Jens Kilgenstein

CSS

einfach erklärt

Eine Einführung ins Webdesign anhand konkreter Beispiele

www.css-einfach.de

Vorwort

Detailwissen ist beim Erlernen von CSS zweitrangig. Deswegen sind anfangs weder dicke Bücher hilfreich, die alle erdenklichen Befehle durchkauen, noch Internetseiten, die Lösungen für Spezialprobleme bieten. Notwendig ist es vielmehr, die grundsätzlichen Prinzipien, auf denen CSS beruht, zu verstehen. Wer z. B. den Aufbau mehrspaltiger Layouts beherrschen will, muss zunächst die Grundlagen der Positionierung verstanden haben. Ebenso wird man vor scheinbar unlösbaren Problemen stehen, wenn man nicht versteht, wie der Browser einen Quelltext einliest und interpretiert. Ziel dieses Buches ist es, genau dieses zwingend notwendige Basiswissen leicht verständlich zu vermitteln. Hierzu erstellen wir eine konkrete Webseite und fangen buchstäblich bei null an. Wir durchlaufen genau die Schritte, die jeder Webentwickler auch abarbeiten muss. Theoretische Grundlagen werden nur so weit erläutert, wie es die Praxis verlangt. Die einzelnen Schritte sind bewusst einfach gehalten und werden stets mit Beispielen flankiert. Am Ende des Buches werden Sie in über fünfzig Etappen Ihre Webseite komplettiert haben. Durch diese Art des Lernens am konkreten Beispiel werden Sie nicht nur die Fähigkeit erlangen, eigenen Quellcode von Grund auf selbst zu entwickeln. Sie werden auch fremde Codes verstehen und verändern können, was Sie in die Lage versetzt, beispielsweise das Design einer Blog-Software anzupassen.

Auf der Webseite zum Buch können Sie sich auch die einzelnen Codebeispiele herunterladen. Dort finden Sie auch vertiefende Hinweise, Neuigkeiten und weiterführende Links. Die Adresse lautet:

► http://www.css-einfach.de/leserbereich

Viel Spaß beim Scripten!

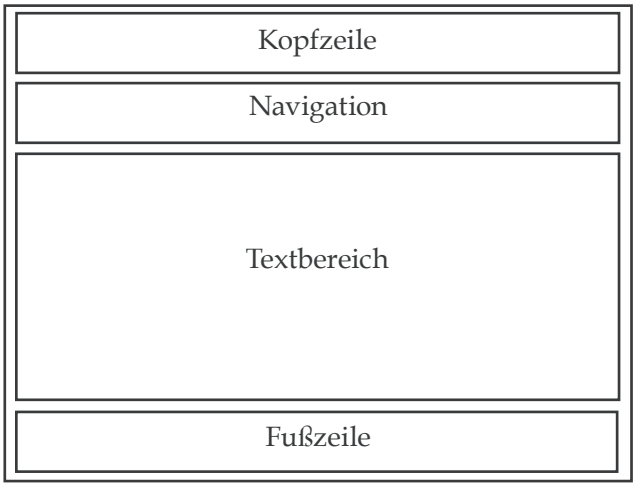

Abb. 3: Einspaltiges Layout mit vier Bereichseinteilungen

Häufig anzutreffen sind auch komplexere, mehrspaltige Layouts. Im nachfolgenden Beispiel wurde zusätzlich ein Bereich für Nebeninhalte angelegt (als „Textbereich 2“ gekennzeichnet) und die Navigation auf der linken Seite senkrecht angeordnet:

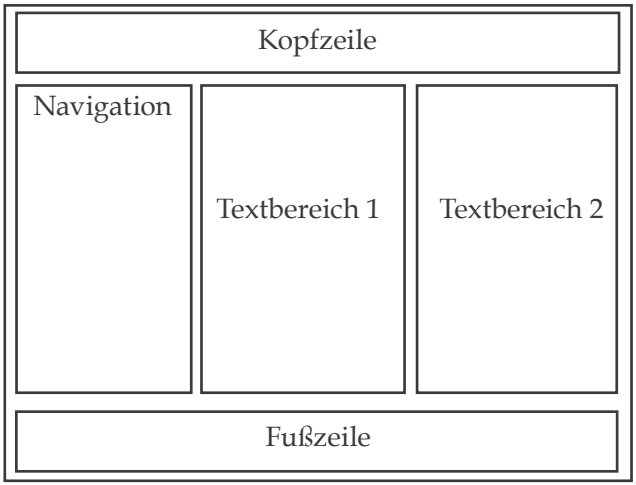

Abb. 4: Mehrspaltiges Layout mit fünf Bereichseinteilungen

Die zwei soeben aufgezeigten Strukturierungen sind bei den allermeisten Seiten anzutreffen. Sie sind aber keinesfalls zwingend, sondern können nach den individuellen Bedürfnissen beliebig hinzugefügt oder weggelassen werden. Wichtig ist, dass man sich selber vergegenwärtigt, dass solche **Bereichseinteilungen in Form rechteckiger**

Kästen realisiert werden. Runde Strukturen kennt HTML folglich nicht. Anstelle des Wortes Kasten wird häufig auch die Bezeichnung Container, Box, Bereich oder Kiste verwendet, gemeint ist aber in diesem Zusammenhang immer dasselbe.

