Training on NodeJS, TypeScript and Angular

Course Objectives:

1. Gain a comprehensive understanding of NodeJS and TypeScript, including asynchronous programming, file system operations, database connectivity, security, and error handling.
2. Build RESTful APIs using ExpressJS, implement authentication and authorization, and secure NodeJS applications against common vulnerabilities.
3. Master TypeScript fundamentals, advanced features like interfaces, classes, generics, and decorators, and integrate TypeScript into NodeJS projects.
4. Develop web applications using Angular, including components, data binding, services, routing, forms, and HTTP communication with RESTful APIs.
5. Learn Angular best practices for code organization, performance optimization, and scalability to build maintainable and efficient applications.

Prerequisites:

1. Basic knowledge of JavaScript and its core concepts (variables, functions, arrays, objects, etc.).
2. Familiarity with HTML, CSS, and DOM manipulation for building web applications.
3. Prior experience with any programming language will be beneficial but not mandatory.

Software Setup Required on Student's Laptop:

1. NodeJS and npm:
    - Download and install the latest version of NodeJS from the official website (https://nodejs.org).
    - npm (Node Package Manager) will be installed automatically with NodeJS.
2. Angular CLI:
    - Install Angular CLI globally using npm by running the following command in the terminal or command prompt:
      ```
      npm install -g @angular/cli
      ```
3. Text Editor or IDE:
    - Choose a preferred text editor or Integrated Development Environment (IDE) for writing code. Recommended options include **Visual Studio Code**, Sublime Text, Atom, or WebStorm.
4. MongoDB (Optional):
    - If the training covers MongoDB database operations, you may choose to install MongoDB Community Edition on your local machine (https://www.mongodb.com/try/download/community). Alternatively, you can connect to a remote MongoDB instance during the training.
5. Web Browser:

    - Any modern web browser like Google Chrome, Mozilla Firefox, or Microsoft Edge.
6. Command Line Interface (CLI):
    - Make sure your laptop has a command-line interface (Command Prompt on Windows or Terminal on macOS/Linux) for running NodeJS, Angular CLI commands, and other terminal operations.
7. Git (Optional):
    - If the course involves version control using Git, install Git on your laptop (https://git-scm.com/downloads).

Note: The software setup may vary depending on the specific training materials and projects provided by the instructor or training provider. It's essential to follow the trainer's instructions or refer to any provided course documentation for any additional setup requirements.

Course duration:

- 20 Days (9:30 AM to 1:00 PM)

Day 1: Introduction to NodeJS

- Introduction to NodeJS and its features
    - What is NodeJS?
    - Advantages of using NodeJS for server-side development
- Setting up NodeJS environment
    - Installing NodeJS and npm
    - NodeJS project structure
- Creating a basic NodeJS application
    - Writing a "Hello World" program
    - Running a NodeJS script
- Understanding modules and npm
    - CommonJS modules and their usage
    - Installing packages using npm
    - Managing dependencies with package.json

Day 2: Asynchronous Programming in NodeJS

- Understanding asynchronous programming
    - Synchronous vs. Asynchronous code execution
    - The event loop and non-blocking I/O
- Working with callbacks and event-driven architecture
    - Using callbacks for asynchronous operations
    - EventEmitter and handling events
- Promises and async/await in TypeScript
    - Introduction to Promises
    - Chaining Promises and handling errors
    - Using async/await for asynchronous code

Day 3: Working with ExpressJS

- Introduction to ExpressJS framework

- Creating a basic Express application
- Handling routes and HTTP methods

- Building RESTful APIs with Express
  - Designing RESTful APIs
  - Implementing CRUD operations
- Middleware in Express
  - Using middleware for request/response processing
  - Creating custom middleware functions

Day 4: File System and Streams

- Working with the File System module
  - Reading and writing files synchronously and asynchronously
  - Working with directories and file paths
- Understanding streams for efficient data processing
  - Readable and writable streams
  - Piping data between streams
- Uploading and downloading files
  - Handling file uploads with multer
  - Downloading files from the server

Day 5: Database Connectivity

- Introduction to databases (focus on MongoDB)
  - Overview of NoSQL vs. SQL databases
  - Setting up and connecting to a MongoDB database
- Connecting to MongoDB from NodeJS using Mongoose
  - Creating Mongoose schemas and models
  - Performing CRUD operations with Mongoose
- CRUD operations using Mongoose: Create, Read, Update, Delete
  - Writing queries using Mongoose
  - Updating and deleting documents in MongoDB

Day 6: Error Handling and Debugging

- Implementing error handling in NodeJS
  - Handling synchronous errors
  - Implementing error middleware in Express
- Debugging NodeJS applications with built-in tools
  - Using console.log and console.debug for debugging
  - Debugging with Node Inspector
- Using logging and debugging tools: Winston, Debug, etc.
  - Integrating Winston for logging
  - Using the Debug module for better debugging

Day 7: Security and Authentication

- Common security threats in NodeJS applications
  - Cross-Site Scripting (XSS) and Cross-Site Request Forgery (CSRF)
  - SQL Injection and NoSQL Injection

