# I. CSS designing

## 1. CSS FlexBox

FlexBox aka Flexible Box Layout makes it easier to layout, align and style items in the container while maintaining the responsiveness of the website.

To create a flexbox you need to set the display of the container as flex
Eg: {display: flex;}

This element is called the flex container, and stores the sub-elements which are known as flex items

**Flex Container Properties**
The flex container properties are:

1. **Flex Direction**
   It defines in which direction the flex elements would be displayed. It takes values like row, column or “reverse” too.

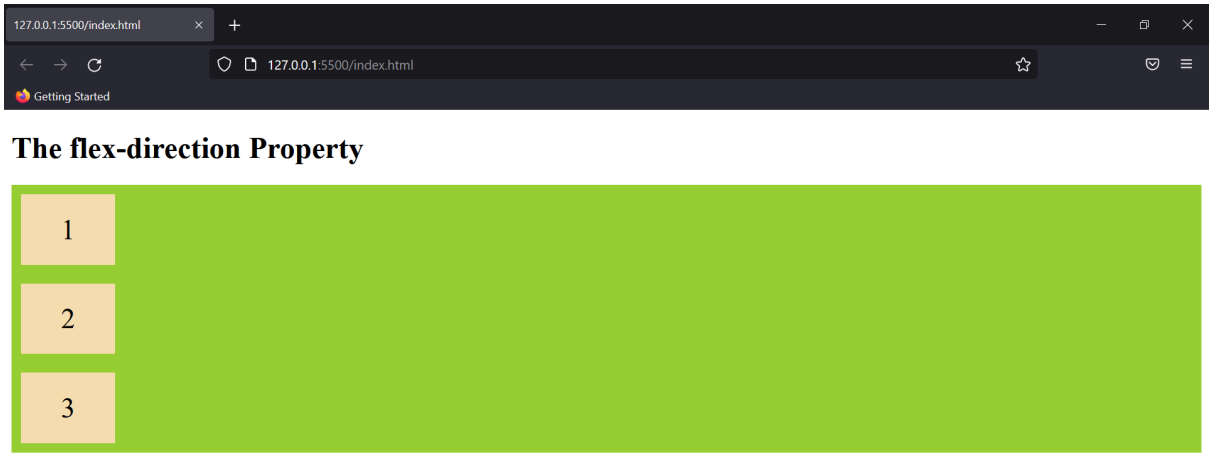
   Eg:

index.html

index.html > html > head > style > div

```
<!DOCTYPE html>
<html>

<head>
    <style>
        .flex-container {
            display: flex;
            flex-direction: column;
            background-color: yellowgreen;
        }

        div {
            background-color: wheat;
            width: 100px;
            margin: 10px;
            text-align: center;
            line-height: 75px;
            font-size: 30px;
        }
    </style>
</head>

<body>

    <h1>The flex-direction Property</h1>
    <span class="flex-container">
        <div>1</div>
        <div>2</div>
        <div>3</div>
    </span>

</body>
```

Output:




2. **Flex Wrap**
   By using this property we can make our elements responsive for different screen sizes.

   Eg:

```
.flex-container {
        display: flex;
        flex-direction: row;
        background-color: yellowgreen;
        flex-wrap: wrap;
    }
```

   Output:

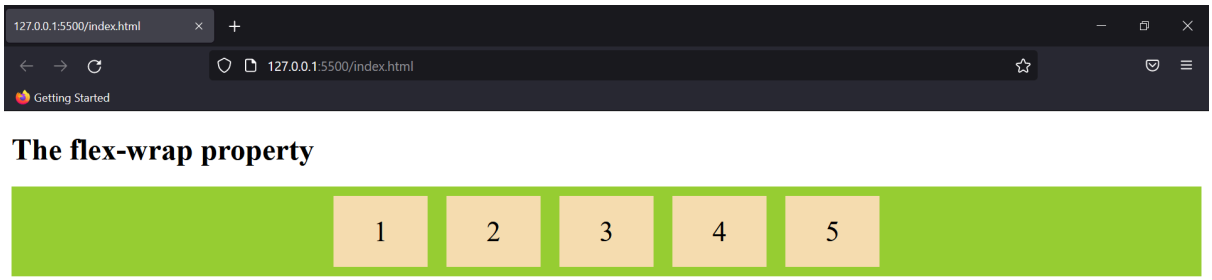

   You can use flex flow as a short to add both these properties. Eg: {flex-flow: row wrap;}

3. **Justify Content**

This property is used to set the position of content or rather align content along the main axis.
Eg:

```
.flex-container {
        display: flex;
        flex-direction: row;
        background-color: yellowgreen;
        justify-content: center;
    }
```

Output:




It can take other values too like “space-around” which distributes flex items equally spaced in the container. Other properties include flex-end, flex-start, space-between, etc. (These could be seen in vs code when the justify-content property is selected).

