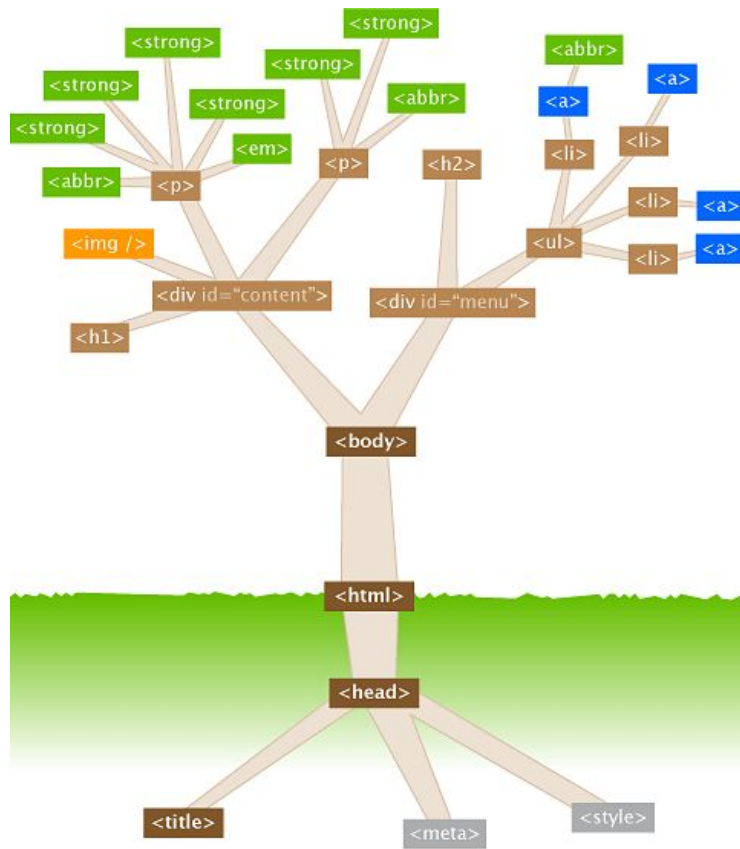


# CSS – FlexBox

A new way of composing layout elements



Dom Tree

# Agenda

- Flexbox
  - Layouting : Flexbox
  - Flexbox Container Properties
  - Flexbox Item Properties
  - Flexbox Support
- Questions

# Learning Objectives

- Why a clean layouting algorithm is so important?
- What is the different in the flexbox's way?
- Know about the concepts of the flexbox
- Be able to apply the flexbox to your design

# Flexbox

The new kid on the block

- Flexbox is ... ?
  - CSS Flexible Box Layout Module
- Css for Layout
- One-dimensional layouting
  - Either Column or Row, adjusting the space for elements in only one direction
- Two-dimensional layouting
  - Nesting of these boxes (horizontal inside vertical, or vertical inside horizontal) can be used to build layouts in two dimensions.
- Many settings possible
  - Compared to
    - Box Model + Float/Clear Model
- W3C Candidate Recommendation
  - <https://www.w3.org/TR/css-flexbox-1/>

# Why FlexBox?

Here Flexbox offers a solution that is flexible compared to the block layout of the box model.

With Flexbox ...

- the flow direction can be freely determined
- elements can be arranged within one line or in several lines by using breaks
- elements are given flexible sizes that are aligned with the viewport
- can be used to position elements in a different order in the markup.











If the width of the viewport is no longer sufficient for a horizontal multi-column layout, media queries can be used to change the direction of the layout to a vertical layout.


# Flexbox Support

## CSS Flexible Box Layout Module - CR

Method of positioning elements in horizontal or vertical stacks.  
Support includes all properties prefixed with `flex`, as well as `display: flex`, `display: inline-flex`, `align-content`, `align-items`, `align-self`, `justify-content` and `order`.

Current aligned Usage relative Date relative Filtered All 

IE	Edge <sup>*</sup>	Firefox	Chrome	Safari	Opera	Safari on iOS <sup>*</sup>	Opera Mini <sup>*</sup>	Android Browser <sup>*</sup>	Opera Mobile <sup>*</sup>	Chrome for Android	Firefox for Android	UC Browser for Android	Samsung Internet	QQ Browser	Baidu Browser	KaiOS Browser
					10-11.5											
		<sup>1</sup> 2-21 	<sup>1</sup> 4-20 	<sup>1</sup> 3.1-6 	12.1	<sup>1</sup> 3.2-6.1 										
6-9		<sup>3</sup> 22-27	21-28 	6.1-8 	15-16 	7-8.4 		<sup>1</sup> 2.1-4.3 	12							
<sup>2,4</sup> 10 	12-93	28-91	29-93	9-14.1	17-78	9-14.7		4.4-4.4.4	12.1				4-14.0			
<sup>4</sup> 11	94	92	94	15	79	15	all	94	64	94	92	12.12	15.0	10.4	7.12	2.5
		93-94	95-97	TP												

