Module (JAVASCRIPT BASIC & DOM) – 4

- **What is JavaScript?**

-> JavaScript is a high-level, versatile, and widely used programming language primarily used for web development. It is a core technology for building interactive and dynamic websites. JavaScript allows developers to add functionality, interactivity, and behavior to web pages, making them more engaging and responsive to user input.

- **What is the use of isNaN function?**

-> The isNaN function in JavaScript is used to check if a given value is "Not-a-Number" (NaN). It returns ‘true’ if the value is ‘NaN’ and ‘false’ if it's a valid number or can be converted to one. It's commonly used for input validation to ensure that a variable holds a numeric value, especially when dealing with user input or calculations to avoid unexpected errors.

Syntax:

- isNan(value)

- **What is negative Infinity?**

-> Negative infinity is the lowest possible numeric value in JavaScript. It is often used to represent values that are unbounded or undefined, typically as the result of mathematical operations that approach negative infinity, such as dividing a negative number by zero.

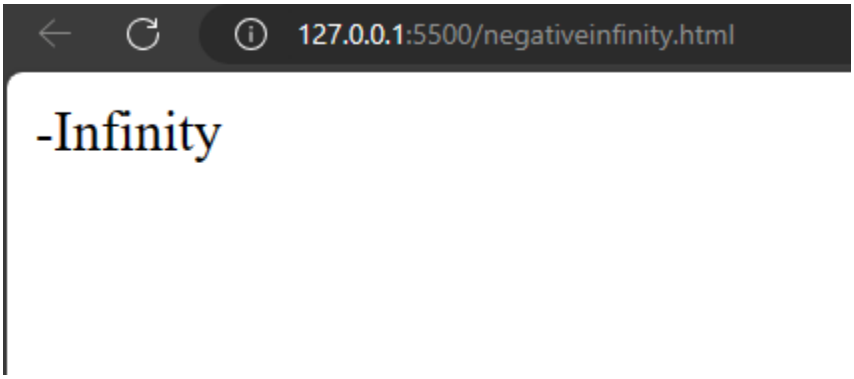
Example:

Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
```

```
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <script>
        var positiveNumber = 10;
        var negativeNumber = -5;
        var result1 = negativeNumber / 0;
        document.write(result1);
    </script>
</body>
</html>
```

Output:



• Which company developed JavaScript?

-> The development of JavaScript was led by Brendan Eich, who created the language in 1995 while working at Netscape Communications Corporation.

• What are undeclared and undefined variables?

-> Undeclared variables: An undeclared variable is one that has not been declared using the ‘var’, ‘let’, or ‘const’ keywords.

Example:

myVariable = 10;

Undefined variables: An undefined variable is one that has been declared but has not been assigned a value.

Example:

var myVariable; // This is a declared but undefined variable

console.log(myVariable);

Output: undefined

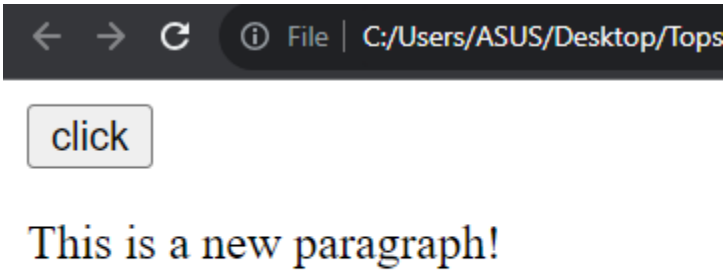
- **Write the code for adding new elements dynamically?**

-> code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <button id="click here">click</button>
    <div id="one"></div>
    <script>
        var Button = document.getElementById("click here");
        var container = document.getElementById("one");
        function addParagraph() {
            // Create a new paragraph element
            var paragraph = document.createElement("p");

            // Create some text content for the paragraph
            var text = document.createTextNode("This is a new paragraph!");

            // Append the text to the paragraph element
            paragraph.appendChild(text);

            // Append the paragraph element to the container
            container.appendChild(paragraph);
        }

        // Add a click event listener to the button
        Button.addEventListener("click", addParagraph);
```

```
    </script>
</body>
</html>
```

