# SSDP 1100 – Day 4

Course: Front-End Web Development Essentials
Instructor: Gabbie Bade

## Agenda

- Organizing CSS
- Flexbox

## Organizing CSS

## Structuring Your CSS

Your CSS files need to be organized because of the cascade and so they can be easily edited by you and other developers.

For reference… a simple WordPress website's CSS file can be 2,000 lines long.

## Generic Styles

This is the CSS that you never change. You can put this in one **.css** file and use it on every project.

It should include the following:

- normalize.css,
- box sizing reset,
- responsive media elements,
- and more later on…

## Base Styles

In your custom CSS file you should begin with base styles:

- **Typography**: Set the default font styles (family, size, color, line height, etc.) on the body element and then set any changes to text elements like headings and paragraphs.
- **Elements**: Set general styles on your site for HTML elements like lists, tables, figures, images, etc.
- **Links**: Style links and all of their pseudo-selectors.
- **Forms**: Style form fields.

## Layout Styles

After setting the base styles of your site, you should define the layout of your site in your custom CSS file.

This is primarily for larger screens, we will cover this today.

## Component Styles

Think of your HTML pages as multiple components…
For example: a header, sections, a footer.

Organize your CSS similarly…

- **Header**: Styles for elements in your page header.
- **Main**: Styles for the main content of your site, likely broken down into sub-sections for organizing.
- **Footer**: Styles for elements in your page footer.

## Example CSS Structure

```
# Generic (can be called in a separate CSS file)
# Base
    - Typography
    - Elements
    - Links
# Layout
# Components
    - Header
        - (Sub-sections for site heading/logo, navigation, etc.)
    - Main
        - (Sub-sections for different sections, articles, etc.)
    - Footer
        - (Sub-sections for copyright, navigation, etc.)
```

## Example CSS Structure

Open the **sample-css-structure.css** file in the **day-04** folder to see an example.

Feel free to use this as a starting CSS file for your websites along with our **normalize-fwd.css** file.

## Writing Your CSS

Now that you have a structure for your CSS file, how should we write our CSS within that structure?

- Keep it modular. Example: put all the styles for one section together.
- Target **elements** to apply changes throughout the site, target **classes** to change specific elements. Example: all paragraphs will have black font but the paragraph with the class “red” will be red.
- Limit the number of selectors you use to target an element to a maximum of three. Example: `.cards article h2 { }`

## CSS Naming Conventions

If you want to get fancy, these are commonly used:

- **BEM** – Block, Element, Modifier
- **OOCSS** – Object-Oriented CSS
- **ACSS** – Atomic CSS
- **SMACSS** – Scalable and Modular Architecture for CSS

