**JAVASCRIPT INTERVIEW QUESTIONS**

1.What is hoisting?

Ans. Hoisting refers to the process whereby the interpreter appears to move the declarations of functions, variables, or classes to the top of their scope, prior to the execution of the code.

2.What is scoping?

Ans. Scoping in JavaScript refers to the current context of code, which determines the accessibility of variables to JavaScript. The scoping is of two types: global scoping and local scoping. Global scoping are those declared outside a block and local scoping are those declared inside a block.

3.How are var, let , const different?

Ans. Var declarations are globally scoped or function scoped , let and const are block scoped.   
Var can be redeclared or updated within its scope. Let variable can be updated but cannot be redeclared, const can neither be redeclared or updated. They are all hoisted to the top of their scope.

4.What are the two differences in the arrow functions?

Ans. Regular functions: Object inside the regular functions contains a list of arguments.

They are constructible and callable, they can be called using the new keyword.

Arrow functions: They on the opposite side doesn’t define arguments, i.e they do not have arguments binding. But one can easily access the arrow function arguments using the rest parameter …args.

They are callable but not constructable i.e arrow functions can never be used as constructor functions.

5.Does call apply bind work for arrow functions?

Ans. They don’t work for arrow functions as arrow functions doesn’t have their own this, the call() and apply() can only pass in parameters, thisArg is ignored.

6.What does call apply bind do?

Ans. Call() and apply() are similar methods. They both execute the bound function on the object immediately. The bind() method does not execute the function right away. Instead it creates and returns a bound function which can be executed later.

7.What are closures?

Ans. Closures is when a function is defined inside another function, that inner function has access to all the variables outside of that function , even if the outside function finishes executing and those variables are no longer accessible outside of that function.

8. Write a program to debounce a searchbar?

Ans. <!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <link rel="stylesheet" href="index.css">

    <title>Document</title>

</head>

<body>

    <div id="container">

        <input id="food" type="text" placeholder="search for food">

        <div id="searchbox"></div>

        <p id="showcase"></p>

    </div>

</body>

</html>

<script>

    var searchbox=document.querySelector("#searchbox")

    let food\_wait;

    var showcase=document.querySelector("#showcase")

    async function searchfood(){

        try{

            var food=document.querySelector("#food").value

            let response = await fetch(`https://themealdb.com/api/json/v1/1/search.php?s=${food}`)

            var data=await response.json()

            var food\_arr=data.meals;

            appendFood(food\_arr)

            console.log(data)

            console.log("hello")

        }

        catch(err){

            console.log("err",err)

        }

    }

    function appendFood(food){

        searchbox.innerHTML=null;

        if(food==undefined){

            return false;

        }

        food.forEach(function(element){

            var p=document.createElement('p')

            p.innerText=element.strMeal

            showcase.innerHTML=""

            p.onclick=function(event){

                searchbox.innerHTML=null

                showcase.append(p)

                console.log(p)

            }

            searchbox.append(p)

        })

    }

    document.querySelector("#food").addEventListener("input",debounce)

    function debounce(func,delay){

        if(food\_wait){

            clearInterval(food\_wait)

            console.log("hello")

        }

        food\_wait=setTimeout(function(){

            searchfood()

        },1000)

    }

</script>

9.Write a program to throttle a search bar?

Ans.

10. Create a custom method for an array called MyApp, use prototype chain to achieve this.

Ans.

11.What is event bubbling?

Ans. Event bubbling is a type of event propagation where the event first triggers on the innermost target element and then successively triggers on the ancestors(parents) of the target element in the same nesting hierarchy till it reaches the outermost DOM element or the document object.

12.Explain promises to a 5 year old, with some examples?

Ans. We can take a story for this as a reference, Jack and Jill went on a hill, to fetch a pail of water, and lets bring a twist to this poem which includes their grandparents. Now jack and Jill have promised their grandparents that they will fetch the water. Now what happens that Jack fell down and broke his crown. And Jill came tumbling after.

Now we have two options :

1. Jack’s crown didn’t fall and they both fetched the water and resolved the promise.
2. Jack’s crown fell down and due to this disaster, the promise got rejected.

But note one thing that when the kids were fetching water/executing a function then grandparents were not sitting idle, they were planning for the day.

The JavaScript promises work similarly, as developers we create them to fetch something(data from data store, configurations and many more). Usually the fetching may not happen instantly. We want to fetch things asynchronously. It means we do not want the applications to wait for the response, but we can continue to work on the response when its available.

13. Write a function called sleep that will return a promise, if you do not provide a number to the function, then it will return an error and go to the catch block?

Ans. const sleep = new Promise ((resolve,reject) =>{

try{

console.log(`Sleep for ${mili} seconds`)

}.catch(err){

console.log(err) // look at it afterwards

}

})

14. What does async-await mean?

Ans. Async: It simply allows us to write promises-based code as if it was synchronous and it checks that we are not breaking the execution thread. It operates asynchronously via the event loop. Async functions will always return a value.

Await: Await function is used to wait for the promise. It could be used within the async block only. It makes the code wait until the promise returns a result. It only makes the async block wait.

15. What does this keyword mean in JavaScript?

Ans. This keyword refers to an object that is executing a current piece of code.

16. What are classes in JavaScript? What are getters and setters in JavaScript?

Ans. Classes in JavaScript determine what the properties of the object are and how it behaves. Classes are not directly used by programs but they create objects that programs use. JavaScript classes introduced in ECMAScript, 2015 are syntactic sugar over JavaScript’s existing prototype-based-inheritance.

Getters and Setters are used to protect your data, particularly when creating classes. They are code constructs that help developers access the properties of objects in a secure way. With getters you can access(“get”) the values of properties from external code, with setters you can change (“set”) their values.

17. How do you declare private and static variables in JavaScript?

Ans. Private variables are declared by # names and static variables can be declared by using static keyword.

18. Create a calculator class, it should be able to add, reduce multiply and divide. it should have a value getter, and that should return final output. keep the history of changes made as well, and keep that private, and a user should be able to see previous changes made to the value?

Ans.

19. What is currying?

Ans. Currying is when a function, instead of taking all the arguments at one time, takes the first one and returns a new one, takes the second and returns a new one, third and new until all arguments are completed. Currying allows us the easily get partial.

20. Write a program to flatten an array?

// input: [ 1, [ 2, 3 ], [ 3 ], [ [ [ 5]], 6] ]

// output => [ 1, 2, 3, 3, 5, 6 ]

const flattenAnArray =(input)=>{

let newArray=[]

input.forEach((item) => {

if(input.isArray(item))

newArray.push(…flattenAnArray(item));

else

newArray.push(item);

})

return newArray;

}

Console.log(flattenAnArray([1,[2,3],[3],[[[5]],6]]))