JavaScript Interview Questions

Saikrishna Nangunuri | SDE2 @HCCI |
https://www.linkedin.com/in/saikrishnanangunuri/

1. What is the difference between 'Pass by Value' and 'Pass by Reference'?

In JavaScript, whenever a function is called, the arguments can be passed in two ways, either pass by value or pass by reference.

- Primitive datatypes such as string, number,boolean,null and undefined are passed by value.
- Non -primitive datatypes such as object,arrays or functions are passed by reference.

In Pass by value, parameters passed as an arguments creates their own copy. So any changes made inside the function are made to the copied value so it will not affect the original value.

In Pass by reference, parameters passed as an arguments does not creates their own copy. so any changes made inside the function will affect the original value.

---

2. What is the difference between map and filter ?

- Both map and filter are useful in JavaScript when working with an arrays.
- map transforms each element of an array and creates a new array which contains the transformed elements. whereas filter will creates a new array with only those elements which satisfies the specified condition.

---

3. What is the difference between map() and forEach()

- map method is used to transform the elements of an array. Whereas forEach method is used to loop through the elements of an array.
- map method will return a new array with the transformed values. forEach method does not return a new array.
- map method can be used with other array methods like filter method. whereas forEach method cannot be used with other array methods as it does not return any array.

---

4. What is the difference between Pure and Impure functions?

Pure Functions:

- Pure functions are the functions which will return same output for same arguments passed to the function.
- This will not have any side effects.
- It does not modify any non local state.

```
function greeting(name) {
  return `Hello ${name}`;
}
console.log(greeting("Saikrishna Nangunuri"));
```

Impure Functions:

- Impure functions are the functions which will return inconsistent output for same arguments passed to the function.
- This will have side effects.
- This will modify non local state.

```
let message = "good morning";
function greeting1(name) {
  return `Hello ${name} , ${message}`;
}
console.log(greeting1("Saikrishna Nangunuri"));
```

Ref: https://www.scaler.com/topics/pure-function-in-javascript/

---

5. What is the difference between for-in and for-of ?

Both for-in and for-of are used to iterate over the datastructure.

for-in:

- for-in iterates over the enumerable property keys of an object.

for-of:

- for-of is used to iterate over the values of an iterable object.
- Examples of iterable objects are array,string,nodelists etc. (for of on object returns error)

https://stackoverflow.com/questions/29285897/difference-between-for-in-and-for-of-statements?answertab=scoredesc#tab-top

---

6. What are the differences between call(), apply() and bind() ?

- Call method will invokes the function immediately with the given this value and allow us to pass the arguments one by one with comma separator.
- Apply method will invokes the function immediately with given this value and allow us to pass the arguments as an array.
- Bind method will return a new function with the given this value and arguments which can be invoked later.

Call, Apply and Bind

---

7. List out some key features of ES6 ?

1. Let and Const declarations.
2. Arrow functions
3. Template literals
4. Destructuring assignment
5. Spread and Rest operators
6. Default parameters
7. Promises
8. Modules
9. Map, Set, Weakmap, Weakset
10. Classes

---

8. What's the spread operator in javascript ?

Spread operator is used to spread or expand the elements of an iterable like array or string into individual elements.

Uses:

1. Concatenating arrays.

```
let x = [1,2];
let y = [3,4];

let z = [...x,...y]    ⇒⇒ 1,2,3,4
```

2. Copying arrays or objects.

```
let a = [...x] // 1,2
```

3. Passing array of values as individual arguments to a function.

```
function createExample(arg1,arg2){
  console.log(arg1,arg2);
}

createExample(...a)
```

---

9. What is rest operator in javascript ?

Rest operator is used to condense multiple elements into single array or object.

This is useful when we dont know how many parameters a function may receive and you want to capture all of them as an array.

```
function Example(...args){
    console.log(args)
}

Example(1,2,3,4);
```

---

10. What are DRY, KISS, YAGNI, SOLID Principles ?

- **DRY:** Do not repeat yourself.
  - Avoid duplicates. This make software more maintainable and less error-prone.
- **KISS:** Keep it simple stupid.
  - Keep the software design and implementation as simple as possible. This make software more testable, understandable and maintainable.

