1. What do you understand about JavaScript?

JavaScript is a popular web scripting language and is used for client-side and server-side development. The JavaScript code can be inserted into HTML pages that can be understood and executed by web browsers while also supporting object-oriented programming abilities.

2. What's the difference between JavaScript and Java?

JavaScript	Java
JavaScript is an object-oriented scripting language.	Java is an object-oriented programming language.
JavaScript applications are meant to run inside a web browser.	Java applications are generally made for use in operating systems and virtual machines.
JavaScript does not need compilation before running the application code.	Java source code needs a compiler before it can be ready to run in realtime.

3. What are the various data types that exist in JavaScript?

These are the different types of data that JavaScript supports:

- Boolean - For true and false values
- Null - For empty or unknown values
- Undefined - For variables that are only declared and not defined or initialized
- Number - For integer and floating-point numbers
- String - For characters and alphanumeric values
- Object - For collections or complex values
- Symbols - For unique identifiers for objects

4. What are the features of JavaScript?

These are the features of JavaScript:

- Lightweight, interpreted programming language
- Cross-platform compatible
- Open-source
- Object-oriented
- Integration with other backend and frontend technologies
- Used especially for the development of network-based applications

5. What are the advantages of JavaScript over other web technologies?

These are the advantages of JavaScript:

Enhanced Interaction

JavaScript adds interaction to otherwise static web pages and makes them react to users' inputs.

Quick Feedback

There is no need for a web page to reload when running JavaScript. For example, form input validation.

Rich User Interface

JavaScript helps in making the UI of web applications look and feel much better.

Frameworks

JavaScript has countless frameworks and libraries that are extensively used for developing web applications and games of all kinds.

6. How do you create an object in JavaScript?

Since JavaScript is essentially an object-oriented scripting language, it supports and encourages the usage of objects while developing web applications.

```
const student = {
   name: 'John',
   age: 17
}
```

7. How do you create an array in JavaScript?

Here is a very simple way of creating arrays in JavaScript using the array literal:

var a = [];

var b = ['a', 'b', 'c', 'd', 'e'];

8. What are some of the built-in methods in JavaScript?

Built-in Method	Values
Date()	Returns the present date and time
concat()	Joins two strings and returns the new string
push()	Adds an item to an array
pop()	Removes and also returns the last element of an array
round()	Rounds of the value to the nearest integer and then returns it
length()	Returns the length of a string

9. What are the scopes of a variable in JavaScript?

The scope of a variable implies where the variable has been declared or defined in a JavaScript program. There are two scopes of a variable:

Global Scope

Global variables, having global scope are available everywhere in a JavaScript code.

Local Scope

Local variables are accessible only within a function in which they are defined.

10. What is the 'this' keyword in JavaScript?

The 'this' keyword in JavaScript refers to the currently calling object. It is commonly used in constructors to assign values to object properties.

11. What are the conventions of naming a variable in JavaScript?

Following are the naming conventions for a variable in JavaScript:

- Variable names cannot be similar to that of reserved keywords. For example, var, let, const, etc.
- Variable names cannot begin with a numeric value. They must only begin with a letter or an underscore character.
- Variable names are case-sensitive.

12. What is Callback in JavaScript?

In JavaScript, functions are objects and therefore, functions can take other functions as arguments and can also be returned by other functions.

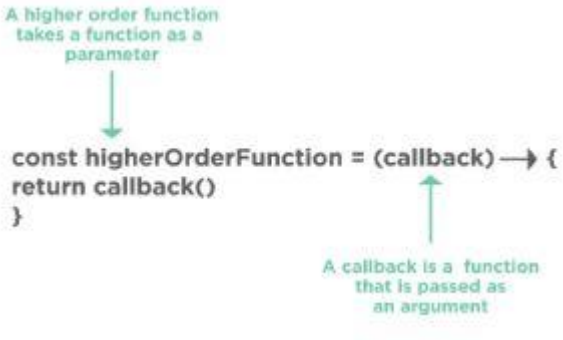


Fig: Callback function

A callback is a JavaScript function that is passed to another function as an argument or a parameter. This function is to be executed whenever the function that it is passed to gets executed.

13. How do you debug a JavaScript code?

All modern web browsers like Chrome, Firefox, etc. have an inbuilt debugger that can be accessed anytime by pressing the relevant key, usually the F12 key. There are several features available to users in the debugging tools.