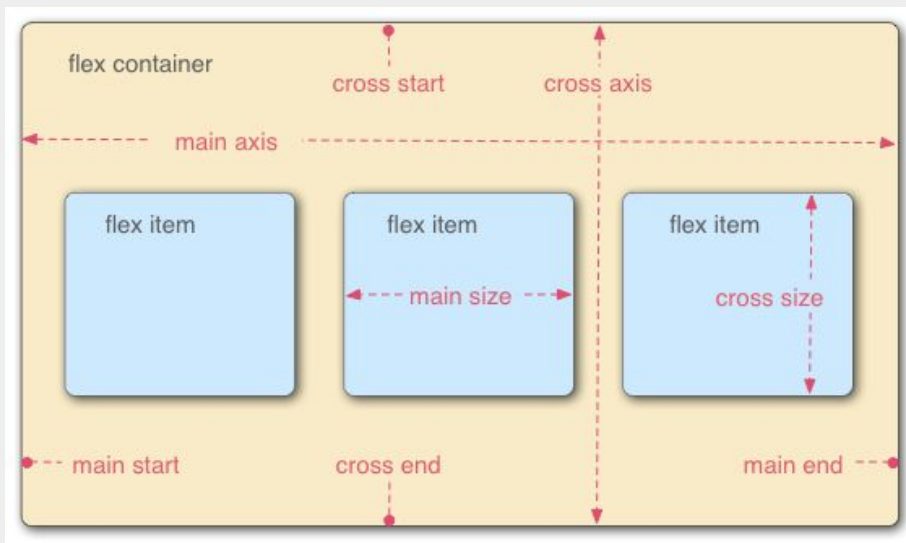
Usage % of all users  ?  
Global 98.14% + 1.29% = 99.43%  
unprefixed: 98% + 0.79% = 98.8%

# FlexBox

The main concepts.

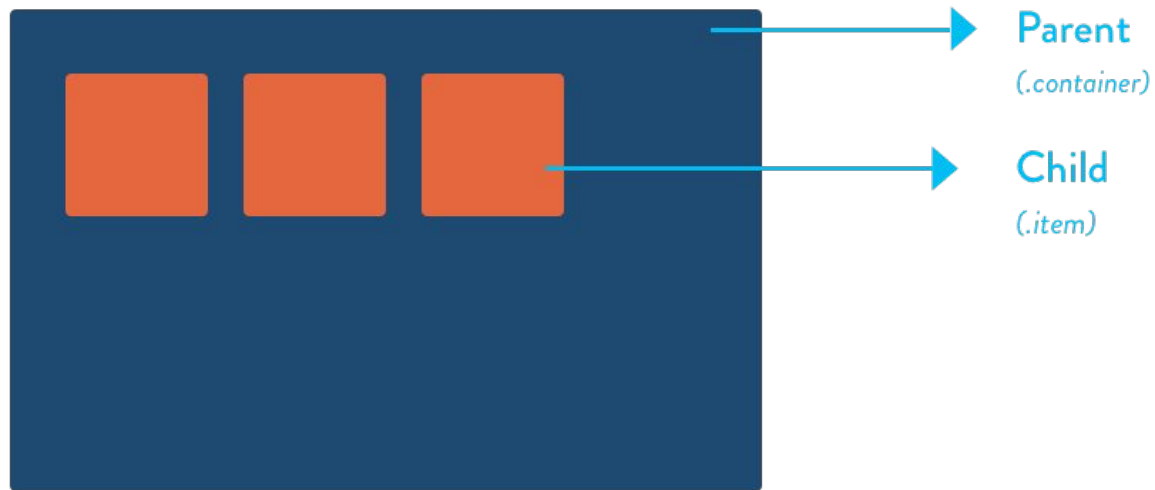
Learn them before you go in deeply!

- Container und Items
  - Or: parent element and child elements
- [https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS\\_layout/Flexbox](https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS_layout/Flexbox)
- With **display:flex**

