**Javascript Quick Questions**

**Basics of the Javscript**

1. **What are different types of variables in javascript?**

* We have three variables in the javascript -> **var,let,const**

1. **What is the difference between the var , let and const ?**

* **Var🡪** This is the old version of variables declaration . var promotes
  + Var has **hoisting**
  + The var is the function scope variable
  + The var value if declared on the global scope will store the variable in the global context window

`🡪 **Let and const 🡪** These are the newer version of the variables which are declared in the ECMA 6 guidelines

* + Let and Const doesn’t have hoisting
  + These are the block scoped variables
  + Const is used to declare the constant variables
  + The Let and const has the temporal dead zone

1. **What is the temporal dead zone ?**

* **Temporal dead zone** -> Temporal dead zone is the region where the let and const variables are inaccessible. Before declaration we can’t access the variables unlike the var type of variables

See the below example to see the temporal dead zone

**{**

// TDZ starts here (at the beginning of this block’s local scope)

// bestFood’s TDZ continues here

// bestFood’s TDZ continues here

// bestFood’s TDZ continues here

// bestFood’s TDZ continues here

// bestFood’s TDZ continues here

// bestFood’s TDZ continues here

**let bestFood = "Vegetable Fried Rice"; // bestFood’s TDZ ends here**

**console.log(bestFood); // returns "Vegetable Fried Rice" because bestFood’s TDZ does not exist here**

// bestFood’s TDZ does not exist here

// bestFood’s TDZ does not exist here

**}**

1. **What are the difference between the == and === (strict operator)**?

* == is type of the comparison operator in javascript which will implicitly type cast comparison of two variables
* === infers to strict variable comparison which the two comparison variables are not type casted.

1. **What are the null and undefined variables in javascript?**

* The undefined variables – are data type which the javascript compiler automatically provides to the variables which are not being assigned.
* The null variables – are data type which the user sets for the values to keep empty placeholder for the variables

1. **What are lambda function in javascript ?**

* The lambda or anonymous functions are the short hand version of describing the functions where only the arguments and logic of the functions needs to be defined . In some cases the we don’t need to also define the return type as it can be inferred if the logic is of single line only .

**Example of the lambda function:-**

**(args)=> logic or (args)=>{**

**//logic**

**}**

1. **What are the function scoped and block scoped variables?**

* **Function scoped ->** The function scoped variable are those types of variable which are scoped which are available is throught the whole function body . **var is the function scoped variable**
* **Block Scoped 🡪** The block scoped variables concept is introduced in the ECMA6 where the **let and const variables which are only scoped within the block like if , else or for blocks**

See the below example for the block scoped and function scoped variables

// Function scoped Variable Definition

function abc(){

    var a = 10

    function def(){

        console.log(a) *// a=10 will be printed function scoped only*

    }

}

console.log(a) *// a is not defined*

*// Block Scoped Variable*

*if(10>8){*

*const d=20 // only available in the block scoped*

*}*

*Console.log(d)*

1. **What are the type of functions are present in the javascript?**

* There are the three syntax to define the functions in the javascript
  + **Primitive function declaration with function keyword and name of the function**:-
    - function abc(a,b){

**return a+b;**

}

* + **Anonymous function declaration with function keyword only and no name generally used for iife**
    - function (){

**return a+b;**

}

* + **Lambda function or arrow function where only arguments and return logic should be used**

(args)=> logic or (args)=>{

//logic

}

1. **What is destructuring in javascript object and array ?**

* The destructuring is the concept or syntax where we can access the elements of the object and array without the looping mechanism we can directly access the elements with the following syntax
* *// Destructing for arrays*
* const arr = [1,2,3,4,5]
* const [a,b]=arr *// a=1, b=2*
* *// Destructuring in objects*
* const obj = {name:'John',age:25,city:'New York'}
* const {name,age} = obj *// name='John', age=25 in objects we can destructure only through keys names only.*

1. **What are the spread and rest operators in javascript?**

* The spread and the rest operators in the javascript are used to merge and converge the arrays and object into single array or object . or can make another object or arrays also
* One Layman language difference between the spread and rest operators are both the operators are quite similar to each other , the difference could be found it spread operator are generally used on the left hand side of the assignment and the rest is generally used on the right hand side of the assignment.

