C#Corner

JavaScript Interview Q & A

JavaScript Interview Q & A

Publisher – C# Corner
Editorial Team – Deepak Tewatia, Baibhav Kumar
Publishing Team – Praveen Kumar
Promotional & Media – Rohit Tomar, Rohit Sharma

1. What is JavaScript?

Answer: JavaScript is a high-level, interpreted programming language primarily used for building web applications. It is known for its versatility and ability to create interactive user interfaces.

2. What are the key features of JavaScript?

Answer: Key features of JavaScript include:

- It is a lightweight, interpreted programming language.
- It is designed for creating network-centric applications.
- It supports object-oriented, imperative, and functional programming styles.
- It provides powerful features for manipulating HTML and CSS.
- It supports asynchronous programming through callbacks and promises.

3. What are the differences between JavaScript and Java?

Answer:

Purpose and Context:

- JavaScript is primarily used for client-side web development and increasingly for server-side development (Node.js).
- Java is used for a wide range of applications including server-side development, desktop applications, and Android app development.

Syntax and Structure:

- JavaScript has a lighter syntax and is dynamically typed.
- Java has a more rigid syntax and is statically typed.

Execution Environment:

- JavaScript code runs in a browser's JavaScript engine or on server-side platforms like Node.js.
- Java code runs on the Java Virtual Machine (JVM), requiring compilation to bytecode before execution.

Inheritance and Typing:

- JavaScript uses prototype-based inheritance and dynamic typing.
- Java uses class-based inheritance and static typing.

4. What are closures in JavaScript?

Answer: Closures are functions that have access to variables from their containing lexical scope even after the scope has closed. They "remember" the environment in which they were created.

Example:

```
function outerFunction() {
    let outerVariable = 'I am outer';
    return function innerFunction() {
        console.log(outerVariable);
    };
}
let innerFunc = outerFunction();
innerFunc(); // Output: I am outer
```

5. Explain the concept of prototypal inheritance in JavaScript.

Answer: In JavaScript, objects can inherit properties and methods from other objects. Each object has a prototype object, and when a property or method is accessed on an object, JavaScript looks for it on the object itself first and then on its prototype chain.

Example:

```
let parent = {
    hello: function() {
        console.log('Hello from parent!');
    }
};
let child = Object.create(parent);
child.hello(); // Output: Hello from parent!
```

6. What is the difference between 'null' and 'undefined'?

Answer:

- **Null:** 'null' represents the intentional absence of any object value.
- **Undefined:** 'undefined' represents the uninitialized or undefined value of a variable.

7. What are the features of JavaScript?

Answer:

- **Dynamic Typing:** Variables in JavaScript can hold values of any data type, and their types can change dynamically.
- **First-class Functions:** Functions in JavaScript can be treated as first-class citizens, meaning they can be assigned to variables, passed as arguments, and returned from other functions.
- **Prototypal Inheritance:** Objects in JavaScript can inherit properties and methods from other objects through prototypes.
- **Asynchronous Programming:** JavaScript supports asynchronous operations, allowing for non-blocking code execution through mechanisms like callbacks, promises, and async/await.
- **Event-Driven:** JavaScript is event-driven, enabling interactive web development by responding to user actions and events in the browser.

8. When should generators be used in ES6?

Answer: Generators in ES6 are used for creating iterators with a simpler syntax. They're beneficial for handling asynchronous operations in a synchronous-like manner, iterating over large datasets lazily, implementing custom iteration behaviour, and creating infinite sequences.

9. Why are promises used in JavaScript?

Answer: Promises in JavaScript are used to handle asynchronous operations more easily and elegantly. They provide a cleaner alternative to nested callbacks, improving code readability and maintainability. Promises also to enable better error handling through chaining '.then()' and '.catch()' methods, facilitating asynchronous flow control and avoiding callback hell. Additionally, promises to support composing asynchronous operations, making them essential for modern asynchronous programming in JavaScript.

10. What are the different data types present in JavaScript?

Answer: In JavaScript, there are several data types:

Primitive data types:

- **String:** Represents textual data.
- **Number:** Represents both integer and floating-point numbers.
- **Boolean:** Represents true or false values.
- **Undefined:** Represents a variable that has been declared but has not been assigned a value.
- **Null:** Represents the intentional absence of any object value.
- **Symbol:** Represents a unique identifier.

Non-primitive data types:

- **Object:** Represents a collection of key-value pairs.
- **Function:** A special type of object that can be invoked.

11. What are the main differences between Java and JavaScript?

Answer:

Purpose and Context:

- Java is primarily used for server-side development, desktop applications, and Android app development.
- JavaScript is used for client-side web development, with increasing use in server-side development through platforms like Node.js.

Syntax and Structure:

- Java has strict syntax rules, with static typing and explicit declaration of types, classes, and methods.
- JavaScript has a more flexible syntax, with dynamic typing and a lighter structure allowing for more concise code.

