Kapitel 1
Einführung in das HTML-Universum

Egal ob Sie Einsteigerin, Entwickler, Programmiererin oder Blogger sind, als Leserin oder Leser werden Sie gewisse Ansprüche an dieses Buch stellen. In diesem Kapitel geht es zunächst darum, einige Formalitäten zu klären, die dieses Buch betreffen (bzw. nötig sind), und auszuführen, was Sie erwartet, bevor Sie mit der eigentlichen Praxis beginnen können.

Sicherlich haben Sie schon das Inhaltsverzeichnis dieses Buches überflogen. Dabei dürften Sie festgestellt haben, dass seine Schwerpunkte auf HTML und CSS liegen. Mit HTML lernen Sie die Auszeichnungssprache zur Erstellung von Websites kennen, und mit CSS erfahren Sie, wie Sie Websites designen und stylen. Darüber hinaus wird hier, mittlerweile unverzichtbar, die Webprogrammierung mit JavaScript behandelt.

In diesem Kapitel gehe ich es noch langsam an, und Sie erfahren Folgendes:

- welche Typen von Websites es gibt, welche Technologien dabei verwendet werden und welche Kenntnisse dafür nötig sind
- was dynamische und statische Websites sind
- welche grundlegenden Sprachen Sie als Webentwickler kennen und können sollten
- was Sie alles benötigen, um ein HTML-Dokument zu erstellen und es in einem Webbrowser anzeigen zu lassen
- wie Sie das HTML-Dokument auf Fehler hin überprüfen können

1.1 Ist dieses Buch überhaupt etwas für mich?

Dieses Buch richtet sich an *Einsteiger*, die zunächst einfach nur eine eigene Homepage erstellen oder sich mit den grundlegenden Webtechnologien vertraut machen wollen, wie auch *Webautoren*, die nach einer umfassenden Lektüre zu den aktuellen Themen HTML, CSS und JavaScript (und mehr) suchen.

Auch (künftige) *Entwickler* und *Programmierer* von Webanwendungen für Web-Templates (Vorlagen) oder dynamische Websites kommen nicht mehr um fundierte Kenntnisse rund um HTML herum.

Und selbst *Blogger* oder *Onlineverkäufer*, die eine Plattform nutzen, auf der man in der Regel den Inhalt in einem Formular ohne spezielle Kenntnisse eingeben kann und ein spezielles

System die Erstellung der Webdarstellung für den Betrachter generiert, können von tieferen Kenntnissen nur profitieren, um den Inhalt ordentlicher oder gegebenenfalls nach eigenen Bedürfnissen auszurichten bzw. anzupassen.

Und selbst wenn Sie noch nicht wissen, zu welcher Gruppe Sie gehören (wollen), haben Sie zumindest Ambitionen zur Webentwicklung (da Sie dieses Buch in den Händen halten). Mit dem Hintergrundwissen rund um HTML stehen Ihnen sehr viele Türen offen.

Von vorne nach hinten oder kreuz und quer?

Einsteigern in diese Materie empfehle ich, das Buch von vorne nach hinten durchzuarbeiten. Sofern dies möglich ist, sind die einzelnen Kapitel in diesem Buch so aufgebaut, dass so wenig wie möglich auf spätere Kapitel vorgegriffen wird. Natürlich lässt sich das nicht ganz vermeiden, um ein Thema zu erklären. In solch einem Fall finden Sie zumindest einen Querverweis auf das vorgegriffene Thema. Trotzdem habe ich weitgehend versucht, auf solche Querverweise im Buch zu verzichten.

Wiedereinsteiger oder erfahrenere Leser können das Buch kreuz und quer lesen und zu den einzelnen Themen blättern, wenn diese benötigt werden. Dieses Buch eignet sich somit auch gut als Nachschlagewerk.

1.2 Die verschiedenen Typen von Websites

An dieser Stelle folgt eine kurze Übersicht, welche gängigen Typen von Websites es heute gibt und wie diese erzeugt werden. Die Website-Typen zu trennen ist zunächst nicht so einfach, weil sie auch vom Ziel und vom Technologieansatz abhängen und einige Typen auch ineinander verschwimmen. Lässt man den Technologieansatz außen vor, können die Typen grob in sechs Kategorien eingeteilt werden:

- Webpräsenz (Homepage/Corporate Website)
- Blog/Magazin/Portfolio
- E-Commerce-Website
- Webplattform (Social-Media-Webseiten)
- Landingpage/Microsite
- Web-App

1.2.1 Webpräsenz

Eine Webpräsenz kann entweder eine private Homepage oder ein Webauftritt von Unternehmen, Vereinen, Behörden, Geschäftspersonen usw. sein. Bei Unternehmen, der Website einer Stadt oder gemeinnützigen Organisationen fällt hierbei dann häufig auch der Begriff *Corporate Website* (auch *Informational Website*). Gerade in der Businesswelt von kleineren Unternehmen oder Selbstständigen gehört es zum guten Ton, im Web mit Informationen, Angeboten, Kontaktmöglichkeiten etc. mit einer Webadresse präsent zu sein. Auch in Zeiten der sozialen Netzwerke wie Facebook und Co. erstellen und pflegen noch immer viele Privatleute eine eigene Homepage. Meistens finden Sie dort Näheres zur Person und deren Interessen. Allerdings kommt zurzeit, gerade bei jüngeren Leuten, die private Homepage aus der Mode und wird durch Facebook, Instagram oder Twitter ersetzt. Größere Unternehmen, Vereine, Anwälte, Künstler, Restaurants, Ärzte, Handwerker, Autoren usw. sind häufig neben einer Webpräsenz auch bei Facebook und Co. vertreten. In der Regel dienen solche Websites vorrangig der Information für Besucher.

Nötige Kenntnisse für eine Webpräsenz

Zum Erstellen von privaten Homepages oder Webauftritten für Unternehmen, Vereine usw. sind gute Kennnisse in HTML, CSS und JavaScript sinnvoll, wenn Sie die Website manuell erstellen wollen. Gerade wenn es sich dabei um den Webauftritt von kleineren Unternehmen oder öffentlichen Persönlichkeiten wie Künstler, Anwälte usw. handelt, sollte der Code unbedingt fehlerfrei sein. Wie erwähnt, gilt dies vorwiegend nur, wenn Sie eine statische Website erstellen. Gerade Corporate Websites von Unternehmen enthalten neben statischem Inhalt auch dynamische Elemente wie News oder Kontaktformulare. Es gibt mittlerweile viele Unternehmen und Privatpersonen, die fertige (dynamische) Systeme (Stichwort *Content-Management-System*) wie z. B. WordPress für ihre Webpräsenz verwenden. Ist ein solches System erst einmal eingerichtet, sind tiefer greifende Kenntnisse nicht mehr unbedingt nötig, weil eine webbasierte Software verwendet wird und die Formatierung ähnlich wie bei einer Office-Anwendung realisiert werden kann. Auch hier sind HTML- und CSS-Kenntnisse jedoch sinnvoll und hilfreich.

1.2.2 Blog/Onlinemagazin/Portfolio

Die Bezeichnung *Blog* ist aus der Mischung der Begriffe *Web* und *Log* entstanden. Die Person, die diesen Blog führt, wird häufig als *Blogger* bezeichnet.

Ein Blog ist eine Website mit gewöhnlich chronologisch sortierten Einträgen, die voneinander getrennt sind. Häufig ist ein Blog auch die Startseite einer Webpräsenz, auf der Besucher die neuesten Beiträge und aktuelle Informationen zu einem bestimmen Thema, einem Unternehmen usw. lesen können. Ebenso sind moderierte Kommentare und Diskussionen für und mit den Besuchern oder das Teilen der Beiträge auf sozialen Medien möglich. Zu dieser Kategorie passen auch sogenannte Magazin-Webseiten, die in der Regel ebenfalls viele aktuelle Artikel, Fotos und Videos enthalten sowie informativ und lehrreich sind. Was früher die Zeitschriftenbranche war, sind heute die *Onlinemagazine*.

Hier verschwimmen oft die Begriffe Webpräsenz und Blog miteinander. Viele Unternehmen oder Privatpersonen verwenden für die Webpräsenz häufig ein fertiges System wie z. B. WordPress oder Drupal. Neben einem Blog findet man auf diesen Websites auch die üblichen Informationen wie Kontaktmöglichkeiten, Angebote und noch vieles mehr. Solche Blogsysteme eignen sich allerdings nicht für jedes Unternehmen. So wird man in diskreteren Berufssparten wie denen der Anwälte und Ärzte wohl eher eine einfache Webpräsenz vorfinden. Viele kleinere Betriebe wie Handwerksfirmen oder auch Privatpersonen haben gar nicht die Zeit, regelmäßig einen Blog zu führen. Es sieht nicht gerade gut aus, wenn man die Website der Firma Mustermann besucht und der letzte Blogeintrag schon ein Jahr alt ist. Da fragt sich manch einer, ob die Firma überhaupt noch existiert. Die Blogkultur (oder auch Netzkultur) ist übrigens kein triviales Thema, das man hier mit ein paar Zeilen abhandeln kann. So lassen sich z. B. die Blogs noch in verschiedene Typologien und dann wiederum in verschiedene Betreiber (Privatpersonen, Körperschaften, Künstler usw.) aufteilen. Ein offizieller Unternehmensblog wird z. B. *Corporate Blog* (Stichwort: Corporate Website) genannt. Selbst das beliebte Twitter hat mit *Mikroblogging* einen eigenen Begriff.

Blogs und Onlinemagazine unterscheiden sich von Webpräsenzen oder Corporate Websites allerdings im Wesentlichen darin, dass sie die Besucher nicht nur über das Unternehmen oder die Person informieren, sondern regelmäßig neue und relevante Inhalte mit Mehrwert präsentieren.

Um einen Blog bzw. eine Magazin-Website zu erstellen, stehen Ihnen zwei Möglichkeiten zur Verfügung: Entweder Sie installieren eine Blogsoftware auf einem Server oder Webspace, oder Sie verwenden eine fertige gehostete Lösung. Die Installation einer Blogsoftware wie z. B. WordPress auf einem Server oder Webspace ist wesentlich flexibler, weil Sie hier den Blog um viele weitere vorhandene Module und Vorlagen erweitern können. Und ist mal kein passendes Modul vorhanden, könnten Sie selbst eines programmieren.

Ebenfalls in diese Kategorie gehören noch die *Portfolio-Websites* für Designer, Fotografen, Künstler und Kreative, auf denen diese ihre Arbeiten visuell präsentieren können. Häufig werden dabei Website-Themes mit minimalistischen Designs für eine Blogsoftware (wie beispielsweise WordPress) installiert. Die Textmenge ist bei solchen Websites häufig deutlich reduziert.

Was sind PHP und MySQL?

PHP ist eine Skriptsprache, die eine ähnliche Syntax wie die Programmiersprache C besitzt und vorwiegend für die Erstellung von dynamischen Websites und Webanwendungen benutzt wird.

MySQL ist ein relationales Datenbankverwaltungssystem, das meistens für dynamische Webauftritte im Internet in Verbindung mit dem Webserver Apache und der Skriptsprache PHP eingesetzt wird.

Für die Installationen müssen allerdings auf dem Server bzw. Webspace bestimmte Voraussetzungen erfüllt sein (z. B. Zugriff auf PHP und eine MySQL-Datenbank), und ein grundlegendes Wissen darüber ist von Vorteil, wenn es mal nicht auf Anhieb mit der Installation klappen sollte. Bei einer gehosteten Lösung wie z. B. *www.blogger.com* oder *www.tumblr.com* müssen Sie sich diesbezüglich wenig Gedanken machen und können gewöhnlich nach einer kurzen Anmeldung und Auswahl der Vorlage gleich losbloggen.

Nötige Kenntnisse für einen Blog

Auch hier sind Kenntnisse von HTML, CSS und JavaScript von Vorteil, um diverse Feinheiten selbst in die Hand zu nehmen. Besonders CSS-Kenntnisse sind extrem hilfreich, weil Sie damit häufig das komplette Webdesign ändern können. Generell werden die Beiträge solcher Blogs mit einer webbasierten Software erstellt. Hierbei handelt es sich um eine Webanwendung, die im Browser läuft und sich meistens recht einfach wie eine Office-Anwendung zur Texterstellung verwenden lässt. Im Grunde müssen Sie sich bei solchen Blogsystemen nur um den Inhalt kümmern. Das Layout, Speichern, Hinzufügen oder Archivieren von Blogartikeln übernimmt das Blogsystem für Sie. Sind Sie bereits Entwicklerin oder Programmierer und im Umgang mit z. B. PHP und MySQL vertraut oder wollen künftig programmieren lernen, um eigene Module zu schreiben, sind auf jeden Fall tiefer greifende Kenntnisse in HTML, CSS und JavaScript nötig.

1.2.3 E-Commerce-Websites – Geschäfte ohne Öffnungszeiten

Das Onlineshopping erfreut sich immer größerer Beliebtheit, und so ist es kein Wunder, dass viele Unternehmen mit einem Webshop vertreten sein wollen. Die Vorteile liegen ganz klar auf der Hand: rund um die Uhr geöffnet, weniger Personalkosten, keine Kosten und Miete für das Geschäft und die Einrichtung dafür und noch einige Gründe mehr.

In der Praxis verwendet man für einen Webshop eine fertige Software, da der Aufwand damit wesentlich geringer ist, um z. B. den Produktkatalog zu aktualisieren oder zu pflegen, und, noch wesentlich wichtiger, weil diese Webshop-Software bereits vielfach erprobt und somit wesentlich sicherer ist, was gerade beim Bezahlvorgang von besonderer Bedeutung ist und sein sollte.

Dank fertiger Webshop-Software kann ein solcher Onlinestore recht schnell von jedem Erwachsenen eingerichtet werden. Allerdings gibt es hier eine lange Liste von rechtlichen Voraussetzungen, die Sie unbedingt beachten müssen, damit der Shop rechtlich gültig ist. Das fängt mit der Impressumspflicht an, allgemeine Geschäftsbedingungen (AGB) müssen vorhanden sein, die Widerrufsbelehrung darf nicht fehlen, korrekte Angaben zur Lieferzeit und zu den Preisen und noch einiges mehr. Wer hier als Laie einen Onlineshop einrichtet, der sollte vielleicht noch einen Anwalt zur Beratung in Erwägung ziehen.

Abhängig vom Funktionsumfang der Webshop-Software kann ein solcher Onlinestore ziemlich teuer werden. Hier müssen Sie selbst einschätzen, was sich für Sie lohnt. Die Lösungen reichen von Komplettlösungen, die von Hosting-Providern angeboten werden, bis hin zu Profi-Webshop-Software zum Installieren auf einem Server bzw. Webspace. Die Preise variieren hier von 0 € bis zu fünfstelligen Summen. Häufig wird hierbei eine spezialisierte Software wie Shopify, Magento oder WooCommerce eingesetzt.

Nötige Kenntnisse für einen Webshop

Der Webshop wird gewöhnlich über eine zugangsgeschützte Nutzeroberfläche (meistens über den Webbrowser) ähnlich wie schon bei einem Content-Management-System für einen Blog bedient. Daher gilt hier dasselbe wie für einen Blog: Kenntnisse in HTML und CSS sind zwar nicht unbedingt nötig, aber von Vorteil, wenn man das Produkt besser präsentieren will.

Nicht für jeden ist es geeignet und sinnvoll, gleich einen eigenen Webshop einzurichten und zu eröffnen. Dies hängt davon ab, was Sie verkaufen wollen, und von der Größe des Unternehmens. Wer nur eine Handvoll Produkte verkaufen will und frisch in die E-Commerce-Welt einsteigt, für den kann es auch ausreichend sein, seine Produkte z. B. bei *www.ebay.de* anzubieten. Bedenken Sie, dass Sie, wenn der Webshop eingerichtet ist und eine Menge Geld investiert wurde, erst mal Besucher für Ihren Onlineshop benötigen. Ein Besucher allein ist wiederum noch lange kein Käufer.

1.2.4 Landingpage/Microsite

Eine *Landingpage* besteht in der Regel aus nur einer Webseite, die auf ein bestimmtes Ziel aus ist, die Besucher eine bestimmte Handlung (*Call-to-Action*) ausführen zu lassen. Das wäre zum Beispiel das Starten einer Testphase für ein Produkt, das Kaufen eines Produkts oder einfach die Kontaktaufnahme. Das Ziel einer solchen Landingpage ist es, den Besuchern alle Elemente eines Produkts auf einer Seite vorzustellen, damit aus ihnen potenzieller Kunden werden. Des Weiteren werden solche Seiten stark für Suchmaschinen optimiert, um gezielt über Social-Media-Kampagnen oder Werbung in Suchmaschinen die Zielgruppe zu erreichen.

Oftmals wird auch der Begriff *Microsite* als Synonym für eine Landingpage genannt, aber dies ist nicht ganz richtig, da eine Microsite eher wieder eine Informational Website ist, die aus wenigen Seiten besteht und sich ausschließlich mit einem bestimmen Thema befasst. Dies wird sehr häufig von Unternehmen verwendet, um ein einzelnes Produkt gezielt auf einer separaten Domain zu bewerben, anstatt das Produkt innerhalb einer umfangreichen Corporate Website zu platzieren.

Nötige Kenntnisse für eine Landingpage/Microsite

Eine Landingpage/Microsite können Sie theoretisch bereits mit HTML und CSS erstellen. Aber auch hier gibt es Webbaukästen, spezielle Plug-ins oder Themes für ein Content-Management-System, die einem die Arbeit komplett abnehmen. Aber auch JavaScript-Technologien wie React oder Angular bieten sich dafür an, eine Landingpage/Microsite zu entwickeln.

1.2.5 Webplattform – sich ein eigenes soziales Netzwerk bauen

Dieser Begriff kann übergeordnet für die anderen Typen von Websites verwendet werden. Ich verwende ihn hier für Websites, die von registrierten Benutzern nicht nur gelesen, sondern auch online mit einem Webbrowser mit eigenen Inhalten erweitert werden können. Die Funktionalität wird häufig von einem Content-Management-System zur Verfügung gestellt. Typische soziale Netzwerk-Plattformen wie Facebook, MySpace usw. oder sogenannte Wiki-Software (wie z. B. von *www.wikipedia.de* verwendet) gehören auch dazu.

Gerade im kommerziellen Bereich kann mit solchen Plattformen ein wesentlich besserer kundenorientierter Support erreicht und in kleineren bis größeren Unternehmen auch ein fruchtbarer Austausch von Erfahrungen und Wissen über die Abteilungsgrenzen hinaus erzielt werden.

Die Grundidee einer solchen Webplattform liegt gewöhnlich darin, dass der Inhalt durch gemeinschaftliche Arbeiten von registrierten Benutzern mit Texten, Bildern, Grafiken usw. erweitert wird, um so eine Sammlung von nützlichen Informationen anzubieten. Auch wenn der Inhalt von anderen Benutzern erstellt wird, ist ein Moderator unabkömmlich, der den Inhalt verwaltet und überprüft.

