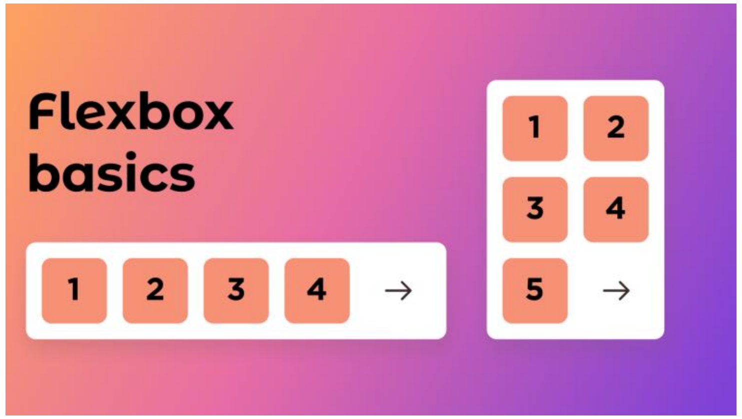
Module 3 - 4

CSS Flexbox

Objectives

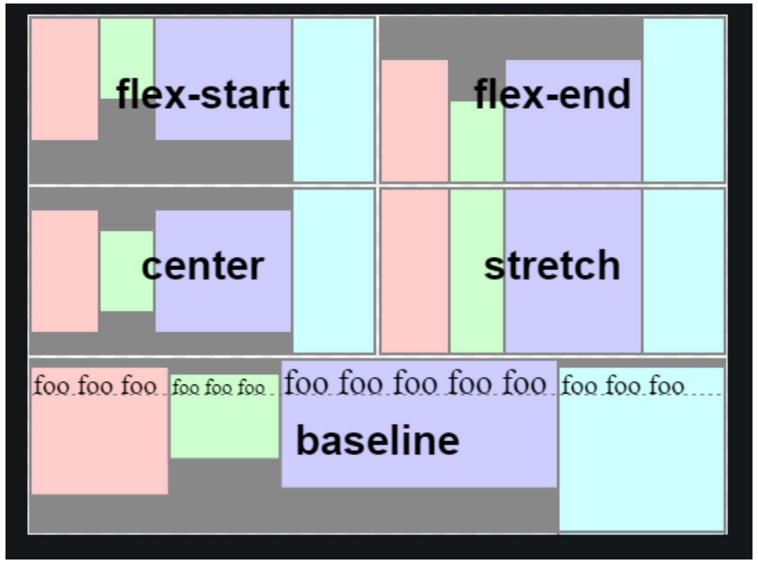


- What is Flexbox
- Define a Flexbox container using a row or a column
- Apply normal flow to Flexbox items
- Apply content alignment to flex items
- Arrange items
- Size items
- Understand how to add Flexbox layouts to existing Responsive CSS Grid layouts
- Understand when to use Flexbox or Grid or combine them both

Flexbox: Introduction

Flexboxes are specialized containers that have within them several HTML elements arranged in a specific fashion.

The flexbox will also automatically adjust to external stimuli (i.e. resizing the window).



Lecture

WHEN TO USE FLEXBOX

Main considerations when trying to determine which technology to use. While both CSS Grid and Flexbox appear to compete with each other, the use cases they support differ. Here are some questions you should ask yourself:

1. Is the layout two-dimensional or one-dimensional?

CSS Grids help with two-dimensional layouts—meaning rows and columns—while Flexbox helps with one-dimensional layouts—that is, either rows or columns. But what does this mean?

For any layout that has symmetry in both the vertical and horizontal orientations at the same time, Grid is the best choice. One reason is that Grid is more performant for this type of structure. Another reason is that Grid provides a way of spacing rows and columns—called gap—that Flexbox doesn't have. It's difficult to provide this spacing manually and requires a lot of CSS to accomplish.

The strength of Flexbox is its ability to organize, align, and adjust related content. Specifically, when considering how information or components tell a story together as a component, Flexbox is generally the answer.

WHEN TO USE FLEXBOX OR GRID

Lecture

Main considerations when trying to determine which technology to use. While both CSS Grid and Flexbox appear to compete with each other, the use cases they support differ. Here are some questions you should ask yourself:

2. Is the design based upon content or structure?

Think of a house blueprint. If your task is to build the house, then structure is important. An architect is responsible for the overall task of laying out the house and how the spaces relate to one another. An interior designer is responsible for focusing on each room of that house blueprint and thinking about how to arrange the furniture.

Solving problems with CSS means interpreting a user interface design that is much like that blueprint. If the task involves applying CSS to solve a page structure or layout problem, then Grid is the go-to for solving this problem. This is often referred to as "outside in" or "layout first" design.

On the other hand, if the task involves placing, spacing, aligning, and adjusting related content together (furniture in a room) the go-to tool is likely to be Flexbox. This is often referred to as "inside out" or "content first" design.

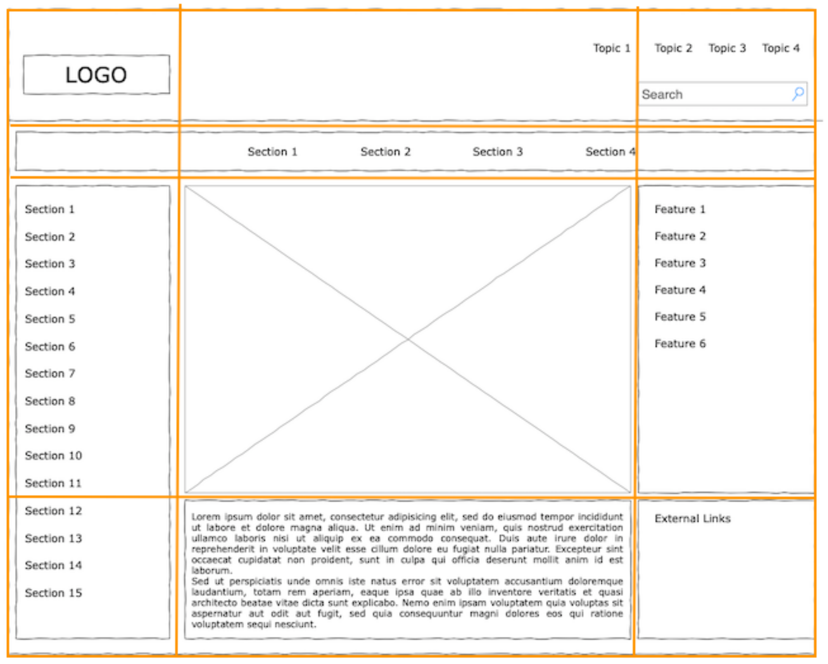
Lecture

WHEN TO USE FLEXBOX OR GRID

Grid and Flexbox don't have to be used separately. They're often used together to solve problems related to their individual strengths.

Layout symmetry in both rows and columns. There is spacing between the rows and columns. These are strengths of the grid layout.

Grid is most appropriate for this design

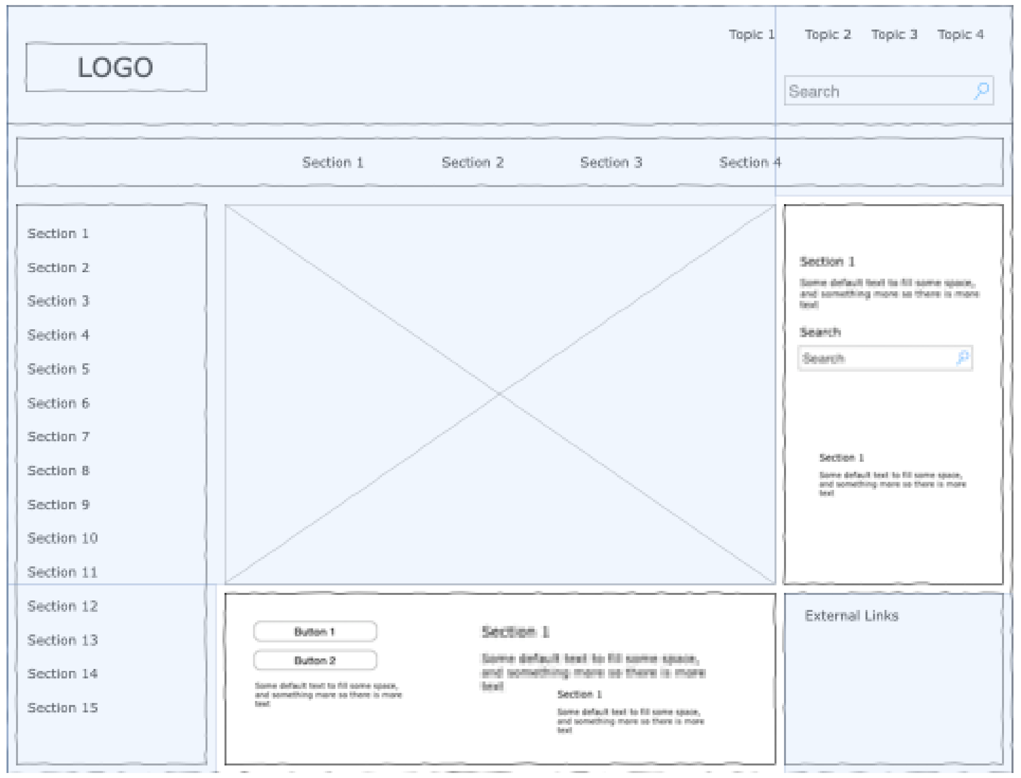


Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.
Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore veritatis et quasi architecto beatae vitae dicta sunt explicabo. Nemo enim ipsam voluptatem quia voluptas sit aspernatur aut odit aut fugit, sed quia consequuntur magni dolores eos qui ratione voluptatem sequi nesciunt.

