ESPE Armed Forces University

Information Technology Engineering

Fundamentals of web systems

Moreno Quiñonez Douglas Dalyn

Delivery Date: January 06, 2023

Sangolqui – Ecuador

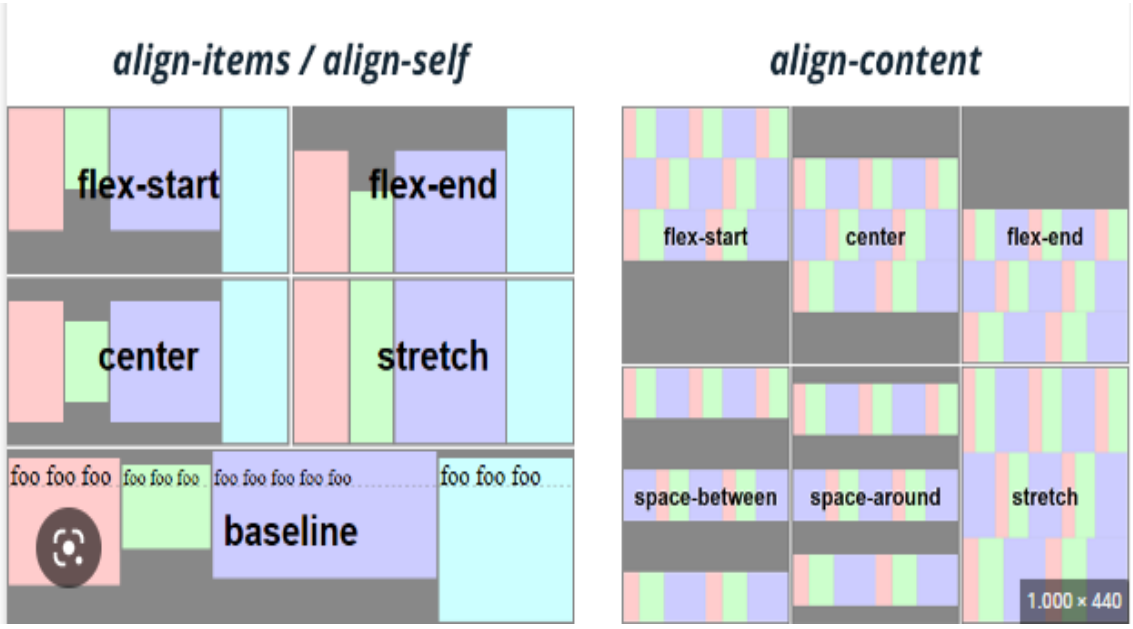
FLEXBOX

In the early days of CSS it was very common to see the float property in style.css files and although this property was created to specify whether an element should break out of the normal flow and be placed to the left or right of the content and surrounding text to be docked or not to said element.

Flexbox is a CSS3 layout module that was created to improve the way responsive design is done, thus avoiding the use of floats, writing less code, and making it easier to position elements, even if you don't know their size. Basically the idea of Flexbox is to be able to alter the width, height and positioning of elements in the best way with the space we have.

✓ STRUCTURE

The Flexbox structure is made up of parent and child containers (Flex-Container and Flex - Elements respectively) .



- ***The Flex-Container*** is our parent container, and it is he who will contain all the child elements that we want to position.

- ***The Flex-Element*** is our child element, which will use all the available space to locate itself according to the properties to which it is submitted together with the other contained Flex-Elements.

