

FWDP 1000 – Day 4

Course: Web Development 1

Instructor: Gabbie Bade

Morning Review

- Download the files.
- Open the HTML file and CSS file in your code editor.
- Read the comments in the files and complete the tasks.
- Compare today's services.html and services.css to your files for Assignment #3.

We will go over this in 10 minutes.

GitHub Practice

- Switch to **main**
- Create and push your **day-4** branch
- While on branch day-4, copy the files provided over to your repo directory
- Add, commit, push

Agenda

- Organizing CSS
- Responsive Design
- Media Queries
- Flexbox
- Assignment #4

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. You will cover this in Web Development 2.*

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 { }`

Example CSS Code

Look at **services.css** in the **day-04** folder to see...

- Sections are separated to be modular.
- Classes are only used to style specific elements.
- A maximum of three selectors are used.

CSS Naming Conventions

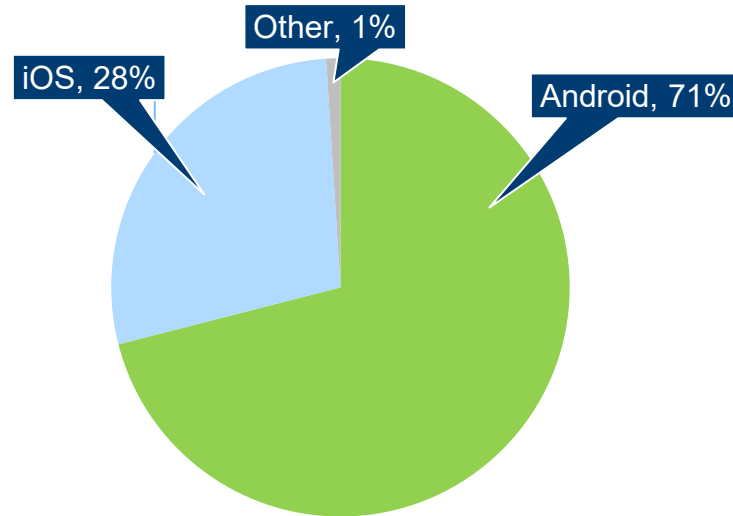
If you want to get fancy, these are commonly used:

- **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>

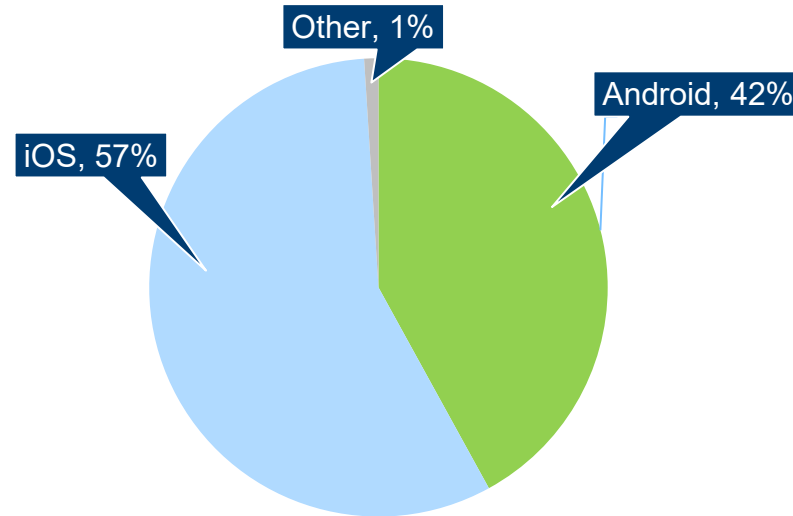
Responsive Design

Smartphone Market Share (Global)



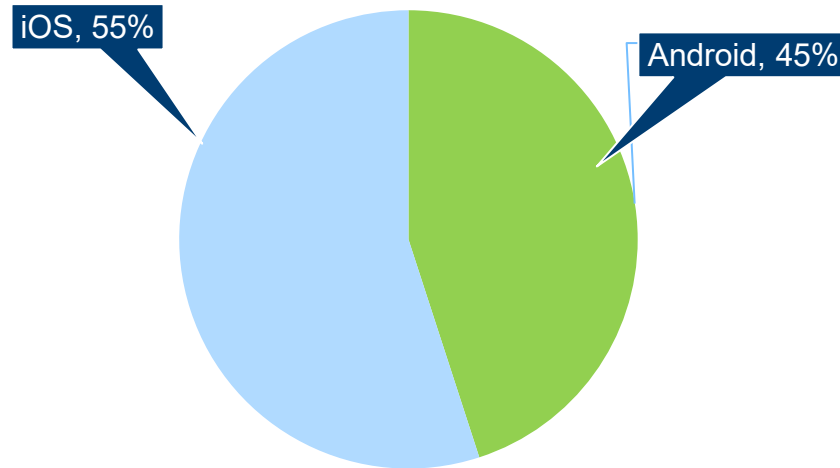
<https://gs.statcounter.com/os-market-share/mobile/worldwide>

