CSS Flex Property

September 9th, 2019

#css #flex

Table of Contents

CSS Flex

This value for display property (`display: flex`) defines a block-level flex container. The flex container makes it easy to control your elements on bigger and smaller screens. Hence, it provides ease in responsiveness.

As the name 'flex' implies, it renders your elements flexible. Meaning, they can adjust themselves (somethings irrespective of a set width or height) to fit in the container.

Let's take an example

```
<!-- index.html -->
<link rel="stylesheet" href="style.css" type="text/css" />
<div class="flexContainer1">
  <div></div>
```

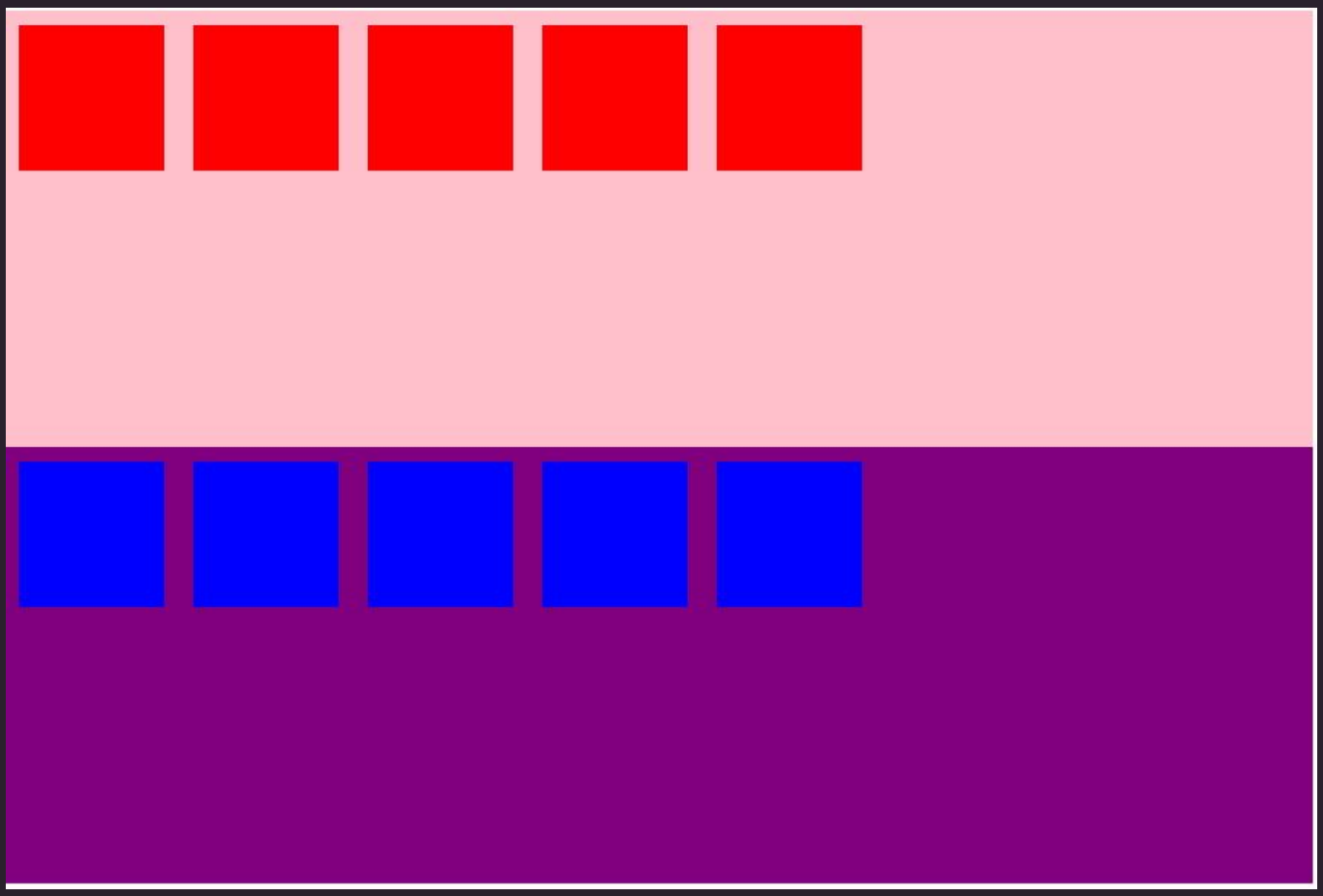
```
  <div></div>
  <div></div>
  <div></div>
  <div></div>
</div>

<div class="flexContainer2">
  <div></div>
  <div></div>
  <div></div>
  <div></div>
  <div></div>
</div>
```

```
/* style.css */
.flexContainer1,
.flexContainer2 {
  display: flex;
  height: 300px;
  width: 900px;
}
.flexContainer1 {
  background-color: pink;
}
.flexContainer2 {
  background-color: purple;
}
.flexContainer1 div,
.flexContainer2 div {
  height: 100px;
  width: 100px;
```

```
    margin: 10px;
}
.flexContainer1 div {
    background-color: red;
}
.flexContainer2 div {
    background-color: blue;
}
```

