JavaScript Exercises Report

Hussein Hussein
Student ID: 58301

Repository Information

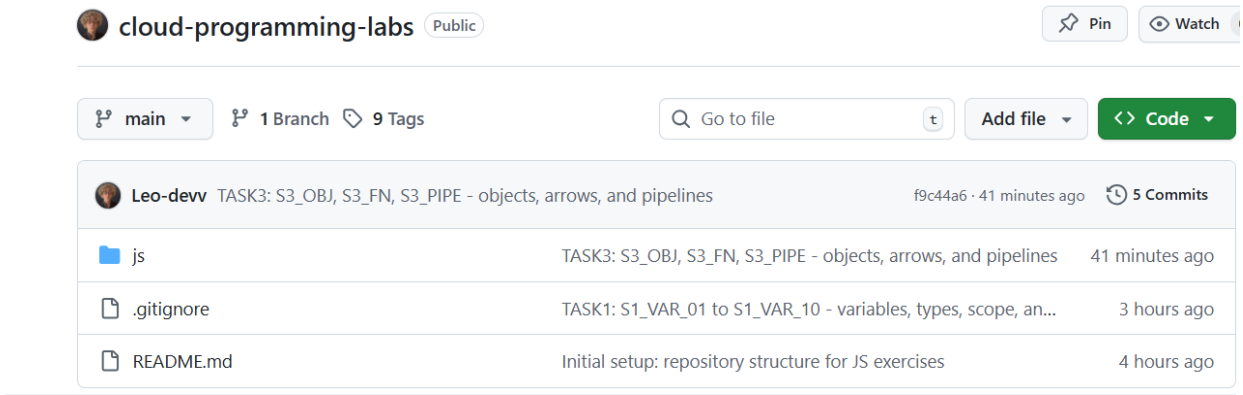
GitHub Repository: https://github.com/Leo-devv/cloud-programming-labs
Online Editor: https://stackblitz.com/github/Leo-devv/cloud-programming-labs