Nötige Kenntnisse für Webplattformen

Hier gilt dasselbe wie zuvor schon für den Webshop und den Blog. Auch hängen die erforderlichen Kenntnisse davon ab, ob Sie Benutzer oder Moderatorin einer solchen Webplattform sind. Mit HTML-Kenntnissen können Sie hier den Inhalt nach eigenem Belieben besser strukturieren und mit CSS gestalten. Dies hängt allerdings von der verwendeten Plattform ab. Manche Plattformen erlauben nur bedingt, HTML-Elemente zu verwenden. Sollten Sie vorhaben, eine eigene Webplattform zu entwickeln, genügen HTML-Kenntnisse allein nicht mehr. Dann sind weiter reichende Kenntnisse in der Entwicklung in einer serverseitigen Webprogrammiersprache wie PHP, Ruby, Python oder (mittlerweile auch serverseitig möglich) JavaScript-Technologien wie z. B. React oder Angular nötig.

1.2.6 Web-Apps

Hinter diesem Begriff verbergen sich im Grunde nichts anderes als gewöhnliche Webanwendungen, die sich wie Desktopanwendungen anfühlen. Dabei handelt es sich um Internetanwendungen mit vielen Interaktionsmöglichkeiten, wie Sie dies etwa von einer gewöhnlichen Desktopanwendung her kennen. Solche Anwendungen müssen nicht zwangsläufig in einem Webbrowser ausgeführt werden. Der Vorteil solcher Anwendungen gegenüber klassischen Webanwendungen kann eine verbesserte Benutzung und, mit einer entsprechenden Technik, auch eine schnellere Performance sein.

Nötige Kenntnisse für Rich Internet Applications

Zur Erstellung solcher Web-Apps wurden in der Vergangenheit bevorzugt externe Technologien in Form von Plug-ins von Drittanbietern wie Flash Player, Java Virtual Machine, Silverlight, AIR oder Flex verwendet. Mittlerweile lassen sich auch Web-Apps über die klassischen Webtechnologien wie HTML, CSS, JavaScript und Ajax ohne Plug-ins erstellen. Hierfür stehen Ihnen fertige HTML-/JavaScript-basierte Frameworks und Bibliotheken wie Angular, React, jQuery, Ext JS oder Google Web Toolkit zur Verfügung.

1.3 Dynamische und statische Websites

Im Abschnitt zuvor war des Öfteren die Rede von Content-Management-System (kurz: CMS) oder Blogsystem. Bekannte Vertreter solcher Systeme sind z. B. WordPress, Joomla!, Drupal, Contao oder TYPO3. Sind solche Systeme erst einmal auf einem Server oder Webspace installiert, sind im Prinzip kaum noch weitere Kenntnisse nötig, aber wie immer trotzdem hilfreich. CMS werden auf der Serverseite ausgeführt, sind mit modernen Webprogrammiersprachen (meistens PHP, Ruby oder Python) programmiert und benötigen häufig auch eine serverseitige Datenbank (wie MySQL oder PostgreSQL). Dies setzt voraus, dass entsprechende Ressourcen (PHP, Ruby, Python und/oder MySQL) auf dem Server vorhanden sind und verwendet werden dürfen. Solche CMS erzeugen dynamische Websites. Hierzu soll der Unterschied zwischen statischen und dynamischen Websites erklärt werden.

1.3.1 Statische Websites

Bei einer statischen Website wird der komplette Inhalt wie Texte und Bildangaben unveränderbar in einzelnen Dateien auf dem Webserver gespeichert. Der Inhalt einer solchen Datei wird mit der Auszeichnungssprache HTML erstellt. Bei Änderungen statischer Websites muss die betreffende Datei gewöhnlich auf dem lokalen Rechner manuell geändert und wieder auf den Webserver hochgeladen werden. Die Verwendung statischer Websites dürfte sich daher für kleinere Webpräsenzen lohnen, bei denen relativ selten Änderungen nötig sind.

Mögliche *Vorteile* von statischen Websites:

- Die Kosten für das Webhosting sind günstiger, da keine speziellen Features wie Datenbanken oder Skriptsprachen benötigt werden. Dazu muss gesagt werden, dass die sogenannten Profi-Features bei den größeren Webhostern nicht mehr die Welt kosten.
- Der Seitenaufbau und die Ladezeit sind eventuell schneller, weil die Seite sofort vom Webserver als Antwort auf die Anfrage zurückgegeben werden kann.
- Die Entwicklung von statischen Websites kann einfacher und kostengünstiger sein. Dies hängt allerdings vom Umfang des Projekts und den Kenntnissen ab.

Mögliche *Nachteile* von statischen Websites:

- Hierbei sind gute Kenntnisse in HTML und Co. nötig, um die Website zu aktualisieren. Wenn Sie vorhaben, für jemanden eine Webpräsenz mit statischen Webseiten zu erstellen, sollten Sie sich bewusst sein, dass Sie die Änderungen meistens selbst vornehmen müssen.
- Die Ersterstellung vieler einzelner Dateien für die statische Website kann sehr zeitintensiv werden.
- Wenn Sie das Design der Website ändern wollen, kann dies ziemlich aufwendig werden. Im ungünstigsten Fall müssen Sie jede einzelne Datei ändern. Idealerweise basiert allerdings das Webdesign einer statischen Website auf CSS, und somit müsste nur diese CSS-Datei geändert werden.

In Abbildung 1.1 sehen Sie eine vereinfachte Darstellung, wie eine statische Webseite zurückgegeben wird. Hierbei sendet der Webbrowser zunächst eine Anfrage (*Request*) für eine Webseite an den Webserver, der die Website hostet. Der Webserver findet diese Seite und sendet sie als Antwort (*Response*) an den Webbrowser zurück. Wird diese Webseite nicht auf dem Webserver gefunden, liefert dieser eine Fehlermeldung (meistens mit dem Fehlercode 404) zurück, die besagt, dass die Ressource auf dem Server nicht gefunden werden konnte.

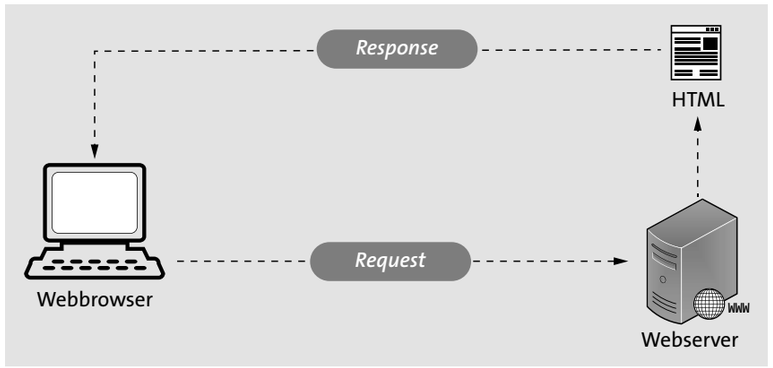

Abbildung 1.1 Anfrage des Webbrowsers und Rückgabe einer statischen Webseite, die auf einem Webserver gespeichert ist

1.3.2 Dynamische Websites

Bei dynamischen Websites generiert gewöhnlich ein CMS (Content-Management-System) die Webseiten. Dabei werden in der Regel die Inhalte wie Texte und Bilder von den technischen Elementen wie dem Layout oder den Skripten getrennt aufbewahrt. Wenn ein Besucher die Website besucht, werden die Inhalte und technischen Elemente auf dem Webserver aus einer Datenbank gelesen und dynamisch zu einer Webseite zusammengefügt, bevor der Besucher diese Webseite zurückgeliefert bekommt.

Auf jeden Fall muss ein solches CMS in einer Webserver-Umgebung installiert werden/sein, wo je nach CMS unterschiedliche Skriptsprachen wie PHP oder Python und meistens Datenbanken wie MySQL oder PostgreSQL vorhanden sein müssen, bevor Sie das CMS installieren/verwenden können.

Mögliche *Vorteile* von dynamischen Websites:

- Die Aktualisierung und das Hinzufügen neuer Inhalte lassen sich wesentlich schneller über eine webbasierte Nutzeroberfläche durchführen. Auch um die Datenspeicherung (wo und wie) müssen Sie sich dabei in der Regel nicht mehr selbst kümmern.
- Designänderungen und Designwechsel lassen sich an einer zentralen Stelle durchführen. Häufig existieren viele fertige Vorlagen (auch *Templates* genannt). Ändern Sie das Design, wirkt sich dies auf alle vorhandenen Webseiten gleichzeitig aus.
- Solche Systeme lassen sich ohne HTML- und weitere Programmierkenntnisse pflegen oder gar von mehreren Personen verwalten. Neue Funktionalitäten lassen sich jederzeit dank vieler vorhandener Module/Plug-ins (z. B. eine Suche, eine Sitemap, ein Onlineshop oder ein Forum) hinzufügen.

Mögliche *Nachteile* einer dynamischen Website:

- Beim Webhosting fallen höhere Kosten an, da spezielle Features wie Skriptsprachen und Datenbanken benötigt werden. Allerdings sind die Kosten nicht mehr – wie noch vor ein paar Jahren – so signifikant höher als für eine statische Website.
- Wenn Sie eigene oder spezielle Module oder Plug-ins erstellen müssen, werden Kenntnisse in der Programmierung mit Skriptsprachen nötig. Dadurch könnte die Entwicklung etwas länger dauern und teurer werden.

In Abbildung 1.2 sehen Sie eine (sehr) vereinfachte Darstellung, wie eine Webseite dynamisch erzeugt wird. Ein Webbrowser stellt mit einer Webadresseingabe die Anfrage (Request) für eine Webseite an einen Webserver. Der Webserver sucht und findet die Seite und übergibt sie an den Applikationsserver. Der Applikationsserver durchsucht die gefundene Seite nach Befehlen und stellt die Webseite fertig. Zusätzlich können hier Anweisungen für eine Datenbankabfrage enthalten sein. In dem Fall wird eine solche Abfrage (*Query*) an die Datenbank (genauer: den Datenbanktreiber) gestartet. Der Datenbanktreiber gibt dann (wenn gefunden) den angeforderten Datensatz (auch *Recordset* genannt) an den Applikationsserver zu-

rück, wo diese Daten in die Webseite eingefügt werden. Die so auf dem Webserver erstellte dynamische Webseite wird als Antwort (Response) an den Webbrowser geschickt.

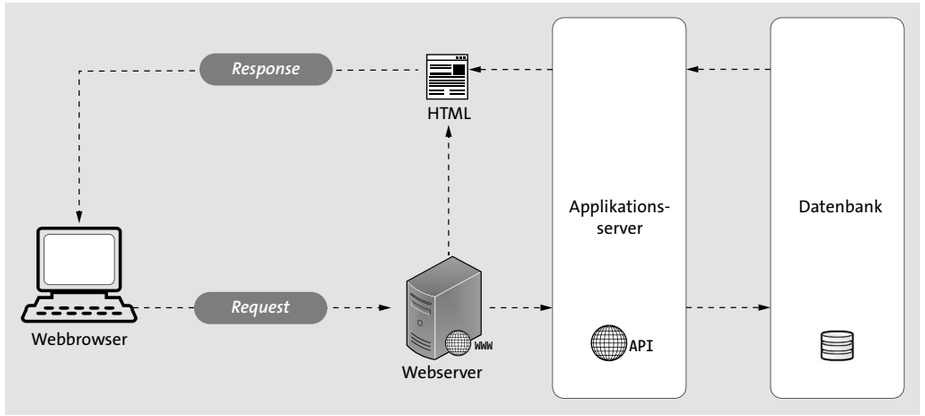

Abbildung 1.2 Vereinfachte Darstellung, wie nach einer Webbrowser-Anfrage auf dem Webserver eine Webseite zusammengebaut und zurückgesendet wird

Webserver, Applikationsserver, Datenbank

Zugegeben, hier habe ich mit vielen Begriffen um mich geworfen, und diese sollen hier kurz erklärt werden. Zwar werden Sie in diesem Buch kaum mit dynamischen Websites zu tun haben, aber trotzdem kann es hilfreich sein, diese Begriffe etwas genauer zu kennen.

Webserver: Hierbei handelt es sich um einen Computer, auf dem eine Webserver-Software (meistens nur diese) installiert ist. Ein solcher Webserver wird in der Regel dazu verwendet, Dokumente lokal, im Intranet oder über das Internet zur Verfügung zu stellen und an Clients, wie z. B. einen Webbrowser, zu übertragen. Die wohl bedeutendsten Webserver dürften der *Apache HTTP Server* und *Microsoft Internet Information Services* (kurz IIS) sein. Der Ort des Webservers kann übrigens überall in der Welt sein.

Applikationsserver: Ein solcher Server (auch *Anwendungsserver*) stellt eine Umgebung für Client-Server-Anwendungen und einen Webserver zur Verfügung. Bei einer Webanwendung ist z. B. der Webbrowser der Clientteil der Anwendung. Dieser Applikationsserver stellt bestimmte Dienste (z. B. Zugriff auf Datenbanken, Authentifizierung) zur Verfügung.

Datenbank: Eine Datenbank wird verwendet, um eine große Menge an Daten möglichst effizient zu speichern und auf Anfrage bereitzustellen. Gewöhnlich besteht eine Datenbank aus einem Datenbankmanagementsystem und den Daten selbst. Das Managementsystem einer Datenbank kümmert sich um die strukturierte Speicherung und Zugriffe auf die Daten. Für die Verwaltung und Abfrage der Daten stellen die Datenbanksysteme eine eigene Datenbanksprache bereit. Es gibt viele verschiedene Datenbanksysteme, wobei derzeit im Web MySQL und PostgreSQL die größten Marktanteile haben.

1.4 Sprachen für die Gestaltung und Entwicklung im Web

Nach den verschiedenen Typen von Websites finden Sie hier einen Überblick über die Sprachen, die Sie als Webentwickler kennen müssen und in diesem Buch kennenlernen werden. Kurz zusammengefasst sind dies:

1. *HTML*, um den Inhalt einer Website zu erstellen
2. *CSS*, um das Layout der Website zu erstellen
3. *JavaScript*, um das Verhalten der Website zu programmieren

1.4.1 HTML – die textbasierte Hypertext-Auszeichnungssprache

HTML (kurz für *Hypertext Markup Language*) ist eine rein textbasierte Auszeichnungssprache zur strukturierten Darstellung von Texten, Grafiken und Hyperlinks in HTML-Dokumenten. HTML-Dokumente können von jedem Webbrowser dargestellt werden und gelten somit auch als Basis für das Internet. Da HTML eine rein textbasierte Sprache ist (*Klartextformat*), können HTML-Dokumente mit jedem beliebigen Texteditor bearbeitet und gespeichert werden. Neben den im Webbrowser dargestellten Daten kann ein HTML-Dokument weitere Metainformationen enthalten. Gelegentlich hört man, dass Websites mit »HTML-Befehlen« programmiert werden. Es ist allerdings falsch, bei HTML von einer Programmiersprache zu sprechen. Bei einer Programmiersprache werden bestimmte Aufgaben durch eine Abfolge von Befehlen gelöst. HTML besitzt allerdings keine Befehle oder Anweisungen, sondern *Marken* (auch *Tags* genannt). Mithilfe dieser Marken werden die einzelnen Bereiche eines HTML-Dokuments strukturiert. Auch wenn weiterhin von »HTML programmieren« die Rede sein wird, sollten Sie sich bemühen, hier fachlich korrekt zu bleiben, weil HTML nicht programmiert, sondern geschrieben wird.

Metainformationen

Metainformationen (oder auch Metadaten) sind Daten, die gewöhnlich nicht angezeigt werden, sondern Informationen über die Merkmale der Daten (hier das HTML-Dokument) enthalten, wie z. B. die Sprache oder den Verfasser des Dokuments.

1.4.2 CSS – die Gestaltungssprache Cascading Style Sheets

Zwar wurden im Laufe der Jahre auch Elemente zu HTML hinzugefügt, die sich mit der optischen Gestaltung eines Dokuments befasst haben, aber glücklicherweise hat man sich dann doch für eine Trennung von Strukturierung und Layout durch die Definition mit CSS (kurz für *Cascading Style Sheets*) entschieden. So wird in der gängigen Praxis die Auszeichnungssprache HTML nur noch für die logische Strukturierung von Webseiten verwendet.

Natürlich ist es möglich, HTML ohne CSS-Angaben im Browser darzustellen. Wenn Sie z. B. einen Text für Überschriften im `h1`-Element (z. B. `<h1>Überschrift</h1>`) verwenden, wird dieser Text im HTML-Dokument zwischen `<h1>` und `</h1>` größer als der restliche Text dargestellt. Verwenden Sie HTML-Elemente für Tabellen, so werden sie sichtbar als Tabelle strukturiert. Texte können in HTML auch fett oder kursiv dargestellt werden. Wie diese HTML-Elemente (ohne CSS) im Webbrowser letztendlich angezeigt werden, wird vom Webbrowser-Anbieter festgelegt. Die HTML-Spezifikation enthält nur Empfehlungen, wie eine solche Standardeinstellung im Browser aussehen könnte. Die Auszeichnungssprache HTML dient lediglich dazu, ein HTML-Dokument mit den HTML-Elementen semantisch zu strukturieren, und sollte in der Praxis ausschließlich dafür verwendet werden. Für das Layouten und Stylen wird CSS benutzt.

Durch die Trennung von HTML und CSS werden Inhalt und Layout voneinander getrennt. Dabei liegt gewöhnlich in der Praxis der HTML-Code (zur Strukturierung) in einer und der CSS-Code (zum Formatieren und Stylen) in einer anderen Datei. Wenn Sie diese Trennung konsistent auf alle Webseiten anwenden, können Sie mit nur einer CSS-Datei das komplette Layout aller Webseiten ändern. Die HTML-Dateien müssen nicht geändert werden. Die optische Darstellung und das Styling der HTML-Elemente regeln Sie daher idealerweise mit CSS. CSS kann wie die Auszeichnungssprache HTML mit einem reinen Texteditor bearbeitet werden.

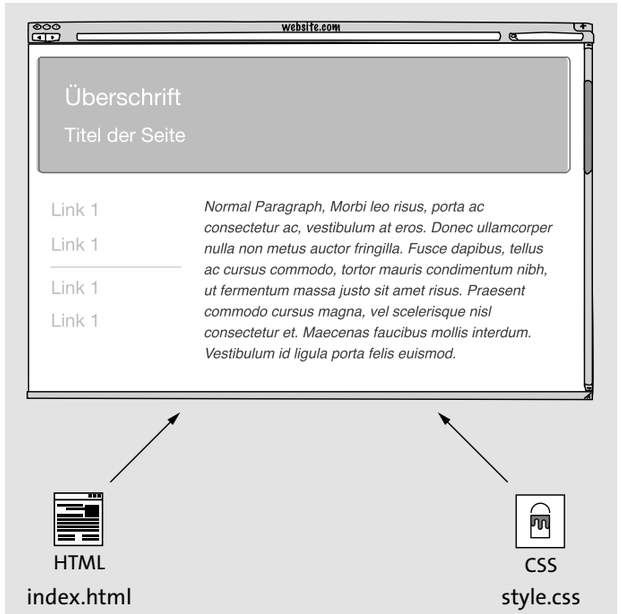

Abbildung 1.3 Mittels HTML und CSS werden Inhalt und Layout voneinander getrennt. Gewöhnlich liegt der HTML-Code zur semantischen Strukturierung in einer und der CSS-Code zum Stylen und Layouten in einer anderen Datei.

