Working with UI means working with below technologies.

1. HTML. 🡪 HTML describes the structure of a Web page, HTML consists of a series of elements, HTML elements tell the browser how to display the content
2. CSS. 🡪 or styling (color), size representations. it tells how HTML elements are to be displayed on screen.
3. **JavaScript**. 🡪 validations, Event 🡪 Action. **It performs actions DYNAMICALLY.**

**Event means? 🡪 onclick, ondbclik, onmouseover, etc..**

**Actions means? 🡪 to perform validations on user given input data, changing element colors dynamically on any event, changing elements size and width dynamically on any event.**

**JavaScript Actions:**

1. Predefined actions (functions). Eg: alert(), console.log()…etc.
2. Custom actions (functions). Eg: myFunction()…..etc.

In above code we can see we have written JavaScript code as INTERNL JS, means we have written in **<script>** tag.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Inline** | **Internal** | **External** |
| **CSS** | <tagName **style=”p:v;p:v**”  Here **p – property** of css  **V – values** of that css property.  Eg:  <p **style=”color:red;”>**  </p> | <style>  P{ **// selector**  Color:red;  }  </style> | Create **test.css**  In this CSS file Write direct code.  Import this file in HTML:  **<link href="test.css" rel="stylesheet">** |
| **JavaScript** | eventName=function  eg:  onclick=”myFunctionName()” | <script>  JS code..  Functions etc…  </script> | Create Test.js  In this JS file Write direct code.  Import this file in HTML:  **<script src="test.js"></script>** |

**What is Browser Inspect?** What options are there in it? Please explain all options.

Brower inspect is used for debugging the errors, for checking elements, for checking requests, for checking source paths.



We use above technologies for creating UI (User Interface) pages.

UI – User Interface – Which ever user experiences that page is called as UI page / Browser page.





**Validation?** – Checking the user entered details (user name, password, email, phone number etc..) at client side and at server side is called as Validation.



Server or Client always perform validation on user given input data.

Response is two types🡪 Valid(Success) response / Error response.

Browser**?** 🡪 A web browser, or simply a browser, is a software application used to access and view content on the World Wide Web. It acts as an interface between the user and the internet, allowing them to navigate and interact with websites.



Browser:

How many browsers are there in the market? 🡪 So many browsers are available in the market.

What are they? Tell me 4 browsers?

Different companies created different browsers. Based on **browsing speed, memory usage** we use different browser.



Which technologies can a browser understand? **HTML, CSS, Javascript**.

A browser can execute/run only **HTML, CSS, Javascript** files.



Then what about other UI technologies? How they run in browser? ->

For example, Angular or ReactJs how they run in browser? 🡪 First, we convert/compile/transpile the angular/reactjs into JavaScript code then we run in browser.





What is meant by Extension 🡪 It is File format.

How to create these files? 🡺 Left/Right Arrow mark.

**<anyName>.extension**

<anyName>.html, <anyName>.css, <anyName>.js

Extension? -> Which tells which type/format of the file we are going to create.

Eg: for TEXT file we use extension as **.txt**

for WORD file we use extension as **.docx**

for PDF file we use extension as **.pdf**

for JAVA file we use extension as **.java**

for IMAGES file we use extension as **.jpeg, .png, .gif….**

for HTML file we use extension as **.html / .htm**

for CSS file we use extension as **.css**

for Javascript file we use extension as **.js**

**etc…………………………..**

**How many ways we can write HTML code, CSS code, JS code?**



**Abbreviations:**

HTML 🡪 Hyper Text Markup Language.

CSS 🡪 Cascading Style Sheet.

**“id”(# - in CSS) attribute and “class”(. DOT – in CSS) attribute:**

Attributes always give additional information about element.

**we use “id” to identify element uniquely.**

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>    <title>Javascript</title>  </head>  <style>  </style>  <body>    <form action="**#**">      Enter Name: <input type="text**" id="uName"** class="abcClass" style="color: red;" value="Kamal"/><br>      Enter password: <input type="password" **id="pwd"**><br>      Select country: <select **id="country**">        <option>India</option>        <option>US</option>        <option>UK</option>      </select>      <br>      <input type="submit" value="Login" **onclick="test1();"**><br>    </form>  **<script>**  **function test1() {**  **console.log(document.getElementById('uName'));**  **}**  **</script>**  </body>  </html> |

Here **document.getElementById('uName')** is a Javascript selector.

Selector 🡪 Find and Apply. (CSS)

**JS selector 🡪 Find and Fetch (JacaScript)**

|  |
| --- |
| function validateInpuData() {      document.getElementById('uName'**).value = "AAAAAAAAA";**      document.getElementById('pwd').value = "BBBBBBB";      var v = document.getElementById('country').value;      document.getElementById("male**").checked** = true;      document.getElementById('uName'**).style = "background-color:red;";**      document.getElementById('pwd').style = "background-color:red;";    } |

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  </head>  <style>  </style>  <script>    function validateInpuData() {      document.getElementById('uName'**).disabled = false;**      document.getElementById('pwd').disabled = true;  **document**.getElementById('male').disabled = true;      document.getElementById('fMale').disabled = true;      document.getElementById('country').disabled = true;      document.getElementById('sbmt').disabled = true;      document.getElementById('fMale').**checked = true;**      document.getElementById('eng').checked = true;      document.getElementById('tel').checked = true;      document.getElementById('tml').checked = true;      document.getElementById('eng').disabled = true;      document.getElementById('tel').disabled = true;      document.getElementById('tml').disabled = true;    }  </script>  <body>    <form action="#">      Enter user Name: <input type="text" id="uName" style="color: red;" value="Kamal" disabled="true"/> <br>      password : <input type="password" id="pwd" value="abc123" /> <br>      Gender : Male : <input type="radio" name="gender" id="male" />      Female : <input type="radio" name="gender" id="fMale" /><br>      Select Country:      <select id="country">        <option>Select</option>        <option>US</option>        <option selected="true">India</option>        <option>Aus</option>      </select>      <br>      Select languages:      English: <input type="checkbox" id="eng" />      Telugu: <input type="checkbox" id="tel" />      Tamil: <input type="checkbox" id="tml" />      <br>      <input type="submit" **onclick="validateInpuData()"** value="Submit" id="sbmt" /> <br>    </form>  </body>  </html> |

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  </head>  <body>    <form action="#">      Enter Name : <input type="text" id="userName" /><br>      Enter Password: <input type="password" id="pwd" /><br>      Select coubtry: <br>      <select id="country">        <option>Select</option>        <option>Aus</option>        <optio>India</option>          <option>UK</option>      </select><br>      Gender: <br>      Male : <input type="radio" id="male" name="g" />      Female : <input type="radio" id="female" name="g" />      <br>      Select Langualges:      English : <input type="checkbox" id="eng" />      Telugu: <input type="checkbox" id="tel" />      Tamil: <input type="checkbox" id="tml" />      <br>      <input type="submit" value="Submit..." onclick="testFunction()" />      <input type="submit" value="Enable Form" onclick="enalbeElements()" />    </form>    <script>      function testFunction() {        document.getElementById('userName').value = "Kamal";        document.getElementById('pwd').value = "ASASAS";        document.getElementById('male').checked = true;        document.getElementById('eng').checked = true;        document.getElementById('tel').checked = true;        document.getElementById('country').selectedIndex = 2;        var uName = document.getElementById("userName").value;        console.log(uName);        var passordV = document.getElementById('pwd').value;        console.log(passordV);        /\*          int uName = "Kamal";          System.out.println(uName);        \*/      }      function enalbeElements() {      }      </script>  </body>  </html> |

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  </head>  <body>    <form action="#">      Enter Name : <input type="text" id="userName" /><br>      Enter Password: <input type="password" id="pwd" /><br>      Select coubtry: <br>      <select id="country">        <option>Select</option>        <option>Aus</option>        <option>India</option>        <option>UK</option>      </select><br>      Gender: <br>      Male : <input type="radio" id="male" name="g" />      Female : <input type="radio" id="female" name="g" />      <br>      Select Langualges:      English : <input type="checkbox" id="eng" />      Telugu: <input type="checkbox" id="tel" />      Tamil: <input type="checkbox" id="tml" />      <br>      <input type="submit" value="Submit..." onclick="testFunction()" id="btn"/>      <input type="submit" value="Enable Form..." onclick="enableForm()" id="btn"/>      </form>    <script>      function testFunction() {        // document.getElementById('userName').value= "AAAAAA";        // document.getElementById('pwd').value= "BBBBB";        // document.getElementById("country").selectedIndex = 2;        // document.getElementById('male').checked = true;        // document.getElementById('tel').checked = true;        // document.getElementById('tml').checked = true;        document.getElementById('userName').disabled = true;        document.getElementById('pwd').disabled = true;        document.getElementById("country").disabled = true;        document.getElementById('male').disabled = true;        document.getElementById('female').disabled = true;        document.getElementById('tel').disabled = true;        document.getElementById('tml').disabled = true;        document.getElementById('eng').disabled = true;        document.getElementById('btn').disabled = true;        document.getElementById('pContent').style.display = 'block';        document.getElementById('divContent').style.display = 'none';      }        function enableForm(){        document.getElementById('userName').disabled = false;        document.getElementById('pwd').disabled = false;        document.getElementById("country").disabled = false;        document.getElementById('male').disabled = false;        document.getElementById('female').disabled = false;        document.getElementById('tel').disabled = false;        document.getElementById('tml').disabled = false;        document.getElementById('eng').disabled = false;        document.getElementById('btn').disabled = false;      }    </script>  </body>  </html> |

**What is the purpose of “innerHTML” attribute?**

To fetch the content of <p> element or fetch the content of <div> elements…etc related contents we will use “innerHTML” aatribute.