- **YAGNI:** You are not going to need it.
  - Avoid adding unnecessary features/functionalities to the software. This makes software focussed on essential requirements and makes it more maintainable.
- **SOLID:**
  - **O - Open/Closed principle:** Classes must be open for extension and closed to modification. This way we can stop ourselves from modifying the existing code and causing potential bugs.
  - **S - Single responsibility:** means each class should have one job or responsibility.
  - **L - Liskov Substitution:** If class A is subtype of class B then classB should be able to replace classA with out disrupting the behaviour of our program.
  - **I - Interface segregation:** Larger interfaces must be split into smaller ones.
  - **D - Dependency inversion:** High level modules should not depend on low level modules. Both should depend on abstraction.

---

11. What is temporal dead zone ?

- It is a specific time period in the execution of javascript code where the variables declared with let and const exists but cannot be accessed until the value is assigned.
- Any attempt to access them result in reference errors.

---

12. Different ways to create object in javascript ?

https://www.scaler.com/topics/objects-in-javascript/

- **Object literal :**

```
let userDetails = {
    name: "Saikrishna",
    city: "Hyderabad"
}
```

- **Object constructor :**

```
let userDetails = new Object();
userDetails.name = "Saikrishna";
userDetails.city = "Hyderabad";
```

- **Object.Create() :**

  This is used when we want to inherit properties from an existing object while creating a new object.

```
let animal = {
 name: "Animal name"
}

let cat = Object.create(animal);
```

- **Object.assign() :**

  This is used when we want to include properties from multiple other objects into new object we are creating.

```
let lesson = {
 lessonName: "Data structures"
};

let teacher= {
    teacher: "Saikrishna"
};

let course = Object.assign({},lesson,teacher);
```

13. Whats the difference between Object.keys,values and entries

- **Object.keys():** This will return the array of keys
- **Object.values():** This will return the array of values
- **Object.entries():** This will return array of [key,value] pairs.

```
let data = {
 name: "Sai",
 lang: "English"
};

Object.keys(data)  // ["name","lang"]
Object.values(data) // ["Sai","english"]
Object.entries(data) // [["name","Sai"],["lang","English"]]
```

---

14. Whats the difference between Object.freeze() vs Object.seal()

- **Object.freeze:**
  - Will make the object immutable (prevents the addition of new propeties and prevents modification of existing properties)

```
let data = {
    a : 10
};

Object.freeze(data);
data.a= 20;
data.c = 30;

console.log(data)
```

```
output: {
a: 10
}
```

- **Object.Seal():**
  - Will prevent the addition of new properties but we can modify existing properties.

```
let data = {
    a : 10
};

Object.seal(data);
data.a = 20;
data.c = 30;

console.log(data)

Output:
data: {
 a: 20
}
```

---

15. What is a polyfill in javascript ?

- A polyfill is a piece of code which provides the modern functionality to the older browsers that does not natively support it.
- **Polyfill for foreach:**

```
Array.prototype.forEach = function(callback) {
  for (var i = 0; i < this.length; i++) {
    if (i in array) {
      callback.call(array[i], i, array);
```

```
      }
    }
  };


let array = [1,2,3,4,5];

array.forEach((element,id,arrd)=>{
  console.log(`${element},${id}`,arrd)
})
```

---

16. What is generator function in javascript ?

- A generator function is a function which can be paused and resumed at any point during execution.
- They are defined by using function* and it contains one or more yield expressions.
- The main method of generator is next(). when called, it runs the execution until the nearest yield.
- It returns an object which contains 2 properties. i.e., done and value.
  - **done:** the yielded value
  - **value:** true if function code has finished. else false.
- https://javascript.info/generators

```
function* generatorFunction() {
    yield 1;
    yield 2;
    yield 3;
    return 4
}

const generator = generatorFunction();
```

```
console.log(generator.next()); // Output: { value: 1, done: f
console.log(generator.next()); // Output: { value: 2, done: f
console.log(generator.next()); // Output: { value: 3, done: f
console.log(generator.next()); // Output: { value: 4, done: t
```

---

17. What is prototype in javascript ?

- If we want to add properties at later stage to a function which can be accessible across all the instances. Then we will be using prototype.
- https://www.tutorialsteacher.com/javascript/prototype-in-javascript

```
function Student(){
        this.name = "Saikrishna",
        this.exp= "8"
}

Student.prototype.company = "Hexagon"

let std1 = new Student();
std1.exp = "9"

let std2 = new Student();
std2.exp = "10"

console.log(std1);
console.log(std2)
```

---

18. What is IIFE ?

- IIFE means immediately invoked function expression.
- functions which are executed immediately once they are mounted to the stack is called iife.

