Internet

Victor Hahn
Kontakt: info@victor-hahn.de

Version 0.5 – letztes Update: 07.11.2013

Inhaltsverzeichnis

1 Einleitung

Herzlich willkommen in der Welt der Informatik!

Dieses Script soll dir die Möglichkeit geben, alles, was wir im Unterricht besprochen haben, noch einmal nachzulesen. Ich bemühe mich, alles wichtige auch ins Script zu schreiben und möglichst die gleiche Reihenfolge einzuhalten.

Es wird aber sicher vorkommen, dass wir im Unterricht mal etwas einschieben oder auslassen. Auch kann es sein, dass das ein oder Andere hier im Script vielleicht etwas „theoretischer“ dargestellt ist, um fachlich korrekt zu sein. Lass dich davon bitte nicht verwirren.

Dieses Script ist noch nicht fertig! Es wird im Laufe des Kurses wachsen und neue Kapitel werden hinzukommen. Das bedeutet auch, dass es wahrscheinlich nicht frei ist von Fehlern und anderen hässlichen Dingen. Wenn du einen Fehler findest oder einfach etwas, was man besser hätte erklären können, sag mir bitte Bescheid. Danke!

An einigen Stellen findest du in diesem Script Fußnoten. Manchmal erkläre ich Dinge etwas „zu einfach“, damit wir uns nicht in Details verfangen. Immer, wenn ich das tue und dabei das Gefühl habe, etwas eigentlich Wesentliches weggelassen zu haben, weise ich darauf in einer Fußnote drin. Du kannst Fußnoten also immer einfach überspringen, ohne dadurch nicht mehr mitzukommen – für diesen Kurs sind sie nicht wichtig.

Was haben wir vor?

Das Internet – alle kennen es, alle benutzen es. Wir surfen auf Webseiten, schreiben Mails und chatten online. In diesem Kurs beschäftigen wir uns damit, wie das alles eigentlich funktioniert. Wie erstellt man eine Webseite? Wie findet eine eMail den richtigen Weg von A nach B? Welche Sprache sprechen Computer untereinander, wenn sie Daten austauschen?

Im ersten Teil des Kurses legen wir einen besonderen Schwerpunkt auf das Erstellen von Webseiten. Wir werden die Auszeichnungssprache HTML und die Designsprache CSS lernen, damit unsere eigene, kleine Webseite erstellen und ins Internet hochladen.

Teil A.

Erstellen von Webseiten

2 Erstellen von Webseiten in HTML

Darf ich vorstellen? Raffaele Esposito, seines Zeichens Pizzabäcker. Ein echtes Urgestein. Gerüchten zufolge hat er schon vor über hundert Jahren Pizzen in den Ofen geschoben und in Neapel die Margherita erfunden.

Aber wir schweifen ab. Raffaele hat sich im schönen Örtchen Pizzingen irgendwo in Schwaben niedergelassen und betreibt doch sehr erfolgreich die Pizzeria Fessa. Sein einziges Problem: Die Pizzeria hat keine Website. Total altmodisch.

Das wollen wir ändern! In diesem Teil werden wir eine kleine Webseite für die Pizzeria basteln. Nichts großartiges. Aber Schritt für Schritt werden wir von einfachem Text zu etwas kommen, was zumindest ein klein Bisschen bunt ist. Wie das aussehen soll, wenn wir fertig sind, siehst du in Abbildung 1.

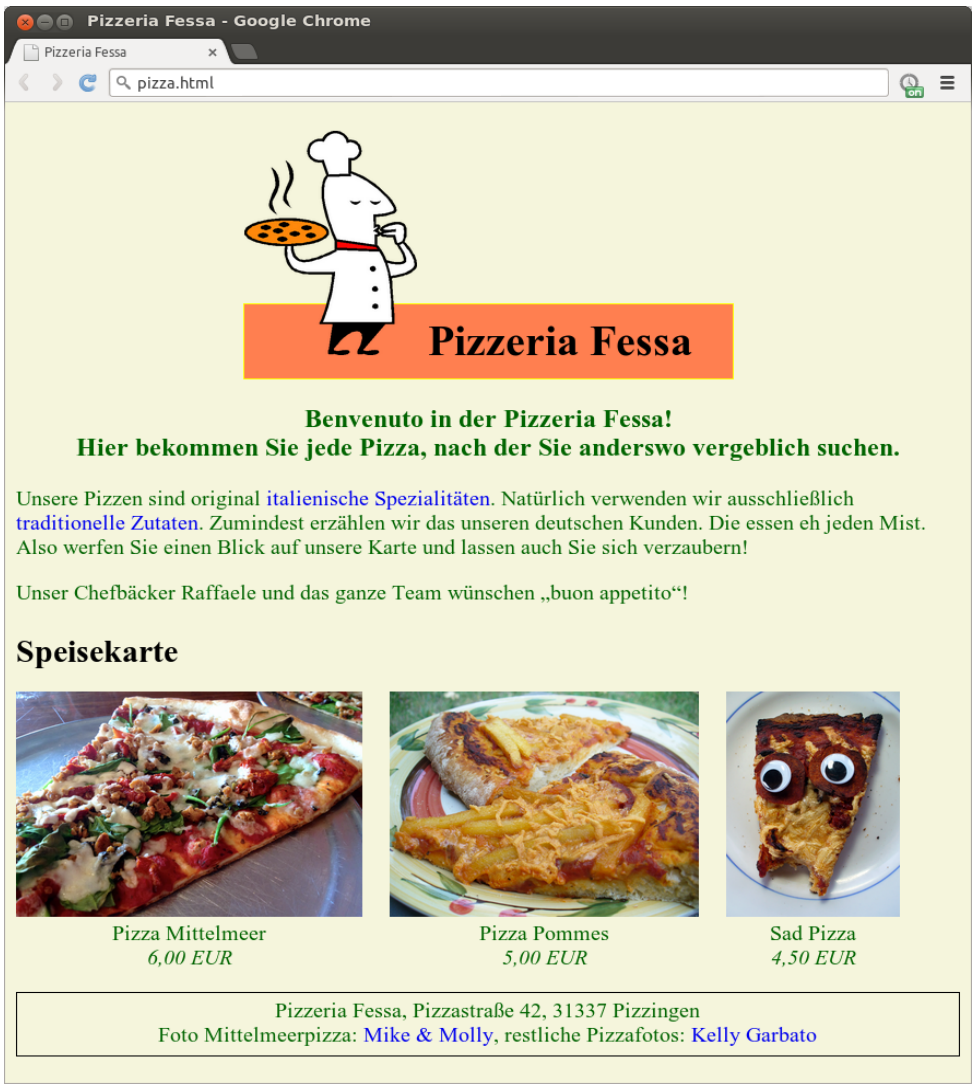

Abb. 1: Das soll sie mal werden: Die Webseite der Pizzeria Fessa

Webseite, wie geht das überhaupt?

Um zu wissen, was beim Erstellen eigener Webseiten auf uns zukommt, untersuchen wir zunächst, woraus eine Webseite eigentlich besteht.

In Abbildung 2 auf der nächsten Seite findest du einen Screenshot einer Webseite. Der Hauptteil bzw. Inhalt ist das, was wir eigentlich sehen wollen, wenn wir eine Webseite besuchen. Er nimmt den größten Teil des Fensters ein. Hier finden wir alle Inhalte der Webseite – Texte, Bilder, Links etc. In HTML werden wir diesen Teil den *body*, „Körper“ einer Webseite nennen.

Daneben gibts es aber auch noch weitere Informationen, die zu einer Webseite gehören. Die Wichtigste davon ist der Titel – er wird auch in der Titelzeile des Browserfensters angezeigt. Zu diesen weiteren Informationen gehören auch unsichtbare Teile, zum Beispiel Informationen, die der Browser braucht, um Um-

Abb. 2: Eine Webseite im Browser

laute und andere Sonderzeichen richtig darstellen zu können (den verwendeten *Zeichensatz*). Diese Zusatzinformationen werden wir in HTML in den Abschnitt *head,* den „Kopf" einer Webseite schreiben.

Eine wichtige Information, die auch mit Webseiten zu tun hat, ist die *Adresse*. Sie ist aber nicht Teil der Webseite selbst und taucht daher im HTML-Code auch nicht auf. Vielmehr ist die Adresse der „Weg" zu einer bestimmten Webseite – sie gibt an, wo sie im Internet abgespeichert ist, wo unser Browser sie suchen muss.

In diesem Kapitel werden wir uns in die Grundlagen der Auszeichnugssprache HTML (*„Hypertext Markup Language"*) einarbeiten, in der die Struktur und der Inhalt von Webseiten geschrieben werden. Im nächsten Kapitel beschäftigen wir uns dann mit dem grafischen Design von Webseiten, für das die Sprache CSS (*„Cascading Style Sheets"*) wichtig ist.

Webseite und Browser

Webseiten sind Dateien – man sagt auch gerne „Dokumente" – die wir uns meistens in einem *Browser* ansehen. Ein Browser ist ein spezielles Programm auf unserem Computer, das eine Webseite aus dem Internet laden und darstellen kann. Auf dem Screenshot im letzten Abschnitt verwende ich den Browser *Google Chrome*. Andere aktuelle Browser sind z.B. *Mozilla Firefox*, *Internet Explorer*, *Opera* oder *Safari*.

Eigentlich sind Webseiten Textdateien, die Code in der Sprache HTML[1] enthalten. Diese Sprache wollen wir hier lernen. Ein Browser interpretiert also diesen von uns geschriebenen Text und baut die Webseite daraus zusammen.

Wenn wir die Begriffe Webseite, HTML-Datei oder HTML-Dokument verwenden, ist also eigentlich immer das Gleiche gemeint.

Den HTML-Text, aus dem eine Webseite besteht, nennt man auch den *Quellcode* dieser Webseite. Diesen sehen wir immer, wenn wir eine HTML-Datei in einem Texteditor, zum Beispiel dem Windows-Editor, öffnen. Wir können diesen Quellcode aber auch in praktisch allen Browsern anzeigen lassen, indem wir auf einer Webseite mit der rechten Maustaste auf eine freie Fläche klicken und „Quelltext anzeigen“ auswählen.

2.1 Grundstruktur einer Webseite

In diesem Abschnitt werden wir Schritt für Schritt die wichtigsten HTML-Elemente kennen lernen, die wir brauchen, um eine ziemlich minimale und hässliche, aber gültige Webseite zu erstellen. Es ist in der Informatik eine Art Tradition, dass man immer, wenn man eine neue Sprache lernt, erstmal etwas schreibt, das einfach nur den Text „hallo, Welt!“ anzeigt. Nicht mehr als das soll deshalb auch auf unserer ersten Webseite stehen.

Vorsicht: Erst am Ende dieses Abschnittes werden wir wirklich eine echte Webseite haben – alle Beispiele bis dahin sind unvollständig und deshalb falsch!

Objekte und Tags – oder: Wie komische Informatiker die Welt sehen

Eine Webseite ist ein aus mehreren Teilen zusammengebautes Ding. Diese Teile nennt man „Objekte“. Ein Objekt kann man sich wie ein Ding aus realen Welt vorstellen, die man anfassen kann: Unser Schulgebäude zum Beispiel ist ein Objekt. Unser Informatikraum ist auch ein Objekt. Das Objekt Informatikraum steckt dabei im Objekt Schulgebäude drin. Im Objekt Schulgebäude gibt es noch viele andere Objekte – zum Beispiel das Objekt Musikraum, das Objekt Aula oder das Objekt Treppe. In unserem Informatikraum befinden sich wiederum viele andere Objekte – vor allem ganz viele Tische, Stühle, Computer und Bildschirme.

So ähnlich ist das auch mit einer Webseite: Zunächst einmal ist die ganze Webseite als solches ein Objekt – das *html*-Objekt. Alles, was auf der Webseite drauf ist oder sonstwie zur Webseite gehört, befindet sich hier drin. Jedes Objekt einer Webseite fängt irgendwo an und hört irgendwo auf – und dazwischen kommt sein Inhalt. Den Anfang markiert man mit einem *öffnenden Tag* (Tag: englisch „Zeichen”), das Ende mit einem *schließenden Tag*.

Das öffnende html-Tag sieht so aus:

```
<html>
```

Das schließende html-Tag sieht ganz ähnlich aus, nur mit einem zusätzlichen Schrägstrich:

```
</html>
```

Also fängt der Quelltext einer Webseite immer mit „<html>“ an und hört mit „</html>“ wieder auf.

Die eigentliche Webseite: der „body“

Text befindet sich im Hauptteil der Webseite. Der Hauptteil ist sozusagen der „Körper“ bzw. *body* der Webseite. Innerhalb des *html*-Objektes befindet sich also ein *body*-Objekt, in dem sich wiederum Text befinden kann.

1 Und ggf. weiteren Sprachen wie CSS und Javascript.

Die erste – unvollständige und deshalb noch falsche – Version unserer Webseite sieht also so aus:

```
<html>
  <body>
    hallo, Welt!
  </body>
</html>
```

In meinem Beispiel habe ich manche Zeilen eingerückt – sie also nicht ganz am linken Rand sondern weiter rechts anfangen lassen. Das ist im Prinzip nicht nötig; in HTML kannst du auch alles direkt untereinander oder sogar hintereinander in eine einzige Zeile schreiben. Ich finde solche Einrückungen aber sinnvoll, damit man direkt sieht, wo ein Objekt anfängt und wo es wieder aufhört. Deshalb rücke ich öffnende und schließende Tags meistens so ein, dass sie gerade untereinander stehen.

Sonstiges Blabla: Der „head“

Wie wir bereits erwähnt haben, besitzen Webseiten neben dem *body* auch einen „Kopf“ bzw. *head*, in dem zusätzliche Informationen stehen. Jede gültige Webseite braucht vor dem Körper einen Kopf, in dem sich mindestens ein Titel – ein *title*-Objekt – befindet.