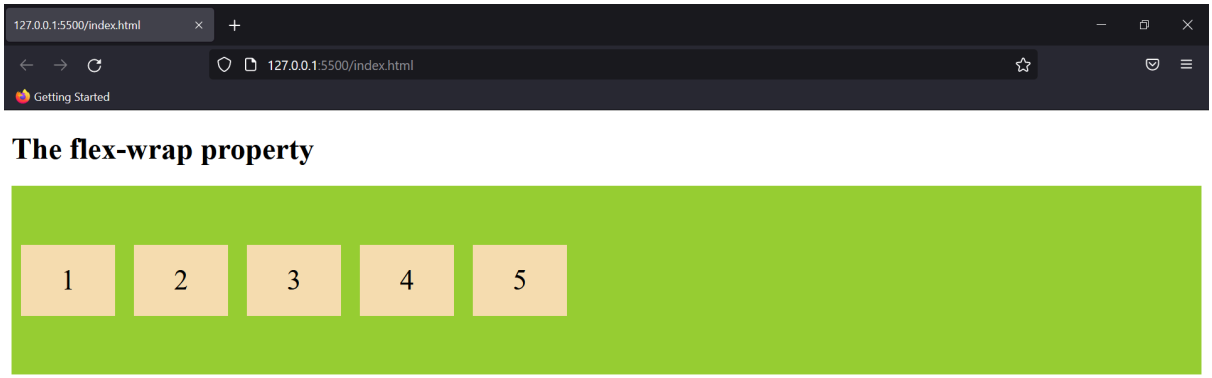
4. **Align Items**

Just like the justify-content property, align-items define the alignment of the flex container but along the cross-axis.

Eg:

```
.flex-container {
        display: flex;
        height: 200px;
        flex-direction: row;
        background-color: yellowgreen;
        align-items: center;
    }
```

Output:

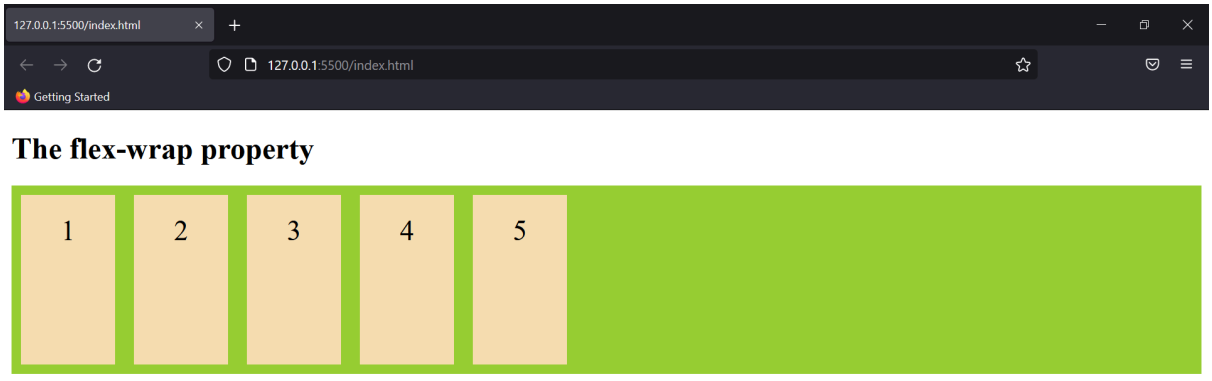



5. **Align Content**
   This property is very similar to align item but here rather than the flex items, the content present in the item is selected for the property.

   Eg:

```
.flex-container {
        display: flex;
        height: 200px;
        flex-direction: row;
        background-color: yellowgreen;
        align-content: center;
    }
```

   Output:




**Flex Items Properties**

The flex item properties are:

1. **Order:**

As the name suggests, this property sets the order in which the flex items are shown.

