ES7(ES2016) & Beyond

WE'LL COVER THE FOLLOWING

There was a lot of things added in ES6, luckily ES7 or ES2016 has one two features to talk about! That's right TWO! Much easier to digest and get caught up on. These features are the Exponents operator and the Array `.includes` method.

Exponents

In JavaScript expressions are evaluated based on BEDMAS, brackets, exponents, division, multiplications, additions, and subtractions. However we have never been able to express an exponent elegantly. There is a method on `Math` that let's us express a base to some exponent.

For example, 2 to the power of 8.

```
console.log(Math.pow(2,8)); //256
```

In ES7 we now have the `**` operator that allows us express this more simply.

```
console.log(2 ** 8); //256
```

That is all you need to know about the exponent operator!

Array.includes

If you want to see if an array has a specific element in it, you have had multiple ways of doing this. You can use `.indexOf`, as we saw in Chapter 9 we can also use `.find`. The `.indexOf` method returns the index of the element if it exists or `-1` if it does not, and `.find` will actually find and return the element. However maybe we just want a simple `true` or `false` to see if something exists or not. Enter the `.includes` method!

```
const cities = ["New York","Toronto","Miami","Montreal"];
console.log(cities.includes("Paris")); //false
console.log(cities.includes("Toronto")); //true
```

The method is that straightforward, given an array you can check to see if an element exists in it or not. There is an optional second argument the method can take, and that is the index of the array that you would like to start from. If no second argument is provided it will default to `0`. Let's add some more cites to our array and see how this works.

```
const cities = ["New York","Toronto","Miami","Montreal","Vancouver","Berlin","Tokyo"];
console.log(cities.includes("Toronto",3)); //false
console.log(cities.includes("Berlin",3)); //true
```

The `fromIndex` argument tells `.includes` to start only from index 3, meaning it starts looking at `"Montreal"`. This will be beneficial if you only want to search the last half of your array for some element.