

Auf einen Blick

1	Einleitung	9
2	JavaScript und DOM	17
3	JavaScript und CSS	65
4	JavaScript und OOP	97
5	JavaScript und XML	127
6	JavaScript und HTTP	165
7	JavaScript und Libraries	237
8	Praxisbeispiele	301
9	Google & Yahoo!	375
A	Inhalt der Buch-CD-ROM	423
	Index	425

Inhalt

1	Einleitung	9
1.1	Eine kurze Reise durch die AJAX-Welt	10
1.2	Zielgruppe des Buches	12
1.3	Aufbau des Buches	13
1.4	Hinweise zu den Inhalten	14
1.5	Danksagung	15
1.6	Support zum Buch	15
2	JavaScript und DOM	17
2.1	Der Dokumentenbaum	18
2.1.1	Knoten	19
2.1.2	Konstanten	20
2.1.3	Eigenschaften	21
2.1.4	Attribute	26
2.1.5	Methoden	29
2.1.6	Einfaches Beispiel	35
2.2	Zugriff auf einzelne Elemente	40
2.2.1	Eigenschaften	40
2.2.2	Elemente selektieren	44
2.2.3	Attribute bearbeiten	50
2.2.4	Elemente erzeugen	54
2.2.5	Einfaches Beispiel	56
2.3	Beispielprojekt »Planetensystem«	59
3	JavaScript und CSS	65
3.1	Grundlagen	66
3.2	Style-Eigenschaften	67
3.2.1	getComputedStyle() und currentStyle	71
3.3	StyleSheets-Eigenschaften	72
3.3.1	Praxisbeispiel StyleSwitcher	78
3.3.2	setProperty(), getPropertyValue() und removeProperty()	82
3.3.3	Eigenschaften des styleSheets[]-Arrays	84
3.4	Regeln für StyleSheets	92

4 JavaScript und OOP 97

4.1	Klassen	98
4.2	Eigenschaften	99
4.3	Abfragen	101
4.4	Methoden	102
4.5	Prototypen	105
4.6	Literale	107
4.7	JSON	114
4.8	Praxisbeispiel	118

5 JavaScript und XML 127

5.1	Grundlagen	128
5.2	XML laden	129
5.3	XML parsen	134
5.3.1	XML mit dem DOM parsen	134
5.3.2	Gecko-Browser und das DOM	136
5.3.3	Geckos DOMParser	138
5.4	Beispielprojekt »Buchladen«	142
5.5	Vorschau auf E4X	149
5.5.1	Elemente auslesen	151
5.5.2	Attribute auslesen	154
5.5.3	Filter verwenden	154
5.5.4	Struktur verändern	158
5.5.5	Platzhalter verwenden	160
5.5.6	Elemente löschen	162
5.5.7	Fazit	163

6 JavaScript und HTTP 165

6.1	Grundlagen	166
6.1.1	Client-Request-Methoden	167
6.1.2	Server-Antwortcodes	168
6.1.3	HTTP-Header	173
6.2	XMLHttpRequest	179
6.2.1	Das Objekt erzeugen	180
6.2.2	Methoden	183
6.2.3	Eigenschaften	186
6.2.4	Hallo Ajax	189
6.2.5	ajaxRequest-Klasse	190
6.2.6	Den Ladezustand anzeigen	202
6.2.7	Eine Verbindung unterbrechen	203

6.2.8	Automatische Updates	205
6.2.9	JavaScript ausführen	207
6.2.10	Probleme mit dem Cache	209
6.2.11	AJAX mit JSON	211
6.2.12	Externe Quellen nutzen	212
6.2.13	Das Historie-Problem	219
6.2.14	Beispielprojekt ShoutBox	226

7 JavaScript und Libraries 237

7.1	Prototype	238
7.1.1	ajax.js	239
7.1.2	base.js	251
7.1.3	compat.js	252
7.1.4	dom.js	252
7.1.5	form.js	259
7.1.6	string.js	262
7.2	script.aculo.us	264
7.2.1	Drag&Drop	265
7.2.2	Visuelle Effekte	279
7.3	Behaviour	292

8 Praxisbeispiele 301

8.1	ajaxBooks	301
8.1.1	Die Daten abrufen	309
8.2	ajaxChat	310
8.2.1	Das Anmelden	319
8.2.2	Das Abmelden	320
8.2.3	Beiträge speichern	322
8.2.4	Die Userliste und Beiträge anzeigen	322
8.3	ajaxComplete	323
8.3.1	Lokale Auswahl	325
8.3.2	Formatierungen	327
8.3.3	Auswahl per AJAX	328
8.4	ajaxDict	332
8.4.1	Erklärung abrufen und eintragen	338
8.5	ajaxDir	341
8.5.1	Auslesen der Verzeichnisstruktur	350
8.5.2	Anzeige des Dateiinhalts	351
8.5.3	Zippen der Packliste	351
8.6	ajaxPass	353
8.6.1	Die Passwortsicherheit überprüfen	357
8.6.2	Zufällige Passwörter erzeugen	358