Smartphone Market Share (Canada)



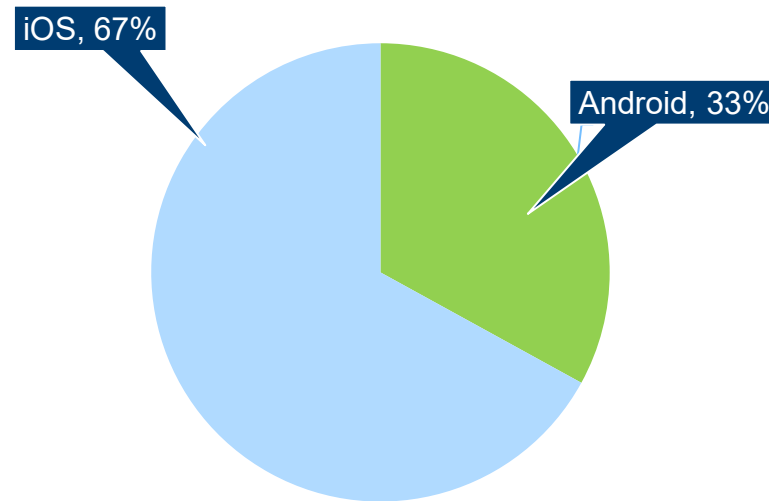
<https://gs.statcounter.com/os-market-share/mobile/canada>

Tablet Market Share (Global)



<https://gs.statcounter.com/os-market-share/tablet/worldwide>

Tablet Market Share (Canada)



<https://gs.statcounter.com/os-market-share/tablet/canada>

Responsive Design

Responsive Web Design (RWD) means creating a single website that adjusts according to the screen size, orientation and/or device.

Instead of making separate websites for mobile and desktop, we make one website that adjusts to the situation.

Some visual examples: <https://mediaqueri.es/>

Device Resolution vs Viewport Size

To understand pixel density, consider the iPhone 12...

- It has a device resolution of 1170px by 2532px.
- It has a viewport size of 390px by 844px.

Simply put... three pixels are being squished into one pixel on the screen to make the display crisper.

Viewport Size

For our CSS, we only care about the viewport size.

Using our iPhone 12 example, that means all of our content needs to fit in a space with a maximum width of 390px.

Device viewports and resolutions:

<https://experienceleague.adobe.com/docs/target/using/experiences/vec/mobile-viewports.html>

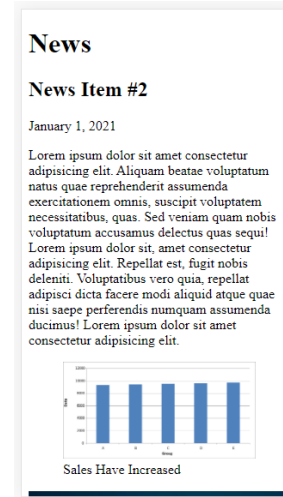
Viewport Size

There's just one problem... browsers don't use the device's viewport size by default when rendering our webpages.



This is the default way smartphones handle websites.

The same webpage with the proper HTML meta tag added.



Viewport Meta Tag

To use the device's viewport width, we add the following code to into the <head> element of all HTML files:

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

This tells the browser to use the device's viewport width when rendering the page.

This tells the browser to set the initial device zoom level to 1.

https://developer.mozilla.org/en-US/docs/Mozilla/Mobile/Viewport_meta_tag

Viewport Meta Tag

Place that code in the `<head>` section of every HTML file you create going forward.

Now we can write CSS for responsive designs!

Recap

Why is it important to keep your CSS organized?

- ❑ Easier to track CSS Inheritance, Cascade, Specificity
- ❑ Modular code prepares you for working with Sass
- ❑ So other developers don't hate working with you
- ❑ Chunks of code can be easily re-used on other projects

Recap

Responsive Web Design means creating...

