

Auf einen Blick

Teil I JavaScript lernen	27
1 Einleitung	29
2 Vorbereitung	41
3 Programmierung I	71
4 Programmierung II	87
5 Erste Schritte	103
6 Fenster I	123
7 Formulare I	141
8 Fenster II: Frames	163
9 Images	193
10 Fenster III	217
11 Cookies	237
12 Formulare II	255
13 Objekte und Arrays	279
14 Musik	297
15 Events	307
16 DHTML I: Netscape 4	331
17 DHTML II: Internet Explorer	353
18 DHTML III: Mozilla & Co.	365
19 DHTML IV: Für alle Browser	377
20 Java	395
21 Signierte Skripten	409
22 DOM	417
23 Fehler	427
24 Web Services	441
25 JavaScript goes .NET	465
Teil II JavaScript anwenden.....	481
26 JavaScript einbauen	483
27 Zufall	499
28 Cookies	513

29	Code schützen	535
30	Top Secret: Passwortschutz	555
31	Grafiken	569
32	Frames	589
33	DHTML V – für die Praxis	601
34	Fenster(In)	621
35	Fenster(In) für Fiese	657
36	Laufschrift	673
37	Navigation	701
38	Warenkorb	719
39	Eingaben überprüfen I	759
40	Eingaben überprüfen II	789
41	Multimedia steuern	825
42	Flash & Co.	847
43	Spaß serverseitig	869
	Anhang	887
A	Lösungen	889
B	Referenz	917
C	Quellen im Web	997
	Index	1001

Inhalt

Teil I JavaScript lernen 27

1 Einleitung 29

1.1	Die Geschichte von JavaScript	30
1.2	Voraussetzungen	32
1.3	Danksagungen zu »JavaScript« (1. Auflage)	34
1.4	Danksagungen zu »JavaScript-Rezepte« (1. Auflage)	35
1.5	Vorwort zur 5. Auflage	36
1.6	Vorwort zur 6. Auflage	37
1.7	Die Icons in diesem Buch	39

2 Vorbereitung 41

2.1	Webbrowser	41
2.1.1	Netscape Navigator (und Konsorten)	41
2.1.2	Microsoft Internet Explorer	46
2.1.3	Opera	49
2.1.4	Konqueror	50
2.1.5	Safari	51
2.1.6	Marktanteile	52
2.1.7	Testsystem	54
2.2	Verwendung von <script>	55
2.2.1	Das language-Attribut	57
2.2.2	Browser ohne JavaScript	61
2.2.3	Externe Dateien	65
2.3	JavaScript-Links	67
2.4	Event-Handler	69
2.5	JavaScript-Entities	69

3 Programmierung I 71

3.1	Variablen	71
3.1.1	Namensgebung	71
3.1.2	Numerische Variablen	72
3.1.3	Zeichenketten	72

3.1.4	Boolesche Variablen	73
3.1.5	Variablendeklaration	73
3.2	Operatoren	74
3.2.1	Arithmetische Operatoren	74
3.2.2	Boolesche Operatoren	76
3.2.3	String-Operatoren	78
3.2.4	Umwandlung zwischen den Variablentypen	79
3.3	Kontrollstrukturen: Schleifen	80
3.3.1	For-Schleifen	80
3.3.2	Do-While-Schleife	82
3.3.3	While-Schleife	83
3.3.4	For-In-Schleife	83
3.3.5	Schleifensteuerung	84
3.4	Fragen & Aufgaben	84

4 Programmierung II 87

4.1	Fallunterscheidung	87
4.1.1	If-Anweisung	87
4.1.2	Switch-Anweisung	90
4.2	Datenspeicherung	92
4.2.1	Die eval-Methode	92
4.2.2	Arrays	94
4.3	Funktionen	96
4.4	Objekte	100
4.5	Fragen & Aufgaben	101

5 Erste Schritte 103

5.1	JavaScript-Objekte	103
5.1.1	Das Objekt Date	103
5.1.2	Das Objekt Math	109
5.2	Browser-Erkennung	112
5.3	Event-Handler	120
5.4	Fragen & Aufgaben	121

6 Fenster I 123

6.1	Modale Fenster	123
6.1.1	Warnung – nur im Notfall	124
6.1.2	Bestätigungen	126
6.1.3	Benutzereingaben	128

6.2	Navigationsleiste mit JavaScript	129
6.2.1	Das History-Objekt	129
6.2.2	Vorwärts und rückwärts, Teil 2	130
6.2.3	Drucken mit JavaScript	131
6.3	Die Statuszeile	132
6.3.1	Erläuternde Links	133
6.3.2	Laufschrift	133
6.4	Das location-Objekt	138
6.5	Fragen & Aufgaben	139

7 Formulare I 141

7.1	Überprüfung auf Vollständigkeit	141
7.1.1	Allgemeiner Aufbau	143
7.1.2	Texteingabefelder	144
7.1.3	Radiobuttons	145
7.1.4	Checkboxes	146
7.1.5	Auswahllisten	146
7.1.6	Fehlermeldung ausgeben	147
7.1.7	Konstruktive Vorschläge	148
7.2	Automatische Überprüfung	150
7.2.1	Texteingabefelder	151
7.2.2	Radiobuttons	152
7.2.3	Checkboxes	153
7.2.4	Auswahllisten	153
7.2.5	Zusammenfassung	153
7.3	Anwendungsmöglichkeiten für Formulare	155
7.3.1	Währungsrechner	155
7.3.2	Währungsrechner, Teil 2	157
7.3.3	Formularfelder für die Textausgabe nutzen	158
7.3.4	Navigation mit Auswahllisten	159
7.4	Fragen & Aufgaben	161

8 Fenster II: Frames 163

8.1	Mit Frames arbeiten	163
8.1.1	Frames mit HTML	164
8.1.2	Frames mit JavaScript füllen	165
8.2	Auf Daten von Frames zugreifen	167
8.2.1	Auf übergeordnete Frames zugreifen	169
8.2.2	Auf Daten von Unterframes zugreifen	172
8.2.3	Mehrere Frames gleichzeitig ändern	174
8.2.4	Gefährliche Fallen	175
8.3	Ein Warenkorb in JavaScript	176
8.3.1	Daten in den Warenkorb eintragen	177

8.3.2	Daten aus dem Warenkorb auslesen	180
8.3.3	Den Warenkorb verändern	183
8.4	Diashow	186
8.4.1	Vorbereitungen	187
8.4.2	Diashow starten	189
8.4.3	Diashow anhalten	189
8.4.4	Vorwärts und rückwärts springen	190
8.4.5	Diashow verlassen	190
8.5	Fragen & Aufgaben	190

9 Images 193

9.1	Bildlein-Wechsle-Dich	193
9.1.1	Zugriff auf Grafiken	194
9.1.2	Prüfungen auf Kompatibilität	195
9.2	Animierte JPEGs	198
9.2.1	Eine Animation mit JavaScript	199
9.2.2	Bilder in den Cache laden	200
9.3	Animierte Navigation	203
9.3.1	Vorüberlegungen	204
9.3.2	Auf- und Zuklappen	205
9.3.3	Die einzelnen Menüpunkte	206
9.3.4	Verlinkung der Menüpunkte	208
9.3.5	Einbau in die HTML-Datei	208
9.4	Erweiterung der Navigation	209
9.4.1	Vorbereitungen	209
9.4.2	Leichte Änderungen	210
9.4.3	Doppeltes Mouseover	211
9.4.4	Das komplette Beispiel im Überblick	212
9.5	Tipps aus der Praxis	212
9.5.1	Vorladen – aber richtig	212
9.5.2	Ladestand einer Grafik	212
9.6	Fragen & Aufgaben	215

10 Fenster III 217

10.1	Ein neues Fenster öffnen	217
10.1.1	Ein Fenster öffnen und füllen	217
10.1.2	Ein Fenster öffnen und verlinken	219
10.1.3	Ein Fenster öffnen und anpassen	220
10.1.4	Modale Fenster	225
10.2	Fernsteuerung	226
10.2.1	Links mit JavaScript	226
10.2.2	Links ohne JavaScript	228

10.3	Fenster schließen	229
10.3.1	Andere Fenster schließen	230
10.3.2	Lösung für ältere Browser	231
10.4	Fenster in den Vordergrund holen	232
10.5	Fenster bewegen mit JavaScript	233
10.5.1	Fenster verschieben	234
10.5.2	Fensterinhalt scrollen	235
10.6	Fragen & Aufgaben	236

11 Cookies 237

11.1	Was ist ein Cookie?	237
11.2	Wie sieht ein Cookie aus?	238
11.3	Cookies mit JavaScript	240
11.3.1	Cookies setzen	241
11.3.2	Cookies löschen	241
11.3.3	Cookies lesen	242
11.3.4	Cookie-Unterstützung überprüfen	242
11.3.5	Warenkorb mit Cookies	245
11.4	Informationen behalten ohne Cookies	248
11.5	Fragen & Aufgaben	253

12 Formulare II 255

12.1	Daten behalten	255
12.1.1	Das Eingabeformular	255
12.1.2	Die Ausgabeseite	257
12.2	Dynamische Auswahllisten	260
12.2.1	Ein erster Ansatz	260
12.2.2	Ein fortgeschrittener Ansatz	263
12.3	Überprüfungsfunktionen	264
12.3.1	Ganze Zahlenwerte	264
12.3.2	Dezimalzahlen	266
12.3.3	Telefonnummern	267
12.3.4	In Zahlenwerte umwandeln	268
12.4	Reguläre Ausdrücke	269
12.4.1	Kurzeinführung	269
12.4.2	Ein Objekt erzeugen	271
12.4.3	Mit dem Objekt arbeiten	272
12.5	Fragen & Aufgaben	278

13 Objekte und Arrays **279**

13.1	Array-Erweiterungen	279
13.1.1	Einfügen, nicht anfügen	280
13.1.2	Anfügen und löschen	280
13.1.3	Karten mischen	282
13.1.4	Sortieren	282
13.2	Eigene Objekte	286
13.2.1	Allgemeines	286
13.2.2	Methoden definieren	287
13.2.3	Eigene Sortiermethode	288
13.2.4	Eigene Sortiermethode, Teil 2	290
13.2.5	Zusammenfassung	291
13.3	Fragen & Aufgaben	295

14 Musik **297**

14.1	Plugins erkennen	297
14.1.1	Zugriff auf Plugins	298
14.1.2	Zugriff auf MIME-Typen	299
14.1.3	Refresh	299
14.2	Zugriff auf Musikdateien	300
14.2.1	Browsertest	300
14.2.2	Soundsteuerung	301
14.2.3	Jukebox	302
14.3	Fragen & Aufgaben	305

15 Events **307**

15.1	Events mit dem Netscape Navigator	307
15.1.1	Neue Ereignisse	308
15.1.2	Ereignisse als Objekteigenschaften	309
15.1.3	Ereignisse abfangen	310
15.1.4	Ereignisbehandlung	311
15.1.5	Ereignisse umleiten	314
15.1.6	Ereignisse durchleiten	315
15.1.7	Tastatureingaben	316
15.2	Events mit dem Internet Explorer	318
15.2.1	Neue Ereignisse	318
15.2.2	Ereignisse als Objekteigenschaften	318
15.2.3	Spezielle Skripten	319
15.2.4	Ereignisse abfangen	319
15.2.5	Bubbling	320
15.2.6	Das Event-Objekt	322

15.3	Events mit beiden Browsern	324
15.3.1	Browserunabhängigkeit	324
15.3.2	Benutzereingaben	325
15.4	Fragen & Aufgaben	329

16 DHTML I: Netscape 4 331

16.1	Grundlagen	331
16.1.1	Begriffsbestimmung	331
16.1.2	Cascading Style Sheets	332
16.1.3	Positionierung von Elementen	333
16.1.4	JavaScript Style Sheets	333
16.1.5	Layer	334
16.2	Beispiele	336
16.2.1	Animiertes Logo	336
16.2.2	Drag&Drop	339
16.2.3	Sichtbar und unsichtbar	343
16.2.4	Neuer Mauszeiger	346
16.2.5	Permanentes Werbebanner	348
16.3	Fragen & Aufgaben	351

17 DHTML II: Internet Explorer 353

17.1	Grundlagen	353
17.1.1	HTML-Tags	353
17.1.2	Objektzugriff	354
17.2	Beispiele	354
17.2.1	Animiertes Logo	355
17.2.2	Drag&Drop	356
17.2.3	Sichtbar und unsichtbar	359
17.2.4	Neuer Mauszeiger	361
17.2.5	Permanentes Werbebanner	362
17.3	Fragen & Aufgaben	364

18 DHTML III: Mozilla & Co. 365

18.1	Grundlagen	366
18.1.1	HTML-Tags	366
18.1.2	Objektzugriff	366
18.2	Beispiele	367
18.2.1	Animiertes Logo	367
18.2.2	Drag&Drop	369
18.2.3	Sichtbar und unsichtbar	371

18.2.4	Neuer Mauszeiger	373
18.2.5	Permanentes Werbebanner	374
18.3	Fragen & Aufgaben	376

19 DHTML IV: Für alle Browser 377

19.1	Animiertes Logo	377
19.2	Drag&Drop	381
19.3	Sichtbar und unsichtbar	386
19.4	Neuer Mauszeiger	390
19.5	Permanentes Werbebanner	391
19.6	Fragen & Aufgaben	393

20 Java 395

20.1	Allgemeines	395
20.1.1	Wie funktioniert Java?	395
20.1.2	Kurzeinführung in Java	397
20.2	Java und das WWW	398
20.2.1	Ein Beispiel-Applet	398
20.2.2	HTML-Integration	399
20.3	Java ohne Applet	402
20.3.1	Exemplarische Java-Objekte	402
20.3.2	Blackjack	403
20.3.3	Karten initialisieren	404
20.3.4	Karten mischen	404
20.4	Fragen & Aufgaben	407

21 Signierte Skripten 409

21.1	Zusätzliche Rechte	409
21.1.1	Allgemeines	410
21.1.2	Surfüberwachung	411
21.1.3	Besondere Fenster	413
21.2	Signieren	415
21.2.1	Zigbert	415
21.2.2	HTML-Code anpassen	416

22 DOM 417

22.1	Der DOM-Baum	417
22.2	Navigation im Baum	418

22.3	Den Baum modifizieren	419
22.4	Fragen & Aufgaben	425

23 Fehler 427

23.1	Fehler abfangen	427
23.1.1	Keine Fehlermeldung	428
23.1.2	Besondere Fehlermeldung	428
23.1.3	Ausblick: Fehlermeldungen verschicken	429
23.2	JavaScript Debugger	434
23.2.1	Wo ist der Fehler?	435
23.2.2	Breakpoints	437
23.2.3	Watches	437
23.2.4	Einzelne Werte anzeigen	438
23.2.5	Schrittweise Programmausführung	439
23.3	Fragen	440

24 Web Services 441

24.1	Was sind Web Services?	442
24.1.1	Verteiltes Arbeiten	442
24.1.2	WSDL	443
24.1.3	Web Service aufrufen	444
24.2	Web Services mit JScript .NET	446
24.2.1	Installation	447
24.2.2	Programmierung	448
24.3	Mit dem Internet Explorer auf Web Services zugreifen	452
24.4	Mit Mozilla auf Web Services zugreifen	456
24.5	Web Services ohne Web Service	462
24.6	Fazit	464

25 JavaScript goes .NET 465

25.1	Erste Schritte	465
25.2	HTML Controls	470
25.3	Web Controls	472
25.4	Validation Controls	475
25.5	Fazit	479

26 JavaScript einbauen 483

26.1	JavaScript: ja oder nein?	483
26.2	JavaScript-Versionen	487
26.3	Browserversionen	491
26.4	Methoden und Objekte prüfen	495

27 Zufall 499

27.1	Zufallszahlen erstellen	499
27.1.1	JavaScript-Zufallszahlen	499
27.1.2	HP-Verfahren	500
27.1.3	Datumswert	502
27.2	Hilfsfunktionen	503
27.2.1	Zufallszahl aus einem Bereich	503
27.2.2	Mehrere Zufallszahlen	504
27.3	Anwendungsbeispiele	506
27.3.1	Lottozahlen	506
27.3.2	Zufallsbanner	508

28 Cookies 513

28.1	Allgemeines	514
28.1.1	Cookie-Elemente	515
28.1.2	Cookies mit HTML setzen	519
28.2	Cookies schreiben	520
28.3	Cookies lesen	522
28.4	Cookies löschen	524
28.5	Anwendungen	525
28.5.1	Cookie-Unterstützung prüfen	525
28.5.2	Alle Cookies auslesen	527
28.5.3	Ein Cookie statt vieler Cookies	530

29 Code schützen 535

29.1	Quellcode einsehen	535
29.1.1	Menübefehle	536
29.1.2	Tastenkürzel	536
29.1.3	Kontextmenü	537
29.1.4	Dateisystem	538

29.2	Code im Frame verstecken	539
29.3	Mausklick verhindern	541
29.4	Code codieren	544
29.4.1	Optisch verschleiern	544
29.4.2	Inhaltlich verschleiern	545
29.5	Dateien auslagern	549
29.6	Caching verhindern	550
29.7	Code serverseitig generieren	551

30 Top Secret: Passwortschutz 555

30.1	URL aus Passwort	556
30.1.1	Passwort \approx URL	556
30.1.2	Passwort \rightarrow URL	557
30.1.3	f : Passwort R URL	558
30.2	Seiten mit Cookies schützen	559
30.2.1	Passwort im Quelltext	560
30.2.2	Mit Java	561
30.3	Ein Blick über den Tellerrand	563
30.3.1	PHP	563
30.3.2	ASP	565
30.3.3	.htaccess	566

31 Grafiken 569

31.1	Ein Image-Objekt einbinden	570
31.2	Vorladen	575
31.3	Fortschrittsanzeige	577
31.4	Rollover	583

32 Frames 589

32.1	Frames füllen	589
32.2	Framezugriff	591
32.3	(Mehrere) Frames ändern	592
32.4	Frames forever	595
32.5	Alternativen	597

33 DHTML V – für die Praxis 601

33.1	Browserabhängiges DHTML	603
33.1.1	Der Ansatz von Netscape	603
33.1.2	Der Ansatz von Microsoft	605
33.1.3	Der Ansatz des W3C	606
33.2	Browserunabhängiges DHTML	607
33.3	Hilfsfunktionen	610
33.3.1	Zugriff	611
33.3.2	Verstecken	612
33.3.3	Position	613
33.3.4	Text ändern	615
33.4	Beispiele	616
33.5	Weitere Hinweise	620

34 Fenster(In) 621

34.1	Fenster öffnen	622
34.2	Fensteroptionen	625
34.3	Anwendung: Hilfsskripten	637
34.3.1	Fenster positionieren	638
34.3.2	Fenster in der Ecke	639
34.3.3	Fenster zentrieren	643
34.4	Auf das öffnende Fenster zugreifen	645
34.5	Anwendung: Sitemap	647
34.6	Anwendung: Adressbuch	650

35 Fenster(In) für Fiese 657

35.1	Fenster im Hintergrund	658
35.1.1	Immer im Hintergrund	658
35.1.2	Einmal im Hintergrund	659
35.2	Fenster im Vordergrund	660
35.2.1	Immer im Vordergrund	660
35.2.2	Einmal im Vordergrund	660
35.2.3	Manchmal im Vordergrund	661
35.3	Fenster forever	664
35.3.1	Fenster schließen? Schwierig!	665
35.3.2	Fenster schließen? Sinnlos!	666
35.4	Contra WebWasher & Co.	669

36 Laufschrift 673

36.1	Exkurs: Laufschrift mit HTML	674
36.2	Grundsätzlicher Aufbau	676
36.3	Laufschrift in der Statuszeile	679
36.3.1	Einfache Variante	679
36.3.2	Besser: Sonderbehandlung für Links	681
36.4	Laufschrift im Textfeld	683
36.4.1	Einfache Variante	683
36.4.2	Besser: Links in der Laufschrift	687
36.5	Laufschrift mit DHTML	692
36.5.1	Einfache Variante	692
36.5.2	Besser: Links direkt integriert	695

37 Navigation 701

37.1	Navigation mit Pulldown-Menüs	701
37.2	Navigation mit DHTML	709
37.2.1	Baumstrukturen	710
37.3	Alternativen im Web	715
37.3.1	Joust	716
37.3.2	DHTML Menu Builder	718

38 Warenkorb 719

38.1	Datenstruktur	720
38.2	Mit unsichtbaren Frames arbeiten	723
38.2.1	Warenkorb füllen	725
38.2.2	Artikel anzeigen	726
38.2.3	Warenkorb ändern	735
38.3	Mit Cookies arbeiten	739
38.3.1	Warenkorb füllen	739
38.3.2	Artikel anzeigen	740
38.3.3	Warenkorb ändern	745
38.4	Über die URL	747
38.4.1	Den Warenkorb füllen	749
38.4.2	Artikel anzeigen	750
38.4.3	Den Warenkorb ändern	755
38.5	Fazit	757

39 Eingaben überprüfen I 759

39.1	Theorie: Formularelemente	759
39.1.1	Allgemeines	760
39.1.2	Textfelder	760
39.1.3	Checkboxes	760
39.1.4	Radiobuttons	761
39.1.5	Auswahllisten	762
39.2	Vollständigkeit	762
39.2.1	Vorbereitung	763
39.2.2	Textfelder	764
39.2.3	Checkboxes	764
39.2.4	Radiobuttons	765
39.2.5	Auswahllisten	766
39.2.6	Globale Überprüfung	768
39.3	Musterprüfung	770
39.3.1	Numerische Werte	771
39.3.2	Postleitzahlen	774
39.3.3	Telefonnummern	775
39.3.4	Geburtsdatum	775
39.3.5	E-Mail-Adressen	778
39.4	Reguläre Ausdrücke	781
39.4.1	Theorie: RegExp & Co.	782
39.4.2	Numerische Werte	783
39.4.3	Postleitzahlen	784
39.4.4	Telefonnummern	784
39.4.5	Geburtsdatum	784
39.4.6	E-Mail-Adressen	786

40 Eingaben überprüfen II 789

40.1	Theorie: Den Formularversand abfangen	791
40.2	Überprüfung mit Fehlermeldung	792
40.3	Überprüfung mit grafischer Fehlermeldung	795
40.4	Überprüfung mit Korrekturmöglichkeit	808
40.5	Vollautomatische Überprüfung	813

41 Multimedia steuern 825

41.1	Musik	826
41.1.1	Einbau in HTML	826
41.1.2	Standardkontrollen des Internet Explorer	827
41.1.3	Standardkontrollen des Netscape Navigator	829
41.1.4	Browserunabhängige Ansteuerung	830
41.1.5	Anwendung: Wurlitzer	836

41.2	Microsoft Windows Media Player	837
41.2.1	Einbau in HTML	838
41.2.2	Browserunabhängige Ansteuerung	840
41.2.3	Anwendung: Heimkino	844

42 Flash & Co. 847

42.1	Prinzipielles	848
42.2	Director	850
42.2.1	Standardeinbau	850
42.2.2	Erkennung mit dem Internet Explorer	851
42.2.3	Erkennung mit dem Netscape Navigator	852
42.2.4	Browserunabhängige Erkennung	854
42.3	Flash	856
42.3.1	Standardeinbau	856
42.3.2	Erkennung mit dem Internet Explorer	857
42.3.3	Erkennung mit dem Netscape Navigator	858
42.3.4	Browserunabhängige Erkennung	859
42.4	Mit Flash kommunizieren	861
42.4.1	Flash ruft JavaScript	861
42.4.2	JavaScript ruft Flash	864
42.4.3	Beispiele	865

