JavaScript Basics

Nikhil Jaunjal

INTRODUCTION

Why To learn JavaScript?

- JavaScript is the programming language of the web. It's one of the most popular and in demand skills in today's job market for good reason.
- JavaScript not only enables you to add powerful interactions to websites, but is also the foundation of a lot of commonly used libraries (like jQuery) and frameworks (like AngularJS, ReactJS and NodeJS).
- As a web developer, it is essential that you have a solid understanding of this versatile language.

What is JavaScript?

- JavaScript is an interpreted, object-based scripting 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.
- The JavaScript language uses a syntax like that of C, and supports structured constructs, such as if...else, for, and do...while. Braces ({}) are used to delimit statement blocks.
- The language supports various data types, including String, Number, Boolean, Object, and Array. It includes support for enhanced date features, trigonometric functions, and regular expressions.
- JavaScript is a loosely typed language, which means you do not declare the data types of variables explicitly.
- In many cases JavaScript performs conversions automatically when they are needed. For example, if you add a number to an item that consists of text (a string), the number is converted to text.

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

- **Client-Side execution**: No matter where you host JavaScript, Execute always on client environment to save a bandwidth and make execution process fast.
- **User Interface Interactivity**: JavaScript used to fill web page data dynamically such as drop-down list for a Country and State. Base on selected Country, State drop down list dynamically filled. Another one is Form validation, missing/incorrect fields you can alert to users using alert box.
- **Rapid Development**: JavaScript syntaxes are easy and flexible for the developers. JavaScript small bit of code you can test easily on Console Panel (inside Developer Tools) at a time browser interpret return output result. In-short easy language to get pick up in development.
- **Browser Compatible**: The biggest advantages to a JavaScript having an ability to support all modern browser and produce the same result.

Disadvantages

- **Code Always Visible**: The biggest disadvantage is code always visible in developer mode, anyone can view the code.
- **Bit of Slow execute**: No matter how much fast JavaScript interpret, JavaScript DOM (Document Object Model) is slow with HTML.

SYNTAX

Initialization

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

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>
```

Case sensitivity

JavaScript is case sensitivity language. For eg:

```
<script language="javascript" type="text/javascript">
        var a;
        var A;                // var a and var A are two different variable
</script>
```

Comments

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

