

Introduction

- JS is the programming language of the web.
- JavaScript is a very powerful client-side scripting language. JavaScript is used mainly for enhancing the interaction of a user with the webpage.
- Make web page dynamic.
- Runs on client-side.
- Interpreted language with object-oriented capabilities (ES6).
- The programs in this language are called scripts and saves with extension ".js". They can be written right in the HTML and execute automatically as the page loads.
- JavaScript is a Dynamically Typed Language.
- JavaScript is Weakly(loosely) Typed Language & case-sensitive language.
- The main advantage of JavaScript is that all modern web browsers support JavaScript.
- JavaScript runs on any operating system including Windows, Linux or Mac using the web browser.
- JavaScript is an object-based scripting language which is lightweight and cross-platform.
- JavaScript is not a compiled language, but it is a interpreted language. The JavaScript interpreter (embedded in the browser) is responsible for translating the JavaScript code for the web browser.
- JavaScript has no connectivity with Java programming language. The name was suggested and provided in the times when Java was gaining popularity in the market.
- JavaScript is designed to add interactivity and dynamic effects to the web pages by manipulating the content returned from a web server.
- JavaScript is officially maintained by ECMA (European Computer Manufacturers Association) as ECMAScript. ECMAScript 6 (or ES6) is the latest major version of the ECMAScript standard.
- JavaScript is so hard to learn because it's an asynchronous programming language. It's also single-threaded, which means it uses its asynchronous nature in a radically different way than most other programming languages.

History

- a) Developed by Brendan Eich in 1995 at Netscape in 10 days.
- b) JS has other names like Mocha, ES6, JAVASCRIPT, LiveScript.

Features of JavaScript

There are following features of JavaScript: -

- All popular web browsers support JavaScript as they provide built-in execution environments.
- JavaScript follows the syntax and structure of the C programming language. Thus, it is a structured programming language.
- JavaScript is a weakly typed language, where certain types are implicitly cast (depending on the operation).
- JavaScript is an object-oriented programming language that uses prototypes rather than using classes for inheritance.
- It is a light-weighted and interpreted language.
- It is a case-sensitive language.
- JavaScript is supportable in several operating systems including, Windows, macOS, etc.
- It provides good control to the users over the web browsers.

Application of JavaScript

Basically, JavaScript is used to create interactive websites. It is mainly used for:

- Client-side validation,
- Dynamic drop-down menus,
- Displaying date and time,
- Displaying pop-up windows and dialog boxes (like an alert dialog box, confirm dialog box and prompt dialog box),
- Displaying clocks etc.
- Dynamic Animations.
- Develop User Interfaces

Different JavaScript names

- 1 - Mocha
- 2 - Live Script
- 3 - JavaScript
- 4 - ECMAScript

How JavaScript executes

Because JavaScript is a scripting language so, JavaScript cannot run on its own therefore a browser is responsible for running JavaScript code.

When a user requests an HTML page with JavaScript linked in it, the script is sent to the browser and it is up to the browser to execute it.



Scripting Language

- A script or scripting language is a computer language with a series of commands within a file that is capable of being executed without being compiled.
- Good examples of server-side scripting languages include Perl, PHP, and Python. The best example of a client-side scripting language is JavaScript.
- Among the more common uses of scripting languages is the production of Common Gateway Interface, or CGI, scripts. CGI scripts let Web browsers make use of programs running on a Web server. Common scripting languages include Perl, JavaScript, VBScript, and REXX.

Features of scripting language

The different features of Scripting languages are as below :-

- Scripting languages run faster and easy to use.
- It is easy to learn and develop the code if basic programming knowledge is there.
- The code can be easily developed as no separate IDEs are not required.
- Minimal development environment set up is needed.
- Many editors available to edit the code.
- Good for automation tasks.
- To enhance the automated tasks in server-side scripts.
- Easier to automate the tasks in web servers to monitor the server functionalities.
- Objects do exist similar to object-oriented languages like JavaScript.
- Contains less code compared to traditional programming languages.
- To extract the information from the large datasets.
- Memory allocation is not required for the scripting languages to be run.