Ergänzt wird die Strukturierung häufig durch einen weiteren Bereich, der inhaltlich keine Bedeutung hat, allerdings beim späteren Designen mit CSS wichtig wird. Er wird oft als „Wrapper" bezeichnet, was auf Deutsch so viel wie „Schutzhülle" heißt. Ähnlich wie ein Schutzumschlag bei einem gebundenen Buch umschließt der Wrapper die anderen Bereiche und ist wichtig, um die komplette Seite zentrieren zu können. In ihn werden jegliche andere HTML-Objekte geschrieben, die beim Bau der Seite eine Rolle spielen.

Doch mit welchen Tags wird die Einteilung in Bereiche konkret realisiert? Nun, mit dem sogenannten ***div***-Tag wird eine HTML-Seite strukturiert. Mit **<*div*>** leiten Sie einen Bereich ein, in den Sie mehrere Elemente einschließen können (div = division = Bereich). Alles, was zwischen dem öffnenden **<*div*>** und dem abschließenden **</*div*>** steht, wird als Teil des Bereichs interpretiert. Sie können so also mehrere Absätze, bestehend aus ganz verschiedenen Elementen wie Text, Grafiken, Tabellen usw., in einen gemeinsamen Bereich einschließen und dann gemeinsam ausrichten. Um im HTML-Quelltext die einzelnen Bereiche voneinander unterscheiden zu können, erhalten Sie mithilfe des Attributs ***id*** eine eindeutige Namenszuordnung, die zwischen die beiden Anführungszeichen geschrieben werden. Die Namen können Sie frei wählen, d. h. ob Sie den Namen „kopfzeile" oder „obere Zeile" wählen, ist vollkommen egal. Bei dem oben genannten Beispiel mit den vier strukturellen Unterteilungen plus Schutzumschlag würden wir den Quelltext folgendermaßen ergänzen:

```
<div id=”wrapper”>
  <div id=”kopfzeile”> </div>
  <div id=”navigation”> </div>
  <div id=”textbereich”> </div>
  <div id=”fusszeile”> </div>
</div>
```

Wenn wir jetzt diese insgesamt fünf Bereichseinteilungen auf die Abbildung 3 anwenden, ergibt sich folgende Struktur:

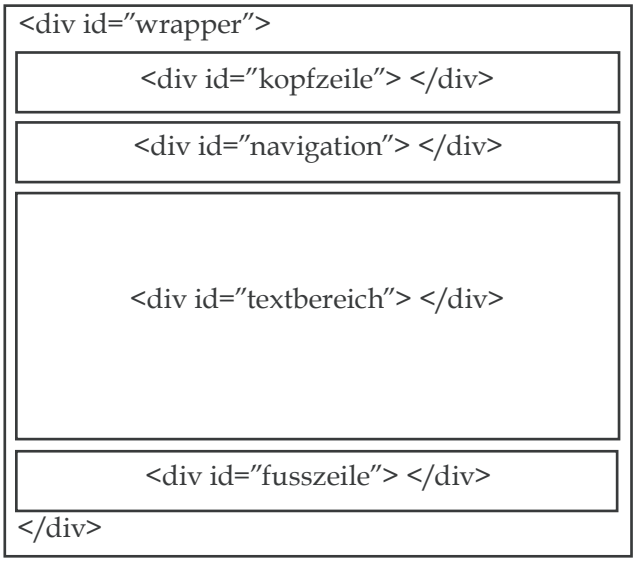

Abb. 5: Fünf Bereichseinteilungen im Quelltext

To-Do 2: Die HTML-Datei mit Hilfe des div-Tags in Bereiche einteilen
1. Öffnen Sie Datei „to-do1.html".
2. Ergänzen sie den Quelltext im body-Bereich mit den div-Tags für die Bereiche "wrapper", "kopfzeile", "navigation", "textbereich" und "fusszeile".
3. Speichern sie die Datei unter dem Namen to-do2.html.

div-Elemente werden als **Blockelemente** bezeichnet, da das öffnende und das schließende **<*div*>**-Tag **zu einem Zeilenumbruch führen**. Ansonsten haben ***div***-Elemente keine weiteren Eigenschaften. In erster Linie dienen die mithilfe der **<*div*>**-Tags strukturierten Bereiche als Schnittstelle zum späteren Layout innerhalb der CSS-Datei. Deshalb hatten unsere soeben gemachten Erweiterungen im Quelltext keinerlei Auswirkungen auf die Darstellung, was Sie selbst nachvollziehen können, wenn sie die Datei to-do2.html in Ihrem Browser öffnen.