1. **Show the use of the spread operator on the arrays with concatenation of arrays using spread operator?**

🡪

*//  Spread operator in arrays*

const arr1=[1,2,3]

console.log(...arr1) *// 1 2 3 spreading the array elements*

const arr2 = [4,5,6]

const arr3 = [...arr2] *// 4 5 6 copying the array elements to another array*

const arr4 = [...arr1,...arr2] *// 1 2 3 4 5 6 combining/concatenating the arrays*

1. **Show the use of the spread operators on the objects with merging of two objects?**

🡪

*// spread operators in objects*

const obj1 = {name:"Rishabh",age:24}

const obj2 = {...obj1} *// copying one object into another using the spread operators*

const obj3 = {school:'JMS',college:'UPES'}

const obj4 = {...obj1,...obj3} *// merging two object into one new object using the spread operators*

1. **Different Types of for Loops we can used in javascript?**

* In the javascript broadly there are four types of for Loops which are
  + **Basic for Loop :** for(let i=0 ; i<10;i++){}
  + For of Loop in Array :- for (let item of items){}
  + For in loop of Object:- for(let item in items){}
  + ForEach array prototype function.:Array.forEach()

1. **What is the closure in javascript ? (One of the Most Important)**

* Closure is the property in the javascript where the function can access the variables and arguments of it parents function even when the function execution of the parent function is completed.
* This is due to the part where the returning function internally gets attached object in **[[Scopes]] with the name of the closure and it contains access to the variables and arguments.**
* *// closures*
* function abc(){
* const a=10
* return function(){
* console.log(a)
* }
* }
* const callBackFunc=abc()
* callBackFunc(); *// 10*

1. **What are firstClassFunctions , higherOrderFunction and IIFE Functions?**

* **FirstClassFunctions->**The first Class Function is considered to simple functions in the paradigm where the functions are considered to be variables .which can be stored in the variables , pass as arguments to another functions and also returned from the functions
* **HigherOrderFunctions🡪** The Higher Order functions are those types of functions which either takes another functions as argument or returns another function as an argument.

**Syntax ->** function abc(func){

**return func**

}

* **IIFE Functions 🡪** The iife functions are other functions which are declared and invoked immediately and simulatenoulsy . The IIFE functions are generally used in the cdn js library which needs to inject the functions and functionality as soon as it is injected.

**Syntax 🡪 (**function (){

**// any logic**

Console.log(‘abc’)

)()

1. **What is short circuiting in javascript ?**

* Short circuiting in javascript . The short circuiting is the methodology in javascript where results of the || and && operators
  + See the Below Examples
    - For || operator:- It Returns first true and last False
    - For && operator : It Returns first False and last True

1. **What are the datatypes in javascript ?**

**🡪**Number , Boolean , string, Object , null ,undefined, BigInt,Symbol

1. **What is the null coalescing Operator ?**

**🡪** Null coalescing operator (??) is upgradation of the (||) operator where the only **null and undefined values** will be considered as the falsy values else all the values are considered to truthy values

**Arrays Questions**

1. **What is the data type of the array function?**

* The array is the type of object
  + typeof(Array)==object

1. **What function helps to check whether its array or not ?**

* The Array.isArray(arr) function is used to check whether the passed object is array or not.

1. **What is the difference between the splice and slice operator?**

* Both splice and slice operators are used to modify the elements from the existing array but they have difference in both of them
  + **Slice :-** the Slice returns the new array with the elements from existing array from start index to endindex-1
    - **It does’nt modify the existing array**
    - **It returns the new array with elements which has been sliced**

Syntax :- newArray=Array.slice(startIndex,index)

* + **Splice :-**  the splice is the function which modifies the existing array and deletes the elements from the start index to the count we provide , its also takes the third arguments which can add the new elements in the place from the start index.
    - **Its Does modify the original array**
    - **It takes two mandatory arguments rest it not mandatory , non mandatory arguments are those arguments which will products which need to be added**
* *// Difference between the slice and splice operators*
* const sliceArr =[1,2,3,4,5,6]
* const slicedArr=sliceArr.slice(1,3)*// returns new Array [2,3]*
* console.log(sliceArr) *// same Array [1,2,3,4,5,6]*
* *// splice*
* const spliceArr=[1,2,3,4,5,6]
* const deletedElemt=spliceArr.splice(1,4) *// return the deleted Element*
* console.log(spliceArr) *// [1,6]*

1. **What are the functions which will insert the elements into the array ?**

* **Push –** this function will insert the element at the last of the array
* **Unshift –** This function will insert the elements into the first place of the array

1. **What are the function which will remove the elements from the array ?**

* **Pop –** This function will remove the last element of the array
* **Shift –** This function will remove the first element of the array

1. **What are some searchFunction in arrays ?**

* **indexOf(element)-**This function will return the **first index of the instance** of the element present in the array . **If the no element is found then (-1) will be returned**
* **includes(element)-** This function will return boolean value if the element is present or not present
* **lastIndexOf(element)-** This function will return the last index of the element present in the array
* **firstIndexof(element) –** This function will return the first Index of the element present in the array .

