JAVASCRIPT ASSIGNMENT

1). What is JavaScript?

- JavaScript is a scripting language that enables you to create dynamically updating content, control multimedia, animate images, and pretty much everything else.
- JavaScript is the Programming Language for the Web.
- JavaScript can update and change both HTML and CSS.
- JavaScript can calculate, manipulate and validate data.

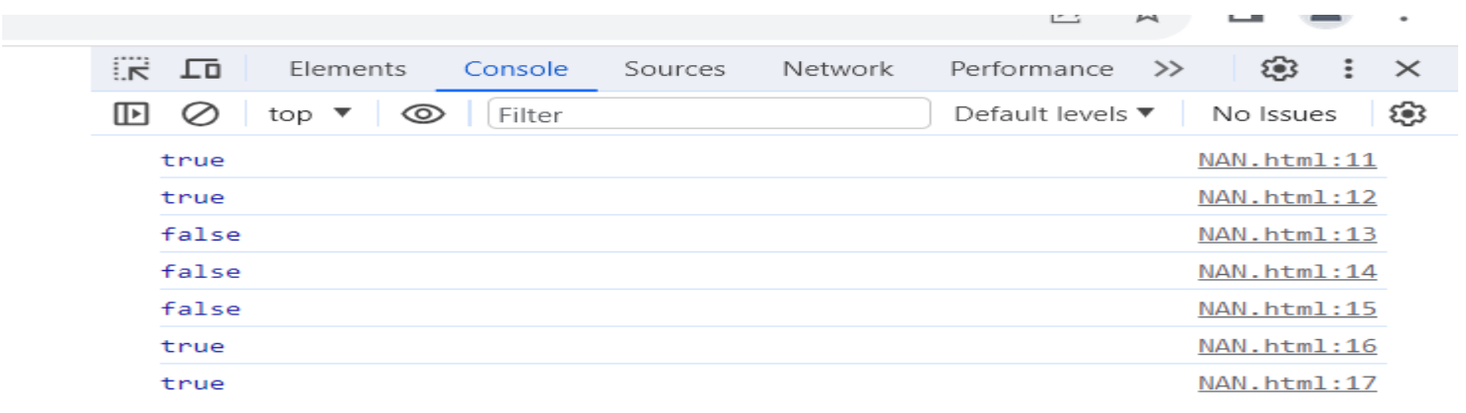
2). What is the use of isNaN function?

- 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.

Example

```
NAN.html > html > body > script
<!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>
    <div></div>
    <script>
        console.log(isNaN("priyanka"));
        console.log(isNaN(0 / 0));
        console.log(isNaN(15));
        console.log(isNaN(24.15));
        console.log(isNaN(-10));
        console.log(isNaN("22/09/2023"));
        console.log(isNaN(NaN));
    </script>
</body>
</html>
```

Output



JAVASCRIPT ASSIGNMENT

3).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.
- 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.

4).Which company developed JavaScript?

- JavaScript was created at Netscape Communications by Brendan Eich in 1995. Netscape and Eich designed JavaScript as a scripting language for use with the company's flagship web browser, Netscape Navigator.

5).What are undeclared and undefined variables?

- **Undefined:** It occurs when a variable has been declared but has not been assigned any value. Undefined is not a keyword.

- **Undeclared**: It occurs when we try to access any variable that is not initialized or declared earlier using the *var* or *const keyword*.

6).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.

- 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");

JAVASCRIPT ASSIGNMENT

Example

newelement.html > html > head > meta

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

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content= "width=device-width, initial-scale=1.0">
</head>

<body>
    <div class="button">
        <button id="addTask">Add task</button>
    </div>
    <div class="task"></div>
    <script type="text/javascript">

        let task = document.getElementsByClassName("task");
        let addTask = document.getElementById("addTask");
        addTask.addEventListener('click', function () {
            for (let i = 0; i < task.length; i++) {
                let newDiv = document.createElement("div");
                newDiv.setAttribute("class", "list");
                newDiv.innerText = "New Div created";
                task[i].append(newDiv);
            }
        })
    </script>
</body>

