CSS Properties

We will learn the most common properties of CSS which are used frequently

# CSS Backgrounds

The CSS background properties are used to add background effects for elements.

We have the following CSS background properties:

* background-color
* background-image
* background-repeat
* background-attachment
* background-position
* background (shorthand property)

## 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: lightpink;

}

You can set the background color for any HTML elements:

**Example**

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

h1

{

background-color: red;

}

div

{

background-color: green;

}

p

{

background-color: blue;

}

## 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**

Set the background image for a page:

body

{

background-image: url("cute-dog.jpg");

}

The background image can also be set for specific elements, like the <p> element:

**Example**

p

{

background-image: url("cute-cat.jpg");

}

## background-repeat

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

The background-repeat property sets if/how a background image will be repeated.

**Example**

Repeat a background image only horizontally:

body

{

background-image: url("cute-dog.jpg");

background-repeat: repeat-x;

}

**Example**

Repeat a background image only vertically:

body

{

background-image: url("cute-dog.jpg");

background-repeat: repeat-y;

}

**Example**

Does not repeat the background image:

body

{

background-image: url("cute-dog.jpg");

background-repeat: no-repeat;

}

## background-position

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

The background-position property sets the starting position of a background image.

By default, a background-image is placed at the top-left corner of an element, and repeated both vertically and horizontally.

**Example**

Position the background image in the top-right corner:

body

{

background-image: url("cat.jpg");

background-repeat: no-repeat;

background-position: right top;

}

You can specify the position using the following keywords:

* left top
* left center
* left bottom
* right top
* right center
* right bottom
* center top
* center center
* center bottom

**Note:** If you only specify one keyword, the other value will be "center".

You can also use % for position or px for position.

**x% y%** The first value is the horizontal position and the second value is the vertical. The top left corner is 0% 0%. The right bottom corner is 100% 100%. If you only specify one value, the other value will be 50%. Default value is: 0% 0%

**xpos ypos** The first value is the horizontal position and the second value is the vertical. The top left corner is 0 0. Units can be pixels (0px 0px) or any other CSS units. If you only specify one value, the other value will be 50%. You can mix % and positions.

## 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("cat.jpg");

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.

Instead of writing:

body

{

background-color: #ffffff;

background-image: url("cute-cat.jpg");

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.

# Borders

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

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;}

## 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;

}

### 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):

**Example**

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 \*/

}

## 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;

}

## Border - Individual Sides

From the examples on the previous pages, you have seen that it is possible to specify a different border for each side.

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

**Example**

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:

**Example**

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\

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

## Border - Shorthand Property

When dealing with borders we have many properties. 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

**Example**

p

{

border: 5px solid red;

}

## Rounded Borders

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

**Example**

p

{

border: 2px solid red;

border-radius: 5px;

}

# 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

**Note:** 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;

}

If the margin property has one value:

margin: 25px;

all four margins are 25px

**Example**

Use the margin shorthand property with one value:

p {

margin: 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.

**Example**

Use margin: auto:

div

{

width: 300px;

margin: auto;

border: 1px solid red;

}

## 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;

}

In the example above, the <h1> element has a bottom margin of 50px and 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.

# 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

**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 - 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

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

**Example**

Use the padding shorthand property with four values:

div {

padding: 25px 50px 75px 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

**Example**

Use the padding shorthand property with three values:

div {

padding: 25px 50px 75px;

}

If the padding property has two values:

padding: 25px 50px;

top and bottom paddings are 25px

right and left paddings are 50px

**Example**

Use the padding shorthand property with two values:

div {

padding: 25px 50px;

}

If the padding property has one value:

padding: 25px;

all four paddings are 25px

**Example**

Use the padding shorthand property with one value:

div {

padding: 25px;

}

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 (the box

model).

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 actual width; if you increase the padding, the

available content space will decrease.

**Example**

Use the box-sizing property to keep the width at 300px, no matter the amount of padding:

div {

width: 300px;

padding: 25px;

box-sizing: border-box;

}

# CSS Box Model

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.



This is the content.



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: 400px;

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.

**Note:** 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

**Note**: Outline differs from borders! Unlike border, the outline is drawn outside the element's border, and may overlap other content. Also, the outline is NOT a part of the element's dimensions; the element's total width and height is not affected by the width of the outline.

# Text Formatting

## 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 Align Last

The text-align-last property specifies how to align the last line of a text.

Example

Align the last line of text in three <p> elements:

p.a {

text-align-last: right;

}

p.b {

text-align-last: center;

}

p.c {

text-align-last: 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;

}

## Text Decoration

In this chapter you will learn about the following properties:

* text-decoration-line
* text-decoration-color
* text-decoration-style
* text-decoration-thickness

## text-decoration

Add a Decoration Line to Text

The text-decoration-line property is used to add a decoration line to text.

**Note:** You can combine more than one value, like overline and underline to display lines both over and under a text.

**Example**

h1

{

text-decoration-line: overline;

}

h2

{

text-decoration-line: line-through;

}

h3

{

text-decoration-line: underline;

}

p

{

text-decoration-line: overline underline;

}

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

### Specify a Color for the Decoration Line

The text-decoration-color property is used to set the color of the decoration line.

Example

h1 {

text-decoration-line: overline;

text-decoration-color: red;

}

h2 {

text-decoration-line: line-through;

text-decoration-color: blue;

}

h3 {

text-decoration-line: underline;

text-decoration-color: green;

}

p {

text-decoration-line: overline underline;

text-decoration-color: purple;

}

Specify a Style for the Decoration Line

The text-decoration-style property is used to set the style of the decoration line.

