Saturday 1-02-2026

- Global context referes to the default environment where code runs.
- When the Javascript engine starts execating code.

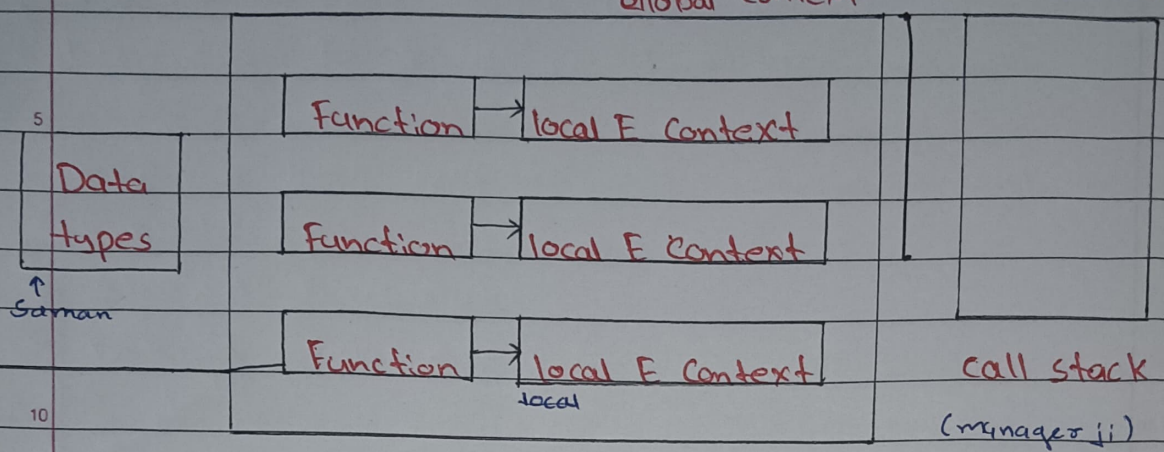
M T W T F S

Date : / /

Page No:

Javascript Classes

QA :- Quality Aanalysis (factory me jo kam hota hai)



- local execution context, more formally known as a function Execution Context (FEC), is a private environment created by the Javascript engine every time a function is called.

- Every function has these own:

actual cook	Board
main function run execution thread	variable, function ko rakhana memory

```
Const number = 5;
Function addTwo(num) {
        return num+2
}
Const valueOne = addTwo(number)
```

memory
number=5
addTwo(num)

mini factory

new variable who calles the function

main thread:- check the code from top to bottom like interpretator

M T W T F S

Date : / /

Page No:

How all work together

- When new variable calles the function.
- addTwo number execates to function locally.
- But the first it is add to the call stack
- then memory provide Data to the function which is sitting in the call stack.
- and the memory is the global execution memory.
- then execution thread execute line by line.

Programming is all about

- data --- string, ~~function~~ boolean
- Processing -- loop, function, variable

- Broaser only understand

HTML →
CSS → Broaser
JS — parse to HTML

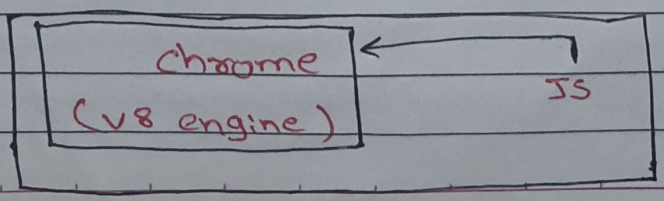
JS Engine Names

- V8 engine → crome
- Spider Monkey → Firefox
- Safari → Webkit

without engine you can't run JS in your browser.

- Which compiler need to run code

C, C++ = CPP Compiler
Java = Java Compiler
Javascript = JS ~~compiler~~ engine.



M T W T F S

Date : / /

Page No: _____

nodeJs inventor

Ryan Dahl

- It is not a framework or not a library
- It's run time environment for js
- for run Js ~~on~~ ~~runs~~ outside the browser

- there is two way to add Javascript in your HTML file
- external Javascript
- Internal Javascript

Developer tools

- Element ⇒ for HTML SSS
- Console ⇒ it's like terminal or the output of your Javascript
- Sources ⇒ to see the code
  Kaha ~~ka~~kaha se lode hora vo
- network ⇒ to see if your browser calles external api's
- performance ⇒ as name says

- Js is loose typed language
- TypeScript is strongly typed languages.
  and to browser don't anderstand typeScript you have to convert typeScript in to Javascript then it's works.
  typescript is just pro version of Javascript

JavaScript is dynamicaly type language.

SHREE

Date: / / Page No.

Functions - set of instruction

- single source + use again and again.

~~function addNumbers () {}~~

name of function

```
function addNumbers (num1, num2) {
        var result = num1 * num2;
        Console.log ('Result is', result);
        }
addNumbers (2,3);
```

Parameters

Conditionals

- Kisi perticular conditions ke upper kisi perticular block of code ko run karna.

