Introduction to Responsive Design

From Grid to Flex

What is Responsive Design?

Responsive Design is an approach to web design that makes web pages render well on a variety of devices and window or screen sizes. This ensures a **consistent user experience** across desktop, tablet, and mobile devices.

Why is it important?: Users access websites from various devices (smartphones, tablets, laptops, desktops).

- **Responsive websites** adjust their layout and content based on the **screen size** and **resolution** of the device.
- It allows for a better **user experience** and ensures **greater accessibility**.

The CSS display Property

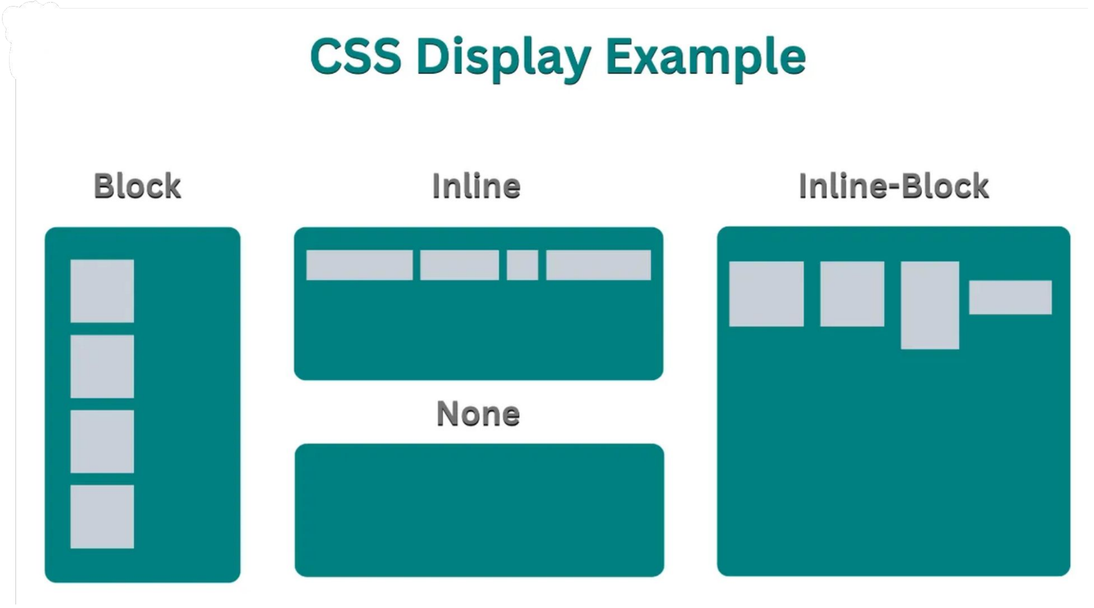
What is the display Property?

- The display property defines how an element is displayed on the webpage.
- It is used to set the layout behavior of an element, such as whether it behaves like a block or inline element.

The CSS display Property

Common Values:

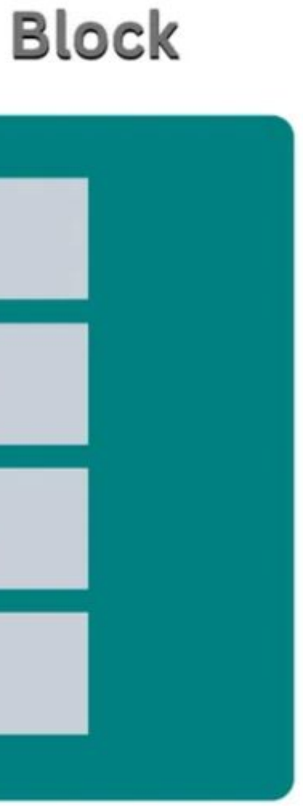
- block: The element takes up the full width available, pushing subsequent elements to the next line (e.g., <div>, <p>).
- inline: The element only takes up as much width as it needs and doesn't break the flow of the document (e.g., <span>, <a>).
- inline-block: A hybrid between block and inline; allows width and height adjustments while flowing inline.
- flex: Enables Flexbox layout.
- grid: Enables Grid layout.



Common display Property Values:block

block: The element takes up the full width available and starts on a new line. Block-level elements include <div>, <p>, and <h1>.