WHEN TO COMBINE FLEXBOX AND GRID

Grid and Flexbox don't have to be used separately. They're often used together to solve problems related to their individual strengths.

Using them together


LOGO
Topic 1
Topic 2
Topic 3
Topic 4
Search
Section 1
Section 2
Section 3
Section 4
Section 1
Section 2
Section 3
Section 4
Section 5
Section 6
Section 7
Section 8
Section 9
Section 10
Section 11
Section 12
Section 13
Section 14
Section 15
Section 1
Some default text to fill some space, and something more so there is more text
Search
Search
Section 1
Some default text to fill some space, and something more so there is more text
Button 1
Button 2
Some default text to fill some space, and something more so there is more text
Section 1
Some default text to fill some space, and something more so there is more text
Section 1
Some default text to fill some space, and something more so there is more text
External Links

Grid is for building the layout of the page. Flexbox is for positioning elements relative to each other.

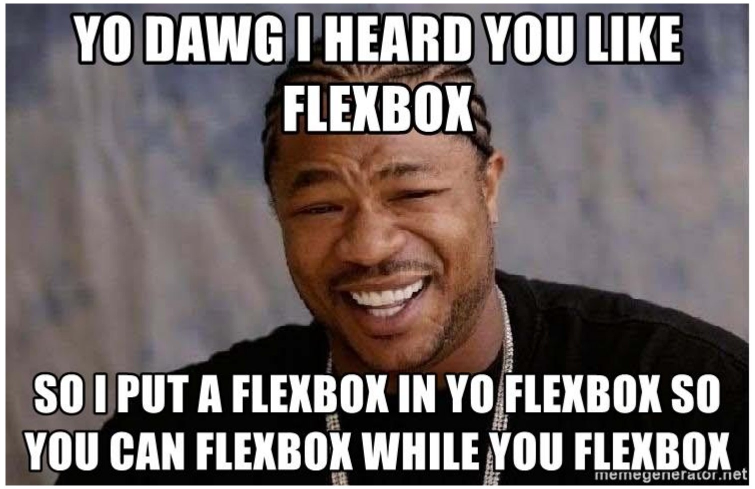
Grid is used for two-dimensional layouts, and Flexbox is used for one-dimensional layouts.

Flexbox: Defining a container

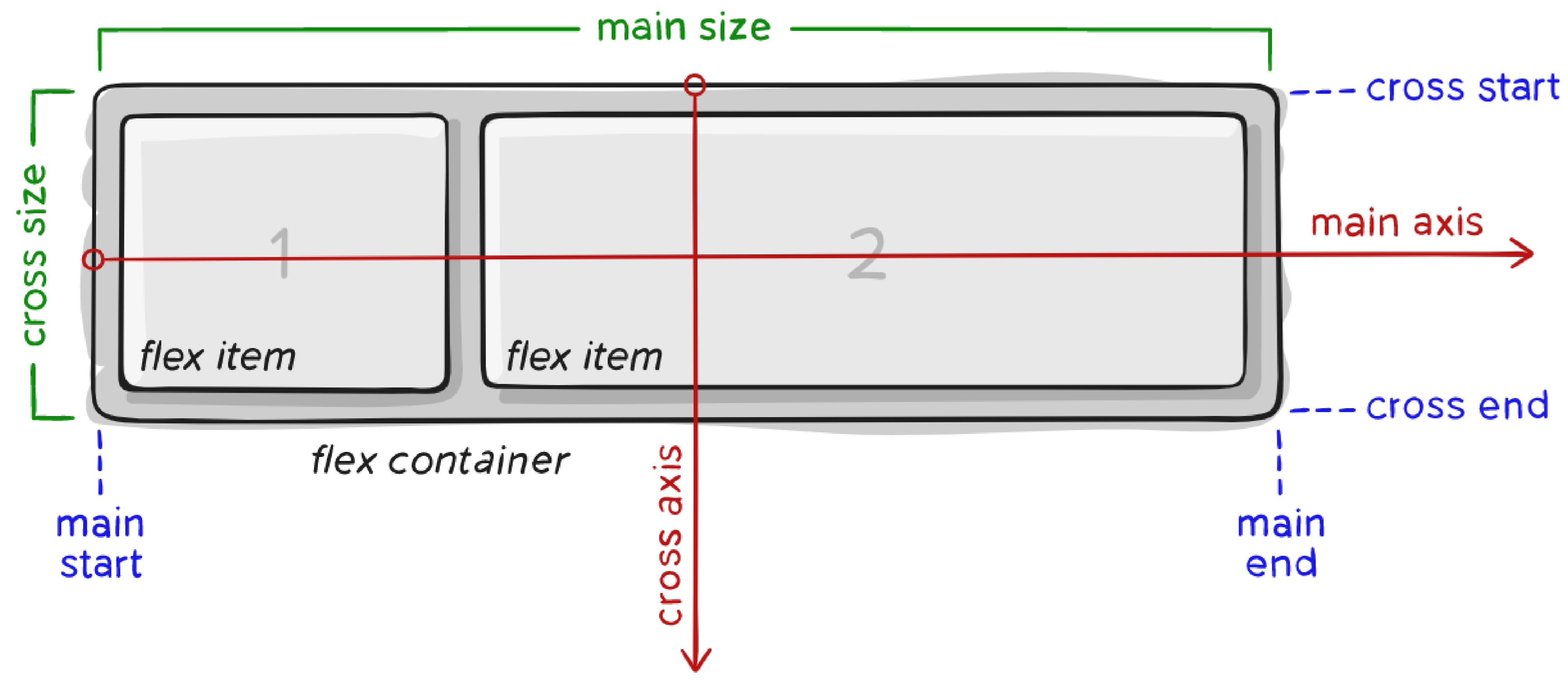
To define a flexbox we must specify a display attribute with a value of flex:

```
nav {
    display: flex;
}
```

Here, all HTML elements of type nav will be defined as flex box containers.


YO DAWG I HEARD YOU LIKE FLEXBOX
SO I PUT A FLEXBOX IN YO FLEXBOX SO YOU CAN FLEXBOX WHILE YOU FLEXBOX
memegenerator.net

Flexbox: Anatomy



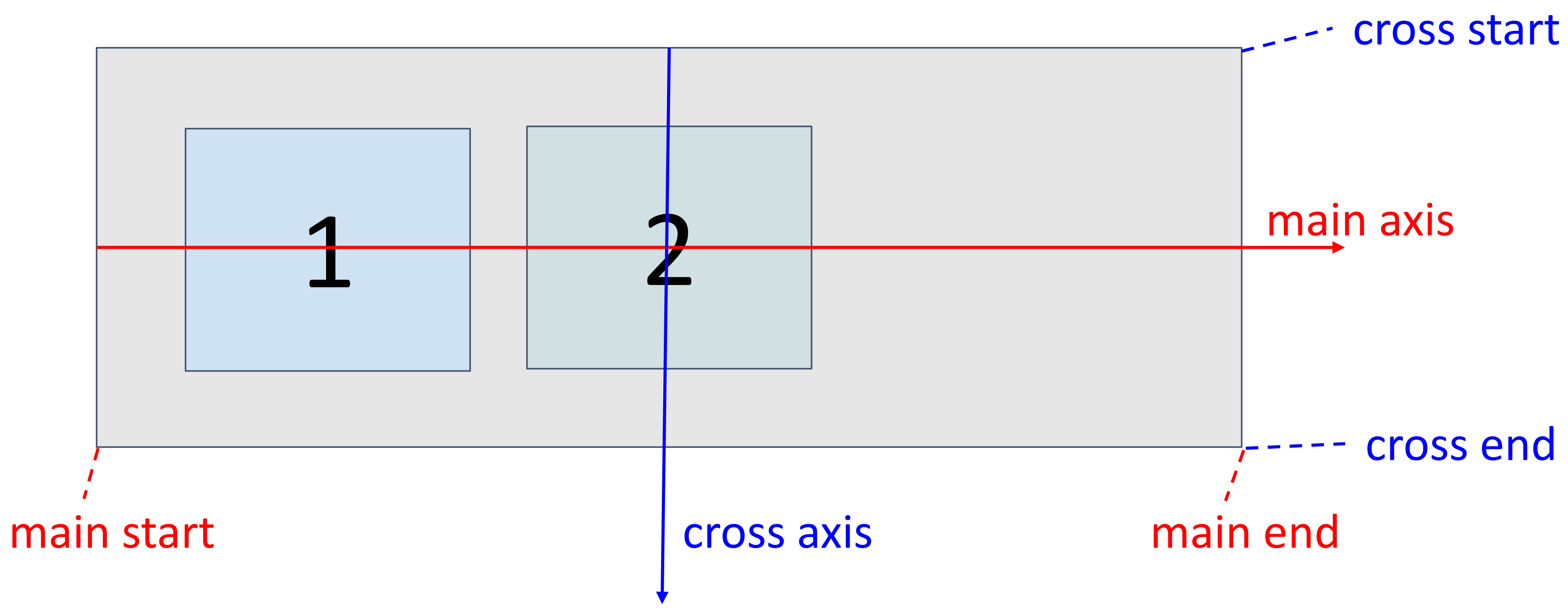
flex-direction

This establishes the direction and rotation of the main-axis, thus defining the direction flex items are placed in the flex container.

