

LAB03 – TypeScript Basics and JSON in Node.js

Objective: Students will learn how to

1. Set up a simple Node.js + TypeScript project.
2. Declare basic TypeScript types and work with typed arrays and functions.
3. Explain what JSON is and how it differs from JavaScript objects.
4. Parse JSON strings and JSON files into typed objects/arrays.
5. Convert TypeScript objects back into JSON and write them to a file.

Lab instruction

- The LAB03 instruction is posted on Mango CMU of the subject 953261.
- The assignment is scheduled and assigned to your account on Mango CMU.
 - ***Check the calendar for the ‘due date’ and ‘close date.’***
 - The submission later than the ‘due date’ will get 50% off your score.
 - At the ‘close date’, you cannot submit your assignment to the system.
- When you finish each question, capture the screen of your execution output for later upload to the MS Teams assignment.
- When you finish all the questions,
 1. Make sure you upload all capture files to Mango CMU assignment.
 2. Post your name and student ID as a comment under the Lab03 assignment to queue for TA to verify your work.
- This lab is worth for 12 points in total.

Grading

Students should have:

- **Part 1 – Project Setup:** 2 points
- **Part 2 – TypeScript Basics in Node:** 3 points
- **Part 3 – JSON String → Typed Objects:** 2 points
- **Part 4 – JSON Files + fs in Node:** 2 points
- **Part 5 – Wrap-up Questions & Evidence (Deliverables):** 3 point

(Optional) Up to **0.5 bonus point** for proper Git commit and/or GitHub repository.

PART 1 — Project Setup (2 points)

1 Create a project folder

- Open **Terminal** or **VS Code integrated terminal**.
- Go to your working directory
- Create and enter a folder called LAB03

```
mkdir LAB03  
cd LAB03
```

Your terminal path must now be inside the LAB03 directory. (E.g, .../Desktop/LAB02 \$)

2 Initialize Node.js Project

- Run the following command in the terminal:

```
npm init -y
```

This creates a package.json file.

3 Install TypeScript, ts-node, and Node type definitions as dev dependencies:

- Run the following command in the terminal:

```
npm install -D typescript ts-node @types/node
```

4 Initialize TypeScript configuration

- Run the following command in the terminal:

```
npx tsc --init
```

This creates tsconfig.json file.

- Recommended changes for Express
- Open tsconfig.json and ensure

```
{  
  "compilerOptions": {  
    "target": "es2020",  
    "module": "commonjs",  
    "rootDir": "./src"  
    "outDir": "./dist",  
    "strict": true  
  }  
}
```

5 Create Source Folder and Entry File

- Create a src folder and an index.ts file inside it:
- In src/index.ts, add:

```
console.log("Hello TypeScript + JSON");
```

6 In package.json, add these scripts:

```
"scripts": {  
  "dev": "ts-node src/index.ts",  
  "build": "tsc",  
  "start": "node dist/index.js"  
}
```

7 Run npm run dev in the terminal and you should see the “Hello TypeScript + JSON”.

PART 2 — TypeScript Basics in Node (3 points)

In this part, you will define a Student type and write some basic functions in TypeScript.

1. Define Student Type and Sample Data

- Edit src/index.ts to contain:

```
type Student = {  
  id: number;  
  name: string;  
  grade: number;  
  isActive: boolean;  
};  
  
// Create at least 5 students as Sample Data  
const students: Student[] = [  
  { id: 1, name: "Ann", grade: 3.5, isActive: true },  
  { id: 2, name: "Bob", grade: 2.7, isActive: false },  
];
```

2. Implement Functions:

- Below the array, implement these functions:

```
function getActiveStudents(students: Student[]): Student[] {  
    return students.filter((s) => s.isActive);  
}  
  
function calculateAverageGrade(students: Student[]): number {  
    if (students.length === 0) return 0;  
    const total = students.reduce((sum, s) => sum + s.grade, 0);  
    return total / students.length;  
}
```

3. At the bottom, log the results:

```
console.log("All students:", students);  
  
console.log("Active students:", getActiveStudents(students));  
  
console.log("Average grade:", calculateAverageGrade(students));
```

4. Run the file:

```
npm run dev
```

5. Checkpoint Questions (write your answers in text)

Answer these in a separate text/Word file or at the end of your lab report:

- In your own words, explain how filter is used in getActiveStudents.
- In your own words, explain how reduce is used in calculateAverageGrade.
- What TypeScript error do you get if you remove isActive from one of the students in the array?
 - Why is this error helpful before running the code?

PART 3 — JSON String → Typed Objects (2 points)

In this part, you will simulate an API response using a JSON string and map it to your Student type.

