Introduction to Java Script

By

Vaibhav P. Vasani

Assistant Professor

Department of Computer Engineering

K. J. Somaiya College of Engineering

Somaiya Vidyavihar University

vaibhav.vasani@gmail.com

Introduction to Java Script

scripts

created to "make web pages alive".

Mocha

LiveScript

specification called ECMAScript

- Most popular language
- Language of the Web
- easy to learn
- Client side validation
- Manipulating HTML Pages
- User Notifications
- Back-end Data Loading (AJAX)
- Presentations (RevealJS and BespokeJS libraries)
- Server Applications (NodeJS)

- “safe” programming language (Not provide low-level access to memory or CPU as runs on browser)
- lightweight, interpreted programming language
- Designed for creating network-centric applications.
- Integration with JAVA/HTML/ETC
- Open and cross-platform
- Less server interaction
- Immediate feedback to the visitors
- Increased interactivity
- Richer interfaces

Javascript frameworks and libraries available

- Ember.js
- Meteor
- Mithril
- Node.js
- Polymer
- Aurelia
- Backbone.js
- Angular
- React
- jQuery
- Vue.js
- Ext.js

the JavaScript engine

The browser has an embedded engine sometimes called a "JavaScript virtual machine".

V8 – in Chrome and Opera.
SpiderMonkey – in Firefox.

Working Engines

- The engine (embedded if it's a browser) reads ("parses") the script.
- Then it converts ("compiles") the script to the machine language.
- And then the machine code runs, pretty fast.

For instance, in-browser JavaScript is able to:

- Add new HTML to the page, change the existing content, modify styles.
- React to user actions, run on mouse clicks, pointer movements, key presses.
- Send requests over the network to remote servers, download and upload files (so called AJAX and COMET technologies).
- Get and set cookies, ask questions to the visitor, show messages.
- Remember the data on the client-side ("local storage").

Restrictions of Java Script

- May not read/write arbitrary files on the hard disk, copy them or execute programs. It has no direct access to OS functions. (Modern browsers allow through <Input> tag)
- No camera/microphone and other devices, but they require a user's explicit permission.
- Different tabs/windows generally do not know about each other.
- Doesn't have any multi-threading or multiprocessor capabilities

There are at least *three* great things about JavaScript:

- Full integration with HTML/CSS.
- Simple things are done simply.
- Support by all major browsers and enabled by default.

Languages, which are converted to JavaScript before they run in the browser.

- **CoffeeScript** is a “syntactic sugar” for JavaScript. It introduces shorter syntax, allowing us to write clearer and more precise code. Usually, Ruby devs like it.
- **TypeScript** is concentrated on adding “strict data typing” to simplify the development and support of complex systems. It is developed by Microsoft.
- **Flow** also adds data typing, but in a different way. Developed by Facebook.
- **Dart** is a standalone language that has its own engine that runs in non-browser environments (like mobile apps), but also can be transpiled to JavaScript. Developed by Google.
- **Brython** is a Python transpiler to JavaScript that enables the writing of applications in pure Python without JavaScript.
- **Kotlin** is a modern, concise and safe programming language that can target the browser or Node.

Code editors

- IDE
  - Visual Studio Code (cross-platform, free).
  - WebStorm (cross-platform, paid).
- Lightweight editors
  - Atom (cross-platform, free).
  - Visual Studio Code (cross-platform, free).
  - Sublime Text (cross-platform, shareware).
  - Notepad++ (Windows, free).
  - Vim and Emacs are also cool if you know how to use them.

JavaScript - Syntax

```
<script ...>
   JavaScript code
</script>
```

The <script> tag alerts the browser program to start interpreting all the text between these tags as a script.

```
<script language = "javascript" type = "text/javascript">
   JavaScript code
</script>
```

First Sample Program

```
<html>
  <body>
    <script language = "javascript" type = "text/javascript">
      <!--
        document.write("Hello World!")
      //-->
    </script>
  </body>
</html>
```

Hello World!

Whitespace and Line Breaks

- Ignores spaces, tabs, and newlines
- Free to format and indent programs

Semicolons

- Semicolons are Optional

```
<script language = "javascript" type =
"text/javascript">
  <!--
    var1 = 110
    var2 = 220
  //-->
</script>
```

Optional If placed on a separate line.

```
<script language = "javascript" type =
"text/javascript">
  <!--
    var1 = 110; var2 = 220;
  //-->
</script>
```

Case Sensitive

- JavaScript is a case-sensitive language.
- **STUD** and **STUDENT** different meanings

Comments

- // ←A single line Comment
- /* and */ ←A multi line Comment
- HTML comment opening sequence <!-- ←A single line Comment
- <!-- //--> ←A multi line Comment

```
<script language = "javascript" type =
"text/javascript">
  <!--
     // This is a comment
     /*
     * This is a multi-line comment
     */
   //-->
</script>
```

Warning for Non-JavaScript Browsers

```
<html>
  <body>
    <script language = "javascript" type = "text/javascript">
      <!--
        document.write("Hello World!")
      //-->
    </script>
    <noscript>
      Sorry... Bro You have to enable JavaScript.
    </noscript>
  </body>
</html>
```

if the user's browser does not support JavaScript or JavaScript is not enabled, then the message from </noscript> will be displayed on the screen.

