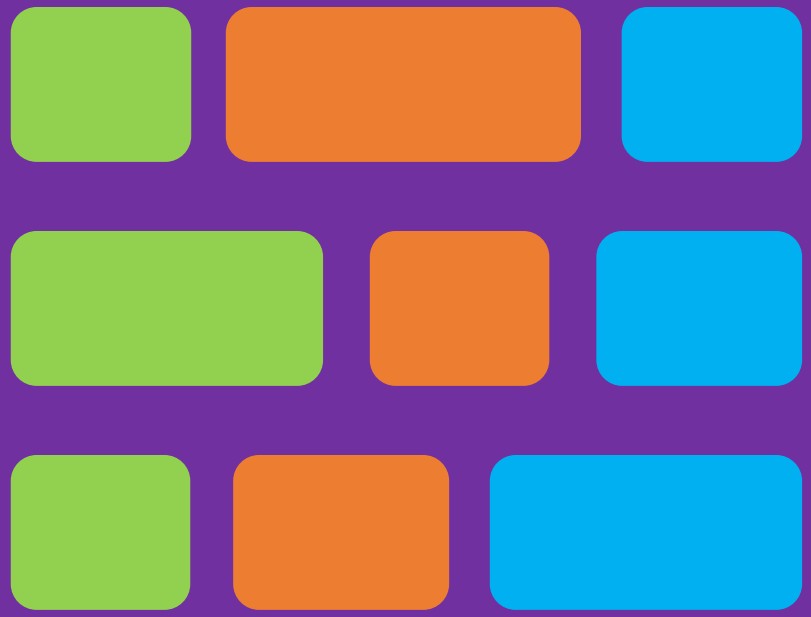
CSS3 Flexbox



Learn CSS3 Flexbox The Right Way

Siavash Sattari

Front-end Developer

Section 1 :

Getting Started With Flexbox

Lecture 1.3 :

Introduction

Get started with FLEXBOX :

- What is Flexbox ?
- Browser Support for Flexbox
- Setup Starter Files
- Our First Flexbox Layout

Section 1 :

Getting Started With Flexbox

Lecture 2.3 :

What is Flexbox ?

What Is Flexbox ?

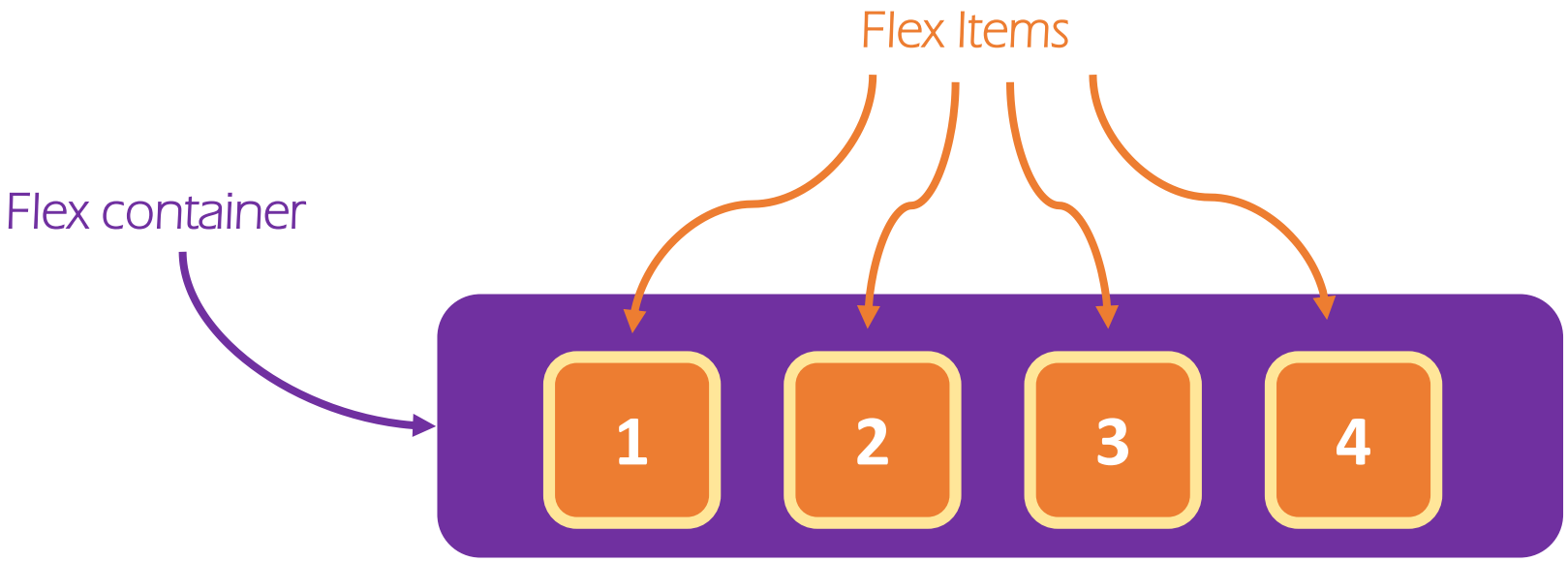
- Flexbox is a new layout module in CSS3 designed to help us create CSS layouts much easier.
- Flexbox layout model provides a more efficient way to layout , align and distribute space among elements within our document - even when the viewport and the size of our elements is dynamic or unknown .
- The main idea behind the flex layout is to give the container the ability to change it’s elements width , height and order to best fill the available free space .
- A flex container expands elements to fill available free space or shrink them to prevent overflow .
- Flexbox is a one-dimensional layout model which controls the layout based on a row or on a column but not together at the same time .
- Flexbox really makes it easier to create flexible responsive layouts without using float or positioning .
- Flexbox is the Future!

Why Do We Need Flexbox ?

- Flexbox is the tool that let's us forgot about using :
  - Table layouts
  - Floats
  - Clearfix hacks
  - Positioning
- With new CSS3 Flexbox model , we can now write more less and maintainable code for various common layout tasks such as :
  - Vertical centering
  - Equal height columns
  - Take up the whole space
  - Simpler grids

Flexbox consists of two main components :

- **Flex Container :** The parent element in which Flex items are contained .
- **Flex Items :** The direct child elements within a Flex container .



Flexbox is a set of properties :

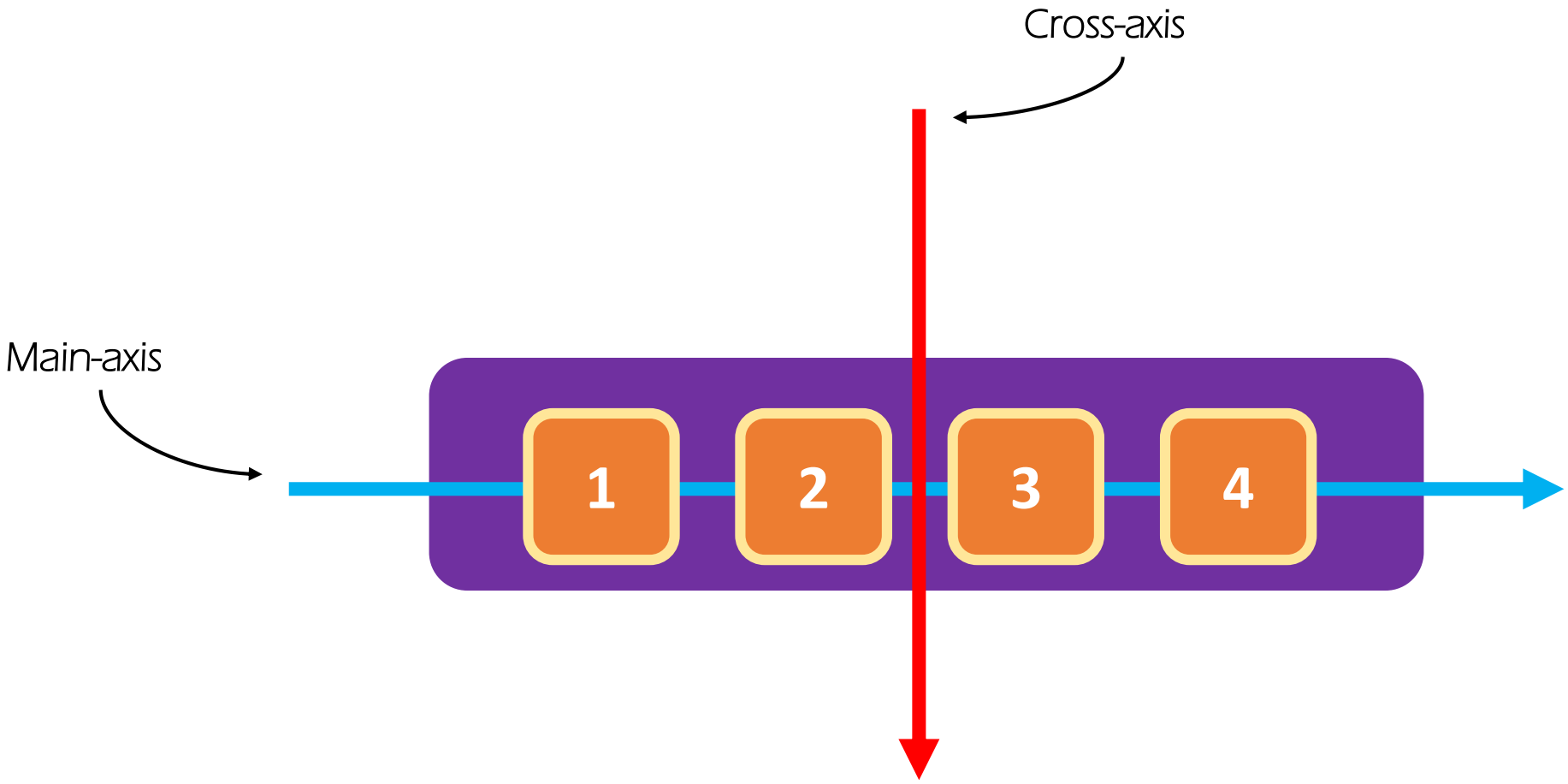
Flex Container Properties

1. flex-direction
2. flex-wrap
3. flex-flow
4. justify-content
5. align-items
6. align-content

Flex Item Properties

1. order
2. align-self
3. flex-grow
4. flex-shrink
5. flex-basis
6. flex

Main-axis & Cross-axis :



Section 1 :

Getting Started With Flexbox

Lecture 3.3 :

Browser Support For Flexbox

Browser Support For Flexbox :

CSS Flexbox is supported in all the modern browsers

Section 2 :

Working On Flex Container

Lecture 1.10 :

Introduction

Working on Flex Container :

- display - display : flex ; **VS** display : inline-flex
- main-axis and Cross-axis
- flex-direction
- flex-wrap
- The shorthand ‘flex-flow’ property
- justify-content
- align-items
- align-content
- Flex container properties overview

Section 2 :

Working On Flex Container

Lecture 2.10 :

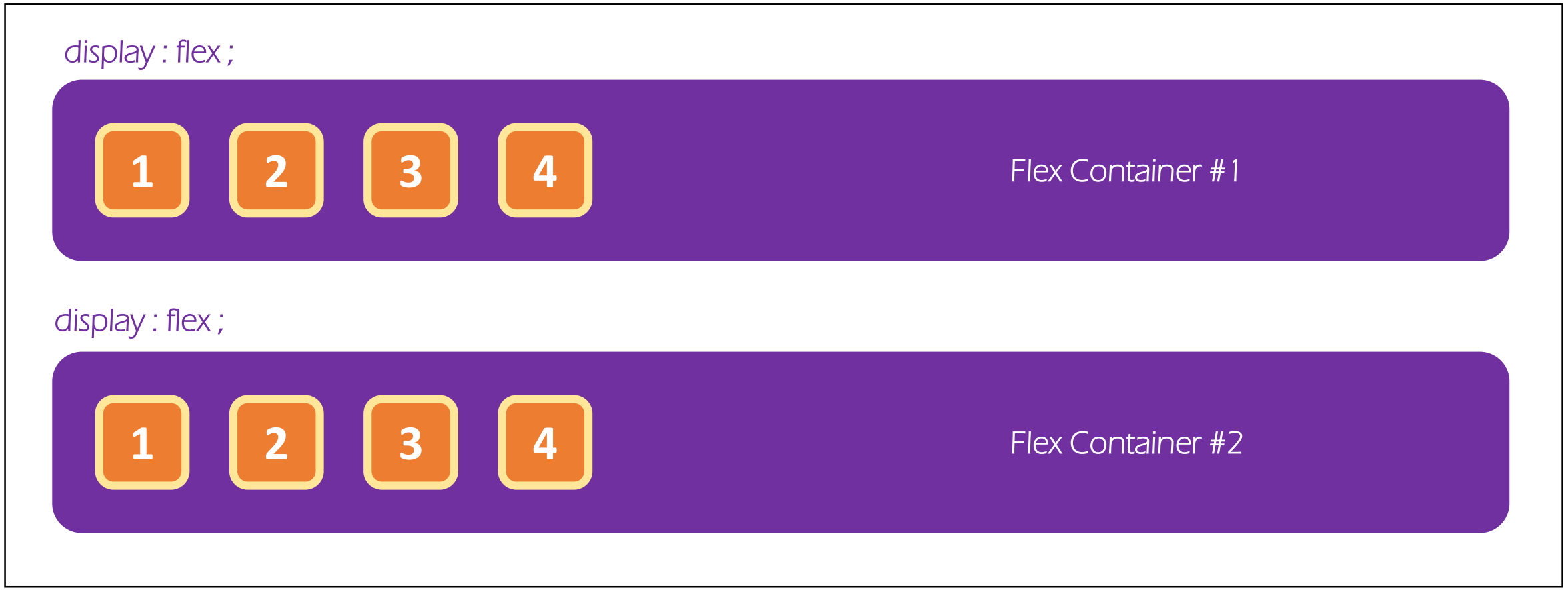
display : flex VS display : inline-flex

Creating a Flex Container :

- Flexbox layout activated by declaring the **display** property :

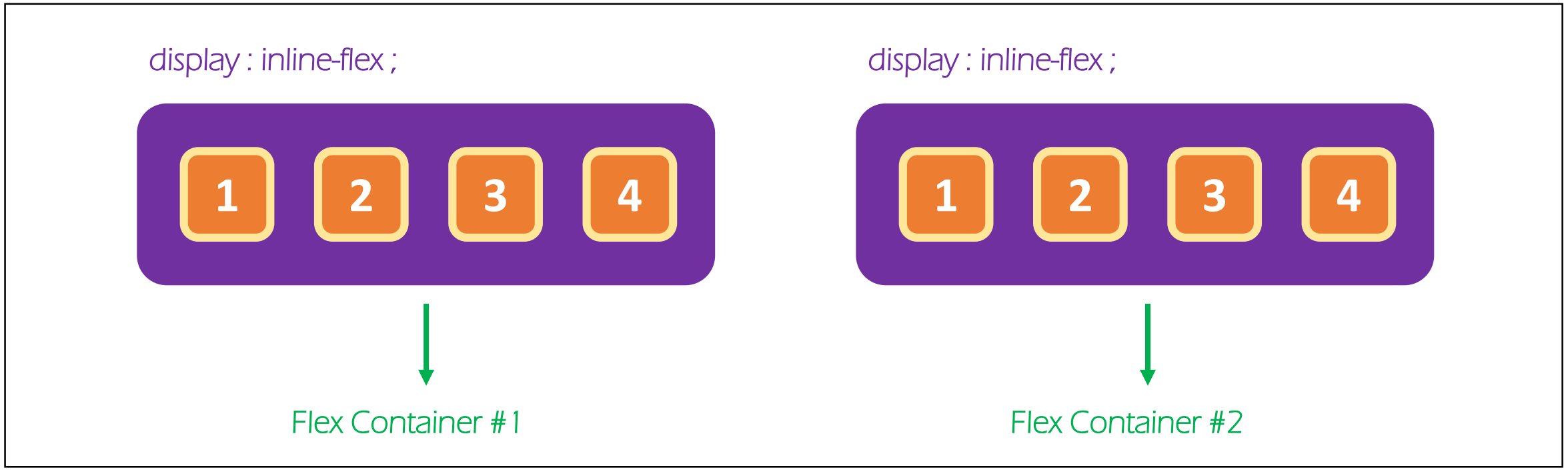
  - **display : flex ;** - The element becomes a **block-level** flex container .
  - **display : inline-flex ;** - The element becomes an **inline-level** flex container .

display : flex



Note : Both flex containers take 100% width of screen .

display : inline-flex



Note : Inline flex containers do not take 100% width of screen .

Section 2 :

Working On Flex Container

Lecture 3.10 :

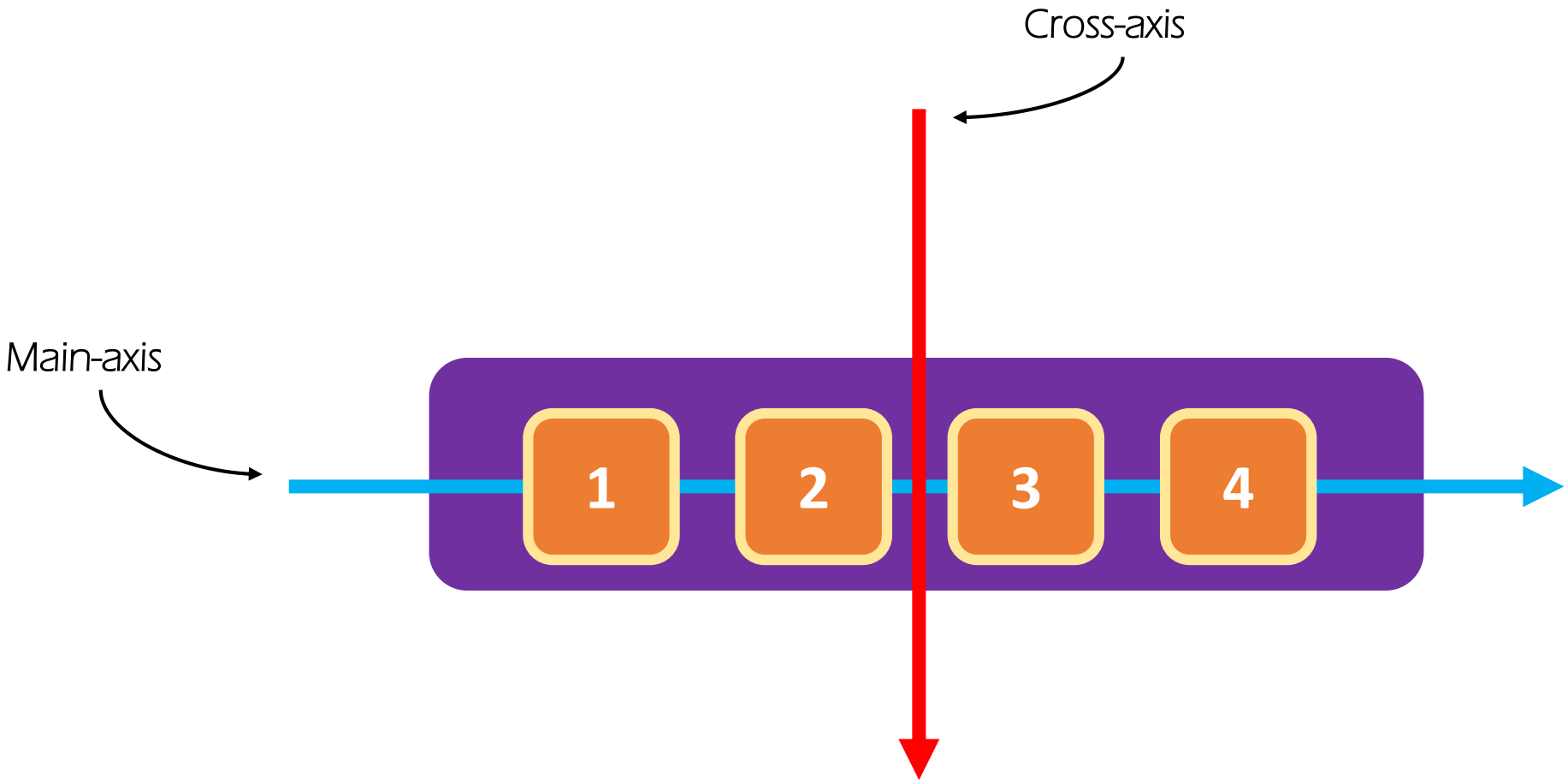
Main-Axis & Cross-Axis

The Two Axis Of Flexbox :

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

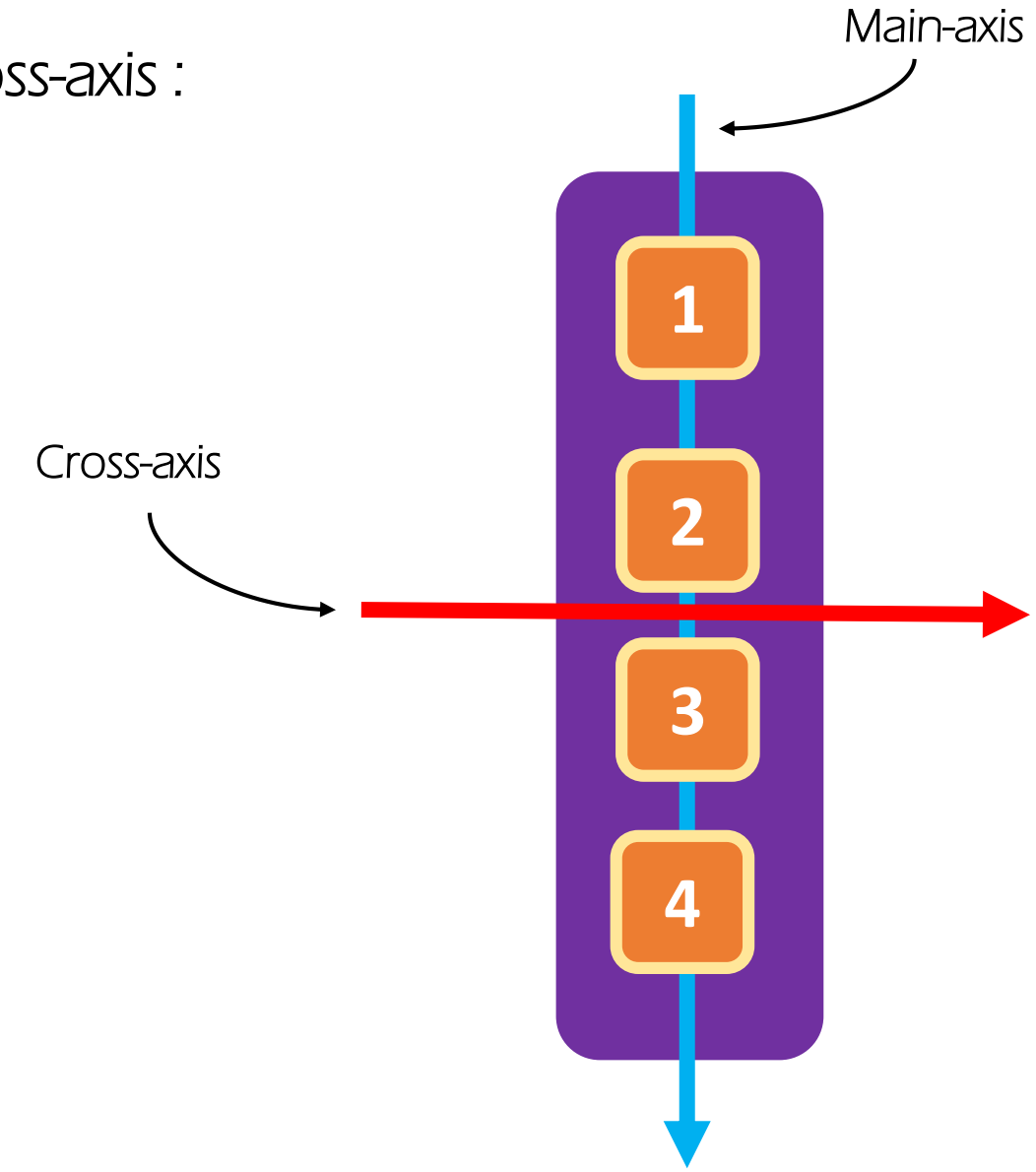
- **2. Cross-axis :** This is the axis that is perpendicular to the main axis . The direction depends on the main axis direction .

Main-axis & Cross-axis :



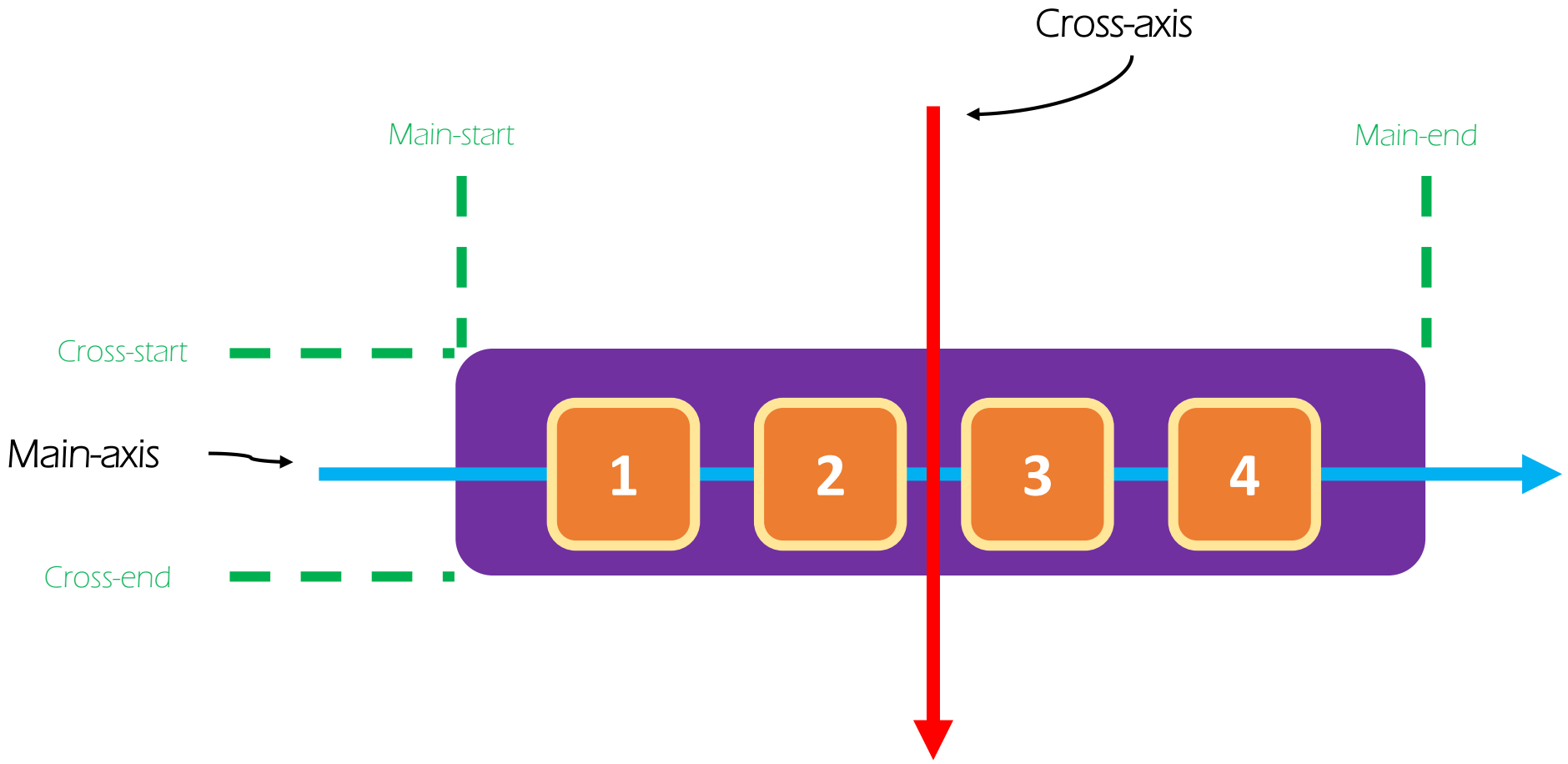
By default , Flex items laid out horizontally .

Main-axis & Cross-axis :



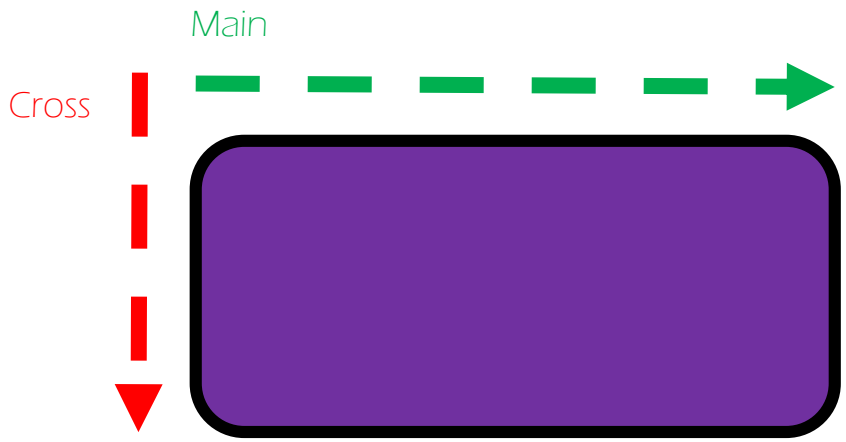
Flex items laid out vertically .