Äquivalent zum **<*div*>**-Tag gibt es das **<*span*>**-Tag, das benutzt wird, wenn einer Gruppe von HTML-Elementen **Inlinestile** – d. h. Stile **ohne Zeilenumbruch** – zugewiesen werden sollen. Beispiele für die konkrete Anwendung finden Sie auf Seite 84 (Abbildung 28) und im To-Do 31.

Halten wir also fest: **<*div*>** und **<*span*>** sind bedeutungsleere, rechteckige Container, die zur Kennzeichnung von Block- oder Textabschnitten genutzt werden. Blockele-

Kapitel 3:
CSS - Inhalte gestalten

Machen wir uns nun daran, die mit HTML angelegten und strukturierten Inhalte mithilfe von CSS zu gestalten. Zur Erinnerung: Im HTML-Dokument haben wir die Bereichseinteilungen in Form rechteckiger Kästen (Containern, Boxen, Bereichen, Kisten) grundsätzlich angelegt. Ab jetzt geht es um das Design dieser Kästen. Die Gestaltungsmöglichkeiten, die uns CSS dabei bietet, lassen sich in drei Bereiche unterteilen:

1. Den **Inhalt** der Kästen (Farben und Schriften) gestalten (ab S. 41).
2. Die **Abstände** der Kästen und deren Rahmenlinien gestalten (ab S. 51).
3. Die **Position** der Kästen definieren (ab S. 81).

In genau dieser Reihenfolge werden wir nachfolgend die einzelnen CSS-Bestandteile kennenlernen.

Der allgemeine Aufbau einer CSS-Regel

Genauso wie HTML hat auch CSS seine eigene Syntax. Dreh- und Angelpunkt bei den Ausführungen bezüglich HTML waren die Elemente. Bei CSS sind es die sogenannten **Regeln** (häufig auch als Stilregeln oder Style bezeichnet). Der schematische Aufbau einer CSS-Regel sieht folgendermaßen aus:

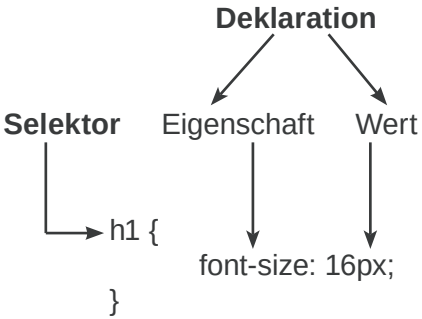

Abb. 10: Schematischer Aufbau einer CSS-Regel

Eine Regel besteht also in der Grundform aus

- einem Selektor
- und einer Deklaration bestehend aus Eigenschaft **und** Wert.

Selektor: Der Selektor bestimmt (selektiert), auf welche HTML-Elemente (rechteckige Kästen) sich die Regel bezieht.

Deklaration: Die Deklarationen müssen in geschweiften Klammern eingeschlossen werden. Der erste Teil benennt die zu gestaltende Eigenschaft (z. B. Schriftgröße oder Farbe) gefolgt von einem Doppelpunkt. Der zweite Teil bestimmt den Wert, den die Eigenschaft annehmen soll. Hinter jedem Eigenschaft-Wert-Paar steht ein Semikolon.

Schauen wir uns nun ein konkretes Beispiel für eine Regel an:

```
h1 {
font-size: 16px;
color: red;
}
```

Der Selektor ***h1*** definiert, dass diese Stilregel auf alle ***h1***-Überschriften des Dokumentes angewendet werden soll. Korrespondierend wurden zwei Deklarationen angelegt: Als Erstes definiert die Eigenschaft ***font-size*** den Schriftgröße aller Absätze auf den Wert 16 Pixel. Die zweite Eigenschaft ***color*** bewirkt, dass alle ***h1***-Überschriften in roter Schrift (= Wert) dargestellt werden.