1.4.3 JavaScript – die clientseitige Skriptsprache des Webbrowsers

Bei clientseitigen Skriptsprachen spricht man von webbasierten Skripten, die auf dem lokalen Rechner, in der Regel meistens vom Webbrowser, ausgeführt werden. In der gängigen Praxis ist JavaScript die bedeutendste clientseitige Skriptsprache. Mit JavaScript können Sie die beschränkten Möglichkeiten der Auszeichnungssprache HTML um Benutzerinteraktionen erweitern, um z. B. Inhalte auszuwerten, dynamisch zu verändern oder zu erzeugen. JavaScript wurde als *ECMAScript* (genauer: ECMA 262) standardisiert, und heutzutage kommt keine moderne Website mehr ohne JavaScript aus.

Leider wurde in der Vergangenheit JavaScript für allerlei Unfug missbraucht, sodass der Ruf dieser Programmiersprache nicht unbedingt der beste war. Zusätzlich kam ein Browserstreit zwischen Netscape und dem Internet Explorer dazu, bei dem Microsoft mit JScript eine eigene JavaScript-Sprache durchboxen wollte.

Mittlerweile hat sich das Blatt zugunsten von JavaScript gewendet: Das ist zum einen dem W3-Konsortium mit der Einführung eines *Document Object Models* (kurz DOM), das von den Webbrowser-Herstellern nach und nach übernommen wurde, und zum anderen den vielen *JavaScript-Frameworks* zu verdanken. Ein großer Teil der Neuerungen des HTML-Standards hat nichts mit den gewöhnlichen HTML-Elementen zu tun, sondern vielmehr mit *JavaScript-APIs*.

Auch wenn Sie in diesem Buch JavaScript vorwiegend als Möglichkeit kennenlernen, HTML und CSS zu erweitern, so finden Sie diese Sprache mittlerweile schon außerhalb von Webbrowsern für mobile Anwendungen und Desktopanwendungen oder auf Servern oder Microcontrollern wieder.

1.4.4 Die serverseitigen Skriptsprachen und Datenbanken

Serverseitige Skriptsprachen werden zwar in diesem Buch nicht direkt behandelt, sollten aber trotzdem kurz erwähnt werden, da erst mit serverseitigen Skriptsprachen Webseiten dynamisch erzeugt werden können. Den Begriff *dynamische Website* habe ich bereits in Abschnitt 1.3.2 kurz erklärt. Bekannte und gängige serverseitige Skriptsprachen sind PHP, Python oder Ruby. Mit diesen Skriptsprachen können z. B. Blogs, Foren, Formmails, Gästebücher oder Wikis realisiert werden. Die meisten größeren Websites sind heutzutage mit serverseitigen Techniken ausgestattet und verwenden häufig zusätzlich eine Datenbankanbindung zu MySQL oder PostgreSQL.

Viele größere Blog- oder Content-Management-Systeme wie WordPress, Drupal, TYPO3, Contao oder Joomla! beruhen auf solchen serverseitigen Skriptsprachen mit Datenbankanbindung. Meistens werden diese Systeme mit PHP als Skriptsprache und MySQL als Datenbank realisiert.

Es kann recht hilfreich sein, wenn Sie Grundkenntnisse in einer Skriptsprache wie PHP haben, um z. B. Formmails, Gästebücher oder Umfragen für kleinere Webpräsenzen erstellen zu

können. Auch der Umgang mit Datenbanken wie MySQL ist recht nützlich. Wenn Sie nach diesem Buch mit HTML, CSS und JavaScript vertraut sind, steht Ihnen nichts mehr im Wege, sich mit einer Skriptsprache wie PHP und einer Datenbank wie MySQL zu befassen, um noch tiefer in die Webentwicklung einzutauchen. Gerade zur Kombination PHP und MySQL gibt es bereits sehr gute Literatur vom Rheinwerk Verlag.

1.5 Was brauche ich, um hier anzufangen?

Gerade Einsteiger fragen sich zunächst, was für die Erstellung von Webseiten und zum Erlernen von HTML benötigt wird. Im Grunde würden Sie gar nichts benötigen, weil sich von Haus aus alles an Bord Ihres Betriebssystems befindet. Genau genommen brauchen Sie zum Erstellen von Webseiten nur einen reinen *Texteditor* und zur Darstellung der Webseiten einen *Webbrowser*.

1.5.1 (HTML-)Editor zum Schreiben von HTML-Dokumenten

Als *Texteditor* könnten Sie theoretisch den auf dem System installierten Editor verwenden. Bei Microsoft Windows wäre dies der Microsoft Editor (englischer Originalname: Notepad). Auch der Editor TextEdit auf dem Mac versteht sich prächtig mit HTML. Bei den Linux-Systemen hängt der Standardeditor von der verwendeten Distribution ab. Häufig kommt hier gEdit zum Einsatz, und auch dieser eignet sich bestens für die Erstellung von HTML-Seiten.

In der Praxis greift kaum ein ambitionierter Webentwickler zu den Standardeditoren des Betriebssystems und verwendet stattdessen echte HTML-Editoren (oder zumindest universelle Texteditoren). Der Vorteil solcher speziellen HTML-Editoren ist, dass Sie mit ihnen eine Syntaxhervorhebung und viele weitere hilfreiche Funktionen für die Erstellung von Webseiten an Bord haben. Es gibt eine Menge kostenlose und kommerzielle HTML-Editoren auf dem Markt. Die Office-Programme zur Textverarbeitung wie z. B. MS Word sind weniger gut bis gar nicht für die Erstellung von reinem HTML-Code geeignet, weil sie dem HTML-Quellcode oft unnötig viel Ballast hinzufügen (wenn die Dateien im HTML-Format gespeichert werden).

Sollten Sie sich noch nicht für einen bestimmten Editor entschieden haben oder vielleicht noch nicht so recht wissen, was Sie denn verwenden wollen, dann finden Sie hier kurze Empfehlungen von mir, die alle für Windows, Linux und macOS (kostenlos) erhältlich sind:

- **Visual Studio Code** (*code.visualstudio.com*): Der Editor kommt von Microsoft und hat sich mittlerweile bei vielen Webentwicklern zum Standardwerkzeug gemausert. Er ist auch der Editor meiner Wahl und erleichtert mit unzähligen Erweiterungen und Sprachunterstützungen das Entwicklerleben enorm.
- **Adobe Brackets** (*brackets.io*): Brackets wurde von Adobe als Communityprojekt rein für die Entwicklung von Webanwendungen entworfen.

- **Atom** (*atom.io*): Ebenfalls sehr beliebt ist der Editor Atom, der neben unzähligen Plug-ins auch um Themes erweitert und den persönlichen Bedürfnissen angepasst werden kann.
- **Sublime Text 3** (*sublimetext.com*): Bevor es unzählige Editoren auf dem Markt gab, war Sublime Text häufig der Editor schlechthin für Webentwickler. Allerdings ist Sublime Text 3 nicht kostenfrei, auch wenn man diesen Editor ohne Zeitbegrenzung testen kann.

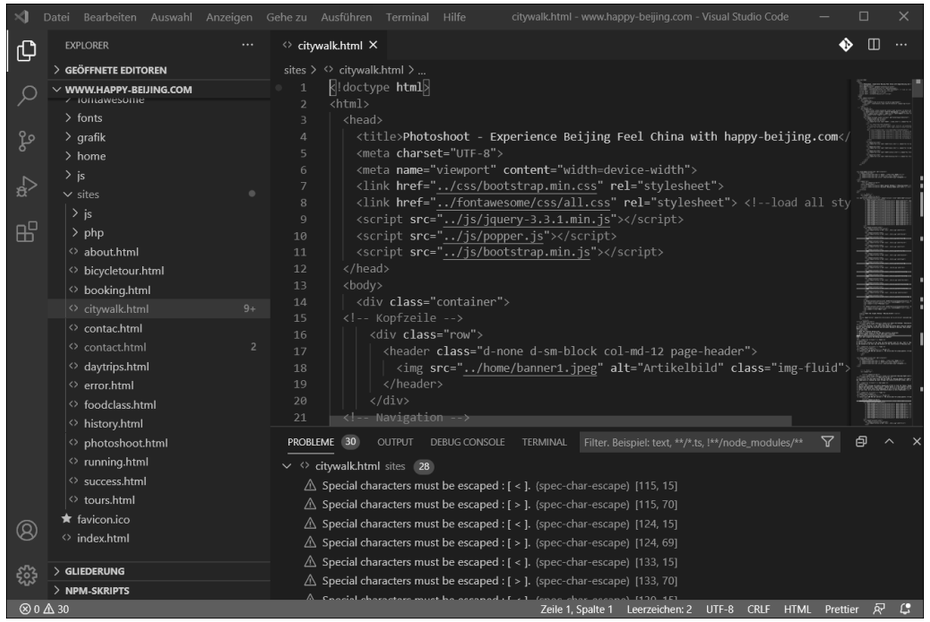

Abbildung 1.4 Visual Studio Code von Microsoft ist der Editor, den ich in der Praxis bevorzugt einsetze.

WYSIWYG-Editor

Es gibt auch WYSIWYG-Editoren; die Abkürzung steht im Englischen für *What You See Is What You Get* und bedeutet so viel wie »Was du siehst, ist das, was du bekommst«. Mit diesen Editoren können Sie eine Webseite quasi wie mit einem Office-Programm zur Textverarbeitung formatieren und »zusammenklicken«. Der WYSIWYG-Editor erzeugt einen fertigen HTML-Code in Form einer Datei. Bekanntester Vertreter ist Dreamweaver von Adobe oder der Google Web Designer (*webdesigner.withgoogle.com*). Solche Editoren sind sicherlich hilfreich, wenn es schnell gehen soll oder man über mehr Erfahrung verfügt, aber zum Erlernen von HTML sind sie meiner Meinung nach zunächst weniger gut geeignet, auch wenn diese Programme zugleich auch über einen Texteditor verfügen. Diese Umgebungen benötigen jedoch eine gewisse Einarbeitungszeit und gute Kenntnisse in der Webentwicklung, bevor man damit effektiv Webseiten entwerfen kann.

1.5.2 Webbrowser für die Anzeige der Website

Für die Darstellung des im (HTML-)Editor Ihrer Wahl erstellten HTML-Dokuments benötigen Sie einen Webbrowser. Allerdings sollten Sie sich als Entwickler von Websites nicht nur mit einem Webbrowser zufriedengeben, sondern möglichst viele zum Testen verwenden, da es viele kleine Unterschiede zwischen den unterschiedlichen Webbrowsern und ihren jeweiligen Versionen gibt. Ebenfalls empfiehlt es sich, eine Webseite auf verschiedenen Geräten zu betrachten. Wenn Sie moderne Websites auf Geräten mit verschiedenen Bildschirmgrößen wie einem Desktoprechner, einem Laptop, einem Tablet und einem Smartphone betrachten, werden Sie feststellen, dass diese häufig unterschiedlich dargestellt werden. Dies liegt daran, dass sich solche Websites (im Idealfall) an die Umgebung anpassen, in der sie angezeigt werden. Diese Anpassungsfähigkeit wird als *responsives Webdesign* bezeichnet. Die Anpassung geschieht nicht automatisch, sondern liegt in der Verantwortung des Webdesigners. Darauf gehe ich in diesem Buch gesondert ein.

Abbildung 1.5 Dieselbe Webseite wird hier auf »ami.responsivedesign.is« für verschiedene Geräte getestet.

Die wichtigsten Webbrowser sind derzeit Google Chrome, Firefox von Mozilla, Safari von Apple und Edge von Microsoft, wobei Google Chrome derzeit den größten Marktanteil besitzt. Daneben gibt es noch viele weitere Browser wie Vivaldi, Opera oder Brave, die allerdings nur einen geringen Marktanteil haben. Auch bei den mobilen Geräten liefern die Hersteller oftmals eigene Browser mit. So ist auf Samsung-Geräten besonders der Browser Samsung Internet stark vertreten.

Das Herz eines jeden Browsers ist der HTML-Renderer (häufig auch Browser-Engine), der den Quelltext, der vom Webserver kommt, in eine sichtbare Webseite umwandelt (man sagt auch *rendert*). Die zur Drucklegung aktuellen HTML-Renderer sind in Tabelle 1.1 aufgelistet.

Renderer	Browser
Quantum	Firefox
Webkit	Safari und alle (!) Webbrowser auf iOS
Blink	Chrome, Edge, Samsung Internet, Vivaldi, Opera, Brave etc.

Tabelle 1.1 Verschiedene Webbrowser und die von ihnen verwendete HTML-Rendering-Engine

Dass viele Browserhersteller den von Google zur Verfügung gestellten Blink als HTML-Renderer verwenden, macht Ihnen die Sache mit dem Testen etwas einfacher: Sie können davon ausgehen, dass eine Webseite, die in einem Browser mit Blink als Renderer gut aussieht, auch in den anderen in der Regel gut aussieht. Ähnliches gilt auch beim Webkit von Apple.

1.5.3 Schritt für Schritt: eine Webseite erstellen und im Webbrowser betrachten

Damit Sie für den weiteren Verlauf des Buches gerüstet sind, zeige ich hier in vier Schritten, wie Sie eine einzelne Webseite mit einem Editor (Ihrer Wahl) erstellen und im Webbrowser betrachten können.

Erstens: Öffnen Sie einen Text- oder HTML-Editor (siehe Abschnitt 1.5.1, »(HTML-)Editor zum Schreiben von HTML-Dokumenten«), und erstellen Sie ein neues leeres (Text-)Dokument. Meistens dürfte dies über das Menü DATEI • NEUE DATEI zu erreichen sein.

Zweitens: Tippen Sie den HTML-Code in den Editor. Zur Demonstration soll für dieses Beispiel folgendes Grundgerüst verwendet werden (machen Sie sich um die Bedeutung der einzelnen Zeilen jetzt noch keine Gedanken):

```
<!doctype html>
<html lang="de">
  <head>
    <meta charset="UTF-8">
    <title>Fenstertitel</title>
  </head>
  <body>
    <h1>Eine Überschrift</h1>
    <p>Hier steht ein gewöhnlicher Absatztext.</p>
  </body>
</html>
```

Listing 1.1 /Beispiele/Kapitel001/index.html

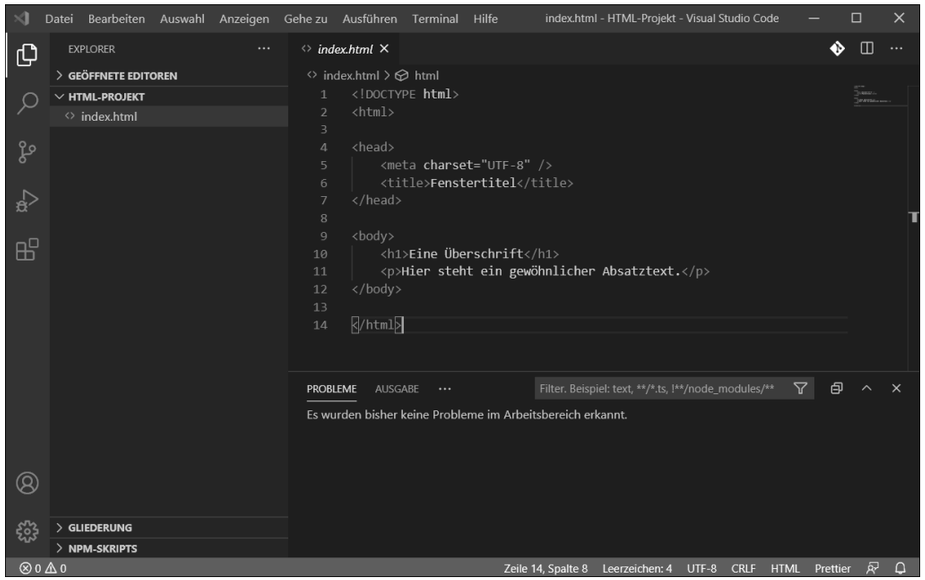

Abbildung 1.6 Hier habe ich den HTML-Code im Editor Visual Studio von Microsoft auf Windows geschrieben.

Drittens: Als Nächstes speichern Sie den HTML-Code ab. Idealerweise legen Sie hierfür ein eigenes Verzeichnis an. Die meisten Editoren bieten den Befehl DATEI • SPEICHERN oder DATEI • SPEICHERN UNTER an. Hier gilt es zwei Dinge zu beachten, und zwar die Dateiendung und die Codierung der Datei: Sie müssen diese Webseite mit der Dateiendung *.htm* oder *.html* abspeichern und darauf achten, dass die Datei wirklich als reine Textdatei (*Plain Text*) gespeichert wird, ohne dass herstellerspezifische Formatierungen enthalten sind. Als Codierung (Encoding) ist UTF-8 immer eine gute Wahl (Sie können zunächst auch ANSI verwenden). Bei den meisten Editoren müssen Sie sich darum nicht kümmern und finden häufig auch die Dateiendung zur Auswahl vor.

Dateiendung ».htm« oder ».html« verwenden

Ob Sie *seite.htm* oder *seite.html* als Dateiendung verwenden, ist dem Webbrowser im Grunde egal. Dass es überhaupt zwei Namen gibt, hat historische Gründe, die auf die DOS-Welt zurückgehen, in der die Dateinamen nur nach der 8+3-Regel benannt werden durften, also 8 Zeichen für den Dateinamen und 3 Zeichen für die Dateiendung. Wie bereits erwähnt: Was Sie hier verwenden, bleibt Ihnen überlassen. Ich empfehle Ihnen aber, sich auf eine Version festzulegen und diese künftig immer zu benutzen. Ich habe mich in diesem Buch für die Endung *.html* entschieden.

Je nach System wird das HTML-Dokument mit der Endung *.html* bzw. *.htm* im Dateibrowser mit einem entsprechenden Icon des installierten Standard-Webbrowsers angezeigt.

Viertens: Um die Datei im Webbrowser zu betrachten, brauchen Sie sie gewöhnlich nur noch doppelt anzuklicken.

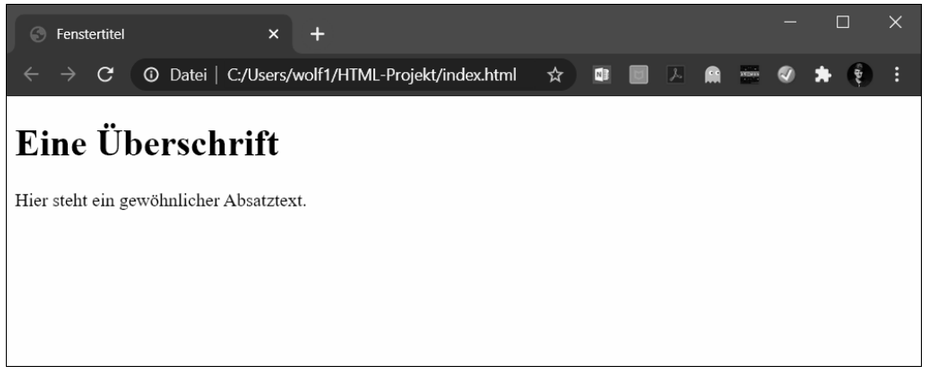

Abbildung 1.7 Das gespeicherte HTML-Dokument »index.html« in Google Chrome unter Windows

1.5.4 Geschriebenes HTML überprüfen

Um zu überprüfen, ob der HTML-Code korrekt ist, und um aus Fehlern zu lernen, lohnt es sich, den HTML-Code oder die Webseite zu validieren. Am einfachsten geht dies mit dem Onlinewerkzeug vom W3C, das Sie unter *http://validator.w3.org* finden.

