Class 7 Notes

Tables

In addition to , , <h1>, <a>, <img>, etc, an important set of tags used to represent tabular data is the <table> element. Before the days of flexbox and grid and floats this was also often used for layouts - do not let me catch you using tables to help layout your page!

Here is the file we built to demonstrate tables. Please look at the new tags introduced and understand them. If you do not, please look them up or ask me: <table> <thead> <tbody> <tr> <th> and <td>

```
    <style>
      table {
        border-collapse: collapse;
      }
      tr:nth-child(odd) {
        background: #000;
        color:#FFF;
      }
    </style>

  </head>

  <body>

    <h1>Tables</h1>
    <!-- This is a comment -->
    <table border="1">
      <thead>
        <tr>
          <th>Bird</th>
          <th>Length</th>
        </tr>
      </thead>
      <tbody>
        <tr>
          <td>Parrot</td>
          <td>6</td>
        </tr>
        <tr>
          <td>Finch</td>
          <td>5</td>
        </tr>
      </tbody>
    </table>
```

&& and ||

These are two operators that represent AND **&&** and OR **||** - These are very important for checking if multiple conditions are true.

Try these lines of javascript out in a new file or your console. Again, if you do not understand, research them or ask.

```
const thereAreDogs = true;
const thereAreCats = false;

// && means AND
// || means OR
if (thereAreDogs && thereAreCats) {
  console.log("There are Dogs and cats!");
} else if (thereAreDogs) {
  console.log("Just Dogs");
} else if (thereAreCats) {
  console.log("Just Cats")
} else {
  console.log("No animals");
}

if (thereAreDogs || thereAreCats) {
  console.log("There are Dogs OR cats!");
}
```

String Interpolation

This was introduced in an earlier class, but just to reiterate, “String Interpolation” is a fancy new way of concatenating strings that will help our code look cleaner. It is optional.

```
const amount = 10;
// const myVar = "There are" + amount + " dogs."
// is the same as: (notice we can wrap the variable in ${})
const myVar = `There are ${amount} dogs`;
```

APIs

The main goal for this class is to learn about APIs. These are Application Protocol Interfaces that allows us to query an endpoint and get data back from a service. A few of the examples we go through will be pulling from a pokemon API, the class API, and the NY Times Article API.

To start we'll use the pokemon API. Create a new file called pokemon.html. Make sure you bring in your normalize.css and jQuery. To start we want on <h1> tag with an id on it.

```
<!DOCTYPE html>
<html>

  <head>

    <link href="../normalize.css" rel="stylesheet" />
    <script src="https://code.jquery.com/jquery-2.2.4.min.js"
integrity="sha256-BbhdlvQf/xTY9gja0Dq3HiwQF8LaCRTXxZKRutelT44="
crossorigin="anonymous"></script>

  </head>

  <body>
    <h1 id="name"></h1>
  </body>

</html>
```

Now if we look at the bottom of the jQuery homepage (https://jquery.com/) you will notice a section on AJAX. This is an abstraction on the more complex javascript that's required to query an external web address. Let's start by creating a constant with a pokemon id:

```
    <script>
      $(document).ready(function(){

        const pokeId = 138;

      });
    </script>
```

Now bring in the AJAX

```
<script>
  $(document).ready(function(){

    const pokeId = 138;

    const url = "https://pokeapi.co/api/v2/pokemon/";
    $.ajax({
      url: url + pokeId,
      success: function(data){
        console.log(data);
      }
    })

  });
</script>
```

Open up your page in a browser and pull open the console. It may take some time (this is actually a pretty slow API) but you should see data about that pokemon in your console.

Next we can populate our page with some of that data. Notice the forEach loop that is used to loop over an array of types. This is a little different than what we've seen previously (for loops) but it can be a more concise way of achieving the same thing.

```
      success: function(data){
        console.log(data);
        $("#name").text(data.name);

        const imgURL = data.sprites.front_shiny;
        $("body").append(`<img src="${imgURL}" />`);

        data.types.forEach(function(type){
          console.log(type.type.name);
          $("body").append(`<div>${type.type.name}</div>`);
        });

      }
```

Go ahead and try refreshing your page. After your pokemon data loads it should be displayed in the browser. Again, be patient and give the API 10-15 seconds to respond before thinking something is wrong.

The next two tutorials will go over the class API and the NY Times Article API.

API Tutorial

Head here for the API tutorial:
https://docs.google.com/document/d/1ps6cMPpGC8EhKIyQ3LjO0sNWxl-Ueqg1T4VKWpUv3C4/edit?usp=sharing

Advanced API Tutorial

Head here for advanced APIs:
https://docs.google.com/document/d/1nfvCMUPfaIE8KSnWQTT0wPzBCS4OFQsdeB6s-2Wm41s/edit?usp=sharing

Homework

- Finish the Advanced API Tutorial
- Finish your low fidelity wireframes for your final project. If you haven't told me what your final project topic is, have that prepared for class on Thursday.