Output:



- **What is the difference between ViewState and SessionState?**

ViewState	SessionState
State management is on the client side.	State management is on the server side.
Data is accessible within the same web page.	Data is accessible from other web pages of the website.
Loss of information occurs when different web page is loaded.	Loss of information occurs due to timeout.
Data is stored in a hidden field of the same web page.	Data is stored in sessions and cookies.
It's less secure.	It's more secure.

- **What is === operator?**

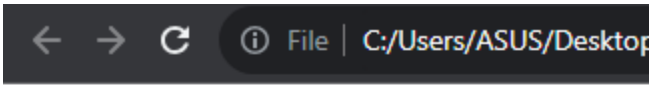
-> The '===' operator is a strict equality operator in JavaScript. It is used to compare two values for equality while taking both the data type and the value into account. This means that not only must the values being compared be the same, but they must also have the same data type.

Example:

Code:

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <p id="one"></p>
    <script>
        var a = 1;
        document.getElementById("one").innerHTML = (a === 1);
    </script>
</body>
</html>
```

Output:



true

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

-> To change the style or class of an HTML element using JavaScript, you can use the DOM (Document Object Model) to access the element and modify its attributes.

Example:

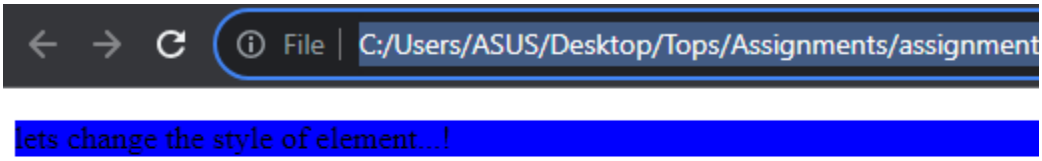
Changing style of element:

Code:

```html
<!DOCTYPE html>
<html lang="en">
<head>
```

```
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <p id="one">
        lets change the style of element...!
    </p>
    <script>
        var element = document.getElementById("one");
        element.style.backgroundColor = "blue";
    </script>
</body>
</html>
```

Output:



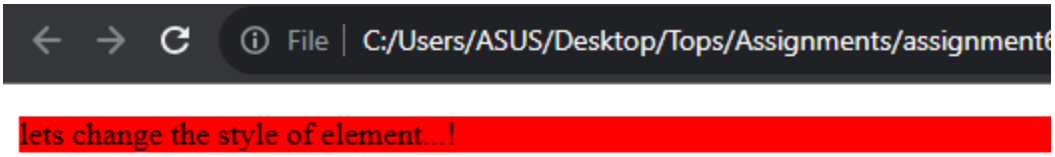
Changing class of element:

Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <style>
        .highlight{
            background-color: red;
        }
    </style>
</head>
<body>
    <p id="one">
        lets change the style of element...!
```

```
    </p>
    <script>
        var element = document.getElementById("one");
        element.classList.add("highlight");
    </script>
</body>
</html>
```

Output:



- **How to read and write a file using JavaScript?**

-> There are two ways to do it:

1. Using javascript extensions (runs from javascript editor)
2. using a webpage and ActiveX objects (internet Explorer only)

- **What are all the looping structures in JavaScript?**

-> For - loops through a block of code a number of times.

-> For/in - loops through the properties of an object.

-> For/of - loops through the value of an iterable object.

-> While - loops through a block of code while a specified condition is true.

-> do-while - Also loops through a block of code while a specified condition is true.

