TypeScript

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

For modern larger-scale applications that target the JavaScript VM, either in browsers (e.g. Angular 8) or on the server and command-line tools (Node 12), or mobile apps (Ionic 4) or desktop apps (Electron 5), 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. This is convincing more and more project teams to adopt it as their core programming language. We see it use internally with Angular, Zone.js, RxJS, NgRx and many commercial applications (including very large ones).

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 create modern apps using TypeScript

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
Language tour
What we should be familiar with and what
may be new to us (any, never, tuple)
tsconfig.json and transpiling

Object Foundations

Type system hierarchy
Type inferencing
Visibility and immutability

Object Types

Duck typing
Type system
ObjectType definition
Properties and accessors
Call signatures
Tuples

Classes, Interfaces & Mixins

Defining a class
Constructors
Inheritance
Specifying an interface
What happens to interfaces after
transpilation (they disappear!)
Partial or full implementation of interface
Additional construct which can be very
useful in certain circumstances

Generics and Constraints

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

Namespace & Modules

Modules as a unit of delivery and unit of
code management
Importing and exporting
Sub-dividing module types in namespaces
Use in conjunction with module naming

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
The reflect-metadata package

Ambient Declarations

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

CallableFunction / NewableFunction
project refs, unknown, enhanced tuple
Properties on function declarations

Project

Using TypeScript in a project
to build a modern flexible framework