We can also debug a JavaScript code inside a code editor that we use to develop a JavaScript application—for example, Visual Studio Code, Atom, Sublime Text, etc.

14. What is the difference between Function declaration and Function expression?

Function declaration	Function expression
Declared as a separate statement within the main JavaScript code	Created inside an expression or some other construct
Can be called before the function is defined	Created when the execution point reaches it; can be used only after that
Offers better code readability and better code organization	Used when there is a need for a conditional declaration of a function

Example:
function abc() {
return 5;
}
	Example:
	var a = function abc() {
	return 5;
	}

15. What are the ways of adding JavaScript code in an HTML file?

There are primarily two ways of embedding JavaScript code:

- We can write JavaScript code within the script tag in the same HTML file; this is suitable when we need just a few lines of scripting within a web page.
- We can import a JavaScript source file into an HTML document; this adds all scripting capabilities to a web page without cluttering the code.

Intermediate JavaScript Interview Questions and Answers

16. What do you understand about cookies?

A cookie is generally a small data that is sent from a website and stored on the user's machine by a web browser that was used to access the website. Cookies are used to remember information for later use and also to record the browsing activity on a website.

17. How would you create a cookie?

The simplest way of creating a cookie using JavaScript is as below:
document.cookie = "key1 = value1; key2 = value2; expires = date";

18. How would you read a cookie?

Reading a cookie using JavaScript is also very simple. We can use the document.cookie string that contains the cookies that we just created using that string.
The document.cookie string keeps a list of name-value pairs separated by semicolons, where 'name' is the name of the cookie, and 'value' is its value. We can also use the split() method to break the cookie value into keys and values.

19. How would you delete a cookie?

To delete a cookie, we can just set an expiration date and time. Specifying the correct path of the cookie that we want to delete is a good practice since some browsers won't allow the deletion of cookies unless there is a clear path that tells which cookie to delete from the user's machine.

```
function delete_cookie(name) {
  document.cookie = name + "=; Path=/; Expires=Thu, 01 Jan 1970 00:00:01 GMT;";
}
```

20. What's the difference between let and var?

Both let and var are used for variable and method declarations in JavaScript. So there isn't much of a difference between these two besides that while var keyword is scoped by function, the let keyword is scoped by a block.

21. What are Closures in JavaScript?

Closures provide a better, and concise way of writing JavaScript code for the developers and programmers. Closures are created whenever a variable that is defined outside the current scope is accessed within the current scope.

```
function hello(name) {
  var message = "hello " + name;
  return function hello() {
    console.log(message);
  };
}
//generate closure
var helloWorld = hello("World");
//use closure
helloWorld();
```

22. What are the arrow functions in JavaScript?

Arrow functions are a short and concise way of writing functions in JavaScript. The general syntax of an arrow function is as below:

```
const helloWorld = () => {
  console.log("hello world!");
};
```

23. What are the different ways an HTML element can be accessed in a JavaScript code?

Here are the ways an HTML element can be accessed in a JavaScript code:

- getElementByClass('classname'): Gets all the HTML elements that have the specified classname.
- getElementById('idname'): Gets an HTML element by its ID name.
- getElementbyTagName('tagname'): Gets all the HTML elements that have the specified tagname.
- querySelector(): Takes CSS style selector and returns the first selected HTML element.

24. What are the ways of defining a variable in JavaScript?

There are three ways of defining a variable in JavaScript:

Var

This is used to declare a variable and the value can be changed at a later time within the JavaScript code.

Const

We can also use this to declare/define a variable but the value, as the name implies, is constant throughout the JavaScript program and cannot be modified at a later time.

Let

This mostly implies that the values can be changed at a later time within the JavaScript code.

25. What are Imports and Exports in JavaScript?

Imports and exports help in writing modular code for our JavaScript applications. With the help of imports and exports, we can split a JavaScript code into multiple files in a project. This greatly simplifies the application source code and encourages code readability.

calc.js

```
export const sqrt = Math.sqrt;
export function square(x) {
  return x * x;
```

```
}
export function diag(x, y) {
  return sqrt(square(x) + square(y));
}
```

This file exports two functions that calculate the squares and diagonal of the input respectively.

main.js

```
import { square, diag } from "calc";
console.log(square(4)); // 16
console.log(diag(4, 3)); // 5
```

Therefore, here we import those functions and pass input to those functions to calculate square and diagonal.

