3

Die grundlegende HTML-Struktur

In diesem Kapitel geht es um die HTML-Elemente, mit denen Sie Ihrem Dokument Grundlage und Struktur verleihen, also die primären semantischen Container Ihrer Inhalte.

Sie lernen Folgendes:

- Wie man eine neue Webseite aufbaut und strukturiert
- Die Elemente `h1`-`h6`, `header`, `nav`, `main`, `article`, `section`, `aside`, `footer` und `div` (die meisten in HTML5 neu)
- 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

Schafft man eine klare und einheitliche Struktur, sind die Webseiten nicht nur unter semantischen Gesichtspunkten gut vorbereitet, sondern man kann das Dokument auch leichter anhand von Cascading Style Sheets (CSS) formatieren (siehe Kapitel 7).

Übersicht

Elizabeth Castro / Bruce Hyslop, Praxiskurs HTML5 & CSS3, dpunkt.verlag, ISBN 978-3-86490-161-4

Eine neue Webseite beginnen

Ihre HTML-Dokumente sollten mindestens folgende Komponenten enthalten – siehe A:

- Den DOCTYPE
- Das `html`-Element (mit dem Attribut `lang`, optional, aber ratsam)
- Das `head`-Element
- Die Zeichenkodierung in einem `meta`-Element
- Das `title`-Element (dessen Inhalt fügen Sie gleich ein)
- Das `body`-Element

Wie bereits in Kapitel 1 angemerkt, sorgen diese HTML-Komponenten sozusagen für ein leeres Blatt, da im `body`-Element noch kein Inhalt vorhanden ist B.

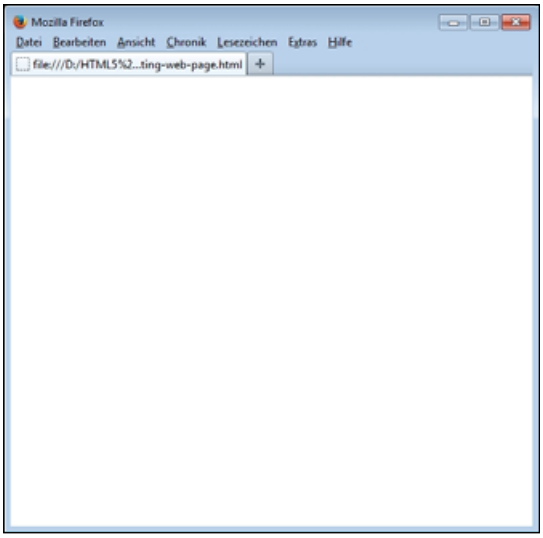
Bevor Sie Inhalte einfügen, müssen Sie das Grundgerüst der Seite vorbereiten:

Eine HTML5-Seite beginnen

1. Tippen Sie `<!DOCTYPE html>`, um Ihre Seite als HTML5-Dokument zu deklarieren (siehe „Der verbesserte DOCTYPE von HTML5“ auf Seite 58).
2. Tippen Sie `<html lang="`*`sprach-code`*`">`, um den eigentlichen HTML-Bereich Ihres Dokuments zu beginnen. *`sprach-code`* ist die Sprachkodierung für die Standardsprache Ihrer Seiteninhalte. Zum Beispiel steht `<html lang="es">` für Spanisch oder `<html lang="fr">` für Französisch. Sie werden noch spezifischer, wenn Sie `<html lang="en-US">` für Amerikanisches Englisch oder `<html lang="de-DE">` für Hochdeutsch schreiben.
3. Tippen Sie `<head>` für den Dokumentkopf der Seite.

A Dies ist die Grundlage aller HTML-Seiten. Die Einrückungen des Codes sind nebensächlich, aber die Struktur ist wesentlich. Das `html`-Element, das auf DOCTYPE folgt, muss alle anderen Elemente Ihrer Seite einschließen. 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 kommen die Inhalte.

Rückblick auf Kapitel 1

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

4. Tippen Sie `<meta charset="utf-8" />` oder `<meta charset="UTF-8" />`, um UTF-8 als Zeichenkodierung Ihres Dokuments zu deklarieren. Leerzeichen und Schrägstrich sind optional, also funktionieren `<meta charset="utf-8">` und `<meta charset="UTF-8">` genauso gut. (Auch andere Zeichenkodierungen als UTF-8 sind gültig. Weil dies aber am vielseitigsten ist, brauchen Sie selten davon abzuweichen.)
5. Tippen Sie `<title></title>`. Darin steht der Titel der Seite. Sie fügen den Text dafür im Abschnitt „Einen Titel erstellen“ ein.
6. Tippen Sie `</head>`, um den Dokumentkopf der Seite zu beenden.
7. Tippen Sie `<body>` für den Textkörper der Seite. Hier kommen dann Ihre Inhalte hinein.
8. Lassen Sie hier 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, erstellen Sie am besten eine HTML-Seite und speichern sie als Vorlage ab, um sich die viele Tipparbeit zu ersparen. Tatsächlich kann man bei den meisten Code-Editoren den Startcode für jede neue Seite abspeichern, was alles noch mehr erleichtert. Falls Sie im Editor keine Optionen wie Einstellungen oder Präferenzen finden, schauen Sie in der Hilfe nach.

Die beiden Bereiche einer Seite: Kopf und Körper

Eine kurze Wiederholung aus Kapitel 1: Seiten werden innerhalb des `html`-Elements in zwei Bereiche aufgeteilt: den `head` (Seitenkopf) und den `body` (Seitenkörper) Ⓐ. Der DOCTYPE, der jede Seite einleitet, ist so etwas wie eine Präambel. Hier erfährt der Browser die HTML-Version der Seite. Sie sollten ihn stets in Ihre Seiten aufnehmen.

Im `head` des Dokuments definieren Sie den Seitentitel, schreiben Infos über Ihre Seite für Suchmaschinen, laden Stylesheets und gelegentlich auch JavaScript-Dateien (in Kapitel 19 erfahren Sie, warum man hier lieber kein JavaScript laden sollte). Beispiele dafür lernen Sie im Verlaufe dieses Buchs kennen. Der Besucher sieht außer dem `title` erst einmal nichts vom Inhalt des Kopfabschnitts `head`, wenn er Ihre Seite besucht.

Das `body`-Element umfasst den Inhalt der Seite, z.B. Text, Bilder, Formulare, Audio- und Videodateien. In diesem Buch widme ich mehrere Kapitel speziell den inhaltsbezogenen HTML-Elementen und auf ein paar davon weise ich gleich noch hin.

Der verbesserte DOCTYPE von HTML5

Ach, um wie vieles einfacher ist es seit HTML5 geworden, eine Webseite zu beginnen! Der DOCTYPE von HTML5 ist erfrischend kurz, verglichen mit den DOCTYPEs von früher.

Zu Zeiten von HTML 4 und XHTML 1.0 musste man sich noch für einen der DOCTYPEs entscheiden. Man musste sie dauernd kopieren, weil sie einfach zu lang und zu kryptisch 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!

Heute schreiben Sie nur noch <!DOCTYPE html>.

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.

TIPP Der DOCTYPE von HTML5 gewährleistet, dass die Browser alles korrekt darstellen, und informiert die HTML-Validatoren, dass Ihr Code anhand der für HTML5 erlaubten Elemente und Syntax geprüft werden soll. HTML-Validatoren sind Thema von Kapitel 20.

TIPP Beim DOCTYPE von HTML5 wird nicht zwischen Groß- und Kleinschreibung unterschieden. So ist es üblicher: `<!DOCTYPE html>` A.

TIPP Eine Suchmaschine nutzt die in `lang` angegebene Sprache, um Suchergebnisse zu kategorisieren und nur solche darzustellen, die zur Sprache der gesuchten Phrase passen. Screenreader passen die Aussprache eines Worts anhand des angegebenen Sprachcodes an.

TIPP Mit dem Such-Tool für Sprach-Subtags von Richard Ishida (http://rishida.net/utils/subtags/) finden Sie Sprachcodes.

TIPP Ihr Code-Editor sollte so konfiguriert sein, dass Dateien als UTF-8 gespeichert werden, damit sie zu dem über `<meta charset="utf-8" />` A gespeicherten Code passen. (Falls Sie ein anderes `charset` angeben, speichern Sie Ihre Dateien damit.) Nicht alle Code-Editoren speichern die Seiten standardmäßig als UTF-8, aber meist kann die Kodierung per Menü oder Übersicht gewählt werden (siehe „Die Webseite speichern“ in Kapitel 2). Wird kein UTF-8 verwendet, erscheinen vielleicht statt des gewünschten Buchstabens wie ein *í* oder ein *ñ* komische Zeichen in Ihrem Inhalt.

TIPP Den im `head`-Element verschachtelten Code müssen Sie nicht einrücken A. Allerdings erkennen Sie dann gleich, wo das `head`-Element anfängt, was darin steht und wo es endet. Es kommt nicht selten vor, dass auf manchen Seiten `head` sehr lang wird.

Den Seitentitel erstellen

Im HTML-Basiscode des vorigen Abschnitts haben wir `<title></title>` als Platzhalter eingefügt. Jetzt werde ich das `title`-Element näher erläutern.

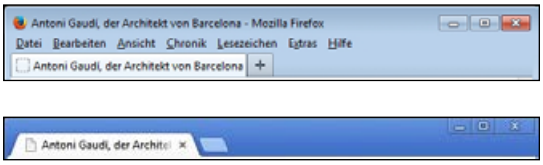
Jede HTML-Seite muss ein `title`-Element für den Seitennamen enthalten. Der Seitentitel sollte kurz, aussagekräftig und für jede Seite eindeutig sein A. Bei den meisten Browsern erscheint der Name der Seite in der Titelleiste des Fensters (Ausnahme Chrome) und im Browser-Reiter B. Der Seitenname erscheint auch im Browserverlauf sowie in den Favoriten- bzw. Bookmark-Listen C.

Vielleicht noch wichtiger: Der Titel wird von Suchmaschinen wie Google berücksichtigt. Er sagt, welche Informationen Ihre Seite enthält, und er erscheint auch im Link in den Suchergebnissen D.

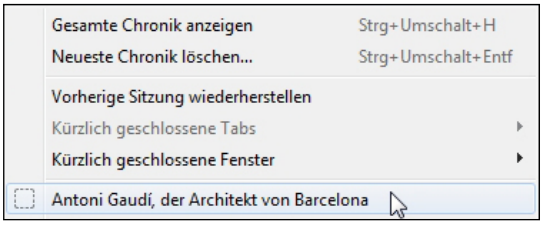
Kurz gesagt, verbessern Sie Suchmaschinenresultate und User Experience, wenn Sie das `title`-Element für jede Seite möglichst relevant und aussagekräftig gestalten.

