Angular documentation project (https://angular.io)

Everything in this folder is part of the documentation project. This includes:

- the web site for displaying the documentation
- the dgeni configuration for converting source files to rendered files that can be viewed in the web site.
- the tooling for setting up examples for development; and generating live-example and zip files from the examples.

Developer tasks

We use Yarn to manage the dependencies and to run build tasks. You should run all these tasks from the `angular/aio` folder. Here are the most important tasks you might need to use:

- `yarn` - install all the dependencies.
- `yarn setup` - install all the dependencies, boilerplate, stackblitz, zips and run dgeni on the docs.
- `yarn setup-local` - same as `setup`, but build the Angular packages from the source code and use these locally built versions (instead of the ones fetched from npm) for aio and docs examples boilerplate.
- `yarn build` - create a production build of the application (after installing dependencies, boilerplate, etc).
- `yarn build-local` - same as `build`, but use `setup-local` instead of `setup`.
- `yarn start` - run a development web server that watches the files; then builds the doc-viewer and reloads the page, as necessary.
- `yarn serve-and-sync` - run both the `docs-watch` and `start` in the same console.
- `yarn lint` - check that the doc-viewer code follows our style rules.
- `yarn test` - watch all the source files, for the doc-viewer, and run all the unit tests when any change.
- `yarn test --watch=false` - run all the unit tests once.
- `yarn e2e` - run all the e2e tests for the doc-viewer.
- `yarn docs` - generate all the docs from the source files.
- `yarn docs-watch` - watch the Angular source and the docs files and run a short-circuited doc-gen for the docs that changed.
- `yarn docs-lint` - check that the doc gen code follows our style rules.
- `yarn docs-test` - run the unit tests for the doc generation code.
- `yarn boilerplate:add` - generate all the boilerplate code for the examples, so that they can be run locally.
- `yarn boilerplate:remove` - remove all the boilerplate code that was added via `yarn boilerplate:add`.
- `yarn create-example` - create a new example directory containing initial source files.

- `yarn generate-stackblitz` - generate the stackblitz files that are used by the `live-example` tags in the docs.

- `yarn generate-zips` - generate the zip files from the examples. Zip available via the `live-example` tags in the docs.

- `yarn example-e2e` - run all e2e tests for examples. Available options:

  - `--setup`: generate boilerplate, force webdriver update & other setup, then run tests.
  - `--local`: run e2e tests with the local version of Angular contained in the "dist" folder. *Requires `--setup` in order to take effect.*
  - `--filter=foo`: limit e2e tests to those containing the word "foo".

> ***Note for Windows users***
>
> *Setting up the examples involves creating some symbolic links (see here for details). On Windows, this requires to either have Developer Mode enabled (supported on Windows 10 or newer) or run the setup commands as administrator.*
>
> *The affected commands are:*
>
> - *`yarn setup` / `yarn setup-*`*
> - *`yarn build` / `yarn build-*`*
> - *`yarn boilerplate:add`*
> - *`yarn example-e2e --setup`*

Using ServiceWorker locally

Running `yarn start` (even when explicitly targeting production mode) does not set up the ServiceWorker. If you want to test the ServiceWorker locally, you can use `yarn build` and then serve the files in `dist/` with `yarn http-server dist -p 4200`.

Guide to authoring

There are two types of content in the documentation:

- **API docs**: descriptions of the modules, classes, interfaces, decorators, etc that make up the Angular platform. API docs are generated directly from the source code. The source code is contained in TypeScript files, located in the `angular/packages` folder. Each API item may have a preceding comment, which contains JSDoc style tags and content. The content is written in markdown.

- **Other content**: guides, tutorials, and other marketing material. All other content is written using markdown in text files, located in the `angular/aio/content` folder. More specifically, there are sub-folders that contain particular types of content: guides, tutorial and marketing.

- **Code examples**: code examples need to be testable to ensure their accuracy. Also, our examples have a specific look and feel and allow the user to copy the source code. For larger examples they are rendered in a tabbed interface (e.g. template, HTML, and TypeScript on separate tabs). Additionally, some are live examples, which provide links where the code can be edited, executed, and/or downloaded. For details on working with code examples, please read the Code snippets, Source code markup, and Live examples pages of the Authors Style Guide.

We use the dgeni tool to convert these files into docs that can be viewed in the doc-viewer.

The Authors Style Guide prescribes guidelines for writing guide pages, explains how to use the documentation classes and components, and how to markup sample source code to produce code snippets.

Generating the complete docs

The main task for generating the docs is `yarn docs`. This will process all the source files (API and other), extracting the documentation and generating JSON files that can be consumed by the doc-viewer.

Partial doc generation for editors

Full doc generation can take up to one minute. That's too slow for efficient document creation and editing.

You can make small changes in a smart editor that displays formatted markdown:

> *In VS Code, Cmd-K, V opens markdown preview in side pane; Cmd-B toggles left sidebar*

You also want to see those changes displayed properly in the doc viewer with a quick, edit/view cycle time.

For this purpose, use the `yarn docs-watch` task, which watches for changes to source files and only re-processes the files necessary to generate the docs that are related to the file that has changed. Since this task takes shortcuts, it is much faster (often less than 1 second) but it won't produce full fidelity content. For example, links to other docs and code examples may not render correctly. This is most particularly noticed in links to other docs and in the embedded examples, which may not always render correctly.

The general setup is as follows:

- Open a terminal, ensure the dependencies are installed; run an initial doc generation; then start the doc-viewer:

```
yarn setup
yarn start
```

- Open a second terminal and start watching the docs

```
yarn docs-watch
```

> *Alternatively, try the consolidated `serve-and-sync` command that builds, watches and serves in the same terminal window*

```
yarn serve-and-sync
```

- Open a browser at https://localhost:4200/ and navigate to the document on which you want to work. You can automatically open the browser by using `yarn start -o` in the first terminal.

- Make changes to the page's associated doc or example files. Every time a file is saved, the doc will be regenerated, the app will rebuild and the page will reload.

- If you get a build error complaining about examples or any other odd behavior, be sure to consult the Authors Style Guide.