**Flexbox**

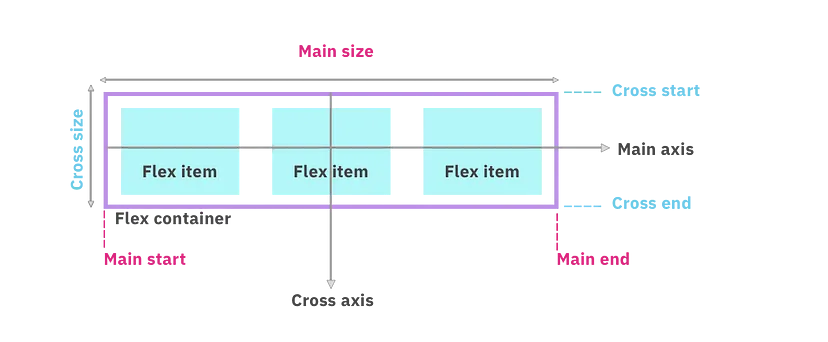
we'll be delving into the world of Flexbox in CSS. It's a powerful layout system that simplifies the process of building flexible and responsive web designs.

**Introduction:**

Flexbox is a CSS layout module that provides a flexible way to align and distribute elements within a container. It allows you to create dynamic layouts, adjust the spacing between elements, and control their behavior in different screen sizes.

Flexbox, or CSS3 Flexible Box, is a super lightweight way to lay out UI components on a web page. Not only is it easy (and fun) to learn, but it makes designing for responsive screen sizes painless.

**Flexbox Terminology**



**Main axis**: This is the primary axis along which flex items are distributed. This will change depending upon the value of flex-direction.

**Main-start and Main-end**: Flex items are placed within a container beginning at main-start and resting at main-end.

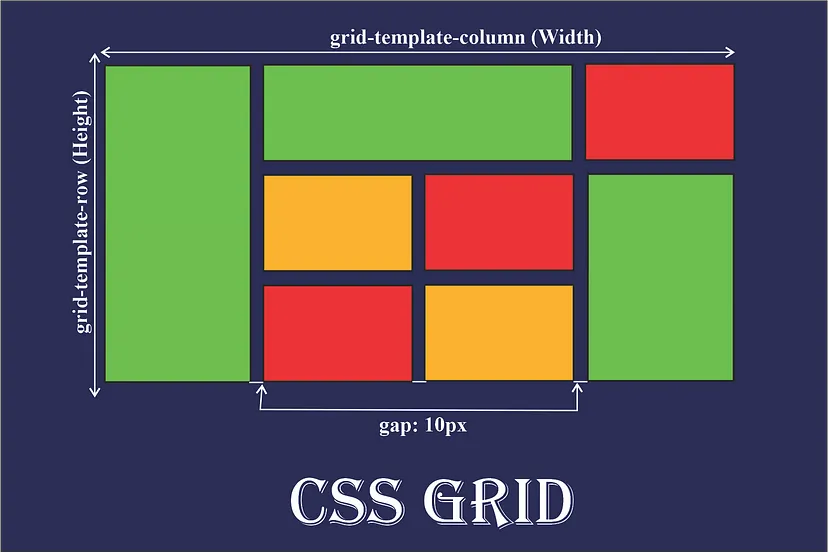
**Main-size**: A flex item’s width or height is the main size. This depends upon which is the main direction. The main size property will either be ‘width’ or ‘height’.

**Cross axis**: This is the axis that is perpendicular to the main axis. The direction depends on the main axis direction.

**Cross-start and Cross-end:** Items are placed beginning at the cross-start side of the flex container and move towards the cross-end side.

**Cross-size:** The width or height of a flex item, which again depends on the main direction. This property will be either ‘width’ or ‘height’.

In the further topics we will understand the properties of flex-box in-depth.



Here only we have walked through a brief intro to the grid. Moving forward, we’ll go deep into each section.

