

## Module 6) JAVASCRIPT BASIC & DOM (Basic logic Question)

### Q.1 What is JavaScript? How to use it?

- Javascript is a scripting language. It was designed to add interactivity to HTML Pages.
- Javascript is an interpreted language that means the script executes without preliminary compilation.
- It's code directly embedded to the HTML Page. It works for all major browsers.
- Javascript is a client side, interpreted, Object Oriented, high level scripting language which cannot run stand alone.
- It was created By Netscape in 1995, evolved from Netscape's Live Script.
- Javascript works with HTML and CSS (Cascading Style Sheets).
- You can attach javascript file to html file using two methods:  
Internal and external file.

- **Internal javascript**

- **It can be written in head tag or in body tag both ways.**

```
<html>
<head>
<title></title>
<script src="path">
– write your code here –
</script>
</head>
<body></body>
</html>
```

```
<html>
<head>
<title></title>
</head>
<body>
<script src="path">
– write your code here –
</script>
</body>
</html>
```

- **External file**

- **Make an external js file and attach it in html.**

```
<html>
<head>
<title></title>
<script src="script.js"></script>
```

```

</head>
<body></body>
</html>

```

## Q.2 How many types of Variables in JavaScript?

- Variables are containers for storing data.
- Javascript variables can be define in 4 ways:
  - Using **const** = use const if the value should not be changed.
  - Using **var** = Only use var if you MUST support old browsers.
  - Using **let** = Only use let if you can't use const.
- The var keyword was used in all JavaScript code from 1995 to 2015.
- The let and const keywords were added to JavaScript in 2015.
- The var keyword should only be used in code written for older browsers.

## Q.3 Define a Data Types in js?

- In programming, data types are an important concept.
- To be able to operate on variables, it is important to know something about the type.
- JavaScript has 8 Data Types:
- **String,Number,Bigint,Boolean,Undefined,Null,Symbol,Object.**

## Q.4 Write a mul Function Which will Work Properly When invoked With Following Syntax.

```

function mul() {
  // If no arguments are there return 0
  if (arguments.length === 0) {
    return 0;
  }
  // If there's only one arguments return
  if (arguments.length === 1) {
    return arguments[0];
  }
  // If there are multiple arguments then multiply them
  let result = 1;
  for (let i = 0; i < arguments.length; i++) {
    result *= arguments[i];
  }
  return result;
}
console.log(mul());
console.log(mul(2));
console.log(mul(2, 3));
console.log(mul(2, 3, 4));

```

PS E:\Jayna Projects\front-end course\javascript\programs> node script.js  
0  
2  
6  
24

## Q.5 What is the difference between undefined and undeclared in JavaScript?

- **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.

**Q.6 Using console.log() print out the following statement: The quote 'There is no exercise better for the heart than reaching down and lifting people up.' by John Holmes teaches us to help one another. Using console.log() print out the following quote by Mother Teresa:**

The screenshot shows a dark-themed VS Code interface. In the center is a terminal window titled 'powershell'. The code in 'script.js' is as follows:

```
1 console.log("There is no exercise better for the heart than reaching down and lifting people up.");
2 console.log("Peace begins with a smile");
```

The terminal output shows the two lines of text printed to the console:

```
PS E:\Jayna Projects\front-end course\javascript\programs> node script.js
There is no exercise better for the heart than reaching down and lifting people up.
Peace begins with a smile
PS E:\Jayna Projects\front-end course\javascript\programs>
```

**Q.7 Check if typeof '10' is exactly equal to 10. If not make it exactly equal?**

The screenshot shows a dark-themed VS Code interface with the 'EXPLORER' sidebar open, displaying files like 'script.js' and 'compareNumber'. The terminal window is titled 'powershell' and contains the following code in 'script.js':

```
1 // Check if typeof '10' is exactly equal to 10
2 if (typeof '10' !== 'number') {
3     // Convert '10' to a number
4     var num = parseInt('10');
5
6     // Define the number to compare with
7     var compareNumber = 10;
8
9     // Check if num is exactly equal to compareNumber
10    if (num === compareNumber) {
11        console.log("'10' is exactly equal to " + compareNumber);
12    } else {
13        console.log("'10' is not exactly equal to " + compareNumber);
14    }
15 } else {
16     console.log("No conversion needed");
17 }
```

The terminal output shows the script running and printing the result of the comparison:

```
PS E:\Jayna Projects\front-end course\javascript\programs> node script.js
'10' is not exactly equal to 5
PS E:\Jayna Projects\front-end course\javascript\programs> node script.js
'10' is exactly equal to 10
PS E:\Jayna Projects\front-end course\javascript\programs>
```

## Q.8 Write a JavaScript Program to find the area of a triangle?

The screenshot shows a dark-themed instance of Visual Studio Code. In the Explorer sidebar, there is one open editor named "script.js". The code in the editor is as follows:

```
function triangleArea(side1, side2, side3) {
    let s = (side1 + side2 + side3) / 2;

    // Heron's formula
    return Math.sqrt(s * (s - side1) * (s - side2) * (s - side3));
}

let side1Length = 5;
let side2Length = 4;
let side3Length = 3;
let area = triangleArea(side1Length, side2Length, side3Length);
console.log(`The area of the triangle is: ${area}`);
```

Below the editor, the terminal window shows the output of running the script:

```
PS E:\Jayna Projects\front-end course\javascript\programs> node script.js
The area of the triangle is: 6
PS E:\Jayna Projects\front-end course\javascript\programs>
```

## Q.9 Write a JavaScript program to calculate days left until next Christmas?

The screenshot shows a dark-themed instance of Visual Studio Code. In the Explorer sidebar, there is one open editor named "script.js". The code in the editor is as follows:

```
function daysUntilChristmas() {
    // Get today's date
    var today = new Date();

    // Get the current year
    var currentYear = today.getFullYear();

    // Set the date for Christmas
    var christmasDate = new Date(currentYear, 11, 25); // Month is 0-indexed, so 11 represents December

    // Check if Christmas has already passed this year
    if (today.getMonth() === 11 && today.getDate() > 25) {
        // If Christmas has passed, set the Christmas date for next year
        christmasDate.setFullYear(currentYear + 1);
    }

    // Calculate the difference in milliseconds between today and Christmas
    var timeDiff = christmasDate.getTime() - today.getTime();

    // Convert milliseconds to days
    var daysLeft = Math.ceil(timeDiff / (1000 * 3600 * 24));
    return daysLeft;
}

// Test the function
console.log("Days left until next Christmas:", daysUntilChristmas());
```

Below the editor, the terminal window shows the output of running the script:

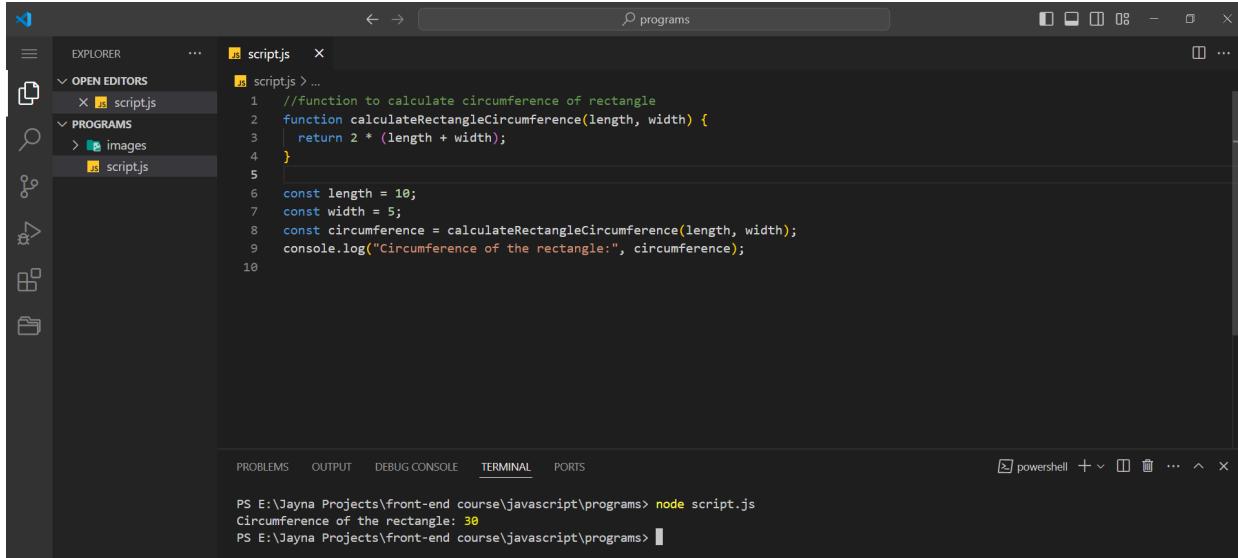
```
PS E:\Jayna Projects\front-end course\javascript\programs> node script.js
Days left until next Christmas: 265
PS E:\Jayna Projects\front-end course\javascript\programs>
```

## Q.10 What is a Condition Statement?

- Conditional statement is very often used when you write code and you want to perform different actions for different decisions.
- In JavaScript these are the following conditional statements:
  - Use **if** to specify a block of code to be executed, if a specified condition is true

- Use **else** to specify a block of code to be executed, if the same condition is false
- Use **else if** to specify a new condition to test, if the first condition is false
- Use **switch** to specify many alternative blocks of code to be executed.

### Q.11 Find circumference of Rectangle formula : C = 4 \* a ?



The screenshot shows a dark-themed instance of Visual Studio Code. In the Explorer sidebar, there are two open files: 'script.js' and 'script.js'. The main editor area contains the following JavaScript code:

```

1 //function to calculate circumference of rectangle
2 function calculateRectangleCircumference(length, width) {
3     return 2 * (length + width);
4 }
5
6 const length = 10;
7 const width = 5;
8 const circumference = calculateRectangleCircumference(length, width);
9 console.log("circumference of the rectangle:", circumference);
10

```

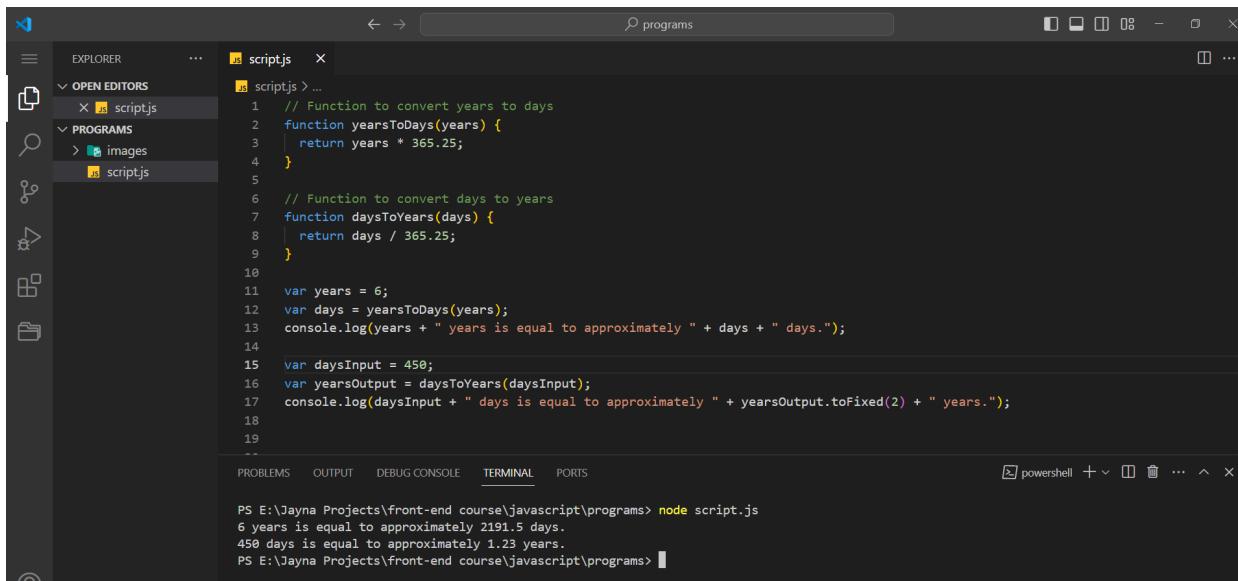
Below the editor, the terminal window shows the output of running the script:

```

PS E:\Jayna Projects\front-end course\javascript\programs> node script.js
Circumference of the rectangle: 30
PS E:\Jayna Projects\front-end course\javascript\programs>

