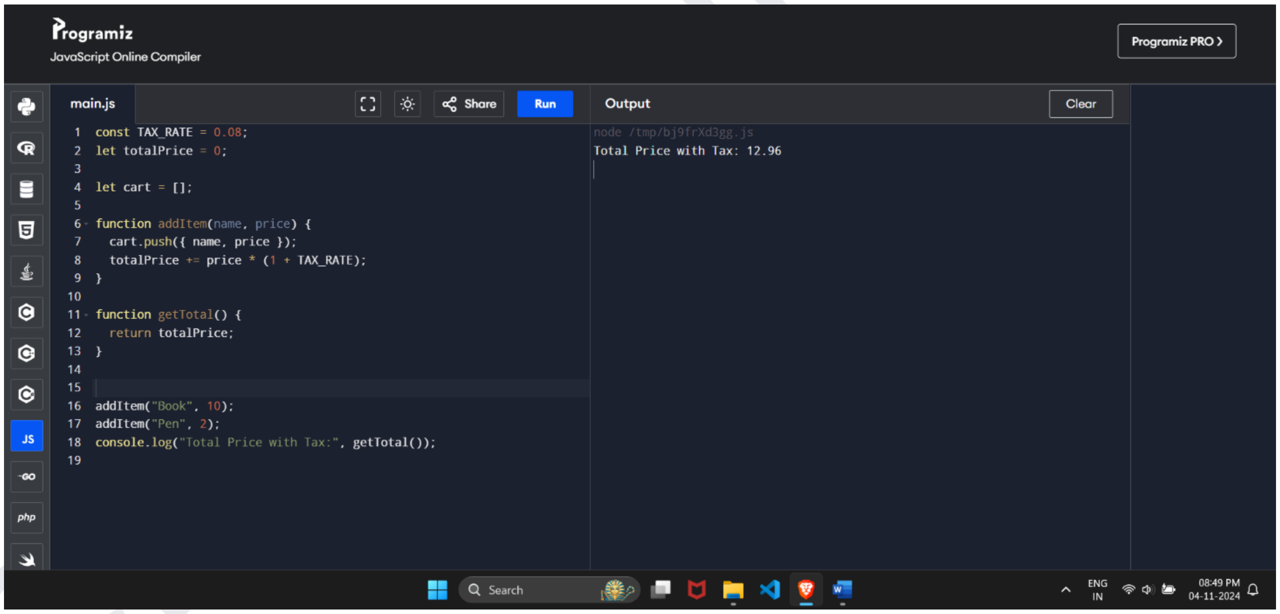
CDAC Mumbai

Module: WPT
Topic: Assignment - 6

<u>Section 1</u>

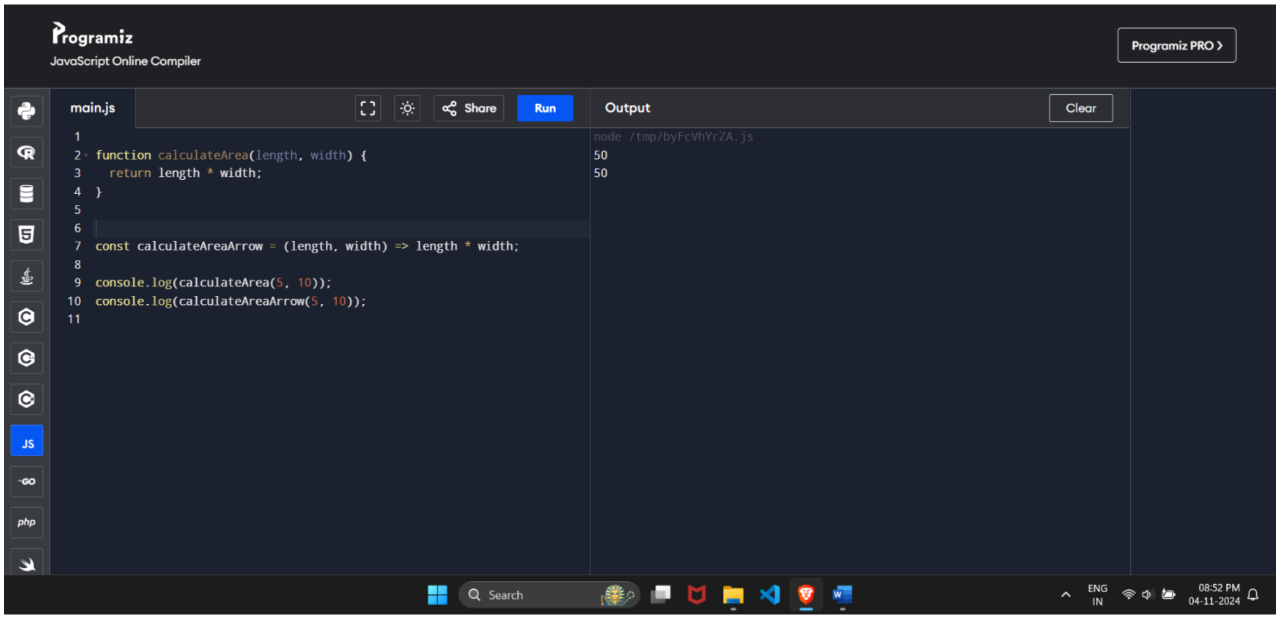
1. **Question:** Create a simple shopping cart application using let, const, and var to manage items. Use const for constant values (like tax rates) and let for variables that might change (like total price).



```
const TAX_RATE = 0.08;
let totalPrice = 0;

let cart = [];

function addItem(name, price) {
  cart.push({ name, price });
  totalPrice += price * (1 + TAX_RATE);
}

function getTotal() {
  return totalPrice;
}


addItem("Book", 10);
addItem("Pen", 2);
console.log("Total Price with Tax:", getTotal());
```

```
node /tmp/bj9frXd3gg.js
Total Price with Tax: 12.96
```