Object Based Vs Object-Oriented Programming Language

a) Object Based Programming Language

- Object based languages supports the usage of objects.
- They do not support abstraction or, polymorphism or, both.
- Object based languages does not support built-in objects.
- Prototype-based systems.
- Object-based languages that do not support inheritance or subtyping are usually not considered to be true object-oriented languages.
- Till ES5 JavaScript is Object based language.

b) Object-Oriented Programming Language

- Object Oriented Languages supports all the features of OOPS including inheritance and polymorphism.
- They support built-in objects.
- C#, Java, .Net are the examples of object-oriented languages.
- Class based systems.
- After ES5 JavaScript becomes an Object-Oriented (Scripting) Language.

LEARN 2 EARN
LABS

Interpreted Languages Vs Compiled Languages

Interpreted Languages

- These Languages execute the code line by line or we can say that interpreted languages takes single instructions as input.
- Every line is read, analyzed, and executed. Having to reprocess a line every time in a loop is what makes interpreted languages so slow. This overhead means that interpreted code runs between 5 - 10 times slower than compiled code.
- Example include JavaScript, PHP etc.

Compiled Languages

- Compiled languages are converted directly into machine code that the processor can execute.
- Compiled languages are converted directly into machine code that the processor can execute. As a result, they tend to be faster and more efficient to execute than interpreted languages.
- Example Include C, C++ etc.

Dynamically Vs Statically Typed Languages

Dynamically Typed Languages

- A language is dynamically typed if the type of a variable is interpreted at runtime.
- Example includes JavaScript, Python etc.
- We use variables directly.

Statically Typed Languages

- A language is statically typed if the type of a variable is known at compile time
- Example includes C, C++ , Java etc.
- We don't use variables directly.

Who uses JavaScript

Major companies uses JavaScript that includes :-

- Microsoft
- Paypal
- Netflix
- Uber
- Facebook
- Google
- Ebay
- Walmart
- LinkedIn

JavaScript Adoption

Major companies adopted JavaScript with their own names includes :-

- Microsoft - JScript
- Google - JavaScript
- Mozilla - ECMAScript

Why do people disable JavaScript

People intentionally turn off JavaScript in their web browsers. This could be for a number of reasons including to allow web pages to load faster, or maybe to prevent scripts that may be harmful to their computer, such as downloading things they didn't want downloaded (unlikely but possible).

What does a browser have

In order to execute JavaScript code we need a browser, so a browser have three main programs :-

- a) DOM interpreter -- Takes HTML Code and displays it into the browser.
- b) CSS interpreter -- Takes CSS code and styles the page.
- c) JavaScript Engine(JS Interpreter + JIT Compiler) -- To Execute JavaScript Code.

The job of JavaScript engine is to take JavaScript files that would be downloaded from the web server and interpret them into byte code that can be on the user's computer.

Or,

- The job of JavaScript engine is to take JavaScript syntax in human readable form and converts it automatically into byte code / machine code so that our computer can understand our instructions.
- The above three process of browser are called JIT compiler.
- The DOM interpreter + CSS interpreter combines to make a Rendering Engine / Layout Engine.

JavaScript VSCode Extensions

- 1 - Debugger for Chrome – Microsoft (deprecated)
- 2 - JavaScript(ES6) code snippets -- Charalampos Karypidis
- 3 - Babel JavaScript - Michael McDermott
- 4 - Prettier – Prettier

JavaScript File Linking

There are typically three ways to add JavaScript to a web page :-

1) Internal Linking

- In this case we embed the JavaScript code between a pair of <script> and </script> tag.
- These script tags contain JavaScript placed either just above the closing </head> tag or, either place just above the closing </body> tag.
- But ideally, scripts should be placed at the end of the body section, just before the closing </body> tag, it will make your web pages load faster, since it prevents obstruction of initial page rendering.
- Each <script> tag blocks the page rendering process until it has fully downloaded and executed the JavaScript code, so placing them in the head section (i.e. <head> element) of the document without any valid reason will significantly impact your website performance.