8.7	ajaxTic	359
8.7.1	Die XML-Datei erzeugen	370
8.7.2	Die XML-Datei aktualisieren	373

9 Google & Yahoo! 375

9.1	Google Maps	376
9.1.1	Grundlagen	376
9.1.2	Steuerelemente	381
9.1.3	Markierungspunkte	382
9.1.4	Detailinformationen	387
9.1.5	Eventmodell	390
9.1.6	Linien zeichnen	395
9.1.7	AJAX	397
9.1.8	Beispielanwendung	398
9.2	Yahoo! Maps	403
9.2.1	Grundlagen	405
9.2.2	Steuerelemente	407
9.2.3	Markierungspunkte	410
9.2.4	Detailinformationen	413
9.2.5	Beispielanwendung	416

A Inhalt der Buch-CD-ROM 423

Index 425

2 JavaScript und DOM

Die Zeit ist reif für ein neues Verständnis der Webprogrammierung. Offizielle Standards haben sich entwickelt, die zum großen Teil von aktuellen Browsern gleichermaßen interpretiert werden. Das Document Object Model ist ein wesentlicher Baustein, wenn nicht sogar das Fundament moderner Webanwendungen. Das WWW bricht auf in die nächste Generation.

Das W3C Document Object Model (DOM)¹ ist eine plattform- und sprachunabhängige Schnittstelle für den Zugriff auf XML- und XHTML-Dokumente. Mit dem DOM können Sie dynamisch auf einzelne Elemente eines Dokuments zugreifen, diese auswerten, bearbeiten und sogar löschen. Für diese Aufgaben wird das Dokument als hierarchisch geordnete Struktur erfasst, was wiederum eine korrekte Anwendung bestehender Formate erfordert. Eine Seite, die auf diese Weise bearbeitet wird, muss nicht einmal neu geladen werden, um vorgenommene Änderungen anzuzeigen, diese werden sofort dargestellt. Für die Arbeit mit dem DOM an einer Webseite dient als Schnittstelle die Skriptsprache JavaScript, welche im ECMA-262 Standard² definiert ist.

Historisch betrachtet ist das DOM aus der Skriptsprache JavaScript³ hervorgegangen, die seit ihrer Einführung zunehmend an Bedeutung gewonnen hat. Anfangs beschränkte sich JavaScript auf sinnvolle Erweiterungen für Formularelemente und ein paar kleine Spielereien wie beispielsweise das dynamische Austauschen von Grafik beim Darüberstreifen mit der Maus, so genannte Rollover-Effekte. Später entstand durch neue Möglichkeiten seitens der großen Browserhersteller Netscape und Microsoft das Dynamic HTML. Hier wurde es erstmals möglich, interaktive Anwendungen zu erzeugen und die Elemente einer Seite über ein Objekt-Modell zu manipulieren und zu animieren. Dies war auch die Zeit der großen Browserkriege, welche zu uneinheitlichen Implementierungen verschiedener Möglichkeiten von JavaScript führten. Für Entwickler war dies eine denkbar ungünstige Zeit, da man immer für mindestens zwei verschiedene Browser programmieren musste.⁴ Der erste DOM-Standard des W3C war der Versuch, eine einheitliche Schnittstelle für dynamische Skriptanwendungen zu definieren. Dar-

1 <http://www.w3.org/DOM/>

2 <http://www.ecma-international.org/publications/standards/Ecma-262.htm>

3 <http://developer.mozilla.org/en/docs/JavaScript>

4 Bedauerlicherweise ist es heutzutage auch nicht besser. Solange sich die Browserhersteller nicht umfassend an den Standards des W3C orientieren, gehört Mehrarbeit leider zum Tagesgeschäft.

aus hervorgegangen ist das heutige DOM⁵, das eine Schnittstelle für den Zugriff auf strukturierte Dokumente im XML-Format⁶ bietet.

Die Möglichkeiten, welche der aktuelle Stand des DOM bietet, sind vielfältig und garantieren bei kreativem Gebrauch das Erstaunen Ihrer Besucher. Anwendungen im Stil traditioneller Applikationen sind nun kein Wunschgedanke mehr. Die Zukunft hat gerade erst begonnen.

In diesem Kapitel erhalten Sie die wesentlichen Grundlagen, um mit dem DOM⁷ eigene Anwendungen mit JavaScript programmieren zu können.

