Cascading Style Sheets

CSS describes how HTML elements are to be displayed on screen, paper, or in other media

CSS saves a lot of work. It can control the layout of multiple web pages all at once

External style sheets are stored in CSS files

CSS is used to define styles for your web pages, including the design, layout and variations in display for different devices and screen sizes.

CSS Solved a Big Problem

HTML was NEVER intended to contain tags for formatting a web page!

HTML was created to **describe the content** of a web page, like:

<h1>This is a heading</h1>

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

When tags like <font>, and colour attributes were added to the HTML 3.2 specification, it started a nightmare for web developers. Development of large websites, 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.

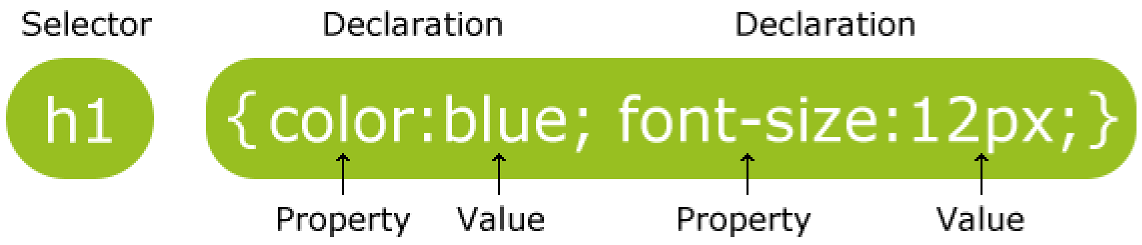
CSS removed the style formatting from the HTML page

The style definitions are normally saved in external .css files.

With an external stylesheet file, you can change the look of an entire website by changing just one file!

CSS Syntax

A CSS rule-set consists of a selector and a declaration block:



The selector points to the HTML element you want to style.

The declaration block contains one or more declarations separated by semicolons.

Each declaration includes a CSS property name and a value, separated by a colon.

A CSS declaration always ends with a semicolon, and declaration blocks are surrounded by curly braces.

In the following example all <p> elements will be center-aligned, with a red text color:

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

CSS Selectors

CSS selectors are used to "find" (or select) HTML elements based on their element name, id, class, attribute, and more.

The element Selector

The element selector selects elements based on the element name.

You can select all <p> elements on a page like this (in this case, all <p> elements will be center-aligned, with a red text color):

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

The id Selector

The id selector uses the id attribute of an HTML element to select a specific element.

The id of an element should be unique within a page, so the id selector is used to select one unique element!

To select an element with a specific id, write a hash (#) character, followed by the id of the element.

The style rule below will be applied to the HTML element with id="para1":

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

Note: An id name cannot start with a number

The class Selector

The class selector selects elements with a specific class attribute.

To select elements with a specific class, write a period (.) character, followed by the name of the class.

In the example below, all HTML elements with class="center" will be red and center-aligned:

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

You can also specify that only specific HTML elements should be affected by a class.

In the example below, only <p> elements with class="center" will be center-aligned:

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

HTML elements can also refer to more than one class.

In the example below, the <p> element will be styled according to class="center" and to class="large":

```
<p class="center large">This paragraph refers to two classes.</p>
```

Grouping Selectors

If you have elements with the same style definitions, like this:

```
h1 {
   text-align: center;
   color: red;
}

h2 {
   text-align: center;
   color: red;
}

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

It will be better to group the selectors, to minimize the code.

To group selectors, separate each selector with a comma.

In the example below we have grouped the selectors from the code above:

```
h1, h2, p {
   text-align: center;
   color: red;
}
```

CSS Comments

Comments are used to explain the code, and may help when you edit the source code at a later date.

Comments are ignored by browsers.

A CSS comment starts with /* and ends with */. Comments can also span multiple lines:

```
p {
   color: red;
   /* This is a single-line comment */
   text-align: center;
}
```

```
/* This is a multi-line comment */
```

Exercise 1

```
<!DOCTYPE html>

<html>

<head>

<style>

</style>

</head>

<body>

<h1>This is a Heading</h1>

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

<p>This is another paragraph.</p>

</body>

</html>
```

Exercise 2

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

</style>
</head>
<body>

<h1>This is a Heading</h1>
<p id="para1">This is a paragraph.</p>
<p>This is another paragraph.</p>

</body>
</html>
```

Exercise 3

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

</style>
</head>
```

```
<body>

<h1>This is a Heading</h1>
<p>This is a paragraph.</p>
<p class="colortext">This is another paragraph.</p>
<p class="colortext">This is also a paragraph.</p>

</body>
</html>
```

Exercise 4

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

</style>
</head>
<body>

<h1>This is a heading</h1>
<h2>This is a smaller heading</h2>
<p>This is a paragraph.</p>
<p>This is another paragraph.</p>

</body>
</html>
```

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

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

With an external style sheet, you can change the look of an entire website by changing just one file!

Each page must include a reference to the external style sheet file inside the <link> element. The <link> element 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. The style sheet file must be saved with a .css extension.

Here is how the "mystyle.css" looks:

```
body {
   background-color: lightblue;
}

h1 {
   color: navy;
   margin-left: 20px;
}
```

Note:
Do not add a space between the property value and the unit (such as margin-left: 20 px;). The correct way is: margin-left: 20px;

Internal Style Sheet

An internal style sheet may be used if one single page has a unique style.

Internal styles are defined within the <style> element, inside the <head> section of an HTML page:

```
<head>
<style>
body {
   background-color: linen;
}

h1 {
   color: maroon;
   margin-left: 40px;
}
</style>
</head>
```

Inline Styles

An inline style may be used to apply a unique style for a single element.

To use inline styles, add the style attribute to the relevant element. The style attribute can contain any CSS property.

The example below shows how to change the color and the left margin of a <h1> element:

```
<h1 style="color:blue;margin-left:30px;">This is a heading</h1>
```

Multiple Style Sheets

If some properties have been defined for the same selector (element) in different style sheets, the value from the last read style sheet will be used.

Assume that an external style sheet has the following style for the <h1> element:

```
h1 {
    color: navy;
}
```

then, assume that an internal style sheet also has the following style for the <h1> element:

```
h1 {
    color: orange;
}
```

If the internal style is defined after the link to the external style sheet, the <h1> elements will be "orange":

```
<head>
<link rel="stylesheet" type="text/css" href="mystyle.css">
<style>
h1 {
    color: orange;
}
</style>
</head>
```

However, if the internal style is defined before the link to the external style sheet, the <h1> elements will be "navy":

```
<head>
<style>
h1 {
    color: orange;
}
</style>
<link rel="stylesheet" type="text/css" href="mystyle.css">
</head>
```

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 one has the highest priority:

1. Inline style (inside an HTML element)
2. External and internal style sheets (in the head section)
3. Browser default

So, an inline style (inside a specific 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 a browser default value.

Exercise

```
<!DOCTYPE html>

<html>

<head>
```

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

<style>

p {

   color: red;

}

</style>

</head>

<body style="background-color: lightcyan">

<h1>This is a Heading</h1>

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

<p>This is another paragraph.</p>

</body>

</html>
```

Colors in CSS are most often specified by:

- a valid color name - like "red"
- an RGB value - like "rgb(255, 0, 0)"
- a HEX value - like "#ff0000"
- Color names are case-insensitive: "Red" is the same as "red" or "RED".
- HTML and CSS supports 140 standard color names.

RGB (Red, Green, Blue)

- RGB color values can be specified using this formula: rgb(red, green, blue).
- Each parameter (red, green, blue) defines the intensity of the color between 0 and 255.
- For example, rgb(255,0,0) is displayed as red, because red is set to its highest value (255) and the others are set to 0. Experiment by mixing the RGB values below:

Hexadecimal Colors

RGB values can also be specified using **hexadecimal** color values in the form: *#RRGGBB*, where RR (red), GG (green) and BB (blue) are hexadecimal values between 00 and FF (same as decimal 0-255).

For example, #FF0000 is displayed as red, because red is set to its highest value (FF) and the others are set to the lowest value (00). **Note:** HEX values are case-insensitive: "#ff0000" is the same as "FF0000".

Background Color

The background-color property specifies the background color of an element.

The background color of a page is set like this:

```
body {
   background-color: lightblue;
}
```

With CSS, a color is most often specified by:

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

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:

```
h1 {
   background-color: green;
}

div {
   background-color: lightblue;
}

p {
   background-color: yellow;
}
```

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:

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

Below is an example of a **bad** combination of text and background image. The text is hardly readable:

```
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:

```
body {
   background-image: url("gradient_bg.png");
}
```

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

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

Background Image - Set position and no-repeat

Showing the background image only once is also specified by the background-repeat property:

```
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:

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

Background Image - Fixed position

To specify that the background image should be fixed (will not scroll with the rest of the page), use the background-attachment property:

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

Background - Shorthand property

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

The shorthand property for background is background:

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

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

- background-color
- background-image
- background-repeat

- background-attachment
- background-position

It does not matter if one of the property values is missing, as long as the other ones 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 Border Properties

The CSS border properties allow you to specify the style, width, and color of an element's border.

I have borders on all sides.

I have a red bottom border

I have rounded borders.

I have a blue left border.

Border Style

The border-style property specifies what kind of border to display.

The following values are allowed:

- dotted - Defines a dotted border
- dashed - Defines a dashed border
- solid - Defines a solid border
- double - Defines a double border
- 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
- none - Defines no border
- hidden - Defines a hidden border

The border-style property can have from one to four values (for the top border, right border, bottom border, and the left border).

Example

```
p.dotted {border-style: dotted;}
p.dashed {border-style: dashed;}
p.solid {border-style: solid;}
p.double {border-style: double;}
p.groove {border-style: groove;}
p.ridge {border-style: ridge;}
p.inset {border-style: inset;}
p.outset {border-style: outset;}
p.none {border-style: none;}
p.hidden {border-style: hidden;}
p.mix {border-style: dotted dashed solid double;}
```

Result:

A dotted border.

A dashed border.

A solid border.

A double border.

A groove border. The effect depends on the border-color value.

A ridge border. The effect depends on the border-color value.

An inset border. The effect depends on the border-color value.

An outset border. The effect depends on the border-color value.

No border.

A hidden border.

A mixed border.

Border Width

The border-width property specifies the width of the four borders.

The width can be set as a specific size (in px, pt, cm, em, etc) or by using one of the three pre-defined values: thin, medium, or thick.

The border-width property can have from one to four values (for the top border, right border, bottom border, and the left border).

5px border-width

Example

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

p.two {
    border-style: solid;
    border-width: medium;
}

p.three {
    border-style: solid;
    border-width: 2px 10px 4px 20px;
}
```

Border Color

The border-color property is used to set the color of the four borders.

The color can be set by:

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

The border-color property can have from one to four values (for the top border, right border, bottom border, and the left border).

If border-color is not set, it inherits the color of the element.

Red border

```
p.one {
    border-style: solid;
    border-color: red;
}

p.two {
    border-style: solid;
```

```
  border-color: green;
}

p.three {
  border-style: solid;
  border-color: red green blue yellow;
}
```

Border - Individual Sides

From the examples above you have seen that it is possible to specify a different border for each side.

In CSS, there is also properties for specifying each of the borders (top, right, bottom, and left):

Different Border Styles

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

The example above gives the same result as this:

```
p {
  border-style: dotted solid;
}
```

So, here is how it works:

If the border-style property has four values:

- **border-style: dotted solid double dashed;**
  - top border is dotted
  - right border is solid
  - bottom border is double
  - left border is dashed

If the border-style property has three values:

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

If the border-style property has two values:

- **border-style: dotted solid;**
  - top and bottom borders are dotted
  - right and left borders are solid

If the border-style property has one value:

- **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 individual border properties in one property.

