## What is CSS?

- **CSS** stands for **C**ascading **S**tyle **S**heets
- Styles define **how to display** HTML elements
- Styles were added to HTML 4.0 **to solve a problem**
- **External Style Sheets** can save a lot of work
- External Style Sheets are stored in **CSS files**

## Styles Solved a Big Problem

HTML was never intended to contain tags for formatting a document.

HTML was intended to define the content of a document, like:

<h1>This is a heading</h1>

This is a paragraph.

When tags like <font>, and color attributes were added to the HTML 3.2 specification, it started a nightmare for web developers. Development of large web sites, where fonts and color information were added to every single page, became a long and expensive process.

To solve this problem, the World Wide Web Consortium (W3C) created CSS.

In HTML 4.0, all formatting could be removed from the HTML document, and stored in a separate CSS file.

All browsers support CSS today.

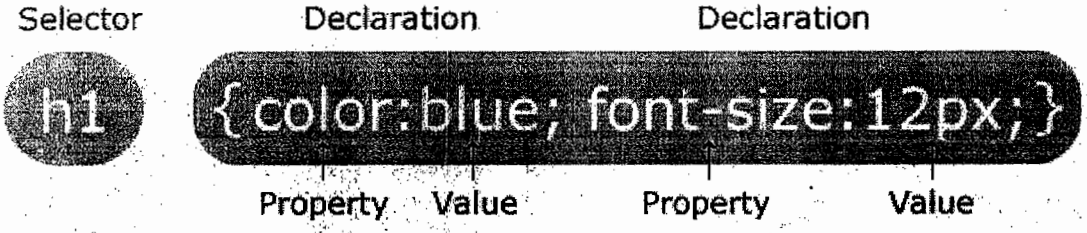
## CSS Saves a Lot of Work!

CSS defines HOW HTML elements are to be displayed.

Styles are normally saved in external .css files. External style sheets enable you to change the appearance and layout of all the pages in a Web site, just by editing one single file!

## CSS Syntax

A CSS rule has two main parts: a selector, and one or more declarations:



The selector is normally the HTML element you want to style.

Each declaration consists of a property and a value.

The property is the style attribute you want to change. Each property has a value.

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. 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, 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 ingle 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: lem;
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;
}
```

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 <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 required attribute. |
| media | screen<br>tty<br>tv<br>projection<br>handheld<br>print<br>braille<br>aural<br>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>
```

## 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:

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

## 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="..." href="mystyle.css"   />
</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<br>tty<br>tv<br>projection<br>handheld<br>print<br>braille<br>aural<br>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" rel="stylesheet" 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 HTML document:

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

## 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.

## 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 you can 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 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: lem;
text-transform: lowercase;
}
/* end of style rules. */
```

## CSS Units

Before we start actual exercise, I 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 alogwith 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;} |

## 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;} |

## Setting Backgrounds using CSS

You can set following background properties of an element:

- The **background-color** property is used to set the background color of an element.
- The **background-image** property is used to set the background image of an element.
- The **background-repeat** property is used to control the repetition of an image in the background.
- The **background-position** property is used to control the position of an image in the background.
- The **background-attachment** property is used to control the scrolling of an image in the background.
- The **background** property is used as shorthand to specify a number of other background properties.

## Set the background color:

Following is the example which demonstrates how to set the background color for an element.

```
<p style="background-color:yellow;">
    This text has a yellow background color.
</p>
```

## Set the background image:

Following is the example which demonstrates how to set the background image for an element.

```
<table style="background-image:url(/images/pattern1.gif);">
<tr>
   <td>
      This table has background image set.
   </td>
</tr>
</table>
```

## Repeat the background image:

Following is the example which demonstrates how to repeat the background image if image is small. You can use *no-repeat* value for *background-repeat* property if you don't want to repeat an image, in this case image will display only once.

By default *background-repeat* property will have *repeat* value.

```
<table style="background-image:url(/images/pattern1.gif);
              background-repeat: repeat;">
<tr>
  <td>
      This table has background image which repeats multiple times.
  </td>
</tr>
</table>
```

Following is the example which demonstrates how to repeat the background image vertically.

```
<table style="background-image:url(/images/pattern1.gif);
              background-repeat: repeat-y;">
<tr>
  <td>
     This table has background image set which will repeat vertically.
  </td>
</tr>
</table>
```

Following is the example which demonstrates how to repeat the background image horizontally.

```
<table style="background-image:url(/images/pattern1.gif);
              background-repeat: repeat-x;">
<tr><td>
This table has background image set which will repeat horizontally.
</td></tr>
</table>
```

Following is the example which demonstrates how to **not repeat** the background image.

```
<table style="background-image:url(/images/pattern1.gif); background-repeat: no-repeat;">
<tr>
 <td>
    This table has background image set which will repeat horizontally.
 </td>
</tr>
</table>
```

## Set the background image position:

Following is the example which demonstrates how to set the background image position 100 pixels away from the left side.

```
<table style="background-image:url(/images/pattern1.gif);
              background-position:100px;">
<tr>
   <td>
      Background image positioned 100 pixels away from the left.
   </td>
</tr>
</table>
```

Following is the example which demonstrates how to set the background image position 100 pixels away from the left side and 200 pixels down from the top.

```
<table style="background-image:url(/images/pattern1.gif);
              background-position:100px 200px;">
<tr>
   <td>
      This table has background image positioned 100
      pixels away from the left and 200 pixels from the top.
   </td>
</tr>
</table>
```

## Set the background attachment:

Background attachment determines whether a background image is fixed or scrolls with the rest of the page.

Following is the example which demonstrates how to set the fixed background image.

```
<p style="background-image:url(/images/pattern1.gif);
          background-attachment:fixed;">
   This parapgraph has fixed background image.
</p>
```

Following is the example which demonstrates how to set the scrolling background image.

```
<p style="background-image:url(/images/pattern1.gif);
          background-attachment:scroll;">
   This parapgraph has scrolling background image.
</p>
```

## Background - Shorthand property

As you can see from the examples above, there are many properties to consider when dealing with backgrounds.

To shorten the code, it is also possible to specify all the properties in one single property. This is called a shorthand property.

The shorthand property for background is simply "background":

```
background:  color position size repeat origin clip attachment image;
```

## Example

```
body {
        background:#ffffff url('img_tree.png') no-repeat right top;
}
```

When using the shorthand property the order of the property values are:

- background-color
- background-image
- background-repeat
- background-attachment
- background-position

```
<html>
<head>
<style type="text/css">
body
{
margin-left:200px;
background:#5d9ab2 url('images/tree.png') no-repeat top left;
}

.container
{
text-align:center;
}

.center_div
{
border:1px solid gray;
margin-left:auto;
margin-right:auto;
width:90%;
background-color:#d0f0f6;
text-align:left;
padding:8px;
}
</style>
</head>
```

```
<body>
<div class="container">
<div class="center_div">
<h1>Hello World!</h1>
<p>This example contains some advanced CSS methods you may
not have learned yet. But, we will explain these methods in a
later chapter in the tutorial.</p>
</div>
</div>
</body>
</html>
```

It does not matter if one of the property values is missing, as long as the ones that are present are in this order.

## Setting Fonts using CSS

You can set following font properties of an element:

- The **font-family** property is used to change the face of a font.
- The **font-style** property is used to make a font italic or oblique.
- The **font-variant** property is used to create a small-caps effect.
- The **font-weight** property is used to increase or decrease how bold or light a font appears.
- The **font-size** property is used to increase or decrease the size of a font.
- The **font** property is used as shorthand to specify a number of other font properties.

### CSS Font Families

| Generic family | Font family | Description |
|---|---|---|
| Serif | Times New Roman<br>Georgia | Serif fonts have small lines at the ends on some characters |
| Sans-serif | Arial<br>Verdana | "Sans" means without - these fonts do not have the lines at the ends of characters |
| Monospace | Courier New | All monospace characters have the same width |

| | | |
|---|---|---|
| | Lucida Console | |

**Difference Between Serif and Sans-serif Fonts**


On computer screens, sans-serif fonts are considered easier to read than serif fonts.

## Font Family

The font family of a text is set with the **font-family** property.

The **font-family** property should hold several font names as a "fallback" system. If the browser does not support the first font, it tries the next font.

Start with the font you want, and end with a generic family, to let the browser pick a similar font in the generic family, if no other fonts are available.

**Note**: If the name of a font family is more than one word, it must be in quotation marks, like **font-family: "Times New Roman"**.

More than one font family is specified in a comma-separated list:

## Example

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

Following is the example which demonstrates how to set the font family of an element. Possible value could be any font family name.