In der nächsten Version erweitern wir unsere unfertige Webseite also um ein *head*-Objekt, in dem sich ein *title*-Objekt befindet. In dem *title*-Objekt befindet sich der Titel der Seite, zum Beispiel „Hallo-Seite“.

```
<html>
  <head>
    <title>
      Hallo-Seite
    </title>
  </head>

  <body>
    hallo, Welt!
  </body>
</html>
```

Umlaute und der „Zeichensatz“

Webseiten gibt es in allen möglichen Sprachen – und nicht alle verwenden die gleichen Buchstaben. Umlaute und das „ß“ braucht man zum Beispiel auf Deutsch, auf Englisch oder Französisch aber nicht. Chinesische Zeichen muss der Computer für deutsche Texte dagegen nicht kennen.

Deshalb gibt es auch viele verschiedene Arten, wie die Computer solche Zeichen speichern kann: Die sogenannten Zeichensätze. Um Platz zu sparen, war es lange üblich, dass es für fast jede Sprache einen eigenen Zeichensatz gibt, der auch wirklich nur die Zeichen enthält, die für diese Sprache gebraucht werden. Inzwischen ist Speicherplatz billiger geworden und mit „UTF-8“ gibt es einen Zeichensatz, der einfach für alle Schriftzeichen der Welt funktioniert[2]. Den wollen wir deshalb auch benutzen.

Trotzdem werden die vielen Zeichensätze für einzelne Sprachen immer noch benutzt. Damit der Computer weiß, welcher hier der richtige ist (und nicht aus Versehen z.B. ein chinesisches Zeichen statt einem Ö anzeigt), müssen wir ihm erstmal sagen, dass er UTF-8 verwenden soll. Auch das ist eine Zusatzinformationen, die in den *head* der Webseite kommt[3].

2 Genauer gesagt ist UTF-8 ist eine spezielle Darstellungsart der Unicode-Codierung. Diese Darstellungsart wird im westlichen Raum gerne verwendet, da sie für einfache lateinische Buchstaben nicht mehr Speicher verbraucht, als die sprachspezifischen Codierungen. Um das zu erreichen, haben UTF-8-Zeichen eine variable Länge zwischen einem und vier Byte.

```
<!DOCTYPE html>

<html>
  <head>
    <title>
      Hallo-Seite
    </title>
    <meta charset="utf-8" />
  </head>

  <body>
    hallo, Welt!
  </body>
</html>
```

Beispiel 1: Eine einfache „hallo, Welt“-Seite

Diese Information verpacken wir in ein *meta*-Objekt, das für alle unsere Webseiten immer genau so aussieht:

```
<meta charset=“utf-8“ />
```

Wie du merkst, sieht dieses Objekt etwas anders aus als alle anderen, die wir bisher kennengelernt haben. Es hat zum Beispiel gar kein schließendes Tag. Warum das so ist, sehen wir später an zwei anderen Beispielen[4].

Ein letzter Feinschliff

Damit ist unsere Webseite an sich fertig – weitere Objekte müssen wir nicht hinzufügen. Was fehlt, ist noch eine kurze Einleitung, mit der wir einfach nur definieren, dass diese Datei eine Webseite, also ein HTML-Dokument, sein soll. Diese sogenannte *Dokumenttyp-Deklaration* lautet „<!DOCTYPE html>“[5] und muss ganz am Anfang jeder HTML-Datei stehen.

Wie das HTML für unsere Mini-Seite fertig aussieht, siehst du oben in Beispiel 1.

Probiere aus, diese Webseite in deinem Browser zu öffnen! Dazu speicherst du sie im Texteditor am Besten unter einem Namen ab, der mit „.html“ aufhört, also z.B. „webseite.html“. So weiß dein Computer, dass er diese Datei mit einem Browser öffnen muss. Auch wichtig: Wir haben angegeben, dass unsere Webseite UTF-8 als Zeichencodierung benutzt. Beim Speichern müssen wir aufpassen, dass sie das auch wirklich tut! Wenn du deine Webseiten im Windows-Editor schreibst, musst du dazu beim Speichern unter *Codierung* die Option *UTF-8* auswählen.

Das war's! Mit einem Doppelklick auf die Datei, unter der du sie gespeichert hast, sollte sich deine Webseite anzeigen.

---

3 Traditionell wird der Zeichensatz dem Browser unabhängig von der eigentlichen Website direkt vom Server über eine HTTP-Kopfzeile mitgeteilt. Der HTML-Standard schreibt aber eine Zeichensatz-Deklaration direkt im HTML-Kopf vor, wenn dies nicht der Fall ist – oder, wie in unserem Fall, gar kein HTTP-Server verwendet wird, der die Information übermitteln könnte.

4 Stichworte: Inhaltleeres Objekt und Attribute

5 Früher enthielt die Dokumenttyp-Deklaration zusätzlich die genaue HTML-Version sowie einen Link zu einer formalen Sprachdefinition (*DTD*). Seit HTML5 entfällt dies, womit die Dokumenttyp-Deklaration ihren eigentlichen Sinn verliert. Sie zeigt modernen Browsern jedoch an, dass diese Webseite „neu“ ist nicht etwa auf die fehlerhafte Seitendarstellung älterer Browser „optimiert“ wurde. Der HTML5-Doctype stellt also sicher, dass der Browser nicht in den sogenannten „Quirks mode“ schaltet.

```
<!DOCTYPE html>

<html>
  <head>
    <title>
      Hallo-Seite
    </title>
    <meta charset="utf-8" />
  <head>

  <body>
    hallo, Welt!
    <b>hallo, Welt!</b>
    <i>hallo, Welt!</i>
    <u>hallo, Welt!</u>
  </body>
</html>
```

Beispiel 2: „hallo, Welt" – die Zweite

Einschub: Browser und der HTML-Standard

Wie schon erwähnt sind Webseiten eigentlich Textdateien. Wie sie am Ende wirklich aussehen, interpretiert der Browser aus diesem Text – und Browser gibt es viele verschiedene! Damit Webseiten in jedem Browser trotzdem immer gleich, immer „richtig" aussehen, gibt es einen HTML-Standard[6], der vorschreibt, wie eine gültige Webseite aufgebaut sein muss.

Wir lernen in diesem Kurs, gültige, standardkonforme Webseiten zu schreiben. Das ist nicht selbstverständlich: So ein Browser muss bei seiner Reise durchs Internet ziemlich viel aushalten. Viele „Webdesigner" schreiben HTML-Code, der falsch ist, das heißt nicht dem Standard entspricht. Browser sind darauf ausgelegt, aus solchem falschen Code das Beste herauszuholen – sie versuchen die Webseite also trotzdem irgendwie darzustellen.

Wenn du bei den einfachen Webseiten, die wir hier schreiben, mal etwas vergisst und deine Webseite deshalb falsch ist, kann es passieren, dass du das gar nicht so leicht merkst. Vielleicht stellt der Browser deine Seite trotzdem richtig dar – schließlich sind Browser dafür gemacht, möglichst oft richtig zu raten, was der Autor gemeint haben könnte! Sobald du aber mit deiner Webseite den Standard verletzt, also falsches HTML benutzt, ist nicht mehr garantiert, dass deine Seite in allen Browsern richtig aussieht.

Ein nützliches Hilsmittel ist der *W3C-Validator*, den du unter *http://validator.w3.org* findest. Er wurde von den Machern des HTML-Standards geschrieben, damit du testen kannst, ob deine Webseiten gültiges HTML sind. Dort kannst du entweder eine Internetadresse eingeben, die du checken willst („Validate by URI") oder deine eigene HTML-Datei zum Testen hochladen („Validate by File Upload").

2.2 Einfache Textformatierung

Wenn du dich mit Textverarbeitungsprogrammen wie OpenOffice oder Microsoft Word auskennst, kennst du sicher drei wichtige Textformatierungen: Fettdruck, Kursivdruck und Unterstreichen. Die gibt es natürlich auch in HTML!

Wir fügen unserer „hallo, Welt"-Webseite deswegen nach dem normalen „hallo, Welt" einfach mal noch drei weitere hinzu: Ein fettes, ein kursives und ein unterstrichenes. Fangen wir mit dem fetten an.

6 Und Standards für weitere Web-Sprachen wie CSS und Javascript.

```
<!DOCTYPE html>

<html>
  <head>
    <title>
      Hallo-Seite
    </title>
    <meta charset="utf-8" />
  </head>

  <body>
    <p>
      hallo, Welt!
    </p>
    <p>
      <b>hallo, Welt!</b><br />
      <i>hallo, Welt!</i><br />
      <u>hallo, Welt!</u><br />
    </p>
  </body>
</html>
```

Beispiel 3: „hallo, Welt“ – die Dritte

Um Fettdruck zu verwenden, musst du den Text, der fett werden soll, in ein entsprechendes Objekt hineinpacken. In HTML ist das ein *b*-Objekt (b steht für engl. *bold*: fett). Wie allen anderen Arten von HTML-Objekten, die wir kennen gelernt haben, hat auch ein *b*-Objekt ein öffnendes und ein schließendes *Tag*, zwischen dem der Inhalt steht:

```
<b>hallo, Welt!</b>
```

Packen wir jetzt noch zwei weitere „hallo, Welt!“-s auf die Seite – das kursive und das unterstrichene. Für Kursivdruck brauchen wir ein *i*-Objekt (i steht für englisch *italic*: „kursiv“), fürs Unterstreichen ein *u*-Objekt (das englische *underline* fängt praktischerweise auch mit u an).

Wie unsere Mini-Webseite jetzt aussieht, siehst du in Beispiel 2.

Wenn du diese Seite jetzt in deinem Browser öffnest, fällt dir auf, dass fett, kursiv und unterstrichen wie erwartet funktionieren. Die Zeilenumbrüche, die wir zwischen den drei „hallo, Welt!“-s eingefügt haben, sind aber verschwunden – während in unserem Quelltext jedes „hallo, Welt!“ in einer eigenen Zeile steht, hängen sie im Browser hintereinander in einer Zeile.

Zeilenumbrüche, Leerzeilen oder auch mehrere Leerzeichen hintereinander werden in HTML einfach ignoriert. Wenn du deine Seite in mehrere Absätze aufteilen willst, musst du den Text in entsprechende Absatz-Objekte hineinpacken. In HTML sind das die *p*-Objekte (p steht für englisch *paragraph*: „Absatz“).

In unserer Beispielwebseite legen wir jetzt zwei Absätze an: Im ersten soll das ganz normale „hallo, Welt!“ stehen, im zweiten die fette, kursive und unterstrichene Version. Dann befindet sich auf der Seite oben nur das normale „hallo, Welt!“ und mit etwas Abstand darunter stehen in einer Zeile die drei speziell formatierten Versionen.

Manchmal möchte man Textstücke gar nicht durch einen ganzen Absatz trennen, sondern einfach nur in der nächsten Zeile weiterschreiben. Um einen solchen Zeilenumbruch einzufügen brauchen wir in HTML ein *br*-Objekt (br steht für englisch *break*: „Umbruch“). Ein *br*-Objekt ist ein etwas besonderes Ding: Es fängt nicht an einer Stelle an und hört an einer anderen wieder auf, wie wir das von anderen Objektarten kennen – es kann nichts in ihm drin stehen. Wie soll es ach aussehen, wenn in einem Zeilenumbruch etwas

drin steht? Entsprechend hat es auch nicht einen öffnenden und einen schließenden Tag. Ein *br*-Objekt ist inhaltsleer: Es fängt an und hört sofort wieder auf. *br* hat deshalb nur einen Tag, der öffnend und schließend gleichzeitig ist:

```
<br />
```

Dieses *br*-Objekt kommt vor den Text, der in der nächsten Zeile stehen soll. In unserer Beispielseite werden wir jetzt den speziell formattierten „hallo, Welt"-s je eine eigene Zeile geben – sie also durch Zeilenumbrüche trennen. Wie der Quellcode der Seite danach aussieht, siehst du in Beispiel 3.

Übrigens: Eine andere Art inhaltsleerer Objekten kennen wir schon seit unserer allerersten Test-Webseite. *meta*-Objekte – die wir benutzen, um den Zeichensatz der Webseite anzugeben – funktionieren nach genau dem gleichen Prinzip.

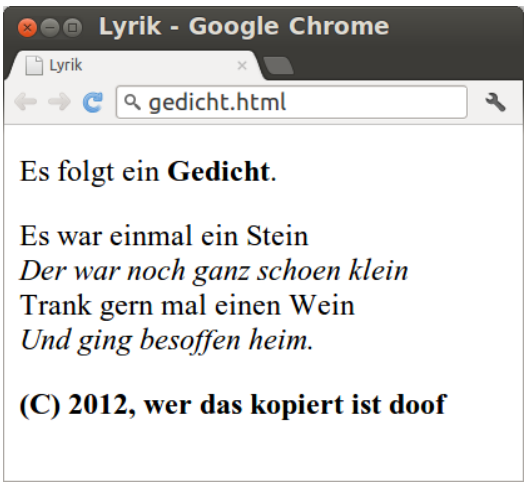

Abb. 3: Schreibe diese Webseite!

Übung: Textformatierung

In Abb. 3 siehst du einen Screenshot einer ganz einfachen Webseite, die nur formatierten Text enthält. Baue diese Webseite ganz genau nach! Denk dabei an den Unterschied zwischen Zeilenumbrüchen und Absätzen.

Wenn du fertig bist, vergiss nicht, mit dem Validator (*http://validator.w3.org*) zu checken, ob dein HTML auch wirklich richtig ist.

Remember the Pizza

Damit verabschieden wir uns auch endlich wieder von unserem „hallo, Welt". Kommen wir zurück zu Raffaele und seiner Pizzeria-Webseite. Wir sind jetzt soweit, dass wir schonmal anfangen können, einen Teil von ihr zu schreiben.

Schau nochmal auf den Screenshot ganz am Anfang in Abbildung 1 (das war auf Seite 5). Wir sehen verschiedene Absätze Text und einige Blöcke, die zumindest irgendwie ähnlich wie Absätze aussehen (zum Beispiel die einzelnen Pizzen mit Foto, Name und Preis).

