What is CSS?

As we have mentioned, in this chapter we are going to be focusing on style, presentability and the appearance of our webpage. The main language used for styling a web page is CSS.

Fasten your seatbelts because in this chapter we are going to learn:

* What is CSS?
* How to integrate CSS into HTML?
* How to select an HTML element?
* How to change the position of an element?

# What is CSS?

Cascading Style Sheets, commonly referred to as **CSS**, is a simple design language intended to simplify the process of making web pages presentable.

**CSS** determines the look and feel of a web page. With **CSS**, you can control the color of a text, the style of fonts, the spacing between paragraphs, how columns are sized and laid out. It also defines what background images or colors are used, the layout designs, and most importantly, the variations in display to adapt to different devices and screen sizes.

CSS is easy to learn and understand but it offers powerful control over the presentation of an HTML document. Most commonly, CSS is combined with the markup language HTML.

What is CSS?

CSS takes part in building a beautiful and colorful website instead of a bland, static, black and white web page. CSS is responsible for adding flair and sprucing up your web page. So What is CSS?

* CSS (Cascading Style Sheets) is a language used to style HTML pages.
* CSS allows us to add colors and fonts to our texts, adjust the layout, etc...
* CSS is the tool that reads the CSS code and understands how the web page is displayed.
* CSS separates the presentation of a document from its structure and content.

=> HTML describes the structure of the webpage while CSS describes the way it should be displayed on the screen.

Linking CSS to HTML:

Before starting with CSS and how it works, we should beforehand, know how to link CSS to our HTML code.

There are three possible ways to attach CSS to HTML:

* Inline : by using the style attribute in HTML elements.
* Internal : by using a < style > element in the < head > section.
* External : by using an external CSS file.



# Inline CSS:

The first way to style our HTML elements is via the attribute style where we can add the design we want to this element.

In situations where we have a small number of elements to deal with, this could be very helpful. But once we’re dealing with large pages with lots of moving pieces, it will become extremely tedious to apply a separate style attribute to each element.

<body>

<h1 style="color: aqua">Hello World</h1>

<p style="border: 1px blue; color: crimson">

This is my first paragraph

</p>

</body>

# Internal CSS:

The second way to add the CSS style is to include it directly inside the HTML document.  
Notice the use of the style HTML element nested in the head element. The style element can be used to place CSS rules inline with an HTML document, like in the example below.  
Although this is an absolutely valid way of adding CSS to your web pages, it will not let us reuse the same CSS style in other HTML files.

<!DOCTYPE html>

<html lang="">

<head>

<meta charset="" />

<meta name="viewport" content="width=, initial-scale=" />

*<!-- Internal style -->*

<style>

h1 {

color: aqua;

font: sans-serif;

}

p {

color: darkblue;

font: small;

}

</style>

<title></title>

</head>

<body></body>

</html>

# External CSS

The last and probably the best way to use CSS (in term of code reuse) is to create an external file with the extension **.css** and import it into our HTML document with the special link tag.

The link tag is placed inside the head of the document, with the attribute href that indicates the path of the CSS file.

<!DOCTYPE html>

<html lang="">

<head>

<meta charset="" />

<link rel="stylesheet" href="./styles.css" />

<meta name="viewport" content="width=, initial-scale=" />

<title></title>

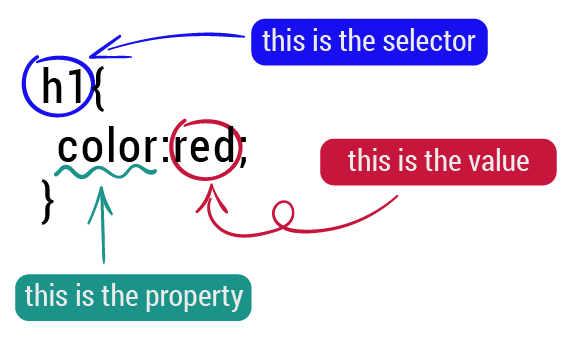
</head>

<body></body>

</html>

# CSS Selector:

After attaching the CSS style, our next step is figuring out how to attach a chosen style to the proper element.  
If we want to change the style of a particular element in our HTML page, We have to select that element first.  
To do so, CSS provides a set of rules, each of which consists of a selector (To indicate which elements you are trying to modify) , followed by a declaration block that contains a set of properties and those properties’ values.