- a) ...two sites: one for desktop and one for mobile.
- b) ...one site with different design and content for desktop and mobile.
- c) ...one site for all device sizes and changing the layout with CSS.

True or False: The CSS for larger screen sizes should be written first, then the CSS for smaller screen sizes.

Media Queries

Media Queries

Media queries are CSS conditional statements that run code only when the condition is true.

For example, a media query may say “run the following code if the viewport is 400px or wider”.

Media queries can be used to apply styles based on screen size, screen orientation, user preferences, print mode, and more.

Media Query Syntax

The most common way to write a media query is within your existing stylesheets.

```
@media (min-width: 400px) {  
  
    body {  
        background-color: red;  
    }  
  
}
```

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

Media Query Syntax

This tells the browser to only run the code within the curly brackets if the viewport width is 400px or higher.

```
@media (min-width: 400px) {  
  
    body {  
        background-color: red;  
    }  
  
}
```

These styles will only apply when the condition above is met.

Complex Media Queries

This media query has three conditions and they all must be true for the code inside to apply.

```
@media screen and (min-width: 25em) and (max-width: 3.125em) {  
  
    body { background-color: red; }  
  
}
```

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

Using ems in Media Queries

We use the **em** unit in our media query conditions for accessibility.

If the user has changed the text size in their browser, then our media queries will use that new base font value.

In a media query, the value of **em** will be the same as the value of **rem**.

- This is only true for media queries! Anywhere else in your CSS and the **em** value inherits from its ancestors so *may* be different than **rem**.

Calculating the em Value

The default value of **rem** and **em** for most browsers is 16px.

If you want a breakpoint at 800 pixels...

$$800 / 16 = 50$$

So your media query would be...

```
@media (min-width: 50em) { }
```

Writing Mobile First CSS

When you start writing your CSS, follow this process:

1. Resize the browser down to mobile screen size manually or by using your browser's Developer Tools (see Day 3 slides).
2. Write your general styles (for all screen sizes) and your mobile styles.
3. Resize the browser up until the layout breaks or looks bad and note the screen width.
4. Write a media query with that screen width and put your tablet and/or desktop styles inside the media query.
5. Repeat steps 3 and 4 until styling is complete.

Testing Screen Sizes

In addition to resizing your browser or using Developer Tools to test for different screen sizes...

It's always ideal to test on a real device (your phone, your tablet, etc.)

There are also services like Browser Stack:

<https://www.browserstack.com/>

Container Queries

A new CSS featured called a **Container Query** works like a Media Query but is based on an element's width instead of the entire document's width.

It has [~85% browser support](#) but you can practice it now to use it later.

