

# \*BASIC JAVASCRIPT & DOM\*

## 1.What is JavaScript?

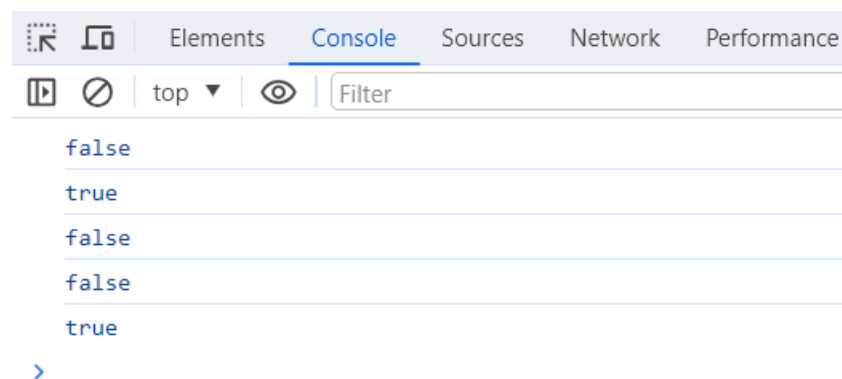
**ANS:** JavaScript is a programming language that allows we to add interactivity and dynamic features to websites. It's used for things like form validation, creating interactive elements, and manipulating web page content. It's a powerful tool for web development.

## 2.What is the use of isNaN function?

**ANS:**The `isNaN()` function in JavaScript is used to check if a value is "Not a Number" (NaN). It returns `true` if the value is NaN, and `false` if it is a valid number or can be converted into one.

Here's an example:

```
</head>
<body>
  <script>
    console.log(isNaN(123)); // Output: false
    console.log(isNaN('Hello')); // Output: true
    console.log(isNaN('123')); // Output: false
    console.log(isNaN(true)); // Output: false
    console.log(isNaN(undefined)); // Output: true
  </script>
</body>
</html>
```



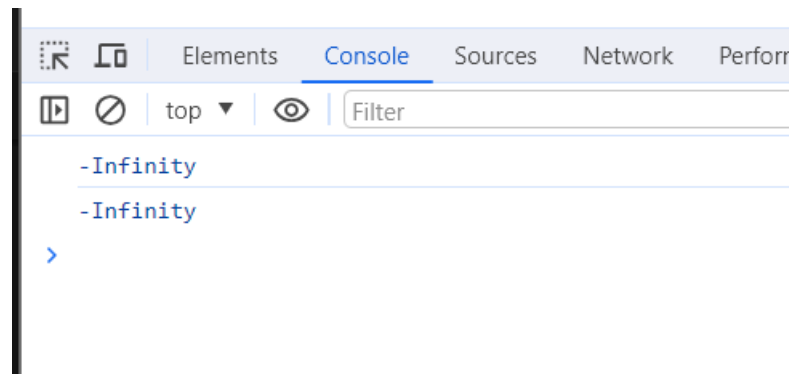
In the example above, the `isNaN()` function is used to check if different values are NaN or not. It's handy when we need to validate user input or perform conditional checks in your JavaScript code.

### 3.What is negative Infinity?

**ans:** Negative Infinity is a special value in JavaScript that represents a number that is infinitely small, or in other words, a value that is less than any other number. It is denoted by the keyword `Number.NEGATIVE\_INFINITY`. For example, if we divide a negative number by zero, the result will be negative infinity.

Here's an example:

```
</head>
<body>
  <script>
    console.log(-10 / 0); // Output: -Infinity
    console.log(Number.NEGATIVE_INFINITY); // Output: -Infinity
  </script>
</body>
</html>
```



Negative Infinity is often used in mathematical calculations or to represent an underflow condition.

### 4.Which company developed JavaScript?

**ans:**JavaScript was initially developed by Netscape Communications Corporation,

which is a software company. It was created by Brendan Eich in 1995. JavaScript has come a long way since then and is now supported by multiple companies and organizations. It's widely used for web development.

## 5.What are undeclared and undefined variables?

**ans:** Undeclared variables refer to variables that have not been declared or defined in the code. when we try to use an undeclared variable, it will result in an error.

