JAVASCRIPT ASSIGNMENT

Q.5 What is Condition Statement?

-> A condition statement in programming is a construct that allows the code to make decisions based on whether a condition (or set of conditions) is true or false. It enables the program to execute specific code blocks depending on the outcome of an evaluation.

Q.21 What are the looping structures in JavaScript? Any one Example?

->JavaScript provides several looping structures to execute a block of code repeatedly until a specified condition is met. These are:

1. For loop
2. While loop
3. Do-while loop

Example:

for (let i = 1; i <= 5; i++) {

console.log(i);

}

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

->Declaring methods directly in JavaScript objects can lead to potential drawbacks depending on the context of use. Here are the primary issues:

**1. Redundant Method Definitions**

If multiple instances of an object are created, each instance will have its own copy of the method, leading to redundant memory usage.

**2. Inefficient Memory Usage**

Declaring methods directly in objects can lead to increased memory consumption, especially in applications with many object instances.

**3. Difficulty in Reuse**

If a method needs to be reused across multiple object types or instances, you would need to duplicate it across objects. This makes the code less modular and harder to maintain.

**4. Lack of Inheritance**

Methods declared directly in objects are not easily extendable or inheritable. This makes it difficult to create a hierarchy or share behavior between objects.

Q.54 Form Validtion in JS?

-> Form validation in JavaScript ensures that the data entered by users into a form meets the required criteria before it is sent to the server. This helps improve data quality and user experience by catching errors early.

Q.57 how many type of JS Event? How to use it ?

-> JavaScript events are actions or occurrences that happen in the browser and can be used to execute specific pieces of code. Events are typically triggered by user interactions (e.g., clicking a button) or by the browser (e.g., page load).

1. **Mouse Events**
   * click: Triggered when an element is clicked.
   * dblclick: Triggered on a double click.
   * mouseover: Triggered when the mouse pointer is over an element.
   * mouseout: Triggered when the mouse pointer leaves an element.
   * mousedown: Triggered when the mouse button is pressed.
   * mouseup: Triggered when the mouse button is released.
2. **Keyboard Events**
   * keydown: Triggered when a key is pressed.
   * keyup: Triggered when a key is released.
   * keypress: (Deprecated) Triggered when a key is pressed.
3. **Form Events**
   * submit: Triggered when a form is submitted.
   * change: Triggered when the value of an input, select, or textarea changes.
   * focus: Triggered when an element gains focus.
   * blur: Triggered when an element loses focus.
4. **Window Events**
   * load: Triggered when the entire page (including images and scripts) loads.
   * unload: Triggered when the user leaves the page.
   * resize: Triggered when the browser window is resized.
   * scroll: Triggered when the user scrolls the document.
5. **Touch Events** (For touch screens)
   * touchstart: Triggered when a finger touches the screen.
   * touchend: Triggered when a finger is lifted from the screen.
   * touchmove: Triggered when a finger moves across the screen.
6. **Drag and Drop Events**
   * dragstart: Triggered when an element starts being dragged.
   * drag: Triggered while an element is being dragged.
   * dragend: Triggered when an element is released.
7. **Media Events**
   * play: Triggered when media starts playing.
   * pause: Triggered when media is paused.
   * ended: Triggered when media playback ends.
8. **Clipboard Events**
   * copy: Triggered when content is copied.
   * cut: Triggered when content is cut.
   * paste: Triggered when content is pasted.

Q.60 What is Bom vs Dom in JS?

-> In JavaScript, BOM (Browser Object Model) and DOM (Document Object Model) are two essential concepts that deal with the interaction between JavaScript and the web browser. They serve different purposes but often work together in web development.

BOM : Represents the browser environment.