https://www.creativebloq.com/features/a-web-designers-guide-to-css-methodologies

# Flexbox

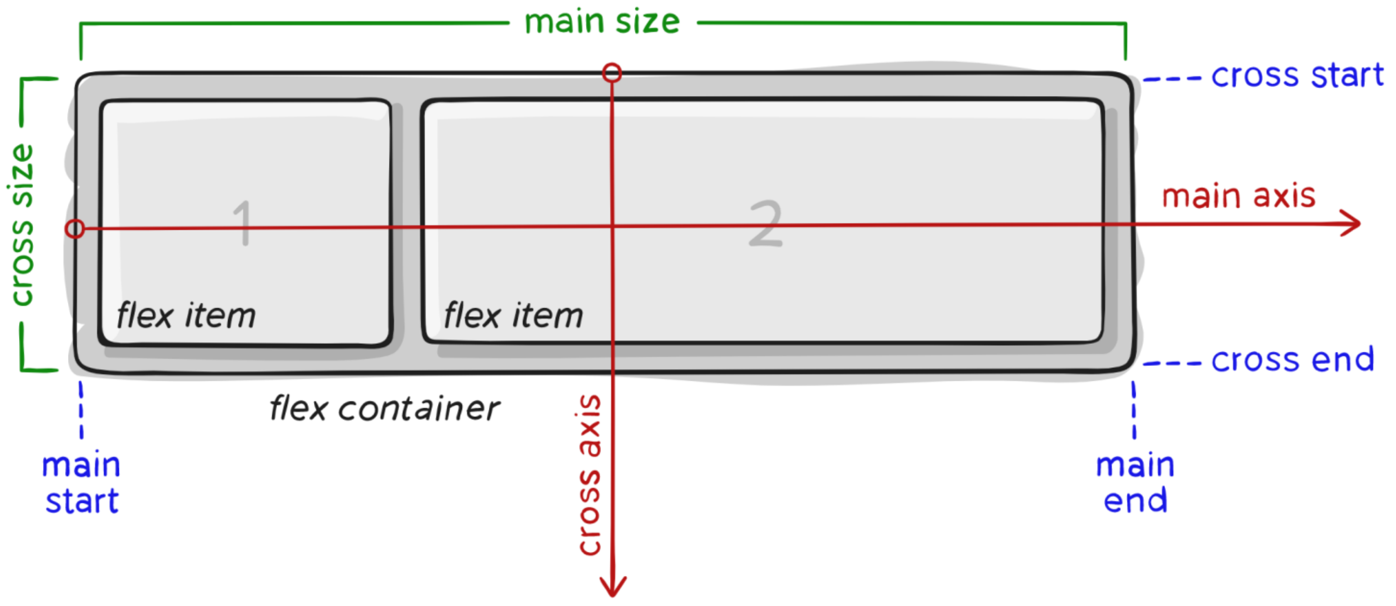
## Flexbox

"Flexbox is a one-dimensional layout method for laying out items in rows or columns. Items flex to fill additional space and shrink to fit into smaller spaces." – *MDN*

https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS_layout/Flexbox

https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Flexible_Box_Layout

# Flexbox Terminology



https://css-tricks.com/snippets/css/a-guide-to-flexbox/

## Basic Flexbox

At it's simplest, you can add `display: flex;` to an element.

This will make all of the children of that element flex items.

```
.container {
    display: flex;
}
```

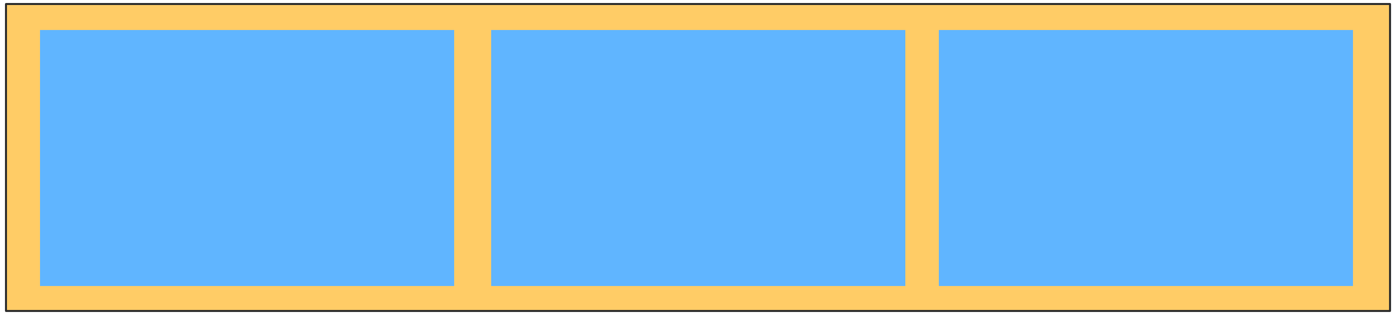
The "flex container"

```
<div class="container">
    <div>Box 1</div>
    <div>Box 2</div>
    <div>Box 3</div>
</div>
```

The direct children will be "flex items".

## Default Flexbox

By default, a flexed container will display ALL of the children in a single row with equal heights.

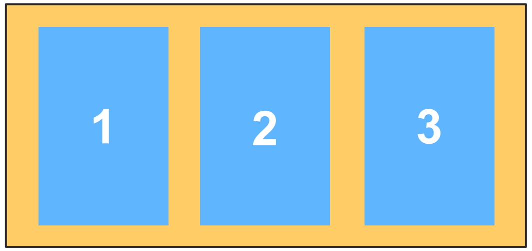


# Flex Direction

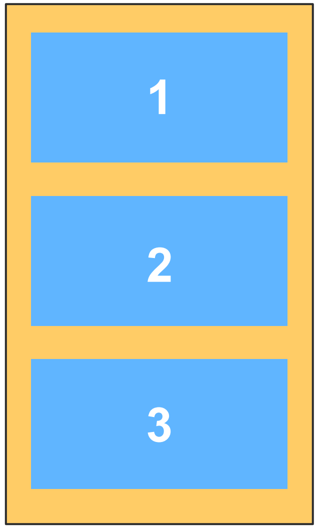
You can also display flex items in a column.

