3. Callbacks

M01 W02 | Callbacks

Github Repository | Vimeo Video Recording

Topics to cover

- [X] Functions as values
- [X] Intro to anonymous functions & arrow functions
- [X] Function calling vs passing (reference to a function)
- [X] Callback functions and Higher Order functions
  - [X] Why they exist
  - [X] Implementing our own forEach and forEachInReverse
- [X] Nested scope and "scope chain"

Functions as Values

Just like everything else in JavaScript, functions are values. As a result, they can be stored in variables just like any other value

```
const myFunction = function() {
  // do something
};
```

They can also be passed around just like any other variable

```
const myFunction = function() {
  // do something
}

const myVar = myFunction;

myVar(); // equivalent to calling myFunction()
```

And they can be passed to other functions as arguments

```
const myFunction = function() {
  // do something
}

const myHigherOrderFunction = function(callback) {
  callback(); // equivalent to myFunction()
}

myHigherOrderFunction(myFunction);
```

Callbacks and Higher Order Functions

A callback is a function that gets passed to another function to be executed by that function * Callback functions are used all over the place in JavaScript * They encapsulate reusable code that can be passed around like any other JS variable * We call the function that accepts another function as an argument a **higher order function**

Anonymous Functions

We can pass callback functions *inline* to a higher order function rather than storing the callback in a variable first

```
const myHigherOrderFunction = function(callback) {
  callback();
}

// the function we pass as an argument has no name
myHigherOrderFunction(function() {});
```

Anonymous functions are simply functions that do not have a name. Don't forget that naming things is hard.

Arrow Functions

Arrow functions give us a syntactic alternative to using the `function` keyword

```
// function keyword
const myFunc = function() {
  // do something
};

// arrow function
const myArrowFunc = () => {
  // do something
};
```

- There are some *gotchas* around using the `this` keyword inside an arrow function, but if you aren't using `this`, arrow functions can be used interchangeably with "regular" functions
- Arrow functions are often used as callbacks because they are shorter/cleaner to type

```
arr.forEach(function(element) {});

// vs
arr.forEach((element) => {});
```

Useful Links

- Wikipedia: Callbacks
- MDN: Arrow Functions
- Understanding scope and context in JS

View on Compass | Leave Feedback