

## The History of Cascading Style Sheets

When Tim Berners-Lee first established the “World Wide Web” in the late eighties, browsers were only text based and consequential the need of layout options was very limited.

When 1992 the first graphical browsers were introduced, this changed. While before a website could only consist of plain text, there were now pictures, movies, sound etc. all with there own needs on the layout. To satisfy this new needs people used some “dirty” tricks. For example transparent and therefore not visible GIFs as place holders. While the most problems with the layout were solved by CSS some of these tricks are still very common. Especially using HTML-tables not to present content, but design the layout, is still used in some pages.

Of course Cascading Style Sheets weren't the only idea that came up to solve this problem. The two big browser companies “Microsoft” and “Netscape” developed their own extensions to HTML to provide a better control of the layout for their users, others defined their own stylesheet-languages, which were often only build for one browser.



CSS itself was first introduced in 1994 on the “Mosaic and the Web” conference in Chicago, Illinois. The inventor of CSS was Hakon Wium Lie, a Norwegian, which is today technical director at Opera. One year later Bert Bos joined Wium Lie and together they continued the work on CSS. After another presentation in 1995, the World Wide Web Consortium noticed CSS. Together with Wium-Lie and Bos, Steve Pemberton lead a workshop at the w3c, which lead to the first CSS-recommendation in 1996 .

*Hakon Wium Lie*

Only two years later in 1998 CSS 2, the second recommendation, was published.

Although the work on CSS 3 started in 1998 its is not finished till the present day.

### CSS and Browsers

When the **CSS 1** recommendation was published in 1996 no browser had fully implemented it. Both big players in browser business, Microsoft and Netscape, still had a lot of bugs in their browser, which lead a lot of web designers into writing different style sheets for the two browsers. Finally at the beginning of the new millennium Microsoft published the Internet Explorer 5.0 for Macintosh, the first browser which implemented more than 99% of the CSS1 recommendation and therefore is considered the first browser to fully implement CSS1.

Today all mayor browsers claim to fully implement the CSS1 recommendation.

Although CSS 2 was only published two years after CSS 1 there is till the present day no browser which implements the full recommendation. Anyway, a lot of rules specified in CSS 2 can already be used with most browsers.

Mozilla, Opera and Konqueror/Safari are constantly improving their CSS2 support. Sadly they seem to have set different priorities, so the development of the browser is going in different directions. Microsoft is compared to the above mentioned browsers far behind concerning CSS2. With the new version 7.0 of the Internet Explorer, which isn't published yet, they want to catch up with the others.

As some of the rules in this paper are taken from the CSS 2 recommendation, I will mark them and tell you if some browsers don't display them at all or properly.

## How to define rules in CSS

We already heard that it should be possible to define the layout of a documents with CSS. To do this CSS uses one simple scheme.

```
h1 { color:red; }
```

This very simple rule changes the text color of all Elements between opening and closing “<h1>” -Tags into red.

```
selector{definition;}
```

The **selector** defines which elements are effected by this rule. In the example above the selector is “h1”. The **definition** it self is divided into two different parts: **property** and **value**. The property describes how the the objects are changed, in our example the property is “color”, so this rule defines that the text-colors of the objects are changed. The value defines in which way the property is changed. In the example the value is “red”, so all element's property (color) is changed to the value (red), if they are between the two selector tags(<h1>... </h1>). Of course in one rule more than one definition can be included, therefore all definitions are separated by a semicolon.

## Where to define CSS rules

Now we know how we define a simple rule in CSS but where should we define them?  
There are three different possibilities:

### 1. In the HTML-file

You can define CSS-rules directly in the head-area of a HTML-file. Therefore you only have to write

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

and the closing equivalent

```
</style>
```

Between this two tags you can just write down the CSS rules. Including the rule “h1{color:red;}” would change the layout like above discussed in the whole HTML-file.

As this method only affects tags in this HTML-file it should only be used for rules that are specially designed for this file and are only used in it

### 2. Separated CSS-files

CSS although offers the possibility of defining CSS-rules in a separated CSS-file. This file can be written with any text-editor.

The advantages of CSS-files is of course that you can use it in more than one document. Especially when you design websites you want them to look of a piece, so without the possibility to include separated CSS-files you had to copy/paste all the CSS-rules in every file. But separated CSS-files also improve the speed of your website. All new browsers load the CSS-file once while entering the site first and then store them in their cache, so when the user changes the HTML-file the CSS-file doesn't have to be loaded again. Note that pictures included in HTML by CSS are stored in the cache too, which even in times of fast internet connections, improves the speed of a website noticeable.