If you want to fetch the content we need to use “innerHTML” attribute.

|  |
| --- |
| var pContetnt = document.getElementById('pContent').**innerHTML**;  console.log(pContetnt); |

**How to hide <p> and <div> or any other elements on loading html page?**

We write “hidden=true” attribute in that cooresponding element.

|  |
| --- |
| <p id="pContent" **hidden="true"** style="background-color: greenyellow;">      Form Submitted successfull...!!!<br>      Please wait for 10 working Days to get mail.    </p> |

**How to show elements on click of submit button ? How to show elements on any javacript event?**

We use **“.style.display = 'block'**” on document object.

|  |
| --- |
| document.getElementById('pContent').**style.display = 'block';** |

Here block means not blocking the content. **Block** **means showing the content**.

**How to HIDE elements on click of submit button ? How to HIDE elements on any javacript event?**

We use **“.style.display = none**” on document object.

|  |
| --- |
| document.getElementById('divContent'**).style.display = 'none';** |

**How to fetch the “select” element data, selected innerHTML and selected value?**

|  |
| --- |
| function testFunction() {        var index = document.getElementById('country').**selectedIndex**;        console.log(index);          var optns = document.getElementById('country')**.options**;        console.log(**optns[index].innerHTML**);        console.log(**optns[index].value**);          if(optns[index].value == 111){                  alert('You selected Austriala');        }      } |

|  |
| --- |
| var index = document.getElementById("country").**selectedIndex**;        var options = document.getElementById('country').**options**;        var finalValue = **options[index].value**;  var htmlContent = **optns[index].innerHTML**; |

**List of attributes we used so far:**

|  |
| --- |
| **Syntax:**  **document.getElementById("ID\_NAME")🡪 it gives complete element and all attributes details.**  **document.getElementById("ID\_NAME").ANY\_ATTRIBUTE\_NAME;**  document.getElementById("country").**value; -🡪 to get the values from text fields/boxes.**  document.getElementById("country").**checked = true; -🡪 to get the values from radio buttons or multi checkboxes.**  document.getElementById("country").**disabled = true/false; 🡪 To disable any kind of element in html page.**  document.getElementById("country").**selectedIndex; 🡪 To get selected index value from drop down.**  document.getElementById("country").**options; 🡪 To get all the options available in dropdown.**  document.getElementById("country").**options[index].value; 🡪 To get value of selected dropdown.**  document.getElementById("country").**options[index].innerHTML; 🡪 To get inner text / label of selected dropdown / get the content of <p> or <div> tags.**  document.getElementById("country").**innerHTML; 🡪 to inner content of <p> tag, <div> tag or etc tags…**  document.getElementById("country").**style.display = ‘block’; 🡪 To show hidden element / content.**  document.getElementById("country").**style.display = ‘none; 🡪 To hide element / content.**  document.getElementById('uName'**).style = "background-color:red;";** |

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  </head>  <body>    <form action="#">      Enter Name : <input type="text" id="userName" /><br>      Enter Password: <input type="password" id="pwd" /><br>      Select coubtry: <br>      <select id="country" onchange="showElements();">        <option value="slect">Select</option>        <option value="111">AUS</option>        <option value="222">INDIA</option>        <option value="333">UK</option>      </select><br>      <div id="sec1">        Gender: <br>        Male : <input type="radio" id="male" name="g" />        Female : <input type="radio" id="female" name="g" />        <br>        Select Langualges:        English : <input type="checkbox" id="eng" />        Telugu: <input type="checkbox" id="tel" />        Tamil: <input type="checkbox" id="tml" />        <br>      </div>      <input type="submit"value="Submit..." onclick="submitFunction(); return false;" id="btn1" />      <input type="submit" disabled="true" value="Enable Form..." onclick="enableForm(); return false;" id="btn2" />    </form>    <p id="pContent" hidden="true" style="background-color: greenyellow;">      Form Submitted successfull...!!!<br>      Please wait for 10 working Days to get mail.    </p>    <div id="myDIV" style="color: green;">      Form submitted successfully..!!!<br>      100K+ creators trust Graphy to teach online      𝕏      Logiclabs Technologies 2025<br>      Privacy policy<br>      Terms of use<br>      Contact us<br>      Refund policy<br>    </div>    <script>      function submitFunction() {        document.getElementById('myDIV').style.display = "block";     }      function enableForm() {        document.getElementById('myDIV').style.display = "none";      }      function showElements() {        var index = document.getElementById("country").selectedIndex;        var options = document.getElementById('country').options;        var finalValue = options[index].value;        if (finalValue == 222) {          document.getElementById('sec1').style.display = 'block';          document.getElementById('btn1').disabled = false;          document.getElementById('btn2').disabled = false;        } else {          alert("You are unautorized to submit form!!");        }      }    </script>  </body>  </html> |

**->How to fetch the values from form & How to assign values to existing form?**

Write two functions, one is for fetching the all values, another is for assigning new values.

**“document” object:**

Why always we are writing **document**.getElementById()???

What is this “document”? why are we writing?

In the browser, what ever white/below yellow screen is visible is called as “document” object.



Above we can see yellow area we called as “document” object.

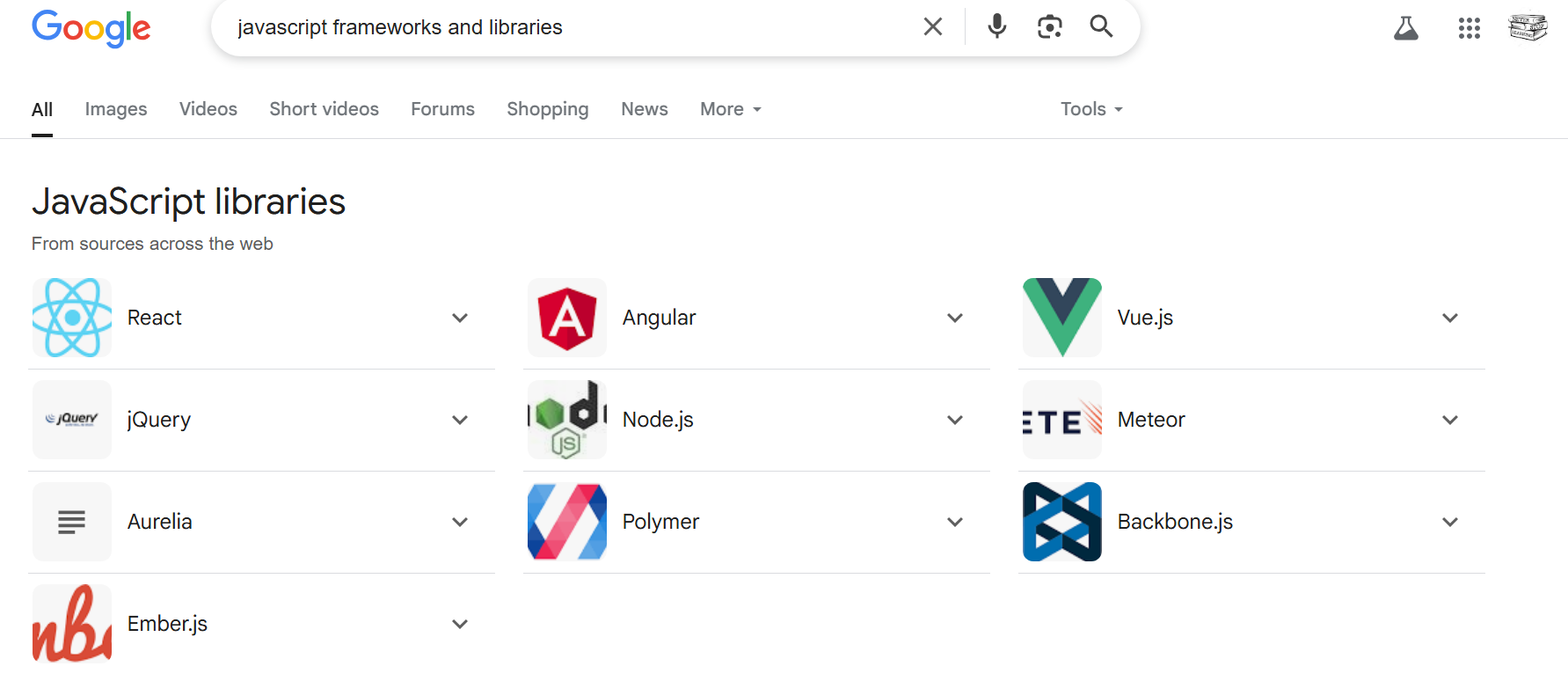
Complete browser is called as “window” object.





**Tutorial reading:**

<https://www.w3schools.com/js/default.asp>

****

If we see above listed frameworks, all those are developed based on the JavaScript only.

Those framework code finally converted or transpiled or compiled into JavaScript code only, why because, browser only understands HTML, CSS and JavaScript.