26. What is the difference between Document and Window in JavaScript?

Document	Window
The document comes under the windows object and can also be considered as its property.	Window in JavaScript is a global object that holds the structure like variables, functions, location, history, etc.

27. What are some of the JavaScript frameworks and their uses?

JavaScript has a collection of many frameworks that aim towards fulfilling the different aspects of the web application development process. Some of the prominent frameworks are:

- React - Frontend development of a web application
- Angular - Frontend development of a web application
- Node - Backend or server-side development of a web application

28. What is the difference between Undefined and Undeclared in JavaScript?

Undefined	Undeclared
Undefined means a variable has been declared but a value has not yet been assigned to that variable.	Variables that are not declared or that do not exist in a program or application.

29. What is the difference between Undefined and Null in JavaScript?

Undefined	Null
Undefined means a variable has been declared but a value has not yet been assigned to that variable.	Null is an assignment value that we can assign to any variable that is meant to contain no value.

30. What is the difference between Session storage and Local storage?

Session storage	Local storage
The data stored in session storage gets expired or deleted when a page session ends.	Websites store some data in local machine to reduce loading time; this data does not get deleted at the end of a browsing session.

31. What are the various data types that exist in JavaScript?

Javascript consists of two data types, primitive data types, and non-primitive data types.

- Primitive Data types: These data types are used to store a single value. Following are the sub-data types in the Primitive data type.
- Boolean Data Types: It stores true and false values.

Example:

```
var a = 3;
var b =  4;
var c = 3;
(a == b) // returns false
(a == c) //returns true
```

- Null data Types: It stores either empty or unknown values.

Example:

```
var z = null;
```

- Undefined data Types: It stores variables that are only declared, but not defined or initialized.

Example:

```
var a; // a is undefined
var b = undefined; // we can also set the value of b variable as undefined
```

- Number Data Types: It stores integer as well as floating-point numbers.

Example:

```
var x = 4; //without decimal
var y = 5.6; //with decimal
```

- String data Types: It stores characters and alphanumeric values.

Example:

```
var str = "Raja Ram Mohan"; //using double quotes
var str2 = 'Raja Rani'; //using single quotes
```

- Symbols Data Types: It stores unique identifiers for objects.

Example:

```
var symbol1 = Symbol('symbol');
```

- BigInt Data Types: It stores the Number data types that are large integers and are above the limitations of number data types.

Example:

```
var bigInteger =  234567890123456789012345678901234567890;
```

- Non-Primitive Data Types

Non-Primitive data types are used to store multiple as well as complex values.

Example:

```
// Collection of data in key-value pairs
var obj1 = {
  x:  43,
  y:  "Hello world!",
  z: function(){
    return this.x;
  }
}
// Data collection with an ordered list
var array1 = [5, "Hello", true, 4.1];
```

32. What is the ‘this’ keyword in JavaScript?

The Keyword ‘this’ in JavaScript is used to call the current object as a constructor to assign values to object properties.

33. What is the difference between Call and Apply? (explain in detail with examples)

- Call

Call uses arguments separately.

Example:

```
function sayHello()
{
  return "Hello " + this.name;
}
var obj = {name: "Sandy"};
sayHello.call(obj);
// Returns "Hello Sandy"
```

- Apply

Apply uses an argument as an array.

Example:

```
function saySomething(message)
{
  return this.name + " is " + message;
}
var person4 = {name:  "John"};
saySomething.apply(person4, ["awesome"]);
```

34. What are the scopes of a variable in JavaScript?

The scope of variables in JavaScript is used to determine the accessibility of variables and functions at various parts of one's code. There are three types of scopes of a variable, global scope, function scope, block scope

- Global Scope: It is used to access the variables and functions from anywhere inside the code.

Example:

```
var globalVariable = "Hello world";
function sendMessage(){
  return globalVariable; // globalVariable is accessible as it's written in global space
}
function sendMessage2(){
  return sendMessage(); // sendMessage function is accessible as it's written in global space
}
sendMessage2();  // Returns "Hello world"
```

- Function scope: It is used to declare the function and variables inside the function itself and not outside.

Example:

```
function awesomeFunction()
{
  var a = 3;
```

```
  var multiplyBy3 = function()
{
    console.log(a*3); // Can access variable "a" since a and multiplyBy3 both are written inside the same function
  }
}
console.log(a); // a is written in local scope and can't be accessed outside and throws a reference error
multiplyBy3(); // MultiplyBy3 is written in local scope thus it throws a reference error
```

- Block Scope: It uses let and const to declare the variables.

