Module (JAVASCRIPT BASIC & DOM) – 4

1. What is Javascript?

Ans. JavaScript is a high-level, dynamic, and interpreted programming language that is primarily used to create and control dynamic website content. Client-Side Scripting: JavaScript is mainly used for client-side scripting, meaning it runs on the user's web browser rather than on the web server.

Versatility: Although it was originally developed for web browsers, JavaScript is now used in many other environments.

Interactivity: JavaScript can change HTML content, modify CSS styles, and perform asynchronous tasks like fetching data from APIs without reloading the webpage.

Integration: It works well with other web technologies such as HTML and CSS.

Event-Driven: JavaScript is event-driven, meaning it can react to user inputs such as mouse clicks, keyboard inputs, or other actions.

Standardization: JavaScript is standardized by the ECMAScript specification, ensuring consistency and interoperability across different web browsers and environments.

2.What is the use of isNaN function?

Ans. The isNaN function in JavaScript is used to determine whether a value is NaN (Not-a-Number). NaN is a special value in JavaScript that represents a value that is not a legal number. The isNaN function helps to identify such values.

Here is how isNaN works:

Syntax: isNaN(value)

value: The value to be tested.

Return Value: The function returns true if the value is NaN, and false otherwise.

How isNaN Works:

When you pass a value to isNaN, it first attempts to convert the value to a number. If the conversion results in NaN, the function returns true. Otherwise, it returns false.

3.What is negative Infinity?

Ans. Negative infinity (−∞) is a concept in mathematics that represents an unbounded quantity that is less than all real numbers. It is used in various branches of mathematics, such as calculus, number theory, and set theory, to describe the idea of a value that decreases without limit.

Negative infinity (−∞) is a concept in mathematics that represents an unbounded quantity that is less than all real numbers. It is used in various branches of mathematics, such as calculus, number theory, and set theory, to describe the idea of a value that decreases without limit.

Key Properties of Negative Infinity:

Comparison with Real Numbers: For any real number 𝑥x, −∞<𝑥

−∞<x. This means negative infinity is less than any finite number.

Use in Calculus:

Limits: Negative infinity is used to describe limits where a function decreases without bound.

Integrals: In definite integrals, negative infinity can serve as a limit of integration, such as in improper integrals where the interval extends to

−

∞

4.Which company developed JavaScript?

Ans. JavaScript was developed by Netscape Communications Corporation. The language was created by Brendan Eich in 1995 while he was working at Netscape. Initially, it was called Mocha, then it was renamed to LiveScript, and finally, it was named JavaScript to align with the popularity of Java at the time, even though the two languages are quite different.

5.What are undeclared and undefined variables?

Ans. In programming, particularly in languages like JavaScript, the terms "undeclared" and "undefined" refer to different states of variables:

Undeclared Variables

Undeclared variables are those that have never been declared in the current scope.

Attempting to access an undeclared variable typically results in a runtime error.

In strict mode ('use strict';), using an undeclared variable will throw a ReferenceError. Undefined Variables

Undefined variables are variables that have been declared but not yet assigned a value.

Accessing an undefined variable does not result in an error but returns the special value undefined.

This can happen if you declare a variable using var, let, or const but do not initialize it.

6.Write the code for adding new elements dynamically?

Ans. document.addEventListener('DOMContentLoaded', (event) => {

const container = document.getElementById('container');

const addButton = document.getElementById('addButton');

addButton.addEventListener('click', () => {

Create a new div element

const newDiv = document.createElement('div');

Add some content to the new div

newDiv.textContent = 'This is a new dynamically added element

Optionally, add some style to the new div

newDiv.style.padding = '10px';

newDiv.style.margin = '5px';

newDiv.style.border = '1px solid black';

Append the new div to the container

container.appendChild(newDiv);

});

});

7.What is the difference between ViewState and SessionState?

Ans. ViewState

Scope:

ViewState is page-specific. It retains values between postbacks on the same page.

Storage Location:

ViewState data is stored in a hidden field within the page itself. It is encoded and sent to the client's browser with each request and response cycle.

Usage:

ViewState is used to persist control values and small amounts of data across postbacks for a single page.

Lifetime:

The data stored in ViewState is available only for the duration of the page lifecycle. It is lost when the user navigates away from the page.

Performance:

ViewState can impact performance if large amounts of data are stored because it increases the page size and the amount of data sent over the network.

Security:

ViewState data is not secure by default as it can be viewed and modified by the client. It can be encrypted for additional security using ViewStateEncryptionMode.

Example Use Case:

Retaining the values of form fields, maintaining control states such as text in a textbox, or the selected value in a dropdown list during postbacks.