Example -- Just Above the closing </head> tag

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>Document</title>
  <script>
    alert("Hello JavaScript You're awesome");
  </script>

</head>
<body>

  </body>
</html>
```

Example -- Just Above the closing `</body>` tag


```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>Document</title>

</head>
<body>

  <script>

    alert("Hello JavaScript You're awesome");

  </script>
</body>
</html>
```

The logo for 'Learn 2 Earn Labs' is centered on the page. It features the word 'LEARN' in blue, a large red '2' inside a hexagon, and the word 'EARN' in blue. Below this, the word 'LABS' is written in a large, grey, sans-serif font. A horizontal line with small circles at each end passes behind the '2' hexagon.

2) External Linking

- In this case we create an external JavaScript file with the ".js" extension and then load it within the page through the "src" attribute of the <script> tag.
- While linking external JavaScript files we must have to use "src" attribute this attribute defines the location/source of the JavaScript file.
- The <script> tag indicates the browser that the contained statements are to be interpreted as executable script and not HTML.

Example

```
<!DOCTYPE HTML>
```

```
<html>
```

```
<body>
```

```
<p>Before the script...</p>
```

```
<script src="file.js">
```

```
  alert(1); // the content is ignored, because src is set
```

```
</script>
```

```
<p>...After the script.</p>
```

```
</body>
```

```
</html>
```

Example -- Linking Multiple Files

We can place any number of <script> element in a single document. However, they are processed in the order in which they appear in the document, from top to bottom.

```
<!DOCTYPE HTML>
<html>

<body>

  <p>Before the script...</p>

  <script src="file_1.js"> </script>
  <script src="file_2.js"> </script>
  <script src="file_3.js"> </script>

  <p>...After the script.</p>
</body>
</html>
```



3) Inline Scripting

- We can also place the JavaScript code directly inside an HTML tag using the special tag attributes / event handlers such as onclick, onmouseover, onkeypress, onload, etc.
- But, we avoid placing large amount of JavaScript code inline as it clutters up your HTML with JavaScript and makes your JavaScript code difficult to maintain.

Example

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>Document</title>
</head>
<body>
  <button onclick="alert('You Just Clicked Button')">Button</button>
</body>
</html>
```

LEARN 2 EARN
LABS

Statements In JavaScript

- Instructions that a JavaScript interpreter can execute are called statements.
- JavaScript does not require to end a statement with a semicolon (;), but it is recommended to always use the semicolon to end a statement cause the semicolon make our code more readable and helps us to avoid many issues.
- All the JavaScript code that you will write will, for the most part, be comprised of many separate statements.
- A statement can set a variable equal to a value.
- A statement can also be a function call, i.e. document.write().

Example -- Single Line Statements

```
var a = 20;
```

Example -- Multi-line Statements

```
x = 10;
```

```
y = 20
```

```
g = 30
```

```
var z =
```

```
x + y
```

```
+ g
```

```
console.log(z)
```

Comments

- JavaScript Comments is a programmer's tool. Comments are used to explain the code, and the interpreter ignores them.
- Like other languages JavaScript also have: -

a) Single Line Comments

JavaScript use double forward slashes(//) in order to declare single line comments.

Example

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>Document</title>
</head>
<body>

  <script>

    // Single Line Comment In JavaScript

  </script>
</body>
</html>
```

b) Multiple Line Comments


For multiple line comments a block comment starts with a forward slash and asterisk (/*) and ends with the opposite (*/).

Example

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>Document</title>
</head>
<body>
  <script>

    /* Multiple Line
    Comment
    In
    JavaScript */

  </script>
</body>
</html>
```

The logo for 'Learn 2 Earn Labs' is positioned in the background. It features the word 'LEARN' in blue, a large red '2' inside a hexagon, the word 'EARN' in blue, and the word 'LABS' in grey below them.

Whitespace Sensitivity

- Whitespace are necessary in separating the keywords from the variables or other keywords.
- JavaScript ignores multiple spaces. We can add whitespaces to your script to make it more readable.
- The most common whitespace include a space, tab & newline.
- Whitespace enhances the readability of the JavaScript code.

Example

```
console.log("Hello    World")
```



Basic Output Functions

For getting output from JavaScript file, we have to use some predefined JavaScript functions & properties.