2.1 Der Dokumentenbaum

Eine vollständig übertragene Webseite repräsentiert gemäß dem W3C-DOM-Standard einen hierarchisch geordneten Dokumentenbaum. Die Hierarchie orientiert sich dabei am Aufbau der XHTML-Struktur der Seite. Man kann sich dies als eine Baumstruktur vorstellen, welche wiederum in Knotenelemente aufgeteilt ist. Jeder dieser Knoten beinhaltet ein bestimmtes Element (z.B. das `img`-Tag), die entsprechenden Attribute, den Inhalt und gegebenenfalls eigene zusätzliche Merkmale. Über die DOM-API können Sie nun die jeweiligen Merkmale dieser Objekte auslesen und dynamisch manipulieren. Dabei ist es nicht notwendig, die angezeigte Seite neu zu laden. Wird die Seite verlassen oder wieder neu geladen, sind die zuvor vorgenommenen Änderungen natürlich hinfällig. Genau hier greift AJAX ein, indem es dynamische Änderungen serverseitig abspeichern kann, ohne die Seite neu laden zu müssen. Details dazu erfahren Sie in Kapitel 6, *Javascript und HTTP*.

Technisch betrachtet werden diese Möglichkeiten über das `node`-Objekt realisiert, dessen Merkmale in diesem Abschnitt vorgestellt werden.

Das folgende Beispiel soll Ihnen bei den weiteren Ausführungen dieses Abschnitts helfen, sich eine bessere Vorstellung über die Struktur einer XHTML-Seite und ihrer Ausgabe in DOM machen zu können.

```
<html>
<head>
  <title>JavaScript und DOM</title>
</head>
<body>
<h1>Hallo DOM</h1>
```

5 <http://developer.mozilla.org/en/docs/DOM>

6 <http://www.w3.org/XML/>

7 Viele Fragen zum DOM werden unter <http://www.w3.org/DOM/faq.html> beantwortet.

```

<p>Ein <strong><em>kurzes</em></strong> Beispiel</p>
</body>
</html>

```

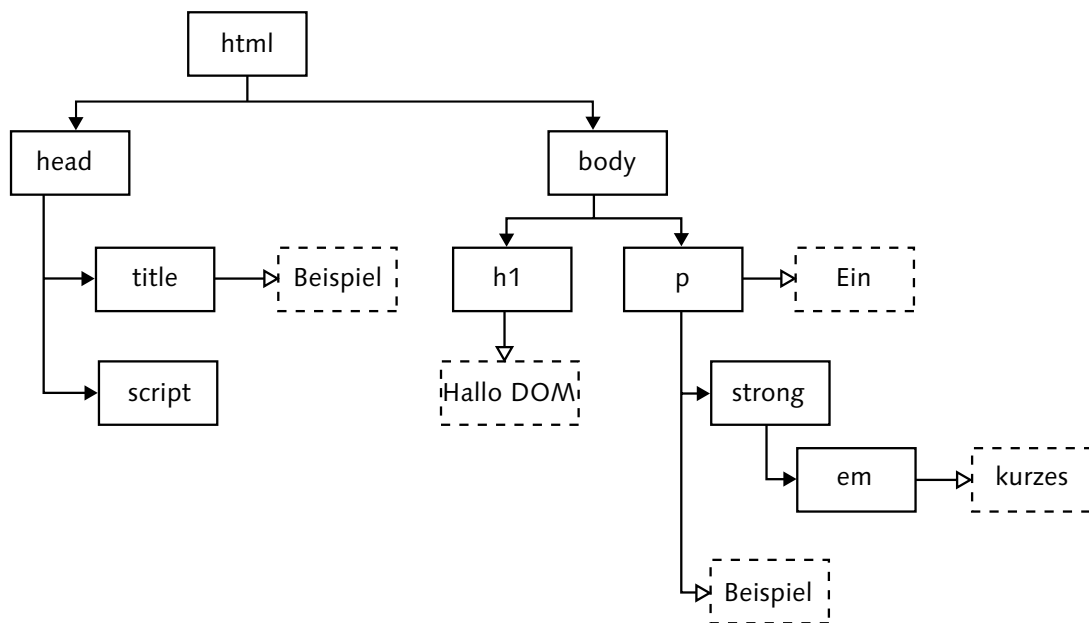


Abbildung 2.1 Darstellung der Beispielseite als Baumstruktur

2.1.1 Knoten

Wie Sie der Abbildung 2.1 entnehmen können, wird das Dokument aus obigem Beispiel in einer Baumstruktur gemäß der XHTML-Auszeichnung dargestellt. Daraus ergeben sich der Reihe nach geordnet acht Elementknoten:

- ▶ html
- ▶ head
- ▶ title
- ▶ body
- ▶ h1
- ▶ p
- ▶ strong
- ▶ em

Diese Elementknoten unterteilen sich wiederum in eine so genannte Mutter-/Kindbeziehung. In unserem Beispiel ergeben sich folgende Abhängigkeiten, die als einzelne Elemente repräsentiert sind.

- ▶ html → head
- ▶ head → title

- ▶ html → body
- ▶ body → h1, p, em
- ▶ p → strong
- ▶ strong → em

Innerhalb der einzelnen Elemente können sich beliebig tief verschachtelte weitere Elementknoten befinden. Jeder Knoten besitzt Konstanten, Eigenschaften und Methoden, mit denen Sie den Dokumentenbaum auslesen und bearbeiten können.

