Deine Trainerin

Florence
MAURICE

- > Die richtige HTML-Struktur
- > So gestaltest du mit CSS
- > Designe Text, Tabellen, Bilder und Links
- > Entwirf Seitenlayouts
- > Gestalte Navigationsmenüs
- > Umschiffe Browserbugs
- > Bringe deine Website online

from zero 2 hero

HTML&CSS

Markt+Technik

Kapitel 3 Was für die Optik: Formatierungen mit CSS

Genügend oft hast du im letzten Kapitel den Satz gehört: „aber diese Formatierung lässt sich mit CSS ändern". Jetzt sehen wir uns an, wie das geht.

Du erfährst, wie CSS-Regeln funktionieren und wie du CSS-Angaben mit deinen HTML-Dateien verknüpfst. Danach lernst du wichtige CSS-Formatierungen kennen – von Farben über Schriftformatierungen wie Schriftart oder Schriftgröße, fett, kursiv und Kapitälchen bis hin zu den Absatzformatierung. Außerdem zeige ich dir viele nützliche Tools. Schließlich erfährst du, wie du Fehler im CSS-Code aufspürst, damit du für alles gerüstet bist.

HTML ist rein für die Strukturierung der Inhalte gedacht – nicht aber für ihre **Formatierung**, die realisiert man heute über CSS.

CSS steht für **Cascading Stylesheets**, also kaskadierende Stilvorlagen. Kaskadierend heißen die **Stilvorlagen**, da mehrere Angaben auf ein Element wirken können – das sehen wir uns später genauer an. Stilvorlagen ist hingegen schön konkret und darunter kann man sich mehr vorstellen: Denn du kannst über CSS nicht nur einzelne Elemente formatieren, sondern **Formatierungsvorlagen für ganze Webseiten** erstellen. Damit ist gesichert, dass alle Webseiten eines Projekts ein **einheitliches Layout** haben. Außerdem lassen sich die Formatierungen gut warten: Eine Änderung musst du nur an einer Stelle durchführen und sie wirkt sich auf alle Seiten des Projekts aus.

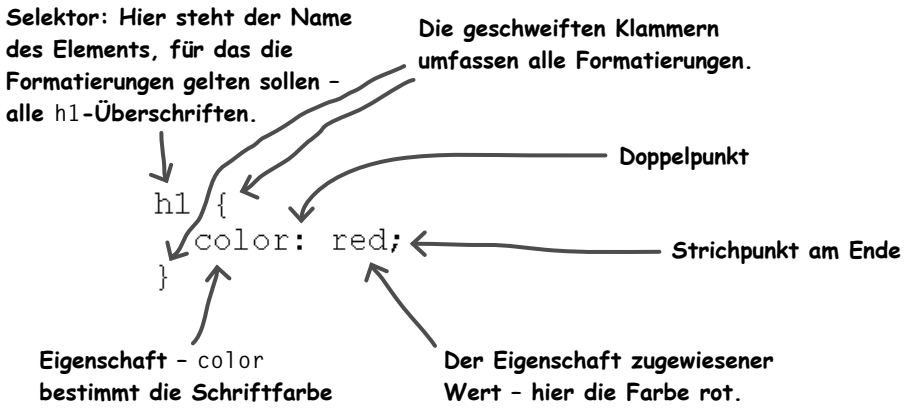
Genau wie HTML wird auch CSS vom **W3C** betreut. Aktuell ist gerade CSS in der Version 2.1l, aber an der nächsten Version – CSS 3 – wird schon fleißig gearbeitet.

So, genug der Theorie! Sehen wir uns an, wie CSS funktioniert und beginnen mit den CSS-Regeln.

AUFBAU VON CSS-REGELN

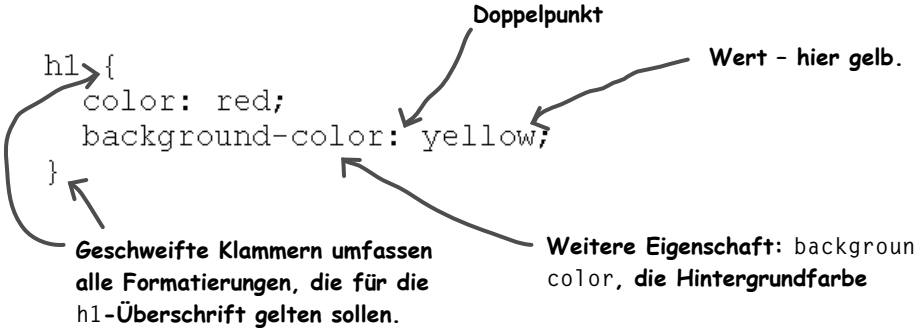
CSS besteht aus CSS-Regeln, die einem bestimmten Aufbau folgen:
Zuerst steht der Selektor. Er wählt (selektiert) die Elemente aus, für die die Formatierungen gelten sollen.

Die einzelnen Formatierungen stehen dann innerhalb der geschweiften Klammern.
Die Formatierungen werden über Eigenschaft-Wert-Paare definiert.



Zuerst steht die Eigenschaft – im Beispiel dient `color` zur Bestimmung der Textfarbe. Dahinter steht der Wert – im Beispiel `red`. Alle `h1`-Überschriften werden so rot eingefärbt.

Meist werden in den geschweiften Klammern mehrere Formatierungen festgelegt, wie das folgende Beispiel zeigt.



Die Einrückungen sind nicht unbedingt notwendig, sie sorgen dafür, dass es übersichtlicher aussieht.

Auch die Leerzeichen sind nicht notwendig: Du kannst vor und nach dem Doppelpunkt Leerzeichen machen oder sie auch weglassen.

CSS-ANGABEN IM DOKUMENTKOPF

Jetzt weißt du, wie eine CSS-Regel aufgebaut ist. Stellt sich aber noch die Frage, wo man sie hinschreibt. Da gibt es mehrere Möglichkeiten.

Die erste Möglichkeit: Du schreibst alle CSS-Regeln im `head`-Bereich des Dokuments. Sie stehen dann zwischen `<styletype="text/css">` und `</style>`.
Die dort angegebenen Formatierungen gelten für das ganze Dokument.

EIN DOKUMENT MIT CSS FORMATIEREN

Jetzt praktisch am Beispiel. Nimm hierfür am besten die Datei `ueberschriften.html` aus dem letzten Kapitel.
Innerhalb des `head`-Bereichs ergänzt du die Angabe:
`<style type="text/css">`

Dann folgen alle CSS-Befehle in der CSS-typischen Syntax, wie du sie eben gesehen hast. Beendet wird der CSS-Bereich durch das End-Tag von `style`.
Ergänze einmal die folgenden Zeilen im `head`-Bereich deines Dokuments.

`style`-**Start-Tag**

Sogenannter MIME-Typ, die Angabe lautet für CSS immer so.

```
<style type="text/css">
h1 {
  color: red;
  background-color: yellow;
}
h2 {
  color: blue;
  background-color: gray;
}
</style>
```

CSS-Regeln

`style`-**End-Tag**

MIME steht für Multipurpose Internet Mail Extensions und ist ein Standard zur Bezeichnung verschiedener Medientypen. MIME-Typ-Angaben bestehen immer aus zwei Teilen, dem Hauptmedientyp – im Beispiel `text` – und dem Untertyp – im Beispiel `css`. Eine Auflistung möglicher MIME-Typen findest du unter http://www.iana.org/assignments/media-types.

Wenn du dein Dokument im Browser aufrufst, sollte es jetzt so aussehen.

Die `h1`-Überschrift ist eingefärbt.

Und die `h2`-Überschriften sind ebenfalls eingefärbt.

`h1` und `h2` sind Blockelemente und diese werden so breit, wie sie werden können – hier so breit wie das Browserfenster.

Nicht schlecht, oder? Das Praktische ist, dass die Regeln dann für alle Elemente im Dokument gelten. Ergänzt du eine weitere `h1`-Überschrift oder eine weitere `h2`-Überschrift, so haben sie ebenfalls die angegebenen Formatierungen.
Wir werden uns nachher auch noch genauer ansehen, wie du Farben auswählen kannst, aber auf jeden Fall kannst du englische Farbbezeichnungen nehmen wie im Beispiel.

CSS-ANGABEN IN EINER EXTERNEN DATEI

CSS-Angaben im Kopfbereich sind ganz praktisch. Aber bei einem echten Webprojekt fährst du mit einer externen Datei besser. Das heißt, du speicherst die CSS-Angaben in einer eigenen Datei, auf die du in deinen HTML-Dokumenten verweist.
Und das geht so: Zuerst schneidest du CSS-Befehle aus und speicherst sie in einem eigenen Dokument.

```
<style type="text/css">
h1 {
  color: red;
  background-color: yellow;
}
h2 {
  color: blue;
  background-color: gray;
}
</style>
```

Du schneidest nur die CSS-Angaben aus – d.h. im Beispiel beginnen sie mit `h1 { ...`

Das kopierst du in eine leere Datei und speicherst diese unter dem Namen `formatierung.css`.

... bis hin zur letzten geschweiften Klammer `}`

Jetzt hast du also eine eigene Datei mit dem Namen `formatierung.css`, die die CSS-Befehle enthält. Die Endung `.css` ist übrigens wichtig, damit alles klappt.
Dann brauchst du noch einen Verweis in der HTML-Datei auf die CSS-Datei mit den Formatierungen. Dafür notierst du im `head`-Bereich:

```
<link rel="stylesheet"
type="text/css"
href="formatierung.css" />
```

Der Verweis auf die CSS-Datei über ein Element mit dem Namen `link`.

Die Angabe gehört in den `head`**-Bereich.**

Ein Element ohne Inhalt hat in XHTML immer einen Slash vor der schließenden spitzen Klammer.

```
<head>
<meta http-equiv="content-type" content="text/html; charset=ISO-8859-1" />
<title>Überschriften</title>

<link rel="stylesheet" type="text/css" href="formatierung.css" />

</head>
```

Den MIME-Typ kennst du schon. Er wird immer über `type="text/css"` **angegeben.**

`rel="stylesheet"` **bedeutet, dass das Dokument, auf das du verlinkst, ein Stylesheet ist.**

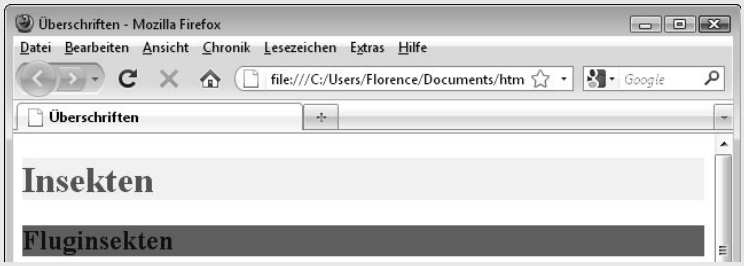
Hier – gaaanz wichtig – verrätst du dem Browser, wo er die Datei findet.

