HTML5 und CSS3

Der Meisterkurs

ELIZABETH CASTRO BRUCE HYSLOP

Markt+Technik

Auf der Website zum Buch finden Sie alle Beispiele und weitere nützliche Ressourcen

ALWAYS LEARNING

PEARSON

Elizabeth Castro, Bruce Hyslop

HTML5 und CSS3

Der Meisterkurs

Aus dem Englischen von Jürgen Dubau

Für die Familie

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über <http://dnb.dnb.de> abrufbar.

Autorisierte Übersetzung der englischsprachigen Originalausgabe mit dem Titel Visual Quickstart Guide, HTML5 and CSS3, von Elizabeth Castro und Bruce Hyslop, erschienen bei Peachpit Press, ein Imprint von Pearson Education, Inc., Copyright 2012

10 9 8 7 6 5 4 3 2 1

14 13 12

ISBN 978-3-8272-4762-9 (Buch); 978-3-86325-566-4 (PDF); 978-3-86325-087-4 (ePUB)

ein Imprint der Pearson Deutschland GmbH,
Martin-Kollar-Straße 10-12, D-81829 München/Germany

Covergestaltung: Marco Lindenbeck, webwo GmbH, mlindenbeck@webwo.de
Lektorat: Dorothea Krist, dkrist@pearson.de
Übersetzung: Jürgen Dubau, Freiburg/Elbe
Fachlektorat: Dr. Claudia Nölker, Bielefeld
Korrektorat: Petra Kienle
Herstellung: Monika Weiher, mweiher@pearson.de
Satz: text&form GbR, Fürstenfeldbruck
Druck und Verarbeitung: GRAFOS S. A., Barcelona
Printed in Spain

Danksagungen

Die Danksagungen stellen einen beim Schreiben eines Buchs mit vor die größte Herausforderung, weil man sicher sein will, dass alle, die es verdienen, entsprechend ihrer Mitarbeit gewürdigt werden. Dieses Buch konnte nur entstehen, weil so viele Menschen mich unterstützt, unermüdlich daran mitgearbeitet und ihre gute Laune nicht verloren haben. Ich hoffe, allen gerecht zu werden, und ebenso erhoffe ich mir von Ihnen, dass Sie mir meinen Wunsch nachsehen, mich umfassend bedanken zu wollen.

Mein aufrichtiger Dank richtet sich an die folgenden Personen:

Nancy Aldrich-Ruenzel und Nancy Davis, die mir diese Auflage eines Buchs anvertraut haben, das für Peachpit schon viele Jahre lang sehr wichtig gewesen ist.

Cliff Colby, der mich weiterempfahl und alles ermöglichte, für sein Vertrauen in mich und seine Geduld, Flexibilität und Anleitung – und für unsere zahllosen Gespräche und dass wir so viel gelacht haben.

Robyn Thomas für ihre riesengroßen Mühen, uns alle in der Spur zu halten. Ihr ermutigender Zuspruch sorgte regelmäßig für neuen Schwung.

Michael Bester für all das genau richtige Feedback und die Vorschläge. Er fing technische Fehler ab, deckte Auslassungen auf und half uns dabei, den Lesern die richtige Botschaft zu vermitteln. Es war wirklich eine Freude, zusammen mit ihm wieder an einem Buch zu arbeiten.

Chris Casciano im selben Zusammenhang für all seine technischen Expertisen, Vorschläge und das unverzichtbare Feedback. Ich weiß es wirklich sehr zu schätzen, dass du in den letzten Wochen vor Schluss zu uns gestoßen bist – wirklich ein Glück, dich dabei zu haben.

Cory Borman, weil er die Beaufsichtigung der Produktion dieses Buchs so exzellent überwachte und notfalls auch Diagramme erstellte – danke für deine wunderbar humorvolle Art.

Scout Festa für die sorgfältige Grammatik- und Rechtschreibkorrektur.

David Van Ness für seine große Sorgfalt beim Layout der Seiten und den Blick für die Details.

Nolan Hester für sein Expertenwissen, das er bei der Prüfung der layouteten Seiten einbrachte.

Valerie Haynes Perry für die Erstellung eines sinnvollen Indexes.

Die zahlreichen Kollegen und Kolleginnen aus den Abteilungen Marketing, Sales und anderen bei Peachpit, weil sie hinter den Kulissen für den Erfolg des Buchs sorgten.

Meine Familie und Freunde, die sich stetig nach meinem Vorankommen erkundigten und immer wieder mal für willkommene Pausen vom Schreiben sorgten. Insbesondere gilt mein Dank all jenen Freunden, die es wahrscheinlich leid waren, von mir immer wieder zu hören zu bekommen, ich hätte keine Zeit für ein Treffen, mich aber trotzdem immer wieder eingeladen haben.

Robert Reinhard wie stets, der mich überhaupt zum Schreiben von Büchern gebracht hat, und für seine Anleitung, dieses Buch in Angriff zu nehmen.

Die Web-Community für all ihre Innovationen und dafür, dass ihr euer Wissen weitergebt, damit viele davon profitieren (viel von euch habe ich im Verlauf dieses Buchs zitiert).

An Sie als Leser und als Leserin für Ihr Interesse, HTML und CSS zu erlernen und sich für dieses Buch zu entscheiden: Ich weiß, dass es auch noch viele andere gibt, die Sie hätten wählen können. Ich hoffe, dass Ihnen dieses Buch eine hilfreiche Unterstützung ist.

Vielen, vielen Dank auch an die folgenden Autoren, die an diesem Buch mitgearbeitet haben. Unsere Leser halten durch euren Einsatz nun ein wertvolleres Buch in Händen, und dafür bin ich dankbar.

Ich möchte mich bei Erik Vorhes entschuldigen, dass wir es nicht geschafft haben, die Anhänge A und B ins Buch aufzunehmen. Die Leser, die auf der Website für dieses Buch darauf stoßen, werden deine Arbeit sicher zu schätzen wissen.

In alphabetischer Reihenfolge handelt es sich bei den Mitautoren um folgende:

Scott Boms (Kapitel 14)

Scott ist ein mit Preisen ausgezeichneter Designer, Autor und Redner, der in seinen über 15 Jahren der Arbeit im Web mit solchen Organisationen wie PayPal, HSBC, Hyundai, DHL, XM Radio, dem Magazin *Toronto Life* und Masterfile zusammengearbeitet hat. Wenn er nicht gerade am Computer sitzt, macht er vielleicht Polaroids, spielt Schlagzeug bei seiner Band George oder genießt die Zeit mit seiner wunderbaren Frau und den beiden Kindern. Auf Twitter finden Sie ihn unter @scottboms.

Ian Devlin (Kapitel 17)

Ian kommt aus Irland und ist Webentwickler, Blogger und Autor. Ihm macht Programmieren und das Schreiben über neu entstehende Webtechnologien wie HTML5 und CSS3 sehr viel Spaß. Neben der Frontend-Entwicklung erstellt Ian auch Lösungen mit Backend-Technologien wie .NET und PHP. Kürzlich hat er das Buch *HTML5 Multimedia: Develop and Design* (Peachpit Press, 2011) veröffentlicht.

Seth Lemoine (Kapitel 5 und 16)

Seth ist Softwareentwickler und Lehrer aus Atlanta. Seit über zehn Jahren arbeitet er an herausfordernden Projekten und reizt unter Einsatz solcher Technologien wie HTML, JavaScript und CSS bis zu Objective-C und Ruby aus, was möglich ist. Egal, ob es darum geht, innovative Wege zu finden, seinen Schülern HTML5 und CSS beizubringen oder in seinem Outdoor-Wok ein Rezept aus Sezuchan zu perfektionieren: Kreativität ist seine Leidenschaft.

Erik Vorhes (Anhang A und B, erhältlich auf der Website des Buchs)

Erik Vorhes erstellt Sachen im Web mit VSA Partners und ist Redaktionsleiter bei Typedia (*http://typedia.com/*). Er lebt und arbeitet in Chicago.

Brian Warren (Kapitel 13)

Brian ist Senior Designer bei Happy Cog in Philadelphia. Wenn er nicht gerade schreibt oder designt, verbringt er seine Zeit damit, mit seiner wunderbaren Familie zu spielen, Musik zu hören und Bier zu brauen. Er bloggt zeitweise unter *http://begoodnotbad.com*.

Und schließlich möchte ich gerne mein ganz besonderes Dankeschön an Elizabeth Castro aussprechen. Sie hat vor über 15 Jahren die erste Ausgabe dieses Buchs verfasst und mit jeder folgenden Neuauflage ihr Publikum weiter versorgt und auf dem Laufenden gehalten. Ihre Art zu lehren hat in diesen Jahren bei buchstäblich Hunderttausenden von Lesern und Leserinnen für guten Nachhall gesorgt. Ich bin außerordentlich dankbar für die Gelegenheit, an diesem Buch mitarbeiten zu dürfen, und habe sehr aufmerksam versucht, den Vorgaben gerecht zu werden, als ich an dieser Auflage arbeitete.

—Bruce

3

Die grundlegende HTML-Struktur

In diesem Kapitel geht es um die HTML-Elemente, mit denen Sie Ihrem Dokument Grundlage und Struktur verleihen. Damit sind die Gliederung und die primären semantischen Container für Ihre Inhalte gemeint.

Sie lernen Folgendes:

- Eine neue Webseite beginnen
- Die Gliederung eines HTML5-Dokuments
- Die Elemente `h1`-`h6`, `hgroup`, `header`, `nav`, `article`, `section`, `aside`, `footer`, und `div` (von denen die meisten in HTML5 neu hinzugekommen sind)
- Wie die ARIA-Attribute für `role` die Accessibility Ihrer Seite verbessern
- Wie man eine `class` oder `id` auf Elemente anwendet
- Wie man das Attribut `title` auf Elemente anwendet
- Wie man Kommentare in den Code einfügt

Wenn man eine klare und einheitliche Struktur schafft, werden die Webseiten nicht nur unter semantischen Gesichtspunkten gut vorbereitet, sondern es erleichtert auch, das Dokument anhand von Cascading Style Sheets (CSS) zu formatieren (darum geht es ab Kapitel 7).

Übersicht

Die erste Webseite beginnen

Ihre HTML-Dokumente sollten als Mindestvoraussetzungen die folgenden Komponenten enthalten – siehe A:

- Den DOCTYPE
- Das `html`-Element (mit dem Attribut , das optional, aber ratsam ist)
- Das `head`-Element
- Die Zeichenkodierung in einem `meta`-Element
- Das `title`-Element (dessen Inhalte Sie gleich einfügen werden)
- Das `body`-Element

Dies ist nun die HTML-Entsprechung eines leeren Blatts, da im `body`-Element ja noch kein Inhalt vorhanden ist B.

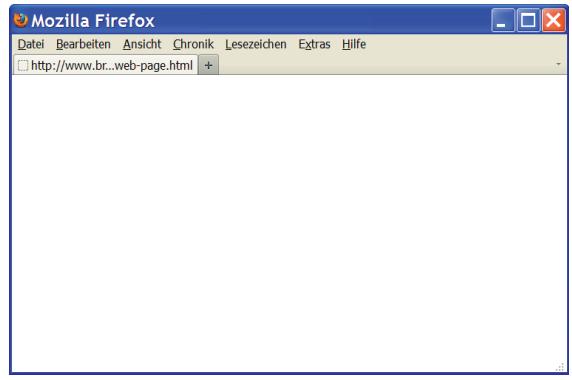
Bevor Sie nun weitere Inhalte einfügen, müssen Sie die Grundlage Ihrer Seite vorbereiten.

Eine HTML5-Seite beginnen

1. Tippen Sie `<!DOCTYPE html>`, um Ihre Seite als HTML5-Dokument zu deklarieren (im Kasten „Der verbesserte DOCTYPE von HTML5“ lesen Sie mit den früheren Versionen von HTML zusammenhängende Informationen).
2. Tippen Sie `<html lang="`*sprach-code*`">`, um den eigentlichen HTML-Bereich Ihres Dokuments zu beginnen, wobei `<html lang="`*sprach-code*`">` die Sprachkodierung ist, die zur Standardsprache Ihrer Seiteninhalte passt. Zum Beispiel steht `<html lang="de">` für Deutsch oder `<html lang="fr">` für Französisch. Weitere Sprachkodierungen finden Sie unter *www.bruceontheloose.com/references/language-codes.html*.

A *Dies ist die Grundlage aller HTML-Seiten. Die Einrückungen sind nebensächlich, aber die Struktur ist ganz wesentlich. In diesem Beispiel wird die Standardsprache (anhand des Attributs `lang`) auf `de` für Deutsch gesetzt. Die Zeichenkodierung ist UTF-8.*

```
<!DOCTYPE html>
<html lang="de">
<head>
   <meta charset="UTF-8" />
   <title></title>
</head>
<body>


</body>
</html>
```



B *So sieht diese einfache HTML-Grundlage in Firefox aus. Wie Sie sehen, sehen Sie nichts! Doch gleich werden wir Inhalte einbringen.*

Wenn Sie Kapitel 1 noch nicht gelesen haben, rate ich nachdrücklich dazu, vor dem Weiterlesen sich erst noch dieses Kapitel vorzunehmen. Es zeigt eine einfache HTML-Seite und erklärt einige Grundlagen. Weil dies nun Ihr erster Blick auf eine Webseite sein wird, wiederhole ich einige (aber nicht alle) Infos hier noch einmal und gehe davon aus, dass Ihnen der Rest so vertraut ist, um auf diesen Konzepten aufbauen zu können.

3. Tippen Sie **<head>**, um den Dokumentkopf der Seite zu beginnen.
4. Tippen Sie **<meta charset="UTF-8" />**, um für die Zeichenkodierung Ihres Dokuments UTF-8 festzulegen. Sie können das auf Wunsch auch im HTML als `utf-8` schreiben. Außerdem sind Leerzeichen und Schrägstrich optional und somit funktioniert `<meta charset="UTF-8">` genauso gut. (Andere Zeichenkodierungen als UTF-8 sind auch gültig, aber UTF-8 ist am vielseitigsten. Also brauchen Sie nur selten davon abzuweichen.)
5. Tippen Sie **<title></title>**. Darin wird der Titel der Seite stehen. Sie werden den Text dafür im Abschnitt „Einen Titel erstellen" einfügen.
6. Tippen Sie **</head>**, um den Dokumentkopf der Seite zu beenden.
7. Tippen Sie **<body>**, um mit dem Textkörper der Seite zu beginnen. Hier kommen dann schließlich und endlich Ihre Inhalte hinein.
8. Lassen Sie ein paar Zeilen leer, um darin (anhand dieses Buchs) die Seiteninhalte zu erstellen.
9. Tippen Sie **</body>**, um den Textkörper zu beenden.
10. Tippen Sie **</html>**, um die Seite abzuschließen.

Das sind ganz schön viele Schritte, aber weil diese bei all Ihren Seiten auf diese Weise nötig sind, nehmen Sie am besten eine HTML-Seite und speichern Sie als Vorlage ab, um sich die viele Tipparbeit zu ersparen. Tatsächlich erlauben es die meisten Code-Editoren, dass man den Startcode für jede neue Seite abspeichert, was alles noch mehr erleichtert. Falls Sie in Ihrem Editor keine Optionen wie Einstellungen oder Präferenzen finden, schauen Sie in der Hilfe-Sektion nach.

Die beiden Bereiche einer Seite: Kopf und Körper

Als kurze Erinnerung des in Kapitel 1 Gelernten hier der Hinweis, dass HTML-Seiten in zwei Bereiche geteilt werden: `head` und `body` (A). Der DOCTYPE, der jede Seite einleitet, ist eine Art Präambel.

Im `head` des Dokuments definieren Sie den Seitentitel, schreiben über Ihre Seite Infos für Suchmaschinen wie Google, laden Stylesheets und gelegentlich auch JavaScript-Dateien (obwohl es aus Gründen der Performance meist zu bevorzugen ist, JavaScript direkt vor dem End-Tag `</body>` unten auf der Seite zu laden). Beispiele dafür werden Sie im Verlauf dieses Buchs kennenlernen. Der Besucher sieht außer dem `title`, um den es gleich gehen wird, erst einmal nichts vom Inhalt des Kopfabschnitts `head`, wenn er Ihre Seite besucht.

Das `body`-Element schließt den Inhalt der Seite ein, z.B. Text, Bilder, Formulare, Audio- und Videodateien sowie andere interaktive Inhalte. Mit anderen Worten sind es die Sachen, die Ihre Besucher sehen können. In diesem Buch gibt es mehrere Kapitel, die sich speziell den auf Inhalte bezogenen HTML-Elementen widmen, und auf ein paar dieser Elemente wird gleich noch hingewiesen.

