Module-05

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>
<p>This is a paragraph.</p>
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!

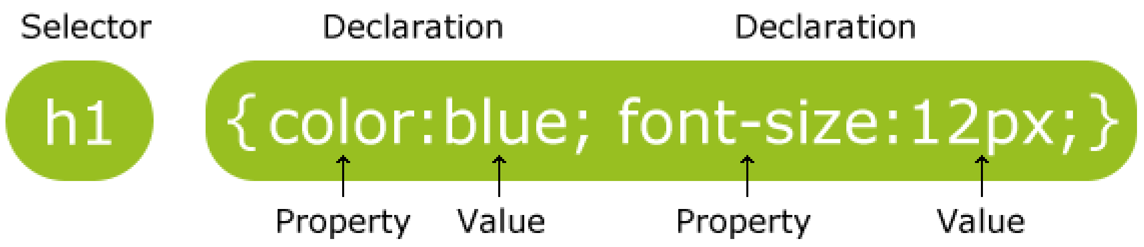
CSS Syntax

Examples

- Look at Example 1
- Look at Example 2

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.

CSS Example

A CSS declaration always ends with a semicolon, and declaration groups are surrounded by curly brackets:

```
p {color:red;text-align:center;}
```

To make the CSS more readable, you can put one declaration on each line, like this:

Example

```
p
{
	color:red;
	text-align:center;
}
```

CSS Comments

Comments are used to explain your code, and may help you when you edit the source code at a later date. Comments are ignored by browsers.
A CSS comment begins with "/*", and ends with "*/", like this:

```
/*This is a comment*/
p
{
	text-align:center;
	/*This is another comment*/
	color:black;
	font-family:arial;
}
```

CSS Id and Class

The id and class Selectors

In addition to setting a style for a HTML element, CSS allows you to specify your own selectors called "id" and "class".

The id Selector

The id selector is used to specify a style for a single, unique element.
The id selector uses the id attribute of the HTML element, and is defined with a "#".
The style rule below will be applied to the element with id="para1":

Example

```
#para1
{
text-align:center;
color:red;
}
```

Tip of the Day: Do **NOT** start an ID name with a number! It will not work in Mozilla/Firefox.

The class Selector

The class selector is used to specify a style for a group of elements. Unlike the id selector, the class selector is most often used on several elements.
This allows you to set a particular style for many HTML elements with the same class.
The class selector uses the HTML class attribute, and is defined with a "."
In the example below, all HTML elements with class="center" will be center-aligned:

Example

```
.center {text-align:center;}
```

You can also specify that only specific HTML elements should be affected by a class.
In the example below, all p elements with class="center" will be center-aligned:

Example

```
p.center {text-align:center;}
```

Tip of the Day: Do **NOT** start a class name with a number! This is only supported in Internet Explorer.

CSS How To...

When a browser reads a style sheet, it will format the document according to it.

Three Ways to Insert CSS

There are three ways of inserting a style sheet:

- External style sheet
- Internal style sheet
- Inline style

External Style Sheet

An external style sheet is ideal when the style is applied to many pages. With an external style sheet, you can change the look of an entire Web site by changing one file. Each page must link to the style sheet using the <link> tag. The <link> tag goes inside the head section:

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

An external style sheet can be written in any text editor. The file should not contain any html tags. Your style sheet should be saved with a .css extension. An example of a style sheet file is shown below:

```
hr {color:sienna;}
p {margin-left:20px;}
body {background-image:url("images/back40.gif");}
```

Do not leave spaces between the property value and the unit (such as margin-left:20 px). Correct way: margin-left:20px

Internal Style Sheet

An internal style sheet should be used when a single document has a unique style. You define internal styles in the head section of an HTML page, by using the <style> tag, like this:

```
<head>
        <style type="text/css">
                hr {color:sienna;}
                p {margin-left:20px;}
                body {background-image:url("images/back40.gif");}
        </style>
</head>
```

Inline Styles

An inline style loses many of the advantages of style sheets by mixing content with presentation. Use this method sparingly!
To use inline styles you use the style attribute in the relevant tag. The style attribute can contain any CSS property. The example shows how to change the color and the left margin of a paragraph:

```
<p style="color:sienna;margin-left:20px">This is a paragraph.</p>
```

Multiple Style Sheets

If some properties have been set for the same selector in different style sheets, the values will be inherited from the more specific style sheet.
For example, an external style sheet has these properties for the h3 selector:

```
h3
{
	color:red;
	text-align:left;
	font-size:8pt;
}
```

And an internal style sheet has these properties for the h3 selector:

```
h3
{
	text-align:right;
	font-size:20pt;
}
```

If the page with the internal style sheet also links to the external style sheet the properties for h3 will be:

```
color:red;
text-align:right;
font-size:20pt;
```

The color is inherited from the external style sheet and the text-alignment and the font-size is replaced by the internal style sheet.

Multiple Styles Will Cascade into One

Styles can be specified:

- inside an HTML element
- inside the head section of an HTML page
- in an external CSS file

Tip: Even multiple external style sheets can be referenced inside a single HTML document.

Cascading order

What style will be used when there is more than one style specified for an HTML element?
Generally speaking we can say that all the styles will "cascade" into a new "virtual" style sheet by the following rules, where number four has the highest priority:

1. Browser default
2. External style sheet
3. Internal style sheet (in the head section)
4. Inline style (inside an HTML element)

So, an inline style (inside an HTML element) has the highest priority, which means that it will override a style defined inside the <head> tag, or in an external style sheet, or in a browser (a default value).

Note: If the link to the external style sheet is placed after the internal style sheet in HTML <head>, the external style sheet will override the internal style sheet!

CSS Background

CSS background properties are used to define the background effects of an element.

CSS properties used for background effects:

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

Background Color

The background-color property specifies the background color of an element.
The background color of a page is defined in the body selector:

Example

```
body {background-color:#b0c4de;}
```

With CSS, a color is most often specified by:

- a HEX value - like "#ff0000"
- an RGB value - like "rgb(255,0,0)"
- a color name - like "red"

Look at CSS Color Values for a complete list of possible color values.
In the example below, the h1, p, and div elements have different background colors:

Example

```
h1 {background-color:#6495ed;}
p {background-color:#e0ffff;}
div {background-color:#b0c4de;}
```

Background Image

The background-image property specifies an image to use as the background of an element.
By default, the image is repeated so it covers the entire element.
The background image for a page can be set like this:

Example

```
body {background-image:url('paper.gif');}
```

Below is an example of a bad combination of text and background image. The text is almost not readable:

Example

```
body {background-image:url('bgdesert.jpg');}
```

Background Image - Repeat Horizontally or Vertically

By default, the background-image property repeats an image both horizontally and vertically.
Some images should be repeated only horizontally or vertically, or they will look strange, like this:

Example

```
body
{
        background-image:url('gradient2.png');
}
```

If the image is repeated only horizontally (repeat-x), the background will look better:

Example

```
body
{
        background-image:url('gradient2.png');
        background-repeat:repeat-x;
}
```

Background Image - Set position and no-repeat

Tip of the Day: When using a background image, use an image that does not disturb the text.
Showing the image only once is specified by the background-repeat property:

Example

```
body
{
	background-image:url('img_tree.png');
	background-repeat:no-repeat;
}
```

In the example above, the background image is shown in the same place as the text. We want to change the position of the image, so that it does not disturb the text too much.
The position of the image is specified by the background-position property:

Example

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

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":

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

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

All CSS Background Properties

Property	Description
background	Sets all the background properties in one declaration
background-attachment	Sets whether a background image is fixed or scrolls with the rest of the page
background-color	Sets the background color of an element
background-image	Sets the background image for an element
background-position	Sets the starting position of a background image
background-repeat	Sets how a background image will be repeated

CSS Text

Text Color

The color property is used to set the color of the text.
With CSS, a color is most often specified by:

- a HEX value - like "#ff0000"
- an RGB value - like "rgb(255,0,0)"
- a color name - like "red"

Look at CSS Color Values for a complete list of possible color values.
The default color for a page is defined in the body selector.

