1. Explain the advantages and limitations of using JavaScript for client-side scripting.

=> **Advantages of javascript**

1. Less server interaction
2. Immediate feedback to the visitors
3. Increased interactivity
4. Richer interfaces

=> **Limitations of JavaScript**

We cannot treat JavaScript as a full-fledged programming language. It lacks the following important features –

- Client-side JavaScript does not allow the reading or writing of files. This has been kept for security reason.
- JavaScript cannot be used for networking applications because there is no such support available.
- JavaScript doesn't have any multi-threading or multiprocessor capabilities.

JavaScript can be implemented using JavaScript statements that are placed within the **<script>... </script>** HTML tags in a web page.

You can place the **<script>** tags, containing your JavaScript, anywhere within your web page, but it is normally recommended that you should keep it within the **<head>** tags.

The <script> tag alerts the browser program to start interpreting all the text between these tags as a script. A simple syntax of your JavaScript will appear as follows.

```
<script ...>

  JavaScript code

</script>
```

```
<html>
  <body>
    <script language = "javascript" type = "text/javascript">
      <!--
        document.write("Hello World!")
      //-->
    </script>
  </body>
</html>
```

2. Discuss the various methods and properties available in the Math object in JavaScript.

The Math Object

Unlike other objects, the Math object has no constructor.

The Math object is static.

All methods and properties can be used without creating a Math object first.

Number to Integer

There are 4 common methods to round a number to an integer:

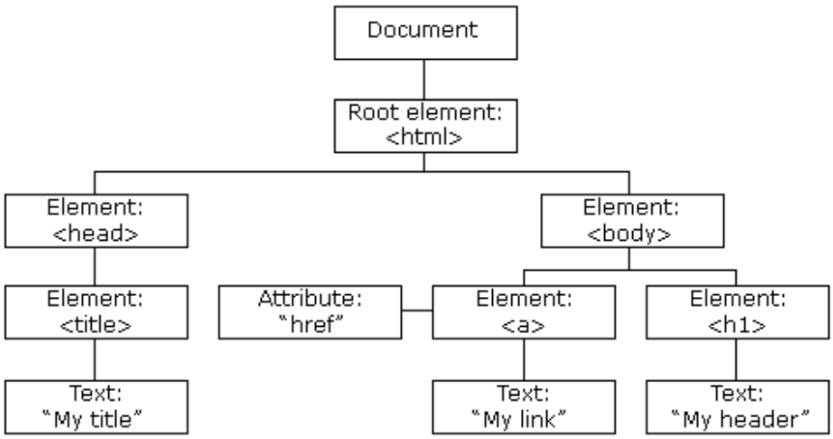
Math.round(x)	Returns x rounded to its nearest integer
Math.ceil(x)	Returns x rounded up to its nearest integer
Math.floor(x)	Returns x rounded down to its nearest integer
Math.trunc(x)	Returns the integer part of x (new in ES6)

3) How does the DOM tree represent an HTML document, and how can JavaScript manipulate it?

When a web page is loaded, the browser creates a **D**ocument **O**bject **M**odel of the page.

The **HTML DOM** model is constructed as a tree of **Objects**:

The HTML DOM Tree of Objects



With the object model, JavaScript gets all the power it needs to create dynamic HTML:

- JavaScript can change all the HTML elements in the page
- JavaScript can change all the HTML attributes in the page
- JavaScript can change all the CSS styles in the page
- JavaScript can remove existing HTML elements and attributes
- JavaScript can add new HTML elements and attributes
- JavaScript can react to all existing HTML events in the page
- JavaScript can create new HTML events in the page

The HTML DOM is a standard **object** model and **programming interface** for HTML. It defines:

- The HTML elements as **objects**
- The **properties** of all HTML elements

In other words: **The HTML DOM is a standard for how to get, change, add, or delete HTML elements.**

Finding HTML Elements

Method	Description
document.getElementById(*id*)	Find an element by element id
document.getElementsByTagName(*name*)	Find elements by tag name
document.getElementsByClassName(*name*)	Find elements by class name

4. Describe the difference between for, while, and do-while loops in JavaScript, with examples.

Feature	`for` Loop	`while` Loop	`do-while` Loop
Syntax	for (initialization; condition; increment/decrement) {}	while (condition) { }	do { } while (condition);
Initialization	Declared within the loop structure and executed once at the beginning.	Declared outside the loop; should be done explicitly before the loop.	Declared outside the loop structure
Condition	Checked before each iteration.	Checked before each iteration.	Checked after each iteration.
Update	Executed after each iteration.	Executed inside the loop; needs to be handled explicitly.	Executed inside the loop; needs to be handled explicitly.
Use Cases	Suitable for a known number of iterations or when looping over ranges.	Useful when the number of iterations is not known in advance or based on a condition.	Useful when the loop block must be executed at least once, regardless of the initial condition.
Initialization and Update Scope	Limited to the loop body.	Scope extends beyond the loop; needs to be handled explicitly.	Scope extends beyond the loop; needs to be handled explicitly.

