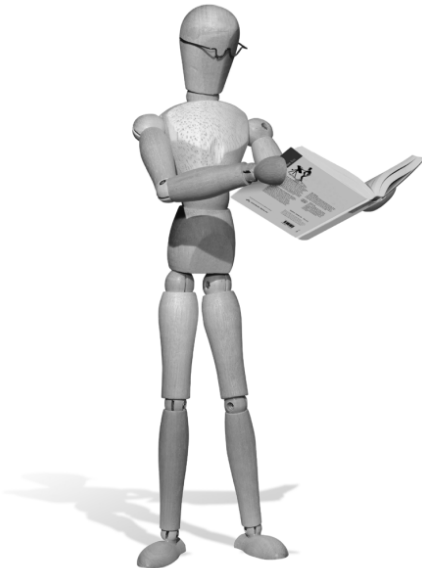


AJAX

mit Java-Servlets und JSP

So bringen Sie Speed in Ihre Webpräsenz





3 (X)HTML) und DOM für AJAX-Programmierer

Die Basis des WWW ist ohne Zweifel HTML respektive dessen neuere Variante XHTML. Auch wenn Sie mit AJAX Daten von einem Webserver anfordern, machen Sie das aus einer (X)HTML-Seite heraus. Und letztendlich werden die angeforderten Daten wieder im (X)HTML-Kontext durch einen Webbrowser dargestellt, selbst wenn sie vom Webserver als reiner Text oder XML geschickt wurden. Kenntnis von HTML ist das absolut notwendige Fundament. Am besten sollten Sie sogar über Erfahrungen mit XHTML verfügen. Ein grundlegendes Verständnis von (X)HTML soll – wie bereits angedeutet – nun beim Leser jedoch vorausgesetzt werden. Wir fassen hier kurz die wichtigsten Details zusammen und arbeiten Unterschiede zwischen HTML und XHTML heraus. Schwerpunkt dieses Abschnitts wird der AJAX-Blick auf (X)HTML. Dies bedeutet die Behandlung der wichtigsten Container-Tags, wie sie aus AJAX zum Austausch von Inhalt verwendet werden, der Eventhandler, die zum Auslösen von Anfragen verwendet werden, sowie der Hintergründe zum Document Object Model (DOM) aus Sicht von (X)HTML.

3.1 Der Aufbau von (X)HTML-Dateien

(X)HTML-Dateien selbst bestehen immer aus reinem Text (ASCII bzw. Unicode). Damit sind (X)HTML-Dokumente insbesondere plattformunabhängig und Sie benötigen zur Erstellung nur einen primitiven Klartexteditor.



Tipp

Ein sehr beliebter, kompakter und kostenloser HTML-Editor ist Phase 5 (<http://www.ftp-uploader.de/>). Aber auch das OpenSource-Programm Nvu (ehemals Netscape Composer – <http://www.mozilla.nightrat.net/nvu/> bzw. <http://www.nvu-composer.de/>) gewinnt immer mehr Freunde und geht in seiner Leistungsfähigkeit noch viel weiter. Es bietet unter anderem einen WYSIWYG¹-Modus, aber auch einen hervorragenden CSS-Editor.

1 What You See Is What You Get



Abbildung 3.1: Der OpenSource-HTML-Editor Nvu

Eine (X)HTML-Datei muss jedoch im Browser interpretiert werden, um dem Dokument eine über reinen Text hinausgehende Bedeutung zu verleihen. Webbrowser wurden nun von Anfang an so konzipiert, dass sie im Fall von HTML nach dem **Prinzip der Fehlertoleranz** arbeiten. Damit können auch syntaktisch fehlerhafte Dokumente im Client weitgehend ausgewertet und optisch aufbereitet werden. Das Prinzip der Fehlertoleranz vereinfacht die Bereitstellung von Informationen erheblich und hat in der Anfangszeit ohne Zweifel erst den Erfolg des Webs ermöglicht. Allerdings verlieren solche nur lose reglementierten Klartextdokumente die Möglichkeit, zuverlässig von automatischen Analysesystemen (Parsern) ausgewertet zu werden. Und natürlich geht ein Stück Information verloren, wenn es keine Eindeutigkeit von Strukturen gibt. Bei XHTML wurde deshalb das Prinzip der Fehlertoleranz explizit abgeschafft!