“=” is assignment operator. We use it for assigning values. Means for storing the values in variables.

“==” 🡪 comparison operator, used in mostly in “if” conditions.

This tutorial covers every version of JavaScript:

ES means – ECMA Script.

* The Original JavaScript ES1 ES2 ES3 (1997-1999)
* The First Main Revision **ES5** (**2009**)
* The Second Revision **ES6** (**2015**)
* The Yearly Additions (2016, 2017 ... 2021, 2022)

**What is JavaScript?**

JavaScript is the programming language of the web.

It can update and change both HTML and CSS.

It can calculate, manipulate and validate data.

|  |
| --- |
| function showContent() {         document.getElementById('myDIV').**innerHTML** = new Date();         document.getElementById('myDIV'**).style.color** = 'blue';         var nameValue = document.getElementById('nameId').value;         if(nameValue == ''){          alert('Please enter Name!');         }else{          console.log(nameValue);         }      } |

Why Study JavaScript?

JavaScript is one of the **3 languages** all web developers must learn:

   1. [**HTML**](https://www.w3schools.com/html/default.asp) to define the content of web pages,

   2. [**CSS**](https://www.w3schools.com/css/default.asp) to specify the layout of web pages, colors applying, size applying..

   3. **JavaScript** **to program the behaviour (action) of web pages. Onclick,onchange,onbdclick,onmousehover..etc.**

**To perform client side validations.**

JavaScript accepts both double and single quotes.

**JavaScript Can Change HTML Attribute Values.**

|  |
| --- |
| document.getElementById('mtInput').value = "Kamal"; |

|  |
| --- |
| **<img src="pic\_bulb\_off.gif" id="imageId" />**    <button type="button" onclick="bulbONFunction();" id="aaaa">Bulb ON</button><br>    <button type="button" onclick="bulbOFFFunction();">Bulb OFF</button><br>    <script>      function myFunction() {        document.getElementById("demo").innerHTML = "Hello JavaScript!";        document.getElementById('mtInput').value = "Kamal";      }      function bulbONFunction() {        document.getElementById('imageId'**).src = 'pic\_bulb\_on.gif'**      }      function bulbOFFFunction() {        document.getElementById('imageId'**).src = 'pic\_bulb\_off.gif'**      }    </script> |

JavaScript Can Change HTML Styles (CSS).

|  |
| --- |
| function myFunction() {         document.getElementById('demo').**style = "background-color:red;font-size:100px";**      } |

JavaScript Can Hide and shoe HTML Elements:

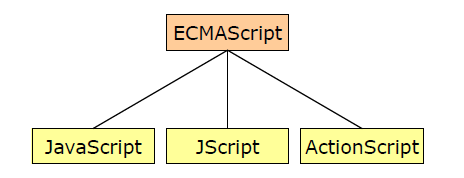
|  |
| --- |
| document.getElementById("demo").**style.display = "none";** 🡪 **to HIDE** the content / element.  document.getElementById("demo**").style.display = "block";🡪** **to SHOW** the content / element. |

JavaScript and [Java](https://www.w3schools.com/java/default.asp) are completely different languages, both in concept and design.

**JavaScript was invented by Brendan Eich in 1995, and became an ECMA standard in 1997**.

ECMA-262 is the official name of the standard. ECMAScript is the official name of the language.

ECMA Script is documentation.



JavaScript and others like JScript and ActionScript are all different implementations of ECMAScript.

Like JavaScript, we have Jscript and ActionScript other scripting languages, but JavaScript is very famous because of its syntax and performance. So developers more use JavaScript only.

Old JavaScript examples may use a type attribute: **<script type="text/javascript">.**  
But now in latest browsers the “**type**” attribute is not required. JavaScript is the default scripting language in HTML.

**JavaScript Functions and Events:**

A JavaScript function is a block of JavaScript code, that can be executed when "called" for.

For example, a function can be called when an **event** occurs, like when the user clicks a button.

You can place any number of **<script>** tags in an HTML document.

**<script>**s can be placed in the <body>, or in the <head> section of an HTML page, or in both.

**Placing <script>s at the bottom of the <body> element improves the display speed, because script interpretation slows down the display.**

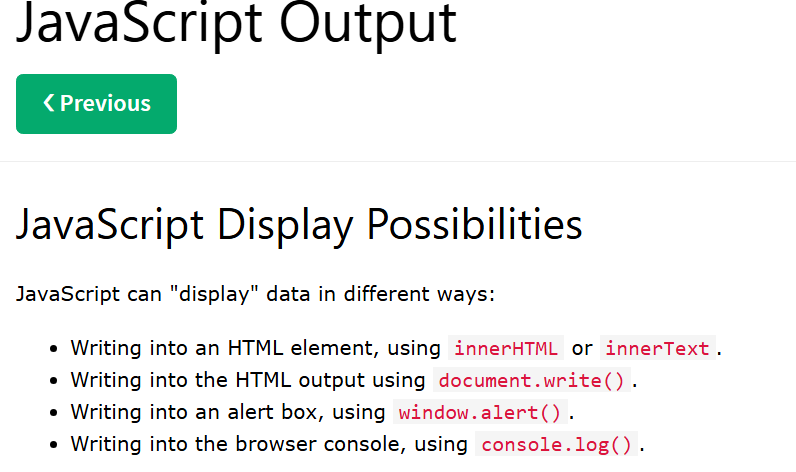
**Interpretation 🡪 line by line execution. In HTML, Browser engine will do the Interpretation.**

**In java JVM performs Interpretation. Means line by line execution.**

External script file should not contain <script> tags. Means in .js file we no need to write <script> tag.

**Which is the recommended place to keep <script> tag in html page?**

At the bottom of the <body> tag/ element.



**What is the difference between innerHTML and innerText attributes?**

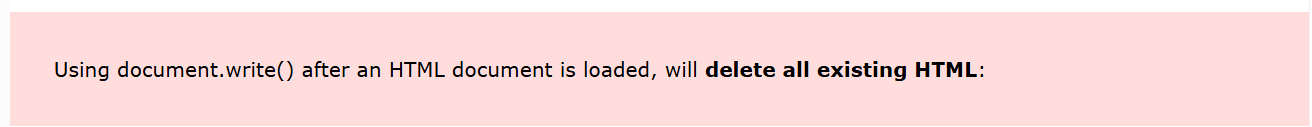
innerHTML = html contetnt + html tags.

innerTest = html content.

**Note:**

Use innerHTML when you want to change an HTML element.

Use innerText when you only want to change the plain text.



We should never use “document.write()” on events, why because it ls erasing existing all html content.

**Print:**

It is used to print the current page of html code.

|  |
| --- |
| <!DOCTYPE html> <html> <body>  <button onclick="**window.print()">**Print this page</button>  </body> </html> |

***what is the difference between instance variable and local variable java?***

Only the scope.

we declare Local variables in either methods or constructors or blocks, their scope is up to methods or constructors or blocks only.

We declare instance variable / class level variable at the top of the class. This variable scope is up to class. Means all methods, constructors and blocks can use/access this this variable.

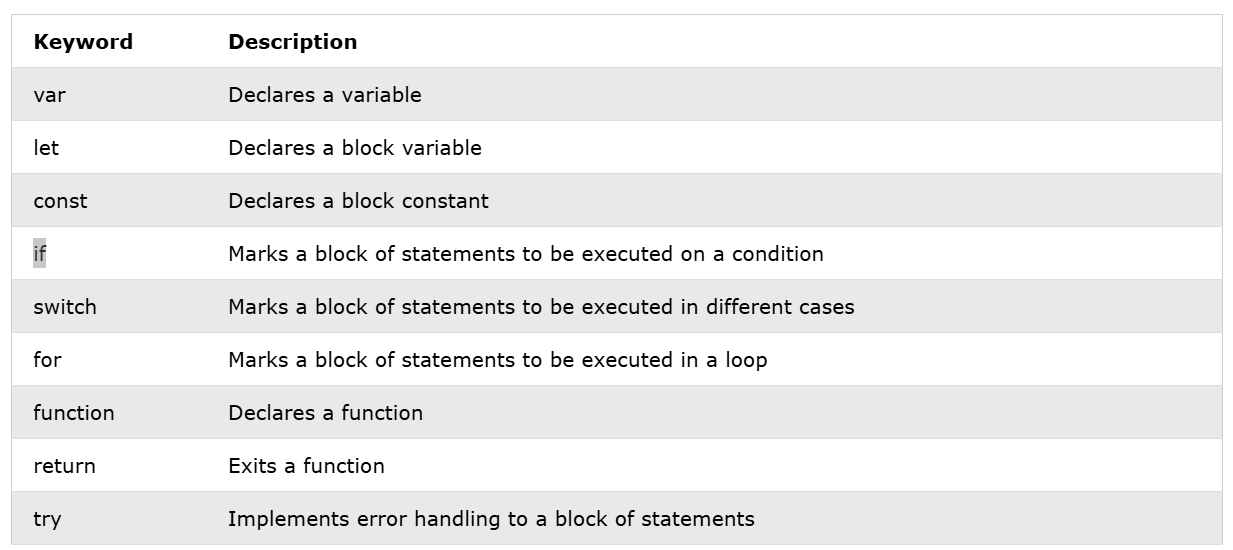
|  |
| --- |
| public class App {  **int a; *// variable declaration1***{  **int f = 10; *// declaration + initialization***}  public App(){  int x = 10; *// declaration + initialization  //f = 556; ==> Error* a = 54545;*// initialization* }   public void m1(String[] args) {  int b = 30; *// declaration + initialization* a = 34; *// declaration + initialization* }  } |

**What is meant by declaration and Initialization?**

JavaScript code works without termination / semi column, but it is always recommended to use termination for each line. Why because in PROD JS code will be minified or compressed into single line. If we don’t have termination then code will overlap and code will fail.