> **Validierung mit Editoren und IDEs**
>
> In vielen HTML-Editoren oder Entwicklungsumgebungen sind häufig bereits Funktionen zum Validieren von HTML vorhanden oder lassen sich als Erweiterung nachträglich integrieren. So bietet zum Beispiel Visual Studio die Extension »HTMLHint« dafür an.

Auf dieser Website können Sie eine bestehende Webseite (VALIDATE BY URI) validieren, ein bestehendes HTML-Dokument hochladen (VALIDATE BY FILE UPLOAD) und prüfen lassen oder einfach nur einen HTML-Code kopieren und einfügen (VALIDATE BY DIRECT INPUT) und dann validieren.

Da Sie zu Beginn wohl meistens noch einfache HTML-Dokumente auf dem lokalen Rechner testen werden, bietet sich das Hochladen bzw. einfach das Kopieren und Einfügen an. Im Beispiel soll Letzteres kurz demonstriert werden (siehe Abbildung 1.8). Wählen Sie daher VALIDATE BY DIRECT INPUT (bzw. *http://validator.w3.org/#validate_by_input*) aus, kopieren Sie den in den Editor eingegebenen HTML-Code in die Zwischenablage, und fügen Sie ihn in das Textfeld unter ENTER THE MARKUP TO VALIDATE ein.

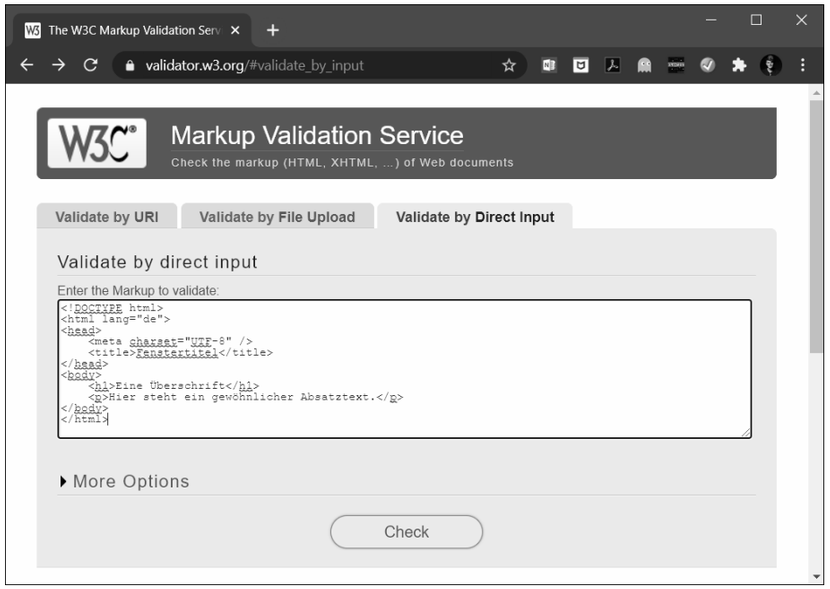

Abbildung 1.8 Hier wurde der HTML-Code zum Validieren eingefügt.

Wenn Sie anschließend auf die Schaltfläche CHECK klicken, wird die Validierung durchgeführt. War der HTML-Code fehlerfrei, bekommen Sie einen grünen Balken mit dem Hinweis, dass das HTML-Dokument in Ordnung war, was Abbildung 1.9 zeigt.

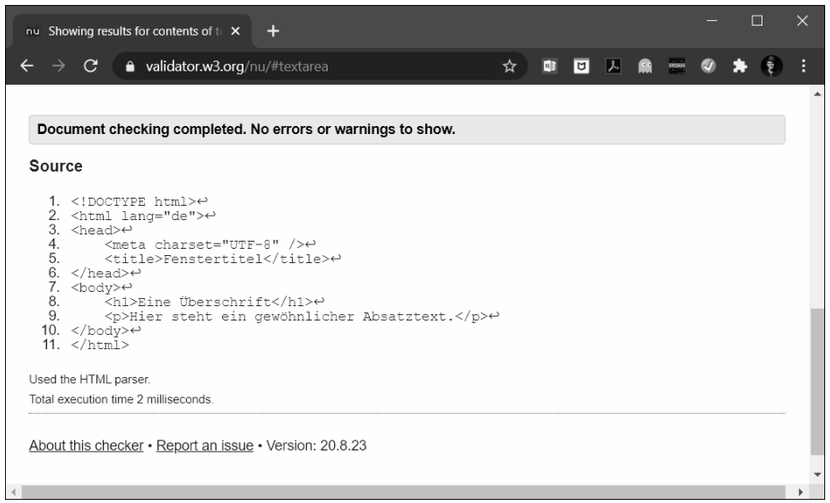

Abbildung 1.9 Der HTML-Code hat die Prüfung bestanden und ist valide.

Wenn die Prüfung ungültig war, wird der oder die Fehler mit einer Meldung aufgelistet und im HTML-Code markiert, wie Sie in Abbildung 1.10 sehen. Die Warnungen und Fehlermeldungen können Sie lesen, wenn Sie ein wenig nach unten scrollen. Sie können gerne mit Ihrem HTML-Code experimentieren und absichtlich einige (Tipp-)Fehler einbauen oder einfach eine Zeile im Code entfernen. Als Einsteiger werden Sie vermutlich an dieser Stelle noch nicht viel mit den Fehlermeldungen anfangen können.

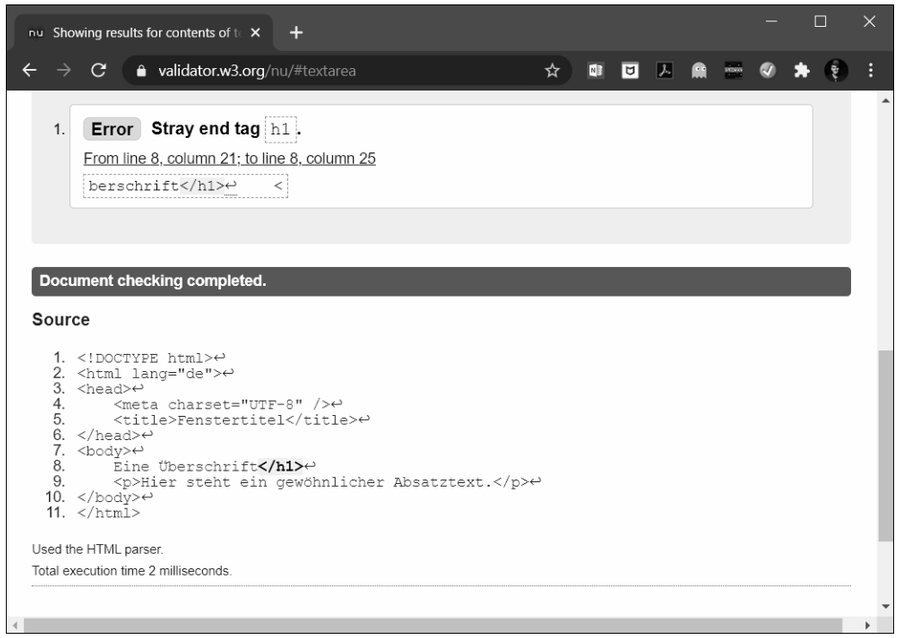

Abbildung 1.10 Diese Prüfung ergab Fehler, wie Sie an der Ausgabe der Fehlermeldung erkennen.

Eine Fehlerausgabe bedeutet übrigens nicht, dass die Webseite nicht dargestellt werden kann. Die Webbrowser sind relativ fehlertolerant und haben auch ihre eigenen Regeln. Trotzdem *kann* es im schlimmsten Fall sein, dass die Webseite in einem bestimmten Webbrowser nicht richtig oder (noch schlimmer) gar nicht angezeigt wird.

Tipp zur Validierung

Geben Sie mal unter Validate by URI einige Webadressen größerer Internetseiten ein. Es wird Sie wohl überraschen, dass es kaum Websites mit zu 100 % validem Code gibt. Bei einigen größeren Websites werden durchaus zwischen 100 und 400 Fehler angezeigt. Da werden Sie sich sicherlich fragen, wozu Sie HTML-konformen Code schreiben sollen, wenn sich nicht einmal die Ersteller großer Websites daran halten. Darauf werde ich im nächsten Abschnitt kurz eingehen.

1.5.5 Gute Gründe, den HTML-Code (trotzdem) zu validieren

Eine Webseite oder einen HTML-Code zu validieren hat viele Vorteile, die im Folgenden in Stichpunkten aufgelistet werden:

- Anzeige in jedem Browser: Das wohl gewichtigste Argument dürfte wohl die Anzeige der Webseite in jedem Webbrowser sein. Fehler im HTML-Quellcode können dazu führen, dass die Webseite in einigen Browsern nicht oder nicht richtig dargestellt werden kann. Zwar sind die Webbrowser ziemlich fehlertolerant, aber gerade bei mobilen Geräten sind weniger mächtige Fehlerkorrekturroutinen im verwendeten Webbrowser enthalten.
- Suchmaschinen: Die Suchmaschinen suchen nach Text und Schlüsselwörtern. Was nützt die schönste Website, wenn die Suchmaschine nichts mit dem Dokument anfangen kann und die Site somit nicht im Web gefunden wird.
- Barrierefreie Webseiten: Menschen mit einer körperlichen Einschränkung wie z. B. einer starken Sehschwäche sind auf spezielle Aufbereitungen des Webangebots angewiesen, die über die übliche Darstellung hinausgehen. Fehlerhafte Webseiten mit schlechter oder falscher Textauszeichnung können dazu führen, dass z. B. die Vorlesesoftware nicht richtig oder nur lückenhaft funktioniert. Ein Manko übrigens bei einigen Content-Management-Systemen, da diese oft Code generieren, der für behinderte Menschen schlechter zugänglich ist.
- Das Validieren ist gerade für Einsteiger enorm wichtig, um nicht gleich mit falschen Dingen anzufangen. Gerade, weil HTML so fehlertolerant ist, wird man schnell verleitet, unsauberen Code zu schreiben. Durch das Validieren erhalten Anfänger ein erstes Feedback.
- Auch sorgt ein ordentlicher Code für eine Qualitätssicherung, womit die Webseite auch in zukünftigen Browserversionen funktioniert, wenn diese vielleicht nicht mehr so fehlertolerant sind.
- Und zu guter Letzt zeigt man damit auch, dass man ein professioneller Entwickler ist, der Wert darauf legt, ordentliche Arbeit abzuliefern.

Es gibt sicherlich noch weitere Gründe, auf sauberen HTML-Code zu achten. Da Sie HTML (und CSS) in diesem Buch lernen, sollten Sie die Validierung von HTML immer ein wenig im Hinterkopf behalten. Auch wenn Sie als angehender Entwickler und Programmierer vorhaben, künftig eigene Webanwendungen oder gar ein eigenes CMS zu entwickeln, sollte sauberer HTML-Code immer im Fokus bleiben. Leider ist es nämlich so, dass dynamisch erstellte Webseiten oft weniger sauberen Code enthalten. Ebenso sieht es mit den WYSIWYG-Editoren aus. Auch hier ist der Code nicht immer sauber validiertes HTML, aber mit einer HTML-Validierung können Sie den Code jederzeit per Hand nachbearbeiten (wenn die Kenntnisse vorhanden sind).

Weitere Werkzeuge zum Validieren

Sie müssen nicht jedes Mal zur Website *http://validator.w3.org* gehen, um Ihren HTML-Code oder die Webseite zu validieren. Auch hier gibt es für jeden Webbrowser passende Erweiterungen, die sich nachinstallieren lassen. Für andere Browser finden Sie sogenannte *Favelets* unter *http://validator.w3.org/favelets.html*. Manche HTML-Editoren bieten ebenfalls eine grundlegende Validierung des Codes an.

1.6 Verwendete Konventionen im Buch

Für die abgedruckten Beispiele gibt es folgende Konventionen: Finden Sie dort die Auslassungspunkte (. . .), dann wurde der Code aus Platzgründen gekürzt. Das komplette und ungekürzte Beispiel hingegen finden Sie auf der Webseite zum Buch (*www.rheinwerk-verlag.de/5271*) und auf *html-beispiele.pronix.de*. Die Listingunterschrift entspricht dem genauen Pfad innerhalb der ZIP-Datei. Teile im Listing, die durch Fettdruck hervorgehoben wurden, sind besonders relevant in dem Beispiel.

```
...
<html>
  <head>
    <meta charset="UTF-8">
    <title>Fenstertitel</title>
  </head>
  <body>
  ...
  </body>
</html>
```

Listing 1.2 /Pfad_zum_Beispiel/beispielname.html

1.7 Zusammenfassung

In diesem Kapitel haben Sie verschiedene Typen von Websites kennengelernt und erfahren, was hinter Begriffen wie Webpräsenz, Blog, Webshop, Landingpage oder Webplattform steckt. Sie wissen jetzt, was dynamische und statische Websites sind. Auch haben Sie gelesen, dass HTML, CSS und JavaScript die grundlegenden Sprachen eines Webentwicklers sein sollten und dass Sie in diesem Buch alle drei erlernen werden. Zu guter Letzt haben Sie erfahren, wie Sie ein HTML-Dokument erstellen, speichern und in einem Webbrowser anzeigen – und wie Sie den HTML-Code auf Fehler überprüfen.

Kapitel 5
Tabellen und Hyperlinks

In diesem Kapitel lernen Sie weitere Elemente von HTML kennen. Genauer gesagt erfahren Sie hier, wie Sie Tabellen und Hyperlinks hinzufügen und verwenden können.

Weitere essenzielle HTML-Elemente, die ich bisher noch nicht beschrieben habe, erläutere ich in diesem Kapitel. Sie erfahren mehr zu folgenden Themen:

- **Tabellen**: Sie lernen, wie Sie Tabellen zur Darstellung von Informationen oder Daten in einem Raster verwenden können.
- **Hyperlinks**: Alle Nutzerinnen und Nutzer des Internets kennen Hyperlinks, mit denen sie sich von einer Website zur anderen bewegen können. Hier erfahren Sie, wie Sie ein HTML-Dokument mit anderen (HTML-)Dokumenten verlinken.

5.1 Daten in einer Tabelle strukturieren

Tabellen sind hilfreich, wenn Sie zusammenhängende Daten wie z. B. Börsenkurse, Finanzdaten, Reisepläne, Zugfahrpläne, Busfahrpläne, Reiseberichte oder Sportergebnisse in einem Raster aus Zeilen und Spalten darstellen wollen. HTML bietet gute Möglichkeiten an, eine solche Tabelle zu strukturieren.

HTML-Element	Bedeutung
`<table>`	Tabelle
`<tr>`	Tabellenzeile
`<td>`	Tabellenzelle
`<th>`	Tabellenkopfzelle für Überschrift
`<thead>`	Tabellenkopfbereich
`<tbody>`	Tabellenkörper
`<tfoot>`	Tabellenfußbereich

Tabelle 5.1 Schnellübersicht über die hier behandelten Tabellenelemente

HTML-Element	Bedeutung
`<colgroup>`	Gruppe von Tabellenspalten
`<col>`	Tabellenspalte
`<caption>`	Tabellenüberschrift/-legende

Tabelle 5.1 Schnellübersicht über die hier behandelten Tabellenelemente (Forts.)

Formatierung mit CSS

HTML wird nur für eine semantische und strukturelle logische Auszeichnung verwendet, und dies gilt auch für Tabellen in HTML. Tabellen in HTML bieten keinerlei Formatierungsmöglichkeiten an. Alle Attribute zur Formatierung aus altem HTML, abgesehen von einem Rahmen mit `border`, werden vom aktuellen HTML-Standard nicht mehr unterstützt. Daher gilt auch hier: Tabellen werden mit CSS formatiert.

5.1.1 Eine einfache Tabellenstruktur mit <table>, <tr>, <td> und <th>

Jede Tabelle in HTML wird zwischen den Elementen `<table>` und `</table>` erstellt (`table`, deutsch: Tabelle). Die Inhalte der Tabelle werden Zeile für Zeile hingeschrieben. Den Beginn einer Zeile notieren Sie mit einem öffnenden `<tr>` und das Ende der Zeile mit einem schließenden `</tr>` (`tr` = *table row*, deutsch: Tabellenzeile). Innerhalb einer Tabellenzeile zwischen `<tr>` und `</tr>` notieren Sie die einzelnen Zellen (oder auch Spalten) mit `<td>` und `</td>` (`td` = *table data*, deutsch: Tabellendaten).

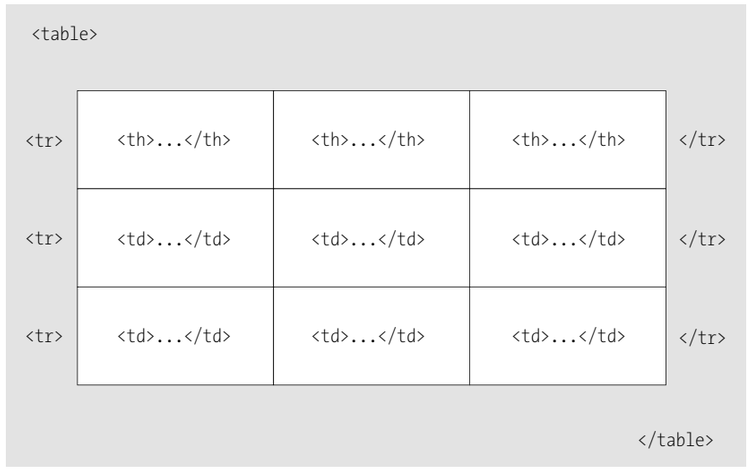

Abbildung 5.1 Eine grundlegende Tabellenstruktur in HTML

Wollen Sie Zellen oder Spalten als Überschrift einer Tabelle verwenden, können Sie die Daten zwischen `<th>` und `</th>` stellen (`th` = *table heading*, deutsch: Tabellenüberschrift). Das `th`-Element können Sie genauso verwenden wie das `td`-Element, nur dass die Webbrowser dieses Element gewöhnlich durch eine in der Spalte zentrierte Fettschrift hervorheben. Wenn es sinnvoll ist, sollten Sie Tabellenüberschriften verwenden, da dies zum einen für die Besucherinnen und Besucher mit Screenreadern hilfreich ist und zum anderen gegebenenfalls für die Suchmaschinen, die Ihre Website damit besser indizieren können.

