**Module (JAVASCRIPT BASIC & DOM) – 4**

**Q. What is JavaScript? How to use it?**

🡪 JavaScript is a high-level versatile, and interpreted

programming language primarily known for its role in

web development. It enables developers to add

interactivity, manipulate the Document Object

Model(DOM), and Create dynamic content within

browsers. JavaScript is an essential component of modern

web applications and is supported by all major web

browser.

🡪 How to use

1) In the Browser :- You can include JavaScript code

directly within HTML documents using **<script>**

tags. Place the script either in the **<head>** or at the

end of the **<body>** section.

2) External JavaScript File :- Create a separate

JavaScript file with the ‘.js’ extension and include it

in your HTML file using the **<script>** tag.

3) In Node.js(Server-Side) :- JavaScript can also be used

on the server-side with Node.js. Install Node.js on

your machine, create a JavaScript file, and run it

using the Node.js runtime.

4) Integrated in HTML/CSS :- JavaScript is often used

to enhance HTML and CSS interactions. For

example, responding to user events, manipulating the

DOM, or making asynchronous requests.

**Q. How many type of Variable in JavaScript?**

🡪 In JavaScript, there are three types of variables based on

how they are declared: **var**, **let**, and **const**. Each has

different characteristics and use cases:

1) var :- **var** was the original way to declare variables in

JavaScript. It has function-scoping (not block-scoping)

and lacks some of the features of let and **const.**

Variables declared with **var** are hoisted to the top of

heir scope.

Var x = 10;

2) let: **let** was introduced in ECMAScript 6 (ES6) and

provides block-scoping. It allows you to declare

variables that are limited to the block, statement, or

expression where they are used. Variables declared with

**let** are not hoisted to the top of their scope.

let y = 20;

3) const : **const** is also introduced in ES6 and is used to

declare constants. The value of a **const** variable cannot

be reassigned once it has been assigned. Like **let**, **const**

is block-scoped.

const PI = 3.14;

Variables declared with **let** and **const** are recommended

over **var** in modern JavaScript development due to

their improved scoping rules and additional features.

Choosing between **let** and **const** depends on whether

the variable's value needs to be reassigned or remain

constant.

**Q. Define a Data types in Js?**

🡪 JavaScript has several built-in data types that are used to

represent different kinds of values. The primary data

types in JavaScript can be categorized as follows:

1) Primitive Data Types:

1) Number :- Represents numeric values. It includes

integers and floating-point numbers.

var age = 25; // integer

var price = 19.99; // floating-point number

2) String :- Represents textual data enclosed in single

or double quotes.

var name = “Arnav”;

var message = “Hello, I am Aranv!”;

3) Boolean : Represents either ‘true’ or ‘false’.

var isStudent = true;

var hasLicense = false;

4) Null:- Represents the international absence of any

object value.

var myVariable = null;

5) Undefined :- Represent a variable that has been

declared but has not been assigned a value.

var x;

console.log(x); // Output: undefined

2) Non- Primitive Data Type :- Non-primitive data

types in JavaScript are objects. Unlike primitive

data types (such as numbers, strings, booleans,

null, undefined, and symbols), objects are more

complex and can hold multiple values and

methods. Objects

1) Object :- The most basic non- primitive data

type is the object itself. Objects are collections

of key-value pairs where each key is a string

(or a symbol) and each value can be any valid

JavaScript data type, including other objects.

var person = {

name: 'Arnav',

age: 30,

isStudent: false,

address: {

street: 'Railway Station Road',

city: 'Dehgam'

},

sayHello: function() {

console.log('Hello!');

}

};

2) Array :- Arrays are special types of objects

that store ordered lists of values. Each value

in an array can be of any data type,

including other arrays or objects.

var colors = ['red', 'green', 'blue'];

var matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]];

3) Function :- Function are also objects in

JavaScript. They can be assigned to

variables, passed as arguments, and returned

from other functions.

function greet(name) {

console.log('Hello, ' + name + '!');

}

4) Date :- The ‘Date’ object represtents a

specific point in time.

var currentDate = new Date();

**Q. Write a mul Function Which will Work Properly**

**when invoked With Following Syntax**

🡪 The MUL function is a miniature of the multiplication

function. In this function, we call the function that

required an argument as a first number, and that function

calls another function that required another argument and

this step goes on.

The first function’s argument is x, the second function’s

argument is y and the third is z, so the return value will

be xyz.

Syntax :-

function mul(x) {

return function (y) {

return function (z) {

return x \* y \* z;

};

};

}

Example :-

<script>

function mul(x) {

return function(y) {

return function(z) {

return x\*y\*z;

};

}

}

document.write(mul(2)(3)(5));

document.write(mul(2)(3)(4));

</script>

Output :-

30

24

**Q. What the deference between undefined and**

**undeclare in JavaScript?**

🡪 The difference between undefined and undeclared

variables in JavaScript is : Undefined variable means a

variable has been declared but does not have a value.

Undeclared variable means that the variable does not

exist in the program at all.

Example :-

Undefined

let Arnav;

undefined

document.write(‘Arnav’);

Undeclared

// Error :- myVariable is not defined

document.write(‘Arnav’);

**Q. 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 (“There is no exercise better for the heart

than reaching down and lifting people up.' by John

Holmes teaches us to help one another.");

console.log("Spread love everywhere you go. Let no one

ever come to you without leaving happier. –

Mother Teresa");

**Q. Check if typeof '10' is exactly equal to 10. If not**

**make it exactly equal?**

🡪 The ‘typeof’ operator in JavaScript returns a string

indicating the type of the operand. In this case, ‘typeof

‘10’’ will return the string ‘string’. To check if it is

exactly equal to the number ’10’, you can convert the

string to a number using the ‘parseINT’ function or the

‘Number’ constructor.

In this example, **parseInt(stringValue, 10)** converts the

string **'10'** to the number **10**. The subsequent **if**

statement checks if the result is a number and is exactly

equal to **10**. If not, it updates the value to **10**.

**Q. Write a JavaScript Program to find the area of**

**triangle?**

🡪 area = (base \* height) / 2

Example :-

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Document</title>

</head>

<body>

<script>

// Function to calculate the area of a triangle

function calculateTriangleArea(base, height) {

// Calculate the area using the formula

var area = 0.5 \* base \* height;

return area;

}

// Example usage

var baseLength = 10;

var heightLength = 5;

// Calculate the area of the triangle

var triangleArea = calculateTriangleArea(baseLength, heightLength);

// Display the result

document.write("The area of the triangle is:", triangleArea);

</script>

</body>

</html>

Output :- The area of the triangle is:25

**Q. Write a JavaScript Program to calculate days left**

**until next Christmas?**

🡪

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Document</title>

</head>

<body>

<h3>

Program to calculate days left until

next Christmas using JavaScript?

</h3>

<script>

// Get the current date

let today = new Date();

// Get the year of the current date

let christmasYear = today.getFullYear();

// Check if the current date is

// already past by checking if the month

// is December and the current day

// is greater than 25

if (today.getMonth() == 11 &&

today.getDate() > 25) {

// Add an year so that the next

// Christmas date could be used

christmasYear = christmasYear + 1;

}

// Get the date of the next Christmas

let christmasDate =

new Date(christmasYear, 11, 25);

// Get the number of milliseconds in 1 day

let dayMilliseconds =

1000 \* 60 \* 60 \* 24;

// Get the remaining amount of days

let remainingDays = Math.ceil(

(christmasDate.getTime() - today.getTime()) /

(dayMilliseconds)

);

// Write it to the page

document.write("There are " + remainingDays +

" days remaining until Christmas.");

</script>

</body>

</html>

Output :-

### **Program to calculate days left until next Christmas using JavaScript?**

There are 343 days remaining until Christmas.

**Q. What is Condition Statement?**

🡪 A conditional statement in programming is a construct

that allows the execution of different code blocks based

on whether a specified condition evaluates to true or

false. It enables a program to make decisions and take

different actions depending on the given conditions. The

primary types of conditional statements in most

programming languages include:

1) if Statement :- The **if** statement is the most basic

conditional statement. It evaluates a specified

condition and, if the condition is true, executes a block

of code.

var x = 10;

if (x > 5) {

console.log('x is greater than 5');

}

2) if-else Statement :- The **if-else** statement extends

the **if** statement. It executes one block of code if the

condition is true and another block if the condition

is false.

var y = 3;

if (y > 5) {

console.log('y is greater than 5');

} else {

console.log('y is not greater than 5');

}

3) Switch Statement :- The **switch** statement is used

when there are multiple possible values for a

variable. It provides a way to compare a variable

against multiple values and execute different

blocks of code accordingly.