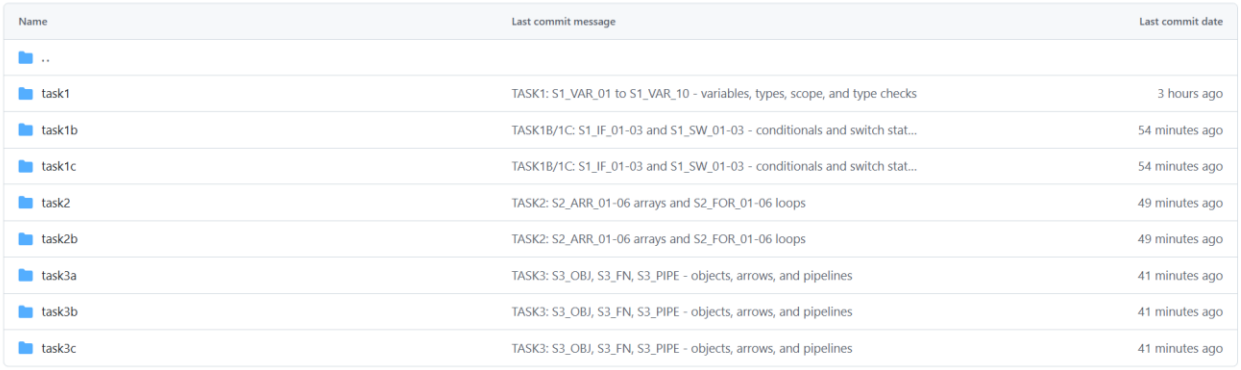
Project Overview

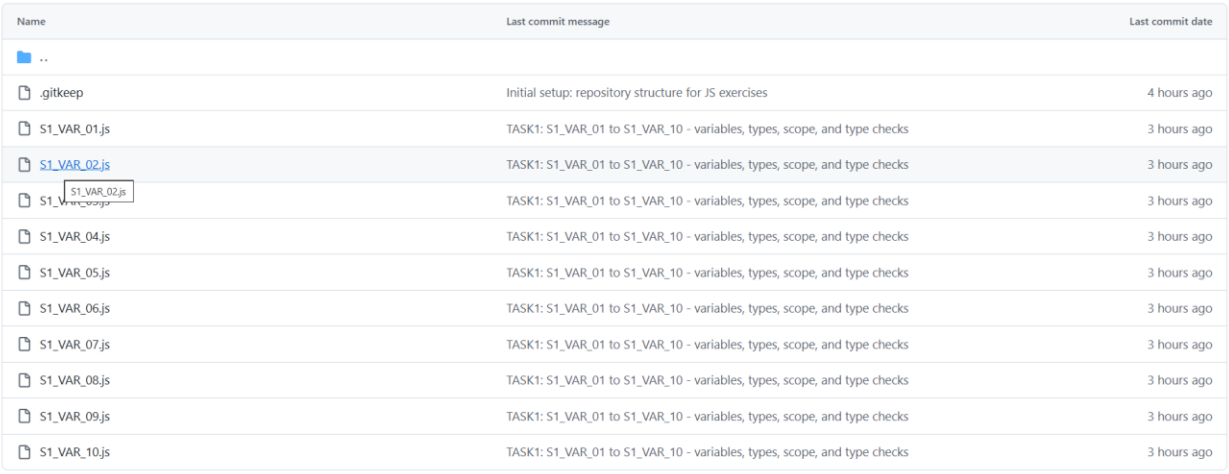
This report documents all JavaScript exercises covering ES6 fundamentals, organized into three stages focusing on different aspects of the language.

Repository Structure

The project structure follows course requirements:







Name	Last commit message	Last commit date
..		
.gitkeep	Initial setup: repository structure for JS exercises	4 hours ago
S1_VAR_01.js	TASK1: S1_VAR_01 to S1_VAR_10 - variables, types, scope, and type checks	3 hours ago
S1_VAR_02.js	TASK1: S1_VAR_01 to S1_VAR_10 - variables, types, scope, and type checks	3 hours ago
S1_VAR_03.js	TASK1: S1_VAR_01 to S1_VAR_10 - variables, types, scope, and type checks	3 hours ago
S1_VAR_04.js	TASK1: S1_VAR_01 to S1_VAR_10 - variables, types, scope, and type checks	3 hours ago
S1_VAR_05.js	TASK1: S1_VAR_01 to S1_VAR_10 - variables, types, scope, and type checks	3 hours ago
S1_VAR_06.js	TASK1: S1_VAR_01 to S1_VAR_10 - variables, types, scope, and type checks	3 hours ago
S1_VAR_07.js	TASK1: S1_VAR_01 to S1_VAR_10 - variables, types, scope, and type checks	3 hours ago
S1_VAR_08.js	TASK1: S1_VAR_01 to S1_VAR_10 - variables, types, scope, and type checks	3 hours ago
S1_VAR_09.js	TASK1: S1_VAR_01 to S1_VAR_10 - variables, types, scope, and type checks	3 hours ago
S1_VAR_10.js	TASK1: S1_VAR_01 to S1_VAR_10 - variables, types, scope, and type checks	3 hours ago

Files follow the naming convention S[stage]_[type]_[number].js (e.g., S1_VAR_01.js).

Stage 1: Variables, Types, and Control Flow

Task 1: Variables and Types

Variable declarations, type checking, and type system operations.

File	Description
`S1_VAR_01.js`	Declaring variables with var, let, and const
`S1_VAR_02.js`	Block scope differences between let and var
`S1_VAR_03.js`	Demonstrating that const prevents reassignment but not mutation
`S1_VAR_04.js`	Safe type checking that handles null correctly
`S1_VAR_05.js`	Detecting arrays using Array.isArray
`S1_VAR_06.js`	Working with NaN and Number.isNaN
`S1_VAR_07.js`	String to number coercion using unary plus
`S1_VAR_08.js`	Safe addition with BigInt for large numbers
`S1_VAR_09.js`	Using typeof on undeclared variables
`S1_VAR_10.js`	Building an inspect utility for type analysis

Task 1B: Conditional Statements

File	Description
`S1_IF_01.js`	Shipping cost calculator with member discounts
`S1_IF_02.js`	Score to letter grade conversion

