

# How to ES6

**A look into the future of JavaScript**

# **Sam Couch**

**Software Engineer  
@SamuelCouch**

**[www.samcouch.com](http://www.samcouch.com)**

# What's an ES6?

**Also known as** ECMAScript 6...

*ECMAScript is the "proper" name for the language commonly referred to as JavaScript*

**OK, cool. So what?**

*ES6 provides a lot of really  
awesome new features!*

# What's new?

- let keyword
- Classes
- Arrow Functions
- Modules
- Promises
- Rest Parameters
- Function Generators

**That's not everything,**

**But it's a good start!**

**Let's get into it!**



# let

Typically in JavaScript we're used to

```
var myVar = 'variable';
```

ES6 gives us enhanced scoping capabilities with variables with the use of the **let** keyword.

# let **vs.** var

var is scoped to the nearest **function** block (Or global if declared outside of a function)

let is scoped to the nearest **enclosing** block (Or global is declared outside of any block)

**Gibberish... What's that even mean?**

# let vs. var

```
let dog = 'good'; //globally scoped
```

```
var cat = 'bad'; //globally scoped
```

```
function simpleSampleFunctionBlock() {  
    let kobe = 'GOAT'; //function block scoped  
    var lebron = 'NOT GOAT'; //function block scoped  
};
```

**These samples are all the same, nothing particularly new**

# let **vs.** var

```
for(let i = 0; i < 10; i++){  
    // `i` is ONLY visible here  
}
```

```
for(var j = 0; j < 10; j++){  
    // `j` is visible here  
}  
// `j` is ALSO visible here
```

# Classes

```
var Outdated = function(thing){  
    this.lame = thing;  
}  
Outdated.prototype.getLameThing = function(){  
    return this.lame;  
}
```

**This isn't awful, but it could be prettier.**

# Classes

```
class Modern {  
    constructor(thing){  
        this.cool = thing;  
    }  
  
    getCoolthing(){  
        return this.cool  
    }  
}
```

**Now that's more like it!**

# Arrow Functions

//ES5

```
app.get('/es5', function(req, res){  
  res.send('This works');  
});
```

//ES6

```
app.get('/es6', (req, res) => {  
  res.send('Also works, and is awesome!');  
});
```

**That may not seem all too useful, but it get's better.**



# Arrow Functions

```
var numbers = [2, 4, 6, 8, 10];  
var squares = numbers.map((num) => num*num);  
console.log(squares); // [4, 16, 36, 64, 100]
```

**Single line expressions are pretty rad I think!**

# Modules

**Back in the day we would rely on** `module.exports`

# Modules (Export)

```
//utils.js
function multiply(a, b){
  return (a*b)
}
function raddify(orig){
  return orig.replace('bad', 'rad')
}
export {multiply, raddify}
//Can even specify names
// export {multiply as mult, raddify as rad}
```

# Modules (Import)

```
//app.js
```

```
import {multiply, raddify} from 'utils'
```

```
console.log(multiply(2, 4)); //8
```

```
var ex = 'This is pretty bad'
```

```
console.log(raddify(ex)); // 'This is pretty rad'
```

# Modules (...one more export...)

```
//utils.js (ES6-ified)
var utils = {
  multiply: (a, b) => {
    return (a*b);
  },
  raddify: (orig) => {
    return orig.replace('bad', 'rad')
  }
}

export default utils;
```

And now...

```
//app.js
import utils from 'utils'
console.log(utils.multiply(2, 2)); //4
```

# Promises

Better than callbacks, **PROMISES** provide representation of a value that may be made available in the future.

## Packages that have been available

- `promise`
- `q`
- `bluebird`

# Promises

```
var ourPromise = new Promise(function(resolve, reject) {  
  // do a thing, possibly async, then...  
  if (/* everything turned out fine */) {  
    resolve("Stuff worked!");  
  }  
  else {  
    reject(Error("It broke"));  
  }  
});
```

```
ourPromise.then(function(result) {  
  console.log(result); // "Stuff worked!"  
}, function(err) {  
  console.log(err); // Error: "It broke"  
});
```

# Promises

A "promisified" function is considered **thenable**. Which is to say, it has access to the **then** method.

```
Promise.then = (resolve, reject) => {  
  if SUCCESS {  
    resolve();  
  }  
  if ERROR {  
    reject();  
  }  
}
```



# Rest Parameters

Sometimes you just don't know how many parameters you'll be receiving in a given function.

```
//old.js
function getNames(){
  var names = Array.prototype.slice.call(arguments);
  names.forEach(function(name){
    console.log(name);
  })
}
```

You have to convert your arguments to an array before you can do anything with it.

# Rest Parameters

```
//new.js  
function getNames(...names){  
    names.forEach(function(name){  
        console.log(name);  
    });  
}
```

**Ain't that fancy!?**

# Function Generators<sup>1</sup>

Generators are *most* ideal for defining sequences of undetermined lengths.

**A generator requires one (or more) `yield` expressions**

Uses the notation of **`function* thing()`**

---

<sup>1</sup> Function generators aren't **fully** supported yet, but it's getting there!

# Function Generators

```
function* fibonacci(){
  let [prev, curr] = [0, 1];
  for (;;) {
    [prev, curr] = [curr, prev + curr];
    yield curr;
  }
}
```

```
//loop example
for (n of fibonacci()) {
  if (n > 20) //stop at 20
    break;
  console.log(n);
}
```

```
//using iterator methods
let fib = fibonacci();
for(var i =0; i<20; i++){
  console.log(fib.next());
}
```

**What now?**

# What now?

To get started with ES6 **today** we have to rely on other tools in order to take our modern (ES6) code and translate it into the forms that our current tools can use.

# Compilers

**(It's plural, but there's really only one that I'm recommending)**

# Babel.js



# Babel.js

*Babel is a JavaScript compiler*  
**[www.babeljs.io](http://www.babeljs.io)**

# Using Babel

The easiest way (in my opinion) is to use the **require** hook

**npm install babel**

```
//index.js  
require("babel/register");  
module.exports = require('./lib');
```

# Using Babel

## 2 Assumptions:

```
//index.js  
require("babel/register");  
module.exports = require('./lib');
```

1. You have a **lib** directory which contains your ES6-ified app
2. You run your app via **node index.js**

# More Resources

- [www.babeljs.io](http://www.babeljs.io)
- [www.github.com/ericdouglas/ES6-Learning](http://www.github.com/ericdouglas/ES6-Learning)
- [www.mzl.la/1dd9br6](http://www.mzl.la/1dd9br6)

**@SamuelCouch**