**JAVASCRIPT INTERVIEW QUESTIONS**

1. What is JavaScript?

2. What are the features of JavaScript?

3. JavaScript Data Types

4. What is ECMAScript?

5. What are different versions of ECMAScript?

6. What are the features of ES6?

7. What are scopes in JavaScript?

8. Difference between let, var, and const.

9. Which one should we use between let, var, and const?

10. What is Temporal Dead Zone?

11. What is hoisting?

12. Whether JavaScript is object-oriented or object-based?

13. Is JavaScript synchronous or asynchronous?

14. How does JavaScript execute asynchronous code?

15. What is a callback?

16. What is callback hell?

17. How can we avoid callback hell?

18. What is the promise?

19. What are the states of promise?

20. What is promise chaining?

21. How can we achieve multiple parallel asynchronous requests using promise?

22. What is async and await?

23. Difference between Promise and Callback

24. Which one is better between callback, promise, and async-await?

25. What is a closure?

26. What is an anonymous function?

27. What is a function expression?

28. What is an arrow function?

29. What is the first-class function?

30. What is the first-order function?

31. What is the higher-order function?

32. What is the unary function?

33. What is a pure function?

34. What is IIFE?

35. What is the currying function?

36. What is a nested function?

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

38. What is the call() method?

39. What is apply() method?

40. What is bind() method?

41. What is the difference between Call, Apply and Bind?

42. What is a class in JavaScript?

43. How to create private members in class?

44. How to create static members in class?

45. What is a template string?

46. What are the default parameters?

47. What is a spread operator?

48. What is rest parameter?

49. What is array destructuring?

50. What is event bubbling?

51. What is event capturing?

52. What is DOM?

53. What is the difference between a document and a window?

54. What is a cookie?

55. What is session storage?

56. What is local storage?

57. What is the difference between a cookie, session storage, and local storage?

58. What is the preventDefault() method?

59. How do you access history in JavaScript?

60. What is call stack?

61. What is an event loop?

62. How can we handle exceptions in JavaScript?

63. What is the throw keyword?

64. What is finally block?

65. What is JSON?

66. JSON methods in JavaScript

67. JSON data types

68. What is strict mode in JavaScript?

69. What is NaN?

70. What is undefined?

71. What is null?

72. What is the difference between undefined and null?

73. What is the difference between == and ===?

74. What is typeof operator?

75. What is call by value?

76. What is call by reference?

77. Shallow copy vs deep copy

78. Which memory does JavaScript use?

79. Does JavaScript use garbage collection?

**Basic JavaScript interview questions**

**1. What is JavaScript?**

**JavaScript** is an **interpreted** language. Because of the newer **ECMAScript** feature, it can be compiled as well. A browser is responsible to execute or run the JavaScript using **web APIs.** We can use **Node.js** to execute the JavaScript outside of the browser as well. So, web APIs are responsible to execute JavaScript inside the browser. While **Node.js APIs** are responsible to execute the JavaScript outside the browser.

**2. What are the features of JavaScript?**

* Interpreted
* Dynamically typed
* Platform independent
* Object-based
* Light-weight
* Synchronous
* Single-threaded

**3. JavaScript Data Types**

JavaScript supports 2 data types

* Primitive
* Object

**Primitive Data Types**

* Bigint
* Boolean
* Null
* Number
* String
* Symbol
* Undefined

**Object Data Types**

* Object
* Array
* Map
* Set
* Date
* Error
* Json

**4. What is ECMAScript?**

**ECMAScript** is a specification to write standard JavaScript. It provides some rules and regulations. Browsers implement these rules to create JavaScript engines.

**5. What are different versions of ECMAScript?**

* Version Description and Release
* ES1 June 1997
* ES2 June 1998
* ES3 June 1999
* ES4 Abandoned
* ES5 December 2009
* ES6 June 2015 (ES2015)
* ES7 June 2016 (ES2016)
* ES8 June 2017 (ES2017)
* ES9 June 2018 (ES2018)
* ES10 June 2019 (ES2019)
* ES11 June 2020 (ES2020)
* ES12 June 2021 (ES2021)

**6. What are the features of ES6?**

Here is the list of ES6 features.

