Lesson 5.2 - Parameter Handling

We're not done with the great new features that ES2015 has when it comes to dealing with parameters and variables. There are more powerful new features available that help with handling parameters in functions.

Default Parameter Values

If you've ever seen code that looks like this and cringed:

```
function add(x, y) {
  if (!x) x = 1;
  if (!y) y = 2;
  return x + y;
}
```

This is typically how you would have handled assigning default parameters to a function in past versions of JavaScript. With new ES2015 features, this can now be handled seamlessly:

```
function add(x = 1, y = 2) {
  return x + y;
}
```

Now that's pretty neat! What we're saying there is exactly the same as the code above. If no x value is passed in, then assign it the value of `1`. The same goes for the y value. This new feature makes it very easy to assign default values to your function parameters, without all the extra code to check if they exist like we did above. Awesome!

Rest Parameters

We're not done yet! There's another great new feature that we can use in a variety of ways to make parameter declarations for functions incredibly dynamic and easy to use. The next one we'll talk about is rest parameters.

Using rest parameters, we can allow a dynamic number of arguments to be passed into a function:

```
function add(...numbers) {
  return numbers.reduce((prev, curr) => prev + curr, 0);
}
```

So what's going on there? Any time you declare a parameter in a function using binding identifiers, it returns all of those parameters as an array. We can then just iterate over the collection of params and perform our addition! This is very handy in a lot of cases.

It also works when you have leading parameters:

```
function product(multiplier = 1, ...numbers) {
  const total = numbers.reduce((prev, curr) => prev + curr, 0);
  return total * multiplier;
}
```

Spread Operator

Great! We're learning some really cool new features here, and there's one more that we haven't yet uncovered. It's called the spread operator. The spread operator allows us to assign an entire array of variables all in one fell swoop. Here's an example:

```
let ary = [1, 2, 3]
let other = [4, 5, 6, ...ary]
console.log(other)
-> [4, 5, 6, 1, 2, 3]
```

That's pretty neat! We were able to define a new array that contained the contents of the other array by simply using the spread operator! You can also use this syntax to unpack strings into an array:

```
let str = 'foo'
let chars = [ ...str ]
console.log(chars)
-> ['f', 'o', 'o']
```