HTML5: Flexbox

After completing this section, you will:

- [ ] Understand the viewport meta tag.
- [ ] Be able to use flexbox layout tools.
- [ ] Use direction, wrap, justification and alignment to create a more responsive design.

Introduction

In the previous section, we used floating divs to lay out our page. We also addressed some common page design techniques.

Now, we'll upgrade a bit and use the new flexbox layout tools which are part of CSS3. **Flexbox** is quite a bit easier because it consists of not one but two rule types: The **container** rules and the **item** rules. The central idea behind flex layout is to set rules on parent containers and the items within it to make layout easier.

Once introduced to the power and ease of Flexbox most developers abandon older layout techniques. However, there is well over a decade of production code running on the web that was created before Flexbox, so, it's important to understand both older layout techniques and Flexbox.

Viewport Meta Tag

Today, over 52% of all internet users browse websites from mobile devices. The viewport meta tag is what makes a regular page more responsive and usable on mobile devices

To use this tag, we'll begin with the HTML5 basic document structure.

```
<!DOCTYPE html>
<html>
    <head>
        <title>Flex Box Demo</title>
        <meta name="viewport" content="width=device-width,initial-scale=1">
    </head>
    <body>

    </body>
</html>
```

The viewport tag is placed in the HTML document head which tells the browser how to render the page. This tag tells the browser to set the content width to the width of the device itself and to scale it to 1 (100%) when the page is loaded.

You should probably get in the habit of using the viewport meta tag with most HTML documents.

HTML Layout

Let's start with an HTML document that we'll use to demonstrate Flexbox.

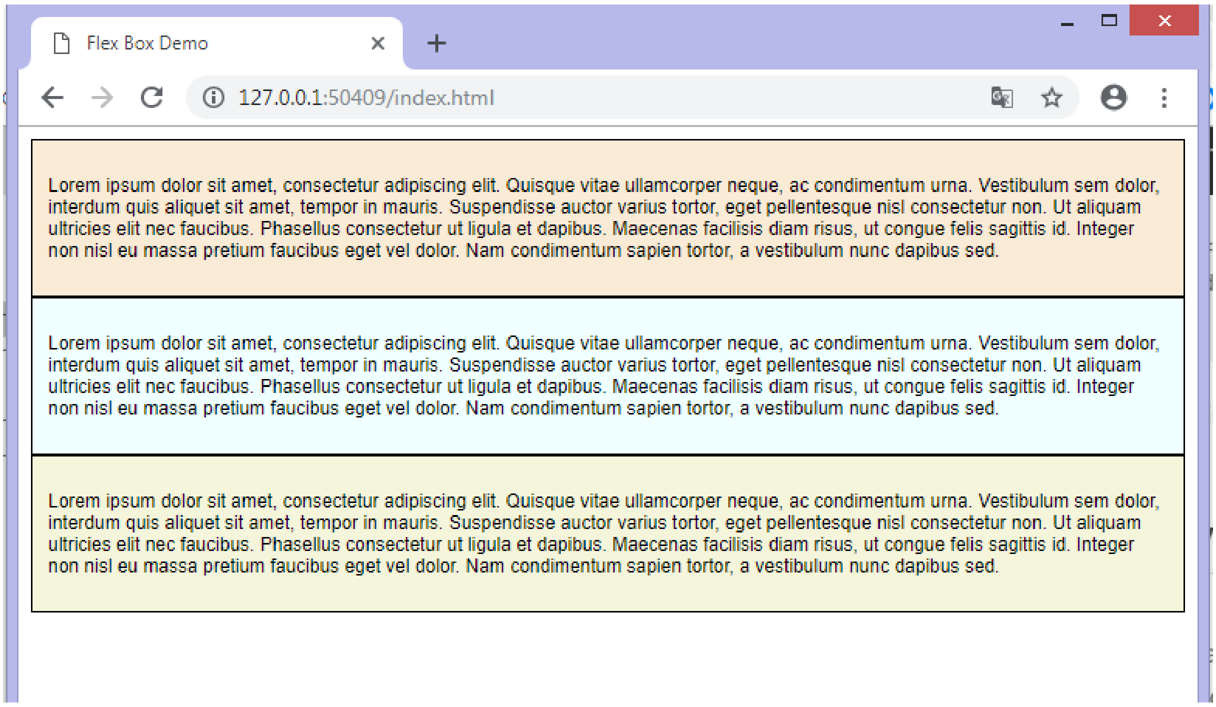
We have created a `container div` and three logical divisions with the ids `one`, `two`, and `three`. We'll add conmon styles to all the logical divisions with a `border: 1px solid black`, `font-family: Arial`, `font-size: .75em`, `padding: 10px`.

To make them easier to identify, we will also add a different background color to each logical division.

```
<!DOCTYPE html>
<html>
    <head>
        <title>Flex Box Demo</title>
        <meta name="viewport" content="width=device-width,initial-scale=1">
        <style>
            #one, #two, #three {
                border: 1px solid black;
                font-family: Arial;
                font-size: .75em;
                padding: 10px;
            }
            #one { background-color: antiquewhite; }
            #two { background-color: azure; }
            #three { background-color: beige; }
        </style>
    </head>
    <body>
        <div id="container">
            <div id="one">
                <p>
                Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque vitae ullamco
                </p>
            </div>
            <div id="two">
                <p>
                Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque vitae ullamco
                </p>
            </div>
            <div id="three">
                <p>
                Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque vitae ullamco
                </p>
            </div>
        </div>
    </body>
</html>
```

> Don't forget that you can get lipsum text (filler text) at www.lipsum.com.

