Everything new in ES2016

Let's check out the new features introduced in ES2016.

WE'LL COVER THE FOLLOWING

ES2016 introduced just only two new features:

- `Array.prototype.includes()`
- The exponential operator

`Array.prototype.includes()`

The `includes()` method will return `true` if our array includes a certain element, or `false` if it doesn't.

```
let array = [1,2,4,5];

console.log(array.includes(2));
// true
console.log(array.includes(3));
// false
```

Combine `includes()` with `fromIndex`

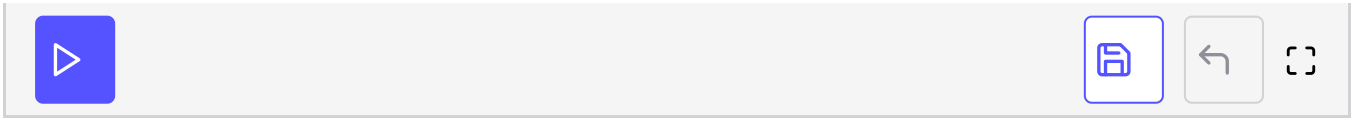
We can provide `.includes()` with an index to begin searching for an element.

Default is 0, but we can also pass a negative value.

The first value we pass in is the element to search and the second one is the index:

```
let array = [1,3,5,7,9,11];

console.log(array.includes(3,1));
// find the number 3 starting from array index 1
// true
console.log(array.includes(5,4));
//false
console.log(array.includes(1,-1));
// find the number 1 starting from the ending of the array going backwards
// false
console.log(array.includes(11,-3));
// true
```



At line 6, `array.includes(5,4);` returned `false` because- despite the array actually containing the number 5- it's found at the index 2. But we started looking at position 4. That's why we couldn't find it and it returned `false`.

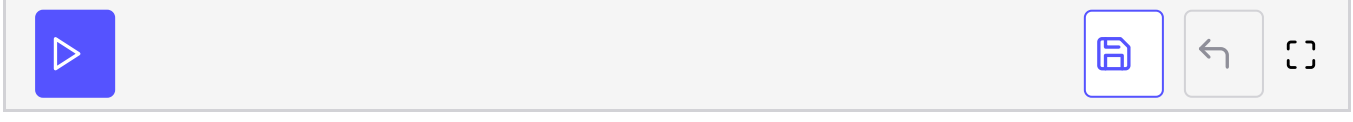
`array.includes(1,-1);` returned `false` because we started looking at the index -1 (which is the last element of the array) and then continued from that point onward.

`array.includes(11,-3);` returned `true` because we went back to the index -3 and moved up, finding the value 11 on our path.

The exponential operator

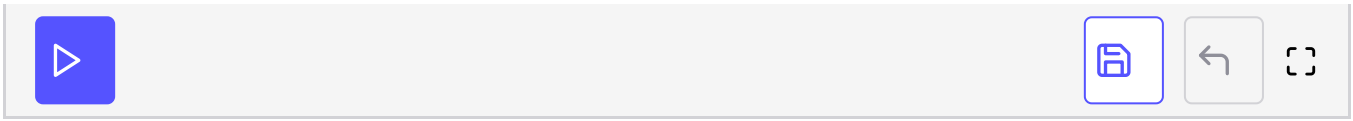
We would have done the following prior to ES2016:

```
console.log(Math.pow(2,2));
// 4
console.log(Math.pow(2,3));
// 8
```



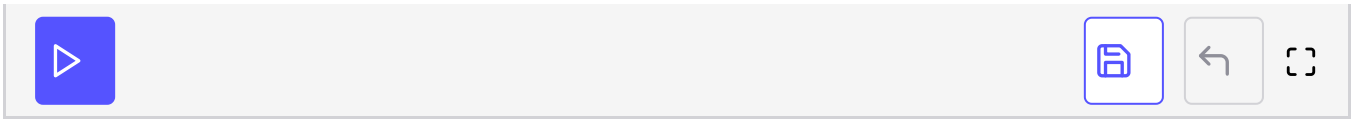
Now with the new exponential operator, we can do the following:

```
console.log(2**2);
// 4
console.log(2**3);
// 8
```



It'll become pretty useful when combining multiple operations like in this example:

```
console.log(2**2**2);
// 16
console.log(Math.pow(Math.pow(2,2),2));
// 16
```



Using `Math.pow()` you need to continuously concatenate them, which can get pretty long and messy. The exponential operator provides a faster and cleaner way of doing the same thing.

In the next lesson, we will solve a quiz and a coding challenge to test the concepts covered in this lesson.

Another quiz is on the docket next.