JavaScript Assignment:

Theory questions:

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

Ans: java Script is high level interpreted programming language used to make web pages interactive and dynamic.

It allows developers to:

Validate user input(like checking forms)

Create dynamic effects (like sliders, popups)

Interact with APIs and servers

Modify html and css content dynamically

Html = structure

Css = styling

javaScript = behavior

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

Ans:

Feature	JavaScript	Python	Java
Execution	Runs in browsers	Runs on server	Runs on JVM
Typing	Dynamic	Dynamic	Static
Syntax	Similar to C	Indentation-based	Class-based
Use	Web apps, frontend/backend	Data science, AI, backend	Enterprise apps
Compilation	Interpreted	Interpreted	Compiled to bytecode

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

Ans: The <script> tag is used to embed or reference JavaScript code in an HTML document.

Inline script example:

```
<script>
  alert("Hello from inline JavaScript!");
</script>
```

External script example:

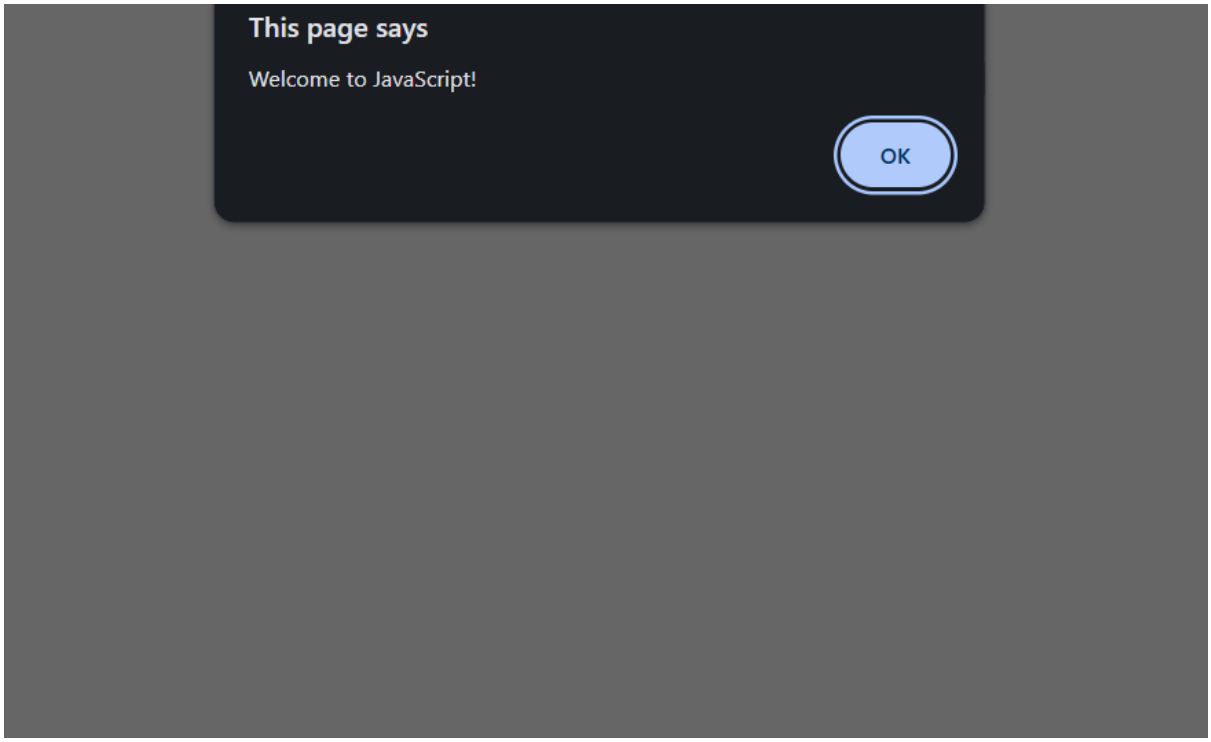
```
<script src="script.js"></script>
```

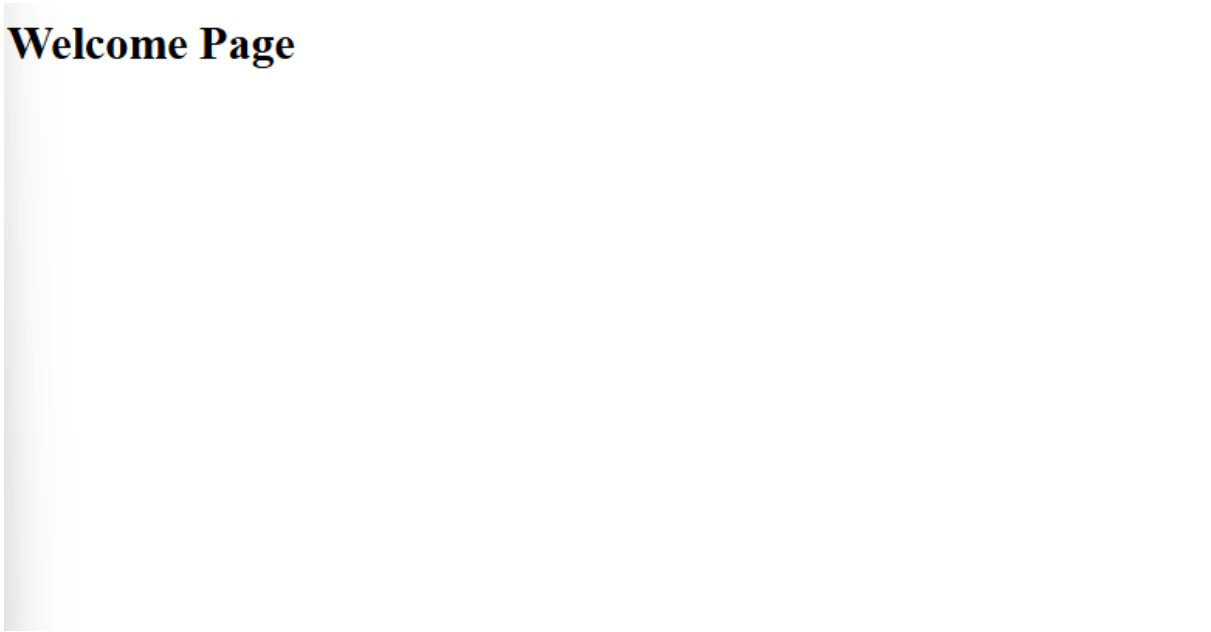
Lab Assignment

- **Task:**
- **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.**

```
<!DOCTYPE html>
```

```
<html>
<head>
  <title>JavaScript Intro</title>
</head>
<body>
  <h2>Welcome Page</h2>
  <script>
    alert("Welcome to JavaScript!");
  </script>
</body>
</html>
```





Module 2: Variables and data types:

Theory assignment:

Q1. What are variables in JavaScript?

Variables are containers used to store data values.

Keyword	Scope	Reassign	Redeclare
var	Function	yes	yes
let	Block	yes	no
const	Block	no	no

Example:

```
var name = "John";
let age = 25;
const country = "India";
```

Q2. Explain the different data types in JavaScript.

