**JavaScript Introduction**

Q1: What is JavaScript? Explain the role of JavaScript in web development.

Ans: - JavaScript is a scripting or programming language. This is a lightweight, interpreted programming language primarily used to add dynamic and implement complex features on web pages.

Q2: How is JavaScript different from other programming languages like Python or Java?

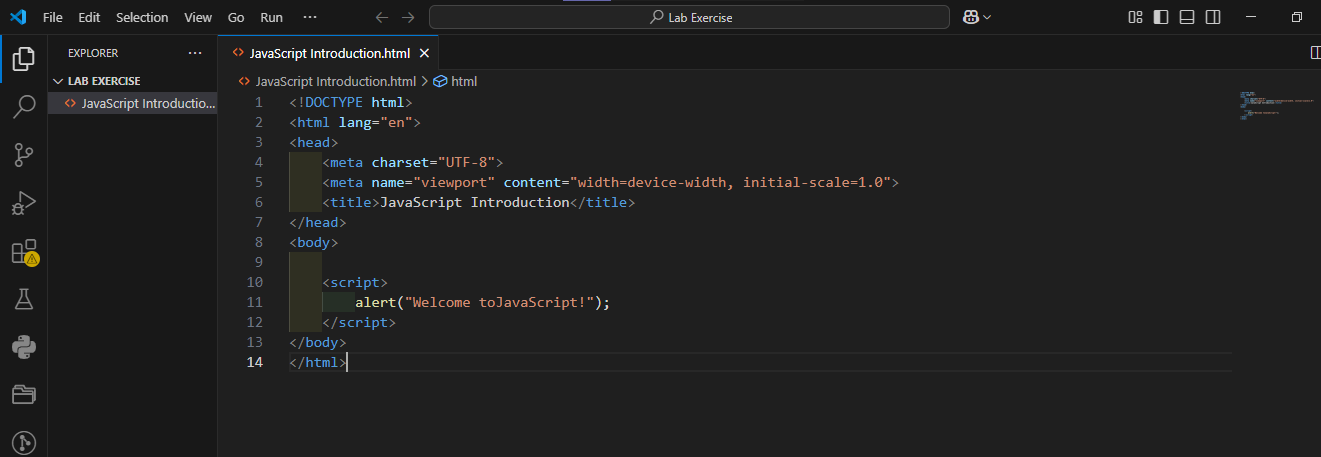
Ans: - JavaScript is primarily used for client-side web development, while Python versatility, especially in data science, data mining and backend development, Java is a compiled language, run it through a compiler, and create bytecode.

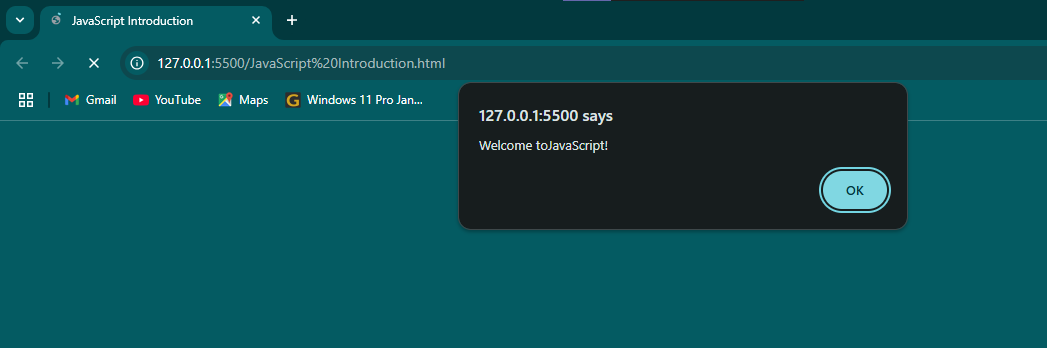
Q3: Discuss the use of <script> tag in HTML. How can you link an external JavaScript file to an HTML document?

Ans: - I use the src attributewithin the <script> tag in the HTML file to specify the source file. The external JavaScript file is then linked to the HTML document, and the execution of its code.

**Lab Tasks**

* Create a simple HTML page and add a <script> tag within the page.
* Write JavaScript code to display an alert box with the message "Welcome toJavaScript!" when the page loads.

Code:

Output:

Variables and Data Types

Q1: What are variables in JavaScript? How do you declare a variable using var, let, and const?

Ans: - variables are named containers that store values that can be accessed and manipulated throughout your code. You can declare variables using var, let, or const. let and const are keywords introduced in ES6 for declaring variables.

Q2: Explain the different data types in JavaScript. Provide examples for each.

Ans: - JavaScript has two categories of data types:

* Primitive: String, Number, Boolean, Null, Undefined, Symbol, BigInt
* let name = "John" (String)
* let age = 30 (Number)
* let isOnline = true (Boolean)
* let value = null (Null)
* let x (Undefined)
* let symbol1 = Symbol("id") (Symbol)
* let largeNumber = BigInt(12345678900987654321) (BigInt)
* Non-Primitive: Object, Array, Function
* (Object)

person = {  
 name: "Kuldeep",  
 surname: "Gohil"  
 }

* (array)

const fruits = ["apple", "banana", "orange"];

* (function)

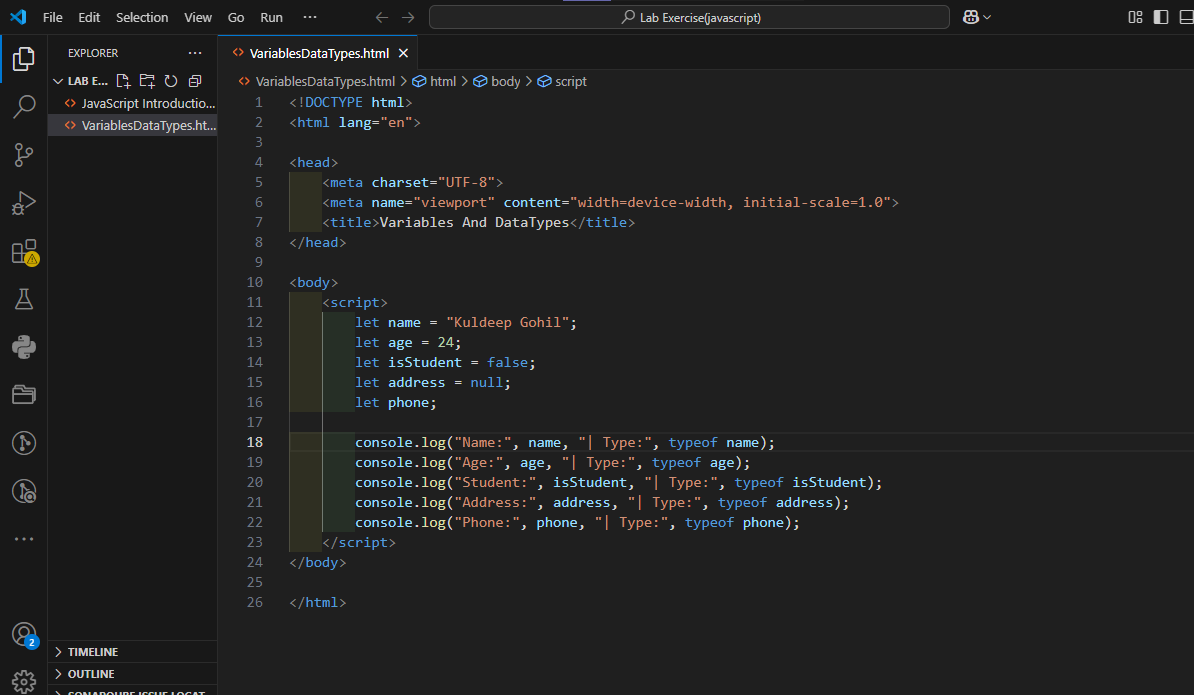
function add(a, b) {  
 return a + b;  
 }

Q3: What is the difference between undefined and null in JavaScript?

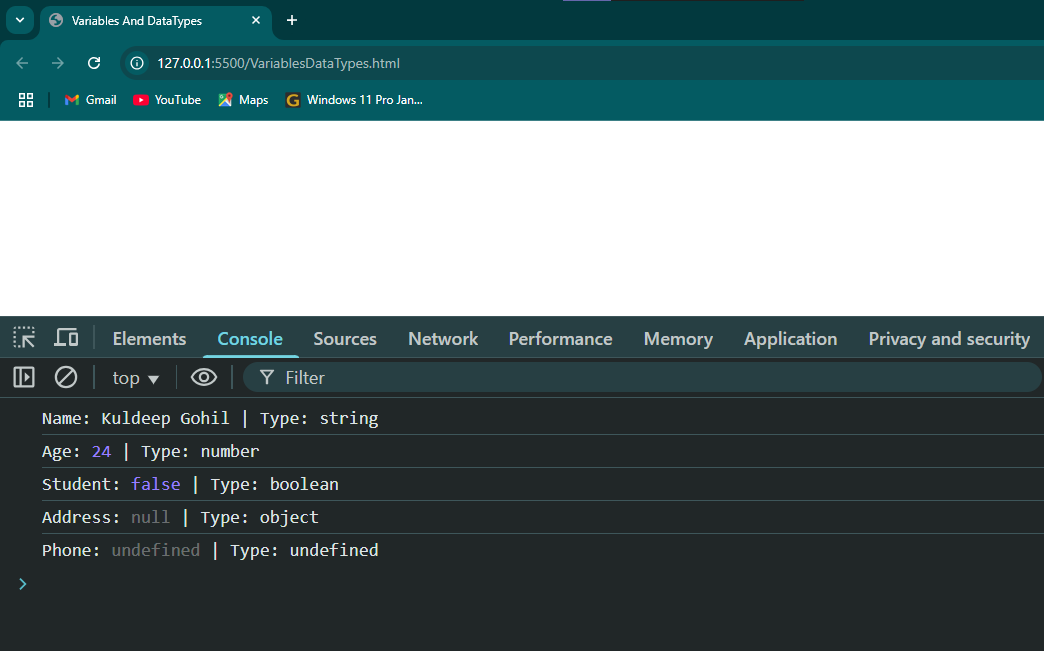
Ans: - undefined indicates a variable has been declared but hasn't been assigned a value. Null in JavaScript means an empty value and return object.