TIPP **Der DOCTYPE von HTML5 gewährleistet, dass die Browser alles in einem verlässlichen Modus darstellen, und informiert die HTML-Validatoren, dass Ihr Code anhand der für HTML5 erlaubten Elemente und Syntax geprüft werden soll. Um HTML-Validatoren geht es in Kapitel 20.**

TIPP **Beim DOCTYPE von HTML5 wird nicht zwischen Groß- und Kleinschreibung unterschieden. Manche schreiben das z.B. so: `<!doctype html>`. Aber die andere Schreibweise ist eher üblich: `<!DOCTYPE html>` (A).**

TIPP **Das `html`-Element, das auf DOCTYPE folgt, muss alle anderen Elemente Ihrer Seite einschließen.**

TIPP **Achten Sie darauf, dass Ihr Code-Editor so konfiguriert ist, dass Dateien als UTF-8 gespeichert werden, damit sie zu dem über `<meta charset="UTF-8" />` gespeicherten Code passen (A) (falls Sie eine andere Kodierung nehmen wollen, dann müssen die Dateien natürlich darin gespeichert werden). Nicht alle Editoren speichern die Seiten standardmäßig als UTF-8, aber bei den meisten können Sie die Kodierung aus einem Menü oder einer Übersicht wählen (siehe „Speichern der Webseite“ in Kapitel 2). Wenn keine UTF-8-Einstellung vorgenommen wird, könnten statt eines erwünschten Buchstabens wie ein i mit Akzent oder ein n mit einer Tilde (˜) gelegentlich komische Zeichen in Ihrem Inhalt erscheinen.**

TIPP **Den Code im `head`-Element müssen Sie nicht einrücken (A). Allerdings hätten Sie dann den Vorteil, gleich sehen zu können, wo das `head`-Element anfängt, was darin steht und wo es endet. Es kommt nicht selten vor, dass `head` auf manchen Seiten sehr lang wird.**

Der verbesserte DOCTYPE von HTML5

Ach, um wie vieles einfacher ist es geworden, eine Webseite zu beginnen, seit wir HTML5 haben. Der DOCTYPE von HTML5 ist erfrischend kurz, vor allem verglichen mit den DOCTYPEs von damals.

Zu Zeiten von HTML 4 und XHTML 1.0 musste man sich noch unter den verschiedenen DOCTYPEs entscheiden und jeder stand sowohl für die HTML-Version als auch die Info, ob er im Transitional- oder Strict-Modus war. Man musste sich das von irgendwoher kopieren, weil sie einfach zu umfangreich waren, um sie sich einprägen zu können.

Dies ist z.B. der DOCTYPE für Dokumente in XHTML Strict:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

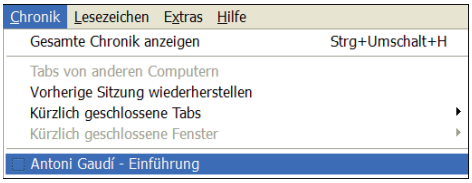
Was für ein Aufwand!

Zum Glück verstehen alle Browser – egal ob alt oder neu – den DOCTYPE von HTML5. Also können Sie sich für all Ihre Seiten daran halten und vergessen einfach, dass es auch mal andere gab. (Das wäre höchstens dann noch relevant, wenn Sie eine ältere Site geerbt haben und der Besitzer es Ihnen nicht erlaubt, den DOCTYPE auf die HTML5-Version zu ändern.)

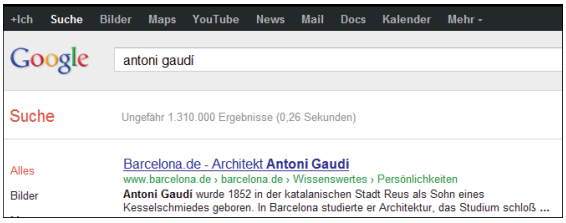
A *Das `title`-Element sollte im Header-Bereich platziert werden. Schreiben Sie es nach dem `meta`-Element, das die Zeichenkodierung angibt.*

```
<!DOCTYPE html>
<html lang="de">
<head>
   <meta charset="UTF-8" />
   <title>Antoni Gaudí - Einführung</title>
</head>
<body>


</body>
</html>
```

B *Bei den meisten Browsern wie Firefox wird der Titel der Webseite sowohl in der Titelleiste des Fensters als auch im Tab (Reiter) gezeigt. Doch Chrome (unten) zeigt den Titel nur im Tab.*



C *Der Seitenname erscheint auch in der (hier gezeigten) Chronik sowie in den Favoriten- bzw. Bookmark-Listen.*



D *Vielleicht am wichtigsten ist, dass Ihr Seitentitel als verlinkter Text in den Suchresultaten von Google und anderen Suchmaschinen verwendet wird. Überdies ist es ein wichtiger Faktor, um die Relevanz einer Seite bei den Suchergebnissen zu bestimmen. Hier sehen Sie den Titel und einen Ausschnitt aus dem Textkörper, wie er in Google erscheint.*

Den Seitentitel erstellen

Im HTML-Basiscode des vorigen Abschnitts stand `<title></title>` als Platzhalter, bis wir endlich Zeit finden, das `title`-Element weiter zu erläutern. Dieser Moment ist nun gekommen!

Jede HTML-Seite muss ein `title`-Element für den Seitennamen aufweisen. Ein Titel soll kurz, aussagekräftig und für jede Seite einmalig sein A. Bei den meisten Browsern erscheint der Seitenname in der Titelleiste des Fensters (Chrome stellt da die Ausnahme dar) B. Außerdem taucht er im Reiter der Seite auf, wenn der Browser das Tabbed Browsing (Registernavigation) unterstützt – also praktisch bei allen großen Browsern, die in den vergangenen Jahren erschienen. Der Seitenname erscheint auch im Browserverlauf sowie in den Favoriten- bzw. Bookmark-Listen C.

Vielleicht noch wichtiger ist, dass der Titel von Suchmaschinen wie Google, Bing und Yahoo! herangezogen wird. Das dient einerseits als Hinweis, mit welchen Informationen auf Ihrer Seite zu rechnen ist, und andererseits erscheint der Titel auch im Link, der in den Suchergebnissen dargestellt wird D.

Kurz gesagt verbessern Sie Ihre Suchmaschinenresultate und die Nutzererfahrung, wenn Sie das `title`-Element für jede Seite unverwechselbar und einmalig gestalten.

Einen Titel erstellen

1. Platzieren Sie den Cursor zwischen `<title>` und `</title>` im `head` des Dokuments.
2. Geben Sie den Namen Ihrer Webseite ein.

TIPP Das `title`-Element muss vorhanden sein.

TIPP Ein Titel darf keine Formatierungen, HTML, Bilder oder Links auf andere Seiten enthalten.

TIPP Manche Code-Editoren füllen den `title` mit Standardtext, wenn Sie eine neue Seite beginnen, außer Sie haben – wie in „Die erste Webseite beginnen" beschrieben – Ihren Editor so eingestellt, dass spezieller Startcode verwendet wird. Also achten Sie jeweils darauf und sorgen Sie ggf. dafür, Standardtext durch einen eigenen `title` zu ersetzen.

TIPP Wenn Ihr `title` Sonderzeichen wie Akzente oder Symbole enthält, müssen Sie diese entweder in Ihre Kodierung aufnehmen (bei UTF-8 normalerweise kein Problem) oder sie als Referenzen schreiben (eine Liste der verfügbaren Zeichen-Entity-Referenzen finden Sie unter *www.elizabethcastro.com/html/extras/entities.html*). Vergessen Sie außerdem nicht, Ihren Code-Editor so einzustellen, dass die Seiten mit der entsprechenden Kodierung gespeichert werden, also z.B. UTF-8, damit die Sonderzeichen korrekt gespeichert werden (siehe „Speichern der Webseite" in Kapitel 2).

Seitentitel unter der Lupe

Viele Entwickler – darunter sogar wohlmeinende mit großer Erfahrung – schenken dem `title`-Element wenig Aufmerksamkeit. Sie geben einfach den Namen der Site ein und kopieren diesen dann auf alle anderen HTML-Seiten. Oder schlimmer noch – sie lassen einfach den `title`-Text stehen, den der Editor vielleicht standardmäßig einfügt. Wenn Sie es sich vorgenommen haben, Traffic auf Ihre Site zu leiten, leisten Sie sich und Ihren potenziellen Lesern einen riesengroßen Bärendienst, falls Sie genauso handeln.

Suchmaschinen arbeiten mit verschiedenen Algorithmen, um den Rang einer Seite zu bestimmen und dessen Inhalt zu indexieren. Bei allen spielt der `title` dabei allerdings eine Schlüsselrolle. Suchmaschinen schauen sich den `title` an, um festzustellen, worum es bei der Seite geht, und indexieren den Inhalt einer Seite auf der Suche nach dazugehörigem Text. Ein effektiver `title` konzentriert sich auf eine Handvoll Schlüsselwörter, die für den Seiteninhalt zentral sind.

Die empfehlenswerte Vorgehensweise ist, einen `title`-Text zu wählen, der den Dokumentinhalt kurz zusammenfasst – davon profitieren sowohl die Nutzer von Screenreadern als auch Ihr Ranking bei den Suchmaschinen. Zweitens geben Sie (optional) den Namen Ihrer Site im `title` an. Den Namen einer Site sieht man üblicherweise am Anfang des `title`, aber besser noch ist es, den typischen, für die Seite spezifischen `title`-Text stattdessen an den Anfang zu setzen.

Ich rate Ihnen, die zentrale Botschaft des `title` in den ersten 60 Zeichen (inklusive Leerzeichen) unterzubringen. Denn als Faustregel kann man sagen, dass Suchmaschinen die Darstellung ihrer Resultate etwa nach dieser Zeichenzahl abschneiden. Browser stellen in der Titelleiste oben im Fenster verschieden viele Zeichen dar (aber nie mehr als 60), bevor sie den Text abschneiden. Bei Browser-Tabs wird der Titel sogar noch eher beschnitten, weil es weniger Platz gibt.

Überschriften erstellen

Bei HTML können Sie für Ihre Webseite bis zu sechs Überschriftsebenen vergeben, um Ihre Seite zu unterteilen. Zeichnen Sie jede Überschrift mit einem der Elemente `h1` bis `h6` aus. Dabei ist `h1` eine Überschrift erster Ordnung, `h2` eine Unterüberschrift davon, `h3` wiederum eine Unterüberschrift von `h2` usw. Wie Sie bald erfahren, gehören Überschriften zu den wichtigsten HTML-Elementen jeder Seite.

Stellen Sie sich die `h1`-`h6`-Überschriften ähnlich vor wie jene aus Nicht-HTML-Dokumenten, die Sie sonst so verfassen, z.B. ein Verkaufsbericht, eine Hausarbeit, ein Produkthandbuch, ein Zeitungsartikel. Wenn Sie solche Arten von Dokumenten schreiben, kennzeichnen Sie jeden Hauptabschnitt des Inhalts passenderweise mit einer Überschrift und einer beliebigen Zahl von Unterüberschriften (und weiteren *Unter*-Unterüberschriften). Zusammengenommen repräsentieren diese Überschriften die Gliederung des Dokuments. Das Gleiche gilt für Ihre Webseiten **A**. Im nächsten Abschnitt „Die Gliederung eines HTML5-Dokuments“ werde ich das weiter ausführen.

Eine Webseite mit Überschriften strukturieren

1. Im `body`-Abschnitt des HTML-Dokuments tippen Sie **`<hn>`** ein, wobei abhängig von der Überschriftsebene, die Sie erstellen wollen, *n* für eine Zahl zwischen 1 bis 6 steht. `h1` ist die wichtigste und `h6` die am wenigsten wichtige Überschrift.
2. Geben Sie den Inhalt der Überschrift ein.
3. Tippen Sie **`</hn>`** ein, wobei *n* die gleiche Zahl ist wie in Schritt 1.

A *Sie können Ihr Dokument anhand von Überschriften in der Art einer Gliederung strukturieren. Hier sind „La Casa Milà” und „La Sagrada Família“ – als `h2`-Elemente ausgezeichnet – Unterüberschriften der obersten Überschrift „Antoni Gaudí“, weil diese ein `h1` ist. (Der Teil `lang="es"` verweist darauf, dass der Inhalt auf Spanisch ist – er wirkt sich nicht auf die Gliederung aus.) Wäre „La Sagrada Família“ stattdessen ein `h3`, wäre es eine Unterüberschrift von „La Casa Milà“ (und eine Unterunterüberschrift von „Antoni Gaudí“.) Die leere Zeile zwischen den Überschriften ist optional und wirkt sich nicht auf die Darstellung der Inhalte aus. Wenn Sie gleich darauf den Rest der Seite schreiben würden, dann würden die damit zusammenhängenden Inhalte (Absätze, Bilder, Videos usw.) direkt nach jeder Überschrift folgen. Auf den nächsten Seiten werden Sie Beispiele davon kennenlernen.*

```
<!DOCTYPE html>
<html lang="de">
<head>
   <meta charset="UTF-8" />

<title>Antoni Gaudí - Einführung</title>
</head>
<body>

<h1>Antoni Gaudí</h1>

<h2 lang="es">La Casa Milà</h2>

<h2 lang="es">La Sagrada Família</h2>


</body>
</html>
```

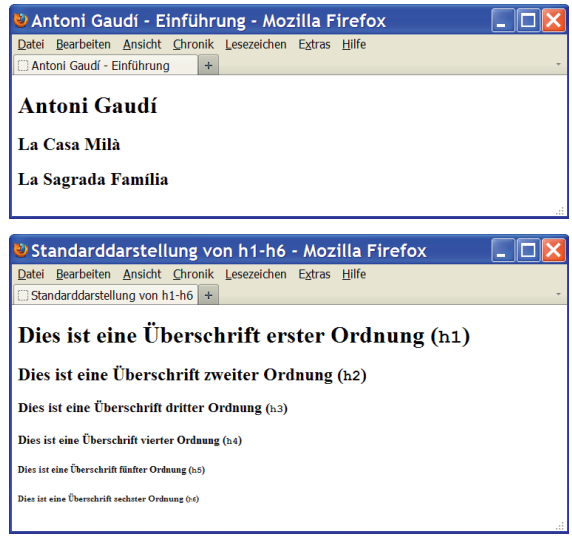
TIPP **Ihre Überschriften h1-h6 sind wegen ihrer Wirkung auf die Definition Ihrer Seitengliederung besonders wichtig. Standardmäßig stellen Browser Überschriften fortschreitend kleiner dar, wenn sie von h1 nach h6 gehen B. Aber vergessen Sie nicht, Ihre Überschriftenebenen einzig danach auszurichten, welche Hierarchie für Ihren Inhalt passend ist, und nicht danach, wie groß oder klein der Text erscheinen soll. Damit wird Ihre Seite semantisch stärker, was im Gegenzug sowohl die SEO als auch die Accessibility (Zugänglichkeit) verbessert. Sie können die Überschriften z.B. mit einer bestimmten Schriftart, Schriftgröße bzw. Schriftfarbe usw. formatieren. Wie man das mit CSS macht, erfahren Sie in Kapitel 10.**

TIPP **Suchmaschinen bewerten Ihre Überschriften sehr stark, vor allem solche in h1 (was nicht bedeutet, dass Sie Ihre Seite mit h1s vollstopfen sollten – dafür sind die Suchmaschinen zu clever). Mittlerweile nutzen Screenreader die Überschriften, um auf einer Seite per Tastatur navigieren zu können, weil man damit schnell auf die Inhalte einer Seite zugreifen und finden kann, was einen am meisten interessiert, ohne sich erst alles vorlesen lassen zu müssen. Noch ein weiterer Grund, sich eine logische Überschriftenhierarchie zuzulegen.**

TIPP **Gestalten Sie die Überschriftenebenen auf Ihrer Site durchgehend konsistent, um eine bessere Nutzererfahrung zu schaffen.**

TIPP **Sie können einer Überschrift auch ein id mitgeben, wenn Sie einen direkten Link auf diese Überschrift erstellen wollen (siehe „Anker erstellen“ in Kapitel 6).**

TIPP **Anmerkung: In A habe ich das Attribut lang bei jedem h2 genommen, um anzuzeigen, dass dessen Inhalte in einer anderen Sprache sind (Spanisch, wie man dem Sprachcode es entnehmen kann) als der Rest der Seite (Deutsch), der durch <html lang="de"> deklariert wird.**



B *Zwar sind alle Überschriften standardmäßig fett, aber h1 hat eine größere Schrift als h2, die wiederum größer ist als h3 usw. Aber wie Sie ja wissen, ist das Aussehen irrelevant, wenn man sich überlegt, welche Überschriftenebene man einsetzen will. Die Darstellung können Sie mit CSS ändern.*

Die Gliederung eines HTML5-Dokuments

Im vorigen Abschnitt haben Sie erfahren, dass die Überschriftenelemente `h1-h6` zur Gliederung Ihrer HTML-Seite beitragen. In diesem Abschnitt werfen wir noch einen Blick hinter die Kulissen und lernen, wie eine Handvoll Elemente, die in HTML5 neu hinzukamen, sich ebenfalls auf die Gliederung auswirken.

Noch eine syntaktische Anmerkung, bevor Sie weitermachen: In den folgenden Erläuterungen und Seiten verwende ich oft den Begriff „Sektion" als generischen Begriff und meine damit einen Teil einer Seite – im Gegensatz zum speziellen `section`-Element (von dem Sie später noch mehr erfahren). Wenn ich genau das `section`-Element meine, dann wird das Wort genauso wie in diesem Satz formatiert.