(X)HTML verfügt nun im Gegensatz zu vollständigen Programmier- oder Skriptsprachen (wie etwa JavaScript) über keine Kontrollstrukturen in Form von Bedingungen, Sprüngen oder Schleifen. Ebenso werden Sie in (X)HTML keine Variablen finden². Es gibt ebenso keine Befehle im Sinne von Befehlswörtern, die eine Aktion auslösen. Allerdings werden in HTML ab der Version 4 eine Reihe an Schlüsselwörtern bereitgestellt, die Voraussetzung für das Aufrufen von Funktionen sind (die angedeuteten Eventhandler, die wir gleich genauer betrachten). Besagte HTML-Version 4, die bereits seit 1997 verfügbar ist, ist auch der letzte Entwicklungsschritt von HTML gewesen und derzeit im Web trotz des schon antik zu nennenden Alters immer noch

² Im engeren Sinn. Allerdings können Sie Formularfelder als eine Form von Variablen sehen. Diese können aber nicht mit HTML selbst verwertet werden.

aktuell. Allerdings gibt es seit Januar 2000 XHTML, das als Ablösung von HTML konzipiert wurde. XHTML 1.0, das im Grunde HTML 5 darstellt, ist eine auf XML basierende Neuformulierung von HTML 4.01 und enthält dabei alle Elemente von HTML 4.01. Nur ist die Syntax von XHTML bedeutend strenger als von HTML. Die Konsequenz ist, dass ein nicht XHTML-fähiger Webbrowser XHTML-Dokumente jederzeit richtig darstellen kann. Für ihn erscheinen sie als normales HTML. Im Fall einer unklaren Situation wird das Prinzip der Fehlertoleranz genutzt. Gleichzeitig kann XHTML von neueren Browser gemäß den strengen XHTML-Regeln verarbeitet werden. XHTML hat sich jedoch faktisch bis heute in der Web-Community **nicht** durchgesetzt, da die wesentlichen Vorteile von XHTML (strengere Syntax und Reduzierung nichtstandardisierter Anweisungen) dem normalen Webdesigner oder auch Hobby-Webseitenersteller kaum verständlich sind. Und da die Masse der Webseiten-ersteller einfach HTML statt XHTML (oder eine Mischform) einsetzt³, werden sich Browser-Hersteller hüten, die HTML-Unterstützung einzustellen und die strengen XHTML-Regeln einzufordern.

3.2 Steueranweisungen

(X)HTML-Anweisungen bestehen aus Steueranweisungen, die aus so genannten **Tags** aufgebaut sind. Ein Tag beginnt immer mit einer geöffneten spitzen Klammer `<` und endet mit der geschlossenen spitzen Klammer `>`. Im Inneren der beiden Klammern befindet sich der konkrete Befehl. Ein Tag sieht von der Struktur her immer so aus:

Listing 3.1: Ein schematischer HTML-Tag

```
<Anweisung>
```



Hinweis

Ein Tag kann unter HTML sowohl klein als auch groß geschrieben werden. Auch Mischen von Groß- und Kleinbuchstaben ist erlaubt. Das hat keine Auswirkung. **Dies gilt jedoch nicht für XHTML.** Dort werden Anweisungen ausschließlich klein geschrieben. Um nun Webseiten konform zu den offiziellen Empfehlungen des W3C⁴ und den strengen XHTML-Regeln zu erstellen, sollten Sie in neuen Seiten Tags ausschließlich klein schreiben.

³ Und auch ich muss zugeben, dass ich mich nicht immer an die strengen XHTML-Regeln halte. Oft aus Faulheit, oft aber auch aus Vergesslichkeit. Wenn man nicht gezwungen wird und solche Lässigkeit keinerlei negative Konsequenzen hat, siegt halt die Schludrigkeit. Das ist menschlich ;-).

⁴ Das W3C – World Wide Web Consortium (<http://www.w3c.org>) – standardisiert nahezu alle relevanten Techniken im Web.

In (X)HTML gibt es zwei Formen von Tags:

1. Einen einleitenden Tag (Anfangs-Tag oder Beginn-Tag genannt)
2. Einen beendenden Tag (Abschluss-Tag oder Ende-Tag genannt)

Der einleitende Tag eröffnet eine Anweisung, während der beendende Tag sie wieder schließt. Beide Tags sehen fast identisch aus, außer einem vorangestellten Zeichen, mit dem der beendende Tag zusätzlich beginnt – dem Slash (Schrägstrich) /. Wenn beide Tags angegeben werden, bilden sie immer einen Container (in XML **Element** genannt). Dies bedeutet, die im einleitenden Tag angegebene Anweisung (etwa eine Formatierung) wirkt sich auf sämtliche Dinge (Objekte) aus, die sich im Inneren des Containers befinden. Dies wird in vielen Fällen ein Text sein, es kann sich aber auch um Grafik oder andere Multimediaobjekte handeln. Der Ende-Tag hebt die Wirkung eines Anfangs-Tag auf.

