JavaScript: DOM Manipulation

What is DOM manipulation?

- Remember that DOM stands for **Document Object Model**
- The DOM is a representation of the HTML structure on the page that JavaScripts can interact with
- DOM manipulation simply means changing an element's text or inner html, or perhaps replacing it entirely

The HTML `<form>` element

- Most forms you see online (login, signup) all share a common tag: `<form>`!
- Inside of `<form>` are several elements that make up forms: text input boxes, dropdowns, radio buttons, checkboxes, etc.
- Today, we'll just be using the text input and button elements, but in future classes you'll learn about all of them!

<form> example

```
<form>
  <label>
    <span>Username</span>
    <input type="text" name="username"/>
  </label>

  <label>
    <span>Password</span>
    <input type="password" name="password"/>
  </label>

  <button type="submit">Submit!</button>
</form>
```

There are many different types of inputs, don't worry about them for now.

Retrieving elements

- Now that we have our form, how do we reference the elements in javascript?
- The globally available `document` object has many functions for this:
  - `getElementById(string)` - Get one and only one element by the `id` attribute
  - `getElementsByClassName(string)` - Get a list of elements that all have a class
  - `querySelectorAll(string)` - CSS style selector, returns a list of everything it matches

Retrieving elements (examples)

For this element:

```
<form>
  <input type="text" name="username" class="form-username" id="my-username"/>
</form>
```

These would all work:

```
document.getElementById("my-username");
document.getElementsByClassName("form-username")[0];
document.querySelectorAll("#my-username")[0];
document.querySelectorAll(".form-username")[0];
document.querySelectorAll("input")[0];
```

Accessing elements

- Imagine the previous `<form>` had an `<h1>` tag above it that has the form title
- We can use the attributes `.innerText` and `.innerHTML` to read or change the title

```
<h1 id="title">
  Enter your <strong>information</strong>
</h1>
```

```
var heading = document.getElementById('title');
console.log(heading.innerText); // Prints "Enter your information"
console.log(heading.innerHTML); // Prints "Enter your <strong>information</strong>"

var name = "Will";

// Can only change text
heading.innerText = "Enter " + name + "'s information";
// Can change text AND add new html elements
heading.innerHTML = "Enter " + name + "'s <strong>information</strong>";
```

Accessing multiple elements

```
<ul class="food">
  <li class="food-item">Avocados</li>
  <li class="food-item">Spinach</li>
  <li class="food-item">Broccoli</li>
</ul>
```

```
var foodItems = document.getElementsByClassName("food-item");
console.log(foodItems); // HTMLCollection[3], behaves same as an array
console.log(foodItems[1]); // <li class="foods-item">Spinach</li>

for (var i = 0; i < foodItems.length; i++) {
  foodItems[i].innerHTML = "PIZZA";
}

foodItems[foodItems.length - 1].innerHTML = "AND MORE PIZZA";
```

Exercise

- Create a page that has a parent element that contains many child elements, and a header on top.
- Each child element should have some of text in it. It doesn't matter what.
- Using javascript, select the child elements and add its index to the text. I.e. `<li>Taco</li>` should become `<li>0: Taco</li>` if it is the first element.
- Finally, again in js, add the number of children to the title text. I.e. `<h1>Movies</h1>` becomes `<h1>10 Movies</h1>` if there are 10 items.
- Try adding and removing elements to make sure it still works.

Resources

Css Tricks

CSS Tricks - DOM Explanation

Helpful Blog Posts

The Basics of Javascript DOM Manipulation by CallMeNick