Jan Winkler

Know-how ist blau.

2. aktualisierte Auflage

JavaScript und Ajax

Das Praxisbuch für Web-Entwickler

> JavaScript-Grundlagen beherrschen und anwenden
> Ajax-Anwendungen verstehen und selbst programmieren
> Inklusive umfassender Objektreferenz zum Nachschlagen

FRANZIS

Jan Winkler

JavaScript und Ajax

Jan Winkler

2. aktualisierte Auflage

JavaScript und Ajax

Das Praxisbuch für Web-Entwickler

Mit 35 Abbildungen

FRANZIS

Bibliografische Information der Deutschen Bibliothek

Die Deutsche Bibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie;
detaillierte Daten sind im Internet über http://dnb.ddb.de abrufbar.

Lektorat: Franz Graser
Satz: DTP-Satz A. Kugge, München
art & design: www.ideehoch2.de
Druck: Bercker, 47623 Kevelaer
Printed in Germany

ISBN 978-3-7723-6262-0

Inhaltsverzeichnis

1 Einführung

Was ist JavaScript? Was steckt eigentlich dahinter? Wie wendet man es an? Was versteht man unter Ajax, und wozu ist es überhaupt gut? Genau diesen Fragen werden wir in diesem Buch nachgehen. Am Ende werden Sie die Grundzüge von JavaScript, DHTML und Ajax beherrschen, die gängigen Frameworks kennen und außerdem reichlich Praxisluft geschnuppert haben, sodass Sie in der Lage sein werden, eigene JavaScript- oder Ajax-Anwendungen zu entwickeln.

1.1 Gliederung dieses Buches

Der Aufbau ist so gestaltet, dass die einzelnen Kapitel mehr oder minder aufeinander aufbauen beziehungsweise die in einem Kapitel erworbenen Kenntnisse für das Verständnis der darauf folgenden Kapitel benötigt werden. Los geht es mit dem Grundlagenkapitel zum Thema JavaScript – hier erlernen Sie, was JavaScript eigentlich ist und wie Sie eigene Programme in JavaScript schreiben können. Als Nächstes folgt eine Einführung in das Thema DHTML (Dynamic HTML), mit dessen Hilfe Sie mehr Abwechslung in eine Website bringen können. Anschließend erlernen Sie, was es mit Ajax (Asynchronous JavaScript and XML) auf sich hat und wie Sie es für Ihre Zwecke einsetzen können. Den Abschluss bilden dann ein Kapitel mit zahlreichen Praxisbeispielen und hilfreichen Tricks sowie eine umfassende Objektreferenz, mit der Sie alle wichtigen JavaScript-Objekte nachschlagen können.

1.2 Was muss ich können, um dieses Buch zu verstehen?

Grundsätzliche Voraussetzung, um den Inhalt dieses Buches verstehen und anschließend umsetzen zu können, sind solide Grundkenntnisse in den Bereichen Internet (Funktionsweise, Protokolle usw.) sowie gefestigte Kenntnisse in Sachen HTML. Letzteres ist für die Anwendung von JavaScript unerlässlich. Für die Arbeit mit DHTML sollten Sie darüber hinaus bereits mit CSS (Cascading Style Sheets) gearbeitet haben und – für das Verständnis von Ajax – über ein Basiswissen in XML (Extensible Markup Language) verfügen. Da Ajax in der Regel im Verbund mit einer auf dem Webserver operierenden Sprache agiert, sind ebenfalls entsprechende Grundkenntnisse in zumindest einer serverseitigen Programmiersprache wie PHP, ASP/ASP.NET oder Java (Java Server Pages, JSP) von Vorteil. Insbesondere bei größeren Anwendungen kommt

zudem MySQL beziehungsweise Microsofts SQL-Server als Datenbanksystem hinzu, für das ebenfalls entsprechende Kenntnisse vorliegen sollten.

Lassen Sie sich von der Fülle der eben aufgezählten Programmiersprachen und Technologien jedoch nicht verunsichern: In der Regel wird je nach Anwendung und Zielsetzung nur ein Bruchteil davon benötigt. Um den Einstieg zu erleichtern, werden wir zudem an den passenden Stellen kurze Erläuterungen und Exkurse in die jeweiligen Bereiche bieten.

1.3 Welche Werkzeuge benötige ich?

JavaScript und damit alle davon abhängigen Technologien begnügen sich mit recht wenig softwaretechnischem Beiwerk. Für die Programmierung von JavaScript-Anwendungen sind daher im simpelsten Fall ein Texteditor (vgl. Windows Notepad) zum Entwickeln sowie ein gängiger Browser zum Testen notwendig. Da JavaScript- und Ajax-Anwendungen allerdings auch schon mal ein paar Hundert bis Tausend Zeilen an Code umfassen können, empfiehlt es sich hier, auf einen entsprechend ausgerüsteten Editor zu setzen. Die meisten der verbreiteten Editoren unterstützen das Editieren von JavaScript-Code mit zahlreichen nützlichen Features wie Syntaxhervorhebung, Auto-Vervollständigung und weiteren Funktionen, die das Programmieren deutlich vereinfachen und damit effizienter machen können.

