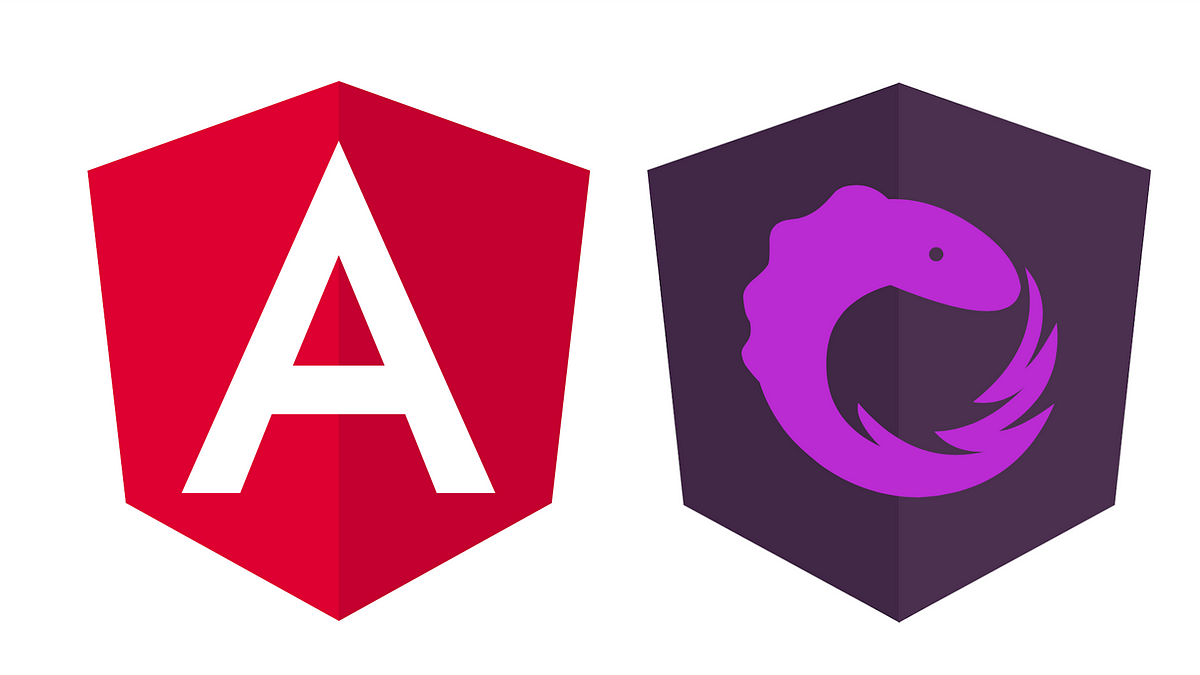
What is NgRx and why is it used in Angular?

******

**Introduction**

If you're developing an Angular application, you might have come across the term NgRx. NgRx is a state management library for Angular that helps you manage complex application states in a consistent and predictable way. In this blog, we will discuss what NgRx is, why it is used in Angular, and how it can benefit your application.

**What is NgRx?**

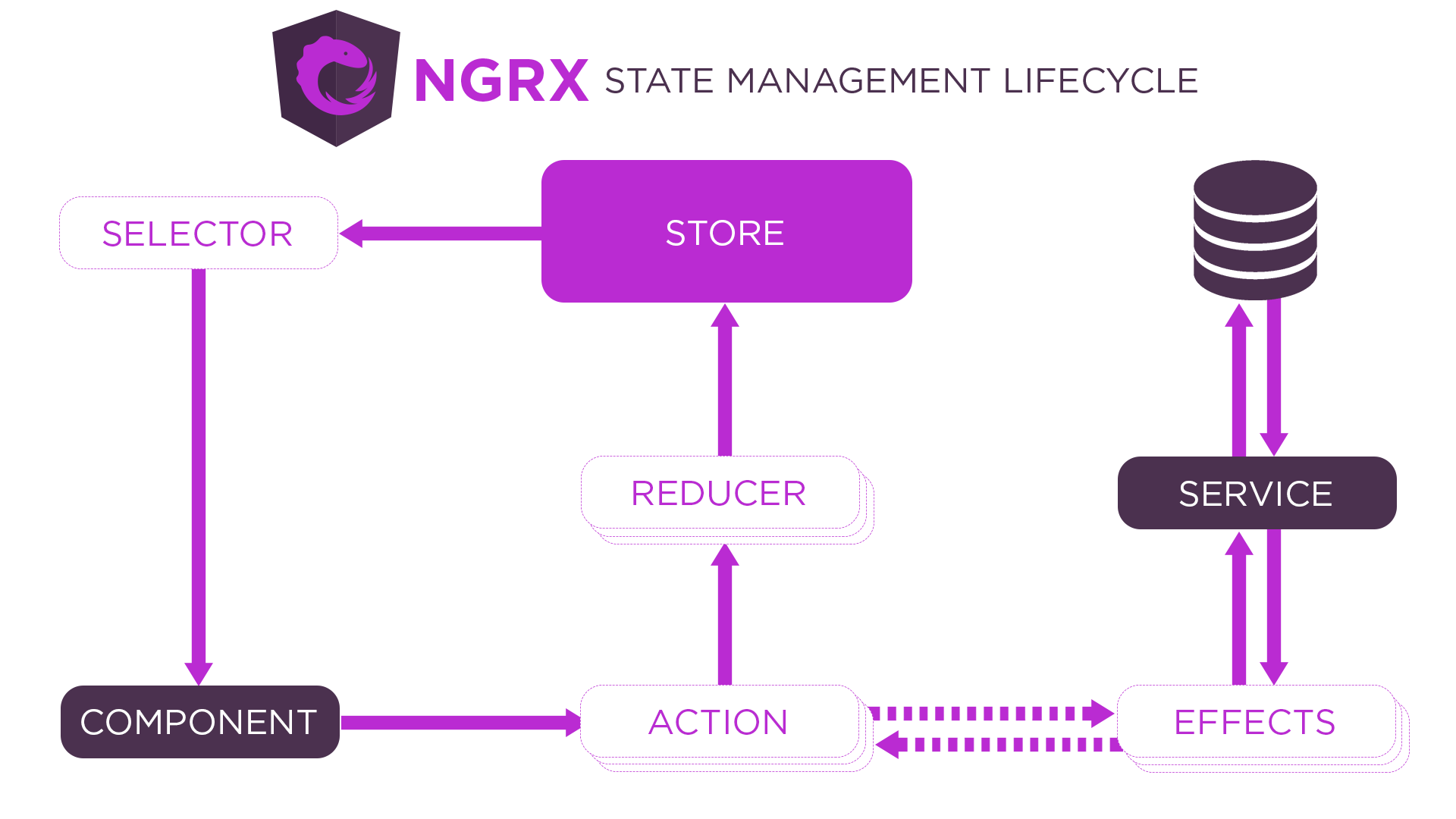
NgRx is a group of Angular libraries for reactive extensions and state management. It makes Angular development easier by simplifying the application’s state in objects and enforcing unidirectional data flow.