43 Spaß serverseitig 869

43.1	Variablentausch	870
43.2	Anwendungen	873
43.2.1	Newsticker	874
43.2.2	Bankleitzahlen	880

Anhang 887

A Lösungen 889

B Referenz 917

B.1	Das Anchor-Objekt	919
B.1.1	Allgemeines	919
B.1.2	Eigenschaften	920

B.2	Das Array-Objekt	920
B.2.1	Allgemeines	921
B.2.2	Methoden	921
B.2.3	Eigenschaften	924
B.3	Das Button-Objekt	924
B.3.1	Allgemeines	924
B.3.2	Event-Handler	924
B.3.3	Methoden	924
B.3.4	Eigenschaften	925
B.4	Das Checkbox-Objekt	925
B.4.1	Allgemeines	925
B.4.2	Event-Handler	925
B.4.3	Methoden	925
B.4.4	Eigenschaften	926
B.5	Das Date-Objekt	926
B.5.1	Allgemeines	926
B.5.2	Methoden	927
B.6	Das document-Objekt	933
B.6.1	Allgemeines	933
B.6.2	Event-Handler	933
B.6.3	Methoden	933
B.6.4	Eigenschaften	935
B.7	Das Event-Objekt	940
B.7.1	Netscape-Eigenschaften	940
B.7.2	Internet Explorer-Eigenschaften	941
B.8	Das FileUpload-Objekt	943
B.8.1	Allgemeines	943
B.8.2	Event-Handler	943
B.8.3	Methoden	943
B.8.4	Eigenschaften	944
B.9	Das Form-Objekt	944
B.9.1	Allgemeines	944
B.9.2	Event-Handler	944
B.9.3	Methoden	944
B.9.4	Eigenschaften	945
B.10	Das Frame-Objekt	946
B.11	Das Hidden-Objekt	946
B.11.1	Allgemeines	946
B.11.2	Eigenschaften	946
B.12	Das History-Objekt	947
B.12.1	Allgemeines	947
B.12.2	Methoden	947
B.12.3	Eigenschaften	947

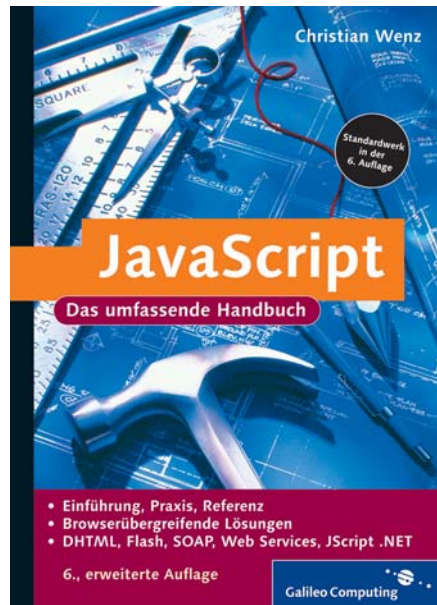
B.13	Das Image-Objekt	948
B.13.1	Allgemeines	948
B.13.2	Event-Handler	948
B.13.3	Eigenschaften	948
B.14	Das Layer-Objekt	950
B.14.1	Allgemeines	950
B.14.2	Event-Handler	950
B.14.3	Methoden	950
B.14.4	Eigenschaften	952
B.15	Das Link-Objekt	954
B.15.1	Allgemeines	954
B.15.2	Event-Handler	954
B.15.3	Eigenschaften	954
B.16	Das Location-Objekt	955
B.16.1	Methoden	955
B.16.2	Eigenschaften	956
B.17	Das Math-Objekt	957
B.17.1	Methoden	957
B.17.2	Eigenschaften	960
B.18	Das MimeType-Objekt	961
B.18.1	Eigenschaften	961
B.19	Das Navigator-Objekt	961
B.19.1	Methoden	961
B.19.2	Eigenschaften	962
B.20	Das Number-Objekt	963
B.20.1	Allgemeines	963
B.20.2	Eigenschaften	963
B.21	Das Object-Objekt	964
B.21.1	Allgemeines	964
B.21.2	Methoden	964
B.21.3	Eigenschaften	965
B.22	Das Option-Objekt	965
B.22.1	Allgemeines	965
B.22.2	Eigenschaften	965
B.23	Das Password-Objekt	966
B.23.1	Allgemeines	966
B.23.2	Event-Handler	966
B.23.3	Methoden	966
B.23.4	Eigenschaften	966
B.24	Das Plugin-Objekt	967
B.24.1	Eigenschaften	967
B.25	Das Radio-Objekt	967
B.25.1	Allgemeines	967
B.25.2	Event-Handler	968
B.25.3	Methoden	968
B.25.4	Eigenschaften	968

B.26	Das RegExp-Objekt	969
B.26.1	Allgemeines	969
B.26.2	Eigenschaften	969
B.26.3	Methoden	970
B.27	Das Reset-Objekt	971
B.27.1	Allgemeines	971
B.27.2	Event-Handler	971
B.27.3	Methoden	971
B.27.4	Eigenschaften	971
B.28	Das Screen-Objekt	972
B.28.1	Eigenschaften	972
B.29	Das Select-Objekt	972
B.29.1	Allgemeines	973
B.29.2	Event-Handler	973
B.29.3	Methoden	973
B.29.4	Eigenschaften	973
B.30	Das String-Objekt	974
B.30.1	Allgemeines	974
B.30.2	Methoden	974
B.30.3	Eigenschaften	979
B.31	Das Submit-Objekt	979
B.31.1	Allgemeines	980
B.31.2	Event-Handler	980
B.31.3	Methoden	980
B.31.4	Eigenschaften	980
B.32	Das Text-Objekt	981
B.32.1	Allgemeines	981
B.32.2	Event-Handler	981
B.32.3	Methoden	981
B.32.4	Eigenschaften	981
B.33	Das Textarea-Objekt	982
B.33.1	Allgemeines	982
B.33.2	Event-Handler	982
B.33.3	Methoden	982
B.33.4	Eigenschaften	983
B.34	Das Window-Objekt	983
B.34.1	Allgemeines	983
B.34.2	Event-Handler	983
B.34.3	Methoden	984
B.34.4	Eigenschaften	992
B.35	Top-Level-Eigenschaften und -Methoden	995
B.35.1	Methoden	995
B.35.2	Eigenschaften	996

C	Quellen im Web	997
C.1	Websites	997
C.2	Newsgroups	998
C.3	Mailinglisten	999
C.4	MyGalileo	1000
	Index	1001

Teil I

JavaScript lernen



1 Einleitung

So können Angreifer beispielsweise Nutzerkennung mit Passwörtern oder auch lokal gespeicherte Daten von privaten und kommerziellen Internetnutzern ausspionieren. [...] Das BSI empfiehlt deswegen den Betreibern von WWW-Servern dringend, vollständig auf JavaScript zu verzichten oder zumindest für ihre sicherheitsbewussten Kunden Alternativangebote bereitzustellen, die ohne aktive Inhalte dargestellt werden können.

Pressemitteilung des Bundesamtes für Sicherheit in der Informationstechnik, BSI (<http://www.bsi.bund.de/fachthem/sinet/java99.htm>)

Vorsicht ist geboten, wenn Unternehmen für ihre Web-Seiten das Gestaltungssystem JavaScript verwenden. Dahinter verbergen sich kleine Programme, die sich unbemerkt auf dem Kunden-PC einnisten und ihn angreifbar für fremde Zugriffe machen.
BILD am SONNTAG, 21. Mai 2000

JavaScript ist in aller Munde – sowohl im positiven als auch im negativen Sinne. Immer wieder geistern Horrormeldungen wie obige Empfehlung des BSI oder der Unfug aus der BamS durch die Presse. Der Grund dafür ist, dass den Browser-Herstellern – in diesem Falle vor allem Netscape (Netscape Navigator) und Microsoft (Microsoft Internet Explorer) immer wieder Fehler bei der Programmierung ihrer Browser unterlaufen (bei Interesse: <http://www.guninski.com/>). Zwar werden eiligst Bugfixes zur Verfügung gestellt, aber der (Image-) Schaden ist schon eingetreten. Andererseits sind keine Fälle bekannt, in denen Sicherheitslecks zu Schäden geführt hätten. So hat die obige Pressemitteilung des BSI in der Fachpresse zu scharfer Kritik geführt; teilweise war von überzogener Panikmache die Rede. Die inhaltliche Qualität der Aussage aus der BamS bedarf keines Kommentars.

Ohne die Diskussion allzu sehr vertiefen zu wollen: Mit JavaScript kann man die eher beschränkten Möglichkeiten von HTML erweitern. Es handelt sich hierbei um eine clientseitige Programmiersprache. Das heißt, alles läuft im Browser ab, und man muss keine besonderen Server-Voraussetzungen erfüllen. Letzterer Punkt ist (noch) ein großer Nachteil von serverseitigen Sprachen wie ASP, Perl, PHP und Konsorten, denn viele Hoster (und vor allem die günstigen Hoster) gestatten keine serverseitige Programmierung. Mit JavaScript ist dies jedoch alles kein Problem.

In den nun folgenden Kapiteln finden Sie eine komplette Einführung in die Sprache, stets anhand von praxisnahen Beispielen illustriert. Ich habe versucht,

mich bei den Beispielen auf das Wesentliche – nämlich auf die JavaScript-Programmierung – zu konzentrieren. Daher sind die Beispiele grafisch bei weitem nicht ausgefeilt, aber sie erfüllen die gestellten Aufgaben.

Jedes Kapitel wird mit Fragen und Übungen abgeschlossen, so dass Sie das Erlernte gleich in die Tat umsetzen können. Kapitel 45 schließlich ist eine komplette Sprachreferenz, die Ihnen auch nach der Lektüre des Buches immer wieder als Nachschlagewerk dienen wird.

MyGalileo Des Weiteren möchte ich noch auf den Online-Dienst **MyGalileo** verweisen, der im World Wide Web unter <http://www.galileo-press.de/> zu erreichen ist. Wenn Sie dieses Buch registrieren, erhalten Sie Zugriff auf die Webseiten, die dieses Buch begleiten. Nach und nach werden dort Inhalte eingefügt, und Sie können mit mir in Kontakt treten. Nutzen Sie diese Gelegenheit; jede Art von konstruktiver Kritik ist willkommen. Wenn Sie einen Fehler finden, teilen Sie mir diesen bitte mit, damit er in einer weiteren Auflage dieses Buches nicht mehr auftritt. Sie helfen damit auch anderen Lesern. In **MyGalileo** werden Sie auch eine aktualisierte Fehlerliste für dieses Buch finden, sobald Fehler entdeckt werden sollten. Auch meine Homepage enthält einen Support-Bereich: Unter <http://www.hauser-wenz.de/support/> finden Sie Errata, sofern verfügbar, sowie Kontaktmöglichkeiten.

1.1 Die Geschichte von JavaScript

Gegen Ende des Jahres 1995 stellte die Firma Netscape die neueste Version ihres Internet-Browsers vor. Der Versionssprung von 1.1 auf 2.0 war berechtigt, da solch revolutionäre Dinge wie etwa die Unterstützung von Frames als neues Feature hinzugekommen waren. Unter anderem war auch eine Sprache namens **LiveScript** eingebaut, mit der man auf HTML-Seiten Einfluss nehmen konnte. Die Syntax dieser Sprache lehnte sich an Java an, und aus marketingtechnischen Gründen wurde die Programmiersprache schließlich in **JavaScript** umbenannt.

Schon bald begann der Siegeszug dieser Programmiersprache. Zwar war die Implementierung im Browser eher mangelhaft (In der Netscape-Version 2.0 konnte man JavaScript nicht einmal deaktivieren, was zu einem bösen Sicherheitsproblem werden sollte.), aber mit der Beta-Version des Netscape Navigator 3 wurde die JavaScript-Version 1.1 vorgestellt, die deutlich mehr Möglichkeiten bot. Microsoft wollte nun auch auf den Zug aufspringen und kündigte an, in Version 3 des Internet Explorers ebenfalls eine Skriptunterstützung anzubieten. Aus lizenzrechtlichen Gründen wurde die Sprache **JScript** getauft. Von der Syntax her war sie aber mit JavaScript praktisch identisch.

ECMAScript Seitdem läuft das übliche Wettrennen zwischen Netscape und Microsoft. Während der Internet Explorer 3 praktisch nur JavaScript 1.0 unterstützte, beherrschte der Internet Explorer 4 schon JavaScript 1.1 – und enthielt einige Features aus dem Sprachschatz von JavaScript 1.2, das mit dem Netscape Navi-

gator 4 eingeführt wurde, der parallel zum »IE4« erschien. Seit dem Netscape Navigator 4.06 gibt es JavaScript in der Version 1.3, die zwar nur rudimentäre Verbesserungen anbietet, sich aber an dem ECMA-262-Standard, ECMAScript, orientiert. Netscape hatte erkannt, dass man auf Standards setzen muss, und behauptete gleichzeitig, dass der Netscape Navigator 4.06 der Browser sei, der der Spezifikation von ECMAScript am nächsten komme. Microsoft wiederum verkleinerte mit dem Internet Explorer 5 den Rückstand weiter, setzte aber inzwischen eher auf andere Standards, wie **DOM (Document Object Model)**. Unter dem Codenamen **Mozilla** wurde währenddessen die nächste Netscape-Version als Open Source entwickelt. Das heißt, der Sourcecode lag offen, und jeder konnte an der neuen Version mitentwickeln. Nichtsdestotrotz wurde der Löwenanteil der Arbeit von Netscape-Angestellten geleistet, das »Linux-Wunder« funktioniert eben nicht überall!

Im Spätherbst 2000 begannen sich die Ereignisse zu überschlagen. Netscape geriet unter Druck, da der Internet Explorer 5.5 sowie der Internet Explorer 6, der als Teil des neuen Microsoft-Betriebssystems Windows XP (Codename: Whistler) angeboten wurde, schnell den aktuellen Zwischenstand des Open-Source-Projekts erreichten. Netscape bastelte eine Installationsroutine und bot eben diesen Zwischenstand als Netscape 6 zum Download an (bezeichnenderweise war zu dieser Zeit eine fast identische Version auf der Mozilla-Website mit der Versionsnummer 0.6 als Download zu haben). Dieser Schritt stieß in der Fachwelt auf große Kritik, denn die Version war noch weit davon entfernt, für den Produktiveinsatz zu taugen. Erst die späteren Unterversionen von Netscape 6 und die nächste Version, Netscape 7, sorgten hier für spürbare Besserung. Und – was aus Sicht der Webdesigner viel schlimmer war – die neue Version erwies sich in Sachen JavaScript als nicht abwärtskompatibel, wenn es um »DHTML« ging. So mussten (und müssen immer noch) eine Reihe von Skripten umprogrammiert werden. Doch keine Sorge, denn um eines vorwegzunehmen: In diesem Buch gehen wir ausführlich auf den neuesten Netscape-Browser ein. Die neuen Versionen (Netscape 6 und 7) unterstützen übrigens JavaScript 1.5, das aber nur unwesentliche Änderungen gegenüber den Vorgängerversionen aufweist. (Es wird eine aktuellere Version von ECMAScript unterstützt.) Dank dem Abkömmling Firefox nehmen Mozilla-Browser mittlerweile dem Internet Explorer kräftig Marktanteile ab.

Die Crux bei der Programmierung von Webseiten besteht darin, dass die beiden großen Browserhersteller immer wieder versucht haben, sich dadurch zu übertrumpfen, dass sie immer neue Features in ihre Browser einbauten. Das ist innerhalb des eigenen Mikrokosmos eine gute Sache, aber wenn die Technik inkompatibel zur Konkurrenz ist, schaut etwa die Hälfte der Websurfer (die nämlich den jeweils anderen Browser benutzen) bildlich gesprochen in die Röhre. Keine Firma kann es sich leisten, ein Web-Angebot nur für die halbe Zielgruppe zu erstellen.

Cross Browser In diesem Buch geht es nicht nur um die neuesten Effekte und Kniffe. Vielmehr geht es darum, wie Sie Ihre Zielgruppe erweitern können, indem JavaScript-Programme so gestaltet werden, dass sie von den Benutzern beider Browser und auch von den Benutzern älterer Versionen verwendet werden können. Bei bekannten Fehlern in der Implementierung einzelner Browser werden – soweit möglich – Workarounds angeboten. Ihr Ziel sollte es nicht nur sein, eine technisch eindrucksvolle Seite zu erstellen, sondern auch Ihre Zielgruppe zu erweitern, indem Sie ältere Browser nicht ausschließen oder zumindest vor Fehlermeldungen bewahren.

Sonstige Browser Es ist ja nicht so, dass nur der Netscape Navigator und der Internet Explorer JavaScript unterstützen. Die Nummer 3 im Browsermarkt, der Opera-Browser, unterstützt JavaScript-Version 1.3, und das sogar recht gut. Auch der in Star-Office integrierte Webbrowser sowie die Version 3 von Suns HotJava-Browser sind JavaScript-fähig. Diese Browser haben jedoch einen so kleinen Marktanteil, dass sie im Referenzteil nicht extra berücksichtigt werden. Mit umsichtiger Programmierung lassen sich aber auch hier Fehler vermeiden.

1.2 Voraussetzungen

**HTML-Kenntnisse
vorausgesetzt**

Wenn Sie dieses Buch durcharbeiten wollen, sollten Sie zumindest Kenntnisse in HTML vorweisen können – denn dieses Buch soll nicht künstlich durch einen zusätzlichen HTML-Teil aufgebläht werden. Ebenfalls sollten Sie sicher mit Ihrem Betriebssystem umgehen können und beispielsweise Editoren starten und Dateien speichern können. Apropos Editoren: Im Gegensatz zu HTML gibt es bei der JavaScript-Programmierung keinen großen Markt für Editoren. Hier geschieht das meiste immer noch textbasiert. Die einfachen Texteditoren, die mit dem Betriebssystem mitgeliefert werden (beispielsweise **Notepad** oder **SimpleEdit**), reichen sogar schon aus; wer mehr will, sollte einen der folgenden Editoren ausprobieren, die hier in der Reihenfolge meiner Präferenzen geordnet sind. (Sie sind aber Shareware und damit nach Ablauf einer Testperiode kostenpflichtig.):

- ▶ UltraEdit (<http://www.ultraedit.com>) – sehr leistungsfähiger Editor inklusive Syntax-Highlighting (Windows).
- ▶ Programmer's File Editor (<http://www.lancs.ac.uk/people/cpaap/pfe/>) – Die Entwicklung wurde mittlerweile eingestellt, aber unter der genannten Adresse ist dieser Editor immer noch erhältlich (Windows).
- ▶ NoteTab (<http://www.notetab.ch>) – mehrfach ausgezeichnete Klassiker unter den Texteditoren, auch in kostenfreier Light-Variante erhältlich (Windows).
- ▶ BBEEdit (<http://www.bbedit.com>) – der Macintosh-Klassiker, auch in einer Light-Version erhältlich.

Außerdem sollten Sie möglichst viele Browser installieren. Sie sollten nämlich Ihre JavaScript-Programme auf möglichst vielen Zielplattformen testen, um sicherzustellen, dass die Programme eben nicht nur bei Ihnen laufen. Wenn Sie den Internet Explorer 5 oder 5.5 über den Internet Explorer 4 installieren, müssen Sie bei der Installation unbedingt angeben, dass die alte Version (im so genannten **Kompatibilitätsmodus**) erhalten bleibt. Der Internet Explorer 3 lässt sich in der 32-Bit-Variante nicht parallel installieren, aber wenn Sie ein wenig beherzt sind, verwenden Sie die 16-Bit-Variante; das funktioniert (angeblich: Alle Installationen sind Ihr Risiko). Der Internet Explorer 6 überschreibt leider ausnahmslos alle Vorgängerversionen. Beim Netscape Navigator ist alles ein wenig einfacher. Die Versionen 7, 6, 4.x und 3 lassen sich problemlos parallel installieren. Ebenfalls sollten Sie den Opera-Browser installieren (<http://www.opera.com>), da dieser auch eine gute JavaScript-Unterstützung hat. Benutzern von Mac OS X steht der Safari-Browser von Apple zur Verfügung, der auf dem Linux-Browser Konqueror basiert. Weitere Informationen zu den verschiedenen Browsern können Sie Kapitel 2 entnehmen.

Während der Entwicklung Ihrer Programme sollten Sie unbedingt den Netscape Navigator verwenden. Diese Empfehlung hat nichts mit meinem persönlichen Geschmack zu tun, sondern beruht auf der Tatsache, dass die Fehlermeldungen des Netscape Navigators viel aussagekräftiger sind als die des Internet Explorers. Läuft ein Programm einmal im Netscape Navigator einigermaßen zuverlässig, dann können Sie natürlich sofort dazu übergehen, es im Internet Explorer zu testen.

Seit der Version 4.06 führt ein JavaScript-Fehler beim Netscape Navigator nicht mehr zu einem extra Warnfenster; stattdessen werden alle Fehlermeldungen in der JavaScript-Konsole angezeigt. Diese wird sichtbar, wenn Sie in der Adresszeile des Browsers `javascript:` eintippen (inklusive des Doppelpunkts) und auf `[Enter]` drücken. Im Netscape 6/7 geht das mit dem Menübefehl **Extras • Web-Entwicklung • JavaScript-Konsole**.



Es bleibt mir nur noch, Ihnen viel Spaß bei der Lektüre dieses Buches und viel Erfolg und Kreativität beim Experimentieren mit JavaScript zu wünschen. Und noch einmal die Bitte: Geben Sie mir Rückmeldung über **MyGalileo**, damit ich Ihre Anregungen für eine Neuauflage in Erwägung ziehen kann. Da ich sehr viele E-Mails bekomme, kann es mit einer Antwort etwas dauern, also werden Sie bitte nicht ungeduldig. Fragen zu Buchthemen beantworte ich gern, sofern es möglich ist; bei allgemeinen Fragen zu JavaScript sind Sie in den folgenden Newsgroups wohl besser aufgehoben und bekommen auch schneller eine Antwort:

- `de.comp.lang.javascript`
- `comp.lang.javascript`

Wenn Ihr Provider keine Newsgroup-Anbindung unterstützt, sollten Sie ihn wechseln oder sich ein kostenloses Konto bei <http://groups.google.de/> oder <http://netnews.web.de/> einrichten lassen.

Vielleicht finden Sie eine Antwort auf Ihre Frage aber auch in den FAQ der Newsgroup **de.comp.lang.javascript**, die Sie auf der Buch-CD finden. An dieser Stelle möchte ich mich sehr für die Erlaubnis bedanken, die FAQ beilegen zu dürfen. Lesen Sie die FAQ unbedingt durch, bevor Sie eine Frage in der Newsgroup stellen!

1.3 Danksagungen zu »JavaScript« (1. Auflage)

Manche Menschen haben den Danksagungsteil in einem Buch, ich dagegen lese ihn immer als Erstes. Hier folgen die Namen einiger Leute, die mich vor oder während der Bucherstellung in irgendeiner Art und Weise unterstützt oder motiviert haben oder mir anderweitig behilflich waren.

Zunächst einmal gilt der Dank meiner Lektorin Judith Stevens und ihrem Team bei Galileo, die mich fabelhaft betreut haben und stets ein offenes Ohr für meine Wünsche und Anregungen hatten. Herzlichen Dank dafür. Rudolf Krahm ist übrigens »schuld« daran, dass ich überhaupt an dieses Projekt gekommen bin.

Die sprachlichen Korrekturen wurden von Friederike Daenecke ausgeführt. Sie hat dadurch die sprachliche Qualität dieses Buches deutlich gesteigert.

Das schöne Cover wurde von Barbara Thoben entworfen, die Herstellung lag in den Händen von Petra Strauch, in Vertretung Claudia Lucht.

Für die USA-Erlebnisse habe ich folgenden Leuten zu danken: den Verkehrspolizisten von Colby, Tigger und Jerry S. Für China waren das Thommi, Reinhold, Klaus, Axel, Wei Dan und Qu Jian Ben. Zu Hause danke ich neben meiner Familie meinen Freunden Christian, Marianne, Markus und Thorsten; den PP-Kollegen, allen Caipirinha-Anhängern sowie den Sneak-Preview-Fans. Vielen Dank für die »Sabotage« (d.h. Ablenkung von) meiner Arbeit!

Besonders danken möchte ich außerdem noch meinen Freunden Tobias und Matthias, die immer hinter mir stehen und ohne deren Unterstützung ich das Buch wohl nicht hätte schreiben können. Tobias hat übrigens alle Vektorgrafiken für dieses Buch erstellt.