1.3.1 Ajax testen

Da Ajax allein auf dem Client-Rechner, das heißt auf dem Browser, recht wenig Sinn ergeben würde, benötigt man für diese Funktionen zusätzlich eine Web-Umgebung, auf der sich Techniken wie PHP, ASP oder JSP testen lassen. Im einfachsten Fall lädt man die Dateien jeweils auf den eigenen Webspace hoch und testet sie dann dort. Wem dies zu umständlich beziehungsweise zeitaufwendig ist, der kann sich mit einem sogenannten XAMPP-Installationspaket[1] behelfen. Dabei handelt es sich um ein Programmpaket, welches unter anderem die Software Apache (ein Webserver), MySQL (ein Datenbanksystem) und PHP (eine Programmiersprache für serverseitige Programmierung) beinhaltet und diese Komponenten in der Regel schon komplett vorgefertigt und vorkonfiguriert mit sich bringt. Einmal installiert, hat man dann auf dem heimischen PC einen eigenen Webserver samt Datenbank und PHP laufen – ganz ähnlich wie bei den meisten Webhostern – und kann so Ajax-Skripte in Verbindung mit PHP direkt von zu Hause aus testen.

[1] XAMPP ist für zahlreiche Betriebssysteme, darunter Windows, Linux und Mac OS X, unter der Adresse `http://www.apachefriends.org/de/xampp.html` kostenlos zum Download erhältlich.

1.4 Nützliche Editoren und Programme

Auf dem Markt und im Internet existieren außerdem zahlreiche Tools und Programme, die Ihnen die Arbeit am Code deutlich erleichtern können. Einige davon möchten wir Ihnen hier kurz vorstellen, um Ihnen die Auswahl zu erleichtern.

1.4.1 Der Alleskönner: Adobe Dreamweaver

Wer es gern übersichtlich hat und sich beim Design seiner Website direkt anschauen möchte, was er programmiert, ist mit der Lösung von Adobe gut beraten. Insbesondere das Erstellen des Designs einer Seite fällt mit diesem Programm sehr leicht. Zudem wartet es mit Editorfunktionen auf, die einen direkten Eingriff in den Webseiten-Code und damit die Möglichkeit zur Einbindung von JavaScript bieten. Für den Einsteiger ist dieses Programm sehr zu empfehlen – einige Profis setzen jedoch auf reine Texteditoren. Das Programm ist ab 570 Euro erhältlich.

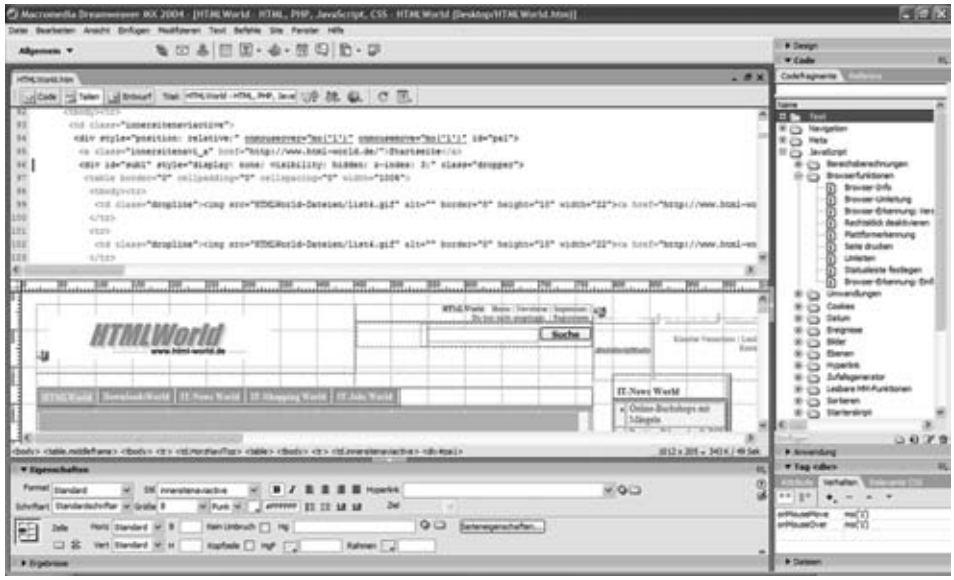

Bild 1.1: Die Oberfläche von Dreamweaver lässt sowohl die Code-Bearbeitung als auch die Seitengestaltung mit WYSIWYG-Editor und Vorschaufunktion zu.

1.4.2 Alleskönner Nr. 2: Microsoft Expression Web Designer

Ebenfalls sehr für Anfänger zu empfehlen sind die Programme aus dem Hause Microsoft, namentlich der Expression Web Designer beziehungsweise das ältere Frontpage. Ähnlich wie Dreamweaver bietet es einfache Funktionen zum Erstellen der Oberflächen, aber zugleich auch die Möglichkeit, den Code direkt zu bearbeiten. Die aktuelle Version von Microsoft Expression Web schlägt mit rund 299 Euro zu Buche.

1.4.3 Der Allrounder: Eclipse mit Eclipse Web Tools Platform

Eclipse ist ursprünglich aus einer Entwicklungsumgebung für die Programmiersprache Java hervorgegangen, mit einigen ergänzenden Tools und Plugins nun aber auch sehr gut für HTML und JavaScript geeignet. Das Programm ist ein reiner Texteditor – das heißt, eine grafische Bearbeitung wie bei Dreamweaver oder Expression Web ist nicht möglich. Dafür bietet die Umgebung umfangreiche Funktionen, die das Programmieren am Code deutlich vereinfachen und beschleunigen können. Da es sich um eine Open-Source-Lösung handelt, kann es kostenlos unter `www.eclipse.org` heruntergeladen werden. Fortgeschrittene Programmierer und Profis setzen mehrheitlich auf derartige Entwicklungsumgebungen.

