3. Cascading Style Sheets

3.1. Introduction

Cascading Style Sheets, fondly referred to as CSS, is a simple design language intended to simplify the process of making web pages presentable. CSS handles the look and feel part of a web page.

Using CSS, you can control the color of the text, the style of fonts, the spacing between paragraphs, how columns are sized and laid out, what background images or colors are used, as well as a variety of other effects.

CSS is easy to learn and understand but it provides powerful control over the presentation of an HTML document. Most commonly, CSS is combined with the markup languages HTML or XHTML.

3.1.1. Advantages of CSS

- **CSS saves time** - You can write CSS once and then reuse same sheet in multiple HTML pages. You can define a style for each HTML element and apply it to as many Web pages as you want.
- **Pages load faster** - If you are using CSS, you do not need to write HTML tag attributes every time. Just write one CSS rule of a tag and apply to all the occurrences of that tag. So less code means faster download times.
- **Easy maintenance** - To make a global change, simply change the style, and all elements in all the web pages will be updated automatically.
- **Superior styles to HTML** - CSS has a much wider array of attributes than HTML so you can give far better look to your HTML page in comparison of HTML attributes.
- **Multiple Device Compatibility** - Style sheets allow content to be optimized for more than one type of device. By using the same HTML document, different versions of a website can be presented for handheld devices such as PDAs and cell phones or for printing.
- **Global web standards** - Now HTML attributes are being deprecated and it is being recommended to use CSS. So its a good idea to start using CSS in all the HTML pages to make them compatible to future browsers.

3.1.2. CSS Syntax – Selectors

A CSS comprises of style rules that are interpreted by the browser and then applied to the corresponding elements in your document. A style rule is made of three parts:

- **Selector:** A selector is an HTML tag at which style will be applied. This could be any

tag like <h1> or <table> etc.

- **Property:** A property is a type of attribute of HTML tag. Put simply, all the HTML attributes are converted into CSS properties. They could be color or border etc.
- **Value:** Values are assigned to properties. For example color property can have value either red or #F1F1F1 etc.

You can put CSS Style Rule Syntax as follows:

```
selector { property: value }
```

Example: You can define a table border as follows:

```
table{ border :1px solid #C00; }
```

Here table is a selector and border is a property and given value 1px solid #C00 is the value of that property. You can define selectors in various simple ways based on your comfort.

The Type Selectors:

This is the same selector we have seen above. Again one more example to give a color to all level 1 headings :

```
h1 {
 color: #36CFFF;
}
```

The Universal Selectors:

Rather than selecting elements of a specific type, the universal selector quite simply matches the name of any element type :

```
* {
 color: #000000;
}
```

This rule renders the content of every element in our document in black.

The Descendant Selectors:

Suppose you want to apply a style rule to a particular element only when it lies inside a particular element. As given in the following example, style rule will apply to element only when it lies inside <ul> tag.

```
ul em {
 color: #000000;
}
```

The Class Selectors:

You can define style rules based on the class attribute of the elements. All the elements having that class will be formatted according to the defined rule.

```
.black {
 color: #000000;
}
```

This rule renders the content in black for every element with class attribute set to black in our document. You can make it a bit more particular. For example:

```
h1.black {
 color: #000000;
}
```

This rule renders the content in black for only <h1> elements with class attribute set to black. You can apply more than one class selectors to given element. Consider the following example :

```
<p class="center bold">
This para will be styled by the classes center and bold.
</p>
```

The ID Selectors:

You can define style rules based on the id attribute of the elements. All the elements having that id will be formatted according to the defined rule.

```
#black {
 color: #000000;
}
```

This rule renders the content in black for every element with id attribute set to black in our document. You can make it a bit more particular. For example:

```
h1#black {
 color: #000000;
}
```

This rule renders the content in black for only <h1> elements with id attribute set to black.

The true power of id selectors is when they are used as the foundation for descendant selectors,

For example:

```
#black h2 {
 color: #000000;
```

```
}
```

In this example all level 2 headings will be displayed in black color only when those headings will lie with in tags having id attribute set to black.

The Child Selectors:

You have seen descendant selectors. There is one more type of selectors which is very similar to descendants but have different functionality. Consider the following example:

```
body > p {
 color: #000000;
}
```

This rule will render all the paragraphs in black if they are direct child of <body> element. Other paragraphs put inside other elements like or <td> etc. would not have any effect of this rule.