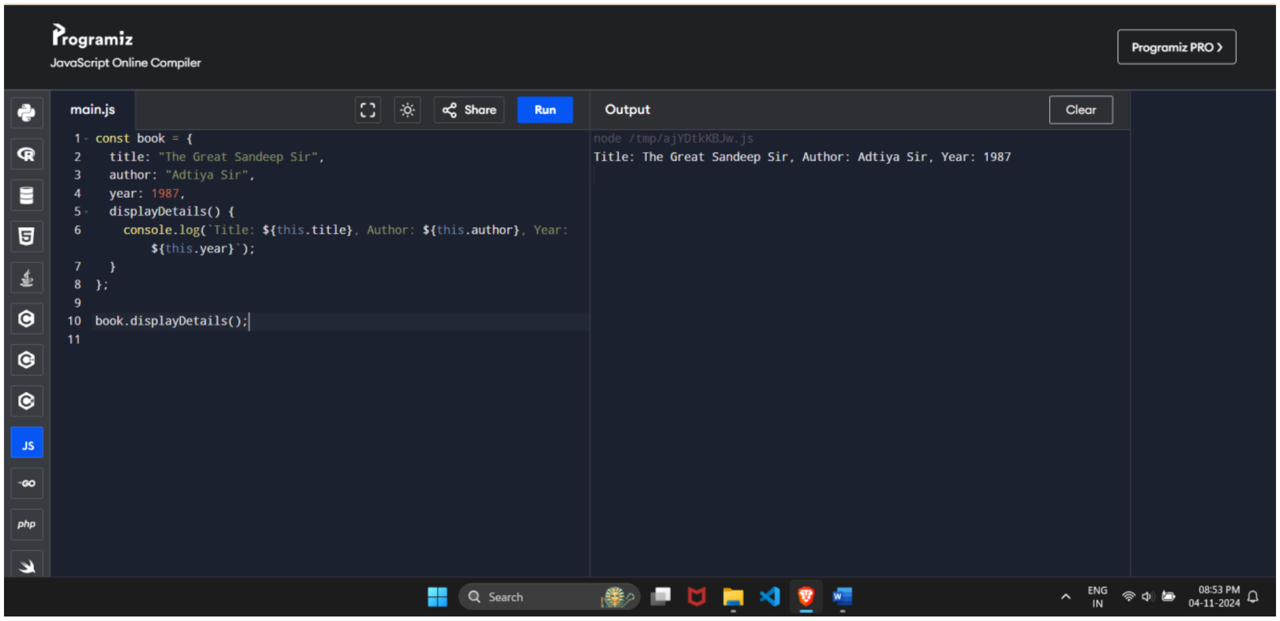
2. **Question:** Write a function to calculate the area of a rectangle using both a regular function and an arrow function.



```
function calculateArea(length, width) {
  return length * width;
}


const calculateAreaArrow = (length, width) => length * width;

console.log(calculateArea(5, 10));
console.log(calculateAreaArrow(5, 10));
```

```
node /tmp/byFcVhYrZA.js
50
50
```

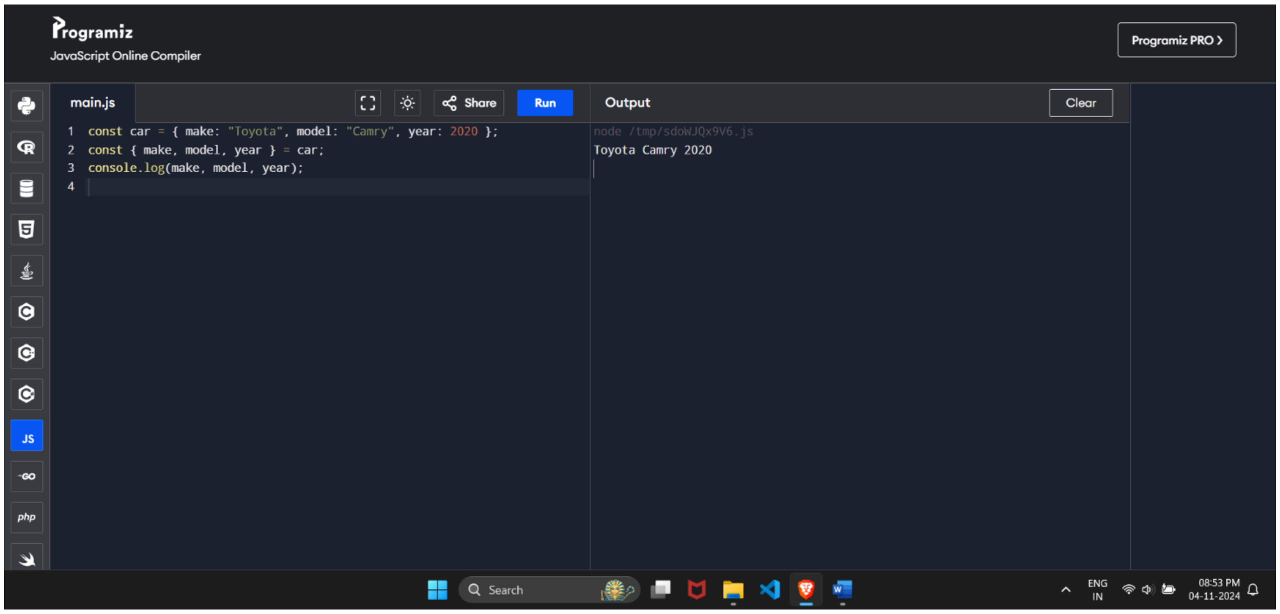
3. **Question:** Create an object to represent a book with properties such as title, author, and year published. Add a method to display the book details.



```
const book = {
  title: "The Great Sandeep Sir",
  author: "Adtiya Sir",
  year: 1987,
  displayDetails() {
    console.log(`Title: ${this.title}, Author: ${this.author}, Year:
      ${this.year}`);
  }
};

book.displayDetails();
```

```
node /tmp/ajYDtkKBJw.js
Title: The Great Sandeep Sir, Author: Adtiya Sir, Year: 1987
```

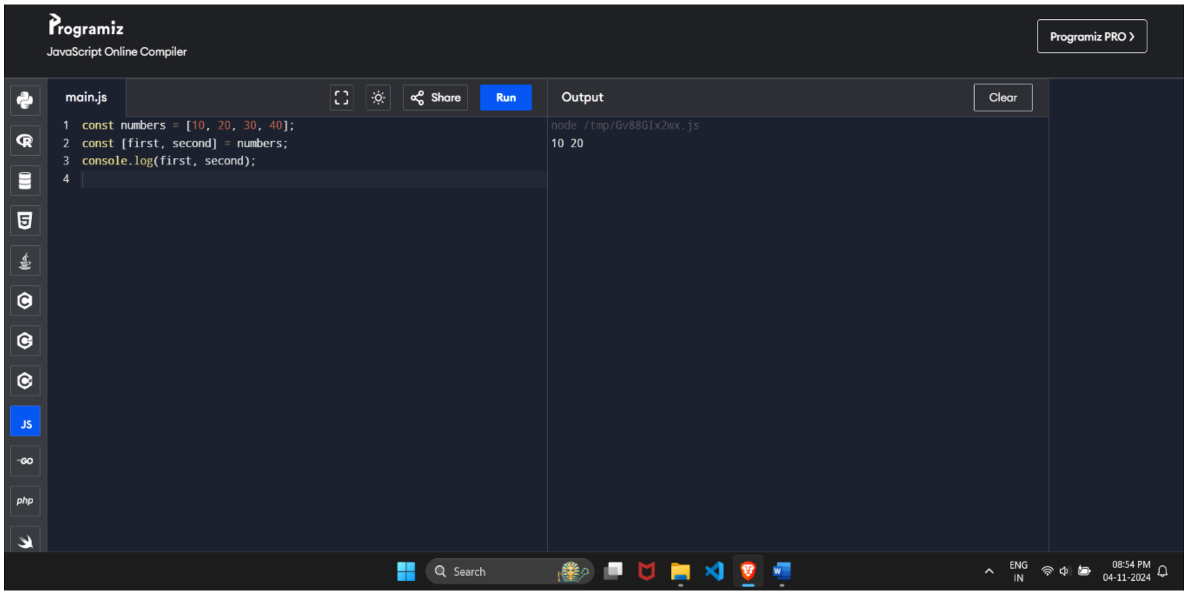
4. **Question:** Given an object representing a car, use object destructuring to extract its properties.



```
const car = { make: "Toyota", model: "Camry", year: 2020 };
const { make, model, year } = car;
console.log(make, model, year);
```

```
node /tmp/sdoWJQx9V6.js
Toyota Camry 2020
```