1.4.4 Die Alternativen: UltraEdit, SuperHTML, Phase 5, Aptana ...

Neben den drei vorgenannten Programmen gibt es eine ganze Reihe kostenloser und kostenpflichtiger Editoren im Internet, die im Wesentlichen die gleichen oder zumindest ähnliche Funktionen aufweisen. Im Besonderen zu nennen wären hier UltraEdit (49,90 Euro, `www.ultraedit.at`), SuperHTML (79,95 Euro, `www.superhtml.de`), Phase 5 (kostenlos, `www.qhaut.de`) sowie Aptana Studio (kostenlos, `www.aptana.com/studio`).

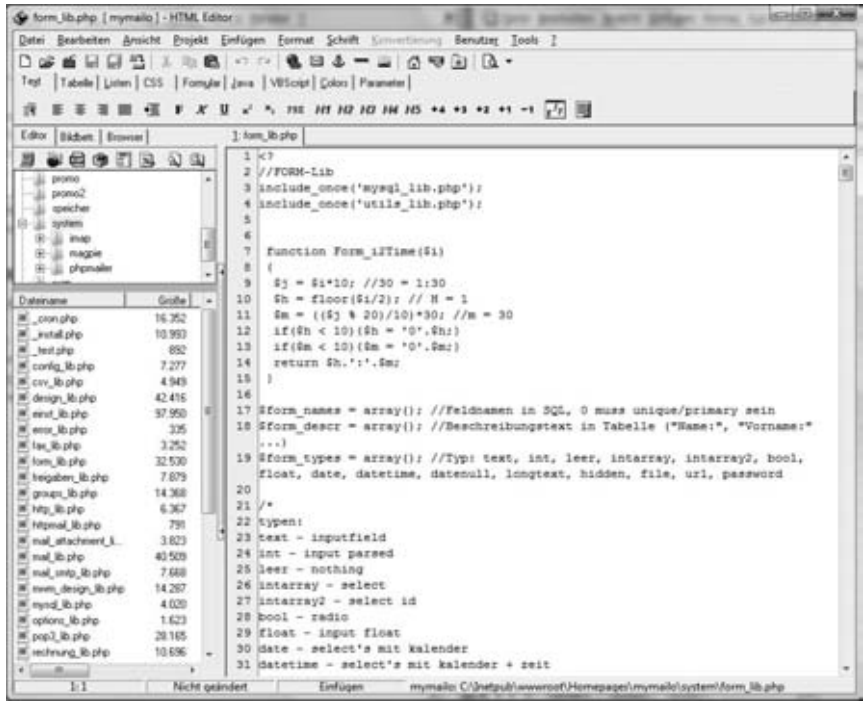

Bild 1.2: Im Gegensatz zu den grafisch orientierten Tools Expression Web und Dreamweaver wirkt der HTML-Editor Phase 5 relativ monoton.

5 JavaScript, DHTML und Ajax in der Praxis

Nachdem wir uns mit den Grundlagen beschäftigt haben, wollen wir JavaScript nun in der Praxis anwenden. Da sich natürlich nicht jede Situation und jede Programmieraufgabe innerhalb dieses Kapitels abbilden lassen – dafür sind die Möglichkeiten von JavaScript einfach zu umfangreich –, wollen wir uns hier den wichtigsten Dingen kurz widmen.

5.1 JavaScript und der Browser: Die wichtigsten Objekte

Wird JavaScript-Code in einem Browser ausgeführt (in der Regel also in einer Webseite), stellt dieser für gewöhnlich eine Reihe von Objekten zur Verfügung, mit denen man auf die einzelnen Bestandteile des Browserfensters bzw. des angezeigten Dokuments zugreifen kann. Je nach verwendetem Browser und benutzter Version unterscheidet sich zwar die Ausprägung der Objekte, im Wesentlichen beherrschen die wichtigen Browser wie Microsoft Internet Explorer, (Mozilla) Firefox, Opera oder Safari jedoch die gleichen Objekte. Folgende Objekte sind dabei für quasi jeden JavaScript-Programmierer unerlässlich:

5.1.1 Number-, Date-, Array- und String-Objekt

Bei diesen Objekten handelt es sich so gesehen um erweiterte Datentypen, da etwa Zahlen, Strings und Datumswerte in vielen Programmiersprachen nicht als Objekt, sondern als normaler Datentyp behandelt werden. In JavaScript ist es etwas anders, hier ist jeder Datentyp auch gleich ein Objekt und kann als solches Eigenschaften und Methoden tragen. Beispiel:

```
alert((3.666667).toFixed(2)); //auf 2 Kommastellen brechen
a = 'Hans Hase';
alert(a.length); //Anzahl der Zeichen ausgeben
```

5.1.2 window-Objekt

Das `window`-Objekt ist quasi das oberste Objekt in der Hierarchie – alle anderen Objekte, die im Browser zur Verfügung stehen, stammen von ihn ab. Das Objekt selbst

stellt die Schnittstelle zum – wie sollte es anders sein – Fenster dar, in dem das Skript gerade ausgeführt wird. Hierzu bietet es Eigenschaften, die es etwa erlauben, die Statusanzeige zu ändern, neue Fenster zu öffnen oder das Fenster zu vergrößern bzw. zu verkleinern. Beispiel:

```
window.status = 'JavaScript in der Statusleiste';
a = window.open('http://www.html-world.de','a');
a.resizeTo(500,500); //neues Fenster a auf 500x500 ändern
```

Bild 5.1: JavaScript ermöglicht auch den Zugriff auf die Statusleiste des Browserfensters

