Methods in Objects

In this lesson, we will learn about methods in objects, how to declare them, and how to use the "this" keyword to set and get values in methods.

We'll cover the following

As discussed earlier, an object contains properties, where the property value can either be a value or a function. In case the property is a function, it is referred to as an **object method**.

`this` Keyword

Let's consider a scenario where you have an object named `employee` with the properties `name`, `age` and `designation`. Now you are required to write an object method `display()` that returns the `designation` of that employee. The first approach that comes to mind is something like this:

```
var employee = {
  name: 'Joe',
  age: 28,
  designation: 'developer',
  display() {
    return designation
  }
}
//this will generate an error
console.log(employee.display())
```

When you run the code, you probably saw a `designation is not defined` *error*, the reason being that you can't directly access the properties inside the object. The properties are accessed similarly to how they would be accessed outside the object, i.e., you need to provide a reference to the object whose property you are trying to access.

So how does an object refer to itself?

It refers to itself using the `this` keyword. Here, `this` points to the current object, i.e., the object in which the code is being written. In the above example, `this` refers to the `employee` object.

Syntax

Let's take a look at the syntax below:

```
this.propertyName
```

Examples

Let's modify the code above to implement `this`:

```
var employee = {

  name: 'Joe',
  age: 28,
  designation: 'developer',
  //function returning designation of the employee
  display() {
    return this.designation //using this to refer to the "employee" object
  }
}
//this will display the designation
console.log(employee.display())
```

Explanation

In the example above, **line 8** translates to *"returning the `designation` property of* ***this*** *object"*. Here, `this` refers to the `employee` object.

of this object". Here, `this` refers to the `employee` object.

Now let's return all three properties!

```
var employee = {

  name: 'Joe',
  age: 28,
  designation: 'developer',
  //function returning all three properties of the employee
  display() {
    return " Name is " + this.name + "\n Age is " + this.age + "\n Designation is " + this.de
  }
}
//this will display all three properties
console.log(employee.display())
```

Setting Value Using `this`

Till now we were using `this` to get a property within an object; however, we can also use `this` to set the value of a property within an object.

Example

Let's take a look at an example below:

```
var employee = {

  name: 'Joe',
  age: 28,
  designation: 'developer',
  //function setting the value of "designation" equal to the parameter being passed to the fu
  setDesignation(parameterValueOfDesig) {
    this.designation = parameterValueOfDesig
  }
}
//displaying the value of "designation" at start
console.log("Old designation was:",employee.designation)
//updating the value of designation
employee.setDesignation('engineer')
//displaying new value of designation
console.log("New designation is:",employee.designation)
```

Explanation

As seen in the code above:

- In **line** 7, the function `setDesignation(parameterValueOfDesig)` is defined with `parameterValueOfDesig` being passed to it as a parameter.
- In **line 8**, the value of `designation` is set as equal to `parameterValueOfDesig`.
- In **line 14**, the `setDesignation` function is called with the parameter `engineer`. Hence, the original value of `designation` is updated, as can be seen in the result displayed.

We learned how to get and set values using `this`. In the next lesson, let's learn how to use the keywords `get` and `set`.