Selektoren

Soeben haben wir einen Selektor kennengelernt, der sich auf ein bestimmtes HTML-Element (***h1***) bezog. Verwendet man den Selektor in dieser Form, hat man das Problem, dass sich die Deklaration **auf ausnahmslos alle im Selektor genannten HTML-Elemente des gesamten Webprojektes bezieht**. Wollen Sie z. B. auf einer Unterseite die ***h1***-Überschriften nicht in Rot mit ***16px*** darstellen, ist das mit dem soeben verwendeten Selektor nicht möglich. Um dieses Problem zu umgehen, gibt es zwei spezielle Selektoren, die nur in einem bestimmten, zuvor von ihnen definierten Bereich zur Anwendung kommen: ID-Selektoren und Klassen-Selektoren. Beide Selektoren bieten die Möglichkeit, Stilregeln für bestimmte HTML-Elemente zu definieren. ID-Selektoren sind **einmalig** und dürfen innerhalb eines Webprojektes nur ein einziges Mal vorkommen. Klassen-Selektoren können hingegen innerhalb eines Webprojektes **mehrfach** verwendet werden.

Kapitel 4:
Den Inhalt der Kästen (Farben und Schriften) gestalten

Farben ins Spiel bringen

Nachfolgend werden wir uns damit beschäftigen, wie man Text, Hintergründe und andere Elemente mit Farbe ausstattet.

Vordergrund- und Hintergrundfarbe definieren

Wir beginnen nun mit der farblichen Gestaltung unserer Website. Als ersten Schritt werden wir den in HTML angelegten Kästen Farben zuweisen. Hierfür wird die Eigenschaft ***background-color*** verwendet. Die Definition (also das Anlegen und Strukturieren) der Bereiche haben wir ja bereits im HTML-Quelltext mithilfe der Elemente ****** und *** *** durchgeführt. Wollen wir jetzt beispielsweise den Kasten, den wir für die Kopfzeile angelegt haben, gelb einfärben, müssen wir folgende Stilregel anlegen:

```
#kopfzeile {
  background-color: yellow;
}
```

Damit ist Gelb (Wert) als Hintergrundfarbe (Eigenschaft) für den Kasten kopfzeile (Selektor) definiert. Nun gibt es bei CSS die Besonderheit, dass man eine Hintergrundfarbe stets zusammen mit der Vordergrundfarbe festlegt. Mit Vordergrundfarbe ist die Schriftfarbe des Textes gemeint. Der Grund hierfür ist, dass jeder Benutzer in seinem Browser Voreinstellungen unter anderem bezüglich der Vordergrund- und Hintergrundfarbe tätigen kann. Hat ein Betrachter Ihrer Website in seinem Browser z. B. eine Vordergrundfarbe eingestellt, die in Ton und Helligkeit der von Ihnen definierten Hintergrundfarbe ähnelt, erscheint Ihre Seite ohne Inhalt. Mehr zu dieser Problematik erfahren Sie im Kapitel über die Kaskade von CSS (siehe Seite 127). Um die Vordergrundfarbe zu definieren, verwenden wir die Eigenschaft ***color***. Möchten wir für den Text die Farbe schwarz definieren, sieht unsere Stilregel komplettiert wie folgt aus:

Kapitel 5:
Die Abstände der Kästen und deren Rahmenlinien gestalten

Das Box-Modell

Jedes Element in HTML erzeugt einen rechteckigen Kasten (Box), in dem der darzustellende Inhalt angezeigt wird. Prinzipiell würde nun jeder Kasten an dem nachfolgenden Kasten „kleben". Vielleicht möchte man aber den Abstand zum nachfolgenden Kasten (Außenabstand) verändern. Oder man möchte den Kasten einrahmen. Und vielleicht möchte man zusätzlich bestimmen, dass diese Rahmenlinie vom eigentlichen Inhalt einen gewissen Abstand haben soll (Innenabstand). Alles kein Problem, CSS bietet eine Reihe von Eigenschaften und Werten, mit denen sich auf den Außenabstand, die Rahmenlinie und den Innabstand Einfluss nehmen lässt. Dieses Zusammenspiel wird als „Box-Modell" bezeichnet. Schauen sie sich einmal nachfolgende Grafik an:

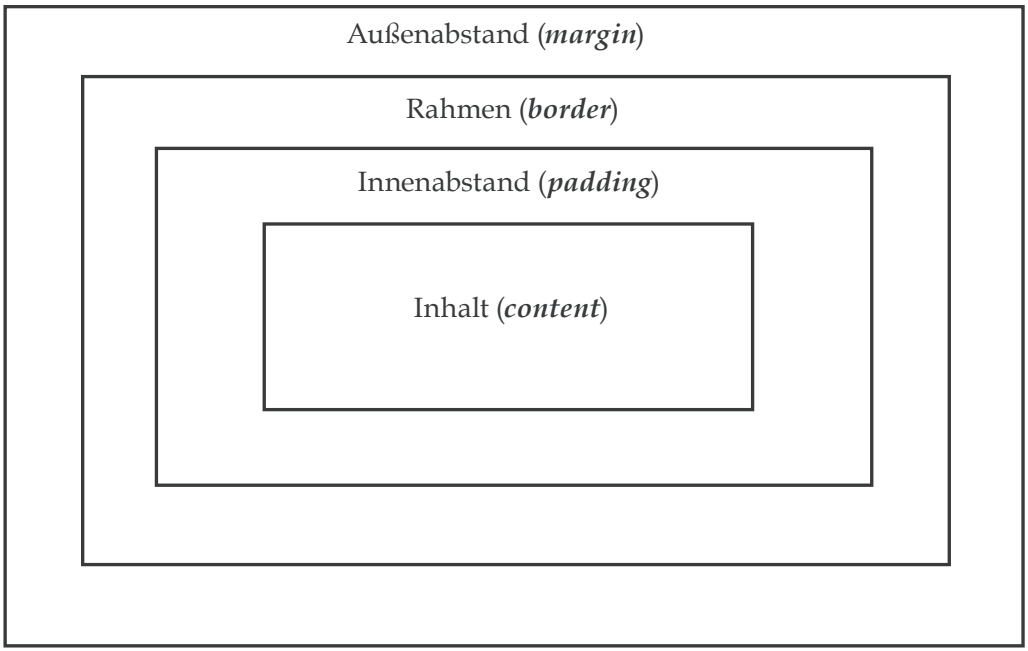

Abb. 11: Das Box-Modell beschreibt die verschiedenen Bestandteile einer rechteckigen Box

Die Grafik im Ganzen zeigt einen rechteckigen Kasten, wie er als HTML-Element generiert wird. Sie zeigt aber auch die einzelnen Bestandteile im Inneren der Box. Diese Elemente bestehen aus:

1. dem eigentlichen Inhalt wie z. B. Text und Grafik (***content***)
2. dem Innenabstand von Inhalt zum Rahmen (***padding***)
3. dem Rahmen (***border***)
4. dem Außenabstand um den Inhalt herum (***margin***)

Die einzelnen Eigenschaften, auf die durch CSS-Befehle Einfluss genommen werden kann, finden Sie in der nachfolgenden Grafik. Anschließend werden wir die Eigenschaften im Detail besprechen.

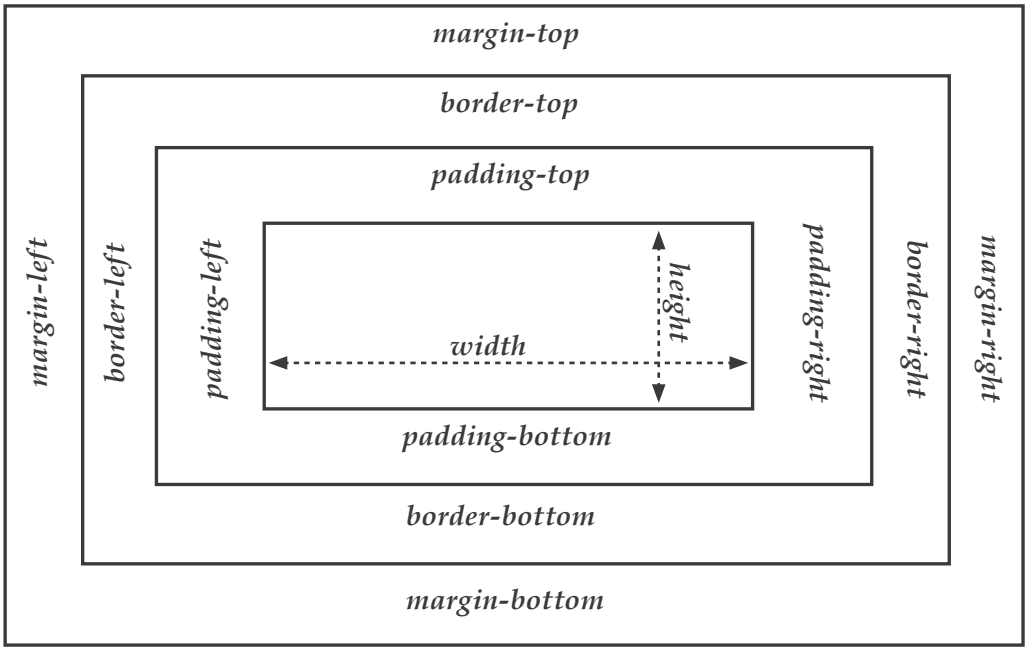

Abb. 12: Das Box-Modell mit den möglichen Eigenschaften

Inhalt (***content***)