Type	Example
String	"Hello"
Number	25
Boolean	true
Null	null

Type	Example
Undefined	let x;
Object	{name:"John", age:25}
Array	["apple","banana"]

Q3. Difference between undefined and null

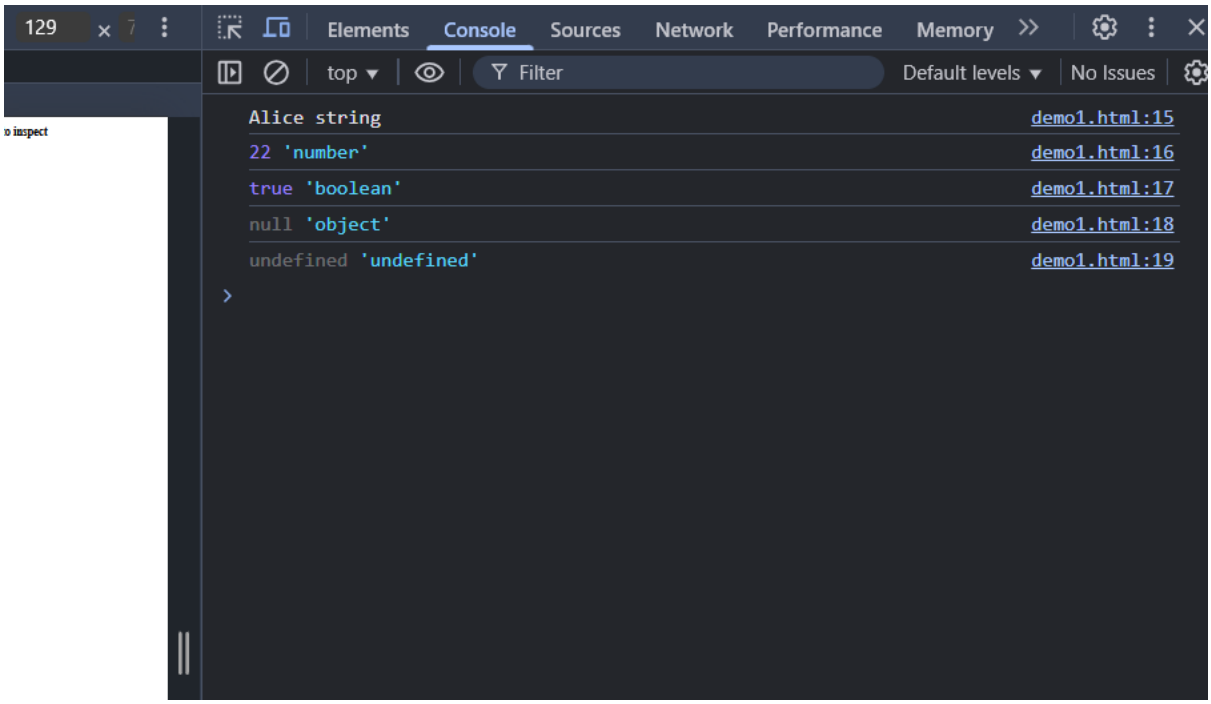
Type	Description
undefined	Variable declared but not assigned a value
null	Represents an intentionally empty or unknown value

Lab Assignment • Task: • Write a JavaScript program to declare variables for different data

types (string,number, boolean, null, and undefined). <!DOCTYPE html>

```
<html>
<head>
  <title>Variables Example</title>
</head>
<body>
  <script>
    let name = "Alice";
    let age = 22;
    let isStudent = true;
    let emptyValue = null;
    let notAssigned;

    console.log(name, typeof name);
    console.log(age, typeof age);
    console.log(isStudent, typeof isStudent);
    console.log(emptyValue, typeof emptyValue);
    console.log(notAssigned, typeof notAssigned);
```

```
  </script>
</body>
</html>
```



Module 3: JavaScript operators:

Theory Assignment

Q1. Types of operators

1. **Arithmetic:** + - * / %
2. **Assignment:** = += -=
3. **Comparison:** == === != < > <= >=

4. **Logical:** && || !

Q2. Difference between == and ===

- == → Compares values only
- === → Compares **values + data types**

Example:

5 == "5"   // true

5 === "5" // false

Lab Assignment

• Task:

• 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 onenumber is greater than the other.

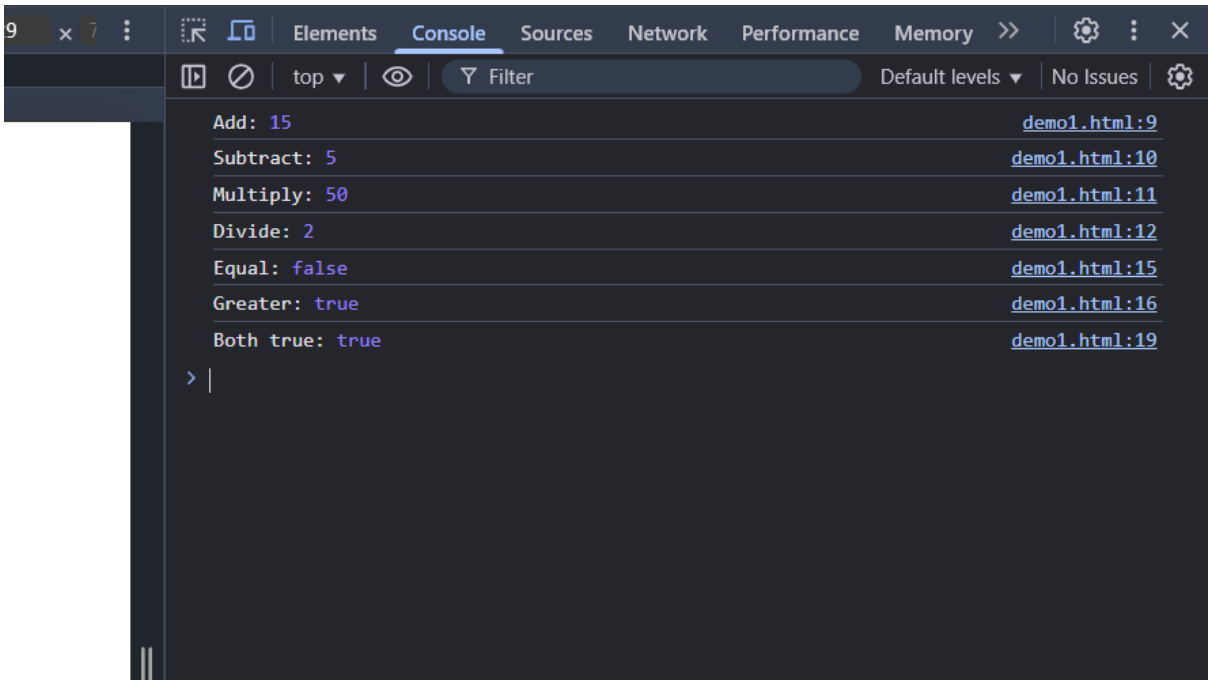
• Use logical operators to check if both conditions (e.g., a > 10and b < 5)are true

```
<!DOCTYPE html>
<html>
<head><title>Operators Example</title></head>
<body>
<script>
let a = 10, b = 5;

// Arithmetic
console.log("Add:", a + b);
console.log("Subtract:", a - b);
console.log("Multiply:", a * b);
console.log("Divide:", a / b);

// Comparison
console.log("Equal:", a == b);
console.log("Greater:", a > b);

// Logical
console.log("Both true:", a > 5 && b < 10);
```

</script>

</body>

</html>



Module 4: Control flow

Theory Assignment

Q1. What is control flow?

Control flow decides **which code block runs based on conditions.**

if-else example:

let num = 5;

```
if(num > 0) console.log("Positive");
else if(num < 0) console.log("Negative");
else console.log("Zero");
```

Q2. switch statement

Used when comparing one value to multiple possible cases.

```
switch(day) {
  case 1: console.log("Monday"); break;
  case 2: console.log("Tuesday"); break;
  default: console.log("Invalid day");
}
```