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

```
<!DOCTYPE html>
<html lang="de-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 Ihr Seitentitel wird üblicherweise als Link-Text in den Suchresultaten verwendet und verweist auf Ihre Seite. Überdies bestimmt er die Relevanz einer Seite bei den Suchergebnissen. Üblicherweise stellen Suchmaschinen den Titel und Teil des Texts aus dem `body` Ihrer Seite dar.

Seitentitel unter der Lupe

Viele Entwickler – darunter sogar welche mit ziemlicher 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 standardmäßig einfügt. Wenn Sie Traffic auf Ihre Site leiten wollen, leisten Sie sich und potenziellen Lesern einen Bärendienst, falls Sie genauso handeln.

Suchmaschinen arbeiten mit verschiedenen Algorithmen, um die Relevanz einer Seite zu bestimmen und ihren Inhalt zu indexieren. Allerdings spielt der `title` dabei eine Schlüsselrolle. Suchmaschinen erfahren aus dem `title`, 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 relevant sind.

Empfehlenswert ist also, den `title`-Text als Zusammenfassung des Dokumentinhalts zu wählen – davon profitieren sowohl Ihr Ranking bei den Suchmaschinen als auch die Nutzer von Screenreadern (der Titel kann vorgelesen werden).

Außerdem 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, stattdessen den typischen, für die Seite spezifischen `title`-Text zu Anfang zu setzen.

Ich rate, die zentrale Botschaft des `title` in den ersten 60 Zeichen (inklusive Leerzeichen) unterzubringen. Als Faustregel: Nach etwa 60 Zeichen schneiden Suchmaschinen die Darstellung ihrer Resultate ab. Browser stellen in der Titelleiste oben im Fenster verschieden viele Zeichen dar, bevor sie den Text abschneiden. Bei Browser-Tabs wird der Titel sogar noch früher beschnitten, weil es weniger Platz gibt.

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 bei Beginn einer neuen Seite den `title` mit Standardtext, außer Sie haben – siehe „Eine neue Webseite beginnen“ auf Seite 56 – den Editor so konfiguriert, dass spezieller Vorlagenstartcode verwendet wird. Also achten Sie jeweils darauf und sorgen Sie ggf. dafür, Standardtext durch einen eigenen `title` zu ersetzen.

Überschriften erstellen

In HTML können Sie für Ihre Webseite bis zu sechs Überschriftenebenen vergeben, um die Hierarchie der Informationen Ihrer Seite zu beschreiben. Zeichnen Sie jede Überschrift mit dem Element **h1**, **h2**, **h3**, **h4**, **h5** oder **h6** aus. Organisatorisch ist **h1** eine Überschrift erster Ordnung, **h2** die Überschrift unter **h1**, **h3** eine Überschrift unter **h2** usw. Ich bezeichne sie jeweils kollektiv als **h1**-**h6** und liste sie nicht einzeln auf.

Stellen Sie sich **h1**–**h6** vor wie Überschriften aus sonstigen Nicht-HTML-Dokumenten, z.B. Verkaufsberichte, Hausarbeiten, Produkthandbücher oder Zeitungsartikel. Wenn Sie solche Dokumente schreiben, beginnen Sie jeden Hauptabschnitt des Inhalts mit einer Überschrift. Jeder Unterabschnitt bekommt eine Unterüberschrift (und ggf. weitere Unterunterüberschriften) usw. Zusammen bilden diese Überschriften die Gliederung des Dokuments. Das Gleiche gilt für Ihre Webseiten (A und B).

Die Bedeutung von Überschriften

Überschriften gehören zu den wichtigsten HTML-Elementen jeder Seite. Weil Überschriften die Themen Ihrer Seiten vermitteln, werden sie von Suchmaschinen besonders gewichtet, wenn die Seiten indexiert werden. Das gilt vor allem für die Überschriften erster Ordnung **h1** (was nicht bedeutet, dass Sie Ihre Seite nur mit **h1**s vollstopfen dürfen – dafür sind die Suchmaschinen zu clever).

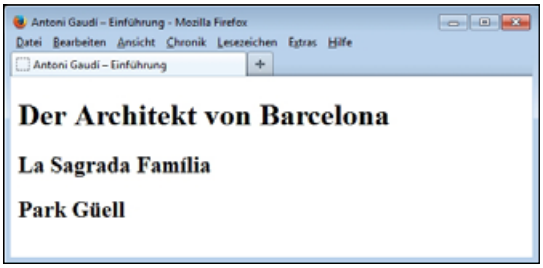
Auch Menschen finden gute Überschriften toll. User ohne visuelle Einschränkungen überfliegen die Überschriften und verschaffen sich so einen Eindruck vom Seiteninhalt. Die User von Screenreadern machen das Gleiche, nur per Hand und Ohr. Sie navigieren auf den Seiten oft per Tastatur durch die Überschriftenstruktur. Durch Vorlesen der Überschriften erfassen sie schnell die Inhalte einer Seite und finden für sie Interessantes, ohne dass die gesamte Seite vorgelesen werden muss. Stoßen sie auf eine interessante Überschrift, ent-

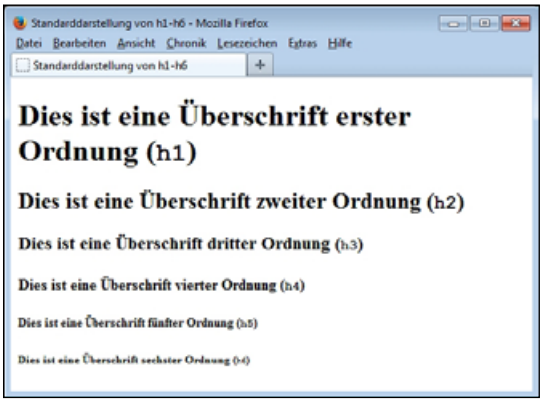
A Sie können anhand von Überschriften Ihr Dokument wie eine Gliederung strukturieren. Hier sind „Park Güell" und „La Sagrada Família" – als **h2**-Elemente ausgezeichnet – Unterüberschriften der obersten Überschrift „Der Architekt von Barcelona", weil diese ein **h1** ist. Wäre „Park Güell" stattdessen ein **h3**, wäre es eine Unterüberschrift von „La Sagrada Família" (und eine Unterunterüberschrift von **h1**.) Würden Sie gleich danach den Rest der Seite schreiben, dann folgten die damit zusammenhängenden Inhalte (Absätze, Bilder, Videos usw.) direkt nach jeder Überschrift.

```
<!DOCTYPE html>
<html lang="de-de">
<head>
    <meta charset="UTF-8" />
    <title>Antoni Gaudí – Einführung
    →</title>
</head>
<body>
<h1>Der Architekt von Barcelona</h1>
<h2 lang="es">La Sagrada Família</h2>
<h2>Park Güell</h2>
</body>
</html>
```

B In diesem Beispiel gibt es drei Hauptbereiche innerhalb des Produktleitfadens, jeder mit verschiedenen Unterüberschriften. Ich habe Leerzeichen und Einrückungen nur zur Verdeutlichung der Hierarchie eingefügt. In der Praxis würde ich das nicht so einrücken, obwohl es möglich ist, denn es wirkt sich nicht auf die Darstellung in den Browsern aus.

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

    <h2>Basisfeatures</h2>
       <h3>Videowiedergabe</h3>
          <h4>Grundfunktionen der Steuerung</h4>
          <h4>Zu Markierungen springen</h4>

       <h3>Videoaufnahme</h3>
          <h4>Manuelle Aufnahme</h4>
          <h4>Aufnahmezeitpunkt planen</h4>

    <h2>Weitere Features</h2>
       <h3>Video weitergeben</h3>
       <h3>Video komprimieren</h3>
</body>
</html>
```





C Zwar sind alle Überschriften standardmäßig fett, aber `h1` hat eine größere Schrift als `h2`, die wiederum größer ist als `h3` usw. Der Abstand zwischen den Überschriften ist ebenfalls die Folge des Standard-CSS im Browser und nicht von leeren Zeilen, die Sie womöglich in Ihr HTML-Dokument einfügen.

scheiden sie, ob auch der damit verknüpfte Inhalt vorgelesen werden soll. Die Nutzbarkeit (Usability) und Zugänglichkeit (Accessibility) profitiert sehr von `h1`-`h6`.

Kurz gesagt nützt eine gute Verwendung von Überschriften Ihnen und den Besuchern.

Eine Webseite mit Überschriften strukturieren

1. Im `body`-Abschnitt Ihres HTML-Dokuments tippen Sie `<hn>`, wobei *n* abhängig von der Überschriftsebene für eine Zahl zwischen 1 und 6 steht. `h1` ist die Überschrift höchster Ordnung und `h6` die mit dem niedrigsten Rang.
2. Geben Sie den Inhalt der Überschrift ein.
3. Tippen Sie `</hn>`, wobei *n* die gleiche Zahl ist wie in Schritt 1.

TIPP Standardmäßig stellen Browser die Überschriften von `h1` bis `h6` stufenweise kleiner dar C. (Bei manchen Browsern sehen `h1` und `h2` standardmäßig gleich aus, wenn sie in bestimmten Elementen verschachtelt sind.) Aber die Überschriftenebenen sollten Sie einzig danach ausrichten, welche Hierarchie für den Inhalt passt, und nicht danach, wie groß oder klein der Text erscheinen soll. Sie formatieren die Überschriften z.B. mit einer bestimmten Schriftart, Schriftgröße bzw. Schriftfarbe usw. Wie man das mit CSS macht, erfahren Sie in Kapitel 10.

TIPP Überspringen Sie beim Erstellen der Überschriften keine Ebene. Wechseln Sie also nicht von `h3` gleich zu `h5`. Umgekehrt können Sie allerdings von einem niedrigeren Rang direkt zu einer Überschrift *höherer* Ordnung springen, falls das passt. Beispielsweise gibt es in B `<h4>Aufnahme zeitpunkt planen</h4>` und dann `<h2>Weitere Features</h2>`. Diese Sequenz ist so sinnvoll, da im Abschnitt mit den Grundfunktionen (ebenfalls `h2`) der Punkt Aufnahmezeitpunkt planen endet und der Punkt Weitere Features beginnt.

weiter auf der nächsten Seite

TIPP Nehmen Sie kein **h1**-**h6**, um Untertitel, Slogans oder Unterüberschriften auszuzeichnen, wenn es sich nicht um Überschriften für eigene Abschnitte handelt. Nehmen wir beispielsweise eine Nachrichtenmeldung, die eine Hauptüberschrift und direkt danach eine Unterüberschrift enthält. In einem solchen Fall verwenden Sie einen Absatz D oder ein anderes Element, das keine Überschrift ist.

TIPP Früher enthielt HTML5 ein Element namens **hgroup**, um aufeinanderfolgende Überschriften zu gruppieren, aber das W3C hat es aus der HTML5-Spezifikation entfernt.

