Next steps: tools and techniques

After you understand the basic Angular building blocks, you can learn more about the features and tools that can help you develop and deliver Angular applications.

- Work through the Tour of Heroes tutorial to get a feel for how to fit the basic building blocks together to create a well-designed application.
- Check out the Glossary to understand Angular-specific terms and usage.
- Use the documentation to learn about key features in more depth, according to your stage of development and areas of interest.

Application architecture

- The **Main Concepts** section located in the table of contents contains several topics that explain how to connect the application data in your components to your page-display templates, to create a complete interactive application.
- The NgModules guide provides in-depth information on the modular structure of an Angular application.
- The Routing and navigation guide provides in-depth information on how to construct applications that allow a user to navigate to different views within your single-page application.
- The Dependency injection guide provides in-depth information on how to construct an application such that each component class can acquire the services and objects it needs to perform its function.

Responsive programming

The template syntax and related topics contain details about how to display your component data when and where you want it within a view, and how to collect input from users that you can respond to.

Additional pages and sections describe some basic programming techniques for Angular applications.

- Lifecycle hooks: Tap into key moments in the lifetime of a component, from its creation to its destruction, by implementing the lifecycle hook interfaces.
- Observables and event processing: How to use observables with components and services to publish and subscribe to messages of any type, such as user-interaction events and asynchronous operation results.
- Angular elements: How to package components as *custom elements* using Web Components, a web standard for defining new HTML elements in a framework-agnostic way.
- Forms: Support complex data entry scenarios with HTML-based input validation.
- Animations: Use Angular's animation library to animate component behavior without deep knowledge of animation techniques or CSS.

Client-server interaction

Angular provides a framework for single-page applications, where most of the logic and data resides on the client. Most applications still need to access a server using the `HttpClient` to access and save data. For some platforms and applications, you might also want to use the PWA (Progressive Web App) model to improve the user experience.

- HTTP: Communicate with a server to get data, save data, and invoke server-side actions with an HTTP client.
- Server-side rendering: Angular Universal generates static application pages on the server through server-side rendering (SSR). This allows you to run your Angular application on the server in order to improve performance and show the first page quickly on mobile and low-powered devices, and also facilitate web crawlers.
- Service workers and PWA: Use a service worker to reduce dependency on the network and significantly improve the user experience.
- Web workers: Learn how to run CPU-intensive computations in a background thread.

Support for the development cycle

- CLI Command Reference: The Angular CLI is a command-line tool that you use to create projects, generate application and library code, and perform a variety of ongoing development tasks such as testing, bundling, and deployment.
- Compilation: Angular provides just-in-time (JIT) compilation for the development environment, and ahead-of-time (AOT) compilation for the production environment.
- Testing platform: Run unit tests on your application parts as they interact with the Angular framework.
- Deployment: Learn techniques for deploying your Angular application to a remote server.
- Security guidelines: Learn about Angular's built-in protections against common web-application vulnerabilities and attacks such as cross-site scripting attacks.
- Internationalization: Make your application available in multiple languages with Angular's internationalization (i18n) tools.
- Accessibility: Make your application accessible to all users.

File structure, configuration, and dependencies

- Workspace and file structure: Understand the structure of Angular workspace and project folders.
- Building and serving: Learn to define different build and proxy server configurations for your project, such as development, staging, and production.
- npm packages: The Angular Framework, Angular CLI, and components used by Angular applications are packaged as npm packages and distributed using the npm registry. The Angular CLI creates a default `package.json` file, which specifies a starter set of packages that work well together and jointly support many common application scenarios.
- TypeScript configuration: TypeScript is the primary language for Angular application development.
- Browser support: Make your applications compatible across a wide range of browsers.

Extending Angular

- Angular libraries: Learn about using and creating re-usable libraries.
- Schematics: Learn about customizing and extending the CLI's generation capabilities.
- CLI builders: Learn about customizing and extending the CLI's ability to apply tools to perform complex tasks, such as building and testing applications.

@reviewed 2021-09-15