Other Types

There are a lot of cases where you might want to store some data that is more complicated than a number, a boolean, or a string. What if you wanted to store a list of items? Or maybe a ton of properties for an item? There are two data types which you will allow you to do this: `arrays` and `objects`.

Arrays

An `Array` is a list of items. These items can be whatever you want: numbers, words, or even another entire array. Arrays are denoted by two non-curly brackets `[]`.

```
const movieGenres = ["Action", "Comedy", "Horror", "Romance"];
```

As you can see, an array is created the same way that you would any other variable, with either let or const, followed by an equal sign and then the array.

Often, you're going to want to get a certain value from an array. To do this, we're going to need to know the `index` of an item. An `index` is the position of the item in the array. Something very important to keep in mind is that arrays are `zero-indexed`, meaning that they start AT 0, and not at 1.

To access an item, you type the name of the array, followed by two brackets with the index inside.

```
// This will give us the genre in the 0th index, which is Action.
movieGenres[0];
```

What would it look like if we wanted to log "Horror" to the console?

```
console.log(movieGenres[2]);
```

What do you think would happen if we tried to log something outside the array?

```
// This will log: undefined
console.log(movieGenres[4]);
```

What do you think we should do if we wanted to the change the value of the item in the first index?

```
movieGenres[1] = "Thriller";
```

It's common practice to declare arrays with the `const` keyword. Why would we use `const` rather than `let` even if we're going to change the values in the array?

- We are only changing the items in the array, not the entire array itself.

Objects

Think about if you were running a car dealership, and each car had a price, color, and weight. You would most likely want to group that information under one item so that way they are all attributed to that item. `Objects` allow you to do this.

An object is created by using two curly brackets:

```
const car = {}
```

Again, we declare objects with 'const' (though you can use 'let') and name the variable whatever we want. Then, we put all of the information in between the curly brackets. A single piece of information inside the object is called a `property`. A property is stored as a key/value pair. The `key` is the name, and the `value` is what you want it to be set to.

```
const car = {
    price: 40000,
    color: "blue",
    weight: 3500
}
```

As you can see, it looks very similar to properties in CSS. The key is followed by a colon, and then the value afterward. Each property is separated with a comma. The values can be any type: integer, string, array, or even an entire different object.

Let's say we want to access the price of this car. In order to get the value of that key, all we need to do is type the name of the variable followed by a dot and the name of the key:

```
car.price;
```

How would we log the weight of the car to the console?

```
console.log(car.weight);
```

Sometimes, you might want to add another property to an already existing object. Let's say we wanted to add another property called 'condition'; all we have to do is specify the object name, the key we want to add, and the value:

```
car.condition = "New";
```

If the 'condition' property doesn't already exist, it will add it to the object. If condition did already exist within the object, it would overwrite the old value and set it to "New". Our car object now looks like this:

```
const car = {
    price: 40000,
    color: "blue",
    weight: 3500,
    condition: "New"
}
```

If you want to store objects within objects, you will have to traverse both objects using dot notation.

```
const car = {
    price: 40000,
    color: "blue",
    weight: 3500,
    condition: "New",
    engine: {
        cylinders: 4,
        size: 2.2
    }
}
```

How would we get the amount of cylinders of the engine?

```
car.engine.cylinders;
```

Methods

There are a few different methods that will be helpful when it comes to arrays.

Array methods

```
const fruits = ["Orange", "Apple", "Pear"];

fruits.length; // Gives you the length of the array
// Note: length starts AT 1, not 0
fruits.push("Banana"); // Pushes (adds) an item to the array at the end
fruits.unshift("Grape"); // Adds an item to the array at the beginning
fruits.pop(); // Removes the last element in the array and returns that element
fruits.shift(); // Removes the first element in the array and returns that element
```

```
// Sometimes, you don't know the index of an item. In that case, you can use
indexOf which will give you the index in return
let pearIndex = fruits.indexOf("Pear"); // Will return the number 2 and store it
in a variable
```

Object methods

Objects have methods, but they are not as useful since we can edit and access the properties of the object directly using dot notation. We aren't going to cover them here, but you can still check some of them out on MDN or W3Schools: https://www.w3schools.com/js/js_object_methods.asp