Overview of Angular libraries

Many applications need to solve the same general problems, such as presenting a unified user interface, presenting data, and allowing data entry. Developers can create general solutions for particular domains that can be adapted for re-use in different applications. Such a solution can be built as Angular *libraries* and these libraries can be published and shared as *npm packages*.

An Angular library is an Angular project that differs from an application in that it cannot run on its own. A library must be imported and used in an application.

Libraries extend Angular's base functionality. For example, to add reactive forms to an app, add the library package using `ng add @angular/forms`, then import the `ReactiveFormsModule` from the `@angular/forms` library in your application code. Similarly, adding the service worker library to an Angular application is one of the steps for turning an application into a Progressive Web App (PWA). Angular Material is an example of a large, general-purpose library that provides sophisticated, reusable, and adaptable UI components.

Any application developer can use these and other libraries that have been published as npm packages by the Angular team or by third parties. See Using Published Libraries.

Creating libraries

If you have developed functionality that is suitable for reuse, you can create your own libraries. These libraries can be used locally in your workspace, or you can publish them as npm packages to share with other projects or other Angular developers. These packages can be published to the npm registry, a private npm Enterprise registry, or a private package management system that supports npm packages. See Creating Libraries.

Whether you decide to package functionality as a library is an architectural decision, similar to deciding whether a piece of functionality is a component or a service, or deciding on the scope of a component.

Packaging functionality as a library forces the artifacts in the library to be decoupled from the application's business logic. This can help to avoid various bad practices or architecture mistakes that can make it difficult to decouple and reuse code in the future.

Putting code into a separate library is more complex than simply putting everything in one application. It requires more of an investment in time and thought for managing, maintaining, and updating the library. This complexity can pay off, however, when the library is being used in multiple applications.

Note that libraries are intended to be used by Angular applications. To add Angular functionality to non-Angular web applications, use Angular custom elements.