Module (JAVASCRIPT BASIC & DOM) – 4

(Basic logic Questions and answers)

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

- JavaScript (js) is a light-weight object-oriented programming language which is used by several websites for scripting the webpages.
- JavaScript is *an object-based scripting language* which is lightweight and cross-platform.
- JavaScript is not a compiled language, but it is a translated language.
- The JavaScript Translator (embedded in the browser) is responsible for translating the JavaScript code for the web browser.

➢ **How to use :**

- JavaScript provides 3 places to put the JavaScript code:

1. Between the body tag of html
2. Between the head tag of html
3. In .js file (external JavaScript)

1) **JavaScript Example : code between the body tag**

- In the above example, we have displayed the dynamic content using JavaScript. Let's see the simple example of JavaScript that displays alert dialog box.

```
<script type="text/javascript">
        alert("Hello ");
</script>
```

2) **JavaScript Example : code between the head tag**

- Let's see the same example of displaying alert dialog box of JavaScript that is contained inside the head tag.

```
<html>
  <head>
  <script type="text/javascript">
        function msg(){
              alert("Hello");
        }
```

```
</script>
</head>
<body>
        <p>Welcome to JavaScript</p>
    <form>
        <input type="button" value="click" onclick="msg()"/>
    </form>
</body>
</html>
```

3) External JavaScript file

- We can create external JavaScript file and embed it in many html page.
- It provides **code re usability** because single JavaScript file can be used in several html pages.
- An external JavaScript file must be saved by .js extension. It is recommended to embed all JavaScript files into a single file. It increases the speed of the webpage.
- Let's create an external JavaScript file that prints Hello Javatpoint in a alert dialog box.

message.js

```
function msg(){
        alert("Hello ");
}
```

- Let's include the JavaScript file into html page. It calls the JavaScript function on button click.

index.html

```
<html>
    <head>
        <script type="text/javascript" src="message.js"></script>
    </head>
    <body>
        <p>Welcome to JavaScript</p>
      <form>
        <input type="button" value="click" onclick="msg()"/>
      </form>
    </body>
```

```
</html>
```

Q-2. How many types of variable in JavaScript ?

- There are three types of variables.
  1. let
  2. var
  3. const

Q-3. Define a Data Type in JS ?

1. **Primitive Data Type :**

- null
- number
- boolean
- bigint
- string
- symbol()
- undefined

2. **Non-Primitive Data Type :**

- object : Object is return keys and values.
- array : Array is return index and length.

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

```
function mul(x,y){
    return x*y;
};
console.log(mul(6,4));
//ans = 24;
```

Q-5. What the deference between undefined and undeclare in JavaScript?

- **undefined** : refers to a variable that has been declared but has not been assigned a value or a function that does not explicitly return a value.
- **undeclare** : refers to a variable that has been referenced but has not been formally declared using var, let or const

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:

console.log("The quote 'There is no exercise better for the heart than reaching down and lifting people up.**");**

console.log(“Spread love everywhere you go. Let no one ever com to you without leaving happier. -Mother Teresa**”);**

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

```
let a = 10;
let b = “10”; //10
console.log(typeof a);
console.log(typeof b);
if(typeof a === typeof b){
console.log(“True”);
} else {
        console.log(“False”);
}
//ans : False because datatype is not same, if b = 10 then answer is true.
```

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

```
let base = prompt(“Enter the triangle base size : ”);
let height = prompt(“Enter the triangle height : ”);
let area = (base*height)/2;
console.log(`base size : ${base}`);
console.log(`height size : ${height}`);
console.log(“Area of triangle is : ”, area);
```

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

```
let today = new Date();
let Year_2023 = today.getFullYear();
 if(today.getMonth() === 11 && today.getDate()>25){
charistmas.setFullYear(Year_2023 + 1);
}
let Time = charistmas.getTime() – today.getTime();
```

```
let DayCount = Month.ceil(Time/(1000*60*60*24));
consol.log("Days left until next Christmas:"+DayCount);
//ans: Days left until next Christmas : 350
```

Q-10. What is Condition Statement?

- 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**.**
- There are some types of conditional statements **:**
  1. if...else...
  2. nested if...else...
  3. switch...case...
  4. nested switch...case...

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

```
// Define the length and width of the rectangle
let length = 5; // Example length
let width = 3; // Example width
// Calculate the circumference of the rectangle
let circumference = 2 * (length + width);
// Print the circumference
console.log("The circumference of the rectangle is:", circumference);
```

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

```
// Function to convert years to days
function yearsToDays(years) {
   return years * 365; // Assuming 1 year has 365 days
}

// Function to convert days to years
function daysToYears(days) {
   return days / 365; // Assuming 1 year has 365 days
}

// Test the functions
let years = 2;
```

```
let days = 730; // 2 years * 365 days/year = 730 days
console.log(years + " years is equal to " + yearsToDays(years) + " days.");
console.log(days + " days is equal to " + daysToYears(days) + " years.");
```

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

```
function fahrenheitToCelsius(fahrenheit) {
    // Convert Fahrenheit to Celsius using the formula
    let celsius = (fahrenheit - 32) / 1.8;
    return celsius;
}

// Test the function with a sample temperature in Fahrenheit
let fahrenheit = 98.6; // Example temperature in Fahrenheit (normal human body
    temperature)
let celsius = fahrenheitToCelsius(fahrenheit);

console.log(fahrenheit + " degrees Fahrenheit is equal to " + celsius.toFixed(2) + "
    degrees Celsius.");
```

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

```
function getFileExtension(filename) {
    // Split the filename by dot (.)
    let parts = filename.split(".");
    // If there's no dot or only one part in the filename, return an empty string
    if (parts.length === 1 || parts[parts.length - 1] === "") {
        return "";
    }
    // Otherwise, return the last part of the filename (which is the extension)
    return parts[parts.length - 1];
}// Test the function
let filename1 = "example.txt";
```

```
let filename2 = "script.js";
console.log("Extension of " + filename1 + ": " + getFileExtension(filename1));
console.log("Extension of " + filename2 + ": " + getFileExtension(filename2));
```

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

consol.log(5 > 3 && 2 < 4);

Ans : true

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

consol.log(true && 1 && "hello");

Ans : hello

Q-17. What is the result of the expression true && false || false && true?

consol.log(true && false || false && true);

Ans : false

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

- **Loop :**

  loop is used in JavaScript to perform repeated tasks based on a condition

- **Switch Case :**

  The switch statement is used to perform different actions based on different conditions.

Q-19. What is the use of isNan function?

- Nan means Not-a-number.
- When the value is NaN than is NaN is return the true and when the value is not NaN than is NaN is return the false.
- **Exp:**

```
let str = "Hyy";
let int = 123;
consol.log(isNaN(str)); //ans : true
consol.log(isNaN(int)); //ans : false
```

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

- **&&(And) :** && is the logical AND operator, which returns true only if both of its operands are true. On the other hand.
- **||(Or) :** || is the logical OR operator, which returns true if at least one of its operands is true.

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

- Void(0) stops pages from refreshing, and "zero" is used to pass the parameter while calling.
- Void(0) is used to call another method without refreshing the page.