Example

```
body {color:blue;}
h1 {color:#00ff00;}
h2 {color:rgb(255,0,0);}
```

For W3C compliant CSS: If you define the color property, you must also define the background-color property.

Text Alignment

The text-align property is used to set the horizontal alignment of a text.
Text can be centered, or aligned to the left or right, or justified.
When text-align is set to "justify", each line is stretched so that every line has equal width, and the left and right margins are straight (like in magazines and newspapers).

Example

```
h1 {text-align:center;}
p.date {text-align:right;}
p.main {text-align:justify;}
```

Text Decoration

The text-decoration property is used to set or remove decorations from text.
The text-decoration property is mostly used to remove underlines from links for design purposes:

Example

```
a {text-decoration:none;}
```

It can also be used to decorate text:

Example

```
h1 {text-decoration:overline;}
h2 {text-decoration:line-through;}
h3 {text-decoration:underline;}
h4 {text-decoration:blink;}
```

It is not recommended to underline text that is not a link, as this often confuses users.

Text Transformation

The text-transform property is used to specify uppercase and lowercase letters in a text.
It can be used to turn everything into uppercase or lowercase letters, or capitalize the first letter of each word.

Example

```
p.uppercase {text-transform:uppercase;}
p.lowercase {text-transform:lowercase;}
p.capitalize {text-transform:capitalize;}
```

Text Indentation

The text-indentation property is used to specify the indentation of the first line of a text.

Example: p {text-indent:50px;}

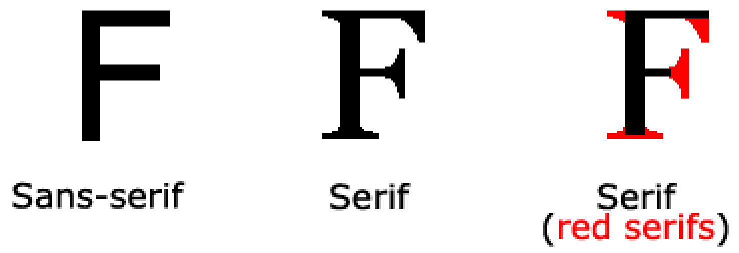
All CSS Text Properties

Property	Description
color	Sets the color of text
direction	Specifies the text direction/writing direction
letter-spacing	Increases or decreases the space between characters in a text
line-height	Sets the line height
text-align	Specifies the horizontal alignment of text
text-decoration	Specifies the decoration added to text
text-indent	Specifies the indentation of the first line in a text-block
text-shadow	Specifies the shadow effect added to text
text-transform	Controls the capitalization of text
unicode-bidi	
vertical-align	Sets the vertical alignment of an element
white-space	Specifies how white-space inside an element is handled
word-spacing	Increases or decreases the space between words in a text

CSS Font

CSS font properties define the font family, boldness, size, and the style of a text.

Difference Between Serif and Sans-serif Fonts



Tip of the day: On computer screens, sans-serif fonts are considered easier to read than serif fonts.

CSS Font Families

In CSS, there are two types of font family names:

- **generic family** - a group of font families with a similar look (like "Serif" or "Monospace")
- **font family** - a specific font family (like "Times New Roman" or "Arial")

Generic family	Font family	Description
Serif	Times New Roman Georgia	Serif fonts have small lines at the ends on some characters
Sans-serif	Arial Verdana	"Sans" means without - these fonts do not have the lines at the ends of characters

Monospace	Courier New Lucida Console	All monospace characters have the same width

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;}
```

Font Style
The font-style property is mostly used to specify italic text.
This property has three values:

- normal - The text is shown normally
- italic - The text is shown in italics
- oblique - The text is "leaning" (oblique is very similar to italic, but less supported)

Example

```
p.normal {font-style:normal;}
p.italic {font-style:italic;}
p.oblique {font-style:oblique;}
```

Font Size
The font-size property sets the size of the text.
Being able to manage the text size is important in web design. However, you should not use font size adjustments to make paragraphs look like headings, or headings look like paragraphs.
Always use the proper HTML tags, like <h1> - <h6> for headings and <p> for paragraphs.
The font-size value can be an absolute, or relative size.
Absolute size:

- Sets the text to a specified size
- Does not allow a user to change the text size in all browsers (bad for accessibility reasons)
- Absolute size is useful when the physical size of the output is known

Relative size:

- Sets the size relative to surrounding elements
- Allows a user to change the text size in browsers

Tip of the day: If you do not specify a font size, the default size for normal text, like paragraphs, is 16px (16px=1em).

Set Font Size With Pixels
Setting the text size with pixels gives you full control over the text size:

Example

```
h1 {font-size:40px;}
```

```
h2 {font-size:30px;}
p {font-size:14px;}
```

The example above allows Internet Explorer 9, Firefox, Chrome, Opera, and Safari to resize the text.

Note: The example above does not work in IE, prior version 9.

The text can be resized in all browsers using the zoom tool (however, this resizes the entire page, not just the text).

Set Font Size With Em

To avoid the resizing problem with older versions of Internet Explorer, many developers use em instead of pixels.

The em size unit is recommended by the W3C.

1em is equal to the current font size. The default text size in browsers is 16px. So, the default size of 1em is 16px.

The size can be calculated from pixels to em using this formula: *pixels*/16=*em*

Example

```
h1 {font-size:2.5em;} /* 40px/16=2.5em */
h2 {font-size:1.875em;} /* 30px/16=1.875em */
p {font-size:0.875em;} /* 14px/16=0.875em */
```

In the example above, the text size in em is the same as the previous example in pixels. However, with the em size, it is possible to adjust the text size in all browsers.

Unfortunately, there is still a problem with older versions of IE. The text becomes larger than it should when made larger, and smaller than it should when made smaller.

Use a Combination of Percent and Em

The solution that works in all browsers, is to set a default font-size in percent for the <body> element:

Example

```
body {font-size:100%;}
h1 {font-size:2.5em;}
h2 {font-size:1.875em;}
p {font-size:0.875em;}
```

Our code now works great! It shows the same text size in all browsers, and allows all browsers to zoom or resize the text!

All CSS Font Properties

Property	Description
font	Sets all the font properties in one declaration
font-family	Specifies the font family for text
font-size	Specifies the font size of text
font-style	Specifies the font style for text
font-variant	Specifies whether or not a text should be displayed in a small-caps font
font-weight	Specifies the weight of a font

CSS Links

Links can be styled in different ways.

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 */
```

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

Common Link Styles

In the example above the link changes color depending on what state it is in.
Lets go through some of the other common ways to style links:

Text Decoration

The text-decoration property is mostly used to remove underlines from links:

Example

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

Background Color

The background-color property specifies the background color for links:

Example

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

CSS Lists

The CSS list properties allow you to:

- Set different list item markers for ordered lists
- Set different list item markers for unordered lists
- Set an image as the list item marker

List

In HTML, there are two types of lists:

- unordered lists - the list items are marked with bullets
- ordered lists - the list items are marked with numbers or letters

With CSS, lists can be styled further, and images can be used as the list item marker.

