# CSS Introduction

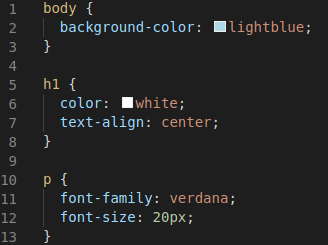
## What is CSS?

* CSS stands for 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 stylesheets are stored in CSS files

## Why Use CSS?

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 Example



## 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 color 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.

CSS removed the style formatting from the HTML page!

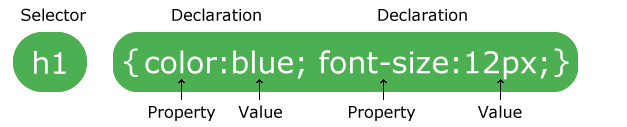
## CSS Saves a Lot of Work!

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.

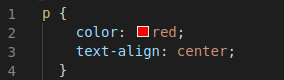
The declaration block contains one or more declarations separated by semicolons.

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

Multiple CSS declarations are separated with semicolons, and declaration blocks are surrounded by curly braces.

### Example

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



#### Example Explained

* p is a selector in CSS (it points to the HTML element you want to style: <p>).
* color is a property, and red is the property value
* text-align is a property, and center is the property value

## CSS Selectors

CSS selectors are used to "find" (or select) the HTML elements you want to style.

We can divide CSS selectors into five categories:

* Simple selectors (select elements based on name, id, class)
* Combinator selectors (select elements based on a specific relationship between them)
* Pseudo-class selectors (select elements based on a certain state)
* Pseudo-elements selectors (select and style a part of an element)
* Attribute selectors (select elements based on an attribute or attribute value)

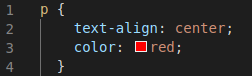
This page will explain the most basic CSS selectors.

## The CSS element Selector

The element selector selects HTML elements based on the element name.

### Example

Here, all <p> elements on the page will be center-aligned, with a red text color:



The CSS id Selector

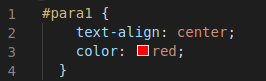
The id selector uses the id attribute of an HTML element to select a specific element.

The id of an element is 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.

### Example

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



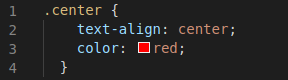
## The CSS class Selector

The class selector selects HTML elements with a specific class attribute.

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

### Example

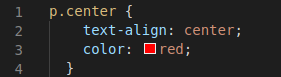
In this example all HTML elements with class="center" will be red and center-aligned:



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

### Example

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



HTML elements can also refer to more than one class.

### Example

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

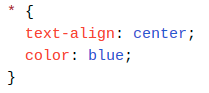
<p class="center large">This paragraph refers to two classes.</p>

## The CSS Universal Selector

The universal selector (\*) selects all HTML elements on the page.

### Example

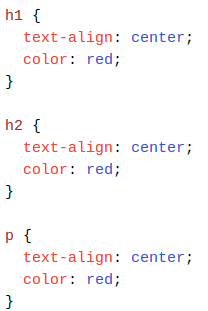
The CSS rule below will affect every HTML element on the page:



## The CSS Grouping Selector

The grouping selector selects all the HTML elements with the same style definitions.

Look at the following CSS code (the h1, h2, and p elements have the same style definitions):

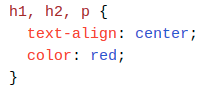


It will be better to group the selectors, to minimize the code.

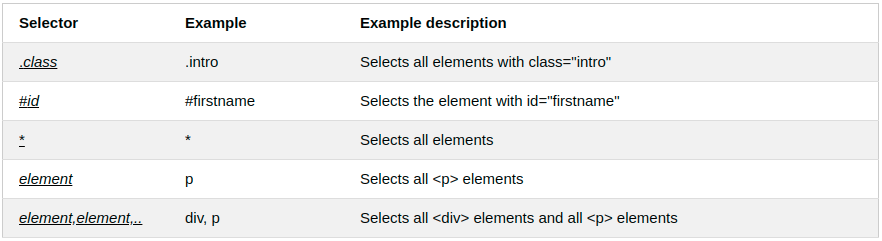
To group selectors, separate each selector with a comma.

### Example

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



## All CSS Simple Selectors



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 CSS
* Internal CSS
* Inline CSS

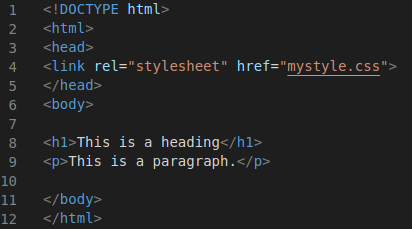
## External CSS

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

Each HTML page must include a reference to the external style sheet file inside the <link> element, inside the head section.

### Example

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

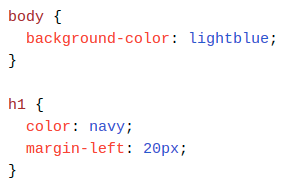


An external style sheet can be written in any text editor, and must be saved with a .css extension.

The external .css file should not contain any HTML tags.

Here is how the "mystyle.css" file looks like:

### "mystyle.css"



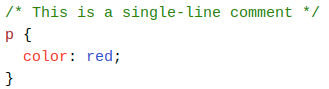
## CSS Comments

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

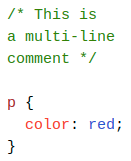
Comments are ignored by browsers.

