Module 6) JAVASCRIPT BASIC & DOM

• What is JavaScript?

-It's a versatile programming language primarily used for building interactive and dynamic content on websites. Originally created to make web pages more interactive, it's now a key component in web development.

JavaScript allows you to add features like user authentication, form validation, and dynamic content updates without requiring the user to refresh the page. It's commonly used in conjunction with HTML and CSS to create modern, interactive web applications.

• What is the use of isNaN function?

-The JavaScript isNaN() Function is used to check whether a given value is an illegal number or not. It returns true if the value is a NaN else returns false. It is different from the Number.isNaN() Method.

Parameter Values: This method accepts a single parameter as mentioned above and described below:

.value: It is a required value passed in the isNaN() function.

Return Value: It returns a Boolean value i.e. returns true if the value is NaN else returns false.

• What is negative Infinity?

The **negative infinity** in JavaScript is a constant value that is used to represent a value that is the lowest available. This means that no other number is lesser than this value. It can be generated using a self-made function or by an arithmetic operation.

Note: JavaScript shows the NEGATIVE_INFINITY value as -Infinity.

Negative infinity is different from mathematical infinity in the following ways:-

.Negative infinity results in -0(different from 0) when divided by any other number.

.When divided by itself or positive infinity, negative infinity return NaN

.Negative infinity, when divided by any positive number (apart from positive infinity) is negative

infinity.

.Negative infinity, divided by any negative number (apart from negative infinity) is positive infinity.

.If we multiply negative infinity with NaN, we will get NaN as a result.

.The product of 0 and negative infinity is Nan.

.The product of two negative infinities is always a positive infinity.

.The product of both positive and negative infinity is always negative infinity.

• Which company developed JavaScript?

-JavaScript was developed by Netscape Communications Corporation.

-It was created by Brendan Eich in 1995 while he was working at Netscape. Initially named "Mocha," it was later renamed to "LiveScript," and finally, due to a partnership with Sun Microsystems, it was named JavaScript to leverage the popularity of Java at that time. Despite the name, JavaScript and Java are distinct and unrelated programming languages.

• What are undeclared and undefined variables?

1.**Undeclared Variable:-**

.An undeclared variable is a variable that has been used in the code without being declared using a var, let, or const keyword.

.In some languages, this might lead to an error, while in others, the interpreter or compiler may create a global variable with that name.

.Using undeclared variables can lead to unexpected behavior and is generally considered bad practice.

Example:-

```
// Undeclared variable

myVar = 10;

// Later in the code, it's being used
```

console.log(myVar); // Outputs 10

2.**Undefined Variable:-**

.An undefined variable is a variable that has been declared but has not been assigned a value.

.In many programming languages, variables are initialized with a default value of `undefined` until a value is explicitly assigned.

.Attempting to use or perform operations on an undefined variable may lead to errors or unexpected behavior.

Example:-

// Declared but not assigned

let myVar;

// Using an undefined variable

console.log(myVar); // Outputs undefined

• Write the code for adding new elements dynamically?

-Javascript is a very important language when it comes to learning how the browser works. Often there are times we would like to add dynamic elements/content to our web pages. This post deals with all of that.

Creation of new element: New elements can be created in JS by using the createElement() method.

syntax:

document.createElement("<tagName>");

// Where <tagName> can be any HTML

// tagName like div, ul, button, etc.

// newDiv element has been created

For Eg: let newDiv = document.createElement("div");

• What is the difference between ViewState and SessionState?

1.ViewState:-

.Scope: ViewState is used to store state information that is specific to a single web page.

.Storage: The information stored in ViewState is maintained on the client side, usually in a hidden field. It's included in the page's HTML and sent back and forth between the client and the server with each request/response.

.Lifetime: ViewState is short-lived and is only available during the lifespan of a single page. Once the user navigates away from the page, the ViewState is lost.

.Usage: ViewState is often used to persist state information between postbacks (round-trips between the client and server) for a specific page. It helps in maintaining the state of controls on the page across postbacks.

2.SessionState:-

.Scope: SessionState is used to store state information that needs to be shared across multiple pages during a user's session.

.Storage: The information stored in SessionState is maintained on the server. It can be stored in-memory, in a separate process, or in a database, depending on the configuration.

.Lifetime: SessionState persists throughout the user's session, which starts when the user accesses the website and ends when they close the browser or their session times out.

.Usage: SessionState is commonly used to store user-specific information, such as user preferences, authentication details, or shopping cart contents, across multiple pages.

• What is === operator?

-The === operator is a strict equality operator in JavaScript. It is used to compare two values for equality without performing type coercion. This means that not only the values being compared must be equal, but they must also be of the same data type.

Here's how it works:

If the operands are of the same type and have the same value, `=== `returns `true`.

If the operands are of different types or have different values, `===` returns `false`.

