The TypeScript Handbook

About this Handbook

Over 20 years after its introduction to the programming community, JavaScript is now one of the most widespread cross-platform languages ever created. Starting as a small scripting language for adding trivial interactivity to webpages, JavaScript has grown to be a language of choice for both frontend and backend applications of every size. While the size, scope, and complexity of programs written in JavaScript has grown exponentially, the ability of the JavaScript language to express the relationships between different units of code has not. Combined with JavaScript's rather peculiar runtime semantics, this mismatch between language and program complexity has made JavaScript development a difficult task to manage at scale.

The most common kinds of errors that programmers write can be described as type errors: a certain kind of value was used where a different kind of value was expected. This could be due to simple typos, a failure to understand the API surface of a library, incorrect assumptions about runtime behavior, or other errors. The goal of TypeScript is to be a static typechecker for JavaScript programs - in other words, a tool that runs before your code runs (static) and ensures that the types of the program are correct (typechecked).

If you are coming to TypeScript without a JavaScript background, with the intention of TypeScript being your first language, we recommend you first start reading the documentation on either the Microsoft Learn JavaScript tutorial or read JavaScript at the Mozilla Web Docs. If you have experience in other languages, you should be able to pick up JavaScript syntax quite quickly by reading the handbook.

How is this Handbook Structured

The handbook is split into two sections:

- **The Handbook**

You should expect each chapter or page to provide you with a strong understanding of the given concepts. The TypeScript Handbook is not a complete language specification, but it is intended to be a comprehensive guide to all of the language's features and behaviors.

A reader who completes the walkthrough should be able to:

- Read and understand commonly-used TypeScript syntax and patterns
- Explain the effects of important compiler options
- Correctly predict type system behavior in most cases

In the interests of clarity and brevity, the main content of the Handbook will not explore every edge case or minutiae of the features being covered. You can find more details on particular concepts in the reference articles.

- **Reference Files**

  The reference section below the handbook in the navigation is built to provide a richer understanding of how a particular part of TypeScript works. You can read it top-to-bottom, but each section aims to provide a deeper explanation of a single concept - meaning there is no aim for continuity.

Non-Goals

The Handbook is also intended to be a concise document that can be comfortably read in a few hours. Certain topics won't be covered in order to keep things short.

Specifically, the Handbook does not fully introduce core JavaScript basics like functions, classes, and closures. Where appropriate, we'll include links to background reading that you can use to read up on those concepts.

The Handbook also isn't intended to be a replacement for a language specification. In some cases, edge cases or formal descriptions of behavior will be skipped in favor of high-level, easier-to-understand explanations. Instead, there are separate reference pages that more precisely and formally describe many aspects of TypeScript's behavior. The reference pages are not intended for readers unfamiliar with TypeScript, so they may use advanced terminology or reference topics you haven't read about yet.

Finally, the Handbook won't cover how TypeScript interacts with other tools, except where necessary. Topics like how to configure TypeScript with webpack, rollup, parcel, react, babel, closure, lerna, rush, bazel, preact, vue, angular, svelte

Get Started

Before getting started with The Basics, we recommend reading one of the following introductory pages. These introductions are intended to highlight key similarities and differences between TypeScript and your favored programming language, and clear up common misconceptions specific to those languages.

- TypeScript for the New Programmer
- TypeScript for JavaScript Programmers
- TypeScript for Java/C# Programmers
- TypeScript for Functional Programmers

Otherwise, jump to The Basics.

Next

The Basics
Step one in learning TypeScript: The basic types.

The TypeScript docs are an open source project. Help us improve these pages by sending a Pull Request ♥

Contributors to this page:

 MW  N

 5+

Last updated: Mar 01, 2024

This page loaded in 0.148 seconds.

Customize

Site Colours: System

Font: Cascadia

Popular Documentation Pages

More on Objects
How to provide a type shape to JavaScript objects

Narrowing
How TypeScript infers types based on runtime behavior

Variable Declarations
How to create and type JavaScript variables

TypeScript in 5 minutes
An overview of building a TypeScript web app

TSConfig Options
All the configuration options for a project

Classes
How to provide types to JavaScript ES6 classes

Community

Get Help
Blog
GitHub Repo
Community Chat
@TypeScript
Mastodon
Stack Overflow
Web Repo

Using TypeScript

Get Started
Download
Community
Playground
TSConfig Ref
Why TypeScript
Design
⊙ Code Samples

Made with ♥ in Redmond,
Boston, SF & Dublin

Docs **Community** **Tools**