Following are the predefined JavaScript functions & properties :-

- console.log()
- document.write()
- document.body
- innerHTML Vs innerText Vs textContent
- alert()
- window.alert()

Example 1

```
console.log("Hello World")
```

Example 2

```
document.write("Hello World")
```

Example 3

```
document.body.innerHTML = "Hello World"
```

Example 4

- a) document.body.innerHTML = "Hello InnerHTML"
- b) document.body.innerText = "Hello InnerText"
- c) document.body.textContent = "Hello TextContent"

Example 5

```
alert("Hello Alert Function")
```

Example 6

```
window.alert("Hello Window Alert")
```

Variables in JavaScript

- Variables are fundamental to all programming languages.
- Variables are used to process the stored data, like string of text, numbers, etc. The data or value stored using the variables can be set, updated, and retrieved whenever needed.
- A JavaScript variable is simply a name of storage location.
- Basically, Variables are used to process the stored values (name = "John") or expressions (sum = x + y).
- Variables in JavaScript are containers which hold reusable data. It is the basic unit of storage in a program.
- Basically, variables are symbolic names for values.
- JavaScript variables are dynamically-typed.
- JavaScript variables are loosely-typed which means it does not require a data type to be declared. It can process the stored value of any data type throughout its life time.

Procedure to create variables

We can create a variable with the "var" keyword and assign a value to this variable.

Syntax

```
var variable_name = "value";
```

Or,

```
var <variable-name>; // declaration
```

Or,

```
var <variable-name> = <value>; // declaration and initialization
```

Variable declarations

a) Implicit Type Declarations --

- Implicitly typed variables are those variables which are declared without specifying the type explicitly.
- In Implicitly typed variable, the type of the variable is automatically deduced at compile time by the compiler from the value used to initialize the variable.

b) Explicit Type Declarations --

- Those languages where the variable value explicitly defines its type are known as Explicit Type Declarations.

NOTE :- JavaScript doesn't use explicit variable declaration but it can change the data type of variable explicitly.

Implicit & Explicit Variable Declaration --

int abc = 23 # Explicit Variable Declaration -- Java,C
abc = 23 # Implicit Variable Declaration -- JS,Python

Points to remember about JavaScript variables

- The value stored using a variable can be changed during program execution.
- A variable is only a name given to a memory location, all the operations done on the variable effects that memory location.
- In JavaScript, all the variables must be declared before they can be used.
- In JavaScript, variables can also be declared without having any initial values assigned to them. This is useful for variables which are supposed to hold values like user inputs.

Example

```
var abc // Declaring a variable  
var bac = 20 // Initializing(or, assigning) a variable
```

Example

```
var abc = bac = ghi = "Hello"  
console.log(abc)  
console.log(bac)  
console.log(ghi)
```

Example -- Declaring Multiple Variables

```
var abc,bac,ghi = "Hello"  
console.log(abc) //undefined  
console.log(bac) //undefined  
console.log(ghi) //Hello
```

Example

```
var x = 10;  
var y = 20;  
var z=x+y;  
document.write(z);
```

Example

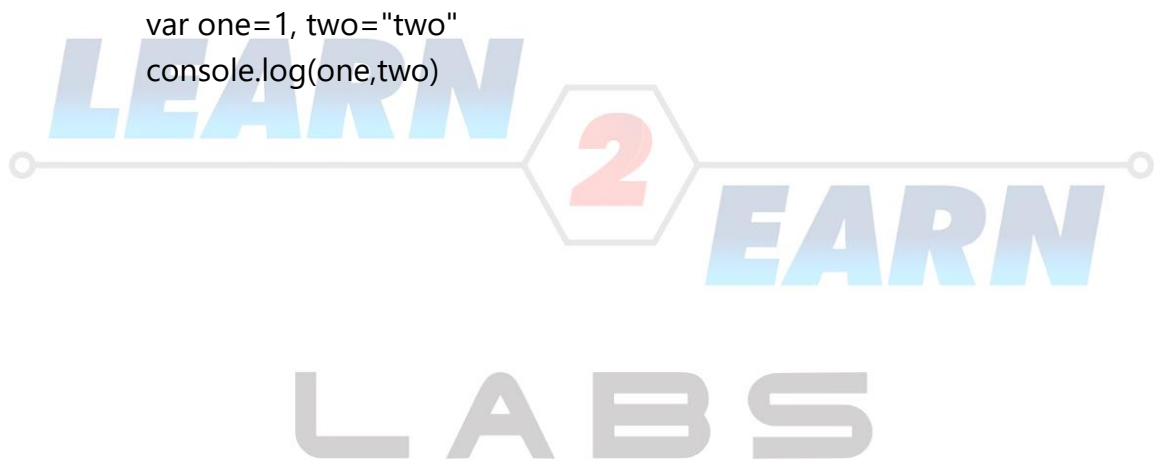
```
var x = 5 + 10 + 1;  
console.log(x); // 16
```