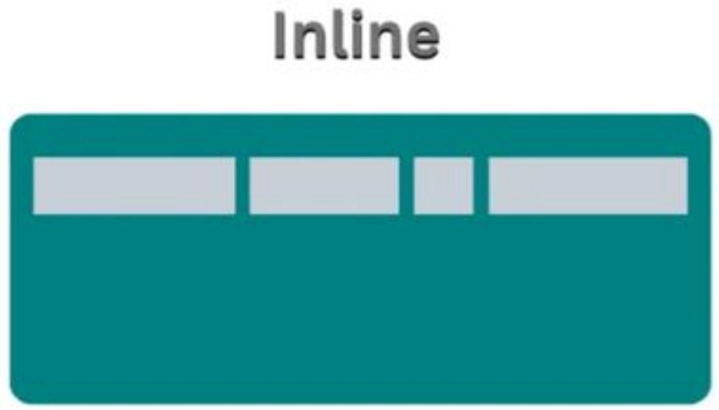
```
div {
    display: block;
  }
```



Common display Property Values: inline

inline: The element only takes up as much width as it needs and doesn't break the flow of the document. Inline elements include <span>, <a>, and <strong>.

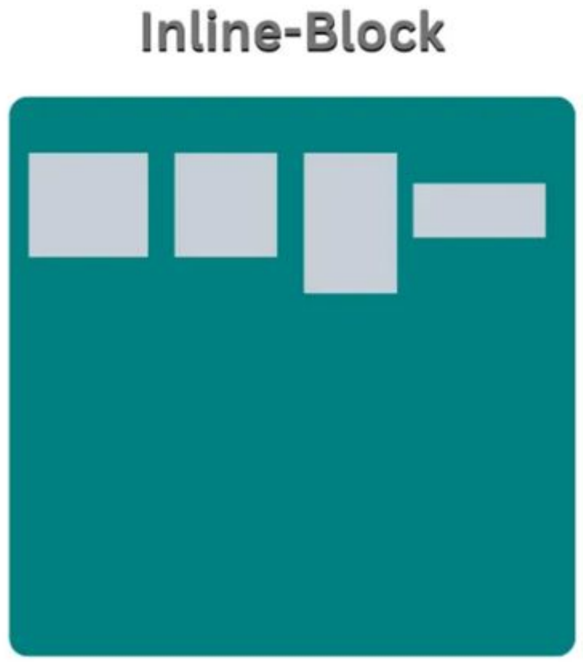
```
a {
    display: inline;
 }
```



Common display Property Values: inline-block

inline-block:A hybrid between block and inline. The element behaves like an inline element, but you can set its width and height.

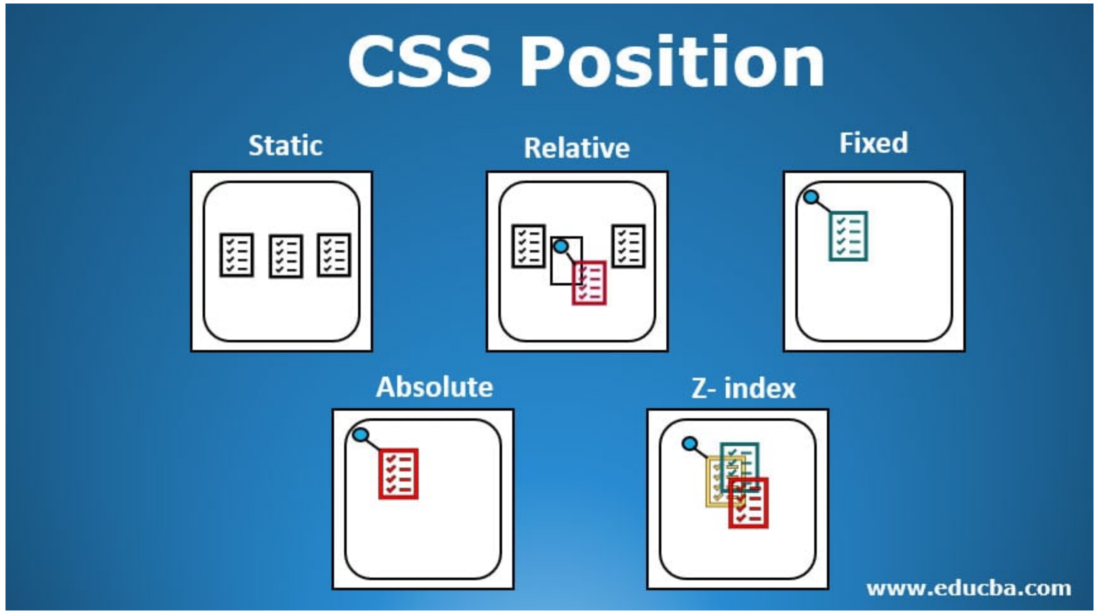
```
div {
    display: inline-block;
  }
```



