Working with Objects

In JavaScript, all values, except primitive values, are objects

In JS, object is a collection of Name and Value pairs:

The values are written as **name : value** pairs

```
var person = {
   firstName: "John",
   lastName: "Doe",
   id: 5566,
   fullName : function() {
      return this.firstName + " " + this.lastName;
   }
};
```

Using an Object Constructor

```
//Following is Account class
      function Account(id, name, balance) {
         //Following code will execute when the object is created
         alert("Object of Account is instantiated");
         //Following are properties of the class
         this.Id = id;
         this.Name = name;
         this.Balance = balance;

         //Following are methods of the class.
         //Drawback of this is that this method is recreated for every new object.
         this.Deposit = function (amount) {
            this.Balance += amount;
         }
         //Reusing the same method of all objects - Use the address of existing object.
         this.ShowDetails = ShowAccountDetails;
      }
      function ShowAccountDetails() {
         alert(this.Id + " " + this.Name + " " + this.Balance);
      }
```

You can add new properties to an existing object by simply giving it a value.

The standard way to create an object prototype is to use an object constructor function as in the Account example above.

The **prototype property** allows you to add properties and methods to an existing object.

```
Account.prototype.Withdraw = function (amount) {
   if (this.Balance - amount < 500)
      throw "Insufficient funds";
   this.Balance -= amount;
}
```

Call Method in JavaScript

The **call** method is used to call a method on behalf of another object. It allows you to change the "*this*" object of a function from the original context to the new object specified by *thisObj*.

call([*thisObj* [, *arg1* [, *arg2*[, [, *argN*]]]]])

```
<!DOCTYPE html>
<html lang="en" xmlns="http://www.w3.org/1999/xhtml">
<head>
   <meta charset="utf-8" />
   <title></title>
</head>
<body>
   <script>
      function callMe(arg1, arg2) {
         var s = "";
         s += "this value: " + this;
         s += "<br />";
         for (i in callMe.arguments) {
            s += "arguments: " + callMe.arguments[i];
            s += "<br />";
         }
         return s;
      }

      document.write("Original function: <br/>");
      document.write(callMe(1, 2));
      document.write("<br/>");

      document.write("Function called with call: <br/>");
      document.write(callMe.call(3, 4, 5));
   </script>
</body>
</html>
```

Inheritance in JavaScript using prototype

With **call**, you can write a method once and then inherit it in another object, without having to rewrite the method for the new object.

```
//Current Account class
function CurrentAccount(id, name, balance, companyName) {
   //Call base class constructor
   Account.call(this, id, name, balance);
   //Initialize current class properties
   this.CompanyName = companyName;
}

//Inherit Account Members
CurrentAccount.prototype = Account.prototype;

//Correct the constructor pointer because it points to Account
CurrentAccount.prototype.constructor = CurrentAccount;

//Replace Withdraw Implementation
CurrentAccount.prototype.Withdraw = function (amount) {
   if (this.Balance - amount < 0)
      throw "Current Account - Insufficient funds";
   this.Balance -= amount;
}

//Using the class
try {
   var a1 = new CurrentAccount(1, "A1", 10000,"Demo");
   a1.Deposit(1000);
   a1.Withdraw(10800);
   a1.ShowDetails();
}
catch (err) {
   alert(err);
}
```