In reinem HTML werden die meisten Tags paarweise vorkommen. Es gibt jedoch Situationen, wo HTML-Tags keine Container bilden. In einigen Fällen greift das Prinzip der Fehlertoleranz. Das betrifft die Tags, die nach offizieller Vorgabe paarweise auftreten müssten, aber deren Wirkungsende sich auch ohne Abschluss-Tag auf Grund einer anderen Situation eindeutig ergibt (etwa bei Listen – ein neuer Listeneintrag beendet den vorherigen). Der andere Fall betrifft die HTML-Tags, die gar keinen offiziellen Abschluss-Tag haben (etwa ein Zeilenvorschub mit dem Tag `
`). In XHTML muss aber jeder Tag mit einem Abschluss-Tag versehen werden oder als so genanntes **leeres Element** definiert werden⁵ (also im Fall eines Zeilenumbruchs so: `
`).

Achtung



Der Zwang zur richtigen paarweisen Verwendung von Tags oder der expliziten Auszeichnung als leeres Element erweist sich vor allem dann als wichtig, wenn Sie eine Webseite mittels JavaScript und des DOM-Konzepts nachträglich manipulieren wollen. Falls hier keine korrekte Auszeichnung vorgenommen wurde, kann es zu Problemen kommen (hauptsächlich wenn Sie ein Ende-Tag weglassen – ein leeres Element wie einen ``-Tag nicht als solches zu kennzeichnen, macht im derzeitigen WWW nach meiner Erfahrung keinerlei Schwierigkeiten⁶).

Container bzw. Elemente können beliebige – sinnvolle – andere Tags (**Kindelemente**) enthalten, aber die Reihenfolge der Auflösung sollte eingehalten werden!

⁵ Im Sinn von XML – das behandeln wir im XML-Abschnitt.

⁶ Rein um sich einen sauberen Stil anzugewöhnen, ist eine Kennzeichnung dennoch sinnvoll.

Beispiel:

Listing 3.2: Ein kursiver, unterstrichener Text

```
<i></u>Es wird Sommer</i></u>
```

Wenn ein Container weitere Container enthält, sollten diese wieder von innen nach außen beendet werden – in umgekehrter Reihenfolge der Einleitungs-Tags. **In XHTML ist das auch zwingend.**

3.3 Attribute

Viele Tags sind erst dann sinnvoll einzusetzen, wenn sie genauer spezifiziert werden. Nicht jeder, denn beispielsweise ein Zeilenumbruch ist auch ohne die genauere Spezifizierung in (X)HTML durch `
` vollständig beschrieben. Aber nicht alle Anweisungen sind eindeutig. Parameter bzw. Attribute erweitern einen einleitenden (X)HTML-Tag und spezifizieren damit genauer die Bedeutung des Tag. Der Abschluss-Tag wird niemals mit Parametern erweitert. In HTML gibt es zwei Formen von Parametern:

1. Parameter mit einer Wertzuweisung
2. Parameter, die bereits einen Wert repräsentieren

Parameter mit einer Wertzuweisung bekommen über einen Zuweisungsoperator – das Gleichheitszeichen (=) – den entsprechenden Wert zugeordnet. Dies kann ein Text oder eine Zahl oder auch ein URL sein. Ein Tag mit einem Parameter mit einer Wertzuweisung sieht schematisch so aus:

Listing 3.3: Schema eines HTML-Tags mit Parameter und Wertzuweisung

```
<[Anweisung] [Parameter] = "[Wert]">
```

Viele Befehle lassen sich über mehr als einen Parameter spezifizieren. Diese werden dann einfach durch Leerzeichen getrennt aufgelistet. Bei mehreren Parametern spielt die Reihenfolge der Parameter keine Rolle.