A CSS comment is placed inside the <style> element, and starts with /\* and ends with \*/:

### Example



Comments can also span multiple lines:

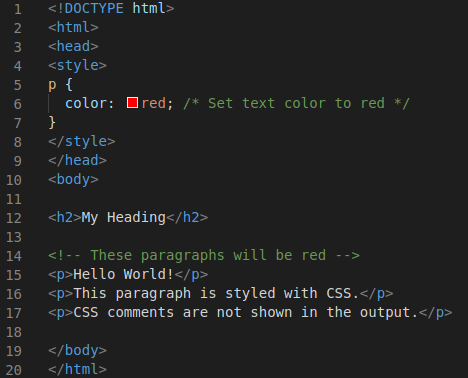


## HTML and CSS Comments

From the HTML tutorial, you learned that you can add comments to your HTML source by using the <!--...--> syntax.

In the following example, we use a combination of HTML and CSS comments:

### Example



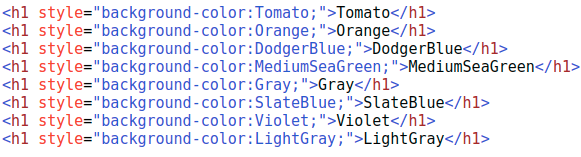
# 

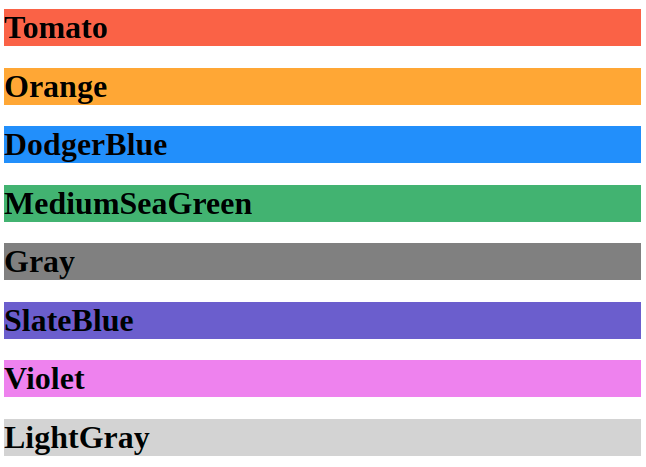
# CSS Colors

Colors are specified using predefined color names, or RGB, HEX, HSL, RGBA, HSLA values.

## CSS Color Names

In CSS, a color can be specified by using a color name:





## CSS Background Color

You can set the background color for HTML elements:

### Example

<h1 style="background-color:DodgerBlue;">Hello World</h1>

<p style="background-color:Tomato;">Lorem ipsum...</p>

## CSS Text Color

You can set the color of text:

### Example

<h1 style="color:Tomato;">Hello World</h1>

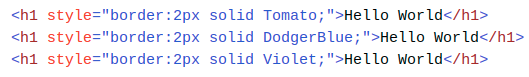
<p style="color:DodgerBlue;">Lorem ipsum...</p>

<p style="color:MediumSeaGreen;">Ut wisi enim...</p>

## CSS Border Color

You can set the color of borders:

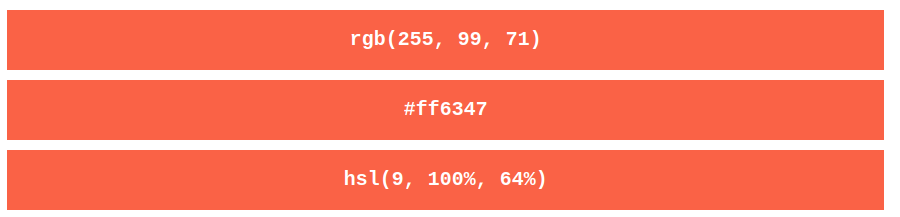
### Example



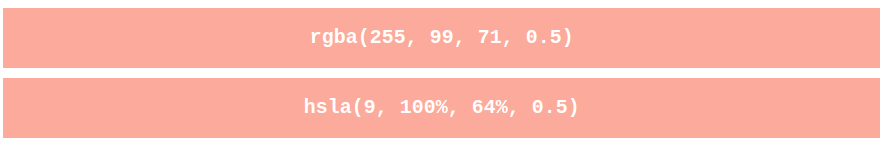
## CSS Color Values

In CSS, colors can also be specified using RGB values, HEX values, HSL values, RGBA values, and HSLA values:

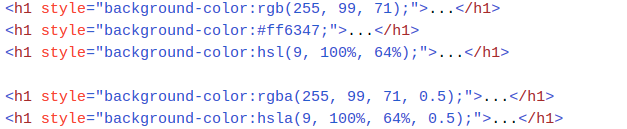
Same as color name "Tomato":



Same as color name "Tomato", but 50% transparent:



### Example



## RGB Value

In CSS, a color can be specified as an RGB value, using this formula:

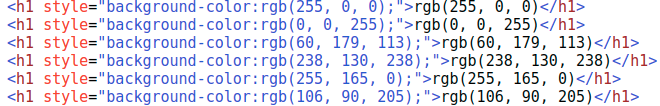
**rgb(*red,* *green*, *blue*)**

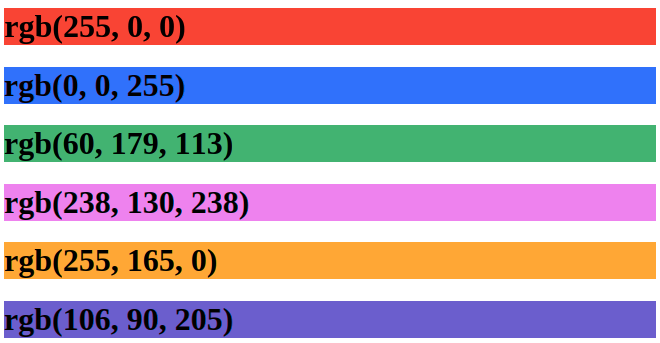
Each parameter (red, green, and blue) defines the intensity of the color between 0 and 255.

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.