Hierzu soll ein einfaches Beispiel einer Tabelle erstellt werden, in der Daten einer Webbrowser-Statistik von einer Website in einem Raster zusammengefasst und übersichtlich dargestellt werden:

```
...
  <table>
    <tr>
      <th>Browser</th>
      <th>Zugriffe</th>
      <th>Prozent</th>
    </tr>
    <tr>
      <td>Chrome</td>
      <td>14478</td>
      <td>59,6 %</td>
    </tr>
    <tr>
      <td>Firefox</td>
      <td>3499</td>
      <td>14,4 %</td>
    </tr>
    <tr>
      <td>Safari</td>
      <td>1619</td>
      <td>6,6 %</td>
    </tr>
  </table>
...
```

Listing 5.1 /Beispiele/Kapitel005/5_1_1/index.html

Wie Sie in Abbildung 5.2 sehen, stellen Webbrowser die Tabelle ohne jede Formatierung dar. Die Höhe und Breite einer Tabelle werden gewöhnlich gemäß dem enthaltenen Inhalt ausgegeben.

Browserstatistik November 2021

Browser	Zugriffe	Prozent
Chrome	14478	59,6 %
Firefox	3499	14,4 %
Safari	1619	6,6 %

Abbildung 5.2 Die strukturierte Darstellung einer grundlegenden Tabelle in HTML

Was darf in eine Tabellenzelle alles hinein?

In einer Zelle zwischen `<td>` und `</td>` können Sie neben einem Text auch weitere HTML-Elemente verwenden. Theoretisch könnten Sie darin eine weitere komplette Tabelle einfügen. Wenn Sie eine leere Zelle ohne Inhalt verwenden wollen, müssen Sie trotzdem ein leeres `<td></td>` bzw. `<th></th>` angeben, da die Tabelle ansonsten nicht richtig dargestellt wird. Bei alten Webbrowsern können Sie zudem zur Sicherheit ein erzwungenes Leerzeichen mit der HTML-Entität ` ` in die Zelle schreiben, weil es dort bei leeren Zellen zu Problemen kommen könnte.

5.1.2 Spalten bzw. Zeilen mit »colspan« bzw. »rowspan« zusammenfassen

Wenn Sie Tabelleneinträge über mehrere Zellen zusammenfassen (oder auch überspannen) wollen, können Sie dies mit dem HTML-Attribut `colspan` und `rowspan` machen. Anhand des Zahlenwerts, den Sie diesen Attributen übergeben, wird die Anzahl der Zellen angegeben, die Sie zusammenfassen wollen. Wie Sie anhand des Attributnamens vielleicht erahnen, wird `colspan` für das Zusammenfassen von Spalten und `rowspan` für das Zusammenfassen von Zeilen verwendet.

Hierzu ein einfaches Beispiel, in dem der Tagesplan eines Fotografieseminars in einer Tabelle zusammengefasst wurde:

```
...
  <table>
    <tr>
      <th></th>
      <th scope="col">Vormittag</th>
      <th scope="col">Mittag</th>
      <th scope="col">Nachmittag</th>
    </tr>
    <tr>
      <th scope="row">Montag</th>
      <td colspan="2">Fotoshooting (outdoor)</td>
```



```
      <td>Workshop Bildbearbeitung</td>
    </tr>
    <tr>
      <th scope="row">Dienstag</th>
      <td>Straßenfotografie (Stadt)</td>
      <td colspan="2">Fotoshooting (Porträt)</td>
    </tr>
    <tr>
      <th scope="row">Mittwoch</th>
      <td>Aktfotografie</td>
      <td>Workshop Bildbearbeitung</td>
      <td>Abschlussfeier</td>
    </tr>
  </table>
...
```

Listing 5.2 /Beispiele/Kapitel005/5_1_2/index.html

Wie Sie in Abbildung 5.3 sehen, wurde beim Rahmen der Tabelle mit CSS nachgeholfen, damit das Ergebnis von `colspan` deutlicher sichtbar ist.

Sie sehen hierbei, wie sich am Montag die Zelle `Fotoshooting (outdoor)` dank `colspan="2"` über die beiden Spalten `Vormittag` und `Mittag` erstreckt. Das Gleiche gilt für den Dienstag und die Spalte `Fotoshooting (Porträt)`, in der `Mittag` bis `Nachmittag` zusammengefasst wurde.

Bei der Verwendung von `colspan` müssen Sie beachten, dass Sie die Anzahl der Zellen reduzieren müssen, wenn Sie z. B. einen `colspan` über zwei Zellen zusammenfassen. Im Beispiel von `Montag` müssen Sie somit nur zwei `td`-Elemente notieren anstelle von drei, da sich das erste `td`-Element bereits über zwei Spalten erstreckt.

Tagesplanung

	Vormittag	Mittag	Nachmittag
Montag	Fotoshooting (Outdoor)		Workshop Bildbearbeitung
Dienstag	Straßenfotografie (Stadt)	Fotoshooting (Porträt)	
Mittwoch	Aktfotografie	Workshop Bildbearbeitung	Abschlussfeier

Abbildung 5.3 Zusammenfassen von Spalten mit dem Attribut »colspan«

Es spricht übrigens nichts dagegen, mehr als zwei Spalten zusammenzufassen. Hierbei müssen Sie auf die Anzahl der tatsächlich vorhandenen Spalten achten. Folgendermaßen könnten Sie z. B. am `Dienstag` das `Fotoshooting (Porträt)` über drei Spalten zusammenfassen:

```
...
    <tr>
      <th scope="row">Dienstag</th>
      <td colspan="3">Fotoshooting (Porträt)</td>
    </tr>
    <tr>
...
```

Die Zelle `Straßenfotografie (Stadt)` müsste dann allerdings ebenfalls entfernt werden.

Das »scope«-Attribut von <th>

Im Beispiel wurde das `scope`-Attribut beim `th`-Element verwendet. Damit können Sie angeben, ob die Tabellenüberschrift für eine Spalte (`scope="col"`) oder eine Zeile (`scope="row"`) gelten soll.

Alles, was ich eben beschrieben habe, gilt auch, wenn Sie Tabelleneinträge mit `rowspan` über mehrere Zeilen zusammenfassen wollen. Hierzu nochmals das Beispiel, in dem der Tagesplan für das Fotoseminar etwas geändert wurde, denn jetzt findet am `Dienstag` und `Mittwoch` die `Straßenfotografie (Stadt)` am Vormittag statt:

```
...
  <table>
    <tr>
      <th></th>
      <th scope="col">Vormittag</th>
      <th scope="col">Mittag</th>
      <th scope="col">Nachmittag</th>
    </tr>
...
      <th scope="row">Dienstag</th>
      <td rowspan="2">Straßenfotografie (Stadt)</td>
      <td colspan="2">Fotoshooting (Porträt)</td>
    </tr>
    <tr>
      <th scope="row">Mittwoch</th>
      <td>Workshop Bildbearbeitung</td>
      <td>Abschlussfeier</td>
    </tr>
  </table>
...
```

Listing 5.3 /Beispiele/Kapitel005/5_1_2/index2.html

Im letzten `tr`-Element müssen Sie das `td`-Element mit `Aktfotografie` entfernen, weil Sie den Eintrag `Straßenfotografie (Stadt)` darüber mit dem Attribut `rowspan` nach unten ausgedehnt haben, wodurch dieser Eintrag den Platz in der darunter liegenden Zelle einnimmt, wie Sie in Abbildung 5.4 sehen.

Tagesplanung

	Vormittag	Mittag	Nachmittag
Montag	Fotoshooting (Outdoor)		Workshop Bildbearbeitung
Dienstag	Straßenfotografie (Stadt)	Fotoshooting (Porträt)	
Mittwoch		Workshop Bildbearbeitung	Abschlussfeier

Abbildung 5.4 Zusammenfassen von Zeilen mit dem Attribut »rowspan«

5.1.3 HTML-Attribute für die Tabellenelemente

Für das `table`-Element unterstützt HTML lediglich das `border`-Attribut, um einen Rahmen anzuzeigen; der Wert darf `"1"` oder `""` sein. Hier wird CSS als bessere Alternative empfohlen. Um z. B. `border="1"` nachzubilden, fügen Sie einfach folgendes CSS-Konstrukt im Kopf des HTML-Dokuments hinzu:

```
...
<style>
  table, td, th { border: 1px solid gray }
</style>
...
```

Für die Tabellenzeile mit `<tr>` gibt es keine Attribute. Die Attribute von `<td>` und `<th>` mit `colspan`, `rowspan` und `scope` haben Sie bereits kennengelernt.

Layout mit Tabellen?

Sie sollten Tabellen nicht mehr verwenden, um das Layout einer Website zu erstellen. Dies wurde im vorherigen Jahrtausend gemacht. Ich erwähne es nur, weil Sie sich vielleicht schon den einen oder anderen Quelltext einer älteren Website angesehen haben oder sich noch ansehen werden und noch zahlreiche Websites aus dieser Zeit vorhanden sind, die eine Tabelle zum Layouten bzw. Ausrichten des Dokumentinhalts verwenden. Meistens handelt es sich um Sites, die nicht weitergepflegt werden, oder sie stammen von Entwicklern, die nicht mehr auf dem Laufenden sind. Heute greifen Sie für das Layout einer Website auf CSS zurück.

5.1.4 Tabellen mit <thead>, <tbody> und <tfoot> strukturieren

Optional zu den grundlegenden Tabellenelementen von HTML können Sie eine Tabelle noch mit den Elementen `<thead>`, `<tbody>` und `<tfoot>` in einen Kopf-, Daten- und Fußbereich einteilen.

Den Tabellenkopf schließen Sie zwischen `<thead>` und `</thead>` ein (`thead` = *table head*, deutsch: Tabellenkopf). Sinnvollerweise sollten Sie dafür das `th`-Element für die einzelnen Zellen verwenden. Die eigentlichen Daten für die Tabelle markieren Sie mit `<tbody>` und `</tbody>` (`tbody` = *table body*, deutsch: Tabellenkörper). Wollen Sie einen Bereich als Tabellenfuß notieren, fassen Sie ihn mit `<tfoot>` und `</tfoot>` (`tfoot` = *table foot*, deutsch: Tabellenfuß) ein.

Hierzu ein Beispiel, das diese drei Elemente in einer Tabelle verwendet:

```
...
  <table>
  <thead>
    <tr>
      <th>Monat</th>
      <th>Besucher</th>
      <th>Bytes</th>
    </tr>
  </thead>
  <tfoot>
    <tr>
      <th>Gesamt</th>
      <th>23423</th>
      <th>3234 MB</th>
    </tr>
  </tfoot>
  <tbody>
    <tr>
      <td>Januar</td>
      <td>3234</td>
      <td>132 MB</td>
    </tr>
...
...
    <tr>
      <td>Dezember</td>
      <td>7193</td>
```



```
      <td>894 MB</td>
    </tr>
  </tbody>
  </table>
...
```

Listing 5.4 /Beispiele/Kapitel005/5_1_4/index.html

Wenn Sie den HTML-Quelltext und die dazugehörende Darstellung in Abbildung 5.5 betrachten, werden Sie feststellen, dass der Webbrowser in der Lage ist, die Reihenfolge der Tabelle selbstständig richtig wiederzugeben. Obwohl im Quelltext der Fußbereich oben angegeben wurde, wird er vom Webbrowser passend unten angezeigt.

Besucherstatistik 2021

Monat	Besucher	Bytes
Januar	3234	132 MB
Februar	3499	235 MB
März	2092	129 MB
April	1062	102 MB
Mai	4302	324 MB
Juni	2352	192 MB
Juli	4862	424 MB
August	3957	252 MB
September	5032	624 MB
Oktober	4957	612 MB
November	6334	784 MB
Dezember	7193	894 MB
Gesamt	**23423**	**3.234 MB**

Abbildung 5.5 Eine längere Tabelle mit den Elementen <thead>, <tbody> und <tfoot> im Einsatz

Die Aufteilung einer Tabelle in drei verschiedene Bereiche ist optional und beeinflusst in der Regel nicht die Darstellung im Webbrowser. Es handelt sich um eine rein semantische Darstellung. Allerdings werden diese Elemente häufig verwendet, um das Erscheinungsbild dieser Bereiche mit CSS zu formatieren.

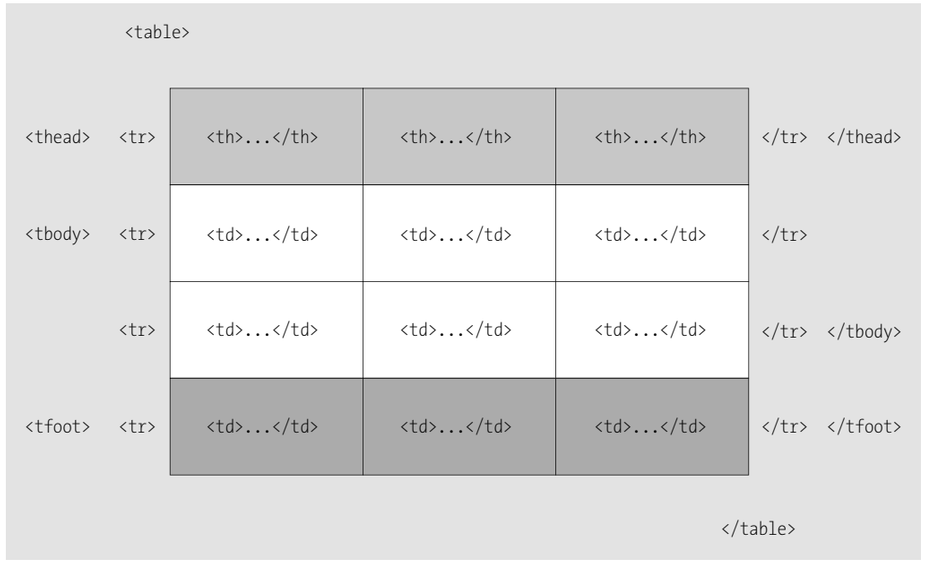

Abbildung 5.6 Die Aufteilung einer Tabelle in drei Bereiche ist zunächst rein semantischer Natur. Erst mit CSS können Sie diese Bereiche gesondert visualisieren.

Auch beim Ausdruck langer Tabellen über mehrere Seiten könnte der Webbrowser diese Aufteilung verwenden, um auf jeder Seite den Kopf- und Fußbereich der Tabelle mit auszudrucken. Damit lässt sich besser erkennen, in welcher Spalte die einzelnen Daten stehen bzw. was die Daten bedeuten. Eine weitere Möglichkeit wäre, bei langen Tabellen nur den Körperbereich zwischen `<tbody>` und `</tbody>` zu scrollen, während die Kopf- und Fußzeile stehen bleiben. Leider unterstützt noch kein Webbrowser diese Funktionen, aber das können Sie unter Umständen selbst mit CSS und gegebenenfalls JavaScript realisieren.

5.1.5 Spalten einer Tabelle gruppieren mit <colgroup> und <col>

So wie Sie eben die Tabellenzeilen mit `<thead>`, `<tbody>` und `<tfoot>` in drei Bereiche aufteilen konnten, können Sie mit den Elementen `<colgroup>` und `<col>` auch die einzelnen Spalten in semantische und logische Bereiche aufteilen, sofern dies sinnvoll erscheint. Eine Gruppierung von Spalten ist z. B. sinnvoll, um eine bestimmte Spalte oder eine bestimmte Gruppe von Spalten mit einer bestimmten CSS-Formatierung zu versehen, anstatt den Style für jede Zelle der Spalte zu wiederholen.

Die Elemente `<colgroup>` und `<col>` müssen Sie hinter dem öffnenden `table`-Element und vor allen anderen Elementen wie `tr`, `thead`, `tfoot` oder `tbody` notieren. Eine Spaltengruppe öffnen

Sie mit `<colgroup>` und schließen sie wieder mit `</colgroup>` (`colgroup` = *column group*, deutsch: Spaltengruppe). Um eine Spalte zu gruppieren, verwenden Sie das allein stehende Tag `<col>`. Wollen Sie mehrere Spalten in einem `col`-Element zusammenfassen, können Sie dies mit dem Attribut `span` und der Anzahl der Spalten als Attributwert machen.

Hierzu ein einfaches Beispiel, das das eben Beschriebene in der Praxis erläutert:

```
...
<table>
  <colgroup>
    <col span="2" style="background-color:lightgrey;">
    <col style="background-color:snow;">
  </colgroup>
    <tr>
      <th>Browser</th>
      <th>Zugriffe</th>
      <th>Prozent</th>
    </tr>
    <tr>
      <td>Chrome</td>
      <td>14478</td>
      <td>59,6 %</td>
    </tr>
    ...
    ...
  </table>
...
```

Listing 5.5 /Beispiele/Kapitel005/5_1_5/index.html

Browserstatistik November 2021

Browser	Zugriffe	Prozent
Chrome	14478	59,6 %
Firefox	3499	14,4 %
Safari	1619	6,6 %

Abbildung 5.7 Hier wurden die ersten zwei Spalten mit »span="2"« zu einer Gruppe zusammengefasst und zur Demonstration farblich mit CSS hervorgehoben.
Die letzte Spalte ist eine eigene Spaltengruppe.

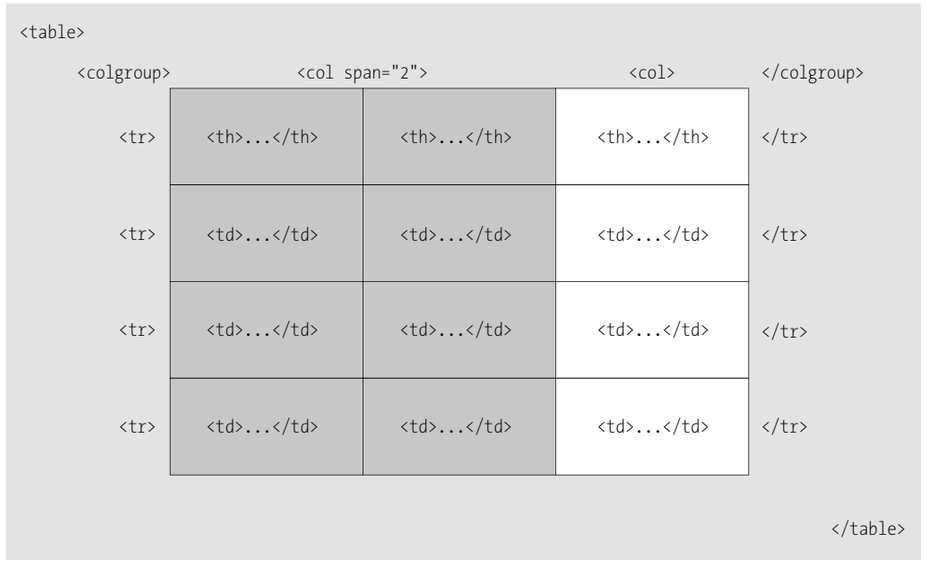

Abbildung 5.8 Die semantische Aufteilung von Spalten in Gruppen. In der Abbildung sehen Sie eine Gruppe mit zwei Spalten und eine Gruppe mit einer Spalte.