**Q. Find circumference of Rectangle formula : C = 4 \***

**a ?**

🡪 The formula you've mentioned, C = 4 \* a seems to be

more applicable to a square than a rectangle. The

formula for the circumference of a rectangle involves

the lengths of its sides. For a rectangle with sides a and

b, the circumference (or perimeter) is given by:

C = 2 X (a+b)

**Q. WAP to convert years into days and days into**

**years?**

🡪

<!DOCTYPE html>

<html>

<head>

<title>Year to Days and Days to Years Converter</title>

</head>

<body>

<script>

function yearsToDays(years) {

// Assuming a standard year has 365 days

return years \* 365;

}

function daysToYears(days) {

// Assuming a standard year has 365 days

return days / 365;

}

// Example usage:

var inputYears = 2;

var inputDays = 730; // 2 years \* 365 days

// Convert years to days

var resultDays = yearsToDays(inputYears);

document.write(inputYears + " years is equal to " + resultDays + " days<br>");

// Convert days to years

var resultYears = daysToYears(inputDays);

document.write(inputDays + " days is equal to " + resultYears + " years");

</script>

</body>

</html>

Output :-

2 years is equal to 730 days  
 730 days is equal to 2 years

**Q. Convert temperature Fahrenheit to Celsius?**

🡪

<!DOCTYPE html>

<html>

<head>

<title>Fahrenheit to Celsius Converter</title>

</head>

<body>

<script>

function fahrenheitToCelsius(fahrenheit) {

return (5 / 9) \* (fahrenheit - 32);

}

// Example usage:

var inputFahrenheit = 98.6; // Body temperature in Fahrenheit

// Convert Fahrenheit to Celsius

var resultCelsius = fahrenheitToCelsius(inputFahrenheit);

document.write(inputFahrenheit + " degrees Fahrenheit is equal to " + resultCelsius.toFixed(2) + " degrees Celsius");

</script>

</body>

</html>

Output :-

98.6 degrees Fahrenheit is equal to 37.00 degrees Celsius

**Q. Write a JavaScript exercise to get the extension of**

**a filename?**

🡪 To get file extension using JavaScript, there are so

many ways.

1. split() and pop() method

2. substring() and lastIndexOf() method

3. match() method

1. Using split() and pop() method

🡪 The full filename is first obtained by selecting the

file input and getting its value property. This returns

the filename as a string.

🡪 By the help of the split() method, we will split the

filename into 2 parts. The first part will be the

filename and the second part will be the extension of

the file.

🡪 The extension can then be got by popping from the

array the last string with the pop()method. This is

hence the file extension of the file selected.

Syntax

filename.split(separator,limit).pop();

Example :-

<!DOCTYPE html>

<html>

<head>

<title>How to get file extensions using JavaScript? </title>

</head>

<body style="text-align: center;">

<h2>Here we will get "Extension" of selected file</h2>

<p>Select a file and click on the button

to check the file extension.</p>

<form>

<input type="file" id="file1" />

<input type="button" value="Check Extension"

onclick="checkFileExtension();"/>

</form>

<p>The file extension is: <span class="output"></span></p>

<script language="javascript">

function checkFileExtension() {

fileName = document.querySelector('#file1').value;

extension = fileName.split('.').pop();

document.querySelector('.output')

.textContent = extension;

};

</script>

</body>

</html>

Output:-

A screenshot of a computer

Description automatically generated

**Q. What is the result of the expression (5 > 3 && 2 <**

**4)?**

🡪 The result of the expression (5 > 3 && 2 < 4) is true.

This is because the && (logical AND) operator returns

‘true’ if both of its operands are true; otherwise, it

returns ‘false’.

In this case :

The expression 5 > 3 evaluates to ‘true’

The expression 2 > 4 evaluates to ‘true’

Therefore, when you combine them using the **&&**

operator, the overall result is **true**. The entire

expression can be read as "5 is greater than 3, and 2

is less than 4," which is logically true.

**Q. What is the result of the expression (true && 1**

**&& "hello")?**

🡪 The result of the expression (true && 1 &&

"hello") is “hello”.

In JavaScript, the **&&** (logical AND) operator

returns the value of the last operand if all operands

are truthy. Here's how the expression is evaluated:

‘true’ is truthy.

‘1’ is truthy.

“hello” is truthy.

🡪 Since all operands are truthy, the result of the **&&**

operation is the last truthy operand, which is

**"hello"**. Therefore, the overall result of the

expression is **"hello"**.

**Q. What is the result of the expression true && false ||**

**false && true?**

🡪 The result of the expression **true && false ||**

**false && true** is ‘false’.

The expression is evaluated based on the order of

precedence of logical AND (**&&**) and logical OR (**||**)

operators. The logical AND operator has higher

precedence than the logical OR operator.

1. ‘true && false’ - is false because the logical AND of

‘true’ and ‘false’ is false.

2. ‘ false || false && true’ - At this point, we have 'false ||

false && true’.

3. ‘false && true’ is false – because the logical AND of

‘false’ and ‘true’ is false.

4. ‘false || false’ is false because the logical OR of two

‘false’ value is ‘false’.

So, the overall result of the expression is ‘false’.

**Q. What is a Loop and Switch Case in JavaScript define**

**that ?**

🡪 In JavaScript, loops and switch cases are control flow

structures that allow you to execute a block of code

repeatedly or make decisions based on different

conditions.

Loops:

A loop is a programming construct that repeatedly

executes a block of code until a specified condition is

false. JavaScript supports several types of loops.

1. for Loop :- The ‘for’ loop is used when you know

the number of iterations in advance.

for (var i = 0; i < 5; i++) {

console.log(i);

}

2. While Loop :- The ‘while’ loop is used when the

number of iterations is not known in advance, and

the loop continues as long as a specified condition

is true.

var i = 0;

while (i < 5) {

console.log(i);

i++;

}

3. do-while Loop :- Similar to the ‘while ’loop, but

the condition is checked after the loop body is

executed, ensuring that body is executed at least

once.

var i = 0;

do {

console.log(i);

i++;

} while (i < 5);

4. for... in Loop :- Iterates over the properties of an

object.

var person = { name: 'John', age: 30 };

for (var key in person) {

console.log(key + ': ' + person[key]);

}

5. for ...of Loop :- Iterates over iterable objects like

arrays and strings.

var colors = ['red', 'green', 'blue'];

for (var color of colors) {

console.log(color);

}

Switch Case :- The ‘switch’ statement is used to

perform different actions based on different conditions.

It provides a more readable and compact way to write

multiple ‘if..else’ statements.

var day = 'Monday';

switch (day) {

case 'Monday':

console.log('It\'s the start of the week.');

break;

case 'Friday':

console.log('It\'s almost the weekend!');

break;

default:

console.log('It\'s a regular day.');

}

In this example, the **switch** statement checks the value

of the variable **day** and executes the corresponding block of

code based on the matching **case**. The **break** statement is

used to exit the **switch** statement after a match.

**Q. What is the use of NaN function?**

🡪 The ‘NaN’ function in JavaScript is used to determine

whether a given value is NaN (Not-a-Number). It returns

a boolean indicating whether the provided value is NaN

or can be converted to a NaN.

Syntax – isNaN(value)

1. If the provided value is NaN, it returns ‘true’

2. If the provided value is a non-number type (e.g., a

string, object, boolean, etc.), or a string that cannot be

converted to a number, it returns **true**

3. If the provided value is a number (including positive

or negative infinity), it returns ‘false’.

**Q. What is the difference between && and || in**

**JavaScript?**

🡪 In JavaScript ‘&&’ (logical AND) and ‘||’ (logical OR)

are logical operators used for combining or comparing

boolean values. Here’s the difference between them.

