CSS 3 -- Flexbox

Prepared By: Web Team

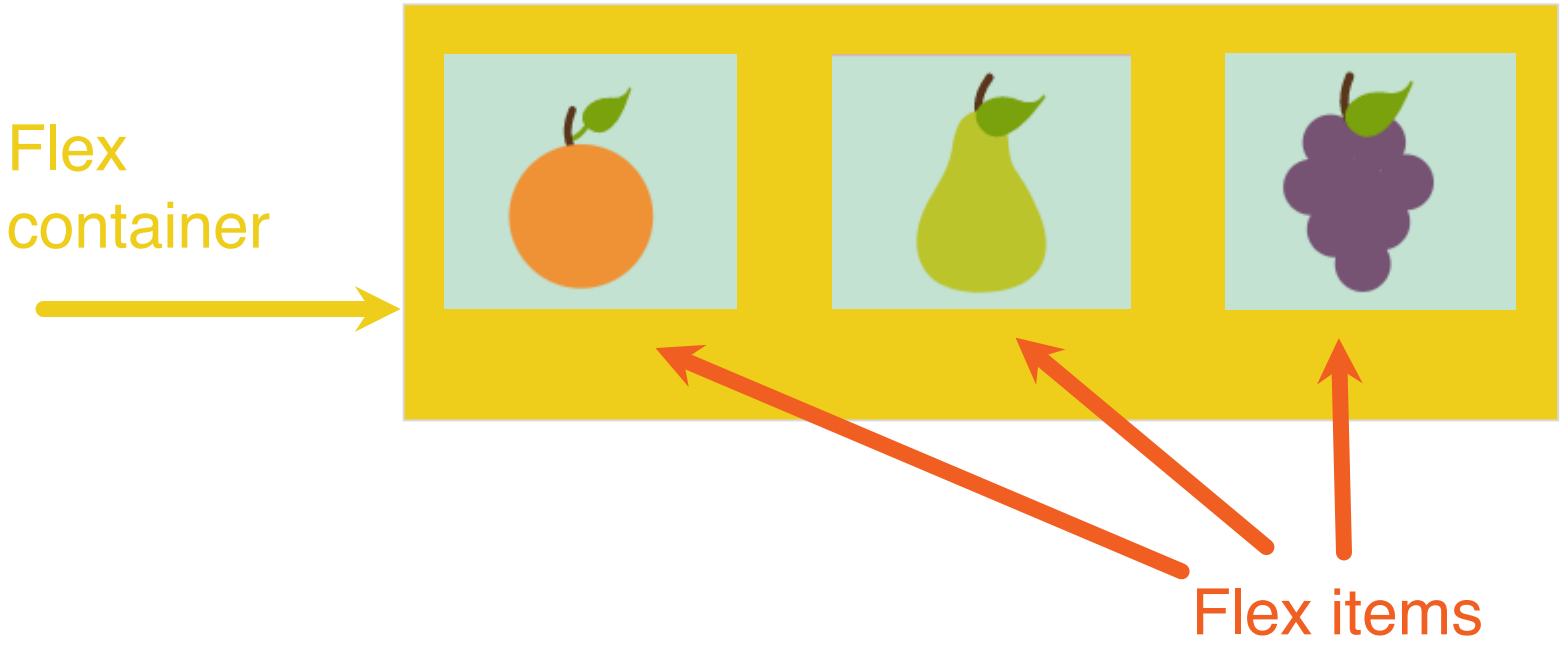
Background

- **Flexbox** a new form of layout that is included in CSS3 to help you to create responsive content without the use of **Float** and **Table layout** in CSS.
- It comes with many default properties which make it very easy to create complex layout with only a few lines of code.
- Flexbox has become more stable and quite popular among developers these days.