The border property is a shorthand property for the following individual border properties:

- border-width
- border-style (required)
- border-color

```
p {
    border: 5px solid red;
}
```

Result:

Some text

You can also specify all the individual border properties for just one side:

Left Border

```
p {
    border-left: 6px solid red;
    background-color: lightgrey;
}
```

Result:

Some text

Bottom Border

```
p {
    border-bottom: 6px solid red;
    background-color: lightgrey;
}
```

Result:

Some text

Rounded Borders

The border-radius property is used to add rounded borders to an element:

Normal border

Round border

Rounder border

Roundest border

```
p {
    border: 2px solid red;
    border-radius: 5px;
}
```

Note: The border-radius property is not supported in IE8 and earlier versions.

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-radius	Sets all the four border-*-radius properties for rounded corners
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 Margins

The CSS margin properties are used to generate space around elements.

The margin properties set the size of the white space outside the border.

With CSS, you have full control over the margins. There are CSS properties for setting the margin for each side of an element (top, right, bottom, and left).

Margin - Individual Sides

CSS has properties for specifying the margin for each side of an element:

- margin-top
- margin-right
- margin-bottom
- margin-left

All the margin properties can have the following values:

- auto - the browser calculates the margin
- *length* - specifies a margin in px, pt, cm, etc.
- *%* - specifies a margin in % of the width of the containing element
- inherit - specifies that the margin should be inherited from the parent element

Tip: Negative values are allowed.

The following example sets different margins for all four sides of a <p> element:

```
p {
   margin-top: 100px;
   margin-bottom: 100px;
   margin-right: 150px;
   margin-left: 80px;
}
```

Margin - Shorthand Property

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

The margin property is a shorthand property for the following individual margin properties:

- margin-top
- margin-right
- margin-bottom
- margin-left

```
p {
   margin: 100px 150px 100px 80px;
}
```

So, here is how it works:

If the margin property has four values:

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

If the margin property has three values:

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

If the margin property has two values:

- **margin: 25px 50px;**
    - top and bottom margins are 25px
    - right and left margins are 50px

If the margin property has one value:

- **margin: 25px;**
    - all four margins are 25px

The auto Value

You can set the margin property to auto to horizontally center the element within its container.

The element will then take up the specified width, and the remaining space will be split equally between the left and right margins:

```
div {
   width: 300px;
   margin: auto;
   border: 1px solid red;
}
```

The inherit Value

This example lets the left margin be inherited from the parent element:

```
div.container {
   border: 1px solid red;
   margin-left: 100px;
}

p.one {
   margin-left: inherit;
}
```

Margin Collapse

Top and bottom margins of elements are sometimes collapsed into a single margin that is equal to the largest of the two margins.

This does not happen on left and right margins! Only top and bottom margins!

Look at the following example:

```
h1 {
   margin: 0 0 50px 0;
}

h2 {
   margin: 20px 0 0 0;
}
```

In the example above, the <h1> element has a bottom margin of 50px. The <h2> element has a top margin set to 20px.

Common sense would seem to suggest that the vertical margin between the <h1> and the <h2> would be a total of 70px (50px + 20px). But due to margin collapse, the actual margin ends up being 50px.

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 are used to generate space around content.

The padding clears an area around the content (inside the border) of an element.

With CSS, you have full control over the padding. There are CSS properties for setting the padding for each side of an element (top, right, bottom, and left).

Padding - Individual Sides

CSS has properties for specifying the padding for each side of an element:

- padding-top
- padding-right
- padding-bottom
- padding-left

All the padding properties can have the following values:

- *length* - specifies a padding in px, pt, cm, etc.
- *%* - specifies a padding in % of the width of the containing element
- inherit - specifies that the padding should be inherited from the parent element

The following example sets different padding for all four sides of a <p> element:

```
p {
   padding-top: 50px;
   padding-right: 30px;
   padding-bottom: 50px;
   padding-left: 80px;
}
```

Padding - Shorthand Property

To shorten the code, it is possible to specify all the padding properties in one property.

The padding property is a shorthand property for the following individual padding properties:

- padding-top
- padding-right
- padding-bottom
- padding-left

```
p {
   padding: 50px 30px 50px 80px;
}
```

So, here is how it works:

If the padding property has four values:

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

If the padding property has three values:

- **padding: 25px 50px 75px;**

  - top padding is 25px
  - right and left paddings are 50px
  - bottom padding is 75px

If the padding property has two values:

- **padding: 25px 50px;**
  - top and bottom paddings are 25px
  - right and left paddings are 50px

If the padding property has one value:

- **padding: 25px;**
  - all four paddings are 25px

```
div.ex1 {
   padding: 25px 50px 75px 100px;
}

div.ex2 {
   padding: 25px 50px 75px;
}

div.ex3 {
   padding: 25px 50px;
}

div.ex4 {
   padding: 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

Setting height and width

The height and width properties are used to set the height and width of an element.

The height and width can be set to auto (this is default. Means that the browser calculates the height and width), or be specified in *length values*, like px, cm, etc., or in percent (%) of the containing block.

This element has a height of 200 pixels and a width of 50%

```
div {
   height: 200px;
   width: 50%;
   background-color: powderblue;
}
```

This element has a height of 100 pixels and a width of 500 pixels.

```
div {
   height: 100px;
   width: 500px;
   background-color: powderblue;
}
```

Note: The height and width properties do not include padding, borders, or margins; they set the height/width of the area inside the padding, border, and margin of the element!

Setting max-width

The max-width property is used to set the maximum width of an element.

The max-width can be specified in *length values*, like px, cm, etc., or in percent (%) of the containing block, or set to none (this is default. Means that there is no maximum width).

The problem with the <div> above occurs when the browser window is smaller than the width of the element (500px). The browser then adds a horizontal scrollbar to the page.

Using max-width instead, in this situation, will improve the browser's handling of small windows.

Tip: Drag the browser window to smaller than 500px wide, to see the difference between the two divs!

This element has a height of 100 pixels and a max-width of 500 pixels.

Note: The value of the max-width property overrides width.

The following example shows a <div> element with a height of 100 pixels and a max-width of 500 pixels:

```
div {
   max-width: 500px;
   height: 100px;
   background-color: powderblue;
}
```

CSS Dimension Properties

Property	Description
height	Sets the height of an element
max-height	Sets the maximum height of an element
max-width	Sets the maximum width of an element
min-height	Sets the minimum height of an element
min-width	Sets the minimum width of an element
width	Sets the width of an element

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 every HTML element. It consists of: margins, borders, padding, and the actual content. The image below illustrates the box model:

Explanation of the different parts:

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

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

```
div {
   width: 300px;
   border: 25px solid green;
```

```
    padding: 25px;
    margin: 25px;
}
```

Width and Height of an Element

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

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 padding, borders and margins.

Assume we want to style a <div> element to have a total width of 350px:

```
div {
    width: 320px;
    padding: 10px;
    border: 5px solid gray;
    margin: 0;
}
```

Here is the math:

320px (width) + 20px (left + right padding) + 10px (left + right border)+ 0px (left + right margin)
= 350px

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

Note for old IE: Internet Explorer 8 and earlier versions, include padding and border in the width property. To fix this problem, add a <!DOCTYPE html> to the HTML page.

CSS Outline

The CSS outline properties specify the style, color, and width of an 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.

This element has a thin black border and an outline that is 10px wide and green.

Outline Style

The outline-style property specifies the style of the outline.

The outline-style property can have one of the following values:

- dotted - Defines a dotted outline
- dashed - Defines a dashed outline
- solid - Defines a solid outline
- double - Defines a double outline
- groove - Defines a 3D grooved outline. The effect depends on the outline-color value
- ridge - Defines a 3D ridged outline. The effect depends on the outline-color value
- inset - Defines a 3D inset outline. The effect depends on the outline-color value
- outset - Defines a 3D outset outline. The effect depends on the outline-color value
- none - Defines no outline
- hidden - Defines a hidden outline

The following example first sets a thin black border around each <p> element, then it shows the different outline-style values:

```
p {
   border: 1px solid black;
   outline-color: red;
}

p.dotted {outline-style: dotted;}
p.dashed {outline-style: dashed;}
p.solid {outline-style: solid;}
p.double {outline-style: double;}
p.groove {outline-style: groove;}
p.ridge {outline-style: ridge;}
p.inset {outline-style: inset;}
p.outset {outline-style: outset;}
```

Result:

A dotted outline.

A dashed outline.

A solid outline.

A double outline.

A groove outline. The effect depends on the outline-color value.

A ridge outline. The effect depends on the outline-color value.

An inset outline. The effect depends on the outline-color value.

An outset outline. The effect depends on the outline-color value.

Note: None of the OTHER CSS outline properties described below will have ANY effect unless the outline-style property is set!

Outline Color

The outline-color property is used to set the color of the outline.

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"
- invert - performs a color inversion (which ensures that the outline is visible, regardless of color background)

```
p {
   border: 1px solid black;
   outline-style: double;
   outline-color: red;
}
```

Result:

A colored outline.

Outline Width

The outline-width property specifies the width of the outline.

The width can be set as a specific size (in px, pt, cm, em, etc) or by using one of the three pre-defined values: thin, medium, or thick.

```
p {border: 1px solid black;}

p.one {
   outline-style: double;
   outline-color: red;
   outline-width: thick;
}

p.two {
   outline-style: double;
   outline-color: green;
   outline-width: 3px;
}
```

Result:

A thick outline.

A thinner outline.

Outline - Shorthand property

To shorten the code, it is also possible to specify all the individual outline properties in one property.

The outline property is a shorthand property for the following individual outline properties:

- outline-width
- outline-style (required)
- outline-color

```
p {
   border: 1px solid black;
   outline: 5px dotted red;
}
```

Result:

An outline.

All CSS Outline Properties

Property	Description
outline	Sets all the outline properties in one declaration
outline-color	Sets the color of an outline
outline-offset	Specifies the space between an outline and the edge or border of an element
outline-style	Sets the style of an outline
outline-width	Sets the width of an outline

Text Color

The color property is used to set the color of the text.

With CSS, a color is most often specified by:

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

Look at CSS Color Values for a complete list of possible color values.

The default text color for a page is defined in the body selector.

