JAVASCRIPT BAISC &DOM—4

1. What is javascript?

🡪 JavaScript is a versatile programming language primarily used for creating dynamic content on web pages. Initially developed by Brendan Eich in 1995, JavaScript has evolved into one of the most popular programming languages globally. It's commonly used for tasks like validating form data, creating interactive elements on websites, building web-based applications,

2)what is the use of isNaN function?

🡪The NaN value is often used to indicate an error condition, such as attempting to perform arithmetic operations on non-numeric values. For example:

var result = 10 / "apple";

console.log(result); // Output: NaN

JavaScript provides the isNaN() function to check if a value is NaN:

console.log(isNaN("hello")); // Output: true

console.log(isNaN(123)); // Output: false

1. What is negative Infinity?

🡪console.log(Number.NEGATIVE\_INFINITY); // Output: -Infinity

console.log(-Infinity === Number.NEGATIVE\_INFINITY); // Output: true

var min = Math.min(-1, -10, -100);

console.log(min); // Output: -100

var max = Math.max(-1, -10, -100);

console.log(max); // Output: -1

var sum = -Infinity + 1;

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

4)which company developed javascript?

🡪JavaScript was developed by Netscape Communications Corporation, specifically by Brendan Eich, in 1995. At that time, Netscape was one of the leading companies in the emerging field of web browsers. JavaScript was initially created under the name "Mocha" and later renamed "LiveScript" before finally being named JavaScript. Despite its name, JavaScript has little to do with the Java programming language; the similarity in name was mainly for marketing purposes, as Java was a popular programming language at the time. Later, JavaScript was standardized under the name ECMAScript by Ecma International, an international standards organization, to ensure its compatibility and interoperability across different web browsers and platforms.

5)what are undeclared and undefined variabls?

🡪 **Undeclared variables**: These are variables that have been used in code without being formally declared using the var, let, or const keywords. When you try to use an undeclared variable, JavaScript will throw a ReferenceError. For example:

console.log(x); // ReferenceError: x is not defined

**Undefined variables**: These are variables that have been declared but have not been assigned a value. In JavaScript, when a variable is declared but not initialized, its value by default is undefined. For example:

var y;

console.log(y); // Output: undefined

Top of Form

Bottom of Form

6)write the code for adding new elemant dynamically?

🡪 <!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Dynamically Add Element</title>

</head>

<body>

<button onclick="addElement()">Add New Element</button>

<script>

function addElement() {

// Create a new element

var newElement = document.createElement("div");

// Set some attributes for the new element (optional)

newElement.textContent = "This is a new element";

newElement.style.backgroundColor = "lightblue";

newElement.style.padding = "10px";

newElement.style.marginTop = "10px";

// Append the new element to an existing element on the page

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

container.appendChild(newElement);

}

</script>

<div id="container">

<!-- Existing elements will be appended here -->

</div>

</body>

</html>

7)what is the difference between viewstate and sessionstate?

🡪 **ViewState**:

* ViewState is used to persist state information of a particular page across postbacks. It's a client-side state management technique.
* The data stored in ViewState is serialized and sent to the client browser as a hidden field within the page.
* ViewState is specific to a single page. Each page maintains its own ViewState data.

**SessionState**:

* SessionState is used to persist state information across multiple page requests within the same session. It's a server-side state management technique.
* The data stored in SessionState is maintained on the server and is associated with a unique session identifier (usually stored in a cookie or URL) that links subsequent requests from the same client to the same session data.

8)what is === operator?

🡪 For example, in JavaScript:

5 === 5 // true, because both values are the same (5) and of the same type (number)

5 === '5' // false, because although the values look the same, they are of different types (number and string)

9)how can the style/ class of an elemant be changed?

🡪 <!DOCTYPE html>

<html>

<head>

<title>Change Element Style with CSS</title>

<style>

/\* Define some styles \*/

.box {

width: 100px;

height: 100px;

background-color: red;

}

.blueBox {

background-color: blue;

}

.bigBox {

width: 200px;

}

</style>

</head>

<body>

<div id="myElement" class="box">Hello</div>

<script>

// JavaScript to add class

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

element.classList.add("blueBox"); // Add a class

element.classList.add("bigBox"); // Add another class

</script>

</body>

</html>

10) how to read and write a file using javascript?

🡪 read file

const fs = require('fs');

// Read a file asynchronously

fs.readFile('example.txt', 'utf8', (err, data) => {

if (err) {

console.error(err);

return;

}

console.log(data); // Log the contents of the file

});

Write file

const fs = require('fs');

// Write to a file asynchronously

const content = "This is the content that will be written to the file.";

fs.writeFile('example.txt', content, err => {

if (err) {

console.error(err);

return;

}

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

});

11)what are all the looping structures in javacript?

🡪 for loop

for (initialization; condition; increment/decrement) {

// code to be executed

}

While loop

while (condition) {

// code to be executed

}

Do while

while (condition) {

// code to be executed

}

12)how can convert the sring of any base to an integer in javacsript?

// Convert a string in any base to an integer

const str = "1010"; // Example string in binary representation

const base = 2; // Base of the number (binary)

const intValue = parseInt(str, base);

console.log(intValue); // Output: 10 (integer representation of the binary string "1010")

13)what is the function of the delet operator?

const obj = {

name: 'John',

age: 30,

city: 'New York'

};

console.log(obj); // Output: { name: 'John', age: 30, city: 'New York' }

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

console.log(obj); // Output: { name: 'John', city: 'New York' }

14) what are all the types of pop boxes available in javascript?

🡪 Alert Box

alert("This is an alert message!");

Confirm Box

if (confirm("Do you want to proceed?")) {

// Proceed with the action

} else {

// Cancel the action

}

Prompt Box

var name = prompt("Please enter your name:", "John Doe");

if (name !== null) {

alert("Hello, " + name + "!");

}

15)what is the use of void(0)?

🡪 The void(0) expression is used in JavaScript to evaluate an expression and return undefined. It's often seen in conjunction with anchor (<a>) tags in HTML to prevent the browser from following the link when clicked.

16)how can a page be forced to load another page in javacript?

🡪in JavaScript, you can force a page to load another page by setting the window.location property to the URL of the page you want to load. This triggers a navigation to the specified URL, effectively loading the new page.

17)what are the force disadvanages of using innerHTML in javascript?

1. 🡪 **Security Risks**: One of the primary concerns with using innerHTML is the potential for introducing security vulnerabilities such as cross-site scripting (XSS) attacks. If the content being inserted via innerHTML contains user input that hasn't been properly sanitized, it could execute malicious scripts.

**2 Performance Overhead**: Manipulating innerHTML can be slower than other DOM manipulation methods, especially when dealing with large amounts of content. This is because setting innerHTML causes the browser to reparse and re-render the entire contents of the targeted element, which can be inefficient.

**3 Event Handlers**: If you use innerHTML to replace or modify HTML content that contains event handlers attached via JavaScript, those event handlers may be lost. This is because setting innerHTML essentially removes all existing DOM nodes within the targeted element and replaces them with new ones, which don't have the same event handlers bound to them.

**4 Loss of Element References**: When you use innerHTML to replace the content of an element, any references to elements inside that element are lost. This means that if you have JavaScript code that references specific elements within the replaced content, those references will become invalid after setting innerHTML.

**5 Browser Inconsistencies**: Although innerHTML is widely supported across modern browsers, there may be inconsistencies in how different browsers handle certain aspects of its usage. This can lead to unexpected behavior and make it harder to write cross-browser compatible code.

Top of Form

Bottom of Form