**CSS | Display Property**

The **Display property** in CSS defines how the components(div, hyperlink, heading, etc) are going to be placed on the web page. As the name suggests, this property is used to define the display of the different parts of a web page.

**Inline and Block Elements**

HTML elements divide into **two major categories**: **block-like** and **inline** elements.

[Block-like elements](https://www.beta-labs.in/2020/06/html-block-elements.html) (<div>, <p>, <h1>, etc.) always stretch out as far to the sides as possible and start on a new line.



[Inline elements](https://betalabsindia.blogspot.com/2020/06/html-inline-elements.html) (<span>, <img>, <a>, etc.) only take the space that is necessary. They don't have to start on a new line.

## The Use of the display Property

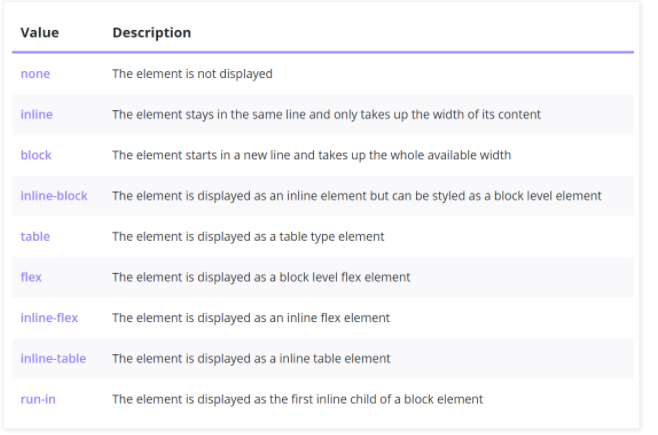
By using the CSS display property, you can specify manually what kind of container the element should use:

The syntax is rather intuitive:

display: value;

In the table below, you can see all the available values. The three most common ones will be discussed separately in the following sections.

#### **Property Values**



### display : inline

Here are a few **characteristics** of elements set to display: inline:

* Elements only take the **necessary space**.
* They also appear **next** to each other on the **same line**.
* One disadvantage is that you can't control the height and width properties of inline elements.
* The display: inline disregards the padding and margin settings.
* Can have only **inline** elements.

It is also possible to make **block elements** appear in one line by setting the display: inline. This example **overrides** the default settings of [<p>](https://www.bitdegree.org/learn/html-li)  and presents them in  **same line**:

p {

  display: inline;

}

The same **overriding** of default settings happens to this [<span>](https://www.bitdegree.org/learn/html-span) element:

span {

  display: block;

}

### display : block

Here are the characteristics of **block** elements:

* Elements take the **largest possible width**.
* Every block element appears in a **new line**.
* Elements **react** to width and height properties.
* Elements can contain both **inline** and **block** elements.

**display: inline-block**

The CSS display: inline-block is a combination of both **inline** and **block-level** features. The **main difference** is that inline-block responds to width and height properties.

This feature makes the CSS display: inline-block more suitable for creating layouts. One of the more popular ways of using inline-block elements is creating horizontal navigation menus.

Here is another example of the use of display: inline-block:

## Hiding Elements: display or visibility

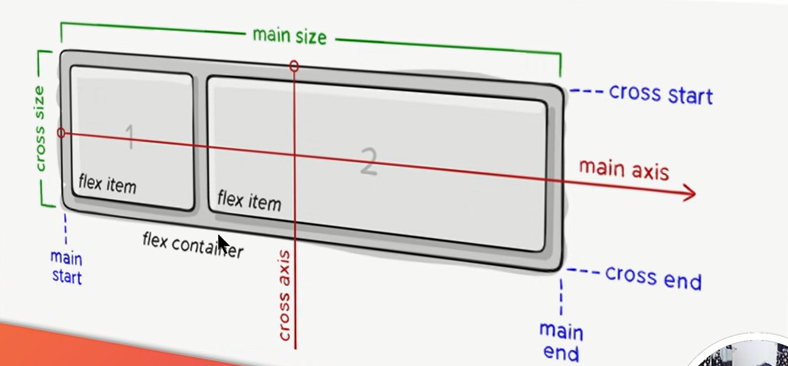
There is a difference in using visibility: hidden and display: none. In the following example, we **hide an element** with the display: none.

