Working With Sets

Checking The Length

Once you've constructed your Set, there are a couple of different properties and methods you can use to work with Sets.

Use the `.size` property to return the number of items in a Set:

```
const months = new Set(['January', 'February', 'March', 'April', 'May', 'June', 'July',
'August', 'September', 'October', 'November', 'December']);
console.log(months.size);
```

```
12
```

Remember, Sets can't be accessed by their index like an array, so you use the `.size` property instead of `.length` property to get the size of the Set.

Checking If An Item Exists

Use the `.has()` method to check if an item exists in a Set. If the item is in the Set, then `.has()` will return `true`. If the item doesn't exist in the Set, then `.has()` will return `false`.

```
console.log(months.has('September'));
```

```
true
```

Retrieving All Values

Finally, use the `.values()` method to return the values in a Set. The return value of the `.values()` method is a `SetIterator` object.

```
SetIterator {'January', 'February', 'March', 'April', 'May', 'June', 'July',
'August', 'September', 'October', 'November', 'December'}
```

More on the `SetIterator` object in a second!

> **TIP**: The `.keys()` method will behave the exact same way as the `.values()` method by returning the values of a Set within a new Iterator Object. The `.keys()` method is an alias for the `.values()` method for similarity with maps. You'll see the `.keys()` method later in this lesson during the Maps section.