```
<p style="font-family:georgia,garamond,serif;">
   This text is rendered in either georgia, garamond, or the default
   serif font depending on which font  you have at your system.
</p>
```

## Set the font style:

Following is the example which demonstrates how to set the font style of an element. Possible values are *normal, italic and oblique.*

```
<p style="font-style:italic;">
   This text will be rendered in italic style
</p>
```

## Set the font variant:

Following is the example which demonstrates how to set the font variant of an element. Possible values are *normal and small-caps.*

```
<html>
<head>
<style type="text/css">
   p.normal {font-variant:normal;}
   p.small {font-variant:small-caps;}
</style>
</head>

<body>
   <p class="normal">My name is Hege Refsnes.</p>
   <p class="small">My name is Hege Refsnes.</p>
</body>

</html>
```

## Set the font weight:

Following is the example which demonstrates how to set the font weight of an element. The font-weight property provides the functionality to specify how bold a font is. Possible values could be *normal, bold, bolder, lighter, 100, 200, 300, 400, 500, 600, 700, 800, 900.*

```
<p style="font-weight:bold;">
   This font is bold.
</p>
<p style="font-weight:bolder;">
   This font is bolder.
</p>
<p style="font-weight:900;">
   This font is 900 weight.
</p>
```

## Set the font size:

Following is the example which demonstrates how to set the font size of an element. The font-size property is used to control the size of fonts. Possible values could be *xx-small, x-small, small, medium, large, x-large, xx-large, smaller, larger, size in pixels or in %*

```
<p style="font-size:20px;">
    This font size is 20 pixels
</p>
<p style="font-size:small;">
    This font size is small
</p>
<p style="font-size:large;">
    This font size is large
</p>
```

## Shorthand property :

You can use the *font* property to set all the font properties at once. For example:

```
<p style="font:italic small-caps bold 15px georgia;">
    Applying all the properties on the text at once.
</p>
```

## Manipulating Text using CSS

You can set following text properties of an element:

- The **color** property is used to set the color of a text.
- The **direction** property is used to set the text direction.
- The **letter-spacing** property is used to add or subtract space between the letters that make up a word.
- The **word-spacing** property is used to add or subtract space between the words of a sentence.
- The **text-indent** property is used to indent the text of a paragraph.
- The **text-align** property is used to align the text of a document.
- The **text-decoration** property is used to underline, overline, and strikethrough text.
- The **text-transform** property is used to capitalize text or convert text to uppercase or lowercase letters.
- The **white-space** property is used to control the flow and formatting of text.
- The **text-shadow** property is used to set the text shadow around a text.

## Set the text color:

Following is the example which demonstrates how to set the text color. Possible value could be any color name in any valid format.

```
<p style="color:red;">
    This text will be written in red.
</p>
```

## Set the text direction :

Following is the example which demonstrates how to set the direction of a text. Possible values are *ltr or rtl.*

```
<p style="direction:rtl;">
    This text will be renedered from right to left
</p>
```

## Set the space between characters:

Following is the example which demonstrates how to set the space between characters. Possible values are *normal or a number specifying space..*

```
<p style="letter-spacing:5px;">
    This text is having space between letters.
</p>
```

## Set the space between words:

Following is the example which demonstrates how to set the space between words. Possible values are *normal or a number specifying space..*

```
<p style="word-spacing:5px;">
     This text is having space between words.
</p>
```

## Set the text indent:

Following is the example which demonstrates how to indent the first line of a paragraph. Possible values are *% or a number specifying indent space..*

```
<p style="text-indent:1cm;">
    This text will have first line indented by 1cm
    and this line will remain at its actual position
    this is done by CSS text-indent property.
</p>
```

## Set the text alignment:

Following is the example which demonstrates how to align a text. Possible values are *left, right, center, justify.*.

```
<p style="text-align:right;">
    This will be right aligned.
</p>
<p style="text-align:center;">
    This will be center aligned.
</p>
<p style="text-align:left;">
    This will be left aligned.
</p>
```

## Decorating the text:

Following is the example which demonstrates how to decorate a text. Possible values are *none, underline, overline, line-through, blink.*.

```
<p style="text-decoration:underline;">
    This will be underlined
</p>
<p style="text-decoration:line-through;">
    This will be striked through.
</p>
<p style="text-decoration:overline;">
    This will have a over line.
</p>
<p style="text-decoration:blink;">
    This text will have blinking effect
</p>
```

## Set the text cases:

Following is the example which demonstrates how to set the cases for a text. Possible values are *none, capitalize, uppercase, lowercase.*.

```
<p style="text-transform:capitalize;">
     This will be capitalized
</p>
<p style="text-transform:uppercase;">
     This will be in uppercase
</p>
<p style="text-transform:lowercase;">
     This will be in lowercase
</p>
```

### Set the white space between text:

Following is the example which demonstrates how white space inside an element is handled. Possible values are *normal, pre, nowrap.*

```
<p style="white-space:pre;">
    This text has a line break and the white-space pre setting tells the browser to
    honor it just like the HTML pre tag.
</p>
```

### Set the text shadow:

Following is the example which demonstrates how to set the shadow around a text. This may not be supported by all the browsers.

```
<p style="text-shadow:4px 4px 8px blue;font-size:50px;">
    If your browser supports the CSS text-shadow property,
    this text will have a  blue shadow.
</p>
```

# Styling Links

Links can be styled with any CSS property (e.g. color, font-family, background, etc.).

Special for links are that they can be styled differently depending on what state they are in.

The four links states are:

- a:link - a normal, unvisited link
- a:visited - a link the user has visited
- a:hover - a link when the user mouses over it
- a:active - a link the moment it is clicked

### Example

```
a:link {color:#FF0000;}    /* unvisited link */
a:visited {color:#00FF00;} /* visited link */
a:hover {color:#FF00FF;} /* mouse over link */
a:active {color:#0000FF;} /* selected link */
```

### Example

```
a:link {text-decoration:none;}
a:visited {text-decoration:none;}
a:hover {text-decoration:underline;}
a:active {text-decoration:underline;}
```

**Example**

```
a:link {background-color:#B2FF99;}
a:visited {background-color:#FFFF85;}
a:hover {background-color:#FF704D;}
a:active {background-color:#FF704D;}
```

When setting the style for several link states, there are some order rules:

- a:hover MUST come after a:link and a:visited
- a:active MUST come after a:hover

## CSS - Lists

Lists are very helpful in conveying a set of either numbered or bulleted points. This teaches you how to control list type, position, style etc. using CSS

There are following five CSS properties which can be used to control lists:

- The **list-style-type** Allows you to control the shape or appearance of the marker.
- The **list-style-position** Specifies whether a long point that wraps to a second line should align with the first line or start underneath the start of the marker.
- The **list-style-image** Specifies an image for the marker rather than a bullet point or number.
- The **list-style** Serves as shorthand for the preceding properties.
- The **marker-offset** Specifies the distance between a marker and the text in the list.

Now we will see how to use these properties with examples.

## The list-style-type Property:

The *list-style-type* property allows you to control the shape or style of bullet point (also known as a marker) in the case of unordered lists, and the style of numbering characters in ordered lists.

Here are the values which can be used for an unordered list:

| Value | Description |
|---|---|
| None | NA |
| disc (default) | A filled-in circle |
| Circle | An empty circle |

| | |
|---|---|
| Square | A filled-in square |

Here are the values which can be used for an ordered list:

| Value | Description | Example |
|---|---|---|
| Decimal | Number | 1,2,3,4,5 |
| decimal-leading-zero | 0 before the number | 01, 02, 03, 04, 05 |
| lower-alpha | Lowercase alphanumeric characters | a, b, c, d, e |
| upper-alpha | Uppercase alphanumeric characters | A, B, C, D, E |
| lower-roman | Lowercase Roman numerals | i, ii, iii, iv, v |
| upper-roman | Uppercase Roman numerals | I, II, III, IV, V |
| lower-greek | The marker is lower-greek | alpha, beta, gamma |
| lower-latin | The marker is lower-latin | a, b, c, d, e |
| upper-latin | The marker is upper-latin | A, B, C, D, E |
| Hiragana | The marker is hiragana | a, i, u, e, o, ka, ki |
| Katakana | The marker is katakana | A, I, U, E, O, KA, KI |
| hiragana-iroha | The marker is hiragana-iroha | i, ro, ha, ni, ho, he, to |
| katakana-iroha | The marker is katakana-iroha | I, RO, HA, NI, HO, HE, TO |

Here is the example:

```
<ul style="list-style-type:circle;">
  <li>Maths</li>
```

```
  <li>Social Science</li>
  <li>Physics</li>
</ul>

<ul style="list-style-type:square;">
   <li>Maths</li>
   <li>Social Science</li>
   <li>Physics</li>
</ul>

<ol style="list-style-type:decimal;">
   <li>Maths</li>
   <li>Social Science</li>
   <li>Physics</li>
</ol>

<ol style="list-style-type:lower-alpha;">
   <li>Maths</li>
   <li>Social Science</li>
   <li>Physics</li>
</ol>

<ol style="list-style-type:lower-roman;">
   <li>Maths</li>
   <li>Social Science</li>
   <li>Physics</li>
</ol>
```

## The list-style-position Property:

The *list-style-position* property indicates whether the marker should appear inside or outside of the box containing the bullet points. It can have one the two values:

| Value | Description |
|---|---|
| None | NA |
| Inside | If the text goes onto a second line, the text will wrap underneath the marker. It will also appear indented to where the text would have started if the list had a value of outside. |
| outside | If the text goes onto a second line, the text will be aligned with the start of the first line (to the right of the bullet). |

Here is the example:

```
<ul style="list-style-type:circle; list-stlye-position:outside;">
```

```
    <li>Maths</li>
    <li> If the text goes onto a second line, the text will be aligned with the start of
         the first line (to the right of the bullet </li>
    <li>Physics</li>
</ul>

<ul style="list-style-type:square;list-style-position:inside;">
    <li>Maths</li>
    <li> If the text goes onto a second line, the text will be aligned with the start of
         the first line (to the right of the bullet </li>
    <li>Physics</li>
</ul>

<ol style="list-style-type:decimal;list-stlye-position:outside;">
    <li>Maths</li>
    <li>Social Science</li>
    <li>Physics</li>
</ol>

<ol style="list-style-type:lower-alpha;list-style-position:inside;">
    <li>Maths</li>
    <li>Social Science</li>
    <li>Physics</li>
</ol>
```

## The list-style-image Property:

The *list-style-image* allows you to specify an image so that you can use your own bullet style. The syntax is as follows, similar to the background-image property with the letters url starting the value of the property followed by the URL in brackets. If it does not find given image then default bullets are used.

Here is the example:

```
<ul style="list-style-image: url(images/mobile.gif);" >
    <li >Maths</li>
    <li>Social Science</li>
    <li>Physics</li>
</ul>

<ol style="list-style-image: url(images/mobile.gif);" >
    <li style="list-style-image: url(images/rose.gif);" >Maths</li>
    <li>Social Science</li>
    <li>Physics</li>
</ol>
```

## The list-style Property:

The *list-style* allows you to specify all the list properties into a single expression. These properties can appear in any order.

Here is the example:

```
<ul style="list-style: inside square;">
   <li>Maths</li>
   <li>Social Science</li>
   <li>Physics</li>
</ul>

<ol style="list-style: outside upper-alpha;">
   <li>Maths</li>
   <li>Social Science</li>
   <li>Physics</li>
</ol>
```

## The marker-offset Property:

The *marker-offset* property allows you to specify the distance between the marker and the text relating to that marker. Its value should be a length as shown in the following example:

Unfortunately, however, this property is not supported in IE 6 or Netscape 7.

Here is the example:

```
<ul style="list-style: inside square; marker-offset:2em;">
   <li>Maths</li>
   <li>Social Science</li>
   <li>Physics</li>
</ul>

<ol style="list-style: outside upper-alpha; marker-offset:2cm;">
   <li>Maths</li>
   <li>Social Science</li>
   <li>Physics</li>
</ol>
```

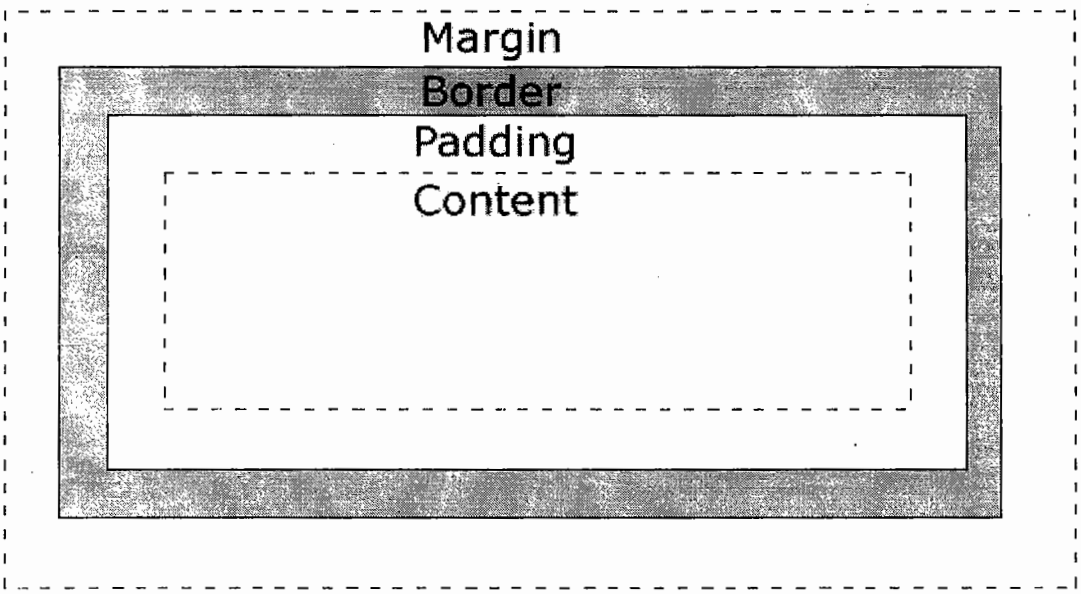
## The CSS Box Model

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 place a border around elements and space elements in relation to other elements.

The image below illustrates the box model:




Explanation of the different parts:

- **Margin** - Clears an area around the border. The margin does not have a background color, it is completely transparent
- **Border** - A border that goes around the padding and content. The border is affected by the background color of the box
- **Padding** - Clears an area around the content. The padding is affected by the background color of the box
- **Content** - The content of the box, where text or images appear

In order to set the width and height of an element correctly in all browsers, you need to know how the box model works.

## Width and Height of an Element

**Important:** When you set the width and height properties of an element with CSS, you just set the width and height of the **content area**. To calculate the full size of an element, you must also add the padding, borders and margins.

The total width of the element in the example below is 300px:

```
width:250px;
padding:10px;
border:5px solid gray;
margin:10px;
```

Let's do the math:
250px (width)
+ 20px (left and right padding)
+ 10px (left and right border)
+ 20px (left and right margin)
= 300px

Assume that you had only 250px of space. Let's make an element with a total width of 250px:

**Example**

```
width:220px;
padding:10px;
border:5px solid gray;
margin:0px;
```

The total width of an element should be calculated like this:

Total element width = width + left padding + right padding + left border + right border + left margin + right margin

The total height of an element should be calculated like this:

Total element height = height + top padding + bottom padding + top border + bottom border + top margin + bottom margin

## Browsers Compatibility Issue

The example above does not display properly in IE8 and earlier versions.

IE8 and earlier versions includes padding and border in the width, if a **DOCTYPE is NOT declared**.

To fix this problem, just add a DOCTYPE to the HTML page:

**Example**

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
<style type="text/css">
  div.ex
  {
    width:220px;
    padding:10px;
    border:5px solid gray;
    margin:0px;
  }
</style>
</head>
```

## CSS - Borders

The *border* properties allow you to specify how the border of the box representing an element should look. There are three properties of a border you can change

- The **border-color** Specifies the color of a border.
- The **border-style** Specifies whether a border should be solid, dashed line, double line, or one of the other possible values.
- The **border-width** Specifies the width of a border.