(default)

```
flex-direction: row;
```



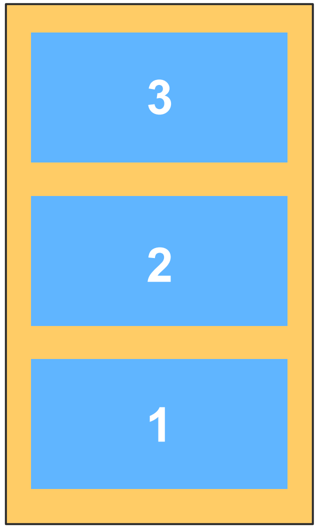
```
flex-direction: column;
```



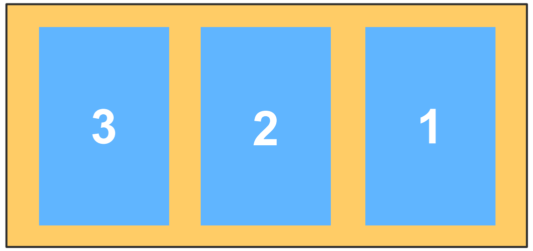
## Flex Direction - Reverse

You can reverse the order in either a row or column too.

`flex-direction: column-reverse;`



`flex-direction: row-reverse;`

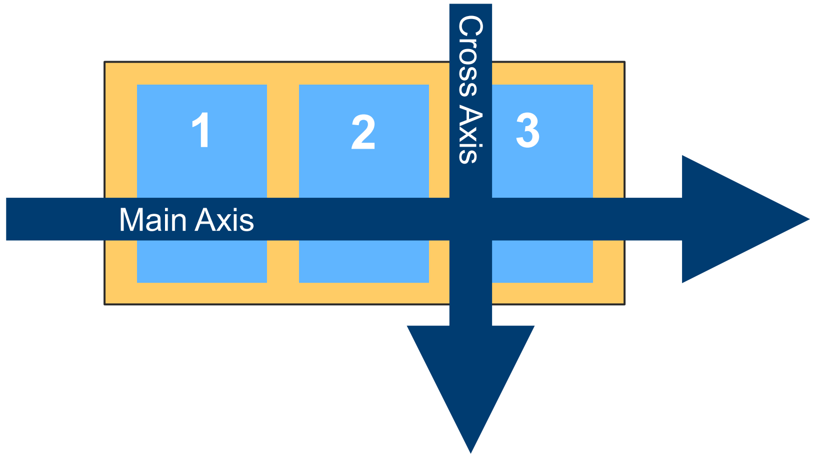


# Main Axis and Cross Axis

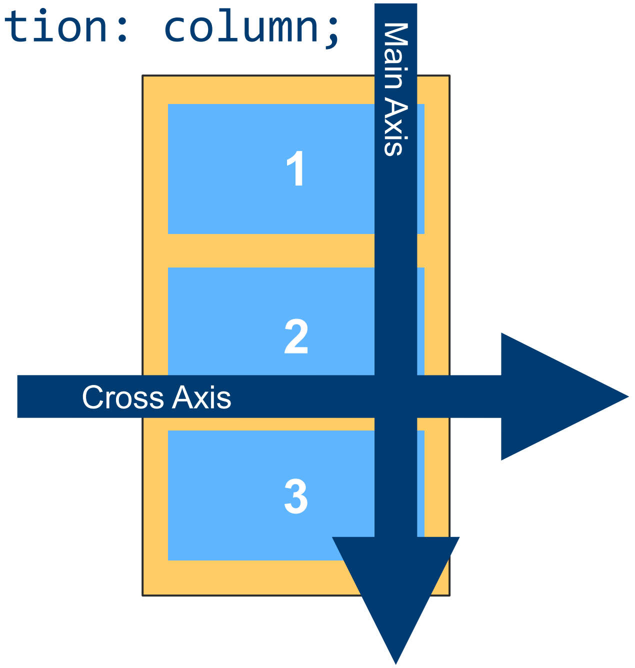
The flex direction determines the main and cross axis.

