Hypertext Markup Language und Cascading Style Sheets Teil 1

Jobst-Hartmut Lüddecke

29. September 2011

Zusammenfassung

Kurzer, pragmatischer Einstieg in die *Hypertext Markup Language* (HTML) und die *Cascading Style Sheets* (CSS) mit hoffentlich gleich stattfindendem Erfolgserlebnis in der Umsetzung.

Erklärt wird die *Klammerstruktur* von HTML, der Kopf- und Rumpfteil, der Titel der Seite, sowie Überschriften und Textblöcke. Nebenbei werden *Steuerzeichen* für Sonderzeichen eingeführt. Dann geht es zum CSS und es wird das *Erscheinungsbild* dieser Überschriften und Textblöcke definiert.

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Hypertext Markup Language (HTML)

HTML ist eine *Tag*-orientierte Programmiersprache um einen *Textsatz*[1] zu programmieren. Diese *Tags* sind Befehle im *Klartext*, die das Erscheinungsbild des zu setzenden Textes direkt bestimmen. Sie haben in *HTML* **fast immer** eine *Klammerstruktur* und sind in *spitzen Klammern* (<>) gesetzt.

Die Reihenfolge der Klammern ist im mathematischen Sinn zu beachten und auf einen Befehl (Klammer auf) folgt in fast allen Fällen der gleiche Befehl mit einem einleitenden *Slash*[2] als *Klammer zu* der den *Gültigkeitsbereich* des Befehls abschließt.

Dieses *HTML-Programm* kann mit jedem normalen *Editor*[3] programmiert werden. Nicht mit einem *Textverarbeitungsprogramm* wie *Word*, denn dieses verwendet intern seine eigenen – nicht sichtbaren – Befehle zur Formatierung. Diese *unsichtbaren* Teile[4] der Seite bereiten in der Regel bei *jedem Programm* **heftige Probleme**.

Die html-Seite wird dann von einem *Browser* (Mozilla Firefox, Opera, Internet Explorer, Safari usw.) *interpretiert* und dargestellt. Leider haben alle mir bekannten *Browser* ihre Macken und es ist nicht selbstverständlich, dass auch alle *Tags* funktionieren und richtig umgesetzt werden.

Deshalb fangen wir auch ganz einfach an:

Jede html-Seite fängt mit **<html>** an und hört mit **</html>** wieder auf.

```
<html>

</html>
```

Abbildung 1: html-Seite, 1. Schritt

Dann gibt es in jeder html-Seite zwei wesentliche Blöcke. Dies ist der *Kopf* (head) mit *Informationen über die Seite* und der *Rumpf* (body) mit dem eigentlichen *Inhalt der Seite*.

```
<html>
<head>
</head>
<body>
</body>
</html>
```

Abbildung 2: html-Seite mit Kopf und Rumpf

Eine wichtige Information über die Seite ist der *Titel* der Seite, den wir folglich gleich mit <title> in den *Head* einfügen.

In jedem *Body* sind Überschriften und Textblöcke zu finden. Die Überschriften (header) gibt es in verschiedenen Größen und der *Tag* dafür ist <h1>, <h2>, <h3> usw. wobei <h1> die größte Überschrift ist.

[1] *Textsatz* sind in der Regel Anweisungen, wie der *Text gesetzt* werden soll. Diese Anweisungen bilden dann ein Programm, welches entweder, wie bei *html* durch einen *Browser interpretiert* werden, oder wie z.B. bei TEX und LATEX für ein bestimmtes Ausgabegerät *compiliert*, also übersetzt werden.

[2] ein *Slash* ist der Strich von links-unten nach rechts oben. Im Gegensatz dazu der *Backslash* der Strich von links-oben nach rechts-unten.

[3] ein *Editor* ist hier kein *Herausgeber eines Buches*, sondern ein Programm mit dem man reine Texte, ohne Formatierung, erstellen kann. Hauptsächlich wird der *Editor* dazu verwendet ein Programm mit ASCII-Zeichen zu schreiben.

[4] *Steuerzeichen* und *Steuersequenzen* (Kombination aus mehreren Steuerzeichen hintereinander). Siehe dazu auch *ASCII* im letzten Skript.