Example:

```
{
  let x = 45;
}
console.log(x); // Gives reference error since x cannot be accessed outside of the block
for(let i=0; i<2; i++){
  // do something
}
console.log(i); // Gives reference error since i cannot be accessed outside of the for loop block
```

35. What are the arrow functions in JavaScript?

Arrow functions are used to write functions with a short and concise syntax. Also, it does not require a function keyword for declaration. An arrow function can be omitted with curly braces { } when we have one line of code.

syntax of an arrow function:

```
const helloWorld = () => {
  console.log("hello world!");
};
```

Example:

```
// Traditional Function Expression
var add = function(a,b)
```

```
{
  return a + b;
}
// Arrow Function Expression
var arrowAdd = (a,b) => a + b;
```

36. Explain Hoisting in javascript. (with examples)

Hoisting in javascript is the default process behavior of moving declaration of all the variables and functions on top of the scope where scope can be either local or global.

Example 1:

```
hoistedFunction();  // " Hi There! " is an output that is declared as function even after it is called
function hoistedFunction(){
  console.log(" Hi There! ");
}
```

Example 2:

```
hoistedVariable = 5;
console.log(hoistedVariable); // outputs 5 though the variable is declared after it is initialized
var hoistedVariable;
```

37. Difference between “ == “ and “ === “ operators (with examples)

1. “==” operator is a comparison operator that used to compare the values
2. “===” operator is also a comparison operator that is used to compare the values as well as types.

Example:

```
var x = 3;
var y = "3";
(x == y)  // it returns true as the value of both x and y is the same
(x === y) // it returns false as the typeof x is "number" and typeof y is "string"
```

38. Difference between var and let keyword

- Keyword “var”

- In JavaScript programming, the “var” keyword has been used from the very initial stages of JavaScript.
- We can perform functions with the help of the keyword “var’ by accessing various variables.

- Keyword “let”

- The Keyword “let” was added later in ECMAScript 2015 in JavaScript Programming.
- Variable declaration is very limited with the help of the “let” keyword that is declared in Block. Also, it might result in a ReferenceError as the variable was declared in the “temporal dead zone” at the beginning of the block.

39. Implicit Type Coercion in javascript (in detail with examples)

When the value of one data type is automatically converted into another data type, it is called Implicit type coercion in javascript.

- String coercion

Example:

```
var x = 4;
var y = "4";
x + y // Returns "44"
```

- Boolean coercion

Example:

```
var a = 0;
var b = 32;
if(a) { console.log(a) }   // This code will run inside the block as the value of x is 0(Falsy)
```

if(b) { console.log(b) } // This code will run inside the block as the value of y is 32 (Truthy)

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

Yes, JavaScript is a dynamically typed language and not statically.

41. NaN property in JavaScript

NaN property in JavaScript is the “Not-a-Number” value that is not a legal number.

42. Passed by value and passed by reference

- Passed By Values Are Primitive Data Types.

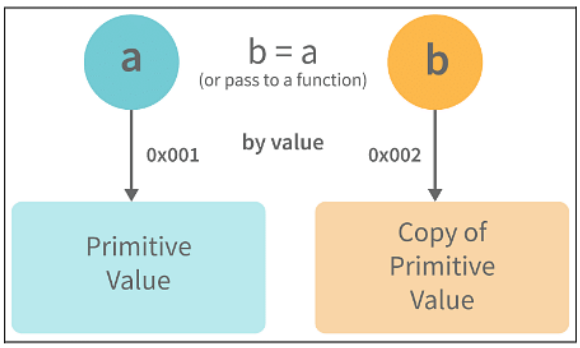
Consider the following example:

Here, the a=432 is a primitive data type i.e. a number type that has an assigned value by the operator. When the var b=a code gets executed, the value of ‘var a’ returns a new address for ‘var b’ by allocating a new space in the memory, so that ‘var b’ will be operated at a new location.

