Angular TS – Data Structures in TypeScript

[**Task 2: Interfaces & Enums — Data Structures** 1](#_Toc205974063)

[**1. Purpose** 1](#_Toc205974064)

[**2. Theory** 1](#_Toc205974065)

[**3. Prerequisites** 2](#_Toc205974066)

[**4. Code Example — task2-interfaces-enums.ts** 2](#_Toc205974067)

[**5. Build & Run Procedure** 3](#_Toc205974068)

[**6. Summary** 3](#_Toc205974069)

**Task 2: Interfaces & Enums — Data Structures**

**1. Purpose**

To learn how to define and use **interfaces** for structured data and **enums** for named constants in TypeScript, improving code clarity and maintainability.

**2. Theory**

**Interfaces**

* A way to define the shape of an object (its properties and their types).
* Used to enforce consistency across objects.
* Supports optional properties (?) and readonly properties (readonly).

**Example shape:**

interface Person {

name: string;

age: number;

}

**Enums**

* A set of named constants that represent possible values.
* Makes code more readable and prevents using magic strings/numbers.

**Example:**

enum Direction {

North,

South,

East,

West

}

**3. Prerequisites**

* You must have **Node.js** and **TypeScript** installed.
* You already have your typescript-fundamentals folder from Task 1.

**4. Code Example — task2-interfaces-enums.ts**

// Interface example

interface Student {

id: number;

name: string;

course: string;

isActive?: boolean; // optional property

}

// Enum example

enum CourseCategory {

Programming = "Programming",

Design = "Design",

Marketing = "Marketing"

}

// Using Interface & Enum

let student1: Student = {

id: 1,

name: "Kiran",

course: CourseCategory.Programming

};

let student2: Student = {

id: 2,

name: "John",

course: CourseCategory.Design,

isActive: true

};

// Function to display student info

function printStudentDetails(student: Student): void {

console.log(`ID: ${student.id}`);

console.log(`Name: ${student.name}`);

console.log(`Course: ${student.course}`);

console.log(`Active: ${student.isActive ? "Yes" : "No"}`);

console.log("--------------------");

}

printStudentDetails(student1);

printStudentDetails(student2);

**5. Build & Run Procedure**

In your terminal:

cd C:\DotNet\JIRARecap\AngularTS\typescript-fundamentals

tsc task2-interfaces-enums.ts

node task2-interfaces-enums.js

**Explanation:**

* tsc file.ts → compiles TypeScript into JavaScript.
* node file.js → executes the JavaScript output.

**6. Summary**

* **Interfaces** help define object structure, ensuring consistency.
* **Enums** group related constants under a readable name.
* Both improve **type safety** and **readability** in TypeScript projects.