JavaScript

JavaScript is a client-side scripting language and one of the most efficient, commonly used scripting languages. The term .**client-side scripting language means** that it runs at the client-side(or on the client machine) inside the web-browsers, but one important thing to remember is that client's web-browser also needs to support the JavaScript or it must be JavaScript enabled. Nowadays, most of the modern web browsers support JavaScript and have their JavaScript engines. For example, Google Chrome has its own JavaScript engine called V8.

Some other web-browsers with their JavaScript engines

	Web Browser	JavaScript engines
1.	Edge	Chakra
2.	Safari	JavaScript Core
3.	Firefox	Spidermonkey

It totally depends on the web-developers how they use JavaScript and for what, because it can be used for several things in web development. One of the most common uses of JavaScript is to validate data given by the user in the form fields.

Editors For JavaScript

Before knowing, how to start with JavaScript, we have to know what are the available editors for JavaScript. Some of the recommend editors for JavaScript are:

- Notepad
- Notepad++
- Sublime Text
- Visual Studio Code
- Atom
- BBEdit
- Komodo Edit

- Emacs etc.

With the help of following example, we can understand how JavaScript works:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>JavaScript Working Process</title>
</head>
<body>
<h1>This how javascript works</h1>
<script>
   alert("Hi,Their");
   console.log("JavaScript");
  </script>
</body>
</html>
```

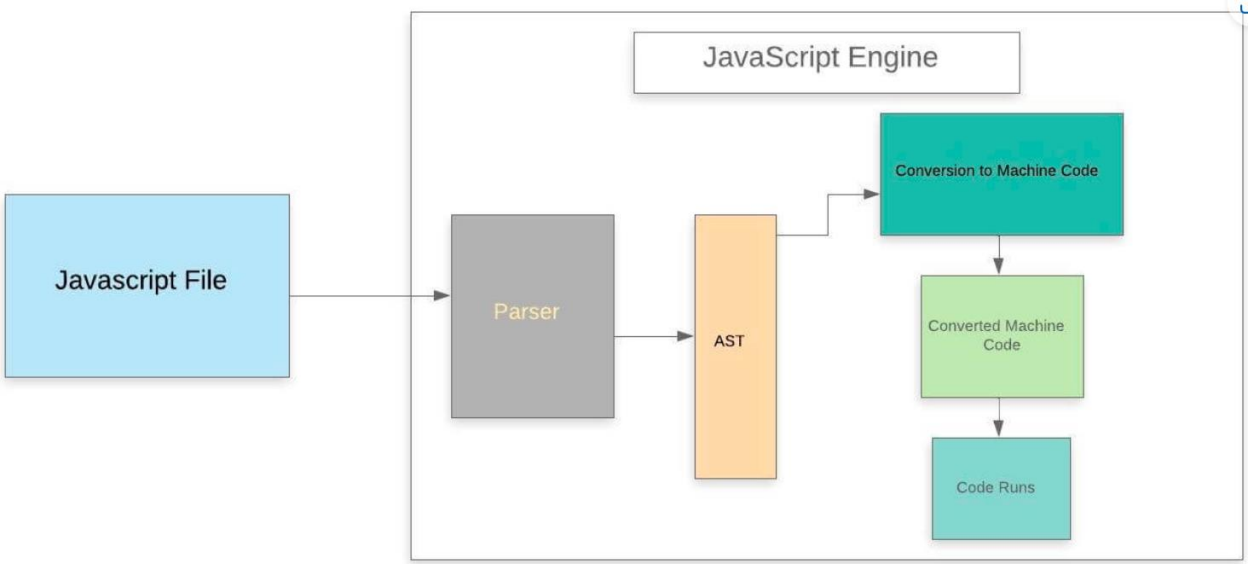
In the above program, we just displayed a alert message to the user by using the "alert()" method that is a pre-defined method of JavaScript. We also used the "console.log();" method and passed "JavaScript" as the String and in the inspect mode we can see "JavaScript" in the console as shown in the below output.

"How the browser understands the JavaScript code and runs it."

