Code Warm Up!

https://github.com/fccOttawa/coding-challenges/blob/master/mar-5.md

Wifi: Shopify Guests
Password:

freeCodeCamp [Ottawa] 2019

< Welcome! />

Wifi: Shopify Guests
Password:

FCC [Ottawa] Online

- Meetup: https://www.meetup.com/freeCodeCampOttawa/
- Facebook: https://www.facebook.com/groups/freeCodeCampOttawa/
- GitHub: https://github.com/fccOttawa

Agenda:

- Code warm up
- Welcome & announcements
- Introduction to OOP
- Q&A

Object-oriented programming in Javascript

OOP there it is!

Lauren Mayers & Sarah McCue

What we will cover

- What is object oriented programming?
- Creating objects in Javascript
- Working with objects
- Additional resources

Follow along with examples

https://fccottawa.github.io/oop-workshop/

- script.js
- Open the console
- Click “Run”

What is Object Oriented Programming (OOP)?

A programming paradigm (model) organised around objects rather than actions, and data rather than logic.

Each object can be viewed as an independent little machine with a distinct role or responsibility.

Pros and Cons

PROS

- Parallel development
- Reusable
- Maintainable
- Easier to read

CONS

- Expensive
- Bloated code

Main Principles of OOP in JS

1. **Encapsulation**
   Bundling data and methods together, maintaining separation from other objects.

2. **Inheritance**
   A child class can inherit characteristics from a parent class.

3. **Abstraction**
   Keeping data separate and “concealed” from other aspects of the code - only “exposing” what is necessary.

4. **Polymorphism**
   Objects can take on many forms. E.g., the same method can be called on different objects and produce a different response.

Object 101

```
var person = {
        firstName: “Marty”,
        lastName: “Graw”,
        favoriteHobby: “carnivals”,
        greeting: function() {
                console.log(“Hello!”)
                }
};
```

Key terminology:

- Properties
  - key:value pairs
- Methods

Accessing properties

Dot notation:
person.firstName // “Marty”

Bracket notation:
person[“firstName”] // “Marty”

Accessing methods

Dot notation:
person.greeting() // “Hello!”

Creating objects in Javascript

- Object literals
- ES6 constructor method
- ES6 class syntax
- Object.create()
- Prototypal instantiation

Classes & the constructor method

- We can define a ‘class’ (i.e. template) using the constructor method, and create multiple instances of it
- Invoked with the ***new*** keyword
- The ***this*** keyword refers to the object it belongs to in the context of the code executing.
- Be careful → ***new*** and arrow functions are incompatible

Prototypal Inheritance & Object.create()

- Objects have prototypes: Think templates, blueprints, or models.
- Try: `console.log(Array.prototype)`
- Object.create() is a built-in method on the Object object.

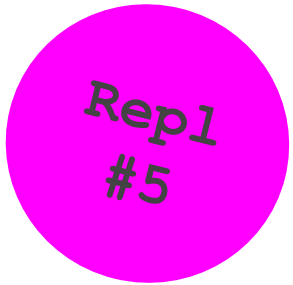
  Creates a new instance of an object, taking the prototype object as an argument.
- The prototype object passed in can also be an object of methods.

Prototypal Instantiation & Object.prototype

- We can declare custom methods on the object prototype.
- Now we can create new instances of this object (prototypal instantiation) and...
- Call the methods directly from the new instance (JS “walks” up the prototype chain to locate them).
- Walking up the prototype chain is expensive.
- Use Object.hasOwnProperty() to avoid unnecessary chain-walking!

ES6 Class Syntax

- Syntactic sugar - still uses prototype-based inheritance under the hood.
- Two ways to define classes: class declarations and class expressions (named or unnamed).
- This time we name the constructor method. Again, we invoke it with ***new*** keyword.
- Classes are not hoisted, so we need to first define them before instantiating objects from them.



ES6 Classes cont...

- We can create child classes of a parent class using the ***extends*** keyword.
- If we use a constructor in a child class, we need to first call ***super()*** before using ***this***.
- We can also extend traditional function-based “classes” → try it on repl #2 if you are interested!

Working with Javascript objects

- Modifying properties
- Looping over objects
- Handy built-in methods
- Getters & setters

Adding, updating & deleting properties - it's a piece of cake!

Taking an empty object `cake={}`, we can:

- **Add properties** - simply declare the key and value:
  ```
  cake.flavour = "vanilla";
  ```

- **Update properties** - simply redefine the previous value:
  ```
  cake.flavour = "chocolate";
  ```

- **Delete properties** - use the ***delete*** keyword:
  ```
  delete cake.flavour;
  ```

Looping over objects with for...in

- Loops through the properties of an object and executes the code once for each property

- Useful for quickly checking the properties of an object, e.g. when debugging.

Handy Built-In Methods

- **Object.keys(*obj*)** - returns an array of a given object's own property names
- **Object.values(*obj*)** - returns an array of a given object's own property values
- **Object.entries(*obj*)** - returns an array of a given object's own property [key, value] pairs (tip - pair with **Array.forEach()** to iterate through object data!)

Getters & Setters

- Getters - `get` the value of a specific property
- Setters - `set` the value of a specific property
- `set` must be passed exactly one parameter
- Created using object literal syntax

Recommended resources

- MDN Web Docs:
  https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Objects/Object-oriented_JS
  https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Working_with_Objects

- freeCodeCamp on Medium:
  https://medium.freecodecamp.org/an-introduction-to-object-oriented-programming-in-javascript-8900124e316a

- FunFunFunction on YouTube:
  https://www.youtube.com/watch?v=GhbhD1HR5vk&list=PL0zVEGEvSaeHBZFy6Q8731rcwk0Gtuxub

Putting it all into practise →

https://www.theodinproject.com/courses/javascript/lessons/library

Thank you!