To display black, set all color parameters to 0, like this: rgb(0, 0, 0).

To display white, set all color parameters to 255, like this: rgb(255, 255, 255).

****

****

## HEX Value

In CSS, a color can be specified using a hexadecimal value 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).

### Example

****



## HSL Value

In CSS, a color can be specified using hue, saturation, and lightness (HSL) in the form:

**hsl(*hue*, *saturation*, *lightness*)**

Hue is a degree on the color wheel from 0 to 360. 0 is red, 120 is green, and 240 is blue.

Saturation is a percentage value, 0% means a shade of gray, and 100% is the full color.

Lightness is also a percentage, 0% is black, 50% is neither light or dark, 100% is white

### Saturation

Saturation can be described as the intensity of a color.

100% is pure color, no shades of gray

50% is 50% gray, but you can still see the color.

0% is completely gray, you can no longer see the color.

### Lightness

The lightness of a color can be described as how much light you want to give the color, where 0% means no light (black), 50% means 50% light (neither dark nor light) 100% means full lightness (white).

Shades of gray are often defined by setting the hue and saturation to 0, and adjust the lightness from 0% to 100% to get darker/lighter shades:

## HSLA Value

HSLA color values are an extension of HSL color values with an alpha channel - which specifies the opacity for a color.

An HSLA color value is specified with:

**hsla(*hue,* *saturation*, *lightness, alpha*)**

The alpha parameter is a number between 0.0 (fully transparent) and 1.0 (not transparent at all):

## CSS background-color

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

### Example

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)"

## Other Elements

You can set the background color for any HTML elements:

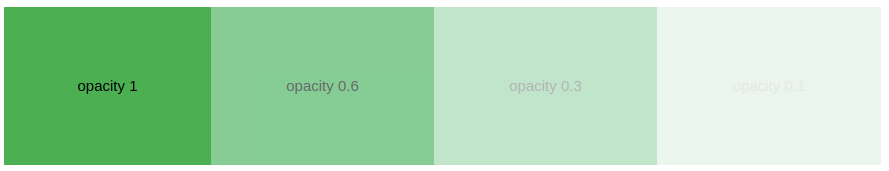
### Example

Here, the <h1>, <p>, and <div> elements will have different background colors:

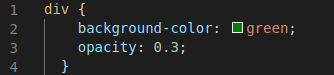


## Opacity / Transparency

The opacity property specifies the opacity/transparency of an element. It can take a value from 0.0 - 1.0. The lower value, the more transparent:



### Example



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

## 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:



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).

### Example

div {

background: rgba(0, 128, 0, 0.3) /\* Green background with 30% opacity \*/

}

## CSS 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.

### Example

The background image for a page can be set like this:

body {

background-image: url("hack.gif");

}

### Example

This example shows a bad combination of text and background image. The text is hardly readable:

body {

background-image: url("hackearth\_logo.jpg");

}

## CSS background-repeat

By default, the background-image property repeats an image both horizontally and vertically.

Some images should be repeated only horizontally or vertically, or they will look strange, like this:

### Example

body {

background-image: url("gradient\_bg.png");

}

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

### Example

body {

background-image: url("gradient\_bg.png");

background-repeat: repeat-x;

}

## CSS background-repeat: no-repeat

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

### Example

Show the background image only once:

body {

background-image: url("img\_tree.png");

background-repeat: no-repeat;

}

## CSS background-position

The background-position property is used to specify the position of the background image.

## CSS background-attachment

The background-attachment property specifies whether the background image should scroll or be fixed (will not scroll with the rest of the page):

### Example

Specify that the background image should be fixed:

body {

background-image: url("img\_tree.png");

background-repeat: no-repeat;

background-position: right top;

background-attachment: fixed;

}

### Example

Specify that the background image should scroll with the rest of the page:

body {

background-image: url("img\_tree.png");

background-repeat: no-repeat;

background-position: right top;

background-attachment: scroll;

}

## CSS 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.

Instead of writing:

body {

background-color: #ffffff;

background-image: url("img\_tree.png");

background-repeat: no-repeat;

background-position: right top;

}

You can use the shorthand property background:

### Example

Use the shorthand property to set the background properties in one declaration:

body {

background: #ffffff url("img\_tree.png") no-repeat right top;

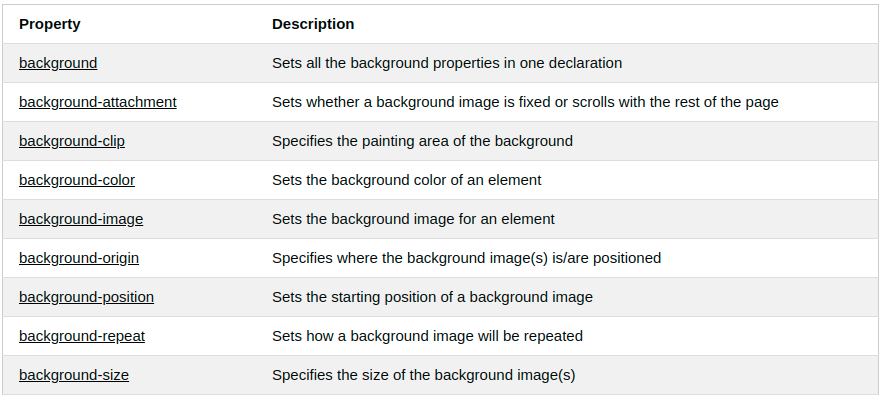
}

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. Note that we do not use the background-attachment property in the examples above, as it does not have a value.