(default)

```
flex-direction: row;
```

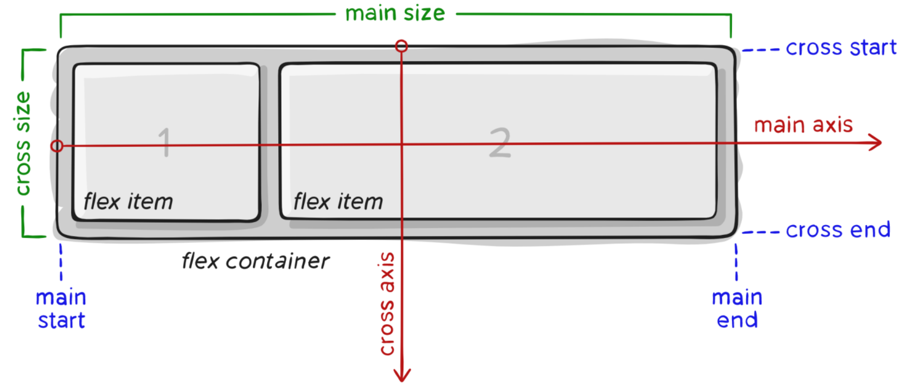


```
flex-direction: column;
```

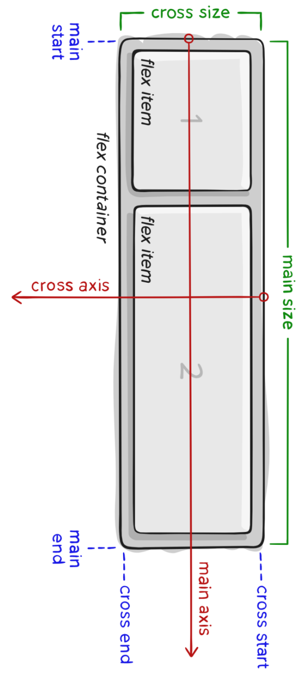


# Main Axis and Cross Axis

(default)
`flex-direction: row;`



`flex-direction: column;`



## Using the Axes

Flexbox has CSS properties that can change how items are laid out on the main axis and the cross axis.

To know which property to use, you need to know which axis you are trying to change.

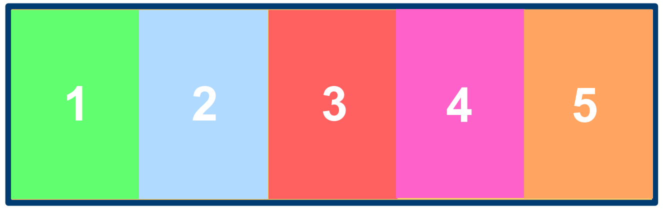
More on this shortly…

## Flex Wrap

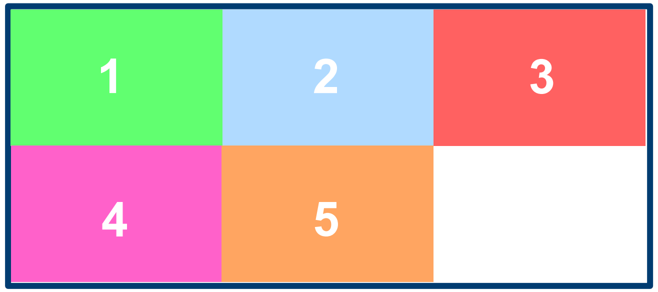
By default, flex items will stay in a single row (or column).

To allow multiple rows (or columns), change the flex wrap property.