(True & false are the Boolean value)

fancy word :- Verbose using more words than necessary to express

Ex.

```
Var age = 22;

Var childCondition = age <= 12;
var teenCondition = age <= 19;
var adaltCondition = age <= 40;
var seniorCondition = age > 40;

if (childCondition) {
    Console.log ('You are a child');
  }
if (teenCondition) {
   Console.log ('You are a teen');
  }
```

```
if (teenCondition
if (adultCondition) {
    Console.log ('You are an adult');
  }
if (seniorCondition) {
    Console.log ('You are a senior');
  }
```

Output

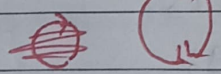
you are an adult

&& check bothe the condition

|| check either this or this
dono mese kon sa bhi chalega

! to represent not

else if use if you have too ~~condition~~ output But you want to print base on condition.
if condition maches print this or this.

loops  gol-gol

- doing thing again and again

- for loop
- While loop
- Do while loop
- forEach loop
- forof loop
- map
- forIn loop
- Iterators
- filters
- Reduce
- Entries
- keys

for loop

Syntax

```
for (initializer ; condition ; increment ) {
    // code jisko loop karwana hai
  }
```

example,

```
for ( var x = 1 ; x <= 10 ; x = x + 1 ) {
      console.log ('Value of x', x);
    }
```

(init = initializer, condition = condition, increment = increment)

- condition are always be boolean
- increment mean add +1 in next iteration until condition matches.

Dry Run

x	condition = x <= 10	x = x + 1	Console
1	1 ≤ 10	true	1
2	2 ≤ 10	true	2
3	3 ≤ 10	true	3
4	4 ≤ 10	true	4
:	:	:	:
10	10 ≤ 10	true	10
11	11 ≤ 10	false	

Loop tut gai

- loop check the condition ~~to~~ ~~tot~~ 11 th time
- But run the code only 10 th time

Use:- When you exactly know kitni baar code ko ghumana hai, matlab run karna hai.

use: code ko tab tak ghumao jab tak kaam na para hojaye
- only knoco the condition but don't knoco hoco many iterations

SHREE
Date: / / Page No.

While loop

ex,

When you doonloading a file, you don't knoco how much time it's gona take, but end goal is you get the file from the server.

```
get random number

function getRandomNumber (min, max) {
    return Math.floor (Math.random () *
                        (max - min + 1)) + min;
}
```

example,

```
var fileSize = 1024;
var currentfileDoonloaded = 0;

while (currentfileDoonloaded < fileSize) {
    console.log ('file ko Doonload krte Jao');
    currentfileDownloaded = currentfileDoonloaded
                            + 512;
}
```

Or using random number

```
while (currentfileDoonloaded < fileSize) {
    console.log ('file ko Doonload krte Jao');
    currentfileDoonloaded = currentfileDownloaded +
                            getRandomNumber (20, 40);
}
```

- check the condition then iterate

use:- Do-while ek tarh ki while loop hi hai, but first he execute the code the check the condition
- It ensure that you code at lest one runs.

SHREE
Date: / / Page No.

do-while loop

example,

```
Var fileSize = 1024;
Var currentfileDownloaded = 0;

do {
   console.log ('file ko Download krde jaO');
   currentfileDownloaded = currentfileDownloaded+
                           512;
} while
      (currentfileDownloaded < fileSize);
```

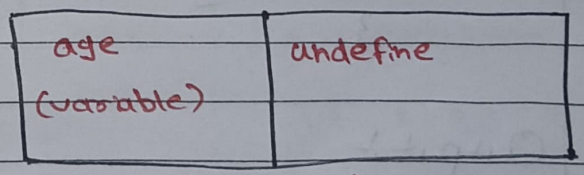
example

```
Console.log ('Age is', age);
var age = 32;
Console.log ('Age is', age);
```

pehle memory phase pehle run hota hai ise Hoisting kehte hai

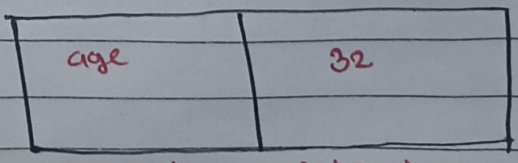
```
Console.log ('Age is', age);
var age = 32;
```

scane line by line



output:- Age is undefined.

```
var age = 32;
Console.log ('Age is', age);
```



output:- Age is 32

Hoisting is a javascript mechanism where variable, function, and class declarations are conceptually moved to the top of their containing scope (script or function) during the compilation phase, before code execution. !

- This allows functions to be used before they are defined and allows variable to be accessed without throwon error, though var variable return undefined.

example:

```
age = 45;
Console.log ('Age is', age);
Var age = 32;

hello ();
Console. log ('Age is', age);

Function hello () {
    Console.log ('This is hello');
    }
```

Output :-

Age is 45
This is hello
Age is 32

example

```
Console.log ('value of age is', age);
var age = 45;
Console.log ('Adding 5 to 10', addFive(10));
Console.log ('value of age is', age);

function addFive(number) {
    var result = number + 5;
    return result;
}
```

call the function

Dry Run

memory phase	Code phase
age = ~~undefined~~ 45	undefined

function addFive(number){}
var result = number + 5;
return result;
}
	MP: number = 10 ~~undefined~~, result = 15 ~~undefined~~
	CP:
	addFive ka execution context
	find 45

Output :-

value of age is undefined
Adding 5 to 10 15
value of age is 45