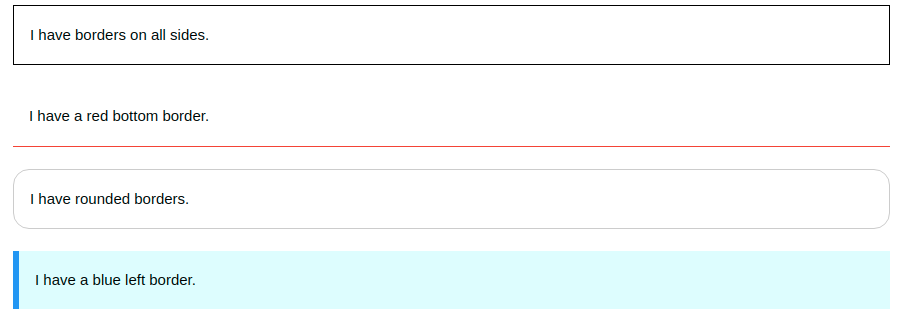
## All CSS Background Properties



## 

## CSS Border Properties

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



## CSS 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

Demonstration of the different border styles:

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;}

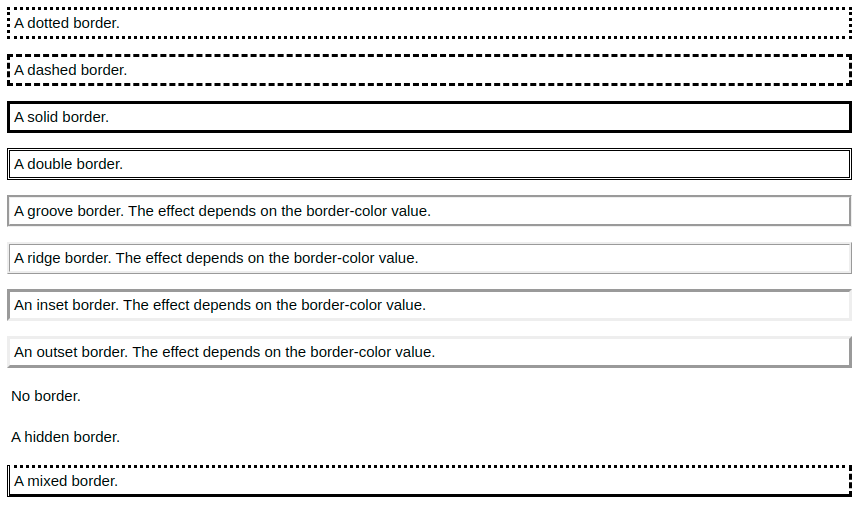
p.outset {border-style: outset;}

p.none {border-style: none;}

p.hidden {border-style: hidden;}

p.mix {border-style: dotted dashed solid double;}

Result:



Note: None of the OTHER CSS border properties (which you will learn more about in the next chapters) will have ANY effect unless the border-style property is set!

## CSS 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:

### Example

Demonstration of the different border widths:

p.one {

border-style: solid;

border-width: 5px;

}

p.two {

border-style: solid;

border-width: medium;

}

p.three {

border-style: dotted;

border-width: 2px;

}

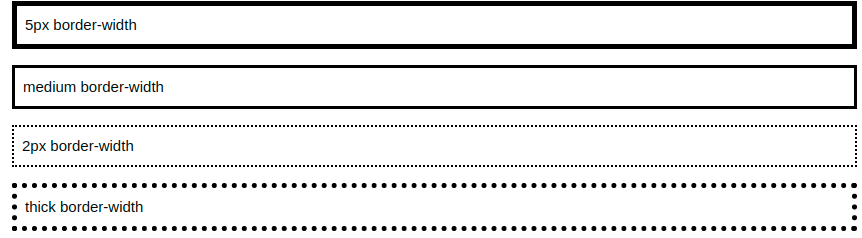
p.four {

border-style: dotted;

border-width: thick;

}

Result :



## Specific Side Widths

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

p.one {

border-style: solid;

border-width: 5px 20px; /\* 5px top and bottom, 20px on the sides \*/

}

p.two {

border-style: solid;

border-width: 20px 5px; /\* 20px top and bottom, 5px on the sides \*/

}

p.three {

border-style: solid;

border-width: 25px 10px 4px 35px; /\* 25px top, 10px right, 4px bottom and 35px left \*/

}

## CSS 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)"
* HSL - specify a HSL value, like "hsl(0, 100%, 50%)"
* transparent

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

### Example

Demonstration of the different border colors:

p.one {

border-style: solid;

border-color: red;

}

p.two {

border-style: solid;

border-color: green;

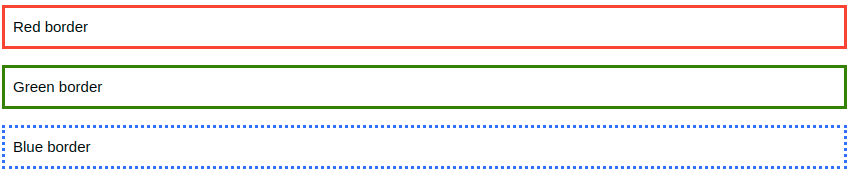
}

p.three {

border-style: dotted;

border-color: blue;

}



## Specific Side Colors

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

### Example

p.one {

border-style: solid;

border-color: red green blue yellow; /\* red top, green right, blue bottom and yellow left \*/

}

## HEX Values

The color of the border can also be specified using a hexadecimal value (HEX):

### Example

p.one {

border-style: solid;

border-color: #ff0000; /\* red \*/

}

## CSS Margins

The CSS margin properties are used to create space around elements, outside of any defined borders.