Main-axis & Cross-axis :

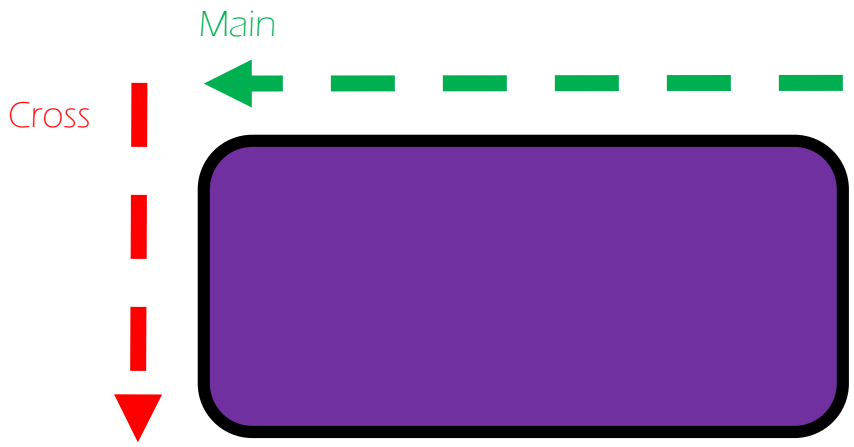


Flex items laid out horizontally .

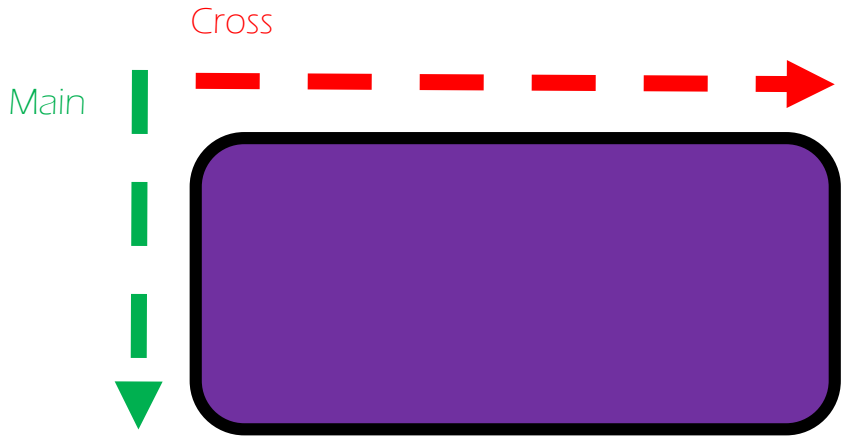
Main-axis & Cross-axis :



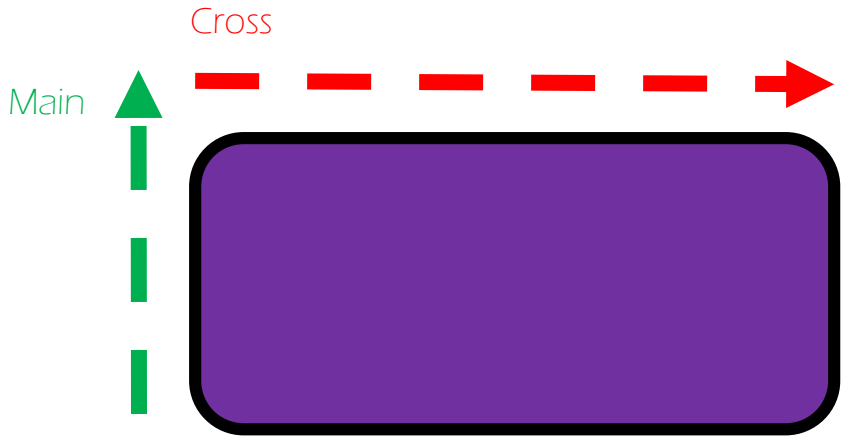
(Left to Right , Top to Bottom)



(Right to Left , Top to Bottom)



(Top to Bottom , Left to Right)



(Bottom to Top , Left to Right)

Section 2 :

Working On Flex Container

Lecture 4.10 :

flex-direction Property

What Is flex-direction ?

- The flex-direction allows to set the direction that how flex items are placed within flex container .
- It establishes the main-axis which determines the direction that flex items are laid out in .

flex-direction :

- The **flex-direction** property sets direction for flex items within flex container . It accepts 4 values :

  - **row** - (Default) : Main-axis runs from left to right .
  - **column** - Main-axis runs from top to bottom .
  - **row-reverse** – Main-axis runs from right to left .
  - **column-reverse** – Main-axis runs from bottom to top .

- Syntax for **flex-direction** property :

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

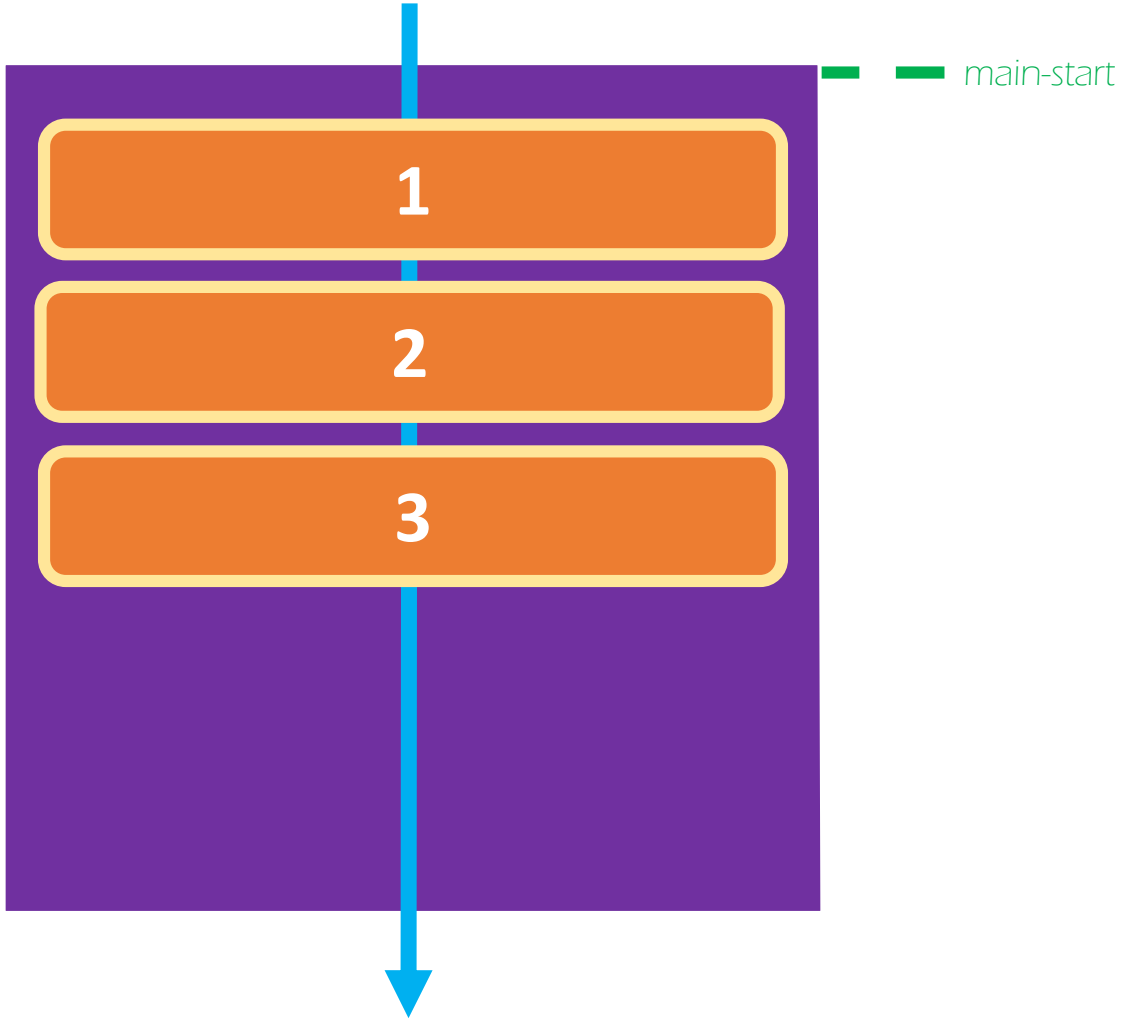
flex-direction : row ;

Main-axis runs from left to right

flex-direction : row-reverse ;

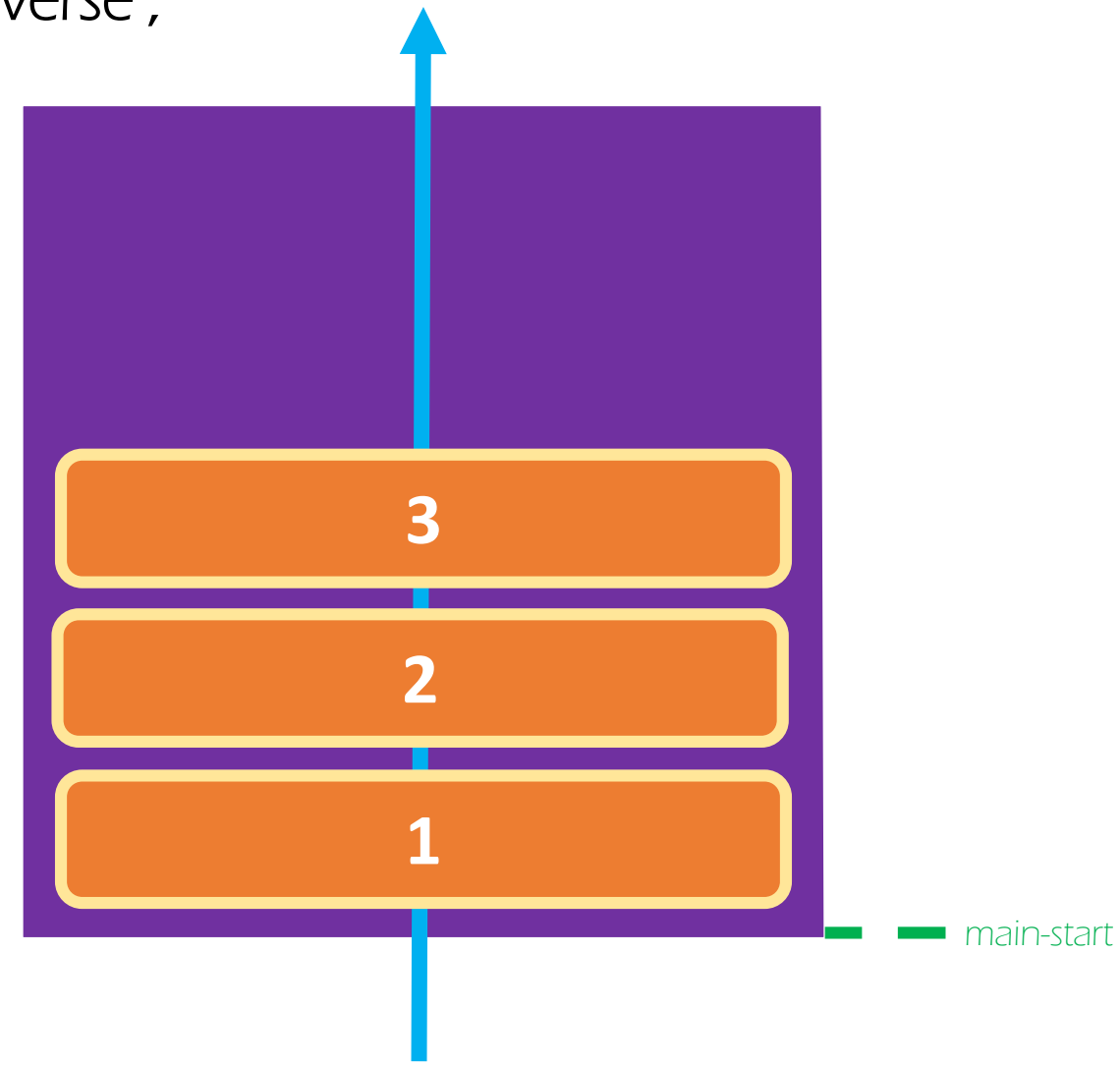
Main-axis runs from right to left

flex-direction : column ;



Main-axis runs from top to bottom

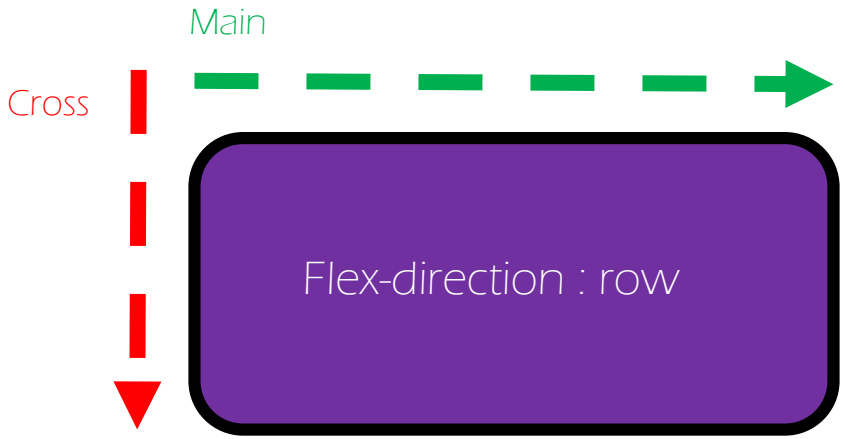
flex-direction : column-reverse ;



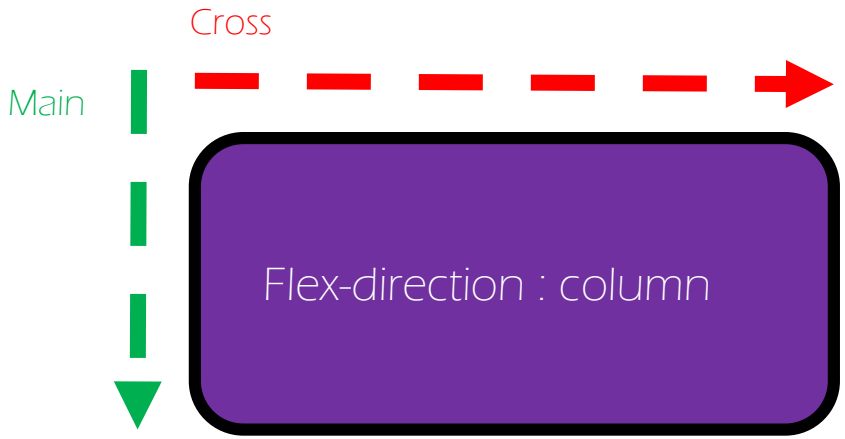
Main-axis runs from bottom to top

Main-axis & Cross-axis :

Normal



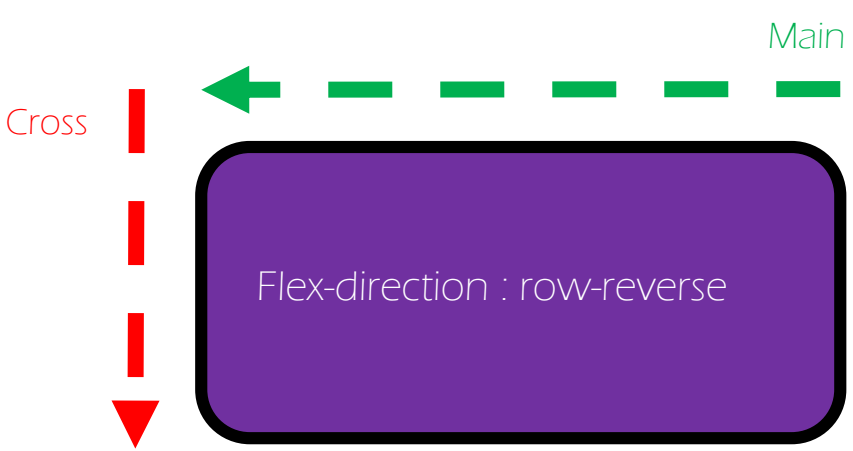
(Left to Right , Top to Bottom)



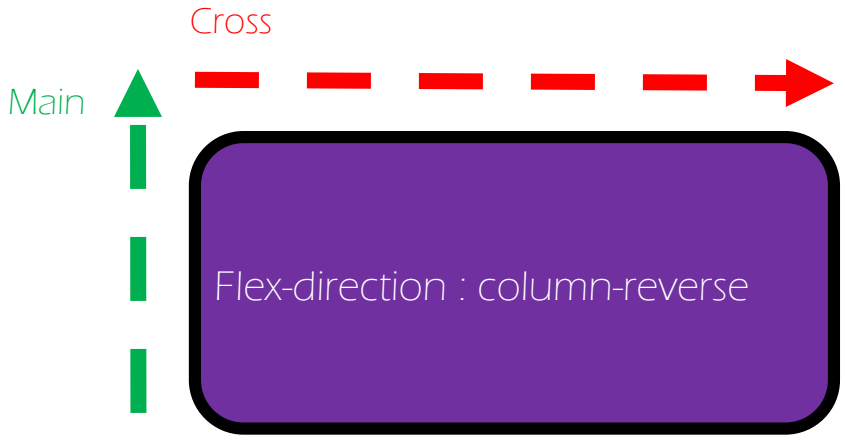
(Top to Bottom , Left to Right)

Main-axis & Cross-axis :

Reverse



(Right to Left , Top to Bottom)



(Bottom to Top , Left to Right)

Section 2 :

Working On Flex Container

Lecture 5.10 :

flex-wrap Property

What Is flex-wrap ?

- The flex-wrap defines whether the flex container is **single-line** or **multi-line** , and it also defines the direction of the cross-axis , which determines the direction new lines are stacked in .

flex-wrap :

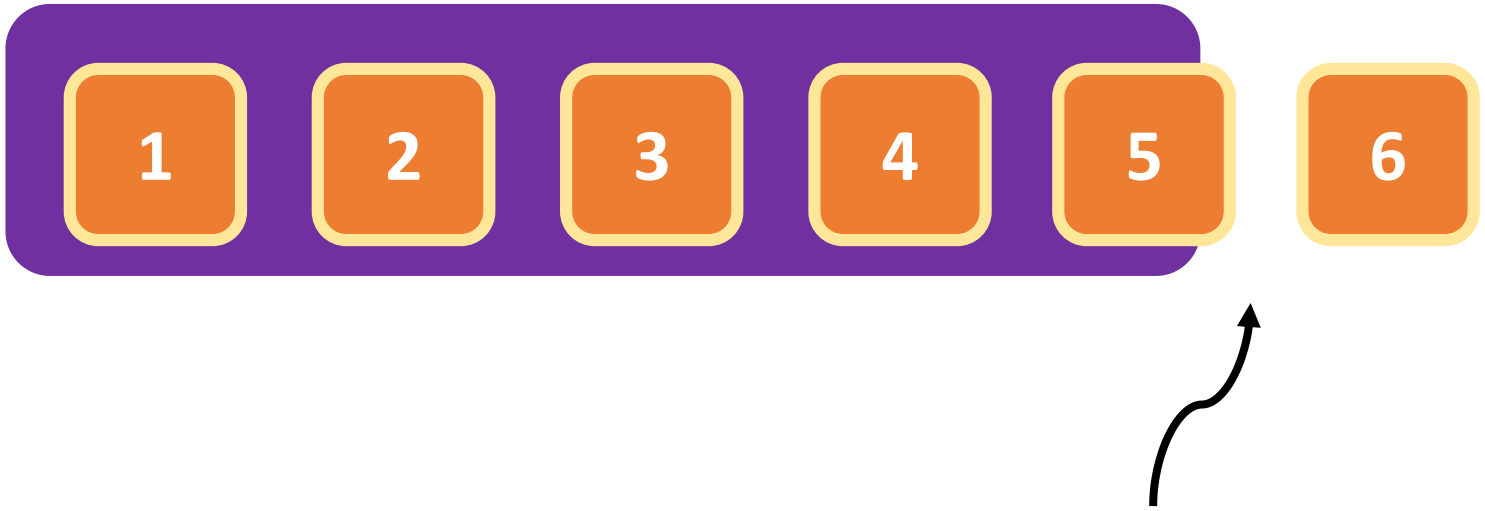
- The **flex-wrap** property specifies whether the flex items should wrap or not . It accepts 3 values :

  - **nowrap** - This is the default . The flex container creates only **one** row or column .
  - **wrap** - The flex container is multi-line & Flex items will wrap onto multiple lines .
  - **wrap-reverse** - Same as wrap ; but in opposite direction .

- Syntax for flex **flex-wrap** property :

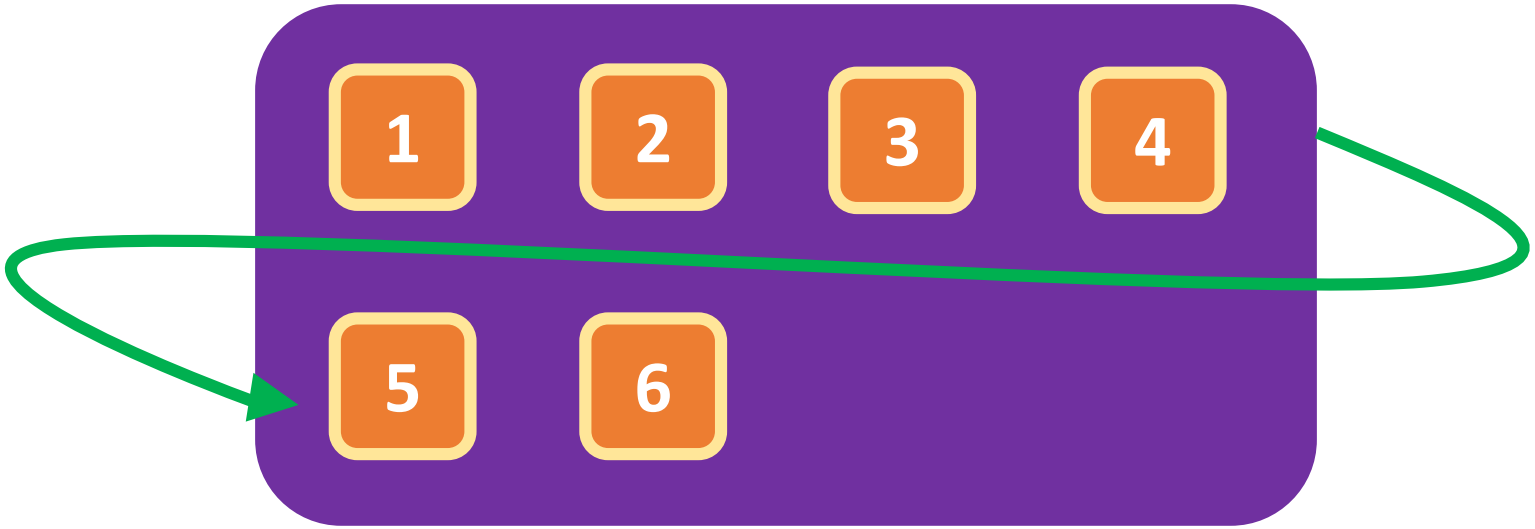
```
flex-wrap : nowrap | wrap | wrap-reverse ;
```

flex-wrap : nowrap ;



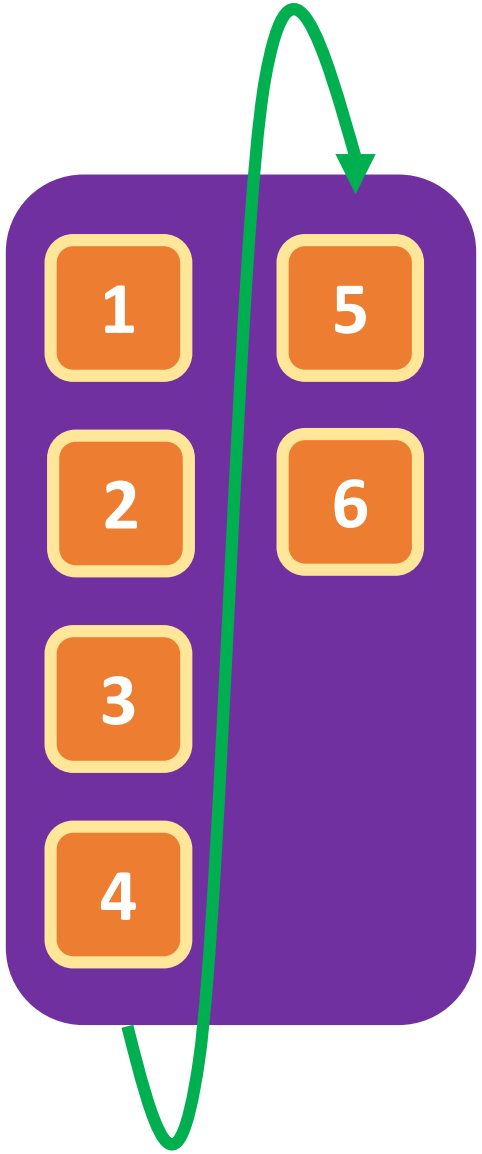
No-wrap may cause flex container to overflow , breaking the layout

flex-wrap : wrap ;



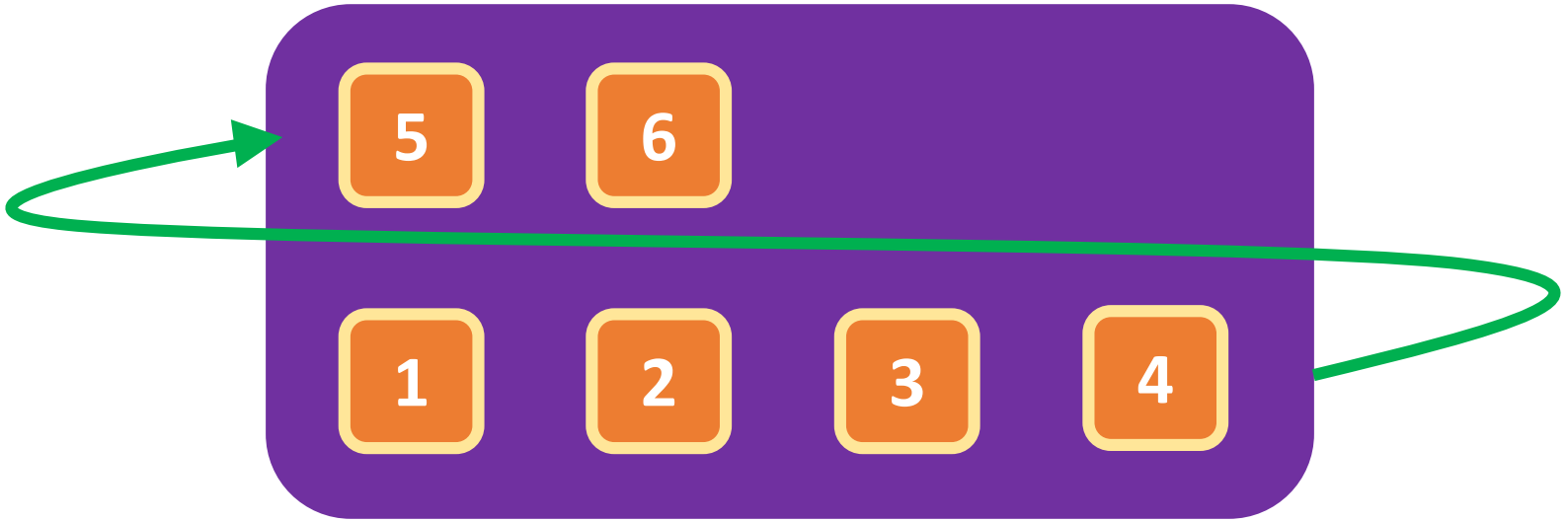
Flex items wrapped on the new next line / row along the cross axis

flex-wrap : wrap ;



Flex items wrapped on the new next line / column along the cross axis

flex-wrap : wrap-reverse ;



Flex items wrapped on the new next line / row but in the reverse / opposite direction of the the cross axis

Section 2 :

Working On Flex Container

Lecture 6.10 :

The Shorthand ‘ flex-flow ‘ Property

The shorthand ‘ flex-flow ’ Property :

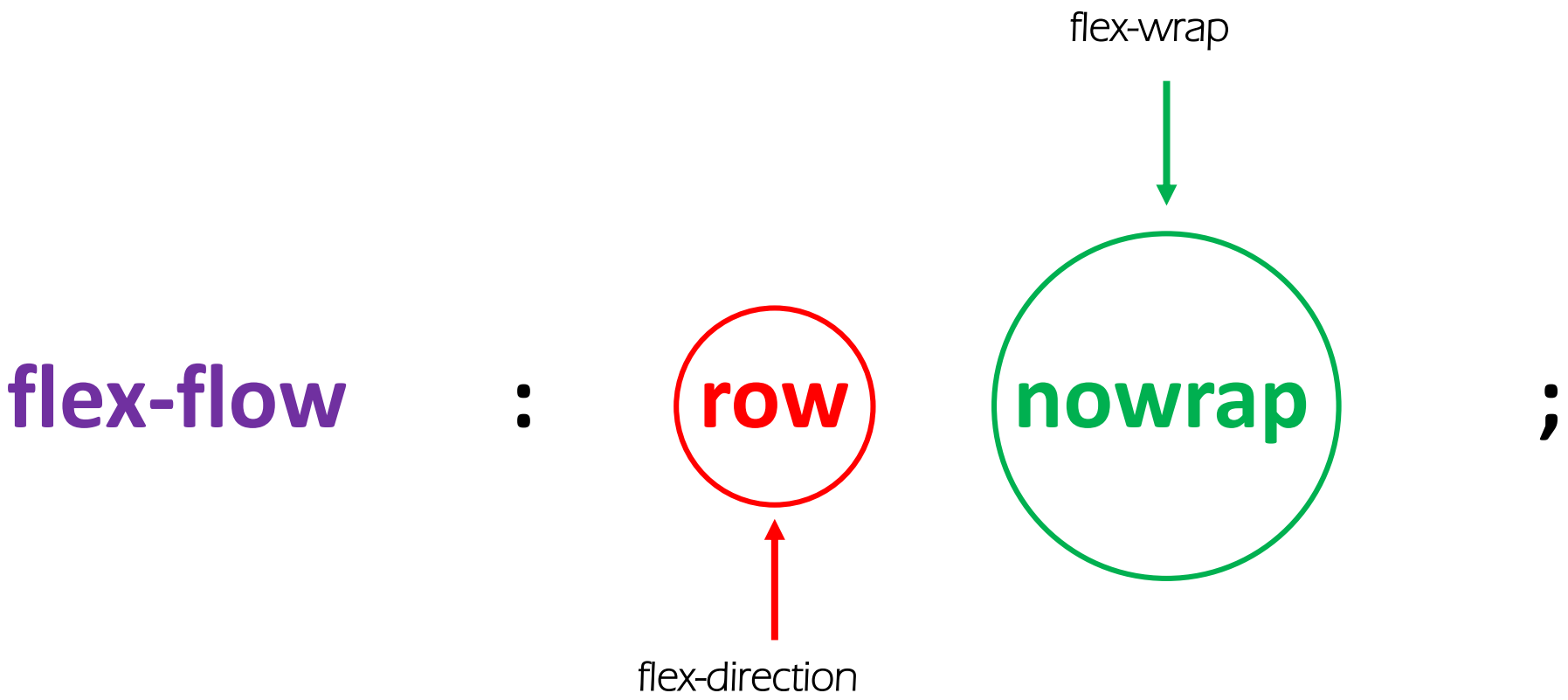
- The **flex-flow** property is a shorthand for flex-direction and flex-wrap , in allows us to specify both of them using just one property name :

  - **row nowrap – This is the default .**

- Syntax for **flex-flow** property :

flex-flow : row nowrap [flex-direction flex-wrap]

The shorthand ‘ flex-flow ’ Property :



Section 2 :

Working On Flex Container

Lecture 7.10 :

Justify-content Property

What Is *justify-content* ?

