

**JS Assignment**

1. **What is JavaScript?**

* JavaScript (often abbreviated as JS) is a programming language that along with HTML and CSS, is considered one of the core technologies of the web. It's what makes webpages dynamic and interactive. For example, if you click a button on a webpage and something happens (like a new page loads or content is hidden/shown), that's Possible because of JavaScript.

1. **What is the use of isNaN function?**

* The **isNaN** function in javascript is used to determine whether a value **is** **NaN (Not-a-number)** or not. It returns **true** if the value passed to it **is NaN**, and **false** otherwise. The **isNaN** function is often used to validate user input, especially when the input is expected to be a number, to avoid errors and unexpected behavior in the code.
* For Ex :

Console.log(isNaN(123)); // false

Console.log(isNaN(“hello”)); // true

1. what is negative infinity?

* In javascript, -Infinity represents negative infinity, which is a numeric value that represents a quantity that is less than negative infinity. This value is obtained by dividing a negative number by zero. -Infinity is a special value in the javascript language and is used to represents the result of certain mathematical operations, such as when the result of a calculation is smaller than the smallest representable number.
* For ex :

Console.log(1 / -0) // -Infinity

1. Which company developed JavaScript?

* JavaScript was developed by Netscape Communications Corporation in the mid-1990s as a scripting language for web browsers. It was created by Brendan Eich, who was a programmer at Netscape at the time. JavaScript was initially called Mocha and was later changed to Livescript before finally being named JavaScript. JavaScript quickly became popular due to its versatility and ability to add interactivity and dynamic elements to websites, and it is now an essential technology for front-end web development.

1. What are undeclared and undefined variables?

* In JavaScript, variables must be declared using the var, let, or const keywords before they can be used in the code.
* An undeclared variable is a variable that has not been declared using one of these keywords and is therefore not recognized by the JavaScript interpreter. Attempting to use an undeclared variable will result in a reference error.
* An undefined variable, on the other hand, is a variable that has been declared using one of the keywords, but has not been assigned a value. The value of an undefined variable is undefined.
* For example:

// undeclared variable

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

// undefined variable

var y;

console.log(y); // undefined

1. Write the code for adding new elements dynamically?
2. There are several ways to dynamically add new elements to a web page using JavaScript, here are a few common ones:
3. Using innerHTML property: You can use the innerHTML property to dynamically add new elements to an existing HTML element.

// select the target element

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

// add a new element

target.innerHTML = "<p>This is a new paragraph.</p>";

1. Using the createElement and appendChild methods: You can use the createElement method to create a new HTML element, and the appendChild method to add it to an existing element.

// select the target element

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

// create a new element

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

newElement.innerHTML = "This is a new paragraph.";

// add the new element to the target

target.appendChild(newElement);

1. Using the insertAdjacentHTML method: You can use the insertAdjacentHTML method to add new HTML elements to an existing element.

// select the target element

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

// add a new element

target.insertAdjacentHTML("beforeend", "<p>This is a new paragraph.</p>");

1. What is the difference between ViewState and SessionState?

* **View state :** The values of controls of a particular page of the client browser is persisted by ViewState at the time of post back operation is done. If the user requests another page, the data of previous page is no longer available.
* **SessionState :** The data of a particular server persists in the server by SessionState. The availability of the user data is up to the completion of a session or closure of the browser.

1. What is === operator?

* The === operator in JavaScript is known as the strict equality operator. It compares two values for equality and returns true if the values are equal and false otherwise. Unlike the equality operator ==, the strict equality operator === performs type coercion. That means it compares values after converting both operands to the same type.
* For example:

console.log(1 === 1); // true

console.log("1" === 1); // false

console.log(true === 1); // false

console.log(null === undefined); // false

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

* In JavaScript, you can change the style or class of an HTML element in several ways:

1. Using the style property: You can directly access the style property of an element and set its CSS properties.

// select the element

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

// change the style of the element

element.style.backgroundColor = "red";

element.style.height = "100px";

element.style.width = "100px";

1. Using the className property: You can access the className property of an element and set it to the class name you want to apply. This will change the styles applied to the element using the specified class in your CSS.

