STACK LEARNER

Welcome to

TypeScript Essentials

HM Nayem

Your Instructor 😎

What Are We Going to Do in This Workshop?

We are going to Understand

1. TypeScript's core features, including type annotations, type inference, and advanced structures like interfaces, unions, and generics.

2. How to integrate TypeScript into real-world projects & work seamlessly with third-party libraries.

3. How to write clean, maintainable, and robust TypeScript code by following industry best practices, handling common pitfalls, and utilizing utility types effectively.

Today's Agenda

1 **Why**

do we actually need TypeScript?

2 **How**

to setup environment for TypeScript?

3 **What**

are the common types and how to use them?

Let’s **Begin . . .**

⚠ TypeScript is just a development tool

⚠ Typescript = JavaScript + Type

⚠ The Job of TypeScript is Static Type Checking

Before doing anything else, let's understand the Importance of TypeScript

Step 1: **Watch me code.**

Step 2: **Watch me panic when it doesn't work!**

Real-World Benefits of TypeScript

Improved Developer Productivity

Error Detection at Compile-Time

TypeScript’s static type system catches errors during development rather than at runtime, significantly reducing the risk of bugs. This helps developers avoid common issues such as null or undefined errors and mismatched types.

Better Autocompletion and Tooling

With TypeScript, IDEs like Visual Studio Code offer rich autocompletion, method suggestions, and inline documentation, making it easier and faster for developers to write and maintain code.

Enhanced Code Quality

Type Safety

TypeScript ensures that variables and functions are used correctly according to the defined types. This minimizes the chance of passing the wrong data type, leading to more predictable and stable applications.

Refactoring

TypeScript makes refactoring easier and safer, as the type system ensures that changes are consistent across the entire codebase. When renaming variables or changing function signatures, TypeScript will flag any inconsistencies, preventing errors.

Better Maintainability

Clearer Codebase

With TypeScript's strong typing, it's easier to understand what each part of the code is doing. The types serve as an effective form of documentation, making it easier for new developers to understand and work with the codebase.

Easier Debugging

TypeScript's static analysis tools allow for faster debugging. Errors are identified during development rather than during runtime, and the debugging process becomes more straightforward.

Improved Collaboration and Teamwork

Explicit Contracts

TypeScript’s types act as an explicit contract between different parts of the application or between different team members. This clarity helps in communicating how data should flow and reduces misunderstandings between developers working on various parts of a project.

Consistency Across the Codebase

With type annotations, TypeScript enforces consistency in how data structures are used across the team. This leads to a more uniform approach to development, helping prevent bugs caused by inconsistent data usage.

Scalability

Supports Large Codebases

TypeScript is especially useful in large-scale applications where managing a large codebase with JavaScript becomes unwieldy. TypeScript helps scale projects by catching issues early and allowing developers to handle complex code more easily.

Modular Development

TypeScript's strong typing makes it easier to divide a large project into smaller, well-defined modules, improving collaboration and team productivity.

Type Inference and Automatic Type Checking

Minimal Type Annotations

TypeScript provides powerful type inference, where it can deduce types automatically based on the values assigned to variables or the return types of functions. Developers can write less boilerplate code while still benefiting from type safety.

Auto-Completion

TypeScript’s type system enables better auto-completion, reducing the chances of making syntactic errors, improving developer speed, and making the code more self-explanatory.

Better Integration with Modern JavaScript Frameworks

Seamless Integration

TypeScript works smoothly with popular JavaScript frameworks like React, Angular, and Vue.js, improving the development experience with better tooling support. These frameworks have adopted TypeScript because of its type safety and ease of integration into large-scale applications.

Strict Mode in JavaScript

TypeScript's type system allows for stricter checks in JavaScript, enforcing best practices and reducing the chances of making mistakes.

Backward Compatibility with JavaScript

No Need for Rewriting

TypeScript is a superset of JavaScript, meaning that all JavaScript code is valid TypeScript. Teams can gradually adopt TypeScript in their existing JavaScript codebase without needing to rewrite everything at once. This allows for incremental adoption of TypeScript while still benefiting from its features.

Enhanced Documentation

Self-Documenting Code

With TypeScript's static type system, code is inherently more self-documenting. Types provide a clear understanding of what kind of data is expected, making the code more readable and easier to maintain.

Type Definitions for Libraries

Many third-party libraries offer type definitions in TypeScript, improving the development experience when working with external packages.

Community and Ecosystem Support

Growing Ecosystem

TypeScript is widely adopted across the industry and has an extensive ecosystem of libraries, frameworks, and tools that support it. The availability of community-driven type definitions (via DefinitelyTyped) for JavaScript libraries makes it easier to work with non-TypeScript libraries.