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

```
let num = prompt(“Enter the number : ”);
if(num >= 0){
    console.log(“Number is positive : ”, num);
}else if(num < 0){
    console.log(“Number is nagative : ”, num);
} else {
    console.log(“Your entered number is invalid.”);
}
```

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

```
let Vovels = prompt("Enter any alphabates : ");
Vovels = Vovels.toUpperCase();
        switch (Vovels){
        case 'A':
                console.log("You enatered alphabate is vovel");
        break;
        case 'E':
                console.log("You enatered alphabate is vovel");
        break;
        case 'I':
                console.log("You enatered alphabate is vovel");
        break;
        case 'O':
                console.log("You enatered alphabate is vovel");
        break;
        case 'U':
                console.log("You enatered alphabate is vovel");
```

```
        break;
        default:
            console.log("You enatered alphabate is not vovel");}
```

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

```
let Whether = prompt("Enter the number : ");
if(Whether = 0){
        console.log("Whether is zero : ",Whether);
}else if(Whether > 0) {
        console.log("Whether is positive : ",Whether);
}else if(Whether < 0){
        console.log("Whether is nagative : ",Whether);
}else {
        console.log("You entered number is invalid");
}
```

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

```
let EvenOdd = prompt("Enter the number : ");
let Operat = EvenOdd % 2 === 0 ? "Number is Even" : "Number is Odd";
        console.log(Operat);
```

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

```
let num1 = prompt("Enter the num 1 : ");
let num2 = prompt("Enter the num 2 : ");
let num3 = prompt("Enter the num 3 : ");
let max = num1>num2 ? (num1>num3 ? num1 : num3): (num2>num3 ? num2 : num3);
console.log(`${max} is max.`);
```

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

```
let num1 = prompt("Enter the num 1 : ");
let num2 = prompt("Enter the num 2 : ");
let num3 = prompt("Enter the num 3 : ");
```

```
let min = num1<num2 ? (num1<num3 ? num1 : num3): (num2<num3 ? num2 : num3);

console.log(`${min} is min.`);
```

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

```
let num1 = prompt("Enter the num 1 : ");

let num2 = prompt("Enter the num 2 : ");

let num3 = prompt("Enter the num 3 : ");

if(num1>num2){
        if (num1>num3) {
                console.log(`${num1} is large.`);
        }else{
                console.log(`${num3} is large.`);
        }
}else {
        if (num2>num3) {
                console.log(`${num2} is large.`);
        }else{
                console.log(`${num3} is large.`);
        }
}
```

Q-29. Write to show:

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

```
let Days = prompt("Enter number (1 to 7) : ");
   Days = Number.parseInt(Days);
      switch (Days){
         case 1:
            console.log("Sunday");
            break;
         case 2:
            console.log("Monday");
            break;
         case 3:
            console.log("Monday");
         break;
         case 4:
```

```
        console.log("Wednesday");
        break;
    case 5:
        console.log("Thersday");
        break;
    case 6:
        console.log("Friday");
        break;
    case 7:
        console.log("Saturday");
        break;
    default:
        console.log("Invalid number.");
        break;
    }
```

ii. Vowel or Consonant using switch case in JS?

```
let Vovels = prompt("Enter any alphabates : ");
    Vovels = Vovels.toUpperCase();
    switch (Vovels){
        case 'A':
            console.log("You enatered alphabate is vovel");
            break;
        case 'E':
            console.log("You enatered alphabate is vovel");
            break;
        case 'I':
            console.log("You enatered alphabate is vovel");
            break;
        case 'O':
            console.log("You enatered alphabate is vovel");
            break;
        case 'U':
            console.log("You enatered alphabate is vovel");
            break;
        default:
```

```
        console.log("You enatered alphabate is consonant");
    }
```

(Conditional looping logic Question)

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

- Loops are handy, if you want to run the same code over and over again, each time with a different value.
- Exp :

```
for(let num = 0; num <= 10; num++){
        console.log(num);
}// Ans : print 1 to 10.
```

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

```
for(let num = 992; num >= 897; num--){
        console.log(num);
}// Ans : print 992 to 897(Dicriment).
```

Q-32. Write to print factorial of given number?

```
let num = prompt("Enter a number for find factorial : ");
num = Number.parseInt(num);
let fact = 1;
for(let i = 1; i <= num; i++){
        fact = fact * i;
}
        console.log(fact);
// Ans: if num = 6, elase factorial of num = 720.
```

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

```
let Fibbo = parseInt(prompt("Enter the number : "));
let n1 = 0, n2 = 1, n3;
        console.log("Fibonacci series.....");
for(let i = 1; i <= Fibbo; i++){
        console.log(n1);
        n3 = n1 + n2;
```

```
        n1 = n2;
        n2 = n3;
}
```

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

```
function reverseNumber(number) {
  // Convert the number to a string, split it into an array of digits, reverse the array, and join the digits back into a string
  return parseInt(number.toString().split('').reverse().join(''));
}
// Test the function
console.log(reverseNumber(64728)); // Output: 82746
```

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

```
let num = parseInt(prompt("Enter the number for sum : "));
let sum = 0;
        while(num > 0){
                let r = num % 10;
                sum = sum + r;
                num = parseInt(num / 10);
        }
console.log(sum);
```

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

```
function sumFirstAndLastDigit(number) {
  let numString = number.toString();
  let firstDigit = parseInt(numString[0]);
  let lastDigit = parseInt(numString[numString.length - 1]);
  let sum = firstDigit + lastDigit;
  return sum;
}
```

```
let number = 1234;
let result = sumFirstAndLastDigit(number);
console.log("Sum of first and last digit of", number, "is:", result);
```

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

```
const numRows = 5;
for (let i = 1; i <= numRows; i++) {
   let row = "";
   for (let j = 1; j <= 5; j++) {
      if (j === 1) {
         row += i + " "; // Print the row number for the first column
      } else {
         // Calculate the value based on the pattern
         let value = Math.pow(i, j);
         row += value + " ";
      }
   }
   console.log(row);
}
```

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

```
const numRows = 5;
for (let i = 1; i <= numRows; i++) {
 let row = "";
for (let j = 1; j <= i; j++) {
// Print either 1 or 0 based on the column index
if (j % 2 === 0) {
        row += "0 ";
} else {
        row += "1 ";
}
  }
  console.log(row);
}
```

2) A
B C
D E F
G H I J
K L M N O

```
let charCode = 65; // ASCII code for 'A'
const numRows = 5;
for (let i = 0; i < numRows; i++) {
   let row = "";
   for (let j = 0; j <= i; j++) {
      // Convert the ASCII code to a character and append it to the row
      row += String.fromCharCode(charCode) + " ";
      charCode++; // Increment the ASCII code for the next character
```

```
        }
        console.log(row);
    }
```

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

```
const numRows = 5;
let counter = 1;
for (let i = 1; i <= numRows; i++) {
    let row = "";
    for (let j = 1; j <= i; j++) {
        row += counter + " ";
        counter++;
    }
    console.log(row);
}
```

4) *

*** ***

*** * ***

*** * * ***

*** * * * ***

```
const numRows = 5;
for (let i = 1; i <= numRows; i++) {
    let row = "";
    for (let j = 1; j <= numRows - i; j++) {
        row += " ";
    }
    for (let k = 1; k <= i; k++) {
        row += "* ";
    }
    console.log(row);
```

```
}
```

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

```
// Function to check if a number is palindrome
function isPalindrome(number) {
    let numString = number.toString();
    let reversedString = numString.split('').reverse().join('');
    if (numString === reversedString) {
        return true;
    } else {
        return false;
    }
}
// Initialize variables to store user input
let counter = 0;
let numbers = [];

while (counter < 3) {
    let userInput = parseInt(prompt("Enter number " + (counter + 1) + ":"));

    // Check if the number is a palindrome
    if (isPalindrome(userInput)) {
        console.log(userInput + " is a palindrome.");
    } else {
        console.log(userInput + " is not a palindrome.");
    }

    // Store the number in the array
    numbers.push(userInput);
```

```
    counter++;
}
```