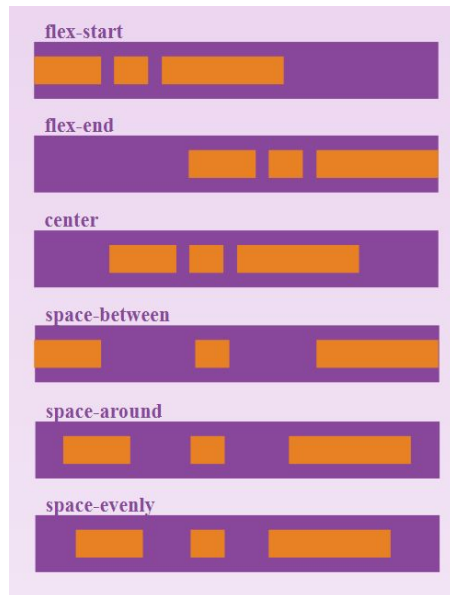
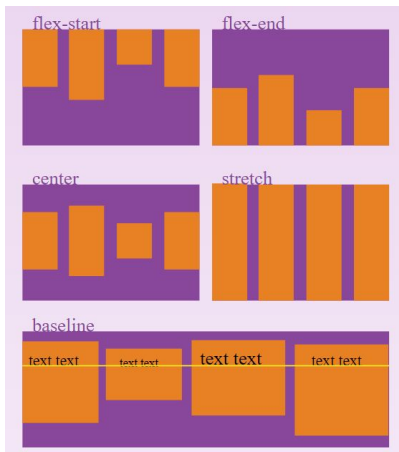
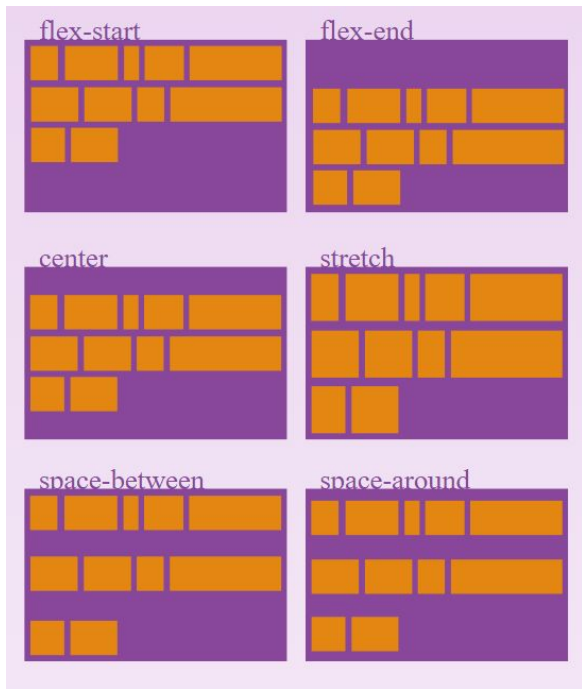
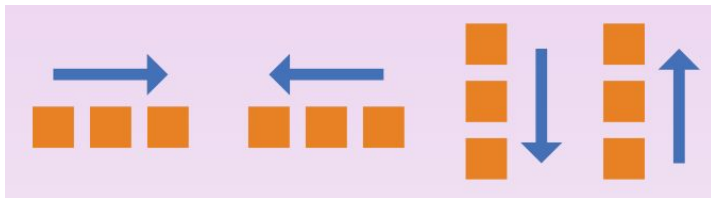
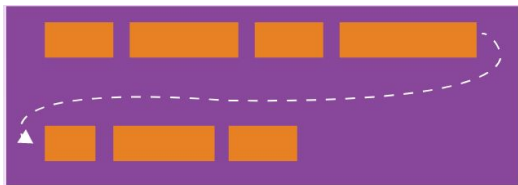




# CSS Flexbox



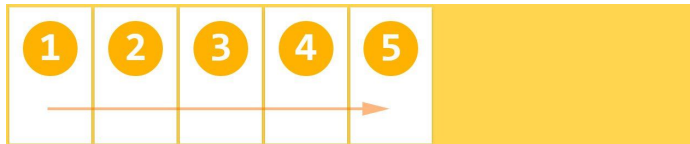
flexbox parent, child



# **flex container**

the parent

# Flexbox Container Properties



- **flex-direction**

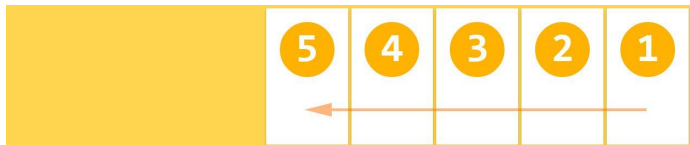
- With **row** direction the flex items are stacked in a row from left-to-right in **ltr** context

```
.flex-container {  
  display: flex;  
  flex-direction: row;  
}
```

<https://scotch.io/tutorials/a-visual-guide-to-css3-flexbox-properties>

Default value: **row**

# Flexbox Container Properties



- **flex-direction**

- With **row-reverse** direction the flex items are stacked in a row from right-to-left in **ltr** context

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

# Flexbox Container Properties



- **flex-direction**

- With **column** direction the flex items are stacked in a column from top-to-bottom

```
.flex-container {  
  display: flex;  
  flex-direction: column;  
}
```

# Flexbox Container Properties



- **flex-direction**

- With **column-reverse** direction the flex items are stacked in a column from bottom-to-top