```
body {
   color: blue;
}
```

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

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

Text Alignment

The text-align property is used to set the horizontal alignment of a text.

A text can be left or right aligned, centered, or justified.

The following example shows center aligned, and left and right aligned text (left alignment is default if text direction is left-to-right, and right alignment is default if text direction is right-to-left):

```
h1 {
   text-align: center;
}

h2 {
   text-align: left;
}

h3 {
   text-align: right;
}
```

When the text-align property 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):

```
div {
   text-align: justify;
}
```

Text Decoration

The text-decoration property is used to set or remove decorations from text.

The value text-decoration: none; is often used to remove underlines from links:

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

The other text-decoration values are used to decorate text:

```
h1 {
   text-decoration: overline;
}

h2 {
   text-decoration: line-through;
}
```

```
h3 {
    text-decoration: underline;
}
```

Note: It is not recommended to underline text that is not a link, as this often confuses the reader.

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:

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

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

Text Indentation

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

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

Letter Spacing

The letter-spacing property is used to specify the space between the characters in a text.

The following example demonstrates how to increase or decrease the space between characters:

```
h1 {
    letter-spacing: 3px;
}

h2 {
    letter-spacing: -3px;
}
```

Line Height

The line-height property is used to specify the space between lines:

```
p.small {
    line-height: 0.8;
}

p.big {
    line-height: 1.8;
}
```

Text Direction

The direction property is used to change the text direction of an element:

```
div {
   direction: rtl;
}
```

Word Spacing

The word-spacing property is used to specify the space between the words in a text.

The following example demonstrates how to increase or decrease the space between words:

```
h1 {
   word-spacing: 10px;
}

h2 {
   word-spacing: -5px;
}
```

Text Shadow

The text-shadow property adds shadow to text.

The following example specifies the position of the horizontal shadow (3px), the position of the vertical shadow (2px) and the color of the shadow (red):

```
h1 {
   text-shadow: 3px 2px red;
}
```

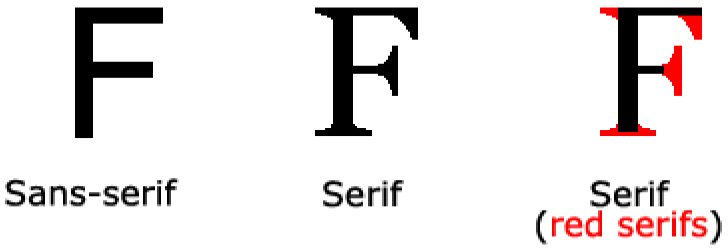
All CSS Text Properties

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
text-overflow	Specifies how overflowed content that is not displayed should be signaled to the user
unicode-bidi	Used together with the direction property to set or return whether the text should be overridden to support multiple languages in the same document
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

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

Difference Between Serif and Sans-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

Note: 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, and so on.

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: "Times New Roman".

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

```
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)

```
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

Note: 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;
}
```

Tip: If you use pixels, you can still use the zoom tool to resize the entire page.

Set Font Size With Em

To allow users to resize the text (in the browser menu), 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*

```
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!

Font Weight

The font-weight property specifies the weight of a font:

Example

```
p.normal {
   font-weight: normal;
}

p.thick {
   font-weight: bold;
}
```

Font Variant

The font-variant property specifies whether or not a text should be displayed in a small-caps font.

In a small-caps font, all lowercase letters are converted to uppercase letters. However, the converted uppercase letters appears in a smaller font size than the original uppercase letters in the text.

```
p.normal {
   font-variant: normal;
}

p.small {
   font-variant: small-caps;
}
```

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

How To Add Icons

The simplest way to add an icon to your HTML page, is with an icon library, such as Font Awesome.

Add the name of the specified icon class to any inline HTML element (like <i> or<span>).

All the icons in the icon libraries below, are scalable vectors that can be customized with CSS (size, color, shadow, etc.)

Font Icons

To use the Font Awesome icons, add the following line inside the <head> section of your HTML page:

```
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
```

Note: No downloading or installation is required!

```
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet"href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
</head>
<body>

<i class="fa fa-cloud"></i>
<i class="fa fa-heart"></i>
<i class="fa fa-car"></i>
<i class="fa fa-file"></i>
<i class="fa fa-bars"></i>
```

```
</body>
</html>
```

Result:

Bootstrap Icons

To use the Bootstrap glyphicons, add the following line inside the `<head>` section of your HTML page:

```
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
```

Note: No downloading or installation is required!

```
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
</head>
<body>

<i class="glyphicon glyphicon-cloud"></i>
<i class="glyphicon glyphicon-remove"></i>
<i class="glyphicon glyphicon-user"></i>
<i class="glyphicon glyphicon-envelope"></i>
<i class="glyphicon glyphicon-thumbs-up"></i>

</body>
</html>
```

Result:

Google Icons

To use the Google icons, add the following line inside the `<head>` section of your HTML page:

```
<link rel="stylesheet" href="https://fonts.googleapis.com/icon?family=Material+Icons">
```

Note: No downloading or installation is required!

```
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" href="https://fonts.googleapis.com/icon?family=Material+Icons">
</head>
<body>

<i class="material-icons">cloud</i>
<i class="material-icons">favorite</i>
<i class="material-icons">attachment</i>
<i class="material-icons">computer</i>
<i class="material-icons">traffic</i>
```

```
</body>
</html>
```

Styling Links

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

Example

```
a {
   color: hotpink;
}
```

In addition, links 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

```
/* unvisited link */
a:link {
   color: red;
}

/* visited link */
a:visited {
   color: green;
}

/* mouse over link */
a:hover {
   color: hotpink;
}

/* selected link */
a:active {
   color: blue;
}
```

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

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 can be used to specify a background color for links:

Example

```
a:link {
    background-color: yellow;
}

a:visited {
    background-color: cyan;
}

a:hover {
    background-color: lightgreen;
}

a:active {
    background-color: hotpink;
}
```

Advanced - Link Buttons

This example demonstrates a more advanced example where we combine several CSS properties to display links as boxes/buttons:

```
a:link, a:visited {
    background-color: #f44336;
    color: white;
    padding: 14px 25px;
    text-align: center;
    text-decoration: none;
    display: inline-block;
}

a:hover, a:active {
    background-color: red;
}
```

HTML Lists and CSS List Properties

In HTML, there are two main types of lists:

- unordered lists (<ul>) - the list items are marked with bullets
- ordered lists (<ol>) - the list items are marked with numbers or letters

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
- Add background colors to lists and list items

Different List Item Markers

The list-style-type property specifies the type of list item marker.

The following example shows some of the available list item markers:

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;
}
```

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

An Image as The List Item Marker

The list-style-image property specifies an image as the list item marker:

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

Position The List Item Markers

The list-style-position property specifies whether the list-item markers should appear inside or outside the content flow:

Example

```
ul {
    list-style-position: inside;
}
```

Remove Default Settings

The list-style-type:none property can also be used to remove the markers/bullets. Note that the list also has default margin and padding. To remove this, add margin:0 and padding:0 to <ul> or <ol>:

Example

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

List - Shorthand property

The list-style property is a shorthand property. It is used to set all the list properties in one declaration:

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

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

- list-style-type (if a list-style-image is specified, the value of this property will be displayed if the image for some reason cannot be displayed)
- list-style-position (specifies whether the list-item markers should appear inside or outside the content flow)
- list-style-image (specifies an image as the list item marker)

If one of the property values above are missing, the default value for the missing property will be inserted, if any.

Styling List With Colors

We can also style lists with colors, to make them look a little more interesting.

Anything added to the <ol> or <ul> tag, affects the entire list, while properties added to the <li> tag will affect the individual list items:

Example

```
ol {
    background: #ff9999;
    padding: 20px;
}

ul {
    background: #3399ff;
    padding: 20px;
}

ol li {
    background: #ffe5e5;
    padding: 5px;
    margin-left: 35px;
}

ul li {
    background: #cce5ff;
    margin: 5px;
}
```

Result:

1. Coffee
2. Tea
3. Coca Cola

- Coffee
- Tea
- Coca Cola

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

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

Company	Contact	Country
Alfreds Futterkiste	Maria Anders	Germany
Berglunds snabbköp	Christina Berglund	Sweden

Centro comercial Moctezuma	Francisco Chang	Mexico
Ernst Handel	Roland Mendel	Austria
Island Trading	Helen Bennett	UK
Königlich Essen	Philip Cramer	Germany
Laughing Bacchus Winecellars	Yoshi Tannamuri	Canada
Magazzini Alimentari Riuniti	Giovanni Rovelli	Italy

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> and <td> elements have separate borders.

Collapse Table Borders

The border-collapse property sets whether the table borders should be collapsed into a single border:

Example

```
table {
   border-collapse: collapse;
}

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

If you only want a border around the table, only specify the border property for <table>:

Example

```
table {
   border: 1px solid black;
}
```

Table Width and Height

Width and height of a table are 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;
}
```

Horizontal Alignment

The text-align property sets the horizontal alignment (like left, right, or center) of the content in <th> or <td>.

By default, the content of <th> elements are center-aligned and the content of <td> elements are left-aligned.

The following example left-aligns the text in <th> elements:

Example

```
th {
    text-align: left;
}
```

Vertical Alignment

The vertical-align property sets the vertical alignment (like top, bottom, or middle) of the content in <th> or <td>.

By default, the vertical alignment of the content in a table is middle (for both <th> and <td> elements).

The following example sets the vertical text alignment to bottom for <td> elements:

Example

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

Table Padding

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

Example

```
th, td {
    padding: 15px;
    text-align: left;
}
```

Horizontal Dividers

First Name	Last Name	Savings
Peter	Griffin	$100
Lois	Griffin	$150
Joe	Swanson	$300

Add the border-bottom property to <th> and <td> for horizontal dividers:

Example

```
th, td {
    border-bottom: 1px solid #ddd;
}
```

Hoverable Table

Use the :hover selector on <tr> to highlight table rows on mouse over:

First Name	Last Name	Savings

Peter	Griffin	$100
Lois	Griffin	$150
Joe	Swanson	$300

Example

```
tr:hover {background-color: #f5f5f5}
```

Striped Tables

First Name	**Last Name**	**Savings**
Peter	Griffin	$100
Lois	Griffin	$150
Joe	Swanson	$300

For zebra-striped tables, use the nth-child() selector and add a background-color to all even (or odd) table rows:

Example

```
tr:nth-child(even) {background-color: #f2f2f2}
```

Table Color

The example below specifies the background color and text color of <th> elements:

First Name	Last Name	Savings
Peter	Griffin	$100
Lois	Griffin	$150
Joe	Swanson	$300

Example

```
th {
    background-color: #4CAF50;
    color: white;
}
```

Responsive Table

A responsive table will display a horizontal scroll bar if the screen is too small to display the full content:

First Name	Last Name	Points	Points	Points	Points	Points	Points	Points	Points	Poin
Jill	Smith	50	50	50	50	50	50	50	50	50
Eve	Jackson	94	94	94	94	94	94	94	94	94
Adam	Johnson	67	67	67	67	67	67	67	67	67

Add a container element (like <div>) with overflow-x:auto around the <table> element to make it responsive:

Example

```
<div style="overflow-x:auto;">

<table>
... table content ...
</table>

</div>
```

CSS Table Properties

Property	Description
border	Sets all the border properties in one declaration
border-collapse	Specifies whether or not table borders should be collapsed
border-spacing	Specifies the distance between the borders of adjacent cells
caption-side	Specifies the placement of a table caption
empty-cells	Specifies whether or not to display borders and background on empty cells in a table
table-layout	Sets the layout algorithm to be used for a table

CSS Layout - The display Property

The display property is the most important CSS property for controlling layout.

The display Property

The display property specifies if/how an element is displayed.

Every HTML element has a default display value depending on what type of element it is. The default display value for most elements is block or inline.

Click to show panel

Block-level Elements

A block-level element always starts on a new line and takes up the full width available (stretches out to the left and right as far as it can).

The <div> element is a block-level element.

Examples of block-level elements:

- <div>
- <h1> - <h6>
- <p>
- <form>
- <header>
- <footer>
- <section>

Inline Elements

An inline element does not start on a new line and only takes up as much width as necessary.

This is an inline <span> element inside a paragraph.

Examples of inline elements:

- <span>
- <a>
- <img>

Display: none;

display: none; is commonly used with JavaScript to hide and show elements without deleting and recreating them. Take a look at our last example on this page if you want to know how this can be achieved.

