Flex-Box

By: Ahmed Mahdy

Contents

1. Introduction
2. What Is Flex-Box ?
3. Flex-Box Architecture
4. Flex-Box Container

Introduction

...

Before the invention of CSS Flexbox, web designers were limited in their ability to control the position and size of elements on a page.

They had to use a combination of:

- `float`s,
- `position`ing,
- and other hacks to create the layouts they wanted.

But then, in 2017, a new technology was introduced to the world of web design - **CSS Flexbox**.

Introduction

...

With Flexbox, designers were finally able to easily control the alignment, direction, and distribution of elements within a container. They could create flexible and responsive designs with ease, and make their websites look great on any device.

"And so, Flexbox became a game changer in the web design industry, and is now an essential tool for any modern web designer."

What is Flex-Box ?

The main idea behind the flex layout is to give the container the ability to alter its items' width/height (and order) to best fill the available space (mostly to accommodate to all kind of display devices and screen sizes). A flex container expands items to fill available free space or shrinks them to prevent overflow.

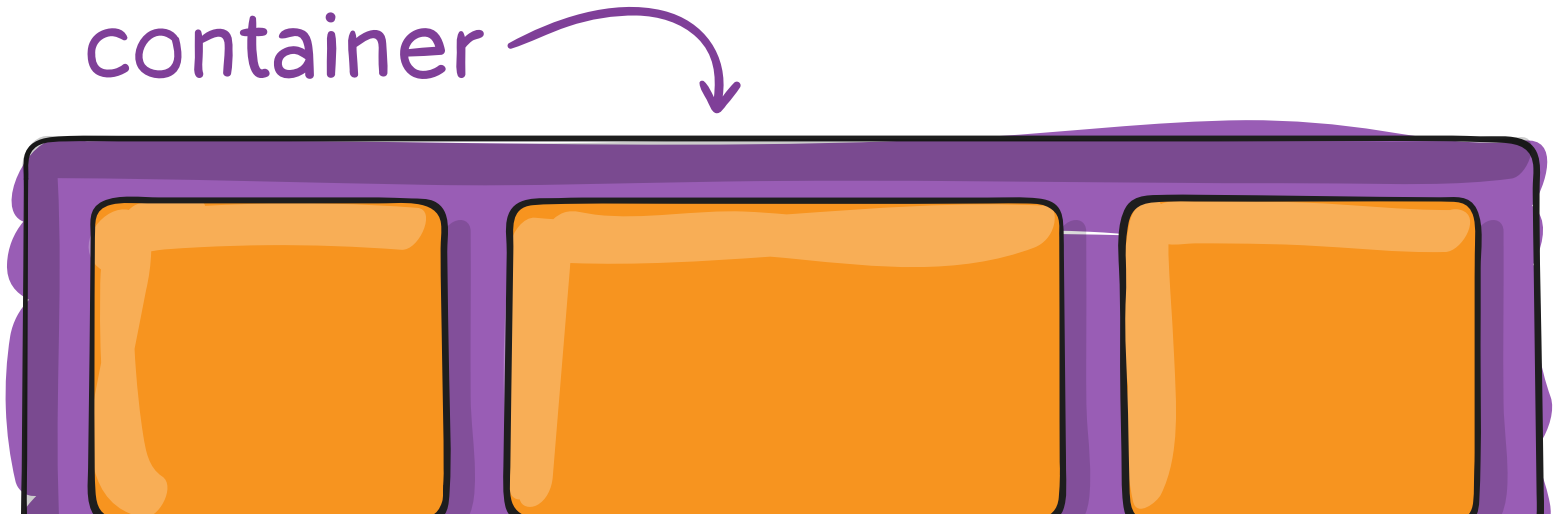
Most importantly, the flexbox layout is direction-agnostic as opposed to the regular layouts (block which is vertically-based and inline which is horizontally-based). While those work well for pages, they lack flexibility to support large or complex applications (especially when it comes to orientation changing, resizing, stretching, shrinking, etc.).

Note: Flexbox layout is most appropriate to the components of an application, and small-scale layouts, while the Grid layout is intended for larger scale layouts.

Flex-Box Architecture

The structure of a flexible layout is based on dividing elements to:

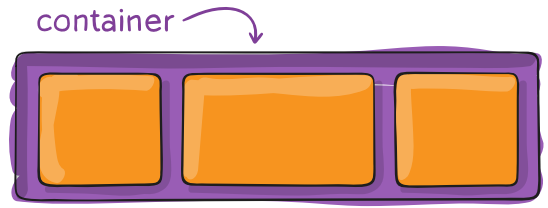
- containers (parent elements)
- and items (child elements).
- **containers (parent elements)**
- and items (child elements).