(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 ?

```
function getCurrentDay() {
    const days = ['Sunday', 'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday'];
    const now = new Date();
    return days[now.getDay()];
}
function getCurrentTime() {
    const now = new Date();
    let hours = now.getHours();
    let ampm = hours >= 12 ? 'PM' : 'AM';
    hours = hours % 12;
    hours = hours ? hours : 12; // handle midnight (0 hours)
    const minutes = now.getMinutes();
    const seconds = now.getSeconds();
    return `${hours} ${ampm} : ${minutes} : ${seconds}`;
}
const currentDay = getCurrentDay();
const currentTime = getCurrentTime();

console.log(`Today is ${currentDay}. Current Time is ${currentTime}`);
```

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

```
function getCurrentDate() {
    const now = new Date();
    const year = now.getFullYear();
```

```
    const month = now.getMonth() + 1; // Adding 1 because getMonth() returns zero-based month index
    const day = now.getDate();
    return `${year}-${month < 10 ? '0' + month : month}-${day < 10 ? '0' + day : day}`;
}
const currentDate = getCurrentDate();
console.log(`Current date is: ${currentDate}`);
```

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

```
function compareObjects(obj1, obj2) {
    // Get the keys of both objects
    const keys1 = Object.keys(obj1);
    const keys2 = Object.keys(obj2);
    // Check if the number of keys is the same
    if (keys1.length !== keys2.length) {
        return false;
    }
    // Check if all keys in obj1 exist in obj2 and have the same values
    for (let key of keys1) {
        if (!obj2.hasOwnProperty(key) || obj1[key] !== obj2[key]) {
            return false;
        }
    }
    return true;
}
const obj1 = { a: 1, b: 2, c: 3 };
const obj2 = { a: 1, b: 2, c: 3 };
const obj3 = { a: 1, b: 2, d: 4 };

// Comparing objects
console.log(compareObjects(obj1, obj2)); // Output: true
```

```
console.log(compareObjects(obj1, obj3)); // Output: false
```

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

```
function arrayObjectsToCSV(data) {
    // Extract keys from the first object to use as header row
    const header = Object.keys(data[0]);
    // Create CSV header row
    const csvHeader = header.join(',') + '\n';

    // Create CSV rows
    const csvRows = data.map(obj => {
        return header.map(key => {
            return obj[key];
        }).join(',');
    }).join('\n');

    // Concatenate header and rows to form CSV string
    return csvHeader + csvRows;
}

// Example array of objects
const data = [
    { name: 'John', age: 30, city: 'New York' },
    { name: 'Alice', age: 25, city: 'Los Angeles' },
    { name: 'Bob', age: 35, city: 'Chicago' }
];

// Convert array of objects to CSV string
const csvString = arrayObjectsToCSV(data);
```

```
console.log(csvString);
```

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

```
function capitalizeFirstLetter(str) {
   return str.charAt(0).toUpperCase() + str.slice(1);
}
const inputString = "hello world";
// Capitalize the first letter
const capitalizedString = capitalizeFirstLetter(inputString);
console.log(capitalizedString); // Output: "Hello world"
```

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

```
function isArray(variable) {
   return Array.isArray(variable);
}
const arr = [1, 2, 3];
const notArr = "Hello";
console.log(isArray(arr)); // Output: true
console.log(isArray(notArr)); // Output: false
```

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

```
function cloneArray(arr) {
   return [...arr];
}
const originalArray = [1, 2, 3];
const clonedArray = cloneArray(originalArray);
console.log(clonedArray); // Output: [1, 2, 3]
```

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

```
function Person(name) {
   this.name = name;
```

```
    this.greet = function() { // Method declared directly in the object
        console.log(`Hello, my name is ${this.name}`);
    };}
// Create two instances of Person
const person1 = new Person('Alice');
const person2 = new Person('Bob');
// Modify the greet method for person1
person1.greet = function() {
    console.log('Custom greeting for person1');
};
// Call greet method for both instances
person1.greet(); // Output: Custom greeting for person1
person2.greet(); // Output: Hello, my name is Bob
```

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

```
const myString = "Hello, world!";
console.log(myString.length);
```

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

```
const myString = "hello, world!";
const capitalizedString = myString.toUpperCase();
console.log(capitalizedString);
```

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

```
function Person(name) {
    this.name = name;
    this.greet = function() {
        console.log(`Hello, my name is ${this.name}`);
    };
}
const person1 = new Person('Alice');
```

```
const person2 = new Person('Bob');


console.log(person1.greet === person2.greet); // Output: false
```

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?

```
function getCurrentDate(format) {
   const now = new Date();
   const year = now.getFullYear();
   let month = now.getMonth() + 1;
   let day = now.getDate();
   // Add leading zero if month/day is less than 10
   if (month < 10) {
      month = '0' + month;
   }
   if (day < 10) {
      day = '0' + day;
   }
   // Determine the separator based on the format
   let separator;
   if (format.includes('-')) {
      separator = '-';
   } else if (format.includes('/')) {
      separator = '/';
   } else {
      separator = '-';
   }
   const formattedDate = format.replace('mm', month).replace('dd',
day).replace('yyyy', year);
   return formattedDate;
}
console.log(getCurrentDate('mm-dd-yyyy')); // Output: "04-23-2024"
```

```
console.log(getCurrentDate('mm/dd/yyyy')); // Output: "04/23/2024"
console.log(getCurrentDate('dd-mm-yyyy')); // Output: "23-04-2024"
console.log(getCurrentDate('dd/mm/yyyy')); // Output: "23/04/2024"
```

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

```
const sentence = "30 Days Of JavaScript";
const position = sentence.indexOf('a');
console.log("The position of the first occurrence of 'a':", position);
//Output : The position of the first occurrence of 'a': 5
```

Because the pdf :

Q.52 Use indexOf to determine the position of the first occurrence of a in 30 Days Of JavaScript?
Q,53 Use lastIndexOf to determine the position of the last occurrence of a in 30 Days Of JavaScript?

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

```
const sentence = "30 Days Of JavaScript";
const position = sentence.lastIndexOf('a');
console.log("The position of the last occurrence of 'a':", position);
//Output : The position of the last occurrence of 'a': 19
```

Q-54. Form Validtion in JS?

