Q.47 What is the drawback of declaring methods directly in JavaScript objects?

One drawback of declaring methods directly in JavaScript objects is that each object instance will have its own copy of the method, leading to increased memory usage. It's more memory-efficient to define methods in the prototype of the object.

Q.50 What is the drawback of declaring methods directly in JavaScript objects?

One drawback of declaring methods directly in JavaScript objects is that each object instance will have its own copy of the method, leading to increased memory usage. It's more memory-efficient to define methods in the prototype of the object.

Q.54 Form Validation in JS?

Form validation in JavaScript involves checking user input in HTML forms to ensure it meets specified criteria before submission. This can include checking for empty fields, valid email formats, numeric values, password strength, etc. Validation is often done using event handlers like onSubmit or onBlur, along with conditional statements and regular expressions.

Q.58 What is BOM vs DOM in JavaScript?

BOM (Browser Object Model) and DOM (Document Object Model) are two different models in JavaScript:

* BOM deals with browser-specific objects like **window**, **navigator**, **history**, **screen**, etc., which provide functionalities related to the browser environment.
* DOM deals with the structured representation of HTML or XML documents, allowing JavaScript to access and manipulate document elements like nodes, elements, attributes, etc.

In summary, BOM is related to browser functionalities, while DOM is related to document structure and manipulation.

Q.59 Array vs Object differences in JS?

Arrays and objects are both data structures in JavaScript, but they serve different purposes:

* Arrays: Ordered collections of elements accessed by numeric indices. They are suitable for storing lists of data and provide methods like **push**, **pop**, **splice**, etc., for manipulation.
* Objects: Unordered collections of key-value pairs where keys are strings (or symbols in ES6+). They are suitable for representing structured data and allow access to values using keys.

Q.66 What is JavaScript?

JavaScript is a high-level, interpreted programming language primarily used for creating interactive and dynamic content on web pages. It is widely used for front-end development, including client-side scripting, user interface interactions, and browser automation. JavaScript can also be used on the server-side (Node.js) for back-end development.

Q.67 What is the use of isNaN function?

The **isNaN()** function in JavaScript is used to determine whether a value is NaN (Not-a-Number). It returns true if the value is NaN; otherwise, it returns false. It's commonly used to check if a value is a valid number.

Q.68 What is negative Infinity?

Negative Infinity is a special value in JavaScript representing the mathematical concept of negative infinity, i.e., a value lower than any other number (including negative numbers). It's typically the result of operations like dividing by zero or exceeding the lower limit of a numeric type.

Q.69 Which company developed JavaScript?

JavaScript was developed by Netscape Communications Corporation, with Brendan Eich creating the initial version in 1995. It was originally named LiveScript but later renamed JavaScript to leverage the popularity of Java.

Q.70 What are undeclared and undefined variables?

* Undeclared variables: These are variables that have not been declared using **var**, **let**, or **const**. Accessing an undeclared variable will result in a ReferenceError.
* Undefined variables: These are variables that have been declared but not assigned a value. By default, JavaScript initializes variables to undefined until they are assigned a value.

Q.72 What is the difference between ViewState and SessionState?

* ViewState: ViewState is a client-side state management technique used in ASP.NET to store and track the state of controls on a web page between postbacks. It is stored as a hidden field and sent back and forth between the client and server with each request.
* SessionState: SessionState is a server-side state management technique used to store user-specific data throughout their session on a web application. It stores data in server memory or external storage and is accessible across multiple pages for a particular user session.

Q.73 What is the === operator?

The **===** operator in JavaScript is called the strict equality operator. It compares two values for equality without type conversion. It returns true if the operands are equal in value and data type; otherwise, it returns false.

Q.76 What are all the looping structures in JavaScript?

JavaScript supports several looping structures:

1. for loop
2. while loop
3. do-while loop
4. for...in loop (for iterating over object properties)
5. for...of loop (for iterating over iterable objects like arrays)

Q.79 What are all the types of Pop-up boxes available in JavaScript?

JavaScript provides three types of pop-up boxes:

1. Alert box: **alert('Message')**
2. Confirm box: **confirm('Are you sure?')**
3. Prompt box: **prompt('Enter your name:')**

Q.82 What are the disadvantages of using innerHTML in JavaScript?

The disadvantages of using **innerHTML** in JavaScript include:

1. Security risk: It can expose your application to cross-site scripting (XSS) attacks if the content is not properly sanitized.
2. Performance overhead: Manipulating **innerHTML** can be slower than DOM methods like **createElement** and **appendChild**, especially for large operations.
3. Potential loss of event handlers: Replacing **innerHTML** can remove event handlers attached to existing elements.

Q.84 What are the differences between var, let, and const in JavaScript?

* **var**: Declares a variable with function scope (or global scope if declared outside a function). It can be reassigned and re-declared within its scope.
* **let**: Declares a block-scoped variable that can be reassigned but not re-declared within the same block.
* **const**: Declares a block-scoped variable that cannot be reassigned or re-declared after initialization. However, its properties can be modified for objects and arrays.