Understanding the position Property in CSS

What is the position Property?

- The position property is used to control the positioning of an element on the webpage. It specifies how an element is positioned within its container or relative to other elements.



Values of the position Property :**static**

```
.static-example {
  position: static;
}
```

static (default): This is the default positioning for elements. They are positioned **according to the normal document flow**

Elements are placed based on the order in the HTML (top to bottom).

Values of the position Property :**relative**

```
.relative-example {
    position: relative;
    top: 20px; /* Moves the
element 20px down from its normal
position */
    left: 30px; /* Moves the
element 30px to the right */
  }
```

relative: The element is positioned **relative to its normal position** (i.e., where it would be in the normal document flow).

You can move the element from its original position using top, right, bottom, or left.

Values of the position Property: **absolute**

```
  .absolute-example {
    position: absolute;
    top: 50px; /* Moves the
element 50px from the top of its
nearest positioned ancestor */
    left: 100px; /* Moves the
element 100px from the left of
its nearest positioned ancestor
*/
  }
```

absolute: The element is positioned **relative to its nearest positioned ancestor** (an ancestor with position: relative, absolute, or fixed).

The element is **removed from the normal document flow** and can be precisely positioned using top, right, bottom, and left properties.

Values of the position Property: **fixed**

```
  fixed-example {
    position: fixed;
    bottom: 10px; /* Keeps the
element 10px from the bottom of
the viewport */
    right: 20px;  /* Keeps the
element 20px from the right side
of the viewport */
  }
```

fixed: The element is positioned **relative to the browser window** (viewport), and it stays in place even when the page is scrolled.

The element is **removed from the normal document flow**, and you can position it using top, right, bottom, and left. The element remains fixed in the same position as you scroll..

Values of the position Property: **sticky**

```
.sticky-example {
    position: sticky;
    top: 0; /* Makes the element
stick to the top of the viewport
once it's scrolled to that
position */
  }
```

sticky: The element is treated as relative until it reaches a defined scroll position, then it behaves like fixed.

How it works: This is commonly used for **sticky headers** that remain visible as the user scrolls down the page.

Flexbox

CSS layout

What is Flexbox?

