# Flexbox

## Introduction

Using CSS to control the layout has never been straightforward. It can be especially confounding when, you are designing a webpage to display properly on a mobile device, a full HD monitor, and everything inbetween. With increasingly rapid changes in the devices that web pages are expected to displayed on, CSS has been playing catchup. Luckily  some smart folk created libraries to make life easier for us when creating layouts, one of which is called Flexbox.

Although the syntax may be initially tricky, Flexbox lives up to its name. It creates *smart* boxes that can stretch, squeeze and even reorder dynamically. A flex container expands items to fill available free space, or shrinks them to prevent overflow.This provides simple layout solutions that CSS has always struggled with, for example, vertical centering and keeping even heights among elements. After you have mastered the syntax, the flexbox model can make organising dynamic layouts a pleasure.

## Learning Outcomes

By the end of this lesson, you will be able to use the flexbox model to dynamically control the layout of CSS items. You will also apply this model to the example you already created.

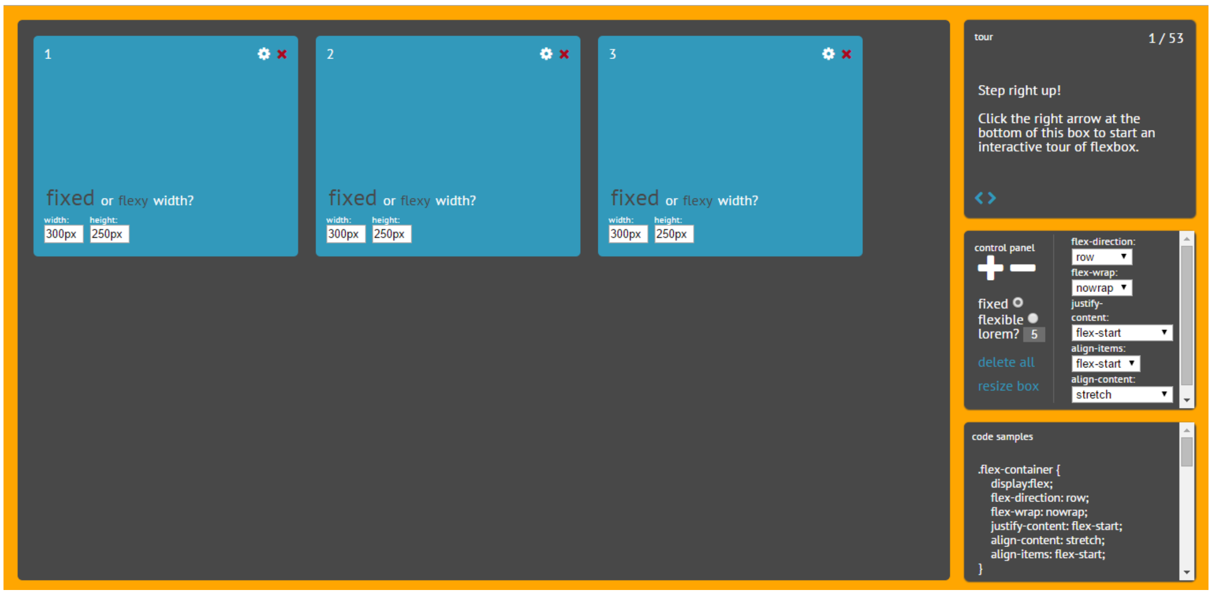
Topics include:

- Getting Started with Flexbox
- This Way Up
- Alignment

# Topic 1: Getting Started with Flexbox

To begin, you define a container to have a `display: flex`. Any items that you place within this (flex) container are now **flex items** and have intelligent responsive properties. These flex items can automatically expand to fill the container space, both horizontally and vertically. You can resize and reorder flex items simply with a few lines of CSS. You do not have to recalculate margins or worry about overflowing elements. Let's take a look at an example of what it can do.

There is a great demo site for flexbox: http://www.flexboxin5.com



Step right up!