- The **justify-content** Property aligns the flex items along the **main-axis** , the direction in which flex-direction has set the flow .

- Typically it helps to distribute extra free space leftover between and around the flex items along the main-axis of a flex container .

justify-content :

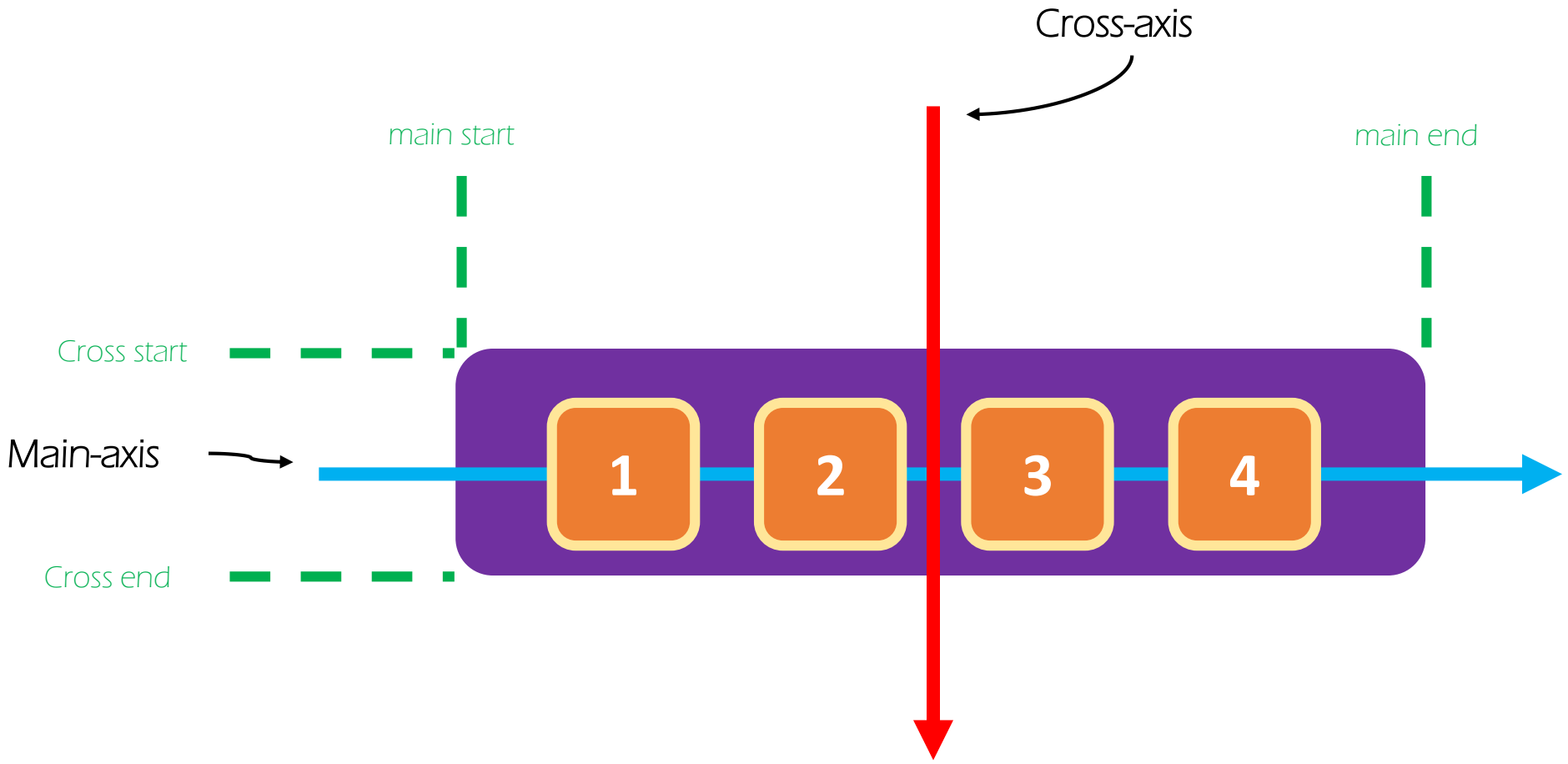
- The **justify-content** property is used to align the flex items along the main-axis . It accepts 6 values :

  - **flex-start** - Aligns the flex items at the start of the flex container . This is default .
  - **flex-end** - Aligns the Flex items at the end of the flex container .
  - **center** - Aligns the flex items in the middle of the flex container .
  - **space-between** - Aligns the flex items with equal spacing between them .
  - **space-around** - Aligns the flex items with equal spacing around them .
  - **space-evenly** - Aligns the flex items with equal spacing between any two flex items.

- Syntax for **justify-content** property :

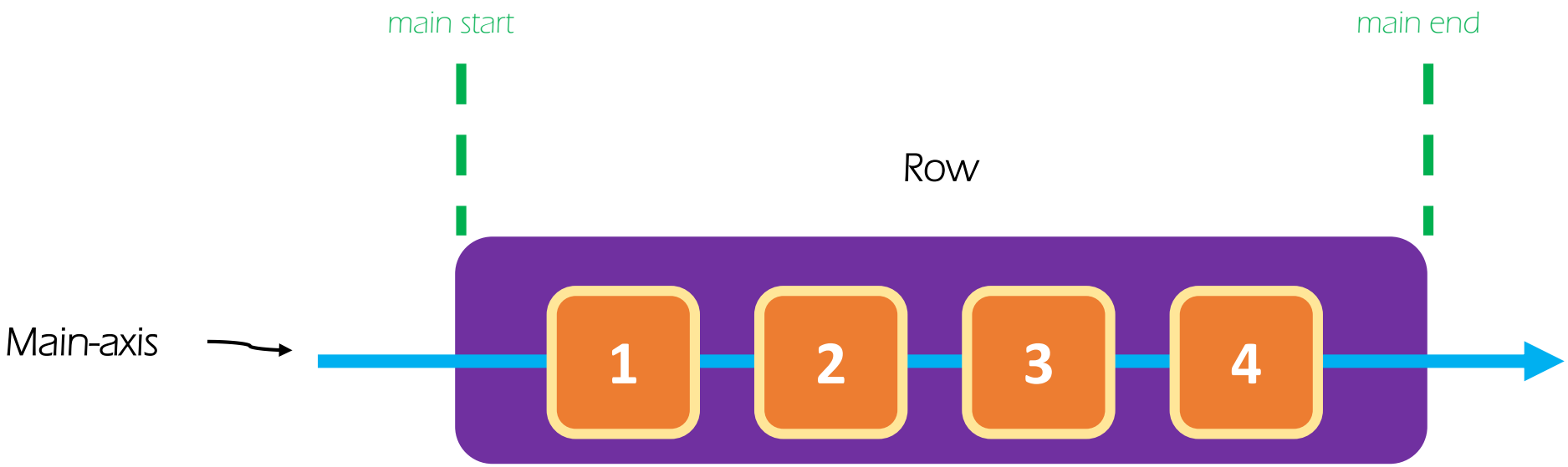
```
justify-content : flex-start | flex-end | center | space-between | … ;
```

Main-axis & Cross-axis :



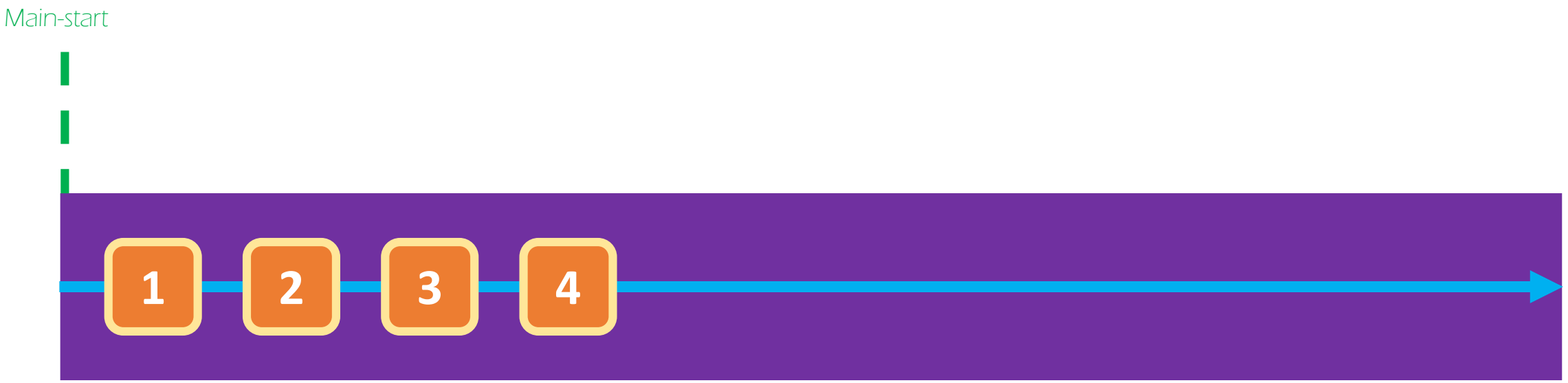
Flex items laid out horizontally .

Main-axis :



Note : justify-content positions flex-items along the Main-axis

justify-content : flex-start ;

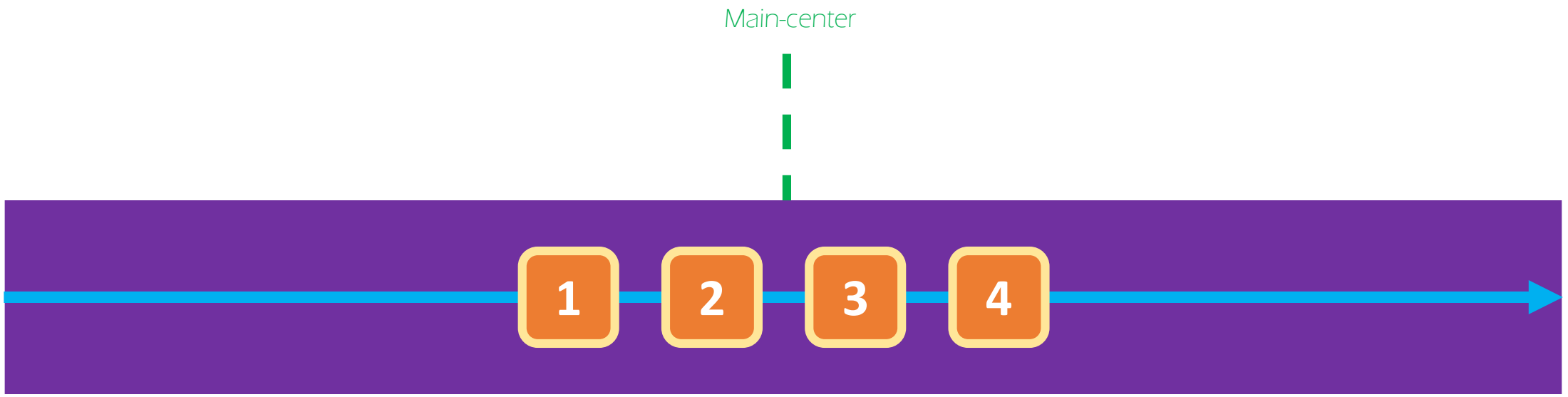


All flex items are placed at the start of main-axis . This is default

justify-content : flex-end ;

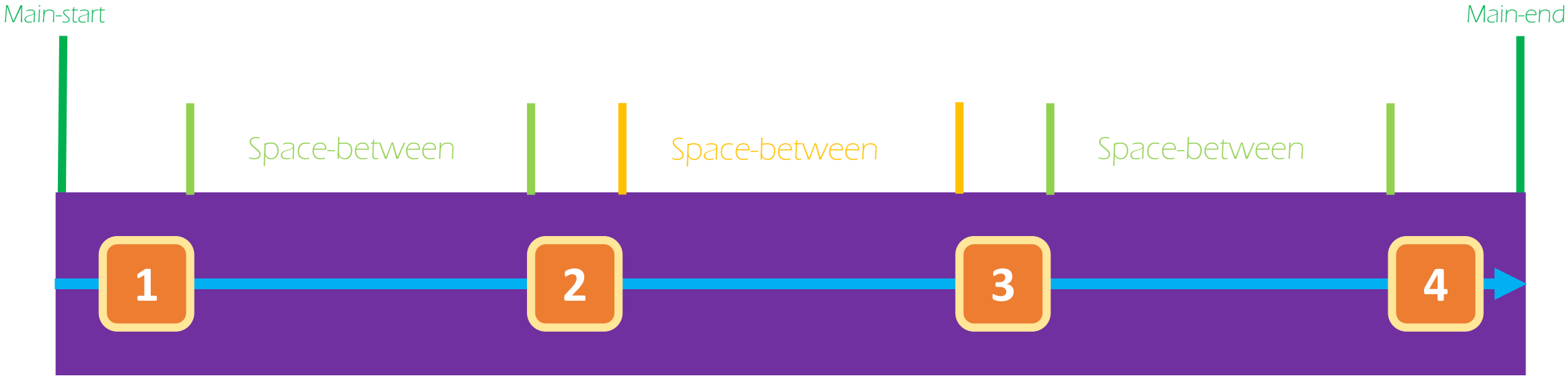
All flex items are placed at the end of main-axis

justify-content : center ;



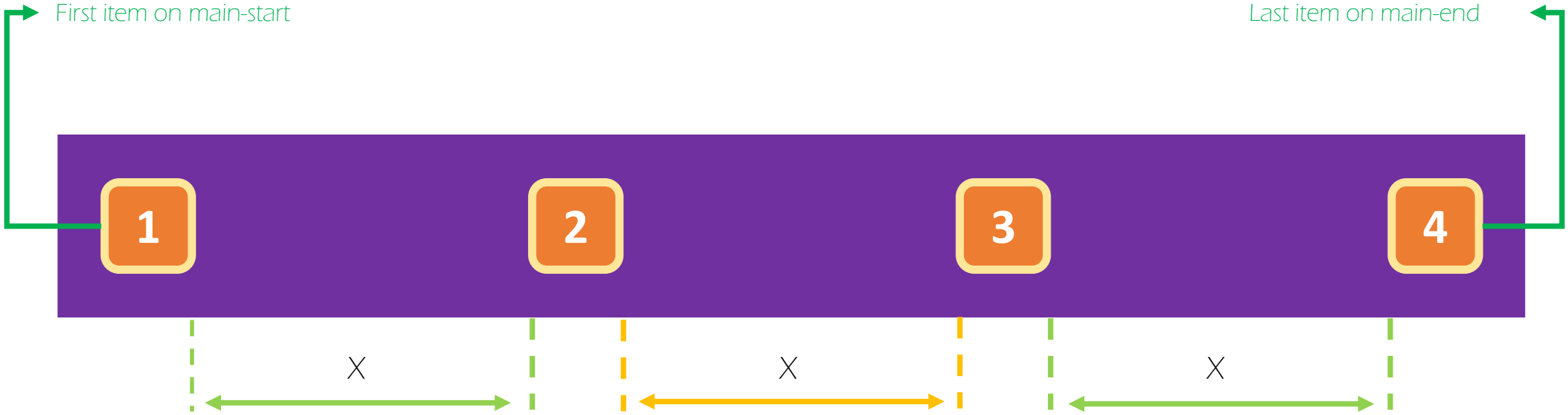
All flex items are placed at the center of main-axis

justify-content : space-between ;



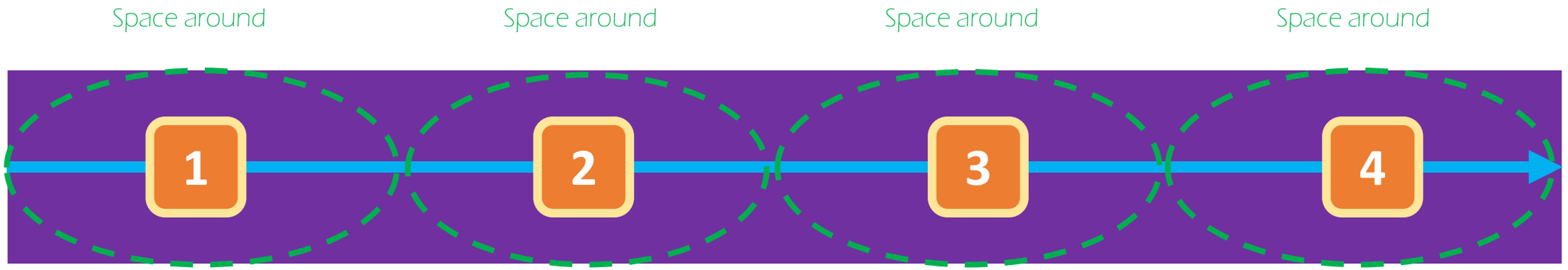
Flex items are evenly distributed along the main-axis , first item is on the main-start and last item is on the main-end

justify-content : space-between ;



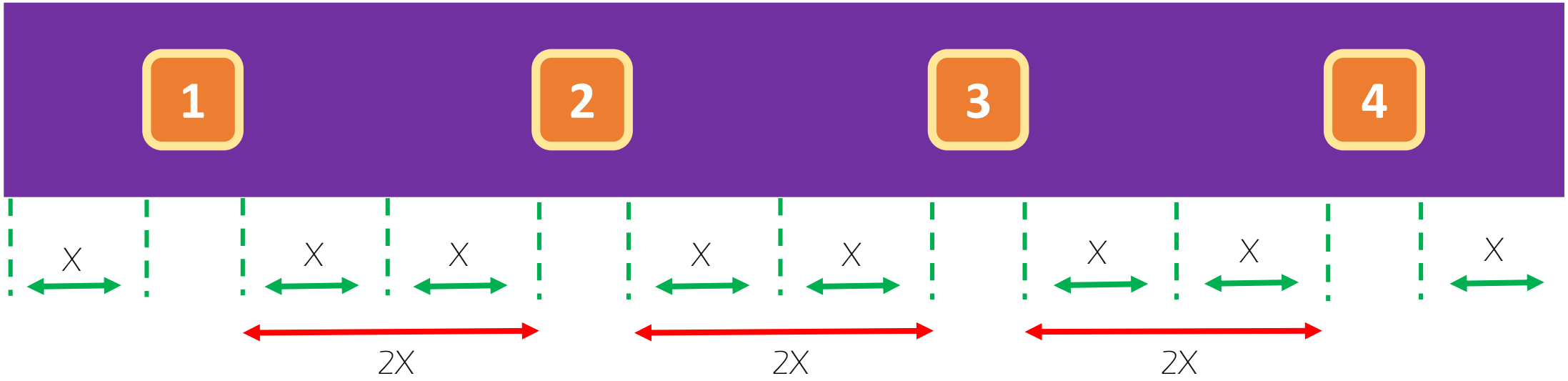
Same spacing between flex-items

justify-content : space-around ;



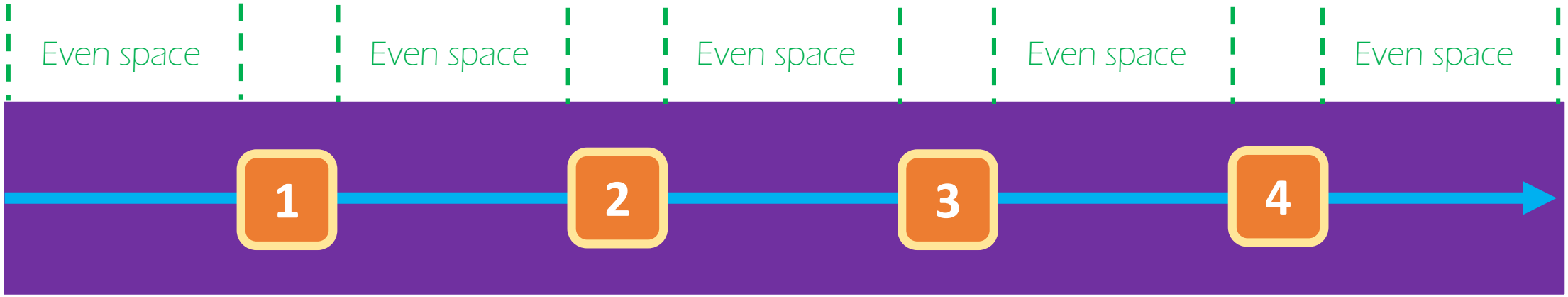
Flex items are evenly distributed along the main-axis with equal space around them

justify-content : space-around ;



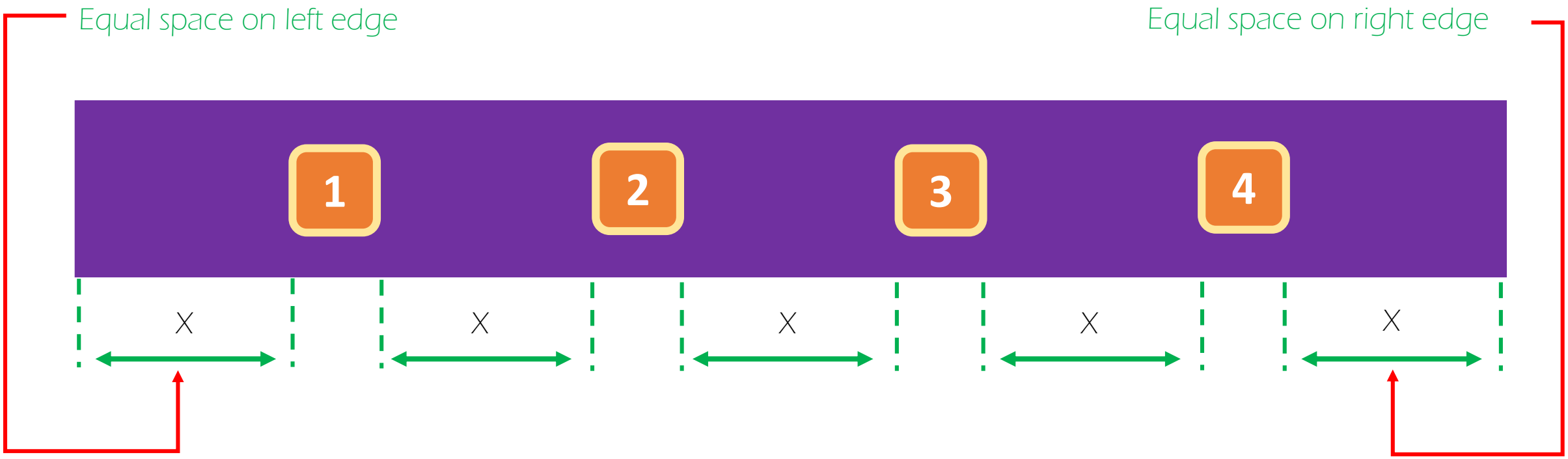
Same spacing around flex-items

justify-content : space-evenly ;



Flex items are distributed so that the spacing between any two flex items is equal .
And also the space to the edges is equal too

justify-content : space-evenly ;



Same spacing between flex-items , And also the space to the edges is equal too

justify-content – row :

flex-start

center

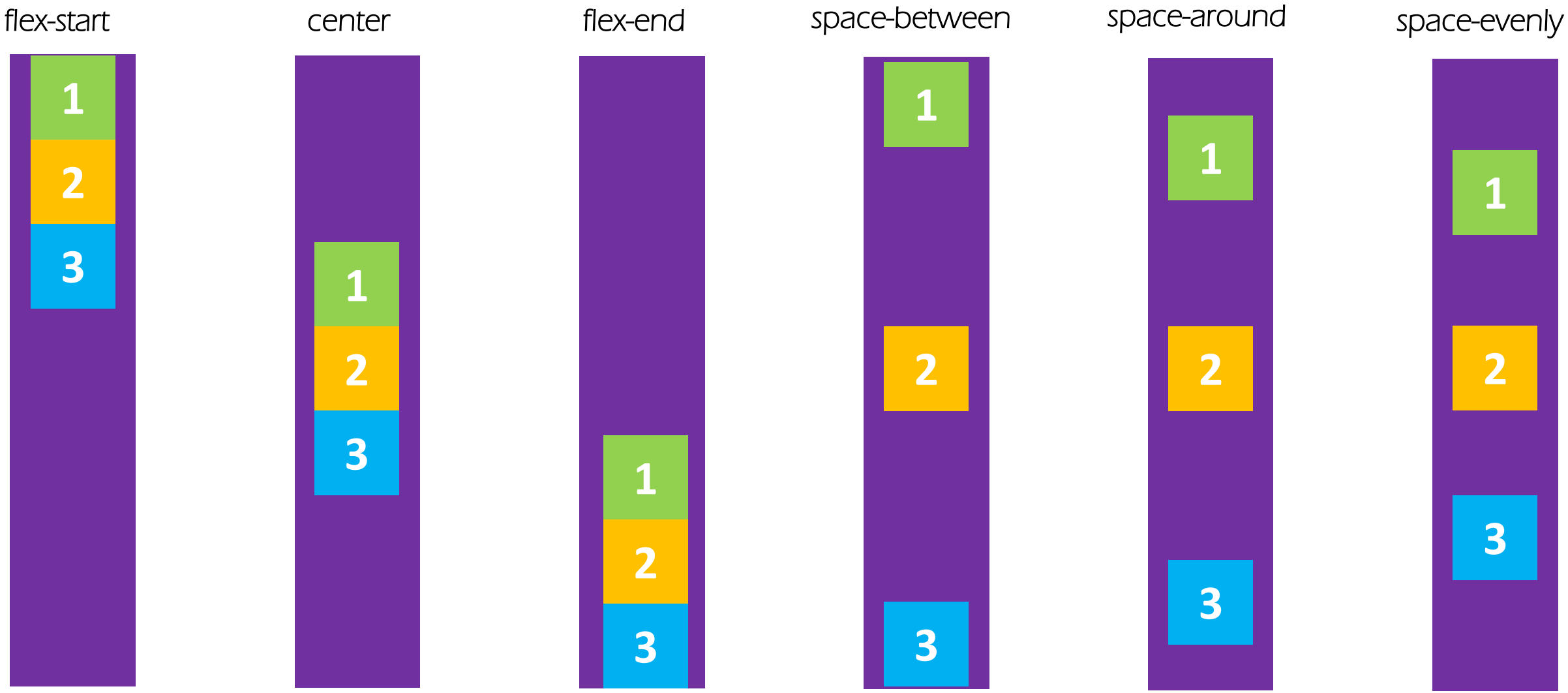
flex-end

space-between

space-around

space-evenly

justify-content – column :



Section 2 :

Working On Flex Container

Lecture 8.10 :

align-items Property

What Is *align-items* ?

- The align-items allows to align flex items along the **cross-axis** on the current **row/column** .

align-items Properties :

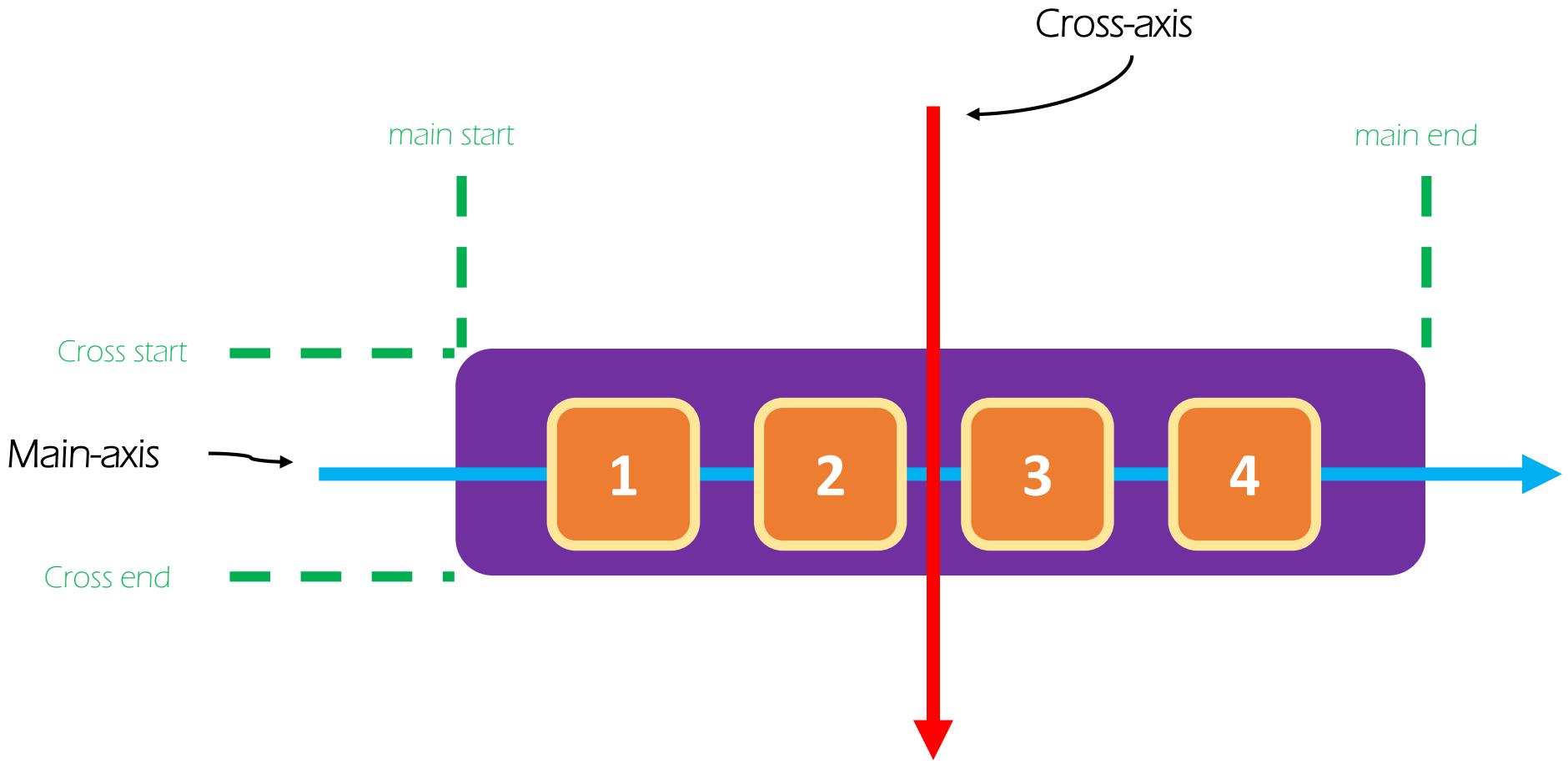
- The **align-items** property is used to align the flex items along the **cross-axis** . It accepts 5 values :

  - **stretch** - Stretches the flex items to fill the flex container . This is default .
  - **flex-start** - Aligns the flex items at the top of the flex container .
  - **center** - Aligns the flex items in the middle of the flex container .
  - **flex-end** - Aligns the flex items at the bottom of the flex container .
  - **baseline** - Aligns the flex items along their baselines .