SessionState

Scope:

SessionState is user-specific but not tied to a specific page. It can be used to store data that needs to persist across multiple pages within a user’s session.

Storage Location:

SessionState data is stored on the server. Depending on the configuration, it can be stored in memory, a state server, or a database.

Usage:

SessionState is used to store user-specific data that needs to be accessed across different pages, such as user preferences, shopping cart contents, or authentication tokens.

Lifetime:

The data in SessionState persists for the duration of the user's session. The session can expire after a set timeout period of inactivity or can be manually ended by the application.

Performance:

SessionState can impact server performance, especially if a large amount of data is stored for many users, as it consumes server memory or other resources.

Security:

SessionState is more secure compared to ViewState since it is stored on the server and is not exposed to the client.

Example Use Case:

Storing a user’s login information, user preferences, or any data that needs to persist across multiple web pages during the user's session.

8.What is === operator?

Ans. The === operator is a strict equality operator in many programming languages, including JavaScript. It compares two values for equality without performing type conversion. This means that both the value and the type of the operands must be the same for the comparison to return true.

Here's a breakdown of how the === operator works:

Type Comparison: It first checks if the two operands are of the same type. If they are not, the result is false.

Value Comparison: If the types are the same, it then checks if the values are exactly the same. If they are, the result is true; otherwise, it is false.

9.How can the style/class of an element be changed?

Ans. To change the style or class of an element in a web page, you can use JavaScript or directly manipulate the CSS. Here are the common methods:

Using JavaScript

1. Changing Styles Directly with JavaScript

You can change the inline style of an element by accessing its style property. 2. Changing the Class of an Element

Changing the class of an element can be more efficient, especially when you want to apply multiple style changes or manage complex styles. Here are different ways to handle classes Using CSS

1. Using External or Internal CSS

Define styles in an external CSS file or within a <style> block in your HTML file Using jQuery

If you are using jQuery, you can simplify the process of changing styles and classes <!DOCTYPE html>

<html>

<head>

<style>

.newClass {

color: red;

background-color: blue;

font-size: 20px;

}

</style>

</head>

<body>

<div id="myElement">Hello World</div>

<button onclick="changeStyle()">Change Style</button>

<button onclick="changeClass()">Change Class</button>

<script>

function changeStyle() {

var element = document.getElementById("myElement");

element.style.color = "green";

element.style.backgroundColor = "yellow";

element.style.fontSize = "25px";

}

function changeClass() {

var element = document.getElementById("myElement");

element.classList.toggle("newClass");

}

</script>

</body>

</html>

10.How to read and write a file using JavaScript?

Ans. Reading and writing files using JavaScript depends on the environment you are working in, as JavaScript capabilities differ between running in a browser and running in a Node.js environment.

Reading and Writing Files in Node.js

Node.js provides a powerful file system module (fs) to interact with the file system. Writing to a File:

Synchronous method:

javascript

const fs = require('fs');

try {

fs.writeFileSync('path/to/file.txt', 'Hello, World!', 'utf8');

console.log('File has been written');

} catch (err) {

console.error(err);

}

Asynchronous method:

javascript

const fs = require('fs');

fs.writeFile('path/to/file.txt', 'Hello, World!', 'utf8', (err) => {

if (err) {

console.error(err);

return;

}

console.log('File has been written');

});

11.What are all the looping structures in JavaScript?

Ans. JavaScript provides several looping structures to handle repetitive tasks efficiently. Here are the main looping structures available in JavaScript:

1.for Loop:

The for loop is used to repeat a block of code a known number of times. 2.while Loop:

The while loop is used to repeat a block of code as long as a specified condition is true. 3. do...while Loop:

The do...while loop is similar to the while loop, but it guarantees that the block of code is executed at least once before the condition is tested. 4.for...in Loop:

The for...in loop is used to iterate over the enumerable properties of an object. 5. for...of Loop:

The for...of loop is used to iterate over iterable objects 6. foreach Method:

The forEach method is an array method that executes a provided function once for each array element.

12.How can you convert the string of any base to an integer in JavaScript?

Ans. In JavaScript, you can convert a string representing a number in any base to an integer using the parseInt function. The parseInt function takes two arguments: the string to be converted and the base (radix) of the number system. Important Considerations

Invalid Characters: If the string contains characters that are invalid for the specified base,parseInt will stop parsing at the first invalid character.

Leading Whitespace: Leading whitespace in the string is ignored.

Negative Numbers: parseInt can handle negative numbers if they are properly formatted.

Non-Numeric Leading Characters: If the string does not start with a numeric character or a valid sign for the base, parseInt will return NaN.

13.What is the function of the delete operator?

Ans. The delete operator in programming, specifically in languages like C++ and JavaScript, serves different purposes depending on the context.