Click the right arrow at the bottom of this box to start an interactive tour of flexbox.

flex-direction: row
flex-wrap: nowrap
justify-content: flex-start
align-items: flex-start
align-content: stretch

```
.flex-container {
    display:flex;
    flex-direction: row;
    flex-wrap: nowrap;
    justify-content: flex-start;
    align-content: stretch;
    align-items: flex-start;
}
```

## 1.1 Parent-Child Relationship

The flexbox container and the flex items have a parent-child relationship. Each child of a flexbox container automatically becomes a flex item. The parent container controls much of the behavior of the children (alignment, spacing, and direction).

### >_ Note:

You define a flex container using the **flex** value of the **display** property.

```
div{
    display:flex;
}
```

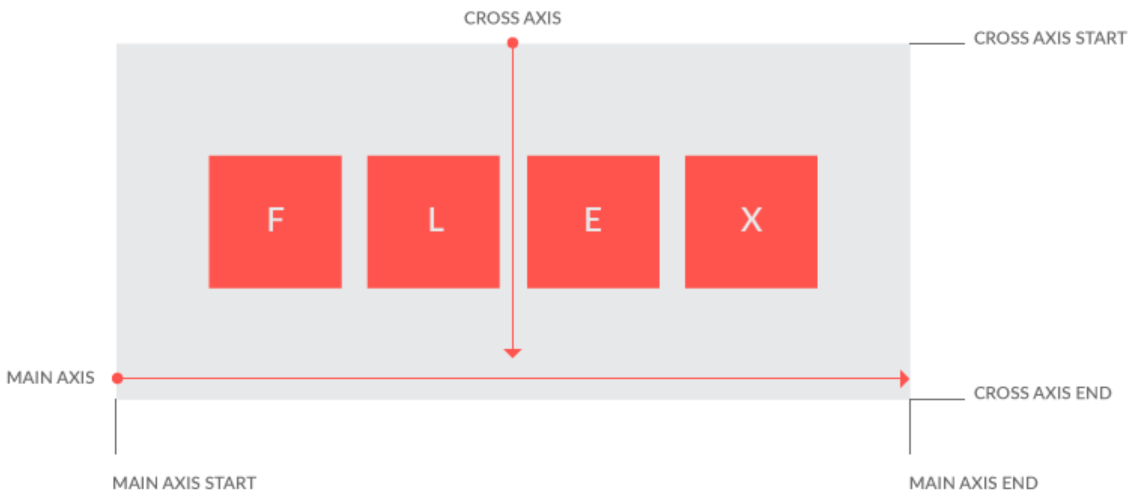
## Topic 2: This Way Up

Before you can define how your **flex items** behave, you need to establish a frame of reference for them i.e tell them which way is up. In a conventional web page layout, this may seem like an extraneous step. However, altering the direction makes new and interesting layout options available to you. Also, consider the complications of handheld devices: you may need to control how your layout behaves as the user switches their device from landscape to portrait.

### 2.1 Axes

The layout of a flexbox follows two axes; the **main axis** and the **cross axis**.

- The main axis is the axis along which the flex items follow each other.
- The cross axis is the axis perpendicular to the main axis.