Flex-Box Container

To define a flex container for element that has class `.flex-container`; inline or block depending on the given value. It enables a flex context for all its direct children.

```
.flex-container{
    display: flex; /* or inline-flex */
}
```



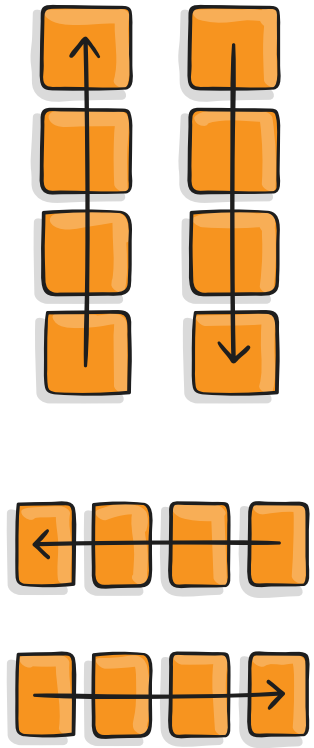
The CSS flexbox can control the following aspects of the layout:

- The direction to which the items are displayed
- The wrapping of items when the window is resized
- The Justification of items and space in between
- The vertical alignment of items
- The alignment of lines of items
- The order of items in a line
- The items' ability to grow or shrink when the window resizes
- The alignment of an individual item

Flex-Direction

This property establishes the main-axis, thus defining the direction flex items are placed in the flex container. Flexbox is (aside from optional wrapping) a single-direction layout concept. Think of flex items as primarily laying out either in horizontal rows or vertical columns.

```
.container {
  display: flex;
  flex-direction: row;
}
```



- `row` (default): left to right in ltr; right to left in rtl
- `row-reverse`: right to left in ltr; left to right in rtl
- `column`: same as row but top to bottom
- `column`-reverse: same as row-reverse but

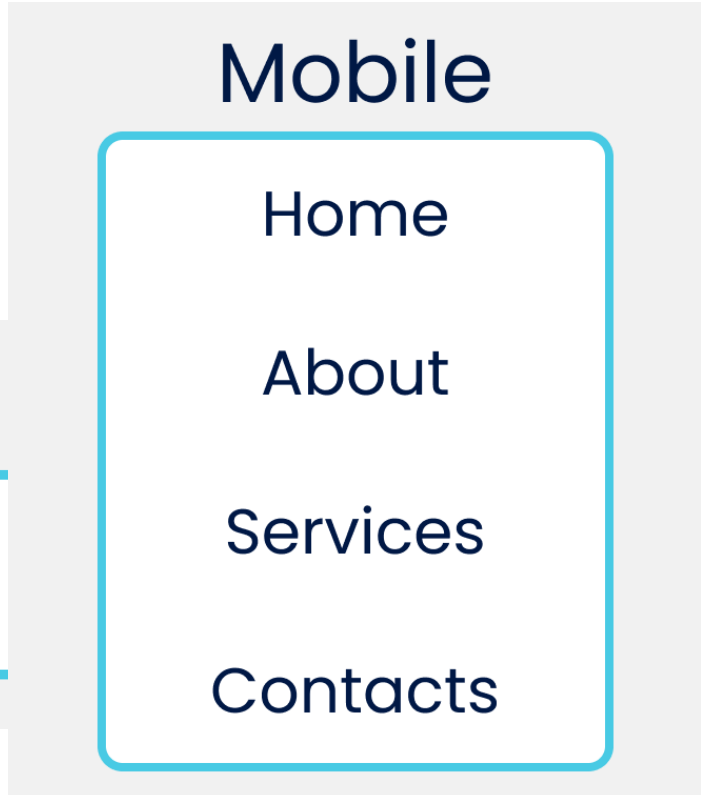
Flex-Direction

Real life Examples

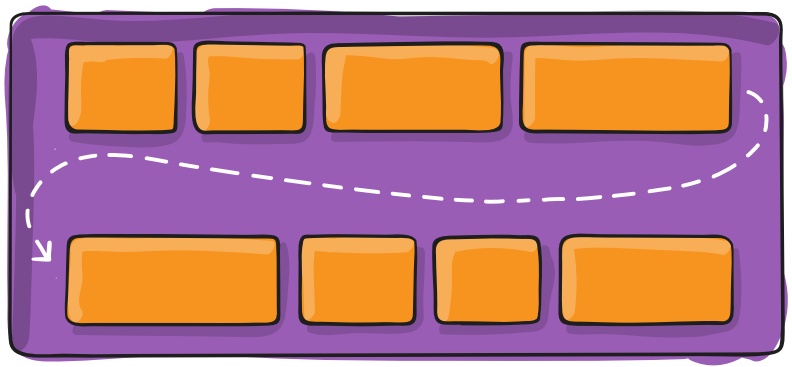
If we want to create responsive navigation bar we can use:

- `flex-direction: row;` in desktop case:

- `flex-direction: column;` in mobile case:



Flex-Wrap



By default, flex items will all try to fit onto one line. You can change that and allow the items to wrap as needed with this property.

```
.container {
  display: flex;
  flex-wrap: nowrap;
}
```

- nowrap (default): all flex items will be on one line
- wrap: flex items will wrap onto multiple lines, from top to bottom.
- wrap-reverse: flex items will wrap onto multiple lines from bottom to top.