The above css file renders our divs - `flexContainer1` and `flexContainer2` div as a flex container by `display: flex`

Our result for the above program is,



Before continuing with our program, I'd like to state the properties of flex containers

- flex-direction
- justify-content

- align-items
- align-content
- flex-wrap
- flex-flow

> These properties are used only in flex containers, i.e `display: flex` is required.

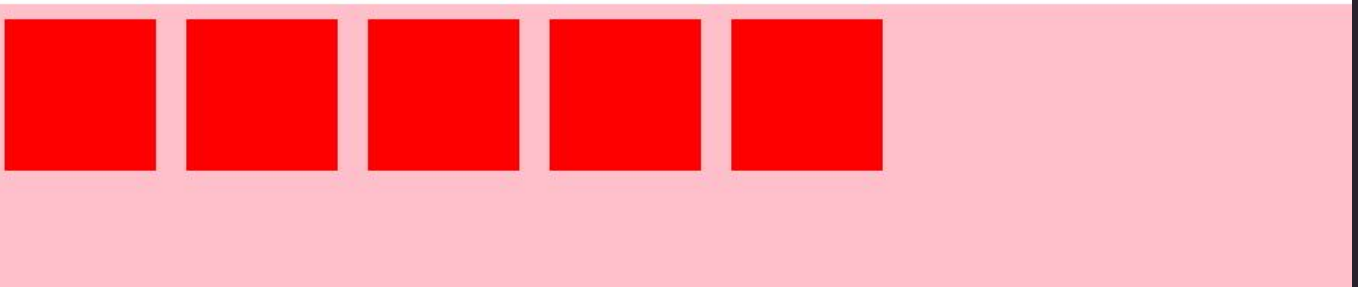
`flex-direction`

This specifies the path along which the elements would be flexible. The values available for this property are `row, column, row-reverse, column-reverse` of which the default is `row`.

Adding to our CSS stylesheet from above, if we had

```
.flexContainer1 {
    display: flex;
    flex-direction: row;
}
.flexContainer2 {
    display: flex;
    flex-direction: column;
```

