Here's the link to today's lecture.

Review

The Box Model!

You've got the hang of this now, I don't need to harp on about it :)

Flex Box Properties

Declare our flexbox parent! `display: flex`

Parent Properties

- flex-direction: arranges elements vertically or horizontally
- justify-content: arranges elements along the main axis
- align-items: arranges elements along the cross axis
- flex-wrap: if many elements, determines whether or not multiple lines should be created

Child Properties

- align-self: allows you to align a specific item
- flex-grow: determines if the child element can grow larger
- flex-shrink: determines if an element can shrink, if needed
- flex -lets you specify how much space to take up

Advanced Selectors

So far, we've been selecting elements by: - Tag Name - Class - ID

How do you select the above three selectors?

CSS also gives us advanced selectors that we can use to get even more specific about element selection by using their relationships to each other.

- Parent
- Child
- Sibling

Stacking Selectors

Stacking selectors let you apply the same styles to different tags or classes in the same CSS block.

To stack, we separate with a comma:

```
.child-a,
.child-b,
.child-c,
.child-d {
  height: 50px;
  width: 50px;
}
```

Adjacent Sibling

The adjacent sibling combinator (+) separates two selectors and matches the second element only if it immediately follows the first element, and both are children of the same parent element.

This will change the background color of just the second div

```
div + div {
  background-color: deepskyblue;
}
```

General Sibling Selector

This will let you select all of the siblings that follow an element.

Select the `h2` tag that follows any `p` tag:

```
p ~ h2 {
  color: red;
}
```

Direct Descendant Selector

This selector allows you to target elements that are direct descendants of that parent. To target direct descendants, use the > selector.

```
div > p {
  margin: 0;
}
```

Pseudo Selectors

Pseudo selectors let us style elements when they are in a specific state, like when your mouse is hovering over it or you clicked it.

This pseudo-selector is used to style elements when the mouse “hovers” over it.

```
section:hover {
  background-color: green;
}
```

First Of Type

This will select the first element of a type

```
div:first-of-type {
  background: yellow;
}
```

Nth of Type

“nth-of-type” lets you select an element based on its type and order.

For example, if we wanted to only style the second div in our child elements.

To use, target the element you want to style followed by “:nth-of type()”.

Inside the brackets you can put which element of that type you want to style (either a number or even / odd)

Selecting the second divs:

```
div:nth-of-type(2) {
  background-color: navy;
}
```

First Child

Styles the element only if it is the first child of the selected parent.

To use this selector, target the parent element, then the child you’d like to style followed by “:first-child”.

```
div p:first-child {
  font-size: 40px;
}
```

Last Child

Styles the last child of the selected parent.

The syntax of this selector is the same as it's opposite, ":first-child"

```
section div:last-child {
  color: seagreen;
}
```

Pseudo Elements

While pseudo-selectors style specific element states, pseudo elements style specific parts of the element itself.

```
p::selection {
  color: orange;
}
```

First Letter

Will style the first letter contained in the element.

```
p::first-letter {
  text-transform: lowercase;
  font-size: 40px;
  color: yellow;
}
```

CSS Transitions

When the value of a CSS property changes we can use CSS transitions to animate how the property will change.

There are four different properties we need to use for a CSS transition.

Transition Property

This is the property we are going to animate. The animation happens when the property is changed. For example, when you "hover" over a "p" tag, transition the background-color property. The hover state would be defined in the same way we learned earlier.

First add a hover:

```
.parent:hover {
  background-color: lawngreen;
}
```

This code will target the background-color property, but it wont work alone.

```
.parent {
  transition-property: background-color;
}
```

Transition Delay

This defines how long the transition will wait after being triggered before it starts.

Add this property to ".parent". Now it's delayed!

```
transition-delay: 1s;
```

Transition Duration

This property defines how long a transition will take to complete once it starts.

```
transition-duration: 2s;
```

Transition Timing Function

The timing function dictates the speed of the transition over time. Should it start out really slow and then speed up, or evenly change

throughout the duration?

It accepts these properties:

- ease-in:(default) transition will start out slow before picking up speed near the end of the duration.
- linear: The speed of this transition will be the same throughout the duration.
- ease-in-out: The transition will be slow at the start and at the end.

```
transition-timing-function: ease-in;
```

```
transition-timing-function: linear;
```

```
transition-timing-function: ease-in-out;
```

Bonus Extras

CSS Transitions are cool. Here are two examples of some pretty nifty CSS stuff:

- CSS Zen Garden
- CSS Animations