Cascading Style Sheets (CSS)

Name: Kumuda Devi Goddati
Matriculation Number: 230473
Course: BCM SS08
Subject: E-Business Technologies
Professor: Dr. Eduard Heindl
June 25, 2008

CERTIFICATE OF DECLARATION

It is to be declared that all the work done in this document is solely and explicitly done by the author of this document.

The list of literatures, whose text has been adapted, is mentioned in the document with reference.

KUMUDA DEVI GODDATI
JUNE 25, 2008

TABLE OF CONTENTS

1. Introduction to Cascading Style Sheets

Cascading Style Sheets, in short CSS, is a major breakthrough in web page design, enabling the designers to control and enhance the appearance of web pages. It is helpful in overcoming the limitations of web document formats as it works with HTML (Hypertext Markup Language), which is the primary document format on the Web. HTML describes the structure of the document, and it concentrates more on the structure rather than its appearance. As HTML is a markup language it indicates the roles that the various parts of the document have to play and has limited capability in influencing the appearance. CSS can be used to make the appearance of the HTML page better and also make it available to more users worldwide, thereby enhancing the potential of HTML and the Web.
There was a web browser war which resulted in adding various style features to the HTML documents like font size, font style etc and due to this managing the webpage was difficult and the structure and appearance couldn't be differentiated. Therefore, style information was kept in a separate document called the Cascading style sheet document (CSS). This information is put within the HTML document. For single files it is not difficult that this info is included automatically but for a web page this is done by using a separate file (.css). This file contains information about the layout of the resulting page. There are two versions of CSS. One is the CSS1 and the latest is CSS2. CSS1 has all the basic fundamentals but it is not more specific which is overcome by CSS2. Today CSS2 is supported by all the browsers. In this paper, both the version are taken into consideration and the latest features are explained.
"A style sheet is a set of guidelines that tell a browser how a HTML document is to be presented to the users"[1]. Using CSS, styles such as the size, color, spacing of text, and a whole lot of other features can be specified which will be explained in the coming sections of this paper. The main feature of CSS is that style sheets can cascade, i.e. different style sheets can be attached to a document to influence its presentation. CSS is the perfect solution for satisfying the needs of the Web as it is a powerful style sheet language as it enables

2. Basic Concepts of CSS

2.1 CSS Rule

A simple CSS rule contains a rule. "A rule is a statement about one stylistic aspect of one or more elements". "A style sheet is a set of one or more rules that apply to a HTML document".

A rule consists of two parts:

- Selector
- Declaration

"The selector is the link between the HTML document and the style. It specifies what elements are affected by the declaration.

The declaration is that part of the rule that sets forth what the effect will be." The declaration has two parts: property and value.

"The property is a quality or characteristic that something possesses. The value is a precise specification of the property."[2]

[1] Cascading Style Sheets, Hakon Wium Lie and Bert Bos, Printed By Addison-Wesley, Second Edition, Pg2
[2] Cascading Style Sheets, Hakon Wium Lie and Bert Bos, Printed By Addison-Wesley, Second Edition, Pg31,32

E.g. To set the text color of 'H1' elements to blue, we can write as:

H1 { color: blue }

In the above example H1 is the selector and 'color: blue' is the declaration. In the declaration, as mentioned above there are two parts. Color is the property and the value is blue.

The following diagram shows all the components of a rule.

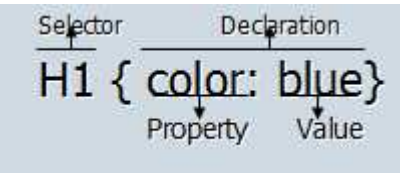


Diagram of a rule

2.2 Linking Style sheets to the HTML document[3]

The style sheet must be linked to the HTML document, in order to influence the presentation. The style sheet and the HTML document must be combined so that they can work together for the document presentation. This can be done in four ways:

1. Apply the basic, document-wide style sheet for the document by using the STYLE element:
   This is an internal style sheet and should be used when a single document has a unique style. The style sheet and the HTML document can be linked together by putting the style sheet inside a STYLE element at the top of the document.
   For ex:

   ```
   <html>
   <style>
   H1 { color: green }
   </style>
   <body>
   <H1>This is the header</H1>
   </body>
   ```

   In the above example, the header color is given as green inside the style tag. The style element is inside the

2. Apply a style sheet to an individual element using the STYLE attribute:

[3] Cascading Style Sheets, Hakon Wium Lie and Bert Bos, Printed By Addison-Wesley, Second Edition, Pg34

STYLE attribute can be used to apply styles to all types of elements. Instead of having a value that can be referred to in a selector, the value of the style attribute is actually one or more CSS declarations.

3. Link an external style sheet to the document using the LINK element:
   The easiest way to apply a style sheet is via the LINK element. For LINK to work, the style sheet being linked to must exist as a separate file. For ex:

   <LINK REL="STYLESHEET" TYPE= "text/css" HREF="mystyle">

   mystyle is the URL of the requested style sheet. The attribute REL="STYLESHEET" tells the browser that link is to a style sheet. Without this attribute, the browser will not attempt to load the style sheet designated by the URL.
4. Import a style sheet using the **CSS @import** notation.
   The LINK element described above is HTML's way to external style sheets. CSS also has a way importing external style sheets by using the @import notation. There is always a URL after @import.