Okay, also weiter.

Sie wissen nun also, dass jedes HTML-Dokument über eine „eingebaute" Gliederung verfügt – ähnlich wie ein Inhaltsverzeichnis –, die durch die Überschriftenelemente definiert wird. Die Gliederung wird nun von der Seite nicht explizit dargestellt – obwohl Browser eines Tages vielleicht über ein Mittel dafür verfügen –, sondern wie bei allen semantischen Elementen ist sie beispielsweise für Screenreader oder Suchmaschinen von Bedeutung, die diese Gliederung nutzen, um die Struktur Ihrer Seite zu erfassen und die Benutzer mit Informationen zu versorgen.

In den Versionen von HTML und XHTML, die HTML5 vorausgingen, blieben einem nur die Überschriftenelemente `h1-h6`, um die Gliederung zu strukturieren. HTML5 hingegen verfügt über vier Elemente, mit denen man eine Seite in Unterbereiche einteilen kann (das bezeichnet man als *sectioning content*), und zwar `article`, `aside`, `nav` und `section`. Diese grenzen Sektionen innerhalb eines Dokuments voneinander ab und definieren den Bereich der Elemente darin (und auch von `header` und `footer`).

Das bedeutet, dass jedes dieser abgrenzenden Elemente seine eigene `h1`-`h6`-Hierarchie hat, und das stellt einen Riesenfortschritt dar verglichen mit den früheren Versionen der Sprache. Außerdem gestattet die HTML5-Spezifikation, auf einer Seite sogar mehr als nur ein `h1` zu haben (machen Sie sich dazu mal einen Knoten ins Taschentuch, denn gleich werde ich erklären, warum Sie die `h1` begrenzen sollten).

All das wirkt sich auf die Gliederung aus. Vergleichen wir einmal zwei äquivalente Gliederungen, um zu sehen, wie das funktioniert. Für beide gehen wir davon aus, dass auf jede Überschrift eine Reihe von Absätzen und andere Inhalte folgen, die die Informationen des jeweiligen Abschnitts repräsentieren.

Die erste Gliederung ist valides HTML5 und allen vertraut, die sich mit HTML und XHTML auskennen. Sie arbeitet nur mit Überschriftenelementen **A**.

Die zweite Version **B** verwendet sowohl Überschriften als auch `section`-Elemente aus HTML5, darunter ein verschachteltes `section`. (Anmerkung: Dass der Code eingerückt ist, ist unwichtig und wirkt sich nicht auf die Gliederung aus, aber dadurch wird deutlicher, welche Elemente in anderen enthalten sind.)

A *Dokumentgliederung Version 1*

```
...
<body>
   <h1>Produktleitfaden für Anwender</h1>
   <h2>Einrichtung</h2>
   <h2>Basisfeatures</h2>
   <h3>Videowiedergabe</h3>
   <h2>Weitere Features</h2>
</body>
</html>
```

B *Dokumentgliederung Version 2 (gleiche Gliederung wie Version 1, aber aussagekräftiger ausgezeichnet)*

```
...
<body>
   <h1>Produktleitfaden für Anwender</h1>
   <section>
      <h2>Einrichtung</h2>
   </section>

   <section>
      <h2>Basisfeatures</h2>
      <section> <!-- verschachtelt und somit ein
      → Unterabschnitt des übergeordneten
      → Elements -->
         <h3>Videowiedergabe</h3>
      </section>
   </section>

   <section>
      <h2>Weitere Features</h2>
   </section>
</body>
</html>
```

C *Die Dokumentgliederung von Version 3 (nicht die gleiche Gliederung wie die Versionen 1 und 2)*

```
...
<body>
   <h1>Produktleitfaden für Anwender</h1>
   <h1>Einrichtung</h1>
   <h1>Basisfeatures</h1>
   <h1>Videowiedergabe</h1>
   <h1>Weitere Features</h1>
</body>
</html>
```

Vorhin hatte ich erwähnt, dass Browser bisher noch keine Gliederung anzeigen. Doch mit dem HTML 5 Outliner von Geoffrey Sneddon können Sie sie schon mal prüfen (*http://gsnedders.html5.org/outliner/*). Dieses einfache, aber tolle Tool gibt eine visuelle Darstellung der Dokumentgliederung aus. Wenn Sie aus beiden Versionen damit jeweils eine Gliederung generieren (A und B), sehen Sie, dass die Überschriftenebenen für `h1-h6` zwar unterschiedlich sind, aber beide diese Gliederung ergeben:

1. Produktleitfaden für Anwender
    1. Einrichtung
    2. Basisfeatures
        1. Videowiedergabe
    3. Weitere Features

Wie Sie sehen, wird in Version 2 B jedes `html`-Element zu einer Untersektion des am nächsten gelegenen `h1-h6` oder Vorfahrens, das die Seite aufteilt (*sectioning content*) (bei dem es sich in diesem Fall auch um `section` handelt). Das gleiche Verhalten gilt auch für die vier bereits erwähnten HTML5-Elemente für den Sectioning-Content (`article`, `aside`, `nav` und `section`) – auch wenn sie miteinander gemischt werden.

Im Vergleich wäre die Gliederung gänzlich anders, wenn Version 2 B keine `sections` hätte (nennen wir dies Version 3 C).

Jede Überschrift wäre nämlich auf der gleichen Wichtigkeitsstufe `h1` und das bedeutet, dass es *gar keine* untergeordneten Überschriften (oder Unterunterüberschriften) mehr gibt:

1. Produktleitfaden für Anwender
2. Einrichtung
3. Basisfeatures
4. Videowiedergabe
5. Weitere Features

Die beiden Gliederungen mit der gleichen Bedeutung (also Versionen 1 und 2) sind gültiges HTML5, aber die zweite wäre vorzuziehen, weil die `section`-Elemente semantisch mehr aussagen. In der Praxis könnten Sie den gesamten Inhalt von Version 2 in ein `article`-Element setzen, da das in diesem Kontext sogar noch angemessener wäre (aber die daraus resultierende Gliederung wäre etwas anders). Abbildung D zeigt ein Beispiel.

Vergessen Sie außerdem nicht, dass nach jeder Überschrift damit zusammenhängender Text, Bilder und anderer Inhalt folgen. Darüber erfahren Sie im weiteren Verlauf mehr. Aber für den Augenblick habe ich das alles weggelassen, damit Sie sich besser auf Überschriften und Gliederungen konzentrieren können.

Was Sie im heutigen Ökosystem machen können

Aber Moment mal! Da gibt es noch eine andere Anpassung, die Sie beim Code vornehmen sollten. Nun können Sie den Knoten wieder aus dem Taschentuch herausmachen: Warum sollen die `h1` nun limitiert werden? Zwar stimmt es, dass jedes Element, mit dem man Inhalt voneinander abgrenzen oder sektionieren kann (also `article`, `aside`, `nav` und `section`), seine Überschriftenhierarchie mit `h1` beginnen *kann*, aber zwingend ist das nicht. Vielmehr ist zutreffend, dass Sie bis in absehbarer Zukunft besser damit fahren, jede solche Sektion mit einem `h2` oder tiefer anzufangen, falls diese eine Unterüberschrift eines bereits vorhandenen `h1`-Elements für einen damit zusammenhängenden Inhalt darstellen soll.

Der Grund:

Das sich ständig weiterentwickelnde Internet enthält eine Vielzahl beweglicher Teile. Neue Spezifikationen wie HTML5 ändern sich täglich, bis sie final sind (das kann Jahre dauern und ist bei HTML5 noch gar nicht so weit). Neue Browser-Versionen erscheinen und neue Versionen von Screenreadern und weiterer assistiver Technologien werden entwickelt und veröffentlicht. Das alles ist nun gar nicht sonderlich synchron oder aufeinander abgestimmt.

D *Gliederungsbeispiel mit expliziter Semantik*

```
...
<body>
   <article>
      <h1>Produktleitfaden für Anwender</h1>
      <section>
         <h1>Einrichtung</h1>
      </section>

      <section>
         <h1>Basisfeatures</h1>
         <section>
         <h1>Videowiedergabe</h1>
         </section>
      </section>

      <section>
         <h1>Weitere Features</h1>
      </section>
   </article>
</body>
</html>
```

Stattdessen läuft das meist so ab, dass jeder Browser sein Spektrum inkrementell ausbaut (was im Allgemeinen eine gute Sache ist) und dabei nicht notwendigerweise wie seine Konkurrenz vorgeht (das ist hingegen meist nicht so gut) und ganz gewiss nicht nach dem gleichen Fahrplan wie die anderen. Das Gleiche gilt für Screenreader. Obwohl also moderne Browser viele der HTML5-Features unterstützen, ist keiner zum Zeitpunkt, während ich dies schreibe, in der Lage, für Screenreader die HTML5-Dokumentgliederung auszugeben, und Screenreader können sie ihren Nutzern gegenüber auch nicht ausgeben.

Kurz gesagt bedeutet das, dass Screenreader und andere assistive Technologien bisher nicht unterscheiden zwischen einem `h1`, das sich direkt im `body` befindet, und einem `h1`, das sich in einem `article`, `aside`, `nav` oder `section` befindet. Aus deren Perspektive ist alles `h1` auf oberster Ebene. Bruce Lawson, der bekannte Webevangelist für Opera, war meines Wissens der erste, der dies aufgezeigt hat. (*www.brucelawson.co.uk/2009/headings-in-html-5-and-accessibility/*; bitte beachten Sie, dass manche Informationen hier möglicherweise veraltet sind, weil sich die Spezifikation seitdem verändert haben kann. Lesen Sie dazu auch sein aktuell erschienenes *HTML5 – Eine Einführung für Umsteiger* (Addison-Wesley 2011), das er mit Remy Sharp verfasst hat.)

In der Zwischenzeit können sich Screenreader nicht den Luxus leisten abzuwarten, bis das Internet ihre Bedürfnisse erfüllt. Sie werden damit weitermachen, sich anhand der Überschriften einen Überblick über den Inhalt der Seite zu verschaffen und gleichzeitig auch über die Navigation darin, wie sie das bisher auch immer gemacht haben. Mit einer aussagekräftigen Hierarchie der Überschriften wird das einfacher und es sorgt für eine bessere Nutzererfahrung auf Ihrer Site.

Bis also das Ökosystem hier ein wenig aufgeholt hat, fahren Sie und Ihre Nutzer besser, wenn Sie in den Überschriften die Hierarchie explizit mit `h1`-`h6` anzeigen – genauso, wie Sie es machen würden, falls keine abgrenzenden Elemente vorhanden wären. Lawson und andere Kollegen der Branche empfehlen diesen Ansatz und ich schließe mich dem an.

Schauen wir uns nun an, wie vorzugehen ist **E**.

Was in der ersten Ebene der `section`-Elemente `h1` war, ist nun `h2`, und die Überschrift „Videowiedergabe“, die sich in einem doppelt verschachtelten `section` befand, ist nun ein `h3` statt ein `h1`. Die Dokumentgliederung hat sich nicht geändert, nur die Überschriftenebenen.

Dieses Beispiel demonstriert nur `h1`-`h3`, aber falls Ihre Inhalte das hergeben, sollten Sie auch mit `h4`-`h6` arbeiten. Eine Unterüberschrift für „Videowiedergabe“ wäre dann beispielsweise eine `h4` (mit oder ohne übergeordnetes Sektionierungselement – wie es Ihnen beliebt) usw.

Und denken Sie daran, dass diese Empfehlung für alle Sektionierungselemente (`article`, `aside`, `nav` und `section`) gilt und nicht nur für die in diesem Beispiel gezeigten.

E *Version 4 (der von allen vier Versionen empfehlenswerte Ansatz für Überschriften)*

```
...
<body>
<article>
   <h1>Produktleitfaden für Anwender</h1>
   <section>
      <h2>Einrichtung</h2>
   </section>

   <section>
      <h2>Basisfeatures</h2>
      <section>
      <h3>Videowiedergabe</h3>
      </section>
   </section>

   <section>
      <h2>Weitere Features</h2>
   </section>
</article>
</body>
</html>
```

Zusammenfassung

Ich schlage vor, dass Sie diese Erläuterungen über die HTML5-Dokumentgliederung erneut lesen, falls etwas davon noch nicht ganz eingängig war. Es ist nicht annähernd so kompliziert, wie es scheint. Ich rate Ihnen nachdrücklich, einfach mal ein paar unterschiedliche Testseiten zu erstellen und die Resultate im HTML5 Outliner zu vergleichen, damit Sie ein besseres Gefühl für die Arbeitsweise des Gliederungsalgorithmus bekommen. Nutzen Sie den Outliner auch in Ihrer Projektarbeit, um sicher zu sein, dass Ihre Struktur wie gewünscht ist. Zuerst prüfen Sie aber die HTML5-Seiten entweder unter *http://validator.nu/* oder *http://validator.w3.org/*, um Coding-Fehler aufzudecken (siehe „Den Code validieren“ in Kapitel 20).

TIPP Hier soll allerdings nicht der Eindruck entstehen, dass Sie immer ein `article` verwenden müssen oder dass ein `section` immer (und ausschließlich) in einem `article` verschachtelt sein kann. Das hier erläuterte Beispiel war nur eine Möglichkeit, wie man diese Elemente verwenden kann, und tatsächlich kann man den gleichen Inhalt auch auf andere Weise auszeichnen und man hätte immer noch valides HTML5. Ich erkläre `article` und `section` später in diesem Kapitel noch genauer und Sie werden merken, dass abhängig von Ihren Inhalten verschiedene Anwendungen dafür möglich sind.

Wie der Gliederungsalgorithmus von HTML5 bei syndizierten Inhalten unterstützt

Wenn Sie bis hierhin alles mitverfolgt haben, wissen Sie, dass alle `article`-, `aside`-, `nav`- und `section`-Elemente aufgrund des Gliederungsalgorithmus von HTML5 und auch, weil sie alle zum Sectioning Content gehören, jeweils eine eigene Gliederung haben können. Diese beginnt mit `h1` und kann bis `h6` reichen.

Das erlaubt nicht nur eine große Flexibilität bei den Überschriften Ihres Dokuments, aber ein weiterer Vorteil ist nicht so offensichtlich: So können Ihre Inhalte auf anderen Seiten und sogar anderen *Websites* erscheinen, ohne dass damit die Gliederung des übergeordneten Dokuments zerschossen wird. Und die jeweils eigene Gliederung bleibt ebenfalls intakt.

Heutzutage werden immer mehr Inhalte von unterschiedlichen Sites gemeinsam genutzt. Es gibt Sites, auf denen Nachrichten aggregiert werden, Blogs mit RSS-Feeds, Twitter-Feeds usw. Wie Sie in „Einen Artikel erstellen“ lernen werden, repräsentiert das `article`-Element eine eigenständige Komposition, die man syndizieren *kann* (das muss nicht sein, aber passenderweise sollte man das so machen).

Nehmen wir an, dass das folgende `article` von einer Site auf einer anderen Site dargestellt wird:

```
...
<h2>News aus dem Netz</h2>

<article>
   <h1>Teenager hört lieber Vinyl als digital</h1>

   <p>Ein Mädchen hat all ihre digitalen Musikstücke durch Vinyl ersetzt. "Das ist echt
   → abgefahren", verkündete sie.</p>

   <h2>Süchtig schon nach dem ersten Album</h2>
   ...
</article>
...
```

Wenn Sie diesen Code mit dem HTML5 Outliner prüfen, bekommen Sie Folgendes zu sehen:

1. News aus dem Netz
    1. Teenager hört lieber Vinyl als digital
        1. Süchtig schon nach dem ersten Album

Obwohl also die Überschrift *Teenager* eine höhere Ebene (`h1`) ist als das `h2`, unter dem es sich befindet, ist es eine Unterüberschrift des `h2`, weil es in einem `article` unter dieser Überschrift steht. Und die `h2`-Überschrift mit *Süchtig* ist eine Unterunterüberschrift von der `h2`-Überschrift mit den News und befindet sich nicht auf gleicher Ebene.

Die News-Überschrift könnte eine `h3`, eine `h4` oder *irgendeine* andere Überschriftenebene sein und die Gliederung wäre exakt die gleiche. Das Gleiche gilt für *Teenager* und *Süchtig*, solange sich *Teenager* auf einer höheren Ebene befindet.

A *Zwei zusammengehörige Überschriften werden gemeinsam gruppiert. In diesem Beispiel ist h2 eine Unterüberschrift der h1-Überschrift des Artikels. Weil es mit der höchsten Überschrift ausgezeichnet wurde, erscheint in der Dokumentgliederung nur „Giraffe entflieht aus Zoo“, aber beide werden erwartungsgemäß im Browser dargestellt* **B**. *Wenn entsprechend ein zweites h1 danach in der hgroup erschiene, fiele es in der Gliederung so wie das h2 unter den Tisch. Weil das h2 in der Gliederung nicht erscheint, könnte die nächste Überschrift im Artikel ein h2 (statt eines h3 sein) und auch als direkte Unterüberschrift des h1 „Giraffe entflieht aus Zoo“ verstanden werden.*

```
...
<body>

<article>
   <hgroup>
      <h1>Giraffe entflieht aus Zoo</h1>
      <h2>Allerorten frohlocken die Tiere</h2>
   </hgroup>

   <p>... [Artikelinhalt] ...</p>
</article>

</body>
</html>
```

B *Beide Überschriften werden im Browser dargestellt, so als ob sie nicht in einer hgroup enthalten wären.*

TIPP Nehmen Sie hgroup nicht, wenn es nur eine einzige Überschrift gibt. Es ist für mindestens zwei gedacht.