</html>
```

Output :

Add task

New Div created
New Div created
New Div created
New Div created
New Div created

JAVASCRIPT ASSIGNMENT

7). What is the difference between ViewState and SessionState?

ViewState:

- It is maintained at only one level that is page-level. Changes made on a single page is not visible on other pages.
- Information that is gathered in view state is stored for the clients only and cannot be transferred to any other place.
- View state is synonymous with serializable data only.
- ViewState has a tendency for the persistence of page-instance-specific data.
- view state is used, the values posted of a particular page persist in the browse area that the client is using and post back only when the entire operation is done. The data of the previous page is no longer available when another page is loaded. Also, Data is not secure in this case because it is exposed to clients. Encryption can be used for data security.
- It can be used to store information that you wish to access from same web page.

SessionState:

- It is maintained at session-level and data can be accessed across all pages in the web application.
- The information is stored within the server and can be accessed by any person that has access to the server where the information is stored.
- It can be used to store information that you wish to access on different web pages.

8).What is === operator?

- The strict equality (===) operator checks whether its two operands are equal, returning a Boolean result. Unlike the equality operator, the strict equality operator always considers operands of different types to be different.

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

- You can use the below-mentioned methods to add classes, remove classes, and toggle between different classes respectively.

1. The add() method: It adds one or more classes.
2. The remove() method: It removes one or more classes.
3. The toggle() method: If the class does not exist it adds it and returns true

- We can change, add or remove any CSS property from an HTML element on the occurrence of any event with the help of JavaScript. There are two common approaches that allow us to achieve this task.
- style.property
- Changing the class itself

10).How to read and write a file using javascript?

- 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.

JAVASCRIPT ASSIGNMENT

- **Syntax:** fs.readFile(file_name, encoding, callback_function)

- **Parameters:**
  - ✓ **filename:** It contains the filename to be read, or the whole path if the file is saved elsewhere.
  - ✓ **encoding:** It stores the file's encoding. 'utf8' is the default setting.
  - ✓ **callback function:** This is a function that is invoked after the file has been read. It requires two inputs:
  - ✓ **err:** If there was an error.
  - ✓ **data:** The file's content.
  - ✓ **Return Value:** It returns the contents contained in the file, as well as any errors that may have occurred.

- The fs.writeFile() function is used to write data to a file in an asynchronous manner. If the file already exists, it will be replaced.

- **Syntax:** fs.writeFile(file_name, data, options, callback)

- **Parameters:**
  - ✓ **file_name**: It's a string, a buffer, a URL, or a file description integer that specifies the location of the file to be written. When you use a file descriptor, it will function similarly to the fs. write() method.
  - ✓ **data**: The data that will be sent to the file is a string, Buffer, TypedArray, or DataView.
  - ✓ **options:** It's a string or object that may be used to indicate optional output options. It includes three more parameters that may be selected.
  - ✓ **encoding**: It's a string value that indicates the file's encoding. 'utf8' is the default setting.
  - ✓ **Mode:** The file mode is specified by an integer number called mode. 0o666 is the default value.
  - ✓ **Flag:** This is a string that indicates the file-writing flag. 'w' is the default value.
  - ✓ **callback**: This function gets invoked when the method is run.
  - ✓ **err**: If the process fails, this is the error that will be thrown.

1.file=fopen(getScriptPath(),0); The function fread() is used for reading the file content.

2.str = fread(file,flength(file) ; The function fwrite() is used to write the contents to the file.

file = fopen("c:\MyFile.txt", 3);

11).What are all the looping structures in JavaScript?

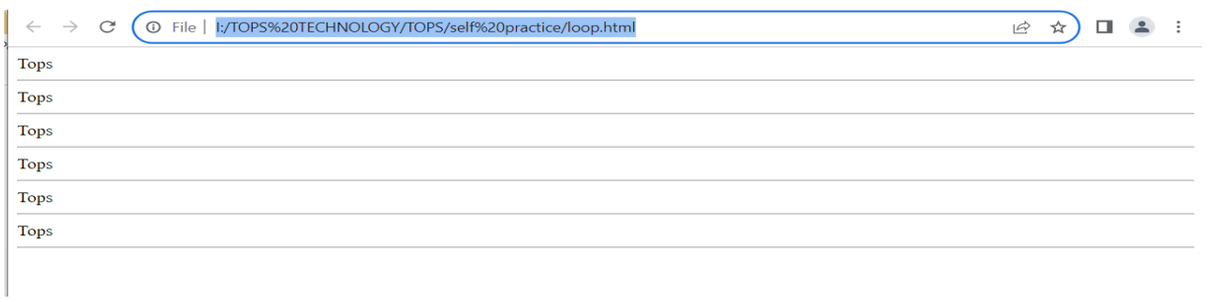
- JavaScript supports different kinds of loops:
  1.for - loops through a block of code a number of times
  2.while - loops through a block of code while a specified condition is true
  3.do/while - also loops through a block of code while a specified condition is true

JAVASCRIPT ASSIGNMENT

1). For loop Example :

