# Solution-style `tsconfig.json` migration

## What does this migration do?

This migration adds support to existing projects for TypeScript's new "solution-style" tsconfig feature.

Support is added by making two changes:

1. Renaming the workspace-level `tsconfig.json` to `tsconfig.base.json`. All project TypeScript configuration files will extend from this base which contains the common options used throughout the workspace.
2. Adding the solution `tsconfig.json` file at the root of the workspace. This `tsconfig.json` file will only contain references to project-level TypeScript configuration files and is only used by editors/IDEs.

As an example, the solution `tsconfig.json` for a new project is as follows:

```
// This is a "Solution Style" tsconfig.json file, and is used by editors and TypeScript's language server to improve development experience.
// It is not intended to be used to perform a compilation.
{
  "files": [],
  "references": [
    {
      "path": "./tsconfig.app.json"
    },
    {
      "path": "./tsconfig.spec.json"
    },
    {
      "path": "./e2e/tsconfig.json"
    }
  ]
}
```

## Why is this migration necessary?

Solution-style `tsconfig.json` files provide an improved editing experience and fix several long-standing defects when editing files in an IDE. IDEs that leverage the TypeScript language service (for example, Visual Studio Code), will only use TypeScript configuration files that are named `tsconfig.json`. In complex projects, there may be more than one compilation unit and each of these units may have different settings and options.

With the Angular CLI, a project will have application code that will target a browser. It will also have unit tests that should not be included within the built application and that also need additional type information present (`jasmine` in this case). Both parts of the project also share some but not all of the code within the project. As a result, two separate TypeScript configuration files (`tsconfig.app.json` and `tsconfig.spec.json`) are needed to ensure that each part of the application is configured properly and that the right types are used for each part. Also if web workers are used within a project, an additional tsconfig (`tsconfig.worker.json`) is needed. Web workers use similar but incompatible types to the main browser application. This requires the additional configuration file to ensure that the web worker files use the appropriate types and will build successfully.

While the Angular build system knows about all of these TypeScript configuration files, an IDE using TypeScript's language service does not. Because of this, an IDE will not be able to properly analyze the code from each part of the project and may generate false errors or make suggestions that are incorrect for certain files. By leveraging the new solution-style tsconfig, the IDE can now be aware of the configuration of each part of a project. This allows each file to be treated appropriately based on its tsconfig. IDE features such as error/warning reporting and auto-suggestion will operate more effectively as well.

The TypeScript 3.9 release blog post also contains some additional information regarding this new feature.