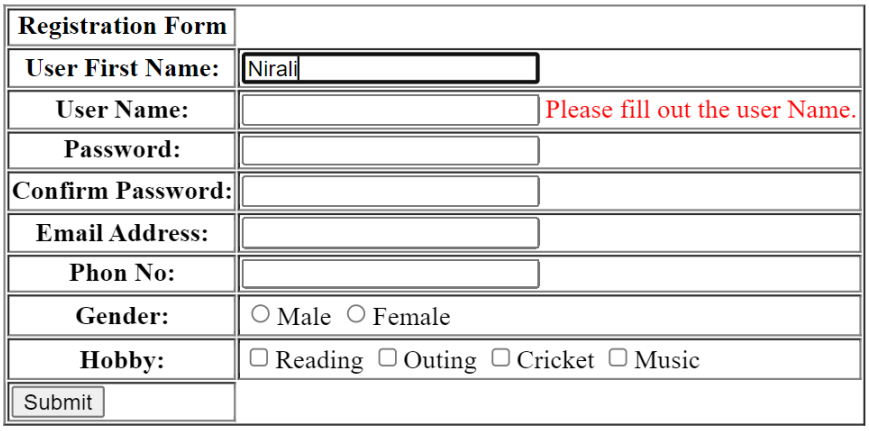
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <script>
        function validate() {
            let ufn = document.forms["myForm"]["ufn"].value;
            if (ufn == "" || ufn == null) {
                alert("Please fill out the User first Name.");
```

```
            return false;
        }
        let alpha = /[A-Za-z]+$/;
        if (!alpha.test(ufn)) {
            alert("Please fill out Alphabate User First Name.");
            return false;
        }
        let un = document.forms["myForm"]["un"].value;
        if (un == "" || un == null) {
            document.getElementById("un").innerText = "Please fill out the user
Name.";
            return false;
        }else{
            document.getElementById("un").innerText = "";
        }
        let pass = document.forms["myForm"]["pass"].value;
        if (pass == "" || pass == null) {
            alert("Please fill out Password.");
            return false;
        }
        if (!(pass.length >= 3 && pass.length <=8)) {
            alert("Password should be min 3 or max 8 char.");
            return false;
        }
        let c_pass = document.forms["myForm"]["c_pass"].value;
        if (c_pass == "" || c_pass == null) {
            alert("Please fill out Confirm Password.");
            return false;
        }
        if (c_pass == pass) {}else{
            alert("Please Enter the same Value.");
```

```
        return false;
    }
    let email = document.forms["myForm"]["mail"].value;
    if (email == "" || email == null) {
        alert("Please fill out Email.");
        return false;
    }
    let mail = /[A-Za-z0-9_]+@[a-zA-Z]+\.[a-zA-Z]{2,4}$/;
    if (!mail.test(mail)) {
        alert("Please fill proper email id.");
        return false;
    }
    let pno = document.forms["myForm"]["pno"].value;
    if (pno == "" || pno == null) {
        alert("Please fill out Phone Number.");
        return false;
    }
    let phone = /[0-9]{10,11}$/;
    if(!phone.test(pno)){
        alert("Please fill only 10 Digits.");
        return false;
    }
    let gen_arr = document.getElementsByName("gender");
    if (gen_arr[0].checked == true) {}else if (gen_arr[1].checked == true)
{}else{alert("!Please Select Gender.");
        return false;}
    let chk_arr = document.getElementsByName("chk");
    if (chk_arr[0].checked == true) {} else if(chk_arr[1].checked == true) {}
    else if(chk_arr[2].checked == true) {}else if(chk_arr[3].checked == true) {}
    else{
        alert("!Please select at least one hobby.");
```

```
            return false;
        }
    }
    </script>
</head>
<body>
    <form action="" name="myForm" method="post" onsubmit="return validate()">
        <table border="2" align="center">
            <tr>
                <th>Registration Form</th>
            </tr>
            <tr>
                <th>User First Name:</th>
                <td><input type="text" name="ufn"></td>
            </tr>
            <tr>
                <th>User Name:</th>
                <td><input type="text" name="un">
                    <span style="color: red;" id="un"></span>
                </td>
            </tr>
            <tr>
                <th>Password:</th>
                <td><input type="password" name="pass"></td>
            </tr>
            <tr>
                <th>Confirm Password:</th>
                <td><input type="password" name="c_pass"></td>
            </tr>
            <tr>
```

```
            <th>Email Address:</th>
            <td><input type="email" name="mail"></td>
        </tr>
        <tr>
            <th>Phon No:</th>
            <td><input type="number" name="pno"></td>
        </tr>
        <tr>
            <th>Gender:</th>
            <td><input type="radio" name="gender" value="m">Male
                <input type="radio" name="gender" value="f">Female</td>
        </tr>
        <tr>
            <th>Hobby:</th>
            <td><input type="checkbox" name="chk">Reading
                <input type="checkbox" name="chk">Outing
                <input type="checkbox" name="chk">Cricket
                <input type="checkbox" name="chk">Music</td>
        </tr>
        <tr>
            <td><input type=”Submit”</td>
        </tr>
    </table>
  </form>
</body>
</html>
```



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

```
<!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>
<h2>Simple Form Validation</h2>
<form id="myForm">
    <label for="email">Email:</label>
    <input type="email" id="email" name="email" required><br><br>
    <label for="phone">Phone Number:</label>
    <input type="tel" id="phone" name="phone" pattern="[0-9]{10}"
required><br><br>
    <label for="password">Password:</label>
    <input type="password" id="password" name="password" required><br><br>
    <button type="submit">Submit</button>
</form>
<script>
document.getElementById('myForm').addEventListener('submit', function(event) {
    const email = document.getElementById('email').value;
```

```
    const phone = document.getElementById('phone').value;
    const password = document.getElementById('password').value;
    if (!isValidEmail(email)) {
        alert('Please enter a valid email address');
        event.preventDefault(); // Prevent form submission
    } else if (!isValidPhoneNumber(phone)) {
        alert('Please enter a valid 10-digit phone number');
        event.preventDefault(); // Prevent form submission
    } else if (password.length < 6) {
        alert('Password should be at least 6 characters long');
        event.preventDefault(); // Prevent form submission
    } else {
        alert('Form submitted successfully');
        // Optionally, you can submit the form here
    }
});
function isValidEmail(email) {
    // Regular expression for email validation
    const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
    return emailRegex.test(email);
}
function isValidPhoneNumber(phone) {
    // Regular expression for 10-digit phone number validation
    const phoneRegex = /^\d{10}$/;
    return phoneRegex.test(phone);
}
</script>
</body>
</html>
```

Q-56. Dynamic Form Validation in JS?

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Dynamic Form Validation</title>
<style>
   .error-message {
      color: red;
   }
</style>
</head>
<body>
<h2>Dynamic Form Validation</h2>
<form id="myForm">
   <label for="email">Email:</label>
   <input type="email" id="email" name="email" required><span id="emailError"
class="error-message"></span><br><br>
   <label for="password">Password:</label>
   <input type="password" id="password" name="password" required><span
id="passwordError" class="error-message"></span><br><br>
   <button type="submit">Submit</button>
</form>
<script>
const form = document.getElementById('myForm');
form.addEventListener('input', function(event) {
   if (event.target.tagName === 'INPUT') {
      validateInput(event.target);
   }
});
form.addEventListener('submit', function(event) {
```

```
    const emailValid = validateInput(document.getElementById('email'));
    const passwordValid = validateInput(document.getElementById('password'));
    if (!emailValid || !passwordValid) {
        event.preventDefault(); // Prevent form submission if any field is invalid
    }
});
function validateInput(input) {
    const errorElement = document.getElementById(input.id + 'Error');
    const value = input.value.trim();
    let valid = true;
    if (value === '') {
        errorElement.textContent = 'This field is required';
        valid = false;
    } else {
        errorElement.textContent = '';
    }
    return valid;
}
</script>
</body>
</html>
```

Q-57. how many type of JS Event? How to use it?

1. Mouse Events:
   - click: Triggered when the user clicks on an element.
   - mouseover: Triggered when the mouse pointer enters the area of an element.
   - mouseout: Triggered when the mouse pointer leaves the area of an element.
   - mousemove: Triggered when the mouse pointer moves over an element.
2. Keyboard Events:
   - keydown: Triggered when a key is pressed down.
   - keyup: Triggered when a key is released.
   - keypress: Triggered when a key is pressed down and then released.
3. Form Events:
   - submit: Triggered when a form is submitted.
   - change: Triggered when the value of an input element changes.