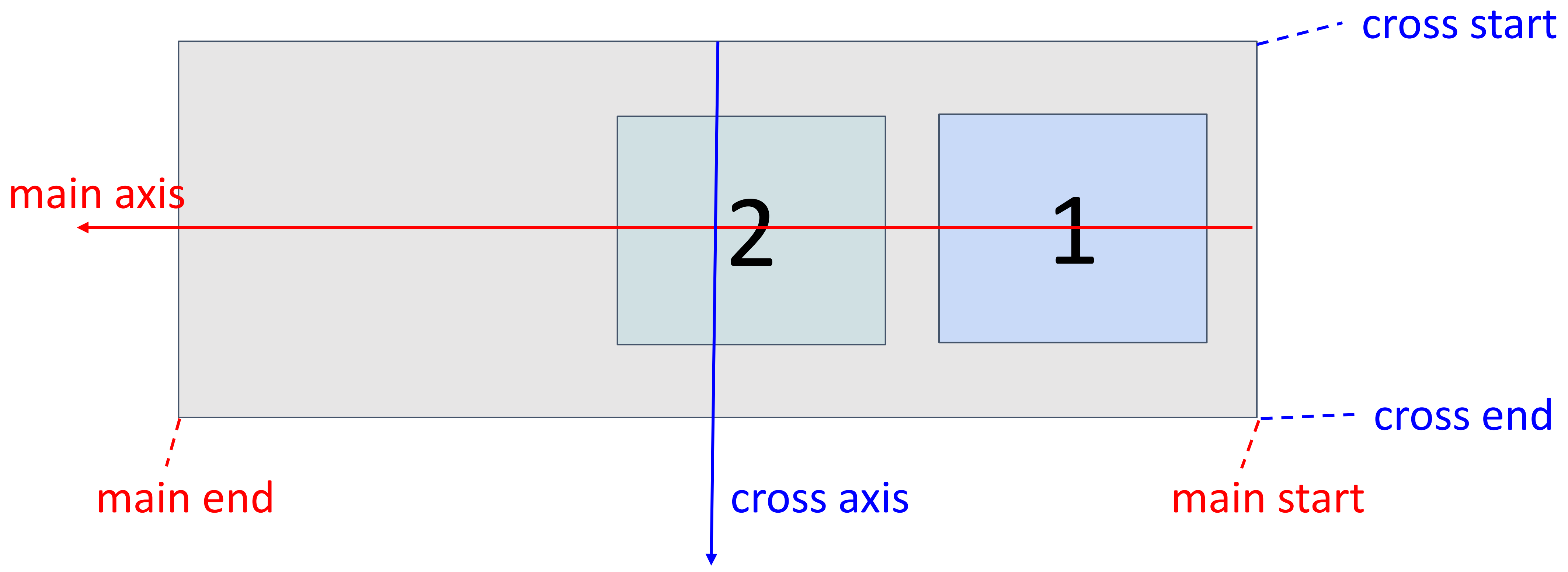
Flexbox is (aside from optional wrapping) a single-direction layout concept. Think of flex items as primarily laying out either in horizontal rows or vertical columns.

1. **flex-direction: row-reverse**
2. **flex-direction: column**
3. **flex-direction: column-reverse**
4. **flex-direction: row**

