Methods in Constructor Functions

This lesson teaches us how to define and add new methods in a constructor function.

WE'LL COVER THE FOLLOWING

Defining Methods

Methods are defined differently in *constructor functions*.

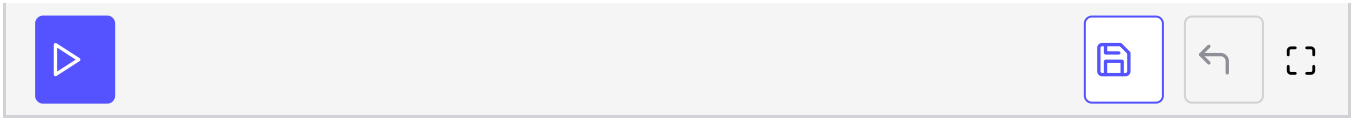
Example

We defined our methods inside object literals in the following way:

```
//creating an object named employee
var employee = {
  //defining properties of the object
  //setting data values
  name : 'Joe',
  age : 28,
  designation : 'Developer',
  //function to display name of the employee
  displayName() {
    console.log("Name is:", this.name)
  }
}
//calling the method
employee.displayName()
```

Now, let's write the same function but for the *constructor function* this time:

```
//creating an object named Employee
function Employee(_name,_age,_designation) {
   this.name = _name,
   this.age = _age,
   this.designation = _designation,
  //function to display name of the Employee
  this.displayName = function() {
    console.log("Name is:", this.name)
  }
}
//creating an object
var employeeObj = new Employee('Joe',22,'Developer')
//calling the method for employeeObj
employeeObj.displayName()
```



Explanation

In **line 9** of the code for object literal, the method declaration was simple: the name of the function followed by the definition, i.e., the body of the function.

However, as seen in **line** 7 of the *constructor function* code, in order to define the method, `this` keyword is used. As discussed earlier, `this` is used to set as well as assign the method and properties to every new object that is created.

> **Note:** The `function` *keyword* provides another way to declare a function. It can also be used while declaring a function in an object literal.

Adding a Method

Just like adding *properties*, adding a *method* to a *constructor* function is different from adding it to an object.

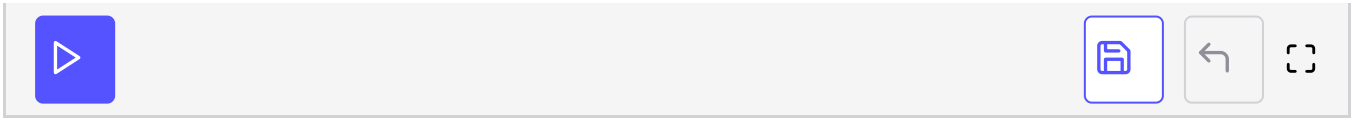
Let's compare the two methods.

Example

A new method is added to an object in the following way:

```
//creating an object named employee
var employee = {
  //defining properties of the object
```

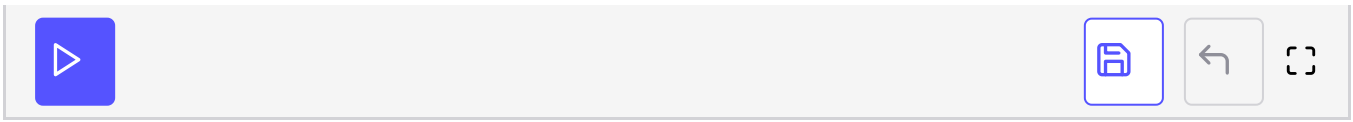
```
  //setting data values
  name : 'Joe',
  age : 28,
  designation : 'Developer',
  //function to display name of the employee
  displayName() {
    console.log("Name is:",this.name)
  }
}
//calling the method
employee.displayName()
//adding a method that returns the age to the object
employee.getAge = function(){
  return this.age
}
//calling the function and displaying the result
console.log("Age is:",employee.getAge())
```



As you can see in **line 16**, new methods can be added directly into an existing object; however, a new method cannot directly be added into an object constructor. If a new method is to be added, it has to be defined inside the constructor function so that it is available for all objects created from it.

Let's take a look at its example below:

```
//creating an object named Employee
function Employee(_name,_age,_designation) {
   this.name = _name,
   this.age = _age,
   this.designation = _designation,
  //function to display name of the Employee
  this.displayName = function() {
    console.log("Name is:",this.name)
  }
  //adding a new setAge function
  this.setAge = function(newAge){
    this.age = newAge
  }
}
//creating an object
var employeeObj = new Employee('Joe',22,'Developer')
//calling the setAge function for employeeObj
employeeObj.setAge(30)
//displaying new age
console.log("New age is",employeeObj.age)
```



Explanation

A new method, `setAge`, is added by being defined inside the *constructor function*, as seen in **line 11**, with the parameter `newAge` passed to it. The function is used to modify the `age` property for an object.

> **Note:** From the code above, you can see that the method to add a new function is the same as that of defining a function in a constructor function.

Is there another way to add properties and methods to a constructor function? An approach using which they can be added directly instead of having to be defined inside the constructor? Let's find out in the next lesson!