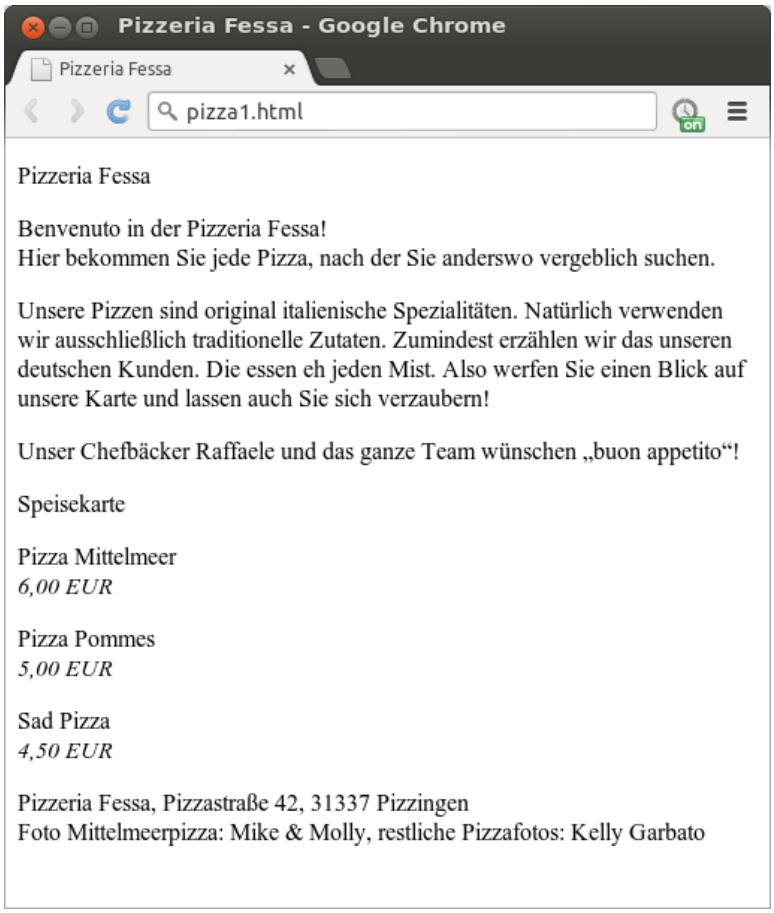

Abb. 4: Pizza, die Erste!

Wir erstellen alle diese Absätze als *p*-Objekte, schnappen uns dann erstmal nur den Text und schreiben ihn in diese Absätze. Das sieht jetzt zugegebenermaßen noch ziemlich unspektakulär aus – nämlich so, wie in Abbildung 4. Den HTML-Quellcode findest du in Beispiel 4.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>Pizzeria Fessa</title>
  </head>

  <body>
    <p>
      Pizzeria Fessa
    </p>

    <p>
      Benvenuto in der Pizzeria Fessa!<br />
      Hier bekommen Sie jede Pizza, nach der Sie anderswo vergeblich suchen.
    </p>
    <p>
      Unsere Pizzen sind original italienische Spezialitäten. Natürlich
      verwenden wir ausschließlich traditionelle Zutaten. Zumindest erzählen
      wir das unseren deutschen Kunden. Die essen eh jeden Mist. Also werfen
      Sie einen Blick auf unsere Karte und lassen auch Sie sich verzaubern!
    </p>
    <p>
      Unser Chefbäcker Raffaele und das ganze Team wünschen „buon appetito“!
    </p>

    <p>Speisekarte</p>
    <p>
      Pizza Mittelmeer<br />
      <i>6,00 EUR</i>
    </p>

    <p>
      Pizza Pommes<br />
      <i>5,00 EUR</i>
    </p>

    <p>
      Sad Pizza<br />
      <i>4,50 EUR</i>
    </p>

    <p>
      Pizzeria Fessa, Pizzastraße 42, 31337 Pizzingen<br />
      Foto Mittelmeerpizza: Mike & Molly, restliche Pizzafotos: Kelly Garbato
    </p>

  </body>
</html>
```

Beispiel 4: Pizza, die Erste!

2.3 Bilder

Welche Webseiten kommt schon ganz ohne Bilder aus? Es wird Zeit, die Fotos der Speisekarte und natürlich das stolze Portrait von Raffaele in die Webseite einzufügen. Für ein Bild auf brauchen wir in HTML ein *img*-Objekt (img steht für englisch *image*: „Bild“) .

Genauso, wie wir es auch von *br*-Objekten kennengelernt haben, ist ein *img*-Objekt *inhaltsleer*. Das heißt, es kann nichts in einem solchen Objekt drinstehen. Es hat nicht einen öffnenden und einen schließenden Tag, sondern einen, der öffnender und schließender Tag zugleich ist: *<img />*

Ein solches *img*-Objekt macht aber noch keinen Sinn: „Zeig mir ein Bild an!“ - und welches?! Einem *img*-Objekt müssen wir deshalb immer noch zusätzliche Informationen mitgeben – Dinge, die das Objekt sich bitte merken soll. Solche Zusatzinformationen an ein bestimmtes Objekt nennt man *Attribute*. Sie werden innerhalb des Tags angegeben.

Das wichtigste Attribut eines *img*-Objekts bestimmt natürlich, welches Bild denn angezeigt werden soll. Dieses Attribut heißt *src* (src steht für englisch *source*: „Quelle“).

Mit diesem Attribut müssen wir entweder die komplette Internetadresse eines Bildes angeben, oder, wenn es sich im gleichen Ordner wie unsere Webseite befindet, einfach seinen Dateinamen[7]. Fangen wir mit dem Bild von Raffaele ganz oben in der Überschrift der Webseite an. Es hat den Dateinamen „raffaele.png“. Das geben wir mit dem Attribut *src="auto.jpg"* an.

Ein weiteres Attribut ist laut HTML-Standard Pflicht für alle *img*-Objekte: Mit dem Attribut *alt* (englisch für *alternative*: „ersatzweise“) muss ein kurzer Beschreibungstext des Bildes angegeben werden. Dieser wird im Browser angezeigt, wenn das Bild nicht geladen werden kann oder noch nicht geladen wurde. Wichtig sind diese *alt*-Attribute aber zum Beispiel auch für blinde Menschen, die Webseiten in Blindenschrift „fühlen“ oder sie sich vom Computer vorlesen lassen. Auch für Suchmaschinen, die (noch?) keine Bilder verstehen, sind solche *alt*-Attribute wichtig, um den Inhalt einer Webseite erfassen zu können.

Als *alt*-Attribut könnten wir also zum Beispiel *alt="Raffaele"* lauten. Wenn ein Bild überhaupt keine Bedeutung haben sollte (also reine Dekoration ist, wie zum Beispiel ein Rahmen), muss ein leeres *alt*-Attribut angegeben werden: *alt=""*. (Auch das könnte man sich in diesem Fall überlegen, da die Zeichnung, so hübsch Raffaele auch sein mag, nur die Überschrift hübscher macht und nichts wirklich eigenes aussagt.)

Unser *img*-Tag für Raffaele sieht also so aus:

```
<img src="raffaele.png" alt="Raffaele" />
```

Wenn wir den Tag so lassen, wird das Bild in seiner Originalgröße eingebunden. Das ist vielleicht nicht das, was wir wollen – vor allem zu groß sind Bilder oft. *img*-Objekte haben noch zwei weitere Attribute, mit denen die Größe bestimmt werden kann, nämlich *width* (englisch „Breite“) und *height* (englisch „Höhe“).

Du kannst entweder nur eines dieser Attribute setzen, also entweder nur Breite oder nur Höhe angeben (wenn du z.B. die Breite angibst, wird die Höhe daraus automatisch ausgerechnet) – oder aber auch beide. Wenn du sowohl Höhe als auch Breite angibst, wird das Bild im Zweifelsfall verzerrt, wenn das Seitenverhältnis nicht stimmt.[8]

Breite und Höhe eines Bilder werden in Anzahl Pixel angegeben[9]. Pixel sind die kleinen, einzelnen Punkte,

7 Natürlich sind auch komplexe relative Pfadangaben möglich. Im Rahmen dieses Kurses sparen wir uns das und packen Bilder immer zusammen mit den HTML-Dateien in einen gemeinsamen Ordner.

8 Es gilt als guter Stil, die Größe des Bildes immer und zwar in Höhe und Breite anzugeben. So kann der Browser beim Laden der Seite schon den richtigen Platz für das Bild reservieren, bevor das Bild geladen wurde. Der Einfachkeit halber geben wir aber erstmal immer nur einen Wert an.

9 Alternativ können Höhe und Breite auch in Prozent angegeben werden. Die Prozentwerte beziehen sich auf die Größe des Elements, in dem das Bild enthalten ist (z.B. ein Absatz). Prozentangaben sind bei Pixelgrafiken nur in Einzelfällen sinnvoll, da diese nicht frei skalierbar sind.

aus denen das Bild zusammengesetzt ist.

Raffaele soll in unserem Beispiel 170 Pixel hoch sein. Wir setzen also das Attribut *height="170"*. Dann sieht unser fertiger *img*-Tag so aus:

```
<img src="raffaele.png" alt="Raffaele" height="170" />
```

Nach dem gleichen Prinzip fügen wir jetzt noch die Speisekarten-Fotos ein. Sie heißen *mittelmeer.jpg*, *pommes.jpg* und *sadpizza.jpg*. Auch sie werden in unserem Beispiel jeweils 170 Pixel hoch sein. Der *alt*-Ersatztext könnte so etwas wie „Foto der Mittelmeerpizza“ lauten.

2.4 Links

Links sind Texte oder Bilder auf einer Webseite, die man anklicken kann, um zu einer anderen Seite zu kommen. Um einen Link zu erstellen, müssen wir den Text (oder auch das Bild), der anklickbar werden soll, in ein *a*-Objekt packen (a steht für englisch *anchor*: „Anker“). Damit der Link funktioniert, müssen wir natürlich angeben, auf welche Seite man kommen soll, wenn man ihn anklickt. Dafür bekommen unsere *a*-Objekte noch ein Attribut *href* (href steht für englisch *hypertext reference*: Hypertext ist ein Fachbegriff für Texte, die Links enthalten und reference bedeutet „Verweis“).

Der Inhalt des *href*-Attributs sieht genauso aus wie der Inhalt des *src*-Attributs bei Bildern (*img*-Objekten): Eine Internetadresse oder der Dateiname einer Datei im gleichen Ordner wie diese Webseite. Nur sollte es diesmal keine Bilddatei sein, sondern eine weitere Webseite, die beim Anklicken des Links geöffnet wird[10].

Nehmen wir also an, wir wollen einen Textlink auf eine Seite *beispiel.html* erstellen, die im gleichen Ordner wie unsere aktuelle Webseite liegt. Das könnte so aussehen:

```
<a href="beispiel.html">Ich bin ein Link, klick mich an!</a>
```

Unsere Pizza-Webseite besteht zwar nur aus einer einzelnen Seite – sie besitzt aber vier Links auf andere Websites. Im Satz:

„Unsere Pizzen sind original italienische Spezialitäten. Natürlich verwenden wir ausschließlich traditionelle Zutaten.“

ist „italienische Spezialitäten“ ein Link auf die Wikipedia-Seite *Italien*, die die Adresse http://de.wikipedia.org/wiki/Italien hat. „traditionelle Zutaten“ verlinkt auf die Wikipedia-Seite *Mononatriumglutamat* (http://de.wikipedia.org/wiki/Mononatriumglutamat), um die besondere Qualität zu unterstreichen.

2.5 Überschriften

Sowohl die große Überschrift ganz oben auf der Seite – „Pizzeria Fessa“ mit dem Bild von Raffaele – als auch die kleinere Überschrift „Speisekarte“ sehen in unserem aktuellen Stand noch nicht wirklich Überschrift-artig aus.

Das lässt sich leicht ändern, indem wir aus ihnen statt normalen Absätzen (p-Objekten) spezielle Überschriften machen: HTML bieten fünf verschiedene Arten von Objekten für unterschiedlich wichtige Überschriften. Die größte und wichtigste Überschrift auf einer Seite – also genau richtig für „Pizzeria Fessa“ – kommt in ein *h1*-Objekt (h steht für englisch *header*: „Überschrift“).

Das Bild von Raffaele in der Überschrift „Pizza Fessa“ stört übrigens überhaupt nicht. Es kommt einfach auch mit in die *h1*-Überschrift rein.

10 Links auf Bilder funktionieren aber auch: Dann wird beim Anklicken des Links das Bild alleine im Browser angezeigt. Auch andere Dateitypen kannst du verlinken: Wenn der Browser die Datei nicht direkt anzeigen kann, wird sie zum Download angeboten.

Einen Zwischenstand, wie unsere Pizza-Webseite inzwischen aussieht, findest du in Abbildung 5. Leider hatte nicht die komplette Seite auf meinem Bildschirm Platz, so dass im Screenshot nur der obere Teil zu sehen ist. Den kompletten HTML-Quellcode zum aktuellen Stand findest du in Beispiel 5.

Ein *h1*-Objekt steht also für eine Überschrift der obersten Ebene, eine echte **Über**-Schrift, die über allem anderen steht.

In ein *h2*-Objekt packt man dagegen eine etwas weniger wichtige, untergeordnete Überschrift – in unserem Beispiel perfekt für die Überschrift „Speisekarte“.

Entsprechend steht ein *h3*-Objekt für eine unter-unter-geordnete Überschrift, *h4*- oder sogar *h5*-Objekte für noch weniger wichtige Überschriften. Für unsere Pizza-Webseite brauchen wir aber nur *h1* und *h2*.

Diese speziellen Überschrift-Objekte sollte man in HTML auch unbedingt verwenden. Man könnte ja stattdessen auf die Idee kommen, einfach normale Absätze (*p*-Objekte) und Fettdruck (*b*-Objekte) zu einer „Überschrift“ zu kombinieren. Sobald wir uns mit Grafikdesign (Kapitel 3) beschäftigt haben, werden wir noch viel mehr Möglichkeiten kennen, einfache Absätze als Überschriften zu „verkleiden“.

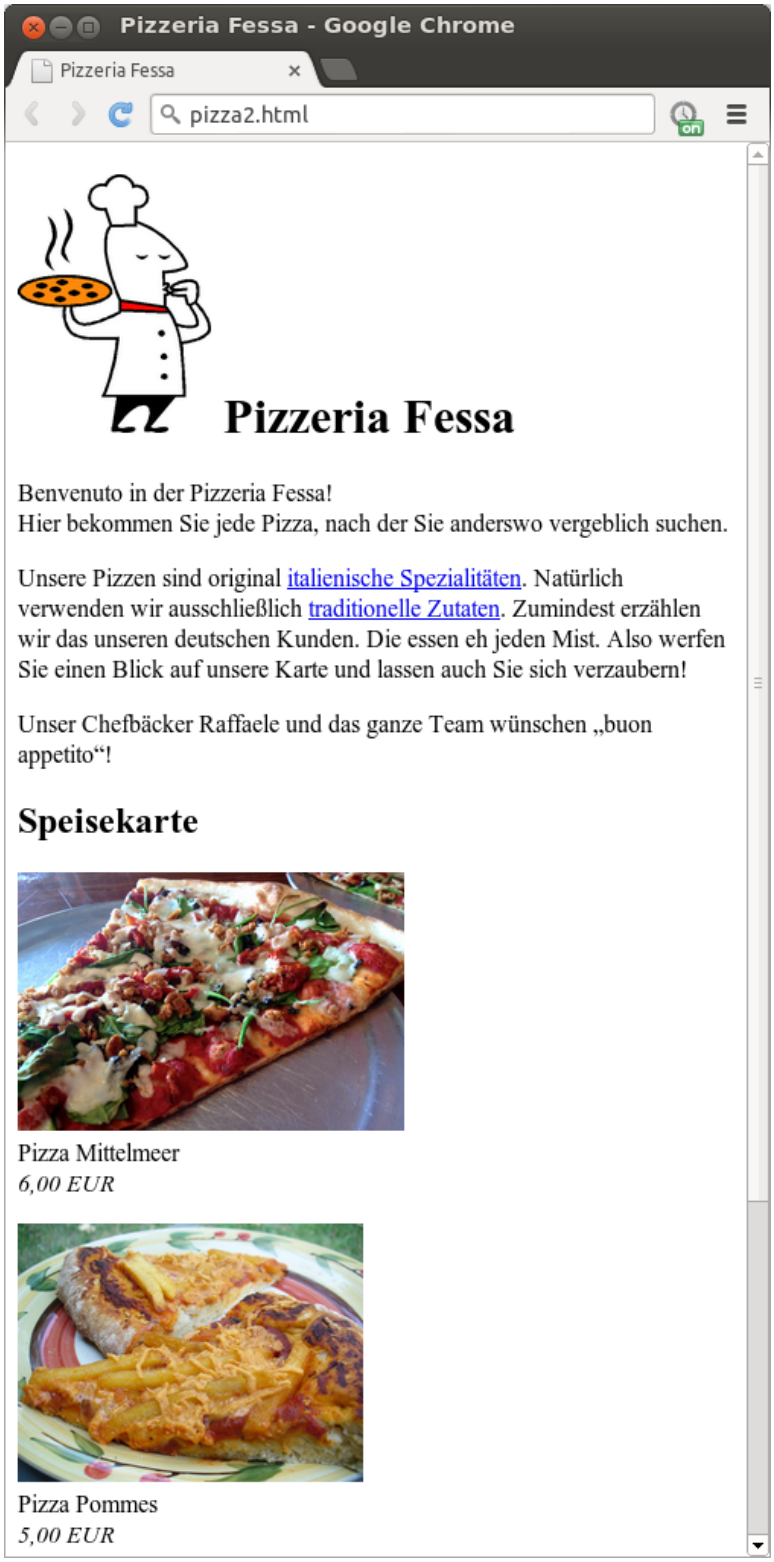

Abb. 5: Pizza – mit Bildern, Links und Überschriften

Trotzdem sollte man das auf keinen Fall tun. Der Grund dafür ist der gleiche, den wir schon im Abschnitt über Bilder (2.3) kennen gelernt haben. Dort sagten wir, dass Bilder immer ein *alt*-Attribut haben müssen, das eine Textbeschreibung über das Bild enthält. Für welche Gruppen von Lesern wir uns solche Mühe geben (müssen), haben wir auch schon im Abschnitt über Bilder (2.3) kennen gelernt: Bestimmte Menschen, wie z.B. Blinde und Sehbehinderte, und Computer, wie z.B. Suchmaschinen. Kurz gesagt alle, die Webseiten nicht anhand ihres Aussehens lesen können, die Grafikdesign nicht verstehen können. Und was dort für Bilder gilt, gilt hier genauso auch für Überschriften.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>Pizzeria Fessa</title>
  </head>

  <body>
    <h1>
      <img src="raffaele.png" height="170" alt="" />
      Pizzeria Fessa
    </h1>

    <p>
      Benvenuto in der Pizzeria Fessa!<br />
      Hier bekommen Sie jede Pizza, nach der Sie anderswo vergeblich suchen.
    </p>
    <p>
      Unsere Pizzen sind original
      <a href="http://de.wikipedia.org/wiki/Italien"> italienische
      Spezialitäten</a>. Natürlich verwenden wir ausschließlich
      <a href="http://de.wikipedia.org/wiki/Mononatriumglutamat">traditionelle
      Zutaten</a>. Zumindest erzählen wir das unseren deutschen Kunden. Die
      essen eh jeden Mist. Also werfen Sie einen Blick auf unsere Karte und
      lassen auch Sie sich verzaubern!
    </p>
    <p>
      Unser Chefbäcker Raffaele und das ganze Team wünschen „buon appetito“!
    </p>

    <h2>Speisekarte</h2>
    <p>
      <img src="mittelmeer.jpg" height="170" alt="Foto der Pizza Mittelmeer" />
      <br />
      Pizza Mittelmeer<br />
      <i>6,00 EUR</i>
    </p>

    <p>
      <img src="pommes.jpg" height="170" alt="Foto der Pizza Pommes" /><br />
      Pizza Pommes<br />
      <i>5,00 EUR</i>
    </p>

    <p>
      <img src="sadpizza.jpg" height="170" alt="Foto der Sad Pizza" /><br />
      Sad Pizza<br />
      <i>4,50 EUR</i>
    </p>

    <p>
      Pizzeria Fessa, Pizzastraße 42, 31337 Pizzingen<br />
      Foto Mittelmeerpizza:
      <a href="http://www.flickr.com/people/veganflower/">Mike & Molly</a>,
      restliche Pizzafotos:
      <a href="http://www.flickr.com/people/smiteme/">Kelly Garbato</a>
    </p>

  </body>
</html>
```

Beispiel 5: Pizza – mit Bildern, Links und Überschriften

Das Prinzip ist also eigentlich ganz einfach: Was die einzelnen Teile einer Webseite bedeuten, muss immer in reiner Textform erkennbar sein – also in einer Form, die man nicht nur durch *sehen* wahrnehmen kann. Dieses Prinzip hat auch einen Namen: *Semantisches Web*.

Semantik ist ein Begriff aus der Kommunikationswissenschaft und bezeichnet die inhaltliche *Bedeutung* einer Nachricht oder eines Textes – im Gegensatz zur sogenannten Syntax, die die Formulierung und die Wortwahl der Nachricht beschreibt.

2.6 Objekte und Klassen

Bevor wir zum Design übergehen und unsere Webseiten endlich aus ihrem ewigen Schwarzweiß befreien, sprechen wir noch kurz über unsere ganzen Objekte. Einen Begriff, der eigentlich total einfach ist und den wir für das nächste Kapitel brauchen, haben wir bis jetzt einfach nicht erwähnt: Den Begriff der Klasse.

Die Klasse beschreibt, von welcher Art ein Objekt ist. Es gibt zum Beispiel die Klasse der *p*-Objekte, mit der wir Absätze definieren. Eine andere Art, also eine andere *Klasse* von Objekten, benutzen wir, wenn wir Bilder einfügen wollen: Die Klasse *img*. Wenn wir Links einfügen wollen, erstellen wir uns Objekte der Klasse *a*. Ohne den Begriff zu verwenden, haben wir bis jetzt schon viele verschiedene Klassen – verschiedene Arten von Objekten – kennen gelernt.

Erinnern wir uns an unsere Definition eines Objekts: Ein Objekt ist ein konkretes Ding – wie ein bestimter Bildschirm in der realen Welt. Zwei Bildschirme sind zwei verschiedene Objekte. Ich kann entweder den einen Bildschirm anfassen, oder den anderen, oder beide gleichzeitig. Ein grauer Bildschirm ist kein blauer Bildschirm. Dein Bildschirm ist nicht mein Bildschirm. Die beiden Bildschirm-Objekte sind verschieden. Trotzdem gehören beide zur Klasse Bildschirm.

Eine Klasse ist eine abtrakte Idee und kann sozusagen der Bauplan für eine Art von Objekten sein. Die Klasse „Bildschirm“ in der realen Welt können wir nicht anfassen. Trotzdem wissen wir, dass jeder Bildschirm grundsätzlich *das gleiche hat* und *das gleiche kann*. Jeder Bildschirm hat einen Anschluss, um ihn mit dem Computer zu verbinden und jeder Bildschirm hat einen Stromanschluss. Jeder Bildschirm kann ein Bild anzeigen. Jedes Objekt der Klasse Bildschirm hat diese Eigenschaften.

Ähnlich ist es mit Objekten und Klassen im Computer. Ein bestimmtes Bild ist ein konkretes, eigenständiges Objekt. Haben wir auf einer Webseite zwei Bilder, sind das zwei eigenständige Objekte, die eigentlich überhaupt nichts miteinander zu tun haben. Man kann sagen, sie kennen sich untereinander nicht einmal. Trotzdem sind beide Objekte der gleichen Klasse, nämlich der Klasse *img*.

Das bedeutet zum Beispiel, dass beide *img*-Objekte die gleichen Eigenschaften haben: Beide haben z.B. das Attribut *src*, beide sind inhaltsleere Objekte (also Objekte, die nicht einen öffnenden und später einen schließenden Tag haben sondern immer sofort wieder zu gehen) und natürlich sorgen beide dafür, dass ein Bild auf unserer Webseite erscheint.

3 Das Web wird bunt: Design mit CSS

Jetzt wird's kreativ: In diesem Kapitel geht es um Grafikdesign für Webseiten. Wie wir im Abschnitt über das Überschriften und das semantische Web (2.5) erwähnt haben, soll eine HTML-Datei – also die Hauptdatei einer Webseite – nach dem Inhalt der Webseite aufgebaut werden und nicht danach, wie die Webseite grafisch aussehen soll.

In diesem Kapitel lernen wir deshalb eine eigene Sprache, die speziell nur für das grafische Design gemacht ist: Cascading Style Sheets, kurz CSS.

In einer extra Datei, dem *Stylesheet* (*sheet*: englisch „Blatt"), kann man verschiedene Design-Vorgaben angeben, zum Beispiel Farbe, Schriftart, Größe, Rahmen etc.

Stylesheet in die Webseite einbinden

Ein Stylesheet ist eine eigene Textdatei, die meistens unter einem Namen gespeichert wird, der mit „.css" aufhört. Für unsere Pizza-Seite wollen wir das Stylesheet zum Beispiel „pizza.css" nennen. Damit der Browser beim Aufruf unserer Webseite auch weiß, dass es noch ein Stylesheet dazu gibt, müssen wir das im HTML-Quellcode angeben. Das tun wir im *head* mit einem *link*-Objekt.

```
<link href="pizza.css" rel="stylesheet" />
```

Schauen wir uns das *link*-Objekt kurz genauer an: Es ist ein inhaltsleeres Objekt (hat also nicht einen öffnenden und später einen schließenden Tag sondern nur einen Tag, der auf und gleichzeitig wieder zu geht) mit drei Attributen:

- Das Attribut *rel* teilt dem Browser mit, zu welchem Zweck eine Datei eingebunden werden soll – in unserem Fall zur Verwendung als „stylesheet".

- Das Attribut *href* kennen wir schon von Links: Hier geben wir an, wo der Browser nach der einzubindenden Datei suchen soll – in unserem Fall nach einer Datei namens „pizza.css", die im gleichen Ordner wie die HTML-Datei liegt. Wenn wir unser Stylesheet schreiben, sollten wir also daran denken, es auch wirklich unter diesem Namen zu speichern[11].

Grundgerüst von CSS

In CSS kann man Design-Regeln den verschiedenen Klassen von HTML-Objekten zuweisen. Im ersten Schritt lernen wir deshalb, wie man zum Beispiel alle Absätze (also die Klasse der *p*-Objekte) oder alle Links (also die Klasse der *a*-Objekte) designt. Später werden wir auch lernen, wie wir gezielter einzelnen Objekten Designs zuweisen können.

Für jede Klasse, der wir ein Design geben wollen, enthält ein Stylesheet einen eigenen Regelblock. Dieser beginnt immer mit dem Namen der Klasse. Dahinter kommen in geschweiften Klammern eingeschlossen alle Attribute, also die jeweils einzelnen Regeln (wie diese Designregeln genau aussehen, kommt gleich).

Wollen wir zum Beispiel alle Absätze (die Klasse der *p-Objekte*) designen, sieht der Regelblock dafür im Stylesheet also so aus:

```
p
{
  /* Regel 1 */
  /* Regel 2 */
```

11 Neben „pizza.css" und anderen Dateinamen könnte man hier auch einen relativen Pfad oder eine Internetadresse (URL) angeben.

```
  /* Regel 3 */
  /* ... */
}
```

Übrigens: Texte, die zwischen „/*“ und „*/“ stehen, sind in Stylesheets sogenannte Kommentare: Sie werden vom Computer einfach ignoriert. Man kann dort also reinschreiben, was man will – es ändert überhaupt nichts.

3.1 Designregeln für HTML-Klassen

Wir werden jetzt einfach mal anfangen, uns dem Design der Pizza-Seite, wie wir sie ganz am Anfang in Abbildung 1 (Seite 5) gesehen haben, weiter anzunähern. Dabei werden wir verschiedene übliche CSS-Attribute kennenlernen und verschiedene HTML-Klassen auf die eine oder andere Art formatieren. Alle CSS-Regeln lassen sich auf jede beliebige HTML-Klasse anwenden[12].

Textfarbe

Geben wir als erstes allen Absätzen eine besondere Schriftfarbe. Wie wäre es mit einem Dunkelgrün?