* let and const Keywords
* Arrow Functions
* Default Parameters
* Template Strings
* Destructuring Assignment
* Enhanced Object Literals
* Rest Parameters
* Spread Operator
* Promises
* Classes
* Modules

**7. What are scopes in JavaScript?**

The scope refers to the current execution context. It specifies the accessibility of variables and functions. There are three scopes in JavaScript.

* **Global scope**
* **Function scope**
* **Block scope**

**8. Difference between let, var, and const.**

|  |  |  |
| --- | --- | --- |
| **Var** | **Let** | **Const** |
| Available from the beginning. | Introduced in ES6. | Introduced in ES6. |
| Global as well as function scoped. | Block scoped. | Block scoped. |
| Can re-declare and update. | Cannot be re-declared but it can be updated. | Neither re-declare nor update. |
| Hoisted and initialized to undefined. | Hoisted but not initialized. | Hoisted but not initialized. |

**9. Which one should we use between let, var, and const?**

* Always declare variables with the **const** keyword.
* If the value of a variable is going to change in the future, then declare it with the **let** keyword.
* Avoid using the **var** keyword.

**10. What is Temporal Dead Zone?**

The **let** and const variable are not accessible before they are initialized with some value. And the phase between the starting of execution of block in which the **let** and **const** variable is declared till that variable being initialized is calling **Temporal Dead Zone** for the variable.

(OR)

The Temporal Dead Zone is the behaviour of JavaScript where the state of a variable is not reachable. It only happens when we declare a variable with the **let** and **const** keyword. Accessing a variable with let and const before its declaration causes the **Reference Error.**

**11. What is hoisting?**

The hoisting is the process of moving variable and function declarations to the top of the scope. JavaScript only hoists declaration. It does not hoist initializations.

**12. Whether JavaScript is object-oriented or object-based?**

JavaScript is object-based in nature. Because it does not support polymorphism. Polymorphism consists of method overloading and method overriding. There is no concept of overloading or overriding in JavaScript. The default behaviour of JavaScript is to overwrite the function. Hence JavaScript is an object-based programming language.

**JavaScript interview questions on asynchronous operations**

**13. Is JavaScript synchronous or asynchronous?**

JavaScript is synchronous and blocking in nature. Synchronous means to execute code in sequence. It does not execute the next block of code until and unless the execution of earlier code.

**14. How does JavaScript execute asynchronous code?**

JavaScript uses callbacks, promises and async/await to execute asynchronous code.

**15. What is a callback?**

The callback is a function passed to another function as an argument. This passed function executes later. Usually, callbacks achieve asynchronous operations in JavaScript.



**16. What is callback hell? (Pyramid Of Doom)**

The callback hell happens when we use multiple nested callbacks. This problem occurs for dependent asynchronous operations (where we need to wait for the result of the previous asynchronous call).



**17. How can we avoid callback hell?**

We can avoid the callback hell problem by using promise and async-await.

**18. What is the promise?**

The promise is useful to achieve asynchronous programming in JavaScript. It is an object representing the completion or failure of an asynchronous operation.

The promise was introduced in ES6. It was introduced to overcome the limitations caused by the callback. The promise is useful for single as well as multiple asynchronous operations.

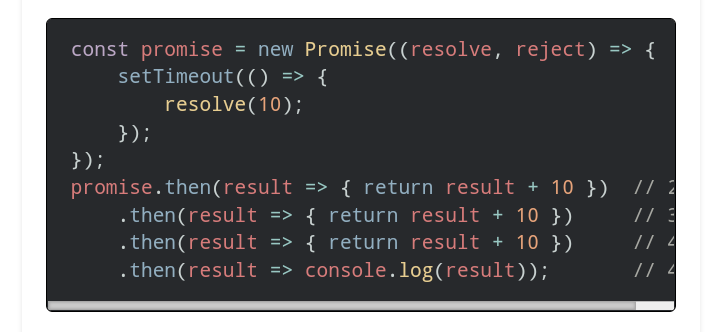
|  |
| --- |
| **Syntax: var promise =** |
| **new Promise((success resolver callback, rejectorCallback ) => {** |
| **});** |
| **• Invoking Promise:** |
| **Method(<optionalparam>). then(successcallback). catch(error callback); 19. What are the states of promise?** |

A promise can be in one of these states:

* Pending: initial state, neither fulfilled nor rejected.
* Fulfilled: meaning that the operation completes successfully.
* Rejected: meaning that the operation failed.

