

Mark Lubkowitz

Galileo Computing

Webseiten

programmieren und gestalten



Auf einen Blick

Teil 1	Grundlagen.....	37
Teil 2	HTML – Die Sprache des Internets.....	109
Teil 3	CSS – Layout fürs Internet.....	271
Teil 4	JavaScript	381
Teil 5	Perl – Dynamisch und Interaktiv I	501
Teil 6	PHP – Dynamisch und Interaktiv II.....	633
Teil 7	MySQL – Der Datenspeicher	751
Teil 8	Dynamische Bildgenerierung mit PHP, Perl und SVG	831
Teil 9	Workshop – Erstellung eines Content Management Systems	939

Inhalt

Einführung 31

Teil 1 Grundlagen

1 Arbeitsmittel 39

1.1	Editoren	39
1.2	Webserver.....	40
1.3	Perl.....	42
1.4	PHP	42
1.5	MySQL.....	42
1.6	PhpMyAdmin	43

2 Das Internet 45

2.1	Entwicklung	45
2.2	Das statische Web	46
2.3	Das Plug-in-Web	47
2.3.1	JavaApplets	47
2.3.2	ActiveX-Controls	48
2.3.3	Macromedia Flash	49
2.4	Das dynamische Web.....	50
2.5	Internetdienste	52
2.5.1	E-Mail	52
2.5.2	Suchmaschinen	53
2.5.3	FTP	56
2.5.4	Telnet	56
2.5.5	Whois	57

3 Die »Netzüberflieger« 59

3.1	Internet Explorer	59
3.2	Netscape Navigator	60
3.3	Opera	62
3.4	Mozilla.....	63

4 Gute Vorbereitung ist wichtig 65

4.1	WAMP.....	65
4.1.1	Apache installieren	65
4.1.2	MySQL installieren.....	68
4.1.3	PHP installieren	70
4.1.4	Perl installieren	73
4.1.5	PhpMyAdmin installieren.....	75
4.2	LAMP.....	76
4.2.1	MySQL installieren.....	77
4.2.2	Apache installieren	78
4.2.3	PDF Library.....	79
4.2.4	PHP installieren	80
4.2.5	Perl installieren	81
4.2.6	PhpMyAdmin	82
4.3	Foxserv	82

5 Der Weg ins Internet 85

5.1	Wahl des Providers.....	85
5.1.1	AOL, T-Online & Co.	90
5.2	Upload der Seite.....	90
5.2.1	Grafischer Client unter Windows	90
5.2.2	Kommandozeilenbasierter Client unter Windows und UNIX/Linux	94
5.3	Suchmaschinen, Webkataloge und anderes	96

6 Adressierung 99

6.1	TCP/IP.....	99
6.1.1	Aufbau und Struktur	100
6.2	DNS	102

6.3	Adressierung, URIs	103
6.3.1	Ports	105
6.3.2	Protokolle	106
6.3.3	URI	107

Teil 2 HTML – Die Sprache des Internets

1 HTML – Die Sprache des Internets 111

1.1	Am Anfang war HTML	111
1.2	HTML-Versionen	112
1.2.1	HTML 1	112
1.2.2	HTML 2	112
1.2.3	HTML 3.2	113
1.2.4	HTML 4 und 4.01	113
1.3	Das erste HTML-Dokument	113
1.4	Elemente und Tags	115
1.5	Hierarchie	116
1.6	Attribute	117
1.7	Kommentare und Quellstrukturierung	118
1.8	Entities	119
1.9	Dokumententyp-Definition	120
1.9.1	Sprachvarianten	121
1.9.2	DTDs für andere HTML-Versionen und für XHTML	122
1.9.3	HTML-Vorlage	122
1.10	Zusammenfassung	123
1.11	Fragen und Übungen	123

2 Textstrukturierung 125

2.1	Absätze	125
2.2	Textausrichtung	126
2.3	Zeilenumbrüche	128
2.4	Vorformatierter Text	130
2.5	Überschriften	131
2.6	Physische Textauszeichnung	133
2.7	Logische Textauszeichnung	136
2.8	Schriftformatierung	137
2.8.1	Vererbung	139
2.8.2	Schriftart	140

2.8.3	Schriftgröße	140
2.8.4	Schriftfarbe	141
2.9	Dateiweite Farben	145
2.9.1	Hintergrund und Textfarbe.....	145
2.9.2	Hyperlink-Farben.....	145
2.10	Listen	146
2.10.1	Nummerierte Listen.....	146
2.10.2	Aufzählungslisten.....	149
2.10.3	Definitionslisten.....	150
2.10.4	Listenattribute	153
2.11	Zusammenfassung	155
2.12	Fragen und Übungen	155

3 Tabellen 157

3.1	Aufbau und Strukturierung.....	157
3.2	Die wichtigsten Elemente.....	158
3.3	Größe, Rahmen und Abstände.....	161
3.4	Tabellen und Inhalte ausrichten	163
3.4.1	Tabelle horizontal ausrichten	163
3.4.2	Zelleninhalt horizontal ausrichten	163
3.4.3	Zelleninhalt vertikal ausrichten.....	164
3.4.4	Ausrichtung an Dezimalzeichen	165
3.4.5	Beispiel für die Ausrichtung von Zelleninhalten	165
3.5	Farben und Schrift	167
3.5.1	Tabellenhintergrund	167
3.5.2	Zellenhintergrund	167
3.5.3	Schriftformatierung.....	168
3.5.4	Farb- und Schriftbeispiel.....	169
3.6	Zellenverbund.....	170
3.6.1	Über- und nebeneinander liegende Zellen verbinden	177
3.7	Gruppierungen	179
3.7.1	Spaltengruppen	180
3.7.2	Logische Gruppierung.....	181
3.8	Tabellen als Designmittel	183
3.9	Zusammenfassung	186
3.10	Fragen und Übungen	186

4 Grafiken und Multimedia 189

4.1	Grafikformate	189
4.1.1	GIF	190
4.1.2	JPEG.....	191

4.1.3	PNG.....	192
4.1.4	SVG	193
4.2	Grafiken einbinden.....	193
4.2.1	Alternativ-Text	195
4.2.2	Breite und Höhe.....	195
4.2.3	Rahmen	196
4.2.4	Grafiken ausrichten	197
4.3	Transparente Grafiken.....	197
4.3.1	Blind- oder Fake-GIFs	199
4.4	Hintergründe.....	199
4.5	Flash-Filme	201
4.6	JavaApplets.....	203
4.6.1	applet-Element	204
4.6.2	object-Element	205
4.7	Grafiken mittels object-Element	206
4.8	Zusammenfassung.....	209
4.9	Fragen und Übungen	209

5 Geknüpftes Netz 211

5.1	Aufbau einer Verknüpfung.....	211
5.2	Lokale Links	212
5.2.1	Gleiches Verzeichnis.....	212
5.2.2	Übergeordnete Verzeichnisse	213
5.2.3	Anderer Verzeichniszweig	213
5.3	Globale Links	213
5.4	E-Mail & Co.	214
5.4.1	Andere Webdienste	215
5.5	Grafiken als Links	216
5.6	Interne Verweise.....	216
5.7	Neue Fenster.....	217
5.8	Imagemaps.....	217
5.9	Zusammenfassung.....	221
5.10	Fragen und Übungen	221

6 Formulare 223

6.1	Was sind Formulare?	223
6.2	Aufbau eines Formulars	223
6.2.1	Das action-Attribut	224
6.2.2	Das method-Attribut.....	224

6.3	Eingabefelder.....	225
6.4	Schaltflächen	229
6.5	Wahlprogramm.....	232
6.6	Elemente gruppieren	236
6.7	Reihenfolge.....	237
6.8	Zusammenfassung	238
6.9	Fragen und Übungen	239

7 Rahmenprogramm 241

7.1	Was sind Frames?.....	241
7.2	Grundgerüst	242
7.3	Verschachtelung	247
7.4	Gestaltung der Frames.....	250
7.5	Anwendungsgebiete.....	252
7.6	Eingebettete Frames.....	253
7.7	Zusammenfassung	255
7.8	Fragen und Übungen	255

8 Was sonst noch wichtig ist 257

8.1	META-Tags	257
8.2	Automatische Weiterleitung.....	259
8.3	ActiveX-Controls.....	260
8.4	Trennlinien	261
8.5	Inoffizielles HTML.....	262
8.6	Goldene Regeln	264
8.7	XHTML – Die nächste Generation von HTML	266
8.7.1	Neu und doch altbekannt	266
8.7.2	Erforderliche Angaben	268

Teil 3 CSS – Layout fürs Internet

1 CSS – Layout fürs Internet 273

1.1	CSS – Make-up fürs Web	273
1.2	Wie funktioniert CSS?	274

1.3	CSS-Versionen	274
1.4	Direkte Formatierung.....	275
1.5	Das erste Stylesheet	276
1.6	Zentrale Formatierung	277
1.7	Externe Formatierung	281
1.8	Ausgabemedien	283
1.9	Einheiten.....	284
1.10	Zusammenfassung.....	285
1.11	Fragen und Übungen	286

2 Textformatierung 287

2.1	Farben in CSS.....	287
2.2	Schriftformatierung.....	290
2.2.1	Schriftart bzw. -typ	290
2.2.2	Schriftgröße	292
2.2.3	Schriftneigung und -variante	293
2.2.4	Schriftdicke	294
2.2.5	Schriftfarbe	296
2.2.6	Wort- und Zeichenabstände.....	296
2.2.7	Textdekoration.....	298
2.2.8	Texttransformation.....	299
2.2.9	Kurznotation zur Schriftformatierung.....	300
2.3	Schriftartendateien	301
2.4	Spezielle Formatierungen.....	305
2.4.1	Klassen	309
2.4.2	Verschachtelte Elemente	311
2.4.3	Individuelle Formate	313
2.5	Hintergrundfarbe	315
2.6	Hintergrundbilder	316
2.7	Zusammenfassung.....	319
2.8	Fragen und Übungen	320

3 Allgemeine Formatierung 321

3.1	Die Elemente div und span	321
3.2	Außenabstand	323
3.3	Innenabstand	325
3.4	Rahmen.....	328
3.4.1	Detaillierte Variante	329

3.5	Positionierung	331
3.6	Anzeige	333
3.7	Zusammenfassung	336
3.8	Fragen und Übungen	337

4 Tabellen und Listen 339

4.1	Tabellenformatierung	339
4.1.1	Tabellenlayout	339
4.1.2	Rahmenlayout	341
4.1.3	Ausrichtung der Überschrift	343
4.1.4	Rahmenabstand	345
4.1.5	Darstellung leerer Zellen	346
4.2	Listenformatierung	347
4.2.1	Aufzählungszeichen	347
4.2.2	Einrückung	350
4.2.3	Ein Bild als Aufzählungszeichen	351
4.2.4	Eigenschaften zusammenfassen	352
4.3	Zusammenfassung	354
4.4	Fragen und Übungen	354

5 Pseudoformate 355

5.1	Verweise	355
5.2	Absätze	357
5.3	Automatischer Text	358
5.4	Automatische Nummerierung	363
5.5	Zusammenfassung	366
5.6	Fragen und Übungen	366

6 Was sonst noch wichtig ist 369

6.1	Cursor	369
6.2	Scrollbar	371
6.3	Special effects	372
6.3.1	Alpha	372
6.3.2	Blur	373
6.3.3	Chroma	373
6.3.4	DropShadow	374

6.3.5	FlipH.....	374
6.3.6	FlipV.....	374
6.3.7	Glow.....	374
6.3.8	Gray.....	375
6.3.9	Invert.....	375
6.3.10	Mask.....	375
6.3.11	Shadow.....	375
6.3.12	Wave.....	376
6.3.13	XRay.....	376
6.3.14	Weitere Informationen.....	376
6.4	Printmedien	376
6.4.1	Medientyp festlegen	377
6.4.2	Seitengröße und -ränder	377
6.4.3	Seitenumbruch.....	378
6.4.4	Allein stehende Zeilen.....	378
6.4.5	Schnittmarken.....	378
6.4.6	Linke, rechte und erste Seite.....	379
6.5	Zusammenfassung.....	379
6.6	Fragen und Übungen	380

Teil 4 JavaScript

1 Grundlagen 383

1.1	Java und JavaScript	384
1.2	JavaScript in HTML	385
1.3	Das erste JavaScript.....	386
1.4	JavaScript in Dateien	389
1.5	Kommentare	390
1.6	Zusammenfassung.....	391
1.7	Fragen und Übungen	391

2 Variablen, Werte und Operatoren 393

2.1	Variablen.....	393
2.2	Werte.....	394
2.3	Operatoren	396
2.4	Einfacher Passwortschutz.....	400
2.5	Bedingte Anweisungen – if.....	401
2.6	Fallunterscheidung – switch	403
2.7	Zusammenfassung.....	405
2.8	Fragen und Übungen	405

3 Funktionen und Schleifen 407

3.1	Funktionen definieren	407
3.2	Funktionen aufrufen	408
3.3	Funktionen und Parameter	409
3.4	Rückgabewerte	411
3.5	Schleifen	412
3.5.1	Die while-Schleife	412
3.5.2	Die do-while-Schleife	413
3.5.3	Die for-Schleife	414
3.6	Zusammenfassung	415
3.7	Fragen und Übungen	416

4 Objektorientierung 417

4.1	Eigenschaften	418
4.2	Methoden	420
4.3	Objekthierarchie	421
4.4	Objekte instanzieren	421
4.5	Mehrere Anweisungen	424
4.6	Eigene Klassen	424
4.7	Zusammenfassung	427
4.8	Fragen und Übungen	427

5 Datums- und Zeitfunktionen 429

5.1	Formatierte Datumsausgabe	429
5.1.1	Arrays	429
5.1.2	Monatsnamen ermitteln	431
5.1.3	Wochentag ermitteln	431
5.1.4	Vierstellige Jahreszahl	432
5.1.5	Kombinierte Ausgabe	432
5.2	Führende Nullen	433
5.3	Besuchsdauer	435
5.4	Countdown	437
5.4.1	Verbesserungen	439

5.5	Datum validieren	439
5.5.1	Schaltjahr	439
5.5.2	Zahlenbereich	441
5.6	Das Jahr 2000	442
5.7	Zusammenfassung.....	443
5.8	Fragen und Übungen	444

6 Informationen vom Browser 445

6.1	Welcher Browser?.....	445
6.1.1	Detailinformationen	446
6.1.2	Browser identifizieren	449
6.2	Bildschirm	451
6.2.1	Farb- und Pixeltiefe.....	453
6.3	Plug-ins.....	455
6.3.1	Auf ein Plug-in prüfen	458
6.4	Zusammenfassung.....	460
6.5	Fragen und Übungen	461

7 Zeichenkettenfunktionen 463

7.1	Länge	463
7.2	Groß- und Kleinschreibung.....	464
7.3	Zeichenposition	464
7.4	Teilzeichen	466
7.5	Teilstring	467
7.5.1	Variante 1	467
7.5.2	Variante 2	468
7.6	String zerlegen	468
7.7	Zusammenfassung.....	469
7.8	Fragen und Übungen	469

8 DHTML 471

8.1	HTML wird dynamisch	471
8.2	Kompatibilität.....	472
8.3	Internet Explorer-DOM	472
8.4	Netscape-DOM.....	473
8.5	W3C-DOM	475

8.6	Problematik und Lösung	478
8.7	Ausklappbare Navigationsleiste	479
8.8	Zusammenfassung	481
8.9	Fragen und Übungen	482

9 Was sonst noch wichtig ist 483

9.1	Ereignisbehandlung	483
9.1.1	onAbort	484
9.1.2	onBlur	485
9.1.3	onChange	485
9.1.4	onClick	485
9.1.5	onDbClick	486
9.1.6	onError	486
9.1.7	onFocus	486
9.1.8	onKeyDown	486
9.1.9	onKeyPress	487
9.1.10	onKeyUp	487
9.1.11	onLoad	487
9.1.12	onMouseDown	487
9.1.13	onMouseMove	488
9.1.14	onMouseout und onMouseover	488
9.1.15	onMouseup	488
9.1.16	onReset und onSubmit	489
9.1.17	onResize	489
9.1.18	onSelect	489
9.1.19	onUnload	490
9.2	Cookies	490
9.2.1	Cookie schreiben	491
9.2.2	Cookie auslesen	491
9.2.3	Verfallsdauer	492
9.2.4	Persönliche Seitenbesuche zählen	493
9.3	Fehlerbehandlung	496
9.4	Zusammenfassung	500
9.5	Fragen und Übungen	500

Teil 5 Perl – Dynamisch und Interaktiv I

1 Perl 503

1.1	Perl ist nicht CGI	504
1.2	Das erste Perl-Script	504
1.3	Textausgabe	505

1.4	Die erste Zeile.....	506
1.5	Notwendige Anweisungen.....	508
1.6	Kommentare und Quellstrukturierung.....	508
1.7	Zusammenfassung.....	509
1.8	Fragen und Übungen	510

2 Variablen und Operatoren 511

2.1	Skalare	511
2.2	Arrays.....	513
2.2.1	Zugriff auf einzelne Elemente	514
2.3	Hashes	515
2.3.1	Zugriff auf einzelne Elemente	516
2.3.2	Unbekannte Schlüssel ermitteln	517
2.4	Vordefinierte Variablen	518
2.5	Operatoren	519
2.5.1	Zuweisungsoperator	519
2.5.2	Berechnungsoperatoren	519
2.5.3	Vergleichsoperatoren	521
2.5.4	Logische Operatoren	522
2.5.5	Zeichenkettenoperator	523
2.6	Zusammenfassung.....	524
2.7	Fragen und Übungen	524

3 Subroutinen 525

3.1	Was sind Subroutinen?.....	525
3.2	Subroutinen erstellen.....	525
3.3	Subroutinen aufrufen	526
3.4	Local und my.....	527
3.4.1	Doppeldefinition	530
3.5	Parameter übergeben.....	531
3.5.1	Elegantere Methode.....	532
3.6	Rückgabewerte	533
3.7	Zusammenfassung.....	535
3.8	Fragen und Übungen	535

4 Ablaufsteuerung 537

4.1	If und unless	537
4.1.1	elsif	538
4.1.2	unless	539
4.2	do-Schleife	539
4.2.1	Variante mit while	540
4.3	for-Schleife	541
4.4	while-Schleife	541
4.4.1	while und Hashes	542
4.5	foreach-Schleife	543
4.5.1	Spezielle Notation	544
4.6	Schleifensteuerung	545
4.6.1	Durchlauf wiederholen	545
4.6.2	Schleife abbrechen	545
4.6.3	Durchlauf überspringen	546
4.7	Zusammenfassung	546
4.8	Fragen und Übungen	546

5 Standardfunktionen 547

5.1	Zeichenkettenfunktionen	547
5.1.1	Groß- und Kleinbuchstaben	547
5.1.2	Zeichenwerte	548
5.1.3	Länge einer Zeichenkette	549
5.1.4	Teilzeichenkette suchen	549
5.1.5	Teilzeichenkette extrahieren	550
5.1.6	Zeichenkette verschlüsseln	551
5.2	Hash- und Listenfunktionen	551
5.2.1	Wert löschen	551
5.2.2	Werte und Schlüssel eines Hashs	552
5.2.3	Elemente aus einem Hash entfernen	553
5.2.4	Überprüfen, ob ein Element existiert	553
5.2.5	Alle Elemente eines Hashs ausgeben	554
5.2.6	Element am Anfang einer Liste hinzufügen oder löschen	554
5.2.7	Element am Ende einer Liste hinzufügen oder löschen	555
5.2.8	Liste sortieren oder umkehren	556
5.2.9	Liste in Zeichenkette konvertieren und umgekehrt	556
5.2.10	Zeichenkettenfunktion auf alle Listenelemente anwenden	557
5.3	Datums- und Zeitfunktionen	558
5.4	Mathematische Funktionen	562
5.4.1	Zufallszahlen	563
5.4.2	Sinus und Kosinus	564
5.4.3	Wurzelberechnung	565

5.5	Umwandlungsfunktionen.....	565
5.5.1	Hexadezimal in dezimal.....	565
5.5.2	Oktalzahl in dezimal.....	566
5.5.3	Zeichenkette in Zahlenwert umwandeln.....	567
5.5.4	Absoluter Zahlenwert.....	568
5.6	Zusammenfassung.....	568
5.7	Fragen und Übungen.....	568

6 Ein- und Ausgabe 571

6.1	Parameterübergabe.....	571
6.1.1	Modularisierung.....	575
6.2	Formulare.....	575
6.2.1	Daten dekodieren.....	579
6.3	Cookies.....	581
6.3.1	Ein Cookie schreiben.....	583
6.3.2	Ein Cookie lesen.....	584
6.4	Zusammenfassung.....	585
6.5	Fragen und Übungen.....	585

7 Dateisystem 587

7.1	Verzeichnisliste.....	587
7.2	Rekursion.....	590
7.3	Weitere Verzeichnisfunktionen.....	596
7.3.1	Verzeichnis wechseln.....	596
7.3.2	Verzeichnis erstellen.....	596
7.3.3	Verzeichnis löschen.....	597
7.4	Datei lesen.....	597
7.4.1	Praxisanwendung.....	598
7.5	Datei schreiben.....	600
7.6	Datei-Uploads.....	605
7.7	Zusammenfassung.....	608
7.8	Fragen und Übungen.....	609

8 Reguläre Ausdrücke 611

8.1	Was sind reguläre Ausdrücke?.....	611
8.2	Reguläre Ausdrücke verwenden.....	611
8.3	Metazeichen.....	613

8.4	Quantifier	615
8.5	Gruppierung	617
8.6	Zeichenklassen	617
8.6.1	Abkürzungen	618
8.7	Alternativen	619
8.8	Flags	619
8.9	Suchen und Ersetzen	620
8.10	Zusammenfassung	621
8.11	Fragen und Übungen	621

9 Was sonst noch wichtig ist 623

9.1	CGI und SSI	623
9.1.1	Ausgabe von Variablen	623
9.1.2	Konfigurationen	625
9.1.3	Programm ausführen	627
9.2	CGI-Umgebungsvariablen.....	629
9.3	Zusammenfassung	631
9.4	Fragen und Übungen	631

Teil 6 PHP – Dynamisch und Interaktiv II

1 PHP 635

1.1	Das erste PHP-Script	636
1.2	PHP und die Dateiendungen	637
1.3	PHP in eigenen Dateien.....	638
1.4	Quellstrukturierung und Kommentare.....	639
1.5	Textausgabe.....	639
1.5.1	Formatierte Textausgabe.....	640
1.6	Alternative PHP-Tags.....	642
1.7	Zusammenfassung	642
1.8	Fragen und Übungen	643

2 Variablen und Operatoren 645

2.1	Variable.....	645
2.1.1	Variablen bezeichnen	645
2.2	Datentypen	646
2.2.1	Integer	646
2.2.2	Fließkommazahl	647
2.2.3	Boolean	647
2.2.4	String	648
2.2.5	Nützliche Funktionen	649
2.2.6	Typumwandlung	650
2.3	Arrays.....	651
2.4	Resource und NULL	652
2.5	Operatoren	653
2.5.1	Arithmetische Operatoren.....	653
2.5.2	Vergleichsoperatoren	655
2.5.3	Zeichenkettenoperatoren	656
2.5.4	Logische Operatoren	656
2.6	Zusammenfassung.....	657
2.7	Fragen und Übungen	658

3 Programmsteuerung 659

3.1	if-Bedingung	659
3.2	do...while-Schleife	661
3.3	while-Schleife	662
3.4	for-Schleife.....	662
3.5	foreach-Schleife	663
3.6	switch-Anweisung	665
3.7	Einbinden von Dateien	667
3.8	Schleifen steuern	668
3.9	Alternative Syntax.....	669
3.9.1	if.....	669
3.9.2	while.....	670
3.9.3	for.....	670
3.9.4	foreach	670
3.9.5	switch	671
3.10	Zusammenfassung.....	671
3.11	Fragen und Übungen	672

4 Objekte und Klassen 673

4.1	Objekte	673
4.2	Funktionen	673
4.3	Klassen	676
4.4	Objekte instanzieren	677
4.5	Konstruktoren	678
4.6	Vererbungslehre	679
4.7	Zusammenfassung	681
4.8	Fragen und Übungen	681

5 Standardfunktionen 683

5.1	Zeichenkettenfunktionen.....	683
5.1.1	Groß- und Kleinschreibung	683
5.1.2	Zeichenwerte.....	684
5.1.3	Länge einer Zeichenkette	684
5.1.4	Teilzeichenkette suchen	685
5.1.5	Teilzeichenkette auslesen.....	686
5.1.6	Teilzeichenkette ersetzen.....	687
5.1.7	Zeichenkette verschlüsseln.....	687
5.2	Array-Funktionen.....	688
5.2.1	Größe eines Arrays.....	688
5.2.2	Arrays sortieren	688
5.2.3	Array mit Wertebereich	690
5.2.4	Element am Ende hinzufügen oder löschen.....	691
5.2.5	Elemente am Anfang hinzufügen oder löschen.....	692
5.2.6	Array durchsuchen.....	692
5.2.7	Schlüssel und Werte	693
5.3	Datums- und Zeitfunktionen	694
5.3.1	Der aktuelle Zeitpunkt	694
5.3.2	Ausgabe formatieren.....	696
5.3.3	Datum umwandeln.....	697
5.4	Mathematische Funktionen.....	698
5.4.1	Zufallsgenerator	698
5.4.2	Zahlensysteme	700
5.4.3	Sinus und Kosinus.....	700
5.5	Zusammenfassung	700
5.6	Fragen und Übungen	701

6 Ein- und Ausgabe 703

6.1	Parameterübergabe	703
6.2	Formulare	706
6.2.1	GET-Methode	706
6.2.2	POST-Methode	708
6.2.3	Checkboxen, Radiobuttons und Auswahllisten	709
6.3	Cookies	711
6.3.1	Cookie schreiben	711
6.3.2	Cookie lesen	712
6.4	Dateiupload	713
6.5	Zusammenfassung	717
6.6	Fragen und Übungen	717

7 Dateisystem 719

7.1	Verzeichnisliste	719
7.1.1	Verzeichnis oder Datei	720
7.1.2	Alternative	721
7.1.3	Weitere Verzeichnisfunktionen	722
7.2	Rekursion	722
7.3	Datei schreiben	724
7.4	Datei lesen	726
7.5	Dateieigenschaften	729
7.6	Dateisystemoperationen	731
7.7	Zusammenfassung	732
7.8	Fragen und Übungen	732

8 Sitzungen 733

8.1	Was sind Sessions?	733
8.2	Session erzeugen	734
8.3	Mit Variablen arbeiten	735
8.4	Session beenden	736
8.5	Weitere Session-Funktionen	737
8.6	Zusammenfassung	738
8.7	Fragen und Übungen	738

9 Was sonst noch wichtig ist 739

9.1	Server-Informationen.....	739
9.2	Netzwerkfunktionen.....	740
9.2.1	IP-Adressen und DNS.....	740
9.2.2	Verbindungen zu anderen Servern.....	740
9.2.3	HTTP-Verbindungen.....	742
9.2.4	POP3-Verbindung.....	743
9.3	Perl-kompatible reguläre Ausdrücke in PHP.....	744
9.4	PDF-Dokumente erzeugen.....	747
9.4.1	Dokument mit Inhalt füllen.....	748
9.5	Zusammenfassung.....	750
9.6	Fragen und Übungen.....	750

Teil 7 MySQL – Der Datenspeicher

1 MySQL – Der Datenspeicher für Ihre Internetseiten 753

1.1	Mein SQL gib mir heute	753
1.2	Datenbanktypen.....	754
1.2.1	Relationale Datenbanksysteme.....	754
1.2.2	Objektorientierte Datenbanksysteme.....	755
1.3	Redundanz und Inkonsistenz.....	755
1.4	Zusammenfassung.....	757
1.5	Fragen und Übungen.....	757

2 Die Sprache SQL 759

2.1	Einstieg.....	759
2.2	Namenskonventionen.....	760
2.3	Datenbank erstellen, löschen oder auswählen.....	761
2.4	Tabellen erstellen und löschen.....	762
2.5	Tabellen verändern.....	764
2.6	MySQL-Datentypen.....	765
2.6.1	Numerische Typen.....	765
2.6.2	Zeichen- und Zeichenketten-Typen.....	766
2.6.3	Vermischte Typen.....	766
2.6.4	Was ist ein BLOB?.....	767
2.6.5	Optionen.....	767

2.7	Zusammenfassung.....	768
2.8	Fragen und Übungen	768

3 Datenaustausch 771

3.1	SELECT.....	771
3.1.1	WHERE	774
3.1.2	ORDER BY	778
3.1.3	GROUP BY.....	779
3.1.4	DISTINCT.....	780
3.1.5	Numerische Ausdrücke	781
3.2	INSERT.....	782
3.3	UPDATE	784
3.4	DELETE	785
3.5	Tabellen importieren und exportieren.....	785
3.5.1	Export.....	785
3.5.2	Import	786
3.6	Tabellenverknüpfung	788
3.6.1	Aliase	789
3.6.2	JOIN	789
3.7	Zusammenfassung.....	790
3.8	Fragen und Übungen	791

4 MySQL und Perl 793

4.1	Vorbereitungen	793
4.2	Datenbankverbindung herstellen.....	794
4.3	Anfragen stellen.....	796
4.4	Ergebnisverarbeitung bei SELECT-Anweisungen.....	799
4.4.1	Datensätze lesen	800
4.5	Ergebnisverarbeitung bei Anweisungen, die keine Ergebnismenge liefern	803
4.6	Datentypen	806
4.6.1	Zeichenketten	806
4.6.2	Datums- und Zeitangaben	807
4.7	Fehlerbehandlung	808
4.8	Metainformationen	810
4.9	Zusammenfassung.....	811
4.10	Fragen und Übungen	811

5 MySQL und PHP 813

5.1	Datenbankverbindung herstellen	813
5.2	Datenbank auswählen	815
5.2.1	Konfigurationsdatei.....	816
5.3	Anfragen stellen	817
5.4	Ergebnisverarbeitung bei SELECT oder SHOW	819
5.5	Ergebnisverarbeitung bei anderen Anweisungen.....	821
5.6	Datentypen.....	822
5.6.1	Zeichenketten.....	822
5.6.2	Datums- und Zeitangaben	823
5.7	Fehlerbehandlung.....	824
5.8	Metainformationen.....	825
5.8.1	Datenbanken	826
5.8.2	Tabellen.....	826
5.8.3	Felder bzw. Spalten	827
5.9	Zusammenfassung	829
5.10	Fragen und Übungen	829

Teil 8 Dynamische Bildgenerierung mit PHP, Perl und SVG

1 Dynamische Bildgenerierung – PHP-Variante 833

1.1	Die GD-Library und PHP.....	833
1.2	Eine erste dynamische Grafik.....	834
1.2.1	Farben	834
1.2.2	Text einfügen.....	835
1.2.3	Ausgabe des Bildes	836
1.3	Grafiken verändern	838
1.3.1	Abmessungen	839
1.4	Zeichnen	841
1.4.1	Punkte und Linien	841
1.4.2	Rechtecke.....	842
1.4.3	Kreise und Kreisbögen.....	843
1.4.4	Flächen füllen.....	846
1.4.5	Transparenz.....	847
1.5	Erweiterte Textausgabe	849
1.6	Thumbnails erzeugen.....	851
1.7	Anwendungsbeispiele.....	856
1.7.1	Kreisdiagramm.....	856
1.7.2	Zeichensalat.....	861

1.8	Zusammenfassung.....	865
1.9	Fragen und Übungen	866

2 Dynamische Bildgenerierung – Perl-Variante 867

2.1	Die GD-Library und Perl.....	867
2.1.1	Manuelle Installation.....	869
2.2	Erste Versuche	870
2.2.1	Farben	871
2.2.2	Text einfügen	871
2.2.3	Ausgabe des Bildes.....	873
2.3	Grafiken verändern.....	877
2.4	In einer Grafik zeichnen	879
2.4.1	Punkte und Linien	879
2.4.2	Rechtecke	881
2.4.3	Kreise und Kreisbögen	882
2.4.4	Flächen füllen	885
2.4.5	Transparenz	887
2.5	Erweiterte Textausgabe.....	889
2.6	Thumbnails erzeugen	892
2.7	Anwendungsbeispiele.....	896
2.7.1	Verlaufdiagramm	896
2.7.2	Grafik als Textdatei.....	903
2.8	Zusammenfassung.....	908
2.9	Fragen und Übungen	908

3 Dynamische Bildgenerierung mit SVG – Scalable Vector Graphics 909

3.1	Was sind Vektoren.....	909
3.2	Das Wichtigste zu XML und SVG.....	911
3.3	Das erste SVG-Dokument	912
3.4	SVG in HTML	914
3.5	Grundformen.....	915
3.5.1	Text ausgeben	915
3.5.2	Linien.....	917
3.5.3	Rechtecke	918
3.5.4	Kreise und Ellipsen.....	920
3.6	Farben.....	921
3.6.1	Transparenz	921
3.6.2	Lineare Farbverläufe.....	921
3.6.3	Radialer Farbverlauf	923

3.7	Pfade	924
3.7.1	Einfache Pfade	924
3.7.2	Kurven	926
3.7.3	Kreisbögen	927
3.8	SVG mit PHP.....	931
3.9	Zusammenfassung	937
3.10	Fragen und Übungen	937

Teil 9 Workshop – Erstellung eines Content Management Systems

1 Workshop – Erstellung eines Content Management Systems 941

1.1	Das Ziel.....	941
1.2	Wählen Sie jetzt!	944
1.3	Strukturierung	945
1.3.1	Verzeichnisschutz	946
1.4	Zusammenfassung	949

2 Die Datenbank 951

2.1	Datenbankstruktur planen und festlegen	951
2.2	Tabellenstruktur planen und erzeugen	951
2.2.1	Planung	951
2.2.2	Erzeugen	955
2.3	Erste Einträge anlegen.....	956
2.4	Zusammenfassung	956

3 Das Design 957

3.1	Die Struktur.....	957
3.2	HTML-Vorlage	958
3.3	Zusammenfassung	960

4 Die Programmierung 961

4.1	Die Konfigurationsdateien.....	961
4.1.1	base.inc.php	961
4.1.2	database.inc.php	961
4.1.3	Sicherheit	961
4.1.4	functions.inc.php.....	962

4.2	Die index.php	962
4.3	Die functions.inc.php.....	966
4.4	Die base.inc.php	968
4.5	Die database.inc.php	969
4.6	Das Projekt im Einsatz	969

5 Der Schluss 971

5.1	Daten pflegen	971
5.2	Verbesserungsmöglichkeiten	972
5.3	Projekt installieren	972

Schlusswort 975

Anhang

A Antworten und Lösungen 977

A.1	Teil 2 HTML	977
A.2	Teil 3 CSS – Layout fürs Internet.....	991
A.3	Teil 4 JavaScript	997
A.4	Teil 5 Perl – Dynamisch und interaktiv I	1006
A.5	Teil 6 PHP – Dynamisch und interaktiv II.....	1019
A.6	Teil 7 MySQL – Datenspeicher.....	1033
A.7	Teil 8 Dynamische Bildgenerierung mit PHP, Perl, und SVG	1043

B Referenztabellen 1051

B.1	ASCII-Zeichentabelle.....	1051
B.2	Zahenumrechnungstabelle	1053
	B.2.1 Binärsystem.....	1054
	B.2.2 Oktalsystem	1055
	B.2.3 Hexadezimalsystem	1056
B.3	Farbworte	1058
B.4	Websichere Farben	1061
B.5	MIME-Typen.....	1062
B.6	Sprachenkürzel.....	1065
B.7	MySQL-Fehlernummern.....	1067

C Reservierte Wörter 1077

C.1 JavaScript 1077
C.2 Perl 1078
C.3 PHP..... 1080

D Installationsanleitungen 1083

D.1 Microsoft Personal Webserver 1083
D.1.1 Den Server installieren 1083
D.1.2 PHP installieren 1084
D.1.3 Perl installieren 1085
D.1.4 MySQL installieren..... 1085
D.2 Microsoft Internet Information Server..... 1086
D.2.1 Den Server installieren 1086
D.2.2 PHP installieren 1087
D.2.3 Perl installieren 1088
D.2.4 MySQL installieren..... 1089
D.3 OmniHTTP 1089
D.3.1 Den Server installieren 1089
D.3.2 Perl installieren 1091
D.3.3 MySQL installieren..... 1091
D.4 Xitami-Webserver..... 1092
D.4.1 Den Server installieren 1092
D.4.2 PHP installieren 1093
D.4.3 Perl installieren 1094
D.4.4 MySQL installieren..... 1095
D.5 MySQL..... 1095

E Weitere Informationen 1099

E.1 WWW-Links 1099
E.2 Buch-Tipps..... 1099
E.3 Quellenverzeichnis 1100

F Inhalt der CD-ROM 1101

G Glossar 1103

Index 1115

Teil 1

Grundlagen

5 Der Weg ins Internet

Suche nicht das Abenteuer, aber gehe ihm nicht aus dem Weg.