The <div> set to display: none completely **disappears** from the page. The next <div> fills its place, leaving **no empty space**.

This is the main difference in display: none vs visibility: hidden. The visibility property keeps the element but makes it **invisible**:

### display : flex

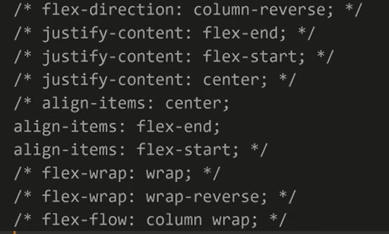
CSS flexbox layout allows you to easily format HTML. Flexbox makes it simple to align items vertically and horizontally using rows and columns. Items will "flex" to different sizes to fill the space. It makes responsive design easier. We can arrange any element to the center of the screen easily with display flex property.



**justify-content** is a property to align items center in the container along the main axis (horizontal) and

**align-items** allows us to align items center along the cross axis (vertically).

Flex – wrap : for responsive behavior .



### display : grid

It is a CSS property that offers a grid-based layout system, with rows and columns, making it easier to design web pages without floats and positioning. We can arrange any element to the center of the screen easily with display grid property.

The **place-items**property in CSS is shorthand for the [align-items](https://css-tricks.com/almanac/properties/a/align-items/) and [justify-items](https://drafts.csswg.org/css-align-3/#propdef-justify-items) properties, combining them into a single declaration.

### CSS | Position Property

In order to make more complex layouts, we need to discuss the position property. It has a bunch of possible values. The **CSS position property** is used to set position for an element. it is also used to place an element behind another and also useful for scripted animation effect.

### CSS Position & Helper Properties

So there are 5 main values of the**Position Property:**

position: static | relative | absolute | fixed | sticky

and additional properties for setting the coordinates of an element (I call them **“helper properties”**):

top | right | bottom | left AND the z-index

**Important Note*:*** Helper properties don’t work without a declared position, or with ***position: static.***

**1. Static**

position: static is the **default value**. Whether we declare it or not, elements are positioned in a normal order on the webpage.

### 2. Relative

### position: relative: An element’s new position relative to its normal position.

**3. Absolute**

In position: relative, the element is positioned **relative to itself.**However, an **absolutely**positioned element is **relative to its parent**.

An element with position: absolute is removed from the normal document flow. It is positioned automatically to the starting point (**top-left corner)** of its parent element. If it doesn’t have any parent elements, then the initial**document <html>** will be its parent.

### 4. Fixed

Like position: absolute, fixed positioned elements are also removed from the normal document flow. The differences are:

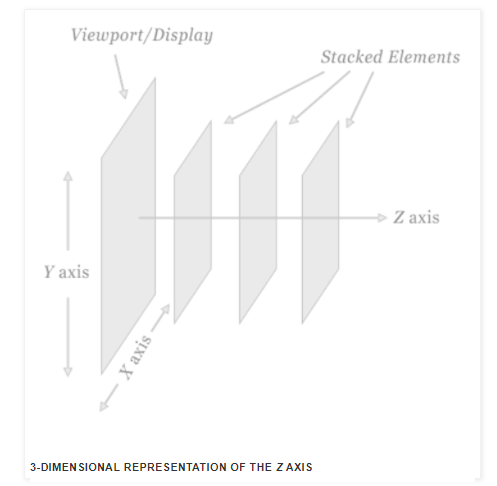
* They are only relative to the <html> document, not any other parents.
* They are not affected by scrolling.

**5. Sticky**

position: sticky can be explained as a mix of position: relative and position: fixed. It behaves until a declared point like position: relative, after that it changes its behavior to position: fixed

**Z-Index**

The z-index property determines the stack level of an HTML element. The “stack level” refers to the element’s position on the *Z-axis* (as opposed to the *X-axis* or *Y-axis*). A higher value means the element will be closer to the top of the stacking order. This stacking order runs perpendicular to the display, or viewport.



n order to clearly demonstrate how z-index works, the image above exaggerates the display of stacked elements in relation to the viewport.