   The following example illustrates the above four ways of linking the style sheets to the HTML document.[4]

```
<HTML>
 <HEAD>
  <TITLE>title</TITLE>
  <LINK REL=STYLESHEET TYPE="text/css"
   HREF="http://style.com/cool" TITLE="Cool">
  <STYLE TYPE="text/css">
   @import url(http://style.com/basic);
   H1 { color: blue }
  </STYLE>
 </HEAD>
 <BODY>
  <H1>Headline is blue</H1>
  <P STYLE="color: green">While the paragraph is green.
 </BODY>
</HTML>
```

The example shows that: 'LINK' element is used to link an external style sheet, a 'STYLE' element inside the HEAD element, an imported style sheet using the CSS '@import' notation, and a 'STYLE' attribute on an element inside 'BODY'. The style attribute option mixes style with content and loses the advantages of traditional style sheets. Hence it is better to use the style attribute for occasional stylistic changes.

2.3 Grouping[5]

Grouping helps in reducing the size of the style sheets which leads to the faster loading. CSS provides several methods to shorten style sheets by way of grouping selectors and declarations.

[4] http://www.w3.org/TR/REC-CSS1
[5] Cascading Style Sheets, Hakon Wium Lie and Bert Bos, Printed By Addison-Wesley, Second Edition, Pg33

For example:

```
H1 { font-weight: bold }
H2 { font-weight: bold }
H3 { font-weight: bold }
```

In the above example, all the three rules have the same declaration i.e. they set the font to be bold by using the font-weight property. Since all the declarations are identical, they can be grouped into a

H1, H2, H3 { font-style: bold }

The above rule will produce the same result as the first three rules.

A selector may have more than one declaration. For example:

```
H1 {
  font-weight: bold;
  font-size: 12pt;
  line-height: 14pt;
  font-family: helvetica;
  font-variant: normal;
  font-style: normal;
}
```

The declarations that relate to the same selector can be grouped into a semicolon-separated list as shown above. All the declarations must be contained within the pair of curly braces

2.4 Inheritance[6]

HTML represents a document with a tree-like structure and elements have children and parents. For style sheets, having tree-structured documents is very a good reason: inheritance Through inheritance, CSS property values set on one element will be transferred down the tree to its descendants. HTML elements inherit stylistic properties.

For example, to have the same color on all elements in your document, we list all the element types in the selector.

```
<style type="text/css">
   H1, H2, P { color: green }
</style>
```

In the above example, instead of setting the style on each element type, it can be set to the BODY element as these elements are its descendants.

```
<style type="text/css">
   BODY { color: green }
</style>
```

Also all the other elements in the body element, inherit its properties. Inheritance distributes stylistic properties to descendants of an element. Since the BODY element is a common

[6] Cascading Style Sheets, Hakon Wium Lie and Bert Bos, Printed By Addison-Wesley, Second Edition, Pg37

ancestor for all visible elements, it is a convenient selector when the stylistic rules are to be set to the entire document. “

To set a default style property for a document, one can set the property on an element from which all the visible elements descend”[7]. In HTML documents, BODY element can serve this function as shown in the above example.

Some style properties such as ‘background’ property does not inherit from the parent element to the child element, but the parent’s background will be by default

3. CSS Selectors[8]

The four types of selector schemes are supported by CSS2. Each is based on certain aspect of an element:

- Type of the element
- Attributes of the element
- The context in which the element is used
- External information about the element

3.1 Type Selectors

Type selector is the simplest selector in CSS, is the name of an element type. This selector is to apply the declaration to every instance of the element type.
For example to affect the rule with a selector H1,

```
H1 { color: blue }
```

The rule written above affects all the H1 elements in a document.
In order to write several style rules that are same, grouping can be used as described in the previous section of this paper.

3.2 Attribute Selectors

- Class as selector
- ID as selector

3.2.1 CLASS Selector

The CLASS attribute enables to apply declarations to a group of elements that have the same value on the CLASS attribute. All elements inside BODY can have a CLASS attribute. When

[7] http://www.w3.org/TR/REC-CSS1
[8] Cascading Style Sheets, Hakon Wium Lie and Bert Bos, Printed By Addison-Wesley, Second Edition, Pg58

the elements are classified using the CLASS attribute, rules are created that refer to the value of the CLASS attribute , then those rules are automatically applied to the group of elements by the browser.

For example[9]:

```
<HTML>
 <HEAD>
  <TITLE>Title</TITLE>
  <STYLE TYPE="text/css">
    H1.pastoral { color: #00FF00 }
  </STYLE>
 </HEAD>
 <BODY>
  <H1 CLASS=pastoral>Way too green</H1>
 </BODY>
</HTML>
```

Class names must be single words although digits and dashes can be used. In contrast to the element names, class names are case-sensitive. The class names in the style sheet must be exact with the uppercase and lowercase letters as in the HTML source file

"The next step is to write style rules with selectors that refer to the class name. A selector that includes a class name is called a class selector."[10]

The anatomy of a rule with a class selector is as follows:

.pastoral { color: green } /* all elements with CLASS pastoral */

Here in the above example,
'.' is the Flag Character – It signals the type of selector what follows (elements with class name)
The above example says: elements with class name pastoral
The color is the property with the value green and is the declaration of the rule.

