JavaScript

- JavaScript is a scripting language most often used for client-side web development.
- JavaScript is a high-level, dynamic and interpreted programming language.
- It has been standardized in the ECMAScript language specification.
- The majority of websites employ it, and all modern web browsers support it without the need for plug-ins.
- JavaScript is one of the 3 languages all web developers must learn:
  1. HTML to define the content of web pages
  2. CSS to specify the layout of web pages
  3. JavaScript to program the behavior of web pages
- JavaScript was designed to add interactivity to HTML pages.
- A JavaScript is usually embedded directly into HTML pages.

JavaScript

- JavaScript is a lightweight and most commonly used language 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.
- JavaScript is designed for creating network-centric applications.
- JavaScript is used to create interactive websites. It is mainly used for:
  1. Client-side validation
  2. Dynamic drop-down menus
  3. Displaying data and time
  4. Displaying popup windows and dialog boxes (like alert dialog box, confirm dialog box and prompt dialog box)
  5. Displaying clocks etc.

Advantages of JavaScript

- **Client-Side execution:** Code is execute on client processor instead of web server thus saving the 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 & State. Base on selected Country, State drop down list dynamically filled. Another one is Form validation, missing/incorrect fields we can alert users using alert box.
- **Rapid Development:** JavaScript syntax's are easy and flexible for the developers. JavaScript small bit of code you can test easily on Console Panel at a time browser interpret return output result.
- **Browser Compatible:** The biggest advantages to a JavaScript having a ability to support all modern browser and produce the same result. JavaScript is platform independent language.
- **Interpreted languages:** JavaScript is an interpreted language. It requires no compilation process so no compiler is required. The browser interprets JavaScript as it HTML tags.

Advantages of JavaScript

- **Easy to learn:** The syntax of JavaScript is very easy. Any person can learn it very easily and use it to develop dynamic and attractive websites.
- **Easy to Debug and Test:** JavaScript code is interpreted line by line. The errors are indicated along with line number. It is very easy to find error in the code, correct it and test it gain.
- **Event-Based Programming:** JavaScript is an event-based language. It means that different code segment are executed when certain event occur. For example, a code segment may execute when the user click a button or moves a mouse over an object etc.
- **Procedural Capabilities:** JavaScript provides all capabilities of a procedural language. It provides condition checking, loops and branching facilities that can be executed in a web page.
- **Rich interfaces:** Drag and drop components or slider may give a rich interface to your site visitors.

Limitations of JavaScript

- **Security Issues:** JavaScript is explicitly added to web pages and client browsers, it can exploit the user's system. Malicious code can be executed on client's machine.
- **Javascript rendering varies:** Different layout engines may render JavaScript differently resulting in inconsistency in terms of functionality and interface.
- **Code always visible :** The biggest disadvantages is code always visible to everyone anyone can view JavaScript code.
- **Bit of slow execute :** No matter how much fast JavaScript interpret, JavaScript DOM (Document Object Model) is slow and will be a never fast rendering with HTML.

Limitations of JavaScript

- **Stop render :** JavaScript single error can stop to render with entire site. However browsers are extremely tolerant of JavaScript errors.
- **No reading and writing of files:** Client-side JavaScript does not allow the reading or writing of files. This has been kept for security reason.
- **Can't be used for networking applications:** JavaScript cannot be used for networking applications because there is no such support available.
- **No Multiprocessor and multithreading capabilities:** JavaScript doesn't have any multithreading or multiprocessor capabilities.

Writing a JavaScript Program

- The Web browser runs a JavaScript program when the Web page is first loaded, or in response to an event.
- JavaScript programs can either be placed directly into the **HTML** file or they can be saved in **external files**.
- Placing a program in an external file allows you to hide the program code from the user.
- Source code placed directly in the HTML file can be viewed by anyone.
- A JavaScript program can be placed anywhere within the HTML file.
- Many programmers favor placing their programs between **<head>** tags in order to separate the programming code from the Web page content and layout.
- Some programmers prefer placing programs within the **<body>** of the Web page at the location where the program output is generated and displayed.

