CSS

cascading style sheets

tutorialspoint

SIMPLYEASYLEARNING

www.tutorialspoint.com

https://www.facebook.com/tutorialspointindia

https://twitter.com/tutorialspoint

About the Tutorial

CSS is used to control the style of a web document in a simple and easy way. CSS stands for Cascading Style Sheets. This tutorial covers both the versions CSS1 and CSS2 and gives a complete understanding of CSS, starting from its basics to advanced concepts.

Audience

This tutorial will help both students as well as professionals who want to make their websites or personal blogs more attractive.

Prerequisites

You should be familiar with:

- Basic word processing using any text editor.
- How to create directories and files.
- How to navigate through different directories.
- Internet browsing using popular browsers like Internet Explorer or Firefox.
- Developing simple Web Pages using HTML or XHTML.

If you are new to HTML and XHTML, then we would suggest you to go through our HTML Tutorial or XHTML Tutorial first.

Copyright & Disclaimer

Table of Contents

1. CSS – OVERVIEW

What is CSS?

Cascading **S**tyle **S**heets, 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 a powerful control over the presentation of an HTML document. Most commonly, CSS is combined with the markup languages HTML or XHTML.

Advantages of CSS

- **CSS saves time** - You can write CSS once and then reuse the 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 it 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 the 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 a far better look to your HTML page in comparison to 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 cellphones or for printing.
- **Global web standards** – Now HTML attributes are being deprecated and it is being recommended to use CSS. So it's a good idea to start using CSS in all the HTML pages to make them compatible with future browsers.

Who Creates and Maintains CSS?

CSS is created and maintained through a group of people within the W3C called the CSS Working Group. The CSS Working Group creates documents called **specifications**. When a specification has been discussed and officially ratified by the W3C members, it becomes a recommendation.

These ratified specifications are called recommendations because the W3C has no control over the actual implementation of the language. Independent companies and organizations create that software.

NOTE: The World Wide Web Consortium or W3C is a group that makes recommendations about how the Internet works and how it should evolve.

CSS Versions

Cascading Style Sheets level 1 (CSS1) came out of W3C as a recommendation in December 1996. This version describes the CSS language as well as a simple visual formatting model for all the HTML tags.

CSS2 became a W3C recommendation in May 1998 and builds on CSS1. This version adds support for media-specific style sheets e.g. printers and aural devices, downloadable fonts, element positioning and tables.

2. CSS – SYNTAX

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 a 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*, *border*, etc.
- **Value:** Values are assigned to properties. For example, *color* property can have the 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 the given value *1px solid #C00* is the value of that property.

You can define selectors in various simple ways based on your comfort. Let me put these selectors one by one.

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, the style rule will apply to element only when it lies inside the <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 a 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 when those headings will lie within tags having *id* attribute set to *black*.

The Child Selectors

You have seen the descendant selectors. There is one more type of selector, 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 a direct child of the <body> element. Other paragraphs put inside other elements like or <td> 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 the input elements having 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 **semicolon (;)**. You can keep them in a single line or multiple lines. For better readability, we keep them in separate lines.

For a while, don't bother about the properties mentioned in the above block. These properties will be explained in the coming chapters and you can find the complete detail about properties in CSS References.

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 the various *class* selectors together as shown below:

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

3. CSS – INCLUSION

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

Embedded CSS - The <style> Element

You can put your CSS rules into an HTML document using the <style> element. This tag is placed inside the <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 Rules
.............
</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 a required attribute.
media	screen tty tv projection handheld print braille	Specifies the device, the document will be displayed on. Default value is *all*. This is an optional attribute.

	aural all	

Example

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

```
<head>
<style type="text/css" media="all">
h1{
color: #36C;
}
</style>
</head>
```

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 the above syntax:

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

It will produce the following result:

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

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	Specifies the device the document will be displayed on. Default value is *all*. This is an optional attribute.

	aural all	

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>
```

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

Following is the example showing you how to import a style sheet file into an HTML document:

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

CSS Rules Overriding

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

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

Handling Old Browsers

There are still many old browsers who do not support CSS. So, we should take care while writing our Embedded CSS in an HTML document. The following snippet shows how to use comment tags to hide CSS from older browsers:

```
<style type="text/css">
<!--
body, td {
   color: blue;
}
-->
</style>
```

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 the style sheet. You can simply 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. */
```

4. CSS – MEASUREMENT UNITS

Before we start the actual exercise, we would like to give a brief idea about the 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. You need these values while specifying various measurements in your Style rules e.g. **border="1px solid red"**.

We have listed out all the CSS Measurement Units along with proper 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 of the font's lowercase letter x.	p {font-size: 24pt; line-height: 3ex;}
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;}

5. 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 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 lists all the 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;}

These formats are explained in more detail in the following sections:

CSS Colors - Hex Codes

A hexadecimal is a 6 digit representation of a color. The first two digits (RR) represent a red value, the next two are a green value (GG), and the last are the blue value (BB).

A hexadecimal value can be taken from any graphics software like Adobe Photoshop, Jasc Paintshop Pro, or even using Advanced Paint Brush.

Each hexadecimal code will be preceded by a pound or hash sign '#'. Following are the examples to use Hexadecimal notation.

Color	Color HEX

	#000000
	#FF0000
	#00FF00
	#0000FF
	#FFFF00
	#00FFFF
	#FF00FF
	#C0C0C0
	#FFFFFF

CSS Colors - Short Hex Codes

This is a shorter form of the six-digit notation. In this format, each digit is replicated to arrive at an equivalent six-digit value. For example: #6A7 becomes #66AA77.

A hexadecimal value can be taken from any graphics software like Adobe Photoshop, Jasc Paintshop Pro or even using Advanced Paint Brush.

Each hexadecimal code will be preceded by a pound or hash sign #. Following are the examples to use the Hexadecimal notation.

End of ebook preview
If you liked what you saw…
Buy it from our store @ **https://store.tutorialspoint.com**