```

### Q.12 WAP to convert years into days and days into years?



The screenshot shows a dark-themed instance of Visual Studio Code. In the Explorer sidebar, there are two open files: 'script.js' and 'script.js'. The main editor area contains the following JavaScript code:

```

1 // Function to convert years to days
2 function yearsToDays(years) {
3     return years * 365.25;
4 }
5
6 // Function to convert days to years
7 function daysToYears(days) {
8     return days / 365.25;
9 }
10
11 var years = 6;
12 var days = yearsToDays(years);
13 console.log(years + " years is equal to approximately " + days + " days.");
14
15 var daysInput = 450;
16 var yearsOutput = daysToYears(daysInput);
17 console.log(daysInput + " days is equal to approximately " + yearsOutput.toFixed(2) + " years.");
18
19

```

Below the editor, the terminal window shows the output of running the script:

```

PS E:\Jayna Projects\front-end course\javascript\programs> node script.js
6 years is equal to approximately 2191.5 days.
450 days is equal to approximately 1.23 years.
PS E:\Jayna Projects\front-end course\javascript\programs>

```

### Q.13 Convert temperature Fahrenheit to Celsius? (Conditional logic Question)

```

scriptjs > ...
1 //function declaration to convert
2 function fahrenheitToCelsius(fahrenheit){
3
4     var celsius;
5
6     if (typeof fahrenheit === 'number'){
7         celsius = (5/9) * (fahrenheit - 32)
8         return celsius;
9     }
10    else{
11        return 'please provide a valid number';
12    }
13 }
14
15 var fahrenheitTemperature = 80;
16 var celsiusTemperature = fahrenheitToCelsius(fahrenheitTemperature);
17 console.log(fahrenheitTemperature + "F is equal to " + celsiusTemperature.toFixed(2) + "c");

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS E:\Jayna Projects\front-end course\javascript\programs> node script.js  
80F is equal to 26.67c  
PS E:\Jayna Projects\front-end course\javascript\programs>

#### Q.14 Write a JavaScript exercise to get the extension of a filename.?

```

scriptjs > ⚡ getFileExtension
1 function getFileExtension(filename) {
2
3     // Split the filename by '.'
4     var parts = filename.split('.');
5
6     // If the filename has at least one dot and the last part is not empty
7     if (parts.length > 1 && parts[parts.length - 1] !== '') {
8
9         // Return the last part
10        return parts[parts.length - 1];
11    } else {
12
13        // If no extension found, return an empty string
14        return '';
15    }
16
17 var filename1 = 'index.txt';
18 var filename2 = 'script.js';
19 var filename3 = 'image.png';
20 var filename4 = 'style.css';
21
22 console.log("Extension of " + filename1 + ":", getFileExtension(filename1));
23 console.log("Extension of " + filename2 + ":", getFileExtension(filename2));
24 console.log("Extension of " + filename3 + ":", getFileExtension(filename3));
25 console.log("Extension of " + filename4 + ":", getFileExtension(filename4));

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS E:\Jayna Projects\front-end course\javascript\programs> node script.js  
Extension of index.txt: txt  
Extension of script.js: js  
Extension of image.png: png  
Extension of style.css: css  
PS E:\Jayna Projects\front-end course\javascript\programs>

#### Q.15 What is the result of the expression (5 > 3 && 2 < 4)?

- The expression `(5>3 && 2<4)` is **true**. Because the condition `5>3` is true and `2<4` is true.
- So, the AND operator returns true because both the conditions are true.

#### Q.16 What is the result of the expression (true && 1 && "hello")?

- The result of the expression is **"hello"**.

- Javascript evaluates `&&` expression from left to right. If all operands are truthy, it returns the value of the last operand.
- So in this case:
  - `true` is truthy.
  - `1` is truthy.
  - `"hello"` is also truthy.

#### **Q.17 What is the result of the expression `true && false || false && true`?**

- The result of the expression is **false**.
- Javascript evaluates logical AND (`&&`) expression before logical OR (`||`) expression.
- Evaluation step by step:
  - `True && false`: This evaluates false because one of the operands is false.
  - `False && true`: This evaluates false for the same reason.
  - `False || false`: This evaluates false because both operands are false.
- So, the final result is false.

#### **Q.18 What is a Loop and Switch Case in JavaScript define that ?**

- In JavaScript, a loop is a programming construct that allows you to execute a block of code repeatedly until a certain condition is met.
- There are several types of loops in JavaScript, including the `for` loop, `while` loop, and `do-while` loop.
- **for loop:** The `for` loop is used when you know the number of times you want to execute a block of code. It consists of three parts: initialization, condition, and iteration. The loop continues to execute as long as the condition evaluates to true.

**Syntax:**

```
for (initialization; condition; iteration) {
  // Code to be executed
}
```

- **while loop:** The `while` loop is used when you want to execute a block of code as long as a condition is true. The loop evaluates the condition before executing the block of code.

**Syntax:**

```
while (condition) {
  // Code to be executed
}
```

- **do-while loop:** The `do-while` loop is similar to the `while` loop, but it always executes the block of code at least once before checking the condition. The loop continues to execute as long as the condition evaluates to true.

**Syntax:**

```
do {
  // Code to be executed
} while (condition);
```

- **Switch case** is another control flow statement in JavaScript used to perform different actions based on different conditions. It provides a cleaner and more organized alternative to using multiple if...else statements.

**Syntax:**

```
switch (expression) {
  case value1:
    value1
    break;
  case value2:
    value2
    break;
  default:
}
```

### Q.19 What is the use of is Nan function?

- In javascript NAN stands for “Not a number”.
- It is a special value that represents the result of an operation that cannot produce a meaningful numeric result.
- The NAN value is returned when:
  - Performing arithmetic operations involving non-numeric values or invalid calculations.
  - Converting non-numeric strings to numbers using parseInt() or parseFloat() functions when the string cannot be parsed as a valid number.

### Q.20 What is the difference between && and || in JavaScript?

- In javascript && and || are both logical operators used for boolean operations, but they behave differently:

**&&(Logical AND):**

- The && operator returns true if both operands are true, otherwise it returns false.
- It short-circuits: if the left operand is false, the right operand is not evaluated because the result will always be false.
- Example: true && true returns true, true && false returns false, false && true returns false, false && false returns false.

**|| (Logical OR):**

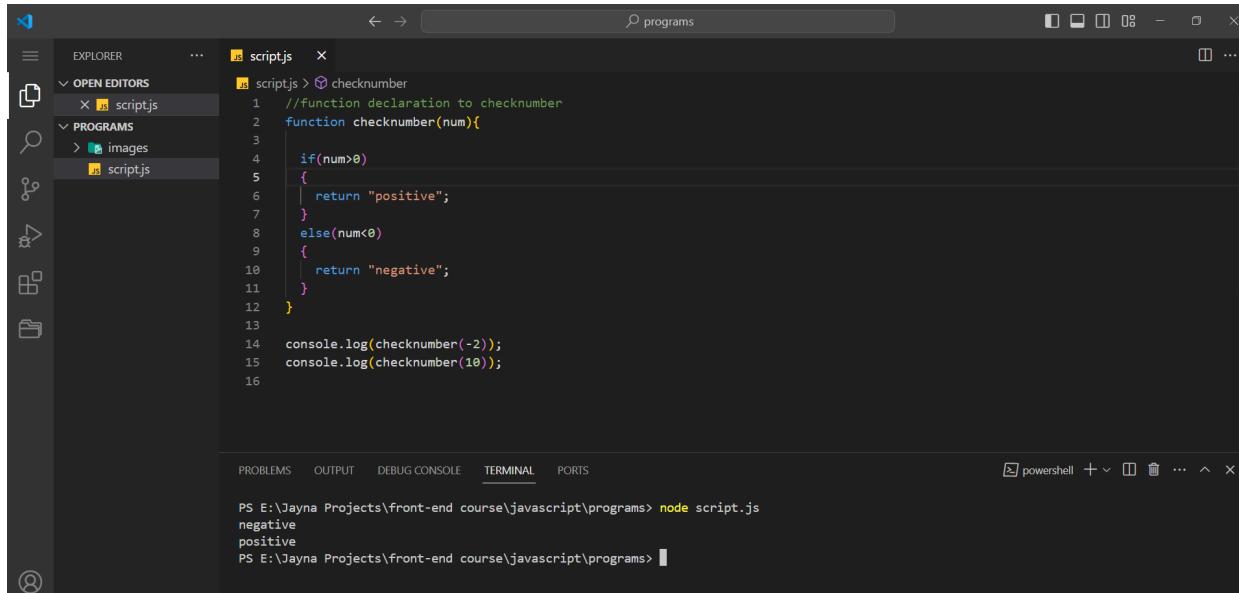
- The || operator returns true if at least one of the operands is true, otherwise it returns false.
- It short-circuits: if the left operand is true, the right operand is not evaluated because the result will always be true.
- Example: true || true returns true, true || false returns true, false || true returns true, false || false returns false.

### Q.21 What is the use of Void (0)?

- In javascript void(0) is an expression that evaluates to undefined.
- It's often used to return undefined from an expression context.

- The primary use case of void(0) is to create an expression that returns undefined without explicitly using the undefined keyword.
- It can be useful in scenarios where you want to ensure that a function or event handler doesn't inadvertently return a value.

## Q.22 Check Number Is Positive or Negative in JavaScript?



```

script.js > checknumber
1 //function declaration to checknumber
2 function checknumber(num){
3
4     if(num>0)
5     {
6         return "positive";
7     }
8     else(num<0)
9     {
10        return "negative";
11    }
12 }
13
14 console.log(checknumber(-2));
15 console.log(checknumber(10));
16

```

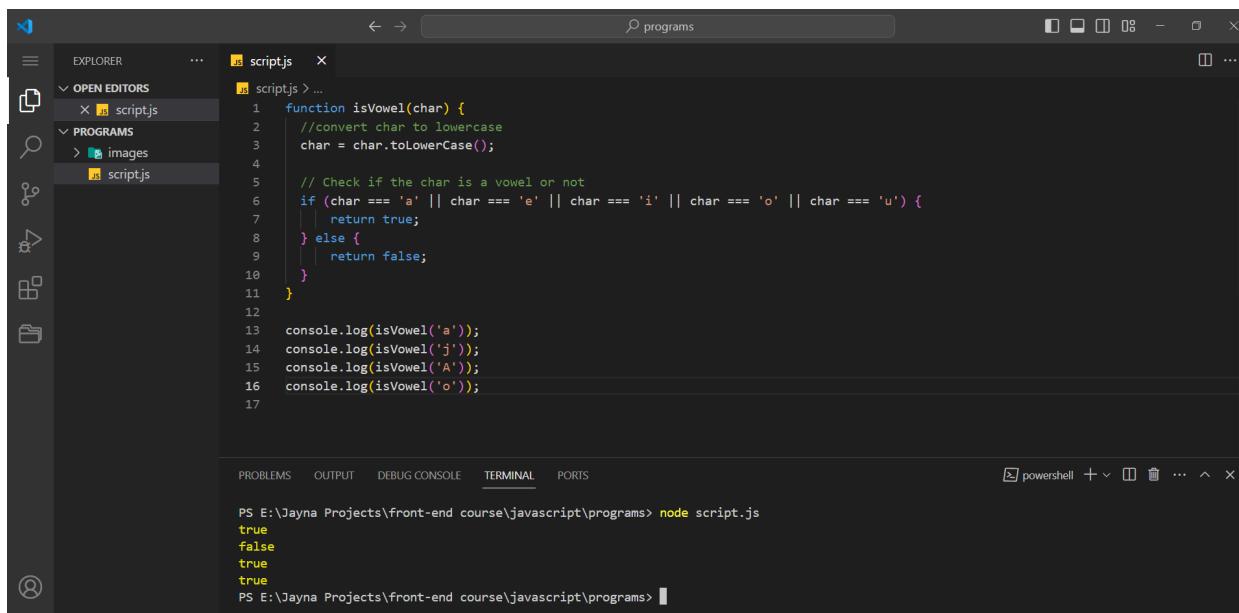
TERMINAL

```

PS E:\Jayna Projects\front-end course\javascript\programs> node script.js
negative
positive
PS E:\Jayna Projects\front-end course\javascript\programs>