TIPP Wie bereits erwähnt, erscheint nur die erste Instanz der höchstrangigen Überschrift in einer hgroup in der Dokumentgliederung. Die Reihenfolge der Überschriften ist irrelevant. Wenn in Ihrem hgroup also ein h3 auf ein h2 folgt, erscheint das h2 in der Gliederung. Üblicherweise sortieren Sie Überschriften nach dem Prioritätslevel. Somit würde eine weniger gewichtete Überschrift wie h3 nicht vor einer höheren wie h2 kommen. Gelegentlich könnte es allerdings Ausnahmen geben.

Überschriften gruppieren

Manchmal hat eine Überschrift mehrere aufeinanderfolgende Ebenen, so wie bei Überschriften mit Unterzeilen, alternativen Titeln oder Slogans. Wenn man die alle in ein `hgroup`-Element gruppiert, zeigt man, dass sie zusammengehörig sind **A**. Jedes `hgroup` kann nur zwei oder mehr `h1`-`h6`-Überschriften enthalten, andere Elemente sind nicht erlaubt.

Nur die erste Instanz der höchstrangigen Überschrift in einer `hgroup` erscheint in der Dokumentgliederung (siehe „Die Gliederung eines HTML5-Dokuments“). Das könnte also ein weiterer zu berücksichtigender Aspekt sein, um sich für `hgroup` zu entscheiden. Der Klarheit halber sollte noch erwähnt werden, dass alle Überschriften in einer `hgroup` im Browser dargestellt werden **B**.

Zwei oder mehr Überschriften gruppieren

1. Tippen Sie **`<hgroup>`**.
2. Tippen Sie **`<hn>`**, wobei abhängig von der Überschriftsebene, die Sie erstellen wollen, *n* für eine Zahl zwischen 1 und 6 steht.
3. Geben Sie den Inhalt der Überschrift ein.
4. Tippen Sie **`</hn>`** ein, wobei *n* die gleiche Zahl ist wie in Schritt 2.
5. Wiederholen Sie die Schritte 2 bis 4 für alle Überschriften, die Sie in `hgroup` aufnehmen wollen. Üblicherweise wird die Überschriftenebene für jede nachfolgende Überschrift um eins hochgezählt (z.B. von `h1` zu `h2` usw.).
6. Tippen Sie **`</hgroup>`**.

Übliche Seitenkonstrukte

Zweifelsohne haben Sie schon Dutzende Sites besucht wie jene, die in A gezeigt wird. Wenn man sie vom Inhalt entkleidet, erkennen Sie vier Hauptkomponenten: eine Kopfzeile (Masthead) mit Navigation, einen Artikel im Hauptinhaltsbereich, eine Seitenleiste (Sidebar) mit weiterführenden Informationen und eine Fußzeile (Footer) B.

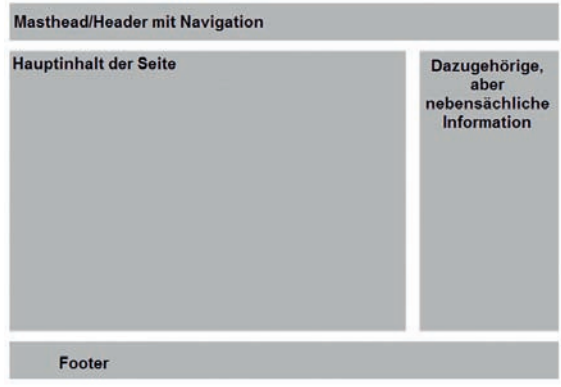
Nun können Sie eine solche Seite nicht so A formatieren oder sie wie gezeigt arrangieren (A und B), ohne CSS hinzuzunehmen. In Kapitel 7 geht es dann mit dem Erlernen von CSS los, über das Formatieren und Färben von Text erfahren Sie mehr in Kapitel 10 und mehrspaltige Layouts lernen Sie in Kapitel 11 kennen.

Doch die Semantik, die für diese üblichen Seitenkonstrukte angewendet werden, ist praktisch immer recht ähnlich, egal wie das Layout aussieht. Darum wird es nun vorrangig auf den restlichen Seiten dieses Kapitels gehen. Wir arbeiten uns von oben nach unten durch die Seite und lernen, wie man die Elemente `header`, `nav`, `article`, `section`, `aside` und `footer` nutzt, um die Struktur der Seiten zu definieren. Dann kommt `div` als generischer Container ins Spiel, der weiteren Formatierungen und anderen Zwecken dient. Außer `div` gab es vor HTML5 diese Elemente noch gar nicht. Bei ein paar haben Sie schon in einigen der früheren Codebeispiele und Erläuterungen einen kleinen Einblick bekommen.

Wenn Sie sich diese Elemente näher aneignen, sollten Sie sich nicht so sehr daran aufhängen, wo sie jeweils in den Seitenlayouts erscheinen, und sich stattdessen auf die semantische Bedeutung konzentrieren.

Auf den folgenden Seiten werden Sie vorab bereits auch andere Elemente kennenlernen, z.B. `ul` (unsortierte Liste) und `a` (für Links). In späteren Kapiteln werden diese alle ausführlich erläutert.

A *Ein übliches Layout mit der Hauptnavigation am oberen Rand, dem Hauptinhalt links, eine Seitenleiste rechts und der Fußzeile unten. Damit die Seite so aussieht, wird CSS benötigt.*



B *Diese Art Informationen findet man üblicherweise auf einer Seite. Dies ist nur ein mögliches Arrangement, allerdings ein recht häufiges.*

A *Dieser `header` repräsentiert den Header für die ganze Seite. Er enthält eine Liste von Links in einem `nav`-Element, um anzuzeigen, dass es sich um eine primäre Navigation auf der Seite handelt. Schauen Sie sich* C *an: Hier wird aus Gründen der besseren Zugänglichkeit das optionale `role="banner"` auf einen `header` auf Seitenebene angewendet (siehe Navigation auszeichnen", wenn Sie mehr über den fürs `nav`-Element spezifischen `role`-Wert wissen wollen).*

```
...
<body>
<header>
   <nav>
      <ul>
         <li><a href="#gaudi">Der Architekt von
         → Barcelona</a></li>
         <li lang="es"><a href="#sagrada-familia">
         → La Sagrada Família</a></li>
         <li><a href="#park-guell">Park Güell</a>
         → </li>
      </ul>
   </nav>
</header>
</body>
</html>
```

B *Der Header auf Seitenebene mit der Navigation*

Einen Header erstellen

Wenn es auf Ihrer Seite einen Bereich gibt, der eine Gruppe von einführenden oder navigationsbezogenen Inhalten hat, zeichnen Sie diesen mit dem `header`-Element aus.

Eine Seite kann eine beliebige Zahl `header`-Elemente bekommen und je nach Kontext kann sich deren Bedeutung unterscheiden. Ein `header` ganz oder recht weit oben auf einer Seite kann den Header (Kopfzeile) für die ganze Seite repräsentieren (manchmal auch Masthead genannt) A. Oft stehen in der Kopfzeile das Logo der Site, die Hauptnavigation B, andere globale Links und manchmal auch ein Suchfeld. Zweifellos wird das `header`-Element dafür am häufigsten eingesetzt, aber täuschen Sie sich nicht: Das ist nicht der einzige Verwendungszweck.

Ein `header` eignet sich auch zum Auszeichnen einer Gruppe von einführendem oder navigationsbezogenem Inhalt, die sich weiter unten auf der Seite befindet. Ein Beispiel dafür ist das Inhaltsverzeichnis eines Bereichs C.

Das `header`-Element ist eines der vier Elemente, die die Seite in Bereiche aufteilen können, von denen ich im Abschnitt „Die Gliederung eines HTML5-Dokuments" gesprochen habe. Das bedeutet, dass jede `h1`-`h6`-Überschrift innerhalb eines `header` als zum Kontext des `header` gehörig betrachtet wird und nicht der Seite insgesamt – zumindest, was die Dokumentgliederung angeht. Ein `header`-Element enthält also oft die Überschrift seines Abschnitts (ein `h1`-`h6` oder `hgroup`), aber das ist nicht zwingend. Sie sehen beispielsweise Überschriften in C, aber nicht in A.

Einen Header erstellen

1. Platzieren Sie den Cursor innerhalb des Elements, für das Sie einen Header erstellen wollen.
2. Tippen Sie **<header>**.
3. Tippen Sie die Inhalte des Headers. Das können die verschiedenen Inhaltstypen sein, die mit ihren jeweiligen HTML-Elementen ausgezeichnet werden (die meisten davon lernen Sie im restlichen Buch kennen). Ein `header` kann beispielsweise Überschriften der Ebene `h1` bis `h6` enthalten, ein oder mehrere Logos, die Navigation, ein Suchfeld usw.
4. Tippen Sie **</header>**.

C *Diese Seite hat zwei `header`: Einer dient als Header der Seite und ein anderer als Header für das übergeordnete `article`-Element mit den Frequently Asked Questions. Beachten Sie, dass im ersten keine `h1`-`h6`-Überschriften sind, aber im zweiten. Im letzten Tipp dieses Abschnitts finden Sie Informationen über das optionale Attribut `role` im ersten `header`.*

```
...
<body>
<header role="banner">
   ... [Site-Logo, Navigation etc.] ...
</header>

<article>
   <header>
      <h1>Frequently Asked Questions</h1>
      <nav>
         <ul>
            <li><a href="#antwort1">Wie lauten Ihre Rücknahmebedingungen?</a><li>
            <li><a href="#antwort2">Wie finde ich eine Filiale?</a><li>
            ...
         </ul>
      </nav>
   </header>

   <!-- Die Links im Header zeigen hierhin -->
   <article id="antwort1">
      <h2>Wie lauten Ihre Rücknahmebedingungen?</h2>
      <p> ... [Antwort] ... </p>
   </article>

   <article id="antwort2">
      <h2>Wie finde ich eine Filiale?</h2>
      <p> ... [Antwort] ... </p>
   </article>
   ...
</article> <!-- Ende des übergeordneten Artikels -->

</body>
</html>
```

TIPP **Nehmen Sie `header` nur, wo es nötig ist. Wenn Sie nur ein `h1`-`h6` oder `hgroup` haben und darin kein Inhalt steht, für den sich eine Gruppierung lohnt, gibt es meist keinen Grund, das alles in einen `header` zu fassen.**

TIPP **Ein `header` ist nicht mit einer Überschrift austauschbar wie die `h1`-`h6`-Elemente (siehe „Überschriften erstellen“.) Alle haben ihren eigenen semantischen Zweck.**

TIPP **Sie dürfen kein `footer`-Element verschachteln oder einen `header` in einen anderen `header` stecken, auch kein `header`-Element in ein `footer`- oder `address`-Element stecken.**

TIPP **Ein `header`-Element muss nicht immer ein `nav`-Element enthalten wie in diesen Beispielen (A und C), aber in den meisten Fällen wird das so sein, wenn das `header`-Element Navigationslinks enthält. Bei C ist das `nav` um die Liste der Links für die FAQs passend, weil es sich um eine wichtige Navigationsgruppe auf der Seite handelt (siehe Erläuterungen in „Navigation auszeichnen“).**

TIPP **In „Generische Container erstellen“ erfahren Sie mehr, wie durch `header` eine der Rollen des `div`-Elements aus der Zeit vor HTML5 ersetzt wurde.**

TIPP **Über die Verwendung von `role="banner"` mit `header` erfahren Sie mehr im Abschnitt „Mit ARIA die Zugänglichkeit verbessern“.**

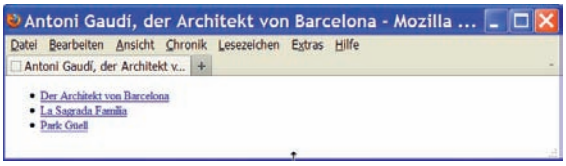
Navigation auszeichnen

Bei früheren Versionen von HTML gab es kein Element, das explizit eine Sektion von wichtigen Navigationslinks repräsentiert, aber HTML5 wartet nun damit auf: das `nav`-Element. Links in einem `nav` können auf Inhalte dieser Seite zeigen (A), auf andere Seiten bzw. Ressourcen oder beides. In jedem Fall sollten Sie `nav` in Ihrem Dokument aber nur für die wichtigsten Gruppen von Links verwenden, nicht für alle.

Wenn Sie sich den Code im vorigen Abschnitt genauer anschauen, sehen Sie das `nav`-Element in Aktion. Ich greife dieses Codebeispiel noch einmal auf dieser Seite auf und hebe `nav` hervor (A). Das `nav`-Element zwingt seinen Inhalten keine Standardformatierung auf (B).

(A) *Diese Links (die `a`-Elemente) repräsentieren eine wichtige Navigationsgruppe, also habe ich sie in ein `nav`-Element verschachtelt. Normalerweise werden Sie eine Liste von Links mit dem `ul`-Element (unordered list, Aufzählungsliste) auszeichnen, außer es handelt sich um Breadcrumb-Links. In einem solchen Fall verwenden Sie `ol` (ordered list, nummerierte Liste). In Kapitel 15 erfahren Sie mehr über Listen. Das Attribut `role` ist nicht erforderlich, kann aber die Zugänglichkeit verbessern. Siehe den letzten Tipp in diesem Abschnitt, wie man `role="navigation"` auf `nav` anwendet.*

```
...
<body>
<header>
   <nav role="navigation">
      <ul>
         <li><a href="#gaudi">Der Architekt von
         → Barcelona</a></li>
         <li lang="es"><a href="#sagrada-familia">
         → La Sagrada Família</a></li>
         <li><a href="#park-guell">Park Güell</a>
         → </li>
      </ul>
   </nav>
</header>
</body>
</html>
```



B *Unsere Navigation sieht standardmäßig recht schlicht aus. Die Gliederungspunkte (Bullets) sind kein Produkt des* `nav`*-Elements, das keine Standardformatierung aufweist, außer dass es in seiner eigenen Zeile beginnt. Die Gliederungspunkte werden dargestellt, weil jeder Link sich in einem* `li`*-Element befindet (list item, Listenelement). Mit CSS können Sie die Gliederungspunkte abschalten oder andere wählen sowie die Links auch horizontal anordnen, deren Farbe ändern, sie wie Schaltflächen aussehen lassen etc. Mehr über CSS erfahren Sie in Kapitel 7.*

Eine Link-Gruppe als wichtige Navigation kennzeichnen

1. Tippen Sie `<nav>`.
2. Tippen Sie Ihre Link-Liste strukturiert als `ul` (Aufzählungsliste), außer es kommt auf die Reihenfolge der Links an (z.B. bei einer Breadcrumb-Navigation). In diesem Fall sollten Sie sie als `ol` (nummerierte Liste) strukturieren (in den Kapiteln 6 bzw. 15 erfahren Sie mehr über Links bzw. Listen).
3. Tippen Sie `</nav>`.

TIPP Falls Sie schon Erfahrung mit HTML oder XHTML haben, sind Sie es wohl schon gewöhnt, Ihre Links je nachdem in einem `ul`- oder einem `ol`-Element zu strukturieren. In HTML5 wird diese Best Practice durch `nav` nicht ersetzt. Arbeiten Sie weiterhin mit diesen Elementen und umschließen Sie sie einfach mit einem `nav` A.

TIPP Während Screenreader bei der neuen HTML5-Semantik generell gesehen immer noch versuchen, Anschluss zu finden, kann das `nav`-Element dabei helfen, die wichtige Navigation auf Ihrer Seite zu identifizieren. Damit können die User sie per Tastatur direkt anwählen. So wird Ihre Seite besser zugänglich, was die Nutzererfahrung verbessert.

TIPP Die HTML5-Spezifikation empfiehlt, nebensächliche Fußzeilen auf der Seite wie „Nutzungsbedingungen“ oder „Datenschutzbestimmungen“ nicht in ein `nav` zu stellen, was sinnvoll ist. Manchmal wird die allgemeine Top-Level-Navigation im Footer wieder aufgenommen oder es stehen andere wichtige Links wie „Suche nach Filialen“ oder „Jobangebote“ darin. Für die meisten Fälle halte ich es für ratsam, diese Arten von Footer-Links in ein `nav` zu stellen.

TIPP HTML5 erlaubt keine Verschachtelung eines `nav` innerhalb eines `address`-Elements.

TIPP Über die Verwendung von `role="navigation"` mit `nav` erfahren Sie mehr im Abschnitt „Mit ARIA die Zugänglichkeit verbessern“ A.

nav unter der Lupe

Wie bereits erwähnt, müssen Sie eine Gruppe von Links auf Ihrer Seite nicht notwendigerweise in ein `nav` stecken.