If you test in the browser, you should see your page should look like this:



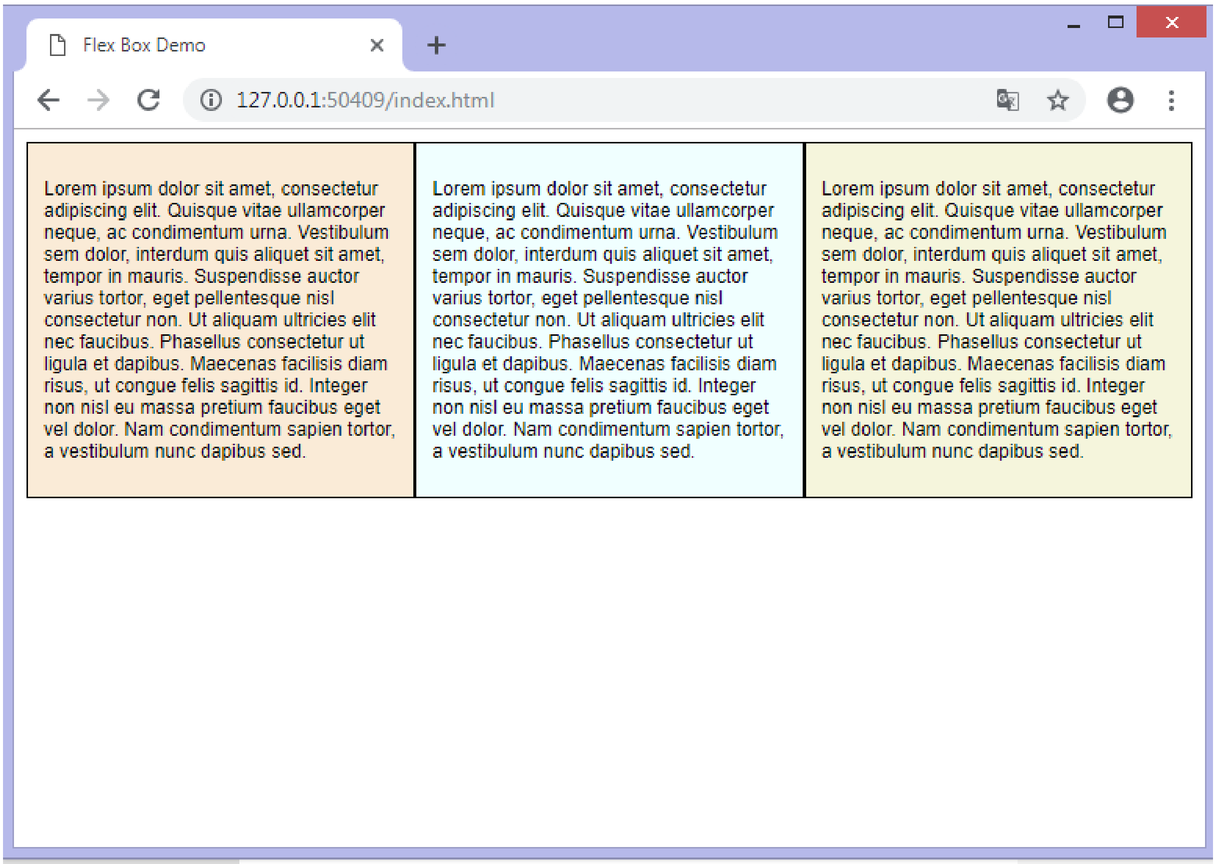
Display: Flex

Flexbox is an entire CSS module and not a single property. Flexbox rules in CSS are either applied o the parent container or child elements. The syntax is the same as any other CSS rule.

Let’s focus on the parent element (also known as the flex container) and activate flexbox through the display rule. (Typically the display rule value is `inline` or `block`.

```
<style>
#container {
    display: flex;
}
</style>
```

You should now see your content laid out in three columns.



The default behavior of a flexbox container is to lay child elements out in a single row. You'll notice that at this point the order of the child elements is unchanged.

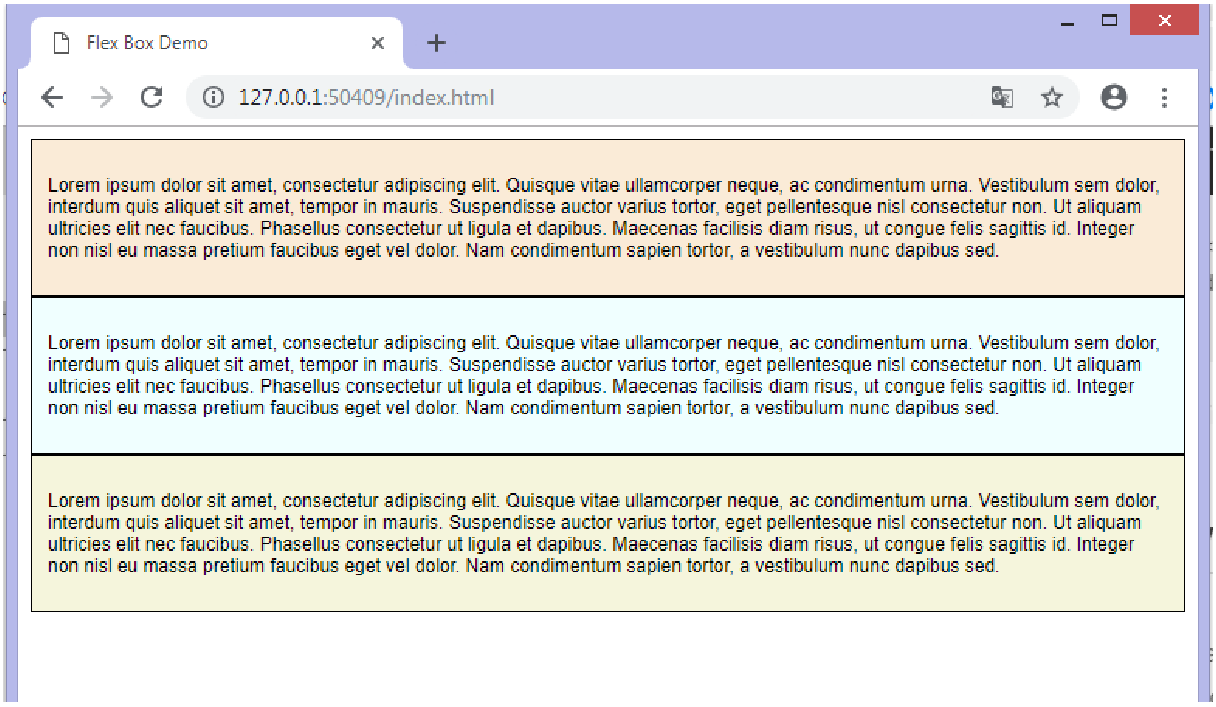
Direction

The general layout behavior of the children within a flex container is controlled by the `flex-direction` rule. Generally, content can be placed in horizontal rows or vertical columns. There are four possible values for `flex-direction`: `column`, `column-reverse`, `row`, and `row-reverse`.