Ich spreche hier von *Textblock* – auch *Paragraph* genannt – und nicht von *Absatz*, denn *Absatz* ist nicht eindeutig und wird mal als Textblock und mal als Abstand zwischen den Textblöcken verwendet.

Der Textblock – also der Text und nicht die Lücke zwischen den Texten – wird mit dem *Tag* <p> für *Paragraph* gesetzt. Die Abstände zwischen den einzelnen Blöcken werden an anderer Stelle definiert. Innerhalb der Blöcke sorgt der *Browser* selbst für einen *Zeilenumbruch* und zwar in Abhängigkeit von der aktuellen *Fensterbreite* des Browsers.

Noch einmal: Das Erscheinungsbild der Seite wird durch die *Tags*, also die Formatierungsbefehle bestimmt und **nicht** durch die Art und Weise, wie Sie den Text im Editor geschrieben haben!

Es ist unbedingt wichtig das Ende des Gültigkeitsbereiches des jeweiligen *Tags* in der Art einer *Klammer zu* kenntlich zu machen. Im Falle von <*p*> und </*p*> werden dadurch die Abstände zu den Teilen vor und nach diesem Textblock bestimmt.

```
<html>
<head>
<title>Erste Flugversuche</title>
</head>
<body>
<h1>Maximale &Uuml;berschrift</h1>
<p>
Ein kleiner Test, ob auch alle Buchstaben da sind.
</p>
<p>
The quick brown fox jumps over the lazy dog.
The quick brown fox jumps over the lazy dog.
The quick brown fox jumps over the lazy dog.
The quick brown fox jumps over the lazy dog.
The quick brown fox jumps over the lazy dog.
</p>
<h2>Hier nun eine mittlere &Uuml;berschrift</h2>
<h3>Noch eine Stufe kleiner</h3>
</body>
</html>
```

Abbildung 3: html-Seite mit Titel, Überschriften und Paragraphen

1.1 Umlaute, Ligaturen und Sonderzeichen

Wie wir bisher gesehen haben, sind im *ASCII-Zeichensatz* viele gewünschte Zeichen nicht enthalten. Man kann zwar die Windows-Umlaute unter Windows im html-Text verwenden, nur wird dann diese Seite auch nur auf Windows-Rechnern korrekt angezeigt. Man denke an *den kleinsten gemeinsamen Nenner*[5] und die Rechnerwelt besteht mitnichten nur aus Windows-Rechnern.[6]

Um trotzdem mehr Zeichen anzeigen zu können, als der ASCII-Code bietet, werden in HTML viele zusätzliche Zeichen als *Steuersequenzen* angeboten. Diese *Steuersequenzen* fangen mit einem **&** (Ampersand oder Und-Zeichen) an und enden mit einem **;** (Semikolon).

Dazwischen ist ein kurzer Name für das Zeichen.

[5] Die ist mal wieder *ASCII*, wie bereits mehrfach betont: Sie kommen nicht daran vorbei!
[6] Auch wenn *Bill Gates* dies gern so hätte.

Bei den *Umlauten* ist dies ganz einfach ein *uml*. Möchte man also ein *ä* darstellen, handelt es sich um einen *kleinen a-Umlaut* in der html-Syntax *ä*. Soll es aber lieber ein *Ä* sein, also ein *großer A-Umlaut*, dann nimmt man auch in HTML ein großes A im Steuerzeichen *Ä*.

Entsprechendes gilt für die anderen Umlaute. Ein *Ö* ist ein großer *O-Umlaut* und wird folglich zum *Ö* und ein *ü* als kleiner u-Umlaut wird zum *ü*.

Etwas schwieriger wird es mit dem im Deutschen noch gebräuchlichen *ß*. Dies ist eine *Ligatur*[7] aus dem nicht mehr gebräuchlichem *langen s*[8] und dem heute nur noch gebräuchlichem *kurzen s*. Ursprünglich wurde das *lange s* innerhalb des Wortes und das *kleine s* an Silben- und Wortenden gebraucht. Dies wurde überall in Europa so gemacht! Endete nun eine Silbe auf *ss*, dann wurde zuerst ein *langes s* und dann ein *kurzes s* geschrieben.[9]

Bei der *Schreibschrift in Druckbuchstaben*, der im Alltag gebräuchlichen *Kanzleischrift*, der *Cancelleresca corsiva*[10] waren dies auch zwei s, ein langes und ein kurzes. Das *kurze s* wanderte immer näher an das *lange s* und wuchs mit der Zeit zusammen.