Name der CSS-Datei, im Beispiel ist es `formatierung.css`.

Den Code `<style type="text/css"> </style>` kannst du rauslöschen, den brauchen wir hier nicht mehr.

Das Beispiel funktioniert nur, wenn sich die Datei `formatierung.css` im selben Ordner befindet wie das HTML-Dokument.

Teste dann das Beispiel aus: Wenn du die HTML-Datei im Browser öffnest, sollte sie aussehen wie vorher – d.h. die Überschriften haben eine andere Text- und Hintergrundfarbe.



Wenn alles aussieht, wie vorher, hat die Auslagerung der CSS-Angaben geklappt.

Falls du das Dokument hingegen ohne Formatierung siehst, kontrolliere noch einmal:

1. Dass deine CSS-Datei wirklich nur den CSS-Code enthält. Es darf keine spitze Klammer drinnen stehen.
2. Dass du sie auch wirklich `formatierung.css` genannt hast.
3. Dass sich HTML- und CSS-Dokument im selben Ordner befinden.
4. Dass der Verweis in der HTML-Datei stimmt.

Der Verweis in der HTML-Datei muss stimmen.

Leicht vertut man sich bei den Anführungszeichen, die immer paarig sein müssen.

```
<link rel="stylesheet" type="text/css" href="formatierung.css" />
```

Und insbesondere beim Dateinamen muss alles stimmen.

Sehr gut – mit dem Auslagern deiner CSS-Anweisungen in eine eigene Datei hast du schon eine Profi-Technik kennengelernt. Denn bei echten Projekten arbeitet man immer mit externen Dateien. Für Experimente ist es hingegen oft praktisch, ein `style`-Element im Kopfbereich zu notieren, dann kannst du z.B. dein Dokument leicht unter verschiedenen Namen abspeichern. Am Schluss solltest du die Angaben aber immer in einer externen Datei abspeichern.

EXKURS: EXTERNE DATEI ÜBER @IMPORT EINBINDEN

Um externe CSS-Dateien einzubinden, gibt es noch eine Alternative – die `@import`-Regel.
Diese kann dort stehen, wo CSS-Regeln stehen können. Beispielsweise in einem internen Stylesheet. Sie beginnt mit `@import`, dann folgt das Schlüsselwort `url` und in runden Klammern die Datei, die eingebunden werden soll.

Jetzt wird die externe Datei `formatierung.css` über `@import` eingebunden.

```
<style type="text/css">
  @import url(formatierung.css);

</style>
```

Danach könnten weitere CSS-Regeln folgen, aber vor der `@import`-Anweisung darf keine andere CSS-Regel stehen.

Warum gibt es zwei verschiedene Wege, externe CSS-Dateien einzubinden? Gute Frage! Die Variante über `@import` kann ja selbst auch wiederum in einem externen Stylesheet stehen. Und dann kann sie bei sehr großen Projekten helfen, die Formatierungen auf unterschiedliche Stylesheets zu verteilen und immer nur die gewünschten einzubauen.

Die unterschiedlichen CSS-Dateien werden über `@import` eingebunden.

Hier siehst du einen Ausschnitt aus einer CSS-Datei des YAML-CSS-Frameworks.

```
/* import core styles | Basis-Stylesheets einbinden */
@import url(../../../yaml/core/base.css);

/* import screen layout | Screen-Layout einbinden */
@import url(../../../yaml/navigation/nav_shinybuttons.css);

@import url(screen/basemod.css);
@import url(screen/content.css);
```

`@import` kann dabei helfen, umfassenden CSS-Code auf mehrere externe CSS-Dateien zu verteilen – und damit übersichtlich zu machen.

Du solltest dir merken, dass es noch die Möglichkeit über `@import` gibt, damit du nicht erstaunt bist, wenn du das in einem Beispiel siehst. Für den Anfang und in den meisten Fällen auch für später wird es aber genügen, wenn du selbst dein externes Stylesheet über `link` einbindest.

FORMATIERUNGEN ÜBER DAS STYLE-ATTRIBUT

Es gibt noch eine dritte Art, CSS-Angaben zu notieren: nämlich hinter einem `style`-Attribut im Element, für das die Formatierung gelten soll. Dann gilt die entsprechende Formatierung nur für das Element, in dem das `style`-Attribut steht.

Jetzt wird ein Attribut mit Namen `style` eingesetzt.

Dahinter stehen die CSS-Angaben in Anführungszeichen.

```
<h3 style="color: yellow; background-color: orange">Ohrwurm </h3>
```

Es folgen direkt Eigenschaft-Wert-Paare, getrennt durch Strichpunkte.

Es gibt keinen Selektor, da die Formatierung automatisch für das aktuelle Element gelten.

Am Schluss kannst du den Strichpunkt auch weglassen.

Und was hältst du von dieser Methode, CSS-Formatierungen durchzuführen? Für schnelle Experimente ist sie durchaus praktisch. Aber bei echten Projekten solltest du sie nur im Ausnahmefall wählen.

Das `style`-Attribut wirkt sich nur auf das Element aus, indem es steht: Hier sind die Farben geändert.

Von den bekannten 1800 Arten leben in Deutschland nur acht.

Eine andere `h3`-Überschrift ist davon nicht betroffen.

Flöhe

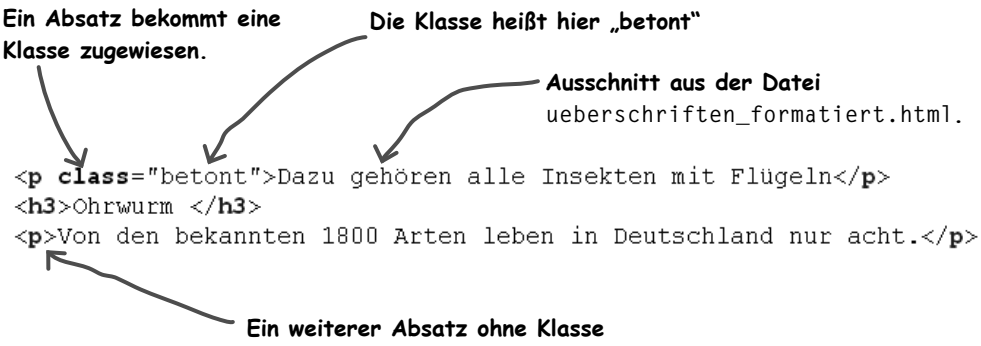
gehören auch zu den Fluginsekten, obwohl sie ihre Flügel verloren haben

Denn diese Methode ist eigentlich unpraktisch: Durch diese Angaben vermischt du Inhalt und Layout. Und das heißt: Es ist beispielsweise mühsam, Änderungen durchzuführen, weil du alle HTML-Dateien durchsuchen musst, ob dort die entsprechende Formatierung auftaucht oder nicht.

KLASSEN ALS SELEKTOREN

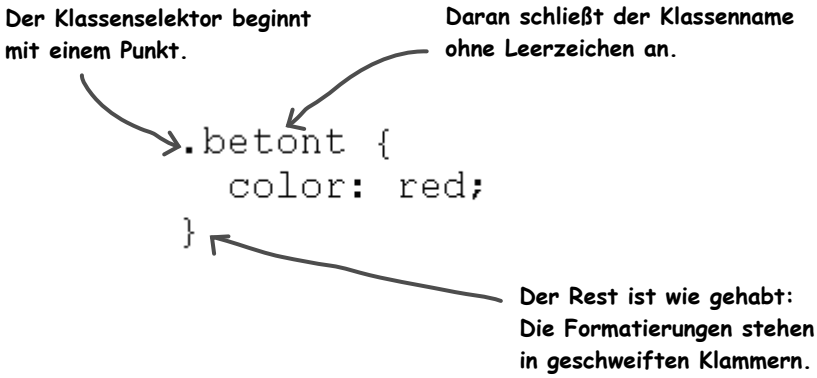
Aber es muss doch eine Möglichkeit geben, einzelne Elemente anders zu formatieren, weil ein Absatz beispielsweise anders aussehen soll als die anderen? Das gibt es: über die sogenannten Klassen.

Zuerst weist du dem Element, das anders sein soll, das Attribut `class` mit deinem selbstgewählten Klassennamen zu, im Beispiel `class="betont"`:



Den Namen für die Klasse kannst du selbst wählen – aber nimm keine Sonderzeichen oder Leerzeichen.

Dann musst du nur noch die Formatierungen in deinem CSS-Teil vornehmen. Als Selektor schreibst du einen Punkt und dann den Namen der Klasse:



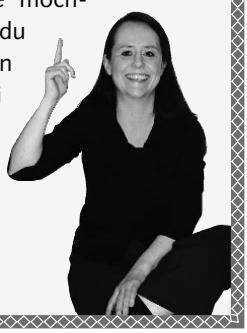
Diese Klasse kannst du jetzt beliebigen Elemente zuweisen, die die entsprechende Formatierung haben sollen.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="content-type" content="text/html; charset=ISO-8859-1" />
<title>Überschriften</title>
<style type="text/css">
h1 {
  color: red;
  background-color: yellow;
}
h2 {
  color: blue;
  background-color: gray;
}
.betont {
  color: red;
}
</style>
</head>
<body>
<h1>Insekten</h1>
<h2>Fluginsekten</h2>
<p class="betont">Dazu gehören alle Insekten mit Flügeln</p>
<h3>Ohrwurm </h3>
<p>Von den bekannten 1800 Arten leben in Deutschland nur acht.</p>
<h3>Flöhe</h3>
<p>gehören auch zu den Fluginsekten, obwohl sie ihre Flügel verloren haben</p>
<h2 class="betont">Fischchen</h2>
</body>
</html>
```

Wie die Klasse aussehen soll, steht im CSS-Teil.

Diese Klasse kann dann beliebigen Elementen zugewiesen werden.

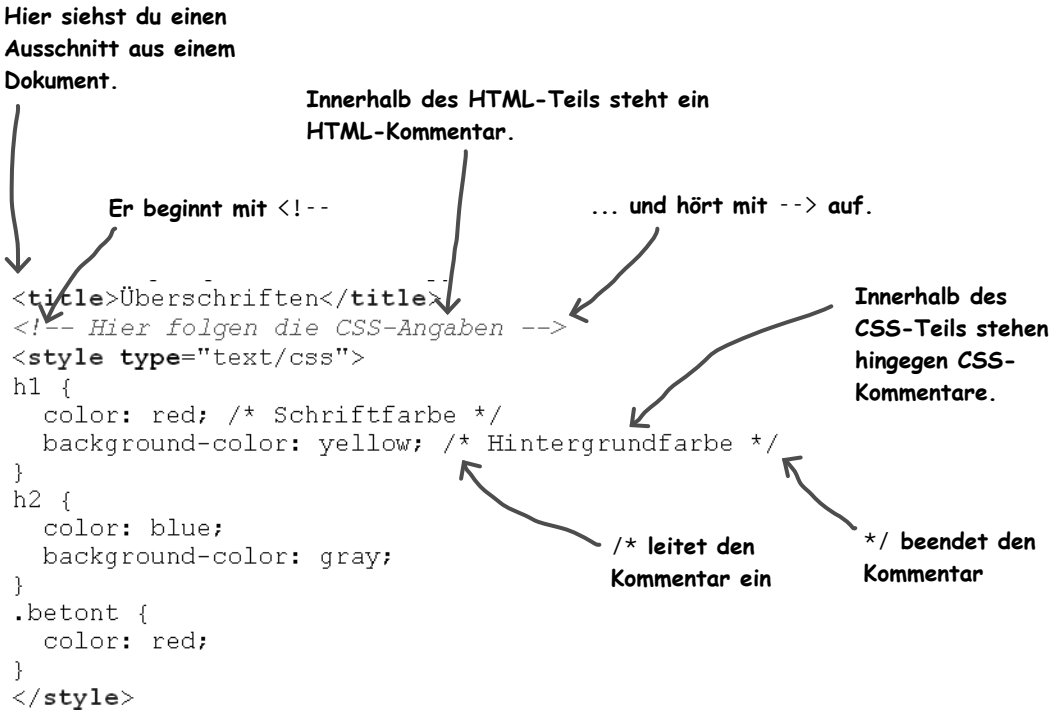
Das Ganze funktioniert selbstverständlich auch, wenn du die Definition in einer externen CSS-Datei durchführst. Klassenselektoren sind äußerst praktisch, denn so bleibt die Trennung von Inhalt und Layout erhalten: Falls du eine andere Farbe möchtest, so nimmst du die Änderung nur in deiner CSS-Datei vor und musst nichts in deiner HTML-Datei ändern.



KOMMENTARE DEFINIEREN

Genauso wie Kommentare in HTML-Dokumenten wichtig sind, um den Überblick bei komplexen Dokumenten zu bewahren, sind Kommentare in längeren Stylesheets hilfreich.

Die Syntax von CSS-Kommentaren ist allerdings anders als die Syntax von HTML-Kommentaren. Sie beginnen mit `/*` und werden mit `*/` beendet.




```
<title>Überschriften</title>
<!-- Hier folgen die CSS-Angaben -->
<style type="text/css">
h1 {
  color: red; /* Schriftfarbe */
  background-color: yellow; /* Hintergrundfarbe */
}
h2 {
  color: blue;
  background-color: gray;
}
.betont {
  color: red;
}
</style>
```

Du musst also immer aufpassen, ob du im CSS-Bereich bist oder im HTML-Teil, und dann den richtigen Kommentar wählen. CSS-Kommentare können natürlich auch über mehrere Zeilen gehen.

Man kann Kommentare auch um CSS-Anweisungen legen, um sie testweise auszuschalten. Das nennt man dann „Auskommentieren“.

FARBEN DEFINIEREN

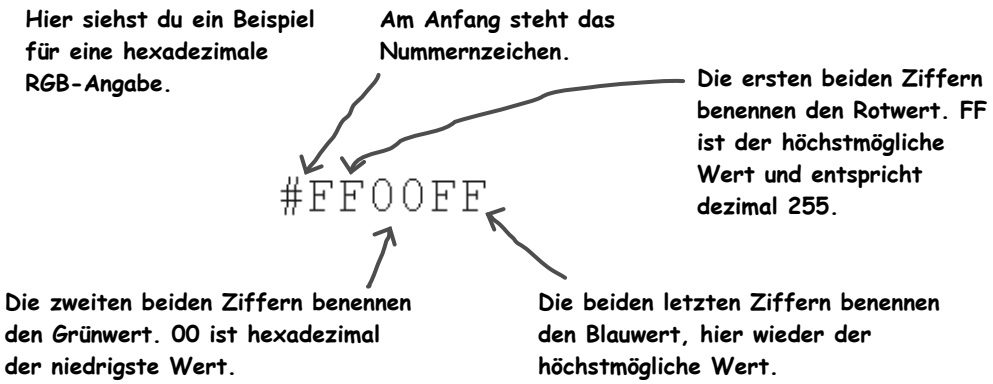
Damit kennst du die wichtigsten CSS-Grundlagen, so dass wir uns jetzt ansehen können, was man denn so alles formatieren kann. Und wir beginnen damit, wie man in CSS Farben angibt.

Eine Art hast du bereits angewandt: die englischen Farbnamen. Offiziell abgesegnet sind nur eine Handvoll von Farbnamen, die du in der Tabelle siehst:

black – schwarz	green – grün	navy – dunkelblau	gray – grau
lime – hellgrün	blue – blau	maroon – dunkelrot	olive – olivgrün
purple – violett	red – rot	yellow – gelb	fuchsia – magenta
silver – hellgrau	aqua – cyan	teal – blaugrün	white – weiß

Eine größere Auswahl von Farben hast du über die RGB-Angaben, die üblicherweise in hexadezimaler Form erfolgt.
RGB heißt das Farbschema, weil alle Farben durch die Farben Rot-Grün-Blau gemischt werden.

Hexadezimal bedeutet, dass das Zahlsystem auf 16, statt üblicherweise auf 10 basiert. Es stehen dir dann neben den Ziffern 0-9 noch die Buchstaben A-F zur Verfügung. F ist das höchste und entspricht 15.



Und welche Farbe ist das? Pink. Leider ist das bei diesem System nicht unbedingt zu erraten. Klar ist nur: Die Farbe #FFFFFF bedeutet die höchste Intensität aller drei beteiligten Farben – und das ist Weiß.

Das ist anders als im Malkasten, weil am Bildschirm Licht gemischt wird. #000000 bedeutet keine Farbe und deswegen Schwarz.

FARBEN WÄHLEN

Du brauchst also Tools, um den hexadezimalen RGB-Wert der gewünschten Farbe zu ermitteln. Wenn du ein Bildbearbeitungsprogramm benutzt, so sollte das eine entsprechende Option bieten.

Ansonsten gibt's natürlich Tools im Internet. Eines meiner Lieblingstools findest du unter http://colorschemedesigner.com/. Es zeigt dir gleichzeitig noch schöne Farbkombinationen.

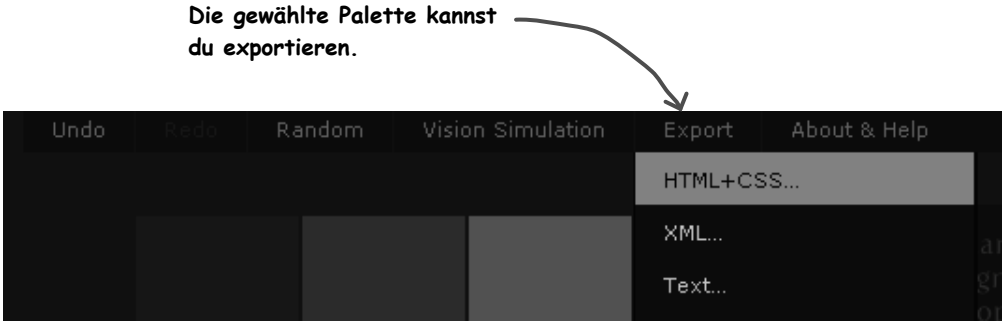
Hier kannst du eine Farbe durch Klick auswählen.

Den Hexadezimalwert deiner Farbe siehst du hier.

Über diese Tabs kannst du den Farbwert noch anpassen, wenn du beispielsweise hellere Töne willst.

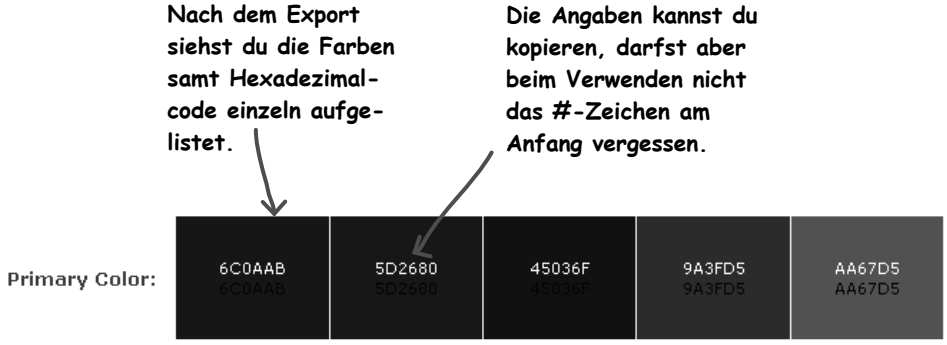
Und hier siehst du mögliche harmonische Kombinationen.

Wenn du oben im Farbwähler „Export“ anklickst und dann „HTML+CSS“ wählst, so werden die gewählten Farben in einer eigenen Datei angezeigt. Du kannst dir den Hexadezimalcode auch einfach herauskopieren.



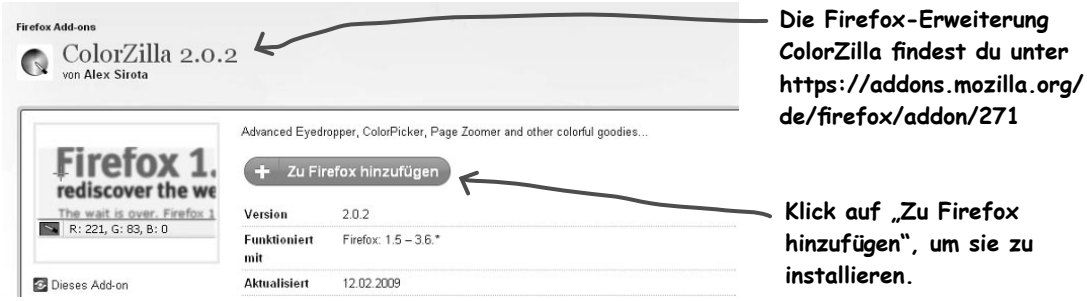
Deine Farben kannst du in gewohnter Weise einsetzen, also beispielsweise als Hintergrundfarbe oder Textfarbe eines Elements:

```
h1 {
color: #AA67D5;
background-color: #45036F;
}
```



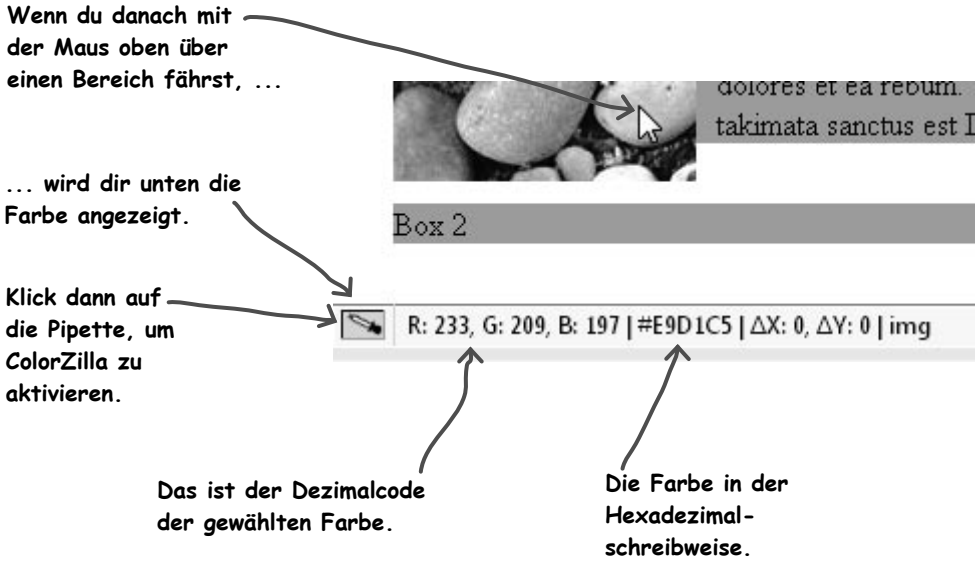
Wichtig ist, dass du daran denkst, dass diese Angaben immer mit einem Nummernzeichen # beginnen müssen.

Das gerade vorgestellte Tool ist wunderbar, um schöne Farbkombinationen zu wählen. Was aber, wenn du schon weißt, wie deine Farbe aussieht, weil sie beispielsweise in einem Bild vorkommt, aber du weißt nicht ihren Code? Dann hilft eine kleine Erweiterung für den Firefox mit dem Namen Colorzilla.



Du installierst die Erweiterung durch „Zu Firefox hinzufügen" und startest dann Firefox neu.

Damit du ColorZilla nutzen kannst, musst du dir die Statusleiste einblenden über „Ansicht/Statusleiste".

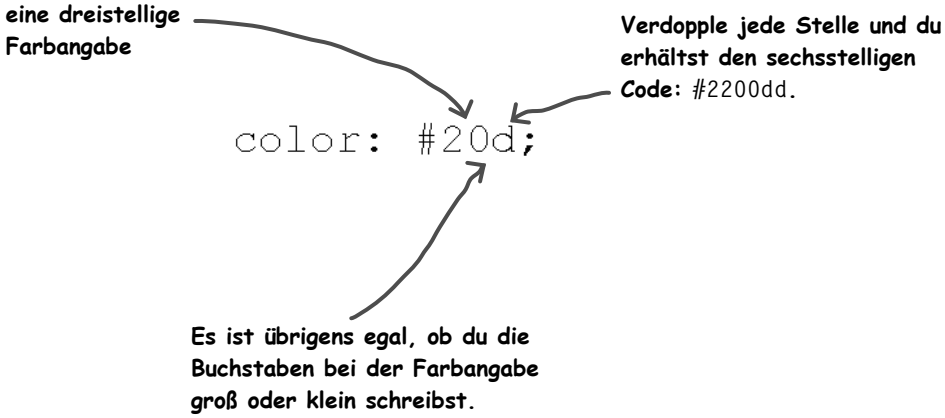


Am Beispiel einer anderen Firefox-Erweiterung, der Webdeveloper Toolbar, siehst du ein bisschen weiter hinten im Kapitel noch einmal genau, wie das mit den Firefox-Erweiterungen geht.

VARIANTEN DER FARBANGABE

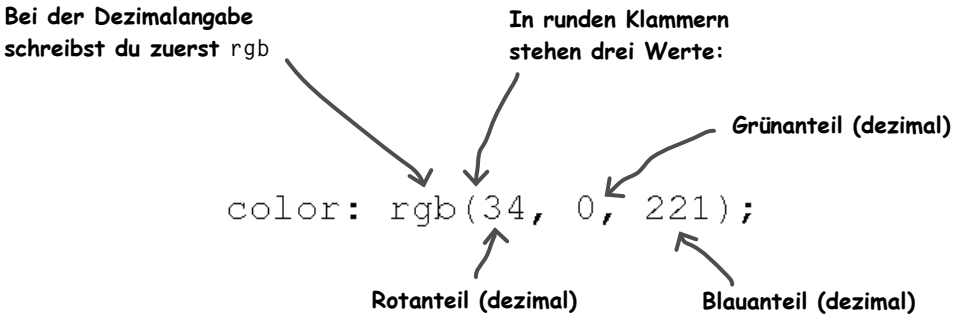
Die Methode, in CSS Farben über die Hexadezimalwerte anzugeben, ist die am häufigsten verwendete Art. Für diese Schreibweise gibt es noch eine Abkürzung, die dreistelligen Farbangaben.

Diese sind immer möglich, wenn bei allen drei Farben die Ziffern identisch sind. Du löst die dreistelligen Farbangaben auf, indem du jede Stelle verdoppeltest:



Diese Angabe ist also identisch mit `color: #2200dd;` aber kürzer. Gerade für Schwarz und Weiß ist es einfacher `#000` für Schwarz zu schreiben und `#fff` für Weiß, als die 6 Stellen, bei denen man sich rasch verzählt.

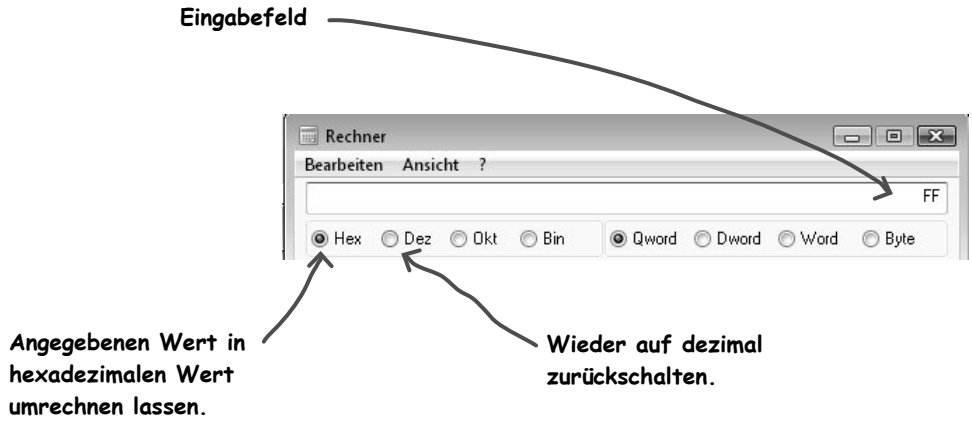
Außerdem kannst du – was aber seltener benutzt wird – Farben auch dezimal angeben:



DEZIMAL IN HEXADEZIMAL UMRECHNEN UND ZURÜCK

Um Hexadezimal- in Dezimalwerte umzurechnen, kannst du beispielsweise den Windows-Taschenrechner nehmen. Du findest ihn unter „Programme“ und „Zubehör“. Schalt im Taschenrechner auf „Wissenschaftlich“ um.

Wenn du eine Dezimalzahl in den hexadezimalen Wert umrechnen möchtest, wählst du unterhalb des Eingabefelds „Dez“ aus. Dann gibst du die Zahl ein, die du umrechnen möchtest und wählst unterhalb des Eingabefelds „Hex“. Jetzt erhältst du den umgerechneten Wert.



Ansonsten findest du auch online Tools, die diese Umrechnung für dich durchführen – z.B. den Umrechner von Selfhtml unter http://de.selfhtml.org/helferlein/dezhex.htm.

SCHRIFTART FESTLEGEN

Jetzt wollen wir uns genauer ansehen, was sich alles formatieren lässt und beginnen mit der Schriftformatierung. Zuerst einmal sollten wir die Schriftart ändern.

Dazu gibt es die Eigenschaft `font-family`. Hier gibst du eine Liste von Schriften an.

`font-family` **steht für „Schriftfamilie", also Schriftart.**

Dahinter steht eine Liste von Schriften, die durch Komma getrennt werden.

Am Schluss solltest du immer eine generische Schriftart bestimmen, im Beispiel `sans-serif`.

```
font-family: Arial, Helvetica, sans-serif;
```

Der Browser des Besuchers versucht dann die ersten Schrift der Liste zu nehmen. Ist diese nicht vorhanden, so versucht er die zweite, ist auch diese nicht vorhanden, so würde er die dritte Schrift der Liste nehmen. Als dritte Schriftart ist im Beispiel eine generische Schriftart angegeben. Der Browser soll, wenn er die beiden ersten Schriften nicht hat, seine eigene serifenlose Standardschrift nehmen.

Serifenlose Schrift

Schrift mit Serifen

Insekten Insekten

Serifen sind die kleinen Verzierungen an den Buchstaben.

Im Allgemeinen sagt man, dass serifenlose Schriften am Bildschirm besser lesbar sind. Deswegen wird für den Fließtext gerne eine serifenlose Schrift benutzt. Bei Überschriften kann dann auch gezielt eine Schrift mit Serifen benutzt werden.

SCHRIFTART DEFINIEREN

Nehmen wir das Beispiel mit den Überschriften und definieren hier Schriftarten. Die allgemein verwendete Schrift für alle Elemente soll Arial, Helvetica oder eine andere serifenlose Schrift sein. Da das für alle Elemente gelten soll, definieren wir es für `body`. `body` ist ja das Element, das alle Elemente umfasst, die dann im Browser dargestellt werden.

Wählt man `body` als Selektor, so gilt die Formatierung für alle Elemente, die sich innerhalb von `body` befinden.