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

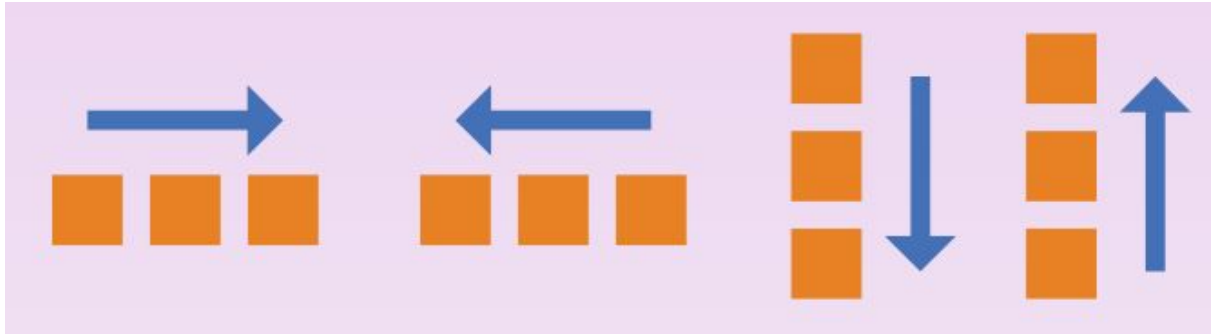
# Flexbox Container Properties

- flex-direction

Values:

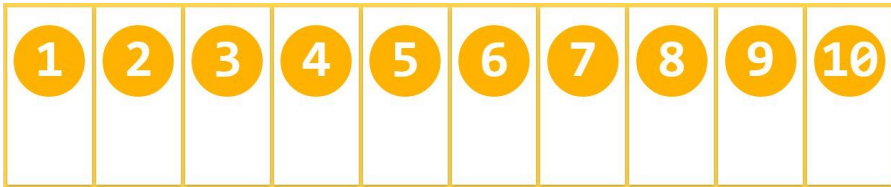
- ☐ `row`
- ☐ `row-reverse`
- ☐ `column`
- ☐ `column-reverse`





flex-directions

# Flexbox Container Properties

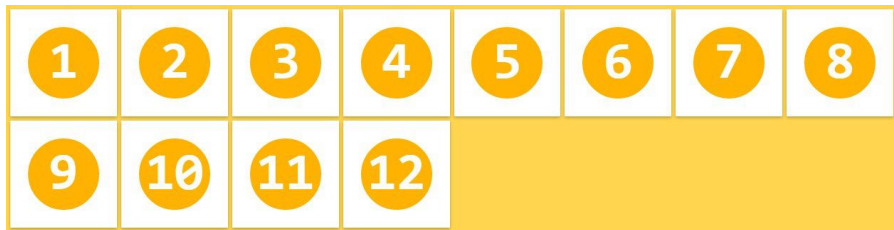


- **flex-wrap**

- Flex items are displayed in one row, by default they are shrunk to fit the flex container's width

```
.flex-container {  
  display: flex;  
  flex-wrap: nowrap;  
}
```

# Flexbox Container Properties



- **flex-wrap**
    - Flex items are displayed in multiple rows if needed from left-to-right and top-to-bottom
- ```
.flex-container {  
  display: flex;  
  flex-wrap: wrap;  
}
```

# Flexbox Container Properties



- **flex-wrap**

- Flex items are displayed in multiple rows if needed from left-to-right and top-to-bottom

```
.flex-container {  
  display: flex;  
  flex-wrap: wrap-reverse;  
}
```