```
<!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>
        for (let i = 0; i <= 5; i++) {
            document.write("Tops"+"<hr>")
        }
    </script>
</body>
</html>
```

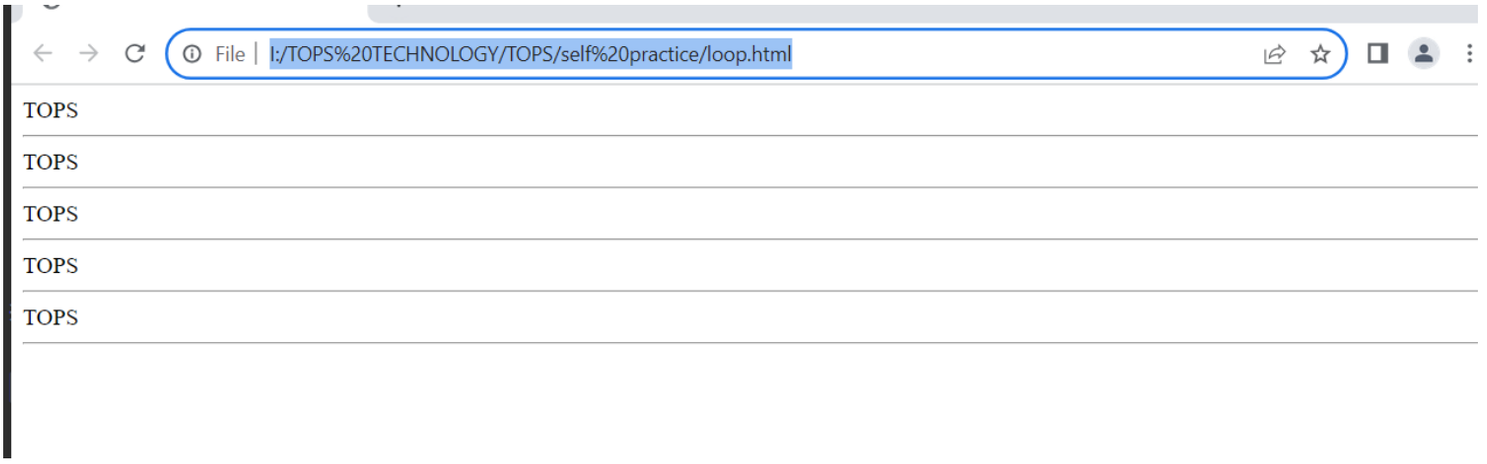
Output :



2). While loop example:

```
<!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 i=0
        while (i<5) {
            document.write("TOPS"+"<hr>")
            i++
        }

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

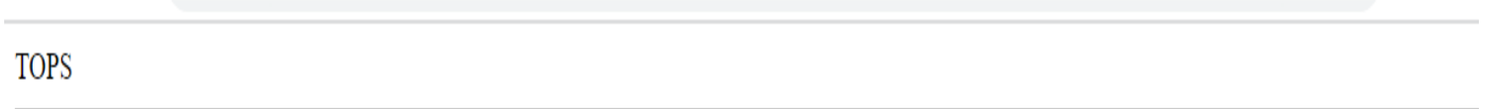
JAVASCRIPT ASSIGNMENT



3). Do while loop example :

```
<!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 i=6
            do {
                document.write("TOPS"+"<hr>")
                i++
            } while (i<5);
    </script>
</body>
</html>
```

Output :



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

- In JavaScript parseInt() function (or a method) is used to convert the passed-in string parameter or value to an integer value itself. This function returns an integer of the base which is specified in the second argument of the parseInt() function.
- To convert a string to an integer parseInt(), Number(), and Unary operator (+) function is used in JavaScript. The parseInt() function returns NaN (not a number) when the string doesn't contain number. If a string with a number is sent, then only that number will be returned as the output. This function won't accept spaces. If any particular number with spaces is sent, then the part of the number that presents before space will be returned as the output.