1. Logical AND (‘&&’)

🡪 The ‘&&’ operators ‘true’ if both operands are true,

otherwise, it returns ‘false’

🡪 It is a short-circuiting operator. If the first operand is

**false**, the second operand is not evaluated because the

overall result will be **false** regardless of the second

operand.

true && true; // true

true && false; // false

false && true; // false (short-circuiting, the second

operand is not evaluated)

false && false; // false

2. Logical OR (‘||’)

🡪 The ‘||’ operators returns ‘true’ if at least one of the

operands is ‘true’, if both operands are false, it

returns ‘false’.

🡪 Like **&&**, it is a short-circuiting operator. If the first

operand is **true**, the second operand is not evaluated

because the overall result will be **true** regardless of

the second operand.

true || true; // true (short-circuiting, the second

operand is not evaluated)

true || false; // true

false || true; // true

false || false; // false

**Q. What is the use of void(0)?**

🡪 The use of ‘void(0)’ in JavaScript is often seen in the

context of using ‘href’ attribute in HTML anchor (‘<a>’)

tags to create “javascript:void(0)” links. This is a

common pattern to prevent the browser from navigating

to a new page when the link is clicked.

**<a href="javascript:void(0);"onclick="myFunction()">Click me</a>.**

1. Javascript : protocol

By using ‘javscript’ in the ‘href’ attribute, you are

essentially executing JavaScript code. However,

‘void(0)’ is often used as a harmless expression that

evaluates to ‘undefined’ in this context.

2. Preventing Default Action

When you have an **onclick** event attached to the link,

using **javascript:void(0);** prevents the default action of

the link (which is navigating to a new page). In this

case, the **onclick** event handler (**myFunction()** in the

example) is responsible for the action.

3. Alternative to ‘#’

**javascript:void(0);** is often used as an alternative to **#**

in the **href** attribute. Both are used to create clickable

links that don't navigate anywhere, but

**javascript:void(0);** is considered more semantically

correct.

**Q. Check Number is Positive or Negative in JavaScript?**

🡪 To check whether a number is positive or negative in

JavaScript, you can use a simple conditional statement (an

**if** statement). Here's an example function that checks the

sign of a number.

function checkSign(number) {

if (number > 0) {

return "Positive";

} else if (number < 0) {

return "Negative";

} else {

return "Zero";

}

}

🡪 If the ‘number’ is greater than ‘0’, the function returns

“Positive”.

🡪 If the ‘number’ is less than ‘0’, the function returns

“Negative”.

🡪 If the ‘number’ equal to ‘0’, the function returns

“Zero”.

**Q. Find the Character is Vowel or Not?**

🡪 To determine whether a character is a vowel or not in

JavaScript, you can create a function that checks if the

character is one of the vowels (a,e,i,o,u).

function isVowel(character) {

// Convert the character to lowercase to handle both

upper and lowercase vowels

var lowerCaseChar = character.toLowerCase();

// Check if the character is a vowel

if (lowerCaseChar === 'a' || lowerCaseChar === 'e' ||

lowerCaseChar === 'i' || lowerCaseChar === 'o' ||

lowerCaseChar === 'u') {

return true; // It's a vowel

} else {

return false; // It's not a vowel

}

}

**Q. Write to check whether a number is negative, positive**

**or zero?**

🡪 You can create a function in JavaScript to check whether

A number is negative, positive or zero.

function checkNumber(number) {

if (number > 0) {

return "Positive";

} else if (number < 0) {

return "Negative";

} else {

return "Zero";

}

}

// Example usage:

console.log(checkNumber(5)); // Output: Positive

console.log(checkNumber(-3)); // Output: Negative

console.log(checkNumber(0)); // Output: Zero

🡪 If the ‘number’ is greater than ‘0’, the function returns

“Positive”.

🡪 If the ‘number’ is less than ‘0’, the function returns

“Negative”.

🡪 If the ‘number’ equal to ‘0’, the function returns

“Zero”.

**Q. Write to find number is even or odd using ternary**

**operator in JS?**

🡪 You can use the ternary operator in JavaScript to determine

whether a number is even or odd.

function checkEvenOdd(number) {

var result = number % 2 === 0 ? "Even" : "Odd";

return result;

}

// Example usage:

console.log(checkEvenOdd(4)); // Output: Even

console.log(checkEvenOdd(7)); // Output: Odd

console.log(checkEvenOdd(0)); // Output: Even (0 is

considered even)

In this example, the ternary operator (‘condition?

expressionIfTrue : expressionIfFalse’) is used to check if

the number is even or odd. If the condition ‘number % 2

=== 0’ Is true (including that number is divisible by 2),

the result is “Even”, otherwise, the result is “Odd”.

**Q. Write find maximum number among 3 numbers**

**using ternary operator in JS?**

🡪 You can use the ternary operator in JavaScript to find

the maximum number among three numbers.

function findMaxNumber(a, b, c) {

var max = (a > b) ? ((a > c) ? a : c) : ((b > c) ? b : c);

return max;

}

// Example usage:

console.log(findMaxNumber(5, 12, 8)); // Output: 12

console.log(findMaxNumber(-3, 0, 7)); // Output: 7

console.log(findMaxNumber(10, 5, 15)); // Output: 15

**Q. Write find minimum number among 3 numbers**

**using ternary operator in JS?**

🡪 You can use the ternary operator in JavaScript to find

the minimum number among three numbers.

function findMaxNumber(a, b, c) {

var max = (a < b) ? ((a < c) ? a : c) : ((b < c) ? b : c);

return max;

}

// Example usage:

console.log(findMaxNumber(5, 12, 8)); // Output: 12

console.log(findMaxNumber(-3, 0, 7)); // Output: 7

console.log(findMaxNumber(10, 5, 15)); // Output: 15

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

🡪 You can write a JavaScript function to find the largest

of three numbers using conditional statements or the

Math.max.function.

function findLargest(a, b, c) {

if (a >= b && a >= c) {

return a;

} else if (b >= a && b >= c) {

return b;

} else {

return c;

}

}

// Example usage:

console.log(findLargest(5, 12, 8)); // Output: 12

console.log(findLargest(-3, 0, 7)); // Output: 7

console.log(findLargest(10, 5, 15)); // Output: 15

Using Math.max

function findLargest(a, b, c) {

return Math.max(a, b, c);

}

// Example usage:

console.log(findLargest(5, 12, 8)); // Output: 12

console.log(findLargest(-3, 0, 7)); // Output: 7

console.log(findLargest(10, 5, 15)); // Output: 15

In this ‘Math.max’ function is used to find the maximum

value among the three numbers.

**Q. Write to show**

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

🡪 <!DOCTYPE html>

<html>

<head>

<title>Day of the Week</title>

</head>

<body>

<script>

function getDayOfWeek(dayNumber) {

switch (dayNumber) {

case 1:

return "Monday";

case 2:

return "Tuesday";

case 3:

return "Wednesday";

case 4:

return "Thursday";

case 5:

return "Friday";

case 6:

return "Saturday";

case 7:

return "Sunday";

default:

return "Invalid day number";

}

}

// Example usage:

var today = new Date().getDay(); // Get the current day of the week (1 to 7, where 1 is Sunday)

document.write("Today is " + getDayOfWeek(today));

</script>

</body>

</html>

Output :-

Today is Wednesday

**2. Vowel or Consonant using switch case in JS?**

🡪 function checkVowelOrConsonant(character) {

switch (character.toLowerCase()) {

case 'a':

case 'e':

case 'i':

case 'o':

case 'u':

return 'Vowel';

default:

return 'Consonant';

}

}

// Example usage:

console.log(checkVowelOrConsonant('a')); // Output:

Vowel

console.log(checkVowelOrConsonant('b')); // Output:

Consonant

console.log(checkVowelOrConsonant('E')); // Output:

Vowel

console.log(checkVowelOrConsonant('z')); // Output:

Consonant

**(Conditional looping logic Question)**

**Q. What are the looping structure in JavaScript? Any**

**one Example?**

🡪

1. for Loop :- The ‘for’ loop is used when you know

the number of iterations in advance.

for (var i = 0; i < 5; i++) {

console.log(i);

}

2. While Loop :- The ‘while’ loop is used when the

number of iterations is not known in advance, and

the loop continues as long as a specified condition

is true.

var i = 0;

while (i < 5) {

console.log(i);

i++;

}

3. do-while Loop :- Similar to the ‘while ’loop, but

the condition is checked after the loop body is

executed, ensuring that body is executed at least

once.

var i = 0;

do {

console.log(i);

i++;

} while (i < 5);

4. for... in Loop :- Iterates over the properties of an

object.

var person = { name: 'John', age: 30 };

for (var key in person) {

console.log(key + ': ' + person[key]);

}

5. for ...of Loop :- Iterates over iterable objects like

arrays and strings.

var colors = ['red', 'green', 'blue'];

for (var color of colors) {

console.log(color);

}

Example :-

for (var i = 1; i <= 5; i++) {

console.log(i);

}

Output :-

1

2

3

4

5

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

🡪 If you want to print numbers from 972 to 897 using a ‘for’

loop in JavaScript, you can set up a loop that starts with the

initial value (972), decrements by 1 in each iteration, and

continues until it reaches the final value(897).

for (var i = 972; i >= 897; i--) {

console.log(i);

}

This loop initializes **i** to 972, executes the loop body as

long as **i** is greater than or equal to 897, decrements **i** by 1

after each iteration, and prints the value of **i** in each

iteration. The output will be the numbers from 972 down

to 897.