<https://developer.mozilla.org/en-US/docs/Web/CSS/@container>

Flexbox

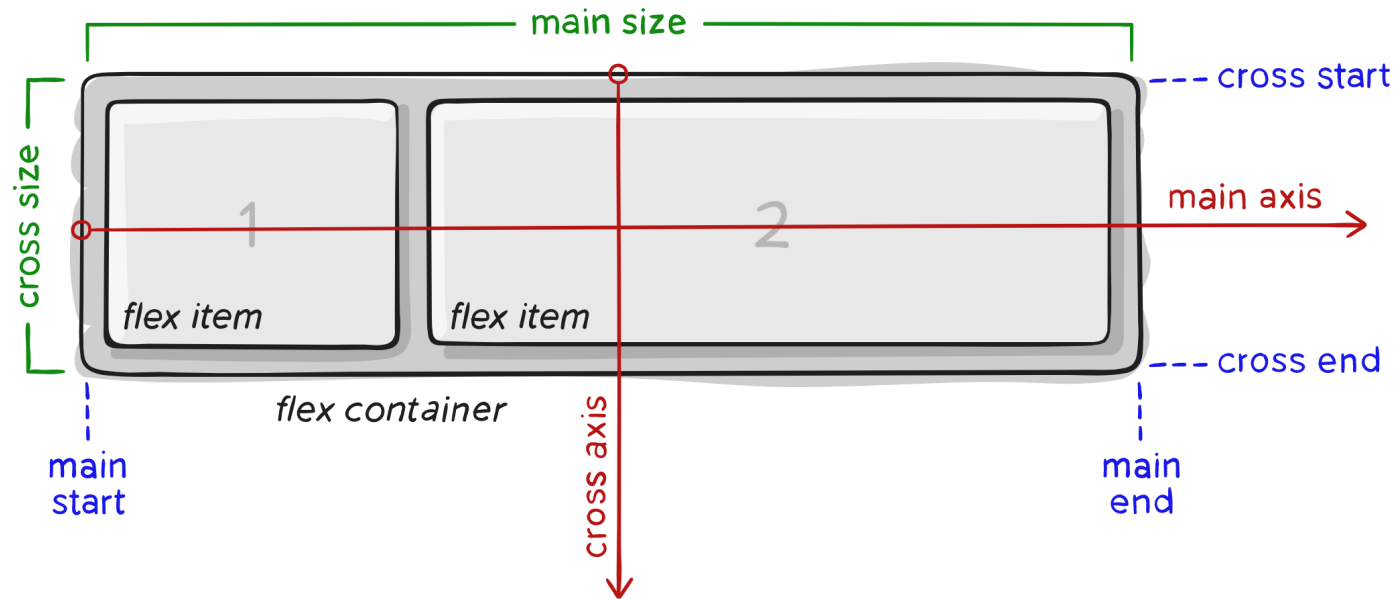
Flexbox

“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.

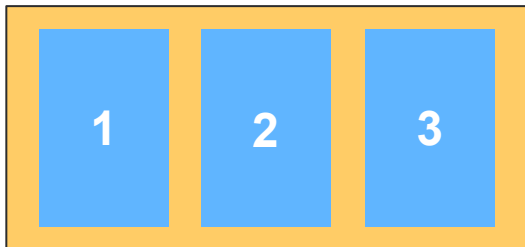


Flex Direction

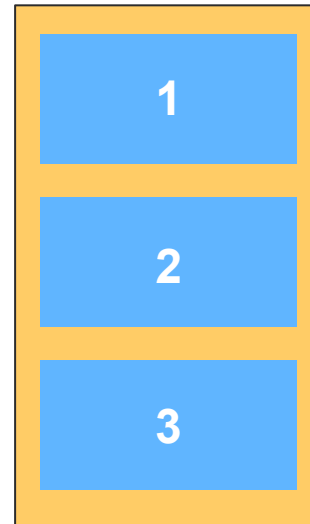
You can also display flex items in a column.

(default)