Flex-Flow

This is a shorthand for the `flex-direction` and `flex-wrap` properties, which together define the flex container's main and cross axes. The default value is `row nowrap`.

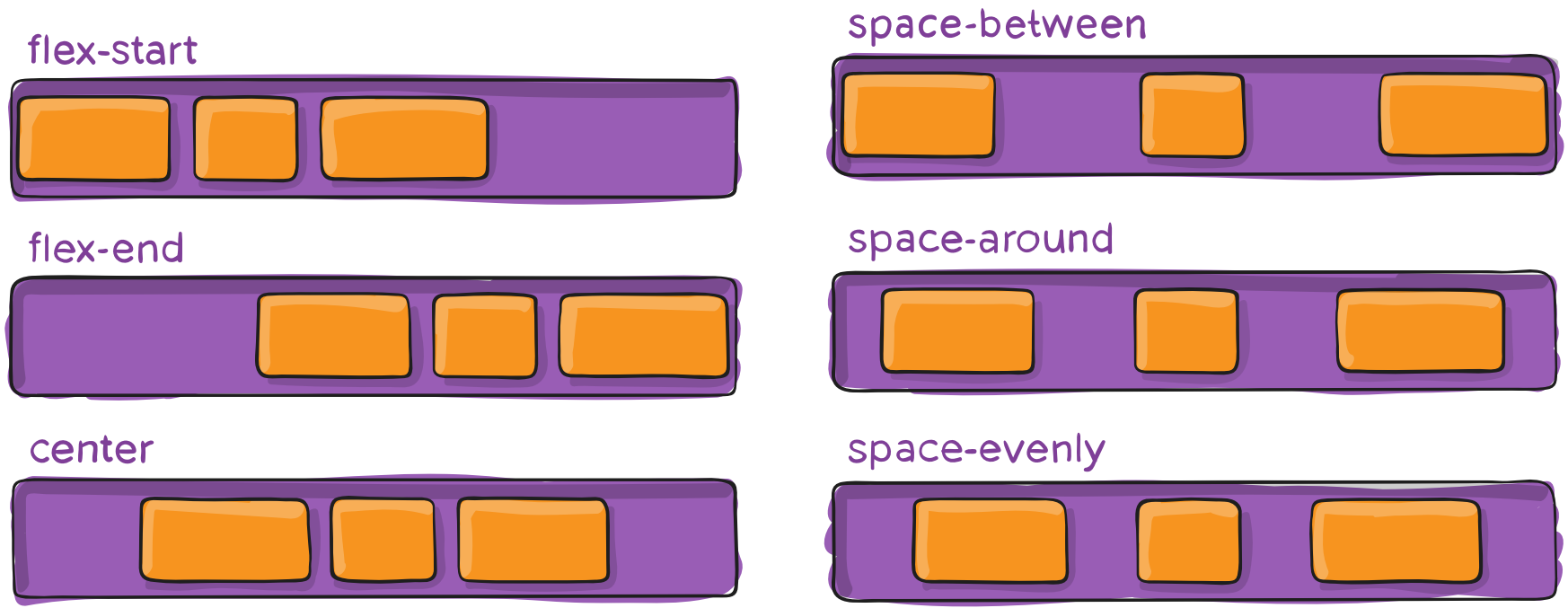
```
.container {
  display: flex;
  flex-flow: column wrap;
}
```

this will equals:

```
.container {
  display: flex;
  flex-direction: column;
  flex-wrap: wrap;
}
```

Justify-Content

This defines the alignment along the main axis. It helps distribute extra free space leftover when either all the flex items on a line are inflexible, or are flexible but have reached their maximum size. It also exerts some control over the alignment of items when they overflow the line.