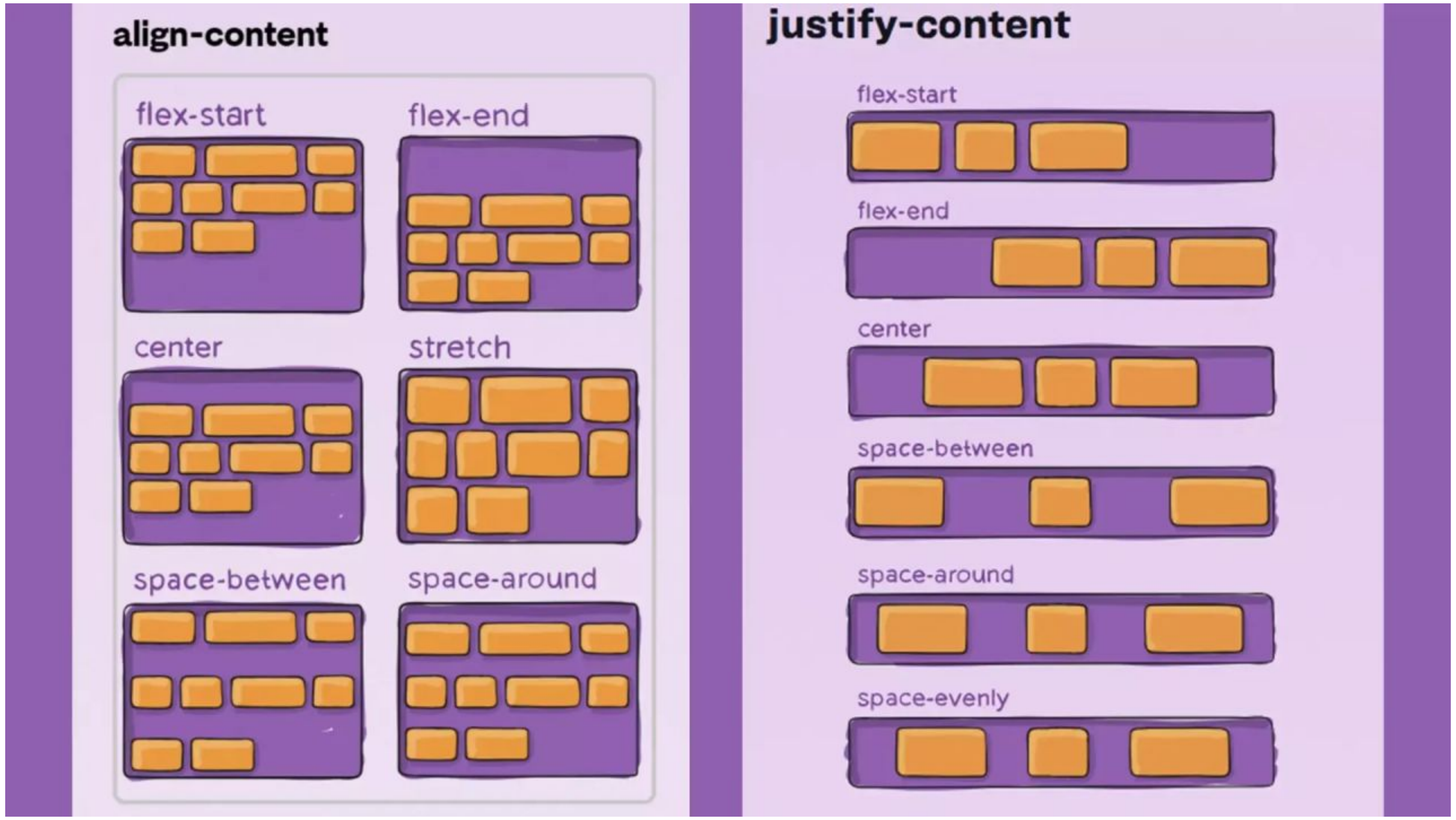
- **Flexbox (Flexible Box Layout)** is a CSS layout module designed to create more efficient layouts and alignments, especially for one-dimensional layouts (either rows or columns).
- It allows elements to grow, shrink, and distribute space in a container, making it easier to build complex layouts without using floats or positioning.

Why Use Flexbox?

•It simplifies the process of creating **responsive** layouts.

•It helps in **aligning** items both **vertically** and **horizontally**.

•It eliminates the need for **calculating widths**, making it easier to create flexible layouts.



Flexbox Container

What is a Flexbox Container?

- The Flexbox container is the parent element that holds all the flex items.
- To make an element a flex container, you apply the CSS property display: flex to the container.