Now, let's set the `flex-direction` value to `column`.

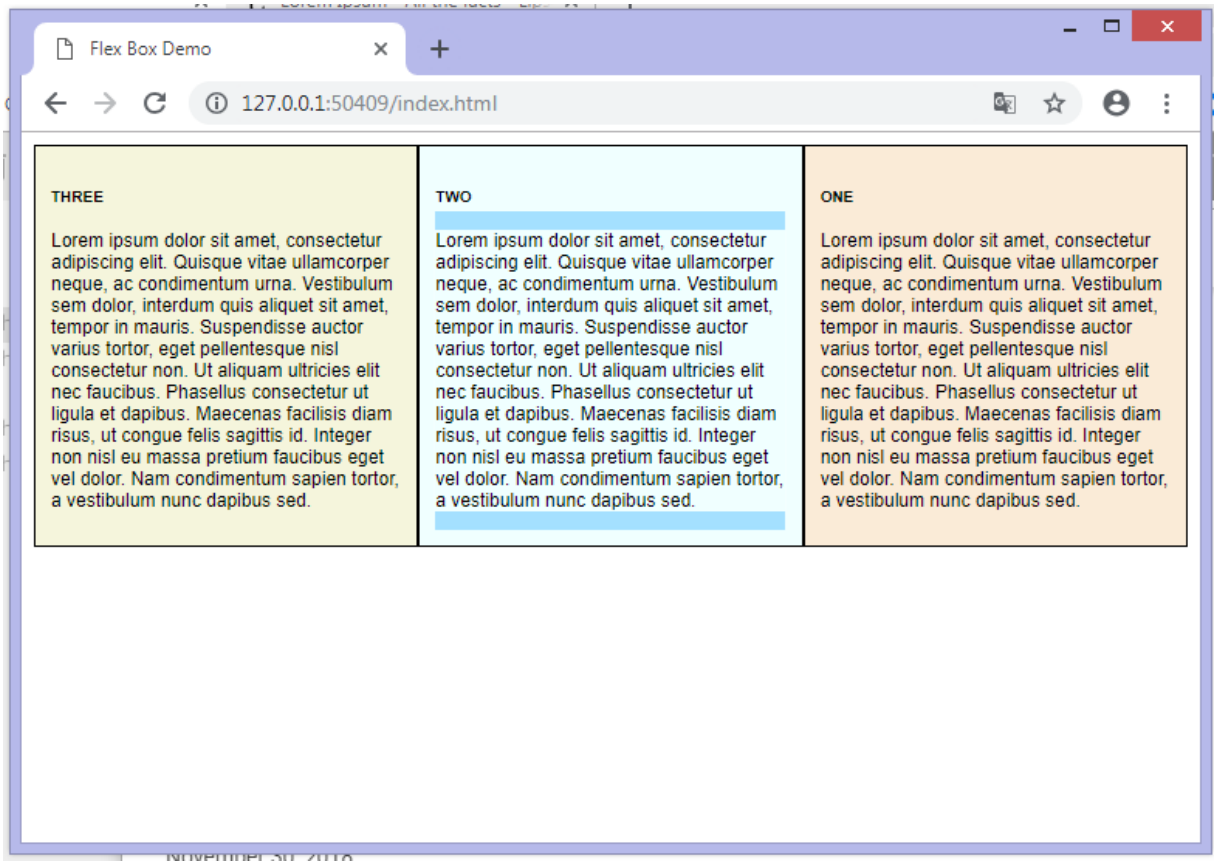
```
<style>
#container {
    display: flex;
    flex-direction: column;
}
</style>
```

With the rule set, our child content is now displayed in a single column.



If we set the flex-direction to `row-reverse`, the column `three` will be the first, as the order is reversed from the order in which the child elements appear in the code.

```
<style>
#container {
    display: flex;
    flex-direction: row-reverse;
}
</style>
```