In fachlicher Hinsicht habe ich vieles von dem, was ich weiß, durch das Diskutieren in Mailinglisten und Newsgroups gelernt. Exemplarisch für viele andere möchte ich hier IT und QA nennen, die durch ihr Wissen und ihre Kreativität schon vielen JavaScript-Programmierern geholfen haben.

Last but not least danke ich Thomas Maier, der mir vor ein paar Jahren die Freude am Schreiben wiedergegeben hat, sowie Cecily, die immer an dieses Projekt geglaubt hat.

Christian Wenz, im November 1999

1.4 Danksagungen zu »JavaScript-Rezepte« (1. Auflage)

Die grobe Inhaltsübersicht und der Name des Buchs entstanden am letzten Tag des Jahres 2000 in Ägypten, in der Nähe des Dreiländerecks. Dafür, dass weder das Konzept noch ich danach im Pool oder im Meer versenkt wurden, möchte ich Bettina danken (und natürlich noch für vieles mehr).

Das Konzept nahm dann immer konkretere Formen an, bis die eigentliche Schreibarbeit dann im Sommer begann, zum Teil in München, zum Teil im toskanischen Exil. Während dieser Zeit hat mich alleine schon die Vorfreude auf den festen Ablauf des Freitagabends zum Arbeiten motiviert: Zunächst das Essen im »Il Piccolo Principe« (mit Andreas, Christian, Hilmar & Special Guests), danach die »Sneak Preview« im Cinema (mit Andreas, Christian, Jörg, Markus, Thorsten, Yvonne & Special Guests). Herzlichen Dank dafür.

Besonderer Dank gebührt des Weiteren Tobias und Matthias, die nicht nur immer hinter mir stehen, sondern mit denen ich auch 18 Stunden am Tag lachen könnte. Danke!

Ebenfalls möchte ich Yvonne danken, die mich während der Bucherstellung toll unterstützt hat.

Die Betreuung von Verlagsseite aus konnte glücklicherweise von meiner Traumkonstellation übernommen werden, die auch schon für die JavaScript-Referenz verantwortlich zeichnete. Das Lektorat hat Judith Stevens-Lemoine übernommen, die sprachliche Korrektur hat Friederike Daenecke durchgeführt (was man, wie ich hoffe, auch beim Durchlesen merkt). Der Umschlag wurde von Barbara Thoben gestaltet.

Die Listings in diesem Buch wurden von Stefan Krumbiegel und Anja Horch überprüft; so konnten noch einige Fehler ausgemerzt werden.

Bleibt mir nur noch, Ihnen viel Spaß beim Lesen und Schmökern in diesem Buch und beim Arbeiten mit den Rezepten zu wünschen!

Christian Wenz, im September 2001

1.5 Vorwort zur 5. Auflage

Eine neue Auflage, und dieses Mal fallen einige Neuerungen bereits beim ersten Blick auf: Das Cover ist neu gestaltet worden, der Buchumfang hat die 1000-Seiten-Grenze überschritten. Was aber hat sich im Inneren getan?

Die größte Neuerung ist der umfangreiche Praxisteil. Damit komme ich dem Wunsch zahlreicher Leser nach, das JavaScript-Buch und den Titel »JavaScript-Rezepte« zu vereinen. Gesagt, getan: Nach der bewährten Einführung in JavaScript werden im hinteren Buchteil zahlreiche fertige JavaScript-Rezepte vorgestellt, die Sie direkt in eigene Webseiten einbauen können – sozusagen ein Werkzeugkasten für JavaScript-Entwickler. Gewisse Redundanzen sind unvermeidlich, aber sehr gering gehalten. Die Ausrichtungen der verschiedenen Buchteile sind auch verschieden: Am Anfang geht es um eine Vorstellung der Thematik und um die Einführung in die JavaScript-Programmierung; der hintere Buchteil versucht dagegen die Grundlagen in nützliche Mosaiksteine zu kapseln, aus denen dann eine Website mit viel JavaScript erstellt werden kann. Die erste Buchhälfte ist also zum Lernen, die zweite zum Einsetzen und Anwenden. (Und nebenbei lernen Sie doch noch den einen oder anderen kleinen Kniff kennen.) Der dritte und letzte Buchteil enthält eine JavaScript-Objektreferenz, die Lösungen zu den Übungsaufgaben aus dem ersten Teil sowie nützliche Internetquellen.

Auch sonst gibt es viele Neuerungen. Jedes Kapitel wurde komplett durchgesehen, ergänzt und korrigiert. Ganz neue Kapitel gibt es auch. Beispielsweise werfen wir einen Blick über den Tellerrand: Ein Teil von Microsofts ominöser .NET-Strategie ist auch eine Sprache namens JScript .NET, bei der JavaScript deutlich Pate gestanden hat. Zudem wurden einige neue Rezepte hinzugefügt. Wie immer sind Sie dazu aufgerufen, dem Autor oder Verlag gegenüber kundzutun, welche Inhalte Sie gern noch ausführlicher dargestellt finden würden; so können Sie zukünftige Auflagen entscheidend mitgestalten.

Aber auch Konkurrenzbrowser sollen nicht unberücksichtigt bleiben. Der Macintosh-Plattform wird in dieser Auflage eine besondere Bedeutung zuerkannt, insbesondere dem neuen Apple-Webbrowser Safari. Unter der Linux-Plattform ist der Konqueror, übrigens die Basis von Safari, sehr verbreitet. Er wird ebenfalls betrachtet. Trotz des eher geringen Marktanteils dieser Webbrowser äußerten viele Leser den Wunsch, diese Browser ausführlicher zu berücksichtigen. Außerdem ist dieses Buch schon in seinen Vorgängerversionen bekannt dafür gewesen, auch ältere oder seltenere Browser zu behandeln. An dieser Stelle ein Hinweis zur aktuellen Marktlage: Der Netscape Navigator 4.x stirbt langsam, aber sicher aus. Die bekannte W3B-Umfrage ermittelte im Jahre 2002, dass der Netscape 4 interessanterweise mit allen neueren Netscape-Versionen inklusive Mozilla gleichauf liegt – eine peinliche Schlappe für das ehrgeizige Mozilla-Projekt. Zwar ist abzusehen, dass der »alte« Netscape mittelfristig deutlich überflügelt werden wird, aber dennoch muss er heutzutage noch

berücksichtigt werden. Professionelles Webdesign zeichnet sich gerade dadurch aus, dass eine möglichst große Zielgruppe bedient wird. Die zurzeit um sich greifende Manie – insbesondere in Zeitschriften –, dass JavaScript-Programme ausschließlich im Internet Explorer laufen (nicht einmal in neuen Netscape-Versionen), sorgt dafür, dass Websites nur noch in einem Webbrowser laufen, obwohl es doch so einfach wäre, auch andere Browser zu unterstützen. Ältere Browserversionen, vornehmlich der Internet Explorer und Netscape Navigator, jeweils Version 3 oder früher, haben mittlerweile keinen messbaren Marktanteil mehr. Aus diesem Grund werden Hinweise auf Fehler und Besonderheiten dieser Versionen nicht mehr in dieser Auflage behandelt, von einigen Randbemerkungen einmal abgesehen.

Bei einem solch umfangreichen Buch können kleinere Fehler leider nie ganz ausgeschlossen werden. Zwar hat dieses Buch eine Reihe von Prüfprozessen durchlaufen, dennoch ist es so gut wie sicher, dass sich der sprichwörtliche Fehlerteufel irgendwo eingeschlichen hat. Wenn Sie also einen Fehler gefunden zu haben, wenden Sie sich über den Verlag an mich. Dann kann der Fehler in der nächsten Auflage ausgemerzt werden. Außerdem werden bekannt gewordene Fehler sowohl im Forum von Galileo Press als auch auf der Support-Seite zum Buch unter <http://www.hauser-wenz.de/support/> veröffentlicht. An genau dieselben Stellen können Sie sich auch wenden, wenn Sie Fragen oder Anregungen zum Buch haben. Ich freue mich auf Ihr Feedback! An dieser Stelle Danke an Felix Siegrist, der viele Verbesserungsvorschläge zum Praxisteil beigesteuert hat.

Allen Neuerungen zum Trotz, eine Sache ist in allen Auflagen gleich geblieben: Judith Stevens-Lemoine hat zum fünften Mal als Lektorin das Projekt geleitet, und Friederike Daenecke hat, ebenfalls zum fünften Mal, als sprachliche Korrektorin meine Ergüsse in lesbare Form gebracht. Der Erfolg des Buchs ist also das Ergebnis eines eingespielten Teams; herzlichen Dank dafür! Und: Danke auch an Sie, dass Sie dieses Buch gekauft haben.

Christian Wenz, im August 2003

P.S.: Besondere Glückwünsche an Yvonne! Sie hat ihr Studium beendet und darf sich jetzt »Dipl.-Ing.« nennen. Ich werde nie wieder etwas gegen FH-Studenten sagen, versprochen!

1.6 Vorwort zur 6. Auflage

JavaScript gewinnt an Fahrt. Das spüren wir nicht nur an den (erfreulichen) Buchverkäufen, die diese Neuauflage möglich machen, sondern auch an dem zunehmenden Interesse an Schulungen und vermehrten Projektanfragen. Mit dem nun schon fast vollzogenen Aussterben des Netscape 4 verabschiedet sich eine große Hürde der browserunabhängigen Webentwicklung langsam, aber sicher von der Bühne. Natürlich gibt es immer noch Unterschiede zwischen den

restlichen Browsern, aber diese sind bei weitem nicht so gravierend wie die teils großen Umwege, die man für den Netscape 4 gehen musste. Das wird sich auch in diesem Buch niederschlagen. Diese Auflage ist die letzte mit spezifischer Unterstützung für Netscape 4; als eine Art Hommage sind ein Großteil der Abbildungen noch mit dem Browser-Urgestein erstellt worden. Die Planungen für die nächste, siebte Auflage laufen bereits in vollen Zügen. Sie wird große Veränderungen bringen; zum Erscheinungszeitpunkt nämlich wird der Netscape 4 aller Voraussicht nach überhaupt keine Rolle mehr spielen.¹ Zurzeit ist es jedoch zumindest noch vertretbar, ein Auge auf Netscape 4 zu werfen und durch seine explizite Unterstützung den Anteil der potenziellen Besucher (und damit Kunden) etwas zu erhöhen.

So viel zu alten Dingen, die Sie (noch) in diesem Buch finden werden. Doch nun zu den Neuerungen. Da in jeder neuen Auflage viel Überarbeitungsaufwand getrieben wird, schleichen sich leider immer wieder kleine Fehler ein. Alle, die zum Zeitpunkt der Manuskriptabgabe bekannt waren, sind behoben worden. Besonderen Dank an dieser Stelle an Kurt Lidwin, der mit Abstand am meisten Anregungen geschickt hat (und sie hoffentlich zu einem großen Teil in dieser Auflage wiederfindet).

Ansonsten finden Sie in fast jedem Kapitel an der einen oder anderen Stelle etwas Neues – in der Summe zu viel, um alles einzeln aufzuzählen. Stark erweitert wurde das Kapitel rund um Web Services. Wie immer benötige ich für zukünftige Auflagen Ihre Mithilfe: Finden Sie bestimmte der im Buch behandelten Themen so interessant, dass ausführlicher darüber geschrieben werden sollte? Dann melden Sie sich; Ihr Feedback geht dann in die Planungen der 7. Auflage mit ein.

Zum Schluss drei Punkte, die schon seit Jahren und in allen Auflagen zur Tagesordnung gehören: Danke an meine Lektorin Judith Stevens-Lemoine, danke an meine Korrektorin Friederike Daenecke, und Ihnen viel Spaß und Erfolg mit JavaScript!

Christian Wenz, im April 2005

¹ Ich hatte unlängst einen Termin bei meiner Bank. Die hatten dort tatsächlich Netscape 4.0 (!) – unter OS/2 (!!). Ich habe dann gleich bei der Konkurrenz ein Konto eröffnet.

1.7 Die Icons in diesem Buch

Wichtige **Hinweise** finden Sie in Abschnitten, die mit diesem Symbol gekennzeichnet sind.



Achtung: Abschnitte mit diesem Symbol enthalten eine **Warnung**!



Dieses Icon markiert Abschnitte, in denen Sie auf Besonderheiten der unterschiedlichen **Browserversionen** und **-typen** aufmerksam gemacht werden. Auch die Eigenschaften der verschiedenen **JavaScript-Versionen** werden an diesen Stellen beschrieben.



Hier werden Sie auf Softwarefehler hingewiesen – jeder kennt sie unter dem Begriff **Bug**.



Arbeitsvereinfachungen, Tastenkürzel und Schreibvereinfachungen finden Sie neben diesem Symbol. Hier stoßen Sie auf nicht ganz alltägliche **Profitipps**.



2 Vorbereitung

*Take three months to prepare your machines and
three months to complete your siege engineering.
– Sun Tzu, The Art Of War*

In diesem Kapitel erfahren Sie zunächst, welche Browser überhaupt JavaScript unterstützen und wie Sie Ihr Testsystem einrichten können. Außerdem lernen Sie, wie JavaScript-Code in HTML-Dokumente eingebunden wird. Besondere Beachtung verdienen hierbei die Unterschiede zwischen den einzelnen Browsern.

2.1 Webbrowser

JavaScript ist eine so genannte clientseitige Programmiersprache. Das heißt, JavaScript-Programme werden im Webbrowser ausgeführt. Dies wird dadurch möglich, dass JavaScript in HTML eingebettet wird. HTML-Dateien können also JavaScript-Code enthalten. Wenn Sie im World Wide Web unterwegs sind, ist auf der Mehrheit der Seiten, die Sie aufsuchen, JavaScript enthalten.

Nicht jeder Browser unterstützt JavaScript, aber die meisten gebräuchlichen Browser tun es. Zunächst ist es interessant zu wissen, welche Webbrowser überhaupt zurzeit gebräuchlich sind.

2.1.1 Netscape Navigator (und Konsorten)

Der wohl erste weithin bekannte Webbrowser ist der Netscape Navigator, der seit Mitte der 90er Jahre für verschiedenste Plattformen verfügbar ist. Als Nachfolger des Mosaic, des Urvaters aller grafischen Webbrowser, war er der erste Webbrowser mit JavaScript-Unterstützung, denn Netscape selbst hat JavaScript erfunden und geschaffen, um die (beschränkten) Möglichkeiten von HTML zu erweitern. Anfangs hieß die Programmiersprache noch LiveScript, was aber die Marketingabteilung von Netscape nicht sonderlich schick fand. Die »Rettung« kam in Form der Firma Sun, die mit ihrer als plattformunabhängig konzipierten Programmiersprache Java Furore machte. Der clevere Schachzug von Netscape bestand darin, den Namen »Java« zu lizenzieren, und schon hieß die Programmiersprache JavaScript. Zugegeben, die Sprachsyntax (Vorschriften, wie ein Programm auszusehen hat, Namen von Kommandos etc.) ist ähnlich, aber ansonsten haben die Sprachen Java und JavaScript nichts, aber auch gar nichts miteinander gemeinsam. Das ist leider auch heute, fast zehn Jahre später, noch nicht überall bekannt, und »Fachleute« reden von Java, meinen aber JavaScript. Die Namensgleichheit ist also eine Marketingmaßnahme und hat keine technische Begründung.



Dennoch gibt es Möglichkeiten, Java und JavaScript zusammenarbeiten zu lassen. In Kapitel 20 wird verraten, wie.

Aber zurück zur Historie des Netscape-Browsers: Version 2 führte JavaScript ein, Version 3 erweiterte die Möglichkeiten. Beispielsweise wurden Rollover-Effekte (auch Mouseover-Effekte genannt) möglich, siehe Kapitel 9. Version 3 gab es zudem als »Gold«-Version, in der auch ein Mailprogramm enthalten war. Netscape hatte damals erkannt, dass eine Kundenbindung an den Browser auch dadurch erzielt werden kann, dass Browser und Mailsoftware untrennbar miteinander verbunden sind. So öffnen sich Links in Mails direkt im zugehörigen Browser.



Abbildung 2.1 Lang' ist's her: Netscape 0.91 (nicht JavaScript-fähig)

Version 4 führte zunächst eine Namensänderung ein: Der Browser als Stand-alone-Produkt hieß weiterhin »Netscape Navigator«; das gesamte Paket inklusive Mailprogramm, Adressbuch und (eine Zeit lang) Terminverwaltung wurde »Netscape Communicator« getauft. Die Entwicklung des Navigator-Produkts, also ohne kundenbindende Zusatzkomponenten, wurde allerdings bald eingestellt; Version 4.08 war die letzte ihrer Art. Das Communicator-Paket dagegen erfuhr auch im Jahre 2003 noch ein Update, wenngleich auch vermutlich das allerletzte: Mit Version 4.8 dürfte die Entwicklung ihr Ende gefunden haben, auch wenn einige Bugs seit Jahren bekannt und unbeboben sind.

Für den JavaScript-Programmierer ist der Netscape 4 insbesondere in Hinblick auf die JavaScript-Unterstützung häufig ärgerlich. Netscape hat in dieser Version einige neue, proprietäre (eigene) Erweiterungen eingeführt. Diese fanden teilweise so wenig Zustimmung, dass sie in keinem anderen Browser und auch in keiner neuen Netscape-Version weitergeführt wurden. Insbesondere in den DHTML-Kapiteln (Kapitel 16 bis 19) werden wir mit einigen Kraftakten ver-

suchen, JavaScript-Programme auch auf dem Netscape 4 zum Laufen zu bringen. Anfangs war das noch kein Problem – Netscape 4 war der Marktführer. Das sollte sich jedoch bald ändern, und ab diesem Zeitpunkt wurde es ein Problem.

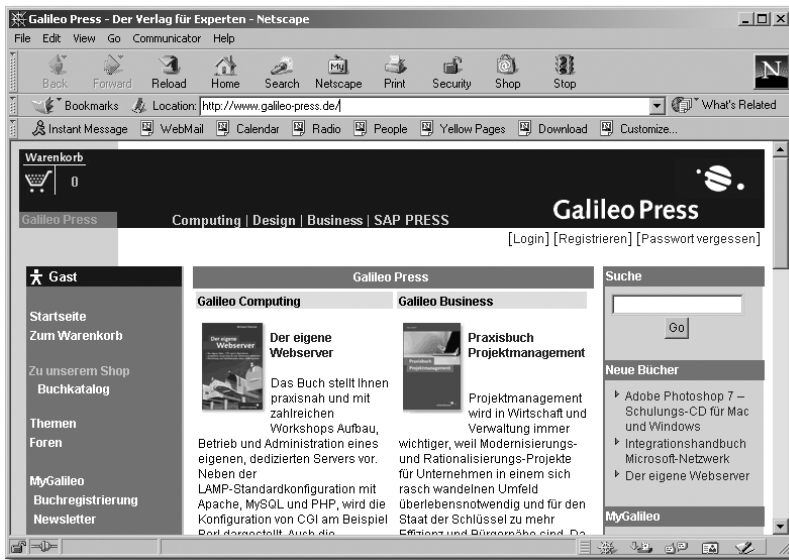


Abbildung 2.2 Ursache vielen (JavaScript-)Übels: Netscape 4

In der Folgezeit ist einiges passiert. Die Firma Netscape wurde von AOL gekauft, und die Entwicklung des Netscape-Browsers wurde zunächst eingestellt. Allerdings wurde ein neuer, zu dieser Zeit noch als revolutionär zu bezeichnender Ansatz gewählt: Das Open-Source-Projekt Mozilla (<http://www.mozilla.org/>) wurde aus dem Boden gestampft. Sein Ziel sollte es sein, einen neuen Netscape-Browser zu entwickeln. Das Wörtchen »Netscape« taucht auf den Mozilla-Seiten selten auf, allerdings sind es größtenteils Netscape- bzw. AOL-Mitarbeiter, die sich an der Entwicklung beteiligen. Als schließlich die ersten Mozilla-Versionen erschienen, war der Marktanteil des Netscape bereits im einstelligen Bereich, denn zu lange hatte sich bei diesem Browser nichts getan. In ihrer Verzweiflung veröffentlichten die AOL-Verantwortlichen eine »Netscape«-Version des Mozillas. Um mit der damals aktuellen Version des Internet Explorer gleichzuziehen, erhielt dieser Netscape die Versionsnummer 6. In Hinblick auf die Produktqualität war diese Version leider ein totaler Reinfall: Die JavaScript-Unterstützung wies zahlreiche Fehler auf, und der Browser war langsam, schwerfällig und stürzte zudem häufig ab. Der Image-Verlust war enorm, und auch Netscape-Enthusiasten geben heute unverwunden zu, dass Version 6 wirklich schlecht war (was sie nicht daran gehindert hatte, Jahre zuvor jede und jeden zu verteufeln, sollte ein böses Wort gegen den Netscape 6 gesagt werden). Version 7, die einige Zeit später erschien, war deutlich besser, und die aktuelle Version 7.1 ist wirklich gut. Eine wenig veränderte Version 7.2 folgte kurze Zeit

später. Doch das kommt vermutlich zu spät. Der Marktanteil ist weiter stetig gesunken, und AOL hat sich außerdem von einer Reihe von Netscape-Mitarbeitern getrennt. Der vorerst letzte Akt in diesem Trauerspiel: Es wurde eine »Mozilla Foundation« gegründet, die sich um die weitere Entwicklung des Open-Source-Projekts kümmern soll. Für den Netscape jedoch hat wohl die letzte Stunde geschlagen: Die Version 7.2 ist aller Voraussicht nach die letzte des einstigen Klassenprimus (und besitzt einige empfindliche Sicherheitslücken). Schade, denn die neuen Versionen sind in Hinblick auf die JavaScript-Unterstützung vorbildlich, und der Browser ist auch nicht mehr so langsam wie früher. Allmählich werden auch die Horden von Bugs unter Kontrolle gebracht. Die Zukunft wird zeigen, ob und wie die Weiterentwicklung ohne größere AOL-Unterstützung laufen wird.

Still und heimlich hat sich jedoch eine wahre Nummer 2 auf dem Browsermarkt etabliert. Nachdem die Kritik über das große und unübersichtliche Mozilla-Projekt lauter wurde, hat ein separates Projekt einen eigenen Webbrowser erstellt, ebenfalls unter dem Dach von Mozilla. Basis ist die Gecko-Engine, also das Herz von Mozilla, das für das Rendern von HTML (und auch für das Ausführen von JavaScript-Code) zuständig ist. Die Oberfläche allerdings ist eine eigene. Der Browser ist dadurch (relativ) schlank und erfreut sich mittlerweile großer Beliebtheit; die Aktion »Spread Firefox« hatte so viele Spenden gesammelt, dass ganzseitige Anzeigen für den Browser in der New York Times und in der Frankfurter Allgemeinen geschaltet werden konnten.¹ Mittlerweile ist der Firefox von allen Mozilla-basierten Browsern der erfolgreichste und jagt dem Marktführer Marktanteile ab. Bis zur Weltherrschaft ist es freilich noch ein langer Weg, denn einige der Schwächen von Mozilla, etwa dass einige Bugs schon seit Jahren auf Behebung warten, hat der Firefox-Browser natürlich auch geerbt.

Mozilla und Firefox gibt es für die wichtigsten Betriebssysteme. Unter dem Mac gibt es jedoch ein eigenes Mozilla-Subprojekt, das speziell auf diese Plattform optimiert ist: Camino, erhältlich unter <http://www.mozilla.org/products/camino/>. Wer auf Linux setzt und dabei Gnome als Fenster-Manager einsetzt, wird möglicherweise mit Galeon (<http://galeon.sf.net/>) glücklich, einer auf dieses System optimierten Mozilla-Variante.