Execution Environment:

- Java code is compiled into bytecode and executed on the Java Virtual Machine (JVM).
- JavaScript code is interpreted by the browser's JavaScript engine or server-side platforms like Node.js.

Typing and Inheritance:

- Java is statically typed and uses class-based inheritance.
- JavaScript is dynamically typed and uses prototype-based inheritance.

12. Explain the rest of the parameters and spread operator.

Answer:

- **Rest parameter:** In JavaScript, the rest parameter allows you to represent an indefinite number of arguments as an array within a function. It is denoted by three dots (‘...’). This parameter gathers all the remaining arguments into an array, making it easy to work with functions that can accept a variable number of arguments.
- **Spread operator:** The spread operator, also denoted by three dots (‘...’), allows an iterable such as an array or string to be expanded into individual elements. It's useful for tasks like passing array elements as arguments to a function, copying arrays, or combining arrays. It essentially "spreads" the elements of an iterable.

13. What is the "this" keyword in JavaScript?

Answer: ‘This’ refers to the context in which a function is executed. It typically represents the object that owns the code being executed.

14. What is Callback in JavaScript?

Answer: A callback in JavaScript is a function that is passed as an argument to another function and is executed after the completion of a specific task or event. It allows for asynchronous programming, enabling functions to continue executing while waiting for an action to finish, such as an API request or a user interaction.

15. Does JavaScript support automatic type conversion?

Answer: Yes, JavaScript supports automatic type conversion, also known as type coercion, which means it can convert data from one type to another when needed, such as during operations or comparisons involving different types.

16. How to find the operating system in the client machine using JavaScript?

Answer: You cannot directly find the operating system using JavaScript due to security restrictions. However, you can infer it indirectly based on user-agent information obtained from the browser.

17. What is NULL in JavaScript?

Answer: In JavaScript, ‘null’ represents the intentional absence of any object value.

18. Differentiate between ViewState and SessionState.

Answer:

ViewState:

- ViewState is used to store values that need to be kept across postbacks on a web page.
- It is stored as a hidden field in the page itself.
- It is scoped to a specific page and keeps its state across postbacks for that page only.

SessionState:

- SessionState is used to store values that need to be kept across multiple requests from the same user.
- It is stored on the server and is accessible across different pages in the web application.
- It is scoped to a specific user session and keeps its state if the session is active.

19. Which character is used to split JavaScript Code spanning into multiple lines?

Answer: The backslash (\) character is used to split JavaScript code spanning into multiple lines.

20. Which is faster, JavaScript or ASP script?

Answer: JavaScript is generally faster than ASP script because JavaScript runs on the client side in the browser, benefiting from optimizations in modern JavaScript engines. ASP script, on the other hand, is executed on the server side and may involve additional processing overhead.

21. What does the prompt box mean in JavaScript?

Answer: The prompt box in JavaScript is a built-in function that displays a dialog box with a message prompting the user to input text.

22. Differentiate between an alert box and a confirmation box.

Answer:

- An alert box displays a message to the user with an OK button.
- A confirmation box displays a message to the user with the OK and Cancel buttons, allowing the user to confirm or cancel an action.

23. What is the use of Void(0)?

Answer: ‘void(0)’ is used to prevent the browser from executing the URL in the href attribute of a link when clicked. It returns ‘undefined’.

24. Explain the purpose of the pop() method.

Answer: The ‘pop()’ method removes the last element from an array and returns that element. It mutates the original array.

25. What are break and continue statements?

Answer:

- ‘break’ is used to exit a loop prematurely.
- ‘Continue’ is used to skip the rest of the loop's current iteration and proceed to the next iteration.

26. Which symbol is used for comments in JavaScript?

Answer: In JavaScript, ‘//’ is used for single-line comments, and ‘/* */’ is used for multi-line comments.

27. What is the use of the isNaN function?

Answer: The ‘isNaN()’ function checks whether a value is NaN (Not-a-Number). It returns true if the value is NaN, and false otherwise.

28. What is negative infinity?

Answer: Negative infinity is a special value in JavaScript representing the mathematical concept of negative infinity, which is smaller than any other real number.

29. Is it possible to break JavaScript Code into several lines?

Answer: Yes, JavaScript code can be broken into several lines using the line continuation character (‘\’) at the end of each line.

30. How do you manipulate the DOM using JavaScript?

Answer: To manipulate the DOM in JavaScript:

- Select elements using methods like ‘getElementById’ or ‘querySelector.’
- Modify content with ‘textContent,’ ‘innerHTML,’ or attributes with ‘setAttribute.’
- Adjust styles using the ‘style’ property or classes with ‘classList.’
- Add or remove elements with ‘createElement,’ ‘appendChild,’ ‘removeChild,’ etc.
- Manage events using ‘addEventListener’ to respond to user interactions.