**Q. Write to print factorial of given number?**

🡪 To calculate the factorial of a given number using a ‘for’

loop in JavaScript, you can create a function as follows.

function calculateFactorial(number) {

if (number < 0) {

return "Undefined (Factorial is not defined for

negative numbers)";

} else if (number === 0 || number === 1) {

return 1;

} else {

var factorial = 1;

for (var i = 2; i <= number; i++) {

factorial \*= i;

}

return factorial;

}

}

// Example usage:

var result = calculateFactorial(5);

console.log("Factorial of 5 is:", result);

In this example

🡪 The ‘calculateFactorial’ function takes a number as an

Argument.

🡪 It checks if the number is negative, in which case it

returns an error message.

🡪 If the number is 0 and 1, the function returns 1 (base

case for the factorial).

🡪 Otherwise, a **for** loop is used to calculate the factorial

by multiplying numbers from 2 to the given number.

**Q. Write to print Fibonacci series up to given**

**numbers?**

🡪

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Document</title>

</head>

<body>

<script>

function factorial(n) {

if (n < 0) {

return "Invalid input. Please provide a non-negative integer.";

}

if (n === 0 || n === 1) {

return 1;

}

let result = 1;

for (let i = 2; i <= n; i++) {

result \*= i;

}

return result;

}

// Example usage:

let number = 5;

document.write(`The factorial of ${number} is: ${factorial(number)}`);

</script>

</body>

</html>

Output :-

The factorial of 5 is:120

**Q. Write to print number in reverse order e.g number**

**= 64728-🡪 reverse = 82746 in JS?**

🡪

<script>

function reverseNumber(number) {

const numberString = number.toString();

const reversedString =

numberString.split('').reverse().join('');

const reversedNumber = parseInt(reversedString, 10);

return reversedNumber;

}

// Example usage:

const originalNumber = 64728;

const reversedNumber = reverseNumber(originalNumber);

</script>

**Q. Write a program make a summation of given**

**number (E.g. 1523 Ans: - 11) in JS?**

🡪

<script>

function calculateDigitSum(number) {

const numberString = Math.abs(number).toString();

let sum = 0;

for (let i = 0; i < numberString.length; i++) {

sum += parseInt(numberString[i], 10);

}

return sum;

}

// Example usage:

const givenNumber = 1523;

const digitSum = calculateDigitSum(givenNumber);

</script>

**Q. Write a program you have to make a summation of**

**first and last Digit. (E.g. 1234 Ans:- 5) in JS?**

🡪

<script>

function sumOfFirstAndLastDigit(number) {

const numberString = Math.abs(number).toString();

const firstDigit = parseInt(numberString[0], 10);

const lastDigit =

parseInt(numberString[numberString.length - 1], 10);

const sum = firstDigit + lastDigit;

return sum;

}

// Example usage:

const givenNumber = 1234;

const result = sumOfFirstAndLastDigit(givenNumber);

</script>

**Q. 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

<script>

for (let i = 1; i <= 5; i++) {

let row = '';

for (let j = 1; j <= 5; j++) {

if (j === 1) {

row += i;

} else {

row += ' ' + Math.pow(i, j);

}

}

console.log(row);

}

</script>

**Q. Use pattern in console.log in JS?**

🡪

1) 1

1 0

1 0 1

1 0 1 0

1 0 1 0 1

<script>

for (let i = 1; i <= 5; i++) {

let row = '';

for (let j = 1; j <= i; j++) {

row += (j % 2 === 1) ? '1' : '0';

if (j < i) {

row += ' ';

}

}

console.log(row);

}

</script>

2) A

B C

D E F

G H I J

K L M N O

🡪

<script>

let currentCharCode = 65;

for (let i = 1; i <= 5; i++) {

let row = '';

for (let j = 1; j <= i; j++) {

row += String.fromCharCode(currentCharCode) + ' ';

currentCharCode++;

}

console.log(row);

}

</script>

3)

1

2 3

4 5 6

7 8 9 10

11 12 13 14 15

🡪

<script>

let currentNumber = 1;

for (var i = 1; i <= 5; i++) {

for (var j = 1; j <= i; j++) {

document.write(" " + currentNumber);

currentNumber++;

}

document.write("</br>");

}

</script>

4)

\*

\* \*

\* \* \*

\* \* \* \*

\* \* \* \* \*

🡪

<script>

for (var i = 1; i <= 5; i++)

{

for(var j=1;j<=i;j++)

{

document.write("\*");

}

document.write("</br>");

}

</script>

**(Array and object Question)**

**Q. Write a JavaScript Program to display the current**

**day and time in the following format.**

**Sample Output: Today is Friday. Current Time is**

**12PM: 12:22 2?**

🡪

<!DOCTYPE html>

<html>

<head>

<title>

print current day and time

</title>

</head>

<body>

<script type="text/javascript">

var myDate = new Date();

var myDay = myDate.getDay();

// Array of days.

var weekday = ['Sunday', 'Monday', 'Tuesday',

'Wednesday', 'Thursday', 'Friday', 'Saturday'

];

document.write("Today is : " + weekday[myDay]);

document.write("<br/>");

// get hour value.

var hours = myDate.getHours();

var ampm = hours >= 12 ? 'PM' : 'AM';

hours = hours % 12;

hours = hours ? hours : 12;

var minutes = myDate.getMinutes();

minutes = minutes < 10 ? '0' + minutes : minutes;

var myTime = hours + " " + ampm + " : " + minutes +

" : " + myDate.getMilliseconds();

document.write("\tCurrent time is : " + myTime);

</script>

</body>

</html>

Output :-

Today is : Wednesday  
 Current time is : 4 PM : 46 : 74

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

🡪

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Document</title>

</head>

<body>

<script>

let date = new Date().toDateString();

document.write(date);

</script>

</body>

</html>

Output :-

Wed Jan 17 2024

**Q. Write a JavaScript program to compare two**

**objects?**

🡪

<!DOCTYPE html>

<html>

<head>

<title>Object Comparison</title>

</head>

<body>

<script>

function areObjectsEqual(obj1, obj2) {

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 each key and its value are the same

for (let key of keys1) {

if (obj1[key] !== obj2[key]) {

return false;

}

}

return true;

}

// Example usage:

const object1 = { a: 1, b: 2, c: 3 };

const object2 = { a: 1, b: 2, c: 3 };

const result = areObjectsEqual(object1, object2);

document.write("Are objects equal? " + result);

</script>

</body>

</html>

Output :-

Are objects equal? true

**Q. Write a JavaScript program to convert an array of**

**objects into CSV string?**

🡪

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Document</title>

</head>

<body>

<script>

function convertArrayToCSV(data) {

const header = Object.keys(data[0]).join(',') + '\n';

const csvRows = data.map(row => Object.values(row).join(',') + '\n');

return header + csvRows.join('');

}

// Example usage:

const dataArray = [

{ name: 'John', age: 30, city: 'New York' },

{ name: 'Alice', age: 25, city: 'San Francisco' },

{ name: 'Bob', age: 35, city: 'Seattle' }

];

const csvString = convertArrayToCSV(dataArray);

document.write(csvString);

</script>

</body>

</html>

Output :-

name,age,city John,30,New York Alice,25,San Francisco Bob,35,Seattle

**Q. Write a JavaScript to capitalize first letter of a**

**string?**

🡪

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Document</title>

</head>

<body>

<script>

function capitalizeFirstLetter(str) {

return str.charAt(0).toUpperCase() + str.slice(1);

}

// Example usage:

const inputString = "example string";

const capitalizedString = capitalizeFirstLetter(inputString);

document.write(`Original String: ${inputString}`);

document.write(`Capitalized String: ${capitalizedString}`);

</script>

</body>

</html>

Output :-

Original String: example stringCapitalized String: Example string

**Q. Write a JavaScript program to determine if a**

**variable is array?**

🡪 You can use the Array.isArray() method in JavaScript to

determine if a variable is an array.

