PROFESSIONAL & CONTINUING EDUCATION
UNIVERSITY *of* WASHINGTON

LESSON 3: JSON & Constructors

HTML 300

OVERVIEW

1. Assignment Review
2. JS Objects Review
3. Intro to Constructors
4. JavaScript Object Methods
5. JSON Overview
6. Working with JSON

PROFESSIONAL & CONTINUING EDUCATION
UNIVERSITY *of* WASHINGTON

Assignment Review

ASSIGNMENT REVIEW

- You are now all Sass experts!
- Show off the component that you built

PROFESSIONAL & CONTINUING EDUCATION
UNIVERSITY *of* WASHINGTON

JS Objects Review

JAVASCRIPT OBJECTS

- What are objects?
  - An object is a collection of related data and/or functionality, usually consisting of several variables and functions
  - These variables and functions are called properties (var) and methods (functions) when they are within objects
  - The data within an object can be any data type, including additional objects

```
const person1 = {
    name: 'Bob',
    age: 25,
    avg-height: ‘6ft',
    siblings: ['Joe', 'Sarah', 'Tom'],
    greeting: function() {
      console.log(`Hello I am ${this.name}`);
    }
};
```

JAVASCRIPT OBJECTS

- Creating Objects
  - Create an object by assigning a variable to an object, with or without additional properties or methods (can just be a blank object)
  - Using Object.create(). This method is typically best suited for creating objects based on a prototype object instead of a constructor.
  - Setting the value passed into Object.create() to null will create a blank object
  - Constructor functions can also create objects

```
const obj = {};
const obj2 = Object.create(null);
// obj = {}
// obj2 = {}
```

JAVASCRIPT OBJECTS

- Accessing Properties
  - Properties can accessed in two methods:
  - Dot Notation - access property via using a “.” and the property’s key name
  - Bracket Notation - access property via using [“key”] and the property’s key name – works with keys that have spaces/characters

```
const name = person1.name;
const age = person1['age'];
const height = person1['avg-height'];
```

JAVASCRIPT OBJECTS

- Setting Properties
  - Properties can set in the object initialization or after the object has been created
  - Access the property via the object and set the new value

```
const person1 = {
   name: 'Bob',
   favoriteMovie: 'Donnie Darko'
};

console.log(person1.favoriteMovie);
// Donnie Darko

person1.favoriteMovie = 'Space Jam';
console.log(person1.favoriteMovie);
// Space Jam
```

JAVASCRIPT OBJECTS

- Methods
  - Methods can be defined in two ways:
    - myMethod: function (params)
    - myOtherMethod(params)
  - Methods are accessed just like properties, and can take in parameters as needed
  - Methods are accessed as functions like object.greeting()

```
const myObj = {
   myMethod: function (params) {
   // ...do something
   }
    myOtherMethod(params) {
   // ...do something else
   }
};
```

JAVASCRIPT OBJECTS

- this
  - this is a difficult concept in JavaScript
  - Typically, this is referring to the object or function scope that is being referenced via the object properties
  - In general, this refers to the calling object in a method

```
const person1 = {
   name: 'Bob',
   greeting: function() {
      console.log(`Hello I am ${this.name}`);
   }
};

person1.greeting();
// Hello I am Bob
```

PROFESSIONAL & CONTINUING EDUCATION
UNIVERSITY *of* WASHINGTON

Intro to Constructors

INTRO TO CONSTRUCTORS

- Constructors
  - Constructor functions are a way in JavaScript to create many objects that all share similar properties and methods but not necessarily the same values
  - Constructors tend to follow the convention of function Name() for the constructor, and new Name() for the instance of the object
  - Constructors are great for collections with related properties or methods

```
function Car(color, make, model, year) {
   this.color = color;
   this.make = make;
   this.model = model;
   this.year = year;
}

const myCar = new Car('Black', 'Chevy', 'Cruze Hatch', 2018);
console.log(myCar);
// Car {color: "black", make: "Chevy", model: "Cruze Hatch", year: 2018}
```