Hinweis

In fast allen Fällen kann man bei der Wertzuweisung in einem HTML-Tag auf Hochkommata verzichten. Die Anweisung `` funktioniert in HTML uneingeschränkt. **In XHTML muss der zugewiesene Wert jedoch zwingend in Hochkommata eingeschlossen werden.**

Parameter, die bereits einen Wert repräsentieren, brauchen in HTML bloß durch ein Leerzeichen von der Anweisung oder anderen Parametern abgetrennt (!) in den Einleitungs-Tag geschrieben zu werden. Sie fungieren immer als Schalter. Wenn sie angegeben werden, wird eine Eigenschaft aktiviert, fehlen sie, ist die jeweilige Eigenschaft deaktiviert. Ein Tag mit einem Parameter, der bereits einen Wert repräsentiert, sieht schematisch so aus:

Listing 3.4: Schema für einen Parameter ohne Wertzuweisung

```
<[Tag] [Parameter]>
```

Ein Beispiel wäre `<table border>`, womit ein Tabellenrahmen gesetzt wird.

Achtung



In XHTML gibt es keine Parameter ohne Wertzuweisung.

Hinweis



Aus Sicht von AJAX ist ein Parameter besonders wichtig – `id`. Über die ID wird ein Bereich der Webseite identifiziert. Darüber kann z. B. dessen Inhalt ausgetauscht werden. Ebenso kann der Parameter in Zusammenhang mit Style Sheets eingesetzt werden. Ein weiterer wichtiger Parameter ist `name`. Damit vergeben Sie einen Namen für ein Element. Auch über diesen Namen können Sie ein Element einer Webseite aus DHTML heraus ansprechen. Zudem wird darüber bei Webformularen die Zuordnung von zu versendenden Werten vorgenommen. Und ebenso ist `class` wichtig, um eine Style Sheet-Klasse zuzuweisen.

3.4 Strukturierung und Gestaltung mit HTML

Obwohl HTML über die Zeit zahlreiche Gestaltungsmöglichkeiten für eine Webseite erworben und die Webdesigner diese in der Vergangenheit auch exzessiv ausgelebt haben, geht die aktuelle Tendenz der Webseitengestaltung dahin, möglichst alle Layoutfragen einer Webseite nicht mehr mit HTML zu lösen. Stattdessen wird das Layout vollkommen in – möglichst externe – Style Sheets (Formatvorlagen) ausgelagert oder zumindest mit Style Sheet-Parametern bei Tags gearbeitet. HTML wird nur noch zu Strukturierung einer Webseite verwendet und nicht mehr in dem Sinn, mit Befeh-

len wie `` eine Schriftgröße festzulegen oder mit `` einen Text fett auszuzeichnen. HTML-Befehle zur Angabe von Überschriften oder Absätzen strukturieren zwar immer noch eine Webseite, aber auch deren Layoutwirkungen werden über Style Sheets modifiziert oder gleich ganz ausgeschaltet. Dieser Schritt ist ein Teil des Bemühens, das WWW in ein semantisches Web umzuwandeln, in dem die reinen Inhalte sehr stark formalisiert und damit maschinenlesbar werden.

3.4.1 Gruppierung von Inhalt

Aus Sicht von AJAX sind bei der Strukturierung einer Webseite besonders Tags zur Gruppierung von Inhalt von Bedeutung. Gruppierung von Inhalt kann man in HTML beispielsweise mit Überschriften (`<h1>` bis `<h6>`) oder auch Absätzen (`<p>`) machen, aber besonders wichtig sind die Strukturelemente `<div>` und ``. Man kann damit eine Unterstruktur innerhalb einer Webseite aufbauen und diesen Container verwenden, um den darin enthaltenden Inhalt zusammenzufassen und ihn formatieren oder ansprechen (etwa mit JavaScript) zu können. Genau das haben wir im letzten Kapitel gemacht, als wir mit der ID eines solchen Containers den Inhalt ausgetauscht haben (ursprünglich war er sogar leer).

Listing 3.5: Die Zeile 27 aus dem Webformular (laender.html) in Kapitel 2

```
<span id="hs"></span>
```

Darauf kommen wir gleich bei der Behandlung von DOM und später noch bei DHTML noch genauer zurück.