Lab Assignment

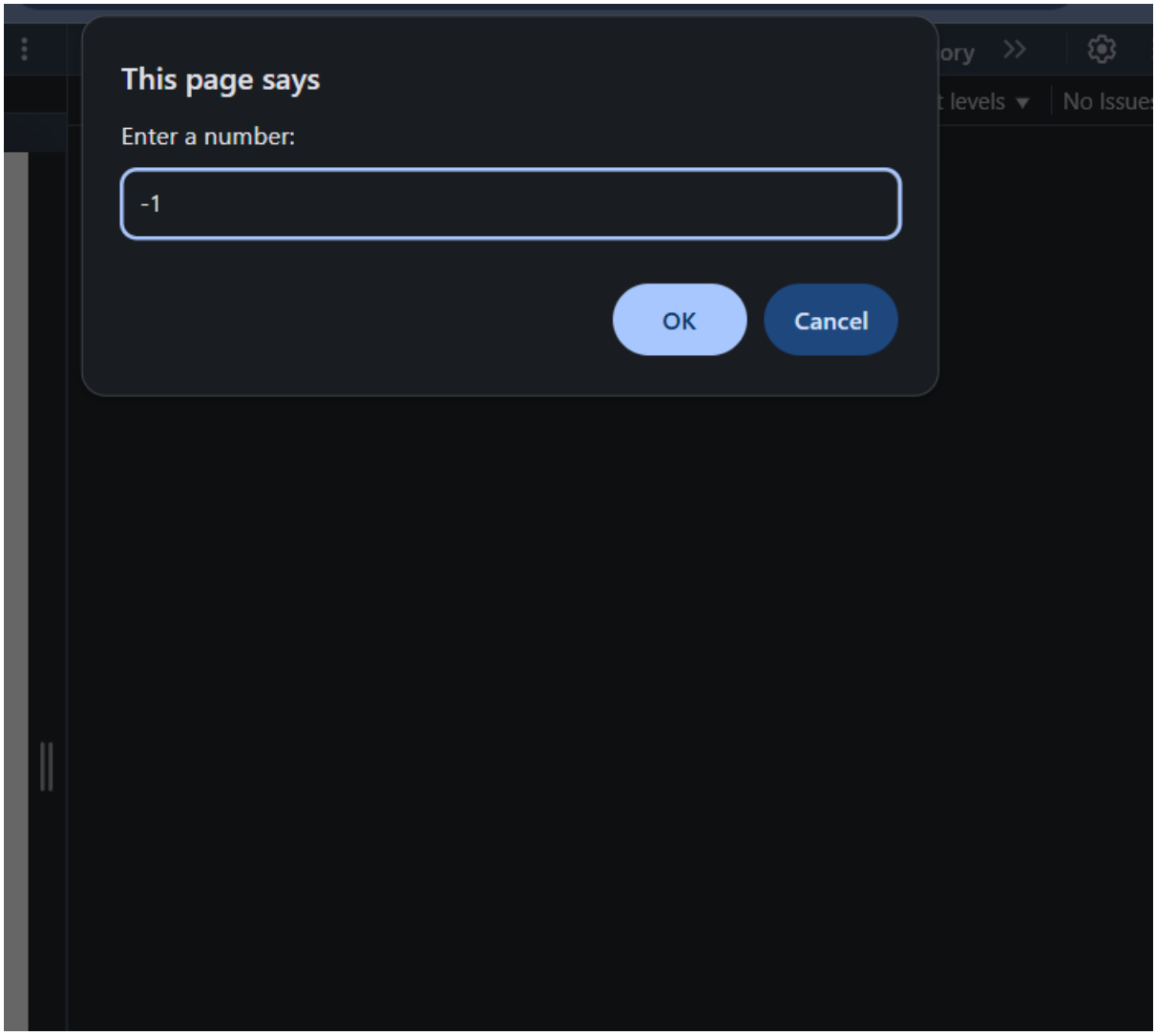
- Task 1:
- Write a JavaScript program to check if a number is positive, negative, or zero usingan if-elsestatement.
- Task 2:

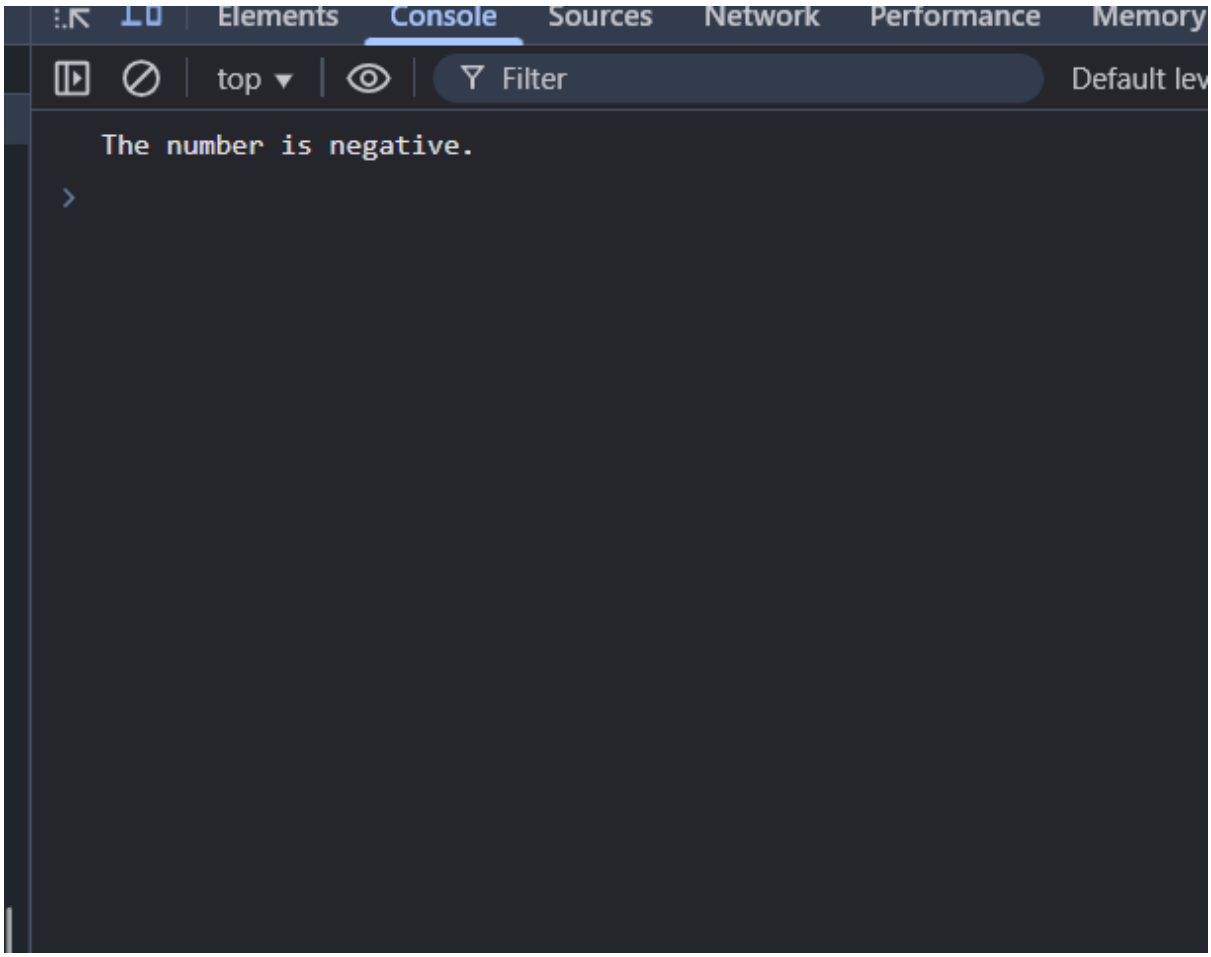
• Create a JavaScript program using a switchstatement to display the day of the week based on the user input (e.g., 1 for Monday, 2 for Tuesday, etc.).

Task 1:

```
<!DOCTYPE html>
<html>
<body>
<script>
let num = parseInt(prompt("Enter a number:"));

if (num > 0) {
  console.log("The number is positive.");
} else if (num < 0) {
  console.log("The number is negative.");
} else {
  console.log("The number is zero.");
}
```

```
</script>
</body>
</html>
```

