Fullstack Web Development Tutorial Lesson 15

Today’s lesson will cover

- Classes

JavaScript fundamentals

Private and protected properties and methods

- One of the most important principles of object oriented programming – delimiting internal interface from the external one.
- In object-oriented programming, properties and methods are split into two groups:
  - *Internal interface* – methods and properties, accessible from other methods of the class, but not from the outside.
  - *External interface* – methods and properties, accessible also from outside the class.
- In JavaScript, there are two types of object fields (properties and methods):
  - Public: accessible from anywhere. They comprise the external interface. Until now we were only using public properties and methods.
  - Private: accessible only from inside the class. These are for the internal interface.
- **Protected properties are usually prefixed with an underscore `_`.**
  - That is not enforced on the language level, but there's a well-known convention between programmers that such properties and methods should not be accessed from the outside.
- Privates should start with `#`. They are only accessible from inside the class. This is a very new addition to JS, not fully supported in JS engines.

Class checking: ‘instanceof’

- The `instanceof` operator allows to check whether an object belongs to a certain class. It also takes inheritance into account.
- Syntax
  - obj `instanceof` `Class`
  - It returns `true` if `obj` belongs to the `Class` or a class inheriting from it.

Mixins

- In JavaScript we can only inherit from a single object. There can be only one `[[Prototype]]` for an object. And a class may extend only one other class.
- *Mixin* – is a generic object-oriented programming term: a class that contains methods for other classes.
- Some other languages allow multiple inheritance. JavaScript does not support multiple inheritance, but mixins can be implemented by copying methods into prototype.
- A *mixin* provides methods that implement a certain behavior, but we do not use it alone, we use it to add the behavior to other classes.

Exercise

JavaScript doesn't really have **classes** like other languages. They are actually functions behind the scenes. There are several ways to create classes.

- Create a `Book` class using a **JavaScript** function - instantiable.
- It should have a **author** and **published** property.
- Create an `Author` class using a **literal object** - singleton.
- It should have a **name** and **books** property.
- Create a `Publisher` class by using the **new** constructor and an anonymous function - singleton.
- It should have a **authors** and **books** property.
- Create a `Review` class using a **class** declaration - instantiable.
- It should have a **rating** and **user** property.

Exercise

- Create the *instance properties* `fullname` and `email` in the `Employee` class. Given a person's first and last names.
- Create the `fullname()` method by simply joining the first and last name together, separated by a space.
- Create the `sendEmail()` method by joining the first and last name together with a `.` in between, and follow it with `@company.com` at the end. Make sure *everything* is in **lowercase**. The method returns “Email sent to firstname.lastname@company.com”
- Create a new object and console log the `fullname()` and `sendEmail()`

Exercise

- Create a class `Smoothie` and do the following:
- Create a constructor property called `ingredients`.
- Create a `getCost` method which calculates the total cost of the *ingredients used* to make the smoothie
- Create a `getPrice` method which returns the number from *getCost* **plus** the number from *getCost* multiplied by **1.5**. Round to **2 decimal places**.
- Create a `getName` method which gets the ingredients and puts them in **alphabetical order** into a nice descriptive sentence. If there are multiple ingredients, add the word *'Fusion'* to the end but otherwise, add *'Smoothie'*. Bonus: Change **'-berries** to **'-berry'**. See the examples below.
- Create an object with one item, and another object with multiple items and log the name, price and cost on console.

```
const prices = {
    Strawberries: 1.50, Banana: 0.50, Mango: 2.50,Blueberries: 1.00, Raspberries: 1.00, Apple: 1.75, Pineapple: 3.50
  }

  class Smoothie {
// Your code
}
```

the shortcut

Self Study Assignments

To Dos

- Continue freecodecamp (FCC) Javascript. Ideally finish before we resume after summer.
- Continue with FCC HTML, CSS lessons. Ideally finish all the lessons by end of this month.
- If you believe FCC exercises aren’t the best for you if you are quite advanced already, please start working on your own project and reach out to mentors for help if needed.