Analysis of

Cascading Style Sheets

Seminar paper

Author: Thomas Piffer

Student-ID: 0551068

IS-Projektseminar SS 2010

Wirtschaftsuniversität Wien

Advisor Prof. Dr. Rony G. Flatscher

Table of content

List of figures....4

List of tables....5

1 Basics of Cascading Style Sheets (CSS Level 1)....6

1.1 Definition....6

1.2 History....7

1.3 Syntax....8

1.3.1 Selectors....8

1.3.1.1 Type selector....9

1.3.1.2 Class and ID selector....9

1.3.1.3 Pseudo-classes and pseudo-elements....10

1.3.2 Implementation of CSS in HTML....11

1.3.2.1 Implementation in the head-element....11

1.3.2.2 Implementation within the HTML-element....12

1.3.2.3 CSS in separate data file....14

1.3.3 Box model....16

1.3.4 Units....18

1.3.4.1 Length Units....18

1.3.4.2 Percentage Units....19

1.3.4.3 Color units....19

1.3.4.4 URL....20

2 CSS Level 2 21

2.1 Overview of updates and modifications....21

2.2 The concept of media types....22

2.3 Aural Style Sheets....25

2.3.1 Browser support....25

2.4 New selectors....26

2.4.1 Universal selector....26

2.4.2 Attribute selector....26

2.4.3 Pseudo classes....27

3 CSS Level 2.1....29

3.1 Important updates....29

3.1.1 Conformity....29

3.1.2 Calculation of specification....30

3.1.3 Aural style sheets....30

4 CSS Level 3....31

4.1 Introduction....31

4.2 Development status codes....31

4.3 High priority modules....32

4.4 Medium priority modules....34

4.5 Low priority modules....36

4.6 Browser support....37

5 References....39

List of figures

Figure 1: CSS example [W3WP10].....6

Figure 2: CSS rule.....8

Figure 3: Implementation within the head-element.....11

Figure 4: Implementation within HTML-elements.....12

Figure 5: Priority example HTML source code.....13

Figure 6: Priority example browser output.....13

Figure 7: CSS in separatley data file, HTML Document.....14

Figure 8: CSS in separately data file, CSS data file.....15

Figure 9: CSS in separately data file, Browser output.....15

Figure 10: Box model [W3C96].....16

Figure 11: Source code of a HTML-Document, Box model.....17

Figure 12: Browser output, Box model.....17

Figure 13: CSS color units.....19

Figure 14: Relationship between media groups and media types [W3C98].....24

Figure 15: Current development status medium-priority modules.....34

Figure 16: Development status low-priority modules [W3C10].....37

List of tables

Table 1: Type selector.....9

Table 2: Class and ID Selector.....9

Table 3: Pseudo-classes.....10

Table 4: Pseudo-elements.....10

Table 5: Universal selector.....26

Table 6: Attribute selector.....27

Table 7: New pseudo classes..... 28

Table 8: Browser prefixes.....38

1 Basics of Cascading Style Sheets (CSS Level 1)

1.1 Definition

CSS is the abbreviation of cascading style sheets and is one of the most popular style sheet language world wide. It's strongly used to design a document, which is written with a markup language. A markup language is a system for annotating a text in a way that is syntactically distinguishable from that text [W3WP10]. Famous markup languages are XML, HTML and XHTML. CSS is strongly used in combination with HTML and XHTML for the web design.

The main idea of CSS is the possibility to divide content and the presentation of the content (design) of a document. On the one hand it's possible to split the work, e.g. the graphic artist is only responsible for the design and eventually for the style sheet and the editor is responsible only for the content and has to waste no thoughts on the style or the presentation of the content. On the other hand the splitting of design and content improve the accessibility and reduces complexity in the structure and thus also of the source code.
Another main advantage is that CSS has the possibility to design content for several output systems like monitors, PDA, mobile phones, smart phones and so on. It also supports handicapped people with voice support displayed documents or with braille-systems.

Figure 1: CSS example [W3WP10]

1.2 History

The birth of style sheets was the publication of the ISO-standard technology Standard Generalized Markup Language (ISO 8879:1986 SGML) in the 1970s which has the mission to define generalized markup languages (for definition look at 1.1.) for documents.
Before the development of CSS, web designers used HTML for displaying content too. But the growth of HTML makes also a wider variety of stylistic capabilities possible and HTML documents, especially the source code becomes more and more complexity, was unreadable and the users had less control over the displayed content in the web.

To confront with this problem, the World Wide Web Consortium (short W3C), which is the main international standards organization of the World Wide Web, chose two of nine proposals of style sheet languages, thereunder Cascading HTML Style Sheets (CHSS) and Stream-based Style Sheet Proposal (SSP). The two developers Håkon Wium Lie, responsible for CHSS, and Bert Bos, responsible for SSP, worked together and finally developed the CSS standard, which was presented at the “Mosaic and the Web” conference in Chicago in 1995.
In principle CSS is similar to CHSS but the 'H' (stands for HTML) was removed to apply CSS to other markup languages beside HTML.

The W3C took an interest in the development of CSS, founded a CSS Working Group and finally, by the end of 1996, the CSS level 1 recommendation was published and thus CSS became official.
The first update of the CSS level 1 recommendation was the CSS Level 2. It was published on May 12 1998 and included some topics that were not mentioned in the CSS level 1 recommendation.
The next step was the start of the actual development of CSS, namely CSS Level 3. It started in 2000 and the development is still running.

1.3 Syntax

The Syntax of a cascading style sheet is very simple and the structure is similar to other programming languages. It uses keywords to define the names of several style properties. The following figure should give a pictorially overview of the CSS syntax.