```
<script language="javascript" type="text/javascript">
        // this is single line comment
        /*
        * this is
        * multi-line
        * comment
         */
 </script>
```

Data Types

There are three primary data types, two composite data types, and two special data types.

Primary data types: (primitive)

- String
- Number
- Boolean

Composite data types: (reference)

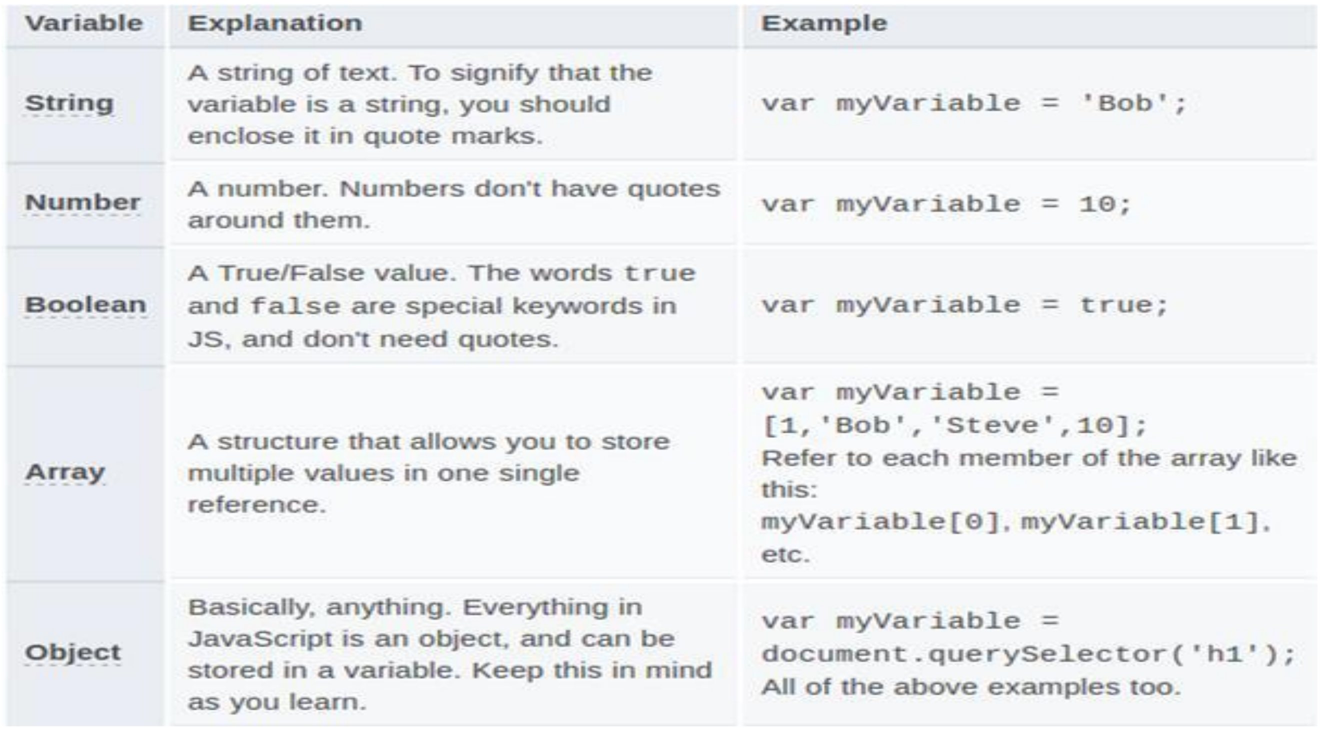
- Object
- Array

Special data types

- Null
- Undefined

Variables

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



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.

OPERATORS

JavaScript has a full range of operators, including arithmetic, logical, bitwise, assignment, as well as some miscellaneous operators.

Computational Operators

Description	Symbol
Unary negation	-
Increment	++
Decrement	--
Multiplication	*
Division	/
Modulus arithmetic	%
Addition	+
Subtraction	-

Logical Operators

Description	Symbol
Logical NOT	!
Less than	<
Greater than	>
Less than or equal to	<=
Greater than or equal to	>=
Equality	==
Inequality	!=
Logical AND	&&
Logical OR	||
Conditional (ternary)	?:
Comma	,
Strict Equality	===
Strict Inequality	!==

Bitwise Operators

Description	Symbol
Bitwise NOT	~
Bitwise Left Shift	<<
Bitwise Right Shift	>>
Unsigned Right Shift	>>>
Bitwise AND	&
Bitwise XOR	^
Bitwise OR	|

Assignment Operators

Description	Symbol
Assignment	=
Compound Assignment	*OP*= (such as += and &=)

Miscellaneous Operators

Description	Symbol
delete	delete
typeof	typeof
void	void
instanceof	instanceof
new	new
in	in

OPERATOR PRECEDENCE

Operator precedence describes the order in which operations are performed when an expression is evaluated.
Operations with a higher precedence are performed before those with a lower precedence.
For example, multiplication is performed before addition.

Operators (Highest to Lowest Order)	Description
.	Field access, array indexing, function calls, and expression grouping
++ -- - ~ ! delete new typeof void	Unary operators, return data type, object creation, undefined values
* / %	Multiplication, division, modulo division
+ - +	Addition, subtraction, string concatenation
<< >> >>>	Bit shifting
< <= > >= instanceof	Less than, less than or equal, greater than, greater than or equal, instanceof
== != === !==	Equality, inequality, strict equality, and strict inequality
&	Bitwise AND
^	Bitwise XOR
|	Bitwise OR
&&	Logical AND
||	Logical OR
?:	Conditional
= OP=	Assignment, assignment with operation (such as += and &=)
,	Multiple evaluation

FUNCTIONS

- A JavaScript function is defined with the ***function*** keyword, followed bya **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: **{}**

Syntax and Examples

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

Anonymous Function

An anonymous function is a function that was declared without any named identifier to refer to it. As such, an anonymous function is usually not accessible after its initial creation.