```
body {
  font-family: Arial, Helvetica, sans-serif;
}
```

Wenn wir die Schriftart für `body` definieren, wirkt das auf alle Elemente, die innerhalb von `<body>` und `</body>` stehen. Man sagt dann in CSS auch: Die Schriftart wird an die ungeordneten Elemente vererbt.

Dann können wir noch für die `h1`-Überschrift eine Schrift mit Serifen festlegen. Wieder setzen wir eine Liste von Schriften ein.

Die `h1`-Überschrift soll in einer Serifenschrift sein.

Schriftnamen, die aus mehreren Wörtern bestehen, müssen in Anführungszeichen stehen.

generische Schriftart – „serif".

```
h1 {
  font-family: "Times New Roman", Times, serif;
  color: red;
  background-color: yellow;
}
```

Wenn weder „Times New Roman" noch „Times" auf dem Computer des Besuchers vorhanden sind, wird die Standard-Serifenschrift des Browsers genommen. Diese beiden Angaben ergänzt du im CSS-Teil des Dokuments.

Und so sieht das Dokument jetzt aus.

Die Überschrift hat eine Serifenschrift.

Alles andere wird in serifenloser Schrift dargestellt – `body` als Selektor hat das bewirkt.

Möchtest du eine Hintergrundfarbe und eine Schriftfarbe für das ganze Dokument festlegen, so solltest du das auch bei `body` machen:

```
body {
font-family: sans-serif;
background-color: black;
color: white;
}
```

Prima – die Festlegung der Schrift ist eine wichtige Design-Entscheidung!

MASSEINHEITEN IN CSS

Als Nächstes wollen wir die Schriftgröße ändern. Und dafür musst du wissen, was für Maßeinheiten dir in CSS zur Verfügung stehen. Auf den ersten Blick hast du eine üppige Auswahl: Das reicht von Zentimetern, Millimetern bis hin zu auf den ersten Blick kryptischen Einheiten wie `em` und `ex`.

Aber nicht alle Einheiten sind gleich gut für die Ausgabe am Bildschirm geeignet und deswegen reduzieren sich die Möglichkeiten gleich.

Mögliche Maßeinheiten in CSS

Abkürzung	Bedeutung
cm	Zentimeter
mm	Millimeter
in	Zoll/Inch. Ein Zoll entspricht 2,54 Zentimetern.
pt	Punkt. Ein Punkt ist 1/72 Zoll, d.h. ca. 0,35 mm.
pc	Pica. Ein Pica entspricht 12 Punkt, ergibt also 0,42 cm.
em	Höhe des großen M. orientiert sich an der aktuell gewählten Schriftgröße.
ex	Die x-Höhe des entsprechenden Fonts. meist ungefähr so groß wie die Höhe des x
px	Pixel sind relativ zur Auflösung des Ausgabegeräts
%	Prozentangaben

relevant für den Bildschirm

DA WAREN ES NUR NOCH DREI: EM, PX UND %

Für die Ausgabe am Bildschirm geeignet sind nur em, px und %. Sehen wir diese mal genauer an. Erst mal zu `em`. Das bezieht sich auf die aktuelle Schriftgröße für das Element.

Ein kleines Beispiel hierzu – du siehst dabei auch, dass du über die Eigenschaft `font-size` die Schriftgröße bestimmst.

Die Schriftgröße für `body` wird auf 20px gesetzt.

```
body {
  font-size: 20px;
}
h1 {
  font-size: 2.5em;
}
h2 {
  font-size: 1.2em;
}
```

Die Schriftgröße der erste Überschrift soll 2.5em sein. Damit ist die Überschrift 2.5 * 20px groß – macht 50px.

Die zweite Überschrift ist 1.2 * 20px groß – macht also 24px.

Achtung: Bei allen Einheiten schreibst du Wert und Einheit direkt aneinander – ohne Leerzeichen dazwischen.

Du siehst, `em` sind ganz relativ und sind eigentlich so etwas wie ein Faktor, der als Ausgangsbasis die aktuell gewählte Schrift nimmt. Entsprechend würden sich die Werte ändern, wenn man die Schriftgröße für `body` auf einen höheren oder niedrigeren Wert setzt.

Wenn niemand etwas an den Einstellungen gedreht hat und du auch in deinem Dokument nichts anders bestimmt hast, so entspricht `1em` übrigens `16px`.

Prozentangaben sind bei der Schriftgröße genauso wie `em`, statt `2.5em` kannst du auch `250%` schreiben, das macht keinen Unterschied.

Bringt uns zur Einheit Pixel, die wir jetzt schon benutzt haben – aber wie groß sind eigentlich `100px`?

WIE GROSS SIND EIGENTLICH 100 PIXEL?

Das hängt ganz davon ab. Mein einer Bildschirm hier beispielsweise ist ca. 38 cm breit und 33 cm hoch. Die eingestellte Auflösung ist 1280 * 1024 Pixel. Demnach entsprechen an diesem Bildschirm 100 Pixel 2,97 cm. Dann habe ich noch ein Netbook, bei dem ist der Monitor 22 cm breit und 13 cm hoch. Die eingestellte Auflösung ist 1024 mal 600 Pixel. Damit sind beim Netbook 100px 2,15 cm breit. Und jemand könnte natürlich auch die Auflösung herunterstellen, damit die Symbole und alles größer dargestellt werden. Also kurz: Eine Umrechnung in cm wird immer etwas anderes ergeben und das Maß der Dinge sind erst mal am Bildschirm die Pixel.

Aber wie groß sind jetzt 100 Pixel? Um ein Gefühl dafür zu bekommen, wie groß die Dinge sind, solltest du einfach mal nachmessen, wenn du im Web unterwegs bist. Das geht mit einer wunderbaren Erweiterung für den Firefox: die Webdeveloper Toolbar. Die natürlich noch mehr bietet und die wir noch weiter brauchen werden.

Für den Firefox ergibt es eine unglaubliche Menge an nützlichen Erweiterungen. Diese sind alle kostenlos und lassen sich problemlos in den Firefox integrieren. Erweiterungen für den Firefox findest du auf der Webseite: https://addons.mozilla.org/de/firefox/.

WEBDEVELOPER TOOLBAR INSTALLIEREN

Die Webdeveloper Toolbar ist eine äußerst nützliche Erweiterung für den Firefox. Sie integriert sich als zusätzliche Symbolleiste und bietet vieles, was das Herz des Webentwicklers begehrt.

Um die Webdeveloper Toolbar zu installieren, gehst du – mit dem Firefox – zuerst auf die Webseite: https://addons.mozilla.org/de/firefox/addon/60.

Klick auf den grünen Button „Zu Firefox hinzufügen".

Wenn du auf den Button geklickt hast, erscheint eine Warnung, dass du nur Add-ons von vertrauenswürdigen Quellen installieren sollst. Die Seite addons.mozilla.org ist eine vertrauenswürdige Quelle. Deswegen klickst du auf „Jetzt installieren".

Dann siehst du einen Fortschrittsbalken. Wenn die Installation abgeschlossen ist, erhältst du die Meldung, dass du Firefox neu starten solltst.

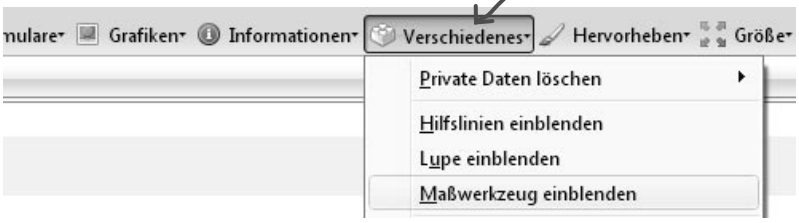
„Firefox neu starten", damit das Ganze klappt.

Nachdem du Firefox neu gestartet hast, siehst du am oberen Rand eine zusätzliche Symbolleiste. Diese kannst du übrigens über das Menü „Ansicht" ein- und ausblenden lassen.

WEBSEITEN MESSEN

Unter den vielen neuen Punkten der Webdeveloper-Toolbar wählst du dann „Verschiedenes“ und darin dann „Maßwerkzeug einblenden“.

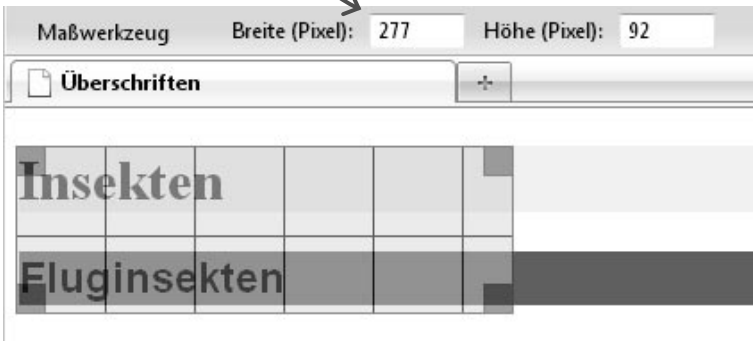
„Maßwerkzeug einblenden“ ist im Menü „Verschiedenes“ zu finden.



Jetzt verwandelt sich der Mauszeiger in ein Kreuz. Du kannst mit gedrückter Maustaste einen Bereich aufziehen. In der Zeile darüber siehst du dann die Ausmaße in Pixeln, die dein Bereich hat.

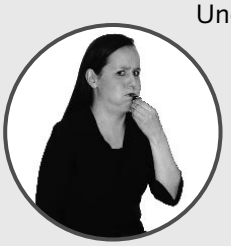
Die Größe des Bereichs kannst du verändern, indem du den Bereich an den Rändern anfasst und dann in die gewünschte Richtung ziehst. Und wenn du einen anderen Bereich messen willst, ziehst du einfach einen neuen Bereich auf.

In der Zeile darüber steht die Breite und die Höhe des aktuell aufgezogenen Bereichs.



Das Maßwerkzeug deaktivierst du, indem du es noch einmal im Menü „Verschiedenes“ auswählst.

SCHRIFTGRÖSSE DEFINIEREN



Und damit wieder zurück zur Schriftgröße, die wir in unserem Beispieldokument ändern wollen. Dafür ist ja die Eigenschaft `font-size` zuständig, hinter der du die gewünschte Größe angibst – in `px`, `%` oder `em`.

Wir wollen einmal `em` und `%` einsetzen, weil sie etwas ungewöhnlicher sind. Dazu ergänzen wir die folgenden Angaben zur Schriftgröße:

`body` erhält eine Schriftgröße von `100%`

Hier sind ein paar der ursprünglichen Formatierungen ausgelassen.

