**Css Selector :**

We can divide CSS selectors into five categories:

* Simple selectors (select elements based on name, id, class)
* [Combinator selectors](https://www.w3schools.com/css/css_combinators.asp) (select elements based on a specific relationship between them)

1. Descendant combinator (space)

The following example selects all <p> elements inside <div> elements:

Example

div p {  
  background-color: yellow;  
}

1. Child combinator (>)

The following example selects all <p> elements that are children of a <div> element:

Example

div > p {  
  background-color: yellow;  
}

1. Next sibling combinator (+)
2. Subsequent-sibling combinator (~)

* [Pseudo-class selectors](https://www.w3schools.com/css/css_pseudo_classes.asp) (select elements based on a certain state)
* [Pseudo-elements selectors](https://www.w3schools.com/css/css_pseudo_elements.asp) (select and style a part of an element)
* [Attribute selectors](https://www.w3schools.com/css/css_attribute_selectors.asp) (select elements based on an attribute or attribute value)

The [attribute="value"] selector is used to select elements with a specified attribute and value.

The following example selects all <a> elements with a target="\_blank" attribute:

Example

a[target="\_blank"] {  
  background-color: yellow;  
}

Simple Selector :

1-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.

.center {  
  text-align: center;  
  color: red;  
}

2 -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.

Ex:

#para1 {  
  text-align: center;  
  color: red;  
}

 specify that only specific HTML elements should be affected by a class. Note:

Example:

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

p.center {  
  text-align: center;  
  color: red;  
}

All CSS Simple Selectors

|  |  |  |
| --- | --- | --- |
| **Selector** | **Example** | **Example description** |
| [#*id*](https://www.w3schools.com/cssref/sel_id.php) | #firstname | Selects the element with id="firstname" |
| [.*class*](https://www.w3schools.com/cssref/sel_class.php) | .intro | Selects all elements with class="intro" |
| [\*](https://www.w3schools.com/cssref/sel_all.php) | \* | Selects all elements |
| [*element*](https://www.w3schools.com/cssref/sel_element.php) | p | Selects all <p> elements |
| [*element,element,..*](https://www.w3schools.com/cssref/sel_element_comma.php) | div, p | Selects all <div> elements and all <p> elements |

**Css Margin**

All CSS Margin Properties

|  |  |
| --- | --- |
| **Property** | **Description** |
| [margin](https://www.w3schools.com/cssref/pr_margin.php) | A shorthand property for setting all the margin properties in one declaration |
| [margin-bottom](https://www.w3schools.com/cssref/pr_margin-bottom.php) | Sets the bottom margin of an element |
| [margin-left](https://www.w3schools.com/cssref/pr_margin-left.php) | Sets the left margin of an element |
| [margin-right](https://www.w3schools.com/cssref/pr_margin-right.php) | Sets the right margin of an element |
| [margin-top](https://www.w3schools.com/cssref/pr_margin-top.php) | Sets the top margin of an element |

**CSS page layout techniques**

CSS page layout techniques  allow us to take elements contained in a web page and control where they're positioned.

[**Normal layout flow**](https://developer.mozilla.org/en-US/docs/Learn_web_development/Core/CSS_layout/Introduction#normal_layout_flow)

Elements on a webpage lay out in **normal flow** if you haven't applied any CSS to change the way they behave.

When used the Normal layout flow ?

For many of the elements on your page, the normal flow will create exactly the layout you need. However, for more complex layouts you will need to alter this default behavior using some of the layout tools available to you in CSS.

[**Overriding normal flow**](https://developer.mozilla.org/en-US/docs/Learn_web_development/Core/CSS_layout/Introduction#overriding_normal_flow)

To [Overriding normal flow](https://developer.mozilla.org/en-US/docs/Learn_web_development/Core/CSS_layout/Introduction#overriding_normal_flow) **we can use one of 3 property :**

1-The [display](https://developer.mozilla.org/en-US/docs/Web/CSS/display) property

* this three value can change how elements behave in normal flow

display:block

display:inline

display:inline-block

* display: grid

this apply the [CSS grid](https://developer.mozilla.org/en-US/docs/Learn_web_development/Core/CSS_layout/Grids) system : A grid will typically have **columns**, **rows**, and then gaps between each row and column.