Different List Item Markers

The type of list item marker is specified with the list-style-type property:

Example

```
ul.a {list-style-type: circle;}
ul.b {list-style-type: square;}
ol.c {list-style-type: upper-roman;}
ol.d {list-style-type: lower-alpha;}
```

Some of the values are for unordered lists, and some for ordered lists.

An Image as The List Item Marker

To specify an image as the list item marker, use the list-style-image property:

Example

```
ul
{
        list-style-image: url('sqpurple.gif');
}
```

The example above does not display equally in all browsers. IE and Opera will display the image-marker a little bit higher than Firefox, Chrome, and Safari.

If you want the image-marker to be placed equally in all browsers, a crossbrowser solution is explained below.

Cross browser Solution

The following example displays the image-marker equally in all browsers:

Example

```
ul
{
        list-style-type: none;
        padding: 0px;
        margin: 0px;
}

li
{
        background-image: url(sqpurple.gif);
        background-repeat: no-repeat;
        background-position: 0px 5px;
        padding-left: 14px;
}
```

Example explained:

- For ul:
    - Set the list-style-type to none to remove the list item marker
    - Set both padding and margin to 0px (for cross-browser compatibility)
- For li:
    - Set the URL of the image, and show it only once (no-repeat)
    - Position the image where you want it (left 0px and down 5px)
    - Position the text in the list with padding-left

List - Shorthand property

It is also possible to specify all the list properties in one, single property. This is called a shorthand property.

The shorthand property used for lists, is the list-style property:

Example

```
ul
```

```
{
        list-style: square url("sqpurple.gif");
}
```

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

- list-style-type
- list-style-position (for a description, see the CSS properties table below)
- list-style-image

It does not matter if one of the values above are missing, as long as the rest are in the specified order.

All CSS List Properties

Property	Description
list-style	Sets all the properties for a list in one declaration
list-style-image	Specifies an image as the list-item marker
list-style-position	Specifies if the list-item markers should appear inside or outside the content flow
list-style-type	Specifies the type of list-item marker

CSS Tables

The look of an HTML table can be greatly improved with CSS:

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

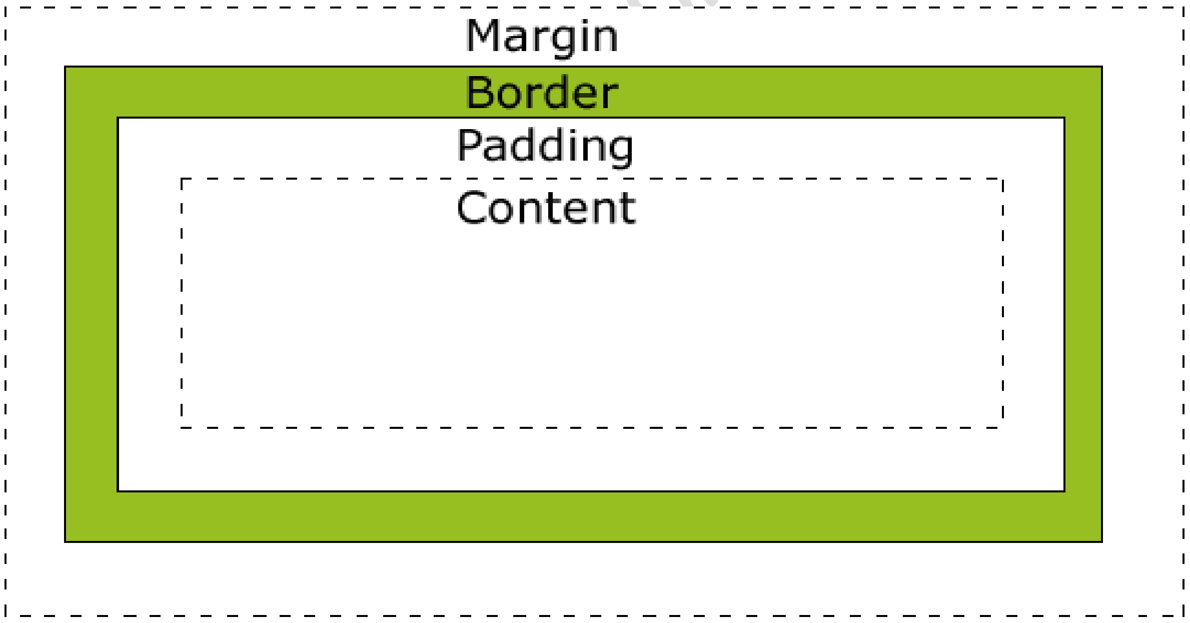
```
table, td, th
{
        border:1px solid green;
}
th
{
        background-color:green;
        color:white;
}
```

CSS Box Model

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 and 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>
</html>
```

CSS Border

Border Style

The border-style property specifies what kind of border to display.
None of the border properties will have ANY effect unless the **border-style** property is set!

border-style values:

none: Defines no border

dotted: Defines a dotted border

dashed: Defines a dashed border

solid: Defines a solid border

double: Defines two borders. The width of the two borders are the same as the border-width value

groove: Defines a 3D grooved border. The effect depends on the border-color value

ridge: Defines a 3D ridged border. The effect depends on the border-color value

inset: Defines a 3D inset border. The effect depends on the border-color value

outset: Defines a 3D outset border. The effect depends on the border-color value

Border Width

The border-width property is used to set the width of the border.
The width is set in pixels, or by using one of the three pre-defined values: thin, medium, or thick.
Note: The "border-width" property does not work if it is used alone. Use the "border-style" property to set the borders first.

Example

```
p.one
{
	border-style:solid;
	border-width:5px;
}
p.two
{
	border-style:solid;
	border-width:medium;
}
```

Border Color

The border-color property is used to set the color of the border. The color can be set by:

- name - specify a color name, like "red"
- RGB - specify a RGB value, like "rgb(255,0,0)"
- Hex - specify a hex value, like "#ff0000"

You can also set the border color to "transparent".
Note: The "border-color" property does not work if it is used alone. Use the "border-style" property to set the borders first.

Example

```
p.one
{
	border-style:solid;
	border-color:red;
}
p.two
{
	border-style:solid;
	border-color:#98bf21;
}
```

Border - Individual sides

In CSS it is possible to specify different borders for different sides:

Example

```
p
{
        border-top-style:dotted;
        border-right-style:solid;
        border-bottom-style:dotted;
        border-left-style:solid;
}
```

The example above can also be set with a single property:

Example