– Fernöstliche Weisheit

In diesem Kapitel soll nicht das Abenteuer beschrieben werden, wie Sie ins Internet kommen, sondern, wie Sie den richtigen Provider wählen, Ihre Seite auf einen Server im Internet hochladen und sie den Suchmaschinen bekannt machen.

5.1 Wahl des Providers

Die Wahl eines Providers gestaltet sich häufig sehr kompliziert, da es viele Dinge gibt, die dabei zu beachten sind. Vor allem, da sich unter den vielen Anbietern auch einige schwarze Schafe tummeln, die es zu erkennen gilt. Die meisten Anbieter locken mit allerhand Versprechungen, umfangreichen Leistungsmerkmalen und günstigen Preisen, lassen dabei scheinbar nebensächliche Kosten geschickt außen vor. Ich möchte Ihnen nun einige Anhaltspunkte nennen, wie Sie die Guten von den Schlechten unterscheiden können, und wie Sie den richtigen Provider für Ihr Projekt finden.

Die Entscheidungsfindung lässt sich anhand von kurzen Fragen sehr erleichtern:

- ▶ Welche Inhalte sollen angeboten werden?
- ▶ Welche Technologien sollen dabei verwendet werden?
- ▶ Was ist für die Zukunft des Projektes an Erweiterungen geplant, und welche Technologien sind dafür notwendig?
- ▶ Wie hoch ist der Administrationsaufwand?
- ▶ Welche Möglichkeiten benötigen Sie zur Administration?

Fünf Fragen

Dass Sie diese Fragen nicht unbedingt sofort beantworten können, ist nicht weiter problematisch. Die Antworten basieren größtenteils auf Erfahrungswerten.

Die Frage der Inhalte ist wichtig, um herauszufinden, wie viel Speicherplatz auf dem Server des Internet Service Providers (kurz ISP) benötigt wird. Möchten Sie lediglich wenige Informationen über sich selbst preisgeben, wird der benötigte Speicherplatz nicht mehr als 1 MB umfassen.

Welche Inhalte?

Möchten Sie hingegen die gesammelten Bilder Ihrer Urlaube der letzten 20 Jahre in bestmöglicher Qualität zur Verfügung stellen, kann die Anforderung an den Speicherplatz schnell in Richtung mehrerer hundert Megabyte gehen. Selbst die Bereitstellung von Downloads, wie Software, Musikstücke oder Videos, kommt nicht ohne genügend Speicherplatz aus. Im Voraus zu sagen, wie viel Megabyte Sie brauchen werden, ist schwer. Man kann hier nur von ungefähren Werten ausgehen, die es zu überschlagen gilt, um wiederum einen ungefähren Wert für den Speicherverbrauch zu ermitteln.

Art	Speicherplatz
HTML-Seite mit kleinen Grafiken zur optischen Gestaltung	ca. 20–100 KB
Grafik, 320 * 200	Je nach Qualität ca. 9–85 KB
Grafik, 640 * 480	Je nach Qualität ca. 25–300 KB
Grafik, 1024 * 768	Je nach Qualität ca. 45–700 KB
Software	Je nach Art ab ca. 100 KB
Musikstücke	Je nach Länge ab ca. 3 MB
Videos	Je nach Qualität ab ca. 5 MB

Tabelle 5.1 Schätzwerte für Medien

Wie Sie der Tabelle 5.1 entnehmen können, ist alles, was über eine einfache HTML-Seite hinausgeht, salopp gesagt, Speicher fressend.

Platz für Erweiterungen?

Außerdem sollten Sie immer bedenken, dass Sie vielleicht nach und nach neue Inhalte hinzufügen möchten. Auch diese kommen ohne Speicherplatz nicht aus, und deshalb sollten auch diese Erweiterungen schon von vornherein in die Planung mit einbezogen werden. Wenn Sie eine ungefähre Summe ermittelt haben, können Sie noch gut und gern 50 Prozent aufschlagen. Dies sollte dann für kommende Erweiterungen ausreichend sein. Sind die Erweiterungen umfangreicher, kommen Sie in der Regel mit den zur Verfügung stehenden Mitteln nicht aus und müssen bei Ihrem Webhosting-Paket aufrüsten.

Welche Technologien?

Wichtig ist natürlich auch, welche Technologien Sie bei Ihrem Projekt einsetzen möchten, da nicht jeder ISP alles in einem Paket anbietet. So müssen Sie überlegen, ob Sie PHP, Perl, MySQL oder auch SSI einsetzen möchten. Vielleicht sollen ja auch ganze Videostreams abrufbar sein. Auf welche dieser Techniken können Sie verzichten und Ihr Projekt anhand

dieser Änderungen anpassen? Auf welche Techniken sind Sie zwingend angewiesen, ohne die Ihr Projekt nicht zu realisieren ist? Welche zukünftigen Erweiterungen des Projektes planen Sie? Welche Techniken werden dafür benötigt? All diese Fragen sind bei einer gründlichen Planung nicht sonderlich schwer zu beantworten. Da diese Fragen speziell vom Projekt abhängen, ist es hier schwer, Schätzungen vorzunehmen. Vermeiden Sie aber den Mix von gleichartigen Technologien, wie den von Perl und PHP. Dies erleichtert Ihnen die Realisierung Ihres Projekts und das Finden des geeigneten Providers enorm.

Abhängig von den Inhalten sind auch der Administrationsaufwand und die dafür erforderlichen Mittel. Bei ein paar einfachen HTML-Seiten ist es nicht weiter schlimm, wenn Sie Ihre Daten mittels eines HTML-Formulars des ISP übertragen müssen, auch wenn dies dann nur Datei für Datei möglich ist. Bei Projekten, die mehr als zehn Dateien umfassen, ist ein FTP-Zugang jedoch Pflicht. Letzteren bieten die meisten ISPs schon bei den kleinsten Paketen an. Bei Verwendung von Servertechnologien wie passwortgeschützten Verzeichnissen ist es oft sehr mühsam, die dafür benötigten Dateien auf dem eigenen Rechner von Hand anzulegen und dann zu übertragen. Ein Hilfsmittel auf Seiten des ISP bzw. des Servers wäre da schon angebrachter. Das Gelbe vom Ei ist dann aber ein so genannter SSH- oder Telnet-Zugang, mit dem Sie bequem alle administrativen Aufgaben am Server erledigen können.

Administration

Unterschätzt wird oft die Frage des Datentransferaufkommens, neu-deutsch auch Traffic genannt. Der Traffic umfasst die Menge der Daten, die zwischen einem Benutzer und einem Server transportiert wird. Ruft ein Benutzer Ihre Webseite auf und sieht sich dabei einige Seiten mit verschiedenen Grafiken an, kann es passieren, dass er schnell mal 500 bis 1.000 KB an Daten vom Server abrufen. Dies ist der Traffic. Pro Monat steht Ihnen als Anbieter nur eine begrenzte Menge an Traffic zur Verfügung. Bei einem zur Verfügung stehenden Volumen von 15 GB werden viele den Traffic außer Acht lassen und der Meinung sein, dass dies vollkommen ausreichend sei. Stellen Sie sich nun einmal vor, pro Besuch werden 1 MB Daten übertragen. Dann können 15.000 Aufrufe pro Monat stattfinden, ohne dass dieses Volumen überschritten wird. Werden jedoch mehr als 15 GB pro Monat übertragen, kann Sie das teuer zu stehen kommen, denn dann kostet ein zusätzliches Megabyte Traffic um die 1 bis 10 Cent. Auf 1 GB hochgerechnet macht das schnell 10 bis 100 €.

Datenverkehr

Wenn alle diese Fragen beantwortet sind, ist nun die Auswahl eines geeigneten Providers zu treffen. Wie bereits erwähnt, tummeln sich dort

Provider finden

einige schwarze Schafe. Aus diesem Grund sollten einige Dinge genau geprüft werden:

- ▶ Prüfen Sie die Angebote mit Sorgfalt. Viele Provider nennen Nebenkosten oft nur in Fußnoten. Achten Sie dabei auf das zur Verfügung stehende Transfervolumen, und rechnen Sie aus, wie teuer ein weiteres Gigabyte Traffic bei Überschreitung ist.
- ▶ Lesen Sie sich immer die allgemeinen Geschäftsbedingungen (AGB) durch, und achten Sie dabei auf Zahlungsmodalitäten, Garantiezusagen und Kündigungsfristen. Sie sollten nicht mehr als sechs Monate im Voraus zahlen müssen und maximal eine Kündigungsfrist bis zum letzten Tag des Folgemonats akzeptieren. Wichtig ist auch, welche Rechte Sie besitzen, wenn die Server, auf denen Ihre Seite gehostet wird, einmal ausfallen sollten. Für Privatperson ist dies vielleicht weniger wichtig als für ein Einzelhandelsunternehmen, welches die Seite als Vertriebskanal verwendet.
- ▶ Suchen Sie auf den Webseiten des Anbieters immer nach dem Impressum. Ein Unternehmen, das nichts zu verbergen hat, wird im Impressum auch Angaben zu Kontaktpersonen bei Rechtsstreitigkeiten machen. Seien Sie aufmerksam bei Firmen, deren Impressum keine detaillierten Angaben enthält.
- ▶ Versuchen Sie im Internet weitere Informationen zu dem ISP zu erhalten. Besuchen Sie Meinungsforen oder -plattformen wie Dooyoo (<http://www.dooyoo.de/>) oder Ciao (<http://www.ciao.de/>). Nehmen Sie gegebenenfalls auch kostenpflichtige Informationssysteme in Anspruch.
- ▶ Häufig lohnt sich ein Besuch auf der Webseite <http://www.webhostlist.de/>. Dort können Sie Anbieter suchen, sich aber auch Erfahrungsberichte von anderen Usern durchlesen.
- ▶ Sprechen Sie notfalls Kunden an, die ihre Webseite bei dem entsprechenden ISP hosten lassen, und fragen Sie nach deren Meinung.
- ▶ Bietet der ISP telefonischen Support an? Dann achten Sie auf teure 0190-Rufnummern. Diese sind für den ISP oft eine Goldgrube. Ein normaler Festnetzanschluss ist angebrachter und auch eine 0180-Rufnummer ist akzeptabel. Außerhalb der Geschäftszeiten sollte jedoch auf jeden Fall ein kostenloser E-Mail-Support zur Verfügung stehen.
- ▶ Preiserhöhungen sollten Sie vertraglich niemals akzeptieren. Sollte der ISP einmal die Preise erhöhen, muss Ihnen ein Sonderkündigungsrecht eingeräumt werden.

- ▶ Achten Sie ebenfalls darauf, dass die Domain auf Ihren Namen bei der Denic oder dem zuständigen NIC eingetragen wird. Andernfalls können ISPs eine sehr hohe Ablösesumme (Beträge von 100 € sind nicht unüblich) verlangen, falls Sie einmal zu einem anderen ISP wechseln möchten.

Die meisten ISPs bieten die verschiedensten Pakete an. Dabei gibt es neben den Unterschieden zwischen den angebotenen Leistungen bezüglich Speicherplatz, E-Mail-Adresse und Traffic noch den Unterschied zwischen virtuellem Hosting und dedizierten Servern. Beim virtuellen Hosting teilen Sie sich die Bandbreite des Servers mit anderen Mietern. Dies führt häufig dazu, dass der Anschluss des Servers an das Internet ein Flaschenhals wird, und Ihre Seiten nicht immer schnell und gut erreichbar sind. Dies ist immer abhängig davon, wie stark die Seiten Ihrer Mitmieter besucht werden. Bei einem dedizierten Server verfügen Sie über einen eigenen vollständigen Server in der Farm des ISP, und Ihnen stehen häufig viel mehr Möglichkeiten zur Administration und Anpassung des Servers nach eigenem Gusto bereit. Dabei gibt es auch den Unterschied zwischen vorkonfigurierten Servern (für Einsteiger mit wenig Kenntnissen optimal) und nicht konfigurierten Servern (für Profis mit viel Know-how bei der Konfiguration eines solchen Servers). Dedizierte Server sind häufig auch wesentlich teurer als das virtuelle Hosting. Wofür Sie sich schlussendlich entscheiden, hängt vom Geldbeutel und natürlich dem Umfang des Projekts ab.

Virtueller Server?

Zum Schluss müssen Sie sich natürlich noch für einen geeigneten Domainnamen entscheiden. In erster Linie sollten Sie natürlich aufgrund des Wiedererkennungswerts Namen, die Ihrem eigenen oder dem Ihrer Firma entsprechen, überprüfen. Meistens haben Sie durch geschicktes Setzen von Gedankenstrichen die Möglichkeit, doch noch einen Domainnamen nach Ihren Wünschen zu finden. Sollte dies nicht erfolgreich sein, können Sie natürlich auch ausländische Endungen testen, wie z. B. AT, CH oder auch Endungen wie NET, ORG und COM. Überprüfen Sie auch die neuen Endungen wie FIRM oder INFO. Immer beliebter werden Namen wie bildform.at, festpara.de oder glueckli.ch, bei denen die Domainendung als Teil des Domainnamens verwendet wird. Viele ISPs, etwa 1&1 Puretec (<http://www.puretec.de/>), bieten auch Namensratgeber an, die Ihnen aus bestimmten Vorgabewerten Namen generieren und gleichzeitig überprüfen, ob diese noch verfügbar sind. Alternativ können Sie auch die Auktionsplattform Ebay (<http://www.ebay.de/>) oder die Domainbörse Sedo (<http://www.sedo.de/>) besuchen, auf denen Domainnamen versteigert werden.

Domain finden

5.1.1 AOL, T-Online & Co.

Viele Anbieter von Internetzugängen, z. B. AOL oder T-Online, bieten Ihren Kunden kostenlosen Speicherplatz für eine eigene Homepage. Dieser Speicherplatz liegt in der Regel zwischen 5 bis 15 MB und ist für Kunden, die ein größeres Projekt planen, natürlich nicht annähernd ausreichend.

Serverseitiges Scripting

Außerdem ist die Verwendung von serverseitigen Scriptsprachen in der Regel ausgeschlossen, sodass Sie Perl, PHP oder andere Sprachen nicht einsetzen können. JavaScript hingegen – da es eine clientseitige Scriptsprache ist – funktioniert. Wenn Sie also erst einmal mit einem kleinen Projekt, bestehend aus ein paar HTML-Seiten und/oder JavaScript, anfangen möchten, können Sie dieses Angebot auf jeden Fall nutzen. Für anspruchsvollere Projekte – um die es in diesem Buch hauptsächlich geht – ist jedoch ein Paket bei einem »professionellen« ISP nicht nur zu empfehlen, sondern dringend notwendig.

5.2 Upload der Seite

Sobald Sie einen ISP gefunden haben, der Ihrem Geschmack entspricht, muss die Webseite natürlich auch ihren Weg auf den Server finden. Dies geschieht in der Regel mittels eines FTP-Clients. Es gibt dabei zwei Varianten: grafische und kommandozeilenbasierte Clients.

5.2.1 Grafischer Client unter Windows



Auf der dem Buch beiliegenden CD-ROM finden Sie im Verzeichnis `x:\extras\win32\wsftp32.exe` einen grafischen FTP-Client namens WS_FTP LE. Für Privatanwender ist dieser Client kostenlos. Firmen müssen jedoch eine Lizenz erwerben.

Installation

Bevor Sie das Programm nutzen können, müssen Sie es zuerst installieren. Die Installation gestaltet sich sehr einfach. Gleich zu Beginn gelangen Sie nach einem Klick auf die Schaltfläche **Continue** zum nächsten Bildschirm. Wenn Sie Privatanwender sind, markieren Sie die Option »Other« und klicken auf **Next ->**. Markieren Sie die Optionen »At home« und »For personal use«, und bestätigen Sie die Auswahl mit **Next ->**. Es erscheint nun der Endbenutzerlizenzvertrag, den Sie sich in Ruhe durchlesen sollten und dann mit einem Klick auf **Accept** akzeptieren können. Im nächsten Bildschirm können Sie nun das Installationsverzeichnis des Programms auswählen. Standardmäßig ist dies `C:\Program Files\WS_FTP`. Egal, wie Sie sich entscheiden, mit einem Klick auf die Schaltfläche **OK**

müssen Sie noch ein Verzeichnis angeben, das standardmäßig für Datei-transfers ausgewählt wird. Dies sollten Sie auf das Verzeichnis setzen, in dem sich die Dateien Ihrer Seite befinden. Durch einen weiteren Klick auf **OK** wird die Installation gestartet. Wenn alle Dateien kopiert wurden, fragt das Installationsprogramm noch nach einem Ordner im Startmenü, in dem die Verknüpfungen abgelegt werden sollen. Bestätigen Sie das Verzeichnis zweimal mit **OK**, um die Installation abzuschließen.

Nachdem die Installation erfolgreich beendet wurde, wird der Ordner des Startmenüs geöffnet. Klicken Sie auf die Verknüpfung **WS_FTP95 LE**. Dadurch wird WS FTP gestartet.

WS_FTP LE
starten

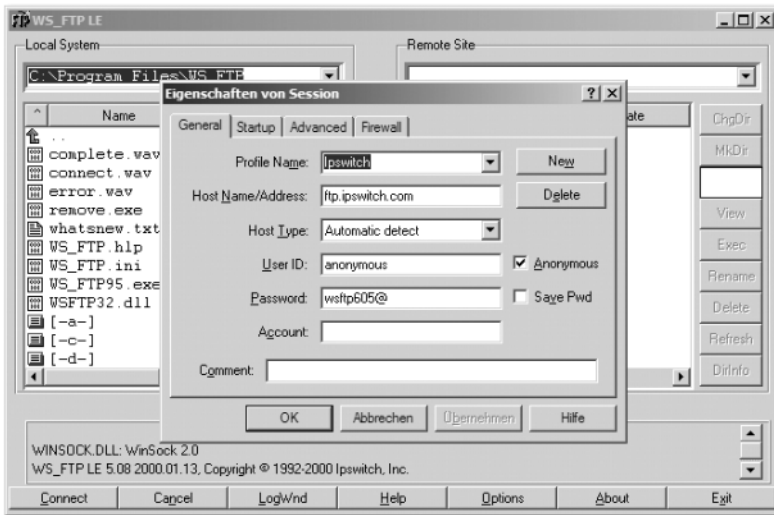


Abbildung 5.1 WS FTP 95 LE

In Abbildung 5.1 sehen Sie den typischen Startbildschirm von WS FTP. Hier können Sie neue Profile anlegen oder ein vorhandenes verwenden und eine Verbindung aufbauen. Als Profile werden in WS FTP Zusammenstellungen von Daten bezeichnet, welche die Adresse eines FTP-Servers, den Benutzernamen und das Passwort enthalten.

Mit einem Klick auf die Schaltfläche **New** wird ein neues Profil angelegt. Von Ihrem ISP sollten Sie Zugangsdaten für den FTP-Zugriff erhalten haben. Nehmen Sie diese nun zur Hand. Die Tabelle 5.2 enthält eine Übersicht über die einzelnen Felder und die Werte, die dort einzutragen sind.

Neues Profil
anlegen

Feld	Eingabe
Profile Name	Name des Profils. Kann von Ihnen frei gewählt werden.
Host Name/Address	Die IP-Adresse oder der Hostname des FTP-Servers.
Host Type	Der Typ des FTP-Servers. Sollten Sie dies nicht wissen, wählen Sie Automatic aus.
User ID	Die Benutzerkennung Ihres Zugangs.
Password	Das Passwort Ihres Zugangs.
Account	Kann in der Regel freigelassen werden.
Comment	Hier können Sie einen Kommentar Ihrer Wahl eingeben.
Anonymous	Entfernen Sie das Häkchen, da Sie sich nicht anonym auf dem Server anmelden möchten.
Save Pwd	Wenn Sie das Häkchen setzen, müssen Sie nicht bei jedem Verbindungsaufbau das Passwort des Zugangs eingeben. Aus Sicherheitsgründen sollten Sie es entfernen.

Tabelle 5.2 Eingabefelder von Profilen

Der Wert, den Sie im Feld »Host Name« eingeben müssen, entspricht bei den bekannten ISPs, wie z. B. Puretec oder Strato, in der Regel dem Domainnamen, den Sie für Ihre Seite haben registrieren lassen. Sobald alle Eingaben vorgenommen wurden, klicken Sie auf **Übernehmen**, um die Daten zu speichern.

Verbindung aufbauen

Nun können Sie mit einem Klick auf die Schaltfläche **OK** eine Verbindung zum Server aufbauen.

Die Oberfläche von WS FTP unterteilt sich bei erfolgreicher Verbindung mit einem FTP-Server in drei größere Bereiche. Im unteren Teil befindet sich ein Ausgabefeld, in dem alle Meldungen angezeigt werden. Das linke Fenster mit der Überschrift Local System zeigt den Inhalt Ihres Computers an, das rechte Fenster mit der Überschrift Remote Site den Inhalt des Servers. Zu jedem Fenster gibt es rechts daneben einen Bereich mit acht Schaltflächen und einem Eingabefeld.

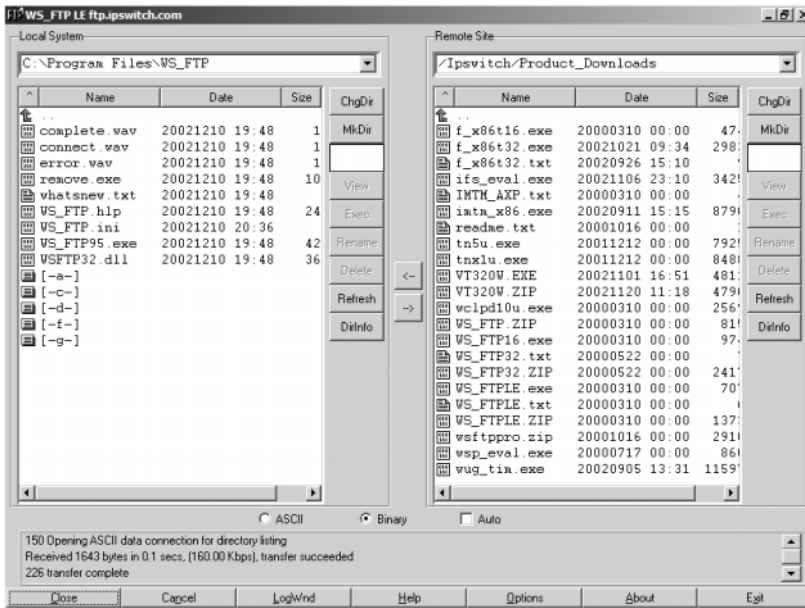



Abbildung 5.2 Oberfläche bei erfolgreicher Verbindung mit einem FTP-Server

Hinter den einzelnen Schaltflächen befinden sich unterschiedliche Funktionen.



Schaltfläche	Erklärung
ChgDir	Wechselt in ein anderes Verzeichnis.
MkDir	Erstellt ein neues Verzeichnis.
View	Zeigt die im Fenster markierte Datei an.
Exec	Führt die im Fenster markierte Datei aus.
Rename	Benennt die/das im Fenster markierte Datei/Verzeichnis um.
Delete	Löscht den im Fenster markierten Eintrag.
Refresh	Aktualisiert die Verzeichnisansicht.
DirlInfo	Zeigt Detailinformationen zum aktuellen Verzeichnis an.

Tabelle 5.3 Die Schaltflächen und ihre Aufgaben

In dem Eingabefeld können Sie ein Muster angeben, anhand dessen die Ausgabe des Verzeichnisses gefiltert wird. So würde die Eingabe *.exe und  nur Dateien anzeigen, die auf .exe enden. Um wieder alle Ver-

zeichniseinträge anzuzeigen, tippen Sie entweder *.* ein oder löschen das Muster.

Dateien transferieren



Zwischen den beiden Fenstern befinden sich zwei weitere Schaltflächen. Eine mit einem Pfeil nach links  und eine weitere mit einem Pfeil nach rechts . Diese Schaltflächen dienen dazu, Dateien und Verzeichnisse zwischen den beiden Rechnern (Ihrem und dem Server) auszutauschen. Ein Klick auf die Schaltfläche mit dem Pfeil nach links transportiert Dateien und Verzeichnisse vom Server auf Ihren Rechner (dies wird auch Download genannt). Ein Klick auf den Pfeil nach rechts transportiert Daten von Ihrem Rechner auf den Server (dies wird auch Upload genannt). Um jedoch Dateien oder Verzeichnisse transferieren zu können, müssen die entsprechenden Verzeichniseinträge markiert werden. Mit gedrückter Umschalttaste können Sie mehrere aufeinander folgende Einträge markieren, mit gedrückter Steuerungstaste beliebige Einträge. Je nachdem, ob Sie nun Dateien hochladen oder herunterladen möchten, müssen Sie natürlich im entsprechenden Fenster die Markierung vornehmen. Mit einem Klick auf eine der beiden Pfeil-Schaltflächen werden die Daten dann transferiert.

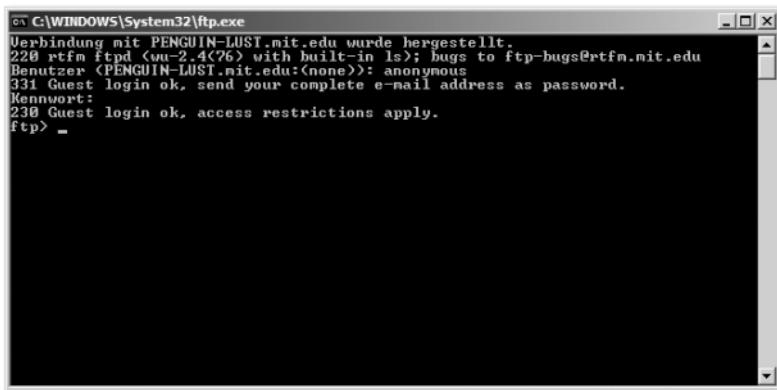
5.2.2 Kommandozeilenbasierter Client unter Windows und UNIX/Linux

Wesentlich weniger komfortabel, aber dafür sowohl bei Windows als auch bei UNIX/Linux grundsätzlich vorhanden, sind die kommandozeilenbasierten FTP-Clients.

Client starten

Unter Windows klicken Sie dafür auf **Start · Ausführen...** und tippen ftp gefolgt vom Hostname des FTP-Servers ein. Unter Linux reicht der Wechsel in eine Konsole. Dort wird dann ebenfalls ftp gefolgt vom Hostname eingegeben.

Die Eingabe `ftp rtfm.mit.edu` und  würde eine Verbindung zum FTP-Server des Massachusetts Institute of Technology (kurz MIT) aufbauen. Auf diesem FTP-Server befindet sich das Dokumentationsarchiv des MIT. Der FTP-Server fordert nun die Eingabe eines Benutzernamens. Auf vielen öffentlichen Servern können Sie sich anonym anmelden. Dafür müssen Sie als Benutzernamen lediglich `anonymous` und  eingeben. Wenn Sie sich auf dem FTP-Server Ihrer Domain anmelden möchten, müssen Sie natürlich die entsprechende Benutzerkennung angeben. Im nächsten Schritt fordert der Server das Passwort. Bei anonymer Anmeldung wird in der Regel Ihre E-Mail-Adresse erwartet. Andernfalls natürlich das Passwort, das zu der eingegebenen Benutzerkennung gehört.



```
C:\WINDOWS\System32\ftp.exe
Verbindung mit PENGUIN-LIST.mit.edu wurde hergestellt.
220 rtfm ftpd (uu-2.4(76) with built-in ls); bugs to ftp-bugs@rtfm.mit.edu
Benutzer <PENGUIN-LIST.mit.edu:(none)>: anonymous
331 Guest login ok, send your complete e-mail address as password.
Kennwort:
230 Guest login ok, access restrictions apply.
ftp> _
```

Abbildung 5.3 Kommandozeilen-FTP-Client unter Windows

Sobald Sie auf einem FTP-Server eingeloggt sind, können Sie verschiedene Befehle eingeben. Wenn Sie `help` und `[↵]` eingeben, erhalten Sie eine Übersicht über die möglichen Befehle. Die wichtigsten werde ich Ihnen an dieser Stelle kurz vorstellen.

- ▶ **dir** oder **ls** – Listet den Inhalt des aktuellen Serververzeichnisses auf. Sowohl unter UNIX/Linux als auch unter Windows sind beide Befehle erlaubt.
- ▶ **delete** – Nach dem Kommando wird ein Dateiname erwartet. Die angegebene Datei wird dann gelöscht.
- ▶ **get** – Erwartet nach dem Befehl den Namen einer Datei auf dem Server. Die angegebene Datei wird dann auf den lokalen Computer kopiert, also heruntergeladen.
- ▶ **mget** – Erwartet nach dem Befehl die Namen von mehreren Dateien, die dann auf den lokalen Computer kopiert werden.
- ▶ **put** – Erwartet nach dem Befehl den Namen einer Datei auf dem lokalen Rechner. Die angegebene Datei wird dann auf den Server kopiert, also hochgeladen.
- ▶ **mput** – Erwartet nach dem Befehl die Namen von mehreren Dateien, die dann auf den Server kopiert werden.
- ▶ **cd** – Wechsel des aktuellen Verzeichnisses auf dem Server. Nach dem Kommando wird das neue Verzeichnis erwartet.
- ▶ **lcd** – Wechsel des aktuellen Verzeichnisses auf dem lokalen Rechner. Nach dem Kommando wird das neue Verzeichnis erwartet.
- ▶ **ascii** – Datei als Textdatei übertragen. Nur bei »echten« Textdateien zu empfehlen.