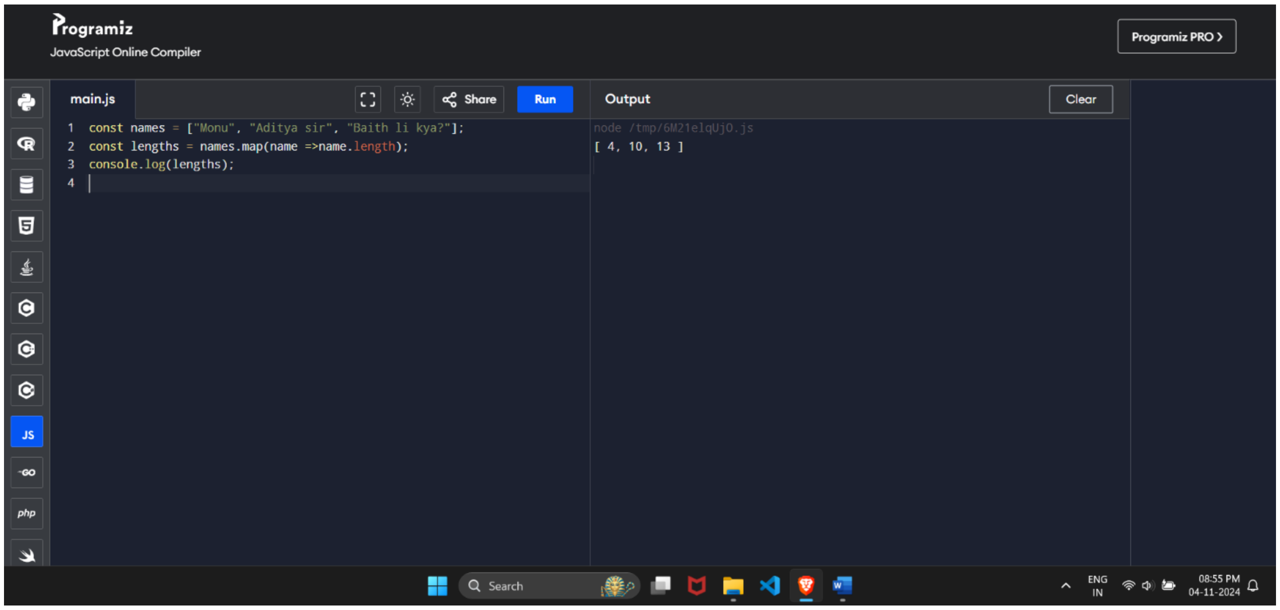
5. **Question:** Given an array of numbers, use array destructuring to extract the first two numbers.



```
const numbers = [10, 20, 30, 40];
const [first, second] = numbers;
console.log(first, second);
```

```
node /tmp/Gv88GIx2wx.js
10 20
```

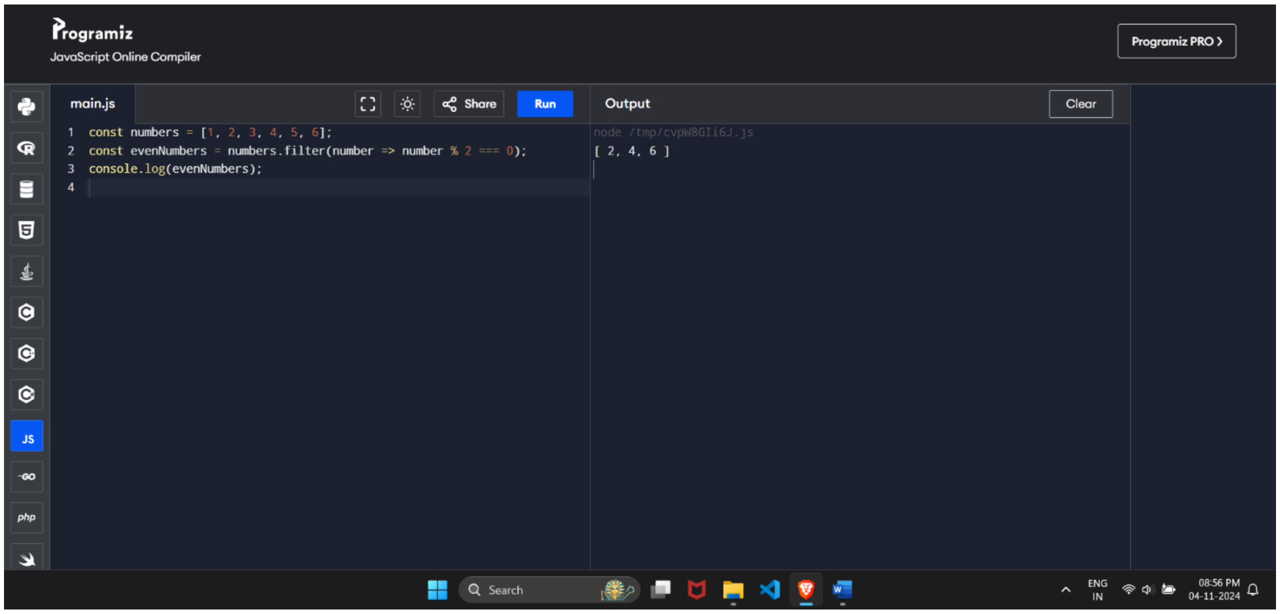
6. **Question:** Use the `map` method to create a new array that contains the lengths of the names in the following array.



```
const names = ["Monu", "Aditya sir", "Baith li kya?"];
const lengths = names.map(name =>name.length);
console.log(lengths);
```

```
node /tmp/6M21elqUjO.js
[ 4, 10, 13 ]
```