Tipp

Während die meisten Webgestalter meist den `<div>`-Container bei DHTML-Effekten und beim Datenaustausch mit AJAX einsetzen, bietet sich in vielen Situationen eher der unverständlicherweise ziemlich unterschätzte ``-Container an. Dieser hat im Gegensatz zu `<div>` (Zeilenvorschub) keinerlei Eigenwirkung und ist damit für eine Inline-Formatierung ideal. Er ist damit prädestiniert zur Auslagerung von Gestaltung in Style Sheets oder die Ansprache aus Programmier- und Skriptsprachen (etwa JavaScript). Der ``-Container ist somit logisch bedeutend radikaler bei der Trennung von Struktur, Layout und Funktionalität als der `<div>`-Container, der unter seiner »Verschmutzung« durch eine HTML-Wirkung »leidet«. Allerdings hat der `<div>`-Container den Vorteil, dass man Breite und Höhe angeben und damit das Layout einer Seite sehr flexibel gestalten kann. Man wird sinnvollerweise damit arbeiten, wenn sowieso komplette Absätze formatiert werden sollen.

3.5 Formulare zur Benutzerinteraktion

Ein – wenn nicht der – zentrale Aspekt von AJAX ist die Möglichkeit, die Benutzerinteraktion zu optimieren. Und die Interaktion mit dem Benutzer erfolgt in vielen Fällen über Webformulare. Besonders interessant ist für uns, dass nicht nur die Interaktion mit dem Anwender oft über ein Webformular erfolgt, sondern dass ebenso die Anforderung von Daten per AJAX auf der gleichen Technik wie die von Webformularen fußt. In AJAX wird ja in einer bestimmten Situation ein Eventhandler oder ein Ereignis allgemein ausgelöst und eine Anforderung nach Ergänzungsdaten per HTTP weggeschickt. Und das ist für den Browser genau der gleiche Vorgang, der beim Verschieken von Benutzereingaben in einem Webformular durchgeführt wird⁷.

3.5.1 HTML-Formular-Tags

Ein Formular in einer Webseite wird im Wesentlichen aus Eingabefeldern, mehrzeiligen Textfeldern, Listen und Aufzählungen, Schaltflächen und beschreibendem Text bestehen. Das Formular besteht aus dem äußeren `<form>`-Container und darin enthaltenen Tags zur Spezifizierung der Eingabelemente. Die meisten Eingabefelder basieren auf dem Tag `<input>`, der mit dem Attribut `type` genauer spezifiziert wird und damit verschiedene Ausprägungen annehmen kann:

- **Einzeilige Eingabefelder** werden ohne den `type`-Parameter oder mit `type="text"` erstellt.
- Mit `type="password"` definiert man ein **Passwortfeld** mit verdeckter Eingabe.
- Mit `type="button"` erstellen Sie eine **Schaltfläche** (über `value` erfolgt die Beschriftung).
- Mit `type="checkbox"` können Sie ein **Kontrollfeld** (das `value`-Attribut bezeichnet den Wert, der beim Versenden des Formulars übermittelt wird) erstellen.
- Die Typangabe `radio` spezifiziert einen **Radiobutton (Optionsfeld)**.
- Eine **Schaltfläche zum Zurücksetzen** der Formulareingaben wird über `type="reset"` erstellt, eine **Schaltfläche zum Abschicken** des Formulars über `type="submit"`. Beide Schaltflächen erhalten eine individuelle Beschriftung über `value` (andernfalls wird eine Default-Beschriftung durch den Browser vergeben).

Mehrzeilige Eingabefelder werden mit einem eigenen HTML-Tag realisiert – dem `<textarea>`-Container. **Auswahllisten** in Formularen werden mit einem äußeren `<select>`-Container für die Listenstruktur und jeweils einen inneren `<option>`-Container für jeden Listeneintrag realisiert. Der **äußere** Container wird um das Attribut `name` erweitert, der Tag für den Listeneintrag benötigt keinerlei Parameter. Bei allen zu versendenden Formularfeldern ist das Attribut `name` wichtig. Darüber erfolgt beim Versenden von Benutzereingaben die Zuordnung, welche Information ein Anwender wo

⁷ Oder aber auch bei der Anforderung einer Seite über die Adressleiste des Browsers.

eingetragen hat. Es werden bei den meisten Browsern nur die Inhalte derjenigen Formularfelder verschickt, die mit dem `name`-Parameter spezifiziert sind. Mit anderen Worten – wenn in einem Webformular Formularfelder enthalten sind, die nicht mit dem `name`-Parameter spezifiziert sind, werden dort vorgenommene Eingaben beim Verschicken des Formulars nicht versendet.