TIPP Anmerkung: In A habe ich das Attribut **lang** bei der ersten **h2** genommen, um anzuzeigen, dass ihre Inhalte in einer anderen Sprache vorliegen (laut Sprachcode ist es Spanisch) als der Rest der Seite (Hochdeutsch), der durch **<html lang="de-de">** deklariert wird.

TIPP Während ich dies schreibe, gibt es Diskussionen darüber, ob man ein **subhead**-Element in HTML aufnehmen soll, um Unterüberschriften, Untertitel, Slogans und Verfasserzeilen auszuzeichnen. Man kann noch nicht einschätzen, ob es übernommen wird.

D So wird eine Unterüberschrift oder ein Untertitel eines Artikels, Blogeintrags o.Ä. ausgezeichnet. Sie können eine **class** (z.B. namens **subhead**) einfügen, um das Formatieren mit CSS zu erleichtern.

```
...
    <h1>Giraffe entflieht aus Zoo</h1>
    <p class="subhead">Allerorten frohlocken
    → die Tiere </p>

    <p>... Inhalt der Geschichte ... </p>
...
```

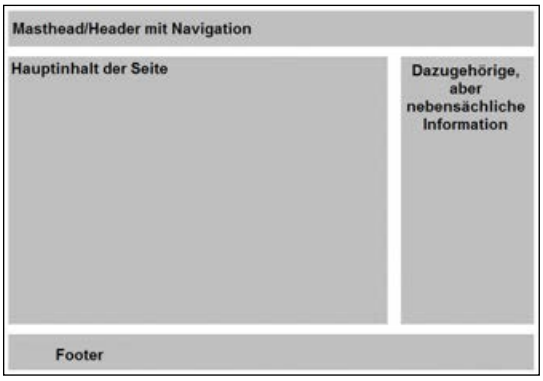
Die Gliederung eines HTML5-Dokuments

Zur Zeit enthält HTML5 einen Algorithmus, wie mit **h1**–**h6** umzugehen ist, wenn sie in den Elementen **article**, **aside**, **nav** und **section** verschachtelt sind. Dieser Algorithmus wird oft als *HTML-Dokumentstruktur* bezeichnet. Überdies wird er unter den Screenreadern nur von JAWS (für Windows) beachtet, aber dessen Implementierung ist fehlerhaft.

Vor diesem Hintergrund hat das W3C die Dokumentstruktur auf die Liste der Features gesetzt, die irgendwann möglicherweise aus der Spezifikation fallen. Auch wenn sie in der Spezifikation verbleibt oder Browser sie implementieren, implementieren Sie weiterhin Ihre Überschriften wie hier von mir gezeigt. Es entspricht aktueller Praxis und ist zukunftssicher – die Dokumentstruktur macht Ihre Seiten nicht kaputt.

Anders ausgedrückt gibt es hier nichts zu sehen! Ich erwähne dies alles nur deswegen, falls Sie an anderer Stelle auf die Dokumentstruktur stoßen und wissen wollen, ob Sie sie lernen sollten.

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



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

Übliche Seitenkonstrukte

Zweifelsohne haben Sie schon Dutzende Sites besucht wie die in A gezeigte. Entkleidet man sie vom Inhalt, erkennen Sie vier Hauptkomponenten: eine Kopfzeile (Header) mit Navigation, ein Artikel im Hauptinhaltsbereich, eine Seitenleiste (Sidebar) mit weiterführenden Informationen und eine Fußzeile (Footer) B.

Nun können Sie eine solche Seite nicht ohne CSS so formatieren oder wie gezeigt arrangieren. In Kapitel 7 steigen wir dann mit CSS ein, über das Formatieren von Text erfahren Sie mehr in Kapitel 10 und mehrspaltige Layouts lernen Sie in Kapitel 11 kennen.

Doch egal welches Layout, die für die üblichen Seitenkonstrukte angewendete Semantik ist praktisch immer recht ähnlich. Darum wird es nun vorrangig im restlichen Kapitel gehen. Wir arbeiten uns von oben nach unten durch die Seite und lernen, wie man die Elemente `header`, `nav`, `main`, `article`, `section`, `aside` und `footer` nutzt, um die Struktur der Seiten zu definieren. Dann kommt `div` als generischer Container ins Spiel für weitere Formatierungen und andere Zwecke. Mit Ausnahme von `div` gab es diese Elemente vor HTML5 noch gar nicht.

Um sich mit diesen Elementen vertraut zu machen, sollten Sie sich nicht so sehr daran klammern, wo sie im Seitenlayout jeweils erscheinen, sondern sich auf die semantische Bedeutung konzentrieren.

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

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.

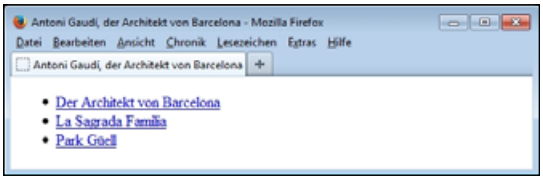
Auf einer Seite können beliebig viele **header**-Elemente erscheinen, deren Bedeutung sich je nach Kontext unterscheiden kann. Ein **header** ganz oder recht weit oben auf der Seite repräsentiert die Kopfzeile für die ganze Seite (manchmal auch Masthead genannt) A. Oft stehen in der Kopfzeile das Site-Logo, die Hauptnavigation B, globale Links und manchmal auch ein Suchfeld C. Sicher wird das **header**-Element dafür am häufigsten eingesetzt, doch das ist nicht der einzige Verwendungszweck.

Ein **header** kann auch an anderer Stelle auf der Seite erscheinen, wenn er dort passt. Ein Beispiel dafür wäre das Inhaltsverzeichnis eines Bereichs C.

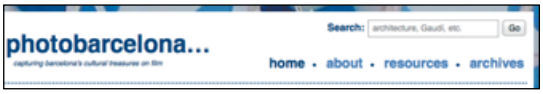
Ein **header**-Element enthält also oft die Überschrift seines Abschnitts (ein **h1**-**h6**), ist aber nicht zwingend. Sie sehen beispielsweise Überschriften in C, aber nicht in A.

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. Der optionale Bereich **role="banner"** passt nicht für jeden Header. Er verbessert die Accessibility, indem der Header für die Seite explizit ausgezeichnet wird. In D finden Sie ein weiteres Beispiel und erfahren in „Mit ARIA die Zugänglichkeit verbessern“ auf Seite 90 mehr darüber (siehe „Navigation auszeichnen“, auf Seite 70 wenn Sie mehr über den fürs **nav**-Element spezifischen **role**-Wert wissen wollen).

```
...
<body>
<header role="banner">
    <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.



C Hier ist ein formatierter Header für eine andere Site. Diese Art von Seiten-Header trifft man im Web vielerorts an. Er enthält den Namen der Site (meist ein Logo), Links zur Navigation für die Hauptbereiche der Site und ein Suchfeld.

D Hier sehen Sie zwei **header-Elemente**: eines als Header für die gesamte Seite und ein anderes 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. Außerdem ist nur im ersten **header** das **role="banner"** enthalten, weil es der Header auf Seitenebene ist.

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

<main role="main">
<article>
    <header>
       <h1>Häufig gestellte Fragen</h1>
       <nav>
          <ul>
             <li><a href="#answer1">Wie lauten
             →Ihre Rücknahmebedingungen?</a><li>
             <li><a href="#answer2">Wie finde ich
             →eine Filiale?</a><li>
             ...
          </ul>
       </nav>
    </header>

    <!-- Links im zweiten Header zeigen
    →hierher -->
    <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 -->
</main>
</body>
</html>
```

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 (viele lernen Sie in den folgenden Kapiteln kennen). Ein **header** enthält beispielsweise Überschriften der Ebene **h1** bis **h6**, ein oder mehrere Logos, Navigation, Suchfeld usw.
4. Tippen Sie **</header>**.

TIPP Nehmen Sie **header** nur, wo es nötig ist. Falls es nur ein **h1**-**h6** gibt, braucht man meist nicht alles in einen **header** zu fassen.

TIPP Ein **header** ist nicht mit einer Überschrift austauschbar wie die **h1**-**h6**-Elemente (siehe „Überschriften erstellen“ auf Seite 62). Alle haben ihren eigenen semantischen Zweck.

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

TIPP Ein **header**-Element muss nicht immer ein **nav**-Element enthalten wie hier (A und D), doch meist ist das so, wenn das **header**-Element Navigationslinks enthält. Bei D passt das **nav** für die Liste der Links für die FAQs, da es sich um eine wichtige Navigationsgruppe auf der Seite handelt (siehe „Navigation auszeichnen“ auf Seite 68).

TIPP In „Generische Container erstellen“ auf Seite 85 erfahren Sie mehr, wie **header** eine der Rollen des **div**-Elements aus der Zeit vor HTML5 ersetzte.

Navigation auszeichnen

Bei früheren Versionen von HTML gab es kein Element, das explizit einen Bereich von wichtigen Navigationslinks repräsentiert, aber in HTML5 haben wir nun 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 nur für die wichtigsten Link-Gruppen verwenden, nicht für alle.

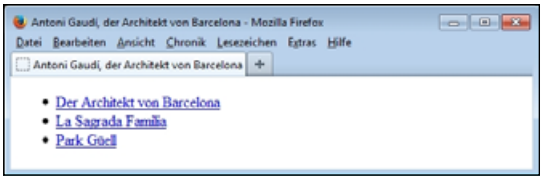
Schauen Sie sich den Code im vorigen Abschnitt genauer an: das `nav`-Element in Aktion. Ich greife dieses Codebeispiel noch einmal auf und hebe `nav` hervor. Das `nav`-Element zwingt seinen Inhalten keine Standardformatierung auf (B).