Eg:

```
<div style="order: 4;">1</div>
<div style="order: 3;">2</div>
<div style="order: 1;">3</div>
<div style="order: 5;">4</div>
<div style="order: 2;">5</div>
```

Output:

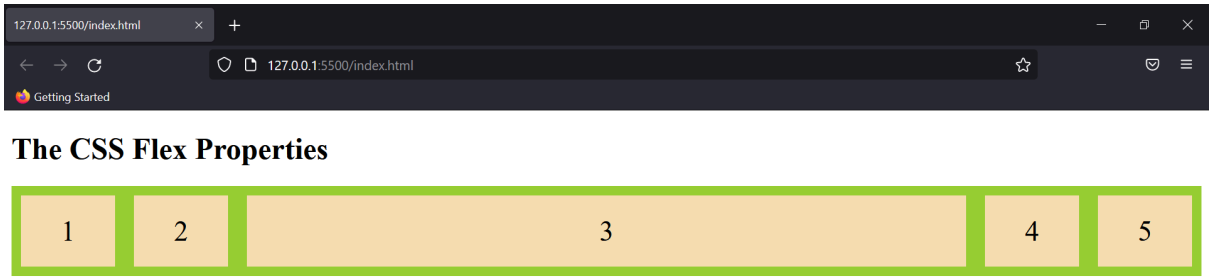

2. **Flex Grow & Shrink**

Decides the relative size of flex items. By default, this property is zero and thus items have the same size.

Eg:

```
<div>1</div>
<div>2</div>
<div style="flex-grow: 3;">3</div>
<div>4</div>
<div>5</div>
```

Output:




We can also use flex-shrink to decrease size of an element.

3. **Align Self**
   This property allows default alignment to be overridden for the individual flex items. Try adding inline CSS to see how this property is used.

---

## 2. CSS Grid

Just like FlexBox, CSS Grid with the use of rows and columns, make it easier to style website elements.

CSS display property allows two grid properties: Grid and Inline Grid. The elements placed in the grid container are called grid items.

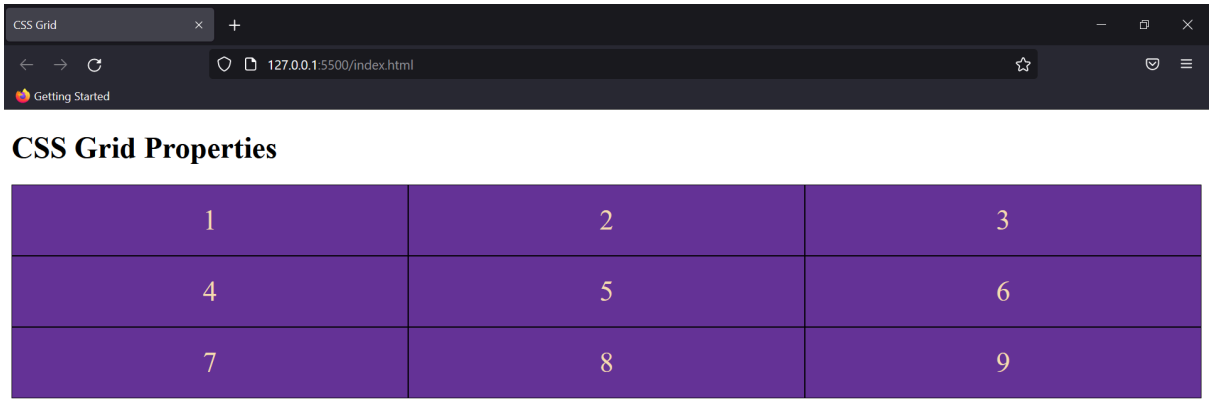
Most of the properties of Grid are similar to FlexBox.

Let’s look at how you can create a grid in CSS.

index.html

```
<!DOCTYPE html>
<html lang="en">

<head>
    <title>CSS Grid</title>
    <style>
        span {
            display: grid;
            grid-template-columns: auto auto auto;
            background-color: rebeccapurple;
        }

        div {
            color: wheat;
            padding: 20px;
            border: 1px solid black;
            font-size: 30px;
            text-align: center;
        }
    </style>
</head>

<body>
    <h1>CSS Grid Properties</h1>
    <span class="grid-container">
        <div class="grid-item">1</div>
        <div class="grid-item">2</div>
        <div class="grid-item">3</div>
        <div class="grid-item">4</div>
        <div class="grid-item">5</div>
        <div class="grid-item">6</div>
        <div class="grid-item">7</div>
        <div class="grid-item">8</div>
        <div class="grid-item">9</div>
    </span>
</body>
```

Output:



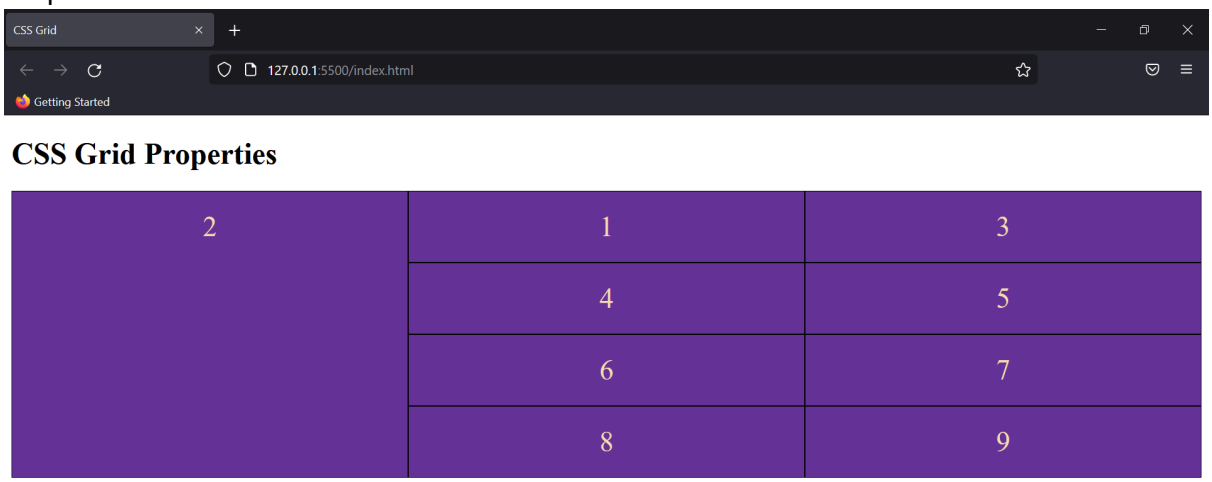

**Grid Row & Column Property**

This is just adjacent to the flex-grow property in Flex Box. It decides how many columns would the selected element be taking in the Grid.

Eg:

```
<div class="grid-item" style="grid-row: 1/5;">2</div>
```

Output:




This way you can style Grid dimensions to create Header, Footer, and other structures of your website.

---

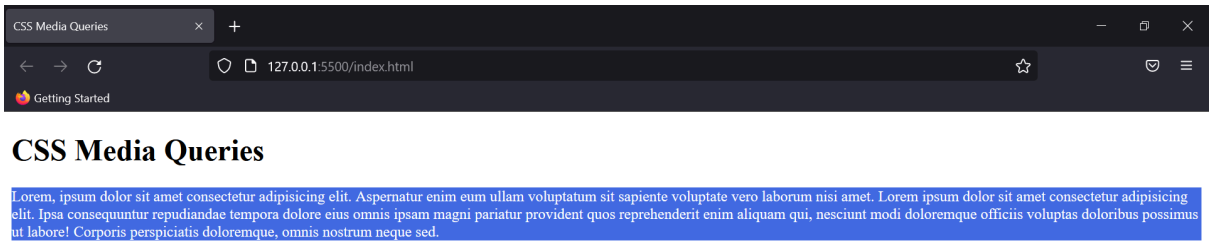
## 3. CSS Media queries

Media queries help to add responsiveness to the website by adding breakout points or when only a certain condition is true.

Eg:

```html
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>CSS Media Queries</title>
    <style>
        p {
            color: white;
            background-color: royalblue;
        }

        @media only screen and (max-width:800px) {
            p {
                color: yellowgreen;
            }
        }
    </style>
</head>

<body>
    <h1>CSS Media Queries</h1>
    <p>Lorem, ipsum dolor sit amet consectetur adipisicing elit. Aspernatur enim eum ullam voluptatum sit sapiente voluptate vero laborum nisi amet.
    Lorem ipsum dolor sit amet consectetur adipisicing elit. Ipsa consequuntur repudiandae tempora dolore eius
        omnis ipsam magni pariatur provident quos reprehenderit enim aliquam qui, nesciunt modi doloremque officiis voluptas doloribus possimus ut
        labore! Corporis perspiciatis doloremque, omnis nostrum neque sed.</p>
</body>

</html>
```

Output:




The output of this code would change when the dimension would be 800px.

# Features Are Used in Real Websites

- **Colors/Gradients** → Gmail background themes.
- **Navigation Bar** → Amazon, Flipkart, YouTube top navbars.
- **Box Model (cards)** → Facebook posts, Twitter tweets.
- **Buttons** → Amazon “Add to Cart” buttons.

- **Hover Transformations** → Flipkart product zoom, Instagram photo effects.
- **Animations** → Google Doodles, loading spinners on Netflix.
- **Tooltips** → YouTube video controls, Facebook icons.
- **Responsive Design** → Netflix, Zomato, Swiggy mobile layouts.