AXES

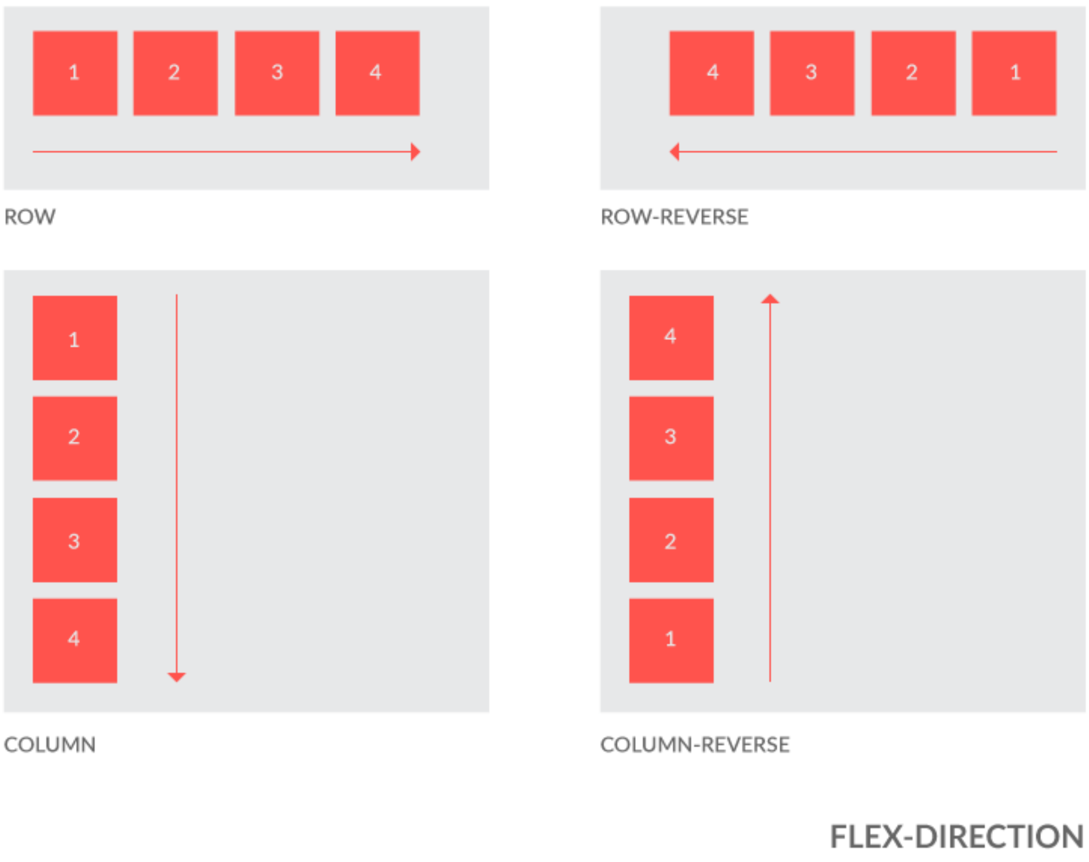
## 2.2 The `flex-direction` Property

Within the properties of the flexbox container, you use the **flex-direction** property to establish the main axis. There is no need to establish the cross axis, as this is always perpendicular to the main axis. For example, if you choose the **row** value, flex items display horizontally from left to right and this is the main axis. Then, the cross axis will be vertical from top to bottom.

```
flex-direction: row;
```

The values are as follows:

| **Value** | **Description** |
|---|---|
| `row` | Displays flex items horizontally, from left to right. (Default) |
| `row-reverse` | Displays flex items horizontally, from right to left. |
| `column` | Displays flex items vertically, from top to bottom. |
| `column-reverse` | Displays flex items vertically, from bottom to top. |