Example:

var a = 432;

var b = a;



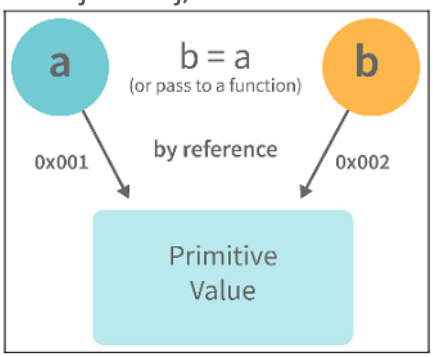
- Passed by References Are Non-primitive Data Types.

Consider the following example:

The reference of the 1st variable object i.e. ‘var obj’ is passed through the location of another variable i.e. ‘var obj2’ with the help of an assigned operator.

Example:

```
var obj = { name: "Raj", surname: "Sharma" };
var obj2 = obj;
```



43. Immediately Invoked Function in JavaScript

An Immediately Invoked Function also abbreviated as IIFE or IIFY runs as soon as it is defined. To run the function, it needs to be invoked otherwise the declaration of the function is returned.

Syntax

```
(function()
{   // Do something;
})
();
```

44. Characteristics of javascript strict-mode

- Strict mode does not allow duplicate arguments and global variables.
- One cannot use JavaScript keywords as a parameter or function name in strict mode.
- All browsers support strict mode.
- Strict mode can be defined at the start of the script with the help of the keyword ‘use strict’.

45. Higher Order Functions (with examples)

Higher-order functions are the functions that take functions as arguments and return them by operating on other functions

Example:

```
function higherOrder(fn)
 {
  fn();
}
higherOrder(function() { console.log("Hello world") });
```

46. Self Invoking Functions

Self Invoking Functions is an automatically invoked function expression followed by (), where it does not need to be requested. Nevertheless, the declaration of the function is not able to be invoked by itself.

47. difference between exec () and test () methods

- exec()

- It is an expression method in JavaScript that is used to search a string with a specific pattern.
- Once it has been found, the pattern will be returned directly, otherwise, it returns an “empty” result.

- test ()

- It is an expression method in JavaScript that is also used to search a string with a specific pattern or text.
- Once it has been found, the pattern will return the Boolean value 'true', else it returns ‘false’.

48. currying in JavaScript (with examples)

In JavaScript, when a function of an argument is transformed into functions of one or more arguments is called Currying.

Example:

```
function add (a) {
  return function(b){
    return a + b;
  }
}
add(3)(4)
```

49. Advantages of using External JavaScript

- External Javascript allows web designers and developers to collaborate on HTML and javascript files.
- It also enables you to reuse the code.
- External javascript makes Code readability simple.

50. What are object prototypes?

Following are the different object prototypes in javascript that are used to inherit particular properties and methods from the Object.prototype.

1. Date objects are used to inherit properties from the Date prototype
2. Math objects are used to inherit properties from the Math prototype
3. Array objects are used to inherit properties from the Array prototype.

51. Types of errors in javascript

Javascript has two types of errors, Syntax error, and Logical error.

52. What is memoization?

In JavaScript, when we want to cache the return value of a function concerning its parameters, it is called memoization. It is used to speed up the application especially in case of complex, time consuming functions.

53. Recursion in a programming language

Recursion is a technique in a programming language that is used to iterate over an operation whereas a function calls itself repeatedly until we get the result.

54. Use of a constructor function (with examples)

Constructor functions are used to create single objects or multiple objects with similar properties and methods.

Example:

```
function Person(name,age,gender)
{
  this.name = name;
  this.age = age;
  this.gender = gender;
}
var person1 = new Person("Vivek", 76, "male");
console.log(person1);
var person2 = new Person("Courtney", 34, "female");
console.log(person2);
```

55. Which method is used to retrieve a character from a certain index?

We can retrieve a character from a certain index with the help of charAt() function method.

56. What is BOM?

BOM is the Browser Object Model where users can interact with browsers that is a window, an initial object of the browser. The window object consists of a document, history, screen, navigator, location, and other attributes. Nevertheless, the window's function can be called directly as well as by referencing the window.

57. Difference between client-side and server-side

- Client-side JavaScript

- Client-side JavaScript is made up of fundamental language and predefined objects that perform JavaScript in a browser.
- Also, it is automatically included in the HTML pages where the browser understands the script.

- Server-side Javascript

- Server-side JavaScript is quite similar to Client-side javascript.
- Server-side JavaScript can be executed on a server.
- The server-side JavaScript is deployed once the server processing is done.

58. What is the prototype design pattern?

The Prototype design Pattern is also known as a property or prototype pattern that is used to produce different objects as well as prototypes that are replicated from a template with a specific value.