```

## Q.23 Find the Character Is Vowel or Not ?



```

script.js > ...
1 function isVowel(char) {
2     //convert char to lowercase
3     char = char.toLowerCase();
4
5     // Check if the char is a vowel or not
6     if (char === 'a' || char === 'e' || char === 'i' || char === 'o' || char === 'u') {
7         return true;
8     } else {
9         return false;
10    }
11 }
12
13 console.log(isVowel('a'));
14 console.log(isVowel('j'));
15 console.log(isVowel('A'));
16 console.log(isVowel('o'));
17

```

TERMINAL

```

PS E:\Jayna Projects\front-end course\javascript\programs> node script.js
true
false
true
true
PS E:\Jayna Projects\front-end course\javascript\programs>

```

### **Q.24 Write to check whether a number is negative, positive or zero?**

The screenshot shows the VS Code interface with the following details:

- Explorer View:** Shows an open editor for "script.js" and a folder named "images".
- Editor Content:** The code defines a function `checknumber` that takes a parameter `num`. It checks if the number is greater than zero, less than zero, or equal to zero, returning "positive", "negative", or "zero" respectively. It then logs the results to the console.
- Terminal Output:** The terminal shows the execution of `node script.js` followed by three lines of output: "negative", "positive", and "zero".

```
//function declaration to checknumber
function checknumber(num){
    if(num>0){
        return "positive"
    }if(num<0){
        return "negative"
    }else{
        return "zero"
    }
}
console.log(checknumber(-2));
console.log(checknumber(10));
console.log(checknumber(0));
```

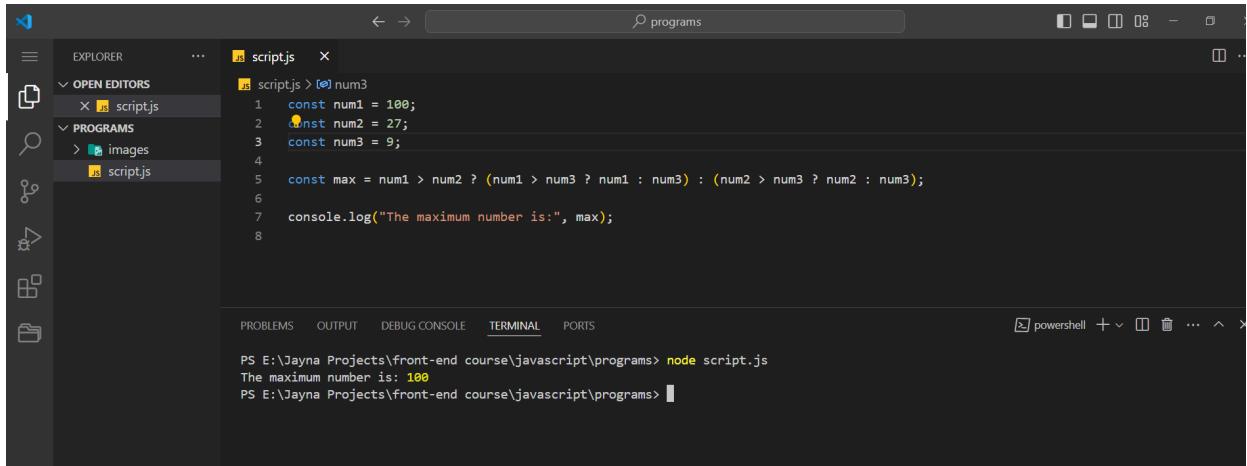
### **Q.25 Write to find number is even or odd using ternary operator in JS?**

The screenshot shows the VS Code interface with the following details:

- Explorer View:** Shows an open editor for "script.js" and a folder named "images".
- Editor Content:** The code uses a ternary operator to determine if a number is even or odd. It then logs the result to the console.
- Terminal Output:** The terminal shows the execution of `node script.js` followed by two lines of output: "10 is even" and "5 is odd".

```
const number = 5;
const result = number % 2 === 0 ? "even" : "odd";
console.log(`${number} is ${result}.`);
```

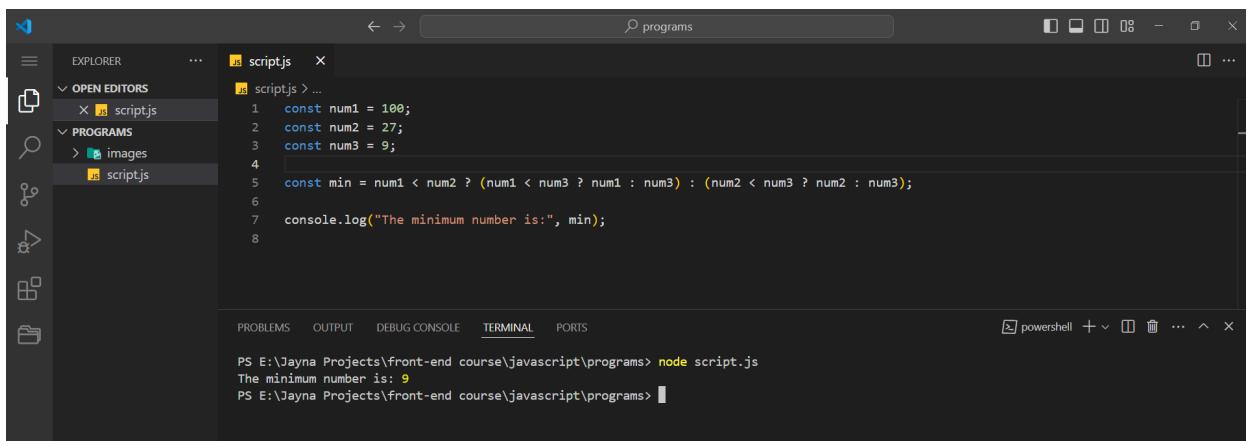
### **Q.26 Write find maximum number among 3 numbers using ternary operator in JS?**



```
script.js
1 const num1 = 100;
2 const num2 = 27;
3 const num3 = 9;
4
5 const max = num1 > num2 ? (num1 > num3 ? num1 : num3) : (num2 > num3 ? num2 : num3);
6
7 console.log("The maximum number is:", max);

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS E:\Jayna Projects\front-end course\javascript\programs> node script.js
The maximum number is: 100
PS E:\Jayna Projects\front-end course\javascript\programs>
```

**Q.27 Write to find minimum number among 3 numbers using ternary operator in JS?**



```
script.js
1 const num1 = 100;
2 const num2 = 27;
3 const num3 = 9;
4
5 const min = num1 < num2 ? (num1 < num3 ? num1 : num3) : (num2 < num3 ? num2 : num3);
6
7 console.log("The minimum number is:", min);

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS E:\Jayna Projects\front-end course\javascript\programs> node script.js
The minimum number is: 9
PS E:\Jayna Projects\front-end course\javascript\programs>
```

**Q.28 Write to find the largest of three numbers in JS?**

```
function findLargest(num1, num2, num3) {
    let largest = num1;
    if (num2 > largest) {
        largest = num2;
    }
    if (num3 > largest) {
        largest = num3;
    }
    return largest;
}

const num1 = 101;
const num2 = 201;
const num3 = 150;
const largestNumber = findLargest(num1, num2, num3);
console.log("The largest number is:", largestNumber);
```

PS E:\Jayna Projects\front-end course\javascript\programs> node script.js  
The largest number is: 201  
PS E:\Jayna Projects\front-end course\javascript\programs>

## Q.29 Write a program to show :

### i. Monday to Sunday using switch case in JS?

```
function getDayName(dayNumber) {
    let dayName;
    switch(dayNumber) {
        case 1:
            dayName = "Monday";
            break;
        case 2:
            dayName = "Tuesday";
            break;
        case 3:
            dayName = "Wednesday";
            break;
        case 4:
            dayName = "Thursday";
            break;
        case 5:
            dayName = "Friday";
            break;
        case 6:
            dayName = "Saturday";
            break;
        case 7:
            dayName = "Sunday";
            break;
        default:
            dayName = "Invalid day number";
    }
    return dayName;
}
```

A screenshot of the Visual Studio Code interface. The left sidebar shows the Explorer and Open Editors sections, with 'script.js' listed under 'OPEN EDITORS'. The main editor area displays the following JavaScript code:

```

31
32  for (let i = 1; i <= 7; i++) {
33    console.log(`Day ${i}: ${getDayName(i)})`);
34  }
35

```

The terminal at the bottom shows the output of running the script with 'node script.js':

```

PS E:\Jayna Projects\front-end course\javascript\programs> node script.js
Day 1: Monday
Day 2: Tuesday
Day 3: Wednesday
Day 4: Thursday
Day 5: Friday
Day 6: Saturday
Day 7: Sunday
PS E:\Jayna Projects\front-end course\javascript\programs>

```

## ii. Vowel or Consonant using switch case in JS?

A screenshot of the Visual Studio Code interface. The left sidebar shows the Explorer and Open Editors sections, with 'script.js' listed under 'OPEN EDITORS'. The main editor area displays the following JavaScript code:

```

1  function checkVowelOrConsonant(character) {
2    switch(character.toLowerCase()) {
3      case 'a':
4      case 'e':
5      case 'i':
6      case 'o':
7      case 'u':
8        return "Vowel";
9      default:
10        return "Consonant";
11    }
12  }
13
14 const testCharacter = 'j';
15 console.log(`${testCharacter} is a ${checkVowelOrConsonant(testCharacter)}.`);
16

```

The terminal at the bottom shows the output of running the script with 'node script.js':

```

PS E:\Jayna Projects\front-end course\javascript\programs> node script.js
o is a Vowel.
PS E:\Jayna Projects\front-end course\javascript\programs> node script.js
A is a Vowel.
PS E:\Jayna Projects\front-end course\javascript\programs> node script.js
j is a Consonant.
PS E:\Jayna Projects\front-end course\javascript\programs>

```

## (Conditional looping logic Question)

### Q.30 What are the looping structures in JavaScript? Any one Example?

- JavaScript Loops are powerful tools for performing repetitive tasks efficiently. Loops in JavaScript execute a block of code again and again while the condition is true.
- Looping structure in javascript are:
  - 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 values 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

- Example:

```
for (let i = 0; i < 5; i++) {
  console.log(i);
}
```

### Q.31 Write a print 972 to 897 using for loop in JS?

The screenshot shows a dark-themed interface of VS Code. On the left is the Explorer sidebar with a 'PROGRAMS' section containing a file named 'script.js'. The main area is a terminal window with the following content:

```
PS E:\Jayna Projects\front-end course\javascript\programs> node script.js
972
971
970
969
968
967
966
965
964
963
962
961
960
959
958
957
956
955
954
953
952
951
950
949
948
```

The status bar at the bottom indicates 'Ln 1, Col 1' and other settings like 'Spaces: 4', 'UTF-8', and 'JavaScript'.

### Q.32 Write to print the factorial of a given number?

The screenshot shows a dark-themed interface of VS Code. On the left is the Explorer sidebar with a 'PROGRAMS' section containing a file named 'script.js'. The main area is a terminal window with the following content:

```
function factorial(number) {
  if (number < 0) {
    return "Factorial is not defined for negative numbers";
  }
  if (number === 0) {
    return 1;
  }
  let result = 1;
  for (let i = 1; i <= number; i++) {
    result *= i;
  }
  return result;
}

const num = 5;
console.log("Factorial of", num, "is", factorial(num));
```

Below the code, the terminal shows the output:

```
PS E:\Jayna Projects\front-end course\javascript\programs> node script.js
Factorial of 5 is 120
PS E:\Jayna Projects\front-end course\javascript\programs>
```

The status bar at the bottom indicates 'Ln 1, Col 1' and other settings like 'Spaces: 4', 'UTF-8', and 'JavaScript'.