JAVASCRIPT ASSIGNMENT

Example

```
<!DOCTYPE html>
<html>
<body>
<p id="demo"></p>
<script>
document.getElementById("demo").innerHTML =
parseInt("10", 10)+ "<br>" +
parseInt("010")+ "<br>" +
parseInt("10", 8)+ "<br>" +
parseInt("0x10")+ "<br>" +
parseInt("10", 16);
</script>

</body>
</html>
```

Output :

10
10
8
16
16

13). What is the function of the delete operator?

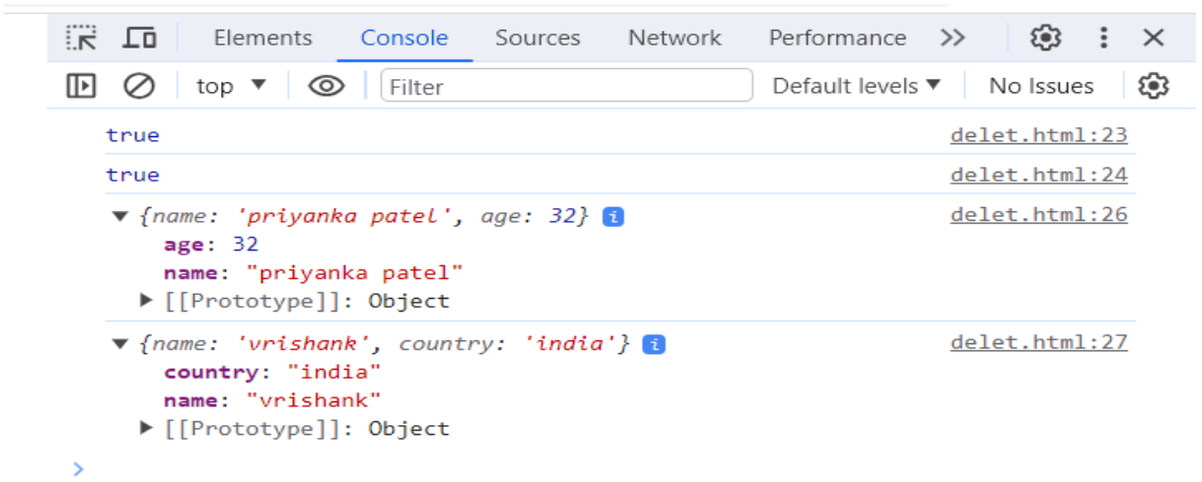
- The delete operator in JavaScript is used to delete an object's property.
- If it is used to delete an object property that already exists, it returns true and removes the property from the object. However, deleting an object property that doesn't exist will not affect the object, but will still return true.
- The only time false will be returned is when the delete operator is used to delete a variable or a function.
- **Syntax** : delete object.property ; or delete object["property"]
- **parameters : object :** this is the object whose property we want to delete.
  Property : this is the property to be deleted.
- **Return Value :** the delete operetors returns true if the specified property is deleted, or false if the property is not deleted.

JAVASCRIPT ASSIGNMENT

Example :

```
<!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>
    <div></div>
    <script>
        let human = {
    name: "priyanka patel",
    age: 32,
    country: "india"
}

let person = {
    name: "vrishank",
    age: 1,
    country : "india"
}
// log retured values after delete
console.log(delete human["country"])
console.log(delete person.age)
// log affected objects
console.log(human)
console.log(person)
    </script>
</body>
</html>
```

Output :



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

There are three types of pop-up boxes in JavaScript

1.Alert Box :

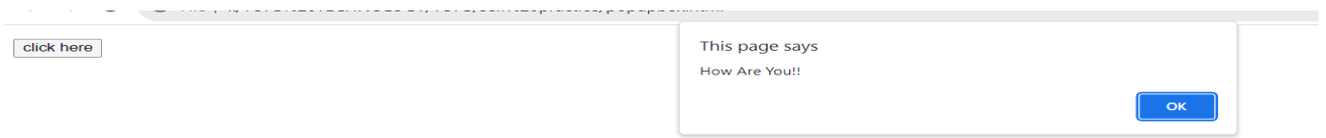
- An alert box is often used if you want to make sure information comes through to the user.
- When an alert box pops up, the user will have to click "OK" to proceed.
- **Syntax :** alert("I am an alert box!");