(default)
`flex-wrap: nowrap;`



`flex-wrap: wrap;`



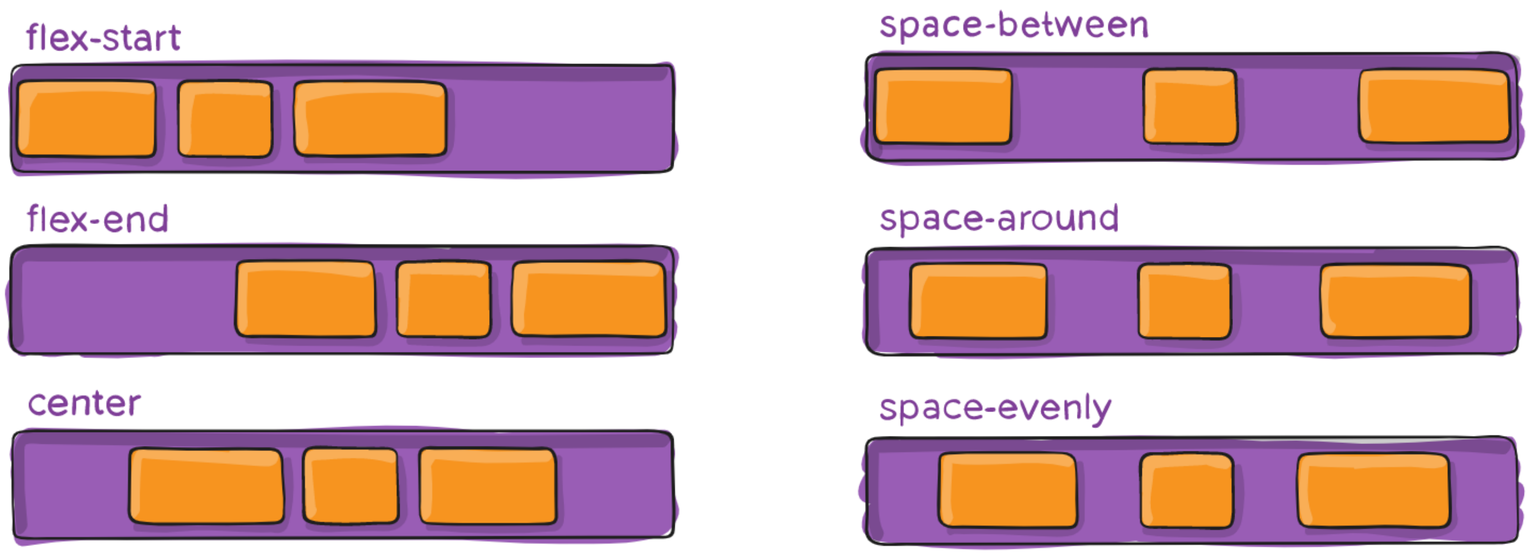
## Justify Content

To change how flex items are aligned along the **main axis**, you can use the justify content property.

The justify content property has five primary values:

| | | |
|---|---|---|
| (default) `flex-start` | `flex-end` | `center` |
| `space-between` | `space-around` | `space-evenly` |

## Justify Content Values



https://css-tricks.com/snippets/css/a-guide-to-flexbox/

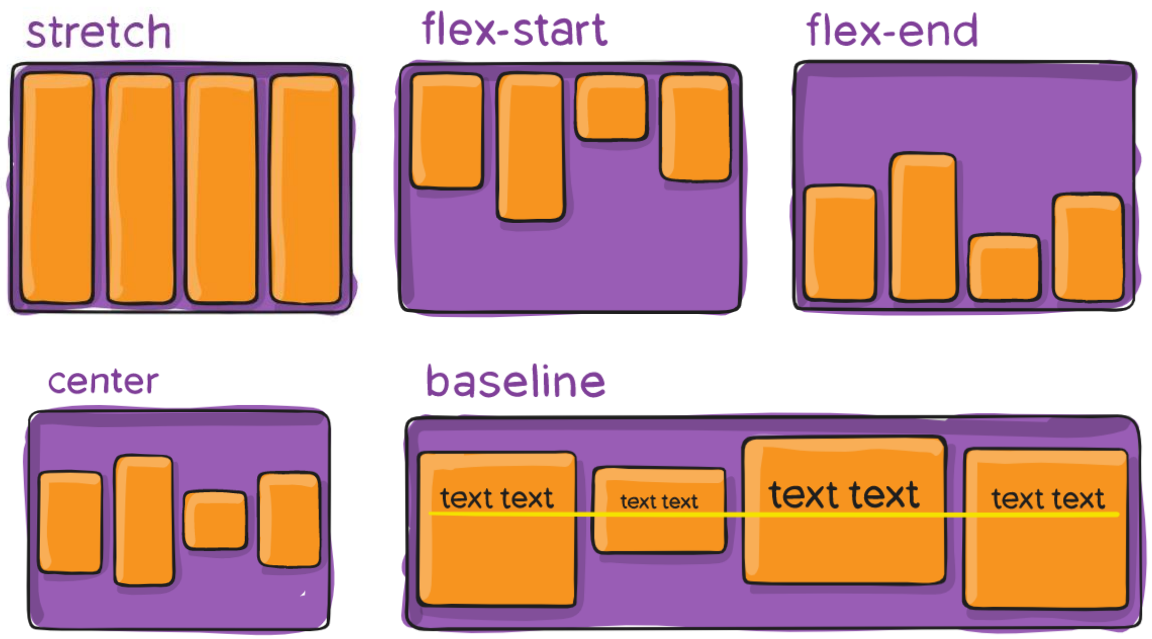
## Align Items

To change how flex items are aligned along the **cross axis**, you can use the align items property.

The align items property has five primary values:

| | | |
|---|---|---|
| (default) `stretch` | `flex-start` | `flex-end` |
| `center` | `baseline` | |