Wie wir einen Regelblock für die Klasse *p* anlegen, haben wir schon am Anfang dieses Kapitels besprochen. In diesen Block schreiben wir jetzt ein einziges Attribut, nämlich für dunkelblaue Schriftfarbe. Dazu schreiben wir zuerst den Namen des Attributs – in unserem Fall *color* – und dahinter einen Doppelpunkt. Anschließend schreiben wir, welchen Wert wir für dieses Attribut setzen wollen, z.B. *DarkGreen*. Anführungszeichen sind in CSS hierfür nicht nötig, können aber verwendet werden. Ans Ende jedes Attributs, hinter den Wert, kommt ein Semikolon.

```
p
{
  color: DarkGreen;
}
```

Hintergrundfarbe

Formatieren wir als nächstes das ganze *body*-Objekt und geben damit unserer ganzen Webseite eine Hintergrundfarbe. Für die Pizzaseite soll es *beige* sein, ein leichtes grau. Das Attribut für Hintergrundfarbe heißt *background-color*, wir setzen es also auf den Wert *beige*.

```
body
{
  background-color: beige;
}
```

Ausrichtung und Rahmen

Als nächstes möchten wir, dass bei Überschriften erster Ebene – also der Klasse *h1* – der Text zentriert erscheint, einen korallenroten (*coral*) Hintergrund und einen gelben Rahmen hat.

Zentrierung erreichen wir, indem wir das Attribut *text-align* auf *center* setzen.

```
h1
{
  background-color: Coral;
  text-align: center;
```

12 Es gibt Ausnahmen, die wir hier aber nicht behandeln.

```
}
```

Jetzt soll noch ein gelber Rahmen dazu kommen. Hierfür müssen wir ganze drei Attribute setzen: *border-width* bestimmt, wie breit der Rahmen werden soll. Wir wollen nur einen minimal dünnen Rahmen von einem einzigen Pixel, setzen es also auf *2px*. Mit *border-color* legen wir die Farbe fest, *yellow*.

Schließlich müssen wir uns mit dem Attribut *border-style* entscheiden, ob wir unseren Rahmen gerne normal durchgezogen (*solid*), gestrichelt (*dashed*) oder gepunktet (*dotted*) hätten. Wir nehmen *solid*.

```
h1
{
  background-color: Coral;
  text-align: center;
  border-width: 2px;
  border-color: darkred;
  border-style: solid;
}
```

Leider sind Rahmen und Hintergrundfarbe jetzt um den kompletten Überschrift-Block, also über die gesamte Breite und in der Höhe auch komplett um das Bild. Wie wir das beheben, sehen wir gleich noch.

Übrigens: Dass wir hier eine eigene rote Hintergrundfarbe gesetzt haben, ist ein guter Moment, einen kurzen Blick auf die grundsätzliche Funktionsweise von CSS zu werfen. Schließlich hat unser ganzes Dokument – der *body* – doch eigentlich schon die Hintergrundfarbe beige. Diese überträgt automatisch auf alle Objekte, die im body drin sind: Also z.B. alle Absätze, in den Absätzen alle Links – und eben auch alle Überschriften.

Für dieses Verhalten steht das C in CSS: *Cascading*, „wasserfallartig". Ein für ein Objekt gesetztes Attribut rollt quasi wie ein Wasserfall auf alle darin enthaltenen Objekte weiter. Aber eben nur, so lange für das sich weiter innen befindende Objekt (in unserem Fall das *h1* im *body*) nichts eigenes festgelegt ist.

Standard-Attribute – und wie man sie los wird

In CSS können wir nicht nur eigene Design-Attribute (wie zum Beispiel unsere erwähnte beige Hintergrundfarbe) wieder ändern. Es gibt auch bestimmte Standardattribute, die immer da sind, noch bevor wir überhaupt ein Stylesheet geschrieben haben. Wenn wir die loswerden wollen, müssen wir das extra hinschreiben-

Jede HTML-Klasse hat ein bestimmtes Standard-Design, das vom Browser bestimmt wird. Deshalb sind, ganz ohne eigenes Stylesheet, standardmäßig zum Beispiel Überschriften groß und Links unterstrichen. Diese Standard-Design-Attribute gelten so lange, wie wir in unserem eigenen Stylesheet nichts anderes festlegen.

Nehmen wir an, wir finden unterstrichene Links eigentlich total doof. Wir wollen also eine Regel für die Klasse *a* schreiben, die dem ein Ende setzt.

Unterstrichen – *underline* – ist ein in CSS eine „Dekoration", ein möglicher Wert des Attributs *text-decoration*. Das Attribute *text-decoration* kann Text neben unterstrichen auch überstrichen (*overline*), durchgestrichen (*line-through*) oder blinkend (*blink*) machen. Wir wollen den ganzen Schickschnack nicht: Wir setzen es deshalb auf *none* – nix. Dann verschwindet die Unterstreichung.

```
a
{
      text-decoration: none;
}
```

3.2 Eigene Klassen fürs Design

Natürlich ist es viel zu unflexibel, immer nur *alle* Objekte einer Klasse gemeinsam designen zu wollen. In unserer Pizza-Seite soll der Absatz direkt unter der Überschrift („Benvenuto...“) zentriert, fett und mit größerer Schrift geschrieben sein. Wir können das aber nicht einfach als Design für die HTML-Klasse p setzen – denn es sollen ja nicht alle Absätze so aussehen, sondern nur dieser eine.

Das gleiche Problem haben wir mit dem allerletzten Absatz. Der ist eigentlich auch nur ein *p*-Objekt wie alle anderen, soll aber plötzlich einen Rahmen haben.

Um solche Probleme zu lösen, ist es möglich, eigene Klassen zu erschaffen. Wir wir in Abschnitt 2.6 gelernt haben, gehört ein Objekt immer einer bestimmten Klasse an: Die Klasse beschreibt, um was für eine Art von Objekt es sich handelt. Jedes HTML-Objekt gehört immer genau einer HTML-Klasse an, zum Beispiel jeder Absatz der Klasse *p*.

Zusätzlich kann jedes Objekt aber auch einer oder mehreren Klassen angehören, die wir uns selbst ausdenken. Der „Benvenuto“-Absatz unserer Pizzaseite soll zusätzlich zu seiner HTML-Klasse – *p* – auch einer von uns selbst erstellten Klasse *einleitung* angehören. Dafür müssen wir unsere HTML-Datei bearbeiten: Dem *p*-Objekt, das für den ersten Absatz steht, fügen wir ein Attribut *class* (englisch: „Klasse“) hinzu, das wir auf *einleitung* setzen:

```
<p class="einleitung">
```

Eigene Klassen können wir so viele erstellen, wie wir wollen, und sie nennen, wie wir wollen. Sobald wir einem HTML-Objekt mit dem Attribut *class* irgendeine eigene Klasse zuweisen, haben wir diese Klasse erschaffen. Einer Klasse können beliebig viele Objekte angehören. Wir könnten also noch beliebig viele weitere Objekte unserer Klasse *einleitung* hinzufügen, indem wir ihnen das Attribut *class="einleitung"* setzen. In diesem Beispiel soll der erste Absatz aber das einzige Objekt in der Klasse *einleitung* bleiben.

Wir sagten: Die Klasse beschreibt, um was für eine Art von Objekt es sich handelt. Die HTML-Klasse *p* beschreibt, dass unser Absatz-Objekt alles das hat, was ein Absatz nunmal so hat (er ist ein eigener Block auf der Seite, hat nach oben und unten Abstand zu anderen Objekten, und so weiter). Was beschreibt aber unsere eigene Klasse *einleitung*? Bis jetzt noch gar nichts – genau das legen wir jetzt nämlich im Stylesheet fest.

Um eine eigene Klasse im Stylesheet zu designen, erstellen wir wieder einen Regelblock. Dieser beginnt, wie wir das gewöhnt sind, wie dem Namen der Klasse – allerdings mit einem Punkt davor, um anzuzeigen, dass es sich nicht um eine HTML-Klasse handelt, sondern um eine von uns selbst erstellte. Das Grundgerüst des Regelblockes für unsere eigene Klasse *einleitung* sieht also so aus:

```
.einleitung
{
  /* Regel 1 */
  /* Regel 2 */
  /* ... */
}
```

Schriftgröße und Schriftgewicht

Um die Schriftgröße zu erhöhen, verändern wir das Attribut *font-size*. Für dieses Attribut können wir Werte in vielen verschiedenen Maßen angeben – von Pixeln über „Punkte“, wie wir das aus Textverarbeitungsprogrammen kennen, bis hin zu Zentimeterangaben ist alles möglich[13].

13 Ob absolute Angaben wie Zentimeter oder Punkt beim Benutzer richtig wiedergegeben werden, hängt von der korrekten Einstellung des DPI-Wertes (Dots per Inch) auf den verwendeten Monitor ab. Davon kann nicht unbedingt ausgegangen werden.

Sinnvoller ist eine Angabe in Prozent: Diese bezieht sich auf die im Browser des Benutzers eingestellte (das heißt vom Benutzer gewünschte!) Standardschriftgröße[14].

Um für die Einleitung eine 20% größere Schrift zu wählen, setzen wir das Attribut *font-size* auf 120%.

```
.einleitung
{
  font-size: 120%;
}
```

Die Eigenschaft „fett" ist ein sogenanntes Schriftgewicht (*font-weight*). Neben *bold* für fetten Text kann dieses Attribut auch die Werte *bolder* (extrafett), *normal* und *lighter* (dünnere Schrift als normal) haben[15].

```
.einleitung
{
  font-size: 120%;
  font-weight: bold;
}
```

Nach dem gleichen Prinzip können wir jetzt auch dem allerletzten Absatz der Pizza-Webseite seinen Rahmen verpassen. In meiner Version habe ich dafür die Klasse „fusszeile" erfunden.

3.3 Block- und Inline-Objekte

Kümmern wir uns noch einmal um unsere *h1*-Überschrift, „Pizzeria Fessa". Zwar haben wir ihr erfolgreich Rahmen und Hintergrundfarbe verpasst – trotzdem will das ganze einfach noch nicht so aussehen, wie im Screenshot in Abbildung 1. Warum geht der Rahmen über die ganze Breite und Höhe, und wie bekommen wir ihn kleiner?

In seinem *body* unterscheidet HTML grundsätzlich zwischen Block-Objekten und Inline-Objekten.

Ein Block-Objekt ist zum Beispiel eine Überschrift wie ein *h1*-Objekt oder ein Absatz (*p*-Objekt). Sie erzeugen einen eigenen Block auf der Webseite: Egal, hinter welchem anderen Objekt sie stehen, sie fangen immer mit etwas Abstand darunter am linken Seitenrand an . Und alles, was nach ihnen kommt und sich nicht innerhalb des Block-Objekts befindet, darf auch nicht in ihren Block rein, sondern kann wieder erst mit etwas Abstand anfangen, meistens wieder darunter am linken Seitenrand. Deshalb können Block-Objekte sich in der Regel auch nicht innerhalb von anderen Block-Objekten befinden.

Inline-Objekte sind dagegen zum Beispiel die *b*-Objekte oder die Links (*a*-Objekte). Sie bilden keinen eigenen Block auf der Seite sondern erscheinen immer genau da, wo wir sie hingesetzt haben. Sie können fast beliebig innerhalb anderer Objekte vorkommen. Innerhalb eines Absatzes (*p*-Block-Objekts) kann ein Link (*a*-Inline-Objekt) vorkommen, das wiederum Fettdruck (ein *b*-Inline-Objekt) enthalten kann).

Wir brauchen also ein Inline-Objekt für die Überschrift „Pizzeria Fessa", damit Hintergrundfarbe und Rahmen wirklich nur um den Inhalt gehen. Nun soll unsere Überschrift „Pizzeria Fessa" aber weder fett noch ein Link werden...

Textabschnitte

Wir wissen jetzt, wie wir sowohl die standardmäßig vorhandenen HTML-Klassen mit CSS um-designen können, als auch, wie wir mit eigenen Klassen das Design noch genauer festlegen können – wenn wir wollen, für jedes einzelne HTML-Objekt.

14 Oder auf eine für ein übergeordnetes Objekt festgelegte Schriftgröße.

15 Außerdem lässt sich das Schriftgewicht auch durch Zahlenwerte festlegen.

Manchmal reicht das aber nicht, wie eben im Beispiel unserer Überschrift „Pizzeria Fessa“.

Extra für solche Fälle – in denen man ein extra Inline-Objekt für Textabschnitte braucht, aber keines so wirklich passen will – gibt es die Klasse *span* (englisch „Spanne“, „Bereich“). *span*-Objekte bewirken an sich überhaupt nix. Sie sind nur dazu da, dass man sie mit CSS formatieren kann.

Wie allen anderen Objekte auch, können wir auch einem *span* eine zusätzliche, eigene Klasse zuweisen – in unserem Beispiel habe ich sie „mittig“ genannt.

Verpacken wir nun alles, was wir vorher direkt in der h1-Überschrift hatten, in das in ihr enthaltene span-Objekt, sieht der Rahmen schon fast richtig aus.

```
<h1>
  <span class="mittig">
    <img src="raffaele.png" height="170" alt="" />
    Pizzeria Fessa
  </span>
</h1>
```

Das mit dem Verpacken gilt natürlich sowohl für den HTML- als auch für den CSS-Teil! Das einzige CSS-Attribut, das unser *h1*-Objekt selber behalten darf, ist die Zentrierung – denn unser *span*-Objekt muss ja komplett innerhalb des Überschrift-Blocks zentriert werden (würden wir die Zentrierung auch ins *span*-Objekt verlagern, wären zwar Text und Bild theoretisch innerhalb des *span*-Objekts zentriert, aber das *span*-Objekt selber würde immernoch am linken Seitenrand kleben).

```
h1
{
  text-align: center;
}

.mittig
{
  border-style: solid;
  border-width: 1px;
  border-color: yellow;
  background-color: Coral;
}
```

Übrigens: Neben der „neutralen“ Inlineobjekt-Klasse span gibt es auch eine neutrale Blockobjekt-Klasse: *div* (für englisch division: „Abschnitt“, „Bereich“). *div*-Objekte gehören zu den wenigen, die wieder andere Block-Objekte enthalten dürfen.

