SOFTWARE DEVELOPMENT AND ENGINEERING PRACTICES

TypeScript for JavaScript Developers

Level: Practitioner • 2 days (expandable to 3) • Virtual, In-person

Overview

TypeScript has become the default for serious JavaScript work, on the front end and the back end, because it catches whole classes of bugs before the code runs and makes large codebases navigable. But TypeScript rewards a specific way of thinking: its type system is structural and unusually expressive, and developers who treat it as "JavaScript with annotations" end up fighting the compiler, sprinkling any everywhere, and getting little of the safety they adopted it for.

This is a hands-on, practitioner course. It builds the type system in dependency order: first the mental model of structural typing and inference, then the everyday work of typing functions, objects, and unions, then generics, the feature that unlocks the rest of the language, and finally the practical craft of configuring, migrating, and living with TypeScript in real projects. Rather than catalog every type-system feature, it goes deep on the subset professional TypeScript is actually made of. Every module ends with a lab and builds on the one before.

Who Should Attend

- JavaScript developers adopting TypeScript for new or existing projects
- Front-end developers whose framework work now assumes TypeScript
- Node.js developers who want type safety on the server

Developers whose JavaScript predates ES6 should take *Modern JavaScript (ES6+)* first.

Prerequisites

- Solid modern JavaScript, including ES modules, destructuring, and async/await
- Comfortable with npm and a code editor, ideally VS Code
- No prior TypeScript experience required

What You Will Learn

- Explain how TypeScript's structural typing and inference actually work
- Type functions, objects, and data structures without fighting the compiler
- Model real-world data with unions, literal types, and narrowing
- Write and read generic functions and types with confidence
- Use utility types and type manipulation to keep types DRY and honest
- Configure a TypeScript project and migrate an existing JavaScript codebase incrementally

Course Outline

Day one: thinking in types

- The TypeScript Mental Model
  - What TypeScript is: a compile-time layer that erases at runtime
  - Structural typing: shapes, not names

  - Inference first: letting the compiler do the work, annotating where it counts
  - Lab: set up a TypeScript project and use the compiler to find real bugs in a working JavaScript module
- Typing Everyday Code
  - Primitives, arrays, objects, and type aliases versus interfaces
  - Typing functions: parameters, returns, optionals, and overloads worth avoiding
  - Strictness options, and why any is a loan you must repay
  - Lab: fully type a small untyped codebase with zero any and strict mode on
- Unions, Literals, and Narrowing
  - Union types and literal types: modeling the values that can actually occur
  - Narrowing: typeof, in, and discriminated unions
  - Making illegal states unrepresentable, the payoff of good type design
  - Lab: model a realistic API response as a discriminated union and handle every case the compiler demands

Day two: generics, real projects, and adoption

- Generics
  - Why generics exist: types that keep their relationships
  - Generic functions, generic types, and constraints with extends
  - Reading the generic signatures in library code without flinching
  - Lab: build a small typed utility library where generics carry the types all the way through
- Types at Scale
  - Utility types: Partial, Pick, Omit, Record, and ReturnType in daily use
  - keyof and indexed access: deriving types instead of duplicating them
  - Typing third-party code: bundled types, DefinitelyTyped, and declaration files
  - Lab: refactor duplicated types into derived ones and type an untyped third-party dependency
- TypeScript in Real Projects
  - tsconfig that makes sense: the options that matter and sane defaults
  - Incremental migration: taking a JavaScript codebase to TypeScript without stopping the world
  - TypeScript with frameworks and toolchains: where it fits in React and Node.js builds
  - Lab: migrate a working JavaScript application to TypeScript module by module, keeping it running throughout

Extended Version

The three-day version keeps the same gradient and adds depth on the advanced layer:

- Advanced type manipulation: conditional types, mapped types, and template literal types, with judgment about when they are worth it
- Runtime validation at the boundaries: pairing static types with schema validation
- Typing asynchronous code and API clients end to end
- A capstone: migrate and harden a complete small application, then defend its type design decisions