2.1.2 Konstanten

Jeder Knoten besitzt zahlreiche Eigenschaften, auf die wir gleich noch zu sprechen kommen. Eine dieser Eigenschaften lautet `nodeType`, mit ihr können Sie die Art des entsprechenden Elements ermitteln. Darunter zählen in XHTML beispielsweise die Typen `Text` oder `Attribut`, welche zum einen Text-Objekte – also einen Stringwert aus dem eigentlichen Inhalt – und zum anderen die vorhandenen Attribute eines Knotens auflisten. Eine Abfrage würde sich syntaktisch wie folgt darstellen.

```
if (Knoten.nodeType == 8) alert('Ich bin ein Kommentar.');
```

Über den Sinn und Zweck der Abfrage brauchen Sie sich hier noch keine Gedanken zu machen, dieser wird später deutlich werden.

Die Typendefinitionen sind gemäß dem DOM als Konstanten mit festen Werten hinterlegt. Intern ist diese Definition über Eigenschaften in einem `Node`-Objekt hinterlegt. Dies können Sie nutzen, um eine Abfrage auch über Konstanten anstatt in numerischen Werten zu formulieren.

```
if (Knoten.nodeType == Node.COMMENT_NODE) ...
```

Ein Vorteil dabei ist, dass sich dadurch die Lesbarkeit des Quellcodes verbessert, da Sie nun sofort erkennen können, was in der Bedingung abgefragt wird. Leider wird dies nur in Gecko-Browsern und nicht durch den Microsoft Internet Explorer unterstützt. Über eine Helferoutine können Sie aber die Kompatibilität dennoch herstellen. Hierbei wird über ein Literal-Objekt (siehe dazu Kapitel 4, *JavaScript und OOP*) ein eigenes `Node`-Objekt erzeugt, das nur dann zur Anwendung kommt, falls das eigentliche `Node`-Objekt im verwendeten Browser nicht implementiert ist.

```
if (!window.Node)
{
    var Node = {
```

```

ELEMENT_NODE      : 1,
ATTRIBUTE_NODE    : 2,
TEXT_NODE         : 3,
COMMENT_NODE      : 8,
DOCUMENT_NODE     : 9
};
}

```

Die wichtigsten Konstanten und deren Bedeutung für die Arbeit mit JavaScript in Verbindung mit XHTML sind in der folgenden Tabelle kurz dargestellt.

nodeType	Konstante	Beschreibung
1	ELEMENT_NODE	Element
2	ATTRIBUTE_NODE	Attribut
3	TEXT_NODE	Text
8	COMMENT_NODE	Kommentar
9	DOCUMENT_NODE	Dokument

Tabelle 2.1 Knotentypen, die in HTML häufig vorkommen

2.1.3 Eigenschaften

Wie bereits erwähnt, besitzt jeder Knoten des Dokumentbaumes bestimmte Eigenschaften, von denen Sie bereits im Abschnitt zuvor eine kennen gelernt haben. Über die Eigenschaften erhalten Sie wichtige Informationen zu den Beziehungen der Knoten untereinander und natürlich zu einem Knoten selbst.

childNodes[], firstChild, lastChild

Für die Arbeit mit dem Dokumentenbaum müssen Sie zunächst einen Wurzelknoten (den Einstiegsknoten) auswählen. Der oberste Knoten in einem XHTML-Dokument wird durch das `document`-Objekt repräsentiert, welches als erstes Element auf das `HTML`-Tag zeigt.

```

alert(
  "Typ: "+document.firstChild.nodeType+"\n"+
  "Tag: "+document.firstChild.nodeName
);

```

Als Ergebnis dieser Abfrage erhalten Sie für die Eigenschaft `nodeType` den Wert 9, welcher in der Konstante `DOCUMENT_NODE` (siehe obigen Abschnitt) hinterlegt ist. Als Name des Knotens wird der String »HTML« angezeigt. Zur Verdeutli-

Index

`$()` 252
`$F()` 261

A

`abort()` 183, 186
ActiveX 129
ActiveXObject 130, 190
`addClassName` 257
`addRule()` 93
Adressbuch 118
AJAX 9
 `abort()` 186, 203
 ActiveXObject 190
 ajaxRequest-Klasse 190
 Automatische Updates 205
 Cache 209
 Den Ladezustand anzeigen 202
 Eigenschaften 186
 Externe Quellen 212
 GET-Request 196
 Hallo Ajax 189
 HEAD-Request 193
 Historie-Problem 219
 JavaScript ausführen 207
 JSON 211
 Methoden 183
 Microsoft.XMLHTTP 182
 MSXML-Parser 180
 Objekt erzeugen 180
 onreadystatechange 186, 190, 192
 `open()` 184, 191
 POST-Request 199
 `readyState` 186, 190, 203, 205
 `responseText` 187–188, 190, 207
 `responseXML` 187–188, 242
 `send()` 185
 `setRequestHeader()` 185
 Shoutbox 226
 `status` 187, 190, 204
 `statusText` 187, 204
 Verbindung unterbrechen 203
 W3C 180
 XMLHTTP 179–180
 XMLHttpRequest 141, 179, 190
Ajax blog 12