## Align Items Values



https://css-tricks.com/snippets/css/a-guide-to-flexbox/

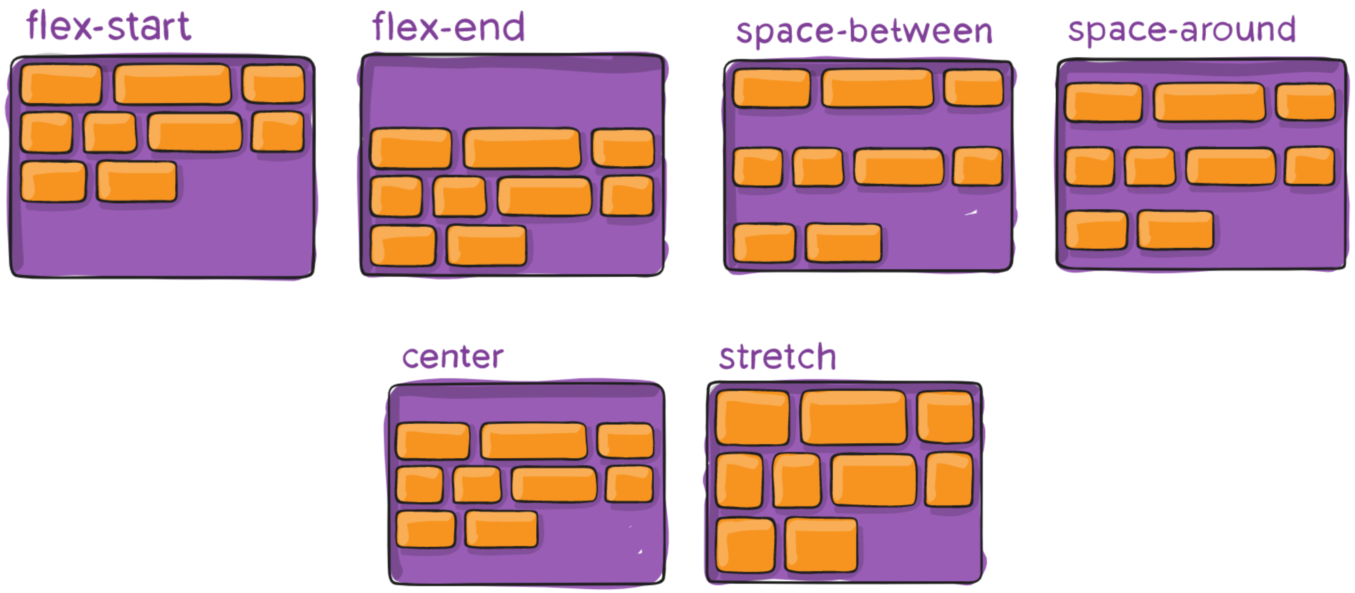
## Align Content

If the **cross axis** of the flex container is larger than the flex items, you can use the align content property.

The align content property has five primary values:

| | | |
|---|---|---|
| (default) `flex-start` | `flex-end` | `center` |
| `space-between` | `space-around` | `space-evenly` |