1. **What are the string related method in the array?**

* The string related method in arrays are:
  + **Join :** This method joins the array of string to a string using the separator
* const stringArr=['a','b','c']
* const joinedStr = stringArr.join('.')
* console.log(joinedStr)*// a.b.c*
* **toString():** This method will convert the array to string
* *// TO string*
* const stringArr=['a','b','c']
* const joinedStr = stringArr.toString()
* console.log('joined String',joinedStr) *// Joins the element with comma seperator*

**This , Call , Apply and Bind (Most Most Important)**

1. **What is the this keyword in the javascript?**

* The this keyword is the most hot topic and most important concept in javascript . The this keyword in javascript works differently as compare to other programming languages .

The this keyword **refers to the context where the piece of code is in lexical scope .**

The value of this keyword differs from function to function lets understand the each concept of this keyword in various places .

* + **Value of this keyword in global statement function when use-strict is not their 🡪** In this scenario the value of this inside the function refers to **global window object** in case of browser

function def(){

    console.log(this) *// window object*

}

* + **Value of this keyword in global statement function when use Strict is their -🡪** In this scenario the value of this keyword in the **global statement function will be undefined**

‘use strict’

function def(){

    console.log(this) *// undefined*

}

* + **(Most Most Important)Value of this keyword for the anonymous function inside the Object ( Most Important):🡪** when we use anonymous function inside the object as store it as the object key **, the value of this keyword will be the object itself .References to the object**

*// This keyword of the anonymous function inside the object*

const thisObj={

    name:"Rishabh",

    age:"20",

    anon:function(){

        console.log(this) *// {name:"Rishabh",age:"20",anon:func}*

    }

}

thisObj.anon()

1. **(Most Most Important) This keyword and Arrow Function Relations?**

* Arrow functions doesn’t have their own this keyword . **They inherit the value of this keyword of parent environment where there are lexically scoped .**

Lets see the below example to understand it better and how it works in different environments

* + **Arrow function this keyword inside the object method**
    - When this keyword is used as the property of the object . Then the this keyword will reference to parent environment . for global object it will print the window object

*// This keyword of the anonymous function inside the object*

const thisObj={

    name:"Rishabh",

    age:"20",

    anon:()=>{

        console.log(this) *// window object will be printed*

    }

}

thisObj.anon()

* + **Arrow function this keyword inside the anonymous function inside the object** 
    - When arrow function resides inside the anonymous function which is the property object , **then it will reference to the parent environment this which is the this of the anonymous function . Hence the object will be referenced**
* *// This keyword of the anonymous function inside the object*
* const thisObj={
* name:"Rishabh",
* age:"20",
* anon:function(){
* const def=()=>{
* console.log(this) *// {name:"Rishabh",age:"20",anon:func}*
* }
* def();
* }
* }
* thisObj.anon()

**Internals of Javascript :- Call Stack , Execution Context , MicroTask Queue,Macrotask Queue**

1. **What is the main pillars of the javascript internals ?**

**🡪**The three main pillars of the javascript internals are

* + **Global and Local Memory:** It’s the region where the variables memory and function bodies are stored which are referenced during the execution phase
  + **Call Stack :** The Javascript has only single call stack which follows FILO process of stack and each function enters the call stack and after execution leaves the call stack
  + **Execution Context :**For each and every function call new execution context is created and each execution scopes its local memory , this keyword and reference to outer variables
  + **MacroTask Queuue and Microtask Queue:** The macro and micro task queue holds the async functions execution in queue format first in first out and micro task queue has more priority than microtask queue , Generally the callback functions are stored in microtask queue

1. **What is the event loop in the javascript ?**

* The event loop in the javascript is the constant check process which is being runned for checking and execution of the micro and microtask queue functions if the call stack is empty. So keep on checking whether the call stack is empty or not . Constant loop

1. **Where Did the SetTimeout function is being thrown when executed?**

* The Set Timeout function is thrown in the macro task queue.