Active Development and Maintenance

TypeScript is actively maintained by Microsoft and has a strong and growing developer community. This ensures ongoing improvements, bug fixes, and new features to enhance developer productivity.

Setup the Development Environment

STEP 1

Create a **npm** project

STEP 2

Install the following dependencies:

- `typescript`: TypeScript compiler and essential tooling.
- `ts-node`: A TypeScript execution engine for Node.js, useful for running TypeScript directly.
- `ts-node-dev`: A faster version of `ts-node` , useful for development as it supports hot-reloading.
- `tsc-alias`: A package to handle custom path aliases.
- `tsconfig-paths`: A helper for resolving custom TypeScript path aliases.

STEP 3

Install `@types/node` for built in node modules types

STEP 4

Create a ***tsconfig.json*** file

STEP 5

Create the `src/index.ts` file

STEP 6

Update the npm scripts

STEP 7

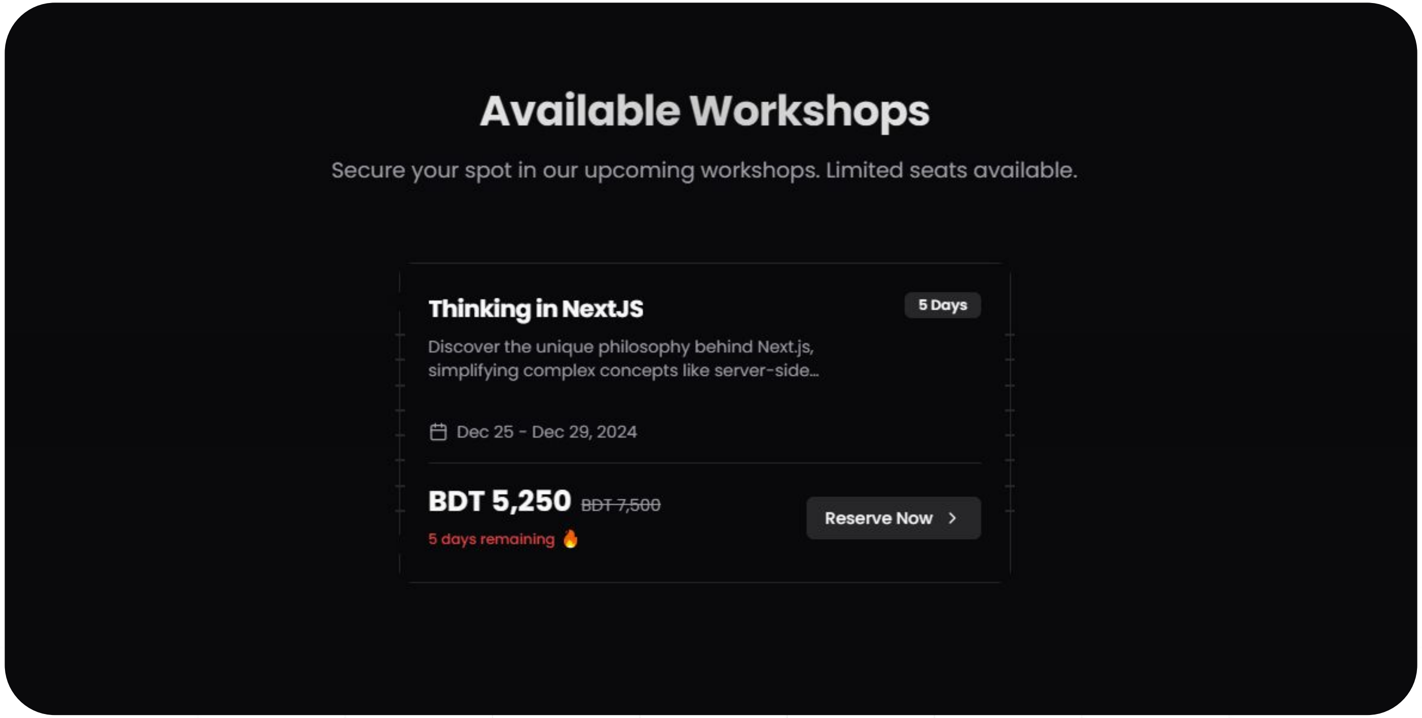
Run and test

STEP 8

Add a debug launch script `.vscode/launch.json`

Hands On Example

Our Upcoming Workshop



Available Workshops

Secure your spot in our upcoming workshops. Limited seats available.

Thinking in NextJS

5 Days

Discover the unique philosophy behind Next.js, simplifying complex concepts like server-side...

Dec 25 - Dec 29, 2024

BDT 5,250 ~~BDT 7,500~~

5 days remaining 🔥

Thank You!