Now we will see how to use these properties with examples.

## The border-color Property:

The border-color property allows you to change the color of the border surrounding an element. You can individually change the color of the bottom, left, top and right sides of an element's border using the properties:

- **border-bottom-color** changes the color of bottom border.
- **border-top-color** changes the color of top border.
- **border-left-color** changes the color of left border.
- **border-right-color** changes the color of right border.

Here is the example which shows effect of all these properties:

```
<style type="text/css">
p.example1{
   border:1px solid;
   border-bottom-color:#009900; /* Green */
   border-top-color:#FF0000;    /* Red */
   border-left-color:#330000;   /* Black */
   border-right-color:#0000CC;  /* Blue */
}
p.example2{
   border:1px solid;
   border-color:#009900;        /* Green */
}
</style>
<p class="example1">
This example is showing all borders in different colors.
</p>
<p class="example2">
This example is showing all borders in green color only.
</p>
```

## The border-style Property:

The border-style property allows you to select one of the following styles of border:

- **none:** No border. (Equivalent of border-width:0;)
- **solid:** Border is a single solid line.
- **dotted:** Border is a series of dots.
- **dashed:** Border is a series of short lines.
- **double:** Border is two solid lines.
- **groove:** Border looks as though it is carved into the page.
- **ridge:** Border looks the opposite of groove.
- **inset:** Border makes the box look like it is embedded in the page.
- **outset:** Border makes the box look like it is coming out of the canvas.
- **hidden:** Same as none, except in terms of border-conflict resolution for table elements.

You can individually change the style of the bottom, left, top, and right borders of an element using following properties:

- **border-bottom-style** changes the style of bottom border.
- **border-top-style** changes the style of top border.
- **border-left-style** changes the style of left border.
- **border-right-style** changes the style of right border.

Following is the example to show all these border styles:

```
<p style="border-width:4px; border-style:none;">
This is a border with none width.
</p>
<p style="border-width:4px; border-style:solid;">
This is a solid border.
</p>
<p style="border-width:4px; border-style:dashed;">
This is a dahsed border.
</p>
<p style="border-width:4px; border-style:double;">
This is a double border.
</p>
<p style="border-width:4px; border-style:groove;">
This is a groove border.
</p>
<p style="border-width:4px; border-style:ridge">
This is aridge  border.
</p>
<p style="border-width:4px; border-style:inset;">
This is a inset border.
</p>
<p style="border-width:4px; border-style:outset;">
This is a outset border.
</p>
<p style="border-width:4px; border-style:hidden;">
This is a hidden border.
</p>
<p style="border-width:4px;
            border-top-style:solid;
            border-bottom-style:dashed;
            border-left-style:groove;
            border-right-style:double;">
This is a a border with four different styles.
</p>
```

## The border-width Property:

The border-width property allows you to set the width of an element borders. The value of this property could be either a length in px, pt or cm or it should be set to *thin, medium or thick.*

You can individually change the width of the bottom, top, left, and right borders of an element using the following properties:

- **border-bottom-width** changes the width of bottom border.
- **border-top-width** changes the width of top border.
- **border-left-width** changes the width of left border.
- **border-right-width** changes the width of right border.

Following is the example to show all these border width:

```
<p style="border-width:4px; border-style:solid;">
This is a solid border whose width is 4px.
</p>
<p style="border-width:4pt; border-style:solid;">
This is a solid border whose width is 4pt.
</p>
<p style="border-width:thin; border-style:solid;">
This is a solid border whose width is thin.
</p>
<p style="border-width:medium; border-style:solid;">
This is a solid border whose width is medium;
</p>
<p style="border-width:thick; border-style:solid;">
This is a solid border whose width is thick.
</p>
<p style="border-bottom-width:4px;
            border-top-width:10px;
            border-left-width: 2px;
            border-right-width:15px;
            border-style:solid;">
This is a a border with four different width.
</p>
```

## Border - Shorthand property

As you can see from the examples above, there are many properties to consider when dealing with borders.

To shorten the code, it is also possible to specify all the border properties in one property. This is called a shorthand property.

The shorthand property for the border properties is "border":

**Example**

```
border:5px solid red;
```

When using the border property, the order of the values are:

- border-width
- border-style
- border-color

It does not matter if one of the values above are missing (although, border-style is required), as long as the rest are in the specified order.

## CSS - Margin

The margin property defines the space around an HTML element. It is possible to use negative values to overlap content.

The values of the margin property are not inherited by child elements. Remember that the adjacent vertical margins (top and bottom margins) will collapse into each other so that the distance between the blocks is not the sum of the margins, but only the greater of the two margins or the same size as one margin if both are equal.

There are following four properties to set an element margin.

- The margin A shorthand property for setting the margin properties in one declaration.
- The margin-bottom Specifies the bottom margin of an element.
- The margin-top Specifies the top margin of an element.
- The margin-left Specifies the left margin of an element.
- The margin-right Specifies the right margin of an element.

Now we will see how to use these properties with examples.

## The margin Property:

The margin property allows you set all of the properties for the four margins in one declaration. Here is the syntax to set margin around a paragraph:

```
<style type="text/css">
p {margin: 15px}
all four margins will be 15px

p {margin: 10px 2%}
top and bottom margin will be 10px, left and right margin will be 2% of the total width of the document.

p {margin: 10px 2% -10px}
top margin will be 10px, left and right margin will be 2% of the total width of the document, bottom margin will be -10px

p {margin: 10px 2% -10px auto}
top margin will be 10px, right margin will be 2% of the total width of the document, bottom margin will be -10px, left margin will be set by the browser

</style>
```

Here is the example:

```
<p style="margin: 15px; border:1px solid black;">
all four margins will be 15px
</p>

<p style="margin:10px 2%; border:1px solid black;">
top and bottom margin will be 10px, left and right margin will be 2% of the total width of the document.
</p>

<p style="margin: 10px 2% -10px; border:1px solid black;"> top margin will be 10px, left and right margin will be 2% of the total width of the document, bottom margin will be -10px </p>

<p style="margin: 10px 2% -10px auto; border:1px solid black;"> top margin will be 10px, right margin will be 2% of the total width of the document, bottom margin will be -10px, left margin will be set by the browser
</p>
```

## The margin-bottom Property:

The margin-bottom property allows you set bottom margin of an element. It can have a value in length, % or auto.

Here is the example:

```
<p style="margin-bottom: 15px; border:1px solid black;">
This is a paragraph with a specified bottom margin
</p>

<p style="margin-bottom: 5%; border:1px solid black;">
This is another paragraph with a specified bottom margin in percent
</p>
```

## The margin-top Property:

The margin-top property allows you set top margin of an element. It can have a value in length, % or auto.

Here is the example:

```
<p style="margin-top: 15px; border:1px solid black;">
This is a paragraph with a specified top margin
</p>

<p style="margin-top: 5%; border:1px solid black;">
This is another paragraph with a specified top margin in percent
</p>
```

## The margin-left Property:

The margin-left property allows you set left margin of an element. It can have a value in length, % or auto.

Here is the example:

```
<p style="margin-left: 15px; border:1px solid black;">
This is a paragraph with a specified left margin
</p>

<p style="margin-left: 5%; border:1px solid black;">
This is another paragraph with a specified top margin in percent
</p>
```

## The margin-right Property:

The margin-right property allows you set right margin of an element. It can have a value in length, % or auto.

Here is the example:

```
<p style="margin-right: 15px; border:1px solid black;">
This is a paragraph with a specified right margin
</p>

<p style="margin-right: 5%; border:1px solid black;">
This is another paragraph with a specified right margin in percent
</p>
```

## Margin - Shorthand property

To shorten the code, it is possible to specify all the margin properties in one property. This is called a shorthand property.

The shorthand property for all the margin properties is "margin":

**Example**

```
margin:100px 50px;
```

The margin property can have from one to four values.

- **margin:25px 50px 75px 100px;**
  - top margin is 25px
  - right margin is 50px
  - bottom margin is 75px
  - left margin is 100px
- **margin:25px 50px 75px;**
  - top margin is 25px
  - right and left margins are 50px
  - bottom margin is 75px
- **margin:25px 50px;**
  - top and bottom margins are 25px
  - right and left margins are 50px
- **margin:25px;**
  - all four margins are 25px

## CSS - Paddings

The *padding* property allows you to specify how much space should appear between the content of an element and its border:

There are following five CSS properties which can be used to control lists:

The value of this attribute should be either a length, a percentage, or the word inherits. If the value is inherit it will have the same padding as its parent element. If a percentage is used, the percentage is of the containing box.

