**Module 17) Javascript For Full Stack Assignment**

1. **JavaScript Introduction**

**Theory Assignment**

**Question 1**: What is JavaScript? Explain the role of JavaScript in web development.

**JavaScript** is a high-level, interpreted programming language primarily used to create interactive and dynamic content on websites. It is one of the core technologies of the World Wide Web, alongside **HTML** and **CSS**.

#### ****Role of JavaScript in Web Development:****

**Client-Side Interactivity**: JavaScript allows developers to add dynamic behavior to web pages, such as form validation, interactive maps, image sliders, and real-time updates without refreshing the page.

**DOM Manipulation**: JavaScript can access and manipulate the Document Object Model (DOM) to change the structure, style, and content of a webpage dynamically.

**Event Handling**: It enables the handling of user events like mouse clicks, keyboard input, or scrolling.

**Asynchronous Programming**: Using tools like **AJAX** and the **Fetch API**, JavaScript can communicate with servers in the background, enabling features like live chat, instant search, or data updates.

**Back-End Development**: With platforms like **Node.js**, JavaScript is also used for server-side programming, making it possible to build full-stack web applications using a single language.

**Rich Ecosystem**: JavaScript has a vast ecosystem of frameworks and libraries (e.g., React, Angular, Vue.js) that simplify and speed up web development.

**Question 2**: How is JavaScript different from other programming languages like Python or Java?

| **Feature** | **JavaScript** | **Python** | **Java** |
| --- | --- | --- | --- |
| **Type** | Interpreted, scripting language | Interpreted, high-level language | Compiled, object-oriented language |
| **Use Case** | Mainly for web development (front-end and back-end) | Data science, AI, scripting, web (with frameworks) | Enterprise applications, Android apps |
| **Syntax** | C-style, semi-colons, curly braces | English-like, indentation-based | Strict syntax, requires class structures |
| **Typing** | Dynamically typed | Dynamically typed | Statically typed |
| **Runtime** | Runs in browsers or Node.js | Runs in Python interpreter | Runs on Java Virtual Machine (JVM) |
| **Performance** | Fast in browsers (V8 engine) | Slower than JavaScript or Java in many web tasks | High performance due to compiled nature |
| **Object-Oriented** | Prototype-based | Class-based (since Python 3.x) | Purely class-based |
| **Concurrency** | Event-driven (single-threaded with async) | Multi-threaded with GIL limitations | Multi-threaded, robust concurrency support |

**Question 3**: Discuss the use of <script> tag in HTML. How can you link an external JavaScript file to an HTML document?

#### Use of <script> Tag in HTML:

The <script> tag\*\* is used in HTML to embed JavaScript code or to link to an external JavaScript file. It tells the browser to execute JavaScript, either directly within the HTML file or from an external source.

### ****1. Embedding JavaScript Internally:****

You can write JavaScript code directly within the <script> tag in the HTML file.

### ****2. Linking an External JavaScript File:****

To keep code modular and maintainable, it's common to write JavaScript in a separate file and link it using the <script> tag with the src attribute.

**Lab Assignment**

**Task**:

o Create a simple HTML page and add a <script> tag within the page.

o Write JavaScript code to display an alert box with the message "Welcome to

JavaScript!" when the page loads.

1. **Variables and Data Types**

**Theory Assignment**

**Question 1**: What are variables in JavaScript? How do you declare a variable using var, let, and const?

#### ****What are Variables?****

Variables in JavaScript are containers used to store data values such as numbers, strings, or objects. They allow you to reference and manipulate values in your code.

#### ****Ways to Declare Variables:****

### var

**Function-scoped**

Can be **re-declared** and **updated**

**Hoisted** (moved to the top of the scope)

var name = "Arjun";

### let

**Block-scoped**

Can be **updated**, but **not re-declared** in the same scope

Introduced in ES6

let age = 22;

### const

**Block-scoped**

Cannot be **updated** or **re-declared**

Must be **initialized** at the time of declaration

const country = "India";

**Question 2**: Explain the different data types in JavaScript. Provide examples for each.

JavaScript has **8 data types**:

#### ****1. String****

Represents textual data.

let name = "Arjun";

#### ****2. Number****

Represents both integers and floating-point numbers.

let age = 25;let price = 99.99;

#### ****3. Boolean****

Represents logical values: true or false.