3.4 Layout mit float

Layout ist ein wichtiger Teil des Designs und bedeutet festzulegen, wo sich bestimmte Objekte auf einer Seite befinden. Natürlich findet in HTML auch ohne, dass wir besonders etwas festlegen, jedes Objekt seinen Platz. Diese Standardwerte sind nur vielleicht nicht immer ganz so hübsch, wie wir das gerne hätten.

Auf unserer Pizza-Webseite wollen wir die einzelnen Absätze der Speisekarte dazu bringen, nebeneinander statt untereinander zu liegen.

Kaum haben wir Blockobjekte kennen gelernt, wollen wir also auch schon anfangen, ihnen ihre Macken abzugewöhnen. Zur Erinnerung: Standardmäßig sind alle Blockobjekte so breit wie die gesamte Seite und beginnen mit etwas Abstand unter dem Objekt vor ihnen, am linken Seitenrand.

Unsere Speisekarten-Absätze dagegen sollen nun nur so breit sein, wie es unbedingt sein muss. Zwar sollen sie prinzipiell immer noch links anfangen, aber erlauben, dass sich rechts von ihnen noch andere Blockobjekte breit machen können.

Kleine Vorarbeit dazu: Wir möchten die drei Absätze (*p*-Blockobjekte) der Speisekarte bearbeiten – und nur diese. Diese b ekommen deshalb die eigene Klasse „pizza“, was in HTML für die erste der drei Pizzen so aussieht:

```
<p class="pizza">
  <img src="mittelmeer.jpg" height="170" alt="Foto der Pizza Mittelmeer" />
  <br />
  Pizza Mittelmeer<br />
  <i>6,00 EUR</i>
</p>
```

Für diese Klasse können wir nun per CSS das Attribut *float* setzen. Dieses kann die Werte *left* und *right* annehmen. *left* macht genau, was wir wollen: Das Blockobjekt setzt sich immer noch so weit links wie möglich, lässt aber rechts von sich Platz für weitere Blockobjekte. Mit dem Wert *right* wäre das entsprechend umgekehrt.

Unser CSS-Regelblock für die eigene Klasse pizza sieht jetzt so aus. Nebenbei habe ich den Text innerhalb der einzelnen Pizza-Blöcke auch noch zentriert.

```
.pizza
{
  float: left;
  margin-right: 20px;
  text-align: center;
}
```

float wieder aufheben

Meistens will man gar nicht, dass sich nach einem ge-*float*-eten Block wirklich jedes andere Objekt in den freien Platz setzen darf. Im Beispiel unserer Pizza-Seite ist zum Beispiel auch der letzte Block der Speisekarte (Sad Pizza) links am *float*-en. Wenn das Browserfenster groß genug ist, würde also auch das darauffolgende Block-Objekt (die Fußzeile) sich den noch freien Platz schnappen und sich daneben setzen. Für eine Fußzeile ziemlich unpraktisch.

Deshalb kann per CSS auch festlegen, wo ein *float* wieder beendet werden soll – in unserem Fall genau an der Fußzeile. Dafür setzen wir für die Fußzeile das CSS-Attribut *clear*. Ein „*clear: left;*“ hebt ein „*float: left;*“ wieder auf. Weitere mögliche Attribut-Werte für *clear* sind *right* und *both*.

```
.fusszeile
{
  clear: both;
  text-align: center;
  border-style: solid;
  border-color: black;
  border-width: 1px;
}
```

3.5 Größen, Abstände und das Box-Modell

Wir sind schon fast fertig mit unserer Pizza-Webseite!Wenn du den aktuellen Stand aber mit dem Screenshot in Abbildung 1 vergleichst, stellst du fest, stellst du fest, dass so ein paar Kleinigkeiten nicht stimmen. Manche Objekte hängen irgendwie zu dicht aufeinander, manche sind ein bisschen zu weit voneinander weg.

Natürlich kann man mit CSS nicht nur die Objekte selbst, sondern auch die Abstände genau definieren.

Der eigentliche Abstand zwischen Objekten heißt *margin* (englisch „Rand“).

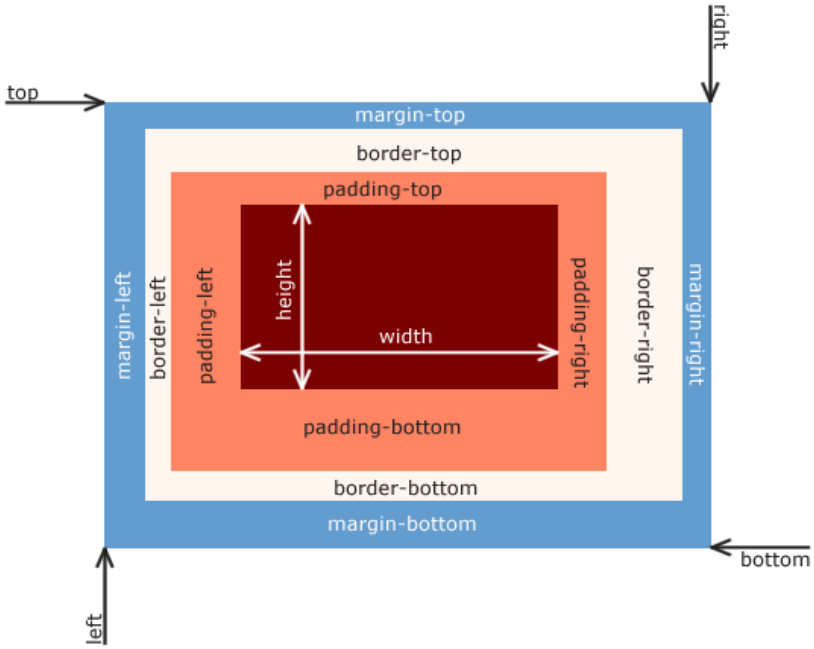

Abb. 6: Das Box-Modell von CSS
Bild: Matthias Apsel

Außerdem kennt CSS noch einen zweiten Abstand, nämlich den zwischen einem Objekt und seinem Rand: Das *padding* (englisch „Polster“).

Beide diese Abstände können wir in CSS setzen. Genau, wie wir es in Abschnitt 3.1 schon mit den Rahmen gemacht haben, können wir die Größe dieser Abstände in Pixel angeben. Das werden wir für unsere Pizza-Webseite so tun. Wie das aussieht, siehst du im fertigen CSS-Code in Beispiel 6. Padding, Rahmen und auch Margin lassen sich auch nur für einzelne Seiten eines Objekts setzen (z.B. mit *border-bottom* für einen Rahmen nur unter dem Objekt – eine Art grafische Unterstreichung).

CSS bietet neben Pixel auch noch viele andere Maßeinheiten – zum Beispiel Prozentangaben oder Maße abhängig von der Schriftgröße (u.A. *em*), die in vielen Fällen Sinn machen.

CSS erlaubt auch, für Objekte eine Größe (Breite mit dem Attribut *width*, Höhe mit dem Attribut *height*) angeben. Padding, Rahmen und Margin bilden dabei drei „Schalen“ um das eigentliche Objekt und werden für die Breite und Höhe des eigentlichen Objektes nicht mitgerechnet. Die mit *height* angegebene Höhe ist dabei nicht in Stein gemeißelt: Wenn der Inhalt eines Objekts nunmal nicht in die angegebene Höhe reinpasst, wächst das Objekt automatisch so viel wie nötig[16].

In CSS kann man sich Objekten also als Art „Boxen“ vorstellen, die eine bestimmte Größe, eine bestimmte Position und ggf. „Schalen“ haben. Dabei bilden auch Inline-Objekte immer eine eigene Box, nicht nur Blockobjekte.

Zum Testen kannst du mit folgender CSS-Regel alle Objekte, alle „Boxen“ auf einer Webseite sichtbar machen:

```
*
{
  border: 1px solid red;
}
```

Regeln für „*“ gelten für alle Objekte, egal welcher Klasse. Diese Regel gibt also jedem einzelnen Objekt einen roten Rahmen (außer, das Objekt hatte vorher schon einen eigenen Rahmen).

16 Verhindern lässt sich dieses Verhalten mit Hilfe des CSS-Attributs *overflow*.

Pizza-Webseite, HTML-Teil:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <link href="pizza.css" rel="stylesheet" />
    <title>Pizzeria Fessa</title>
  </head>

  <body>
    <h1>
      <span class="mittig">
        <img src="raffaele.png" height="170" alt="" />
        Pizzeria Fessa
      </span>
    </h1>


    <p class="einleitung">
      Benvenuto in der Pizzeria Fessa!<br />
      Hier bekommen Sie jede Pizza, nach der Sie anderswo vergeblich suchen.
    </p>
    <p>
      Unsere Pizzen sind original <a
href="http://de.wikipedia.org/wiki/Italien">
      italienische Spezialitäten</a>. Natürlich verwenden wir ausschließlich
      <a href="http://de.wikipedia.org/wiki/Mononatriumglutamat">traditionelle
      Zutaten</a>. Zumindest erzählen wir das unseren deutschen Kunden. Die
essen
      eh jeden Mist. Also werfen Sie einen Blick auf unsere Karte und lassen
auch
      Sie sich verzaubern!
    </p>
    <p>
      Unser Chefbäcker Raffaele und das ganze Team wünschen „buon appetito“!
    </p>

    <h2>Speisekarte</h2>
    <p class="pizza">
      <img src="mittelmeer.jpg" height="170" alt="Foto der Pizza Mittelmeer" />
      <br />
      Pizza Mittelmeer<br />
      <i>6,00 EUR</i>
    </p>

    <p class="pizza">
      <img src="pommes.jpg" height="170" alt="Foto der Pizza Pommes" /><br />
      Pizza Pommes<br />
      <i>5,00 EUR</i>
    </p>

    <p class="pizza">
      <img src="sadpizza.jpg" height="170" alt="Foto der Sad Pizza" /><br />
      Sad Pizza<br />
```

```
      <i>4,50 EUR</i>
    </p>

    <p class="fusszeile">
      Pizzeria Fessa, Pizzastraße 42, 31337 Pizzingen<br />
      Foto Mittelmeerpizza: <a
href="http://www.flickr.com/people/veganflower/">
      Mike & Molly</a>, restliche Pizzafotos:
      <a href="http://www.flickr.com/people/smiteme/">Kelly Garbato</a>
    </p>

  </body>
</html>
```

Pizza-Webseite, CSS-Teil:

```
body
{
  background-color: beige;
}

p
{
  color: DarkGreen;
}

h1
{
  text-align: center;
  margin-bottom: 30px;
}
```

```
h2
{
  margin-bottom: 0;
}


.mittig
{
  border-style: solid;
  border-width: 1px;
  border-color: yellow;
  padding-top: 10px;
  padding-bottom: 10px;
  padding-right: 30px;
  padding-left: 0;
  background-color: Coral;
}

.einleitung
{
  font-weight: bold;
  font-size: 120%;
  text-align: center;
}

a
{
  text-decoration: none;
}

.pizza
{
  float: left;
  margin-right: 20px;
  text-align: center;
}

.fusszeile
{
  clear: both;
  text-align: center;
  border-style: solid;
  border-color: black;
  border-width: 1px;
  padding: 5px;
}
```

Beispiel 6: Unsere fertige Pizza-Webseite

4 Mehr zu HTML und CSS

Wir wissen jetzt, wie wir Webseiten erstellen und mit Hilfe von Stylesheets verschönern. In diesem Kapitel – dem letzten über das Erstellen von Webseiten – gehen wir kurz noch auf ein paar Dinge, ein paar Klassen in HTML ein, die wir bisher ausgelassen haben.

4.1 Tabellen

Eine wichtige HTML-Klasse, die wir bisher ausgelassen haben, ist *table* zum Erstellen von Tabellen.

Wenn wir eine Tabelle erstellen, können wir uns entscheiden, ob zwischen den einzelnen Tabellenzellen Linien sein sollen. Wenn wir eine Tabelle mit Rändern haben wollen, sieht das öffnende HTML-Tag so aus:

```
<table border="1">
```

Wollen wir keine Rahmen um die Zellen, lassen wir das Attribut *border* entweder einfach weg oder lassen es leer:

```
<table border="">
```

Eine Tabelle besteht aus einzelnen Tabellenzeilen. Deshalb enthält auch ein *table*-Objekt in HTML mehrere *tr*-Objekte, die für eben diese Zeilen stehen (tr steht für englisch *table row*: „Tabellenzeile“).

Tabellen sind in HTML arme, missbrauchte Wesen: Lange Jahre wurde sie fürs Layout missbraucht, indem man einfach so tat, als sei die ganze Webseite eine große Tabelle mit Spalten für jeden Text, jedes Bild, jeden Rahmen, jeden Abstand – einfach alles.

Das war nicht gerade leicht zu erstellen und entsprach vor allem nicht der Idee des semantischen Webs – also HTML-Dokumente nach ihrem Inhalt aufzubauen und nicht danach, wie sie auf dem Bildschirm aussehen sollen. Eine Alternative, wie man mit CSS Layouts erstellen kann, haben wir im letzten Kapitel gesehen.

Innerhalb jeder Zeile besteht die Tabelle aus je einem *td*-Objekt für jede Spalte (td steht für englisch *table data*: „Tabelleninhalt“). Ausnahme: Wenn in einer Tabellenzelle nicht „normaler“ Inhalt steht, sondern eine Überschrift, dann benutzt man statt dem td- ein *th*-Objekt (th steht für englisch *table header*: „Tabellenüberschrift“).

Unsere Speisekarte auf der Pizza-Webseite ist ein Beispiel, in dem man durchaus auch eine Tabelle hätte nehmen können. Sinnvoll wäre zum Beispiel eine Tabelle mit mit zwei Spalten und vier Zeilen, bei der die oberste Zeile eine Überschrift bilder. Dann sähe das in HTML so aus:

```
<table border="1">
  <tr>
    <th>Pizza</th>
    <th>Preis</th>
  </tr>
  <tr>
    <td>Pizza Mittelmeer</td>
    <td>6,00 EUR</td>
  </tr>
  <tr>
    <td>Pizza Pommes</td>
    <td>5,00 EUR</td>
  </tr>
  <tr>
    <td>Sad Pizza</td>
    <td>4,50 EUR</td>
  </tr>
```