```
body {
  font-family: Arial, Helvetica, sans-serif;
  font-size: 100%;
}
h1 {
  font-family: "Times New Roman", Times, serif;
  font-size: 2.5em;
  color: red;
  background-color: yellow;
}
h2 {
  font-size: 1.2em;

}
```

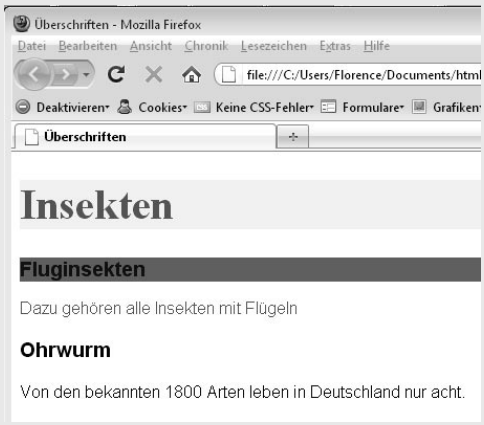
Die `h1` wird auf `2.5em` gesetzt.

Die `h2`-Überschrift erhält die Schriftgröße von `1.2em`.

Und bei „Fließkommazahlen" wird ein Punkt und kein Komma geschrieben.

Achtung: Zwischen Zahl und Einheit steht kein Leerzeichen!

Und so sieht es dann aus – prima!



Die `h1`-Überschrift ist größer geworden, die `h2`-Überschrift etwas kleiner.
Welche der drei Einheiten für die Schrift sollte man wählen? Über diese Frage gibt es endlose Diskussionen. Das W3C selbst empfiehlt `em`, da es wunderbar flexibel ist. Teilweise ist es aber schwerer zu händeln, da die Schriftgrößen weiter vererbt werden und das kann manchmal kompliziert werden.
Pixel als Einheit haben den Nachteil, dass sich die Schrift dann im Internet Explorer nicht über „Ansicht/Textgröße“ vergrößern lässt. Seit Internet Explorer 7 gibt es jedoch einen Seitenzoom, der auch bei Pixeln funktioniert, so dass das Problem nicht mehr so dringlich ist.

Wenn du `em` einsetzt, so solltest du für `body` eine Bezugsgröße angeben. Wenn du die ursprüngliche Größe belassen willst, dann gibt 100% an. Eigentlich bräuchte man das nicht – aber manche Internet Explorer-Versionen haben sonst einen Bug und zeigen die Schrift unter Umständen unleserlich klein an.
Die Bezugsgröße brauchst du nicht, wenn du alle Schriftgrößen in `px` angibst – `em` sind aber prinzipiell flexibler.

FETT UND KURSIV

Wie man Schriftgröße und Schriftart bestimmt, weißt du jetzt schon. Jetzt sehen wir uns an, wie man Schrift fett oder kursiv macht.

Die Dicke einer Schrift bestimmst du über `font-weight`. `font-weight: bold` macht die Schrift fett, `font-weight: normal` setzt eine fette Schrift normal. Und dafür, eine Schrift kursiv zu machen, dient `font-style`.

Wieder ergänzt du die Formatierungen in deinem Stylesheet mit den Überschriften.

```
h1 {
  /* die anderen Formatierungen wie gehabt */
  font-weight: normal;
  font-style: italic;
}
.betont {
  font-weight: bold;
}
```

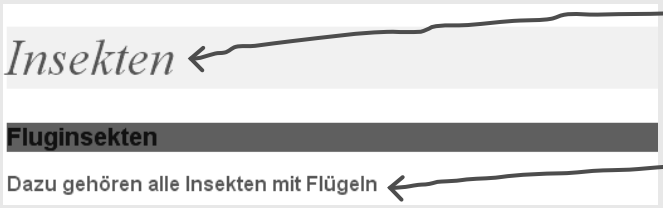
`font-weight: normal` **bewirkt, dass die** `h1`**-Überschrift nicht mehr fett dargestellt wird.**

`font-style: italic` **sorgt dafür, dass sie kursiv wird.**

Alle Elemente mit der Klasse `betont` **sollen fett dargestellt werden.**

Im Beispiel siehst du auch, wofür du `font-weight: normal` brauchst. Überschriften sind – so legt es das browserinterne Stylesheet fest – erst mal fett. Das kannst du mit `font-weight: normal` aufheben.

Im Beispiel wird dann bei der `h1`-Überschrift `font-style: italic` eingesetzt, um sie kursiv zu machen. Ein kursives Element könntest du über `font-style: normal` auch wieder normal setzen.



Die Überschrift ist nicht mehr fett, dafür aber kursiv.

Der Absatz mit `class="betont"` **ist jetzt fett.**

Es gibt eigentlich eine feinere Abstufung bei `font-weight`, du kannst hier die Zahlen 100 bis 900 eingeben, wobei 100 ganz dünn und 900 ganz dick ist. Allerdings werden diese feinen Nuancen von den Browsern derzeit nicht unterstützt, so dass du eigentlich nur `font-weight: bold` oder `font-weight: normal` brauchst.

KAPITÄLCHEN UND ZEILENHÖHE

Noch zwei weitere Eigenschaften sind wichtig bei der Schriftformatierung: Mit `font-variant: small-caps` wird ein Element mit Kapitälchen dargestellt.

`line-height` bestimmt die Zeilenhöhe, also den Abstand zwischen den Zeilen. Hier solltest du einen Wert ohne Einheit angeben.

Weitere Ergänzung unserer Formatierungen

```
body {
  /* Alles andere wie gehabt */
  line-height: 1.4;
}
h2 {
  /* Alles andere wie gehabt */
  font-variant: small-caps;
}
```

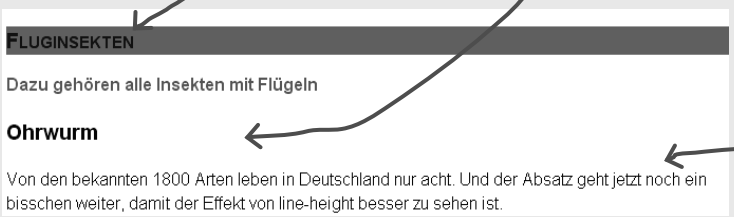
Bei der Formatierung von `body` ergänzt du `line-height: 1.4;`

Die `h2`-Überschrift soll in Kapitälchen dargestellt werden.

Bei `line-height` könntest du auch eine Einheit angeben – also beispielsweise `px` oder `em`. Ohne Einheit wird die Zeilenhöhe aber an das untergeordnete Element vererbt, passt sich also der Schriftgröße an.

Die `h2`-Überschrift steht in Kapitälchen.

Der Abstand zwischen den Zeilen ist über `line-height: 1.4` vergrößert.

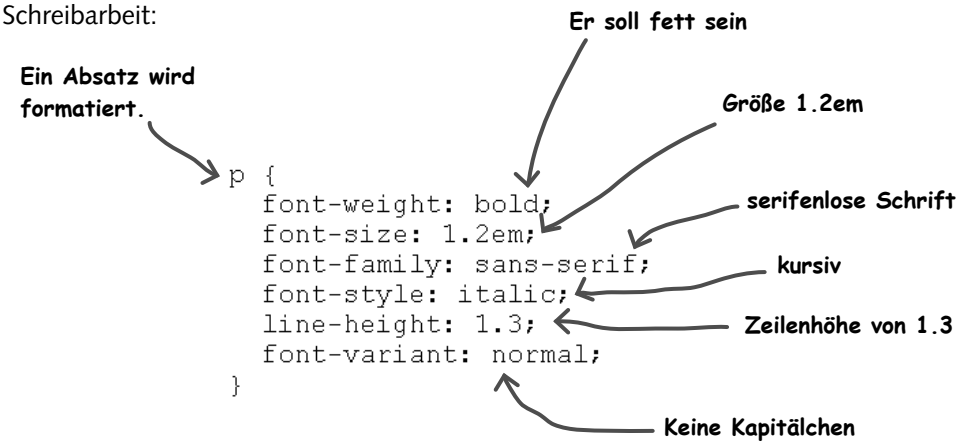


Damit man den Effekt von `line-height` sieht, habe ich noch etwas mehr Text eingefügt.

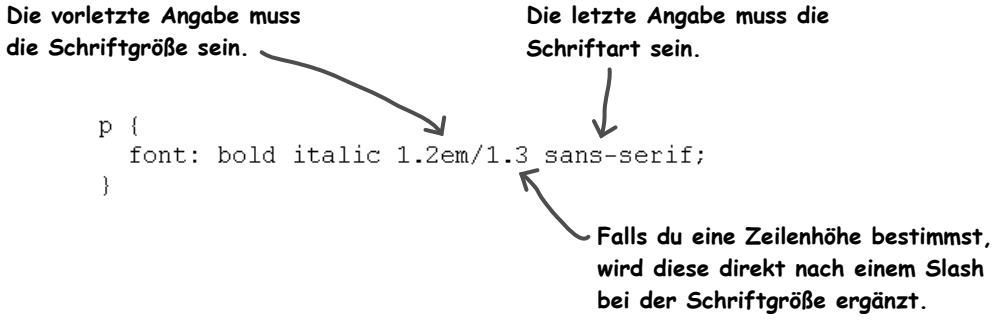
An der `h2`-Überschrift siehst du den Effekt von `font-variant: small-caps`: Alles wird zu Großbuchstaben und die ursprünglichen Großbuchstaben sind noch einen Tick größer. Ein netter Effekt für Überschriften. Längere Texte würden dadurch schlechter lesbar werden.

FONT-KURZSCHREIBWEISE

Für die Schrift hast du jetzt viele Eigenschaften kennen gelernt. Wenn du alle auf einmal angibst, ist das ziemlich viel Schreibarbeit:



Und deswegen gibt es eine verkürzte Schreibweise. Bei dieser gibst du einfach die Eigenschaft `font` an. Dahinter kommen alle gewünschten Eigenschaften, die nur durch Leerzeichen getrennt sind. Dabei ist die Reihenfolge nicht wichtig bis auf die letzten beiden Wert: die letzten beiden Werte müssen zuerst die Schriftgröße und danach die Schriftart sein. Und außerdem gibst du die Zeilenhöhe direkt hinter der Schriftgröße mit einem Slash getrennt an.



Obligatorisch ist bei dieser Kurzschreibung nur Schriftgröße und Schriftart in der gezeigten Reihenfolge.

Wenn eine der anderen Eigenschaften fehlt, wird sie automatisch auf „normal“ gesetzt. Deswegen brauchst du auch im Beispiel „normal“ für `font-variant: normal` nicht zu schreiben.

ABSATZFORMATIERUNGEN

Nach der Schriftformatierung kommt die Absatzformatierung. Da sind zwei Eigenschaften wichtig: `text-align` und `text-indent`. `text-align` dient zur Ausrichtung von Absätzen. Es kann die Werte `center` (zentriert), `right` (rechtsbündig) und `left` (linksbündig) annehmen. Mit `text-indent` realisierst du einen Erstzeileneinzug.

Ergänzen wir Absatzformatierungen bei unserem Dokument.

Bei den Formatierungen der `h1`-Überschrift ergänzt du ein `text-align: center;`

