Introduction to CSS Flexbox

Things getting flexy

Agenda

Introduction to CSS Flexbox

Flex Container Properties

Flex Item Properties

Flex Practical Demo Examples

What is CSS Flexbox

What is CSS Flexbox

CSS Flexbox Layout Module is a CSS3 layout mode that provides an easy, professional and advanced clean way to arrange child elements/items within a container.
is designed for one-dimensional layout model that makes it easy to:

- Design flexible and efficient layouts
- Distribute space among items
- Control alignments of objects/elements/items in given container
- Flexbox layout is most appropriate to the components of an application, and small-scale layouts
- Flexbox is a new CSS display type designed to craft CSS layouts in a much easier way
- Control the position, size, and spacing of child elements relative to parent container and other child elements

Layout modes

Before CSS Flexbox there were 4 layout modes/models:

1. Block (for different sections in a webpage)
2. Inline (for text and spans in the same line)
3. Table (for two-dimensional tabular data)
4. Position (for the explicit position of an element)

Pre-Flexbox we have used:

- Different positional properties like fixed, absolute to set alignment at the exact required place
- Floats and clear fixes to create navigation, detailed section
- Fixed heights columns to show equality
- Above mentioned layout modes does not provide enough flexibility to create/build or design professional/nested/modern layouts
- we need to include another CSS mechanism like Floats, Vertical alignment of text/elements and other hacks with Margin, Padding to create the accurate/desired layout
- CSS Flexible Box Layout Module (Flexbox), makes it easier to design flexible responsive layout structure without using float or positioning

Why Flexbox?

- Flex/CSS Flexbox layout provides lots of flexibilities while creating complex layouts like:
- No CSS Float used
- arrange items from left to right or top to bottom and vice versa
- adjust the spacing between objects
- Position and alignments of items
- order and placements of various elements

Why Flexbox?

- improve the items alignment, directions and order in the container even when they are with dynamic or even unknown size
- equal height columns
- ability to modify the width or height of its children to fill the available space in the best possible way on different screen sizes
- Flexbox is a new CSS display type designed to craft CSS layouts in a much easier way
- Control the position, size, and spacing of child elements relative to parent container and other child elements

Why Flexbox?

- CSS flexbox works great responsively (RWD - Responsive Web Design)
- Responsive and Mobile friednly
- Bootstrap 4/5 is built-on with Flex layout Model

Important Terminology

Let's learn some of the Important Terminology/concepts related to CSS flexbox to get proper understandings and how it works. There are main three terms or entities while dealing with flexbox:

1. Flex Container (Parent/Container to hold sub-items)
2. Flex Items (Child/sub-items)
3. Flexbox axis (Main axis [horizontal] and Cross axis [vertical])