You can also set different values for the padding on each side of the box using the following properties:

- The **padding-bottom** Specifies the bottom padding of an element.
- The **padding-top** Specifies the top padding of an element.

- The **padding-left** Specifies the left padding of an element.
- The **padding-right** Specifies the right padding of an element.
- The **padding** Serves as shorthand for the preceding properties.

Now we will see how to use these properties with examples.

## The padding-bottom Property:

The padding-bottom property sets the bottom padding (space) of an element. This can take a value in terms of length of %.

Here is the example:

```
<p style="padding-bottom: 15px; border:1px solid black;">
This is a paragraph with a specified bottom padding
</p>

<p style="padding-bottom: 5%; border:1px solid black;">
This is another paragraph with a specified bottom padding in percent
</p>
```

## The padding-top Property:

The *padding-top* property sets the top padding (space) of an element. This can take a value in terms of length of %.

Here is the example:

```
<p style="padding-top: 15px; border:1px solid black;">
This is a paragraph with a specified top padding
</p>

<p style="padding-top: 5%; border:1px solid black;">
This is another paragraph with a specified top padding in percent
</p>
```

## The padding-left Property:

The *padding-left* property sets the left padding (space) of an element. This can take a value in terms of length of %.

Here is the example:

```
<p style="padding-left: 15px; border:1px solid black;">
This is a paragraph with a specified left padding
</p>
```

```
<p style="padding-left: 15%; border:1px solid black;">
This is another paragraph with a specified left padding in percent
</p>
```

## The padding-right Property:

The *padding-right* property sets the right padding (space) of an element. This can take a value in terms of length of %.

Here is the example:

```
<p style="padding-right: 15px; border:1px solid black;">
This is a paragraph with a specified right padding
</p>

<p style="padding-right: 5%; border:1px solid black;">
This is another paragraph with a specified right padding in percent
</p>
```

## The padding Property:

The *padding* property sets the left, right, top and bottom padding (space).of an element. This can take a value in terms of length of %.

Here is the example:

```
<p style="padding: 15px; border:1px solid black;">
all four padding will be 15px
</p>

<p style="padding:10px 2%; border:1px solid black;">
top and bottom padding will be 10px, left and right padding will be 2% of the total width of the document.
</p>

<p style="padding: 10px 2% 10px; border:1px solid black;"> top padding will be 10px, left and right padding will be 2% of the total width of the document, bottom padding will be 10px </p>

<p style="padding: 10px 2% 10px 10px; border:1px solid black;"> top padding will be 10px, right padding will be 2% of the total width of the document, bottom padding and top padding will be 10px
</p>
```

# CSS Tables

## Table Borders

To specify table borders in CSS, use the border property.

The example below specifies a black border for table, th, and td elements:

**Example**

```
table, th, td
{
    border: 1px solid black;
}
```

Notice that the table in the example above has double borders. This is because both the table and the th/td elements have separate borders.

To display a single border for the table, use the border-collapse property.

## Collapse Borders

The border-collapse property sets whether the table borders are collapsed into a single border or separated:

**Example**

```
table
{
    border-collapse:collapse;
}
table,th, td
{
    border: 1px solid black;
}
```

## Table Width and Height

Width and height of a table is defined by the width and height properties.

The example below sets the width of the table to 100%, and the height of the th elements to 50px:

**Example**

```
table
{
   width:100%;
}
th
{
   height:50px;
}
```

## Table Text Alignment

The text in a table is aligned with the text-align and vertical-align properties.

The text-align property sets the horizontal alignment, like left, right, or center:

**Example**

```
td
{
```

```
    text-align:right;
}
```

The vertical-align property sets the vertical alignment, like top, bottom, or middle:

**Example**

```
td
{
height:50px;
vertical-align:bottom;
}
```

## Table Padding

To control the space between the border and content in a table, use the padding property on td and th elements:

**Example**

```
td
{
    padding:15px;
}
```

## Table Color

The example below specifies the color of the borders, and the text and background color of th elements:

**Example**

```
table, td, th
{
    border:1px solid green;
}
th
{
    background-color:green;
    color:white;
}
```

**Example**

```
<style type="text/css">
    tr:nth-child(odd)   {
            background-color:red;
    }
    tr:nth-child(even)   {
            background-color:green;
    }
    table,tr,td,th{
```

```
            border:1px solid black;
            border-collapse:collapse;
        }
</style>
```

**Example**

```
<html>
    <head>
        <style type="text/css">
            #customers
            {
                font-family:"Trebuchet MS", Arial, Helvetica, sans-serif;
                width:100%;
                border-collapse:collapse;
            }
            #customers td, #customers th
            {
                font-size:1em;
                border:1px solid #98bf21;
                padding:3px 7px 2px 7px;
            }
            #customers th
            {
                font-size:1.1em;
                text-align:left;
                padding-top:5px;
                padding-bottom:4px;
                background-color:#A7C942;
                color:#ffffff;
            }
            #customers tr.alt td
            {
                color:#000000;
                background-color:#EAF2D3;
            }
        </style>
    </head>
    <body>
        <table id="customers">
            <tr>
              <th>Company</th>
              <th>Contact</th>
              <th>Country</th>
            </tr>
            <tr>
```

```
                    <td>Alfreds Futterkiste</td>
                    <td>Maria Anders</td>
                    <td>Germany</td>
                </tr>
                <tr class="alt">
                    <td>Berglunds snabbköp</td>
                    <td>Christina Berglund</td>
                    <td>Sweden</td>
                </tr>
                <tr>
                    <td>Centro comercial Moctezuma</td>
                    <td>Francisco Chang</td>
                    <td>Mexico</td>
                </tr>
                <tr class="alt">
                    <td>Ernst Handel</td>
                    <td>Roland Mendel</td>
                    <td>Austria</td>
                </tr>
                <tr>
                    <td>Island Trading</td>
                    <td>Helen Bennett</td>
                    <td>UK</td>
                </tr>
                <tr class="alt">
                    <td>Königlich Essen</td>
                    <td>Philip Cramer</td>
                    <td>Germany</td>
                </tr>
            </table>
        </body>
</html>
```

## CSS - Cursors

The *cursor* property of CSS allows you to specify the type of cursor that should be displayed to the user.

One good usage of this property is in using images for submit buttons on forms. By default, when a cursor hovers over a link, the cursor changed from a pointer to a hand. For a submit button on a form this does not happen. Therefore, using the cursor property to change the cursor to a hand whenever someone hovers over an image that is a submit button. This provides a visual clue that they can click it.

The table that follows shows possible values for the cursor property:

| Value | Description |
|---|---|

| | |
|---|---|
| Auto | Shape of the cursor depends on the context area it is over. For example an I over text, a hand over a link, and so on... |
| Crosshair | A crosshair or plus sign |
| Default | An arrow |
| Pointer | A pointing hand (in IE 4 this value is hand) |
| Move | The I bar |
| e-resize | The cursor indicates that an edge of a box is to be moved right (east) |
| ne-resize | The cursor indicates that an edge of a box is to be moved up and right (north/east) |
| nw-resize | The cursor indicates that an edge of a box is to be moved up and left (north/west) |
| n-resize | The cursor indicates that an edge of a box is to be moved up (north) |
| se-resize | The cursor indicates that an edge of a box is to be moved down and right (south/east) |
| sw-resize | The cursor indicates that an edge of a box is to be moved down and left (south/west) |
| s-resize | The cursor indicates that an edge of a box is to be moved down (south) |
| w-resize | The cursor indicates that an edge of a box is to be moved left (west) |
| Text | The I bar |
| Wait | An hour glass |
| Help | A question mark or balloon, ideal for use over help buttons |
| <url> | The source of a cursor image file |

**NOTE:** You should try to use only these values to add helpful information for users, and in places they would expect to see that cursor. For example, using the crosshair when someone hovers over a link can confuse visitors.

