(1) What is JavasCript?

Ans:

JavaScript is a programming language that developers use to make interactive webpages.

From refreshing social media feeds to displaying animations and interactive maps, JavaScript functions can improve a website's user experience.

As a client-side scripting language, it is one of the core technologies of the World Wide Web.

(2) What is the use of isNaN function?

Ans:

The isNaN function is a built-in JavaScript function that stands for "is Not a Number." It is used to determine whether a given value is not a valid number.

The primary purpose of the isNaN function is to check if a value is a numeric value or can be converted into a numeric value. It returns a Boolean value indicating whether the provided value is "not a number" (NaN) or not.

The syntax for using the isNaN function is as follows:

- isNaN(value)

(3) What is negative Infinity?

Ans:

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.

(4) Which company developed JavaScript?

Ans:

JavaScript was developed by Netscape Communications Corporation.

JavaScript was created at Netscape Communications by Brendan Eich in 1995.

(5) What are undeclared and undefined variables?

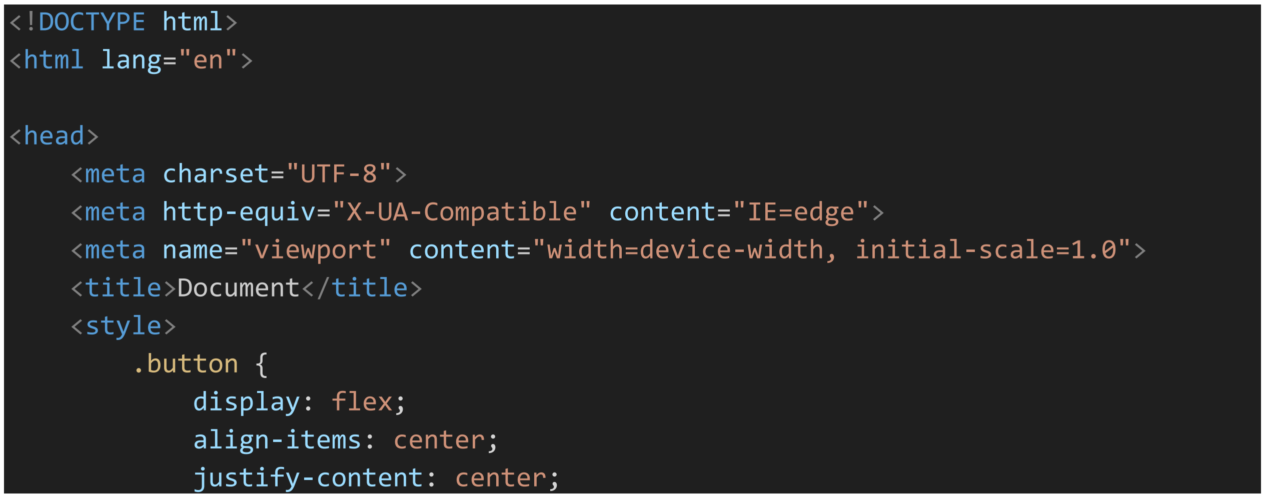
Ans:

Undeclared Variables: An undeclared variable is a variable that has not been explicitly defined or declared in the program before it is used. This typically occurs when a programmer tries to access or assign a value to a variable that has not been declared using the appropriate syntax or declaration statement.

Undefined Variables: An undefined variable is a variable that has been declared or referenced in the program, but it does not have a meaningful value assigned to it. In other words, the variable does not have a defined initial value, or it has been assigned a value that is indeterminate or uninitialized.

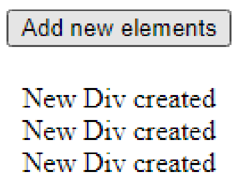
(6) Write the code for adding new elements dynamically?