For example: below bootstrap plugin.

<https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js>



JavaScript keywords are reserved words. Reserved words cannot be used as names for variables.

**JavaScript Variables:**

In a programming language, **variables** are used to **store** data values.

**JavaScript uses the keywords var, let and const to declare variables.**

An **equal sign** (=) is used to **assign values** to variables.

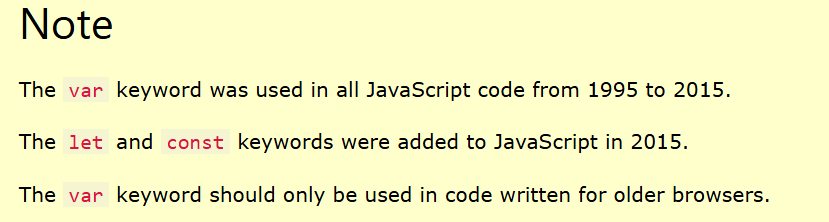
let name **=** “AZAD”;

**JavaScript Variables:**

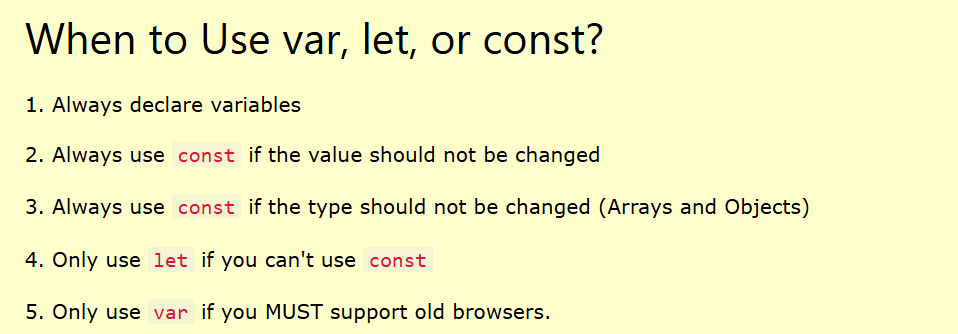
Variables are Containers for Storing Data

JavaScript Variables can be declared in 4 ways:

* Automatically (**no variable declaration**)
* Using **var**
* Using **let \*\***
* Using **const**



**As per the above conclusion at present in the latest browsers we always use “let” and “const” variable types only.**



mostly we use “let” variable type only.

|  |
| --- |
| */\* JAVA primitive types:  BSIL  float  double  boolean  char  +  classes + abstract classes +interfaces.  \*/* |

**Value = undefined:**

A variable declared without a value will have the value **undefined**. In JavaScript default value of any variable is “**undefined**”.

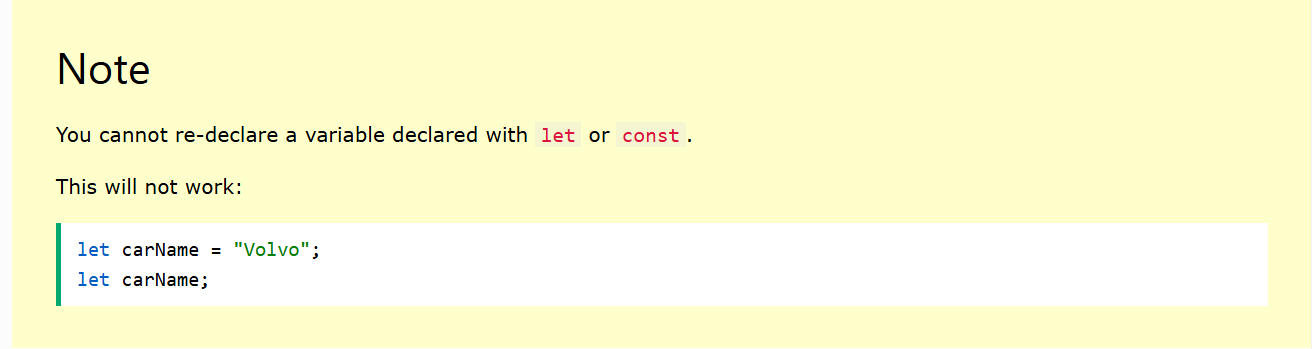
let name; 🡺 it contains ‘undefined’ value internally.

**Re-Declaring JavaScript Variables;**

If you re-declare a JavaScript variable declared with var, it will not lose its value.

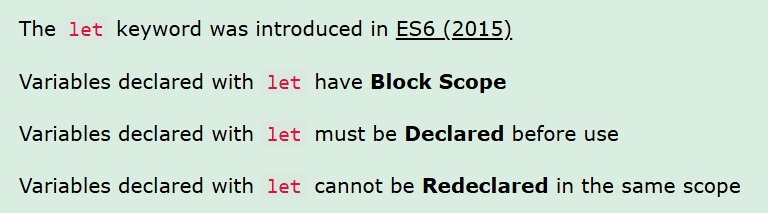
The variable carName will still have the value "Volvo" after the execution of these statements:

|  |
| --- |
| **var carName = "Volvo";**  **var carName;**  document.getElementById("demo").innerHTML = carName; |



With ”var” we have lot’s of issues, so it is always recommended to use “let” and “const” variable types only.

**JavaScript “Let”:**



What is meant by backtick? What is the use of it? IT will be used to concatenate with values without using “+” operator.

        let x = 10;

        console.log(**`**vaue is **: ${x}`**);

To represent dynamic value in backtick approach we need to use **${dynamicVariableName}**

What is the main difference between let and const keywords?

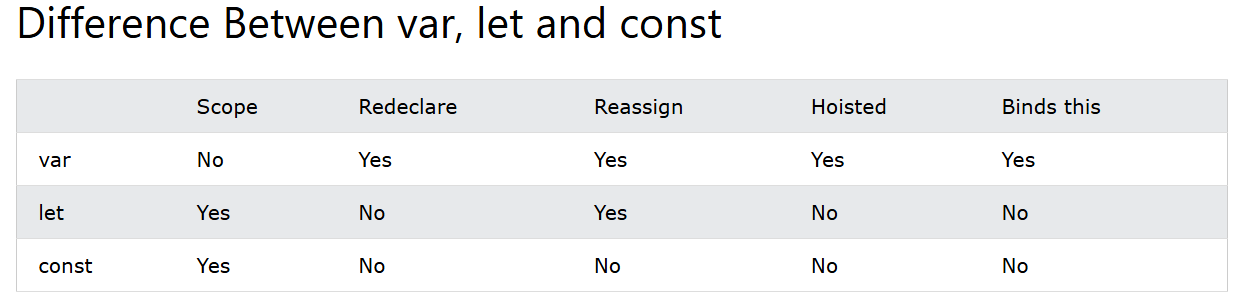
Values assigned to “**const**” variables are FIXED / **CONST**ANT. Their value we cannot changes in future.

|  |
| --- |
| **const** x = 10;          x = 30; |

In **Java** also we can declare **constants**. To do that we will use **“final” keyword**.

|  |
| --- |
| **final** String name ="AAAA"; // it is CONSTANT in java |

We cannot reassign value to constant variable again and again.





ES is an api document.

<https://ecma-international.org/publications-and-standards/standards/ecma-262/>

Java also have an API document.

[https://docs.oracle.com/en/java/javase/**21**/docs/api/java.base/java/lang/package-tree.html](https://docs.oracle.com/en/java/javase/21/docs/api/java.base/java/lang/package-tree.html)

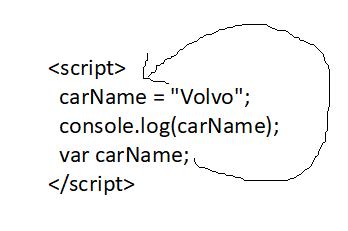
<https://docs.oracle.com/javase/7/docs/api/>

**Hoisting:**

Variables defined with var are **hoisted** to the top and can be initialized at any time.

Meaning: You can use the variable before it is declared:

|  |
| --- |
| <script>      carName = "Volvo";      console.log(carName);      var carName;    </script> |



Let and const are not hoisted.

 Using a let variable before it is declared will result in a ReferenceError:

**JavaScript Const:**

The const keyword was introduced in [ES6 (2015)](https://www.w3schools.com/js/js_es6.asp)

Variables defined with const cannot be **Redeclared**

Variables defined with const cannot be **Reassigned. It is not possible to assign value to variable multiple times.**

Variables defined with const have **Block Scope.**

Must be Assigned: JavaScript const variables must be assigned a value when they are declared:

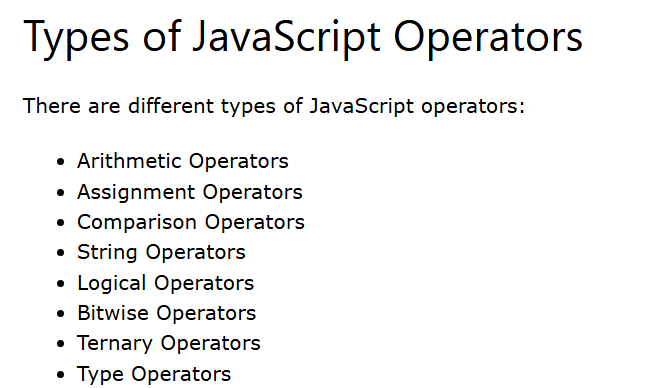
When to use JavaScript const?