The <script> element uses display: none; as default.

Override The Default Display Value

As mentioned, every element has a default display value. However, you can override this.

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 the web standards.

A common example is making inline <li> elements for horizontal menus:

Example

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

Note: Setting the display property of an element only changes **how the element is displayed**, NOT what kind of element it is. So, an inline element with display: block; is not allowed to have other block elements inside it.

The following example displays <span> elements as block elements:

Example

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

The following example displays <a> elements as block elements:

Example

```
a {
  display: block;
}
```

Hide an Element - display:none or visibility:hidden?

display:none

Remove

visibility:hidden

Hide

Reset

Reset All

Hiding an element can be done by setting the display property to none. The element will be hidden, and the page will be displayed as if the element is not there:

Example

```
h1.hidden {
   display: none;
}
```

visibility:hidden; also hides an element.

However, the element will still take up the same space as before. The element will be hidden, but still affect the layout:

Example

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

CSS Display/Visibility Properties

Property	**Description**
display	Specifies how an element should be displayed
visibility	Specifies whether or not an element should be visible

Using width, max-width and margin: auto;

Setting the width of a block-level element will prevent it from stretching out to the edges of its container. Then, you can set the margins to auto, to horizontally center the element within its container. The element will take up the specified width, and the remaining space will be split equally between the two margins:

This <div> element has a width of 500px, and margin set to auto.

Note: The problem with the <div> above occurs when the browser window is smaller than the width of the element. The browser then adds a horizontal scrollbar to the page.

Using max-width instead, in this situation, will improve the browser's handling of small windows. This is important when making a site usable on small devices:

This <div> element has a max-width of 500px, and margin set to auto.

Tip: Resize the browser window to less than 500px wide, to see the difference between the two divs!

Here is an example of the two divs above:

Example

```
div.ex1 {
   width: 500px;
   margin: auto;
   border: 3px solid #73AD21;
}

div.ex2 {
   max-width: 500px;
   margin: auto;
   border: 3px solid #73AD21;
}
```

Using width, max-width and margin: auto;

As mentioned in the previous chapter; a block-level element always takes up the full width available (stretches out to the left and right as far as it can).

Setting the width of a block-level element will prevent it from stretching out to the edges of its container. Then, you can set the margins to auto, to horizontally center the element within its container. The element will take up the specified width, and the remaining space will be split equally between the two margins:

This <div> element has a width of 500px, and margin set to auto.

Note: The problem with the <div> above occurs when the browser window is smaller than the width of the element. The browser then adds a horizontal scrollbar to the page.

Using max-width instead, in this situation, will improve the browser's handling of small windows. This is important when making a site usable on small devices:

This <div> element has a max-width of 500px, and margin set to auto.

Tip: Resize the browser window to less than 500px wide, to see the difference between the two divs!

Here is an example of the two divs above:

```
div.ex1 {
   width: 500px;
   margin: auto;
   border: 3px solid #73AD21;
}

div.ex2 {
   max-width: 500px;
   margin: auto;
   border: 3px solid #73AD21;
}
```

CSS Layout - The position Property

The position property specifies the type of positioning method used for an element (static, relative, fixed or absolute).

The position Property

The position property specifies the type of positioning method used for an element.

There are four different position values:

- static
- relative
- fixed
- absolute

Elements are then 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 position value.

position: static;

HTML elements are positioned static by default.

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

An element with position: static; is not positioned in any special way; it is always positioned according to the normal flow of the page:

This <div> element has position: static;

Here is the CSS that is used:

Example

```
div.static {
   position: static;
   border: 3px solid #73AD21;
}
```

position: relative;

An element with position: relative; is positioned relative to its normal position.

Setting the top, right, bottom, and left properties of a relatively-positioned element will cause it to be adjusted away from its normal position. Other content will not be adjusted to fit into any gap left by the element.

This <div> element has position: relative;

Here is the CSS that is used:

Example

```
div.relative {
   position: relative;
   left: 30px;
   border: 3px solid #73AD21;
}
```

position: fixed;

An element with position: fixed; is positioned relative to the viewport, which means it always stays in the same place even if the page is scrolled. The top, right, bottom, and left properties are used to position the element.

A fixed element does not leave a gap in the page where it would normally have been located.

Notice the fixed element in the lower-right corner of the page. Here is the CSS that is used:

Example

```
div.fixed {
   position: fixed;
   bottom: 0;
   right: 0;
   width: 300px;
   border: 3px solid #73AD21;
}
```

This <div> element has position: fixed;

position: absolute;

An element with position: absolute; is positioned relative to the nearest positioned ancestor (instead of positioned relative to the viewport, like fixed).

However; if an absolute positioned element has no positioned ancestors, it uses the document body, and moves along with page scrolling.

Note: A "positioned" element is one whose position is anything except static.

Here is a simple example:

This <div> element has position: relative;
This <div> element has position: absolute;

Here is the CSS that is used:

Example

```
div.relative {
   position: relative;
   width: 400px;
   height: 200px;
   border: 3px solid #73AD21;
}

div.absolute {
   position: absolute;
   top: 80px;
   right: 0;
   width: 200px;
   height: 100px;
   border: 3px solid #73AD21;
}
```

Overlapping Elements

When elements are positioned, 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:

This is a heading

Because the image has a z-index of -1, it will be placed behind the text.

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.

All CSS Positioning Properties

Property	Description
bottom	Sets the bottom margin edge for a positioned box
clip	Clips an absolutely positioned element
cursor	Specifies the type of cursor to be displayed
left	Sets the left margin edge for a positioned box
overflow	Specifies what happens if content overflows an element's box

overflow-x	Specifies what to do with the left/right edges of the content if it overflows the element's content area
overflow-y	Specifies what to do with the top/bottom edges of the content if it overflows the element's content area
position	Specifies the type of positioning for an element
right	Sets the right margin edge for a positioned box
top	Sets the top margin edge for a positioned box
z-index	Sets the stack order of an element

CSS Overflow

The CSS overflow property specifies whether to clip content or to add scrollbars when the content of an element is too big to fit in a specified area.

This text is really long and the height of its container is only 100 pixels. Therefore, a scrollbar is added to help the reader to scroll the content. Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi. Nam liber tempor cum soluta nobis eleifend option congue nihil imperdiet doming id quod mazim placerat facer possim assum. Typi non habent claritatem insitam; est usus legentis in iis qui facit eorum claritatem.

The overflow property has the following values:

- visible - Default. The overflow is not clipped. It renders outside the element's box
- hidden - The overflow is clipped, and the rest of the content will be invisible
- scroll - The overflow is clipped, but a scrollbar is added to see the rest of the content
- auto - If overflow is clipped, a scrollbar should be added to see the rest of the content

Note: The overflow property only works for block elements with a specified height.

Note: In OS X Lion (on Mac), scrollbars are hidden by default and only shown when being used (even though "overflow:scroll" is set).

Visible

By default, the overflow is visible, meaning that it is not clipped and it renders outside the element's box:

You can use the overflow property when you want to have better control of the layout. The overflow property specifies what happens if content overflows an element's box.

Example

```
div {
    width: 200px;
    height: 50px;
    background-color: #eee;
    overflow: visible;
}
```

Hidden

With the hidden value, the overflow is clipped, and the rest of the content is hidden:

You can use the overflow property when you want to have better control of the layout. The overflow property specifies what happens if content overflows an element's box.

Example

```
div {
    overflow: hidden;
}
```

Scroll

Setting the value to scroll, the overflow is clipped and a scrollbar is added to scroll inside the box. Note that this will add a scrollbar both horizontally and vertically (even if you do not need it):

You can use the overflow property when you want to have better control of the layout. The overflow property specifies what happens if content overflows an element's box.

Example

```
div {
    overflow: scroll;
}
```

Auto

The auto value is similar to scroll, only it add scrollbars when necessary:

You can use the overflow property when you want to have better control of the layout. The overflow property specifies what happens if content overflows an element's box.

Example

```
div {
    overflow: auto;
}
```

overflow-x and overflow-y

The overflow-x and overflow-y properties specifies whether to change the overflow of content just horizontally or vertically (or both):

overflow-x specifies what to do with the left/right edges of the content.
overflow-y specifies what to do with the top/bottom edges of the content.

You can use the overflow property when you want to have better control of the layout. The overflow property specifies what happens if content overflows an element's box.

Example

```
div {
   overflow-x: hidden; /* Hide horizontal scrollbar */
   overflow-y: scroll; /* Add vertical scrollbar */
}
```

CSS Layout - float and clear

The float property specifies whether or not an element should float.

The clear property is used to control the behavior of floating elements.

The float Property

In its simplest use, the float property can be used to wrap text around images.

The following example specifies that an image should float to the right in a text:

Example

```
img {
   float: right;
   margin: 0 0 10px 10px;
}
```

The clear Property

The clear property is used to control the behavior of floating elements.

Elements after a floating element will flow around it. To avoid this, use the clearproperty.

The clear property specifies on which sides of an element floating elements are not allowed to float:

Example

```
div {
   clear: left;
}
```

The clearfix Hack

If an element is taller than the element containing it, and it is floated, it will overflow outside of its container.

Then we can add overflow: auto; to the containing element to fix this problem:

Example

```
.clearfix {
   overflow: auto;
}
```

The overflow:auto clearfix works well as long as you are able to keep control of your margins and padding (else you might see scrollbars). The **new, modern clearfix hack** however, is safer to use, and the following code is used for most webpages:

Example

```
.clearfix::after {
  content: "";
  clear: both;
  display: table;
}
```

You will learn more about the ::after pseduo-element in a later chapter.

Web Layout Example

It is common to do entire web layouts using the float property:

Example

```
.header, .footer {
  background-color: grey;
  color: white;
  padding: 15px;
}

.column {
  float: left;
  padding: 15px;
}

.clearfix::after {
  content: "";
  clear: both;
  display: table;
}

.menu {
  width: 25%;
}

.content {
  width: 75%;
}
```

All CSS Float Properties

Property	Description
clear	Specifies on which sides of an element where floating elements are not allowed to float

float	Specifies whether or not an element should float
overflow	Specifies what happens if content overflows an element's box
overflow-x	Specifies what to do with the left/right edges of the content if it overflows the element's content area
overflow-y	Specifies what to do with the top/bottom edges of the content if it overflows the element's content area

CSS Layout - inline-block

The inline-block Value

It has been possible for a long time to create a grid of boxes that fills the browser width and wraps nicely (when the browser is resized), by using the floatproperty.

However, the inline-block value of the display property makes this even easier.

inline-block elements are like inline elements but they can have a width and a height.

Examples

The old way - using float (notice that we also need to specify a clear property for the element after the floating boxes):

Example

```
.floating-box {
    float: left;
    width: 150px;
    height: 75px;
    margin: 10px;
    border: 3px solid #73AD21;
}

.after-box {
    clear: left;
}
```

The same effect can be achieved by using the inline-block value of the display property (notice that no clear property is needed):

Example

```
.floating-box {
    display: inline-block;
    width: 150px;
    height: 75px;
    margin: 10px;
    border: 3px solid #73AD21;
}
```

CSS Layout - Horizontal & Vertical Align

◀ ▶

Center elements
horizontally and vertically

Center Align Elements

To horizontally center a block element (like <div>), use margin: auto;

Setting the width of the element will prevent it from stretching out to the edges of its container.

The element will then take up the specified width, and the remaining space will be split equally between the two margins:

This div element is centered.