Eine Link-Gruppe als wichtige Navigation kennzeichnen

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

(A) Diese Links (die `a`-Elemente) repräsentieren eine wichtige Navigationsgruppe, also habe ich sie in einem `nav`-Element verschachtelt. Das `role`-Attribut ist nicht erforderlich, verbessert aber die Zugänglichkeit. Siehe den letzten Tipp in diesem Abschnitt, wie man `role="navigation"` auf `nav` anwendet.

```
...
<body>
<header role="banner">
    <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 sind kein Produkt des `nav`-Elements, für das es, abgesehen vom eigenen Absatz, keine Standardformatierung gibt. Die Gliederungspunkte erscheinen, weil sich jeder Link in einem `li`-Element befindet (list item, Listenelement). Mit CSS können Sie die Gliederungspunkte abschalten oder andere wählen und die Links z.B. horizontal anordnen, deren Farbe ändern, sie wie Schaltflächen aussehen lassen etc. In den Kapiteln 11 bzw. 15 finden Sie Beispiele für formatierte Link-Listen.

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

TIPP Screenreader ringen immer noch mit der neuen HTML5-Semantik und so hilft das `nav`-Element, die wichtige Navigation auf der Seite zu identifizieren. Dann können User sie per Tastatur direkt anwählen. So ist Ihre Seite besser zugänglich und bietet eine bessere User Experience. Übertreiben Sie es allerdings nicht. Falls zu viele Link-Gruppen mit `nav` ausgezeichnet sind, mindert das die Bedeutung der einzelnen Gruppe und die Seite wird für Screenreader unruhiger.

TIPP Die HTML5-Spezifikation empfiehlt, nebensächliche Fußzeilen wie „Nutzungsbedingungen“ oder „Datenschutzbestimmungen“ nicht in ein `nav` zu stellen (sinnvoll). Manchmal wollen Sie aber in die Fußzeile mehrere wichtige Links packen, z.B. „Filialen finden“, „Jobangebote“ etc. Sind diese nicht anderswo auf Ihrer Seite in einem `nav` gruppiert, könnten Sie sich überlegen, solche 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 in „Mit ARIA die Zugänglichkeit verbessern“ auf Seite 90 A.

nav unter der Lupe

Wie bereits erwähnt, müssen Sie eine Gruppe von Links nicht notwendigerweise in ein **nav** stecken.

Nehmen wir an, Sie haben eine Nachrichtenseite wie die in C. Es geht in dem Artikel aus der Rubrik „Kunst & Unterhaltung“ um eine Galerieeröffnung. Das folgende Beispiel mit einer Nachrichtenseite enthält vier Link-Listen, von denen nur zwei so relevant sind, dass es sich lohnt, sie mit einem **nav**-Element zu umschließen (der Code ist gekürzt).

Die sekundäre Navigation im **aside** erlaubt es dem User, auf die anderen Kategorienseiten im Verzeichnis „Kunst & Unterhaltung“ (z.B. Filme und Musik) zu navigieren, und konstituiert m.E. somit einen wichtigen Navigationsbereich auf der Seite. Aber beim **aside** „Andere Storys“ mit Links ist das nicht der Fall, ebenso nicht bei den **footer**-Links (in „Ein aside angeben“ auf Seite 77 erfahren Sie mehr über das **aside**-Element).

Wonach entscheiden Sie also, ob eine Gruppe mit Links ein **nav** verdient? Letzten Endes fällen Sie diese Beurteilung aufgrund des Inhalts. Zumindest die globale Navigation der Site sollten Sie mit **nav** auszeichnen (also den Bereich, über den User auf andere Bereiche der Site springen können). Meist erscheint dieses spezielle **nav** innerhalb eines **header**-Elements für die ganze Seite (siehe „Einen Header erstellen“ auf Seite 66).

C Nur zwei der Link-Gruppen dieser Seite sind in **nav**s eingefasst. Die anderen beiden gehören nicht zu den Hauptgruppen der Navigation.

```
...
<body>
    <!-- ==== Start Seitenkopf ==== -->
    <header role="banner">
       <!-- hier könnte das Site-Logo hin -->
       <!-- globale Navigation der Site -->
       <nav role="navigation">
          <ul> ... </ul>
       </nav>
    </header>

    <!-- ==== Beginn Hauptinhalt ==== -->
    <main role="main">
       <h1>Kunst & Unterhaltung: Museen</h1>

       <article>
          <h2>Galerieeröffnung stellt
          →inspirierte Arbeiten vor</h2>
          <p>... [Inhalt der Story] ... </p>
       </article>

       <aside>
          <h2>Andere Storys</h2>
          <!-- Nicht in ein nav  gefasst -->
          <ul> ... [Story- Links] ... </ul>
```

Code wird in nächster Spalte fortgesetzt

```
       </aside>
    </main>

    <!-- ==== Beginn Sidebar ==== -->
    <aside>
       <!-- Sekundäre Navigation -->
       <nav role="navigation">
          <ul>
             <li><a href="/kunst/filme/">
             →Filme</a></li>
             <li><a href="/kunst/musik/">
             →Musik</a></li>
             ...
          </ul>
       </nav>
    </aside>

    <!-- ==== Start Seitenfuß ==== -->
    <footer role="contentinfo">
       <!-- Weniger relevante Links stehen
       →nicht im nav. -->
       <ul> ... </ul>
    </footer>
</body>
</html>
```

Ⓐ Hier ist schon mal eine Vorschau auf die endgültige Seite, die wir in diesem Kapitel entwickeln werden. Manche Elemente werden für Sie neu sein, aber wir machen uns gleich damit vertraut. Das `main`-Element umgibt den Inhalt für das zentrale Thema der Seite. Es ist eine gute Praxis, `role="main"` im Start-Tag von `main` aufzunehmen (siehe letzter Tipp).

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

<main role="main">
    <article>
       <h1 id="gaudi">Der Architekt von
       →Barcelona</h1>
       <p>Viele Touristen kommen extra nach
       →Barcelona, um sich die unbeschreibliche
       →Architektur ...</p>

       ... [restlicher Hauptinhalt der Seite] ...
    </article>
</main>

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

    ... [restlicher Inhalt des aside]
</aside>

<footer role="contentinfo">
    ... [Copyright] ...
</footer>
</body>
</html>
```

Den Hauptbereich einer Webseite markieren

Die meisten Webseiten haben verschiedenste Bereiche: einen Kopf- und Fußbereich (Header und Footer), vielleicht einen Kasten mit weiteren Informationen oder Links zu anderen Sites etc. Doch nur ein Teil einer Seite repräsentiert den Hauptinhalt, also der, der den primären Fokus besitzt. Fassen Sie diesen Inhalt mit dem passend benannten `main`-Element ein, aber pro Seite nur einmal (Ⓐ und Ⓑ).

TIPP Das `main`-Element gehört zu den neuen HTML5-Elementen. Sie sollten es nur einmal pro Seite verwenden.

TIPP Wenn Sie eine Webapplikation erstellen, sollten Sie die Hauptfunktionalität in `main` einschließen.

TIPP Sie können `main` nicht in ein `article`-, `aside`-, `footer`-, `header`- oder `nav`-Element setzen.

TIPP Das Attribut-Wert-Paar `role="main"` hilft Screenreadern Ⓐ, den Hauptinhalt Ihrer Seiten zu lokalisieren. Über ARIA-Landmarks wie diese erfahren Sie mehr in „Mit ARIA die Zugänglichkeit verbessern“.

Ⓑ Die Darstellung des `main`-Elements beginnt in einer eigenen Zeile so wie `p`, `header`, `footer` und einige andere Elemente, wirkt sich aber ansonsten nicht auf die Formatierung der Seite aus. (Das hier sichtbare Layout stammt vom CSS, nicht von `main`.) Ich zeige mit der blauen Linie, welchen Bereich der formatierten, fertigen Seite `main` umfasst.

Einen Artikel erstellen

Ein anderes Element, das wir HTML5 zu verdanken haben, ist **`article`** (A und B). Nachdem Sie es ein paar Mal in Aktion gesehen haben, erfahren Sie nun mehr über dessen Funktionsweise.

Am Namen merken Sie, dass man in **`article`** Inhalte wie Zeitungsartikel packen kann. Doch darauf ist es nicht beschränkt. In HTML5 ist „article“ nicht wörtlich gemeint.

HTML5 definiert das so:

> Das **`article`**-Element ist ein eigenständiger Teil in einem Dokument, einer Seite, Anwendung oder Site und im Prinzip unabhängig wiederverwendbar oder weitergebbar, z.B. über einen RSS-Feed. Es ist beispielsweise anwendbar auf Forum-Postings, Magazin- oder Zeitungsartikel, Blog-Einträge, Benutzerkommentare, interaktive Widgets oder Gadgets oder irgendein anderes, inhaltlich eigenständiges Element.

Weitere **`article`**-Elemente sind z.B. Film- oder Konzertrezensionen, Fallstudien, Produktbeschreibungen usw. Sie sind vielleicht überrascht, dass es auch ein interaktives Widget oder Gadget sein kann, aber auch dies sind unabhängige, wiederverwendbare Inhaltselemente.

A 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 http://goo.gl/sBYdTn. Obwohl dieses Beispiel nur Absätze und Bilder aufweist, kann ein **`article`** inhaltlich sehr unterschiedliche Typen wie Video, Abbildungen, Listen usw. enthalten.

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

<main role="main">
    <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
        →... , 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 ... </p>

        <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 ...</p>

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

        ... [Bild und Absätze] ...
    </article>
</main>
</body>
</html>
```

B Nun besitzt die Seite `header`-, `nav`-, `main`- und `article`-Elemente und auch entsprechende Inhalte. 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 in beliebiger Anzahl ein, z.B. Absätze, Listen, Audio, Video, Fotos, Abbildungen usw.
3. Tippen Sie `</article>`.

TIPP Sie können **`article`**s verschachteln, solange das innere **`article`** mit dem **`article`** insgesamt zusammenhängt. Ein Beispiel dafür finden Sie im Kasten auf der nächsten Seite.

TIPP Eine Seite kann mehrere **`article`**-Elemente (oder keine) enthalten. Die Homepage eines Blogs enthält beispielsweise meist ein paar aktuelle Postings und alle könnten eigene **`article`**s sein.

TIPP In einem **`article`** kann ein **`section`**-Element (oder auch mehrere) enthalten sein – dazu gleich mehr. Es ist selbstverständlich auch möglich, allein durch die **`h1-h6`**-Elemente die Teile innerhalb Ihres **`article`** anzuzeigen, wie ich das hier gemacht habe.

Weitere `article`-Beispiele

Das vorige Beispiel A ist nur eine Einsatzmöglichkeit für **`article`**. Schauen wir uns weitere an.

Abbildung C illustriert, wie Sie einen einfachen News-Artikel oder Bericht auszeichnen. Beachten Sie, wie hier die Elemente **`footer`** und **`address`** eingesetzt werden (weitere Erläuterungen darüber finden Sie in diesem bzw. in Kapitel 4). Hier wird **`address`** nur auf das übergeordnete **`article`** (hier gezeigt) angewendet, nicht auf die Seite oder andere **`article`**-Elemente, die innerhalb dieses **`article`** verschachtelt sind, z.B. die Leserkommentare in D.

Abbildung D demonstriert verschachtelte **`article`**-Elemente in Form von Benutzerkommentaren zu dem übergeordneten **`article`**. Das kennen Sie aus dem Kommentarbereich von Blogs oder News-Sites. Es zeigt auch eine Verwendung des **`section`**-Elements (siehe „Einen Bereich (**`section`**) definieren“ auf Seite 75 und das in Kapitel 4 besprochene **`time`**-Element.

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

C Ein üblicher Weg, um einen Artikel auszuzeichnen, in dem Informationen über den Autor enthalten sind.

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

    ... [restlicher Inhalt]

    <!-- Footer des Artikels, nicht der der
    →Seite -->
    <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>
...
```