History

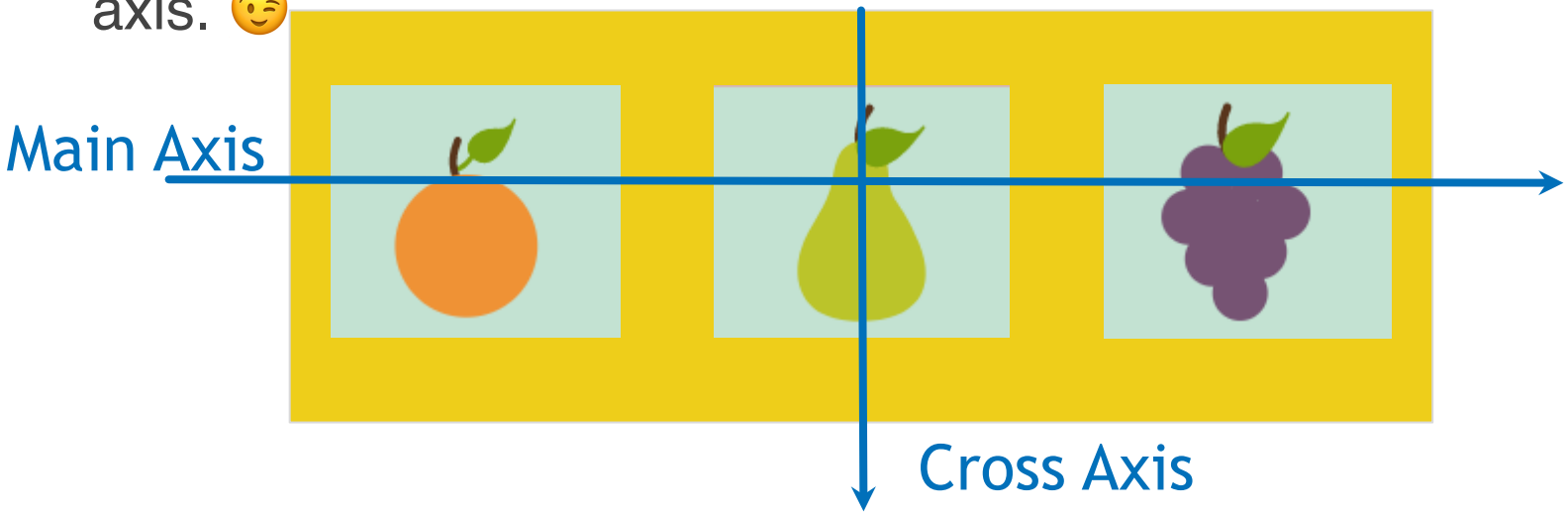
- **Flexbox** was final released in 2010s by **Mr. Tab Atkin Jr**.
- There has been 3 stages of flexbox release:
  - The **old 2009 spec** -- display: box (outdated version)
  - The **2011 “tweener” spec** -- display: flexbox (was created to implement on IE10)
  - The **final 2012 spec** -- display: flex (The last spec which is supported on most modern browsers.)

Basic & Terminology



Basic & Terminology

- A Flexbox layout consists of a Flex Container that holds Flex Items. The flex container can be laid out horizontally or vertically based on what it's referred which one is the main axis. 😉



Flex layout properties

- **display**: This defines a flex container; **inline** or **block** depending on the given value. It enables a flex context for all its direct children.

Example:

```
.container {
  display: flex;  /* or inline-flex */
}
```

Flex Container properties

- **The flex container properties are :**
  - **flex-direction**
  - **flex-wrap**
  - **flex-flow**
  - **justify-content**
  - **align-items**
  - **align-content**

Flex Container properties

- **flex-direction**: This establishes the **main-axis**, thus defining the direction flex items are placed in the flex container.
- Flexbox is (aside from optional wrapping) **a single-direction layout concept**. Think of flex items as primarily laying out either in horizontal rows or vertical columns.

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

Flex Container properties

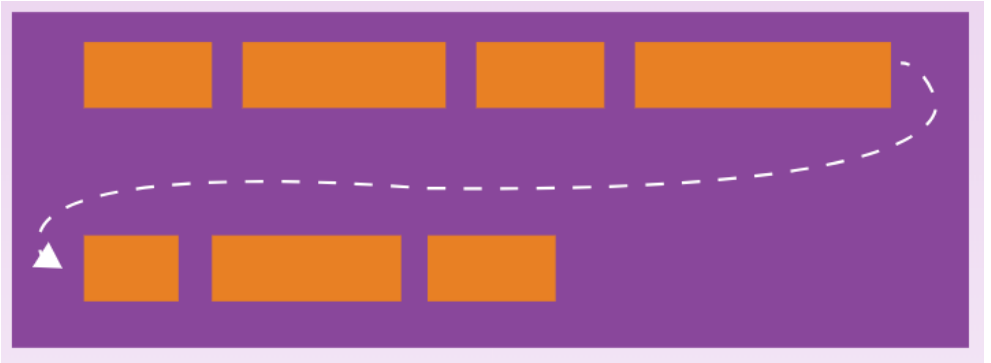
Definition:

- **row** (default): left to right in ltr ; right to left in rtl ;
- **row-reverse**: right to left in ltr ; left to right in rtl ;
- **column**: same as row but top to bottom
- **column-reverse**: same as row-reverse but bottom to top.