Ajax Info 12
Ajax Matters 12
Ajax.Autocompleter 325, 328
Ajax.PeriodicalUpdater 248
Ajax.PeriodicalUpdater() 313, 365
Ajax.Request 239
Ajax.Updater 244
Ajaxian 12
ajaxRequest-Klasse 190
 `abortReq` 204
 `doRequest()` 191, 193
 `setRequestHeader()` 192
 `wState` 191
 `wState()` 193, 195
Allgemeine Header 176
Amazon E-Commerce Service 302
anonyme Funktion 196
Antwortcodes 168
`appendChild` 135, 146
`appendChild()` 30–31, 56, 228
application/x-www-form-urlencoded
 199
Archive_ZIP 352
Array
 `ismember()` 303, 343
 `remove()` 304
Arrayoperator 100
assoziatives Array 28, 100
async 131–132
Attribute 20, 26, 50
ATTRIBUTE_NODE 21, 26
attributes 25, 27, 52
Attributknoten 26
Autocomplete 323
Autocompleter.Base 325
Autocompleter.Local 325
Autocompleter-Klasse 324
Automatische Updates 205

B

Base64 355
Baumstruktur 38
Behaviour 292
body-Element 22
Buchladen 142

C

Cascading Style Sheets 42, 65
CDATA 29
CGI 165
childNodes 134–135, 144
clearTimeout() 205
Client-Request 167
cloneNode() 30–31
#comment 26
COMMENT_NODE 20–21, 26
Common Gateway Interface 165
Conditional Comments 222
createAttribute() 53, 55
createDocument() 131–132
createElement 55, 135, 146
createElement() 55, 228
createRange() 334
createTextNode 147
createTextNode() 56, 228, 388
CSS 65
 addRule() 93
 after 77
 alternative CSS 82
 before 77
 color 74
 cssQuery() 87
 cssRules 83
 cssText 83, 85–86
 currentStyle 71
 deleteRule() 92
 disabled 87
 getComputedStyle() 71
 getPropertyValue() 72, 82
 href 84
 IEtoW3C 84
 insertRule() 93
 left 69
 Maßeinheit 69
 media 88
 Notationsregel 67
 position 70
 Positionsangaben 67
 Pseudoelemente 77
 Regeln für StyleSheets 92
 Regel-Objekt 74
 removeProperty() 82
 removeRule() 92
 Schriftfarbe 68
 selectorText 86

Selektor 87
setProperty() 82
style 71, 82, 85
Style-Eigenschaften 67
style-Objekt 66, 69, 71, 82
StyleSwitcher 75, 78
title 88
top 69
type 88
cssQuery() 87
cssRules 83
CSS-Selektoren 296
cssText 83, 86
currentStyle 71

D

Datenaustausch 115
Datentyp xml 151
defaultView 41
DELETE 167
delete 102, 162
deleteRule() 92
disabled 87
DOCTYPE 29, 132
doctype 41
#document 26
Document Object Model 17
Document Type Definition 28, 128
document.implementation 131
DOCUMENT_NODE 21, 26
documentElement 41, 130, 134, 139, 144
document-Objekt 21, 40
Dokumentenbaum 18
DOM 17
 #comment 26
 #document 26
 #text 23, 26, 56, 136
 addRule() 93
 appendChild() 30–31, 56, 146, 228
 Attribute 20, 26, 50
 Attribute bearbeiten 50
 ATTRIBUTE_NODE 21, 26
 attributes 25, 27, 52
 Attributknoten 26
 Baumstruktur 38
 body-Element 22
 cloneNode() 30–31
 COMMENT_NODE 21, 26
 createAttribute() 53, 55

