JavaScript(ES6) Advanced Concepts Assignment 2

Topics Covered

- Array Advanced Methods
- JavaScript Behind the Scenes
- Destructuring, Rest & Spread Operators

Assignment Overview

This assignment is designed to build **strong internal understanding** of advanced JavaScript concepts that are often misunderstood when learned only from examples.

You are required to **observe JavaScript behavior by writing, running, breaking, and fixing code**. The tasks are structured so that true understanding comes only through execution and reasoning.

This assignment cannot be completed correctly by copy-pasting solutions or relying on AI-generated answers. Each task requires **personal experimentation and runtime evidence**.

Learning Objectives

By completing this assignment, students will be able to:

- Use advanced array methods correctly and intentionally
- Understand how JavaScript works behind the scenes (execution context, call stack, hoisting, closures)
- Apply destructuring, rest, and spread operators in real-world scenarios
- Reason about memory, references, and execution flow
- Write expressive, predictable, and maintainable JavaScript

Rules & Academic Integrity (Strict)

- Use **vanilla JavaScript (ES6+) only**
- No frameworks, libraries, or helper utilities
- Code must be written, executed, and modified by **you**
- Every task must produce **unique runtime output** based on your own data
- Identical logic or output across submissions will be rejected

AI Usage Policy: This assignment is intentionally designed so that AI-generated answers will fail. If your submission shows patterns of auto-generated code without experimentation evidence, it will be marked invalid.

Section 1: Array Advanced Methods (Deep Practice)

Task 1.1 – map vs forEach vs reduce (Behavior Study)

Create the same data transformation using:

- `map`
- `forEach`
- `reduce`

Requirements:

- Use your own dataset (numbers or objects)
- Log intermediate steps inside each method
- Clearly show how data flows differently

Do not explain in text. The **logs must tell the story**.

Task 1.2 – filter + reduce Combination

Build a small program that:

- Filters data based on a condition you define
- Aggregates the filtered result using `reduce`

Constraint:

- You must change the condition at least twice and re-run the program
- Show how output changes

Task 1.3 – Custom Array Method Simulation (Hard)

Simulate the behavior of **one array method** (`map`, `filter`, or `reduce`) using a normal loop.

Mandatory:

- Your function must accept a callback
- You must log when the callback is executed
- Compare output with the real method

Section 2: JavaScript Behind the Scenes

Task 2.1 – Hoisting Reality Check

Write code that:

- Uses variables and functions before declaration
- Mixes `var`, `let`, `const`, and function declarations

Requirement:

- Wrap risky code in try/catch
- Log what runs, what fails, and in which order

Task 2.2 – Call Stack Observation

Create multiple nested function calls that:

- Log entry and exit points
- Clearly show execution order

Add one asynchronous operation and observe the difference.

Task 2.3 – Closure Proof

Create a function that:

- Returns another function
- Uses variables from the outer scope

Change the outer variables after creation and observe results.

Section 3: Destructuring, Rest & Spread Operators

Task 3.1 – Destructuring with Defaults

Create an object and:

- Destructure properties with default values
- Intentionally omit some properties

Log results and observe behavior.

Task 3.2 – Rest Operator in Functions

Write a function that:

- Accepts an unknown number of arguments
- Processes them meaningfully (not just logging)

Demonstrate behavior with different argument counts.

Task 3.3 – Spread Operator & References (Important)

Demonstrate the difference between:

- Shallow copy using spread
- Direct reference assignment

Modify nested values and observe effects.

Final Project (Medium → Hard)

JavaScript Data Processor App

Build a small **JavaScript data-processing app** that:

- Accepts a dataset (array of objects you design)
- Uses **map, filter, reduce** together
- Applies destructuring and spread for transformations
- Demonstrates at least one closure
- Logs execution flow clearly

Proof of Work Requirements:

- Add a section called `Experiment Logs`
- Log at least **3 behaviors that surprised you** during execution
- These logs must be based on real runs

Submission Requirements

1. GitHub Repository (Mandatory)

Repository must include:

```
/assignment
  ├── index.js
  ├── README.md
  └── notes.md
```

README.md must include:

- How to run the code
- What concepts are demonstrated
- Link to demo video (if applicable)

---

2. Demo Video (Mandatory for Final Project)

- 3–5 minutes maximum
- Show code running
- Explain logic briefly
- Show outputs changing

Evaluation Criteria

- Depth of understanding shown via runtime behavior
- Correct and intentional use of concepts
- Code clarity and structure
- Evidence of experimentation
- Original thinking

Final Note: This assignment rewards curiosity, patience, and hands-on practice. Memorization and shortcuts will not work here.

Code it. Break it. Understand it.