- ▶ **binary** – Datei im Binärmodus übertragen. Empfohlener Modus für alle Dateitransfers.
- ▶ **help** – Zeigt eine Übersicht über die möglichen Befehle an.
- ▶ **bye** – Beendet eine Verbindung zu einem FTP-Server.
- ▶ **status** – Zeigt eine Übersicht zum Status der FTP-Verbindung an.

Abkürzungen Alle Befehle lassen sich auch abkürzen. Dabei muss die Abkürzung jedoch eindeutig sein. So können Sie anstelle von `delete` auch `del` schreiben, nicht jedoch `d`. Denn das könnte sowohl für `delete` als auch für `dir` stehen.

Sobald Sie sich ein wenig mehr mit einem kommandozeilenbasierten FTP-Client beschäftigt haben, werden Sie schnell die Vorzüge eines grafischen Clients zu schätzen wissen. Zum einen müssen Sie nicht jedes Mal Benutzernamen und Passwort eingeben und zum anderen, da das Übertragen von mehreren Dateien wesentlich schneller vonstatten geht als über die Kommandozeile. Der einzige Vorteil der Kommandozeilen-Variante ist die Verfügbarkeit eines solchen Clients auf fast jedem beliebigen System.

5.3 Suchmaschinen, Webkataloge und anderes

Suchmaschinen lassen sich in zwei Gruppen einteilen: Datenbanken, die von Suchrobotern gefüttert werden, und Kataloge, die von Redakteuren gefüllt werden.

Verbreitung = Erfolg? Gerade für den Erfolg und die Bekanntheit einer Webseite ist es wichtig, dass diese auch von Suchmaschinen gefunden wird. Und je öfter Ihre Webseite in einer Suchmaschine auftaucht, desto mehr Besucher werden sich Ihre Seite auch ansehen. Dies sagt aber noch nichts über den Erfolg Ihrer Webseite aus, denn dieser hängt immer noch vom angebotenen Inhalt ab.

Webseite bekannt machen Um eine Webseite bekannt zu machen, stehen Ihnen deshalb auch die verschiedensten Möglichkeiten zur Verfügung. Suchroboter surfen quasi durch das Internet, indem sie sich von Link zu Link hangeln. Dabei organisieren Sie eine Zusammenfassung der Webseite anhand von META-Informationen und einem kleinen Ausschnitt des Anfangs der Webseite. Wie Sie META-Informationen hinterlegen, werden Sie später noch in den Kapiteln zu HTML erfahren.

Um in einen Katalog aufgenommen zu werden, müssen Sie sich erst bei dem entsprechenden Katalog anmelden. Ein Redakteur überprüft dann

Ihre Anmeldung und fügt sie dann in die richtige Kategorie ein. Da es jedoch ein Menge solcher Kataloge gibt, ist es sehr zeitaufwändig, sich von Hand bei all diesen Katalogen anzumelden. Deshalb gibt es die verschiedensten Webseiten, welche Ihnen diese Arbeit abnehmen, wie z. B. <http://www.submit-it.com/> oder <http://www.webmasterplan.com/>. Bei diesen Webseiten geben Sie verschiedene Informationen ein, und ein entsprechendes Script kümmert sich dann um den Rest.



Abbildung 5.4 Webmasterplan hilft Ihnen dabei, Ihre Seite in verschiedene Kataloge einzutragen.

Eine weitere beliebte Variante ist es, sich einem so genannten Webring anzuschließen. Dies ist eher für Privatanwender oder Vereine zu empfehlen, nicht aber für Firmen. Bei einem Webring schließen sich mehrere Webseiten mit ähnlichem Inhalt oder dem gleichen Thema zu einem Ring zusammen. Der Benutzer einer der Webseiten kann dann über verschiedene Links zu einer zufällig ausgewählten Seite dieses Rings gelangen, zur nächsten Seite oder zu einer der Top-5-Webseiten. Sie müssen dabei lediglich auf Ihrer Homepage die notwendigen Links und Grafiken für einen solchen Webring einfügen. Auf der Seite <http://www.webring.de/> finden Sie eine Zusammenstellung der verschiedensten Webringe.

Webringe

Natürlich können Sie auch eine Nachricht in einer der zahlreichen Newsgroups hinterlassen und hoffen, dass sich ein Abonnent der Newsgroup

Newsgroups

Ihre Webseite ansieht. Davon möchte ich jedoch abraten, da dies nicht besonders gern gesehen ist und einen Rattenschwanz von Beschwerden nach sich ziehen kann. Wenn Sie dies trotzdem tun möchten, dann bitte in der Newsgroup **de.comm.infosystems.www.pages.announce**. Dies ist eine moderierte Newsgroup, in der Ankündigungen von Webseiten gemacht werden.

Teil 2
HTML – Die Sprache des
Internets

1 HTML – Die Sprache des Internets

Die Menschen sind heute nicht mehr verwurzelt, sondern vernetzt.

Sie werden in diesem Kapitel die wichtigsten HTML-Grundlagen und Regeln kennen lernen und Ihr erstes HTML-Dokument erstellen. Auch wenn Sie bereits über Kenntnisse verfügen, werden Sie hier sicherlich noch die ein oder andere Information erhalten.

1.1 Am Anfang war HTML ...

Zunächst möchte ich ein weit verbreitetes Missverständnis aus dem Weg räumen: HTML ist keine Programmiersprache. Auch wenn dies für viele Leser eine Überraschung sein dürfte. HTML ist ein Akronym für Hypertext Markup Language, was ins Deutsche übersetzt soviel wie Hypertext-Auszeichnungssprache bedeutet. Es handelt sich bei HTML um eine Weiterentwicklung von SGML (Structured Generalized Markup Language) und baut auf dem HTTP-Protokoll auf. Mit HTML lassen sich die Struktur, der Inhalt und das Verhalten eines Dokuments beschreiben bzw. auszeichnen, und es wird kein Programm im eigentlichen Sinne erstellt. Die Folge ist, dass HTML-Dokumente in verschiedenen Browsern unterschiedlich dargestellt werden. Ich werde später noch genauer darauf eingehen.

Die Wurzeln von SGML reichen bis in die 1960er zurück, obwohl der eigentliche SGML-Standard erst Anfang 1980 publiziert wurde. Der Anstoß zur Entwicklung von SGML war die Schaffung eines Dokumentformats, das die Struktur klar vom Inhalt trennen sollte, da durch den technologischen Fortschritt die Formatierung von Texten automatisiert worden war. Der häufig verwendete Begriff »Markup« (dt. *Auszeichnung*) geht auf handgeschriebene Anmerkungen von Autoren und Textdesignern zurück, die Textvorlagen mit Markups versahen, die dem Setzer als Anweisungen darüber dienten, wie einzelne Textabsätze und Wörter gedruckt werden sollten. Heutzutage verbindet man mit dem Begriff »Markup« alle Mittel, die die Interpretation eines Dokuments ermöglichen.

SGML

HTML-Dokumente sind also reine Textdokumente mit Inhalt (Texte, Überschriften, Bilder, Tabellen etc.), dem Sie eine Struktur (und somit ein Aussehen) zuweisen. Dies geschieht über spezielle Zeichenfolgen im Dokument, den so genannten Tags. Ein Browser ist in der Lage, den Inhalt

Hyper werden

von der Struktur zu unterscheiden und den Inhalt optisch entsprechend darzustellen. Solche textbasierten Datenbeschreibungssprachen werden gemeinhin auch als Meta-Sprachen bezeichnet. Der Begriff »Hypertext« steht für miteinander verknüpfte Texte. Das Anklicken eines speziell hervorgehobenen Wortes innerhalb eines Textes bzw. eines Bildes führt den Benutzer auf eine weitere Seite oder ein weiteres Dokument, das in inhaltlicher Beziehung zum Ausgangstext steht – ebenfalls eine auf den technologischen Fortschritten basierende Entwicklung. Während ein Buch linear aufgebaut ist (in der Regel liest man es von vorne nach hinten), sind Hypertexte ein nicht lineares Medium zum Verfassen und Darstellen von Texten und so die optimale Lösung für die Bereitstellung von Texten in einem Computernetzwerk (wie dem World Wide Web).

1.2 HTML-Versionen

In den vergangenen zehn Jahren unterlag auch HTML (wie jede computerspezifische Technologie) einer stetigen Entwicklung und Erneuerung aufgrund wachsender Anforderungen und Veränderungen im World Wide Web. Dies führte dazu, dass bislang vier Ausbaustufen von HTML vorliegen.

1.2.1 HTML 1

Die erste Sprachversion ist mit der heutigen Version 4 kaum zu vergleichen. Zwar war es bereits in dieser Version möglich, Dokumente mit Überschriften und Grafiken zu versehen, und auch Verweise konnten gesetzt werden, aber alles in allem war es das schon. Sie werden im Internet auch keine Dokumente mehr finden, die sich auf diese Sprachversion beziehen bzw. beschränken. An dieser Stelle sollte aber erwähnt werden, dass HTML 1 zum Zeitpunkt der Veröffentlichung (1990) ein bahnbrechender Erfolg war.

1.2.2 HTML 2

Es dauerte gute fünf Jahre bis die zweite Sprachversion von HTML dann 1995 veröffentlicht wurde. Sie enthielt aber nur wenig mehr als kleine Verbesserungen, und die Mehrzahl der Elemente, die in dieser Sprachversion neu hinzugekommen sind, werden von den meisten Browsern bis heute nicht unterstützt. Das Kuriose war jedoch die Entwicklung von Netscape, dessen Navigator in der Version 2.0 weit mehr und ausgefallene Möglichkeiten bot, als es der HTML 2.0-Standard vorsah. Der Navigator implementierte bereits die Frame-Technologie (das Browserfenster

konnte dadurch in mehrere kleine unterteilt, und somit konnten verschiedene HTML-Dokumente gleichzeitig dargestellt werden) und Multi-Media-Referenzierungen (Einbinden von Videos, Sounds und Ähnlichem). Dies verschaffte dem Netscape Navigator die alleinige Marktführung.

1.2.3 HTML 3.2

Im Januar 1997 folgte dann die HTML-Sprachversion 3.2. Die eigenartige Versionsnummer (3.2 anstatt 3.0) basiert auf den Unterschieden zwischen der gewünschten Entwicklung des WWW durch das W3C und den tatsächlichen Entwicklungen und Anforderung durch die Internet-Gemeinschaft. Grundzüge der Version 3.0 waren überaus ansprechend, da sie viele neue Elemente und Möglichkeiten enthalten sollte. Sie waren aber allesamt realitätsfern. Dieser Versions-Standard wurde niemals offiziell, sondern erst die vollkommen überarbeitete Version 3.2, die alles in allem auch nicht das Gelbe vom Ei war. Mittlerweile ließen sich zwar offiziell Tabellen und physische Textauszeichnungen in einem HTML-Dokument verwenden, die meisten dieser Elemente sind aber schon wieder als deprecated (dt. *ablehnend*) deklariert.

1.2.4 HTML 4 und 4.01

Am 18.02.1998 wurde dann die HTML-Sprachversion 4 offiziell verabschiedet, die dann am 24.12.1998 (sozusagen als Weihnachtsgeschenk) durch die überarbeitete Version 4.01 ersetzt wurde. Prinzipiell besann sich das W3C bei der Entwicklung von HTML 4.x wieder auf die Version 2 und lagerte die optische Gestaltung eines HTML-Dokuments erstmals in CSS aus. Die Version 4.x regelte hauptsächlich die Einbindung von Scripts und die Internationalisierung auf Unicode-Basis. Außerdem bot sie verschiedene Sprachvarianten (strict, transitional und frameset), mit denen Sie sich noch detaillierter auseinander setzen werden. Die Grundlage in diesem Buch bildet die Version 4.x von HTML.

1.3 Das erste HTML-Dokument

Starten Sie jetzt Ihren Editor, und legen Sie ein neues Dokument an. Den bekannten Windows-Editor Notepad finden Sie normalerweise unter **Start • Programme • Zubehör • Editor**. Sie können aber auch auf **Start** und **Ausführen...** klicken. Geben Sie in die Textzeile »notepad« ein, und klicken Sie auf **OK**. Tippen Sie das Listing 1.1 in den Editor, und speichern Sie das Dokument unter **list1.1.htm** ab. Sollte Notepad das Dokument

mit der Endung .txt abspeichern, müssen Sie im Dialogfeld **Datei • Speichern unter...** als Dateityp **Alle Dateien (*.*)** auswählen und als Dateinamen **list1.1.htm** eintippen.



Speichern Sie HTML-Dokumente immer mit der Endung **.htm** oder **.html** ab. Das sind die Standard-Dateiendungen für HTML-Dokumente. Nach einem Doppelklick auf eine so abgespeicherte Datei öffnet der installierte Browser die Datei und zeigt sie an.

Öffnen Sie die Datei **list1.1.htm** nun in Ihrem Browser. Ihr Ergebnis sollte dem in Abbildung 1.1 gezeigten entsprechen.

```
<html>
  <head>
    <title>Listing 1.1</title>
  </head>
  <body>Hallo WorldWideWeb!</body>
</html>
```

Listing 1.1 Hallo WorldWideWeb!

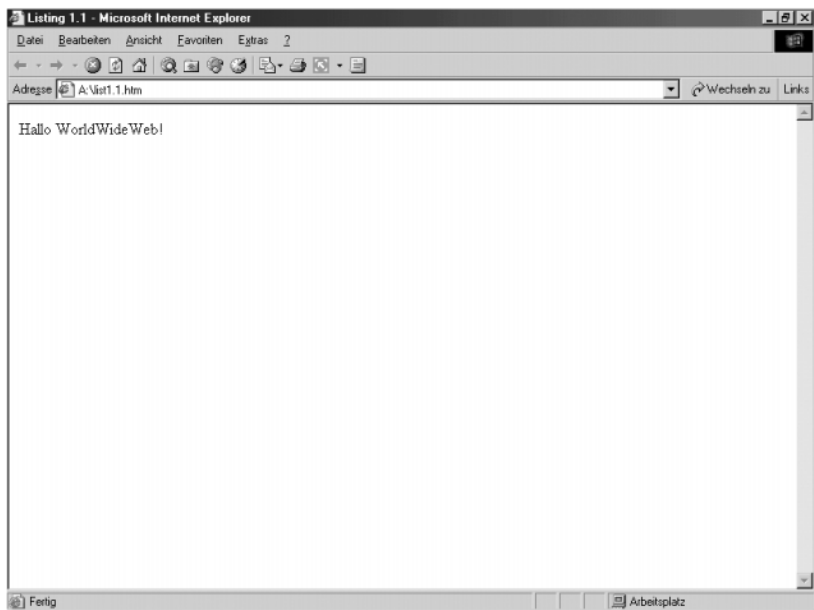


Abbildung 1.1 Darstellung des Listings 1.1 im Internet Explorer 6.0

Herzlichen Glückwunsch! Dies ist das erste von Ihnen erstellte HTML-Dokument.

1.4 Elemente und Tags

Wenn Sie sich das Listing 1.1 genauer ansehen, entdecken Sie gewisse Regelmäßigkeiten. Sie können Schlüsselnamen erkennen, die in spitzen Klammern geschrieben wurden. Diese Schlüsselnamen mit den spitzen Klammern nennt man Tags (dt. *Etiketten*), und sie unterteilen sich in Start- und Ende-Tags. Letztere kann man von den Start-Tags durch einen vorangestellten Schrägstrich / unterscheiden. Die Schlüsselnamen ergeben sich aus der Bezeichnung des Elements. Was jedoch sind Elemente? Ein HTML-Dokument besteht aus einer Vielzahl unterschiedlicher Elemente. Diese Elemente weisen gewisse Eigenschaften und Verhaltensformen auf und haben somit unterschiedliche spezifische Aufgaben, z. B. erzeugt das Element `h1` eine Überschrift der ersten Ordnung. Der zwischen den Tag-Paaren (Start- und Ende-Tag) stehende Text (auch Gültigkeitsbereich genannt) hebt sich deutlich vom restlichen Inhalt ab. In Listing 1.1 können Sie insgesamt vier dieser Elemente ausmachen: `html`, `head`, `title` und `body`. Diese werden Sie auch in jedem anderen HTML-Dokument wiederfinden, da sie die Grundausstattung eines HTML-Dokuments bilden. Ich nenne sie deshalb Basiselemente.

Sehen Sie sich die Zeilen `<title>Listing 1.1</title>` und `<body>Hallo WorldWideWeb!</body>` genauer an. Die jeweiligen Start- und Ende-Tags der Elemente `title` und `body` umrahmen unterschiedlichen Text. In Abbildung 1.1 können Sie diese Texte wiedererkennen: Der Text im Gültigkeitsbereich von `title` (also zwischen Start- und Ende-Tag) steht in der Titelseite des Browserfensters, der Text im Gültigkeitsbereich von `body` im Anzeigebereich des Browsers.

Kopf und Körper

Somit können Sie die Aufgaben dieser beiden Elemente schon festlegen. Das `title`-Element definiert den Dokument-Titel und das `body`-Element enthält den darzustellenden Inhalt eines Dokuments. Das Pendant zum `body`-Element ist das `head`-Element. Es enthält alle wichtigen Dokumentinformationen (wie den Titel), die nicht im Anzeigebereich des Browsers dargestellt werden sollen. Im Gültigkeitsbereich eines Elements können auch weitere Elemente stehen. Dies ist insbesondere am `html`-Element zu erkennen. Alle Elemente eines Dokuments müssen in den Gültigkeitsbereich des `html`-Elements geschrieben werden.

Die typische syntaktische Notation für ein Element, das einen beliebigen Text enthält, sieht also folgendermaßen aus:

```
<Elementname>Text</Elementname>
```

Um in einem Element ein weiteres Element notieren zu können (auch als Verschachtelung bezeichnet), ersetzen Sie den Text einfach durch das weitere Element:

```
<Elementname1>  
  <Elementname2>Text</Elementname2>  
</Elementname1>
```

Eine Verschachtelung lässt sich beliebig lange fortsetzen, bis das hierarchisch letzte Element kein weiteres Element, sondern nur Text enthält.

1.5 Hierarchie

Sie kennen nun die vier Basiselemente. Diese und auch alle anderen Elemente unterliegen einer gewissen Ordnung. Sie legt fest, wie oft und in welcher Reihenfolge ein Element im HTML-Dokument vorkommen darf. Vergleichen Sie ein HTML-Dokument mit einem Haus, dann stellt das `html`-Element das Haus dar. Die Elemente `head` und `body` bilden unterschiedliche Stockwerke, z. B. wäre `body` das Erdgeschoss und `head` der 1. Stock. Das `title`-Element kommt nur im `head`-Element vor und steht sinnbildlich für einen Raum im 1. Stock. Text, der im Gültigkeitsbereich des `title`-Elements notiert wird, ist dann gleichsam die Innenausstattung dieses Raums im 1. Stock des Hauses.

Wenn Sie nun diese Hierarchie (Ordnung) anhand des Hauses zurückverfolgen, stellen Sie fest, dass das `title`-Element dem `head`-Element untergeordnet ist, genauso wie die Elemente `head` und `body` dem `html`-Element untergeordnet sind. Man spricht auch von Eltern- und Kindelementen. Das `title`-Element ist das Kindelement des `head`-Elements, und anders herum ist `head` das Elternelement des `title`-Elements.

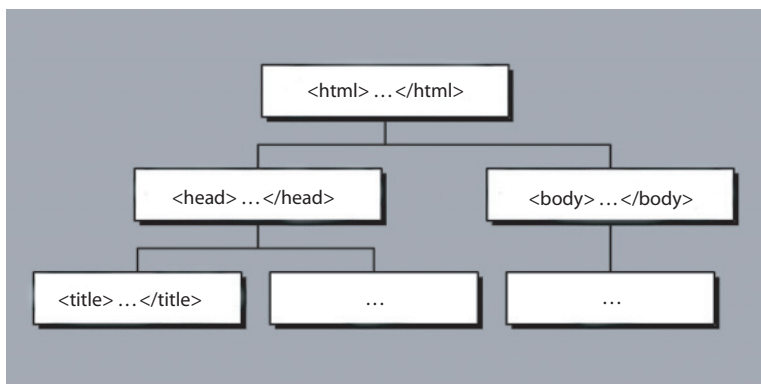


Abbildung 1.2 Schematische Darstellung der Element-Hierarchie

Die Felder mit den drei Punkten in Abbildung 1.2 stehen für weitere Kindelemente, die innerhalb von `head` oder `body` vorkommen dürfen. Versuchen Sie sich diese Hierarchie gut einzuprägen, da sie essenziell für das weitere Verständnis von HTML ist.

Neben dem bereits bekannten `title`-Element können im Kopfteil weitere Elemente notiert werden, die Sie später noch etwas näher kennen lernen werden. Dies sind alle Elemente, die grundlegende Informationen zum Dokument, wie Titel, Autor oder Erstellungsdatum oder auch JavaScripts und StyleSheets, zur Verfügung stellen. Die Gemeinsamkeit liegt darin, dass sie im Anzeigebereich des Browsers nicht dargestellt werden, sondern die Darstellung teilweise nur beeinflussen oder aber Informationen für verschiedene Suchmaschinen bereitstellen.

head-Element

Im Rumpfteil (`body`) wird der wesentliche Inhalt des HTML-Dokuments, wie Texte, Bilder, Hyperlinks, Tabellen, Listen und Ähnliches notiert. Diese Elemente werde ich Ihnen in den nächsten Kapiteln noch genauer erläutern.

body-Element

1.6 Attribute

Jedes Element besitzt gewisse Eigenschaften oder Auszeichnungsmerkmale. Diese Eigenschaften führen dazu, dass z. B. der Text im Gültigkeitsbereich eines `h1`-Elements (Überschrift 1. Ordnung) größer dargestellt wird, als wenn er in einem `h2`-Element (Überschrift 2. Ordnung) notiert worden wäre. Nun lässt sich die Darstellung des Textes aber in einem gewissen Rahmen durch Ändern der Eigenschaften an die eigenen Wünsche anpassen.

Dafür werden Attribute und Parameter verwendet. Diese werden in Kombination innerhalb des Start-Tags eines Elements gesetzt. Dabei wird zuerst der Name des Attributs, gefolgt von einem `=`-Zeichen (Gleichheitszeichen) und dem Parameter in Anführungszeichen notiert. Die Syntax sieht wie folgt aus:

```
<Elementname Attribut="Parameter">Text oder Element</Elementname>
```

Um mehrere verschiedene Eigenschaften eines Elements verändern zu können, werden die gewünschten Attribute in beliebiger Reihenfolge im Start-Tag notiert.

```
<Elementname Attribut1="Parameter1"
Attribut2="Parameter2" AttributN="ParameterN">Text oder
Element</Elementname>
```

Natürlich ist auch eine Verschachtelung von Elementen mit den veränderten Attributen möglich.

```
<Elementname1 Attribut="Parameter">
  <Elementname2 Attribut="Parameter">Text oder Element</
Elementname2>
</Elementname1>
```

1.7 Kommentare und Quellstrukturierung

Bei kleineren HTML-Dokumenten ist eine Kommentierung oder saubere Strukturierung des Quelltextes nicht weiter erforderlich. Sobald sich aber ein solches Dokument über mehrere Bildschirmseiten erstreckt, kann es ohne Kommentare und Quellstrukturierung schnell unübersichtlich werden. Wenn Sie sich das Listing 1.1 noch einmal ansehen, können Sie erkennen, dass nach den Start-Tags der Elemente `html` und `head` in der nächsten Zeile um zwei Leerzeichen eingerückt wurde. Damit wird der Quelltext besser lesbar.

Kommentare Für Kommentare in HTML gibt es kein spezielles Element, da sie nicht der Hierarchie eines Dokuments unterliegen und an jeder beliebigen Stelle (ja, auch vor dem `html`-Start-Tag oder nach dem `html`-Ende-Tag) im Dokument notiert werden dürfen. Er wird durch die Zeichenfolge `<!--` eingeleitet und endet mit der Zeichenfolge `//-->`. Zwischen diesen beiden Zeichenfolgen steht dann der Kommentar.

```
<!-- Dies ist ein Kommentar //-->
```

Regeln zur Strukturierung Für die Strukturierung in HTML gibt es keine Norm, jedoch haben sich im Laufe der Zeit einige Regeln durchgesetzt:

- ▶ Die Start- und Ende-Tags der Elemente, die weitere Elemente enthalten, werden in unterschiedlichen Zeilen notiert, wobei die Einrückung dieser Tags gleich sein sollte. Ein Beispiel dafür sind die Elemente `html` und `head` in Listing 1.1.
- ▶ Die Start- und Ende-Tags eines Elements, das kein Kindelement enthält, sollten nach Möglichkeit in einer Zeile notiert werden. In Listing 1.1 können Sie dies an den Elementen `title` und `body` erkennen.

- Browser unterscheiden bei Tags nicht zwischen Groß- und Kleinschreibung. Daher ist es egal, ob Sie die Tags groß (`<HTML>...</HTML>`), klein (`<html>...</html>`) oder vollkommen unterschiedlich (`<hTmL>...</HTmL>`) schreiben. Sie sollten sich aber für eine Schreibweise entscheiden. Wenn Sie einen Editor ohne farbliche Hervorhebung der Tags verwenden, empfiehlt es sich, alle Tags großzuschreiben, um sie vom restlichen Text unterscheiden zu können.

1.8 Entities

Wenn Sie in einem HTML-Dokument die spitzen Klammern `<` (kleiner als) und `>` (größer als) in einem normalen Text verwenden, wird dies zwangsläufig zu einer Fehldarstellung führen, da die Klammern zum Notieren der HTML-Tags reserviert sind. Ähnlich verhält es sich mit den Anführungszeichen, mittels derer Parameter einem Attribut zugewiesen werden. Für solche Fälle gibt es so genannte Kurzcodes oder Entities, um Fehldarstellungen zu vermeiden. Dies gilt aber auch für Umlaute (Ä, Ö, Ü) und verschiedene andere Sonderzeichen. Auch wenn in Ihrem Browser Umlaute und Sonderzeichen korrekt dargestellt werden, hängt die Darstellung solcher Sonderzeichen von der landesspezifischen Spracheinstellung eines Browsers ab. Umlaute gibt es im englischen Alphabet beispielsweise nicht, und die dargestellten Zeichen wären somit nicht die gewünschten Umlaute.

Entities werden durch das Zeichen `&` (kaufmännisches Und) eingeleitet, gefolgt von einer Buchstaben- und/oder Zahlenkombination und enden mit dem Zeichen `;` (Semikolon oder Strichpunkt).

`&zeichencode;`

Die Entities für die spitzen Klammern lauten beispielsweise `<` (`lt` = engl. *less than*, dt. *kleiner als*) für `<` und `>` (`gt` = engl. *greater than*, dt. *größer als*) für `>`. In der Tabelle 1.1 finden Sie eine Übersicht über die wichtigsten Entities. Eine vollständige Tabelle finden Sie im Anhang.

Sonderzeichen	Entity	ISO-Code
"	<code>&quot;</code>	<code>&#34;</code>
&	<code>&amp;</code>	<code>&#38;</code>
<	<code>&lt;</code>	<code>&#60;</code>
>	<code>&gt;</code>	<code>&#62;</code>

Tabelle 1.1 Die wichtigsten Entities auf einen Blick

Sonderzeichen	Entity	ISO-Code
Leerzeichen	 	
©	©	©
®	®	®
Ä	Ä	Ä
Ö	Ö	Ö
Ü	Ü	Ü
ß	ß	ß
ä	ä	ä
ö	ö	ö
ü	ü	ü

Tabelle 1.1 Die wichtigsten Entities auf einen Blick (Forts.)



Der Einfachheit und Lesbarkeit halber habe ich in den vorangegangenen Listings auf diese Entities und Sonderzeichen verzichtet. Dies wird sich in den folgenden Listings jedoch ändern. Prägen Sie sich die wichtigsten Entities (auf jeden Fall die in der Tabelle 1.1 aufgeführten) also gut ein.

1.9 Dokumententyp-Definition

In SGML ist es üblich, zu Beginn des Dokuments die DTD (engl. *Document Type Definition*) zu notieren. Da HTML ja bekanntlich auf SGML basiert, ist es sinnvoll, auch in HTML-Dokumenten die entsprechende DTD anzuführen, da sie alle Besonderheiten, welche ein Dokument bezüglich der Struktur und des Erscheinungsbildes haben darf, beschreibt.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
```

Die Dokumententyp-Definition wird durch `<!DOCTYPE` eingeleitet. `HTML PUBLIC` steht für die öffentlich verfügbare HTML-DTD. `W3C` ist der Herausgeber der DTD – in diesem Fall das World-Wide-Web-Konsortium. `DTD HTML 4.01 Transitional` bedeutet, dass Sie in der Datei den Dokumententyp HTML in der Version 4.01 und der Variante `Transitional`

verwenden. `EN` ist das Kürzel für die Sprache, in der die Tags notiert werden (in diesem Fall Englisch), nicht etwa für die Sprache, in welcher der Inhalt des Dokuments verfasst wurde.

Wie Sie bereits am Ausrufezeichen zu Beginn der DTD-Syntax erkennen können, ist dies kein normales Element. Ähnlich wie Kommentare unterliegen sie nicht den Regeln der Hierarchie für HTML-Dokumente. Im Gegensatz zu Kommentaren dürfen sie aber nicht an beliebiger Stelle stehen, sondern müssen außerhalb des `html`-Elements zu Beginn des Dokuments angeführt werden. Die erste Zeile in einem Dokument drängt sich als Ort für die DTD quasi auf. Dies ist aber nicht zwingend erforderlich. Hauptsache, sie steht vor dem Start-Tag (`<html>`) des `html`-Elements.



1.9.1 Sprachvarianten

Die HTML-Sprachversion 4.x liegt in den drei verschiedenen Sprachvarianten `strict`, `transitional` und `frameset` vor. Eine davon müssen Sie bei der Angabe einer DTD verwenden.

Die Variante `strict` wird verwendet, wenn bestimmte Elemente und Attribute, die einmal zum HTML-Standard gehörten, nicht mehr verwendet werden sollen. Außerdem darf im `body`-Element nicht einfach nur Text stehen, sondern dieser muss innerhalb eines entsprechenden Elements notiert werden, da die Verschachtelungsregeln dieser Variante sehr viel strenger und genauer definiert sind. Die entfallenden Elemente und Attribute werden dann durch StyleSheets ersetzt. Eine Beispielangabe für diese Variante sieht dann so aus:

Strict

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Strict//EN">
```

Wenn Sie sich nicht nur auf Browser beschränken möchten, die CSS unterstützen und interpretieren können (darunter fallen Netscape und Internet Explorer 3.x und älter), sollten Sie die Variante `transitional` verwenden (was auch die meisten tun). Dadurch dürfen Sie ältere Elemente und Attribute verwenden und auch einfach nur Text zwischen den `body`-Tags notieren, ohne diese in ein entsprechendes Element setzen zu müssen. Ein Beispiel:

Transitional

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
```

Frameset Die Variante `frameset` wird nur in HTML-Dokumenten angeführt, welche die Frame-Technologie verwenden. Nehmen Sie von dieser Variante lieber etwas Abstand, damit Ihre Dokumente auch in den Browsern richtig dargestellt werden, die sich explizit an die DTD halten. Ein Beispiel:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Frameset//EN">
```

1.9.2 DTDs für andere HTML-Versionen und für XHTML

Sowohl für die HTML-Version 2.0 als auch für die Version 3.2 gibt es eine eigene Variante der Dokumententyp-Definition. Für die auf XML basierende HTML-Version XHTML 1.0 und 1.1 gibt es ebenfalls eine spezielle DTD. Die folgende Tabelle zeigt die entsprechenden Anpassungen.

