Guides

TypeScript Fundamentals

Core TypeScript concepts for Java developers transitioning to web development. Start here if you're new to TypeScript.

A Brief History

JavaScript: 10 Days That Changed the Web

JavaScript was created in 1995 by Brendan Eich at Netscape Communications — famously in just 10 days. Originally called Mocha, then LiveScript, it was renamed JavaScript as a marketing play to ride Java's popularity (despite having almost nothing in common with Java beyond curly braces).

The early web needed client-side interactivity. Before JavaScript, every user action required a full page reload from the server. JavaScript changed that — form validation, dynamic content, and responsive interfaces became possible without round-trips.

JavaScript grew fast. The language was standardized as **ECMAScript** in 1997, and major versions followed:

Version	Year	Key Additions
ES5	2009	`strict mode`, JSON, array methods (`forEach`, `map`, `filter`)
ES6/ES2015	2015	`let` / `const`, arrow functions, classes, Promises, modules
ES2017	2017	`async` / `await`
ES2020	2020	Optional chaining (`?.`), nullish coalescing (`??`)

In 2009, Ryan Dahl created **Node.js**, bringing JavaScript to the server. Suddenly one language could power both the browser and the back end. The Node.js ecosystem exploded — npm now hosts over 2 million packages, and JavaScript consistently ranks as the most-used programming language in developer surveys.

TypeScript: JavaScript That Scales

By the early 2010s, JavaScript was being used to build large, complex applications — but the language's dynamic typing made it hard to catch bugs early, refactor safely, or navigate large codebases. Microsoft saw the problem and in **2012**, Anders Hejlsberg (the creator of C# and Turbo Pascal) released **TypeScript**.

TypeScript is a **superset of JavaScript** — every valid JavaScript program is also a valid TypeScript program. TypeScript adds:

- **Static type annotations** — catch type errors at compile time, not runtime
- **Interfaces and type aliases** — describe the shape of data
- **Generics** — write reusable, type-safe code
- **IDE support** — autocomplete, inline errors, and safe refactoring

TypeScript compiles down to plain JavaScript — the browser and Node.js never see the types. This means you get the safety of a typed language during development with zero runtime cost.

Adoption has been massive. As of 2024, TypeScript is used by:

- **Microsoft** — VS Code, Azure, Office 365
- **Google** — Angular (built in TypeScript from the start)
- **Airbnb, Slack, Stripe, Shopify** — large-scale web applications
- **Every major framework** — React, Next.js, Express, Prisma all have first-class TypeScript support

Why TypeScript for TCSS 460?

You already know Java — a statically typed, object-oriented language. TypeScript bridges the gap:

Concept	Java	TypeScript
Type system	Static, nominal	Static, structural
Compilation	javac → bytecode → JVM	tsc → JavaScript → Node.js/browser
Null safety	`NullPointerException` at runtime	`strictNullChecks` at compile time

Concept	Java	TypeScript
Interfaces	`implements` keyword	Structural ("duck typing")
Generics	`List<String>`	`Array<string>`
Package manager	Maven/Gradle	npm
Build tool	javac/Maven	tsc/esbuild/SWC

TypeScript gives you the type safety you're used to from Java, while introducing you to the JavaScript ecosystem that powers modern web development. In this course, every project — from the Express API to the Next.js front end — is written in TypeScript.

Sources and Further Reading ›

Guides

These guides are your reference throughout the quarter. Start with the Fundamentals, then explore Async and Tools as needed.

Fundamentals

The core language — types, functions, objects, modules, and array methods. Work through these in order during Week 1.

- JavaScript for Java Developers — Bridge your Java knowledge to JavaScript
- TypeScript Essentials — Type annotations, interfaces, generics
- Building & Running TypeScript — Compiler, tsconfig, ts-node
- Objects, Arrays & Destructuring — Working with data in TypeScript
- Modules & Imports — ES modules, CommonJS, barrel exports
- Array Methods — `map`, `filter`, `reduce` and friends

Async Programming

JavaScript is asynchronous by nature. These guides cover the patterns you'll use constantly in Express and Next.js.

- Async Concepts — Event loop, callbacks, Promises
- Async in TypeScript — `async` / `await`, error handling, typed Promises

Tools

Your development environment and AI coding assistants.

- Node.js Setup — Install Node.js, npm, and create your first project
- VS Code / WebStorm — Editor setup and recommended extensions
- Using a Coding Agent — How to work effectively with AI tools
- AI Coding Tools Setup — Gemini CLI, Copilot, and other tools