5. How do JavaScript functions work, and what is the difference between function declarations and function expressions?

- A function declaration also known as a function statement declares a function with a function keyword. The function declaration must have a function name.
- Function declaration does not require a variable assignment as they are standalone constructs and they cannot be nested inside a functional block.
- These are executed before any other code.
- The function in the function declaration can be accessed before and after the function definition.

```
// Function Declaration
```

```
function geeksforGeeks(paramA, paramB) {
    return paramA + paramB;
}
```

Function Declaration	Function Expression
A function declaration must have a function name.	A function expression is similar to a function declaration without the function name.
Function declaration does not require a variable assignment.	Function expressions can be stored in a variable assignment.
These are executed before any other code.	Function expressions load and execute only when the program interpreter reaches the line of code.
The function in function declaration can be accessed before and after the function definition.	The function in function expression can be accessed only after the function definition.
Function declarations are hoisted	Function expressions are not hoisted
Syntax: function geeksforGeeks(paramA, paramB) { // Set of statements }	**Syntax:** var geeksforGeeks= function(paramA, paramB) { // Set of statements }

6. Explain the process of creating, accessing, and modifying objects in JavaScript.

8. How can you create and manipulate date objects in JavaScript? Explain with examples.

Date objects are created with the new Date() constructor.

There are 4 ways to create a new date object:

new Date()

new Date(year, month, day, hours, minutes, seconds, milliseconds)

new Date(milliseconds)

new Date(date string)

new Date()

new Date()

creates a new date object with the current date and time:

Example

const d = new Date();

Date objects are static. The computer time is ticking, but date objects are not.

new Date(year, month, ...)

new Date(year, month, ...)

creates a new date object with a specified date and time.

7 numbers specify year, month, day, hour, minute, second, and millisecond (in that order):

7. Describe the different ways to handle events in JavaScript. Provide examples.

Handle event in HTML :

```
<element onclick="myScript">
```

Various HTML event attributes:

Form Event

- **onblur:** This event occurs when an object loses focus.

```
<element onblur="myScript">
```

- **onchange:** This event occurs when the value of an element has been changed.

```
<element onchange="myScript">
```

- **onfocus:** This event occurs when element get focus.

```
<element onfocus="myScript">
```

```
<script>
    function BlurFunction() {
        let x = document.getElementById("blur");
        x.value = x.value.toUpperCase();
    }

    function OnChangeFunction() {
        let x = document.getElementById("course").value;
        document.getElementById("demo")
            .innerHTML = "You selected: " + x;
    }

    function FocusFunction(x) {
        document.getElementById(x)
            .style.background = "green";
    }
</script>
```

- **onclick:** This event occurs when user clicks on an element.

```
<element onclick="myScript">
```

- **onmouseover:** This event occurs when user hover mouse pointer over an element.

```
<element onmouseover="myScript">
```

9. Discuss the different dialog boxes available in JavaScript and their use cases.

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

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

window.alert("sometext");

The window.alert() method can be written without the window prefix.

Example

alert("I am an alert box!");

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");

The window.confirm() method can be written without the window prefix.

Example

```
if (confirm("Press a button!")) {
txt = "You pressed OK!";
} else {
txt = "You pressed Cancel!";
}
```

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");

The window.prompt() method can be written without the window prefix.

```
let person = prompt("Please enter your name", "Harry Potter");
let text;
if (person == null || person == "") {
text = "User cancelled the prompt.";
} else {
text = "Hello " + person + "! How are you today?";
}
```

10. Explain the purpose and use of conditional statements in JavaScript. Provide examples of if, else, else if, and switch statements.

1. If Statement:

if statement to specify a block of JavaScript code to be executed if a condition is true.

Example:

```
if (hour < 18) {
greeting = "Good day";
}
```

2.If..Else Statement:

The else Statement

Use the `else` statement to specify a block of code to be executed if the condition is false.

```
if (condition) {
  //  block of code to be executed if the condition is true
} else {
  //  block of code to be executed if the condition is false
}
```

3. If..ElseIf.. Statement:

Use the else if statement to specify a new condition if the first condition is false.

Example:

```
if (time < 10) {
  greeting = "Good morning";
} else if (time < 20) {
  greeting = "Good day";
} else {
  greeting = "Good evening";
}
```

4. Switch Statement

In Switch statement, switch block is used to select one of many code blocks to be executed.

Syntax:

```
switch(expression) {

case x:

   // code block

break;

case y:

   // code block

 break;

 default:

    // code block

 }
```