createElement() 55, 146, 228
createTextNode() 55–56, 228, 388
currentStyle 71
defaultView 41
deleteRule() 92
doctype 41
DOCUMENT_NODE 21, 26
documentElement 41
document-Objekt 21
Dokumentenbaum 18
DOM-API 18
Eigenschaften 20–21, 40
ELEMENT_NODE 21, 23, 25, 136–137
Elemente erzeugen 54
Elemente selektieren 44
Elementknoten 19
firstChild 21–23, 45, 145, 230
getAttribute() 51–52, 139, 145
getAttributeNode() 51–52
getComputedStyle() 71
getElementById() 40, 44, 68–69, 146, 190
getElementsByClassName() 47
getElementsByName() 44–45
getElementsByTagName() 44, 46, 67, 134, 146
getPropertyValue() 82
hasAttribute() 51–52
hasAttributes() 30–31
hasChildNodes() 30, 36
hasChildNodes() 30
implementation 41
innerHTML 43
insertBefore() 30, 32
insertRule() 93
Kindelemente 31
Kindknoten 22
Knoten 19
Konstante 21
Konstanten 20
lastChild 21–23, 30
Methoden 29
Mutter-/Kindbeziehung 19
nextSibling 24
Node Tree Viewer 36
nodeName 23, 37
Node-Objekt 20
node-Objekt 18, 40
nodeType 20–21, 23, 38, 137, 145, 230
nodeValue 23, 37, 43, 45, 145, 230
ownerDokument 25
parentNode 25
Planetensystem 59
previousSibling 24
removeAttribute() 51, 53, 77
removeAttributeNode() 51, 53
removeChild() 30, 33, 145
removeProperty() 82
removeRule() 92
replaceChild() 30, 34
setAttribute() 51, 53, 77, 79
setAttributeNode() 51, 53
setProperty() 82
styleSheet-Objekt 42
styleSheets 41–42
tagName 49
Text 20
TEXT_NODE 21, 23, 26, 56
Text-Objekte 20
Undo-Funktion 33
Wurzelknoten 22, 39
Zugriff auf einzelne Elemente 40
zusätzliche Attribut 27
DOM Node Tree Viewer 36
DOM-API 18
DOMParser 138, 140
DOMParser() 134
Drag&Drop 265, 302, 307, 312, 341
DTD 28–29, 128
Dynamic HTML 17

E

E4X 127, 149, 189
 @-Zeichen 154
 Attribute auslesen 154
 Elemente auslesen 151
 Elemente löschen 162
 Filter 154
 for each 152
 length() 151
 Platzhalter 160
 rekursiver Operator 152
 Struktur verändern 158
 Vergleichsoperanden 155
 Wildcard-Zeichen 153, 159
ECMA-262 Standard 17
ECMAScript 107
ECMAScript for XML 149

ECS 302
 eigene Datentypen 108
 Eigenschaften 20–21, 40
 ELEMENT_NODE 21, 23, 25, 136–137
 Element-Knoten 19
 empty() 335
 enablePrivilege() 214
 Entity-Header 176
 escape() 198
 Escape-Sequenz 139
 eval() 117, 207
 Event
 onload 69, 79
 Externe Quellen 212

F

File_Archive 352
 firstChild 21–23, 45, 139, 145, 148, 230
 Flickr 10
 for each 152
 for/each 160
 for/in-Schleife 100, 110
 Fußnoten 14

G

Geocoding 378
 Geokoordinaten 399
 GET 167, 174, 196, 242
 GET oder POST 202
 getAllheaders() 185
 getAllResponseHeaders() 183, 186
 getAttribute 139
 getAttribute() 51–52, 139, 145
 getAttributeNode() 51–52
 getComputedStyle() 71
 getElementById 146
 getElementById() 40, 44, 68–69, 121, 190
 getElementsByClassName() 47, 252
 getElementsByTagName() 44–45
 getElementsByTagName() 134–135, 139, 146
 getElementsByTagName() 44, 46, 67, 121, 134
 getPropertyValue() 72, 82
 GET-Request 196
 getResponseHeader() 183, 186
 getSelection() 334

Globale Variablen 104
 Google Maps 10, 375
 addControl() 381
 addListener() 391
 addOverlay() 382
 Ajax 397
 Beispielanwendung 398
 Bewegung 380
 centerAndZoom() 378
 clearListeners() 392
 clearOverlays() 383
 Detailinformationen 387
 Drag&Drop 380
 Eigene Grafiken 383
 Entwickler-ID 376
 Eventmodell 390
 Events und Markierungspunkte 393
 GEvent-Klasse 390
 GIcon-Klasse 383
 GMap-Klasse 378
 GPoint-Klasse 378
 GPolyline 395
 GXmlHttp-Klasse 398
 Linien zeichnen 395
 Markierungspunkte 382
 openInfoWindow() 388
 recenterOrPanToLatLng() 380
 removeListener() 392
 removeOverlay() 383
 showMapBlowup() 389
 Sprechblase 387
 Steuerelemente 381
 VML-Namensraum 395
 Zoomlevel 379
 Google Suggest 10, 324
 GoogleMapi 403

H

Hallo Ajax 189
 hasAttribute() 51–52
 hasAttributes() 30–31
 hasChildNodes() 30, 36
 hasChildNotes() 30
 hasClassName 256
 Hash-Struktur 144
 HEAD 167, 193
 header 80
 header() 189

Headerinformationen 166, 174
HEAD-Request 193
Hexadezimal 86
Historie 220
Historie-Problem 219
href-Eigenschaft 75–76
HTML_AJAX 118
HTTP 165
 Allgemeine Header 176
 Antwortcodes 168
 DELETE 167
 Entity-Header 176
 GET 167, 174, 196, 242
 HEAD 167, 193
 Headerinformation 174
 OPTIONS 167
 POST 167, 199
 PUT 167
 Request-Header 178
 Response-Header 177
 TRACE 168
 Wertebereich 168
HTTP-Header 173, 185
HTTP-Request 242
Hypertext Transfer Protocol 166