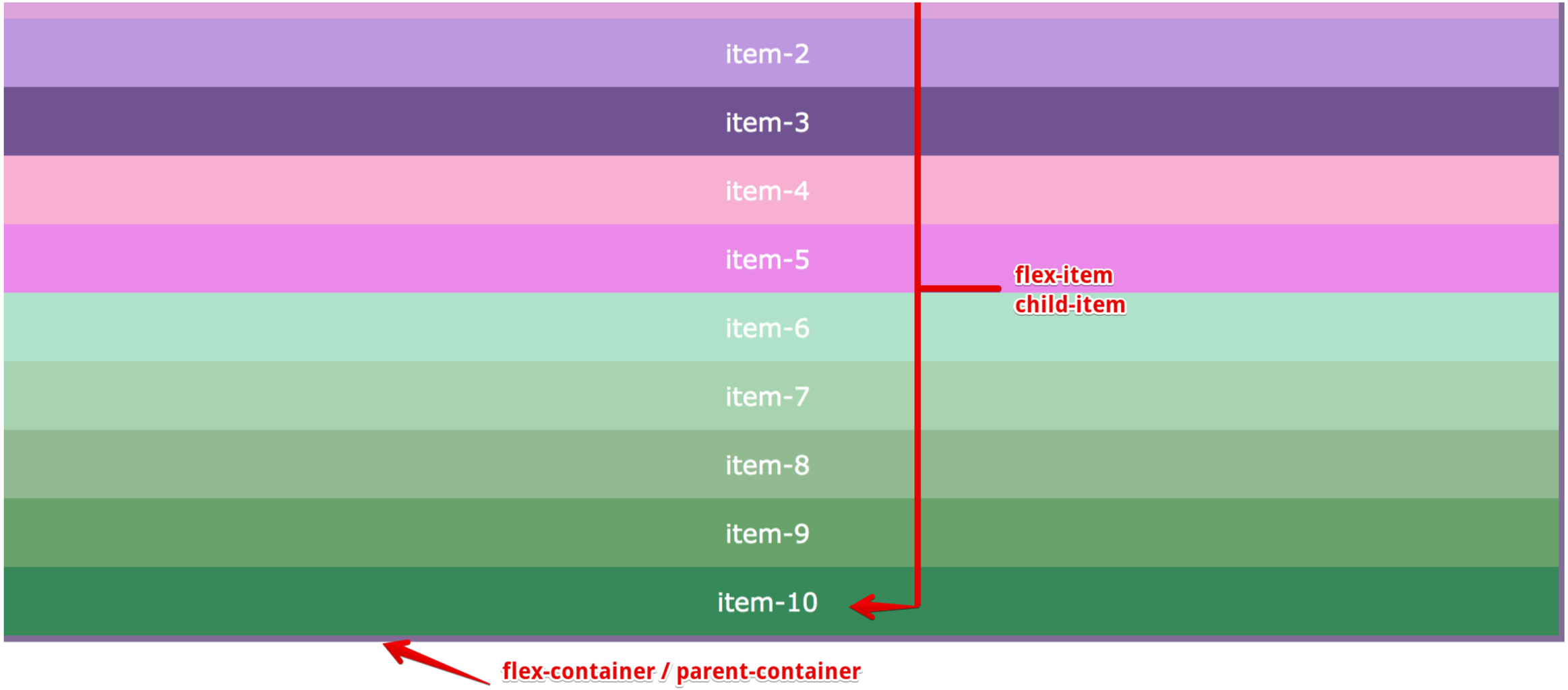
Important Terminology

To start with Flexbox just apply a display type of flex to the parent container

Syntax & Example - link

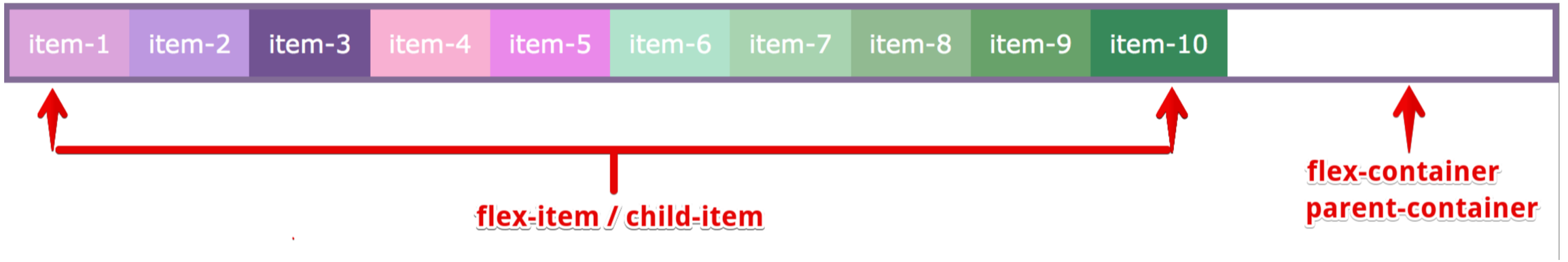
```
<div class="main-container">
  <div class="item item-1">item-1</div>
  <div class="item item-2">item-2</div>
  <div class="item item-3">item-3</div>
  <div class="item item-4">item-4</div>
  <div class="item item-5">item-5</div>
  <div class="item item-6">item-6</div>
  <div class="item item-7">item-7</div>
  <div class="item item-8">item-8</div>
  <div class="item item-9">item-9</div>
  <div class="item item-10">item-10</div>
</div>
```