Typ, Version, Variante	DTD
HTML 2.0	<code><!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 2.0//EN"></code>
HTML 3.2	<code><HTML PUBLIC "-//W3C//DTD HTML 3.2//EN"></code>
XHTML 1.0 Transitional	<code><!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"></code>
XHTML 1.0 Strict	<code><!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"></code>
XHTML 1.0 Frameset	<code><!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"></code>
XHTML 1.1 (nur Strict)	<code><!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"></code>

Tabelle 1.2 DTDs für weitere Versionen

1.9.3 HTML-Vorlage

Da Sie im Verlauf des Buches noch des Öfteren HTML-Dokumente erstellen werden, sollten Sie sich eine Vorlage erstellen und abspeichern. Sie können eine Vorlage nach eigenen Wünschen erstellen oder sich auf Listing 1.2 berufen.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<!-- Kommentar zum Inhalt des Dokuments //-->
<html>
  <head>
```



```
<title>Titel</title>
</head>
<body>
  Inhalt
</body>
</html>
```

Listing 1.2 Beispiel für eine Dokumentvorlage

1.10 Zusammenfassung

- ▶ HTML ist eine Erweiterung der Auszeichnungssprache SGML und basiert auf dem HTTP-Protokoll.
- ▶ Ein HTML-Dokument besteht aus verschiedenen Elementen, welche den Inhalt, die Struktur und das Verhalten eines Dokuments definieren.
- ▶ Elemente lassen sich mittels Attributen und Parametern an eigene Wünsche anpassen.
- ▶ Die Kommentierung und Quellstrukturierung eines HTML-Dokuments erzeugt Übersicht und bessere Lesbarkeit.

1.11 Fragen und Übungen

1. Welche Basiselemente gibt es in einem HTML-Dokument?
2. Notieren Sie einen Kommentar mit dem Text »Das ist die Startseite«.
3. Erstellen Sie ein HTML-Dokument mit dem Dokumententitel Startseite und dem Text im Anzeigebereich »Willkommen auf meiner Startseite!«
4. Wofür wird die DTD (Document Type Definition) benötigt?
5. Worin besteht der Unterschied zwischen einem Eltern- und einem Kindelement?

2 Textstrukturierung

Die intrinsisch motivierte Interaktion zwischen Mensch und Computer verifiziert die funktionalstrukturelle Hypothese, der zufolge sich die Evidenz der Realität als Reduktion zur Komplexität interpretieren lässt ... Mit anderen Worten: Der Mensch drückt >RETURN<, der Computer liefert ein falsches Ergebnis ...

– Niels Sedat, deutscher Systemprogrammierer

In diesem Kapitel werden Sie sich mit den allgemeinen Elementen zur Textstrukturierung und zu Element-Attributen befassen und die Regeln zu Texten und Zeichen in HTML-Dokumenten kennen lernen.

2.1 Absätze

Bisher haben Sie Text, der im Anzeigebereich des Browsers ausgegeben werden soll, einfach im Gültigkeitsbereich des `body`-Elements geschrieben. In früheren HTML-Versionen war dies auch vollkommen in Ordnung. Seit der Version 4.0 (mit der wir uns in diesem Buch befassen) ist dies aber nicht mehr HTML-konform. Zwar stellen die gängigsten Browser einen solchen Text fehlerfrei dar, dies bedeutet aber nicht, dass es kein Fehler ist, da Browser in der Regel über eine sehr hohe Fehlertoleranz verfügen.

Anstatt den Text einfach im `body`-Element zu notieren, wird dafür das `p`-element (`p` = engl. *paragraph*, dt. *Absatz*) verwendet. Dieses Element bietet auch sogleich einige Vorteile:

Textabsatz

- ▶ Nach einem `p`-Element ist zum folgenden Element ein deutlicher Absatz zu erkennen.
- ▶ Mit einem speziellen Attribut lässt sich der Text wie in Textverarbeitungsprogrammen links, rechts, zentriert oder als Blocksatz ausrichten.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
```

```
<!-- Beispiel für das p-Element //-->
```

```
<html>
```

```
  <head>
```

```
    <title>Listing 2.1</title>
```

```
  </head>
```

```
  <body>
```

```
<p>Dieser Text steht in einem p-Element.</p>
<p>Dies ist ein weiterer
    Text in einem
    zweiten Absatz.
</p>
</body>
</html>
```

Listing 2.1 Beispiel für das p-Element

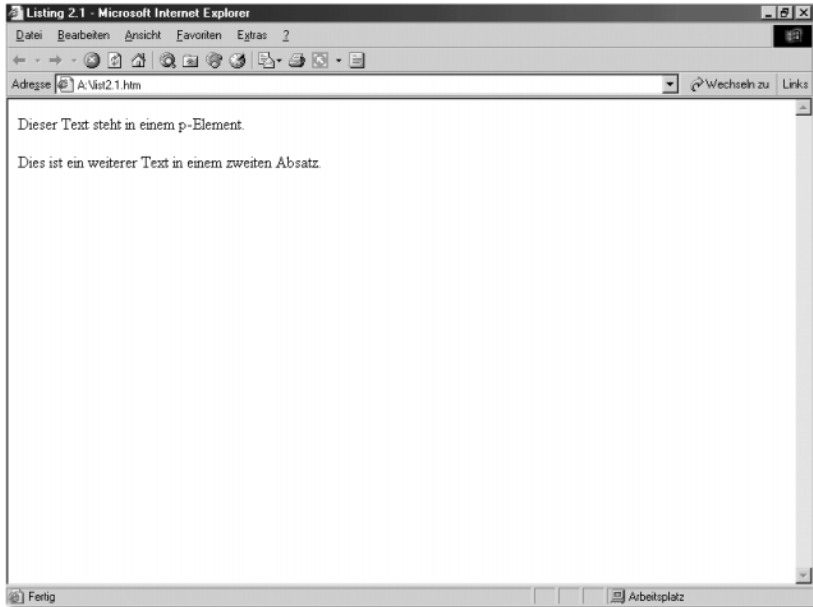


Abbildung 2.1 Listing 2.1 im Internet Explorer 6.0 dargestellt

Wenn Sie das Listing 2.1 mit den Zeilenumbrüchen in Ihren Editor übertragen, werden Sie bei der Darstellung im Browser zwei Dinge erkennen: den deutlichen Abstand zwischen beiden Absätzen und, obwohl der Text im zweiten Absatz im HTML-Dokument in drei Zeilen steht, dass der Browser den Text in nur einer Zeile darstellt. Diesen Effekt werde ich Ihnen an späterer Stelle noch einmal genauer erklären.

2.2 Textausrichtung

Standardmäßig wird der mit Elementen der Textstrukturierung ausgezeichnete Text linksbündig ausgerichtet. Das heißt, dass ein Text, der sich über mehrere Zeilen erstreckt, an einem imaginären Rand auf der linken

Seite ausgerichtet ist und eine vertikale Linie bildet. Um dies zu beeinflussen, gibt es das Attribut `align` und vier Parameter, um den Text, wie in den meisten Textverarbeitungsprogrammen auch, linksbündig, rechtsbündig, zentriert oder als Blocksatz auszurichten. Die entsprechenden Parameter lauten:

Wert	Erklärung
Left	Richtet den Absatz am linken Seitenrand aus.
Right	Richtet den Absatz am rechten Seitenrand aus.
Center	Zentriert den Absatz in der Seitenmitte.
Justify	Richtet den Absatz als Blocksatz aus.

Tabelle 2.1 Mögliche Werte für das Attribut `align`

Nun können wir das bereits bekannte `p`-Element mit einer neuen Ausrichtung versehen. Ein Beispiel-Listing, das alle vier Absatzausrichtungen darstellt, finden Sie in Listing 2.2 und die entsprechende Darstellung im Internet Explorer 6.0 in Abbildung 2.2.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<!-- Textausrichtung //-->
<html>
  <head>
    <title>Listing 2.2</title>
  </head>
  <body>
    <p align="left">Dieser Absatz ist am linken Seitenrand ausgerichtet.</p>
    <p align="right">Dieser Absatz ist am rechten Seitenrand ausgerichtet.</p>
    <p align="center">Dieser Absatz ist zentriert.</p>
    <p align="justify">Dieser Absatz wurde mit Hilfe des align-Attributs als Blocksatz ausgerichtet und wurde mit Absicht mit einem l&auml;ngeren Text versehen, um den Parameter justify auch beispielhaft darstellen zu k&ouml;nnen. Dieser Absatz wurde mit Hilfe des align-Attributs als Blocksatz ausgerichtet und wurde mit Absicht mit einem l&auml;ngeren Text versehen, um den Parameter justify auch beispielhaft darstellen zu
```

```

k&ouml;nne.</p>
  </body>
</html>

```

Listing 2.2 Textabsätze mit unterschiedlicher Ausrichtung

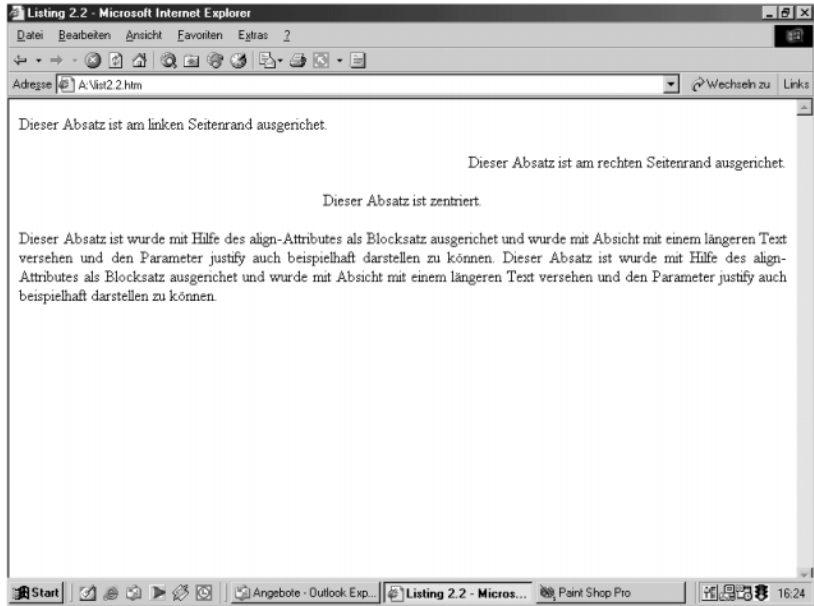


Abbildung 2.2 Textabsätze mit unterschiedlichen Ausrichtungen



Bei der Verwendung des Parameters `justify` ist eine wichtige Tatsache zu berücksichtigen. Damit der Absatz auch wirklich als Blocksatz ausgerichtet wird, muss er sich mindestens über zwei Zeilen erstrecken. Ansonsten wäre zum Parameter `left` kein optischer Unterschied erkennbar. Berücksichtigen Sie dies, falls Sie der Meinung sind, dass der Absatz vom Browser nicht korrekt ausgerichtet dargestellt wird.

2.3 Zeilenumbrüche

In Listing 2.1 konnten Sie eine Besonderheit feststellen: Der Text, der im zweiten Absatz im HTML-Dokument über drei Zeilen verteilt war, wurde im Browser in einer Zeile dargestellt. Das liegt daran, dass HTML Zeilenumbrüche oder mehrere Leerzeichen hintereinander in einem `p`-Element ignoriert. Um trotzdem einen Zeilenumbruch erzwingen zu können, wird das `br`-Element (engl. *line break*, dt. *Zeilenumbruch*) verwendet.

Dieses Element, wie auch viele weitere, die Sie kennen lernen werden, besitzt eine Besonderheit: Es gibt kein Ende-Tag zum `br`-Element, da es einen Text nicht direkt formatiert und keine weiteren Elemente enthalten darf – es gibt also keinen Gültigkeitsbereich.



```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<!-- Beispiel für Zeilenumbrüche im p-Element //-->
<html>
  <head>
    <title>Listing 2.2</title>
  </head>
  <body>
    <p>Dieser Absatz enth&auml;lt
    sehr viel Text<br> der im
    HTML-Dokument aber          ein vollkommen
    anderes          Erscheinungsbild<br>
    besitzt als die Ausgabe      im Browser vermuten
    l&auml;sst.
    </p>
  </body>
</html>
```

Listing 2.3 Zeilenumbrüche

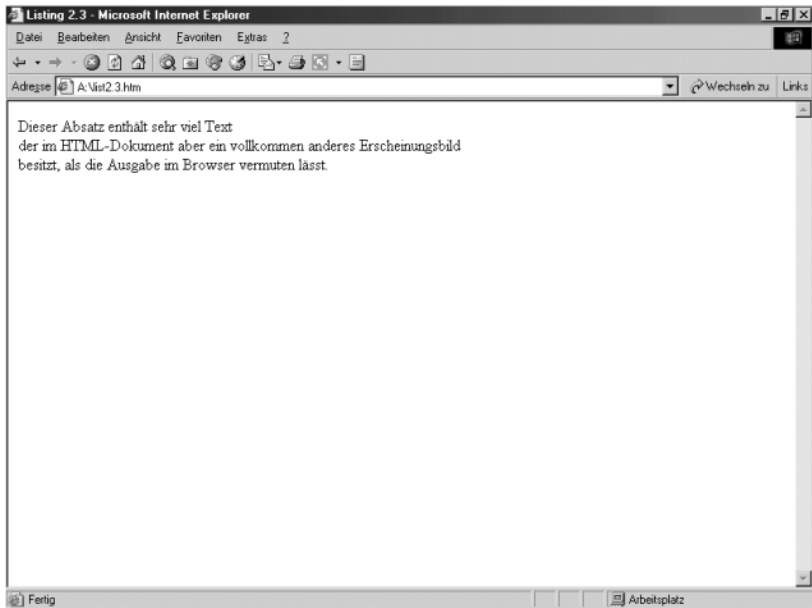


Abbildung 2.3 Darstellung des Listings 2.3 im Internet Explorer 6.0

2.4 Vorformatierter Text

Gelegentlich ist es hilfreich, Text, der in einem HTML-Dokument notiert wurde, exakt so im Browser ausgeben zu lassen, z. B. bei Listings, Verzeichnislisten, der Darstellung von Messwerten und Ähnlichem. In einem solchen Fall sollten Sie das `pre`-Element (`pre` = engl. *preformatted*, dt. *vorformatiert*) verwenden. Text, der im Gültigkeitsbereich eines `pre`-Elements steht, wird außerdem mit einer nicht-proportionalen Schrift dargestellt.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<!-- Vorformatierter Text //-->
<html>
  <head>
    <title>Listing 2.4</title>
  </head>
  <body>
    <pre>Dieser Text ist vorformatiert
      Alle Zeilenumbrüche und Leerzeichen werden
        exakt so, wie im HTML-Dokument
      eingegeben, dargestellt.

```

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

Listing 2.4 Vorformatierter Text

Die Ausrichtung vorformatierten Textes lässt sich nicht mit dem `align`-Attribut oder einer anderen Methode beeinflussen, wohl aber die optische Darstellung (Farbe, Größe, Schriftart) mittels des `font`-Elements (das Sie später noch kennen lernen werden) oder CSS.

Sollten Sie jedoch den Wunsch haben, dass bei der Ausgabe des Textes alle Formatierungen mittels der Leerzeichen korrekt beibehalten werden, sollten Sie als Schriftart immer eine dicktengleiche Schrift wie Courier wählen und nicht etwa Arial oder Times New Roman.

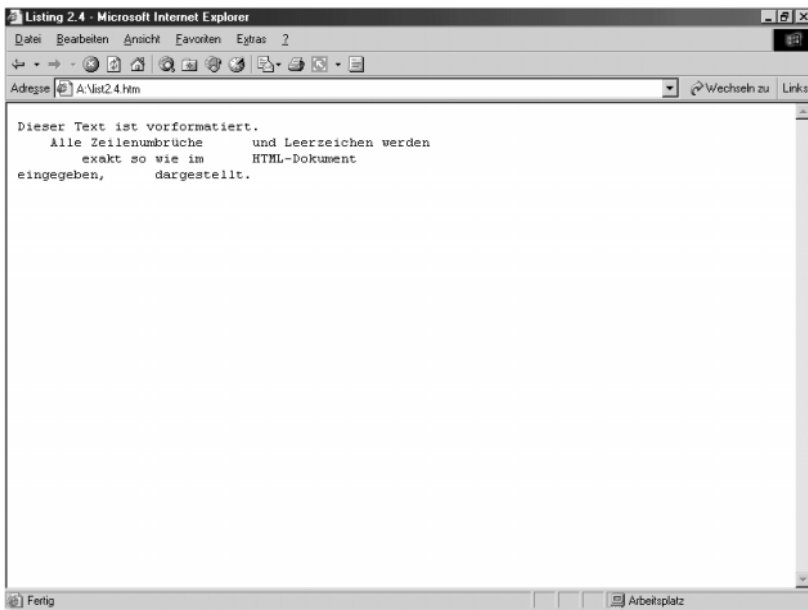


Abbildung 2.4 Vorformatierter Text

2.5 Überschriften

Für Überschriften gibt es in HTML sechs verschiedene Elemente, die sich in der Darstellung der Schrift unterscheiden. Die Bezeichnung der Elemente beginnt mit dem Buchstaben `h` (`h` = engl. *Header*, dt. *Titel*), gefolgt von einer Zahl zwischen 1 und 6. Dabei ist eine Überschrift mit dem `h1`-Element die größte und eine Überschrift mit dem `h6`-Element die kleinste.


```

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<!-- Beispiel für Überschriften in HTML //-->
<html>
  <head>
    <title>Listing 2.5</title>
  </head>
  <body>
    <h1>Überschrift 1. Ordnung</h1>
    <h2>Überschrift 2. Ordnung</h2>
    <h3>Überschrift 3. Ordnung</h3>
    <h4>Überschrift 4. Ordnung</h4>
    <h5>Überschrift 5. Ordnung</h5>
    <h6>Überschrift 6. Ordnung</h6>
  </body>
</html>

```

Listing 2.5 Überschriften in HTML

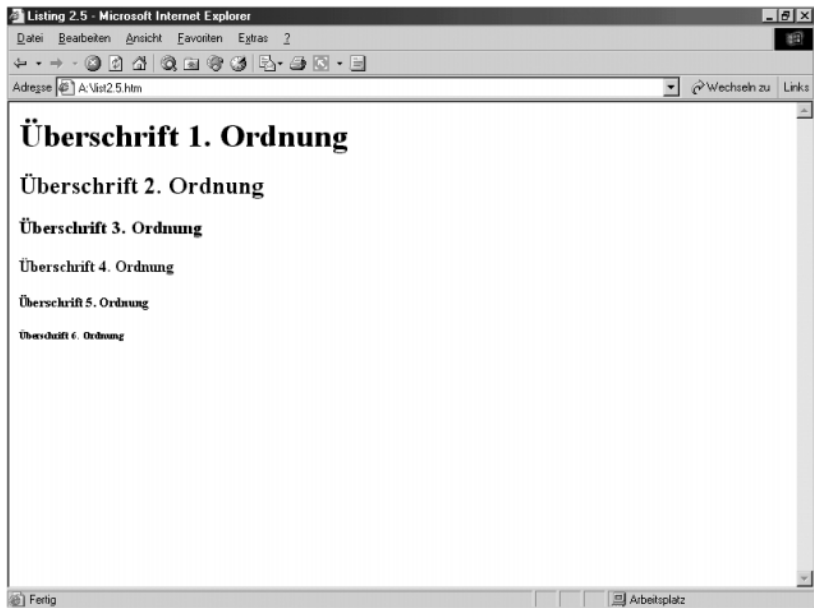


Abbildung 2.5 Listing 2.5 im Internet Explorer 6.0

Standardmäßig werden Überschriften im Browser linksbündig dargestellt. Mittels des `align`-Attributs können Sie dies wie beim `p`-Element jedoch beeinflussen.

Auch eine Ausrichtung mit Hilfe des `justify`-Parameters ist möglich, wobei auch hier wieder gilt: Die Überschrift muss sich über mindestens zwei Zeilen erstrecken, um als Blocksatz ausgerichtet werden zu können.



```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//
EN">
<!-- Beispiel für Überschriften in HTML //-->
<html>
  <head>
    <title>Listing 2.6</title>
  </head>
  <body>
    <h1 align="center">Zentrierte &Uuml;berschrift
      1. Ordnung</h1>
    <h2 align="left">Linksb&uuml;ndige &Uuml;berschrift
      2. Ordnung</h2>
    <h3 align="right">Rechtsb&uuml;ndige &Uuml;berschrift
      3. Ordnung</h3>
    <h4 align="justify">Blocksatz-&Uuml;berschrift
      4. Ordnung</h4>
  </body>
</html>
```

Listing 2.6 Überschriften mit unterschiedlicher Ausrichtung

Damit Sie die Ausrichtung des `h4`-Elements in Listing 2.6 als Blocksatz erkennen können, sollten Sie das Browserfenster einfach von Hand verkleinern.

2.6 Physische Textauszeichnung

Neben den verschiedenen Absatzformaten (Textabsätze, vorformatierte Absätze und Überschriften) gibt es spezielle Elemente, um Wörter, Sätze oder Absätze von anderen Texten hervorzuheben, z. B. fett, kursiv oder unterstrichen. Die Elemente zur Textauszeichnung werden als Inline-Elemente bezeichnet. Die Elemente für Absatzformate (`p`, `h1-6` und `pre`) gehören zu den Blockelementen. Der Unterschied zwischen diesen beiden Elementgruppen liegt darin, dass nach Blockelementen ein neuer Absatz eingeleitet wird. In den Abbildungen 2.1, 2.2 und 2.5 ist dies sehr gut zu erkennen. Inline-Elemente hingegen integrieren sich in den Textfluss und leiten keinen neuen Absatz ein.

Zehn Inline-Elemente

Insgesamt stellt Ihnen HTML zehn verschiedene Inline-Elemente zur Textauszeichnung zur Verfügung:

- ▶ `b` (engl. *Bold*, dt. *fett*) – zeichnet einen Text fett aus.
- ▶ `i` (engl. *Italic*, dt. *kursiv*) – zeichnet einen Text kursiv aus.
- ▶ `u` (engl. *Underlined*, dt. *unterstrichen*) – zeichnet einen Text unterstrichen aus.
- ▶ `s` oder `strike` (dt. *durchgestrichen*) – zeichnet einen Text durchgestrichen aus.
- ▶ `big` (dt. *groß*) – zeichnet einen Text größer aus.
- ▶ `small` (dt. *klein*) – zeichnet einen Text kleiner aus.
- ▶ `sup` (engl. *Superior*, dt. *hochgestellt*) – zeichnet einen Text hochgestellt aus.
- ▶ `sub` (dt. *tiefgestellt*) – zeichnet einen Text tiefgestellt aus.
- ▶ `tt` (engl. *Teletype*, dt. *Fernschreiber*) – zeichnet einen Text mit nicht proportionaler Schrift aus (wie in einem Fernschreiben).

Um die Verwendung der Inline-Elemente zu veranschaulichen, sehen Sie sich Listing 2.7 und Abbildung 2.6 an.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<!-- Elemente für physische Textauszeichnung //-->
<html>
  <head>
    <title>Listing 2.7</title>
  </head>
  <body>
    <p>Mit verschiedenen Inline-Elementen ist es
    möglich, Texte <b>fett</b>, <i>kursiv</i>, <u>unter-
    strichen</u>, <s>durchgestrichen</s>,
    <big>größer</big>, <small>kleiner</small>,
    <sup>hochgestellt</sup>, <sub>tiefgestellt</sub> oder
    <tt>nicht-proportional</tt> dazustellen.</p>
    <p>Auch beliebige Kombinationen dieser Inline-Elemente
    sind möglich, um Text <b><i>fettkursiv</i>
    </b>oder <big><s>größer und durchgestrichen
    </s></big> etc. darzustellen.</p>
```

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

Listing 2.7 Verschiedene Inline-Elemente zur physischen Textauszeichnung

In Listing 2.7 ist auch schon darauf eingegangen worden, dass sich die Elemente für die physische Textauszeichnung beliebig kombinieren lassen, wobei es keinen Sinn machen würde, einen Text mit zwei verschachtelten `b`-Elementen oder `strike`-Elementen auszuzeichnen, weil der Text dadurch nicht doppelt fett oder doppelt durchgestrichen dargestellt werden würde. Einen Text als hoch- und tiefgestellt (durch die Verschachtelung von `sup` und `sub`) oder größer und kleiner (`big` und `small`) auszuzeichnen, macht genauso wenig Sinn.

Achten Sie bei der Sprachvariante `strict` darauf, dass die vorangestellten Elemente innerhalb eines Blockelements verwendet werden müssen und nicht einfach so im Gültigkeitsbereich des `body`-Elements, wie es in der Variante `transitional` erlaubt wäre.

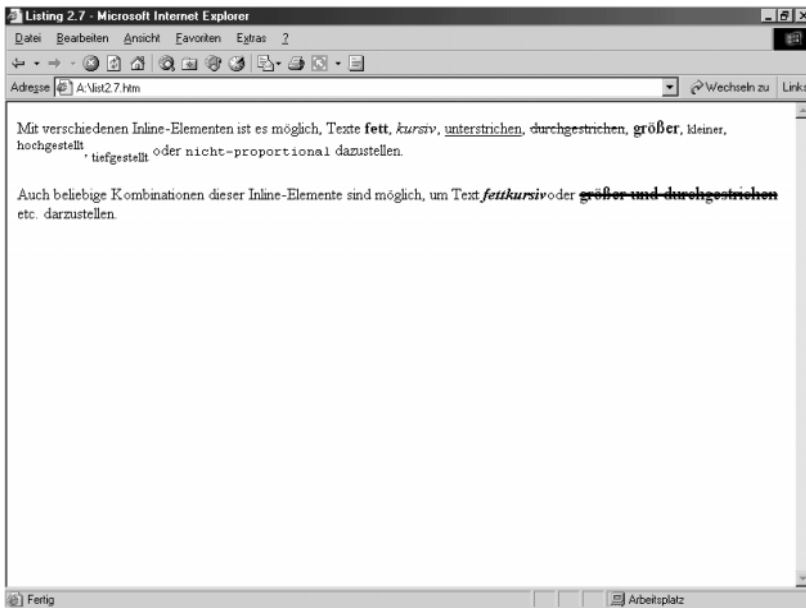


Abbildung 2.6 Darstellung des Listings 2.7 im Internet Explorer 6.0

2.7 Logische Textauszeichnung

Ähnlich der physischen Textauszeichnung ist es mit logischer Textauszeichnung möglich, Texte und Textbausteine hervorzuheben. Dabei ist die optische Darstellung nicht explizit festgelegt und kann von Browser zu Browser (selbst in den einzelnen Versionen) variieren. Auch für diesen Fall stellt Ihnen HTML zehn verschiedene Inline-Elemente zur Verfügung.

- ▶ `abbr` – zeichnet einen Text als abgekürzte Schreibweise aus.
- ▶ `acronym` – zeichnet einen Text als Abkürzung aus.
- ▶ `cite` – zeichnet einen Text als Zitat aus.
- ▶ `code` – zeichnet einen Text als Quelltext aus.
- ▶ `dfn` – zeichnet einen Text als Definition aus.
- ▶ `em` – zeichnet einen Text emphatisch aus.
- ▶ `kbd` – zeichnet einen Text als Tastatureingabe aus.
- ▶ `samp` – zeichnet einen Text als Beispiel aus.
- ▶ `strong` – zeichnet einen Text stärker betont aus.
- ▶ `var` – zeichnet einen Text als Variable/variabel aus.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<!-- Elemente für logische Textauszeichnung //-->
<html>
  <head>
    <title>Listing 2.8</title>
  </head>
  <body>
    <p>Es folgen Beispiele für logische Textauszeichnung: <abbr>abgekürzte Schreibweise</abbr>, <acronym>Abkürzung</acronym>, <cite>Zitat</cite>, <code>Quellcode</code>, <dfn>Definition</dfn>, <em>emphatisch</em>, <kbd>Tastatureingabe</kbd>, <samp>Beispiel</samp>, <strong>stark betont</strong> und <var>Variable/variabler Text</var>.</p>
    <p>Auch eine Kombination ist möglich: <kbd><strong>stark betonte Tastatureingabe</strong></kbd>.</p>
  </body>
</html>
```

Listing 2.8 Beispiele für logische Textauszeichnungen

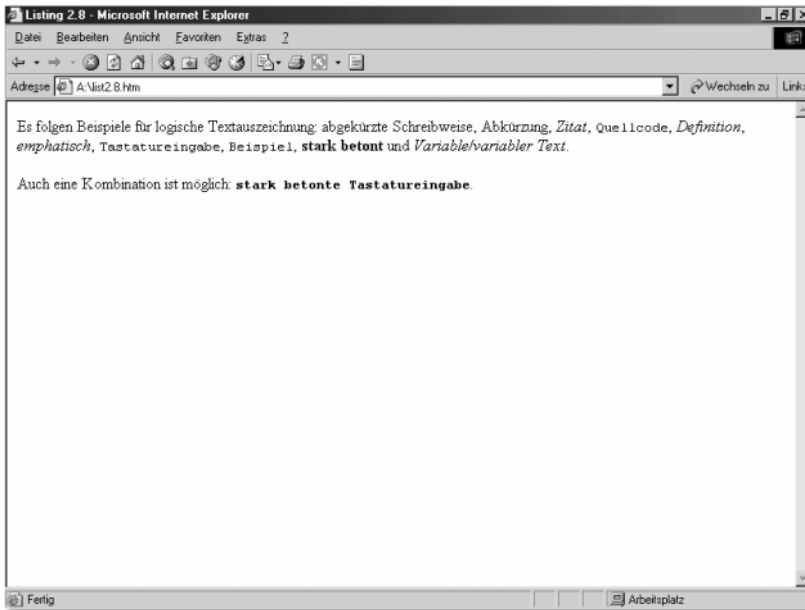


Abbildung 2.7 Darstellung des Listings 2.8 im Internet Explorer 6.0

Auch logische Textauszeichnungen dürfen miteinander verschachtelt werden. Dies macht in vielen Kombinationen aber keinen Sinn, und Sie sollten deshalb davon Abstand nehmen (auch wenn ich Ihnen diese Möglichkeit aufgezeigt habe). Die Regeln für die HTML-Varianten *strict* und *transitional* entsprechen denen der physischen Textauszeichnungen und sollten eingehalten werden, um eine korrekte Darstellung gewährleisten zu können.

Ebenfalls möglich ist eine Kombination der logischen und physischen Textauszeichnungen mit Überschriften. So ließe sich eine Überschrift 1. Ordnung (`<h1>...</h1>`) auch mit der Textauszeichnung kursiv (`<i>...</i>`) oder Tastatureingabe (`<kbd>...</kbd>`) kombinieren, um nur ein paar Beispiele zu nennen.

2.8 Schriftformatierung

Die bereits bekannten Elemente `h1-6`, `p` und `pre` lassen sich auch mit einem weiteren Element nach Ihren Wünschen verändern und anpassen. Anstatt den Texten mittels physischer oder logischer Textauszeichnung ein bereits definiertes Aussehen zuzuweisen, können Sie mit Hilfe des `font`-Elements die Schriftgröße, die Schriftfarbe und die Schriftart individuell anpassen.

```

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<!-- Schriftformatierung //-->
<html>
  <head><title>Listing 2.9</title>
  </head>
  <body>
    <h1><font face="sans-serif" size="7">Schriftformatierung</font></h1>
    <p><font size="+1" color="blue" face="cursive">Umformatierter Absatz mit groesserer Schrift, blauer Farbe und einer Schreibschrift.</font></p>
  </body>
</html>

```

Listing 2.9 Schriftformatierungen

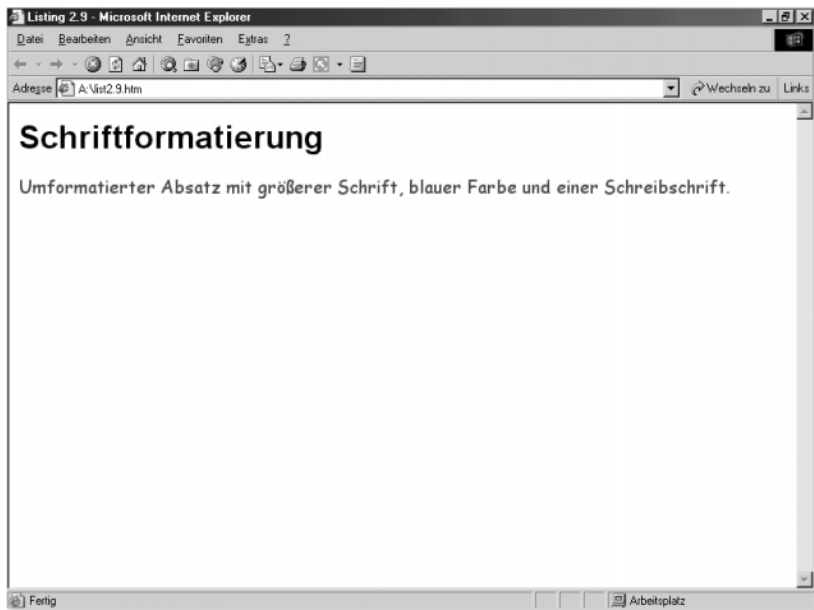


Abbildung 2.8 Darstellung des Listings 2.9 im Internet Explorer 6.0



In Abbildung 2.8 können Sie deutlich die Auswirkung des `font`-Elements auf die anderen Elemente erkennen. Es sei jedoch gesagt, dass das `font`-Element als deprecated gekennzeichnet ist. Das W3C plant, diese Elemente in kommenden HTML-Version nicht mehr einzusetzen.

2.8.1 Vererbung

Das Besondere bei HTML und den Elementen ist die so genannte Vererbung. Bereits zu Beginn des ersten Kapitels sprach ich von Eltern- und Kindelementen. Ein Element, das im Gültigkeitsbereich eines anderen Elements notiert wurde, ist das Kindelement. Das Element, welches das Kindelement enthält, ist das Elternelement.

Ähnlich wie bei der Beziehung zwischen Eltern und Kindern, erben auch die Kindelemente in HTML die Eigenschaften ihrer Elternelemente – im bildlichen Sinne erben Sie die Gene. Dabei erhalten sie alle Eigenschaften, die das Elternelement besitzt, und erweitern oder verändern diese Eigenschaften, z. B. durch eine andere Schriftdicke, Schriftgröße oder Schriftart.