Example

```
.center {
    margin: auto;
    width: 50%;
    border: 3px solid green;
    padding: 10px;
}
```

Note: Center aligning has no effect if the width property is not set (or set to 100%).

Center Align Text

To just center the text inside an element, use text-align: center;

This text is centered.

Example

```
.center {
    text-align: center;
    border: 3px solid green;
}
```

Tip: For more examples on how to align text, see the CSS Text chapter.

Center an Image

To center an image, use margin: auto; and make it into a **block** element:

Example

```
img {
    display: block;
    margin: auto;
    width: 40%;
}
```

Left and Right Align - Using position

One method for aligning elements is to use position: absolute;:

In my younger and more vulnerable years my father gave me some advice that I've been turning over in my mind ever since.

Example

```
.right {
    position: absolute;
    right: 0px;
    width: 300px;
    border: 3px solid #73AD21;
    padding: 10px;
}
```

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

Tip: When aligning elements with position, always define margin and paddingfor the <body> element. This is to avoid visual differences in different browsers.

There is also a problem with IE8 and earlier, when using position. 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. So, always set the !DOCTYPE declaration when using position:

Example

```
body {
    margin: 0;
    padding: 0;
}

.container {
    position: relative;
    width: 100%;
}

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

Left and Right Align - Using float

Another method for aligning elements is to use the float property:

Example

```
.right {
    float: right;
    width: 300px;
    border: 3px solid #73AD21;
    padding: 10px;
}
```

Tip: When aligning elements with float, always define margin and padding for the <body> element. This is to avoid visual differences in different browsers.

There is also a problem with IE8 and earlier, when using float. 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. So, always set the !DOCTYPE declaration when using float:

Example

```
body {
    margin: 0;
    padding: 0;
}

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

Center Vertically - Using padding

There are many ways to center an element vertically in CSS. A simple solution is to use top and bottom padding:

I am vertically centered.

Example

```
.center {
    padding: 70px 0;
    border: 3px solid green;
}
```

To center both vertically and horizontally, use `padding` and `text-align: center`:

I am vertically and horizontally centered.

Example

```
.center {
    padding: 70px 0;
    border: 3px solid green;
    text-align: center;
}
```

Center Vertically - Using line-height

Another trick is to use the `line-height` property with a value that is equal to the `height` property.

I am vertically and horizontally centered.

Example

```
.center {
    line-height: 200px;
    height: 200px;
    border: 3px solid green;
    text-align: center;
}

/* If the text has multiple lines, add the following: */
.center p {
    line-height: 1.5;
    display: inline-block;
    vertical-align: middle;
}
```

Center Vertically - Using position & transform

If `padding` and `line-height` is not an option, a third solution is to use positioning and the `transform` property:

I am vertically and horizontally centered.

```
.center {
    height: 200px;
    position: relative;
    border: 3px solid green;
}

.center p {
    margin: 0;
    position: absolute;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
}
```

CSS Combinators

A combinator is something that explains the relationship between the selectors.

A CSS selector can contain more than one simple selector. Between the simple selectors, we can include a combinator.

There are four different combinators in CSS3:

- descendant selector (space)
- child selector (>)
- adjacent sibling selector (+)
- general sibling selector (~)

Descendant Selector

The descendant selector matches all elements that are descendants of a specified element.

The following example selects all <p> elements inside <div> elements:

Example

```
div p {
    background-color: yellow;
}
```

Child Selector

The child selector selects all elements that are the immediate children of a specified element.

The following example selects all <p> elements that are immediate children of a <div> element:

Example

```
div > p {
    background-color: yellow;
}
```

Adjacent Sibling Selector

The adjacent sibling selector selects all elements that are the adjacent siblings of a specified element.

Sibling elements must have the same parent element, and "adjacent" means "immediately following".

The following example selects all <p> elements that are placed immediately after <div> elements:

Example

```
div + p {
    background-color: yellow;
}
```

General Sibling Selector

The general sibling selector selects all elements that are siblings of a specified element.

The following example selects all <p> elements that are siblings of <div> elements:

Example

```
div ~ p {
   background-color: yellow;
}
```

CSS Pseudo-classes

What are Pseudo-classes?

A pseudo-class is used to define a special state of an element.

For example, it can be used to:

- Style an element when a user mouses over it
- Style visited and unvisited links differently
- Style an element when it gets focus

Mouse Over Me

The syntax of pseudo-classes:

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

Anchor Pseudo-classes

Links can be displayed in different ways:

Example

```
/* unvisited link */
a:link {
   color: #FF0000;
}

/* visited link */
a:visited {
   color: #00FF00;
}

/* mouse over link */
a:hover {
  color: #FF00FF;
}

/* selected link */
a:active {
  color: #0000FF;
}
```

Note: a:hover MUST come after a:link and a:visited in the CSS definition in order to be effective! a:active MUST come after a:hover in the CSS definition in order to be effective! Pseudo-class names are not case-sensitive.

Pseudo-classes and CSS Classes

Pseudo-classes can be combined with CSS classes:

When you hover over the link in the example, it will change color:

Example

```
a.highlight:hover {
  color: #ff0000;
}
```

Hover on <div>

An example of using the :hover pseudo-class on a <div> element:

Example

```
div:hover {
  background-color: blue;
}
```

Simple Tooltip Hover

Hover over a <div> element to show a <p> element (like a tooltip):

Hover over me to show the <p> element.

Example

```
p {
  display: none;
  background-color: yellow;
  padding: 20px;
}

div:hover p {
  display: block;
}
```

CSS - The :first-child Pseudo-class

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

Match the first <p> element

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

Example

```
p:first-child {
   color: blue;
}
```

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

```
p i:first-child {
   color: blue;
}
```

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

```
p:first-child i {
   color: blue;
}
```

CSS - The :lang Pseudo-class

The :lang pseudo-class allows you to define special rules for different languages.

In the example below, :lang defines the quotation marks for <q> elements with lang="no":

Example

```
<html>
<head>
<style>
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

Selector	Example	Example description
:active	a:active	Selects the active link
:checked	input:checked	Selects every checked <input> element
:disabled	input:disabled	Selects every disabled <input> element
:empty	p:empty	Selects every <p> element that has no children
:enabled	input:enabled	Selects every enabled <input> element
:first-child	p:first-child	Selects every <p> elements that is the first child of its parent
:first-of-type	p:first-of-type	Selects every <p> element that is the first <p> element of its parent
:focus	input:focus	Selects the <input> element that has focus
:hover	a:hover	Selects links on mouse over
:in-range	input:in-range	Selects <input> elements with a value within a

		specified range
:invalid	input:invalid	Selects all <input> elements with an invalid value
:lang(*language*)	p:lang(it)	Selects every <p> element with a lang attribute value starting with "it"
:last-child	p:last-child	Selects every <p> elements that is the last child of its parent
:last-of-type	p:last-of-type	Selects every <p> element that is the last <p> element of its parent
:link	a:link	Selects all unvisited links
:not(selector)	:not(p)	Selects every element that is not a <p> element
:nth-child(n)	p:nth-child(2)	Selects every <p> element that is the second child of its parent
:nth-last-child(n)	p:nth-last-child(2)	Selects every <p> element that is the second child of its parent, counting from the last child
:nth-last-of-type(n)	p:nth-last-of-type(2)	Selects every <p> element that is the second <p> element of its parent, counting from the last child
:nth-of-type(n)	p:nth-of-type(2)	Selects every <p> element that is the second <p> element of its parent
:only-of-type	p:only-of-type	Selects every <p> element that is the only <p>

		element of its parent
:only-child	p:only-child	Selects every <p> element that is the only child of its parent
:optional	input:optional	Selects <input> elements with no "required" attribute
:out-of-range	input:out-of-range	Selects <input> elements with a value outside a specified range
:read-only	input:read-only	Selects <input> elements with a "readonly" attribute specified
:read-write	input:read-write	Selects <input> elements with no "readonly" attribute
:required	input:required	Selects <input> elements with a "required" attribute specified
:root	root	Selects the document's root element
:target	#news:target	Selects the current active #news element (clicked on a URL containing that anchor name)
:valid	input:valid	Selects all <input> elements with a valid value
:visited	a:visited	Selects all visited links

All CSS Pseudo Elements

Selector	Example	Example description
::after	p::after	Insert content after every <p> element
::before	p::before	Insert content before every <p> element
::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
::selection	p::selection	Selects the portion of an element that is selected by a user

CSS Pseudo-elements

What are Pseudo-Elements?

A CSS pseudo-element is used to style specified parts of an element.

For example, it can be used to:

- Style the first letter, or line, of an element
- Insert content before, or after, the content of an element

The syntax of pseudo-elements:

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

Notice the double colon notation - ::first-line versus :first-line

The double colon replaced the single-colon notation for pseudo-elements in CSS3. This was an attempt from W3C to distinguish between **pseudo-classes** and **pseudo-elements**.

The single-colon syntax was used for both pseudo-classes and pseudo-elements in CSS2 and CSS1.

For backward compatibility, the single-colon syntax is acceptable for CSS2 and CSS1 pseudo-elements.

The ::first-line Pseudo-element

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

The following example formats the first line of the text in all <p> elements:

Example

```
p::first-line {
   color: #ff0000;
   font-variant: small-caps;
}
```

Note: The ::first-line pseudo-element can only be applied to block-level elements.

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.

The following example formats the first letter of the text in all <p> elements:

Example

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

Note: The ::first-letter pseudo-element can only be applied to block-level elements.

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:

Example

```
p.intro::first-letter {
    color: #ff0000;
    font-size:200%;
}
```

The example above will display the first letter of paragraphs with class="intro", in red and in a larger size.

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 the content of 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 the content of each <h1> element:

Example

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

CSS - The ::selection Pseudo-element

The ::selection pseudo-element matches the portion of an element that is selected by a user.

The following CSS properties can be applied to ::selection: color,background, cursor, and outline.

The following example makes the selected text red on a yellow background:

Example