Here is the example:

```
<p>Move the mouse over the words to see the cursor change:</p>
<div style="cursor:auto">Auto</div>
<div style="cursor:crosshair">Crosshair</div>
<div style="cursor:default">Default</div>
<div style="cursor:pointer">Pointer</div>
<div style="cursor:move">Move</div>
<div style="cursor:e-resize">e-resize</div>
<div style="cursor:ne-resize">ne-resize</div>
<div style="cursor:nw-resize">nw-resize</div>
<div style="cursor:n-resize">n-resize</div>
<div style="cursor:se-resize">se-resize</div>
<div style="cursor:sw-resize">sw-resize</div>
<div style="cursor:s-resize">s-resize</div>
<div style="cursor:w-resize">w-resize</div>
<div style="cursor:text">text</div>
<div style="cursor:wait">wait</div>
<div style="cursor:help">help</div>
```

## CSS - Scrollbars

There may be a case when an element's content might be larger than the amount of space allocated to it. For example given width and height properties did not allow enough room to accommodate the content of the element.

CSS provides a property called *overflow* which tells the browser what to do if the box's contents is larger than the box itself. This property can take one of the following values:

| Value | Description |
|---|---|
| visible | Allows the content to overflow the borders of its containing element. |
| hidden | The content of the nested element is simply cut off at the border of the containing element and no scrollbars is visible. |
| scroll | The size of the containing element does not change, but the scrollbars are added to allow the user to scroll to see the content. |
| auto | The purpose is the same as scroll, but the scrollbar will be shown only if the content does overflow. |

Here is the example:

```
<style type="text/css">
.scroll{
        display:block;
        border: 1px solid red;
        padding:5px;
        margin-top:5px;
        width:300px;
        height:50px;
        overflow:scroll;
}
.auto{
        display:block;
        border: 1px solid red;
        padding:5px;
        margin-top:5px;
        width:300px;
        height:50px;
        overflow:auto; /* to get horizantal scroll in mozilla : overflow: -moz-scrollbars-horizontal */

}
</style>
  <p>Example of scroll value:</p>
    <div class="scroll">
       I am going to keep lot of content here just to show
       you how scrollbars works if there is an overflow in
       an element box. This provides your horizontal as well
       as vertical scrollbars.
     </div>
    <br />

    <p>Example of auto value:</p>
     <div class="auto">
      I am going to keep lot of content here just to show
      you how scrollbars works if there is an overflow in
      an element box. This provides your horizontal as well
```

```
    as vertical scrollbars.
  </div>
```

## CSS - Images

Images are very important part of any Web Page. Though it is not recommended to include lot of images but it is still important to use good images wherever it is required.

CSS plays a good role to control image display. You can set following image properties using CSS.

- The **border** property is used to set the width of an image border.
- The **height** property is used to set the height of an image.
- The **width** property is used to set the width of an image.
- The **-moz-opacity** property is used to set the opacity of an image.

## The image border Property:

The *border* property of an image is used to set the width of an image border. This property can have a value in length or in %.

A width of zero pixels means no border.

Here is the example:

```
<img style="border:0px;" src="/images/css.gif" />
<br />
<img style="border:3px dashed red;" src="/images/css.gif" />
```

## The image height Property:

The *height* property of an image is used to set the height of an image. This property can have a value in length or in %. While giving value in %, it applies it in respect of the box in which an image is available.

Here is the example:

```
<img style="border:1px solid red; height:100px;"
        src="/images/css.gif" />
<br />
<img style="border:1px solid red; height:50%;"
        src="/images/css.gif" />
```

## The image width Property:

The *width* property of an image is used to set the width of an image. This property can have a value in length or in %. While giving value in %, it applies it in respect of the box in which an image is available.

Here is the example:

```
<img style="border:1px solid red; width:100px;"
        src="/images/css.gif" />
<br />
<img style="border:1px solid red; width:100%;"
        src="/images/css.gif" />
```

## The -moz-opacity Property:

The ***-moz-opacity*** property of an image is used to set the opacity of an image. This property is used to create a transparent image in Mozilla. IE uses **filter:alpha(opacity=x)** to create transparent images.

In Mozilla (-moz-opacity:x) x can be a value from 0.0 - 1.0. A lower value makes the element more transparent (The same things goes for the CSS3-valid syntax opacity:x).

In IE (filter:alpha(opacity=x)) x can be a value from 0 - 100. A lower value makes the element more transparent.

Here is the example:

```
<img style="border:1px solid red;-moz-opacity:0.4;filter:alpha(opacity=40);" src="/images/css.gif" />
```

## Example 1 - Creating a Transparent Image

The CSS3 property for transparency is **opacity**.

First we will show you how to create a transparent image with CSS.

```
img
{
opacity:0.4;
filter:alpha(opacity=40); /* For IE8 and earlier */
}
```

IE9, Firefox, Chrome, Opera, and Safari use the property **opacity** for transparency. The opacity property can take a value from 0.0 - 1.0. A lower value makes the element more transparent.

IE8 and earlier use **filter:alpha(opacity=x)**. The x can take a value from 0 - 100. A lower value makes the element more transparent.

### Example 2 - Image Transparency - Hover Effect

The CSS looks like this:

```
img
{
   opacity:0.4;
   filter:alpha(opacity=40); /* For IE8 and earlier */
}
img:hover
{
   opacity:1.0;
   filter:alpha(opacity=100); /* For IE8 and earlier */
}
```

The first CSS block is similar to the code in Example 1. In addition, we have added what should happen when a user hover over one of the images. In this case we want the image to NOT be transparent when the user hover over it.

The CSS for this is: **opacity=1**.

IE8 and earlier: **filter:alpha(opacity=100)**.

When the mouse pointer moves away from the image, the image will be transparent again.

### Example 3 - Text in Transparent Box

```
<html>
<head>
<style type="text/css">
        div.background
        {
                width:500px;
                height:250px;
                background:url(klematis.jpg) repeat;
                border:2px solid black;
        }
        div.transbox
        {
                width:400px;
                height:180px;
                margin:30px 50px;
                background-color:#ffffff;
                border:1px solid black;
                opacity:0.6;
                filter:alpha(opacity=60); /* For IE8 and earlier */
```

```
        }
        div.transbox p
        {
          margin:30px 40px;
          font-weight:bold;
          color:#000000;
        }
</style>
</head>

<body>

        <div class="background">
                <div class="transbox">
                        <p>This is some text that is placed in the transparent box.
                        This is some text that is placed in the transparent box.
                        This is some text that is placed in the transparent box.
                        This is some text that is placed in the transparent box.
                        This is some text that is placed in the transparent box.
                        </p>
                </div>
        </div>

</body>
</html>
```

First, we create a div element (class="background") with a fixed height and width, a background image, and a border. Then we create a smaller div (class="transbox") inside the first div element. The "transbox" div have a fixed width, a background color, and a border - and it is transparent. Inside the transparent div, we add some text inside a p element.

## CSS Display and Visibility

The display property specifies if/how an element is displayed, and the visibility property specifies if an element should be visible or hidden.

### Hiding an Element - display:none or visibility:hidden

Hiding an element can be done by setting the **display** property to "**none**" or the **visibility** property to "**hidden**". However, notice that these two methods produce different results:

**visibility:hidden** hides an element, but it will still take up the same space as before. The element will be hidden, but still affect the layout.

### Example

```
h1.hidden {visibility:hidden;}
```

```
<html>
```

```
<head>
  <style type="text/css">
    h1.hidden {visibility:hidden;}
  </style>
</head>

<body>
  <h1>This is a visible heading</h1>
  <h1 class="hidden">This is a hidden heading</h1>
  <p>Notice that the hidden heading still takes up space.</p>
</body>
</html>
```

display:none hides an element, and it will not take up any space. The element will be hidden, and the page will be displayed as the element is not there:

## Example

h1.hidden {display:none;}

```
<html>
<head>
<style type="text/css">
        h1.hidden {display:none;}
</style>
</head>

<body>
        <h1>This is a visible heading</h1>
        <h1 class="hidden">This is a hidden heading</h1>
        <p>Notice that the hidden heading does not take up space.</p>
</body>

</html>
```

CSS Display - Block and Inline Elements

A block element is an element that takes up the full width available, and has a line break before and after it.

Examples of block elements:

- <h1>
- <p>
- 

An inline element only takes up as much width as necessary, and does not force line breaks.

Examples of inline elements:

- 
- <a>

## Changing How an Element is Displayed

Changing an inline element to a block element, or vice versa, can be useful for making the page look a specific way, and still follow web standards.