- focus: Triggered when an element receives focus.
- blur: Triggered when an element loses focus.

4. Window Events:
   - load: Triggered when the page and all its resources have finished loading.
   - resize: Triggered when the browser window is resized.
   - scroll: Triggered when the user scrolls the page.

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Event Handling</title>
</head>
<body>
<button id="myButton">Click Me!</button>
<script>
const button = document.getElementById('myButton');
// Add a click event listener to the button
button.addEventListener('click', function(event) {
   alert('Button clicked!');
});
button.addEventListener('mouseover', function(event) {
   console.log('Mouse over the button');
});
document.addEventListener('keydown', function(event) {
   console.log('Key pressed:', event.key);
});
const form = document.createElement('form');
```

```
document.body.appendChild(form);
form.addEventListener('submit', function(event) {
   event.preventDefault(); // Prevent form submission
   console.log('Form submitted!');
});
</script>
</body>
</html>
```

Q-59. What is Bom vs Dom in JS?

Q.57 how many type of JS Event? How to use it?
Q.59 What is Bom vs Dom in JS?

1. **DOM (Document Object Model):**

- The DOM represents the structure of an HTML document as a hierarchical tree of objects. Each element in the HTML document, such as , <p>, , etc., is represented by a DOM object.
- The DOM provides a way for JavaScript to interact with and manipulate the content and structure of the HTML document dynamically. You can use DOM methods and properties to access, modify, add, or remove elements and their attributes, content, and styles.
- For example, you can use DOM methods like getElementById(), querySelector(), appendChild(), setAttribute(), etc., to interact with HTML elements.

2. **BOM (Browser Object Model):**

- The BOM represents everything else in the browser outside of the web page content. It includes objects and interfaces that provide interaction with the browser window and its components like the address bar, history, navigator, screen, etc.
- The BOM provides JavaScript access to browser-specific features and functionalities such as controlling the browser window, managing cookies, detecting the user's browser and operating system, navigating the browser history, opening new windows or tabs, and more.
- For example, you can use BOM properties like window.location, window.open(), window.alert(), navigator.userAgent, etc., to interact with the browser itself.

Q-60. Array vs object defences in JS?

Array	Objects
Arrays are best to use when the elements are **numbers**.	Objects are best to use when the elements' **strings (text)**.
The data inside an array is known	The data inside objects are known

Array	Objects
as **Elements.**	as **Properties** which consists of a **key** and a **value.**
The elements can be manipulated using [].	The properties can be manipulated using both**.DOT** notation and [].
The elements can be popped out of an array using the **pop()** function.	The keys or properties can be deleted by using the **delete** keyword.
Iterating through an array is possible using **For loop, For..in, For..of**, and **ForEach().**	Iterating through an array of objects is possible using **For..in, For..of, and ForEach().**

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

```
const sentence = "Hello, world! How are you?";
const wordsArray = sentence.split(' ');
console.log(wordsArray);
```

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

```
var myString = "This is a JavaScript script";
if (myString.includes("Script")) {
   console.log("The string contains the word 'Script'");
} else {
   console.log("The string does not contain the word 'Script'");
}
```

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

```
var myString = "This is a JavaScript Script";
var lowerCaseString = myString.toLowerCase();
console.log(lowerCaseString);
//Output : this is a javascript script
```

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

```
var myString = '30 Days of JavaScript';
```

```
// Get the character at index 15
var charAtIndex15 = myString.charAt(15);
console.log("Character at index 15:", charAtIndex15);
//Output : Character at index 15: S
```

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

```
var sourceString = "This is the source string.";
// Copy the source string to another string
var destinationString = sourceString;
console.log("Source String:", sourceString);
console.log("Destination String:", destinationString);
```

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

```
var myString = "Hello, World!";
var length = 0;
// Iterate over each character in the string
for (var i = 0; i < myString.length; i++) {
   length++;
}
console.log("Length of the string:", length);
//Output : Length of the string: 13
```

- **What is JavaScript?**
  - JavaScript is a programming language commonly used in web development. It's versatile, allowing developers to add interactive elements, dynamic content, and behavior to websites.
  - It runs on the client side, meaning it's executed by your web browser, enhancing user experience without needing constant communication with the server.
  - JavaScript is what makes websites more than just static pages.
  - It brings them to life, enabling things like interactive forms, animations, and dynamic updates, creating a more engaging and interactive online experience.
- **What is the use of isNaN function?**
  - The **'isNaN()'** function in JavaScript stands for "is Not a Number".
  - It's used to check whether a value is not a valid number.
  - It returns **'true'** if the value is not a number, and **'false'** if it is.
  - **'isNaN()'** helps us figure out if something we're dealing with is actually a number or not.

- It's handy for validation, like checking user input to make sure it's a number before doing calculations with it.

➢ What is negative Infinity?

- Negative Infinity in JavaScript represents the smallest possible negative number, which is beyond the normal range of numbers that JavaScript can represent.
- It's denoted by the keyword **'Number.NEGATIVE_INFINITY'**.
- Negative Infinity is like the opposite of Infinity.
- While Infinity represents the largest possible number, Negative Infinity represents the smallest possible number.
- It's often used to represent mathematical operations or values that tend towards negative infinity.

➢ Which company developed JavaScript?

- JavaScript was developed by Netscape Communications Corporation, a company that played a significant role in the early days of the internet.
- Later, JavaScript was standardized by Ecma International as ECMAScript. So, Netscape is credited with the initial development of JavaScript.

➢ What are undeclared and undefined variables?

1) **Undeclared Variables :**
   - These are variables that have not been declared using a **var, let or const** statement before they are used.
   - If you try to access the value of an undeclared variable, JavaScript will throw a reference error.
2) **Undefined Variables :**
   - These are variables that have been declared but have not been assigned a value.
   - When you try to access the value of an undefined variable, JavaScript will return **undifined**.
   - This means the variable exists, but it hasn't been given a value yet.

➢ Write the code for adding new elements dynamically?

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Dynamic Element Addition</title>
</head>
<body>
<!-- This is where we'll add new elements -->
<div id="container">
  <!-- Existing content -->
  <p>Existing content here.</p>
</div>
<!-- Button to add a new element -->
<button onclick="addElement()">Add Element</button>
```

```
<script>
function addElement() {
  // Create a new element
  var newElement = document.createElement("p");

  // Create some text content
  var text = document.createTextNode("This is a new paragraph.");

  // Append the text to the new element
  newElement.appendChild(text);

  // Get the container where we want to add the new element
  var container = document.getElementById("container");

  // Append the new element to the container
  container.appendChild(newElement);
}
</script>
</body>
</html>
```

- **What is the difference between ViewState and SessionState?**

ViewState	SessionState
Maintained at page level only.	Maintained at session level.
View state can only be visible from a single page and not multiple pages.	Session state value availability is across all pages available in a user session.
It will retain values in the event of a postback operation occurring.	In session state, user data remains in the server. Data is available to user until the browser is closed or there is session expiration.
Information is stored on the client's end only.	Information is stored on the server.
used to allow the persistence of page-instance-specific data.	used for the persistence of user-specific data on the server's end.
ViewState values are lost/cleared when new page is loaded.	SessionState can be cleared by programmer or user or in case of timeouts.

➢ What is === operator?

