Module (JAVASCRIPT BASIC & DOM) - 4

1. What is JavaScript?

Ans. JavaScript is a high-level programming language primarily used for web development. It is a flexible scripting language that enables website creators to include functionality, interactivity, and dynamic behaviour. Both the client side (via a web browser) and the server side (with the aid of tools like Node.js) are capable of running JavaScript.

1. What is the use of isNaN function?

Ans. To determine whether a value is "Not-A-Number" (NaN) or may be transformed into a valid number, the isNaN function is a built-in JavaScript function. The isNaN function is primarily used to determine whether or not a number is a valid numeric value. It only requires one parameter, the value to be tested, and gives a Boolean value indicating whether the value is NaN or not.

Example:- let num1 = 42;

let num2 = "Hello";

console.log(isNaN(num1)); // Output: false console.log(isNaN(num2)); // Output: true

1. What is negative Infinity?

Ans. A number that is infinitely negative is denoted by the JavaScript special value known as Negative Infinity. It is represented by the symbol -Infinity. When a negative number is divided by zero or its limit is taken as a negative number gets closer to negative infinity, the outcome is known as negative infinity. Additionally, it is returned in JavaScript when a value underflows (becomes smaller than the minimum representable value).

Example:- let x = -1 / 0;

console.log(x);

let y = Math.log(0);

console.log(y); // Output: -Infinity

let z = Number.NEGATIVE\_INFINITY;

console.log(z);

1. Which company developed JavaScript?

Ans. JavaScript was created by the now-defunct software business Netscape Communications Corporation. Brendan Eich was in charge of the initial creation of JavaScript, which was first known as "Mocha" and then "LiveScript," in 1995. It was developed as a scripting language for the well-known web browser at the time, Netscape Navigator. Later, Netscape and Sun Microsystems (now Oracle Corporation) collaborated to change the name of the language to "JavaScript" in an effort to capitalise on Java's growing popularity. JavaScript was advertised as a language that could be used in conjunction with Java for client-side scripting on web pages.

1. What are undeclared and undefined variables?

Ans. Undeclared and undefined variables are two different concepts in JavaScript that relate to the use and state of variables within a program.

1. Undeclared Variables: Variables that have not been explicitly declared or defined using the ‘var’, ‘let’, or ‘const’ keywords before to use are referred to as undeclared variables. JavaScript will generate a ReferenceError if you attempt to access a variable that has not been declared. It indicates that the variable is not understood or has not been assigned a value that is inside the scope.

Example:- console.log(x);

1. Undefined Variables: Variables that have been declared but not given a value are known as undefined variables. When a variable in JavaScript is declared without being assigned, its initial value is undefined. It stands for the lack of a significant value.

Example:- let y;

console.log(y);

1. Write the code for adding new elements dynamically?

Ans. <!DOCTYPE html>

<html>

<head>

<title>Dynamically Add Elements</title>

</head>

<body>

<button id="addButton">Add Element</button>

<div id="container"></div>

<script>

// Get the button and container elements

var addButton = document.getElementById("addButton");

var container = document.getElementById("container");

// Counter to track the number of added elements

var counter = 0;

// Event listener for the button click

addButton.addEventListener("click", function() {

// Create a new paragraph element

var newParagraph = document.createElement("p");

// Set the content of the paragraph

newParagraph.textContent = "Element " + counter;

// Append the paragraph to the container

container.appendChild(newParagraph);

// Increment the counter

counter++;

});

</script>

</body>

</html>

1. What is the difference between ViewState and SessionState?

Ans.

|  |  |
| --- | --- |
| ViewState | SessionState |
| ViewState is a client-side state management technique in ASP.NET. | SessionState is a server-side state management technique in ASP.NET. |
| It is used to store the state of controls (such as values, properties, and events) on a web page across postbacks. | It is used to store and retrieve user-specific data across multiple requests and pages within a web application. |
| The ViewState is stored as a hidden field in the web page and is sent back and forth between the server and the client. | SessionState data is stored on the server, and a unique session ID is sent to the client, usually in the form of a cookie or as part of the URL. |
| ViewState is specific to a single web page and is not shared between different pages. | SessionState is not limited to a single web page but can be accessed and shared across different pages within the same session. |
| It is primarily used to maintain the state of controls during postbacks, ensuring that their values are preserved. | It is commonly used to store user-specific information such as login credentials, user preferences, or shopping cart items. |

1. What is === operator?

Ans. The === operator, also known as the strict equality operator, is a comparison operator in JavaScript. It compares two values for equality without performing type coercion.