Wollen Sie hingegen für jede Spalte eine eigene Gruppe verwenden, können Sie dies wie folgt realisieren:

```
  <table>
  <colgroup>
    <col style="background-color: lightgrey;">
    <col style="background-color: snow;">
    <col style="background-color: lightgrey;">
  </colgroup>
    <tr>
      <th>Browser</th>
      <th>Zugriffe</th>
      <th>Prozent</th>
    </tr>
...
  </table>
...
```

Jetzt wurde jede Spalte in einer eigenen `col`-Gruppe zusammengefasst. Der Vorteil wird erst ersichtlich, wenn Sie Spalten mit CSS stylen wollen. Die semantische Aufteilung in drei Spalten finden Sie in Abbildung 5.9 dargestellt.

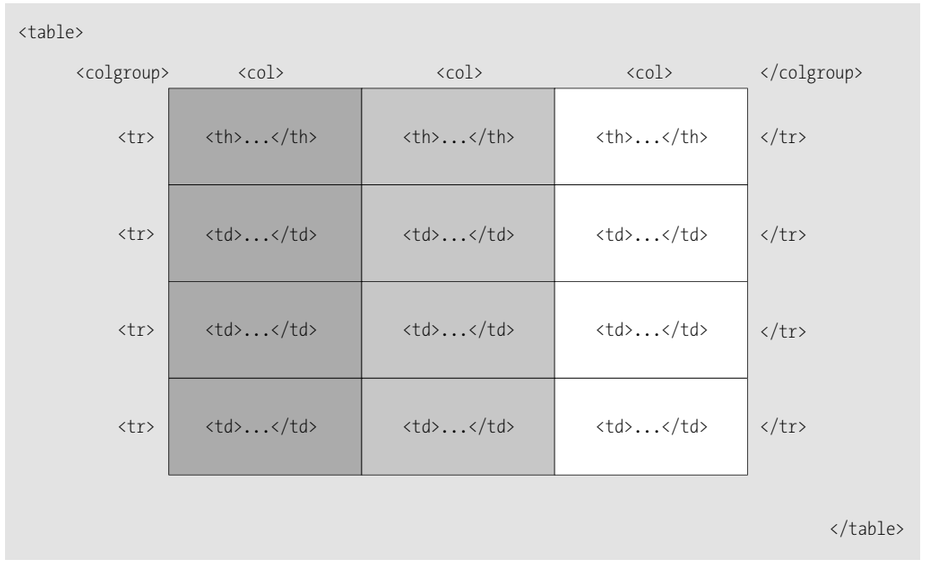

Abbildung 5.9 Semantische Aufteilung in drei Spalten

5.1.6 Tabellen beschriften mit <caption> bzw. <figcaption>

Zur Beschriftung einer Tabelle mit einem Titel können Sie entweder das `caption`-Element verwenden, das unmittelbar nach dem öffnenden `<table>`-Tag verwendet werden muss, oder Sie benutzen die neuen `figure`- und `figcaption`-Elemente.

Tabelle beschriften mit <caption>

Wie bereits erwähnt, muss das `caption`-Element gleich nach dem öffnenden `<table>`-Tag folgen. Außerdem kann nur eine Beschriftung pro Tabelle verwendet werden. Hierzu ein einfaches Beispiel:

```
...
  <table>
    <caption>Browserstatistik 11/2021</caption>
  <tr>
      <th>Browser</th>
      <th>Zugriffe</th>
      <th>Prozent</th>
    </tr>
    <tr>
      <td>Chrome</td>
      <td>14478</td>
```

```
      <td>59,6 %</td>
    </tr>
    ...
    ...
  </table>
...
```

Listing 5.6 /Beispiele/Kapitel005/5_1_6/index.html

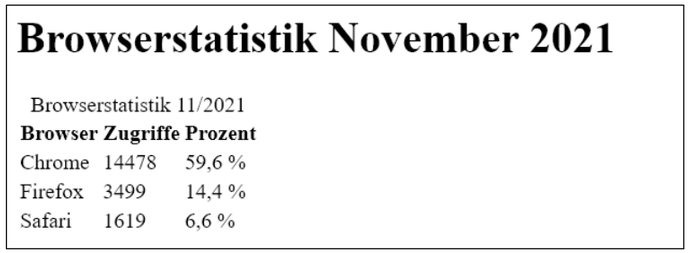

Abbildung 5.10 Die Überschrift wird standardmäßig zentriert über der Tabelle angezeigt.

<caption> mit CSS formatieren

Die Webbrowser stellen die Beschriftung gewöhnlich zentriert über der Tabelle dar. Mit CSS ist es kein Aufwand, mithilfe der CSS-Eigenschaften `text-align` und `caption-side` die Tabellenbeschriftung anders auszurichten und woanders zu platzieren.

Wollen Sie einer Tabellenbeschriftung noch weitere Hinweise hinzufügen, setzen Sie die HTML-Elemente `details` und `summary` zwischen `<caption>` und `</caption>`.

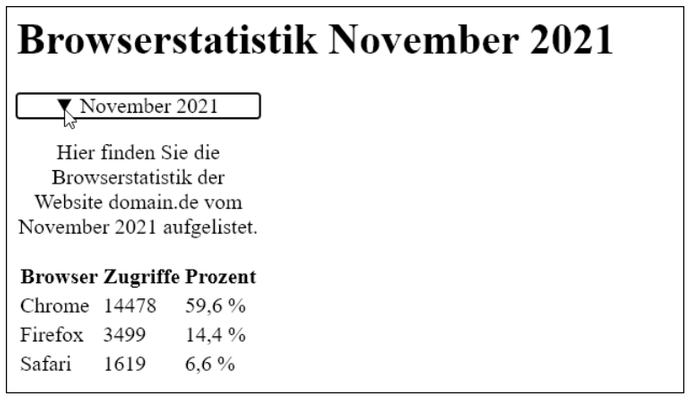

Abbildung 5.11 Informationen zum Auf- und Zuklappen dank der HTML-Elemente <details> und <summary>. Das Beispiel dazu finden Sie unter »/Beispiele/Kapitel005/5_1_6/index2.html«.

Beschriften einer Tabelle mit <figcaption>

Auf das `figcaption`- und `figure`-Element bin ich bereits in Abschnitt 4.2.9, »Gesonderte Beschriftung von Inhalten mit <figure> und <figcaption>«, eingegangen. Es bietet sich für Tabellen an, diese zwischen `<figure>` und `</figure>` zu verpacken und eine Beschriftung dieser Tabelle am Anfang nach dem öffnenden `<figure>` oder am Ende vor dem schließenden `</figure>` einzufügen. Hierzu ein Beispiel, wie Sie eine Tabelle mit den neuen `figure`- und `figcaption`-Elementen beschriften:

```
...
  <article>
  <h1>Browserstatistik November 2021</h1>
  <figure>
  <table>
  <tr>
      <th>Browser</th>
      <th>Zugriffe</th>
      <th>Prozent</th>
    </tr>
    <tr>
      <td>Chrome</td>
      <td>14478</td>
      <td>59,6 %</td>
    </tr>
...
  </table>
  <figcaption>Tabelle 1: Browserstatistik 11/2021</figcaption>
  </figure>
  </article>
...
```

Listing 5.7 /Beispiele/Kapitel005/5_1_6/index3.html

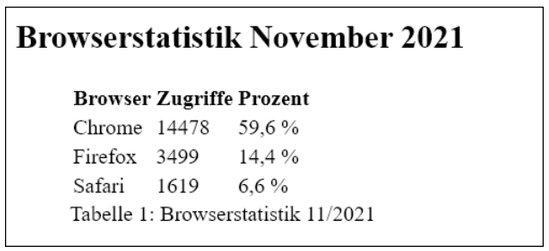

Abbildung 5.12 Tabellen beschriften mit <figure> und <figcaption>

5.2 »Elektronische« Verweise aka Hyperlinks mit <a>

Die Hyperlinks dürften wohl zu den wichtigsten Elementen von HTML gehören, weil es damit erst möglich wird, sich von einer Website zur anderen zu bewegen. Hyperlinks, oft nur Links oder Verweise genannt, werden Sie benötigen, um Ihr Projekt zu strukturieren und zu verlinken. Ausgehend von Ihrer Hauptseite benötigen Sie oft Verweise zu weiteren Unterseiten und eventuell wieder Verweise zurück zur Hauptseite. Erst durch die Verlinkung mehrerer Dateien wird eine Website zu einer sinnvoll bedienbaren Website. Neben der Verlinkung eigener Inhalte können Sie Links zu anderen Websites oder anderen Dokumenten erstellen, die sich woanders im Internet befinden.

Einen Link erstellen Sie in HTML mit dem `a`-Element (`a` = *anchor*, deutsch: Anker). Der Text, den Sie zwischen `<a>` und `</a>` schreiben, heißt Linktext oder Verweistext und wird aktiviert, indem Sie im öffnenden `<a>`-Tag das `href`-Attribut verwenden. Als Linktext können Sie notieren, was Sie wollen, aber nicht immer ist es hilfreich, einfach `Bitte hier klicken` hinzuschreiben. Mit einem sinnvollen Linktext helfen Sie Ihren Besuchern, schneller dorthin zu gelangen, wo sie hinwollen, und auch Besuchern mit Screenreadern. Zwischen `<a>` und `</a>` können auch andere Elemente als ein Text stehen. Häufig finden Sie hier z. B. eine Grafik als Link.

**Erlaubtes zwischen <a> und **

Wie bereits erwähnt, können Sie neben Text auch andere HTML-Elemente wie Grafiken zwischen `<a>` und `</a>` verwenden. Sie dürfen sogar gruppierende Elemente wie Absätze, Listen, Artikel und Blocksätze verwenden. Praktisch können Sie fast alles zwischen `<a>` und `</a>` einsetzen, abgesehen von interaktiven Elementen wie Links, Formularelementen, `audio` und `video`. Trotzdem empfehle ich Ihnen, nicht zu viel Inhalt in einen einzelnen Link zwischen `<a>` und `</a>` zu stecken. Screenreader würden den Text mehrmals vorlesen, und Besucher könnten damit überfordert sein, da sie daran gewöhnt sind, einzelne Links im traditionellen Linkstil zu aktivieren. Natürlich hängt dies vom Inhalt der Webseite ab. Ich werde hier nicht mehr näher darauf eingehen, aber Sie wissen jetzt, dass Ihnen in HTML »mehr« HTML-Elemente für Links zur Verfügung stehen. Wenn Sie extrem viel zwischen `<a>` und `</a>` gesteckt haben und sich nicht mehr sicher sind, ob es noch gültig ist, können Sie den Quelltext validieren.

Das wichtigste Attribut, mit dem das `a`-Element verwendet wird, ist das `href`-Attribut. Mit dem `href`-Attribut geben Sie den Verweis an, zu dem der die Benutzerinnen und Benutzer gelangen, wenn sie auf den Linktext klicken.

Dies ist die Seite, wohin der Hyperlink führt

```
<a href ="http://rheinwerk-verlag.de/">Verlagsseite</a>
```

Der Text, den der Benutzer anklicken kann

Abbildung 5.13 Der klassische Aufbau eines Hyperlinks

Ein Linktext wird gewöhnlich vom Webbrowser (meistens in Blau) unterstrichen. Wurde der Link bereits besucht, hat er eine andere Farbe (gewöhnlich Lila). Die Farbe von Links und besuchten Links kann abhängig vom verwendeten Webbrowser variieren. Es gibt hier also keine Standardlinkfarbe. Jeder Webbrowser hat diesbezüglich sein eigenes Standard-Stylesheet. Beides können Sie mit CSS jederzeit ändern. Wenn Sie mit dem Mauszeiger über dem Link stehen, verändert sich gewöhnlich der Cursor und wird zu einer Hand, wobei der Zeigefinger auf den Link zeigt. Die meisten Webbrowser zeigen zusätzlich links unten die Adresse an, bei der der Browser nach einem Mausklick auf den Link landen würde.

Wurde der Link angeklickt, sucht der Webbrowser die im Link angegebene Adresse (auch URL genannt), lädt diese ins Browserfenster und ersetzt die alte Webseite. Wird die Adresse des angegebenen Links nicht gefunden, wird eine Fehlermeldung ausgegeben, wie z. B. »404 – Webseite nicht gefunden«. Wenn die neue Webseite in das Browserfenster geladen wurde, können Sie mit der Zurück-Schaltfläche wieder zur vorherigen Seite wechseln.

Zum Nachlesen

Auf die Begriffe Verzeichnisname, Verzeichnisstruktur, vollständige, absolute und relative Pfadangabe bin ich bereits in Abschnitt 3.3, »Exkurs: Namenskonvention und Referenzierung«, eingegangen. Lesen Sie gegebenenfalls dort nach, wenn Sie in den folgenden Abschnitten Probleme mit den dort verwendeten Begriffen haben.

5.2.1 Links zu anderen HTML-Dokumenten der eigenen Website einfügen

Wenn Sie Ihre Website erstellen, dürften diese Verweise wohl die ersten Links sein, die Sie verwenden, um die losen Sammlungen von HTML-Dokumenten zu einer zusammenhängenden Website zu strukturieren – genauer: die Navigation der Website zu erstellen. Wenn Sie einen Link zu einer anderen Seite derselben Website angeben wollen, müssen Sie in der Regel nicht den kompletten Domain-Namen mitangeben, sondern verwenden gewöhnlich eine *relative URL*. Die in Abbildung 5.14 abgedruckte Verzeichnisstruktur soll als Beispiel dienen.

Die Verlinkung für die Startseite, hier *index.html*, zu den anderen Seiten *links.html*, *about.html* und *impressum.html* sieht demnach in der Praxis wie folgt aus:

```
...
  <nav>
    Blog |
    <a href="seiten/links.html">Links</a> |
    <a href="seiten/about.html">Über mich</a> |
    <a href="seiten/impressum.html">Impressum</a>
  </nav>
```

```
  <h1>Mein Blog</h1>
  <p>Neueste Berichte zu HTML</p>
...
```

Listing 5.8 /Beispiele/Kapitel005/5_2_1/index.html

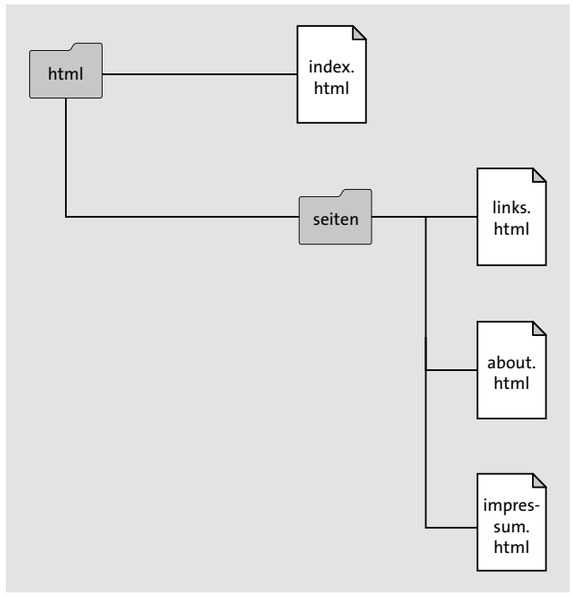

Abbildung 5.14 Verzeichnisstruktur für ein Beispiel von Links zu anderen Seiten derselben Website

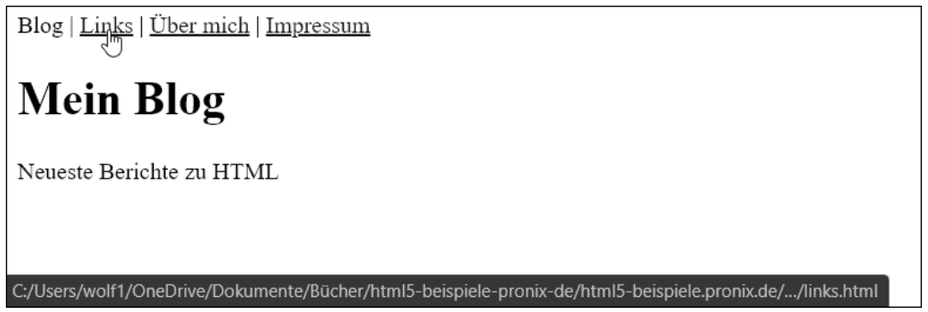

Abbildung 5.15 Dank der Verlinkung über eine relative URL kann innerhalb der Seiten derselben Website jede Seite besucht und betrachtet werden.

Natürlich müssen Sie auch die Links zu den anderen Seiten – wie hier im Beispiel *links.html*, *about.html* und *impressum.html* – anpassen. Hierbei müssen Sie bei der Angabe der relativen

URL beachten (siehe Abbildung 5.14), dass sich die Seiten (in diesem Beispiel) in einem Unterordner namens *seiten* befinden. Bezogen auf die Seite *links.html* würden die Angaben für das Attribut `href` wie folgt aussehen:

```
...
  <nav>
    <a href="../index.html">Blog</a> |
    Links |
    <a href="about.html">Über mich</a> |
    <a href="impressum.html">Impressum</a>
  </nav>
...
```

Listing 5.9 /Beispiele/Kapitel005/5_2_1/seiten/links.html

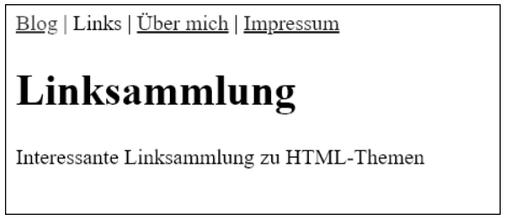

Abbildung 5.16 Das HTML-Dokument »links.html«

Hier sehen Sie, wie Sie aus dem Unterordner *seiten* mit *../* (hier *../index.html*) zum übergeordneten Ordner navigieren, in dem sich *index.html* befindet. Die anderen beiden Dateien, *about.html* und *impressum.html*, befinden sich im selben Ordner wie *links.html*, daher reicht es aus, nur den Dateinamen anzugeben. Ähnlich müssen Sie auch die Dateien *about.html* und *impressum.html* verlinken.

5.2.2 Links zu anderen Websites einfügen

Links zu anderen Websites werden genauso notiert wie die Links zu den Seiten derselben Website, nur mit dem Unterschied, dass Sie im Attribut `href` die komplette Adresse, also die *absolute URL*, zu dieser Seite angeben müssen. Hierzu ein einfaches Beispiel, in dem Links auf externe Seiten enthalten sind:

```
...
  <article>
    <header>
      <h2>Empfehlung zu HTML</h2>
    </header>
```

```
    <p>Wie bereits berichtet, hat das
       <a href="http://www.w3.org/">World Wide Web Consortium
       </a> eine
       <a href=" https://www.w3.org/TR/html53/">
       neue Empfehlung</a> für HTML vorgelegt,...
    </p>
    <aside>
    <h3>Weiterführende Links</h3>
      <nav>
      <ul>
       <li>
       <a href="https://www.w3.org/TR/html53/">
        HTML Recommendation</a></li>
       <li><a href="http://www.w3.org/">W3C</a></li>
       <li><a href="http://www.whatwg.org/">WHATWG</a></li>
      </ul>
      </nav>
    </aside>
  </article>
...
```

Listing 5.10 /Beispiele/Kapitel005/5_2_2/index.html

Abbildung 5.17 Viele Webbrowser zeigen die Zieladresse des Links unten in der Statusleiste an, wenn Sie mit der Maus darüber stehen. Aktivieren Sie den Link, ...