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Document</title>

</head>

<body>

<script>

function isArray(variable) {

return Array.isArray(variable);

}

const exampleArray = [1, 2, 3];

const exampleString = "Not an array";

document.write(isArray(exampleArray)); // Output: true

document.write(isArray(exampleString)); // Output: false

</script>

</body>

</html>

Output :- truefalse

The Array.isArray(variable) method returns ‘true’ if the variable is an array and ‘false’ otherwise.

The isArray function is a simple wrapper around ‘Array.isArray()’.

**Q. write a JavaScript program to clone an array?**

🡪

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Document</title>

</head>

<body>

<script>

function cloneArray(originalArray) {

return originalArray.slice();

}

const originalArray = [1, 2, 3, 4, 5];

const clonedArray = cloneArray(originalArray);

document.write("Original Array:", originalArray);

document.write("Cloned Array:", clonedArray);

</script>

</body>

</html>

Output:-

Original Array:1,2,3,4,5Cloned Array:1,2,3,4,5

**Q. Print the length of the string on the browser console**

**using console.log()?**

🡪

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>String Length Example</title>

</head>

<body>

<script>

const myString = "Hello, World!";

console.log("Length of the string:", myString.length);

</script>

</body>

</html>

Output :-

The length of the string is :13

**Q. Change all the string characters to capital letters**

**using to UpperCase() method?**

🡪

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>String Length Example</title>

</head>

<body>

<script>

const originalString = "Hello, World!";

const capitalizedString = originalString.toUpperCase();

document.write("Original String:", originalString);

document.write("Capitalized String:", capitalizedString);

</script>

</body>

</html>

Output :-

Original String:Hello, World!Capitalized String:HELLO, WORLD!

**Q. What is the drawback of declaring methods directly**

**in JavaScript objects?**

🡪 Declaring methods directly in JavaScript objects,

especially in the context of constructor functions or

object literals, has some drawbacks

1. Memory Consumption:- When you declare methods

directly in the object, each instance of the object will have

its own copy of the methods. This can lead to increased

memory consumption, especially if you have many

instances of the object. If the methods don't rely on

instance-specific data, it's more memory-efficient to

attach them to the prototype.

2. Inefficiency for Repeatedly Created Objects:- If you

create multiple instances of an object with methods

declared directly in the object, JavaScript has to allocate

memory for the methods every time a new instance is

created. This can be less efficient compared to attaching

methods to the prototype, where the methods are shared

among all instances.

3. Difficulty in Achieving Method Overriding :- When

methods are declared directly in an object, it can be

challenging to achieve method overriding or

inheritance, especially if you want to extend the

functionality of an existing method. Prototypal

inheritance provides a more flexible way to handle

method overriding.

4. Readability and Maintainability :- As your object

grows, declaring methods directly in the object may

lead to code that is less readable and maintainable.

Separating the definition of methods from the object

creation can enhance code organization and readability.

**Q. Write a JavaScript program to get the current date.**

**Expected Output : mm-dd-yyyy,**

**mm/dd/yyyy or dd-mm-yyyy, dd/mm/yyyy?**

🡪

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Document</title>

</head>

<body>

<script>

function getCurrentDate(format) {

const currentDate = new Date();

let day = currentDate.getDate();

let month = currentDate.getMonth() + 1;

const year = currentDate.getFullYear();

day = (day < 10) ? '0' + day : day;

month = (month < 10) ? '0' + month : month;

switch (format) {

case 'mm-dd-yyyy':

return `${month}-${day}-${year}`;

case 'mm/dd/yyyy':

return `${month}/${day}/${year}`;

case 'dd-mm-yyyy':

return `${day}-${month}-${year}`;

case 'dd/mm/yyyy':

return `${day}/${month}/${year}`;

default:

return 'Invalid format';

}

}

const a = getCurrentDate('mm-dd-yyyy');

const b = getCurrentDate('mm/dd/yyyy');

const c = getCurrentDate('dd-mm-yyyy');

const d = getCurrentDate('dd/mm/yyyy');

document.write('mm-dd-yyyy:', a + "<br>");

document.write('mm/dd/yyyy:', b + "<br>");

document.write('dd-mm-yyyy:', c + "<br>");

document.write('dd/mm/yyyy:', d + "<br>");

</script>

</body>

</html>

Output :-

mm-dd-yyyy:01-25-2024  
 mm/dd/yyyy:01/25/2024  
 dd-mm-yyyy:25-01-2024  
 dd/mm/yyyy:25/01/2024

**Q. Use indexOf to determine the position of the first**

**occurrence of a in 30 Days Of JavaScript?**

🡪 If you want to use the ‘IndexOf’ method to determine

the position of the first occurrence of a specific substring

in a string, you can do so as follow.

Example :-

<!DOCTYPE html>

<html>

<body>

<h1>JavaScript Strings</h1>

<h2>The IndexOf() Method</h2>

<p id= “demo”> </p>

<script>

let text = "30 Days of JavaScript";

let result = text.IndexOf("a");

document.getElementById("demo").innerHTML =

result;

</script>

</body>

</html>

Output :-

# **JavaScript Strings**

**The IndexOf() Method**

3

🡪 In this example, the ‘IndexOf’ method is used on the

‘text’ variable to find the position of the first

occurrence of the substring “a”.

**Q. Use lastIndexOf to determine the position of the last**

**occurrence of a in 30 Days Of JavaScript?**

🡪 If you want to use the ‘lastIndexOf’ method to

determine the position of the last occurrence of a

specific substring in a string, you can do so as follows.

Example :-

<!DOCTYPE html>

<html>

<body>

<h1>JavaScript Strings</h1>

<h2>The lastIndexOf() Method</h2>

<p id= “demo”> </p>

<script>

let text = "30 Days of JavaScript";

let result = text.lastIndexOf("a");

document.getElementById("demo").innerHTML =

result;

</script>

</body>

</html>

Output :-

# **JavaScript Strings**

**The lastIndexOf() Method**

14

**Q. Form Validation in JS?**

**Q. Form in Email, number, Password, Validation?**

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

**🡪**

**HTML File**

<!DOCTYPE html>

<html lang="en">

<head>

<title>Bootstrap Example</title>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css" rel="stylesheet">

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/js/bootstrap.bundle.min.js"></script>

<script src="validation.js"> </script>

<style>

span {

color: red;

font-size: 1.5vw;

margin-left: 40vw;

}

</style>

</head>

<body>

<span id="Error"> </span>

<div class="container mt-3">

<h2>Registration Form</h2>

<form action="" name="myform" onsubmit="return Formvalidation()">

<div class="mb-3 mt-3">

<label for="Username">Username:</label>

<input type="text" class="form-control" id="un" placeholder="Enter your Name" name="Enter your Name">

</div>

<div class="mb-3 mt-3">

<label for="email">Email:</label>

<input type="email" class="form-control" id="email" placeholder="Enter your email" name="email">

</div>

<div class="mb-3">

<label for="pwd">Password:</label>

<input type="password" class="form-control" id="pwd" placeholder="Enter your password" name="pswd">

</div>

<div class="mb-3">

<label for="pwd">Retype Password:</label>

<input type="password" class="form-control" id="repwd" placeholder="Enter Re-Password" name="repswd">