Our result,



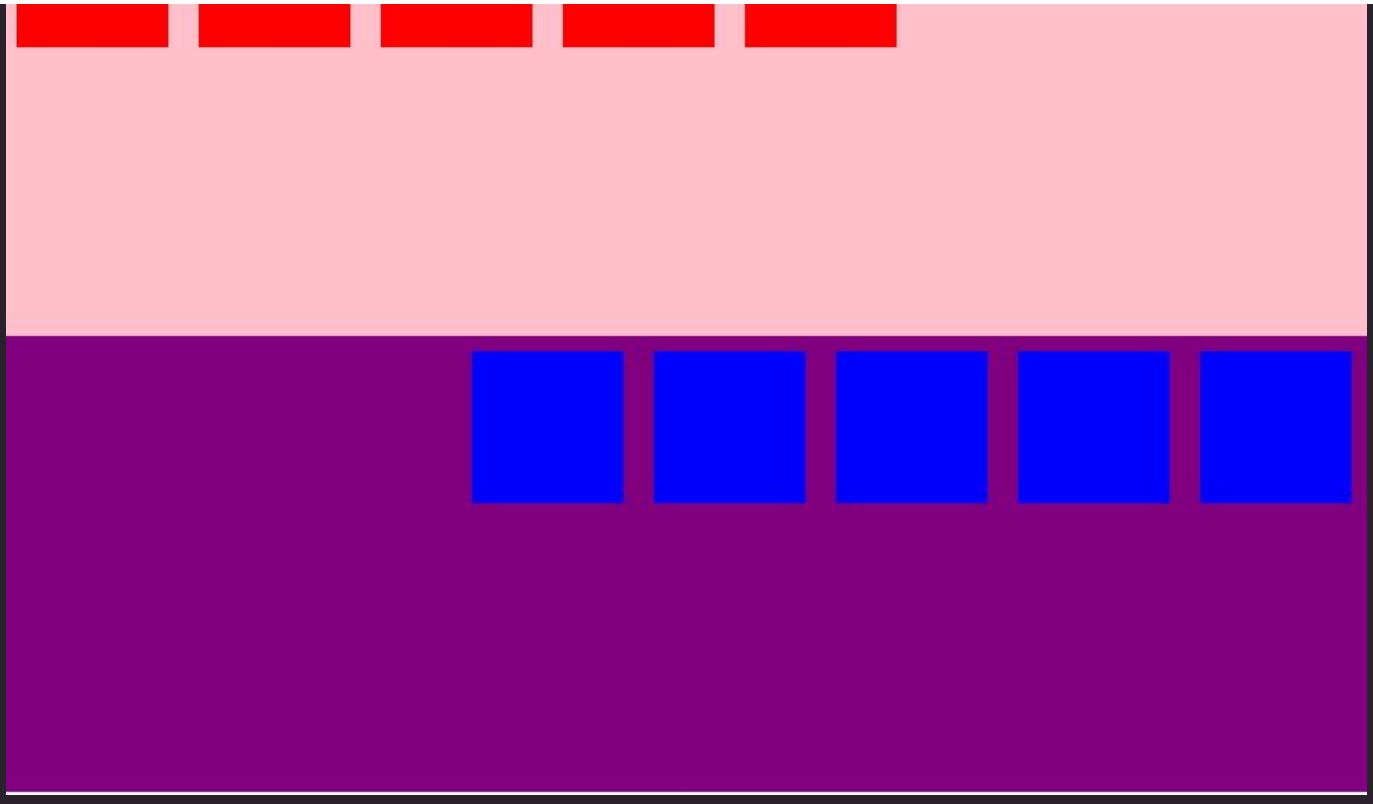
> Notice that the divs at flexContainer2 lost a little bit of their height? We'll discuss that at `flex-wrap`

I guess this result is self-explanatory as the `flex-direction determines the path along which the elements should be flexible.`
The remaining two values - `row-reverse and column-reverse` do the same thing with these values but with a difference. For row-reverse, the container stacks the elements from right to left instead. For column-reverse, the container stacks the elements from bottom to top.

Let's take `row-reverse` for example. When we change the flex-direction of the .flexContainer2, such that

```
.flexContainer2 {
  display: flex;
  flex-direction: row-reverse;
}
```

We have,



flex-wrap

As seen in the above display, the elements align well while still retaining their specified properties (specifically width and height). Let's try increasing the values of the width.

```
.flexContainer1 div,
.flexContainer2 div {
  width: 600px;
  height: 100px;
}
```

Result,

What do you notice?

I used 600px so my point is well explained. Obviously, the boxes are not 600px wide.

The elements have lost the specified width values. Remember I said `flexible`?
That was the same thing that happened in the `flex-direction: column` diagram above.

