Das DOM-Konzept beinhaltet verschiedene Teilaspekte. Es veranlasst beispielsweise einen Browser, eine (X)HTML-Seite zwar wie eine gewöhnliche Textdatei zu lesen und entsprechende (X)HTML-Anweisungen auszuführen. Darüber hinaus wird der Browser jedoch beim Laden der Webseite alle ihm im Rahmen des Konzepts bekannten und einzeln identifizierbaren Elemente einer Webseite bezüglich ihres Typs, ihrer relevanten Eigenschaften und ihrer Position innerhalb der Webseite indizieren.

Dies ist eine Art Baum im Hauptspeicher des Rechners, der beim Laden der Webseite aufgebaut und beim Verlassen der Seite wieder gelöscht wird. Die Elemente in dem Baum werden als Knoten bezeichnet.

Ähnliche Elemente werden dabei bei der Indizierung vom Browser gemeinsam in einem Feld verwaltet. Auf diese Weise hat der Browser nach dem Laden der Webseite genaue Kenntnis über alle relevanten Daten sämtlicher eigenständig für ihn ansprechbarer Elemente in der Webseite. Welche das jedoch sind und was er damit anstellen kann, das kann sich je nach Browser erheblich unterscheiden.

Jedes ansprechbare Element (etwa ein bestimmter (X)HTML-Tag) kann bei Bedarf auch während der Lebenszeit der Webseite aktualisiert werden. Etwa wenn mittels eines Skripts die Position eines Elementes in der Webseite verändert oder über Style Sheets nach dem vollständigen Laden der Webseite das Layout eines Elementes dynamisch verändert wird. Das haben wir ja im letzten Kapitel in den Beispielen gemacht.

Viele Objekte im DOM-Konzept sind in Form einer Objekthierarchie verfügbar. Wenn ein Objekt einem anderen untergeordnet ist, notiert man das in der DOT-Notation, indem man erst den Namen des oberen Objekts und dann den des darunter angesiedelten Objekts notiert.

Wenn man beispielsweise eine Webseite nimmt, ist sie über das Objekt `document` aus JavaScript heraus verfügbar. Da sich die Webseite in einem Browserfenster befindet und dieses als `window` ansprechbar ist, erfolgt der Zugriff über `window.document`.

Über diese streng entlang der Verschachtelung der Objekte orientierten Zugriffsmöglichkeiten auf DOM-Objekte hinaus gibt es weitere syntaktische Weg, die aber letztendlich alle das gleiche Objekt referenzieren und unabhängig von der Art des Zugriffs immer die gleichen Eigenschaften und Methoden bereitstellen. So gibt es zum Zugriff die Methoden `getElementById()`, `getElementsByTagName()` und `getElementsByName()` sowie die direkte Angabe des Namens. Aber wie wir schon besprochen haben, gibt es bei allen Standardmethoden zum Zugriff auf den DOM gewisse Probleme, und jQuery vereinheitlicht den Zugriff über eine übergeordnete Syntax wie `$()` und diverse andere Techniken.

TIPP

Im Anhang finden Sie eine Auflistung der verfügbaren DOM-Objekte.

3.5 Style Sheets und DHTML

Bei vielen Webseiten ist es ein zentraler Aspekt, dass eine Webseite verändert wird, nachdem sie im Browser geladen wurde. Und das ist im Grunde genau die Definition von DHTML. Wobei DHTML oft auch als die Verbindung von HTML bzw. XHTML, JavaScript und Style Sheets bezeichnet wird.

Formatvorlagen bzw. Style Sheets bestehen aus reinem Klartext, der Regeln zur Formatierung von Elementen beschreibt. Moderne Webseiten reduzieren die Bedeutung von HTML und XHTML fast vollständig auf die reine Strukturierung der Seite, während das Layout gänzlich Style Sheets übertragen wird. Insbesondere eröffnen Style Sheets die Möglichkeit, das Vermischen von Gestaltungsbefehlen und Informationsträgern aufzuheben. Es kann eine klare Trennung von Struktur und Layout erreicht werden. Und Sie können mittels JavaScript und Style Sheets eben auch gezielt Layout- oder Positionseigenschaften eines Elements der Webseite verändern, nachdem sie bereits in den Browser geladen wurde.

3.5.1 CSS – die Standardsprache des Webs

Style Sheets bezeichnen keine eigene Sprache, sondern nur ein Konzept. Und es gibt nicht nur eine einzige Style-Sheet-Sprache, sondern diverse Ansätze bzw. verschiedene Sprachen. Die genauen Regeln und die syntaktischen Elemente für die Style Sheets werden je nach verwendeter Sprache etwas differieren, aber oft ähnlich aussehen. Im Web kommen derzeit hauptsächlich die sogenannten **CSS** (Cascading Style Sheets) zum Einsatz.