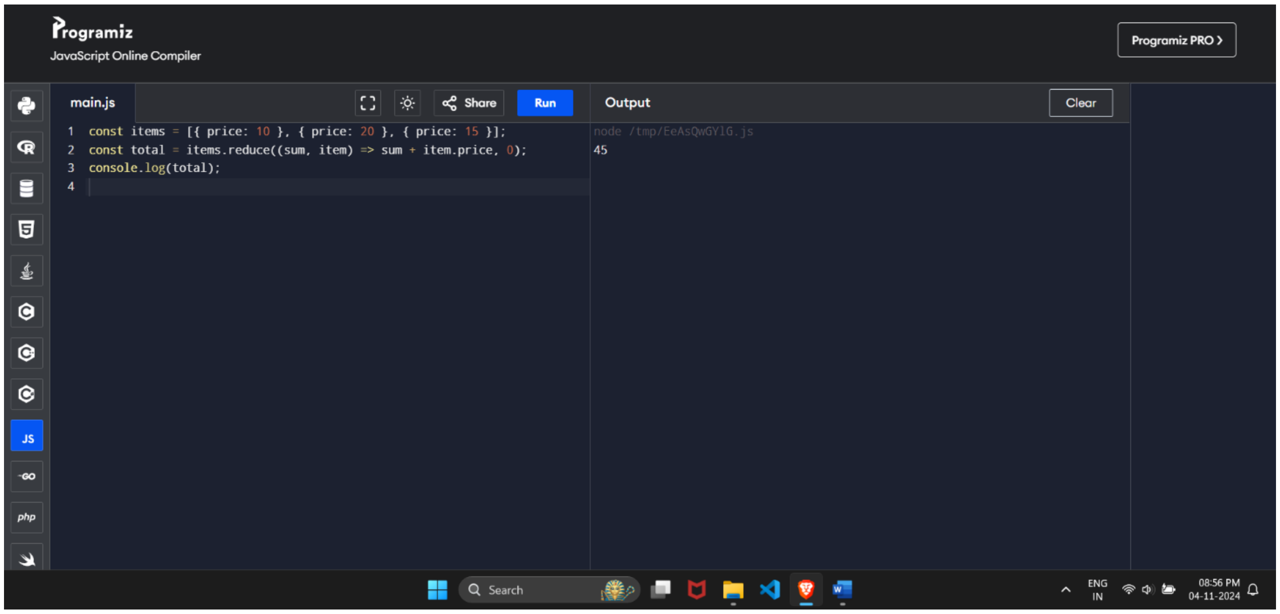
7. **Question:** Use the filter method to create a new array containing only the even numbers from the given array.



```
const numbers = [1, 2, 3, 4, 5, 6];
const evenNumbers = numbers.filter(number => number % 2 === 0);
console.log(evenNumbers);
```

```
node /tmp/cvpW8GIi6J.js
[ 2, 4, 6 ]
```

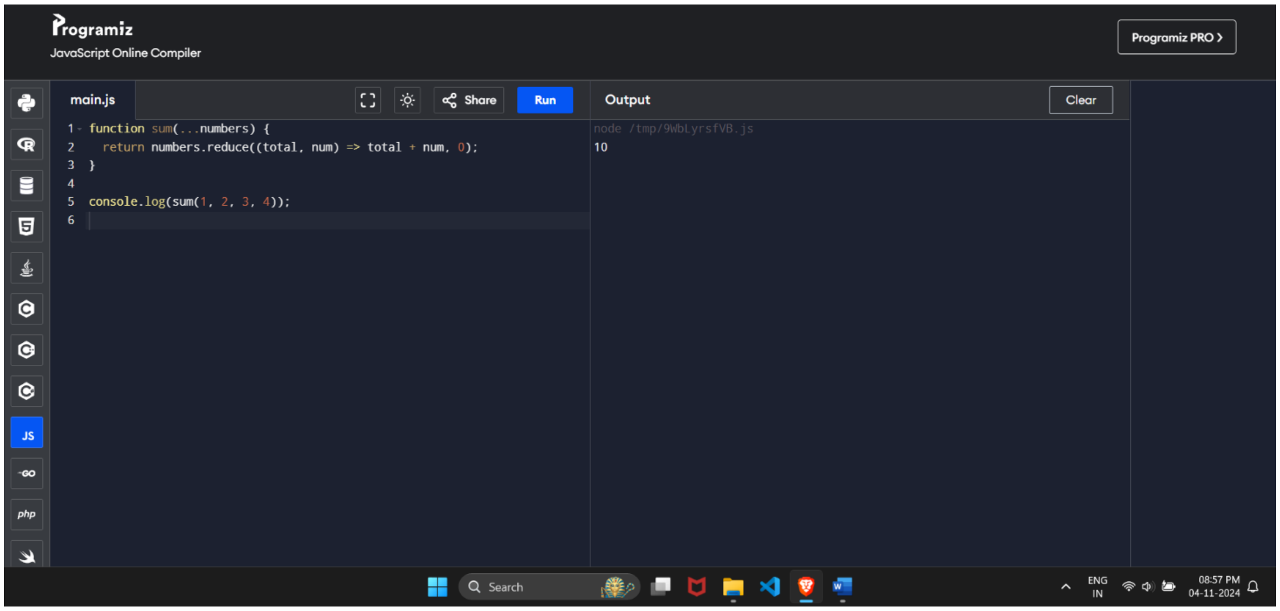
8. **Question:** Use the reduce method to find the total price of items in a shopping cart.



```javascript
const items = [{ price: 10 }, { price: 20 }, { price: 15 }];
const total = items.reduce((sum, item) => sum + item.price, 0);
console.log(total);
```

```
node /tmp/EeAsQwGYlG.js
45
```

