Cloud Contraptions
TRAINING & CONSULTING

To discuss this course and customizations:
Call: 434-509-5680 or Email: sales@cloudcontraptions.com

TypeScript for C# Programmers

Class Duration

14 hours of live training delivered over 2 days.

Student Prerequisites

- Professional C# development experience
- Comfortable with C# classes, interfaces, generics, LINQ, and async/await
- No prior JavaScript or TypeScript experience required

Target Audience

This course is designed for professional C# developers moving into TypeScript — whether for front-end work, Node.js services, or full-stack roles on .NET-adjacent teams. Every topic is taught by translation: each TypeScript concept is introduced through its C# counterpart, with explicit attention to where the mapping holds, where it breaks, and where C# instincts will lead you astray.

Description

TypeScript for C# Programmers gets experienced C# developers productive in TypeScript fast by building on what they already know. The course is translation-focused throughout: it starts with the single biggest mental shift — C#'s nominal classes and interfaces versus TypeScript's structural typing, where compatibility is determined by shape rather than declaration — and uses that lens to work through generics (reified in .NET, erased in TypeScript), LINQ versus array methods like `map`, `filter`, and `reduce`, and async/await, which looks identical but runs on a single-threaded event loop rather than a thread pool.

The second day covers the idioms and ecosystem. Participants map C# nullable reference types onto `strictNullChecks`, translate enums and records into union types, literal types, and `readonly` structures, and move from namespaces and assemblies to ES modules. An ecosystem survey maps NuGet onto npm and pnpm, MSBuild onto `tsconfig` and Vite, and xUnit onto Vitest. The course confronts type erasure head-on: TypeScript types vanish at compile time, so participants learn what replaces reflection — schema

Copyright 2025 Cloud Contraptions LLC – All Rights Reserved

validation, type guards, and code generation. The class concludes with two hands-on builds: a small TypeScript web API and a small front-end exercise.

Learning Outcomes

- Translate C# class and interface designs into idiomatic structurally-typed TypeScript
- Explain structural typing and predict when two unrelated types are compatible
- Write TypeScript generics and articulate how erased generics differ from .NET's reified generics
- Replace LINQ pipelines with `map`, `filter`, `reduce`, and modern array and iterator methods
- Use async/await correctly on the event loop and contrast it with .NET's task-based concurrency
- Map C# nullable reference types onto `strictNullChecks` and handle `null` versus `undefined`
- Replace C# enums and records with union types, literal types, and `readonly` structures
- Organize code with ES modules in place of namespaces and assemblies
- Navigate the npm/pnpm ecosystem with NuGet instincts: packages, lockfiles, and versioning
- Map familiar tooling: MSBuild to `tsconfig` and Vite, xUnit to Vitest
- Explain type erasure and apply runtime alternatives to reflection such as schema validation and type guards
- Build a small TypeScript web API and a small front-end application

Training Materials

Comprehensive courseware is distributed online at the start of class. All students receive a downloadable MP4 recording of the training.

Software Requirements

- Node.js 24 LTS
- Visual Studio Code (recommended for this course) or another TypeScript-aware editor
- Permission to install npm packages and editor extensions
- Git and a free GitHub account for lab repositories

Copyright 2025 Cloud Contraptions LLC – All Rights Reserved

Cloud Contraptions
TRAINING & CONSULTING

To discuss this course and customizations:
Call: 434-509-5680 or Email: sales@cloudcontraptions.com

Training Topics

Orientation: Two Type Systems, One Goal

- Where TypeScript came from and how it relates to JavaScript
- Compile-time types over a dynamic runtime
- The C#-to-TypeScript translation mindset
- First program: from `dotnet new` to `npm create`

Classes, Interfaces, and Structural Typing

- The big mental shift: nominal versus structural typing
- Interfaces without `implements`: shape compatibility
- Classes in TypeScript: fields, accessors, access modifiers
- `type` aliases versus `interface`
- When structural typing surprises C# developers — and how to brand types when you need nominal behavior

Generics: Reified vs Erased

- Generic syntax: familiar on the surface
- .NET's reified generics versus TypeScript's erasure
- Constraints: `where T :` versus `extends`
- No `typeof(T)` at runtime: implications and workarounds
- Default type parameters and inference

LINQ to Array Methods

- `Select`/`Where`/`Aggregate` to `map`/`filter`/`reduce`
- `First`, `Any`, `All` to `find`, `some`, `every`
- Deferred execution versus eager arrays; iterators and generators
- Method chaining style and immutability habits
- Grouping and flattening: `flatMap`, `Object.groupBy`

Async/Await and the Event Loop

- Same keywords, different machine: event loop versus thread pool
- `Task<T>` to `Promise<T>`
- No `ConfigureAwait`, no thread affinity, no data races — and no parallelism without workers
- `Promise.all` versus `Task.WhenAll`
- Common pitfalls: floating promises and blocking the loop

Copyright 2025 Cloud Contraptions LLC – All Rights Reserved

Cloud Contraptions
TRAINING & CONSULTING

To discuss this course and customizations:
Call: 434-509-5680 or Email: sales@cloudcontraptions.com

Nullability

- Nullable reference types versus `strictNullChecks`
- `null` versus `undefined`: two nothings instead of one
- Optional properties and parameters
- `?.`, `??`, and narrowing — familiar operators, structural rules
- Modeling absence deliberately

Enums, Records, and Their TypeScript Counterparts

- C# enums versus TypeScript enums — and why unions of literal types usually win
- Records to `readonly` object types and immutability patterns
- Pattern matching versus discriminated unions and exhaustiveness
- `with` expressions versus spread updates

Namespaces and Assemblies to ES Modules

- File-based modules instead of namespaces
- `import`/`export` versus `using` and assembly references
- Module resolution and the `exports` field
- Project layout conventions

Ecosystem Survey: NuGet to npm/pnpm

- Packages, registries, and lockfiles
- `package.json` versus `.csproj`
- Semantic versioning and dependency hygiene
- pnpm and workspace monorepos

Tooling Mapping

- MSBuild to `tsconfig` and Vite
- The TypeScript compiler and the fast-rising native compiler (TypeScript 7 beta)
- xUnit to Vitest: test structure, assertions, mocking
- Debugging TypeScript in VS Code
- Linting and formatting: analyzers to ESLint and Prettier

Runtime Types: What Replaces Reflection

- Type erasure: where the types go at compile time
- No `GetType()`, no attributes: what that rules out
- Schema validation with Zod as the reflection substitute at boundaries

Copyright 2025 Cloud Contraptions LLC – All Rights Reserved

To discuss this course and customizations:
Call: 434-509-5680 or Email: sales@cloudcontraptions.com

- Type guards and assertion functions
- Code generation and decorators in brief

Copyright 2025 Cloud Contraptions LLC – All Rights Reserved