Flex Container properties

- **flex-wrap**: By default, **flex items** will all try to fit onto one line. You can change that and allow the items to wrap as needed with this property.

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



Flex Container properties

Definition:

- **nowrap** (default): all flex items will be on 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.

Flex Container properties

- **flex-flow** (Applies to: parent flex container element): This is a shorthand of **flex-direction** and **flex-wrap** properties, which together define the flex container's main and cross axis. Default is **row nowrap**.

flex-flow: <‘flex-direction’> || <‘flex-wrap’>

Flex Container properties

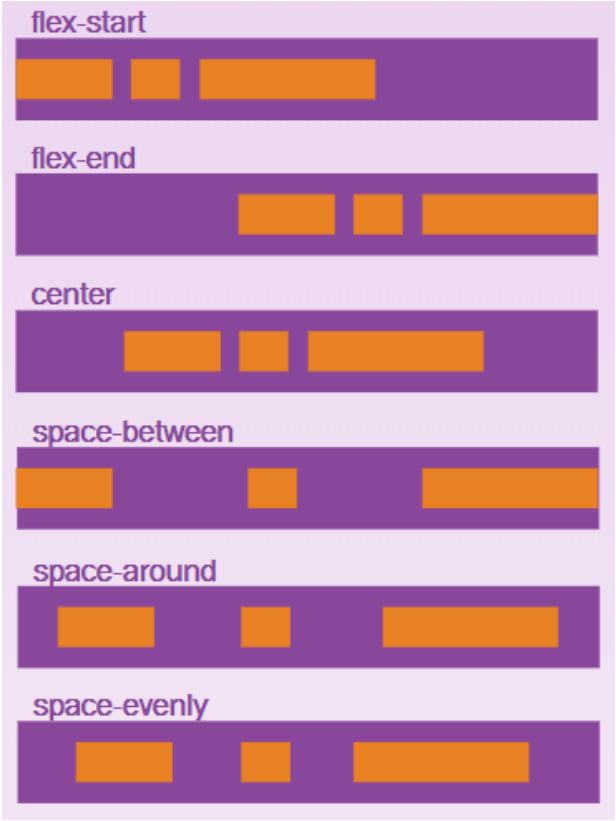
- **justify-content:** This defines the alignment along the main axis. It helps distribute extra free space left over when either all the flex items on a line are inflexible, or are flexible but have reached their maximum size.

Flex Container properties

- It also exerts some control over the alignment of items when they overflow the line.

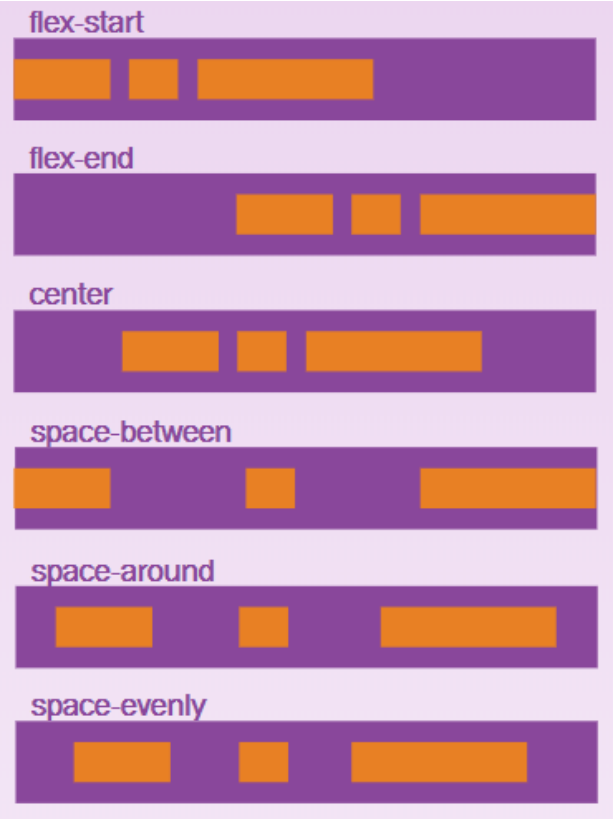
```
.container {
  justify-content: flex-start | flex-end | center | space-between | space-around | space-evenly
}
```