- Syntax for **align-items** property :

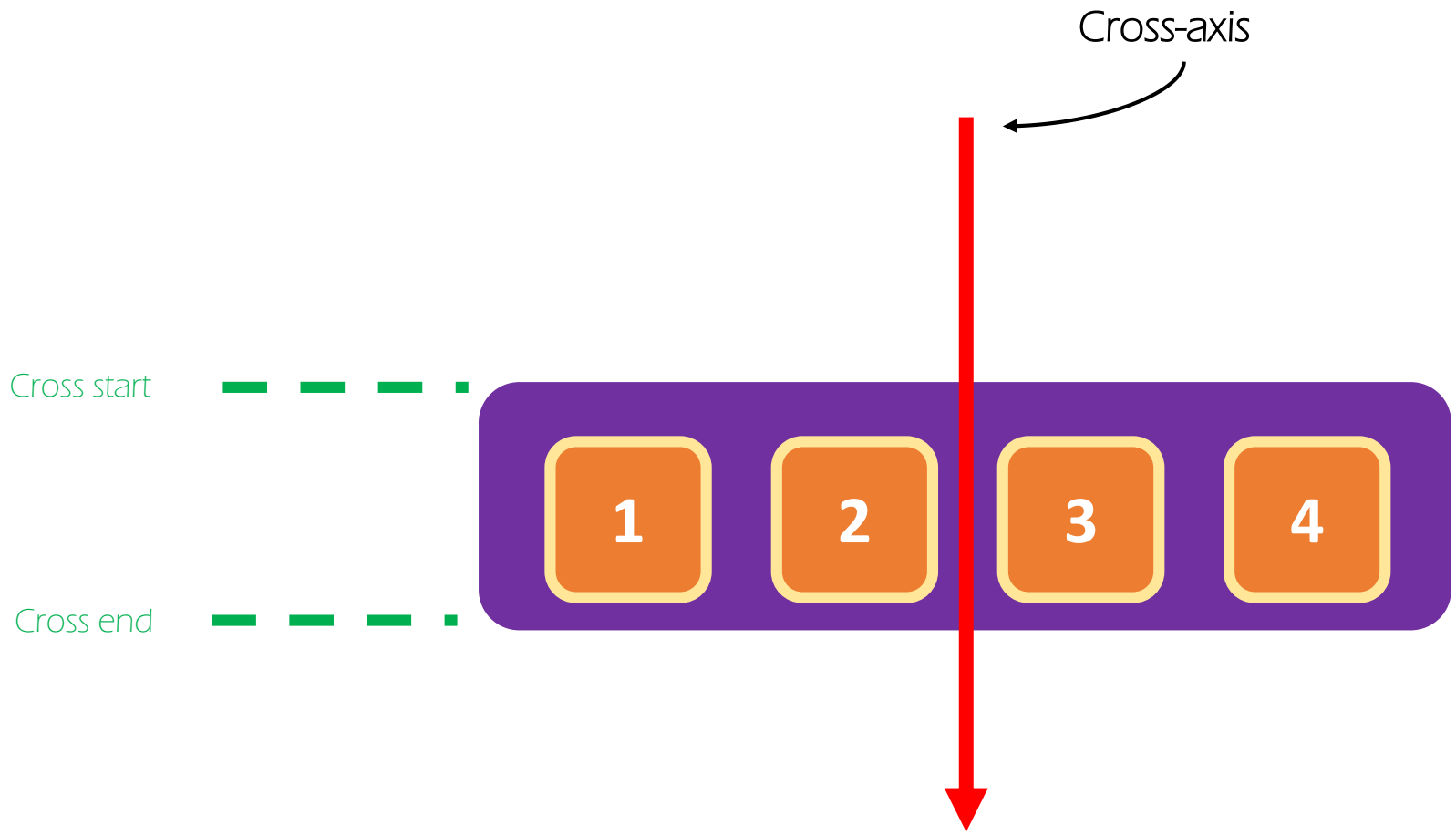
```
align-items : stretch | flex-start | center | flex-end | baseline ;
```

Main-axis & Cross-axis :



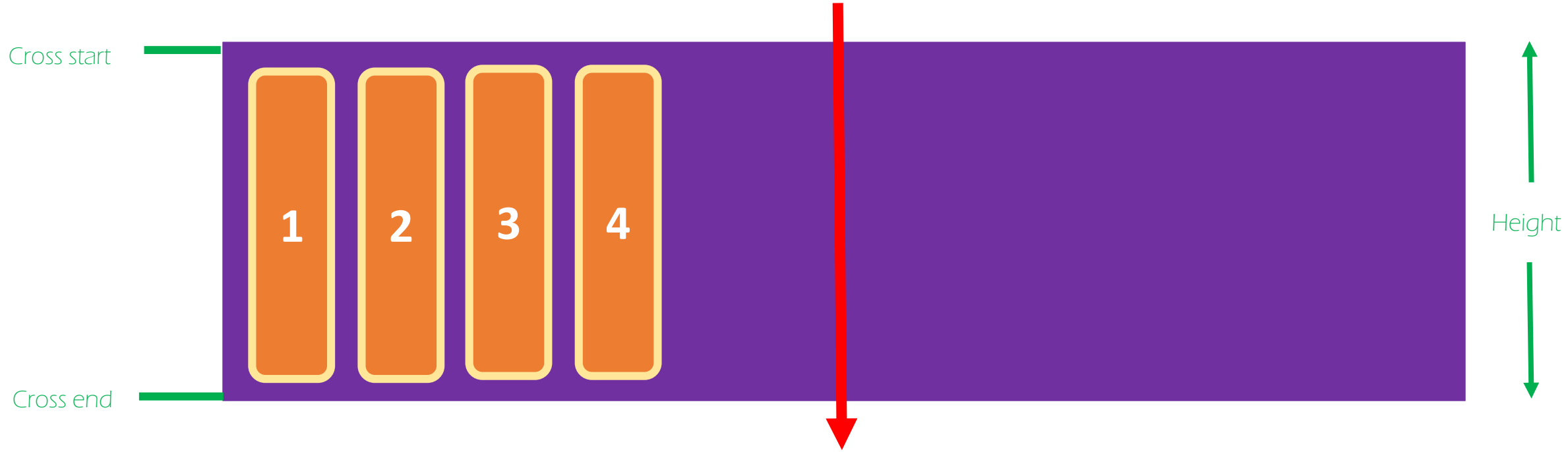
Flex items laid out horizontally .

Cross-axis :



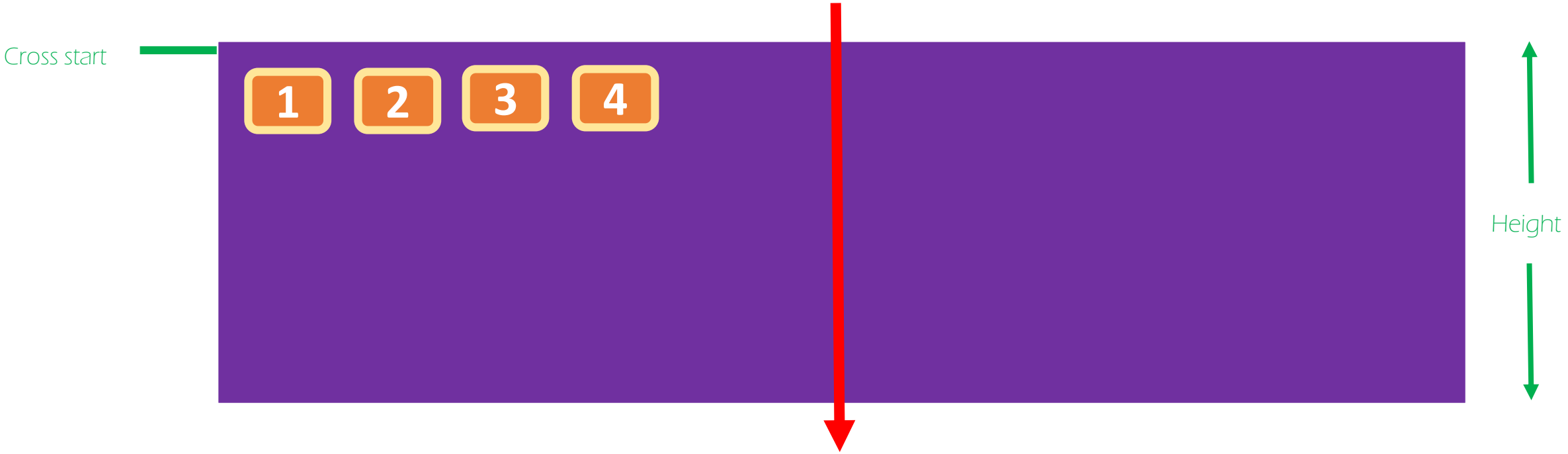
Note : align-items positions flex-items along the Cross-axi

align-items : stretch ;



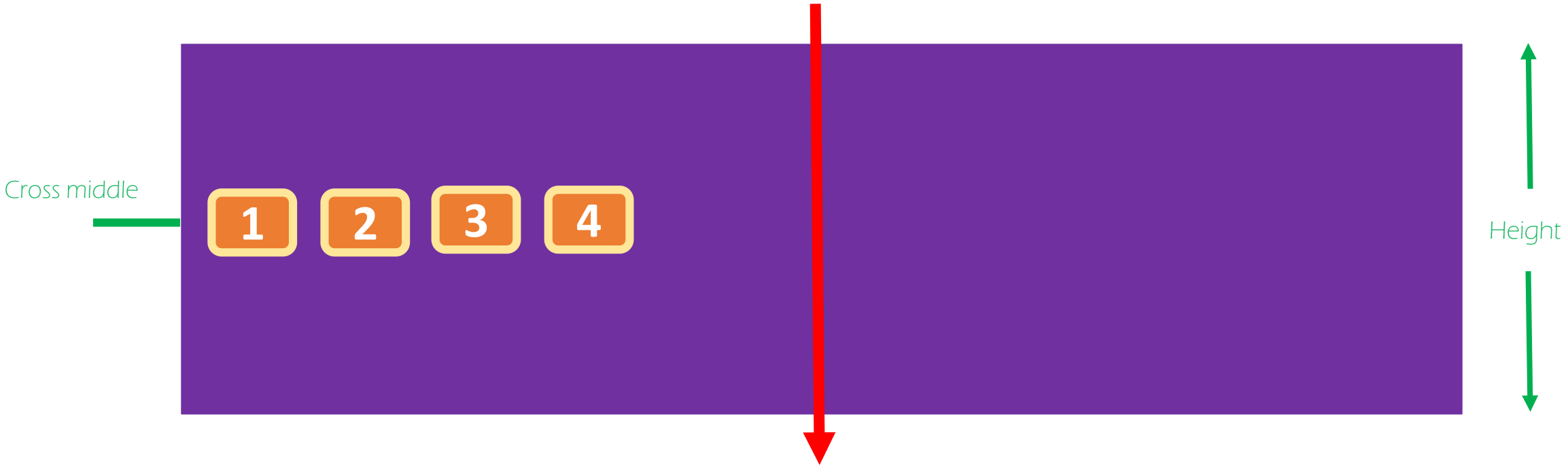
All flex items stretch along the cross-axis to fill the flex container . This is default .

align-items : flex-start ;



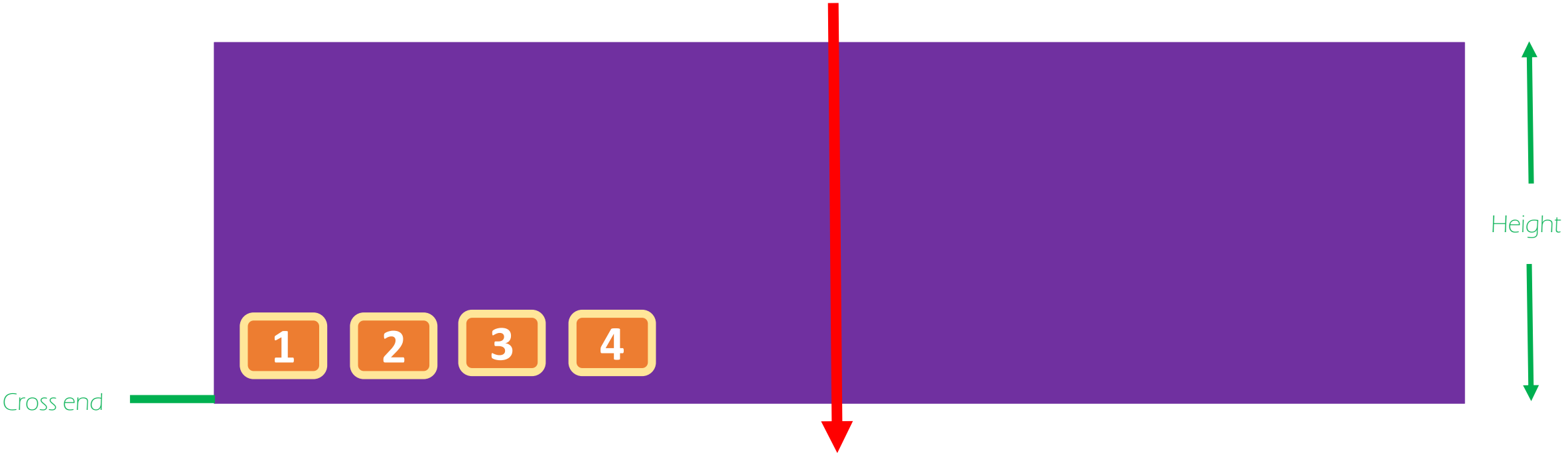
All flex items are placed at the start of the cross-axis .

align-items : center ;



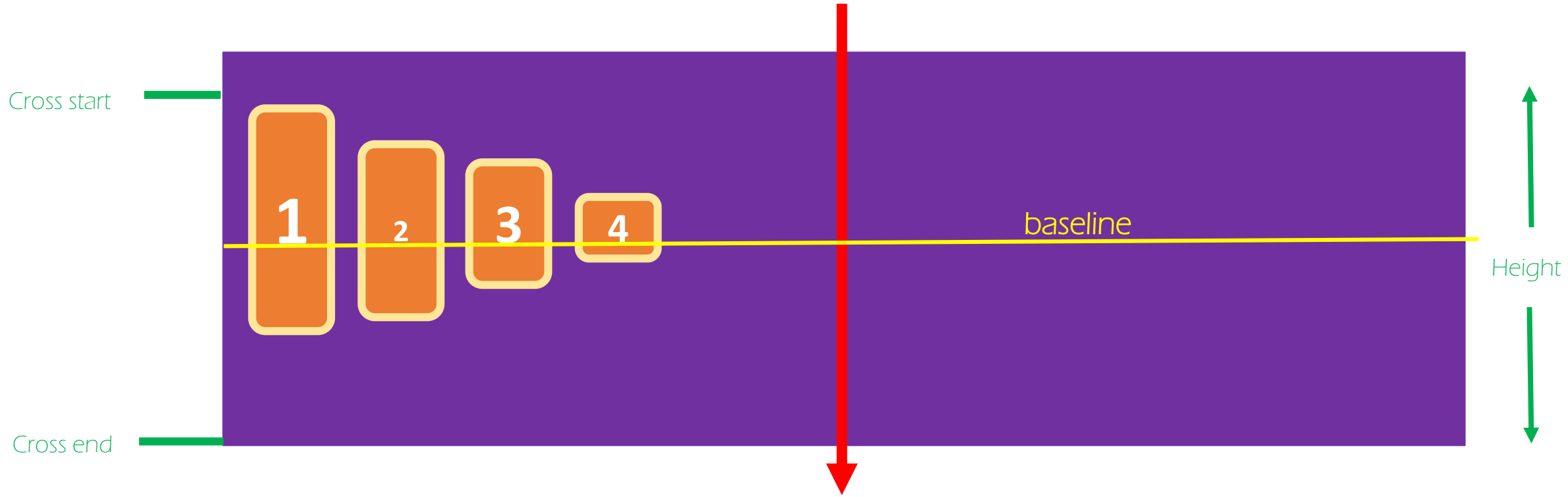
All flex items are placed in the middle of the cross-axis .

align-items : flex-end ;



All flex items are placed at the end of cross-axis .

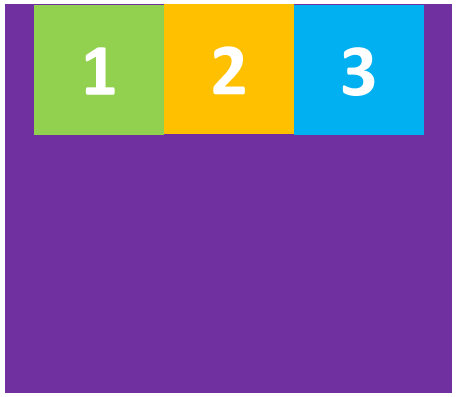
align-items : baseline ;



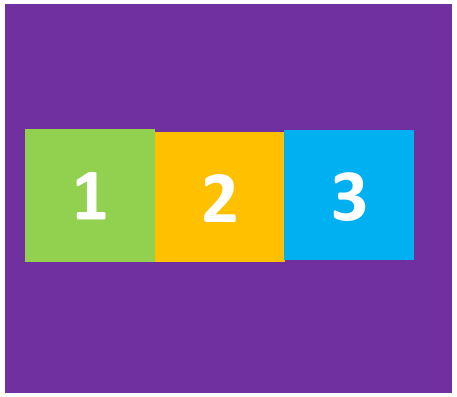
All flex items are aligned in such a way that the baseline are equal .

align-items – row :→

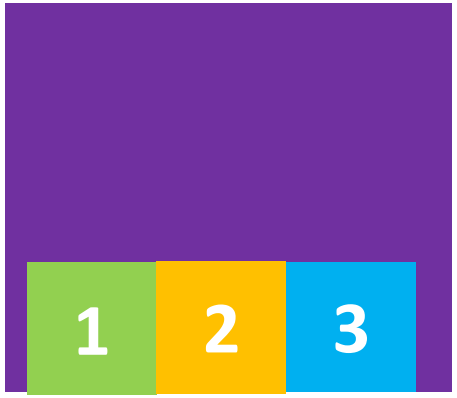
flex-start



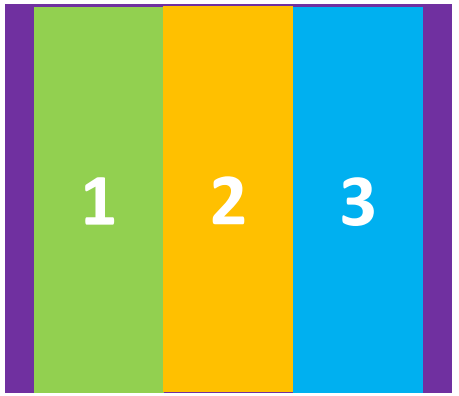
center



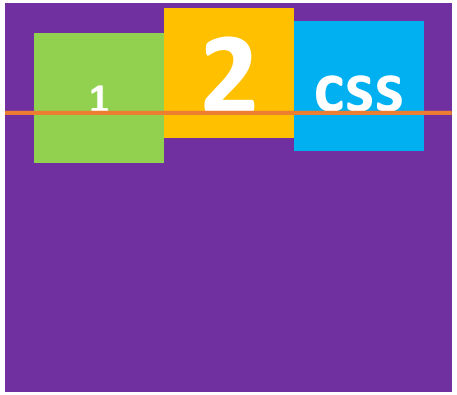
flex-end



strech

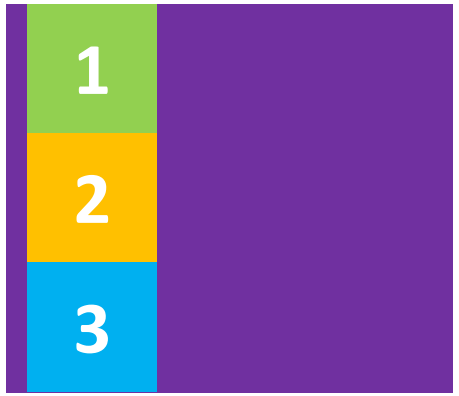


baseline

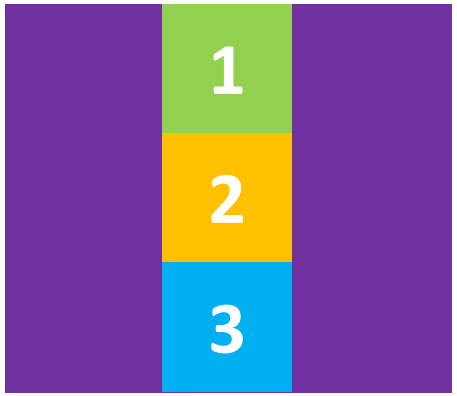


align-items – column :

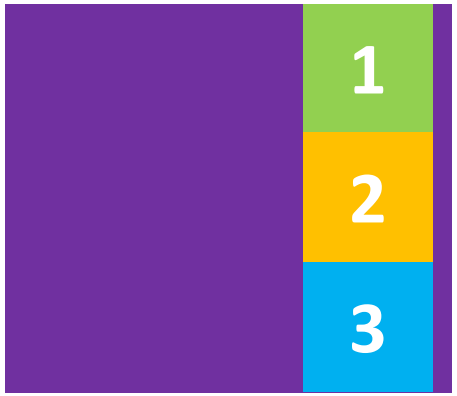
flex-start



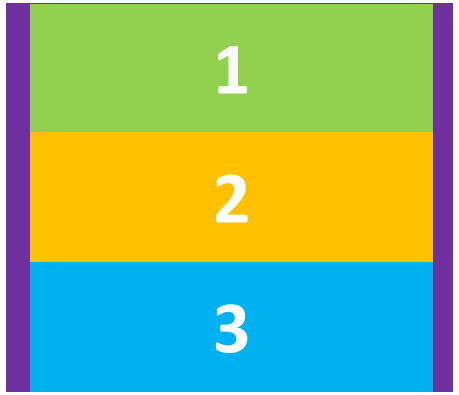
center



flex-end



strech



Section 2 :

Working On Flex Container

Lecture 9.10 :

align-content Property

What Is align-content ?

- The align-content aligns flex lines within the flex container when there is an extra space along the **cross-axis** .
- **It has no effect on single line flex containers** .

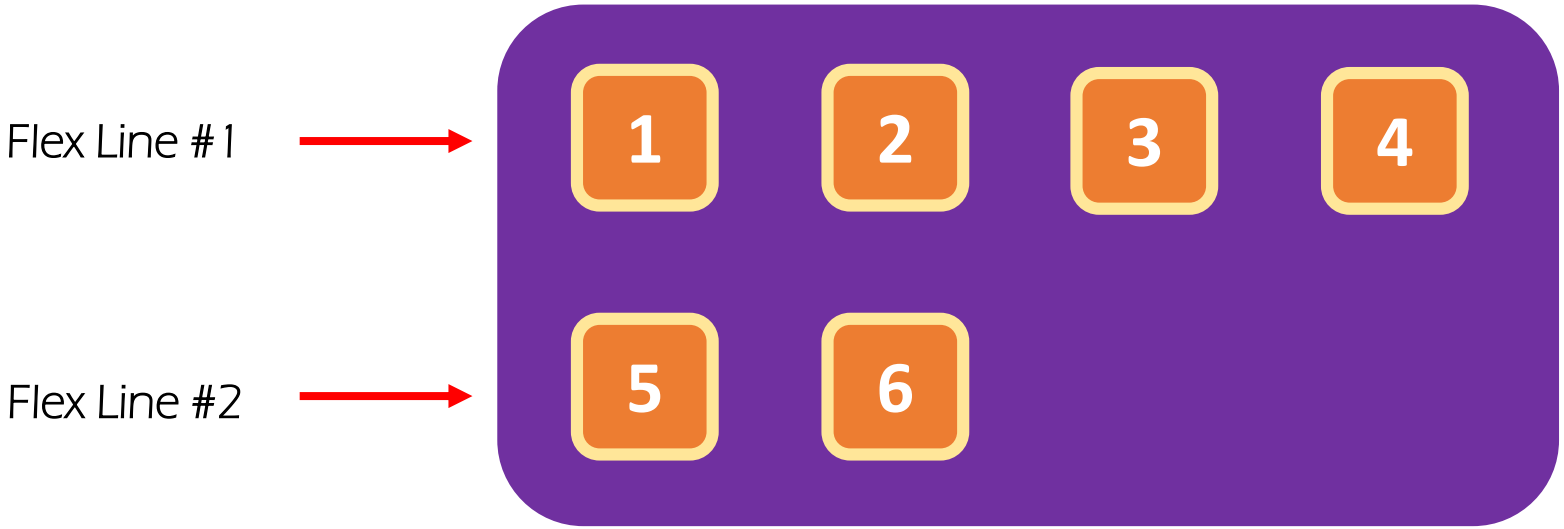
Flex lines :

- Flex items in a flex container are laid out and aligned within flex lines . A flex container can be either single-line or multi-line depending on the flex-wrap property .

  - A **single-line** flex container lays out all of its flex items in a single line , even if that would cause its contents to overflow .