Achtung

Sie sollten aufpassen, dass kein Name für ein Webformularelement mehrfach vorkommt. Sie können zwar die Reaktion nicht für jeden Browser vorhersehen, aber in der Regel werden die Inhalte der mehrfach benannten Felder versendet und dann besteht das Problem, auf Serverseite eine richtige Zuordnung vorzunehmen.



Tipp

Die Gestaltung von Formularen erweist sich durch die unterschiedliche Größe von Formularfeldern als etwas diffizil. Zur Anordnung von Formularen bieten sich deshalb Tabellen an. Dabei können sowohl das Formular in die Tabelle als auch die Tabelle in das Formular eingeschlossen werden. Alternativ sieht man in der letzten Zeit verstärkt die Gestaltung über Style Sheets, um vor allem den Anforderungen des barrierefreien Webs zu genügen.

3.5.2 Der `<form>`-Tag – remember AJAX

Innerhalb des `<form>`-Tags werden in der Regel einige, durch ein Leerzeichen getrennte Angaben gemacht, die Sie auch zum Teil bei den Funktionen im AJAX-Umfeld wiederfinden. Besonders wichtig sind folgende Parameter des `<form>`-Tags:

- Eine Adresse, wohin die eingegebenen Daten zur Verarbeitung geschickt werden. Dazu dient die Erweiterung `action=[URL]`. Diese Angabe eines URL finden wir auch bei der `open()`-Methode eines `XMLHttpRequest`-Objekts unter AJAX als zweiten Parameter wieder.
- Eine Methode, wie die Daten, die im Formular eingetragen wurden, zur Auswertungsstelle gelangen. Die Erweiterung `method=[Methode]` legt die Art der Übertragung fest. Die Art und Weise der Übertragung finden wir auch bei der `open()`-Methode eines `XMLHttpRequest`-Objekts unter AJAX als ersten Parameter wieder. Sie werden dort in der Praxis entweder `GET` oder `POST` als Wert finden. Beide Werte des `method`-Parameters beschreiben wir im Kapitel der AJAX-Grundlagen genauer. Die Werte spezifizieren unterschiedliche Möglichkeiten, mit HTTP Daten vom Client zum Server zu schicken.

3.6 Ab in den DOM – aus Sicht von HTML

Das **Document Object Model (DOM)** bezeichnet einen Standard des W3C. Er ist wie fast alle Spezifikationen des W3C plattform- und programmiersprachenunabhängig definiert. Das Konzept beschreibt ein Verfahren, bei dem ein Dokument nicht als statisch aufgebaute, fertige und nicht unterscheidbare Einheit, sondern als differenzierbare Struktur betrachtet wird. Deren einzelne Bestandteile sind Programmen und Skripten dynamisch zugänglich. Über das DOM lässt sich ein Dokument als Baumstruktur mit hierarchischer Verschachtelung der Elemente auffassen.

Dieser Ansatz ermöglicht bei Webseiten unter anderem die individuelle Behandlung von Bestandteilen der Webseite auch dann, wenn diese bereits in den Browser geladen ist. Jeder Container, jede Grafik, jede Referenz lassen sich nach dem Laden der Webseite wieder ansprechen. Das ist genau das, was wir in unserem ersten AJAX-Beispiel genutzt haben, als wir mittels der ID den Inhalt von einem ``-Container ausgetauscht haben. Erinnern Sie sich an die Funktion zum Behandeln der zurückgelieferten Daten aus dem Beispiel in Kapitel 2:

Listing 3.6: Die Funktion zum Behandeln der zurückgelieferten Daten

```
19 function handleResponse() {
20   if(resObjekt.readyState == 4){
21     document.getElementById("hs").innerHTML =
22       resObjekt.responseText;
23   }
24 }
```

Das DOM ist aber kein Konzept, das speziell auf HTML-Dokumente gemünzt ist, sondern allgemein auf Klartextdokumente angewendet werden kann, die klare Strukturen aufweisen. Dies betrifft alle auf SGML zurückgehenden Dokumente und vor allem XML-Dokumente, deren Verarbeitung oder Validierung darüber möglich ist. Je nach Dokumenttyp und angewendeter Programmieretechnik nutzt man dabei unter Umständen verschiedene Facetten des DOM.