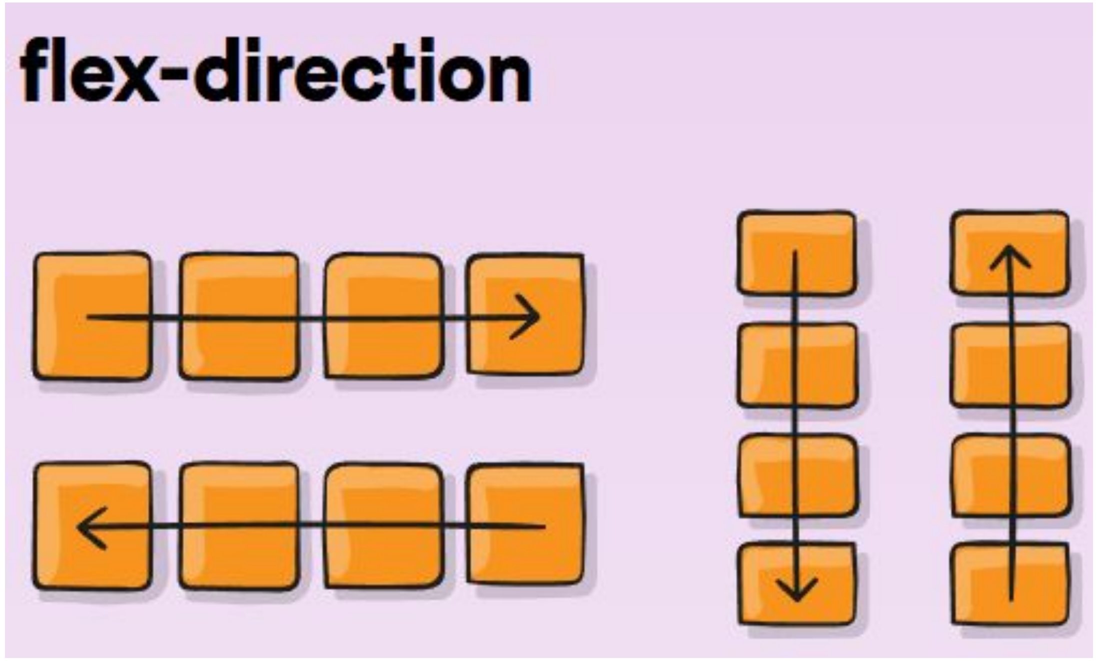
```
.container {
    display: flex;
 }
```

EMPOWERING
YOUNG ENTREPRENEURS

Flexbox Container Properties

flex-direction: Specifies the direction of the main axis (row, column).

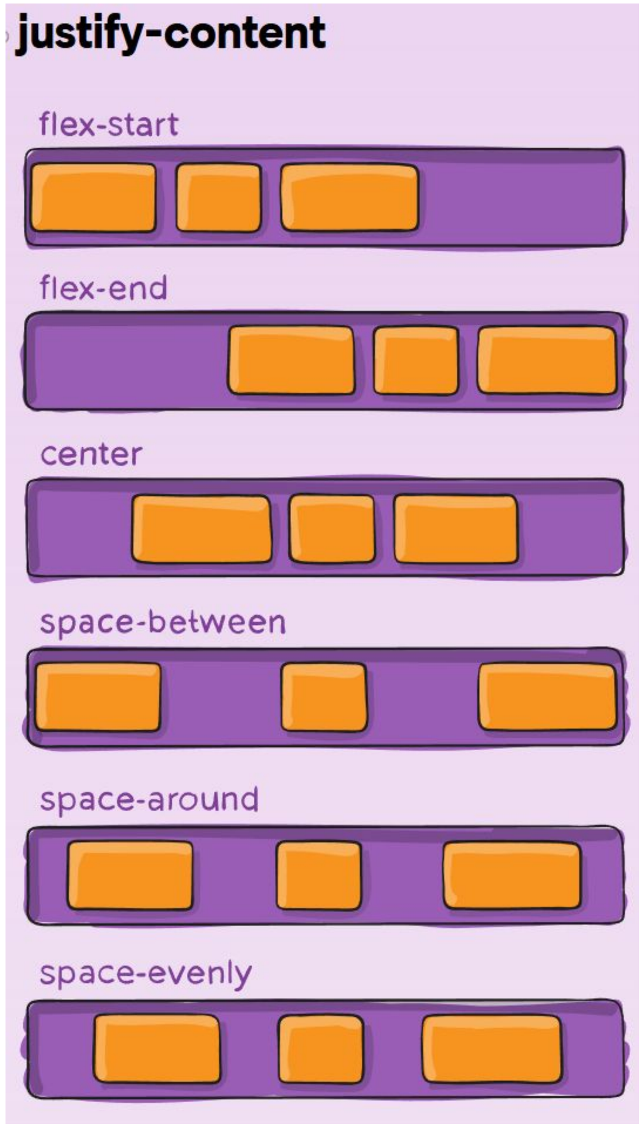
```
.container {
   flex-direction: row | row-reverse |
column | column-reverse;
}
```



Flexbox Container Propertie

justify-content: Aligns flex items along the main axis (horizontal by default).