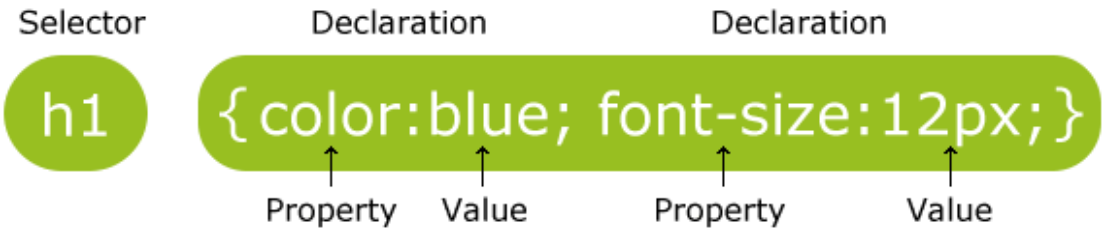

Figure 2: CSS rule

This graphic shows one correct rule of a cascading style sheet. In most cases one cascading style sheet consists of a list of rules. Every rule consists of one or more selectors and declarations. There are several types of selectors, which will be described in this paper. The declaration block begins and ends with a curved clamp. Within there are a property, a colon (:), a value and then a semicolon (;).

1.3.1 Selectors

A selector selects the element, which will be formatted by a CSS rule. In CSS there are several selectors available. We distinguish three various main types of selectors in CSS Level 1: type selectors, class selectors and ID selectors. Each selector type has its own purpose in CSS.

1.3.1.1 Type selector

Selector	Designation	Importance
E	type selector	element E

Table 1: Type selector

The type selector addresses all elements with their names and thus all rules will be applied to the selected elements. It does not matter where the element is positioned in the document and which class it belongs to.

1.3.1.2 Class and ID selector

Selector	Designation	Importance
E.class	Class selector	Element E of class class
E#identifier	ID-selector	Element E with ID identifier

Table 2: Class and ID selector

In focus of web design it is possible that elements often occur on other sites but they should always looks similar on each site. In that case there is the possibility to summarize all equal elements in one class or ID with one common name. And exactly this name is the class or ID selector. Condition of the choice of the class/ID name is that the name doesn't contain spaces or special characters and has to begin with a point.

1.3.1.3 Pseudo-classes and pseudo-elements

Pseudo classes and elements define features which don't deduce from the document root tree and are also not visible there and in the source code. They are no existing classes but classes we make believe. Examples for pseudo classes are the appearance of links (not visited links looks different to visited links) or the first letter of a paragraph. The following pseudo classes are available in CSS Level 1 (mainly use with the *a-element*):

Pseudo-class	Description
:link	For links of not visited sites
:visited	For links of visited sites
:active	For active links

Table 3: Pseudo-classes

Pseudo-elements are used to address sub-parts of elements, while pseudo-classes allow style sheets to differentiate between different element types [W3C08]. The following pseudo elements were defined in CSS Level 1:

Pseudo-element	Description
:first-line	Selects only the first line of a paragraph or marked element
:first-letter	Selects only the first letter of a paragraph or marked element

Table 4: Pseudo-elements

1.3.2 Implementation of CSS in HTML

There are three different methods to integrate CSS in a HTML data file. Each method has its own advantages and it is only a subjectively decision which method will be used.

1.3.2.1 Implementation in the head-element

The first method is to define the CSS formats directly into the head-element of a HTML data file. With *<style...> … </style>* you have to define the area for the CSS definitions. In the beginning <style>-element it's necessary to indicate the MIME-Type.

MIME-Types (Multipurpose Internet Mail Extensions) come originally from E-mails with attachments. The mainly use of MIME-Types in E-mail with attachments was to describe which data type the next part of the E-mail (e.g. text of E-mail and attached ZIP-data file) is. Nowadays several HTML elements have attributes which expect an assigned MIME-type, like a (*type*), form (*accept and anctype*), input (*accept*) etc.

For CSS the MIME-Type is *type="text/css".* Between the beginning and the ending of the <style>-element, the central CSS formats definitions were given. The following example of a HTML-data file shows the above content again.

Neues Dokument

```
1  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">
2  <html>
3  <head>
4  <title>Integration of CSS in the head area</title>
5  <style type="text/css">
6  h1 { color:red; font-size:48px; }
7  </style>
8  </head>
9  <body>
10 <h1>48 pixel and red!</h1>
11 </body>
12 </html>
```

Figure 3: Implementation within the head-element

1.3.2.2 Implementation within the HTML-element

The integration within the HTML-elements is another method to integrate CSS in HTML. With this method it is possible to format single HTML-elements in the HTML-document.

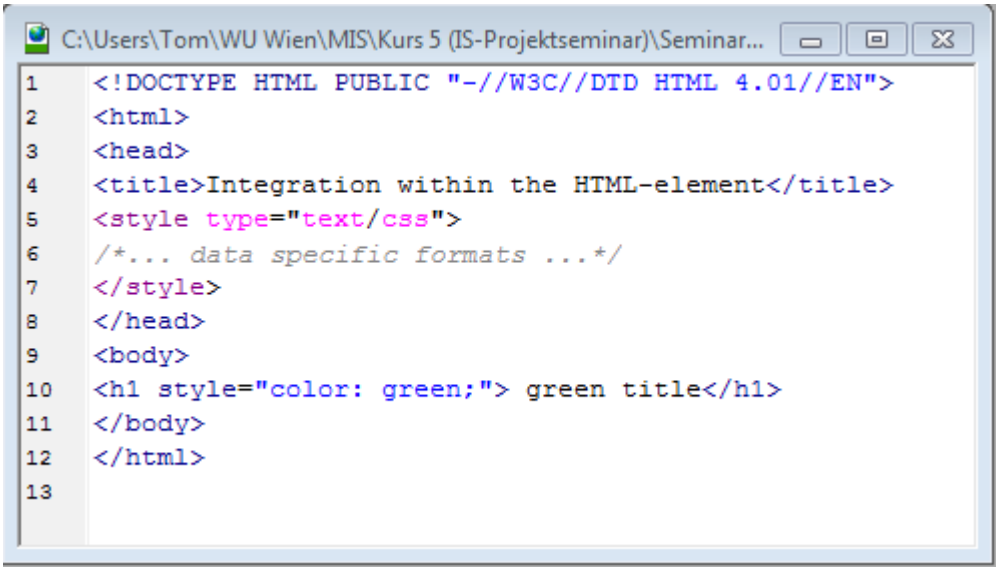

Figure 4: Implementation within HTML-elements

With the attribute *style* at the beginning of the HTML-element *<h1 style=...>* there is the possibility to notate or to define the CSS format for exactly this HTML-element. Important is that this CSS instruction only apply to the HTML-element which is described with the *style*-element. In the graphic above only the HTML-element *<h1>* and all his inherited elements are affected by the CSS instruction.