**Du hast die Augen
deiner Mutter**

Exakt so verhält es sich auch mit den Textauszeichnungen (physischen und logischen): Der Text im Gültigkeitsbereich eines Textabsatzes wird standardmäßig in der Schriftart Times New Roman (beim Internet Explorer) und einer absoluten Schriftgröße von 3 dargestellt. Zeichnen Sie einen bestimmten oder den gesamten Teil des Textes mit dem `i`-Element (kursiv) aus, behält der Text zwar seine Eigenschaften wie Schriftgröße und -art bei (Times New Roman, Größe 3), erhält aber zusätzlich die Eigenschaft kursiv.

```
<p><i>Textabsatz</i></p>  
<!-- Stellt Absatz kursiv dar //-->
```

Würden Sie den zuvor in kursiv geänderten Text anstelle dessen mit dem Element `kbd` (Tastatureingabe) auszeichnen, würden Sie keine neue Eigenschaft hinzufügen, sondern die vorhandene Eigenschaft Schriftart von Times New Roman in Courier New verändern.

```
<p><kbd>Textabsatz</kbd></p>  
<!-- Stellt Absatz mit Courier New dar //-->
```

Würden Sie nun innerhalb der Elemente `i` oder `kbd` weitere Elemente für die Textauszeichnung notieren, würden auch diese Elemente die Eigenschaften ihres Elternelements erben.

```
<p><kbd>Text<b>absatz</b></kbd></p>  
<!-- Stellt Absatz mit Courier New dar und das Teil-  
wort absatz zusätzlich in fett //-->
```

Um nun die Schrift eines Elements anzupassen, kennt das `font`-Element drei verschiedene Attribute:

Attribut	Erklärung
face	Ändert das »Gesicht« bzw. die Schriftart des Textes.
size	Ändert die Schriftgröße des Textes.
color	Ändert die Farbe, in welcher der Text dargestellt wird.

Tabelle 2.2 Attribute des font-Elements

2.8.2 Schriftart

Mit dem Attribut `face` können Sie die eingestellte Standard-Schriftart des Elements verändern. Sie können sowohl den Namen einer installierten Schriftart angeben (z. B. Times New Roman oder Arial) als auch einen Schrifttyp angeben: `serif` (Serifenschrift wie Times New Roman), `sans-serif` (serifenlose Schrift wie Arial), `cursive` (Schreibschrift wie Comic Sans), `fantasy` (Fantasieschrift) oder `monospace` (nicht-proportionale Schrift wie Courier New).



Die Angabe eines Schrifttyps (also `serif` oder `monospace`) ist eindeutig sinnvoller, da nicht auf jedem Rechner die gleichen Schriftarten installiert sind. Dies ist vor allem abhängig vom Betriebssystem. Ist die von Ihnen gewünschte Schriftart nicht installiert, wird die Standard-Schriftart verwendet, was zu unschönen Ergebnissen führen kann. Geben Sie aber lediglich einen Schrifttyp an, kann der Browser eine passende Schriftart heraussuchen und diese verwenden.



Sie sollten nun aber nicht anfangen, Schriftarten auf Ihrer Homepage zum Download anzubieten, damit der Benutzer sie herunterladen und installieren kann. Zum einen kann das für Sie rechtliche Konsequenzen bedeuten (Schriftarten sind Copyright-geschützt), und zum anderen lässt sich nicht jede Schriftart auf jedem Betriebssystem installieren.

Ein Beispiel für die Angabe einer Schriftart könnte lauten

```
<font face="Times New Roman">Angabe einer Schriftart
</font>
```

und für die Angabe eines Schrifttyps

```
<font face="cursive">Angabe eines Schrifttypes</font>
```

2.8.3 Schriftgröße

Bei der Wahl der Schriftgröße stehen Ihnen zwei Möglichkeiten zur Verfügung: absolut oder relativ. Möchten Sie die Schriftgröße absolut ange-

ben, müssen Sie als Wert eine Zahl von 1 bis 7 mittels des Attributs `size` übergeben (wobei 7 am größten und 1 am kleinsten ist). Bei einer relativen Angabe beziehen Sie sich auf die Standard-Schriftgröße und erhöhen oder verringern diese durch das Voranstellen eines + (Plus) oder eines – (Minus) und der Zahl, um welche die Größe erhöht oder verringert werden soll.

Möchten Sie also die Schriftgröße eines Textes in die absolute Größe 5 umwandeln, müssen Sie das folgende Konstrukt verwenden:

```
<font size="5">Text wird in der Größe 5 dargestellt</font>
```

Um die Schriftgröße um eine Stufe zu erhöhen, z. B. von 3 auf 4, würde die Notation folgendermaßen aussehen:

```
<font size="+1">Textgröße wird um 1 erhöht</font>
```

Und um den Text beispielsweise um zwei Stufen zu verringern, würden Sie Folgendes notieren:

```
<font size="-1">Textgröße wird um 2 verringert</font>
```

2.8.4 Schriftfarbe

Mit dem `color`-Attribut können Sie die Farbe der Schrift verändern. Auch hier stehen Ihnen wieder zwei Möglichkeiten zur Verfügung: die Angabe eines der 16 Farbworte oder eines Hex-Tripel-Werts. Farbworte sind `black`, `blue`, `red` usw. Das Basiskonstrukt lautet:

```
<font color="Farbe">Text</font>
```

Um einem Text die Farbe Grün (engl. *green*) mittels eines Farbwortes zuzuweisen, müssten Sie Folgendes notieren: **Farbworte**

```
<font color="green">Textfarbe Grün</font>
```

Äquivalent verhält sich dies mit allen anderen Farbwörtern, nur dass als Parameter das entsprechende Farbwort angegeben werden muss. Nachfolgend finden Sie eine Tabelle mit den 16 wichtigsten Farbwörtern. Die vollständige Liste ist im Anhang untergebracht.

HTML-Farbwort	Beschreibung
Black	Schwarz
Maroon	Kastanienbraun, dunkles Rot
Green	Grün
Olive	Olivgrün
Navy	Marineblau
Purple	Purpur
Teal	Dunkles Türkis
Gray	Grau
Silver	Silber
Red	Rot
Lime	Limone/Hellgrün
Yellow	Gelb
Blue	Blau
Fuchsia	Violett
Aqua	Hellblau
White	Weiß

Tabelle 2.3 Die wichtigsten HTML-Farbwoorte und ihre Beschreibung

```
<font color="red">Textfarbe Gr&uuml;n</font>
<font color="purple">Textfarbe Purpur</font>
<font color="aqua">Textfarbe Hellblau</font>
```

Hex-Tripel-Werte Die Angabe eines Hex-Tripel-Werts ist ein wenig komplizierter. Anstelle eines der Farbwoorte zu verwenden (und somit eine vordefinierte Farbe), wird die gewünschte Farbe zusammengemischt. Dabei wird das für Computer übliche RGB-System verwendet. Die Abkürzung RGB steht für die drei Grundfarben **R**ot, **G**rün und **B**lau. Die Farben werden additiv gemischt, und je höher der jeweilige Anteil der Farben ist, desto heller ist der daraus resultierende Farbton. Äquivalent dazu: Je geringer die einzelnen Farbanteile, desto dunkler ist das Resultat.

Wertebereich Der Wertebereich der roten, grünen und blauen Anteile liegt jeweils zwischen 0 und 255 (8 Bit). Da Sie für jede Grundfarbe einen Maximalwert von 255 einsetzen dürfen, stehen uns 16.581.375 Farben zur Verfügung

(255 * 255 * 255), wesentlich mehr also, als die Angabe eines Farbworts ermöglicht.

Um nun eine Farbe zu mischen, müssen Sie den jeweiligen Farbanteil in hexadezimalen Zahlen angeben. Anders als beim dezimalen Zahlensystem mit der Basis 10, besitzt das hexadezimale Zahlensystem die Basis 16. Die zusätzlichen 6 Zahlen werden durch die Buchstaben von A bis F (entspricht dezimal den Zahlen 10 bis 15) repräsentiert. Die Zahlenreihe lautet also 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E und F. Für eine hexadezimale Angabe stehen Ihnen zwei Stellen zur Verfügung. Der Dezimalwert 0 würde in der entsprechenden Schreibweise 00 lauten und der Dezimalwert 255 lautet FF. Dies mag am Anfang zwar etwas kompliziert klingen, es ist aber relativ einfach.

Farbe mischen

Achten Sie darauf, immer zwei Stellen pro Farbanteil zu notieren. Falls der gewünschte Farbanteil einmal nur eine Stelle besitzt, notieren Sie davor eine 0, um auf jeden Fall auf zwei Stellen zu kommen. Ansonsten wird der Text entweder überhaupt nicht farbig dargestellt oder in einer falschen Farbe.



Ich werde Ihnen nun ein paar Beispiele durchrechnen. Ich möchte die Dezimalzahl 163 in eine Hexadezimalzahl umrechnen. Ich weiß, dass Hexadezimalzahlen als Basis den Wert 16 verwenden. Teile ich also 163 durch 16, erhalte ich das Ergebnis 10,1875. Der ganzzahlige Anteil der Division lautet 10, und das wird in hexadezimaler Schreibweise durch A dargestellt. Dies ist die erste Stelle. Der Rest aus der Division (163/16) beträgt 3, die im hexadezimalen System genauso notiert wird. Dies ist die zweite Stelle. Die Umrechnung von 163 in eine hexadezimale Zahl lautet also A3.

Rechenbeispiel

So würde die Dezimalzahl 250 als Hexadezimalzahl z. B. FA lauten ($250/16 = 15$ Rest 10).

Andersherum lässt sich eine hexadezimale Zahl auch wieder in eine dezimale Zahl umrechnen. Nehmen Sie als Beispiel den hexadezimalen Wert BC. Zuerst wandeln Sie die erste Stelle (Zahl B) in eine für Sie verwendbare Zahl um und multiplizieren sie mit 16. Da B der dezimalen Zahl 11 entspricht, rechnen Sie also $11 * 16$ und erhalten das Ergebnis 176. Dann wandeln Sie die zweite Stelle (Zahl C) in eine dezimale Zahl um und addieren sie zu der umgerechneten ersten Stelle. C steht für die dezimale Zahl 12. Das Ergebnis lautet also 188 ($176+12$).

Schreibweise Damit hexadezimale Zahlen von dezimalen zu unterscheiden sind, werden sie mit # (Doppelkreuz oder Raute) eingeleitet. Danach folgen in der Reihenfolge rot, grün und blau die hexadezimalen Zahlen:

#RRGGBB

Für die Farbe Gelb (Rot = 255, Grün = 255 und Blau = 0) würde die entsprechende hexadezimale HTML-Notation #FFFF00 lauten (255 = FF und 0 = 00). Die entsprechenden Hex-Tripel-Werte zu den HTML-Farbworten finden Sie in Tabelle 2.4.

HTML-Farbwort	Hex-Tripel-Wert
Black	#000000
Maroon	#7F0000
Green	#007F00
Olive	#7F7F00
Navy	#00007F
Purple	#7F007F
Teal	#007F7F
Gray	#7F7F7F
Silver	#BFBFBF
Red	#FF0000
Lime	#00FF00
Yellow	#FFFF00
Blue	#0000FF
Fuchsia	#FF00FF
Aqua	#00FFFF
White	#FFFFFF

Tabelle 2.4 HTML-Farbworter als Hex-Tripel-Werte

Graustufen bzw. Grautöne erzeugen Sie durch gleich hohe Farbanteile. Dadurch stehen Ihnen insgesamt 256 verschiedene Grautöne zur Verfügung. Ein dunkles Grau könnte lauten #303030, ein mittleres Grau #868686 und ein helles Grau #CBCBCB. Um nun die Hex-Tripel-Werte zu verwenden, nehmen Sie die Konstrukte (inkl. #-Zeichen) anstelle eines

Farbwortes als Parameter für das `color`-Attribut. Es folgen verschiedene Beispiele:

```
<font color="#303030">Dunkles Grau</font>
<font color="#868686">Mittleres Grau</font>
<font color="#CBCBCB">Helles Grau</font>
<font color="#00BB00">Gr&uuml;n</font>
<font color="#003399">Gr&uuml;nblau</font>
```

2.9 Dateiweite Farben

Auch eine Hintergrundfarbe lässt sich für ein HTML-Dokument festlegen. Standardmäßig ist die Farbe auf weiß oder grau gesetzt bzw. richtet sich danach, wie der Benutzer die Einstellungen in seinem Browser angepasst hat. Und auch die Farbe der Hyperlinks in einem HTML-Dokument kann dateiweit an die eigenen Wünsche und Vorstellungen angeglichen werden, genauso wie die Textfarbe.

2.9.1 Hintergrund und Textfarbe

Zum Anpassen des Hintergrunds wird das Attribut `bgcolor` (engl. *background color*, dt. *Hintergrundfarbe*) verwendet. Es wird im Start-Tag des `body`-Elements notiert und bekommt als Parameter die gewünschte Farbe übergeben. Dabei ist es egal, ob Sie die Farbangabe als Farbwort oder Hex-Tripel-Wert formulieren. HTML akzeptiert beides.

```
<body bgcolor="blue">
  <!-- blauer Hintergrund //-->
<body bgcolor="#00FF00">
  <!-- grüner Hintergrund //-->
```

Für die Textfarbe wird das Attribut `text` verwendet, das ebenfalls im `body`-Start-Tag notiert wird. Auch hier sind Farbangaben in Farbworten oder Hex-Tripel-Werten gültig.

```
<body text="red">
  <!-- rote Textfarbe //-->
<body text="#00FFFF">
  <!-- gelbe Textfarbe //-->
```

2.9.2 Hyperlink-Farben

Ein Hyperlink kennt drei verschiedene Zustände: normal, bereits besucht oder momentan aktiv. Jeder dieser drei Zustände wird im Dokument

durch eine andere Farbe dargestellt. Standardmäßig sind Hyperlinks blau, besuchte Links violett und aktive Links rot. Mittels verschiedener Attribute lassen sich auch diese Farben verändern:

Attribut	Übersetzung	Erklärung
link	Verknüpfung	Ändert die Farbe für normale Hyperlinks.
vlink	Besuchte Verknüpfung	Ändert die Farbe für bereits besuchte Hyperlinks.
alink	Aktive Verknüpfung	Ändert die Farbe eines Hyperlinks der gerade aktiven Seite.

Tabelle 2.5 Attribute zur farblichen Anpassung von Hyperlinks

Alle drei Attribute akzeptieren Farbangaben als Farbwort oder Hex-Tripel-Wert und werden im Start-Tag des `body`-Elements notiert.

```
<body link="green" vlink="blue" alink="#000000">  
  <!-- Hyperlinks sind nun grün, besuchte Hyperlinks blau  
  und aktive Hyperlinks schwarz //-->
```



Natürlich müssen Sie nicht immer alle drei Attribute angeben. Es reicht die Angabe der Attribute aus, die Sie auch wirklich ändern möchten.

2.10 Listen

Oftmals möchte man eine übersichtliche Dateiliste, die Eigenschaften eines Produkts enthalten, oder ein Inhaltsverzeichnis erstellen. Zwar ließe sich das auch mit einem Textabsatz und mehreren Zeilenumbrüchen realisieren, aber für genau diesen Fall bietet HTML verschiedene Elemente an. Damit ist es möglich, schnell und unkompliziert Aufzählungslisten, nummerierte Listen, Definitions- oder Menülisten zu erstellen.

2.10.1 Nummerierte Listen

Typischerweise werden nummerierte Listen für Inhaltsverzeichnisse, Schritt-für-Schritt-Erklärungen oder Hierarchien verwendet. Das dafür zu verwendende Element ist `ol` (engl. *ordered list*, dt. *sortierte Liste*). Alle Einträge einer solchen Liste werden automatisch durchnummeriert. Ein Beispiel:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
```

```

<!-- Nummerierte Listen //-->
<html>
  <head>
    <title>Listing 2.10</title>
  </head>
  <body>
    <h1>Bundesligatabelle 2001/2002</h1>
    <ol>
      <li>Borussia Dortmund</li>
      <li>Bayer Leverkusen</li>
      <li>Bayern M&uuml;nchen</li>
      <li>Hertha BSC Berlin</li>
      <li>FC Schalke 04</li>
      <li>Werder Bremen</li>
      <li>1. FC Kaiserslautern</li>
      <li>VfB Stuttgart</li>
      <li>1860 M&uuml;nchen</li>
      <li>VfL Wolfsburg</li>
      <li>Hamburger SV</li>
      <li>Borussia M&ouml;nchengladbach</li>
      <li>Energie Cottbus</li>
      <li>Hansa Rostock</li>
      <li>1. FC N&uuml;rnberg</li>
      <li>SC Freiburg</li>
      <li>1. FC K&ouml;ln</li>
      <li>FC St. Pauli</li>
    </ol>
  </body>
</html>

```

Listing 2.10 Beispiel für eine nummerierte Liste

Das Listing 2.10 zeigt ein Beispiel für eine nummerierte Liste (als Grundlage wurde der letzte Tabellenstand der Fußballsaison 2001/2002 verwendet). Wenn Sie sich das Listing im Browser ansehen, können Sie feststellen, dass die einzelnen Einträge von oben nach unten mit der Nummer 1 beginnend durchnummeriert wurden.

Zuerst wurde die Liste mit dem Start-Tag des `ol`-Elements eingeleitet. Die einzelnen Einträge der Liste werden mit Hilfe des Elements `li` (engl. list item, dt. Listeneintrag) definiert. Auch hier wird im Gültigkeitsbereich des Elements der darzustellende Text notiert. Die Nummerierung erfolgt

automatisch, sodass im `li`-Element lediglich die Bezeichnung des Listeneintrags steht. Zum Schluss wird die Listendefinition mit `` beendet.

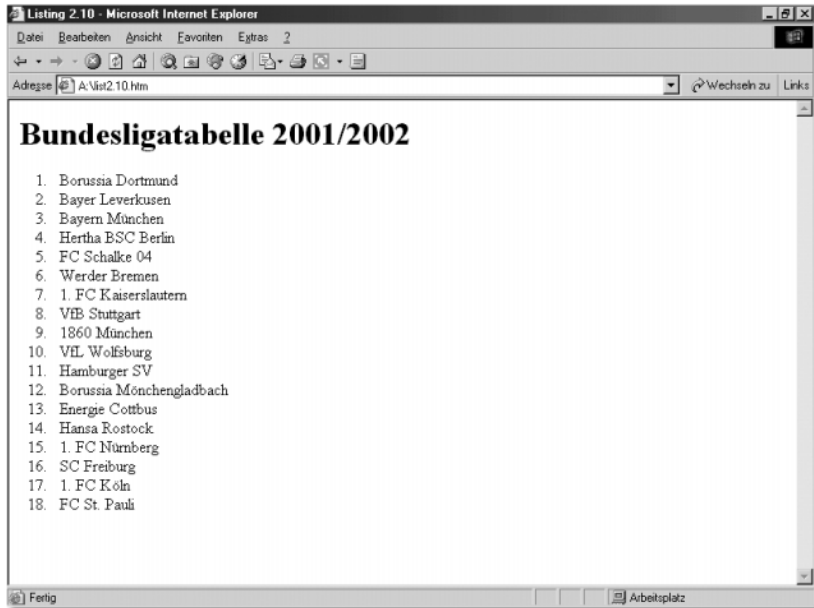


Abbildung 2.9 Listing 2.10 im Internet Explorer 6.0

Zusätzlich zum Text im Gültigkeitsbereich eines Listeneintrags wäre es durchaus möglich, eine weitere Liste zu definieren. Gerade bei Inhaltsverzeichnissen würde dies Sinn machen. Leider wird in HTML aber nicht konsequent nummeriert, sodass bei einer verschachtelten Liste keine – wie man eigentlich erwarten könnte – Nummerierungsfolge wie 1, 1.1, 1.1.1 möglich ist. Die Nummerierung beginnt für die untergeordnete Liste dann einfach wieder bei 1.



Um eine solche verschachtelte Liste zu definieren, müssen Sie zusätzlich zum Text eines `li`-Elements einfach eine weitere Liste notieren.

```
<ol>
  <li>Grundlagen
    <ol>
      <li>Am Anfang war HTML...</li>
      <li>HTML-Versionen</li>
    </ol>
  </li>
</ol>
```

Die Ausgabe dieses kurzen Beispiels könnte folgendermaßen aussehen:

1. Grundlagen
 1. Am Anfang war HTML...
 2. HTML-Versionen

2.10.2 Aufzählungslisten

Im Gegensatz zu einer nummerierten Liste wird bei einer Aufzählungsliste nicht automatisch durchnummeriert, sondern ein Aufzählungszeichen (engl. *bullet*) vor jedem Eintrag im Browser dargestellt. Diese Listen finden besonders dann Anwendung, wenn Produkteigenschaften oder Ähnliches aufgelistet werden sollen, die nicht zwingend nummeriert sein müssen. Das Element für Aufzählungslisten ist `ul` (engl. *unordered list*, dt. *unsortierte Liste*).

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<!-- Aufzählungslisten //-->
<html>
  <head>
    <title>Listing 2.11</title>
  </head>
  <body>
    <h1>Einkaufsliste</h1>
    <ul>
      <li>Essen
        <ul>
          <li>Brot</li>
          <li>Käse</li>
          <li>Aufschnitt</li>
          <li>Senf</li>
        </ul>
      </li>
      <li>Getränke
        <ul>
          <li>Cola</li>
          <li>Limonade</li>
          <li>Saft</li>
        </ul>
      </li>
      <li>Weiterer Eintrag...</li>
    </ul>
  </body>
</html>
```

```

        <li>Weiterer Eintrag...</li>
        <li>Weiterer Eintrag...</li>
    </ul>
</body>
</html>

```

Listing 2.11 Einkaufsliste mit einer Aufzählungsliste

Die Art und Darstellung des Aufzählungszeichens zwischen den einzelnen Browsern ist unterschiedlich. Wie bei einer nummerierten Liste wird auch bei einer Aufzählungsliste ein Listeneintrag mit dem `li`-Element definiert, und auch die Verschachtelung ist möglich. Doch wird das Bullet einer Unterliste anders dargestellt als das der Hauptliste.

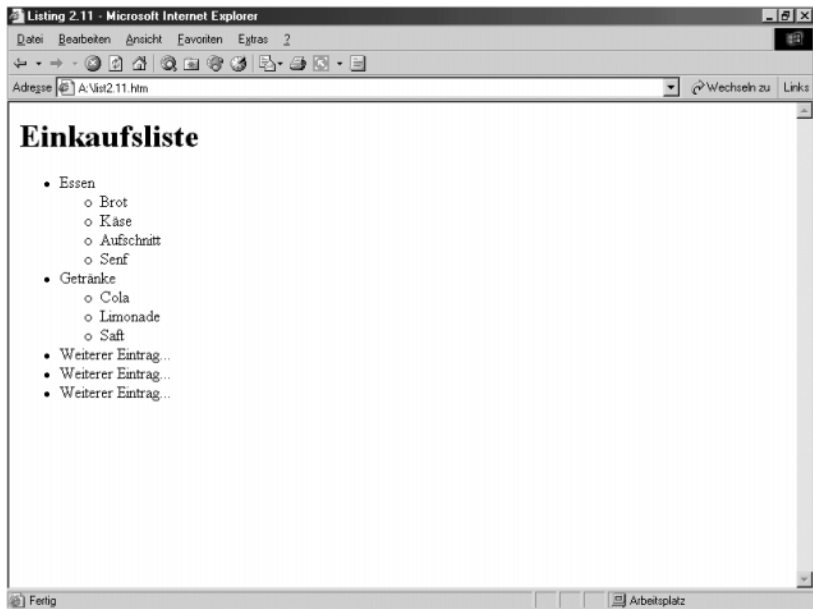


Abbildung 2.10 Darstellung des Listings 2.11 im Internet Explorer 6.0



Auch die einzelnen Bullets einer verschachtelten Aufzählungsliste hängen vom Browser ab und sind nicht festgeschrieben.

2.10.3 Definitionenlisten

Obwohl Definitionenlisten für das Erstellen von Glossaren (Erklärung verschiedener Abkürzungen, Fachbegriffe und Fremdwörter) verwendet werden sollen, lassen sie sich für unterschiedliche Dinge verwenden, z. B.

für Menülisten mit zusätzlichen Erklärungen oder als Vokabelliste für Fremdsprachen. Anders als bei den vorherigen Listen (nummerierte und Aufzählungslisten) werden drei verschiedene Elemente zur Definition verwendet:

- ▶ `dl` (engl. *definition list*, dt. *Definitionsliste*) – ist das Hauptelement einer Definitionsliste und enthält die Elemente `dt` und `dd`.
- ▶ `dt` (engl. *definition list term*, dt. *Definitionsbegriff*) – enthält den zu erklärenden Begriff.
- ▶ `dd` (engl. *definition list definition*, dt. *Definition*) – enthält die Definition des zuvor angeführten Begriffs.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//
EN">
<!-- Definitionslisten //-->
<html>
  <head>
    <title>Listing 2.12</title>
  </head>
  <body>
    <h1>Abk&uuml;rungen in diesem Buch</h1>
    <dl>
      <dt>IE</dt><dd>Microsoft Internet Explorer 1 - 6</dd>
      <dt>NN</dt><dd>Netscape Navigator 1 - 5</dd>
      <dt>N6</dt><dd>Netscape 6</dd>
      <dt>OP</dt><dd>Opera 1 - 6</dd>
      <dt>MZ</dt><dd>Mozilla 1</dd>
    </dl>
  </body>
</html>
```

Listing 2.12 Beispiel für eine Definitionsliste

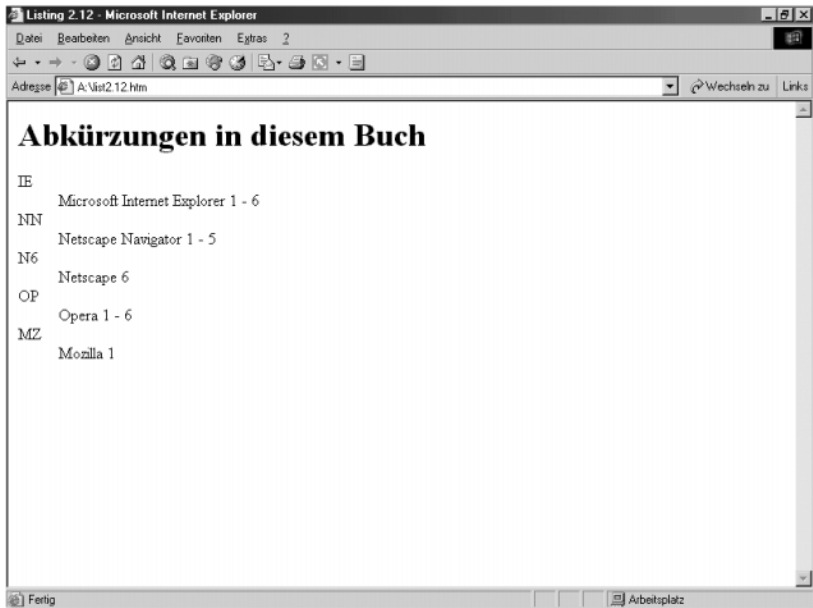


Abbildung 2.11 Darstellung des Listings 2.12 im Internet Explorer 6.0

Zuerst wird die Definitionsliste mit dem `dl`-Element eingeleitet. Das bereits bekannte Element `li` wird bei Definitionslisten nicht verwendet, sondern die Elemente `dt` für den zu definierenden Begriff und `dd` für die entsprechende Erklärung. Reihenfolge und Anzahl dieser Elemente sind innerhalb einer Definitionsliste (`dl`) nicht fest vorgeschrieben. Sie können also mehrere Begriffe oder Definitionen hintereinander notieren, wenn es zu einem Begriff beispielsweise zwei oder mehr Definitionen gibt. Durch die Verschachtelung von Definitionslisten ist es außerdem möglich, eine Baumstruktur aufzubauen.

```
<dl>
  <dt>IE</dt>
  <dd>Microsoft Internet Explorer 4</dd>
  <dd>Microsoft Internet Explorer 5.x</dd>
  <dd>Microsoft Internet Explorer 6</dd>
  <dt>NN</dt>
  <dd>Netscape Navigator 3</dd>
  <dd>Netscape Navigator 4.x</dd>
  <dt>N6</dt>
  <dd>Netscape 6</dd>
```

```
<dd>Netscape 7</dd>
</dl>
```

Im Browser könnte dies dann folgendermaßen aussehen:

```
IE
  Microsoft Internet Explorer 4
  Microsoft Internet Explorer 5.x
  Microsoft Internet Explorer 6
NN
  Netscape Navigator 3
  Netscape Navigator 4.x
N6
  Netscape 6
  Netscape 7
```

2.10.4 Listenattribute

Speziell für Listen stellt HTML einige Attribute zur Verfügung, um die Darstellung der Listen beeinflussen zu können. So ist es beispielsweise möglich, das Aufzählungszeichen einer Liste zu ändern, die Nummerierung ab einem bestimmten Wert zu starten oder die Art der Nummerierung festzulegen.

Die Attribute sind jedoch als deprecated (dt. ablehnend) eingestuft und sollten daher durch CSS ersetzt werden. Der Vollständigkeit halber werden sie an dieser Stelle trotzdem noch einmal aufgeführt.



Das Bullet einer Aufzählungsliste wird mittels des Attributs `type` im Start-Tag der Liste (``) verändert. Dabei stehen Ihnen drei Typen zur Verfügung:

Bullet ändern

Parameter	Erklärung
circle	Die Bullets der Aufzählungsliste werden rund.
disc	Die Bullets der Aufzählungsliste werden in Form eines Datenträgers dargestellt.
square	Die Bullets der Aufzählungsliste werden eckig dargestellt.

Tabelle 2.6 Mögliche Parameter für das Attribut `type`

```
<ul type="square">
  <!-- eckiges Bullet //-->
<ul type="disc">
```

```
<!-- Bullet in Form e. Datenträgers, häufig eine CD //-->
```



Bei verschachtelten Listen wird für untergeordnete Listen das neu angegebene Bullet jedoch nicht übernommen. Sie müssen also für jede Liste das Bullet explizit angeben.

Nummerierung ändern

Die Art der Nummerierung einer nummerierten Liste wird ebenfalls mittels des Attributs `type` bestimmt. Die verschiedenen Parameter finden Sie in Tabelle 2.7:

Parameter	Erklärung
I	Große römische Nummerierung, z. B. I, II, III, IV
i	Kleine römische Nummerierung, z. B. i, ii, iii, iv
A	Alphanumerische Nummerierung in Großbuchstaben, z. B. A, B, C, D
a	Alphanumerische Nummerierung in Kleinbuchstaben, z. B. a, b, c, d

Tabelle 2.7 Mögliche Parameter für das Attribut `type`

```
<ol type="I">
  <!-- große römische Nummerierung //-->
<ol type="a">
  <!-- kleine alphanumerische Nummerierung //-->
```



Auch hier gilt wieder bei verschachtelten Strukturen, dass die Art der Aufzählung für untergeordnete Listen nicht übernommen wird.

Startwert setzen

Um die Nummerierung einer Liste mit einem bestimmten Wert zu beginnen, müssen Sie im Start-Tag des `ol`-Elements das Attribut `start` setzen und als Parameter den gewünschten Startwert angeben. Für eine Nummerierung mit dem Startwert 3 lautet die Angabe:

```
<ol start="2">
```

Um die Nummerierung innerhalb einer Liste zu verändern, müssen Sie im Start-Tag des `li`-Elements das Attribut `value` mit dem Startwert als Parameter angeben. Für einen Startwert von 6 lautet die Angabe:

```
<li value="6">
```

2.11 Zusammenfassung

- ▶ Für einfache Textabsätze wird in HTML das Element `p` verwendet. Zum Darstellen von Listings, Verzeichnislisten oder Messwerten wird das Element `pre` verwendet.
- ▶ Überschriften werden mit den Elementen `h1`-`h6` dargestellt.
- ▶ Schriftart, Schriftfarbe und Schriftgröße von Textelementen lassen sich mittels des `font`-Elements und den Attributen `face`, `color` und `size` verändern.
- ▶ Die Ausrichtung eines Textabsatzes oder einer Überschrift wird mittels des `align`-Attributs und den vier Parametern `left`, `right`, `center` und `justify` verändert.
- ▶ Für Listen stehen in HTML die Varianten nummerierte Liste (`ol`), Aufzählungsliste (`ul`) und Definitionsliste zur Verfügung (`dl`).

2.12 Fragen und Übungen

1. Wie viele verschiedene Überschriftenelemente gibt es und worin unterscheiden sich diese?
2. Welche Besonderheit zeichnet das `p`-Element bezüglich der Zeilenumbrüche im HTML-Dokument aus?
3. Welche Besonderheit zeichnet das `pre`-Element bezüglich der Zeilenumbrüche im HTML-Dokument aus?
4. Erstellen Sie ein HTML-Dokument mit einer Überschrift der 1. Ordnung und dem Text Grüzi und Herzlich Willkommen!. Weisen Sie der Überschrift eine Schreibschrift zu, und ändern Sie die Schriftfarbe in grün.
5. Erweitern Sie das HTML-Dokument aus Frage/Übung 9 um einen zentrierten Absatz mit einem Text Ihrer Wahl, und verringern Sie die Schriftgröße um 2.
6. Fügen Sie dem HTML-Dokument einen vorformatierten Absatz mit einem Auszug aus einem HTML-Listing hinzu.

Teil 3

CSS – Layout fürs Internet

1 CSS – Layout fürs Internet

Wenn man bedenkt, wie viele Fehler die Computer machen, dann kann man sie als die teuersten Trottel der Welt bezeichnen.

– Renzo Favalli

CSS ist wohl die interessanteste Entwicklung zur Gestaltung einer Webseite und wird mittlerweile im gleichem Atemzug mit HTML genannt. Die Möglichkeiten sind vielfältig und übersteigen die bescheidenen Gestaltungsmöglichkeiten von HTML bei weitem. Ich werde Ihnen zu Beginn die Grundstrukturen und die Implementation in HTML erklären, sodass Sie schließlich alle wichtigen CSS-Elemente kennen gelernt haben.

1.1 CSS – Make-up fürs Web ...

CSS ist die Abkürzung von **C**ascading **S**tylesheets und stellt eine Erweiterung zu HTML dar, die vom W3C entwickelt wurde. Ziel ist es, zu den Wurzeln von HTML zurückzukehren und den Inhalt von der Gestaltung zu trennen. Schließlich ist die eigentliche Aufgabe von HTML, Daten und Inhalte zu strukturieren. Diese Aufgabe kann HTML nur schwerlich erfüllen, wenn es mit `font`-Elementen überladen wird. An dieser Stelle greift nun CSS ein. Es kümmert sich um die optische Gestaltung der Inhalte, während HTML diese nur kennzeichnet und zur Verfügung stellt.

Die Übersetzung von Stylesheet ins Deutsche macht diese Aufgabe sogar noch konkreter: Formatvorlage. Die dürften Ihnen vielleicht aus Programmen wie Microsoft Word bekannt vorkommen. Denn auch dort können Sie Textbausteine als Überschrift oder Absatz kennzeichnen und das Aussehen dieser Elemente durch eine Formatvorlage verändern. In HTML bedeutet das für Sie, nicht jedes einzelne Element durch das `font`-Element verändern zu müssen, sondern eine Vorlage anzufertigen, die für jedes HTML-Dokument verwendet werden kann. So können Sie einer Überschrift der 2. Ordnung (`h2`-Element) in jedem Dokument die gleiche Farbe zuweisen und müssen eine Änderung der Farbe nur einmal durchführen. Sie führen also eine logische Formatierung des Inhalts des HTML-Dokuments durch, ähnlich wie mit dem `em`- oder `kbd`-Element, ganz wie es dem ursprünglichen Gedanken einer Auszeichnungssprache entspricht, und überlassen die optische Formatierung dann CSS.

Formatvorlagen

1.2 Wie funktioniert CSS?

Bisher haben Sie das Aussehen eines HTML-Elements über das `font`-Element verändert.