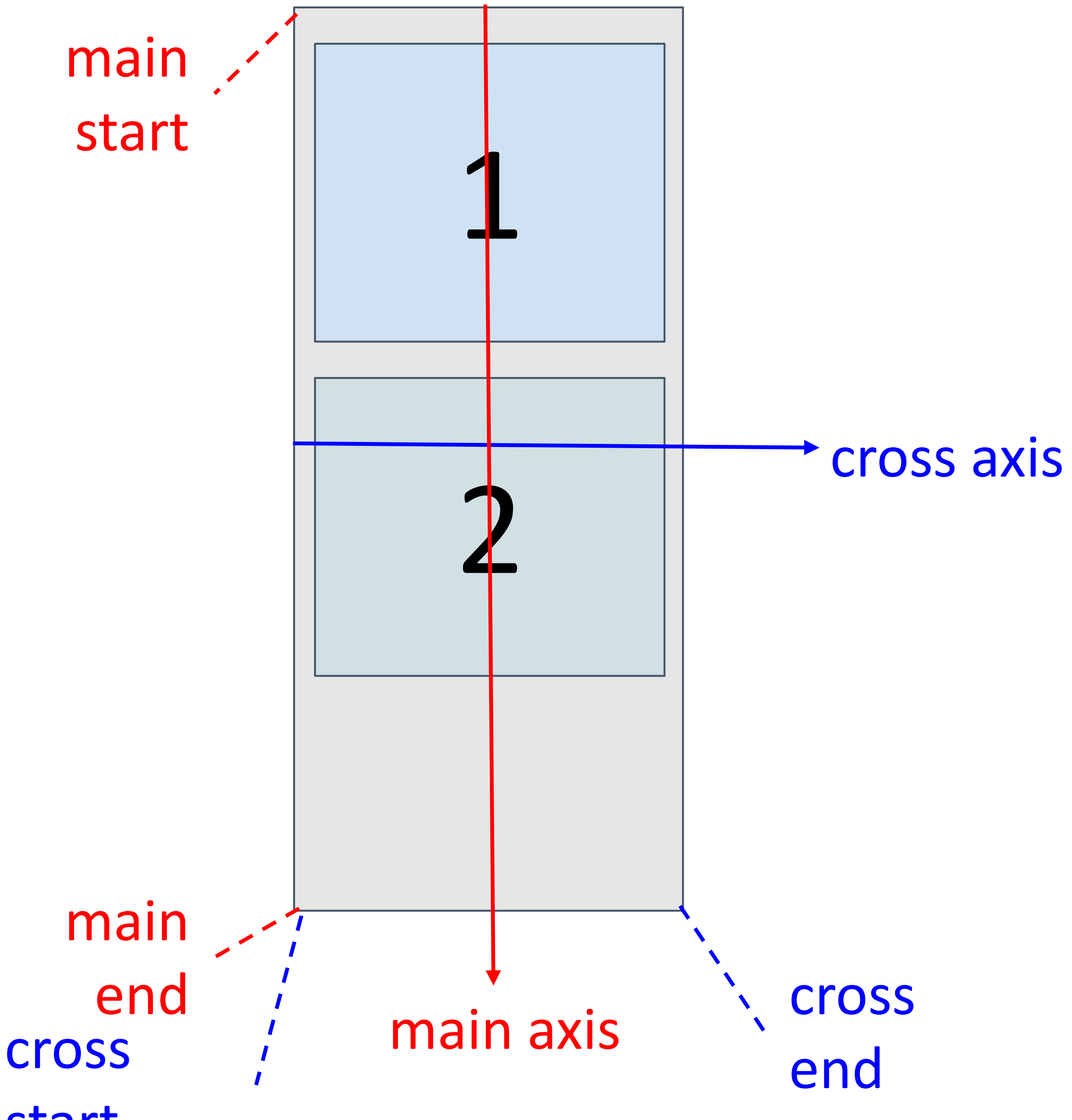
flex-direction: row



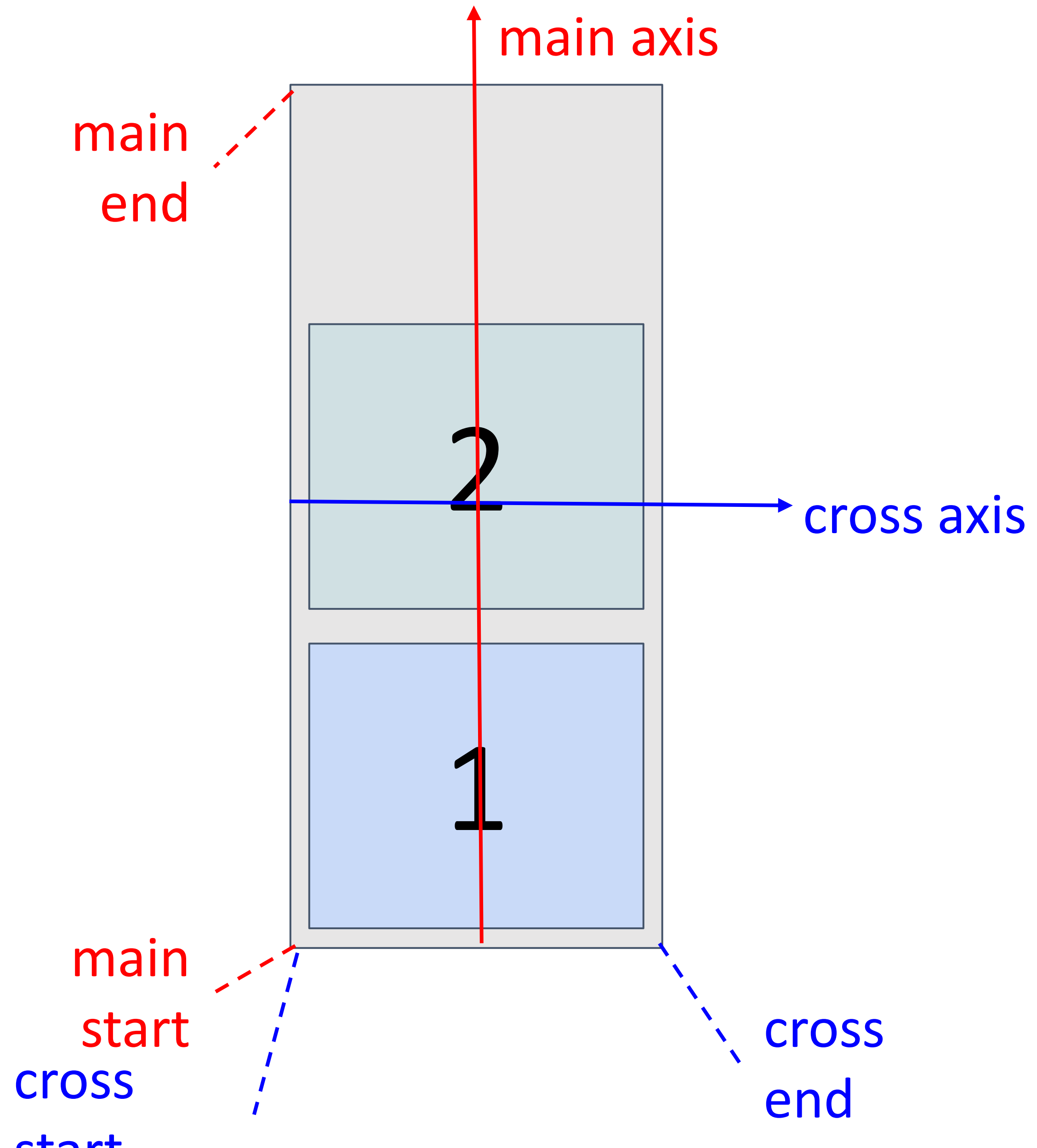
flex-direction: row-reverse



flex-direction: column



flex-direction: column-reverse

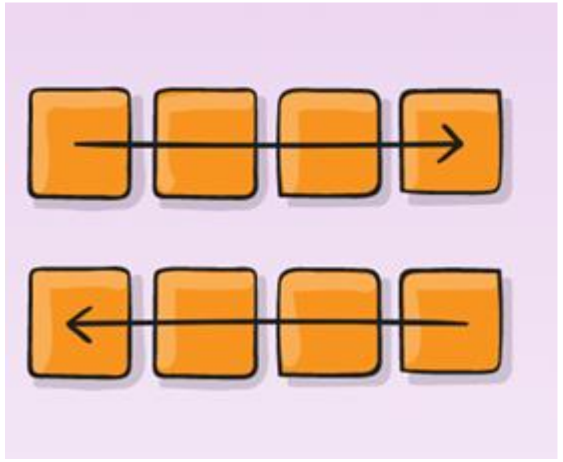


Flexbox: flex-direction

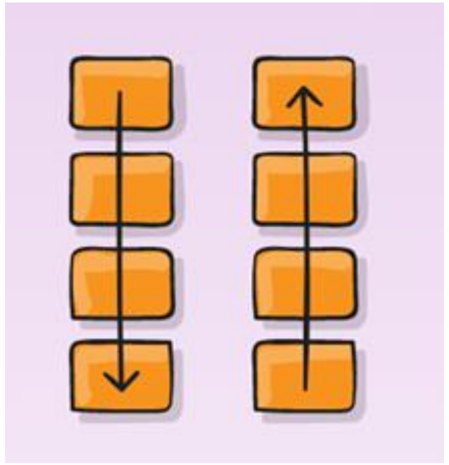
The flex-direction attribute determines whether or not the layout of items inside the box will be in either column or rows. **The default direction is row**.

```
nav {
        display: flex;
        flex-direction: column;
}
```

row:



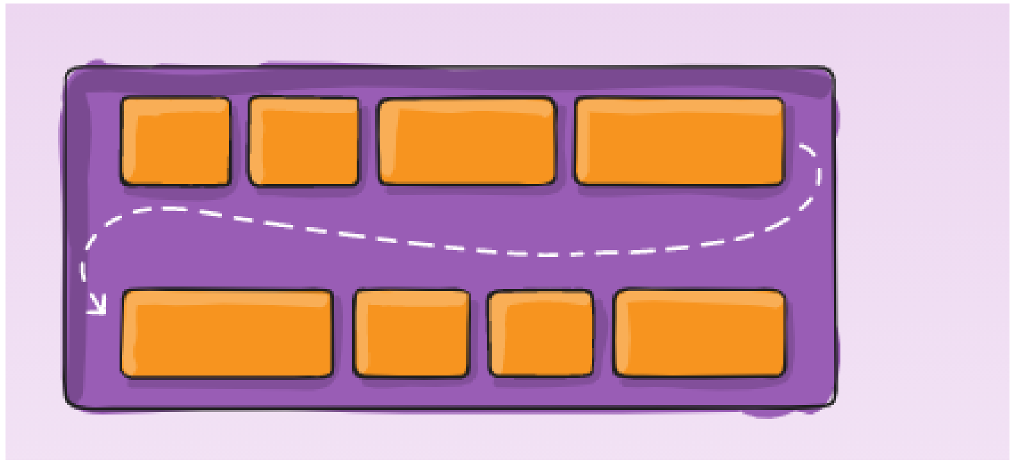
column:



Flexbox: flex-wrap

The flex-wrap attribute determines whether items will fit all on one line, wrap, or reverse wrap (from bottom to top). **The default is nowrap.**

```
nav {
        display: flex;
        flex-wrap: wrap;
}
```



Flexbox: flex-flow

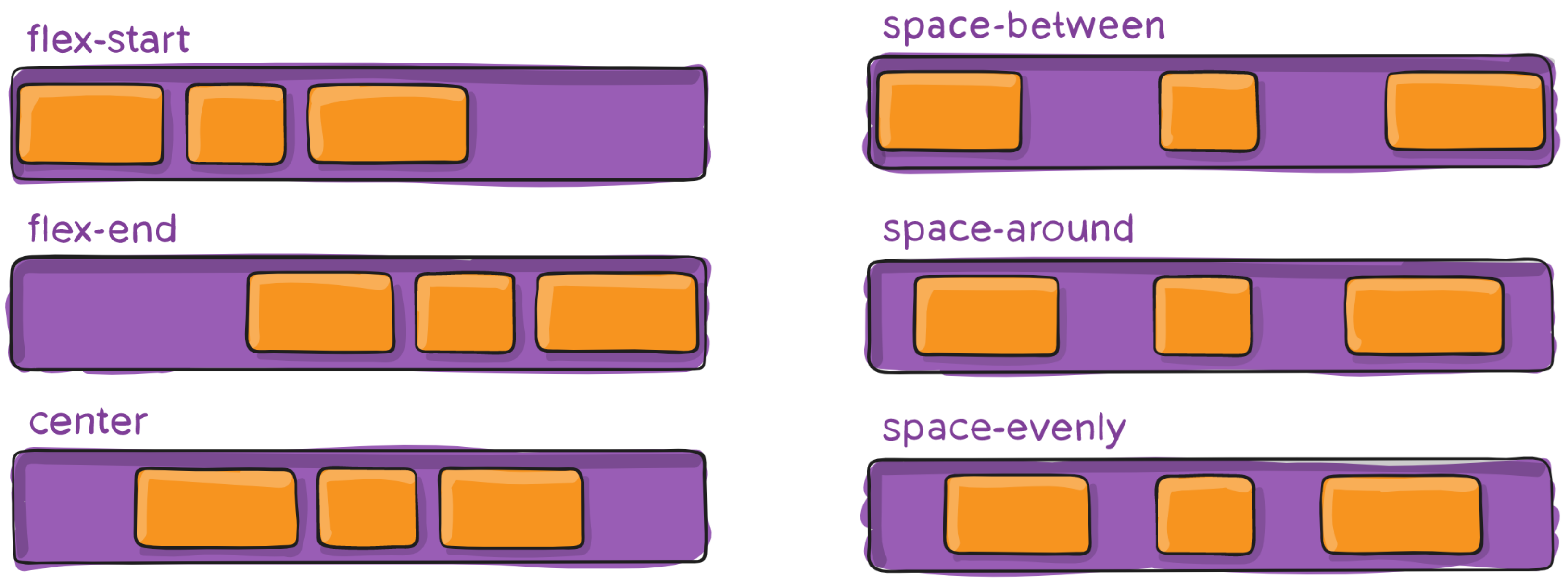
The flex-flow attribute is shorthand for flex-direction and flex-wrap. **The default is row nowrap.**

```
nav {
        display: flex;
        flex-flow: column wrap;
}
```

Flexbox: justify-content

The justify-content property defines how items are aligned across the main axis.