D Jeder Leserkommentar befindet sich in einem **`article`**, der im Haupt-**`article`** verschachtelt ist.

```
...
<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="2014-02-26">
          →26. Februar 2014</time>:
          →</footer>
          <p>Toller Artikel! Ich war schon
          →immer neugierig auf Papua
          →Neuguinea.</p>
       </article>

       <article>
          ... [nächster Leserkommentar] ...
       </article>
    </section>
</article>
...
```

A Schauen Sie sich eine beliebige News-Site an und Sie treffen höchstwahrscheinlich überall auf Nachrichtenschlagzeilen, die in verschiedenen Kategorien gruppiert sind. Jede Gruppe kann mit **section** ausgezeichnet werden.

```
...
<body>
...
<main role="main">
     <h1>Aktuelle Nachrichten aus aller Welt</h1>
     <section>
        <h2>Eilmeldungen</h2>
        <ul>... [Liste der Schlagzeilen] ...</ul>
     </section>

     <section>
        <h2>Business</h2>
        <ul>... [Liste der Schlagzeilen] ...</ul>
     </section>

     <section>
        <h2>Kunst</h2>
        <ul>... [Liste der Schlagzeilen] ...</ul>
     </section>
</main>
...
</body>
</html>
```

B Das folgende Beispiel stammt leicht verändert aus der HTML5-Spezifikation: **section** wird verwendet, um Bereiche im Programm einer Abschlussfeier abzugrenzen C.

```
...
<h1>Programm der Abschlussfeier</h1>
<section>
     <h2>Zeremonie</h2>
     <ol>
        <li>Eröffnungsprozession</li>
        <li>Verabschiedungsrede</li>
        <li>Rede des Jahrgangssprechers</li>
        ...
     </ol>
</section>

<section>
     <h2>Absolventen (alphabetisch)</h2>
     <ol>
        <li>Molly Carpenter</li>
        ...
     </ol>
</section>
...
```

Einen Bereich (section) definieren

Ein weiteres neues Element in HTML5 ist **section** (A bis C). Dies wird in HTML5 so definiert:

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

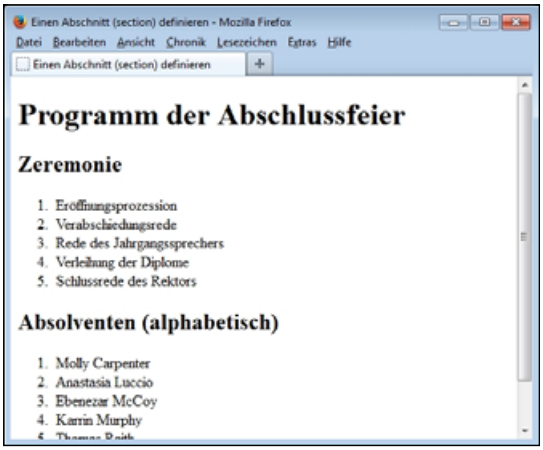
Obwohl **section** teilweise als „generischer" Bereich definiert wird, sollten Sie ihn nicht mit dem wirklich generischen **div**-Element verwechseln (siehe „Generische Container erstellen" auf Seite 85). Semantisch kennzeichnet **section** einen bestimmten Bereich der Seite, während **div** keine irgendwie geartete Bedeutung vermittelt.

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>`.

TIPP Sie können **section** in einem **article** verschachteln, um die verschiedenen Bereiche oder Kapitel eines Berichts, einer Geschichte oder eines Handbuchs usw. zu kennzeichnen. Ich könnte es für das Beispiel mit Antoni Gaudí in diesem Kapitel nehmen: Ein **section** umgibt das **h2** für La Sagrada Família und den damit zusammenhängenden Absatz und ein weiterer **section** das **h2** für den Park Güell und die entsprechenden Absätze.

TIPP Wenn Sie rein wegen der Formatierung einen Container um Inhalte legen wollen, nehmen Sie besser **div** statt **section**.



C Wie bei den meisten Elementen in diesem Kapitel wirkt sich **section** nicht auf die Darstellung der Seite aus. Die Zahlen erscheinen wegen der nummerierten Liste (**ol**) in B (mehr über Listen in Kapitel 15).

Der Einsatz des `section`-Elements

Ich zitierte die HTML5-Definition des **section**-Elements direkt aus dem Original. Das **section**-Element wurde in der Webcommunity ziemlich heiß diskutiert, da sein Einsatz scheinbar offen interpretiert wird.

Will man entscheiden, wann man **section** einsetzt, ist es hilfreich, auf den Teil mit der „thematischen Gruppierung“ zu achten. Das ist ein weiterer Grund, warum sich **section** von **div** unterscheidet. Um sich zwischen **section** und **article** zu entscheiden, sollten Sie bedenken, dass **section** eher organisierend und strukturierend gedacht ist, während **article** eine eigenständige Komposition ist.

Wie bereits in Kapitel 1 erwähnt, gibt es nicht immer eine einzige richtige Wahl beim Auszeichnen von Inhalten, sondern nur meistens. Bei anderen Gelegenheiten bleibt es eine persönliche Entscheidung, welche HTML-Elemente den Inhalt am besten beschreiben.

Überlegen Sie also gut, wann Sie **section** wählen, aber zerbrechen Sie sich nicht jedes Mal den Kopf, ob Sie dieses oder jenes Element immer exakt richtig verwenden. Manchmal ist das auch eine subjektive Entscheidung und in jedem Fall funktioniert Ihre Seite auch weiterhin. Außerdem wird Sie schon niemand deswegen mitten in der Nacht herausklingeln.

Ein aside angeben

Manchmal haben Sie bestimmte Inhalte, die zum Hauptinhalt Ihrer Seite eine Beziehung haben, aber auch eigenständig sind (vom Begrifflichen her, wenn nicht gar visuell) (A und B). Wie können Sie das semantisch verdeutlichen?

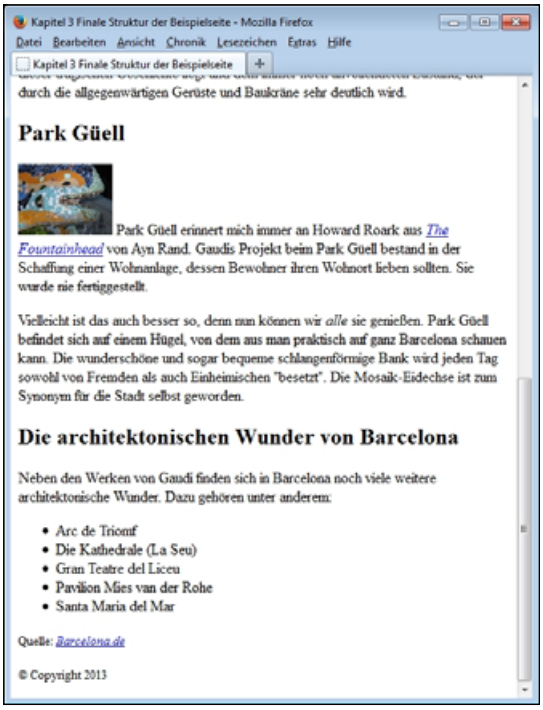
A **`aside`** stellt hier Infos über architektonische Sehenswürdigkeiten von Barcelona vor und gehört zum Inhalt über Antoni Gaudí, der den Schwerpunkt der Seite bildet. Aber es könnte auch ohne diesen Kontext für sich stehen. Ich hätte das ebenso im **`article`** verschachteln können, da beides zusammenhängt, aber ich will es lieber hinter dem **`article`** aufführen, damit ich es später visuell als Sidebar mit CSS gestalten kann C. Das Attribut **`role="complementary"`** für das **`aside`** ist optional, verbessert aber die Zugänglichkeit. Weitere Infos im letzten Tipp.

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

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

    ... [restlicher Artikel] ...
    </article>
</main>

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

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

aside (wörtlich: Nebenbemerkung) kann ein hervorgehobenes Zitat sein, eine Seitenleiste C, ein Kasten mit zum Artikel gehörigen Links auf einer News-Site, Gruppen von **nav**-Elementen (z.B. eine Blogroll), Twitter-Feeds oder eine Liste von Produkten auf einer kommerziellen Site.

Üblicherweise stellt man sich das **aside** wie eine Sidebar (Seitenleiste) vor, je nach Kontext kann ein **aside** auch woanders auf der Seite stehen. Ein innerhalb des primären Inhalts einer Seite verschachteltes **aside** (anstatt es außerhalb in einer Seitenleiste zu platzieren) sollte spezifisch zu diesem Inhalt gehören und nicht nur zur Seite insgesamt. Beispiel 1 in „Weitere Beispiele mit **aside**“ auf Seite 80 demonstriert dies.



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

C Wenn Sie die fertige Seite mit CSS formatieren, können Sie das `aside` („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 ein zweispaltiges CSS-Layout erstellt wird).

Ein aside erstellen

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

TIPP Platzieren Sie den Inhalt der Sidebar im HTML nach dem `main`-Inhalt der Seite A. Für SEO und Zugänglichkeit ist es besser, den wichtigsten Inhalt zuerst aufzuführen. In welcher Reihenfolge die Inhalte im Browser dargestellt werden, können Sie mit CSS ändern.

TIPP Nehmen Sie das `figure`-Element (siehe Kapitel 4) und kein `aside` für Abbildungen, die zum Inhalt gehören, z.B. Diagramme, Schaubilder oder eingefügte Fotos 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“ auf Seite 90.

Weitere Beispiele mit `aside`

Wie erwähnt, kann **`aside`** innerhalb oder außerhalb des Hauptinhalts erscheinen.

Beispiel 1 (verschachtelt im zugehörigen Hauptinhalt):

```
...
<body>
<main role="main">
    <article>
        <h1>Die Vielfalt von Papua-Neuguinea</h1>
        ... [Inhalt des Artikels] ...
        <aside>
            <h2>Kurzinfos über Papua-Neuguinea</h2>
            <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>
</main>
</body>
</html>
```

Im gleichen Text kann z.B. ein Zitat aus dem Artikeltext erscheinen. Das wäre dann auch ein **`aside`**. Oder es könnte ein **`aside`** mit „Verwandte Storys“ geben mit einer Link-Liste auf andere Berichte über das Land. Alternativ könnte dieses **`aside`** auch in einem anderen Seitenbereich stehen statt umschlossen von **`article`** in **`main`**.

