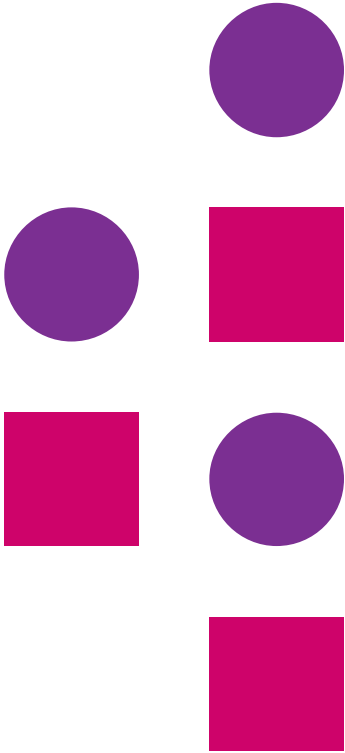
WSA | Forward looking IT finishing school

Flexbox Layout

Cascading Style Sheets (CSS3)

Table of Content

- Flexbox layout



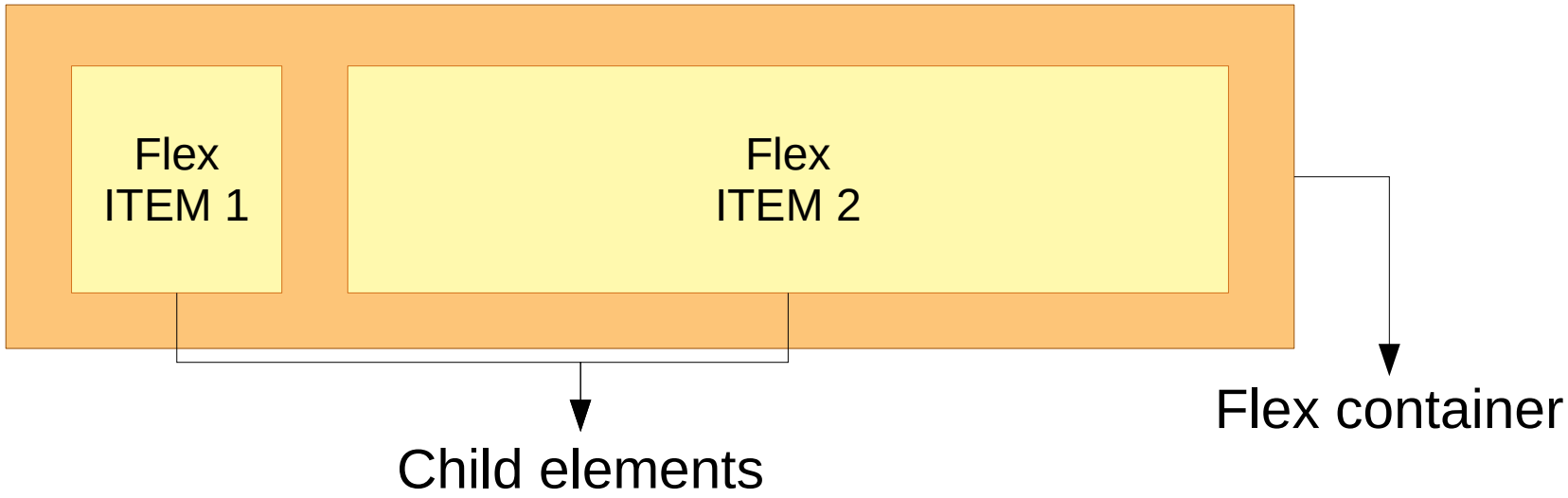
Flexbox

(Cascading Style Sheets 3)

Flexbox Layout

- The Flexible Box Layout, is used to design flexible responsive layout structure without having to use floats or positioning
- Flexbox consists of
  - Parent (called flex container)
  - And child elements (called flex items)