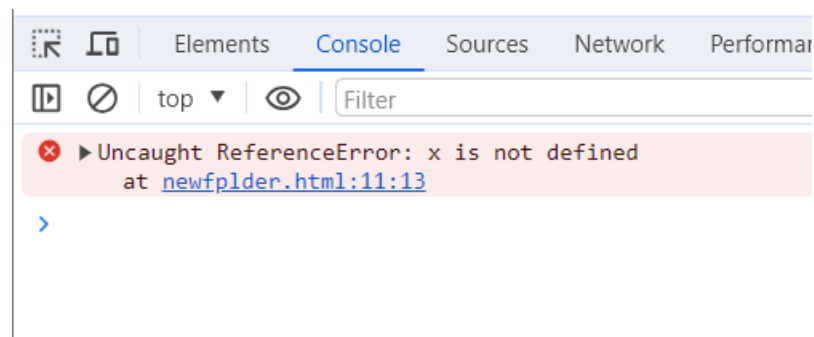
Undefined variables, on the other hand, are variables that have been declared but have not been assigned a value. They exist in the code, but their value is undefined until a value is assigned to them.

For example:

```
<script>
  // Undeclared variable
console.log(x); // Output: ReferenceError: x is not defined

// Undefined variable

let y;
console.log(y); // Output: undefined
</script>
</body>
</html>
```



It's important to declare variables before using them and assign values to them to avoid errors and ensure proper functionality in our code.

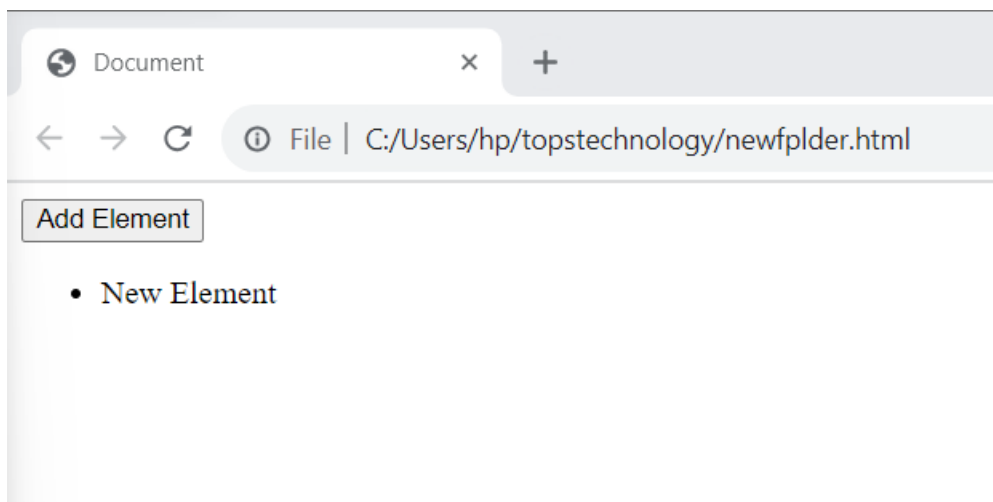
## 6.Write the code for adding new elements dynamically?

**ans:**here's an example of how we can dynamically add new elements to a webpage using JavaScript:

```
<body>
  <div id="container">
    <button id="addButton">Add Element</button>
    <ul id="list"></ul>
  </div>
  <script>
    // Get the necessary elements
    const addButton = document.getElementById("addButton");
    const list = document.getElementById("list");

    // Add event listener to the button
    addButton.addEventListener("click", function() {
      // Create a new list item
      const newItem = document.createElement("li");
      newItem.textContent = "New Element";

      // Append the new item to the list
      list.appendChild(newItem);
    });
  </script>
</body>
</html>
```



In this example,

we have an HTML container with a button and an empty list. When the button is clicked, a new list item is created and added to the list dynamically. You can

modify the code to add different types of elements or customize the content as per your needs. Have fun experimenting with dynamic element creation.

## **7.What is the difference between ViewState and SessionState?**

**ans:**In web development, ViewState and SessionState are both concepts related to maintaining state or preserving data in a web application. However, they are used in different contexts and have distinct purposes:

1. ViewState: ViewState is a feature in ASP.NET web forms that allows the server to track the state of controls on a web page. It is used to maintain the state of controls between postbacks (when a form is submitted to the server and the page reloads). ViewState stores the values of controls and their properties, ensuring that they retain their state across postbacks. It is stored as a hidden field in the HTML markup.

2. SessionState: SessionState is a mechanism for storing user-specific data on the server. It allows you to store and retrieve data throughout a user's session on a website. SessionState is used to maintain state across multiple requests from the same user. It can store any type of data and is accessible to all pages within the same user session. SessionState is typically stored in memory on the server or in an external storage mechanism.