Unabhängig von der Art der strukturierten Klartextdatei werden beim Laden eines Dokuments durch einen DOM-kompatiblen Parser alle ihm im Rahmen des Konzepts bekannten und einzeln identifizierbaren Elemente bezüglich ihres Typs, ihrer relevanten Eigenschaften und ihrer Position innerhalb des Dokuments indiziert. Dies ist eine Art dynamische Datenstruktur im Hauptspeicher des Rechners, über die später alle indizierten Bestandteile des Dokuments in Form einer Objekthierarchie verfügbar sind. Im DOM werden alle Elemente eines Dokuments, vom Tag über Texte bis hin zum Attribut, als eine Menge zusammenhängender **Knoten** repräsentiert, die einen Baum abbilden und bei der man zwischen Knoten Eltern-Kind-Beziehungen beschreiben kann, wenn diese ineinander geschachtelt sind. Dies wird beispielsweise bei XML über eine Technik mit Namen **XPath** genutzt, um Wege von einem Knoten zu einem

anderen Knoten anzugeben. Aber auch bei CSS kann man angeben, dass sich eine Formatierung nur dann auf ein Element auswirkt, wenn dieses im DOM-Baum auf eine ganz bestimmte Weise verschachtelt ist.

Aber um es an dieser Stelle nicht zu kompliziert zu machen – das DOM ist kurz gefasst ein komplettes Modell, das jedes Element eines Dokuments beinhalten kann⁸ und es erlaubt, diese nachträglich zu verwenden. Für JavaScript bedeutet dies, dass das DOM den Zugriff auf alle Bestandteile einer Webseite gewährleistet. Dies erfolgt mit geeigneten Methoden und Eigenschaften über verschachtelte Standardobjekte, benannte Elemente (über den Parameter `id`) oder standardisierte Datenfelder für bestimmte Elementtypen⁹. Um dies aber genauer besprechen zu können, brauchen wir JavaScript. Deshalb werden wir beim Abschnitt über JavaScript auf DOM zurückkommen.

3.7 Ein tolles Ereignis

Explizit zu HTML gehören auch **Eventhandler**. Das sind Schlüsselwörter, die als Attribute bei HTML-Tags notiert werden können und beim Auftreten von einem bestimmten Ereignis¹⁰ eine nachfolgend notierte Funktion (so gut wie immer in JavaScript geschrieben) auslösen. Da Eventhandler vom W3C offiziell in den HTML-Sprachstandard aufgenommen wurden, spielt damit Groß- und Kleinschreibung bei der Notation keine Rolle. Es hat sich aber die Notation von Eventhandlern eingebürgert, klein zu beginnen und die zusammengesetzten Bezeichner bei jedem neuen Begriff mit einem Großbuchstaben zu notieren. Das W3C hat ebenfalls festgelegt, in welchen HTML-Tags welcher Eventhandler vorkommen darf, wobei sich einige Browser-Hersteller an diese Vorgaben nur sehr mangelhaft halten. Als Exempel für die manchmal unterschiedliche Unterstützung von Eventhandler können Sie wieder unser AJAX-Beispiel aus Kapitel 2 heranziehen. Dort hatte ich als Alternative für den Einsatz des Eventhandlers `onClick` bei einem Listefeld auch die Möglichkeit demonstriert, diesen beim `<option>`-Tag direkt zu verwenden (etwa so: `<option onClick="sndReq2(7)">Hessen</option>`). Das unterstützen zum Beispiel Opera oder Firefox, aber nicht der Internet Explorer¹¹.

-
- 8 Leider ist es nicht ganz eindeutig, welche Elemente von Webbrowsern bei einer Webseite tatsächlich indiziert werden. Beim Parsen einer Webseite merken sich manche Browser in der Tat jedes Element, andere dagegen nur ausgewählte Elemente. Besonders ältere Browser unterscheiden sich hier gewaltig.
- 9 Alle Formulare in einer Webseite werden etwa in ein Objektfeld mit Namen `forms` einsortiert, das dem DOM-Objekt `document` untergeordnet ist. Dieses wiederum ist `window` untergeordnet. Also erfolgt der Zugriff auf das dritte Formular in einer Webseite über `window.document.forms[2]` – die Indizierung beginnt bei 0.
- 10 Etwa wenn ein Anwender auf ein Element einer Webseite klickt oder mit dem Mauszeiger den Bereich eines Webseitenelements (zum Beispiel ein Bild) überstreicht.
- 11 Bei einem Einsatz von Eventhandlern hilft nur, diese in allen relevanten Browsern zu testen. Glücklicherweise passen sich aber die Verhaltensweisen der gängigen Browser immer mehr an.