```
border-style:dotted solid;
```

The border-style property can have from one to four values.

- **border-style:dotted solid double dashed;**
  - top border is dotted
  - right border is solid
  - bottom border is double
  - left border is dashed
- **border-style:dotted solid double;**
  - top border is dotted
  - right and left borders are solid
  - bottom border is double
- **border-style:dotted solid;**
  - top and bottom borders are dotted
  - right and left borders are solid
- **border-style:dotted;**
  - all four borders are dotted

The border-style property is used in the example above. However, it also works with border-width and border-color.

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.

All CSS Border Properties

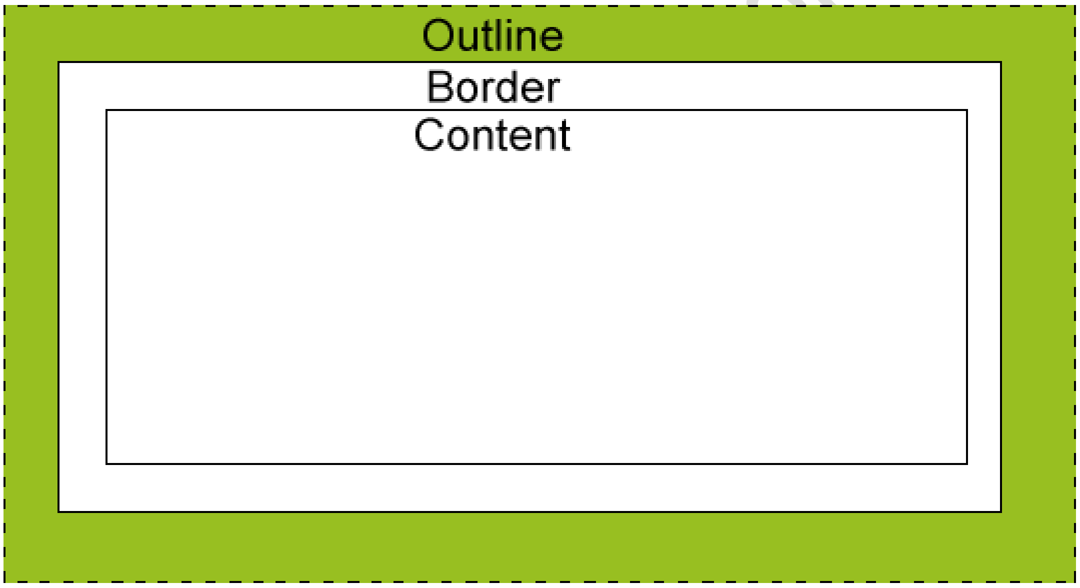
Property	Description
border	Sets all the border properties in one declaration
border-bottom	Sets all the bottom border properties in one declaration
border-bottom-color	Sets the color of the bottom border
border-bottom-style	Sets the style of the bottom border
border-bottom-width	Sets the width of the bottom border
border-color	Sets the color of the four borders
border-left	Sets all the left border properties in one declaration
border-left-color	Sets the color of the left border
border-left-style	Sets the style of the left border
border-left-width	Sets the width of the left border
border-right	Sets all the right border properties in one declaration
border-right-color	Sets the color of the right border
border-right-style	Sets the style of the right border
border-right-width	Sets the width of the right border
border-style	Sets the style of the four borders
border-top	Sets all the top border properties in one declaration
border-top-color	Sets the color of the top border
border-top-style	Sets the style of the top border
border-top-width	Sets the width of the top border
border-width	Sets the width of the four borders

CSS Outlines

An outline is a line that is drawn around elements (outside the borders) to make the element "stand out".
The outline properties specifies the style, color, and width of an outline.

CSS Outline

An outline is a line that is drawn around elements (outside the borders) to make the element "stand out".
However, the outline property is different from the border property.
The outline is not a part of an element's dimensions; the element's total width and height is not affected by the width of the outline.



All CSS Outline Properties

The number in the "CSS" column indicates in which CSS version the property is defined (CSS1 or CSS2).

Property	Description	Values	CSS
outline	Sets all the outline properties in one declaration	*outline-color* *outline-style* *outline-width* inherit	2
outline-color	Sets the color of an outline	*color_name* *hex_number* *rgb_number* invert inherit	2
outline-style	Sets the style of an outline	none dotted dashed solid double groove ridge inset outset inherit	2
outline-width	Sets the width of an outline	thin medium thick *length* inherit	2

CSS Margin

The CSS margin properties define the space around elements.

Margin

The margin clears an area around an element (outside the border). The margin does not have a background color, and is completely transparent.
The top, right, bottom, and left margin can be changed independently using separate properties. A shorthand margin property can also be used, to change all margins at once.

Possible Values

Value	Description
auto	The browser calculates a margin
length	Specifies a margin in px, pt, cm, etc. Default value is 0px
%	Specifies a margin in percent of the width of the containing element
inherit	Specifies that the margin should be inherited from the parent element

It is possible to use negative values, to overlap content.

Margin - Individual sides

In CSS, it is possible to specify different margins for different sides:

Example

```
margin-top:100px;
margin-bottom:100px;
margin-right:50px;
margin-left:50px;
```

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

All CSS Margin Properties

Property	Description
margin	A shorthand property for setting the margin properties in one declaration
margin-bottom	Sets the bottom margin of an element
margin-left	Sets the left margin of an element
margin-right	Sets the right margin of an element
margin-top	Sets the top margin of an element

CSS Padding

The CSS padding properties define the space between the element border and the element content.

Padding

The padding clears an area around the content (inside the border) of an element. The padding is affected by the background color of the element.
The top, right, bottom, and left padding can be changed independently using separate properties. A shorthand padding property can also be used, to change all paddings at once.

Possible Values

Value	Description
length	Defines a fixed padding (in pixels, pt, em, etc.)
%	Defines a padding in % of the containing element

Padding - Individual sides

In CSS, it is possible to specify different padding for different sides:

Example

```
padding-top:25px;
padding-bottom:25px;
padding-right:50px;
padding-left:50px;
```

Padding - Shorthand property

To shorten the code, it is possible to specify all the padding properties in one property. This is called a shorthand property.
The shorthand property for all the padding properties is "padding":

Example

```
padding:25px 50px;
```

The padding property can have from one to four values.

- **padding:25px 50px 75px 100px;**
  - top padding is 25px
  - right padding is 50px
  - bottom padding is 75px

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

All CSS Padding Properties

Property	Description
padding	A shorthand property for setting all the padding properties in one declaration
padding-bottom	Sets the bottom padding of an element
padding-left	Sets the left padding of an element
padding-right	Sets the right padding of an element
padding-top	Sets the top padding of an element

CSS Grouping and Nesting Selectors

Grouping Selectors

In style sheets there are often elements with the same style.

