**Java script**

1: Word vs Keyword

anything which don't have any meaning in js is called word or those words which having some meaning in js is called key word i.e

words: beautiful, brush,

keyword: for, let, if

**2: variables(var) & constant(const) & let**

if we want to store any type of data in our code then the variable comes in action.

variables and constants both are English words used for storing data but the main diff is...

variables mean which change their value

constant means which do not change their value.

**3: defined & un-defined & not defined**

any variable which are declared or also initialized is called defined

any variable which are declared but not initialized is called un-defined

any variable which are not declared or not even initialized is called not defined.

example:

var a; print(a) // un defined

var b = 12; print(b) // defined

print(c) // not defined

**3: hoisting**

any function or variable which is used before its declaration is called hoisting

when we declared any variable or function after their calling is moved on the top of the code which is called hoisting

4: types in JS

primitive & reference

primitive types wo hoti hain jo haqeeqi data ki copy banati hain, jab ham inhy kisi dusre variable ma assign karty hain to in ma data ki real copy hoti hai yani agar ham asal value ko modify karain to copy data par koi asar nahi hoga.

i.e

number, string, null, undefined, boolean

reference types wo hoty hain Jahan variables ma asal data k bajaye is ka reference ya address store kiya jata hai, jab ham kisi reference type ki value ko dusre variable ko assign karty hain to dono variables aik hi memory address ko point karty hain, is ka matlab ye hai k agar ham aik varaibe ki value ko modify karain to dono variables ki value change ho jaye gi q k wo asal object k sath munsalik hoty hain.

(any values which are inside the brackets "(), [], {}" is called reference types)

i.e

arrays, object, class

**5: conditionals "if, else, else-if"**

6**: loops (for, while)**

**7: functions**

its means that we provide some name to the specific code

it is mainly used for three main purposes

jab ma apna code foran na chalana chahta hun.

jab mujhy apne code ko reuse karna ho

jab mujhy apna code reuse karna ho har baar with diff data

Parameters & arguments

arguments are real values jo ham function call karty wakt pass karty hain function ma.

parameter are those variables jin ma values store hoti hain arguments waali. Jo values function catch karta hai

**8: Arrays**

ham aik variable ma aik value store kar sakte hain par jab hame aik se ziyada value aik hi variable ma store karni ho to ham array use karty hain (array = group of values)

i.e

var a = 12,13; // wrong

var a = [1,2,3,4] //correct

**9: push pop shift unshift splice:**

push will add the value after the last index in array

pop will delete the last index value from an array

unshift will add the value at first index in array

shift will delete the first index value from an array

splice will remove value from middle of an array it will take two values first is index where the value is placed second how many values we want to remove if we provide 1 it will delete only one value if we provide 2 i will delete two values and so on

**10: Objects**

for example we discus about persons

let suppose if we discuss about multiple persons then we use arrays

if we want to discuss about multiple things related to one person then we use objects, means objects will save details of one individual person

blank object:

var person = {};

filled object:

var person = {

age: 24,

name: "syed",

email: "syed@gmail.com",

contact: 03210000111

}

how to call properties of objects

print(person.email) // syed@gmail.com

**Methods in object:**

jab ham object ki kisi b property ko function assign kar dain to wo method ban jata hai. i.e

var person = {

age: 24,

name: "syed",

email: "syed@gmail.com",

contact: 03210000111,

class: function(){}

}

**Delete from objects**

If we want to delete complete property from object not only value so we use

Delete person.email;

**Advanced JS**

**var let const (diff)**

i: JS having two versions es5, es6. in version es5 there are only var datatype but in es6 there are newly introduced data types let and const for declaration variables. It means es6 introduced let and const as new ways to declare variables, alongside it already existing var.

ii) var is function scoped means var is used any where inside there parent function.

let is braces(brackets {}) scoped means it is only used inside there braces.

i.e,

function abcd(){

for(var i=1; i<12; i++)

{

print(i); // 1,2,3,.....,11

}

print(i); // 12

}

// 1,2,3,.....,12

function abcd(){

for(let i=1; i<12; i++)

{

print(i);

}

print(i); // i is not defined

}

// 1,2,3,.....,11, i is not defined

so both code having diff results due to var and let keyword.

iii) var adds itself to the window object.

let, const doesn't add itself to the window object.

**now what is window:**

JS ma kafi features available hain par kafi features aise hain jo JS ma ni hain laikin wo features pir b JS use kar laity hai with the help of window, window aik box of feature hai jo browser daita hai JS ko.

i.e,

Imagine you have a pen, a piece of paper, and a box (which we'll call the window). The pen represents JavaScript.

i) Basic Task (Pen alone): You can ask the pen to make a hole in the paper, and the pen will do it. You can also ask the pen to write something on the paper, and the pen will handle that as well. These are basic tasks that the pen (JavaScript) can perform directly.

ii) Complex Task (Pen needs help): Now, suppose you ask the pen to make a mobile phone holder. The pen can't do this on its own, because it's not designed for that specific task. So, the pen will go to the box (the window object) and ask it for a mobile phone holder. The box (window) provides the holder, and the pen then gives it to you. The pen still helps you, but it relies on the box for the task.

some window functions which is used with the help of window:

alert(),

prompt(),

print(),

console(),

clearInterval(),

clearTimeout(),

document(), etc many more.