3.2.2 ID Selector

The ID attribute works like the CLASS attribute except that the value of an ID attribute must be unique throughout the document. All the elements inside BODY can have an ID attribute but all the values must be different. ID attribute is useful for setting style rules on individual elements.
"A selector that includes an ID attribute is called an ID selector".[11]

For example:

[9] http://www.w3.org/TR/REC-CSS1
[10] Cascading Style Sheets, Hakon Wium Lie and Bert Bos, Printed By Addison-Wesley, Second Edition, Pg60
[11] Cascading Style Sheets, Hakon Wium Lie and Bert Bos, Printed By Addison-Wesley, Second Edition, Pg62

#xyz34 { text-decoration: underline }

Where,
#xyz34 is the ID selector
(hash mark)is the Flag character. It alerts the browser that an ID value is coming up next.
Xyz34 is the ID value
The above example says: an element with an ID value equal to xyz34
The declaration has a property text-decoration with a value underline.

The HTML syntax of the element on which we want to use the ID attribute resembles that of other elements with attributes, for ex:

<P ID=xyz34>Underlined text</P>

Combined with the style sheet rule as mentioned above, the content of the element will be underlined. Because the value of the ID attribute must be unique, it cannot be used again even in the same document. The style properties are set on a per-element basis.

3.3 Contextual Selectors

A contextual selector is a selector that takes into account the context in which the style is to be applied. The specified style is applied to the element only if it is in the specified context. An element's context is formed by its ancestor elements and the elements that precede it.

To give 'EM' elements within 'H1' elements a different color, the following two rules may be written:

```
H1 { color: blue }
EM { color: red }
```

When this style sheet is in effect, all emphasized sections within or outside 'H1' will turn red. If we want only 'EM' elements to turn red then it can be specified with:

```
H1 EM { color: red }
```

Contextual selectors consist of several simple selectors separated by whitespace. As shown in the above example only elements that match the last simple selector EM element are addressed and search pattern matches if EM is a descendant of H1. Contextual selectors can look for element types, CLASS attributes, ID attributes or combination of these

For example[12]:

```
DIV P         { font: small sans-serif }
.reddish H1     { color: red }
#x78y CODE      { background: blue }
DIV.sidenote H1 { font-size: large }
```

[12] http://www.w3.org/TR/REC-CSS1

The first selector matches all 'P' elements that have a 'DIV' among the ancestors.
The second selector matches all 'H1' elements that have an ancestor of class 'reddish'.
The third selector matches all 'CODE' elements that are descendants of the element with 'ID=x78y'.
The fourth selector matches all 'H1' elements that have a 'DIV' ancestor with class 'sidenote'.
Several contextual selectors can be grouped together:

```
H1 B, H2 B, H1 EM, H2 EM { color: red }
```

4. Pseudo-classes and pseudo-elements

In CSS1, style is normally attached to an element based on its position in the document structure. This simple model doesn't cover some common effects designers want to achieve. Pseudo-classes and pseudo-elements were devised to fill these gaps.
Using pseudo-classes the style of a document's links can be changed whether the links have been visited or based on the user interaction with the document. Also, the style of the first letter and first line of an element can be changed and the elements that are not present in the source document can be added.

4.1 Anchor Pseudo-classes

In HTML, there is only one element that can be a hyperlink: the A (anchor) element. "An anchor pseudo-class is a mechanism by which a browser indicates to a user the status of a hyperlink in the document the user is viewing."[13]
As there is no way for the author to know if the user has visited a link, colors that indicate the status of links can be set in the style sheet. This is done by including an anchor pseudo-class in the selector.

For example:

```
A:visited { color: blue }   /* visited links */
```

In the above examples,
A:visited is the selector
A is the element type
: is the flag character
visited is the pseudo-class name
Declaration containing the property with a value blue

As it can be seen in the example, the selector is a combination of a type selector and a pseudo-class selector. The flag character is a colon (:) and is used by both pseudo-classes and pseudo-elements

[13] Cascading Style Sheets, Hakon Wium Lie and Bert Bos, Printed By Addison-Wesley, Second Edition, Pg68

4.2 Typographical Pseudo-elements

Some common typographical effects are associated not with structural elements but with typographical elements as formatted on the canvas. Pseudo-elements allow you to set style on a subpart of an element's content. They have been introduced to allow for designs that would not have been possible otherwise.

Two of these pseudo-elements are:

- first-line pseudo-element
- first-letter pseudo-element

The effects of these elements are not related to the structure of the HTML document but based on how the element is formatted. They enable to impose styles on the first letter of a word and on the first line of a paragraph respectively.

4.2.1 First-line pseudo-element

A common usage is to increase the size of the first letter or make the first line use uppercase letters.
An example of pseudo-element :

P.INITIAL:FIRST-LINE { text-transform: uppercase }

Where,
P.INITIAL:FIRST-LINE is the selector
P is the element type
. is the flag character
INITIAL is the pseudo-element name
text-transform-uppercase is the declaration

In the above example, the pseudo-element selector is combined with two other selectors; one is a type selector (P) and a class selector (INITIAL). The resulting selector reads: the first line of P elements with class name initial. The first-line pseudo-element can only be attached to a block-level element.
Only the following properties apply to a 'first-line' element: font properties, color and background properties, 'word-spacing', 'letter-spacing', 'text-decoration', 'vertical-align', 'text-transform', 'line-height', 'clear'.

4.2.2 First-letter pseudo-element

The 'first-letter' pseudo-element is used for "initial caps" and "drop caps", which are common typographical effects. A drop-cap initial is a common trick in typography: the first letter of a text is enlarged and dropped into the formatted paragraph.
It is similar to an inline element if its 'float' property is 'none, otherwise it is similar to a floating element.