Wie man an diesem kleinen Text sieht, sind durchaus zwei verschiedene Formen vom s vorhanden. Dies ist das lange ſ im Text und dies ist das kleine s an Silben-Enden. Endet die Silbe auf zwei s, dann ergibt sich ein ſs.[11]

Etwas anders sah es aus, wenn in *Fraktur* geschrieben wurde. Diese *gebrochene* Schrift wurde verwendet um die Buchstaben schmaler zu machen, damit mehr Text auf eine Seite passt, um Pergament bzw. Papier zu sparen. Als *Gutenberg* die erste Bibel mit beweglichen Lettern gedruckt hat, hat er selbstverständlich in Fraktur gedruckt und die scharfen *ss-Laute* an Silben-Enden wurden als *sz* geschrieben. So ist aus *ss* an Silben-Enden in Fraktur ein *sz* geworden.

Aus ſ und z wurde ein ß. Wie man an dieſem kleinen Text ſieht, ſind durchaus zwei verſchiedene Formen vom ſ gebräuchlich. Dies iſt das große ſ im Text und dies iſt das kleine s an Silbenenden.

Da zu Zeiten Gutenbergs diese Bibel auch die Funktion eines Nachschlagewerkes – ähnlich dem heutigen *Duden* – hatte, und die Fraktur bis zum *Dritten Reich*[12] gebräuchlich war, wird diese Ligatur noch heute als *sz* angesprochen.

Bis zur letzten Rechtschreibreform ist der Gebrauch der *sz-Ligatur* auch den Regeln mit den *ss* auf Silbenenden gefolgt. Nach der Reform ist der Gebrauch für mich nicht mehr logisch und nachvollziehbar! Meiner Meinung nach ist diese Ligatur völlig überholt und überflüssig, denn das *lange s* wird ja auch nicht mehr verwendet. Alle anderen Staaten haben die *sz-Ligatur* einfach und konsequent durch zwei kleine s ersetzt.

Für alle, die das *ß*[13] in ihren html-Seiten verwenden möchten, sei hier *ß* für sz-Ligatur genannt.

Selbst *&*[14]ist eine Ligatur aus *et* für lateinisch *und*. Da dieses Zeichen aber auch *Ampersand* heißt, wird es in HTML als *&* geschrieben.

Andere gebräuchliche Ligaturen sind æ (ae) = *æ* und œ (oe) = *œ* usw.

Wer noch mehr davon haben will, sei auf folgende Seite verwiesen:

http://de.selfhtml.org/html/referenz/zeichen.htm

[7] Ein *Zusammenwachsen* zweier Buchstaben

[8] Sieht in etwa so aus wie ein kleines *f* ohne Querstrich. Als bewegliche Letter im Druck wurde der untere Rückschwung immer mehr beschnitten, da dieser Rückschwung in den Bereich des Buchstabens davor hereinragt und damit Probleme macht.

[9] Damit erklärt sich auch die alte Verwendungsregel des ß. Die neue Regel ist mir nicht schlüssig. Abgesehen davon halte ich diese Ligatur für völlig überflüssig, da das *lange s* auch nicht mehr gebraucht wird.

[10] Daher kommt die Bezeichnung *Kursive*. Da hauptsächlich italienische *Schriftmeister* des 15. und 16. Jahrhunderts, wie *Aldus Manutius*, *Francesco Griffo* und *Lodovico Arrighi* zu der Zeit Grundlagenwerke über den Gebrauch der *Kanzleischrift* verfasst haben und erste *Schriftschnitte* entworfen haben. Die *Kursive* wird in Englisch auch als *italic* bezeichnet.

[11] Dies ist übrigens die *Cancelleresca corsiva* von *Lodovico Arrighi*, genannt *Vicentino* aus dem 15. Jahrhundert.