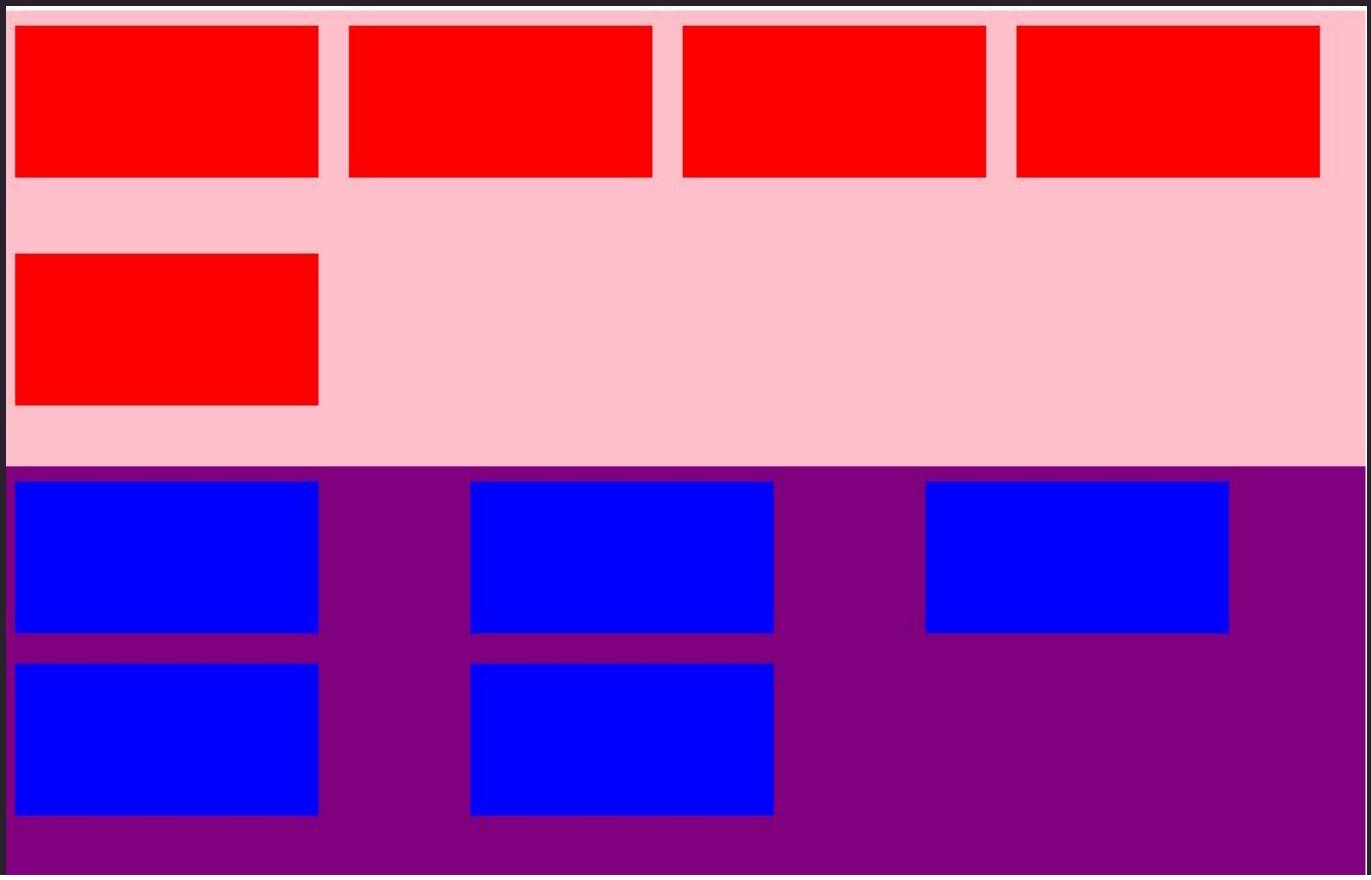
The flex container tries to squeeze the elements in to ensure that they are still in the container.
This is where `flex-wrap` comes in. The values available for this property are `wrap, nowrap and wrap-reverse`

`wrap` simply informs the container to wrap the elements when needed. The container no longer tries to squeeze in the elements. As soon as elements have occupied the width space (or height space for `column` direction), the elements wrap to the next row (or column). Let's apply these values,

```
.flexContainer1,
```

```css
.flexContainer2 {
  display: flex;
  width: 900px;
  height: 300px;
  flex-wrap: wrap;
}
.flexContainer2 {
  flex-direction: column;
}
.flexContainer1 div,
.flexContainer2 div {
  width: 200px;
  height: 200px;
}
```

Note, that the other values are still present from our very first program, I only added this lines to it.



This is our result.
`nowrap` simply tells the container not to wrap the elements. This is the default value for `flex-wrap` property.
`wrap-reverse` wraps the elements where necessary but in reverse order.