A Simple Script

```
<html>
<head><title>First JavaScript
   Page</title></head>
<body>
<h1>First JavaScript Page</h1>
<script type="text/javascript">
   document.write("<hr>");
   document.write("BCA Students");
   document.write("<hr>");
</script>
</body>
</html>
```

- JavaScript commands and names are case-sensitive. In JavaScript, semicolons are optional. However, semicolons are required if we want to put more than one statement on a single line.

JavaScript in <body> ...</body> Section

```
<html>
<head><title>First JavaScript
   Page</title></head>
<body>
<h1>First JavaScript Page</h1>
<script type="text/javascript">
  document.write("<hr>");
  document.write("Hello Students");
  document.write("<FONT COLOR='magenta'><H1>
                  Welcome to JavaScript
                  Programming!!!</H1></FONT>");
 document.write("<hr>");
</script>
</body>
</html>
```

JavaScript in <head> ...</head> Section

- If you want to have a script run on some event, such as when a user clicks somewhere, then you will place that script in the head as follows.

```
<html>
<head>
<script type="text/javascript">
  function sayHello() { alert("Hello World")}
</script>
</head>
<body>
Click here for the result
<input type="button" onclick="sayHello()"
  value="Say Hello" />
</body>
</html>
```

JavaScript in <head> and <body> Sections

```
<html>
<head>
  <script type="text/javascript">
     function sayHello() { alert("Hello
                                Everyone")}
  </script>
</head>
<body>
  <script type="text/javascript">
     document.write("Hello world")
  </script>
  <input type="button" onclick="sayHello()"
   value="Say Hello" />
</body>
</html>
```

Embedding JavaScript

```
<html>
<head><title>First JavaScript
   Program</title></head>
<body>
<script type="text/javascript"
        src="second.js"></script>
</body>
</html>
```

```
document.write("<hr>");
document.write("Hello students");
document.write("<hr>");
```

alert(), confirm(), and prompt()

```
<script type="text/javascript">
alert("This is an Alert method");
confirm("Are you OK?");
prompt("What is your name?");
prompt("How old are you?","20");
</script>
```

alert() and confirm()

```
alert("Text to be displayed");
```

- Display a message in a dialog box.
- The dialog box will block the browser.

```
var answer = confirm("Are you sure?");
```

- Display a message in a dialog box with two buttons: "OK" or "Cancel".
- confirm() returns true if the user click "OK". Otherwise it returns false.

prompt()

```
prompt("What is your student id number?");
prompt("What is your name?", "No name");
```

- Display a message and allow the user to enter a value
- The second argument is the "default value" to be displayed in the input textfield.
- Without the default value, "undefined" is shown in the input textfield.
- If the user click the "OK" button, prompt() returns the value in the input textfield as a string.
- If the user click the "Cancel" button, prompt() returns null.

Data Types

- Primitive data types
  - Number: integer & floating-point numbers
  - Boolean: true or false
  - String: a sequence of alphanumeric characters
- Composite data types (or Complex data types)
  - Object: a named collection of data
  - Array: a sequence of values (an array is actually a predefined object)
- Special data types
  - Null: the only value is "null" – to represent nothing.
  - Undefined: the only value is "undefined" – to represent the value of an uninitialized variable

Data types

- In JavaScript there are three primitive data types:-
- **Numbers**, e.g., **711**, **450.76** etc.
- **Strings** of text, e.g. "**This is my classroom**" etc.
- **Boolean**, e.g. **true** or **false**.
- JavaScript also defines two trivial data types, **null** and **undefined**, each of which defines only a single value.
- In addition to these primitive data types, JavaScript supports a composite data type known as **object**. For Example:-
- var person = {
  firstName : "John",
  lastName : "Doe",
  age : 50,
  eyeColor : "blue" };
- Java does not make a distinction between integer values and floating point values. All numbers in JavaScript are represented as floating-point values.