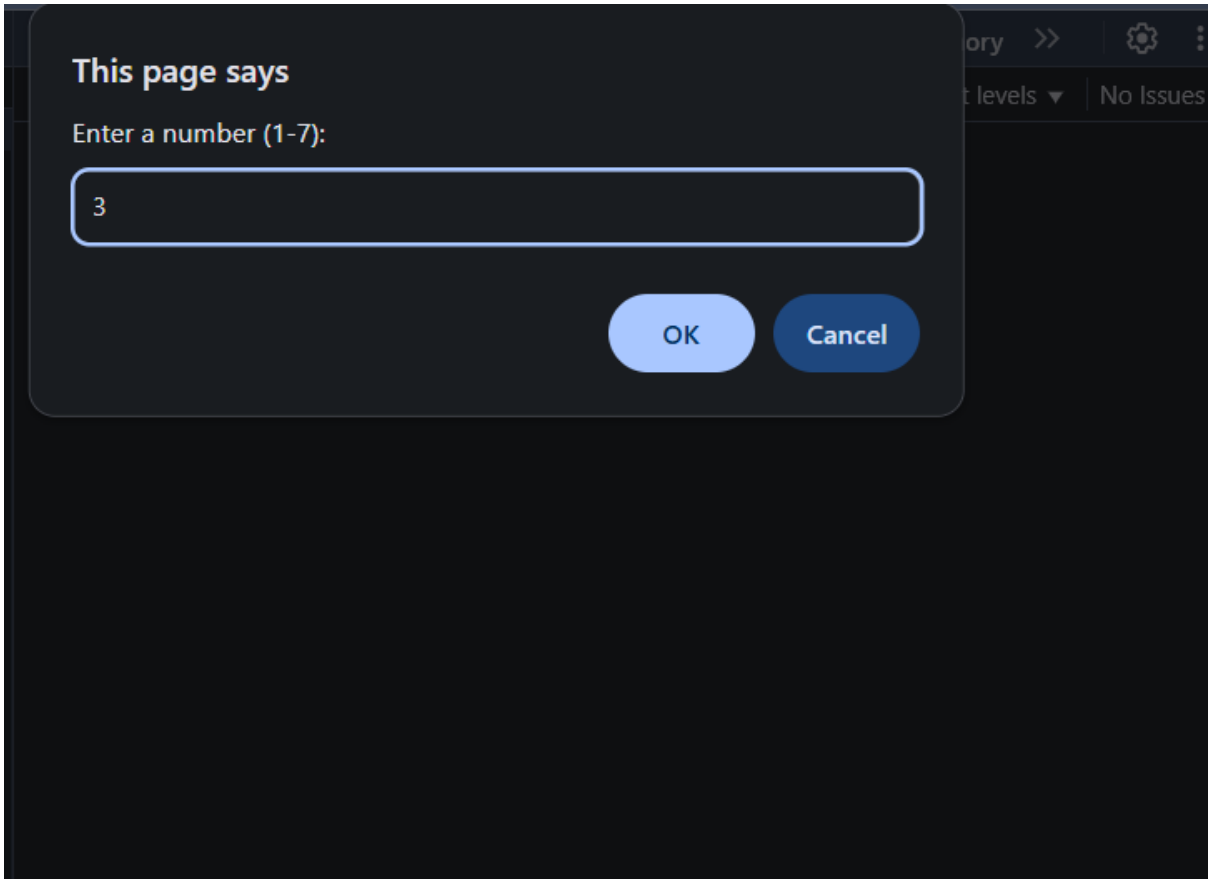


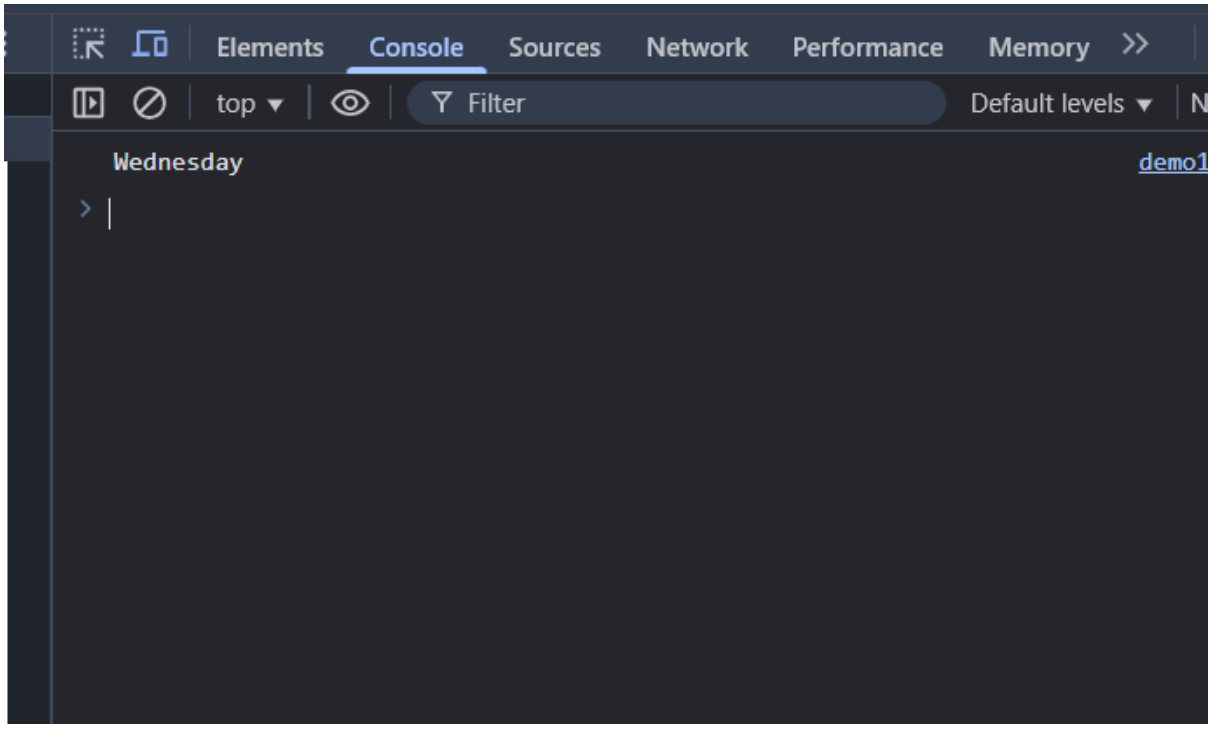


Task 2:

```
<!DOCTYPE html>
<html>
<body>
<script>
let day = parseInt(prompt("Enter a number (1-7):"));
```

```
let dayName;


switch (day) {
  case 1: dayName = "Monday"; break;
  case 2: dayName = "Tuesday"; break;
  case 3: dayName = "Wednesday"; break;
  case 4: dayName = "Thursday"; break;
  case 5: dayName = "Friday"; break;
  case 6: dayName = "Saturday"; break;
  case 7: dayName = "Sunday"; break;
  default: dayName = "Invalid input!";
}
console.log(dayName);
</script>
</body>
</html>
```


This page says
Enter a number (1-7):
3
OK
Cancel


Elements
Console
Sources
Network
Performance
Memory
top
Filter
Default levels
Wednesday
demo1

Module 5: 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.

Ans:

1. **for loop** – runs a block a specific number of times.

   for (let i = 1; i <= 5; i++) console.log(i);

2. **while loop** – runs as long as condition is true.

   let i = 1;

   while (i <= 5) { console.log(i); i++; }

3. **do-while loop** – runs once, then repeats while condition is true.

   let i = 1;

   do { console.log(i); i++; } while (i <= 5);

• Question 2: What is the difference between a whileloop and a do-whileloop?

Ans: **Difference:**

- while checks the condition **before** running.
- do-while runs **at least once**, even if condition is false.

Lab Assignment

• Task 1:

• Write a JavaScript program using a forloop to print numbers from 1 to 10.

• Task 2:

• Create a JavaScript program that uses a whileloop to sum all even numbers between 1 and 20.

