Getting “MEAN”
- A Practical Workshop

JavaScript Basics

JavaScript :

Let's explain just some of the basic features of the JavaScript language, to give you some more understanding of how it all works.

Better yet, these features are common to all programming languages. If you can understand these fundamentals, you should be able to start programming just about anything!

What is JavaScript ?

- Javascript is a dynamic computer programming language.

- It is lightweight and most commonly used as a part of web pages, whose implementations allow client-side script to interact with the user and make dynamic pages.

- It is an interpreted programming language with object-oriented capabilities.

History :

- JavaScript was first known as **LiveScript,** but Netscape changed its name to JavaScript, possibly because of the excitement being generated by Java.
- JavaScript made its first appearance in Netscape 2.0 in 1995 with the name **LiveScript**. The general-purpose core of the language has been embedded in Netscape, Internet Explorer, and other web browsers.

Advantages :

Less server interaction− You can validate user input before sending the page off to the server. This saves server traffic, which means less load on your server.

Immediate feedback to the visitors− They don't have to wait for a page reload to see if they have forgotten to enter something.

JavaScript Syntax :

i. Initialization of JS :

JavaScript can be implemented using JavaScript statements that are placed within the**<script>... </script>**HTML tags in a web page.

ii. Semicolons are Optional :

```
<script language="javascript" type="text/javascript">
        <!--
                var1 = 10
                var2 = 20
        //-->
</script>
```

but when formatted in a single line then you must use semicolon.

```
<script language="javascript" type="text/javascript">
        <!--
                var1 = 10; var2 = 20
        //-->
</script>
```

iii. Case sensitivity :

JavaScript is case sensitivity language.
Eg : var a and var A are two different variables.

iv. Comments in JavaScript :

JavaScript supports C, C++, Java as well as HTML type style comments.

```
<script language="javascript" type="text/javascript">
    <!- -
        // this is a single line comment. Similar to C , C++, Java styles
                /*
                *       This is a
                *       multline comment.
                */
        //-->
</script>
```

JavaScript Datatypes

JavaScript Datatypes : Variables

Variables are containers that you can store values in. You start by declaring a variable with the **var** keyword, followed by any name you want to call it.

Example :

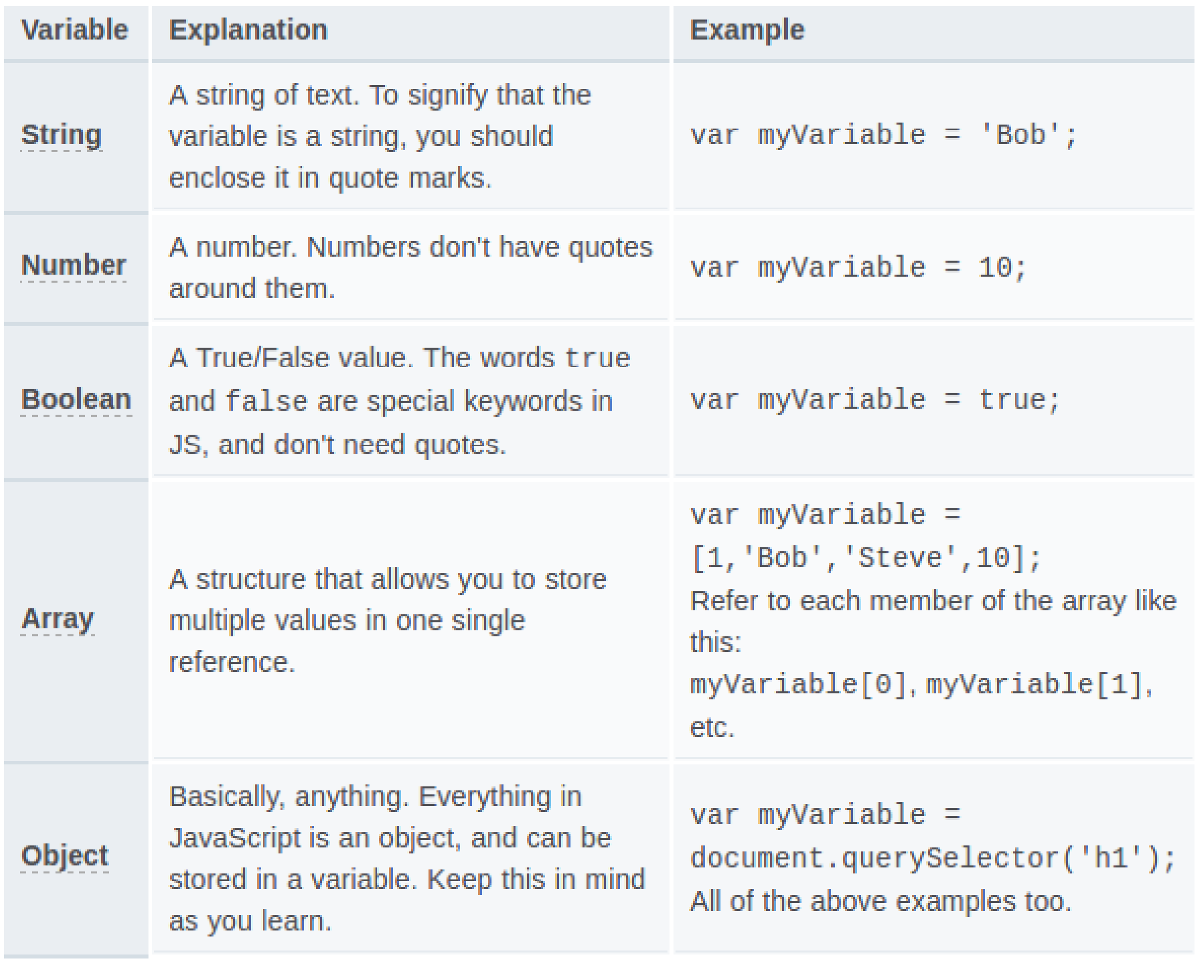
```
var myVariable = ‘Bob’;                // here myVariable holds string

var myVariable;
myVariable = “Bob”;                    // here myVariable holds string

var myVaribale = 1991;                 // here it holds number
```