Das folgende Beispiel mit einer Nachrichtenseite enthält vier Listen mit Links, von denen nur zwei als so relevant erachtet werden, dass es sich lohnt, sie mit einem `nav` zu umschließen (der Code ist an bestimmten Stellen gekürzt).

```
...
<body>
   <header>
      <!-- Hier könnte das Site-Logo stehen -->
     <!-- Globale Navigation der Site -->

      <nav>
         <ul> ... </ul>
      </nav>
   </header>

   <div id="main">
      <h1>Kunst &amp;amp; Unterhaltung</h1>
      <article>
         <h1>Galerieeröffnung stellt inspirierte Arbeiten vor</h1>
         <p>... [Inhalt der Story] ... </p>

         <aside>
            <h1>Andere Storys</h1>

            <!-- Nicht in nav gefasst -->
            <ul> ... [Story-Links] ... </ul>
         </aside>
      </article>
   </div>
```

```
    <aside id="sidebar">
      <nav><!-- Sekundäre Navigation -->
        <ul>
          <li><a href="/kunst/filme/">Filme</a></li>
          <li><a href="/kunst/musik/">Musik</a></li>
          ...
        </ul>
      </nav>
    </aside>

    <footer>
      <!-- Weniger relevante Links stehen nicht im nav. -->
      <ul> ... </ul>
    </footer>
</body>
</html>
```

Die sekundäre Navigation im `aside` (siehe „Ein `aside` angeben“) erlaubt es dem User, auf die anderen Seiten im Verzeichnis „Kunst & Unterhaltung“ zu navigieren, und konstituiert somit einen wichtigen Navigationsbereich auf der Seite. Aber beim `aside` „Andere Storys“ mit Links ist das nicht der Fall.

Wonach entscheiden Sie nun also, ob eine Gruppe mit Links ein `nav` verdient? Letzten Endes fällen Sie diese Beurteilung basierend auf Ihrer inhaltlichen Organisation. Zumindest die globale Navigation der Site sollten Sie mit `nav` auszeichnen (also den Bereich, über den User auf andere Bereiche der Site springen können). Oft, aber nicht immer, erscheint dieses spezielle `nav` innerhalb eines `header`-Elements auf Seitenebene (siehe „Einen Header erstellen“).

Einen Artikel erstellen

Ein anderes Element, das wir HTML5 zu verdanken haben, ist `article` Ⓐ. Das haben Sie schon ein paar Mal in Aktion gesehen. Nun sollen Sie mehr darüber erfahren, wie es eigentlich funktioniert.

Am Namen können Sie schon merken, dass man in `article` Inhalte wie einen Zeitungsartikel packen kann. Doch darauf ist es nicht beschränkt. In HTML5 ähnelt ein „Artikel“ eher einem „Element“.

So wird das in HTML5 definiert:

> Das `article`-Element ist ein eigenständiger Teil in einem Dokument, einer Seite, Anwendung oder Site und im Prinzip unabhängig wiederverwendbar oder weitergebbar, z.B. in einer Syndizierung. Es ist beispielsweise anwendbar auf ein Forum-Posting, einen Magazin- oder Zeitungsartikel, einen Blog-Eintrag, einen Benutzerkommentar, ein interaktives Widget oder Gadget oder irgendein anderes, inhaltlich unabhängiges Element.

Andere `article`-Elemente könnten eine Film- oder Konzertrezension, eine Fallstudie, eine Produktbeschreibung usw. sein. Sie sind vielleicht überrascht, dass es auch ein interaktives Widget oder Gadget sein kann, aber auch dabei handelt es sich um unabhängige, wiederverwendbare Inhaltselemente.

Ⓐ *Ich habe den Artikelinhalt und den nav-Code aus dem vorigen Abschnitt gekürzt, um das Beispiel überschaubar zu halten. Die komplette Version des Seitencodes finden Sie auf der Buchseite unter www.bruceontheloose.com/htmlcss/examples/. Obwohl dieses Beispiel nur Absätze und Bilder enthält, kann ein `article` inhaltlich sehr unterschiedliche Typen wie Video, Abbildungen, Listen usw. enthalten.*

```
...
<body>
<header>
   <nav role="navigation">
      ... [ul mit Links] ...
   </nav>
</header>
<article>
   <h1 id="gaudi">Der Architekt von Barcelona
   → </h1>

   <p>Viele Touristen kommen extra nach
   → Barcelona, um sich die unbeschreibliche
   → Architektur von Antoni Gaudí anzuschauen.
   → </p>

   <p>Seine Nonkonformität, die schon in
   → Jugendjahren deutlich wurde, verbunden
   → mit seiner ruhigen, aber innigen Hingabe an
   → die Kirche, schufen eine einzigartige
   → Grundlage für seine Gedanken und Ideen.
   → Seine Suche nach Einfachheit basierte auf
   → sorgfältigen Naturbeobachtungen, was sich
   → sehr deutlich in seinem Werk niederschlägt
   → - sei es der <a href="#park-guell">Park
   → Güell</a> mit seinen märchenhaften
   → Skulpturen und Mosaiken oder auch die Kirche
   → der <a href="#sagrada-familia">Heiligen
   → Familie</a> mit ihren organischen, an
   → Bischofsmützen erinnernden Türme. </p>

   <h2 id="sagrada-familia" lang="es">La Sagrada
   → Família</h2>

   ... [Bild und Absätze] ...

   <h2 id="park-guell">Park Güell</h2>

   ... [Bild und Absätze] ...
</article>

</body>
</html>
```

B *Nun enthält die Seite `header`-, `nav`- und `article`-Elemente und auch die entsprechenden Inhalte dafür. Die `article`-Überschriften können je nach Browser standardmäßig auch unterschiedliche Größen haben. Sie können das Aussehen auch browserübergreifend mit CSS vereinheitlichen (siehe Kapitel 10).*

Einen Artikel erstellen

1. Tippen Sie `<article>`.
2. Geben Sie die Inhalte des Artikels ein – eine beliebige Anzahl von Elementen wie Absätze, Listen, Audio, Video, Fotos, Abbildungen usw.
3. Tippen Sie `</article>`.

TIPP Wie Sie im Abschnitt „Die Gliederung eines HTML5-Dokuments“ erfahren haben, ist `article` neben `header`, `section` und `aside` eines der vier Elemente, das die Seite in Bereiche aufteilen kann.

TIPP Sie können einen `article` in einem anderen verschachteln, solange das innere `article` mit dem `article` insgesamt zusammenhängt. Sie können allerdings ein `article` nicht in einem `address`-Element verschachteln.

TIPP Eine Seite kann mehrere `article`-Elemente (oder gar keins) enthalten. Die Homepage eines Blogs enthält beispielsweise üblicherweise ein paar der aktuellen Postings, und davon kann jedes ein eigener `article` sein.

TIPP Es ist nicht zwingend notwendig, dass ein `article` ein oder mehrere `section`-Elemente enthält. Valide wäre auch, wenn die `h1`-`h6`-Elemente innerhalb eines `article` für sich stehen, obwohl Sie durch die Definition von `sections` die Semantik des `article` expliziter machen. Und jede `section` kann ihre eigene Hierarchie von Überschriftenebenen haben – siehe die Erläuterungen in „Die Gliederung eines HTML5-Dokuments“.

TIPP Die Elemente `article` und `section` kann man leicht (und verständlicherweise) miteinander verwechseln, und deswegen habe ich direkt aus der HTML5-Definition zitiert. Ich wollte nicht, dass Sie davon sozusagen gefiltert erfahren. Ich werde `section` und die Wahl zwischen den beiden im Abschnitt „Einen Abschnitt (section) definieren“ erläutern.

TIPP Über die Verwendung von `role="main"` mit `article` unter bestimmten Umständen erfahren Sie mehr im Abschnitt „Mit ARIA die Zugänglichkeit verbessern“. Es wäre auch passend, ihn beim `article` in A aufzunehmen, weil es der Container für den Hauptinhalt der Seite ist. Aber ich habe es da weggelassen, um nicht den Eindruck zu erwecken, dass `role="main"` für alle `article`-Elemente richtig ist.

Mehr article-Beispiele

Das vorige Beispiel (A) ist nur eine Einsatzmöglichkeit für `article`. Schauen wir uns einige andere an.

Beispiel 1 (einfacher Artikel):

```
<article>
   <h1>Die Vielfalt von Papua-Neuguinea</h1>
   <p>Papua-Neuguinea ist die Heimat von über 800 Stämmen und Sprachen ... </p>
   ... [restlicher Inhalt]

   <footer> <!-- Der Footer des Artikels, nicht der der Seite -->
      <p>Leandra Allen ist eine freiberufliche Journalistin, die ihren Abschluss in Anthropologie
      → an der Universität Kopenhagen gemacht hat.</p>
      <address>
      Sie erreichen sie unter <a href="mailto:leandra@therunningwriter.com">
      → leandra@therunningwriter.com</a>.
      </address>
   </footer>
</article>
```

Beachten Sie, wie hier die Elemente `footer` und `address` eingesetzt werden (weitere Erläuterungen darüber finden Sie in diesem Kapitel bzw. in Kapitel 4). Hier wird `address` nur auf das übergeordnete `article` (hier gezeigt) angewendet, nicht auf die Seite oder andere `article`, die innerhalb dieses `article` verschachtelt sind, z.B. die Leserkommentare in Beispiel 2.

Beispiel 2 demonstriert verschachtelte `article`-Elemente in Form von Benutzerkommentaren über das übergeordnete `article`. Das kennen Sie aus dem Kommentarbereich von Blogs oder News-Sites. Es zeigt auch eine Verwendung des `section`-Elements (siehe „Einen Abschnitt (section) definieren“ und das in Kapitel 4 besprochene `time`-Element.

Beispiel 2 (verschachtelte Artikel):

```
<article>
   <h1>Die Vielfalt von Papua-Neuguinea</h1>
   ... [Inhalt des übergeordneten article] ...

   <footer>
      ... [Footer des übergeordneten article] ...
   </footer>

   <section>
      <h2>Leserkommentare</h2>
      <article>
         <footer>travelgal schrieb am <time datetime="2011-11-17"
         → pubdate>17. November 2011</time>:</footer>
         <p>Toller Artikel! Ich war schon immer neugierig auf Papua-Neuguinea.</p>
      </article>
      <article>
         ... [nächster Leserkommentar] ...
      </article>
   </section>
</article>
```

Dies sind nur ein paar übliche Beispiele, wie man `article` und dessen begleitende Elemente einsetzen kann.

Einen Abschnitt (section) definieren

Das `article`-Element hat einen semantisch weniger spezifischen Cousin namens `section` – auch dies ist eine Neuerung in HTML5.

`section` wird in HTML5 so definiert:

> Das `section`-Element repräsentiert einen generischen Bereich eines Dokuments oder einer Anwendung. Mit Bereich ist in diesem Kontext eine thematische Gruppierung von Inhalten gemeint, üblicherweise zusammen mit einer Überschrift A.
>
> Beispiele für solche Bereiche wären Kapitel, die verschiedenen, mit Reitern (Tabs) versehenen Seiten in einer Dialogbox mit Tabs oder die nummerierten Abschnitte einer Hausarbeit. Die Homepage einer Website kann in die Bereiche Einleitung, News und Kontaktinformationen aufgeteilt werden.

Die Elemente `article` und `section` sind sich recht ähnlich. Wenn Sie nicht ganz sicher sind, wie beide zu unterscheiden sind, lesen Sie den Kasten „Wie unterscheiden Sie zwischen `article` und `section`?“.

Einen Abschnitt definieren

1. Tippen Sie **`<section>`**.
2. Geben Sie die Inhalte des Abschnitts ein – eine beliebige Anzahl von Elementen wie Absätze, Listen, Audio, Video, Fotos, Abbildungen usw.
3. Tippen Sie **`</section>`**.

A *Der Code ist der gleiche wie vorher, außer dass ich beide Abschnitte des `article`, der auf die Einführung folgt, mit `section` eingefasst habe. Auch dieser Code wurde aus Platzgründen gekürzt.*

```
...
<body>
<header>
   <nav role="navigation">
      ... [ul mit Links] ...
   </nav>
</header>

<article>
   <h1 id="gaudi">Der Architekt von Barcelona
   → </h1>

   <p>Viele Touristen kommen extra nach
   → Barcelona, um sich die unbeschreibliche
   → Architektur von Antoni Gaudí anzuschauen.
   → </p>

   ... [noch ein Absatz aus der Einleitung] ...

   <section>
      <h2 id="sagrada-familia" lang="es">La
      → Sagrada Família</h2>

      <p><img src="img/towers.jpg" width="75"
      → height="100" alt="Türme der Sagrada
      → Família" />Die unvollendete Kirche mit
      → dem eigenartigen Namen Sühnekirche der
      → Heiligen Familie ist das am meisten
      → besuchte Gebäude in Barcelona. Darin
      → kombiniert Gaudí seine Vision von Natur
      → und Architektur mit seiner Hingabe an
      → seinen Glauben. Die Sagrada Família zieht
      → auch nicht religiöse Menschen an, was
      → zum großen Teil an dieser tragischen
      → Geschichte liegt und dem immer
      → noch unvollendeten Zustand, der durch
      → die allgegenwärtigen Gerüste und
      → Baukräne sehr deutlich wird.</p>
   </section>

   <section>
      <h2 id="park-guell">Park Güell</h2>

      ... [noch ein Bild und weitere Absätze] ...
   </section>
</article>
</body>
</html>
```

B *Nun besitzt die Seite `header`-, `nav`-, `article`- und `section`-Elemente und auch die entsprechenden Inhalte dafür. Die Standarddarstellung ist genauso wie vor dem Einfügen der `section`-Elemente.*

TIPP Wie Sie im Abschnitt „Die Gliederung eines HTML5-Dokuments“ erfahren haben, ist `section` neben `nav`, `article` und `aside` eines der vier Elemente, die die Seite in Bereiche aufteilen kann.

TIPP Standardmäßig werden Sie sicher keinen Unterschied erkennen, wenn Sie eine Seite mit `sections` (bzw. `articles`) ansehen. Wichtig ist hier aber, dass Sie die Semantik Ihres Dokuments stärken, wenn Sie sie einsetzen B. Natürlich können Sie die Elemente `section` und `article` nach Belieben mit CSS formatieren.

TIPP Behalten Sie im Hinterkopf, dass `section` kein generischer Container wie `div` ist, weil `section` immer eine bestimmte Bedeutung vermittelt und `div` hingegen gar keine semantische Aussagekraft hat (siehe „Generische Container erstellen“).

TIPP Es gibt mehrere Beispiele in diesem Kapitel, die Ihnen helfen, ein Gefühl dafür zu bekommen, wie man sowohl `article` als auch `section` auf vielerlei Art verwendet.

TIPP Über die Verwendung von `role="main"` mit `section` unter bestimmten Umständen erfahren Sie mehr im Abschnitt „Mit ARIA die Zugänglichkeit verbessern“.

Wie unterscheiden Sie zwischen article und section?

Ich habe absichtlich die Definition von HTML5 von `section` und `article` zitiert (siehe „Einen Artikel erstellen“), damit Sie die Abgrenzung nachvollziehen können, weil sie manchmal sehr subtil ist. Gehen Sie wie folgt vor: Ist Ihr Inhalt ein unabhängiges Stück Inhalt, das sich für eine Syndizierung eignet? Falls ja, nehmen Sie `article`. (Anderenfalls nehmen Sie in den meisten Fällen `section`, und lesen Sie „Generische Container erstellen“, um zu erfahren, wann man doch besser `div` nimmt.) Das bedeutet nicht, dass Sie eine Syndizierung vornehmen *müssen* oder Inhalte von `article` auf andere Weise zu veröffentlichen haben, sondern nur, dass sich der Inhalt dafür eignet.

Wenn Sie immer noch der Ansicht sind, dass sich `article` und `section` gelegentlich sehr ähneln – keine Sorge, das geht auch anderen so. Auch erfahrene Entwickler setzen diese beiden Elemente gelegentlich unterschiedlich ein.

Wie bereits in Kapitel 1 erwähnt, gibt es nicht *immer* die eine richtige Wahl, wenn es um das Auszeichnen von Inhalten geht, sondern nur *meist*. Bei den anderen Gelegenheiten läuft es auf eine persönliche Entscheidung hinaus, welche HTML-Elemente Ihres Erachtens den Inhalt am besten beschreiben.

Also überlegen Sie sich gut, wenn Sie zwischen `article` und `section` wählen, aber zerbrechen Sie sich nicht jedes Mal den Kopf, ob Sie diese Elemente immer exakt richtig verwenden. Manchmal ist das auch eine subjektive Entscheidung und in jedem Fall wird Ihre Seite auch weiterhin funktionieren. Außerdem wird Sie niemand deswegen mitten in der Nacht herausklingeln.

Tja, ich vielleicht schon, aber dann auch nur, weil es draußen dunkel und unheimlich ist.

Ein Beispiel für section ohne article

Bisher haben Sie Beispiele kennengelernt, wo `section` in einem `article` verschachtelt war (A). Das ist nur eine Einsatzmöglichkeit für dieses Element.