**Asynchronous Javascript:Promises,Async- Await , Promise Chaining**

1. **What are the promises in javascript ?**

* The promises in javascript is an object which kindoff handles delayed functions which has three states either **Pending , Fullfilled or Rejected ,** For each state completion we can assign a callback function which will be automatically called when the particular state is reached

1. **(Most Most Important) What are the properties of promises?**

* A promise object has important properties assigned to it
  + **[[PromiseState]]-** This property holds the value of the promise current state , it can have three values **Pending , FulFilled and Rejected**
  + **[[PromiseResult]]-**This property holds the data or result which is emitted when the promise is fulfilled or rejected
  + **[[PromiseFullFillmentReactions]]-**This property holds the callback function which will be executed automatically when the promise is fulfilled and the promise result will be automatically passed as the first argument
  + **[[PromiseRejectionReactions]]-**This property holds the callback function which will be executed automatically when the promise is rejected and the promise rejection result will be automatically passed as the first argument.

1. **(Most Most Important) Syntax to declare the promise using the constructor function?**

*// Promises*

const promise = new Promise((resolve,reject)=>{

    resolve(2)

}).then((data)=>{console.log(data)}).catch((rejection)=>console.log(rejection))

Above syntax is to show to declare the promise object using the constructor function with new keyword

* (Most Most Important)Some Insights of the above syntax first one is **Promise Object will take one callback function which has two argument resolve and reject and both are function itself these function will set the PromiseResults and If the Resolved is called the PromiseState will set to fulfilled and reject is Called then PromiseState will set to rejected**
* Then then and catch call back function is being called based on the promise state.

34 ) **What is Promise Chaining ?**

* To be learnt

**Async Await**

1. **What are async functions in javascript and what are the return type of the async function**

* The async in javascript is special keyword which when used with function wraps the function return type to a promise .
  + **The async function always returns a promise if returning value is not a promise it wraps it into the promise**

1. **Show me the syntax to declare the async function?**

**🡪**Below is the syntax to declare the async function

*// async function*

async function abc(){

    return 2

}

const asyncPromise = abc() *// the async function will return a promise object*

*//  To access the Results we need to call a thenable function on it*

asyncPromise.then((data)=>console.log(data)) *// 2*

1. **Using the await with the async function (Most Important Concept)?**

* As we know that Promises are non blocking in nature . The both **async and await** is used to handle promises and kinda make them synchronous in nature.
  + The await keyword can only be used with async function
  + (**Most Important Part the await keyword resolves the promise and return the resolved value see the example below)**
* *// some promise*
* const p = new Promise((resolve,reject)=>{
* setTimeout(()=>{
* resolve(2)
* },10000)
* })
* async function def(){
* const data = await p *// output 2 , here the data value will be resolved value which is 2*
* }
* def()

1. **If the await keyword returns resolved value how to handle error when using the await ?(Most Important)**

* Yes , it is true the await keyword used with promise will always return the resolved value .

We will use the **try and catch block to handle the** reject state , because internally when the promise is rejected , it throws a Error ( with message) and as function is wrapped with try and catch will be handled in catch block

See the below Example

const p2 = new Promise((resolve,reject)=>{

    reject('Some Error Occurred')

})

async function ghi(){

    try {

        const data = await p2

    }

    catch(error){

        console.log(error) *// The reject value will be printed here*

    }

}

ghi();

1. **How Fetch Works with async and await ?**

* To understand the syntax each every feature of the fetch see the below syntax with the explanation
* *// How Fetch Works*
* async function jkl(){
* const fetchResponse = await fetch('API') *// fetch resolves into a promise so fetchResponse will be promise*
* const data = await fetchResponse.json() *// now fetchResponse is readable stream to convert we need json function will also a promise and await resolves a promise*
* }

This is how the fetch Works will async await

1. **Do the await holds the main thread of execution as the function is stopped ?**

* No , The await don’t holds the main thread of execution
  + Instead when the await hits the execution phase , the javascript engine suspends the function from removing it into the call stack and after the promise is resolved or rejected then the function re-enters the call stack and then the execution start from next line of await keyword**(!!Most Most Important concept)**