**Lab Tasks**

* Write a JavaScript program to declare variables for different data types (string,number, boolean, null, and undefined).
* Log the values of the variables and their types to the console using console.log()

Code:

Output:



JavaScript Operators

Q1: What are the different types of operators in JavaScript? Explain with examples. Arithmetic operators, Assignment operators, Comparison operators, Logical operators.

Ans: - JavaScript provides various types of operators to perform different operations on variables.

1. **Arithmetic operators:-** These operators perform mathematical calculations.

**(+.**-,\*,/,%,++,--)

Example:

let x = 10;

let y = 5;

x + y = 15 (Addition),

x – y = 5 (Subtraction),

x \* y = 50 (Multiplication),

x / y = 2 (Division),

x % y = 0 (Modulus),

x++ = 11 (Increment),

y-- = 4 (Decrement)

1. **Assignment** **operators:-** These operators assign values to variables.

(=,+=,-=,\*=,/=,%=)

Example:

let x = 10;

x += 5 (x = x + 5 → 15),

x -= 3 (x = x - 3 → 7),

x \*= 2 (x = x \* 2 → 20),

x /= 2 (x = x / 2 → 5),

x %= 5 (x = x % 5 → 0)

1. **Comparison operators:-** These operators compare values and return true or false.

(==,===,!=,!==,>,<,>=,<=)

Example:

2 == "2" → true (value check) (Equal to),

8 === "8" → false (both check value and type) (Strict Equal to),

4 != 6 → true (Not equal to),

4 !== 4→ false (Strict not equal to),

14 > 2 → true (Greater than),

1 < 6 → true (Less than),

8 >= 8 → true (Greater than or equal to),

8 <= 4 → false (Less than or equal to),

1. **Logical operators:-** These operators compare values and return true or false.

(&&, ||, ! )

Example:

let x = true, y = false;

x && y → false (Logical AND) (both true),

x || y → true (Logical OR) (one true),

(!(10 > 5)) → false (Logical NOT) (inverts truth value),

Q2: What is the difference between == and === in JavaScript?

Ans: -

Loose Equality (==): These are only check value.

For example: 1 == "1". The string "1" is implicitly converted to the number 1, so the comparison evaluates to true.

Strict Equality (===): checks both value and DataType.

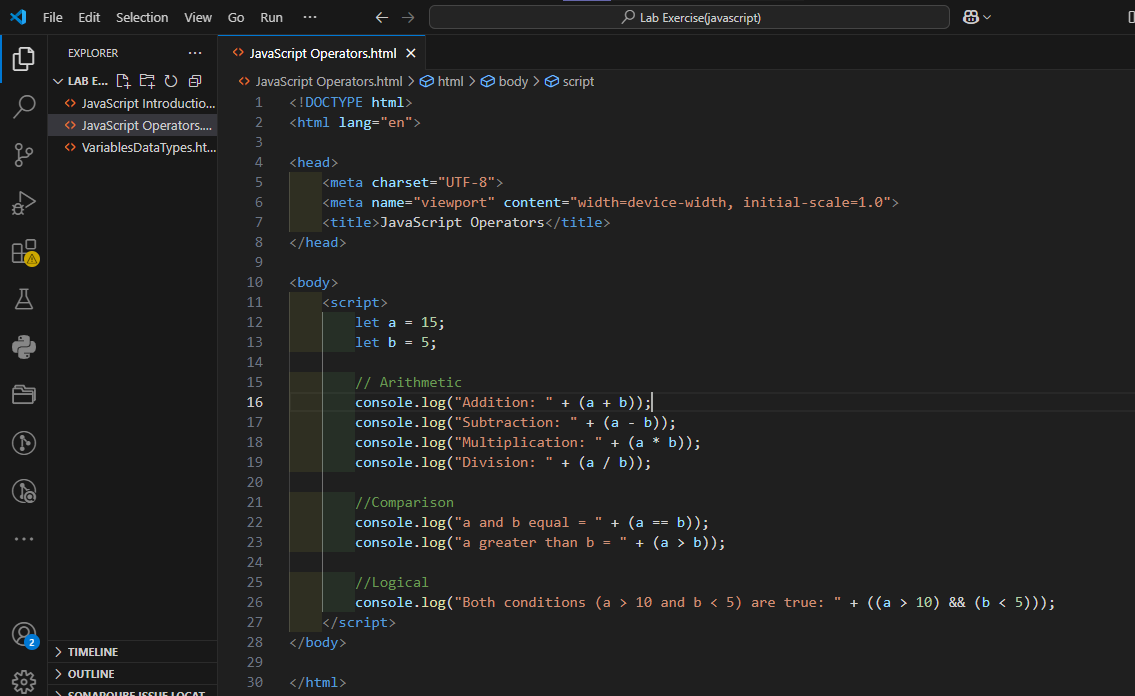
For example: 2 === "2". The Number 2 is DataType(Number) vs String “2” is DataType (String), so the comparison evaluates to false.

**Lab Tasks**

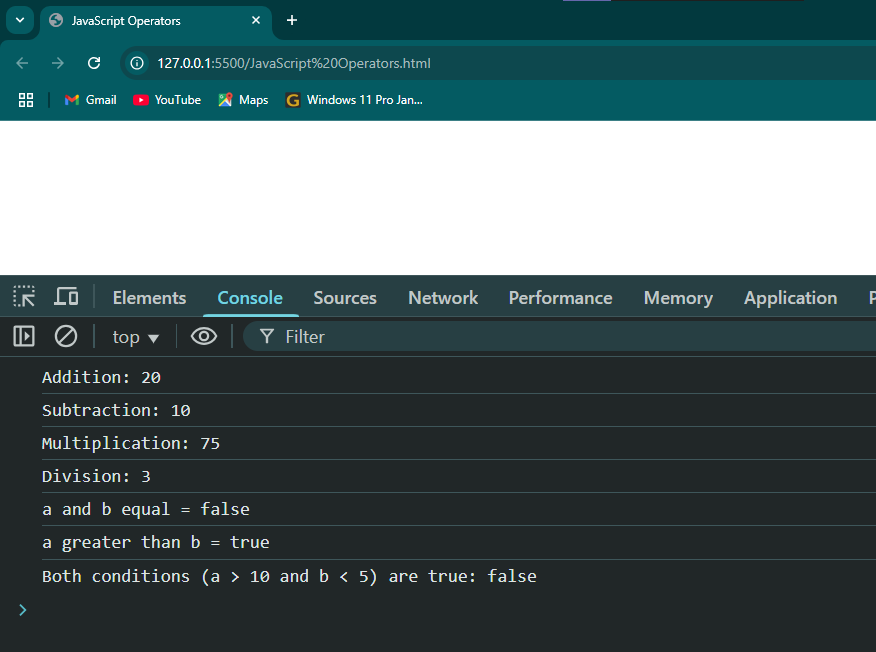
Create a JavaScript program to perform the following:

* 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.

Code:



Output:



Control Flow (If-Else, Switch)

Q1: What is control flow in JavaScript? Explain how if-else statements work with an example.

Ans: - Control flow statement is an if-else statement. The if/else statement executes a block of code if a specified condition is true. If the condition is false, another block of code can be executed.

Example:

let marks = 75;

if (marks >= 90) {

console.log("Grade: A");

} else if (marks >= 70) {

console.log("Grade: B");

} else {

console.log("Grade: C");

}

Output:

Grade: B

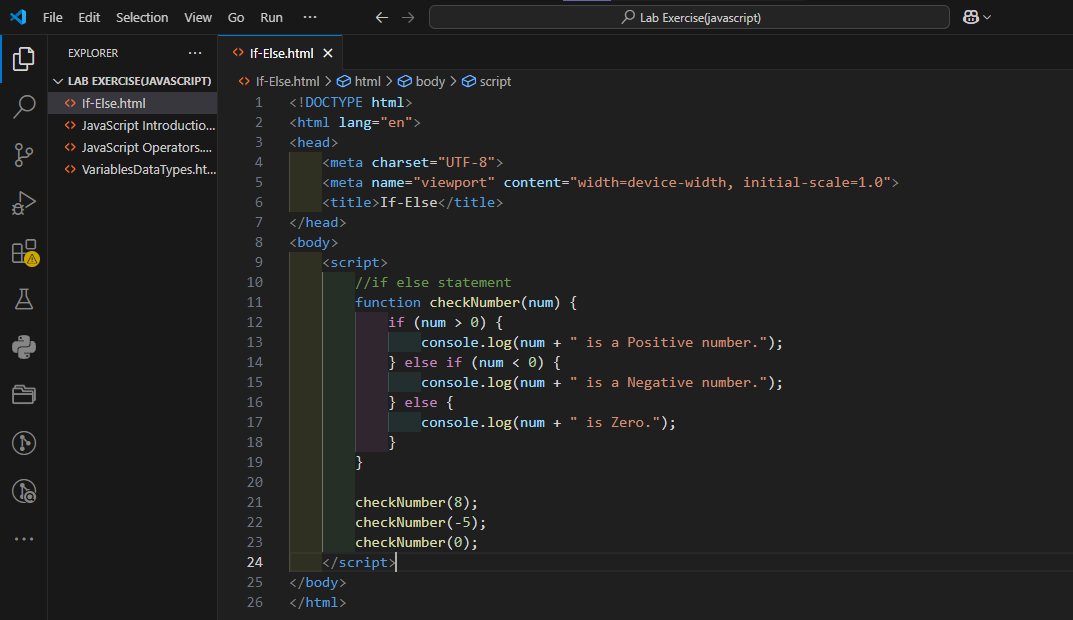
Q2: Describe how switch statements work in JavaScript. When should you use a switch statement instead of if-else?