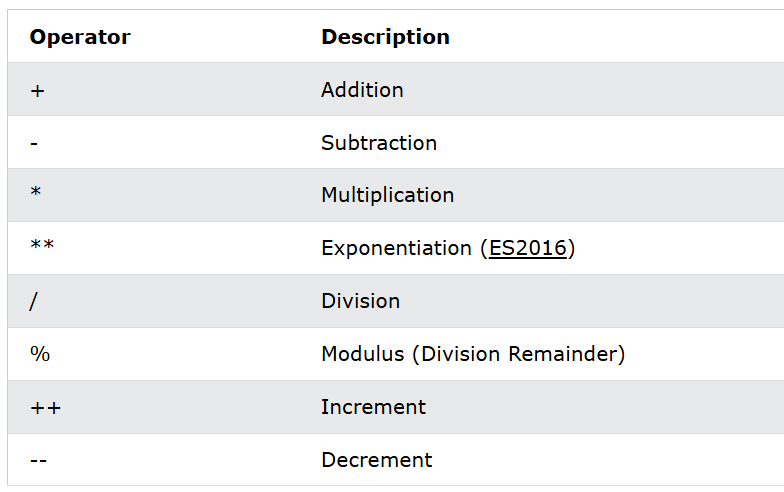
**Always declare a variable with const when you know that the value should not be changed.**

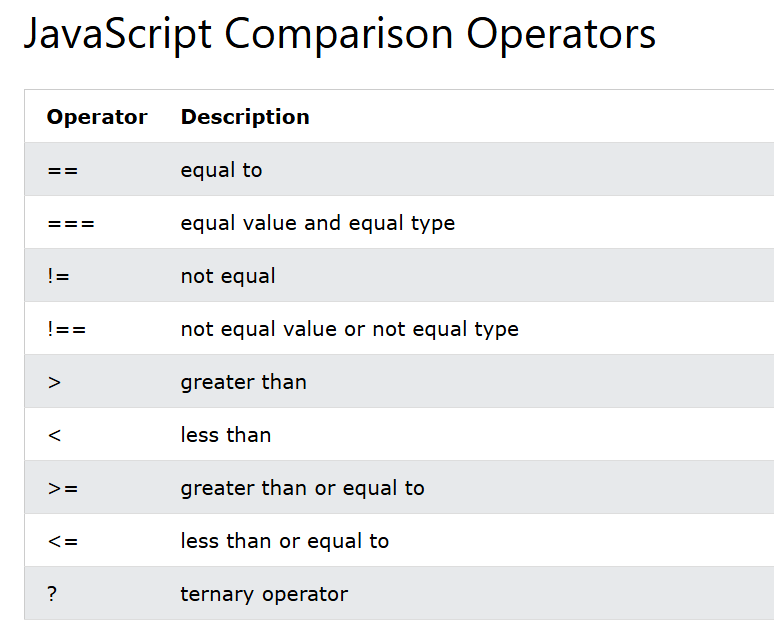
Read “const” after completing arrays and objects.

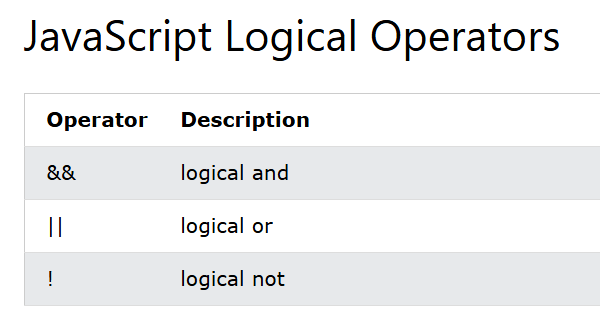
<https://www.w3schools.com/js/js_const.asp>

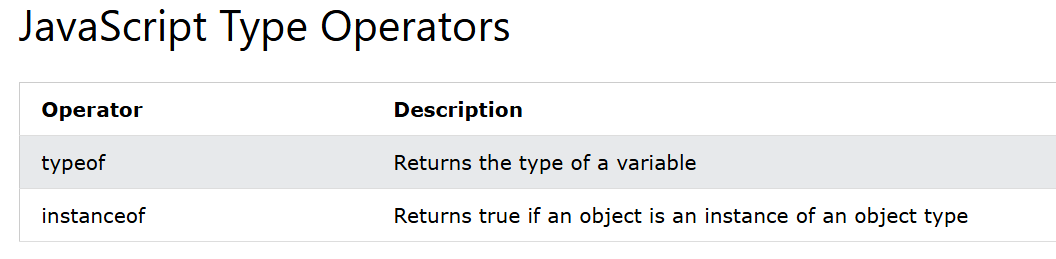
**Operators:**

****



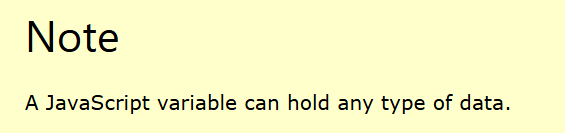






**JavaScript Data Types:**

|  |
| --- |
| // Object: const person = {firstName:"John", lastName:"Doe"};  // Array object: const cars = ["Saab", "Volvo", "BMW"]; 🡺 cars[0], cars[1], cars[2]. |



JavaScript evaluates expressions from left to right. Different sequences can produce different results.

|  |
| --- |
| let x = 16 + 4 + "Volvo";  let x = "Volvo" + 16 + 4; |

**JavaScript Types are Dynamic**

JavaScript has dynamic types. This means that the same variable can be used to hold different data types:

|  |
| --- |
| let x;       // Now x is undefined x = 5;       // Now x is a Number x = "John";  // Now x is a String |

Strings are written with quotes. You can use single or double quotes:

You can use quotes inside a string, as long as they don't match the quotes surrounding the string.

We can not use double quotes inside double quotes.

We can not use single quotes inside single quotes.

But still if you want to use then we need to use escaping(“\”) character.

We can use single quotes in double quotes and double quotes in single quotes.

|  |  |
| --- | --- |
| **Java script Object:**  {  empId: 90,  empName: "ec40884d",  empDOJ: "2025-07-11T11:56:02.591+00:00",  empSalary: 40,  isPermenentEmp: true  } | **JSON Object:** [**J**ava**S**cript**O**bject**N**otation]  {  **"**empId**"**: 90,  "empName": "ec40884d",  "empDOJ": "2025-07-11T11:56:02.591+00:00",  "empSalary": 40,  "isPermenentEmp": true  } |

The main difference between JavaScript object and JSON object is, JSON object keys have doublw quotes.

<https://www.w3schools.com/js/js_datatypes.asp>



Note

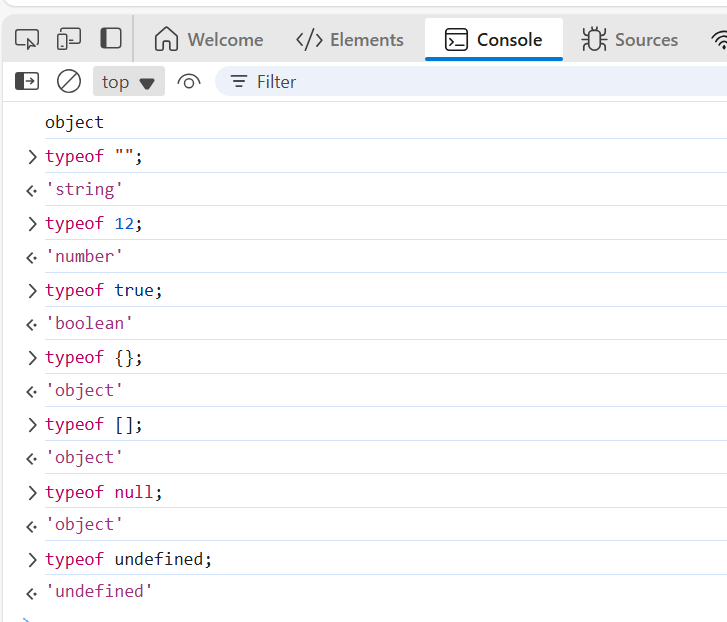
A JavaScript variable can hold any type of data.

**The typeof Operator:**

You can use the JavaScript typeof operator to find the type of a JavaScript variable.

The typeof operator returns the type of a variable or an expression:

|  |
| --- |
| typeof ""             // Returns "string"  typeof 0              // Returns "number" typeof 314            // Returns "number" typeof 3.14           // Returns "number"  typeof undefined // returns “undefined”,.  typeof null // returns object. |



**JavaScript Functions:**

A JavaScript function is a block of code designed to perform a particular task. 🡺 code within {} is called as block of code.