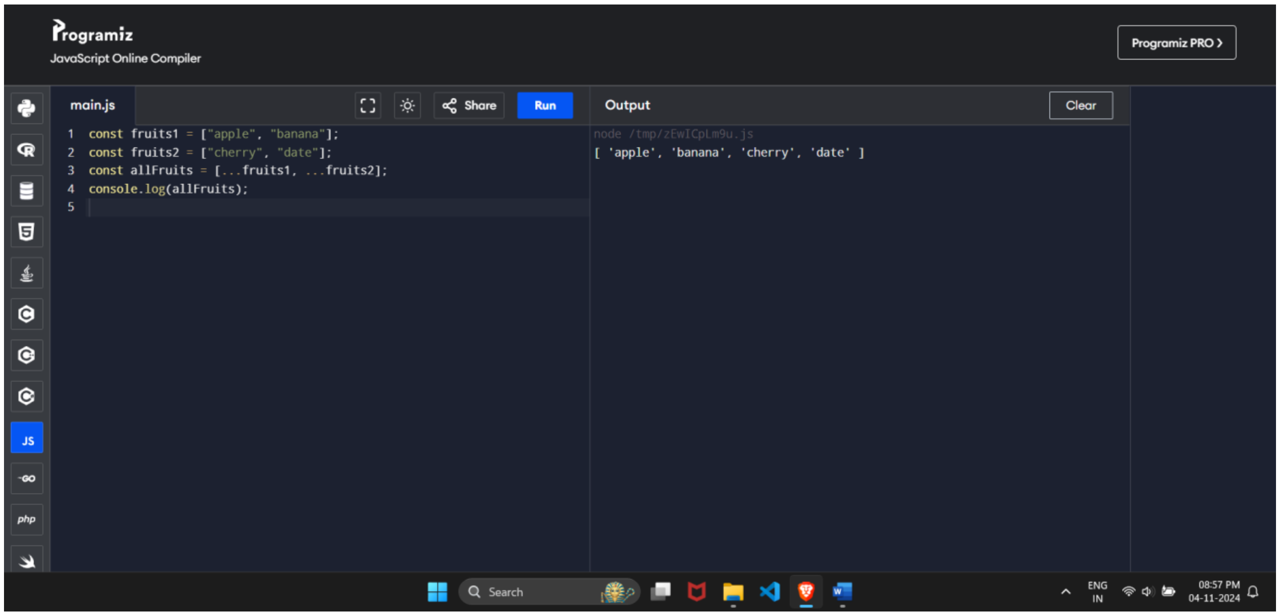
9. **Question:** Create a function that takes any number of arguments and returns their sum using the rest operator.



```javascript
function sum(...numbers) {
  return numbers.reduce((total, num) => total + num, 0);
}

console.log(sum(1, 2, 3, 4));
```

```
node /tmp/9WbLyrsfVB.js
10
```

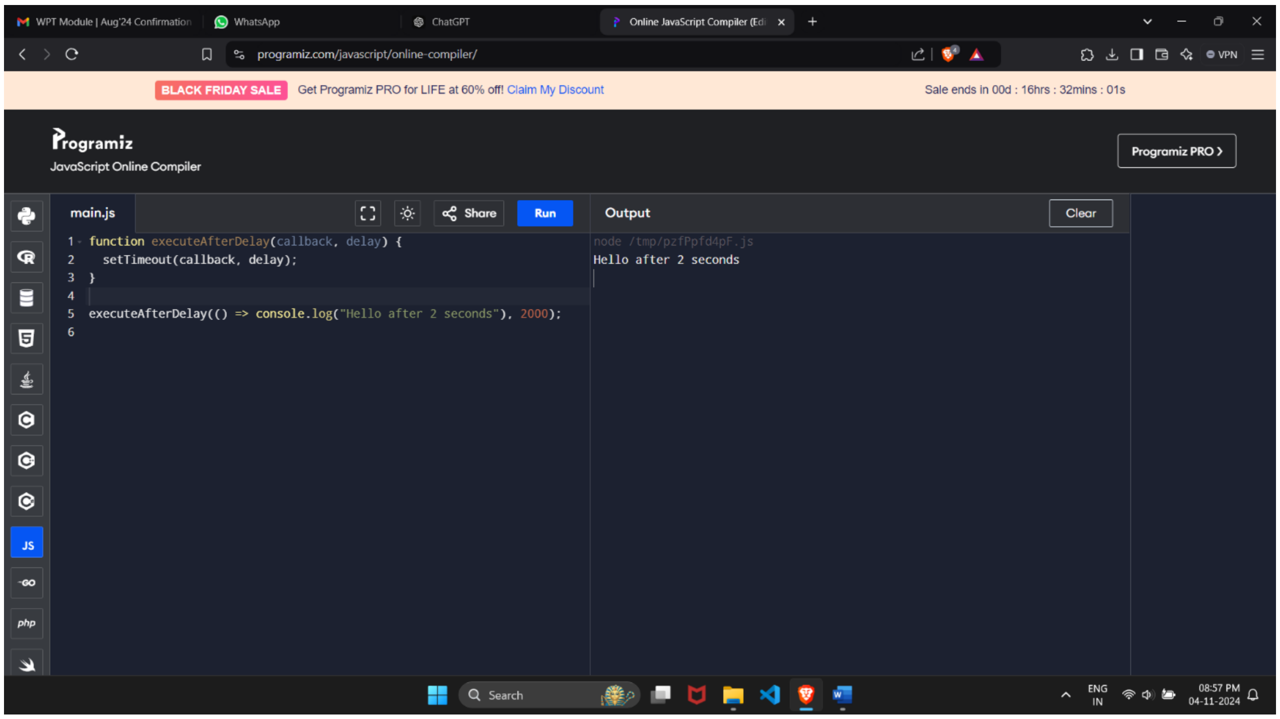
10. **Question:** Use the spread operator to merge two arrays of fruits.



```
const fruits1 = ["apple", "banana"];
const fruits2 = ["cherry", "date"];
const allFruits = [...fruits1, ...fruits2];
console.log(allFruits);
```

```
node /tmp/zEwICpLm9u.js
[ 'apple', 'banana', 'cherry', 'date' ]
```

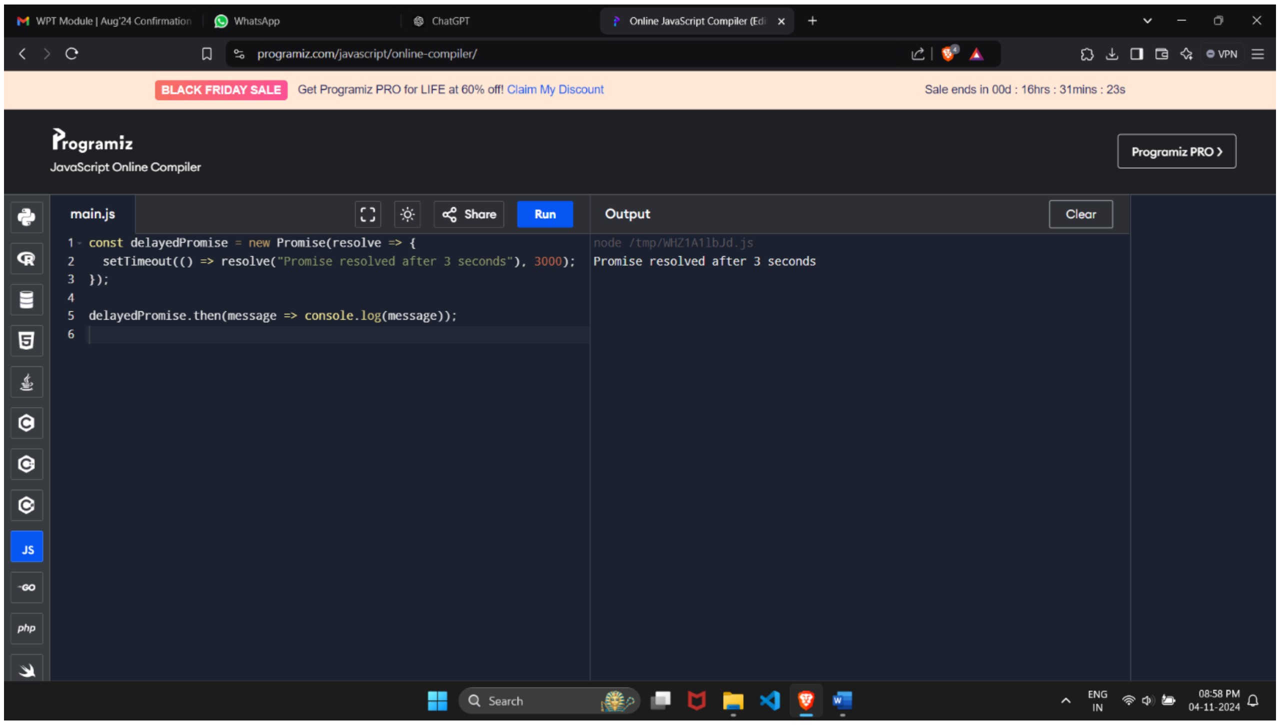
11. **Question:** Write a function that accepts a callback and executes it after a delay.