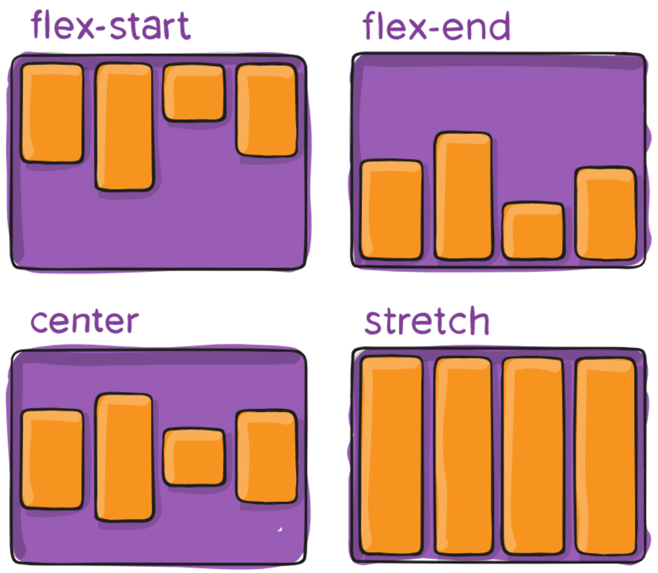
```
nav {
        display: flex;
        justify-content: flex-start;
}
```

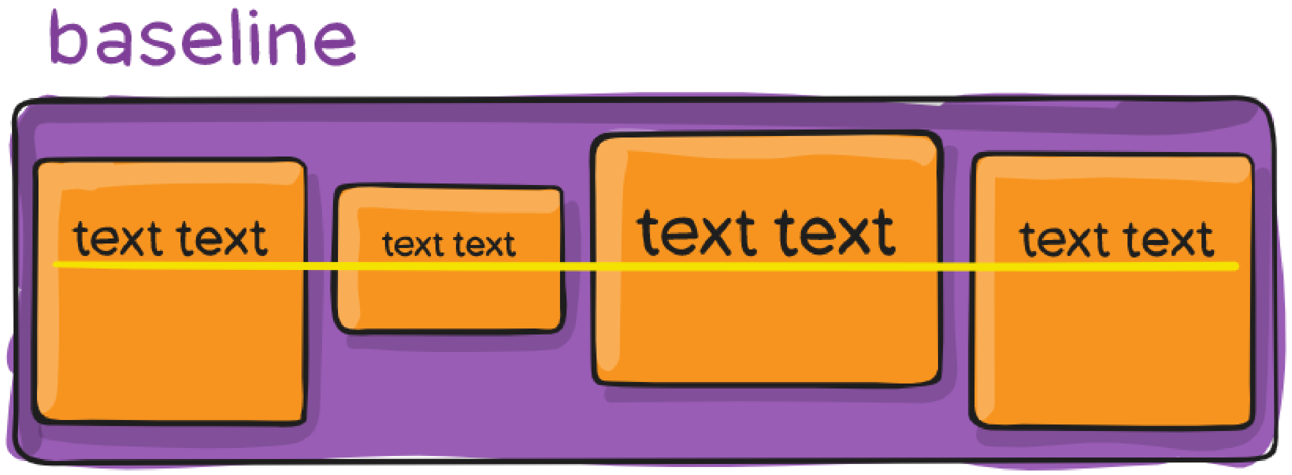


Flexbox: align-items

The justify-content property defines how items are aligned across the cross axis.

```
nav {
        display: flex;
        align-items: flex-start;
}
```

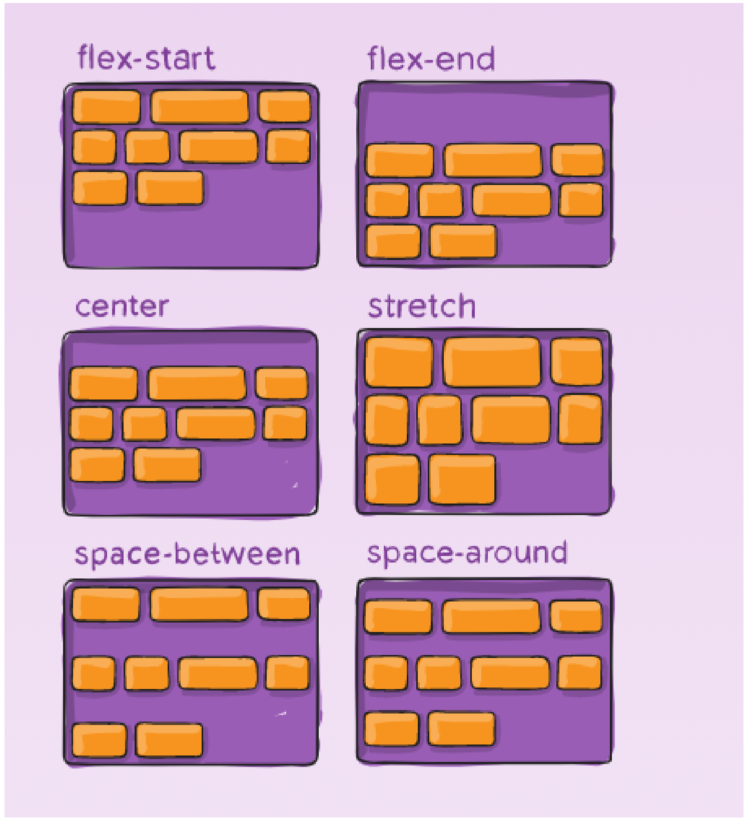




Flexbox: align-content

The align-content aligns a flex container's lines within when there is extra space in the cross-axis. This only takes effect on multi-lines flexible containers.

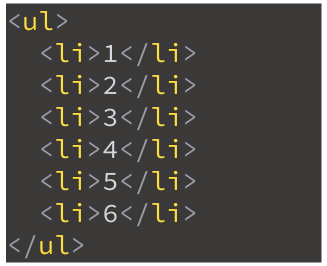
```
nav {
        display: flex;
        align-content: flex-start;
}
```



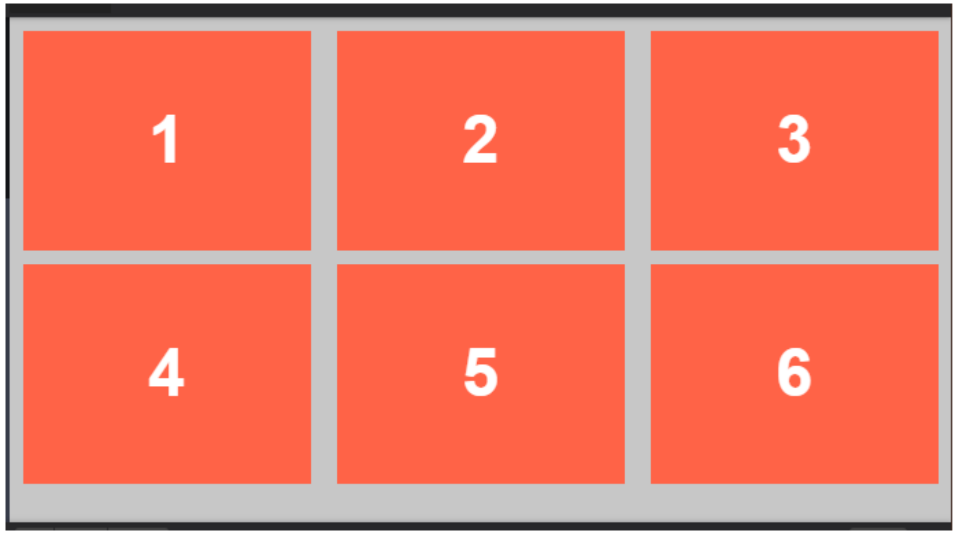
Flexbox: Defining a child

A child is any element within the flex container.

```
ul {
      display: flex;
      flex-flow: row wrap;
}
li {
      background: tomato;
      padding: 5px;
}
```