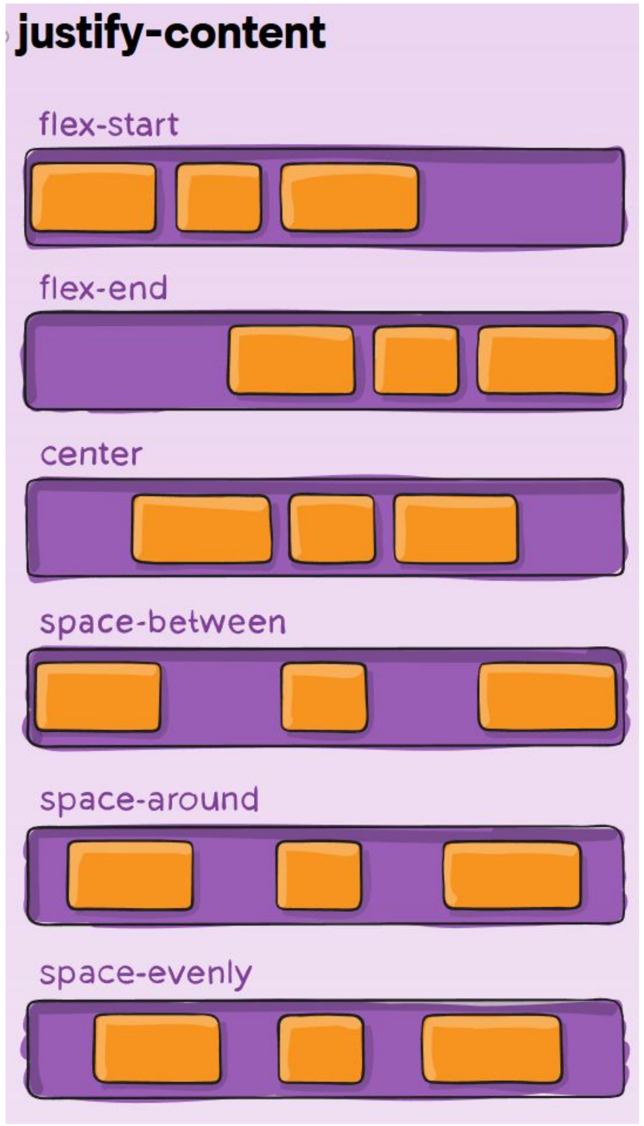
```
.container {
  justify-content: flex-start | flex-end | center
| space-between | space-around | space-evenly;
}
```



Flexbox Container Propertie

align-items: Aligns flex items along the cross axis (vertical by default).

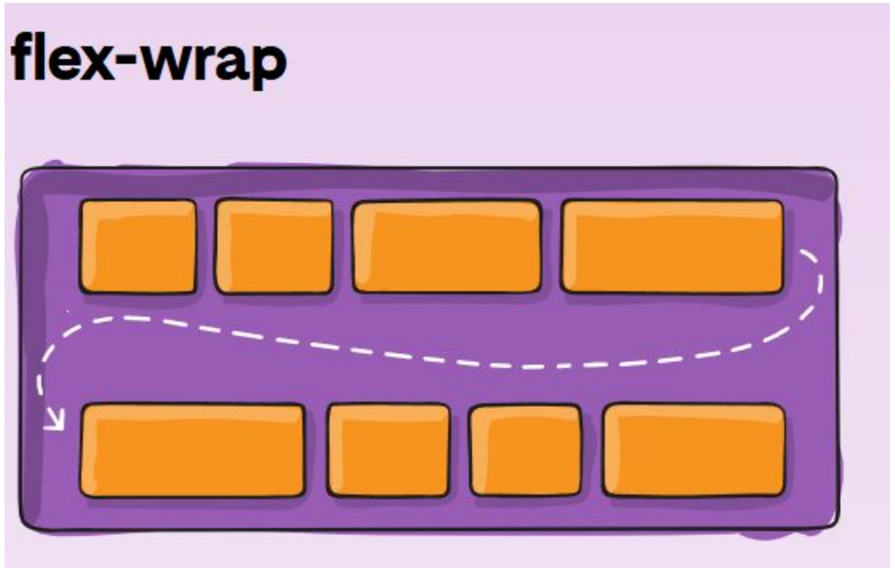
```
.container {
  align-items:  flex-start | flex-end |
center | space-between | space-around |
space-evenly ;
}
```



Flexbox Container Properties

Flex-wrap: Determines if items should wrap onto multiple lines.