- They does not require any explicit call to invoke the function.
- https://www.geeksforgeeks.org/immediately-invoked-function-expressions-iife-in-javascript/
- https://www.tutorialsteacher.com/javascript/immediately-invoked-function-expression-iife

```
(function(){
  console.log("2222")
})()
```

Arrow functions: https://www.codingninjas.com/studio/library/difference-between-arrow-function-and-normal-function

---

19. What is CORS ?

- CORS means cross origin resource sharing.
- It is a security feature that allows the webapplications from one domain to request the resources like Api's/scripts from another domain.
- cors works by adding specific http headers to control which origins have access to the resources and under what conditions.

---

20. What are the different datatypes in javascript ?

- **Primitive datatypes:**
  - String
  - number
  - boolean

    - bull
    - undefined
    - Bigint
    - symbol
- **Non-Primitive datatypes:**
    - Object
    - Array
    - Date

---

21. What are the difference between typescript and javascript ?

- Typescript points out the compilation errors at the time of development. Because of this, getting runtime errors is less likely.
- Typescript supports interfaces whereas javascript does not.
- Typescript is better suited for large scale applications where as javascript is suited for small scale applications.
- Typescript is the superset of javascript and has all the object oriented features.
- Functions have optional parameters in typescript whereas in javascript does not have it.
- Typescript takes longer time to compile code.

---

22. What is authentication vs authorization ?

- **Authentication:**
    - Its the process of verifying who the user is.
- **Authorization:**

- Its the process of verifying what they have access to. What files and data user has access to.

---

23. Difference between null and undefined ?

- **Null:**
  - If we assign null to a variable, it means it will not have any value
- **Undefined:**
  - means the variable has been declared but not assigned any value yet.

---

24. What is the output of 3+2+"7" ?

- 57

---

25. Slice vs Splice in javascript ?

- **Slice:**
  - If we want to create an array that is subset of existing array with out changing the original array, then we will use slice.

```
let arr = [1,2,3,4];
let newArr = arr.slice(1,3);

console.log(newArr) // [2,3]
```

- **Splice:**
  - If we want to add/delete/replace the existing elements in the array, then we will use splice.

```
let arr = [1,2,3,4,5,0,10];
let newArr = arr.splice(2,4,8,9,6);
// splice(startIndex,numberOfItemsToRemove,replaceElements

console.log(arr); //  [1,28,9,6,10]
console.log(newArr); // [3,4,5,0]
```

---

26. What is destructuring ?

- It is introduced in Es6.
- It allows us to assign the object properties and array values to distinct variables.

```
const user = {
    "age": 10,
    "name": "Saikrishna"
}

const {age,name} = user;
console.log(age,name) // 10,"Saikrishna"
```

```
const [a,b] = [1,2];
console.log(a,b) // 1,2
```

---

27. What is setTimeOut in javascript ?

- setTimeOut is used to call a function or evaluate an expression after a specific number of milliseconds.

```
setTimeOut(function(){
 console.log("Prints Hello after 2 seconds")
},2000);

// Logs message after 2 seconds
```

---

28. What is setInterval in javascript ?

- setInterval method is used to call a function or evaluate an expression at specific intervals.

```
setInterval(function(){
        console.log("Prints Hello after every 2 seconds");
},2000);
```

---

29. What are Promises in javascript ?

- Promise is an object which represents the eventual completion or failure of an asynchronous operation in javascript.
- At any point of time, promise will be in any of these below states.,
  - **Fulfilled:** Action related to promise is succeded.
  - **Rejected:** Action related to the promise is failed.
  - **Pending:** Promise is neither fulfilled nor rejected
  - **Settled:** Promise has been fulfilled or rejected.
- Promise can be consumed by registering the functions using .then() and .catch() methods.
- **Promise constructor:** will take one argument which is a callback function. This callback function takes 2 arguments resolve and reject.