|  |
| --- |
| Handles browser-level operations. |

|  |
| --- |
|  |
| Not standardized (browser-specific). | |

|  |
| --- |
|  |
| Deals with browser features like navigation, alerts, etc. | |
| DOM :   |  | | --- | | Represents the document structure. |  |  | | --- | |  | | Handles web page content and layout. | |  |  | | --- | |  | | Standardized by W3C. | |  |  | | --- | |  | | Deals with HTML elements and their attributes. | |  |  | | --- | |  |   Q.61 Array vs object defences in JS?  **Array vs Object Differences in JavaScript**   | **Aspect** | **Array** | **Object** | | --- | --- | --- | | **Definition** | An ordered collection of values (elements) indexed by numerical keys. | A collection of key-value pairs, where keys can be strings or symbols. | | **Usage** | Typically used to store lists or sequential data. | Used to represent entities with properties and behaviors. | | **Access** | Accessed using numeric indices (e.g., array[0]). | Accessed using keys (e.g., object.key or object["key"]). | | **Indexing** | Automatically indexed numerically starting from 0. | Keys are explicitly defined and can be any valid string or symbol. | | **Order** | Maintains the order of elements. | Does not guarantee order of properties (though modern engines often maintain insertion order). | | **Methods** | Includes built-in methods for array manipulation, such as push, pop, map, filter. | Limited built-in methods for objects, like Object.keys(), Object.values(). | | **Best Use Case** | When dealing with ordered data or collections of similar items. | When dealing with structured data or representing objects with named properties. | | **Performance** | Optimized for numerical index operations. | Optimized for key-value lookups. |   **1. Arithmetic Operators**  These are used to perform mathematical operations.   | **Operator** | **Description** | **Example** | **Result** | | --- | --- | --- | --- | | + | Addition | 5 + 3 | 8 | | - | Subtraction | 5 - 3 | 2 | | \* | Multiplication | 5 \* 3 | 15 | | / | Division | 10 / 2 | 5 | | % | Modulus (Remainder) | 10 % 3 | 1 | | ++ | Increment | let x = 5; x++ | 6 | | -- | Decrement | let x = 5; x-- | 4 |   **2. Assignment Operators**  These are used to assign values to variables.   | **Operator** | **Description** | **Example** | **Result** | | --- | --- | --- | --- | | = | Assign | x = 5 | x = 5 | | += | Add and assign | x += 3 | x = x + 3 | | -= | Subtract and assign | x -= 2 | x = x - 2 | | \*= | Multiply and assign | x \*= 4 | x = x \* 4 | | /= | Divide and assign | x /= 2 | x = x / 2 | | %= | Modulus and assign | x %= 3 | x = x % 3 |   **3. Comparison Operators**  These are used to compare two values and return a boolean (true or false).   | **Operator** | **Description** | **Example** | **Result** | | --- | --- | --- | --- | | == | Equal to | 5 == '5' | true | | === | Strict equal to | 5 === '5' | false | | != | Not equal to | 5 != '5' | false | | !== | Strict not equal to | 5 !== '5' | true | | > | Greater than | 5 > 3 | true | | < | Less than | 5 < 3 | false | | >= | Greater than or equal to | 5 >= 5 | true | | <= | Less than or equal to | 5 <= 4 | false |   **4. Logical Operators**  These are used to perform logical operations.   | **Operator** | **Description** | **Example** | **Result** | | --- | --- | --- | --- | | && | Logical AND | (true && false) | false | | ` |  | ` | Logical OR | | ! | Logical NOT | !(true) | false |   **2. Difference Between == and === in JavaScript**  **== (Equality Operator)**   * Compares two values for equality **after converting their types** (type coercion). * Example:   javascript  CopyEdit  5 == '5'; // true, because '5' is converted to a number before comparison  **=== (Strict Equality Operator)**   * Compares two values for equality **without converting their types** (no type coercion). * Example:   javascript  CopyEdit  5 === '5'; // false, because the types (number and string) are different  **Summary:**   * == checks for value equality with type conversion. * === checks for both value and type equality.   Top of Form  Bottom of Form | |
| **What is Control Flow in JavaScript?**  **Control flow** in JavaScript refers to the order in which statements are executed in a program. By default, JavaScript code runs sequentially, from top to bottom. However, control flow structures like **conditional statements**, **loops**, and **functions** can alter this order, allowing for more dynamic and complex behavior.  **If-Else Statements**  An if-else statement is used to execute code blocks based on a condition. The condition is evaluated as either true or false:   * If the condition evaluates to true, the block inside the if statement is executed. * If the condition evaluates to false, the block inside the else statement is executed.   **2. Switch Statements**  A **switch statement** is used to execute one block of code out of multiple options based on the value of an expression. It’s an alternative to using multiple if-else conditions.  **How it Works**   1. The switch expression is evaluated once. 2. The value of the expression is compared with the values of each case. 3. If a match is found, the code block for that case is executed. 4. Use break to stop further execution after a case block. If break is omitted, the code continues to the next case. 5. The default case (optional) executes if no match is found.   q) while and do while loops:  The do...while statements combo defines a code block to be executed once, and repeated as long as a condition is true.  The do...while is used when you want to run a code block **at least one time**.  **For Loop,** **While Loop**, and **Do-While** **Loop**are different loops in programming. A For loop is used when the number of iterations is known. A While loop runs as long as a condition is true. A Do-While loop runs at least once and then continues if a condition is true.   * The for loop is used when you know in advance how many times you want to execute the block of code. * It iterates over a sequence (e.g., a list, tuple, string, or range) and executes the block of code for each item in the sequence. * The loop variable (variable) takes the value of each item in the sequence during each iteration. * The while loop is used when you don’t know in advance how many times you want to execute the block of code. It continues to execute as long as the specified condition is true. * It’s important to make sure that the condition eventually becomes false; otherwise, the loop will run indefinitely, resulting in an infinite loop. * The do-while loop is similar to the while loop, but with one key difference: it guarantees that the block of code will execute at least once before checking the condition. * This makes it useful when you want to ensure that a certain task is performed before evaluating a condition for continuation. * The loop continues to execute as long as the specified condition is true after the first execution. It's crucial to ensure that the condition eventually becomes false to prevent the loop from running indefinitely, leading to an infinite loop.   Functions in javascript:  Functions in JavaScript allow us to carry out some set of actions, important decisions, or calculations and even make our website more interactive. In this article, we will learn the difference between ‘function declaration’ and ‘function expression’. The similarity is both use the keyword function and the most prominent difference is that the function declaration has a function name while the latter doesn’t have one.  **Function Declaration**   * A function declaration also known as a function statement declares a function with a function keyword. The function declaration must have a function name. * Function declaration does not require a variable assignment as they are standalone constructs and they cannot be nested inside a functional block. * These are executed before any other code. * The function in the function declaration can be accessed before and after the function definition.   **Function Expression**   * A function Expression is similar to a function declaration without the function name. * Function expressions can be stored in a variable assignment. * Function expressions load and execute only when the program interpreter reaches the line of code. * The function in the function expression can be accessed only after the function definition.   Functions can have parameters and return values. A return value is a result of the function's execution. It can be returned to the block of code that called the function and then used as needed. Parameters are the input for a function that are necessary for the it to be executed and produce a result.28 Mar 2024  q) what is an array? How do you declare and initialize an array?  The easiest way to declare and initialize an array of a primitive type such as int in Java is by using the following syntax. int[] myArray = new int[]{1, 2, 3}; This does a few things at once. The int[] myArray says that we want to declare a new variable called myArray that has the type of an array of integers.15 Apr 2023  q) Explain the methods push(), pop(), shift(), and unshift() used in arrays.  push() and pop() are two Array methods in JavaScript. push() is used to add an element to the end of an array, and pop() is used to remove the last element from an array.  In JavaScript, shift() and unshift() are methods that can be used with arrays to manipulate the elements in the beginning of the array. So, to summarize: shift() removes the first element from an array and returns it, while unshift() adds one or more elements to the beginning of an array and returns the new length.    What is the DOM (Document Object Model) in JavaScript? How does JavaScript interact with the DOM?  The Document Object Model (DOM) is a programming interface for web documents. It represents the page so that programs can change the document structure, style, and content. The DOM represents the document as nodes and objects; that way, programming languages can interact with the page.  Explain the methods getElementById(), getElementsByClassName(), and querySelector() used to select elements from the DOM.  getElementById returns a single DOM element whose ID matches your query. getElementsByClassName returns an HtmlCollection - an array-like structure containing the DOM elements that matched your query. You have to iterate through each element in the array to apply your style.12 Aug 2015  What is error handling in JavaScript? Explain the try, catch, and finally blocks with an example.  Error handling in JavaScript plays an important role to make the application stable. Structures such as try-catch blocks, throw statement and finally block are used to deal with unexpected situations and ensure that the application runs correctly.  The code in the try block is executed first, and if it throws an exception, the code in the catch block will be executed. The code in the finally block will always be executed before control flow exits the entire construct.  Why is error handling important in JavaScript applications?  Errors directly affect the user experience because they can create a negative perception about the reliability and quality of the application. Therefore, correct error management increases application reliability and ensures that users have a smooth experience.7 Feb 2024  Error handling is a part of nearly every computer program, but it is rarely the main focus of a program's developers. Nevertheless, correct error handling is important because it can enable a program to recover from abnormal circumstances and continue to function and serve its purpose. | |

|  |
| --- |
|  |