• Task 3:

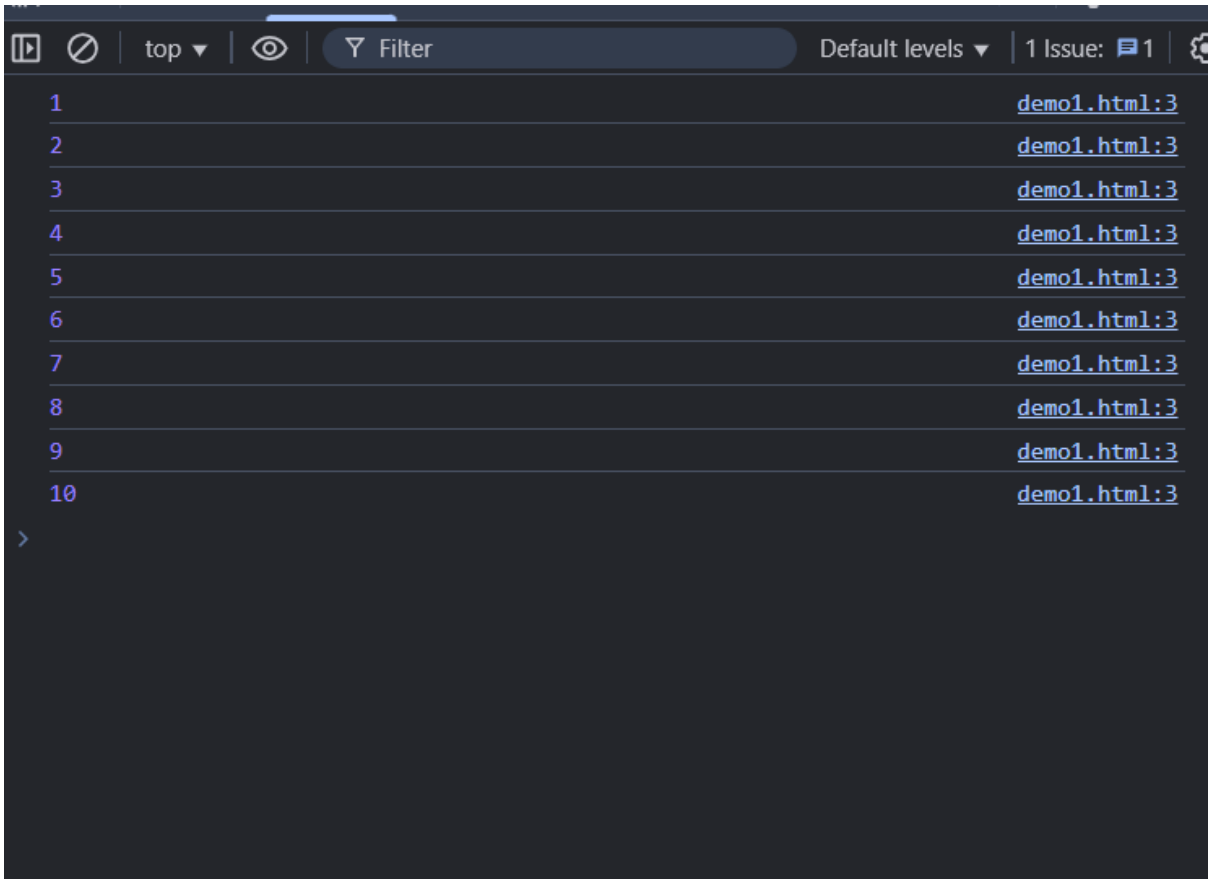
• Write a do-whileloop that continues to ask the user for input until they enter a number greater than 10

Task 1:

<script>

for (let i = 1; i <= 10; i++) {

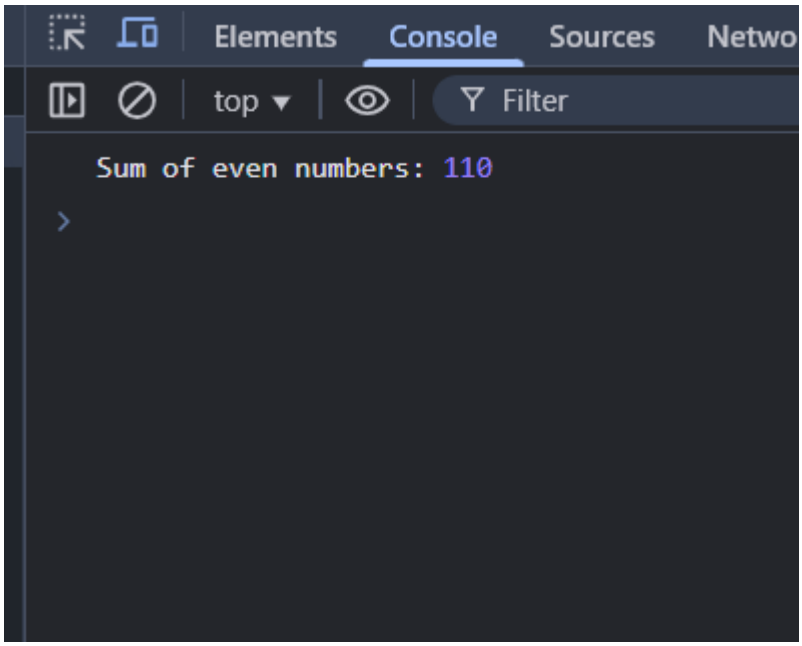
```
  console.log(i);
}
</script>
```



Task 2:

```
<script>
let i = 1, sum = 0;
while (i <= 20) {
```

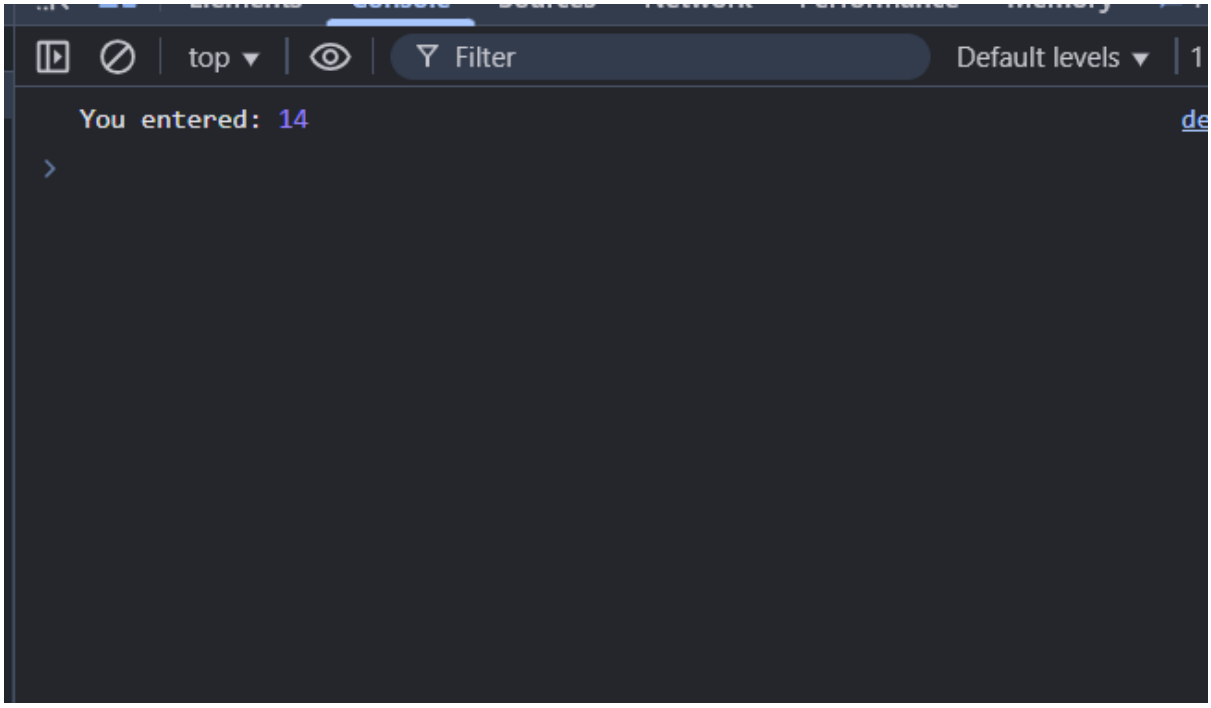
```
  if (i % 2 === 0) sum += i;
  i++;
}
console.log("Sum of even numbers:", sum);
</script>
```