- JavaScript variables can hold many **data types**:

- numbers,
- strings,
- arrays,
- booleans,
- objects and more



Variable	Explanation	Example
String	A string of text. To signify that the variable is a string, you should enclose it in quote marks.	`var myVariable = 'Bob';`
Number	A number. Numbers don't have quotes around them.	`var myVariable = 10;`
Boolean	A True/False value. The words `true` and `false` are special keywords in JS, and don't need quotes.	`var myVariable = true;`
Array	A structure that allows you to store multiple values in one single reference.	`var myVariable = [1,'Bob','Steve',10];` Refer to each member of the array like this: `myVariable[0]`, `myVariable[1]`, etc.
Object	Basically, anything. Everything in JavaScript is an object, and can be stored in a variable. Keep this in mind as you learn.	`var myVariable = document.querySelector('h1');` All of the above examples too.

JavaScript Functions

JavaScript Functions :

A JavaScript function is defined with the ***function*** keyword, followed by a name, followed by parentheses ().

Function names can contain letters, digits, underscores, and dollar signs (same rules as variables).

The parentheses may include parameter names separated by commas: (parameter1, parameter2, ...)

The code to be executed, by the function, is placed inside curly brackets: {}

Example :

```
function name (parameter1, parameter2, parameter3) {
        ... code to be
        ... executed
}
```

Note : You can even write anonymous function. (function without name).

Example :

```
function (x, y) {
        return a+b;
}
```

- You can assign the anonymous function to a variable for further use.

Example :

```
var myVariable = function () {
        return a +b;
}

console.log("Addition is : " + myVariable(2,3) ) ;

//      Output : 5
```

JavaScript Objects

JavaScript Objects :

- In JavaScript, objects are vital. If you understand objects, you understand JavaScript ;)

- In JavaScript**, almost "everything" is an object.**

- Booleans can be objects (or primitive data treated as objects),
- Numbers can be objects (or primitive data treated as objects),
- Strings can be objects (or primitive data treated as objects),
- Dates are always objects,
- Maths are always objects,
- Regular expressions are always objects,
- Arrays are always objects,
- Functions are always objects
- Objects are objects

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

- Primitive values are: strings ("John Doe"), numbers (3.14), true, false, null, and undefined.

- In short, “Objects are Variables Containing Variables”.

i. JavaScript variables can contain single values:

Example :

var person = "Sharif Malik";

ii. Objects are variables too. But objects can contain many values.

The values are written as **name : value** pairs (name and value separated by a colon).

Example :

var person = { firstName : "Sharif", lastName : "Malik", age : 24 , city : “Pune”};

iii. Object Properties :
The named values, in JavaScript objects, are called **properties**.
i.e.

```
 firstName  : "Sharif",
 lastName:"Malik",
 age :24 ,
 city : “Pune”
```

Note : Objects written in name- value pairs are similar to :

- Associative arrays in PHP
- Dictionaries in Python
- Hash tables in C
- Hash maps in Java
- Hashes in Ruby and Perl

iv. Creating JavaScript Objects :

With JavaScript, you can define and create your own objects.

There are different ways to create new objects:

a. Define and create a single object, using an object literal.

b. Define and create a single object, with the keyword new.
c .Define an object constructor, and then create objects of the constructed type.

a. Using an Object literal :
This is the easiet way to create a JS object.

Example :

```
var person = { firstName : "Sharif", lastName : "Malik", age : 24 ,
city : “Pune”};
```

b. Using JavaScript keyword *new* :
Example :

```
var person = new Object();
person.firstName = "Sharif";
person.lastName = "Malik";
person.age = 24;
person.city = “Pune”;
```

c. Using an Object constructor :
The examples above are limited in many situations. They only create a single object. Sometimes we like to have an "object type" that can be used to create many objects of one type.

