



CSS Properties



Learning Objectives

This lesson provides an overview of commonly used CSS properties. By the end of this lesson, learners will be able to:

- Apply display properties to elements in an HTML page.
- Apply position properties to elements in an HTML page.
- Modify fonts and text elements with CSS properties.
- Use CSS to fit objects to their containers and desired dimensions.
- Create box shadows on HTML elements.
- Modify the color of elements.
- Create borders on elements.
- Use CSS to align text in specific ways.
- Use the float property to position elements on an HTML page.

Table of Contents

- CSS Shorthand Properties.
- CSS Properties:
 - Display.
 - Position.
 - Font.
 - Color.
 - Background.
 - Border.
 - Text-Align.
 - Float and Clear.
 - Box Shadow.
 - Object-Fit.
- CSS Property Values: Gradients.

CSS Shorthand Properties

[Shorthand properties](#) are CSS properties that let you set the values of multiple CSS properties simultaneously. Using a shorthand property, you can write more concise (and often more readable) stylesheets, saving time and energy. Some shorthand properties include: **background**, **font**, **border**, **margin**, **padding**, **position**, and **all**, most of which we will cover in more detail in the following slides.

As an example:

- **border** is a shorthand property that sets the values of **border-width**, **border-style**, and **border-color**, among others.
- If we want to create a one pixel, dashed, red border around our HTML **body** element, we can use the following long-form syntax with each of the individual border properties, or the simpler shorthand:

```
body {  
  border-width: 1px;  
  border-style: dashed;  
  border-color: red;  
}
```

```
body {  
  border: 1px dashed red;  
}
```

- It is important to note that there are some tricky edge cases when using shorthand properties, especially in the case of multiple properties that are of the same value type. For details on these edge cases and more, [visit this reference section](#).

CSS Properties: Display

The [display](#) property defines how an element will be displayed on the DOM. The **display** property consists of two basic ways of displaying an element:

- **Outer Display Type:** Tells the element if it needs to be displayed with elements to either side of it or if it should not allow other boxes to be right next to it.
- **Inner Display Type:** Tells the children of the box how to be displayed.

Display Outside Values:

- display: block;
- display: inline;
- display: run-in;

Display Inside Values:

- display: flow;
- display: flow-root;
- display: table;
- display: flex;
- display: grid;
- display: ruby;

Display Outside and Inside:

- display: block flow;
- display: inline table;
- display: flex run-in;

Display Box Values:

- display: contents;
- display: none;

Display Global Values:

- display: inherit;
- display: initial;
- display: unset;

Display Internal Values:

- display: table-row-group;
- display: table-header-group;
- display: table-footer-group;
- display: table-row;
- display: table-cell;
- display: table-column-group;
- display: table-column;
- display: table-caption;

Display Example: inline block

Use the provided [inline.html](#) file.

Inline block allows other boxes to sit right next to each other. If the box with this display property does not fit in the space available, it will break to a new line.

- The parent `<div>` has the dashed border and its measurements are 210 x 205 pixels.
- The inner divs, or child divs, have a width and height of 100px. Also, we have to take into account the border width.
- The second child `<div>` will sit right next to the first because of the space available. The third child `<div>` will create a new line.
- Change the width of the parent `<div>` from 210px to 400px.
- Now all of the child divs fit on the same line.

Display Example: block

Use the provided [block.html](#) file.

Block does not allow other boxes to sit right next to each other. If the box with this display property does not fit in the space available, it will try to fit some of its content, and whatever does not fit goes to a new line. To do this, shrink your browser and see it break, and then toggle between **block** and **inline block** to see the difference.

- The parent div has the dashed border and its measurements are 210 x 205
- The child divs have a width and height of 100px. Also, we have to take into account the border width.
- Comment out the width of the child divs and its size will become the width of the container.
- Shrink your browser and you will see the text break into a new line.
- Comment out the span's **display** property and value, and you will see that it no longer creates a new line because **span** is **inline** by default.