`flex-flow`

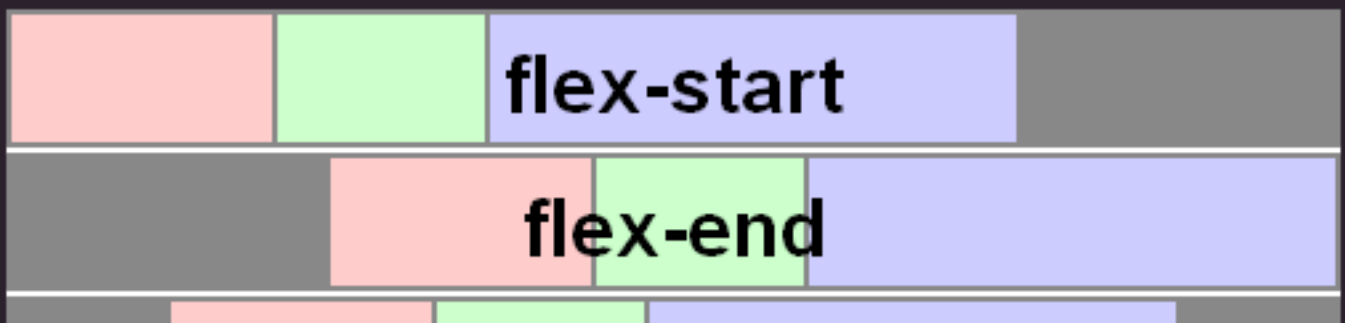
This is simply a short hand property for `flex-direction` and `flex-wrap`. An example is

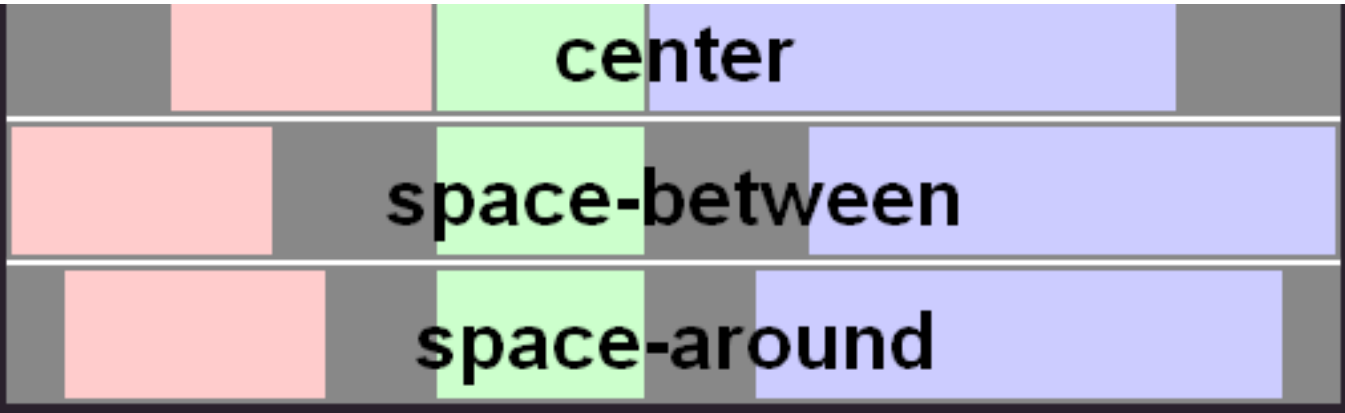
```
.flexContainer1 {
  display: flex;
  flex-flow: column wrap;
}
```

`justify-content`

This property is used to align the elements in a container along the horizontal direction.

> The horizontal direction here refers to `flex-direction: row`. When the direction becomes column, `justify-content` aligns the elements along the vertical direction.





The available values are `flex-start, flex-end, center, space-between, space-evenly, space-around`.

`flex-start` – which is the default, aligns the element at the beginning of the container.