Container and child elements



Flexbox Layout

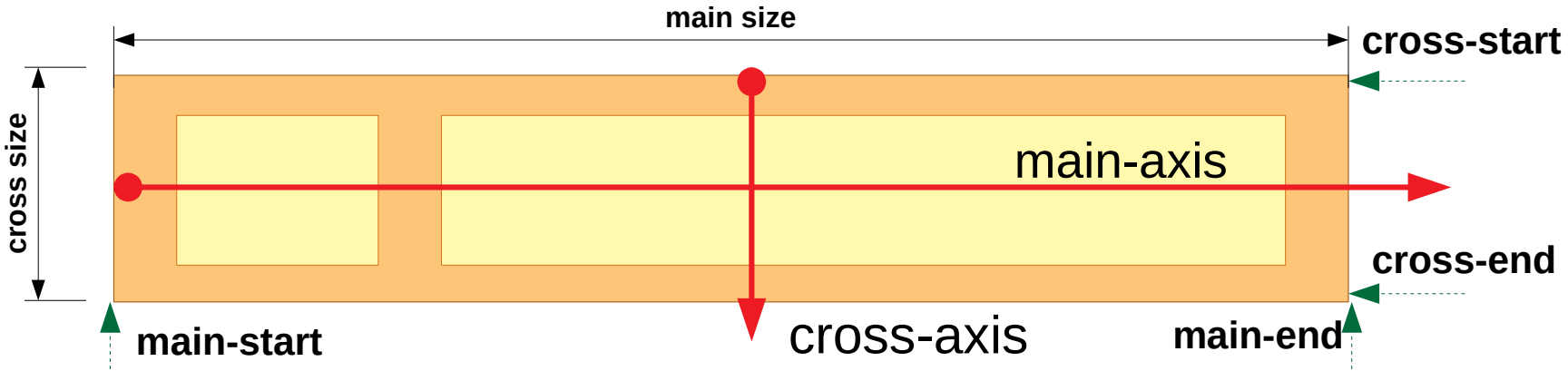
- The motivation of flex layout is to enable the container to alter its items' width/height (and order) to best fill the available space

  (mostly to accommodate to all kind of display devices and screen sizes)

- A flex container
  - Expands items to fill available free space
  - Or shrinks them to prevent overflow

Flexbox Layout

- The flexbox was designed as a one-dimensional layout model
- Which means that flexbox deals with layout in one dimension at a time (either as a row or as a column)



Flexbox Terminologies

- Basically, items will be laid out following
  - either the main axis (from main-start to main-end)
  - or the cross axis (from cross-start to cross-end)
- **Main axis** - It is the primary axis of container along which flex items are laid out, please note
  - It is not necessarily horizontal
  - It depends on the flex-direction property

Flexbox Terminologies

- **Main-start, main-end** - The flex items are placed within the container starting from main-start and going to main-end
- **Main size** - A flex item's width or height, whichever is in the main dimension, is the item's main size
- The flex item's main size property is either the ‘width’ or ‘height’ property, whichever is in the main dimension

Flexbox Terminologies

- **Cross axis** –
  - The axis perpendicular to main axis is called cross axis
  - Its direction depends on the main axis direction
- **Cross-start, cross-end** - Flex lines are filled with items and placed into the container starting on the cross-start side of the flex container and going toward the cross-end side
- **Cross size** - The width or height of a flex item, whichever is in the cross dimension, is the item's cross size

