1. **What is JavaScript?**
   * JavaScript is a versatile and widely-used programming language primarily used for adding interactivity to websites. It can be executed in web browsers and on server-side platforms using Node.js.
2. **What are the data types in JavaScript?**
   * JavaScript has several primitive data types: string, number, boolean, null, undefined, and symbol. Additionally, there is the object data type, which includes arrays, functions, and objects.
3. **Explain the concept of closures in JavaScript.**
   * A closure is a function that remembers the variables in the scope where it was created, even if it's executed outside that scope. Closures are commonly used for encapsulation and maintaining state in JavaScript.
4. **What is the difference between null and undefined?**
   * **null** is an intentional absence of any value and is a primitive data type, while **undefined** signifies an uninitialized variable or missing property.
5. **Explain the concept of prototypal inheritance.**
   * In JavaScript, objects can inherit properties and methods from other objects via their prototypes. Each object has a prototype object, and if a property or method is not found on the object itself, JavaScript looks for it in the prototype chain.
6. **How does event delegation work?**
   * Event delegation is a technique where you add a single event listener to a common ancestor of multiple elements, rather than attaching listeners to each individual element. This improves performance and simplifies event handling, especially for dynamically generated elements.
7. **What is the purpose of the this keyword in JavaScript?**
   * **this** refers to the current execution context. Its value depends on how a function is called. In a method, it refers to the object that the method belongs to. In a simple function, it refers to the global object (e.g., **window** in browsers).
8. **Explain the same-origin policy and how it relates to AJAX requests.**
   * The same-origin policy is a security feature that restricts web pages from making requests to a different domain than the one that served the web page. AJAX requests (XMLHttpRequest or Fetch API) must adhere to this policy.
9. **What are promises in JavaScript?**
   * Promises are a way to handle asynchronous operations in a more readable and manageable manner. They represent a value that might be available now, in the future, or never.
10. **How does ES6's let, const, and block-scoping differ from var?**
    * **let** and **const** are block-scoped declarations, which means they are confined to the nearest curly braces **{}** enclosing them. **var** has function scope or global scope, which can lead to unintended variable hoisting issues.
11. **What is the Event Loop in JavaScript?**
    * The Event Loop is a core concept in JavaScript's concurrency model. It's responsible for managing the execution of code, handling asynchronous tasks, and maintaining the responsiveness of applications. It allows non-blocking execution by queuing tasks in the "callback queue" and executing them when the call stack is empty.
12. **Explain hoisting in JavaScript.**
    * Hoisting is a behavior in JavaScript where variable and function declarations are moved to the top of their containing scope during compilation, regardless of where they are defined. This means you can use variables and functions before they are declared, although they won't be assigned their values until their actual declarations.
13. **What are the differences between null and undefined?**
    * While both represent absence of value, **null** is a deliberate assignment indicating no value, while **undefined** typically means a variable has been declared but hasn't been assigned a value. They have different types (**null** is an object, **undefined** is of type **undefined**) and behave differently in certain operations.
14. **Explain the concept of callback functions and provide an example.**
    * Callback functions are functions passed as arguments to other functions to be executed at a later time, often asynchronously. They're commonly used in scenarios like handling asynchronous operations, such as when making AJAX requests or using the **setTimeout** function.
15. **What is the purpose of the bind, call, and apply methods?**
    * These methods are used to control the value of the **this** keyword when invoking a function. **bind** creates a new function with a specific **this** context, while **call** and **apply** immediately invoke the function with a specified **this** context. **call** takes individual arguments, while **apply** takes an array of arguments.
16. **What are arrow functions, and how do they differ from regular functions?**
    * Arrow functions are a more concise way to define functions in JavaScript, introduced in ES6. They have a shorter syntax and, unlike regular functions, do not bind their own **this**. Instead, they inherit the **this** value from their containing function or context.
17. **Explain the concept of Promises and how they help in handling asynchronous operations.**
    * Promises are a way to handle asynchronous operations in a more structured and readable manner. They represent a value that might be available now, later, or never. Promises allow you to chain multiple asynchronous operations and handle success or failure using **.then()** and **.catch()**.
18. **What is the purpose of the async and await keywords in JavaScript?**
    * The **async** keyword is used to define asynchronous functions, which return Promises. The **await** keyword can be used inside an **async** function to pause execution until a Promise is resolved. This makes asynchronous code look more like synchronous code and is easier to read and understand.
19. **Explain the concept of closures and provide an example.**
    * A closure is a function that retains access to variables from its containing scope even after that scope has finished executing. Closures are often used to create private variables or to encapsulate functionality. An example is creating a counter using a closure to retain the count value.
20. **What are modules in JavaScript, and how do they improve code organization?**
    * Modules are a way to encapsulate and organize code in separate files. They prevent polluting the global namespace by keeping variables and functions private within the module unless explicitly exported. The introduction of ES6
21. **Basic Concepts and Fundamentals:**
    * Questions about data types, variables, and basic operators.
    * Explaining **null** vs. **undefined**.
    * Understanding truthy and falsy values.
    * Difference between **let**, **const**, and **var**.
22. **Functions:**
    * Explaining function declarations vs. function expressions.
    * Understanding the concept of scope and closures.
    * Using callback functions and their purpose.
    * Arrow functions and their benefits.
23. **Asynchronous JavaScript:**
    * Understanding callbacks and their role in handling asynchronous operations.
    * Promises, including chaining and error handling.
    * The **async** and **await** keywords.
    * The Event Loop and how it manages asynchronous tasks.
24. **Object-Oriented Programming:**
    * Creating and working with objects.
    * Prototypal inheritance and the prototype chain.
    * Constructors and the **new** keyword.
    * ES6 classes and their advantages.
25. **DOM Manipulation and Events:**
    * Selecting and modifying DOM elements.
    * Event handling and delegation.
    * Understanding the concept of the Event Object.
    * Modifying CSS classes and attributes dynamically.
26. **Web APIs and AJAX:**
    * Fetch API and making HTTP requests.
    * Working with JSON data.
    * Same-Origin Policy and Cross-Origin Resource Sharing (CORS).
    * Handling errors and responses from APIs.
27. **ES6 and Modern JavaScript:**
    * Destructuring assignments.
    * Spread and rest operators.
    * Template literals and tagged templates.
    * Modules (**import** and **export** statements).
28. **Testing and Debugging:**
    * Using **console** methods for debugging.
    * Explaining common debugging techniques.
    * Writing assertions and unit tests using frameworks like Jest.
29. **Coding Problem or Algorithm:**
    * Solving a coding challenge to assess problem-solving skills.
    * Evaluating the candidate's approach to algorithmic problems.
30. **Specific Frameworks or Libraries (if applicable):**
    * Questions related to frameworks like React, Angular, or Vue.js, if mentioned in the job description.
    * Redux or state management questions (for React interviews, for instance).
    * Questions about popular libraries like Axios for HTTP requests.