* + - If the operands are of the same type, strict equality is used by JavaScript to compare their values. For example: 5 === 5 evaluates to true.
    - If the operands are of different types, JavaScript considers them unequal without attempting any type conversion. For example: '5' === 5 evaluates to false because the string '5' is not strictly equal to the number 5.

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

Ans. To change the style or class of an element in JavaScript, you can access the element using its ID, class, or other selectors, and then modify its properties.

1. Changing the style: You can directly modify the inline style of an element using the ‘style’ property. Here's an example of changing the background color of an element with the ID "myElement":

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

element.style.backgroundColor = "red";

1. Adding/removing classes: To add or remove classes from an element, you can use the classList property. It provides methods like ‘add()’, ‘remove()’, and ‘toggle()’ to modify the classes of an element.

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

element.classList.add("newClass"); // Adds the "newClass" to the element

element.classList.remove("oldClass"); // Removes the "oldClass" from the element

element.classList.toggle("active"); // Toggles the "active" class on the element

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

Ans. In JavaScript, you can interact with files using the File API, which provides methods for reading and writing files from the client-side. It's important to remember that the File API doesn't offer direct access to the server's file system and instead primarily works with files chosen by the user through file input forms.

1. To read a file using JavaScript, you can use the ‘FileReader’ object.

Example: // HTML file input element

<input type="file" id="fileInput">

// JavaScript code

var fileInput = document.getElementById('fileInput');

fileInput.addEventListener('change', function(event) {

var file = event.target.files[0]; // Get the selected file

var reader = new FileReader();

reader.onload = function(e) {

var contents = e.target.result; // The file contents

// Do something with the file contents

console.log(contents);

};

reader.readAsText(file); // Read the file as text

});

1. To write a file using JavaScript, the process is not as straightforward because JavaScript running in a web browser has limited access to the local file system. However, you can generate file downloads by creating a data URL and linking it to an ‘<a>’ element with the download attribute.

Example: var content = "This is the file content.";

var filename = "example.txt";

var data = new Blob([content], { type: 'text/plain' });

if (window.navigator.msSaveOrOpenBlob) {

window.navigator.msSaveOrOpenBlob(data, filename);

} else {

var downloadLink = document.createElement('a');

downloadLink.href = window.URL.createObjectURL(data);

downloadLink.download = filename;

document.body.appendChild(downloadLink);

downloadLink.click();

document.body.removeChild(downloadLink);

}

1. What are all the looping structures in JavaScript?

Ans. JavaScript provides several looping structures that allow you to iterate over a set of values or execute a block of code repeatedly. The looping structures in JavaScript include:

1. ‘for’ loop: The ‘for’ loop is the most commonly used looping structure in JavaScript. It repeats a block of code for a specified number of times.
2. ‘while’ loop: The ‘while’ loop repeatedly executes a block of code as long as a specified condition evaluates to true.
3. ‘do...while’ loop: Similar to the ‘while’ loop, the ‘do...while’ loop executes a block of code repeatedly as long as a specified condition evaluates to true. However, the block of code is executed at least once before checking the condition.
4. ‘for...in’ loop: The ‘for...in’ loop iterates over the properties of an object. It executes the code block for each enumerable property of the object.
5. ‘for...of’ loop: The ‘for...of’ loop is used to iterate over iterable objects such as arrays, strings, or collections. It executes the code block for each element in the iterable.
6. How can you convert the string of any base to an integer in JavaScript?