Flexbox flex-direction

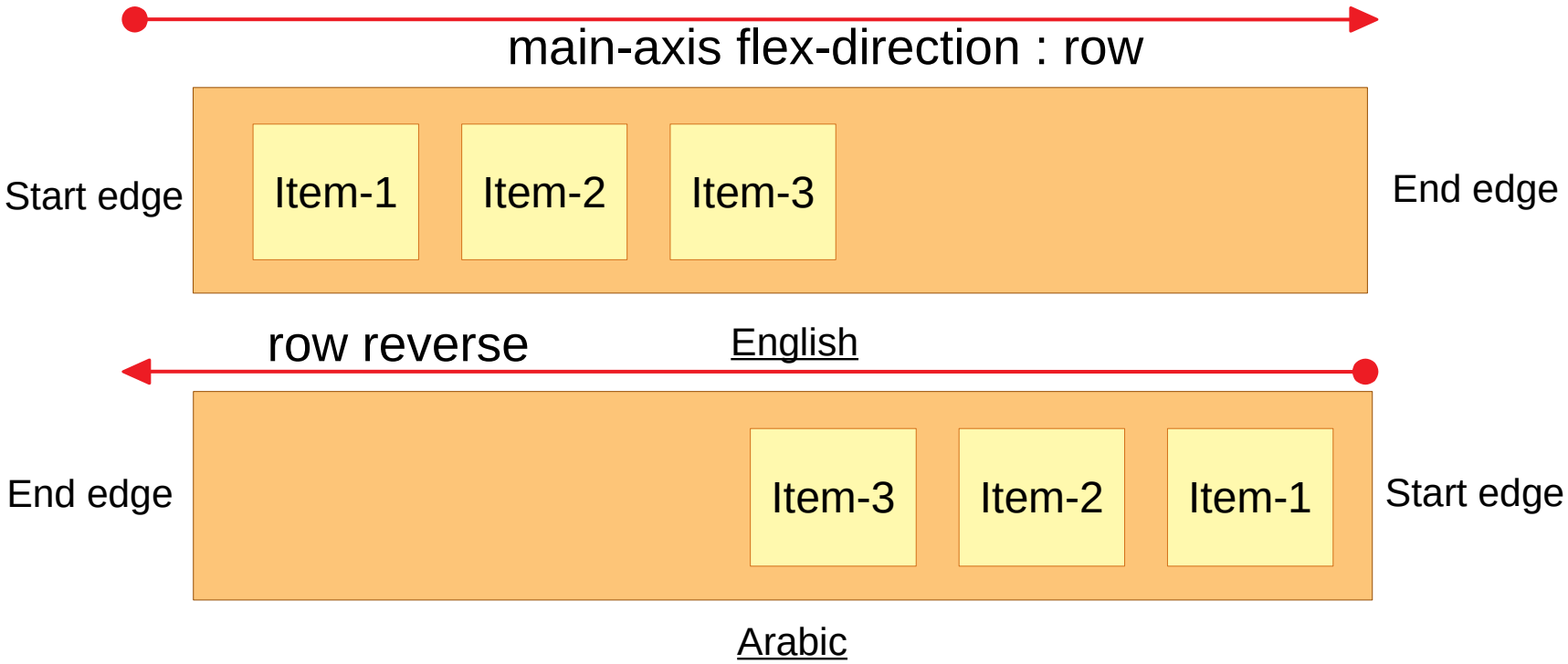
- The main axis is defined by flex-direction property, which has four possible values:

Syntax: */* row is default value */*

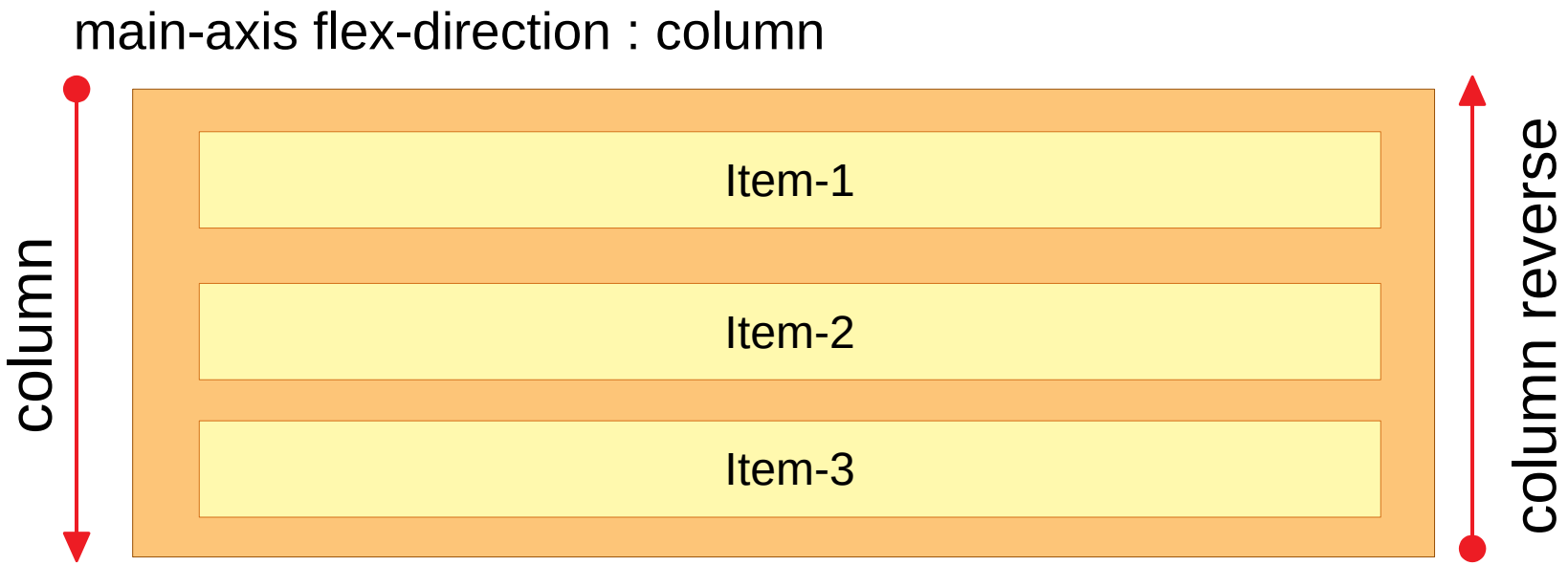
flex-direction: row | row-reverse | column | column-reverse

- If you choose row or row-reverse, your main axis will run along the row in inline direction
- Choose column or column-reverse and your main axis will run from the top of the page to the bottom — in block direction