3.7.1 Die konkreten Eventhandler

Eventhandler stellen in jedem Fall das wichtigste Bindeglied zwischen HTML und JavaScript (und damit auch AJAX) dar. Ein Eventhandler sieht aus wie jedes gewöhnliche HTML-Attribut. Gemeinhin beginnen die Namen von Eventhandlern mit der Silbe `on`, gefolgt von einer sprechenden Beschreibung des Ereignisses. Hinter dem Namen des Ereignisses wird ein Gleichheitszeichen notiert, gefolgt von einer in Anführungszeichen notierten JavaScript-Anweisung oder -Funktion. Beispiel:

Listing 3.7: Ein Beispiel für den Aufruf einer Funktion mit einem Eventhandler

```
onClick="mfkt()"
```

Innerhalb der Regeln des W3C und der Unterstützung durch die verschiedenen Browser können Eventhandler relativ beliebig in einer Webseite eingesetzt werden, um die Reaktionslogik einer Webseite zu beschreiben. Die wichtigsten Ereignisse, bei denen eine Reaktion sinnvoll implementiert werden kann, sind

- das Laden (`onLoad`) und Verlassen (`onUnload`) einer Webseite,
- das Abschicken (`onSubmit`) und Zurücksetzen (`onReset`) von Formulardaten,
- das Überstreichen eines Bereichs der Webseite mit dem Mauszeiger (`onMouseOver` beim Erreichen des Bereichs und beim Verlassen (`onMouseOut`) oder auch allgemein bei der Bewegung der Maus (unabhängig davon, ob die Maustaste gedrückt ist oder nicht – `onMouseMove`),
- der Anwenderklick auf eine Referenz, ein Element der Webseite oder eine Schaltfläche (`onClick`)¹² und
- Änderung (`onChange`), Aktivierung (`onFocus`) und Verlassen (`onBlur`) eines Elements in einer Webseite – hauptsächlich in Zusammenhang mit Formularen.

Wie wir im letzten Kapitel schon angesprochen haben, werden insbesondere bei AJAX auch Tastatureingaben wichtig sein. Dabei kann man zwischen dem vollständigen Tastendruck (`onKeyPress`), dem Drücken einer Taste (`onKeyDown`) und dem Loslassen einer Taste (`onKeyUp`) unterscheiden. Tastaturbezogene Eventhandler werden hauptsächlich bei freien Eingabefeldern in einem Webformular ihren Einsatz finden.

¹² Es gibt natürlich nicht nur den einfachen Klick auf eine Referenz, sondern auch die Auswertung des gleichzeitigen Drückens einer Maustaste, des anschließenden Loslassens einer Maustaste und des Doppelklicks. Auch die jeweiligen Teilphasen bilden jeweils ein eigenes Ereignis. Diese werden über `onMouseDown`, `onMouseUp` und `onDoubleClick` beschrieben. Diese Verwendung (insbesondere eines Doppelklicks) spielt im Web aber in der Praxis keine Rolle.

**Hinweis**

Mit einem Klick auf eine Referenz ist gemeint, dass Sie auf ein Element wie eine Schaltfläche in einem Formular mit der Maus klicken. Auch ein `<a>`-Element ist im engen Sinn eine Referenz. In einigen neueren Browsern gelten aber auch die meisten anderen Tags als Referenzen. Ob Anwender nun Überschriften, Bilder, Trennlinien etc. für einen Klick wählen. Im Grunde braucht der HTML-Tag nur noch einen Bereich zu beschreiben, auf den ein Anwender den Mauszeiger bewegen und klicken kann. Sie erweitern den entsprechenden Tag durch das Attribut `onClick`. Allerdings sollten Sie auf jeden Fall verschiedene Browser testen, ob der Eventhandler auch unterstützt wird.

**Hinweis**

Weitere Eventhandler zur Reaktion auf das Selektieren von Text (`onSelect`) oder die Unterbrechung eines Bildladevorgangs (`onAbort` und `onError`) sind in der Praxis von geringerer Bedeutung.

3.8 Zusammenfassung

Wir haben in diesem Kapitel die bedeutenden Fakten zu (X)HTML rekapituliert und abstrahiert zusammengefasst. Sie kennen nun die wichtigsten Unterschiede zwischen XHTML und HTML, die ersten Details zum DOM-Konzept und können Eventhandler zum Aufruf von JavaScript-Funktionen einsetzen.