- If performed operations inside callback function wents well then we will call resolve() and if does not go well then we will call reject()

```
let promise = new Promise(function(resolve,reject){
         const x = "Saikrishna";
         const y = "Saikrishna";

         if(x === y){
             resolve("Valid")
         } else{
             let err = new Error("Invalid")
             reject(err)
         }
})

promise.then((response)=>{
        console.log("success",response)
}).catch((err)=>{
        console.log("failed",err)
})
```

---

30. What is a callstack in javascript ?

- Callstack will maintain the order of execution of execution contexts.

---

31. What is a closure ?

- **Defination:** A function along with its outer environment together forms a closure (or) Closure is a combination of a function along with its lexical scope bundled together.
- Each and every function in javascript has access to its outer lexical environment means access to the variables and functions present in the environments of its parents

- Even when this function is executed in some outer scope(not in original scope) it still remembers the outer lexical environment where it was originally present in the code.

```
function Outer(){
        var a = 10;
        function Inner(){
            console.log(a);
        }
        return Inner;
}

var Close = Outer();
Close();
```

---

32. What are callbacks in javascript ?

- A callback is a function which is passed as an argument to another function which can be executed later in the code.
- **Usecases:**
  - setTimeOut
  - Higher order functions (Like map,filter,forEach).
  - Handling events (Like click/key press events).
  - Handling asynchronous operations (Like reading files, making Http requests).

```
function Print(){
        console.log("Print method");
}

function Hello(Print){
```

```
        console.log("Hello method");
        Print();
}

Hello(Print);

Output:
Hello method
Print method
```

---

33. What are Higher Order Functions in javascript ?

- A function which takes another function as an argument or returns a function as an output.
- **Advantages:**
  - callback functions
  - Asynchronous programming (functions like setTimeOut,setInterval often involves HOF. they allow to work with asynchronous code more effectively.)
  - Abstraction
  - Code reusability
  - Encapsulation
  - Concise and readable code

---

34. What is the difference between == and === in javascript ?

- == will check for equality of values where as === willl check for equality as well as datatypes.

---

35. Is javascript a dynamically typed language or a statically typed language ?

- Javascript is a dynamically typed language.
- It means all type checks are done at run time (When program is executing).
- So, we can just assign anything to the variable and it works fine.

```
let a;
a = 0;
console.log(a) // 0
a = "Hello"
console.log(a) // "Hello"
```

- Typescript is a statically typed language. All checks are performed at compile time.

---

36. What is the difference between Indexeddb and sessionstorage ?

- **IndexedDb:**
  - It is used for storing large amount of structured data.
  - It uses object oriented storage model.
  - Persist data beyond the duration of page session.
- **SessionStorage:**
  - Limited storage, around 5mb of data.
  - Simple key-value storage.
  - Available only for the duration of page session.

---

37. What are Interceptors ?

- Interceptors allows us to modify the request or response before its sent to the server or received from the server.

```
axios.interceptors.request.use((config)=>{
        if(longUrls.include(url)){
                config.timeout = 1000;
        }
        return config;
}

axios.interceptors.response.use((response)=>{
        return response;
})
```

---

38. What is Hoisting in javascript ?

- In other scripting/server side languages, variables or functions must be declared before using it.
- In javascript, variables and functions can be used before declaring it. The javascript compiler moves all the declarations of variables and functions on top. so there will not be any error. This is called hoisting.

---

39. What are the differences let, var and const ?

- **Scope:**
    - Variables declared with var are function scoped.(available through out the function where its declared) or global scoped(if defined outside the function).