5.1.3 document-Objekt

Das `document`-Objekt ist eigentlich ein Unterobjekt des `window`-Objekts, kann aber aufgrund des häufigen Gebrauchs auch ohne Nennung von `window` verwendet werden. Es bietet den Zugriff auf das angezeigte Dokument sowie die darin enthaltenen Elemente wie Formulare, Links, Bilder usw. Beispiel:

```
document.title = 'Seitentitel';
document.bgColor = 'red'; //Hintergrundfarbe auf Rot
document.writeln('Ich bin JavaScript');
document.meinFormular.meinFeld.value = '23';
```

5.1.4 location- und history-Objekt

Diese beiden Objekte sind ebenfalls Abkömmlinge des `window`-Objekts und dienen dem Zugriff auf die Navigation bzw. History (Verlauf) des Browsers. Damit ist es möglich, z. B. eine neue Seite anzusteuern oder auf die zuvor besuchte Seite zurückzuspringen. Beispiel:

```
location.relaod(); //Seite neu laden

location.href = 'neueseite.html'; //andere Seite anzeigen

history.back(); //letzte Seite anzeigen
```

5.2 Datum & Zeit

Ein guter Einstieg in die JavaScript-Praxis ist die Arbeit mit Datums- und Zeitwerten, da sich Datumsangaben nicht einfach so im Zehnersystem umrechnen lassen – 40 Minuten ergeben eben nicht 0,4 Stunden. JavaScript bietet hierfür bereits vorgefertigte Bibliotheken und Objekte, welche die Verarbeitung von Zeit- und Datumsangaben deutlich vereinfachen.

5.2.1 Das Date-Objekt

Das `Date`-Objekt ist ein von JavaScript zur Verfügung gestelltes Objekt zur Verarbeitung von Datums- und Zeitangaben. Das Objekt besitzt diverse Methoden, die es etwa erlauben, ein neues Datum zu setzen oder Datumsangaben sinnvoll zu formatieren.

Um das `Date`-Objekt zu verwenden, muss es, wie jedes andere Objekt auch, zunächst in einer Variablen erstellt werden:

```
a = new Date();
```

Außerdem können dem Constructor diverse Parameter übergeben werden, um das Datum auf einen definierten Wert zu setzen. Die Möglichkeiten sind hier:

- Die Möglichkeit 1 besteht darin, als ersten und einzigen Parameter die Anzahl an Millisekunden seit dem 01. Jan. 1970 00:00:00 Uhr zu notieren. Da JavaScript intern jeden Datumswert als eine Millisekundenzahl speichert, ist dies sozusagen die Grundform aller Daten. Diese Variante wird in der Regel dann verwendet, wenn Datumswerte z. B. aus anderen Programmiersprachen übernommen werden, die ebenso in Millisekunden rechnen.
- Die zweite Möglichkeit besteht darin, einen Datumsstring anzugeben, aus dem dann der Computer das zu setzende Datum herleiten kann. Damit eben dies der Fall ist, muss der String z. B. wie folgt aufgebaut sein: »Mon Apr 2 22:23:26 UTC+0200 2007«.
- Als dritte Möglichkeit können die Werte für Jahr, Monat und Tag sowie optional für Stunde, Minute, Sekunde und Millisekunde jeweils als einzelner Parameter übergeben werden (in dieser Reihenfolge). Dabei ist zu beachten, dass der Januar den Wert 0 und nicht 1 besitzt und somit alle Monate um 1 subtrahiert werden müssen (der 11. November wäre somit im übertragenen Sinne nicht der 11.11., sondern der 11.10.).

```
a = new Date(1176804794015);
alert(a.toString()); // Tue Apr 17 12:13:14 UTC+0200 2007

a = new Date('Tue Apr 17 12:13:14 UTC+0200 2007');
alert (a.toString()); // Tue Apr 17 12:13:14 UTC+0200 2007

a = new Date(2007,3,17);
```

```
alert (a.toString()); // Tue Apr 17 00:00:00 UTC+0200 2007

a = new Date(2007,3,17,12);
alert (a.toString()); // Tue Apr 17 12:00:00 UTC+0200 2007

a = new Date(2007,3,17,12,13);
alert (a.toString()); // Tue Apr 17 12:13:00 UTC+0200 2007

a = new Date(2007,3,17,12,13,14);
alert (a.toString()); // Tue Apr 17 12:13:14 UTC+0200 2007

a = new Date(2007,3,17,12,13,14,15);
alert (a.toString()); // Tue Apr 17 12:13:14 UTC+0200 2007
```

Darüber hinaus kann das Datum mit diversen Funktionen gesetzt werden. Hier einige Beispiele:

```
a = new Date();
a.setYear(2007);
a.setMonth(3);
a.setDate(17);
a.setHours(12);
a.setMinutes(13);
a.setSeconds(14);
a.setMilliseconds(15);
```

5.2.2 Die Uhrzeit anzeigen

Ebenso wie das `Date`-Objekt diverse Funktionen zum Setzen des Datums bereitstellt, werden natürlich auch zahlreiche Funktionen zum Auslesen der einzelnen Datumswerte zur Verfügung gestellt. Um dem Besucher etwa die aktuelle Uhrzeit anzuzeigen, müssen jeweils nur ein neues `Date`-Objekt erstellt und die Zeitangaben ausgelesen werden:

```
function zeitanzeige()
{
 jetzt = new Date();
 zeit = jetzt.getHours()+':'+
        jetzt.getMinutes()+':'+
        jetzt.getSeconds();
 window.status = 'Es ist '+zeit+' Uhr';
}
window.setInterval('zeitanzeige()',1000);
```

Die `status`-Eigenschaft des `window`-Objekts bewirkt hier, dass der übergebene String in der Status-Leiste des Browserfensters angezeigt wird. Die `setInterval`-Methode wiederum bewirkt, dass die notierte Funktion im Sekundentakt aufgerufen und die Uhrzeit damit fortlaufend angezeigt wird.