The following example displays list items as inline elements:

### Example

```
li {display:inline;}
```

```
<html>
<head>
<style type="text/css">
        li{display:inline;}
</style>
</head>
<body>

        <p>Display this link list as a horizontal menu:</p>

        <ul>
                <li><a href="/html/default.asp" target="_blank">HTML</a></li>
                <li><a href="/css/default.asp" target="_blank">CSS</a></li>
                <li><a href="/js/default.asp" target="_blank">JavaScript</a></li>
                <li><a href="/xml/default.asp" target="_blank">XML</a></li>
        </ul>

</body>
</html>
```

The following example displays span elements as block elements:

### Example

```
span {display:block;}
```

```
<html>
<head>
<style type="text/css">
        span
        {
                display:block;
        }
```

```
</style>
</head>
<body>

        <h2>Nirvana</h2>
                <span>Record: MTV Unplugged in New
York</span>
                <span>Year: 1993</span>

        <h2>Radiohead</h2>
                <span>Record: OK Computer</span>
                <span>Year: 1997</span>

</body>
</html>
```

**Note:** Changing the display type of an element changes only how the element is displayed, NOT what kind of element it is. For example: An inline element set to display:block is not allowed to have a block element nested inside of it.

## CSS Positioning

### Positioning

The CSS positioning properties allow you to position an element. It can also place an element behind another, and specify what should happen when an element's content is too big.

Elements can be positioned using the **top, bottom, left,** and **right** properties. However, these properties will not work unless the **position** property is set first. They also work differently depending on the positioning method.

There are four different positioning methods.

### Static Positioning

HTML elements are positioned static by default. A static positioned element is always positioned according to the normal flow of the page.

Static positioned elements are not affected by the top, bottom, left, and right properties.

### Fixed Positioning

An element with fixed position is positioned relative to the browser window.

It will not move even if the window is scrolled:

**Example**

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
<style type="text/css">
        img.pos_fixed
        {
                position:fixed;
                top:30px;
                right:5px;
        }
</style>
</head>
<body>

<img src="images/facebook.jpg" class="pos_fixed" style="height:50px;width:50px;" />

<p><b>Note:</b> IE7 and IE8 supports the fixed value only if a
!DOCTYPE is specified.</p>

<p>Some text</p><p>Some text</p><p>Some text</p><p>Some text</p><p>Some
text</p><p>Some text</p><p>Some text</p><p>Some text</p><p>Some text</p><p>Some
text</p><p>Some text</p><p>Some text</p><p>Some text</p><p>Some text</p><p>Some
text</p><p>Some text</p>
<p>Some text</p><p>Some text</p><p>Some text</p><p>Some text</p><p>Some
text</p><p>Some text</p><p>Some text</p><p>Some text</p><p>Some text</p><p>Some
text</p><p>Some text</p><p>Some text</p><p>Some text</p><p>Some text</p><p>Some
text</p><p>Some text</p>
</body>
</html>
```

**Note:** IE7 and IE8 support the fixed value only if a !DOCTYPE is specified.

Fixed positioned elements are removed from the normal flow. The document and other elements behave like the fixed positioned element does not exist.

Fixed positioned elements can overlap other elements.

## Relative Positioning

A relative positioned element is positioned relative to its normal position.

## Example

```
<html>
<head>
<style type="text/css">
        h2.pos_left
        {
```

```
                position:relative;
                left:-20px;
        }

        h2.pos_right
        {
                position:relative;
                left:20px;
        }
</style>
</head>

<body>
<h2>This is a heading with no position</h2>
<h2 class="pos_left">This heading is moved left according to its normal position</h2>
<h2 class="pos_right">This heading is moved right according to its normal position</h2>
<p>Relative positioning moves an element RELATIVE to its original position.</p>
<p>The style "left:-20px" subtracts 20 pixels from the element's original left position.</p>
<p>The style "left:20px" adds 20 pixels to the element's original left position.</p>
</body>

</html>
```

The content of relatively positioned elements can be moved and overlap other elements, but the reserved space for the element is still preserved in the normal flow.

## Example

```
<html>
<head>
<style type="text/css">
        h2.pos_top
        {
                position:relative;
                top:-50px;
        }
</style>
</head>

<body>
<h2>This is a heading with no position</h2>
<h2 class="pos_top">This heading is moved upwards according to its normal position</h2>
<p><b>Note:</b> Even if the content of the relatively positioned element is moved, the reserved space for the
element is still preserved in the normal flow.</p>
</body>

</html>
```

Relatively positioned elements are often used as container blocks for absolutely positioned elements.

## Absolute Positioning

An absolute position element is positioned relative to the first parent element that has a position other than static. If no such element is found, the containing block is <html>:

## Example

```
<html>
<head>
<style type="text/css">
        h2
        {
                position:absolute;
                left:100px;
                top:150px;
        }
</style>
</head>

<body>
<h2>This is a heading with an absolute position</h2>
<p>With absolute positioning, an element can be placed anywhere on a page. The heading below is placed
100px from the left of the page and 150px from the top of the page.</p>
</body>

</html>
```

Absolutely positioned elements are removed from the normal flow. The document and other elements behave like the absolutely positioned element does not exist.

Absolutely positioned elements can overlap other elements.

## CSS - Layers

CSS gives you opportunity to create layers of various divisions. The CSS layers refer to applying the *z-index* property to elements that overlap with each other.

The z-index property is used alongwith *position* property to create an effect of layers. You can specify which element should come on top and which element should come at bottom.

A z-index property can help you to create more complex webpage layouts. Following is the example which shows how to create layers in CSS.

```
<div style="background-color:red;
        width:300px;
```

```
        height:100px;
        position:relative;
        top:10px;
        left:80px;
        z-index:2">
</div>
<div style="background-color:yellow;
        width:300px;
        height:100px;
        position:relative;
        top:-60px;
        left:35px;
        z-index:1;">
</div>
<div style="background-color:green;
width:300px;
        height:100px;
        position:relative;
        top:-220px;
        left:120px;
        z-index:3;">
</div>
```

## CSS Float

### What is CSS Float?

With CSS float, an element can be pushed to the left or right, allowing other elements to wrap around it.

Float is very often used for images, but it is also useful when working with layouts.

### How Elements Float

Elements are floated horizontally, this means that an element can only be floated left or right, not up or down.

A floated element will move as far to the left or right as it can. Usually this means all the way to the left or right of the containing element.

The elements after the floating element will flow around it.

The elements before the floating element will not be affected.

If an image is floated to the right, a following text flows around it, to the left:

### Example

```
<html>
<head>
<style type="text/css">
img
{
```

```
float:right;
}
</style>
</head>

<body>
<p>In the paragraph below, we have added an image with style <b>float:right</b>. The result is that the image
will float to the right in the paragraph.</p>
<p>
<img src="logocss.gif" width="95" height="84" />
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
</p>
</body>

</html>
```

## Floating Elements Next to Each Other

If you place several floating elements after each other, they will float next to each other if there is room.

Here we have made an image gallery using the float property:

## Example

```
<html>
<head>
<style type="text/css">
        .thumbnail
        {
                float:left;
                width:110px;
                height:90px;
                margin:5px;
        }
</style>
</head>

<body>
<h3>Image Gallery</h3>
<p>Try resizing the window to see what happens when the images does not have enough room.</p>
```

```
        <img class="thumbnail" src="images/home.jpg" width="107" height="90">
        <img class="thumbnail" src="images/facebook.jpg" width="107" height="80">
        <img class="thumbnail" src="images/chamanti.jpg" width="116" height="90">
        <img class="thumbnail" src="images/tree.png" width="120" height="90">
        <img class="thumbnail" src="images/Desert.jpg" width="107" height="90">
        <img class="thumbnail" src="images/Koala.jpg" width="107" height="80">
        <img class="thumbnail" src="images/home2.jpg" width="116" height="90">
        <img class="thumbnail" src="images/home3.jpg" width="120" height="90">
</body>
</html>
```

Turning off Float - Using Clear

Elements after the floating element will flow around it. To avoid this, use the **clear** property.

The **clear** property specifies which sides of an element other floating elements are not allowed.

Add a text line into the image gallery, using the clear property:

## Example