With CSS, you have full control over the margins. There are 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

### Example

Set 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

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

### Example

Use the margin shorthand property with four values:

p {

margin: 25px 50px 75px 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

### Example

Use the margin shorthand property with three values:

p {

margin: 25px 50px 75px;

}

If the margin property has two values:

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

### Example

Use the margin shorthand property with two values:

p {

margin: 25px 50px;

}

## 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.

### Example

Use margin: auto:

div {

width: 300px;

margin: auto;

border: 1px solid red;

}

## The inherit value

This example lets the left margin of the <p class="ex1"> element be inherited from the parent element (<div>):

## 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

### Example

Demonstration of margin collapse:

h1 {

margin: 0 0 50px 0;

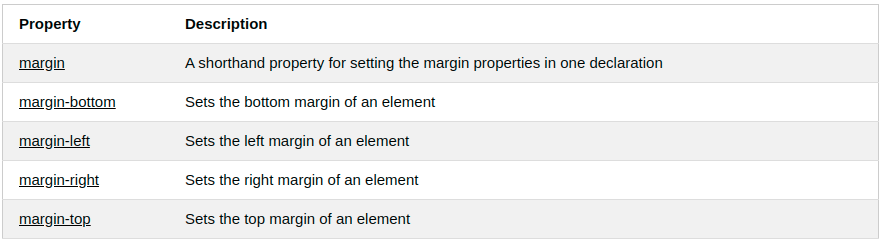
}

h2 {

margin: 20px 0 0 0;

}

## All CSS Margin Properties



## CSS Padding

The CSS padding properties are used to generate space around an element's content, inside of any defined borders.

With CSS, you have full control over the padding. There are 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

Note: Negative values are not allowed.

### Example

Set different padding for all four sides of a <div> element:

div {

padding-top: 50px;

padding-right: 30px;

padding-bottom: 50px;

padding-left: 80px;

}

## Padding and Element Width

The CSS width property specifies the width of the element's content area. The content area is the portion inside the padding, border, and margin of an element.

So, if an element has a specified width, the padding added to that element will be added to the total width of the element. This is often an undesirable result.

### Example

Here, the <div> element is given a width of 300px. However, the actual width of the <div> element will be 350px (300px + 25px of left padding + 25px of right padding):

div {

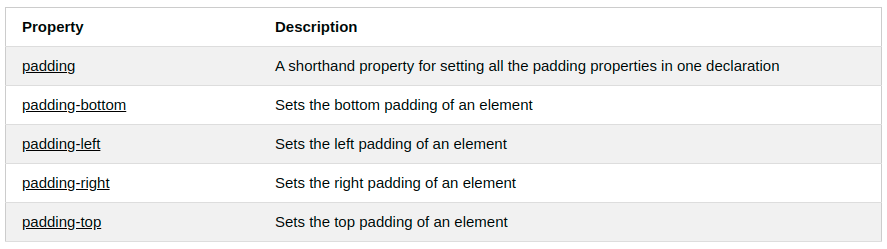
width: 300px;

padding: 25px;

}

To keep the width at 300px, no matter the amount of padding, you can use the box-sizing property. This causes the element to maintain its width; if you increase the padding, the available content space will decrease.

## All CSS Padding Properties



# 

## CSS Setting height and width

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

The height and width properties do not include padding, borders, or margins. It sets the height/width of the area inside the padding, border, and margin of the element.

## CSS height and width Values

The height and width properties may have the following values:

* auto - This is default. The browser calculates the height and width
* length - Defines the height/width in px, cm etc.
* % - Defines the height/width in percent of the containing block
* initial - Sets the height/width to its default value
* inherit - The height/width will be inherited from its parent value

### Example

Set the height and width of a <div> element:

div {

height: 200px;

width: 50%;

background-color: powderblue;

}

Note: Remember that 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

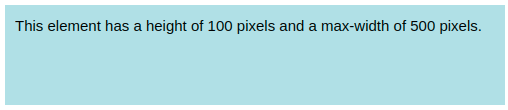
The max-width property is used to set the maximum width of an element.

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!



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

### Example

This <div> element has a height of 100 pixels and a max-width of 500 pixels:

div {

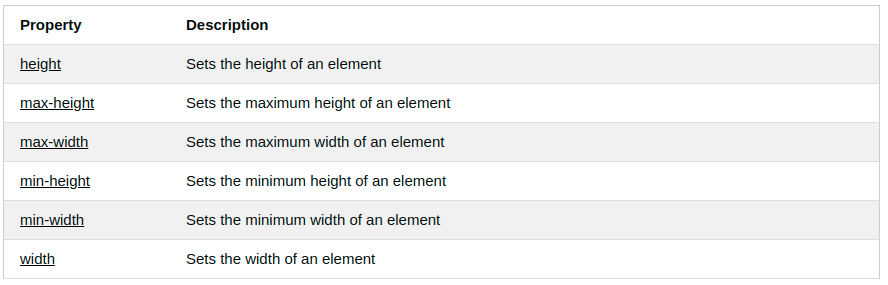
max-width: 500px;

height: 100px;

background-color: powderblue;

}

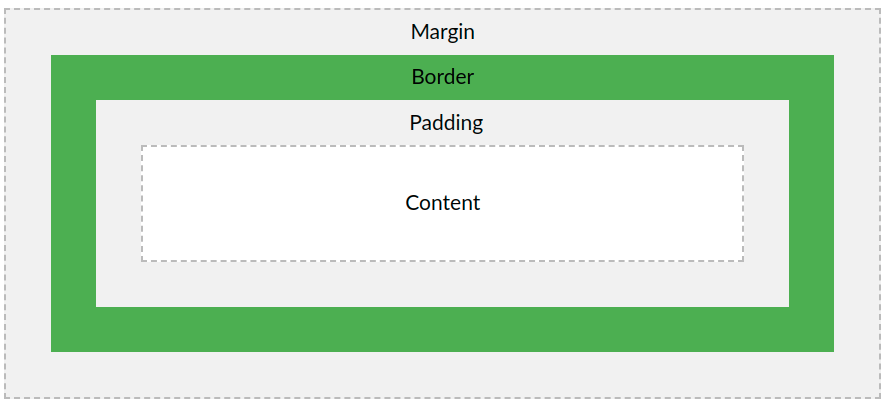
All CSS Dimension Properties



## 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.

### Example

Demonstration of the box model:

div {

width: 300px;

border: 15px solid green;

padding: 50px;

margin: 20px;