Q.85 What are the differences between null and undefined in JavaScript?

* **null**: Represents an intentional absence of any value. It is explicitly assigned to indicate that a variable has no value.
* **undefined**: Represents a variable that has been declared but not initialized, or a property that does not exist in an object. It is automatically assigned by JavaScript.

Q.86 What are the differences between a function declaration and a function expression in JavaScript?

* Function declaration: Declares a named function using the **function** keyword, and the function is hoisted to the top of its scope. Example: **function myFunction() {}**
* Function expression: Assigns a function to a variable or property, and the function is not hoisted. Example: **const myFunction = function() {};**

Q.90 What is the difference between a shallow copy and a deep copy in JavaScript?

* Shallow copy: A shallow copy of an object/array creates a new object/array but does not recursively clone nested objects/arrays. Changes made to nested elements in the copy affect the original, and vice versa.
* Deep copy: A deep copy creates a completely independent copy of an object/array, including all nested objects/arrays. Changes made to the copy do not affect the original, and vice versa.

Q.91 What is the use of the **this** keyword in JavaScript?

The **this** keyword in JavaScript refers to the context in which a function is executed. Its value depends on how a function is called:

1. In a global context or outside of any function, **this** refers to the global object (e.g., **window** in browsers, **global** in Node.js).
2. In a method (function inside an object), **this** refers to the object that owns the method.
3. In a constructor function (used with **new** keyword), **this** refers to the newly created instance of the object.
4. In event handlers or callback functions, **this** refers to the element that triggered the event or the context specified during callback execution.
5. **What is JavaScript?** JavaScript is a programming language commonly used for creating interactive and dynamic web content. It is often used for client-side scripting in web development.
6. **What is the use of isNaN function?** The isNaN function is used to determine whether a value is NaN (Not a Number). It returns true if the value is NaN, otherwise false. It is helpful for validating numerical inputs.
7. **What is negative Infinity?** Negative Infinity is a special value in JavaScript representing a number that is smaller than any other real number. It is the result of mathematical operations that yield a value that is too small to be represented.
8. **Which company developed JavaScript?** JavaScript was developed by Netscape Communications Corporation, primarily by Brendan Eich, in the early 1990s.
9. **What are undeclared and undefined variables?** Undeclared variables are those that have been referenced in code but have not been declared using the var, let, or const keywords. Undefined variables are those that have been declared but not assigned a value, resulting in a value of undefined.
10. **Write the code for adding new elements dynamically?** To add new elements dynamically in JavaScript, you can use methods like createElement to create the element, appendChild to add it to the DOM, and setAttribute to set attributes if needed.
11. **What is the difference between ViewState and SessionState?** ViewState and SessionState are concepts related to ASP.NET web applications. ViewState stores the state of the page and its controls between postbacks, while SessionState stores user-specific data across multiple pages during a user session.
12. **What is === operator?** The === operator in JavaScript is the strict equality operator. It checks whether two values are equal in both value and type. It returns true if the values are exactly equal, otherwise false.
13. **How can the style/class of an element be changed?** The style or class of an element can be changed in JavaScript using the style property to directly manipulate CSS properties or by adding/removing classes using the classList property or className property.
14. **How to read and write a file using JavaScript?** Reading and writing files using JavaScript is typically done in the context of server-side JavaScript (Node.js) rather than client-side JavaScript. In Node.js, you can use modules like fs (File System) to read and write files.
15. **What are all the looping structures in JavaScript?** The looping structures in JavaScript include for loop, while loop, do...while loop, and for...in loop (for iterating over object properties) and for...of loop (for iterating over iterable objects like arrays).
16. **How can you convert the string of any base to an integer in JavaScript?** To convert a string of any base to an integer in JavaScript, you can use parseInt() function with the base as the second argument. For example, parseInt('1010', 2) converts the binary string '1010' to its decimal equivalent.
17. **What is the function of the delete operator?** The delete operator in JavaScript is used to delete an object's property or an element from an array. It does not delete variables or functions, only properties.
18. **What are all the types of Pop up boxes available in JavaScript?** The types of pop-up boxes available in JavaScript are alert (to display a message), confirm (to ask for user confirmation with OK and Cancel buttons), and prompt (to ask for user input with an input field).
19. **What is the use of Void (0)?** The use of Void(0) in JavaScript is often seen in href attributes of anchor tags to create "clickable" elements that do not perform any action when clicked. It prevents the browser from following the link or executing the default action.
20. **How can a page be forced to load another page in JavaScript?** A page can be forced to load another page in JavaScript using methods like window.location.href or window.open to navigate to a new URL or open a new window/tab with a specified URL.
21. **What are the disadvantages of using innerHTML in JavaScript?** The main disadvantage of using innerHTML in JavaScript is that it can potentially lead to security vulnerabilities like XSS (Cross-Site Scripting) if not handled properly. Manipulating innerHTML directly with user-generated content can expose your application to malicious scripts. It's recommended to use safer methods like textContent or createTextNode for inserting text content into elements.