S1_IF_03.js	Input normalization with falsy value handling

Task 1C: Switch Statements

File	Description
S1_SW_01.js	Day name lookup (1-7 to Monday-Sunday)
S1_SW_02.js	Command router returning status messages
S1_SW_03.js	Basic calculator with division by zero handling

Stage 2: Arrays and Loops

Task 2: Array Operations

File	Description
S2_ARR_01.js	Cleaning and parsing string arrays to numbers
S2_ARR_02.js	Removing duplicates without using Set
S2_ARR_03.js	Splitting arrays into chunks of specified size
S2_ARR_04.js	Flattening nested arrays by one level
S2_ARR_05.js	Computing min, max, and average statistics
S2_ARR_06.js	Filtering and transforming user records

Task 2B: Loop Constructs

File	Description
S2_FOR_01.js	FizzBuzz implementation
S2_FOR_02.js	Finding the first even number in an array
S2_FOR_03.js	Summing values until a threshold is reached
S2_FOR_04.js	Counting occurrences of values
S2_FOR_05.js	Generating a 10x10 multiplication table
S2_FOR_06.js	Summing values in nested arrays

Stage 3: Objects, Functions, and Pipelines

Task 3A: Object Operations

File	Description
`S3_OBJ_01.js`	Safe property access with path notation
`S3_OBJ_02.js`	Merging configuration objects
`S3_OBJ_03.js`	Selecting specific keys from an object
`S3_OBJ_04.js`	Excluding specific keys from an object
`S3_OBJ_05.js`	Inverting object keys and values
`S3_OBJ_06.js`	Grouping items by a property value

Task 3B: Arrow Functions

File	Description
`S3_FN_01.js`	Converting regular functions to arrow syntax
`S3_FN_02.js`	Sorting objects by property using arrow comparators
`S3_FN_03.js`	Creating adder functions with closures
`S3_FN_04.js`	Chaining filter, map, and reduce
`S3_FN_05.js`	Building higher-order predicate functions
`S3_FN_06.js`	Transforming object values with a mapping function

Task 3C: Pipelines

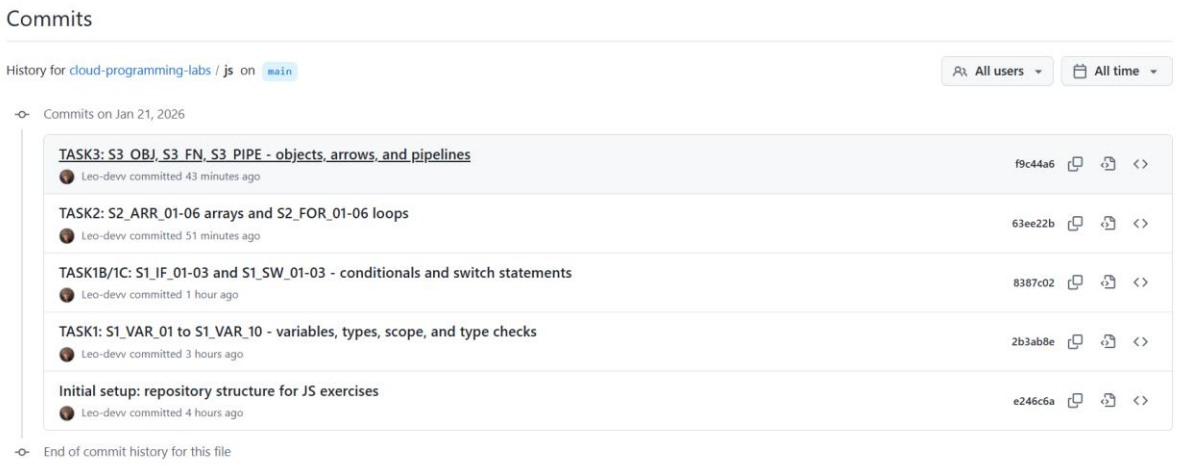
File	Description
`S3_PIPE_01.js`	Left-to-right function composition (pipe)
`S3_PIPE_02.js`	Right-to-left function composition (compose)
`S3_PIPE_03.js`	String normalization pipeline
`S3_PIPE_04.js`	Array processing pipeline
`S3_PIPE_05.js`	Log line parsing pipeline
`S3_PIPE_06.js`	Error-safe pipeline with result objects