.Example:-

```
5 === 5       // true, because both operands are of the same type and have the same value
```

5 === '5' // false, because the operands are of different types (number and string)

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

***Changed in style:-**

1.Direct Style Property Modification:

```
// Get the element by its ID
var myElement = document.getElementById('myElementId');


// Change the background color directly
myElement.style.backgroundColor = 'red';
```

2.Using `setAttribute`:

```
// Get the element by its ID
var myElement = document.getElementById('myElementId');


// Set the style attribute directly
myElement.setAttribute('style', 'background-color: blue; color: white;');
```

***Changed in Class:-**

1.Direct Class Property Modification:

```
// Get the element by its ID
var myElement = document.getElementById('myElementId');
// Add a class
myElement.className = 'newClass';
// If you want to append a class without removing existing ones
myElement.className += ' anotherClass';
```

2.Using `classList` Propert:

```
// Get the element by its ID

var myElement = document.getElementById('myElementId');


// Add a class

myElement.classList.add('newClass');


// Remove a class

myElement.classList.remove('oldClass');


// Toggle a class (add if not present, remove if present)

myElement.classList.toggle('toggleClass');
```

• How to read and write a file using JavaScript?

-On the client side, you can't read or write files in JavaScript browsers. The fs module in Node.js may be used to accomplish this on the server-side. It has methods for reading and writing files on the file system that are both synchronous and asynchronous. Let's demonstrate some examples of reading and writing files with the node.js fs module.

The fs.readFile() and rs.writeFile() methods are used to read and write of a file using javascript. The file is read using the fs.readFile() function, which is an inbuilt method. This technique reads the full file into memory and stores it in a buffer.

• What are all the looping structures in JavaScript?

-JavaScript supports several looping structures that allow you to execute a block of code repeatedly. The main looping structures in JavaScript are:

1.For loop:-

The `for` loop is a common loop that repeats a block of code a specified number of times.

```
-for (initialization; condition; iteration) {
    // code to be repeated
```

}

2.While loop:-

The `while` loop repeats a block of code as long as a specified condition is true.

-while (condition) {

    // code to be repeated

}

3.Do-While Loop:-

Similar to the `while` loop, but the block of code is executed at least once, even if the condition is initially false.

-do {

    // code to be repeated

} while (condition);

4.For...in loop:-

Iterates over the enumerable properties of an object, including inherited properties.

-for (variable in object) {

    // code to be repeated

}

5.For...of loop:-

Introduced in ECMAScript 2015 (ES6), the `for...of` loop iterates over the values of an iterable object (arrays, strings, etc.).

-for (variable of iterable) {

    // code to be repeated

}

6.forEach loop:-

Available for arrays, the `forEach` method executes a provided function once for each array

element.

-array.forEach(function(element) {

    // code to be repeated

});

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

• What is the function of the delete operator?

- Delete is comparatively a lesser-known operator in JavaScript. This operator is more specifically used to delete JavaScript object properties.

-The JavaScript **pop()**, **shift()**, or **splice()** methods are available to delete an element from an array. But because of the key-value pair in an object, deleting is more complicated. Note that, the delete operator only works on objects and not on variables or functions.

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

-In JavaScript, there are three types of popup boxes that you can use to interact with the user:

1. Alert Box (`alert`):

- The `alert` box is used to display a message to the user.
- It has only one button ("OK") to acknowledge the message.

2. Confirm Box (`confirm`):

- The `confirm` box is used to ask the user for confirmation.
- It has two buttons ("OK" and "Cancel").

3.Prompt Box(`prompt`):

- The `prompt` box is used to prompt the user to enter some information.
- It has an input field for the user to type in and two buttons ("OK" and "Cancel").

• What is the use of Void (0)?

The use of `void(0)` in JavaScript is often seen in the context of using it as the href attribute in an anchor (`<a>`) tag to create a "void" or "no-operation" link. This is typically done to prevent the page from navigating to a new URL when the link is clicked.

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

- One way to make a page load another page in JavaScript is by using the `window.location` object. You can set the` href `property of this object to the URL of the page you want to load.

• What are the disadvantages of using innerHTML in JavaScript?

- `innerHTML` is a convenient and commonly used property in JavaScript for manipulating the content of HTML elements, it comes with some disadvantages:

1. Security Risks:

Using `innerHTML` to manipulate content with user input can expose your application to cross-site scripting (XSS) vulnerabilities. If you're injecting user-generated content into the HTML using `innerHTML`, you need to ensure that the content is properly sanitized to prevent malicious scripts from being executed.

2.Performance Concerns:

- Manipulating the `innerHTML` of an element involves re-parsing and re-rendering the entire content of that element. If you're frequently updating large portions of the DOM, it can lead to performance issues. In such cases, using other methods like DOM manipulation directly might be more efficient.

3.Event Handlers and References:

- If your HTML contains elements with attached event handlers or references to JavaScript objects, using `innerHTML` to replace or modify the content might inadvertently remove these bindings. You would need to reattach event handlers or re-establish references after using `innerHTML`.

4.Overwrites Existing Content:

- When you use `innerHTML` to set content, it overwrites the existing content of the element. If there are other elements or data within the container that you want to preserve, you'll need to manually handle merging or appending new content.

5.Limited to HTML Markup:

- **`innerHTML`** is designed for HTML content. If you are working with XML or need to manipulate non-HTML content, other methods like **`textContent`** or **`createDocumentFragment`** might be more appropriate.

6.Browser Inconsistencies:

- While **`innerHTML`** is widely supported, there can be slight variations in behavior across different browsers. It's essential to test and ensure consistent behavior, especially when working with more complex HTML structures.

In summary, while **`innerHTML`** is a powerful tool for HTML manipulation, it should be used carefully, especially when dealing with user input or dynamic content. Consider the context and potential security implications, and explore alternative approaches like DOM manipulation methods when necessary.

- Create password field with show hide functionalities