```
h1
{
	color:green;
}
h2
{
	color:green;
}
p
{
	color:green;
}
```

To minimize the code, you can group selectors.
Separate each selector with a comma.
In the example below we have grouped the selectors from the code above:

Example

```
h1,h2,p
{
	color:green;
}
```

Nesting Selectors

It is possible to apply a style for a selector within a selector.
In the example below, one style is specified for all p elements, one style is specified for all elements with class="marked", and a third style is specified only for p elements within elements with class="marked":

Example

```
p
{
	color:blue;
	text-align:center;
}
.marked
{
	background-color:red;
}
.marked p
{
	color:white;
}
```

CSS Dimension

The CSS dimension properties allow you to control the height and width of an element.

All CSS Dimension Properties

The number in the "CSS" column indicates in which CSS version the property is defined (CSS1 or CSS2).

Property	Description	Values	CSS
height	Sets the height of an element	auto *length* *%* inherit	1
max-height	Sets the maximum height of an element	none *length* *%* inherit	2
max-width	Sets the maximum width of an element	none *length* *%* inherit	2
min-height	Sets the minimum height of an element	*length* *%* inherit	2
min-width	Sets the minimum width of an element	*length* *%* inherit	2
width	Sets the width of an element	auto	1

length
%
inherit

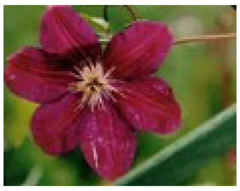
CSS Display and Visibility

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

Box 1

Box 2

Box 3



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;}

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 if the element is not there:

Example

h1.hidden {display:none;}

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>
- <div>

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

- <span>

- <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;}
```

The following example displays span elements as block elements:

Example

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

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

```
p.pos_fixed
{
        position:fixed;
        top:30px;
        right:5px;
}
```

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

```
h2.pos_left
```

```
{
	position:relative;
	left:-20px;
}
h2.pos_right
{
	position:relative;
	left:20px;
}
```

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

```
h2.pos_top
{
	position:relative;
	top:-50px;
}
```

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

```
h2
{
	position:absolute;
	left:100px;
	top:150px;
}
```

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.

Overlapping Elements

When elements are positioned outside the normal flow, they can overlap other elements.
The z-index property specifies the stack order of an element (which element should be placed in front of, or behind, the others).
An element can have a positive or negative stack order:

Example

```
img
{
	position:absolute;
	left:0px;
	top:0px;
	z-index:-1;
}
```

An element with greater stack order is always in front of an element with a lower stack order.
Note: If two positioned elements overlap, without a z-index specified, the element positioned last in the HTML code will be shown on top.

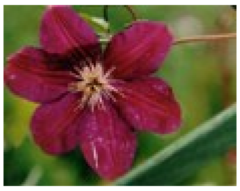
All CSS Positioning Properties

The number in the "CSS" column indicates in which CSS version the property is defined (CSS1 or CSS2).

Property	Description	Values	CSS
bottom	Sets the bottom margin edge for a positioned box	auto *length* *%* inherit	2
clip	Clips an absolutely positioned element	*shape* auto inherit	2
cursor	Specifies the type of cursor to be displayed	*url* auto crosshair default pointer move e-resize ne-resize nw-resize n-resize se-resize sw-resize s-resize w-resize text wait help	2
left	Sets the left margin edge for a positioned box	auto *length* *%* inherit	2
overflow	Specifies what happens if content overflows an element's box	auto hidden scroll visible inherit	2
position	Specifies the type of positioning for an element	absolute fixed relative static inherit	2
right	Sets the right margin edge for a positioned box	auto *length* *%* inherit	2

top	Sets the top margin edge for a positioned box	auto *length* % inherit	2
z-index	Sets the stack order of an element	*number* auto inherit	2

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

```
img
{
        float:right;
}
```

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

```
.thumbnail
{
	float:left;
	width:110px;
	height:90px;
	margin:5px;
}
```

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

```
.text_line
{
	clear:both;
}
```

All CSS Float Properties

The number in the "CSS" column indicates in which CSS version the property is defined (CSS1 or CSS2).

Property	Description	Values	CSS
clear	Specifies which sides of an element where other floating elements are not allowed	left right both none inherit	1
float	Specifies whether or not a box should float	left right none inherit	1

CSS Horizontal Align

Aligning Block 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>
- <div>

Center Aligning Using the margin Property

Block elements can be aligned by setting the left and right margins to "auto".

Note: Using margin:auto will not work in IE8 and earlier, **unless a !DOCTYPE is declared.** Setting the left and right margins to auto specifies that they should split the available margin equally. The result is a centered element:

Example

```
.center
{
	margin-left:auto;
	margin-right:auto;
	width:70%;
	background-color:#b0e0e6;
}
```

Tip: Aligning has no effect if the width is 100%.

Note: In IE5 there is a margin handling bug for block elements. To make the example above work in IE5, add some extra code. Try it yourself

Left and Right Aligning Using the position Property

One method of aligning elements is to use absolute positioning:

Example

```
.right
{
	position:absolute;
	right:0px;
	width:300px;
	background-color:#b0e0e6;
}
```

Note: Absolute positioned elements are removed from the normal flow, and can overlap elements

Crossbrowser Compatibility Issues

When aligning elements like this, it is always a good idea to predefine margin and padding for the <body> element. This is to avoid visual differences in different browsers.

There is a problem with IE8 and earlier, when using the position property. If a container element (in our case <div class="container">) has a specified width, and the !DOCTYPE declaration is missing, IE8 and earlier versions will add a 17px margin on the right side. This seems to be space reserved for a scrollbar. Always set the !DOCTYPE declaration when using the position property:

Example

```
body
{
	margin:0;
	padding:0;
}
.container
{
	position:relative;
	width:100%;
}
.right
{
        position:absolute;
        right:0px;
        width:300px;
        background-color:#b0e0e6;
}
```

Left and Right Aligning Using the float Property

One method of aligning elements is to use the float property:

Example

```
.right
{
        float:right;
        width:300px;
        background-color:#b0e0e6;
}
```

Crossbrowser Compatibility Issues

When aligning elements like this, it is always a good idea to predefine margin and padding for the <body> element. This is to avoid visual differences in different browsers.
There is a problem with IE8 and earlier when using the float property. If the !DOCTYPE declaration is missing, IE8 and earlier versions will add a 17px margin on the right side. This seems to be space reserved for a scrollbar. Always set the !DOCTYPE declaration when using the float property:

Example

```
body
{
        margin:0;
        padding:0;
}
.right
{
        float:right;
        width:300px;
        background-color:#b0e0e6;
}
```

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:

```
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:blue;
                        }
                </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:blue;
            }
        </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 - The :lang Pseudo-class

The :lang pseudo-class allows you to define special rules for different languages.
Note: IE8 supports the :lang pseudo-class only if a <!DOCTYPE> is specified.
In the example below, the :lang class defines the quotation marks for q elements with lang="no":

Example

```
<html>
    <head>
        <style type="text/css">
            q:lang(no) {quotes: "~" "~";}
        </style>
    </head>
    <body>
        <p>Some text <q lang="no">A quote in a paragraph</q> Some text.</p>
    </body>
</html>
```

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 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;
        font-variant:small-caps;
}
```

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.