CSS Properties: Position

The **position** property is used to align elements on the page and organize content. It can accept many different values. For now, we are going to look at four values; later, we are going to specifically look at **flex**.

- **static**: Static is the default position for HTML elements. Elements with **position: static** are positioned based on the normal flow of the page, as you would expect them to be without any CSS styling. They are not affected by the **top**, **right**, **bottom**, or **left** properties. **Z-index** also does not apply to static elements.
- **relative**: Displayed as a static box; the relative position is relative to the parent container. However, with **relative** or any value other than **static**, you have access to additional properties such as: **top**, **left**, **right**, and **bottom**, which allows you to modify the actual position of the box.
- **fixed**: A fixed element is positioned relative to the viewport, which means it always stays in the same place even if the page is scrolled.
- **absolute**: The element is removed from the normal document flow, and no space is created for the element in the page layout. It is positioned relative to its closest positioned ancestor, if any; otherwise, it is placed relative to the initial containing block. Its final position is determined by the values of **top**, **right**, **bottom**, and **left**.

Example: Relative Positioning

Use the provided [relative.html](#) file.

Displayed as a static box; the element's position is relative to the parent container. However, with **relative** or any value other than **static**, you have access to additional properties such as: **top**, **left**, **right**, and **bottom**, which allows you to modify the actual position of the box.

- Open your developer tools (browser inspector) and select the third div to see the position added to the box model. Its position is currently set to zero.
- In the **relative.html** page, go to the class **exampleBox** code block in the style tag.
- Add one or more of these properties: **top**, **left**, **right**, or **bottom** with values.
- The div with the class **exampleBox** will move only relative to the flow of the documents, which means that the div will only move from where it is currently positioned.
- Depending on the property used, your value will change the div from a position of zero, moving from its current position to the new value.

Example: Fixed Positioning

Use the provided [fixed.html](#) file.

A fixed element is positioned relative to the viewport, which means that it always stays in the same place even if the page is scrolled. This makes fixed elements particularly useful for user interface, much like navigation bars.

- Open your developer tools (browser inspector) and select the third div to see the position added to the box model. Its position is currently set to zero.
- In the **fixed.html** page, go to the class **exampleBox** code block in the style tag.
- Add one or more of these properties: **top**, **left**, **right**, or **bottom** with values.
- When the div is moved, space is not created as compared with **display: relative**.
- Notice how scrolling the page does not affect the location of the fixed element.

Example: Absolute Positioning

Use the provided [absolute.html](#) file .

The element is removed from the normal document flow, and no space is created for the element in the page layout. It is positioned relative to its closest positioned ancestor, if any; otherwise, it is placed relative to the initial containing block. Its final position is determined by the values of **top**, **right**, **bottom**, and **left**.

- Open your developer tools and select the third div to see the position added to the box model. Its position is currently *not* set to zero as compared with `display: relative`.
- In the **absolute.html** page, go to the class **exampleBox** code block in the `<style>` tag.
- Add one or more of these properties: **top**, **left**, **right**, or **bottom** with values.
- When the div is moved, space is not created as compared with **display: relative**.

CSS Properties: Font

The [font](#) property is one of the most basic properties of a page, and controls the look and feel of text on the page. The **font** property is shorthand for a number of other properties, and can contain a wide variety of values.

- When specified as a shorthand property, **font** must include values for **font-size** and **font-family**.
- The **font** can also be specified as a system keyword with a single value, but that value must be one of: **caption**, **icon**, **menu**, **message-box**, **small-caption**, or **status-bar**.
- You can obtain additional font families by using the [<link>](#) HTML element or the [@import](#) CSS at-rule.
 - The [@import](#) CSS at-rule is used to import style rules from other stylesheets.
- You are able to reference font families directly from the internet or by downloading them locally.
- To use or download Google fonts, go to: <https://fonts.google.com/>.