Ans: - A switch statement evaluates an expression is compared against multiple possible values, or the default case if no match is found. Switch case is considered faster and more readable than nested if-else statements. If-else statement is used to decide between two options.

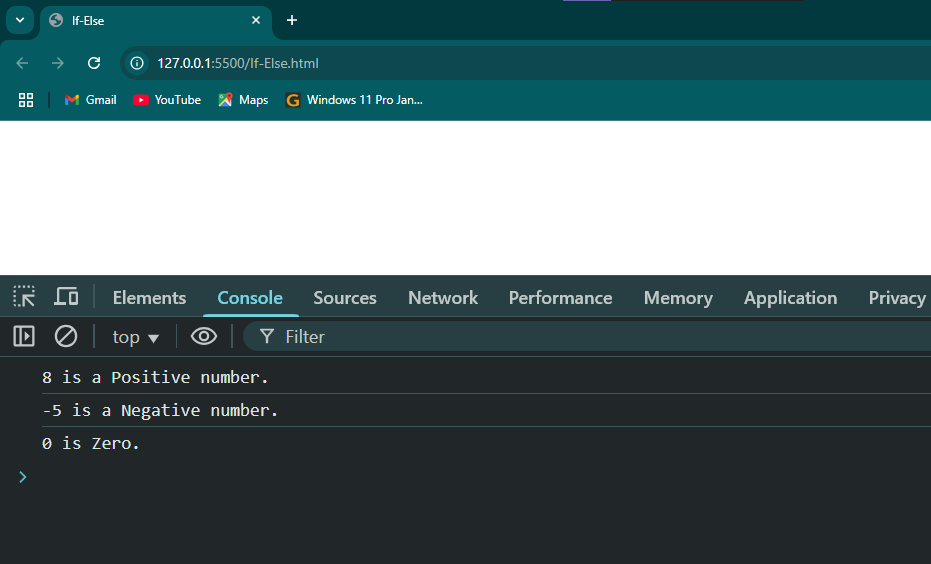
**Lab Tasks**

**Task 1 :** Write a JavaScript program to check if a number is positive, negative, or zero using an if-else statement.

Code:

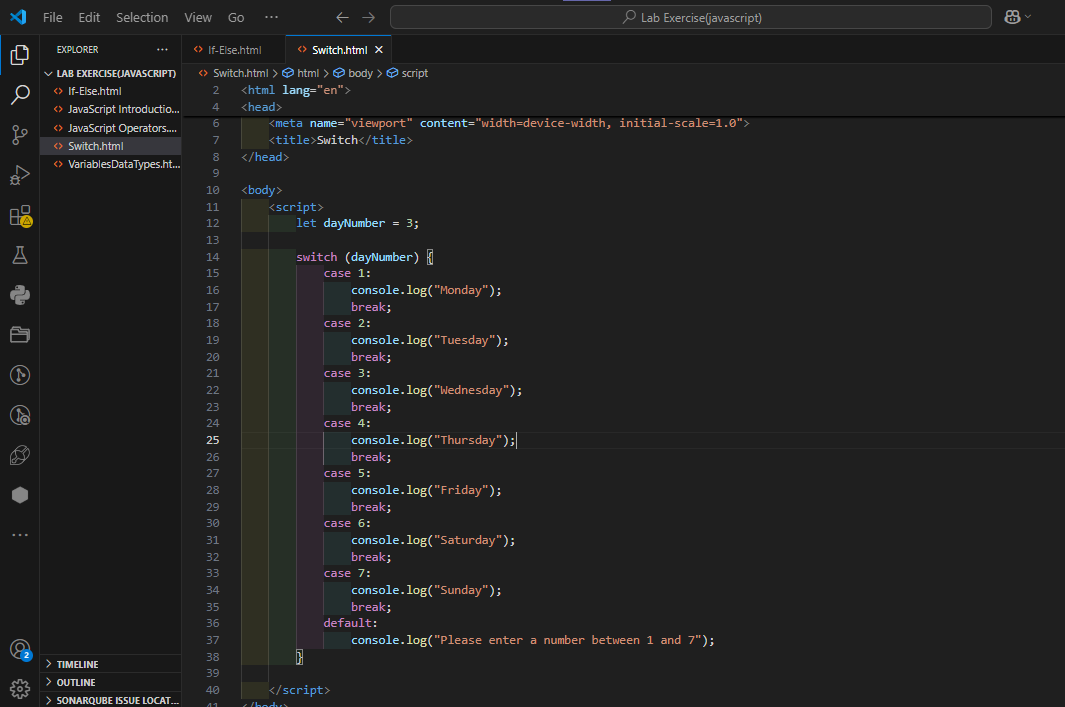
****

**Output:**

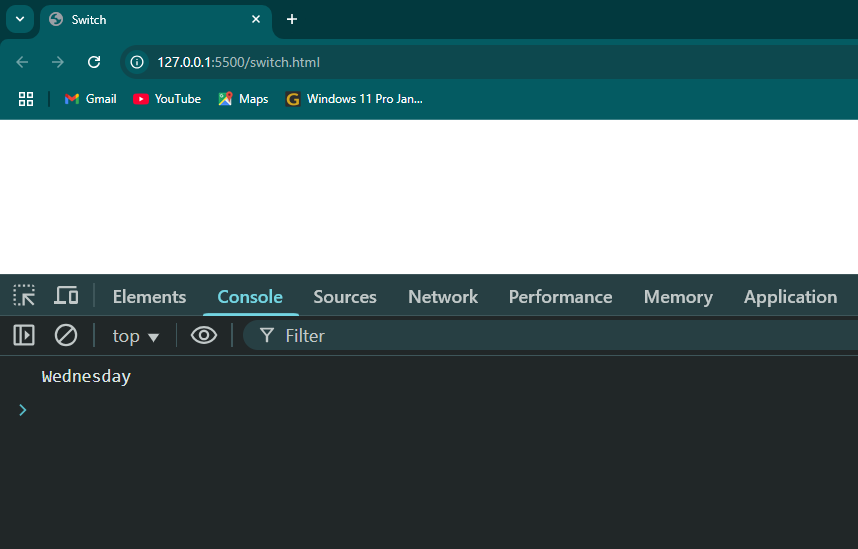
****

**Task 2 :** 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.).

**Code:**



**Output:**

****

Loops (For, While, Do-While)

Q1: Explain the different types of loops in JavaScript (for, while, do-while). Provide a basic example of each.

Ans: -

* **For Loop** : - The for loop through a block of code a number of times. In first intial value, second condition, and third increment or decrement after code executed.

Example:

for (let i = 0; i < 4; i++) {

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

}

* **While Loop** : - The while loop repeatedly executes a block of code as long as a specified condition is true.  loop first evaluates the condition inside “(condition)”. In first condition check after block of code executed.

Example:

let i = 0;

while (i < 6) {

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

i++;

}

* **Do-While Loop** : - The do…while loop executes the code inside “{ }”. If the condition evaluates to true, the code inside { } is executed again. In first block of code executed after condition check.

Example:

let i = 0;

do {

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

i++;

} while (i < 5);

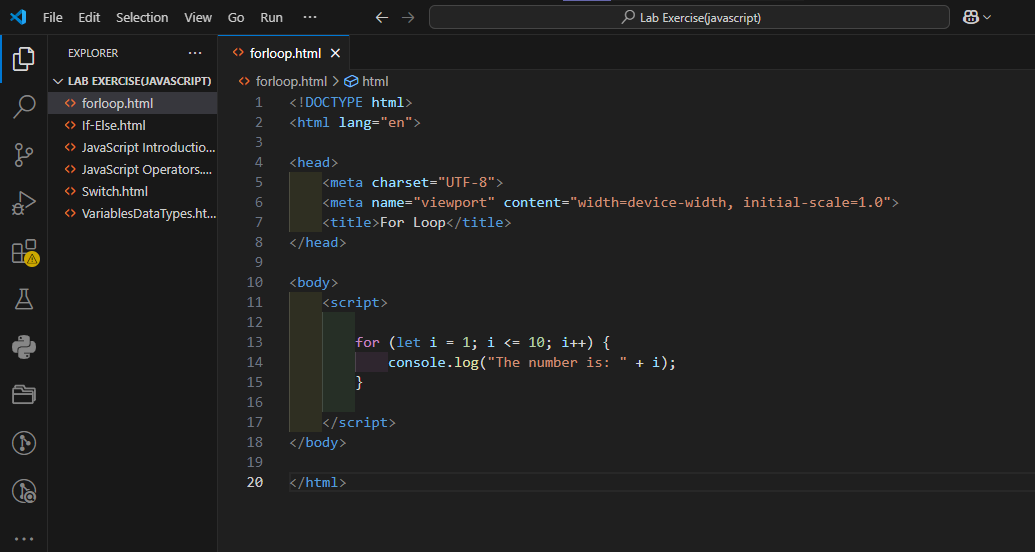
Q2: What is the difference between a while loop and a do-while loop?