59. Differences between declaring variables using var, let and const.

var	let	const
There is a global scope as well as a function scope.	There is neither a global scope nor a function scope.	There is neither a global scope nor a function scope.
1. There is no block scope.	There is no block scope.	There is no block scope.
It can be reassigned.	clt cannot be reassigned.	It cannot be reassigned.

Example 1: Using 'var' and 'let' variable

```
var variable1 = 31;
let variable2 = 89;
function catchValues()
{
  console.log(variable1);
  console.log(variable2);
// Both the variables are accessible from anywhere as their declaration is in the global
scope
}
window.variable1; // Returns the value 31
window.variable2; // Returns undefined
```

Example 2: Using 'const' variable

```
const x = {name:"Vijay"};
x = {address: "Mumbai"}; // Throws an error
x.name = "Radha"; // No error is thrown
const y = 31;
y = 44; // Throws an error
```

60. Rest parameter and spread operator

- Rest Parameter(...)

- Rest parameter is used to declare the function with improved handling of parameters.
- Rest parameter syntax can be used to create functions to perform functions on the variable number of arguments.
- It also helps to convert any number of arguments into an array as well as helps in extracting some or all parts of the arguments.

- Spread Operator(...)

- In a function call, we use the spread operator.
- It's also to spread one or more arguments that are expected in a function call.
- The spread operator is used to take an array or an object and spread them.

61. Promises in JavaScript

Promises in JavaScript have four different states. They are as follows:

Pending	Fulfilled	Rejected	Settled
Pending is an initial state of promise. It is the initial state of promise where it is in the pending state that neither is fulfilled nor rejected.	It is the state where the promise has been fulfilled that assures that the async operation is done.	It is the state where the promise is rejected and the async operation has failed.	It is the state where the promise is rejected or fulfilled.

Example:

```
function sumOfThreeElements(...elements)
{
  return new Promise((resolve,reject)=>{
    if(elements.length > 3 )
{
```

```
    reject("Just 3 elements or less are allowed");
  }
  else
{
    let sum = 0;
    let i = 0;
    while(i < elements.length)
{
      sum += elements[i];
      i++;
    }
    resolve("Sum has been calculated: "+sum);
  }
 })
}
```

62. Classes in JavaScript

classes are syntactic sugars for constructor functions mentioned in the ES6 version of JavaScript. Classes are not hoisted-like Functions and can't be used before it is declared. Also, it can inherit properties and methods from other classes with the help of extended keywords. If the strict mode ('use strict') is not followed, an error will be shown.

63. What are generator functions?

Generator functions are declared with a special class of functions and keywords using function*. It does not execute the code, however, it returns a generator object and handles the execution.

64. What is WeakSet?

WeakSet is a collection of unique and ordered elements that contain only objects which are referenced weakly.

65. What is the use of callbacks?

- A callback function is used to send input into another function and is performed inside another function.
- It also ensures that a particular code does not run until another code has completed its execution.

66. What is a WeakMap?

Weakmap is referred to as an object having keys and values, if the object is without reference, it is collected as garbage.

67. What is Object Destructuring? (with examples)

Object destructuring is a method to extract elements from an array or an object.

Example 1: Array Destructuring

```
const arr = [1, 2, 3];
const first = arr[0];
const second = arr[1];
const third = arr[2];
```

Example 2: Object Destructuring

```
const arr = [1, 2, 3];
const [first,second,third,fourth] = arr;
console.log(first); // Outputs 1
console.log(second); // Outputs 2
console.log(third); // Outputs 3
```

68. Prototypal vs Classical Inheritance

- Prototypal Inheritance
- Prototypal inheritance allows any object to be cloned via an object linking method and it serves as a template for those other objects, whether they extend the parent object or not.
- Classical Inheritance
- Classical inheritance is a class that inherits from the other remaining classes.

69. What is a Temporal Dead Zone?

Temporal Dead Zone is a behavior that occurs with variables declared using let and const keywords before they are initialized.

70. JavaScript Design Patterns

When we build JavaScript browser applications, there might be chances to occur errors where JavaScript approaches it in a repetitive manner. This repetitive approach pattern is called JavaScript design patterns. JavaScript design patterns consist of Creational Design Pattern, Structural Design Pattern, and Behavioral Design patterns.