Git Workflow

Commit History

Development progressed through incremental commits:

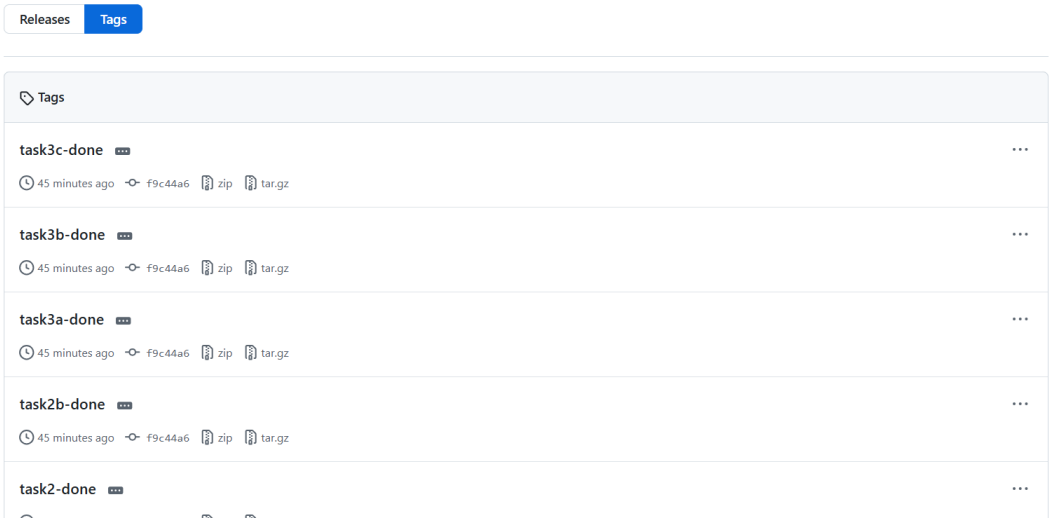
1. Initial setup with folder structure
2. Stage 1 Variables (S1_VAR_01 to S1_VAR_10)
3. Stage 1 Conditionals and Switch (S1_IF and S1_SW)
4. Stage 2 Arrays and Loops (S2_ARR and S2_FOR)
5. Stage 3 Objects, Arrows, and Pipelines (S3_OBJ, S3_FN, S3_PIPE)



Tags

Tags mark completion of each task section:

task1-done, task1b-done, task1c-done, task2-done, task2b-done, task3a-done, task3b-done, task3c-done, all-done



Testing

All exercises were tested using Node.js. Each file contains executable code with output verification.

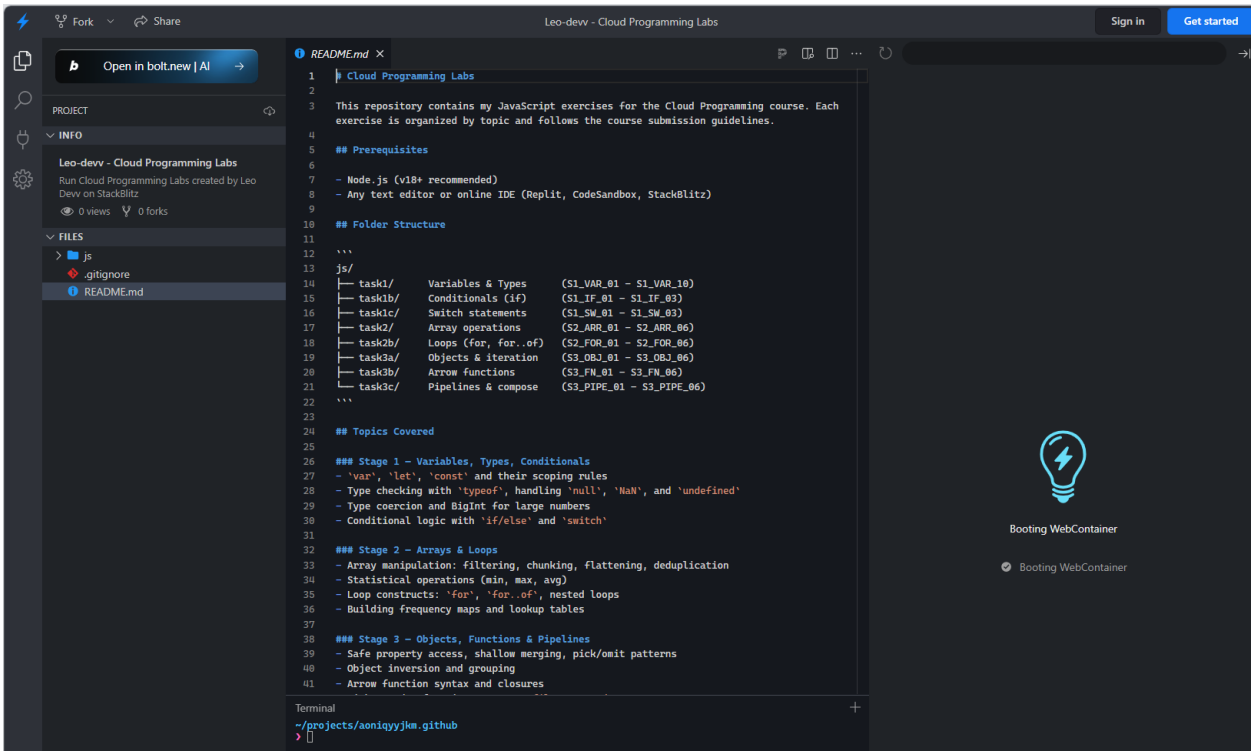
To run any exercise:

```
node js/task1/S1_VAR_01.js
```