Für Nostalgiker: Ein neuer Netscape-Browser ist ebenfalls in Vorbereitung, allerdings nur als Browser, ohne integriertes E-Mail-Programm. Ein geschlossener Kreis darf eine erste Version beta-testen. Im Hintergrund werkelt die Firefox-Engine, der Browser besitzt allerdings noch einige Zusatz-Features wie etwa die Möglichkeit, bei extrem auf den Internet Explorer optimierten Seiten per Mausklick die IE-Engine zu laden – dann läuft der Internet Explorer innerhalb des Netscape-Browsers ab. Das ist natürlich für eingefleischte Netscape-Fans eine gruselige Vorstellung. Richtig gruselig ist allerdings die Oberfläche des neuen Browsers; in Abbildung 2.4 sehen Sie den aktuellen Stand (aus dem

¹ Ich gebe zu, auch einige Buchtantiemen sind der Organisation zu Gute gekommen.

nicht-öffentlichen Beta-Test). Seien Sie froh, dass dieses Buch nicht farbig ist – die Hauptfarbe des Browsers ist Giftgrün. Es gibt aber Bestrebungen, das noch zu ändern. Doch in Wirklichkeit ist der Netscape-Browser nur ein besserer Firefox, so dass die Motivation für viele nicht groß ist, dem »Feuerfuchs« (der in Wirklichkeit ein Panda ist) den Rücken zu kehren.



Abbildung 2.3 Der (vorerst letzte) Netscape 7

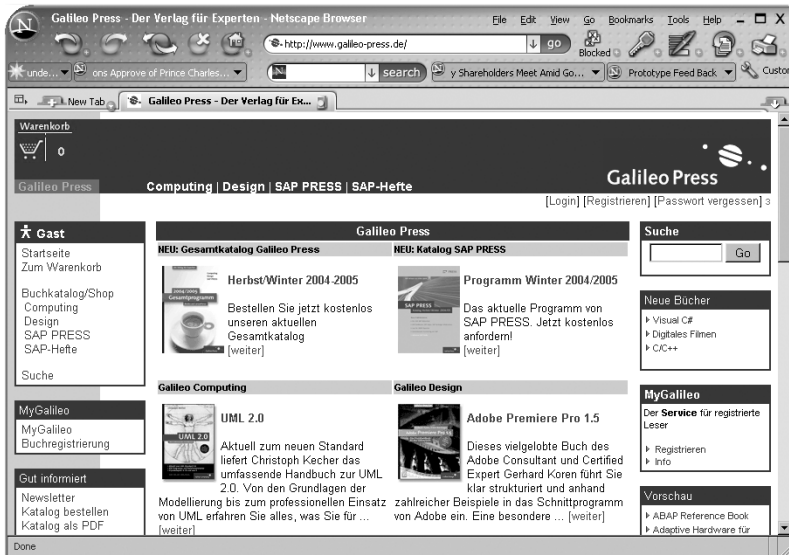


Abbildung 2.4 Die (mögliche) Zukunft: Netscape 8

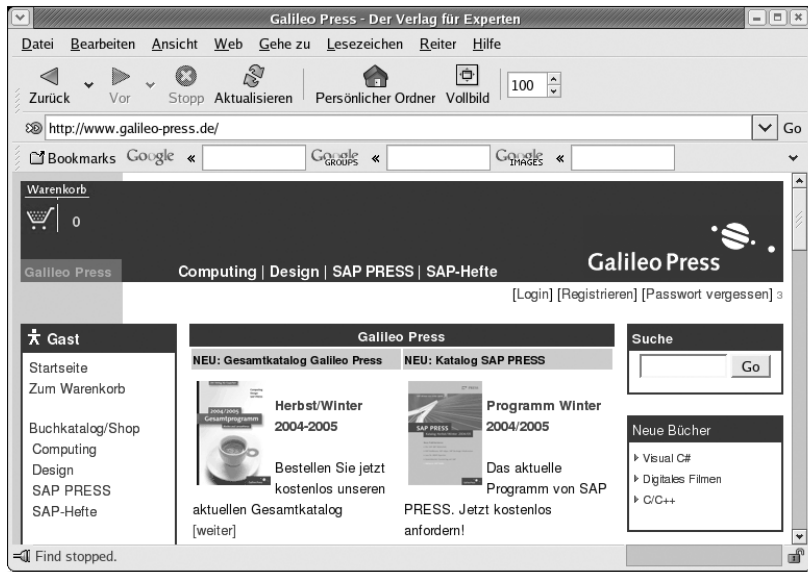


Abbildung 2.5 Mozilla speziell für Gnome: Der Galeon-Browser

2.1.2 Microsoft Internet Explorer

Der Microsoft Internet Explorer ist seit einiger Zeit unangefochtener Spitzenreiter im Browsermarkt. Zunächst hatte Microsoft bekanntermaßen das Internet »verschlafen«; die ersten Internet Explorer-Versionen 1.x und 2.x waren kaum zu gebrauchen. Mit dem Internet Explorer 3 änderte sich dies radikal. Der Browser wurde langsam, aber sicher konkurrenzfähig. Viele Eigenschaften des damals technologisch und auch in Sachen Verbreitung haushoch überlegenen (und in einer Beta-Version vorliegenden) Netscape Navigator 3 wurden integriert. Die Version 3 ist auch die erste Version des »IE«, die JavaScript unterstützt. Die Nachfolgeversion 4 schließlichschloss die technologische Lücke zum Netscape. (Die Versionsnummern waren lange Zeit gleich, Netscape 4 erschien annähernd zeitgleich zum IE 4.) Aufgrund einiger Netscape-Macken (siehe den vorangegangenen Abschnitt) begann der IE mit dieser Version langsam, aber stetig an Marktanteilen zu gewinnen. Zusätzlich zu den technologischen Vorteilen (die mit den Nachfolgeversionen 5.0, 5.5 und 6.0 den Vorsprung noch vergrößern sollten) nutzte Microsoft auch geschickt, aber bedenklich, die Marktmacht auf dem Desktop: Der Internet Explorer ist bei Windows automatisch dabei, er kann nicht oder kaum entfernt werden, und mit etwas »Glück« verlangt eine neue Version von Microsoft Office oder einer anderen Anwendung eine neuere Browserversion.



Abbildung 2.6 Der Marktführer: Microsoft Internet Explorer

Mittlerweile kann sich Microsoft über eine eindeutige Marktführerschaft freuen. Allerdings scheint sich auch hier die Browserentwicklung einem Stillstand zu nähern. Microsoft hat Mitte 2003 bekannt gegeben, dass Version 6.0 die wohl letzte sein wird. Das einzige nicht sicherheitsrelevante Update in letzter Zeit gab es als Teil von Windows XP in Form des Internet Explorer 6.05, inklusive integriertem Popup-Blocker. Neuere Versionen des Internet Explorer – so die Logik aus der Microsoft-Zentrale in Redmond – verlangen eine neue Windows-Version. Das spricht also für einen »IE7« in Longhorn, dem Nachfolger von Windows XP, mit dem 2006 oder 2007 gerechnet wird (möglicherweise gibt es noch im Sommer 2005 eine Beta-Version des IE7). Wer sich also über die (bekannten) Bugs des IE ärgert, wird sich unter Umständen etwas gedulden müssen, bis sie behoben werden.

Der IE ist häufig im Zentrum der Kritik, weil er (auch aufgrund seiner großen Verbreitung im Markt) oft zum Ziel von Virenautoren wird. Wenn Sie Windows frisch installieren, ist der Internet Explorer anfällig! Abhilfe schafft ein regelmäßiger Besuch bei Windows Update (<http://windowsupdate.microsoft.com/>), um aktuelle Sicherheitspatches zu installieren.

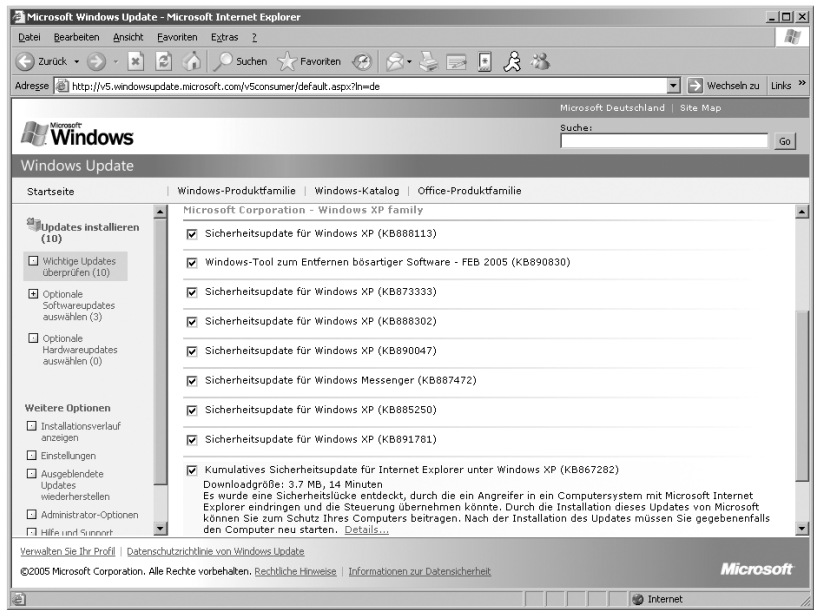


Abbildung 2.7 Windows Update ist fündig geworden.



Abbildung 2.8 Der Internet Explorer unter Mac OS X

Den Internet Explorer gibt es natürlich hauptsächlich für Windows, aber auch Mac-Versionen stehen zur Verfügung, allerdings alle mit der Versionsnummer 5. Auch hier ist es fraglich, ob die Entwicklung weitergeht. Immerhin, die OS-X-Version ist neuer als die für OS 9 – aber was heißt das schon? Die Macintosh-Versionen hinken schon immer den Windows-IEs hinterher, und insbesondere die JavaScript-Unterstützung hat Lücken (siehe Abbildung 2.8).

2.1.3 Opera

Nummer 3 im großen Browser-Reigen ist der kleine Opera-Webbrowser aus Norwegen. »Klein« deshalb, weil er in seinen früheren Versionen marketing-technisch geschickt auf eine 1,44-MByte-Diskette kopiert werden konnte, so klein (und auch schnell) war er. Mittlerweile ist der Browser schon größer geworden, aber immer noch äußerst schlank. Die JavaScript-Unterstützung ist spätestens seit Version 7 auch wirklich gut; Version 6 bot häufiger Anlass zum Ärgern aufgrund zahlreicher Fehler. Bei einem maximal einstelligen Marktanteil lohnt es sich hier nicht, alte, fehlerbehaftete Versionen weiterhin zu unterstützen, deswegen wird auch in diesem Buch der Fokus auf Version 7 gelegt (zurzeit, also im Februar 2005, ist Version 8 gerade als erste Beta verfügbar).



Abbildung 2.9 Der Opera-Browser

Eine »Besonderheit« des Opera ist, dass der Browser zwar kostenlos ist, aber immer ein Werbebanner rechts oben sichtbar ist. Wer eine werbefreie Version will, muss zurzeit 39 \$ berappen.

2.1.4 Konqueror

Das freie Betriebssystem Linux gewinnt immer mehr an Fahrt. Zwar ist es von der oft beschworenen Bezwungung von Microsoft Windows auf dem Desktop noch sehr weit entfernt, aber trotzdem ist Linux »in«, und auf Servern hat es Microsoft bereits abgehängt.

Für die grafische Darstellung der Desktop-Oberfläche ist der so genannte Fenstermanager zuständig. Hier gibt es immer wieder Kleinkriege innerhalb der Linux-Gemeinde, welches System denn nun das beste sei, aber auch hier gibt es einen Marktführer: KDE (<http://www.kde.org/>). Ein Teil des Systems ist ein eigener Webbrowser, der Konqueror. Positiv für JavaScript-Entwickler ist, dass die Skriptsprache von Konqueror überaus gut unterstützt wird. Aufgrund des in Zukunft wohl noch weiter steigenden Marktanteils dieses Browsers ist also ein Linux-System für Sie als JavaScript-Entwickler Pflicht. Für ganz Faule gibt es sogar Linux-Distributionen, die von CD laufen und keine Installation benötigen. Bekanntestes Beispiel hierfür ist Knoppix (<http://www.knoppix.de/>). Für ernst zu nehmende Tests ist freilich eine »stationäre« Linux-Installation Pflicht. Unter <http://www.linuxiso.org/> finden Sie CD-Images einiger Linux-Distributionen, oder Sie kaufen gleich eine Distribution inklusive Handbuch und Installations-support.



Abbildung 2.10 Der Linux-Browser Konqueror, Teil des KDE

2.1.5 Safari

Die Beziehung zwischen Microsoft und Apple ist äußerst undurchsichtig. Apple-Chef Steve Jobs präsentierte vor ein paar Jahren Bill Gates als Retter seines Unternehmens (Microsoft erstand damals Apple-Anteile und spülte so Geld in Jobs' arg leere Kassen), allerdings ist Microsoft einer der ärgsten Hauptkonkurrenten auf dem Markt. So passt es auch, dass der Internet Explorer bei Mac OS sowie OS X automatisch mit dabei ist, Apple aber seit 2003 ebenfalls einen eigenen Webbrowser anbietet. Sein Name ist Safari, und er steht nur für OS X zur Verfügung. Der Grund: OS X basiert auf Unix, die Nähe zu Linux ist insofern offensichtlich. Apple hat nun keinen neuen Browser von Grund auf programmiert – dies wäre in Anbetracht des technologischen Vorsprungs der Konkurrenz auch ein unkalkulierbares Risiko –, sondern hat einfach den Konqueror auf OS X portiert! So profitieren gleich zwei Gruppen von dieser Entwicklung: Das Konqueror-Projekt (bzw. die KDE-Entwickler) profitiert von Fixes und unter Umständen auch von veröffentlichten Erweiterungen der Apple-Techniker am Konqueror; und Apple selbst freut sich natürlich über einen verfügbaren, frei einzusetzenden Webbrowser sowie über eine große Anzahl von Programmierern, die den Konqueror gratis weiterentwickeln. Die Macintosh-Plattform ist, allen Unkenrufen zum Trotz, nicht totzukriegen und insbesondere in Werbeagenturen sehr weit verbreitet. Der Safari ist also ein Webbrowser, den es – wie den Konqueror auch – zu beobachten gilt.



Abbildung 2.11 Apples eigener Browser: Safari (auf Konqueror-Basis)

Von diesen fünf Browsern einmal abgesehen, gibt es nur noch wenige Browser mit einer nennenswerten JavaScript-Unterstützung. Sun hat den HotJava-Browser im Angebot, einen komplett Java-basierten Webbrowser, der allerdings überhaupt keine nennenswerte Verbreitung (und auch keine besonders nennenswerte JavaScript-Unterstützung) aufweist. Fazit: Die zuvor genannten Webbrowser werden für ein professionelles Testsystem benötigt, weitere noch nicht, zumindest zu diesem Zeitpunkt noch nicht.

2.1.6 Marktanteile

»Glaube nie einer Statistik, die du nicht selbst gefälscht hast« – dieser Spruch wurde lange Zeit irrtümlich Winston Churchill zugeschrieben. Trotzdem ist an dieser Aussage etwas dran, denn je nach ideologischer Ausrichtung einer Website und deren Zielgruppe sind die Marktanteile von Browsern andere. Abhilfe schaffen hier unabhängige Instanzen, die global über das Internet untersuchen, welcher Webbrowser welche Verbreitung hat.

In Deutschland ist hier insbesondere die Unternehmensberatung Fittkau & Maaß zu nennen, die regelmäßig die W3B-Umfrage (<http://www.w3b.de/>) durchführt und dabei auch die Browserverteilung untersucht. Anfang 2004 ergab diese Untersuchung die folgenden Marktanteile:

Browser	Marktanteil
Internet Explorer	92,2 %
Netscape/Mozilla	5,4 %
Andere	2,4 %

Tabelle 2.1 Browsermarktanteile gemäß W3B²

Interessant sind hier zusätzliche Details: Immerhin 0,6 % aller Internetnutzer verwenden die »uralte« Version 4 von Netscape, die neueren Versionen sorgen für weitere 4,8 % Marktanteil (2,6 % für Version 6, 2,2 % für Version 7). Der alte Netscape wird also immer noch von über 10 % aller Netscape/Mozilla-Nutzer eingesetzt! Deswegen wird auch in diesem Buch noch ein gewisses Augenmerk darauf gelegt, dass möglichst alle Beispiele gerade im Netscape 4 problemlos laufen, allen Makeln und Mankos dieses Browsers zum Trotz. Wer sich damit schmückt, neuere Netscapes zu unterstützen, aber dabei die alten vernachlässigt, ignoriert einen nicht so unerheblichen Anteil vom Kuchen. Opera ist mit 1,5 % Marktanteil zwar die Nummer drei, aber deutlich abgeschlagen.

² Quelle: <http://www.w3b.org/trends/browserwatch.html>

Während W3B sich auf deutsche Websurfer beschränkt, führt OneStat.com (<http://www.onestat.com/>) weltweite Untersuchungen durch. Im November 2004 veröffentlichte OneStat.com folgende Zahlen:

Browser	Marktanteil
Internet Explorer	88,9 %
Firefox	7,35 %
Mozilla/Netscape	2,77 %
Opera	1,3 %
Safari	0,9 %

Tabelle 2.2 Browsermarktanteile gemäß OneStat.com³

Auch hier ist eine weitere Aufschlüsselung der Ergebnisse interessant. Der Internet Explorer 6 hat einen Marktanteil von 80,95 %, vor den Versionen 5.5 (3,66 %) und 5.0 (4,18 %, also mehr als beim IE 5.5). Dies zeigt, dass auch ältere Versionen des IE noch berücksichtigt werden müssen (später hierzu mehr). Der alte IE4 ist kaum mehr messbar (etwa 0,1 %). Recht klein ist noch der Marktanteil des Safari-Browsers (0,9 %), aber das geht wohl mit dem relativ niedrigen Marktanteil von Macs einher. Ebenfalls nur geringe Marktanteile weist der Opera auf (1,3 %).

OnStat.com und W3B versuchen jeweils, einen repräsentativen Querschnitt zu ermitteln. Bei Special-Interest-Seiten sehen die Marktanteile natürlich anders aus. Wenn Sie beispielsweise ein Info-Portal für Linux-Nutzer betrachten, wird dort der Konqueror-Anteil höher und der Anteil des Internet Explorer niedriger sein; auf der Microsoft-Homepage sieht das sicher ganz anders aus. Die Zahlen von OneStat.com und W3B dagegen entsprechen wohl dem tatsächlichen Querschnitt.

Einen Makel haben diese Erhebungen trotzdem. Viele (schlechte) JavaScript-Programmierer setzen für ihre Websites den Internet Explorer voraus und ignorieren »kleinere« Browser. Aus diesem Grund bieten beispielsweise Konqueror und Opera an, dass der Browser der Website gegenüber als Internet Explorer ausgegeben wird. Teilweise ist es sogar möglich, die Browser-Identifikation (also wie der Browser heißt und welche Versionsnummer er hat) selbst, d.h. als Benutzer, anzugeben. Wenn dies nicht erkannt und abgefangen wird, verfälscht eine solche Angabe natürlich die Statistik zugunsten des Internet Explorer. An der Führungsposition des Microsoft-Browsers besteht natürlich trotzdem kein Zweifel.

³ Quelle: http://www.onestat.com/html/aboutus_pressbox34.html

2.1.7 Testsystem

Um es noch einmal zu betonen: Je mehr Browser unterstützt werden, desto weniger Nutzer suchen enttäuscht oder verärgert das Weite. Im Verlauf dieses Buchs werden Sie sehen, dass es gar nicht so schwer ist, eine solche Unterstützung umzusetzen. Voraussetzung dafür ist jedoch, dass Sie Zugriff auf entsprechende Testsysteme haben.

Windows ist natürlich Pflicht. Dort können zumindest die Netscape-Versionen 4, 6, 7 parallel installiert werden, zusätzlich noch der jeweils aktuelle Mozilla (er ist meist dem Netscape mindestens um eine Nasenlänge voraus). Die Netscape-Versionen 1 bis 3 sind ebenfalls noch erhältlich (z.B. im Browser-Archiv von evolt.org unter <http://browsers.evolt.org/?navigator/>), allerdings ist ihr Marktanteil so stark gesunken, dass es mittlerweile nicht mehr notwendig ist, sie zu unterstützen. Netscape 4 allerdings ist – wie zuvor gesehen – mit der wichtigste Testbrowser neben dem Internet Explorer; aus Gründen der Stabilität und zum Schutz Ihrer Nerven sollten Sie Version 4.8 verwenden.

Der Internet Explorer ermöglicht keine parallele Installation diverser Versionen. Außerdem werden beispielsweise für den Internet Explorer 5 keine Updates mehr angeboten (außer, Sie setzen Windows 2000 ein), und der Support (einschließlich Updates) für Internet Explorer 5.5 wurde Ende 2003 eingestellt (es gibt seitdem nur noch kritische Sicherheitspatches). Dennoch zeigen die Statistiken deutlich, dass die Versionen 5 und 5.5 noch eine relevante Marktposition haben. Deswegen sollten Sie hierfür ein extra Testsystem (oder eine zweite Windows-Partition) einrichten. Bedenken Sie aber, dass ein »Downgrade« nicht möglich ist. Windows XP beispielsweise wird mit dem Internet Explorer 6 ausgeliefert, hier können Sie keine Vorgängerversion installieren.

Wenn Sie den Internet Explorer 4 installiert haben, können Sie die Version 5 oder 5.5 parallel installieren. Dazu müssen Sie bei den Installationsoptionen angeben, dass Sie den IE4 weiterhin im Kompatibilitätsmodus ausführen möchten. Ab dem Internet Explorer 6 ist dieser Modus nicht mehr möglich. Wenn Sie die Möglichkeit haben und ohnehin ein Testsystem für IE5 oder IE5.5 einrichten, sollten Sie sich aber diesen minimalen Aufwand leisten und Ihre JavaScript-Programme auch auf dem alten Internet Explorer 4 testen. Der uralte Internet Explorer 3 kann übrigens auch parallel installiert werden – in der 16-Bit-Version (Windows 3.1x). Dieser Aufwand ist allerdings heutzutage wirklich nicht mehr notwendig.

Auf einem Macintosh-System ist natürlich der IE Pflicht, ebenfalls die Mac-Versionen von Netscape und Mozilla und neuerdings auch der Safari. Zusätzlich ist noch ein Linux-System empfehlenswert, auf dem neben den Netscape-Derivaten vor allem der Konqueror Beachtung verdient. Sie benötigen hierzu aber unbedingt den KDE als Fenstermanager.

Sie sehen also: Es ist eine Menge Aufwand erforderlich, aber es lohnt sich, um einen möglichst großen Besucherstamm bedienen zu können. Auch bei der Erstellung dieses Buches wurde auf ein umfangreiches Testsystem zurückgegriffen. Sie erhalten also eine Reihe von Tipps, wie Sie Kompatibilitätsprobleme umschiffen können.

2.2 Verwendung von `<script>`

Nun kommen wir endlich zur Erstellung von JavaScript. Wie ich bereits erwähnt habe, wird JavaScript in HTML integriert; Sie arbeiten also hauptsächlich mit HTML-Dateien, die Sie in einem einfachen Texteditor erstellen können.

JavaScript-Kommandos können an mehreren Stellen einer HTML-Datei untergebracht werden:

- ▶ zwischen den Tags `<script>` und `</script>`
- ▶ in einer externen Datei
- ▶ in Form eines HTML-Links
- ▶ als Parameter von HTML-Tags

In den folgenden Abschnitten werden die einzelnen Möglichkeiten der Reihe nach vorgestellt und erläutert.

Als Beispiel hierzu dient die Anweisung `document.write("The weather means the seasons")`, die den Text "The weather means the seasons" ausgibt. Warum dieser Befehl so funktioniert, erfahren Sie in den nächsten Kapiteln; fürs Erste müssen Sie mir einfach vertrauen.

Die naheliegendste Methode, JavaScript-Befehle auszuführen, besteht darin, das⁴ `<script>`-Tag zu verwenden. Folgender Code sorgt dafür, dass »The weather means the seasons« ausgegeben wird:

```
<script>
document.write("The weather means the seasons");
</script>
```

Befehle werden in JavaScript untereinander – einer pro Zeile – dargestellt. Wenn Sie mehrere Kommandos in einer Zeile unterbringen wollen, müssen Sie die Anweisungen durch ein Semikolon voneinander trennen. Im Gegensatz zu anderen Programmiersprachen (beispielsweise Java) muss aber keineswegs jedes Kommando mit einem Strichpunkt enden. In der ersten Version der JavaScript-Sprachspezifikation war das Semikolon am Ende jeder Anweisung strikt vorgeschrieben. Inzwischen wurde aber davon Abstand genommen, und jeder

4 Der Tag oder das Tag? Manche Autoren bevorzugen »das Tag«, um eine eindeutige Abgrenzung vom (Wochen-)Tag zu gewährleisten; andere bevorzugen »der Tag«. Ich habe mich auf die Seite der Mehrheit geschlagen: »das Tag«.