5.3 Arrays

Arrays sind eine besondere Objekt- bzw. Datentyp-Form. Im Gegensatz zu normalen Objekten speichern sie Daten nicht innerhalb von Eigenschaften, sondern als eine durchnummerierte Liste von quasi beliebiger Länge. Damit ist es möglich, Daten sortiert abzulegen und ebenso wieder abzurufen. In den meisten Fällen sind Arrays die einzige Möglichkeit, mit Daten umzugehen, da der Inhalt eines Arrays jederzeit erweitert werden kann, ohne dabei die Logik des Arrays durcheinander zu bringen.

Ebenso wie jedes andere Objekt wird auch ein Array zunächst mit dem Aufruf `new Array()` erstellt und kann dann gefüllt werden. Im Gegensatz zu einem Objekt werden die Felder eines Arrays jedoch über einen Index angesprochen. Der Index des ersten Feldes beträgt hierbei 0, der des zweiten Feldes 1, der des dritten 2 usw. und wird jeweils innerhalb von eckigen Klammern notiert, die sich an den Arraynamen schmiegen. Beispiel:

```
a = new Array();
a[0] = 'erstes Feld';
a[1] = 'zweites Feld';
a[133] = '134. Feld';
z = a[1]; //Inhalt von Feld 2 abrufen
```

Ein Feld kann dabei wie eine normale Variable jeden Datentyp und sogar weitere Arrays enthalten.

5.3.1 Mit Arrays arbeiten

Der Vorteil, den Arrays gegenüber anderen Objekten bieten, besteht darin, dass alle Daten über einen Index statt über eine Eigenschaft abgefragt werden können. Damit ist es möglich, ein Array etwa innerhalb einer `for`-Schleife zu durchlaufen und alle Werte herauszuholen, zu berechnen oder andere Dinge damit zu tun.

```
umsatz = new Array(
                   4223.23,
                   789.03,
                   8373.62,
                   7223.55,
                   3099.62
                   );
gesamt = 0;
for(i=0; i<umsatz.length; i++)
{
 gesamt += umsatz[i];
}
alert('Gesamtumsatz: '+gesamt.toFixed(2)+' EUR');
```

5.3.2 Mehrdimensionale Arrays

Da ein einfaches Array in manchen Fällen noch zu wenig ist, kann man sich mit einem kleinen Umweg leicht ein mehrdimensionales Array schaffen, um so etwa Tabellen (2D) oder Raum-Koordinaten (3D) abbilden zu können. Hierfür nutzt man aus, dass der Inhalt eines Array-Feldes jeglichen Datentyp, also auch ein Array selbst, annehmen kann. Um nun ein zweidimensionales Array zu bekommen, wird einfach jedes Feld eines eindimensionalen Arrays mit einem weiteren Array bestimmter Länge gefüllt. Benötigt man noch eine Dimension mehr, werden auch die Felder der neu eingefügten Arrays wiederum mit Arrays gefüllt usw. Beispiel:

```
matrix1 = new Array();
matrix1[0] = new Array(1,2,3);
matrix1[1] = new Array(3,4,5);
matrix1[2] = new Array(5,6,7);

matrix2 = new Array();
matrix2[0] = new Array(5,6,7);
matrix2[1] = new Array(7,8,9);
matrix2[2] = new Array(9,0,1);

function matrizenaddition(m1, m2)
{
 matrix = new Array();
 for(i=0;i<m1.length;i++)
 {
  matrix[i] = new Array();
  for(j=0;j<m2[i].length;j++)
  {
   matrix[i][j] = m1[i][j]+m2[i][j];
  }
 }
 return matrix;
}

m = matrizenaddition(matrix1, matrix2);
```

5.3.3 Arrays sortieren

Für das Sortieren von Daten eigenen sich Arrays ganz besonders – vor allem weil es schon eine vordefinierte Funktion zum Sortieren von Arrays gibt. Eindimensionale Arrays können leicht über die `sort`-Methode sortiert werden. Sie sortiert das gegebene Array nach dem Alphabet. Aber Vorsicht: Große Buchstaben werden hier vor kleinen Buchstaben eingereiht und Zahlen werden ebenfalls wie Zeichen und nicht nach ihrem Zahlenwert behandelt. Das folgende Beispiel sortiert also nicht 3, 11, A, a, B, b sondern 11, 3, A, B, a, b – was unter Umständen zu Problemen führen könnte:

```
a = new Array();
a[0] = 'a';
a[1] = 'b';
a[2] = '11';
a[3] = 'A';
a[4] = 'B';
a[5] = '3';
a.sort(); // sortiert a nach: 11 3 A B a b
```