**20. What is promise chaining?**

Promise chaining means passing a promise to another promise. Sometimes, we have to make an asynchronous call based on the result of the previous asynchronous call. For this purpose, we can use promise chaining. We can achieve the promise chaining by using promise instance methods (then(), catch(), and finally()).



**21. How can we achieve multiple parallel asynchronous requests using promise?**

Sometimes we need to call multiple requests on a single page. We can achieve this by sending requests in parallel. The promise provides some helpful methods to work with parallel requests. Here is the list of methods.

* all()
* allSettled()
* any()
* race()

**22. What is async and await?**

The async keyword with a function represents that function as asynchronous. The async function returns the promise. Await keyword is used to represent the asynchronous operation inside the async function.

The async and await is syntactic sugar over the promise.



**23. Difference between Promise and Callback**

|  |  |
| --- | --- |
| **Callback** | **Promises** |
| The callback is a function passed to another function as an argument. | A promise is an object representing the completion or failure of an asynchronous operation. |
| It suffers from a callback hell problem for sequential asynchronous calls. | We can use promise chaining for sequential asynchronous code. |
| It is difficult to use callback for parallel asynchronous calls. | It provides different methods to work with parallel multiple requests. |
| Error handling is complex. | Error handling is easy. |

**24. Which one is better between callback, promise, and async-await?**

The callback is the old way to handle asynchronous operations. The promise provides a better way to handle asynchronous operations. It provides different methods.

Async and await is the syntactic sugar over the promise. It provides a way to write asynchronous code in a synchronous way. In modern applications, a lot of developers prefer **async and await** since it is easy to write and maintain.

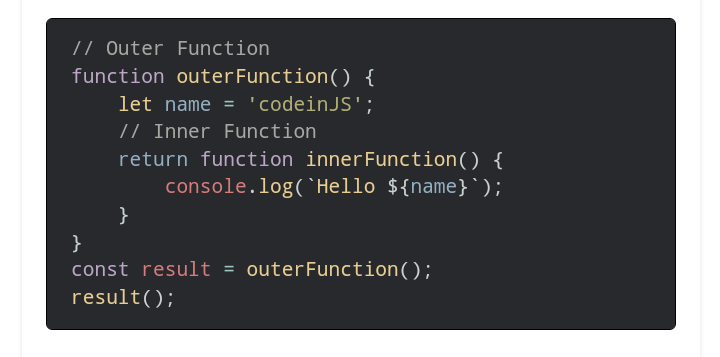
**JavaScript interview questions on functions**

**25. What is a closure?**

\_The Function along with its lexical scope bound together they form a **Closure.**

The closure is a function declared inside another function. The inner function has access to the outer function’s variables and parameters. The outer function should return an inner function to the outside world.

Generally, closures were useful to achieve encapsulation in JavaScript before the introduction of class.



**26. What is an anonymous function?**

An anonymous function is a function **without a name.** Mostly, it is useful for three purposes.

We can **store** an anonymous function **in a variable.**

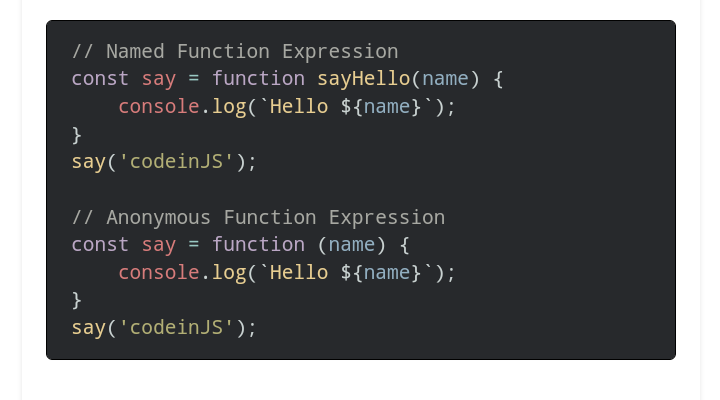
It can be **passed as an argument** to another function.

Similarly, we can **return an anonymous function from a function.**

**27. What is a function expression and function Statements?**

It is possible to store a function into a variable in JavaScript. The entire term is known as a function expression. We can call the variable to execute the stored function.

And function Statements is simple as function Definition.



**28. What is an arrow function?**

An arrow function is a just syntactic sugar over function expression. It is also termed a fat arrow function. It is the shortest syntax to declare a function.