</div>

<div class="mb-3">

<label for="Mobile Number">Mobile Number:</label>

<input type="number" class="form-control" id="mnumber" placeholder="Enter your Mobile Number"

name="MobileNumber">

</div>

<p> Gender:</p>

<div class="form-check">

<input type="radio" class="form-check-input" id="" name="gender" value="M">

<label class="form-check-label" for="radio1">Male</label>

</div>

<div class="form-check mb-3">

<input type="radio" class="form-check-input" id="" name="gender" value="F">

<label class="form-check-label" for="radio2">Female</label>

</div>

<p> Select Country:</p>

<select class="form-select mb-3" id="drop">

<option>Select Country</option>

<option>CANADA</option>

<option>INDIA</option>

<option>USA </option>

<option>UNITED KINGDOM</option>

</select>

<p> Select Hobbies:</p>

<div class="form-check">

<input type="checkbox" class="form-check-input" id="" name="chk" value="something">

<label class="form-check-label" for="check2">Outdoor Games</label>

</div>

<div class="form-check">

<input type="checkbox" class="form-check-input" id="" name="chk" value="something">

<label class="form-check-label" for="check2">Reading</label>

</div>

<div class="form-check mb-3">

<input type="checkbox" class="form-check-input" id="" name="chk" value="something">

<label class="form-check-label" for="check2">Indoor Games</label>

</div>

<button type="submit" class="btn btn-primary">Submit</button>

</div>

</form>

</body>

</html>

**JS File :-**

function Formvalidation() {

var un = document.forms["myform"]["un"].value;

if (un == "" || un == "null") {

document.getElementById('Error').innerHTML = "Please Fill Out the User Name";

return false;

}

var email = document.forms["myform"]["email"].value;

if (email == "" || email == "null") {

document.getElementById('Error').innerHTML = "Please Enter Your Email";

return false;

}

var mail = /^[a-zA-Z0-9\_]+@[a-zA-Z]+\.[a-zA-Z]{2,4}$/;

if (!mail.test(email)) {

document.getElementById('Error').innerHTML = "Please Enter Proper Email";

return false;

}

var pass = document.forms["myform"]["pwd"].value;

if (pass == "" || pass == "null") {

document.getElementById('Error').innerHTML = "Please Enter Your Password";

return false;

}

if (!(pass.length >= 3 && pass.length <= 8)) {

document.getElementById('Error').innerHTML = "Please, Enter min 3 & max 8 Char in Pass";

return false;

}

var cpass = document.forms["myform"]["repwd"].value;

if (cpass == "" || cpass == "null") {

document.getElementById('Error').innerHTML = "Please Enter Your Retype Password";

return false;

}

if (pass == cpass) {

}

else {

document.getElementById('Error').innerHTML = "Please Enter The Same Value";

return false;

}

var mno = document.forms["myform"]["mnumber"].value;

if (mno == "" || mno == "null") {

document.getElementById('Error').innerHTML = "Please Enter Your Mobile Number";

return false;

}

var mobile = /^[0-9]{10,11}$/;

if (!mobile.test(mno)) {

document.getElementById('Error').innerHTML = "Please Enter Valid Number";

return false;

}

var gender = document.getElementsByName("gender");

if (gender[0].checked == true) {

}

else if (gender[1].checked == true) {

}

else {

document.getElementById('Error').innerHTML = "Please Select Gender";

return false;

}

var scountry = document.forms["myform"]["drop"].value;

if (scountry === " ") {

document.getElementById('Error').innerHTML = "Please Select Country";

return false;

}

var Hobbies = document.getElementsByName("chk");

if (Hobbies[0].checked == true) {

}

else if (Hobbies[1].checked == true) {

}

else if (Hobbies[2].checked == true) {

}

else {

document.getElementById('Error').innerHTML = "Please Select Hobby";

return false;

}

}

**Q. How many type of Js Event? How to use it?**

🡪 HTML events are “things” that happen to HTML

elements.

When JavaScript is used in HTML pages,

JavaScript can “react” on these events.

HTML Events

An HTML event can be something the browser

does, or something a user does.

Here are some examples of HTML events

1. An HTML web page has finished loading

2. An HTML input filed was changed

3. An HTML button was clicked

HTML allows event handler attributes, with

JavaScript code, to be added to HTML elements.

<element event= “some JavaScript”>

**HTML Events**

|  |  |
| --- | --- |
| Event | Description |
| onchange | AHTML element has been  changed |
| onclick | The user clicks an HTML element |
| onmouseover | The user moves the mouse away from an HTML element |
| onmouseout | The user moves the mouse away from an HTML element |
| onkeydown | The user pushes a keyboard key |
| onload | The browse has finished loading the page |

Example :-

<!DOCTYPE html>

<html>

<body>

<h1>JavaScript HTML Events</h1>

<h2>The onclick Attribute</h2>

<button

onclick="document.getElementById('demo').

innerHTML =Date()">The time is?

</button>

<p id="demo"></p>

</body>

</html>

Output





**Q. What is Bom vs Dom in JS?**

🡪 BOM (Browser Object Model) and DOM (Document Object

Model) are two distinct models in JavaScript that provide

interfaces for interacting with different aspects of web

browsers.

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

🡪 The DOM represents the structure of an HTML or XML

document as a tree-like structure where each node

corresponds to a part of the document (e.g. elements,

attributes, text).

🡪 It Provides a programming interface that allows scripts to

dynamically access, manipulate, and update the content,

structure, and style of a document.

🡪 The DOM is primarily concerned with the document’s

structure and content.

🡪 Example of DOM interfaces include ‘document’, ‘Element’.

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

🡪 The BOM represents the browser itself and provides

interfaces for interacting with browser-specific features,

such as the window, history, location, navigator, and screen.

🡪 It allows scripts to control browser behavior, manipulate the

browser window, and gather information about the browser

and user environment.

🡪 The BOM is concerned with the browser and its -

functionalities rather than the document structure

🡪 Examples of BOM interfaces include ‘window’,

‘navigator’, ‘location’, etc.

In summary, while the DOM is focused on representing and manipulating the structure and content of the document, the BOM

deals with browser-specific features and interactions. Both work together to enable dynamic and interactive web applications.

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

🡪 In JavaScript, arrays and objects serve different purposes and

have distinct characteristics. Here are some key differences

between arrays and objects:

🡪 In JavaScript, arrays use numbered indexes. And objects use

named indexes.

🡪 Arrays are used for ordered collections of data, while objects are

used for key-value pairs and representing more complex data

structures.

🡪 In JavaScript, an object is a collection of properties, and a

property is an association between a name (or key) and a value.

Example of Array :-

const cars = ["Saab", "Volvo", "BMW"];  
 let car = cars[0];

Example of object :-

const person = {  
  firstName: "Arnav",  
  lastName: "Pandav",  
    age: 28,  
   eyeColor: "blue"  
 };

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

🡪 The ‘split()’ method in JavaScript is used to split a string into an

Array of substring based on a specified separator.

Example :-

<!DOCTYPE html>

<html>

<body>

<h1>JavaScript Strings</h1>

<h2>The split() Method</h2>

<p>split() splits a string into an array of substrings, and returns

the array.</p>

<p>The second word is:</p>

<p id="demo"></p>

<script>

let text = "How are you doing today?";

const myArray = text.split(" ");

document.getElementById("demo").innerHTML =

myArray[2];

</script>

</body>

Output

# JavaScript Strings

## The split() Method

split() splits a string into an array of substrings, and returns the array.

The second word is:

You

**Q. Check if the string contains a word script using**

**includes() method?**

🡪 The includes() method returns true if a string contains a

specified string, otherwise it returns false.

🡪 The includes() method is case sensitive.

Syntax

string.includes(searchvalue, start)

Example :-

<!DOCTYPE html>

<html>

<body>

<h1>JavaScript Strings</h1>

<h2>The includes() Method</h2>

<p>includes() returns true if a string contains a specified

string.</p>

<p>Find "world":</p>

<p id="demo"></p>

<p>includes() is not supported in Internet Explorer.</p>

<script>

let text = "Hello world, welcome to the universe.";

let result = text.includes("world");

document.getElementById("demo").innerHTML = result;

</script>

</body>

</html>

Output :-

# JavaScript Strings

## The includes() Method

includes() returns true if a string contains a specified string.

Find "world":

true

includes() is not supported in Internet Explorer.

**Q. Change all the string characters to lowercase letters**

**using to Lowercase() Method.**

🡪 The toLowerCase() method converts a string to lowercase

letters.

