

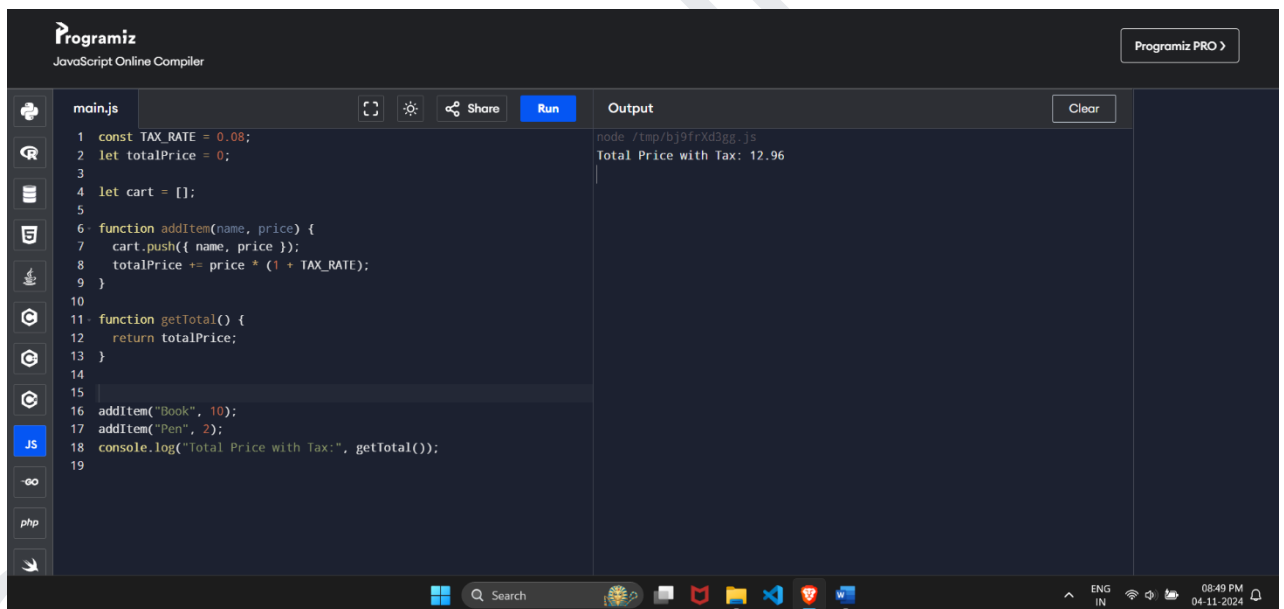
# CDAC Mumbai

Module: WPT

Topic: Assignment - 6

## Section 1

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).



The screenshot displays the Programiz JavaScript Online Compiler interface. The code editor on the left contains the following JavaScript code:

```
1 const TAX_RATE = 0.08;
2 let totalPrice = 0;
3
4 let cart = [];
5
6 function addItem(name, price) {
7   cart.push({ name, price });
8   totalPrice += price * (1 + TAX_RATE);
9 }
10
11 function getTotal() {
12   return totalPrice;
13 }
14
15
16 addItem("Book", 10);
17 addItem("Pen", 2);
18 console.log("Total Price with Tax:", getTotal());
19
```

The output panel on the right shows the result of the code execution:

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

The interface includes a file explorer on the left with 'main.js' selected, a 'Run' button, and a 'Share' link. The bottom status bar shows the system time as 08:49 PM on 04-11-2024.

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

The screenshot shows the Programiz JavaScript Online Compiler interface. The editor contains the following code in `main.js`:

```
1 function calculateArea(length, width) {  
2   return length * width;  
3 }  
4  
5  
6  
7 const calculateAreaArrow = (length, width) => length * width;  
8  
9 console.log(calculateArea(5, 10));  
10 console.log(calculateAreaArrow(5, 10));  
11
```

The output window shows the results of the execution:

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

The interface includes a sidebar with icons for file management, a top bar with 'Run' and 'Share' buttons, and a bottom status bar showing the time as 08:52 PM on 04-11-2024.

- 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.