`flex-direction: row;`



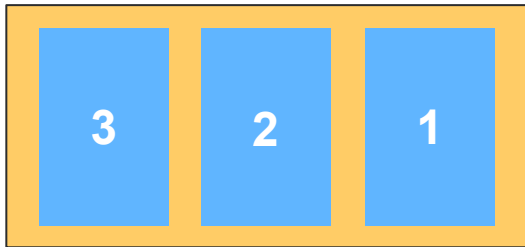
`flex-direction: column;`



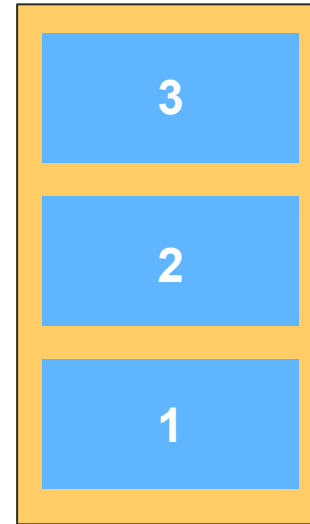
Flex Direction - Reverse

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

`flex-direction: row-reverse;`



`flex-direction: column-reverse;`

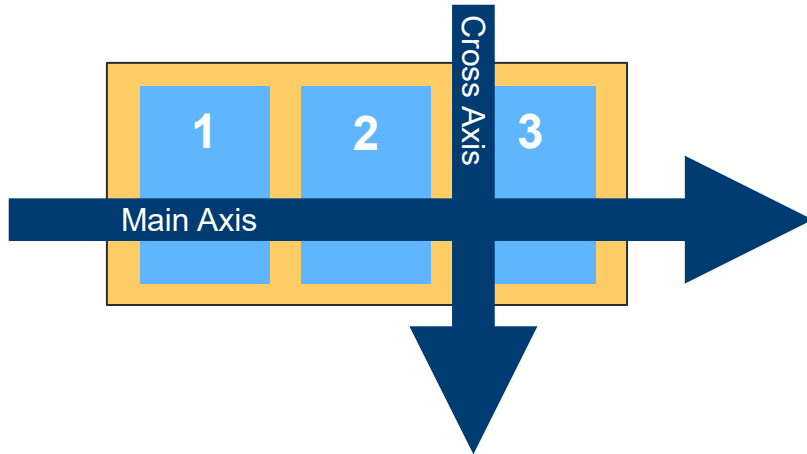


Main Axis and Cross Axis

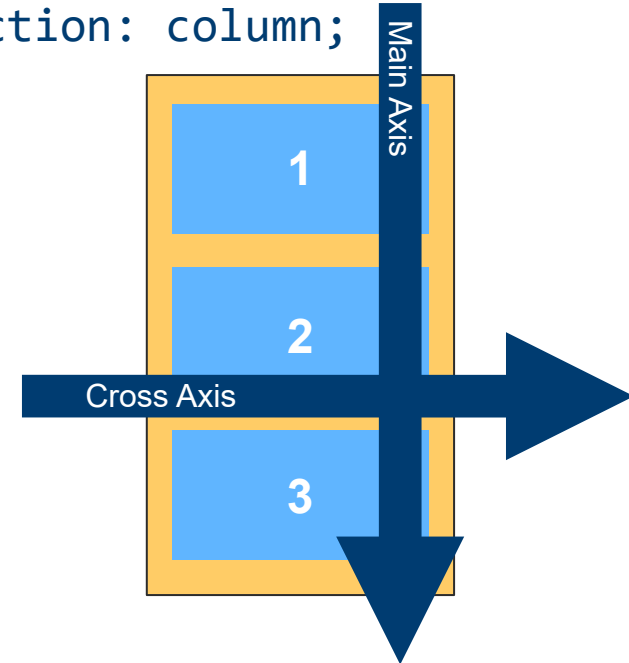
The flex direction determines the main and cross axis.

(default)

`flex-direction: row;`



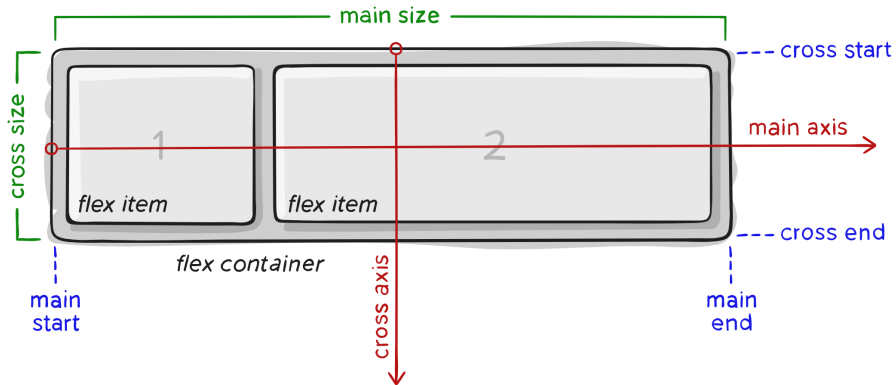
`flex-direction: column;`



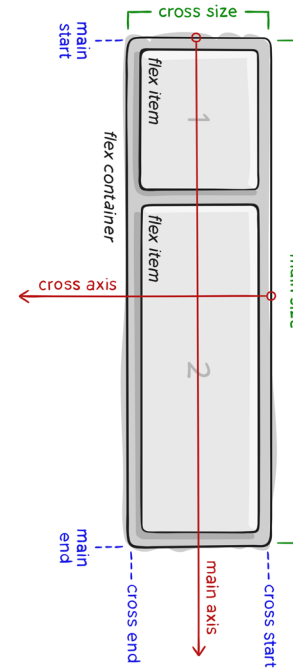
Main Axis and Cross Axis

(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...

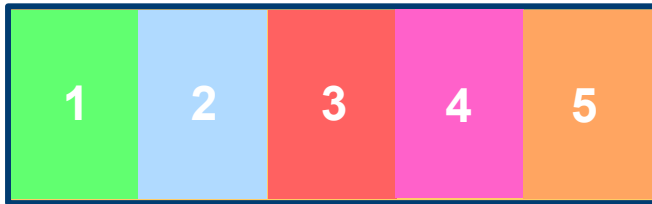
Flex Wrap

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

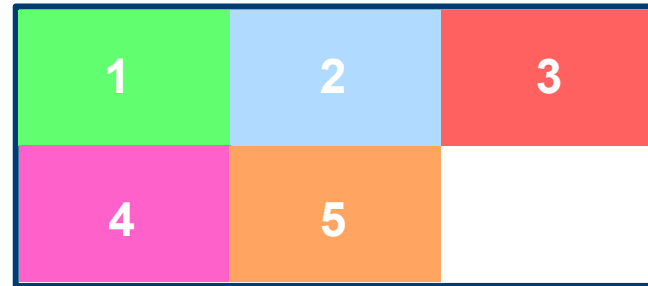
To allow multiple rows (or columns), change the flex wrap property.