JAVASCRIPT ASSIGNMENT

Example :

```
<!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>
    <button onclick="a()">click here</button>
    <script>

        function a() {
            alert("How Are You!!")
        }

            document.getElementById("one").innerHTML=txt

    </script>
</body>

</html>
```

Output :



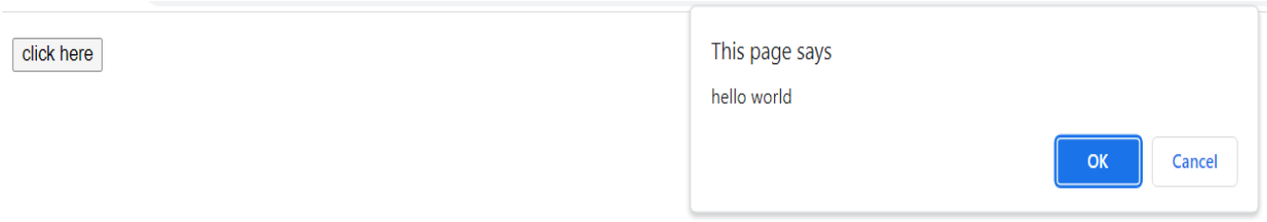
2.Confirm Box :

- A confirm box is often used if you want the user to verify or accept something.
- When a confirm box pops up, the user will have to click either "OK" or "Cancel" to proceed.
- If the user clicks "OK", the box returns **true**. If the user clicks "Cancel", the box returns **false**.
- **Syntax :** window.confirm("sometext");

Example :

```
<!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>
    <button onclick="a()">click here</button>
    <script>

        function a() {
            var txt
            if (confirm("hello world")) {
                txt = "OK"
            } else {
                txt = "NO"
            }
        }

            document.getElementById("one").innerHTML=txt

    </script>
</body>

</html>
```

JAVASCRIPT ASSIGNMENT



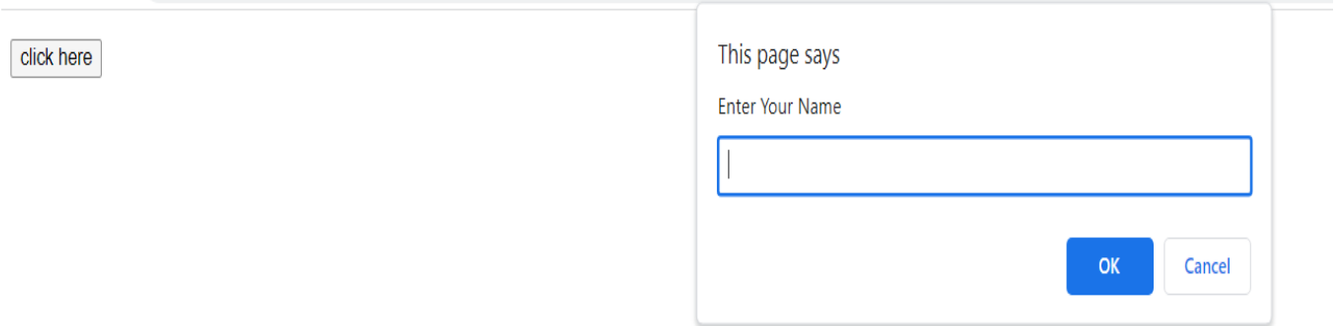
3.Prompt Box :

- A prompt box is often used if you want the user to input a value before entering a page.
- When a prompt box pops up, the user will have to click either "OK" or "Cancel" to proceed after entering an input value.
- If the user clicks "OK" the box returns the input value. If the user clicks "Cancel" the box returns null.
- **Syntax :** window.prompt("sometext","defaultText");

Example :

```
<!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>
    <button onclick="a()">click here</button>
    <script>

        function a() {
            var txt
            var per=prompt("Enter Your Name")
            if (per =="" || per == null) {
                txt="plz enter your name"
            } else {
                txt="HII "+per+" How Are You"
            }
        }
            document.getElementById("one").innerHTML=txt

    </script>
</body>
```

Output :