I

implementation 41
#IMPLIED 29
in 101
innerHTML 33, 43, 121
insertBefore() 30, 32
insertRule() 93
instanceof 102
Instanz 99
Instanzen 103
ismember() 303, 342
ISO-8859-1 128

J

JavaScript Object Notation 114
JSON 114, 120, 188, 211, 305, 346
 HTML_AJAX 118
 JSON-PHP 118
 parse() 117, 212
 php-json 118
 stringify() 117, 212
JSON-Struktur 117

K

Kindelemente 31
Kindknoten 22
Klassen 98
Knoten 19
komplexe Strukturen 108
Konstanten 20–21
Konstruktorfunktion 98

L

lastChild 21–23, 30
Last-Modified 195
Leerzeichen 24, 136
length 90, 117
length() 151
length-Eigenschaft 134
Literale 107
Literal-Objekt 20, 60, 107, 115, 120, 144,
 211, 227, 293, 304, 334, 355, 417
load 130
loadXML() 144

M

match() 57–58
mebbo 11
media 88
Methoden 29
Microsoft.XMLHTTP 182
MSXML-Parser 180
Mutter-/Kindbeziehung 19

N

Netvibes 10
new 98
nextSibling 24
nodeName 23, 37
node-Objekt 18, 20, 40
nodeType 20–21, 23, 38, 135, 137, 145,
 230
nodeValue 23, 37, 43, 45, 139, 145, 230
null 101, 105

O

ObjectGraph Dictionary 11
Object-Objekt 107
Objekteigenschaft 101, 104
Objektinstanz 98
Objektklasse 102

- Objektmethoden 103
- Objektvariable 98
- onkeyup 353
- onload 121, 149, 189, 308, 318, 330, 349, 370
- onload-Event 132
- onreadystatechange 186, 190, 192
- onunload 318
- OOP 97
 - Closures* 126
 - eigene Datentypen* 108
 - Eigenschaften* 99
 - JSON* 114
 - Klassen* 98
 - komplexe Strukturen* 108
 - Literale* 107
 - Methoden* 102
 - Object-Objekt* 107
 - prototype* 105
 - Prototypen* 105
 - Vererbung* 126
- OOP-Notation 97
- open() 183, 191
- openGeoDb-Projekt 399
- Operator
 - delete* 102, 162
 - in* 101
 - instanceof* 102
 - new* 98
 - typeof* 101
- OPTIONS 167
- ownerDokument 25

P

- Parameter 103
- parentNode 25
- parse() 212
- parseFromtString() 138
- parseInt() 70
- PasswordStrength 358
- Passwort Tipp 354
- PclZip 352
- PEAR
 - Archive_ZIP* 352
 - File_Archive* 352
 - Serializer* 232
 - Serialzer* 371
 - Services_JSON* 351
 - Text_Password* 358
 - Unserializer* 232, 371
- PHP
 - \$_GET* 197
 - array_push()* 233
 - base64_decode()* 358
 - file_exists()* 319
 - file_get_contents()* 310, 351
 - flock()* 234
 - getAllResponseHeaders()* 186
 - getResponseHeader()* 186
 - header()* 189
 - PasswordStrength* 358
 - PclZip* 352
 - switch* 201
- Planetensystem 59
- POST 167, 199
- POST-Request 199
- Praxisbeispiele 301
 - ajaxBooks* 301
 - ajaxChat* 310
 - ajaxComplete* 323
 - ajaxDict* 332
 - ajaxDir* 341
 - ajaxPass* 353
 - ajaxTic* 359
- previousSibling 24
- Protopage 10
- Prototype 238, 313
 - \$()-Funktion* 252
 - \$F()-Funktion* 261
 - addClassName* 257
 - ajax* 239
 - Ajax.PeriodicalUpdater* 248
 - Ajax.Request* 239
 - Ajax.Updater* 244
 - escapeHTML* 263
 - Formulare* 259
 - getElementsByClassName()* 252
 - getHeight* 256
 - hasClassName* 256
 - hide* 255
 - Insertion* 257
 - onComplete* 240
 - onFailure* 240
 - onSuccess* 244
 - removeClassName* 257
 - show* 255

stripTags 263
toColorPart() 251
toggle 255
unescapeHTML 264
prototype 105
Prototype-Objekt 106
Punktnotation 28, 103, 151, 211
Punktoperator 66, 99
PUT 167

R

random 149
readyState 130, 186, 190, 203
Regeln für StyleSheets 92
RegExp 157
regulärer Ausdruck 208
Rekursion 37–38, 135
rekursiv 35, 70
rekursiver Operator 152
remove() 303
removeAllRanges() 335
removeAttribute() 51, 53, 77
removeAttributeNode() 51, 53
removeChild() 30, 33, 145
removeClassName 257
removeProperty() 82
removeRule() 92
replace() 24
replaceChild() 30, 34
Request-Header 178
Response-Header 177
responseText 187–188, 190, 207
responseXML 187–188, 242
Returnwert 104
rgb-Triple 86
RGB-Werte 86
RSS-Newsfeed 215