Das folgende Beispiel stammt leicht verändert aus der HTML5-Spezifikation: Hier wird `section` ohne `article` verwendet. (Sie bekommen auch schon mal einen Eindruck von nummerierten Listen. Mehr über `ol` und die anderen Listenelemente lernen Sie in Kapitel 15.)

```
...
<body>
   <h1>Programm der Abschlussfeier</h1>

   <section>
      <h2>Zeremonie</h2>
      <ol>
         <li>Eröffnungsprozession</li>
         <li>Rede des Jahrgangsbesten</li>
         <li>Rede des Klassensprechers</li>
         <li>Präsentation der Diplome</li>
         <li>Abschiedsansprache des Rektors</li>
      </ol>
   </section>

   <section>
      <h2>Absolventen (alphabetisch)</h2>
      <ol>
         <li>Molly Carpenter</li>
         <li>Anastasia Luccio</li>
         <li>Ebenezar McCoy</li>
         <li>Karrin Murphy</li>
         <li>Thomas Raith</li>
         <li>Susan Rodriguez</li>
      </ol>
   </section>
</body>
</html>
```

Ein aside angeben

Manchmal haben Sie bestimmte Inhalte, die zum Hauptinhalt Ihrer Seite eine Beziehung haben, aber auch eigenständig sind A. Wie können Sie das semantisch verdeutlichen?

A *Dieses `aside` stellt Informationen über die architektonischen Sehenswürdigkeiten von Barcelona vor und hängt mit den Inhalten über Antoni Gaudí zusammen, die den Schwerpunkt der Seite bilden. Aber sie könnten auch für sich stehen. Ich hätte das auch im `article` verschachteln können, weil beides zusammenhängt, aber ich habe beschlossen, es hinter dem `article` aufzuführen, damit ich es später visuell als Sidebar behandeln kann C. Das `role="complementary"` für das `aside` ist optional, kann aber die Zugänglichkeit verbessern. Weitere Infos im letzten Tipp.*

```
...
<body>

<header>
   <nav role="navigation">
      ... [ul mit Links] ...
   </nav>
</header>

<article>
   <h1 id="gaudi">Der Architekt von Barcelona</h1>
   ... [einleitende Absätze] ...

   <section>
      <h2 id="sagrada-familia" lang="es">La Sagrada Família</h2>
       ... [Bild und Absatz] ...
   </section>

   <section>
      <h2 id="park-guell">Park Güell</h2>
      ... [weiteres Bild und mehr Absätze] ...
   </section>
</article>

<aside role="complementary">
   <h1>Die architektonischen Wunder von Barcelona</h1>

   <p>Neben den Werken von Gaudí finden sich in Barcelona noch viele weitere architektonische Wunder. Dazu
   → gehören unter anderem:</p>
   <ul>
      <li lang="es">Arc de Triomf</li>
      <li>Die Kathedrale <span lang="es">(La Seu)</span></li>
      <li lang="es">Gran Teatre del Liceu</li>
      <li lang="es">Pavilion Mies van der Rohe</li>
      <li lang="es">Santa Maria del Mar</li>
   </ul>

   <p>Quelle: <a href="http://www.barcelona.de/en/barcelona-architecture-buildings.html"
   → rel="external"><cite>Barcelona.de</cite></a>.</p>
</aside>

</body>
</html>
```

Erst seit HTML5 kann man das explizit angeben, und zwar mit dem `aside`-Element B.

Üblicherweise stellt man sich das `aside` wie eine Sidebar (Seitenleiste) vor C, aber man kann je nach Kontext ein `aside`-Element auch an andere Orte auf der Seite stellen. Es kann sich um einen Kasten (entweder vom Konzept her oder buchstäblich) innerhalb des Hauptinhalts selbst handeln, in der gleichen Spalte, aber nicht im Hauptinhalt verschachtelt, oder in einer (bzw. als eine) sekundäre Spalte wie eine Sidebar. Ein `aside` (wörtlich: Nebenbemerkung) kann ein hervorgehobenes Zitat sein, eine Seitenleiste, ein Kasten mit zum Artikel gehörigen Links auf einer News-Site, Gruppen von `nav`-Elementen (z.B. eine Blogroll), ein Twitter-Feed oder eine Liste von passenden Produkten auf einer kommerziellen Site.

B *Das `aside` erscheint unterhalb des Artikels, weil es im HTML auch danach kommt* A*. Wie Sie sehen, formatieren Browser ein `aside` standardmäßig nicht irgendwie speziell (außer dass es in einer eigenen Zeile beginnt). Sie können dessen Erscheinungsbild jedoch komplett mit CSS steuern* C.

A *Wenn Sie die fertige Seite mit CSS formatieren, können Sie das* `aside` *(darin steht „Die architektonischen Wunder von Barcelona“) neben dem Hauptinhalt erscheinen lassen statt darunter. Also haben Sie in diesem Fall das* `aside` *wie eine Sidebar eingesetzt (Sie erfahren in Kapitel 11, wie Sie ein zweispaltiges CSS-Layout erstellen).*

Ein aside erstellen

1. Tippen Sie `<aside>`.
2. Geben Sie die Inhalte des `aside` ein – eine beliebige Anzahl von Elementen wie Absätze, Listen, Audio, Video, Fotos, Abbildungen usw.
3. Tippen Sie `</aside>`.

TIPP Obwohl mit einem `aside` auch Inhalte in einer Sidebar ausgezeichnet werden können C, wirkt sich das `aside`-Element nicht auf das Layout der Seite aus B.

TIPP Wenn Sie ein oder mehrere `aside`s als Sidebar oder in einer Sidebar verwenden, platzieren Sie den Inhalt der Sidebar im HTML nach dem Hauptinhalt der Seite A. Für SEO und Zugänglichkeit ist es besser, den wichtigsten Inhalt zuerst aufzuführen. Sie können mit CSS die Reihenfolge ändern, wie sie im Browser dargestellt werden.

TIPP Nehmen Sie das `figure`-Element (siehe Kapitel 4) und kein `aside`, um Abbildungen auszuzeichnen, die zum Inhalt gehören, z.B. ein Diagramm, ein Schaubild oder ein eingefügtes Foto mit Bildunterschrift.

TIPP HTML5 verbietet das Verschachteln eines `aside` innerhalb eines `address`-Elements.

TIPP Über die Verwendung von `role="complementary"` mit `aside` erfahren Sie mehr im Abschnitt „Mit ARIA die Zugänglichkeit verbessern“.

Andere Beispiele mit aside

Wie bereits erwähnt kann `aside` in der gleichen Spalte wie der Hauptinhalt erscheinen oder in Hauptinhalt verschachtelt sein oder zu einer Sidebar werden.

Beispiel 1 zeigt ein `aside`, das mit dem dazugehörigen Inhalt verschachtelt ist.

Beispiel 1 (verschachtelt im Hauptinhalt):

```
...
<body>
<article>
   <h1>Die Vielfalt von Papua-Neuguinea</h1>
   ... [Inhalt des Artikels] ...
   <aside>
      <h1>Kurzinfos über Papua-Neuguinea</h1>
      <ul>
         <li>In diesem Land gibt es 38 Arten der 43 bekannten Paradiesvögel</li>
         <li>Obwohl das Land in einigen Regionen recht tropisch ist, gibt es in anderen doch
         → gelegentlich Schneefälle.</li>
         ...
      </ul>
   </aside>
   ... [weiterer Artikelinhalt] ...
</article>
</body>
</html>
```

Das gleiche `article` kann z.B. ein Zitat aus dem Artikeltext hervorheben. Das wäre dann auch ein `aside`. Oder es könnte ein `aside` mit „Verwandte Storys“ geben, in dem eine Liste mit Links auf andere Essays über das Land oder Regionen dieser Ecke der Welt enthalten ist (z.B. Indonesien, Australien usw.). Alternativ könnte dieses `aside` auch in einer anderen Seitenspalte stehen statt verschachtelt im `article`.

Sie haben bereits ein Beispiel eines solchen `aside` in einer Sidebar gesehen (A und C). Nun wollen wir uns als Beispiel einmal ein Design-Portfolio oder eine Gruppe Fallstudien anschauen, bei dem bzw. der jede HTML-Seite sich jeweils auf ein Projekt konzentriert. Sie bieten dann Links (die in einem `nav` verschachtelt sind) auf andere Projektseiten in einer benachbarten Spalte an (die von CSS und nicht einfach durch die Anordnung des Codes wie in Beispiel 2 gesteuert wird).

Beispiel 2 (`aside` ist nicht im Hauptinhalt verschachtelt und enthält ein `nav`):

```
...
<body>
<!-- Hauptinhalt der Seite -->
<article>
   <h1>... [Name des Projekts] ...</h1>
   <figure>... [Projektfoto] ...</figure>
   <p>... [Projektrezension] ...</p>
</article>

<!-- Dieses aside ist nicht im article verschachtelt -->
<aside>
   <h1>Andere Projekte</h1>
   <nav>
      <ul>
         <li><a href="habitat-for-humanity.html">Broschüre über Habitat for Humanity</a></li>
         <li><a href="royal-philharmonic.html">Site des Royal Philharmonic Orchestra</a></li>
         ...
      </ul>
   </nav>
</aside>
</body>
</html>
```

Hier wäre es absolut angemessen, wenn man gerade diese Nebenbemerkung auch im Projektartikel verschachtelt, weil beide sich aufeinander beziehen.

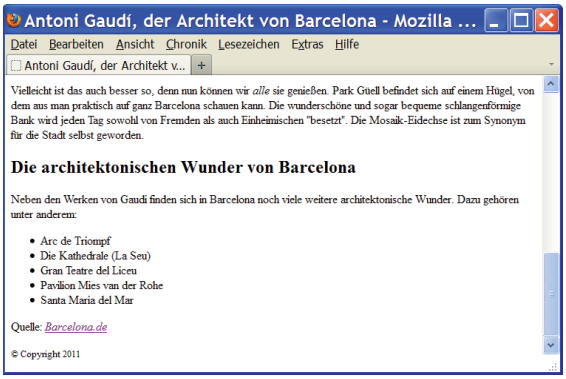
Einen Footer erstellen

Wenn es um einen Footer oder eine Fußzeile geht, dann denken Sie wahrscheinlich an die Fußzeile auf einer Webseite. Das `footer`-Element von HTML5 passt dafür, aber so wie `header` können Sie es auch woanders einsetzen.

Das `footer`-Element repräsentiert einen Footer für das nächstgelegene Element (z.B. `article`, `aside`, `blockquote`, `body`, `details`, `fieldset`, `figure`, `nav`, `section` oder `td`), in dem es verschachtelt ist. Es ist nur dann der Footer für die *ganze* Seite, wenn der nächste Vorfahr der `body` ist (A und B). Und wenn ein `footer` den *gesamten* Inhalt seines Abschnitts einfasst (z.B. ein `article`), dann repräsentiert er je nach Inhalt so etwas wie einen Anhang, einen Index, einen langen Kolophon oder ein langes Lizenzabkommen.

A *Dieser `footer` repräsentiert den Footer für die ganze Seite, weil der nächste Vorfahr das `body`-Element ist. Nun besitzt die Seite `header`-, `nav`-, `article`-, `section`-, `aside`- und `footer`-Elemente. Nicht jede Seite braucht das alles, aber sie repräsentieren die in HTML verfügbaren primären Seitenkonstrukte.*

```
...
<body>
<header>
   <nav role="navigation">
      ... [ul mit Links] ...
   </nav>
</header>

<article>
   <h1 id="gaudi">Der Architekt von Barcelona
   → </h1>
   ... [einleitende Absätze] ...

   <section>
      <h2 id="sagrada-familia" lang="es">La
      → Sagrada Família</h2>
      → ... [Bild und Absatz] ...
   </section>

   <section>
      <h2 id="park-guell">Park Güell</h2>
      ... [weiteres Bild und mehr Absätze] ...
   </section>
</article>

<aside role="complementary">

   <h1>Die architektonischen Wunder von
   → Barcelona</h1>
   ... [restlicher Inhalt des aside]
</aside>

<footer>
   <p><small>&copy; Copyright 2011</small></p>
</footer>

</body>
</html>
```



B *Das `footer`-Element selbst zwingt dem Text standardmäßig keine Formatierung auf. Hier ist der Copyright-Hinweis kleiner als normaler Text, weil es in einem `small`-Element verschachtelt ist, um semantisch das Kleingedruckte zu repräsentieren (siehe Kapitel 4). Wie bei allem anderen auch können Sie die Schriftgröße mit CSS ändern.*

Eine Fußzeile erstellen

1. Platzieren Sie den Cursor innerhalb des Elements, für das Sie einen Footer erstellen wollen.
2. Tippen Sie **`<footer>`**.
3. Geben Sie den Inhalt des Footer ein.
4. Tippen Sie **`</footer>`**.

TIPP Ein `footer` enthält üblicherweise Informationen über seinen Abschnitt, z.B. Links zu verwandten Dokumenten, Copyright-Informationen, den Autor und ähnliche Elemente. Siehe die Beispiele 1 und 2 im Kasten „Andere `footer`-Beispiele“.

TIPP Ein `footer` muss nicht unbedingt am Ende des Elements stehen, in dem es enthalten ist (es ist aber meistens so).

TIPP Es ist nicht erlaubt, in einem `footer` einen `header` oder einen anderen `footer` zu verschachteln. Außerdem können Sie keinen `footer` innerhalb eines `header`- oder `address`-Elements verschachteln.

TIPP In „Generische Container erstellen“ erfahren Sie mehr, wie durch `footer` eine der Rollen des `div`-Elements aus der Zeit vor HTML5 ersetzt wurde.

TIPP Über die Verwendung von `role="contentinfo"` mit `footer` unter bestimmten Umständen erfahren Sie mehr im Abschnitt „Mit ARIA die Zugänglichkeit verbessern“. Es wäre auch passend, ihn beim `footer` in A aufzunehmen, weil es den Footer für die ganze Seite repräsentiert. Aber ich habe es da weggelassen, um nicht den Eindruck zu erwecken, dass `role="contentinfo"` für alle `footer`-Elemente richtig ist. In „Andere `footer`-Beispiele“ sehen Sie ein Beispiel, das sowohl den Unterschied zeigt als auch die Rolle korrekt anwendet.

Andere footer-Beispiele

Sie haben ein kleines Beispiel gesehen, wo ein Footer für die ganze Seite gilt (A und B). Hier ist ein anderer Seiten-Footer, aber mit mehr Inhalt.

Beispiel 1 (als Seiten-Footer):

```
...
<body>
... [Seitenkopf und Inhalt] ...

<!-- Dies ist ein Seiten-Footer, weil der nächste Vorfahr body ist -->
<footer role="contentinfo">
   <p><small>&copy; Copyright 2011 The Corporation, Inc.</small></p>

   <ul>
      <li><a href="terms-of-use.html">Nutzungsbedingungen</a></li>
      <li><a href="privacy-policy.html">Datenschutzbestimmungen</a></li>
   </ul>
</footer>
</body>
</html>
```

Das nächste Beispiel demonstriert einen `footer` im Kontext eines Seitenabschnitts (in diesem Fall ein `article`) und einen zweiten `footer` für die ganze Seite (in „Mehr `article`-Beispiele“ wird erklärt, welchen Geltungsbereich das `address`-Element hier hat).

Beispiel 2 (ein Footer für einen Seitenabschnitt und die ganze Seite):

```
...
<body>
...
<article>
   <h1>... [Artikelüberschrift] ...</h1>
   <p>... [Artikelinhalt] ...</p>

   <!-- Der footer des article -->
   <footer>
      <p>Leandra Allen ist eine freiberufliche Journalistin, die ihren Abschluss in Anthropologie
      → an der Universität Kopenhagen gemacht hat.</p>
      <address>
      Sie erreichen sie unter <a href="mailto: leandra@therunningwriter.com">leandra@
      → therunningwriter.com</a>.
      </address>
   </footer>
</article>

<!-- der Footer der Seite -->
<footer id="footer-page" role="contentinfo">
   ... [Copyright, Nutzungsbedingungen, Datenschutzbestimmungen] ...
</footer>
</body>
</html>
```

Die `id="footer-page"` (Sie können jede gültige `id` angeben) im Seiten-Footer ist optional und dient nur dazu, sie zur Steuerung der Formatierung vom anderen `footer` unterscheiden zu können. Beachten Sie, dass nur der Seiten-`footer` die optionale `role="contentinfo"` bekommt. Über diese Rolle erfahren Sie mehr im Abschnitt „Mit ARIA die Zugänglichkeit verbessern“.

Generische Container erstellen

Manchmal müssen Sie ein Inhaltssegment in einen Container legen, weil Sie CSS-Formatierungen anwenden oder einen Effekt mit JavaScript erzielen wollen. Ihre Seite wäre ohne das einfach nicht die gleiche A. Aber wenn Sie sich den Inhalt genau anschauen, kommen Sie zu der Beurteilung, dass solche Elemente wie `article`, `section`, `aside` oder `nav` semantisch nicht angemessen wären. Sie brauchen eigentlich einen generischen Container, also einen ohne jegliche semantische Bedeutung. Dieser Container ist das `div`-Element (steht für *division*, also Abschnitt) B. Über dieses `div` können Sie die gewünschte Formatierung oder den gewollten JavaScript-Effekt anwenden. Lesen Sie den Kasten zu diesem Thema, in dem steht, wann Sie auf Ihren Seiten mit `div` arbeiten sollten.