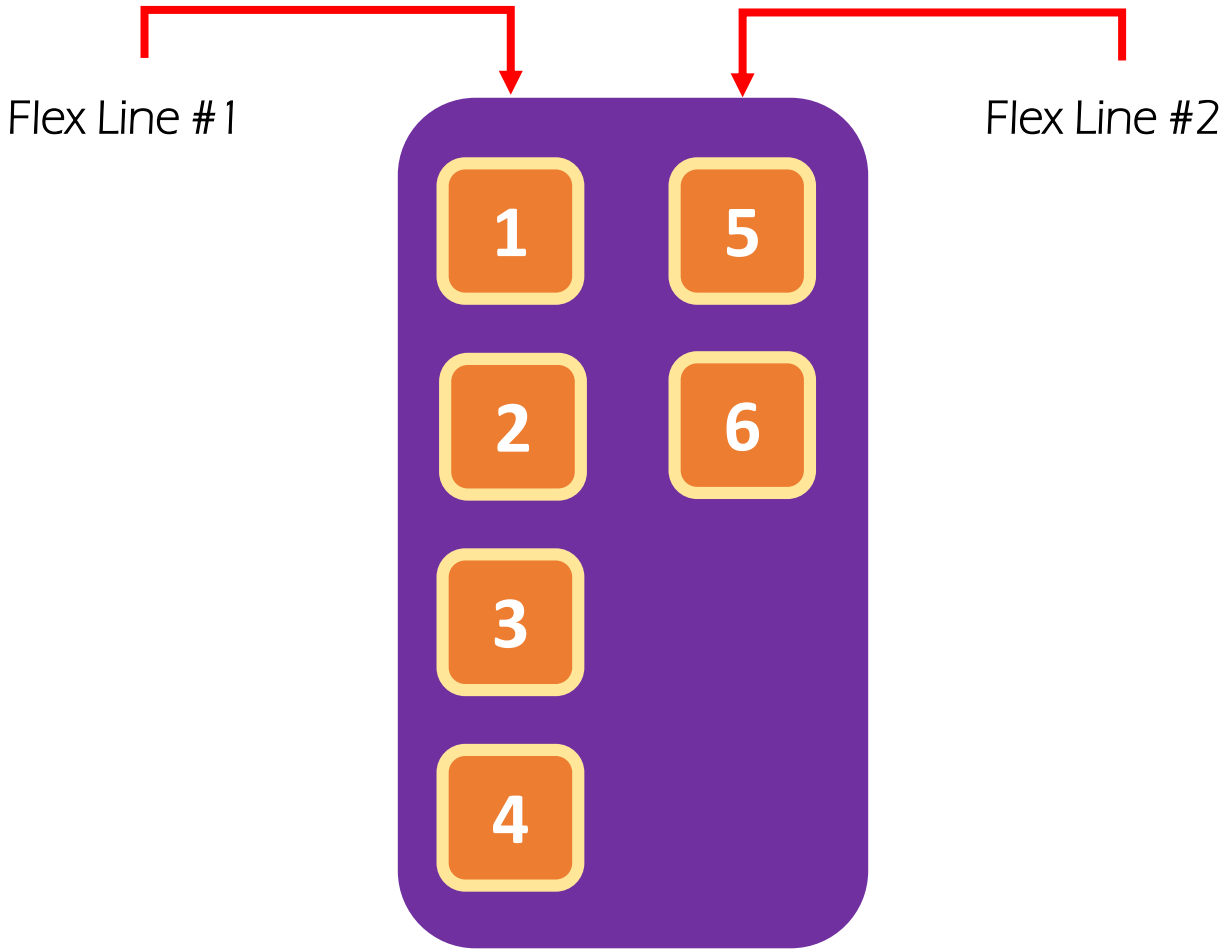
  - A **multi-line** flex container breaks its flex items across multiple lines . When additional lines are created , they are stacked in the flex container along the cross-axis according to the flex-wrap property .

flex-wrap : wrap ;



Flex items wrapped on the new next line / row along the cross-axis

flex-wrap : wrap ;



Flex items wrapped on the new next line / column along the cross-axis

align-content :

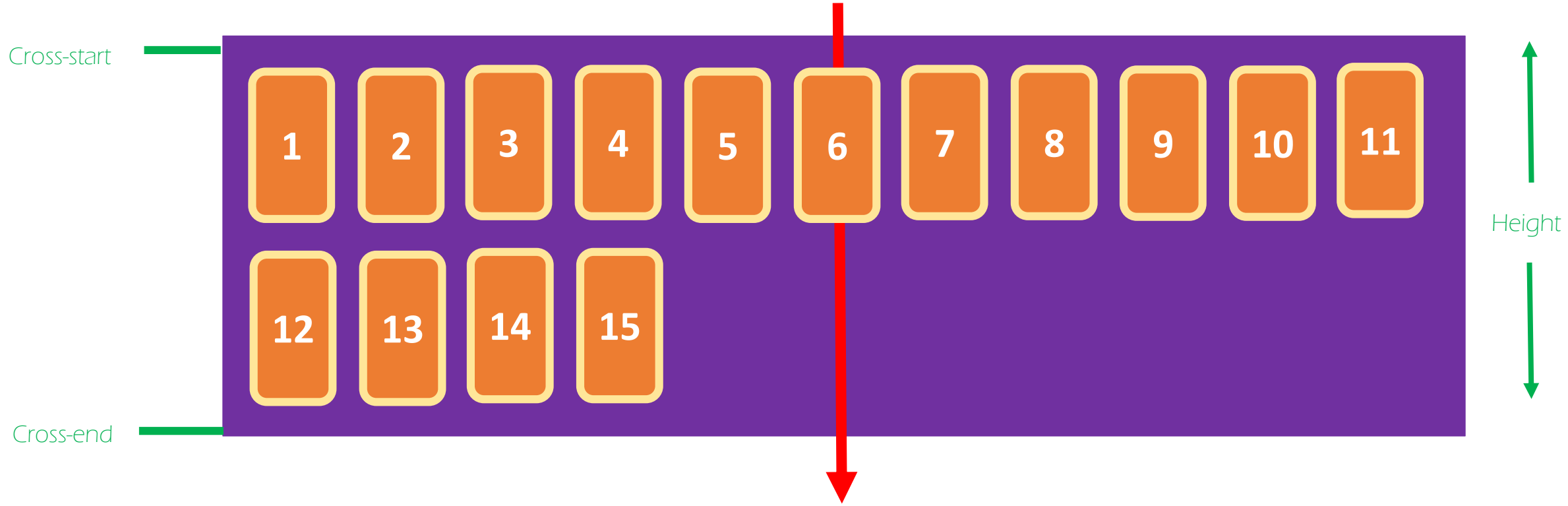
- The **align-content** property is used to align the flex lines along the **cross-axis** . It accepts 6 values :

  - **stretch** - Stretches the flex lines to take up the remaining space . This is default.
  - **flex-start** - Displays the flex lines at the start of the flex container .
  - **center** - Displays the flex lines in the middle of the flex container .
  - **flex-end** - Displays the flex lines at the end of the container .
  - **Space-between** - Displays the flex lines with equal space between them .
  - **Space-around -** Displays the flex lines with equal space around each line .

- Syntax for ***align-content*** property :

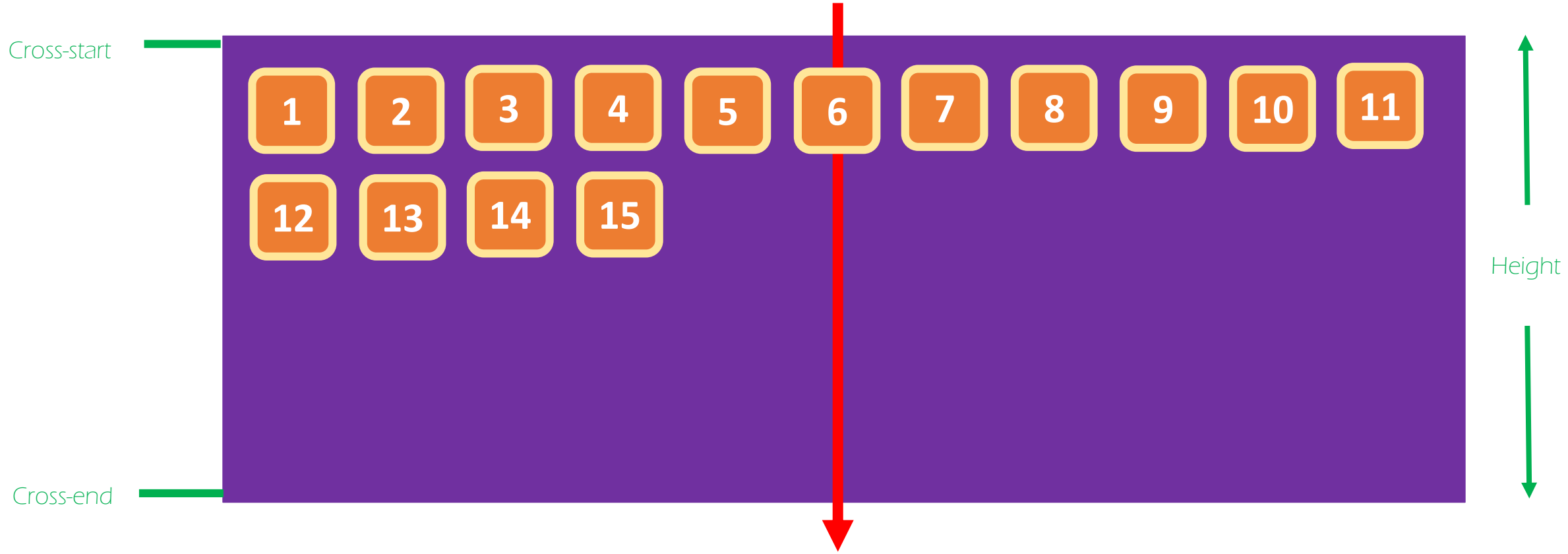
```
align-content : stretch | flex-start | center | flex-end | space-between | space-around ;
```

align-content : stretch ;



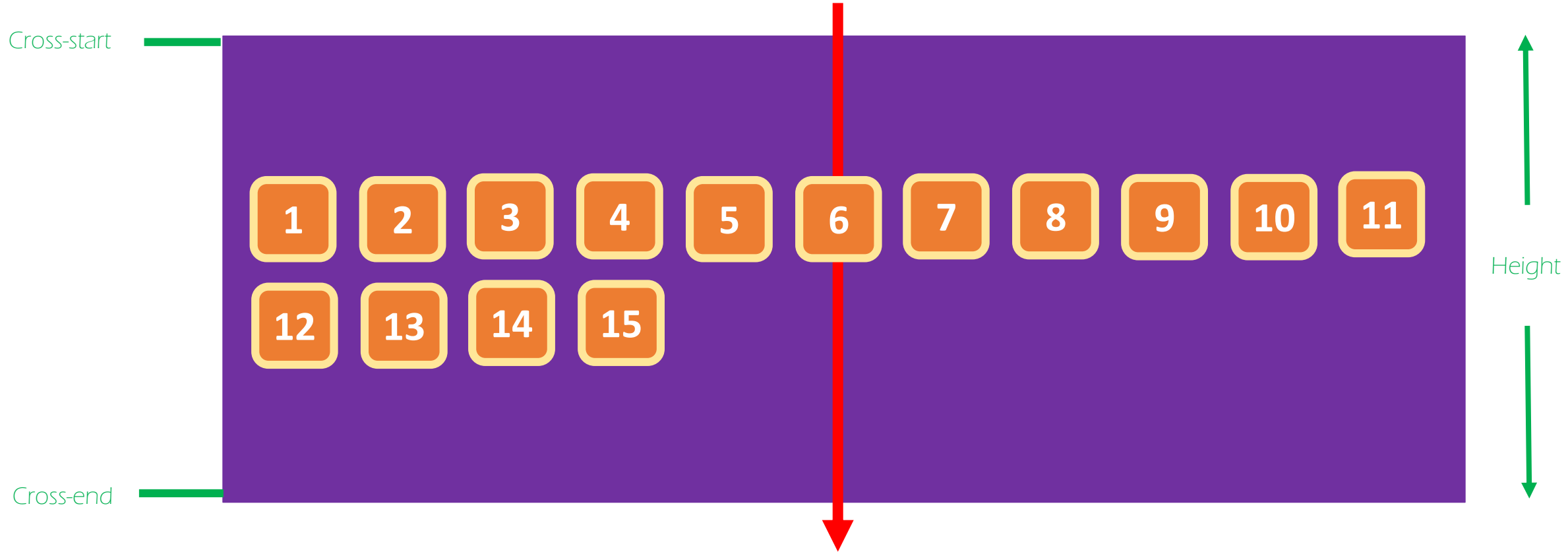
All flex lines stretch to take up the remaining space in flex container . This is default.

align-content : flex-start ;



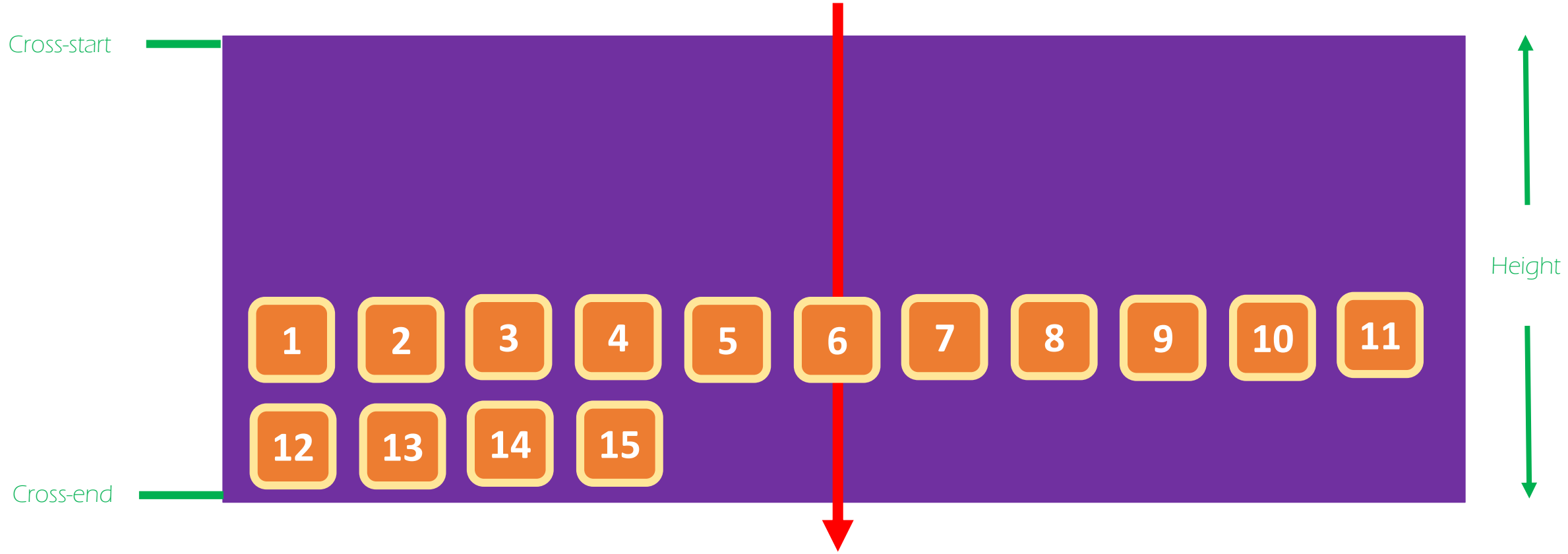
All flex lines are placed at the start of the flex container .

align-content : center ;



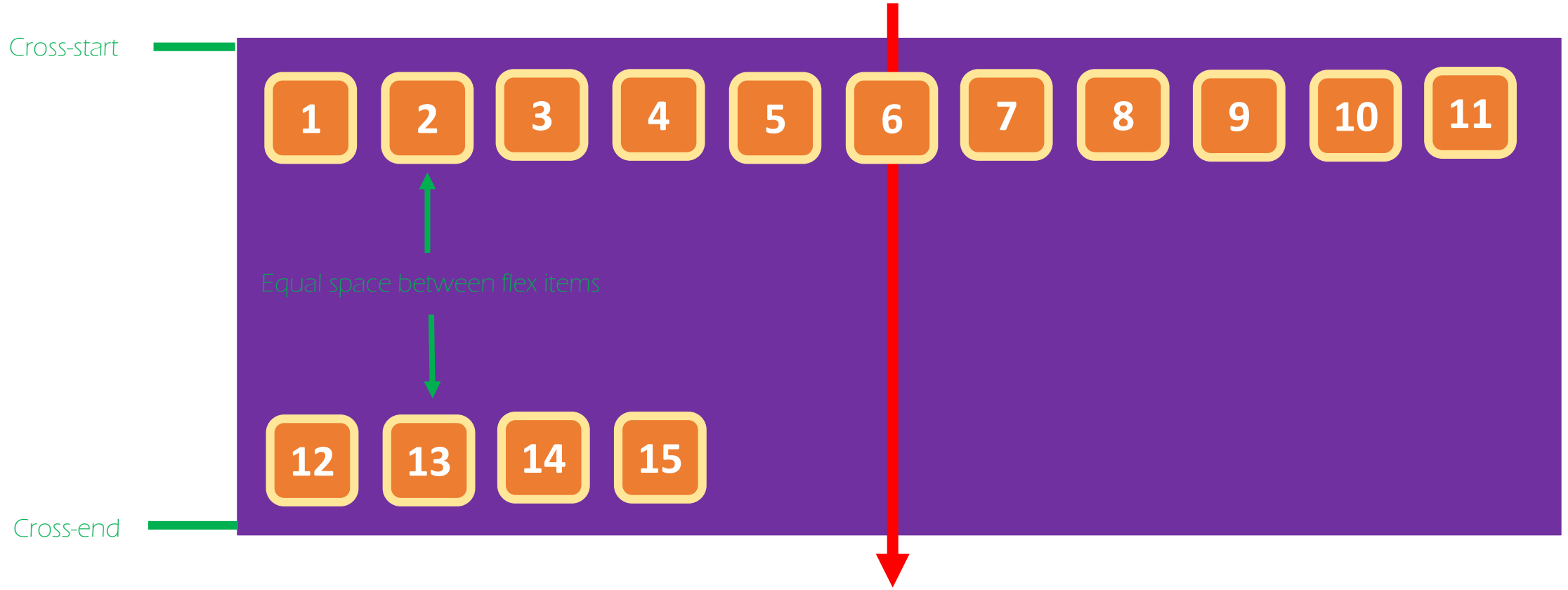
All flex lines are placed in the middle of the flex container .

align-content : flex-end ;



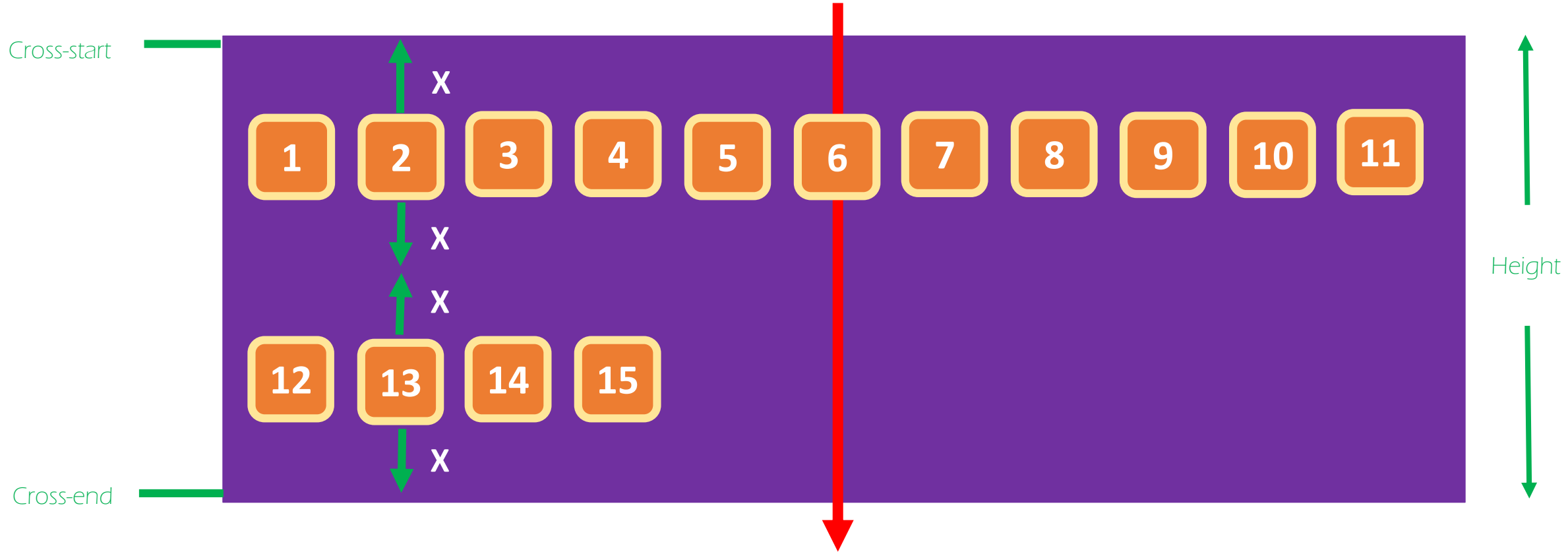
All flex lines are placed at the end of the flex container .

align-content : space-between ;



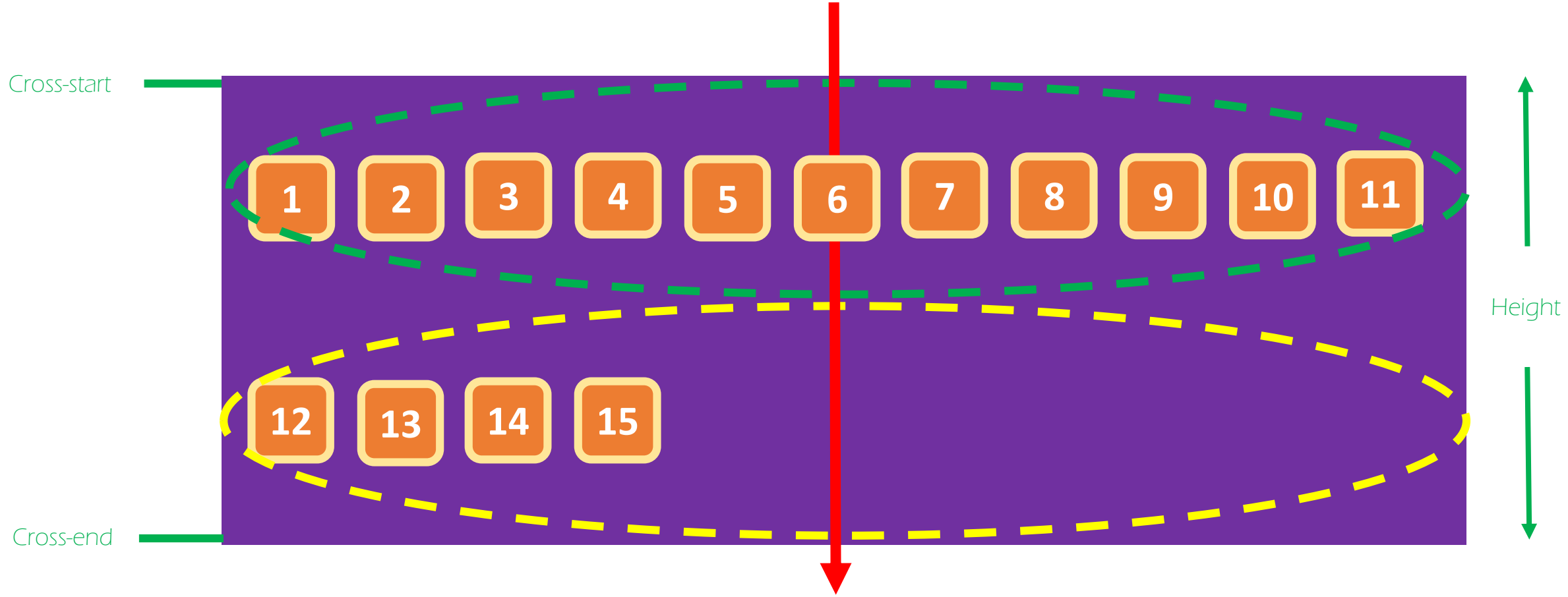
Flex lines are evenly distributed the first line is at the start of the flex container while the last one is at the end

align-content : space-around ;



Flex lines evenly distributed with equal space before and after each flex line .

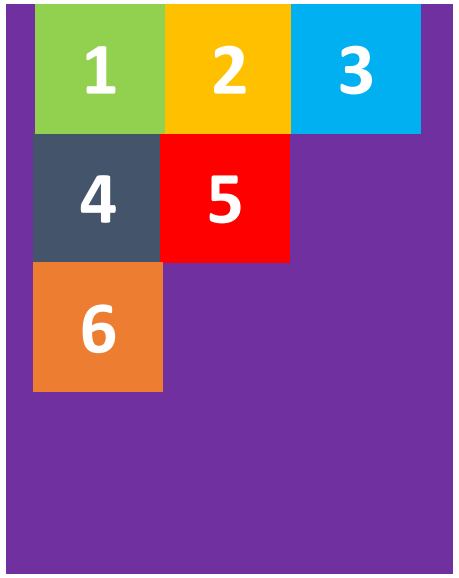
align-content : space-around ;



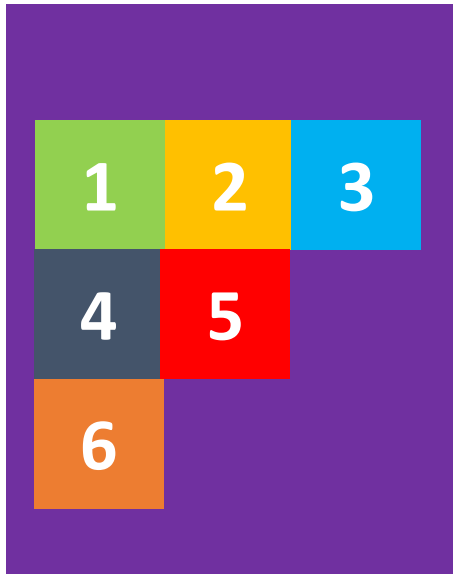
Flex lines evenly distributed with equal space around each line.

align-content :

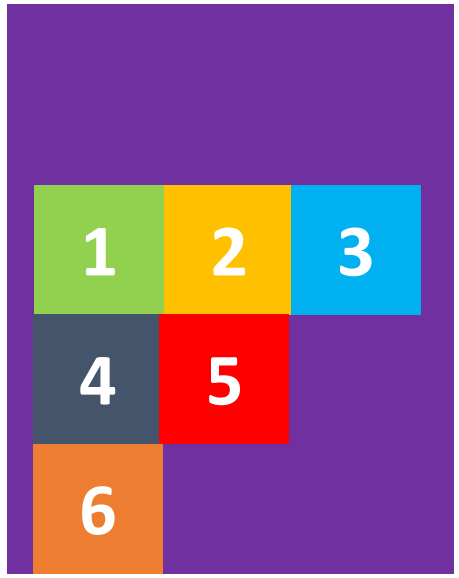
flex-start



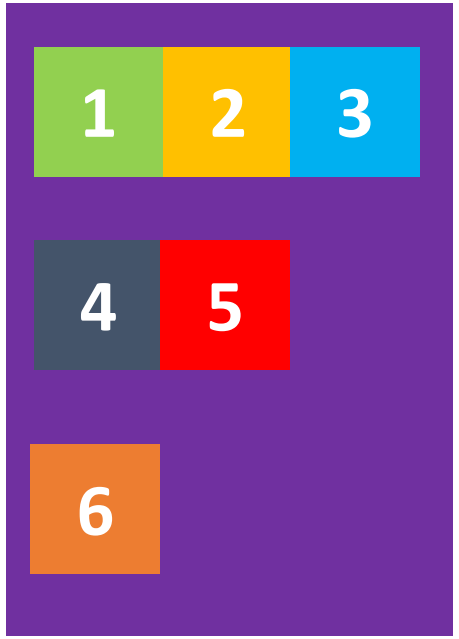
center



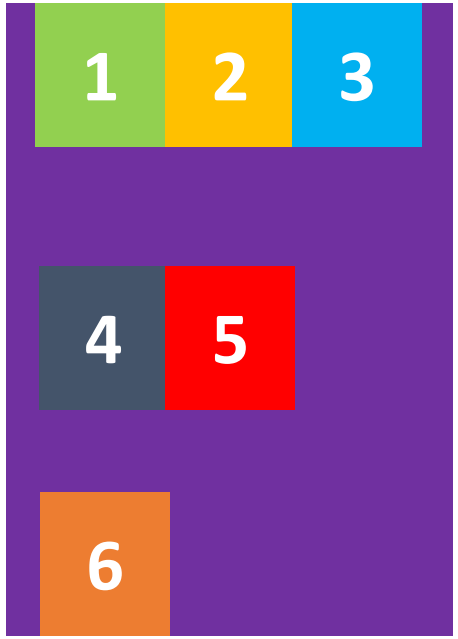
flex-end



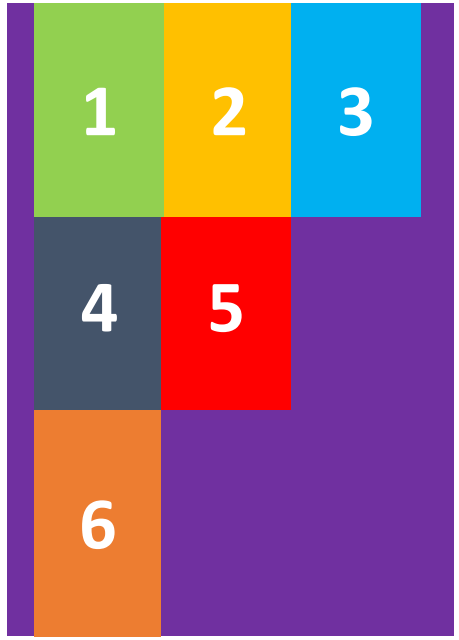
space-around



Space-between



stretch



Section 2 :

Working On Flex Container

Lecture 10.10 :

Flex Container Properties Overview

Flex Container Properties Overview :

1. **flex-direction : row** | row-reverse | column | column-reverse
2. **flex-wrap : nowrap** | wrap | wrap-reverse
3. **flex-flow : row nowrap** [**flex-direction flex-wrap**]
4. **justify-content : flex-start** | flex-end | center | space-between | space-around | space-evenly
5. **Align-items : stretch** | flex-start | flex-end | center | baseline
6. **align-content**

Section 3 :

Working On Flex Items

Lecture 1.8 :

Introduction

Working on Flex Items :

- order
- align-self
- flex-grow
- flex-shrink
- flex-basis
- The Shorthand ‘flex’ Property
- Flex Item properties overview

Section 3 :

Working On Flex Items

Lecture 2.8 :

order Property

What Is order ?

The flexbox ordering controls the order in which flex items of a flex container appear inside the flex container . By default they are ordered as initially added in the flex container .

order Property :

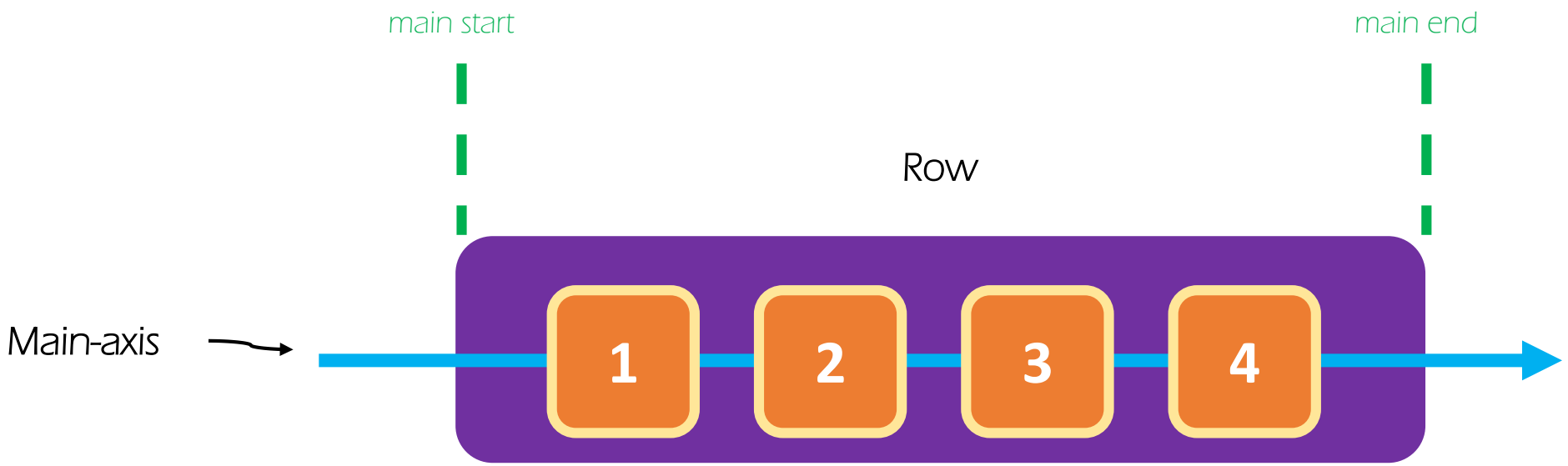
- The **order** property specifies the order of a flex item relative to the rest of flexible items inside the same flex container . It accepts positive or negative integer as a value .