31. How to submit a form using JavaScript?

Answer: You can submit a form using JavaScript by calling the ‘submit()’ method on the form element.

32. How do you connect Redux with React applications?

Answer: Redux can relate to React applications using the ‘react-redux’ library. This library provides the ‘Provider’ component to wrap the root of the application and the ‘connect’ function to connect React components to the Redux store.

33. How do you handle asynchronous operations in JavaScript?

Answer: Asynchronous operations can be handled using callbacks, promises, or async/await syntax.

34 What is event delegation in JavaScript?

Answer: Event delegation is a technique where you attach a single event listener to a parent element rather than multiple event listeners to individual child elements. This is useful for dynamically created elements or elements with many children.

Example:

```
document.getElementById('parent').addEventListener('click',
function(event) {
    if (event.target.tagName === 'LI') {
        console.log('Clicked on an li element!');
    }
});
```

35. What is closure in JavaScript? Provide an example.

Answer: A closure is the combination of a function and the lexical environment within which that function was declared. Closures allow functions to retain access to variables from their containing scope even after the parent function has finished executing.

Example:

```
function outerFunction() {
  let outerVar = "I am from outer function";
  function innerFunction() {
    console.log(outerVar); // Accessing outerVar from the outer scope
  }
  return innerFunction;
}
let myFunc = outerFunction();
myFunc(); // Output: I am from outer function
```

36. What is the difference between ‘==‘ and ‘===‘ in JavaScript?

Answer: ‘==‘ is the equality operator that performs type coercion, while ‘===‘ is the strict equality operator that checks both value and type without coercion.

Example:

```
console.log(5 == "5"); // true, because of type coercion
console.log(5 === "5"); // false, because types are different
```

37. What is the purpose of the ‘isFinite’ function?

Answer: The **‘isFinite’** function in JavaScript checks whether a value is a finite number. It returns ‘true’ if the value is a finite number, otherwise ‘false’.

38. How do you read and write a file using JavaScript?

Answer: To read and write files in JavaScript, you can use the FileReader API for reading files and the FileWriter API for writing files. Alternatively, in server-side JavaScript environments like Node.js, you can use built-in modules like fs for file system operations.

39. How do you get the status of a CheckBox?

Answer: In JavaScript, you can get the status of a checkbox by accessing its ‘checked’ property. For example, you can use ‘checkbox.checked’ to determine if a checkbox is checked or not.

40. What is the difference between ‘call()’ and ‘apply()’ methods?

Answer: Both ‘call()’ and ‘apply()’ are used to invoke functions with a specific ‘this’ context, but the difference lies in how arguments are passed. ‘call()’ accepts arguments individually, while ‘apply()’ accepts arguments as an array.

41. What is the difference between ‘innerHTML’ & ‘innerText’?

Answer: ‘innerHTML’ retrieves or sets the HTML markup contained within an element, including any HTML tags. ‘innerText’, on the other hand, retrieves or sets only the text content of the element, excluding any HTML tags.

42. Explain event bubbling and event capturing in JavaScript.

Answer: Event bubbling is the propagation of an event from the target element up through its ancestors to the root of the document. Event capturing is the opposite: the event is captured by the outermost element and propagated to the target element.

43. How do you validate an email in JavaScript?

Answer: To validate an email in JavaScript, you can use regular expressions. For example, **‘**/^\w+([\.-]?\w+)*@\w+([\.-]?\w+)*(\.\w{2,3})+$/**’** is a commonly used regular expression to validate email addresses.

44. What is a conditional operator in JavaScript?

Answer: The conditional operator, also known as the ternary operator (**‘**? :**’**), is a shorthand way of writing an if-else statement. It evaluates a condition and returns one of two expressions depending on whether the condition is true or false.

45. Explain the concept of web workers in JavaScript.

Answer: Web workers in JavaScript are a browser feature that enables running scripts concurrently in the background, separate from the main execution thread. They improve performance by handling heavy computational tasks independently, allowing the main thread to remain responsive. Workers communicate with the main thread through message passing and are commonly used for tasks like data processing or intensive calculations in web applications.

46. What is the purpose of the ‘bind()’ method in JavaScript?

Answer: The ‘bind()’ method in JavaScript is used to create a new function that, when called, has a specified ‘this’ value, and optionally, a specified sequence of arguments.

47. Explain the concept of prototypes in JavaScript.

Answer: In JavaScript, every object has a prototype, which serves as a blueprint for that object. Prototypes allow objects to inherit properties and methods from other objects, enabling code reuse and object-oriented programming.

48. How can you clone an object in JavaScript?

Answer: One way to clone an object in JavaScript is by using the ‘Object.assign()’ method or the spread operator (‘...’). These methods create a shallow copy of the object, meaning that nested objects are still referenced.

49. What is implicit type coercion in JavaScript?

Answer: Implicit type coercion in JavaScript is the automatic conversion of values from one data type to another during operations or comparisons, often leading to unexpected results if not handled carefully.