JavaScript-Programmierer hat seinen eigenen Stil. Prinzipiell haben die Strichpunkte den Sinn, dem JavaScript-Interpreter (also dem Bestandteil des Browsers, der den JavaScript-Code ausführt) mitzuteilen, an welcher Stelle eine Anweisung endet. Es gibt auch Programmiersprachen, bei denen das Zeilenende das Ende einer Anweisung markiert. In JavaScript ist beides möglich. Um den Code sauber zu halten und um bei Programmierfehlern schneller die Fehlerquelle zu finden, verzichte ich **nicht** auf optionale Strichpunkte. Es ist Ihnen aber natürlich freigestellt, sich einen anderen Stil anzueignen – insbesondere, wenn Sie bereits Erfahrungen in einer Programmiersprache gesammelt haben, in der keine Strichpunkte vorkommen (z.B. Visual Basic/VBScript/VB.NET).

Die beiden folgenden Anweisungen sind also äquivalent. Einmal stehen die beiden Befehle in verschiedenen Zeilen, einmal in einer Zeile.

```
<script>
document.write("The weather ");
document.write("means the seasons");
</script>
```

und

```
<script>
document.write("The weather "); document.write("means the
    seasons");
</script>
```

JavaScript-Code wird hierbei vom JavaScript-Interpreter des verwendeten Browsers ausgeführt. Betrachten Sie zum Beispiel folgendes HTML-Dokument:

```
<html>
<head>
<title>JavaScript</title>
</head>
<body>
<script>
document.write("The weather means the seasons");
</script>
</body>
</html>
```

Wenn es vom Browser interpretiert worden ist und dieser Browser JavaScript unterstützt, verhält es sich so wie folgendes HTML-Dokument:

```
<html>
<head>
<title>JavaScript</title>
</head>
```

```
<body>
The weather means the seasons
</body>
</html>
```

Nehmen Sie es mir bitte nicht übel, wenn die ersten Beispiele in diesem Kapitel nicht unbedingt die breite Funktionspalette von JavaScript demonstrieren.

2.2.1 Das language-Attribut

Das obige Beispiel ist streng genommen etwas unsauber. Das `<script>`-Tag eignet sich auch für andere Programmiersprachen, die in HTML-Dokumente eingebettet werden können, beispielsweise für Visual Basic Script (VBScript) oder JScript. Dazu dient das Attribut `language` des `<script>`-Tags. Ist es nicht gesetzt – wie im obigen Beispiel –, so wird angenommen, dass die zwischen den Tags stehenden Kommandos in JavaScript verfasst wurden (deswegen funktioniert obiges Beispiel auch). Aber um auf Nummer Sicher zu gehen – es könnte ja sein, dass eine neue Version des Microsoft Internet Explorer als Standardsprache VBScript annimmt –, schreiben wir:

```
<html>
<head>
<title>JavaScript</title>
</head>
<body>
<script language="JavaScript">
document.write("The weather means the seasons");
</script>
</body>
</html>
```

XHTML schreibt vor, dass das `type`-Attribut gesetzt werden muss, in unserem Fall auf `"text/javascript"`. Wir haben davon vorerst Abstand genommen, da ältere Browser nur `language` auslesen und neuere damit auch klarkommen.



Wie Sie in Kapitel 1 erfahren haben, gibt es mehrere Versionen von JavaScript. Sie können im `language`-Attribut auch explizit eine der Versionen angeben. Der folgende Code wird nur von Browsern ausgeführt, die JavaScript Version 1.1 unterstützen (das sind insbesondere der Netscape Navigator ab Version 3 und der Microsoft Internet Explorer ab Version 4):

```
<html>
<head>
<title>JavaScript</title>
</head>
<body>
```

```

<script language="JavaScript1.1">
document.write("The weather means the seasons");
</script>
</body>
</html>

```

Ältere Browser, beispielsweise der Internet Explorer 3, ignorieren den JavaScript-Befehl und geben nichts aus.

Zur Zeit der Drucklegung (März 2005) sind die in Tabelle 2.3 aufgeführten Parameter gültig.

Parameter	Bedeutung
JavaScript	Jeder Browser, der JavaScript unterstützt
JavaScript1.1	Alle Browser, die mindestens die JavaScript-Version 1.1 unterstützen (ab NN3, IE4)
JavaScript1.2	Alle Browser, die mindestens die JavaScript-Version 1.2 unterstützen (ab NN4, IE5)
JavaScript1.3	Alle Browser, die mindestens die JavaScript-Version 1.3 unterstützen (ab Netscape Navigator 4.06, IE5)
JavaScript1.4	Ab Netscape 6
JavaScript1.5	Ab Netscape 6

Tabelle 2.3 Die Parameter für <script language="...">

Mit der folgenden HTML-Seite können Sie überprüfen, welche JavaScript-Versionen der jeweilige Browser unterstützt:

```

<html>
<head>
<title>JavaScript</title>
</head>
<body>
<script language="JavaScript">
document.write("Der Browser unterstützt JavaScript
<hr />");
</script>
<script language="JavaScript1.1">
document.write("Der Browser unterstützt JavaScript v1.1<hr />");
</script>
<script language="JavaScript1.2">
document.write("Der Browser unterstützt JavaScript v1.2<hr />");
</script>

```

```
<script language="JavaScript1.3">
document.write("Der Browser unterstützt JavaScript v1.3<hr />");
</script>
<script language="JavaScript1.4">
document.write("Der Browser unterstützt JavaScript v1.4<hr />");
</script>
<script language="JavaScript1.5">
document.write("Der Browser unterstützt JavaScript v1.5<hr />");
</script>
</body>
</html>
```

In Abbildung 2.12 sehen Sie beispielsweise, was der Netscape Navigator 4.78 ausgibt.

Beachten Sie bei obigem Beispiel, dass man mit `document.write()` insbesondere auch HTML-Code, in diesem Fall das `<hr>`-Tag für eine horizontale Linie, ausgeben kann.

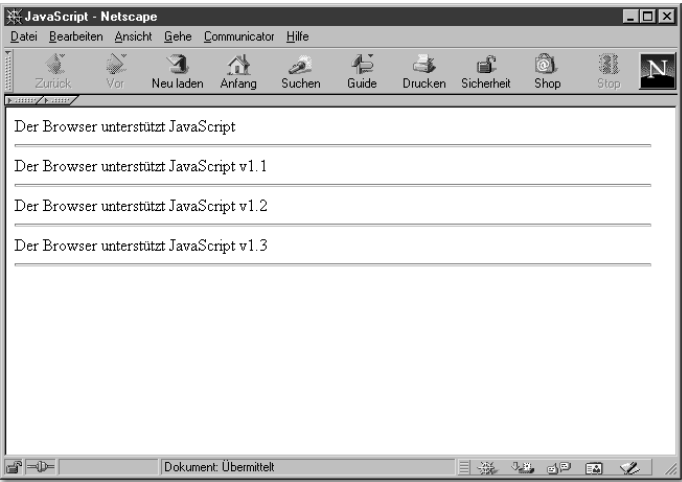


Abbildung 2.12 Die vom Netscape Navigator 4.78 unterstützten JavaScript-Versionen

In Tabelle 2.4 sehen Sie, welche Parameter ausgewählte Browserversionen erkennen bzw. unterstützen (Browserverfügbarkeit Stand März 2005):

Browser	Unterstützte Parameter
Netscape 4.00–4.05	JavaScript1.0 bis JavaScript1.2
Netscape 4.06–4.8	JavaScript1.0 bis JavaScript1.3

Tabelle 2.4 Die von ausgewählten Browsern unterstützten Parameter `<script language="...">`

Browser	Unterstützte Parameter
Netscape 6/7, Mozilla, Firefox	JavaScript1.0 bis JavaScript1.5
Internet Explorer 4	JavaScript1.0 bis JavaScript1.2
Internet Explorer 5	JavaScript1.0 bis JavaScript1.3
Internet Explorer 5.5	JavaScript1.0 bis JavaScript1.3
Internet Explorer 6	JavaScript1.0 bis JavaScript1.3
Opera 5.x/6/7	JavaScript1.0 bis JavaScript1.4
Konqueror/Safari	JavaScript1.0 bis JavaScript1.4

Tabelle 2.4 Die von ausgewählten Browsern unterstützten Parameter
`<script language="...">` (Forts.)

Durch die Verwendung spezieller **language**-Attribute können Sie Fehlermeldungen vermeiden, die bei der Verwendung von zu modernen JavaScript-Kommandos erscheinen würden. Wenn Sie also Sprachelemente von JavaScript verwenden, die erst ab Version 1.1 unterstützt werden, sollten Sie das `language`-Attribut auf "JavaScript1.1" setzen; ältere Browser ignorieren dann die Befehle völlig.

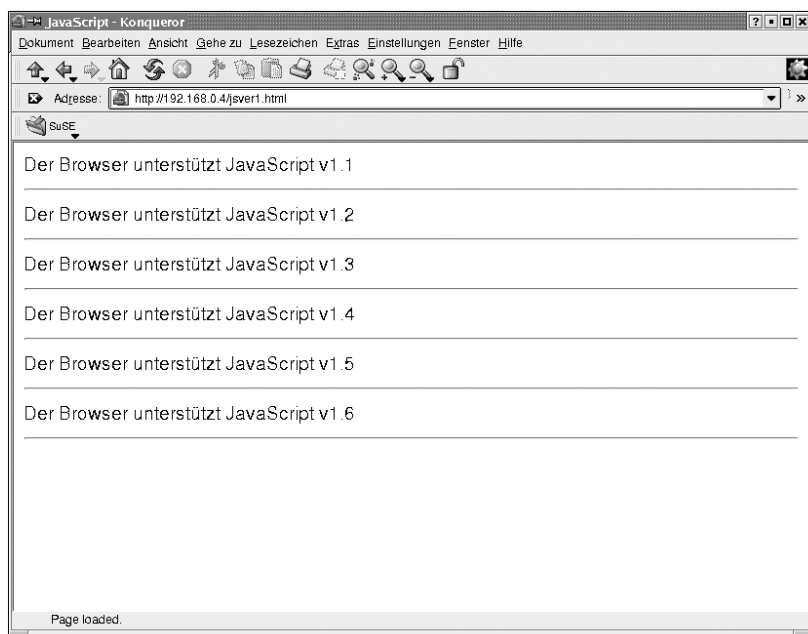


Abbildung 2.13 Konqueror kennt (angeblich) schon JavaScript 1.6!

Das stimmt leider nicht immer. Einige Versionen des Netscape Navigator 3 beispielsweise haben einen Bug und versuchen, auch JavaScript-Blöcke mit `language="JavaScript1.2"` und `language="JavaScript1.3"` zu interpretieren. Auch ältere Versionen des Internet Explorer führen manchmal Code aus, der nicht für sie bestimmt ist. Besonders »schlimm« ist in diesem Zusammenhang jedoch Konqueror und damit auch Safari. Es scheint zu genügen, dass der Wert des `language`-Attributs mit "JavaScript" beginnt. Fügt man beispielsweise in obiges Listing noch einen Block mit `<script language="JavaScript1.6">` ein, würde auch dieser ausgeführt werden, obwohl es diese Sprachversion nicht gibt (siehe Abbildung 2.13).



2.2.2 Browser ohne JavaScript

So schön ein Programm auch auf dem eigenen Rechner laufen mag – es kommt darauf an, dass es beim Kunden und bei allen Besuchern (nun gut, sagen wir, bei den meisten Besuchern) der Website läuft. In vielen Firmennetzwerken ist es beispielsweise so, dass JavaScript aus Sicherheitsgründen nicht aktiviert werden darf. Die Beispiele von oben sehen dann so aus wie in Abbildung 2.14.

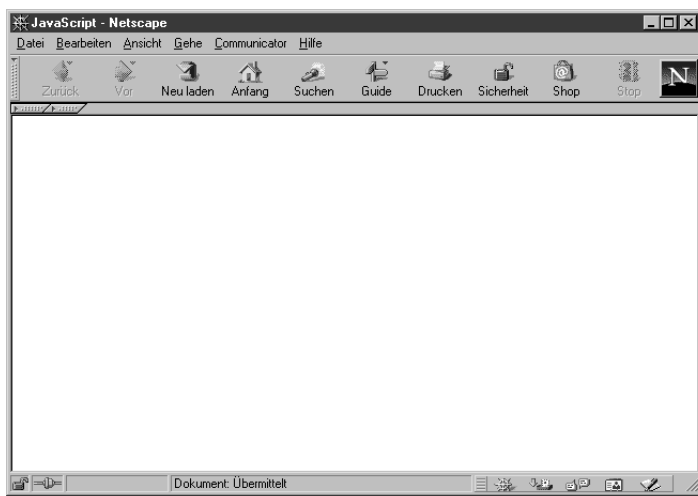


Abbildung 2.14 Die Ausgabe bei deaktiviertem JavaScript: Gähnende Leere

Bei ganz alten Browsern ist es noch schlimmer, zuweilen sieht man sogar den JavaScript-Code. Sie können sich aber hier mit einem kleinen Trick behelfen: Mit `<!--` wird ein HTML-Kommentar eingeleitet; alles dahinter wird vom HTML-Interpreter ignoriert, jedoch nicht vom JavaScript-Interpreter! Es empfiehlt sich also, JavaScript-Code durch folgenden Befehl einzuleiten:

```
<script language="JavaScript"><!--
```

Nun stellt sich die Frage, wie der HTML-Kommentar beendet wird. Probieren Sie einmal die einfachste Variante aus, nämlich einfach ein Kommentar-Ende-Tag (`-->`) vor dem `</script>`:

```
<html>
<head>
<title>JavaScript</title>
</head>
<body>
<script language="JavaScript"><!--
document.write("The weather means the seasons");
--></script>
</body>
</html>
```

Wenn Sie diese Seite im Browser laden, sollten Sie eine Fehlermeldung erhalten. Der Grund: Der JavaScript-Interpreter interpretiert `-->` als JavaScript-Befehl und liefert eine Fehlermeldung.

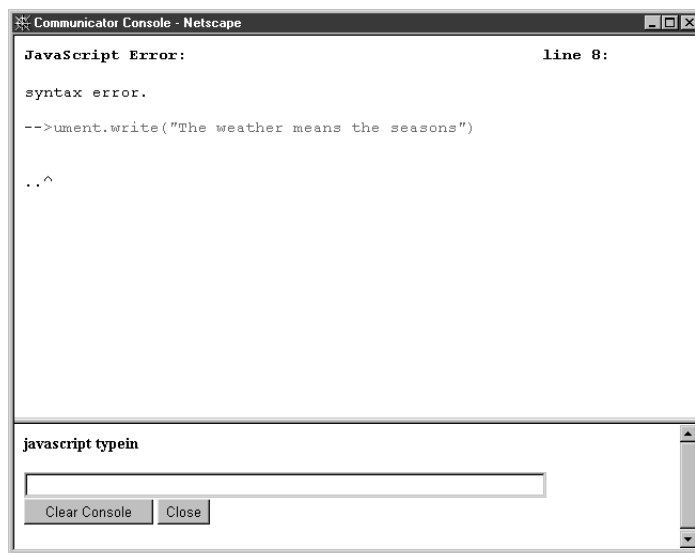


Abbildung 2.15 Fehlermeldung des Netscape Navigator

Aber auch hier gibt es einen kleinen Trick, mit dem Sie dieses Hindernis aus dem Weg räumen können. Mit `//` leitet man einen Kommentar im JavaScript-Code ein. Die Verwendung von Kommentaren ist bei der Programmierung sehr wichtig, damit man auch Wochen später noch weiß, was man damals eigentlich beabsichtigt hat, und damit auch andere Menschen mit dem Code arbeiten können. Es gibt zwei Arten von Kommentaren:

- ▶ `//`: Hiermit wird ein einzeliger Kommentar eingeleitet; alles hinter den beiden Querstrichen in der jeweiligen Zeile wird vom JavaScript-Interpreter ignoriert.
- ▶ `/*` und `*/`: Hiermit wird ein mehrzeiliger Kommentar eingeleitet; alles nach `/*` wird vom JavaScript-Interpreter ignoriert, bis die Zeichenfolge `*/` kommt und den Kommentar abschließt.

Im folgenden Listing sehen Sie Beispiele für Kommentare:

```
<html>
<head>
<title>JavaScript</title>
</head>
<body>
<script language="JavaScript">
// Hier wird auf generelle JavaScript-Unterstützung geprüft
document.write("Der Browser unterstützt
JavaScript<hr>");
</script>
<script language="JavaScript1.1">
/* Hier geht es um JavaScript 1.1 */
document.write("Der Browser unterstützt JavaScript v1.1<hr />");
</script>
<script language="JavaScript1.2">
// JavaScript 1.2 wird überprüft,
// und zwar gründlich
document.write("Der Browser unterstützt JavaScript v1.2<hr />");
</script>
<script language="JavaScript1.3">
/* JavaScript 1.3 wurde
   mit dem Netscape Navigator 4.06 eingeführt */
document.write("Der Browser unterstützt JavaScript v1.3<hr />");
</script>
<script language="JavaScript1.4">
document.write("Der Browser unterstützt JavaScript v1.4<hr />");
</script>
<script language="JavaScript1.5">
document.write("Der Browser unterstützt JavaScript v1.5<hr />");
</script>
</body>
</html>
```

Kommen wir zum ursprünglichen Problem zurück. Der Browser gibt eine Fehlermeldung aus, weil `-->` als JavaScript-Code interpretiert wird und nicht als HTML-Element. Wenn dem `-->` jedoch ein `//` vorangestellt wird, ignoriert der

JavaScript-Interpreter diesen Code; der HTML-Interpreter jedoch stellt fest, dass der Kommentar zu Ende ist. Der folgende Code wird von Browsern, die JavaScript unterstützen, ausgeführt. Browser, die kein JavaScript unterstützen oder bei denen JavaScript deaktiviert ist, sehen einen HTML-Kommentar, ignorieren das Innere und geben folglich nichts aus, auch keinen reinen JavaScript-Code.

```
<html>
<head>
<title>JavaScript</title>
</head>
<body>
<script language="JavaScript"><!--
document.write("The weather means the seasons");
//--></script>
</body>
</html>
```

Browser und Textausgabe

Sie sehen an den obigen Beispielen, dass es relativ einfach ist, bei Browsern, die JavaScript unterstützen, einen Text auszugeben. Der andere Weg ist aber auch möglich. Es gibt hierfür (ab Netscape Navigator 3 bzw. Internet Explorer 3) ein besonderes HTML-Element, `<noscript>`, das so ähnlich wie `<noframes>` funktioniert. Damit sind folgende Szenarien denkbar:

- ▶ Der Browser unterstützt kein JavaScript, egal, ob das `<noscript>`-Tag bekannt ist oder nicht. Es wird notfalls ignoriert, und die darauf folgenden HTML-Elemente werden interpretiert (bzw. der Inhalt wird angezeigt).
- ▶ Der Browser unterstützt JavaScript, und es ist auch eingeschaltet. Dann wird alles, was zwischen `<noscript>` und `</noscript>` steht, nicht dargestellt.
- ▶ Der Browser unterstützt JavaScript, es ist jedoch ausgeschaltet. Dann wird auch all das dargestellt, was zwischen `<noscript>` und `</noscript>` steht.

```
<html>
<head>
<title>JavaScript</title>
</head>
<body>
<script language="JavaScript"><!--
document.write("The weather means the seasons");
//--></script>
<noscript>
Ihr Browser kann mit JavaScript nichts anfangen, oder es ist aus-
geschaltet!
</noscript>
</body>
</html>
```

Denken Sie immer auch an diejenigen Besucher, die JavaScript deaktiviert haben oder deren Browser (man denke nur an die eingeschränkten Browser von Handhelds) kein JavaScript unterstützt. Erstellen Sie notfalls eine Version Ihrer Website, die auch ohne JavaScript funktioniert.



Abbildung 2.14 zeigt, dass das tatsächlich funktioniert: Sie sehen obiges Dokument im klassischen Text-Webbrowser Lynx.

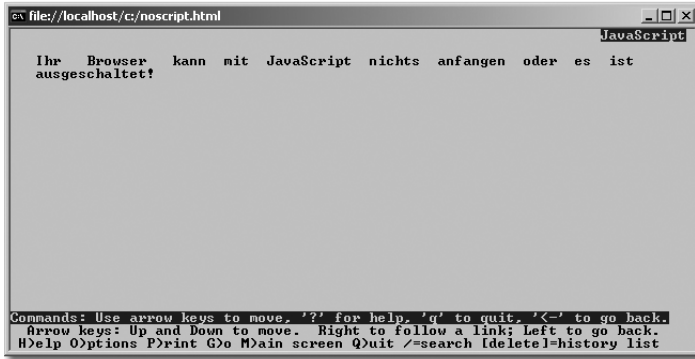


Abbildung 2.16 Kein JavaScript beim Text-Browser Lynx

2.2.3 Externe Dateien

Wenn Sie später einmal JavaScript-Programme schreiben, die auf mehreren unterschiedlichen Seiten benötigt werden, wäre es eigentlich ziemlich töricht, dasselbe Skript in mehrere Seiten zu kopieren – der Aufwand bei Änderungen am Skript wäre beträchtlich, da mehrere Dateien geöffnet und geändert werden müssten.

Es gibt hier auf den ersten Blick einen Ausweg, der aber auf den zweiten Blick auch seine Nachteile hat. Zuerst zur grauen Theorie: Man kann beim `<script>`-Tag im Attribut `src` den Namen einer externen Datei mit JavaScript-Kommandos angeben. Als Dateierweiterung hat sich hierbei `.js` durchgesetzt. Es empfiehlt sich, für alle externen Dateien ein eigenes Verzeichnis anzulegen, damit diese alle gesammelt an einem zentralen Ort zu finden sind.

Angenommen, folgende Datei ist auf dem Webserver im virtuellen Verzeichnis `js` unter dem Namen `weather.js` gespeichert:

```
//erste externe JavaScript-Datei
document.write("The weather means the seasons");
```

Sie wird folgendermaßen in ein HTML-Dokument eingebunden, um dieselbe Wirkung zu erzielen wie das Dokument aus dem vorigen Beispiel:

```
<html>
<head>
```

```

<title>JavaScript</title>
</head>
<body>
<script language="JavaScript" src="/js/weather.js">
</script>
</body>
</html>

```

Natürlich kann auch hier das `language`-Attribut gesetzt werden. Bei dem folgenden Dokument wird nur etwas ausgegeben, wenn der Browser JavaScript Version 1.1 unterstützt:

```

<html>
<head>
<title>JavaScript</title>
</head>
<body>
<script language="JavaScript1.1" src="/js/weather.js">
</script>
</body>
</html>

```

Dieses Vorgehen birgt aber auch einen kleinen Fallstrick: Probieren Sie doch einmal Folgendes in Ihrem Browser aus:

```

<html>
<body>
<script language="JavaScript" src="/js/weather.js">!--
document.write("<br>Invincibility is in oneself,
vulnerability is in the opponent");
//--></script>
</body>
</html>

```

Das Ergebnis sehen Sie in Abbildung 2.17: Der Inhalt des `<script>`-Elements wird ignoriert. Der Grund: Ist das `src`-Attribut des `<script>`-Tags gesetzt, wird eingeschlossener JavaScript-Code nicht betrachtet; ist `src` nicht gesetzt, so wird der eingeschlossene Code ausgeführt. Um also beide Sätze auszugeben, muss das HTML-Dokument folgendermaßen abgeändert werden:

```

<html>
<body>
<script language="JavaScript" src="/js/weather.js"></script>
<script language="JavaScript">!--
document.write("<br>Invincibility is in oneself, vulnerability
    is in the opponent");
//--></script>

```

```
</body>
</html>
```

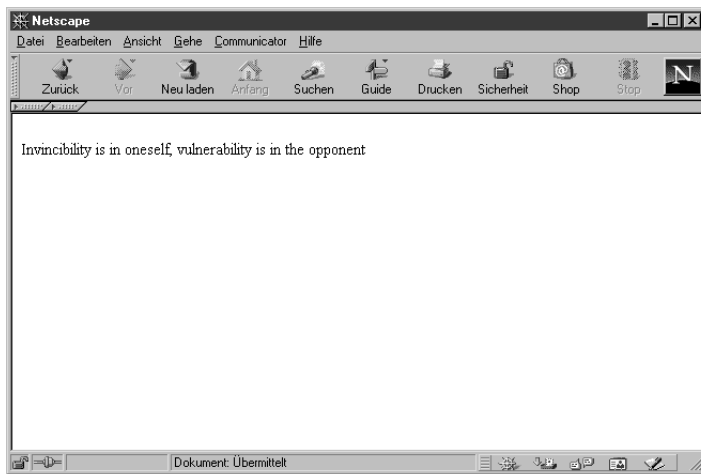


Abbildung 2.17 Der Text wird nur einmal angezeigt.