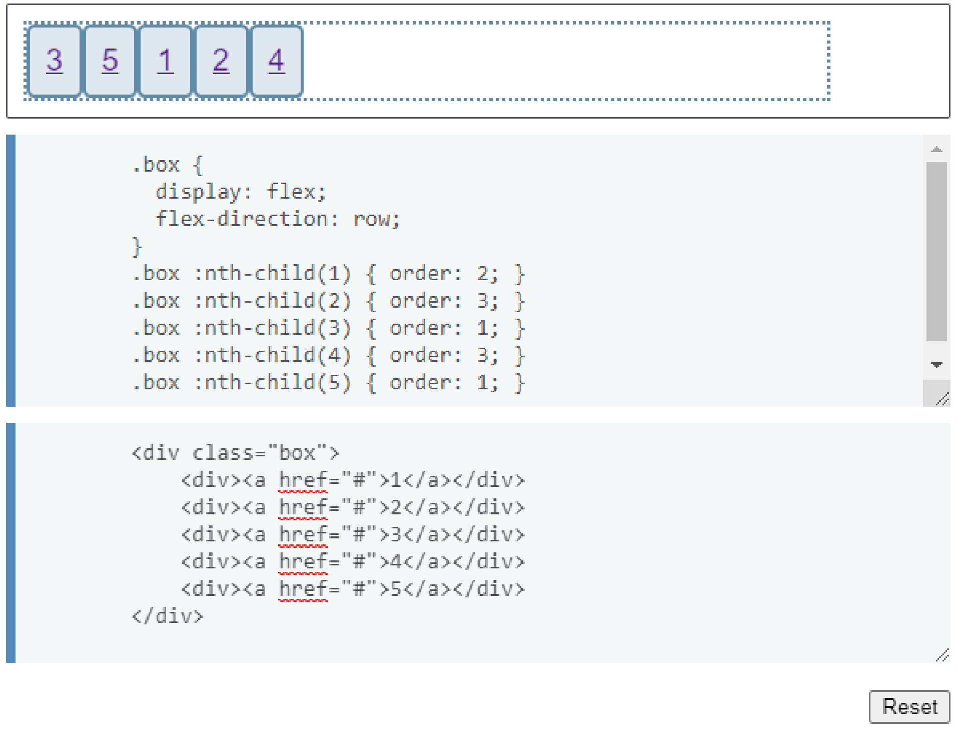


```
<ul>
  <li>1</li>
  <li>2</li>
  <li>3</li>
  <li>4</li>
  <li>5</li>
  <li>6</li>
</ul>
```



Flexbox child property: order

By default, flex items are laid out in source order. The order property allows you to control the order in which they appear in the flex container. The first position is position 0.



```
.box {
  display: flex;
  flex-direction: row;
}
.box :nth-child(1) { order: 2; }
.box :nth-child(2) { order: 3; }
.box :nth-child(3) { order: 1; }
.box :nth-child(4) { order: 3; }
.box :nth-child(5) { order: 1; }
```

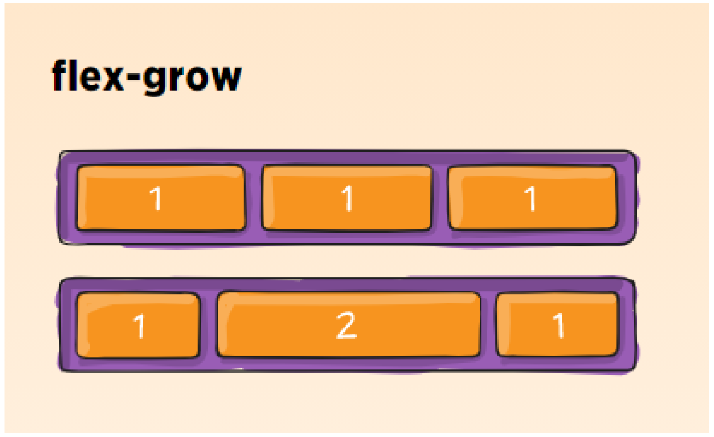
```
<div class="box">
    <div><a href="#">1</a></div>
    <div><a href="#">2</a></div>
    <div><a href="#">3</a></div>
    <div><a href="#">4</a></div>
    <div><a href="#">5</a></div>
</div>
```

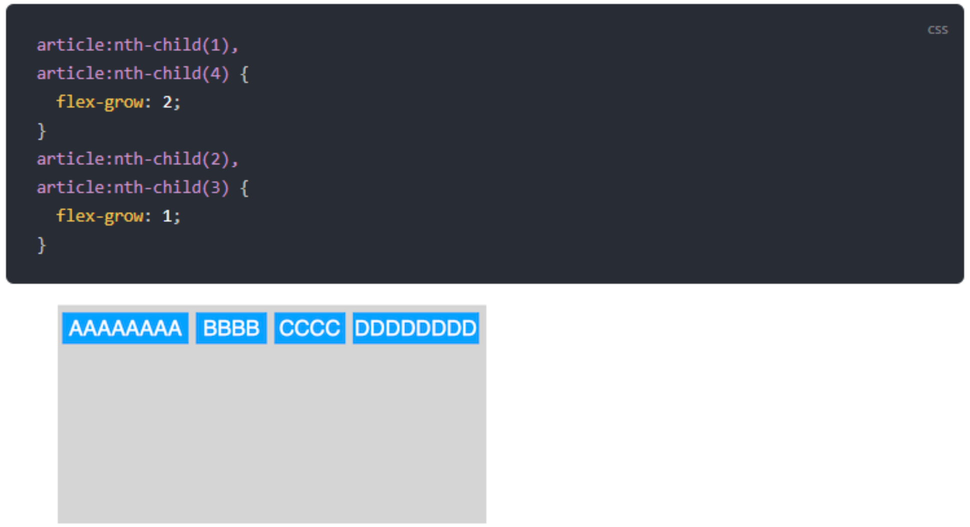
Flexbox child property: flex-grow

The flex-grow property defines the ability for a flex-item to grow if necessary.

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

This child will be given 4 times as much space as others





```css
article:nth-child(1),
article:nth-child(4) {
  flex-grow: 2;
}
article:nth-child(2),
article:nth-child(3) {
  flex-grow: 1;
}
```

AAAAAAAA BBBB CCCC DDDDDDDD

Flexbox child property: flex-shrink

The flex-shrink property defines the ability for a flex-item to shrink if necessary. You remove space rather than add.

```
.item {
        flex-shrink: 2;
}
```

This child will be given 2 times less space as others

Flexbox child property: flex-basis

The flex-basis property defines the default size of an element before the remaining space is distributed.

```
.item {
        flex-basis: 20rem;
}
```

Or

```
.item {
        flex-basis: 100px;
}
```

Flexbox child property: flex

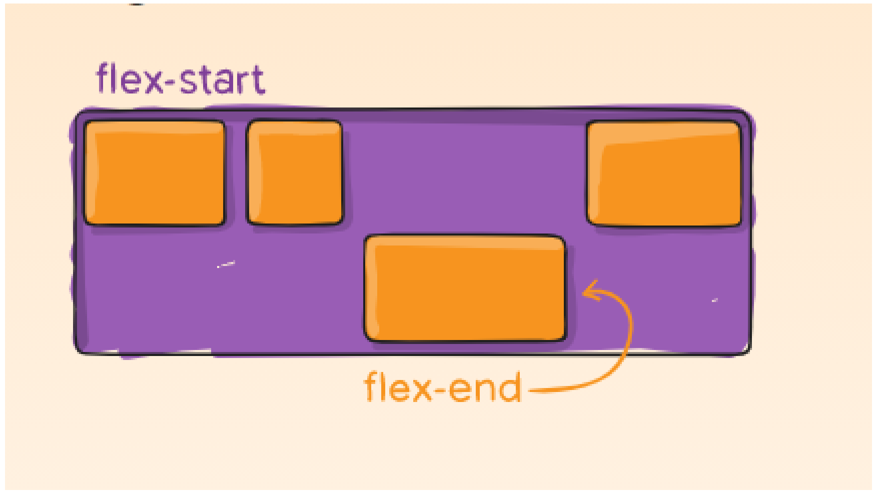
The flex property is shorthand for flex-grow, flex-shrink and flex-basis combined. The default is 0 1 auto. This is recommended over the individual properties.

```
.item {
        flex: 2 2 20rem;
}
```