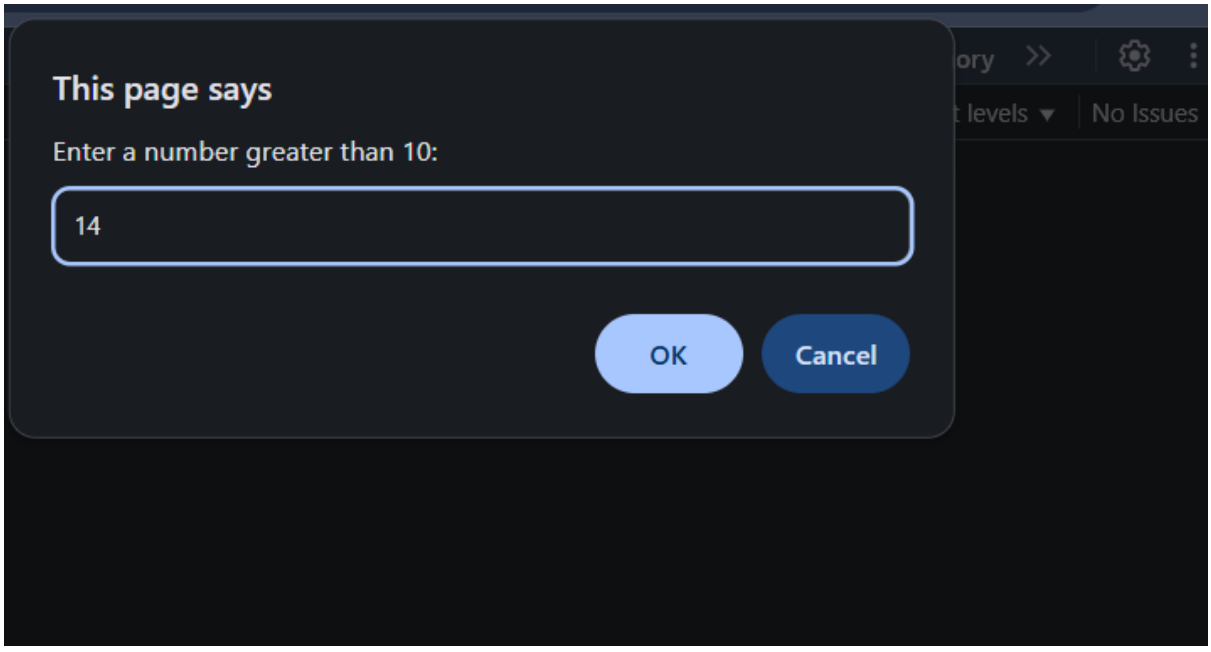


Task 3:

```
<script>
let num;
do {
```

```
  num = parseInt(prompt("Enter a number greater than 10:"));
} while (num <= 10);
console.log("You entered:", num);
</script>
```





Module 6 : functions:

Theory

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

Functions are reusable blocks of code.

Ans:

Syntax:

```
function name(param1, param2) {
  return param1 + param2;
}
```

Call using name(arg1, arg2).

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

- **Function Declaration:** hoisted (can be called before defined)
- function add(a, b) { return a + b; }
- **Function Expression:** stored in variable, not hoisted
- const add = function(a, b) { return a + b; }

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

- **Parameters**: placeholders in function definition.
- **Return Value**: value sent back using return.

Lab Assignment

• Task 1:

• Write a function greetUserthat accepts a user's name as a parameter and displaysa greeting message (e.g., "Hello, John!").

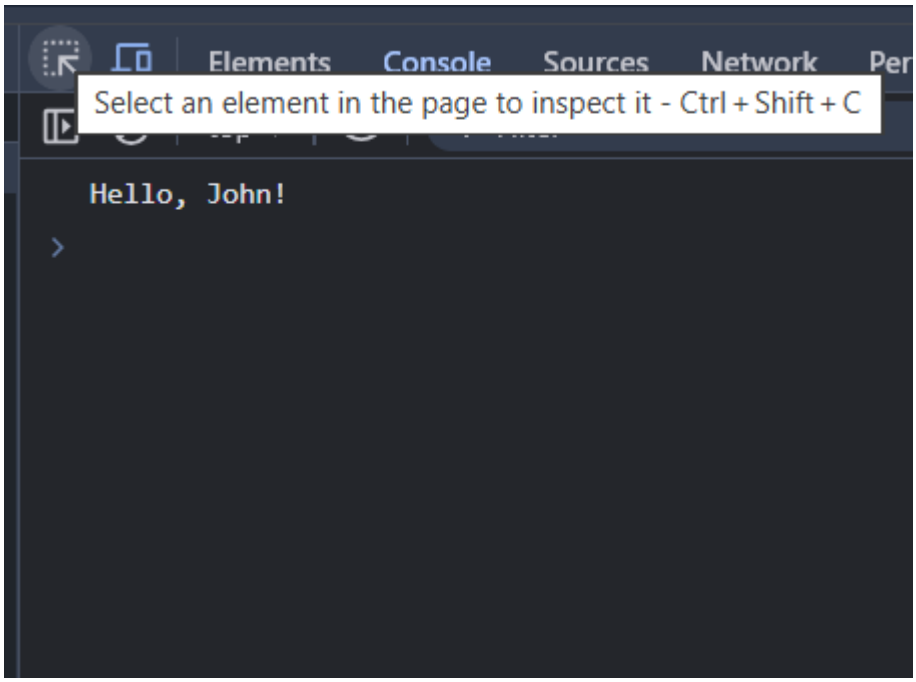
• Task 2:

• Create a JavaScript function calculateSumthat takes two numbers as parameters, adds them, and returns the result.

Task 1:

```
<script>
function greetUser(name) {
  console.log("Hello, " + name + "!");
```

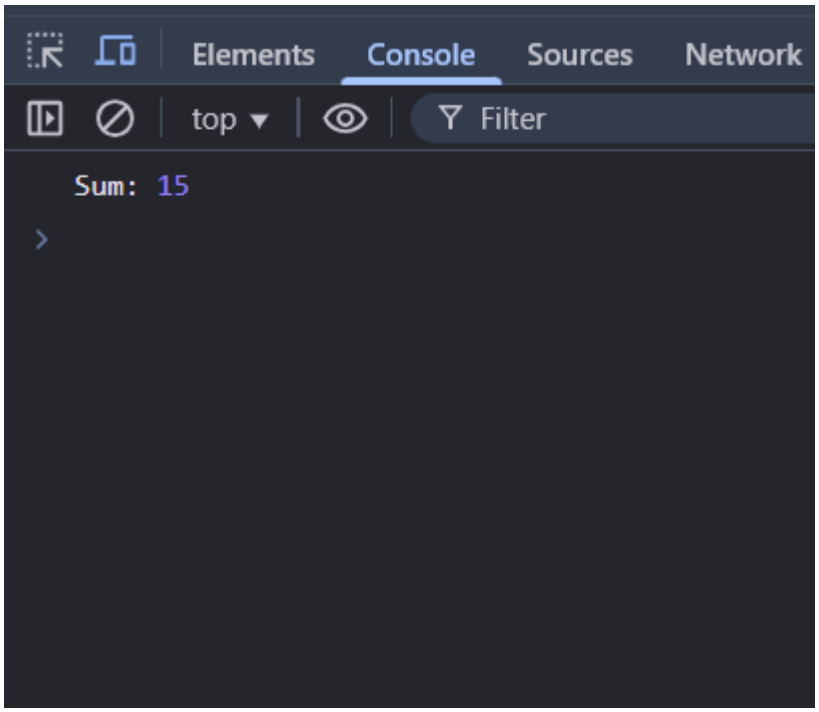
```
}
greetUser("John");
</script>
```



Task 2:

```
<script>
function calculateSum(a, b) {
  return a + b;
}
let result = calculateSum(5, 10);
console.log("Sum:", result);
```

</script>



Module 7: array:

Theory Assignment

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

Ans:

An **array** stores multiple values in one variable.

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

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

- Ans: push() → add end
- pop() → remove end
- shift() → remove first
- unshift() → add first

Lab Assignment

• 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.