1. Create new file: src/json-string.ts

2. Add the following content

```
type Student = {
    id: number;
    name: string;
    grade: number;
    isActive: boolean;
};

const studentsJson = `[
    { "id": 1, "name": "Ann", "grade": 3.5, "isActive": true },
    { "id": 2, "name": "Bob", "grade": 2.7, "isActive": false },
    { "id": 3, "name": "Chen", "grade": 3.9, "isActive": true },
    { "id": 4, "name": "Dai", "grade": 1.9, "isActive": true },
    { "id": 5, "name": "Eve", "grade": 2.0, "isActive": false }
]`;

// Parse JSON to Object
const students: Student[] = JSON.parse(studentsJson);

function getActiveStudents(students: Student[]): Student[] {
    return students.filter((s) => s.isActive);
}

function calculateAverageGrade(students: Student[]): number {
    if (students.length === 0) return 0;
    const total = students.reduce((sum, s) => sum + s.grade, 0);
    return total / students.length;
}

console.log("Students from JSON:", students);
console.log("Active students:", getActiveStudents(students));
console.log("Average grade:", calculateAverageGrade(students));

// Convert Object to JSON
const activeStudentsJson =
JSON.stringify(getActiveStudents(students), null, 2);
console.log("Active students JSON:\n", activeStudentsJson);
```

3. Temporarily change your dev script (or run ts-node directly):

- a. Option 1 (Change script):

```
"dev": "ts-node src/json-string.ts"
```

Then run npm run dev

- b. Option 2 (run directly)

```
npx ts-node src/json-string.ts
```

You should see the full list of students, active students, average grade, and formatted JSON of active students.

4. Checkpoint Questions (Write your answers)

1. In `JSON.stringify(getActiveStudents(students), null, 2)`, what does each argument mean?
 1. First argument:
 2. Second argument (`null`):
 3. Third argument (`2`):
2. If the JSON string is missing the `grade` field for one student, what could happen when you parse and use it in TypeScript?

PART 4 — JSON files and fs in Node (2 points)

Now you will move from JSON strings to actual JSON files on disk.

1. Create a data folder in the project root
2. Inside data, create `students.json` with the following content:

```
[  
  { "id": 1, "name": "Ann", "grade": 3.5, "isActive": true },  
  { "id": 2, "name": "Bob", "grade": 2.7, "isActive": false },  
  { "id": 3, "name": "Chen", "grade": 3.9, "isActive": true },  
  { "id": 4, "name": "Dai", "grade": 1.9, "isActive": true }  
]
```

3. Read, Process, and Write JSON File

- Create src/json-file.ts with the following content:

```
import * as fs from "fs";

type Student = {
    id: number;
    name: string;
    grade: number;
    isActive: boolean;
};

const raw = fs.readFileSync("data/students.json", "utf-8");
const students: Student[] = JSON.parse(raw);

const topStudents = students.filter((s) => s.grade >= 3.0);
console.log("Top students:", topStudents);

const topJson = JSON.stringify(topStudents, null, 2);
fs.writeFileSync("data/top-students.json", topJson);

console.log("top-students.json written!");
```

Make sure you run this from the project root folder, so "data/students.json" is a correct relative path.

- Run the file:

```
npx ts-node src/json-file.ts
```

- Open data/top-students.json and verify the JSON structure is correct.

PART 5 — Wrap up Questions (3 Points)

Answer the following questions (in the same document where you wrote previous answers):

1. What is the difference between a JavaScript object in a .ts/.js file and JSON in a .json file?
2. Why is it better to use a specific type like Student instead of any when working with parsed JSON data?
3. What are the two main functions used to convert between JSON text and JavaScript/TypeScript objects?

PART 6 — Deliverables (Screenshots to Submit)

Students must submit:

1. Screenshot 1 – Project Structure
2. Screenshot 2 – Part 1 Output
 - Terminal showing npm run dev executing src/index.ts with: Hello TypeScript + JSON
3. Screenshot 3 – Part 2 Output
 - Terminal output showing:
 1. All students: ...
 2. Active students: ...
 3. Average grade: ...
4. Screenshot 4 – Part 3 Output
 - Terminal output from running src/json-string.ts, showing:
 1. Students from JSON: ...
 2. Active students: ...
 3. Average grade: ...
 4. Active students JSON: ... (pretty printed)
5. Screenshot 5 – Part 4 Output + Files
 - Terminal output from src/json-file.ts showing:
 1. Top students: ...
 2. top-students.json written!
 - And screenshot of the opened data/top-students.json file.
6. (Optional Bonus)
 - Screenshot of VS Code Source Control showing a commit like: "Lab03: TypeScript + JSON"
 - GitHub repository URL (if you push your project).
 - This can give up to 0.5 bonus point.