Example:

```
var x = 5;
var y = “10”;
console.log(x  +  y);        // 15
```

50. What is the difference between a function declaration and a function expression in JavaScript?

Answer: A function declaration defines a named function using the **‘**function**’** keyword, which is hoisted to the top of its scope. A function expression, on the other hand, defines a function as part of an expression and can be named or anonymous. Function expressions are not hoisted.

51. What is the difference between ‘let’, ‘const’, and ‘var’?

Answer: ‘let’ and ‘const’ are block-scoped variable declarations introduced in ES6, while **‘**var**’** is function-scoped. Additionally, variables declared with **‘**const**’** cannot be reassigned, whereas variables declared with ‘let’ and ‘var’ can be.

52. Explain hoisting in JavaScript.

Answer: Hoisting is a JavaScript mechanism where variables and function declarations are moved to the top of their containing scope during the compilation phase.

53. How do you handle errors in JavaScript?

Answer: Errors in JavaScript can be handled using ‘try’, ‘catch’, and ‘finally’ blocks. Additionally, you can use the ‘throw’ statement to generate custom errors.

54. How do you filter elements in an array using JavaScript?

Answer: You can use the ‘filter()’ method in JavaScript to filter elements in an array based on a given condition. This method creates a new array containing only the elements that pass the condition.

55. Explain the concept of event-driven programming in JavaScript.

Answer: Event-driven programming is a paradigm where the flow of the program is determined by events such as user actions, system events, or messages from other programs.

56. How do you declare variables in JavaScript?

Answer: In JavaScript, we can declare variables using keywords like ‘var’, ‘let’, or ‘const’. ‘var’ is function-scoped, while ‘let’ and ‘const’ are block-scoped. ‘const’ is used for variables whose value doesn't change, while ‘let’ is for variables that may change. It's generally recommended to use ‘const’ whenever possible.

Example:

```
// Using let
let age = 25;
age = 26; // Valid, age can be reassigned
console.log(age); // Output: 26

// Using const
const pi = 3.14;
// pi = 3.14159; // Error: Assignment to constant variable
console.log(pi); // Output: 3.14
```

57. What are the benefits of using arrow functions in JavaScript?

Answer: Arrow functions provide concise syntax and lexically scoped ‘this’ keyword, making them particularly useful for writing concise and readable code, especially for callbacks and functional programming.

58. How do you handle CORS (Cross-Origin Resource Sharing) in JavaScript?

Answer: CORS can be handled in JavaScript by configuring the server to include the appropriate CORS headers or by using techniques like JSONP or proxy servers.

59. How do you create custom events in JavaScript?

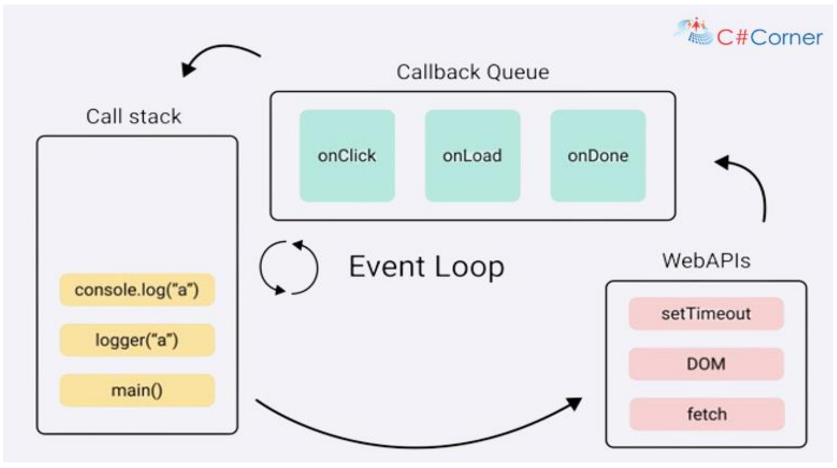
Answer: To create custom events in JavaScript, you can use the ‘CustomEvent’ constructor. Instantiate it with the event type and optional configuration options, then dispatch the event using the ‘dispatchEvent()’ method on the target element.

60. What is the purpose of the ‘String.fromCharCode()’ method in JavaScript?

Answer: The ‘String.fromCharCode()’ method in JavaScript is used to create a string from a sequence of Unicode values.