Soll die Sortierung umgestellt werden, kann man sich etwas behelfen. Hier ist es möglich, der `sort`-Methode eine Funktion zu übergeben, die für den Vergleich zweier Daten zuständig ist. Bei jedem Sortiervorgang ruft die Methode die definierte Funktion auf und übergibt ihr die zwei zu vergleichenden Werte. Die Funktion kann nun entscheiden, welcher größer, gleich oder kleiner ist und entsprechend einen Wert zurückgeben. Ist der erste Wert kleiner, sollte die Funktion -1 ausgeben, sind die Werte gleich sollte die Funktion 0 ausgeben, und ist der zweite Wert kleiner sollte die Funktion 1 ausgeben. Das obige Beispiel richtig (nach Wertigkeit) sortiert, könnte z. B. so aussehen:

```
function vergleiche(x,y)
{
 x = x.toLowerCase();
 y = y.toLowerCase();
 //zwei Zahlen; Hinweis: isNAN prüft ob x keine Zahl ist
 if(isNaN(x) == false && isNaN(y) == false){res = x - y;}

 //Zahl und String
 else if(isNaN(x) == false && isNaN(y) == true){res = -1;}
 else if(isNaN(y) == false && isNaN(x) == true){res = 1;}

 //zwei Strings
 else if(x < y){res = -1;}
 else if(y < x){res = 1;}

 //zwei gleiche Strings
 else {res = 0;}
 return(res)
}

a = new Array();
a[0] = 'a';
a[1] = 'b';
a[2] = '3';
a[3] = 'A';
a[4] = 'B';
a[5] = '11';
a.sort(vergleiche);
```

... ergibt die richtige Reihenfolge 3, 11, a, A, b, B.

Etwas schwieriger wird es da schon bei zweidimensionalen Arrays. Hierfür muss im Einzelfall je nach benötigter Sortierung die Sortierfunktion angepasst werden. Das folgende Beispiel zeigt eine einfache Sortierfunktion, welche die Daten zeilenweise nach Spalte 2 sortiert (wenn man davon ausgeht, dass das oberste Array die Zeilen darstellt):

```
function vergleiche(x,y)
{
 return x[2] - y[2];
}

a = new Array();
a[0] = new Array(7,8,9);
a[1] = new Array(4,5,6);
a[2] = new Array(1,2,3);
a.sort(vergleiche);
```

5.4 Formulare kontrollieren

Das Auswerten von Formulardaten gehört zum Einmaleins von JavaScript und ist eine der am meisten benutzen Fähigkeiten, die JavaScript zu bieten hat. Ein Grund: Man kann noch vor dem eigentlichen Absenden des Formulars möglichen Fehlern leicht auf die Schliche kommen und damit dafür sorgen, dass am Server (also wenn das Formular tatsächlich abgeschickt wurde) möglichst nur Daten ankommen, die auch einigermaßen korrekt sind.

5.4.1 Zugriff auf Formulare

Um ein Formular verarbeiten zu können, muss man erst einmal Zugriff darauf haben. Jedes Formular ist auf mehrere Arten ansprechbar:

- `document.forms[x]`, wobei `x` der Index des Formulars ist. Das erste Formular (`<form ...>`) in der Webseite (im Code am weitesten vorn) trägt den Index 0, das zweite 1, das dritte 2 usw.
- `document.Formularname`, wobei `Formularname` durch den Namen (`<form name="Formularname" ...>`) ersetzt wird. Hierbei ist zu beachten, dass, sofern es mehrere Formulare in einem Dokument gibt, jedes einen eindeutigen Namen tragen muss.
- Via `this`-Anweisung kann das Formular direkt aus dem `onsubmit`-Ereignis heraus adressiert werden (`<form onsubmit="return machwas(this)" ...>`).

Üblicherweise wird die zweite Schreibweise verwendet – es spricht jedoch auch nichts gegen die beiden anderen.

5.4.2 Zugriff auf Formularelemente

Hat man Zugriff auf das Formular, kann man von dort aus auf die Formularelemente (Eingabefelder, Buttons, Checkboxen usw.) zugreifen. Dies kann wiederum entweder über den Index des Elements (`document.formular.elements[x]`) geschehen, wird allerdings üblicherweise über den Namen des Elements gehandhabt. Hierzu wird ebenso wie beim Formular selbst dem Element ein eindeutiger (!) Name gegeben, über den es dann ansprechbar ist:

```
<script language="JavaScript">
<!--
 function sagan()
 {
  alert('Deine E-Mail ist: '+document.form1.email.value);
 }
//-->
</script>
<form name="form1">
 E-Mail: <input type="Text" name="email" value="">
 <input type="button" onclick="sagan()" value="Sag an!">
</form>
```

5.4.3 Formular überprüfen

Je nachdem, welche Formularelemente verwendet wurden, unterscheidet sich die Überprüfung ein wenig. Die wichtigsten sollen hier kurz vorgestellt werden. Als Grundlage verwenden wir folgendes Formular:

```
<form onsubmit="return pruefen()" method="post"
      action="absenden.php" name="form1">
 Name: <input type="Text" name="name" value=""><br>
 E-Mail: <input type="Text" name="email" value=""><br>
 <br>
 Passwort: <input type="Password" name="pass1"><br>
 Wiederholung: <input type="Password" name="pass2"><br>
 <br>
 Newsletter empfangen:
 <input type="Radio" name="news" value="ja"> Ja /
 <input type="Radio" name="news" value="nein"> Nein<br>
 <br>
 Wie sind Sie auf uns gekommen?<br>
 <select name="quelle" size="1">
  <option value="-">- Bitte wählen -</option>
  <option value="Google">Google</option>
  <option value="Freund">Empfehlung von Freund</option>
  <option value="Werbung">Werbung</option>
```

```
  <option value="Sonstiges">Sonstiges</option>
 </select><br>
 <br>
 <input type="Checkbox" name="agb" value="ja">
  Ja, ich akzeptiere die AGB.<br>
 <br>
 <input type="Submit" value="Anmelden ...">
</form>
```