Separated CSS-files are also included in the head-area. You only need one line to include CSS-files:

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

This statement includes the file “formats.css” into the HTML-file. Note that the ending of a CSS-file has to be “.css”. In a valid CSS-file only CSS-rules and comments are allowed.

### 3. Defining CSS-rules directly at a tag

```
<h1 style="color:red;"> TEXT </h1>
```

Like in the above example you can define CSS-rules directly at a tag. This method should only be used for rules that are only used once in the document. They only affect the elements between the opening tag and the closing.



Note that this rule is only applied if the HTML-tag is conform to the HTML 4.0 specification. Especially tags that didn't need a closing tag like “<i>” need a closing one for the CSS-rule to affect them.

**Using more than one method** of the three in one document can easily lead to conflicts. If two rules with the same selector and the same properties are defined with the same method the one that was defined last is taken.

```
<link rel="stylesheet" type="text/css" href="formats1.css">  
<link rel="stylesheet" type="text/css" href="formats2.css">
```

If there are rules in the two CSS-files with the same selector and the same property the one from “formats2.css” is taken.

If two different methods define rules that would lead to a conflict, rules defined directly at the tag (method 3) are applied before the other two. Rules defined in the head-area with method 1 are taken over rules defined in a separated CSS-file.

### Defining different CSS-files for different output media

With CSS 2 the possibility to define different separated CSS-files for different output media was included. In web design the most commonly used output media beside the screen is the printer.

```
<link rel="stylesheet" media="screen" href="website.css">  
<link rel="stylesheet" media="print" href="print.css">
```

So if the user wants to view your website in his browser the CSS-file “website.css” is used. If he decides to print it the file “print.css” is used.

<i>Media type</i>	<i>Description</i>
media="all"	Applies to all media types.
media="aural"	For synthetic computer voice output.
media="braille" media="embossed"	Braille is used for braille output media, which is a device for blind people. Embossed is used for braille printers.
media="handheld"	Is for output on small devices like self phones or hand helds.
media="projection"	Is used for output on projectors.
media="screen"	Output on a computer screen.
media="tv"	Output on a tv.

### Comments in CSS

Comments in CSS are similar to programming languages like “C”. Comments start with “/\*” and end with “\*/”. Text between this two signs is not interpreted. **Note** that “/” cant be used in CSS!

```
/* Hi i am a comment */
```

## The selector

We already learned that the selector defines which elements are affected by a CSS-Rule, but often you want more than one selector to indicate that a definition should be used. For example we want all our headlines to be in red color. With our knowledge we had to write something like this

```
h1{color:red;}
h2{color:red;}
...
h6{color:red;}
```

But there is a shorter version which does the same

```
h1,h2,h3,h4,h5,h6{color:red;}
```

So you can apply one definition to more than one selector by simply dividing them by commas (“,”).

```
h1 span {color:blue;}
```

Note that this time there is no comma between the two selectors. This means that this rule is only used if the opening “span” tag is between the “h1” tags:

```
<h1> <span> this text is blue </h1> </span>
```

If you have used HTML before you might have mentioned that span is no real HTML-tag. This tag was included with CSS to be used when laying out text, so that you don't have to overwrite all the basic HTML-tags.

There are ways to define the relation between the two selectors even more:

<i>Symbol</i>	<i>Description</i>	<i>Example</i>
h1 * span{...}	This rule is only used if the “span” tag is at least one layer beneath the “h1” rule	<h1> <i> <span> </h1> </i> </span>
h1 > span{...}	This rule is only used if the “span” tag is between the “h1” tag	<b>Not used if:</b> <h1> <span> </h1> </span>
h1 + span{...}	This rule is used when a “span” tag directly follows a “h1” tag.	<h1> </h1> <span> </span>

## Selectors and classes

```
span.red{color:red;}
span.red.blue{color:red; background-color:blue;}
```

In CSS Classes were included. They allow to use one selector for more than one definition. The class name is defined directly behind the selector. A point separates the two. You can define classes consisting out of more than one word by separating each word with a point. **Note**, use descriptive class names to simplify the code.

```
<span class="red"> This text is red </span>
<span class="red blue"> This text is red with blue background </span>
```

Classes are used in HTML by the class property.

```
*.red{color:red;}
```

The universal selector “\*” can be used to create universal classes. This means that these classes can be applied to any HTML-tag. **Note** that you can leave the “\*” out and the rule will have the same effect.

### Individual formats

Another method similar to classes are individual formats.

```
#red{color:red;}
```

To use them in HTML the id-property is used:

```
<span id="red"> This text is red </span>
```

## Font formation

### font-family

To change the font of a text the font-family tag is used. If you have written one website yourself using some fancy font you found on the internet you may have noticed that if you view the website on a pc where the font is not installed the homepage of yours looks really poor. The font-family property can't solve this problem, but you can specify a row of fonts which are all tried and only if none of them is installed on the PC the browser selects one.

```
span{font-family:"Times New Roman", serif}
```

The different fonts are simply divided by a comma. **Note**, you should use “” or “'” around fonts that consist of more than one word.