### Q.33 Write to print Fibonacci series up to given numbers?

A screenshot of the Visual Studio Code interface. The left sidebar shows a tree view with 'PROGRAMS' expanded, containing 'script.js'. The main editor area displays the following code:

```
function fibonacciSeries(limit) {
    let prev = 0;
    let curr = 1;
    console.log(prev);
    console.log(curr);
    while (true) {
        let next = prev + curr;
        if (next <= limit) {
            console.log(next);
            prev = curr;
            curr = next;
        } else {
            break;
        }
    }
const limit = 20;
console.log("Fibonacci series up to", limit, "is:");
fibonacciSeries(limit);
```

The terminal below the editor shows the output of running the script:

```
Fibonacci series up to 20 is:
0
1
1
2
3
5
8
13
```

The status bar at the bottom indicates the file path: PS E:\Jayna Projects\front-end course\javascript\programs>

### Q.34 Write to print number in reverse order e.g.: number = 64728 ---> reverse =82746 in JS?

A screenshot of the Visual Studio Code interface. The left sidebar shows a tree view with 'PROGRAMS' expanded, containing 'script.js'. The main editor area displays the following code:

```
function reverseNumber(number) {
    const reversedString = number.toString().split('').reverse().join('');
    const reversedNumber = parseInt(reversedString);
    return reversedNumber;
}
const number = 64728;
const reversedNumber = reverseNumber(number);
console.log("Original number:", number);
console.log("Reversed number:", reversedNumber);
```

The terminal below the editor shows the output of running the script:

```
PS E:\Jayna Projects\front-end course\javascript\programs> node script.js
Original number: 64728
Reversed number: 82746
PS E:\Jayna Projects\front-end course\javascript\programs>
```

### Q.35 Write a program make a summation of given number (E.g., 1523 Ans: - 11) in JS?

The screenshot shows a dark-themed interface of VS Code. In the Explorer sidebar, there's a 'PROGRAMS' folder containing a 'script.js' file. The main editor area displays the following JavaScript code:

```
function digitSum(number) {
    const numString = number.toString();
    let sum = 0;

    for (let i = 0; i < numString.length; i++) {
        sum += parseInt(numString[i]);
    }

    return sum;
}

const number = 1523;
const sum = digitSum(number);
console.log("Summation of digits of", number, "is:", sum);
```

Below the editor, the terminal tab is active, showing the output of running the script: "Summation of digits of 1523 is: 11".

**Q.36 Write a program you have to make a summation of first and last Digit. (E.g., 1234 Ans: - 5) in JS?**

The screenshot shows a dark-themed interface of VS Code. In the Explorer sidebar, there's a 'PROGRAMS' folder containing a 'script.js' file. The main editor area displays the following JavaScript code:

```
function sumFirstLastDigit(number) {
    const lastDigit = number % 10;

    let firstDigit = number;
    while (firstDigit >= 10) {
        firstDigit = Math.floor(firstDigit / 10);
    }

    const sum = firstDigit + lastDigit;
    return sum;
}

const number = 1234;
const sum = sumFirstLastDigit(number);
console.log("Sum of first and last digit of", number, "is:", sum);
```

Below the editor, the terminal tab is active, showing the output of running the script: "Sum of first and last digit of 1234 is: 5".

**Q.37 Use console.log() and escape characters to print the following pattern in JS?**

1 1 1 1 1  
2 1 2 4 8  
3 1 3 9 27  
4 1 4 16 64  
5 1 5 25 125

The screenshot shows the VS Code interface. In the Explorer sidebar, there is a 'PROGRAMS' folder containing a 'script.js' file. The code in 'script.js' is as follows:

```
1 console.log('1 1 1 1 1');
2 console.log('2 1 2 4 8');
3 console.log('3 1 3 9 27');
4 console.log('4 1 4 16 64');
5 console.log('5 1 5 25 125');
```

The terminal below shows the output of running the script with 'node script.js':

```
PS E:\Jayna Projects\front-end course\javascript\programs> node script.js
1 1 1 1
2 1 2 4 8
3 1 3 9 27
4 1 4 16 64
5 1 5 25 125
PS E:\Jayna Projects\front-end course\javascript\programs>
```

### Q.38 Use pattern in console.log in JS?

1) 1  
1 0  
1 0 1  
1 0 1 0  
1 0 1 0 1

The screenshot shows the VS Code interface. In the Explorer sidebar, there is a 'PROGRAMS' folder containing a 'script.js' file. The code in 'script.js' is as follows:

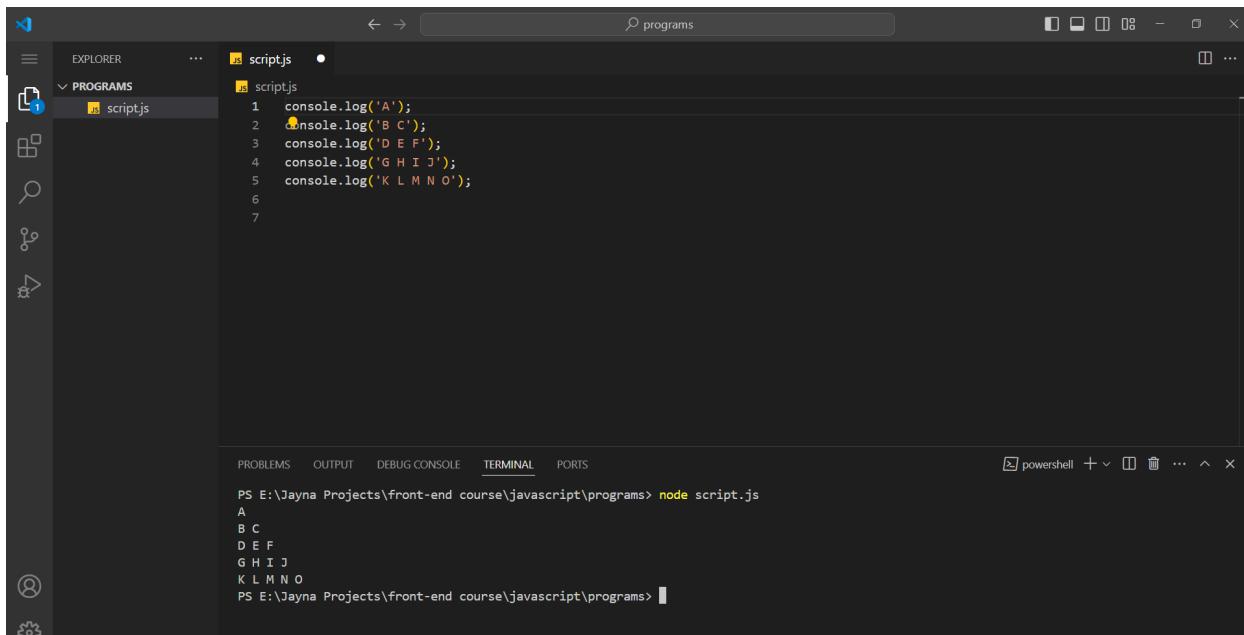
```
1 console.log('1');
2 console.log('1 0');
3 console.log('1 0 1');
4 console.log('1 0 1 0');
5 console.log('1 0 1 0 1');
```

The terminal below shows the output of running the script with 'node script.js':

```
PS E:\Jayna Projects\front-end course\javascript\programs> node script.js
1
1 0
1 0 1
1 0 1 0
1 0 1 0 1
PS E:\Jayna Projects\front-end course\javascript\programs>
```

2) A  
B C  
D E F

G H I J  
K L M N O



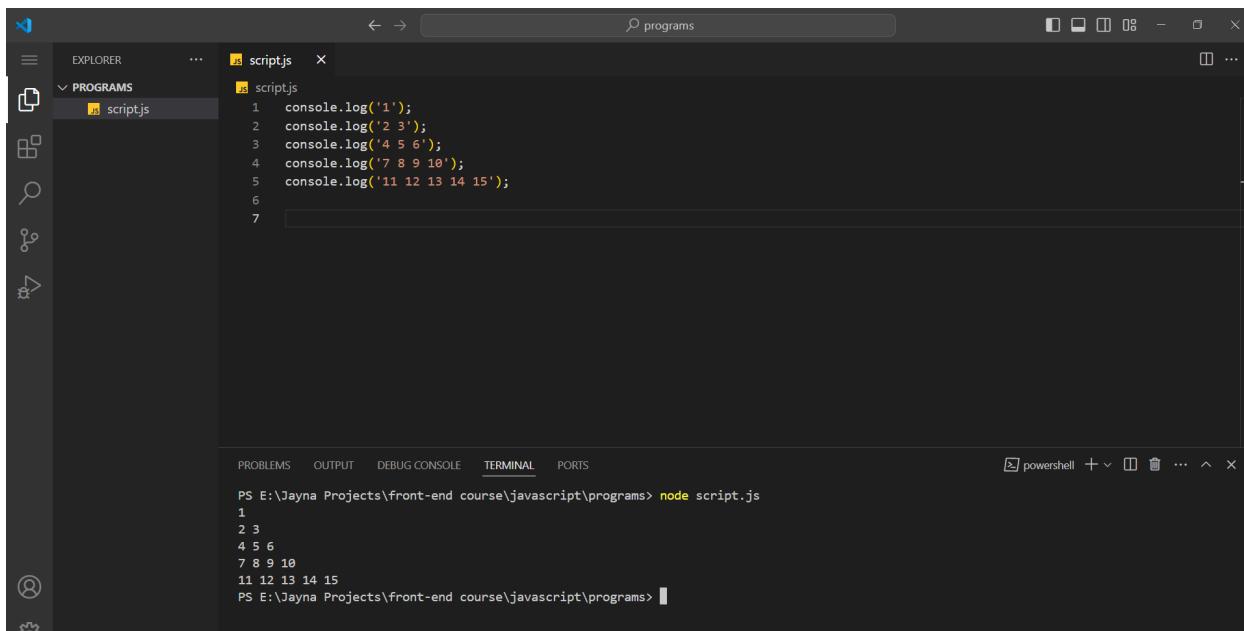
A screenshot of the Visual Studio Code interface. The Explorer sidebar shows a folder named 'PROGRAMS' containing a file 'script.js'. The code editor displays the following JavaScript code:

```
1 console.log('A');
2 console.log('B C');
3 console.log('D E F');
4 console.log('G H I J');
5 console.log('K L M N O');
```

The terminal at the bottom shows the output of running the script with 'node script.js':

```
PS E:\Jayna\Projects\front-end course\javascript\programs> node script.js
A
B C
D E F
G H I J
K L M N O
PS E:\Jayna\Projects\front-end course\javascript\programs>
```

3) 1  
2 3  
4 5 6  
7 8 9 10  
11 12 13 14 15



A screenshot of the Visual Studio Code interface. The Explorer sidebar shows a folder named 'PROGRAMS' containing a file 'script.js'. The code editor displays the following JavaScript code:

```
1 console.log('1');
2 console.log('2 3');
3 console.log('4 5 6');
4 console.log('7 8 9 10');
5 console.log('11 12 13 14 15');
```

The terminal at the bottom shows the output of running the script with 'node script.js':

```
PS E:\Jayna\Projects\front-end course\javascript\programs> node script.js
1
2 3
4 5 6
7 8 9 10
11 12 13 14 15
PS E:\Jayna\Projects\front-end course\javascript\programs>
```

4)\*  
\* \*

\* \* \*

\* \* \* \*

\* \* \* \* \*

The screenshot shows a Microsoft Visual Studio Code interface. The left sidebar has icons for Explorer, Search, and Problems. The main area shows a file named 'script.js' with the following content:

```
1 console.log('*');
2 console.log('* *');
3 console.log('* * *');
4 console.log('* * * *');
5 console.log('* * * * *');
```

The terminal at the bottom shows the output of running the script with 'node script.js':

```
PS E:\Jayna\Projects\front-end course\javascript\programs> node script.js
*
*
*
*
*
```