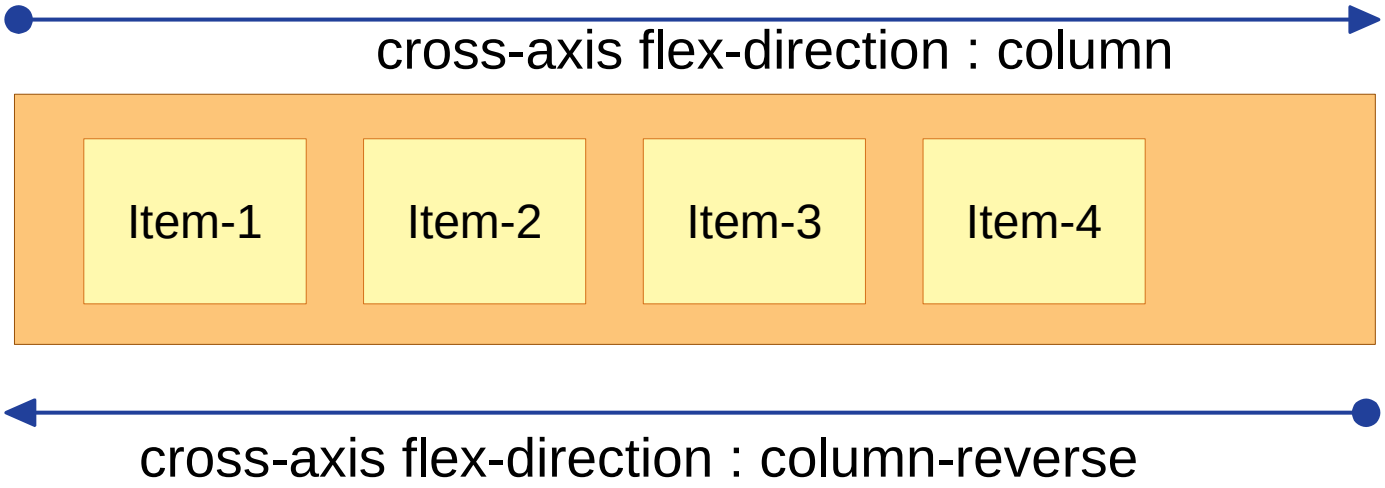
Flexbox flex-direction



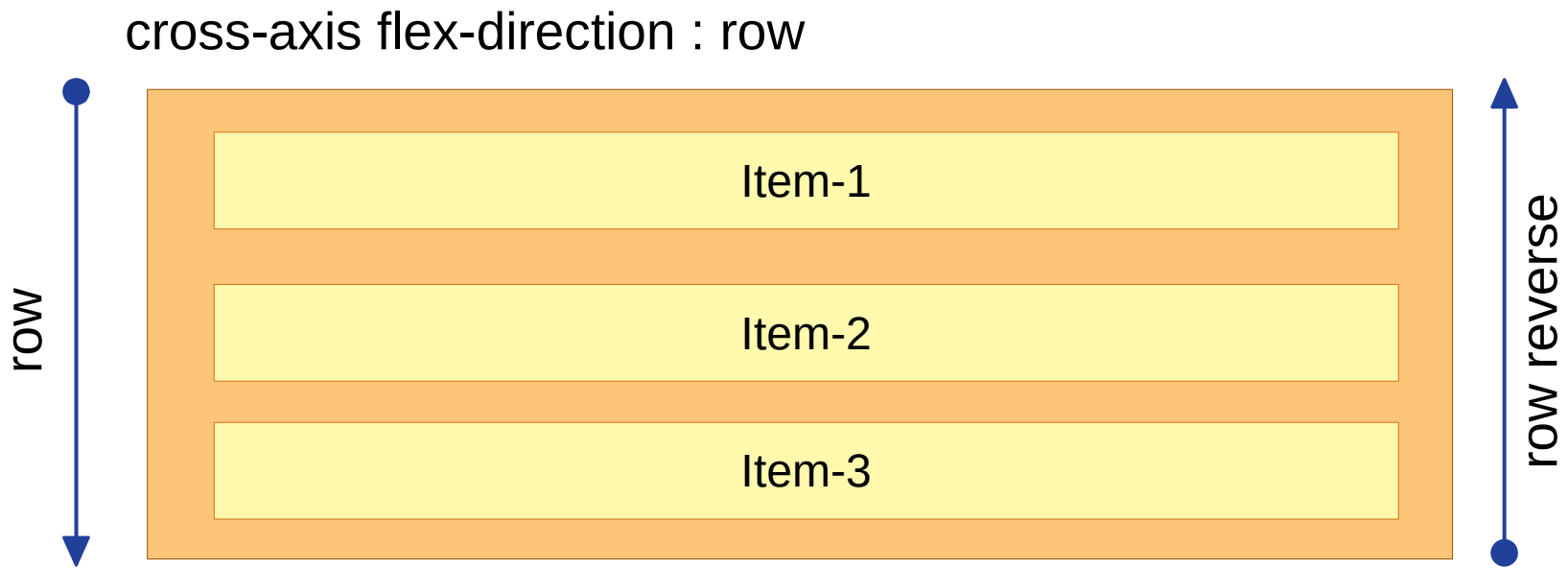
Flexbox flex-direction



Flexbox flex-direction



Flexbox flex-direction



Flexbox display

- The parent element becomes flexible by setting display property to flex

Syntax:

```
.container {
    display : flex; /* values – flex | inline-flex */
}
```

Flexbox example

- Items display in a row (the flex-direction property's default is row)
- Flex items start from the start edge of the main axis
- Flex items do not stretch on the main dimension, but can shrink
- Flex items will stretch to fill the size of the cross axis

