## Maps

If Sets are similar to Arrays, then Maps are similar to Objects because Maps store key-value pairs similar to how objects contain named properties with values.

Essentially, a Map is an object that lets you store key-value pairs where both the keys and the values can be objects, primitive values, or a combination of the two.

### How to Create a Map

To create a Map, simply type:

```
const employees = new Map();
console.log(employees);
```

```
Map {}
```

This creates an empty Map `employee` with no key-value pairs.

### Modifying Maps

Unlike Sets, you can't create Maps from a list of values; instead, you add key-values by using the Map's `.set()` method.

```
const employees = new Map();

employees.set('james.parkes@udacity.com', {
    firstName: 'James',
    lastName: 'Parkes',
    role: 'Content Developer'
});
employees.set('julia@udacity.com', {
    firstName: 'Julia',
    lastName: 'Van Cleve',
    role: 'Content Developer'
});
employees.set('richard@udacity.com', {
    firstName: 'Richard',
    lastName: 'Kalehoff',
    role: 'Content Developer'
});

console.log(employees);
```

```
Map {'james.parkes@udacity.com' => Object {...}, 'julia@udacity.com' => Object {...}, 'richard@udacity.com'
=> Object {...}}
```

The `.set()` method takes two arguments. The first argument is the key, which is used to reference the second argument, the value.

To remove key-value pairs, simply use the `.delete()` method.

```
employees.delete('julia@udacity.com');
employees.delete('richard@udacity.com');
console.log(employees);
```

```
Map {'james.parkes@udacity.com' => Object {firstName: 'James', lastName: 'Parkes', role: 'Course
Developer'}}
```

Again, similar to Sets, you can use the `.clear()` method to remove all key-value pairs from the Map.

```
employees.clear()
console.log(employees);
```

```
Map {}
```

> **TIP**: If you `.set()` a key-value pair to a Map that already uses the same key, you won't receive an error, but the key-value pair will overwrite what currently exists in the Map. Also, if you try to `.delete()` a key-value that is not in a Map, you won't receive an error, and the Map will remain unchanged.
>
> The `.delete()` method returns `true` if a key-value pair is successfully deleted from the `Map` object, and `false` if unsuccessful. The return value of `.set()` is the `Map` object itself if successful.