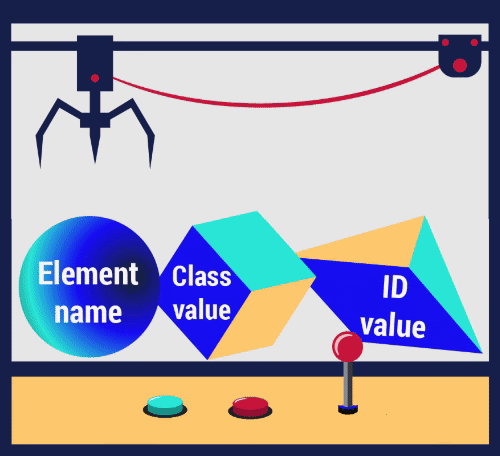


Selector types:

CSS rules can select an element in a variety of ways. The three basic kinds of selectors are:

1. **Type** selectors: used to select HTML elements by element name.
2. **Class** selectors: used to select HTML elements by a specific class value.
3. **Id** selectors: used to select an HTML element associated with a specific id value.

There are other types of CSS selectors, but we are not going to cover them right now. You can find out more in this [link](https://developer.mozilla.org/en-US/docs/Learn/CSS/Building_blocks/Selectors/Combinators).



# Type Selector:

In the previous slides, we have learned how to pick an HTML element in order to style it and we have encountered several types of CSS selectors.  
Straight away, we will view the first type of selector and implement it in a real example.

## **Type:**

Using a type selector is as simple as typing the name of the element. However, it’s preferable if you know when to use it. This selector is used when we want to apply the same rule on every element in the page.

For example, we want every h1 title to be in red, the font will be roboto and the text-align will be in the center.

* HTML

<!DOCTYPE html>

<html lang="">

<head>

<title>Type selector</title>

</head>

<body>

<h1>Welcome to my blog</h1>

<p>

Lorem ipsum dolor sit amet consectetur adipisicing elit. Blanditiis

ratione alias, eaque temporibus tenetur similique accusamus quas

laudantium ea reiciendis itaque atque earum provident. Ab nisi sunt quo

repellat vitae.

</p>

<h1>I am a super web developer</h1>

</body>

</html>

* CSS

h1 {

color: red;

font-family: roboto;

text-align: center;

}

* Output:



Class Selectors:

The selection by type has worked fine until now, but let's say we have shared style not only for one type but for multiple types. For example, we are going to put all the text in a blue section and the other section will be in grey. The type selector will be extremely helpful here. That is where **class** selectors come in to save the day!

A Class is a tag attribute where we can name an identifier for the element to adopt a certain style, you can specify a class on CSS with a (.) period mark.

To use the class selector, you can follow the example below:

* HTML

<!DOCTYPE html>

<html lang="">

<head>

<title>Type selector</title>

</head>

<body>

<h1 class="title">Welcome to my blog</h1>

<p class="title">

Lorem ipsum dolor sit amet consectetur adipisicing elit. Blanditiis

ratione alias, eaque temporibus tenetur similique accusamus quas

</p>

<h1>I am a super web developer</h1>

</body>

</html>

* CSS

.title{

color:blue

}

* Output:



Id selector:

The final type of CSS selector is the selection by Id.

ID selectors in CSS allow you to target elements (Tags) by their ID values. ID selectors are unique, so you can only apply them to the content of one element. To reference an ID, you precede the ID name with a hash mark (#).

* HTML

<h1 id="blue-bordered">Now we are using an Id Selector</h1>

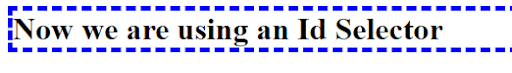
* CSS

#blue-bordered {

border: 5px dashed blue;

}

* Output:



# Multiple selectors :

One of the most important principles in programming is the “DRY” principle: Don’t repeat Yourself.

Let's assume that we have a p tag, a div tag and section tag that share the same background color. So, by following the “DRY” principle, we must not repeat the styling three times.