```
h1 {
  text-align: center;
}
h2 {
  text-align: right;
}
p {
  text-indent: 2em;
}
```

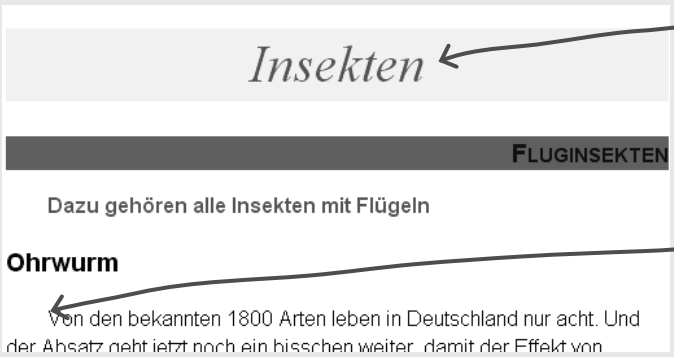
Bei der `h2`-Überschrift hingegen ein `text-align: right`, damit wird die Überschrift rechts angeordnet.

Ganz neu sind hingegen diese drei Zeilen, darüber werden alle Absätze formatiert.

Über `text-indent: 2em` werden die ersten Zeilen der Absätze um 2em eingerückt.

Damit wird die `h1`-Überschrift zentriert und die `h2`-Überschrift rechts angeordnet.

Alle Absätze erhalten einen Erstzeileneinzug über `text-indent`. Damit du den Effekt deutlich siehst, muss der Absatz länger sein als eine Zeile.



Die `h1`-Überschrift ist zentriert.

Die `h2`-Überschrift rechts angeordnet.

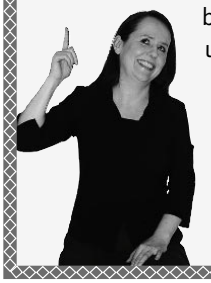
Erstzeileneinzug beim Absatz

Prinzipiell kannst du bei `text-indent` alle Einheiten einsetzen, die du vorher bei der Schriftgröße kennengelernt hast. Aber wie auch bei der Schriftgröße sind nur wenige Einheiten sinnvoll: nämlich nur `px`, `em` und `%`.

NOCH MEHR TEXTFORMATIERUNGEN

Das waren jetzt die wichtigsten CSS-Formatierungen für Schrift und Absätze. Daneben gibt's noch ein paar weitere, die nur kurz erwähnt werden sollen:
Was eigentlich bei der Schriftformatierung noch fehlt, ist die Unterstreichung. Diese wird über `text-decoration: underline` realisiert. Darauf kommen wir auführlich noch bei der Formatierung von Links zu sprechen.
Über `letter-spacing` kannst du Buchstaben gesperrt drucken lassen. Dahinter schreibst du einfach einen Wert mit einer Einheit, also z.B. `letter-spacing: 0.3em`.
`word-spacing` beeinflusst hingegen den Abstand zwischen den Wörtern.
Mit `text-transform: lowercase` verwandelst du allen Text in Kleinbuchstaben, `text-transform: uppercase` macht alles zu Großbuchstaben. Bei `text-transform: capitalize` wird jeweils der erste Buchstabe der Wörter großgeschrieben. Das braucht man bei deutschen Texten kaum.

Über `text-decoration: underline` realisierst du eine Unterstreichung. Standardmäßig sind Links unterstrichen – und deswegen denkt jeder bei einem unterstrichenen Wort, dass es ein Link ist. Verwende deshalb Unterstreichungen nur bei Links.
Es gibt übrigens einen Unterschied zwischen `text-transform: uppercase` (Großbuchstaben) und `font-variant: small-caps` (Kapitälchen). Bei Kapitälchen werden zwar auch durchwegs Großbuchstaben benutzt, aber die ursprünglichen Großbuchstaben sind einen Tick größer.



gesperrt gedruckt: sinnvoll für einzelne Hervorhebungen

letter-spacing: 0.3em sorgt für gesperrt gedruckte Buchstaben.

größerer Abstand zwischen Wörtern

word-spacing: 0.3em vergrößert den Abstand zwischen den Wörtern.

TEXT-TRANSFORM: UPPERCASE: LOREM IPSUM DOLOR SIT AMET, CONSECTETUR ADIPISICING ELIT, SED DO EIUSMOD TEMPOR INCIDIDUNT UT LABORE ET DOLORE MAGNA ALIQUA.

Durchgängig Großbuchstaben

text-transform: lowercase: lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

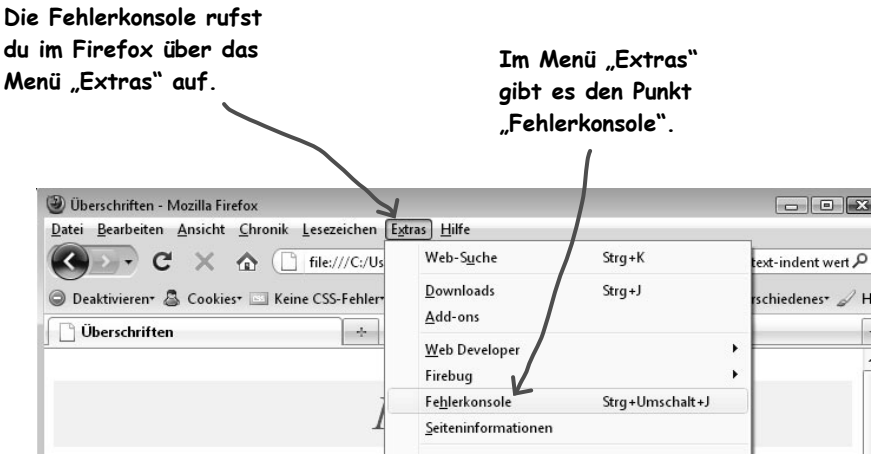
alles in Kleinbuchstaben

Text-transform: Capitalize: Lorem Ipsum Dolor Sit Amet, Consectetur Adipisicing Elit, Sed Do Eiusmod Tempor Incididunt Ut Labore Et Dolore Magna Aliqua.

Nur der erste Buchstabe von jedem Wort ist groß

FEHLER IM CSS-CODE AUFSPÜREN

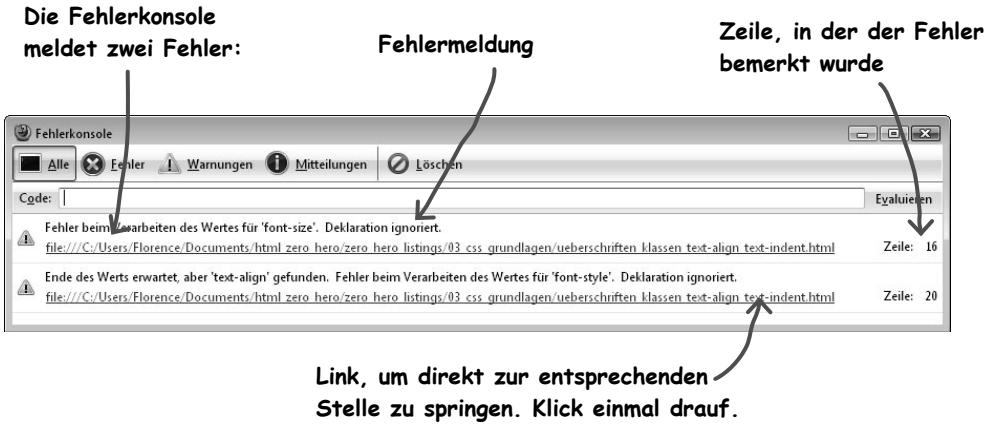
Du hast in diesem Kapitel schon einiges an CSS-Code geschrieben. Und sicher ist dir auch schon der eine oder andere Tippfehler unterlaufen. Es genügt schon, einen Strichpunkt zu vergessen oder einen Buchstabendreher bei einer Eigenschaft zu machen, damit die Formatierungen nicht angewandt werden.
Bei der Fehlersuche hilft die Fehlerkonsole von Firefox.



Wenn du die Fehlerkonsole aufrufst, so zeigt sie dir die Fehler und Hinweise aller Webseiten, die du in letzter Zeit besucht hast. Dich interessieren aber die Fehler und Hinweise zu deinem Dokument. Lösch deswegen erst einmal den ganzen Inhalt der Fehlerkonsole.

Wenn du die anderen Meldungen gelöscht hast, aktualisiert du die Webseite, die du nach Fehlern durchsuchen möchtest, über den Aktualisieren-Button im Browser. Dann rufst du die Fehlerkonsole erneut auf.

Im folgenden Screenshot siehst du, was die Fehlerkonsole dir bei einem Fehler zeigt. Ich habe zur Demo einmal zwei Fehler in das Beispiel eingebaut.

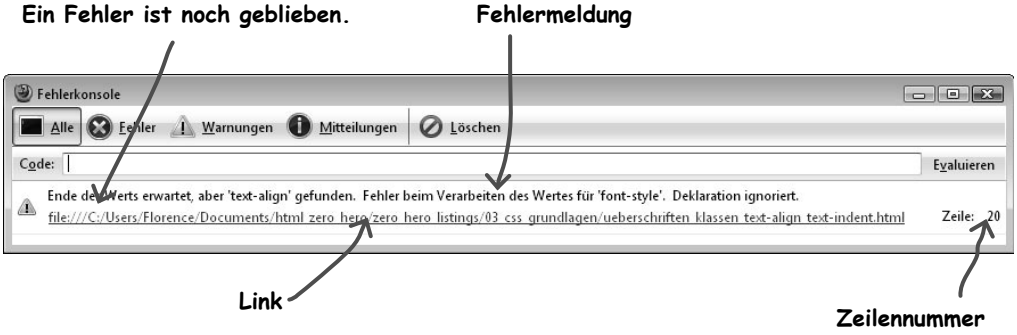