A JavaScript function is executed when "something" invokes/accessing/ calling it (calls it).

|  |
| --- |
| <body>    Enter First value: <input type="text" id="fValue"/> <br>    Enter Second value: <input type="text" id="sValue"/><br>    <button value="Submit" onclick="**add()**"> Submit </button>    <script>      function add(){        let fValue = document.getElementById("fValue").value;        let sValue = document.getElementById("sValue").value;        console.log(parseInt(fValue) + parseInt(sValue));      }        </script>  </body> |

Functions are **reusable block of code** designed to perform a particular task.

Functions **execute** when they are "called" or "invoked".

|  |
| --- |
| <html>  <!DOCTYPE html>  <html lang="en">  <head>      <title>Employee Form</title>  </head>  <body>      <input type="text" id="employeeId" placeholder="Employee ID"><br>      <input type="text" id="employeeName" placeholder="Employee Name"><br>      <input type="number" id="employeeSalary" placeholder="Employee Salary"><br>      <input type="text" id="employeePhone" placeholder="Employee Phone"><br>      <button onclick="sendEmployeeData()">Save Employee</button>  </body>  <script>      function sendEmployeeData() {          console.log(document.getElementById('employeeId').value);          console.log(document.getElementById('employeeName').value);          console.log(document.getElementById('employeeSalary').value);          console.log(document.getElementById('employeePhone').value);      }      function calculateSum(engMarks, mathMarks, scienceMarks) {          return engMarks + mathMarks + scienceMarks;      }      let result = calculateSum(10, 20, 30);      console.log("Sum of marks: " + result);  </script>l̥  </html> |

**Functions with parameters:**

While declaring parameters in JS functions we no need to mention data types like we do in java.

Directly we can declare parameter names, like below.

|  |
| --- |
| <script>      function calculateSum(**engMarks, mathMarks, scienceMarks**) {          return engMarks + mathMarks + scienceMarks;      }  **Example 1:**      let result1 = calculateSum("Kamal ", 23, true);      console.log("Sum of marks: " + result);      let result = calculateSum("Kamal ", 23, true);      console.log("Sum of marks: " + result);  **Example 2:**      let result2= calculateSum(12, “asasas”, true);      console.log("Sum of marks: " + result);  </script>l̥ |

In case of Java, because of datatypes in parameter declaation, we can pass arguments similar to data types we declared.

But in case of JS we can pass any kind of data to parameters. Means we can pass string values, integer values, Boolean values …objects, arrays …etc.

**Why Functions?**

With functions you can reuse code. Write code once and use it for multiple times in multiples places(html/js).

You can write code that can be used many times.

You can use the same code with different arguments, to produce different results.

**Function Invocation:**

The code inside the function will execute when "something" **invokes** (calls) the function:

* When it is invoked (called) from JavaScript code
* **When an event occurs (a user clicks a button)\*\*\*\***
* Automatically (self invoked) 🡪 put it in <script> tag.

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*FAT Arrow Functions? [A short cut form to regular function syntax].**

**We just convert normal functions into arrow functions.**

How to create Array function or Fat arrow function?

For our normal method remove “**function**” keyword. Remove the function name declaration.

Add fat arrow (=>) symbol after parenthesis.

**Below are the rules to convert normal function as fat arrow function. First create one normal function and apply below rules.**

* Remove “function” keyword.
* Remove the function name declaration.
* Add **=>** fat arrow symbol after parenthesis().
* Assign a variable name to fat arrow function. To utilize in some other places like calling on events etc.

|  |  |
| --- | --- |
| Regular function | Fat arrow function |
| function addition(a, b) {          return a + b;      } | let addition = (a, b) **=>** {          return a + b;      } |
| Let result = addition(45,657);  Consol.log(result); | Let result = addition (45,657);  Consol.log(result); |

Examples:

|  |  |  |
| --- | --- | --- |
| function myFunction(myvalue){  alert(myvalue);  } | function addition(firstValue, secondValue){  alert(firstValue + secondValue);  } | let a = () => {  console.log("Welcome to JS");  } |
| let aa = (vf, sv) => {  console.log(vf \* sv);  } | function abc(name){  alert(`my name is ${name}`);  } | let displayName = currentCount => {  alert(`my name is ${currentCount }`);  } |
|  |  |  |
| Functiom display(){  console.log(‘welcome to JS’);  } | Let display = () => {  console.log(‘welcome to JS’);  } |  |

Try converting normal functions to fat arrow functions with examples with NO Parameter function, with SINGLE Parameter function and with TWO Parameters function.

**Local Variables?** -> Variables which we declare inside functions or blocks or constructors are called as local variables.

Variables declared **within** a JavaScript function, become **LOCAL** to the function.

Local variables can only be accessed from within the function.

Since local variables are only recognized inside their functions, variables with the same name can be used in different functions.

Local variables are created when a function starts, and deleted when the function is completed.

**Parameters vs. Arguments**

In JavaScript, function parameters and arguments are distinct concepts:

**Parameters** are the **names** listed in the function definition.

**Parameters**  are the **names** of the values that will be passed.

|  |
| --- |
| <script>      function addition(a, b) **{// here a,b are called as parameters.**          return a + b;      }      let result = addition(5, 10); **// here 5,10 are called as arguments.**      console.log(result);  </script>l̥ |

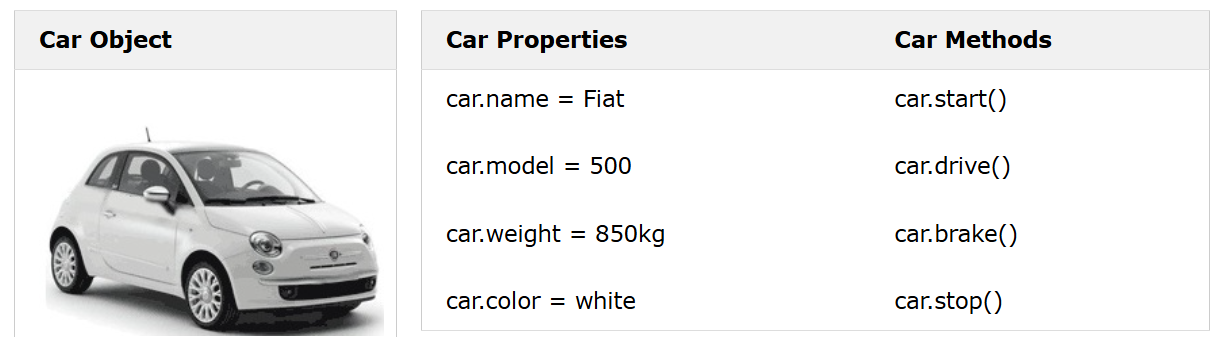
**Arguments** are the **values** passed to the function when it is invoked or called.

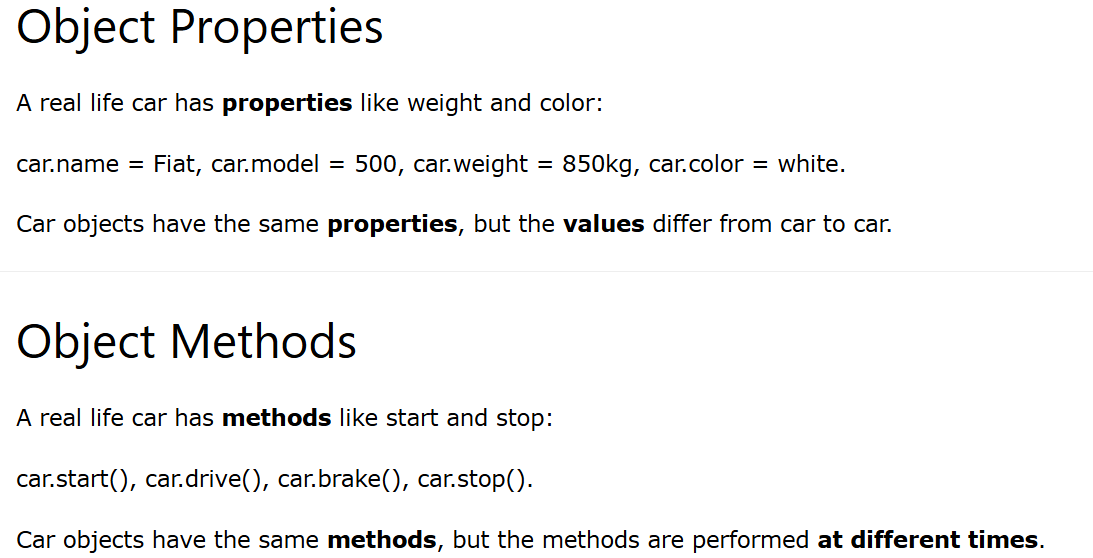
**Arguments** are the **values** received by the function.

**JavaScript Objects:** [State, Behaviour, Identity][SBI].

State, Behaviour, Identity(optional)

In real life, **objects** are things like: houses, cars, people, animals, or any other subjects.





**JavaScript Variables:**

JavaScript variables are containers for data values.