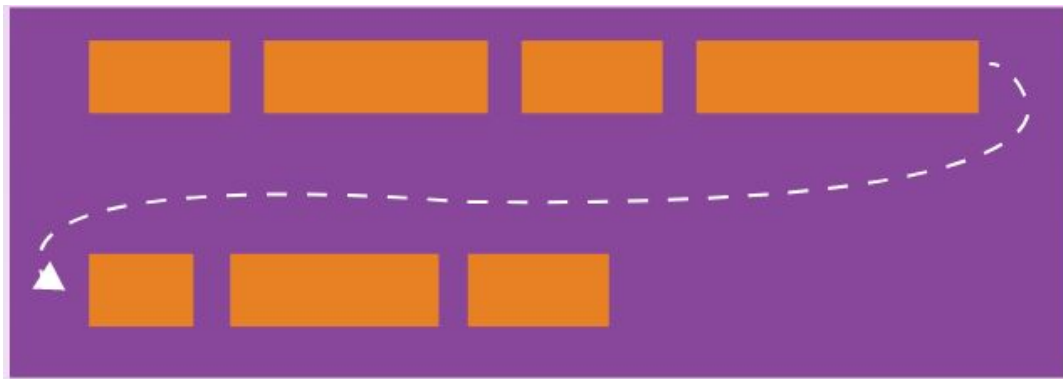
Default value: nowrap

# Flexbox Container Properties

- flex-wrap

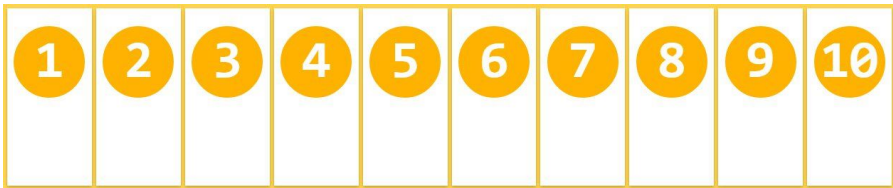
Values:

- ☐ nowrap
- ☐ wrap
- ☐ wrap-reverse



flex-wrap

# Flexbox Container Properties



- **flex-flow**

- This property is a shorthand for setting the **flex-direction** and **flex-wrap** properties.

```
.flex-container {  
  display: flex;  
  flex-flow: row nowrap;  
}
```

# Flexbox Container Properties



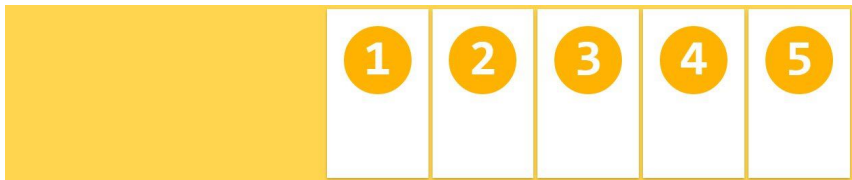
- **justify-content**

- The **justify-content** property aligns flex items along the main axis of the current line of the flex container. It helps distribute left free space when either all the flex items on a line are inflexible, or are flexible but have reached their maximum size.
- **flex-start** Flex items are aligned to the left side of the flex container in **ltr** context

```
.flex-container {  
  display: flex;  
  justify-content: flex-start;  
}
```



# Flexbox Container Properties

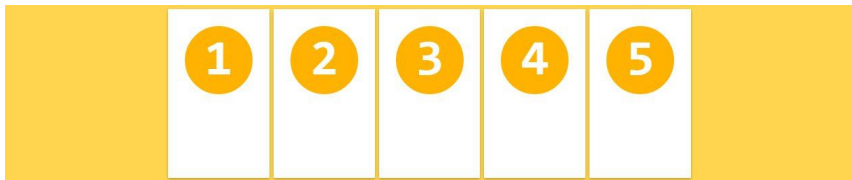


- **justify-content**

- **flex-end** Flex items are aligned to the right side of the flex container in **ltr** context

```
.flex-container {  
  display: flex;  
  justify-content: flex-end;  
}
```

# Flexbox Container Properties



- **justify-content**
  - **center** Flex items are aligned at the center of the flex container

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

# Flexbox Container Properties

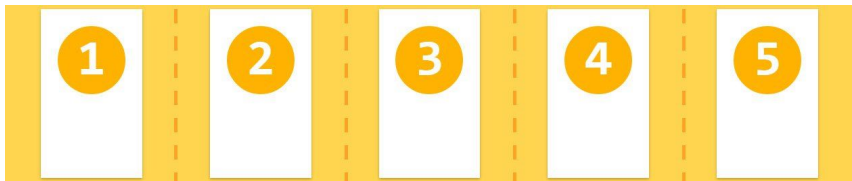


- **justify-content**

- **space-between** Flex items are displayed with equal spacing between them, first and last flex items are aligned to the edges of the flex container

```
.flex-container {  
  display: flex;  
  justify-content:  
    space-between;  
}
```

# Flexbox Container Properties



- **justify-content**

- **space-around** Flex items are displayed with equal spacing around every flex item, even the first and last flex items

```
.flex-container {  
  display: flex;  
  justify-content: space-around;  
}
```

# Flexbox Container Properties

- **justify-content**

Values:

- *flex-start*
- *flex-end*
- *center*
- *space-between*
- *space-around*
- *space-evenly*

