TypeScript Syllabus for Fullstack Web Development

1. Introduction to TypeScript

- What is TypeScript?
- History and Evolution
- Why Use TypeScript?
  - Benefits over JavaScript
  - Type Safety & Static Typing
  - Readability & Maintainability
  - Improved Developer Experience
- JavaScript vs TypeScript
  - Key Differences
  - Pros and Cons
- Course Prerequisites
- Use Cases: Frontend vs Backend

2. Setting Up TypeScript

- Installing TypeScript
  - `npm install -g typescript`
  - Verifying: `tsc --version`
- TypeScript Compiler Setup
  - `tsconfig.json`, compiler options
- Running TS Files: `tsc file.ts`, `tsc -w`
- Using with Node.js: `ts-node`, `tsconfig-paths`
- Static vs Dynamic Typing
- Code Execution Flow
- Transpilation & Compilation
- TypeScript + HTML
- Using TypeScript with:
  - Webpack
  - Vite
  - Parcel

3. TypeScript Basics

- Type Inference & Annotation
- Core Types:
  - `string`, `number`, `boolean`, `object`, `any`, `unknown`, `never`, `undefined`, `null`
  - Literal Types, Union (`|`), Intersection (&)
- Arrays, Tuples (optional, readonly)
- Type Aliases
- Enums (`enum`, `const enum`)
- Type Assertions: `as`, `<Type>`
- Functions:
  - Return Types

    - Optional, Default, and Rest Parameters
    - Arrow Functions
    - Function Overloading

4. Object-Oriented Programming

- Classes & Constructors
- Public / Private / Protected
- Readonly, Getters & Setters
- Inheritance & Overriding
- Abstract Classes
- Interfaces vs Abstract Classes
- Static Methods & Props
- `instanceof` Usage

5. Interfaces & Type System

- Interfaces vs Type Aliases
- Optional & Readonly Props
- Extending Interfaces
- Interface Merging
- Function Types with Interfaces
- Index Signatures
- Structural Typing

6. Decorators

- What Are Decorators?
- Enabling Decorators via `tsconfig.json`
- Decorator Factories
- Types of Decorators:
    - Class
    - Property
    - Method
    - Accessor
    - Parameter
- Real-World Examples:
    - Angular
    - NestJS
    - Logging, Metadata, DI

7. Advanced TypeScript

- Generics:
    - Functions, Interfaces, Classes
    - Constraints and Defaults
- Type Guards:
    - `typeof`, `instanceof`, Custom Guards
- Discriminated Unions
- Mapped Types
- Conditional Types

- Utility Types:
  - `Partial`, `Required`, `Readonly`, `Pick`, `Omit`, `Record`, `Exclude`, `Extract`, `NonNullable`
- Template Literal Types
- `keyof`, `typeof`, `in` Usage

8. Modules & Namespaces

- ES Modules:
  - Named / Default Exports
  - Import Variants
- Namespaces (legacy use)
- Module Resolution
- Path Aliases

9. Error Handling & Debugging

- `try` / `catch` / `finally`
- Custom Errors
- Type Narrowing with Errors
- Debugging TS in VSCode
- Source Maps

10. TypeScript in Frontend Frameworks

- **React + TS:**
  - `React.FC`, Props, State
- **Angular + TS:**
  - Decorators, DI, Modules
- **Vue + TS:**
  - Composition API, SFCs
- Project Setup:
  - Vite + TS
  - CRA + TS
  - Webpack Manual Config

11. TypeScript in Backend Development

- Node.js + TypeScript
- Express + TS Setup
- Typing Middlewares & Routes
- Validation Tools: Zod, Joi
- Working with ORMs:
  - Prisma
  - TypeORM
- Type-safe Services & Controllers
- REST + GraphQL

12. Tooling, Testing, and Best Practices

- Linting: ESLint + TypeScript Plugin

- Formatting: Prettier
- Clean Code Principles
    - Avoiding `any`
    - Modular Typing
- Testing:
    - Jest + TypeScript
    - Mocks / Type-safe Tests
- Best Practices:
    - Always use `strict`
    - Favor Interfaces
    - Use Generics for Reuse
    - Immutability
    - Type-First Development