Die Größe des innersten Bereichs kann von unterschiedlichen Faktoren abhängen: Wenn keine Angaben zur Breite (***width***) und Höhe (***height***) gemacht sind, bestimmt der Inhalt selbst die Größe. Ferner können auch Eigenschaften wie Vorder- und Hintergrundfarben zugewiesen werden. Schauen Sie sich zur Verdeutlichung nachfol-

Kapitel 6:
Positionierung der Kästen

Bereits im HTML-Dokument haben wir Festlegungen über die grundsätzliche Struktur der Seite getroffen. Mithilfe von **<*div*>** und **<*span*>** (Block- und Inline-Elemente) haben wir dort „generische Behälter“ für Inhalte aller Art angelegt (siehe Seite 17). Bisher haben wir in die Positionierung der rechteckigen Container, die im HTML-Dokument durch die Reihenfolge vorgegeben ist, nicht eingegriffen. Eine Änderung wäre bei unserer einspaltigen Beispielseite auch nicht wirklich sinnvoll gewesen. Bei mehrspaltigen Dokumenten stellt sich diese Frage hingegen schon bei der grundsätzlichen Anordnung: Soll z. B. die Box mit den Inhalten rechts, links oder immer in der Mitte liegen? Soll die Navigationsleiste auf der linken oder der rechten Seite zu sehen sein? Doch auch bei unserem Einspalter werden wir gleich innerhalb der Kopfzeile Bild und Text nach unseren Wünschen anordnen und damit die vorgegebene Reihenfolge verändern. Man kommt also nicht herum, sich ein Grundverständnis bezüglich der Positionierung der rechteckigen Container anzueignen, wenn man CSS ernsthaft anwenden möchte.

Die Eigenschaft *position*

Veränderungen bezüglich der Anordnung der Elemente erfolgen mithilfe der Eigenschaft ***position***. Sie definiert, wo einzelne Bereiche oder Elemente auf einer Seite platziert werden sollen. Eine Positionierung von Elementen in CSS kann statisch (***static***), absolut (***absolute***), relativ (***relative***) oder fixiert (***fixed***) sein.

- ***static***: Der Standardwert der Eigenschaft ***position***, d. h. er gilt für jedes Element einer Seite, bis etwas anderes definiert wird. Die Elemente werden in der Reihenfolge angezeigt, in der sie im HTML-Code vorkommen.
- ***relative***: Hiermit wird die Position relativ zu der Position angegeben, die das Element im normalen Textfluss gehabt hätte. Es wird also von der Position ausgegangen, an der es normalerweise wäre, wenn es die Positionierungsart ***static*** hätte, und von dort wird es um die angegebenen Werte verschoben.
- ***absolute***: Das Element wird durch diese Positionierungsart völlig unabhängig von den restlichen Elementen auf der Seite.
- ***fixed***: Entspricht der Angabe ***absolute***, nur bleibt das Element beim Scrollen stehen.

Kapitel 7:
Mehrspaltige Layouts

Bei unserer bisherigen Beispielseite handelt es sich um ein sogenanntes einspaltiges Layout, da auf horizontaler Ebene lediglich eine Hauptspalte vorhanden ist. Als nächsten Schritt werden wir mehrspaltige Layouts erstellen, bei der wir die Navigation nicht mehr über, sondern neben dem Inhaltsbereich anordnen.

Bei der Realisation steht man als Webdesigner vor der Herausforderung, dass CSS leider keine Möglichkeit vorsieht, mit der sich ein mehrspaltiges Layout direkt erzeugen lässt. Es gibt schlichtweg keine CSS-Eigenschaft, die direkt für die Anordnung in Spalten sorgt oder zumindest einem Element das typische Verhalten einer Spalte zuweist. Dessen ungeachtet gibt es Ersatzlösungen, mit denen sich trotzdem funktionsfähige, mehrspaltige CSS-Layouts erzeugen lassen. Dabei gibt es nicht die eine „richtige“ Lösung. Im Laufe der Zeit haben sich einige Workarounds durchgesetzt, die nachfolgend mit ihren jeweiligen Vor- und Nachteilen erklärt werden.

Exkurs: Höhen und Breiten – absolut und relativ

Vorweg wollen wir uns einmal anschauen, welche grundsätzlichen Gestaltungs- und Kontrollmöglichkeiten uns CSS in Bezug auf die Ausdehnung der Container bietet. Im To-Do 15 hatten wir bereits dem Wrapper eine relative (prozentuale) Breitenangabe zugewiesen. Die Auswirkungen dieser Definition sieht man sehr schön, wenn das Browserfenster in der Breite verkleinert wird.

