JavaScript Classes

Learning Objectives

- Understand what classes are
- Understand how to create and use JavaScript classes

Material

Classes

Let's think about some of the software we use everyday. A social media site, for example, might have a set of users who add comments to posts. A user will have a username and a profile picture, but these will be different for each user. Likewise, comments will have an author (user) and some text. What about a video game? Our game might have characters, items and levels. Again, each character may be different, but they're all still characters and will share behaviour.

In Object Oriented Programming, we want to represent these entities as objects. When we have a set of objects that are similar - like users, for example - we want to create a *class* to encapsulate that type of entity.

Classes are templates for creating objects.

In our social media example, we would create a separate class for User, Comment and Post - since these are different classes of entity.

Creating a JavaScript Class

Let's look at an example for creating a class to represent a person. A person will have the following properties and methods:

- `name` - the person's name as a string
- `parents` - a 2-element array containing references to the person's parents
- `printParents()` - a function that console.logs the names of the person's parents

In JavaScript, this can be written like so:

```
class Person {
    constructor(name, parent1, parent2) {
        this.name = name;
        this.parents = [parent1, parent2];
```

```
    }

    printParents() {
        console.log(this.parents[0].name);
        console.log(this.parents[1].name);
    }
}
```

This is our blueprint for creating people - let's unpack it.

`class Person { ... }` declares a new `Person` class. The convention for classes is to name them with an upper-case letter and in singular form (i.e. "Person" not "People" or "Persons").

The `constructor` is a special, optional function that gets run when we use our class to create an object. It allows us to pass in the custom data that makes a new Person different from all the other people. In our case, people can have different names and parents, so we put them in the constructor.

We call the process of creating an object using a class *instantiation*, and the object produced is called an *instance* of the class.

Within our constructor, we save the passed in arguments to our object using the `this` keyword. We also can add a method to our class called `printParents()` which prints the parents of the instance it runs on by using the same `this` keyword.

Using a Class to Create an Object

Once we've defined our class, we can get started using it as a template for instantiating objects. We can achieve this by using the `new` keyword and passing in the arguments that will be used in the class' `constructor`:

```
const ned = new Person("Ned");
const cat = new Person("Catelyn");
const sansa = new Person("Sansa", ned, cat);
console.log(sansa);
// Person {
//     name: 'Sansa',
//     parents: [
//       Person { name: 'Ned', parents: [Array] },
//       Person { name: 'Catelyn', parents: [Array] }
//     ]
// }
```

Here, we created three instances of the `Person` class: `ned`, `cat` and `sansa`. Each of them is simply an object.

Core Assignments

Classic Literature

Create two classes: `Book` and `Author`.

Books should have:

- A `title` (string)
- An `author` (Author)
- A `latestEdition` (integer, starting at 1 for all instances)
- A `newEdition()` method (increments the book's `latestEdition` by 1)

Authors should have:

- A `name` (string)
- A `yearOfBirth` (integer)

Create some instances of each class to check they work correctly.

Ideally, we should only define one class per file. See if you can separate your Book and Author classes into separate files and import/export them as required.

Additional Resources

- Short video on JavaScript class basics https://youtu.be/AV6ZD8NX_Tc