INTRO TO CONSTRUCTORS

- When to use constructors?
  - Knowing you will need to create at least 3 objects that share properties or methods in a collection is a good time to think about creating a constructor function
  - If you are looking to bind methods to the object and access the this keyword across the collection
  - Using a front-end framework like React will utilize constructors using the ES6+ class syntax

INTRO TO CONSTRUCTORS

- ES6+ Class
  - The class keyword gives a new syntax for defining constructors
  - The constructor function is within the class declaration
  - Methods are defined on the class in a slightly different syntax

```
class Car {
  constructor(color, make, model, year) {
    this.color = color;
    this.make = make;
    this.model = model;
    this.year = year;
  }
  carAge() {
    let curr = new Date().getFullYear();
    console.log(curr - this.year);
  }
}

const myCar = new Car('Black', 'Cadillac', 'Escalade', 2010);
myCar.carAge();
// 9
```

PROFESSIONAL & CONTINUING EDUCATION
UNIVERSITY *of* WASHINGTON

Constructor Activity

CONSTRUCTORS IN PRACTICE

- Let’s give it a try
  - Clone the activity repo on the UW Front-End Cert
  - Create a constructor function for dogs
  - This function should have properties for the dog’s name, age, breed, color, bark level, energy level, and a pat method.
  - The name, breed, and color can be strings the levels are numbers
  - In the pat method, increase the bark and energy levels by one each pat
  - Log out to the console a message when pat is invoked, telling us the dog’s name and current bark/energy levels.

PROFESSIONAL & CONTINUING EDUCATION
UNIVERSITY *of* WASHINGTON

JavaScript Object Methods

JAVASCRIPT OBJECT METHODS

- for…in()
  - For…in will loop over an object’s enumerable properties
  - For…in is a specific loop for objects
  - This is a good method for finding a value when not working with arrays of data but storing as a key-value pair in an object

```
const obj = { prop1: 1, prop2: 2, prop3: 3 };

for (const prop in obj) {
    console.log(`obj.${prop} = ${obj[prop]}`);
}

// "obj.prop1 = 1"
// "obj.prop2 = 2"
// "obj.prop3 = 3"
```

JAVASCRIPT OBJECT METHODS

- Object.keys()
  - Object.keys will loop over an object’s enumerable properties and return an array of keys as strings
  - This acts as the opposite method as for…in
  - This is a good method for finding a value when not working with arrays of data but storing as a key-value pair in an object

```
const obj = { prop1: 'string', prop2: 2, prop3: false };

console.log(Object.keys(obj));

// ['prop1', 'prop2', 'prop3']
```

JAVASCRIPT OBJECT METHODS

- Object.getOwnPropertyNames()
  - Object. getOwnPropertyNames will loop over both an object’s enumerable properties non-enumerable properties and return an array of keys as strings
  - This acts as the essentially the same as Object.keys but will also pick up non-enumerable properties

```
const obj = { prop1: 'string', prop2: 2, prop3: false };

console.log(Object.getOwnPropertyNames(obj));

// ['prop1', 'prop2', 'prop3']
```

JAVASCRIPT OBJECT METHODS

- Object.values()
  - Object.values will loop over an object’s enumerable properties and return an array of values as strings
  - This acts as very similar to the for…in method
  - ES6+

```
const obj = { prop1: 'string', prop2: 2, prop3: false };

console.log(Object.values(obj));

// [‘string’, 2, false]
```

JAVASCRIPT OBJECT METHODS

- Object.entries()
  - Object. entries will loop over an object’s enumerable properties and return an array of key/value pairs as arrays
  - ES6+ browser support

```
const obj = { prop1: 'string', prop2: 2, prop3: false };

console.log(Object.entries(obj));

// [['prop1', "string"], ['prop2', 2], ['prop3', false]]
```