`flex-end` – aligns the element at the end of the container

`center` – aligns the element at the center of the container

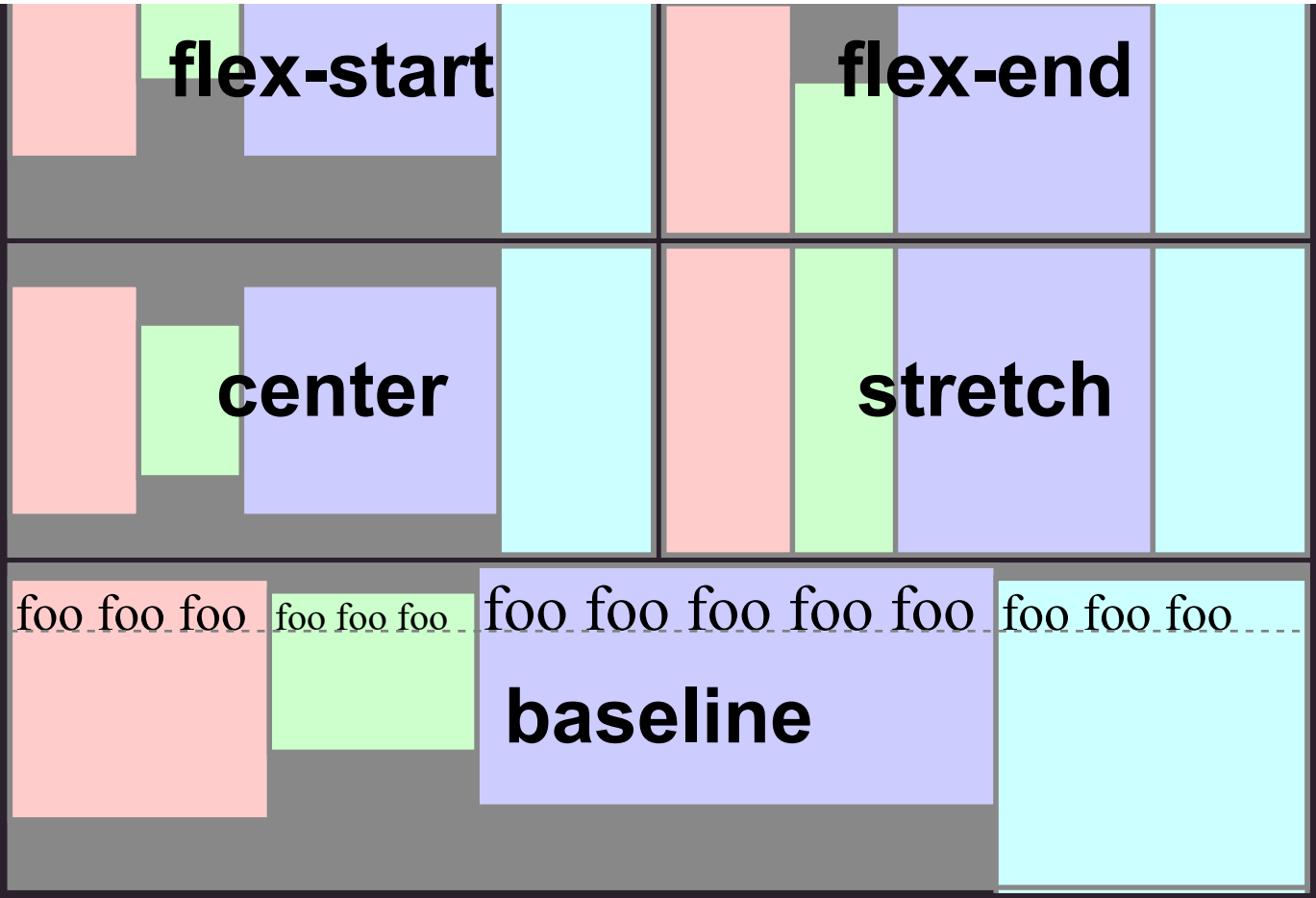
`space-evenly` – aligns the element such that the spaces between them and the container are even

`space-around` – similar to `space-evenly` but the spaces are not exactly even

`space-between` – somewhat similar to `space-around` but the spaces are only between the elements. There are no spaces in respect to the container.d

`align-items`

This property is similar to justify-content in some ways. It is used to align the elements along the opposite direction of justify-content i.e By default (`flex-direction: row`), `align-items` aligns the elements along the vertical direction. When the flex-direction changes to column, `align-items` uses the horizontal direction for alignment.



But,

`align-items` property has it's own values which are `flex-start, flex-end, center, stretch and baseline`

`flex-start` – just as with justify-content, it starts alignment from the beginning of the container.

`flex-end` – as you may have rightly guessed, starts it's alignment from the end of the container.

`center` – aligns the elements at the center

`stretch` – which is the default, stretches the height (or width) of the elements to occupy the available spaces in the container.