The standard way to create an "object type" is to use an object constructor function:

Example :
```
function person (firstName, lastName, age, city) {
 this.firstName = first;
 this.lastName = lastName;
 this.age = age;
 this.city = city;
}

var myBrother = new person("Shahid", "Malik", 20, "Pune");
var mySister = new person("Zubeida", "Malik", 31, "Pune");
```

Built-in JavaScript Constructors:

Example :
```
var x1 = new Object();          // A new Object object
var x2 = new String();          // A new String object
var x3 = new Number();  // A new Number object
var x4 = new Boolean();  // A new Boolean object
var x5 = new Array();           // A new Array object
var x6 = new RegExp();   // A new RegExp object
var x7 = new Function(); // A new Function object
var x8 = new Date();            // A new Date object
```

Note :
There is no reason to create complex objects. Primitive values execute much faster.

And there is no reason to use new Array(). Use array literals instead: []

And there is no reason to use new RegExp(). Use pattern literals instead: /()/

And there is no reason to use new Function(). Use function expressions instead: function () {}.

And there is no reason to use new Object(). Use object literals instead: {}

Example :

```
var x1 = {};                    // new object
var x2 = "";                          // new primitive string
var x3 = 0;                           // new primitive number
var x4 = false;                 // new primitive boolean
var x5 = [];                          // new array object
var x6 = /()/                   // new regexp object
var x7 = function(){};                // new function object
```

iv. Accessing JavaScript properties :

```
 i.    objectName.propertyName             // person.firstName
 ii.   objectName[“propertyName”]// person[“firstName”]
 iii.  objectName[expression]         //var x= “firstName”; person[x]
```

v. Adding new property to an object :

```
var mySister = new person(“Zubeida”, “Malik”, 31, “Pune”);
mySister.gender= “female”;
```

vi. Deleting property from an object :

```
var mySister = new person(“Zubeida”, “Malik”, 31, “Pune”);
delete mySister.age;
```

The delete keyword deletes both the value of the property and the property itself.

vi. Object Methods :

A JavaScript **method** is a property containing a **function definition**.
Example :

```
function person (firstName, lastName, age, city) {
 this.firstName = first;
 this.lastName = lastName;
 this.age = age;
 this.city = city;
```

```
        //property containing a function definition
        this.fullName : function () {
                return this.firstName + “ ” + this.lastName;
        }
}
```

vii. Accessing Object Method :

You can create object of defined construtor and access the method.

Syntax : ObjectName.methodName();

Example :

```
var myBrother = new Person(“Shahid”, “Malik”, 20, “Pune”);
myBrother.fullName();                              // returns Shahid Malik
```

JavaScript Prototypes

JavaScript Prototypes

- All JavaScript objects inherit the properties and methods from their prototype.
- Objects created using an object literal, or with new Object(), inherit from a prototype called Object.prototype.
- Objects created with new Date() inherit the Date.prototype.
- The Object.prototype is on the top of the prototype chain.

- **Adding Properties and methods to Objects :**

a. Sometimes you want to add new properties (or methods) to an existing object.

Solution is simple :
Example : myFather.nationality = "Indian";

b. Sometimes you want to add new methods to an existing objects of a given type.
Solution is simple :
Example : myFather.getAge = function () { return this.age; };

c. Sometimes you want to add new properties (or methods) to an object prototype.
Solution is not easy :
You cannot add a new property to a prototype the same way as you add a new property to an existing object, because the prototype is not an existing object.

Solution 1 : To add a new property to a constructor, you must add it to the constructor function.
Example :
if you want to add the below property to the existing prototype :

Person.nationality = “Indian”

then the updated constructor will look like :

```
function Person(first, last, age, city) {
        this.firstName = first;
        this.lastName = last;
        this.age = age;
        this.city = city;
        this.nationality ="Indian";
}
```

Solution 2 :
Using the prototype Property

The JavaScript prototype property allows you to add new properties to an existing prototype:

Example :

```
function Person(first, last, age, city) {
        this.firstName = first;
        this.lastName = last;
        this.age = age;
        this.city = city;
}

Person.prototype.nationality = “Indian”;
```

Note : The JavaScript prototype property also allows you to add new methods to an existing prototype:

Example :

```
function Person(first, last, age, city) {
        this.firstName = first;
        this.lastName = last;
        this.age = age;
        this.city = city;
}

Person.prototype.fullName = function () {
                return this.firstName + “ ” + this.lastName;
        };
```

JSON

JSON :

JSON stands for **J**ava**S**cript **O**bject **N**otation

JSON is lightweight data interchange format.

JSON is language independent.

JSON is "self-describing" and easy to understand.

Note :
- The JSON syntax is derived from JavaScript object notation syntax, but the JSON format is text only.
- Code for reading and generating JSON data can be written in any programming language.

Example :

```
{
"employees":[

      {"firstName":"John",      "lastName":"Doe"},
      {"firstName":"Anna",      "lastName":"Smith"},
      {"firstName":"Peter",     "lastName":"Jones"}
      ]
}
```