Main use of this method is to format some HTML-elements different to other elements which were formated from an external CSS data file. Another possibility to bring this method into action is the rare use of CSS. For example web developers who mainly dispense with CSS but need CSS for some exceptions, will prefer this method.

Some points must be respected for the use of the integration of CSS in the HTML-element. All HTML-elements must be conform to HTML-4.0 and have a start and end-tag. Furthermore all CSS definitions in the HTML-element have more priority than CSS definitions outside, for example in the head area of the HTML document.

The two figures below should illustrate this more detailed.

C:\Users\Tom\WU Wien\MIS\Kurs 5 (IS-Projektseminar)\Seminar...

```
1   <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">
2   <html>
3   <head>
4   <title>Integration within the HTML-element</title>
5   <style type="text/css">
6   h1 {color:red;}
7   </style>
8   </head>
9   <body>
10  <h1 style="color: green;"> green title</h1>
11  </body>
12  </html>
13
```

Figure 5: Priority example HTML source code

The source code of this HTML document shows two methods of the integration of CSS in HTML documents. In the head area the CSS formats are defined for all HTML-elements of the type <h1>. The only CSS instruction in the head-element for <h1> is the font color which should be red. Due to that rule, that all CSS instructions, that are notated within the HTML-elements, have priority before the CSS instructions in the head-element for the same HTML-element, the content within the HTML-elements <h1> follows to the CSS formats in the HTML-element.

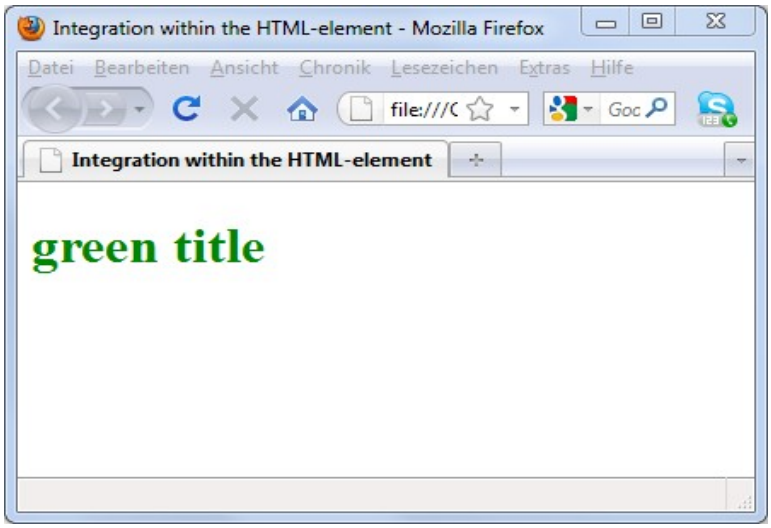

Figure 6: Priority example browser output

1.3.2.3 CSS in separate data file

In most cases web applications or web pages consist of more HTML documents, especially for each web page one HTML document will be needed. The use of the methods where CSS instructions are implemented in the head element or in a HTML-element are in view of maintenance a very time intensive and also a complicated working and the error rate is very high. CSS offers the possibility to format all HTML documents from a separately CSS data file. The big advantage is that all CSS instructions for the HTML document are collected in the CSS data file what makes it possible to change some CSS instructions that have impact on all HTML elements that were selected centrally in the CSS data file and not in every HTML document separately. Reducing the time expenditure and the complexity, this method is the effective of creating bigger web applications or web pages based on HTML.

The approach for this method is to set in every HTML document a <link>-element to reference to the CSS data file in the head area. Within the <link>-element there must be the attributes *rel="stylesheet" type="text/css".*

The first two figures show the source code of the HTML document and the CSS data file.

```
1   <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">
2   <html>
3   <head>
4   <title>CSS in separately data file</title>
5   <link rel="stylesheet" type="text/css" href="separat.css">
6   </head>
7   <body>
8   <h1>This is a green Title</h1>
9   <p> And this should be the text of this title!</p>
10  </body>
11  </html>
```

Figure 7: CSS in separately data file, HTML-Document

As you can see the <link>-element is positioned in the head area of the HTML document with the specific reference to the CSS data file. There are no other CSS instructions within the HTML document.

```
1   body{font-family: Arial, sens serif;}
2
3   h1 {font-size: 20px; color: green;}
4   p {font-size: 12px;}
```

Figure 8: CSS in separately data file, CSS data file

In the separate CSS data file there are three instructions for the content design. In line one there are the general font style instructions for all elements in the HTML document. If some elements within the body-element have no own instructions they get the definitions like the hole body-element, i.e. only the font-family is defined by arial, sens serif in the CSS data file above.