Ans:



```
<!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">
    <title>Document</title>
    <style>
        .button {
            display: flex;
            align-items: center;
            justify-content: center;
        }

        .tasks {
            display: flex;
            justify-content: center;
            align-items: center;
            flex-direction: column;
            margin-top: 20px;
        }
    </style>
</head>

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

        // Getting the parent element in which
        // the new div will be created
        let task = document.getElementsByClassName("tasks");

        // Getting the addTask button element
        let addTask = document.getElementById("addTask");

        // Adding onclick event to the button
        addTask.addEventListener('click', function () {

            // Traversing through collection of HTML
            // elements (tasks here)
            for (let i = 0; i < task.length; i++) {

                let newDiv = document.createElement("div");

                newDiv.setAttribute("class", "list");

                // innerText used to write the text in newDiv
                newDiv.innerText = "New Div created";

                task[i].append(newDiv);
            }
        })
    </script>
</body>

</html>
```

Output:



```
New Div created
New Div created
New Div created
```

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

Ans:

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.

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.

(8) What is === operator?

Ans:

The === operator is a strict equality comparison operator used in various programming languages, including JavaScript. It is typically used to compare two values for equality while also considering their data types. In JavaScript, the === operator checks if the operands are equal in both value and type.

If the operands are of different types, return false.
If both operands are objects, return true only if they refer to the same object.

Ex:
```
    <script>
                var num = 42;
                var str = "42";
        console.log(num === 42);   // true
        console.log(num === "42"); // false
    </script>
```

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

Ans: 1:Changing CSS with the help of the style property:

Syntax: document.getElementById("id").style.property = new_style

Ex:
```
    <input type="text" id="pan" />
    <p></p>
    <button id="submit">Validate</button>

    <script>
        const btn = document.getElementById("submit");
        btn.addEventListener("click", function () {
            const pan = document.getElementById("pan").value;
            const para = document.querySelector("p");

            let regex = /([A-Z]){5}([0-9]){4}([A-Z]){1}$/;
            if (regex.test(pan.toUpperCase())) {
                para.innerHTML = "Hurrey It's correct";

                // Inline style
                para.style.color = "green";
            } else {
                para.innerHTML = "OOps It's wrong!";

                // Inline style
                para.style.color = "red";
            }
        });
    </script>
```

2. Changing the class itself – We can use two properties that can be used to manipulate the classes.

Syntax: document.getElementById("id").classList

Ex:

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

<head>
    <style>
        .hide {
            display: none;
        }

        .blueColor {
            color: blue;
        }
    </style>
</head>

<body>
    <h1 style="color: green;">
        GeeksforGeeks
    </h1>

    <h2>
        How can the style/class of
        an element be changed?
```

```
</h2>

<h3>Hide and Show the Para</h3>

<p>
        computer science portal
        for geeks. This platform has been designed
        for every geek wishing to expand their
        knowledge, share their knowledge, and is
        ready to grab their dream job. GFG have
        millions of articles,
</p>

<button id="hide">Hide</button>
<button id="show">Show</button>
<button id="color">Change Color</button>

<script>
        const btn_hide = document.getElementById("hide");
        const btn_show = document.getElementById("show");
        const btn_color = document.getElementById("color");

        const para = document.querySelector("p");
        btn_hide.addEventListener("click", function () {
                para.classList.add("hide");
        });

        btn_show.addEventListener("click", function () {
                para.classList.remove("hide");
        });
```

```
            btn_color.addEventListener("click", function () {
                para.classList.toggle("blueColor");
            });
    </script>
</body>

</html>
```

(10) How to read and write a file using JavaScript?

Ans:

- Path – The first parameter is the path of the test file from which the contents are to read. If the current location or directory is the same directory where the file which is to be opened and read is located then, only the file name has to be given.
- Format – The second parameter is the optional parameter which is the format of the text file. The format can be ASCII, utf-8 etc.
- CallBackFunc – The third parameter is the call back function which takes the error as the parameter and displays the fault is any raised due to the error.