Kapitel 8:
Kaskade, Vererbung und Standardwerte

Nachfolgend werden wir detailliert beleuchten, nach welchen Regeln ein Browser einen Quelltext einliest und interpretiert. Diese sog. Kaskadierungs- und Vererbungsregeln sind für den Anfänger schwer verdauliche Kost. Versuchen Sie deshalb, das nachfolgende Kapitel konzentriert durchzuarbeiten, und seien Sie nicht frustriert, wenn Sie nicht alle Zusammenhänge auf Anhieb verstehen.

Der Job des Browsers: parsen und interpretieren

Ein Browser, der von einem Webserver einen HTML-Quelltext einliest (parst), erstellt zunächst einen virtuellen Dokumentenbaum, der den hierarchischen Aufbau beschreibt. Die Fachbezeichnung hierfür lautet Document Object Model, abgekürzt DOM. Die nachfolgende Abbildung zeigt, wie ein solches Modell für einen einfachen Dokumentenbaum aussehen kann:

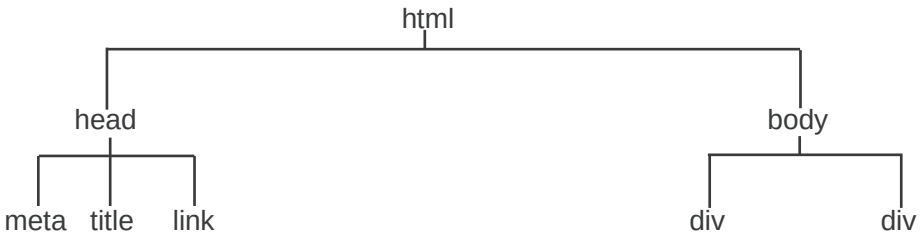

Abb. 47: Grafische Darstellung eines HTML-Dokumentenbaums

Das oberste Element einer jeden Webseite ist immer ***html***. Von ***html*** gehen zwei Elemente ab, nämlich ***head*** und ***body***. Bildlich gesprochen sind ***head*** und ***body*** Kinder von ***html*** und somit Geschwister. ***body*** wiederum hat weitere Nachfahren, die man wiederum als Eltern- und Kind-Elemente zuordnen kann.

Aus dem HTML-Quelltext kennt der Browser jetzt den hierarchischen Aufbau der Seite. Er weiß aber noch nicht, wie er die Elemente konkret darstellen soll. Deshalb versucht er als nächsten Schritt, jedem Element eine Eigenschaft und einen Wert (z. B. für die Eigenschaft ***border-style*** den Wert ***solid***) zuweisen. Hierzu untersucht der Browser die CSS-Stilregeln. Nun kann es passieren, dass im Quelltext mehrere De-

Kapitel 9:
Wie teste ich meinen Quellcode?

Die Grammatik für HTML und CSS wird vom W3C verbindlich vorgegeben. Für Webdesigner gibt es Validatoren, mit deren Hilfe man überprüfen kann, ob der geschriebene Seitenquellcode der W3C-Standardkonformität entspricht (siehe Seite 31 und 124). Das ist die erste Voraussetzung, damit der abrufende Quelltext von allen Browsern richtig verstanden und dargestellt werden kann. Hierauf hat man als Webdesigner direkten Einfluss.

Als zweite Voraussetzung muss der abrufende Browser den Quellcode auch tatsächlich entsprechend den Standards interpretieren. Glücklicherweise funktioniert die Interpretation bei den aktuellen Browserversionen in den allermeisten Konstellationen sehr gut, aber leider nicht zu einhundert Prozent. Das liegt sowohl an kleinen Ungenauigkeiten der Standards, die Interpretationsspielraum der Browserhersteller zulassen, wie auch daran, dass das jeweilige Betriebsystem die Darstellung im Browser beeinflussen kann. Auf die browserspezifische Umsetzung der Standards hat man als Webdesigner keinen direkten Einfluss.

Ziel eines Webdesigners sollte es sein, dass die von ihm erstellten Seiten immer korrekt dargestellt werden, unabhängig davon, welcher Browser und welches Betriebssystem vom Betrachter verwendet werden. Die einzige Möglichkeit, diesen Anspruch umzusetzen, bedeutet: testen, testen und noch mal testen. Bei kommerziellen Webprojekten ist es durchaus üblich, dass bis zu 30 Prozent der veranschlagten Gesamtzeit auf das Testen entfällt.