WPT Module | Aug'24 Confirmation
WhatsApp
ChatGPT
Online JavaScript Compiler (Edi
programiz.com/javascript/online-compiler/
BLACK FRIDAY SALE
Get Programiz PRO for LIFE at 60% off! Claim My Discount
Sale ends in 00d : 16hrs : 32mins : 01s
Programiz
JavaScript Online Compiler
Programiz PRO
main.js
Share
Run
Output
Clear

```
function executeAfterDelay(callback, delay) {
  setTimeout(callback, delay);
}

executeAfterDelay(() => console.log("Hello after 2 seconds"), 2000);
```

```
node /tmp/pzfPpfd4pF.js
Hello after 2 seconds
```

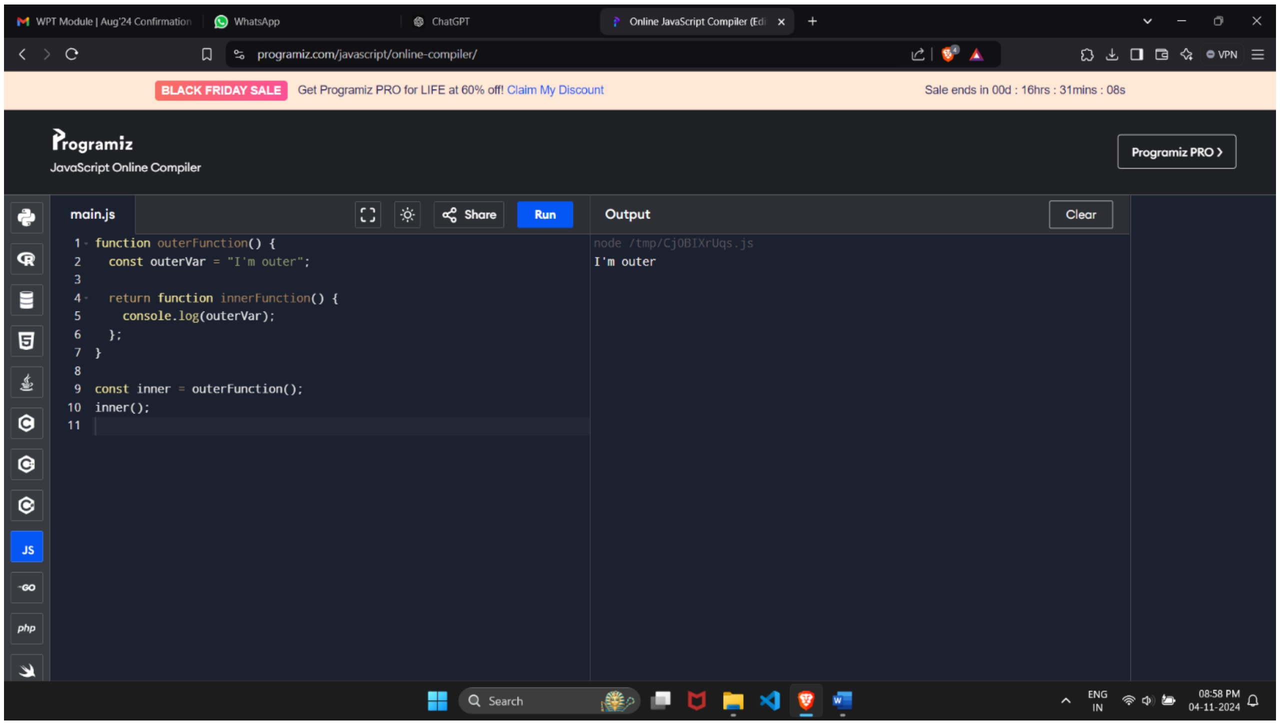
12. **Question:** Create a promise that resolves with a message after 3 seconds.


programiz.com/javascript/online-compiler/
BLACK FRIDAY SALE Get Programiz PRO for LIFE at 60% off! Claim My Discount
Sale ends in 00d : 16hrs : 31mins : 23s
Programiz
JavaScript Online Compiler
Programiz PRO
main.js
Share
Run
Output
Clear

```javascript
const delayedPromise = new Promise(resolve => {
  setTimeout(() => resolve("Promise resolved after 3 seconds"), 3000);
});

delayedPromise.then(message => console.log(message));
```

```
node /tmp/WHZ1A1lbJd.js
Promise resolved after 3 seconds
```