let isStudent = true;

#### ****4. Undefined****

A variable declared but not assigned any value.

let x;console.log(x); // undefined

#### ****5. Null****

Represents intentional absence of any object value.

let y = null;

#### ****6. Object****

Used to store collections of data or more complex entities.

let person = { name: "Arjun", age: 22 };

#### ****7. Array**** (Technically a type of Object)

Used to store ordered lists.

let fruits = ["Apple", "Banana", "Mango"];

#### ****8. Symbol**** (ES6)

Unique and immutable primitive value.

let id = Symbol("uniqueId");

**Question 3**: What is the difference between undefined and null in JavaScript?

| **Feature** | **undefined** | **null** |
| --- | --- | --- |
| **Meaning** | Variable declared but no value assigned | Intentional absence of value |
| **Type** | undefined (primitive type) | object (this is a known JS quirk) |
| **Set By** | JavaScript itself | Developer manually |
| **Example** | let x; → x === undefined | let x = null; → x === null |

**Lab Assignment**

**Task**:

o Write a JavaScript program to declare variables for different data types (string,

number, boolean, null, and undefined).

o Log the values of the variables and their types to the console using console.log().

1. **JavaScript Operators**

**Theory Assignment**

**Question 1**: What are the different types of operators in JavaScript? Explain with examples.

o Arithmetic operators

o Assignment operators

o Comparison operators

o Logical operators

JavaScript supports several types of **operators** used to perform operations on values and variables.

### ****1. Arithmetic Operators****

Used to perform basic mathematical operations.

| **Operator** | **Description** | **Example** | **Result** |
| --- | --- | --- | --- |
| + | Addition | 5 + 3 | 8 |
| - | Subtraction | 5 - 3 | 2 |
| \* | Multiplication | 5 \* 3 | 15 |
| / | Division | 10 / 2 | 5 |
| % | Modulus (Remainder) | 10 % 3 | 1 |
| \*\* | Exponentiation | 2 \*\* 3 | 8 |
| ++ | Increment | a++ | a = a + 1 |
| -- | Decrement | a-- | a = a - 1 |

### ****2. Assignment Operators****

Used to assign values to variables.

| **Operator** | **Description** | **Example** | **Equivalent To** |
| --- | --- | --- | --- |
| = | Assign | x = 5 | x = 5 |
| += | Add and assign | x += 3 | x = x + 3 |
| -= | Subtract and assign | x -= 2 | x = x - 2 |
| \*= | Multiply and assign | x \*= 4 | x = x \* 4 |
| /= | Divide and assign | x /= 2 | x = x / 2 |
| %= | Modulus and assign | x %= 3 | x = x % 3 |

### ****3. Comparison Operators****

Used to compare values and return a boolean (true or false).

| **Operator** | **Description** | **Example** | **Result** |
| --- | --- | --- | --- |
| == | Equal to (loose) | 5 == '5' | true |
| === | Strict equal (type + value) | 5 === '5' | false |
| != | Not equal | 5 != '5' | false |
| !== | Strict not equal | 5 !== '5' | true |
| > | Greater than | 6 > 3 | true |
| < | Less than | 3 < 6 | true |
| >= | Greater than or equal | 5 >= 5 | true |
| <= | Less than or equal | 4 <= 3 | false |

### ****4. Logical Operators****

Used to combine or invert boolean expressions.

| **Operator** | **Description** | **Example** | **Result** |
| --- | --- | --- | --- |
| && | Logical AND | true && false | false |
| ` |  | ` | Logical OR |
| ! | Logical NOT | !true | false |

**Question 2**: What is the difference between == and === in JavaScript?

| **Feature** | **== (Loose Equality)** | **=== (Strict Equality)** |
| --- | --- | --- |
| Type Conversion | Performs type conversion if needed | Does **not** perform type conversion |
| Compares | Value only | Value **and** data type |
| Safer to use | ❌ Can cause unexpected results | ✅ Recommended for predictable behavior |

**Lab Assignment**

**Task**:

o Create a JavaScript program to perform the following:

 Add,subtract, multiply, and divide two numbers using arithmetic operators.



Use comparison operators to check if two numbers are equal and if one

number is greater than the other.

Use logical operators to check if both conditions (e.g., a > 10 and b < 5) are

true.