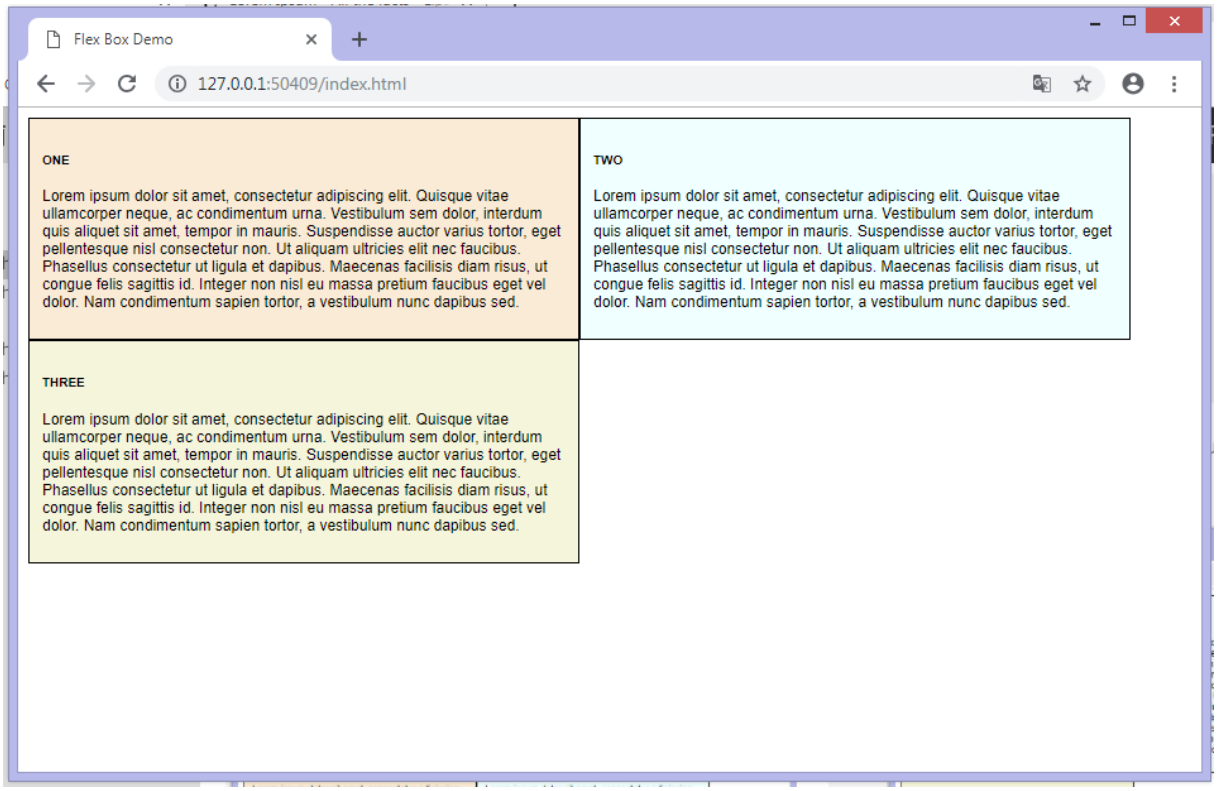
Wrap

Let's revert our `flex-direction` rule to `row` and use the `flex-wrap` rule. By default, flex items (child items) try to fit onto one baseline. We can adjust how child elements flow with 'flex-direction'. There are three possible values for `flex-wrap`: `wrap`, `nowrap` and `wrap-reverse`.

Let's use `wrap` as our `flex-wrap` value and add a `45%` width in our three logical divisions to demonstrate how the wrap works.

```
<style>
#container {
    display: flex;
    flex-direction: row;
    flex-wrap: wrap;
}
</style>
```

In the browser, you should see that the column three will wrap onto the next line from top to bottom.



If we use the `wrap-reverse`, flex items will wrap onto the next line from bottom to top. While the default setting, `nowrap`, is used, the flex items are in one line.

Order

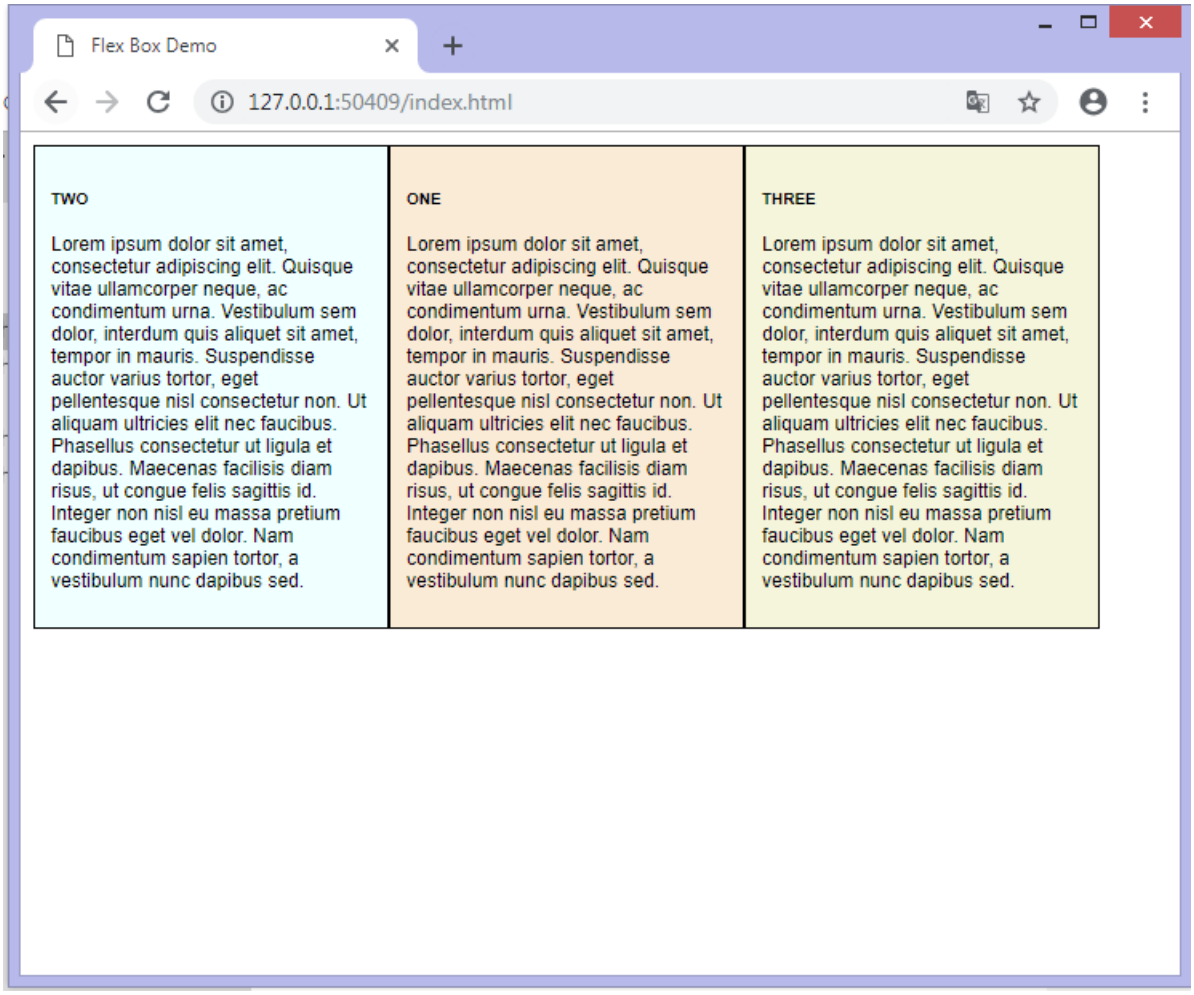
By default, flex items are laid out in the source order. You can override the default order with the `order` property which controls the order in which the items appear in the flex container.