A *Dieses Design habe ich ohne jegliches `div`-Element auf der Seite erzielt. Doch indem ich ein `div` um den gesamten Inhalt der Seite gelegt habe* B*, steht mir nun ein generischer Container zur Verfügung, den ich formatieren kann (siehe die Ergebnisse in* C*).*

B *Nun fasst ein `div` den gesamten Inhalt ein. Die Semantik der Seite bleibt unverändert, aber ich habe einen generischen Container, den ich mit CSS formatieren kann* C*.*

```
...
<body>

<div>
   <header>
      <nav role="navigation">
         ... [ul mit Links] ...
      </nav>
   </header>

   <article>
      <h1 id="gaudi">Der Architekt von
      → Barcelona</h1>
      ... [einleitende Absätze] ...

      <section>
         ... [Überschrift, Bild und Absatz] ...
      </section>

      <section>
         ... [Überschrift, weiteres Bild und mehr
         → Absätze] ...
      </section>
   </article>

   <aside role="complementary">
      <h1>Die architektonischen Wunder von
      → Barcelona</h1>
      ... [restlicher Inhalt des aside]
   </aside>

   <footer>
      ... [Copyright] ...
   </footer>
</div>

</body>
</html>
```

C *Ein `div`-Element hat standardmäßig keine eigene Formatierung, außer dass es in einer neuen Zeile beginnt. Doch Sie können `div` formatieren, um Ihre Designs zu implementieren. Hier habe ich einen hellblauen Hintergrund und einen Schatten ins `div` eingefügt. So kann ich den Hintergrund des `body`-Elements in Violett ändern, um den Inhalt hervorzuheben. Dem `aside` habe ich auch einen dünnen Rahmen gegeben. Wie ich das gemacht habe, können Sie dem HTML und CSS dieser Seite unter www.bruceontheloose.com/htmlcss/examples/ entnehmen.*

D *Das ist die gleiche Seite, aber weder für das `div` noch für Überschriften, Absätze oder andere Elemente ist CSS angewendet worden. Wie Sie sehen, schafft das `div` auf sich allein gestellt es nicht, die Seite schicker aussehen lassen.*

Einen generischen Container erstellen

1. Tippen Sie `<div>`.
2. Erstellen Sie die Inhalte des Containers mit beliebig vielen Elementen.
3. Am Ende des Containers tippen Sie `</div>` ein.

TIPP **Wie `header`, `footer`, `article`, `section`, `aside`, `nav`, `h1-h6`, `p` und viele andere Elemente auch wird `div` standardmäßig automatisch in eine neue Zeile gestellt.**

TIPP **`div` ist ebenfalls praktisch, wenn man bestimmte Interaktionen oder Effekte mit JavaScript einbauen will. Damit können Sie beispielsweise ein Foto oder eine Dialogbox mit einem halbtransparenten „Deckblatt“ darstellen, das die Seite abdeckt (dieses Deckblatt ist üblicherweise ein `div`).**

TIPP **Ich betone zwar dauernd, dass HTML die Bedeutung Ihrer Inhalte beschreibt, aber `div` ist das einzige Element, das keinen semantischen Wert hat. Das `span`-Element ist das Gegenstück zu `div`. Während `div` ein Container ohne Semantik für inhaltliche Blöcke ist, ist `span` (das wird `<span>hier kommt der Inhalt</span>` geschrieben) nur für Text gedacht, z.B. innerhalb eines `p`-Elements für Absätze. Mehr über `span` erfahren Sie in Kapitel 4.**

TIPP **Wie Sie Landmark-Rollen mit `div` verwenden, erfahren Sie im Abschnitt „Mit ARIA die Zugänglichkeit verbessern“.**

Ein paar historische Angaben über div und wie man es in HTML5 einsetzt

Von den in diesem Kapitel vorgestellten strukturellen Elementen ist `div` neben `h1-h6` das einzige, das noch aus den Zeiten vor HTML5 stammt. Bis zum Erscheinen von HTML5 war `div` die De-facto-Wahl, um inhaltlich zusammengehörige Blöcke wie Header, Footer, Hauptinhalt, Einfügungen und Sidebars einer Seite einzufassen, damit man diese alle mit CSS formatieren kann. Aber damals wie heute hatte `div` keine semantische Bedeutung.

Darum wurden in HTML5 `header`, `footer`, `article`, `section`, `aside` und `nav` eingeführt. Diese Typen von Grundbausteinen waren auf Webseiten derart verbreitet, dass sie ihre eigenen Elemente *mit* Bedeutung verdienten. `div` ist in HTML5 nun nicht untergegangen, aber es gibt einfach weniger Gelegenheiten als früher, es einzusetzen.

Schauen wir uns einige übliche Instanzen an, bei denen `div` die richtige Wahl ist.

Eine kennen Sie bereits: wenn man eine ganze Seite zu Formatierungszwecken mit einem Container einfassen will (B und C).

Wie habe ich das zweispaltige Layout mit `div` hinbekommen? Auf das `article`-Element habe ich CSS angewendet, damit dieses als Spalte 1 und das `aside`-Element als Spalte 2 dargestellt wird. (In Kapitel 7 geht's dann mit dem CSS richtig los und in Kapitel 11 erfahren Sie mehr über Layouts mit CSS.)

Meistens (wenn nicht gar immer) haben Spalten allerdings mehr als nur einen inhaltlichen Bereich. Vielleicht soll beispielsweise ein weiteres `article`-Element (oder `section` oder `aside` usw.) im Hauptbereich unterhalb des ersten `article` stehen. Und vielleicht möchten Sie in der zweiten Spalte ja noch ein weiteres `aside` unterbringen, das z.B. eine Liste mit Links zu anderen Sites über Gaudí enthält. Oder in diese Spalte soll gar eine ganz andere Art Element eingefügt werden.

Sie müssen dafür die Inhalte, die in jeder Spalte repräsentiert werden, in einem `div` gruppieren E und dieses `div` dann entsprechend formatieren. (Falls Sie auf den Gedanken kommen, dass auch `section` dafür eine Option wäre: Dies ist nicht als generischer Container fürs Formatieren gedacht.) Ich stelle Ihnen hier ein Diagramm F vor, das Ihnen helfen soll, die Beziehung zwischen dem Code E und dem möglichen CSS-Layout zu visualisieren. Bedenken Sie aber, dass es sich dabei nur um eine Layoutmöglichkeit für dieses HTML handelt: CSS ist sehr leistungsfähig.

Es ist also recht üblich, dass ein `div` um jede Gruppe von Inhalten steht, die Sie als eine Spalte formatieren wollen (natürlich können Sie auch mehr als zwei nehmen). Was dann da jeweils hineinkommt, kann – basierend auf den jeweiligen Inhalten für Ihre Seite – sehr breit variieren. Vergessen Sie nicht, dass `article`, `section`, `aside` und `nav` als Ihre primären Container für Inhaltsbereiche praktisch überall auftauchen können. Und in diesem Kapitel haben Sie auch gelernt, dass das auch für `header` und `footer` gilt. Lesen Sie nicht zu viel in die Tatsache hinein, dass das Beispiel (E und F) nur `article`s im Hauptinhaltsbereich zeigt und in der Sidebar `aside`s stehen.

Um da allerdings sicher zu sein, sollte `div` Ihre letzte Zuflucht für einen Container sein, weil es keinen semantischen Wert hat. Meist ist es richtig, stattdessen so etwas wie `header`, `footer`, `article`, `section`, `aside` und vielleicht noch `nav` zu verwenden. Doch Sie sollten diese *nicht* nehmen, bloß um `div` zu vermeiden, wenn das semantisch unpassend ist. `div` hat seinen Platz, Sie sollten es einfach nur beschränkt nutzen.

Nachdem das nun geklärt ist, soll noch auf eine Situation verwiesen werden, in der es gerechtfertigt ist, statt der neuen HTML5-Elemente `div` für alle Container (oder die meisten, das liegt an Ihnen) auf einer Seite zu nehmen. Mehr Infos darüber finden Sie im Abschnitt „HTML5-Elemente in älteren Browsern formatieren“ in Kapitel 11.

E *Diese Seite hat ein `div`, das die ganze Seite beinhaltet, plus zwei neue. Ein `div` mit `id="content"` gruppiert den Hauptinhalt, damit er als Spalte eins formatiert werden kann. Ein anderes `div` mit `id="sidebar"` umgibt den Inhalt, den Sie als Spalte zwei darstellen wollen. Dann nehmen Sie die `id` in Ihrem CSS, um jedes spezifische `div` präzise formatieren zu können.*

```
...
<body>

<!-- Start Seiten-Container -->
<div id="container">
   <header>
      ...
   </header>

   <!-- Spalte eins, wenn CSS angewendet wird -->
   <div id="content">
      <article>
         ...
      </article>

      <article>
         ...
      </article>

      ... [weitere Abschnitte nach Bedarf] ...
   </div>
   <!-- Ende Spalte eins -->

   <!-- Spalte zwei, wenn CSS angewendet wird -->
   <div id="sidebar">
      <aside>
         ...
      </aside>

      <aside>
         ...
      </aside>

      ... [weitere Abschnitte nach Bedarf] ...
   </div>
   <!-- Ende Spalte zwei -->

   <footer>
      ...
   </footer>
</div>
<!-- Ende Seiten-Container -->

</body>
</html>
```

F *Dieses Diagramm illustriert, wie der Code in* E *zu einem CSS-Layoutkonzept passen kann. Zwar ist dies ein sehr weit verbreitetes Arrangement, aber auch nur eine der vielen Möglichkeiten, die Ihnen mit dem gleichen HTML zur Verfügung stehen. Werfen Sie einen Blick in den nächsten Abschnitt „Mit ARIA die Zugänglichkeit verbessern" – hier erfahren Sie, wie man Semantik und Zugänglichkeit Ihrer Seiten noch weiter optimiert.*

Mit ARIA die Zugänglichkeit verbessern

Wie Sie in Abschnitt „Warum Semantik wichtig ist“ aus Kapitel 1 gelernt haben, wird die Zugänglichkeit einfach dadurch verbessert, dass man den Inhalt mit dem HTML auszeichnet, das ihn am besten beschreibt. Falls Sie bereits so arbeiten, ist das großartig. In diesem Abschnitt erzähle ich Ihnen, wie Sie mit einigen einfachen Attributen im HTML Ihren Besuchern noch mehr helfen.

WAI-ARIA (Web Accessibility Initiative's Accessible Rich Internet Applications) oder kurz ARIA ist eine Spezifikation, die sich selbst zur „Brückentechnologie“ erklärt hat. Das heißt, sie füllt semantische Lücken mit Attributen, bis Sprachen wie HTML ihre eigenen semantischen Äquivalente gefunden haben.

Mit welchem HTML-Markup lassen Sie beispielsweise einen Screenreader wissen, wie er zum Hauptinhalt Ihrer Seite (oder daran vorbei) wechseln kann? Oder zu einem Suchfeld? Wie Sie gleich erfahren, gibt es Überlappungen zwischen ARIA und HTML5 (Letzteres hat ebenfalls versucht, ein paar der Lücken zu füllen), aber nicht einmal HTML5 hat Lösungen für beides. Die *Landmark Roles* von ARIA hingegen sorgen dafür: Sie identifizieren eine Gruppe von Seitenbereichen vor allem zu diesem Zweck: `application`, `banner`, `complementary`, `contentinfo`, `form`, `main`, `navigation` und `search`.

Es gibt zwar Schnittpunkte zwischen Landmark Roles und HTML5-Elementen, aber die Screenreader-Unterstützung ist für ARIA weiter fortgeschritten. Also können Sie wie gehabt mit dem Erstellen von HTML weitermachen und ARIA-Roles einbauen, um die Zugänglichkeit Ihrer Seiten zu verbessern.

In Ⓐ habe ich ARIA Landmark Roles und ein `nav`-Element in das Beispiel aus „Generische Container erstellen“ eingebaut. Obwohl ich in jedes `aside`-Element eine `complementary`-Rolle platziert habe, wäre `<div id="sidebar" role="complementary">` genauso valider Code, mit dem stattdessen die ganze Sidebar markiert wäre. Bevor Sie bei Ihren Seiten so vorgehen, sollten Sie ganz sicher sein,

Ⓐ *Das Beispiel aus „Generische Container erstellen“ mit einem ergänzten nav-Element und fünf verschiedenen Landmark Roles*

```
...
<body>

<!-- Start Seiten-Container -->
<div id="container">
   <header role="banner">
      ...
      <nav role="navigation">
         ... [ul mit Links] ...
      </nav>
   </header>

   <!-- Spalte eins, wenn CSS angewendet wird -->
   <div id="content" role="main">
      <article>
         ...
      </article>

      <article>
         ...
      </article>

      ... [weitere Abschnitte nach Bedarf] ...
   </div>
   <!-- Ende Spalte eins -->

   <!-- Spalte zwei, wenn CSS angewendet wird -->
   <div id="sidebar">
      <aside role="complementary">
         ...
      </aside>

      <aside role="complementary">
         ...
      </aside>

      ... [weitere Abschnitte nach Bedarf] ...
   </div>
   <!-- Ende Spalte zwei -->

   <footer role="contentinfo">
      ...
   </footer>
</div>
<!-- Ende Seiten-Container -->

</body>
</html>
```

B *Dies ist das Layoutdiagramm aus „Generische Container erstellen“, aber nun enthält es die ARIA-Rollen. Wie angemerkt, könnte das Sidebar-div auch* `role="complementary"` *statt der* `aside`*-Elemente enthalten.*

dass alle Ihre `div`-Inhalte als `complementary`-Inhalt geeignet sind.

Hier folgen nun ein paar Definitionen für Landmark Roles aus der ARIA-Spezifikation, im Anschluss daran meine Empfehlungen zur Nutzung. Sie werden in A demonstriert und in einem Diagramm B, ähnlich wie das aus „Generische Container erstellen“.

- `role="banner"`

  Ein Bereich, der statt seitenspezifischer Inhalte vor allem solche Inhalte enthält, die an der Site insgesamt orientiert sind.

  Site-orientierte Inhalte bestehen üblicherweise aus solchen Dingen wie Logo oder Identität des Sponsors dieser Site und ein für die ganze Site gedachtes Suchwerkzeug. Ein Banner erscheint üblicherweise oben auf der Seite und erstreckt sich meist über die ganze Breite.

  Verwendung: Fügen Sie es auf dem für die Seite gedachten Masthead ein (meist ein `header`-Element) und nehmen Sie es pro Seite nur einmal.

- `role="navigation"`

  Eine Sammlung navigationsbezogener Elemente (meist Links), um im Dokument oder damit zusammenhängenden Dokumenten zu navigieren.

  Verwendung: Dies spiegelt das `nav`-Element von HTML5 wider. Also fügen Sie es in jedes `nav`-Element ein, und wenn keines vorhanden ist, legen Sie es in den Container, der Ihre Links umgibt. Sie können diese `role` mehr als einmal pro Seite verwenden.

- `role="main"`

  Der Hauptinhalt eines Dokuments.

  Verwendung: Fügen Sie es in den Container Ihres Hauptinhalts ein. Oft wird es sich dabei um ein `div`-Element handeln, aber es kann auch ein `article` oder `section` sein. Außer in ganz seltenen Umständen sollte Ihre Seite nur über einen Bereich verfügen, der mit `main` ausgezeichnet ist.

- `role="complementary"`

  Ein ergänzender Abschnitt des Dokuments, der als komplementär zum Hauptinhalt gedacht ist . . . aber auch dann noch aussagekräftig ist, wenn er vom Hauptinhalt getrennt wird.

  Die komplementäre Rolle zeigt an, dass der Inhalt darin für den Hauptinhalt relevant ist.

  Verwendung: Dies spiegelt das `aside`-Element von HTML5 wider. Also fügen Sie es in ein `aside` oder ein `div` ein, das den gesamten komplementären Inhalt enthält. Sie können mehr als eine `complementary`-Rolle auf jeder Seite einbauen.

- `role="contentinfo"`

  Eine große, wahrnehmbare Region, die Informationen über das übergeordnete Dokument enthält.

  Zu den Beispielen in diesem Bereich der Seite gehören Copyright-Angaben und Links auf Datenschutzbestimmungen.

  Verwendung: Fügen Sie diese einmal in den Footer auf Seitenebene ein (üblicherweise ein `footer`-Element).

Zusammengefasst ist es generell eine gute Idee, ARIA Landmark Roles in Ihr HTML einzubauen. Ich habe sie bei verschiedenen anderen Beispielen des Buchs integriert wie auch auf der Site für das Buch. Um es deutlich zu sagen: Ihre Seiten funktionieren auch ohne sie, aber die Nutzererfahrung wird für bestimmte User verbessert. Gewiss finden Sie die Screenreader-Testergebnisse, die in den Tipps aufgeführt werden, hilfreich für die Entscheidung, ob Sie diese auch selbst nutzen sollten (diese Rollen werden durchgängig unterstützt, nur vom Screenreader Window Eyes 7.5 nicht).