[12] Folglich ist es völlig falsch eine Fraktur mit *Nazis* zu verbinden, denn die haben den Gebrauch der Fraktur **abgeschafft**

[13] An dieser Kursive sieht man sehr schön das zusammengewachsene große und kleine s.

[14] An dieser Kursiven sieht man auch sehr schön das zusammengewachsene e und t.

2 Cascading Style Sheets (CSS)

Dies ist die bessere Art seine Seiten *anzuhübschen*. Kaum können die Leute eine Überschrift und einen Textblock in *html* definieren, wünschen sie sich ein vom Standard abweichendes Erscheinungsbild.[15]

Dies kann man auf 2 verschiedene Arten machen.

1. Man kann für jeden in der html-Seite verwendeten *Tag* direkt an der Stelle zusätzliche Attribute in den *Tag* einfügen. Damit gilt dieses Attribut für genau den Gültigkeitsbereich dieses einen *Tags*. Hat man mehr als eine html-Seite zu pflegen, dann wird es richtig *haarig* mit einem enormen Arbeitsaufwand. Dies ist höchst unpraktisch und ich möchte Sie gleich entmutigen dies zu tun.
2. Man definiert sich für jeden *Tag* **einmal** das Erscheinungsbild an zentraler Stelle und hat den riesigen Vorteil, dass **alle** Seiten, die auf diese Definition zugreifen gleich sind und mit einer Änderung in diesem *Style Sheet* alle betroffenen Seiten *gleich* geändert werden.

2.1 HTML-Seite mit der Verwendung eines CSS

Dann also gleich zum Kochrezept: *Man nehme* seine *html-Seite* und füge in den Kopf die Information ein, *welche* Datei die *Definitionen* enthält. In unserem Beispiel nennen wir diese Datei *meine.css*[16] Hier sehen wir auch gleich einen *Tag* ohne *Klammer zu*, da er selbst alle Informationen enthält und daher nicht noch einmal abgeschlossen werden muss.

```
<html>
<head>
<title>Erste Flugversuche</title>
<link rel="stylesheet" type="text/css" href="meine.css">
</head>
<body>
<h1>Maximale &Uuml;berschrift</h1>
<p>
Ein kleiner Test, ob auch alle Buchstaben da sind.
</p>
<p>
The quick brown fox jumps over the lazy dog.
The quick brown fox jumps over the lazy dog.
</p>
<h2>Hier nun eine mittlere &Uuml;berschrift</h2>
</body>
</html>
```

Abbildung 4: html-Seite mit Link auf eine CSS-Datei

[15] Dies ist durchaus zulässig, dann aber bitte *sauber* und *ordentlich* mit *Cascading Style Sheets*.

[16] Die Datei kann einen beliebigen Namen haben, muss aber eine *Extension* **css** haben. Weiterhin muss der Dateinamen mit dem *link* übereinstimmen.

2.2 Das verwendete CSS

Als erstes und als einziger HTML-Tag kommt die Definition, dass es sich um ein *style sheet* vom Typ *Text* und genauer vom Typ *css* handelt. Mit folgendem *Tag* wird dies erledigt:

```
<style type="text/css">
```

Danach wird das ganze *style sheet* im Sinne für die HTML-Seite *auskommentiert*. D.h. alles was zwischen den HTML-Kommentaren

```
<!--                                    //-->
```

steht wird nicht als html-Code vom Browser wahrgenommen. Diese *Klammer* dürfen Sie auf keinen Fall vergessen.

Auch in *Cascading style sheets* kann man Kommentare schreiben. Diese stehen zwischen **/*** und ***/** und ich möchte Sie unbedingt ermutigen diese Kommentare auch zu verwenden. Sie sind ein elementarer Bestandteil der Dokumentation, erleichtern die Lesbarkeit des Programmes und sind eine unverzichtbare *Gedankenstütze* für Passagen, bei denen man sich *etwas mehr* gedacht hat.[17]

Als nächstes kann man jeden verwendeten *html-Tag* näher definieren, indem man am Anfang der Zeile diesen *Tag* hinschreibt und danach in *geschweiften Klammern* verschiedene *Attribute* näher bestimmt. Diese *Attribute* werden durch ein *Semikolon* voneinander getrennt.

Gehen wir jetzt das Beispiel der Reihe nach durch:

dummy *Mozilla* und alle *Stammesangehörigen*[18] wie *Netscape*, *Firefox*, *Sea-Monkey* usw. haben bisher den gleichen Fehler, dass genau die **erste** CSS-Definition **nicht** ausgewertet wird. Dies kann man damit umgehen, dass man einen *Tag* definiert, den man gar nicht verwendet. Ich nenne ihn in diesem Fall *dummy*. Danach folgt eine beliebige, gültige Definition eines Attributes und man spart sich viel Zeit der *Fehlersuche*, warum eine Definition einfach nicht mit *Mozilla* funktionieren will.

body dies ist die Definition der html-Seite selbst. Es kann Sinn machen Ränder (margin) zu definieren, oder eine Hintergrundfarbe, ein Hintergrundbild usw. In diesem Fall:

margin-left: 20mm; bedeutet 20mm linker Rand, also 2cm Platz zwischen dem linken Rand des Browserfensters und dem Anfang eines linkbündigen Textes.

margin-right: 20mm; sind 20mm Rand zwischen dem rechten Textrand und dem Browserfenster.

margin-top: 20mm; sind 2cm Platz oben und

margin-bottom: 35mm; sind 3,5cm unten.

background-color:rgb(90%,60%,90%); jetzt mischen wir die *Hintergrundfarbe*. Der einfachste Weg ist der, dies über Prozentangaben in *RGB* zu tun. *RGB* sind die *additiven* Farben[19], die beim *Farbschema* eines Bildschirmes verwendet werden. Hierbei gibt es 3 Farbkanäle. Dies sind *Rot*, *Grün* und *Blau*. Schaltet man alle 3 Farben aus, also 0%, dann hat man

[17] Nach 14 Tagen wissen Sie selbst nicht mehr genau, warum Sie eine Sache *genau so* und nicht anders gemacht haben. Da greift die *Gedankenstütze*. Abgesehen davon, wie soll ich nachvollziehen, was Sie sich dabei gedacht haben. Habe ich eine *Kristallkugel*? ☺

[18] viele Browser sind aus der ersten, freien Mozilla-Version als Weiterentwicklung entstanden.

[19] Man spricht auch von *Lichtmischung*.

Schwarz und alle 3 Farben voll an, also 100%, dann sollte es ein *Blütenweiss* sein. Dazwischen kann man mischen, also der erste Wert ist der *Rotanteil*, der zweite der *Grünanteil* und der dritte der *Blauanteil*. Nimmt man also 100% Rot, plus 0% Grün, plus 100% Blau, sollte dies ein recht knalliges Lila ergeben. Die oben gewählte Einstellung ergibt ein kaltes Hellgrau. Man kann die Farbe auch als *Hexadezimalwert* angeben, dabei wäre 00_{hex} = 0% und FF_{hex} = 100%.

Das Gegenteil dieser *additiven* Mischung ist die *subtraktive* Farbmischung. Dies ist das *CYM*-System. Dabei steht *C* für *Cyan*[20], *Y* für *Yellow*[21] und *M* für *Magenta*[22]. Mischt man alle diese Farben zusammen, sollte sich ein *Schwarz* ergeben. Dieses *CYM*-System wird im Druckbereich verwendet und gelegentlich um *Schwarz* zum *CYMK*-System[23] erweitert. Dann braucht man nicht für das häufig verwendete Schwarz immer alle 3 Farben übereinander zu drucken.

CYM findet auch im klassischen Fotobereich Verwendung und pflanzt sich folglich auch in der Bildbearbeitung mit z.B. *Photoshop* fort. Zunächst haben Farbnegativfilme normal 3 sensible Schichten. Wer wundert sich jetzt, dass diese Farben jetzt *CYM* sind? *Fuji* bietet auch mit dem *X-TRA* einen Film mit einer zusätzlichen Schwarz/Weiss-Schicht an um mit der Erweiterung zum *CYMK* den Kontrast zu erhöhen. Auch wird mit diesen *CYM*-Farben bei der Aufnahme und in der Dunkelkammer gefiltert. Fügt man z.B. ein Gelbfilter hinzu, wird Gelb abgeschwächt, also heller, aber auch reicher an Nuancen und die Komplementärfarbe Blau kräftiger und dunkler. Aus dem gleichen Grund tragen Kampfpiloten aber auch Sonnenbrillen mit *grünen* Gläsern. Das Grün bekommt mehr Nuancen und ein künstliches Tarnnetz lässt sich viel besser von einem frischen, natürlichem Grün unterscheiden.

Wem jetzt alle Farbpaletten durch den Kopf schwirren, sei beruhigt, denn es gibt auch eine vordefinierte *Farbpalette* in Klartext, nur da hat wieder jeder Browser seine eigene. Die Überschneidungen belaufen sich lediglich auf die Grundfarben. Ein *red* oder *black* sollte aber jeder Browser verstehen.

p dies ist der Textblock, der *Paragraph*. Über und unter diesem Block wird Platz, als *Absatz* gelassen. Dies funktioniert aber auch nur richtig, wenn man nicht vergisst die *Klammer* auch wieder *zuzumachen*. Ansonsten beschreiben wir jetzt das Erscheinungsbild der Schrift *innerhalb* dieses *Paragraphs*. Dieses Beispiel erhebt auch keinen Anspruch auf Vollständigkeit.

font-family: Arial,sans-serif; Beginnen wir mit der Familie des Zeichensatzes. In so einem *Paket* befinden sich *Schriftschnitte* (Buchstabenformen) die zueinander passen.

Also es gibt ein Erscheinungsbild für die *normal verwendete Grundschrift*, sowie eine dazu passende *fette Variante* und eine dazu harmonierende *Kursive*, sowie Zwischenformen.

In diesem Fall wollen wir die Schrift *Arial* verwenden. *Arial* ist eine Schrift ohne *Serifen*[24], sollte der anzeigende Computer nicht über den *Font Arial* verfügen, dann soll er irgendeine vorhandene Schrift *ohne Serifen* (sans-serif) verwenden (z.B. *Helvetica*).

font-style: normal; Hier wählen wir aus, ob wir die für uns *normale* Antiqua, oder als *italic* die *Kursive* aus der o.a. *font-family* verwenden wollen. Mögliche Werte für den *font-style* sind also *normal*, *italic* und *oblique*. Bei fast allen Browsern liefern *italic* und *oblique* aber das gleiche Ergebnis und wo nun der Nutzwert dieser Unterscheidung liegt, kann ich auch nicht erklären.

[20] Blaugrün

[21] ein schönes Kanariengelb

[22] das von der *Deutschen Telekom* verwendete Lila

[23] Das K steht für *blacK* um eine Verwechselung mit *Blue* zu vermeiden.

[24] Dies sind die kleinen Häkchen und Striche an den Buchstaben, die es dem Auge erleichtern der Zeile zu folgen. Bei Medien mit schlechter Auflösung, wie einem Bildschirm, können diese Striche aber zu hässlichen Klötzen mutieren. Prinzipiell ist sich die mir bekannte *typographische* Fachliteratur aber darin einig, dass eine Schrift **mit Serifen** – wie dieser Text – **leichter lesbar ist**.

font-variant: normal; Die nächste Unterscheidung in der *font-family*. Hier gibt es die Möglichkeiten *normal* und *small-caps*. Die Einstellung *normal* ist die für uns normale Erscheinung mit *Großbuchstaben* und *Kleinbuchstaben*. Wählt man aber *small-caps*, dann habe wir große Großbuchstaben und etwas kleinere Großbuchstaben für die Kleinbuchstaben.

font-weight: normal; Hier geht es ganz offensichtlich ums Gewicht, also ob wir **magersüchtige, normale** oder **fette** Buchstaben wollen. Mögliche Einstellungen sind: *normal*, *bold*, *bolder*, *lighter*, *100*, *200*, *300*, *400*, *500*, *600*, *700*, *800* und *900* sind. Bei den Zahlen entspricht *500* dem Wert *normal*, *100* leidet an völliger Magersucht und *700* wäre *bold*. Bevor Sie sich aber zu große Hoffnung machen, viele Browser unterstützen es aber nur etwas dünner und dicker neben normal. Alle Zahlenwerte werden einfach zum nächsten möglichen Wert gerundet.