The Attribute Selectors:

You can also apply styles to HTML elements with particular attributes. The style rule below will match all input elements that has a type attribute with a value of text:

```
input[type="text"]{
 color: #000000;
}
```

The advantage to this method is that the <input type="submit" /> element is unaffected, and the color applied only to the desired text fields.

There are following rules applied to attribute selector.

- **p[lang]** - Selects all paragraph elements with a lang attribute.
- **p[lang="fr"]** - Selects all paragraph elements whose lang attribute has a value of exactly "fr".
- **p[lang~="fr"]** - Selects all paragraph elements whose lang attribute contains the word "fr".
- **p[lang|="en"]** - Selects all paragraph elements whose lang attribute contains values that are exactly "en", or begin with "en-".

Multiple Style Rules:

You may need to define multiple style rules for a single element. You can define these rules to combine multiple properties and corresponding values into a single block as defined in the following example:

```
h1 {
color: #36C;
```

```
font-weight: normal;
letter-spacing: .4em;
margin-bottom: 1em;
text-transform: lowercase;
}
```

Here all the property and value pairs are separated by a semi colon (;). You can keep them in

a single line or multiple lines. For better readability we keep them into separate lines.

Grouping Selectors:

You can apply a style to many selectors if you like. Just separate the selectors with a comma as given in the following example:

```
h1, h2, h3 {
color: #36C;
font-weight: normal;
letter-spacing: .4em;
margin-bottom: 1em;
text-transform: lowercase;
}
```

This define style rule will be applicable to h1, h2 and h3 element as well. The order of the list is irrelevant. All the elements in the selector will have the corresponding declarations applied to them.

You can combine various class selectors together as shown below:

```
#content, #footer, #supplement {
position: absolute;
left: 510px;
width: 200px;
}
```

3.2. Levels of Style Sheets

There are four ways to associate styles with your HTML document. Most commonly used methods are inline CSS and External CSS.

3.2.1. Embedded CSS - The <style> Element:

You can put your CSS rules into an HTML document using the <style> element. This tag is placed inside <head>...</head> tags. Rules defined using this syntax will be applied to all the elements available in the document. Here is the generic syntax:

```
<head>
<style type="text/css" media="...">
.....
</style>
</head>
```

Attributes:

Attributes associated with <style> elements are:

Attribute	Value	Description
type	text/css	Specifies the style sheet language as a content-type (MIME type). This is required attribute.
media	screen tty tv projection handheld print braille aural all	Specifies the device the document will be displayed on. Default value is all. This is optional attribute.

Example:

Following is the example of embed CSS based on above syntax:

```
<head>
<style type="text/css" media="all">
h1{
```

```
color: #36C;
}
</style>
</head>
```

3.2.2. Inline CSS - The style Attribute:

You can use style attribute of any HTML element to define style rules. These rules will be applied to that element only. Here is the generic syntax:

```
<element style="...style rules....">
```

Attributes:

Attribute	Value	Description
style	style rules	The value of style attribute is a combination of style declarations separated by semicolon (;).

Example:

Following is the example of inline CSS based on above syntax:

```
<h1 style ="color:#36C;"> This is inline CSS </h1>
```

3.2.3. External CSS - The <link> Element:

The <link> element can be used to include an external stylesheet file in your HTML document.

An external style sheet is a separate text file with .css extension. You define all the Style rules within this text file and then you can include this file in any HTML document using <link> element.

Here is the generic syntax of including external CSS file:

```
<head>
<link type="text/css" href="..." media="..." />
</head>
```

Attributes:

Attributes associated with <style> elements are:

Attribute	Value	Description
type	text/css	Specifies the style sheet language as a content-type (MIME type). This attribute is required.
href	URL	Specifies the style sheet file having Style rules. This attribute is a required.
media	screen tty tv projection handheld print braille aural all	Specifies the device the document will be displayed on. Default value is all. This is optional attribute.