Ans: - while loops check the condition before executing the loop body. If the condition is initially false, the loop body will never be executed. do-while loops execute the body once before checking the condition. First the loop body will be executed at least once, even if the condition is initially false.

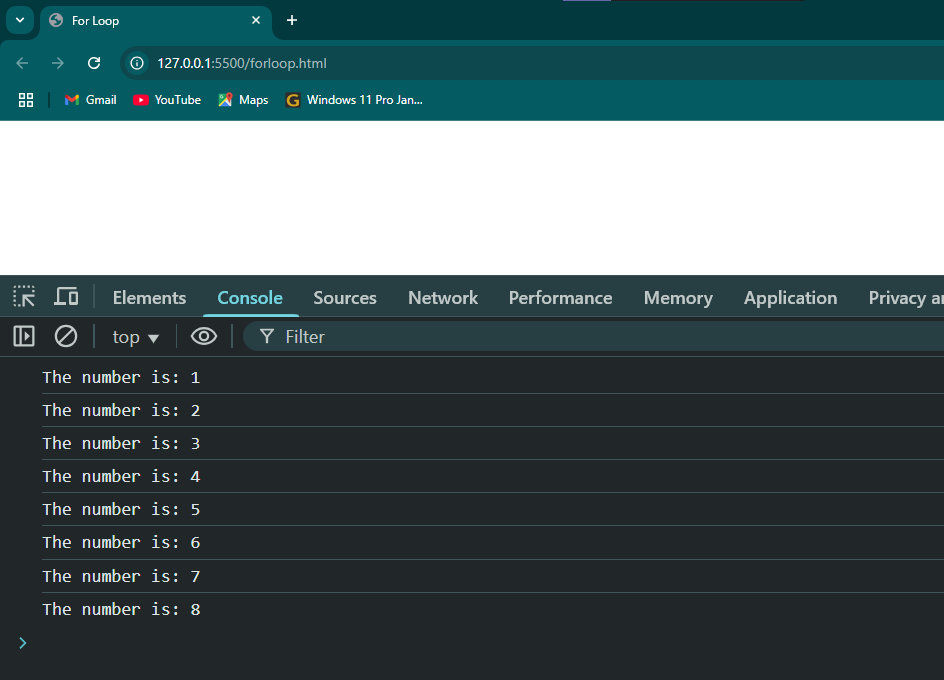
**Lab Tasks**

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

Code:

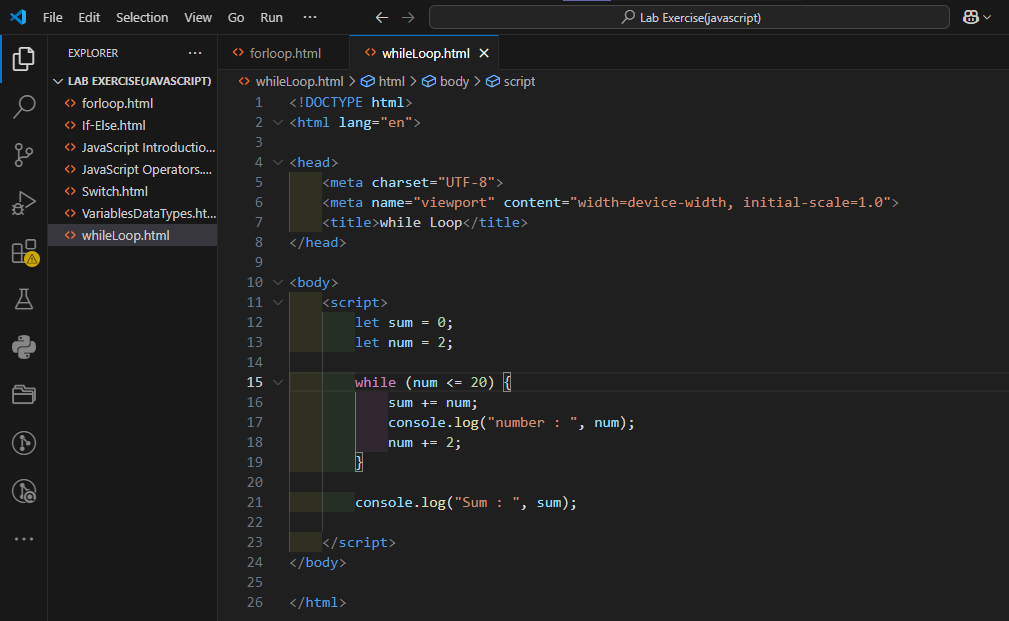


Output:

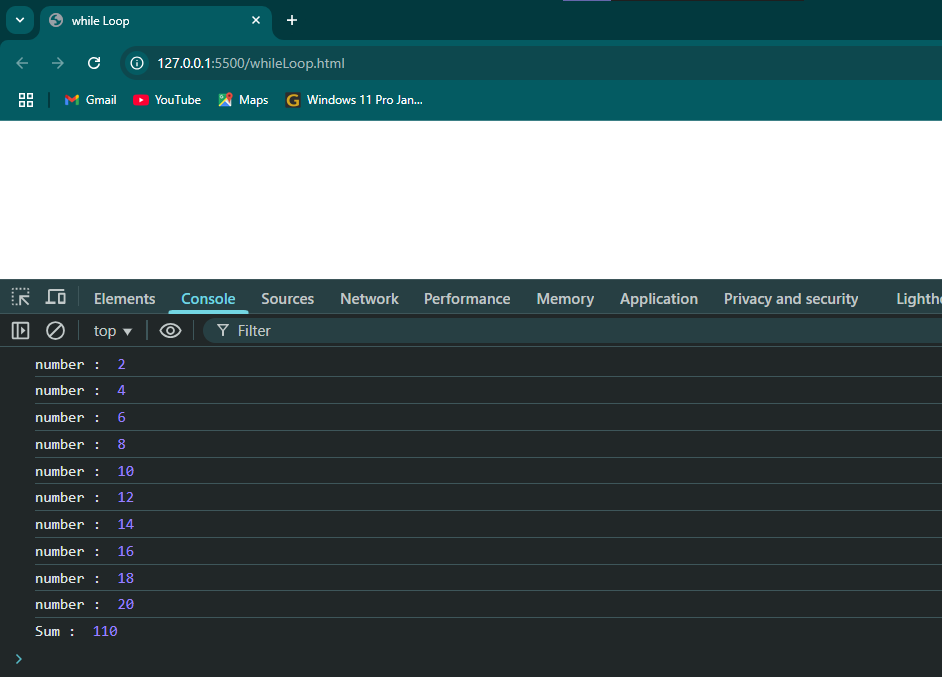


**Task 2 :** Create a JavaScript program that uses a while loop to sum all even numbers between 1 and 20.

Code:

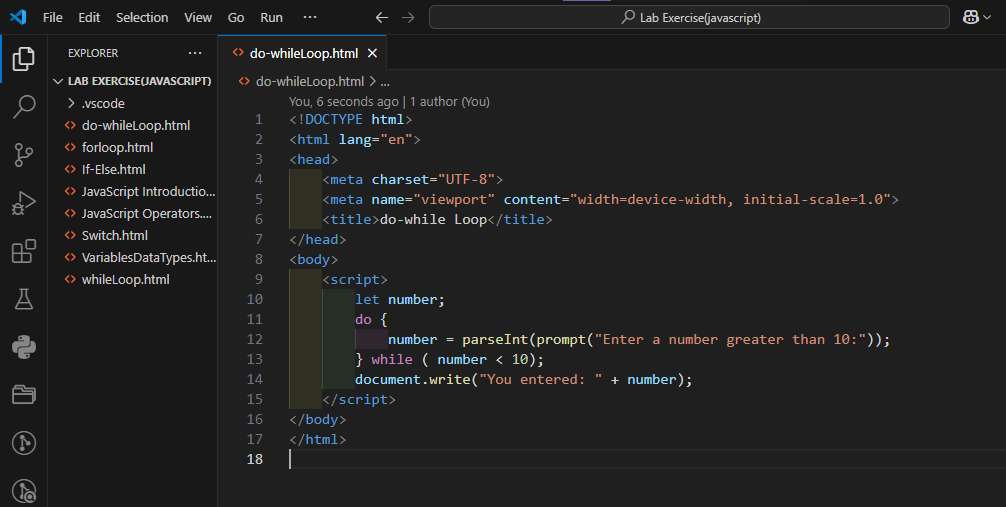


Output:

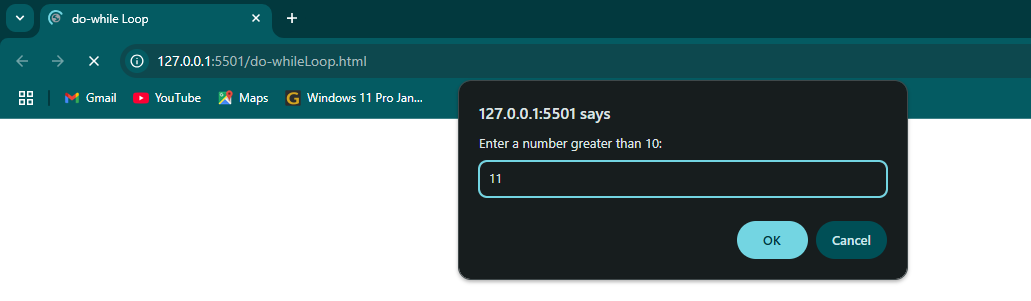


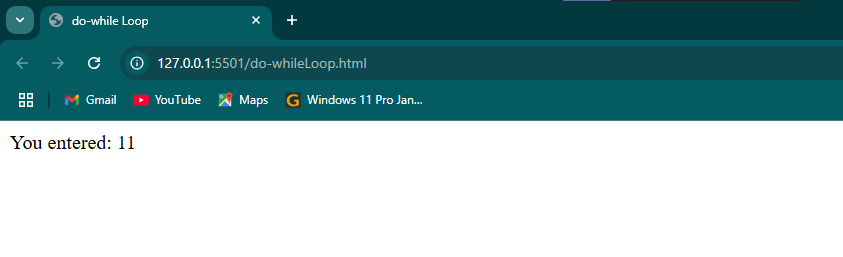
**Task 3 :** Write a do-while loop that continues to ask the user for input until they enter a number greater than 10.

Code:



Output:





Functions

Q1: What are functions in JavaScript? Explain the syntax for declaring and calling a function.

Ans: - A function is a small part of the program that does some work. We can write it once and use it many times.

Function declare:-

function greet() {

console.log("Hello, Kuldeep!");

}

Function calling :-

Code:- greet();

Output:- Hello, Kuldeep!

Q2: What is the difference between a function declaration and a function expression?

Ans: - Function declaration write a function using the function keyword with a name. It can be used even before it is written in the code. Function expression create a function and store it in a variable. It cannot be used before it is written.

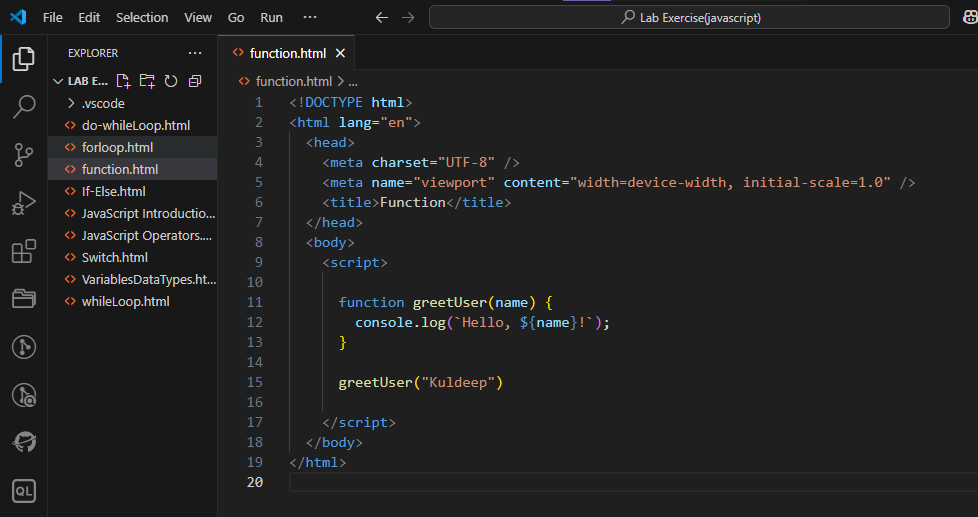
Q3: Discuss the concept of parameters and return values in functions.

Ans: - Parameters are like placeholders or input names in a function. When you create a function, you use parameters to tell the function what kind of information it needs to work.return value is the result that a function gives back after doing its job. We use the return keyword to send the result back to where the function was called.

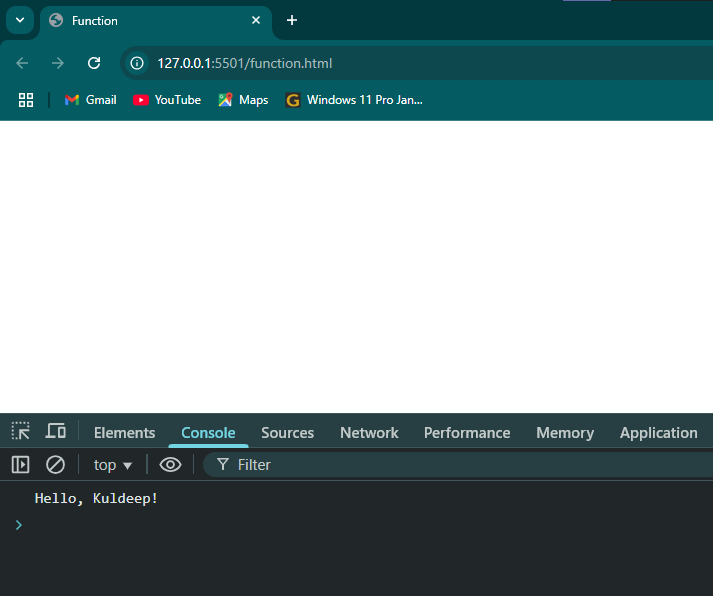
**Lab Tasks**

**Task 1 :** Write a function greetUser that accepts a user’s name as a parameter and displays a greeting message (e.g., "Hello, John!").

Code:

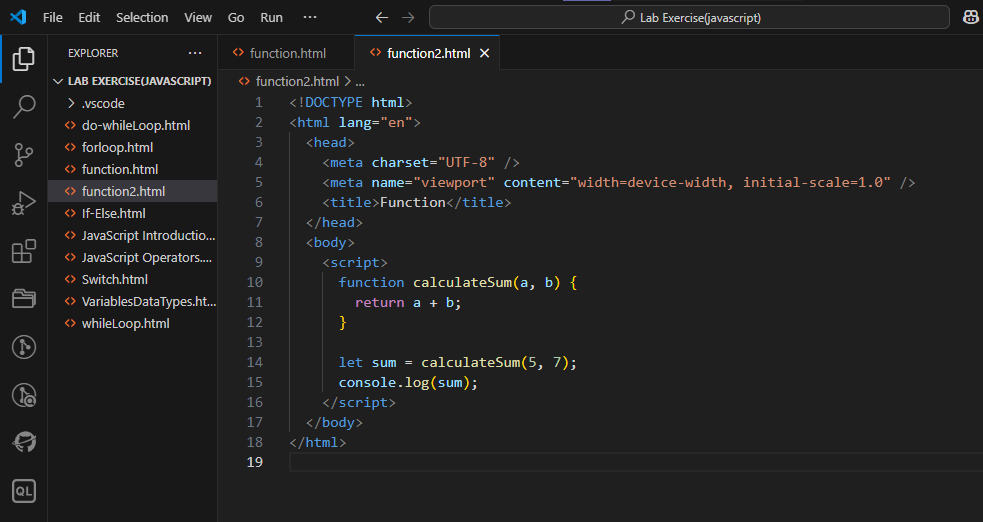


Output:

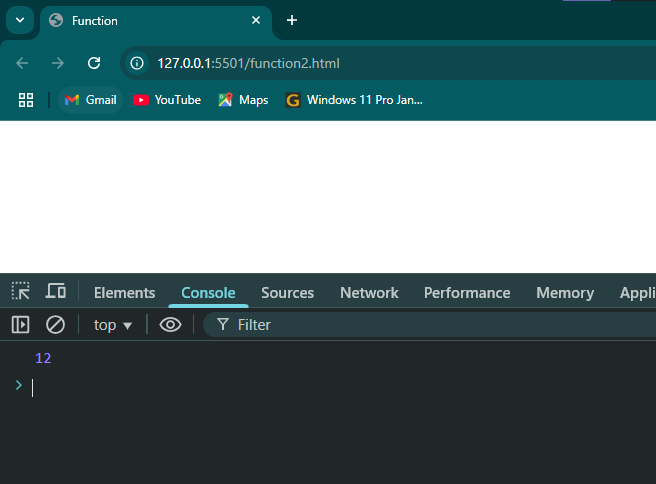


**Task 2 :** Create a JavaScript function that takes two numbers as parameters,adds them, and returns the result.

**Code:**

****

**Output:**

****

Q1: What is an array in JavaScript? How do you declare and initialize an array?

Ans: - Array store multiple values in a single variable. It can hold values of numbers, strings, objects. Arrays are indexed collections, meaning each item in the array has a numeric index, starting from 0.

Q2: Explain the methods push(), pop(), shift(), and unshift() used in arrays.

