ES6 CLASSES

Ultimate JavaScript Objects

WHY ES6 CLASSES?

- Many attempts have been made over the years to give JavaScript OOP features like classes and inheritance
- JavaScript functions are class-like without supporting inheritance
- Support for features like private and public variables, as well as constructors, has long been hacked into existing structures
- ES6 Classes resolve all this by providing actual classes
- Use an ES6 to make lots of similar objects

WHAT IS AN ES6 CLASS?

- Any ES6 class can be compiled down into a JavaScript function with a large amount of sugar to make things like inherited methods possible
  - ES6 classes are therefore just fancy functions
- Way to access high-level functionality without libraries or hacks
- Can be more easy to understand than functions
- Suitable for many situations where the end result is a complex object with many methods and properties

SETTING UP ATOM FOR NODE DEVELOPMENT

- Install Atom (v1.8.0 or compatible)
- Install platformio-ide-terminal (2.1.0 or compatible plugin)
- Install NodeJS
- NPM install Nodemon

CREATING AN ES6 CLASS

- ES6 classes can be created with the *class* keyword
- Can also be created with *class expressions*
- Class cannot be defined twice
- Classes have a special property called constructor that is a function which runs whenever the class is created

CLASS METHODS AND PROPERTIES

- Classes can have properties and methods just like objects
- No truly private properties
- Methods have convenient syntax without the word *function*

INHERITANCE

- Classes can inherit from other classes
  - Gain access to methods and properties
- Classes that inherit from classes can themselves be inherited from
- Best used sparingly
- Indicate inheritance with the *extends* keyword
- Access parent methods with the *super* keyword

CLASS CONCLUSION

- Classes are versatile and convenient
- Not necessary to use
- At its heart, a function that returns objects
- Only works in ES6+ but can be readily compiled to ES5