**Why and When do we use Grid?**

**Why we use Grid:-**

CSS grid comes in handy when we excel at dividing a page into major regions or defining the relationship in terms of size, position, and layer, between parts of a control built from HTML primitives.

**When should we use Grid:-**

**- When you have a complex design to implement:-**

The two-dimensional layout system here is perfect to create a complex design, we can use it in our favour to create more complex and maintainable web pages.

**- When you need to have a gap between block elements:-**

Without using margin properties, we can define the gap between our rows or columns very easily in the grid-gap property.

**- When you need to overlap elements:-**

You just need to use the grid-column and grid-row properties and you can have overlapping elements very easily.

**- When you need a layout-first setup**

Imagine we have the layout structures, using the rows and the columns together we can position the elements in the way we want.

**Difference between Flexbox and Grid Property?**

**01) Layout**

**Grid**:- CSS Grid Layout, is a two-dimensional grid-based layout system with rows and columns.

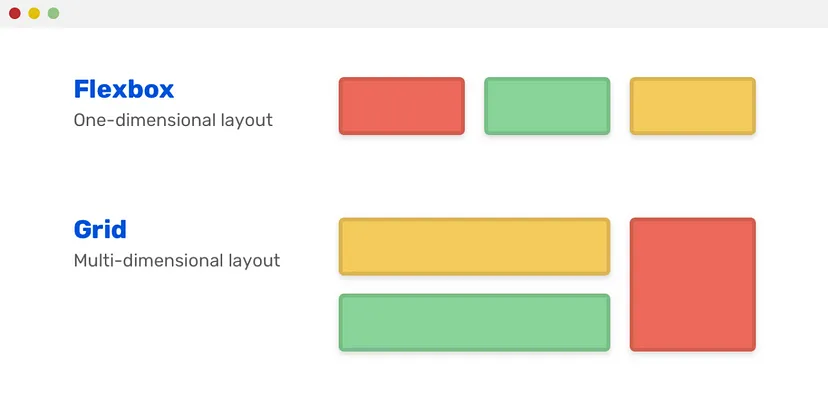
**FlexBox**:- CSS Flexbox, is a one-dimensional layout system.

Before moving to the next point let us briefly explain what is two-dimensional and one-dimensional layouts?

**Two-dimensional Layout:-** If you can take the components in your layout and draw a grid over them, complete with rows and columns then what you have is a two-dimensional layout.

**One-dimensional Layout:-** A one-dimensional layout is a layout in which you position items on individual rows and columns without respect to the next row or column.

**Note**:- This simply means Flexbox can work on either row or columns at a time, but Grids can work on both.

**02) Overlap**

* **Grid**:- Grid easily permits the overlapping of items. Thus we have the freedom to place items across grid lines or even within the same grid cell.
* **FlexBox**:- Overlap of flex items can be achieved but not without its cons. To overlap flex items, one would have to use negative margins or absolute positioning. Which invariably removes items from the flex layout.

**03) Dimensionality and Flexibility**

* **Grid**:- Grid allow flexible widths as a unit of length. This balances the limitations in Flex.
* **FlexBox**:- Flexbox offers greater control over alignment and space distribution between items.

**04) Alignment**

* **Grid**:- CSS Grid deploys fractional measure units for grid fluidity and auto-keyword functionality to automatically adjust columns or rows.
* **FlexBox**:- Flex Direction allows developers to align elements vertically or horizontally, which is used when developers create and reverse rows or columns.

**05) Item Management**

* **Grid**:- Grid supports both implicit and explicit content placement. Its inbuilt automation allows it to automatically extend line items and copy values into the new creation from the preceding item.
* **FlexBox**:- Flex Container is the parent element while Flex Item represents the children. The Flex Container can ensure balanced representation by adjusting item dimensions. This allows developers to design for fluctuating screen sizes.

**NOW LET US MOVE TO THE MAIN CONTENT OF THIS ARTICLE**

**Display Property:**