Important Terminology



Important Terminology

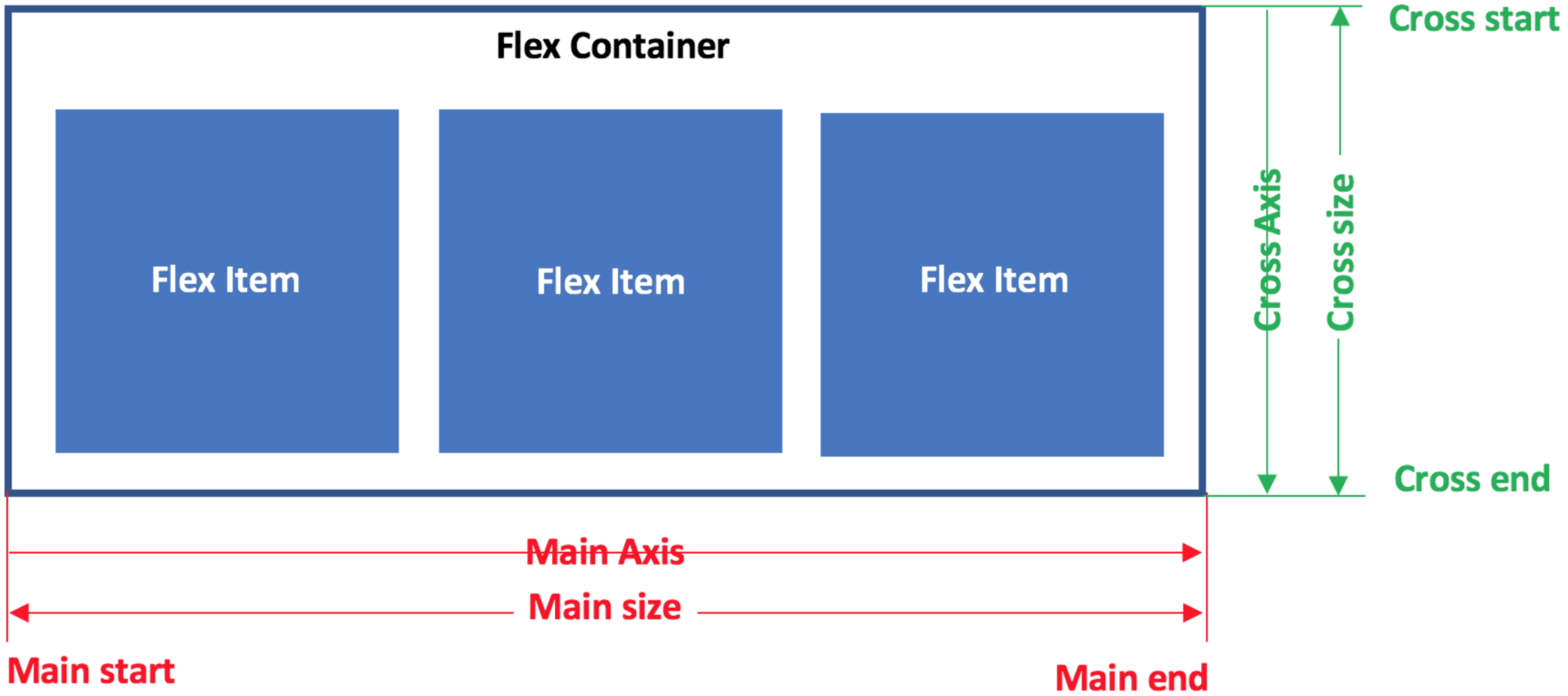
```
<style type="text/css">
  .main-container {
    border: 4px solid #826a98;
    display: flex; /* block level flex container */
  }
</style>
```



Flexbox axis

1. Main Axis
    - By default, main/primary axis runs Left to Right
    - Its denoted with terms like the main start and main end
    - The length between the main start to the main end is known as the main size
2. Cross Axis
    - By default, cross-axis runs Top to Bottom
    - Its denoted with terms like cross start and cross end
    - The length between cross start to cross end is known as cross size

Flexbox axis



Flex Container Properties

Flex Container Properties

- display
  - It defines the Flex Container inline or block
  - An important and mandatory property to work with Flexbox
- flex-direction
  - Defines direction for flex items to be placed inside Flex Container
- flex-wrap
  - Set wrapping of items within Flex Container (overflow item placement)
- flex-flow
  - Short-hand property for a combination of flex-direction and flex-wrap

Flex Container Properties

- justify-content
  - Defines alignment of items along the main axis
- align-items
  - How flex items layout/laid out with the cross axis

Flex Display

- To create a flex container, we set the value of the container's display property to flex or inline-flex
- It enables a flex context for all its direct children

display property creates either a block-level or inline-level flex container:

- display: flex;
  - block level flex, covers 100% width
- display: inline-flex;
  - inline-level, same line flex, covers only required width as per items width

Flex Display

```
.container {
  /* display: flex; */ /* block level flex container */
  /* display: inline-flex; */ /* inline flex container */

  display : flex | inline-flex;
}
```