The <h1>-element which marks the title of the content has the property to be green and the size will be 20px. All content within the <p>-element only have the size 12px. As you can see below the browser shows the HTML document in the following style:

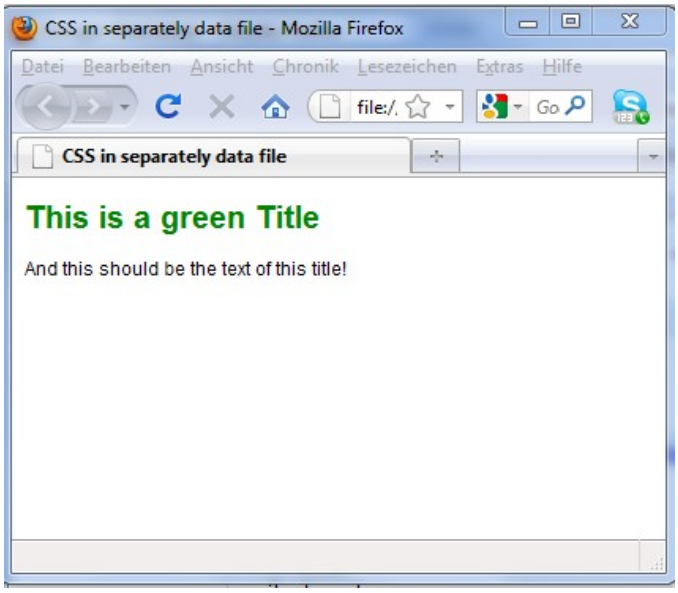

Figure 9: CSS in separately data file, Browser output

1.3.3 Box model

For the positioning of objects, CSS uses the box model. The box model components are the width, the padding, the border-width and the margin. To calculate the overall width of an object or element it's only necessary to add all components. The figure below represents the box model of CSS:

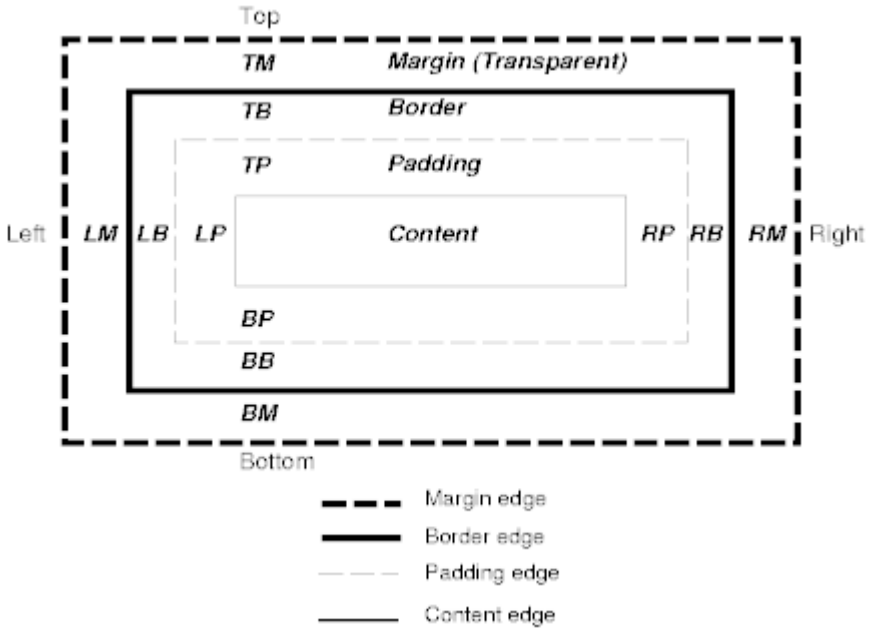

Figure 10: Box model [W3C96]

CSS Level 1 assumes a simple box-oriented formatting model, as you can see in the figure above. Each formatted element results in one or more rectangular boxes. All boxes have a core content area with optional surrounding padding, border and margin areas [W3C96]. Important is that the padding area uses the same background as the element itself. The settings for the color of the border area will be set with the border properties and the margins are always transparent what has the result that the parent element will shine through.

We distinguish several forms of formatted elements: block-level elements, list-item elements, inline elements and replaced elements.

Elements with a *display* value of 'block' or 'list-item' are block-level elements [W3C96]. That means that every element gets a box with its own dimensions. List-item elements are also block-level elements but preceded by a list-item marker. The type of the marker is determined by the 'list-style' property.

Inline elements are no block-level elements what means, that such an inline element can share line space with other elements.

Replaced element is an element which is replaced by content pointed from the element. E.g., in HTML, the img-element is replaced by the image pointed to by the src-attribute. [W3C98]