}

## 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.

### Example

This <div> element will have a total width of 350px:

div {

width: 320px;

padding: 10px;

border: 5px solid gray;

margin: 0;

}

Here is the calculation:

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:

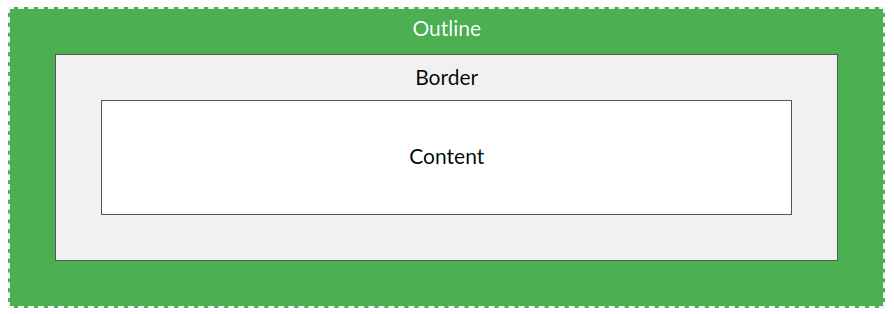
Total element width = width + left padding + right padding + left border + right border + left margin + right margin

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

## CSS Outline

An outline is a line that is drawn around elements, OUTSIDE the borders, to make the element "stand out".



CSS has the following outline properties:

* outline-style
* outline-color
* outline-width
* outline-offset
* outline

## CSS Outline Style

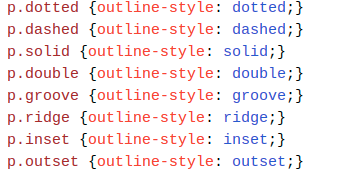
The outline-style property specifies the style of the outline, and can have one of the following values:

* 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
* ridge - Defines a 3D ridged outline
* inset - Defines a 3D inset outline
* outset - Defines a 3D outset outline
* none - Defines no outline
* hidden - Defines a hidden outline

The following example shows the different outline-style values:

### Example

Demonstration of the different outline styles:

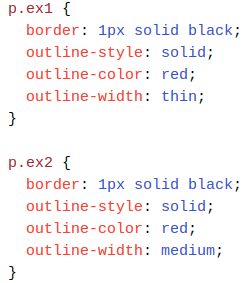


The outline-width property specifies the width of the outline, and can have one of the following values:

* thin (typically 1px)
* medium (typically 3px)
* thick (typically 5px)
* A specific size (in px, pt, cm, em, etc)

The following example shows some outlines with different widths:

### Example



### Text Color

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

### Example

body {

color: blue;

}

h1 {

color: green;

}

## Text Color and Background Color

In this example, we define both the background-color property and the color property:

### Example

body {

background-color: lightgrey;

color: blue;

}

h1 {

background-color: black;

color: white;

}

## 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):

### 

### Example

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):

### Example

div {

text-align: justify;

}

## Text Direction

The direction and unicode-bidi properties can be used to change the text direction of an element:

### Example

p {

direction: rtl;

unicode-bidi: bidi-override;

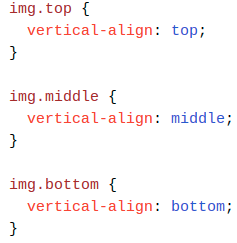
}

## Vertical Alignment

The vertical-align property sets the vertical alignment of an element.

This example demonstrates how to set the vertical alignment of an image in a text:

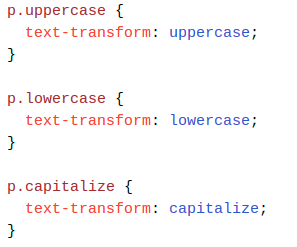
### Example



## 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:



## 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:

### Example

h1 {

letter-spacing: 3px;

}

h2 {

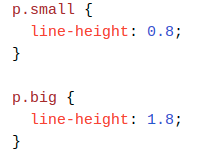
letter-spacing: -3px;

}

## Line Height

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

### Example



## 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:

### Example

h1 {

word-spacing: 10px;

}

h2 {

word-spacing: -5px;

}

## White Space

The white-space property specifies how white-space inside an element is handled.

This example demonstrates how to disable text wrapping inside an element:

### Example

p {

white-space: nowrap;

}

## Text Shadow

The text-shadow property adds shadow to text.

In its simplest use, you only specify the horizontal shadow (2px) and the vertical shadow (2px):

### Example

h1 {

text-shadow: 2px 2px;

}

Next, add a color (red) to the shadow:



# CSS Fonts

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

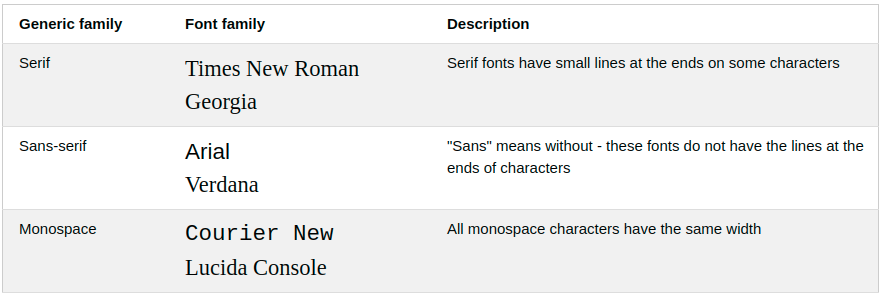
## Difference Between Serif and Sans-serif Fonts



## 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")



## 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.

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

### 

### Example

Specify the font for three paragraphs:

.serif {

font-family: "Times New Roman", Times, serif;

}

.sansserif {

font-family: Arial, Helvetica, sans-serif;

}

.monospace {

font-family: "Lucida Console", Courier, monospace;

}

## Font Style

The font-style property is mostly used to specify italic text.