PROFESSIONAL & CONTINUING EDUCATION
UNIVERSITY *of* WASHINGTON

JSON Overview

JSON OVERVIEW

- What is JSON?
  - JSON stands for JavaScript Object Notation
  - JSON came from a time when XML reigned supreme
  - JSON was meant to allow APIs to have a standardized structure for returning data
  - It is meant to be fairly easily human-readable
- JSON is comprised of objects and arrays
  - These ubiquitous data structures allow JSON to work across many programming languages

JSON OVERVIEW

- When do we use JSON?
  - Typically APIs on the web will return their data in JSON format (or with an option to return as JSON)
  - The data that is fetched needs to be parsed into “real” JavaScript arrays/objects – typically done with the JSON.stringify() method
  - Turning the data into JavaScript usable JSON allows us to pipe the data into something like a class or constructor to template out the data. By leveraging the data/view split, the workload is reduced by not repeating our work when unnecessary .

PROFESSIONAL & CONTINUING EDUCATION
UNIVERSITY *of* WASHINGTON

Working with JSON

WORKING WITH JSON

- How do we use JSON to template data?
  - APIs generally have endpoints that allow access to various depths of the data nodes so you can pinpoint the data set you’re looking to use.
  - APIs will return an array of objects. We can use the array methods we learned in lesson 1 to loop through each object and do something.
  - Using a combination of our constructor/class, string template literals, and array methods we can loop through data, apply our HTML template with injected data, and display it on the page.

WORKING WITH JSON

- The 'cars' array represents some local JSON data
- Looping over the array with a method
- Using map() to apply a template string

```
const cars = [
    {
        "make": 'Ford',
        "model": 'Mustang',
        "year": 2010,
        "color": 'black',
    },
    {
        "make": 'Chevy',
        "model": 'Corvette',
        "year": 1984,
        "color": 'red',
    },
    {
        "make": 'Jeep',
        "model": 'Wrangler',
        "year": 1999,
        "color": 'silver',
    },
];
```

```
cars.map(function(el) {
    let car = `
        <article class="car">
          <ul>
            <li class="car__make">${el.make}</li>
            <li class="car__model">${el.model}</li>
            <li class="car__details">Color: ${el.color}</li>
            <li class="car__details">Year: ${el.year}</li>
          </ul>
        </article>
    `;
});
```

JSON OVERVIEW

- The output for each iteration of the previous loop

```
<article class="car">
  <ul>
    <li class="car__make">Ford</li>
    <li class="car__model">Mustang</li>
    <li class="car__details">Color: black</li>
    <li class="car__details">Year: 2010</li>
  </ul>
</article>
<article class="car">
 <ul>
    <li class="car__make">Chevy</li>
    <li class="car__model">Corvette</li>
    <li class="car__details">Color: red</li>
    <li class="car__details">Year: 1984</li>
 </ul>
</article>
<article class="car">
 </ul>
    <li class="car__make">Jeep</li>
    <li class="car__model">Wrangler</li>
    <li class="car__details">Color: silver</li>
    <li class="car__details">Year: 1999</li>
```

JSON OVERVIEW

- Now, just a hook is needed to display the data on the page.
- Typically would use append (jQuery) append or insertAdjacentHTML (vanilla) to stick the result of each map iteration into an element on the page.

```
// In main.js
let carsHTML =  cars.map(function(el) {
    let car = `<article class="car">
         <ul>
            <li class="car__make">${el.make}</li>
            <li class="car__model">${el.model}</li>
            <li class="car__details">Color: ${el.color}</li>
            <li class="car__details">Year: ${el.year}</li>
         </ul>
        </article>`;
});
 $(".cars").append(carsHTML);
// In index.html
<section class="cars">code ends up here</section>
```

DID YOU GET IT?

Can you…

- Use JS Object methods and Constructors?
- Use JSON in your assignment?

QUESTIONS?

As always feel free to contact me though Canvas if you have any questions. I do have a full-time job, so I might not get back to you immediately.

If you don't hear back from me in 24 hours, please ping me again.