

Final Words



Duncan Hunter

Lead Frontend Developer

@dunchunter | duncanhunter.com.au



Enterprise Support

- Managing global and local state.
- Isolation of side effects to promote a cleaner component architecture.
- Entity collection management.
- Integration with the Angular Router.
- Developer tooling that enhances developer experience when building many different types of applications.

NgRx packages are divided into a few main categories

- **Store** - RxJS powered global state management for Angular apps, inspired by Redux.
- **Effects** - Side effect model for @ngrx/store.
- **Router Store** - Bindings to connect the Angular Router to @ngrx/store.
- **Entity** - Entity State adapter for managing record collections.
- **ComponentStore** - Standalone library for managing local/component state.

- **Data** - Extension for simplified entity data management.

- **Component** - Extension for building reactive Angular templates.

- **Store Devtools** - Instrumentation for @ngrx/store that enables visual tracking of state and time-travel debugging.
- **Schematics** - Scaffolding library for Angular applications using NgRx libraries.

NgRx - What is NgRx?

ngrx.io/docs

GETTING STARTED

DOCS

ENTERPRISE SUPPORT^{NEW}

BLOG

RESOURCES

EVENTS

SPONSOR

Search

Introduction

State

@ngrx/store

@ngrx/effects

@ngrx/router-store

@ngrx/entity

@ngrx/component-store

Data

@ngrx/data

View

@ngrx/component

Developer Tools

@ngrx/store-devtools

@ngrx/schematics

@ngrx/eslint-plugin

Recipes

Nightly Builds

Migrations

API

Contributing

Enterprise Support

What is NgRx?

NgRx is a framework for building reactive applications in Angular. NgRx provides libraries for:

- Managing global and local state.
- Isolation of side effects to promote a cleaner component architecture.
- Entity collection management.
- Integration with the Angular Router.
- Developer tooling that enhances developer experience when building many different types of applications.

Packages

NgRx packages are divided into a few main categories

State

- Store - RxJS powered global state management for Angular apps, inspired by Redux.
- Effects - Side effect model for @ngrx/store.
- Router Store - Bindings to connect the Angular Router to @ngrx/store.
- Entity - Entity State adapter for managing record collections.
- ComponentStore - Standalone library for managing local/component state.

Data

- Data - Extension for simplified entity data management.

View

- Component - Extension for building reactive Angular templates.

Developer Tools

- Store Devtools - Instrumentation for @ngrx/store that enables visual tracking of state and time-travel debugging.
- Schematics - Scaffolding library for Angular applications using NgRx libraries.

NgRx - What is NgRx?

ngrx.io/docs

GETTING STARTEDDOCSENTERPRISE SUPPORT^{NEW}BLOGRESOURCESEVENTSSPONSOR

Search

TwitterGitHub

Introduction

State

@ngrx/store

@ngrx/effects

@ngrx/router-store

@ngrx/entity

@ngrx/component-store

Data

@ngrx/data

View

@ngrx/component

Developer Tools

@ngrx/store-devtools

@ngrx/schematics

@ngrx/eslint-plugin

Recipes

Nightly Builds

Migrations

API

Contributing

Enterprise Support

What is NgRx?

NgRx is a framework for building reactive applications in Angular. NgRx provides libraries for:

- Managing global and local state.
- Isolation of side effects to promote a cleaner component architecture.
- Entity collection management.
- Integration with the Angular Router.
- Developer tooling that enhances developer experience when building many different types of applications.

Packages

NgRx packages are divided into a few main categories

State

- Store - RxJS powered global state management for Angular apps, inspired by Redux.
- Effects - Side effect model for @ngrx/store.
- Router Store - Bindings to connect the Angular Router to @ngrx/store.
- Entity - Entity State adapter for managing record collections.
- ComponentStore - Standalone library for managing local/component state.

Data

- Data - Extension for simplified entity data management.

View

- Component - Extension for building reactive Angular templates.

Developer Tools

- Store Devtools - Instrumentation for @ngrx/store that enables visual tracking of state and time-travel debugging.
- Schematics - Scaffolding library for Angular applications using NgRx libraries.

@ngrx/schematics

Schematics: scaffolding library for generating code using the CLI

- ng new
- ng generate

@ngrx/schematics: set of schematics for generating NgRx

- ng generate store
- ng generate action
- ng generate reducer
- ng generate effect
- ng generate feature
- ng generate container
- ng generate entity



ngrx/data

Abstracts away the NgRx entity code

Configuration and convention, not code

- No actions or action creators
- No reducers
- No selectors
- No effects
- No code generation

Extension points for customization



ngrx/ component

Set of helpers to enable more fully reactive applications

Currently provides:

- ngrxPush pipe
- ngrxLet directive



Angular - Getting started with : x

+

angular.io/guide/standalone-components

Search

Twitter GitHub YouTube

Getting started with standalone components

Standalone components provide a simplified way to build Angular applications. Standalone components, directives, and pipes aim to streamline the authoring experience by reducing the need for `NgModule`s. Existing applications can optionally and incrementally adopt the new standalone style without any breaking changes.

Creating standalone components

Getting Started with Standalone Components in An...
#ngUpdate

Share

Standalone Components

Watch on YouTube

The standalone flag and component imports

Components, directives, and pipes can now be marked as `standalone: true`. Angular classes marked as standalone do not need to be declared in an `NgModule` (the Angular compiler will report an error if you try).

Getting started with standalone components

Creating standalone components

The standalone flag and component imports

Using existing NgModules in a standalone component

Using standalone components in NgModule-based applications

Bootstrapping an application using a standalone component

Configuring dependency injection

Routing and lazy-loading

Lazy loading a standalone component

Lazy loading many

Angular - Angular Signals

+

angular.io/guide/signals

signals

Twitter GitHub YouTube

Angular Signals

Angular Signals is a system that granularly tracks how and where your state is used throughout an application, allowing the framework to optimize rendering updates.

Angular signals are available for [developer preview](#). They're ready for you to try, but may change before they are stable.

What are signals?

A **signal** is a wrapper around a value that can notify interested consumers when that value changes. Signals can contain any value, from simple primitives to complex data structures.

A signal's value is always read through a getter function, which allows Angular to track where the signal is used.

Signals may be either *writable* or *read-only*.

Writable signals

Writable signals provide an API for updating their values directly. You create writable signals by calling the `signal` function with the signal's initial value:

```
const count = signal(0);

// Signals are getter functions - calling them reads their value.
console.log('The count is: ' + count());
```

Angular Signals

What are signals?

Writable signals

Computed signals

Reading signals in OnPush component

Effects

Uses for effects

Injection context

Destroying effects

Advanced topics

Signal equality functions

Reading without tracking dependencies

Effect cleanup functions



Duncan Hunter

Lead Frontend Developer

@dunchunter | duncanhunter.com.au