```
• const fs = require('fs')
• fs.readFile('tp.txt', (err, inputD) => {
•    if (err) throw err;
•       console.log(inputD.toString());
• })
```

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

Ans:

1. <u>for loop</u>: The for loop is used to execute a block of code a specific number of times. It consists of three parts: initialization, condition, and iteration. The loop continues until the condition evaluates to false.

Here's the basic syntax:

```
for (initialization; condition; iteration) {

  // code to be executed

}
```

2. while loop: The while loop repeatedly executes a block of code as long as the specified condition is true. It only has a condition, and the loop continues until the condition becomes false.

Here's the basic syntax:

```
while (condition) {

  // code to be executed

}
```

3. do...while loop: The do...while loop is similar to the while loop, but the condition is checked after executing the block of code. This ensures that the code block is executed at least once, even if the condition is initially false.

Here's the basic syntax:

```
do {

  // code to be executed

} while (condition);
```

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

Ans: Given a string containing an integer value and along with that user passes a base value. We need to convert that string of any base value to an integer in JavaScript.

String   Integer

"1002"   1002

For performing the above-illustrated task, we would be using a method (or a function) provided by JavaScript called as **parseInt().**
This is a special method, provided by JavaScript, that takes an integer value (of any base which is either specified or not) and further converts the string into an integer value.
Syntax:

- Following is the syntax that a user may use to convert a string into an integer value (of any base)-

  parseInt(string_value, base)

- Alternatively, if we don't want to specify the base value and just want to convert our string value into an integer value itself, then we may use the following syntax also-

  parseInt(string_value)

Default value returned by base or radix of parseInt() method is **10.** In other words, if we don't specify any base or radix value then it by default converts the string value to an integer value by taking into regard the base or radix value as 10.
Let us visualize all of the above-illustrated facts with some of the following examples-
Example: In this example, we would be passing the string value in a method (which is explicitly declared for ease purpose) and further that string value is passed inside the parseInt() method which then further converts that string value in the corresponding integer value.

- JavaScript

```
<script>
  let stringConversion = (string_value) => {
    console.log("Initial Type: " + typeof string_value);
    let integer_value = parseInt(string_value);
    console.log("Final Type: " + typeof integer_value);
    console.log(integer_value);
  };
```

```
    stringConversion("512000");
    stringConversion("126410");
    stringConversion("0x8975");
</script>
```

Output:

```
Initial Type: string

Final Type: number

512000

Initial Type: string

Final Type: number

126410

Initial Type: string

Final Type: number

35189
```

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

Ans:

In JavaScript, the delete operator is used to remove a property from an object or delete an element from an array. Its main function is to delete a specific property or element and update the object or array accordingly. Here's how the delete operator is used:

For Object:

```
<script>
    // For Object
    const obj = { Cool: 'Hey', with: 'yes' };
    delete obj.Cool;
```

```
        console.log(obj); // Output: { with: 'yes' }
    </script>
```

For Array:

```
<script>
        // For array
        const arr = [1, 2, 3, 4, 5];
        delete arr[2];
        console.log(arr); // Output: [1, 2, undefined, 4, 5]
</script>
```

Note that when you delete an element from an array using the delete operator, the array length remains the same, and the deleted element is replaced with undefined. The array indexes are not reorganized or updated.

It's important to mention that the delete operator only works for deleting properties from objects and elements from arrays. It cannot be used to delete variables or function declarations.

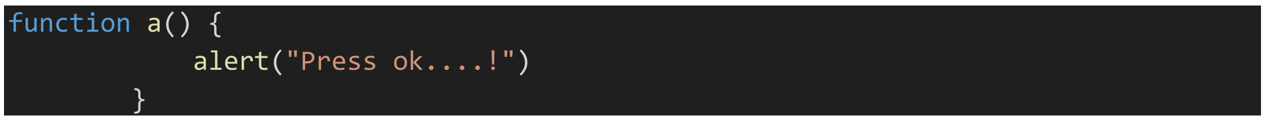
(14) What are all the types of Pop up boxes available in JavaScript?

Ans:

JavaScript has three kind of popup boxes: Alert box, Confirm box, and Prompt box.

1. Alert box:

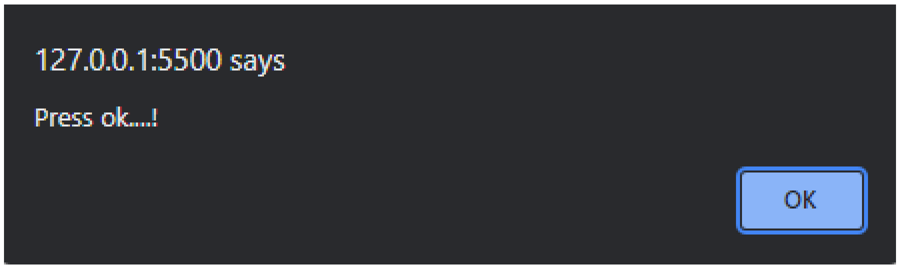
```
<button onclick="a()">Click here!</button>
<script>
```