**29. What is the first-class function?**

A programming language is said to have a first-class function if the function in that language is treated like any other variable. In such a language, we can pass the function to another function (callback). We can store a function into a variable. Moreover, we can return a function from another function (closure).

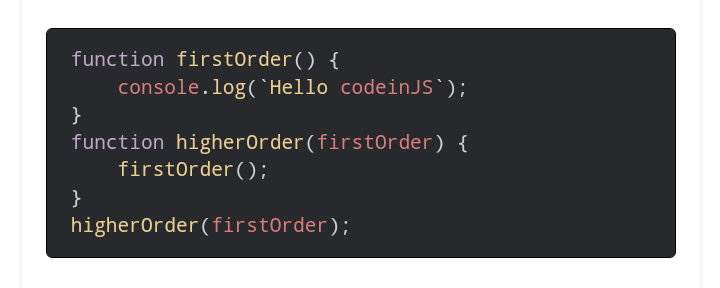
**30. What is the first-order function?**

A first-order function is a function that doesn’t accept another function as an argument. As well as it doesn’t return a function.



**31. What is the higher-order function?**

The higher-order function accepts a function as an argument. Besides that, it also returns the function as a value.



**32. What is the unary function?**

A unary function is a function that accepts exactly one parameter.



**33. What is a pure function?**

A pure function is a function that returns the same output over the input. It never mutates the global variables. A pure function never modifies the global state of variables or any other data members. It only depends on the input arguments of the function.



Some common terminologies for the pure function are:

* A function that doesn’t modify the existing memory location is called a pure function.
* A function that doesn’t modify the actual parameter is called a pure function.
* A mutation is a bad thing. So never modify the actual parameter of the function.

**34. What is IIFE?**

IIFE stands for **Immediately** **Invoked Function Expression.** It starts its execution immediately after the definition. There is a general rule in JavaScript. **Never pollute a global scope**. Writing function in global scope is not a good thing. IIFE creates a function in a local scope.

IIFE prevents access of variables to the outside world. It also prevents pollution of the global scope. Here is an example of IIFE.



**35. What is the currying function?**

Currying is the process of converting a function wit**h n arguments into n number of functions.**

****

**36. What is a nested function?**

The nested function is a function that is defined inside another function.



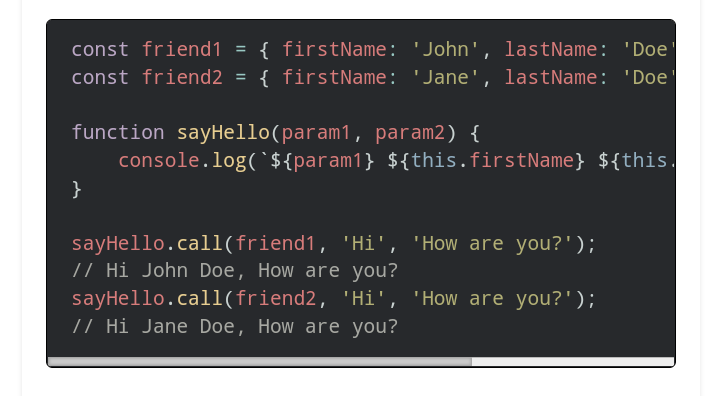
**37. What is the ‘this’ keyword in JavaScript?**

JavaScript runtime sets the context for each function. The **context** is identified by using **this** keyword. The **this** keyword refers to the context where the function is **called**. It does not belong to the context where the function was created.

In an arrow function, this always points to the current surrounding context. It means where the function is created. We cannot change the context of an arrow function.

**38. What is the call() method?**

We can change the context of the function using the call() method. We can explicitly specify where this should refer to.



**39. What is apply() method?**

We can use apply() method to change the context of function similar to like call() method. The only difference is we can pass the parameters as an array.



**40. What is bind() method?**

The bind() method is also useful to change the context of a function. It was introduced in ES5. The bind() method does not execute the function immediately. Instead, it returns a new function with the context.