Zum Testen bieten sich prinzipiell drei Möglichkeiten an:

1. Auf dem eigenen Rechner diverse Betriebssysteme und Browserversionen installieren
2. Entsprechende Webdienste nutzen
3. Betriebssysteme über virtuelle Maschinen emulieren und dort die entsprechenden Browser installieren

Kapitel 10:
Software für Webentwickler

Notwendige Software: der CSS-Editor

Zum erstellen von HTML- und CSS-Quellcode ist prinzipiell der im Betriebssystem enthaltene Texteditor (Notepad) ausreichend, wenn auch nicht empfehlenswert. Ungeeignet ist ein Texteditor vor allem deshalb, weil er keine Syntax-Hervorhebung anbietet, d. h. die HTML- und CSS-Befehle nicht farblich hervorhebt und anordnet. Durch die optische Eintönigkeit wird der Quelltext extrem schwer lesbar, unübersichtlich und damit fehleranfällig. Gute, kombinierte HTML- und CSS-Editoren bieten aber zusätzliche Funktionen, die das Arbeiten am Quelltext erheblich erleichtern. Folgende Features sind sinnvoll:

- Anzeige der Änderungen live in einem Vorschaufenster
- Vorgabe der CSS-Eigenschaften mit ihren erlaubten Werten für alle Elemente, flankiert von einer Autovervollständigung (diese Funktion ist insbesondere für Anfänger hilfreich!)
- Integrierte Validatoren
- Eingebaute HTML- / CSS-Referenz
- Projekt-Manager

Leider ist mir kein kostenloses Programm bekannt, das diese Funktionen kombiniert anbietet. Im Leserbereich habe ich deshalb eine Auswahl an kostenpflichtigen Programmen zusammengestellt. Testen Sie insbesondere den Bedienkomfort, bevor Sie sich für den Kauf eines Editors entscheiden. Anschauen sollten Sie sich nach meiner subjektiven Meinung auf jeden Fall die Software „HTMLPad“, die trotz des Namens auch CSS vollständig unterstützt. HTMLPad erfüllt alle oben geforderten Funktionen, ist intuitiv zu bedienen und vom Kaufpreis angemessen.

Sinnvolle Software für Webentwickler

Zur Überprüfung und Optimierung der eigenen Quelltexte wie auch zur Analyse fremder Websites gibt es Tausende Miniprogramme und Browser-Plugins. Egal ob Messwerkzeuge, Lupen oder Farbwähler – viele dieser Programme sind tatsächlich sinnvoll und erleichtern die Arbeit erheblich. Früher musste man zu diesem Zweck unzählige Einzelprogramme installieren. Mittlerweile gibt es zwei Programme, die

(fast) alles Notwendige und Sinnvolle zur Analyse und Diagnose vereinen: „Firebug“ und „Web Developer“. Beide Programme sind als Firefox-Plugins verfügbar, d. h., sie lassen sich unkompliziert über die Add-On-Schnittstelle des Browsers integrieren. Gerade HTML- und CSS-Einsteiger sollten sich unbedingt mit Firebug beschäftigen, da man mit diesem Programm sehr einfach den Designaufbau fremder Websites studieren kann.

Firebug

Firebug wird uns bei folgenden drei Aufgaben unterstützen:

- HTML/CSS identifizieren,
- analysieren,
- live editieren.

Weitere Features von Firebug (insbesondere Java-Debugging) werden hier nicht besprochen, da sie nicht im thematischen Zusammenhang mit dieser CSS-Einführung stehen.

Nach der Installation im Firefox-Browser kann Firebug über ein eigenes Käfer-Icon in der Statusleiste oder über die F12-Taste auf- und zugeklappt werden. Im eingeblendeten Firebug-Fenster sind auf der linken Seite mehrere Tabs verfügbar. Für unsere Zwecke interessant ist der Tab „HTML“, der das Fenster in zwei Bereiche einteilt: links die HTML-Grundstruktur, rechts die zugehörigen CSS-Elemente (achten Sie darauf, dass rechts oben der Tab „Styles“ aktiviert ist).

Abb. 50: Firebug in Aktion

HTML/CSS identifizieren

Öffnen Sie nun die Datei index.html aus dem To-Do 53 und aktivieren Sie Firebug (Grundlage ist eine leicht modifizierte Version der Datei aus dem To-Do 31).

Auf der linken Seite finden Sie im unteren Bereich die HTML-Grundstruktur. Die einzelnen Elemente können Sie durch Klick auf das Pluszeichen öffnen. Klicken Sie auf das Pluszeichen vor ***<body id="startseite">*** und dann vor *** ***.

Stichwortverzeichnis