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

Example: var binaryString = "101010";

var decimalNumber = parseInt(binaryString, 2);

console.log(decimalNumber); // Output: 42

1. What is the function of the delete operator?

Ans. The ‘delete’ operator in JavaScript is used to delete an object property or an element from an array. Its function depends on the context in which it is used:

1. Deleting object properties: When used with an object, the ‘delete’ operator removes a property from that object.

Example: var obj = {

name: "John",

age: 30

};

delete obj.age; // Deletes the 'age' property from the 'obj' object

1. Deleting array elements: When used with an array, the ‘delete’ operator removes the specified element from the array. However, it does not reindex the array or change its length. Instead, it sets the value of the specified index to ‘undefined’.

Example: var arr = [1, 2, 3, 4, 5];

delete arr[2]; // Sets the value at index 2 to undefined

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

Ans. In JavaScript, popup boxes are used to display the message or notification to the user. There are several types of pop-up boxes that you can use to interact with users. They are: -

1. Alert: The alert box is used to display the user a brief message. It has an OK button and a message. It is often used for displaying information or notifying the user of something.

Example: alert("This is an alert box!");

1. Confirm: The confirm box is used to get a yes or no response from the user. It contains a message, an OK button, and a Cancel button. You can use it to ask the user for confirmation before performing an action.

Example: var result = confirm("Are you sure you want to delete this item?");

if (result) {

// Delete the item

} else {

// Cancel the deletion

}

1. Prompt: The prompt box is used to get input from the user. It contains a message, a text input field, an OK button, and a Cancel button. You can use it to ask the user for input or to get values from them.

Example: var name = prompt("Please enter your name:");

if (name !== null) {

// Process the entered name

} else {

// User clicked Cancel

}

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

Ans. In JavaScript, the void operator is used to evaluate an expression and return undefined as its result. When void is followed by an expression, it discards the resulting value and always returns undefined. The most common usage of void is with the value 0 as void (0).

The main purpose of using void(0) is to create a hyperlink or button that doesn't have any effect when clicked. It is often used in the href attribute of an anchor (<a>) tag to prevent the browser from navigating to a new page or performing any action.

Example: <a href="javascript:void(0);" onclick="doSomething()">Click Me</a>

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

Ans. To force a page to load another page in JavaScript, you can use the ‘window.location’ object to manipulate the browser's location. There are a few methods you can use depending on your specific requirements:

1. Using ‘window.location.href’: This will immediately redirect the current page to the specified URL.

Example: window.location.href = "https://www.example.com";

1. Using ‘window.location.assign()’: This method also redirects the current page to the specified URL.

Example: window.location.assign("https://www.example.com");

1. Using ‘window.location.replace()’: This method replaces the current page in the browser's history with the specified URL, effectively redirecting to the new page. The user won't be able to navigate back to the original page using the browser's "Back" button.

Example: window.location.replace("https://www.example.com");

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

Ans. While the innerHTML property in JavaScript is a convenient way to manipulate the content of an element, it does have some potential disadvantages:

1. Security risks: Using innerHTML to insert or modify content can expose your application to security vulnerabilities, such as cross-site scripting (XSS) attacks. Untrusted user input may execute unauthorised scripts in material that is being added, potentially resulting in security breaches. Consider using other methods for injecting material, such as textContent or directly generating DOM nodes, or appropriately sanitising and validating user input to reduce this risk.
2. Performance implications: Modifying an element's innerHTML can be computationally expensive, particularly for complex HTML structures or when done frequently. The browser must parse the HTML, update the DOM tree, and start reflows and repaints whenever innerHTML is changed. This can impact the performance and responsiveness of your web page, particularly on slower devices or when dealing with complex layouts. In some cases, it may be more efficient to manipulate specific parts of the DOM using DOM manipulation methods like ‘createElement’ and ‘appendChild’.
3. Potential loss of event listeners or data: When you set the ‘innerHTML’ of an element, any existing event listeners or data associated with the elements inside that element may be lost. This is because setting ‘innerHTML’ replaces the entire content of the element. If you need to preserve event listeners or data, you would need to reattach them after setting the ‘innerHTML’.