ADVANCED JAVASCRIPT

CLASSES, OOPS AND CLOSURES

CLASSES

- Classes are in fact "special functions", and just as you can define function expressions and function declarations.

```
49  class Superhero {
50      constructor(name, age, villains) {
51          this.name = name
52          this.age = age
53          this.villains = villains
54      }
55
56      speak(dialogue) {
57          console.log(dialogue)
58      }
59  }
60
61  let batman = new Superhero('Batman', 30, ['Joker', 'Penguin', 'Deathstroke'])
62  console.log(batman)
63  batman.speak('You either die a hero, or live long enough to see yourself become a villain.')
```

Note that this is extremely similar to the object created using function with the new keyword (covered in previous lecture). That's because they are essentially the same thing.

```
Superhero {
  name: 'Batman',
  age: 30,
  villains: [ 'Joker', 'Penguin', 'Deathstroke' ]
}
You either die a hero, or live long enough to see yourself become a villain.
```

INHERITANCE

```
15  class Superhero {
16      constructor(name, age, villains) {
17          this.name = name
18          this.age = age
19          this.villains = villains
20      }
21
22      speak(dialogue) {
23          console.log(dialogue)
24      }
25  }
26
27  class Avenger extends Superhero {
28      constructor(name, age, villains, species) {
29          super(name, age, villains)
30          this.species = species
31      }
32  }
33
34  let thor = new Avenger('Thor', 1000, ['Surtur', 'Gorr', 'Malekith'], 'Asgardian')
35  let ironman = new Avenger('Tony Stark', 35, ['Iron Monger', 'Mandarin'], 'Human')
36
37
38  thor.speak('You're big. I've fought bigger.')
39  console.log(ironman.age)
40
```

```
You're big. I've fought bigger.
35
```

CLOSURES

A *closure* is the combination of a function and the lexical environment within which that function was declared. This environment consists of any local variables that were in-scope at the time the closure was created.

```
20  function incrementCreator() {
21      let counter = 0     //Should execute only once
22      return function () {
23          counter++;
24          return counter;
25      };
26  }
27
28  const increment = incrementCreator()
29
30  console.log(increment())
31  console.log(increment())
32  console.log(increment())
```

```
1
2
3
```

CALLBACKS

- Callback functions are functions are used to maintain synchronization in async functions. They are passed as arguments to async functions and called after async process is finished.

```
1  function startTimer(callback) {
2      setTimeout(() => {
3          console.log("Timer of 2 seconds");
4          callback();
5      }, 2000);
6  }
7
8  function afterTimer() {
9      console.log("Timer finished");
10 }
11
12 startTimer(afterTimer);
```

```
Timer of 2 seconds
Timer finished
```