This code assigns a **simple value** (Fiat) to a **variable** named car:

|  |
| --- |
| let car = "Fiat"; |

**JavaScript Objects: {}**

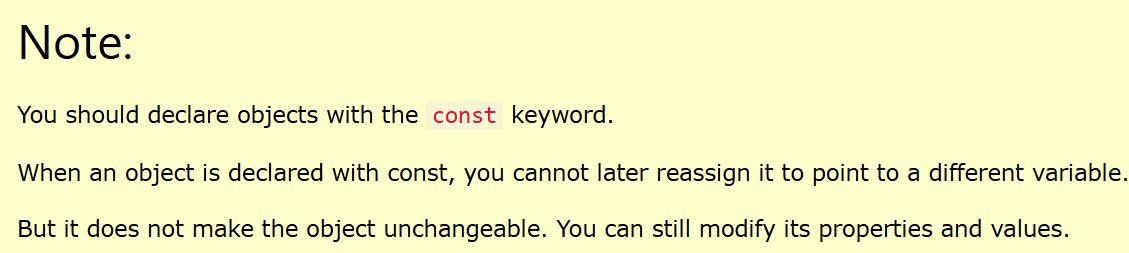
Objects are variables too. But objects can contain many values.

This code assigns **many values** (Fiat, 500, white) to an **object** named car:

|  |
| --- |
| const car = {type:"Fiat", model:"500", color:"white"}; |

|  |  |
| --- | --- |
| <script>  const car = **{**  **name**:"TATA",  **model**:"2025",  **price**:232323,  **color**:"red",  **drive**: **function**(){  return 'drive very fast....';  }  **};**  </script> | console.log(car.name);  console.log(car.model);  console.log(car.price);  console.log(car.color);  console.log(car.drive**()**); |

We can access object properties with object reference name, we can access object functions or methods using object reference variable, but while calling function or method we must use **“()”.**

****

|  |  |
| --- | --- |
| **WRONG approach of declaring object with let** |  |
| <script>  **let car = {**  name:"TATA",  model:"2025",  price:232323,  color:"red",  drive: function(){  return 'drive very fast....';  }  **};**  **car = "WHITE";**  console.log(car.name);    </script> | <script>  **const car** = {  name:"TATA",  model:"2025",  price:232323,  color:"red",  drive: function(){  return 'drive very fast....';  }  };  **car.color = "WHITE";**  **car.color = "WHITE";**  console.log(car.name);  console.log(car.model);  console.log(car.price);  console.log(**car.color);**  console.log(car.drive());  </script> |

**How many ways we can create JS Objects?**

* Using an **Object Literal** -> **{key:value, key: value,……….}\*\*\* here : is Column.**
* Using the new Keyword
* Using an Object Constructor

**Using an Object Literal:**

An object literal is a list of named values inside curly braces { }.

|  |
| --- |
| **{}** 🡪 empty Object.  **{key:value, key: value, functionName: function(){ ….. }, ……….} -> object with some data.** |

: 🡪 Column

; 🡪 Semi-Column.

**In JavaScript first we can create an empty object and then later we can assign properties also dynamically.**

|  |  |
| --- | --- |
| After creating object declaring the properties. | **While creating** object declaring properties also. |
| **const person = {}; // creating one empty object.**  // Add Properties person.firstName = "John"; person.lastName = "Doe"; person.age = 50; person.eyeColor = "blue";   console.log(person); | const studentObject = {        stuName: "Kamal",        stuId: 12,      reading: function () {          console.log("i am reading..asdadad..");        },        writing: function () {          console.log("i am writing..adasdasds..");        }      }      studentObject.stuName = "Azad..."; |

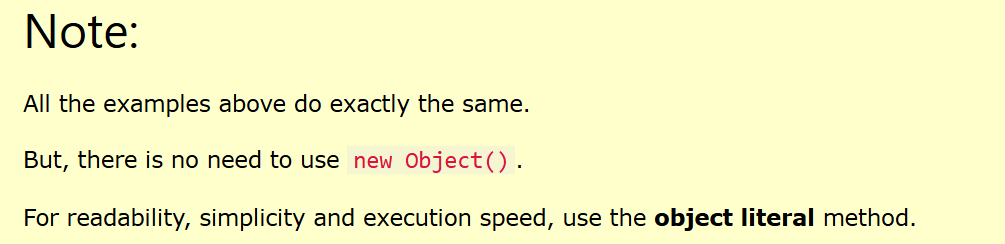
**2nd approach of creating JS object:**

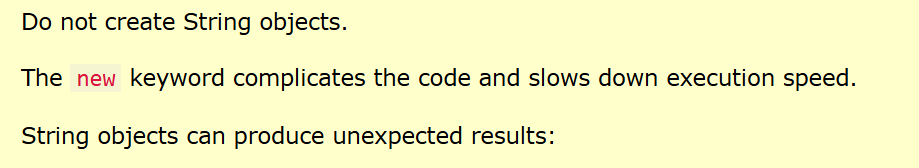
Creating JS object Using the **new** Keyword:

Create a new JavaScript object using **new Object():**

|  |  |
| --- | --- |
|  |  |
| // Create an Object const person = **new Object({ }); // empty object** | // Create an Object const person = **new Object({**   firstName: "John",   lastName: "Doe",   age: 50,   eyeColor: "blue" **}**); |

If we observe above, again in new Object() constructor we are passing “Object Literal - {}” only.





In JS or in Java we should not use “new” keyword multiple times. It will put burden on JVM in java also.

We should avoid creating objects repeatedly in JS or in Java or in any other technology.

In JS if we use “new” keyword for creating objects, browser performance will be slowdown. Loading the page will be slow.

**Accessing Object Properties:**

You can access object properties in two ways:

|  |  |
| --- | --- |
| *objectName.propertyName* | *objectName["propertyName"]* |
| console.log(**student.id**); | console.log(**student["id"]);** |

**“this” – always represents “current” object / current element.**

|  |  |
| --- | --- |
| **“this” with Correct example.** | **WRONG example.** |
| const person = {              firstName: "Shaik",              lastName: "Azad",              id: 5566,              fullName: function () {                  return **this**.firstName + " " + **this**.lastName;              }          };          console.log(person.fullName()); | **const person = {**  **firstName: "Shaik",**  **lastName: "Azad",**  **id: 5566,**  **fullName: function () {**  **return firstName + " " + lastName;**  **}**  **};**  **console.log(person.fullName());** |

**How many ways we can create functions in JavaScript?**

Total 4 ways we can create functions in JS.

|  |  |  |  |
| --- | --- | --- | --- |
| **Normal function** | **Variable function** | **FAT Arrow function** | **Nameless function** |
| function myFunction1() {        console.log(`myFunction1`);      } | let myFunction2 = function () {        console.log(`myFunction1`);      } | let myFunction3 = () => {        console.log(`myFunction1`);      } | function () {        console.log(`myFunction1`);      }  **Nameless FAT Arrow function:**  () => {        console.log(`myFunction1`);      }  **Single line FAT arrow function:**  () =>console.log(`myFunction1`); |

**Example:**

|  |  |
| --- | --- |
| setTimeout**(() =>** {        console.log("Hi…");      },2000); | setInterval**(() =>** {        console.log("HI…");      },2000); |

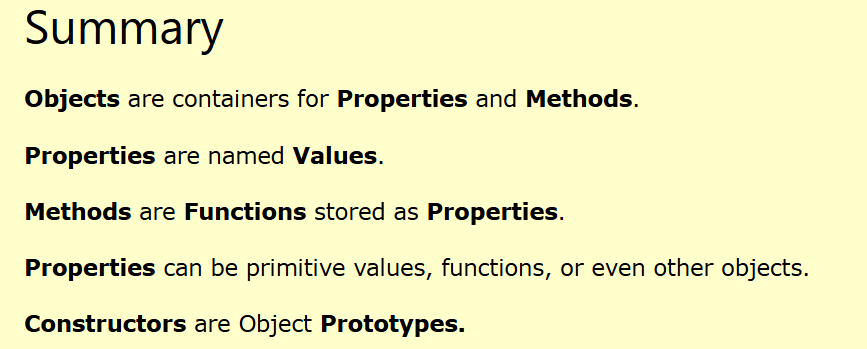
When a function have single line return statement, we can **remove curly braces {}** and we can remove “**return**” keyword also from the code.

|  |  |  |
| --- | --- | --- |
| Normal function with single line return statement | FAT arrow function with single line return statement | Single line FAT arrow function. |
| function sum(fValue, sValue) {  **return fValue + sValue;**     }    console.log(f33(23, 56)); | let f22 = (fValue, sValue) => **{**  **return** fValue + sValue;  **}**  console.log(f22 (23, 56)); | **let f33 = fValue => fValue + sValue;**  console.log(f33(23, 56)); |

**Below is WRONG:**

|  |  |
| --- | --- |
| function sum(fValue, sValue) {        console.log("this is Kamal");        alert(`this is Azad...`);        return fValue + sValue;      } | **let f123 = (fValue, sValue) => console.log("this is Kamal");**  **alert(`this is Azad...`);**  **return fValue + sValue;** |

**If we have multiple line to execute, we must use block symbol means we must use Curly braces {} in FAT arrow function.**



**In JavaScript, Objects are King.**

**If you Understand Objects, you Understand JavaScript.**

In java script for writing strings we can use either double quotes or single quotes.

|  |
| --- |
| let carName1 = **"**Volvo XC60**"**;  // Double quotes let carName2 = 'Volvo XC60';  // Single quotes |

<https://www.w3schools.com/js/js_strings.asp>

Template Strings? 🡪 for Template Strings we use back tick character. ``. For concatenation purpose and to overcome breaking lines into multiple lines.

|  |
| --- |
| let text = `He's often called "Johnny"`; |

**Interpolation: 🡪 `${variableName}`**

**Template Strings** allow variables in strings.

Template strings provide an easy way to **interpolate** variables in strings.

|  |
| --- |
| let firstName = "John"; let lastName = "Doe";  let text = **`**Welcome **${**firstName**},** ${lastName}!**`**; |

  == --> compares **content in object**.