[Specific layout systems accessed through display](https://developer.mozilla.org/en-US/docs/Learn_web_development/Core/CSS_layout/Introduction#specific_layout_systems_accessed_through_display)

We also have entire layout methods that are enabled via specific display values. The most important ones for you to know about are [CSS grid](https://developer.mozilla.org/en-US/docs/Learn_web_development/Core/CSS_layout/Grids) and [Flexbox](https://developer.mozilla.org/en-US/docs/Learn_web_development/Core/CSS_layout/Flexbox), which both alter how child elements are laid out inside their parents.

2- [Floats](https://developer.mozilla.org/en-US/docs/Learn_web_development/Core/CSS_layout/Introduction#floats)

Applying a [float](https://developer.mozilla.org/en-US/docs/Web/CSS/float) value such as left can cause block-level elements to wrap along one side of an element, like the way images sometimes have text floating around them in magazine layouts.

3- [Positioning](https://developer.mozilla.org/en-US/docs/Learn_web_development/Core/CSS_layout/Introduction#positioning)

The [position](https://developer.mozilla.org/en-US/docs/Web/CSS/position) property allows you to precisely control the placement of boxes inside other boxes. static positioning is the default in normal flow, but you can cause elements to be laid out differently using other values, for example, fixing them to the top of the browser viewport using position: fixed.

**Css position**

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

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

There are five different position values:

* static
* relative
* fixed
* absolute
* sticky

1- 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

2-position: relative

An element with position: relative; is positioned relative to its normal position.

3-position: fixed;

An element; 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.

4- position: absolute

An element with position: absolute; is positioned relative to the nearest positioned ancestor (instead of positioned relative to the viewport, like fixed).

Ex div inside div the inner if the inner div position as absolute the inner div eill position relative to the outer div

A screenshot of a computer

AI-generated content may be incorrect.

div.relative {

position: relative;

width: 400px;

height: 200px;//dd

border: 3px solid #73AD21;

}

div.absolute {

position: absolute;

top: 80px;

right: 0;

width: 200px;

height: 100px;

border: 3px solid #73AD21;

}

</style>

</head>

<body>

<h2>position: absolute;</h2>

<p>An element with position: absolute; is positioned relative to the nearest positioned ancestor (instead of positioned relative to the viewport, like fixed):</p>

<div class="relative">This div element has position: relative;

<div class="absolute">This div element has position: absolute;</div>

</div> inner div position top 80 px according from the top of the outer div top ets ….

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.

**5-** **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 :

Sticky

Like the relative that it position relative to it is flow page position

Like fixed that it stuck when the user scroll the page

Deferant from fixed that the fixed position relative the viewport but the sticky positioned relative to the normal position of element

**Ex :**

**First use position:sticky**

[**https://www.w3schools.com/css/tryit.asp?filename=trycss\_position\_sticky**](https://www.w3schools.com/css/tryit.asp?filename=trycss_position_sticky)

**second use position :fixed**

[**https://www.w3schools.com/css/tryit.asp?filename=trycss\_position\_sticky**](https://www.w3schools.com/css/tryit.asp?filename=trycss_position_sticky)

**what the deference?**

Top:10 px

In fixed the div element will positioned 10 px from top the viewport so it will appear over the first <p> element

In sticky the element will positioned 10 px from the normal position

The normal position of div element here is bello the first <p>element so it will positioned 10 px top so the first <p> will apper and bello of it the div .

FlexBox

Notes:

CSS Layout – display:

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

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.

<https://appbrewery.github.io/flex-layout/>

<https://css-tricks.com/snippets/css/a-guide-to-flexbox/>

متلى نستخدم خاصية flex

Ex:

Flex:1

| **لما تستخدم Flex على...** | **الأفضل تعمل** | **السبب** |
| --- | --- | --- |
| صف (row) | flex لتقسيم **العرض** | لأن Flexbox يشتغل بشكل مثالي أفقيًا |
| عمود (column) | استخدم height أو min-height بدل flex | عشان تتحكم فعليًا في الطول، خصوصًا إذا ما فيه محتوى |

**🔧 في عمود طيب متى نستخدم Flex ؟**

ممكن، لكن لازم تكون:

* محدد height على العنصر الأب (container).
* وتكون العناصر تشتغل داخل تلك المساحة.

مثلًا:

css

CopyEdit

.container {

display: flex;

flex-direction: column;

height: 100vh; /\* لازم أحدد الارتفاع \*/

}

.header, .content, .footer {

flex: 1; /\* يقسموا المساحة العمودية بالتساوي \*/

}

✅ هذا مفيد لو بدك تقسيم الشاشة لنسب مثل: 20% Header, 60% Content, 20% Footer.