CSS allows us to do that, we can select multiple elements, separate the selectors by commas, like this:

*/\* Selecting multiple HTML element types \*/*

h1, p{

border: 1px solid black;

}

*/\* Selecting styles to be applied to several classes \*/*

.FirstItem, .LastItem {

font-size: 1.2em;

}

*/\* Using multiple Kinds of selectors \*/*

h3, .red, #redElement {

color: red

}

Descendant Combinator:

The descendant combinator is typically represented by a single space character. It combines two selectors in a way that elements matched by the second selector are selected if they have an ancestor (parent, parent's parent, parent's parent's parent, etc.) element matching the first selector. Selectors that utilize a descendant combinator are called descendant selectors.

* html

<div class="box"><p>Text in .box</p></div>

<p>Text not in .box</p>

* CSS

.box p {

color: red;

}

* Output



Child Combinator:

The child combinator (>) is placed between two CSS selectors. It only matches those elements matched by the second selector that are the direct children of elements matched by the first. Descendent elements further down the hierarchy don't match.  
For example, to select only <p> elements that are direct children of <article> elements:

* html

<article>

<h1>A heading</h1>

<p>I am a paragraph.</p>

<div>I am a div</div>

<p>I am another paragraph.</p>

</article>

Css

article > p {

font-weight: bold;

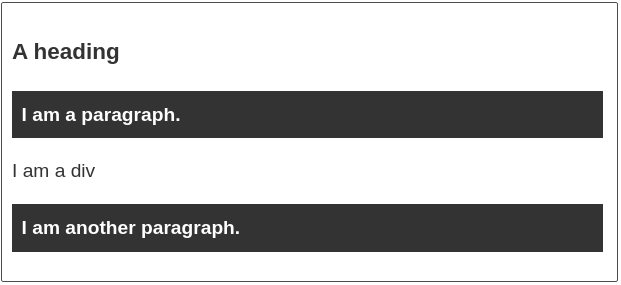
background-color: #333;

color: #fff;

padding: .5em;

}

* Output



Adjacent sibling Combinator:

The adjacent sibling selector (+) is used to select something if it is right next to another element at the same level of the hierarchy.  
For example, to select all <p> elements that come right after <h1> elements:

* **HTML** :

<article>

<h1>A heading</h1>

<p>Veggies es bonus vobis, proinde vos postulo essum magis kohlrabi welsh onion daikon amaranth tatsoi tomatillo

melon azuki bean garlic.</p>

<p>Gumbo beet greens corn soko endive gumbo gourd. Parsley shallot courgette tatsoi pea sprouts fava bean collard

greens dandelion okra wakame tomato. Dandelion cucumber earthnut pea peanut soko zucchini.</p>

</article>

* **CSS**:

h1 + p {

font-weight: bold;

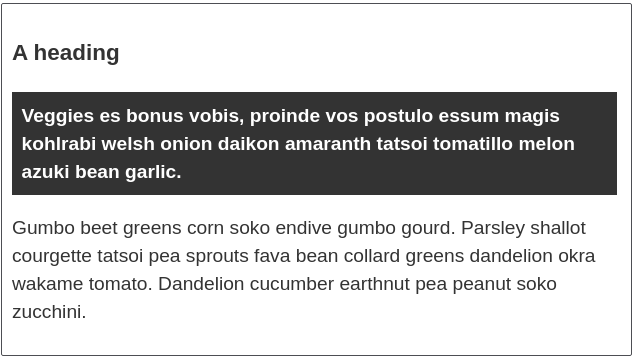
background-color: #333;

color: #fff;

padding: .5em;

}

* Output:



# General sibling Combinator

If you want to select siblings of an element even if they are not directly adjacent, then you can use the general sibling combinator (~). To select all <p> elements that come anywhere after <h1> elements, we'd do this:  
Html

<article>

<h1>A heading</h1>

<p>I am a paragraph.</p>

<div>I am a div</div>

<p>I am another paragraph.</p>

</article>

Css

h1 ~ p {

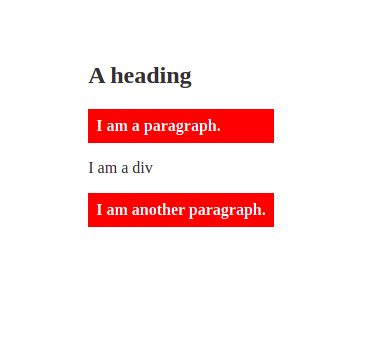
font-weight: bold;

background-color:red;

color: #fff;

padding: .5em;

}

Output  


# CSS pseudo classes:

Have you ever tried to put your cursor on a button and its color changed? Or Have you ever clicked on a link and the same thing happened? (marked as clicked)  
Well, it’s all done with CSS pseudo-classes.