  === --> compares content + object types are same or not in in object.

|  |
| --- |
| let name1 = "kamal";  let name2 = "kamal";  console.log(name1 === name2); 🡪 true |

|  |
| --- |
| let name1 = "kamal";  let name2 = **new** String("kamal");  console.log(name1 === name2); 🡪 false |

Comparing two JavaScript objects created with “new” keyword, **always** returns **false**.

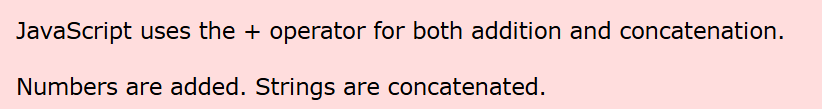
|  |
| --- |
| let name1 = **new** String("kamal");  let name2 = **new** String("kamal");  console.log(name1 === name2); 🡪 false |

Numbers:

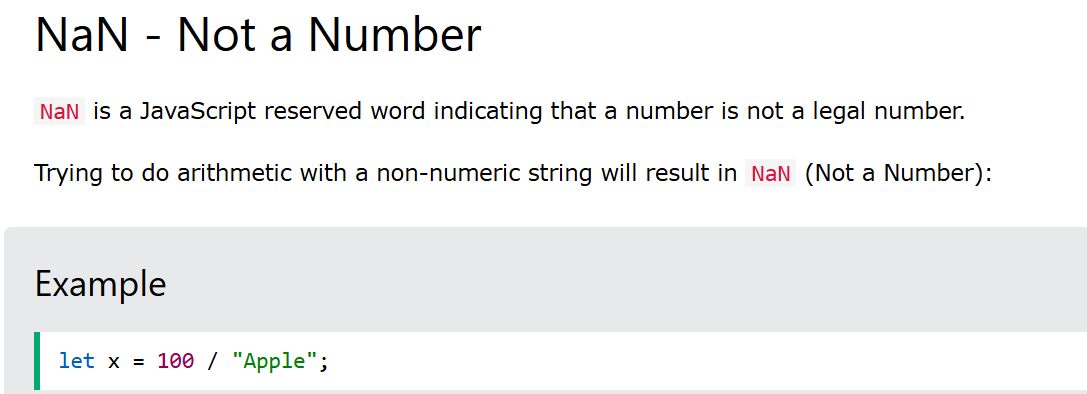
In java for decimals and non-decimals we have different datatypes. 🡺 byte, int, double, float.

In JavaScript for decimals and non-decimals we use same datatypes. 🡺 let OR const.

|  |
| --- |
| let x = 123**e5**;    // 123**00000** let y = 123e-5;   // 0.00123 |



|  |  |  |
| --- | --- | --- |
| let x = "100";          let y = "10";          let z = x / y;          console.log(z); 🡪 10 | let x = "100A";          let y = "10";          let z = x / y;          console.log(z); 🡪 NaN[Not A Number] | let x = "100";          let y = "10";          let z = x \* y;          console.log(z); 🡪 1000 |



You can use the global JavaScript function **isNaN()** to find out if a value is a not a number:

|  |
| --- |
| let x = 100 / "Apple"; isNaN(x);  console.log(NaN + 10); 🡪 NaN |

|  |
| --- |
| <input type="text" id="fNumber" placeholder="Employee ID">      <input type="text" id="sNumber" placeholder="Employee Name">   <button onclick="callMyFunction();">Click</button>   function callMyFunction() {              let x = document.getElementById('fNumber').value;                  let y = document.getElementById('sNumber').value;                  let result = x / y;                if **(isNaN(result))** {                  alert("Please enter valid number for division...");              }else{                  console.log(result);              }          } |

**JavaScript Arrays:**

Java Array example:

|  |
| --- |
| int[] intArr = {12,23,323,23,2,343,5,343,2323,4545,2323,56,467, 345}; System.*out*.println(intArr.length); System.*out*.println(intArr [intArr.length -1] ); |

But why we should go for arrays?

|  |
| --- |
| let name1 = 'Kamal'; // 1MB.          let name2 = 'Azd...';          let name3 = 'adasd';          let name4 = 'adasd';          // 1000 varialbes --> 1MB.          let studentName = ['121212','34343434']; // --> 5MB. |

**An array can hold many values under a single name, and you can access the values by referring to an index number.**

In Java we are going to use curly braces {}, but in JavaScript we are going to use square brackets []

**Java 🡪 {}**

**JavaScript 🡪 []**

|  |
| --- |
| const cars = **[**"Saab", "Volvo", "BMW"**]**; |

Key characteristics of JavaScript arrays are:

* **Elements**: An array is a list of values, known as elements.
* **Ordered**: Array elements are ordered based on their index.
* **Zero indexed**: The first element is at index 0, the second at index 1, and so on.
* **Dynamic size**: Arrays can grow or shrink as elements are added or removed. [**In java NOT allowed**]
* **Heterogeneous**: Arrays can store elements of different data types (numbers, strings, objects and other arrays). [**In Java NOT allowed**]

**let namesArr = ['A', true, {name: "kamal",id: 1234}, 123, [12,23,12,23]];**

**It is a common practice to declare arrays with the const keyword.**

What is the recommended approach for **javascript objects** and **javascript arrays** declaration? Which one use can use? Let or const? and why?

**let** objRef = {name: “Azad”} 🡺 **NOT RECOMMONDED**.

**const** objRef = {name: “Azad”} 🡺 **RECOMMONDED**.

Spaces and line breaks are not important. A declaration can span multiple lines:

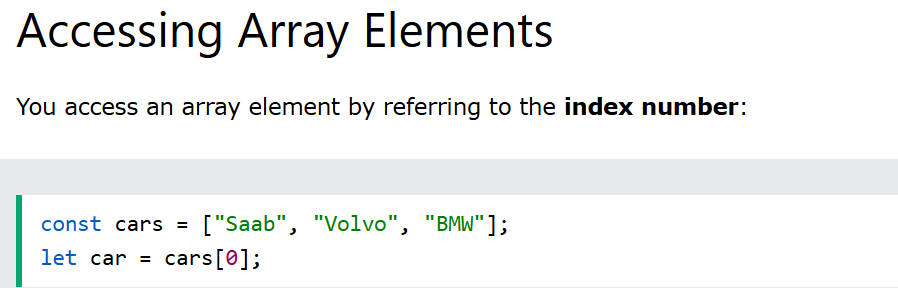
|  |
| --- |
| const cars = [   "Saab",   "Volvo",   "BMW" ]; |

We can do same approach for javascript objects also.

You can also create an empty array, and provide elements later in javascript.

But in java it is not possible.

|  |
| --- |
| const cars = [];  cars[0]= "Saab";  cars[1]= "Volvo";  cars[2]= "BMW"; |



**Note:** Array indexes start with 0.

[0] is the first element. [1] is the second element.

Changing an Array Element

This statement changes the value of the first element in cars:

|  |
| --- |
| const cars = ["Saab", "Volvo", "BMW"]; cars[0] = "Opel"; |

-We can create Arrays using “new” keyword also, but it is not recommended.

|  |
| --- |
| const cars = **new Array("**Saab", "Volvo", "BMW"); 🡪 **NOT RECOMMONDED.** |

Sorting of arrays:

|  |
| --- |
| let myArr  =  [56,1,45,67,3,556,12];             let newArr = myArr.**sort((a,b) => a-b);**             console.log(newArr); |

Looping Array Elements:

|  |
| --- |
| let myArr  =  [56,1,45,67,3,556,12];              for(let i = 0; i < myArr.length; i++)              console.log(myArr[i]); |

You can also use the **Array.forEach()** function for iterating values from array.

|  |  |
| --- | --- |
| let myArr  =  [56,1,45,67,3,556,12];               myArr.**forEach**(callBackFunction)             function callBackFunction(val){              console.log(val);             } | let myArr  =  [56,1,45,67,3,556,12];   myArr.**forEach**(val => console.log(val)); |

Adding and removing Array first index and last index elements Element using javascript methods.:

**Push(), pop();** 🡪 adding at the end of the array and fetching from the last index of the array.

**Unshift(),shift();** 🡪 adding at the start index of the array and fetching from the start index of the array.

|  |
| --- |
| let myArr = [56, 1, 45, 67, 3, 556, 12];              //    myArr.push(343434);              //let lasVal = myArr.pop();              // myArr.unshift(3333);              // let v = myArr.shift();              console.log(v); |

**Note:** In java we don’t have methods support to arrays, due to that reason only they introduced collections of framework.

Java array support only **length** property.

**Means JavaScript arrays have methods support, but java doesn’t have any methods support.**

**Java gives methods support by some additional feature called java 8 streams.**

**The Difference Between Arrays and Objects:**

In JavaScript, **arrays** use **numbered indexes**.

In JavaScript, **objects** use **named indexes**.