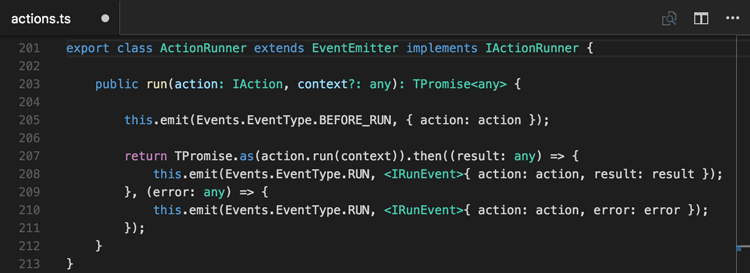
Editing TypeScript

TypeScript is a typed superset of JavaScript that compiles to plain JavaScript. It offers classes, modules, and interfaces to help you build robust components. A language specification can be found [here](https://github.com/Microsoft/TypeScript/tree/master/doc).



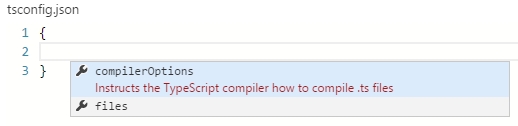
VS Code's TypeScript support can operate in two different modes:

* **File Scope**: in this mode TypeScript files opened in Visual Studio Code are treated as independent units. As long as a file a.ts doesn't reference a file b.ts explicitly (either using [/// reference directives](http://www.typescriptlang.org/docs/handbook/triple-slash-directives.html) or external modules) there is no common project context between the two files.
* **Explicit Project**: a TypeScript project is defined via a tsconfig.json file. The presence of such a file in a directory indicates that the directory is the root of a TypeScript project. The file itself lists the files belonging to the project as well as compiler options. Details about the tsconfig.json file can be found [here](https://www.typescriptlang.org/docs/handbook/tsconfig-json.html).

**Tip:** We recommend that you use explicit projects over file scope projects. Since explicit projects list the files belonging to a project language, features like Find All ReferencesShift+F12 consider the project scope and not the file scope only.

tsconfig.json

Typically the first step in any new TypeScript project is to add in a tsconfig.json file. This defines the TypeScript project settings such as the compiler options and the files that should be included. To do this, open up the folder where you want to store your source and add in a new file named tsconfig.json. Once in this file IntelliSense will help you along the way.



A simple tsconfig.json looks like this for ES5, **CommonJS** [modules](http://www.commonjs.org/specs/modules/1.0) and source maps:

{

"compilerOptions": {

"target": "es5",

"module": "commonjs",

"sourceMap": true

}

}

Now when you create a .ts file as part of the project we will offer up rich editing experiences and syntax validation.

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;

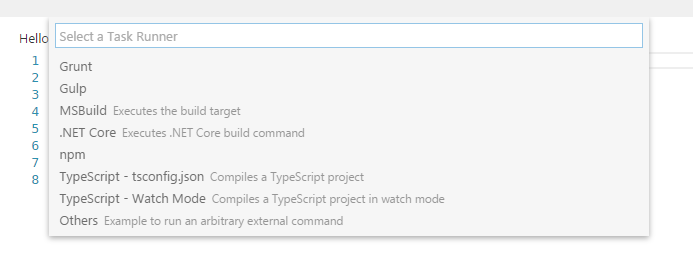
}

}

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 http://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](http://www.typescriptlang.org/).

Step 3: Run the Build Task

As this is the only task in the file, you can execute it by simply 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/runtimes/nodejs#_debugging-your-node-application)

Step 4: Reviewing Build Issues

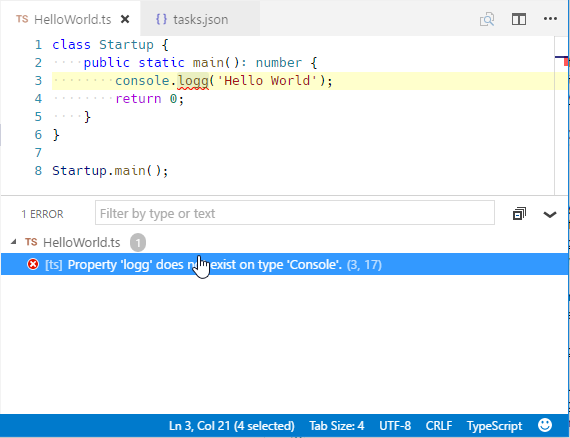
Unfortunately, most builds don't go that smoothly and the result is often some additional information. For instance, if there was a simple error in our TypeScript file, we may get the following output from tsc:

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. Simply 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

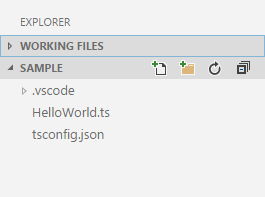
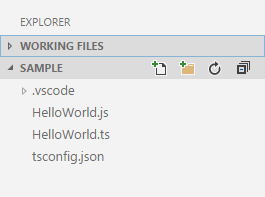
If generated (transpiled) JavaScript files do not live next to their source, you can help the VS Code debugger locate them by setting the outFiles attribute in the launch configuration. Whenever you set a breakpoint in the original source, VS Code tries to find the generated source by searching the files 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 [setting](https://code.visualstudio.com/docs/customization/userandworkspace) (**File** > **Preferences** > **Workspace 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.