```
<h2><font color="#0000FF">Blaue &Uuml;berschrift</font></h2>
```

Diese Angabe mussten Sie aber, um ein einheitliches Aussehen beizubehalten, für jedes `h2`-Element wiederholen und eine Änderung der Farbe brächte zwangsläufig eine erneute Änderung aller `h2`-Elemente mit sich (bzw. der `font`-Elemente). Dies nennt man direkte Formatierung. Lange HTML-Dokumente, die z. B. sehr viele `h2`-Elemente enthalten, benötigen, ohne entsprechendes Tool, viel Zeit, damit sie überarbeitet werden können. Mit CSS können Sie nun dokumentweit die Farbe für alle `h2`-Elemente verändern, und zwar mit der folgenden Angabe:

```
h2 { color:#0000FF; }
```

Zentrale Formatierung

Jedes `h2`-Element, das Sie nun in Ihrem HTML-Dokument notieren, bekommt die Farbe Blau zugewiesen, selbst dann, wenn Sie kein `font`-Element notiert haben. Sie definieren das Aussehen eines HTML-Elements an einer Stelle, nehmen also eine zentrale Formatierung vor.

Formatierungen mit CSS sind natürlich nicht auf `h2`-Elemente beschränkt, sondern können für jedes Element vorgenommen werden. Auch die Art der Formatierung ist nicht nur auf die zentrale festgelegt. Sie können Elemente auch direkt formatieren, wenn Sie das möchten. Im Gegensatz zu HTML stehen Ihnen weitaus mehr Möglichkeiten zur Verfügung. So können Sie beispielsweise jeder Überschrift einen kompletten Rahmen in einer von Ihnen gewählten Farbe, Breite und einem Stil zuweisen.



Von einer direkten Formatierung sollten Sie jedoch Abstand nehmen, da Sie dadurch den eigentlichen Sinn und Nutzen von CSS ad absurdum führen.

1.3 CSS-Versionen

Seit der ersten Veröffentlichung des CSS-Standards 1.0 im Jahre 1996 unterliegt auch CSS einer regelmäßigen Erneuerung und Überarbeitung, sodass 1998 bereits die Version 2.0 als offizieller Standard verabschiedet wurde. Aktuell ist die Version 3.0 in Arbeit, die jedoch noch einige Zeit auf sich warten lassen wird.

Das Hauptproblem ist, ähnlich wie bei HTML, die sehr unterschiedliche Unterstützung durch die Browser, da z. B. Microsoft anfängt, sein eigenes Süppchen zu kochen und viele CSS-Sprachelemente eingeführt hat, die nur mit dem Internet Explorer funktionieren, und die auch nicht als offizieller Standard gelten. Ob Microsoft damit versucht, den Internet Explorer noch stärker auf dem Markt zu verbreiten, ist sicherlich eine Frage wert, sollte uns jedoch nicht weiter interessieren. Ob mit oder ohne die IE-spezifischen Sprachelemente: Ein mit CSS formatiertes HTML-Dokument sieht in jedem Browser und abhängig vom Betriebssystem unterschiedlich aus.

Unterschiede in der Interpretation

Prinzipiell kann man feststellen, dass der Internet Explorer seit der Version 3.0 und Netscape seit der Version 4.0 den CSS-Sprachstandard kennen und ihn auch bedingt unterstützen, vor allem aber Netscape-Benutzer häufig mit Fehlinterpretationen zu kämpfen haben. Der Internet Explorer und Opera ab Version 5 und Netscape ab Version 6 haben sich gegenüber ihren Vorgängern deutlich gebessert und warten mit einer beinahe vollständigen Unterstützung des CSS-Standards 1.0 auf, Defizite bei CSS 2.0 sind aber noch häufig anzutreffen. Es ist deshalb sehr schwer, sich für eine der beiden CSS-Versionen zu entscheiden, und es macht daher Sinn, einen Mittelweg zu wählen.

Browserunterstützung

1.4 Direkte Formatierung

Damit Sie einen schnellen und einfachen Einstieg in CSS bekommen, werde ich Ihnen zuallererst die Direktformatierung eines HTML-Elements erklären. Solche Formatierungen gelten nur für das gewählte Element und haben auf die restlichen Elemente keinen Einfluss.

HTML stellt dafür das Attribut `style` zur Verfügung, das für fast jedes HTML-Element verwendet werden kann. Wie alle HTML-Attribute, wird auch dieses im Start-Tag des Elements notiert.

Neues Attribut

```
<element ... style="formatierung">...</element>
```

Als Parameter für das `style`-Attribut werden nun die unterschiedlichen Formatierungen notiert. Auch dafür gibt es eine standardisierte Schreibweise. Zuerst wird die CSS-Eigenschaft notiert, gefolgt von einem Doppelpunkt »:« und anschließend dem entsprechenden Wert.

Eigenschaft:Wert

In einem HTML-Dokument könnte dies dann folgendermaßen aussehen:

```
<h2 style="color:#00FF00">Überschrift in blau</h2>
```

Möchten Sie mehrere Eigenschaften ändern, müssen Sie die Angaben durch ein Semikolon »;« voneinander trennen.

```
Eigenschaft1:Wert1; Eigenschaft2:Wert2; Eigen-  
schaftN:WertN
```

In einem HTML-Dokument könnte dies dann folgendermaßen aussehen:

```
<h2 style="color:black; background-color:yellow">Text</  
h2>
```



Versuchen Sie, die Eigenschaft, den Doppelpunkt und den Wert immer direkt hintereinander ohne Leerzeichen zu notieren. Zwar sind Leerzeichen erlaubt, ältere Netscape-Versionen haben damit jedoch Probleme, was dazu führen kann, dass eine solche Angabe

```
Eigenschaft: Wert
```

einfach ignoriert wird. Und dies ist sicherlich nicht in Ihrem Interesse.

1.5 Das erste Stylesheet

Übertragen Sie das folgende Listing 1.1 in Ihren Editor, und speichern Sie die Datei als HTML-Dokument ab.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transi-  
tional//EN">  
<!-- Das erste Stylesheet //-->  
<html>  
  <head>  
    <title>Listing 1.1</title>  
  </head>  
  <body>  
    <h1 style="color:#0000FF">Eine blaue &Uuml;ber-  
    schrift der 1. Ordnung</h1>  
    <h1 style="font-family:sans-serif">Eine &Uuml;ber-  
    schrift der 1. Ordnung mit einer serifenlosen  
    Schrift</h1>  
    <h1 style="font-size:16px">Eine &Uuml;berschrift der  
    1. Ordnung mit einer 16 Pixel groß&szlig;en Schrift</h1>  
  </body>  
</html>
```

Listing 1.1 Das erste Stylesheet

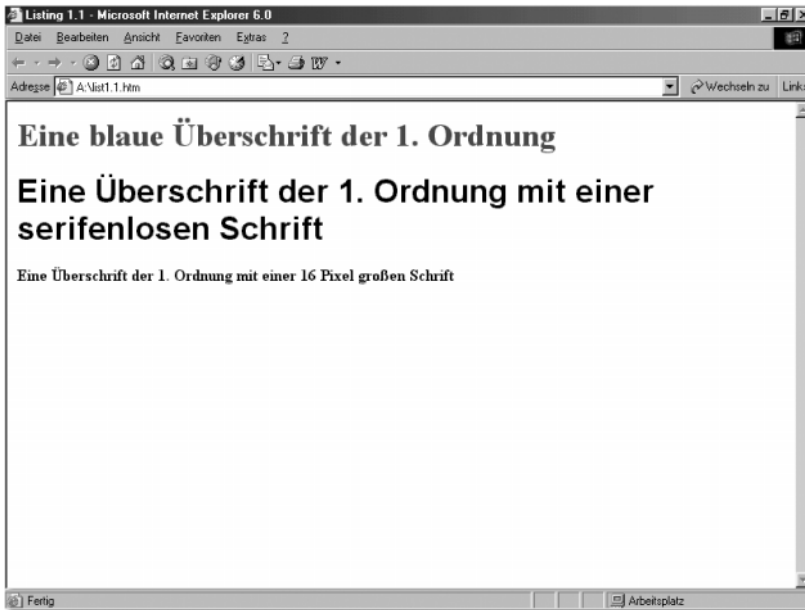


Abbildung 1.1 Darstellung des Listings 1.1 im Internet Explorer 6.0

In diesem ersten Beispiel wurden drei `h1`-Elemente mit unterschiedlichen Formaten verändert. Dem ersten `h1`-Element wurde mit der Anweisung `color:#0000FF` eine andere Farbe zugewiesen, dem zweiten wurde mit der Anweisung `font-family:sans-serif` eine andere Schriftart zugeteilt und dem letzten Element eine Schrifthöhe von 16 Pixel durch die Anweisung `font-size:16px`.

Wie Sie sehen, ist die Formatierung mit CSS sehr einfach. An diesem Beispiel können Sie jedoch schon den Nachteil der Direktformatierung erkennen. Wenn Sie nur eine Eigenschaft des Elements verändern, wirkt der Quelltext des Dokuments noch sehr sauber und aufgeräumt. Sobald es aber mehr werden, kann es schnell unübersichtlich werden.



1.6 Zentrale Formatierung

Eingangs erwähnte ich bereits, dass sich HTML-Elemente zentral formatieren lassen, und dies eine der großen Stärken von CSS ist. Dafür benötigen Sie allerdings ein neues Element: das `style`-Element. Damit ist es jedoch nicht getan. Bei der direkten Formatierung müssen Sie nicht explizit angeben, für welches Element diese Formatierung angewendet werden soll, da Sie dies ja schon festlegen, wenn Sie das `style`-Attribut im entsprechenden Element notieren. Bei der zentralen Formatierung müs-

sen Sie zuerst den Selektor angeben. Der Selektor basiert auf dem zu formatierenden Element, genauer gesagt entspricht der Selektor dem Namen des Elements. Für das `h1`-Element lautet der Selektor also `h1`, für das `p`-Element `p`.

Syntax Die bereits bekannte Syntax aus der direkten Formatierung wird nun also um den Selektor erweitert. Der Selektor wird zuerst notiert. Danach folgen die CSS-Eigenschaften und die entsprechenden Werte. Diese werden in geschweiften Klammern »{« und »}« angeführt, woraus sich dann die folgende Syntaxstruktur ergibt:

```
Selektor { Eigenschaft:Wert }
```

Ein Beispiel:

```
h2 { color:#0000FF; }
```

Mehrere Formatierungen

Es wäre allerdings ein wenig umständlich, für jede Eigenschaft eines HTML-Elements, das Sie formatieren möchten, immer wieder aufs Neue den Selektor, die Eigenschaft und den Wert zu notieren. Deshalb ist es auch hier möglich, die einzelnen Eigenschaften und Werte durch ein Semikolon zu trennen.

```
Selektor { Eigenschaft1:Wert1;
           Eigenschaft2:Wert2;
           EigenschaftN:WertN; }
```

Beispiel:

```
h2 { color:#0000FF;
     font-family:sans-serif;
     font-size:16px; }
```

Damit haben Sie nun gesehen, wie Sie die CSS-Eigenschaften und Werte notieren und den jeweiligen Elementen zuordnen können. Fehlt nur noch die Implementierung in HTML. Das `style`-Element wird im Gültigkeitsbereich des `head`-Elements notiert und enthält in seinem Gültigkeitsbereich wiederum die Formatierungen. Außerdem sollten Sie den MIME-Typ angeben, der für CSS `text/css` lautet und als Parameter für das Attribut `type` notiert wird.

```
<head>
...
<style type="text/css">
  h1 { color:#0000FF;
      font-family:sans-serif;
```

```

        font-size:16px; }
    </style>
    ...
</head>

```

In diesem Beispiel würden Sie jedem im HTML-Dokument verwendeten h1-Element die Farbe Blau (#0000FF), den Schrifttyp `sans-serif` und eine Schriftgröße von 16 Pixel zuweisen. Browser, die kein CSS unterstützen, könnten mit einer solchen Angabe jedoch Probleme haben, die CSS-Formatierungen fälschlicherweise als Text interpretieren und dann entsprechend im Browser ausgeben. Um dies zu verhindern, werden die Formatierungen als Kommentar notiert. Browser, die kein CSS unterstützen, ignorieren die Angaben vollständig. Browser, die CSS unterstützen, interpretieren diese Angaben korrekt und ignorieren dann einfach den Kommentar.

Fehlinterpretationen

```

<head>
    ...
    <style type="text/css">
        <!--
            h1 { color:#0000FF;
                font-family:sans-serif;
                font-size:16px; }
        //-->
    </style>
    ...
</head>

```

Natürlich können Sie CSS-Formatierungen für mehrere Elemente innerhalb des `style`-Elements notieren. Die einzelnen Konstrukte werden dann hintereinander bzw. untereinander notiert.

Mehrere Elemente formatieren

```

<style type="text/css">
    <!--
        Selektor1 { Eigenschaft1:Wert1;
                    Eigenschaft2:Wert2;
                    EigenschaftN:WertN; }
        Selektor2 { Eigenschaft1:Wert1;
                    Eigenschaft2:Wert2;
                    EigenschaftN:WertN; }
        SelektorN { Eigenschaft1:Wert1;
                    Eigenschaft2:Wert2;
                    EigenschaftN:WertN; }
    //-->
</style>

```




Sehen Sie sich dazu einfach ein vollständiges Beispiel an. Wenn Sie sich die Arbeit ersparen möchten, das Listing abzutippen, finden Sie es auf der beiliegenden CD-ROM im Verzeichnis `x:\listings\css\list1.2.htm`. Ich möchte Sie aber darauf hinweisen, dass selbst das sture Abtippen eines Listings bereits übt. Und Übung macht ja bekanntlich den Meister.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<!-- Zentrale Element-Formatierung //-->
<html>
  <head>
    <title>Listing 1.2</title>
    <style type="text/css">
      <!--
        h1 { color:#336699;
            font-family:sans-serif; }
        p  { font-size:14px;
            font-family:sans-serif; }
      //-->
    </style>
  </head>
  <body>
    <h1>Zentrale Formatierung</h1>
    <p>Die Elemente, die in diesem HTML-Dokument verwendet werden, wurden durch CSS zentral formatiert.</p>
    <p>Dies kann besonders viel Arbeit sparen, wenn man dabei geschickt vorgeht.</p>
  </body>
</html>
```

Listing 1.2 Beispiel für zentrale Element-Formatierung

In Listing 1.2 werden zentral zwei verschiedene Elemente formatiert: zum einen das `p`-Element und zum anderen das `h2`-Element. Beide Elemente bekommen ihre eigenen Eigenschaften zugewiesen. So werden alle Elemente des Typs `h2` in der Farbe `#336699` und in einer serifenlosen Schrift dargestellt. Auch die `p`-Elemente erhalten eine serifenlose Schrift. Nur die Farbe ist eine andere, und zwar `#000000`.



Abbildung 1.2 Darstellung des Listings 1.2 im Internet Explorer 6.0

1.7 Externe Formatierung

Je mehr HTML-Elemente Sie formatieren, umso länger werden die CSS-Angaben. Wenn Sie außerdem die CSS-Formatierungen in mehreren HTML-Dokumenten verwenden möchten, ist es sinnvoll, diese in eine externe Datei auszulagern. Das Auslagern der Formatierungen kann man auch als eine externe Formatierung bezeichnen. Als Dateiendung hat sich für Cascading-StyleSheets **.css** durchgesetzt.

Der Aufbau einer solchen Datei ist einfach. Exakt so, wie Sie die Formatierungen im `style`-Element notieren würden, werden sie auch in einer externen Datei als reiner Text gespeichert (natürlich ohne `style`- und Kommentar-Tags).

Dateiaufbau

Um nun diese Datei in einem HTML-Dokument als Vorlage verwenden zu können, müssen Sie das `link`-Element einsetzen (dt. Verweis). Dieses Element wird ebenfalls im Gültigkeitsbereich des `head`-Elements notiert.

Datei einbinden

```
<head>
...
<link rel="stylesheet" type="text/css"
      href="beispiel.css">
...
</head>
```

Die Angaben `rel="stylesheet"` und `type="text/css"` weisen die einzubindende Datei als Stylesheet des Typs `css` aus (`rel` = engl. *relation*, dt. *Bezug*). Im Attribut `href` (engl. *hyper reference*, dt. *Hyperreferenz*) geben Sie dann die entsprechende CSS-Datei an. Dabei gelten die gleichen Regeln wie bei der Angabe eines Verweiszies im `a`-Element bezüglich der Verzeichnisse. Sie müssen also absolute und relative Pfadangaben unterscheiden und natürlich auch berücksichtigen, wo die CSS-Datei, von der einbindenden HTML-Datei aus gesehen, liegt.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<!-- Externe Element-Formatierung //-->
<html>
  <head>
    <title>Listing 1.3</title>
    <link rel="stylesheet" type="text/css"
href="list1.3.css">
  </head>
  <body>
    <h1>Externe Formatierung</h1>
    <p>Die Elemente, die in diesem HTML-Dokument verwen-
det werden, wurden durch CSS zentral formatiert.</p>
    <p>Diese Formatierungen wurden aber in einer exter-
nen Datei gespeichert, um sie auch in anderen HTML-
Dokumenten verwenden zu können.</p>
  </body>
</html>
```

Listing 1.3 Beispiel für eine externe Formatierung

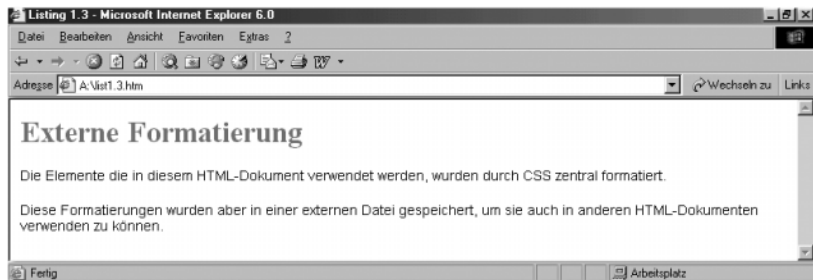


Abbildung 1.3 Darstellung des Listings 1.3 im Internet Explorer 6.0

Die Datei **list1.3.css**, die in Listing 1.3 als CSS-Stylesheet referenziert wurde, sieht wie folgt aus:

```
h1 { color:#996633;
      font-size:30px; }
p  { font-size:14px;
      font-family:sans-serif; }
```

Auch in diesem Beispiel werden wieder nur die Elemente `p` und `h1` formatiert. Das Element `h1` erhält die Farbe `#996633` und die Schriftgröße 30 Pixel. Das `p`-Element erhält hingegen die Schriftgröße 14 Pixel und eine serifenlose Schrift.

1.8 Ausgabemedien

Ein weiterer interessanter Aspekt in CSS ist die Möglichkeit, verschiedene Stylesheets für unterschiedliche Ausgabemedien zu definieren. So ist beispielsweise ein Bildschirm ein ganz anderes Ausgabemedium als ein Drucker, insbesondere in Anbetracht der Darstellungsform. Die Darstellung auf einem Monitor ist im Querformat, während die Ausgabe auf einem Drucker in der Regel im Hochformat erfolgt.

Leider ist Netscape nicht einmal in der Version 6 in der Lage, Stylesheets für unterschiedliche Ausgabemedien zu verwenden. Zwar erkennt er das richtige Stylesheet für die Bildschirmausgabe, ignoriert aber konsequent alle anderen. Anders der Internet Explorer: Dieser ist bereits seit der Version 4 in der Lage, das passende Stylesheet für das entsprechende Medium zu verwenden.

**Netscape macht
Zicken**

Um verschiedene Stylesheets für unterschiedliche Ausgabemedien verwenden zu können, benötigen Sie erstens eine CSS-Datei und zweitens das Attribut `media` (dt. *Medium*), das Sie im `link`-Start-Tag notieren.

Folgende Angaben für das `media`-Attribut sind erlaubt:

Parameter	Beschreibung
screen	Bildschirmausgabe
all	Alle Ausgabemedien
aural	Sprachausgabe
braille	Brailledisplays (Monitor für Blindenschrift)

Tabelle 11 Schlüsselwörter für die verschiedenen Ausgabemedien

Parameter	Beschreibung
embossed	Brailledrucker (Drucker für Blindenschrift)
handheld	Ausgabe auf tragbaren Kleincomputern wie Handhelds, PDAs etc.
print	Druckerausgabe
projection	Beamerausgabe und Ähnliche
tty	Fernschreiber und Ähnliche (tty = engl. <i>teletyper</i>)
tv	Ausgabe auf Fernsehgeräten und Ähnlichen

Tabelle 1.1 Schlüsselwörter für die verschiedenen Ausgabemedien (Forts.)

Eine Angabe für die Ausgabemedien Bildschirm oder Drucker könnte dann wie folgt lauten:

```
<link rel="stylesheet" type="text/css" media="screen"
href="screen.css">
<link rel="stylesheet" type="text/css" media="print,
embossed" href="print.css">
```



Die vorherigen Angaben sind HTML-Syntax, für CSS gibt es entsprechend eine eigene Notation.

```
@import url(screen.css) screen;
@import url(print.css) print, embossed;
```

1.9 Einheiten

Anders als HTML kennt CSS die unterschiedlichsten Einheiten, um eine Wertangabe zu machen. Während Sie sich bei HTML auf Prozent- und Pixelangaben beschränken müssen bzw. auf die Angabe einer relativen Schriftgröße, kennt CSS Angaben in Zentimetern, Zoll, Prozent, Pixel etc.

Wertangaben in CSS erfolgen durch die Angabe des Werts, gefolgt von der gewünschten Maßeinheit.

Einheit	Typ	Beschreibung
cm	absolut	Zentimeterangabe (1/100 m)
em	relativ	Relative Angabe in Bezug auf den höchsten Buchstaben im Elternelement
ex	relativ	Relative Angabe zum niedrigsten Buchstaben im Elternelement

Tabelle 1.2 Maßeinheiten in CSS

Einheit	Typ	Beschreibung
in	absolut	Zollangabe (engl. <i>inch</i>) = 2,54 cm
mm	absolut	Millimeterangabe (1/1000 m)
pc	absolut	Pica (1 Pica = 12 Punkt)
pt	absolut	Punkt (1 Punkt = 1/72 in)
px	absolut	Pixelangabe
%	relativ	Prozentangabe

Tabelle 1.2 Maßeinheiten in CSS (Forts.)

Ein paar Beispiele:

1cm
1pt
6pc
16px
10%

Als Dezimaltrennzeichen wird nicht das Komma, sondern der Punkt verwendet, da CSS auf der englischen Sprache und den entsprechenden Regeln basiert.



1.2em
0.8ex
1.5in
2.54cm
6.5pt

Wie Sie sehen, können Sie in CSS die verschiedensten Angaben zu Maßen machen. Es sollte aber erwähnt werden, dass Angaben wie `em` und `ex` sehr selten verwendet werden, und Sie bei Maßen auch immer darauf achten sollten, dass der Benutzer in der Lage sein sollte, den Text auch lesen zu können.



1.10 Zusammenfassung

- ▶ Für die Direktformatierung eines HTML-Elements wird das Attribut `style` verwendet, das im Start-Tag des Elements notiert wird.
- ▶ Für die zentrale bzw. externe Formatierung von HTML-Elementen wird das Element `style` verwendet, das im Gültigkeitsbereich des `head`-Elements notiert wird.

- ▶ CSS-Dateien sollten über die Endung `.css` verfügen.
- ▶ Für unterschiedliche Ausgabemedien können verschiedene Stylesheets definiert werden. Jedoch unterstützt nur der IE diese in vollem Umfang.
- ▶ Maße können in CSS in `cm`, `mm`, `px`, `pt`, `in`, `pc`, `em`, `ex` und Prozent angegeben werden.

1.11 Fragen und Übungen

1. Wie lautet die Syntax für die Direktformatierung eines HTML-Elements?
2. Wie lautet die Syntax für die zentrale Formatierung von HTML-Elementen?
3. Mit welchem Symbol werden mehrere Angaben von Eigenschaften und Werten getrennt?
4. Was sollten Sie bei der zentralen Formatierung notieren, damit Browser, die kein CSS unterstützen, die Angaben nicht fälschlicherweise als Text interpretieren und diesen ausgeben?

Teil 6
PHP – Dynamisch und
Interaktiv II

1 PHP

Aber für was ist das gut?

– Ingenieur von Advanced Computing Systems Division of IBM, 1968, zum Microchip

PHP wurde ursprünglich im Herbst 1994 von Rasmus Lerdorf entwickelt und bestand anfänglich aus einer Sammlung von Perl-Skripten, welche es Lerdorf ermöglichten, Zugriffe auf Webseiten zu erfassen und auszuwerten. Die Abkürzung PHP steht für »**P**ersonal **H**ome **P**age **T**ools«.

Als Rasmus Lerdorf jedoch mehr Funktionalität benötigte, entwickelte er Mitte 1995 den ersten Parser und nannte das Paket PHP/FI Version 2. FI ist die Abkürzung für »**F**orm **I**nterpreter«. Bereits diese Version verfügte über große Ähnlichkeiten mit dem PHP, wie es heute verwendet wird. Es existierten bereits Variablen, Formularvariablen wurden automatisch interpretiert (FI = engl. form interpreter), und er beherrschte eine in HTML eingebettete Syntax. Zwar wies die Syntax große Affinitäten mit Perl auf, sie war jedoch sehr viel einfacher strukturiert. **PHP/FI**

Bis 1997 erfuhr PHP/FI immer wieder Verbesserungen und Änderungen und wurde mittlerweile auf ca. 50.000 Servern eingesetzt (ca. 1% Marktanteil). Zu diesem Zeitpunkt war es allerdings noch immer ein Ein-Mann-Projekt.

Dies änderte sich 1997 als Zeev Suraski und Andi Gutmans den Parser vollständig neu programmierten, da ihnen PHP/FI zu schwach für die Entwicklung ihrer eigenen E-Commerce-Applikationen war. Sie einigten sich mit Lerdorf, und der neue Parser wurde als PHP 3.0 – nach 9 Monaten Testphase – Mitte 1998 freigegeben. Ende 1998 fand sich PHP 3.0 auf ca. 10% der weltweiten Webserver. **PHP 3.0**

Im Mai 2000 wurde dann PHP 4.0 freigegeben, das auf der neuen Scripting Engine Zend (gebildet aus den Vornamen Zeev und Andi) basierte. Viele Erweiterungen fanden Einzug in die neue Version, wie etwa eine stark verbesserte Leistung, HTTP-Sessions und neue Sprachkonstrukte. Diese Verbesserungen wurden entsprechend honoriert, was dazu führte, dass auf ca. 20% der Domains PHP installiert wurde und ca. 100.000 Entwickler mit PHP arbeiteten. **Zend**

Zum Zeitpunkt der Drucklegung dieses Buches liegt PHP in der finalen Version 4.3.0 vor. Die Arbeiten an PHP 5 haben bereits begonnen, wer- **PHP 5.0**

den aber aller Voraussicht nach Ende des Jahres 2003 nicht abgeschlossen sein.

1.1 Das erste PHP-Script

PHP funktioniert ähnlich wie Perl, jedoch mit einem gravierenden Unterschied: PHP kann, ähnlich wie JavaScript, in HTML-Dokumente eingebettet werden. Die Einbettung ist jedoch nicht zwingend erforderlich. PHP kann genauso gut verwendet werden, um HTML-Code zu erzeugen.

PHP-Tags Beiden Varianten, Einbettung in den Code und das Erzeugen des Codes, gemein sind die Start- und Ende-Tags für die Scripts. PHP-Quellcode wird immer zwischen den Tags

```
<?php
```

und

```
?>
```

notiert, egal, ob eine Verwendung innerhalb eines HTML-Dokuments oder als eigenständiges Script erfolgt.

```
<html>
  <head>
    <title>Listing 1.1</title>
  </head>
  <body>
    <?php
      echo "Hallo WorldWideWeb!";
    ?>
  </body>
</html>
```

Listing 1.1 Erstes PHP-Script

Das PHP-Script aus Listing 1.1 ist ein Beispiel für PHP-Code, der in ein HTML-Dokument eingebettet wurde. Kopieren Sie dieses Script in eine neue Datei, und speichern Sie es im Dokumenten-Verzeichnis Ihres Web-servers ab.



Beachten Sie, dass Sie als Dateiendung **.php** wählen müssen, auch wenn das Script äußerlich einem HTML-Dokument gleicht. In einigen Fällen kann es auch sein, dass die Endung der Datei **.php3** oder **.php4** lauten muss. Dies ist abhängig von der Webserver-Konfiguration. Häufig kommt auch die Endung **.phtml** oder **.phtm** zum Einsatz.

Den Grund dafür werde ich Ihnen im nachfolgenden Kapitel noch genauer erläutern. Zurück zum Listing 1.1. Alles, was außerhalb des Tag-Paares `<?php` und `?>` notiert wurde, wird als normaler HTML-Code interpretiert und ganz normal im Browser ausgegeben. Alles, was innerhalb des PHP-Tag-Paares notiert wurde, wird als PHP-Code interpretiert und entsprechend vom PHP-Interpreter verarbeitet. Die Anweisung

```
echo "Hallo WorldWideWeb!";
```

weist den Interpreter an, die Ausgabe `Hallo WorldWideWeb!` zu erzeugen. Der Befehl `echo` erzeugt also eine Ausgabe. Nach `echo` kann entweder eine feste Zeichenkette (wird in doppelten oder einfachen Anführungsstrichen notiert) oder eine Variable folgen. Anschließend wird die vollständige Anweisung mit einem Strichpunkt bzw. Semikolon abgeschlossen (`;`).

Rufen Sie das Listing 1.1 nun in Ihrem Browser auf. Die Ausgabe des Scripts sollte folgendermaßen aussehen:

```
Hallo WorldWideWeb!
```

Herzlichen Glückwunsch, dies war Ihr erstes PHP-Script.

1.2 PHP und die Dateierendungen

Was hat es aber mit der Endung `.php` auf sich? Genauso wie die Endungen `.htm` oder `.html` eine Datei als HTML-Dokument ausweisen, definiert die Endung `.php` eine Datei als PHP-Script. Wenn nun PHP-Code in einem HTML-Dokument eingebettet wurde, muss dem Webserver klar gemacht werden, dass innerhalb dieses Dokuments spezielle Anweisungen enthalten sind, die vor der Übertragung der Datei an den Browser interpretiert werden müssen. Andernfalls würde der PHP-Code einfach mit dem Rest an den Browser gesendet werden. Damit kann der Benutzer freilich nicht viel anfangen.

Damit der Server nun aber den richtigen Interpreter wählt, müssen Sie die Endung `.php` verwenden. In der Regel ist diese Dateierendung mit dem PHP-Interpreter verknüpft, der sich dann um die Ausführung des PHP-Codes und die anschließende Ausgabe im Browser kümmert. Der Interpreter kann nun anhand der PHP-Tags erkennen, ob es sich um HTML-Code oder um PHP-Code handelt.