[For a full reference on fonts, click here.](#)

Example: Font

In an **index.css** file using the type selector, add **font-family** to an **<h1>** tag as below. Link the following fonts from within your **index.html** page, and observe the results on an **<h1>** element. Do not forget to link your **index.css** file as well, if you have not already done so.

```
<link rel="preconnect" href="https://fonts.googleapis.com" />
<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin />
<link href="https://fonts.googleapis.com/css2?family=Bungee+Spice&display=swap"
rel="stylesheet"/>
```

```
h1 {
  font-family: 'Bungee Spice';
}
```

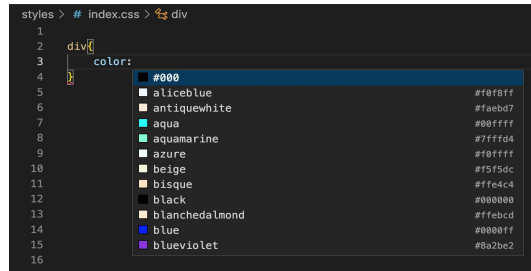
BUNGIE SPICE

CSS Properties: Color

As we know, [color](#) is a very important property for any website, as it aids in bringing style and attention to various parts of a website.

On the next several slides, we will review some of the ways you can assign color to elements using CSS, such as:

- [Color Keywords](#)
- [RGB](#)
- [RGBA](#)
- [Hexadecimal](#)
- [HSL](#)
- [HSLA](#)
- [HWB](#)



[For a full reference on applying color to HTML elements using CSS, click here.](#)

CSS Properties: Color – “rgb()” or “rgba()”

Two CSS functions, “rgb()” and “rgba(),” allow for modification of color:

- **Red (r)**, **green (g)**, and **blue (b)** color values are passed to the functions as arguments.
 - Each of these parameters can take a value from 0 to 255.
- An alpha (a) value can be passed as the fourth argument to **rgba()** to modify the opacity of the color, if desired.
 - This property can take values from 0.00 to 1.00, where 0 is transparent.
 - An alpha value of 0 will cause the content to not be displayed, but it **does** occupy space in the DOM. Take care not to confuse this with **display: none**, which is also not displayed, but **does not** occupy space.
- Other possible values for these parameters are percentages, which range from 0 - 100 percent.

```
div {
  color: rgb(23, 0, 3);
}

p {
  color: rgba(100%, 50%, 50%, .33);
}
```

CSS Properties: Color – “hex”

Hexadecimal notation (#), also known as hex, is another way to assign color values to properties. Hexadecimal refers to the number base for this system, which is base 16. Hexadecimal is counted such as: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 1A, 1B, 1C, 1D, 1E, 1F, 20, and so on.

- **Red (r)**, **green (g)**, and **blue (b)** color values are defined following the # symbol.
 - Each of these parameters can take a value from 00 to FF.
- An alpha (a) value can be passed as the fourth argument to modify the opacity of the color, if desired.
 - An alpha value of 0 will cause the content to not be displayed, but it **does** occupy space in the DOM. Take care not to confuse this with **display: none**, which is also not displayed, but **does not** occupy space.
- Hexadecimal notation can also be shortened with three-digit or four-digit notation, where each character is repeated once.
 - For example, #F09 is the same as #FF0099, and #0F38 is the same as #00FF3388.

```
div {
  color: #A9F;
}

p {
  color: #AA3423CC;
}
```


CSS Properties: Background

The [background](#) property modifies the background of a container. The **background** property is shorthand for a number of other properties, and can contain a wide variety of values.

- When specified as a shorthand property, **background** does not have any required values, but component properties that are not explicitly set will be set to their default values.
- This property is shorthand for the following additional properties, and can accept values for one or all of them:
 - background-attachment
 - background-clip
 - background-color
 - background-image
 - background-origin
 - background-position
 - background-repeat
 - background-size

```
<div class="imgContainer"></div>
<div>
  <p><strong>Lorem</strong><sub>ipsum dolor</sub>
  sit amet, consectetur adipiscing elit.</p>
</div>
```

```
.imgContainer {
  height: 500px;
  background: no-repeat scroll center contain url("../images/test.png");
}
```

[For a full reference on backgrounds, click here.](#)

CSS Properties: Border

The [border](#) property modifies the border of an element. Just like the **background** property, the **border** property is shorthand for a number of other properties, and can contain a wide variety of values.

