Transpiling TypeScript into JavaScript

VS Code integrates with tsc through our integrated [task runner](https://code.visualstudio.com/docs/editor/tasks). We can use this to transpile .ts files into .js files. Let's walk through transpiling a simple TypeScript Hello World program.

Step 1: Create a simple TS file

Open VS Code on an empty folder and create a HelloWorld.ts file, place the following code in that file...

class Startup {

public static main(): number {

console.log('Hello World');

return 0;

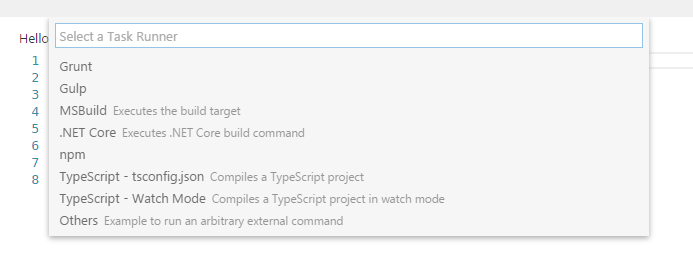
}

}

Startup.main();

Step 2: Create tasks.json

The next step is to set up the task configuration. To do this open the **Command Palette** with Ctrl+Shift+P and type in **Configure Task Runner**, press Enter to select it. This shows a selection box with templates you can choose from:



Select TypeScript - tsconfig.json. This will create a tasks.json file in the workspace .vscode folder.

The content of the tasks.json file looks like this:

{

// See https://go.microsoft.com/fwlink/?LinkId=733558

// for the documentation about the tasks.json format

"version": "0.1.0",

"command": "tsc",

"isShellCommand": true,

"args": ["-p", "."],

"showOutput": "silent",

"problemMatcher": "$tsc"

}

**Tip:** While the template is there to help with common configuration settings, IntelliSense is available for the tasks.json file as well to help you along. Use Ctrl+Space to see the available settings.

Under the covers we interpret tsc as an external task runner exposing exactly one task: the compiling of TypeScript files into JavaScript files. The command we run is: tsc -p .

**Tip:** If you don't have the TypeScript compiler installed, you can [get it here](https://www.typescriptlang.org/).

Step 3: Run the Build Task

As this is the only task in the file, you can execute it by pressing Ctrl+Shift+B (**Run Build Task**). At this point you will see an additional file show up in the file list HelloWorld.js.

The example TypeScript file did not have any compile problems, so by running the task all that happened was a corresponding HelloWorld.js and HelloWorld.js.map file was created.

If you have [Node.js](https://nodejs.org/) installed, you can run your simple Hello World example by opening up a terminal and running:

node HelloWorld.js

**Tip:** You can also run the program using VS Code's Run/Debug feature. Details about running and debugging node apps in VS Code can be found [here](https://code.visualstudio.com/docs/nodejs/nodejs-tutorial#_debugging-your-node-application)

Step 4: Reviewing Build Issues

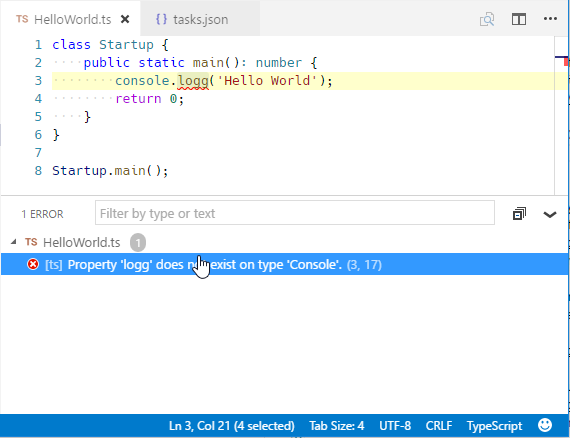
Unfortunately, most builds don't go that smoothly and the result is often some additional information. For instance, if there was a simple error in our TypeScript file, we may get the following output from tsc:

HelloWorld.ts(3,17): error TS2339: Property 'logg' does not exist on type 'Console'.

This would show up in the output window (which can be opened using Ctrl+Shift+U) and selecting Tasks in the output view dropdown. We parse this output for you and highlight detected problems in the Status Bar.

Problems in Status Bar

You can click on that icon to get a list of the problems and navigate to them.



You can also use the keyboard to open the list Ctrl+Shift+M.

**Tip:** Tasks offer rich support for many actions. Check the [Tasks](https://code.visualstudio.com/docs/editor/tasks) topic for more information on how to configure them.

Goto Symbol & Show All Symbols

Ctrl+Shift+O: lists all defined symbols of the current open TypeScript and lets you navigate in it.

Ctrl+T: lets you search all symbols defined in the current project or file scope. You need to have a TypeScript file open in the active editor.

Format Code

Shift+Alt+F: formats the whole document. Ctrl+K Ctrl+F: formats the currently selected source code.