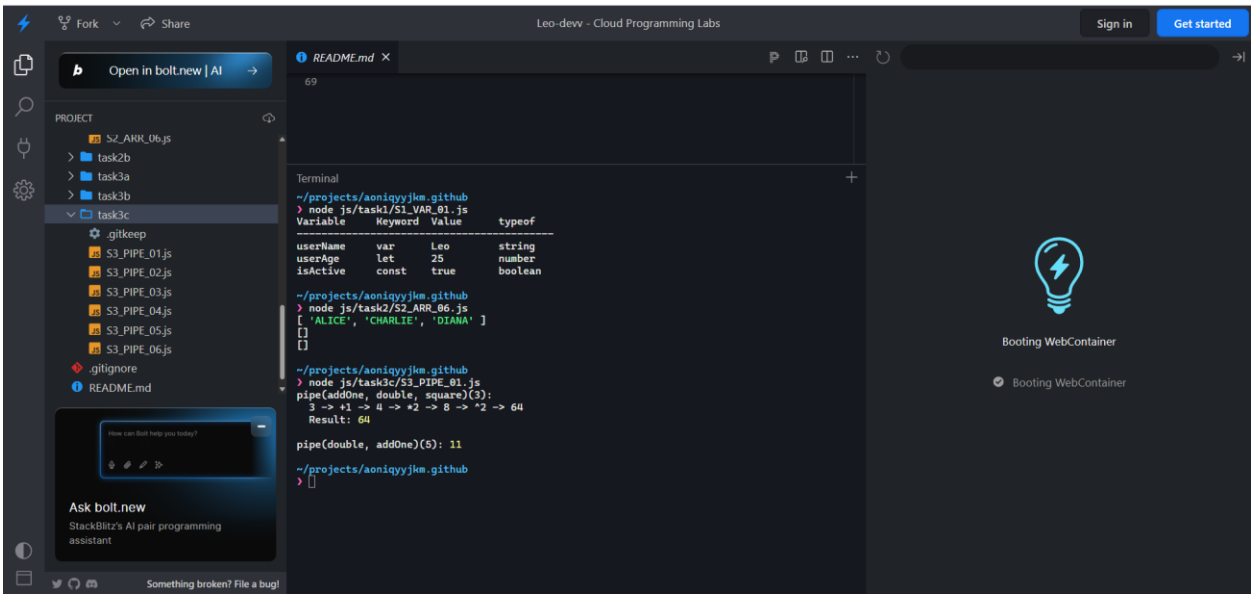
The repository can be tested in StackBlitz using the link above.

Screenshots

StackBlitz Interface



Sample Exercise Output



Summary

All exercises are complete and tested. The project follows course guidelines for structure, file naming, and version control. Tags enable easy navigation during evaluation.