Mostly every web browsers nowadays have their own JavaScript engines, as we have discussed above. So, it is the JavaScript engine that understands the code and runs it.

Now let's see how the JavaScript engine handles and runs .js code.

In this case, we have used a chrome browser to run our program that has the "V8" JavaScript engine, which is also used for creating the Node.js. As we already know, JavaScript is an interpreted language that means it gets executed in line by line manner (or which means the JavaScript engine converts the Js code line by line and runs in the same manner instead of converting the whole program once).
We can understand how a typical JavaScript engine works with help of a diagram:



Whenever we run a JavaScript program inside a web browser, JavaScript code is received by the browser's engine and the engine runs the source code to obtain the output.
In a standard JavaScript engine, the source code goes through several steps and gets executed as you can see in the above given diagram.
Let us understand each of these steps in more detail.

Step 1: Parser

This is the first stage of the engine, every time we run a JavaScript program, our code is first received by the "parser" inside the JS engine. The parser's job is to check the JavaScript code for syntactic errors in line by line manner because JavaScript is an interpretive scripting language, so whenever an error is detected by the parser, it throws a kind of error and stops execution of the code.

In short, we can say that it parses JavaScript code.

Step 2: AST

Once the parser checks all JavaScript codes and gets satisfied that there are no mistakes/errors in the code, it creates the data structure called AST (it stands for Abstract Syntax Tree).

We can easily understand what is AST with help of following example.

Example

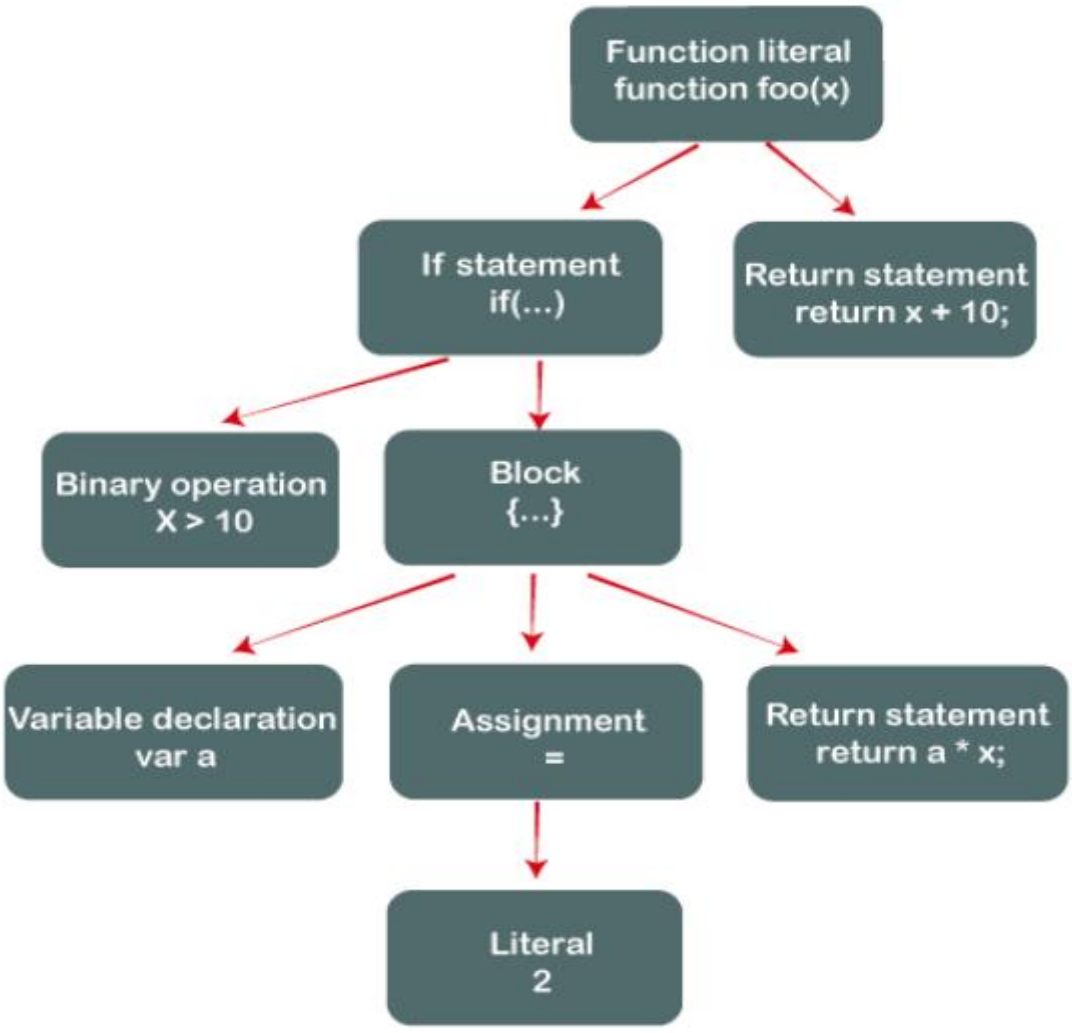
Let's suppose we have a JavaScript program as given below:

Program