font-size: 12pt; Dies ist die Größe der Schrift. Die Größe kann man in den Masseinheiten *pt* (Points), *pc* (Picas), *in* (Inches), *cm* (Zentimeter), und *mm* (Millimeter) angeben. Üblich in der *Typographie* ist die Angabe in Punkten (pt). Dabei ergeben 72 Punkte (points) einen Zoll (inch). Üblich im Satz eines Buches ist eine 10-Punkt-Schrift, also ein *font-size* von *10pt*. Dieses Skript ist in 11pt gesetzt und auf Papier ist mehr als 12pt nicht üblich. Allerdings kann es am Bildschirm und in Kinderbüchern schon Sinn machen größer zu werden.

h1 Die größte Überschrift.

font-family: Arial,sans-serif; siehe oben unter *p*.

font-style: normal; siehe oben unter *p*.

font-variant: normal; siehe oben unter *p*.

font-weight: normal; siehe oben unter *p*.

font-size: 24pt; Die Schriftgröße ist bereits unter *p* besprochen worden. Neu ist, hier handelt es sich um eine 24-Punkt Schrift. In der *Typographie* stuft man in der Regel nach dem *Goldenen Schnitt* ab und da steckt die $\sqrt{2}$ als Vergrößerungsfaktor drin.

line-height: 36pt; Etwas Neues. Bisher hatte wir mit dem *font-size* die Länge eines Großbuchstabens von seiner Untergrenze bis zu seiner Obergrenze. Jetzt haben wir mit *line-height* die Länge von der Untergrenze eines Großbuchstabens bis zur Untergrenze des darüber liegenden Großbuchstabens. Im Beispiel ist der Buchstabe 24pt groß und der Abstand von Untergrenze bis Untergrenze beträgt 36pt, also das eineinhalbfache des Buchstabens. Dies macht also eine halbe Buchstabenhöhe Platz zwischen den Zeilen. Schon einmal von *eineinhabfachen Zeilenabstand* gehört?

Dies funktioniert natürlich entsprechend genauso gut bei der Definition von *p*.

text-align: left; Hiermit kann man *linksbündigen Text* definieren. Zulässige Werte sind *left* für linksbündig, *right* für rechtsbündig, *center* für zentriert und *justify* für Blocksatz bei dem die Leerzeichen soweit aufgefüllt werden, bis der Text gleichzeitig links- und rechtsbündig ist.

color: green; Die Schriftfarbe. In diesem Fall sind die h1-Überschriften immer grün. Ansonsten gilt das oben gesagte für die Hintergrundfarbe.

```
<STYLE type="text/css">
<!--
/* Dies ist ein Kommentar */

/* Mozilla-Problem mit der 1. Definition, daher hier ein Dummy */
dummy { font-style: normal;}

/* Jetzt zu den richtigen Definitionen */
body    {margin-left: 20mm;
        margin-right: 20mm;
        margin-top: 20mm;
        margin-bottom: 35mm;
        background-color:rgb(90%,60%,90%);
    }

p   {   font-family:Arial,sans-serif;
        font-style: normal;
        font-variant: normal;
        font-weight: normal;
        font-size: 12pt;
    }

h1   {   font-family:Arial,sans-serif;
        font-style: normal;
        font-variant: normal;
        font-weight: normal;
        font-size: 24pt;
        line-height: 36pt;
        text-align: left;
        color: green;
    }

h2   {   font-family:Arial,sans-serif;
        font-style: normal;
        font-variant: normal;
        font-weight: normal;
        font-size: 17pt;
        line-height: 26pt;
    }
//-->
```

Abbildung 5: CSS-Datei: *meine.css* in der ersten Ausbaustufe.

3 Aufgaben

1. Nachdem Sie dieses Skript verstanden haben, bringen Sie bitte Ihre erste eigene html-Seite *zum Laufen*.

2. Erweitern Sie Ihre html-Seite um ein *Cascading Style sheet*.

3. Entwickeln Sie einen *Spieltrieb* und verändern Sie verschiedene Attribute. Versuchen Sie sich vorher vorzustellen, wie die Sache damit aussieht und kontrollieren Sie dies, indem Sie sich die Seite mit den veränderten Attributen mit verschiedenen Browsern ansehen.