Let's add `order: -1` in #two div:

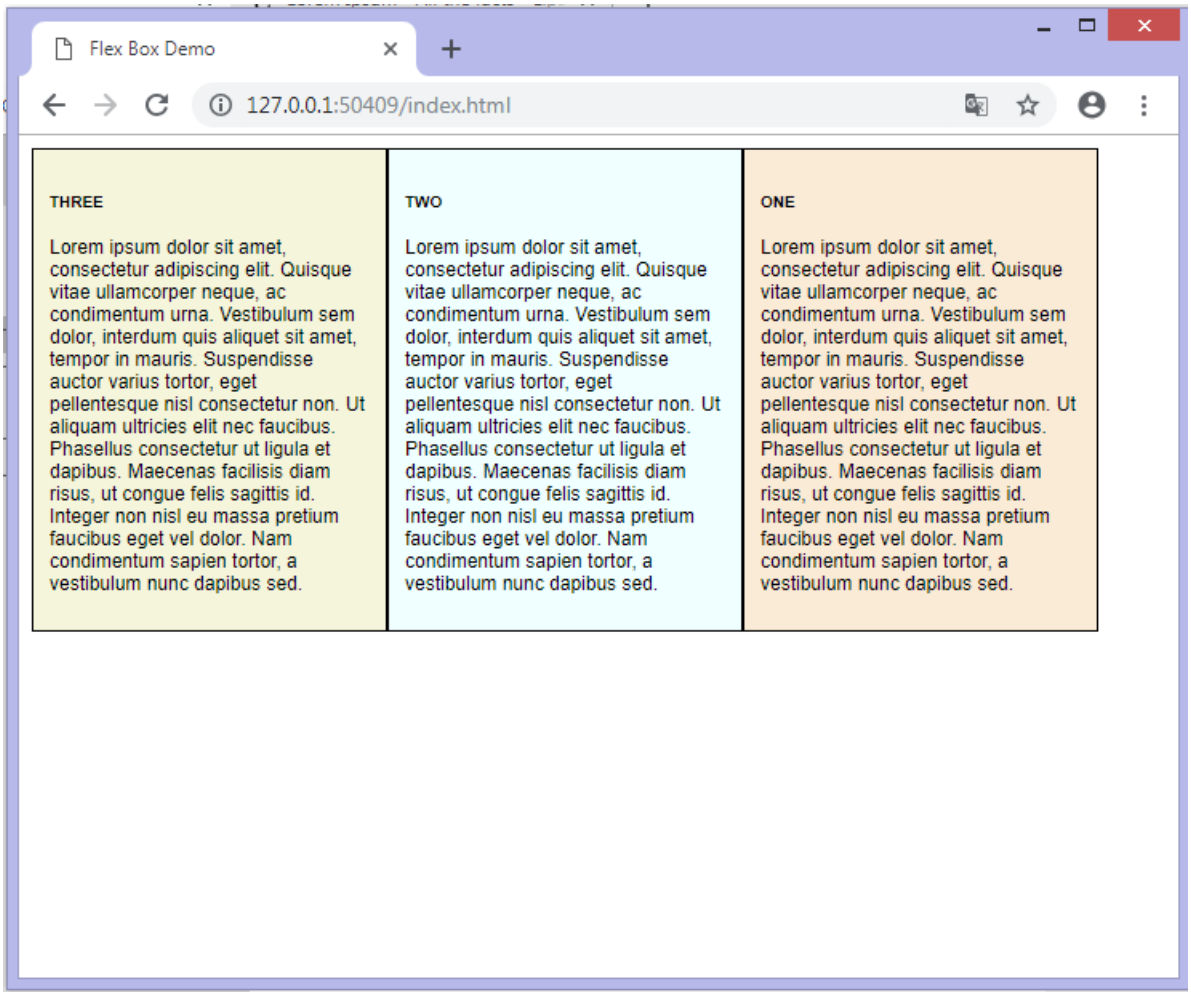
```
<style>
    #one { background-color: antiquewhite; }
    #two { background-color: azure; order: -1; }
    #three { background-color: beige; }

</style>
```

You will notice that column two will go first.



So, if we set column three to `order: -2` then column three will go first.