Die Fehlermeldungen sind recht aussagekräftig. Die erste Fehlermeldung heißt beispielsweise „Fehler beim Verarbeiten des Werts für font-size. Deklaration ignoriert". Der erste Teil ist die eigentliche Meldung und der zweite Teil sagt, was der Browser bei diesem Fehler macht: Er ignoriert die entsprechende Stelle. Wenn du auf die unterstrichene Zeile klickst, gelangst du direkt zur entsprechenden Stelle in der Quellcode-Ansicht.

Mit der Fehlerkonsole ist es es ähnlich wie mit dem HTML-Validator: Sie meldet den Fehler oft später, als er aufgetreten ist. Also kontrolliere am besten auch immer die Zeile(n) davor, wenn du den Fehler nicht in der Zeile selbst findest. Den Fehler kannst du nicht hier direkt beheben, dafür wechselst du wieder in deinen Editor.

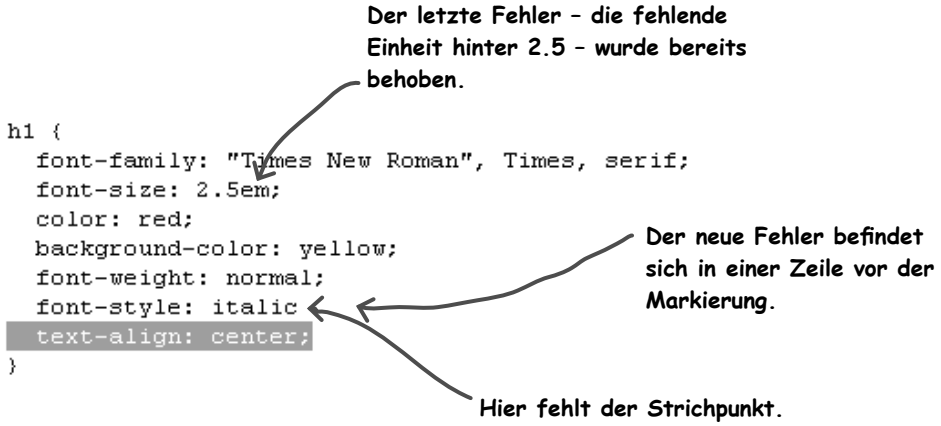
Geh also in den Editor, um deinen Fehler zu beheben. Dann löscht du den Inhalt der Fehlerkonsole, aktualisierst das Dokument und siehst nach, was die Fehlerkonsole jetzt meldet.

Im Beispiel meldet sie „Ende des Werts erwartet, aber text-align gefunden. Fehler bei Verarbeitung des Wertes für font-style“.



Klick auf den Link, um dir die entsprechende Stelle im Quellcode anzuschauen: Hervorgehoben ist die Zeile „text-align: center“. Diese sieht ganz gut aus, aber was ist mit der Zeile davor?

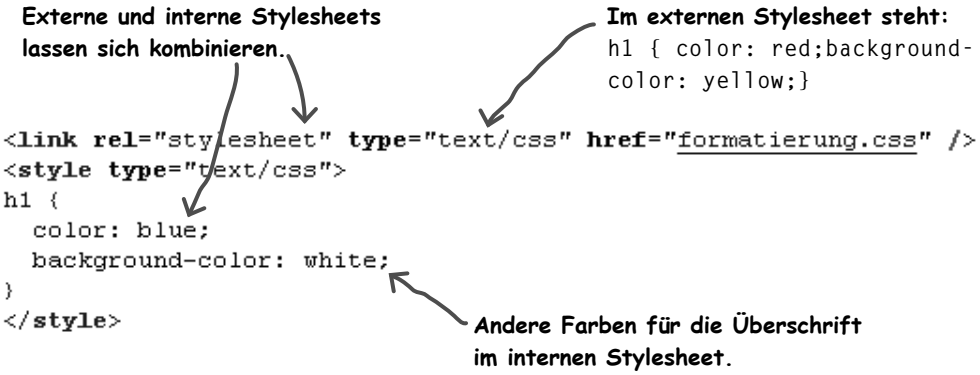
„Ende des Werts erwartet, aber text-align ... gefunden“, das bedeutet, dass die Zeile davor nicht korrekt beendet wurde: Es fehlt der Strichpunkt.



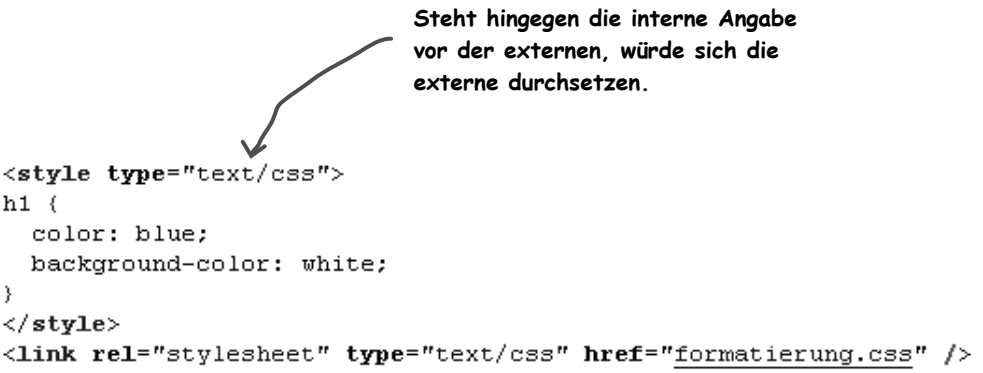
Die Fehlerkonsole im Firefox ist ein recht nützliches Tool. Alternativ dazu gibt es natürlich auch ein Validierungstool des W3C, das du unter http://jigsaw.w3.org/css-validator/ findest und das ganz ähnlich wie der im letzten Kapitel vorgestellte HTML-Validator funktioniert.

WER SETZT SICH DURCH?

Du hast in diesem Kapitel bisher schon viele Textformatierungen kennen gelernt und gesehen, dass sich CSS-Angaben an verschiedenen Stellen einsetzen lassen. Beispielsweise in einem internen oder einem externen Stylesheet. Das lässt sich natürlich auch kombinieren. Stellt sich dann bloß die Frage: Wer setzt sich durch, wenn die Angaben sich widersprechen?

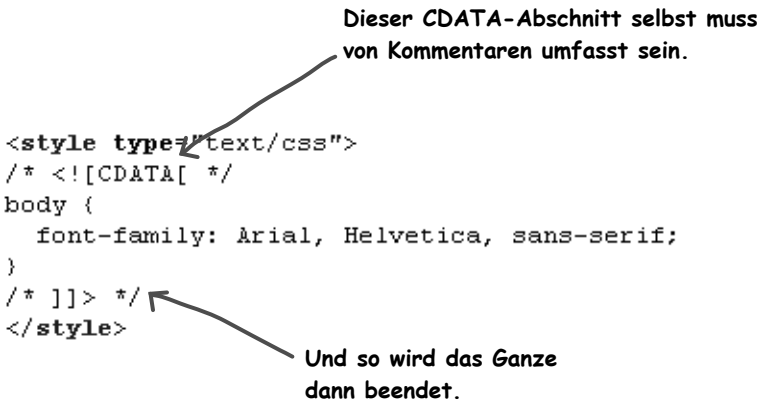


Die Antwort: Das, was im Code zuletzt steht, in diesem Fall also die interne Angabe. Die Überschrift erhält eine blaue Schriftfarbe und einen weißen Hintergrund.



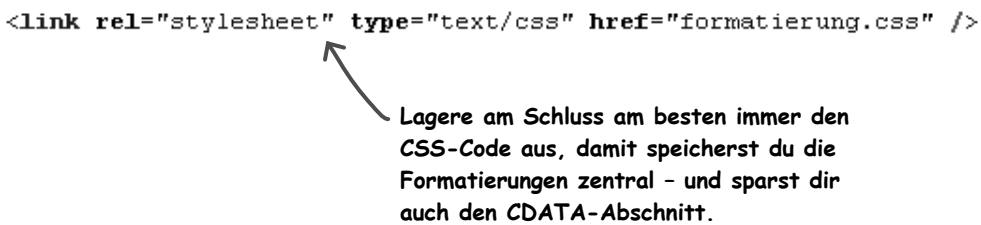
KRYPTISCHE ASCII-ART: DER CDATA-ABSCHNITT

Noch eine letzte Anmerkung zur Angabe des CSS-Codes. Wenn du mit einem internen Stylesheet arbeitest, so solltest du bei der Verwendung von XHTML streng genommen den CSS-Code innerhalb eines CDATA-Abschnitts unterbringen.



Eigentlich würde ich dir raten, prinzipiell am Schluss immer den CSS-Code in einer eigenen Datei auszulagern – dann kannst du dir diese CDATA-Abschnitte sparen.

Für die ersten Tests und Experimente hingegen ist es praktisch, mit den internen Stylesheets zu arbeiten, aber da spielt es keine Rolle, ob du den CDATA-Abschnitt ergänzt hast oder nicht.



Puh! Der Einstieg in CSS ist geschafft.

Du kennst jetzt das Grundprinzip von CSS und weißt, wie man Schrift- und Absatzformatierungen durchführt.

Hier noch einmal die wichtigsten CSS-Konzepte:

Eine CSS-Regel beginnt mit einem Selektor.

Der Selektor bestimmt, für welche Elemente die in geschweiften Klammern folgenden Formatierungen gelten sollen.

```
p { color: red; }
```

Ein Selektor wie `p` ist ein sogenannter Elementselektor, weil er alle entsprechenden Elemente auswählt – also beispielsweise alle Absätze. Über Elementselektoren legst du die allgemeinen Formatierungen fest.

Soll etwas für das ganze Dokument gelten, so gibst du die Formatierung bei `body` an. Dann gelten diese Formatierungen, für alle HTML-Elemente, die innerhalb von `<body>` und `</body>` stehen. Dies gilt nur für Eigenschaften, die vererbt werden – zu Eigenschaften, die nicht vererbt werden, kommen wir später noch.

Einzelnen Elementen, die anders aussehen sollen als die allgemeine Formatierung, weist du im HTML-Code eine Klasse zu, also z.B. `<p class="betont">...</p>`. Der Selektor, über den du die Elemente mit dieser Klasse auswählst, heißt dann `.betont` – der Name der Klasse mit einem Punkt davor.

All diese Konzepte werden uns weiter begleiten – in den nächsten Kapiteln lernst du neue HTML-Elemente kennen und erfährst gleichzeitig, wie man sie per CSS formatiert.