```
function foo(x) {
if (x > 15) {
var a = 2;
return a * x;
    }

return x + 10;
}
```

Once the parser checks the above JavaScript code, it will create a data structure called **AST**. The created AST (Abstract Syntax Tree) looks like the given image.



Step 3: Conversion to Machine code

Once the Abstract Syntax Tree is created by the parser, the JavaScript engine converts the JavaScript code into the machine code (or in the language that machine can understand).

Step 4: Machine code

When the program written in the JavaScript gets converted in the machine language (or in byte code), the converted code is sent to the system for execution, and finally, that byte code run by the system/engine .

Document Object Model

The **document object** represents the whole html document.

When html document is loaded in the browser, it becomes a document object. It is the **root element** that represents the html document. It has properties and methods. By the help of document object, we can add dynamic content to our web page.

As mentioned earlier, it is the object of window. So

window.document

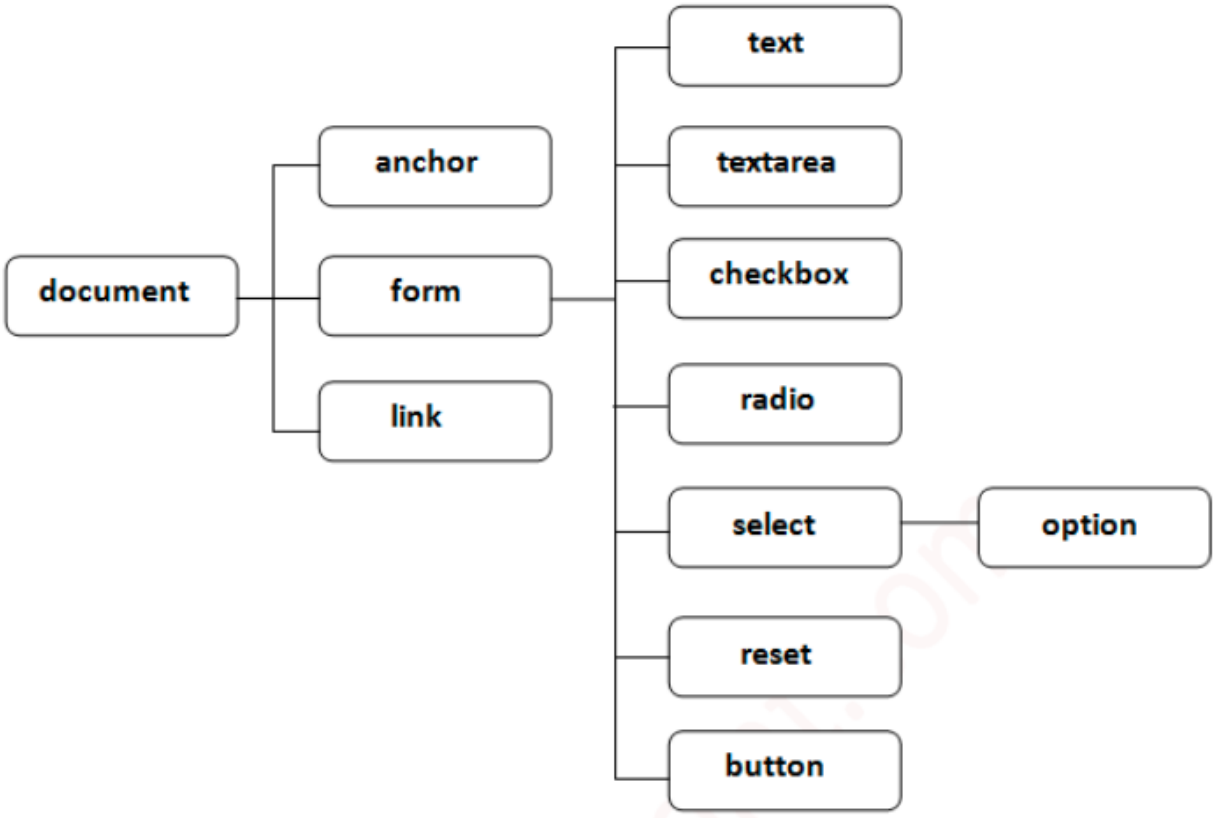
Is same as

document

According to W3C - *"The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document."*

Properties of document object

Let's see the properties of document object that can be accessed and modified by the document object.



Methods of document object

We can access and change the contents of document by its methods.

The important methods of document object are as follows:

Method	Description
write("string")	writes the given string on the doucment.
writeln("string")	writes the given string on the doucment with newline character at the end.
getElementById()	returns the element having the given id value.
getElementsByName()	returns all the elements having the given name value.
getElementsByTagName()	returns all the elements having the given tag name.
getElementsByClassName()	returns all the elements having the given class name.

Accessing field value by document object

In this example, we are going to get the value of input text by user. Here, we are using **document.form1.name.value** to get the value of name field.

Here, **document** is the root element that represents the html document.

form1 is the name of the form.

name is the attribute name of the input text.

value is the property, that returns the value of the input text.

Let's see the simple example of document object that prints name with welcome message.

```
<script type="text/javascript">
function printvalue(){
var name=document.form1.name.value;
alert("Welcome: "+name);
}
</script>

<form name="form1">
Enter Name:<input type="text" name="name"/>
<input type="button" onclick="printvalue()" value="print name"/>
</form>
```

Output of the above example