Style rules can be attached to the first letter of an element by using the first-letter pseudo-element
For example:

P.initial:first-letter { font-size: 200% }

The selector in the above example reads: the first letter of P elements with class name initial. The style rule will apply to the first letter of lines that were affected in the above example. The whole rule says: “the first letter of P elements with class name ‘initial’ should have a font size two times bigger than the surrounding text”.

These are the properties that apply to 'first-letter' pseudo-elements: font properties, color and background properties, 'text-decoration', 'vertical-align' (only if 'float' is 'none'), 'text-transform', 'line-height', margin properties, padding properties, border properties, 'float', 'clear'.

4.2.3 pseudo-elements in selectors[14]

In a contextual selector, pseudo-elements are only allowed at the end of the selector:

BODY P:first-letter { color: purple }

Pseudo-elements can be combined with classes in selectors:

P.initial:first-letter { color: red }

<P CLASS=initial>First paragraph</A>

The above example would make the first letter of all 'P' elements with 'CLASS=initial' red. When combined with classes or pseudo-classes, pseudo-elements must be specified at the end of the selector. Only one pseudo-element can be specified per selector.

4.2.4 multiple pseudo-elements[15]

Several pseudo elements can be combined:

P { color: red; font-size: 12pt }
P:first-letter { color: green; font-size: 200% }
P:first-line { color: blue }

<P>Some text that ends up on two lines</P>

In this example, the first letter of each 'P' element would be green with a font size of 24pt. The rest of the first line (as formatted on the canvas) would be blue while the rest of the paragraph would be red. Assuming that a line break will occur before the word "ends", the fictional tag sequence is:

<P>

[14] http://www.w3.org/TR/REC-CSS1
[15] http://www.w3.org/TR/REC-CSS1

<P:first-line>
<P:first-letter>
</P:first-letter>some text that
</P:first-line>
ends up on two lines
</P>

Note that the 'first-letter' element is inside the 'first-line' element. Properties set on 'first-line' will be inherited by 'first-letter', but are overridden if the same property is set on 'first-letter'.

5. The Cascade[16]

"Cascading refers to the cascade of style sheets from different sources that may influence the presentation of the document. The cascading mechanism is designed to resolve conflicts between these style sheets. Cascading is horizontal i.e. it collects the rules that apply to the same elements.

In CSS, more than one style sheet can influence the presentation simultaneously. There are two main reasons for this feature: modularity and author/reader balance.

Modularity

> A style sheet designer can combine several (partial) style sheets to reduce redundancy:

```
@import url(http://www.style.org/pastoral);
@import url(http://www.style.org/marine);
H1 { color: red }     /* override imported sheets */
```

Author/reader balance

> "Both readers and authors can influence the presentation through style sheets. To do so, they use the same style sheet language thus reflecting a fundamental feature of the web: everyone can become a publisher. The UA is free to choose the mechanism for referencing personal style sheets.

Sometimes conflicts will arise between the style sheets that influence the presentation. Conflict resolution is based on each style rule having a weight. By default, the weights of the reader's rules are less than the weights of rules in the author's documents. I.e., if there are conflicts between the style sheets of an incoming document and the reader's personal sheets, the author's rules will be used. Both reader and author rules override the UA's default values.

The imported style sheets also cascade with each other, in the order they are imported, according to the cascading rules defined below. Any rules specified in the style sheet itself override rules in imported style sheets. That is, imported style sheets are lower in the cascading order than rules in the style sheet itself. Imported style sheets can themselves import and override other style sheets, recursively.

In CSS1, all '@import' statements must occur at the start of a style sheet, before any declarations. This makes it easy to see that rules in the style sheet itself override rules in the imported style sheets."

[16] http://www.w3.org/TR/REC-CSS1

5.1 Important

“Style sheet designers can increase the weights of their declarations:

```
H1 { color: black ! important; background: white ! important }
P  { font-size: 12pt ! important; font-style: italic }
```

In the example above, the first three declarations have increased weight, while the last declaration has normal weight.

A reader rule with an important declaration will override an author rule with a normal declaration. An author rule with an important declaration will override a reader rule with an important declaration.”

5.2 Cascading Order[17]

“Conflicting rules are intrinsic to the CSS mechanism. To find the value for an element/property combination, the following algorithm must be followed:

1. Find all declarations that apply to the element/property in question. Declarations apply if the selector matches the element in question. If no declarations apply, the inherited value is used. If there is no inherited value (this is the case for the 'HTML' element and for properties that do not inherit), the initial value is used.
2. Sort the declarations by explicit weight: declarations marked '!important' carry more weight than unmarked (normal) declarations.
3. Sort by origin: the author's style sheets override the reader's style sheet which override the UA's default values. An imported style sheet has the same origin as the style sheet from which it is imported.
4. Sort by specificity of selector: more specific selectors will override more general ones. To find the specificity, count the number of ID attributes in the selector (a), the number of CLASS attributes in the selector (b), and the number of tag names in the selector (c). Concatenating the three numbers (in a number system with a large base) gives the specificity. Some examples:

```
LI            {...}  /* a=0 b=0 c=1 -> specificity =   1 */
UL LI         {...}  /* a=0 b=0 c=2 -> specificity =   2 */
UL OL LI      {...}  /* a=0 b=0 c=3 -> specificity =   3 */
LI.red        {...}  /* a=0 b=1 c=1 -> specificity =  11 */
UL OL LI.red  {...}  /* a=0 b=1 c=3 -> specificity =  13 */
#x34y         {...}  /* a=1 b=0 c=0 -> specificity = 100 */
```

   Pseudo-elements and pseudo-classes are counted as normal elements and classes, respectively.