## Align Content Values



https://css-tricks.com/snippets/css/a-guide-to-flexbox/

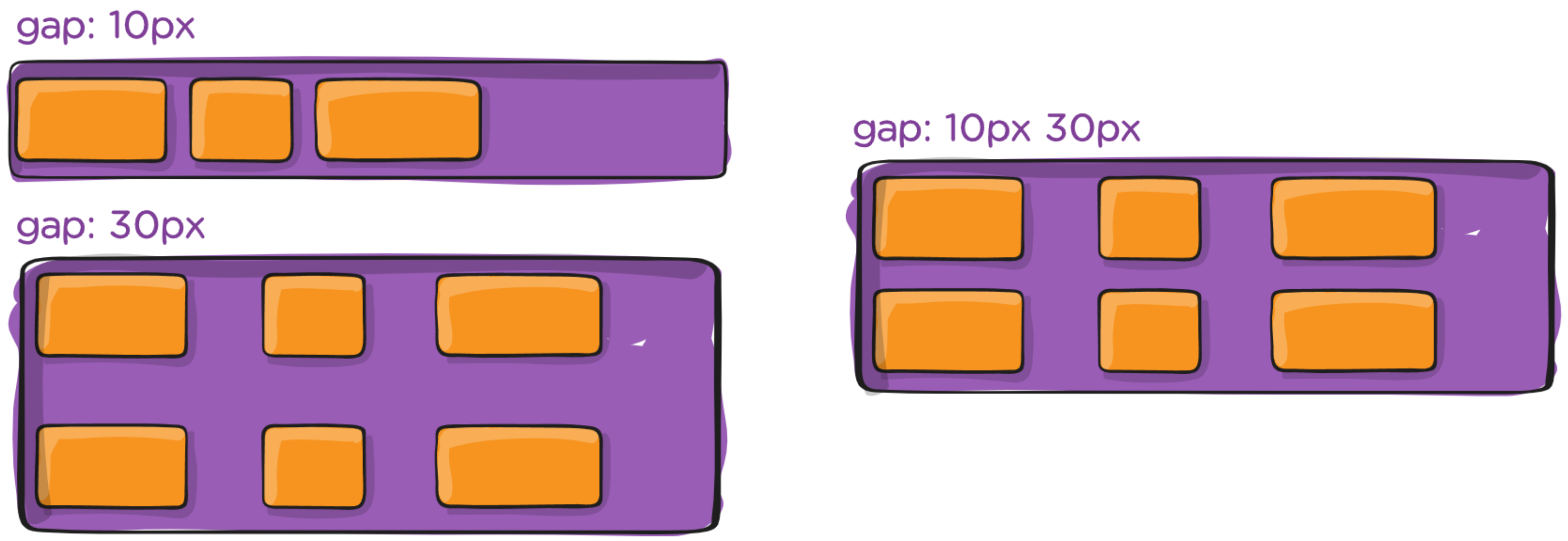
## Gap

The **gap** property sets the space between flex items in a flex container.

The gap property can set the row gap and column to different values or the same value.

*Note:* It works similar to setting “margin” on all flex items.

## Gap Examples



https://css-tricks.com/snippets/css/a-guide-to-flexbox/

## Flex Item Properties

All of the previous CSS properties applied to the **flex container**.

The following CSS properties apply to the **flex items**:

- `flex`
- `align-self`
- `order`

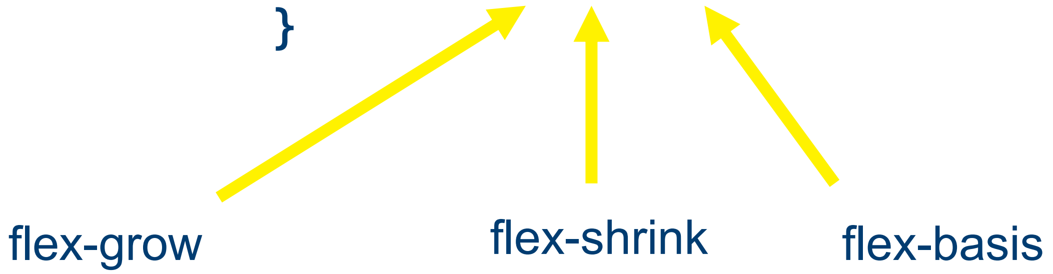
## Flex Property

The flex property is set on flex items and has three values:

- **Flex Grow** – Proportion of the available space this item should take up when there is extra space. Default of 0.
- **Flex Shrink** – Proportion of the available space this item should take up when there is not enough space. Default of 1.
- **Flex Basis** – The default size of the element before flexbox redistributes the space. Default of auto.

## Flex Property Syntax

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



# Flex Property Demos

The flex property is a shorthand property.

MDN has pages for each individual property with demos:

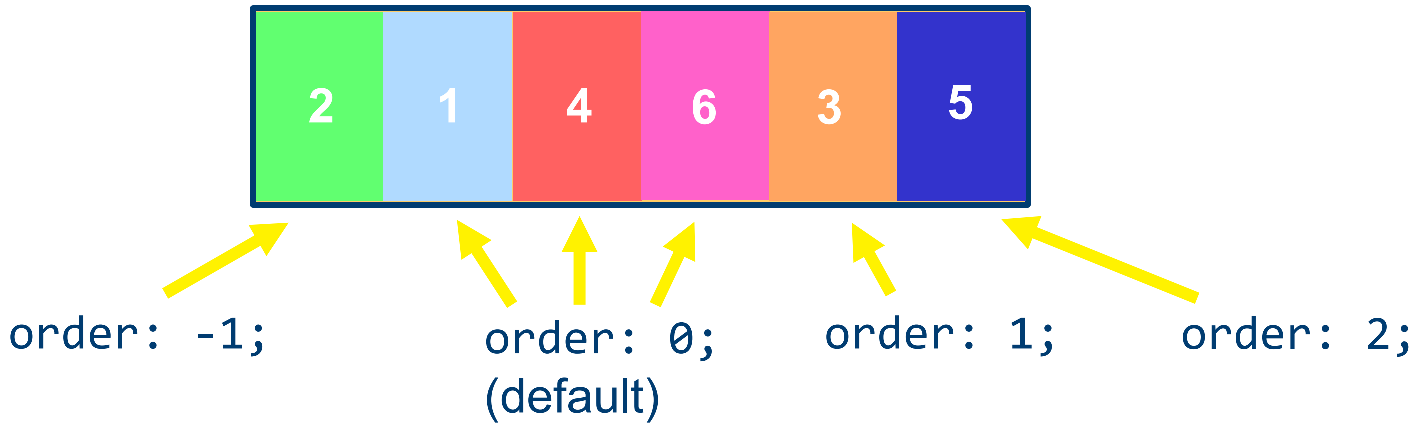
https://developer.mozilla.org/en-US/docs/Web/CSS/flex-grow