✓ PROPERTIES

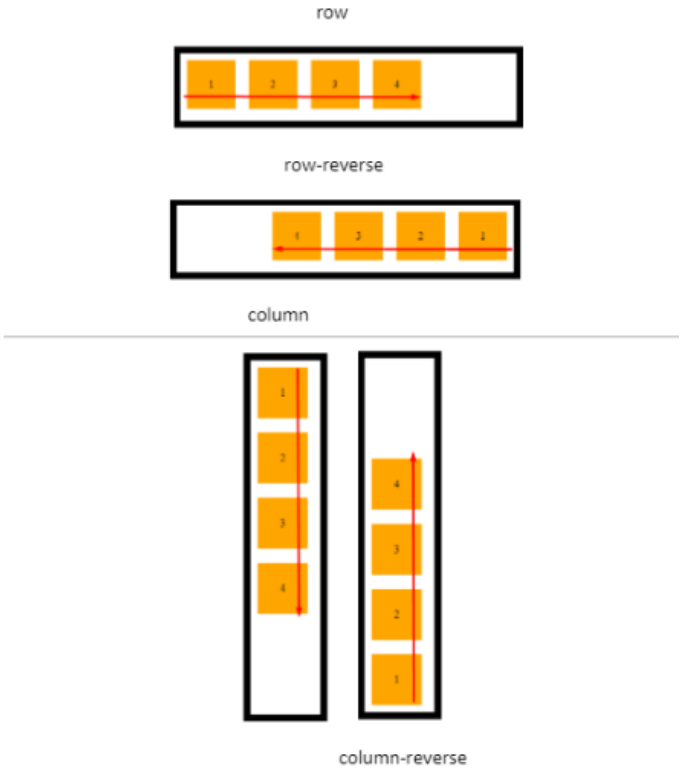
If we want to use the Flexbox properties, we will have to define it through the display property and its flex or inline-flex value within the selector that we define, which is our parent element or Flex-Container .

```
.contenedor-padre {
    display: flex
}
```

✓ FLEX-DIRECTION

It is in charge of defining the main and secondary axis of our child elements. These axes, as already mentioned, can be vertical, forming columns, and horizontal, forming rows.

```
.contenedor-padre {
    display: flex ;
    flex-direction: row | row-reverse | column | column-reverse ;
}
```



✓ FLEX-WRAP

By default, flex tries to arrange the elements in the same line, if that is not the case you need, with flex-wrap you can order the elements in more than one row or column.

```
.contenedor-padre {

    display: flex ;

    flex-wrap: nowrap | wrap | wrap-reverse ;

}
```

✓ JUSTIFY-CONTENT

When we name the main axis of a parent element, that is, the horizontal axis, this property is highlighted to give direction to the child elements.

```
.contenedor-padre {

   display: flex ;

   justify-content: stretch | flex-start | flex-end | center | space-between | space-around | space-evenly ;

}
```

✓ ALIGN-ITEMS

Like justify-content this property allows you to distribute the elements on an axis, but this time on the vertical axis.

```
.contenedor-padre {

    display: flex ;

    align-items: flex-start | flex-end | center | stretch | baseline ;

}
```

EXAMPLES FLEXBOX

HTML

```
<body>

<div class="container">

   <header>
      <nav>
        <ul>
          <li><a href="#"><h1>LOGO</h1></a></li>
          <li><a href="#">Link</a></li>
          <li><a href="#">Link</a></li>
          <li><a href="#">Link</a></li>
          <li><a href="#">Link</a></li>
        </ul>
      </nav>
      <button>Button</button>
   </header>


      <section class="main">
         <h2>Main Content</h2>
      </section>

      <aside class="sidebar">
         <h3>Sidebar</h3>
      </aside>

   <footer>
      <h3>Footer</h3>
      <p>Diseñado con cariño ;) | SiloCreativo.2017</p>
   </footer>

</div>

</body>
```

CSS

```
* {
   margin: 0;
   padding: 0;
   box-sizing: border-box;
}

a {
   text-decoration: none;
}

header {
  height: 150px;
  background-color: #FF4945;
}

.main {
```

```
   height: 300px;
  color: white;
   background-color: #FFA931;
}

.sidebar {
   height: 300px;
  color: white;
    background-color: #9B0579;
}

footer {
   height: 100px;
   color: white;
   background-color: #00778F;
}
.container {

display: flex;

flex-direction: column;

}
.wrapper {
    display: flex;
    flex-direction: row;
}
.main {
    flex: 4;
    margin-right: 40px;
}
.sidebar {
   flex: 1;
}
```

CSS GRID LAYOUT

CSS Grid Layout is a 2-dimensional grid system, created within the CSS language. It's a standard, which means you don't need anything for the browser to understand it. To put it into practice you need to simply apply the new "display: grid" and start using its powerful CSS properties and values. With this you will be able to control every imaginable aspect of the grid, getting practically everything you can think of in a comfortable way.