```
function a() {
            alert("Press ok....!")
        }
```

```
</script>
```

Output:

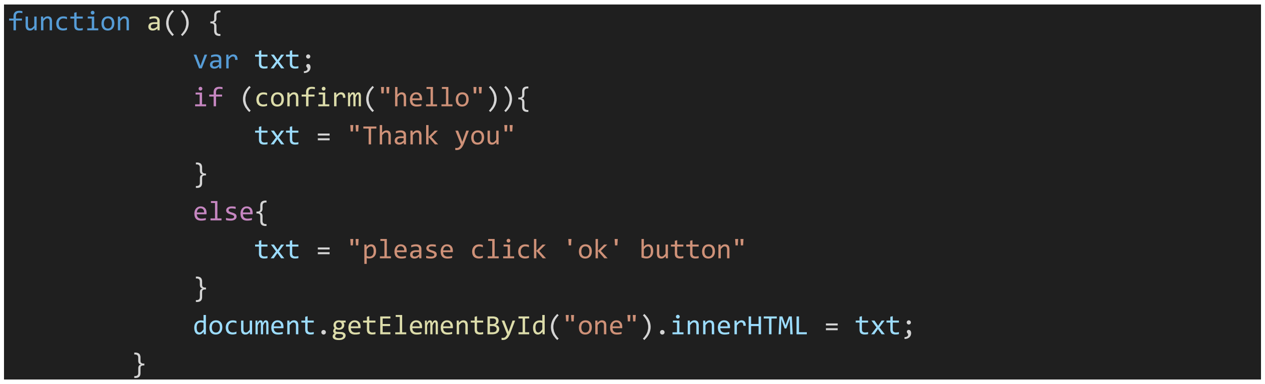


2. Confirm box:

<button onclick="a()">Click here!</button>

<p id="one"></p>

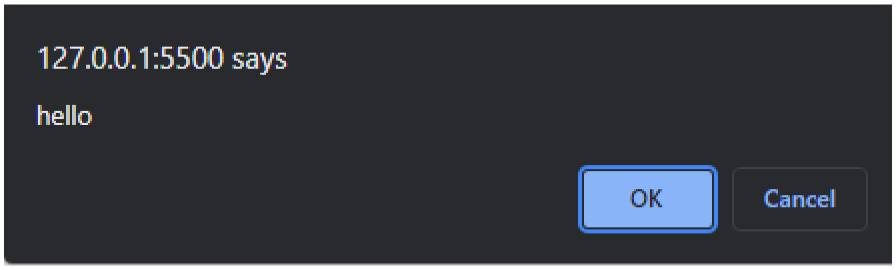
<script>



