**Javascript Question**

**---------------------**

**Question 1:- Why do we call JavaScript as dynamic language?**

**Ans: JavaScript is a dynamic language means data types of the variables can change during the run times.**

**<script>**

**Var x = 0;  
x++;  
x=”Text1”  
x=true**

**</script>**

**Question 2:- how does JavaScript determine data types ?**

**Ans:- JavaScript determines data types depending on the value assigned.**

**Question 3:- What is type of functions?**

**Question 4:- How to check data type in JavaScript ?**

**Ans:- We can get datatype by using “type Of” function.**

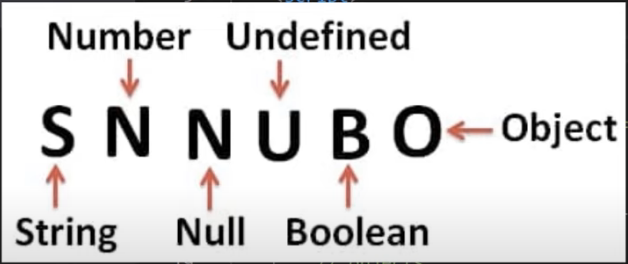
**Question 5:- What are different datatypes in JavaScript ?**

**//primitive**

**Var str = “Shiv” //string  
var num = 10 // number  
var nu = null //Null  
var und = undefined  
var bool = true // bool  
var big = 100 //big int  
var sym = Symbol() // symbol  
  
//Objects  
var obj = new Object() // Object**

**8 types of datatypes in JavaScript and we can classify these 8 datatypes into 2 categories.**

**Most used datatypes – string, number, null, undefined, bool and object**



**Question 6:- Explain Undefined Data types ?**

**Ans: Undefined means the variable has been declared but no value is assigned to it.**

**8:36 Question 7:- What is Null ?**

**Ans: - Null indicate intentional absence of data. Null indicates it not ZERO, its not empty its just absence of data.**

**Question 8:- Differentiate between Null and Undefined ?**

**Undefined: - Variables has been created but value is not assigned.  
Null:- We assign value NULL, it indicate absence of data.**

**Question 9:- Explain Hoisting ?**

**It’s a mechanism where variables and function declaration are moved to the top of the scope.**

**Question 10:- Are JavaScript initialization hoisted ?**

**No**

**Question 11:- What are global variables ?**

**Ans: - Global variables are accessible through out the webpage or the document.**

**Question 12:- What are the issues with Global variables ?**

**Ans:- Problem with global variable is it can make application very hard to debug and buggy.**

**Question 13:- What happens when you declare variable with out VAR ?**

**Ans: - When we declare variable without var keyword, variable become global.**

**Question 14:- What is Use Strict ?**

**Question 15:- How to force developers to use Var keyword ?**

**Use Strict is a directive which says “strictly check if the variables are defined using the var or let keyword” .If not defined the it will throw an exception**

**Question 16:- How can we handle Global Variables ?**

**Question 17:- How can we avoid Global variables ?**

**Summaries Ans : its difficult to avoid global variables. But we can organize it properly by doing two things: -  
  
Put global variables in proper Namespace. Module pattern using closures and IIFE.**

**1) We cannot avoid global variable(singular) but we can avoid global variables. As a best practice we can put global variable in a proper common namespace so that it is not polluting all across the page. Rather than creating individual global**

**Var global = {}  
global.connectionString = “Test”  
global.logDir = d:\Logs;**

**2) Create closure and apply module patten,   
var myGlobal = (function(){  
 var connectionStr = “Test”  
 function getConnect() {  
 return connectionStr  
 }  
return {  
getConnect  
}  
})();**

**Question 18:- What are Closures ?  
Question 19:- Why do we need Closures ?**

**Ans: Closures are function inside function and it makes a normal function stateful.**

**Why we need closures: The only one goal of closures is to create self-contained function/modules, self-contained state. To avoid global variable**

**Self Contained Modules -🡪 self contained State -🡪 Solve global variable problem**

**function ClosureFunction() {  
var x;  
function Increament() { x++}  
function GetXValue() { return x}  
function Init() { x = 0}**

**//Make function public  
return {  
 Increament,  
GetXValue  
}  
}**

**Var ref = ClosureFunction();  
ref. Increament();**

**// I didn’t exposed out init function.. this is again the biggest benefit of clousres…  
Abstraction show only what is needed to the external work.init is called from the   
function but available outside.**

**Question 20:- Explain IIFE ?**

**Immediately invoked function Expression, it’s an anonymous function which get immediately invoked.**

**Question 21:- What is the use of IIFE ?**

**Question 22:- What is name collision in global scope ?**

**Question 23:- IIFE vs Normal Function?**

**Ans:-**

**Name Collision : happened when same name function names and variable name are declared**

**function Init() { var x = 0}  
var Init = 0;  
Init();**

**Because IIFE does not have name. so there is no way you can get name collision**

**IIFE vs Normal Function: - A normal function has a name while IIFE does not have name. So with a normal function you can have a name collision but with IIFE you will not have name, you will not name collision.**