Example:

Consider a simple style sheet file with a name mystyle.css having the following rules:

```
h1, h2, h3 {
color: #36C;
font-weight: normal;
letter-spacing: .4em;
margin-bottom: 1em;
text-transform: lowercase;
}
```

Now you can include this file mystyle.css in any HTML document as follows:

```
<head>
<link type="text/css" href="mystyle.css" media="all" />
```

```
</head>
```

3.2.4. Imported CSS - @import Rule:

@import is used to import an external stylesheet in a manner similar to the <link> element.

Here is the generic syntax of @import rule.

```
<head>
<@import "URL";
</head>
```

Here URL is the URL of the style sheet file having style rules. You can use another syntax as well:

```
<head>
<@import url("URL");
</head>
```

Example:

```
<head>
@import "mystyle.css";
</head>
```

3.3. CSS Rules Overriding

We have discussed four ways to include style sheet rules in a an HTML document. Here is the rule to override any Style Sheet Rule.

- Any inline style sheet takes highest priority. So it will override any rule defined in <style>...</style> tags or rules defined in any external style sheet file.
- Any rule defined in <style>...</style> tags will override rules defined in any external style sheet file.
- Any rule defined in external style sheet file takes lowest priority and rules defined in this file will be applied only when above two rules are not applicable.

3.4. CSS Comments:

Many times you may need to put additional comments in your style sheet blocks. So it is very easy to comment any part in style sheet. You simple put your comments inside

```
/*.....this is a comment in style sheet.....*/.
```

You can use /* ....*/ to comment multi-line blocks in similar way you do in C and C++ programming languages.

Example:

```
/* This is an external style sheet file */
h1, h2, h3 {
color: #36C;
font-weight: normal;
letter-spacing: .4em;
margin-bottom: 1em;
text-transform: lowercase;
}
/* end of style rules. */
```

3.5. CSS - Measurement Units

CSS supports a number of measurements including absolute units such as inches, centimeters, points, and so on, as well as relative measures such as percentages and em units.

Examples:

Unit	Description	Example
%	Defines a measurement as a percentage relative to another value, typically an enclosing element.	p {font-size: 16pt; line-height: 125%;}
cm	Defines a measurement in centimeters.	div {margin-bottom: 2cm;}
em	A relative measurement for the height of a font in em spaces. Because an em unit is equivalent to the size of a given font, if you assign a font to 12pt, each "em" unit would be 12pt; thus, 2em would be 24pt.	p {letter-spacing: 7em;}
ex	This value defines a measurement relative to a font's x-height. The x-height is determined by the height	p {font-size: 24pt; line-height: 3ex;}

	of the font's lowercase letter x.	
in	Defines a measurement in inches.	p {word-spacing: .15in;}
mm	Defines a measurement in millimeters.	p {word-spacing: 15mm;}
pc	Defines a measurement in picas. A pica is equivalent to 12 points; thus, there are 6 picas per inch.	p {font-size: 20pc;}
pt	Defines a measurement in points. A point is defined as 1/72nd of an inch.	body {font-size: 18pt;}
px	Defines a measurement in screen pixels.	p {padding: 25px;}

3.6. CSS – Colors

CSS uses color values to specify a color. Typically, these are used to set a color either for the foreground of an element(i.e., its text) or else for the background of the element. They can also be used to affect the color of borders and other decorative effects.

You can specify your color values in various formats. Following table tells you all possible formats:

Format	Syntax	Example
Hex Code	#RRGGBB	p{color:#FF0000;}
Short Hex Code	#RGB	p{color:#6A7;}
RGB %	rgb(rrr%,ggg%,bbb%)	p{color:rgb(50%,50%,50%);}
RGB Absolute	rgb(rrr,ggg,bbb)	p{color:rgb(0,0,255);}
keyword	aqua, black, etc.	p{color:teal;}

3.7. The CSS Box Model

All HTML elements can be considered as boxes. In CSS, the term "box model" is used when talking about design and layout.

The CSS box model is essentially a box that wraps around HTML elements, and it consists of: margins, borders, padding, and the actual content.

The box model allows us to add a border around elements, and to define space between elements.

The image below illustrates the box model:

Explanation of the different parts:

- **Content** - The content of the box, where text and images appear.
- **Padding** - Clears an area around the content. The padding is transparent.
- **Border** - A border that goes around the padding and content.
- **Margin** - Clears an area outside the border. The margin is transparent.