```
</table>
```

4.2 iFrames

iFrames (*frame* = englisch „Rahmen“, „Fassung“, i steht für *inline*, englisch für etwa „direkt mit drin“) sind eine ganz besondere Klasse von Objekten: Mit ihnen kann man andere Webseiten vollständig in die eigene einbinden.

In der alten HTML-Version 4 gab es neben den iFrames auch noch sogenannte „Framesets“ – eine formalere Art, eine Webseite aus anderen Webseiten aufzubauen. Mit HTML-Version 5 wurden diese für veraltet erklärt.

iFrames funktionieren ganz ähnlich wie Bilder: Ihre wichtigsten drei Attribute sind src (für die Adresse der Seite, die eingebunden werden soll) sowie height und width für die Größe der eingebundenen Seite.

iFrame-Objekte sind formal nicht inhaltsleer: Da sie anfangs von den Browsern nur schlecht unterstützt wurden, dürfen sie Text enthalten, der angezeigt wird, wenn der verwendete Browser keine iFrames darstellen kann. Da heutige Browser aber nahezu alle mit iFrames umgehen können, darf dieser Ersatztext weggelassen werden.

Ein iFrame, das Google auf einer Webseite einbindet, könnte zum Beispiel so erstellt werden:

```
<iframe src="http://google.com" width="1000" height="1000"></iframe>
```

4.3 Ein paar Stichworte zum Weiterlesen

In einem Kurs, der sich ganz allgemein mit dem Thema Internet beschäftigt, bleibt leider keine Zeit, um wirklich zu lernen, wie man ein komplettes Design mit CSS umsetzt. Deshalb möchte ich in diesem Abschnitt noch einige kurze Hinweise geben . Im Internet findest du mit den richtigen Stichpunkten dann schnell weitere Informationen.

Eine ziemlich gute Informationsquelle ist übrigens SelfHTML, zu finden unter http://de.selfhtml.org.

Positionierung

HTML-Objekte, wie wir sie erstellt haben, werden auf der Webseite immer in der Reihenfolge angezeigt, in der wir sie erstellt haben. Eigentlich ja auch logisch! Diese Art der Positionierung von Objekten auf einer Webseite beschreibt man gerne als „im Fluss“. Mit CSS können wir aber auch das ändern: mit dem Attribut *position*.

Setzen wir *position* auf *absolute*, hat die Stelle, an der wir das Objekt im Quelltext geschrieben haben, überhaupt keinen Einfluss mehr darauf, wo es auf der abgezeigten Webseite tatsächlich landet. Stattdessen können wir in CSS ganz genau angeben, wo das Objekt sein soll – zum Beispiel „400 Pixel vom oberen Seitenrand und 100 von links“. Hierzu dienen die CSS-Attribute *top*, *left*, *bottom* und *right*, mit denen wir jeweils den Abstand vom oberen, linken, unteren oder rechten Seitenrand angeben können. Meistens macht es Sinn, die Position an genau zwei Ecken festzumachen – eben zum Beispiel links und oben.

Absolute Positionierung wird gerne für Layouts verwendet. Es ist eine Alternative zum Layout mit *float*.

Absolut positionierte Objekte bekommen im „Fluss“ keinen eigenen Platz reserviert: Wenn an der Stelle, an der wir ein Objekt absolut positionieren eigentlich schon ein anderen Objekt ist, wird dieses andere Objekt einfach überdeckt.

Ganz ähnlich wie absolute funktioniert auch statische Positionierung (*static*). Hierbei bewegt sich das Objekt aber nicht mit, wenn der Benutzer auf der Webseite scrollt – es bleibt einfach stehen. Diese Positionierungsart wird gerne für Navigationsleisten und Ähnliches verwendet, die immer sichtbar sein sollen.

Neben absoluter Positionierung kann man das Attribut *position* auch auf *relative* setzen: Dann befindet sich ein Objekt grundsätzlich ganz normal im Fluss (und reserviert sich dort auch seinen ganz normalen Platz!), kann aber dem gegenüber mit den Attributen *top*, *left*, *bottom* und *right* verschoben werden. Bei relativer Positionierung verschiebt sich aber nur das Objekt selbst, nicht sein reservierter Platz: Im Zweifelsfall überdeckt das verschobene Objekt jetzt ein anderes und sein reservierter Platz bleibt einfach leer.

Style ohne Sheet

Wir haben in diesem Kurs immer brav ein Stylesheet geschrieben und ein Design für eine ganze Objektklasse festgelegt, wenn wir CSS benutzt haben. Es gibt aber auch eine „quick&dirty"-Lösung: Jedem HTML-Objekt können wir ein Attribut style verpassen. Als Wert setzen wir einfach CSS-Code. Das so bestimmte Design gilt genau für dieses eine Objekt.

Ein Beispiel: Der folgende Absatz hätte einen roten Rahmen – ohne, dass dies Auswirkungen auf irgendwelche anderen Absätze auf der Webseite hat.

```
<p style="border-color: red; border-style: solid; border-width: 1px;">...</p>
```

IDs und Rauten

Neben eigenen Klassen kann man HTML-Objekten auch eine eigene ID (Identifikation, also einen eindeutigen Namen) zuweisen. Der wesentliche Unterschied zu einer eigenen Klasse ist, dass es auf einer Webseite immer nur ein einziges Objekt mit der gleichen ID geben darf. Um einem HTML-Objekt eine ID zuzuweisen, können wir jedem Objekt einfach ein Attribut *id* geben:

```
<p id="dertollsteabsatz">...</p>
```

Um ein bestimmtes Objekt anhand seiner ID mit CSS zu designen, müssen wir den Regelblock mit dieser ID einleiten – wobei vor der ID selbst eine Raute (#) steht, um zu kennzeichnen, dass es sich um eine ID handelt.

```
#dertollsteabsatz
{
  border-color: red;
  border-style: solid;
  border-width: 1px;
}
```

Auch so kann ein einzelner Absatz also zum Beispiel einen roten Rahmen bekommen.

Designen geschachtelter Elemente

Diese „Einleitung" eines CSS-Regelblocks nennt man übrigens die *Selektion*: Diese Zeile bestimmt, welche Eigenschaften HTML-Objekte haben müssen, damit die in diesem Block formulierten Regeln für sie gelten.

Bis jetzt haben wir immer nur sehr einfach selektiert: Anhand der HTML-Klasse, anhand einer eigenen Design-Klasse oder anhand einer ID. Man kann aber auch in mehreren Stufen selektieren: Wie wäre es zum Beispiel mit einem roten Rahmen für alle Links, die sich in einer überschrift ersten Grades befinden?

```
h1 a
{
```

```
  border-color: red;
  border-style: solid;
  border-width: 1px;
}
```

Man kann also eine Art „Kette“ von Bedingungen aufstellen. Auch lassen sich HTML-Klassen und eigene Design-Klassen hierbei kombinieren. Ein Regelblock, der für alle p-Objekte der Klasse „supi“, aber nicht etwa für Überschrift-Objekte der Klasse „supi“ gilt, ließe sich so einleiten:

```
p.supi
```

5 Rechtliches

5.1 Urheberrecht

Das Urheber- und Datenschutzrecht ist ein kompliziertes Thema. Die Informationen in diesem Kapitel sind nur ein kurzer, vereinfachter Überblick und sicher nicht vollständig.

Wenn du einen Text, oder ein Webseite, oder ein Programm schreibst oder ein Bild malst, ein Foto schießt, ein Modell baust oder eine Skulptur meißelst – kurzum: wenn du ein „Werk" schaffst – dann gehört es dir. Niemand anderes darf es ohne deine Erlaubnis einfach nachmachen.

Das ist in Deutschland im Urheberrechtsgesetz[17] geregelt. „Urheber" nennt das Gesetz jeden, der ein Werk geschaffen hat. In Paragraph 15 des Gesetzes steht (Hervorhebung von mir):

(1) Der Urheber hat das **ausschließliche** Recht, sein Werk in körperlicher Form zu verwerten; das Recht umfaßt insbesondere
1. das **Vervielfältigungsrecht** (§ 16),
2. das Verbreitungsrecht (§ 17),
3. das Ausstellungsrecht (§ 18).

(2) Der Urheber hat ferner das **ausschließliche** Recht, sein Werk in unkörperlicher Form öffentlich wiederzugeben (Recht der öffentlichen Wiedergabe). Das Recht der öffentlichen Wiedergabe umfasst insbesondere
1. das Vortrags-, Aufführungs- und Vorführungsrecht (§ 19),
2. das Recht der **öffentlichen Zugänglichmachung** (§ 19a),
3. das Senderecht (§ 20),
4. das Recht der Wiedergabe durch Bild- oder Tonträger (§ 21),
5. das Recht der Wiedergabe von Funksendungen und von öffentlicher Zugänglichmachung (§ 22).

Im wesentlichen steht da das, was wir schon gesagt haben: Wer ein Werk geschaffen hat, hat bei quasi allem, was mit dem Werk geschieht, ein Wörtchen mitzureden. Niemand darf es ohne seine Erlaubnis kopieren („vervielfältigen") oder sonstwie veröffentlichen, auch nicht etwa im Fernsehen oder Ähnliches. Also auch nicht im Internet („öffentlich zugänglichmachen").

Auch verboten sind Bearbeitungen oder Änderungen eines Werkes. Wenn ich also zum Beispiel einen fremden Text erweitere, verbessere, übersetze oder umformuliere, ändert das nichts daran, dass es nicht mein Text ist. Ohne Erlaubnis darf ich auch meine geänderte Version nicht veröffentlichen.

Für uns heißt das, das wir auf unserer Webseite grundsätzlich erstmal nichts anbieten dürfen, was wir nicht komplett selbst erstellt haben. Das gilt für Texte, aber auch für Bilder, Videos und Ähnliches.

Lizenzen

Außer natürlich, wir haben eine Erlaubnis dafür! Eine solche Erlaubnis nennt man meistens Lizenz (von lateinisch *licere*: „erlauben"). Es gibt zwei verschiedene Arten von Lizenzen: Entweder die Erlaubnis gilt nur für dich und vielleicht auch nur für eine ganz bestimmte Webseite – oder sie gilt immer und für alle.

Lizenzen können „einfach so" gelten, oder aber mit bestimmten Bedingungen verbunden sein. Auh kann ein Autor Lizenzen kostenlos oder auch gegen Bezahlung vergeben. Letzteres – Lizenzen gegen Bezahlung – ist genau das, womit hauptberufliche Autoren und sonstige Urheber ihr Geld verdienen. Wenn du aber einen fremden Inhalt auf deine Webseite stellen willst, zum Beispiel ein Text oder ein Bild, kann es natür-

17 Online verfügbar unter: http://www.gesetze-im-internet.de/urhg/index.html

lich nicht schaden, dem Macher einfach mal eine Mail zu schreiben und um Erlaubnis zu fragen. Vielleicht sagt er einfach ja.

Wichtiger für und sind Inhalte, die frei verfügbar sind: Für die der Autor also mit einer für alle gültigen Lizenz erlaubt hat, sein Werk (vielleicht unter bestimmten Bedingungen) zu benutzen. Steht zum Beispiel auf einer Webseite, dass man mit den Texten und Bildern machen darf, was man will – oder auch, dass die Inhalte „gemeinfrei" oder „public domain" sind – dann ist das so eine freie Lizenz. Wichtig! Das gilt natürlich nur, wenn der Autor selber diese Erlaubnis erteilt hat. Hat der Betreiber der Webseite die Inhalte selber nur „geklaut", kann er auch keine Lizenz dafür erteilen.

Üblicher als so eine locker formulierte Erlaubnis sind im Internet freie Lizenzen mit bestimmten Bedingungen. Viele davon wurden speziell dafür formuliert und werden von Autoren immer wieder verwendet. Um eine bestimmte davon, geht es gleich.

Creative Commons

Creative Commons ist eine Organisation, die freie Inhalte fördern will. Dafür hat sie bestimmte Lizenzverträge formuliert, die im Internet sehr oft von Autoren verwendet werden, die ihre Inhalte frei verfügbar machen wollen. So steht zum Beispiel das Online-Lexikon Wikipedia unter einer Creative-Commons-Lizenz. Auf der Bilder-Webseite Flickr haben auch viele (aber nicht alle!) Benutzer ihre Bilder unter eine solche Creative-Commons-Lizenz gestellt und mit der Suchfunktion kann man speziell nach diesen freien Bildern suchen.

Bei freien Inhalten, die unter einer Creative-Commons-Lizenz stehen, muss man bestimmte Bedingungen beachten. Nicht alles ist erlaubt – und weil verschiedene Autoren unterschiedlich viel mit ihren Werken erlauben oder auch nicht erlauben wollen, gibt es sieben verschiedene Creative-Commons-Lizenzen.

http://creativecommons.org/licenses/by/3.0/de/

Die Creative-Commons-Lizenz „Namensnennung" (gerne kurz geschrieben als CC-BY) erlaubt dir im Prinzip, alles zu machen, was du willst. Du musst nur immer, wenn du das fremde Werk benutzt:

1. Dazuschreiben, von wem es ist – wenn der Autor angegeben hat, in welcher Art und weise er das gerne hätte, dann auch genau so. Wenn das sinnvoll möglich ist, muss du auch die Internetadresse angeben, unter der das Werk im Original zu finden ist.

2. Dazuschreiben, dass es unter der CC-BY lizensiert ist und einen Link zum Lizenztext anbieten.

http://creativecommons.org/licenses/by-sa/3.0/de/

Die Creative-Commons-Lizenz „Namensnennung – Weitergabe unter gleichen Bedingungen" (CC-BY-SA) funktioniert wie die CC-BY. Eine Einschränkungen gibt es nur, wenn du das Werk weiterbearbeitest. In diesem Fall musst du deine Bearbeitung auch wieder unter genau diese Lizenz stellen.

Eine solche Weiterbearbeitung wäre zum Beispiel, wenn du einen Text verbesserst, oder ein Bild in eine Collage einfügst.

Die CC-BY-SA sorgt also dafür, dass alles, was aus einem freien Werk gemacht wird, wieder genauso frei ist.

http://creativecommons.org/licenses/by-nd/3.0/de/

Die Creative-Commons-Lizenz „Namensnennung – Keine Bearbeitung“ (CC-BY-ND) schränkt dich noch ein Stück mehr ein: Du darfst das Werk zwar so, wie es ist, frei verwenden – verändern darfst du es aber nicht.

http://creativecommons.org/licenses/by-nc/3.0/de/

Die Creative-Commons-Lizenz „Namensnennung – nichtkommerziell“ (CC-BY-NC) erlaubt dir wie die CC-BY alles, auch Weiterbearbeitungen – aber nur, solange du damit kein Geld verdienst.

Wenn du also eine Webseite betreibst, auf der du Werbung verkaufst und damit Geld verdienst, darfst du solche Werke überhaupt nicht benutzen.

http://creativecommons.org/licenses/by-nc-sa/3.0/de/

Die Creative-Commons-Lizenz „Namensnennung – nichtkommerziell – Weitergabe unter gleichen Bedingungen“ (CC-BY-NC-SA) ist eine Kombination aus der CC-BY-SA und der CC-BY-NC.

Du darfst das Werk grundsätzlich nur verwenden, wenn du kein Geld damit verdienst. Wenn du es bearbeitest, musst du es wieder unter genau diese Lizenz stellen.

http://creativecommons.org/licenses/by-nc-nd/3.0/de/

Die Creative-Commons-Lizenz „Namensnennung – nichtkommerziell – keine Bearbeitung“ (CC-BY-NC-ND) ist eine Kombination aus der CC-BY-ND und der CC-BY-NC.

Du darfst das Werk grundsätzlich nur verwenden, wenn du kein Geld damit verdienst. Selbst dann darfst du es aber nur so, wie es ist, verwenden und nicht bearbeiten.

Creative Commons unterstützt auch komplett freie Inhalte, mit denen du wirklich tun und lassen kannst, was immer du willst. Im „CC-Sprech“ nennt man das gerne auch „CC0“.

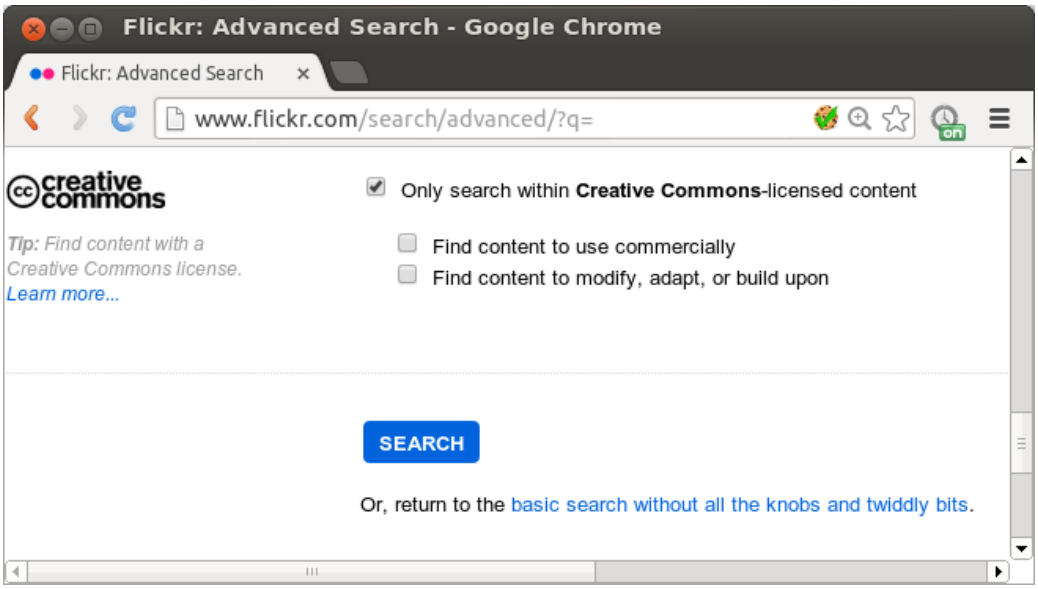

Abb. 7: Auf der Bilder-Webseite flickr kann man speziell nach freien Bildern suchen

Urheberrechtsverletzung

Benutzt du fremde Werke, ohne eine Lizenz dafür zu besitzen, ist das grundsätzlich illegal – eine Urheberrechtsverletzung. Dann droht dir von zwei Seiten Ärger: Zum Einen vom Urheber selbst, der verlangen kann, dass er nachträglich eine faire Bezahlung für die Verwendung bekommt (§ 97 Urheberrechtsgesetz). Zum Anderen auch vom Staat, denn Urheberrechtsverletzung ist eine Straftat. Für diese kann man zu einer Geldstrafe oder – in ausgedehnten Fällen, die über die einfache Verwendung auf einer kleinen, privaten Webseite deutlich hinausgehen – sogar mit Gefängnis bis zu drei Jahren bestraft werden.

Schranken des Urheberrechts

Eine Ausnahme zum oben Gesagten gibt es: Es gibt ein paar wenige, ganz bestimmte Umstände, in denen Teile eines urheberrechtlichen geschützten Werkes für bestimmte Zwecke verwendet werden dürfen, obwohl der Autor keine Lizenz erteilt hat. Diese Sonderregelungen bezeichnet man als „Schranken" des Urheberrechts, weil sie das grundsätzliche Recht des Urhebers, über die Verwendung seines Werkes selbst zu bestimmen, einschränken.

Diese Ausnahmen sind in den Paragraphen 42a bis 63a des Urheberrechtsgesetzes festgelegt. Viele dieser Ausnahmen sind für uns nicht wichtig, da sie sich nur auf ganz bestimmte Situationen beziehen – teilweise muss dabei auch trotz Ausnahme der Urheber für die Benutzung bezahlt werden.

Die für uns wichtigste Ausnahme ist das *Zitatrecht* (§ 51 Urheberrechtsgesetz). Wenn ich mich mit dem Inhalt eines Werkes ernsthaft beschäftigen möchte, kann ich das nur, wenn ich dafür das Werk selbst erstmal zeigen kann. Schreibe ich also zum Beispiel eine Interpretation oder eine Kritik über ein Buch oder ein Bild, darf ich die Textstelle oder das Bild dafür *zitieren*, also kopieren. Dabei darf ich aber nur so viel zitieren, wie unbedingt nötig ist, um mich ernsthaft mit dem Inhalt beschäftigen zu können – also zum Beispiel nur die wichtigsten Sätze aus einem Buch.

Wenn ich zitiere, muss ich immer genau angeben, von wem das Werk im Original kommt und wo ich es her habe (§ 63 Urheberrechtsgesetz). Das nennt man die *Quellenangabe*.

Ende des Urheberrechts

Übrigens: Das Urheberrecht gilt bis 70 Jahre nach dem Tod des Urhebers (§ 64 Urheberrechtsgesetz). Auch wenn ein Autor gestorben ist, dürfen seine Texte also immer noch 70 Jahre lang nicht einfach so kopiert werden.

5.2 Datenschutz

1983 hat das Bunderverfassungsgericht das *Grundrecht auf informationelle Selbstbestimmung* erfunden[18]. Dieses Recht besagt, das jeder grundsätzlich selbst entscheiden darf, wie seine persönlichen Daten verwendet. Es ist also meine eigene Entscheidung, wem ich wann, was und wie viel über mich selbst verrate.

> » *Freie Entfaltung der Persönlichkeit setzt unter den modernen Bedingungen der Datenverarbeitung den Schutz des Einzelnen gegen unbegrenzte Erhebung, Speicherung, Verwendung und Weitergabe seiner persönlichen Daten voraus. Dieser Schutz ist daher von dem Grundrecht des Art. 2 Abs. 1 in Verbindung mit Art. 1 Abs. 1 GG umfaßt. Das Grundrecht gewährleistet insoweit die Befugnis des Einzelnen, grundsätzlich selbst über die Preisgabe und Verwendung seiner persönlichen Daten zu bestimmen.* «
>
> – Bundesverfassungsgericht, „Volkszählungsurteil" (BVergGE 65,1)[19]

Solche Informationen über mich bezeichnet man dabei als *personenbezogene Daten*. Personenbezogene Daten sind alle Aussagen, die klar etwas über mich verraten – also meinen Namen nennen und etwas über

18 Die verfassungsmäßigen Grundrechte finden sich eigentlich in den Paragraphen 1 bis 19 des Grundgesetzes. 1983 hat das Bunderverfassungsgericht in seinem *Volkszählungsurteil* entschieden,

19 http://www.servat.unibe.ch/dfr/bv065001.html

mich verraten. Aber auch, wenn mein Name nicht direkt genannt wird, enthält eine Aussage personenbezogene Daten, wenn sich leicht herausfinden lässt, dass es um mich geht. Wikipedia[20] nennt als Beispiele:

- Augenfarbe: Klaus Mustermann hat blaue Augen.
- PKW: Erika Mustermann besitzt einen VW Golf.
- Geburtsort: Der erste Kanzler der Bundesrepublik Deutschland war gebürtiger Kölner.

Solche personenbezogenen Daten dürfen von einer Person, zum Beispiel dem Benutzer einer Website, nur gespeichert werden, wenn sie sich damit ausdrücklich einverstanden erklärt hat. Außerdem muss der Benutzer informiert werden, wofür die Daten gespeichert werden sollen. Auch, wenn der Benutzer sich also zum Beispiel bereiterklärt, für die Anmeldung in einem Forum seinen Namen und seine eMail-Adresse anzugeben, darf der Betreiber des Forums diese Daten also noch lange nicht für andere Zwecke als den Betrieb genau dieses Forums verwenden und schon gar nicht an Andere weitergeben. Hierfür müsste der Benutzer erneut einwilligen. Personenbezogene Daten dürfen, wann immer das möglich ist, auch nur von der jeweiligen Person direkt abgefragt werden und nicht etwa von Anderen abgefragt werden.

Diese Regelungen stehen in § 4 des Bundesdatenschutzgesetzes[21].

Noch dazu schreibt das Bundesdatenschutzgesetz in § 3 vor, überhaupt so wenig mit personenbezogenen Daten zu arbeiten, wie es nur geht. Das nennt man den Grundsatz der *Datensparsamkeit*.

> » *Die Erhebung, Verarbeitung und Nutzung personenbezogener Daten und die Auswahl und Gestaltung von Datenverarbeitungssystemen sind an dem Ziel auszurichten, so wenig personenbezogene Daten wie möglich zu erheben, zu verarbeiten oder zu nutzen. Insbesondere sind personenbezogene Daten zu anonymisieren oder zu pseudonymisieren, soweit dies nach dem Verwendungszweck möglich ist und keinen im Verhältnis zu dem angestrebten Schutzzweck unverhältnismäßigen Aufwand erfordert.* «

Ausnahmen von diesen Regeln gelten immer dann, wenn es ein Gesetz gibt, das in der jeweiligen Situation die Arbeit mit personenbezogenen Daten vorschreibt.

5.3 Impressumspflicht

Wer eine Website ins Internet stellt, wird in der Sprache des Gestzes zum „Anbieter“ eines „Telemediums“ (§ 2 Abs. 1 Telemediengesetz[22]). § 5 des Telemediengesetzes und § 55 des Rundfunkstaatsvertrags bestimmen, dass der Anbieter auf der Webseite seine Kontaktdaten angeben muss, wenn die Webseite gewisse Kriterien erfüllt. Diese Angabe nennt man meistens in Anlehnung an die entsprechenden Pflichtangaben in Zeitschriften „Impressum“[23].

Diese Daten, die auf jeder Website angegeben werden müssen, sind:

1. Der Name des Anbieters
2. Die vollständige Post-Adresse
3. Die eMail-Adresse
4. Eine weitere Kontaktmöglichkeit, z.B. die Telefonnummer oder ein Kontaktformular

Etwas unklar ist, wie diese Kriterien nun genau aussehen – was nun also konkret dazu führt, dass ich solche Daten über mich veröffentlichen muss (oder eben nicht). Das Telemediengesetz spricht hier von „geschäftsmäßige[n], in der Regel gegen Entgelt angebotene[n]“ (§ 5 Abs. 1) Diensten. Der Teufel steckt aber im Detail. „Geschäftsmäßig“ bedeutet nicht etwa, dass ich „ein Geschäft damit machen“ muss, sondern nur, dass ich die Seite auf längere Zeit und für die Öffentlichkeit ins Internet stelle – und nicht etwa nur mal kurz eine Datei online stelle, damit ein Freund sie runterladen kann. Die Juristensprache unterscheidet

20 http://de.wikipedia.org/wiki/Personenbezogene_Daten

21 http://www.gesetze-im-internet.de/bdsg_1990/BJNR029550990.html

22 http://www.gesetze-im-internet.de/tmg/BJNR017910007.html

23 Telemediengesetz und Rundfunkstaatsvertrag sprechen dagegen nur von „Informationspflichten“.

genau zwischen der Geschäftsmäßig (etwas, was ich länger und regelmäßig tue) und der Gewerblichkeit (etwas, womit ich Geld verdienen will).

„In der Regel gegen Entgelt angeboten“ bedeutet nicht, dass ich damit tatsächlich Geld verdienen muss – sondern nur, dass es sich um Inhalte handeln muss, bei denen es grundsätzlich üblich ist, dass man Geld mit ihnen verdient, und sei es nur durch Werbeschaltungen. Jede Seite, die theoretisch mit Bannerwerbung Geld verdienen könnte, fällt also vielleicht schon darunter. Wie die Formulierung „in der Regel gegen Entgelt angeboten“ tatsächlich zu verstehen ist, ist Interpretationssache – und im Zweifelsfall zählt die Interpretation eines Gerichtes.

Das Bundesministerium der Justiz empfiehlt daher, grundsätzlich ein Impressum anzubieten[24].

24 http://www.bmj.de/SharedDocs/Downloads/DE/pdfs/LeitfadenZurAnbieterkennzeichnung.pdf?__blob=publicationFile

Teil B.

Protokolle

Kommt noch... vielleicht xD