71. Difference between Async/Await and Generators

- Async/Await
- Async-await functions are executed sequentially one after another in an easier way.
- Async/Await function might throw an error when the value is returned.
- Generators
- Generator functions are executed with one output at a time by the generator's yield by yield.
- The 'value: X, done: Boolean' is the output result of the Generator function.

72. Primitive data types

The primitive data types are capable of displaying one value at a time. It consists of Boolean, Undefined, Null, Number, and String data types.

73. Role of deferred scripts

The Deferred scripts are used for the HTML parser to finish before executing it.

74. What is Lexical Scoping?

Lexical Scoping in JavaScript can be performed when the internal state of the JavaScript function object consists of the function's code as well as references concerning the current scope chain.

75. What is this [[[]]]?

This '[[[]]]' is a three-dimensional array.

76. Are Java and JavaScript the same?

Yes, Java and JavaScript are the same.

77. How to detect the OS of the client machine using JavaScript?

The OS on the client machine can be detected with the help of navigator.appVersion string

78. Requirement of debugging in JavaScript

- To debug the code, we can use web browsers such as Google Chrome, and Mozilla Firefox.
- We can debug in JavaScript with the help of two methods, console.log() and debugger keyword.

79. What are the pop-up boxes available in JavaScript?

Pop-up boxes available in JavaScript are Alert Box, Confirm Box, and Prompt Box.

Advanced JS Interview Questions and Answers

Here are some advanced level JavaScript interview questions and answers for you to prepare during your interviews.

80. How do you empty an array in JavaScript?

There are a few ways in which we can empty an array in JavaScript:

- By assigning array length to 0:

```
var arr = [1, 2, 3, 4];
arr.length = 0;
```

- By assigning an empty array:

```
var arr = [1, 2, 3, 4];
arr = [];
```

- By popping the elements of the array:

```
var arr = [1, 2, 3, 4];
while (arr.length > 0) {
  arr.pop();
}
```

- By using the splice array function:

```
var arr = [1, 2, 3, 4];
arr.splice(0, arr.length);
```

81. What is the difference between Event Capturing and Event Bubbling?

Event Capturing	Event Bubbling
This process starts with capturing the event of the outermost element and then propagating it to the innermost element.	This process starts with capturing the event of the innermost element and then propagating it to the outermost element.

82. What is the Strict mode in JavaScript?

Strict mode in JavaScript introduces more stringent error-checking in a JavaScript code.

- While in Strict mode, all variables have to be declared explicitly, values cannot be assigned to a read-only property, etc.
- We can enable strict mode by adding 'use strict' at the beginning of a JavaScript code, or within a certain segment of code.

83. What would be the output of the below JavaScript code?

```
var a = 10;
if (function abc(){})
{
a += typeof abc;
}
console.log(a);
```

The output of this JavaScript code will be 10undefined. The if condition statement in the code evaluates using eval. Hence, eval(function abc(){}) will return function abc(){}. Inside the if statement, executing typeof abc returns undefined because the if statement code executes at run time while the statement inside the if the condition is being evaluated.

84. Can you write a JavaScript code for adding new elements in a dynamic manner?

```
<script type="text/javascript">
function addNode() {
   var newP = document.createElement("p");
   var textNode = document.createTextNode(" This is a new text node");
   newP.appendChild(textNode);
document.getElementById("firstP").appendChild(newP);
}
</script>
```

85. What is the difference between Call and Apply?

Call	Apply
In the call() method, arguments are provided individually along with a ‘this’ value.	In the apply() method, arguments are provided in the form of an array along with a ‘this’ value.

86. What will be the output of the following code?

```
var Bar = Function Foo()
{
return 11;
};
typeof Foo();
```

The output would be a reference error since a function definition can only have a single reference variable as its name.

87. What will be the output of the following code?

```
var Student = {
  college: "abc",
};
var stud1 = Object.create(Student);
delete stud1.college;
console.log(stud1.company);
```

This is essentially a simple example of object-oriented programming. Therefore, the output will be ‘abc’ as we are accessing the property of the student object.

88. How do you remove duplicates from a JavaScript array?

There are two ways in which we can remove duplicates from a JavaScript array:

By Using the Filter Method

To call the filter() method, three arguments are required. These are namely array, current element, and index of the current element.

By Using the For Loop

An empty array is used for storing all the repeating elements.

81. Can you draw a simple JavaScript DOM (Document Object Model)?