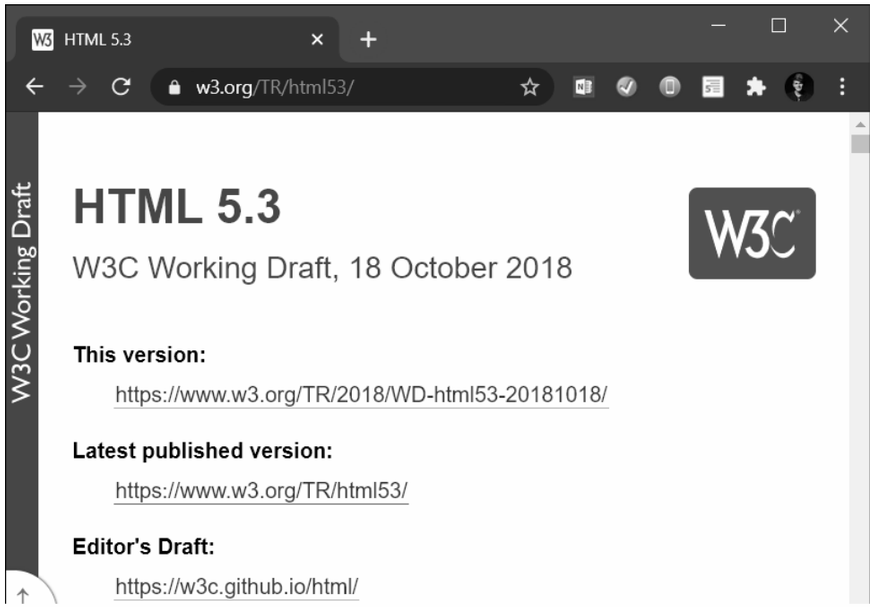

Abbildung 5.18 ... wird die Zieladresse in den Webbrowser geladen und dargestellt.

5.2.3 Links mit dem »target«-Attribut in einem neuen Fenster öffnen

Mit dem HTML-Attribut `target` des `a`-Elements können Sie dafür sorgen, dass ein Verweisziel in einem neuen Fenster oder Tab geöffnet wird. Hierbei müssen Sie `target` nur den Attributwert `_blank` übergeben. Ein Beispiel hierzu:

```
<p>Wie bereits berichtet, hat das
   <a target="_blank" href="http://www.w3.org/">W3C</a> einen
   neuen Entwurf für HTML vorgelegt,...
</p>
```

Würden Sie in diesem Beispiel den Linktext `W3C` aktivieren, würde die Zieladresse (hier: *www.w3.org*) hier tatsächlich in einem neuen Fenster oder Tab geöffnet und geladen. Ziel der Verwendung von `target="_blank"` ist natürlich vorrangig, die Besucherinnen und Besucher der Seite nicht zu »verlieren«, sondern die Seite offen zu lassen, damit sie dorthin zurückkehren, wenn sie die Seite im neu geöffneten Fenster oder Tab gelesen haben.

Neben dem am meisten verwendeten Attributwert `_blank` können Sie hier `_self` (= aktuelles Fenster), `_parent` (= Eltern-Fenster), `_top` (= oberste Fenster-Ebene) und Namen von Fenstern verwenden, die mit JavaScript verarbeitet werden können.

Das Attribut »target="_blank"« verwenden oder nicht?

Auch wenn einige Websites dieses Attribut recht gerne verwenden, sollten Sie nicht auf Teufel komm raus für jeden Link ein neues Fenster öffnen. In der Praxis sollten Sie es dem Nutzer überlassen, ob er für einen Link eine neue Seite öffnen will oder nicht. Auch wenn Sie es vielleicht gewohnt sind, unzählige Tabs und mehrere Websites auf einmal geöffnet zu haben, so sollten Sie an die etwas unerfahreneren Besucher denken, die eben nicht so im World Wide Web unterwegs sind oder eben nicht so unterwegs sein wollen. Setzen Sie das Attribut `target="_blank"` sparsam ein, und weisen Sie, wenn möglich, den Besucher darauf hin, dass ein neues Fenster oder Tab geöffnet wird, wenn er den Link aktiviert.

5.2.4 E-Mail-Links mit »href=mailto:...«

Sicherlich kennen Sie auch die Sorte von Links, bei denen sich die E-Mail-Anwendung mit einer bestimmten E-Mail-Adresse öffnet, wenn Sie sie aktivieren. Auch diese Links werden mit dem `a`-Element und dem `href`-Attribut erzeugt. Solche E-Mail-Verweise beginnen bei `href` mit `mailto:`, gefolgt von der gewünschten E-Mail-Adresse, z. B.:

```
...
  <footer>
    <a href="mailto:1@woafu.de">E-Mail senden</a>
  </footer>
...
```

Listing 5.11 /Beispiele/Kapitel005/5_2_4/index.html

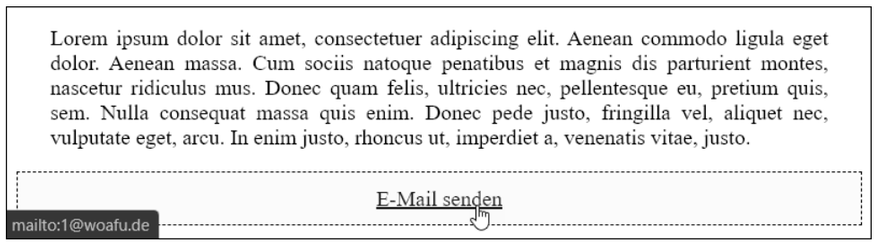

Abbildung 5.19 Wenn Sie mit der Maus über dem Link stehen bleiben, sehen Sie gewöhnlich in der Statusleiste die E-Mail-Adresse, die mit diesem Link verknüpft ist. Aktivieren Sie den Link, ...

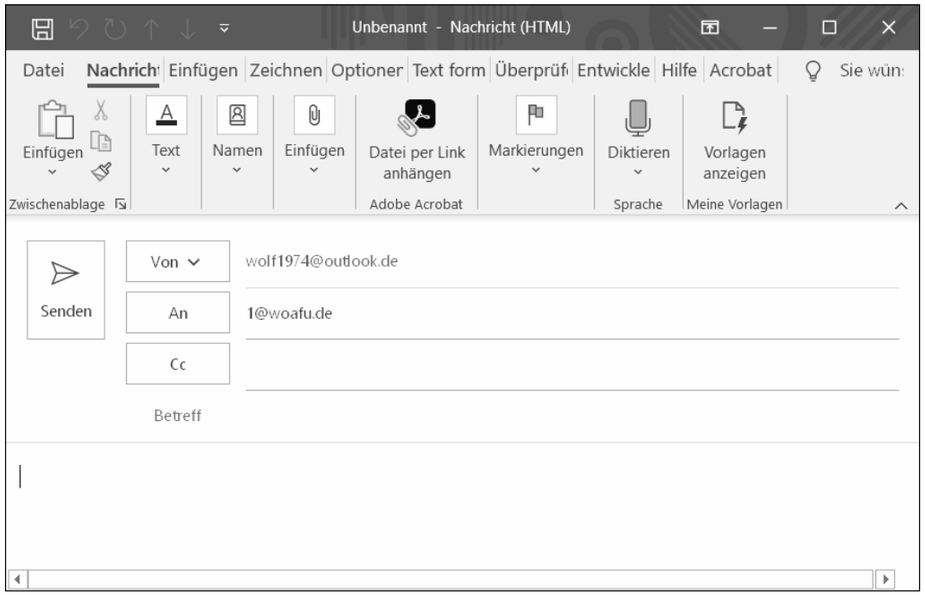

Abbildung 5.20 ... wird häufig die E-Mail-Anwendung geöffnet, automatisch eine E-Mail erstellt und darin die E-Mail-Adresse als Empfänger eingetragen.

Die Funktionalität eines »mailto«-Verweises ist nicht zuverlässig

Es gibt aber keine Garantie, dass eine solche `mailto`-Verknüpfung funktioniert. Dafür muss entweder der Webbrowser das Erstellen und Versenden von E-Mail unterstützen, oder es muss bei einem `mailto`-Verweis eine lokale E-Mail-Anwendung gestartet werden. Wenn die Besucherin oder der Besucher keine lokale E-Mail-Anwendung verwendet bzw. eingerichtet hat, sondern lediglich die klassische Webmail im Webbrowser nutzt, funktioniert der `mailto`-Verweis nur, wenn der Webbrowser entsprechend konfiguriert wurde. Außerdem gibt es Webbrowser, die man dafür gar nicht konfigurieren kann. Es ist daher sinnvoll und empfehlenswert, die E-Mail-Adresse zusätzlich in lesbarer Form anzugeben, sodass Besucher, die den E-Mail-Verweis nicht ausführen können, Ihnen trotzdem eine E-Mail senden können.

Achtung vor Spam!

Leider müssen Sie aufgrund solcher E-Mail-Adressen auf einer Website früher oder später mit unerwünschten Werbe-E-Mails (Spam) rechnen, weil es Webcrawler gibt, die Websites nach E-Mail-Adressen durchsuchen. Sie haben sogar die Pflicht, die E-Mail-Adresse im Impressum zu nennen (§ 5 Allgemeine Informationspflicht; *http://www.gesetze-im-internet.de/tmg/__5.html*). Der einzige Schutz wäre, die E-Mail-Adresse nicht im Quelltext zu nennen.

Die erste Möglichkeit, dies zu vermeiden, wäre die **Einbindung als Grafik**. Allerdings wäre dies diskriminierend gegenüber Personen, die auf Screenreader angewiesen sind, und darüber hinaus ist eine »Grafik-E-Mail-Adresse« auch rechtlich bedenklich. Häufig sind noch Verschleierungsversionen im Einsatz, in denen das @-Zeichen durch *(at)* ausgetauscht wird (z. B. *webmaster (at) dieter-baer.de*). Ebenso wird gerne der Punkt mit *(dot)* ausgeschrieben (z. B. *webmaster (at) dieter-baer (dot) de*). Natürlich bedeutet das dann, dass der Besucher die E-Mail-Adresse von Hand eingeben muss.

Auch die Verschleierung durch JavaScript ist eine Lösung. Hierbei gibt es sehr viele Ansätze. Eine interessante Website mit Informationen darüber, wie Sie es machen können, Ihre E-Mail-Adresse z. B. mit **JavaScript** zu verstecken, finden Sie z. B. hier: *http://alistapart.com/article/gracefulemailobfuscation*.

Ebenfalls bewährt hat sich das **reCAPTCHA**-Modul von Google, mit dem sichergestellt wird, dass die Anzeige der E-Mail-Adresse oder das Absenden von Formularen auch von einem Menschen erfolgt. Allerdings bedeutet dies auch einen Mehraufwand für den Besucher der Website, der eventuell Worte abtippen muss oder eine Bildaufgabe lösen muss. Mehr Informationen dazu mitsamt einem Einführungsvideo finden Sie auf *http://google.com/recaptcha/intro/index.html*.

Achtung beim Impressum

Es gibt neben den erwähnten viele weitere Möglichkeiten, sich vor unerwünschtem Spam zu schützen. Eine tolle Übersicht dazu finden auf dieser Webseite: *https://www.hosteurope.de/blog/15-moeglichkeiten-die-e-mail-adresse-geschuetzt-darzustellen/*

Trotzdem nützt dies nichts bei den Pflichtangaben im Impressum, für die Grafik, CAPTCHA oder komplexere Angaben zur E-Mail-Adresse nicht zulässig sind:

Nach § 5 Abs. 1 TMG wie nach § 55 Abs. 1 RStV müssen die Impressums-Pflichtangaben »leicht erkennbar, unmittelbar erreichbar und ständig verfügbar« sein. Das bedeutet unter anderem, dass keine höheren technischen Anforderungen für die Erreichbarkeit des Impressums gestellt werden dürfen als an den restlichen Inhalt!

5.2.5 Links zu anderen Inhaltstypen setzen

Wenn Sie Links zu anderen, nicht im Web gebräuchlichen Dokumenttypen wie z. B. Word-, Excel-, PDF-Dokumenten setzen, hängt es vom Webbrowser ab, wie er diesen Dokumenttyp weiterbehandelt. Darauf haben Sie als Webentwickler keinen Einfluss. Hier lautet zunächst die allgemeine Empfehlung, weitverbreitete Formate zu verwenden. So ist die Wahrscheinlichkeit höher, dass bei einem Link auf ein PDF-Dokument der Webbrowser einen entsprechenden PDF-Reader aufruft und das Dokument darin zum Lesen öffnet, als wenn der Link zum Inhaltstyp ein plattformabhängiges oder herstellerspezifisches Dokument ist (wie z. B. ein Word-Dokument). Hierzu ein einfaches Beispiel:


```
...
<h1>Verweis auf andere Inhaltstypen</h1>
<p>Ein PDF-Dokument öffnen: <a href="dokument.pdf">PDF</a></p>
<p>Einen MOV-Film öffnen: <a href="ganges.mov">MOV</a></p>
<p>Ein Word-Dokument öffnen: <a href="worddokument.doc"
   type="application/msword">DOC</a></p>
...
```

Listing 5.12 /Beispiele/Kapitel005/5_2_5/index.html

Abbildung 5.21 Hier haben Sie drei Links auf verschiedene Inhaltstypen.

Was bei diesen drei im Beispiel verwendeten Links passiert, kann nicht 100%ig vorhergesagt werden und hängt vom Webbrowser ab. Beim PDF-Dokument dürfte der Webbrowser wissen, wie er damit umzugehen hat. Schwieriger dürfte es mit dem Film im MOV-Format sein, weil dafür gewöhnlich ein QuickTime-Plug-in von Apple benötigt wird. Einige Webbrowser bieten an, das entsprechende Plug-in herunterzuladen und zu installieren. Andere wiederum nicht.

Dasselbe gilt für das Word-Dokument. Ist Word auf Ihrem Rechner installiert, bietet der Webbrowser häufig einen Dialog an, das Dokument mit Microsoft Word zu öffnen, oder zumindest die Möglichkeit, eine entsprechende Anwendung auszuwählen, mit der Sie dieses Dokument öffnen wollen. Meistens wird nur die Möglichkeit zum Herunterladen des Dokuments angeboten. Dies hängt wiederum davon ab, wie Sie den Webbrowser eingestellt haben.

Inhaltstyp mitangeben

Bei besonderen Inhaltstypen können Sie dem Webbrowser den Internet-MIME-Typ mit dem `type`-Attribut im öffnenden `<a>`-Tag mitteilen, wie ich dies im Beispiel mit `application/msword` für ein Word-Dokument gemacht habe. Die Informationen sind für den Webbrowser und auch andere Webclients sehr nützlich. Sinnvoll ist eine solche Angabe des Dateiformats fast immer, wenn das Linkziel kein HTML-Dokument ist. Eine Liste mit bekannten MIME-Typen finden Sie in Abschnitt A.18, »HTML-Zeichenreferenz gängiger benannter Zeichen«, den Sie auf der Verlagswebseite zum Download finden.

Informieren Sie Besucher darüber, was sich hinter einem Link verbirgt

Wenn Sie Nicht-HTML-Dokumente anbieten, sollten Sie den Besucher auf jeden Fall darüber informieren, was sich hinter dem Link versteckt und eventuell auch wie groß diese Datei ist. Sie können hierzu das globale `title`-Attribut im öffnenden `a`-Element verwenden, aber es ist empfehlenswert, genauere Angaben direkt beim Linktext zu erwähnen. Ein Negativbeispiel, wie Sie es nicht machen sollten, sieht so aus:

```
<a href="jahresumsatz.pdf">Jahresumsatz 2020</a>
```

Der Besucher wird hier nur den Linktext `Jahresumsatz 2020` zu sehen bekommen und vielleicht verdutzt reagieren, wenn dieser Link ein PDF-Dokument ist, das vielleicht etwas länger zum Laden benötigt. Besser ist daher, Folgendes zu schreiben:

```
<a title="Öffnet die PDF-Datei mit dem Jahresumsatz von 2020"
   href="jahresumsatz.pdf">
    Jahresumsatz 2020 (PDF, 3,9 MB)
</a>
```

5.2.6 Downloadlinks mit dem »download«-Attribut hinzufügen

Es gibt auch die Möglichkeit, Links als Downloadverweis hinzuzufügen – und dies unabhängig vom Inhaltstyp (= MIME-Typ) des Linkziels. Für diesen Zweck wird das Attribut `download` im öffnenden `<a>`-Tag verwendet. Hier nochmals derselbe HTML-Code vom Beispiel */Beispiele/Kapitel005/5_2_5/index.html* zuvor, nur werden jetzt alle drei Dateien mit dem `download`-Attribut zum Download angeboten:

```
...
<h1>Verweis auf andere Inhaltstypen</h1>
<p>Ein PDF-Dokument herunterladen:
   <a href="dokument.pdf" download>PDF</a></p>
<p>Einen MOV-Film herunterladen:
   <a href="ganges.mov" download="film.mov">MOV</a></p>
<p>Ein Word-Dokument herunterladen: <a href="worddokument.doc"
   download="worddokument.doc">DOC</a></p>
<p>Ein HTML-Dokument herunterladen: <a href="website.html"
   download="website.html">HTML</a></p>
...
```

Listing 5.13 /Beispiele/Kapitel005/5_2_6/index.html

Mit dem Attribut `download` weisen Sie einen Webbrowser an, diese Datei zum Download anzubieten, auch wenn er die Datei selbst anzeigen könnte oder das entsprechende Plug-in bzw. Add-on dazu kennt, das er für gewöhnlich für einen solchen Inhaltstyp verwenden würde.

Das Attribut `download` können Sie als allein stehendes Attribut verwenden, wie im ersten Beispiel mit dem PDF-Dokument zu sehen ist. Der Name der Datei, die heruntergeladen wird, entspricht der Angabe in `href` (hier: *dokument.pdf*). Enthält der Link in `href` keinen sinnvollen Namen, können Sie dem `download`-Attribut auch einen anderen Namen zuweisen, wie es im Beispiel mit dem MOV-Film der Fall ist, dessen eigentlicher Dokumentname *ganges.mov* lautet, der Downloadname der Datei aber *film.mov* heißt. Das letzte Beispiel mit dem HTML-Dokument soll nur demonstrieren, dass selbst webbrowsertypische Inhaltstypen wie hier ein HTML-Dokument mit dem Attribut `download` wirklich nur noch als Download angeboten werden. Beachten Sie allerdings, dass dieses Attribut nur funktioniert, wenn Sie das Beispiel online ausprobieren.

Den Besucher darüber informieren, was hier heruntergeladen wird

Wie schon beim Verlinken von Nicht-HTML-Dokumenten sollten Sie die Leserinnen und Leser darauf hinweisen, was sie herunterladen und womit sie das Dokument betrachten oder weiterverwenden können. Wenn Sie z. B. Excel-Tabellen mit einem Jahresumsatzbericht zum Download anbieten, sollten Sie den Leser darüber informieren, welche Software er benötigt, um diese Tabelle anzuschauen.