https://developer.mozilla.org/en-US/docs/Web/CSS/flex-shrink

https://developer.mozilla.org/en-US/docs/Web/CSS/flex-basis

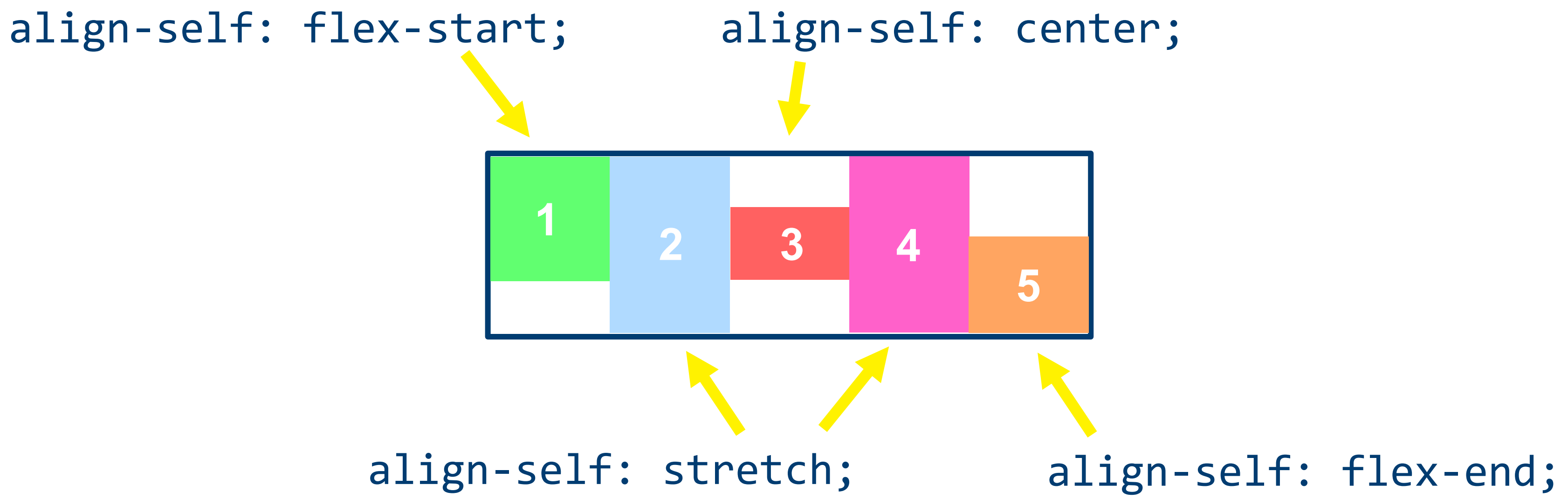
## Order Property

The order property is set on flex items to change the order the item appears relative to other flex items.



## Align Self Property

The align self property is set on flex items to override the default align items property.



# To <div> or to CSS…

Using a <div> or <span> to wrap elements for styling is common but often unnecessary. From our Testimonials code:

Get only direct children of the testimonials class…

…all elements…

…that are not <article> elements.

```
.testimonials > *:not(article) {
    width: 100%;
}
```

## CSS Selectors

This page on MDN has links to all of the CSS selectors:

https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Selectors

We will learn some of these as we go on in this course and in your Web Development 2 course.

## Resources – Flexbox

A Complete Guide to Flexbox
https://css-tricks.com/snippets/css/a-guide-to-flexbox/

Flexbox (MDN -- Web)
https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Flexible_Box_Layout

Flexbox (MDN -- Learn)
https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS_layout/Flexbox

## Flexbox Games & Tools

https://flexboxfroggy.com/

http://www.flexboxdefense.com/

https://geddski.teachable.com/p/flexbox-zombies

https://knightsoftheflexboxtable.com/

https://codingfantasy.com/games/flexboxadventure

https://the-echoplex.net/flexyboxes/

## Video Tutorials

LinkedIn Learning – Advanced Responsive Layouts with CSS Flexbox
https://www.linkedin.com/learning/advanced-responsive-layouts-with-css-flexbox/

LinkedIn Learning – CSS Essential Training: Introduction to Flexbox
https://www.linkedin.com/learning/css-essential-training-3/introduction-to-flexbox

## QUESTIONS & ANSWERS