**Q.39 Accept 3 numbers from user using while loop and check each numbers palindrome?**

## **(Array and object Question)**

**Q.40 Write a JavaScript Program to display the current day and time in the following format. Sample Output: Today is Friday. Current Time is 12 PM: 12 : 22 2 ?**

```
script.js
1  function getCurrentDateTime() {
2      const days = [ 'Sunday', 'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday' ];
3      const currentDate = new Date();
4
5      const day = days[currentDate.getDay()];
6      let hours = currentDate.getHours();
7      const minutes = currentDate.getMinutes();
8      const ampm = hours >= 12 ? 'PM' : 'AM';
9
10     hours = hours % 12;
11     hours = hours ? hours : 12; // "0" should be "12"
12
13     const timeString = `${hours} ${ampm}: ${minutes}`;
14
15     const output = `Today is ${day}. Current Time is ${timeString}`;
16
17     return output;
18 }
19 console.log(getCurrentDateTime());
20
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS E:\Jayna Projects\front-end course\javascript\programs> node script.js  
Today is Monday. Current Time is 5 PM: 7  
PS E:\Jayna Projects\front-end course\javascript\programs>

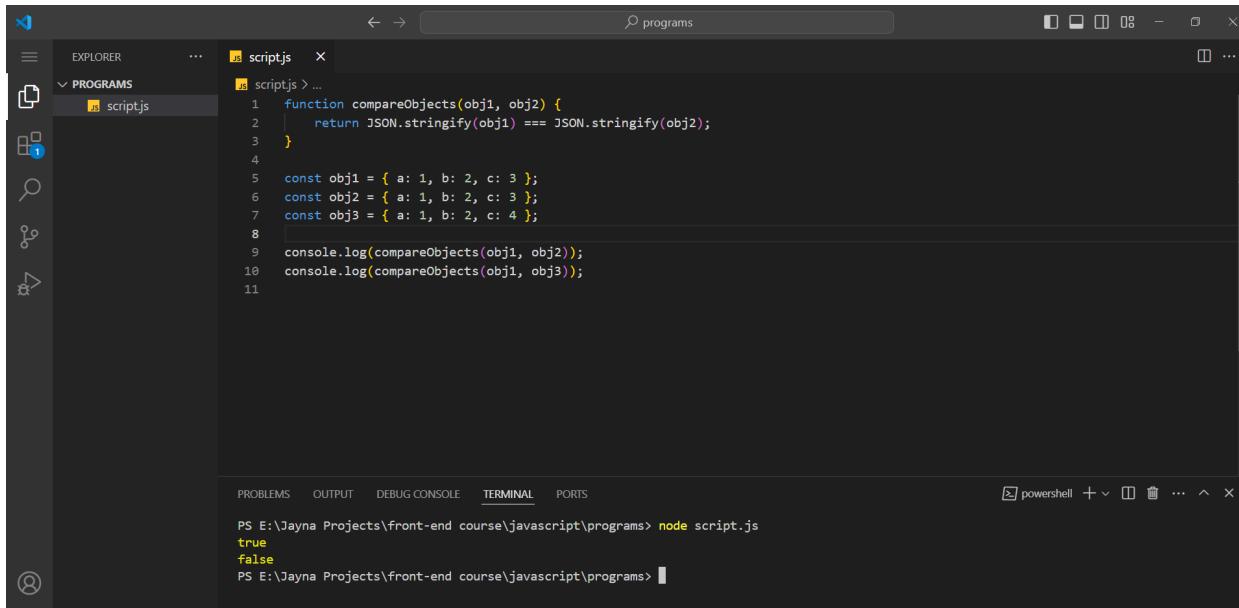
#### Q.41 Write a JavaScript program to get the current date?

```
script.js
1  function getCurrentDate() {
2      const currentDate = new Date();
3
4      const year = currentDate.getFullYear();
5
6      const month = currentDate.getMonth() + 1;
7      const day = currentDate.getDate();
8
9      const formattedMonth = month < 10 ? `0${month}` : month;
10     const formattedDay = day < 10 ? `0${day}` : day;
11
12     return `${year}-${formattedMonth}-${formattedDay}`;
13 }
14
15 console.log(getCurrentDate());
16
17
18
19
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS E:\Jayna Projects\front-end course\javascript\programs> node script.js  
2024-04-22  
PS E:\Jayna Projects\front-end course\javascript\programs>

#### Q.42 Write a JavaScript program to compare two objects?



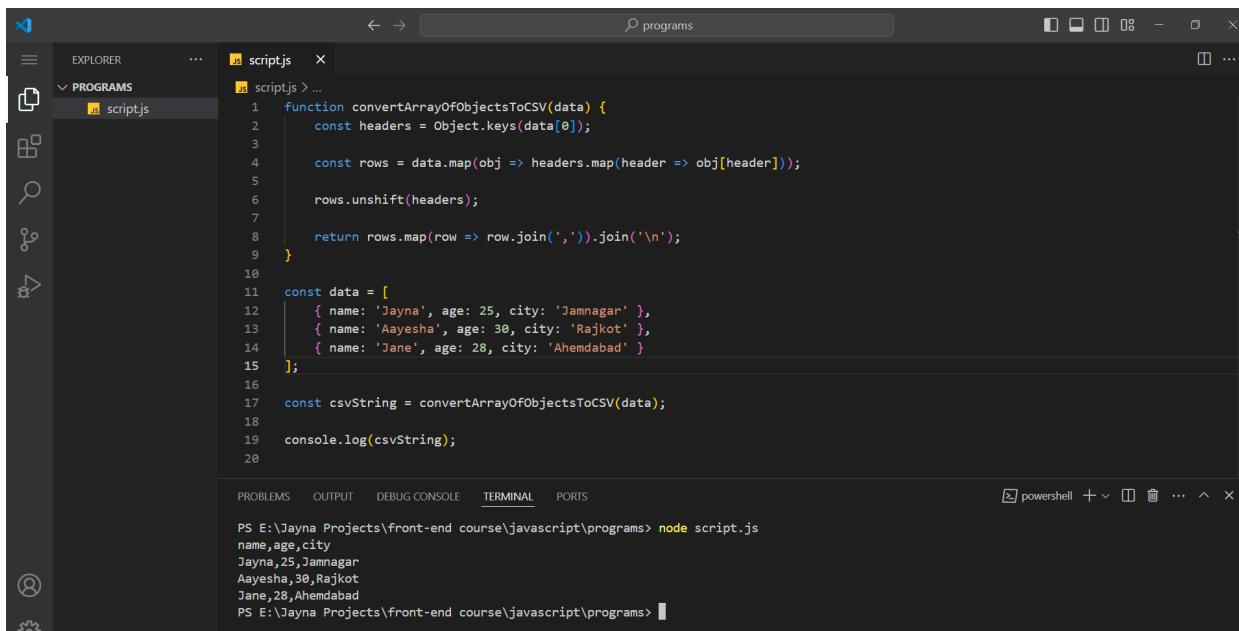
```
function compareObjects(obj1, obj2) {
  return JSON.stringify(obj1) === JSON.stringify(obj2);
}

const obj1 = { a: 1, b: 2, c: 3 };
const obj2 = { a: 1, b: 2, c: 3 };
const obj3 = { a: 1, b: 2, c: 4 };

console.log(compareObjects(obj1, obj2));
console.log(compareObjects(obj1, obj3));
```

PS E:\Jayna Projects\front-end course\javascript\programs> node script.js  
true  
false  
PS E:\Jayna Projects\front-end course\javascript\programs>

#### Q.43 Write a JavaScript program to convert an array of objects into CSV string?



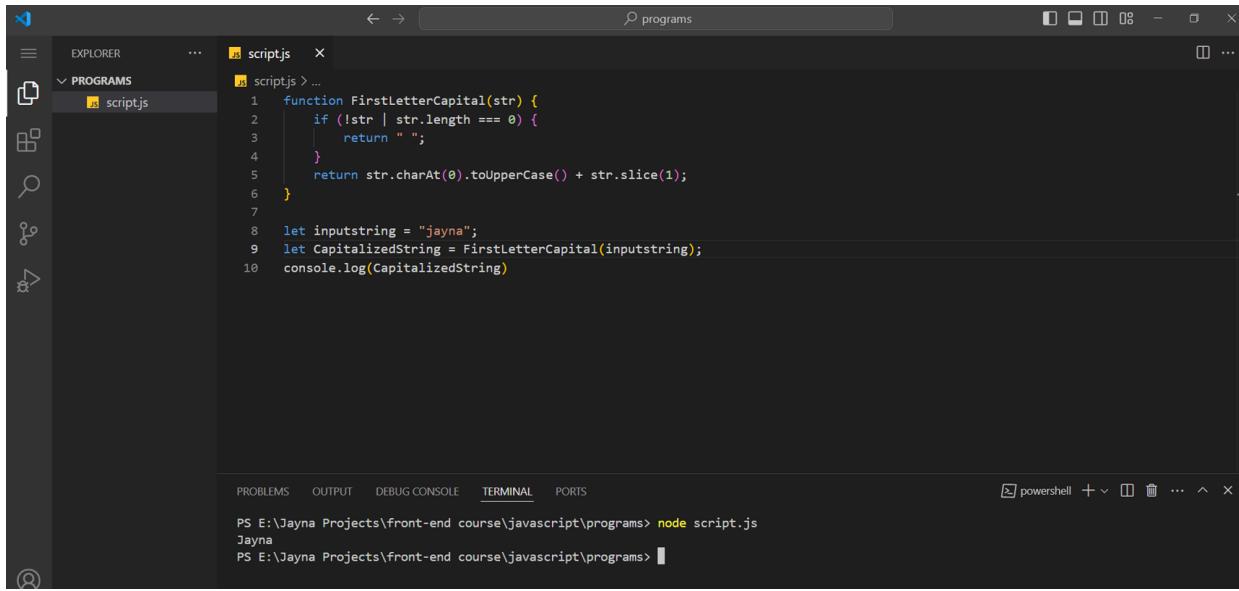
```
function convertArrayOfObjectsToCSV(data) {
  const headers = Object.keys(data[0]);
  const rows = data.map(obj => headers.map(header => obj[header]));
  rows.unshift(headers);

  return rows.map(row => row.join(',')).join('\n');
}

const data = [
  { name: 'Jayna', age: 25, city: 'Jamnagar' },
  { name: 'Aayesha', age: 30, city: 'Rajkot' },
  { name: 'Jane', age: 28, city: 'Ahmedabad' }
];
const csvString = convertArrayOfObjectsToCSV(data);
console.log(csvString);
```

PS E:\Jayna Projects\front-end course\javascript\programs> node script.js  
name,age,city  
Jayna,25,Jamnagar  
Aayesha,30,Rajkot  
Jane,28,Ahmedabad  
PS E:\Jayna Projects\front-end course\javascript\programs>

#### Q.44 Write a JavaScript program to capitalize the first letter of a string?



A screenshot of the Visual Studio Code interface. The left sidebar shows a tree view with 'PROGRAMS' expanded, containing 'script.js'. The main editor area displays the following JavaScript code:

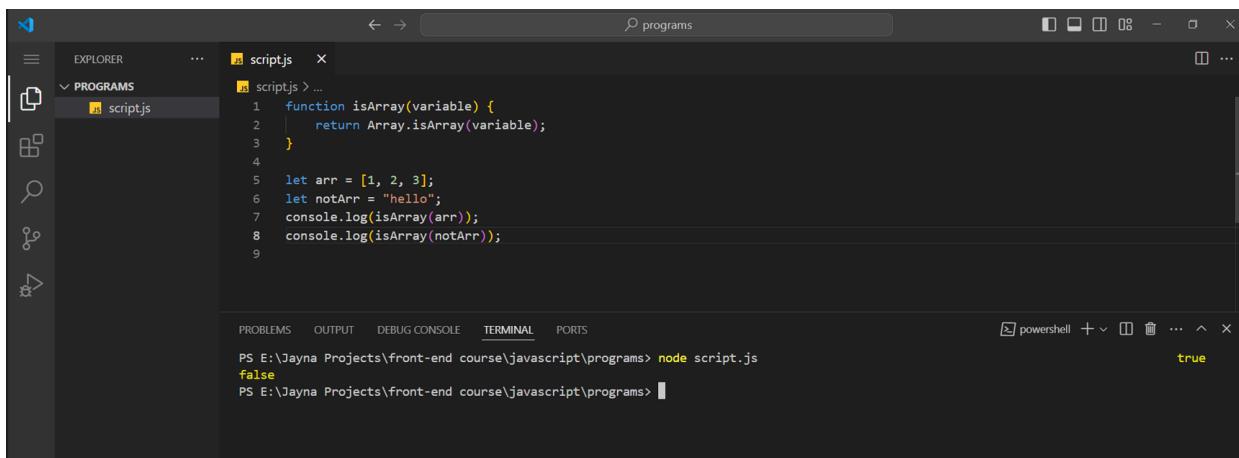
```
function FirstLetterCapital(str) {
  if (!str || str.length === 0) {
    return " ";
  }
  return str.charAt(0).toUpperCase() + str.slice(1);
}

let inputstring = "jayna";
let CapitalizedString = FirstLetterCapital(inputstring);
console.log(CapitalizedString)
```