flex-start



flex-end



center



space-between



space-around

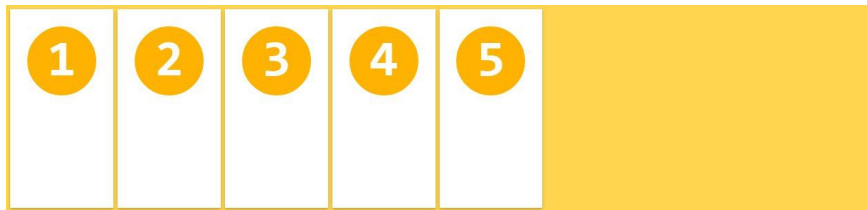


space-evenly



justify-content

# Flexbox Container Properties

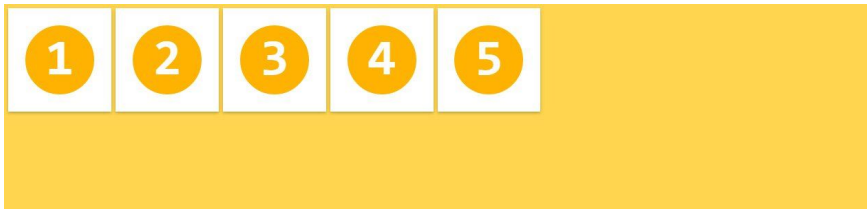


- **align-items**

- Flex items can be aligned in the cross axis of the current line of the flex container, similar to **justify-content** but in the perpendicular direction. This property sets the default alignment for all flex items, including the anonymous ones.
- **stretch** Flex items fill the whole height (or width) from **cross start** to **cross end** of the flex container

```
.flex-container {  
  display: flex;  
  align-items: stretch;  
}
```

# Flexbox Container Properties



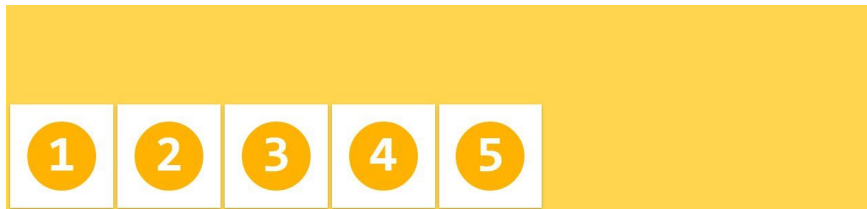
- **align-items**

- **flex-start** Flex items are stacked to the **cross start** of the flex container

```
.flex-container {  
  display: flex;  
  align-item: flex-start;  
}
```



# Flexbox Container Properties



- **align-items**

- **flex-end** Flex items are stacked to the **cross end** of the flex container

```
.flex-container {  
  display: flex;  
  align-items: flex-end;  
}
```

# Flexbox Container Properties



- **align-items**

- **center** Flex items are stacked to the center of the **cross axis** of the flex container

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

# Flexbox Container Properties



- **align-items**

- **baseline** Flex items are aligned in a way that their baselines are aligned

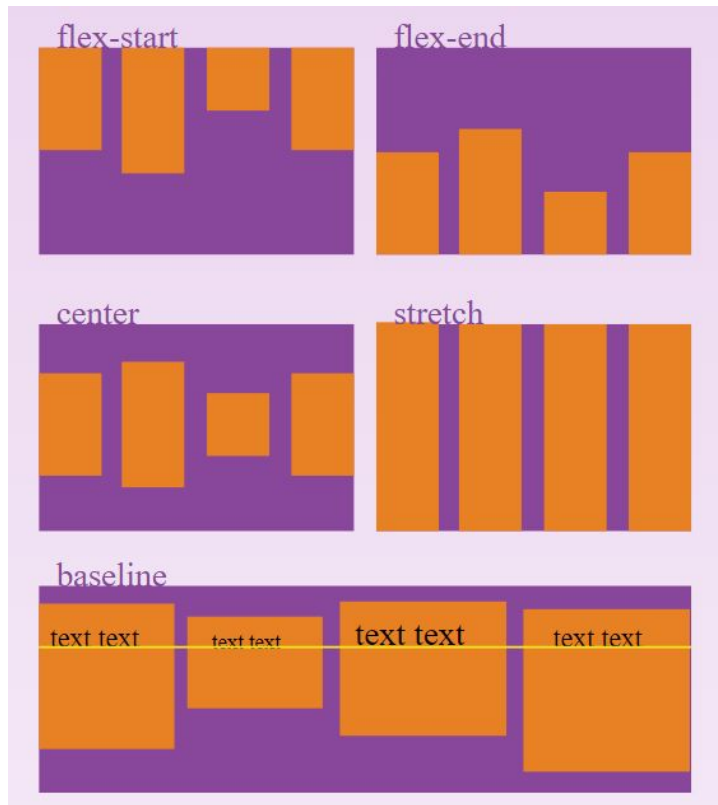
```
.flex-container {  
  display: flex;  
  align-items: baseline;  
}
```