// select the element

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

// change the class of the element

element.className = "new-class";

1. Using the setAttribute method: You can use the setAttribute method to add or modify the class attribute of an element.

// select the element

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

// change the class of the element

element.setAttribute("class", "new-class");

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

* Files can be read and written by using java script functions – fopen(),fread() and fwrite()..
* The function fopen() takes two parameters – 1. Path and 2. Mode (0 for reading and 3 for writing). The fopen() function returns -1, if the file is successfully opened.
* Example:

File=fopen(getScriptPath(),0);

The function fread() is used for reading the file content.

* :-Example:

Str = fread(file,flength(file) ;

The function fwrite() is used to write the contents to the file.

* :-Example:

File = fopen(“c:\MyFile.txt”, 3);// opens the file for writing

Fwrite(file, str);// str is the content that is to be written into the file.

1. What are all the looping structures in JavaScript?

* In JavaScript, there are two main looping structures: for loops and while loops.

1. for loop: The for loop is used to execute a block of code repeatedly a specific number of times. The syntax for a for loop is:

for (initialization; condition; iteration) {

// code to be executed

}

Here's an example:

for (var i = 0; i < 5; i++) {

console.log(i);

}

* This will log the numbers 0 to 4 to the console.

1. while loop: The while loop is used to execute a block of code repeatedly as long as a specified condition is true. The syntax for a while loop is:

while (condition) {

// code to be executed

}

Here's an example:

var i = 0;

while (i < 5) {

console.log(i);

i++;

}

* This will log the numbers 0 to 4 to the console.
* Additionally, there's also the do...while loop, which is similar to the while loop, but the code in the loop will be executed at least once, even if the condition is false. The syntax for a do...while loop is:

do {

// code to be executed

} while (condition);

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

* In JavaScript, you can use the parseInt function to convert a string representation of a number in any base to an integer. The parseInt function takes two arguments: the string to be converted, and the base of the string representation (optional, defaults to base 10).
* Here's an example:

const binaryString = "1010";

const decimal = parseInt(binaryString, 2);

console.log(decimal); // Output: 10

* In this example, the binary string "1010" is converted to the decimal number 10 using the parseInt function with a base of 2. The output is then logged to the console.

1. What is the function of the delete operator?

* The delete operator is used in JavaScript to remove properties from objects. The delete operator takes an object property as its operand and, if the property is deletable, removes the property and returns true. If the property is not deletable or doesn't exist, the delete operator returns false.
* **Here's an example**:

let obj = { name: "Diya", age: 19 };

delete obj.age;

console.log(obj); // Output: { name: "Diya" }

delete obj.gender;

console.log(obj); // Output: { name: "Diya" }

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

* In JavaScript, there are several types of pop-up boxes that can be used to interact with users and receive input:

1. alert: The alert box is a simple pop-up window that displays a message to the user. The alert
2. What is the use of Void (0)?

* The void(0) expression is a common technique used in JavaScript to prevent the default behavior of a page element, such as a link. When a link is clicked, it typically causes the browser to navigate to a different page. By using void(0) as the value of the href attribute, the default behavior of the link is cancelled, and the navigation is prevented.
* Here's an example:

<a href="javascript:void(0);" onclick="alert('Hello World');">Click Me</a>

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

* In JavaScript, you can use the window.location object to force a page to load another page. The window.location object has a property called href that can be used to set the URL of the page to be loaded.
* Here's an example:

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

1. What are the disadvantages of using innerHTML in JavaScript?
2. Security Concerns: Using innerHTML to insert dynamic content from untrusted sources can result in security vulnerabilities such as Cross-Site Scripting (XSS).
3. Performance Issues: Modifying the innerHTML property can be slow, especially for larger amounts of content, as the browser must re-render the entire contents of the element.
4. Inconsistent Behavior: Different browsers may have different implementations of the innerHTML property, leading to inconsistent behavior and difficulties with cross-browser compatibility.
5. Difficult to maintain code: Modifying the contents of an element using innerHTML can result in a cluttered and difficult-to-maintain codebase.
6. Lack of Separation of Concerns: Using innerHTML to manipulate the DOM tightly couples the structure of the HTML and the behavior of the JavaScript code, making it harder to maintain and test.