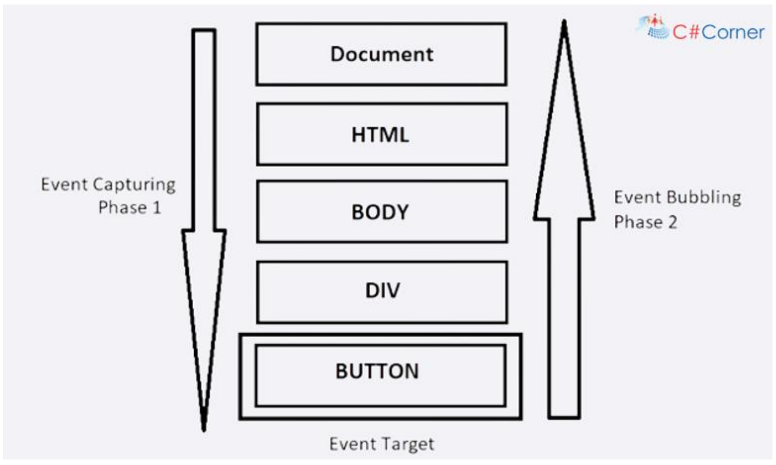
61. What is the event loop in JavaScript?

Answer: The event loop is a mechanism in JavaScript that allows asynchronous tasks to be executed in a non-blocking manner.



62. Explain the concept of event delegation in JavaScript.

Answer: Event delegation is a technique in JavaScript where a single event listener is attached to a parent element to handle events for all of its descendants, reducing the number of event listeners needed and improving performance.



63. What is the purpose of the ‘apply()’ method in JavaScript?

Answer: The ‘apply()’ method in JavaScript is used to call a function with a given ‘this’ value and arguments provided as an array (or an array-like object).

64. How do you convert a string to a number in JavaScript?

Answer: You can convert a string to a number in JavaScript using functions like ‘parseInt()’ or ‘parseFloat()’, or by using the unary plus operator (‘+’).

65. What is a closure in JavaScript? Provide an example.

Answer: A closure is a function that captures and retains references to its outer scope, even after the outer scope has finished executing.

Example:

```
function outer() {
    var outerVar = 'I am outer';
    function inner() {
        console.log(outerVar);
    }
    return inner;
}
var innerFunc = outer();
innerFunc(); // Output: I am outer
```

66. Explain the difference between ‘setTimeout()’ and ‘setInterval()’ in JavaScript.

Answer: ‘setTimeout()’ is used to execute a function once after a specified delay, while ‘setInterval()’ is used to repeatedly execute a function at specified intervals.

67. What is the purpose of the ‘reduce()’ function in JavaScript?

Answer: The ‘reduce()’ function in JavaScript is used to reduce the elements of an array to a single value, applying a provided function from left to right.

68. What are the differences between ‘var’, ‘let’, and ‘const’?

Answer:

- **var:** Function-scoped, hoisted to the top of its function or global scope, can be redeclared and reassigned, not block-scoped.
- **let:** Block-scoped, not hoisted, can be reassigned but not redeclared within the same block.
- **const:** Block-scoped, not hoisted, cannot be reassigned or redeclared after initialization, but the value it holds can still be mutable if it is an object or an array.

69. What is the purpose of the ‘Object.keys()’ method in JavaScript?

Answer: The ‘Object.keys()’ method in JavaScript is used to return an array of a given object's own enumerable property names.

70. What are spread and rest operators in JavaScript?

Answer: Spread (‘...’) and rest (‘...’) operators are both represented by three dots (‘...’). Spread operators are used to expand elements of an array or object into another array or object, while rest operators are used to gather elements of an array into a single variable or parameters of a function into an array.

71. What are the different ways to create objects in JavaScript?

Answer: Objects in JavaScript can be created using object literals, constructor functions, the 'Object.create()' method, or class syntax introduced in ES6.

72. What is the purpose of the 'filter()' function in JavaScript?

Answer: The 'filter()' function in JavaScript is used to create a new array with all elements that pass the test implemented by the provided function.

73. What is the difference between 'slice()' and 'splice()' in JavaScript?

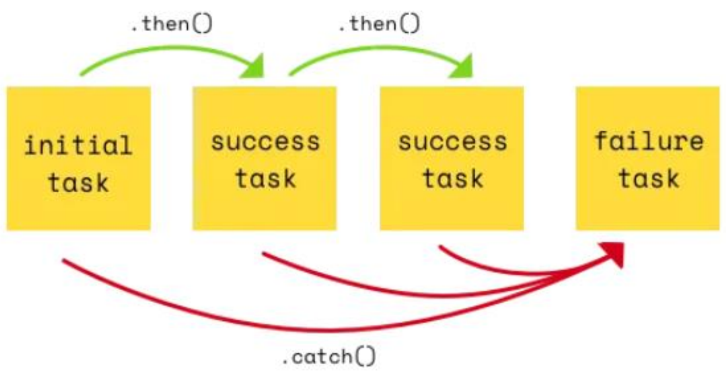
Answer: 'slice()' is used to extract a portion of an array into a new array without modifying the original array, while 'splice()' is used to change the contents of an array by removing or replacing existing elements.

74. Explain the concept of lexical scope in JavaScript.

Answer: Lexical scope is a concept in JavaScript that determines the visibility and accessibility of variables and functions based on their physical location within the source code.