This property has three values:

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

### Example

p.normal {

font-style: normal;

}

p.italic {

font-style: italic;

}

p.oblique {

font-style: oblique;

}

## Font Weight

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

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.

### Example

p.normal {

font-variant: normal;

}

p.small {

font-variant: small-caps;

}

## 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

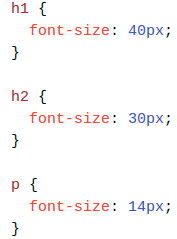
Relative size:

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

## Set Font Size With Pixels

Setting the text size with pixels gives you full control over the text size:

### Example



## Set Font Size With Em

To allow users to resize the text (in the browser menu), many developers use em instead of pixels.

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 \*/

}

## 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;

}

## 

## 

## Responsive Font Size

The text size can be set with a vw unit, which means the "viewport width".

That way the text size will follow the size of the browser window:

### Example

<h1 style="font-size:10vw">Hello World</h1>

Viewport is the browser window size. 1vw = 1% of viewport width. If the viewport is 50cm wide, 1vw is 0.5cm.

## 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 Awesome Icons

To use the Font Awesome icons, go to [fontawesome.com](https://fontawesome.com/), sign in, and get a code to add in the <head> section of your HTML page:

<script src="https://kit.fontawesome.com/*yourcode*.js"></script>

Note: No downloading or installation is required!



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!

## Styling Links

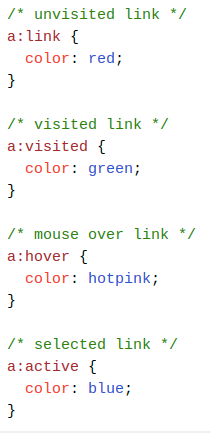
Links can be styled with any CSS property (e.g. color, font-family, background, etc.).

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



### 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;

}

## Link Buttons

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

### Example

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;

### }

## An Image as The List Item Marker

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

### Example

ul {

list-style-image: url('sqpurple.gif');

}

## Position The List Item Markers

The list-style-position property specifies the position of the list-item markers (bullet points).

"list-style-position: outside;" means that the bullet points will be outside the list item. The start of each line of a list item will be aligned vertically. This is default:

* Coffee - A brewed drink prepared from roasted coffee beans...
* Tea
* Coca-cola

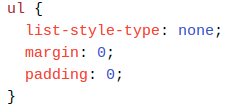
"list-style-position: inside;" means that the bullet points will be inside the list item. As it is part of the list item, it will be part of the text and push the text at the start:

* Coffee - A brewed drink prepared from roasted coffee beans...
* Tea
* Coca-cola

## 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



# CSS Layout - The position Property

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

## The position Property

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

There are five different position values:

* static
* relative
* fixed
* absolute
* sticky

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:



Here is the CSS that is used:

### Example

div.static {

position: static;

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.

## 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.

## position: sticky;

An element with position: sticky; is positioned based on the user's scroll position.

A sticky element toggles between relative and fixed, depending on the scroll position. It is positioned relative until a given offset position is met in the viewport - then it "sticks" in place (like position:fixed).

## 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:

## 

## 

## CSS Overflow

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

The overflow property has the following values:

* visible - Default. The overflow is not clipped. The content 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, and a scrollbar is added to see the rest of the content
* auto - Similar to scroll, but it adds scrollbars only when necessary

# CSS Layout - float and clear

The CSS float property specifies how an element should float.

The CSS clear property specifies what elements can float beside the cleared element and on which side.

## The float Property

The float property is used for positioning and formatting content e.g. let an image float left to the text in a container.

The float property can have one of the following values:

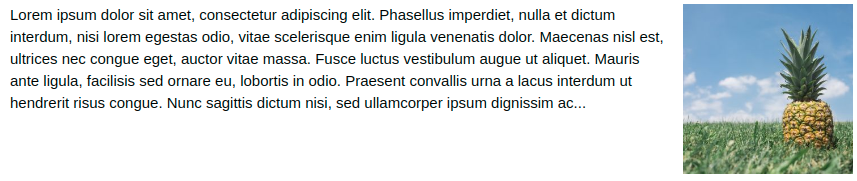
* left - The element floats to the left of its container
* right - The element floats to the right of its container
* none - The element does not float (will be displayed just where it occurs in the text). This is default
* inherit - The element inherits the float value of its parent

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

## 

## Example - float: right;

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



### Example

img {

float: right; //floats the image to right

float: left; //floats the image to left

}

## The display: inline-block Value

Compared to display: inline, the major difference is that display: inline-block allows to set a width and height on the element.

Also, with display: inline-block, the top and bottom margins/paddings are respected, but with display: inline they are not.

Compared to display: block, the major difference is that display: inline-block does not add a line-break after the element, so the element can sit next to other elements.