5.4.4 Eingabefelder

Besonders bei Kontaktanfragen ist meist die Eingabe eines Namens unumgänglich. Um zu prüfen, ob auch wirklich alle Felder ausgefüllt sind, wird einfach geschaut, ob sie einen nichtleeren Inhalt (`value`-Eigenschaft) besitzen:

```
function pruefen()
{
 res = true;
 a = document.form1;
 if(a.name.value == ''){res = false;}
 if(a.email.value == ''){res = false;}
 //... weitere Prüfungen ...

 if(!res)
 {alert('Bitte Formular vollständig ausfüllen!');}
 return res;
}
```

5.4.5 E-Mail-Felder

Bei Feldern, in die E-Mail-Adressen eingegeben werden sollen, sollte neben dem Inhalt auch geprüft werden, ob es sich dabei wirklich um eine E-Mail-Adresse handelt. Handelt es sich um eine E-Mail-Adresse, so steht mindestens ein Buchstabe vor dem `@`-Zeichen. Darüber hinaus sind drei weitere Buchstaben, ein Punkt und nochmals mindestens zwei Buchstaben danach vorhanden. Wem das zu komplex ist, der kann auch

a) lediglich prüfen, ob ein `@`-Zeichen vorhanden ist und der Inhalt mindestens 8 Zeichen lang ist, oder

b) mit sogenannten regulären Ausdrücken eine exakte Prüfung vornehmen.

Erstere Lösung könnte wie folgt aussehen:

```
if(a.email.value.indexOf('@') == -1){res = false;}
if(a.email.value.length < 8){res = false;}
```

Da der Inhalt eines Textfeldes (`value`) ein String ist, lassen sich auf ihn die Methoden des String-Objekts anwenden. Die `indexOf`-Methode fragt ab, an welcher Stelle im Text ein bestimmter anderer Text (hier das `@`-Zeichen) steht. Der Wert `-1` signalisiert hier, dass das `@`-Zeichen nicht vorhanden ist. Anschließend wird mit der `length`-Eigenschaft die Länge (Anzahl der Zeichen) geprüft.

Die zweite und kompliziertere Lösung könnte z.B. so aussehen:

```
a = document.form1;
reg = new RegExp('^([a-zA-Z0-9\\-\\.\\_]+)'+    //Name
                 '(\\@)'+                       //@-Zeichen
                 '([a-zA-Z0-9\\-\\.]+)'+        //Domain
                 '(\\.)'+                       //Punkt
                 '([a-zA-Z]{2,4})$');           //TLD
if(reg.test(a.email.value) == false){res = false;}
```

Bei sogenannten regulären Ausdrücken handelt es sich um besondere String-Konstrukte, die dem RegExp-Parser (sozusagen der String-Prüfer) genaue Anweisungen geben, wonach er suchen soll. In diesem Fall soll er also genau nach den Zeichen suchen, die üblicherweise in einer E-Mail-Adresse vorkommen (eben mindestens ein Buchstabe, ein `@`-Zeichen sowie eine Internetdomain).

5.4.6 Passwortfelder

Passwortfelder werden in JavaScript genauso behandelt wie normale Eingabefelder – es ist sogar möglich, das Passwort auszulesen und mit einem gespeicherten String zu vergleichen. Bei Registrierformularen werden zudem üblicherweise zwei Passwortfelder angegeben: Um Fehleingaben vorzubeugen, wird das Passwort noch mal wiederholt und mit der ersten Eingabe verglichen – stimmen die Eingaben nicht überein, hat sich der User wohl vertippt. Beispiel:

```
a = document.form1;
if(a.pass1.value != a.pass2.value)
{
 alert('Das eingegebene Passwort stimmt nicht mit der '+
       'Wiederholung überein!');
 res = false;
}
```

5.4.7 Radio-Buttons und Checkboxen

Sind Radio-Buttons nicht von vornherein markiert, dann muss getestet werden, ob einer der Buttons beim Absenden des Formulars eine Markierung enthält. Da Radio-Buttons alle den gleichen Namen tragen, ergibt sich bei JavaScript automatisch ein Array für diese Buttons. Soll auf einen Button zugegriffen werden, so kann dies also über den

Index des Buttons geschehen: `document.form1.news[x]...` . Werden alle Buttons durchlaufen, kann getestet werden, ob einer markiert wurde (und ggf. welcher das ist):

```
a = document.form1;
r = a.news;
s = false;
for(i=0;i<r.length;i++)
{
 if(r[i].checked == true){ s = true; break; }
}
if(!s){res = false;}
```

Bei Checkboxen verhält es sich ähnlich: Auch hier wird die `checked`-Eigenschaft geprüft, nur eben ohne dabei ein Array zu durchlaufen:

```
if(!a.agb.checked){res = false;}
```

5.4.8 Select-Felder

Im Beispiel im Abschnitt 5.4.3 (»Formular überprüfen«) ist auch ein `Select`-Feld definiert, über das die Quelle des Besuchs ausgewählt werden kann. Damit der Besucher auch auf jeden Fall eine Quelle auswählt (die Standardauswahl ist ja »Bitte wählen«), sollte überprüft werden, welcher Eintrag gewählt wurde. Auf das `Select`-Feld kann wie bei allen Feldern über den Namen zugegriffen werden – die einzelnen Einträge sind über das `options`-Array abrufbar. Da uns in diesem Beispiel ausreicht zu wissen, dass nicht der erste Eintrag (»Bitte wählen«) gewählt wurde, brauchen wir nur zu prüfen, ob die Eigenschaft `selectedIndex` des `Select`-Feldes größer 0 (Null) ist. Die Eigenschaft gibt den Index des ausgewählten Eintrags an: Wurde der erste ausgewählt, gibt sie 0 aus, für den zweiten 1, für den dritten 2 usw. Wurde kein Eintrag markiert, gibt sie den Wert -1 zurück. In jedem Fall, in dem eine korrekte Auswahl erfolgte, ist der Wert dieser Eigenschaft als größer als 0. Der dafür zuständige Code sieht dann in etwa so aus:

```
if(a.quelle.selectedIndex > 0){res = false;}
```

