TypeScript with NestJs Intermediate Training

Curriculum 3 Days

Day - 1

- **Getting Started with TypeScript**
  - Course introduction
  - Why use TypeScript
  - TypeScript Features
  - TypeScript Syntax, Keywords, and Code Hierarchy
  - Tooling and Framework Options
  - Tooling and Framework Options - TypeScript Playground
  - Tooling and Framework Options - Visual Studio
  - Tooling and Framework Options - Web Advances
  - Tooling and Framework Options - Sublime Text
  - Tooling and Framework Options - TypeScript Compiler
  - Tooling and Framework Options - NodeJS
  - Hello World Example
  - Hello World Example - Creating a Class
- **Typing, Variables, and Functions**
  - Overview
  - Grammar, Declarations, and Annotations
  - Type Inference
  - Grammar
  - Static and Dynamic Typing
  - Compile Time or Run Time
  - Ambient Declarations and Type Definition Files
  - The Any Type and Primitives
  - Applying Types
  - Objects
  - Functions
  - Arrow Functions and Debugging
  - Functions and Interfaces
  - Static Typing Recap
- **Classes and Interfaces**
  - Introduction
  - Defining Classes
  - Demo: Defining Classes
  - Demo: Property Limitations
  - Casting and Type Definition Files
  - Demo: Casting and Type Definition Files
  - Extending Types
  - Demo: Extending Types
  - Using Interfaces
  - Demo: Using Interfaces
  - Extending an Interface
  - Demo: Extending an Interface
- **Modules**
  - Overview
  - Identifying a Module
  - Creating an Internal Module
  - Internal Module Accessibility and IIFE
  - Named Modules
  - Extending Modules and Importing Shortcuts
  - Organizing Internal Modules
  - Separating Internal Modules
  - External Modules and Dependency Resolution
  - Module Dependencies
  - Importing External Modules Using AMD
  - Importing 3rd Party Libraries Using AMD
  - Modules Recap

Day - 2

- **Getting Started - NestJS**
  - Introduction to NestJS
  - Installing the NestJS CLI (command-line interface)
  - What's inside a NestJS Application
  - What we'll be building in this course
  - Beginning your NestJS Journey
- **Creating a REST API application**
  - Install Insomnia
  - Running NestJS in Development Mode
  - Creating a Basic Controller
  - Use Route Parameters
  - Handling Request Body / Payload
  - Response Status Codes
  - Handling Update and Delete Requests
  - Implement Pagination with Query Parameters
  - Creating a Basic Service
  - Send User-Friendly Error Messages
  - Encompass Business-Domain in Modules
  - Introduction to Data Transfer Objects
  - Validate Input Data with Data Transfer Objects
  - Handling Malicious Request Data
  - Auto-transform Payloads to DTO instances
- **Add PostgreSQL with TypeORM**
  - Before we Get Started
  - Install Docker
  - Running PostgreSQL
  - Introducing the TypeORM Module
  - Creating a TypeORM Entity
  - Use Repository to Access Database
  - Create a Relation between two Entities
  - Retrieve Entities with their Relations
  - Using Cascading Inserts and Updates
  - Adding Pagination
  - Use Transactions
  - Adding Indexes to Entities
  - Setting up Migrations
- **Dependency Injection**
  - Understand Dependency Injection
  - Control NestJS Module Encapsulation
  - Diving Into Custom Providers
  - Value-based Providers
  - Non-class-based Provider Tokens
  - Class Providers
  - Factory Providers
  - Leverage Async Providers
  - Create a Dynamic Module
  - Control Providers Scope
  - Diving Deeper Into Request-Scoped Providers

Day - 3

- **Application Configuration**
  - Introducing the Config Module
  - Custom Environment File Paths
  - Schema Validation
  - Using the Config Service
  - Custom Configuration Files
  - Configuration Namespaces and Partial Registration
  - Asynchronously Configure Dynamic Modules

- **Other Building Blocks by Example**
  - Introducing More Building Blocks
  - Understanding Binding Techniques
  - Catch Exceptions with Filters
  - Protect Routes with Guards
  - Using Metadata to Build Generic Guards or Interceptors
  - Add Pointcuts with Interceptors
  - Handling Timeouts with Interceptors
  - Creating Custom Pipes