Finding Array Elements

 (https://github.com/learn-co-curriculum/phase-1-array-element-finding)  (https://github.com/learn-co-curriculum/phase-1-array-element-finding/issues/new)

Learning Goals

- Find elements using a simple condition with `Array.prototype.indexOf()`
- Find elements using more complex conditions with `Array.prototype.find()`

Introduction

As developers, one of the things we need to do on a regular basis is locate things in arrays. It's all well and good to be able to store data, but it's pretty useless unless we're able to get it back out again. In JavaScript, there are two different methods that we use to locate data in arrays. For simple conditions, we use `Array.prototype.indexOf()`. For more complex calculations, we use `Array.prototype.find()`.

Find Elements Using a Simple Condition with `Array.prototype.indexOf()`

`Array.prototype.indexOf()` is called on an array and takes two arguments: the value you are looking for and an optional start position. It compares each element in turn to the value you're looking for using the strict equality operator (`===`) and returns the index of the first matching element. If the element isn't contained in the array, it returns -1.

```
const cards = ['queen of hearts', 'jack of clubs', 'ten of diamonds', 'ace of spades'];

cards.indexOf('jack of clubs'); //=> 1
cards.indexOf('jack of hearts'); //=> -1
```

If you pass in the optional second argument, `indexOf()` will begin the search at the specified position:

```
cards.indexOf('ace of spades', 2); //=> 3
cards.indexOf('jack of clubs', 2); //=> -1
```

In this case, `Array.prototype.indexOf()` returns `-1` if either the value isn't found *or* if the start position you pass in is after the element you're looking for.

Find Elements Using More Complex Conditions with `Array.prototype.find()`

`Array.prototype.find()` allows you to execute more complex searches by passing it a callback function. The method will automatically iterate through the array, call the callback on each value, and return the first element in the array that satisfies the condition defined by the function. If no matching element is found, `undefined` is returned.

`Array.prototype.find()` iterates through the array it's called on and, in each iteration, passes three arguments to the callback: the current element of the array, the index of the current element, and the array itself. These arguments can then be captured as parameters in the callback and used inside the function.

Say we want to determine whether an array of numbers contains any odd values. We can write the following callback function to do this:

```
function isOdd(element, index, array) {
  return (element % 2 === 1);
}
```

`Array.prototype.find()` will iterate through the array, passing each element in turn to `isOdd()`. If the element is not odd, the callback returns `false` and the iteration continues. If an odd element is encountered, the callback will return true, and `Array.prototype.find()` will return that element.

Remember that `Array.prototype.find()` *automatically* passes the three arguments to our function. By defining `isOdd()` with three parameters, we make those values available inside our function. In this example, we're only using the first one, the current element of the array, but all three are being passed in and are available inside our function if we want to use them.

Let's call `.find()` on the array we want to search, and pass our function as an argument:

```
function isOdd(element, index, array) {
  return (element % 2 === 1);
}

[4, 6, 8, 10].find(isOdd); //=> undefined, not found
[4, 5, 8, 10].find(isOdd); //=> 5
[4, 5, 7, 8, 10].find(isOdd); //=> 5
[4, 7, 5,  8, 10].find(isOdd); //=> 7
```

Note that only the first argument — the current element in the array — is required for the callback function. If (as in our example above) your callback doesn't use the other two arguments, you can define your function with only one parameter. This will work as well:

```
function isOdd(element) {
  return (element % 2 === 1);
}
```

Conclusion

Both `Array.prototype.indexOf()` and `Array.prototype.find()` can be very useful in different situations. `Array.prototype.indexOf()` is used when you want to check an array for a simple value; you call `indexOf()` on an array, passing the value you're looking for as the argument. `Array.prototype.find()` is also called on an array, but it takes a *function* as an argument. This enables you to define the condition the element should meet, allowing for more complex searches.

Resources

- Object.prototype (https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object/prototype)
- Array.prototype.indexOf() (https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/indexOf)
- Array.prototype.find() (https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/find)