**41. What is the difference between Call, Apply and Bind?**

|  |  |  |
| --- | --- | --- |
| **call()** | **apply()** | **Bind()** |
| Changes the context of a function (this). | Changes the context of a function (this). | Changes the context of a function (this). |
| Introduced in ES3. | Introduced in ES3. | Introduced in ES3. |
| Parameters can be passed as comma-separated. | Parameters can be passed as an array. | Parameters can be passed as comma-separated. |
| Executes the function immediately. | Executes the function immediately. | Does not execute the function immediately. Instead, return a new function. |

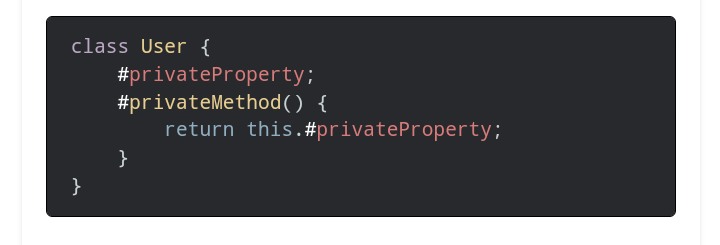
**Javascript interview questions on ES6 features**

**42. What is a class in Javascript?**

The class is a template to create objects. The class is a special function in JavaScript. Basically, it is the syntactic sugar over function prototype. It is introduced in ES6.

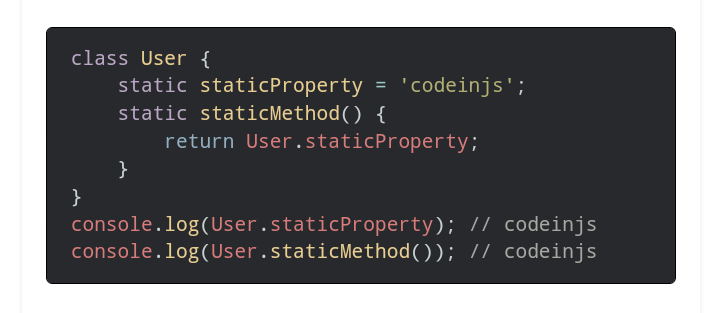
**43. How to create private members in class?**

The class members are public by default. We can create private class members by using the hash (#) prefix.



**44. How to create static members in class?**

We can create static properties and methods in class using the static keyword. We can access static members by using this: className.property or className.method.



**45. What is a template string?**

We can write strings using backticks () in JavaScript. It provides a better way to write multiline strings. Moreover, we can insert a variable inside it using the ${} syntax.



**46. What are the default parameters?**

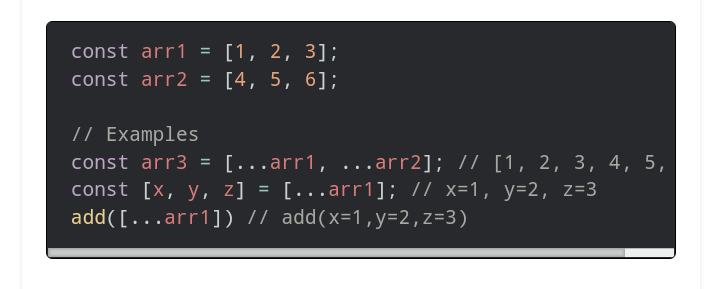
Default parameters allow us to initialize function parameters to default values. They were introduced in ES6. We had to use the logical OR operator for default values prior to default parameters.



**47. What is a spread operator?**

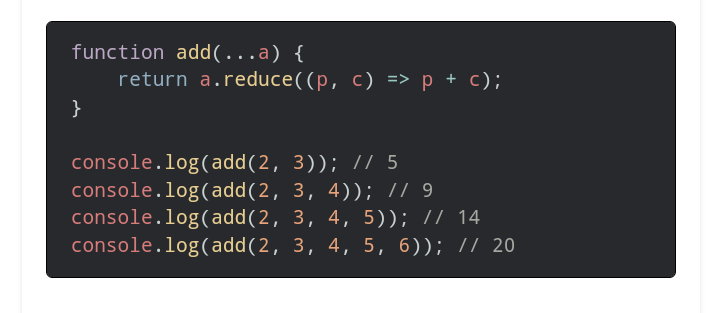
The spread operator expands iterables (array, object, string) into a single value. Here is a list of uses of the spread operator.

* Spreading arrays, objects, and strings into a single value.
* Copying an array.
* Concatenating arrays.
* Spreading element on function call.



**48. What is rest parameter?**

Rest parameter is an improved version of handling function parameters. We can pass any number of parameters to a function. These parameters can be accessed like an array.