**Verknüpfung mit
PHP-Interpreter**

Im Fall des Listings 1.1 findet er zu Beginn lediglich HTML-Code und sendet diesen ohne Umwege an den Browser. Erst wenn er auf das Tag `<?php` stößt, beginnt seine eigentliche Aufgabe. Den nachfolgenden Code inter-

pretiert er nun als PHP-Code und führt die Anweisungen entsprechend aus; solange, bis er auf das Tag `?>` stößt. Alles was danach folgt, wird wieder als HTML-Code bewertet und direkt zum Browser gesendet.

Dies bedeutet jedoch nicht, dass das PHP-Tag-Paar nur einmal in einer Datei verwendet werden darf. Ein Beispiel:

```
<html>
  <head>
    <title>Listing 1.2</title>
  </head>
  <body>
    <?php
      echo "Hallo WorldWideWeb!";
    ?>
    <?php
      echo "</body></html>";
    ?>
```

Listing 1.2 Mehrere PHP-Tag-Paare in einem Script

Der Unterschied zum Listing 1.1 liegt im zweiten PHP-Tag-Paar. Anstatt einfach den restlichen HTML-Code im Browser auszugeben, erfolgt dies mittels der `echo`-Anweisung. Das Ergebnis beider Listings (sowohl 1.1 als auch 1.2) ist das gleiche, wobei das Listing 1.2 eher unsinnig ist.

1.3 PHP in eigenen Dateien

Wie bereits erwähnt, kann ein PHP-Script auch lediglich aus PHP-Code bestehen. Der HTML-Code muss dann aber mittels der `echo`-Anweisung ausgegeben werden.

```
<?php
  echo "<html>";
  echo "<head>";
  echo "<title>Listing 1.3</title>";
  echo "</head>";
  echo "<body>";
  echo "Hallo WorldWideWeb!";
  echo "</body>";
  echo "</html>";
?>
```

Listing 1.3 PHP-Script das ausschließlich PHP-Code enthält

Anders als Perl sendet PHP automatisch den korrekten MIME-Typen an den Browser, also `text/html`. Sie müssen diesen Header nur dann ausgeben, wenn die nachfolgenden Daten nicht zu einem HTML-Dokument, sondern zu einer Grafik oder Ähnlichem gehören.



In Listing 1.3 wird das gesamte HTML-Dokument mittels PHP ausgegeben. Auch bei diesem Beispiel fehlt der wirkliche Sinn, es verdeutlicht aber, dass auch in einem solchen PHP-Script das PHP-Tag-Paar notiert werden muss, da der Interpreter sonst den reinen Quelltext an den Browser sendet.

Würden Sie die Tags `<?php` und `?>` aus dem Listing 1.3 entfernen, sähe die Ausgabe im Browser folgendermaßen aus:

```
echo ""; echo ""; echo ""; echo ""; echo ""; echo "Hallo  
WorldWideWeb!"; echo ""; echo "";
```

1.4 Quellstrukturierung und Kommentare

Auch in PHP ist eine vernünftige Quellstrukturierung und -kommentierung äußerst wichtig. Der Grund dafür dürfte mittlerweile hinlänglich bekannt sein.

Kommentieren können Sie in PHP auf drei Arten, entweder einzeilig oder mehrzeilig. Für einzeilige Kommentare notieren Sie einfach zwei Schrägstriche `//`. Alles was nach diesen beiden Schrägstrichen bis zum Zeilende folgt, wird dann als Kommentar interpretiert und vom PHP-Interpreter ignoriert. Anstelle der Schrägstriche können Sie aber auch das Raute-Zeichen `#` verwenden.

Zwei einzeilige
Kommentararten

Für mehrzeilige Kommentare müssen Sie `/*` und `*/` verwenden. Dabei kennzeichnet `/*` den Beginn eines mehrzeiligen Kommentars und `*/` das Ende des Kommentars.

Eine mehrzeilige
Kommentarart

```
echo "Hallo WordWideWeb!"; // ein einzeiliger Kommentar  
echo "Hallo Welt!"; /* Mehrzeiliger  
                        Kommentar */  
echo "Hallo Alle!";    # ebenfalls einzeiliger Kommentar
```

1.5 Textausgabe

In vielen Fällen ist die Ausgabe von Text mittels der `echo`-Anweisung ungenügend, gerade in Bezug auf Zeilenumbrüche. Zwar können Sie einen optischen Zeilenumbruch im Browser mit dem Element `br` bewir-

ken, wenn Sie aber Text ausgeben möchten, der mit dem `pre`-Element formatiert werden soll, hat das `br`-Element keinerlei Auswirkungen.

Zeilenumbrüche Dafür benötigen Sie das Steuerzeichen `\n`. Dies erzeugt im Quelltext des HTML-Dokuments einen Zeilenumbruch (*n = engl. new line = dt. neue Zeile*). An welcher Stelle Sie das Steuerzeichen notieren ist egal.

```
echo "Hallo Welt!\n";
```

Die Anweisung erzeugt einen im Browser nicht sichtbaren Zeilenumbruch nach der Zeichenkette `Hallo Welt!`.

```
echo "<pre>Hallo \n Welt!</pre>";
```

Diese Anweisung hingegen erzeugt sowohl im Quelltext als auch im Browser einen sichtbaren Zeilenumbruch.

print-Anweisung Äquivalent zur `echo`-Anweisung können Sie auch die `print`-Anweisung verwenden. Sowohl die Syntax als auch die Wirkungsweise beider Anweisungen sind gleich. Die Anweisung `print` kommt noch aus den Anfangszeiten von PHP, woran auch die Abstammung von Perl deutlich wird.

```
echo "Hallo Welt!<br>";  
print "Hallo Welt!<br>";
```

Die Ausgabe:

```
Hallo Welt!  
Hallo Welt!
```

1.5.1 Formatierte Textausgabe

Anders sieht es bei der Funktion `printf` aus. Sie ist stark an die C-Syntax¹ angelehnt. Mit dieser Funktion ist es möglich, Daten für die Ausgabe speziell zu formatieren. Vor allem bei der Ausgabe von Zahlenwerten ist dies hilfreich.

Dabei wird die Ausgabe mit Hilfe von Platzhaltern gesteuert.

```
int printf(string format [, mixed args...])
```

Die Formatierungszeichenkette mit den Platzhaltern wird als erster Parameter an die Funktion übergeben und anschließend folgen die einzelnen Daten. Die Formatierung kann dabei sowohl normalen Text als auch die Platzhalter enthalten.

¹ PHP hat seine Wurzeln auch in der Programmiersprache C.

Platzhalter	Erklärung
%b	Binäre Darstellung von Ganzzahlen
%c	Darstellung einer Ganzzahl als ASCII-Zeichen
%d	Darstellung einer Ganzzahl als Dezimalzahl
%e	Fließkommazahl in Exponentialdarstellung
%f	Fließkommadarstellung
%g	Entweder %e oder %f, je nachdem, was kürzer ist
%o	Darstellung einer Ganzzahl als Oktalzahl
%s	Darstellung als Zeichenkette
%x, %X	Darstellung einer Ganzzahl in Hexadezimaldarstellung, entweder in Kleinbuchstaben (%x) oder Großbuchstaben (%X)

Tabelle 1.1 Platzhalter zur Formatierung mittels der Funktion printf

Hier ein Beispiel für die unterschiedlichen Ausgabeformatierungen.

```
<?php
    printf("Binär: %b<br>",167);
    printf("ASCII: %c<br>",167);
    printf("Dezimal: %d<br>",167);
    printf("Exponential: %e<br>",167);
    printf("Fließkomma: %.2f<br>",167);
    printf("Oktal: %o<br>",167);
    printf("Zeichenkette: %s<br>",167);
    printf("Hexadezimal: %x<br>",167);
    printf("Hexadezimal: %X<br>",167);
?>
```

Listing 1.4 Formatierte Ausgabe von Daten

Im Browser erzeugt das Listing 1.4 die folgende Ausgabe:

```
Binär: 10100111
ASCII: §
Dezimal: 167
Exponential: 1.67000
Fließkomma: 167.00
Oktal: 247
Zeichenkette: 167
```

Hexadezimal: a7

Hexadezimal: A7

1.6 Alternative PHP-Tags

Neben der Verwendung der Tags `<?php` und `?>` können Sie noch weitere Schreibweisen verwenden.

Bei der Kurzform werden die Tags `<?` und `?>` verwendet.

```
<? echo "Hallo Welt!"; ?>
```

Die Langform verwendet das HTML-Element `script` zur Einbindung von Scripts.

```
<script language="php">
  echo "Hallo Welt!";
</script>
```

Bei der ASP-Form werden die Tags `<%` und `%>` verwendet, die normalerweise bei **Active Server Pages** Anwendung finden.

```
<% echo "Hallo Welt!"; %>
```



Sowohl die Kurzform als auch die ASP-Form müssen nicht mit jeder PHP-Installation funktionieren, da dies nur Optionen sind und in der Datei **php.ini** unter Umständen aktiviert werden müssen. Bei den meisten Providern haben Sie jedoch keinen Zugriff auf diese Datei. Aus diesem Grund sollten Sie die bisher verwendete Form `<?php ... ?>` benutzen.

1.7 Zusammenfassung

- ▶ PHP-Code kann sowohl in HTML eingebettet werden als auch in eigenen Dateien notiert werden.
- ▶ Generell müssen Sie PHP-Tags verwenden, innerhalb derer der PHP-Code notiert wird. Die gängigste Variante ist `<?php ... ?>`.
- ▶ Ausgaben können mit den Anweisungen `echo` oder `print` oder mit der Funktion `printf` erfolgen.
- ▶ Einzeilige Kommentare werden durch `//` eingeleitet und mehrzeilige Kommentare werden von `/*` und `*/` umgeben.
- ▶ Die Endung eines PHP-Scripts sollte `.php` lauten.

1.8 Fragen und Übungen

1. Worin liegt der Unterschied zwischen der Anweisung `echo` und der Funktion `printf` für die Ausgabe im Browser?
2. Mit welchem Steuerzeichen können Sie im HTML-Quelltext einen Zeilenumbruch erzeugen?
3. Schreiben Sie ein PHP-Script, welches ein vollständiges HTML-Dokument erzeugt und zusätzlich eine Überschrift der 1. Ordnung mit dem Text »Dynamische Webseiten mit PHP« ausgibt. Strukturieren Sie den HTML-Quelltext mit Zeilenumbrüchen.

2 Variablen und Operatoren

Die Leute, die glauben, dass sie Computer hassen, hassen in Wirklichkeit nur die schlechten Programmierer.

– Larry Niven

2.1 Variable

Wie prinzipiell jede Programmiersprache, die ernst genommen werden will, verfügt auch PHP über die Möglichkeit, Variablen verwenden zu können. Variablen kennzeichnen einen Speicherbereich, in dem während der Ausführung des Programms Werte gespeichert werden können.

In den meisten Sprachen, muss vor der Verwendung einer Variablen jedoch festgelegt werden, von welchem Typ diese Variable sein soll. So z. B. in Perl: Dort wird der Typ der Variablen durch ein vorangestelltes Zeichen identifiziert (\$ für Skalare, @ für Listen und % für Hashes). Diese Unterscheidung existiert in PHP jedoch nicht. Hier beginnen alle Variablen mit dem Dollarzeichen \$ gefolgt von einer beliebigen Zeichenkette. Erst durch die Verwendung und die Zuweisung eines Wertes definiert sich der Typ der Variablen.

Variablentyp

2.1.1 Variablen bezeichnen

Um einer Variablen einen Namen zu geben, können Sie sowohl die Buchstaben von a-z und von A-Z verwenden und die Zahlen von 0–9 sowie den Unterstrich `_`. Zusätzlich dürfen Sie auch Zeichen verwenden, welche dem ASCII-Code 127 bis 255 entsprechen.

Außerdem sind Variablennamen in PHP case-sensitive, d. h., es wird zwischen Groß- und Kleinschreibung unterschieden.

```
$VAR = "Er";  
$var = "Sie";  
$vaR = "Es";  
echo "$VAR, $var, $vaR";
```

Dieses Beispiel würde im Browser die Ausgabe `Er, Sie, Es` erzeugen, da bei allen drei Zuweisungen unterschiedliche Variablen angesprochen werden – trotz gleich klingender Namen.

Nach dem Dollarzeichen \$ darf übrigens nur ein Buchstabe oder der Unterstrich folgen, um einen gültigen Namen für eine Variable zu vergeben.

```
$abc123 = "abc und 123"; // gültig, da Buchstabe am
                                // Anfang
$_abc = "ABC";                // gültig, da Unterstrich am
                                // Anfang
$1_abc = "Ein ABC";          // ungültig, da Zahl am Anfang
$töterö = "Elefant";        // gültig, da ö dem ASCII-Code
246                           // entspricht
```

2.2 Datentypen

Variablen in PHP definieren sich also durch den ihnen zugewiesenen Wert. Welche Datentypen es im Einzelnen gibt, soll nun in diesem Kapitel genauer betrachtet werden.

PHP unterscheidet generell zwischen acht primitiven Datentypen.

- ▶ Integer
- ▶ Fließkommazahl
- ▶ Boolean
- ▶ String
- ▶ Array
- ▶ Object
- ▶ Resource
- ▶ NULL

2.2.1 Integer

Unter den primitiven Datentyp Integer fallen Dezimalzahlen, Oktalzahlen und auch hexadezimale Zahlen. Durch unterschiedliche Schreibweisen bei der Wertzuweisung entscheidet sich dann, welcher Datentyp zum Einsatz kommt.

```
$x = 1024;
$x = -512;
```

Bei einer solchen Zuweisung wird der Typ der Variablen auf Dezimalzahl gesetzt. Dabei darf der Wert positiv oder negativ sein. Die maximale Größe einer solchen Zahl ist plattformabhängig. Üblich ist jedoch eine Größe von 32 Bit (2^{32}).

```
$x = 0123; // Dezimal = 83
```

Diese Schreibweise weist der Variablen `$x` den Typ Oktalzahl zu. Auch bei Oktalzahlen ist ein negativer und positiver Wertebereich erlaubt. Oktalzahlen werden immer durch eine führende 0 definiert.

```
$x = 0xFF; // Dezimal = 255  
$x = 0x99; // Dezimal = 153
```

Dies ist die Schreibweise für eine hexadezimale Zahl. Hexadezimale Zahlen werden immer durch den Vorsatz `0x` definiert.

2.2.2 Fließkommazahl

Fließkommazahlen werden in der Regel auch als floatingpoint oder kurz float bezeichnet. Mögliche Schreibweisen sind:

```
$x = 1.024;  
$x = 1.2e6;  
$x = 5E-10;
```

Auch die Größe einer Fließkommazahl ist plattformabhängig. In der Regel ist sie jedoch auf 64 Bit mit einer Genauigkeit von 14 Nachkommastellen festgelegt.



2.2.3 Boolean

Variablen des Typs Boolean sind Wahrheitswerte und können entweder den Wert `TRUE` oder `FALSE` erhalten.

```
$x = TRUE;  
$x = FALSE;  
$x = True;  
$x = fALSE;
```

Ob Sie `TRUE` bzw. `FALSE` nun groß oder klein schreiben, bleibt Ihnen überlassen. Wegen der besseren Lesbarkeit des Quelltextes wird in der Regel aber die Großschreibung verwendet.

Wenn Sie den Wert eines Booleans ausgeben möchten, müssen Sie beachten, dass `TRUE` als 1 und `FALSE` als leere Zeichenkette ausgegeben wird.



2.2.4 String

Strings sind Zeichenketten und, wie in vielen Scriptsprachen üblich, können Sie sich aussuchen, ob Sie doppelte oder einfache Anführungszeichen verwenden.

```
$x = "Hallo"; // doppelte Anführungszeichen
$x = 'Hallo'; // einfache Anführungszeichen
```

Es gibt jedoch einen Unterschied. Wenn Sie doppelte Anführungszeichen verwenden, werden Variablen, die innerhalb einer Zeichenkette notiert wurden, automatisch aufgelöst.

```
$x = 5;
echo "Es ist nun $x Tage her, dass...";
```

Dieses Beispiel würde die Ausgabe

```
Es ist nun 5 Tage her, dass...
```

erzeugen. Die Variable `$x`, die innerhalb der Zeichenkette notiert wurde, ist also durch ihren Wert ersetzt worden. Wohingegen das Beispiel

```
$x = 5;
echo 'Es ist nun $x Tage her, dass...';
```

die Ausgabe

```
Es ist nun $x Tage her, dass...
```

erzeugen würde. Möchten Sie nun aber auch Anführungszeichen im Browser ausgeben, kann dies zu Problemen führen. Würden Sie das folgende Beispiel ausprobieren

```
echo 'I'll be back';
```

quittiert Ihnen PHP dies mit einer Fehlermeldung. Nun haben Sie zwei Möglichkeiten, das Problem zu lösen. Entweder verwenden Sie doppelte Anführungszeichen oder das Escape-Zeichen `\`.

```
echo "I'll be back";
echo 'I\'ll be back';
```

Beides sind akkurate Lösungen. Um wiederum das Escape-Zeichen auflösen zu können, müssen Sie zwei Backslashes hintereinander notieren.

```
echo "Alle Daten auf Laufwerk C:\\ löschen?";
// Ausgabe: Alle Daten auf Laufwerk C:\ löschen?
```

Bei der Verwendung von doppelten Anführungszeichen kennt PHP jedoch noch mehr Escape-Sequenzen.

Sequenz	Erklärung
<code>\n</code>	Zeilenumbruch
<code>\r</code>	Wagenrücklauf
<code>\t</code>	Tabulator
<code>\"</code>	Doppeltes Anführungszeichen
<code>\\$</code>	Dollarzeichen
<code>\\</code>	Backslash

Tabelle 2.1 Escape-Sequenzen bei Verwendung von doppelten Anführungszeichen

Bei einfachen Anführungszeichen kennt PHP nur die Escape-Sequenzen `\\` und `\'`.

Wenn Sie versuchen, abhängig davon, welche Anführungszeichen Sie benutzen, eine andere als die erlaubte Escape-Sequenz zu verwenden, wird der Backslash mit ausgegeben.



2.2.5 Nützliche Funktionen

Mit der Funktion `isset` können Sie überprüfen, ob eine Variable existiert.

```
int isset(mixed var)
```

Wenn die Variable existiert, liefert die Funktion `TRUE` zurück, andernfalls `FALSE`.

Die Funktion `unset` hingegen löscht eine Variable, und zwar nicht nur den Wert, sondern die vollständige Variable.

```
int unset(mixed var)
```

Wenn die Variable erfolgreich gelöscht wurde, liefert die Funktion `TRUE` zurück. Wenn nicht, dann `FALSE`.

```
$x = 5;  
echo isset($x);  
unset($x);  
echo isset($x);
```

Auch die Funktion `gettype` ist unter Umständen sehr hilfreich. Sie ermittelt den Typ einer Variablen und gibt eine entsprechende Zeichenkette zurück.

```
$x = 5;
echo "Die Variable \ $x ist vom Typ ";
echo gettype($x);
```

Die Ausgabe:

```
Die Variable $x ist vom Typ integer
```

2.2.6 Typumwandlung

Neben der automatischen Typendefinition, können Sie mit speziellen Anweisungen auch eine explizite Typumwandlung vornehmen.

Der gewünschte Datentyp wird dann in runden Klammern nach dem Zuweisungsoperator `=` notiert.

```
$x = "5";
echo gettype($x);
$x = (int) $x;
echo gettype($x);
```

Folgende Zieldatentypen sind erlaubt:

Datentyp	Erklärung
int, integer	Umwandlung in Ganzzahl
string	Umwandlung in Zeichenkette
float, double, real	Umwandlung in Fließkommazahl
array	Umwandlung in Array
object	Umwandlung in Objekt

Tabelle 2.2 Zieldatentypen für eine explizite Typumwandlung

```
$x = 5.123;
echo gettype($x);
$x = (string) $x;
echo gettype($x);
```

2.3 Arrays

Arrays sind Anordnungen von Variablen. Das zugehörige Äquivalent in Perl sind Listen und Hashes, wobei in PHP diese beiden Typen vereint sind. Aus diesem Grund unterscheidet man auch zwischen drei Array-Gruppen: einfache Arrays, assoziative Arrays und mehrdimensionale Arrays.

Einfache Arrays bestehen aus einer bestimmten Menge von Variablen, die durch einen Index angesprochen werden. Dies kann sowohl bei der Wertzuweisung als auch beim Zugriff erfolgen.

Einfache Arrays

```
$staedte[0] = "Berlin";  
$staedte[1] = "Hamburg";  
$staedte[2] = "Bremen";  
$staedte[3] = "Stuttgart";
```

Die Ausgabe eines Wertes eines Arrays erfolgt dann über den Index, z. B.:

```
echo $staedte[2];
```

Bei der Definition eines Arrays können Sie den Index auch auslassen. Alle neuen Einträge im Array werden dann automatisch mit 0 beginnend indiziert.

```
$staedte[] = "Berlin";  
$staedte[] = "Hamburg";  
$staedte[] = "Bremen";  
$staedte[] = "Stuttgart";
```

Analog sind auch die folgenden Schreibweisen möglich, ohne Festlegung eines bestimmten Index:

```
$staedte = Array("Berlin","Hamburg","Bremen","Stutt-  
gart");
```

Wollen Sie auch bei dieser Schreibweise explizit einen Index festlegen, müssen Sie den Index notieren, gefolgt von => und anschließend dem Wert.

```
$staedte = Array(1 => "Berlin",  
                2 => "Hamburg",  
                0 => "Bremen",  
                3 => "Stuttgart");
```

Assoziative Arrays verwenden anstelle einer Indexnummer einen so genannten Schlüssel.

Assoziative Arrays


```
$laender["DE"] = "Deutschland";
$laender["FR"] = "Frankreich";
$laender["ES"] = "Spanien";
$laender["CH"] = "Schweiz";
```

Dementsprechend ist auch wieder eine kürzere Schreibweise möglich.

```
$laender = Array("DE" => "Deutschland",
                 "FR" => "Frankreich",
                 "ES" => "Spanien",
                 "CH" => "Schweiz");
```

Der Zugriff auf ein Element eines assoziativen Arrays erfolgt dann über den Schlüssel, z. B.:

```
echo $laender["DE"];
```

Mehrdimensi- onale Arrays

Solche Arrays enthalten prinzipiell weitere Arrays und werden durch die Angabe mehrere Indizes angesprochen. Stellen Sie sich dazu einfach eine Tabelle vor, die in Spalten und Zeilen unterteilt ist.

```
$tabelle["A"][1] = "Berlin";
$tabelle["B"][2] = "Hamburg";
$tabelle["C"][1] = "Bremen";
$tabelle["D"][3] = "Stuttgart";
```

Zum Auslesen eines Wertes eines mehrdimensionalen Arrays verwenden Sie wieder die Indizes, z. B.:

```
echo $tabelle["D"][3];
```

2.4 Resource und NULL

Resource und NULL sind zwei sehr spezielle Datentypen, welche Sie sicherlich nicht sehr oft verwenden werden.

Resource Variablen vom Typ Resource sind Referenzen auf eine externe Datenquelle, wie z. B. eine Datenbank. Sie dienen dazu, Zugriffe auf Datenbanken und ähnliche Datenquellen zu erleichtern, indem für jeden offenen Zugriff eine Variable des Typs Resource definiert wird und die ID des Zugriffs in dieser Variablen gespeichert wird. Nur dadurch ist es z. B. möglich, auf zwei Datenbanken gleichzeitig zugreifen zu können.

NULL Verwechseln Sie den Datentyp NULL bitte nicht mit dem Wert 0 (denn die Zahl 0 ist ein Wert). NULL ist nämlich ein Wert, dem »nichts« entspricht.

Selbst ein leerer Wert ist immer noch ein Wert, denn er besitzt einen Zustand, nämlich leer. Das folgende Beispiel soll dies verdeutlichen.

```
$a = 5;
echo isset($a);
$a = NULL;
echo isset($a);
```

Die Zuweisung von `NULL` an `$a` endet mit dem gleichen Effekt wie die Anweisung

```
unset($a);
```

Die Schreibweise von `NULL` ist übrigens egal, es ist sowohl `NULL` als auch `null`, `Null` oder `nULL` möglich. Der Übersicht halber sollten Sie `NULL` aber groß schreiben.



2.5 Operatoren

Auch PHP kennt eine Menge unterschiedliche Operatoren, wie z. B. logische Operatoren oder arithmetische Operatoren. Diese Operatoren sind übrigens stark an Perl und C angelehnt, was wieder einmal auf die Verwandtschaft zu diesen Sprachen verweist.

2.5.1 Arithmetische Operatoren

Arithmetische Operatoren dienen der Berechnung bzw. Verknüpfung von Zahlenwerten.

Operator	Rechenart	Beispiel	Ergebnis
+	Addition	<code>\$x = 5 + 3;</code>	Summe 8
-	Subtraktion	<code>\$x = 10 - 5;</code>	Differenz 5
*	Multiplikation	<code>\$x = 2 * 2;</code>	Produkt 4
/	Division	<code>\$x = 32 / 2;</code>	Quotient 16
%	Modulo	<code>\$x = 13 % 7;</code>	Rest 6

Tabelle 2.3 Arithmetische Operatoren in PHP

Der Ergebnistyp einer Division ist im Übrigen immer vom Typ Fließkommazahl, auch wenn die beiden Operanden vom Typ Ganzzahl sind. Nur durch die explizite Typumwandlung mit `int` oder `integer` erhalten Sie eine Ganzzahl.

```
$x = 13 / 7;  
$x = (int) $x;  
$y = 13 % 7;  
echo "13 geteilt 7 ist gleich $x Rest $y";
```

Die Ausgabe:

```
13 geteilt 7 ist gleich 1 Rest 6
```

Die Verkettung von arithmetischen Operatoren ist auch möglich, wobei die Regel Punkt-vor-Strich-Rechnung zum Tragen kommt.

```
$x = 2 + 2 * 2;      // Ergebnis = 6  
$x = 50 / 10 + 40;  // Ergebnis = 45
```

Die Punkt-vor-Strich-Rechnung lässt sich jedoch auch durch das Setzen von runden Klammern beeinflussen.

```
$x = (2 + 2) * 2;    // Ergebnis = 8
```

Damit auch in PHP nicht solche Schreibweisen wie

```
$x = $x + 1;
```

verwendet werden müssen, um den Wert von `$x` um 1 zu inkrementieren, können Sie die kürzere Schreibweise

```
$x++;
```

verwenden. Dies ist natürlich auch beim Dekrementieren möglich.

```
$x = $x - 1
```

oder kürzer

```
$x--;
```

Es gibt auch eine allgemeinere Verkürzung bei der Verwendung von arithmetischen Operatoren. Konstrukte wie

```
$x = $x + 5;  
$y = $y * 3;  
$z = $z / 10;  
$xyz = $xyz - 30;
```

lassen sich auch folgendermaßen schreiben:

```
$x += 5;  
$y *= 3;
```

```
$z /= 10;
$xyz -= 30;
```

2.5.2 Vergleichsoperatoren

Mit Hilfe von Vergleichsoperatoren ist es möglich, zwei Werte miteinander zu vergleichen. In dieser Hinsicht besticht PHP durch seine Einfachheit. Während in Perl zwischen dem Vergleich von Zahlenwerten und Zeichenketten unterschieden werden muss, können Sie dies in PHP getrost ignorieren.

Das Ergebnis eines Vergleiches ist immer vom Typ Boolean.

Vergleich	Operator	Beispiel	Ergebnis
Sind die Werte gleich	==	1 == 1	Wahr
Sind die Werte gleich	==	1 == 2	Unwahr
Sind die Typen identisch	===	1 === 3	Wahr
Sind die Typen identisch	===	1 === »A«	Unwahr
Sind die Werte ungleich	!=	1 != 2	Wahr
Sind die Werte ungleich	!=	1 != 1	Unwahr
Sind die Werte ungleich	<>	1 <> 2	Wahr
Sind die Werte ungleich	<>	1 <> 1	Unwahr
Sind die Werte nicht identisch	!==	1 !== »A«	Wahr
Sind die Werte nicht identisch	!==	1 !== 2	Unwahr
Ist der linke Wert kleiner	<	1 < 3	Wahr
Ist der linke Wert kleiner	<	3 < 1	Unwahr
Ist der linke Wert größer	>	3 > 2	Wahr
Ist der linke Wert größer	>	2 > 3	Unwahr
Ist der linke Wert kleiner oder gleich	<=	1 <= 2	Wahr
Ist der linke Wert kleiner oder gleich	<=	2 <= 2	Wahr
Ist der linke Wert kleiner oder gleich	<=	3 <= 1	Unwahr
Ist der linke Wert größer oder gleich	>=	5 >= 2	Wahr
Ist der linke Wert größer oder gleich	>=	5 >= 5	Wahr
Ist der linke Wert größer oder gleich	>=	3 >= 5	Unwahr

Tabelle 2.4 Vergleichsoperatoren in PHP



Zusätzlich gibt es noch den Trinitätsoperator, der bei den Vergleichsoperatoren eine Sonderrolle spielt. Er entspricht einem einfachen Entweder-oder-Vergleich. Die Syntax:

```
(ausdruck1) ? (ausdruck2) : (ausdruck3);
```

Ist das Ergebnis des ersten Ausdrucks `TRUE`, wird der zweite Ausdruck zurückgegeben. Ist das Ergebnis des ersten Ausdrucks `FALSE`, wird der dritte Ausdruck zurückgegeben.

Ein Beispiel:

```
$x = 5;  
$a = ($x == 5) ? "fünf" : "nicht fünf";  
echo $a;
```

2.5.3 Zeichenkettenoperatoren

Es gibt eigentlich nur einen Zeichenkettenoperator bzw. zwei, wenn man die Kurzschreibweise des Operator mit hinzuzählt. Um zwei Zeichenketten miteinander verbinden zu können, müssen Sie den Konkatenationsoperator `.` verwenden.

```
$str1 = "Hallo ";  
$str2 = "Welt!";  
$str3 = $str1 . $str2;  
echo $str3;
```

Die Ausgabe:

```
Hallo Welt!
```

Mit der Kurzschreibweise `.=` können Sie folgende Konstrukte

```
$str = "Hallo ";  
$str = $str + "Welt!";
```

durch folgende Schreibweise ersetzen:

```
$str = "Hallo ";  
$str .= "Welt!";
```

2.5.4 Logische Operatoren

Die logischen Operatoren dienen der Verknüpfung mehrerer Teilausdrücke.

```
($a == 5) AND ($b == 3)
```

Der AND-Operator verbindet zwei Teilausdrücke und liefert TRUE, wenn beide Teilausdrücke TRUE sind. Alternativ können Sie für AND auch && verwenden.

UND-
Verknüpfung

```
($a == 5) OR ($b == 3)
```

Der OR-Operator verbindet zwei Teilausdrücke in der Form, dass das Ergebnis TRUE ist, sobald einer der beiden Teilausdrücke TRUE ist. Alternativ können Sie für OR auch || verwenden.

ODER-
Verknüpfung

```
($a == 5) XOR ($b == 3)
```

Nur wenn einer der beiden Teilausdrücke TRUE ist, ist das Ergebnis TRUE. Sind beide Teilausdrücke TRUE oder FALSE, ist auch das Ergebnis FALSE.

ENTWEDER-
ODER

```
!($a == 5)
```

Der !-Operator negiert einen Ausdruck. Das Ergebnis lautet TRUE, wenn \$a == 5 FALSE ist.

Negierung

Auch logische Operatoren unterliegen einer Rangordnung, wie z. B. +, -, * und /. Auch hier können Sie mit runden Klammern die Priorität der einzelnen Operatoren beeinflussen. So hat nämlich auch der &&-Operator Vorrang vor dem ||-Operator, und auch der AND-Operator hat Vorrang vor dem OR-Operator. Jedoch haben die symbolischen Operatoren (&& und ||) wiederum Vorrang vor den literalen (AND und OR) Operatoren.

Beispiel:

```
$a == 5 OR $b == 3 AND $c = 4
// TRUE, wenn $c gleich 4 und $b gleich $ ODER $a
gleich 5
($a == 5 OR $b == 3) AND $c = 4
// TRUE, wenn $c gleich 4 UND $a gleich 5 oder $b
gleich 3
```

2.6 Zusammenfassung

- ▶ Alle Variablen in PHP werden durch \$-Zeichen eingeleitet.
- ▶ Variablen können vom Typ Boolean, Integer, Fließkommazahl, String, Array, Object, Resource oder NULL sein.
- ▶ Arrays werden in einfache, assoziative und mehrdimensionale Arrays unterschieden.
- ▶ Resource-Variablen werden z. B. für Datenbankverbindungen verwendet.
- ▶ NULL ist ein Datentyp, dem »nichts« entspricht.

2.7 Fragen und Übungen

1. Welche Operatoren stehen für arithmetische Operationen zur Verfügung (mathematische Berechnungen)?
2. Welche Vergleichsoperatoren gibt es?
3. Welche Zeichen dürfen Sie bei der Namensgebung von Variablen verwenden?
4. Ist der Bezeichner `$tätäää` erlaubt?