JSDoc Support

VS Code offers **JSDoc** support for TypeScript. Besides syntax coloring, we help you enter **JSDoc** comments. Type /\*\* and it will auto insert the closing \*/. Pressing Enter inside a **JSDoc** block will indent the next line and auto insert a \*.

JavaScript Source Map Support

TypeScript debugging supports JavaScript source maps. Enable this by setting the sourceMaps attribute to true in the project's launch configuration file launch.json. In addition, you can specify a TypeScript file with the program attribute.

To generate source maps for your TypeScript files, compile with the --sourcemap option or set the sourceMap property in the tsconfig.json file to true.

In-lined source maps (a source map where the content is stored as a data URL instead of a separate file) are also supported, although in-lined source is not yet supported.

Setting a different outFiles for generated files

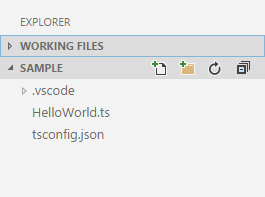
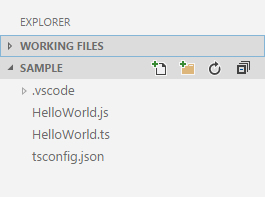
If generated (transpiled) JavaScript files do not live next to their source, you can help the VS Code debugger locate them by setting the outFiles attribute in the launch configuration. Whenever you set a breakpoint in the original source, VS Code tries to find the generated source by searching the files specified by glob patterns in outFiles.

Hiding Derived JavaScript Files

When you are working with TypeScript, you often don’t want to see generated JavaScript files in the explorer or in search results. VS Code offers filtering capabilities with a files.exclude [workspace setting](https://code.visualstudio.com/docs/getstarted/settings)(**File** > **Preferences** > **Settings**) and you can easily create an expression to hide those derived files:

"\*\*/\*.js": { "when": "$(basename).ts"}

This pattern will match on any JavaScript file (\*\*/\*.js) but only if a sibling TypeScript file with the same name is present. The file explorer will no longer show derived resources for JavaScript if they are compiled to the same location.



To exclude JavaScript files generated from both .ts and .tsx source files, use this expression:

"\*\*/\*.js": { "when": "$(basename).ts" },

"\*\*/\*\*.js": { "when": "$(basename).tsx" }

This is a bit of a trick. The search glob pattern is used as a key. The settings above use two different glob patterns to provide two unique keys but the search will still match the same files.

Mixed TypeScript and JavaScript projects

It is now possible to have mixed TypeScript and JavaScript projects. To enable JavaScript inside a TypeScript project, you can set the allowJs property to true in the tsconfig.json.

**Tip:** The tsc compiler does not detect the presence of a jsconfig.json file automatically. Use the –p argument to make tsc use your jsconfig.json file, e.g. tsc -p jsconfig.json.

Using Newer TypeScript Versions

VS Code ships with a recent stable version of the TypeScript language service and it may not match the version of TypeScript installed globally on your computer or locally in your workspace. The active version of the TypeScript language service is displayed in the Status Bar when viewing a TypeScript or JavaScript file:

TypeScript status bar version

When VS Code detects that the TypeScript compiler (tsc) version is different than the active TypeScript language service version, a message is displayed "Version mismatch! global tsc (2.1.5) != VS Code's language service (2.2.1). Inconsistent compiler errors might occur". This message is benign and is meant to alert the user to the possible differences between the compiler and active language service.

You can disable this check with the typescript.check.tscVersion user or workspace [setting](https://code.visualstudio.com/docs/getstarted/settings). This will be set to false in your user settings if you click the **Don't Check Again** in the message banner.

"typescript.check.tscVersion": false

Another option is to install the matching version of TypeScript in your workspace (npm install --save-dev typescript) or globally on your computer (npm install -g typescript). We recommend installing TypeScript locally to avoid possible interactions with other TypeScript projects you may have.

**Tip:** To get a specific TypeScript version, specify @version. For example for TypeScript 2.2.1, you would use npm install --save-dev typescript@2.2.1. To preview the next version of TypeScript, run npm install --save-dev typescript@next.

As VS Code updates the TypeScript language service in subsequent releases, you may see the mismatch message again and want to refresh your installed version of TypeScript.

To use a different TypeScript version by default, configure typescript.tsdk in your user settings to point to a directory containing the TypeScript tsserver.js file. You can find the TypeScript installation location using npm list -g typescript. The tsserver.js file is usually in the lib folder.

For example:

{

"typescript.tsdk": "/usr/local/lib/node\_modules/typescript/lib"

}

You can also configure a specific version of TypeScript in a particular workspace by adding a typescript.tsdk workspace setting pointing to the directory of the tsserver.js file:

{

"typescript.tsdk": "./node\_modules/typescript/lib"

}

If your workspace has a specific TypeScript version, you can switch between the workspace version of TypeScript and the version that VS Code uses by default by opening a TypeScript or JavaScript file in the workspace and clicking on the TypeScript version number in the Status Bar. A message box will appear asking you which version of TypeScript VS Code should use:



You can switch back to the version of TypeScript that comes with VS Code by clicking on the TypeScript version in the Status Bar again.