**49. What is array destructuring?**

Destructuring in JavaScript is a simplified method of extracting multiple properties from an array by taking the structure and deconstructing it down into its own constituent parts through assignments by using a syntax that looks similar to array literals.

**JavaScript interview questions for frontend**

**50. What is event bubbling?**

The event bubbling is the type of event propagation. In such a case, when we trigger the event of the innermost element, it successively triggers the event of all the parent elements till the outermost parent.

**51. What is event capturing?**

Event capturing is the type of event propagation. In such a case, when we trigger the event of the outermost element, it successively triggers the event of all the child elements till the innermost child.

**52. What is DOM?**

The DOM stands for **Document Object Model**. The DOM defines a standard for accessing documents.

It defines:

* The HTML **elements** as objects.
* The **properties** of all HTML elements.
* The **methods** to access all HTML elements.
* The **events** for all HTML elements.

**53. What is the difference between a document and a window?**

|  |  |
| --- | --- |
| **Document** | **Window** |
| It is a direct child element of a window object. It is also known as Document Object Model (DOM). | It is the root element of the web page. |
| We can access it using window.document or document. | We can access it using the window keyword. |
| It provides methods like getElemenetById(), createElement(), etc. | It provides methods like alert(), confirm(), etc. |

**54. What is a cookie?**

The cookie refers to just the textual information about a website. It is used to remember the user information.

**55. What is session storage?**

The session storage can be useful to store information in the browser. The **sessionStorage** object lets us store the data in key/value pair. It stores the data for one session only.

**56. What is local storage?**

The local storage allows us to store the data in the browser similar to session storage. The local storage data has no expiration time.

**57. What is the difference between a cookie, session storage, and local storage?**

|  |  |  |
| --- | --- | --- |
| **Cookies** | **Session storage** | **Local storage** |
| It is accessible on the client as well as server-side. | It is accessible on the client-side only. | It is accessible on the client-side only. |
| The storage capacity is 4KB. | The storage capacity is 5MB. | The storage capacity is 5MB. |
| We can add the expiration time. | It expires as soon as the tab or browser is closed. | The data is not expired till we clear it. |

**58. What is the preventDefault() method?**

The preventDefault() method stops the browser from executing the default action of the selected element. Here is the syntax.

|  |
| --- |
| **event.preventDefault()** |

**59. How do you access history in Javascript?**

The window.history object contains the browser’s history. We can access the history using back() and forward() methods.

|  |
| --- |
| **window.history.back()** |
| **window.history.forward()** |

**Miscellaneous Javascript interview questions**

**60. What is call stack?**

The call stack is used to keep track of multiple function calls. It is like a real stack in data structures. We use a call stack to memorize which function is running right now.

**61. What is an event loop?**

The event loop is a queue of callback functions. When an async function executes, the callback function is pushed into the queue. The queue places it onto the function execution stack whenever the stack becomes empty.

**62. How can we handle exceptions in Javascript?**

We can use a try-catch block to handle exceptions in JavaScript.



**63. What is the throw keyword?**

The **throw** statement is useful to throw user-defined exceptions. The execution of the code after the throw statement is stopped. And the control is passed to the first catch block.



**64. What is finally block?**

The finally block contains statements to execute after the try block and catch block(s) execute, but before the statements following the try...catch...finally block. Control flow will always enter the finally block

**65. What is JSON?**

JSON stands for JavaScript Object Notation. It is a lightweight text-based medium to store and transmit data across networks. It is generally useful for transmitting and storing data in modern applications.

**66. JSON methods in JavaScript**

JSON offers two methods to work with it.

1. **parse():** The method converts a JSON string to a corresponding JavaScript object or value.
2. **stringify():** The method converts a Javascript object or value to a valid JSON string.

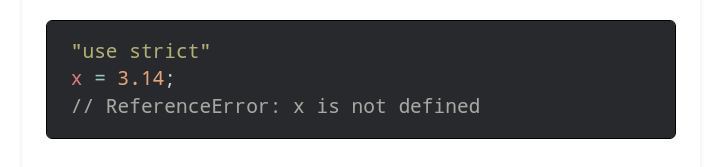
**67. JSON data types**

JSON supports the following data types.

* String
* Number
* Boolean
* Null
* Object
* Array

**68. What is strict mode in JavaScript?**

Strict mode allows writing code in a strict operating context. We can write secure JavaScript code using it. It notifies bad syntax by throwing errors. We can declare strict mode at global as well as function scope.

