

NgRx 8

Reactive + State, State Store, Side Effects, DevTools, Entity, Schematics, Architecture, Project

We use the term “state” to describe nuggets of data whose lifetime outlives that of a single call to an event handler (e.g. auth token, contents of shopping cart, custom color selection for sidenav). An Angular app is composed of a hierarchy of components. Sometimes state that is only used by a single component can be stored within that component; state shared between related components (near each other in the hierarchy) can be passed among them directly. Using services with dependency injection is also an option; but for more substantial applications with many components, managing state needs more attention. Enter NgRx ...

NgRx is a well organized suite of packages to manage application state in a RxJS observable cache. It is very popular, because it comprehensively solves the state management issue that every large Angular 8 app ultimately will face. The three key participants in NgRx are actions, state and reducers (which literally reduce an existing state and an action to a new state). Though it can be used on the server, in the real world NgRx is mostly used on the client (on the server, data usually ends up in a database). In this detailed course we explore the world of NgRx and see how it can be of real benefit to larger Angular apps.

Contents of One-Day Training Course	
Target Audience Developers wishing to manage client-side state in their Angular 8 apps using NgRx 8	Reactive Meets State What problem are we trying to solve The observable pattern and its uses Parts of RxJS that are of interest to us Lets look at state within an Angular app When to use NgRx (and when not to)
Prerequisites Experienced Angular developers building larger applications who need to more seriously consider their client-side state management architecture.	Overview of NgRx Exploring the NgRx Platform Important packages and their interactions Getting it installed and running Adding to development environment
	Concepts State (and the idea of immutability) Actions (state changes) Reducers
	@ngrx/store Introduction OnPush change detection strategy Boilerplate code for NgRx Creating reducers
	Advanced @ngrx/store State composition Selectors ngrx-store-localstorage
	Meta-Reducers Meta actions & meta reducers Wrapping a reducer with a meta reducer Parameter to <code>StoreModule.forRoot()</code>
	@ngrx/effects What is an effects model? Isolating change, making pure components Feeding actions into state cache Event sourcing architecture <code>@Effect()</code> decorator
	@ngrx/router-store Combining Angular Router & <code>@ngrx/store</code> ROUTER_NAVIGATION StoreRouterConnectingModule
	@ngrx/entity What is an entity collection? CRUD operations on entity collections Uses for type-safe adapters/entity selectors
	@ngrx/store-devtools Redux DevTools Instrumentation tooling for the store Instrumentation options
	@ngrx/schematics Idea of schematics A scaffolding library that integrates NgRx with Angular CLI Available blueprints
	Case Study: NgRx & Shopping Cart NgRx is very often used to build the shopping cart feature in eCommerce apps Let's see how to do it properly
	NgRx Internals Tour of NgRx source code https://github.com/ngrx/platform NgRx source is managed as a monorepo Discover how it all fits together
	Architecture Considerations Exploring the architectural issues that need to be considered in order to successfully leverage NgRx rich capabilities in an app
	Project Combining what we have learnt in the course to use in a larger NgRx 8 project