Where to write Script

- Script in <head>...</head> section.
- Script in <body>...</body> section.
- Script in <body>...</body> and <head>...</head> sections.
- Script in an external file and then include in <head>...</head> section.

JavaScript in <head>...</head> section

```
<html>
  <head>
    <script type = "text/javascript">
      <!--
        function sayHello() {
          alert("Hello MERN Stack Developers")
        }
      //-->
    </script>
  </head>
  <body>
    <input type = "button" onclick = "sayHello()" value = “Click Me" />
  </body>
</html>
```

JavaScript in <body>...</body> section

```
<html>
  <head>
  </head>
  <body>
    <script type = "text/javascript">
      <!--
        document.write("Hello TY Students")
      //-->
    </script>
    <p>This is web page body Section  </p>
  </body>
</html>
```

JavaScript in <body> and <head> Sections

```
<html>
  <head>
    <script type = "text/javascript">
      <!--
        function sayHello() {
          alert("This is MERN Lab")
        }
      //-->
    </script>
  </head>
  <body>
    <script type = "text/javascript">
      <!--
        document.write("Welcome TY students")
      //-->
    </script>
    <input type = "button" onclick = "sayHello()" value = "Click Me" />
  </body>
</html>
```

JavaScript in External File

```
<html>
  <head>
    <script type = "text/javascript" src = "filename.js"
></script>
  </head>

  <body>
    .......
  </body>
</html>
```

```
function sayHello() {
  alert("Hello World")
}
```

JavaScript Datatypes

Three primitive data types −

- **Numbers,** eg. 123, 120.50 etc.
- **Strings** of text e.g. "This text string" etc.
- **Boolean** e.g. true or false.

Two trivial data types, **null** and **undefined**

Composite data type known as **object**.

JavaScript Variables

- <script type = "text/javascript">
- <!--
- var Name;
- var age;
- //-->
- </script>

```
<script type = "text/javascript">
  <!--
    var Age, name;
  //-->
</script>
```

Variable Initialization

- <script type = "text/javascript">
- <!--
- var name = “krishna";
- var money;
- money = 112000.50;
- //-->
- </script>

JavaScript Variable Scope

- **Global Variables** − A global variable has global scope which means it can be defined anywhere in your JavaScript code.
- **Local Variables** − A local variable will be visible only within a function where it is defined. Function parameters are always local to that function.

```
<html>
  <body onload = checkscope();>
    <script type = "text/javascript">
      <!--
        var Var1 = "global";      // Declare a global variable
        function checkscope( ) {
          var Var1= "local";    // Declare a local variable
          document.write(Var1);
        }
      //-->
    </script>
  </body>
</html>
```

Output ?

JavaScript Variable Names

- Don’t Use any of the JavaScript reserved keywords
- Should not start with a numeral (0-9).
- begin with a letter or an underscore character.
- variable names are case-sensitive.

Reserved Words

abstract	else	instanceof	switch
boolean	enum	int	synchronized
break	export	interface	this
byte	extends	long	throw
case	false	native	throws
catch	final	new	transient
char	finally	null	true
class	float	package	try
const	for	private	typeof
continue	function	protected	var
debugger	goto	public	void
default	if	return	volatile
delete	implements	short	while
do	import	static	with
double	in	super	

Operator

- Arithmetic Operators
- Comparison Operators
- Logical (or Relational) Operators
- Assignment Operators
- Conditional (or ternary) Operators

typeof Operator

- a unary operator

Type	String Returned by typeof
Number	"number"
String	"string"
Boolean	"boolean"
Object	"object"
Function	"function"
Undefined	"undefined"
Null	"object"

if statement

```
<html>
  <body>
    <script type = "text/javascript">
      <!--
        var age = 20;
        if( age > 18 ) {
          document.write("<b>Qualifies for Computer Games</b>");
        }
      //-->
    </script>
</body>
</html>
```

Output ?

if...else statement

```
<html>
  <body>
    <script type = "text/javascript">
      <!--
        var age = 15;

        if( age > 18 ) {
          document.write("<b>Qualifies for Playing Computer Games</b>");
        } else {
          document.write("<b>Does not qualify </b>");
        }
      //-->
    </script>
</body>
</html>
```

Output ?

if...else if... statement

```
<html>
  <body>
    <script type = "text/javascript">
      <!--
        var book = “DBMS";
        if( book == “TOC" ) {
          document.write("<b>TOC  Book</b>");
        } else if( book == “DTE" ) {
          document.write("<b>DTE Book</b>");
        } else if( book == “MERN" ) {
          document.write("<b>MERN  Book</b>");
        } else {
          document.write("<b>Unknown Book</b>");
        }
      //-->
    </script>
</body>
<html>
```

Output ?

