Workspace npm dependencies

The Angular Framework, Angular CLI, and components used by Angular applications are packaged as npm packages and distributed using the npm registry.

You can download and install these npm packages by using the npm CLI client, which is installed with and runs as a Node.js® application. By default, the Angular CLI uses the npm client.

Alternatively, you can use the yarn client for downloading and installing npm packages.

See Local Environment Setup for information about the required versions and installation of `Node.js` and `npm`.

If you already have projects running on your machine that use other versions of Node.js and npm, consider using nvm to manage the multiple versions of Node.js and npm.

`package.json`

Both `npm` and `yarn` install the packages that are identified in a `package.json` file.

The CLI command `ng new` creates a `package.json` file when it creates the new workspace. This `package.json` is used by all projects in the workspace, including the initial application project that is created by the CLI when it creates the workspace.

Initially, this `package.json` includes *a starter set of packages*, some of which are required by Angular and others that support common application scenarios. You add packages to `package.json` as your application evolves. You may even remove some.

The `package.json` is organized into two groups of packages:

- Dependencies are essential to *running* applications.
- DevDependencies are only necessary to *develop* applications.

Library developers: By default, the CLI command `ng generate library` creates a `package.json` for the new library. That `package.json` is used when publishing the library to npm. For more information, see the CLI wiki page Library Support.

{@a dependencies} ## Dependencies

The packages listed in the `dependencies` section of `package.json` are essential to *running* applications.

The `dependencies` section of `package.json` contains:

- **Angular packages**: Angular core and optional modules; their package names begin `@angular/`.
- **Support packages**: 3rd party libraries that must be present for Angular applications to run.
- **Polyfill packages**: Polyfills plug gaps in a browser's JavaScript implementation.

To add a new dependency, use the `ng add` command.

{@a angular-packages} ### Angular packages

The following Angular packages are included as dependencies in the default `package.json` file for a new Angular workspace. For a complete list of Angular packages, see the API reference.

Package name	Description
@angular/animations	Angular's animations library makes it easy to define and apply animation effects such as page and list transitions. For more information, see the Animations guide.
@angular/common	The commonly-needed services, pipes, and directives provided by the Angular team. The `HttpClientModule` is also here, in the `@angular/common/http` subfolder. For more information, see the HttpClient guide.
@angular/compiler	Angular's template compiler. It understands templates and can convert them to code that makes the application run and render. Typically you don't interact with the compiler directly; rather, you use it indirectly using `platform-browser-dynamic` when JIT compiling in the browser. For more information, see the Ahead-of-time Compilation guide.
@angular/core	Critical runtime parts of the framework that are needed by every application. Includes all metadata decorators, `Component`, `Directive`, dependency injection, and the component lifecycle hooks.
@angular/forms	Support for both template-driven and reactive forms. For information about choosing the best forms approach for your app, see Introduction to forms.

Package name	Description
@angular/platform-browser	Everything DOM and browser related, especially the pieces that help render into the DOM. This package also includes the `bootstrapModuleFactory()` method for bootstrapping applications for production builds that pre-compile with AOT.
@angular/platform-browser-dynamic	Includes providers and methods to compile and run the application on the client using the JIT compiler.
@angular/router	The router module navigates among your application pages when the browser URL changes. For more information, see Routing and Navigation.

{@a support-packages} ### Support packages

The following support packages are included as dependencies in the default `package.json` file for a new Angular workspace.

Package name	Description
rxjs	Many Angular APIs return *observables*. RxJS is an implementation of the proposed Observables specification currently before the TC39 committee, which determines standards for the JavaScript language.
zone.js	Angular relies on zone.js to run Angular's change detection processes when native JavaScript operations raise events. Zone.js is an implementation of a specification currently before the TC39 committee that determines standards for the JavaScript language.

{@a polyfills} ### Polyfill packages

Many browsers lack native support for some features in the latest HTML standards, features that Angular requires. *Polyfills* can emulate the missing features. The Browser Support guide explains which browsers need polyfills and how you can add them.

{@a dev-dependencies}

DevDependencies

The packages listed in the `devDependencies` section of `package.json` help you develop the application on your local machine. You don't deploy them with the production application.

To add a new `devDependency`, use either one of the following commands:

npm install –save-dev <package-name>

yarn add –dev <package-name>

The following `devDependencies` are provided in the default `package.json` file for a new Angular workspace.

Package name	Description
@angular-devkit/build-angular	The Angular build tools.
@angular/cli	The Angular CLI tools.
@angular/compiler-cli	The Angular compiler, which is invoked by the Angular CLI's `ng build` and `ng serve` commands.
@types/...	TypeScript definition files for 3rd party libraries such as Jasmine and Node.js.
jasmine/...	Packages to support the Jasmine test library.
karma/...	Packages to support the karma test runner.
typescript	The TypeScript language server, including the *tsc* TypeScript compiler.

Related information

For information about how the Angular CLI handles packages see the following guides:

- Building and serving describes how packages come together to create a development build.
- Deployment describes how packages come together to create a production build.