🡪 The toLowerCase() method does not change the original string.

Syntax

String.toLowerCase()

Example :-

<!DOCTYPE html>

<html>

<body>

<h1>JavaScript Strings</h1>

<h2>The toLowerCase() Method</h2>

<p>toLowerCase() converts a string to lowercase letters:</p>

<p id="demo"></p>

<script>

let text = "Hello World!";

let result = text.toLowerCase();

document.getElementById("demo").innerHTML = result;

</script>

</body>

</html>

Output :-

# JavaScript Strings

## The toLowerCase() Method

toLowerCase() converts a string to lowercase letters:

hello world!

**Q. What is Character at index 15 in ’30 Days of**

**JavaScript’ string? Use charAt() method.**

🡪 The charAt() method returns the character at specified index

(position) in a string.

🡪 The index of the first character is 0, the second 1,....

Example :-

<!DOCTYPE html>

<html>

<body>

<h1>JavaScript Strings</h1>

<h2>The charAt() Method</h2>

<p>charAt() returns the character at a specified index

(position) in a string.</p>

<p>Get the first character in a string:</p>

<p id="demo"></p>

<script>

let text = "HELLO WORLD";

let letter = text.charAt(0);

document.getElementById("demo").innerHTML =

letter;

</script>

</body>

</html>

Output :-

# JavaScript Strings

## The charAt() Method

charAt() returns the character at a specified index (position) in a string.

Get the first character in a string:

H

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

🡪 slice() in JavaScript – Whenever you use slice() on a string,

the method does the following. It creates a new string. It

duplicates a specified part of its calling string into the newly

created string – without altering the original string.

Syntax

string.slice(start,end)

Example :-

<!DOCTYPE html>

<html>

<body>

​

<h1>JavaScript Strings</h1>

<h2>The slice() Method</h2>

​

<p>slice() extracts a part of a string and returns the

extracted part:</p>

​

<p id="demo"></p>

<script>

let text = "Hello world!";

let result = text.slice(0, 5);

​

document.getElementById("demo").innerHTML = result;

</script>

​

</body>

</html>

​

**Q. Find the length of a string without using library**

**Function?**

🡪 The length property returns the length of a string. And the

length property of an empty string is 0.

Syntax

string.length

Example :-

<!DOCTYPE html>

<html>

<body>

<h1>JavaScript Strings</h1>

<h2>The length Property</h2>

<p>The length property returns the length of a string.</p>

<p>Find the length of "Hello World!":</p>

<p id="demo"></p>

<script>

let text = "Hello World!";

let length = text.length;

document.getElementById("demo").innerHTML = length;

</script>

</body>

</html>

Output :-

# JavaScript Strings

## The length Property

The length property returns the length of a string.

Find the length of "Hello World!":

12

**Q . What is the use of isNaN function?**

🡪 The “isNaN” function in JavaScript is used to determine

whether a value isNaN(Not a Number) or not. The primary

purpose of “isNaN” is to check if a given value is valid

numeric value or if it represents the special NaN value in

JavaScript.

Syntax 🡪 isNaN(value)

**Q. What is negative Infinity?**

🡪 In JavaScript, negative infinity is a special numeric value

representing the mathematical concept of negative infinity.

It is often denoted as “Infinity”. Negative infinity is used to

represent a value that is smaller than any other number

including negative numbers.

1) Comparison – Any finite number is greater than

negative infinity, and negative infinity is less than any

finite number.

2)Operations – Arithmetic operations involving negative

infinity result in negative infinity.

3) Division - Dividing a nonzero finite number by

negative infinity results in zero.

**Q. Which company developed JavaScript?**

🡪 JavaScript was developed by Netscape Communications

corporation, a technology company that played a

significant role in the early days of the internet. Brendan

Eich, a programmer at Netscape, created JavaScript in

1995. The development of JavaScript was influenced by

the need for a scripting language that could enable more

interactive and dynamic web pages.

**Q. What are undeclared and undefined Variables?**

🡪 “undeclared” and “undefined” are terms related to

variables in JavaScript, and they refer to different states

of variable.

Undeclared Variables :- An undeclared variable is a

variable that has been used in the code without being

formally declared using var, let, or const. When you

reference a variable that has not been declared,

JavaScript will typically treat it as a global variable if

strict mode is not enabled. This can lead to unexpected

behaviour and is generally considered bad practice.

Undefined Variables : An undefined variable is a

variable that has been declared but has not been assigned

a value. In JavaScript, when you declare a variable

without assigning an initial value to it, the variable is

automatically assigned the value **undefined**.

**Q. 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>Document</title>

</head>

<body>

<div id="demo">

</div>

<script>

let newParagraph = document.createElement('p');

newParagraph.textContent = 'This is a dynamically added paragraph.';

let demo = document.getElementById("demo");

demo.appendChild(newParagraph);

</script>

</body>

</html>

Output :- This is a dynamically added paragraph.

**Q. What is the difference between ViewState and**

**SessionState?**

🡪 ViewState can only be visible from a single page and

not multiple pages. SessionState 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.

**Q. What is === operator?**

🡪 The “===” operator is called the “strict equality”

operator in JavaScript. It is used for comparing two

values for equality without performing type coercion.

This means that in order for the comparison to return

“true” both the value and the data type of the operands

must be the same.

Example

document.write(5===5); 🡪 True

document.write(5=== ‘5’); 🡪 False

document.write( ‘hello’ === ‘hello’); 🡪 True

**Q. How Can the style/class of an element be changed?**

🡪 In JavaScript, you can change the style or class of an

HTML element dynamically by accessing the element

using its DOM reference and modifying its Properties.

Example :-

**Q. How to read and write a file using JavaScript?**

🡪 In a web browser environment, JavaScript has limited

capabilities to directly read and write files on the user's

device due to security restrictions. However, you can

interact with files indirectly through various

mechanisms.

1) File Input Element: You can use the HTML

<input type= “file”> element to allow users to select

files. Once a file is selected, you can access its

contents using the File API.

**Q. What are all the looping structures in JavaScript?**

🡪 Loops:

A loop is a programming construct that repeatedly

executes a block of code until a specified condition is

false. JavaScript supports several types of loops.

1. for Loop :- The ‘for’ loop is used when you know

the number of iterations in advance.

for (var i = 0; i < 5; i++) {

console.log(i);

}

2. While Loop :- The ‘while’ loop is used when the

number of iterations is not known in advance, and

the loop continues as long as a specified condition

is true.

var i = 0;

while (i < 5) {

console.log(i);

i++;

}

3. do-while Loop :- Similar to the ‘while ’loop, but

the condition is checked after the loop body is

executed, ensuring that body is executed at least

once.

var i = 0;

do {

console.log(i);

i++;

} while (i < 5);

4. for... in Loop :- Iterates over the properties of an

object.

var person = { name: 'John', age: 30 };

for (var key in person) {

console.log(key + ': ' + person[key]);

}

5. for ...of Loop :- Iterates over iterable objects like

arrays and strings.

var colors = ['red', 'green', 'blue'];

for (var color of colors) {

console.log(color);

}

**Q. How can you convert the string of any base to an**

**integer in JavaScript?**

🡪 There’s a function called parseInt() in JavaScript, this is

used for parsing a string as an argument and it returns an

integer of the specified radix(basically the base of the

numerical system) as output.

🡪 <!DOCTYPE html>

<html>

<body>

<h2>The parseInt() Method</h2>

<p id="demo"></p>

<script>

document.getElementById("demo").innerHTML =

parseInt("20") + "<br>" +

parseInt("20.00") + "<br>" +

parseInt("33.33") + "<br>" +

parseInt("44 50 78") + "<br>" +

parseInt(" 80 ") + "<br>" +

parseInt("50 years") + "<br>" +

parseInt("He was 30");

</script>

</body>

</html>

Output :-

## The parseInt() Method

20  
 20  
 33  
 44  
 80  
 50  
 NaN

**Q. What is the function of the delete operator?**

🡪 In JavaScript, the delete Operator’s main purpose is to

remove a property from an object. In this article, we are

going to learn about delete Operator in JavaScript.