Flex Container properties



- **flex-start** (default): items are packed toward the start line
- **flex-end**: items are packed toward 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

Flex Container properties



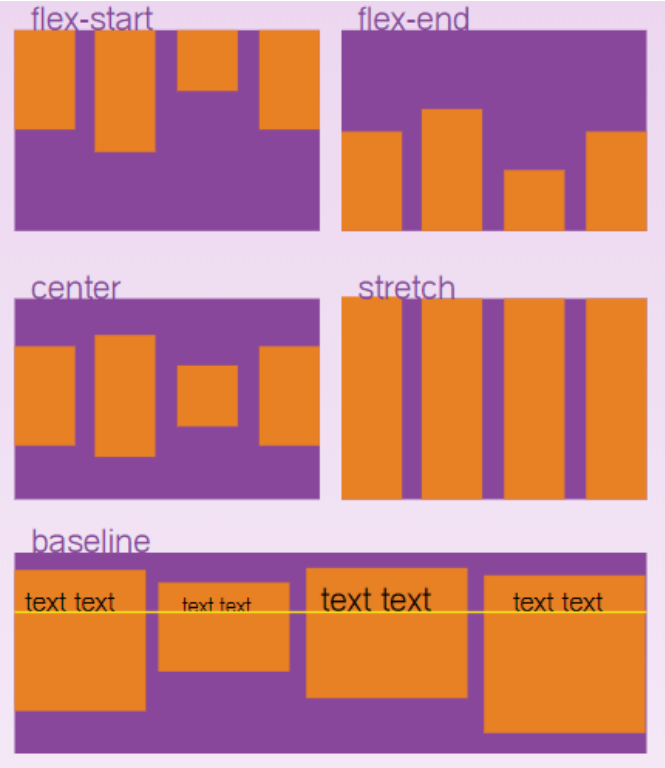
- **space-around**: items are evenly distributed in the line with equal space around them. Note that visually the spaces aren't equal, since all the items have equal space on both sides. The first item will have one unit of space against the container edge, but two units of space between the next item because that next item has its own spacing that applies.
- **space-evenly**: items are distributed so that the spacing between any two items (and the

Flex Container properties

- **align-items:** This defines the default behavior for how flex items are laid out along the cross axis on the current line. Think of it as the **justify-content** version for the cross-axis (perpendicular to the main-axis).

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

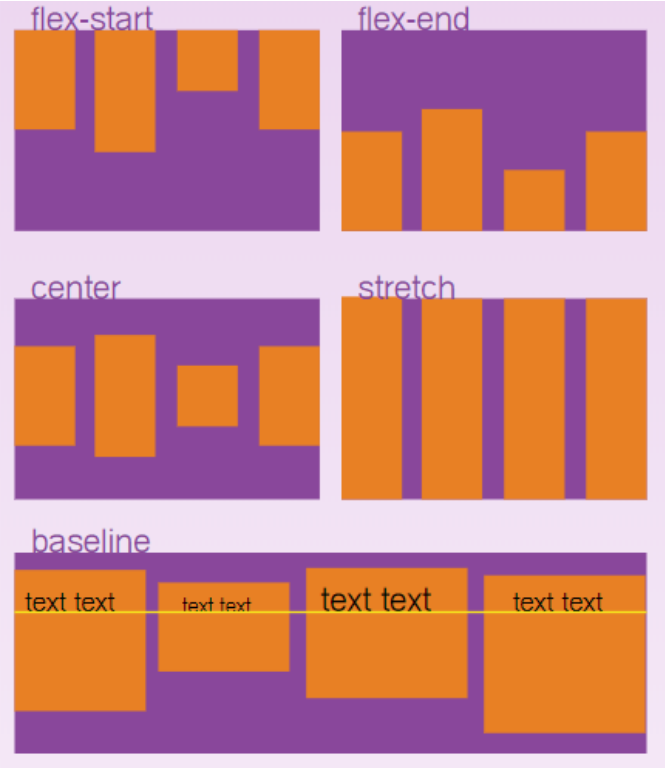
Flex Container properties



Definition:

- **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

Flex Container properties



Definition:

- **baseline**: items are aligned such as their baselines align.
- **stretch** (default): stretch to fill the container (still respect min-width/max-width)

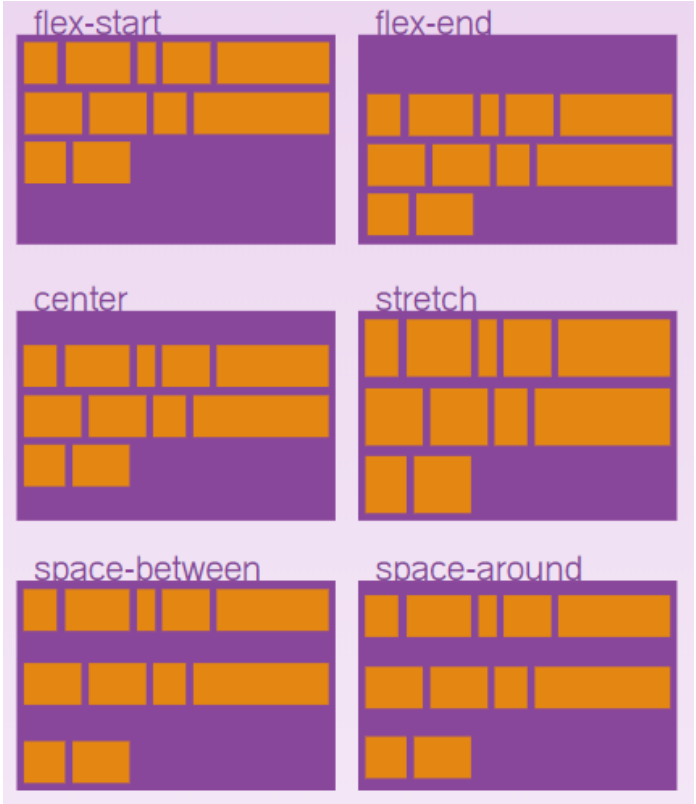
Flex Container properties

- **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.

- ***Note:*** *this property has no effect when there is only one line of flex items.*

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

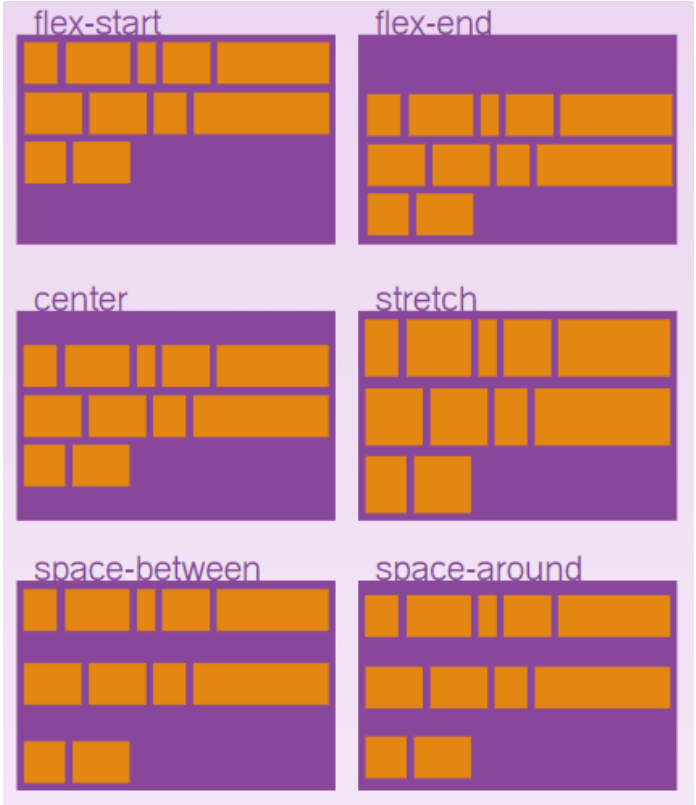
Flex Container properties



Definition:

- **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

Flex Container properties



Definition:

- **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

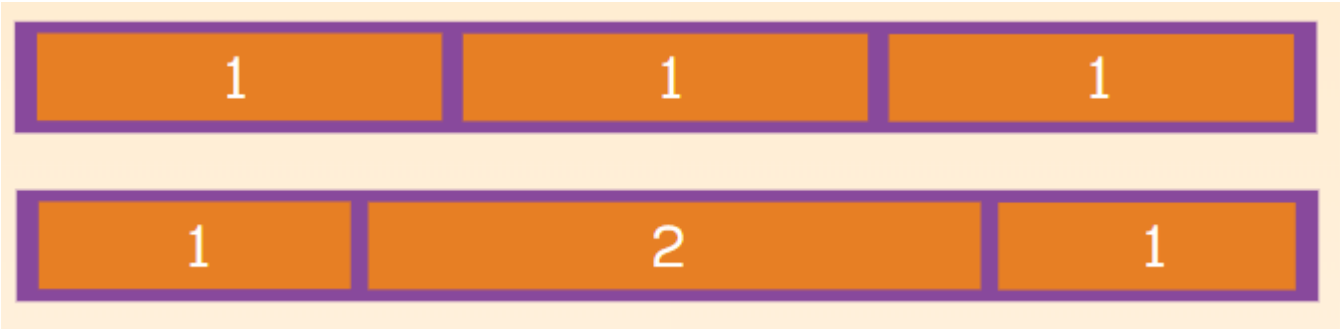
- **The flex item properties are :**
  - order
  - flex-grow
  - flex-shrink
  - flex-basis
  - flex
  - align-self

Flex Item properties

- **flex-grow:** This 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 the available space inside the flex container the item should take up.

- 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 (or it will try to, at least).

Flex Item properties




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

** Note: Negative numbers are invalid;*

Flex Item properties

- **flex-shrink:** This defines the ability for a flex item to shrink if necessary.

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

**Note: Negative numbers are invalid;*

Flex Item properties

- **flex-basis:** This defines the default size of an element before the remaining space is distributed.
- It can be a length (e.g. 20%, 5rem, etc.) or a keyword. The **auto** keyword means "look at my width or height property" (which was temporarily done by the **main-size** keyword until deprecated).

Flex Item properties

- The **content** keyword means "size it based on the item's content" - this keyword isn't well supported yet, so it's hard to test and harder to know what its brethren **max-content**, **min-content**, and **fit-content** do.

```
.item {
  flex-basis: <length> | auto  /* default is auto */ }
```

Flex Item properties

If set to **0, the extra space around content isn't factored in. If set to **auto**, the extra space is distributed based on its **flex-grow** value.*

Flex Item properties

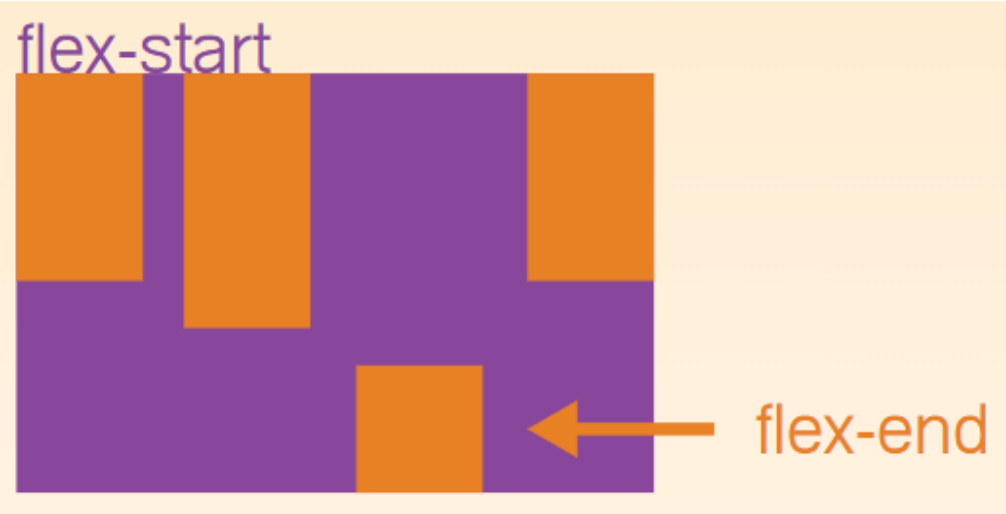
- **flex:** This is the shorthand for **flex-grow**, **flex-shrink** and **flex-basis** combined. The second and third parameters (**flex-shrink** and **flex-basis**) are optional. Default is **0 1 auto**.

```
.item {
  flex: none | [ <'flex-grow'> <'flex-shrink'>? || <'flex-basis'> ]
}
```

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

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



Flex Item properties

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

Thank You