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

```
h1:before
{
	content:url(smiley.gif);
}
```

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(smiley.gif);
}
```

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 Navigation Bar

Demo: Navigation Bar

- Home
- News
- Articles
- Forum
- Contact
- About

Navigation Bars

Having easy-to-use navigation is important for any web site.
With CSS you can transform boring HTML menus into good-looking navigation bars.

Navigation Bar = List of Links

A navigation bar needs standard HTML as a base.
In our examples we will build the navigation bar from a standard HTML list.
A navigation bar is basically a list of links, so using the <ul> and <li> elements makes perfect sense:

Example

```
<ul>
	<li><a href="default.asp">Home</a></li>
	<li><a href="news.asp">News</a></li>
	<li><a href="contact.asp">Contact</a></li>
	<li><a href="about.asp">About</a></li>
</ul>
```

Now let's remove the bullets and the margins and padding from the list:

Example

```
ul
{
	list-style-type:none;
	margin:0;
	padding:0;
}
```

Example explained:

- list-style-type:none - Removes the bullets. A navigation bar does not need list markers
- Setting margins and padding to 0 to remove browser default settings

The code in the example above is the standard code used in both vertical, and horizontal navigation bars.

Vertical Navigation Bar

To build a vertical navigation bar we only need to style the <a> elements, in addition to the code above:

Example

```
a
{
	display:block;
	width:60px;
}
```

Example explained:

- display:block - Displaying the links as block elements makes the whole link area clickable (not just the text), and it allows us to specify the width

- width:60px - Block elements take up the full width available by default. We want to specify a 60 px width

Tip: Also take a look at our fully styled vertical navigation bar example.
Note: Always specify the width for <a> elements in a vertical navigation bar. If you omit the width, IE6 can produce unexpected results.

Horizontal Navigation Bar

There are two ways to create a horizontal navigation bar. Using **inline** or **floating** list items. Both methods work fine, but if you want the links to be the same size, you have to use the floating method.

Inline List Items

One way to build a horizontal navigation bar is to specify the <li> elements as inline, in addition to the "standard" code above:
Example

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

Example explained:

- display:inline; - By default, <li> elements are block elements. Here, we remove the line breaks before and after each list item, to display them on one line

Tip: Also take a look at our fully styled horizontal navigation bar example.

Floating List Items

In the example above the links have different widths.
For all the links to have an equal width, float the <li> elements and specify a width for the <a> elements:
Example

```
li
{
        float:left;
}
a
{
        display:block;
        width:60px;
}
```

Example explained:

- float:left - use float to get block elements to slide next to each other
- display:block - Displaying the links as block elements makes the whole link area clickable (not just the text), and it allows us to specify the width
- width:60px - Since block elements take up the full width available, they cannot float next to each other. We specify the width of the links to 60px

Tip: Also take a look at our fully styled horizontal navigation bar example.

CSS Image Opacity / Transparency

Creating transparent images with CSS is easy.
Note: The CSS opacity property is a part of the W3C CSS3 recommendation.

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.
Regular image:

The same image with transparency:

Look at the following 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.

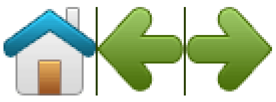
CSS Image Sprites

Image Sprites

An image sprite is a collection of images put into a single image.
A web page with many images can take a long time to load and generates multiple server requests.
Using image sprites will reduce the number of server requests and save bandwidth.

Image Sprites - Simple Example

Instead of using three separate images, we use this single image ("img_navsprites.gif"):



With CSS, we can show just the part of the image we need.

In the following example the CSS specifies which part of the "img_navsprites.gif" image to show:

Example

```
img.home
{
	width:46px;
	height:44px;
	background:url(img_navsprites.gif) 0 0;
}
```

Example explained:

- <img class="home" src="img_trans.gif" /> - Only defines a small transparent image because the src attribute cannot be empty. The displayed image will be the background image we specify in CSS
- width:46px;height:44px; - Defines the portion of the image we want to use
- background:url(img_navsprites.gif) 0 0; - Defines the background image and its position (left 0px, top 0px)

This is the easiest way to use image sprites, now we want to expand it by using links and hover effects.

Image Sprites - Create a Navigation List

We want to use the sprite image ("img_navsprites.gif") to create a navigation list.
We will use an HTML list, because it can be a link and also supports a background image:

Example

```
#navlist{position:relative;}
#navlist li{margin:0;padding:0;list-style:none;position:absolute;top:0;}
#navlist li, #navlist a{height:44px;display:block;}

#home{left:0px;width:46px;}
#home{background:url('img_navsprites.gif') 0 0;}

#prev{left:63px;width:43px;}
#prev{background:url('img_navsprites.gif') -47px 0;}

#next{left:129px;width:43px;}
#next{background:url('img_navsprites.gif') -91px 0;}
```

Example explained:

- #navlist{position:relative;} - position is set to relative to allow absolute positioning inside it
- #navlist li{margin:0;padding:0;list-style:none;position:absolute;top:0;} - margin and padding is set to 0, list-style is removed, and all list items are absolute positioned
- #navlist li, #navlist a{height:44px;display:block;} - the height of all the images are 44px

Now start to position and style for each specific part:

- #home{left:0px;width:46px;} - Positioned all the way to the left, and the width of the image is 46px
- #home{background:url(img_navsprites.gif) 0 0;} - Defines the background image and its position (left 0px, top 0px)
- #prev{left:63px;width:43px;} - Positioned 63px to the right (#home width 46px + some extra space between items), and the width is 43px.

- #prev{background:url('img_navsprites.gif') -47px 0;} - Defines the background image 47px to the right (#home width 46px + 1px line divider)
- #next{left:129px;width:43px;}- Positioned 129px to the right (start of #prev is 63px + #prev width 43px + extra space), and the width is 43px.
- #next{background:url('img_navsprites.gif') no-repeat -91px 0;} - Defines the background image 91px to the right (#home width 46px + 1px line divider + #prev width 43px + 1px line divider)

Image Sprites - Hover Effect

Now we want to add a hover effect to our navigation list.
Our new image ("img_navsprites_hover.gif") contains three navigation images and three images to use for hover effects:

Because this is one single image, and not six separate files, there will be **no loading delay** when a user hovers over the image.

We only add three lines of code to add the hover effect:

Example

```
#home a:hover{background: url('img_navsprites_hover.gif') 0 -45px;}
#prev a:hover{background: url('img_navsprites_hover.gif') -47px -45px;}
#next a:hover{background: url('img_navsprites_hover.gif') -91px -45px;}
```

Example explained:

- Since the list item contains a link, we can use the :hover pseudo-class
- #home a:hover{background: transparent url(img_navsprites_hover.gif) 0 -45px;} - For all three hover images we specify the same background position, only 45px further down

CSS Attribute Selectors

The `[attribute]` selector is used to select elements with a specified attribute.
The following example selects all <a> elements with a target attribute:

Example

```
a[target] {
  background-color: yellow;
}
```

CSS [attribute="value"] Selector

The `[attribute="value"]` selector is used to select elements with a specified attribute and value.
The following example selects all <a> elements with a target="_blank" attribute:

Example

```
a[target="_blank"] {
  background-color: yellow;
}
```

CSS [attribute~="value"] Selector

The `[attribute~="value"]` selector is used to select elements with an attribute value containing a specified word.

The following example selects all elements with a title attribute that contains a space-separated list of words, one of which is "flower":

Example

```
[title~="flower"] {
  border: 5px solid yellow;
}
```

CSS [attribute*="value"] Selector

The `[attribute*="value"]` selector is used to select elements whose attribute value contains a specified value.

The following example selects all elements with a class attribute value that contains "te":

Note: The value does not have to be a whole word!

Example

```
[class*="te"] {
  background: yellow;
}
```

CSS3 Introduced Media Queries

The **Media query** in CSS is used to create a responsive web design. It means that the view of a web page differs from system to system based on screen or media types. The breakpoint specifies for what device-width size, the content is just starting to break or deform.

Media queries can be used to check many things:

- width and height of the viewport
- width and height of the device
- Orientation
- Resolution

A media query consist of a media type that can contain one or more expression which can be either true or false. The result of the query is true if the specified media matches the type of device the document is displayed on. If the media query is true then a style sheet is applied.

Syntax:

```
@media not | only mediatype and (expression) {
   // Code content
}
```

Example: This example illustrates the CSS media query with the different device-width for making it responsive.