To summarize:

- ViewState is used to maintain the state of controls on a single web page during postbacks.
- SessionState is used to store user-specific data across multiple requests during a user's session.

Both ViewState and SessionState are useful for preserving data and maintaining state in web applications, but they serve different purposes and are used in different scenarios.

## 8.What is === operator?

**ans:**The `===` operator is a comparison operator in JavaScript. It is used to compare two values for equality, including their types.

Unlike the `==` operator, which performs type coercion (converting the operands to a common type before comparison), the `===` operator strictly checks for equality without any type conversion.

```
8 <body>
9   <script>
10    const num1 = 5;
11    const num2 = "5";
12
13    console.log(num1 === num2); // Output: false (different types)
14    console.log(num1 == num2); // Output: true (type coercion)
15  </script>
16 </script>
17 </body>
18 </html>
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Filter (e.g. text, !exclude)

false  
true

In the example above, `num1 === num2` returns `false` because `num1` is a number and `num2` is a string, even though their values are the same. On the other hand, `num1 == num2` returns `true` because the `==` operator performs type coercion and considers them equal.

The `===` operator is often preferred for strict equality checks because it ensures

that both the values and types are the same.

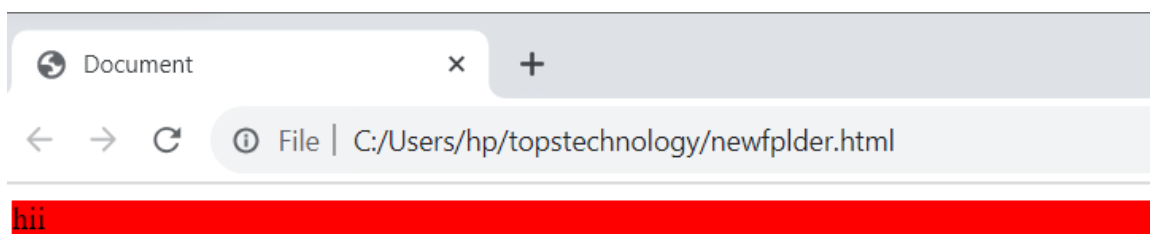
## 9.How can the style/class of an element be changed?

**ans:** To change the style or class of an element in HTML, you can use JavaScript or CSS. Here's how you can do it using JavaScript:

1. Select the element you want to modify. You can use various methods to select an element, such as ``getElementById``, ``getElementsByClassName``, or ``querySelector``.

2. Use the ``style`` property to modify the inline styles of the element. For example, to change the background color of an element with the ID "myElement" to red, we can do:

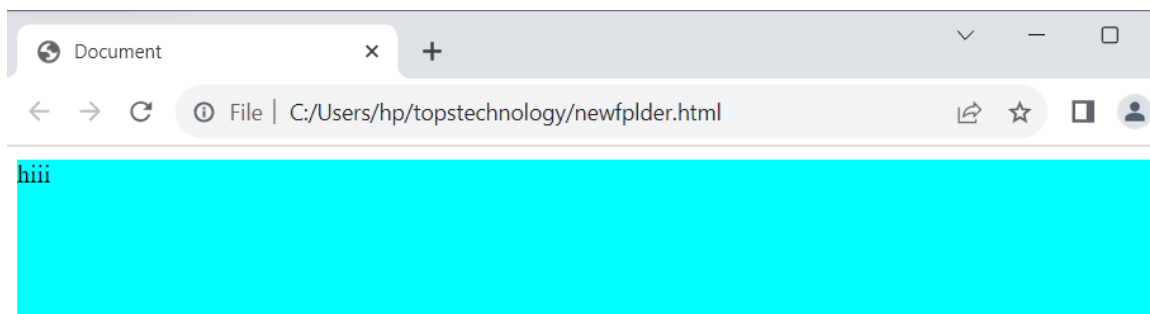
```
</head>
<body>
  <div id="myElement">hii</div>
  <script>
    const element = document.getElementById("myElement");
    element.style.backgroundColor = "red";
  </script>
</body>
</html>
```



3. Alternatively, we can add or remove classes to change the styling. First, define the CSS classes in your CSS file or `<style>` tag. Then, use the ``classList`` property to add or remove classes from the element. For example, to add a class named

"highlight" to an element with the ID "myElement":