Switch Case

```
<html>
  <body>
    <script type = "text/javascript">
      <!--
        var grade = 'A';
        document.write("Entering switch block<br />");
        switch (grade) {
          case 'A': document.write("Good job<br />");
          break;
          case 'B': document.write("Pretty good<br />");
          break;
          case 'C': document.write("Passed<br />");
          break;
          case 'D': document.write("Not so good<br />");
          break;
          case 'F': document.write("Failed<br />");
          break;
          default: document.write("Unknown grade<br />")
        }
        document.write("Exiting switch block");
      //-->
    </script>
</body>
</html>
```

Students Needs to Reply

- What Happen if we don’t write break statement in code ?

While Loops

```
<html>
  <body>
    <script type = "text/javascript">
      <!--
        var count = 0;
        document.write(" Loop Starti");
        while (count < 10) {
          document.write("Count : " + count + "<br />");
          count++;
        }
        document.write("Loop stopped!");
      //-->
    </script>
</body>
</html>
```

Output ?

What is difference between While and Do While

- Students Needs to reply

The do...while Loop

```
<html>
  <body>
    <script type = "text/javascript">
      <!--
        var count = 0;
        document.write("Starting Loop" + "<br />");
        do {
          document.write("Count : " + count + "<br />");
          count++;
        }
        while (count < 5);
        document.write ("Loop stopped!");
      //-->
    </script>
  </body>
</html>
```

Output ?

For Loop

- Students reply

```
<html>
  <body>
    <script type = "text/javascript">
      <!--
        var count;
        document.write("Starting Loop" + "<br />");
        for(count = 0; count < 10; count++) {
          document.write("Count : " + count );
          document.write("<br />");
        }
        document.write("Loop stopped!");
      //-->
    </script>
  </body>
</html>
```

Output ?

Which Loop should we Use ?

- Reply from Students

JavaScript *for...in* loop***

- The **for...in** loop is used to loop through an object's properties.

```
<html>
  <body>
    <script type = "text/javascript">
      <!--
        var Names;
        document.write("MERN Students of <br /> ");
        for (Names in FSDMERN) {
          document.write(Names);
          document.write("<br />");
        }
      //-->
    </script
  </body>
</html>
```

Output ?

Break and continue

- Students should Discuss

label

- Students should Discuss

```
<html>
  <body>
    <script type = "text/javascript">
      <!--
        document.write("Entering the loop!<br /> ");
        outerloop:        // This is the label name
        for (var i = 0; i < 5; i++) {
          document.write("Outerloop: " + i + "<br />");
          innerloop:
          for (var j = 0; j < 5; j++) {
            if (j > 3 ) break ;          // Quit the innermost loop
            if (i == 2) break innerloop;  // Do the same thing
            if (i == 4) break outerloop;  // Quit the outer loop
            document.write("Innerloop: " + j + " <br />");
          }
        }
        document.write("Exiting the loop!<br /> ");
      //-->
    </script>
  </body>
</html>
```

Output ?

Functions

- What is function ?
- Need of Function ?

Syntax

```
<script type = "text/javascript">
   <!--
      function functionname(parameter-list) {
         statements
      }
   //-->
</script>
```

```
<script type = "text/javascript">
   <!--
      function classHello() {
         alert("Hello MERN Stack Developer");
      }
   //-->
</script>
```

Calling a Function

```
<html>
  <head>
<script type = "text/javascript">
  <!--
     function classHello() {
        alert("Hello MERN Stack Developer");
     }
  //-->
</script>
  </head>
  <body>
     <p>Click the following button to call the function</p>
     <form>
        <input type = "button" onclick = " classHello()" value = “Click Me">
     </form>
</body>
</html>
```

Function Parameters

```
<html>
  <head>
    <script type = "text/javascript">
      function fun1(name, age) {
        document.write (name + " is " + age + " years old.");
      }
    </script>
  </head>
  <body>
    <p>Click the following button to call the function</p>
    <form>
      <input type = "button" onclick = "fun1('Manan', 3.5)" value = "Click Me">
    </form>
  </body>
</html>
```

The return Statement

- What is Use of Return ?
- Students Needs to discuss

- 5.html

Event

- Students View over Event

Event

- Interaction with HTML is handled through events
- When the page loads, it is called an event.
- When the user clicks a button
- pressing any key, closing a window, resizing a window, etc.

onclick Event Type

onsubmit Event Type

- to submit a form.
- form validation can be kept against this event type.
  - calling a **validate()** function before submitting a form data to the webserver.
  - If **validate()** function returns true, the form will be submitted, otherwise not
-

onmouseover and onmouseout

- The **onmouseover** event triggers when you bring your mouse over any element and the **onmouseout** triggers when you move your mouse out from that element.
- 6.html

HTML 5 Standard Events

- Event list

Thank You