Teil 7

MySQL – Der Datenspeicher

1 MySQL – Der Datenspeicher für Ihre Internetseiten

*Die Untertreibung des Jahres: »Beim Anlegen einer Datenbank fallen nur wenige einfache Schritte an.«
– Einführung in Microsoft Office XP*

Die Einrichtung einer Datenbank ist mit Sicherheit nicht nach ein paar Schritten erledigt, außer natürlich, alle wichtigen Überlegungen wurden vorher angestellt und das Layout der Datenbank geplant. Dann kann in einem abschließenden Schritt die Datenbank eingerichtet werden. Bis diese Überlegungen jedoch vollständig sind, bedarf es ein wenig an Know-how. Denn ohne zu wissen, wie Datenbanken funktionieren und welche Systeme es gibt, wird aus dem Plan eine Datenbank zu erstellen oder gar zu verwenden, schnell ein mühseliges und aufreibendes Unterfangen.

1.1 Mein SQL gib mir heute ...

MySQL ist seit einiger Zeit in aller Munde. Es wird häufig mit Linux oder auch Webservern in Verbindung gebracht. Man könnte behaupten, dass MySQL das Linux unter den Datenbanksystemen ist. Als OpenSource ist es nicht nur kostenlos, sondern auch der Quelltext ist im Internet frei erhältlich. Dies führt ähnlich wie bei Linux dazu, dass sich Programmierer auf der ganzen Welt an der Entwicklung von MySQL beteiligen und sie stetig vorantreiben.

Viele Linux-Distributoren, wie z. B. SuSE oder RedHat, legen MySQL ihren Distributionen sogar bei, und auch Provider für Webspaces setzen als Datenbanksystem in der Regel MySQL ein.

Linux bereits mit MySQL ausgestattet

Der Name MySQL ist auf die Datenbank-Abfragesprache SQL zurückzuführen. Als Anfang der 70er-Jahre die ersten Datenbanksysteme auf den Markt kamen, wurde kurze Zeit später eine einheitliche Sprache entwickelt, mit der es möglich sein sollte, Abfragen an jedes beliebige Datenbanksystem zu stellen, ohne jeweils eine neue Sprache lernen und einsetzen zu müssen. Diese Sprache ist SQL, was für **structured query language** steht und übersetzt strukturierte Abfragesprache bedeutet. Wie der Name schon vermuten lässt, unterstützt auch MySQL die Abfragesprache SQL.

Warum der Name MySQL?

1.2 Datenbanktypen

Seit das erste Datenbanksystem der Öffentlichkeit vorgestellt wurde, sind mittlerweile ca. 30 Jahre vergangen. In dieser Zeit hat sich natürlich viel getan, und so ist es nur normal, dass es mittlerweile Hunderte verschiedener Datenbanksysteme gibt. In all der ganzen Zeit haben sich jedoch nur zwei Datenbanktypen durchgesetzt: relationale und objektorientierte. MySQL gehört zu den relationalen Datenbanksystemen.

1.2.1 Relationale Datenbanksysteme

Die Urform der Datenbanken sind relationale Datenbanken. In solchen Systemen werden Daten in Tabellen gespeichert. Wenn Sie schon einmal mit einer Tabellenkalkulation wie z. B. Excel eine Übersicht erstellt haben, in die Sie alle Bücher eingetragen haben, die in Ihrem Regal stehen, ist dies bereits eine Form von relationaler Datenbank.

Datenbank gleich
Tabelle?

Ich bleibe zunächst bei dem Beispiel der Bücher-Tabelle. Sicherlich hätten Sie diese Tabelle auch strukturiert, sodass Sie zu jedem Buch z. B. den Namen des Autors, den Titel des Buches, die Seitenzahl, die ISBN und den Namen des Verlages notieren können. In der Regel sind dies die Spalten in einer Tabelle. In die nachfolgenden Zeilen haben Sie dann immer den passenden Wert eingetragen. Die Tabelle 1.1 stellt dieses Beispiel einmal schematisch dar.

Autor	Titel	Seitenzahl	ISBN	Verlag
Mark Lubkowitz	Websites	1000	3-89842-313-1	Galileo Press
Christian Wenz	JavaScript	624	3-89842-234-8	Galileo Press

Tabelle 1.1 Beispiel für eine Datenbanktabelle

Die einzelnen Zeilen der Tabelle 1.1 werden bei Datenbanken auch Datensatz genannt. Ein solcher Datensatz besteht aus mehreren Feldern. In unserem Fall sind dies fünf Felder, nämlich Autor, Titel, Seitenzahl, ISBN und Verlag.

Auf die gleiche Art werden auch Daten in einer relationalen Datenbank gespeichert. Das oberste Element ist die Datenbank, welche sich in mehrere Tabellen unterteilt. Eine Tabelle wiederum unterteilt sich in mehrere Datensätze und diese schlussendlich in mehrere Felder.

1.2.2 Objektorientierte Datenbanksysteme

Die Strukturierung und Ablage einer objektorientierten Datenbank geht einen anderen Weg. Die Daten werden nicht als Datensatz in einer Tabelle gespeichert, sondern als Objekt. Die Entwicklung dieser System erfolgte erst Mitte der 80er-Jahre, als auch die objektorientierte Programmierung (OOP) langsam, aber sicher den Markt der Programmiersprachen eroberte.

Der Vorteil dieser Systeme in Zusammenhang mit der OOP ist, dass die Objekte nicht zuerst in einen Datensatz umgewandelt werden müssen, bevor sie in der Datenbank gespeichert werden und umgekehrt, der Datensatz nicht erst in ein Objekt umgewandelt werden muss, wenn er aus einer Datenbank ausgelesen wird.

Vorteil

Das oberste Element ist auch wiederum die Datenbank, gefolgt von der Klasse. Die Klasse enthält dann die Objekte. Somit werden Objekte also direkt in die entsprechende Klasse der Datenbank eingefügt oder entsprechend ausgelesen.

Aus Tabelle wird Klasse

Eigenartiger Weise konnten sich objektorientierte Datenbanken nicht annähernd so erfolgreich durchsetzen wie objektorientierte Programmiersprachen, was sicherlich auf die Inkompatibilität zurückzuführen ist. Denn der Zugriff auf eine solche Datenbank mit einer nicht objektorientierten Programmiersprache gestaltet sich äußerst schwierig bzw. ist nahezu unmöglich. Relationale Datenbanken hingegen können von prinzipiell jeder Programmiersprache angesprochen werden, egal ob objektorientiert oder nicht.

Kein Durchsetzungsvermögen

1.3 Redundanz und Inkonsistenz

Da in den Anfängen der Datenbanken Speicher noch nicht in den Größen und günstigen Preisen verfügbar war wie heutzutage, versuchte man stets, an allen Ecken zu sparen. Dies führte bekanntermaßen zu dem Jahr-2000-Fehler, da Jahreszahlen nur zweistellig gespeichert wurden und somit ein paar Byte eingespart werden konnten. Wer ahnte damals, dass solch ein System über 20 Jahre lang eingesetzt werden würde?

Sei's drum, auch bei Datenbanken ging man den Weg, Speicher einzusparen. Hauptsächlich indem man versuchte, Redundanz zu vermeiden. Anstatt häufig wiederkehrende Werte abzuspeichern, wurden solche Werte in eine zusätzliche Tabelle verlagert. Eine solche Redundanz ist z. B. auch in Tabelle 1.1 wiederzufinden. Als Verlag wurde zweimal Galileo Press gespeichert. Bei lediglich zwei Einträgen mag dies nicht weiter pro-

Redundanz

blematisch sein. Die Redundanz ist sehr gering, wenn es aber hundert Datensätze wären, die alle als Verlag Galileo Press enthalten, ist die Redundanz schon relativ groß.

Inkonsistenz Problematisch wird dies auch, wenn Sie nun Galileo Press durch Galileo Computing ersetzen möchten. Sie müssten nun alle Datensätze überarbeiten, mit der Gefahr, einen Datensatz zu übersehen. Redundanz führt dann also zu Inkonsistenz, da die abgelegten Daten nicht mehr unbedingt plausibel sind.

Relation Würden Sie die Verlage und die Namen nun in eine zusätzliche Tabelle auslagern, würde zum einen die Redundanz verringert werden und zum anderen mögliche Inkonsistenz vermieden. Gehen Sie einmal davon aus, die Tabelle würde den Namen »Verlage« erhalten. Die Tabelle könnte nun die Felder »Name« und »Webseite« enthalten. Zusätzlich wird dieser Tabelle noch ein Feld hinzugefügt: »ID«. Dieses Feld soll einen eindeutigen Wert enthalten, der innerhalb der Tabelle nur einmal vorkommt und so den Datensatz genau identifiziert. Dieses Feld wird auch Primärschlüssel genannt.

Anstatt nun den Namen des Verlages in der Tabelle mit den Büchern zu speichern, wird dort lediglich der entsprechende Primärschlüssel des Verlages gespeichert.

ID	Verlag	Webseite
1	Galileo Press	www.galileo-press.de

Tabelle 1.2 Datenbanktabelle Verlage

In der Tabelle »Bücher« könnte das nun folgendermaßen aussehen.

Autor	Titel	Seitenzahl	ISBN	Verlag
Mark Lubkowitz	Websites	1000	3-89842-313-1	1
Christian Wenz	JavaScript	624	3-89842-234-8	1

Tabelle 1.3 Datenbanktabelle Bücher mit Bezug auf Verlage

Um nun den Namen des Verlages zu ändern, müsste lediglich der entsprechende Datensatz in der Tabelle »Verlage« geändert werden und zwar nur einmal.

Durch diese Technik werden also Redundanz und Inkonsistenz, jedoch auch übergroße Tabellen vermieden. Im nächsten Schritt könnten nun z. B. die Autoren in eine Tabelle mit dem Namen »Autoren« verschoben und in der Tabelle »Bücher« könnte anstatt des Namens einfach wieder ein Primärschlüssel angegeben werden.

1.4 Zusammenfassung

- ▶ MySQL ist ein kostenloses Datenbanksystem.
- ▶ Die meist verbreitetsten Datenbanksysteme sind relationale und objektorientierte.
- ▶ MySQL ist ein relationales Datenbanksystem.
- ▶ Als Abfragesprache wird bei MySQL die Structured Query Language, kurz SQL, verwendet.

1.5 Fragen und Übungen

1. Wo liegt der Unterschied zwischen relationalen und objektorientierten Datenbanksystemen?
2. Was ist Redundanz?
3. Was ist Inkonsistenz?
4. Wie lassen sich Redundanz und Inkonsistenz vermeiden?

2 Die Sprache SQL

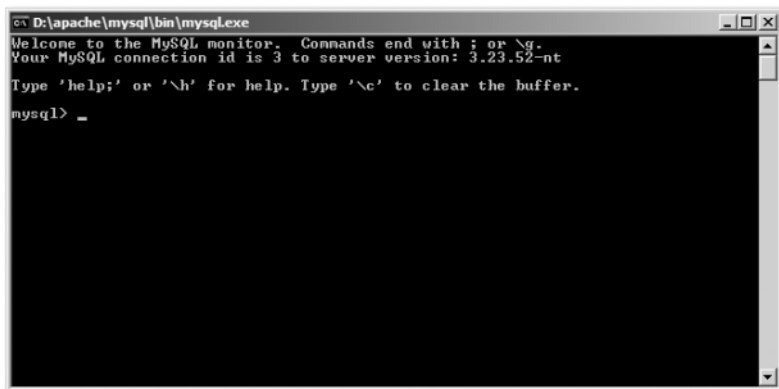
Ich habe die Länge und Breite dieses Landes bereist und mit den besten Leuten geredet, und ich kann Ihnen versichern, dass Datenverarbeitung ein Tick ist, der dieses Jahr nicht überleben wird.

– *The editor in charge of business books for Prentice Hall, 1957*

In diesem Kapitel werden ich Ihnen den grundlegenden Aufbau der Abfragesprache SQL erklären. Damit Sie die einzelnen Konstrukte auch testen können, werde ich Ihnen außerdem den Umgang mit der Konsole von MySQL erklären.

2.1 Einstieg

Nun zur Konsole von MySQL, die ein sehr mächtiges Werkzeug darstellt. Sie kann sowohl zur Administration als auch zum Testen von Abfragen verwendet werden. Wechseln Sie in das Installationsverzeichnis von MySQL, und öffnen Sie das Verzeichnis **bin**. Öffnen Sie das Programm **mysql.exe**. Sie sollten nun eine Konsole sehen, die derjenigen in Abbildung 2.1 ähnelt.



```
D:\apache\mysql\bin\mysql.exe
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 3 to server version: 3.23.52-nt
Type 'help;' or '\h' for help. Type '\c' to clear the buffer.
mysql> _
```

Abbildung 2.1 Konsole von MySQL

Geben Sie nun **help** ein und bestätigen Sie mit . Die Konsole gibt nun eine Übersicht über die einzelnen Befehle mit Erklärungen aus. Die wichtigsten dieser Befehle finden Sie in Tabelle 2.1.

Befehl	Erklärung
help	Gibt eine Übersicht aller Befehle aus
use	Wählt die Datenbank aus, die als Parameter an den Befehl übergeben wurde
exit	Schließt die Konsole
status	Gibt den Status der MySQL-Datenbank aus
source	Führt ein SQL-Skript aus, das als Parameter an den Befehl übergeben wurde
clear	Löscht den Befehlspeicher

Tabelle 2.1 Die wichtigsten MySQL-Konsolen-Befehle

Datenbank auswählen

Mit dem Kommando `use` können Sie eine Datenbank auswählen. Schon von Beginn an verfügt MySQL über zwei Datenbanken: `mysql` und `test`. Geben Sie nun **`use mysql;`** (inkl. des Semikolons) ein. Die Konsole wird bei korrekter Eingabe mit

```
Database changed
```

antworten.

Abfrage ausführen

Tippen Sie nun **`SELECT * FROM db;`** in die MySQL-Konsole ein. Die nun in der Konsole folgende Ausgabe ist der Beweis für Ihre erste erfolgreiche Datenbankabfrage, auch wenn die Ausgabe in der Konsole zu wünschen übrig lässt. Um ein leserliches Ergebnis zu erhalten, können Sie auch einmal **`SELECT user FROM db;`** ausprobieren. Die Ausgabe ist:

```
+-----+
| host |
+-----+
| %    |
+-----+
1 row in set (0.00 sec)
```

Anhand dieser Ausgabe können Sie erkennen, dass die Tabelle einen Datensatz enthält (1 row in set) und dass es 0.00 Sekunden gedauert hat, bis die Abfrage vollständig durchgeführt wurde.

2.2 Namenskonventionen

Bei der Benennung von Datenbanken, Tabellen oder Spalten müssen Sie eine gewisse Namenskonvention einhalten, die Ihnen vorschreibt, welche Zeichen und in welcher Zusammensetzung für den Namen verwenden

det werden dürfen bzw. sollten. Dies bezieht sich auch auf die Schreibweise von SQL-Anweisungen.

Zum Bezeichnen von Datenbanken, Tabellen und Spalten dürfen Sie alle alphanumerischen Zeichen, also Buchstaben und Zahlen von 0 bis 9, verwenden. Zusätzlich sind der Unterstrich `_` und das Dollarzeichen `$` erlaubt. Dabei darf der Name mit einem beliebigen Zeichen anfangen, jedoch darf ein Name nicht nur aus Zahlen bestehen. Außerdem ist die Länge eines Namens auf 64 Zeichen beschränkt.

Bezeichnungen

Einige Beispiele für erlaubte Namen:

```
datenbank
tabelle123
db_12_tabelle$
$tab123
```

Beispiele für ungültige Namen:

```
09876
<tabelle>
tabelle buecher
tabelle=buecher
```

Die Groß- und Kleinschreibung bei Namen hängt stark vom eingesetzten Betriebssystem ab. So wird in UNIX/Linux-Umgebungen z. B. bei Datenbanknamen zwischen Groß- und Kleinschreibung unterschieden, während dies unter Windows keinen Unterschied macht. Dies gilt ebenso für Tabellen. Bei Spalten hingegen wird sowohl unter UNIX/Linux als auch Windows die Groß- und Kleinschreibung ignoriert.

Groß- und Kleinschreibung

Bei Schlüsselwörtern wird nie zwischen Groß- und Kleinschreibung unterschieden, egal ob unter UNIX/Linux oder Windows.



Dies führt auch dazu, dass sich für die Groß- und Kleinschreibung gewisse Regeln eingebürgert haben. So werden SQL-Schlüsselwörter immer großgeschrieben und Namen von Datenbanken, Tabellen oder Spalten immer klein. Dies trägt stark zur Übersicht in komplexen SQL-Anweisungen bei.

2.3 Datenbank erstellen, löschen oder auswählen

Bevor ich eingehender auf Auswahlabfragen eingehe, will ich mich zunächst dem Erstellen und Löschen von Datenbanken widmen. MySQL stellt Ihnen als Benutzer schließlich die Möglichkeit zur Verfügung, nicht

nur eine Datenbank zu verwenden, sondern mehrere. So können Sie logische Gruppierungen verschiedener Tabellen in verschiedene Datenbanken vornehmen. Das Erstellen einer Datenbank erfolgt mittels des Befehls `CREATE DATABASE`.

```
CREATE DATABASE datenbankname
```

Dem Befehl müssen Sie zusätzlich noch einen Namen für die anzulegende Datenbank übergeben. Ein Beispiel:

```
CREATE DATABASE buecher;
```

Dieses Beispiel würde die Datenbank `buecher` anlegen.

Datenbank löschen

Natürlich können Sie erstellte Datenbanken auch wieder löschen. Der entsprechende Befehl dafür lautet `DROP DATABASE`.

```
DROP DATABASE datenbankname
```

Ein Beispiel:

```
DROP DATABASE buecher;
```

Dieser Befehl würde die Datenbank `buecher` löschen.



An dieser Stelle ein wichtiger Hinweis: Wenn Sie mittels `DROP DATABASE` eine Datenbank gelöscht haben, ist diese unwiederbringlich entfernt, inkl. aller enthaltenen Tabellen und Datensätze. Es ist nicht möglich, diese Datenbank wiederherzustellen.

Datenbank auswählen

Mit dem `USE`-Befehl können Sie eine Datenbank auswählen.

```
USE datenbankname
```

Ein Beispiel:

```
USE buecher;
```

Dieses Beispiel würde die Datenbank `buecher` als aktuelle Datenbank auszeichnen, und alle nachfolgenden SQL-Befehle würden sich nun auf diese Datenbank beziehen. Abgesehen natürlich vom Befehl `USE`.

2.4 Tabellen erstellen und löschen

Ohne Tabellen können Sie in einer Datenbank natürlich keine Datensätze speichern. Um eine neue Tabelle zu erstellen, steht der Befehl `CREATE TABLE` zur Verfügung.

```
CREATE TABLE tabellenname (spaltenname datentyp [optionen], [, ...] [, PRIMARY KEY spaltenname])
```

Sowohl die Angabe eines Namens für die Tabelle als auch die Definition mindestens einer Spalte inkl. Datentyp ist zwingend. Ein Beispiel:

```
CREATE TABLE buecher (id INTEGER AUTO_INCREMENT,  
    autor VARCHAR(60),  
    titel VARCHAR(60),  
    seitenzahl INTEGER,  
    isbn VARCHAR(15),  
    verlag INTEGER,  
    PRIMARY KEY (id));
```

Dieses Beispiel legt nun eine Tabelle mit 6 Spalten an. Als Erstes wird eine Spalte namens `id` mit dem Datentyp `INTEGER` angelegt. Als Option für diese Spalte wurde `AUTO_INCREMENT` festgelegt, was bedeutet, dass der Wert dieser Spalte automatisch um 1 hochgezählt wird. Der Datentyp `INTEGER` bedeutet, dass in dieser Spalte ausschließlich Zahlen gespeichert werden können. Dann folgt die Spalte `autor`, die als Datentyp `VARCHAR` zugewiesen bekommt. Der Typ `VARCHAR` steht für eine beliebige Zeichenkette. In Klammern dahinter wird die Länge dieser Spalte festgelegt. In unserem Beispiel 60, was bedeutet, dass ein in der Spalte gespeicherter Wert maximal 60 Zeichen lang sein darf. Dann folgt die Spalte `titel`, die ebenfalls vom Typ `VARCHAR` ist und eine Maximallänge von 60 Zeichen erhält. Die Spalte `seitenzahl` ist vom Typ `INTEGER`, während die Spalte `isbn` wieder vom Datentyp `VARCHAR` ist. Diesmal wurde aber lediglich eine Länge von 15 Zeichen festgelegt. Als letzte Spalte folgt `verlag`, welche den Datentyp `INTEGER` zugewiesen bekommt. In dieser Spalte wird später nur die Kennnummer eines Verlages gespeichert und kein Text. Zum Schluss wird mit `PRIMARY KEY (id)` festgelegt, dass die Spalte `id` der Primärschlüssel der Tabelle werden soll.

Wurde der Befehl erfolgreich durchgeführt, bestätigt die MySQL-Konsole mit Query OK. Falls Sie nun noch einmal die Tabelle auf ihre Struktur überprüfen möchten, können Sie den Befehl `EXPLAIN` verwenden.

Tabellenstruktur anzeigen

```
EXPLAIN tabellenname
```

Beispiel:

```
EXPLAIN buecher;
```

Die MySQL-Konsole gibt nun eine Beschreibung der Tabelle aus. Die Ausgabe der Konsole sollte der in Abbildung 2.2 gezeigten entsprechen.

```

c:\d:\apache\mysql\bin\mysql.exe
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 5 to server version: 3.23.52-nt

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> use buecher;
Database changed
mysql> explain buecher;
+----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+----+-----+-----+-----+-----+-----+
| id | int(11) | YES | PRI | NULL | auto_increment |
| autor | varchar(60) | YES | | NULL | |
| titel | varchar(60) | YES | | NULL | |
| seitenzahl | int(11) | YES | | NULL | |
| isbn | varchar(15) | YES | | NULL | |
| verlag | int(11) | YES | | NULL | |
+----+-----+-----+-----+-----+-----+
6 rows in set (0.04 sec)

mysql>

```

Abbildung 2.2 Konsolen-Ausgabe nach Ausführung des Befehls EXPLAIN buecher;

Tabelle löschen Natürlich ist es auch möglich, eine Tabelle zu löschen. Der zu verwendende Befehl lautet DROP TABLE.

DROP TABLE tabellenname

Ein Beispiel:

DROP TABLE buecher;

Dieser Befehl würde die Tabelle buecher inklusive aller Datensätze unwiederbringlich löschen.

2.5 Tabellen verändern

Eine bereits existierende Tabelle können Sie auch nachträglich verändern bzw. anpassen. Sie können Spalten löschen, hinzufügen, den Datentyp einer Spalte umwandeln oder auch den Primärschlüssel ändern. Der Befehl dafür lautet ALTER TABLE.

ALTER TABLE tabellenname vorgang [, vorgang, ...]

Mit dem Befehl ALTER TABLE weisen Sie MySQL zuerst an, dass Sie eine Änderung an der Struktur einer Tabelle vornehmen möchten. Die zu verändernde Tabelle muss dabei angegeben werden. Welche Änderung Sie dann genau durchführen möchten, geben Sie als Vorgang an. Folgende Vorgänge sind möglich.

- ▶ ADD – Fügt der Tabelle eine Spalte hinzu. Die Definition der Spalte erfolgt dabei äquivalent zur Definition in Zusammenhang mit CREATE TABLE, z. B. ALTER TABLE buecher ADD jahr INTEGER;.

- ▶ **DROP** – Löscht eine Spalte einer Tabelle. Es genügt dabei die Angabe des Spaltennamens, z. B. `ALTER TABLE buecher DROP jahr;`. Achten Sie darauf, dass bereits vorhandene Datensätze von einer solchen Änderung beeinflusst werden, da die Daten der Spalte ebenfalls gelöscht werden.
- ▶ **ADD PRIMARY KEY** – Fügt der Tabelle einen Primärschlüssel hinzu, z. B. `ALTER TABLE buecher ADD PRIMARY KEY (verlag);`.
- ▶ **DROP PRIMARY KEY** – Löscht den Primärschlüssel einer Tabelle, z. B. `ALTER TABLE buecher DROP PRIMARY KEY;`.
- ▶ **CHANGE** – Ändert die Definition einer Spalte inkl. des Namens. Es wird der alte und der neue Spaltenname sowie die Definition der Spalte erwartet. Die Definition erfolgt äquivalent zur Definition bei `CREATE TABLE`. Beispiel: `ALTER TABLE buecher CHANGE seitenzahl seiten integer;`.
- ▶ **MODIFY** – Modifiziert die Definition einer Spalte, ohne den Namen zu ändern, z. B. `ALTER TABLE buecher MODIFY autor VARCHAR(128);`.

2.6 MySQL-Datentypen

Zwei Datentypen haben Sie bereits kennen gelernt: `INTEGER` und `VARCHAR`. MySQL kennt jedoch noch einige Datentypen mehr. Dabei wird zwischen drei Gruppen unterschieden: numerische Typen, Zeichen- oder Zeichenketten-Typen und vermischte Typen.

In den nachfolgenden Tabellen finden Sie den Namen des Datentyps, den Wertebereich bzw. die maximale Größe und den in der Datenbank belegten Speicherplatz.

2.6.1 Numerische Typen

Datentyp	Wertebereich	Speicherplatz
TINYINT	-128 bis 127	8 Bit
SMALLINT	-32.768 bis 32.767	16 Bit
MEDIUMINT	-8.388.608 bis 8.388.607	24 Bit
INTEGER	-2.147.483.648 bis 2.147.483.647	32 Bit
BIGINT	-9.223.372.036.854.775.808 bis 9.223.372.036.854.775.807	64 Bit

Tabelle 2.2 Übersicht aller numerischen Datentypen

Datentyp	Wertebereich	Speicherplatz
FLOAT	Abhängig von den Werten bzw. Nachkommastellen	32 Bit
DOUBLE	Abhängig von den Werten bzw. Nachkommastellen	64 Bit
DECIMAL	Abhängig von den Werten bzw. Nachkommastellen	x+16 Bit

Tabelle 2.2 Übersicht aller numerischen Datentypen (Forts.)

2.6.2 Zeichen- und Zeichenketten-Typen

Datentyp	Maximale Größe	Speicherplatz
CHAR(x)	255 Byte	x Byte
VARCHAR(x)	255 Byte	x+1 Byte
TINYTEXT	255 Byte	x+1 Byte
TINYBLOB	255 Byte	x+1 Byte
TEXT / BLOB	65535 Byte	x+2 Byte
MEDIUMTEXT	16.777.215 Byte = ca. 16 Mbyte	x+3 Byte
MEDIUMBLOB	16.777.215 Byte = ca. 16 Mbyte	x+3 Byte
LONGTEXT	4.294.967.295 Byte = ca. 4 Gbyte	x+4 Byte
LOBLOB	4.294.967.295 Byte = ca. 4 Gbyte	x+4 Byte

Tabelle 2.3 Übersicht aller Zeichen- und Zeichenketten-Datentypen

2.6.3 Vermischte Typen

Datentyp	Erklärung
ENUM('A','B',...)	Eine Liste mit maximal 65.535 Posten. Gespeichert wird immer nur einer der in der Liste angegebenen Posten.
SET('A','B',...)	Eine Listen mit maximal 64 Posten. Speicher alle Posten der Liste.
DATE	Datum im Format YYYY-MM-DD
TIME	Uhrzeit im Format HH:MM:SS
DATETIME	Datum und Zeit im Format YYYY-MM-DD/HH:MM:SS

Tabelle 2.4 Übersicht aller vermishten Datentypen

2.6.4 Was ist ein BLOB?

Ein BLOB ist ein **binary large object**, in Deutsch ein großes binäres Objekt. Der Unterschied zwischen einem TEXT-Typ und einem BLOB-Typ entspricht in etwa demjenigen bei Dateien. Während TEXT-Typen in der Datenbank als Zeichen gespeichert werden, werden BLOB-Typen in der Datenbank binär abgelegt. Dies hängt hauptsächlich mit den Steuerbytes für einen Zeilenumbruch zusammen. Bei TEXT-Typen sind dies in der Regel CR und LF (**carriage return/line feed**) und bei BLOB-Typen nur LF. Würde in binären Daten eine Bitfolge vorkommen, die dem CR entspricht, könnte dies zu fehlerhaften Daten führen. TEXT-Typen werden also normalerweise zum Speicher von Texten und BLOB-Typen zum Speichern von Bildern oder Geräuschen verwendet.

2.6.5 Optionen

Zusätzlich zu der Option `AUTO_INCREMENT` gibt es auch noch weitere Optionen, die Sie zu einer Spalte angeben können. Einige dieser Optionen haben auch Einfluss auf den Wertebereich eines Datentyps. So führt die Option `UNSIGNED` z. B. dazu, dass numerische Datentypen dann kein Vorzeichen besitzen können. Der Wertebereich einer `TINYINT` würde sich also von -128 bis 127 auf 0 bis 255 ändern.

- ▶ `AUTO_INCREMENT` – Zählt den Wert automatisch um 1 hoch; darf nur im Zusammenhang mit Ganzzahlen (`...INT`) verwendet werden und nur einmal pro Tabelle festgelegt werden.
- ▶ `BINARY` – Die Daten werden binär gespeichert.
- ▶ `DEFAULT` – Ermöglicht das Festlegen eines Standardwertes, falls kein Wert angegeben wurde. Normalerweise ist dies `NULL`. Wurde jedoch die Option `NOT NULL` gesetzt, darf in der Spalte kein `NULL` vorkommen. In einem solchen Fall wird der mit `DEFAULT` angegebene Wert eingesetzt.
- ▶ `NULL/NOT NULL` – Mit `NULL` oder `NOT NULL` können Sie festlegen, ob in der Spalte ein Wert enthalten sein muss. Ist die Option `NOT NULL` gesetzt, muss in der Spalte ein Wert eingefügt werden. Dieser kann dann auch durch `DEFAULT` festgelegt werden.
- ▶ `PRIMARY KEY` – Definiert eine Spalte als Primärschlüssel. Diese Spalte muss einen Wert enthalten (darf also nicht `NULL` sein).
- ▶ `UNIQUE` – Diese Option bewirkt, dass in einer Spalte keine doppelten Werte vorkommen dürfen, jeder Wert also einmalig sein muss.

- ▶ `UNSIGNED` – In der Spalte sind im Zusammenhang mit `INT`-Typen keine vorzeichenbehafteten Werte erlaubt, also keine negativen Zahlen.
- ▶ `ZEROFILL` – Füllt den Wert vom Typ `INT` mit führenden Nullen auf. Aus dem Wert 23 wird dann z. B. 000023, wenn die maximale Länge mit `INT(6)` auf sechs festgelegt wurde.

2.7 Zusammenfassung

- ▶ Bei Datenbanken und Tabellen wird abhängig vom Betriebssystem zwischen Groß- und Kleinschreibung unterschieden, nicht jedoch bei Schlüsselwörtern oder Spaltennamen. Aus diesem Grund werden Schlüsselwörter großgeschrieben und Datenbank-, Tabellen- und Spaltennamen kleingeschrieben.
- ▶ Datenbanken können mit dem Befehl `CREATE DATABASE` erzeugt und mit `DROP DATABASE` gelöscht werden.
- ▶ Tabellen werden mit den Befehlen `CREATE TABLE` und `DROP TABLE` angelegt oder gelöscht.
- ▶ Der Befehl `ALTER TABLE` ermöglicht das Ändern der Struktur einer Tabelle.
- ▶ Bei den Datentypen wird zwischen numerischen Typen, Zeichen- oder Zeichenketten- und vermischten Typen unterschieden.
- ▶ Unterschiedliche Optionen nehmen Einfluss auf die Werte, die in einer Spalte eingefügt werden dürfen.

2.8 Fragen und Übungen

1. Welche Zeichen dürfen Sie bei der Vergabe von Datenbank-, Tabellen- und Spaltennamen verwenden? Welche Regeln müssen Sie dabei einhalten?
2. Erstellen Sie eine SQL-Anweisung, welche eine Datenbank mit dem Namen `buecher` erstellt.
3. Erstellen Sie eine SQL-Anweisung, welche eine Tabelle mit dem Namen `verlage` und folgenden Spalten anlegt: `id`, `name`, `webseite`, `strasse`, `ort` und `plz`. Wählen Sie sinnvolle Datentypen für die einzelnen Spalten und weisen Sie der Spalte `id` die Optionen `AUTO_INCREMENT` und `PRIMARY KEY` zu.
4. Erstellen Sie eine SQL-Anweisung, welche eine Tabelle namens `buecher` mit den folgenden Spalten anlegt: `id`, `autor`, `titel`, `seiten-`

zahl, erscheinungsjahr, preis, auflage, verlag, isbn und kommentar. Die Spalte id soll Primärschlüssel sein und automatisch hochgezählt werden. In die Spalten autor und titel soll jeweils eine Zeichenkette von maximal 128 Zeichen eingefügt werden dürfen, und sie sollen nicht NULL sein. Standardmäßig soll dann unbekannt eingefügt werden. Die Spalte erscheinungsjahr soll eine vierstellige Jahresangabe sein. Die Spalte preis soll eine Dezimalzahl mit fünf Vor- und zwei Nachkommastellen enthalten, darf ebenfalls nicht NULL sein und soll als Standardwert dann 0.00 erhalten. Die Spalte auflage soll vom Typ INT sein. In der Spalte verlag soll eine Ganzzahl gespeichert werden können. Die Spalte isbn wiederum soll eine maximal 15 Zeichen lange Zeichenkette und die Spalte kommentar einen längeren Kommentartext bis zu 65.535 Zeichen enthalten können.