  - **0** - This is default .
  - **<integer>** - A positive or negative integer value .

- Syntax for **order** property :

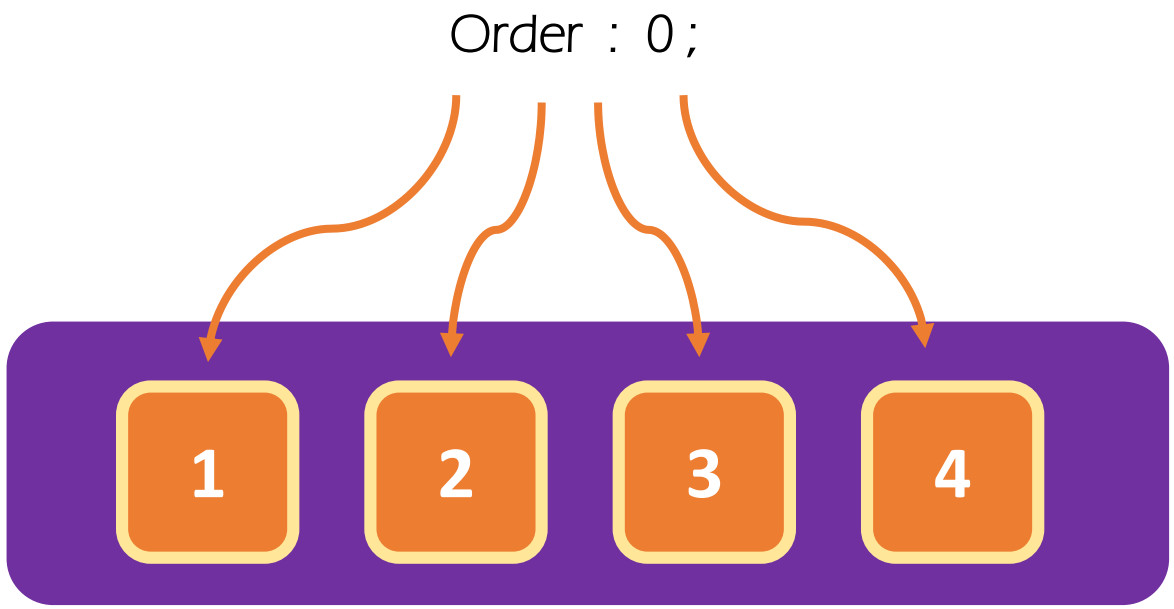
```
order : 0 | <integer> ;
```

order :



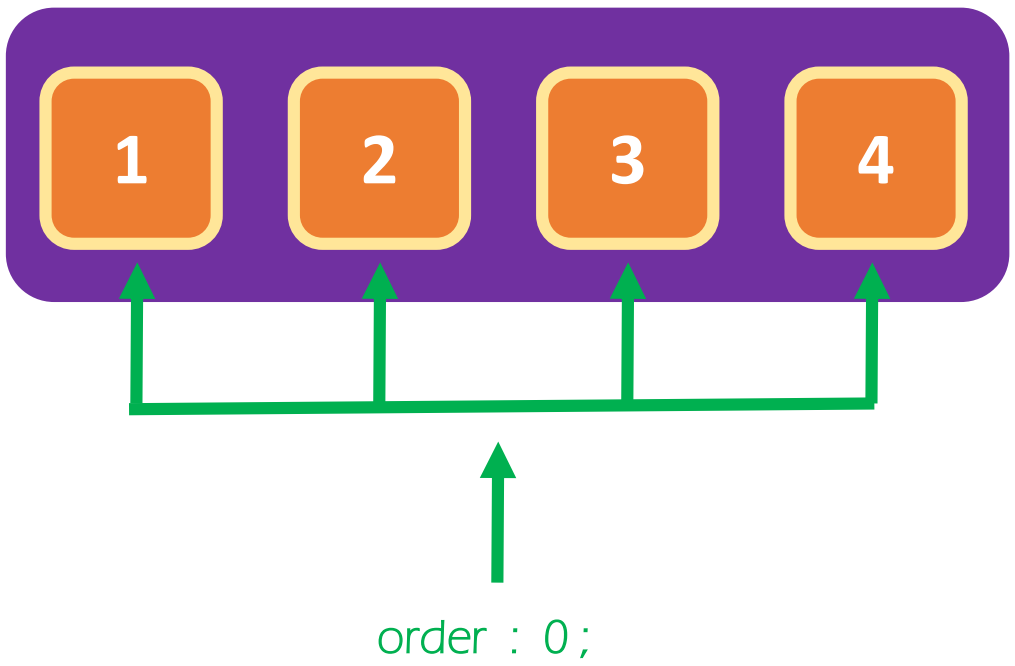
Note : order positions flex-items along the Main-axis

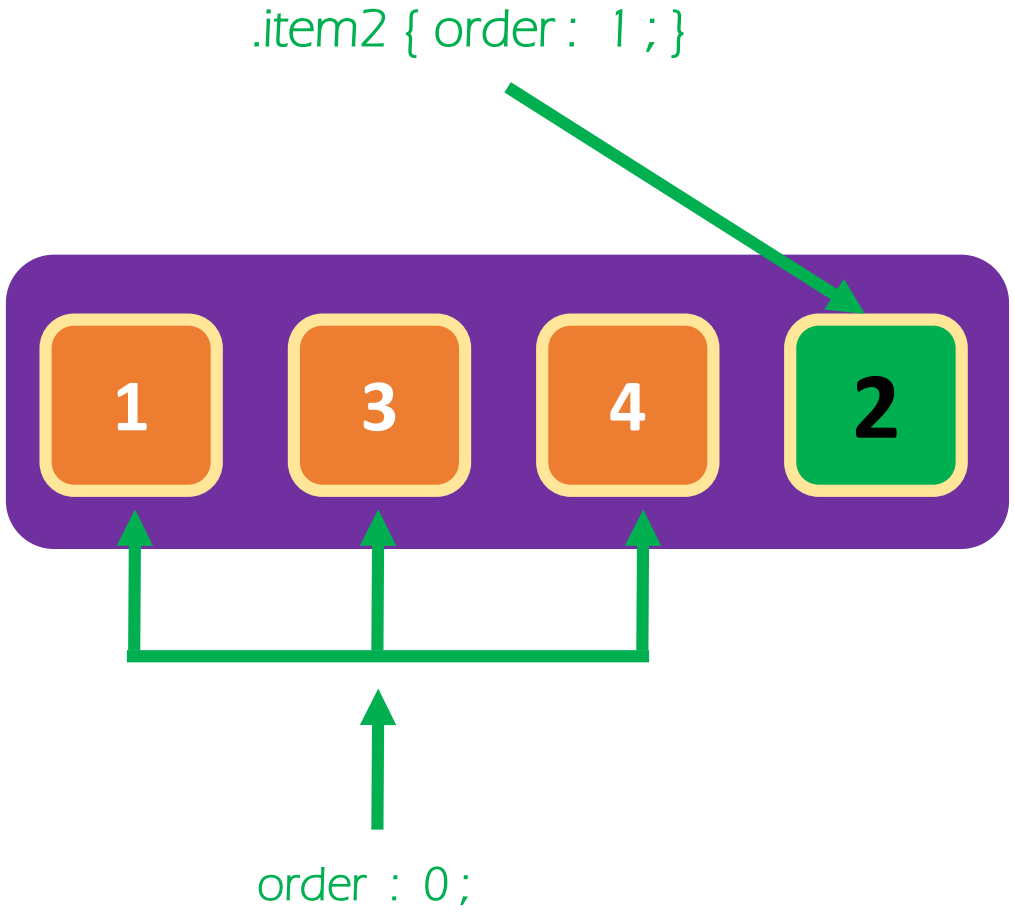
order :



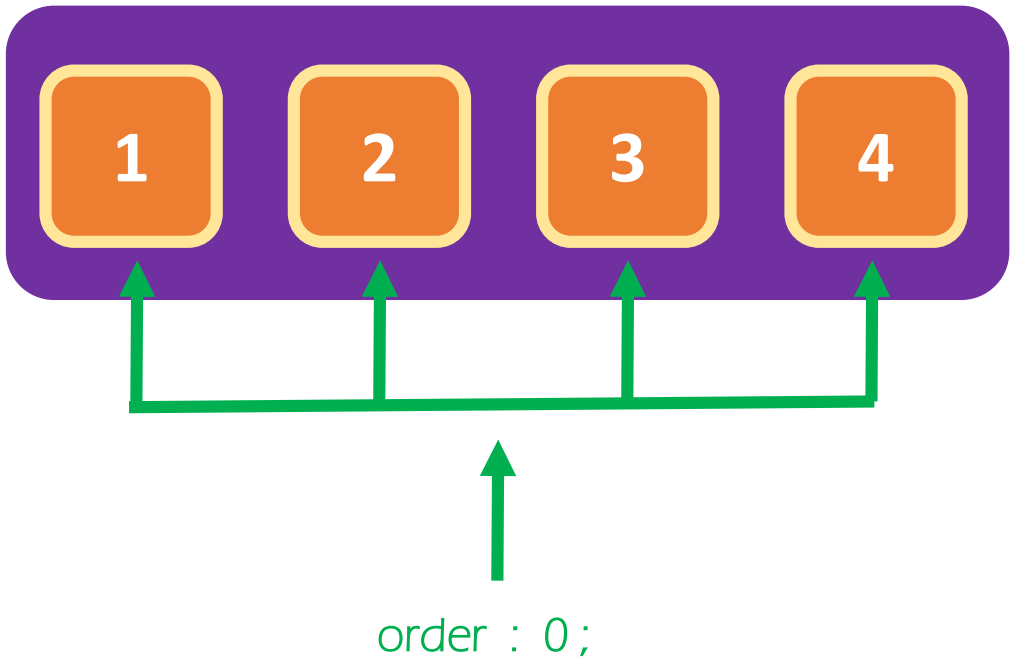
By default , flex items are laid out in the order in which they appear .

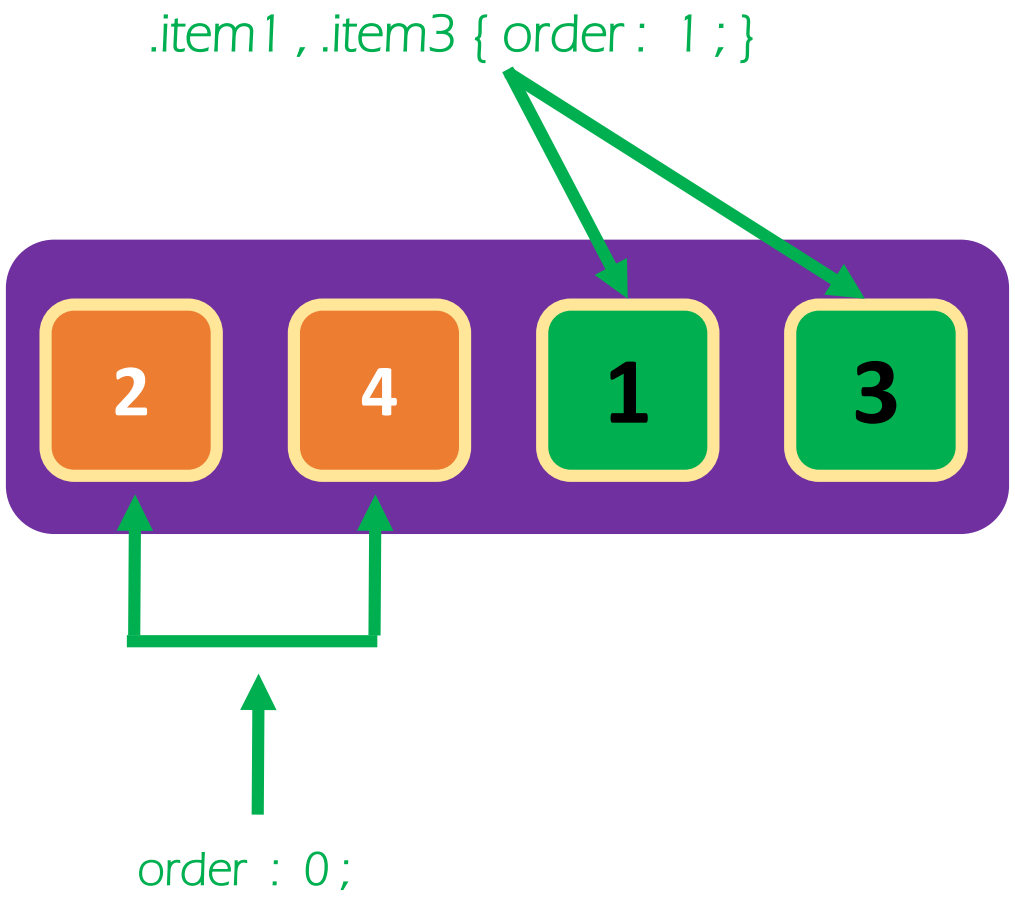
order :



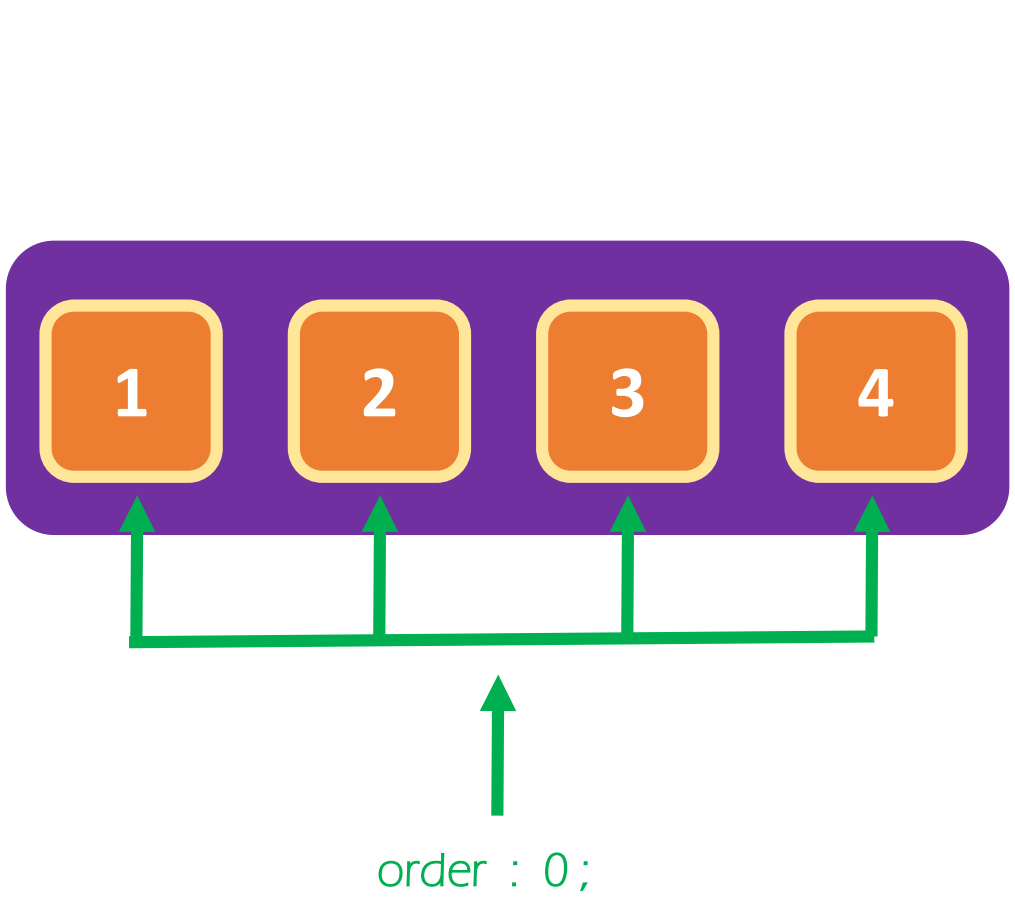


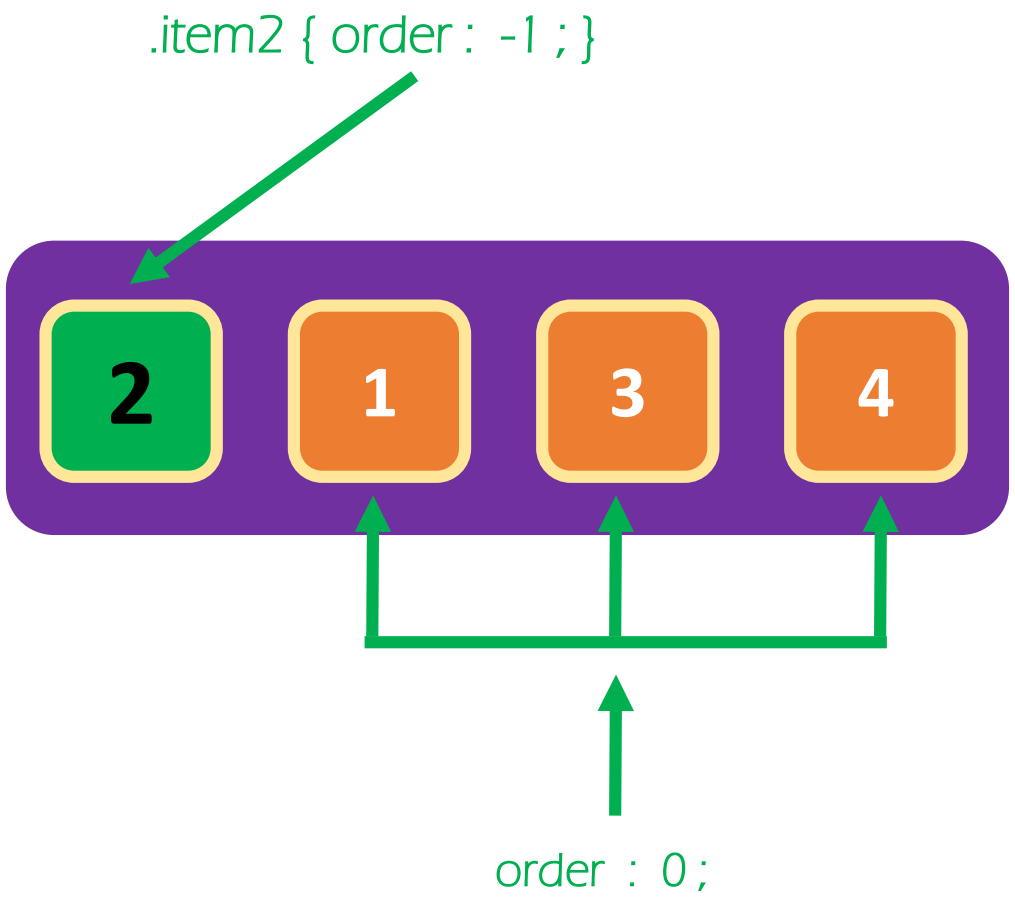
order :





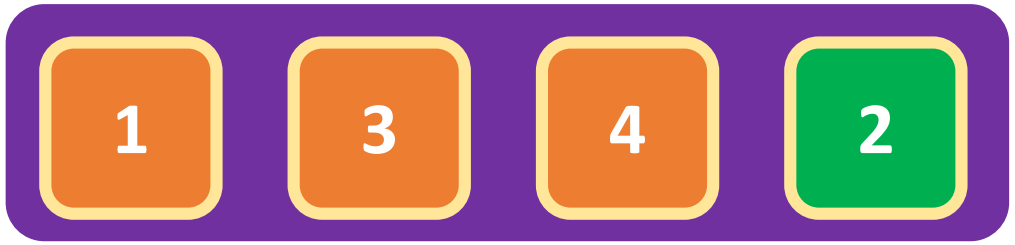
order :





order :

order : 0 ;　　order : 0 ;　　order : 0 ;　　order : 0 ;



order : 0 ;　　order : 0 ;　　order : 0 ;　　order : 1 ;

order : -1 ;　　order : 0 ;　　order : 0 ;　　order : 0 ;

Section 3 :

Working On Flex Items

Lecture 3.8 :

align-self Property

What Is align-self ?

- The align-self defines the alignment of individual flex item along the **cross-axis** .

align-self :

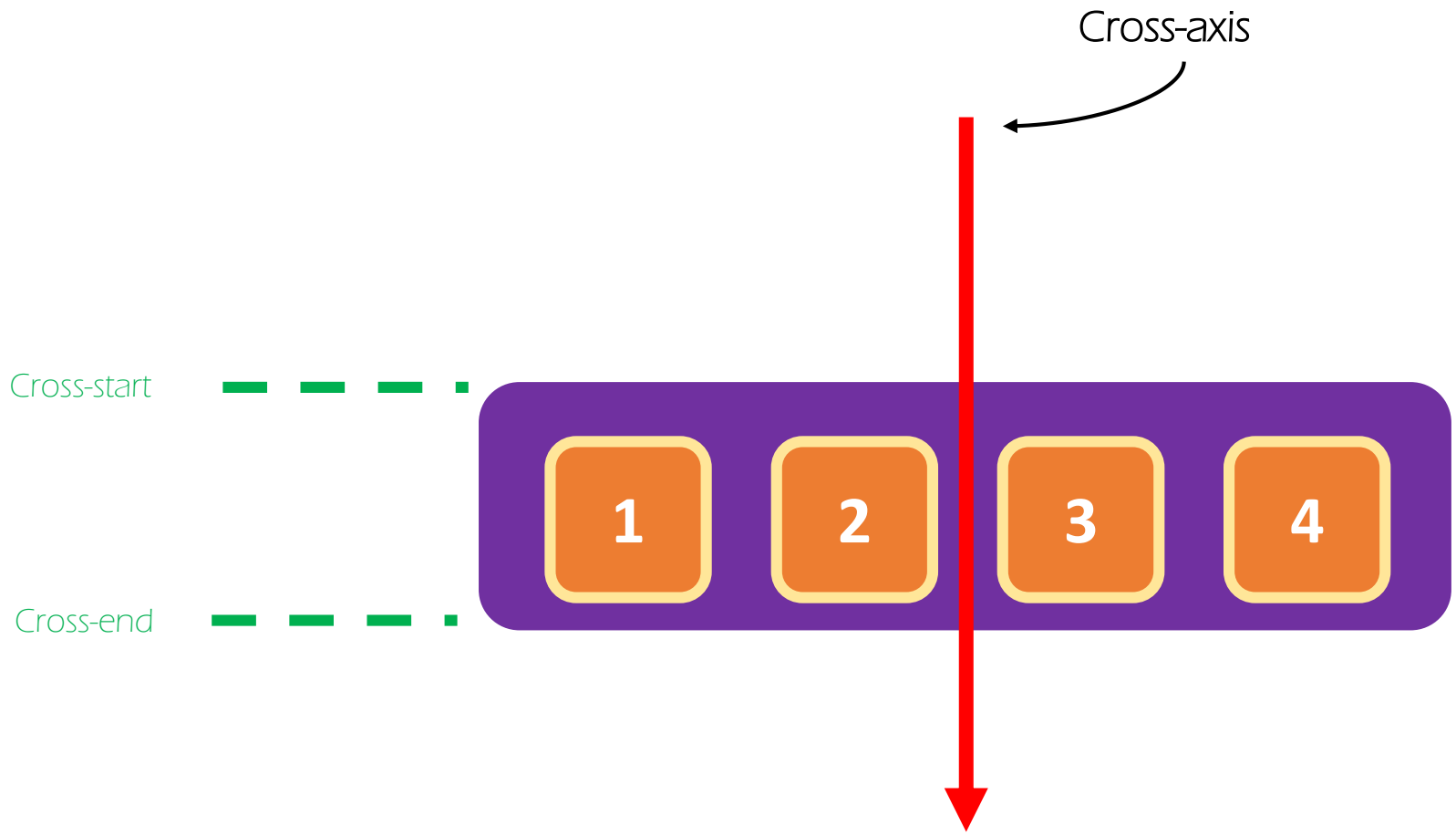
- The **align-self** property specifies the alignment for the selected flex item inside the flex container . It accepts 6 values:

  - **auto** - Selected flex item inherits it’s flex container's align-items property . This is default .
  - **stretch** - Stretches the selected flex item to fill the flex container .
  - **flex-start** - Aligns the selected flex item at the top of the flex container .
  - **center** - Aligns the selected flex item in the middle of the flex container .
  - **flex-end** - Aligns the selected flex item at the bottom of the flex container .
  - **baseline** - Aligns the selected flex item at the baseline of the flex container .

- Syntax for **align-self** property :

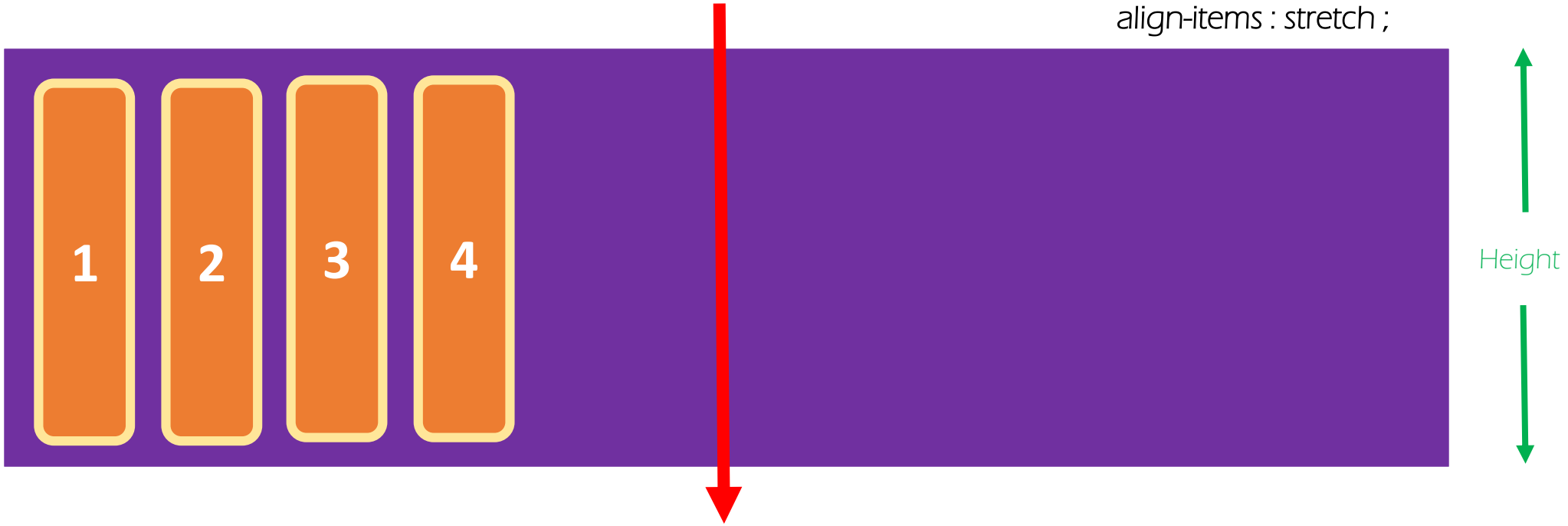
```
align-self: auto | stretch | flex-start | center | flex-end | baseline ;
```

Cross-axis :



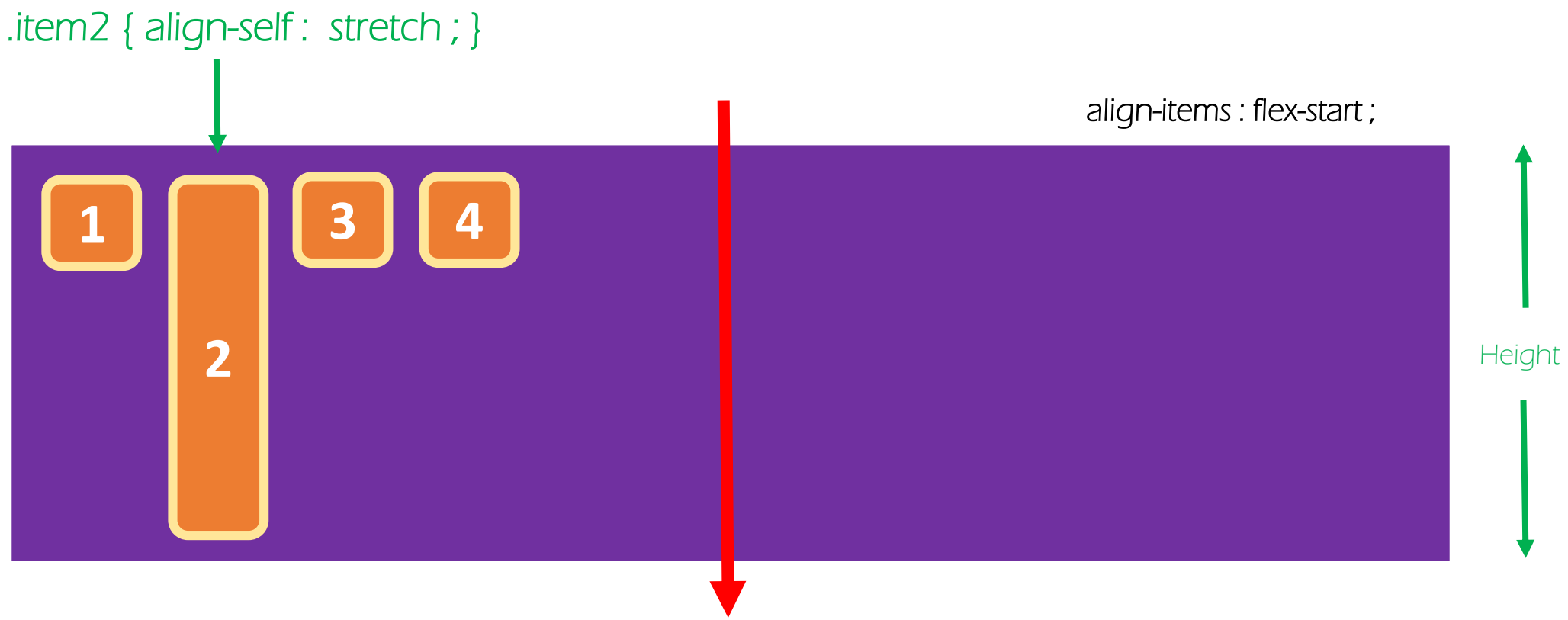
Note : align-self aligns the selected flex-item along the Cross-a

align-self : auto ;