```
::selection {
   color: red;
   background: yellow;
}
```

All CSS Pseudo Elements

Selector	Example	Example description
::after	p::after	Insert something after the content of each <p> element
::before	p::before	Insert something before the content of each <p> element
::first-letter	p::first-letter	Selects the first letter of each <p> element
::first-line	p::first-line	Selects the first line of each <p> element
::selection	p::selection	Selects the portion of an element that is selected by a user

All CSS Pseudo Classes

Selector	Example	Example description
:active	a:active	Selects the active link
:checked	input:checked	Selects every checked <input> element

:disabled	input:disabled	Selects every disabled <input> element
:empty	p:empty	Selects every <p> element that has no children
:enabled	input:enabled	Selects every enabled <input> element
:first-child	p:first-child	Selects every <p> elements that is the first child of its parent
:first-of-type	p:first-of-type	Selects every <p> element that is the first <p> element of its parent
:focus	input:focus	Selects the <input> element that has focus
:hover	a:hover	Selects links on mouse over
:in-range	input:in-range	Selects <input> elements with a value within a specified range
:invalid	input:invalid	Selects all <input> elements with an invalid value
:lang(*language*)	p:lang(it)	Selects every <p> element with a lang attribute value starting with "it"
:last-child	p:last-child	Selects every <p> elements that is the last child of its parent
:last-of-type	p:last-of-type	Selects every <p> element that is the last <p> element of its parent

:link	a:link	Selects all unvisited links
:not(selector)	:not(p)	Selects every element that is not a <p> element
:nth-child(n)	p:nth-child(2)	Selects every <p> element that is the second child of its parent
:nth-last-child(n)	p:nth-last-child(2)	Selects every <p> element that is the second child of its parent, counting from the last child
:nth-last-of-type(n)	p:nth-last-of-type(2)	Selects every <p> element that is the second <p> element of its parent, counting from the last child
:nth-of-type(n)	p:nth-of-type(2)	Selects every <p> element that is the second <p> element of its parent
:only-of-type	p:only-of-type	Selects every <p> element that is the only <p> element of its parent
:only-child	p:only-child	Selects every <p> element that is the only child of its parent
:optional	input:optional	Selects <input> elements with no "required" attribute
:out-of-range	input:out-of-range	Selects <input> elements with a value outside a specified range
:read-only	input:read-only	Selects <input> elements with a "readonly" attribute specified

:read-write	input:read-write	Selects <input> elements with no "readonly" attribute
:required	input:required	Selects <input> elements with a "required" attribute specified
:root	root	Selects the document's root element
:target	#news:target	Selects the current active #news element (clicked on a URL containing that anchor name)
:valid	input:valid	Selects all <input> elements with a valid value
:visited	a:visited	Selects all visited links

CSS Opacity / Transparency

The opacity property specifies the opacity/transparency of an element.

Transparent Image

The opacity property can take a value from 0.0 - 1.0. The lower value, the more transparent:

opacity 0.2

opacity 0.5

opacity 1
(default)

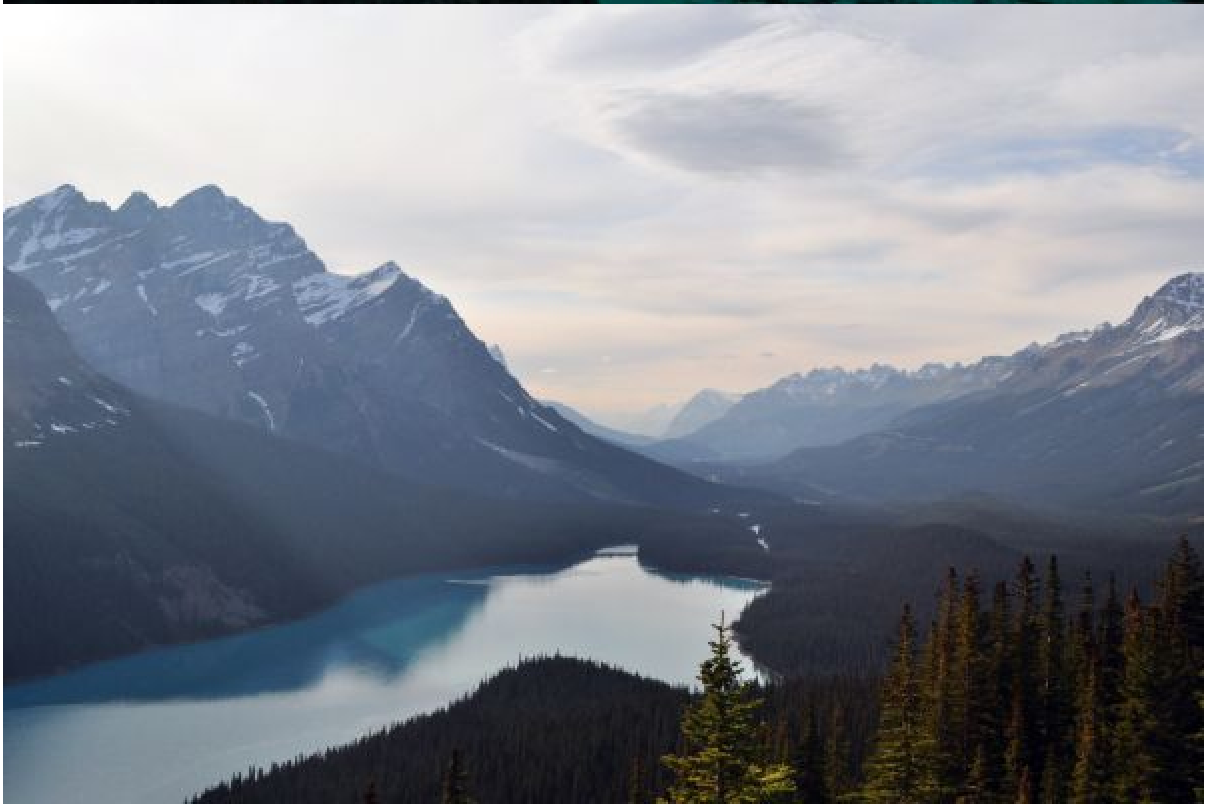
Note: IE8 and earlier use filter:alpha(opacity=x). The x can take a value from 0 - 100. A lower value makes the element more transparent.

Example

```
img {
  opacity: 0.5;
  filter: alpha(opacity=50); /* For IE8 and earlier */
}
```

Transparent Hover Effect

The opacity property is often used together with the :hover selector to change the opacity on mouse-over:



Example

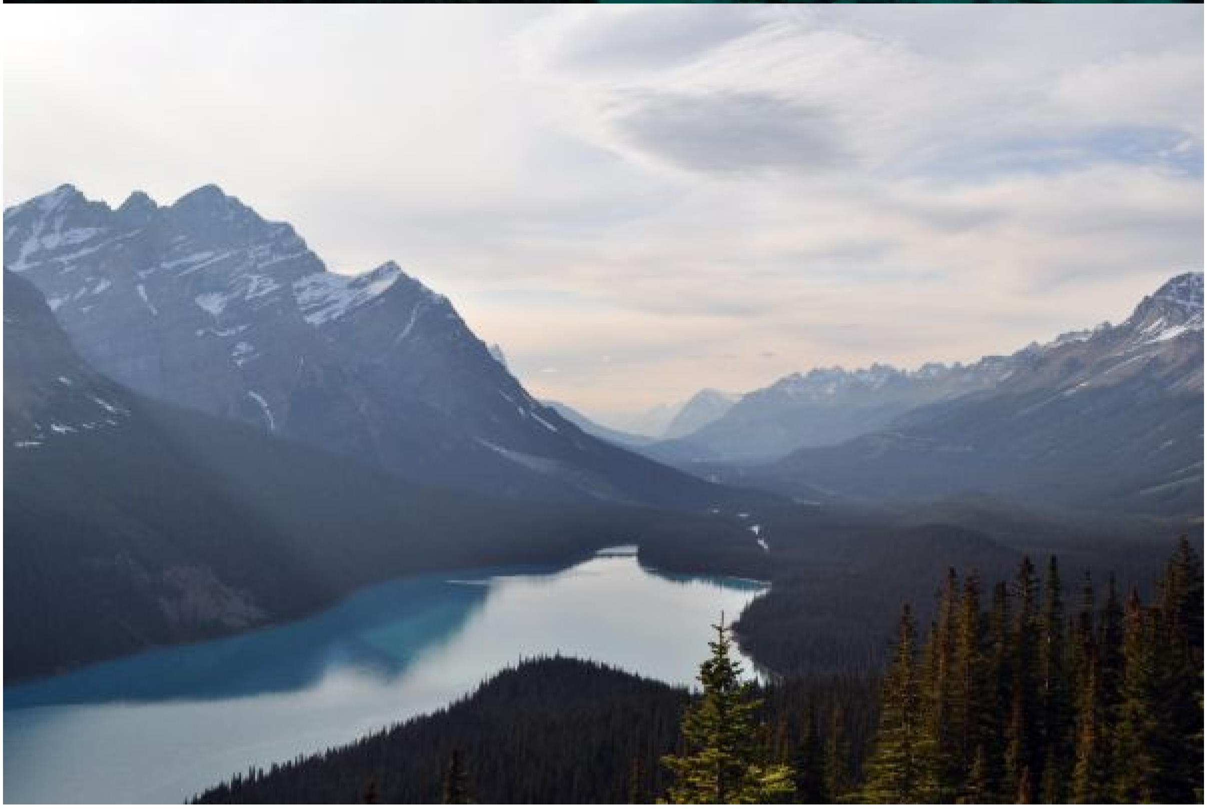
```
img {
   opacity: 0.5;
   filter: alpha(opacity=50); /* For IE8 and earlier */
}

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

Example explained

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

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

An example of reversed hover effect:



Example

```
img:hover {
    opacity: 0.5;
    filter: alpha(opacity=50); /* For IE8 and earlier */
}
```

Transparent Box

When using the opacity property to add transparency to the background of an element, all of its child elements become transparent as well. This can make the text inside a fully transparent element hard to read:

opacity 1

opacity 0.6

opacity 0.3

opacity 0.1

Example

```
div {
    opacity: 0.3;
    filter: alpha(opacity=30); /* For IE8 and earlier */
}
```

Transparency using RGBA

If you do not want to apply opacity to child elements, like in our example above, use **RGBA** color values. The following example sets the opacity for the background color and not the text:

100% opacity

60% opacity

30% opacity

10% opacity

You learned from our CSS Colors Chapter, that you can use RGB as a color value. In addition to RGB, CSS3 introduced an RGB color value with an alpha channel (RGBA) - which specifies the opacity for a color.

An RGBA color value is specified with: rgba(red, green, blue, *alpha*). The *alpha* parameter is a number between 0.0 (fully transparent) and 1.0 (fully opaque).

Tip: You will learn more about RGBA Colors in our CSS3 Colors Chapter.

Example

```
div {
   background: rgba(76, 175, 80, 0.3) /* Green background with 30% opacity */
}
```

Text in Transparent Box

This is some text that is placed in the transparent box.

```
<html>
<head>
<style>
div.background {
   background: url(klematis.jpg) repeat;
   border: 2px solid black;
}

div.transbox {
   margin: 30px;
   background-color: #ffffff;
   border: 1px solid black;
   opacity: 0.6;
   filter: alpha(opacity=60); /* For IE8 and earlier */
}

div.transbox p {
   margin: 5%;
   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.</p>
  </div>
```

```
</div>

</body>
</html>
```

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

CSS Navigation Bar

Demo: Navigation Bars

Vertical

- Home
- News
- Contact
- About

Horizontal

- Home
- News
- Contact
- About

- Home
- News
- 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
- Set margin: 0; and padding: 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, you can style the <a> elements inside the list, in addition to the code above:

Example

```
li 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 (and padding, margin, height, etc. if you want)
- width: 60px; - Block elements take up the full width available by default. We want to specify a 60 pixels width

You can also set the width of <ul>, and remove the width of <a>, as they will take up the full width available when displayed as block elements. This will produce the same result as our previous example:

Example

```
ul {
    list-style-type: none;
    margin: 0;
    padding: 0;
    width: 60px;
}

li a {
    display: block;
}
```

Vertical Navigation Bar Examples

Create a basic vertical navigation bar with a gray background color and change the background color of the links when the user moves the mouse over them:

- Home
- News
- Contact
- About

Example

```
ul {
    list-style-type: none;
    margin: 0;
    padding: 0;
    width: 200px;
    background-color: #f1f1f1;
}

li a {
    display: block;
    color: #000;
    padding: 8px 16px;
    text-decoration: none;
}

/* Change the link color on hover */
li a:hover {
    background-color: #555;
    color: white;
}
```

Active/Current Navigation Link

Add an "active" class to the current link to let the user know which page he/she is on:

- Home
- News
- Contact
- About

Example

```
.active {
    background-color: #4CAF50;
    color: white;
}
```

Center Links & Add Borders

Add `text-align:center` to <li> or <a> to center the links.

Add the `border` property to <ul> add a border around the navbar. If you also want borders inside the navbar, add a `border-bottom` to all <li> elements, except for the last one:

- Home
- News

- Contact
- About

Example