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

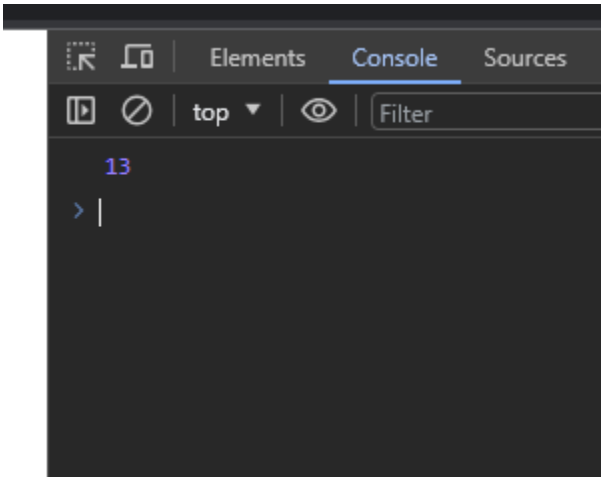
-> In JavaScript, you can convert a string representing a number in any base to an integer (decimal base, base 10) using the ‘parseInt’ function.

Example:

Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <script>
        var a = "1101";
        var b = parseInt(a, 2);
        console.log(b);
    </script>
</body>
</html>
```

Output:



- **What is the function of the delete operator?**

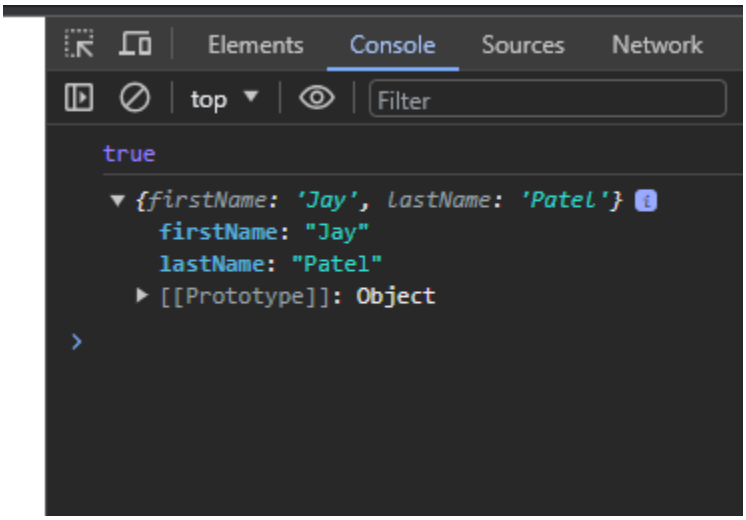
-> The delete operator in JavaScript is used to remove a property from an object. Its primary function is to delete a specific property from an object, which can include object properties, array elements, or object methods.

Example:

Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <script>
    let emp = {
    firstName: "Jay",
    lastName: "Patel",
    salary: 40000
    }
    console.log(delete emp.salary);
    console.log(emp);
    </script>
</body>
</html>
```

Output:



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

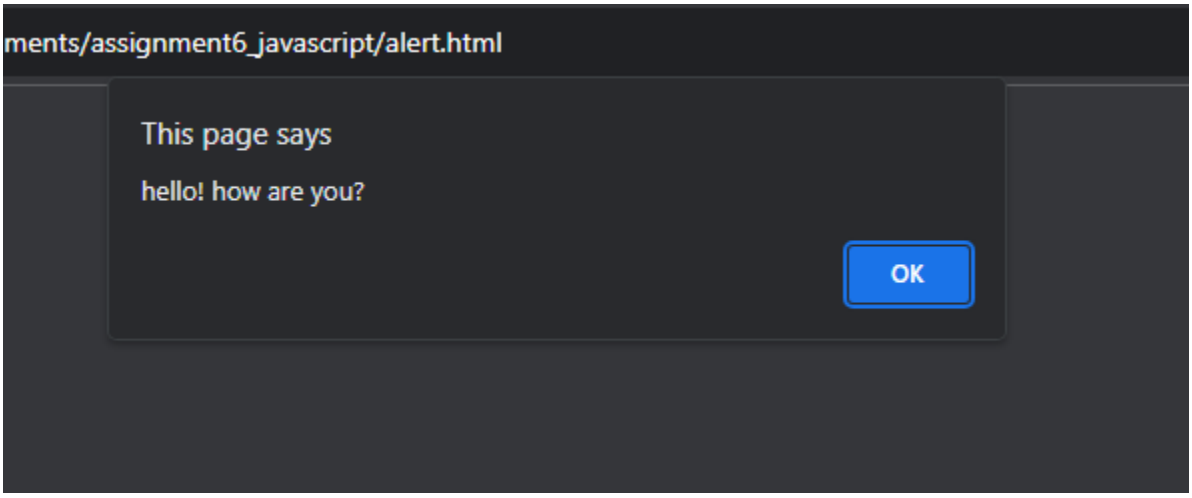
-> Alert Box: The alert dialog displays a message to the user in a pop-up window with an "OK" button. It's typically used to provide information or notifications to the user.

Example:

Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <script>
        alert('hello! how are you?')
    </script>
</body>
</html>
```

Output:



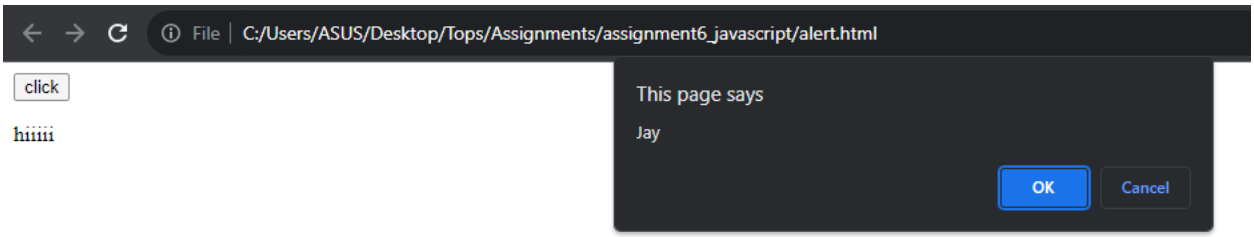
Confirm Box: The confirm dialog is used to ask the user a yes-or-no question. It provides two buttons, typically "OK" and "Cancel," and returns true if the user clicks "OK" and false if they click "Cancel."

Example:

Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <button onclick="a()">click</button>
    <p id="one"></p>
    <script>
        function a() {
            var txt;
            if (confirm("Jay")){
                txt="hiiiii"
            }
            else{
                txt="noooo"
            }
            document.getElementById("one").innerHTML=txt;
        }
    </script>

</body>
</html>
```

Output:



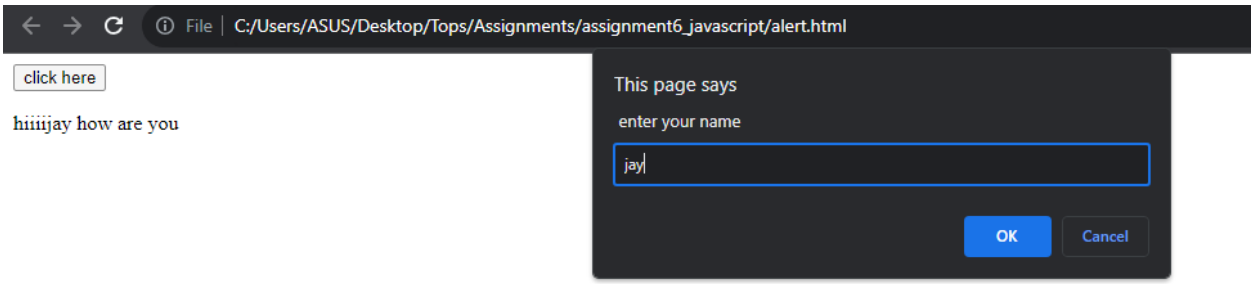
Prompt Box: The prompt dialog allows you to get input from the user. It displays an input field, an "OK" button, and a "Cancel" button. You can use it to collect user data. It returns the user's input as a string, or null if the user clicks "Cancel."