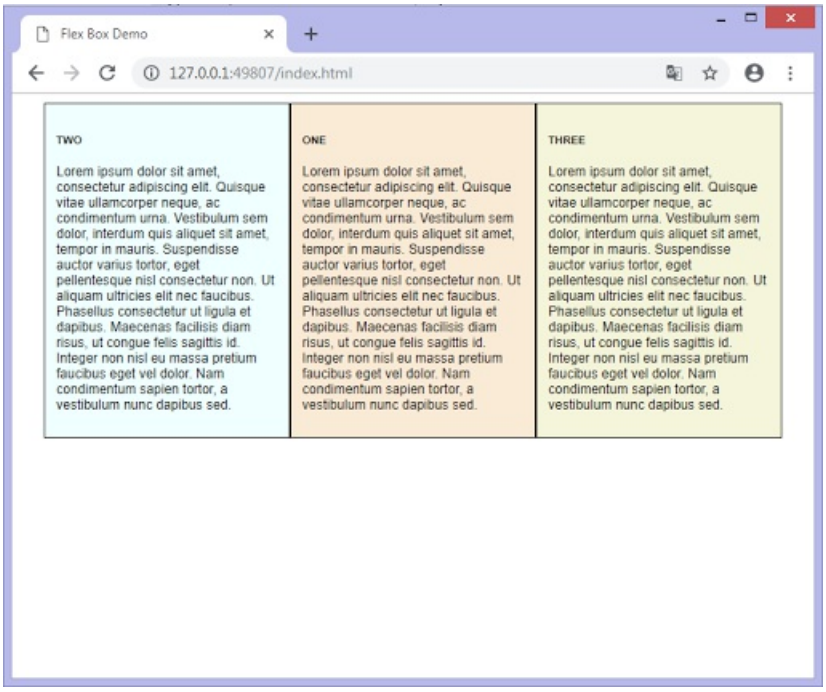


Necessarily, the order element will start with the lowest value and layout larger values in order.

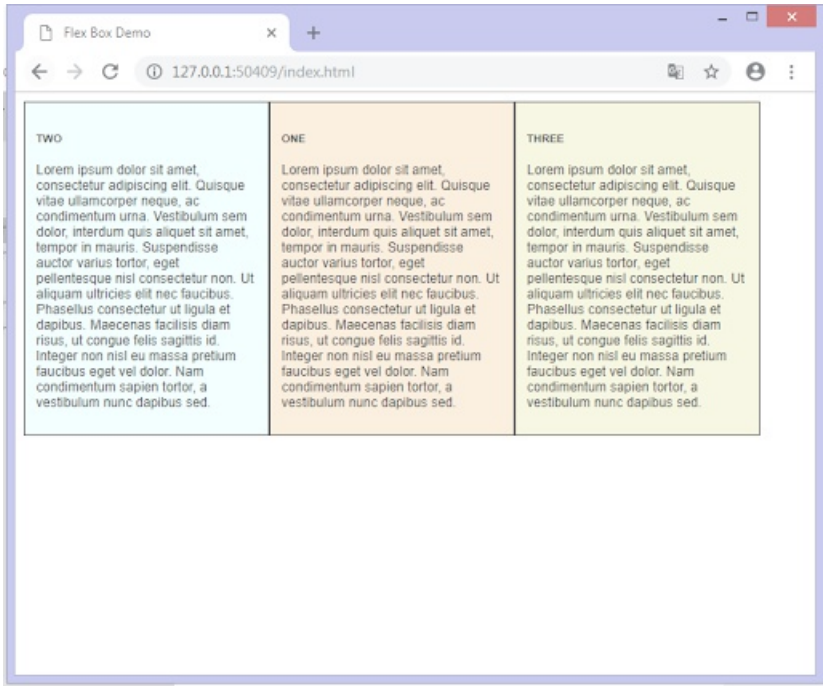
Justification

The `justify-content` rule defines the alignment within the flex container. The rule helps to distribute additional free space when the flexible elements on a line are inflexible or flexible but reach their maximum size.

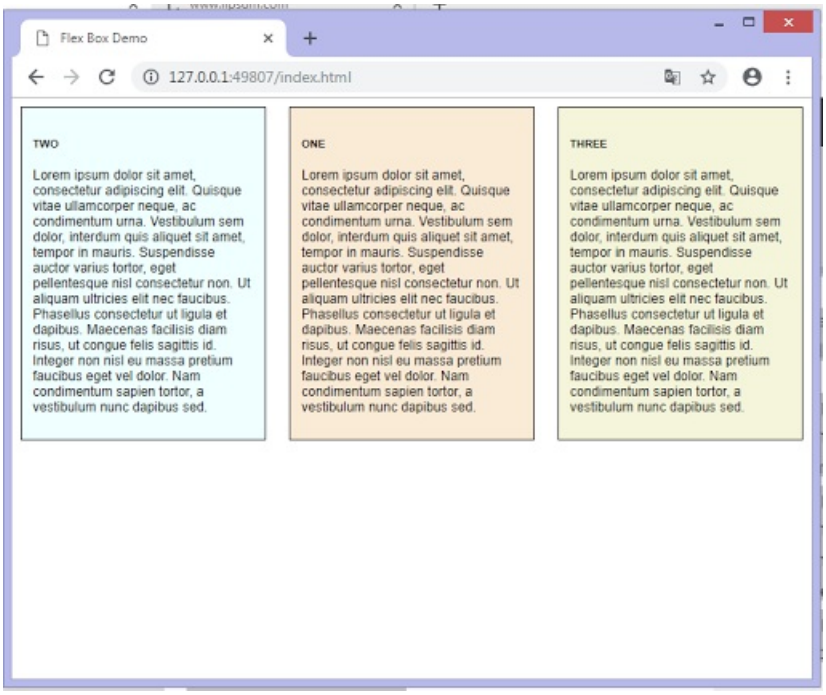
There are five possible settings for the `justify-content` rule:



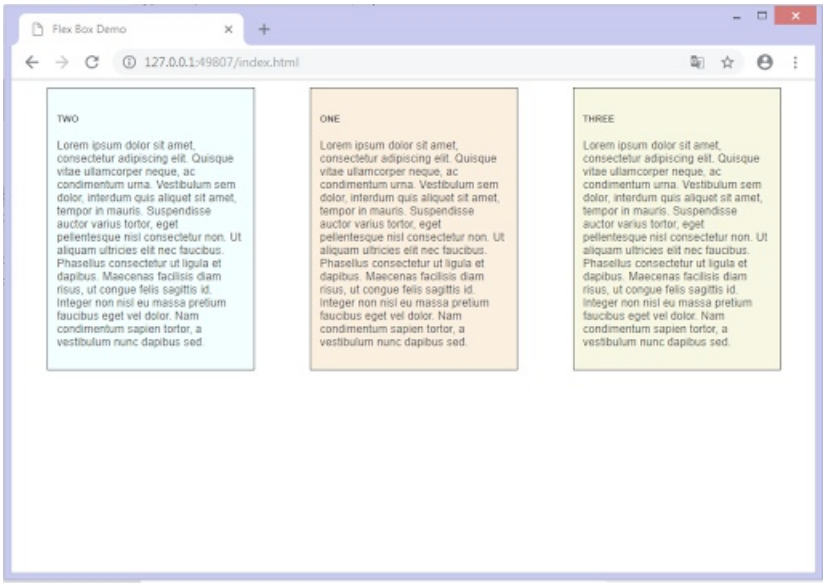
`justify-content: center` - Flex items are centered along the baseline.