Sie haben bereits ein Beispiel eines solchen **`aside`** in einer Sidebar gesehen (A und C). Nun schauen wir uns als Beispiel ein Design-Portfolio oder eine Gruppe Fallstudien an, bei dem/der jede HTML-Seite jeweils ein Projekt fokussiert. Sie bieten dann (in einem **`nav`** verschachtelte) Links auf andere Projektseiten in einer benachbarten Spalte (wird vom CSS gesteuert und nicht bloß durch die Anordnung des Codes wie in Beispiel 2).

Weitere Beispiele mit `aside` *(Fortsetzung)*

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

```
...
<body>
<main role="main">
    <article>
        <h1>... [Name des Projekts] ...</h1>
        <figure>... [Projektfoto] ...</figure>
        <p>... [Projektbericht] ...</p>
    </article>
</main>

<!-- dieses aside ist nicht im Hauptinhalt verschachtelt -->
<aside>
    <h2>Andere Projekte</h2>
    <nav>
        <ul>
            <li><a href="habitat-fuer-humanitaet.html">Habitat für Humanität-
            →Broschüre</a></li>
            <li><a href="royal-philharmonic.html">Royal Philharmonic Orchestra
            →Website</a></li>
            ...
        </ul>
    </nav>
</aside>
</body>
</html>
```

Dieses **`aside`** steht außerhalb des **`article`**, weil es mit der Seite insgesamt teilweise zu tun hat, aber nicht speziell mit dem Inhalt des **`article`**.

Einen Footer erstellen

Eine Fußzeile (Footer) stellen Sie sich wahrscheinlich wie die Seitenfußzeile vor, also dort, wo sich meist Copyright-Hinweise oder Links auf Datenschutzhinweise u.Ä. finden. Das **footer**-Element von HTML5 passt, ist aber wie **header** auch woanders einsetzbar.

Das **footer**-Element repräsentiert einen Footer für das nächstgelegene **article**-, **aside**-, **blockquote**-, **body**-, **details**-, **fieldset**-, **figure**-, **nav**-, **section**- oder **td**-Element, in dem es verschachtelt ist. Es ist nur dann der Footer für die *ganze* Seite, wenn der nächste Vorfahr **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 z.B. einen Anhang, Index oder längere Lizenzbestimmungen etc.

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

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

<main role="main">
    <article>
        <h1 id="gaudi">Der Architekt von
        →Barcelona</h1>
        ... [restlicher Artikel] ...

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

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

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

<footer>
    <p><small>&copy; Copyright All About
    →Gaudí</small></p>
</footer>
</body>
</html>
```

B Dieser Footer erscheint ganz unten auf der Seite nach dem `aside`. Das `footer`-Element selbst zwingt dem Text standardmäßig keine Formatierung auf. Hier ist der Copyright-Hinweis kleiner als normaler Text, weil er 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 in dem Element, 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, Autor oder ähnliche Elemente. Siehe die ersten beiden Beispiele in „Weitere `footer`-Beispiele“ auf Seite 84.

TIPP Ein `footer`-Element muss nicht unbedingt am Ende des Elements stehen, das es enthält (ist aber meist so).

TIPP Man darf in einem `footer` keinen `header` oder einen anderen `footer` verschachteln. Außerdem können Sie keinen `footer` in einem `header`- oder `address`-Element verschachteln.

TIPP In „Generische Container erstellen“ auf Seite 85 erfahren Sie mehr, wie `footer` eine der Rollen des `div`-Elements aus der Zeit vor HTML5 ersetzt hat.

TIPP In „Mit ARIA die Zugänglichkeit verbessern“ auf Seite 90 erfahren Sie, warum man `role="contentinfo"` mit dem `footer` für eine ganze Seite nehmen sollte. Es wäre auch passend, das beim `footer` in A aufzunehmen, weil es den Footer für die ganze Seite repräsentiert. Aber ich habe es weggelassen, damit man nicht denkt, dass `role="contentinfo"` für alle `footer`-Elemente richtig ist. In „Weitere `footer`-Beispiele“ sehen Sie ein Beispiel, das sowohl den Unterschied zeigt als auch die Rolle korrekt anwendet.

Weitere `footer`-Beispiele

Sie haben ein kleines Beispiel gesehen, wo ein Footer für die ganze Seite gilt (A und B). Abbildung C zeigt ein weiteres Beispiel für einen Seiten-Footer, aber mit mehr Inhalt.

Abbildung D demonstriert einen **`footer`** im Kontext eines Seitenabschnitts (in diesem Fall ein **`article`**) und einen zweiten **`footer`** für die ganze Seite („Weitere **`article`**-Beispiele“ auf Seite 74 erklärt, welchen Geltungsbereich das **`address`**-Element hier hat).

Beachten Sie, dass nur der Seiten-**`footer`** die optionale (aber empfohlene) **`role="contentinfo"`** bekommt. Über diese Rolle erfahren Sie mehr in „Mit ARIA die Zugänglichkeit verbessern“ auf Seite 90.

C Der Footer für eine ganze Seite enthält oft einen Copyright-Hinweis und Links, die nicht zur globalen Navigation im **`header`** der ganzen Seite gehören.

```
...
<body>
... [Seitenkopf mit globaler Navigation] ...
... [Seiteninhalt] ...

<!-- Dies ist ein Seiten-Footer, weil der →nächste
Vorfahr body ist -->
<footer role="contentinfo">
    <p><small>&copy; Copyright 2014 The
    →Corporation, Inc.</small></p>

    <ul>
       <li><a href="nutzungsbedingungen.html">
       →Nutzungsbedingungen</a></li>
       <li><a href="datenschutzrichtlinien.
       html">
       →Datenschutzrichtlinien</a></li>
    </ul>
</footer>
</body>
</html>
```

D Der erste **`footer`** gehört zum **`article`**, weil er darin verschachtelt ist. Der zweite **`footer`** ist für die gesamte Seite. Nehmen Sie **`role="contentinfo"`** nur für den Footer der Seite, also nur einmal pro Seite.

```
...
<body>
...
<article>
    <h1>... [Artikelüberschrift] ...</h1>
    <p>... [Inhalt des Artikels] ...</p>

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

<!-- Der Seiten-Footer -->
<footer role="contentinfo">
    ... [Copyright usw.] ...
</footer>
</body>
</html>
```

A Dieses Design erziele ich ohne `div`-Element. Doch weil ich ein `div` um den gesamten Inhalt der Seite legte B, habe ich nun einen generischen Container zum Formatieren (siehe die Ergebnisse in C).

Generische Container erstellen

Manchmal müssen Sie ein Inhaltssegment in einen Container legen, weil Sie CSS-Formatierungen anwenden oder einen Effekt mit JavaScript erzielen wollen. Ohne wäre Ihre Seite einfach nicht so gut A. Aber bei genauerem Blick auf den Inhalt stellen Sie fest, dass Elemente wie **`article`**, **`section`**, **`aside`** oder **`nav`** semantisch nicht angemessen sind.

weiter auf der nächsten Seite

Sie brauchen also einen generischen Container ohne jegliche semantische Bedeutung. Dieser Container ist das **div**-Element (steht für division, also Abschnitt) B. Über dieses **div** wenden Sie die gewünschte Formatierung C oder den gewollten JavaScript-Effekt an. Lesen Sie im Kasten „Über **div** und wie man es in HTML5 einsetzt“ auf Seite 88, wann Sie auf Ihren Seiten mit **div** arbeiten sollten.

C Ein **div**-Element hat standardmäßig keine eigene Formatierung, außer dass es in einer neuen Zeile beginnt D. Doch Sie können **div** formatieren, um Ihre Designs zu implementieren. Hier füge ich einen hellen Hintergrund und Schlagschatten ins **div** ein. So setze ich das **body**-Element vor einen Hintergrund mit rotem Farbverlauf, um den Inhalt hervorzuheben. Wie ich das gemacht habe, können Sie dem HTML und CSS dieser Seite unter http://goo.gl/sBYdTn entnehmen.

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

    <main role="main">
       <article>
          <h1 id="gaudi">Der Architekt
          →von Barcelona</h1>
          ... [restlicher Artikel] ...

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

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

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

    <footer role="contentinfo">
       <p><small>&copy; Copyright All About
       →Gaudí</small></p>
    </footer>
</div>
</body>
</html>
```

D Die gleiche Seite – aber weder für das `div` noch für Überschriften, Absätze oder andere Elemente wurde CSS angewendet. Wie Sie sehen, schafft es das `div` auf sich allein gestellt 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`, `main`, `article`, `section`, `aside`, `nav`, `h1–h6`, `p` und viele andere Elemente auch wird `div` standardmäßig automatisch in eine neue Zeile gestellt D.

TIPP `div` ist ebenfalls praktisch, wenn man bestimmte Interaktionen oder Effekte mit JavaScript einbauen will. Damit können Sie z.B. 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 ohne semantischen Wert. Das `span`-Element ist das Gegenstück zu `div`. Während `div` ein Container ohne Semantik für inhaltliche Blöcke ist, ist `span` (schreibt man `<span>hier ist der Inhalt </span>`) nur für Text gedacht, z.B. innerhalb eines `p`-Elements. Mehr über `span` erfahren Sie in Kapitel 4.

TIPP Wie Sie Landmark Roles mit `div` verwenden, erfahren Sie in „Mit ARIA die Zugänglichkeit verbessern“ auf Seite 90.

Über `div` und wie man es in HTML5 einsetzt

Von den in diesem Kapitel vorgestellten strukturellen Elementen ist **`div`** neben **`h1`**-**`h6`** das einzige noch aus der Zeit vor HTML5. 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, um alles mit CSS zu formatieren. Aber damals wie heute hatte **`div`** keine semantische Bedeutung.

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

Schauen wir uns einige Anlässe an, bei denen **`div`** die richtige Wahl ist.

Einen 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? Für das **`main`**-Element wende ich CSS an, damit es als Spalte 1 und das **`aside`**-Element als Spalte 2 dargestellt wird.

Meistens haben Spalten allerdings mehr als nur einen inhaltlichen Bereich. Vielleicht soll 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 sollen gar ganz andere Elemente eingefügt werden.

Dafür gruppieren Sie die Inhalte der Sidebar in einem **`div`** E und formatieren dieses **`div`** dann. (Falls Sie denken, das ginge auch mit **`section`**: Dies ist nicht als generischer Container fürs Formatieren gedacht.) Ich zeige hier ein Diagramm F, mit dem Sie die Beziehung zwischen Code und dem möglichen CSS-Layout visualisieren können. Bedenken Sie aber, dass es sich dabei nur um eine Layoutmöglichkeit für dieses HTML handelt: CSS ist sehr leistungsfähig.

