JavaScript (ES6) Advanced Concepts Assignment 3

Topics Covered

- Set & Map
- Default Parameters
- First-Class & Higher-Order Functions
- Callback Functions

Assignment Overview

This assignment is designed to strengthen your understanding of **how JavaScript treats functions and data structures internally**, not just how to use them syntactically.

You will learn by **writing code, running it multiple times, changing inputs, and observing real behavior**. The tasks are intentionally designed so that memorized answers or AI-generated code will not be sufficient.

To complete this assignment correctly, you must **experiment, observe, and reason like a developer**.

Learning Objectives

By completing this assignment, students will be able to:

- Understand when and why to use `Set` and `Map`
- Apply default parameters safely and intentionally
- Treat functions as values (first-class citizens)
- Build and use higher-order functions correctly
- Control program flow using callback functions
- Reason about execution order and data flow

Rules & Academic Integrity (Strict)

- Use **plain JavaScript (ES6+) only**
- No frameworks, libraries, or external helpers
- Code must be written, executed, and modified by **you**
- Every task must generate **unique runtime output**
- Submissions with identical logic or predictable outputs will be rejected

AI Usage Policy: This assignment is intentionally designed so that AI-generated solutions fail without real execution and experimentation. Evidence of auto-generated code without runtime proof will be marked invalid.

Section 1: Set & Map (Real Usage)

Task 1.1 – Set Behavior Investigation

Create a program that:

- Stores values of different types in a `Set`
- Attempts to add duplicate values intentionally
- Logs the size and contents after each operation

Mandatory:

- Use at least one object and one array as values
- Modify a stored object and observe the result

Task 1.2 – Map vs Object Comparison

Create the same data structure using:

- A plain JavaScript object
- A `Map`

Perform the following:

- Add entries
- Retrieve values
- Use non-string keys

Log differences clearly through output.

Section 2: Default Parameters (Edge Cases)

Task 2.1 – Default Value Traps

Write functions that:

- Use default parameters
- Receive `undefined`, `null`, and valid values

Observe how defaults behave in each case.

Constraint:

- You must log input values and final results together

Task 2.2 – Defaults with Functions & Objects

Create a function where:

- One default parameter is a function
- Another default parameter is an object

Modify values during execution and observe side effects.

Section 3: First-Class & Higher-Order Functions

Task 3.1 – Functions as Data

Demonstrate that functions can be:

- Stored in variables
- Passed as arguments
- Returned from other functions

Use meaningful execution output instead of explanations.

Task 3.2 – Build Your Own Higher-Order Function (Hard)

Create a higher-order function that:

- Accepts a function as an argument
- Controls *when* and *how many times* that function runs

Change the behavior by passing different functions.

Section 4: Callback Functions (Execution Control)

Task 4.1 – Synchronous Callback Flow

Write a program that:

- Uses callbacks to control execution order
- Logs start and end of each step

Rearrange callbacks and observe changes.

Task 4.2 – Asynchronous Callback Observation

Create an asynchronous operation using callbacks (e.g., timers).

Mandatory:

- Log timestamps
- Clearly show execution order vs written order

Final Project (Medium → Hard)

JavaScript Task Runner App

Build a small **JavaScript task runner application** that:

- Stores tasks using `Map` or `Set`
- Executes tasks using callbacks
- Uses default parameters for configuration
- Treats tasks as first-class functions

Proof of Work Requirements:

- Add a section called `Execution Logs`
- Log at least **3 non-obvious behaviors** you observed
- Logs must be based on actual execution

Submission Requirements

1. GitHub Repository (Mandatory)

Repository structure:

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

README.md must include:

- How to run the project

- What concepts are demonstrated
- Challenges faced
- Link to demo video (if required)

---

2. Demo Video (Required for Final Project)

- 3–5 minutes
- Show code execution
- Explain logic briefly
- Show logs changing in real time

Evaluation Criteria

- Depth of understanding via runtime behavior
- Correct use of ES6 concepts
- Clean structure and readable code
- Evidence of experimentation
- Original problem-solving

Final Note: This assignment rewards developers who experiment, observe, and think deeply. Shortcuts will fail.

Write it. Run it. Change it. Understand it.