Flexbox flex-wrap

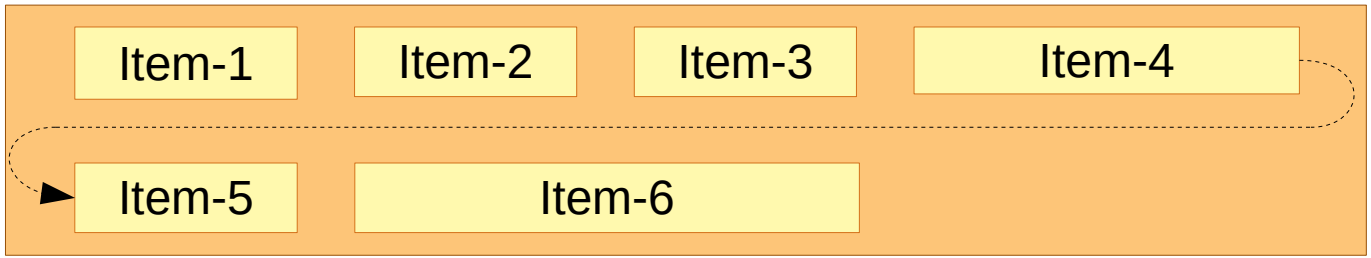
- The flex items can be wrapped onto multiple lines using flex-wrap property
- To cause wrapping behaviour add the property flex-wrap with a value of wrap
- Now, if flex-items be too large to all display in one line, they will wrap onto another line

Syntax:

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

Flexbox flex-wrap

- “**nowrap**” (default): all flex items will be in one line
- “**wrap**”: flex items will wrap onto multiple lines, from top to bottom.
- “**wrap-reverse**”: flex items will wrap onto multiple lines from bottom to top



Flexbox flex-flow

- The flex-flow property is shorthand of flex-direction and flex-wrap
- The first value specified is flex-direction and the second value is flex-wrap

Syntax:

```
.container {
    flex-flow : row wrap;
}
```

Flexbox justify-content

- This property defines alignment of flex items along main axis
- It helps distribute extra free space left over when either all the flex items on a line are inflexible, or are flexible but have reached their maximum size
- It also exerts some control over the alignment of items when they overflow the line

Flexbox justify-content

Syntax:

```
.container {
    justify-content: center;
}
```

Flexbox justify-content

Value	Description
flex-start	Items are placed w.r.t. start line
flex-end	Items are placed w.r.t. to end line
center	Items are centered along the line
space-between	Items are evenly distributed in the line; first item is on the start line, last item on the end line
space-around	Items are evenly distributed in the line with equal space around them
space-evenly	Items are distributed so that the spacing between any two items (and the space to the edges) is equal

Flexbox align-items

- This defines the alignment of flex items along cross axis on the current line
- This can be imagined as justify-content version for the cross-axis

Syntax:

```
.container {
    align-items: center;
}
```

Flexbox align-items

Value	Description
flex-start	Cross-start margin edge of the items is placed on the cross-start line
flex-end	Cross-end margin edge of the items is placed on the cross-end line
center	Items are centered in the cross-axis
baseline	Items are aligned such as their baselines align
stretch (default)	Stretch to fill the container (still respect min-width/max-width)

Flexbox align-content

- This aligns a flex container's lines within when there is extra space in the cross-axis, similar to how justify-content aligns individual items within the main-axis
- This property has no effect on single line flexible boxes (applies to multi-line flex containers)

Syntax:

```
.container {
    align-content: center;
}
```

Flexbox align-content

Value	Description
flex-start	Lines packed to the start of the container
flex-end	Lines packed to the end of the container
center	Lines packed to the center of the container
space-between	Lines evenly distributed; the first line is at the start of the container while the last one is at the end
space-around	Lines evenly distributed with equal space around each line
stretch (default)	Lines stretch to take up the remaining space

Flex item properties

- order
- flex-grow
- flex-shrink
- flex-basis

Flex item properties
(order)

- By default, flex items are laid out in the source order
- The order property controls the order in which they appear in the flex container