Ans: -

1. **push()**: Adds one or more elements to the end of an array.
   * **Syntax**: array.push(element1, element2, ...)
   * **Example**:

let fruits = ['apple', 'banana'];

fruits.push('cherry');

console.log(fruits); // ['apple', 'banana', 'cherry']

1. **pop():** Removes the last element from an array and returns it.
   * **Syntax**: array.pop()
   * **Example**:

let fruits = ['apple', 'banana', 'cherry'];

let removedFruit = fruits.pop();

console.log(removedFruit); // 'cherry'

console.log(fruits); // ['apple', 'banana']

1. **shift():** Removes the first element from an array and returns it.
   * **Syntax**: array.shift()
   * **Example**:

let fruits = ['apple', 'banana', 'cherry'];

let removedFruit = fruits.shift();

console.log(removedFruit); // 'apple'

console.log(fruits); // ['banana', 'cherry']

1. **unshift():** Adds one or more elements to the beginning of an array.
   * **Syntax**: array.unshift(element1, element2, ...)
   * **Example**:

let fruits = ['banana', 'cherry'];

fruits.unshift('apple');

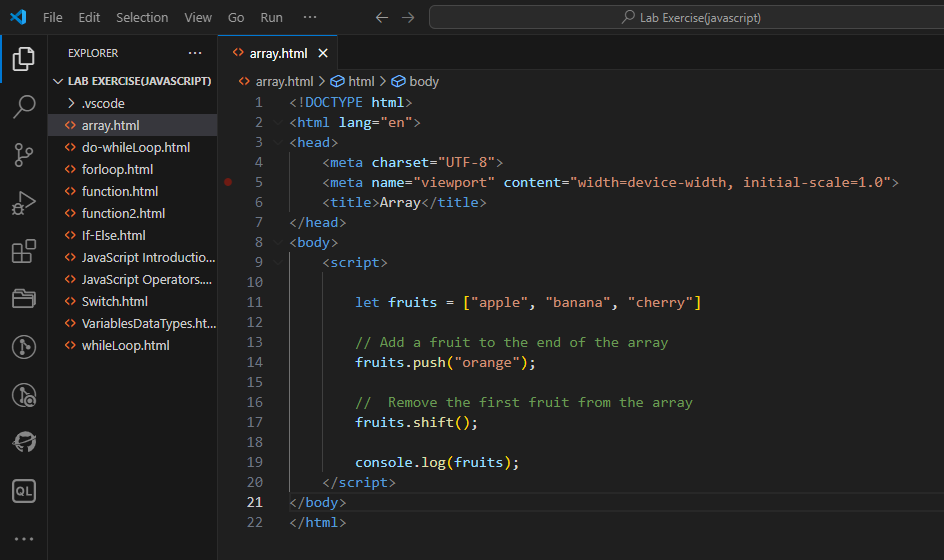
console.log(fruits); // ['apple', 'banana', 'cherry']

**Lab Tasks**

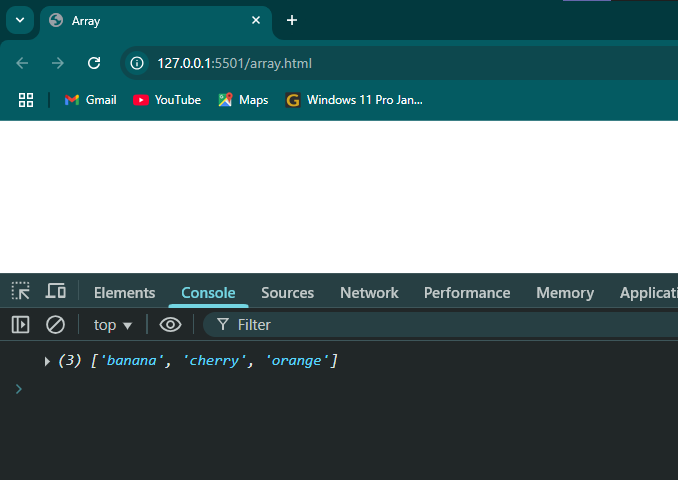
**Task 1 :** Declare an array of fruits (["apple", "banana", "cherry"]).

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.

Code:

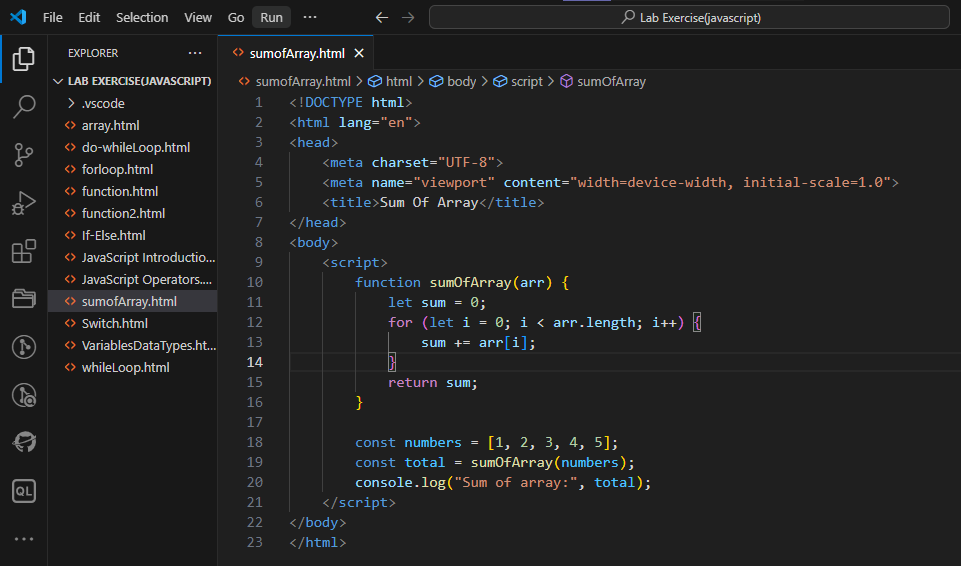


Output:

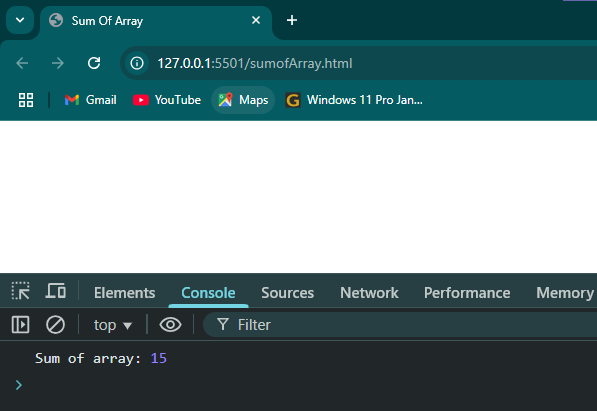


**Task 2 :** Write a program to find the sum of all elements in an array of numbers.

Code:



Output:



Objects

Q1: What is an object in JavaScript? How are objects different from arrays?

Ans: - An object is a complex data type that allows you to store collections of data using key-value pairs. An array is a special type of object used for ordered collections of data.

Q2: Explain how to access and update object properties using dot notation and bracket notation.

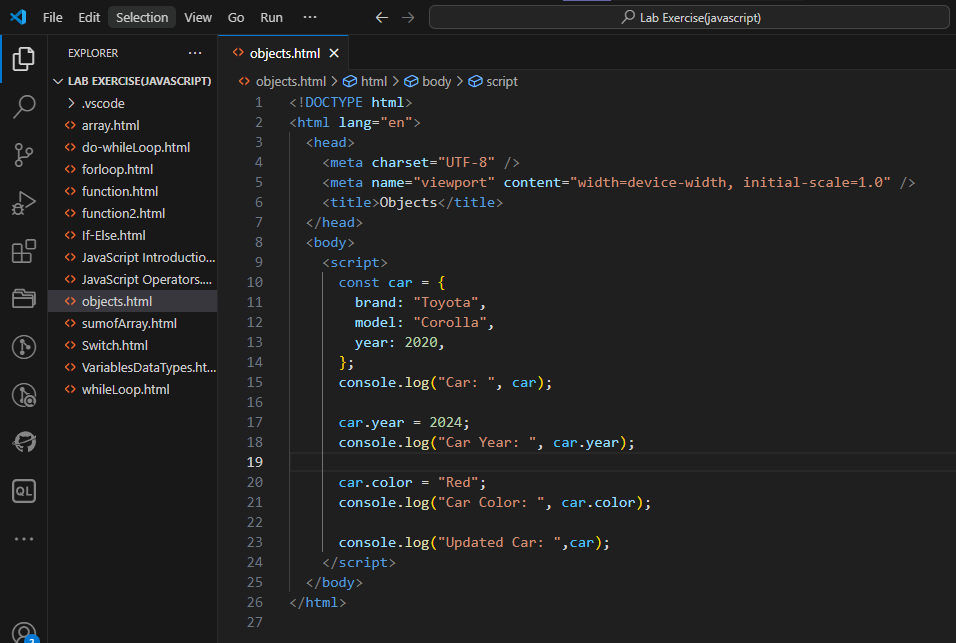
Ans: - you can access and update object properties using dot notation and bracket notation. Both works, but they shine in different situations.

