TypeScript

Object Foundations, Classes, Mixins, Generics, Specialist Types, Iterators, Ambients, Lib.d.ts

For larger-scale applications that target the JavaScript VM, either in browsers (e.g. Angular) or on the server and command-line tools (Node), or mobile apps (Ionic) or desktop apps (Electron), many senior developers have a desire for a more robust and comprehensive programming language compared to JavaScript, and TypeScript is the answer.

TypeScript is a JavaScript-like language that transpiles to JavaScript so can run anywhere JavaScript runs. In addition to everything the JavaScript language offers, TypeScript also offers a much richer type system, generics, decorators, interfaces, mixins, additional tools, ambient type declarations and lots more, which is convincing more and more larger projects to adopt it as their core programming language. We see it use internally with Angular 5, Zone.js, Rx observables and many commercial applications.

The aim of this course is to quickly bring you up to speed with programming in TypeScript. We explore the language syntax, its access to libraries, how to build applications and see why it is more and more being selected instead of JavaScript by senior web developers.

Target Audience

Developers wishing to use TypeScript for browser (e.g. Angular 5) and server (e.g. Node 9) programming

Prerequisites

Software developers with an object-oriented background and some browser programming experience.

Contents of One-Day Training Course

TypeScript Introduction

Relationship to ECMAScript standards
Transpiling to JavaScript
Language tour
What we should be familiar with and what may be new (any, never, tuple)
tsconfig.json

Object Types

Type system hierarchy
Type inferencing
Visibility and immutability
Duck typing

Classes

Defining a class
Constructors
Inheritance
Extending and implementing properties and accessors

Interfaces

Specifying an interface
What happens to interfaces after transpilation (they disappear!)

Mixins

Partial or full implementation of interface
Additional construct which can be very useful in certain circumstances

Modules

Modules as a unit of delivery and unit of code management
Importing and exporting

Namespace

Sub-dividing module types in namespaces
Use in conjunction with module naming

Generics and Constraints

Type-independent code
Separating algorithm from types
Constraining permissible type parameters
Relationship to transpiled code

Iterators & Generators

Symbol.iterator and for..of
Generator function

Specialist Types

Intersection type
Union type
Nullable
Alias

Reflection/Decorators/Metadata

Attaching metadata to a class
Using decorators
reflect-metadata

Using Ambient Declarations

Interacting with non-TypeScript libraries and use of @types with npm

Creating Ambient Declarations

Writing and publishing .d.ts files
Ambient syntax

lib.d.ts Standard Library

A modular collection of ambient declarations for various targets

New for TypeScript 2.5/2.6

Strict function types; Cache tagged template; Optional catch clause variables; --preserveSymlinks; Type assertion/cast

Project

Using TypeScript in a demo project to build a modern flexible framework