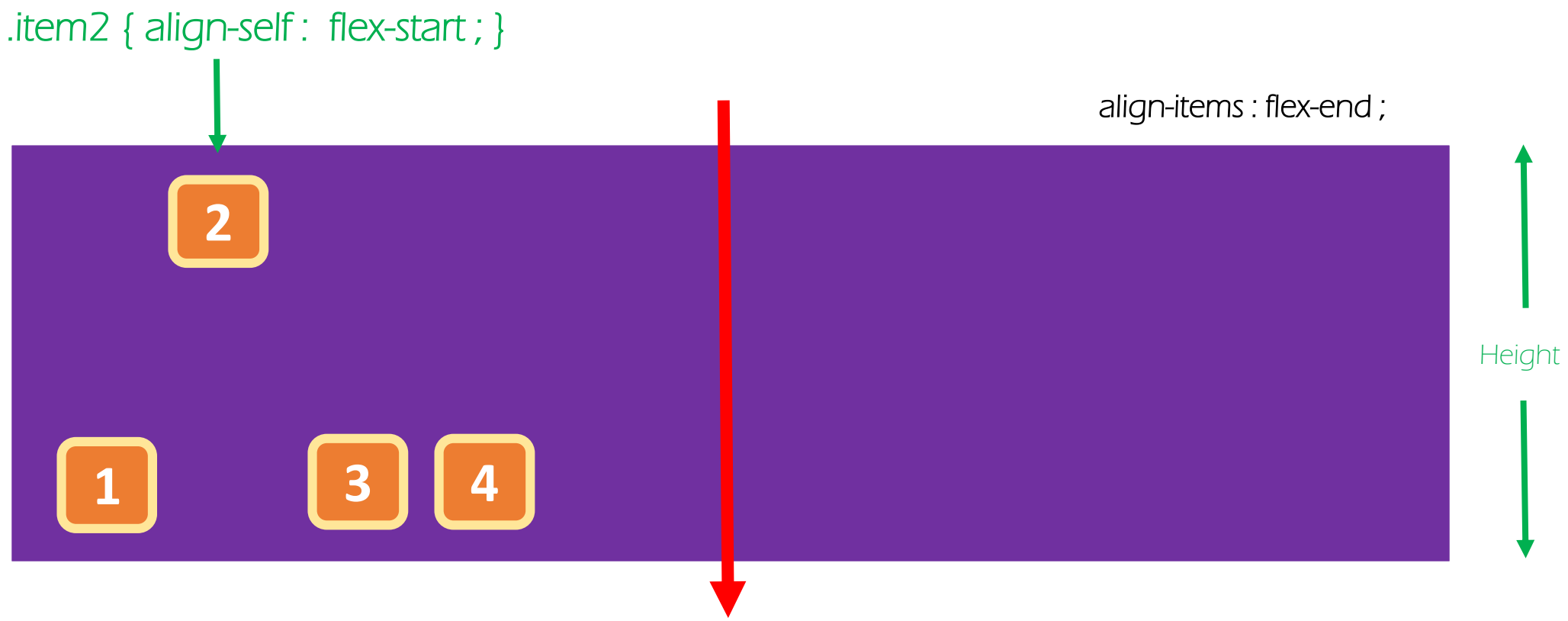
By default , all flex items inherit their flex container's align-items property so therefore stretched .

align-self : stretch ;



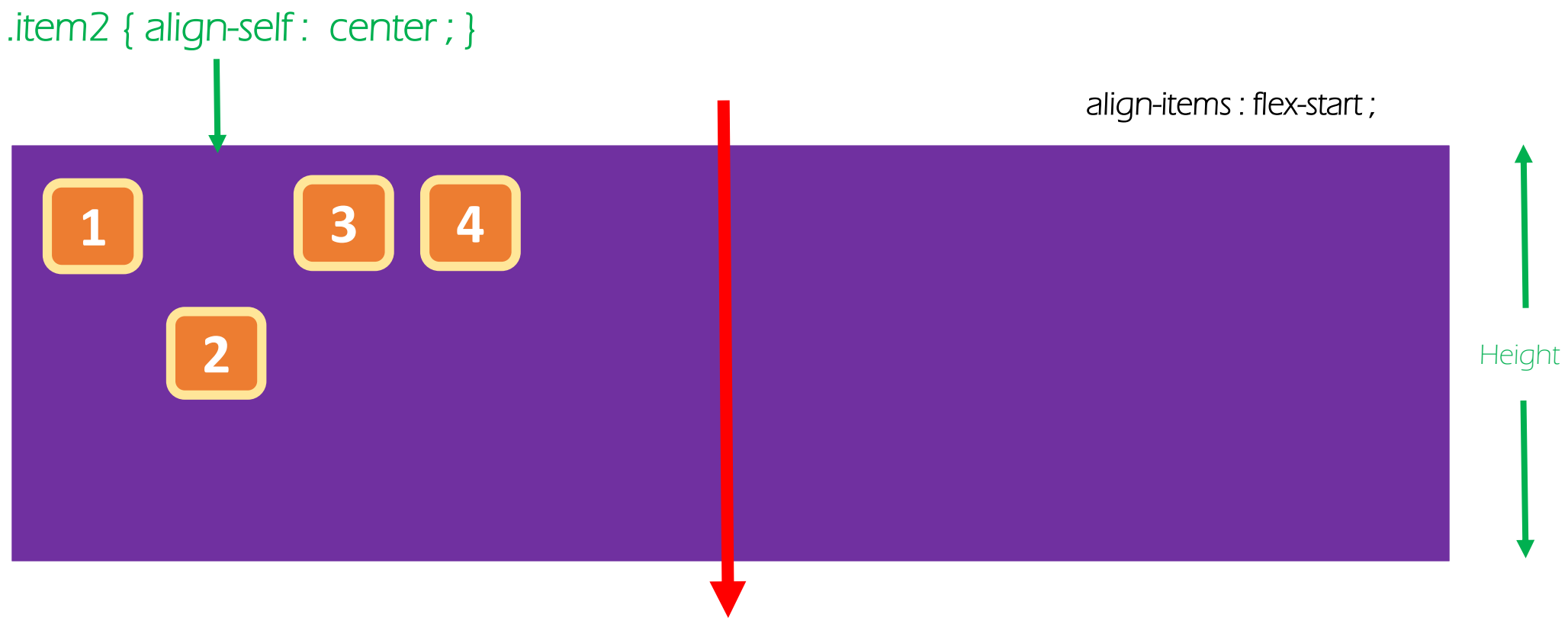
The selected flex item is stretched to fill the flex container .

align-self : flex-start ;



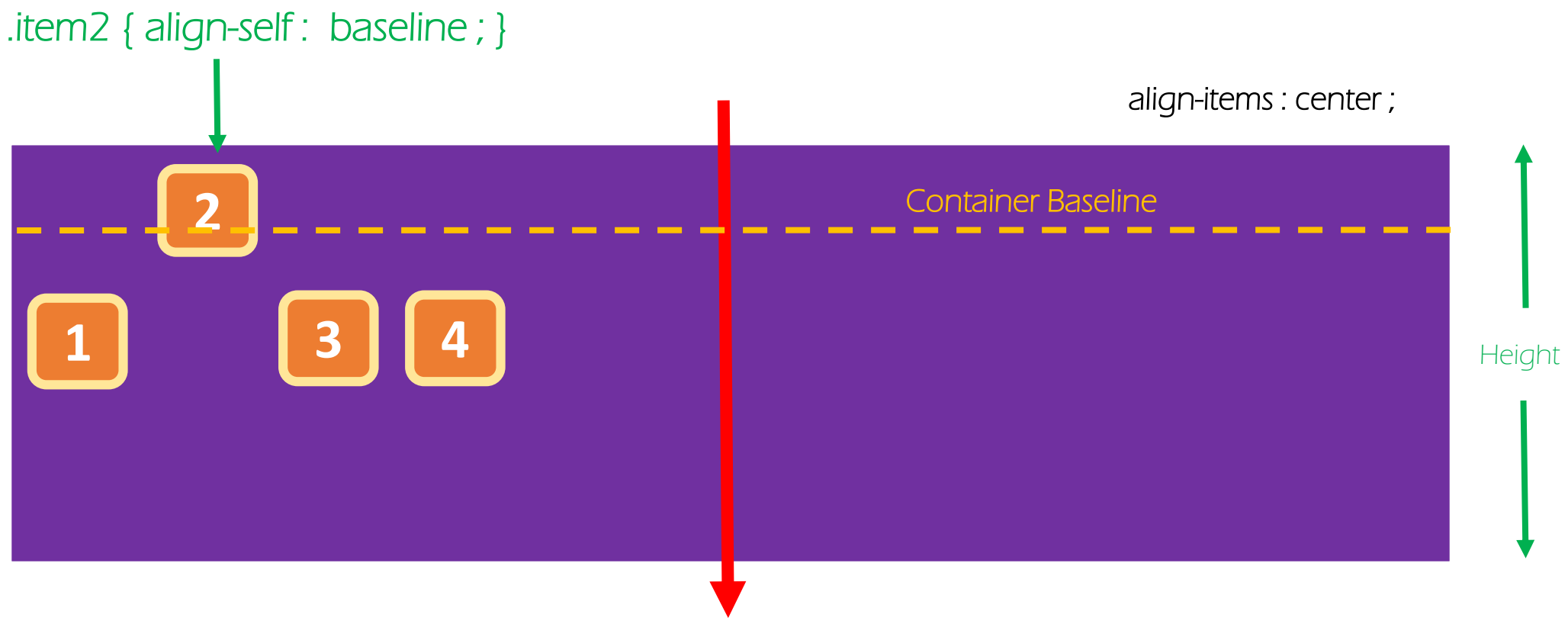
The selected flex item is positioned at the top of the flex container .

align-self : center ;



The selected flex item is positioned in middle of the flex container .

align-self : baseline ;



The selected flex item is positioned at the baseline of the flex container .

Section 3 :

Working On Flex Items

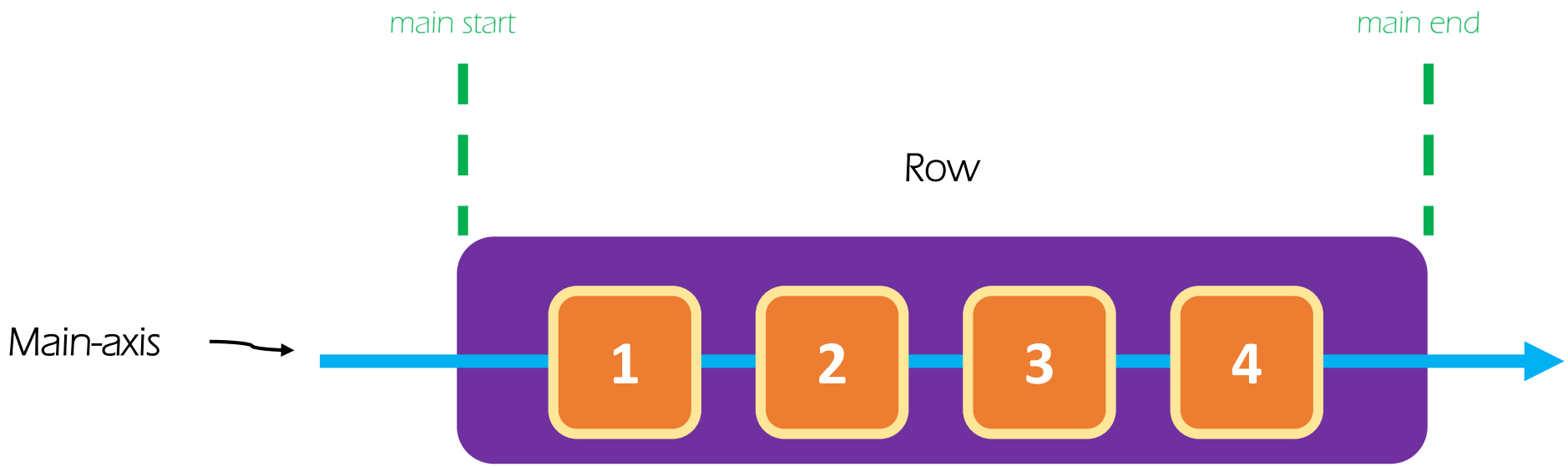
Lecture 4.8 :

flex-grow Property

The Superpower of Flex Items ;

- **1. flex-grow :** Decides how much a flex item should grow , if there is **extra** space .
- **2. flex-shrink :** Decides how much a flex item should shrink , if there is **no extra** space .
- **3. flex-basis :** Defines the **initial main-size** of a flex item .

order :



Note : flex-grow , flex-shrink & flex-basis , all three act along the Main-axis

What is flex-grow ? ;

- The flex-grow defines how much a flex item should grow relative to other flex items if there is an extra space available .

flex-grow :

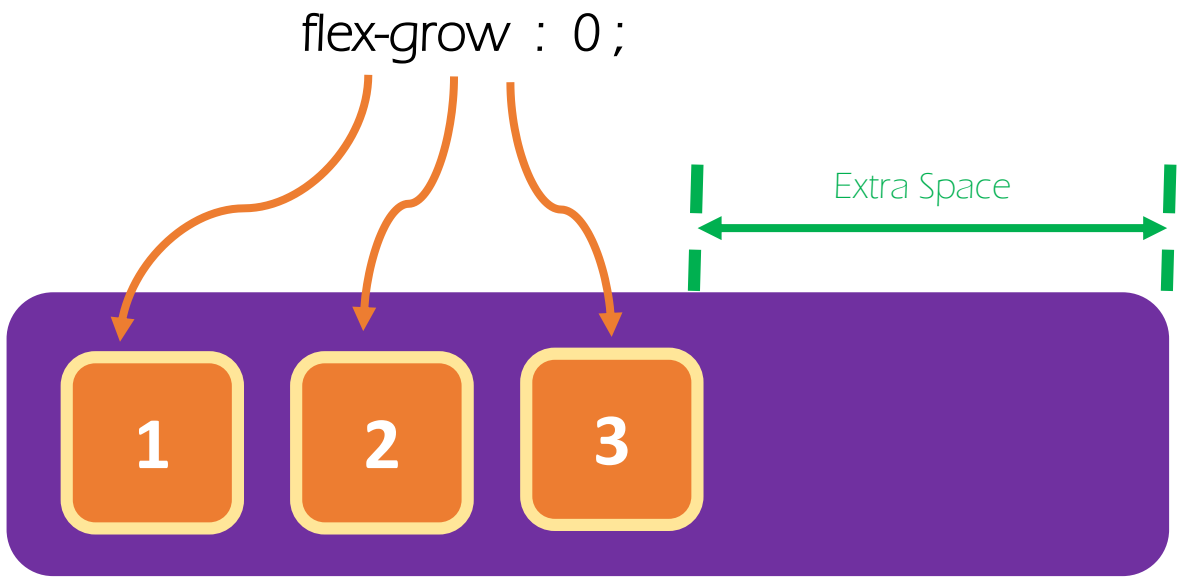
- The flex-grow property specifies how much a flex item can grow relative to the other flex items . It accepts a unitless positive number as a value . Negative numbers are invalid .

  - **0** - This is default .
  - **<number>** - Any unitless positive number.

- Syntax for **flex-grow** property :

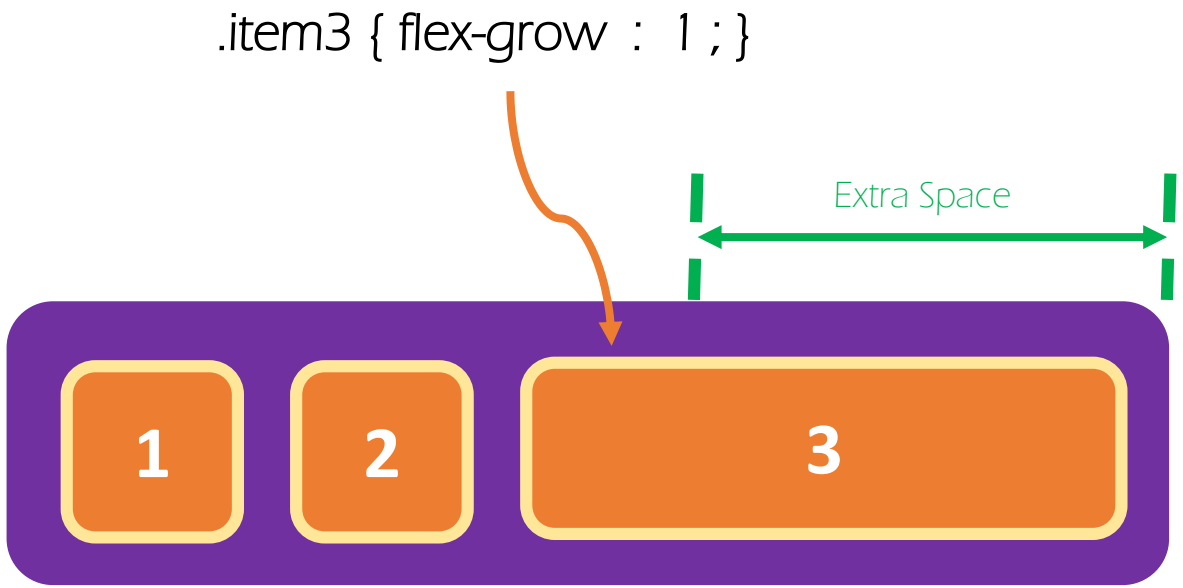
```
flex-grow : 0 | <number> ;
```

flex-grow :



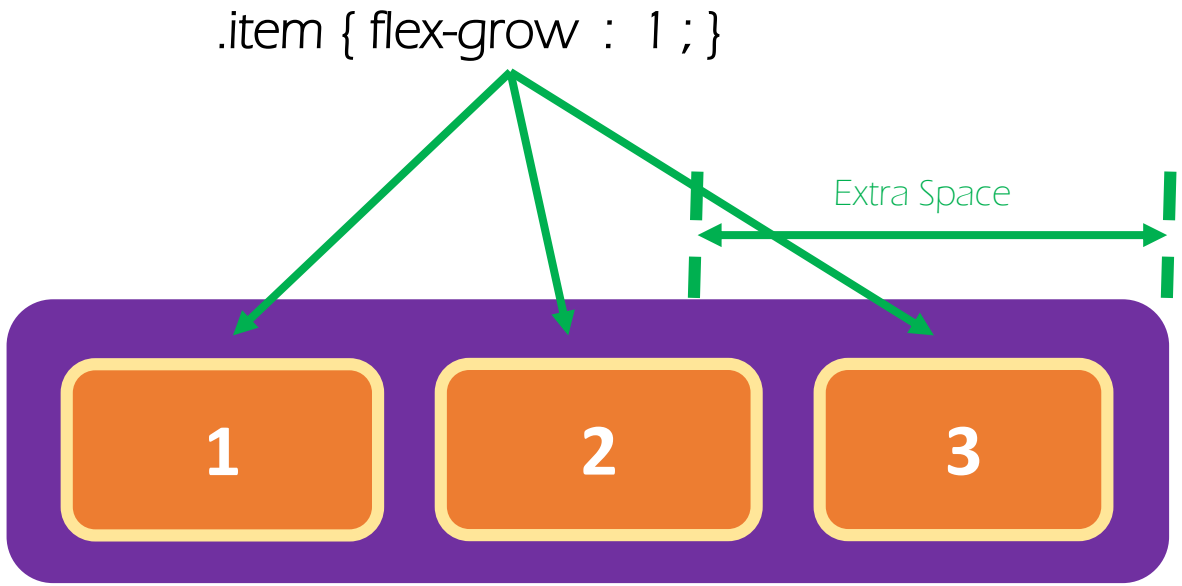
By default , flex items are with flex-grow set to zero , which means , they take only as much space as it is necessary to display their content .

flex-grow :



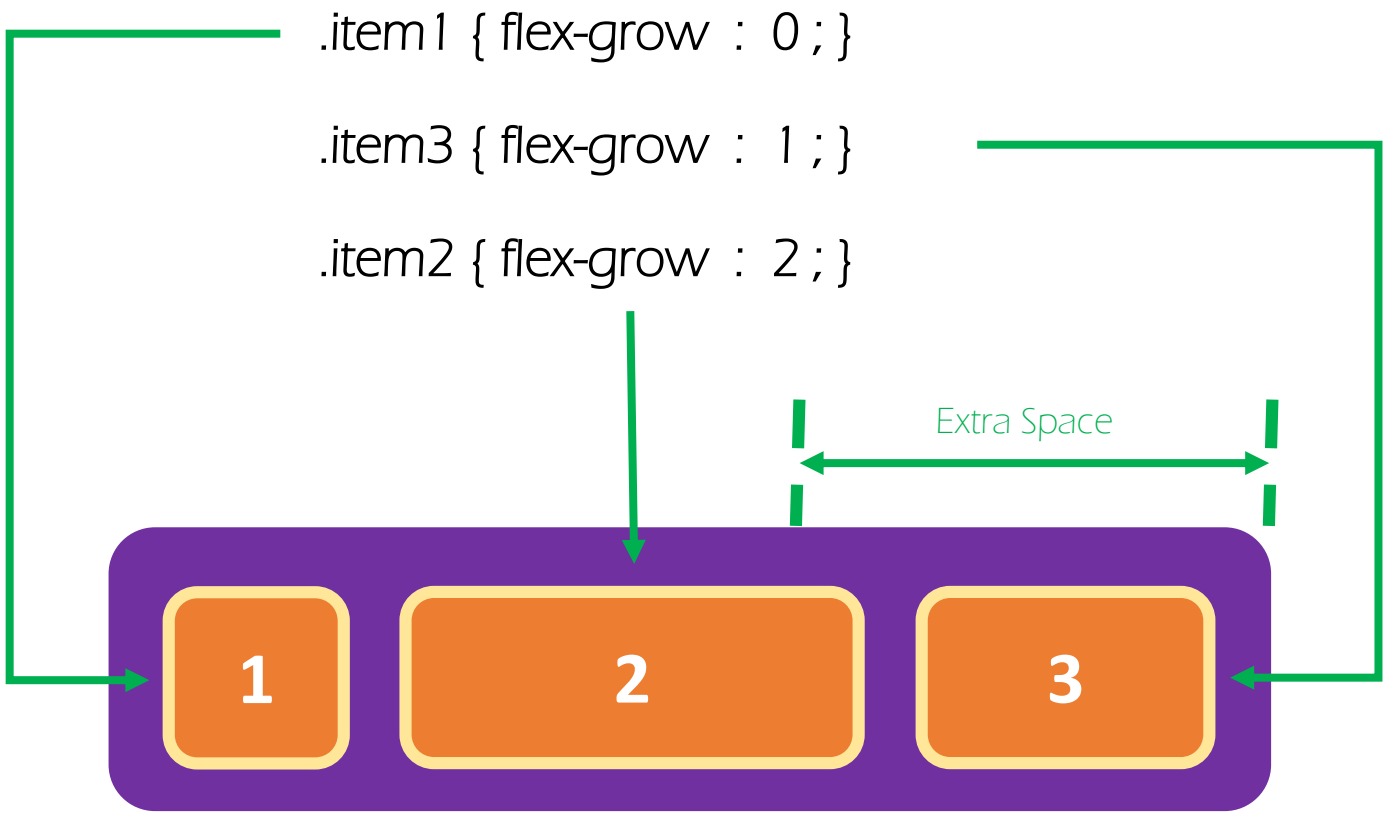
The flex-grow allows us to increase the width of flex-item , so it nicely expand & take all the available extra space of it's flex container .

flex-grow :



If all flex-items have same grow factor , we get flex-items that grow equally and fill up the entire flex container .

flex-grow :

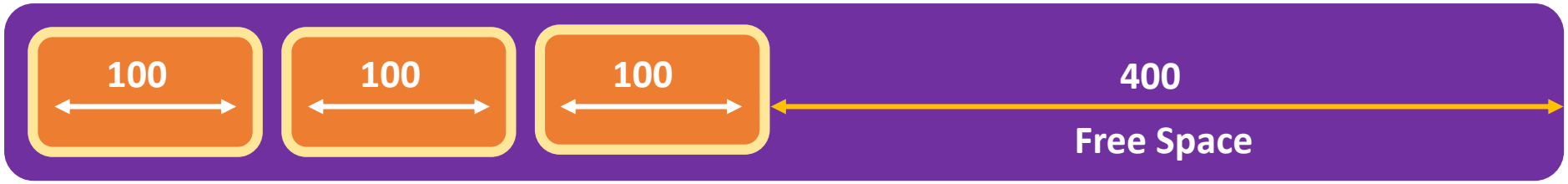


If one of the flex-item has grow factor of 2 , that would mean , that flex-item will take twice as much space as the ones that have grow factor set to 1 .

flex-grow Calculation :

$$\text{new width} = \left(\frac{\text{flex grow}}{\text{total flx grow}} * \text{free space} \right) + \text{width}$$

Width :

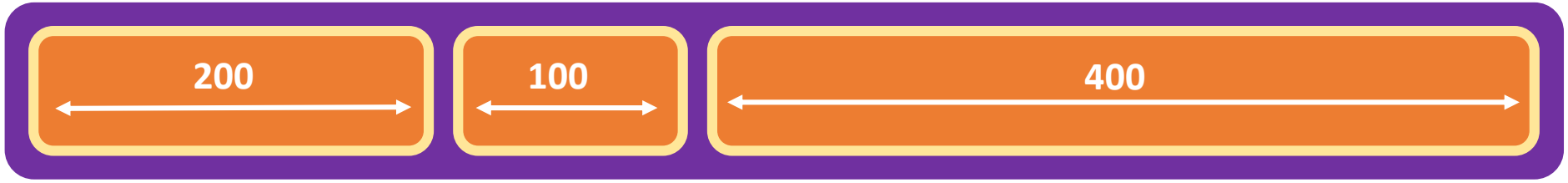


flex–grow : 1 0 3

Calculation :

$$\left(\frac{1}{4} * 400\right) + 100 \qquad \left(\frac{0}{4} * 400\right) + 100 \qquad \left(\frac{3}{4} * 400\right) + 100$$

New Width :



Section 3 :

Working On Flex Items

Lecture 5.8 :

flex-shrink Property

What is flex-shrink ? ;

- The flex-shrink defines how much a flex item should shrink relative to other flex items if there is not enough space available.
- The flex-shrink is only important if the flex container has the **nowrap** setting , so that all of the flex items sit next to each other in a single row .

flex-shrink Property :

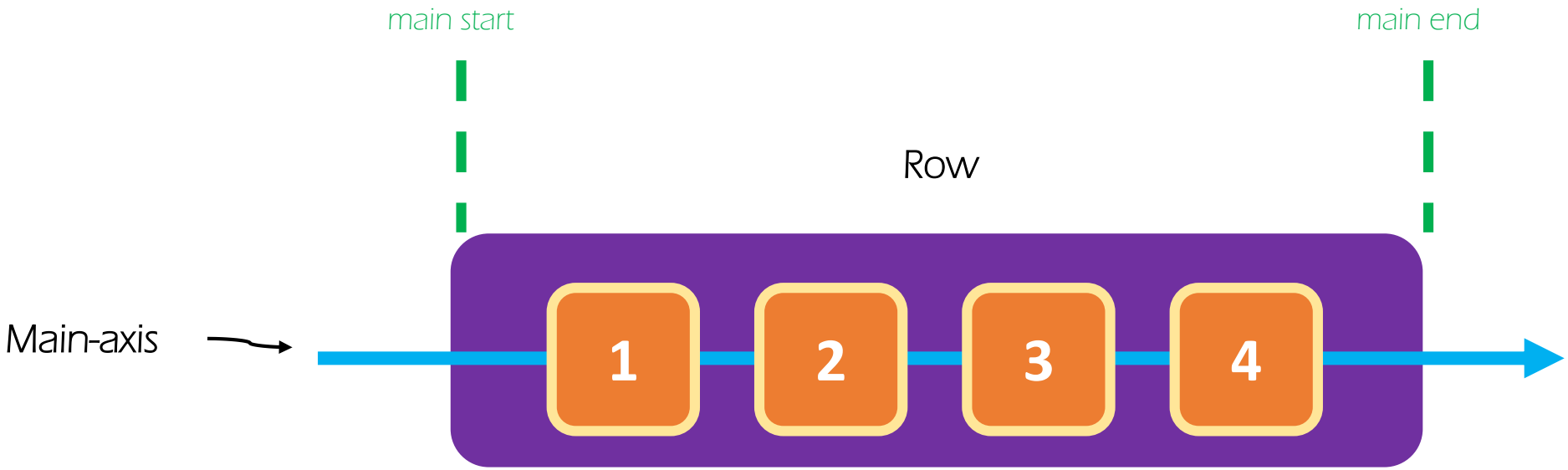
- The flex-shrink property specifies how much a flex item can shrink relative to the other flex items . It accepts a unitless positive number as a value . Negative numbers are invalid .

  - **1** - This is default .
  - **<number>** - Any unitless positive number.