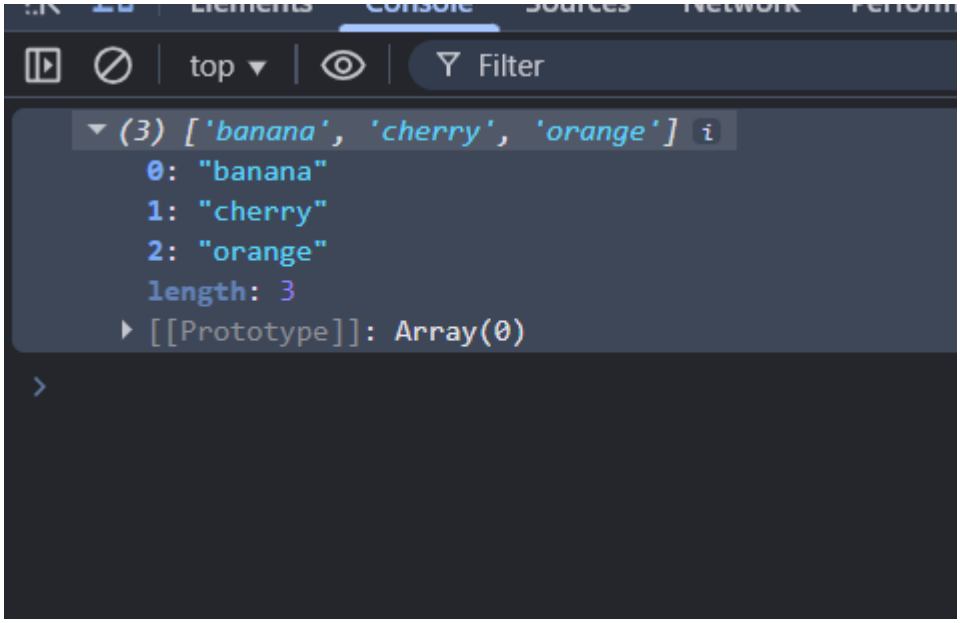
• Log the modified array to the console.

• Task 2:

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

Task 1:

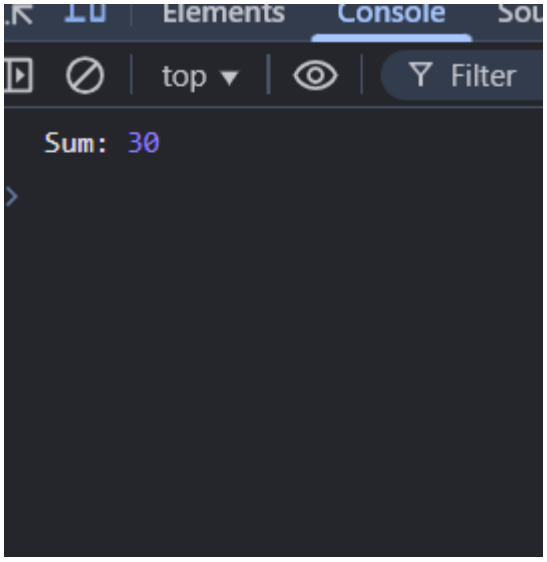
```
<script>
let fruits = ["apple", "banana", "cherry"];
fruits.push("orange");
fruits.shift();
console.log(fruits);
</script>
```



Task 2:

```
<script>
let numbers = [2, 4, 6, 8, 10];
let sum = 0;
```

```
for (let i = 0; i < numbers.length; i++) {
  sum += numbers[i];
}
console.log("Sum:", sum);
</script>
```



Module 8 : object :

Theory

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

Ans: An object holds data as key-value pairs. Different from arrays (which use index-based values).

• Question 2: Explain how to access and update object properties using dot notation andbracket notation.

Ans:

Dot notation: obj.key
Bracket notation: obj["key"]

Lab Assignment

• Task:

• Create a JavaScript object car with properties brand, model, and year. Use JavaScript to:

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

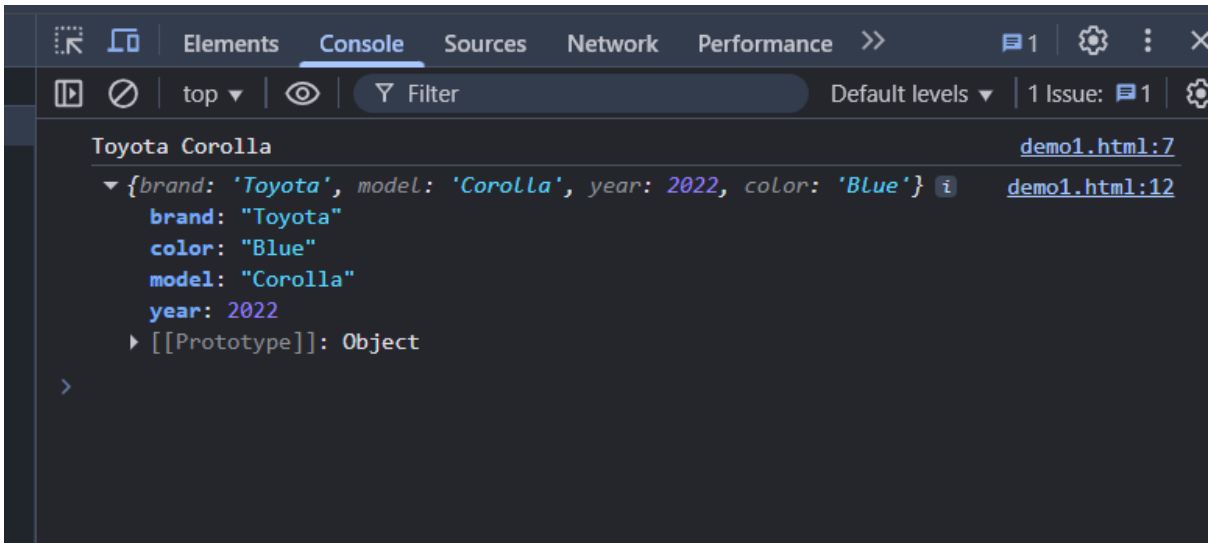
• Update the yearproperty.

• Add a new property color to the car object.

<script>

let car = {

```
  brand: "Toyota",
  model: "Corolla",
  year: 2020
};
console.log(car.brand, car.model);

car.year = 2022;
car.color = "Blue";

console.log(car);
</script>
```



Module 9: JavaScript events:

Theory Assignment

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

Ans:

Events are user actions (click, hover, keypress).
Event listeners run code when events happen.

• Question 2: How does the addEventListener()method work in JavaScript? Provide an example.

Ans:

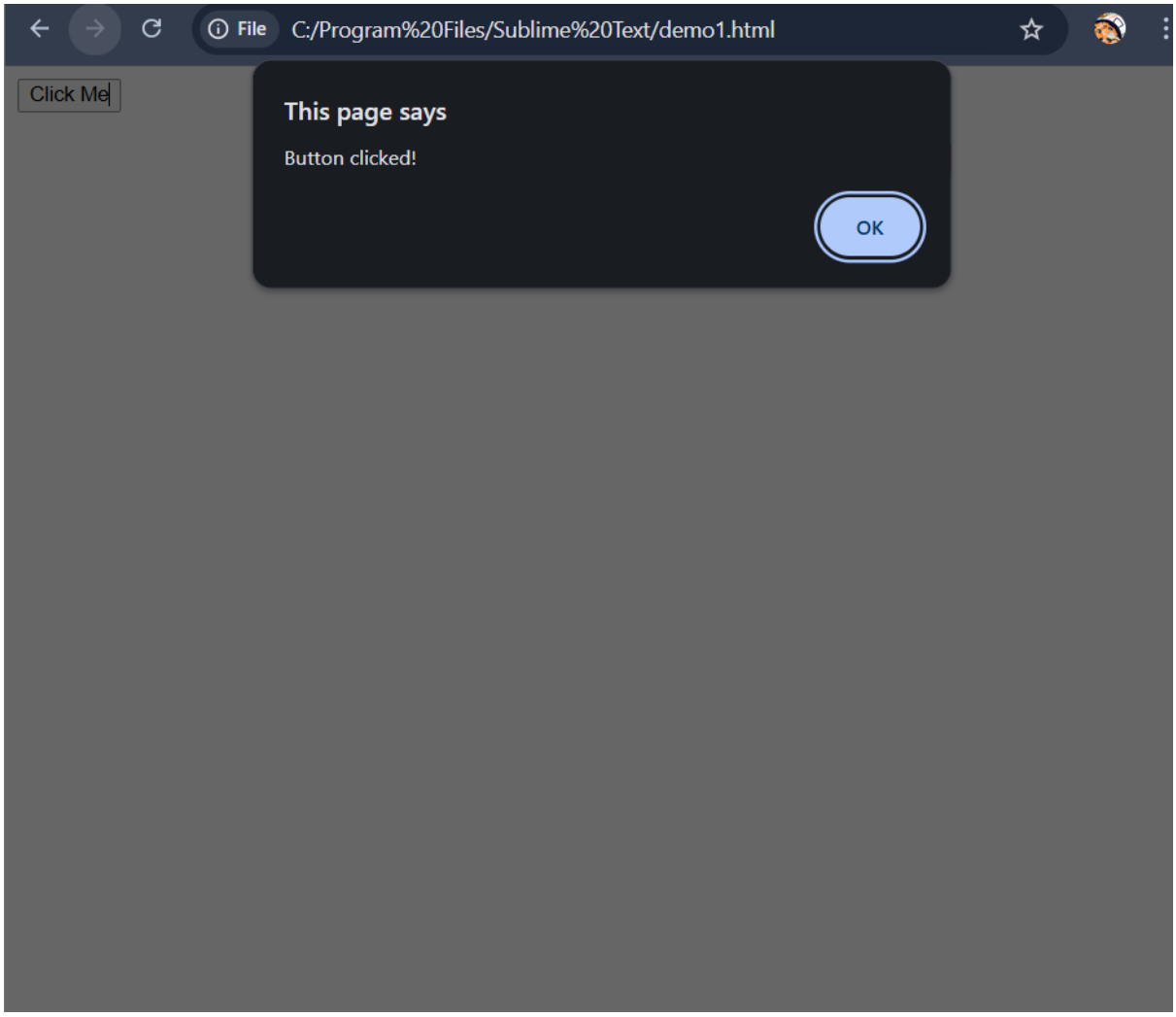
addEventListener(event, function) attaches event handler.