The screenshot shows the Programiz JavaScript Online Compiler interface. The editor contains the following code in `main.js`:

```
1 const book = {  
2   title: "The Great Sandeep Sir",  
3   author: "Aditya Sir",  
4   year: 1987,  
5   displayDetails() {  
6     console.log(`Title: ${this.title}, Author: ${this.author}, Year: ${this.year}`);  
7   }  
8 };  
9  
10 book.displayDetails();  
11
```

The output window shows the result of the execution:

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

The interface includes a sidebar with icons for file management, a top bar with 'Run' and 'Share' buttons, and a bottom status bar showing the time as 08:53 PM on 04-11-2024.

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

The screenshot shows the Programiz JavaScript Online Compiler interface. The editor on the left contains the following JavaScript code in a file named `main.js`:

```
1 const car = { make: "Toyota", model: "Camry", year: 2020 };
2 const { make, model, year } = car;
3 console.log(make, model, year);
4
```

The 'Run' button is highlighted in blue. The output panel on the right shows the result of the execution:

```
node /tmp/sdowJQx9V6.js
Toyota Camry 2020
```

The bottom of the image shows a Windows taskbar with the search bar, task icons, and system tray information indicating the time is 08:53 PM on 04-11-2024.

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

The screenshot shows the Programiz JavaScript Online Compiler interface. The editor on the left contains the following JavaScript code in a file named `main.js`:

```
1 const numbers = [10, 20, 30, 40];
2 const [first, second] = numbers;
3 console.log(first, second);
4
```

The 'Run' button is highlighted in blue. The output panel on the right shows the result of the execution:

```
node /tmp/Gv88G1x2wx.js
10 20
```

The bottom of the image shows a Windows taskbar with the search bar, task icons, and system tray information indicating the time is 08:54 PM on 04-11-2024.

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

The screenshot shows the Programiz JavaScript Online Compiler interface. The editor on the left contains the following code in `main.js`:

```
1 const names = ["Monu", "Aditya sir", "Baith li kya?"];
2 const lengths = names.map(name => name.length);
3 console.log(lengths);
4
```

The 'Run' button is highlighted in blue. The output panel on the right displays the result of the execution:

```
node /tmp/6M21e1qUj0.js
[ 4, 10, 13 ]
```

The bottom status bar shows the system time as 08:55 PM on 04-11-2024.

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

The screenshot shows the Programiz JavaScript Online Compiler interface. The editor on the left contains the following code in `main.js`:

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

The 'Run' button is highlighted in blue. The output panel on the right displays the result of the execution:

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

The bottom status bar shows the system time as 08:56 PM on 04-11-2024.

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

The screenshot shows the Programiz JavaScript Online Compiler interface. The editor contains a file named `main.js` with the following code:

```
1 const items = [{ price: 10 }, { price: 20 }, { price: 15 }];
2 const total = items.reduce((sum, item) => sum + item.price, 0);
3 console.log(total);
4
```

The `Run` button is highlighted in blue. The `Output` panel on the right shows the result of the execution:

```
node /tmp/EeAsQwGylG.js
45
```

The bottom status bar indicates the language is set to `JS` and the time is 08:56 PM on 04-11-2024.

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

The screenshot shows the Programiz JavaScript Online Compiler interface. The editor contains a file named `main.js` with the following code:

```
1 function sum(...numbers) {
2   return numbers.reduce((total, num) => total + num, 0);
3 }
4
5 console.log(sum(1, 2, 3, 4));
6
```

The `Run` button is highlighted in blue. The `Output` panel on the right shows the result of the execution:

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

The bottom status bar indicates the language is set to `JS` and the time is 08:57 PM on 04-11-2024.

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

The screenshot shows the Programiz JavaScript Online Compiler interface. The editor on the left contains the following JavaScript code in a file named `main.js`:

```
1 const fruits1 = ["apple", "banana"];
2 const fruits2 = ["cherry", "date"];
3 const allFruits = [...fruits1, ...fruits2];
4 console.log(allFruits);
5
```

The 'Run' button is highlighted in blue. The output panel on the right displays the result of the execution:

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

The bottom of the image shows a Windows taskbar with the search bar and various application icons. The system clock indicates 08:57 PM on 04-11-2024.

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

This screenshot shows the Programiz JavaScript Online Compiler with a more complex example. The editor contains the following code:

```
1 function executeAfterDelay(callback, delay) {
2   setTimeout(callback, delay);
3 }
4
5 executeAfterDelay(() => console.log("Hello after 2 seconds"), 2000);
6
```

The 'Run' button is highlighted in blue. The output panel shows the result:

```
node /tmp/pzfPpf4pf.js
Hello after 2 seconds
```

At the top of the browser window, there is a 'BLACK FRIDAY SALE' banner for Programiz PRO. The Windows taskbar at the bottom shows the same system time as the first screenshot.