1. dot notation :- (Syntax: object.property)
2. Bracket Notation :- (Syntax: object["property"])

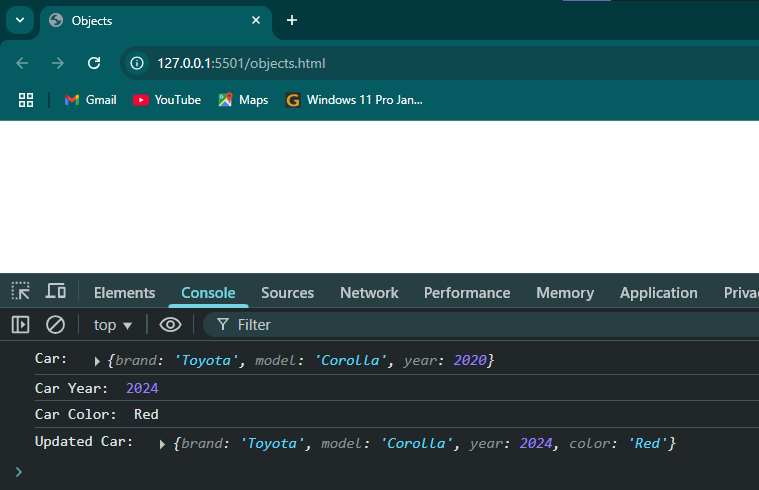
**Lab Tasks**

**Task 1 :** Create a JavaScript object car with properties brand, model, and year. UseJavaScript to: Access and print the car’s brand and model. Update the year property. Add a new property color to the car object.

Code:



Output:



JavaScript Events

Q1: What are JavaScript events? Explain the role of event listeners.

Ans: - JavaScript events are things that happen on a webpage — like when you click a button, move your mouse, press a key, or load the page. An event listener is a function that waits for a specific event to happen on a specific element.

Q2: How does the addEventListener() method work in JavaScript? Provide an example.

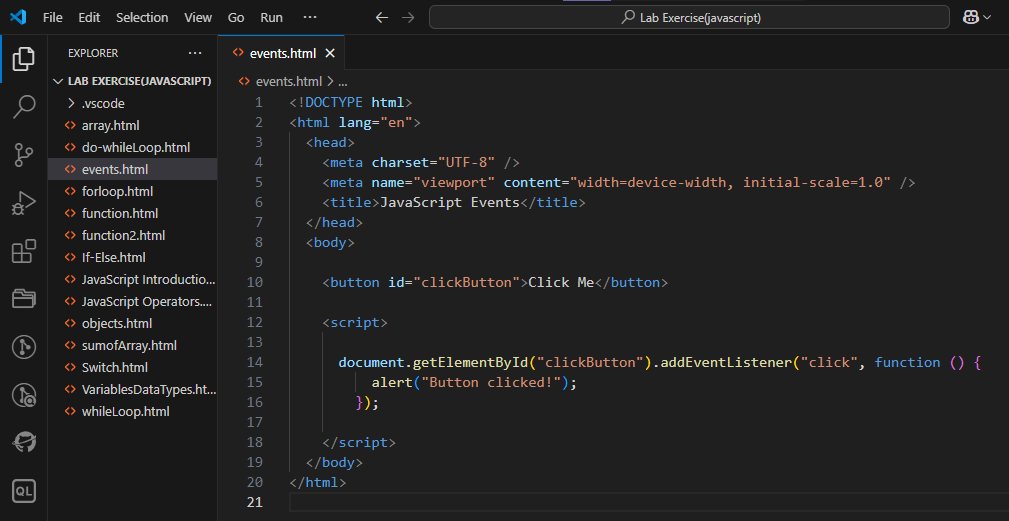
Ans: - An addEventListener() is a method used to listen for events (like clicks, key presses, etc.) on HTML elements.

Syntax: element.addEventListener(eventType, functionToRun);

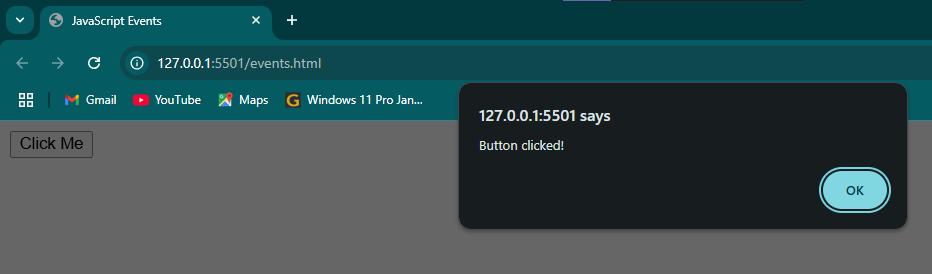
**Lab Tasks**

**Task 1 :** Create a simple webpage with a button that, when clicked, displays an alert saying "Button clicked!" using JavaScript event listeners.

Code:



Output:



DOM Manipulation

Q1: What is the DOM (Document Object Model) in JavaScript? How does JavaScript interact with the DOM?

Ans: - The DOM (Document Object Model) is a programming interface for web documents. It represents the structure of an HTML or XML document as a tree of objects. JavaScript interacts with the DOM to make web pages interactive and dynamic.

Q2: Explain the methods getElementById(), getElementsByClassName(),and querySelector() used to select elements from the DOM.

Ans: -

1. getElementById() :- Selects a single element by its id.

Syntax: document.getElementById("id");

1. getElementsByClassName() :- Selects all elements with the given class name.

Syntax: document.getElementsByClassName("class");

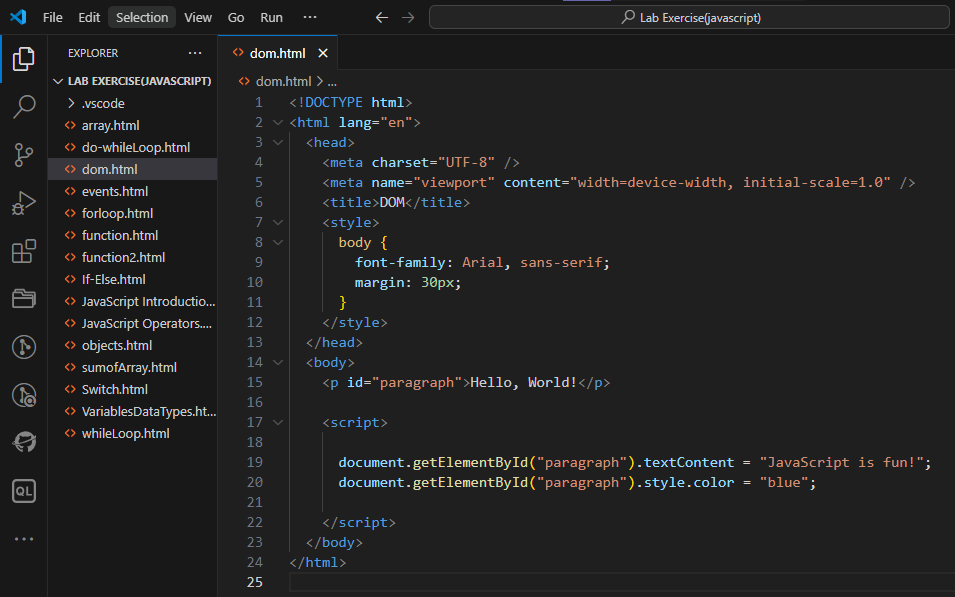
1. querySelector() :- Selects the first matching element based on a CSS selector.

Syntax: document.querySelector("selector");

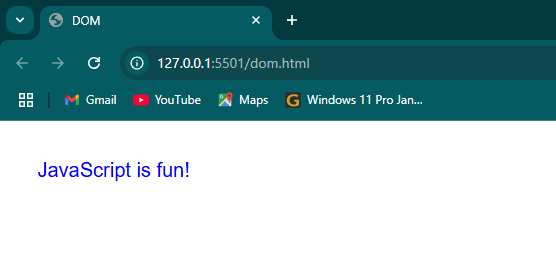
**Lab Tasks**

**Task 1 :** Create an HTML page with a paragraph (<p>) that displays "Hello, World!". Use JavaScript to: Change the text inside the paragraph to "JavaScript is fun!". Change the color of the paragraph to blue.

Code:



Output:



JavaScript Timing Events (setTimeout, setInterval)

Q1: Explain the setTimeout() and setInterval() functions in JavaScript. How are they used for timing events?

Ans: - setTimeout() and setInterval() are used to schedule functions to run after a certain period of time.

1. setTimeout() :- is used to execute a function or code once after a specified delay (in milliseconds).

Syntax: setTimeout(function, delay);

Example:

setTimeout(() => {

console.log('Hello, World!');

}, 2000);

1. setInterval() :- is used to repeatedly execute a function or code at specified intervals (in milliseconds).

Syntax: setInterval(function, interval);

Example:

setInterval(() => {

console.log('repeat every 2 seconds');

}, 2000);

Q2: Provide an example of how to use setTimeout() to delay an action by 2 seconds.