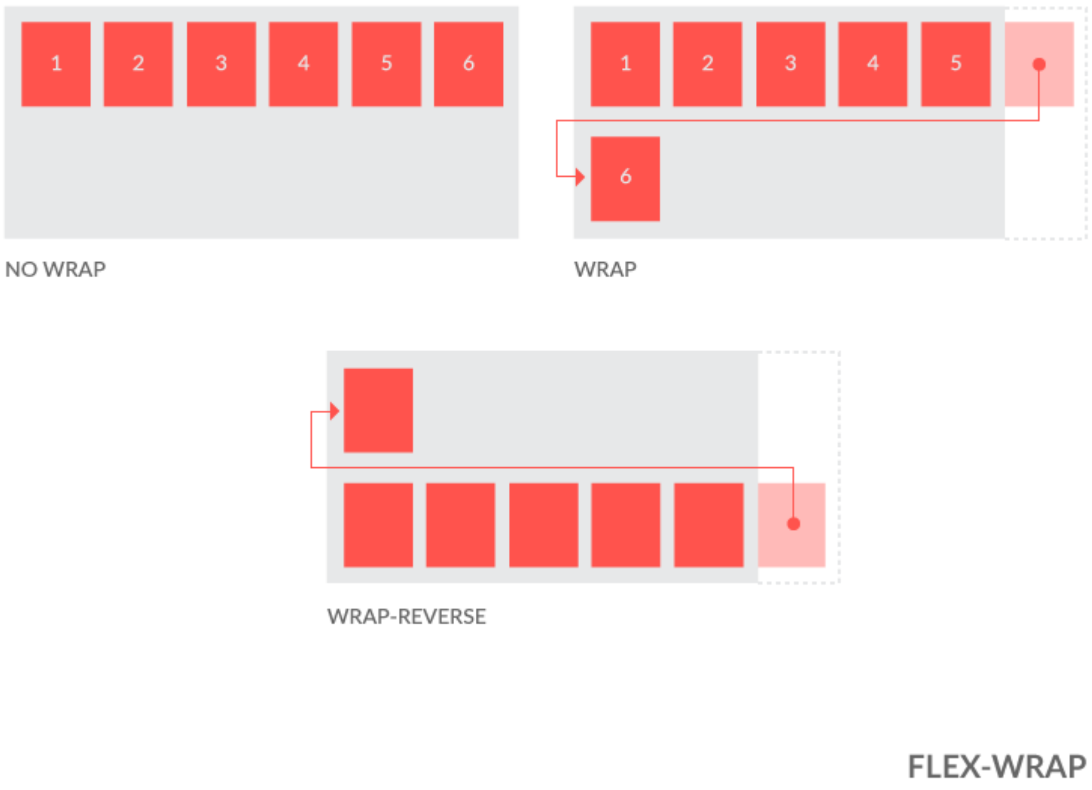
## 2.3 The `flex-wrap` property

Having defined the direction in which your flex items display, you can determine how they behave. The flexbox container's **flex-wrap** property defines whether the flex items are forced into a single line along the main axis or can flow into multiple lines.

```
flex-wrap: wrap;
```

The **flex-wrap** values are:

| **Value** | **Description** |
|---|---|
| `nowrap` | Specifies that flex items do not wrap. (Default) |
| `wrap` | Specifies that flex items wrap if necessary. |
| `wrap-reverse` | Specifies that flex items wrap if necessary but in reverse order. |



## 2.4 Shorthand: The `flex-flow` property

The flex-flow property is a shorthand for setting the flex-direction and flex-wrap properties. It uses the following syntax:

flex-flow: flex-direction flex-wrap;

Example:

```
flex-flow: row nowrap;
```

## ✪ *challenge: flexbox*

## Topic 3: Alignment

Flexbox containers use the properties **justify-content** and **align-items** to handle the previously tricky task of alignment in CSS. Not alone is it much easier but you also have some handy options available to you.

