Setup for upgrading from AngularJS

Audience: Use this guide **only** in the context of Upgrading from AngularJS or Upgrading for Performance. Those Upgrade guides refer to this Setup guide for information about using the deprecated QuickStart GitHub repository, which was created prior to the current Angular CLI.

For all other scenarios, see the current instructions in Setting up the Local Environment and Workspace.

This guide describes how to develop locally on your own machine. Setting up a new project on your machine is quick and easy with the QuickStart seed on github.

Prerequisite: Make sure you have Node.js® and npm installed.

{@a clone}

Clone

Perform the *clone-to-launch* steps with these terminal commands.

git clone https://github.com/angular/quickstart.git quickstart cd quickstart npm install

{@a download}

Download

Download the QuickStart seed and unzip it into your project folder. Then perform the remaining steps with these terminal commands.

cd quickstart npm install

{@a non-essential}

Delete *non-essential* files (optional)

You can quickly delete the *non-essential* files that concern testing and QuickStart repository maintenance (***including all git-related artifacts*** such as the `.git` folder and `.gitignore` !).

Do this only in the beginning to avoid accidentally deleting your own tests and git setup!

Open a terminal window in the project folder and enter the following commands for your environment:

OS/X (bash)

xargs rm -rf < non-essential-files.osx.txt rm src/app/*.spec*.ts rm non-essential-files.osx.txt

Windows

for /f %i in (non-essential-files.txt) do del %i /F /S /Q rd .git /s /q rd e2e /s /q

Update dependency versions

Since the quickstart repository is deprecated, it is no longer updated and you need some additional steps to use the latest Angular.

1. Remove the obsolete `@angular/http` package (both from `package.json > dependencies` and `src/systemjs.config.js > SystemJS.config() > map`).

2. Install the latest versions of the Angular framework packages by running:

```
npm install --save @angular/common@latest @angular/compiler@latest
@angular/core@latest @angular/forms@latest @angular/platform-browser@latest
@angular/platform-browser-dynamic@latest @angular/router@latest
```

3. Install the latest versions of other packages used by Angular (RxJS, TypeScript, Zone.js) by running:

```
npm install --save rxjs@latest zone.js@latest
npm install --save-dev typescript@latest
```

4. Install the `systemjs-plugin-babel` package. This will later be used to load the Angular framework files, which are in ES2015 format, using SystemJS.

```
npm install --save systemjs-plugin-babel@latest
```

5. In order to be able to load the latest Angular framework packages (in ES2015 format) correctly, replace the relevant entries in `src/systemjs.config.js` :

6. In order to be able to load the latest RxJS package correctly, replace the relevant entries in `src/systemjs.config.js` :

7. In order to be able to load the `tslib` package (which is required for files transpiled by TypeScript), add the following entry to `src/systemjs.config.js` :

8. In order for SystemJS to be able to load the ES2015 Angular files correctly, add the following entries to `src/systemjs.config.js` :

9. Finally, in order to prevent TypeScript typecheck errors for dependencies, add the following entry to `src/tsconfig.json` :

```
{
  "compilerOptions": {
    "skipLibCheck": true,
    // ...
  }
}
```

With that, you can now run `npm start` and have the application built and served. Once built, the application will be automatically opened in a new browser tab and it will be automatically reloaded when you make changes to the source code.

{@a seed}

What's in the QuickStart seed?

The **QuickStart seed** provides a basic QuickStart playground application and other files necessary for local development. Consequently, there are many files in the project folder on your machine, most of which you can learn about later.

Reminder: The "QuickStart seed" example was created prior to the Angular CLI, so there are some differences between what is described here and an Angular CLI application.

{@a app-files}

Focus on the following three TypeScript (`.ts`) files in the **`/src`** folder.

src

```
<div class='file'>
  app
</div>

<div class='children'>

  <div class='file'>
    app.component.ts
  </div>

  <div class='file'>
    app.module.ts
  </div>

</div>

<div class='file'>
  main.ts
</div>
```

All guides and cookbooks have *at least these core files*. Each file has a distinct purpose and evolves independently as the application grows.

Files outside `src/` concern building, deploying, and testing your application. They include configuration files and external dependencies.

Files inside `src/` "belong" to your application. Add new Typescript, HTML and CSS files inside the `src/` directory, most of them inside `src/app` , unless told to do otherwise.

The following are all in `src/`

```
<th>
  File
</th>

<th>
  Purpose
</th>
```

```
<td>
  <code>app/app.component.ts</code>
</td>

<td>



  Defines the same `AppComponent` as the one in the QuickStart playground.
  It is the **root** component of what will become a tree of nested components
  as the application evolves.
</td>
```

```
<td>
  <code>app/app.module.ts</code>
</td>

<td>



  Defines `AppModule`, the  [root module](guide/bootstrapping "AppModule: the root
module") that tells Angular how to assemble the application.
  When initially created, it declares only the `AppComponent`.
  Over time, you add more components to declare.
</td>
```

```
<td>
  <code>main.ts</code>
</td>

<td>



  Compiles the application with the [JIT compiler](guide/glossary#jit) and
  [bootstraps](guide/bootstrapping)
  the application's main module (`AppModule`) to run in the browser.
  The JIT compiler is a reasonable choice during the development of most projects and
  it's the only viable choice for a sample running in a _live-coding_ environment such
as Stackblitz.
  Alternative [compilation](guide/aot-compiler), [build](guide/build), and
[deployment](guide/deployment) options are available.

</td>
```

Appendix: Test using `fakeAsync()/waitForAsync()`

If you use the `fakeAsync()/waitForAsync()` helper functions to run unit tests (for details, read the [Testing guide](guide/testing)), you need to import `zone.js/testing` in your test setup file.

If you create project with `Angular/CLI`, it is already imported in `src/test.ts`.
And in the earlier versions of `Angular`, the following files were imported or added in your html file:

```
import 'zone.js/plugins/long-stack-trace-zone';
import 'zone.js/plugins/proxy';
import 'zone.js/plugins/sync-test';
import 'zone.js/plugins/jasmine-patch';
import 'zone.js/plugins/async-test';
import 'zone.js/plugins/fake-async-test';
```

You can still load those files separately, but the order is important, you must import `proxy` before `sync-test` , `async-test` , `fake-async-test` and `jasmine-patch` . And you also need to import `sync-test` before `jasmine-patch` , so it is recommended to just import `zone-testing` instead of loading those separated files.