5.4.9 Das komplette Skript

Das komplette Skript könnte in diesem Fall dann also wie folgt aussehen:

```
function pruefen()
{
 res = true;
 a = document.form1;
 if(a.name.value == ''){res = false;}
 if(a.email.value == ''){res = false;}
 if(a.email.value.indexOf('@') == -1){res = false;}
```

```
if(a.email.value.length < 8){res = false;}
reg = new RegExp('^([a-zA-Z0-9\\-\\.\\_]+)'+    //Name
                 '(\\@)'+                       //@-Zeichen
                 '([a-zA-Z0-9\\-\\.]+)'+        //Domain
                 '(\\.)'+                       //Punkt
                 '([a-zA-Z]{2,4})$');           //TLD
if(reg.test(a.email.value) == false){res = false;}
if(a.pass1.value != a.pass2.value)
{
 alert('Das eingegebene Passwort stimmt nicht mit der '+
       'Wiederholung überein!');
 res = false;
}
r = a.news;
s = false;
for(i=0;i<r.length;i++)
{
 if(r[i].checked == true){ s = true; break; }
}
if(!s){res = false;}
if(!a.agb.checked){res = false;}
if(a.quelle.selectedIndex > 0){res = false;}
if(!res)
{alert('Bitte Formular vollständig ausfüllen!');}
return res;
}
```

5.5 Fenster & Frames

5.5.1 Daten ins Dokument schreiben

Eine der Haupt-Aufgaben von JavaScript ist es, errechnete Daten oder andere Werte in die Ausgabe, also die Webseite, zu schreiben. JavaScript bietet hierzu die Möglichkeit, das Dokument entweder komplett neu zu schreiben oder während des Öffnungsvorgangs des Dokuments Daten einzufügen. Ein nachträgliches Ändern des bereits angezeigten Dokumentinhalts ist indes nicht ohne Weiteres möglich.

5.5.2 Dokument erweitern

Soll ein Dokument um errechnete Daten erweitert werden, geht dies eigentlich nur während des Öffnen-Vorgangs, also wenn der Computer das Dokument liest, den enthaltenen JavaScript-Code verarbeitet und das Dokument schließlich anzeigt. Das »eigentlich« steht im letzten Satz deshalb, da mit DHTML und Ajax eben genau das Gegenteil erreicht wird – nämlich das nachträgliche Ändern des Dokumentinhalts.

Jan Winkler

JavaScript und Ajax

Das Praxisbuch für Web-Entwickler

Fast alle Webseiten haben eines gemeinsam: Sie bestehen aus JavaScript-, HTML- und CSS-Code. Während HTML und CSS stagnieren, hat sich JavaScript mit Ajax zur dominierenden Web-2.0-Technik weiterentwickelt. Dieses Buch vermittelt anhand konkreter Programmierbeispiele die Grundlagen und einige fortgeschrittene Techniken von JavaScript und Ajax.

Lernen Sie JavaScript kennen:
Statische Websites sind von gestern – mit Hilfe von JavaScript können Sie den Dialog mit den Besuchern Ihrer Seite deutlich verbessern. Die Grundlagen und die Syntax von JavaScript lassen sich schnell erlernen, und mit den zahlreichen Praxisbeispielen aus diesem Buch stellen sich schnell Erfolgserlebnisse ein. Zudem dient die umfassende Objektreferenz als ideales Nachschlagewerk.

Dynamic HTML:
Die DHTML-Technik verknüpft Standard-HTML mit JavaScript und CSS. Sie dient unter anderem dazu, neue Elemente in eine Webseite einzufügen, ohne die komplette Seite neu in den Browser laden zu müssen. So erstellen Sie zum Beispiel ohne großen Aufwand MouseOver-Effekte aller Art. Zudem können Sie Seitenobjekte mit DHTML pixelgenau im Browser platzieren, Elemente ein- und ausblenden und auf Ereignisse wie Mausklicks reagieren.

Komfortable Webanwendungen mit Ajax:
Ajax verleiht Internetapplikationen die Benutzerfreundlichkeit lokal installierter Programme. Dieses Buch verrät Ihnen, wie Sie Ajax in Ihre Seite integrieren, und zeigt Ihnen verschiedene Typen von Ajax-Anwendungen an konkreten Beispielen: Sie erfahren, wie Sie einen interaktiven Warenkorb mit Drag & Drop-Eigenschaften programmieren, eine interaktive Suche entwickeln und ein Chat-Modul realisieren. Außerdem erhalten Sie Einblicke in die wichtigsten Ajax-Frameworks, die Sie bei der Entwicklung solcher Applikationen unterstützen.

Alle Infos stets zur Hand:
Die umfassende Objektreferenz bietet Ihnen einen Überblick über die zentralen Code-Objekte von JavaScript und dessen Pendant JScript, das in Microsofts Internet Explorer zuhause ist. Damit haben Sie den Funktionsumfang von JavaScript stets schnell und zuverlässig zur Hand.

Aus dem Inhalt:

Über den Autor:

Jan Winkler ist erfolgreicher Internet-Publisher und leitet ein großes Online-Vermarktungsunternehmen.
Er lebt in Berlin.

Auf www.buch.cd:

Sämtliche Programmbeispiele zum Download

9 783772 362620

35,– EUR [D]

ISBN 978-3-7723-6262-0