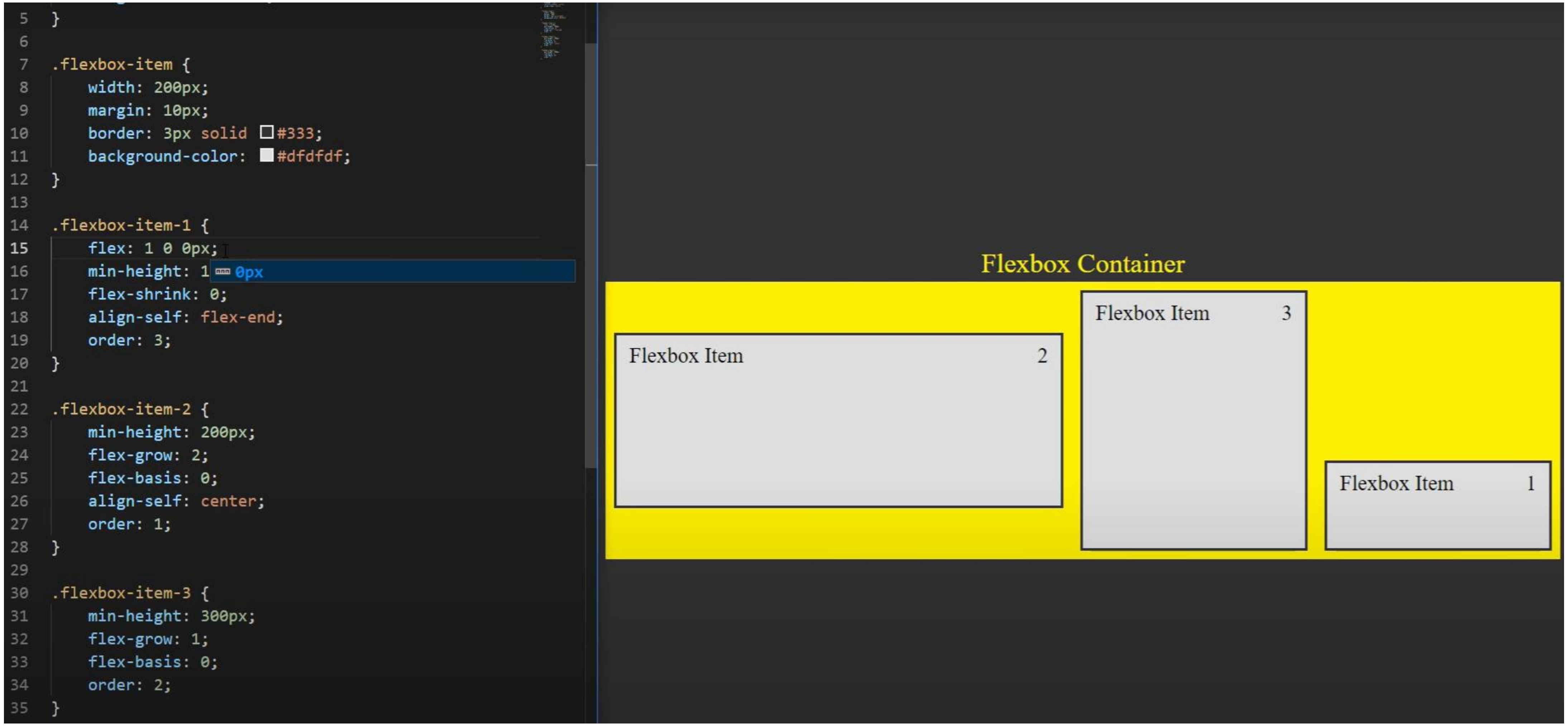
Flexbox child property: align-self

The align-self property allows the default alignment to be overridden for individul flex items.

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



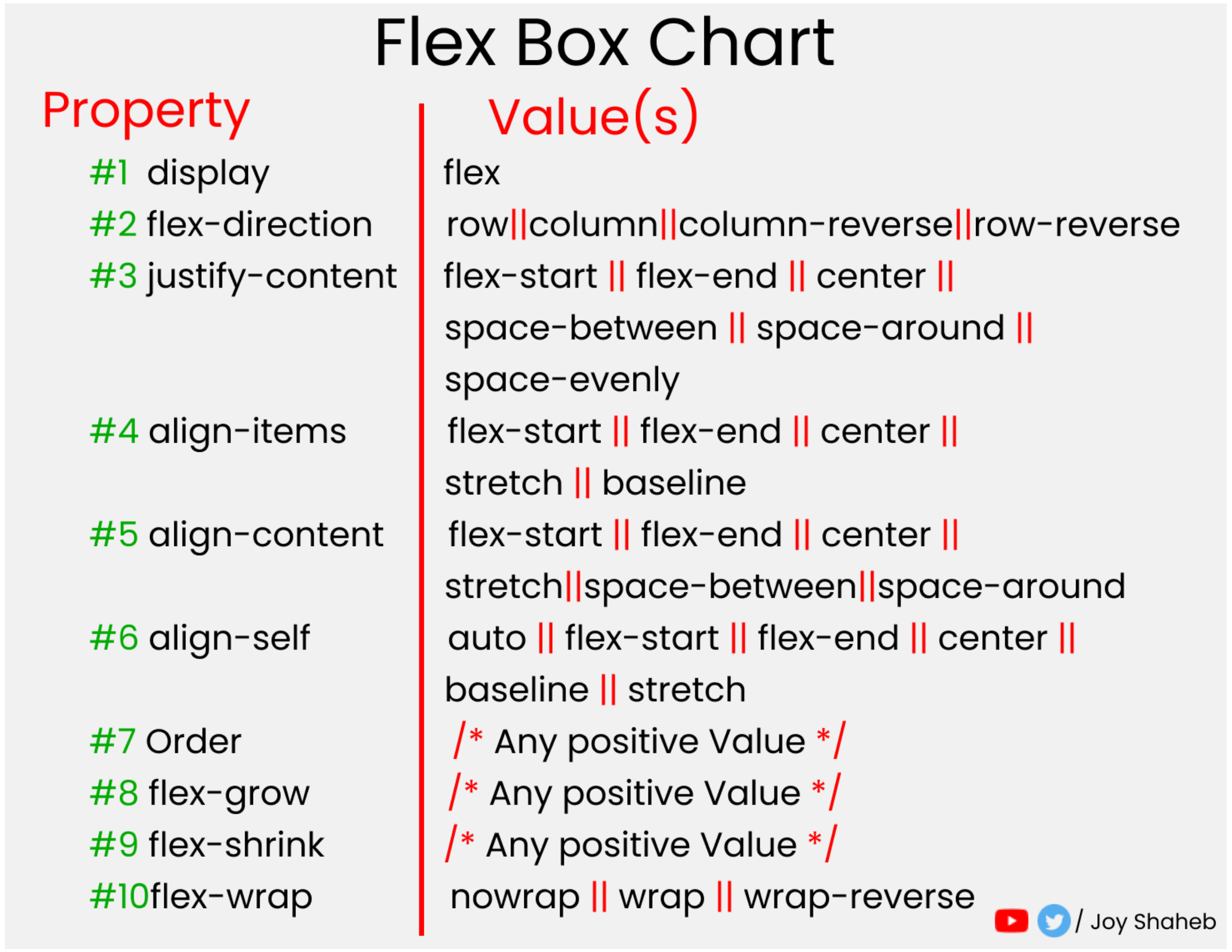
Example:



```css
}

.flexbox-item {
    width: 200px;
    margin: 10px;
    border: 3px solid #333;
    background-color: #dfdfdf;
}

.flexbox-item-1 {
    flex: 1 0 0px;
    min-height: 1 0px
    flex-shrink: 0;
    align-self: flex-end;
    order: 3;
}

.flexbox-item-2 {
    min-height: 200px;
    flex-grow: 2;
    flex-basis: 0;
    align-self: center;
    order: 1;
}

.flexbox-item-3 {
    min-height: 300px;
    flex-grow: 1;
    flex-basis: 0;
    order: 2;
}
```

Objectives

- What is Flexbox
- Define a Flexbox container using a row or a column



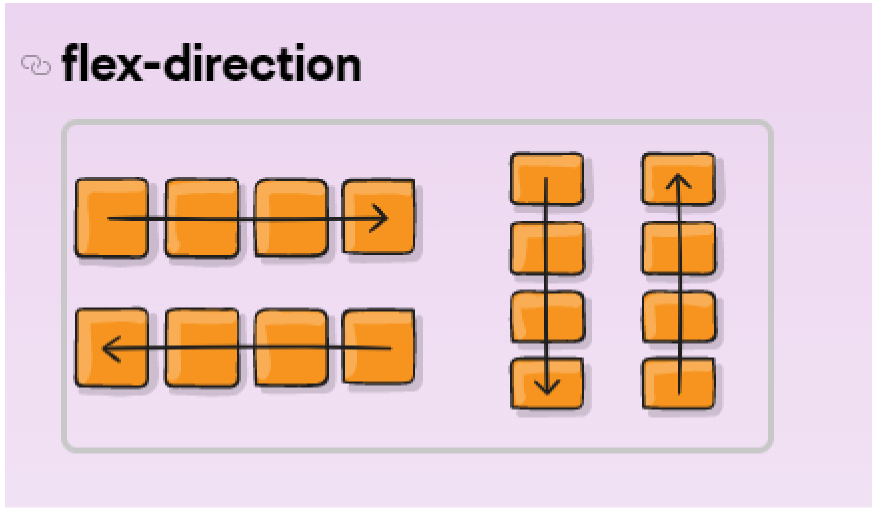
Flex Box Chart

Property	Value(s)
#1 display	flex
#2 flex-direction	row||column||column-reverse||row-reverse
#3 justify-content	flex-start || flex-end || center || space-between || space-around || space-evenly
#4 align-items	flex-start || flex-end || center || stretch || baseline
#5 align-content	flex-start || flex-end || center || stretch||space-between||space-around
#6 align-self	auto || flex-start || flex-end || center || baseline || stretch
#7 Order	/* Any positive Value */
#8 flex-grow	/* Any positive Value */
#9 flex-shrink	/* Any positive Value */
#10flex-wrap	nowrap || wrap || wrap-reverse