75. Explain the concept of promises in JavaScript.

Answer: Promises in JavaScript are objects that represent the eventual completion (or failure) of an asynchronous operation, allowing you to write asynchronous code in a more readable and manageable way.



76. How do you convert JSON to a JavaScript object and vice versa?

Answer: JSON can be converted to a JavaScript object using 'JSON.parse()', and a JavaScript object can be converted to JSON using 'JSON.stringify()'.

77. What is a higher-order function in JavaScript?

Answer: A higher-order function is a function that either takes one or more functions as arguments or returns a function as its result.

78. What is the purpose of the 'forEach()' function in JavaScript?

Answer: The 'forEach()' function in JavaScript is used to iterate over the elements of an array and execute a provided function once for each element.

79. What is the purpose of the ‘atob()’

Answer: The ‘atob()’ function in JavaScript decodes a base64-encoded string into its original binary data.

Example:

```
let encodedString = 'SGVsbG8gV29ybGQh'; // Base64-encoded string
let decodedString = atob(encodedString); // Decoding the string
console.log(decodedString); // Output: Hello World!
```

80. How do you check if a variable is an array in JavaScript?

Answer: You can check if a variable is an array in JavaScript using the ‘Array.isArray()’ method or by using the ‘instanceof’ operator

81. What is the purpose of the ‘startsWith()’ and ‘endsWith()’ methods in JavaScript strings?

Answer: The ‘startsWith()’ method checks whether a string starts with a specified substring, while the ‘endsWith()’ method checks whether a string ends with a specified substring. Both methods return a boolean value.

82. What is the difference between ‘Object.keys()’ and ‘Object.getOwnPropertyNames()’ in JavaScript?

Answer: ‘Object.keys()’ returns an array of a given object's own enumerable property names, while ‘Object.getOwnPropertyNames()’ returns an array of all own property names (enumerable or not) of a given object.

83. How do you handle errors in asynchronous JavaScript code?

Answer: Errors in asynchronous JavaScript code can be handled using ‘try’, ‘catch’, and ‘finally’ blocks inside asynchronous functions, or by attaching error handlers to promises using the ‘catch()’ method.

84. What are the different ways to create classes in JavaScript?

Answer: Classes in JavaScript can be created using ES6 class syntax, constructor functions, or using the ‘Object.create()’ method to create objects with a specified prototype.

85. What is the purpose of the ‘find()’ and ‘findIndex()’ methods in JavaScript arrays?

Answer: The ‘find()’ method returns the first element in an array that satisfies a provided testing function, while the ‘findIndex()’ method returns the index of the first element that satisfies the provided testing function.

86. What is the purpose of the ‘flatMap()’ method in JavaScript arrays?

Answer: The ‘flatMap()’ method in JavaScript arrays is used to first map each element using a mapping function, then flatten the result into a new array.

87. What are the differences between ‘let’, ‘const’, and ‘var’ in JavaScript?

Answer: ‘let’ and ‘const’ are block-scoped, while ‘var’ is function-scoped. ‘const’ variables cannot be reassigned, but the properties of ‘const’ objects can be mutated.

88. What is the purpose of the ‘includes()’ method in JavaScript arrays?

Answer: The ‘includes()’ method in JavaScript arrays is used to check whether an array contains a specific element, returning ‘true’ if found, and ‘false’ otherwise.

89. Explain the concept of currying in JavaScript.

Answer: Currying is the technique of converting a function that takes multiple arguments into a sequence of functions that each take a single argument. This allows for partial application of functions.

90. Explain the concept of memoization in JavaScript.

Answer: Memoization is an optimization technique used to store the results of expensive function calls and return the cached result when the same inputs occur again, instead of re-executing the function.

91. What is the difference between the ‘for...in’ loop and the ‘for...of’ loop in JavaScript?

Answer: The ‘for...in’ loop iterates over the enumerable properties of an object, while the ‘for...of’ loop iterates over the values of an iterable object, such as arrays, strings, or maps.

92. How do you handle optional chaining in JavaScript?

Answer: Optional chaining is a feature introduced in ES2020 that allows you to access nested properties of an object without the need for explicit null checks. It is denoted by the ‘?.’ syntax.

93. What is the purpose of the ‘Symbol’ data type in JavaScript?

Answer: The ‘Symbol’ data type in JavaScript is used to create unique identifiers for object properties, preventing naming collisions.

94. How do you create and use prototypes in JavaScript?

Answer: To create and use prototypes in JavaScript, define a prototype object with methods and properties, then create new objects using this prototype as a template. Access prototype methods and properties through the created objects. Alternatively, use constructor functions with prototype properties to achieve the same effect.

95. Explain the concept of object prototypes in JavaScript.

Answer: JavaScript object prototypes serve as blueprints for objects, enabling inheritance. They allow objects to inherit properties and methods from their prototype, forming a chain known as the prototype chain. Prototypes facilitate behaviour to reuse and code organization through inheritance.