Example:

Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <button onclick="a()">click here</button>
    <p id="one"></p>
    <script>
    function a() {
        var txt;
        var person = prompt (" enter your name","jay")
        if (person== null || person ==" " ) {
            txt= "enter your name"

        } else{
            txt = "hiiii" +person+ " how are you"
        }
        document.getElementById('one').innerHTML=txt;
     }
</script>

</body>
</html>
```

Output:



- **What is the use of Void (0)?**

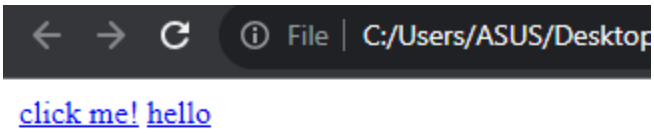
-> void(0) is a construct that is sometimes used in anchor (<a>) elements or links to prevent the browser from navigating to a new page when the link is clicked. It is often used as a placeholder or a way to create "empty" or non-functional links.

Example:

Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <!-- <p>
        <p>click the following, this wont react at all...</p>
    </p>
    <a href="javascript:void(alert('warning!!!'))">click me!</a> -->
    <!-- ********************************************************************** -
->

    <a href="javascript:void(document.write('hello'))">click me!</a>
    <!-- ********************************************************************** -
->

    <a href="javascript:void(0);" onclick="">
    hello
    </a>
</body>
</html>
```

Output:



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

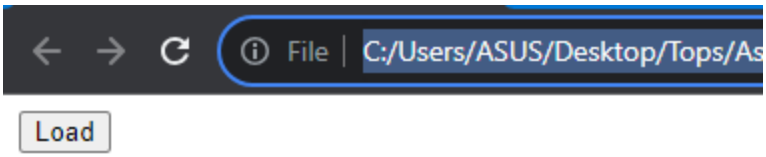
-> we can use window.location property inside the script tag to forcefully load another page in javascript. It is a reference to a location object that is it represents the current location of the document. We can change the URL of a window by accessing it.

Example:

Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <button onclick="forceLoad()">Load</button>
    <script>
        function forceLoad() {
            window.location.href = "https://www.io-tish.com/";
        }
    </script>
</body>
</html>
```

Output:



• **What are the disadvantages of using innerHTML in JavaScript?**

->

Security Risks: When you use innerHTML, you're essentially injecting HTML into your page. If the HTML content is not sanitized properly, it can expose your web application to security vulnerabilities like Cross-Site Scripting (XSS) attacks. Attackers can inject malicious scripts that get executed in the context of your web page.

Performance: When you modify an element's innerHTML, the browser has to re-parse and re-render the entire content within that element. This can be inefficient, especially when dealing with large or complex elements. In contrast, other DOM manipulation methods like createElement and appendChild can be more performant, as they work directly with the DOM structure.

Event Handlers: If you use innerHTML to replace the content of an element, any event handlers attached to child elements within that element will be removed. You need to reattach event handlers after updating innerHTML, which can be error-prone and may lead to unexpected behavior.

Loss of Element References: If you replace an element's innerHTML, any references to child elements stored in variables may become invalid. This can cause issues in your code if you're relying on those references for further interactions with the DOM.

Limited Control: When you use innerHTML, you're essentially replacing the entire content of an element. If you want to update only a part of the content or if you need to perform more granular operations, you may find it challenging to do so using innerHTML.

Non-standard Syntax: While innerHTML is widely supported in modern browsers, it's not part of the official DOM specification. This means that its behavior may not be consistent across different browsers, and it's not guaranteed to work in all contexts.