Obwohl externe Dateien sehr praktisch und auch recht weit verbreitet sind, haben sie einen Nachteil, den viele gar nicht kennen: Der Netscape Navigator 2 kann mit dem `src`-Attribut nichts anfangen, und der Internet Explorer 3 hat immer wieder Probleme damit (und im lokalen Betrieb, also ohne Webserver, funktioniert es überhaupt nicht). Die ersten Versionen des Internet Explorer 3 haben dieses Feature gar nicht unterstützt, während es in den Versionen 3.02 und 3.03 hin und wieder funktioniert. Es hilft manchmal, eine vollständige URL anzugeben (also `src="http://www.ihrefirma.de/skript.js"`), aber auch das funktioniert hin und wieder nicht. Setzen Sie also Ihr Skript für den Internet Explorer 3 in das HTML-Dokument selbst, oder testen Sie es intensiv. Kontaktieren Sie außerdem den Betreiber Ihres Webserver; er muss den Server so konfigurieren, dass Dateien mit der Endung `.js` mit dem MIME-Typ **application/x-javascript** verknüpft werden.



Sollten in diesem Kapitel externe Dateien verwendet werden, so dient das in der Regel der Übersichtlichkeit des Codes. Natürlich muss verhindert werden, dass ältere Browser eine Fehlermeldung ausgeben. Da der Internet Explorer 3 immer seltener verwendet wird (ohnehin ist das nur noch auf Systemen möglich, die unter Windows 95 laufen), verliert auch diese »Gefahr« zunehmend an Brisanz.

2.3 JavaScript-Links

JavaScript-Befehle werden oft aufgrund von Benutzereingaben ausgeführt. Eine Möglichkeit besteht darin, eine Aktion durch einen Mausklick zu starten. Bevor ein paar Beispiele vorgestellt werden, muss noch ein neuer JavaScript-Befehl

eingeführt werden. Mit `window.alert("Invincibility is in oneself, vulnerability is in the opponent")` wird ein modales Fenster ausgegeben, das den Text »Invincibility is in oneself, vulnerability is in the opponent« anzeigt. Je nach verwendetem Browser sieht das etwas anders aus; in Abbildung 2.18 sehen Sie die Darstellung im Netscape Navigator 4 und in Abbildung 2.19 im Safari-Browser. Sie sehen daran, dass Sie auf das grafische Layout keinen Einfluss nehmen können, denn das wird vom jeweiligen Browser übernommen.

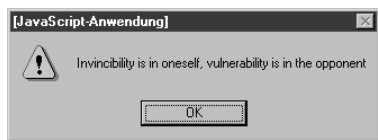


Abbildung 2.18 Ein Warnfenster mit dem Netscape Navigator 4

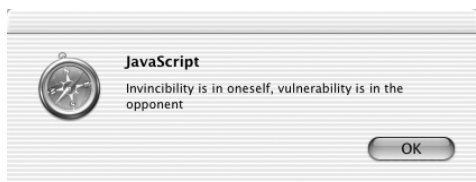


Abbildung 2.19 Dieselbe Meldung, diesmal im Mac-Browser Safari

Protokolle HTML-Links können auf URLs mit den verschiedensten Protokollen verweisen, so zum Beispiel **http:**, **ftp:**, **news:** oder **mailto:**. Nun kommt ein weiteres Protokoll hinzu, das jedoch lediglich im Zusammenhang mit JavaScript eine Bedeutung hat: **javascript:**. Folgender Link gibt »Invincibility is in oneself, vulnerability is in the opponent« in einem modalen Fenster aus, wenn man darauf klickt:

```
<A HREF="javascript:window.alert('Invincibility is in oneself,
vulnerability is in the opponent');">
```

Beachten Sie die Anführungszeichen. In JavaScript ist es prinzipiell egal, ob Sie einfache oder doppelte Anführungszeichen verwenden – Hauptsache, Sie hören so auf, wie Sie angefangen haben.

Folgendes ist also völlig korrekt:

```
document.write("The weather means the seasons");
document.write('<br />Invincibility is in oneself,
vulnerability is in the opponent');
```

Falsch ist dagegen:

```
document.write("The weather means the seasons ' ');
document.write('<br />Invincibility is in oneself,
vulnerability is in the opponent");
```

Obwohl man mit dem Pseudo-Protokoll **javascript:** bequem JavaScript-Befehle aufrufen kann, gibt es doch einen kleinen Nachteil: Browser, die kein JavaScript unterstützen, geben eine Fehlermeldung aus. Im folgenden Abschnitt wird diesem Missstand abgeholfen.

2.4 Event-Handler

Ohne anderen Kapiteln allzu sehr vorgreifen zu wollen – wenn Sie bestimmte Aktionen ausführen, beispielsweise die Maus bewegen oder auf einen Link klicken, tritt JavaScript-intern ein so genanntes **Ereignis** (engl. **event**) ein. Manche dieser Ereignisse können Sie mit JavaScript abfangen und darauf reagieren. Dazu benötigen Sie **Event-Handler**, die als Attribute mancher HTML-Tags in den HTML-Code eingebunden werden. Eine vollständige Auflistung aller Event-Handler finden Sie in der Referenz; davor werden die wichtigsten Handler in den jeweiligen Kapiteln anhand eines Beispiels vorgestellt.

Grundsätzlich gilt Folgendes: Ein Event-Handler beginnt stets mit `on`. Der Event-Handler, der zuständig ist, wenn auf einen Link geklickt wird, heißt `onclick`. Wenn folgender Link angeklickt wird, wird ein modales Fenster geöffnet:

```
<a href="irgendwohin.html" onclick="alert('Invincibility is in oneself, vulnerability is in the opponent');">Sun Tzu</a>
```

Vom Prinzip her wunderbar – Browser, die JavaScript unterstützen, führen den Code aus, folgen danach aber dem Link (in späteren Kapiteln dazu mehr). Ältere Browser »sehen« nur den Link und versuchen, das Ziel des Links aufzurufen. Das ist aber in der Regel nicht erwünscht. Das `href`-Attribut des Links muss gesetzt werden, sonst wird er nicht angezeigt. Es gibt aber eine Möglichkeit, einen Link anzugeben, der keine neue Seite lädt:

```
<a href="#" onclick="alert('Invincibility is in oneself, vulnerability is in the opponent');">Sun Tzu</a>
```

Diesen Trick sollten Sie sich gut merken!



2.5 JavaScript-Entities

Die letzte Möglichkeit, um JavaScript-Code in das HTML-Dokument einzubetten, wird sehr selten verwendet, was unter anderem auch daran liegt, dass hier nur Werte von HTML-Attributen gesetzt werden können – und dass es nur mit dem Netscape Navigator funktioniert. Sie kennen bereits HTML-Entities, die mit `&` beginnen und mit `;` enden. Ein Beispiel ist die HTML-Entity für Ä, `Ä`. JavaScript-Entities sind ganz ähnlich aufgebaut: Sie beginnen mit `&`, enden mit `;`, und dazwischen steht ein JavaScript-Ausdruck. Für dieses Beispiel wird ein weiteres neues JavaScript-Kommando eingeführt (genauer: eine Objekteigenschaft, aber dazu später mehr): `location.protocol` gibt das ver-

wendete Protokoll der aktuellen Seite vor, also beispielsweise `file:` bei lokalen Dateien und `http:` bei Dateien aus dem World Wide Web. In dem folgenden Dokument wird das `value`-Attribut eines Texteingabefelds auf den Wert von `location.protocol` gesetzt. Die JavaScript-Entity heißt `&{location.protocol}`; – beachten Sie insbesondere, dass der JavaScript-Ausdruck in geschweifte Klammern eingeschlossen werden muss!

```
<html>
<head>
<title>JavaScript-Entities</title>
</head>
<body>
<form>
Protokoll: <input type="text"
value="&{location.protocol};">
</form>
</body>
</html>
```



Abbildung 2.20 HTML-Attribute setzen mit Entities

3 Programmierung I

*Es war einmal vor langer Zeit,
und diese Zeit ist schon lange, lange her,
etwa letzten Freitag,
als Winnie-der-Pu ganz allein
unter dem Namen Sanders in einem Wald wohnte.
– A. A. Milne, Pu der Bär, deutsche Übersetzung von Harry Rowohlt*

In diesem und dem folgenden Kapitel werden die Grundprinzipien der Programmierung an sich und der Programmierung mit JavaScript im Besonderen erläutert. An Praxisbeispielen ist dieses Kapitel recht arm, aber Sie werden die hier vorgestellten Techniken in den weiteren Kapiteln noch des Öfteren benötigen. Sie erhalten hier das Rüstzeug, das die Grundlage für alle kommenden Kapitel ist.

Gut gerüstet

Grundbegriffe der Programmierung werden hier ebenfalls erläutert. Leser, die bereits Erfahrungen mit der einen oder anderen Programmiersprache haben, mögen diesen kleinen Exkurs verzeihen, aber so soll auch Neulingen die Chance geboten werden, in die Materie einzusteigen. Allerdings sind die Erklärungen recht knapp gehalten, und nur das Nötigste wird hierzu erläutert. Schließlich geht es ja darum, möglichst schnell brauchbare Anwendungen zu schreiben.

3.1 Variablen

Bei der Programmierung müssen immer wieder Daten zwischengespeichert werden. Hierzu bedient man sich so genannter **Variablen** (in manchen Büchern – aber sehr wenigen – werden sie auch als »Veränderliche« bezeichnet).

3.1.1 Namensgebung

Jede Variable wird anhand ihres Namens angesprochen. Bei der Namensgebung haben Sie größtenteils freie Hand. Ein Variablenname besteht aus einer Folge von Buchstaben, Ziffern und dem Unterstrich (`_`). Das erste Zeichen darf jedoch keine Ziffer sein. Außerdem wird zwischen Groß- und Kleinschreibung unterschieden. Die Bezeichner `Pooh`, `pooh` und `POOH` sind also verschiedene Variablen. Beispiele für Variablennamen sind etwa:

- ▶ `Winnie_The_Pooh`
- ▶ `WinnieThePooh`
- ▶ `Winnie2002`
- ▶ `_Winnie_`
- ▶ `Honigtopf`

Unbrauchbar sind dagegen die folgenden Namen:

- ▶ `1Winnie` (beginnt mit einer Ziffer)
- ▶ `Winnie Pooh` (Leerzeichen)
- ▶ `Winnie-Pooh` (Bindestrich)
- ▶ `Honigtöpfe` (Umlaut)

JavaScript-Schlüsselwörter und -begriffe dürfen Sie nicht als Variablennamen verwenden. Ein Beispiel hierfür ist etwa `alert`, das Sie im vorigen Kapitel schon einmal in Aktion gesehen haben.

Um einer Variablen einen Wert zuzuweisen, wird das Gleichheitszeichen verwendet. Links vom Gleichheitszeichen steht der Variablenname, rechts davon der neue Wert (oder eine andere Variable, deren Wert dann zugewiesen wird).

3.1.2 Numerische Variablen

Es gibt verschiedene Typen von Variablen. Zahlenwerte werden wie im amerikanischen Zahlensystem mit einem Dezimalpunkt statt einem Dezimalkomma angegeben:

```
Pi = 3.14159265;  
Mauerfall = 1989;  
MinusHundert = -100;
```

3.1.3 Zeichenketten

Sehr oft werden Zeichenketten, auch **Strings** genannt, verwendet. Die zweite Bezeichnung kommt daher, dass es ein so benanntes Objekt in JavaScript gibt, aber dazu später mehr. In diesem Buch werden beide Begriffe äquivalent verwendet.

Ein String wird von Anführungszeichen eingeschlossen, entweder von einfachen (Apostrophen) oder doppelten. Hierbei ist zu beachten, dass unbedingt gerade Anführungszeichen (`([Umschalt]+[2])`) und Apostrophe (`([Umschalt]+[#])`) verwendet werden. Im Gegensatz zu beispielsweise Perl oder PHP ist es hier egal, ob einfache oder doppelte Anführungszeichen verwendet werden, Hauptsache, die Zeichenkette wird mit derselben Art von Anführungszeichen beendet, wie sie eingeleitet worden ist.

```
WinniesFreund = "Piglet";  
WinniesAndererFreund = 'Tigger';
```

Die verschiedenen Anführungszeichen haben unter anderem den folgenden Sinn: Wenn Sie beispielsweise ein Apostroph in einer Zeichenkette verwenden wollen, können Sie diese Zeichenkette ja schlecht mit Apostrophen eingrenzen,

da der JavaScript-Interpreter dann nicht weiß, wo die Zeichenkette aufhört. In diesem Fall müssen Sie die andere Sorte von Anführungszeichen verwenden:

```
WinnieSagt = "It's me, Winnie the Pooh";
```

Wenn man aber beispielsweise beide Arten von Anführungszeichen in einer Zeichenkette verwenden muss, kommt man in Schwierigkeiten. Hier hilft der Backslash (\) weiter. Das Zeichen, das dem Backslash folgt, wird »entwertet«, d.h., es nimmt in der Zeichenkette keine besondere Bedeutung ein. Beim Anführungszeichen oder Apostroph bedeutet das: Die Zeichenkette wird hiermit nicht beendet.

```
WinnieSagt = 'It\'s me, Winnie the Pooh';
```

Wenn man nun den Backslash selbst in der Zeichenkette verwenden will, muss man auch ihn entwerten:

```
WindowsVerzeichnis = "C:\\WINDOWS";
```

Mit dem Backslash können auch einige besondere Zeichen dargestellt werden. Tabelle 3.1 zeigt eine Übersicht:

Ausdruck	Bedeutung
\r	Wagenrücklauf
\n	Neue Zeile
\t	Tabulator
\b	Backspace (Löschtaste)
\f	Seitenvorschub

Tabelle 3.1 Sonderzeichen innerhalb von Zeichenketten

3.1.4 Boolesche Variablen

Oft ist man nicht an einem Zahlenwert oder einem String interessiert, sondern an einem Wahrheitswert, also wahr oder falsch. Man spricht hier auch von booleschen Variablen. Solche Variablen können als Wert nur `true` (wahr) oder `false` (falsch) annehmen. Später werden Sie sehen, dass boolesche Werte indirekt auch bei einigen anderen JavaScript-Konstrukten vorkommen.

```
Ist2000einSchaltjahr = true;
Ist2004einSchaltjahr = true;
Ist3000einSchaltjahr = false;
```

3.1.5 Variablendeklaration

Wie Sie bereits gesehen haben, kann mit dem Gleichheitszeichen einer Variablen ein Wert zugewiesen werden. Ein Variablenname kann auch öfter ver-

wendet werden, und ihm kann auch mehrmals ein Wert zugewiesen werden. Es handelt sich dann jedoch immer um dieselbe Variable. Sehr oft findet man in der Praxis das Schlüsselwort `var` vor der ersten Verwendung einer Variablen. Dies dient zur Initialisierung der Variablen und wird ab hier der Übersichtlichkeit halber in diesem Buch konsistent verwendet. So sieht man auf den ersten Blick, welche Variable schon einmal deklariert worden ist und welche nicht. Später, bei der Einführung von Funktionen, wird das Schlüsselwort noch eine besondere Bedeutung erhalten.

Beachten Sie, dass Sie das Schlüsselwort `var` nur einmal pro Variable verwenden sollten. Richtig ist also Folgendes:

```
var AnzahlHonigtoepfe = 5;  
// Befehle zum Einkaufen von mehr Vorräten  
AnzahlHonigtoepfe = 6;
```

Nicht so gut ist dagegen:

```
var AnzahlHonigtoepfe = 5;  
// Befehle zum Einkaufen von mehr Vorräten  
var AnzahlHonigtoepfe = 6;
```

Der Grund: Bei der Verwendung von `var` wird die Variable neu initialisiert. Das heißt, die alte Variable wird gelöscht und eine neue Variable erstellt. Wenn die Variable aber bereits existiert, ist der Einsatz von `var` unnötig.



In diesem und auch in einigen anderen Beispielen in diesem Buch sind Programmzeilen, die gegenüber einem vorherigen Listing verändert wurden, durch Fettdruck hervorgehoben.

3.2 Operatoren

Durch Operatoren wird eine gewisse Anzahl von Variablen miteinander kombiniert. Beispiele für Operatoren sind die Grundrechenarten. Durch den Plus-Operator werden zwei Zahlen miteinander kombiniert, und als Ergebnis erhält man die Summe dieser beiden Zahlen. Man unterscheidet – auch je nach Typ der beteiligten Variablen – verschiedene Arten von Operatoren.

3.2.1 Arithmetische Operatoren

Diese Art von Operatoren arbeitet mit numerischen Variablen. Sie sollten also sicherstellen, dass auch wirklich Zahlenvariablen vorliegen, sonst könnten Sie eine Fehlermeldung erhalten. In Kapitel 12, »Formulare II«, finden Sie Techniken, wie man Zahlenvariablen als solche erkennen kann. Tabelle 3.2 zeigt alle arithmetischen Operatoren anhand eines Beispiels.

Operator	Beschreibung	Beispiel	Ergebnis (Wert von a)
+	Addition	$a = 7 + 4$	11
-	Subtraktion	$a = 7 - 4$	3
*	Multiplikation	$a = 7 * 4$	28
/	Division	$a = 7 / 4$	1.75
%	Modulo (Restrechnung) ¹	$a = 7 \% 4$	3
-	Negation	$b = 7$ $a = -b$	-7

Tabelle 3.2 Arithmetische Operatoren

Will man eine Variable um einen bestimmten Wert erhöhen, kann man sich des folgenden Konstrukts bedienen:

```
AnzahlHonigtoepfe = AnzahlHonigtoepfe + 5;
```

Der Variablen `AnzahlHonigtoepfe` wird als Wert der alte Wert dieser Variablen plus fünf zugewiesen. Der Wert der Variablen wird also de facto um fünf erhöht. In der Praxis kommt es sehr häufig vor, dass der Wert einer Variablen um genau eins erhöht oder verringert werden soll; für diesen Fall sieht JavaScript eine Abkürzung vor:

- `AnzahlHonigtoepfe++` erhöht den Wert der Variablen um eins.
- `AnzahlHonigtoepfe--` verringert den Wert der Variablen um eins.

Die Operatoren `++` und `--` können auch direkt vor dem Variablennamen stehen. Der Unterschied liegt in der Reihenfolge, in der diese Operation im Vergleich mit anderen Operationen ausgeführt werden soll. Am Beispiel des Operators `++` soll das einmal durchexerziert werden; `--` verhält sich analog.

Das Endergebnis des Standalone-Ausdrucks (des »allein stehenden« Ausdrucks)

```
AnzahlHonigtoepfe++;
```

hat zunächst denselben Effekt (nämlich: Erhöhung um 1) wie

```
++AnzahlHonigtoepfe;
```

Einen Unterschied stellt man jedoch fest, wenn der Ausdruck bei einer Zuweisung verwendet wird:

```
var AnzahlHonigtoepfe = 5;
var Anzahl = ++AnzahlHonigtoepfe;
var Anzahl2 = AnzahlHonigtoepfe++;
```

¹ Siehe Beispiel: 7 Modulo 4 liefert 3, weil 7 bei der Division durch 4 den Rest 3 lässt. Analog ist beispielsweise 11 Modulo 4 auch 3.

Welchen Wert hat `Anzahl`, welchen Wert hat `Anzahl2`?

Die (vielleicht etwas überraschende) Antwort lautet: `Anzahl` hat den Wert 6, `Anzahl2` hat auch den Wert 6. Betrachten Sie zunächst die zweite Zeile:

```
var Anzahl = ++AnzahlHonigtoepfe;
```

Der `++`-Operator steht vor dem Variablennamen. Das bedeutet hier, dass zunächst diese Operation (`AnzahlHonigtoepfe` um eins erhöhen) ausgeführt und dann der neue Wert (6) der Variablen `Anzahl` zugewiesen wird.

Bei der dritten Zeile ist es genau andersherum:

```
var Anzahl2 = AnzahlHonigtoepfe++;
```

Zuerst wird der Variablen `Anzahl2` der (aktuelle) Wert von `AnzahlHonigtoepfe` zugewiesen, dann wird der Wert von `AnzahlHonigtoepfe` um eins vergrößert.

Wenn man den Wert einer Variablen nicht um exakt eins erhöhen oder verringern will, kann man sich einer anderen Abkürzung bedienen. Diese Abkürzung existiert für jede der vier Grundrechenarten sowie für den Modulo-Operator:

Operator	Bedeutung	Langform	Kurzform
<code>+=</code>	Addition	<code>a = a + b</code>	<code>a += b</code>
<code>-=</code>	Subtraktion	<code>a = a - b</code>	<code>a -= b</code>
<code>*=</code>	Multiplikation	<code>a = a * b</code>	<code>a *= b</code>
<code>/=</code>	Division	<code>a = a / b</code>	<code>a /= b</code>
<code>%=</code>	Modulo	<code>a = a % b</code>	<code>a %= b</code>

Tabelle 3.3 Abkürzungen für arithmetische Operationen

Auch in JavaScript gilt: Punktrechnung geht vor Strichrechnung. Multiplikationen und Divisionen werden also vor Additionen und Subtraktionen ausgeführt. Der folgende Ausdruck liefert daher 7 und nicht 9:

```
var PunktVorStrich = 1 + 2 * 3;
```

3.2.2 Boolesche Operatoren

Mit Logikoperatoren (oder booleschen Operatoren) kann man Wahrheitswerte miteinander verknüpfen. Die Bedeutung der Operatoren ist die mathematische Bedeutung, nicht unbedingt die umgangssprachliche Bedeutung. Aus diesem Grund werden die einzelnen Operatoren hier explizit vorgestellt.

UND (&&)

Nur, wenn beide (bzw. alle) beteiligten Variablen den Wert `true` haben, liefert die Operation `true` zurück, ansonsten `false`.

```
var t = true;
var f = false;
var bool1 = t && f; //liefert false
var bool2 = t && t; //liefert true
```

ODER (||)

Ist eine der beteiligten Variablen `true`, so liefert die Operation `true` zurück. Das Ergebnis ist nur dann `false`, wenn alle Variablen den Wert `false` haben. Hier liegt ein Unterschied zum Deutschen vor, denn dort bedeutet »oder« eher »entweder-oder«: Das Ergebnis ist nur dann `true`, wenn genau eine der beteiligten Variablen den Wert `true` hat.

```
var t = true;
var f1 = false;
var f2 = false;
var bool1 = t || f1 || f2; //liefert true
var bool2 = f1 || f2;      //liefert false
```

NEGATION (!)

Der Negationsoperator macht `true` zu `false` und `false` zu `true`.

```
var t = true;
var f = false;
var bool1 = !t; //liefert false
var bool2 = !f; //liefert true
```

Short Evaluation

Wie Sie bereits gesehen haben, genügt genau eine Variable mit dem Wert `true`, damit das Ergebnis einer Oder-Verknüpfung ganz sicher den Wert `true` hat. Analog liefert eine Und-Verknüpfung auf jeden Fall den Wert `false`, wenn eine Variable den Wert `false` hat.