```
ul {
    border: 1px solid #555;
}

li {
    text-align: center;
    border-bottom: 1px solid #555;
}

li:last-child {
    border-bottom: none;
}
```

Full-height Fixed Vertical Navbar

Create a full-height, "sticky" side navigation:

Example

```
ul {
    list-style-type: none;
    margin: 0;
    padding: 0;
    width: 25%;
    background-color: #f1f1f1;
    height: 100%; /* Full height */
    position: fixed; /* Make it stick, even on scroll */
    overflow: auto; /* Enable scrolling if the sidenav has too much content */
}
```

Note: This example might not work properly on mobile devices.

Horizontal Navigation Bar

There are two ways to create a horizontal navigation bar. Using **inline** or **floating** list items.

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

Floating List Items

Another way of creating a horizontal navigation bar is to float the <li> elements, and specify a layout for the navigation links:

Example

```
li {
   float: left;
}

a {
   display: block;
   padding: 8px;
   background-color: #dddddd;
}
```

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 padding (and height, width, margins, etc. if you want)
- padding: 8px; - Since block elements take up the full width available, they cannot float next to each other. Therefore, specify some padding to make them look good
- background-color: #dddddd; - Add a gray background-color to each a element

Tip: Add the background-color to <ul> instead of each <a> element if you want a full-width background color:

Example

```
ul {
   background-color: #dddddd;
}
```

Horizontal Navigation Bar Examples

Create a basic horizontal navigation bar with a dark background color and change the background color of the links when the user moves the mouse over them:

- Home
- News
- Contact
- About

```
ul {
   list-style-type: none;
   margin: 0;
   padding: 0;
   overflow: hidden;
   background-color: #333;
}
```

```
li {
    float: left;
}

li a {
    display: block;
    color: white;
    text-align: center;
    padding: 14px 16px;
    text-decoration: none;
}

/* Change the link color to #111 (black) on hover */
li a:hover {
    background-color: #111;
}
```

Active/Current Navigation Link

Add an "active" class to the current link to let the user know which page he/she is on:

- Home
- News
- Contact
- About

Example

```
.active {
    background-color: #4CAF50;
}
```

Right-Align Links

Right-align links by floating the list items to the right (float:right;):

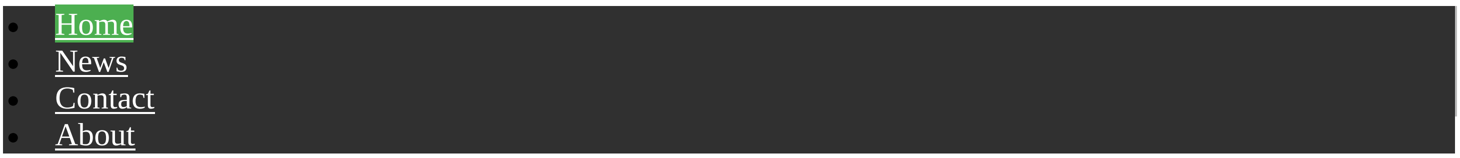
- Home
- News
- Contact
- About

Example

```
<ul>
  <li><a href="#home">Home</a></li>
  <li><a href="#news">News</a></li>
  <li><a href="#contact">Contact</a></li>
  <li style="float:right"><a class="active" href="#about">About</a></li>
</ul>
```

Border Dividers

Add the border-right property to <li> to create link dividers:



- Home
- News
- Contact
- About

Example

```
/* Add a gray right border to all list items, except the last item (last-child) */
li {
   border-right: 1px solid #bbb;
}

li:last-child {
   border-right: none;
}
```

Fixed Navigation Bar

Make the navigation bar stay at the top or the bottom of the page, even when the user scrolls the page:

Fixed Top

```
ul {
   position: fixed;
   top: 0;
   width: 100%;
}
```

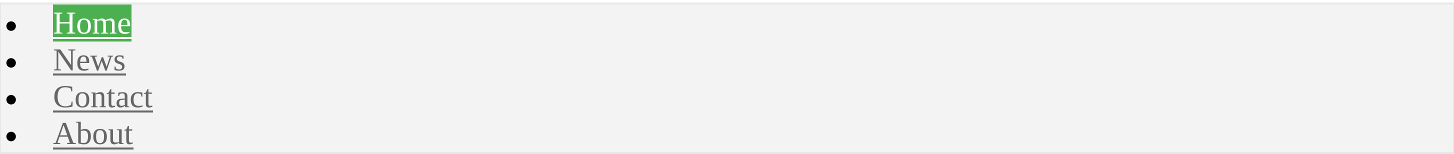
Fixed Bottom

```
ul {
   position: fixed;
   bottom: 0;
   width: 100%;
}
```

Note: These examples might not work properly on mobile devices.

Gray Horizontal Navbar

An example of a gray horizontal navigation bar with a thin gray border:



- Home
- News
- Contact
- About

Example

```
ul {
   border: 1px solid #e7e7e7;
   background-color: #f3f3f3;
}

li a {
   color: #666;
```

}

CSS Dropdowns

Create a hoverable dropdown with CSS.

Demo: Dropdown Examples

Move the mouse over the examples below:

Dropdown Text
Dropdown Menu

Other:

Basic Dropdown

Create a dropdown box that appears when the user moves the mouse over an element.

Example

```
<style>
.dropdown {
   position: relative;
   display: inline-block;
}

.dropdown-content {
   display: none;
   position: absolute;
   background-color: #f9f9f9;
   min-width: 160px;
   box-shadow: 0px 8px 16px 0px rgba(0,0,0,0.2);
   padding: 12px 16px;
   z-index: 1;
}

.dropdown:hover .dropdown-content {
   display: block;
}
</style>

<div class="dropdown">
  <span>Mouse over me</span>
  <div class="dropdown-content">
    <p>Hello World!</p>
  </div>
</div>
```

HTML) Use any element to open the dropdown content, e.g. a <span>, or a <button> element.

Use a container element (like <div>) to create the dropdown content and add whatever you want inside of it.

Wrap a <div> element around the elements to position the dropdown content correctly with CSS.

CSS) The .dropdown class use position:relative, which is needed when we want the dropdown content to be placed right below the dropdown button (using position:absolute).

The .dropdown-content class holds the actual dropdown content. It is hidden by default, and will be displayed on hover (see below). Note the min-width is set to 160px. Feel free to change this. **Tip:** If you want the width of the dropdown content to be as wide as the dropdown button, set the width to 100% (and overflow:auto to enable scroll on small screens).

Instead of using a border, we have used the CSS3 box-shadow property to make the dropdown menu look like a "card".

The :hover selector is used to show the dropdown menu when the user moves the mouse over the dropdown button.

Dropdown Menu

Create a dropdown menu that allows the user to choose an option from a list:

Dropdown Menu

This example is similar to the previous one, except that we add links inside the dropdown box and style them to fit a styled dropdown button:

Example

```
<style>
/* Style The Dropdown Button */
.dropbtn {
   background-color: #4CAF50;
   color: white;
   padding: 16px;
   font-size: 16px;
   border: none;
   cursor: pointer;
}

/* The container <div> - needed to position the dropdown content */
.dropdown {
   position: relative;
   display: inline-block;
}

/* Dropdown Content (Hidden by Default) */
.dropdown-content {
   display: none;
   position: absolute;
   background-color: #f9f9f9;
   min-width: 160px;
   box-shadow: 0px 8px 16px 0px rgba(0,0,0,0.2);
   z-index: 1;
}

/* Links inside the dropdown */
```

```
.dropdown-content a {
   color: black;
   padding: 12px 16px;
   text-decoration: none;
   display: block;
}

/* Change color of dropdown links on hover */
.dropdown-content a:hover {background-color: #f1f1f1}

/* Show the dropdown menu on hover */
.dropdown:hover .dropdown-content {
   display: block;
}

/* Change the background color of the dropdown button when the dropdown content is shown */
.dropdown:hover .dropbtn {
   background-color: #3e8e41;
}
</style>

<div class="dropdown">
  <button class="dropbtn">Dropdown</button>
  <div class="dropdown-content">
    <a href="#">Link 1</a>
    <a href="#">Link 2</a>
    <a href="#">Link 3</a>
  </div>
</div>
```

Right-aligned Dropdown Content

Left
Right

If you want the dropdown menu to go from right to left, instead of left to right, add right: 0;

Example

```
.dropdown-content {
   right: 0;
}
```

CSS Tooltip

Create tooltips with CSS.

Demo: Tooltip Examples

A tooltip is often used to specify extra information about something when the user moves the mouse pointer over an element:

Top
Right
Bottom
Left

Basic Tooltip

Create a tooltip that appears when the user moves the mouse over an element:

Example

```
<style>
/* Tooltip container */
.tooltip {
   position: relative;
   display: inline-block;
   border-bottom: 1px dotted black; /* If you want dots under the hoverable text */
}

/* Tooltip text */
.tooltip .tooltiptext {
   visibility: hidden;
   width: 120px;
   background-color: black;
   color: #fff;
   text-align: center;
   padding: 5px 0;
   border-radius: 6px;

   /* Position the tooltip text - see examples below! */
   position: absolute;
   z-index: 1;
}

/* Show the tooltip text when you mouse over the tooltip container */
.tooltip:hover .tooltiptext {
   visibility: visible;
}
</style>

<div class="tooltip">Hover over me
  <span class="tooltiptext">Tooltip text</span>
</div>
```

Example Explained

HTML) Use a container element (like <div>) and add the "tooltip" class to it. When the user mouse over this <div>, it will show the tooltip text.

The tooltip text is placed inside an inline element (like <span>) with class="tooltiptext".

CSS) The tooltip class use position:relative, which is needed to position the tooltip text (position:absolute). **Note:** See examples below on how to position the tooltip.

The tooltiptext class holds the actual tooltip text. It is hidden by default, and will be visible on hover (see below). We have also added some basic styles to it: 120px width, black background color, white text color, centered text, and 5px top and bottom padding.

The CSS3 border-radius property is used to add rounded corners to the tooltip text.

The :hover selector is used to show the tooltip text when the user moves the mouse over the <div> with class="tooltip".

Positioning Tooltips

In this example, the tooltip is placed to the right (left:105%) of the "hoverable" text (<div>). Also note that top:-5px is used to place it in the middle of its container element. We use the number **5** because the tooltip text has a top and bottom padding of 5px. If you increase its padding, also increase the value of the top property to ensure that it stays in the middle (if this is something you want). The same applies if you want the tooltip placed to the left.

Right Tooltip

```
.tooltip .tooltiptext {
    top: -5px;
    left: 105%;
}
```

Result:

Hover over me

Left Tooltip

```
.tooltip .tooltiptext {
    top: -5px;
    right: 105%;
}
```

Result:

Hover over me

If you want the tooltip to appear on top or on the bottom, see examples below. Note that we use the margin-left property with a value of minus 60 pixels. This is to center the tooltip above/below the hoverable text. It is set to the half of the tooltip's width (120/2 = 60).

Top Tooltip

```
.tooltip .tooltiptext {
    width: 120px;
    bottom: 100%;
    left: 50%;
    margin-left: -60px; /* Use half of the width (120/2 = 60), to center the tooltip */
}
```

Result:

Hover over me

Bottom Tooltip