5. Sort by order specified: if two rules have the same weight, the latter specified wins. Rules in imported style sheets are considered to be before any rules in the style sheet itself.

[17] http://www.w3.org/TR/REC-CSS1

The search for the property value can be terminated whenever one rule has a higher weight than the other rules that apply to the same element/property combination.

This strategy gives author's style sheets considerably higher weight than those of the reader. It is therefore important that the reader has the ability to turn off the influence of a certain style sheet, e.g. through a pull-down menu.

A declaration in the 'STYLE' attribute of an element has the same weight as a declaration with an ID-based selector that is specified at the end of the style sheet:

```
<STYLE TYPE="text/css">
  #x97z { color: blue }
</STYLE>

<P ID=x97z STYLE="color: red">
```

In the above example, the color of the 'P' element would be red. Although the specificity is the same for both declarations, the declaration in the 'STYLE' attribute will override the one in the 'STYLE' element because of cascading rule number 5.

The UA may choose to honour other stylistic HTML attributes, for example 'ALIGN'. If so, these attributes are translated to the corresponding CSS rules with specificity equal to 1. The rules are assumed to be at the start of the author style sheet and may be overridden by subsequent style sheet rules. In a transition phase, this policy will make it easier for stylistic attributes to coexist with style sheets."

6. Media Types[18]

When designing Cascading Style sheets we have to keep in mind about the output type.
One of the most important features of style sheets is that they specify how a document can be presented on different media:

- Screen
- With Speech synthesizer
- With a Braille device
- Printer

Many web users see the web through the computer screen which is a very dynamic device as it refreshes 60 times per second. This allows video clips, animations and other dynamic behaviour to be incorporated into web documents.
The printed page on the other hand is static, still many prefer reading from the paper because of its higher resolution and high legibility. Printouts from the web are often of poor quality and CSS2 adds functionality to improve printing by using the features:

- Page breaks: can be used to say where the page breaks should occur and where they should be avoided.
- Page margins and page orientation

Also CSS2 introduces a way to say that a style sheet or a section of style sheet only takes effect on certain output media, for example printer or a hand-held web device. This feature is known as media-specific style sheets.
Certain CSS properties are only designed for certain media especially if we see aural style sheets there are properties defined specifically for aural users. However, there are some

[18] http://www.w3.org/TR/REC-CSS2/media.html

properties which can be shared by different media types like the font-size property, which can be used for screen and print media. The property may be common but the value differs as a computer screen typically needs a bigger font than on paper.
There are many types of web devices and style sheets ensure that content remains in form so that it can be displayed on all devices. For example, by storing text as characters speech synthesizers can read documents aloud and cell phone screen can show web pages.
For this reason, it is necessary to express that a style sheet or a section of the style sheet applies to different media types.

6.1 Specifying media-dependant style sheets

There are currently two ways of specifying media dependencies for style sheets:

- Specify the target medium from a style sheet with the @media or @import at-rules.
  - For example:

```
@import url("loudvoice.css") aural;
@media print {
/* style sheet for print goes here */
}
```

- Specify the target medium within the document language.
  - For example:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<HTML>
<HEAD>
<TITLE>Link to a target medium</TITLE>
<LINK rel="stylesheet" type="text/css"
 media="print, handheld" href="foo.css">
</HEAD>
<BODY>
  <P>The body...
  </BODY>
</HTML>
```

6.2 The @media rule

The @media rule specifies the target media types (separated by commas) of a set of rules (delimited by curly braces). Inside the curly brackets that follow is a normal style sheet since it applies to a certain media type, it is said to be a media-specific style sheet. The @media rule allows style sheet rules for various media in the same style sheet. The @media keyword is followed by the name of a media type (CSS2 defines 9 media types which will be in the following paragraphs).
For example:

```
@media print {
    BODY { font-size: 10pt }
  }

  @media screen {
    BODY { font-size: 12pt }
  }

  @media screen, print {
    BODY { line-height: 1.2 }
  }
```

In the first example the media is print, in the second example it is screen and in the third there are two media screen and print separated by comma. Also we can see that font-size property is applied to several media types. Other properties such as 'volume' are used only in aural style sheets.
For example:

```
@media aural {
    BODY { volume: soft }
  }
```

6.3 Media Types[19]

A CSS media type names a set of CSS properties. The names are chosen for CSS media types so that they reflect target devices for which the properties are relevant.
The following are the nine media types defined by CSS2.

6.3.1 screen

This media type is intended for scrollable color computer screens. There are examples for this media type in the previous section above (6.2)

6.3.2 print

This media type is intended for paged, opaque material and for documents viewed on screen in print preview mode (For printers, when printing on paper). There are few examples for this media type in the section 6.2

6.3.4 aural

This media type is intended for speech synthesizers. Speech synthesizers read out web documents loud.

6.3.5 braille

This media type is intended for electronic Braille readers which display characters by making tactile patterns.
For example:

```
@media braille {
    * { margin: 0}
  }
```

The above example will set all margins around all elements to 0 to economize the available display space.

6.3.5 embossed

This media type is intended for paged braille printers.
For example:

```
@media embossed {
```

[19] Cascading Style Sheets, Hakon Wium Lie and Bert Bos, Printed By Addison-Wesley, Second Edition, Pg259-262