- The === operator in JavaScript is called the "strict equality" operator. It compares two values without performing type conversion.
- It checks if both the value and the type of two variables are the same. If they are, it returns **true**, otherwise **false**.
- For example**, 3 === 3** would return **true** because both the value and the type of both operands are the same (both are numbers). But **3 === '3'** would return **false** because although the values are the same, the types are different (one is a number and the other is a string).

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

- **Changing Style Directly**

```
// Assuming you have an element with id "myElement"
var element = document.getElementById("myElement");

// Change its background color
element.style.backgroundColor = "blue";

// Change its font size
element.style.fontSize = "20px";
```

- **Adding/Removing Classes**

```
// Assuming you have an element with id "myElement"
var element = document.getElementById("myElement");

// Add a class to the element
element.classList.add("newClass");

// Remove a class from the element
element.classList.remove("oldClass");
```

➢ How to read and write a file using JavaScript?

- In a web browser environment, JavaScript doesn't have direct access to the file system for security reasons.
- However, you can achieve file read and write operations in JavaScript using the FileReader and FileWriter APIs.
- These APIs are typically used in conjunction with user interactions, such as file input fields or drag-and-drop interfaces.

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

<input type="file" id="fileInput">
```

```
<div id="fileContent"></div>

<script>
document.getElementById('fileInput').addEventListener('change',
function(event) {
   var file = event.target.files[0];
   var reader = new FileReader();

   reader.onload = function(event) {
      var contents = event.target.result;
      document.getElementById('fileContent').textContent = contents;
   };

   reader.readAsText(file);
});
</script>

</body>
</html>
```

- We have an <input type="file"> element where the user can select a file.
- We add an event listener to the input element that triggers when the user selects a file.
- Inside the event listener, we create a new FileReader object and use it to read the contents of the selected file.
- Once the file is loaded, the onload event handler is called, and we display the contents of the file in a element.

➢ What are all the looping structures in JavaScript?

- **For Loop:** A for loop repeats a block of code a specified number of times.

```
for (initialization; condition; increment/decrement) {
 // code to be executed
}
```

- **While Loop:** A while loop repeats a block of code as long as a specified condition is true.

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

- **Do...While Loop:** A do...while loop is similar to a while loop, but it will always execute the block of code once before checking the condition.

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

➢ How can you convert the string of any base to an integer in JavaScript?

- To convert a string representing a number in any base to an integer in JavaScript, you can use the parseInt() function.
- The parseInt() function parses a string and returns an integer based on the specified radix (or base).

```
var str = "1010";
// Convert the string to an integer in base 2 (binary)
var num = parseInt(str, 2);
console.log(num); // Output: 10
```

➢ What is the function of the delete operator?

```
var person = {
  name: 'John',
  age: 30,
  city: 'New York'
};

// Delete the 'age' property from the 'person' object
delete person.age;

// Now 'person' object doesn't have the 'age' property
console.log(person); // Output: { name: 'John', city: 'New York' }
```

➢ What are all the types of Pop up boxes available in JavaScript?

- Alert Box: Displays a message to the user in a small pop-up window with an OK button.

```
alert("This is an alert box!");
```

- Confirm Box: Prompts the user with a message and provides OK and Cancel buttons for confirmation. It returns true if the user clicks OK and false if the user clicks Cancel.

```
var result = confirm("Do you want to proceed?");
if (result === true) {
  // Code to execute if user clicks OK
} else {
  // Code to execute if user clicks Cancel
}
```

- Prompt Box: Prompts the user to enter some input. It provides an input field along with OK and Cancel buttons. It returns the text entered by the user as a string if the user clicks OK, and null if the user clicks Cancel.

```
var name = prompt("Please enter your name:");
if (name != null) {
  alert("Hello, " + name + "!");
} else {
  alert("You cancelled the prompt.");
}
```

➢ What is the use of Void (0)?

- The void(0) expression in JavaScript is often used in combination with an anchor <a> tag to prevent the browser from following the link when clicked.

  For example:

  Click here

- This is often used when you want to create a clickable element that performs some JavaScript action, but you don't want the browser to navigate to a new page or reload the current page.

- In simple terms, void(0) is used to create clickable elements that execute JavaScript without causing the browser to navigate away from the current page.

➢ How can a page be forced to load another page in JavaScript?

- In JavaScript, you can force a page to load another page by setting the window.location property to the URL of the page you want to navigate to.

```
// Redirect to another page
window.location.href = "https://www.example.com";
```

  - This code will immediately navigate the browser to the specified URL, loading the new page.
  - setting window.location.href to a new URL is like telling the browser to go to that webpage**.**

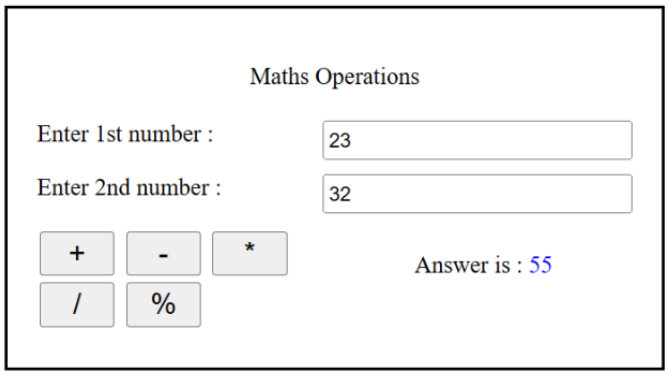
➢ What are the disadvantages of using innerHTML in JavaScript?

- Security Risk: Setting innerHTML with user-provided content can expose your website to cross-site scripting (XSS) attacks if the content is not properly sanitized. This is because innerHTML can execute any JavaScript code included in the content, potentially allowing attackers to inject malicious scripts.
- Performance Overhead: Manipulating innerHTML causes the browser to re-parse and re-render the entire HTML content of the element, which can be inefficient, especially for large or complex elements. This can impact the performance of your web page, especially on mobile devices or older browsers.
- Event Handlers: When using innerHTML to replace or modify content that contains event handlers, the existing event handlers may be lost or need to be reattached manually after the content is updated. This can lead to unexpected behavior and maintenance challenges.
- Lack of DOM Awareness: When you use innerHTML, the browser treats the content as a string of HTML, not as structured DOM elements. This can make it more difficult to interact with the modified content using JavaScript, especially if you need to access or manipulate specific elements within it.

➢ Create password field with show hide functionalities


```
<!DOCTYPE html>
<html>
<head>
<title>JS Toggle Password Visibility</title>
</head>
Enter Password: <input type="password" value="javatpoint" id="pswrd"></br>
<input type="checkbox" onclick="toggleVisibility()"/>Show Password</br>
<body>
<script>
function toggleVisibility() {
  var getPasword = document.getElementById("pswrd");
  if (getPasword.type === "password") {
    getPasword.type = "text";
  } else {
    getPasword.type = "password";
  }
}
</script>
</body>
</html>
```

➢ Create basic math operation in JS

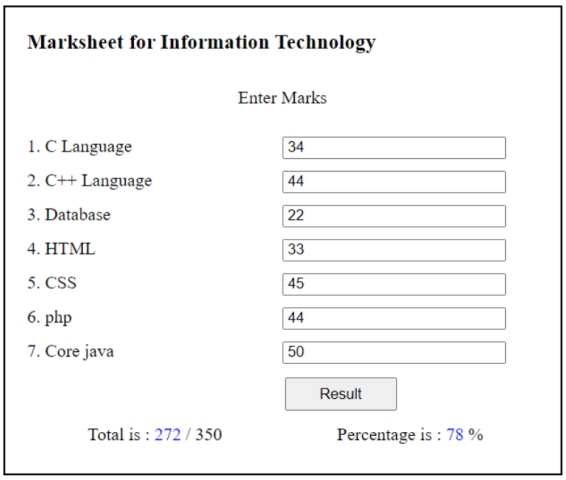