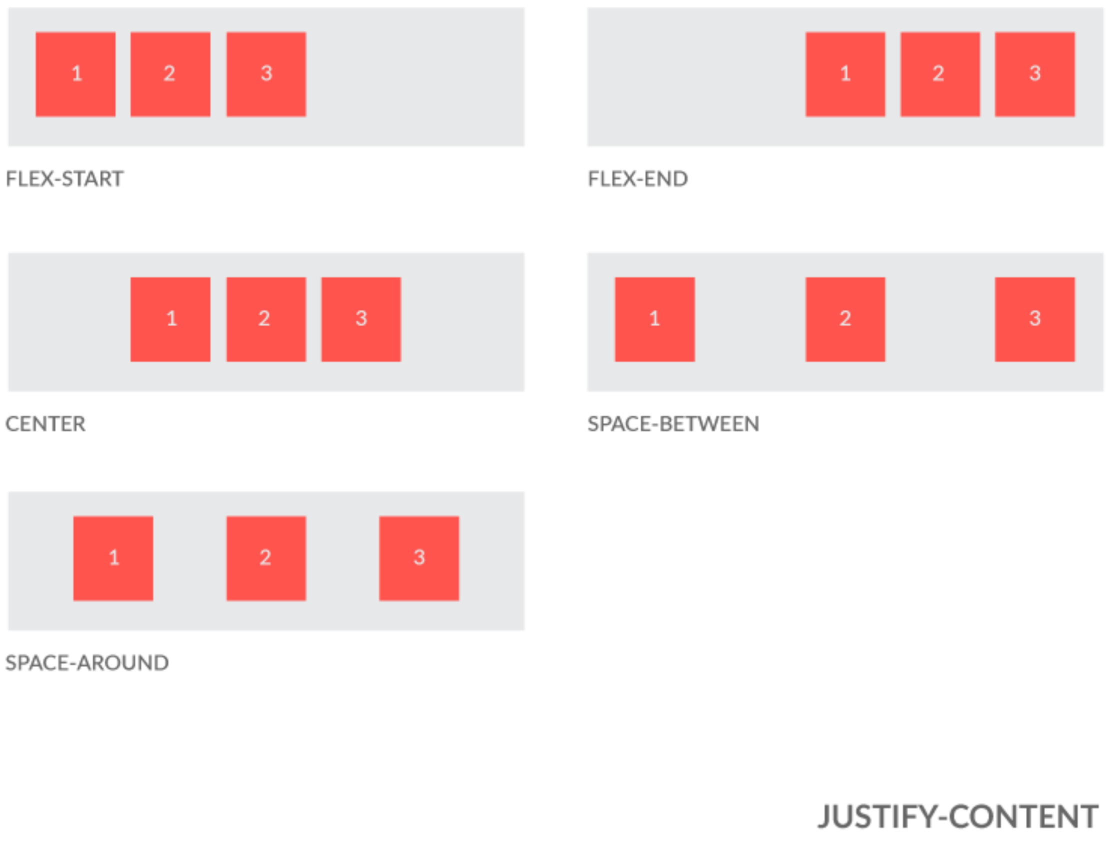
### 3.1 The `justify-content` property

This property defines how **flex items** are laid out along the **main** axis. Essentially, it distributes the extra free space when the flex items do not fill the width available to them on the main axis.

```
justify-content: flex-end;
```

The values of the **justify-content** property are:

| Value | Description |
|---|---|
| `flex-start` | Positions flex items at the beginning of the flexbox container. (Default) |
| `flex-end` | Positions flex items at the end of the flexbox container. |
| `center` | Positions flex items at the center of the flexbox container. |
| `space-between` | Positions flex items with space between them only. |
| `space-around` | Positions flex items with space before, between, and after them. |