1. **Control Flow (If-Else, Switch)**

**Theory Assignment**

**Question 1**: What is control flow in JavaScript? Explain how if-else statements work with an example.

#### ****What is Control Flow?****

Control flow in JavaScript is the **order in which the code is executed**. By default, JavaScript runs code from top to bottom. But using control flow statements like if-else, switch, for, while, etc., you can change this order to make decisions or repeat code.

### ****Using**** if-else ****Statements****

The if-else statement allows you to run different blocks of code based on whether a condition is true or false.

#### ****Syntax:****

if (condition) {

// code runs if condition is true

} else if (anotherCondition) {

// runs if the first is false and this is true

} else {

// runs if all conditions are false

}

**Question 2**: Describe how switch statements work in JavaScript. When should you use a switch statement instead of if-else?

#### ****What is a**** switch ****Statement?****

A switch statement is used to **perform different actions based on multiple possible values of a variable or expression**.

#### ****Syntax:****

switch (expression) {

case value1:

// code block

break;

case value2:

// code block

break;

default:

// code block

}

break stops the execution once a match is found.

default is optional and runs if no case matches.

**Lab Assignment**



**Task 1**:

o Write a JavaScript program to check if a number is positive, negative, or zero using

an if-else statement.



**Task 2**:

o Create a JavaScript program using a switch statement to display the day of the week

based on the user input (e.g., 1 for Monday, 2 for Tuesday, etc.).

1. **Loops (For, While, Do-While)**

**Theory Assignment**

**Question 1**: Explain the different types of loops in JavaScript (for, while, do-while). Provide a basic example of each.

Loops are used to **execute a block of code repeatedly** until a specified condition is false.

### ****1.**** for ****Loop****

Used when the number of iterations is **known**.

#### Syntax:

for (initialization; condition; update) {

// code block

}

#### Example:

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

console.log("Count:", i);

}

🧾 Output:

Count: 1Count: 2Count: 3Count: 4Count: 5

### ****2.**** while ****Loop****

Used when the number of iterations is **unknown**; condition is checked **before** executing the code.

#### Syntax:

while (condition) {

// code block

}

#### Example:

let i = 1;while (i <= 3) {

console.log("Number:", i);

i++;

}

🧾 Output:

Number: 1Number: 2Number: 3

### ****3.**** do-while ****Loop****

Similar to while, but condition is checked **after** the code runs. So the loop **runs at least once**.

#### Syntax:

do {

// code block

} while (condition);

#### Example:

let j = 1;do {

console.log("Step:", j);

j++;

} while (j <= 2);

🧾 Output:

Step: 1Step: 2

**Question 2**: What is the difference between a while loop and a do-while loop?

| **Feature** | **while loop** | **do-while loop** |
| --- | --- | --- |
| Condition checked | Before each iteration | After each iteration |
| Guaranteed to run once | ❌ No | ✅ Yes |
| Use case | When code **might not run at all** | When code **must run at least once** |

The while loop doesn’t run because the condition is false.  
The do-while runs **once**, then stops.

**Lab Assignment**

 **Task 1**:

o

Write a JavaScript program using a for loop to print numbers from 1 to 10.

 **Task 2**:o Create a JavaScript program that uses a while loop to sum all even numbers

between 1 and 20.

 **Task 3**:

o Write a do-while loop that continues to ask the user for input until they enter a

number greater than 10.

**6. Functions**

**Theory Assignment**

**Question 1: What are functions in JavaScript? Explain the syntax for declaring and calling a function.**

#### ****What is a Function?****

A **function** is a reusable block of code designed to perform a specific task. Instead of writing the same code again and again, you define a function once and call it whenever needed.

### ****Function Declaration Syntax:****

function functionName(parameters) {

// code to execute

}

### ****Function Call Syntax:****

functionName(arguments);

**Question 2: What is the difference between a function declaration and a function expression?**

| **Feature** | **Function Declaration** | **Function Expression** |
| --- | --- | --- |
| Definition style | Declared with function keyword | Assigned to a variable |
| Hoisting | ✅ Hoisted (can be called before definition) | ❌ Not hoisted |
| Use in code | Can be defined anywhere | Often used in callbacks |

**Question 3: Discuss the concept of parameters and return values in functions.**

#### ****Parameters:****

Parameters are **placeholders** for values passed to a function.

They're defined in the function declaration.