```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
```

```
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Calculator</title>
    <style>
        *{
            margin: 20px;
            padding: 10px;
        }
        div{
            width: 400px;
            height: auto;
            border: 2px solid gray;
            padding: 30px;
        }

    </style>
</head>
<body>
<div>
    <h1>Maths Operations</h1>

        Enter 1st number : <input type="number" id="one"><br>
        Enter 2nd number : <input type="number" id="two"><br>
    </tr>
    <button onclick="add()">+</button>
    <button onclick="sub()">-</button>
    <button onclick="mul()">*</button><br>
    <button onclick="div()">/</button>
    <button onclick="mod()">%</button>
    <span>Answer is : </span>
    <span id="ans">0</span>
</div>

</body>
<script>

    function add() {
        let a = document.getElementById("one").value
```

```
        let b = document.getElementById("two").value
        document.getElementById("ans").innerText = a+b;
    }
    function sub() {
        let a = document.getElementById("one").value
    let b = document.getElementById("two").value
        document.getElementById("ans").innerText = a-b;
    }
    function mul() {
        let a = document.getElementById("one").value
    let b = document.getElementById("two").value
        document.getElementById("ans").innerText = a*b;
    }
    function div() {
        let a = document.getElementById("one").value
    let b = document.getElementById("two").value
        document.getElementById("ans").innerText = a/b;
    }
    function mod() {
        let a = document.getElementById("one").value
    let b = document.getElementById("two").value
        document.getElementById("ans").innerText = a%b;
    }
</script>
</html>
```

- **Create result**

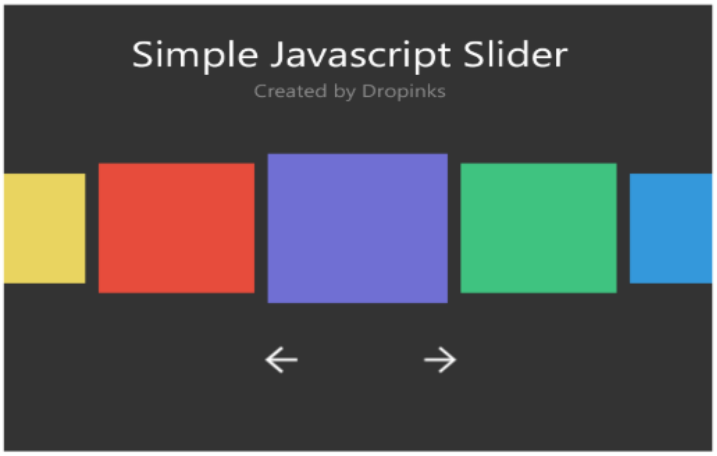



```
<!DOCTYPE html>
```

```
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Calculator</title>
   <style>
      *{
         margin: 20px;
         padding: 10px;
      }
      div{
         width: 400px;
         height: auto;
         border: 2px solid gray;
         padding: 30px;
      }

   </style>
</head>
<body>
<div>
   <h1>Marksheet for Information Technology</h1>

      1. C Language : <input type="number" id="one"><br>
      2. C++ Language : <input type="number" id="two"><br>
      3. Database : <input type="number" id="three"><br>
      4. HTML : <input type="number" id="fore"><br>
      5. CSS : <input type="number" id="five"><br>
      6. php : <input type="number" id="six"><br>
      7. Core java : <input type="number" id="seven"><br>

   </tr>
   <button onclick="add()">Result</button>

   <span>Total is : </span>
   <span id="ans"></span><p>/350</p>
   <span>Percentage is : </span>
```

```
    <span id="ans1"></span><p>%</p>
</div>

</body>
<script>

    function add() {
        let a = document.getElementById("one").value
        let b = document.getElementById("two").value
        let c = document.getElementById("three").value
        let d = document.getElementById("fore").value
        let e = document.getElementById("five").value
        let f = document.getElementById("six").value
        let g = document.getElementById("seven").value
        let ans = a + b + c + d + e + f + g;
        document.getElementById("ans").innerText = ans;
        let ans1 = ans / 7;
        document.getElementById("ans").innerText = ans1;
    }
</script>
</html>
```

- **Create a slider using JavaScript**




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

```
        .slider {
  width: 100%;
  height: 510px;
  position: relative;
}

.slider img {
  width: 100%;
  height: 500px
  position : absolute;

}

.slider img:first-child {
  z-index: 1;
}

.slider img:nth-child(2) {
  z-index: 0;
}
.navigation-button {
  text-align: center;
  position: relative;
}

.dot {
  cursor: pointer;
  height: 15px;
  width: 15px;
  margin: 0 2px;
  background-color: #bbb;
  border-radius: 50%;
  display: inline-block;
}

.active,
.dot:hover {
  background-color: #717171;}
    </style>
</head>
<body>
    <div class="slider">
        <img id="img-1" src="img-1.jpg" alt="Image 1"/>
        <img id="img-2" src="img-2.jpg" alt="Image 2"/>
        <img id="img-3" src="img-3.jpg" alt="Image 3"/>
      </div>
```

```
        <div class="navigation-button">
          <span class="dot active" onclick="changeSlide(0)"></span>
          <span class="dot" onclick="changeSlide(1)"></span>
          <span class="dot" onclick="changeSlide(2)"></span>
        </div>
    </body>
    <script>
       var imgs = document.querySelectorAll('.slider img');
    var dots = document.querySelectorAll('.dot');
    var currentImg = 0; // index of the first image
    const interval = 3000; // duration(speed) of the slide
    function changeSlide(n) {
      for (var i = 0; i < imgs.length; i++) { // reset
        imgs[i].style.opacity = 0;
        dots[i].className = dots[i].className.replace(' active', '');
      }

      currentImg = n;

      imgs[currentImg].style.opacity = 1;
      dots[currentImg].className += ' active';
    }
    </script>
    </html>
```

- **What is JavaScript?**
  - JavaScript is a programming language commonly used in web development. It's versatile, allowing developers to add interactive elements, dynamic content, and behavior to websites.
  - It runs on the client side, meaning it's executed by your web browser, enhancing user experience without needing constant communication with the server.
  - JavaScript is what makes websites more than just static pages.
  - It brings them to life, enabling things like interactive forms, animations, and dynamic updates, creating a more engaging and interactive online experience.
- **What is the use of isNaN function?**
  - The **'isNaN()'** function in JavaScript stands for "is Not a Number".
  - It's used to check whether a value is not a valid number.
  - It returns **'true'** if the value is not a number, and **'false'** if it is.
  - **'isNaN()'** helps us figure out if something we're dealing with is actually a number or not.
  - It's handy for validation, like checking user input to make sure it's a number before doing calculations with it.

➢ What is negative Infinity?

- Negative Infinity in JavaScript represents the smallest possible negative number, which is beyond the normal range of numbers that JavaScript can represent.
- It's denoted by the keyword **'Number.NEGATIVE_INFINITY'**.
- Negative Infinity is like the opposite of Infinity.
- While Infinity represents the largest possible number, Negative Infinity represents the smallest possible number.
- It's often used to represent mathematical operations or values that tend towards negative infinity.

➢ Which company developed JavaScript?

- JavaScript was developed by Netscape Communications Corporation, a company that played a significant role in the early days of the internet.
- Later, JavaScript was standardized by Ecma International as ECMAScript. So, Netscape is credited with the initial development of JavaScript.

➢ What are undeclared and undefined variables?

3) Undeclared Variables :

- These are variables that have not been declared using a **var, let or const** statement before they are used.
- If you try to access the value of an undeclared variable, JavaScript will throw a reference error.

4) Undefined Variables :

- These are variables that have been declared but have not been assigned a value.
- When you try to access the value of an undefined variable, JavaScript will return **undifined**.
- This means the variable exists, but it hasn't been given a value yet.

➢ Write the code for adding new elements dynamically?

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Dynamic Element Addition</title>
</head>
<body>
<!-- This is where we'll add new elements -->
<div id="container">
  <!-- Existing content -->
  <p>Existing content here.</p>
</div>
<!-- Button to add a new element -->
<button onclick="addElement()">Add Element</button>

<script>
```

```
function addElement() {
  // Create a new element
  var newElement = document.createElement("p");

  // Create some text content
  var text = document.createTextNode("This is a new paragraph.");

  // Append the text to the new element
  newElement.appendChild(text);

  // Get the container where we want to add the new element
  var container = document.getElementById("container");

  // Append the new element to the container
  container.appendChild(newElement);
}
</script>
</body>
</html>
```