The terminal at the bottom shows the output of running the script:

```
PS E:\Jayna Projects\front-end course\javascript\programs> node script.js
Jayna
PS E:\Jayna Projects\front-end course\javascript\programs>
```

#### Q.45 Write a JavaScript program to determine if a variable is array?



A screenshot of the Visual Studio Code interface. The left sidebar shows a tree view with 'PROGRAMS' expanded, containing 'script.js'. The main editor area displays the following JavaScript code:

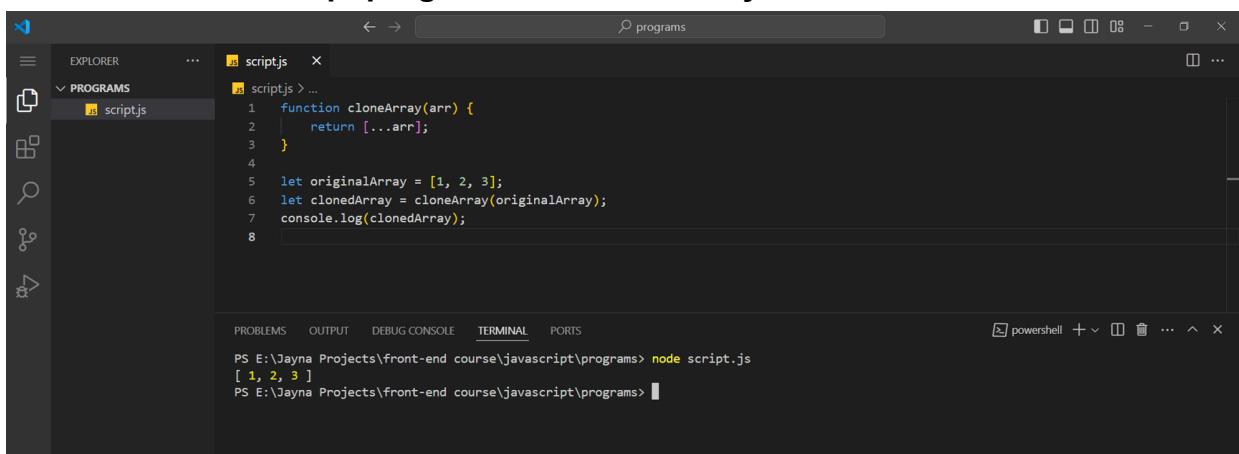
```
function isArray(variable) {
  return Array.isArray(variable);
}

let arr = [1, 2, 3];
let notArr = "hello";
console.log(isArray(arr));
console.log(isArray(notArr));
```

The terminal at the bottom shows the output of running the script:

```
PS E:\Jayna Projects\front-end course\javascript\programs> node script.js
false
true
PS E:\Jayna Projects\front-end course\javascript\programs>
```

#### Q.46 Write a JavaScript program to clone an array?



A screenshot of the Visual Studio Code interface. The left sidebar shows a tree view with 'PROGRAMS' expanded, containing 'script.js'. The main editor area displays the following JavaScript code:

```
function cloneArray(arr) {
  return [...arr];
}

let originalArray = [1, 2, 3];
let clonedArray = cloneArray(originalArray);
console.log(clonedArray);
```

The terminal at the bottom shows the output of running the script:

```
PS E:\Jayna Projects\front-end course\javascript\programs> node script.js
[ 1, 2, 3 ]
PS E:\Jayna Projects\front-end course\javascript\programs>
```

#### **Q.47 What is the drawback of declaring methods directly in JavaScript objects?**

- Declaring methods directly in JavaScript objects can lead to a drawback related to memory usage and performance, especially if those methods are intended to be shared among multiple instances of the object.
- Example:

```
let myObject = {
    method()
}
};
```
- Every time you create a new instance of 'myobject', JavaScript will allocate memory for that method separately for each instance. This can be inefficient if the method is the same for all instances and doesn't need to be recreated every time.

#### **Q.48 Print the length of the string on the browser console using console.log()?**

The screenshot shows the VS Code interface. In the Explorer sidebar, there is a 'PROGRAMS' folder containing a file named 'script.js'. The code in 'script.js' is:1 let str = "Jayna badrakiya";
2 console.log(str.length);In the bottom right terminal window, the command 'node script.js' is run, and the output is '14'.

#### **Q.49 Change all the string characters to capital letters using toUpperCase() method?**

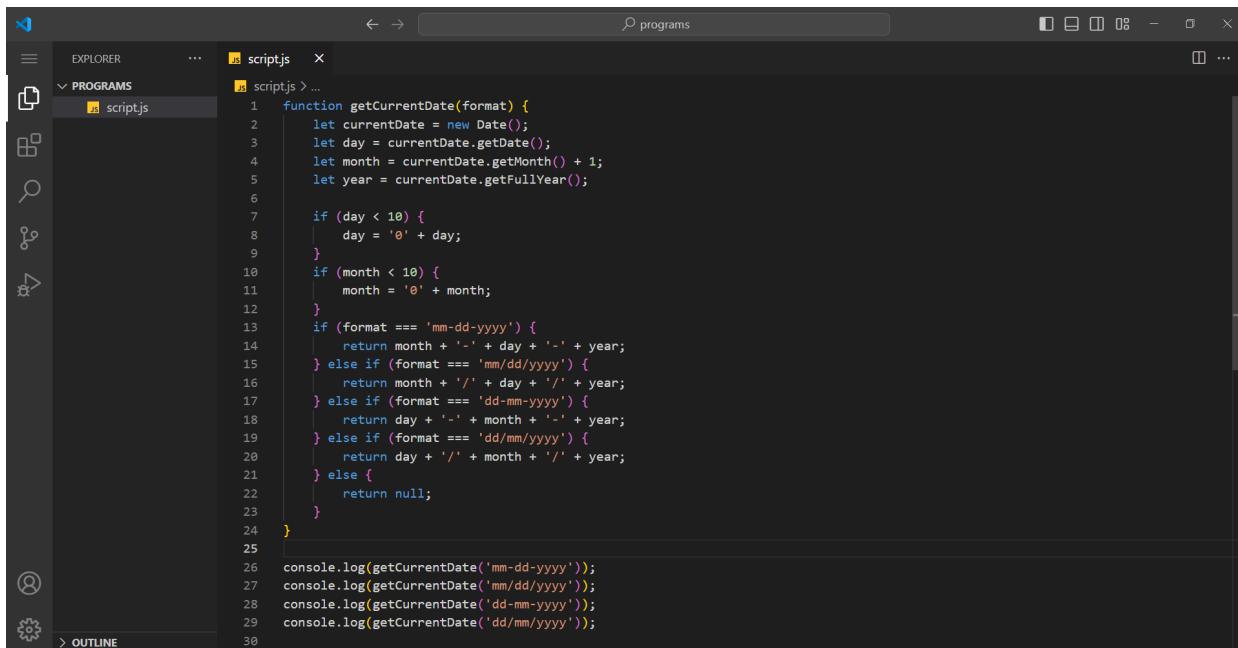
The screenshot shows the VS Code interface. In the Explorer sidebar, there is a 'PROGRAMS' folder containing a file named 'script.js'. The code in 'script.js' is:1 let str = "Jayna badrakiya";
2 let uppercaseStr = str.toUpperCase();
3 console.log(uppercaseStr);In the bottom right terminal window, the command 'node script.js' is run, and the output is 'JAYNA BADRAKIYA'.

## Q.50 What is the drawback of declaring methods directly in JavaScript objects?

- Declaring methods directly in JavaScript objects can lead to a drawback related to memory usage and performance, especially if those methods are intended to be shared among multiple instances of the object.
- Example:

```
let myObject = {
    method()
    {}
};
```
- Every time you create a new instance of 'myobject', JavaScript will allocate memory for that method separately for each instance. This can be inefficient if the method is the same for all instances and doesn't need to be recreated every time.

## Q.51 Write a JavaScript program to get the current date. Expected Output : mm-dd-yyyy, mm/dd/yyyy or dd-mm-yyyy, dd/mm/yyyy?



The screenshot shows a Microsoft Visual Studio Code (VS Code) interface. The left sidebar has icons for Explorer, Search, and Outline. The main area shows an 'EXPLORER' view with a 'PROGRAMS' folder containing a file named 'script.js'. The code editor window displays the following JavaScript code:

```
function getCurrentDate(format) {
    let currentDate = new Date();
    let day = currentDate.getDate();
    let month = currentDate.getMonth() + 1;
    let year = currentDate.getFullYear();

    if (day < 10) {
        day = '0' + day;
    }
    if (month < 10) {
        month = '0' + month;
    }
    if (format === 'mm-dd-yyyy') {
        return month + '-' + day + '-' + year;
    } else if (format === 'mm/dd/yyyy') {
        return month + '/' + day + '/' + year;
    } else if (format === 'dd-mm-yyyy') {
        return day + '-' + month + '-' + year;
    } else if (format === 'dd/mm/yyyy') {
        return day + '/' + month + '/' + year;
    } else {
        return null;
    }
}

console.log(getCurrentDate('mm-dd-yyyy'));
console.log(getCurrentDate('mm/dd/yyyy'));
console.log(getCurrentDate('dd-mm-yyyy'));
console.log(getCurrentDate('dd/mm/yyyy'));
```

The screenshot shows the VS Code interface with the following details:

- Explorer:** Shows a folder named "PROGRAMS" containing a file named "script.js".
- Editor:** Displays the content of "script.js".

```
16     return month + ' / ' + day + ' / ' + year;
17   } else if (format === 'dd-mm-yyyy') {
18     return day + '-' + month + '-' + year;
19   } else if (format === 'dd/mm/yyyy') {
20     return day + '/' + month + '/' + year;
21   } else {
22     return null;
23   }
24 }
25
26 console.log(getCurrentDate('mm-dd-yyyy'));
27 console.log(getCurrentDate('mm/dd/yyyy'));
28 console.log(getCurrentDate('dd-mm-yyyy'));
29 console.log(getCurrentDate("dd/mm/yyyy"));
30
31
```
- Terminal:** Shows the output of running the script with node script.js.

```
PS E:\Jayna Projects\front-end course\javascript\programs> node script.js
04-30-2024
04/30/2024
30-04-2024
30/04/2024
PS E:\Jayna Projects\front-end course\javascript\programs>
```

## Q.52 Use indexOf to determine the position of the first occurrence of a in 30 Days Of JavaScript?

The screenshot shows the VS Code interface with the following details:

- Explorer:** Shows a folder named "PROGRAMS" containing a file named "script.js".
- Editor:** Displays the content of "script.js".

```
1 let str = "30 Days Of JavaScript";
2 let position = str.indexOf('a');
3
4 console.log("Position of 'a':", position);
5
6
7
8
```
- Terminal:** Shows the output of running the script with node script.js.

```
PS E:\Jayna Projects\front-end course\javascript\programs> node script.js
Position of 'a': 4
PS E:\Jayna Projects\front-end course\javascript\programs>
```

## Q.53 Use lastIndexOf to determine the position of the last occurrence of a in 30 Days Of JavaScript?

The screenshot shows the VS Code interface with the following details:

- Explorer:** Shows a folder named "PROGRAMS" containing a file named "script.js".
- Editor:** Displays the content of "script.js".