#### ****Return Values:****

A function can return a value using the return statement.

This allows you to **use the function's output elsewhere** in your code.

**Lab Assignment**



**Task 1**:

o Write a function greetUser that accepts a user’s name as a parameter and displays a

greeting message (e.g., "Hello, John!").



**Task 2**:

o Create a JavaScript function calculateSum that takes two numbers as parameters,

adds them, and returnsthe result.

1. **Arrays**

**Theory Assignment**

**Question 1: What is an array in JavaScript? How do you declare and initialize an array?**

#### ****What is an Array?****

An **array** in JavaScript is a data structure that can hold multiple values in a single variable. These values can be of any type — numbers, strings, objects, etc.

### ****Declaring and Initializing an Array:****

#### Using square brackets [] (most common):

let fruits = ["apple", "banana", "mango"];

#### Using the Array() constructor:

let numbers = new Array(10, 20, 30);

Arrays are **zero-indexed**: the first item is at index 0.

**Question 2: Explain the methods push(), pop(), shift(), and unshift() used in arrays.**

| **Method** | **Action** | **Affects Start/End** | **Returns Value?** |
| --- | --- | --- | --- |
| push() | Adds to **end** | End | ❌ No |
| pop() | Removes from **end** | End | ✅ Yes |
| shift() | Removes from **start** | Start | ✅ Yes |
| unshift() | Adds to **start** | Start | ❌ No |

**Lab Assignment**



**Task 1**:

o Declare an array of fruits (["apple", "banana", "cherry"]). Use JavaScript to:

 Add a fruit to the end of the array.



Remove the first fruit from the array.



Log the modified array to the console.



**Task 2**:

o Write a program to find the sum of all elements in an array of numbers.

1. **Objects**

**Theory Assignment**

**Question 1: What is an object in JavaScript? How are objects different from arrays?**

#### ****What is an Object?****

An **object** in JavaScript is a collection of **key-value pairs**. It is used to store related data and functions (methods) together.

### ****How are Objects Different from Arrays?****

| **Feature** | **Object** | **Array** |
| --- | --- | --- |
| Structure | Key-value pairs | Indexed list |
| Key type | Keys can be strings or symbols | Keys are numeric indices (0, 1…) |
| Use case | Group related properties about an item | Store an ordered list of values |

**Question 2: Explain how to access and update object properties using dot notation and bracket notation.**

There are **two ways** to access or modify object properties:

### ****Dot Notation**** (most common):

Use when you know the exact name of the property.

#### Syntax:

objectName.propertyName

### ****Bracket Notation:****

Use when the property name is stored in a variable or has special characters/spaces.

#### Syntax:

objectName["propertyName"]

| **Feature** | **Dot Notation** | **Bracket Notation** |
| --- | --- | --- |
| Syntax | object.property | object["property"] |
| Use when | Property is a valid name | Property is dynamic or has special characters |
| Safer for spaces? | ❌ No | ✅ Yes |

**Lab Assignment**



**Task**:

o Create a JavaScript object car with properties brand, model, and year. Use JavaScript

to:

 Access and print the car’s brand and model.



Update the year property.

 Add a new property color to the car object.

1. **JavaScript Events**

**Theory Assignment**

**Question 1: What are JavaScript events? Explain the role of event listeners.**

#### ****What are JavaScript Events?****

JavaScript **events** are actions or occurrences that happen in the browser and can be responded to using code. Examples of events include:

Clicking a button (click)

Moving the mouse (mousemove)

Pressing a key (keydown)

Loading a page (load)

#### ****What is an Event Listener?****

An **event listener** is a function in JavaScript that waits for a specific event to occur, and then runs some code in response.

#### ****Why use event listeners?****

They allow your website to be **interactive and dynamic** by responding to user actions.

**Question 2: How does the addEventListener() method work in JavaScript? Provide anexample.**

#### ****Syntax:****

element.addEventListener(event, function, useCapture);

event: Type of event (e.g., "click", "mouseover")

function: Code to run when the event occurs

useCapture: Optional; usually false (used in advanced event handling)

### ****Basic Example:****

<p id="demo">Click the button to change this text.</p><button id="myButton">Click Me</button>

<script>

const btn = document.getElementById("myButton");

btn.addEventListener("click", function() {

document.getElementById("demo").innerText = "You clicked the button!";

});</script>

When the button is clicked, the text in the paragraph changes.

