What ?

JavaScript is a programming language, generally used for web-application….it is used to add functionality to our webpages.

Two ways to link Javascript

1. Internal linking – using script tag inside html body tag
2. External linking – create separate js file and link it to HTML page

Note – In one HTML file we can have n numbers of script tag and they are executed from top to bottom.

How javascript is executed in Web Browser ?

We have two engines in our web browser

1. Rendering engine 🡪 used for rendering HTML and css
2. Execution engine 🡪 used for executing javascript code

Node js is a run-time environment which is used to execute our javascript code outside the web.

Built on google’s V8 javascript engine with some additional feature.

**Variables -** variable are named memory location which we use to store our data.

* In javascript variable are declared using

1. var
2. let
3. const

and not datatypes like java

* For example let a = 30 ;
* In javascript variables are not strictly-typed, they are dynamically typed means we do not have to mention datatype while declaring the variable, on the basis of data we store in the variable, the variable datatype is decided.

**var**

* var scope is global

**let**

* let scope is local
* For ex let a

console.log(a) //undefined

Note- we should avoid var.

**const**

* const scope is local
* must be initialized at time of declaration only and re-initalization is not possible.
* For ex const b= 52

console.log(b) //52

**DATATYPES IN JAVASCRIPT**

PRIMITIIVE DATATYPES

1. number 🡪 In js int,float,double all values comes under number datatype only.

Note – NaN(Not a Number is not a datatype in javascript rather it is a special value which comes under number datatype only, it comes when programmer wants to perform arithmetic operation on values but arithmetic operation is not possible with those values.

For ex- console.log(“major” + “captain”) 🡪 majorcaptain (here concatenation happens)

console.log( “major” - ”captain” ) 🡪 NaN

console.log(“a” \* 1) 🡪 NaN

console.log( “10” \* 1) 🡪 10 (string will be converted to a number)

console.log(“10” - 4) 🡪 6 (string will be converted to a number)

1. string 🡪 “harshan”, “a” , ‘Vivek’

Converting string to a number

console.log(“10” \* 1) 🡪 10 (here 10 will be a number)

1. boolean 🡪 true , false ……..internally true is 1 and false is 0 in javascipt

For example

console.log(10+true) // 11

console.log(10+false) // 10

Note – Every non zero value is true.

1. null
2. undefined

Difference between null and undefined

null

* null is assigned by programmer explicitly, in case of absence of value.

undefined

* undefined is assigned by the javascript interpreter implicitly, when programmer do not initialize the variable.

**NON PRIMITIVE DATATYPES**

1. object

* represented by {} (include arrays, functions, objects)
* for ex - let heroes = {name: ”Captain Manoj Pandey”

medal: “Ashok Chakra”

}

**OPERATORS**

1. Assignment operator (=) -> this operator is used to assign value to the variable.

For ex - let a = 30

modulus operator – gives remainder. For ex 9/2 = 1

1. Arithmetic operator ( + , - , \* , / ,% , \*\* , ++ , --)

Exponential operator - number raised to the power. for ex 2^3 = 8

Division operator – use to divide the number. For ex- 9/2 = 4.5 (in js it will

give floating point answer, even though both the values are of int type

because here int and float both comes under number datatype)

1. Comparison operator – used to compare two values.

That first value is < , > , <= , >= , != , == , === than the second value. It will always return the answer in Boolean datatype .

For ex-> console.log(20>=30) // false

== equal to

In equal to operator only values will be checked not datatype.

For ex console.log(300==”300”) //true because values are equal

=== strict equality operator

In strict equality operator both datatype and values will be checked. Both should be equal only then the answer will be true.

For ex console.log(300===”300”) // false because datatype is different

1. Logical operator( && , || , ! )

&& Logical AND

Both should be true, then only the answer will be true otherwise false

let value1 = true

let value2 = false

For ex- console.log( value1 && value2) //false

!! Logical OR

If any one would be true, answer will be true

For ex- console.log(value1 || value2) //true

! Logical NOT

Inverts the boolean value.

For ex - console.log(!value1) //false

1. typeof() operator

* It is used to know the datatype of values
* For ex- console.log( typeof(“123) ) // string

1. Ternary Operator

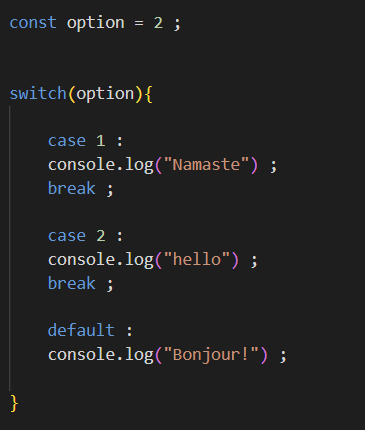
* Shorthand property of if-else
* Syntax

condition ? true : false

**Conditional Statements**

1. if else
2. if else if
3. ternary operator
4. switch

switch case



**LOOPS**

**When to use**

1. for 🡪 when we know the exact number of iteration
2. while 🡪 when we don’t know the exact number of iteration but we know the

condition

1. do while 🡪 when we want atleast one time our code runs because in do while, first

our code runs and then condition will be checked.

**ARRAYS**

In js, array is a variable which can store multiple values of different types.

In js Arrays are dynamic

**How to declare an array**

let a = new Array()  ;

let b = new Array(5) ;

let c = new Array(23,"Captain Vikarm" , 45.23, null) ;

let d = [10,"major Vivek" , 52.36 , false, null] ;

**Methods of Array**

push(ele) 🡪 add element to last 🡪 returns new length of array

pop() 🡪 remove last element 🡪 returns element which is removed

unshift(ele) 🡪 add element to first 🡪 returns new length of the array

shift() 🡪 remove first element 🡪 returns element which is removed

length 🡪 calculate length of array 🡪 returns length of array

concat(ArrayName) 🡪 concat two arrays 🡪 returns new array with elements of both array

join(separator) 🡪 convert array to string 🡪 return String

Note – By which you want to separate string. Give “” (empty string) to simple convert array elements to String.

reverse 🡪 to reverse the array 🡪 returns array with elements in reverse order

indexOf(ele) 🡪 to know the index of particular element 🡪 return index of first occurrence of

element if found otherwise return -1

slice(startingIndex , endingIndex) 🡪 to get some part of array 🡪 return an array of elements from

starting index to ending index-1

splice(start, deleteCount, item1,item2….) 🡪 used to delete, add or replace elements in an array

🡪 modifies the original array and returns an array of the

deleted elements (if any)

Parameters:

* start (required):  
  The index at which to start changing the array.
* deleteCount (optional):  
  The number of elements to remove starting from start.
* item1, item2, ... (optional):  
  Elements to add at the start index.