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

The screenshot shows the Programiz JavaScript Online Compiler interface. The editor contains the following JavaScript code in `main.js`:

```
1- const delayedPromise = new Promise(resolve => {
2-   setTimeout(() => resolve("Promise resolved after 3 seconds"), 3000);
3- });
4-
5- delayedPromise.then(message => console.log(message));
6-
```

The `Run` button is highlighted in blue. The `Output` panel on the right shows the result of the execution:

```
node /tmp/whZ1A11bJd.js
Promise resolved after 3 seconds
```

The interface includes a top banner for a "BLACK FRIDAY SALE" and a sidebar with icons for various programming languages like Python, JavaScript, and PHP.

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

The screenshot shows the Programiz JavaScript Online Compiler interface. The editor contains the following JavaScript code in `main.js`:

```
1- function outerFunction() {
2-   const outerVar = "I'm outer";
3-
4-   return function innerFunction() {
5-     console.log(outerVar);
6-   };
7- }
8-
9- const inner = outerFunction();
10- inner();
11-
```

The `Run` button is highlighted in blue. The `Output` panel on the right shows the result of the execution:

```
node /tmp/Cj0BIXrUqs.js
I'm outer
```

The interface includes a top banner for a "BLACK FRIDAY SALE" and a sidebar with icons for various programming languages like Python, JavaScript, and PHP.

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

The screenshot shows the Programiz JavaScript Online Compiler interface. The editor contains a JavaScript function named `fetchData` that uses `async/await` to fetch data from a placeholder API. The code is as follows:

```
1- async function fetchData() {
2-   try {
3-     const response = await fetch("https://jsonplaceholder.typicode.com/todos/1");
4-     const data = await response.json();
5-     console.log(data);
6-   } catch (error) {
7-     console.error("Error fetching data:", error);
8-   }
9- }
10
11 fetchData();
12
```

The output panel on the right is empty, indicating the code has not been executed. The interface includes a sidebar with file explorer and language selection (JS, PHP), and a top banner for a Black Friday sale.

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.

The screenshot shows the Programiz JavaScript Online Compiler interface. The editor contains a JavaScript function named `processNumbers` that filters even numbers, doubles them, and returns the total using `reduce`. The code is as follows:

```
1- function processNumbers(numbers) {
2-   return numbers
3-     .filter(num => num % 2 === 0)
4-     .map(num => num * 2)
5-     .reduce((total, num) => total + num, 0);
6- }
7
8 console.log(processNumbers([1, 2, 3, 4, 5, 6])); // Output: 20
9
```

The output panel on the right shows the result of the function call: `node /tmp/4KtLaznsig.js` followed by the number `24`. The interface includes a sidebar with file explorer and language selection (JS, PHP), and a top banner for a Black Friday sale.



## 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 }
);

```

The screenshot shows the Programiz JavaScript Online Compiler interface. The code editor on the left contains the following JavaScript code:

```

27 const addMultipleExpenses = (...expenses) => expenses.forEach(exp =>
    addExpense(exp.description, exp.amount, exp.date));
28
29 // Callback function for success message
30 function displaySuccessMessage(callback) {
31   callback();
32 }
33
34 // Simulated API fetch with promise
35 function fetchInitialExpenses() {
36   return new Promise(resolve => {
37     setTimeout(() => {
38       resolve([
39         { description: "Groceries", amount: 50, date: "2024-11-01" },
40         { description: "Electricity Bill", amount: 30, date: "2024-11-02" }
41       ]);
42     }, 1000);
43   });
44 }
45
46 // Async function to load initial expenses
47 async function loadInitialExpenses() {
48   const initialExpenses = await fetchInitialExpenses();

```

The output panel on the right displays the following results:

```

node /tmp/b4T0ZbxmNe.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

```

The interface includes a top bar with the Programiz logo, a "JavaScript Online Compiler" label, and a "Programiz PRO" button. The left sidebar contains icons for file management and language selection (JS, PHP). The bottom status bar shows system information like language (ENG), time (09:02 PM), and date (04-11-2024).