JavaScript – zumindest die existierenden Implementierungen – benutzt hier das Prinzip der so genannten Short Evaluation (wörtlich: kurze Auswertung). Bei einer Und- bzw. Oder-Verknüpfung werden die beteiligten Variablen von links nach rechts durchgegangen. Sollte bei einer dieser Variablen aufgrund ihres Werts das Ergebnis der gesamten Operation schon feststehen, wird der Rest nicht weiter ausgewertet.

Hier ein Beispiel:

```
var f = false;
var bool = f && andereVariable;
```

Auf `andereVariable` wird hier gar nicht erst zugegriffen: Die Variable `f` ist `false`, damit kann die `&&`-Verknüpfung (logisches Und) nur `false` als Ergebnis haben.

Vergleichsoperatoren

Vergleichsoperatoren werden meistens bei Zahlenwerten verwendet. Auch bei Zeichenketten sind diese Vergleiche möglich. Hier richtet sich die Rangfolge der einzelnen Zeichen (Welches Zeichen ist »größer« als ein anderes?) nach dem ASCII-Code des Zeichens.

Operator	Beschreibung	Beispiel	Ergebnis (Wert für a)
==	Gleich	a = (3 == 4) a = ("Pooh" == "Piglet")	false
!=	Ungleich	a = (3 != 4) a = ("Pooh" != "Piglet")	true
>	Größer als	a = (3 > 4)	false
<	Kleiner als	a = (3 < 4)	true
>=	Größer oder gleich	a = (3 >= 4)	false
<=	Kleiner oder gleich	a = (3 <= 4)	true

Tabelle 3.4 Vergleichsoperatoren



Eine häufige Fehlerquelle ist die Verwechslung der Zuweisung `=` mit dem Vergleichsoperator `==`. Ab JavaScript Version 1.3 gibt der Interpreter eine Fehlermeldung aus, wenn offensichtlich ein Vergleich durchgeführt werden soll, aber der Zuweisungsoperator verwendet wird.

3.2.3 String-Operatoren

Auch mit Zeichenketten kann man »rechnen«; man kann zwei Zeichenketten aneinander hängen. Hierzu wird auch der Plus-Operator (+) verwendet:

```
var Vorname = "Winnie";
var Nachname = "Pooh";
var Baer = Vorname + " " + Nachname; //liefert "Winnie Pooh"
```

Ansonsten kann man mit Zeichenketten nicht rechnen. Dennoch sollen an dieser Stelle noch drei Möglichkeiten vorgestellt werden, um mit Zeichenketten etwas anzufangen:

- ▶ `Zeichenkette.length`: Liefert die Anzahl der Zeichen in einer Zeichenkette zurück.
- ▶ `Zeichenkette.charAt(x)`: Liefert das Zeichen an Position `x` in der Zeichenkette zurück. Dabei beginnt die Zählung bei 0, das vierte Zeichen erhält man also mit `Zeichenkette.charAt(3)`.
- ▶ `Zeichenkette.substring(start, ende)`: Liefert eine Teilzeichenkette zurück, und zwar ab dem Zeichen an der Position `start` (die Zählung beginnt wieder bei 0) und bis zu dem Zeichen vor dem Zeichen an der Position `ende`.

Hierzu ein kleines Beispiel. In den Variablen `a` und `b` stehen das erste Zeichen der Zeichenkette und die folgenden Zeichen:

```
var z = "Winnie The Pooh";
var a = z.charAt(0); //a == "W"
var b = z.substring(1, z.length); //b == "innie The Pooh"
```

3.2.4 Umwandlung zwischen den Variablentypen

Die vorgestellten Operatoren können auch dazu verwendet werden, Umwandlungen zwischen den einzelnen Variablentypen durchzuführen. JavaScript ist in Sachen Variablentypus nicht so strikt wie andere Programmiersprachen. Eine Variable kann auch ihren Typ während des Programmablaufs ändern. Beispielsweise werden Sie in einem späteren Kapitel feststellen, dass Formulareingaben stets als Zeichenketten vorliegen. Wenn Sie sich aber sicher sind, dass die Zeichenkette eine korrekt formatierte Zahl enthält, können Sie JavaScript dazu zwingen, die Variable als Zahlenwert zu betrachten. Der Trick besteht darin, die Variable mit eins zu multiplizieren (oder 0 zu addieren). Eine Multiplikation kann nur mit Zahlenwerten durchgeführt werden, so dass JavaScript die Variable in eine Zahl umwandelt – und eine Multiplikation mit eins ändert am Wert der Zahl auch nichts.

Außerdem ist es manchmal notwendig, eine boolesche oder eine numerische Variable in eine Zeichenkette umzuwandeln. Diesmal muss der **Konkatenationsoperator** (Verkettungsoperator), das Plus, verwendet werden. Indem eine Variable mit einer leeren Zeichenkette konkateniert (verkettet) wird, erhält man als Ergebnis eine Zeichenkette, ändert aber ansonsten den Wert der Variablen nicht.

```
var AnzahlHonigtoepfe = "5";
AnzahlHonigtoepfe *= 1; //Zeichenkette in Zahl
var wahrheitswert = true;
wahrheitswert += ""; //Wahrheitswert in Zeichenkette
var Anzahl = 6;
Anzahl += ""; //Zahl in Zeichenkette
```


JavaScript führt zwar eine automatische Typenkonvertierung durch, aber nicht immer in die gewünschte Richtung:

```
var siebenundvierzig = "47";  
var summe = siebenundvierzig + 11; // "4711", nicht 58
```

JavaScript stellt auch ein paar Funktionen zur Verfügung, um Umwandlungen durchzuführen. Durch `parseInt()` wird eine Zeichenkette in eine (ganzzahlige) Zahl umgewandelt, durch `parseFloat()` in eine Fließkommazahl. Innerhalb der runden Klammern wird die Zeichenkette angegeben:

```
var zahl1 = parseInt("47"); //liefert 47 als Zahl  
var zahl2 = parseInt("47.11"); //liefert 47,11 dezimal
```

Hierauf gehen wir an späterer Stelle noch einmal ein detail ein.

3.3 Kontrollstrukturen: Schleifen

Hin und wieder kommt es vor, dass eine Anweisung mehrmals ausgeführt werden muss, beispielsweise bei einer Aufzählung. Hierzu bietet JavaScript mehrere Kontrollstrukturen an. Mit JavaScript Version 1.2 (sowie beim Internet Explorer 4) wurden neue Kontrollstrukturen eingeführt, die aber mit dem Befehlssatz von JavaScript 1.0 vollständig nachgebildet werden können.

3.3.1 For-Schleifen

Diese Art der Schleife führt eine Anweisung eine (in der Regel) bestimmte Anzahl von Malen aus. Die Syntax sieht dabei folgendermaßen aus:

```
for (Initialisierung; Bedingung; Befehlsfolge) {  
    //Anweisungen  
}
```

Die `for`-Schleife hat drei Parameter:

- ▶ **Initialisierung:** Oft läuft bei einer Schleife eine Zählvariable mit, die die Anzahl der Wiederholungen zählt. Diese Variable kann hier initialisiert werden. Sollen mehrere Variablen initialisiert werden, so werden die einzelnen Anweisungen durch Kommata voneinander getrennt.
- ▶ **Bedingung:** Die `for`-Schleife wird so lange ausgeführt, bis diese Bedingung nicht mehr erfüllt ist.
- ▶ **Befehlsfolge:** Nach jedem Durchlauf der Anweisungen wird diese Befehlsfolge (in der Regel ein Befehl; mehrere Befehle werden durch Kommata voneinander getrennt) ausgeführt. Wenn die Schleife irgendwann enden soll, sollten hier in der Regel Befehle ausgeführt werden, die nach einer bestimmten Anzahl von Durchläufen die Bedingung (den zweiten Parameter) nicht mehr erfüllbar machen.

Die geschweiften Klammern um den Anweisungsblock sind dann zwingend, wenn der Block aus mehr als einem Befehl besteht. Handelt es sich um nur einen Befehl, so kann man die geschweiften Klammern weglassen. Man sollte aber zumindest den Code einrücken, damit das Ganze übersichtlich und lesbar bleibt. Generell ist es aber empfehlenswert, immer geschweifte Klammern zu verwenden.

Der folgende Code gibt zehnmal Winnie aus. Dabei wird eine Zählvariable mit 0 initialisiert und in jedem Schleifendurchlauf um eins erhöht. Die Abbruchbedingung prüft, ob die Zählvariable kleiner als zehn ist. Vor dem elften Durchlauf wird die Variable auf zehn erhöht, und die Schleife wird verlassen.

```
for (var i=0; i<10; i++)
    document.write("Winnie<br />");
```

Sehr oft wird die Zählvariable auch direkt in der Schleife verwendet. Der folgende Code gibt alle Quadratzahlen von 0 = 0 bis 10 = 100 aus:

```
for (var i=0; i<=10; i++) {
    document.write("Das Quadrat von " + i + " ist: ");
    document.write(i*i + "<br />");
}
```

Sie sehen hierbei die Verwendung des Plus-Operators: Hier werden eine Zeichenkette und eine Zahl zusammengefügt. Das Ergebnis ist eine Zeichenkette, die Zahl wird also in eine Zeichenkette umgewandelt.

Beachten Sie ebenso, dass Sie einen Zeilenumbruch nicht mit `\r\n` angeben sollten, sondern mit dem entsprechenden HTML-Tag, `
`. Bei Tags müssen Sie noch eine Besonderheit beachten, aber hierzu kommen wir erst im nächsten Kapitel.

Wie bereits erwähnt wurde, kann man auch mehrere Zählvariablen verwenden, die man dann durch Kommata voneinander trennen muss. Anwendungen dafür gibt es ziemlich selten: hier folgt ein sehr praxisfremdes Beispiel. Es werden zwei Zählvariablen verwendet, `i` und `j`. Die erste enthält eine Zahl, die zweite eine Zeichenkette. Der Zahlenwert wird in eine Zeichenkette umgewandelt und an `j` angehängt. Sobald `j` mehr als 15 Zeichen enthält, wird die Schleife verlassen.

```
for (var i=0, j=""; j.length<=15; i++, j += i)
    document.write(i + " - " + j + "<br />");
```

Dieses Programm gibt Folgendes auf dem Bildschirm aus:

```
0 -
1 - 1
2 - 12
```

```

3 - 123
4 - 1234
5 - 12345
6 - 123456
7 - 1234567
8 - 12345678
9 - 123456789
10 - 12345678910
11 - 1234567891011
12 - 123456789101112

```

Vor dem nächsten Schleifendurchlauf würde an die Zeichenkette 13 angehängt, die Länge würde dadurch auf 17 Zeichen anwachsen, also wird die Schleife hier verlassen.

3.3.2 Do-While-Schleife

Nicht immer weiß man, wie oft ein Anweisungsblock hintereinander ausgeführt werden soll. Stattdessen will man den Block so lange ausführen, bis eine Bedingung nicht mehr erfüllt ist. Im folgenden Beispiel sollen in einer Zeichenkette alle As durch Bs ersetzt werden. Hierbei ist die Methode `Zeichenkette.indexOf(Teilstring)` nützlich; diese gibt nämlich zurück, an welcher Position in der Zeichenkette der Teilstring das erste Mal vorkommt. Ist der Teilstring nicht in der Zeichenkette enthalten, wird -1 zurückgegeben. Das erste Zeichen steht, wie auch schon vorher, an Position 0. Diese Art von Schleife gibt es jedoch erst seit Netscape Navigator 4 und Internet Explorer 4.

Die Syntax sieht wie folgt aus:

```

do {
    //Anweisungsblock
} while (Bedingung);

```

Der Anweisungsblock wird ausgeführt, und dann wird die Bedingung überprüft. Ist sie erfüllt, wird der Block erneut ausgeführt (und dann wird wieder die Bedingung geprüft); andernfalls wird die Schleife verlassen.

```

var Zk = "AXAYAZ"; //Zk steht für "Zeichenkette"
do {
    io = Zk.indexOf("A");
    Zk = Zk.substring(0, io) + "B" +
        Zk.substring(io+1, Zk.length);
} while (Zk.indexOf("A") > -1);

```

Nach dem Durchlauf enthält Zk den Wert `BXBYBZ`.

Beachten Sie insbesondere, dass die Schleife auf jeden Fall einmal ausgeführt wird! Im Beispiel führt das zu einem Fehler, wenn die Zeichenkette zu Anfang

überhaupt kein A enthält. Im Folgenden lernen Sie Methoden kennen, um diesen Fehler zu vermeiden.

Noch ein Wort zu der Zuweisung: Wenn Sie bei einer Zeichenkette *z* das Zeichen an der Stelle *x* in das Zeichen *X* ändern wollen (die Zählung beginnt wie immer bei 0), kommen Sie mit folgender Anweisung weiter:

```
z = z.substring(0, x) + "X" + z.substring(x+1, z.length);
```

Alle Zeichen vor und hinter dem zu ändernden Zeichen bleiben durch die beiden `substring()`-Anweisungen erhalten.

3.3.3 While-Schleife

Schon seit JavaScript Version 1.0 gibt es eine weitere Form der Schleifen, und zwar `while`-Schleifen (ohne `do`). Die Syntax ist der von `do-while`-Schleifen sehr ähnlich, der Unterschied steckt im Detail:

```
while (Bedingung) {
    //Anweisungsblock
}
```

Die Bedingung wird hier **vor** dem Durchlaufen des Anweisungsblocks überprüft. Im Beispiel von oben, bei der Ersetzung aller As durch Bs, ist das sehr nützlich, da hier der Anweisungsblock nicht ausgeführt wird, wenn die Zeichenkette von Anfang an keine As enthält:

```
var Zk = "AXAYAZ" //Zk steht für "Zeichenkette"
while (Zk.indexOf("A") > -1) {
    io = Zk.indexOf("A")
    Zk = Zk.substring(0, io) + "B" +
        Zk.substring(io+1, Zk.length)
}
```

3.3.4 For-In-Schleife

Diese Schleife wird recht selten verwendet, und an dieser Stelle fehlt Ihnen noch das Grundwissen über Objekte, um Ihnen eine ausreichende Erklärung geben zu können. Prinzipiell sei gesagt, dass man mit der Schleife durch alle Eigenschaften eines Objekts und alle Elemente einer Variablensammlung (eines Arrays, dazu später mehr) laufen kann. Die folgende Schleife gibt das `name`-Attribut aller Elemente eines Formulars wieder. Spätestens in Kapitel 7, »Formulare I«, werden Sie diesen Code verstehen; vorerst aber müssen Sie mir blind vertrauen. Die `for-in`-Schleifen werden ohnehin sehr selten eingesetzt.

```
for (e in document.forms[0].elements) {
    document.write(e.name+"<br />")
}
```

3.3.5 Schleifensteuerung

Eine Schleife muss nicht unbedingt so oft durchlaufen werden, wie vorgesehen ist. Angenommen, in einer Schleife wird eine Zeichenkette auf das Vorhandensein eines bestimmten Zeichens überprüft (das geht mit `indexOf()` sehr schnell, aber darauf gehen wir an dieser Stelle nicht ein). Sobald das Zeichen gefunden worden ist, muss die Schleife nicht unbedingt weiter ausgeführt werden, denn das Ergebnis (»Das Zeichen ist in der Zeichenkette enthalten«) steht ja jetzt schon fest. Aus diesem Grund gibt es den Befehl `break`, der das sofortige Verlassen der aktuellen Schleife veranlasst. Der Interpreter fährt also hinter dem aktuellen Anweisungsblock fort.

Ebenso kann es bei Schleifen, insbesondere bei `for`-Schleifen, immer wieder vorkommen, dass man zum nächsten Schleifendurchlauf springen möchte (beispielsweise, wenn man genau weiß, dass dieser Durchlauf nicht das gewünschte Ergebnis bringt) und den Rest der Schleife aus Effizienzgründen nicht ausführen lassen will. Der entsprechende Befehl heißt `continue`.

Wieder ist es leider so, dass Ihr momentanes Wissen noch nicht ausreicht, um hier ein sinnvolles Beispiel anzugeben. Am Ende des nächsten Kapitels werden Sie aber in der Lage sein, die Befehle `break` und `continue` einzusetzen.

Allgemein wird die Verwendung von `break` und `continue` als eher schlechter Programmierstil angesehen; zu sehr ähneln diese Befehle dem Spaghetti-Code aus alten BASIC-Zeiten². In der Praxis werden die beiden Befehle jedoch durchaus angewandt, und zumindest im JavaScript-Bereich ist meiner Meinung nach nichts dagegen einzuwenden.

3.4 Fragen & Aufgaben

1. Welche der folgenden Variablennamen sind gültig?

- ▶ `Winnie-The-Pooh`
- ▶ `Piglet1`
- ▶ `1Rabbit`
- ▶ `Christopher Robin`
- ▶ `_1a`
- ▶ `break`

2. Woran scheitert dieser Aufruf?

```
<a href="#" onclick="alert(\"I love Tigger\");">
```

² Bei den Ur-Versionen von BASIC wurde jede Programmzeile anhand ihrer Nummer identifiziert. Schleifen gab es nur in sehr eingeschränkter Form, so dass einer der häufigsten Befehle `GOTO` hieß. Mit ihm konnte eine Zeilennummer direkt angesprungen werden. Dadurch wurde der Quellcode sehr unübersichtlich und auch unsauber.

3. Was sind die prinzipiellen Unterschiede zwischen einer `while`- und einer `do-while`-Schleife?
4. Erzeugen Sie einen JavaScript-Code, der in einer Zeichenkette alle Leerzeichen durch `%20` ersetzt.

Index

! 77
- 75
-- 75
!= 78
574
% 75, 502
%= 76
&& 77
* 75
*/ 63
*= 76
+ 75, 78
++ 75
+= 76
/ 75
/* 63
// 62
/= 76
-= 76
= 78
== 78
| 311
|| 77

A

about:cache 539
absolute Positionierung 603, 692, 699
ActionScript 864
ActiveMovie 827
ActiveMovie Control 297
 Methoden 301
ActivePerl 416
ActiveX 462, 825, 847
Adressbuch 650
alert() 118
Anführungszeichen 678
Animation 198
animierte GIFs 673, 847
Apache 566, 869
appendChild() 420
appName 113
appVersion 113
arguments 98, 576, 593
Array-Objekt 94, 103, 199, 279
 für JavaScript 1.0 94

Index 94
length 95, 280
pop() 281
push() 281, 288
sort() 282, 284
splice() 281
toString() 285
Arrays 279
 sortieren 282
 Sortierfunktion 283
ASP 465, 869
 Bankleitzahlen 881
 Caching verhindern 551
 Newsticker 874
 Newsticker mit Links 877
 Passwortschutz 565
 Referer-Check 554
 Session-Management 565
ASP.NET 465
 HTML Controls 470
 Validation Controls 475
 Web Controls 472
.aspx 465
Auswahllisten 146, 147
 options 146
 selected 146
 selectedIndex 147, 153
 type 153
 value 146
Authorization required 555

B

back() 130
Bankleitzahlen 880
Banner 616
Bannervermarkter 514
base 647
Baumstruktur 710
BBEEdit 32
Behavior 452
Blackjack 403
blur() 159
Bookmark 170
Browser-Erkennung 112, 491, 492
 Internet Explorer 114

- Netscape Navigator 114
- Opera 115
- Versionsnummer 116
- BSI 29
- Bubblesort 288
- Bubbling 320, 356
- Bugs 29

C

- Cache 200, 539, 550
- Camino 44
- ceil() 110
- charAt() 79, 772
- Checkboxes 146
 - checked 146
 - type 153
- classes-Objekt 334
- clearInterval() 137
- clearTimeout() 135
- Client Sniffer 113
- clientseitige Programmierung 29
- cloneNode() 420
- Code auslagern 549
- Code schützen 535
- Code verschlüsseln 544
- Code verstecken 539
- confirm() 126
- continue 84
- Cookies 123, 237, 255, 513
 - alle auslesen 527
 - Domain 517
 - domain 238
 - Einschränkungen 513, 530
 - Elemente 515
 - expires 239
 - Haltbarkeitsdatum 516
 - lesen 242, 522
 - löschen 241, 524
 - mit HTML 519
 - Name 515
 - path 239
 - per HTTP-Header 515
 - permanent 516
 - Pfad 518
 - Reihenfolge der Parameter 519
 - secure 240
 - Seiten schützen 559
 - serialisieren 530

- setzen 241, 520
- Sicherheitsstufe 518
- Spezifikation 513
- temporär 516
- Unterstützung 525
- Unterstützung prüfen 242
- Warenkorb 739
- Wert 515

- Copy & Paste 150
- createElement() 420
- CSS 332, 353

D

- Date-Objekt 103, 500
 - erzeugen 103
 - getDate() 104
 - getFullYear() 106
 - getHours() 104
 - getMilliseconds() 104
 - getMinutes() 104
 - getMonth() 104
 - getSeconds() 104
 - getTime() 107, 500
 - getYear() 104, 105
 - toGMTString() 239, 516, 522
- DCOM 442
- default → switch-Anweisung 91
- DHTML 195, 331, 601
 - Banner 616
 - Breite 612
 - Browserunterscheidung 497
 - Browserunterstützung 602, 607, 610
 - Checkliste 620
 - Definition 601
 - Farbe ändern 605, 606, 797
 - Fensterbreite 619
 - Fensterhöhe 619
 - Höhe 612
 - Internet Explorer 605
 - Konqueror 606
 - Laufschrift 692
 - Navigation 709
 - Netscape 603, 606
 - Position 613, 614
 - Safari 606
 - sichtbar 612, 710
 - Skripten anpassen 602

- Stile 611
- Text ausgeben 579, 604, 605, 606, 615, 632, 699
- unsichtbar 612, 710
- verschieben 614
- W3C 606
- Windows-Plattform 609
- Zugriff 611
- DHTML Menu Builder 718
- Director 847, 850
 - ActiveX überprüfen 851
 - browserunabhängige Erkennung 854
 - in HTML einbauen 850
 - Plugin überprüfen 852
 - Verbreitung 848
 - Veröffentlichungseinstellungen 848
 - Versionsnummer 852, 853
- disablePrivilege() 414
- Distributed Computing 442
- document-Objekt
 - all 354, 378, 497, 605, 797, 831
 - all.style 354
 - body.clientHeight 616
 - body.clientWidth 616
 - body.offsetHeight 640
 - body.offsetWidth 640
 - close() 604, 615, 631
 - cookie 240, 243, 520
 - forms 143, 760
 - getElementById() 366, 378, 497, 606, 797
 - images 194, 198, 574
 - layers 378, 497, 597
 - links 681
 - open() 604, 615
 - style 366
 - write() 55, 59, 96, 218, 485, 549, 604, 615, 849
- DOM 31, 318, 353, 366, 417
- DOM-Baum 417
- do-while-Schleifen 82, 87, 89
- Drag&Drop 332, 339, 356, 359, 381
- Drucken 131, 490, 496
- Druckversion 485

E

- ECMA-262 31, 417
- ECMAScript 31, 417
- E-Commerce 176
- Eigenschaften → Objekte 100
- elements 144
- else → if-else-Anweisung 88
- embed 826, 839, 848, 856
- enablePrivilege() 410, 414, 457
- encodeURIComponent 258
- encodeURIComponent() 258
- Entities 69
- Ereignisbehandlung 307
- Ereignisse 307
 - abfangen 311
 - browser-unabhängig 324
 - Bubbling 324
 - Internet Explorer 318
 - Mausclicks 325
 - mit Opera 311
 - Netscape 307
 - Sondertasten 327
 - Tastatureingaben 316, 326
- escape() 258, 726, 752
- eval() 92, 155, 177, 268, 290, 361, 505, 530, 670, 780
- Event-Handler 69, 120, 427
 - mouseup 340
 - onabort 214
 - onblur 120, 159, 660
 - onChange 881
 - onchange 120, 159, 260, 702
 - onclick 69, 120, 127, 308, 485, 646
 - oncontextmenu 544
 - ondblclick 308, 318
 - onerror 214, 427
 - onfocus 120, 159, 233, 658
 - onkeydown 316, 318
 - onkeypress 316, 318
 - onkeyup 316, 318
 - onload 120, 213, 309, 662
 - onmousedown 308, 318, 340
 - onmousemove 308, 340
 - onmouseout 120, 133, 195, 681
 - onmouseover 120, 133, 195, 681
 - onmouseup 308, 318
 - onselect 159