Undefined & Null

- In computer programs, variables are often declared without a value. The value can be something that has to be calculated, or something that will be provided later, like user input.
- A variable declared without a value will have the value undefined.

```
var carName;
```

- The variable carName will have the value undefined after the execution of this statement.

```
var person = undefined;     // Value is undefined, type is undefined
```

- In JavaScript null is "nothing". It is supposed to be something that doesn't exist.
- Unfortunately, in JavaScript, the data type of null is an object.
- We can empty an object by setting it to null.

```
var person = null;        // Value is null, but type is still an object
```

Variables

- Variables can be thought of as named containers.
- We can place data into these containers and then refer to the data simply by naming the container.
- Before we use a variable in a JavaScript program, we must declare it. Variables are declared with the var keyword as follows:-

```
var Month;
Month = "December";
var Month = "December";
var sum, average, name;
```

- Use the var keyword only for declaration or initialization, once for the life of any variable name in a document. We should not re-declare same variable twice.
- JavaScript is untyped language. This means that a JavaScript variable can hold a value of any data type. We don't have to tell JavaScript during variable declaration what type of value the variable will hold. The value type of a variable can change during the execution of a program and JavaScript takes care of it automatically.
-

```
var x;                  // Now x is undefined
var x = 5;              // Now x is a Number
var x = "John";         // Now x is a String
```

Variable Scope

- The scope of a variable is the region of our program in which it is defined.
- JavaScript variables have only two scopes:-
- **Global Variables:** A global variable has global scope which means it can be defined anywhere in our 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.
- Within the body of a function, a local variable takes precedence over a global variable with the same name.

```
<script type="text/javascript">
    var myVar = "global";          // Declare a global variable
    function checkscope( ) {
        var myVar = "local";       // Declare a local variable
        document.write(myVar); }
</script>
```

- It will produce the following result: Local

Identifiers

- All JavaScript variables must be identified with unique names.
- These unique names are called identifiers.
- Identifiers can be short names (like x and y) or more descriptive names (age, sum, totalVolume).
- The general rules for constructing names for variables (unique identifiers) are:
  - Names can contain letters, digits, underscores, and dollar signs.
  - Names must begin with a letter.
  - Names are case sensitive (y and Y are different variables).
  - Reserved words (like JavaScript keywords) cannot be used as names.

Reserved Words

A list of all the reserved words in JavaScript are given in the following table. They cannot be used as JavaScript variables, functions, methods, loop labels, or any object names.

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	

Operators

Arithmetic Operators

Operator	Description	Example	Result
+	Addition	x=2 y=2 x+y	4
-	Subtraction	x=5 y=2 x-y	3
*	Multiplication	x=5 y=4 x*y	20
/	Division	15/5 5/2	3 2,5
%	Modulus (division remainder)	5%2 10%8 10%2	1 2 0
++	Increment	x=5 x++	x=6
--	Decrement	x=5 x--	x=4

Operators – 2

Assignment Operators

Operator	Example	Is The Same As
=	x=y	x=y
+=	x+=y	x=x+y
-=	x-=y	x=x-y
*=	x*=y	x=x*y
/=	x/=y	x=x/y
%=	x%=y	x=x%y

Operators - 3

Comparison Operators

Operator	Description	Example
==	is equal to	5==8 returns false
===	is equal to (checks for both value and type)	x=5 y="5" x==y returns true x===y returns false
!=	is not equal	5!=8 returns true
>	is greater than	5>8 returns false
<	is less than	5<8 returns true
>=	is greater than or equal to	5>=8 returns false
<=	is less than or equal to	5<=8 returns true

Operators - 4

Logical Operators

Operator	Description	Example
&&	and	x=6 y=3 (x < 10 && y > 1) returns true
||	or	x=6 y=3 (x==5 || y==5) returns false
!	not	x=6 y=3 !(x==y) returns true

Miscellaneous Operators

Conditional Operator (? :)

The conditional operator first evaluates an expression for a true or false value and then executes one of the two given statements depending upon the result of the evaluation.

S.No	Operator and Description
1	**? : (Conditional)** If Condition is true? Then value X : Otherwise value Y

typeof Operator

The **typeof** operator is a unary operator that is placed before its single operand, which can be of any type. Its value is a string indicating the data type of the operand.

The *typeof* operator evaluates to "number", "string", or "boolean" if its operand is a number, string, or boolean value and returns true or false based on the evaluation.

Here is a list of the return values for the **typeof** Operator.

Type	String Returned by typeof
Number	"number"
String	"string"
Boolean	"boolean"
Object	"object"

THANK YOU