(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`

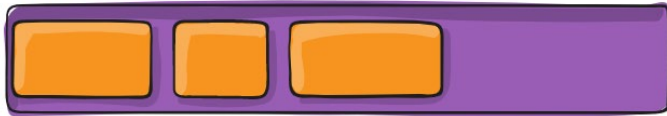
`space-between`

`space-around`

`space-evenly`

Justify Content Values

flex-start



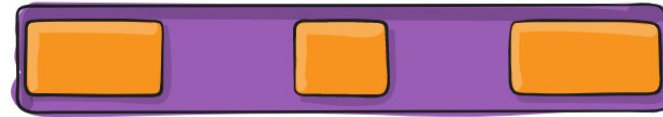
flex-end



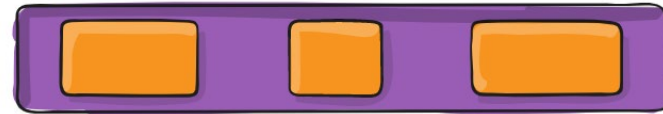
center



space-between



space-around



space-evenly



<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

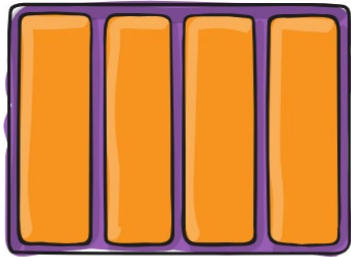
flex-end

center

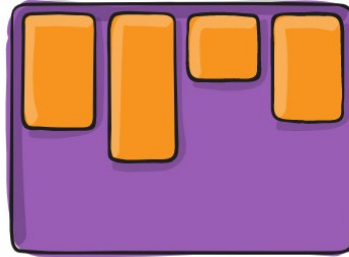
baseline

Align Items Values

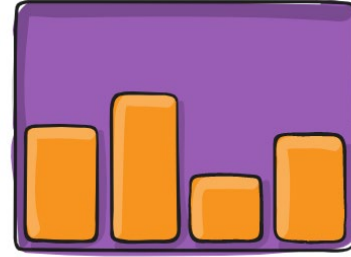
stretch



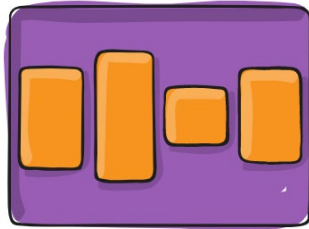
flex-start



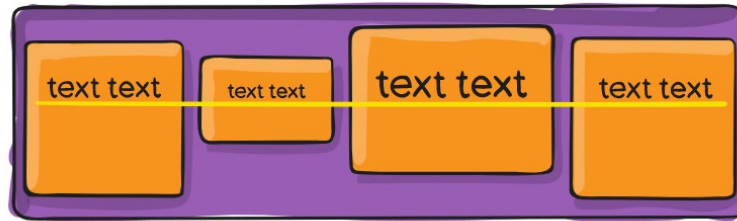
flex-end



center



baseline



<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`

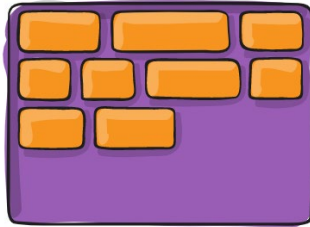
`space-between`

`space-around`

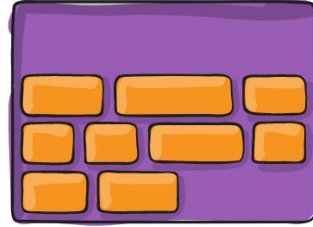
`space-evenly`

Align Content Values

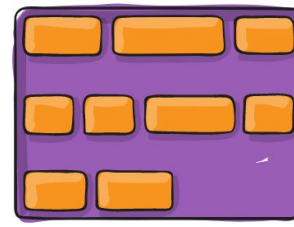
flex-start



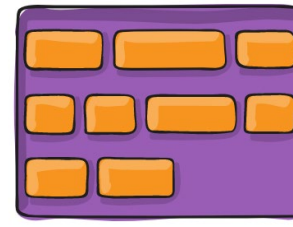
flex-end



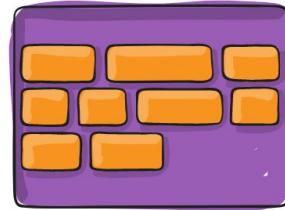
space-between



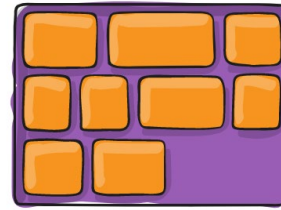
space-around



center



stretch



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

Gap

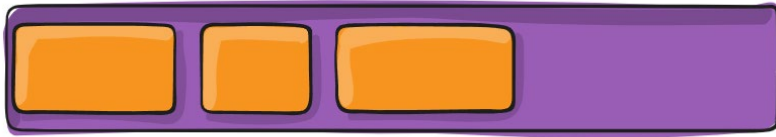
The **gap** property sets the space between flex items in a flex container.

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

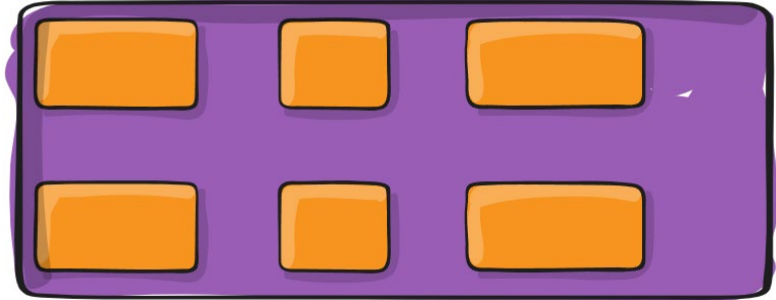
Note: It works similar to setting “margin” on all flex items.

Gap Examples

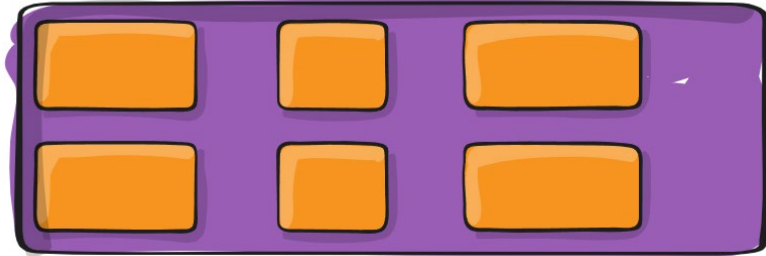
gap: 10px



gap: 30px



gap: 10px 30px



<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**:

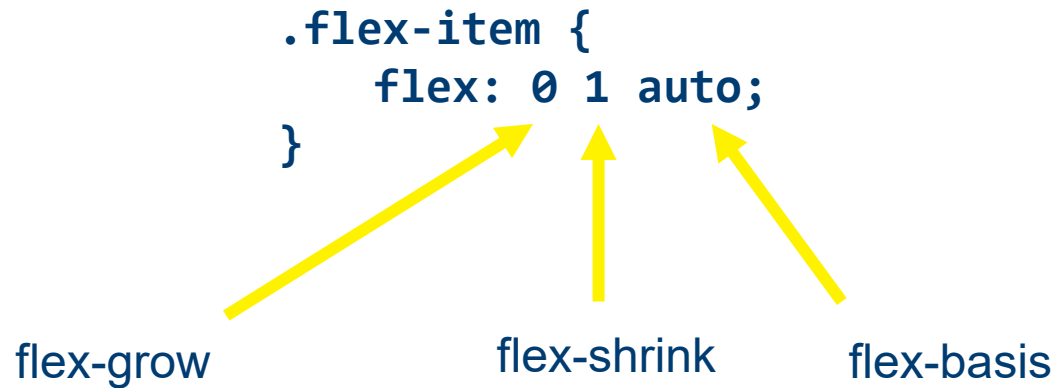
- flex
- align-self
- order

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 Property Demos

The flex property is a shorthand property.

MDN has pages for each individual property with demos:

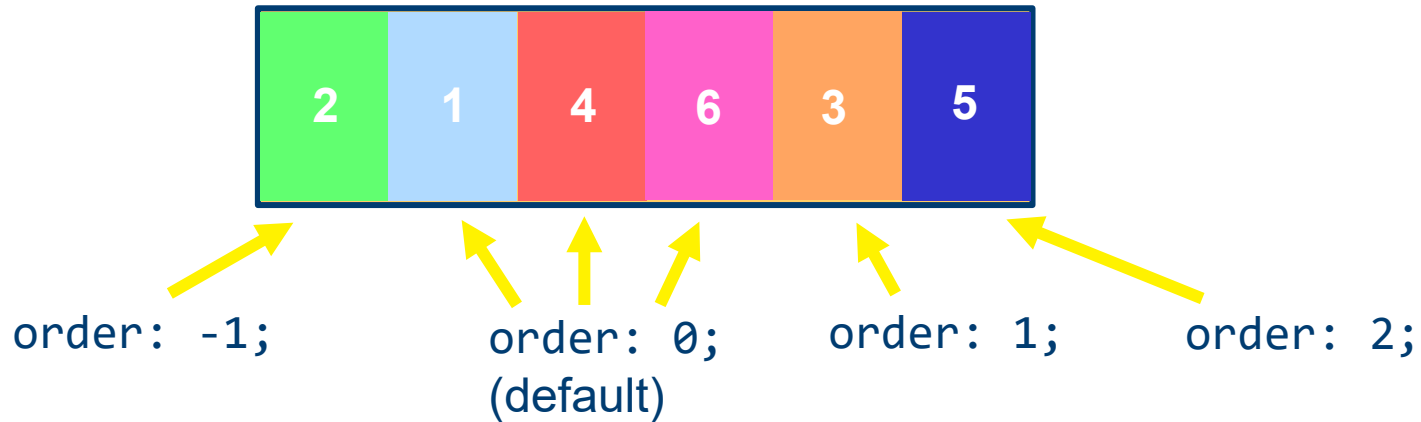
<https://developer.mozilla.org/en-US/docs/Web/CSS/flex-grow>

<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.

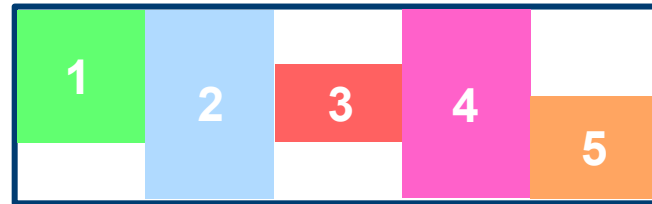


Align Self Property

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

`align-self: flex-start;`

`align-self: center;`



`align-self: stretch;`

`align-self: flex-end;`

Flexbox Practice

Let's use flexbox to style the Testimonials section on our Services page for tablets and desktops.

Happy Customers

Hear from our happy customers below or on [our Facebook page](#).



"They're a great company!"
Maxwell Zack



"I was satisfied with their work."
Clara Jones



"We were in a pinch and they
helped us out!"
Jane Doe

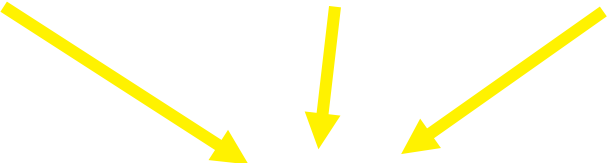
To <div> or to CSS...

Using a <div> or 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.

More Flexbox Practice

Open news.html and news.css from the day-04 folder.

We will finish the styling for the header and navigation.

Then we will style for the desktop layout of the webpage.

Assignment #4

Assignment #4

- Please refer to Assignment #4 in the Learning Hub.
- To submit the assignment, you can do **one** of these:
 - Have me check your assignment in class before 4pm.
 - Zip today's **folder** and upload it to the Learning Hub before next class.
- If you have questions or need guidance, just ask!

Working Together

You can work on and submit this assignment alone, in pairs, or in a group of 3.

I encourage you to work with classmates that you haven't worked with yet.

You will have multiple group projects throughout the program, so start to get to know each other now!

Practice Git & GitHub

This is not required but highly recommended.

Practice adding and committing until it becomes a habit.
Git/GitHub is incredibly helpful especially for more complex projects.

Practicing it even in simple projects make it less daunting!

Resources – Media Queries

Media Queries (MDN -- Web)

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

Media Queries (MDN -- Learn)

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

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