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

Flex item properties
(flex-grow)

- “flex-grow” defines the ability for a flex item to grow if necessary
- It accepts a unitless value that serves as a proportion
- It dictates what amount of available space inside flex container the item should take up

Flex item properties
(flex-grow)

- If all items have flex-grow set to 1, the remaining space in the container will be distributed equally to all children
- If one of the children has a value of 2, the remaining space would take up twice as much space as the others
- Negative numbers are invalid

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

Flex item properties
(flex-shrink)

- This defines the ability for a flex item to shrink if necessary
- Negative numbers are invalid

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

Flex item properties
(flex-basis)

- The flex-basis property specifies the initial length of a flexible item
- An absolute <length>, a <percentage> of the parent flex container's main size property, or the keyword auto

```
.flex-item {
  flex-basis: number | auto | initial | inherit; /* default auto */
}
```

Flex item properties
(flex-basis)

- It can be a number (e.g. 20%, 10rem, 200px etc.)
- If set to 0, the extra space around content isn't factored in
- If set to auto, the length is equal to the length of the flexible item
- If the item has no length specified, the length will be according to its content
- Negative values are invalid

Flex item properties
(flex-basis)

Example :

```
.flex-item {
    /* Specify <'width'> */
    flex-basis: 50%;
    flex-basis: 200px;
    flex-basis: 10rem; /* root em */
    flex-basis: auto;
}
```

1em or 1rem equals the font size of the html element (which for most browsers has a default value of 16px)

Flex item properties
(flex)

- This is the shorthand for flex-grow, flex-shrink and flex-basis combined
- The second and third parameters (flex-shrink and flex-basis) are optional (Default is 0 1 auto)
- It is recommended that you use this shorthand property rather than set the individual properties
- The short hand sets the other values intelligently

Flex item properties
(flex)

Syntax :

```
.flex-item {
    flex: flex-grow flex-shrink flex-basis;
}
```

Example :

```
.flex-item {
    flex: 0 1 auto;
}
```

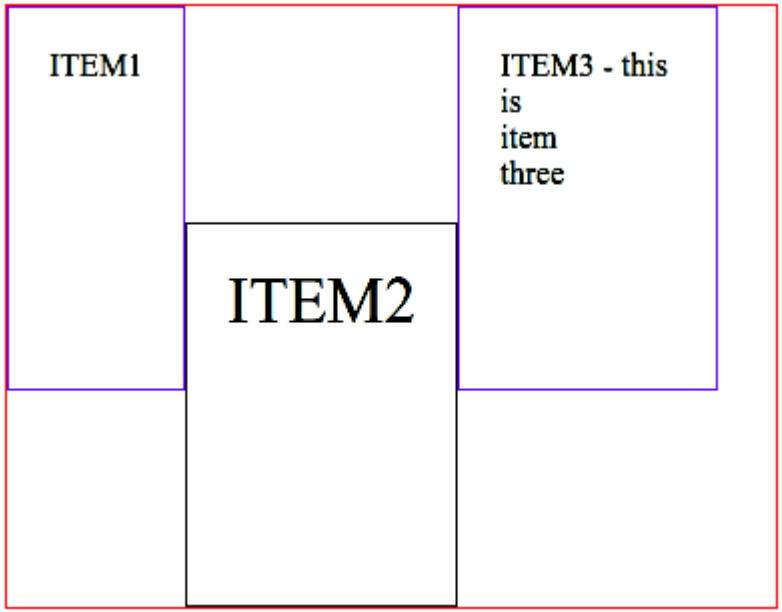
Flex item properties
(align-self)

- This allows the default alignment (or the one specified by align-items) to be overridden for individual flex items

```
.flex-item {
  align-self: auto | flex-start | flex-end | center | baseline | stretch;
}
```