96. What is the purpose of the ‘Number’ function in JavaScript?

Answer: The ‘Number’ function in JavaScript is used to convert a value to a number. If the value cannot be converted to a number, ‘NaN’ (Not-a-Number) is returned.

97. What is the purpose of the ‘Array.from()’ method in JavaScript?

Answer: The ‘Array.from()’ method in JavaScript is used to create a new shallow-copied array from an array-like or iterable object.

98. What is the purpose of the ‘finally’ block in JavaScript ‘try...catch’ statements?

Answer: The ‘finally’ block in a ‘try...catch’ statement is used to execute code after a ‘try’ block regardless of whether an error was thrown or caught.

99. How do you handle deep nested callback functions in JavaScript?

Answer: Deep nested callback functions can lead to callback hell. To handle this, you can use techniques such as named functions, promises, async/await, or libraries like Async.js.

100. What is the purpose of the ‘Array.isArray()’ method in JavaScript?

Answer: The ‘Array.isArray()’ method in JavaScript is used to determine whether a value is an array. It returns to ‘true’ if the value is an array, otherwise ‘false’.

101. Explain the concept of Web APIs in JavaScript.

Answer: Web APIs (Application Programming Interfaces) are interfaces provided by web browsers that allow JavaScript to interact with the browser and manipulate web page elements, handle events, make HTTP requests, and more.

102. What is the difference between the DOMContentLoaded event and the load event in JavaScript?

Answer: The ‘DOMContentLoaded’ event is fired when the initial HTML document has been completely loaded and parsed, without waiting for stylesheets, images, and subframes to finish loading. The ‘load’ event is fired when the entire page, including all external resources, has finished loading.

103. How do you handle state management in large-scale JavaScript applications?

Answer: State management in large-scale JavaScript applications can be handled using libraries like Redux, MobX, or context API in React, or using custom state management patterns such as Flux architecture.

104. How do you handle form validation in JavaScript?

Answer: Form validation in JavaScript can be handled by attaching event listeners to form elements, validating input values against predefined criteria, and displaying error messages accordingly.

105. What is the purpose of the ‘Intl’ object in JavaScript?

Answer: The **‘**Intl**’** object in JavaScript provides internationalization support, allowing you to format numbers, dates, and strings according to different locales and languages.

106. What is the purpose of the ‘Array.reduce()’ method in JavaScript?

Answer: The ‘reduce()’ method in JavaScript is used to reduce the elements of an array to a single value, applying a provided function from left to right.

107. What are higher-order functions in JavaScript?

Answer: Higher-order functions in JavaScript are functions that can accept other functions as arguments or return functions as results. They enable advanced programming techniques like callbacks, mapping, filtering, and reducing collections of data.

108. What are callbacks in JavaScript?

Answer: Callbacks in JavaScript are functions passed as arguments to other functions, to be executed later, often after an asynchronous operation completes. They enable asynchronous programming and are commonly used for event handling and AJAX requests.

109. Explain the difference between function declarations and function expressions in JavaScript.

Answer: Function declarations are defined using the ‘function’ keyword followed by a name and can be called before they are declared due to hoisting. Function expressions, however, are defined by assigning a function to a variable and cannot be called before their definition.

110. Explain the difference between synchronous and asynchronous code in JavaScript.

Answer: Synchronous code executes sequentially, blocking further execution until each operation completes. Asynchronous code allows operations to execute independently, enabling non-blocking behavior and continuation of program execution while waiting for tasks to finish.

111. Explain the difference between arrow functions and regular functions in JavaScript.

Answer: Arrow functions are concise syntax for writing functions with implicit return and lexical ‘this’ binding. Regular functions have more verbose syntax, explicit ‘return’ statements, and dynamically scoped ‘this’.

112. What are the methods available on promises in JavaScript?

Answer: Promises in JavaScript offer methods like ‘then()’ for handling successful resolution, ‘catch()’ for error handling, and ‘finally()’ for code that should execute regardless of promise outcome.

113. What are template literals in JavaScript? Provide an example.

Answer: Template literals in JavaScript allow embedding expressions and variables within strings using backticks (\’). For example: ‘const name = 'John'; console.log(‘Hello, ${name}!’);’

114. Explain the concept of object destructuring in JavaScript.

Answer: Object destructuring in JavaScript is a convenient way to extract multiple properties from an object and assign them to variables in a single statement.

115. How do you copy an object in JavaScript?

Answer: To copy an object in JavaScript, you can use methods like ‘Object.assign({}, originalObject)’ or the spread operator ‘{ ...originalObject }’.

116. Explain the concept of async/await in JavaScript.

Answer: Async/await in JavaScript is syntactic sugar built on top of promises, simplifying asynchronous code by allowing it to be written in a synchronous-like manner using the ‘async’ and ‘await’ keywords.