JAVASCRIPT ASSIGNMENT

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

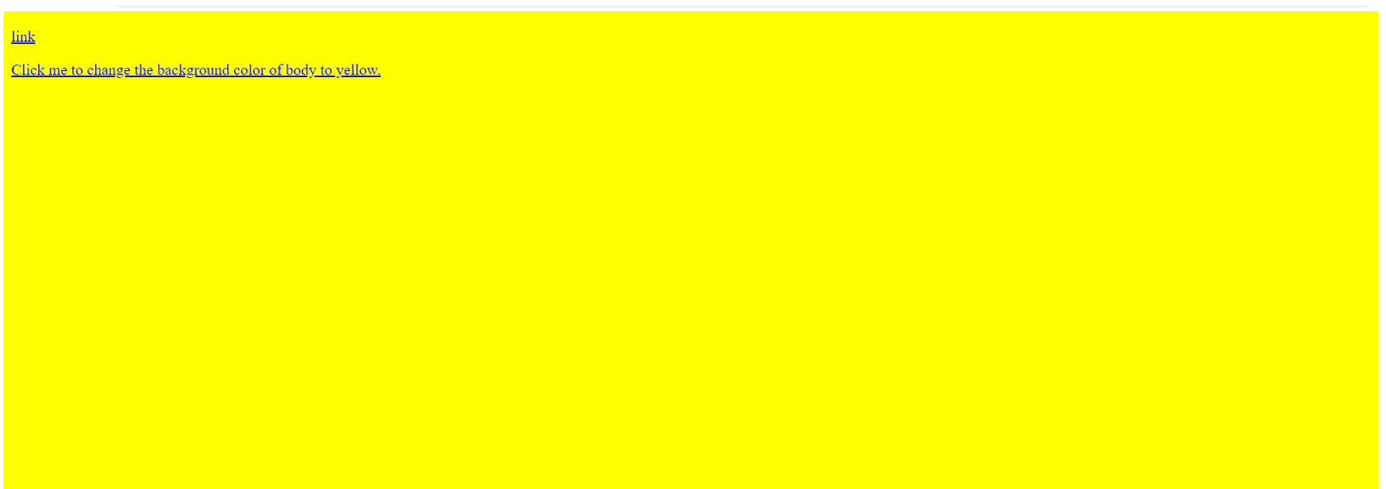
- javascript: void(0) means return undefined as a primitive value. We use this to prevent any negative effects on a webpage when we insert some expression. For example, in the case of URL hyperlinks.

Example :

```
<!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>
<a href="javascript:void(0);"> link</a>
</p>
<p>
<a href="javascript:void(document.body.style.backgroundColor='yellow');">
Click me to change the background color of body to yellow.</a>
</p>

</body>
</html>
```

\

Output :



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

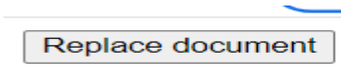
- In JavaScript, we can use window. location object to force a page to load another page. We can use the location object to set the URL of a new page.
- We can use **window.location** property inside the script tag to forcefully load another page in Javascript. It is a reference to a Location object that is it represents the current location of the document. We can change the URL of a window by accessing it.