Currently all browsers support CSS Grid Layout, so we recommend using it. Applying the grid system is a true wonder, allowing you to do things with very little effort that used to require a lot of CSS code, or that were directly impossible to achieve, at least with such versatility.

✓ GRID SYSTEMS IN CSS

CSS Grid Layout is a CSS layout model based on a grid, something that might not seem so new, considering that several CSS libraries have already tried it. Systems like the 960 Grid System were pioneers in creating a CSS code base so that designers could position elements in a row and column layout. Bootstrap itself includes among other things a grid system.

✓ WHERE GRID SYSTEMS CAN BE APPLIED

Another typical question that we ask ourselves when introducing ourselves to grid systems and specifically with the new CSS standard is where can I apply it? Really, you have to think of it as CSS and therefore you can apply it everywhere you currently apply CSS code.

The grid system is not something that you can use only in static sites, in programming with a certain language, in a framework or in a type of devices. You can use it on all those sites, and it is applicable to the web for desktop computers, for devices with small screens and for the development of webapps, like the ones you do with Ionic.

COMPARISON WITH FLEXBOX

There is another standard for layout with great possibilities, called Flexbox . Perhaps you've started using Flexbox as it's been with us the longest. Grid Layout goes one step further, since it applies to two dimensions while Flexbox applies to only one.
With Flexbox you could only define what the items had to do on the horizontal axis or on the vertical axis. When the space ran out on the vertical or horizontal axis, then the items were placed depending on the configuration of the Flexbox attributes. For example, in the case of using flex-wrap, when an item arrived that did not fit horizontally, it went to the next row. But you don't control those two

rows, the items themselves are the ones that are arranged.

The difference in CSS Grid is that you actually have two dimensions. You can decide where you want your boxes to be placed, both horizontally and vertically, perfectly coordinating the dimensions of each row or column, globally or perfectly independently. You don't need to do tricks, or force them to run out of space, you really have detailed control of the position and dimensions of each element on each axis.

EXAMPLES GRID LAYOUT

```
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}

a {
    text-decoration: none;
}

header {
   height: 150px;
   background-color: #FF4945;
}

.main {
   height: 300px;
  color: white;
   background-color: #FFA931;
}

.sidebar {
   height: 300px;
  color: white;
    background-color: #9B0579;
}
footer {
   height: 100px;
   color: white;
   background-color: #00778F;
}
header {
    grid-area: header;
}
```

```
.main {
    grid-area: main;

.sidebar {
    grid-area: sidebar;
}
footer {
    grid-area: footer;
}
.container {
    display: grid;
    grid-template-columns: 4fr 1fr;
    grid-template-areas:
        "header header"
        "main sidebar"
        "footer footer";
    grid-gap: 40px;
}
header{
    display: grid;
    grid-template-columns: 1fr 1fr;
}
header nav {
    justify-self: start;
}
header button {
    justify-self: end;
}
```

BIBLIOGRAFIAS

- ✓ Serrano, A. (2021, 28 junio). *Flexbox vs CSS Grid: Un ejemplo práctico*. Silo Creativo. https://www.silocreativo.com/flexbox-vs-css-grid-ejemplo-practico/
- ✓ Alura Latam. (2021, 2 septiembre). *Flexbox CSS: Guia Completo, Elementos y Ejemplos*. Alura. https://www.aluracursos.com/blog/flexbox-css-guia-completo-elementos-y-ejemplos