Let's take an example

Flex Direction

- The flex-direction property decides/defines how flex items will stack inside flex container along the main axis, by default it's Main Axis runs from Left to Right
- Flex items can be laid out in two main directions, like rows horizontally or like columns vertically

  - **flex-direction:** row;
  - Default direction is row-wise ie. left to right alignment
  - With row direction, the flex items are stacked in a row from left-to-right
  - **flex-direction**: row-reverse;
  - Right to left alignment (item alignment will start from right to left - item 1 will start from right)
  - **flex-direction:** column;
  - The column value stacks the flex items vertically (from top to bottom)
  - **flex-direction:** column-reverse;
  - bottom to top (item alignment will start from bottom to top - item 1 will start from the bottom)

Flex Direction

Syntax:

```
.container {
  display: inline-flex;

  flex-direction: row | row-reverse | column | column-reverse;
}
```

Let's take an example

Flex Wrap

By default, all items in Flex Container try to fit themselves in a container in one row/column. If enough space not available, items simply overflow - with the help of **flex-wrap** property we can set the overflow direction

- The **flex-wrap** property specifies whether the flex items should wrap or not
- The flex-wrap property simply determines how items are wrapped when the parent container runs out of space
- flex-wrap property controls the wrapping of flex items within the container:
- flex-wrap: nowrap; (default is row nowrap)
- flex-wrap: wrap; (wraps overflow items to bottom/next row or right/next column)
- flex-wrap: wrap-reverse; (wraps overflow items to top row or left column)

Flex Wrap

```
.container {
  display: inline-flex;

  flex-wrap: nowrap | wrap | wrap-reverse;
}
```

Let's take an example

Flex Flow

- flex-flow property is shorthand for flex-direction and flex-wrap properties
- By default flex-flow property is set to row and nowrap which is default value of properties like flex-direction and flex-wrap

```
.container {
  /* flex-flow: <flex-direction> && <flex-wrap> */
  flex-flow: row wrap;
}
```

Justify Content

- justify-content defines and set the alignment of flex items along with the main axis
- It helps distribute extra free space leftover when either all the flex items on a line
- The default value of justify-content is flex-start which simply means start all flex items from left side exactly the place where the container starts (justify-content: flex-start;)
- syntax:

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

Justify Content

- justify-content: flex-start;
- Default alignment for items, start from/packed toward the left side of the container
- The flex-start value aligns the flex items at the beginning of the container (this is the default)
- justify-content: flex-end;
- Align item to right or end of the container main axis
- justify-content: center
- Align item at the center of the main container
- center value aligns the flex items at the center of the container

Justify Content

- justify-content: space-between
- Equally, distribute the items [(apart from first-start and last-end item) in between items will be distributed with equal gap/space]
- justify-content: space-around
- space-around property displays the flex items with space before, between, and after the lines(first-start and last-end items have 50% space/gap) in between items will be distributed with 100% gap/space] [space to the edges will be 50%] and spacing between any two items are 100%
- justify-content: space-evenly
- All flex item is distributed with even/equal space-gap
- All flex items are distributed evenly/equally so that the spacing between any two items (and the space to the edges) is also equal

Row wise justify-content

```
.main-container {
  border: 4px solid #826a98;
  display: flex; /* block level flex container */

  /*  justify-content: flex-start; */
  /* justify-content: flex-end; */
  /* justify-content: center; */
  /* justify-content: space-between; */
  /* justify-content: space-around; */
  justify-content: space-evenly;
}
```

Column wise justify-content

```
.main-container {
  border: 4px solid #826a98;
  display: flex; /* block level flex container */

  flex-direction: column;
  height: 600px;
  /* justify-content: flex-start; */
  /* justify-content: flex-end; */
  /* justify-content: center; */
  /* justify-content: space-between; */
  /* justify-content: space-around; */
  justify-content: space-evenly;
}
```

Align Items

- align-items property defines default behavior of flex items, it simply means how flex items align along the cross axis (vertically) of the container
- align-items is similar to justify-content but works in a perpendicular direction to the main-axis
- syntax:

```
.container {
  align-items: stretch | flex-start | flex-end | center | baseline;
}
```

Align Items

- The align-items property will align the items on the cross axis:
- align-item: stretch;
- Default alignment for flex-items, covers/occupy 100% height of the container
- align-item: flex-start;
- All flex items squeezed/stacked or align at start ie. from the top of the container (items line up at the start of the flex container)
- align-item: flex-end;
- All flex items start/stacked from bottom or aligned/pushed at the bottom
- align-item: center;
- Align item vertically center/middle of the container (items are stacked to the center/middle of the container)