- Syntax for **flex-shrink** property :

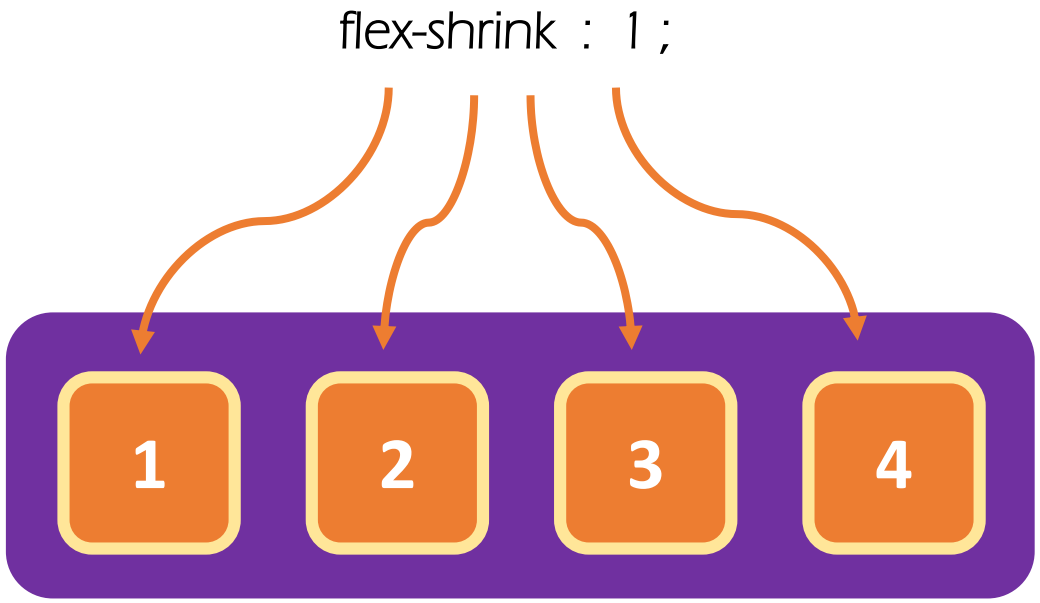
```
flex-shrink : 1 | <number> ;
```

flex-shrink :



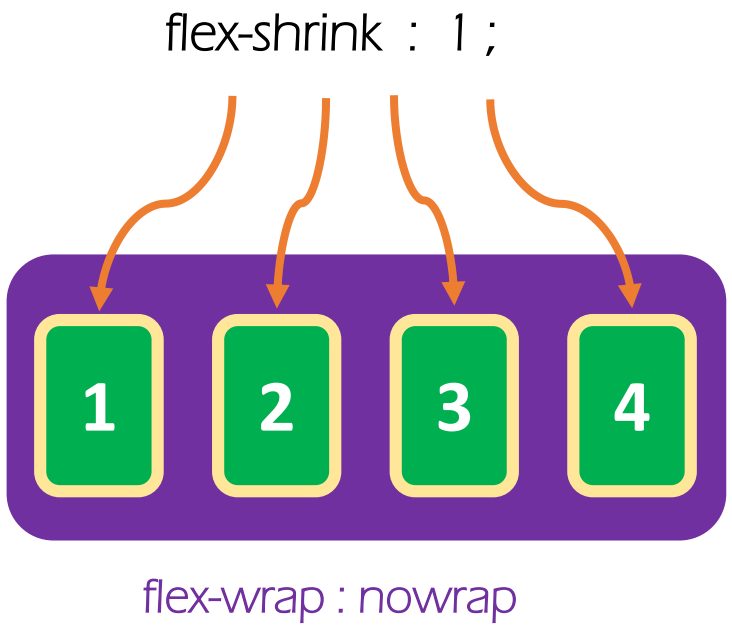
Note : flex-shrink acts along the Main-axis

flex-shrink :



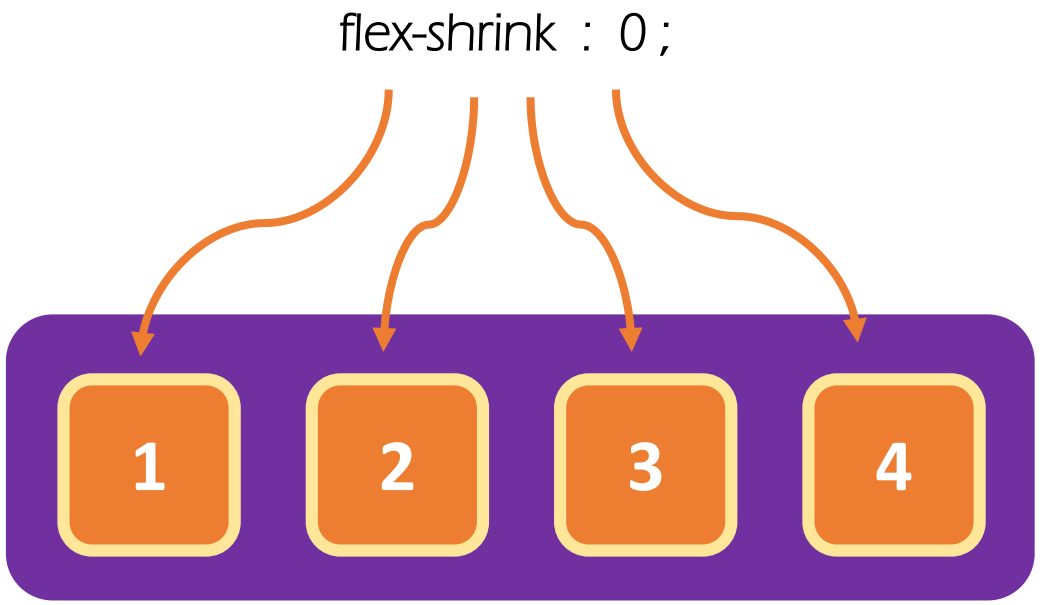
flex-wrap : nowrap

flex-shrink :



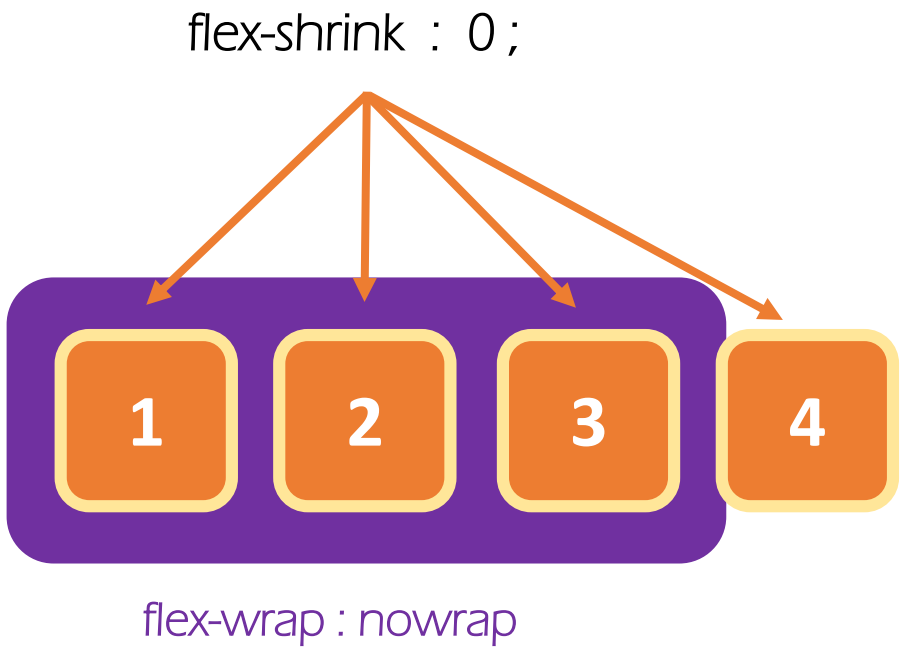
By default , all flex items have flex-shrink set to 1 , which means , they are allowed to shrink equally if there is not enough space to accommodate .

flex-shrink :



flex-wrap : nowrap

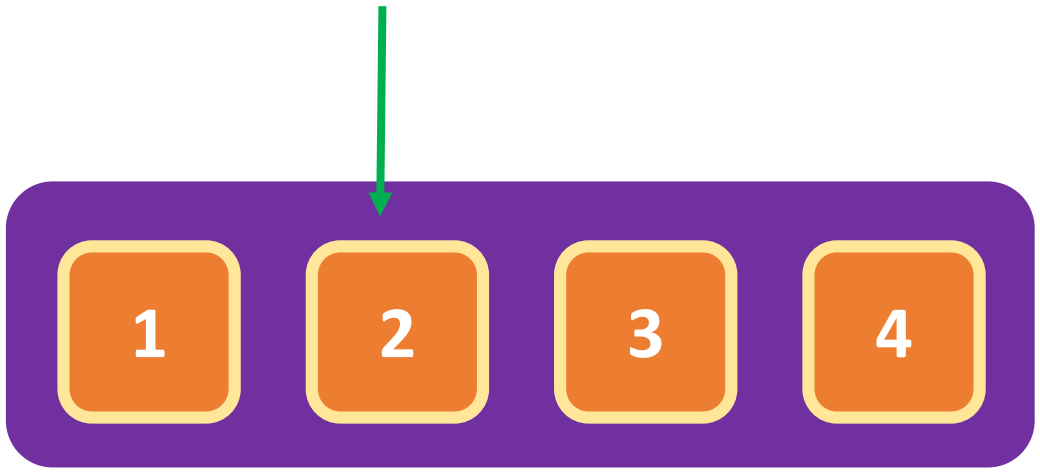
flex-shrink :



If all flex items have flex-shrink set to zero (don’t shrink) , they will simply overflow their flex-container if there is not enough space .

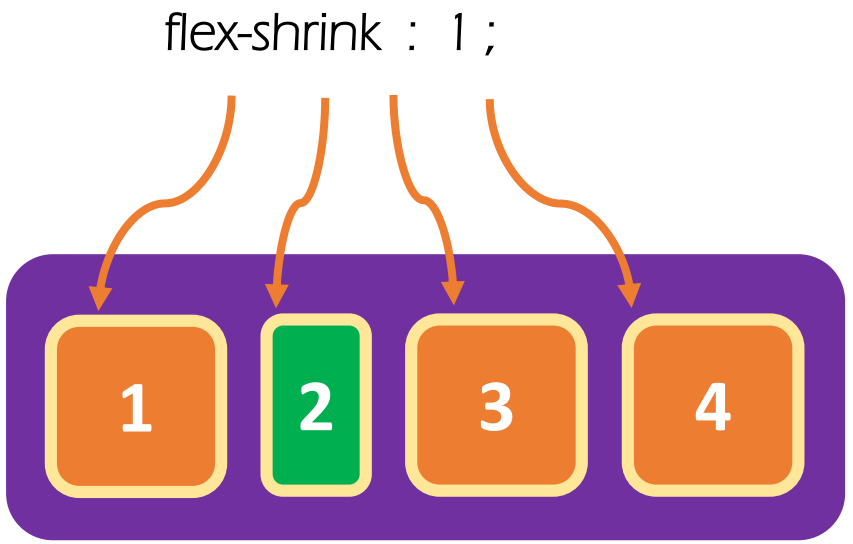
flex-shrink :

.item2 { flex-shrink : 2 ; }



flex-wrap : nowrap

flex-shrink :



flex-wrap : nowrap

Setting one flex-item with flex-shrink of 2 would mean that , that flex-item will shrink 2 times more as compare to others .

Section 3 :

Working On Flex Items

Lecture 6.8 :

flex-basis Property

What is flex-basis ? ;

- It specifies the **initial main size** of the flex-item , before any available space is distributed according to the flex factors (flex-grow & flex-shrink) .

flex-basis :

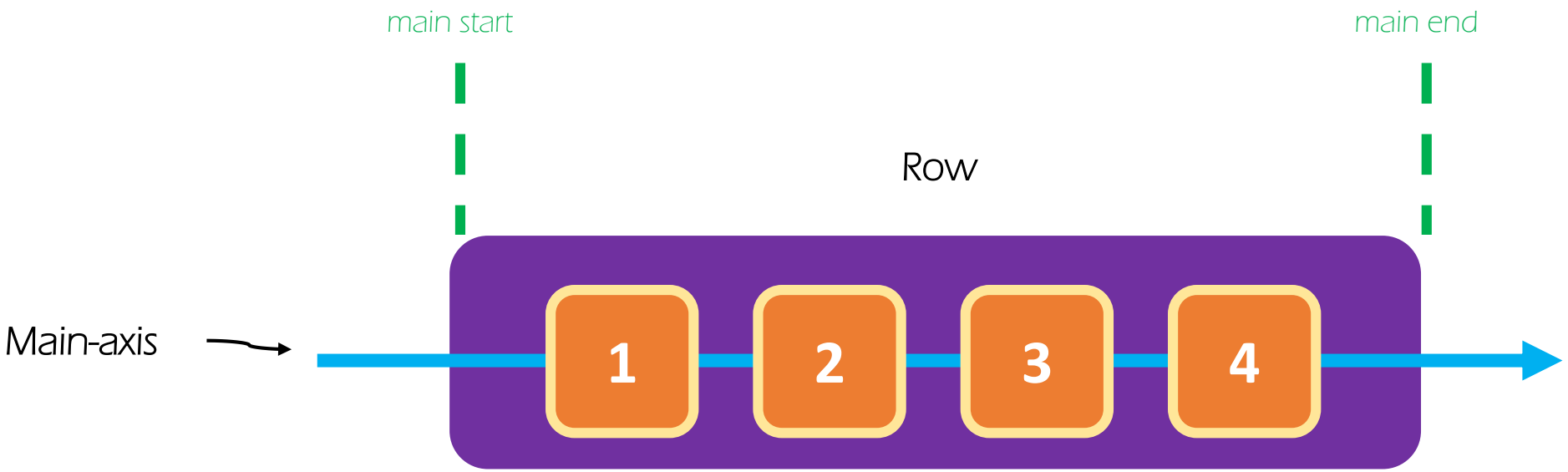
- The **flex-basis** property specifies the initial main size of the flex-item , before any available space is distributed. It takes any CSS size .

  - **auto** - This is default .
  - **<length>** - Any CSS size - percentage | em | rem |pixel etc .

- Syntax for **flex-basis** property :

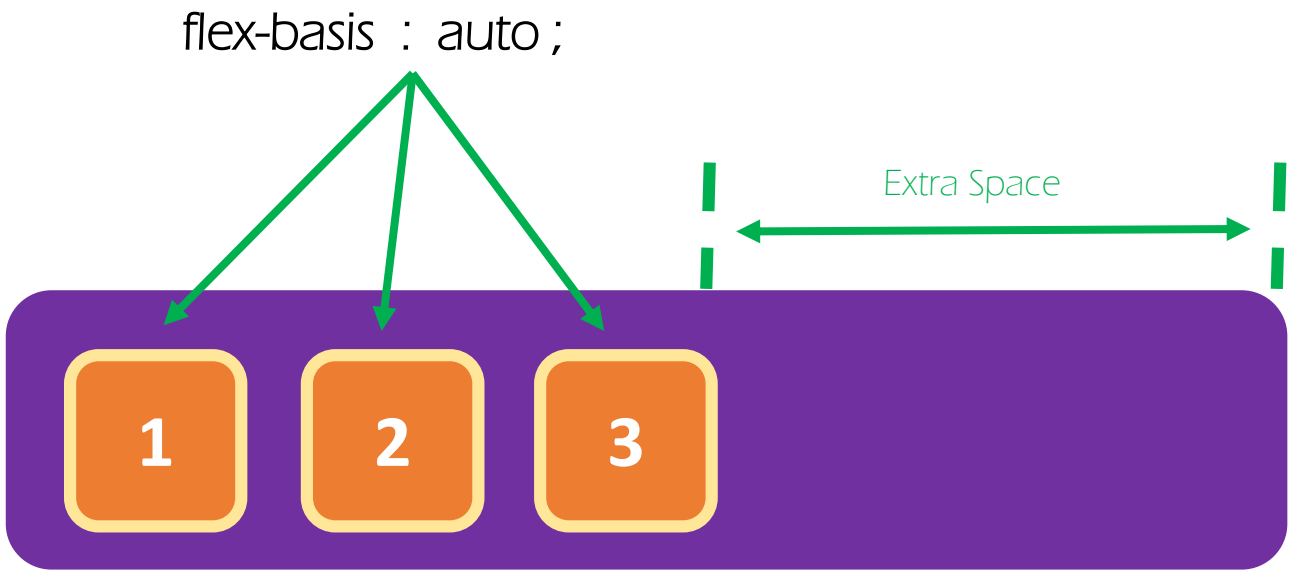
```
flex-shrink : auto | <length> ;
```

flex-basis :



Note : flex-basis acts along the Main-axis

flex-basis :

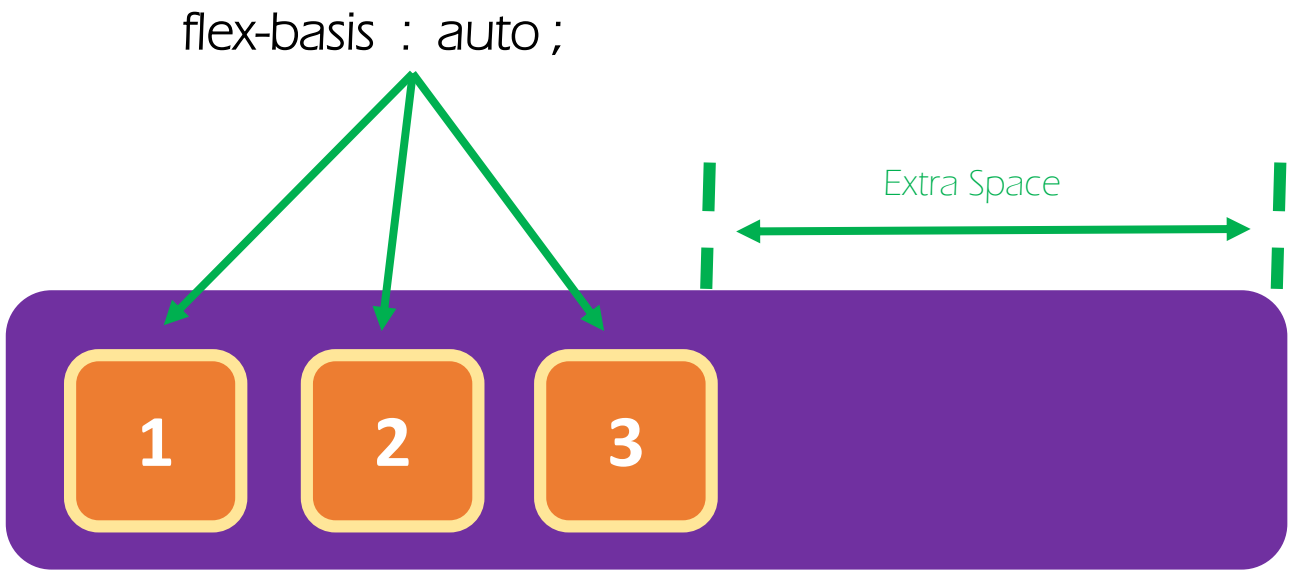


By default , the flex-basis property is set to auto , which means , the flex-basis falls back to the flex item’s width property .

flex-basis :

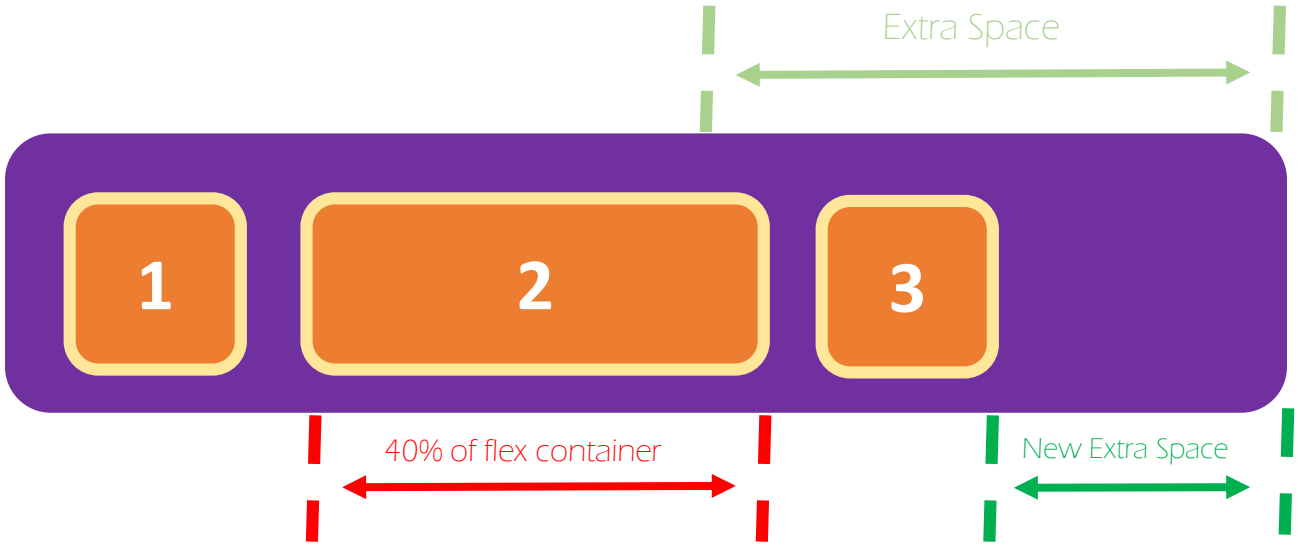
If the flex-items have flex-basis property set to zero , that means , they will take up only the necessary space to display their content .

flex-basis :

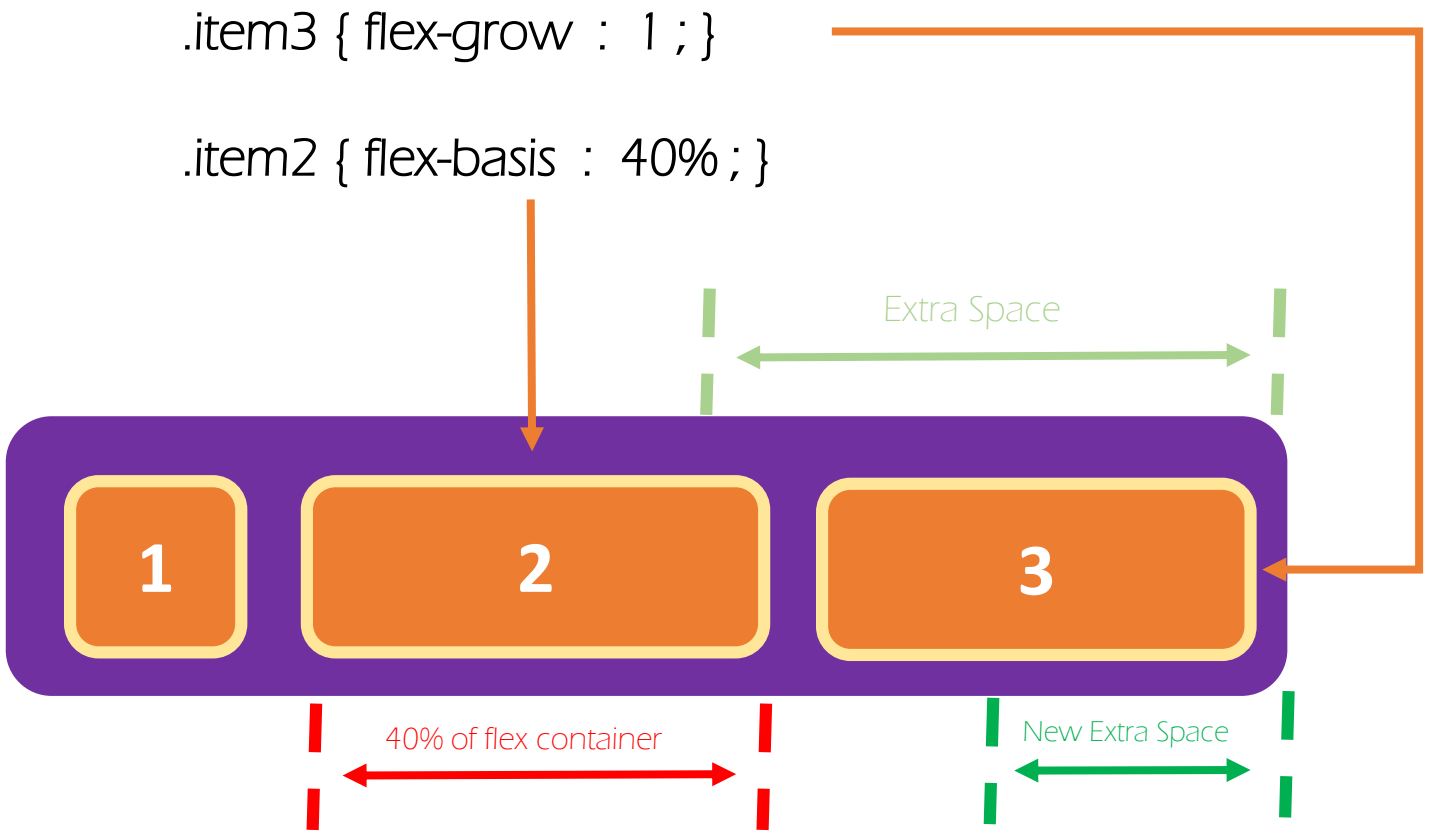


By default , the flex-basis property is set to auto , which means , the flex-basis falls back to the flex item’s width property .

flex-basis :



flex-basis :



The flex-basis specifies the initial main size of a flex-item , before any free space is distributed according to the flex factors . It takes any CSS size in % , rem , em , px , etc .

flex-basis VS widths :

```
.child {
    flex-basis: 500px;
    width: 100px;
}
```


```
.child {
    flex-basis: 100px;
    min-width: 500px;
}
```


```
.child {
    flex-basis: 100px;
    max-width: 700px;
}
```

Section 3 :

Working On Flex Items

Lecture 7.8 :

The Shorthand ‘flex’ Property

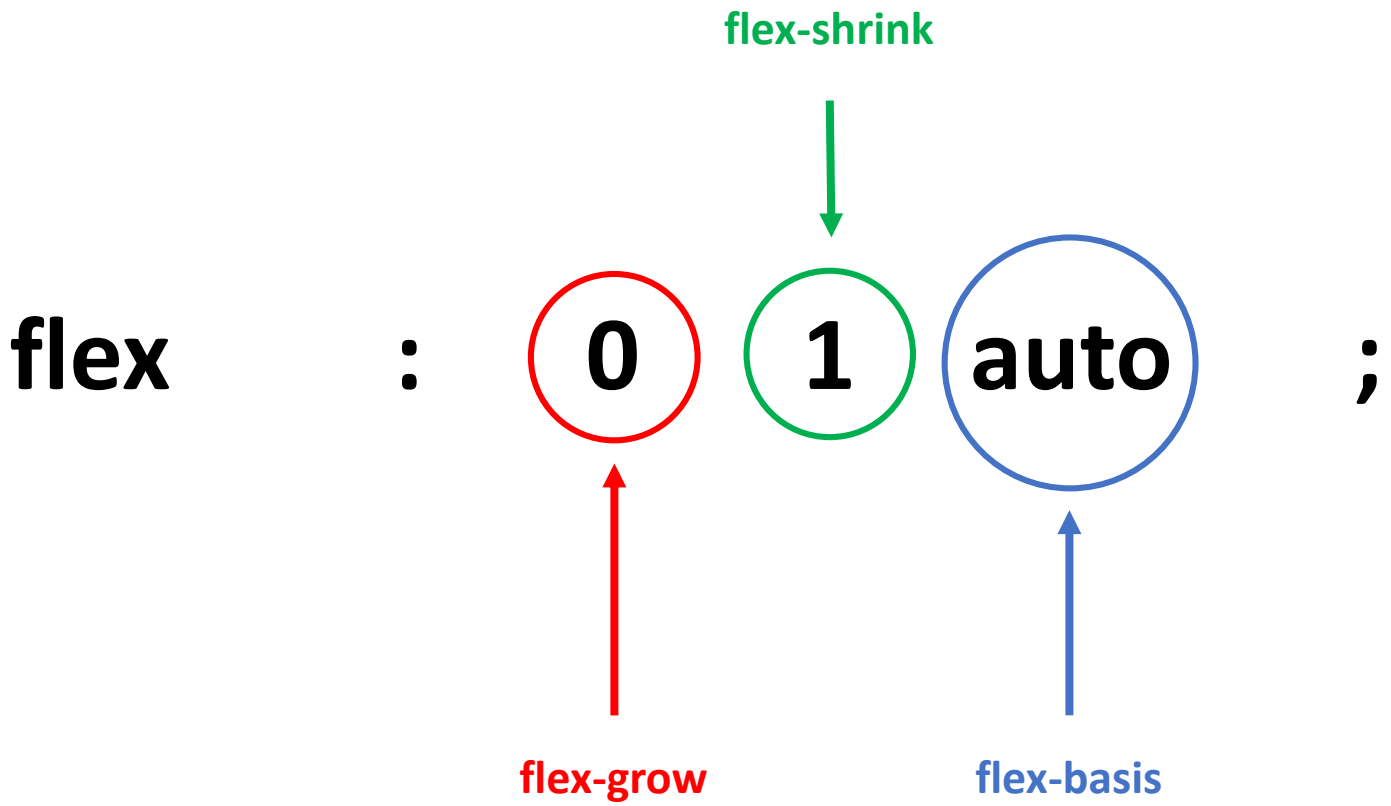
flex Property :

- The **flex** property is a shorthand for the **flex-grow** , **flex-shrink** and **flex-basis** properties .

  - **0 1 auto** - This is default .

- **flex** : **0 1 auto** ;

flex :



Section 3 :

Working On Flex Items

Lecture 8.8 :

Flex Item Properties Overview

Flex Item Properties Overview :

1. **oredr : 0** | <integer>
2. **align-self : auto** | stretch | flex-start | flex-end | center | baseline
3. **flex-grow : 0** | <integer>
4. **flex-shrink : 1** | <integer>
5. **flex-basis : auto** | <length>
6. **flex : 0 1 auto** [**flex-grow flex-shrink flex-basis**]

The End :)

Hope you enjoy it

Email : siavash.sattari.dev@gmail.com