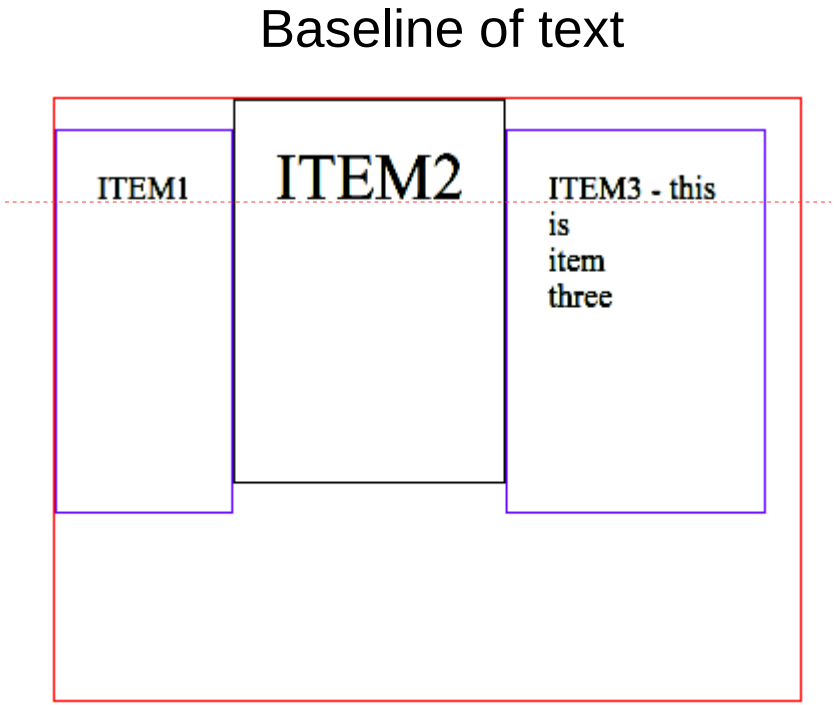
Flex item properties
(align-self)

```
.flex-item {
  align-self: flex-end;
}
```



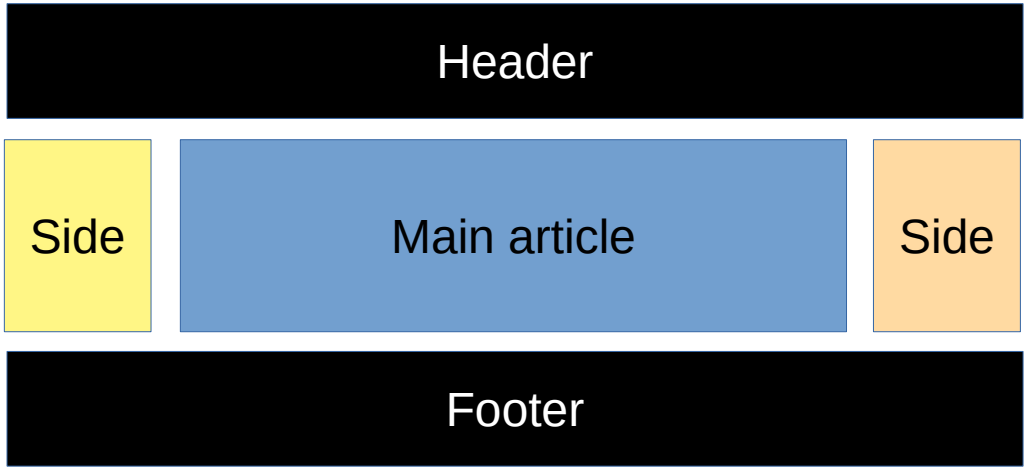
Flex item properties
(align-self)

```
.flex-item {
  align-self: baseline;
}
```



Class work

- Design following layout using flexbox properties
- Use media queries to alter the layout for mobile and desktop



WSA | Forward looking IT finishing school

Web Stack Academy (P) Ltd

#83, Farah Towers,

1st floor,MG Road,

Bangalore – 560001

M: +91-80-4128 9576

T: +91-98862 69112

E: info@www.webstackacademy.com