In C++:

The delete operator is used to free up the memory allocated for an object or an array of objects that were dynamically allocated using the new operator. Here's a basic rundown of its functionality:

Deleting a single object: If a single object was allocated using new, it should be deallocated using delete. Deleting an array of objects: If an array of objects was allocated using new[], it should be deallocated using delete[]. The delete operator calls the destructor for the object being deleted and then releases the memory back to the system. It is important to match each new with a delete and each new[] with a delete[] to avoid memory leaks and undefined behavior.

In JavaScript:

The delete operator is used to remove a property from an object. It does not free memory like in C++, but it does remove the reference to the property within the object, which can indirectly free up memory if no other references to that value exist. Here’s how it works Key points about the delete operator in JavaScript:

It only affects properties directly on the object (not those in the prototype chain).

It can delete dynamically added properties.

It returns true if the property was successfully deleted, or if it didn't exist in the first place. It returns false if the property is non-configurable.

14.What are all the types of Pop up boxes available in JavaScript?

Ans. In JavaScript, there are three main types of popup boxes that can be used to interact with the user. These pop-up boxes are used to display messages, take input from the user, or ask for confirmation. Here are the types of popup boxes available:

Alert Box (alert):

The alert box is used to display a simple message to the user.

It only has an OK button to close the popup. Confirm Box (confirm):

The confirm box is used to ask the user for confirmation.

It displays a message along with two buttons: OK and Cancel.

The function returns true if the user clicks OK, and false if the user clicks Cancel. Prompt Box (prompt):

The prompt box is used to take input from the user.

It displays a message along with a text input field, and OK and Cancel buttons.

The function returns the input value if the user clicks OK, and null if the user clicks Cancel.

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

Ans. The use of void(0) in JavaScript primarily involves creating a hyperlink that does not perform any action or does not navigate to a different page when clicked. This can be useful in scenarios where you want to attach a JavaScript event handler to a link but prevent the default behavior of the link, which is usually to reload the page or navigate to a new URL.

Here are some common use cases and explanations:

Prevent Default Action: By using void(0) in an href attribute, you can create a link that does nothing by default, but allows you to add JavaScript event handlers to perform custom actions. void is a JavaScript operator that evaluates an expression and returns undefined.

void(0) specifically evaluates the expression 0 and returns undefined.

Since undefined does not correspond to any valid URL, the browser effectively ignores the href attribute and does not navigate away from the current page.

Alternatives

Using void(0) is a common practice, but modern JavaScript development often prefers other methods

16.How can a page be forced to load another page in JavaScript?

Ans. In JavaScript, you can force a page to load another page by manipulating the window.location object. Here are a few common methods to achieve this:

Using window.location.href This method sets the URL of the current page to the new URL and effectively redirects the user to that new page.

Using window.location.assign() This method works similarly to window.location.href and changes the current location to the specified URL.

Using window.location.replace() Choosing the Method:

href and assign(): These methods are nearly identical. Use either if you want the redirection to be recorded in the browser's history, allowing the user to navigate back to the original page.

replace(): Use this if you don't want the original page to be saved in the browser's history.

reload(): This is primarily for refreshing the current page, but can be combined with a change to window.location.

17.What are the disadvantages of using innerHTML in JavaScript?

Ans. Security Risks (XSS Attacks):

One of the major drawbacks of innerHTML is its susceptibility to cross-site scripting (XSS) attacks. If user input is directly inserted into the HTML without proper sanitization, it can lead to the execution of malicious scripts.

Performance Issues:

Updating innerHTML causes the browser to reparse and re-render the entire content of the element, which can be inefficient if only a small part of the content needs to be changed. This can lead to performance bottlenecks, especially for large DOM trees or frequent updates.

Loss of Event Handlers:

When innerHTML is used to replace content, any event listeners attached to elements within that content are lost. This happens because the original elements are completely removed from the DOM and replaced with new ones.

Potential for Creating Invalid HTML:

Directly setting innerHTML can result in invalid HTML if the string being assigned is not well-formed. This can lead to unexpected behavior and rendering issues in the browser.

Inconsistent Behavior Across Browsers:

While innerHTML is widely supported, there can be subtle differences in how different browsers handle the parsing and rendering of HTML content set via innerHTML, leading to cross-browser compatibility issues.

Readability and Maintainability:

Embedding HTML directly into JavaScript using innerHTML can make code harder to read and maintain. It can be challenging to track and manage complex HTML structures within JavaScript strings.

Reduced Control Over DOM Structure:

Using innerHTML does not provide fine-grained control over the DOM. For example, appending or modifying specific child elements often requires reworking the entire HTML structure as a string, which is less flexible compared to using DOM manipulation methods like appendChild, removeChild, insertBefore, etc.