S

Schachbrett 111
Schlüsselwort
 null 101, 105
 this 104, 298
 with 101
Schweizer Taschenmesser 10
Script.aculo.us
 Ajax.Autocompleter 325, 328
 Ajax.PeriodicalUpdater() 365

Autocompleter.Base 325
Autocompleter.Local 325
Autocompleter-Klasse 324
Draggable 343
Effect.Appear 357
script.aculo.us 264
 Draggable 265
 Draggable 269
 Effect.Appear 286
 Effect.BlindDown 289
 Effect.BlindUp 289
 Effect.DropOut 288
 Effect.Fade 287
 Effect.Fold 290
 Effect.Grow 290
 Effect.Highlight 284
 Effect.Parallel 283
 Effect.Puff 287
 Effect.Pulsate 290
 Effect.Scale 285
 Effect.Shake 288
 Effect.Shrink 291
 Effect.SlideUp 290
 Effect.Squish 290
 Effect.SwitchOff 289
 Effekt.MoveBy 283
 Effekt.Opacity 282
 Grundeffekte 280
 Kombinationen 286
 Sortable 272
selectedIndex 144
selection 334
selectorText 86
Selektion 334
send() 183, 185
serializeToString 141
Serialzer 371
Services_JSON 351
Session 310
Session-Objekt 78, 81
setAttribute 135, 147
setAttribute() 51, 53, 77, 79
setAttributeNode() 51, 53
setProperty() 82
setRequestHeader() 183, 185
setTimeout() 205
ShoutBox 226
Signet Scripts 218

SpiderMonkey 149
Sprungmarke 220
status 187, 190
statusText 187
String
 trim() 334
stringify() 212
style 66
Style-Attribut 69
Style-Eigenschaften 67
style-Objekt 66, 69, 71
styleSheet-Objekt 42
styleSheets 41–42
StyleSheets-Eigenschaften 72
StyleSwitcher 75, 78, 81, 88
synchrone Datenübertragung 131

T

tagName 49
Taschenrechner 200
test() 157
Text 20
#text 23, 26, 56, 136
text/html 138
text/xml 189
TEXT_NODE 21, 23, 26, 56
Text_Password 358
Text-Objekte 20
this 66, 104, 298
title 88
toLowerCase() 86
TRACE 168
trim() 334
try/catch 181
type 88
typeof 101, 140, 151, 161

U

undefined 101
Ungarische Notation 99
Unicode-Zeichensatz 128
UniversalBrowserRead 214
Unserializer 371
UTF-8 128

V

Validator 29
verschachtelte Schleife 135

Visuelle Effekte 264, 279
VML-Namensraum 395

W

Warenkorb 302
Web 2.0 9
WebFX 43
with 101, 104
writely 10
Wurzelknoten 22, 39, 128

X

XML 127
 async 131–132
 Attribut 129
 createDocument() 132
 Datentyp 151
 document.implementation 131
 documentElement 130
 DOMParser 138
 DOMParser() 134
 E4X 149
 encoding 128
 Header 128
 Kodierung 128
 length() 151
 parseFromtString() 138
 readyState 130
 XMLSerializer() 141
XML laden 129
XML parsen 134
XML vs. Text 198
XML-Datei 128
XMLDOM 129
XML-Elemente 129
XMLHTTP 179–180
XMLHttpRequest 141, 179, 190
 abort() 183
 getAllResponseHeaders() 183
 getResponseHeader() 183
 Methoden 183
 open() 183
 send() 183
 setRequestHeader() 183
XML-Namespace 132
XML-Objekt 150, 189
XMLSerializer() 141
XML-Signatur 128, 151

XPath 152
XPath-Prädikate 154

Y

Yahoo! Maps 10, 375, 403
 addLabel() 411
 addMarker() 412
 addOverlay() 411
 addPanControl() 408
 addZoomLong() 408
 addZoomShort() 408
 Beispielanwendung 416
 Detailinformationen 413
 disableDragMap() 409
 Drag&Drop 409
 drawZoomAndCenter() 407
 enableDragMap() 409
 Entwickler-ID 405
 Markierungspunkte 410
 Nutzungsbestimmungen 405
 setZoomLevel() 408
 Steuerelemente 407
 YEvent-Klasse 413
 YGeoPoint() 405
 YImage-Klasse 410
 YMap() 405
 YMarker-Klasse 410

Z

Zeichensatz 151
Zeilenumbruch 24
Zeilenumbrüche 24, 36, 136
Zeitüberschreitung 186
Zimbra 12
Ziparchiv 341
Zufallsfaktor 149
Zufallszahl 149, 210
zusätzliche Attribute 27
Zuweisungsoperator 99
zweidimensionales Array 110