There are five generic fonts. These are names for a font type and the browser chooses one font of this type. **Note**, you should use a generic font at the last position in the font-family property, because one of every type should be installed on every system.

Name	Description	Example
serif	A font with serifs	Times New Roman
sans-serif	A font without serifs	Arial
cursive	A font similar to handwriting	<i>Script</i>
fantasy	An extraordinary font	
monospace	A font where every letter needs the same space	Courier New

### Font weight font style

```
span{font-weight:bold;}
```

Name	Description
bold	Bold text
bolder	Even bolder text
light	Light text
100, 200, 300, 400, 500, 600, 700, 800, 900	From very light (100) to very bold (900)

```
span{font-style:italic;}
```

```
<span> this font is italic </span>
```

## Font size

```
span{font-size:medium;}
```

The font size and sizes in general can be specified in different ways in CSS. There are so called generic values:

<i>Name</i>	<i>Description</i>	<i>Example</i>
x-small	Very small size	x-small
small	Still small size	small
medium	Standard size	medium
large	Large size	large
x-large	Very large	
xx-large	The largest	

I hope u can imagine where this is going although i made no examples for x-large and xx-large ;)

Numeric values are also possible, but there are two different types:

**absolute** Values should only be used for print media. They are units “like points” or “pixel”.

**relative** Values should be used when designing for the web. They are units like “percent” or “em”.

<i>Name</i>	<i>Type</i>	<i>Description</i>
pt	absolute	Stands for point. Used in print media, equals 1/72 inch
in	absolute	Stands for inch
px	absolute	Stands for pixel
mm / cm	absolute	Stands for millimeters / centimeters
em	relative	1 em equals the standard height of a big letter of the font
ex	relative	1 ex equals the height of a small “x”
%	relative	100% equal the normal font size

For web pages relative values should be used, because visually handicapped people often use very big standard fonts in their system properties and fixed absolute values make web pages for them unreadable.

## Font color

```
span{font-color:#f00;}
span{font-color:red;}
span{font-color:rgb(100%,0,0);}
```

You can define colors in CSS in the same as you can in HTML. First there are generic color-names like “red”, “blue” or “aqua”.

But there is also the possibility to define colors in RGB by using “#RRGGBB”, while every letter is a hexadecimal number. For example “#ff0000” is red. New in CSS is the short form “#RGB”, which doubles every number. “#f00” equals “#ff0000”.

Also added with CSS is the command rgb(R,G,B). R, G and B are replaced either by a number between 0 and 255 or a percent value.

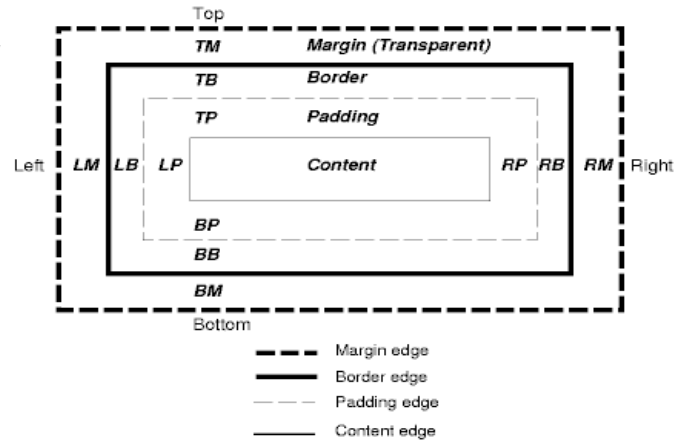
So all the rules above do exactly the same.

New with CSS2 are user specified colors like “menu” or “scrollbar”, which are defined by the browser or the system.

## The box model

The CSS box model describes the rectangular boxes that are generated for elements in the [document tree](#) and laid out according to the [visual formatting model](#).<sup>1</sup>

Every box has a content area. It's the area where the elements are put. Its surround by an optional padding-, border- and/or margin-area.



### The margin

The margin is the distance between the box and the next box's margin. Its standard value is "0". You can define the border for every side of the box separately.

```
div{margin:100px;}
```

Name	Description
margin	Defines the margin for all four sides simultaneously
left-margin	Defines the margin only for the left side
right-margin	Defines the margin only for the right side
top-margin	Defines the margin for the top side
bottom-margin	Defines the margin for the bottom

**Note**, the div-tag is the same as the span-tag just for defining the position of elements.

### The padding

```
div{padding:100px;}
```

The padding defines the distance between the content box and the border of the box. Like the margin "left-", "right-", "top-" and "bottom-" can be added to define different values for different sides.