```
    @page { margin: 0 }
  }
```
The above example will set page margins to 0 so all available space is used to display content.

6.3.6 handheld

This media type is intended for handheld devices which typically have small screens and limited bandwidth.
For example:
```
@media handheld {
    IMG { display: none }
 }
```

The above example will not display images on handheld devices. If the style sheet is evaluated in a proxy server it can save much bandwidth.

6.3.7 projection

This media type is intended for projected presentations, for example projectors or print to transparencies. Typically, these presentations have few lines per page and need large font sizes.
For example:
```
@media projection {
    BODY { font-size: x-large }
    H1 { page-break-before: always }
    }
```
The above style sheet will set the font size to be extra large and ensure a page break before every H1 element.

6.3.8 tty

This media type is used for Intended for media using a fixed-pitch character grid, such as teletypes, terminals, or portable devices with limited display capabilities.
For example:
```
@media tty {
    H1 { margin-below: 1em }
 }
```
Since these devices are based on characters and not pixels, they will not be able to display rich styles. The main benefit of style sheets for these devices is that text can be kept as text rather than images and is viewable at all.

6.3.9 tv

This media type is intended for television-based presentations, typically on screens with colors, low resolution, long viewing distance and limited capabilities for scrolling.
For example:
```
@media tv {
    BODY { color: white;
           Background: black;
 }
 }
```
The above example will set colors to be white on black (which is normal on tv) rather than black on white (which is normal on computer screens).

6.3.10 all

This media type is suitable for all devices.
For example:

```
@media all {
    BODY { line-height: 1.4;
 }
 }
```

The above example, only newer browsers which are aware of media specific style sheets will see the style sheet inside.
Media type names are case-sensitive.

6.4 Media groups[20]

Each CSS property definition specifies the media types for which the property must be implemented by a conforming user. Each property applies to all media types in the media groups listed in its definition.

CSS2 defines the following media groups:

- **continuous** or **paged**. "Both" means that the property in question applies to both media groups.
- **visual**, **aural**, or **tactile**.
- **grid** (for character grid devices), or **bitmap**. "Both" means that the property in question applies to both media groups.
- **interactive** (for devices that allow user interaction), or **static** (for those that don't). "Both" means that the property in question applies to both media groups.
- **all** (includes all media types)

The relation between media types and media groups can be understood by referring to the world wide web consortium page explaining about CSS2.

7. Formatting Model[21]

The box model can be used to depict the structure of an HTML document. CSS assumes a simple box-oriented formatting model where each formatted element results in one or more rectangular boxes. The display property determines whether an element is displayed as a block, inline, list item or other type of element. This property has many possible values, but the most common are block and inline.

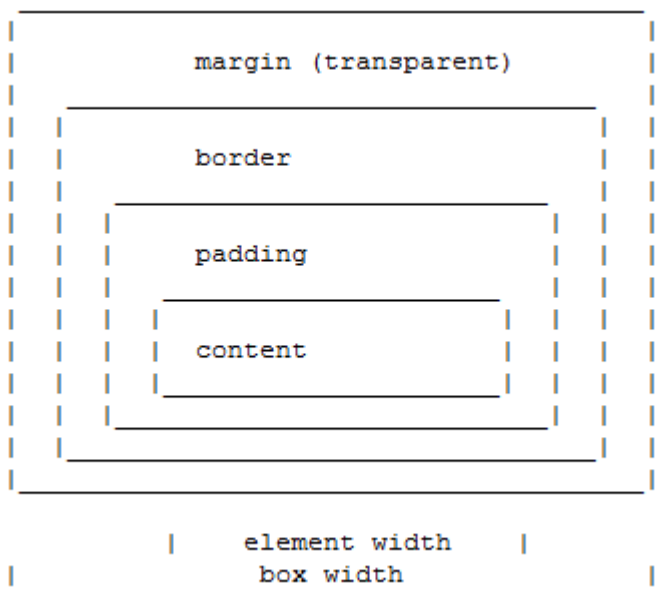
Elements with a display value of block or list-item are block-level elements. Also, floating elements (elements with a float value other than none) are considered to be block-level elements. An element with a box value starts and ends on a new line. An element with an inline value does not start and end on a new line. Elements that are not formatted as block level elements are inline elements.

An inline element can share line space with others. An element with a list-item value is displayed as a box with a label. Elements with a display property value of list-item are formatted as block-level items, but preceded by a list-item marker and is determined by list-

[20] http://www.w3.org/TR/REC-CSS2/media.html
[21] http://www.w3.org/TR/REC-CSS1

style property. To hide an element from view completely, display value can be set with none and will therefore not result in a box. A replaced element is an element which is replaced by content pointed to from the element.

All boxes have a content area with optional surrounding padding, border and margin areas. The properties that are used for adjusting margins, padding and borders of block elements. Then additional properties like width, height, float and clear are used for fine-tuning spacing of elements.



Formatting model

The size of the margin, border and padding are set with the margin, padding, and border properties respectively. The padding area uses the same background as the element itself (set with the background properties. The color and style for the border is set with the border properties. The margins are always transparent, so the parent element will shine through. The size of the box is the sum of the element width (i.e. formatted text or image) and the padding, the border and the margin areas.

8. CSS Properties[22]

8.1 Font Properties

8.1.1 Font

“The 'font' property is a shorthand property for setting 'font-style', 'font-variant','font-weight', 'font-size', 'line-height' and 'font-family' at the same place in the style sheet. The syntax of this property is based on a traditional typographical shorthand notation to set multiple properties related to fonts.

For a definition of allowed and initial values, see the previously defined properties. Properties for which no values are given are set to their initial value.

[22] http://www.w3.org/TR/REC-CSS1