**Question 24:- What are design patterns ?**

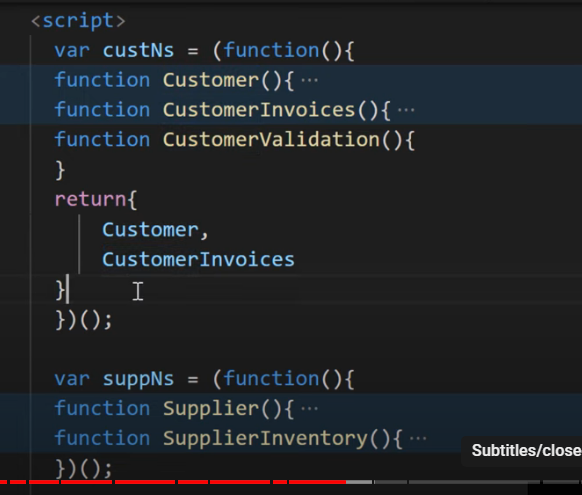
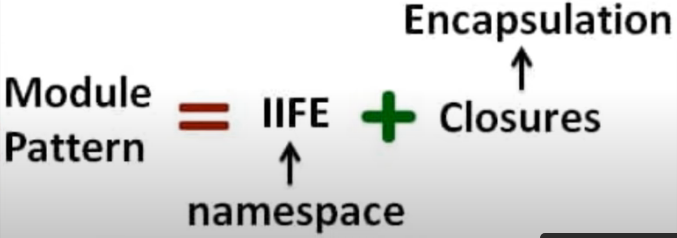
**Question 25:- Which is the most used design pattern?**

**Ans: - Design patterns are time tested architecture solution  
So, to create a single instance we can use singleton pattern, if the object creational  
process is complex, we can use factory pattern and so**

**most used design pattern:- Module design pattern or the module revealing pattern.**

**Question 26:- What is module Pattern and revealing module pattern ?**

**Module design pattern is most used design pattern in the JavaScript community and it helps in two way, it helps you to create self-contained independent component modules namespaces that helps in coupling and well structure code.**

**Module pattern or revealing module pattern has 2 big advantages: -   
-- Self-contained independent component  
-- Provide Encapsulation and Abstraction.  
  
Encapsulation means we can control what we want to make public what not  
  
Module Design pattern is a combination of Closures and IIFE** **var cust = new custNs.Customer() --- it will work, but  
var cust = new custNs.CustomerValidation() --- it will through error**

**Question 27:- What are the various ways to create JavaScript objects?**

**4 ways:   
1) Literal  
var pat = {“Name”: “”, address: “”}  
pat.Admin = function(){alert(“I am admitted“)}  
  
2). Object.create  
var patNew = Object.create(pat)  
patNew.age = 10  
  
3. Constructor  
function Patient() {  
 this.name = “”,  
 this.address = “”,  
 this.Admin = function() {}   
}**

**Var pat1= new Patient();**

**4. ES6 Classes  
class PatientClass {  
 constructor(name. addres) {  
 this.name = “”,  
 this.address= “”  
}  
var p = new PatientClass();**

**}**

**Question 28:- How can we do inheritance in JavaScript ?**

**Question 29:- What is prototype in JavaScript ?**

**Question 30:- Explain Prototype chaining ?**

**Ans: - JavaScript uses object inheritance or prototypical inheritance. Inheritance is done using Prototype Object.  
  
function Employee() {  
 this.name = “”,  
 this.doWork = function() {  
 alert(“Basic Work”)  
 }  
 this.Attendance = function() {}  
}**

**function Manager() {  
 this.cabin = “”;  
 this.doWork = function() {alert(Manages Team)}  
 }  
  
var emp = new Employee();  
 Manager.prototype = emp;  
  
var man = new Manager();  
man.name = “Kishan”;**

**man.Attendance();**

**man.doWork();**

**Every JavaScript object has prototype object,  
It’s an inbuild object provided by JavaScript**

**Prototype Chaining is a process where the property/method are first check in current object, if not found then it checks in the prototype object  
  
if does not found then in that it try checking the prototypes object, until he get the prototype object null**

**Question 31:- What is Let Keyword?**

**It was introduce in ES6 and “Let Help to create immediate block level local scope”**

**Question 32:- Are Let variables hoisted ?**

**Yes, they are hoisted but not initialize with the value so, if you try to access the variable you will get an uninitialized error**

**Question 33:- Explain Temporal Dead Zone ?**

**TDZ it’s a period or it’s a state of variable where variables are named in memory but they are not initialized with any value**

**Question 34:- Let vs Var**

**Scope :   
var : Scope to the immediate function body**

**Let ; Scope to the immediate enclosing block.**

**Intialized value - Value initialized with undefined  
Value initialized with nothing.**

**Shallow copy basically means when we copy an object to another object, one object hold reference to another object this is called shallow copy**

**Question 35:- String Concatenation and Arithmetic puzzle**

**Question 36 :- What is class in ES6 ?**

**Question 37 :- So with class Keyword does it imply JavaScript is a OOP language ?**

**Question 38 :- Differentiate between class and normal function ?**

**Question 39 :- What is a Arrow function ?**

**Question 40 :- Why do we use Arrow function ?**

**Question 41 :- Differentiate between Arrow vs Normal Function ?**

**Question 42 :- Does Arrow function create its own this ?**

**Question 43. Explain Synchronous execution ?**

**Question 44. What is a call Stack ?**

**Question 45. What is a blocking call ?**

**Question 46. How to avoid blocking calls ?**

**Question 47. Explain Asynchronous execution ?**

**Question 48. Synch vs Asynch ?**

**Question 49. How can we do Asynch calls ?**

**Question 50. What is a thread ?**

**Question 51. Explain Multi-threading ?**

**Question 52. Is JavaScript Multi-threaded ?**

**Question 53. Then how does Settimeout run ?**

**Question 54. What is a WebAPI/Browser API ?**

**Question 55. What is a Event loop and callback queue?**

**Question 56. Eventloop and Callback code question**