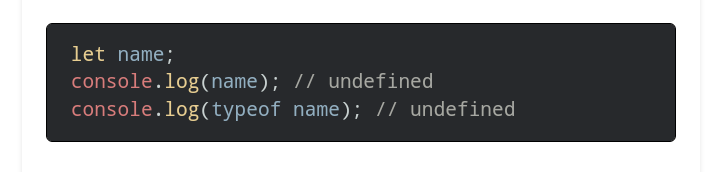


**69. What is NaN?**

NaN stands for **Not a Number**. It represents a value that is not a valid number. We get this value whenever we try to perform mathematical operations on an invalid number.

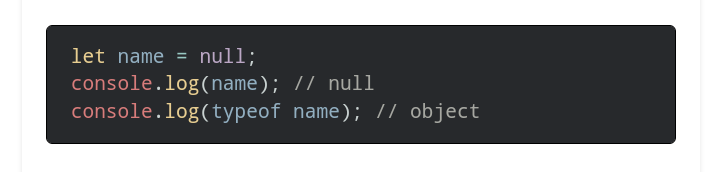
**70. What is undefined?**

The **undefined** property indicates that a **variable has no value or is not declared at all.** The type of undefined is undefined too. We can empty any variable by setting the value to undefined. The function returns undefined if a value is not returned.



**71. What is null?**

The null is one of JavaScript’s primitive values. It represents the **absence of object value.** The type of null is **Object**. We empty a variable by setting the value to null.



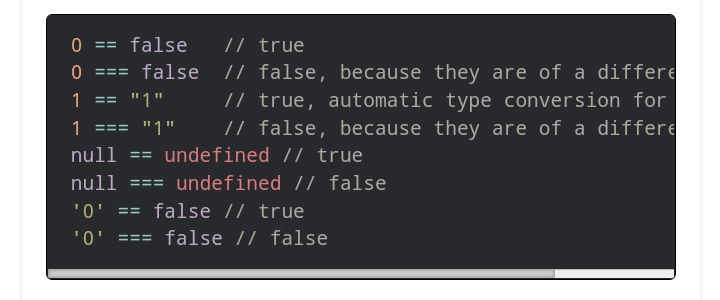
**72. What is the difference between undefined and null?**

|  |  |
| --- | --- |
| **undefined** | **Null** |
| It indicates that a **variable has no value or is not declared at all.** | It represents the **absence of object value.** |
| The type of undefined is **undefined**. | The type of null is **Object**. |
| It represents the absence of a value of a variable. | It represents the absence of a variable itself. |

**73. What is the difference between == and ===?**

The == stands for comparison operator. This operator transforms both the operand to the same type before comparison.

The === stands for strict equality comparison operator. This operator returns false if the operands are not of the same type.



**74. What is typeof operator?**

The typeof operator is useful to check the data type of a variable.

**75. What is call by value?**

We can invoke a function in two ways: call by value or call by reference. While invoking a function, we pass values of variables to it. If the function changes the value of the parameter, then this change does not reflect globally or in the calling function.

Primitive data types are passed to functions by value.

**76. What is call by reference?**

While invoking a function, we pass the address (reference) of variables to it instead of the actual value. If the function changes the value of the parameter, then this change reflects globally or in the calling function.

Object data types are passed to function as a reference.

**77. Shallow copy vs deep copy**

|  |  |
| --- | --- |
| **Shallow copy** | **Deep copy** |
| A shallow copy means copying all the values from the original variable but still keeping connected to the original variable. | Deep copy means copying all the values from the original variable and disconnecting. |
| Certain sub-values can still be connected to original variables. | It disconnects from the original variable and creates a separate copy. |
| Object data types are shallow copied. | Primitive data types are deep copied. |

**78. Which memory does JavaScript use?**

JavaScript uses both stack and heap memory.

**79. Does JavaScript use garbage collection?**

Yes. JavaScript uses the garbage collection technique. Once there is no use of variables, they become ready to be garbage collected. It is taken care of by the JavaScript engine itself.

**80. what is calling function and called function?**

**Calling function:** function which call another function that. Function is known as calling function.

**Called function:** Function has been called by another function that function is known as called function.

While accessing private data from calling function to called function then used parameters.

Visevarsa, used return keyword and stored it to the variable .