```
P { font: 12pt/14pt sans-serif }
P { font: 80% sans-serif }
P { font: x-large/110% "new century schoolbook", serif }
P { font: bold italic large Palatino, serif }
P { font: normal small-caps 120%/120% fantasy }
```

In the second rule, the font size percentage value ('80%') refers to the font size of the parent element. In the third rule, the line height percentage refers to the font size of the element itself.

In the first three rules above, the 'font-style', 'font-variant' and 'font-weight' are not explicitly mentioned, which means they are all three set to their initial value ('normal'). The fourth rule sets the 'font-weight' to 'bold', the 'font-style' to 'italic' and implicitly sets 'font-variant' to 'normal'.

The fifth rule sets the 'font-variant' ('small-caps'), the 'font-size' (120% of the parent's font), the 'line-height' (120% times the font size) and the 'font-family' ('fantasy'). It follows that the keyword 'normal' applies to the two remaining properties: 'font-style' and 'font-weight'."

8.2 Color and Background Properties

8.2.1 Color

This property describes the text color of an element (often referred to as the *foreground* color). There are different ways to specify red:

```
EM { color: red }              /* natural language */
EM { color: rgb(255,0,0) }     /* RGB range 0-255   */
```

8.2.2 Background

"The 'background' property is a shorthand property for setting the individual background properties (i.e., 'background-color', 'background-image', 'background-repeat', 'background-attachment' and 'background-position') at the same place in the style sheet.

Possible values on the 'background' properties are the set of all possible values on the individual properties.

```
BODY { background: red }
P { background: url(chess.png) gray 50% repeat fixed }
```

The 'background' property always sets all the individual background properties. In the first rule of the above example, only a value for 'background-color' has been given and the other individual properties are set to their initial value. In the second rule, all individual properties have been specified."

8.3 Text Properties[23]

8.3.1 Word-spacing

“The length unit indicates an addition to the default space between words. Values can be negative, but there may be implementation-specific limits. The UA is free to select the exact spacing algorithm. The word spacing may also be influenced by justification (which is a value of the 'text-align' property).

```
H1 { word-spacing: 1em }
```

Here, the word-spacing between each word in 'H1' elements would be increased by '1em'.”

8.3.2 Letter-spacing

The length unit indicates an addition to the default space between characters. Values can be negative, but there may be implementation-specific limits. The UA is free to select the exact spacing algorithm. The letter spacing may also be influenced by justification (which is a value of the 'align' property).

```
BLOCKQUOTE { letter-spacing: 0.1em }
```

Here, the letter-spacing between each character in 'BLOCKQUOTE' elements would be increased by '0.1em'.

8.3.3 Text-decoration

The color(s) required for the text decoration should be derived from the 'color' property value.

This property is not inherited, but elements should match their parent. E.g., if an element is underlined, the line should span the child elements. The color of the underlining will remain the same even if descendant elements have different 'color' values.

```
A:link, A:visited, A:active { text-decoration: underline }
```

The example above would underline the text of all links (i.e., all 'A' elements with a 'HREF' attribute).

8.3.4 Vertical-align

The property affects the vertical positioning of the element. One set of keywords is relative to the parent element. Baseline, middle, sub, super, text-top, text-bottom, top, bottom are set of properties.

8.3.5 Text-transform

The text is transformed to capitalize, uppercase, lowercase or none.

```
For example: H1 { text-transform: uppercase }
```

8.3.6 Text-align

This property describes how text is aligned within the element.

[23] http://www.w3.org/TR/REC-CSS1

```
For example: DIV.center { text-align: center }
```
The alignments are relative to the width of the element, not the canvas.

8.3.7 Text-indent

This property specifies the indentation that appears before the first formatted line. The value of the text-indent may be negative, but there may implementation-specific limitations.

```
For example: P { text-indent: 3em }
```

8.3.8 Line-height

The property sets the distance between two adjacent lines' baselines. Negative values are not allowed. A value of 'normal' sets the 'line-height' to a reasonable value for the element's font. It is suggested that UAs set the 'normal' value to be a number in the range of 1.0 to 1.2.

8.4 Box Properties[24]

8.4.1 Margin

The 'margin' property is a shorthand property for setting 'margin-top', 'margin-right', 'margin-bottom' and 'margin-left' at the same place in the style sheet.

If four length values are specified they apply to top, right, bottom and left respectively. If there is only one value, it applies to all sides, if there are two or three, the missing values are taken from the opposite side.

```
BODY { margin: 2em } /* all margins set to 2em */
BODY { margin: 1em 2em } /* top & bottom = 1em, right & left = 2em */
BODY { margin: 1em 2em 3em } /* top=1em, right=2em, bottom=3em, left=2em */
```

The last rule of the example above is equivalent to the example below:

```
BODY {
  margin-top: 1em;
  margin-right: 2em;
  margin-bottom: 3em;
  margin-left: 2em;         /* copied from opposite side (right) */
}
```

Negative margin values are allowed, but there may be implementation-specific limits.

8.4.2 Padding

The 'padding' property is a shorthand property for setting 'padding-top', 'padding-right', 'padding-bottom' and 'padding-left' at the same place in the style sheet.

If four values are specified they apply to top, right, bottom and left respectively. If there is only one value, it applies to all sides, if there are two or three, the missing values are taken from the opposite side.

[24] http://www.w3.org/TR/REC-CSS1

The surface of the padding area is set with the 'background' property:

```
H1 {
  background: white;
  padding: 1em 2em;
}
```

The example above sets a '1em' padding vertically ('padding-top' and 'padding-bottom') and a '2em' padding horizontally ('padding-right' and 'padding-left'). The 'em' unit is relative to the element's font size: '1em' is equal to the size of the font in use.

Padding values cannot be negative.

8.4.3 Border-width

This property is a shorthand property for setting 'border-width-top', 'border-width-right', 'border-width-bottom' and 'border-width-left' at the same place in the style sheet.

There can be from one to four values, with the following interpretation:

- one value: all four border widths are set to that value
- two values: top and bottom border widths are set to the first value, right and left are set to the second
- three values: top is set to the first, right and left are set to the second, bottom is set to the third
- four values: top, right, bottom and left, respectively

In the examples below, the comments indicate the resulting widths of the top, right, bottom and left borders:

```
H1 { border-width: thin }                   /* thin thin thin thin */
H1 { border-width: thin thick }             /* thin thick thin thick */
H1 { border-width: thin thick medium }      /* thin thick medium thin */
H1 { border-width: thin thick medium thin } /* thin thick medium thin */
```

Border widths cannot be negative.

8.4.4 Border

“The 'border' property is a shorthand property for setting the same width, color and style on all four borders of an element. For example, the first rule below is equivalent to the set of four rules shown after it:

```
P { border: solid red }
P {
  border-top: solid red;
  border-right: solid red;
  border-bottom: solid red;
  border-left: solid red
}
```

Unlike the shorthand 'margin' and 'padding' properties, the 'border' property cannot set different values on the four borders. To do so, one or more of the other border properties must be used.

Since the properties to some extent have overlapping functionality, the order in which the rules are specified becomes important. Consider this example:

```
BLOCKQUOTE {
  border-color: red;
  border-left: double
  color: black;
}
```

In the above example, the color of the left border will be black, while the other borders are red. This is due to 'border-left' setting the width, style and color. Since the color value is not specified on the 'border-left' property, it will be taken from the 'color' property. The fact that the 'color' property is set after the 'border-left' property is not relevant.

Note that while the 'border-width' property accepts up to four length values, this property only accepts one

The other properties are width, height, float and clear. The width and height properties can be applied to text elements but are most useful with replaced elements such as images. Negative values are not allowed. For float, with the value 'none', the element will be displayed where it appears in the text. 'left' ('right') the element will be moved to the left (right) and the text will wrap on the right (left) side of the element and is treated as block-level. The value of clear property lists the sides where floating elements are not accepted. If clear is set to left, an element will be moved below any floating element on the left side and if clear is set to none, floating elements are allowed on all sides."

8.5 Classification Properties[25]

These properties classify elements into categories more than they set specific visual parameters.

8.5.1 Display

An element with a 'display' value of 'block' opens a new box. Typically, elements like 'H1' and 'P' are of type 'block'. A value of 'list-item' is similar to 'block' except that a list-item marker is added.
An element with a 'display' value of 'inline' results in a new inline box on the same line as the previous content. The box is dimensioned according to the formatted size of the content. If the content is text, it may span several lines, and there will be a box on each line. The margin, border and padding properties apply to 'inline' elements, but will not have any effect at the line breaks.
A value of 'none' turns off the display of the element, including children elements and the surrounding box.

```
P { display: block }
EM { display: inline }
LI { display: list-item }
IMG { display: none }
```

The last rule turns off the display of images.

[25] http://www.w3.org/TR/REC-CSS1

8.5.2 White-space

This property declares how whitespace inside the element is handled: the 'normal' way (where whitespace is collapsed), as 'pre' (which behaves like the 'PRE' element in HTML) or as 'nowrap' (where wrapping is done only through BR elements):

```
PRE { white-space: pre }
P   { white-space: normal }
```

The initial value of 'white-space' is 'normal'

8.5.3 List

The 'list-style' property is a shorthand notation for setting the three properties 'list-style-type', 'list-style-image' and 'list-style-position' at the same place in the style sheet.

```
UL { list-style: upper-roman inside }
UL UL { list-style: circle outside }
LI.square { list-style: square }
```

9. Conclusion

This paper shows that CSS is very helpful in web design because it allows the developers to enhance and control the style, appearance and layout of multiple web pages all at once and also enables the web developer to define style for each HTML element and apply to as many web pages as desired. By simply changing the style, all elements in the web page are updated automatically.

External style sheets enable to make such changes by simply editing one single CSS document. CSS is supported by all major browsers. W3C, the World Wide Web Consortium which standardized HTML created styles in addition to HTML 4.0. Style sheets are very useful as they save a lot of work. In this paper, the basic concepts of CSS, the formatting model and properties of cascading style sheets have been described briefly.

Various features of cascading style sheets regarding their applicability for various media is also the most important benefit of using style sheets. They can be used with different media thereby enhancing the user-friendliness and ease of using web pages all over the world. They support aural style sheets which are especially designed for aural users and are enriched with various specific properties for the purpose. There is a brief description about the various media types given in this paper.

10. Bibiliography

1. Cascading Style Sheets, Håkon Wium Lie and Bert Bos, Printed By Addison-Wesley, Second Edition

2. URL: http://www.w3.org/TR/REC-CSS1 ,Authors: Håkon Wium Lie, Bert Bos,(Referred on 12.06.08)

3. http://w3schools.com/css/ (Referred on 14.06.08)

4. http://www.w3.org/TR/REC-CSS2/, Editors Bert Bos, Håkon Wium Lie ,Ian Jacobs, Chris Lilley, (Referred on 24.06.08)