As you can see, the structure is clear. You name your SELECTOR, add : then define your action.  
Here are a few awesome actions you can do. Try them on your editor.

*/\* unvisited link \*/*

a:link {

color: tomato;

}

*/\* visited link \*/*

a:visited {

color: tomato;

}

*/\* mouse over element \*/*

div:hover {

color: tomato;

}

*/\* selected link \*/*

a:active{

color: #0000FF

}

CSS box model

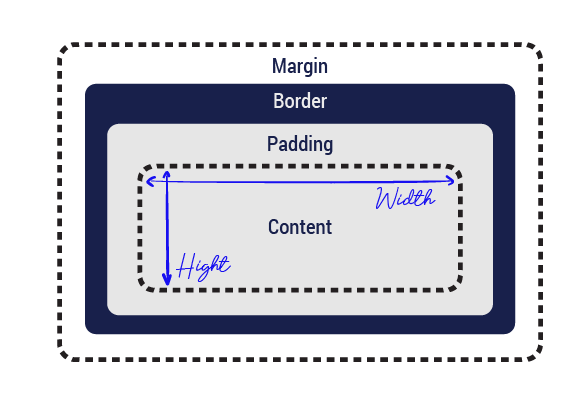
Every element in the web design is a rectangle in box form (even if it is a circle).

The CSS box model is defined by these four layers:

* Content: the content of the box, where text and images appear
* Padding: the padding is transparent, clears an area around the content.
* Border: a border that goes around the padding and content. We have already seen it earlier.
* Margin: clears an area outside the border. The margin is transparent.

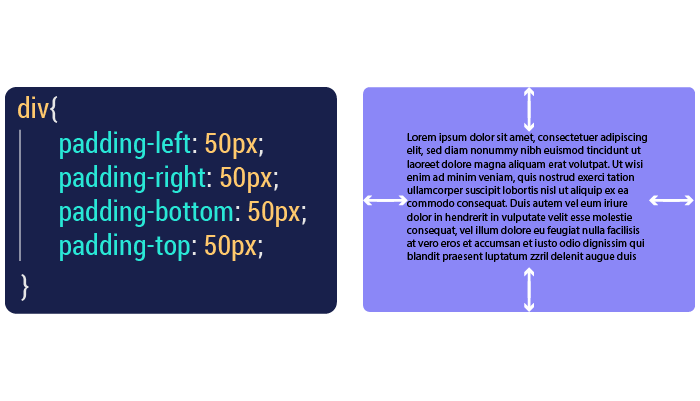
Also, we can determine the dimension of any html element through the width and the height

We’ll get to know them more individually.



Padding

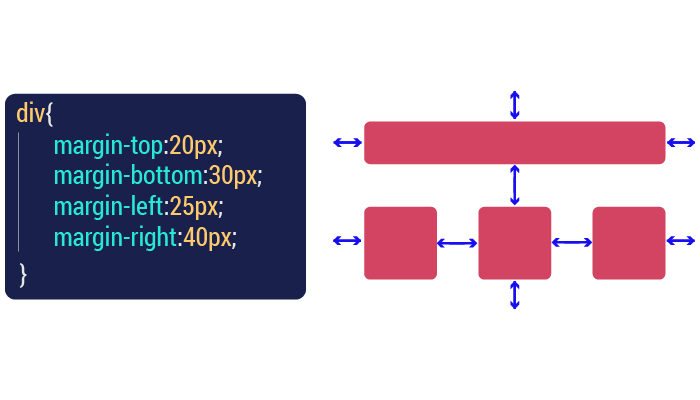
* Use the padding property to create spacing between an element’s content area and border. The padding property applies this spacing in different ways depending on how many values you provide.  
  Padding is the INSIDE of the element.  
  Suppose we have a div with many HTML elements inside. The problem is, we want to move the elements inside without moving the div itself. How can we accomplish that? Well, padding in this situation remains our hero.