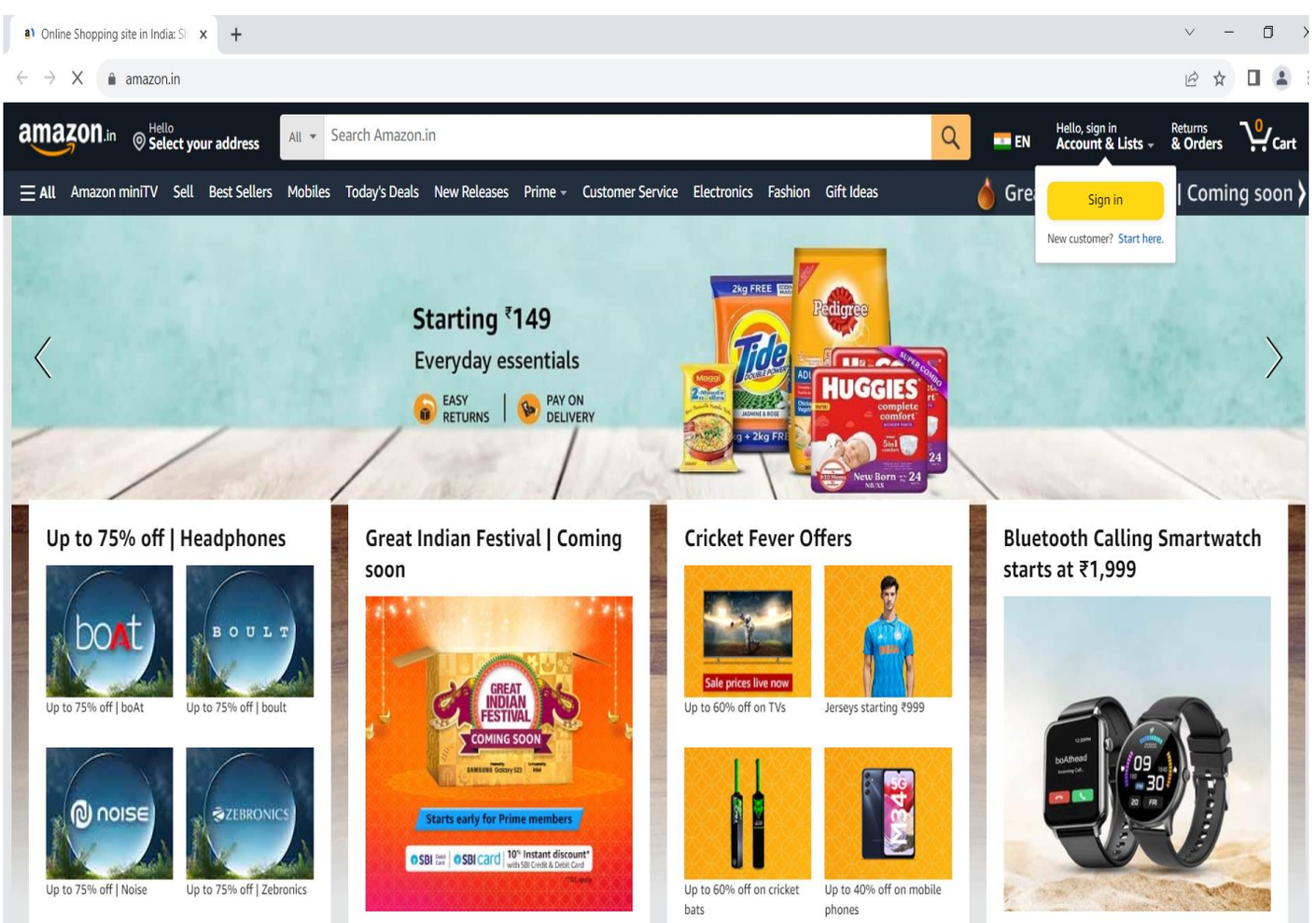
JAVASCRIPT ASSIGNMENT

Example :

```
<!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="myFunction()">Replace document</button>

    <script>
    function myFunction() {
      location.replace("https://www.amazon.in")
    }
    </script>

    </body>
</html>
```

Output :





JAVASCRIPT ASSIGNMENT

17). What are the disadvantages of using innerHTML in JavaScript?

- **The use of innerHTML very slow:** The process of using innerHTML is much slower as its contents as slowly built, also already parsed contents and elements are also re-parsed which takes time.
- **Preserves event handlers attached to any DOM elements:** The event handlers do not get attached to the new elements created by setting innerHTML automatically. To do so one has to keep track of the event handlers and attach it to new elements manually. This may cause a memory leak on some browsers.
- **Content is replaced everywhere:** Either you add, append, delete or modify contents on a webpage using innerHTML, all contents is replaced, also all the DOM nodes inside that element are reparsed and recreated.
- **Appending to innerHTML is not supported:** Usually, += is used for appending in JavaScript. But on appending to an Html tag using innerHTML, the whole tag is re-parsed.
- **Old content replaced issue:** The old content is replaced even if object.innerHTML = object.innerHTML + 'html' is used instead of object.innerHTML += 'html'. There is no way of appending without reparsing the whole innerHTML. Therefore, working with innerHTML becomes very slow. String concatenation just does not scale when dynamic DOM elements need to be created as the plus' and quote openings and closings becomes difficult to track.
- **Can break the document:** There is no proper validation provided by innerHTML, so any valid HTML code can be used. This may break the document of JavaScript. Even broken HTML can be used, which may lead to unexpected problems.
- **Can also be used for Cross-site Scripting(XSS):** The fact that innerHTML can add text and elements to the webpage, can easily be used by malicious users to manipulate and display undesirable or harmful elements within other HTML element tags. Cross-site Scripting may also lead to loss, leak and change of sensitive information.