Allgemein liegen bei einer Anwendung von Style Sheets Daten in einer Rohform oder einer nicht gewünschten Darstellungsweise vor, die auf spezifische Weise verändert werden soll. Die Darstellung der Daten erfolgt dann in einer anderen Form, wobei die Informationen selbst meist erhalten bleiben. Unter Umständen werden allerdings im Ausgabedokument Daten der Quelle unterdrückt und/oder durch Zusatzdaten ergänzt. Die Beschreibung der Transformation bzw. Formatierung erfolgt in der Regel in Form einer externen Datei, kann aber auch in einigen Situation direkt in die Datei mit den Daten notiert werden (etwa eine

Webseite). Style Sheets geben vorhandenen Informationen also einfach ein neues Aussehen. Dazu werden die Daten und die Formatierungsinformationen von einem interpretierenden System zu einer neuen Darstellung verarbeitet.

Stilregeln können in Webseiten im Grunde auf beliebige HTML- beziehungsweise XHTML-Elemente angewendet werden. Es gibt jedoch einige Elemente, die sich besonders nachdrücklich für die Anwendung von Style Sheets eignen und manchmal sogar ihren einzigen Nutzen daraus beziehen, dass sie mit Style Sheets formatiert werden.

Style Sheets bestehen allgemein aus verschiedenen Regeln zur Formatierung von Elementen. Um Style Sheets wie CSS in einer Webseite zu verwenden, müssen Sie diese einer HTML-, XHTML- oder XML-Seite hinzufügen. Dies kann darüber geschehen, dass Sie Style Sheets in ein Dokument einbetten oder aus einer externen Datei importieren. Ob und wie das geht, hängt von dem Typ der Daten ab, auf die eine CSS-Formatierung angewendet werden soll.

Die Einbettung eines internen Style Sheets in eine Webseite erfolgt über den `<style>`-Tag, einen reinen HTML-Container. In dessen Inneren werden alle Stilregeln definiert. In HTML kann so ein Style-Bereich an jeder Stelle der Webseite notiert werden.

Ein Style Sheet wird bei HTML schematisch wie folgt eingebunden:

Listing 3.13: **Schema für einen Style-Container in HTML**

```
<style type="text/css">
  ... irgendwelche CSS-Formatierungen ...
</style>
```

HINWEIS

Oft findet man im Inneren des Containers HTML-Kommentare. Sie sind aber – wie im Fall der Einbindung von JavaScript – unnötig.

Eine Referenz auf ein externes Style Sheet notieren Sie in einer Webseite mithilfe des `<link>`-Tags. Wie auch bei JavaScript gilt, dass man die Referenzierung von externen Dateien eigentlich so gut wie immer vorziehen sollte. Im Fall von Style Sheets erhalten Sie nur so die meist dringend notwendige Trennung von Struktur und Layout.

Nachfolgend sehen Sie das Schema zur Referenz auf eine externe CSS-Datei:

Listing 3.14: **Einbinden einer externen CSS-Datei**

```
<link type="text/css" rel="stylesheet"
  href="[URL der CSS-Datei]" />
```

Wenn Sie bei nur einem Element eine individuelle Stilinformation angeben wollen, können Sie bei Webseiten eine Style-Sheet-Anweisung auch als Inline-Definition eines Elementes verwenden. Dies bedeutet, über einen zusätzlichen `style`-Parameter innerhalb des Tags wird ein Attributwert gesetzt, und die Stilregel gilt ausschließlich innerhalb des definierten Containers. Zudem gibt es noch die Möglichkeit zum Import von Style Sheets.

3.5.2 Die konkrete Syntax von CSS-Deklarationen

Kommen wir nun kurz noch zu der konkreten Syntax von CSS-Deklarationen und den Möglichkeiten zur Angabe von Formatierungsregeln. Die Syntax einer CSS-Deklaration folgt immer dem gleichen Aufbau. Sie geben einen Namen, einen Doppelpunkt und die zu formatierende Eigenschaft an.

Die Syntax einer CSS-Deklaration sieht schematisch so aus:

Listing 3.15: **Schema einer CSS-Deklaration**

```
[name] : [wert]
```

3.5.3 Selektoren

Das zu formatierende Element (der sogenannte Selektor) wird einer solchen Regel vorangestellt. Mehrere Formatregeln für ein Element werden mit Semikolon getrennt und meist in geschweifte Klammern gesetzt. Dabei gibt es verschiedene Formen an Selektoren:

» Elementselektoren

» Attributselektoren

» Generationenselektoren

» den Universalselelektor

» Pseudo-Klassen

Im Bereich der Selektoren hat jQuery eines seiner praktischsten Highlights zu bieten.

3.6 Zusammenfassung

Sie haben in dem Kapitel die zentralen Grundlagen und Besonderheiten der modernen Internet-Programmierung in extrem kompakter Form kennengelernt. Die effektive Anwendung sowie das Verständnis von Frameworks und Toolkits wie jQuery basiert auf elementaren Kenntnissen in diesem Umfeld. Dabei haben wir die konkreten Techniken nur kurz gestreift, da eine gewisse Erfahrung damit (oder eine parallele Einarbeitung mit anderen Quellen) zum erfolgreichen Umgang mit dem Buch vorausgesetzt werden muss. Dennoch sollten Sie nun ein Verständnis für die Funktion der jQuery-Bibliothek haben und sehen, welche Vorteile Ihnen jQuery bieten kann. Insbesondere haben Sie in diesem Kapitel gesehen, wie Sie jQuery mit einer Webseite verbinden können und was Sie dabei zu beachten haben.

Copyright