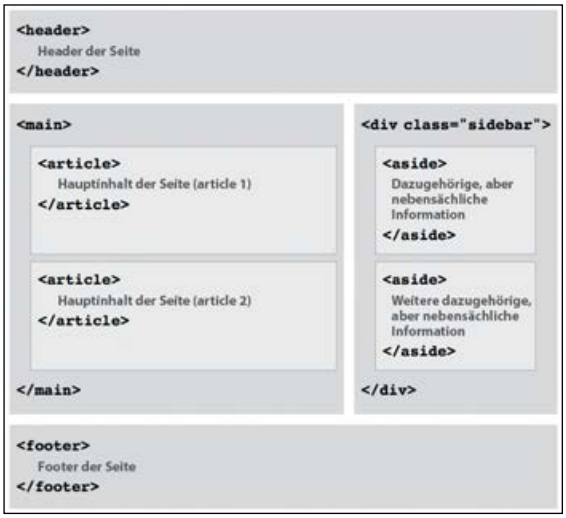
Es ist also recht üblich, dass ein **`div`** um jede Gruppe von Inhalten steht, die Sie als eine Spalte oder auch als Modul innerhalb einer Spalte formatieren wollen. Was dann *da* jeweils in die **`div`**-Elemente hineinkommt, kann abhängig von den jeweiligen Seiteninhalten sehr variieren. Vergessen Sie nicht, dass **`article`**, **`section`**, **`aside`** und **`nav`** als primäre semantische Container für Inhaltsbereiche praktisch überall auftauchen können. Und das gilt auch für **`header`** und **`footer`**. Überinterpretieren Sie nicht die Tatsache, dass im Beispiel (E und F) nur **`article`**s im **`main`**-Inhaltsbereich und in der Sidebar **`aside`**s stehen.

Um es deutlich zu sagen: Auf der Suche nach einem Container sollte **`div`** Ihre letzte Zuflucht sein, da es keinen semantischen Wert hat. Meist nimmt man richtiger **`header`**, **`footer`**, **`main`** (einmal), **`article`**, **`section`**, **`aside`** und vielleicht noch **`nav`**. Doch Sie sollten diese *nicht* nehmen, bloß um **`div`** zu vermeiden, wenn das semantisch unpassend ist. **`div`** hat seinen Platz, aber Sie sollten es nur beschränkt nutzen.

Nachdem das nun geklärt ist, verweise ich auf eine Situation, in der man berechtigterweise statt der neuen HTML5-Elemente **`div`** für alle (oder die meisten) Container einer Seite nimmt. Mehr Infos darüber finden Sie in „HTML5-Elemente in älteren Browsern formatieren“ in Kapitel 11 auf Seite 284.

E Diese Seite hat ein **div**, das die ganze Seite beinhaltet, plus eines für den Inhalt der Sidebar. Das **div** mit **class="sidebar"** (hier wählen Sie den Klassennamen beliebig) umgibt den Inhalt, den Sie als Spalte 2 darstellen. Dann nehmen Sie das **class** in Ihrem CSS, um jedes spezifische **div** präzise zu formatieren. Das **main**-Element wird für Spalte 1 formatiert.

```
...
<body>
<!-- Beginn Seiten-Container -->
<div class="container">
    <header role="banner">
       ...
    </header>

    <!-- Spalte 1, wenn CSS angewendet wird
    →-->
    <main role="main">
       <article>
          ...
       </article>

       <article>
          ...
       </article>

       ... [weitere Abschnitte nach Bedarf] ...
    </main>
    <!-- Ende Spalte 1 -->

    <!-- Spalte 2, wenn CSS angewendet wird
    →-->
    <div class="sidebar">
       <aside role="complementary">
          ...
       </aside>

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

       ... [weitere Abschnitte nach Bedarf] ...
    </div>
    <!-- Ende Spalte 2 -->

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



F Dieses Diagramm illustriert, wie der Code in E (minus die **role**-Attribute) zu einem CSS-Layoutkonzept passt. Dieses Arrangement ist sehr verbreitet, aber auch nur eine der vielen Möglichkeiten, die Sie bei gleichem HTML durch CSS haben. Werfen Sie einen Blick in den Abschnitt „Mit ARIA die Zugänglichkeit verbessern“ und erfahren Sie, wie man Semantik und Zugänglichkeit Ihrer Seiten optimiert.

Mit ARIA die Zugänglichkeit verbessern

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, die Sie in Ihren Seiten verwenden können, bis Sprachen wie HTML eigene semantische Äquivalente gefunden haben.

Sie haben wohl gemerkt, dass die meisten Beispiele dieses Kapitels ein oder mehrere ARIA-`role`-Attribute enthalten, und das soll jetzt also näher erklärt werden! Oh, vorher noch eine Kleinigkeit. Wir kommen gleich zu `role`, aber zuerst fassen wir unser Wissen über Accessibility zusammen.

Die Zugänglichkeit oder Accessibility soll den Inhalt Ihrer Seiten für alle Besucher verfügbar machen. Manche Besucher verlassen sich auf assistive Technologien wie Screenreader (Bildschirmlesegeräte), um Ihre Seiten zu lesen (siehe Kasten „Screenreader ausprobieren").

Sie als überlegter, verantwortungsbewusster Webentwickler sollten Ihre Sites zugänglich (auch barrierefrei genannt) gestalten. Außerdem ist es auch gut für Sie – denn warum sollten Sie Besucher von Ihren Inhalten *aussperren*?

Zum Glück ist es meist recht einfach, Ihre Seiten zugänglich zu gestalten. Die Zugänglichkeit wird einfach dadurch verbessert, dass man Inhalte mit dem HTML auszeichnet, das sie am besten beschreibt. Falls Sie bereits so arbeiten, ist das großartig. In diesem Abschnitt erfahren Sie, wie Sie Ihren Besuchern mit einigen einfachen Attributen im HTML weiterhelfen.

Screenreader ausprobieren

Die folgenden Screenreader zählen zu den am häufigsten verwendeten Bildschirmlesegeräten. VoiceOver gibt es allerdings nur für OS X.

- JAWS: kostenlose Testversion erhältlich unter www.freedomscientific.com
- NVDA: kostenlos unter www.nvda-project.org
- VoiceOver: kostenlos als Bestandteil von OS X und iOS 4+. Tippen Sie ⌘-F5 in OS X, um das Programm zu starten oder zu stoppen. Unter https://support.apple.com/kb/HT3598 finden Sie Hinweise, wie man es auf iOS-Geräten nutzt.
- Window-Eyes: kostenlose Testversion erhältlich unter www.gwmicro.com

Ich kann gar nicht nachdrücklich genug dazu raten, dass Sie zumindest einen davon mal ausprobieren. So können Sie besser würdigen, wie sich Ihre semantischen HTML-Entscheidungen auf die User Experience bei Screenreadern auswirken. Besser noch, wenn Sie Ihre Seiten als Teil Ihres normalen Entwicklungsprozesses beim Erstellen von Websites in einem Screenreader testen.

Landmark Roles

Vorhin habe ich erwähnt, dass ARIA semantische Lücken im HTML füllt. Mit welchem HTML-Markup lassen Sie beispielsweise einen Screenreader wissen, wie er zum Footer Ihrer Seite wechseln kann? Wenn Sie den Inhalt mit `footer` auszeichnen, reicht das nicht – man kann auf der Seite ja mehr als einen `footer` einbauen.

HTML verfügt dafür über keine Lösung, aber die *Landmark Roles* von ARIA! Landmark Roles identifizieren wichtige Bereiche der Webseite, sodass die Nutzer von Screenreadern direkt dorthin navigieren können. Und das geben Sie mit dem Attribut `role` an.

Wie Sie Tabelle 3.1 entnehmen, gibt es Überlappungen zwischen ARIA und HTML5 (HTML5 versuchte ebenfalls, mit den erwähnten neuen Elementen Lücken zu füllen). Es gibt zwar Schnittpunkte zwischen Landmark Roles und HTML5-Elementen, aber ARIA wird von Screenreadern besser unterstützt. Also machen Sie wie gehabt mit dem Erstellen von HTML weiter (einschließlich der neuen Elemente) und bauen Sie ARIA-Roles für die bessere Zugänglichkeit Ihrer Seiten ein.

Tabelle 3.1 **Einige der verfügbaren ARIA Landmark Roles**

Landmark Roles	Wie und wann man sie verwendet
`role="banner"` Site-orientierte Inhalte bestehen üblicherweise aus solchen Dingen wie Logo oder Identität des Sponsors der Site und einem für die ganze Site gedachten Suchwerkzeug. Ein Banner erscheint üblicherweise oben auf der Seite und erstreckt sich meist über die ganze Breite.	Wenden Sie diese Rolle auf das `header`-Element an, das den Header für Ihre gesamte Seite enthält, und nehmen Sie es nur einmal pro Seite.
`role="navigation"` Eine Sammlung navigationsbezogener Elemente (meist Links), um im Dokument oder damit zusammenhängenden Dokumenten zu navigieren.	Dies entspricht dem `nav`-Element. Setzen Sie diese Rolle in jedes `nav`-Element (oder das alternative Element, falls Sie kein `nav` nehmen), das eine Hauptgruppe der Navigation umgibt. Sie können diese Rolle mehr als einmal pro Seite verwenden, aber sollten sie wie `nav` nicht überbeanspruchen.
`role="main"` Der Hauptinhalt eines Dokuments.	Dies entspricht dem `main`-Element. Fügen Sie diese Rolle in `main` ein (oder in das Element, das Sie stattdessen nutzen, wie z.B. `div`). Verwenden Sie es nur einmal pro Seite.
`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.	Dies entspricht dem `aside`-Element. Fügen Sie diese Rolle einem `aside` oder `div` hinzu (solange diese Elemente nur ergänzende Inhalte aufweisen). Sie können mehr als eine `complementary`-Rolle auf jeder Seite einbauen, aber übertreiben Sie es nicht.
`role="contentinfo"` Eine große, wahrnehmbare Region, die Infos über das übergeordnete Dokument enthält. Zum Beispiel gehören in diesen Bereich der Seite Copyright-Angaben und Links auf Datenschutzbestimmungen.	Fügen Sie diese Rolle in den Footer Ihrer ganzen Seite ein (üblicherweise in einem `footer`-Element) und nehmen Sie es pro Seite nur einmal.

Das geht sehr einfach. Ⓐ ist das gleiche Beispiel aus „Generische Container erstellen“, außer dass ich noch ein `nav`-Element und die Landmark Roles ergänzt habe.

Tabelle 3.1 enthält Auszüge der Definitionen für Landmark Roles aus der ARIA-Spezifikation (www.w3.org/TR/wai-aria/roles#landmark_roles), im Anschluss daran meine Empfehlungen zur Nutzung (siehe erster Tipp über die anderen drei Roles). Die Beschreibungen klingen vertraut, weil sie sehr dem ähneln, wie Sie bestimmte HTML-Elemente nutzen (Ⓐ und Ⓑ).

Um es deutlich zu sagen: Ihre Seiten funktionieren auch ohne ARIA Landmark Roles, aber die User Experience wird bei assistiver Technologie verbessert. Aus diesem Grund empfehle ich sie. Ich habe sie bei verschiedenen anderen Beispielen des Buchs integriert wie auch auf der Site für das Buch.

TIPP Tabelle 3.1 enthält drei der Landmark Roles nicht. Die Rolle **`form`** ist semantisch redundant zum **`form`**-Element, **`search`** markiert ein Such**`formular`**, und **`application`** ist für fortgeschrittene Anwendungsmöglichkeiten gedacht. In „Textboxen für E-Mail, Suche, Telefon und URL erstellen“ auf Seite 440 in Kapitel 16 finden Sie ein Beispiel mit **`role="search"`**.

Ⓐ Auf dieser Seite werden alle fünf Landmark Roles aus Tabelle 3.1 verwendet – sechs insgesamt.