```
<!DOCTYPE html>
<html>
<head>
        <title>CSS media query</title>
        <style>
        body {
                text-align: center;
        }
        .gfg {
                font-size: 40px;
                font-weight: bold;
                color: green;
        }
        @media screen and (max-width:800px) {
                body {
                        text-align: center;
                        background-color: green;
                }
                .gfg {
                        font-size: 30px;
                        font-weight: bold;
                        color: white;
                }
                .geeks {
                        color: white;
                }
        }
        @media screen and (max-width:500px)
                {
                body {
                        text-align: center;
                        background-color: blue;
                        }
                }
        </style>
</head>
<body>
        <div class="gfg">GeeksforGeeks</div>
        <div class="geeks">A computer science portal for geeks</div>
</body>
</html>
```

Output: From the output, we can see that if the max-width of the screen is reduced to 800px then the background color changes to green & if the max-width of the screen is reduced to 500px then the background color will turn to blue. For the desktop size width, the background color will be white.

Media Types in CSS: There are many types of media types which are listed below:

- **all:** It is used for all media devices
- **print:** It is used for printer.
- **screen:** It is used for computer screens, smartphones, etc.
- **speech:** It is used for screen readers that read the screen aloud.

Features of Media query: There are many features of media query which are listed below:

- **color:** The number of bits per color component for the output device.
- **grid:** Checks whether the device is grid or bitmap.
- **height:** The viewport height.
- **aspect ratio:** The ratio between width and height of the viewport.
- **color-index:** The number of colors the device can display.
- **max-resolution:** The maximum resolution of the device using dpi and dpcm.
- **monochrome:** The number of bits per color on a monochrome device.
- **scan:** The scanning of output devices.
- **update:** How quickly can the output device modify.
- **width:** The viewport width.

Supported Browsers: The browser supported by CSS *media query* are listed below:

- Chrome 21.0 and above
- Mozilla 3.5 and above
- Microsoft Edge 12.0
- Opera 9.0 and above
- Internet Explorer 9.0 and above
- Safari 4.0 and above

CSS Transitions

SS transitions allows you to change property values smoothly, over a given duration.

- transition
- transition-delay
- transition-duration
- transition-property
- transition-timing-function

CSS Syntax

- **transition:** property duration timing-function delay|initial|inherit;
- **transition-delay:** time|initial|inherit;
- **transition-duration:** time|initial|inherit;
- **transition-property:** none|all|property|initial|inherit;
- **transition-timing-function:** linear|ease|ease-in|ease-out|ease-in-out|step-start|step-end|steps(int,start|end)|cubic-bezier(n,n,n,n)|initial|inherit;

```
<!DOCTYPE html>
<html>
        <head>
        <style>
                div {
```

```
                          width: 100px;
                        height: 100px;
                          background: red;
                          transition: width 2s, height 4s;
                        }
                        div:hover {
                          width: 300px;
                          height: 300px;
                        }
            </style>
            </head>
            <body>
                        <h1>The transition Property</h1>
                        <p>Hover over the div element below, to see the transition effect:</p>
            <div></div>
            </body>
</html>
```

Specify the Speed Curve of the Transition

The transition-timing-function property specifies the speed curve of the transition effect.

The transition-timing-function property can have the following values:

- ease - specifies a transition effect with a slow start, then fast, then end slowly (this is default)
- **linear** - specifies a transition effect with the same speed from start to end
- **ease-in** - specifies a transition effect with a slow start
- **ease-out** - specifies a transition effect with a slow end
- **ease-in-out** - specifies a transition effect with a slow start and end
- **cubic-bezier(n,n,n,n)** - lets you define your own values in a cubic-bezier function

Example 1:

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  width: 100px;
  height: 100px;
  background: red;
  transition: width 2s;
}
```

```
#div1 {transition-timing-function: linear;}
#div2 {transition-timing-function: ease;}
#div3 {transition-timing-function: ease-in;}
#div4 {transition-timing-function: ease-out;}
#div5 {transition-timing-function: ease-in-out;}

div:hover {
  width: 300px;
}
</style>
</head>
<body>
<h1>The transition-timing-function Property</h1>
<p>Hover over the div elements below, to see the different speed curves:</p>
<div id="div1">linear</div><br>
<div id="div2">ease</div><br>
<div id="div3">ease-in</div><br>
<div id="div4">ease-out</div><br>
<div id="div5">ease-in-out</div><br>
</body>
</html>
```

Example 2:

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  width: 100px;
  height: 100px;
  background: red;
  transition-property: width;
  transition-duration: 2s;
  transition-timing-function: linear;
  transition-delay: 1s;
/* transition: width 2s linear 1s; */
}
div:hover {
  width: 300px;
}
</style>
</head>
<body>
<h1>The transition Properties Specified One by One</h1>
<p>Hover over the div element below, to see the transition effect:</p>
<div></div>
<p><b>Note:</b> The transition effect has a 1 second delay before starting.</p>
</body>
</html>
```

CSS Animations

CSS allows animation of HTML elements without using JavaScript or Flash!

CSS Animation Properties

The following table lists the @keyframes rule and all the CSS animation properties:

Property	Description
@keyframes	Specifies the animation code
animation	A shorthand property for setting all the animation properties
animation-delay	Specifies a delay for the start of an animation
animation-direction	Specifies whether an animation should be played forwards, backwards or in alternate cycles
animation-duration	Specifies how long time an animation should take to complete one cycle
animation-fill-mode	Specifies a style for the element when the animation is not playing (before it starts, after it ends, or both)
animation-iteration-count	Specifies the number of times an animation should be played
animation-name	Specifies the name of the @keyframes animation
animation-play-state	Specifies whether the animation is running or paused
animation-timing-function	Specifies the speed curve of the animation

CSS Syntax

animation: name duration timing-function delay iteration-count direction fill-mode play-state;

animation-delay: time|initial|inherit;

animation-direction: normal|reverse|alternate|alternate-reverse|initial|inherit;

animation-duration: time|initial|inherit;

animation-fill-mode: none|forwards|backwards|both|initial|inherit;

animation-iteration-count: number|infinite|initial|inherit;

animation-name: keyframename|none|initial|inherit;

animation-play-state: paused|running|initial|inherit;

animation-timing-function: linear|ease|ease-in|ease-out|ease-in-out|step-start|step-end|steps(int,start|end)|cubic-bezier(n,n,n,n)|initial|inherit;

animation-play-state: paused|running|initial|inherit;

@keyframes animationname {keyframes-selector {css-styles;}}

Browser Support

The numbers in the table specify the first browser version that fully supports the property. Numbers followed by -webkit-, -moz-, or -o- specify the first version that worked with a prefix.

The @keyframes Rule

When you specify CSS styles inside the @keyframes rule, the animation will gradually change from the current style to the new style at certain times.

To get an animation to work, you must bind the animation to an element.

Example 1:

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  width: 100px;
  height: 100px;
  background-color: red;
  position: relative;
  animation-name: example;
  animation-duration: 4s;
}
@keyframes example {
  0%   {background-color:red; left:0px; top:0px;}
  25%  {background-color:yellow; left:200px; top:0px;}
  50%  {background-color:blue; left:200px; top:200px;}
  75%  {background-color:green; left:0px; top:200px;}
  100% {background-color:red; left:0px; top:0px;}
}
</style>
</head>
<body>
<h1>CSS Animation</h1>
<div></div>
<p><b>Note:</b> When an animation is finished, it goes back to its original style.</p>
</body>
```

```
</html>
```