**Lab Assignment**



**Task**:

o Create a simple webpage with a button that, when clicked, displays an alert saying

"Button clicked!" using JavaScript event listeners.

1. **DOM Manipulation**

**Theory Assignment :**

**Question 1: What is the DOM (Document Object Model) in JavaScript? How does JavaScript interact with the DOM?**

#### ****What is the DOM?****

The **DOM (Document Object Model)** is a programming interface for web documents. It represents the page so that programs (like JavaScript) can change the document structure, style, and content dynamically.

The DOM models an HTML document as a **tree of nodes** (elements, text, attributes).

Each element in the HTML becomes an object you can manipulate using JavaScript.

#### ****How JavaScript interacts with the DOM:****

JavaScript can:

**Access** elements on the webpage.

**Modify** elements (change text, styles, attributes).

**Add or remove** elements.

**Respond** to user actions (clicks, typing) by modifying the DOM dynamically.

**Question 2: Explain the methods getElementById(), getElementsByClassName(), and querySelector() used to select elements from the DOM.**

| **Method** | **What it selects** | **Returns** | **Example** |
| --- | --- | --- | --- |
| getElementById() | Element with a specific **id** | A **single element** or null | document.getElementById("demo") |
| getElementsByClassName() | All elements with a specific **class** | A **live HTMLCollection** (array-like) | document.getElementsByClassName("item") |
| querySelector() | The **first** element that matches a CSS selector | A **single element** or null | document.querySelector(".item") |

getElementById is fast and unique since IDs should be unique.

getElementsByClassName returns a live collection of all matching elements.

querySelector allows using any CSS selector (class, id, tag, attribute).

**Lab Assignment**



**Task**:o Create an HTML page with a paragraph (<p>) that displays "Hello, World!".

o Use JavaScript to:



Change the text inside the paragraph to "JavaScript is fun!".



Change the color of the paragraph to blue.

1. **JavaScript Timing Events (setTimeout, setInterval)**

**Theory Assignment**

**Question 1: Explain the setTimeout() and setInterval() functions in JavaScript. How are they used for timing events?**

#### setTimeout()

Executes a function **once** after a specified delay (in milliseconds).

Used to **delay** an action.

#### setInterval()

Executes a function **repeatedly** at specified time intervals (in milliseconds).

Used to run a task **periodically**.

#### ****How They Work:****

| **Function** | **Purpose** | **Runs once or repeatedly?** |
| --- | --- | --- |
| setTimeout() | Delay a function execution | Once |
| setInterval() | Repeat function execution | Multiple times, at intervals |

**Question 2: Provide an example of how to use setTimeout() to delay an action by 2 seconds.**

setTimeout(function() {

console.log("This message is shown after 2 seconds");

}, 2000); // 2000 milliseconds = 2 seconds

Explanation:

The anonymous function inside setTimeout runs once after 2 seconds.

You can replace the function with any code or a named function.

**Lab Assignment**

 **Task 1**:

o Write a program that changes the background color of a webpage after 5

secondsusing setTimeout().

 **Task 2**:

o Create a digital clock that updates every second using setInterval().

**12. JavaScript Error Handling**

**Theory Assignment**

**Question 1: What is error handling in JavaScript? Explain the try, catch, and finally blockswith an example.**

#### ****What is Error Handling?****

Error handling is the process of **detecting and managing errors** in a program so it doesn’t crash unexpectedly. In JavaScript, it helps your code handle problems gracefully.

#### ****The**** try****,**** catch****, and**** finally ****blocks:****

try **block:** Code that may throw an error is placed inside this block.

catch **block:** If an error occurs in try, it is caught here, where you can handle or log the error.

finally **block:** Runs **always** after try and catch regardless of whether an error happened or not. Useful for cleanup.

**Question 2: Why is error handling important in JavaScript applications?**

**Prevents application crashes:** Without error handling, errors can stop your whole program.

**Improves user experience:** Allows showing meaningful messages instead of breaking.

**Helps debugging:** Logs errors for developers to fix issues faster.

**Maintains control flow:** Lets the program continue running or exit gracefully even when unexpected errors happen.

**Lab Assignment**



**Task**:

o Write a JavaScript program that attempts to divide a number by zero. Use try

catchto handle the error and display an appropriate error message.