## ✪ *challenge: alignment*

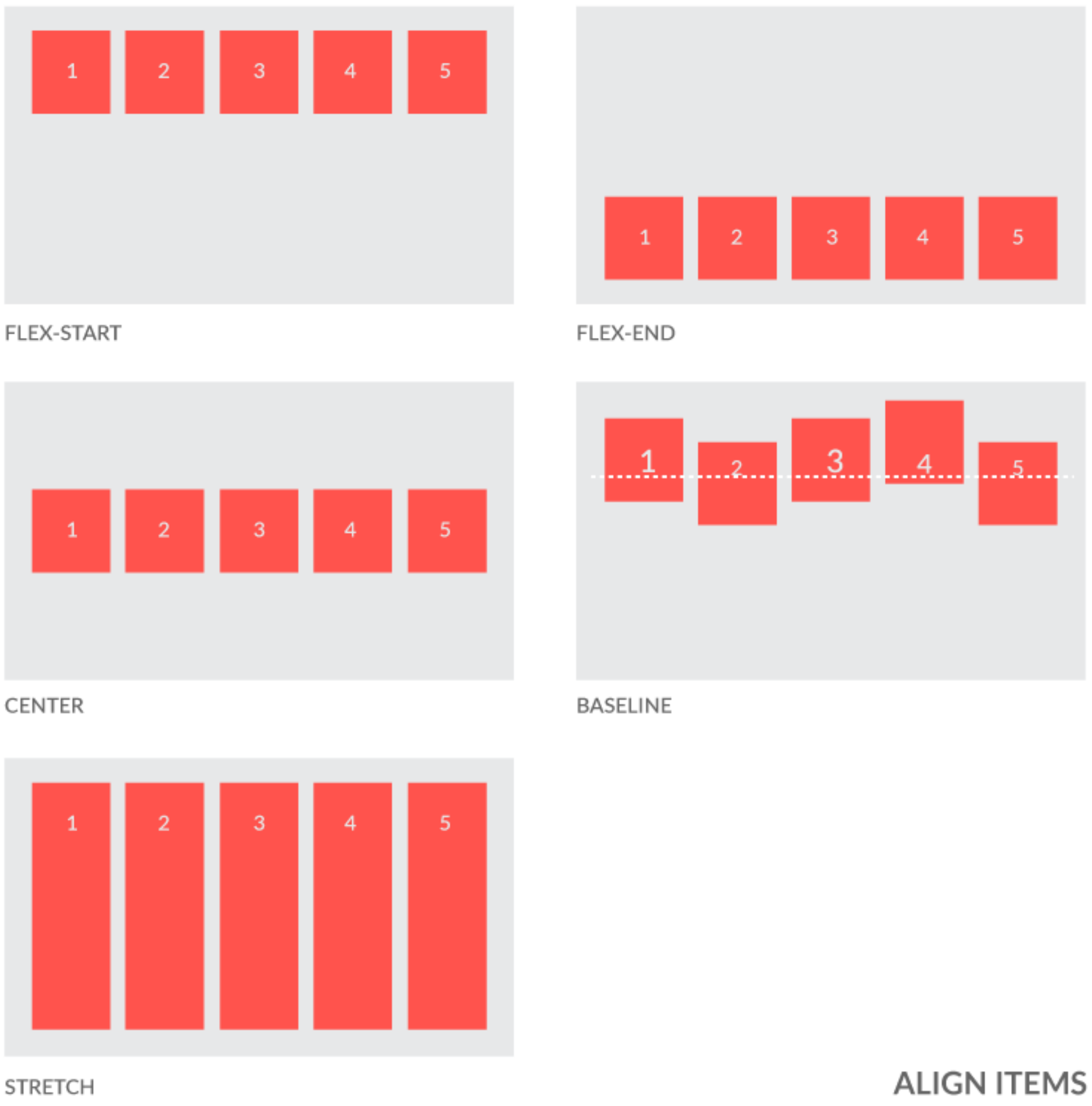
## 3.2 The `align-items` property

The **align-items** property works similar to the **justify-content** property except along the **cross axis**. In addition, it has the two useful values of stretch and baseline.

The **align-items** property values:

| **Value** | **Description** |
|---|---|
| `flex-start` | Positions flex items at the beginning of the flexbox container. |
| `flex-end` | Positions flex items at the end of the flexbox container. |
| `center` | Positions flex items at the center of the flexbox container. |
| `baseline` | Positions flex items along the baseline of their content |
| `stretch` | Stretches flex items to fit the flexbox container. (Default) |

```
align-items: center;
```



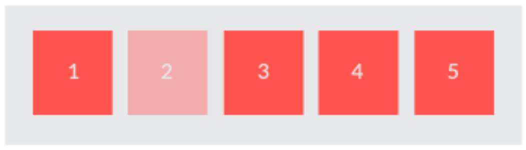
## >_ Note:

One of the trickier layouts in CSS is centering an element vertically and horizontally in a space, and resizing it dynamically. Previously, you would need to employ all sort hacks and round about ways to make this work. Working with **justify-content** and **align-items**, you can achieve this in a few lines of CSS.

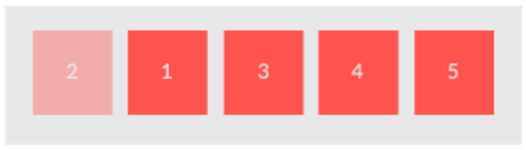
### *challenge: align-items*

## Topic 4: Call to Order

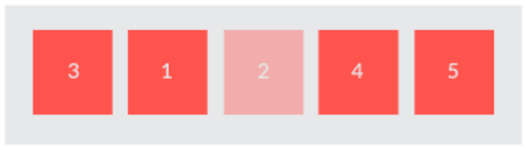
Another aspect that you can control with the flexbox model is the order in which flex-items appear. By default, items appear according to the order they are created within the HTML page. However, you can customize the order in which they appear using the flex-item's `order` property.



DEFAULT



ORDER:1



ORDER:3

ORDER

The `order` property uses the following syntax:

```css
order: integer;
```

## ✪ *challenge: walkthrough project*

## ✪ *challenge: flexbox media query*

## Summary

Although the syntax may be initially tricky, the flexbox model makes laying out CSS items relatively easy and provides simple layout solutions that CSS has previously struggled with.