Example 2:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Example of CSS3 Infinite Translate Animation</title>
<style>
   .box {
      margin: 50px;
      width:103px;
      height:130px;
      background: url("/examples/images/octopus.png") no-repeat;
      position: relative;
      /* Chrome, Safari, Opera */
      -webkit-animation: repeatit 2s linear 0s infinite alternate;
      /* Standard syntax */
      animation: repeatit 2s linear 0s infinite alternate;
   }
   /* Chrome, Safari, Opera */
   @-webkit-keyframes repeatit {
      from {left: 0;}
      to {left: 50%;}
   }
   /* Standard syntax */
   @keyframes repeatit {
      from {left: 0;}
      to {left: 50%;}
   }
</style>
</head>
<body>
   <div class="box"></div>
</body>
```

</html>

CSS transform Property

Definition and Usage

The transform property applies a 2D or 3D transformation to an element. This property allows you to rotate, scale, move, skew, etc., elements.

CSS Syntax

- **transform:** none | transform-functions | initial | inherit;
- **transform-origin:** x-axis y-axis z-axis | initial | inherit;
    - **x-axis** - Possible values: left | center | right | length | %
    - **y-axis** - Possible values: top | center | bottom | length | %
    - **z-axis** - Possible values: length
- **transform-style:** flat | preserve-3d | initial | inherit;

CSS 2D Transforms

CSS transforms allow you to move, rotate, scale, and skew elements.

Mouse over the element below to see a 2D transformation:

CSS 2D Transforms Methods

With the CSS transform property you can use the following 2D transformation methods:

- translate( )
- rotate( )
- scaleX( )
- scaleY( )
- scale( )
- skewX( )
- skewY( )
- skew( )
- matrix( )

CSS Transform Properties

The following table lists all the 2D transform properties:

Property	Description
transform	Applies a 2D or 3D transformation to an element
transform-origin	Allows you to change the position on transformed elements

CSS 2D Transform Methods

Function	Description
matrix(*n,n,n,n,n,n*)	Defines a 2D transformation, using a matrix of six values
translate(*x,y*)	Defines a 2D translation, moving the element along the X- and the Y-axis
translateX(*n*)	Defines a 2D translation, moving the element along the X-axis

translateY(*n*)	Defines a 2D translation, moving the element along the Y-axis
scale(*x,y*)	Defines a 2D scale transformation, changing the elements width and height
scaleX(*n*)	Defines a 2D scale transformation, changing the element's width
scaleY(*n*)	Defines a 2D scale transformation, changing the element's height
rotate(*angle*)	Defines a 2D rotation, the angle is specified in the parameter
skew(*x-angle,y-angle*)	Defines a 2D skew transformation along the X- and the Y-axis
skewX(*angle*)	Defines a 2D skew transformation along the X-axis
skewY(*angle*)	Defines a 2D skew transformation along the Y-axis

The following table lists all the 3D transform properties:

Property	**Description**
transform	Applies a 2D or 3D transformation to an element
transform-origin	Allows you to change the position on transformed elements
transform-style	Specifies how nested elements are rendered in 3D space
perspective	Specifies the perspective on how 3D elements are viewed
perspective-origin	Specifies the bottom position of 3D elements
backface-visibility	Defines whether or not an element should be visible when not facing the screen

CSS 3D Transform Methods

Function	**Description**
matrix3d (*n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n*)	Defines a 3D transformation, using a 4x4 matrix of 16 values
translate3d(*x,y,z*)	Defines a 3D translation
translateX(*x*)	Defines a 3D translation, using only the value for the X-axis
translateY(*y*)	Defines a 3D translation, using only the value for the Y-axis
translateZ(*z*)	Defines a 3D translation, using only the value for the Z-axis
scale3d(*x,y,z*)	Defines a 3D scale transformation
scaleX(*x*)	Defines a 3D scale transformation by giving a value for the X-axis

scaleY(*y*)	Defines a 3D scale transformation by giving a value for the Y-axis
scaleZ(*z*)	Defines a 3D scale transformation by giving a value for the Z-axis
rotate3d(*x,y,z,angle*)	Defines a 3D rotation
rotateX(*angle*)	Defines a 3D rotation along the X-axis
rotateY(*angle*)	Defines a 3D rotation along the Y-axis
rotateZ(*angle*)	Defines a 3D rotation along the Z-axis
perspective(*n*)	Defines a perspective view for a 3D transformed element

```
/* Keyword values */
transform: none;
/* Multiple function values */
transform: translateX(10px) rotate(10deg) translateY(5px);
transform: perspective(500px) translate(10px, 0, 20px) rotateY(3deg);
/* Global values */
transform: inherit | initial | revert | revert-layer | unset
/* Function values */
transform: matrix(1.0, 2.0, 3.0, 4.0, 5.0, 6.0);
transform: matrix3d(1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1);
transform: perspective(17px);
transform: rotate(0.5turn);
transform: rotate3d(1, 2.0, 3.0, 10deg);
transform: rotateX(10deg);
transform: rotateY(10deg);
transform: rotateZ(10deg);
transform: translate(12px, 50%);
transform: translate3d(12px, 50%, 3em);
transform: translateX(2em);
transform: translateY(3in);
transform: translateZ(2px);
transform: scale(2, 0.5);
transform: scale3d(2.5, 1.2, 0.3);
transform: scaleX(2);
transform: scaleY(0.5);
transform: scaleZ(0.3);
```

transform: skew(30deg, 20deg);

transform: skewX(30deg);

transform: skewY(1.07rad);

How to validate your Code of HTML.

1. Open Your web Browser and type this Url : https://jigsaw.w3.org/css-validator/
2. There you will find 3 option you can select one of them and check your code.

Websites for Study Reference

1: https://developer.mozilla.org/en-US/docs/Web/CSS
2: https://css-tricks.com/