```
<title>Document</title>
<style>
  .highlight{
    height: 100px;
    background-color: aqua;
  }
</style>
</head>
<body>
  <div id="myElement">hiii</div>
  <script>
    const element = document.getElementById("myElement");
    element.classList.add("highlight");
  </script>
</body>
</html>
```



To remove a class, we can use `classList.remove("className")`. if we want to toggle a class (add if it's not present, remove if it's already present), we can use `classList.toggle("className")`.

Remember to replace "myElement" with the ID or selector of the actual element we want to modify, and "highlight" with the name of the CSS class we want to add or remove.



Using these JavaScript techniques, we can dynamically change the style or class of an element based on user interactions or other events.

## 10.How to read and write a file using JavaScript?

**ans:**Here's an example of how we can read and write a file using JavaScript with the File System API:

To read a file:

```
```\javascript
function readFile(file) {
    const reader = new FileReader();
    reader.onload = function(event) {
        const contents = event.target.result;
        console.log(contents);
    };
    reader.onerror = function(event) {
        console.error("Error reading file:", event.target.error);
    };
    reader.readAsText(file);
}
```

// Usage example

```
const fileInput = document.getElementById('fileInput');
fileInput.addEventListener('change', function(event) {
    const file = event.target.files[0];
    readFile(file);
});
```

...

To write to a file:

```
````javascript
function writeFile(filename, content) {
    const element = document.createElement('a');
    element.setAttribute('href', 'data:text/plain;charset=utf-8,' +
        encodeURIComponent(content));
    element.setAttribute('download', filename);
    element.style.display = 'none';
    document.body.appendChild(element);
    element.click();
    document.body.removeChild(element);
}
```

// Usage example

```
const filename = 'file.txt';
const content = 'Hello, world!';
writeFile(filename, content);
...
```

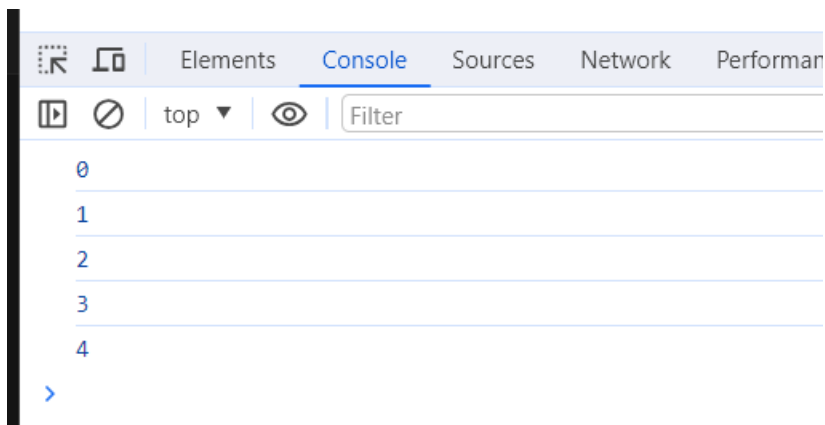
In these examples, we're using the FileReader API to read the contents of a file and the data URL scheme to create a downloadable file for writing. Remember to handle any errors that may occur during the process.

## **11.What are all the looping structures in JavaScript?**

**ans:** In JavaScript, there are several looping structures you can use to execute code repeatedly. The most common ones are:

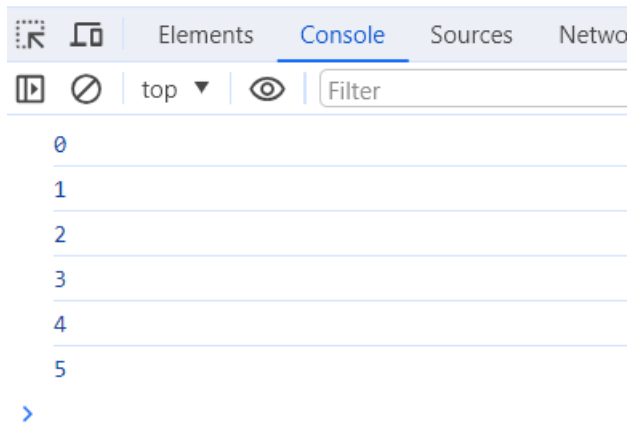
### 1. `for` loop:

```
<script>
for (let i = 0; i < 5; i++) {
  console.log(i);
}
</script>
</body>
</html>
```



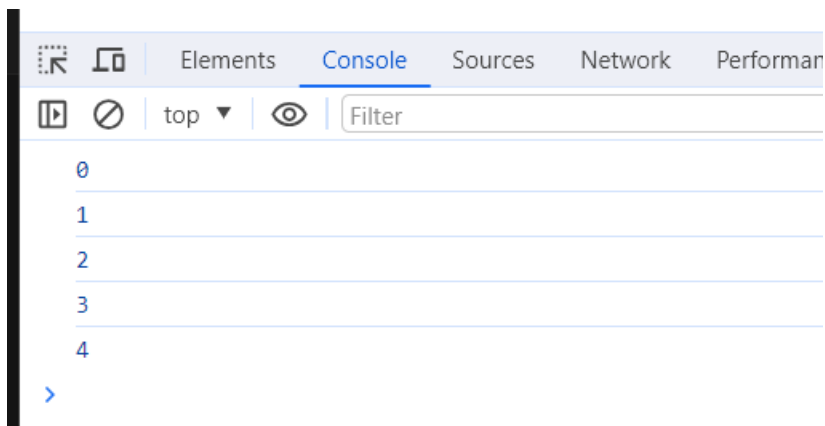
### 2. `while` loop:

```
<script>
let i = 0;
while (i < 6) {
  console.log(i);
  i++;
}
</script>
</body>
```



### 3. `do...while` loop:

```
<script>
let i = 0;
do {
  console.log(i);
  i++;
} while (i < 5);
</script>
```



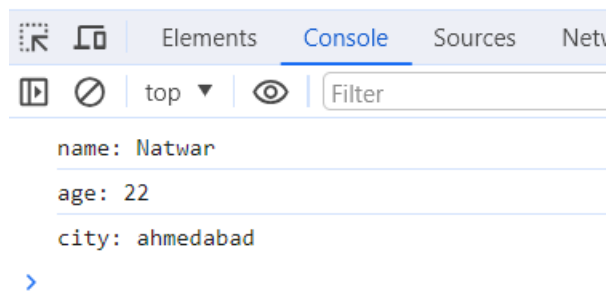
### 4. `for...in` loop (for iterating over object properties):

```

<script>
const person = {
  name: 'Natwar',
  age: 22,
  city: 'ahmedabad'
};

for (let prop in person) {
  console.log(prop + ': ' + person[prop]);
}
</script>
</body>
</html>

```



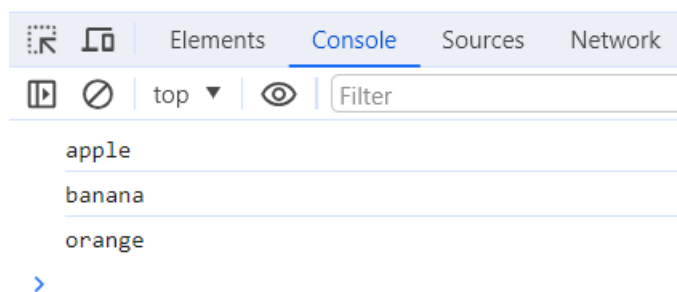
## 5. `for...of` loop (for iterating over iterable objects like arrays or strings):

```

<script>
const fruits = ['apple', 'banana', 'orange'];

for (let fruit of fruits) {
  console.log(fruit);
}
</script>

```



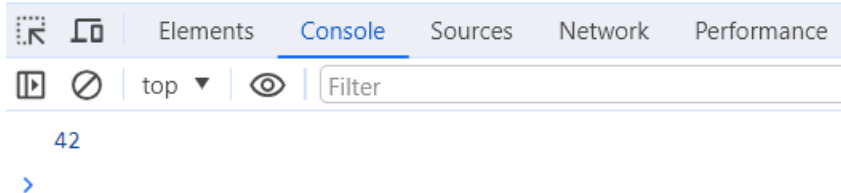
These are some of the looping structures in JavaScript. Each has its own use case depending on the situation.

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

**ans:** To convert a string of any base to an integer in JavaScript, we can use the `parseInt()` function. The `parseInt()` function takes two arguments: the string to be converted and the base of the number system.

Here's an example:

```
<script>
const binaryString = "101010";
const decimalNumber = parseInt(binaryString, 2);
console.log(decimalNumber); // Output: 42
</script>
```



In this example, we convert the binary string `"101010"` to its decimal equivalent by passing the string and the base `2` to `parseInt()`. The resulting decimal number is `42`.

we can replace `2` with the desired base (such as `8` for octal, `16` for hexadecimal, or any other valid base).

Remember to handle any potential errors that may occur during the conversion process.

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

**ans:** The `delete` operator in JavaScript is used to remove a property from an object or to delete an element from an array. It allows us to remove a specific item and modify the structure of an object or array.

Here's an example of using the `delete` operator:

```
<script>
const person = {
  name: "Natwar",
  age: 22,
  city: "AHMEDABAD"
};

console.log(person); // Output: { name: "Natwar", age: 30, city: "AHMEDABAD" }

delete person.age;

console.log(person); // Output: { name: "Natwar", city: "AHMEDABAD" }
</script>
```



In this example, we have an object called `person` with properties like `name`, `age`, and `city`. By using `delete person.age`, we remove the `age` property from the `person` object. After deleting, the `age` property is no longer present in the object.

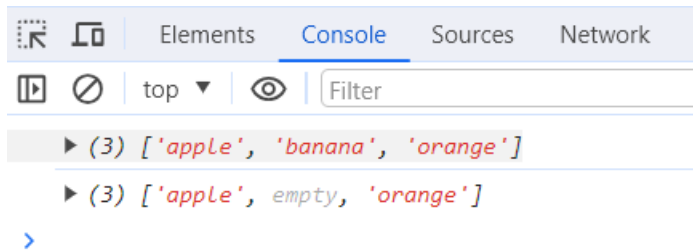
Similarly, We can use the `delete` operator to remove elements from an array:

```
<script>
const fruits = ["apple", "banana", "orange"];

console.log(fruits); // Output: ["apple", "banana", "orange"]

delete fruits[1];

console.log(fruits); // Output: ["apple", empty, "orange"]
</script>
```



In this case, we delete the element at index `1` from the `fruits` array using `delete fruits[1]`. As a result, the element at index `1` becomes `empty`.

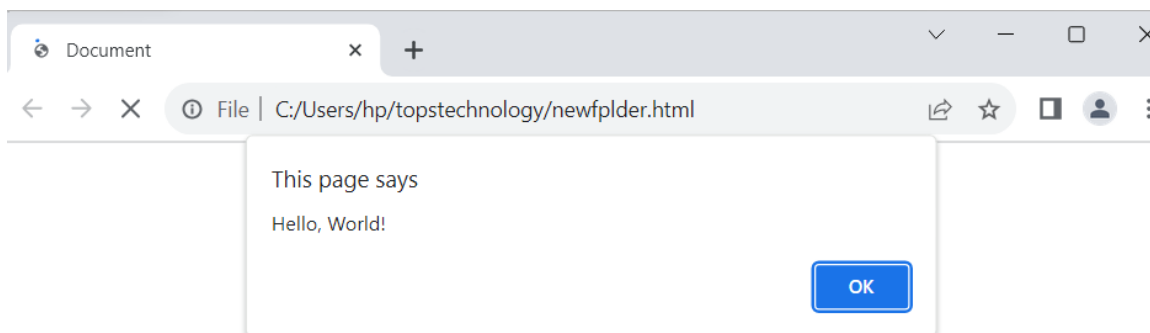
It's important to note that the `delete` operator does not affect variables or functions declared with the `var`, `let`, or `const` keywords. It is primarily used to delete properties from objects or elements from arrays.

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

**ans:** In JavaScript, there are three types of popup boxes that you can use to interact with the user: `alert`, `confirm`, and `prompt`.

1. The `alert` popup box is used to display a message to the user. It only has one button, typically labeled "OK". Here's an example:

```
<script>
alert("Hello, World!");
</script>
```

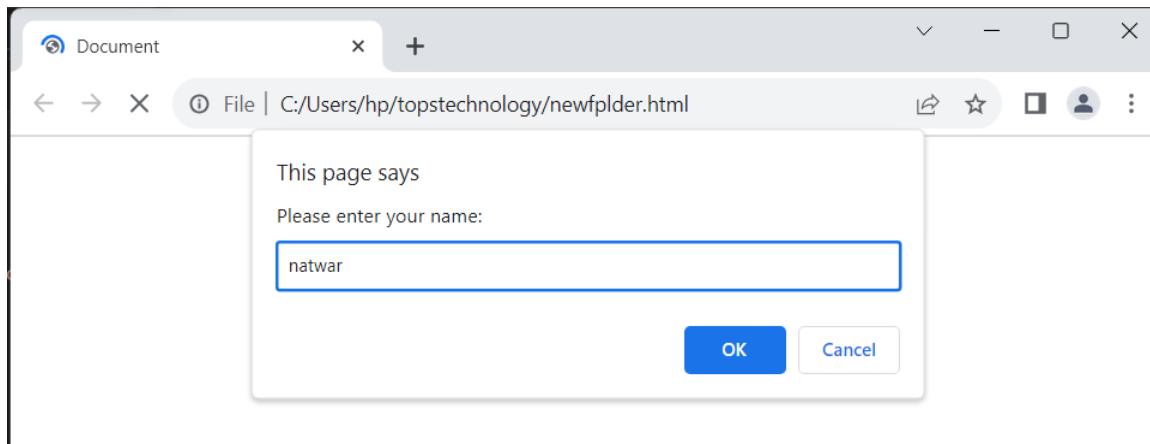




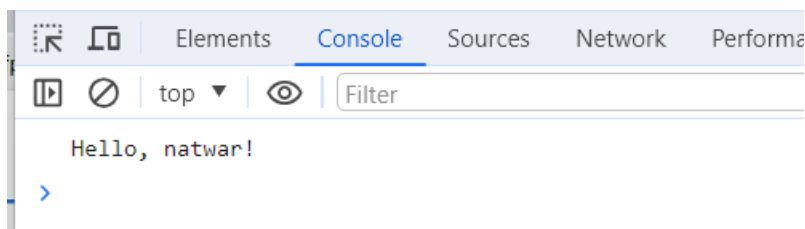
This will display a popup box with the message "Hello, World!" and an "OK" button.

2. The `prompt` popup box is used to get input from the user. It has an input field where the user can enter text, along with buttons for "OK" and "Cancel". Here's an example:

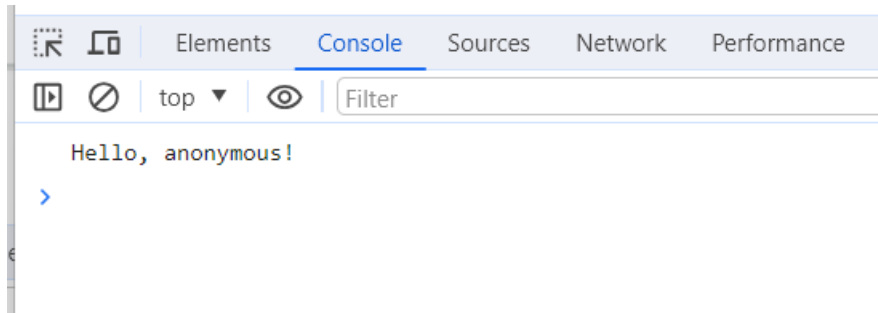
```
<script>
const name = prompt("Please enter your name:");
if (name) {
  // User entered a name
  console.log("Hello, " + name + "!");
} else {
  // User clicked "Cancel" or entered an empty string
  console.log("Hello, anonymous!");
}
</script>
```



**after ok:**



**after cancel:**



### 13.What is the use of Void (0)?

**ans:**The ``void(0)`` expression in JavaScript is often used in conjunction with the ``href`` attribute of an anchor tag (`<a>`). It is typically used to prevent the default behavior of the anchor tag, which is to navigate to a new page when clicked.

By setting the ``href`` attribute to ``javascript:void(0)``, clicking on the anchor tag will not cause any navigation. It essentially does nothing.

Here's an example:

```
```html
<a href="javascript:void(0)">Click me</a>
```
```

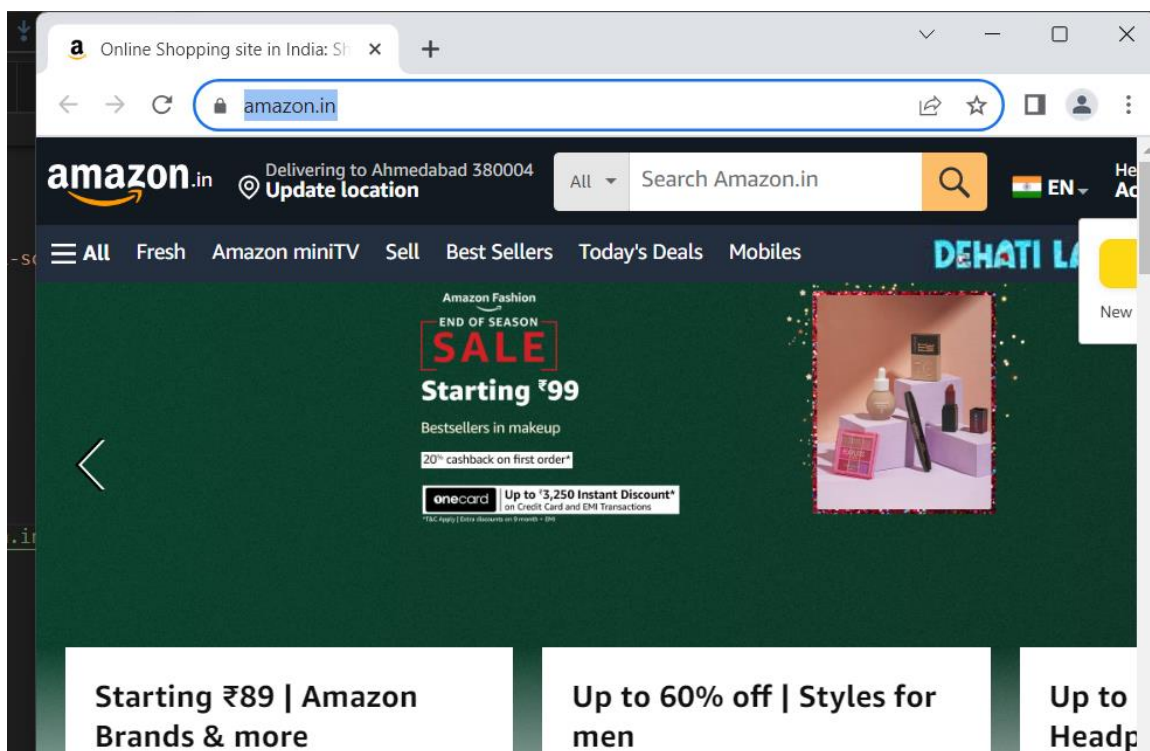
When the user clicks on the "Click me" link, nothing will happen. It's commonly used when you want to attach a click event handler to the anchor tag using JavaScript, but you don't want the page to navigate to a new URL.

However, it's worth noting that in modern web development, it's generally recommended to use event listeners and prevent default behavior using JavaScript instead of relying on ``void(0)``.

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

**ans:** If we want to force a page to load another page in JavaScript, we can use the `window.location` object to change the URL of the current page. Here's an example:

```
8 </head>
9 <body>
10 <a href="www.flipcart.in">flipcart</a>
11 <script>
12 window.location.href = "https://www.amazon.in";
13
14 // In this example, when this JavaScript code is executed,
15 // the current page will be redirected to "https://www.amazon.in".
16 </script>
17
18 </body>
19 </html>
```



It's important to note that this will immediately navigate the user to the new

page, so make sure to use it responsibly and provide a clear indication to the user that they will be redirected.

#### **14.What are the disadvantages of using innerHTML in JavaScript?**

**ans:** Using `innerHTML` in JavaScript does have some disadvantages to consider:

1. **\*\*Security risks\*\***: When using `innerHTML`, we need to be cautious about potential security vulnerabilities like cross-site scripting (XSS) attacks. If we insert untrusted or user-generated content directly into `innerHTML`, it can execute malicious scripts.
2. **\*\*Performance impact\*\***: Updating `innerHTML` can be less efficient compared to other DOM manipulation methods. When we modify `innerHTML`, the browser needs to re-parse and re-render the entire HTML structure of the affected element, which can be slower for large or complex content.
3. **\*\*Event listener loss\*\***: If we update an element's `innerHTML`, any event listeners attached to its child elements will be lost. we'll need to reattach the event listeners after each `innerHTML` update.

To illustrate, here's an example:

```
``javascript
// Example using innerHTML

const container = document.getElementById("container");
container.innerHTML = "<p>Hello, <strong>world!</strong></p>";

// Potential issues:
```

// - Security risks if untrusted content is inserted.

// - Performance impact due to re-parsing and re-rendering.

// - Event listeners attached to child elements are lost and need to be reattached.

...

To mitigate these disadvantages, we can explore alternative approaches like using DOM manipulation methods (`createElement`, `appendChild`, etc.) or utilizing modern frameworks like React or Vue.js that handle updates efficiently and provide security measures.