```
...
<body>
<!-- Beginn Seiten-Container -->
<div class="container">
    <header role="banner">
       ...
       <nav role="navigation">
          ... [ul mit Links] ...
       </nav>
    </header>

    <!-- Spalte 1, wenn CSS angewendet wird -->
    <main role="main">
       <article>
          ...
       </article>

       <article>
          ...
       </article>

       ... [weitere Abschnitte nach Bedarf] ...
    </main>
    <!-- Ende Spalte 1 -->

    <!-- Spalte 2, wenn CSS angewendet wird -->
    <div class="sidebar">
       <aside role="complementary">
          ...
       </aside>

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

       ... [weitere Abschnitte nach Bedarf] ...
    </div>
    <!-- Ende Spalte 2 -->

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

TIPP Treiben Sie es auf Ihren Seiten mit den Landmark Roles nicht zu bunt. Setzen Sie zu viele davon ein, wird Ihre Seite für Screenreader-Nutzer zu „geschwätzig“, was den Wert der Landmarks und die User Experience allgemein mindert.

TIPP Der Accessibility-Experte Steve Faulkner stellt Landmark Roles unter blog.paciellogroup.com/2013/02/using-wai-aria-landmarks-2013/ vor. (Ähnlichkeiten zwischen seiner und der Tabelle 3.1 sind unbeabsichtigt.) Dort finden Sie auch ein Video seiner Kollegin Léonie Watson, das demonstriert, wie der Nutzer eines Screenreaders auf einer Seite navigiert. Sehr empfehlenswert!

TIPP Bei WebAIM gibt es regelmäßig Umfragen unter Screenreader-Nutzern, um deren Präferenzen und Herausforderungen bei der Internetnutzung besser zu verstehen. Die neuesten Ergebnisse sind lesenswert: webaim.org/projects/screenreadersurvey4/.

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

TIPP Sie können diese ARIA-Rollenattribute in Ihren CSS-Selektoren nutzen, um die damit ausgezeichneten Elemente zu formatieren. In Kapitel 11 erfahren Sie mehr darüber.

B Dieses Diagramm ähnelt dem aus „Generische Container erstellen“. Es soll helfen, übliche Einsatzmöglichkeiten für bestimmte Landmark Roles zu visualisieren, aber *nicht* den Eindruck erwecken, dass sie sich irgendwie aufs Layout auswirken. Im `main`-Element könnten Sie beispielsweise die lokale Navigation in ein `nav` mit `role="navigation"` packen. Ohne Zusammenhang könnte das `div` mit `class="sidebar"` ein `role="complementary"` *statt* der `aside`-Elemente haben, falls sich die gesamte Sidebar als ergänzender Inhalt eignet.

Elemente mit einer Klasse oder ID benennen

Obwohl nicht zwingend, können Sie Ihren HTML-Elementen eine eindeutige ID bzw. eine (oder mehrere) bestimmte Klassen zuweisen Ⓐ.

Dann formatieren Sie diese Elemente je nach **id** oder **class**-Namen (allerdings rate ich davon ab, fürs Formatieren **id**s zu nehmen). Oder verknüpfen Sie das Element direkt über eine bestimmte **id** Ⓐ. Sie können mit JavaScript auf die Attribute **id** und **class** zugreifen, um spezielles Verhalten für bestimmte Elemente anzuwenden.

Ein Element mit eindeutiger ID benennen

Im öffnenden Tag des Elements tippen Sie **id=**"*name*" und das Element bekommt mit *name* seinen eindeutigen Namen Ⓐ. *name* kann praktisch alles sein, solange es nicht mit einer Zahl beginnt oder Leerzeichen enthält. Eine **id** darf auf einer Seite nur einmal vorkommen (sonst wäre sie ja nicht mehr eindeutig).

Einem Element eine Klasse zuweisen

Im Start-Tag des Elements tippen Sie **class=**"*name*", wobei *name* der Name der Klasse ist Ⓐ. Wollen Sie mehr als eine Klasse zuweisen, trennen Sie diese mit einem Leerzeichen: **class="***name anderername***"**. Anders als bei **id** können Sie auf einer Seite mehrere gleichnamige **class**-Elemente verwenden.

TIPP Wie man Stylesheets auf ein Element mit einer bestimmten Klasse oder ID anwendet, erfahren Sie in „Elemente mit einer Klasse oder ID auswählen“ auf Seite 220 in Kapitel 9. Doch ich empfehle nachdrücklich, sich fürs Formatieren an Klassen zu halten.

Ⓐ Die Links im **nav** zeigen auf die **id**s in den **h1-** und **h2**-Elementen. Geben Sie einem oder mehreren Elementen das Attribut **class**, um alle Elemente dieser Gruppe auf einmal zu formatieren. Die Klasse **architect** könnte z.B. zur Vereinheitlichung der Formatierung auf Inhalte über andere Architekten angewendet werden. Mit der **gaudi**-Klasse könnte spezieller Inhalt nur über diesen Architekten formatiert werden.

```
...
<body>
<div class="container">
    <header role="banner">
       <nav role="navigation">
          <ul>
             <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>

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

          <p>Viele Touristen kommen extra
          →nach Barcelona...</p>
          ...

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

          <h2 id="park-guell">Park Güell
          →</h2>
          ...
       </article>
    </main>
    ...
</div>
</body>
</html>
```

TIPP Siehe „Einen Link zu einem Anker erstellen“ auf Seite 176 in Kapitel 6, wie man ein Element mit einer `id` verknüpft. Es sprengt den Rahmen dieses Buchs zu erklären, wie man JavaScript bei Elementen mit einer `id` oder `class` einsetzt, aber ein Beispiel finden Sie in Kapitel 19.

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 (wird aber normalerweise so gehandhabt).

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.

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

TIPP Wenn `class`es und `id`-Namen aus mehreren Wörtern bestehen, trennt man sie mit Bindestrich, z.B. `class="footer-page"`.

TIPP Wählen Sie aussagekräftige (d.h. semantische) Namen für Ihre `id`s und `class`es, egal wofür sie sind. Wollen Sie z.B. eine `class` zur Formatierung nutzen, vermeiden Sie die Präsentation beschreibende Namen, etwa `class="rot"`. `class="rot"` ist eine schlechte Wahl, weil Sie vielleicht nächste 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 einfach. Dies wird Ihnen wahrscheinlich klarer, wenn Sie sich mehr mit CSS beschäftigen.

TIPP Sie können mit dem `class`-Attribut sogenannte *Mikroformate* nutzen. Unter http://microformats.org erfahren Sie mehr.

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 bis C). Es 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 Start-Tag des zu kennzeichnenden Elements fügen Sie **`title="`*`label`*`"`** ein, wobei ***`label`*** der kurze, beschreibende Text ist, der im Tooltipp erscheinen soll. Dieser Text wird auch im Screenreader vorgelesen (falls die Anzeige des **`title`** nicht vom Nutzer ausgeschaltet wurde).

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

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

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

B Wenn der Besucher auf das gekennzeichnete Element zeigt, erscheint der Titel als Tooltipp. Wenn Sie auf den Link Der Architekt von Barcelona zeigen ...

C ... sieht man „Erfahren Sie mehr über Antoni Gaudí“, weil es ein eigenes **`title`**-Attribut hat.

Kommentare einfügen

Mit Kommentaren in Ihren HTML-Dokumenten können Sie kennzeichnen, wo Bereiche beginnen oder enden, flechten für sich (oder spätere Bearbeiter) Hinweise ein, wozu ein bestimmter Codeabschnitt gedacht ist, verhindern die Darstellung von Inhalten u.v.m. A. Diese Kommentare erscheinen nur, wenn die Seite mit einem Editor oder über die Browser-Option Quelltext anzeigen geöffnet wird. Besucher sehen sie in ihrem Browser ansonsten nicht B.

B Kommentare sind auf der Seite unsichtbar. Entsprechend werden Inhalte, die Sie in Kommentarzeichen setzen, nicht dargestellt. Hier wird der erste Absatz im Code nicht gezeigt.

A In diesem Beispielcode stehen fünf Kommentare. Vier zeigen kombiniert Anfang und Ende von zwei Bereichen der Seite an (main und sidebar). Mit dem nächsten wird der erste Absatz „auskommentiert“, damit er auf der Seite nicht dargestellt wird (wenn Sie diesen Absatz auf lange Sicht ausschließen wollen, wäre es besser, ihn im HTML zu löschen).

```
...

<!-- ==== Beginn Hauptinhalt ==== -->
<main role="main">
    <article class="architect">
       <h1 id="gaudi">Der Architekt von
       →Barcelona</h1>

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

       <p>Seine Nonkonformität, die schon in
       →Jugendjahren sichtbar wurde ...</p>
       ...
    </article>
</main>
<!-- Ende Hauptinhalt -->

<!-- ==== Beginn Sidebar ==== -->
... [Inhalt der Sidebar] ...
<!-- Ende der Sidebar -->

...
```

Kommentar ins HTML einfügen

1. In Ihrer HTML-Seite tippen Sie dort, wo Sie einen Kommentar einfügen wollen, `<!--` ein.
2. Schreiben Sie den entsprechenden Kommentar.
3. Mit Eingabe von `-->` schließen Sie den Kommentar ab.

TIPP Üblicherweise fügt man zwecks einfacherer Bearbeitung zu Beginn und am Ende von größeren Codeabschnitten Kommentare ein (Webseiten können ganz schön lang werden). Ich nehme gerne ein auffälliges Format, wenn ich einen Kommentar beginne, und ein weniger deutliches fürs Ende, damit ich beim Überfliegen schnell diese beiden wichtigen Punkte im Code erkenne (A).

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

TIPP Kommentare werden ganz einfach sichtbar, wenn der User sich die Seite über die Option Quellcode anzeigen anschaut oder die Seite als HTML-Code speichert.

TIPP Kommentare dürfen nicht in andere Kommentare verschachtelt werden.