Example

```
var a = 10,b = 20,c = 30  
console.log(a+b+c) //60
```

Example

```
var one=1, two="two"  
console.log(one,two)
```



Identifiers

- All JavaScript variables must be identified with unique names called identifiers.
- Identifiers are just the names and can be used with variables, Keywords, Objects & Functions.
- An identifier is a string of alphanumeric characters that begins with an alphabetic character or numeric character or an underscore character that are used to represent various programming elements such as variables, functions, arrays, Objects etc.

Rules for defining an identifier

These are the following rules for naming a JavaScript identifier :-

- An identifier must start with a letter, underscore (_), or dollar sign (\$).
- An identifier cannot start with a number.
- An identifier can only contain alpha-numeric characters (A-z, 0-9) and underscores.
- An identifier cannot contain spaces.
- An identifier cannot be a JavaScript keyword or a JavaScript reserved word.
- Non-English words are also allowed as an identifier.

LABS

Case of joining identifiers

1 - CamelCase

- CamelCase is a naming convention in which a name is formed of multiple words that are joined together as a single word with the first letter of each of the multiple words capitalized so that each word that makes up the name can easily be read.
- With camelCase, the first letter of the first word in the identifier is lower case, while the first letter of every subsequent word is uppercase.

Example

```
var camelCase = "Hello Variable"  
console.log(camelCase)
```

2 - PascalCase / Bumpy / UpperCamelCase

- With PascalCase, the first letter of every word in the identifier is capital.

Example

```
var PascalCase = "Hello Variable"  
console.log(PascalCase)
```

3 - Snake Case / lowerCamelCase

- Snake Case involves writing phrases or compound words in which the words are separated using an underscore symbol (_) instead of a space.
- The first letter of the words in snake_Case is typically written in lowercase.

Example

```
var snake_Case = "Hello Variable"  
console.log(snake_Case)
```


4 - Macro Case

- Macro Case includes of writing compound words in which the letters are capitalized and are separated using an underscore(_) symbol instead of space.

Example

```
var MACRO_CASE = "Hello Variable"  
console.log(MACRO_CASE)
```

Undefined Vs Undeclared Variables

- An undefined variable is a variable that has been declared. Because we have not assigned it a value, the variable used the undefined as its initial value.
- In contrast, an undeclared variable is the one that has not been declared in the accessible scope.

Example

```
var abc;  
console.log(abc) // undefined variable  
console.log(bac) // undeclared variable
```

Variable Hoisting

Variable hoisting is a feature by which we can access the variable before declaring the variable.

Example

```
console.log(name)  
name = "Mohit"
```

JavaScript Variables Examples

Example 1

```
var abc = 23;  
console.log(abc);
```

Example 2

```
var x = 5;  
var y = 6;  
var z = x + y;  
  
console.log(z);
```

Example 3

```
abc = 23;  
console.log(abc);
```

Example 4

```
var abc, bac, ghi = 23;  
console.log(abc); // undefined  
console.log(bac); // undefined  
console.log(ghi); // 23
```

Example 5

```
var abc = true, bac = "HELlo" ,ghi = 23;  
console.log(abc); // true  
console.log(bac); // HELlo  
console.log(ghi); // 23
```

Example 6

```
var abc = null;  
var bac = undefined;  
  
console.log(abc); // null  
console.log(bac); // undefined
```

Example 7

```
var abc;  
var bac;  
  
console.log(abc); // undefined  
console.log(bac); // undefined
```