```
.tooltip .tooltiptext {
    width: 120px;
    top: 100%;
    left: 50%;
    margin-left: -60px; /* Use half of the width (120/2 = 60), to center the tooltip */
}
```

Result:

Hover over me

Tooltip Arrows

To create an arrow that should appear from a specific side of the tooltip, add "empty" content after tooltip, with the pseudo-element class ::after together with the content property. The arrow itself is created using borders. This will make the tooltip look like a speech bubble.

This example demonstrates how to add an arrow to the bottom of the tooltip:

Bottom Arrow

```
.tooltip .tooltiptext::after {
    content: " ";
    position: absolute;
    top: 100%; /* At the bottom of the tooltip */
    left: 50%;
    margin-left: -5px;
    border-width: 5px;
    border-style: solid;
    border-color: black transparent transparent transparent;
}
```

Result:

Hover over me

Position the arrow inside the tooltip: top: 100% will place the arrow at the bottom of the tooltip. left: 50% will center the arrow.

Note: The border-width property specifies the size of the arrow. If you change this, also change the margin-left value to the same. This will keep the arrow centered.

The border-color is used to transform the content into an arrow. We set the top border to black, and the rest to transparent. If all sides were black, you would end up with a black square box.

This example demonstrates how to add an arrow to the top of the tooltip. Notice that we set the bottom border color this time:

Top Arrow

```
.tooltip .tooltiptext::after {
    content: " ";
    position: absolute;
    bottom: 100%;  /* At the top of the tooltip */
    left: 50%;
    margin-left: -5px;
    border-width: 5px;
    border-style: solid;
    border-color: transparent transparent black transparent;
}
```

Result:

Hover over me

This example demonstrates how to add an arrow to the left of the tooltip:

Left Arrow

```
.tooltip .tooltiptext::after {
    content: " ";
    position: absolute;
    top: 50%;
    right: 100%; /* To the left of the tooltip */
    margin-top: -5px;
    border-width: 5px;
    border-style: solid;
    border-color: transparent black transparent transparent;
}
```

Result:

Hover over me

This example demonstrates how to add an arrow to the right of the tooltip:

Right Arrow

```
.tooltip .tooltiptext::after {
    content: " ";
    position: absolute;
    top: 50%;
    left: 100%; /* To the right of the tooltip */
    margin-top: -5px;
    border-width: 5px;
    border-style: solid;
    border-color: transparent transparent transparent black;
}
```

Result:

Hover over me

Fade In Tooltips (Animation)

If you want to fade in the tooltip text when it is about to be visible, you can use the CSS3 transition property together with the opacity property, and go from being completely invisible to 100% visible, in a number of specified seconds (1 second in our example):

Example

```
.tooltip .tooltiptext {
    opacity: 0;
    transition: opacity 1s;
}
```

```
.tooltip:hover .tooltiptext {
    opacity: 1;
}
```

Image Gallery

The following image gallery is created with CSS:

Example

```
<html>
<head>
<style>
div.gallery {
    margin: 5px;
    border: 1px solid #ccc;
    float: left;
    width: 180px;
}

div.gallery:hover {
    border: 1px solid #777;
}

div.gallery img {
    width: 100%;
    height: auto;
}

div.desc {
    padding: 15px;
    text-align: center;
}
</style>
</head>
<body>

<div class="gallery">
  <a target="_blank" href="fjords.jpg">
    <img src="fjords.jpg" alt="Fjords" width="300" height="200">
  </a>
  <div class="desc">Add a description of the image here</div>
</div>

<div class="gallery">
  <a target="_blank" href="forest.jpg">
    <img src="forest.jpg" alt="Forest" width="300" height="200">
  </a>
  <div class="desc">Add a description of the image here</div>
</div>

<div class="gallery">
  <a target="_blank" href="lights.jpg">
    <img src="lights.jpg" alt="Northern Lights" width="300" height="200">
  </a>
  <div class="desc">Add a description of the image here</div>
</div>

<div class="gallery">
  <a target="_blank" href="mountains.jpg">
    <img src="mountains.jpg" alt="Mountains" width="300" height="200">
  </a>
  <div class="desc">Add a description of the image here</div>
</div>

</body>
</html>
```

CSS Image Sprites

Image Sprites

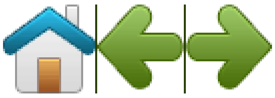
An image sprite is a collection of images put into a single image.

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

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

Example explained:

- <img id="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;
   background: url('img_navsprites.gif') 0 0;
}

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

#next {
   left: 129px;
   width: 43px;
   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)

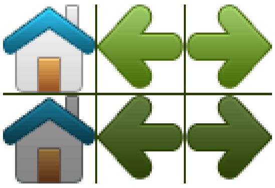
- #prev {left:63px;width:43px;} - Positioned 63px to the right (#home width 46px + some extra space between items), and the width is 43px.
- #prev {background:url('img_navsprites.gif') -47px 0;} - Defines the background image 47px to the right (#home width 46px + 1px line divider)
- #next {left:129px;width:43px;}- 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') -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.

Tip: The :hover selector can be used on all elements, not only on links.

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:

- #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

Style HTML Elements With Specific Attributes

It is possible to style HTML elements that have specific attributes or attribute values.

CSS [attribute] Selector

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;
}
```

The example above will match elements with title="flower", title="summer flower", and title="flower new", but not title="my-flower" or title="flowers".

CSS [attribute|="value"] Selector

The [attribute|="value"] selector is used to select elements with the specified attribute starting with the specified value.

The following example selects all elements with a class attribute value that begins with "top":

Note: The value has to be a whole word, either alone, like class="top", or followed by a hyphen(-), like class="top-text"!

Example

```
[class|="top"] {
   background: yellow;
}
```

CSS [attribute^="value"] Selector

The [attribute^="value"] selector is used to select elements whose attribute value begins with a specified value.

The following example selects all elements with a class attribute value that begins with "top":

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

Example

```
[class^="top"] {
   background: yellow;
}
```

CSS [attribute$="value"] Selector

The [attribute$="value"] selector is used to select elements whose attribute value ends with a specified value.

The following example selects all elements with a class attribute value that ends with "test":

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

Example

```
[class$="test"] {
   background: 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;
}
```

Styling Forms

The attribute selectors can be 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;
}
```

Tip: Visit our CSS Forms Tutorial for more examples on how to style forms with CSS.

CSS Forms

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

First Name Last Name Country Try it Yourself »

Styling Input Fields

Use the width property to determine the width of the input field:

First Name

Example

```
input {
    width: 100%;
}
```

The example above applies to all <input> elements. If you only want to style a specific input type, you can use attribute selectors:

- input[type=text] - will only select text fields
- input[type=password] - will only select password fields
- input[type=number] - will only select number fields
- etc..

Padded Inputs

Use the padding property to add space inside the text field.

Tip: When you have many inputs after each other, you might also want to add some margin, to add more space outside of them:

First Name Last Name

Example

```
input[type=text] {
    width: 100%;
    padding: 12px 20px;
    margin: 8px 0;
    box-sizing: border-box;
}
```

Note that we have set the box-sizing property to border-box. This makes sure that the padding and eventually borders are included in the total width and height of the elements.
Read more about the box-sizing property in our CSS3 Box Sizing chapter.

Bordered Inputs

Use the border property to change the border size and color, and use the border-radius property to add rounded corners:

First Name

Example

```
input[type=text] {
   border: 2px solid red;
   border-radius: 4px;
}
```

If you only want a bottom border, use the border-bottom property:

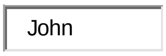
First Name

Example

```
input[type=text] {
   border: none;
   border-bottom: 2px solid red;
}
```

Colored Inputs

Use the background-color property to add a background color to the input, and the color property to change the text color:



Example

```
input[type=text] {
   background-color: #3CBC8D;
   color: white;
}
```

Focused Inputs

By default, some browsers will add a blue outline around the input when it gets focus (clicked on). You can remove this behavior by adding outline: none; to the input.

Use the :focus selector to do something with the input field when it gets focus:

Example

```
input[type=text]:focus {
   background-color: lightblue;
}
```

Example

```
input[type=text]:focus {
    border: 3px solid #555;
}
```

Input with icon/image

If you want an icon inside the input, use the background-image property and position it with the background-position property. Also notice that we add a large left padding to reserve the space of the icon:

Example

```
input[type=text] {
    background-color: white;
    background-image: url('searchicon.png');
    background-position: 10px 10px;
    background-repeat: no-repeat;
    padding-left: 40px;
}
```

Animated Search Input

In this example we use the CSS3 transition property to animate the width of the search input when it gets focus. You will learn more about the transitionproperty later, in our CSS3 Transitions chapter.

Example

```
input[type=text] {
    -webkit-transition: width 0.4s ease-in-out;
    transition: width 0.4s ease-in-out;
}

input[type=text]:focus {
    width: 100%;
}
```

Styling Text areas

Tip: Use the resize property to prevent textareas from being resized (disable the "grabber" in the bottom right corner):

Example

```
textarea {
    width: 100%;
    height: 150px;
    padding: 12px 20px;
    box-sizing: border-box;
    border: 2px solid #ccc;
    border-radius: 4px;
    background-color: #f8f8f8;
    resize: none;
}
```

Styling Select Menus

Example

```
select {
    width: 100%;
    padding: 16px 20px;
    border: none;
    border-radius: 4px;
    background-color: #f1f1f1;
}
```

Styling Input Buttons

Example

```
input[type=button], input[type=submit], input[type=reset] {
    background-color: #4CAF50;
    border: none;
    color: white;
    padding: 16px 32px;
    text-decoration: none;
    margin: 4px 2px;
    cursor: pointer;
}

/* Tip: use width: 100% for full-width buttons */
```

CSS Counters

Pizza

Hamburger

Hotdogs

CSS counters are "variables" maintained by CSS whose values can be incremented by CSS rules (to track how many times they are used). Counters let you adjust the appearance of content based on its placement in the document.

Automatic Numbering With Counters

CSS counters are like "variables". The variable values can be incremented by CSS rules (which will track how many times they are used).

To work with CSS counters we will use the following properties:

- counter-reset - Creates or resets a counter
- counter-increment - Increments a counter value
- content - Inserts generated content
- counter() or counters() function - Adds the value of a counter to an element

To use a CSS counter, it must first be created with counter-reset.

The following example creates a counter for the page (in the body selector), then increments the counter value for each <h2> element and adds "Section <*value of the counter*>:" to the beginning of each <h2> element:

Example

```
body {
   counter-reset: section;
}

h2::before {
   counter-increment: section;
   content: "Section " counter(section) ": ";
}
```

Nesting Counters

The following example creates one counter for the page (section) and one counter for each <h1> element (subsection). The "section" counter will be counted for each <h1> element with "Section <*value of the section counter*>.", and the "subsection" counter will be counted for each <h2> element with "<*value of the section counter*>.<*value of the subsection counter*>":

Example

```
body {
   counter-reset: section;
}

h1 {
   counter-reset: subsection;
}

h1::before {
   counter-increment: section;
   content: "Section " counter(section) ". ";
}

h2::before {
   counter-increment: subsection;
   content: counter(section) "." counter(subsection) " ";
}
```

A counter can also be useful to make outlined lists because a new instance of a counter is automatically created in child elements. Here we use the counters()function to insert a string between different levels of nested counters:

Example

```
ol {
   counter-reset: section;
   list-style-type: none;
}

li::before {
   counter-increment: section;
   content: counters(section,".") " ";
}
```

CSS Counter Properties

Property	Description
content	Used with the ::before and ::after pseudo-elements, to insert generated content
counter-increment	Increments one or more counter values
counter-reset	