    - Variables declared with let and const are block scoped.
- **Reassignment:**
    - var and let can be reassigned.
    - const cannot be reassigned.
- **Hoisting:**
    - var gets hoisted and initialized with undefined.
    - let and const - gets hoisted to the top of the scope but does not get assigned any value.(temporary dead zone)

---

40. What is the output of below logic ?

```
const a = 1<2<3;
const b = 1>2>3;

console.log(a,b) //true,false
```

---

41. Differences between Promise.all, allSettled, any, race ?

- **Promise.all:**
    - Will wait for all of the promises to resolve or any one of the promise reject.
- **Promise.allSettled:**
    - Will wait for all the promises to settle (either fulfilled or rejected).
- **Promise.any:**
    - Will return if any one of the promise fulfills or rejects when all the promises are rejected.
- **Promise.race:**

- Will return as soon as when any one of the promise is settled.

https://medium.com/@log2jeet24/javascript-different-types-of-promise-object-methods-to-handle-the-asynchronous-call-fc93d1506574

---

42. What are limitations of arrow functions in javascript ?

Arrow functions are introduced in ES6. They are simple and shorter way to write functions in javascript.

1. Arrow functions cannot be accessed before initialization
2. Arrow function does not have access to arguments object
3. Arrow function does not have their own this. Instead, they inherit this from the surrounding code at the time the function is defined.
4. Arrow functions cannot be used as constructors. Using them with the ***new*** keyword to create instances throws a TypeError.
5. Arrow functions cannot be used as generator functions.

---

43. What is difference between find vs findIndex ?

- **find:**
  - It will return the first element of array that passes specified condition.

```
function findMethod(){
  let arr = [{id:1,name:"sai"},{id:2,name:"krishna"}];
  let data = arr.find(x=> x.id==2)
  console.log(data)
}

findMethod()
```

```
Output:
{id:2,name:"krishna"}
```

- **findIndex:**
  - It will return the index of first element of an array that passes the specified condition.

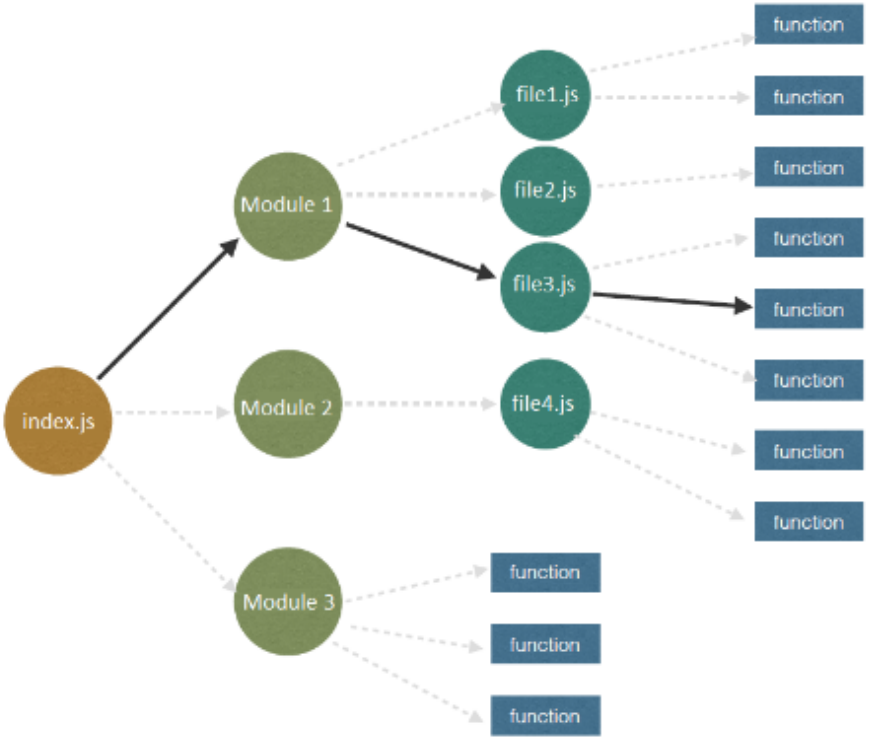
```
function findMethod(){
  let arr = [{id:1,name:"sai"},{id:2,name:"krishna"}];
  let data = arr.findIndex(x=> x.id==2)
  console.log(data)
}

findMethod()

Output:
2
```

---

44. What is tree shaking in javascript ?

- It is one of the optimization technique in javascript which removes the unused code from the bundle during the build process.
- It is commonly used in bundling tools like Webpack and Rollup.
- **Advantages:**
  - It reduces the bundle size by eleminating unused modules and functions.
  - Faster load time.
  - Performance will be improved.
  - Cleaner and maintainable codebases.

Before Tree Shaking



After Tree Shaking