Lab Assignment

• Task:

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

```
<!DOCTYPE html>
<html>
<body>
<button id="myBtn">Click Me</button>

<script>
document.getElementById("myBtn").addEventListener("click", function() {
  alert("Button clicked!");
});
</script>
</body>
</html>
```



Module 10: DOM manipulation

Theory Assignment

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

Ans: DOM = Document Object Model → lets JS change HTML content & style.

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

- Ans: getElementById("id") → one element
- getElementsByClassName("class") → list of elements
- querySelector("cssSelector") → first matching element

Lab Assignment

• Task: • Create an HTML page with a paragraph (

) that displays "Hello, World!".

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

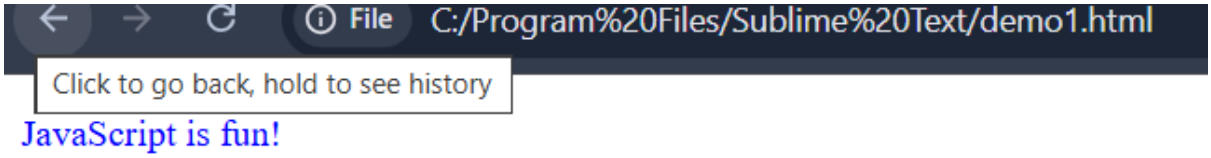
• Change the color of the paragraph to blue.

<!DOCTYPE html>

<html>

<body>

```
<p id="text">Hello, World!</p>


<script>
let p = document.getElementById("text");
p.textContent = "JavaScript is fun!";
p.style.color = "blue";
</script>
</body>
</html>
```



Module 11 JavaScript timing events (setTimeout, setinterval)

Theory

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

Ans:

setTimeout(fn, time) → runs once after delay.

setInterval(fn, time) → repeats every interval.

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

Ans:

Example delay by 2s:

setTimeout(() => console.log("Hello after 2 seconds!"), 2000);

Lab Assignment

• Task 1: • Write a program that changes the background color of a webpage after 5 secondsusing setTimeout().

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

Task 1

<!DOCTYPE html>

<html>

<body>

<script>

```
setTimeout(() => {
  document.body.style.backgroundColor =
"lightgreen";
}, 5000);
</script>
</body>
</html>
```

Task 2

```
<!DOCTYPE html>
<html>
<body>
<h2 id="clock"></h2>

<script>
function showTime() {
  let now = new Date();
  document.getElementById("clock").innerText =
now.toLocaleTimeString();
```

```
}
setInterval(showTime, 1000);
</script>
</body>
</html>
```

10:08:08 AM

Module 12 JavaScript error handling

Theory Assignment

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

Ans:

Use try-catch-finally to handle errors gracefully.

```
try {
  let result = 10 / 0;
} catch (error) {
  console.log("Error occurred:", error.message);
} finally {
  console.log("Execution complete.");
}
```

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

Ans:

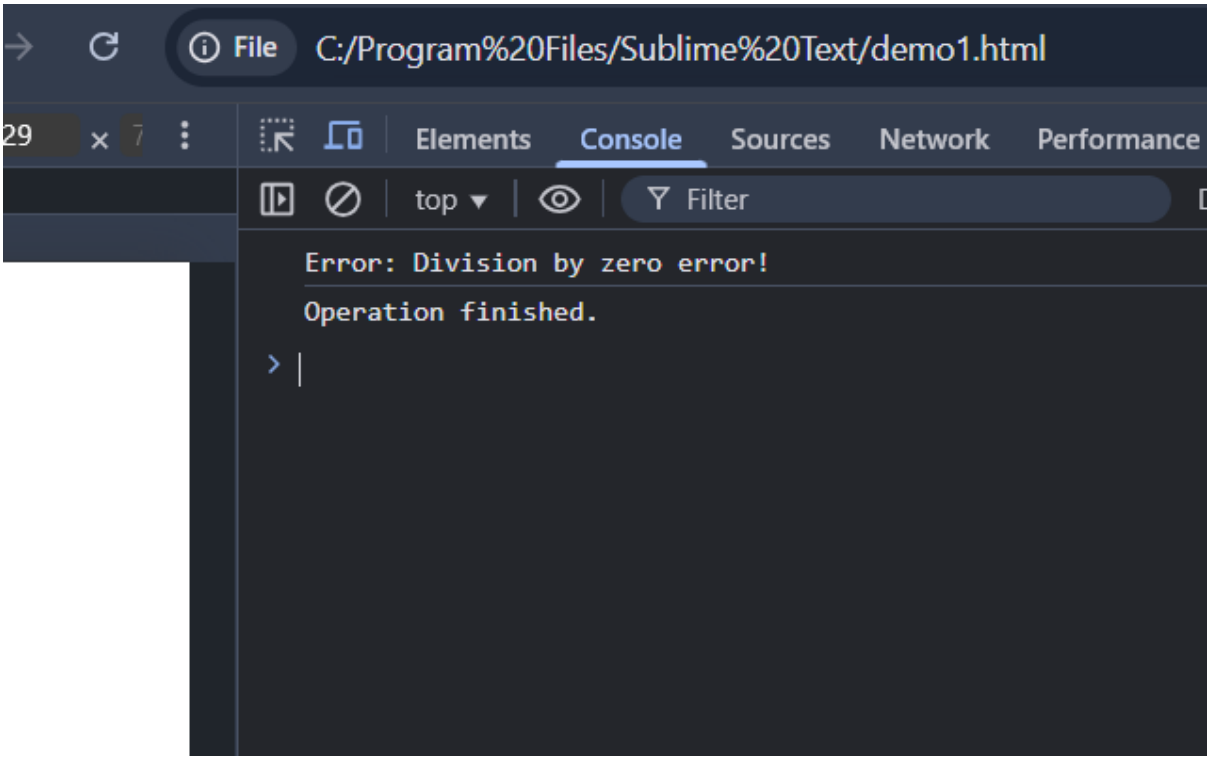
Error handling prevents app crashes and improves reliability.

Lab Assignment

• Task:

• Write a JavaScript program that attempts to divide a number by zero. Use trycatchto handle the error and display an appropriate error message.

```
<script>
try {
  let num = 10;
  let result = num / 0;
  if (!isFinite(result)) throw "Division by zero error!";
  console.log(result);
} catch (error) {
  console.log("Error:", error);
} finally {
  console.log("Operation finished.");
}
</script>
```


File
C:/Program%20Files/Sublime%20Text/demo1.html
Elements
Console
Sources
Network
Performance
top
Filter
Error: Division by zero error!
Operation finished.