To illustrate the definitions of formatted elements, you can see below some visually examples:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html>
<head>
<title>Box model</title>
<style type="text/css">
UL {background: red; margin: 12px 12px 12px 12px; padding: 12px 12px 12px 12px;}
LI {color: white; background: blue; margin: 12px 12px 12px 12px; padding: 12px 12px 12px 12px;}
EM {display: inline; color: red;}
p {display: block; color: blue;}
</style>
</head>
<body>
<h1> Block-level elements </h1>
<UL>
  <LI> 1st element of list
  <li> 2nd element of list
</ul>
<h1> Inline elements </h1>
<p> What a beautiful
  <em> and sunny </em>
  day
</p>
</body>
</html>

```

Figure 11: Source code of a HTML-Document, Box model

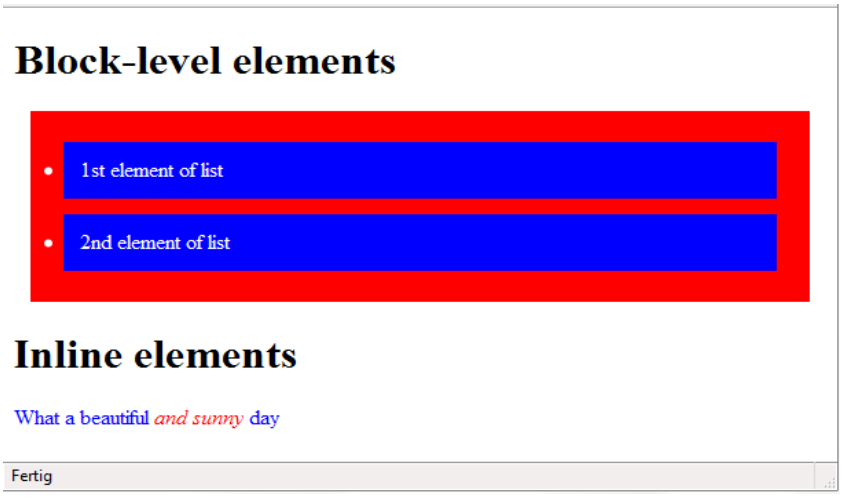

Figure 12: Browser output, Box model

1.3.4 Units

1.3.4.1 Length Units

At the beginning and to illustrate what's the content of this paragraph, here is an example for a length unit:

*H1 {margin: **0.5em**}*

The format of a length value is an optional sign character ('+' or '-', with '+' being the default) immediately by a number (with or without a decimal point) immediately followed by a unit identifier (a two-letter abbreviation) [W3C98].

There are some properties that allow negative length units, but this can complicate the formatting model. We distinguish two types of length units: relative and absolute. Relative units specify a length relative to another length property [W3C98]. Advantage of relative units is that they will scale more easily from one medium to another (e.g. from a computer display to a smart phone). In CSS the following relative units will be supported:

- *ems, the height of the element's font, e.g. 0.5em;*
- *ex, x-height ~ the height of the letter 'x', e.g. 1ex;*
- *pixels, relative to canvas, e.g. 12px;*

The use of absolute units is only useful when the physical properties of the output medium are known [W3C98]. The following absolute units will be supported by CSS:

- inches, e.g. 0.5in;
- centimeters, e.g. 3cm;
- millimeters, e.g. 4mm;
- points, e.g. 12pt;
- picas; e.g. 1pc;

1.3.4.2 Percentage Units

The format of a percentage value is an optional sign character ('+' or '-', with '+' being the default) immediately followed by a number (with or without a decimal point) immediately followed by '%' [W3C98]. They are always relative to another value, e.g. a length unit.

The CSS rule *H1 {line-height: 89%;}* means that the line-height is relative to the font-size of the H1 element.

1.3.4.3 Color units

Color units can be specified by keywords or with a numerical RGB specification. In CSS there are several keywords for colors. The list of keywords of colors in CSS defines the following color names: aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, purple, red, silver, teal, white and yellow. These 16 colors are taken from the Windows VGA palette and their RGB values are not defined in the specification of CSS [W3C98]. Furthermore RGB color model is allowed to specify a color in CSS. Following examples should give a visually understanding:

```
1
2
3   H1 {color: white; background: black;}
4   /* this is an example for define the color with keywords */
5
6   H2 {color: #f00;} /* RGB model for #rgb */
7   H3 {color: #ff0000;} /* RGB model for #rrggbb */
8   H4 {color: rgb(255, 0, 0);} /* RGB model for integer range 0-255*/
9   H5 {color: rgb(100%, 0, 0);} /* RGB model for float range 0.0%-100.0% */
```

Figure 13: CSS color units

1.3.4.4 URL

The format of an URL value is 'url('followed by optional white space followed by an optional single quote (') or double quote (“) character followed by the URL itself followed by an optional single quote (') or double quote(“) character followed by optional whitespace followed by ')' [W3C96].

URL will be used for links and references to other external data, e.g. background images.

2 CSS Level 2

2.1 Overview of updates and modifications

The following functions are new to CSS Level 1 which are about

- concept of media types,
- the 'inherit'-value for all properties,
- paged media,
- aural style sheets,
- several internationalization features, including list numbering styles, support for bidirectional text, and support for language-sensitive quotation marks,
- extended font selection mechanism, including intelligent matching, synthesis, and download able fonts, also, the concept of system fonts has been introduced, and a new property, 'font-size-adjust', has been added,
- tables, including new values on 'display' and 'vertical-align',
- relative and absolute positioning, including fixed positioning,
- new box types: compact and run-in,
- the ability to control content overflow, clipping, and visibility in the visual formatting model,
- the ability to specify minimum and maximum widths and heights in the visual formatting model,
- an extended selector mechanism, including child selectors, adjacent selectors, and attribute selectors,
- generated content, counters and automatic numbering, and markers,

- text shadows through the new 'text-shadow' property,
- several new pseudo-classes, :first-child, :hover, :focus, :lang,
- system colors and fonts,
- cursors,
- and Dynamic outlines [W3C98].

To discuss and analyze all new functions in CSS would be beyond the scope of this paper. This paper will only analyze the bigger and more important functions according to the introduced functions in the part of CSS 1. These will be:

- the concept of media types,
- aural style sheets,
- new pseudo-classes and
- the new selectors.

2.2 The concept of media types

Nowadays there are a few different media types that could present documents from the web. Mobile phones or desktop pc's, representation on the screen or on paper, speech synthesizer or a braille device etc., they all can present documents, but the representation of each media type is different. The main difference or the main problem is the screen solution or dimensions of each media type. Mobile phones have a smaller screen solution than notebooks, documents look good on screen but on paper they look bad, and this has the implication that there is a strong need for representations modification for each media type. To sum up the necessity of different properties is demanded in CSS. In CSS Level 1 there was no possibility to modify documents for different media

types. In CSS Level 2 the media types concept will enable this media type modification.

To specify the current or used media type there are two ways to implement this in style sheets:

- Specification of the target medium with the @media or @import rules
- Specification of the target medium within the document language [W3C98]

The implementation of the @media rule is the most used variant. This rule specifies the target media types of a set of statements [W3C98]. The advantage of this rule is that the @media construct allows style sheets rules for various media in the same style sheet. The following names for media types will be used and reflected in CSS:

- **all**: represents all devices
- **braille:** braille system for people with visual impairment
- **embossed:** for paged braille printers
- **handheld:** for handheld devices which has typically small screen and screen solution (mobile phones)
- **print:** for paged material and for document in print preview mode
- **projection:** represents presentation in a projected size (projectors).
- **screen:** for computer screens
- **speech:** for speech synthesizers (aural css)
- **tty:** for media using a fixed-pitch character grid (teletypes, or portable devices with limited display capabilities)
- **tv:** for television-type devices (typically for this type are low solution, color and limited scroll ability screens)

All the listed media types are exclusive which means that a browser only supports one media type by rendering a document. So it's not possible to represent a document in the media type screen and print. Also important to know is that each browser or user agent use different media types on different canvases. Unknown media types will be not presented in the user agents.

To know which media type property applies to, all media types are specified in groups. In CSS 2 the following media groups are available:

- continuous or paged
- visual, audio, speech, or tactile
- grid or bitmap
- interactive or static
- all

The following table shows the grouping of the media types:

Relationship between media groups and media types

Media Types	Media Groups			
	continuous/paged	**visual/audio/speech/tactile**	**grid/bitmap**	**interactive/static**
braille	continuous	tactile	grid	both
embossed	paged	tactile	grid	static
handheld	both	visual, audio, speech	both	both
print	paged	visual	bitmap	static
projection	paged	visual	bitmap	interactive
screen	continuous	visual, audio	bitmap	both
speech	continuous	speech	N/A	both
tty	continuous	visual	grid	both
tv	both	visual, audio	bitmap	both

Figure 14: Relationship between media groups and media types [W3C98]

2.3 Aural Style Sheets

Aural style sheets make a representation of documents for blind or visual impaired people possible. Also to illiterate people is such a representation of content a very big alleviation.
The text of a document will be represented by an speech synthesizer. With aural style sheets it is possible to modify a few of properties due to the speech or sound representation.

Aural style sheets properties or the representation of document with a synthesizer supports three-dimensional physical space. So it is possible to represent the document in the structure in which the content is represented by the screen. For example: if there is a graphic or a text at the left upper side of the screen, the speech will come from exactly this position. Furthermore this style sheets also support a temporal space and properties of the synthesized speech. Temporal space means that sounds will be before, during or after words played to another sound. With the properties of the synthesized speech, voice type, frequency and inflection etc. can be modified.

2.3.1 Browser support

One of the biggest problems of aural cascading style sheets is the limited or poor support of user agents and such limited support makes aural style sheets useless. Current screen readers like JAWS are furthermore using their own non-CSS-algorithm to analyze the words of a document.
And due to the fact that there are screen readers where you can modify the speed of the speech or the frequency etc. would make the use of aural style sheets marginal.

The only user agent which supports the aural cascading style sheets (ACSS) is Opera and Firefox with the plug in Fire Vox. All other browsers or user agents have no support for the ACSS.

2.4 New selectors

CSS Level 1 distinguishes three types of selectors. Selectors are markers to address CSS rules with HTML-elements. This three selectors are the type selector, the id selector and the class selector. In CSS Level 2 there are two selectors additional to the three selectors from CSS Level 1. These are the universal selector and the attribute selector. This two new selectors will be described in more detail in the following part.

2.4.1 Universal selector

The universal selector has similarity with the type selector. Certainly the universal selector doesn't choose an element with a special type but it choose an element of any type. In general we can say that this selector makes no selection in any way. With the „*“ we can address all elements inclusively the <html> <title> <body> and <table> elements.

Selector	Designation	Importance
*	Universal selector	All elements

Table 5: Universal selector

2.4.2 Attribute selector