- **What is the difference between ViewState and SessionState?**

ViewState	SessionState
Maintained at page level only.	Maintained at session level.
View state can only be visible from a single page and not multiple pages.	Session state value availability is across all pages available in a user session.
It will retain values in the event of a postback operation occurring.	In session state, user data remains in the server. Data is available to user until the browser is closed or there is session expiration.
Information is stored on the client's end only.	Information is stored on the server.
used to allow the persistence of page-instance-specific data.	used for the persistence of user-specific data on the server's end.
ViewState values are lost/cleared when new page is loaded.	SessionState can be cleared by programmer or user or in case of timeouts.

➢ What is === operator?

- The === operator in JavaScript is called the "strict equality" operator. It compares two values without performing type conversion.
- It checks if both the value and the type of two variables are the same. If they are, it returns **true**, otherwise **false**.
- For example**, 3 === 3** would return **true** because both the value and the type of both operands are the same (both are numbers). But **3 === '3'** would return **false** because although the values are the same, the types are different (one is a number and the other is a string).

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

- **Changing Style Directly**

```
// Assuming you have an element with id "myElement"
var element = document.getElementById("myElement");

// Change its background color
element.style.backgroundColor = "blue";

// Change its font size
element.style.fontSize = "20px";
```

- **Adding/Removing Classes**

```
// Assuming you have an element with id "myElement"
var element = document.getElementById("myElement");

// Add a class to the element
element.classList.add("newClass");

// Remove a class from the element
element.classList.remove("oldClass");
```

➢ How to read and write a file using JavaScript?

- In a web browser environment, JavaScript doesn't have direct access to the file system for security reasons.
- However, you can achieve file read and write operations in JavaScript using the FileReader and FileWriter APIs.
- These APIs are typically used in conjunction with user interactions, such as file input fields or drag-and-drop interfaces.

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

<input type="file" id="fileInput">
```

```
<div id="fileContent"></div>

<script>
document.getElementById('fileInput').addEventListener('change',
function(event) {
   var file = event.target.files[0];
   var reader = new FileReader();

   reader.onload = function(event) {
      var contents = event.target.result;
      document.getElementById('fileContent').textContent = contents;
   };

   reader.readAsText(file);
});
</script>

</body>
</html>
```

- We have an <input type="file"> element where the user can select a file.
- We add an event listener to the input element that triggers when the user selects a file.
- Inside the event listener, we create a new FileReader object and use it to read the contents of the selected file.
- Once the file is loaded, the onload event handler is called, and we display the contents of the file in a element.

➢ What are all the looping structures in JavaScript?

- **For Loop:** A for loop repeats a block of code a specified number of times.

```
for (initialization; condition; increment/decrement) {
 // code to be executed
}
```

- **While Loop:** A while loop repeats a block of code as long as a specified condition is true.

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

- **Do...While Loop:** A do...while loop is similar to a while loop, but it will always execute the block of code once before checking the condition.

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

➢ How can you convert the string of any base to an integer in JavaScript?

- To convert a string representing a number in any base to an integer in JavaScript, you can use the parseInt() function.
- The parseInt() function parses a string and returns an integer based on the specified radix (or base).

```
var str = "1010";
// Convert the string to an integer in base 2 (binary)
var num = parseInt(str, 2);
console.log(num); // Output: 10
```

➢ What is the function of the delete operator?

```
var person = {
  name: 'John',
  age: 30,
  city: 'New York'
};

// Delete the 'age' property from the 'person' object
delete person.age;

// Now 'person' object doesn't have the 'age' property
console.log(person); // Output: { name: 'John', city: 'New York' }
```

➢ What are all the types of Pop up boxes available in JavaScript?

- Alert Box: Displays a message to the user in a small pop-up window with an OK button.

```
alert("This is an alert box!");
```

- Confirm Box: Prompts the user with a message and provides OK and Cancel buttons for confirmation. It returns true if the user clicks OK and false if the user clicks Cancel.

```
var result = confirm("Do you want to proceed?");
if (result === true) {
  // Code to execute if user clicks OK
} else {
  // Code to execute if user clicks Cancel
}
```

- Prompt Box: Prompts the user to enter some input. It provides an input field along with OK and Cancel buttons. It returns the text entered by the user as a string if the user clicks OK, and null if the user clicks Cancel.

```
var name = prompt("Please enter your name:");
if (name != null) {
  alert("Hello, " + name + "!");
} else {
  alert("You cancelled the prompt.");
}
```

➢ What is the use of Void (0)?

- The void(0) expression in JavaScript is often used in combination with an anchor <a> tag to prevent the browser from following the link when clicked.

For example:

Click here

- This is often used when you want to create a clickable element that performs some JavaScript action, but you don't want the browser to navigate to a new page or reload the current page.

- In simple terms, void(0) is used to create clickable elements that execute JavaScript without causing the browser to navigate away from the current page.

➢ How can a page be forced to load another page in JavaScript?

- In JavaScript, you can force a page to load another page by setting the window.location property to the URL of the page you want to navigate to.

```
// Redirect to another page
window.location.href = "https://www.example.com";
```

- This code will immediately navigate the browser to the specified URL, loading the new page.
- setting window.location.href to a new URL is like telling the browser to go to that webpage**.**

➢ What are the disadvantages of using innerHTML in JavaScript?

- Security Risk: Setting innerHTML with user-provided content can expose your website to cross-site scripting (XSS) attacks if the content is not properly sanitized. This is because innerHTML can execute any JavaScript code included in the content, potentially allowing attackers to inject malicious scripts.
- Performance Overhead: Manipulating innerHTML causes the browser to re-parse and re-render the entire HTML content of the element, which can be inefficient, especially for large or complex elements. This can impact the performance of your web page, especially on mobile devices or older browsers.
- Event Handlers: When using innerHTML to replace or modify content that contains event handlers, the existing event handlers may be lost or need to be reattached manually after the content is updated. This can lead to unexpected behavior and maintenance challenges.
- Lack of DOM Awareness: When you use innerHTML, the browser treats the content as a string of HTML, not as structured DOM elements. This can make it more difficult to interact with the modified content using JavaScript, especially if you need to access or manipulate specific elements within it.