```
function (x, y) {                           // anonymous function
        return x + y;
}
```

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

```
var add = function (x, y) {
        return x +y;
}
console.log("Addition is : " + add(2,3));      // output: 5
```

NOTE: You can read the other topics in functions such as closures, call and apply.

OBJECTS

Definition

- Objects are variables containing variables.
- Variables can contain **single values** whereas objects are special variables which contain **many values**. E.g.

```
//variables
var person = "John Doe";

// objects
var person = {
        firstName: "John",
        lastName: "Doe",
        age: 25 ,
        city: “Mumbai”
        };
```

Object Properties

The named values, in JavaScript objects, are called **properties**. i.e.

```
firstName: "John", lastName:"Doe", age: 25 , city: “Mumbai”
```

Note: Objects written in name-value pairs are similar to :-Associative arrays in PHP, Dictionaries in Python, Hash maps in Javaand Hashes in Ruby and Perl

Creating Objects

With JavaScript, you can define and create your own objects.
There are different ways to create new objects:

1. Using an **object literal**.
2. Using the keyword **new**.
3. Define an function, and then create objects of the constructed type.

1. Using an Object literal

This is the easiest way to create a JS object. E.g.

```
var person = {
        firstName: "John",
        lastName: "Doe",
        age: 25,
        city: “Mumbai”
};
```

2. Using the keyword '*new*'

First create the variable using the new keyword and later properties one by one. E.g.

```
var person = new Object();

person.firstName = "John";
person.lastName = "Doe";
person.age = 25;
person.city = “Mumbai”;
```

3. Using a function, and then create objects of the constructed type

- 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 function as object constructor. E.g.

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

var myBrother = new person(“John”, “Doe”, 20, “Mumbai”);
var mySister = new person(“Sachin”, “Tendulkar”, 36, “Mumbai”);
```

Built-in Constructors

```
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.

- There is no reason to use new Array() instead of use array literals instead: []
- There is no reason to use new RegExp() instead of use pattern literals instead: /()/
- There is no reason to use new Function() instead of use function expressions instead: function () {}.
- There is no reason to use new Object() instead of use object literals instead: {}

```
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
```

Accessing properties

Syntax:

```
1. objectName.propertyName          // person.firstName
2. objectName[“propertyName”]       // person[“firstName”]
3. objectName[expression]           // var x= “firstName”; person[x]
```

- Adding new property to an object

```
1. var mySister = new person(“Jenny”, “Doe”, 25, “Mumbai”);
2. mySister.gender= “female”;
```

- Deleting property from an object

```
1. var mySister = new person(“Jenny”, “Doe”, 25, “Mumbai”);
2. detete mySister.age;
```

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

Object Methods

A JavaScript method is a property containing a function definition. E.g.

```
function person (firstName, lastName, age, city) {
        this.firstName = firstName;
        this.lastName = lastName;
        this.age = age;
        this.city = city;
 //property containing a function definition
        this.fullName : function () {
                return this.firstName + " " + this.lastName;
         }
 }
```

Accessing Object Method

You can create object of defined constructor and access the method. Syntax :

```
ObjectName.methodName();
```

Example :

```
// create person object

        var myBrother = new Person("John", "Doe", 25, "Mumbai");
        myBrother.fullName();     // returns John Doe
```

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 :

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

references -

https://www.javascript.com

https://docs.microsoft.com/en-us/scripting/javascript/javascript-language-reference