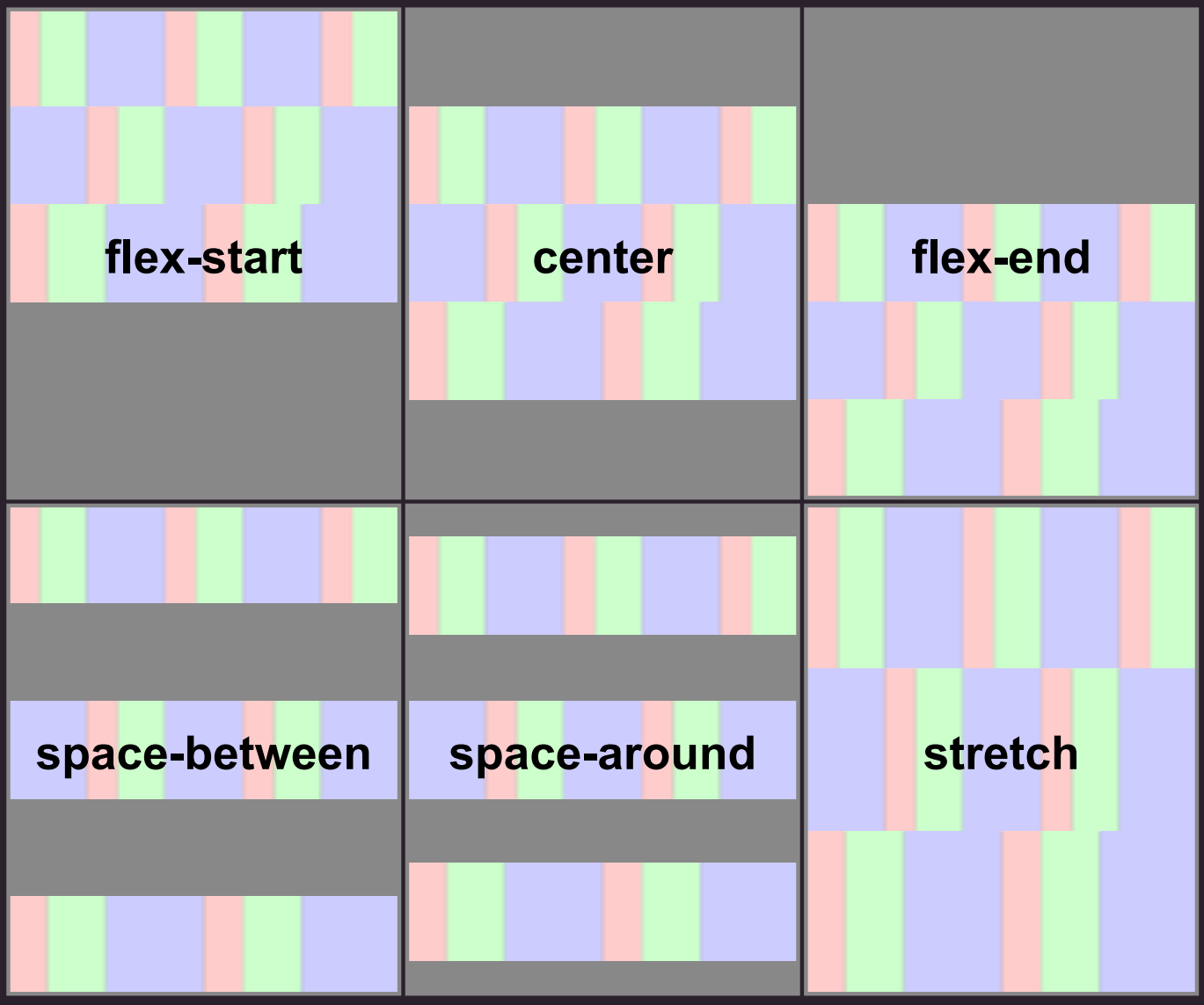
`baseline` – aligns the elements such that their baseline aligns.

> `justify-content` and `align-items` sometimes brings confusion.

For many developers, it's often hard using them effectively on a first try. I wrote an article here – JUSTIFY-CONTENT & ALIGN-ITEMS at your first try 😊 that explains the difference clearly.

`align-content`

This property is a lot more like `align-items` but instead of aligning the elements, it aligns the flex lines around the elements.



All properties of `align-items` are applied here but, much attention is given to the flex lines as seen in the diagram above.

Conclusion

With CSS Flex, you can make any website responsive – causing

elements to wrap where necessary or reduce their sizes in smaller screens.

Thanks for reading 😀

I also have a Frontend Development Series on Dev.to – Frontend Development Series – Dillion Megida

You could reach out to me on twitter @iamdillion.

Kindly ask questions or make contributions in the comment section below.

Connect with me ✨