Ans: -

Example:- console.log('Action will happen in 2 seconds...');

setTimeout(() => {

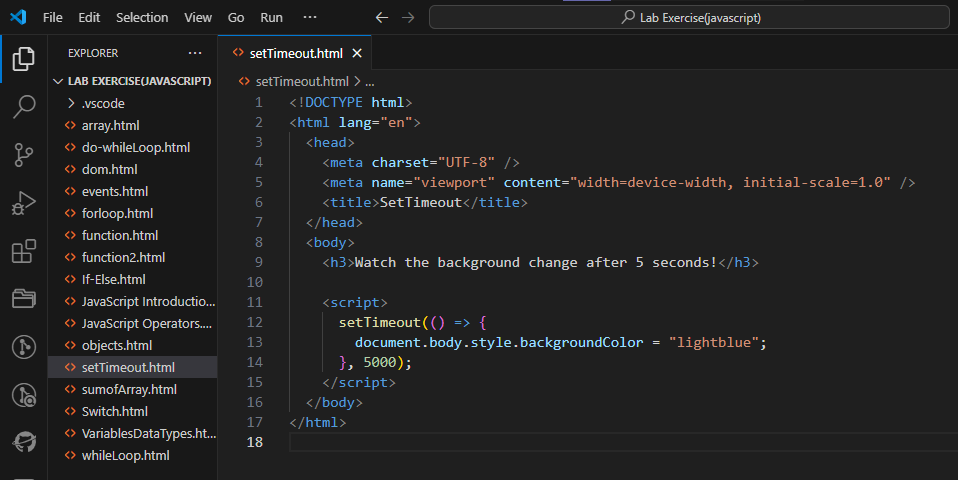
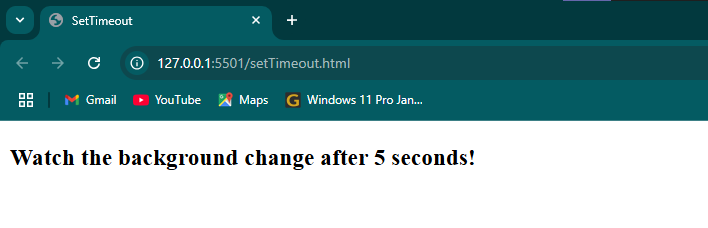
console.log('This is the delayed action after 2 seconds!');

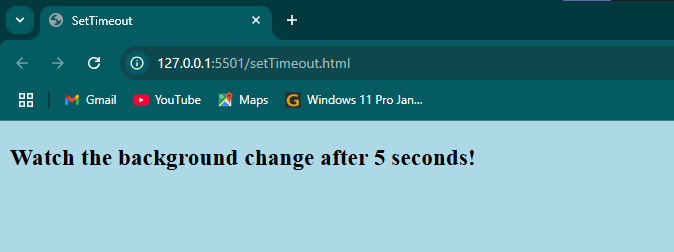
}, 2000);

**Lab Tasks**

**Task 1 :** Write a program that changes the background color of a webpage after 5 seconds using setTimeout().

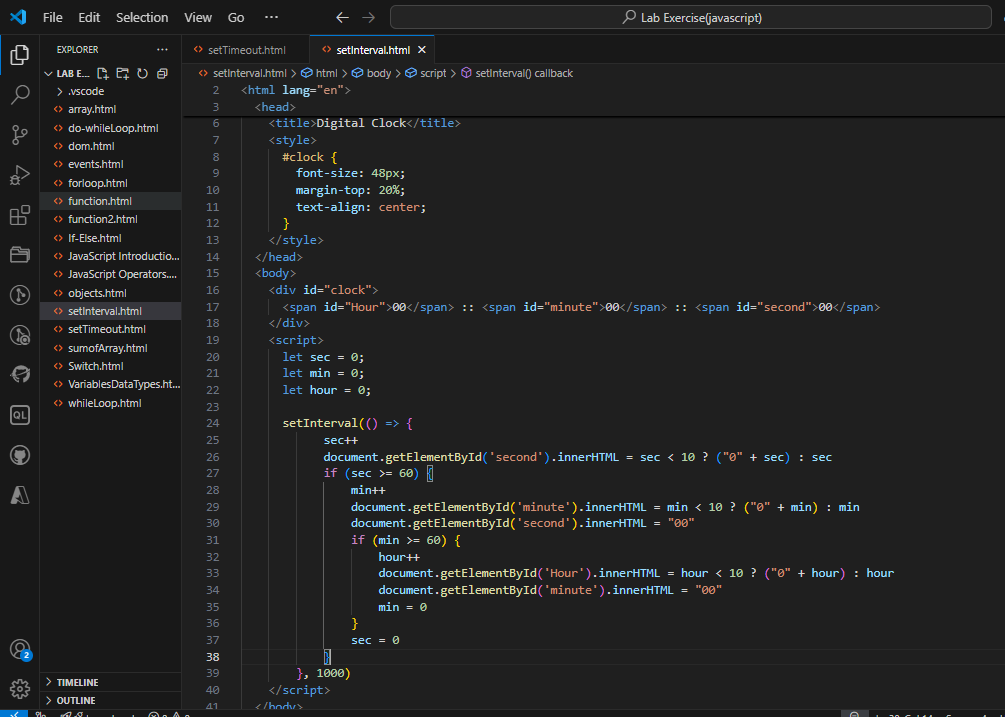
Code:

Output: 

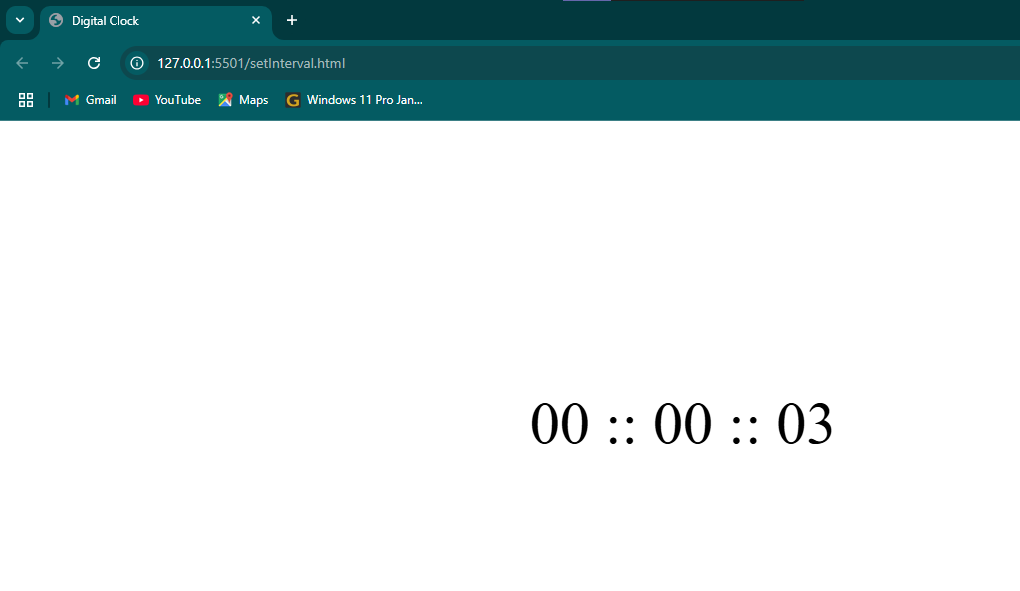


**Task 2 :** Create a digital clock that updates every second using setInterval().

Code:



Output:



JavaScript Error Handling

Q1: What is error handling in JavaScript? Explain the try, catch, and finally blocks with an example.

Ans: - Error handling in JavaScript is a way to deal with problems that might happen when your code is running. Instead of your program crashing, it allows you to handle the problem and keep it running.

1. **try block:** put the code that might cause an error inside the try block. If something goes wrong, JavaScript jumps to the next block.
2. **catch block:** This block catches the error and lets you handle it, like showing an error message or fixing the problem.
3. **finally block:** This block runs no matter what, even if there’s an error. It’s usually used to clean up, like closing a file or releasing a resource.

Example: -

try {

let num = 10;

let result = num / 0;

console.log(result);

} catch (error) {

console.log("Oops! Something went wrong:", error);

} finally {

console.log("This will always run.");

}

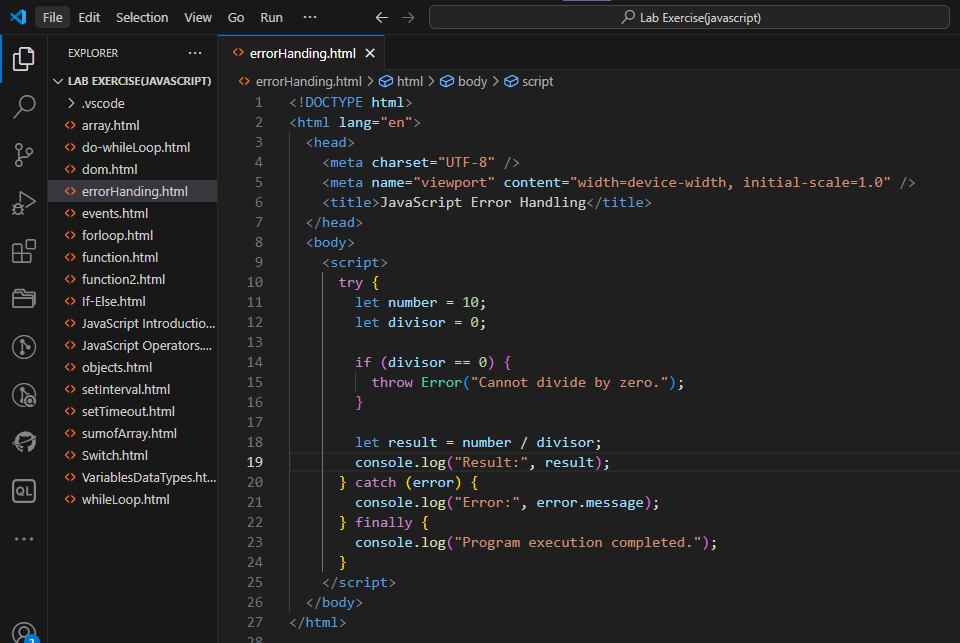
Q2: Why is error handling important in JavaScript applications?

Ans: - Error handling is very important in JavaScript applications because it helps your app run smoothly and avoid crashing when something goes wrong.

**Lab Tasks**

**Task 1 :** Write a JavaScript program that attempts to divide a number by zero. Use try- catch to handle the error and display an appropriate error message.

Code:



Output:

