Intro to Front End - Advanced CSS

Set up the Live Stream

Front End Fundamentals

- How are we feeling about Flexbox?

Objectives

- Advanced Selectors
- Pseudo-selectors
- CSS Transitions

Box Model Review

The Box Model: each element is made up of margin, border, padding and content.

Each contribute to the overall size and appearance of the element.

We can manipulate each using CSS to customize webpages.

Review fiftythree.com for a good example

Flexbox Review

`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

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

In this example, the header, section and article tags would all have padding of 20px.

Live Code

Add this to the CodePen

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

Now all of the child elements have a height of 80px.

We could then define the height and width for all of them instead of individually.

Lets give our text some color:

`color: #FFFFFF;`

Now let's select all elements and align text center:

```
section,
div,
.child-a,
.child-b,
.child-c,
.child-d {
  text-align: center;
}
```

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.

In the example above, only the second paragraph would be styled.

Live Code

```
<section>
  <div>
    <p>I'm a paragraph tag</p>
    <p>I'm a paragraph too</p>
  </div>

  <div>
    <p>I'm a paragraph</p>
    <h2>I'm an h2</h2>
    <p>I am a paragraph too</p>
  </div>
</section>
```

Add this CSS

This will change the background color of *just* the second div

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

Which paragraph is going to turn pink?

```
p + p {
  color: pink;
}
```

General Sibling Selector

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

In this example all of the `p` tags that come after `h1` tags would contain blue text.

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

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

Select only the second “section” tag:

```
div ~ div {
  background-color: seagreen;
}
```

You can also use these selectors together:

```
p ~ h2,
```

```
p + p {
  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.

In this example, all the `h2` elements within a `main` will have font color blue.

This is an example of a direct descendant selector that will not work. Can anyone tell me why?

Live Code

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

Then the correct CSS:

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

You can also combine selectors:

```
div > p,
div > h2 {
  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.

Pseudo Selector Syntax

To use pseudo-selectors, first use a selector (tag, class, id, advanced), then a colon followed by the type of pseudo-selector you want to use.

Hover

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

Live Code

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

First Of Type

This will select the first element of a type

Live Code

```
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:

Live Code

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

Target all odd numbered divs:

```
div:nth-of-type(odd) {
  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".

Live Code
Style the first child under the ".parent" class:

```
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"

Live Code

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

Pseudo Elements

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

Selection

Live Code

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

First Letter

Will style the first letter contained in the element.

Live Code

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

CSS Transitions *NEW CODE PEN*

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

We will go over the four different properties we use to control transitions.

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.

Live Code

In our example, we have a hover state set on both sections.
We are going to create a transition the background-color for the “.parent” section tag only

```
<!DOCTYPE html>
<html>
  <head>
    <title>My Awesome Web Page</title>
    <link href="./index.css" rel="stylesheet" />
  </head>
  <body>
    <section class="parent">
      <div class="child-a">A</div>
      <div class="child-b">B</div>
    </section>
    <section>
      <div class="child-c">C</div>
      <div class="child-d">D</div>
    </section>
  </body>
</html>
```

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.

In this example, once the element was hovered over, the background colour would start to change after a delay of 1.5 seconds.

Live Code

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.

Live Code

```
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.

Live Code

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

`transition-timing-function: linear;`

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

Shorthand

As with most things in CSS, there is a shortcut!

To condense all these transitions into one, use the property “transition” then the values property, duration, timing-function and delay.

This is a great way to write fewer lines of code!

Live Code

Condense the 4 properties we just set into one “transition” shortcut.

`transition: background-color 2s ease-in-out 1s;`

Reminders

- Record Attendance
- Upload lecture
- Send out Notes
- Log hours on Clocktower