- When specified as a shorthand property, **border** does not have any required values, but component properties that are not explicitly set will be set to their default values.
- This property is shorthand for the following additional properties, and can accept values for one or all of them:
 - border-color
 - border-style
 - border-width
- The **border-color** takes any of the color formats previously discussed on the **color** property slide.
- You can also modify a border's sharp corners by using the **border-radius** property.

```
<div class="imgContainer"></div>
<div>
  <p><strong>Lorem</strong><sub>ipsum dolor</sub>
  sit amet, consectetur adipisicing elit.</p>
</div>
```

```
div {
  height: 500px;
  border: 2px ridge rgba(100%, 50%, 50%, .33);
}
```

[For a full reference on borders, click here.](#)

CSS Properties: Text-Align

The `text-align` property sets the horizontal alignment of content inside of a block-level element or table-cell box. The `text-align` property can be specified in a number of ways, including keyword values, character-based alignment in a table column, block alignment values, and global values.

Text-Align Keyword Values:

- `text-align: start;`
- `text-align: end;`
- `text-align: left;`
- `text-align: right;`
- `text-align: center;`
- `text-align: justify;`
- `text-align: justify-all;`
- `text-align: match-parent;`

Character-Based Alignment:

- `text-align: ".";`
- `text-align: "." center;`

Block Alignment Values:

- `text-align: -moz-center;`
- `text-align: -webkit-center;`

Global Values:

- `text-align: inherit;`
- `text-align: initial;`
- `text-align: revert;`
- `text-align: revert-layer;`
- `text-align: unset;`

[For a full reference on text alignment, click here.](#)

CSS Properties: Float and Clear

The **float** property specifies that an element should be placed along the left or right side of its container, allowing text and other inline elements to wrap around it. The element is removed from the normal flow of the webpage, though still remaining a part of the DOM.

If we were to create three divs, and float one to the left and the other to the right, we would notice that the third div displays behind the others, because the **float** property removes the other divs from the normal flow; therefore, not taking up any space. In order to fix issues like this one, we make use of the **clear** property.

The **clear** property sets whether an element must be moved clear of floating elements that precede it. This property can apply to both floating and non-floating elements.



Source: <https://developer.mozilla.org/en-US/docs/Web/CSS/clear>

CSS Properties: Box Shadow

The [box-shadow](#) property adds shadow effects around an element's frame. You can set multiple effects, separated by commas. A box shadow is described by x-axis and y-axis offsets, relative to the element; blur; spread; radius; and color. The **box-shadow** property can take multiple values, but the number and order of those values matters.

- **Offset-X Offset-Y:** This two-value approach is the simplest way to implement a box shadow.
- **Offset-X Offset-Y Blur-Radius:** If a third length value is added, it is interpreted to be a blur radius, which increases the "softness" of a shadow.
- **Offset-X Offset-Y Blur-Radius Spread-Radius:** If a fourth length value is added, it is interpreted to be a spread radius, which either increases (positive values) or decreases (negative values) the size of the shadow.
- **inset:** If the **inset** keyword is added to the properties of a box shadow, that shadow will appear inside of the element, as if the content was debossed inside the box, rather than as if the box were raised above the content.
- **Color:** A color value can be included to modify the color of the box shadow.

```
div {
  box-shadow: 2px 2px 1px rgba(100%, 50%, 50%, .33);
}
```

[For a full reference on box shadows, click here.](#)

CSS Properties: Object-Fit

The **object-fit** property is used to specify how the content of a replaced element, such as an `` or `<video>` element, should be resized to fit its container. There are five values for scaling content via **object-fit**:

- **fill**: The replaced content is sized to match the element's content box, completely filling the box. This will stretch the content to fit the box if the aspect ratios do not match.
- **contain**: The replaced content is scaled to maintain its aspect ratio, while filling at least one dimension of the element's content box, making it "letterboxed" if the ratios do not match.
- **cover**: The replaced content is scaled to maintain its aspect ratio, while filling the entirety of the element's content box, clipping it if the ratios do not match.
- **none**: The replaced content is not resized.
- **scale-down**: The replaced content is sized as if **none** or **contain** were specified, whichever would result in a smaller object size.



