

# Configuring the TypeScript Compiler

---



**Daniel Stern**  
CODE WHISPERER  
[@danieljackstern](#)



# Learning Objectives



Understand the purpose of configuring the TypeScript compiler and what the advantages are



Learn to configure the compiler in practice through several interactive demos



Integrate compilation with VSCode to run automatically or on demand



# Troubleshooting

If you get stuck, verify the following things.

`^3.6.4`

Correct TypeScript version



No local / global conflicts



Correct source code:  
<https://github.com/danielstern/compiling-typescript>



# Installing TypeScript

---



# Different TypeScript Versions



Built-in version of TypeScript used by Visual Studio provides code hints



Global version of TypeScript used for code compilation



# Installing TypeScript Globally

This code will install the latest version of TypeScript on your workstation.

```
ts.bash
```

```
npm install -g typescript@latest
```

```
tsc -v  
Version 3.6.4
```

# Installing TypeScript Locally

This code will install the latest version of TypeScript for your current project only.

```
ts2.bash
```

```
npm install --save-dev typescript@latest
```

```
tsc -v
```

```
Version 3.6.4
```

```
.. && tsc -v
```

```
Version 3.6.2
```

# Demo



## Install TypeScript globally

- Note how tsc is available from any terminal

## Install a local version of TypeScript for our project

- VS Code will automatically detect our new version of TypeScript





# Why Configure the TypeScript Compiler?

---

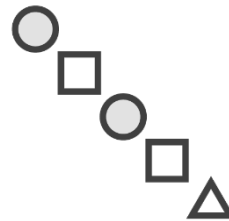


# Why Configure the TypeScript Compiler?

**All TypeScript projects are not created equal. The compiler settings need to suit the project.**



Optimize build times with cutting-edge features



Organize generated and source files as desired



Configuration can be done once by senior dev and used by all other devs



# Choosing Compiler Options

---



# Which Compiler Options Are Right for Your Project?



`declaration` creates `d.ts` files for your generated files



`target` changes the form of the compiled `.js` files



`strict` and `noUnusedLocals` promote good code structure



`composite` tells TypeScript this can be imported by other projects



`sourceMap` assists with debugging in Chrome



More options exist and are added with each version



# Adding Advanced Features to Our TypeScript Configuration File

---



# Demo



**Update `tsconfig.json` with our final values for relevant settings**

- Create declaration files
- Try out different strict tags
- Specify module format



# Extending TypeScript Configuration

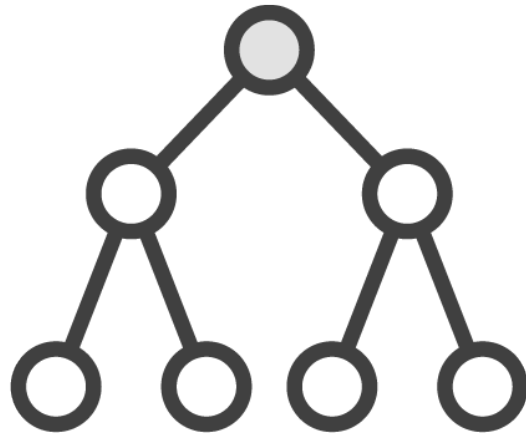
---



# Extending TypeScript Configuration



`tsconfig.json` can  
extend the  
configuration of  
other projects



Multiple projects can  
use single set of base  
settings



Reduces duplication,  
speeds up  
maintenance



# Demo



Create secondary TypeScript project in same folder

Separate TypeScript configuration into base settings and project-specific settings

Create new configuration files for our new project and existing main project



# Configuring a Default Build Task for Your Team

---



# VS Code and tasks.json

**tasks.json not only stores your project settings, but allows you to standardize them across the whole team.**



Specifies which build command to run by default



Can be checked in with version control (GIT, SVN, etc.)



Proper usage reduces onboarding time for new devs



# Creating tasks.json

---



# Demo



Use VS Code to automatically create tasks.json configuration file

Select desired task to execute automatically on build

Note the effects of our selection on tasks.json



# Running the Compiler in Watch Mode

---



# Advantage and Disadvantages of Watch Mode

## Advantages

Eliminates need for developer to manually build, saving only a few seconds but doing so thousands or even millions of times over

Facilitates rapid developer feedback loop

Encourages developers to focus on solving assigned problems, improving morale

## Disadvantages

Needs additional configuration (i.e., specifying files to watch)

Large builds can consume lots of memory, slowing development

Glitch in system can leave junior developers unable to build



# Demo: Running the Compiler in Watch Mode

---





# Demo



Update tasks.json  
so compiler runs in watch mode

Note how application build code is  
recreated with each relevant change



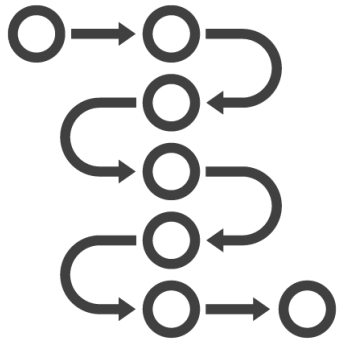
# Configuring Project References

---



# What Are Project References?

Project references allow you to logically separate your TypeScript application, allowing for faster compilation and maintenance.



## Organizational

Separate business logic  
into sub-projects



## Performant

TypeScript optimizes  
builds for referenced  
projects



## Simple

Process of referencing  
projects very similar to  
standard imports

# Demo



Create reference to junior project from main project

Note that TypeScript minimizes amount of code that is recompiled with each build



# Module Summary



**The TypeScript compiler has a wide array of features, each suited to a different type of project or scenario**

- Folders for organization
- Targets for compatibility
- Strict modes for quality control

**Good TypeScript configuration can save many hours when multiplied over a big team**

# Coming Up in the Next Module



**Install type declaration files for standard JavaScript libraries**

**Create a type declaration file for our own custom library**

**Explore VSCode's time-saving interactions with type declaration files**