CSS 2 allows authors to specify rules that match elements which have certain attributes defined in the source document [W3C09] There are four ways to define an attribute selector:

- ***[attribute]*** defines if the element has the specific attribute.
- ***[attribute=value]*** defines if the element has the attribute with a specific value.
- ***[attribute~=value]*** defines if the element has an attribute with a space separated items that defines the value.

- ***[attribute|=value]*** defines if the element has an attribute with a hyphen separated item that defines the value.

An important thing in this case is that user agents or browser from Internet Explorer 6 and lower do not support this type of selector.

Selector	Designation	Importance
E[foo]	Attribute selector	Element E with the attribute foo
E[foo=”bar”]	Attribute selector	Element E with the attribute foo and the value bar
E[foo~=”bar”]	Attribute selector	Element E with the attribute foo, with a space separated list consisting the value bar
E[foo|=”bar”]	Attribute selector	Element E with the attribute foo with a hyphen separated list beginning with the value bar

Table 6: Attribute selector

2.4.3 Pseudo classes

As in chapter 1.3.1.3. already described pseudo-classes are similar to the class selectors. The difference here is that pseudo-classes are no part oft the source code or of the document object model from CSS. In addition to the already described pseudo-classes from CSS 1 there are now three new pseudo-classes due to the recommendation of CSS Level 2. These are:

Pseudo-class	Description
:first-child	Always choose the first child of the document root tree.
:focus	Marks the element which has now the keyboard-focus, e.g. an entry field
:lang	This pseudo-class is similar to the attribute selector [lang|=”language”] to choose elements in the defined language or with the defined value

Table 7: new pseudo-classes

3 CSS Level 2.1

CSS 2 was recommended in 1998. Since the specification was public some experience have been collected. The most of experience will be flow in CSS Level 3 specification but in some parts the implementation and interoperability is hindered by CSS 2 with the actual standard.

Due to this fact CSS 2.1 tries to solve these described problems in a new specification and the following points are main parts of the specification from CSS 2.1:

- maintainance of widely implemented parts of CSS 2
- inserting of all public CSS-2-errata
- delete the parts of CSS which are useless
- parts which will be in CSS Level 3 specification have been deleted
- some new values have been inserted if the experience shows that they are useful

But what means this in practically? To answer this question the following details should give an overview over all important updates of CSS 2.1.

3.1 Important updates

3.1.1 Conformity

Conformity stands for the own choice of a user-style-sheet. The user should have the possibility to change properties of the displayed website for their own output device (pc, handheld, braille-system). This changes enormously the basic idea of CSS, namely the separation of content and presentation or design and of course this contributes the progress of web accessibility.

3.1.2 Calculation of specification

Selectors which are specified in the source code have a higher specificity than all other selectors no matter of the implementation. The specificity was described by a chain of three numbers. Specified selectors in the source code with the style-element had the specificity of 1-0-0. This rule was problematic because in CSS 2 one ID-selector has the same specificity like a style-rule in the source code [W3KL10]. Problems with the cascade or invisible inline-styles were the result. To solve this problem the specificity in CSS 2.1. will be described by a chain of four numbers like 1-0-0-0.

3.1.3 Aural style sheets

As mentioned above at the chapter 2.1.2. aural style sheets are no longer component of the CSS 2.1 recommendation and are in CSS 2.1 only part of the appendix and thus have an informative character [W3KL10]. Also the values of aural style sheets meet the same fate. But the development of this kind of media type doesn't stop here. In CSS 3 there will be an own specification of the media type called “speech”. In CSS 2.1. the media type “aural” is deprecated.

4 CSS Level 3

4.1 Introduction

The start of the development of CSS Level 3 is the year 2000. Since that time CSS level 3 is still in development. But we can't say CSS Level 3 is in development due to the fact that CSS Level 3 is developed by a module system. This means that there is no common specification for the hole CSS 3 but only for each module. This new module system entails some advantages for the user agents or web browsers. Each user agent has the possibility to pick up only the modules that are supported by the user agent. For example: a PDA with black-white display doesn't support the module with the color properties completely.

For a better and controlled development the W3C defines three categories for development of CSS Level 3 where each module is allocated to one category. These three categories are defined by the priority of the development:

- high priority
- medium priority
- low priority

4.2 Development status codes

Each category has one of four different status codes. This status codes are used by the CSS working group, from least to most stable:

- **Working Draft**
  Is a maturity level for work in progress. In practice the working draft is a document which is published to review by the community (W3C members, the public, other technical organizations)

- **Last Call**
  The last call specifies the deadline for review comments; identify known dependencies and solicit review from all dependent Working Groups and solicit public review [W3C05].

- **Candidate Recommendation**
  This is a document which has been widely reviewed and satisfies the technical requirements.

- **Proposed Recommendation**
  The Proposed Recommendation is a mature technical report that, after wide review for technical soundness and implementability, W3C has sent to the W3C Advisory Committee for final endorsement [W3C05].

- **Recommendation**
  The final specification which has received the endorsement of W3C Members and the Director.

In the following each category with their assigned modules will be presented more detailed.

4.3 High priority modules

All assigned modules in this category have the highest priority in development. This means that the W3C is very interested in this modules and their advancing development. The following modules are assigned to this category:

- CSS Level 2 Revision 1
- Selectors
- CSS Mobile Profile 2.0
- CSS Marquee

The module CSS Level 2 Revision 1 is a revised recommendation respectively will be a revised specification because the actual status code is on “candidate recommendation”. As already mentioned the CSS Level 2 Revision 1 corrects all relevant errors of the CSS Level 2 recommendation and adds now a few requested features for CSS 3, which have already been widely implemented [W3C10]. In sum the CSS Level 2 Revision 1 is an actual “snapshot” of the hole CSS usage and represents the complete functionality of CSS.

CSS describes how to design a document on various media with the different media types. But if documents are laid out on the media type or the content is too large for the screen or the given area CSS allows to edit how the overflow is displayed. And one possibility to manage this problem is the “marquee” effect. This effect animates the content whereby the content moves automatically back and forth. The CSS Level 3 module CSS Marquee provides the settings to control that effect. The actual status of this module is “candidate recommendation”.

The status code of the module CSS Mobile Profile 2.0 is currently “candidate recommendation”. Furthermore this is a module with a high priority. In general this module defines a subset of CSS Level 2.1 and is therefore a type of baseline for interoperability between the implementations of CSS at the one hand and the constrained devices (e.g. mobile phones) on the other hand [W3C08]. The subset consist of some selectors, @-rules and properties which are important for this devices.

The module Selectors is an update of the chapter selectors in CSS 2.1. So the main task of selectors is the same. This module is now the most advanced module in the CSS Level 3 development. The current Status code is “proposed recommendation” and the next level of status codes will be the final called “recommendation”. The main changes to CSS 2 are:

- list of basic definitions has been changed
- namespace component in type selectors, universal selectors and attribute selectors is allowed
- Introduction of a new combinator

- new simple selectors including substring
- new pseudo-elements

4.4 Medium priority modules

The most development modules are categorized in the medium priority category. The following table gives an overview of all medium-priority modules an their actual development status codes:

Working Draft:
Values and Units
Cascading and Inheritance
Text
Generated Content for Paged Media
Fonts
Basic Box Model
Template Layout
Speech
Grid Positioning
Flexible Box Layout
Image Values
2D Transformation
3D Transformation
Transition
Animations

Last Call:
Snapshot 2007
Paged Media
Print Profile
Backgrounds and Borders
Color

Candidate Recommendation:
Namespaces
Ruby
Multi-column Layout
Queries
Basic User Interface

Figure 15: Current development status medium-priority modules

This paper gives in the following part an insight in the significant modules from the medium-priority of CSS Level 3.

The multi-column layout module describes, as the name already implies, the multi-column layout in CSS. That means that content of an element is to be laid out in multiple columns without scribing a complex source code. The usage of this module is primary to split up some text into adjacent columns and not primary to replace tables. Main benefit of this module is, that the text can flow from one column to another and the number of columns can vary depending on the size of the canvas [W3C09a]. And this with short and very simple CSS rules. Due to the fact, that this module is a very helpful tool for web design, the development status code is currently “candidate recommendation”.

Aural style sheets are sentenced to death in CSS because of the failing and incorrect implementation and support of the user agents. Nevertheless the Working Group from CSS Level 3 has implied or is implying a module called “speech” which supports and contains properties to specify how a document is rendered by a speech synthesizer [W3C10]. Due to the fact that aural style sheets were not compatible with the Speech Synthesis Markup Language (SSML) the Working group therefore found this module. The Aural Style Sheets in CSS 2 were split into two categories. The one category supported the speech and the other the audio rendering. So the actual module “speech” is a continuation of the category speech from the Aural Style Sheets from CSS Level 2 but have now different values. Due to the fact that this module supports the SSML one hopes a larger support from the user agents. Apart from the speech rendering of a document for the blind and print-impaired communities, there are more advantages and a view more purposes of the CSS Speech. For example the usage for in-car use, industrial and medical documentation systems, home entertainment and to help users learning to read or who have problems at reading [W3C04].

The module “flexible box layout” is a very easy and simple way to solve some usual problems in web design (form layout, page footers, vertical centering, dis-

association of visual flow from HTML flow, etc.) [W3MH10]. This module is in addition to the traditional box model from CSS Level 1 and Level 2. To be more detailed the flexible box model defines how boxes are distributed inside other boxes and how they share the available space. With this module it is easy to create fluid layouts which automatically adapt their own size to the available size of the browser window adapt themselves to the font size. Due to the fact that the module development status code is currently “working draft”, it is not clear how many changes and in which way would this changes impair the future functionality of this of course very useful module in view of web design.

In summary, the actual development status codes of the most modules from the medium-priority category is currently “working draft”. To analyze all this modules would only be a snapshot of the currently situation and thus an inaccurate analysis of the functionality of each module. It is not clear why the development status of some important and useful modules is partially so sluggish. But the most user agents have implemented some useful modules without a valid and final recommendation of the W3C. This topic – implementation of not finished CSS 3 modules in user agents - will be analyzed later in this paper.

4.5 Low priority modules

This part of this paper should only list the remaining modules. All this modules were categorized with a low priority. Significant for this category is the development status code which is widely „working draft“ or the „working draft“ is the upcoming action. This shows that this modules are not so important for the development and also aren't an improvement for the usage. The following graphic lists all modules with low priority:

Low Priority	Current	Upcoming
CSSOM View	Working Draft	Working Draft
CSS Extended Box Model		Working Draft
CSS Object Model		Working Draft
CSS Syntax	Working Draft	Working Draft
CSS Lists	Working Draft	Working Draft
CSS Tables		Working Draft
CSS Reader Media Type	Working Draft	-
CSS Positioning		Working Draft
CSS Generated and Replaced Content	Working Draft	Working Draft
CSS Line Layout	Working Draft	Working Draft
CSS Hyperlink Presentation	Working Draft	Working Draft
CSS `style` Attribute Syntax	Last Call	Candidate Recommendation
CSS Math		Working Draft
CSS Presentation Levels	Working Draft	Working Draft
CSS Aural Style Sheets		Working Draft
CSS TV Profile 1.0	Candidate Recommendation	Proposed Recommendation
Behavioral Extensions to CSS	Working Draft	Working Draft
CSS Introduction	Working Draft	Working Draft

Figure 16: development status low-priority modules [W3C10]

4.6 Browser support

CSS and thus CSS Level 3 is one of the most popular style sheet languages world wide in use. And due to this fact it is important that the user agents support CSS and its functionality. Although CSS Level 3 or the modules are commonly under development, browsers begin to implement some CSS Level 3 modules. To implement these not finished or recommended modules as browser specific extensions, the browser provider marks the properties with prefixes. The following table gives an overview of the browser specific prefixes:

Prefix	Browser	Example
-khtml-	Konqueror	-khtml-border-radius
-moz-	Gecko-based browser: eg. Firefox	-moz-text-shadow
-ms-	Internet Explorer	-ms-background-size
-o-	Opera	-o-box-shadow
-webkit-	Webkit-based browser: eg. Google Chrome or Safari	-webkit-filter

Table 8: Browser prefixes

Important for the implementation of CSS 3 modules is, that browsers have to ignore CSS-properties they don't understand. These prefixes are not valid for all new CSS3-properties but only for the properties which the browser wants to support.

5 References

[W3C96] World Wide Web Consortium: Cascading Style Sheets, level 1. W3C Recommendation 17 Dec. 1996, revised April 2008.
http://www.w3.org/TR/CSS1/

[W3C98] World Wide Web Consortium: Cascading Style Sheets, level 2. CSS2 Spezifikationen. W3C Recommendation 12-May-1998
http://www.edition-w3.de/TR/1998/REC-CSS2-19980512/

[W3C04] World Wide Web Consortium: CSS3 Speech Module W3C Working Draft 16 December 2004
http://www.w3.org/TR/2004/WD-css3-speech-20041216/

[W3C05] World Wide Web Consortium: W3C Process Document. W3C Technical Report Development Process. 2005
http://www.w3.org/2005/10/Process-20051014/tr#RecsWD

[W3C08] World Weide Web Consortium: CSS Mobile Profile 2.0. W3C Candidate Recommendation 10 December 2008
http://www.w3.org/TR/2008/CR-css-mobile-20081210/

[W3C09] World Wide Web Consortium: Cascading Style Sheets Level 2 Revision 1 (CSS 2.1). Candidate Recommendation 08 September 2009.
http://www.w3.org/TR/CSS21/

[W3C09a] World Wide Web Consortium: CSS Multi-column Layout Module. W3C Candidate Recommendation
17 December 2009
http://www.w3.org/TR/2009/CR-css3-multicol-20091217/

[W3C10] World Wide Web Consortium: Cascading Style Sheets Current Work.
http://www.w3.org/Style/CSS/current-work
retrieved on 2010-05-28

[W3KL10] Klaus Langenberg: Änderungen und Ergänzungen in CSS 2.1
http://www.thestyleworks.de/basics/css21-changes.shtml
retrieved on 2010-05-21

[W3MH10] N.N.: Mozilla Hacks, Flexible Box layout
http://hacks.mozilla.org/2010/04/the-css-3-flexible-box-model/
retrieved on 2010-04-22

[W3WP10] N.N.: Wikipedia, Cascading Style Sheets
http://de.wikipedia.org/wiki/Cascading_Style_Sheets
retrieved on 2010-07-08