117. Explain the concept of strict mode in JavaScript.

Answer: Strict mode in JavaScript enforces stricter parsing and error handling rules, improving code quality and security. It's activated by adding ‘'use strict';’ at the beginning of a script or function.

118. What are the different ways to iterate over objects in JavaScript?

Answer: Different ways to iterate over objects in JavaScript include ‘for...in’ loop, ‘Object.keys()’, ‘Object.values()’, and ‘Object.entries()’ methods.

119. What are the different ways to add properties to an object in JavaScript?

Answer: Properties can be added to an object in JavaScript using dot notation (‘object.property = value’), square bracket notation (‘object['property'] = value’), or ‘Object.defineProperty()’ method.

120. What is the purpose of the ‘fetch’ API in JavaScript?

Answer: The ‘fetch’ API in JavaScript is used to make HTTP requests to servers. It provides a modern, promise-based interface for fetching resources asynchronously across the network. It is commonly used to retrieve data from a server and handle responses in web applications.

121. What is functional programming, and how does it relate to JavaScript?

Answer: Functional programming is a programming paradigm focusing on writing functions that operate on data and avoid mutating state or sharing mutable data. JavaScript supports functional programming with features like first-class functions and higher-order functions.

122. What are pure functions in JavaScript? Why are they important?

Answer: Pure functions in JavaScript are functions that always return the same output for the same input, without side effects. They are important for predictable code behavior, easier testing, and avoiding unintended consequences.

123. Explain the concept of immutability in JavaScript.

Answer: Immutability in JavaScript refers to the concept of not changing the state of data once it's created. Immutable data structures help prevent unintended changes and facilitate easier debugging and reasoning about code.

124. How do you check if an object has a specific property in JavaScript?

Answer: To check if an object has a specific property in JavaScript, you can use the ‘hasOwnProperty()’ method or the ‘in’ operator.

125. What is the purpose of the ‘map’ method in JavaScript? Provide an example.

Answer: The ‘map’ method in JavaScript is used to iterate over an array and transform each element into a new array. It returns a new array without modifying the original one. Example:

126. Explain the concept of function hoisting in JavaScript.

Answer: Function hoisting in JavaScript is a behavior where function declarations are moved to the top of their containing scope during the compilation phase. This means you can call a function before it's declared in your code without encountering an error. However, only function declarations are hoisted, not function expressions.

127. What are the differences between a map and weakmap in JavaScript?

Answer:

- **Map:** in JavaScript is a collection of key-value pairs where both keys and values can be any type. It keeps a strong reference to its keys, meaning they will not be garbage-collected if the map exists.
- **WeakMap:** on the other hand, only accepts objects as keys and keeps weak references to these keys. This means that if there are no other references to a key, it can be garbage-collected even if it is still in the WeakMap.

128. Explain the concept of web workers in JavaScript.

Answer: Web workers are a browser feature that allows scripts to run in background threads, separate from the main execution thread. They enable concurrent processing without blocking the UI, improving performance by handling tasks like complex computations or I/O operations.

129. Explain the concept of JSON in JavaScript.

Answer: JSON (JavaScript Object Notation) is a lightweight data interchange format inspired by JavaScript object literal syntax. It's commonly used for transmitting data between a server and a web application. JSON data is text-based and easy to parse and generate in JavaScript.

130. How do you debug JavaScript code?

Answer: You can debug JavaScript code using browser developer tools like Chrome DevTools or Firefox Developer Tools. Techniques include setting breakpoints, inspecting variables, stepping through code, and using console.log statements.

131. How do you convert JSON data to a JavaScript object?

Answer: You can use the ‘JSON.parse()’ method in JavaScript to convert JSON data into a JavaScript object.

132. How do you iterate over an object's properties in JavaScript?

Answer: You can iterate over an object's properties in JavaScript using a ‘for...in’ loop or by using methods like ‘Object.keys()’, ‘Object.values()’, or ‘Object.entries()’. Each property can then be accessed using the loop or array iteration methods.

133. How do you make an HTTP request using JavaScript?

Answer: You can make an HTTP request in JavaScript using the ‘fetch’ API or XMLHttpRequest object. Here's an example using the ‘fetch’ API:

134. How do you create animations using JavaScript and CSS?

Answer: Animations can be created using JavaScript and CSS by applying CSS animations/transitions to HTML elements and controlling them dynamically with JavaScript. You can use JavaScript to add/remove CSS classes, change CSS properties, or trigger animations based on user interactions or events like scrolling or clicking. CSS provides keyframes, transitions, and animation properties to define the animation behaviour, while JavaScript can handle the logic and interaction aspects.

135. How do you prevent the default behaviour of an event in JavaScript?

Answer: We can prevent the default behaviour of an event in JavaScript using the ‘preventDefault()’ method on the event object. This method is commonly used within event handlers to stop the default action associated with the event, such as submitting a form, following a link, or scrolling a page.

C#Corner

OUR MISSION

ABOUT US

MORE BOOKS