# Data Encapsulation in ES5

This lesson explains the steps to protect properties and teaches us how to access these protected properties.

#### We'll cover the following

- Hiding Data
- Example
  - Steps
- Accessing Protected Properties
  - Example

# Hiding Data #

Till now, all the constructor function properties were easily accessible outside of it, and their values could be modified without any restrictions. In both ES5 and ES6 versions of JavaScript, there is no formal support for access modifiers using *keywords* such as public, private or protected. By default, all properties are public; hence, they can easily be accessed.

Let's discuss property protection for the ES5 version. We will look at the ES6 version later on in the course.

Before we get into details, let's take a look at the following term:

**internal property**: a *property* or *method* that can only be accessed by methods present *inside* the *constructor* function.

An *internal property* can then be considered protected from being accessed in an unauthorized manner.

# Example #

Let's start by taking a look at how properties were being defined inside the constructor function previously:

```
//function constructor called Employee
    function Employee(name, age, designation){
                                                                               6
      this.name = name
      this.age = age
      this.designation = designation
    //creating an object called employeeObj
   var employeeObj = new Employee('Joe', 22, 'Developer')
10
   //displaying properties of employeeObj
11
12
   console.log("Name of employee:",employeeObj.name)
   console.log("Age of employee:",employeeObj.age)
13
   console.log("Designation of employee:",employeeObj.designation)
```

Now let's follow the steps below in order to modify the code above and provide protection to the properties.

### Steps #

- remove this
- declare the properties using var keyword instead
- add leading \_ to the names of the properties

Using *underscore* is a convention followed by programmers to indicate that such properties should not be accessed from outside.

Time to convert the properties into internal properties!

```
//function constructor called Employee
function Employee(name, age, designation){
  //declaring properties using "var" keyword to protect them
  var _name = name
  var _age = age
  var _designation = designation
}
```

Protecting properties

The use of var declares the *properties* locally; hence they cannot be accessed directly from the outside. Try running the code below:



The code above displays undefined when property values are accessed. The code tries to retrieve values of the local variables from the <a href="mailto:employee0bj">employee0bj</a> instance created. However, since all values are stored in local variables, only the functions inside the *constructor function* will be able to access them.

# Accessing Protected Properties #

So how will the internal properties be accessed outside of <a href="Employee">Employee</a>?

**Get/Set** functions can be used to retrieve these property values as well as set them. These functions were discussed previously as well. They will be declared as public functions inside <a href="Employee">Employee</a> due to which they will have access to the internal properties.

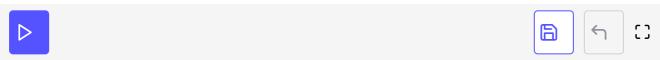
### Example #

Let's implement the *get/set* functions in order to retrieve as well as set the internal property values outside the constructor function.

**Note:** The method to declare a *get/set* function is the same as that of any other function being declared inside a *constructor* function.

```
//constructor function called Employee
function Employee(name, age, designation){
  var _name = name
  var _age = age
```

```
var _designation = designation
  //defining the get function
  this.getName = function() {
    return _name
  this.getDesignation = function() {
    return _designation
  //defining the set function
 this.setDesignation = function(desig) {
    _designation = desig
  }
}
var employeeObj = new Employee('Joe',20,'Chef')
//calling the getName function
console.log("Name of the employee is:", employeeObj.getName())
//calling the setDesignation function
employeeObj.setDesignation('Developer')
//calling the getDesignation function
console.log("New designation is:",employeeObj.getDesignation())
```



**Accessing Protected Properties** 

As seen in **lines 8, 11** and **15** property values are being accessed directly by the functions since they are defined inside **Employee**. These functions are then accessed/called outside in order to get/set the values of the properties.

Now that you're familiar with constructor functions, let's test that knowledge with a quiz in the next lesson.