```
<html>
<head>
<style type="text/css">
        .thumbnail
        {
                float:left;
                width:110px;
                height:90px;
                margin:5px;
        }
        .text_line
        {
                clear:both;
                margin-bottom:2px;
        }
</style>
</head>

<body>
<h3>Image Gallery</h3>
<p>Try resizing the window to see what happens when the images does not have enough room.</p>
        <img class="thumbnail" src="klematis_small.jpg" width="107" height="90">
        <img class="thumbnail" src="klematis2_small.jpg" width="107" height="80">
        <img class="thumbnail" src="klematis3_small.jpg" width="116" height="90">
        <img class="thumbnail" src="klematis4_small.jpg" width="120" height="90">
        <h3 class="text_line">Second row</h3>
        <img class="thumbnail" src="klematis_small.jpg" width="107" height="90">
        <img class="thumbnail" src="klematis2_small.jpg" width="107" height="80">
```

```
        <img class="thumbnail" src="klematis3_small.jpg" width="116" height="90">
        <img class="thumbnail" src="klematis4_small.jpg" width="120" height="90">
</body>
</html>
```

## CSS Pseudo-classes

CSS pseudo-classes are used to add special effects to some selectors.

### Syntax

The syntax of pseudo-classes:

**selector:pseudo-class {property:value;}**

CSS classes can also be used with pseudo-classes:

**selector.class:pseudo-class {property:value;}**

### Anchor Pseudo-classes

Links can be displayed in different ways in a CSS-supporting browser:

### Example

```
a:link {color:#FF0000;}     /* unvisited link */
a:visited {color:#00FF00;} /* visited link */
a:hover {color:#FF00FF;}  /* mouse over link */
a:active {color:#0000FF;}  /* selected link */
```

**Note**: a:hover MUST come after a:link and a:visited in the CSS definition in order to be effective!!

**Note:** a:active MUST come after a:hover in the CSS definition in order to be effective!!

**Note:** Pseudo-class names are not case-sensitive.

### Pseudo-classes and CSS Classes

Pseudo-classes can be combined with CSS classes:

**a.red:visited {color:#FF0000;}**

**<a class="red" href="css_syntax.asp">CSS Syntax</a>**

If the link in the example above has been visited, it will be displayed in red.

## CSS - The :first-child Pseudo-class

The **:first-child** pseudo-class matches a specified element that is the first child of another element.

**Note:** For :first-child to work in IE8 and earlier, a <!DOCTYPE> must be declared.

### Match the first <p> element

In the following example, the selector matches any <p> element that is the first child of any element:

### Example

```
<html>
<head>
<style type="text/css">
        p:first-child
        {
                color:blue;
        }
</style>
</head>

<body>
        <p>I am a strong man.</p>
        <p>I am a strong man.</p>
</body>
</html>
```

### Match the first <i> element in all <p> elements

In the following example, the selector matches the first <i> element in all <p> elements:

### Example

```
<html>
<head>
<style type="text/css">
        p > i:first-child
        {
                color:red;
        }
</style>
</head>

<body>
        <p>I am a <i>strong</i> man. I am a <i>strong</i>
man.</p>
        <p>I am a <i>strong</i> man. I am a <i>strong</i>
man.</p>
</body>
```

```
</html>
```

### Match all <i> elements in all first child <p> elements

In the following example, the selector matches all <i> elements in <p> elements that are the first child of another element:

### Example

```
<html>
<head>
<style type="text/css">
        p:first-child i
        {
                color:red;
        }
</style>
</head>

<body>
        <p>I am a <i>strong</i> man. I am a
<i>strong</i> man.</p>
        <p>I am a <i>strong</i> man. I am a
<i>strong</i> man.</p>
</body>
</html>
```

## CSS Pseudo-elements

CSS pseudo-elements are used to add special effects to some selectors.

### Syntax

The syntax of pseudo-elements:

**selector:pseudo-element {property:value;}**

CSS classes can also be used with pseudo-elements:

**selector.class:pseudo-element {property:value;}**

### The :first-line Pseudo-element

The "first-line" pseudo-element is used to add a special style to the first line of a text.

In the following example the browser formats the first line of text in a p element according to the style in the "first-line" pseudo-element (where the browser breaks the line, depends on the size of the browser window):

## Example

```
p:first-line
{
   color:#ff0000;
}
```

**Note:** The "first-line" pseudo-element can only be used with block-level elements.

**Note:** The following properties apply to the "first-line" pseudo-element:

- font properties
- color properties
- background properties
- word-spacing
- letter-spacing
- text-decoration
- vertical-align
- text-transform
- line-height
- clear

## The :first-letter Pseudo-element

The "first-letter" pseudo-element is used to add a special style to the first letter of a text:

## Example

```
p:first-letter
{
   color:#ff0000;
   font-size:xx-large;
}
```

**Note:** The "first-letter" pseudo-element can only be used with block-level elements.

**Note:** The following properties apply to the "first-letter" pseudo- element:

- font properties
- color properties
- background properties
- margin properties
- padding properties
- border properties
- text-decoration
- vertical-align (only if "float" is "none")

- text-transform
- line-height
- float
- clear

## Pseudo-elements and CSS Classes

Pseudo-elements can be combined with CSS classes:

```
p.article:first-letter {color:#ff0000;}

<p class="article">A paragraph in an article</p>
```

The example above will display the first letter of all paragraphs with class="article", in red.

## Multiple Pseudo-elements

Several pseudo-elements can also be combined.

In the following example, the first letter of a paragraph will be red, in an xx-large font size. The rest of the first line will be blue, and in small-caps. The rest of the paragraph will be the default font size and color:

### Example

```
p:first-letter
{
    color:#ff0000;
    font-size:xx-large;
}
p:first-line
{
    color:#0000ff;
    font-variant:small-caps;
}
```

## CSS - The :before Pseudo-element

The ":before" pseudo-element can be used to insert some content before the content of an element.

The following example inserts an image before each <h1> element:

### Example

```
<html>
<head>
<style type="text/css">
        h1:before
        {
                content:url(images/home.jpg);
```

```
        }
</style>
</head>
<body>
        <h1>welcome to my home</h1>
</body>
</html>
```

## CSS - The :after Pseudo-element

The ":after" pseudo-element can be used to insert some content after the content of an element.

The following example inserts an image after each <h1> element:

## Example

```
h1:after
{
content:url(images/home.jpg);
}
```

## All CSS Pseudo Classes/Elements

| Selector | Example | Example description |
|---|---|---|
| :link | a:link | Selects all unvisited links |
| :visited | a:visited | Selects all visited links |
| :active | a:active | Selects the active link |
| :hover | a:hover | Selects links on mouse over |
| :focus | input:focus | Selects the input element which has focus |
| :first-letter | p:first-letter | Selects the first letter of every <p> element |
| :first-line | p:first-line | Selects the first line of every <p> element |
| :first-child | p:first-child | Selects every <p> elements that is the first child of its parent |
| :before | p:before | Insert content before every <p> element |
| :after | p:after | Insert content after every <p> element |
| :lang(*language*) | p:lang(it) | Selects every <p> element with a lang attribute value starting with "it" |

## CSS Attribute Selectors

### Style HTML Elements With Specific Attributes

It is possible to style HTML elements that have specific attributes, not just class and id.

**Note:** IE7 and IE8 support attribute selectors only if a !DOCTYPE is specified. Attribute selection is **NOT** supported in IE6 and lower.

### Attribute Selector

The example below styles all elements with a title attribute:

### Example

```
[title]
{
    color:blue;
}
```

### Attribute and Value Selector

The example below styles all elements with title="WebApplication":

### Example

```
[title= WebApplication]
{
      border:5px solid green;
}
```

### Attribute and Value Selector - Multiple Values

The example below styles all elements with a title attribute that contains a specified value. This works even if the attribute has space separated values:

### Example

```
[title~=hello]
{
   color:blue;
}
```

The example below styles all elements with a lang attribute that contains a specified value. This works even if the attribute has hyphen ( - ) separated values:

## Example

```
[lang|=en]
{
    color:blue;
}
```

## Styling Forms

The attribute selectors are particularly useful for styling forms without class or ID:

## Example

```
input[type="text"]
{
    width:150px;
    display:block;
    margin-bottom:10px;
    background-color:yellow;
}
input[type="button"]
{
    width:120px;
    margin-left:35px;
    display:block;
}
```