/ Joy Shaheb

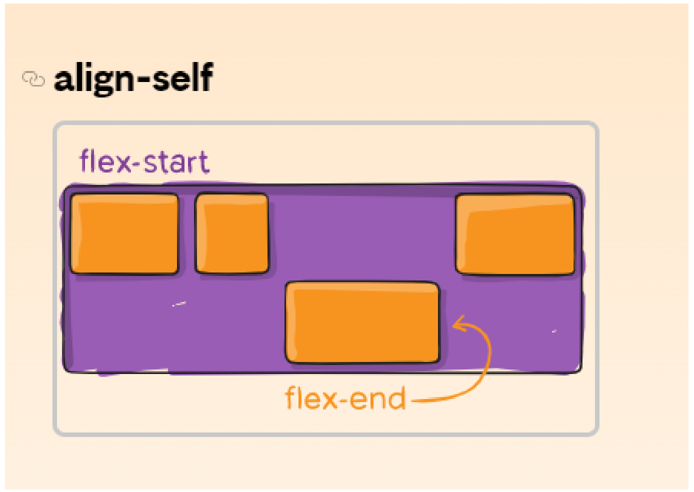
Objectives

- What is Flexbox
- Define a Flexbox container using a row or a column
- Apply normal flow to Flexbox items



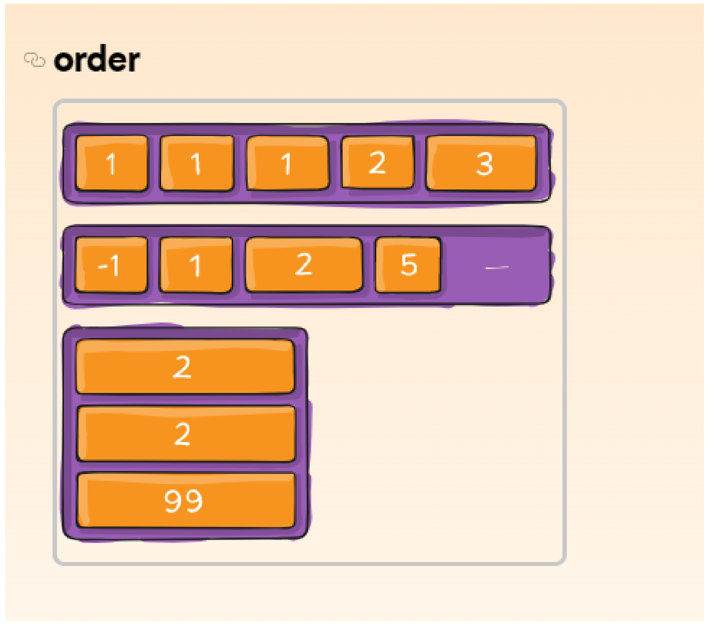
Objectives

- What is Flexbox
- Define a Flexbox container using a row or a column
- Apply normal flow to Flexbox items
- Apply content alignment to flex items



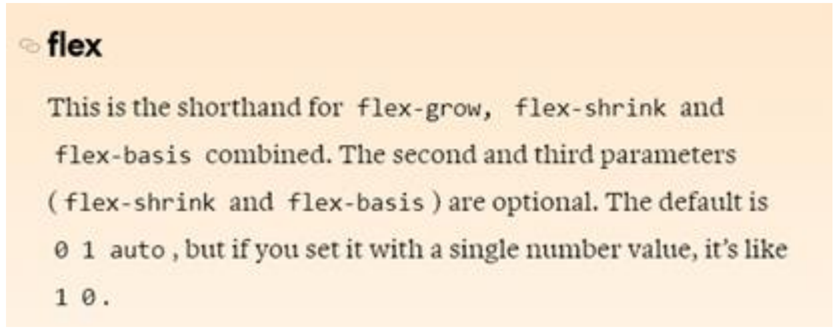
Objectives

- What is Flexbox
- Define a Flexbox container using a row or a column
- Apply normal flow to Flexbox items
- Apply content alignment to flex items
- Arrange items



Objectives

- What is Flexbox
- Define a Flexbox container using a row or a column
- Apply normal flow to Flexbox items
- Apply content alignment to flex items
- Arrange items
- Size items

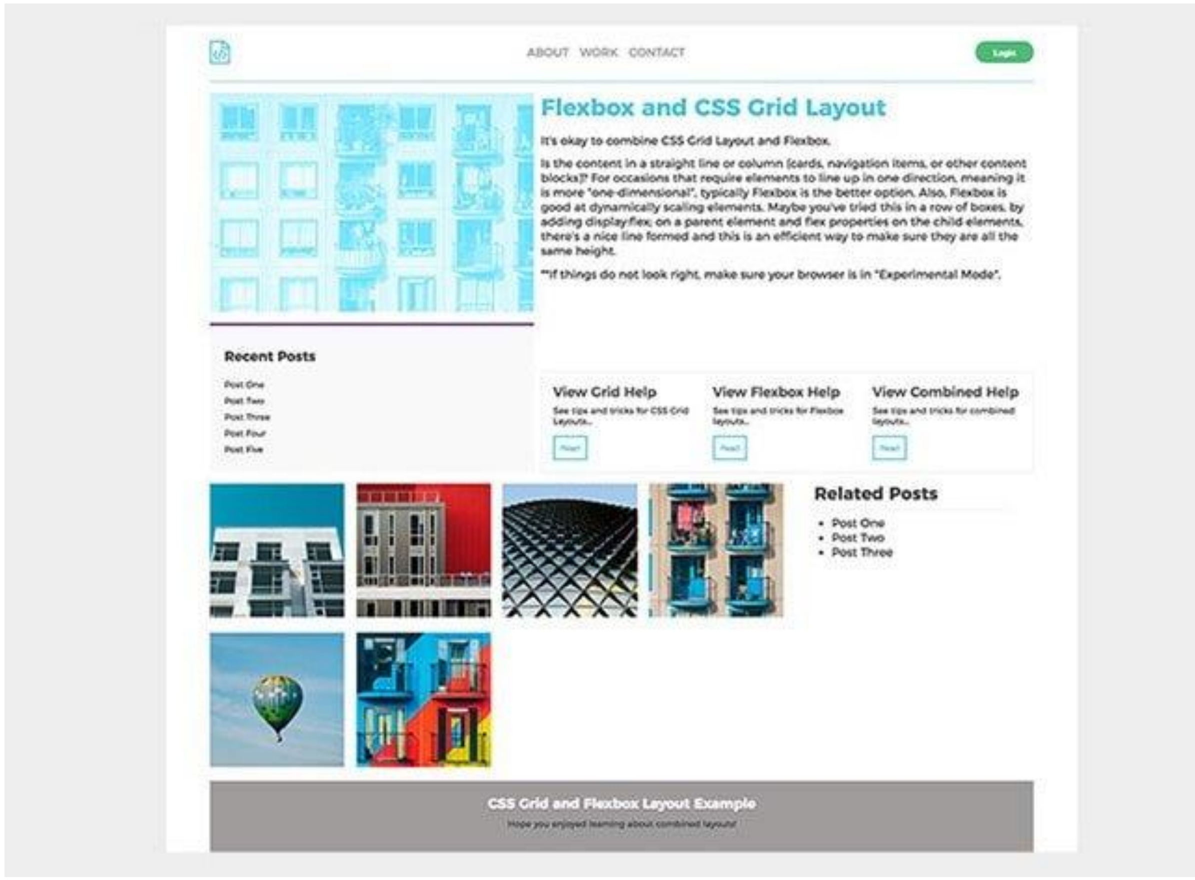


flex

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. The default is `0 1 auto`, but if you set it with a single number value, it's like `1 0`.

Objectives

- What is Flexbox
- Define a Flexbox container using a row or a column
- Apply normal flow to Flexbox items
- Apply content alignment to flex items
- Arrange items
- Size items
- Understand how to add Flexbox layouts to existing Responsive CSS Grid layouts



Flexbox and CSS Grid Layout

It's okay to combine CSS Grid Layout and Flexbox.

Is the content in a straight line or column (cards, navigation items, or other content blocks)? For occasions that require elements to line up in one direction, meaning it is more "one-dimensional", typically Flexbox is the better option. Also, Flexbox is good at dynamically scaling elements. Maybe you've tried this in a row of boxes, by adding display:flex; on a parent element and flex properties on the child elements, there's a nice line formed and this is an efficient way to make sure they are all the same height.

**If things do not look right, make sure your browser is in "Experimental Mode".

Recent Posts

Post One
Post Two
Post Three
Post Four
Post Five

View Grid Help

See tips and tricks for CSS Grid Layouts...

View Flexbox Help

See tips and tricks for Flexbox layouts...

View Combined Help

See tips and tricks for combined layouts...

Related Posts

- Post One
- Post Two
- Post Three

CSS Grid and Flexbox Layout Example

Hope you enjoyed learning about combined layouts!