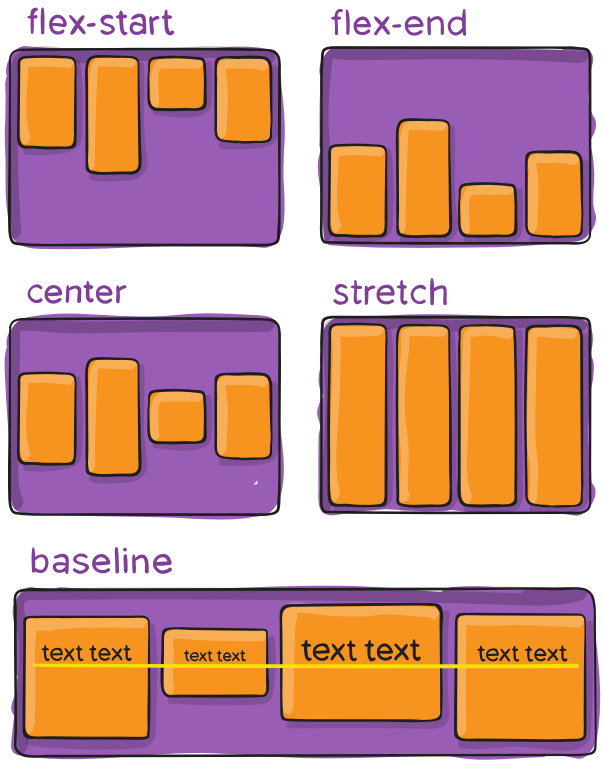
Justify-Content

This defines the alignment along the main axis. It helps distribute extra free space leftover when either all the flex items on a line are inflexible, or are flexible but have reached their maximum size. It also exerts some control over the alignment of items when they overflow the line.

```
.container {
  display: flex;
  justify-content: center;
}
```

- `flex-start` (default): items are packed toward the start of the flex-direction.
- `flex-end`: items are packed toward the end of the flex-direction.
- `center`: items are centered along the line
- `space-between`: items are evenly distributed in the line; first item is on the start line, last item on the end line
- `space-around`: items are evenly distributed in the line with equal space around them. Note that visually the spaces aren't equal, since all the items have equal space on both sides. The first item will have one unit of space against the container edge, but two units of space between the next item because that next item has its own spacing that applies.

Align-Items



This defines the default behavior for how flex items are laid out along the cross axis on the current line. Think of it as the justify-content version for the cross-axis (perpendicular to the main-axis).

```
.container {
  display: flex;
  align-items: stretch;
}
```

- stretch (default): stretch to fill the container (still respect min-width/max-width)
- flex-start / start / self-start: items are placed at the start of the cross axis. The difference between these is subtle, and is about respecting the flex-direction rules or the writing-mode rules.
- flex-end / end / self-end: items are placed at the

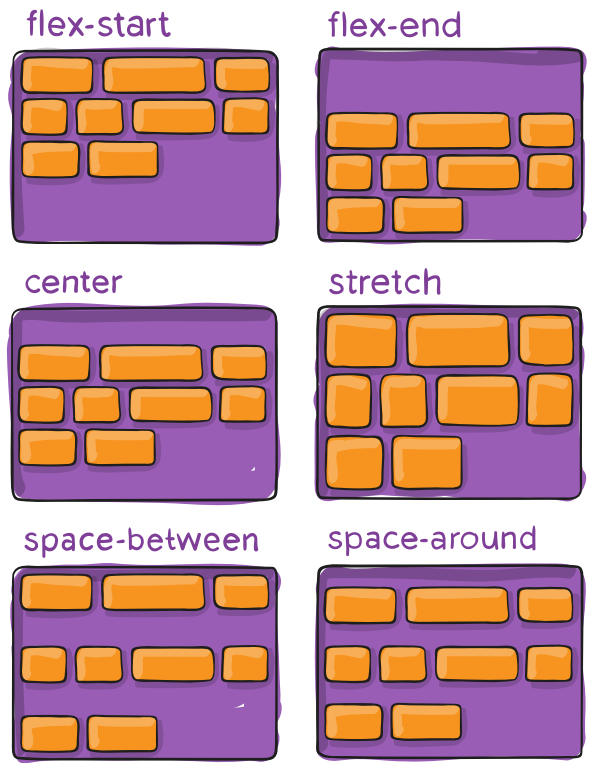
Align-Content

This aligns a flex container's lines within when there is extra space in the cross-axis, similar to how justify-content aligns individual items within the main-axis.

***Note**: This property only takes effect on multi-line flexible containers, where flex-wrap is set to either wrap or wrap-reverse). A single-line flexible container (i.e. where flex-wrap is set to its default value, no-wrap) will not reflect align-content.*

```
.container {
  display: flex;
  align-content: normal;
}
```

- normal (default): items are packed in their default position as if no value was set.
- flex-start / start: items packed to the start of the container. The (more supported) flex-start honors



Questions Time =D