# Margin

As a first step, rewind back to the CSS box model scheme to see where the margin is positioned.

Basically, margins allow us to move our tag wherever we want, in any direction: top, bottom, right or left.  
margin is the OUTSIDE of the element  
The margin property is very similar to the padding property, except it allows you to define the spacing around the outside of an HTML element past the border. Like padding, it allows you to define single or multiple values.



Height & Width

As we have seen, every element in HTML is a rectangular box. By the laws of geometry, every rectangle is characterized by width and height.

We can alter these two properties of any HTML content element, using the CSS attribute width or height.

There is one last thing, always keep in mind that:  
Total\_width= width+padding\_right+padding\_left+margin\_right+margin\_left  
To change the value of these properties, just follow the following example:

* HTML

<div>

<p>I want to get bigger</p>

</div>

* CSS

div{

height: 150px;

width: 90%;

}

Border

Like its name implies, the border CSS property sets the border of an element.The border property is a shorthand syntax in CSS that accepts multiple values for drawing a line around the element it is applied to

border: <border-width> || <border-style> || <color>

* border-width: Specifies the thickness of the border.
  + <length>: A numeric value measured in px, em, rem, vh and vw units.
  + thin: The equivalent of 1px
  + medium: The equivalent of 3px
  + thick: The equivalent of 5px
* border-style: Specifies the type of line drawn around the element, including:
  + solid: A solid, continuous line.
  + none (default): No line is drawn.
  + hidden: A line is drawn, but not visible. this can be useful for adding a little extra width to an element without displaying a border.
  + dashed: A line that consists of dashes.
  + dotted: A line that consists of dots.
  + double: Two lines are drawn around the element.
  + groove: Adds a bevel based on the color value in a way that makes the element appear pressed into the document.
  + idge: Similar to groove, but reverses the color values in a way that makes the element appear raised.  
    inset: Adds a split tone to the line that makes the element appear slightly depressed.
  + outset: Similar to inset, but reverses the colors in a way that makes the element appear slightly raised.
* color: Specifies the color of the border and accepts <rgb()>, <rgba()>, <code><hsl()>, <hsla()>, <hex-color>, <named-color>
* css

.box-1 {

border: none;

}

.box-2 {

border: 5px hidden red;

}

.box-3 {

border: 5px solid orange;

}

.box-4 {

border: 5px dashed orange;

}

.box-5 {

border: 5px dotted orange;

}

.box-6 {

border: 5px double orange;

}

.box-7 {

border: 5px groove orange;

}

.box-8 {

border: 5px ridge orange;

}

.box-9 {

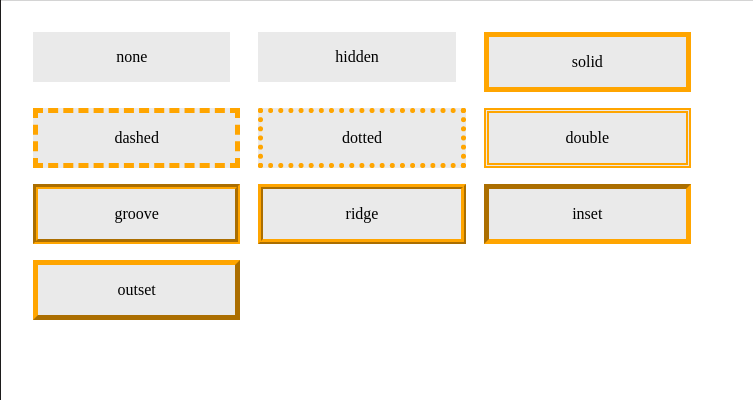
border: 5px inset orange;

}

.box-10 {

border: 5px outset orange;

}

* Output:  
  

Border radius

We can give any element “rounded corners” by applying a border-radius through CSS. You’ll only notice if there is a color change involved. For instance, if the element has a background-color or border that is different than the element, it’s incorrect. Simple examples:

* HTML

<div>

<p>Straight corners</p>

<p class="rounded">Rounded corners</p>

<p class="elliptical">Elliptical corners</p>

</div>

* CSS

p {

border: 10px solid black;

margin: 20px;

}

.rounded {

border-radius: 15px;

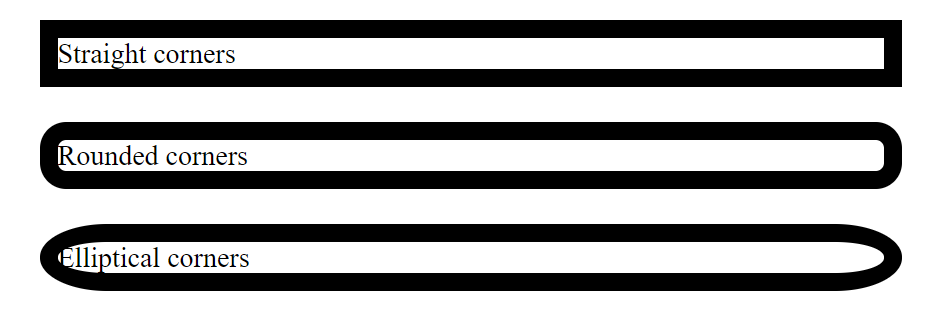
}

.elliptical {

border-radius: 50px / 25px;

}

* Output:



# Font styling:

To style the font, CSS gives us a large set of properties. We’ll be studying a few of them:

**Font-family**

It helps us change the font family of an HTML element, we set one or more fonts for this property .

font-family: Helvetica neue, roboto;

**Color**

It changes the text color, it accepts a named color, hexadecimal value and rgb value.

color: rgb(0, 0, 255);

*/\* same as*

*color: blue;*

*color: #0000ff;*

*color: #00f;*

*color: hsl(0, 100%, 50%); \*/*

**Font-style**

It's used to turn italic text on and off. Possible values are as follows (you'll rarely use this, unless you want to turn some italic styling off for some reason).

font-style: bold;

*/\**

*other possible value*

*font-style:normal;*

*font-style:italic;*

*font-style:oblique;*

*\*/*

**Font-weight**  
It sets how bold the text is. This has many values available in case you have many font variants available (such as -light, -normal, -bold, -extrabold, -black, etc.), but, realistically, you'll rarely use any of them except for normal and bold.

font-weight: normal;

*/\**

*other possible value*

*font-weight:bold;*

*font-weight:bolder;*

*font-weight:lighter;*

*font-weight:100-900*

*\*/*

**Text-decoration**

It sets or unsets text decorations on fonts (you'll mainly use this to unset the default underline on links when styling them.) Available values are:

text-decoration: none;

*/\**

*other possible value*

*text-decoration:underline;*

*text-decoration:overline;*

*text-decoration:line-through;*

*\*/*

# Text layout styles:

With basic font properties out of the way, let's now have a look at properties we can use to affect text layout.

**The text-align**  
It's used to control how a text is aligned within its containing content box.  
The available values are as follows, and work in pretty much the same way as they do in a regular word processor application:  
Left: Left-justifies the text.  
Right: Right-justifies the text.  
Center: Centers the text.  
Justify: Makes the text spread out,

**The line-height**  
It sets the height of each line of text — this includes most length and size units, but can also take a unitless value, which acts as a multiplier and is generally considered the best option — the font-size is multiplied to get the line-height.

**The letter-spacing and the word-spacing**  
They allow you to set the spacing between letters and words in your text. You won't use these very often, but might find a use for them to get a certain look, or to improve the legibility of a particularly dense font. They can take most length and size units.