**Stack:**

**Heap Memory:**

Jitne b variables ya data hum banante hain unhe store karne k liye use hota hai heap memory.

**Execution Context, Lexical Environment:**

it is a container where the function code is executed and it is always created whenever a function is called

jab b ham function chalayn gy to fun apna aik khud ka imaginary container bana le ga jis ma uski 3 cheezain hon gi..

i) variables

ii) fun inside that parent fun

iii) lexical environment of that fun

ye jo imaginary container hai ise ham execution context kahty hain

**what is lexical environment:**

ye hame batata hai k ye fun apny andar bani kon si cheezon ko access kar sakta hai or kin ko nahi.

i.e:

function abcd(){

var a = 12;

function def(){

var b = 21;

}

}

now in lexical environment tell us that abcd() fun only access "a" variable and "def()" function but cannot access "b" variable which is inside the child func "def()", this is the work of lexical environment.

lexical environment aik chart hota hai jis ma ye likha hota hai k ap ka particular function kin cheezon ko access kar sakta hai or kin ko nahi kar sakta, matlab k it holds its scope and scope chain

i.e

We have three functions (or possibly more) that are nested inside each other, like this:

fun first(){

var a = 1;

fun two(){

var b = 2;

fun three(){

var c = 3;

}

}

}

In JavaScript, due to lexical scoping, each child function can access all the variables and properties of its parent functions. For example, function two can access the variable "a" from function first, and function three can access both "a" from first and "b" from two.

However, the parent function cannot access any variables or properties from its child functions. So, function first cannot access "b" or "c", and function two cannot access "c".

This behavior is due to the lexical environment in JavaScript, which defines the scope of variables based on where the functions are written in the code.

**How to copy reference values**

to copy reference values, we use spread operators denoted by three dots (...),

i.e:

var arr1 = [1,2,3,4,5];

var arr2 = [...arr1];

arr2.pop();

print(arr1) // 12345

print(arr2) // 1234

due to copy with spread operator now if we make changes on second array it will not effect on the first array.

same working for objects

var obj = {

name:"syed",

age: 25

}

var copyObj = {...obj}

**truthy or falsy**

JS main kuch b likh lain wo mainly 2 possibilities ma se kisi aik ma belong karti hai truthy ya pir falsy

means ham jo b likhty hain wo convert ho jati hai truth ya false ma.

falsy values:

0, false, undefined, null, NaN, document.all

truthy values:

all remaining values accept falsy values

1, 1.3, "syed", true, -2, many more

**For each, For in loops**

ForEach loop sirf array k liye use hota hai, matlab agar ham array k har element par kuch karna chah rahy hon to ham foreach loop use karty hain.

foreach loop kabi b original array ma change ni karta means ham foreach laga kar kisi b array ma koi b kaam karty hain to original array wohi same rahti hai, means jab b foreach loop lagta hai to wo array ki aik temporary copy bana laita hai jis ma changes ho rahi hoti hain.

foreach aik function laita hai hamesha jis k parameter ma array ki temporary copy ati hai or wo usy har aik elment k liye chalata hai

i.e

var arr = [1,2,3,4,5]

arr.forEach(function(val){

print(val+2);

})

For in loop sirf use hota hai objects k liye.

Var obj = {

Name: “syed”,

Age: 24,

City: “Peshawar”

}

For(var key in obj)

{

Print(key);

}

This code of for in loop will print only the keys of object like name, age, city but we need the values of those keys as well so we achieve it like this

For(var key in obj)

{

Print(obj[key]);

}

This will gives us the values like syed, 24, Peshawar.

This is the syntax of for in loop first we write for then we make a variable which is iterate in object using “in” keyword.

**Callback functions:**

**Aisa code jo kuch dair baad chalta ho hum usy aik function de dait hain k jab ye fun complete ho jaye to chala daina or wo fun jo ham daity hain wo normal func hota ai or usy ham callback fun kahty hain.**

**Callback function wo function hota hai jo doosre function ke kaam khatam hone ke baad call hota hai. Yeh approach tab use ki jati hai jab aapko koi kaam delay ke baad ya kisi event ke hone par chalana ho.**

**First Class functions**

**JS ma aik concept hai jiska matlab hota hai k ap fun ko use kar sakty hain as a value**

**Is concept ka matlab ye hai ke functions ko JavaScript mein kisi bhi value ke tarah treat kiya ja sakta hai. Functions ko variables mein store kiya ja sakta hai, arguments ke tor par pass kiya ja sakta hai, aur doosre functions se return bhi kiya ja sakta hai. Is tarah ke behavior ko hi first-class citizens kahtay hain.**