[For a full reference on object fit, click here.](#)

CSS Property Values: Gradients

CSS gradients are represented by the `<gradient>` data type, which is a special type of `<image>` made of a progressive transition between two or more colors. You can choose between five types of CSS gradients:

- **linear-gradient()** transitions colors progressively among an imaginary line.
- **radial-gradient()** transitions colors progressively from a center point.
- **repeating-linear-gradient()** transitions colors as a linear gradient does, repeating as much as necessary to fill a given area.
- **repeating-radial-gradient()** transitions colors as a radial gradient does, repeating as much as necessary to fill a given area.
- **conic-gradient()** transitions colors progressively around a circle.

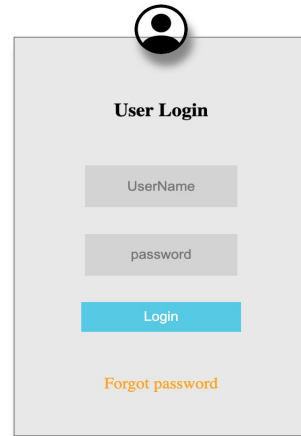


[For a full reference on gradients, click here.](#)

Activity One: Login Interface

Create a login interface similar to the image provided, using everything you have learned about HTML and CSS so far.

As always, let your creativity guide your path.

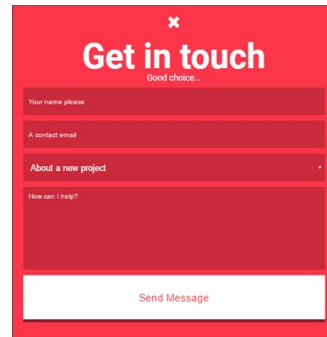


A mockup of a user login interface. It features a circular profile icon at the top. Below it is the title "User Login". There are three input fields: "UserName", "password", and a blue "Login" button. At the bottom, there is a link that says "Forgot password" in orange text.

Activity Two: Generic Form

Create a form similar to the image provided, using everything you have learned about HTML and CSS so far.

As always, let your creativity guide your path.



A red contact form titled "Get in touch" with a small "x" icon in the top right corner. Below the title is the text "Good choice...". The form contains four input fields: "Your name please", "A contact email", "About a new project", and "How can I help?". The "About a new project" field has a small downward arrow on its right side. At the bottom of the form is a white button labeled "Send Message".

Knowledge Check

- What is a shorthand property in CSS?
- What is the CSS **display** property used for?
- What is the difference between **display:inline** and **display:block**?
- What is the difference between absolute and relative positioning?
- What are two ways that you can embed new font families into your document?
- What is the difference between rgb and rgba?
- What is the syntax for hexadecimal colors?
- Name four text-align values.
- Describe the relationship between the CSS **float** property and the CSS **clear** property.
- How do you fit a replaced content object to its container?
- How do you implement a gradient in CSS, and what are the options for doing so?

Summary

In this lesson, we explored CSS properties, which allow you to style elements in specific ways. CSS shorthand properties can make code more efficient and readable by combining many commonly used properties into a single line. For example:

- CSS positioning is one of the key topics in CSS, which allows us to place HTML elements at various locations on a page, creating organized and visually appealing content.
- CSS provides several properties for styling the font of text, including changing its face, controlling its size and boldness, managing variants, and so on.
 - Font properties include: `font-family`, `font-style`, `font-weight`, `font-size`, and `font-variant`.
- CSS provides several properties for styling the background of an element, including coloring the background, placing images in the background and managing their positioning, etc.
 - The background properties include `background-color`, `background-image`, `background-repeat`, `background-attachment` and `background-position`.
- The `color` property defines the text color (foreground color in general) of an element.

Questions?