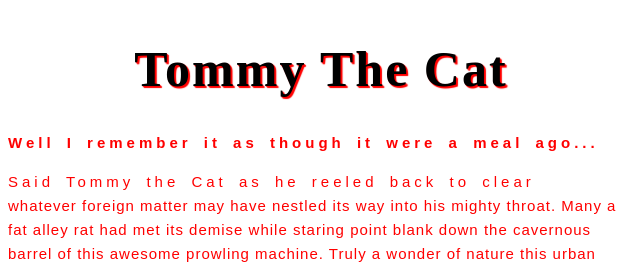
The CSS code would look like:

first-line {

letter-spacing: 4px;

word-spacing: 4px;

}

The code output would be :  


Text Size:

There will be many instances where you will want to change the default size of text elements. The size of your text can be changed using the font-size property. The font-size takes both absolute and relative values. The most common absolute value is px, and the most common relative values are ems and rems.  
em and rem units are both relative measurement values that work similarly to percentages, as they serve as a multiplier in reference to some other unit of measurement.  
In the case of font-size:

* 1 em is equivalent to the font-size of the element’s parent.
* 1 rem is equivalent to the font-size of the root element of the entire HTML document.

h1{

font-size: 50px

}

Css display:

Every element on a web page is a rectangular box. The display property in CSS determines just how that rectangular box behaves.

The CSS display is most useful when :

* we need to align two HTML elements together in the same line.
* we want to display an HTML element as a block.
* we want to hide an HTML element.

In this lesson, we will see 4 types of display: none, inline, block and inline-block.

**Note**: the default display of an element is inline or block.

Display:none

We can hide elements by declaring a display: none value. Another way is to declare visibility: hidden instead of display: none, but there is a difference between them.

To show the difference, let’s hide one of the boxes below:

* The CSS code would be:

#box-2 {

display: none;

width: 100px;

height: 100px;

background: blue;

}

* The code output would be:



* The CSS code would be:

#box-2 {

width: 100px;

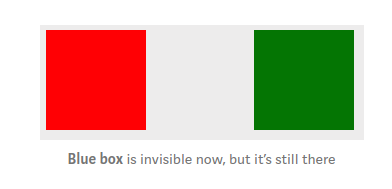
height: 100px;

background: blue;

visibility: hidden;

}

* The output would be :



# Display:inline

Like the name indicates, this feature enables you to make the specified HTML elements in the same line. Let’s understand it with an example.

* The HTML code would be :

<p>One</p>

<p>Two</p>

<p>Three</p>

* The CSS code could be :

p {

display: inline;

}

* The Output code would be :

#### **One Two Three**

Display:block

* HTML elements are divided into default block elements ( automatically force a line-break without the use of <br> ) and inline elements (need <br> to line-break).
* Block-level elements:
  + Take full-width (100% width) by default.
  + Each get displayed in a new line.
  + Width & height properties can be set.
  + Can contain other block or inline elements.

What about forcing an inline element to act like a block?

* The CSS code would be :

div{

display: block;

}

p {

height: 100px;

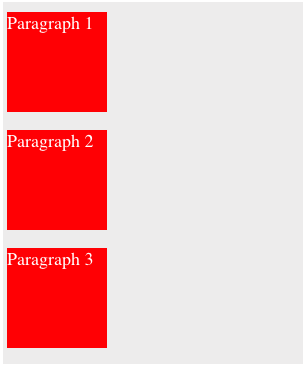
width: 100px;

background: red;

color: white;

}

* The Output code would be :



Display:inline-block

In some cases, both of the display values may not be enough for better web design. In that case, a third display behavior comes to the rescue and makes alignment much easier: display: inline-block.  
As we can understand from its name, display: inline-block declaration shows both the characteristics of inline and block-level elements.  
In other words, we can think of an inline element with width & height properties that can be set, or we can think of a block-level element, that doesn’t have to start with a new line.

* The HTML code would be :

<div class="test"></div>

<div class="test"></div>

* The CSS code would be :

.test {

width: 50px;

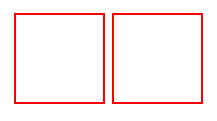
height: 50px;

border: 1px solid red;

display: inline-block;

}

* The Output would be :



# CSS Conclusion:

We have learned a lot of useful information about how to style a web page. Now, we are able to change the style of our text content, change its position and manipulate the elements layout.

CSS is the language for describing the presentation of web pages, including colors, layout, and fonts.

It allows the coder to adapt the presentation to different types of devices, such as large screens, small screens, or printers.

It's independent from HTML and can be used with any XML-based markup language.

The separation of HTML from CSS makes it easier to maintain sites, share style sheets across pages, and tailor pages to suit different environments. This is referred to as the separation of structure or content from presentation.