TIPP **Die Rolle `form` ist semantisch redundant mit dem `form`-Element, `search` markiert ein Suchformular (BBC, Yahoo! und Google nutzen diese sowie auch in manchen Fällen andere Landmark Roles) und `application` ist für fortgeschrittene Anwendungsmöglichkeiten gedacht.**

TIPP **Landmark Roles sind nur eines der vielen Features der ARIA-Spezifikation (*www.w3.org/TR/wai-aria/*). Vielleicht interessiert Sie auch dieser Implementierungsleitfaden: *www.w3.org/WAI/PF/aria-practices/*.**

TIPP **Die Accessibility-Befürworter Steve Faulkner und Jason Kiss haben mehrere Tests über die Unterstützung von Landmark Roles durch Screenreader gepostet, und zwar unter *www.html5accessibility.com/tests/landmarks.html* bzw. *www.accessibleculture.org/research/html5-aria-2011/*. Siehe Faulkners damit zusammenhängende Diskussion unter *www.paciellogroup.com/blog/2011/11/latest-aria-landmark-support-data/* und unter *www.paciellogroup.com/blog/2011/07/html5-accessibility-chops-aria-landmark-support/*.**

TIPP **NVDA (Windows, kostenloser Download unter *www.nvda-project.org/*), VoiceOver (kostenloser Bestandteil von Mac OSX und iOS 4+) sowie JAWS (Windows, kostenlose Probeversion unter *www.freedomscientific.com/*) gehören zu den fortschrittlichsten erhältlichen Screenreadern. Ich kann gar nicht nachdrücklich genug dazu raten, dass Sie zumindest einen davon mal ausprobieren, um besser würdigen zu können, wie sich Ihre semantischen HTML-Entscheidungen auf die Nutzererfahrung bei Screenreadern auswirken. Besser wäre noch, wenn Sie Ihre Seiten als Teil Ihres normalen Entwicklungsprozesses mit einem Screenreader testen.**

TIPP **Sie können diese ARIA-Rollenattribute in Ihren CSS-Selektoren nutzen. Tatsächlich können Sie anhand der korrekten Landmark Roles die Attribute `id="content"` und `id="sidebar"` vom Code-Sample A weglassen. In Kapitel 11 erfahren Sie mehr.**

Das class-Attribut und Mikroformate

Es gibt das übliche Missverständnis, dass das `class`-Attribut einzig dafür erstellt wurde, um CSS auf Gruppen von Elementen anzuwenden. Das ist aber nicht der Fall. Es wurde auch deswegen designt, um die Semantik von HTML zu bereichern, ohne mehr Elemente in die Markup-Sprache einzuführen.

Mikroformate machen genau das: Sie arbeiten mit vereinbarten `class`-Namen, um ein Stück HTML z.B. als Ereignis oder Kalendereintrag zu identifizieren (das `hCalendar`-Mikroformat), um Personen, Organisationen und Firmen zu identifizieren (`hCard`) oder um die Beziehung zwischen Personen zu beschreiben (XFN). Das sind nur ein paar der vielen bis heute definierten Mikroformate, und weitere sind immer in Arbeit.

Anwendungen, Suchroboter und andere Software können die Mikroformate in Ihrem HTML lesen und nutzen. Das Firefox-Add-on *Operator* zeigt beispielsweise die Mikroformate einer beliebigen Seite an.

Über die Implementierung von Mikroformaten erfahren Sie mehr unter *http://microformats.org*.

Elemente mit einer Klasse oder ID benennen

Obwohl es nicht erforderlich ist, können Sie Ihren HTML-Elementen einen eindeutigen Identifikator geben bzw. sie einer bestimmten Klasse (oder mehreren Klassen) zuweisen oder beides. Danach können Sie diese Elemente je nach ihrem `id`- oder `class`-Namen formatieren. Das ist sicher die hauptsächliche, aber nicht die einzige Verwendung (siehe Tipps in diesem Abschnitt).

Ein Element mit einer eindeutigen id benennen

Innerhalb des Start-Tags des Elements tippen Sie **`id="name"`**, wobei **`name`** das Element eindeutig identifiziert (A). **`name`** kann praktisch alles sein, solange es nicht mit einer Zahl beginnt oder Leerzeichen enthält.

Einem Element eine Klasse zuweisen

Innerhalb des Start-Tags des Elements tippen Sie **`class="name"`**, wobei **`name`** der identifizierende Name der Klasse ist (A). Wenn Sie mehr als eine Klasse zuweisen wollen, trennen Sie diese mit einem Leerzeichen so wie hier: **`class="name anderername"`**. (Man kann mehr als zwei Klassennamen zuweisen.)

TIPP Jede `id` in einem HTML-Dokument muss eindeutig sein. Anders gesagt dürfen nicht zwei Elemente in der gleichen Seite mit der gleichen `id` bezeichnet werden und jedes Element darf nur eine `id` haben. Die gleiche `id` kann auf mehreren Seiten erscheinen und muss nicht jedes Mal dem gleichen Element zugewiesen werden, obwohl das normalerweise so gehandhabt wird.

TIPP Umgekehrt kann ein bestimmter `class`-Name einer beliebigen Zahl Elementen auf einer Seite zugewiesen werden und ein Element kann mehr als eine `class` besitzen.

A *Vergeben Sie ein eindeutiges Attribut `id` an ein Element, um es später formatieren oder darauf verlinken zu können oder ein JavaScript-Verhalten zuzuweisen. Geben Sie einer Gruppe von Elementen das Attribut `class`, um alle Elemente in dieser Gruppe auf einen Rutsch formatieren zu können. Die Klassen `architect` und `project` könnten zum Beispiel auf Inhalte über andere Architekten angewendet werden, um die Formatierung zu vereinheitlichen. Die Links im `nav` zeigen auf die `id`s der `h1`- und `h2`-Elemente (siehe Kapitel 6 für weitere Informationen über Links). Die anderen `id`s sind für die Formatierung. In „Generische Container erstellen“ finden Sie weitere Infos über `id`s sowie ein anderes Beispiel für deren Einsatz. Die Attribute `id` und `class` wirken sich nicht auf das Erscheinen eines Elements aus, außer sie werden im CSS referenziert.*

```
...
<body>

<div id="container">
   <header>
      <nav role="navigation">
         <ul id="toc">
            <li><a href="#gaudi">Der Architekt von Barcelona</a></li>
            <li><a href="#sagrada-familia" lang="es">La Sagrada Família</a></li>
            <li><a href="#park-guell">Park Güell</a></li>
         </ul>
      </nav>
   </header>

   <article class="architect" role="main">
      <h1 id="gaudi">Der Architekt von Barcelona</h1>

      <p>Viele Touristen kommen extra nach Barcelona, um sich die unbeschreibliche Architektur von Antoni
      → Gaudí anzuschauen.</p>
      ...

      <section class="project">
         <h2 id="sagrada-familia" lang="es">La Sagrada Família</h2>
         ...
      </section>

      <section class="project">
         <h2 id="park-guell">Park Güell</h2>
         ...
      </section>
   </article>
   ...
</div>
</body>
</html>
```

TIPP **Die Attribute `class` und `id` kann man beliebigen HTML-Elementen hinzufügen. Ein Element kann sowohl eine `id` als auch eine beliebige Anzahl von `class` haben.**

TIPP **Wie man Stylesheets auf ein Element mit einer bestimmten `id` oder `class` anwendet, erfahren Sie im Abschnitt „Elemente nach Klasse oder ID auswählen“ in Kapitel 9.**

TIPP **Wählen Sie aussagekräftige (d.h. semantische) Namen für Ihre `id` und `class`, egal wofür Sie sie nutzen wollen. Wenn Sie z.B. eine `class` zur Formatierung nutzen wollen, vermeiden Sie Namen, die die Präsentation beschreiben, also etwa `class="rot"`. Das wäre eine Todsünde. Im Ernst: `class="rot"` ist eine schlechte Wahl, weil Sie vielleicht in der nächsten Woche beschließen, das Farbschema Ihrer Site auf Blau umzustellen. Zwar ist es unglaublich einfach, mit CSS die einer `class` zugewiesene Farbe zu ändern, aber dann stünde in Ihrem HTML eine `class` namens `rot`, die in Wirklichkeit eine andere Farbe darstellt. Alle `class`-Namen im HTML zu ändern, ist meist nicht so simpel.**

TIPP **Wenn man wählen kann, ob man zur Formatierung eine `class` oder eine `id` auf ein Element anwendet, sollte man meist lieber eine `class` nehmen, weil man die damit verknüpften Formatierungen auf andere Elemente mit der gleichen `class` erneut verwenden kann. Doch es kann aber auch Situationen geben, bei denen Sie Ihre Formatierungen über dessen `id` auf ein Element (und vielleicht auch auf dessen Nachfahren) anwenden wollen.**

TIPP **Das Attribut `id` verwandelt das Element automatisch in einen benannten Anker (*anchor*), mit dem Sie eine direkte Verknüpfung (Link) vornehmen können. Weitere Details erfahren Sie im Abschnitt „Anker erstellen“ in Kapitel 6.**

TIPP **Sie können mit dem `class`-Attribut Mikroformate implementieren (weitere Details dazu im Kasten).**

TIPP **Sie können mit JavaScript auf die Attribute `id` und `class` zugreifen, um Verhalten für bestimmte Elemente anzuwenden.**

Wie man das title-Attribut auf Elemente anwendet

Sie können mit dem `title`-Attribut – nicht zu verwechseln mit dem `title`-*Element* – praktisch alle Bereiche Ihrer Website mit Tooltipps (eine Art Sprechblase) ergänzen (A und B). Das ist allerdings nicht nur für Tooltipps gedacht. Auch Screenreader können den Text im `title` vorlesen und verbessern damit die Zugänglichkeit.

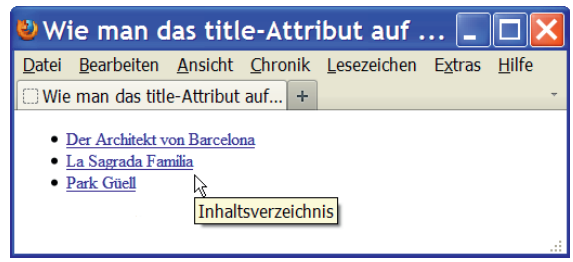
Elemente auf einer Webseite mit title ergänzen

Im HTML-Tag des zu kennzeichnenden Elements fügen Sie **`title="label"`** ein, wobei **`label`** der kurze, beschreibende Text ist, der im Tooltipp erscheinen soll, wenn ein Besucher mit der Maus auf das Element zeigt. Dieser Text wird auch im Screenreader vorgelesen.

TIPP Ältere Versionen des Internet Explorers (IE7 und früher) machen aus dem `alt`-Attribut, das in `img`-Elementen verwendet wird, ebenfalls Tooltipps (siehe Kapitel 5). Doch wenn in einem `img`-Element sowohl das `title`- und auch das `alt`-Attribut vorhanden sind, wird der Tooltipp aus dem Inhalt des `title`-Attributs befüllt und nicht aus dem `alt`-Attribut.

A *Sie können Titel bei jedem beliebigen Element einfügen, meist macht man das aber bei Links.*

```
...
<body>
   <header role="banner">
      <nav role="navigation">
         <ul id="ivz" title="Inhaltsverzeichnis">
            <li><a href="#gaudi" title="Erfahren
            → Sie mehr über Antoni Gaudí">Der
            → Architekt von Barcelona</a></li>
            <li><a href="#sagrada-familia"
            → lang="es">La Sagrada Família</a>
            → </li>
            <li><a href="#park-guell">Park Güell
            → </a></li>
         </ul>
      </nav>
   </header>

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



B *Wenn der Besucher auf das gekennzeichnete Element zeigt, erscheint der Titel. Wenn man die Maus über den Link „Der Architekt von Barcelona" hält, sieht man „Erfahren Sie mehr über Antoni Gaudí", weil es ein eigenes `title`-Attribut hat.*

Kommentare einfügen

Wenn Sie in Ihre HTML-Dokumente Kommentare einfügen, können Sie damit kennzeichnen, wo Sektionen beginnen oder enden, für sich selbst (oder spätere Bearbeiter) Hinweise einflechten, wozu ein bestimmter Codeabschnitt gedacht ist, verhindern, dass Inhalte dargestellt werden, und vieles mehr A. Diese Kommentare erscheinen nur, wenn das Dokument mit einem Editor oder über die Browser-Option Quelltext anzeigen geöffnet wird. Besucher sehen sie in ihrem Browser nicht B.

A *In diesem Beispielcode stehen vier Kommentare. Zwei zeigen kombiniert Anfang und Ende des Artikels an. Mit dem nächsten wird der erste Absatz „auskommentiert", damit er auf der Seite nicht dargestellt wird (wenn Sie auf lange Sicht diesen Absatz ausschließen wollen, wäre es besser, ihn im HTML zu löschen). Der letzte Kommentar ist eine Erinnerung, weitere Inhalte einzufügen, bevor die Seite live geschaltet wird. Achten Sie allerdings genau darauf, solche temporären Kommentar wie „To-do-Listen" zu entfernen, bevor Sie die Seite veröffentlichen, falls sich Besucher Ihren Code ansehen. Im Abschnitt „Generische Container erstellen" finden Sie weitere Kommentarbeispiele.*

```
...
<body>
   ...

   <!-- ==== START ARTICLE ==== -->
   <article class="architect">
      <h1 id="gaudi">Der Architekt von Barcelona</h1>

   <!-- Dieser Absatz wird nicht dargestellt, weil er auskommentiert ist.
      <p>Viele Touristen kommen extra nach Barcelona, um sich die unbeschreibliche Architektur von Antoni
      → Gaudí anzuschauen.</p>
   -->

      <p>Seine Nonkonformität, die schon in Jugendjahren deutlich wurde, verbunden mit seiner ruhigen,
      → aber innigen Hingabe an die Kirche, schufen eine einzigartige Grundlage für seine Gedanken und
      → Ideen. Seine Suche nach Einfachheit ... </p>

      <section class="project">
         <h2 id="sagrada-familia" lang="es">La Sagrada Família</h2>
         ...
      </section>

      <section class="project">
         <h2 id="park-guell">Park Güell</h2>
         ...
      </section>
   </article>
   <!-- Ende article -->

   <!-- To Do: Hier noch einen Artikel über andere berühmte Bauwerke einstellen, bevor die Seite live geht.
   -->

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

Kommentare auf einer HTML-Seite einfügen

1. In Ihrem HTML-Dokument tippen Sie dort, wo Sie einen Kommentar einfügen wollen, **<!--** ein.
2. Schreiben Sie die entsprechenden Kommentare.
3. Mit Eingabe von **-->** schließen Sie den kommentierten Text ab.

TIPP Meist nutzt man Kommentare dafür, sich (und zukünftige Bearbeiter) daran zu erinnern, bestimmte Abschnitte aufzunehmen, zu entfernen oder zu aktualisieren A.

TIPP Ein weiterer Grund für Kommentare ist das Vermerken einer Revisionsnummer.

TIPP Üblicherweise fügt man zu Beginn und am Ende von größeren Codeabschnitten Kommentare ein, damit Sie und andere die Seiten einfacher bearbeiten können (Seiten können ganz schön lang werden). Ich nehme gerne ein auffälliges Format, wenn ich einen Kommentar beginne, und ein weniger deutliches fürs Ende, damit ich auf einen Blick schnell diese beiden wichtigen Punkte erkenne, wenn ich den Code überfliege A.

TIPP Sie sollten sich Ihre kommentierte Seite mit einem Browser anschauen, bevor Sie sie veröffentlichen. So vermeiden Sie es, unabsichtlich vielleicht gar private Kommentare zu publizieren, weil ein Kommentar womöglich falsch formatiert wurde.

TIPP Vorsicht allerdings vor Kommentaren, die tatsächlich allzu privat sind. Man kann sie im Browser zwar nicht gleich sehen, aber sie können ganz einfach sichtbar gemacht werden, wenn der User sich die Seite über die Option QUELLCODE ANZEIGEN anschaut oder die Seite als HTML-Quellcode speichert.

TIPP Kommentare sollten nicht in anderen Kommentaren verschachtelt werden.

TIPP Die hier gezeigte Syntax gilt nur für HTML-Kommentare. CSS und JavaScript haben eine andere Kommentarsyntax. CSS und JavaScript arbeiten beide mit `/* Hier kommt der Kommentar */` für einen Kommentar, der über eine oder mehrere Zeilen geht, während bei JavaScript noch die Schreibweise `// Hier kommt der Kommentar` für einzeilige Kommentare möglich ist.

B Kommentare sind unsichtbar (obwohl man sie lesen kann, wenn man sich den Quellcode anzeigen lässt). Entsprechend werden Inhalte, die Sie in Kommentarzeichen setzen, nicht dargestellt A. Hier wird der erste Absatz im Code nicht gezeigt.

Copyright