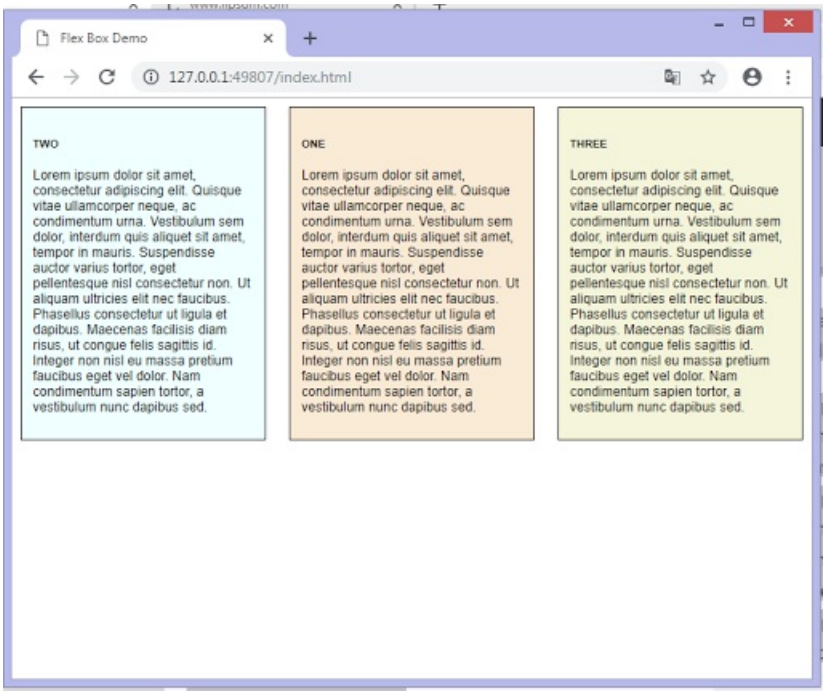
`justify-content: flex-start` - This is the default setting. Flex items are placed toward the start line with extra space being placed to the right of the elements.



`justify-content: space-between` - Flex items are evenly distributed in the baseline. The first item is on the start, the last item on the end line.



`justify-content: space-around` - Flex items are evenly distributed in line with equal space around them. Keep in mind that visually the spaces aren't equal since all the flex items have equal space on both sides.



`justify-content: space-evenly` - Flex items are distributed so that the spacing between any two items is equal.

Alignment

Let's add more content in column one to make the columns unbalanced. This will make it easier to see how the `align-items` rule works.

We'll also set `height: 100%` to flex container. The possible values for the `align-items` rule are `center`, `flex-start`, `flex-end`, `stretch`, and `baseline`.

First let's try `flex-start`:

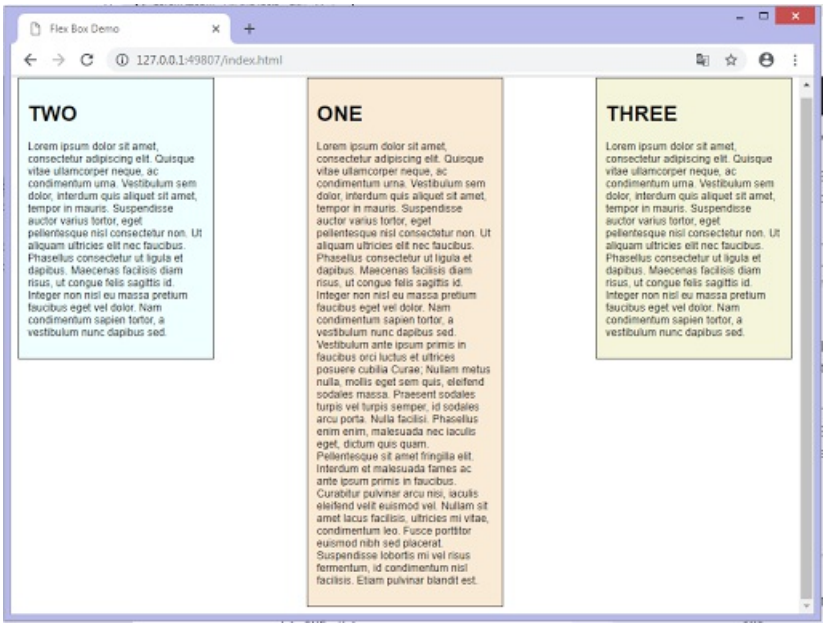
```
<!DOCTYPE html>
<html>
    <head>
        <title>Flex Box Demo</title>
        <meta name="viewport" content="width=device-width,initial-scale=1">
        <style>
            #one, #two, #three {
                border: 1px solid black;
                font-family: Arial;
                font-size: .75em;
                padding: 10px;
 width: 200px;
            }
            #one { background-color: antiquewhite; }
            #two { background-color: azure; order: -1 }
```

```
            #three { background-color: beige; }

 #container {
 display: flex;
 flex-direction: row;
 flex-wrap: wrap;
 justify-content: space-between;
 height: 100%;
 align-items: flex-start;
 }
        </style>
    </head>
    <body>
        <div id="container">
            <div id="one">
 <h1>ONE</h1>
                <p>
                Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque vitae ullamco

 Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Nulla
                </p>
            </div>
            <div id="two">
 <h1>TWO</h1>
                <p>
                Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque vitae ullamco
                </p>
            </div>
            <div id="three">
 <h1>THREE</h1>
                <p>
                Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque vitae ullamco
                </p>
            </div>
        </div>
    </body>
</html>
```