Row wise align-item

Syntax & Example

```
<div class="main-container">
  <div class="item item-1">item-1</div>
  <div class="item item-2">item-2</div>
  <div class="item item-3">item-3</div>
  <div class="item item-4">item-4</div>
  <div class="item item-5">item-5</div>
  <div class="item item-6">item-6</div>
  <div class="item item-7">item-7</div>
  <div class="item item-8">item-8</div>
  <div class="item item-9">item-9</div>
  <div class="item item-10">item-10</div>
</div>
```

Code

Column wise align-item

Syntax & Example

```
<div class="main-container">
  <div class="item item-1">item-1</div>
  <div class="item item-2">item-2</div>
  <div class="item item-3">item-3</div>
  <div class="item item-4">item-4</div>
  <div class="item item-5">item-5</div>
  <div class="item item-6">item-6</div>
  <div class="item item-7">item-7</div>
  <div class="item item-8">item-8</div>
  <div class="item item-9">item-9</div>
  <div class="item item-10">item-10</div>
</div>
```

Code

Flex Item Properties

Flex Item Properties

- Let's look into the some of the important properties used with and applicable to Flex Item or Child Elements (usually known as Items, present inside Flex container):
- order -
- order property controls the order in which flex item appears in Flex Container
- flex-grow -
- flex-grow property defines and sets the ability for flex item to grow/spread if necessary by adding some spaces
- flex-shrink -
- flex-shrink property defines and sets the ability for a flex item to shrink/squeeze if necessary

Flex Item Properties

- flex-basis
- flex-basis property signifies the initial main size of a flex item
- flex
- flex property short-hand property for combination of flex-grow, flex-shrink and flex-basis
- align-self
- align-self property allows and sets the alignment of an individual flex item

Order

- order property specifies and denotes the order of the flex items/children of a flex container appear inside the flex container
- order property controls the order in which flex item appears in Flex Container
- order property accepts integer value and controls the order of items in the flex container
- The default order of all flex item is 0, flex item aligns by items with lower order to higher order in left to the right direction
- order with greater value aligns himself at the very end of flex container (Item with order 1 or more value will place at the end), so the order placement is lower to higher number from left to right direction
- Items with same order number align as per appearance/order in HTML source code
- syntax:

Order

```
.item {
 order: <integer>; /* default is 0 */
}
```

Code

Flex Grow

- flex-grow property denotes and specifies how much space or amount of space flex item to take if necessary
- flex-grow dictates what amount of the available space inside the flex container the item should take
- The flex-grow property specifies how much a flex item will grow relative to the rest of the flex items
- By default, all the flex item have flex-grow value as 0, so they don't occupy any extra-empty or remaining space available in the container
- flex-grow size is always relative to the size of other flex items in flex container (size of the flex-grow item is calculated as per the size of other flex items)

Flex Grow

```
.item {
  flex-grow: <number>; /* default 0 */
}
```

Code

Flex Shrink

- flex-shrink property denotes and specifies an ability of flex-item to shrink if necessary
- The flex-shrink property specifies how much a flex item will shrink relative to the rest of the flex items
- By default, all the flex item have flex-shrink factor value as 1, so they shrink/reduce their width by default, (assigning flex-shrink factor value as 0 represents no adjustments/ no shrinking ie. fixed width - which prevents shrinking)
- flex-shrink size is always relative to the size of other flex items in flex container (size of the flex-shrink item is calculated as per the size of others flex item)

syntax:

```
.item {
  flex-shrink: <number>; /* default 1 */
}
```

Flex Shrink

```
.item {
  color: #ffffff;
  font-size: 1rem;
  padding: 0.75rem;
  text-align: center;
  flex-shrink: 1; /* default 1, so all items will shrink */
}
```

Click for code

Flex Shrink

- flex-shrink property denotes and specifies an ability of flex-item to shrink if necessary
- The flex-shrink property specifies how much a flex item will shrink relative to the rest of the flex items
- By default, all the flex item have flex-shrink factor value as 1, so they shrink/reduce their width by default, (assigning flex-shrink factor value as 0 represents no adjustments/ no shrinking ie. fixed width - which prevents shrinking)
- flex-shrink size is always relative to the size of other flex items in flex container (size of the flex-shrink item is calculated as per the size of others flex item)

syntax:

```
.item {
  flex-shrink: <number>; /* default 1 */
}
```