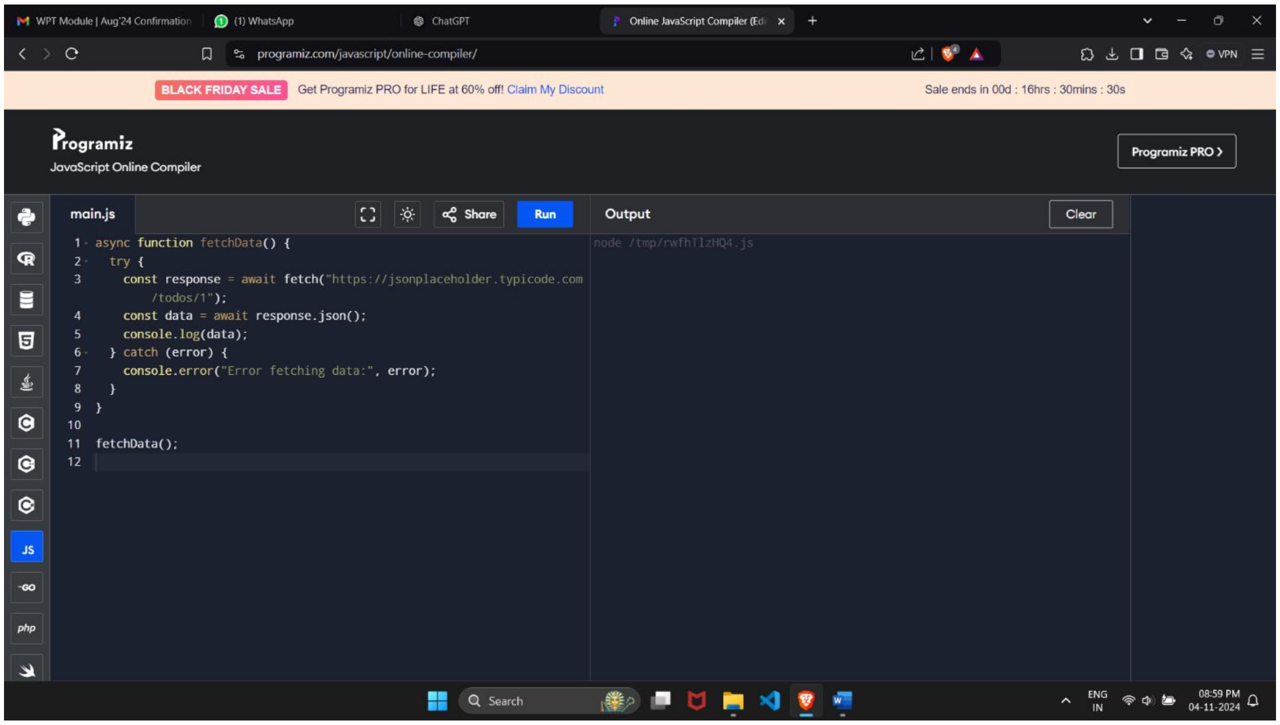
13. **Question:** Create a function that returns another function, demonstrating closure.


programiz.com/javascript/online-compiler/
BLACK FRIDAY SALE Get Programiz PRO for LIFE at 60% off! Claim My Discount
Sale ends in 00d : 16hrs : 31mins : 08s
Programiz
JavaScript Online Compiler
Programiz PRO
main.js
Share
Run
Output
Clear

```javascript
function outerFunction() {
  const outerVar = "I'm outer";

  return function innerFunction() {
    console.log(outerVar);
  };
}

const inner = outerFunction();
inner();
```

```
node /tmp/CjOBIXrUqs.js
I'm outer
```

14. **Question:** Use async/await to fetch data from a public API and log it to the console.



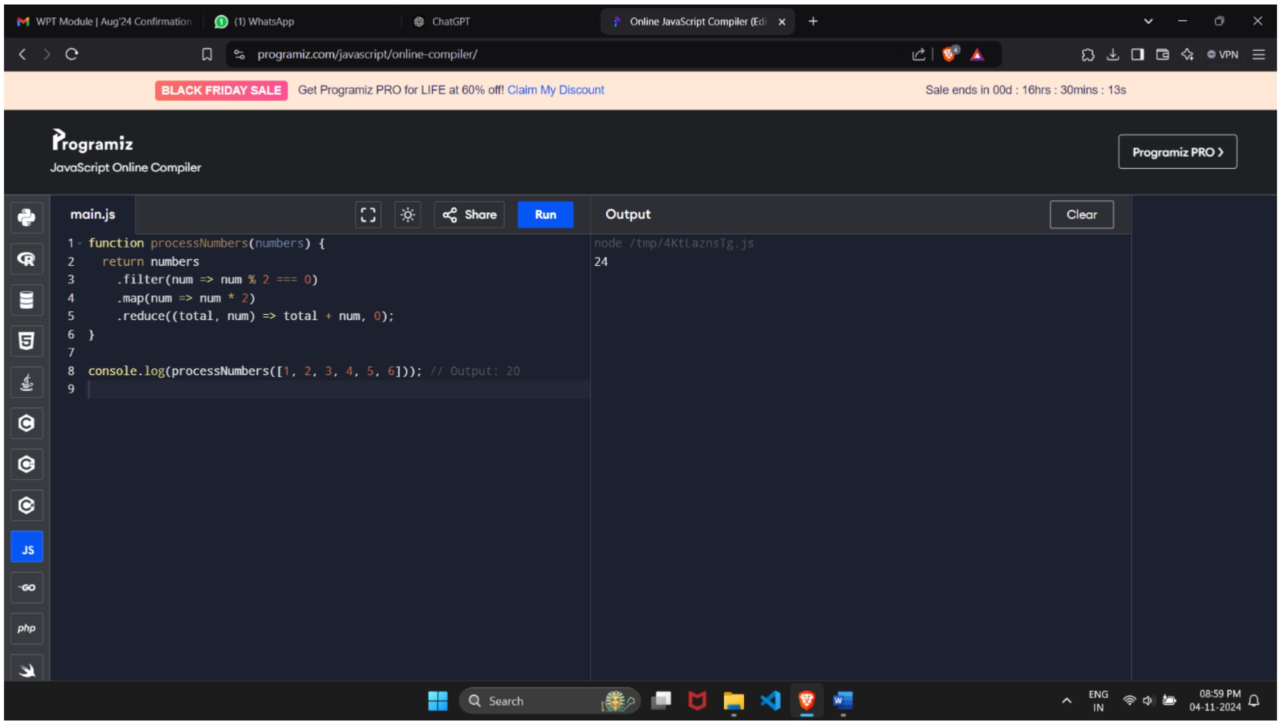
main.js

```javascript
async function fetchData() {
  try {
    const response = await fetch("https://jsonplaceholder.typicode.com
      /todos/1");
    const data = await response.json();
    console.log(data);
  } catch (error) {
    console.error("Error fetching data:", error);
  }
}

fetchData();

```

Output

```
node /tmp/rwfhTlzHQ4.js
```

15. **Question:** Create a function that takes an array of numbers, applies a filter to keep only even numbers, then uses map to double those numbers, and finally returns the total using reduce.



main.js

```javascript
function processNumbers(numbers) {
  return numbers
    .filter(num => num % 2 === 0)
    .map(num => num * 2)
    .reduce((total, num) => total + num, 0);
}

console.log(processNumbers([1, 2, 3, 4, 5, 6])); // Output: 20

```

Output

```
node /tmp/4KtLaznsTg.js
24
```

Section 2

Project Title: Personal Budget Tracker

Duration: 30 Minutes

Description:

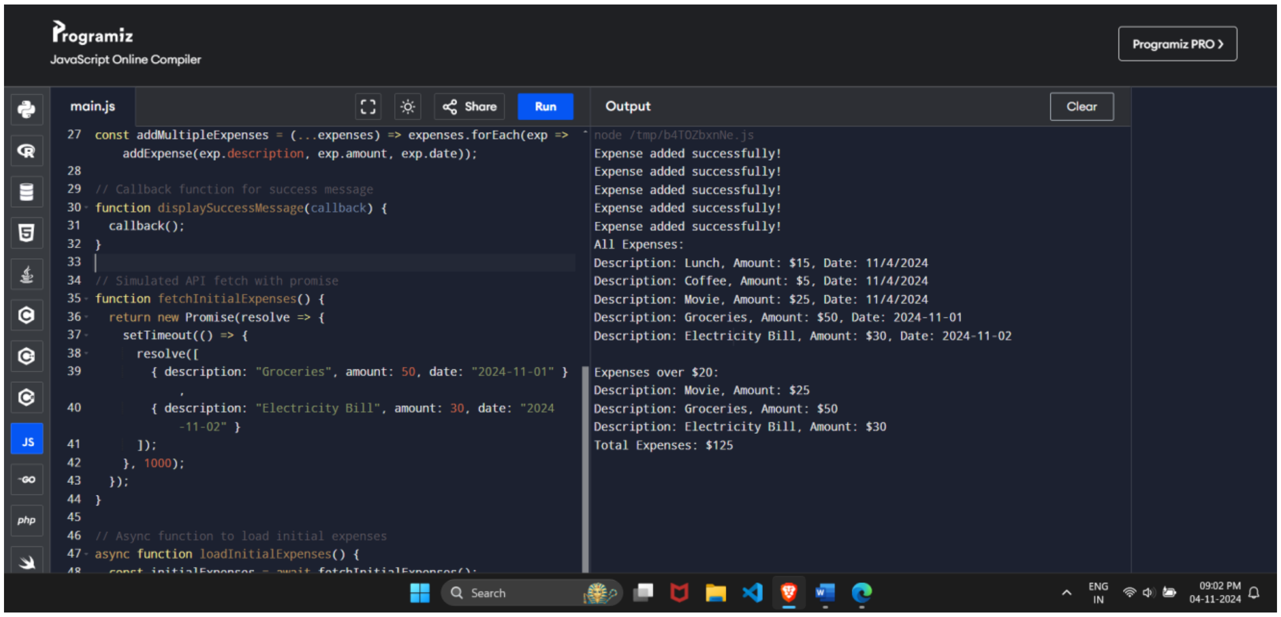
Create simple Personal Budget Tracker application that allows users to manage their expenses. The application should include functionalities to add, view, and calculate the total expenses. You will utilize various JavaScript concepts to implement this application.

Requirements:

1. **Variables:** Use let, const, and var to manage state variables like expense list and total expense.
2. **Functions and Arrow Functions:** Create functions to add an expense, display all expenses, and calculate the total. Use an arrow function for at least one of these.
3. **JavaScript Objects:** Represent each expense as an object with properties such as description, amount, and date.
4. **Destructuring:** Use array and object destructuring when retrieving expense details for display.
5. **Array Methods (Map, Filter, Reduce):**
   - Use map to display a list of expense descriptions.
   - Use filter to show only expenses above a certain amount (e.g., $20).
   - Use reduce to calculate the total expenses.
6. **Rest and Spread Operator:** Use the rest operator to allow adding multiple expenses at once. Use the spread operator to create a new expense list when adding new expenses.
7. **Callback Functions:** Implement a function that takes a callback to display a success message after an expense is added.
8. **Promises:** Create a promise that simulates fetching initial expenses from an API (you can just resolve with a hard-coded array).
9. **Closures:** Use a closure to create a function that maintains the state of total expenses.
10. **Async/Await:** Use async/await to fetch initial expenses and display them in the application when it loads.

Code :

```
// Variables
let expenseList = []; // List of expenses
let totalExpense = 0; // Total expense tracker

// Functions
function addExpense(description, amount, date = new Date().toLocaleDateString()) {
  const expense = { description, amount, date }; // Expense object
  expenseList.push(expense); // Add to list
  totalExpense += amount; // Update total
  displaySuccessMessage(() => console.log("Expense added successfully!"));
}

function displayExpenses() {
  console.log("All Expenses:");
  expenseList.forEach(({ description, amount, date }) =>
    console.log(`Description: ${description}, Amount: $${amount}, Date: ${date}`)
  );

  console.log("\nExpenses over $20:");
  expenseList.filter(exp => exp.amount > 20).forEach(exp =>
    console.log(`Description: ${exp.description}, Amount: $${exp.amount}`)
  );

  console.log(`Total Expenses: $${expenseList.reduce((total, exp) => total +
exp.amount, 0)}`);
}

const addMultipleExpenses = (...expenses) => expenses.forEach(exp =>
addExpense(exp.description, exp.amount, exp.date));

// Callback function for success message
function displaySuccessMessage(callback) {
  callback();
}

// Simulated API fetch with promise
function fetchInitialExpenses() {
  return new Promise(resolve => {
    setTimeout(() => {
      resolve([
        { description: "Groceries", amount: 50, date: "2024-11-01" },
        { description: "Electricity Bill", amount: 30, date: "2024-11-02" }
      ]);
    }, 1000);
  });
}

// Async function to load initial expenses
async function loadInitialExpenses() {
  const initialExpenses = await fetchInitialExpenses();
  addMultipleExpenses(...initialExpenses);
  displayExpenses();
}

// Initialize with async data
loadInitialExpenses();
addExpense("Lunch", 15); // Adding a single expense
addMultipleExpenses(
  { description: "Coffee", amount: 5 },
  { description: "Movie", amount: 25 }
);
```



```
const addMultipleExpenses = (...expenses) => expenses.forEach(exp =>
    addExpense(exp.description, exp.amount, exp.date));

// Callback function for success message
function displaySuccessMessage(callback) {
  callback();
}

// Simulated API fetch with promise
function fetchInitialExpenses() {
  return new Promise(resolve => {
    setTimeout(() => {
      resolve([
        { description: "Groceries", amount: 50, date: "2024-11-01" }
          ,
        { description: "Electricity Bill", amount: 30, date: "2024
          -11-02" }
      ]);
    }, 1000);
  });
}

// Async function to load initial expenses
async function loadInitialExpenses() {
```

```
node /tmp/b4TOZbxnNe.js
Expense added successfully!
Expense added successfully!
Expense added successfully!
Expense added successfully!
Expense added successfully!
All Expenses:
Description: Lunch, Amount: $15, Date: 11/4/2024
Description: Coffee, Amount: $5, Date: 11/4/2024
Description: Movie, Amount: $25, Date: 11/4/2024
Description: Groceries, Amount: $50, Date: 2024-11-01
Description: Electricity Bill, Amount: $30, Date: 2024-11-02

Expenses over $20:
Description: Movie, Amount: $25
Description: Groceries, Amount: $50
Description: Electricity Bill, Amount: $30
Total Expenses: $125
```