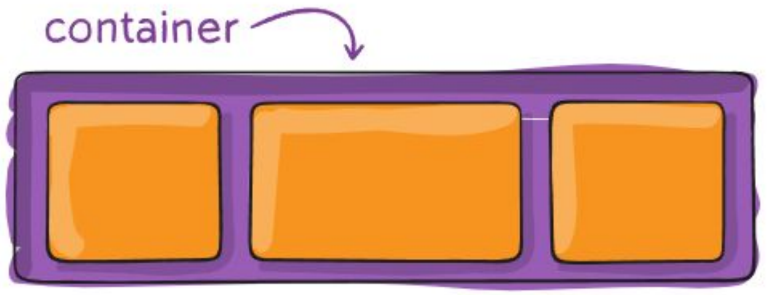
```
    .container {
     flex-wrap: nowrap | wrap | wrap-reverse;
  }
```



Basic Flexbox Syntax: **Flex container**

```
.flex-container {
    display: flex;
    justify-content: space-between; /* Aligns items horizontally */
    align-items: center; /* Aligns items vertically */
  }
```

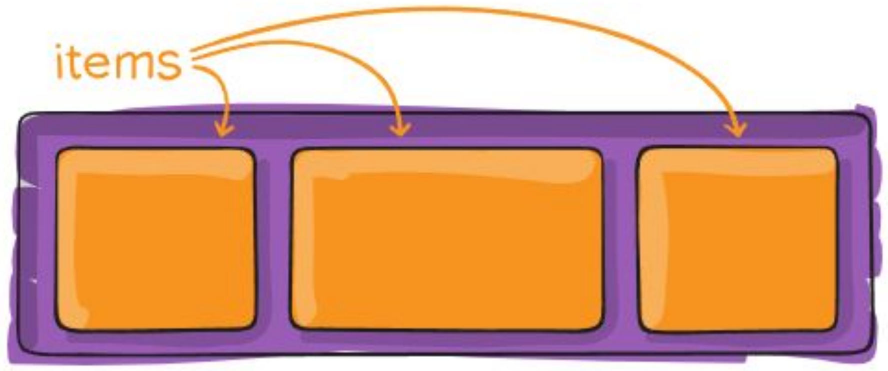
•



Flexbox Items

What are Flexbox Items?

- **Flexbox items** are the direct children of a **flex container**.
- By default, **flex items** will be arranged in a **row** (horizontally). You can change this behavior using flex-direction.



Flex Item Properties:

flex-grow: Defines how much a flex item should grow relative to the rest of the items. (Default is 0, meaning items won't grow.)

flex-shrink: Defines how much a flex item should shrink relative to the others when there's not enough space. (Default is 1, meaning items will shrink.)

flex-basis: Sets the initial size of a flex item before any growing or shrinking occurs.

align-self: Allows a specific flex item to override the align-items property and align itself differently.

```
 .container {
    display: flex;
  }

  .item {
    flex-grow: 1; /* Items
will grow to fill available
space */
  }
```

Class Activity 1: Flexbox Layout Challenge

Objective:

- The goal of this activity is to practice and learn the basics of **Flexbox** by creating a simple webpage layout. You will work with **Flexbox container properties** and **Flexbox item properties** to arrange and align items within a container.

Class Activity 2: Flexbox Froggy Challenges

Objective:

- The goal of this activity is to enhance your understanding of **Flexbox** by completing all the challenges on the **Flexbox Froggy** website. This interactive game will help you learn how to use various **Flexbox properties** in a fun and engaging way.

nstructions:

1. **Visit the Flexbox Froggy Website**:
   1. Go to the following URL: https://flexboxfroggy.com/.
   2. This website will present you with a series of **24 interactive challenges** that teach you how to use **Flexbox** to align and position items on the webpage.
2. **Complete All 24 Challenges**:
   1. Start from the first challenge and complete each one in sequence.
   2. Each challenge will teach you a new **Flexbox property** and how to use it to position the frogs correctly in the game.