Dasselbe gilt für ZIP-verpackte Archive. Auch hier sollten Sie einen zusätzlichen Hinweis, wie ein solches Archiv entpackt werden kann, oder einen Link zu einer entsprechenden Software hinzufügen. Bedenken Sie, dass viele Besucher nichts mit Dateiendungen wie **.odt*, **.xls*, **.zip*, **.tar.bz* usw. anfangen können. Halten Sie das nicht für selbstverständlich, bloß weil Sie täglich mit unzähligen Datenformaten zu tun haben. Es empfiehlt sich, beim Download die Dateigröße mitanzugeben. Den Download eines umfangreichen ZIP-Archivs könnten Sie somit wie folgt notieren:

```
...
<a title="Jahresumsatz im Excel-Format als ZIP-Archiv verpackt"
   href="archiv.zip" download="jahresumsatz2020.zip">
      Jahresumsatz 2020 (ZIP-Archiv; 2,5 MB)</a>
   <small>(Um das ZIP-Archiv zu entpacken, benötigen Sie ein
           Packprogramm wie z. B. 7-Zip. Die Jahresumsätze sind
           im Excel-Format gehalten und benötigen somit eine
           Software, die Excel-Tabellen betrachten kann.)
   </small>
...
```

Hier habe ich neben dem `title`-Attribut das Dateiformat (hier: ZIP-Archiv) und die Dateigröße angegeben. Zusätzlich habe ich ein paar klein gedruckte Informationen zwischen `<small>` und `</small>` notiert.

Alte Webbrowser, die das »download«-Attribut nicht kennen

Bei älteren Webbrowsern, die das neue `download`-Attribut nicht kennen, wird vorgegangen wie bisher, als wäre das `download`-Attribut nicht vorhanden. Inhaltstypen, die der Webbrowser nicht kennt, werden wie gehabt entweder zum lokalen Speichern angeboten, oder Sie können aus einer Liste von Anwendungen eine auswählen, mit der Sie dieses Dokument öffnen wollen. Eine beliebte Methode ist, die Dateien in das ZIP-Format zu packen und anzubieten.

5.2.7 Links zu bestimmten Teilen einer Webseite setzen

Nichts kann für die Besucherinnen und Besucher lästiger sein, als eine lange wissenschaftliche Abhandlung eines speziellen Themas auf einer Webseite zu lesen und dabei lange hoch- und herunterscrollen zu müssen, um zu einem speziellen Abschnitt zu gelangen. Für solche Fälle können Sie sogenannte *Anker* mit dem globalen Attribut `id` setzen, die Sie mit einem gewöhnlichen Link mit dem `a`-Element anspringen können. Vorbildlich werden solche Zielanker z. B. bei Wikipedia für das Inhaltsverzeichnis eines Themas verwendet. Für eine Verlinkung zu einem bestimmten Bereich einer Webseite benötigen Sie nur:

- einen Anker (Sprungmarke), den Sie mit dem Attribut `id="ankername"` erstellen. Zum Beispiel:

  ```
  <h1 id="ankername">Überschrift xyz</h1>
  ```

- einen Link, der auf den Anker mit `href="#ankername"` verweist. Hierzu wird das Doppelkreuz-Zeichen `#` vor den Ankernamen geschrieben. Zum Beispiel:

  ```
  <a href="#ankername">Zur Überschrift xyz springen</a>
  ```

Hierzu ein einfaches Beispiel, wie Sie solche Sprungmarken in der Praxis setzen und verwenden können:

```
...
  <h1 id="top">Inhaltsverzeichnis</h1>
  <ul>
    <li><a href="#intro">Einführung in HTML</a></li>
    <li><a href="#syntax">Die Syntax von HTML</a></li>
    <li><a href="#versionen">Versionen von HTML</a></li>
    <li><a href="#techniken">Techniken rund um HTML</a></li>
    <li><a href="#praxis">Einstieg in die Praxis</a></li>
  </ul>
  <h1 id="intro">Einführung in HTML</h1>
  <p>Lorem ipsum dolor sit amet ... <p>
  <p><a href="#top">Zum Inhaltsverzeichnis</a></p>
  <h2 id="syntax">Die Syntax von HTML</h2>
```



```
  <p>Lorem ipsum dolor sit amet ... <p>
  <p><a href="#top">Zum Inhaltsverzeichnis</a></p>
  <h2 id="versionen">Versionen von HTML</h2>
  <p>Lorem ipsum dolor sit amet ... <p>
  <p><a href="#top">Zum Inhaltsverzeichnis</a></p>
  <h2 id="techniken">Techniken rund um HTML</h2>
  <p>Lorem ipsum dolor sit amet ... <p>
  <p><a href="#top">Zum Inhaltsverzeichnis</a></p>
  <h2 id="praxis">Einstieg in die Praxis</h2>
  <p>Lorem ipsum dolor sit amet ... <p>
  <p><a href="#top">Zum Inhaltsverzeichnis</a></p>
...
```

Listing 5.14 /Beispiele/Kapitel005/5_2_7/index.html

In Abbildung 5.22 sehen Sie das Beispiel im Webbrowser. Dank Sprungmarken gelangen Sie hier schneller zum gewünschten Abschnitt.

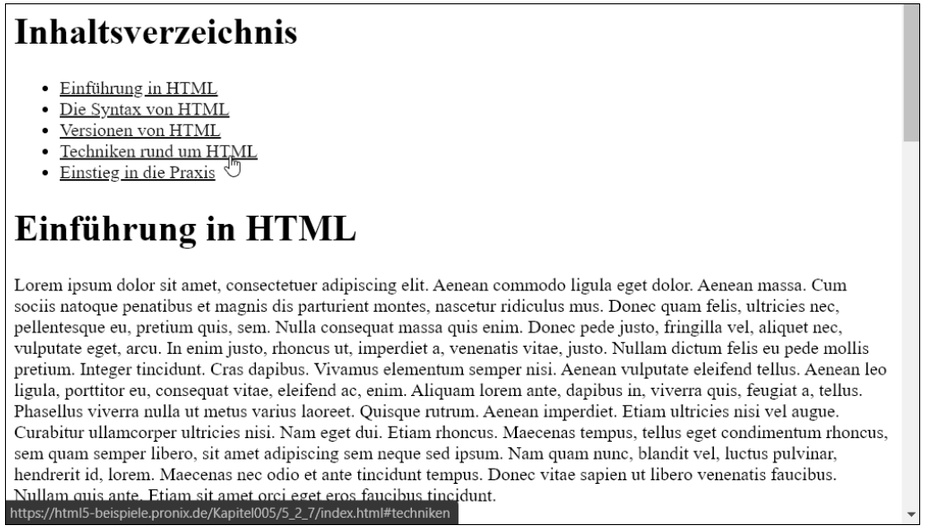

Abbildung 5.22 Ein etwas längeres Dokument, in dem Sie dank Sprungmarken ...

Aktivieren Sie z. B. den Link Techniken rund um HTML, wird direkt zum entsprechenden Abschnitt mit der Sprungmarke gesprungen, wie Sie in Abbildung 5.23 sehen. Unterhalb von jedem Abschnitt wurde außerdem ein weiterer Link zur Sprungmarke zurück zum Inhaltsverzeichnis eingefügt.

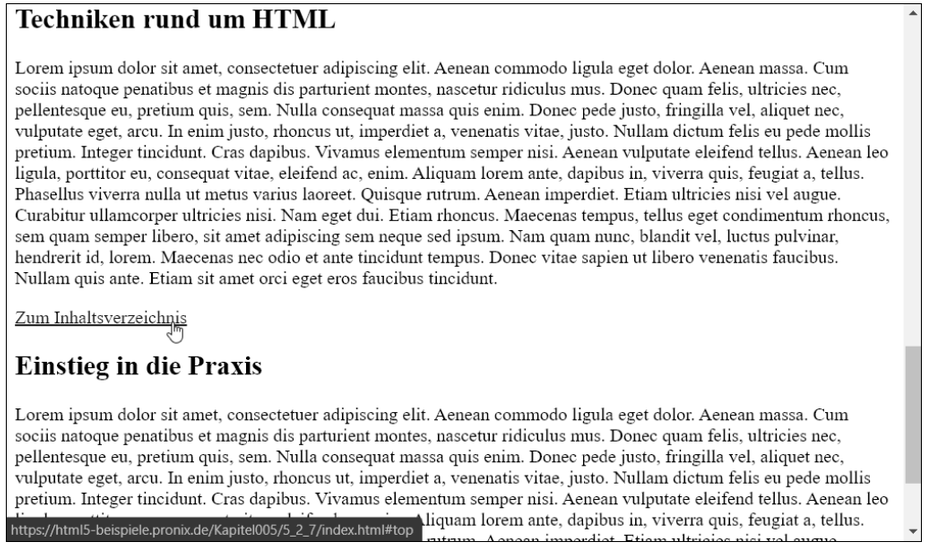

Abbildung 5.23 … schneller zum gewünschten Abschnitt gelangen.

Anker setzen mit dem »id«-Attribut (»id="ankername"«)

Bevor Sie einen Link zu einem bestimmten Teil innerhalb einer Webseite erstellen können, müssen Sie die Sprungmarke (oder auch einen Anker) mit dem globalen Attribut `id` innerhalb eines öffnenden HTML-Tags festlegen. Im Beispiel wurde dies bei den Hauptüberschriften `<h1>` und `<h2>` gemacht (z. B. `<h2 id="techniken">`). Der Attributwert von `id` muss mit einem Buchstaben oder einem Unterstrich anfangen (auf keinen Fall mit einer Zahl) und darf keine Leerzeichen enthalten. Es ist außerdem ratsam, einen aussagekräftigen Namen zu verwenden, um nicht den Überblick zu verlieren. Auf nichtssagende Bezeichnungen wie `anker1`, `anker2` usw. sollten Sie verzichten. Außerdem wird zwischen Groß- und Kleinschreibung unterschieden.

Auf einen Anker verweisen mit »#ankername«

Um einen Link zu den Ankern zu verwenden, geben Sie im öffnenden `<a>`-Tag den Attributwert zum Anker in `href` an. Lautet der Anker z. B. `<h2 id="techniken">`, schreiben Sie vor diesen Ankernamen (hier mit: `"techniken"`) noch das Doppelkreuz-Zeichen `#`. Bezogen auf unser Beispiel müssten Sie dies wie folgt notieren:

```
<li><a href="#techniken">Techniken rund um HTML</a></li>
```

Wenn Sie diesen Link aktivieren, wird im HTML-Dokument zu dem Element gesprungen, bei dem der Wert des `id`-Attributs `"techniken"` lautet. In diesem Beispiel wäre dies das `h2`-Element mit der Überschrift *Techniken rund um HTML*.

Links zu einem bestimmten Bereich einer anderen Website erstellen

Genauso ist es möglich, einen Link zu einem Bereich eines anderen HTML-Dokuments zu erstellen. Voraussetzung hierfür ist, dass das andere HTML-Dokument einen entsprechenden Anker enthält. Wenn sich der Anker in einem anderen Dokument befindet, können Sie wie folgt einen Verweis dorthin erstellen:

```
<a href="tech.html#techniken">Techniken rund um HTML</a>
```

Hiermit würden Sie in einem anderen HTML-Dokument, das sich im selben Verzeichnis befindet und dessen Dateiname *tech.html* lautet, zum Bereich mit dem Anker `#techniken` springen.

Befindet sich die Datei mit dem Anker gar auf einer anderen Website, müssen Sie die komplette URL dorthin angeben:

```
<a href="http://www.domain.de/pfad/tech.html#techniken">...</a>
```

Selbstverständlich ist es möglich, einen Link auf Bereiche von fremden Websites zu verwenden. Allerdings sollte klar sein, dass Sie hier keinen Anker setzen, sondern nur bereits vorhandene Anker verlinken können. Hier z. B. ein Link zu einem verankerten Bereich einer Wikipedia-Seite:

```
<a href="http://de.wikipedia.org/wiki/Html#Versionen">...</a>
```

Hier würden Sie direkt zur Wikipedia-Seite mit dem Eintrag *HTML* zum Anker `#Versionen` springen. Dies setzt voraus, dass der Anker existiert – was zur Drucklegung des Buches zwar noch der Fall war, sich aber jederzeit ändern könnte. Wenn der Anker nicht mehr existiert oder falsch ist, wird die Website aufgerufen und der Anker ignoriert, als wäre `#ankername` bei der Verlinkung mit dem `a`-Element nicht angegeben.

5.2.8 Links auf Telefonnummern erstellen

Da immer mehr Nutzer meistens mit einem mobilen Gerät ins Internet gehen, können Sie auch einen Link zu einer Telefonnummer erstellen. Wenn die Benutzerin oder der Benutzer diesen antippt, kann diese Nummer direkt von der Webseite angerufen werden. Der jeweilige Webbrowser muss allerdings diese Funktion auch unterstützen.

```
…
<p>
  Service-Nummer Rheinwerk-Verlag:
  <a href="tel:+49.228.421.500">+49 228 421 50-0</a>
</p>
…
```

Listing 5.15 /Beispiele/Kapitel005/5_2_8/index.html

Der Telefonverweis beginnt mit `tel:` gefolgt von der Nummer. Hierbei wird empfohlen, das Pluszeichen und die Landesvorwahl zu verwenden und die führende 0 bei der Ortsvorwahl wegzulassen. Leerzeichen können Sie mit einem Punkt notieren. Da es vom Webbrowser abhängt, ob die Nummer gleich gewählt wird, sollten Sie zusätzlich auch die Telefonnummer mit aufführen.

Sie können auch andere Dienste wie Skype als Link setzen, womit Sie mit einem Klick eine Skype-Sitzung starten können. Voraussetzung dafür ist natürlich, dass der Anwender oder die Anwenderin auch Skype verwendet.

```
…
<p>
  Ein Skype-Gespräch starten:
  <a href="skype:pronix74">Skype: pronix74</a>
</p>
…
```

Listing 5.16 /Beispiele/Kapitel005/5_2_8/index.html

Auch hierbei ist nicht garantiert, dass der Webbrowser eine Skype-Sitzung startet, weshalb Sie ebenfalls zusätzlich noch die entsprechenden Daten mit aufführen sollten. Dasselbe funktioniert auch mit Facetime.

Automatisches »tel:«

Es gibt mobile Webbrowser, z. B. Safari oder BlackBerry, die automatisch eine Telefonnummer erkennen und daraus einen *tel:-Link* erzeugen. Das ist praktisch, aber vielleicht nicht immer erwünscht. Besonders dann nicht, wenn es sich vielleicht gar nicht um eine Telefonnummer handelt. Hierfür gibt es `meta`-Tags, mit denen Sie dem Webbrowser mitteilen, dass er diese Funktion auf der Webseite nicht anwenden soll.

Für Safari:

```
<meta name="format-detection" content="telephone=no">
```

Für BlackBerry:

```
<meta http-equiv="x-rim-auto-match" content="none">
```

5.2.9 Die HTML-Attribute für das HTML-Element <a>

Zum Schluss sollen hier noch die HTML-Attribute für die Links erläutert werden, die u. a. für die Suchmaschinen recht nützlich sein können. In Tabelle 5.2 finden Sie eine Übersicht über alle vorhandenen Attribute für das a-Element. Einige davon haben Sie bereits kennengelernt.

Attribut	Beschreibung
`download`	Damit geben Sie an, dass Sie das Verweisziel zum Download anbieten, auch wenn der Webbrowser den Inhaltstyp des Ziels selbst darstellen könnte.
`href`	Damit geben Sie die URL der Seite an, zu der der Link führt, wenn er aktiviert wird.
`hreflang`	Hier können Sie die Sprache des verlinkten Dokuments angeben. Als Angaben sind die üblichen Sprachenkürzel erlaubt (z. B. `de` für Deutschland).
`media`	Damit können Sie Angaben zu den Medien machen, für die das Verweisziel optimiert wurde. Sie können entweder Medientypen, durch Kommata getrennt, aufzählen oder `all` für alle Medientypen angeben.
`rel`	Das Attribut kennen Sie bereits vom `link`-Element aus Abschnitt 3.5.1, »Die HTML-Attribute für das allein stehende HTML-Element <link>«, wohin Sie zurückblättern können, wenn Sie mehr Informationen benötigen. Hiermit bestimmen Sie den Typ der Verlinkung. Speziell für das `a`-Element sind hier noch die `rel`-Attributwerte `bookmark`, `external`, `nofollow` und `noreferrer` von besonderer Bedeutung, da diese nur im `a`-Element verwendet werden können. ▸ `rel="bookmark"`: Hier legen Sie fest, dass das Verweisziel ein übergeordneter Abschnitt (bzw. Dokument) des aktuellen Dokuments ist. Dies stellt praktisch eine Verlinkung zurück zu einem umfangreichen HTML-Dokument dar, wie es bei wissenschaftlichen oder technischen Dokumenten der Fall ist. In der Praxis wird dieser Linktyp auch für Permalinks verwendet, damit Besucher eine ältere Version des aktuellen Dokuments ansehen können. ▸ `rel="external"`: Damit geben Sie an, dass der Link zu einem externen Webangebot gehört. Häufig wird dieser mit CSS noch gesondert formatiert. ▸ `rel="nofollow"`: Damit weisen Sie die Webcrawler an, dem Link nicht zu folgen. ▸ `rel="noreferrer"`: Hiermit weisen Sie den Webbrowser des Besuchers an, beim Anklicken des Links keine Referrer-Adresse zu verwenden, wodurch vermieden werden sollte, dass der Webserver der Zieladresse Informationen erhält, von woher der Besucher gekommen ist. ▸ Nicht verwenden hingegen können Sie die Attributwerte `icon`, `pingback`, `prefetch` und `stylesheet` für `a`-Elemente.

Tabelle 5.2 Attribute für Links mit <a>-Element

Attribut	Beschreibung
target	Hier tragen Sie ein, wo das Verweisziel geöffnet werden soll. Mögliche Werte dafür sind: ▶ _blank: neues Fenster/Tab ▶ _parent: Eltern-Fenster ▶ _self: aktuelles Fenster ▶ _top: oberste Fenster-Ebene ▶ framename: Name des Fensters, das mit JavaScript geöffnet und auch darin vergeben wurde
type	Damit können Sie dem Webbrowser den MIME-Typ (Dateiformat) nennen, zu dem die verlinkte Datei gehört. Eine Liste bekannter MIME-Typen finden Sie in Abschnitt A.18, »HTML-Zeichenreferenz gängiger benannter Zeichen«. Diese Angabe ist sinnvoll, wenn das Ziel kein HTML-Dokument ist.

Tabelle 5.2 Attribute für Links mit <a>-Element (Forts.)

5.3 Zusammenfassung

In diesem Kapitel haben Sie einige essenzielle HTML-Elemente kennengelernt. Die wichtigsten Elemente, die Sie vermutlich auf fast jeder Website vorfinden und verwenden werden, sind:

- a-Element, mit dem Sie Hyperlinks erzeugen
- Tabellen, mit denen Sie zusammenhängende Daten und Informationen in einem Raster aus Zeilen und Spalten präsentieren