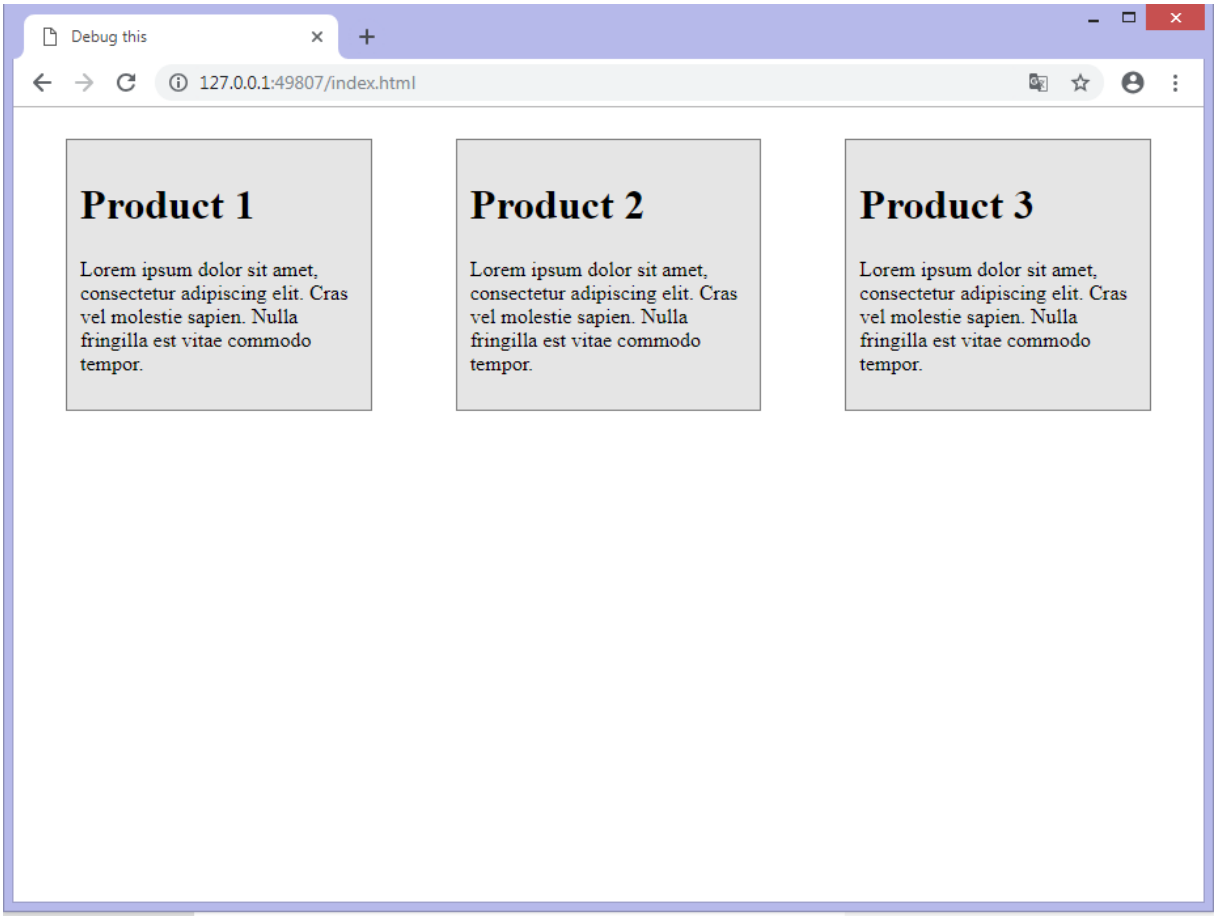
As you can see from the screenshot below, the alignment is at the top.



If we use `flex-end`, the alignment is at the bottom. While `baseline` items are aligned such that their baselines align. The default value, `stretch`, will stretch the child elements to fill the container.

Debug This

There are errors in this code preventing it from displaying the product list correctly. Fix the errors, so the products display correctly in your browser like this:



Here is the code to debug:

```
<!DOCTYPE html>
<html>
    <head>
        <title>Debug this</title>
        <meta name="viewport" content="width=device-width,initial-scale=1">
        <style>
 #products {
 flex-flow: column;
 align-items: flex-start;
 justify-content: center;
 }
 #products > div {
 border: 1px solid grey;
 padding: 10px;
 background: rgba(0,0,0,0.1);
 margin: 1em 2em;
 }
        </style>
    </head>
    <body>
        <div id="products">
 <div>
 <h1> Product 1 </h1>
 <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Cras vel molestie sapien. Nulla f
 </div>
           <div>
 <h1> Product 2 </h1>
 <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Cras vel molestie sapien. Nulla f
 </div>
 <div>
 <h1> Product 3 </h1>
 <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Cras vel molestie sapien. Nulla f
 </div>
        </div>
    </body>
</html>
```

Submit This

Create an HTML5 document from scratch that is correctly formed and coded that displays a photo gallery with 3 columns and 3 rows using flexbox. Label each photo with a title and one-two sentence description. Use additional CSS to make sure the gallery is readable and attractive.

Happy coding!

Remember, when submitting the work please use the following naming convention for your file: HTMLAUTHORING_LastName_SectionName.html . So if your last name is Smith and you are submitting

this section. Your file name should be HTMLAUTHORING_Smith_Flexbox.html . Since you have two or more files for this exercise, please zip them together before uploading.

For this course go to https://www.dropbox.com/request/RhW9kBDXtisq2Fsvg3hY to submit your assignments.