```
function a() {
            var txt;
            if (confirm("hello")){
                txt = "Thank you"
            }
            else{
                txt = "please click 'ok' button"
            }
            document.getElementById("one").innerHTML = txt;
        }
```

</script>

Output:



If you are press 'ok' button, you will get "Thank you" in the output (if condition execute).

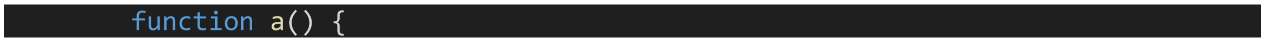
If you are press 'cancel' button, you will get "Please click ok button" in the output (else condition execute).

3. Prompt box:

<button onclick="a()">Click here</button>

<p id="one"></p>

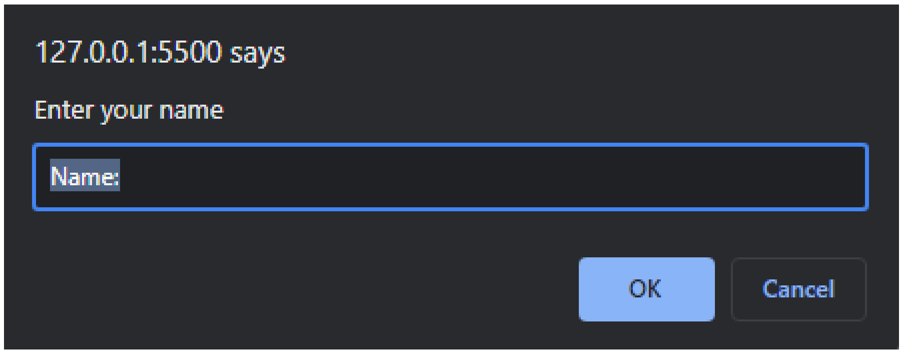
<script>



```
        function a() {
```

```
        var txt;
        var person = prompt("Enter your name","Name:")
        if (person == null || person == ""){
            txt = "please enter your name"
        }
        else{
            txt = "Hii " +person+ " How Are You"
        }
        document.getElementById("one").innerHTML =txt;
    }
```
</script>

Output:


127.0.0.1:5500 says

Enter your name

Name:

If you are without enter a name and press ok button you will get "please enter your name" in the output (if condition execute).

If you are enter a name and press ok button you will get "Hii+your name+ how are you" in the output (else condition execute).

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

Ans:

The void(0) expression in JavaScript is used to evaluate an expression and always return undefined. Its primary use is to create a hyperlink or button that performs an action without navigating to a new page when clicked.

Ex:

<a href="javascript:void(0);" ondblclick="apple()">Double click on me </a>

<p id="one"></p>

<script>

```
    function apple() {
        document.getElementById("one").innerHTML = "Welcome to my website";
    }
```
</script>

(16) How can a page be forced to load another page in JavaScript?
Ans:
In JavaScript, you can force a page to load another page by modifying the window.location object. There are a few ways to achieve this:
1. Using the window.location.href property:

```
    <script>
        window.location.href = "https://www.google.com/";
    </script>
```

This sets the href property of the window.location object to the URL you want to load. The browser will navigate to the specified URL, effectively loading a new page.
2. Using the window.location.replace() method:

```
    <script>
        window.location.replace("https://www.facebook.com");
    </script>
```

The replace() method of the window.location object replaces the current page in the browser's history with the new URL. This means that the user won't be able to navigate back to the current page using the browser's back button.
3. Using the window.location.assign() method:

```
    <script>
        window.location.assign("https://www.google.com");
    </script>
```

The assign() method of the window.location object loads the new URL, similar to setting window.location.href. It adds the new URL to the browser's history, so the user can navigate back to the current page using the browser's back button.

(17) What are the disadvantages of using innerHTML in JavaScript?
Ans:
Security risks: Using innerHTML to insert user-generated or untrusted content directly into your HTML can expose your application to security vulnerabilities such as cross-site scripting (XSS) attacks. If the input is not

properly sanitized or validated, an attacker could inject malicious code that gets executed within your page.

Performance impact: Replacing the entire content of an element using innerHTML can be relatively expensive in terms of performance,
The browser needs to parse and rebuild the entire DOM structure within the element, which can result in reflow and repaint operations, leading to slower rendering and decreased performance.

Loss of event bindings: When you replace the HTML content of an element using innerHTML, any event handlers or bindings attached to the existing elements within that element will be lost. If you have attached event listeners to specific child elements, you will need to reattach them after updating the innerHTML, which can be cumbersome and error-prone.

Potential memory leaks: If you frequently update the innerHTML of an element without properly cleaning up event listeners or other references to the previous content, it can lead to memory leaks. Old DOM elements and their associated resources may not be properly garbage collected, resulting in increased memory usage over time.