- Implementing authentication and authorization with Passport
  - Setting up Passport for local authentication
  - Using Passport strategies for OAuth providers
- Introduction to JSON Web Tokens (JWT) and token-based authentication
  - Creating and verifying JWTs
  - Protecting routes with JWT authentication

Day 8: TypeScript Fundamentals

- Introduction to TypeScript and its benefits
  - What is TypeScript and how it enhances JavaScript
  - TypeScript configuration and tsconfig.json
- Basic types and type annotations
  - Primitive types: number, string, boolean, etc.
  - Type annotations for variables and function parameters

Day 9: Advanced TypeScript

- Interfaces and classes in TypeScript
  - Creating interfaces and using them for type-checking
  - Implementing classes and inheritance in TypeScript
- Generics and their usage
  - Creating generic functions and classes
  - Working with generic types in TypeScript
- Decorators and metadata introspection
  - Understanding decorators and their applications
  - Implementing custom decorators for logging, validation, etc.

Day 10: Finalizing NodeJS and TypeScript Integration

- Integrating TypeScript into NodeJS project
  - Setting up TypeScript in a NodeJS application
  - Compiling TypeScript to JavaScript
- Migrating JavaScript code to TypeScript
  - Converting JavaScript files to TypeScript
  - Handling common migration issues and challenges
- Tools and resources for TypeScript development
  - Useful TypeScript tools and libraries
  - Recap and Q&A for NodeJS and TypeScript topics

Day 11: Introduction to Angular

- Introduction to Angular and its features
  - What is Angular and its advantages
  - Understanding Angular architecture: Modules, Components, Services, etc.
- Setting up Angular development environment
  - Installing Angular CLI
  - Creating a new Angular project using CLI
- Creating a basic Angular application
  - Building a simple Angular component

- Rendering data in the template

Day 12: Components and Data Binding

- Understanding Angular components and their role
  - Anatomy of an Angular component
  - Creating and nesting components
- Data binding: Interpolation, Property binding, and Event binding
  - Using interpolation to display data
  - Binding component properties to the template
  - Handling user interactions with event binding

Day 13: Services and Dependency Injection

- Introduction to services and their significance
  - What are Angular services and why are they important?
  - Creating and registering services in Angular
- Implementing dependency injection in Angular
  - Understanding hierarchical injectors
  - Injecting services into components and other services
- Using services for data sharing and communication
  - Sharing data between components using services
  - Implementing a shared data service

Day 14: Routing and Navigation

- Setting up Angular routing
  - Configuring routes and route parameters
  - Defining route paths and route guards
- Implementing navigation between components
  - Navigating programmatically using the Router service
  - Using router links for navigation in templates
- Route parameters and optional parameters
  - Extracting route parameters in components
  - Handling optional route parameters

Day 15: Forms and Validation

- Working with template-driven and reactive forms
  - Creating template-driven forms with ngForm
  - Implementing reactive forms with FormBuilder
- Implementing form validation in Angular
  - Using built-in validators and custom validation functions
  - Displaying validation errors in the template
- Handling form submission and form reset
  - Submitting form data and handling form events
  - Resetting form values and states

Day 16: HTTP and Observables

- Making HTTP requests with Angular's HttpClient module

    - Configuring HttpClientModule for API communication
    - Sending GET, POST, PUT, DELETE requests
- Handling asynchronous operations with Observables
    - Understanding Observables and their operators
    - Subscribing to Observables and handling responses
- Error handling in HTTP requests using catchError
    - Handling HTTP errors and displaying error messages
    - Implementing retry and fallback strategies

Day 17: Angular and RESTful APIs

- Integrating Angular with backend RESTful APIs
- Creating a service to interact with backend APIs
- Mapping API responses to Angular models
- CRUD operations with Angular and HTTP
    - Implementing Create, Read, Update, Delete operations
    - Interacting with backend APIs for data manipulation
- Handling responses and errors from APIs
    - Processing API responses and updating the UI
    - Handling API errors and displaying meaningful messages
- Communication with backend services
    - Using HttpClient Interceptors for request/response handling
    - Implementing loaders and progress indicators

Day 18: Angular Modules and Lazy Loading

- Understanding Angular modules and their organization
    - Creating feature modules for better organization
    - Declarative vs. Shared modules
- Feature modules and shared modules
    - Creating and using feature modules
    - Sharing common functionality with shared modules
- Implementing lazy loading for optimizing application performance
    - Lazy loading feature modules for better initial page load time
    - Lazy loading preloading strategies for improved user experience

Day 19: Angular Best Practices

- Common best practices in Angular development
    - Writing clean and maintainable code
    - Performance optimization techniques
- Code organization and maintainability
    - Folder structure and naming conventions
    - Separation of concerns and Single Responsibility Principle (SRP)
- Reusability and scalability

  - Creating reusable components, services, and directives
  - Using NgRx for state management and scalability

Day 20: Final Project and Recap

- Developing a small project using Angular
  - Applying the concepts learned throughout the course
  - Building a complete application with CRUD functionality
- Recap and Q&A for Angular topics
  - Reviewing key concepts and important topics
  - Addressing any remaining doubts or questions
- Final assessment and closing remarks
  - Evaluating participants' understanding and progress
  - Concluding the training with final thoughts and suggestions