Syntax

delete object

// or

delete object.property

//

delete object[‘property’]

Example :-

var person = {

firstName: “Arnav”,

lastName: “Pandav”,

age: 28,

eyeColor: “brown”

};

delete person.age; // or delete person[“age”];

**Q. What are all the types of Pop up boxes available in**

**JavaScript?.**

🡪 JavaScript has three kind of popup boxes.

1. Alert box

2. Confirm box

3. Prompt box

🡪 Alert box :- An alert box is often used if you want to

Make sure information comes through to the user.

Syntax :- window.alert("*sometext*");

Example :- alert("I am an alert box!");

🡪 Confirm box :- A confirm box is often used if you

want the user to verify or accept something.

When a confirm box pops up, the user will have to

click either "OK" or "Cancel" to proceed.

If the user clicks "OK", the box returns **true**. If the

user clicks "Cancel", the box returns **false**.

Syntax :- window.confirm("*sometext*");

Example :-

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

🡪 Prompt box :- A prompt box is often used if you

want the user to input a value before entering a page.

When a prompt box pops up, the user will have to

click either "OK" or "Cancel" to proceed after

entering an input value.

If the user clicks "OK" the box returns the input

value. If the user clicks "Cancel" the box returns

null

Syntax :- window.prompt("*sometext*",

“defaultText”);

Example :-

let person = prompt("Please enter your

name", "Arnav Pandav");  
 let text;  
 if (person == null || person == "") {  
    text = "User cancelled the prompt.";  
 } else {  
   text = "Hello " + person + "! How are you today?";  
 }

To display line breaks inside a popup box, use a back-

slash followed by the character n.

alert(“Hello\n How are you?”);

**Q. How can a page be forced to load another page in**

**JavaScript?**

🡪 We can use window.location property inside the script tag

to forcefully load another page in JavaScript. It is a

reference to a Location object that is represents the

current location of the document. We can change the

URL of a Window by accessing it.

Syntax :-

<script>

window.location = < Path / URL >

</script>

Example :-

<script>

window.location = "https://www.google.com/?client=safari"

</script>

**Q. What are the disadvantages of using innerHTML in**

**JavaScript?**

🡪 **The use of innerHTML very slow** : The process of using

innerHTML is much slower as its contents as slowly

built, also already parsed contents and elements are also

re-parsed which takes time.

🡪 **Preserves event handlers attached to any DOM**

**elements:** The event handlers do not get attached to the

new elements created by setting innerHTML

automatically. To do so one has to keep track of the

event handlers and attach it to new elements manually.

This may cause a memory leak on some browsers.

🡪 **Content is replaced everywhere :** Either you add,

append, delete or modify contents on a webpage using

innerHTML, all contents is replaced, also all the DOM

nodes inside that element are reparsed and recreated.

🡪 **Appending to innerHTML is not supported :** Usually,

+= is used for appending in JavaScript. But on appending

to an Html tag using innerHTML, the whole tag is re-

parsed.

🡪 **Can break the document :** There is no proper validation

provided by innerHTML, so any valid HTML code can

be used. This may break the document of JavaScript.

Even broken HTML can be used, which may lead to

unexpected problems.

🡪 **Old content replaced issue :** The old content is replaced

even if object.innerHTML = object.innerHTML + ‘html’

is used instead of object.innerHTML += ‘html’. There is

no way of appending without reparsing the whole

innerHTML. Therefore, working with innerHTML

becomes very slow. String concatenation just does not

scale when dynamic DOM elements need to be created as

the plus’ and quote openings and closings becomes

difficult to track.

**Q. Create password field with show hide functionalities.**

🡪

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Document</title>

</head>

<body>

<form>

<table>

<tr>

<td> Password :- </td>

<td> <input type="password" value="TOPS123" id="myInput"></td>

</tr>

</table>

</form><br>

<input type="checkbox" onclick="Arnav()"> Show Password

<script>

function Arnav() {

let x = document.getElementById("myInput");

if (x.type === "password"){

x.type ="text";

}

else {

x.type = "password";

}

}

</script>

</body>

</html>

**Q. 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>Document</title>

<style>

p {

font-size: 3vw;

text-align: center;

}

input {

height: 2.5vw;

width: 20vw;

font-size: 2vw;

}

button {

height: 4.5vw;

width: 5.5vw;

margin-left: 1vw;

font-weight: bold;

font-size: 2vw;

}

.container {

height: 40vw;

width: 40vw;

border: 2px solid black;

padding: 3vw;

text-align: center;

}

b {

font-size: 2vw;

}

label {

font-size: 2vw;

}

span {

color: blue;

font-size: 2vw;

font-weight: bold;

}

</style>

</head>

<body>

<div class="container">

<p> Maths Operation </p>

<label> Enter 1st number : </label>&ensp;&ensp;&ensp;&ensp;&ensp;

<input type="number" id="num1"> <br><br><br>

<label> Enter 2nd number : </label>&ensp;&ensp;&ensp;&ensp;&nbsp;

<input type="number" id="num2"> <br><br><br>

<button onclick="add()">+</button>

<button onclick="sub()">-</button>

<button onclick="mul()">\*</button><br><br>

<button onclick="Div()">/</button>

<button onclick="Pre()">%</button><br><br><br>

<b> Answer is :- </b> <span id="ans"></span>

</div>

<script>

function add() {

let number\_one = document.getElementById('num1').value;

let number\_two = document.getElementById('num2').value;

let total = Number(number\_one) + Number(number\_two);

document.getElementById('ans').innerHTML = total;

}

function sub() {

let number\_one = document.getElementById('num1').value;

let number\_two = document.getElementById('num2').value;

let total = Number(number\_one) - Number(number\_two);

document.getElementById('ans').innerHTML = total;

}

function mul() {

let number\_one = document.getElementById('num1').value;

let number\_two = document.getElementById('num2').value;

let total = Number(number\_one) \* Number(number\_two);

document.getElementById('ans').innerHTML = total;

}

function Div() {

let number\_one = document.getElementById('num1').value;

let number\_two = document.getElementById('num2').value;

let total = Number(number\_one) / Number(number\_two);

document.getElementById('ans').innerHTML = total;

}

function Pre() {

let number\_one = document.getElementById('num1').value;

let number\_two = document.getElementById('num2').value;

let total = Number(number\_one ) % Number(number\_two);

document.getElementById('ans').innerHTML = total;

}

</script>

</body>

</html>

**Q. Marksheet for Information Technology.**

🡪

<!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>

.container {

height: 45vw;

width: 45vw;

border: 2px solid black;

padding: 5vw;

}

h3 {

text-align: center;

}

input {

height: 2vw;

width: 20vw;

float: right;

font-size: 1.5vw;

}

p {

font-size: 1.5vw;

}

button {

height: 3.5vw;

width: 8.5vw;

font-size: 1.3vw;

margin-left: 24.5vw;

}

b {

font-size: 1.8vw;

}

span {

font-size: 1.8vw;

}

</style>

</head>

<body>

<div class="container">

<h1> Marksheet for Information Technology </h1>

<h3> Enter Marks</h3>

<p> 1. C Language <input type="number" id="c"> </p>

<p> 2. C++ Language <input type="number" id="c++"> </p>

<p> 3. Database <input type="number" id="data"> </p>

<p> 4. HTML <input type="number" id="html"> </p>

<p> 5. CSS <input type="number" id="css"> </p>

<p> 6. PHP <input type="number" id="php"> </p>

<p> 7. Core java <input type="number" id="java"> </p><br>

<button onclick="total()"> Result </button><br><br>

<b> Total is :- &ensp;&ensp;</b><b id="total" style="color: blue;"> </b><span>/350</span> <br><br>

<b> Percentage is % :- &ensp;&ensp;</b><b id="Pre" style="color: blue;"></b><span>/350</span>

</div>

</body>

<script>

function total() {

let a1 = document.getElementById('c').value;

let a2 = document.getElementById('c++').value;

let a3 = document.getElementById('data').value;

let a4 = document.getElementById('html').value;

let a5 = document.getElementById('css').value;

let a6 = document.getElementById('php').value;

let a7 = document.getElementById('java').value;

let Total = Number(a1) + Number(a2) + Number(a3) + Number(a4) + Number(a5) + Number(a6) + Number(a7);

let Percentage= Total \* 100/350;

document.getElementById('total').innerHTML = Total;

document.getElementById('Pre').innerHTML = Percentage;

}

</script>

</html>