Example

h1 {

text-decoration-line: underline;

text-decoration-style: solid;

}

h2 {

text-decoration-line: underline;

text-decoration-style: double;

}

h3 {

text-decoration-line: underline;

text-decoration-style: dotted;

}

p.ex1 {

text-decoration-line: underline;

text-decoration-style: dashed;

}

p.ex2 {

text-decoration-line: underline;

text-decoration-style: wavy;

}

p.ex3 {

text-decoration-line: underline;

text-decoration-color: red;

text-decoration-style: wavy;

}

Specify the Thickness for the Decoration Line

The text-decoration-thickness property is used to set the thickness of the decoration line.

Example

h1 {

text-decoration-line: underline;

text-decoration-thickness: auto;

}

h2 {

text-decoration-line: underline;

text-decoration-thickness: 5px;

}

h3 {

text-decoration-line: underline;

text-decoration-thickness: 25%;

}

p {

text-decoration-line: underline;

text-decoration-color: red;

text-decoration-style: double;

text-decoration-thickness: 5px;

}

### The Shorthand Property

The text-decoration property is a shorthand property for:

text-decoration-line (required)

text-decoration-color (optional)

text-decoration-style (optional)

text-decoration-thickness (optional)

Example

h1 {

text-decoration: underline;

}

h2 {

text-decoration: underline red;

}

h3 {

text-decoration: underline red double;

}

p {

text-decoration: underline red double 5px;

}

**Note:** All links in HTML are underlined by default. Sometimes you see that links are styled with no underline. The text-decoration: none; is used to remove the underline from links, like this:

Example

a {

text-decoration: none;

}

## Text Transformation

The text-transform property is used to specify uppercase and lowercase letters in a text.

It can be used to turn everything into uppercase or lowercase letters, or capitalize the first letter of each word:

**Example**

p.uppercase

{

text-transform: uppercase;

}

p.lowercase

{

text-transform: lowercase;

}

p.capitalize

{

text-transform: capitalize;

}

## Text Indentation

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

**Example**

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:

**Example**

h1 {

letter-spacing: 5px;

}

h2 {

letter-spacing: -2px;

}

## Line Height

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

**Example**

p.small {

line-height: 0.8;

}

p.big {

line-height: 1.8;

}

## 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**

p.one {

word-spacing: 10px;

}

p.two {

word-spacing: -2px;

}

# 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):

h1 {

text-shadow: 2px 2px;

}

You can also add text shadow color and shadow blur:

h1 {

color: white;

text-shadow: 2px 2px 2px #000000;

}

The above code adds a black shadow to the white text.

# 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.

### 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.

**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;

}

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

**Example**

span {

display: block;

}

## 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:

This <div> element has position: static;

Here is the CSS that is used:

**Example**

div.static {

position: static;

border: 2px solid #65fa09;

}

### 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: 2px solid #65fa09;}

### 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: 2px solid #65fa09;

}

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

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: 2px solid #65fa09;

}

div.absolute {

position: absolute;

top: 80px;

right: 0;

width: 200px;

height: 100px;

border: 2px solid #65fa09;

}

### 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).

Note: Internet Explorer does not support sticky positioning. Safari requires a -webkit- prefix (see example below). You must also specify at least one of top, right, bottom or left for sticky positioning to work.

In this example, the sticky element sticks to the top of the page (top: 0), when you reach its scroll position.

**Example**

div.sticky {

position: -webkit-sticky; /\* Safari \*/

position: sticky;

top: 0;

background-color: green;

border: 2px solid #65fa09;

}

# z-index

The z-index property specifies the stack order of an element. An element with greater stack order is always in front of an element with a lower stack order.

**Note**: z-index only works on positioned elements (position: absolute, position: relative, position: fixed, or position: sticky) and flex items (elements that are direct children of display:flex elements).

**Note**: If two positioned elements overlap without a z-index specified, the element positioned last in the HTML code will be shown on top.

<html>

<head>

<style>

img {

position: absolute;

left: 0px;

top: 0px;

z-index: -1;

}

</style>

</head>

<body>

<h1>The z-index Property demo</h1>

<img src="heart.jpg" width="100" height="140">

<p>Because the image has a z-index of -1, it will be placed behind the heading.</p>

</body>

</html>

**Example2:**

<html>

<head>

<style>

img {

position: absolute;

left: 0px;

top: 0px;

z-index: -1;

}

</style>

</head>

<body>

<h1>This is a heading</h1>

<img src="car.jpg">

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

</body>

</html>

**Example3:**

<html>

<head>

<style>

.container {

position: relative;

}

.black-box {

position: relative;

z-index: 1;

border: 2px solid black;

height: 100px;

margin: 30px;

}

.gray-box {

position: absolute;

z-index: 3; /\* gray box will be above both green and black box \*/

background: lightgray;

height: 60px;

width: 70%;

left: 50px;

top: 50px;

}

.green-box {

position: absolute;

z-index: 2; /\* green box will be above black box \*/

background: lightgreen;

width: 35%;

left: 270px;

top: -15px;

height: 100px;

}

</style>

</head>

<body>

<h1>Z-index Example</h1>

<p>An element with greater stack order is always above an element with a lower stack order.</p>

<div class="container">

<div class="black-box">Black box (z-index: 1)</div>

<div class="gray-box">Gray box (z-index: 3)</div>

<div class="green-box">Green box (z-index: 2)</div>

</div>

</body>

</html>