### The border

```
div.border{border-width:1.2em;border-style:solid;border-color:red;}
```

A border has three different properties. The border width can be defined with any distance method described on page 6, except the "%" method. Same with the border-color which can be described with any method to determine colors.

Border-style	Description
solid	A solid border
dotted	A dotted border

<sup>1</sup> <http://www.w3.org/TR/REC-CSS2/box.html>

<i>Border-style</i>	<i>Description</i>
dashed	A dashed border
double	A border consisting of two solid borders
groove	A border with 3D-effect
ridge	A border with 3D-effect
inset	A border with 3D-effect
outset	A border with 3D-effect

## Visual formatting

### Position

```
div{position:absolute;top:100px;left100px;}
```

You can define the position of a model with the “position” property. With top, bottom, left and right you can define the distance to the elements of the sides.

<i>Position</i>	<i>Description</i>
fixed	Fixes the box at the browser window. Its not scrolled and other boxes are not concerned.
static	Its the standard value, top, bottom, left and right may not be specified
relative	Relative to the other boxes.
absolute	Like relative, only boxes that are not static are concerned

### Box size

```
div{width:100px;height:100;overflow:hidden;}
```

Defines the size of a box. Width defines the size in x-Direction and height in y-Direction. Note that long words can soon get to small for the box size. To define what happens with long words the property overflow is used.

<i>Overflow</i>	<i>Description</i>
visible	The content just gets painted over the border of the box. <b>Note</b> , the Internet Explorer just extends the box till all content fits in it
hidden	Hides the part of the content that is to big for the box
scroll	The part of the content that is to big is cut off, but scrollbars are used to make the text visible
auto	The browser decides automatically which method he choses

In **CSS 2** the properties max-width, min-width, max-height and min-height were added to define a maximum or minimum distance.

```
div{max-width:400px;min-width:100px}
```

**Note**, the Internet Explorer doesn't interpret these properties till today.

## Pseudo classes

There are pseudo classes to define the look of links in an HTML-document. These pseudo classes offer a possibility to give a link in different states a different look.

<i>Pseudo classes</i>	<i>Description</i>
<code>:link{ ... }</code>	The rules defined here apply to every link
<code>:visited{ ... }</code>	The rules defined here apply to a visited link
<code>:active{ ... }</code>	The rules defined here apply to a link which was just clicked and is still active
<code>:hover{ ... }</code>	The rules defined here apply to a link above which the mouse is at the moment
<code>:focus{ ... }</code>	The rules defined here apply to a link which has a focus, for example by tabbing through the page

Not that Opera in the current version (8.5) doesn't support the pseudo class "focus".

```
a:link{color=blue;}
a:visited{color=red;}
a:hover (background-color=green;}
```

These rules would paint a link in blue color and a visited link in red color. If the mouse is over a link the background color of the link is set to green.

## The Acid 2 Test

Remember when I told you at the beginning of this text that all up-to-date browsers fully understand the CSS1 recommendation? Well I was lying about that.

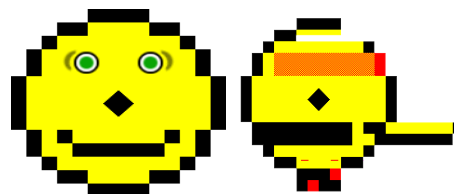
Although all the browser companies claim that their browsers correctly interpret all CSS1 code they still have a lot of bugs. "The Web Standards Project" have made it their task to bring this bugs to the light and help the browser developers to improve their products. Therefore they themselves developed a test using only HTML4, png Graphics, DataURLs and CSS1. Anyway all the common browsers Firefox, Internet Explorer, Netscape and even Opera, which technical director Wium Lie helped developing the test, failed.

The first browser to pass the test was the apple safari browser. Meanwhile iCab, another Mac-browser, Konqueror and Opera in Version 9.0 (still only available as beta version) have passed the test too.

Microsoft and Mozilla have announced to not consider the test in their browser development.

The test itself uses rarely used standards like:

- Absolute and relative positioning
- height / width and their max- /min- equivalents
- Invalid statements that shouldn't be interpreted by the browser.
- Hover effects
- Transparent pngs



*Result of a  
passed Acid  
Test*

*And a try with the  
FireFox browser*

Sources:

- <http://de.selfhtml.org/>
- <http://www.w3.org/>
- <http://www.webstandards.org/>
- Cascading Style Sheets The Definitive Guide. , Eric A. Meyer, O'Reilly Media 2004