# Flexbox Container Properties

- align-items

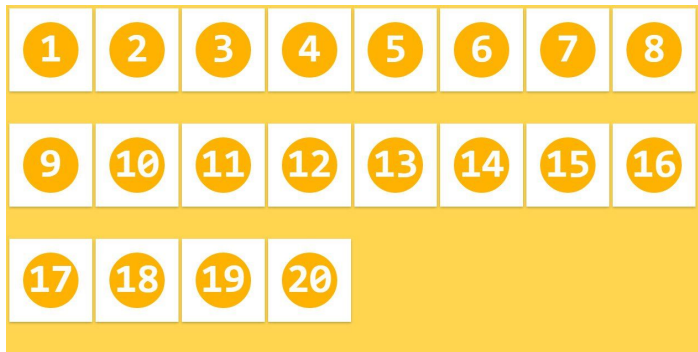
Values:

- ☐ *stretch*
- ☐ *flex-start*
- ☐ *flex-end*
- ☐ *center*
- ☐ *baseline*



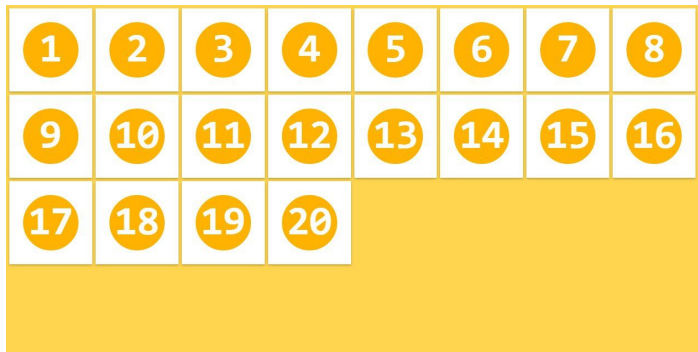
align-items

# Flexbox Container Properties



- **align-content (multiple-line)**
    - The **align-content** property aligns a flex container's lines within the flex container when there is extra space in the cross-axis, similar to how **justify-content** aligns individual items within the main-axis.
    - **stretch** Flex items are displayed with distributed space after every row of flex items
- ```
.flex-container {  
  display: flex;  
  align-content: stretch;  
}
```

# Flexbox Container Properties



- **align-content**

- **flex-start** Flex items are stacked toward the **cross start** of the flex container

```
.flex-container {  
  display: flex;  
  align-content: flex-start;  
}
```

# Flexbox Container Properties

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20				

- **align-content**

- **flex-end** Flex items are stacked toward the **cross end** of the flex container

```
.flex-container {  
  display: flex;  
  align-content: flex-end;  
}
```



# Flexbox Container Properties

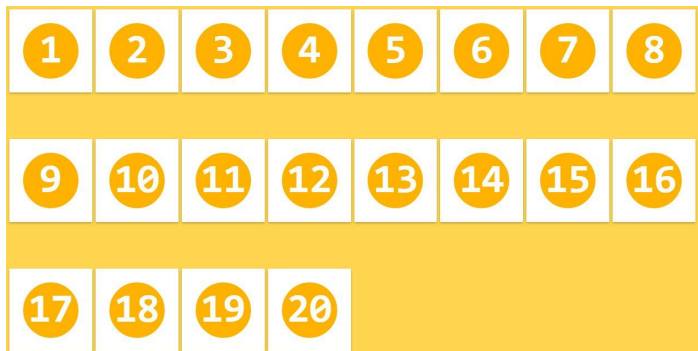
1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20				

- **align-content**

- **center** Rows of flex items are stacked in the center of the **cross axis** of the flex container

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

# Flexbox Container Properties

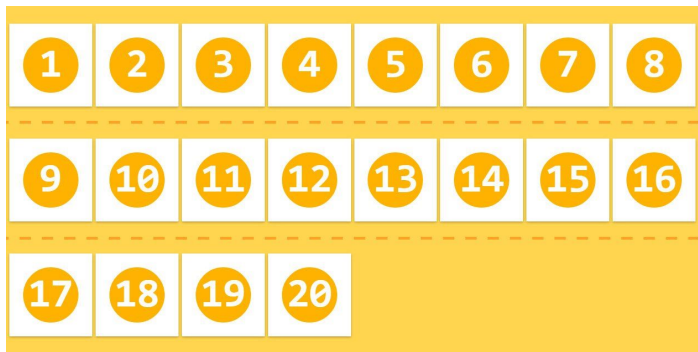


- **align-content**

- **space-between** Rows of flex items are displayed with equal spacing between them, first and last rows are aligned to the edges of the flex container

```
.flex-container {  
  display: flex;  
  align-content: space-between;  
}
```

# Flexbox Container Properties



- **align-content**

- **space-around** Flex items are displayed with equal spacing around every row of flex items.

```
.flex-container {  
  display: flex;  
  align-content: space-around;  
}
```

# Flexbox Container Properties

- align-content

Values:

- flex-start
- flex-end
- center
- space-between
- space-around
- space-evenly

flex-start



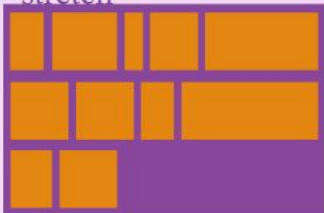
flex-end



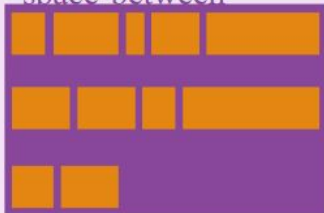
center



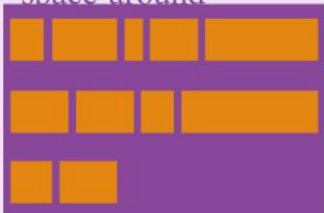
stretch



space-between



space-around



align-content

# **flex item**

child

# Flexbox Item Properties



- **order**

- **order** The order property controls the order in which children of a flex container appear inside the flex container. By default they are ordered as initially added in the flex container.
- Flex items can be reordered with this simple property, without restructuring the HTML code

```
.flex-item {  
  order: 4;  
}
```

# Flexbox Item Properties



- **flex-grow**

- This property specifies the flex grow factor, which determines how much the flex item will grow relative to the rest of the flex items in the flex container when positive free space is distributed.

If all flex items have same value for **flex-grow** then all items have same size in the container

```
.flex-item {  
  flex-grow: 1;  
}
```



# Flexbox Item Properties



- **flex-grow**

- The second flex item takes up more space relative to the size of the other flex items

```
.flex-item {  
  flex-grow: 3;  
}
```

# Flexbox Item Properties



- **flex-shrink**

- The **flex-shrink** specifies the flex shrink factor, which determines how much the flex item will shrink relative to the rest of the flex items in the flex container when negative free space is distributed.

By default all flex items can be shrunk, but if we set it to 0 (don't shrink) they will maintain the original size

```
.flex-item {  
  flex-shrink: 0;  
}
```

# Flexbox Item Properties



- **flex-basis**

- This property takes the same values as the **width** and **height** properties, and specifies the initial main size of the flex item, before free space is distributed according to the flex factors.

**flex-basis** is specified for the 4th flex item and dictates the initial size of the element

```
.flex-item {  
  flex-basis: auto;  
}
```

# Flexbox Item Properties

- **flex**

- This property is the shorthand for the **flex-grow**, **flex-shrink** and **flex-basis** properties. Among other values it also can be set to **auto** (**1 1 auto**) and **none** (**0 0 auto**).

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

Default value: 0 1 auto;

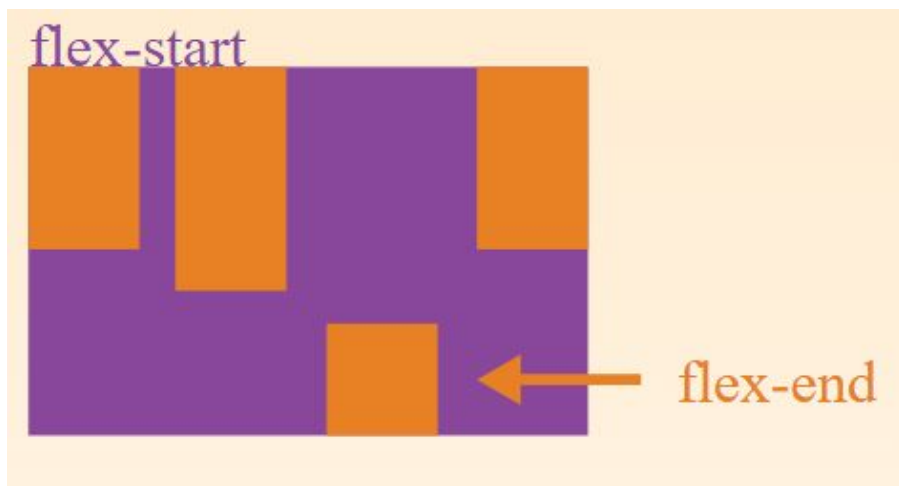
# Flexbox Item Properties



- **align-self**

- This **align-self** property allows the default alignment (or the one specified by **align-items**) to be overridden for individual flex items. Refer to **align-items** explanation for [flex container](#) to understand the available values.

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



align-self

# Flexbox

## Links & Pointers

- Theorie
  - <https://scotch.io/tutorials/a-visual-guide-to-css3-flexbox-properties#align-items>
  - [https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS\\_layout/Flexbox](https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS_layout/Flexbox)
  - <https://css-tricks.com/snippets/css/a-guide-to-flexbox/>
  - <https://caniuse.com/#search=flex>
- Exercise
  - <https://github.com/Zmote/it-club-css-layouts>
  - <http://flexboxfroggy.com/>
- More about the FlexBox (self-study)
  - <https://css-tricks.com/snippets/css/a-guide-to-flexbox/>

# Questions