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 TypeScript. It also performs version checking for any version of TypeScript you may have installed globally or locally in your workspace.

By default, VS Code will warn you if your global version is different than VS Code's TypeScript version since this may cause inconsistent compiler errors when building your project within VS Code and in an external terminal. You can disable this check with the **Dont Check Again** button on the warning which sets the typescript.check.tscVersion User [setting](https://code.visualstudio.com/docs/customization/userandworkspace) (**File** > **Preferences** > **User Settings**) to false:

{

"typescript.check.tscVersion": false

}

VS Code will also detect if a local workspace version of TypeScript is different from the bundled version. You can disable this check in either your User or Workspace settings with typescript.check.workspaceVersion.

You can also direct VS Code to use your workspace TypeScript version with the typescript.tsdk setting pointing to a directory containing the TypeScript tsserver.js file.

To install the latest TypeScript version, run:

npm install typescript@next

**Tip:** To get a specific TypeScript version, specify @version. For example for TypeScript 2.0, you would use npm install typescript@2.0.0.

You can find the installation location using npm list typescript, tsserver.js is usually under the libfolder.

For example:

{

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

}

The directory path can be absolute or relative to the workspace directory. By using a relative path, you can easily share this workspace setting with your team. Refer to this [blog post](https://blogs.msdn.com/b/typescript/archive/2015/07/27/introducing-typescript-nightlies.aspx) for more details on how to install the nightly builds of TypeScript.

After setting typescript.tsdk, restart VS Code and the TypeScript version will be shown in the bottom right of the Status Bar when you open a JavaScript or TypeScript file.

TypeScript Extensions

VS Code provides many features for TypeScript out of the box. In addition to what comes built-in, you can install an extension for greater functionality.

**[TSLint](https://marketplace.visualstudio.com/items?itemName=eg2.tslint" \t "_blank)**

[357.0K](https://marketplace.visualstudio.com/items?itemName=eg2.tslint" \t "_blank)

[eg2](https://marketplace.visualstudio.com/items?itemName=eg2.tslint" \t "_blank)

[TSLint for Visual Studio Code](https://marketplace.visualstudio.com/items?itemName=eg2.tslint" \t "_blank)

[[](https://marketplace.visualstudio.com/items?itemName=johnpapa.Angular2)](https://marketplace.visualstudio.com/items?itemName=johnpapa.Angular2" \t "_blank)

**[Angular 2 TypeScript...](https://marketplace.visualstudio.com/items?itemName=johnpapa.Angular2" \t "_blank)**

[251.4K](https://marketplace.visualstudio.com/items?itemName=johnpapa.Angular2" \t "_blank)

[johnpapa](https://marketplace.visualstudio.com/items?itemName=johnpapa.Angular2" \t "_blank)

[Angular 2 TypeScript snippets](https://marketplace.visualstudio.com/items?itemName=johnpapa.Angular2" \t "_blank)

[[https://formulahendry.gallerycdn.vsassets.io/extensions/formulahendry/code-runner/0.6.6/1484907438168/Microsoft.VisualStudio.Services.Icons.Default](https://marketplace.visualstudio.com/items?itemName=formulahendry.code-runner)](https://marketplace.visualstudio.com/items?itemName=formulahendry.code-runner" \t "_blank)

**[Code Runner](https://marketplace.visualstudio.com/items?itemName=formulahendry.code-runner" \t "_blank)**

[144.5K](https://marketplace.visualstudio.com/items?itemName=formulahendry.code-runner" \t "_blank)

[formulahendry](https://marketplace.visualstudio.com/items?itemName=formulahendry.code-runner" \t "_blank)

[Run code snippet/file for C, C++, Java, JS, PHP, ...](https://marketplace.visualstudio.com/items?itemName=formulahendry.code-runner" \t "_blank)

[[](https://marketplace.visualstudio.com/items?itemName=johnpapa.Angular1)](https://marketplace.visualstudio.com/items?itemName=johnpapa.Angular1" \t "_blank)

**[Angular 1 JavaScript...](https://marketplace.visualstudio.com/items?itemName=johnpapa.Angular1" \t "_blank)**

[96.0K](https://marketplace.visualstudio.com/items?itemName=johnpapa.Angular1" \t "_blank)

[johnpapa](https://marketplace.visualstudio.com/items?itemName=johnpapa.Angular1" \t "_blank)

[Angular 1 JavaScript and TypeScript snippets](https://marketplace.visualstudio.com/items?itemName=johnpapa.Angular1" \t "_blank)

Tip: The extensions shown above are dynamically queried. Click on an extension tile above to read the description and reviews to decide which extension is best for you. See more in the [Marketplace](https://marketplace.visualstudio.com/).