The following example shows the different behavior of display: inline, display: inline-block and display: block:

### Example

span.a {

display: inline; /\* the default for span \*/

width: 100px;

height: 100px;

padding: 5px;

border: 1px solid blue;

background-color: yellow;

}

span.b {

display: inline-block;

width: 100px;

height: 100px;

padding: 5px;

border: 1px solid blue;

background-color: yellow;

}

span.c {

display: block;

width: 100px;

height: 100px;

padding: 5px;

border: 1px solid blue;

background-color: yellow;

}

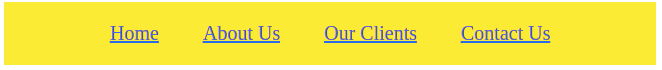
## Using inline-block to Create Navigation Links

One common use for display: inline-block is to display list items horizontally instead of vertically. The following example creates horizontal navigation links:

Example



The above CSS code gives result as :

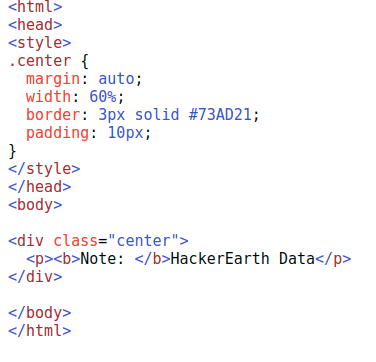


## Center Align Elements

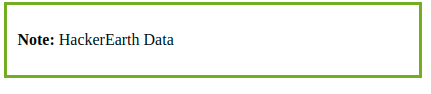
To horizontally center a block element (like <div>), use margin: auto;

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:



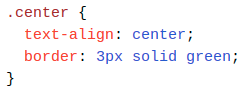
The above CSS code gives the result as follows:



## Center Align Text

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

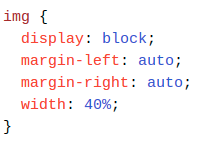
### Example



### Center an Image

### To center an image, set left and right margin to auto and make it into a block element:

### Example



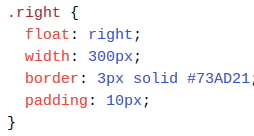
## Left and Right Align - Using position

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

## Left and Right Align - Using float

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

### Example



## 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:

## 

## CSS Combinators

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 CSS:

* 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 children of a specified element.

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

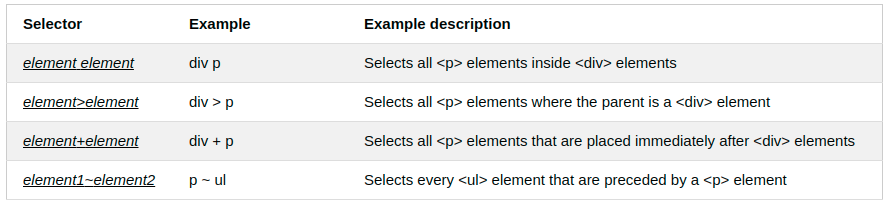
### Example

div ~ p {

background-color: yellow;

}

## All CSS Combinator Selectors



## 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

## Syntax

The syntax of pseudo-classes:



Anchor Pseudo-classes

Links can be displayed in different ways:



## 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:

## 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":

## 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 |

## 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

## Syntax

The syntax of pseudo-elements:

selector::pseudo-element {

property: value;

}

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

## Transparent Image

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



### 

### Example

img {

opacity: 0.5;

}

## 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;

}

img:hover {

opacity: 1.0;

}

### Example explained

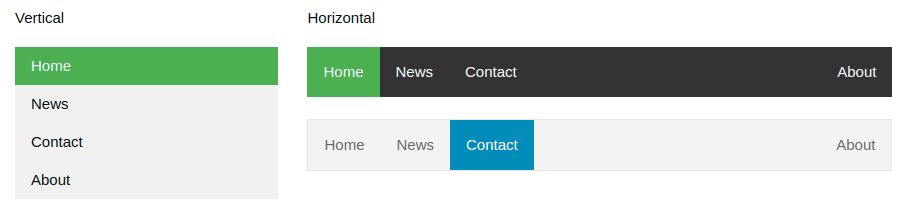
The first CSS block is similar to the code in Example 1. In addition, we have added what should happen when a user hovers over one of the images. In this case we want the image to NOT be transparent when the user hovers over it. The CSS for this is opacity:1;.

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

## 

## 

## Demo: Navigation Bars



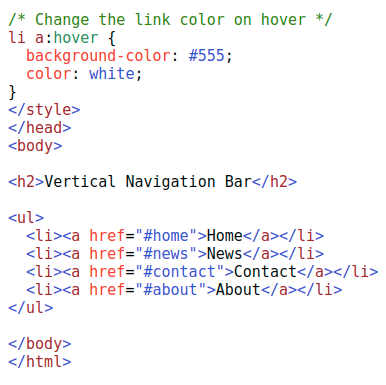
## Navigation Bars

Having easy-to-use navigation is important for any web site.

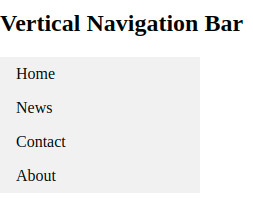
With CSS you can transform boring HTML menus into good-looking navigation bars.

## 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:

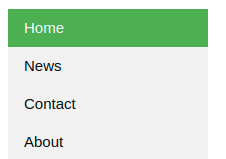


The above CSS code gives the Result as:



### Active/Current Navigation Link

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



### Example

.active {

background-color: #4CAF50;

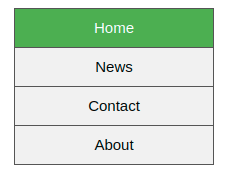
color: white;

}

### 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:



### Example

ul {

border: 1px solid #555;

}

li {

text-align: center;

border-bottom: 1px solid #555;

}

li:last-child {

border-bottom: none;

}

## Basic Dropdown

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

### Example



### Example Explained

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 uses 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 CSS 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.

----------------------------------End of Doc------------------------------------

------------------------Reference Credits: W3SCHOOLS,Tutorialspoint--------------------------