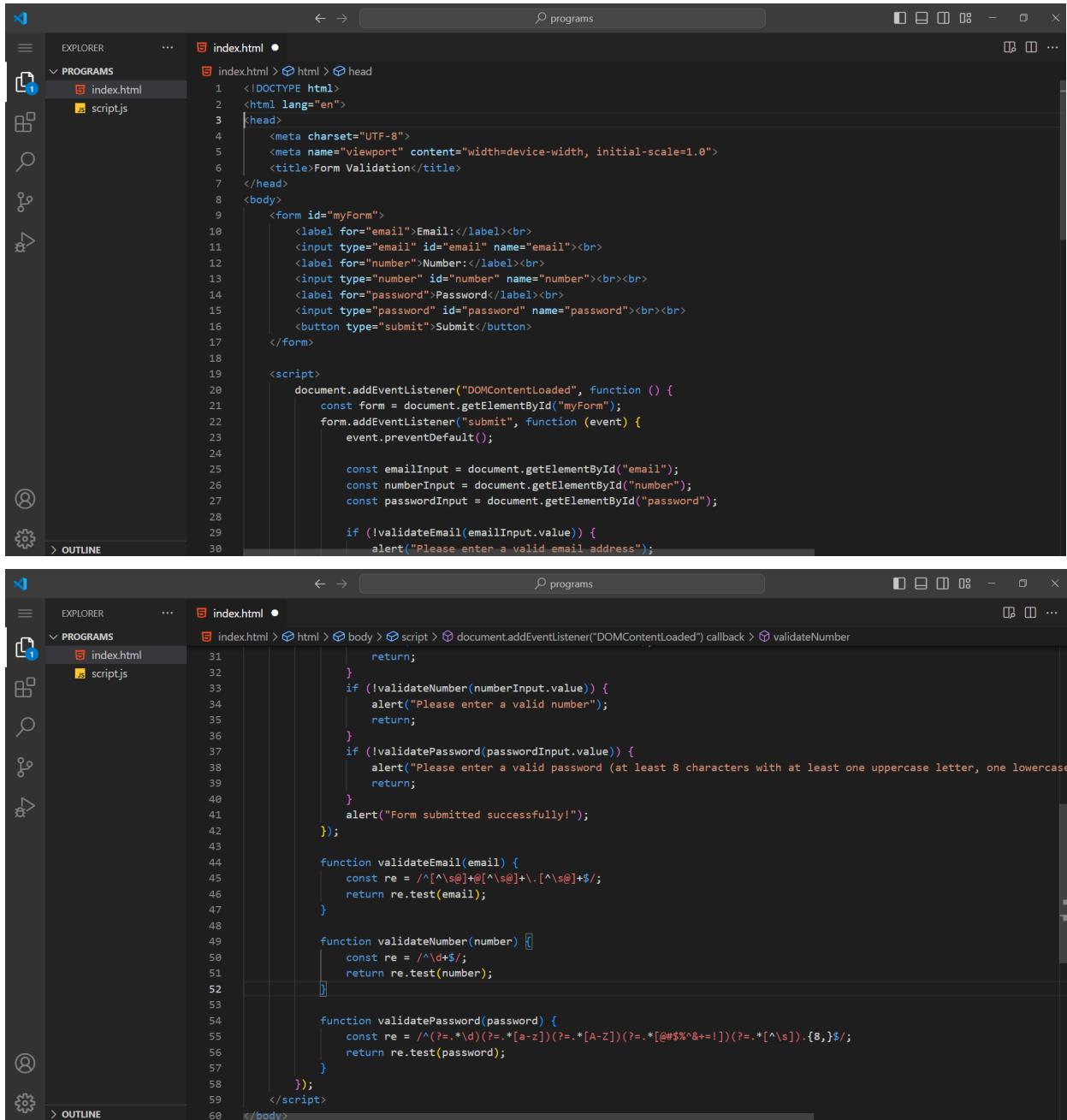
```
1 let str = "30 Days Of JavaScript";
2 let position = str.lastIndexOf('a');
3
4 console.log("Position of 'a':", position);
5
6
7
8
```
- Terminal:** Shows the output of running the script with node script.js.

```
PS E:\Jayna Projects\front-end course\javascript\programs> node script.js
Position of 'a': 14
PS E:\Jayna Projects\front-end course\javascript\programs>
```

## Q.54 Form Validation in JS?

- Form validation in JavaScript involves checking the data entered into form fields to ensure it meets certain criteria or constraints before it is submitted to the server. This helps in ensuring that the data submitted is accurate, complete, and secure.

## Q.55 Form in Email, number, Password, Validation?



The image shows two tabs open in a code editor:

- index.html**: Contains the HTML structure for a form with three input fields: email, number, and password, each with a corresponding label. It also includes a submit button.
- script.js**: Contains the JavaScript logic for validating the form fields. It uses event listeners to check if the email is valid, if the number is valid, and if the password is valid. If any validation fails, an alert is displayed.

```
index.html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Form Validation</title>
</head>
<body>
    <form id="myForm">
        <label for="email">Email:</label><br>
        <input type="email" id="email" name="email"><br>
        <label for="number">Number:</label><br>
        <input type="number" id="number" name="number"><br><br>
        <label for="password">Password:</label><br>
        <input type="password" id="password" name="password"><br><br>
        <button type="submit">Submit</button>
    </form>
</body>
```

```
script.js
document.addEventListener("DOMContentLoaded", function () {
    const form = document.getElementById("myForm");
    form.addEventListener("submit", function (event) {
        event.preventDefault();

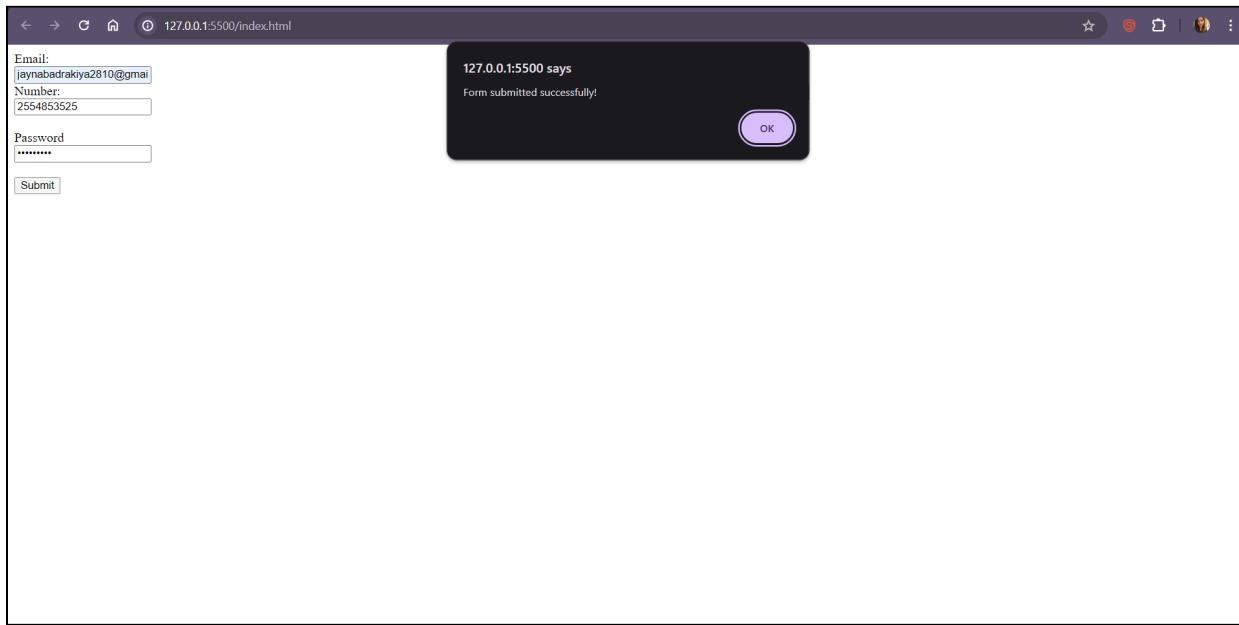
        const emailInput = document.getElementById("email");
        const numberInput = document.getElementById("number");
        const passwordInput = document.getElementById("password");

        if (!validateEmail(emailInput.value)) {
            alert("Please enter a valid email address");
        } else if (!validateNumber(numberInput.value)) {
            alert("Please enter a valid number");
        } else if (!validatePassword(passwordInput.value)) {
            alert("Please enter a valid password (at least 8 characters with at least one uppercase letter, one lowercase letter, and one special character)");
        } else {
            alert("Form submitted successfully!");
        }
    });
}

function validateEmail(email) {
    const re = /^[^\s@]+@[^\s@]+\.[^\s@]+\$/;
    return re.test(email);
}

function validateNumber(number) {
    const re = /\d+/;
    return re.test(number);
}

function validatePassword(password) {
    const re = /^(?=.*\d)(?=.*[a-z])(?=.*[A-Z])(?=.*[@#$%^&+=!])(?=.*[^\s]).{8,}$/;
    return re.test(password);
}
```



#### **Q.56 Dynamic Form Validation in JS?**

- Dynamic form validation in JavaScript involves real-time validation of form inputs as the user interacts with the form. Unlike traditional validation where errors are displayed only upon form submission, dynamic validation provides immediate feedback to the user, often as they type or move focus away from an input field.

#### **Q.57 How many types of JS Event? How to use it ?**

- JavaScript events are actions or occurrences that happen in the browser, triggered by user interaction, browser actions, or by other JavaScript code. Events allow JavaScript to respond to user input or system actions, enabling interactive and dynamic web pages. There are many types of events in JavaScript. Here are some common types:

- 1) Mouse Events: These events are triggered by mouse actions.  
click: Triggered when the element is clicked.  
dblclick: Triggered when the element is double-clicked.  
mouseover: Triggered when the mouse pointer enters the element.  
mouseout: Triggered when the mouse pointer leaves the element.  
mousedown: Triggered when the mouse button is pressed down on the element.  
mouseup: Triggered when the mouse button is released over the element.  
mousemove: Triggered when the mouse pointer moves over the element.  
mouseenter: Similar to mouseover, but not triggered by event bubbling.
- 2) Keyboard Events: These events are triggered by keyboard actions.  
keydown: Triggered when a key is pressed down.  
keyup: Triggered when a key is released.  
keypress: Triggered when a key that produces a character value is pressed down.
- 3) Form Events: These events are triggered by form elements.  
submit: Triggered when a form is submitted.

reset: Triggered when a form is reset.  
change: Triggered when the value of a form element changes (e.g., input field, select box).

input: Triggered when the value of an input field changes.  
focus: Triggered when an element receives focus.  
blur: Triggered when an element loses focus.

- 4) Window Events: These events are triggered by actions related to the window or document.  
load: Triggered when the page/document is fully loaded.  
unload: Triggered when the page/document is being unloaded (e.g., navigating to a new page).  
resize: Triggered when the browser window is resized.  
scroll: Triggered when scrolling occurs on the window or an element.  
DOMContentLoaded: Triggered when the HTML document has been completely loaded and parsed.
- 5) Event Delegation: This is not a specific event type but a concept in JavaScript where a single event listener is attached to a parent element, and events from child elements are handled based on a selector. This is useful for handling events on dynamically added elements or a large number of elements efficiently.

**Example:**

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Event Example</title>
</head>
<body>

<button id="myButton">Click me</button>

<script>
const button = document.getElementById("myButton");
button.addEventListener("click", function() {
    alert("Button clicked!");
});
</script>

</body>
</html>
```

## Q.59 What is BOM vs DOM in JS?

- In JavaScript, "BOM" stands for Browser Object Model, while "DOM" stands for Document Object Model.

### **Browser object model**

- The browser object model is a set of objects provided by the browser, which allows javascript to communicate with the browser. It includes objects such as window, navigator, screen, history and location. These objects provide methods and properties to interact with browser functionalities like controlling the browser window, navigating to different URLs, managing cookies, and so on. Bom doesn't have any standard defined by W3C, and its specific to each browser implementation.

### **Document object model**

- The document object model is a representation of the structure of HTML or XML documents. It provides a way for scripts to dynamically access and manipulate the content, structure and style of documents. The DOM organizes the elements of a document as a hierarchical tree, where each node represents an element like <div>, <p>, <img> or other entities like text or comments.
- It allows javascript to interact with the content of a web page, enabling tasks such as adding or removing elements, changing styles, handling events and so on. DOM is standardized by the W3C and is implemented uniformly across browsers.

## **Q.60 Array vs object defences in JS?**

### **Array defences**

- Boundary Checking: Always check array bounds to avoid accessing elements outside the array's range.
- Type Checking: Ensure that arrays contain elements of expected types to prevent unexpected behavior.
- Avoid Sparse Arrays: Sparse arrays (arrays with holes or gaps) can lead to unexpected behavior, so it's generally better to avoid them.
- Avoid Direct Modification of Array Prototype: Modifying the prototype of the Array object can lead to unexpected behavior. Avoid adding or removing properties from Array.prototype.

### **Object defences**

- Property Existence Checking: Always check if a property exists before accessing it to avoid errors.
- Use hasOwnProperty: When iterating over object properties, use hasOwnProperty to make sure you're only iterating over the object's own properties, not inherited ones.
- Freezing Objects: Use Object.freeze() to make an object immutable, preventing properties from being added, modified, or removed.
- Sealing Objects: Use Object.seal() to prevent new properties from being added and existing ones from being deleted, but still allowing their values to be changed.
- Avoid Circular References: Be cautious of circular references in objects, as they can cause infinite loops when iterating over properties.

### Q.61 Split the string into an array using split() Method?

The screenshot shows the VS Code interface. In the Explorer sidebar, there is a 'PROGRAMS' folder containing a file named 'script.js'. The code in 'script.js' is:

```
const str = "apple,banana,orange,grape";
const separator = ",";
const arrayOfFruits = str.split(separator);
console.log(arrayOfFruits);
```

In the bottom right corner, the terminal window displays the output of running the script:

```
PS E:\Jayna Projects\front-end course\javascript\programs> node script.js
[ 'apple', 'banana', 'orange', 'grape' ]
PS E:\Jayna Projects\front-end course\javascript\programs>
```

### Q.62 Check if the string contains a word Script using includes() method?

The screenshot shows the VS Code interface. In the Explorer sidebar, there is a 'PROGRAMS' folder containing a file named 'script.js'. The code in 'script.js' is:

```
const str = "JavaScript is a scripting language.";
const wordToCheck = "Script";
const containsScript = str.includes(wordToCheck);
console.log(containsScript);
```

In the bottom right corner, the terminal window displays the output of running the script:

```
PS E:\Jayna Projects\front-end course\javascript\programs> node script.js
true
PS E:\Jayna Projects\front-end course\javascript\programs>
```

### Q.63 Change all the string characters to lowercase letters using toLowerCase() Method.

A screenshot of the Visual Studio Code interface. The left sidebar shows a tree view with 'PROGRAMS' expanded, containing 'script.js'. The main editor area has a dark theme and displays the following code:

```
script.js > ...
1 let str = "Hello, World!";
2
3 str = str.toLowerCase();
4
5 console.log(str);
6
7
8
9
```

The terminal at the bottom shows the output of running the script:

```
PS E:\Jayna Projects\front-end course\javascript\programs> node script.js
hello, world!
PS E:\Jayna Projects\front-end course\javascript\programs>
```

**Q.64 What is Character at index 15 in '30 Days of JavaScript' string? Use charAt() method.**

A screenshot of the Visual Studio Code interface. The left sidebar shows a tree view with 'PROGRAMS' expanded, containing 'script.js'. The main editor area displays the following code:

```
script.js > ① charAtIndex15
1 const str = '30 Days of JavaScript';
2 const charAtIndex15 = str.charAt(15);
3
4 console.log(charAtIndex15);
5
6
7
8
9
10
```

The terminal at the bottom shows the output of running the script:

```
PS E:\Jayna Projects\front-end course\javascript\programs> node script.js
S
PS E:\Jayna Projects\front-end course\javascript\programs>
```

**Q.65 copy to one string to another string in JS?**

A screenshot of the Visual Studio Code interface. The left sidebar shows a tree view with 'PROGRAMS' expanded, containing 'script.js'. The main editor area displays the following code:

```
script.js > ...
1 let originalString = "Hello my name is jayna";
2 let copiedString = originalString;
3
4 console.log(copiedString);
5
6
7
8
9
```

The terminal at the bottom shows the output of running the script:

```
PS E:\Jayna Projects\front-end course\javascript\programs> node script.js
Hello my name is jayna
PS E:\Jayna Projects\front-end course\javascript\programs>
```

**Q.66 Find the length of a string without using libraryFunction?**

```
const str = 'Hello, World!';
let length = 0;
for (let i = 0; i < str.length; i++) {
    length++;
}
console.log(length);
```

```
PS E:\Jayna Projects\front-end course\javascript\programs> node script.js
13
PS E:\Jayna Projects\front-end course\javascript\programs>
```

- **What is JavaScript?**

- JavaScript is a lightweight, cross-platform, single-threaded, and interpreted compiled programming language. It is also known as the scripting language for web pages.
- It is well-known for the development of web pages, and many non-browser environments also use it.
- JavaScript can update and change both HTML and CSS.
- JavaScript can calculate, manipulate and validate data.

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

- In JavaScript, NaN is short for "Not-a-Number".
- In JavaScript, NaN is a number that is not a legal number.
- The Number.isNaN() method returns true if the value is NaN, and the type is a Number.

- **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 less than this value. It can be generated using a self-made function or by an arithmetic operation.

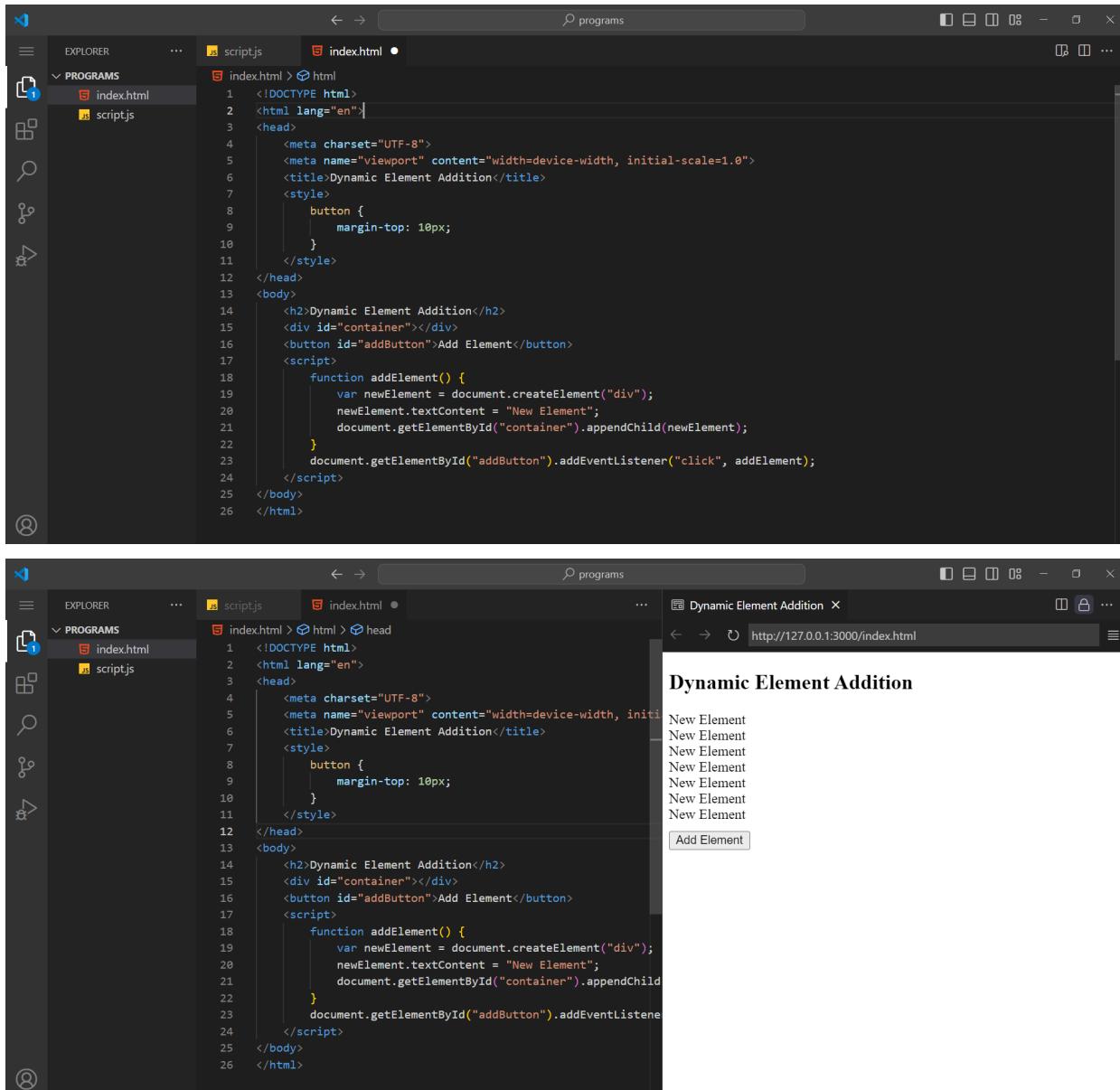
- **Which company developed JavaScript?**

- JavaScript was created and developed 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.

- **What are undeclared and undefined variables?**

- Declared variables are created before any code is executed. Declared variables are a non-configurable property of their execution context (function or global).
- Undeclared variables do not exist until the code assigned to them is executed. Undeclared variables are configurable (e.g. can be deleted).

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



```

index.html > html
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Dynamic Element Addition</title>
7      <style>
8          button {
9              margin-top: 10px;
10         }
11     </style>
12   </head>
13   <body>
14       <h2>Dynamic Element Addition</h2>
15       <div id="container"></div>
16       <button id="addButton">Add Element</button>
17       <script>
18           function addElement() {
19               var newElement = document.createElement("div");
20               newElement.textContent = "New Element";
21               document.getElementById("container").appendChild(newElement);
22           }
23           document.getElementById("addButton").addEventListener("click", addElement);
24       </script>
25   </body>
26 </html>

```

## • What is the difference between ViewState and SessionState?

### ViewState

- ViewState is used to preserve the state of page and control values across HTTP requests.
- It stores data of a single page in a hidden field on the page itself.
- ViewState is useful for maintaining state between postbacks of a single page.
- It works at the control level, meaning each control can maintain its own ViewState.
- ViewState is preserved even if the page is refreshed.

- ViewState is lightweight compared to SessionState.

### **Session state**

- SessionState is used to store user-specific information across multiple HTTP requests.
- It stores data across all pages in a user session, which is typically maintained either in-memory (InProc), in a separate process (StateServer), or in a database (SQLServer).
- SessionState is useful for maintaining user-specific information like login details, user preferences, etc.
- It works at the session level, meaning the data stored in SessionState is available across multiple pages during a user's session.
- SessionState is terminated when the session expires (usually due to inactivity) or when the user closes the browser.
- SessionState requires server resources to store session data, so it can impact scalability if not managed properly.

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

- The === operator in JavaScript is called the strict equality operator. It checks if two values are equal in both value and type without performing type coercion. If the values being compared are of different types, the result is false.

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

- The style of an element can be changed using the style property or by adding/removing classes.
- Example:
 

```
// Changing style directly
document.getElementById('myElement').style.color = 'red';
// Adding a class
document.getElementById('myElement').classList.add('highlight');
// Removing a class
document.getElementById('myElement').classList.remove('highlight');
```

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

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

- for loop
- while loop
- do...while loop
- for...in loop
- for...of loop (introduced in ES6)

- **How can you convert the string of any base to an integer in JavaScript?**
  - You can use the parseInt() function to convert a string representation of a number in any base to an integer.
  - Example:  
`parseInt('1010', 2);`
- **What is the function of the delete operator?**
  - The delete operator is used to remove a property from an object.
  - Example:  
`const obj = {a: 1, b: 2};  
delete obj.a;`
- **What are all the types of Pop up boxes available in JavaScript?**
  - alert(): Displays an alert box with a message and an OK button.
  - confirm(): Displays a dialog box with a message and OK and Cancel buttons.
  - prompt(): Displays a dialog box that prompts the user for input.
- **What is the use of Void (0)?**
  - void(0) is used to return undefined. It is often used in the href attribute of an anchor tag to prevent the page from navigating anywhere when the link is clicked.
  - Example:  
`<a href="javascript:void(0)">Click me</a>`
- **How can a page be forced to load another page in JavaScript?**
  - You can use window.location to redirect the browser to another page.
  - Example:  
`window.location.href = 'https://www.example.com';`
- **What are the disadvantages of using innerHTML in JavaScript?**
  - Security risk : Inserting unsanitized user input via innerHTML can lead to cross-site scripting attacks.
  - Performance : Replacing the entire HTML content of an element using innerHTML is less efficient than using DOM manipulation methods like createElement and appendChild.
  - Event listeners : If elements with event listeners are replaced using innerHTML, those listeners might get detached and need to be reattached.
- **Create password field with show hide functionalities**

The screenshot shows a code editor interface with two tabs: 'index.html' and 'script.js'. The 'index.html' tab is active, displaying the following HTML and JavaScript code:

```
<input type="password" id="password">
<button onclick="togglePassword()">Show/Hide</button>
<script>
  function togglePassword() {
    var passwordField = document.getElementById("password");
    if (passwordField.type === "password") {
      passwordField.type = "text";
    } else {
      passwordField.type = "password";
    }
  }
</script>
```