- onsubmit 142, 791
- onunload 120, 667
- Event-Objekt 311, 313, 322, 340
- cancelBubble 323
- CLICK 311
- clientX 357
- clientY 357
- keyCode 326
- MOUSEDOWN 311
- MOUSEUP 311
- pageX 341
- pageY 341
- srcElement 322, 383
- target 313, 383
- type 313, 383
- which 317
- externe Datei 65

F

- FAQ 34
- Fehler
 - abfangen 427
 - detaillierte Meldung 428
- Fenster
 - Breite 616
 - ewig 664
 - Höhe 616
 - im Hintergrund 658
 - im Vordergrund 660
 - links oben 639
 - links unten 640
 - öffnen 622
 - öffnendes Fenster 623, 645
 - Optionen 622, 625
 - positionieren 638
 - rechts unten 640
 - schließen 625
 - verschieben 666
 - Vollbild 644
 - zentrieren 643
- Fenster-Eigenschaften 221, 225
 - center 225
 - dependent 221
 - dialogHeight 225
 - dialogLeft 225
 - dialogTop 225
 - dialogWidth 225
 - directories 221

- height 221
- help 225
- innerHeight 221
- innerWidth 221
- left 221
- menubar 221
- outerHeight 221
- outerWidth 221
- resizeable 225
- screenX 221, 223
- screenY 221, 223
- scrollbars 221
- status 221, 225
- toolbar 221
- top 221
- width 221
- Firefox 31, 45
- firstChild 419, 460
- Flash 847, 856
 - ActiveX überprüfen 857
 - browserunabhängige Erkennung 859
 - Clip laden 865
 - Fortschrittsanzeige 866
 - in HTML einbauen 856
 - Jukebox 867
 - mit JavaScript kommunizieren 861, 864
 - Plugin überprüfen 858
 - Verbreitung 847
 - Veröffentlichungseinstellungen 848
 - Wiedergabe 864
 - zoomen 865
- floor() 110
- for-in-Schleifen 83
- Form-Objekt
 - elements 144, 145, 816
- Formular
 - Auswahlliste 702, 762, 766
 - Checkbox 760, 764
 - Eingaben prüfen 759, 789
 - Element finden 684, 763
 - Elementstyp 768
 - Muster 770
 - Passwortfeld 760
 - per E-Mail versenden 735
 - Radiobutton 761, 765
 - Textfeld 760, 764

- Versand unterbinden 791
- Vollständigkeitsüberprüfung 762
- Zugriff 760
- Formularüberprüfung 141
 - automatisch 150
 - Checkliste 822
 - Fehlermeldung 792, 794
 - grafische Fehlermeldung 795, 801
 - Korrekturmöglichkeit 808
 - Pflichtfelder 815
 - Versand unterbinden 791
 - vollautomatisch 813, 815
- for-Schleifen 80, 84
- Fortschrittsanzeige 577
- frame 589
- Frames 131, 163
 - Alternativen 597
 - ändern 592, 593
 - behalten 595
 - drucken 172, 596
 - finden 592
 - füllen 589
 - mehrere gleichzeitig ändern 174
 - mit JavaScript füllen 165
 - Nachteile 589
 - Name 595
 - parent 591
 - top 591
 - unsichtbar 540, 723
 - Vorteile 589
 - zugreifen 591
- frameset 589
- fscommand() 861
- Funktionen 96
 - Anzahl Parameter 98
 - arguments 98
 - Parameter 98
 - return 96

G

- Galeon 44
- Gauß 436
- Gecko 331
- getDate() → Date-Objekt 104
- getFullYear() → Date-Objekt 106
- getHours() → Date-Objekt 104
- getMilliseconds() → Date-Objekt 104
- getMinutes() → Date-Objekt 104

- getMonth() → Date-Objekt 104
- getSeconds() → Date-Objekt 104
- getTime() 500
- getYear() → Date-Objekt 104
- GIF
 - animiert 198
 - transparent 203
- gläserner Surfer 513
- GMT-Format 516
- go() → history-Objekt 130
- GP_besteht_aus() 771
- GP_browser_ie() 493
- GP_browser_konqueror() 493
- GP_browser_n6() 493
- GP_browser_nn() 493
- GP_browser_opera() 493
- GP_browser_safari() 494
- GP_browser_version() 494
- GP_codeschutz_decode() 547
- GP_codeschutz_encode() 546
- GP_codeschutz_kontext() 543
- GP_cookie_laden_collection() 532
- GP_cookie_lesen() 523
- GP_cookie_lesen_array() 528
- GP_cookie_lesen_collection(name) 532
- GP_cookie_loeschen() 525
- GP_cookie_schreiben_
 - collection(name, wert) 532
- GP_cookie_setzen() 521
- GP_cookie_speichern_collection(c) 532
- GP_cookie_support() 527
- GP_dhtml_breite() 619
- GP_dhtml_getX() 613
- GP_dhtml_getY() 613
- GP_dhtml_hide() 612
- GP_dhtml_hoehe() 619
- GP_dhtml_moveBy() 615
- GP_dhtml_moveTo() 615
- GP_dhtml_obj() 611
- GP_dhtml_setX() 614
- GP_dhtml_setY() 614
- GP_dhtml_show() 613
- GP_dhtml_style() 611
- GP_dhtml_style_breite() 612
- GP_dhtml_style_hoehe() 612
- GP_dhtml_support() 611

GP_dhtml_text() 616
 GP_email() 779, 786
 GP_email_form() 816
 GP_ermittle_wert_2() 776
 GP_fenster_oeffnen() 622
 GP_fenster_oeffnen_lo() 639
 GP_fenster_oeffnen_lu() 641
 GP_fenster_oeffnen_ro() 641
 GP_fenster_oeffnen_ru() 642
 GP_fenster_oeffnen_vollbild() 645
 GP_fenster_oeffnen_xy() 638
 GP_fenster_oeffnen_zentriert() 643
 GP_fenster_schliessen() 625
 GP_fliesskommazahl() 774, 784
 GP_formular_check_fehler() 792
 GP_formular_check_fehler_automa-
 tisch() 819
 GP_formular_check_fehler_farbig()
 798, 804
 GP_formular_check_fehler_farbig_
 prompt() 810
 GP_formular_check_fehler_text() 794
 GP_formular_prompt() 809
 GP_formular_prompt_x() 819
 GP_frames_finden() 592
 GP_frames_laden() 593
 GP_frames_laden_ref() 593
 GP_ganzzahlig() 773, 783
 GP_geburtsdatum() 778, 785
 GP_images_fortschritt() 577
 GP_images_geladen() 579
 GP_images_preload() 576
 GP_images_preload_array() 576
 GP_images_preload_array_return()
 576
 GP_images_reset() 586
 GP_images_swap() 583
 GP_images_swap_x() 587
 GP_ist_jahr() 777
 GP_ist_monat() 777
 GP_ist_tag() 777
 GP_kein_kontext() 542
 GP_leer() 763
 GP_leer_checkbox() 765
 GP_leer_checkbox_obj() 765
 GP_leer_formular() 769
 GP_leer_radio() 766
 GP_leer_radio_obj() 765
 GP_leer_select() 767
 GP_leer_select_mult() 768
 GP_leer_select_mult_obj() 767
 GP_leer_select_obj() 767
 GP_leer_text() 764
 GP_leer_text_obj() 764
 GP_macromedia_director() 854
 GP_macromedia_flash() 859
 GP_mp_ende() 843
 GP_mp_ende_reset() 843
 GP_mp_getControls() 844
 GP_mp_getVolume() 842
 GP_mp_lade_datei() 843
 GP_mp_nn() 840
 GP_mp_obj() 841
 GP_mp_play() 841
 GP_mp_setControls() 844
 GP_mp_setVolume() 842
 GP_mp_start() 842
 GP_mp_start_reset() 843
 GP_navigation_laden() 703
 GP_nicht_negativ() 772, 783
 GP_nur_ziffern() 771
 GP_positiv() 772, 783
 GP_postleitzahl() 775, 784
 GP_schaltjahr() 785
 GP_setze_class() 797
 GP_setze_image() 802
 GP_sound_ende() 834
 GP_sound_ende_reset() 835
 GP_sound_getVolume() 833
 GP_sound_ie() 831
 GP_sound_lade_datei() 835
 GP_sound_nn() 831
 GP_sound_obj() 832
 GP_sound_pause() 833
 GP_sound_play() 833
 GP_sound_setVolume() 834
 GP_sound_start() 834
 GP_sound_start_reset() 835
 GP_sound_stop() 833
 GP_suche_formular_obj() 763
 GP_telefonnummer() 775, 784
 GP_ticker_div() 693
 GP_ticker_div_link() 697
 GP_ticker_div_link_zeigen() 697
 GP_ticker_pos() 688
 GP_ticker_status() 679

- GP_ticker_text() 685
- GP_ticker_text_link() 687
- GP_ticker_text_link_click() 690
- GP_url_oeffnen_fenster() 645
- GP_warenkorb_artikel() 720
- GP_warenkorb_editieren_cookies() 740
- GP_warenkorb_editieren_frames() 726
- GP_warenkorb_editieren_url() 750
- GP_warenkorb_hinzufuegen_cookies() 740
- GP_warenkorb_hinzufuegen_frames() 725
- GP_warenkorb_hinzufuegen_url() 749
- GP_warenkorb_kategorie() 722
- GP_warenkorb_laden_collection() 748
- GP_warenkorb_lesen_collection() 749
- GP_warenkorb_schreiben_collection() 749
- GP_warenkorb_speichern_collection() 749
- GP_zufall_array() 505
- GP_zufall_array_hp() 505
- GP_zufall_banner() 509
- GP_zufall_datum() 502
- GP_zufall_hp() 501
- GP_zufall_intervall() 503
- Grafiken
 - austauschen 571
 - Ladestatus 577, 579, 581
 - Rollover 573, 583
 - skalieren 570
 - vorladen 574, 575

H

- hasChildNodes() 420
- Heimkino 844
- history-Objekt 129
 - back() 130
 - forward() 130
 - go() 130, 139
- HotJava 32, 197, 494
- HP-Verfahren 500
- href → location-Objekt 118
- .htaccess 566
- HTML Controls 470
- HTML Guard 549

- HTML-Tags
 - a href 69, 195
 - a style 332
 - a target 169, 219, 228
 - applet 396
 - bgsound 300
 - br 81
 - div 354, 357, 366, 377
 - div style 354, 366
 - embed 300
 - embed autostart 305
 - embed name 305
 - embed src 305
 - form 142
 - form action 257
 - form method=get 257
 - form name 143
 - frame 164
 - frame src 164
 - frameset 164, 169
 - frameset cols 164
 - frameset rows 164
 - hr 59
 - img name 195, 340
 - input type=checkbox 146
 - input type=hidden 144
 - input type=password 144
 - input type=radio 145
 - input type=text 70, 144
 - layer 334, 377
 - marquee 137
 - noframes 64
 - noscript 64
 - option 146
 - script 55
 - script event for 319
 - script language 57, 60, 113, 197
 - script src 65, 66, 248
 - select 146
 - select multiple 147
 - style 332, 343
 - textarea 144
- HTTP 237
- HTTP_REFERER 552
- HTTP-Header 238, 257, 515, 527
- HTTP-Protokoll 720
- HTTPS 240, 519

I

- ids-Objekt 334
- if-else-Anweisung 128
- IIS 447, 465, 526
- Image-Objekt 193, 569, 801
 - Browserunterstützung 574
 - Cache 200, 212
 - complete 214, 577
 - erstellen 570
 - Größe der Grafik 570
 - Kompatibilität 195
 - src 195, 201, 570
- indexOf() 524, 770
- indexOf() → String-Objekt 82
- innerHTML 605, 606, 615
- innerText 605, 615
- insertBefore() 420
- IntelliSense 466
- isNaN() 264, 268, 494, 771

J

- Jahr-2000-Problem 105
- Jahrtausendwechsel 107
- JAR 415
- Java 55, 410
 - Datentypen 397
 - Klassen 397
 - Vector 402
 - Virtual Machine 395
- javaEnabled() → navigator-Objekt 400
- JavaScript
 - neu in Version 1.5 277
 - Unterstützung 483
 - Versionen 487
- JavaScript Debugger 434
 - Abort 439
 - Breakpoint 437
 - Inspector 438
 - Installation 434
 - Into 439
 - Konsole 438
 - Out 439
 - Over 439
 - Run 439
 - Watch 437
- JavaScript-Protokoll 68, 166

- Java-Unterstützung 831
- Joust 716
- JPEG 198
- JScript 30, 57
- JScript .NET 446, 465
- JScript.NET
 - Installation 447
 - Web Service 448

K

- KDE 972
- Konqueror 36, 478
- Kontextmenü verhindern 544
- Kontrollstrukturen 80

L

- lastChild 419
- Laufschrift 136, 673
 - Algorithmus 676
 - in der Statuszeile 679
 - in einem Textfeld 683
 - mit DHTML 692
 - mit HTML 674
 - mit Links 687, 695
- Layer 334
 - bewegen 346
 - immer sichtbar 348
 - verstecken 343
- layer 603
- layers-Objekt
 - document 335
 - left 337, 339
 - parentLayer 335
 - posLeft 355
 - posTop 355
 - top 337, 339
 - visibility 344
- length 79, 95, 96
- Links 485, 486, 593, 621, 646
- LiveAudio 297, 300, 829
 - Methoden 301
- LiveConnect 434
- LiveScript 30
- location-Objekt 118, 138
 - href 118, 138, 172, 592, 668, 701
 - protocol 69
 - reload() 139, 186

replace() 139, 608
search 171, 248, 252, 257, 747
Lottozahlen 506

M

Macromedia 847
 Director 847
 Flash 847
MagicCookie 238
marquee 674
Math-Objekt 103, 109
 ceil() 110
 floor() 110, 501
 PI 501
 pow() 501
 random() 111, 499
 round() 110, 643
Mausklick verhindern 541
meta 520
Methoden → Objekte 100
Microsoft SQL Server 876
MIDI-Format 826
MIME-Typ 67, 299
mimeType-Objekt
 description 299
 enabledPlugin 299
 suffixes 299
 type 299
modale Fenster 123, 225
Modulo 502
Modulo-Rechnung 76
Mozilla 126, 417
Multimedia 825
Musik 826
 ActiveMovie 827
 browserunabhängige Ansteuerung
 830
 Clip laden 829, 830, 835
 Einbau in HTML 826
 Lautstärke 828, 830, 833
 LiveAudio 829
 Objekt 832
 Wiedergabe 827, 829, 832
 Wiedergabestatus 828, 829
 Windows Media Player 827
 Wurlitzer 836
Muster 770
 E-Mail 778

Fließkommazahl 773
Geburtsdatum 775
numerischer Wert 771
Postleitzahl 774
reguläre Ausdrücke 269
Telefonnummer 775
Ziffer 771
MyGalileo 30, 124, 141
MySQL 876

N

NaN 494
Navigation 203, 701
 Alternativen im Web 715
 mit Auswahllisten 701
 mit DHTML 709
 verschachteln 704
navigator-Objekt 113, 491
 appName 491
 appName 113, 491
 appVersion 113, 491
 cpuClass 491
 javaEnabled() 305, 400, 831
 mimeType 849, 852, 858
 mimeType → mimeType-Objekt
 299
 platform 491, 851
 plugins 852
 plugins → plugins-Objekt 298
 userAgent 113, 197, 491, 851
.NET 446, 465
 Installation 447
.NET Framework 448, 465
 Versionen 448
Netscape 6/7 365, 424
 DHTML 367
 DOM 366
 Ereignisbehandlung 329
 Erkennung 117
 Objektzugriff 366
netscape-Objekt
 security.PrivilegeManager 628, 658
Newsgroups 33
Newsticker 874
nextSibling 419
nodeName 419
nodeType 419
noscript 484

NoteTab 32
null 98, 128

O

object 826, 839, 848, 856
Object Signing Tool 416
Objekte 100, 103
 eigene erstellen 286
 Eigenschaften 100, 287
 Konstruktor 286
 Methoden 100, 287
 this 286
ODBC 876
onblur 120, 660
onblur → Event-Handler 159
onChange 881
onchange 120, 702
onclick 69, 120, 485, 646
oncontextmenu 544
onerror 427
onerror → Event-Handler 427
onfocus 120, 658
onfocus → Event-Handler 159
onload 120, 662
onmouseout 120, 681
onmouseover 120, 681
onreadystatechange 462
onselect → Event-Handler 159
onsubmit 791
onsubmit → Event-Handler 142
onunload 120, 667
OOP 100
opener 226
Opera 32, 197
 Browser-Identifikation 491
Operatoren
 arithmetische 74
 Boolesche 76
 für Zeichenketten 78
Option-Objekt 263, 706

P

parentNode 419
parseFloat() 155, 268, 492, 493, 494, 774
parseInt() 268, 368, 772
Passwort
 ähnlich URL 557

als URL 556
im Java-Applet 561
im Quelltext 560
verschlüsseln 558

Perl 430

PHP 869

Bankleitzahlen 881
Caching verhindern 551
Code generieren 552
Cookies 526
Newsticker 876
Newsticker mit Links 879
Passwortschutz 563
Referer-Check 553
Session-Management 563

Plugin 297, 825, 847

plugins-Objekt

 filename 298
 refresh() 299

Popup-Blocker 624

Positionierung

 absolute 333
 relative 333, 337

Preloading 574, 575

previousSibling 419

Privileg 410, 457

Privilege Manager 629

Programmer's File Editor 32

prompt() 128

protocol → location-Objekt 69

Punkt vor Strich 76

push() 531, 576

PWS 526

Q

Quellcode einsehen 535

 per Dateisystem 538

 per Kontextmenü 537

 per Menü 536

 per Tastenkürzel 536

Query-String 248

Quicklinks 701

R

Radiobuttons 145

 checked 145

 name 145

 type 152

- Radiobuttons → HTML-Tags 145
- random() → Math-Objekt 111
- RegExp-Objekt 271, 782
 - exec() 273, 782
 - match() 273
 - test() 272, 782
- reguläre Ausdrücke 269, 781
 - E-Mail 786
 - Fließkommazahl 783
 - Geburtsdatum 784
 - Match 272
 - Metazeichen 270
 - Muster 269, 782
 - numerischer Wert 783
 - Postleitzahl 784
 - Telefonnummer 784
- reload() 139
- reload() → location-Objekt 186
- removeNode() 420
- replace() → location-Objekt 139
- replaceNode() 420
- return 96
- return false 128, 142, 144, 317
- return true 128, 133, 142, 151, 317, 428
- Rollover 573, 583
- ROT13 546
- round() 110
- runat= 467

S

- Safari 36
- Schaltjahr 777, 785
- Schleifen 80
- screen-Objekt
 - availHeight 234, 640
 - availWidth 234
- script 488
- Scrollen 235
- selectedIndex
 - Auswahllisten 153
- self 123, 167, 226
- Serialisierung 530
- serverseitig 869
 - Bankleitzahlen 880
 - Code generieren 551
 - Newsticker 874
 - Variablenzugriff 870
- serverseitige Programmierung 29

- Session 519
- setAttribute() 420
- setInterval() 137
- setTimeout() 134
- Short Evaluation 77, 301
- Signierte Skripte 628
- Signiertes Skript 409, 415
- Sitemap 647
- SmartUpdate 124, 299, 415
- SOAP 442, 444
 - Body 445, 460
 - Envelope 445
- Sortierfunktion → Array-Objekt 283
- split() → String-Objekt 180
- SSI 247
- StarOffice 32
- status 133
- Statuszeile 132, 679
- String-Objekt 269
 - charAt() 79
 - fromCharCode() 326
 - indexOf() 82, 269
 - length 79, 96
 - replace() 276
 - Sonderzeichen 73
 - split() 180, 283
 - substring() 79, 269
- substr() 770
- substring() 79
- switch-Anweisung 90
 - default 91

T

- tags-Objekt 333
- Tastatureingaben → Ereignisse 316
- this 123, 167, 286
- Timeouts 134, 189, 205, 661, 663, 677
- toGMTString() 516, 522
- top 169
- toString() 285, 531

U

- UltraEdit 32
- unescape() 258, 726
- UniversalBrowserAccess 410, 413
- UniversalBrowserRead 410, 411
- UniversalBrowserWrite 410, 413
- UniversalFileAccess 410

- UniversalFileRead 410
- UniversalFileWrite 410
- UniversalPreferencesRead 410
- UniversalPreferencesWrite 410
- UniversalSendMail 410
- URI 258
- URL
 - Maximallänge 747
- userAgent 113

V

- Validation Controls 475
- var 74, 99
- Variablen 71
 - Boolsche 73
 - Deklaration 74
 - globale 99, 134, 199
 - lokale 99, 134
 - Namensgebung 71
 - numerische 72
 - Typenumwandlung 79
 - vertauschen 289
 - Zeichenketten 72
- VBScript 57
- Verisign 416
- Verlauf 129
- Visual Studio .NET 466
- Vollständigkeitsüberprüfung 762
 - Auswahlliste 766
 - Checkbox 764, 769
 - global 768
 - Radiobutton 765, 769
 - Textfeld 764

W

- W3C 335, 602
- Währungsrechner 155
- Warenkorb 719
 - ändern 735, 745, 755
 - Artikel anzeigen 728, 741, 751
 - Datenstruktur 720
 - Einzelansicht 729, 742, 752
 - füllen 725, 739, 749
 - Kategorien anzeigen 726, 740, 750
 - mit Cookies 245, 739
 - mit Frames 723
 - mit URL 747
 - Übersicht 733

- Vergleich der Lösungen 757
- Web Controls 472
- Web Matrix 465
- Web Service 441
 - asynchron 454, 459
 - asynclnvoke() 459
 - aufrufen 444
 - callService() 454
 - encode() 459
 - Selbstbeschreibung 443
 - SOAPCall-Objekt 458
 - SOAPParameter-Objekt 458
 - useService() 454
 - Zugriff mit Internet Explorer 452
 - Zugriff mit Mozilla 456
- Webcrypt Pro 549
- Webmail 218, 650
- WebWasher 624, 664, 669
- Weiterleiten 483, 484, 580
- Werbebanner 217
- while-Schleifen 83, 87
- window-Objekt 123, 163
 - alert() 118, 123, 124, 158
 - back() 130
 - blur() 233, 659
 - Bubbling 320
 - captureEvents() 311, 315, 319, 356, 381
 - clearInterval() 137, 677
 - clearTimeout() 135, 677, 681
 - close() 224, 229, 625
 - closed 230, 664
 - confirm() 126
 - focus() 497
 - forward() 130
 - frames 173, 591
 - handleEvent() 314
 - innerHeight 597, 616
 - innerWidth 349, 597, 616
 - moveBy() 234, 614
 - moveTo() 234, 614
 - name 228
 - onfocus 658
 - open() 217, 622, 664
 - open() → Fenster-Eigenschaften 221
 - opener 226, 623, 646
 - outerHeight 234, 640

- outerWidth 234, 640
- pageXOffset 349
- pageYOffset 349
- parent 591
- print() 131, 496
- prompt() 128, 148, 808
- releaseEvents() 311
- resizeTo() 644
- routeEvent() 315
- scroll() 235
- setInterval() 137, 411, 661, 663, 677
- setTimeout() 134, 338, 661, 663, 677
- showModalDialog() 225
- status 133, 679
- top 591
- Windows Media Player 301, 827, 837
 - browserunabhängige Ansteuerung 840
 - Clip laden 843
 - Heimkino 844
 - in HTML einbauen 838
 - Lautstärke 842
 - Objekt 841
 - Steuerelemente anzeigen 844
 - Wiedergabe 841
- write() 55
- WSDL 443, 451
- Wurlitzer 836
- WYSIWYG 465

X

- XML 441, 442
- XMLHttpRequest 462

Y

- Y2K 105

Z

- Zeichenkette
 - String-Objekt 73
- Zigbert 416
- ZIP 415
- Zufall 109, 111, 499
- Zufallsbanner 508
- Zufallszahlen 499
 - Bereich 503
 - mehrere 504