A complete state management system should enable you to model a state — e.g., create a simple representation of what the state should look like, update its value, monitor the state when the value changes, and retrieve the values of the state. NgRx excels in managing complex states, making it ideal for applications with a lot of user interactions and multiple data sources.

**How Does Ngrx Work?**

NgRx is made up of 5 main components - Store, Actions, Reducers, Selectors, and Effects.

NgRx uses the Redux concept of unidirectional data flow, where all application data goes through the same lifecycle. This unidirectional data flow makes the application's state more predictable and thus easier to understand. This flow only applies to the state management layer and is not to be confused with the unidirectional data flow of the presentation layer. The following diagram shows the state management lifecycle in NgRx.



### **What is the Store in NgRx?**

The *S*tore is the most important component of NgRx. It provides a single store to express a global, application-wide state. The Store is based on a single, immutable data structure. The Store is also optimized for retrieving specific state data.

To access the Store, you can simply inject it. In the store, you can select data using selectors or dispatch actions.

### **What are Actions in NgRx?**

Actions express unique events that occur while using your web application. These can be user interaction with a particular page or external interaction through network requests or direct interaction with, for example, the device API.

Actions are dispatched via Store in NgRx and observed by NgRx's Reducers and Effects.

### **What are Reducers in NgRx?**

Reducers are responsible for handling transitions between states. Reducers react to the Actions dispatched and executes a pure function to update the Store. Pure functions are functions that are predictable and have no side effects. Given the same set of inputs, a pure function will always return the same set of outputs.

### **What are Selectors in NgRx?**

Selectors are pure functions for getting slices of the state from the Store. Selectors are how our application can listen to state changes.

### **What are Effects in NgRx?**

Effects handle the side effects of each Action. These side effects range from communicating with an external API via HTTP when a certain Action is dispatched to dispatching another Action to update another part of the State. Effects isolate side effects from components. This gives us "purer" components that select state and perform actions.

### **How to Use Ngrx?**

First you have to install the ngrx library in your Angular application.

Then the first thing you need to do is create a store. A store is a service that contains your application state. To do this, you can use the createStore() method from the @ngrx/store module:

@ngrx/store provides several different ways to create stores. For this example, we’ll use the createStore() method to create a simple store that contains a single piece of data: an array of strings representing our to-do items.

To create our store, we need to provide a reducer function. A reducer is simply a function that takes two arguments: the current state of our store, and an action object. The reducer will use the action object to determine how to update the state and then return the new state. In this example, our reducer simply appends the new string to our array of strings:

Once we have our reducer function, we can pass it into the createStore() method along with our initial state (an empty array):

Now that we have created our store, let’s take a look at how we can access its data and dispatch actions against it.

To access data from our store, we can inject it into any component or service using the Store class.

##### **What are the benefits of using Ngrx?**

Ngrx is a powerful tool for managing state in Angular applications. It provides a simple, yet powerful, way to keep track of data and maintain a consistent state across your application. In addition,Ngrx offers many benefits including:

1. **Improved performance** – ngrx uses a reducer function to update the state tree, which can optimize performance by avoiding unnecessary re-renderings.
2. **Easier debugging –** ngrx includes a built-in dev tools package that makes debugging easier by providing time-traveling capabilities and visualizations of the action stream.
3. **Better modularity** – ngrx promotes well-factored code by encouraging developers to think about their application state as distinct domains. This results in more manageable and reusable codebases.
4. **Greater flexibility** – ngrx supports multiple backends (such as React Native) and can be used with any other frontend libraries or frameworks (such as Vue or Preact).

## **When should you use NgRx?**

NgRx is a popular solution in the Angular ecosystem if you are building **Complex web applications** with sophisticated state management. Characteristics of the need for NgRx are many user interactions and multiple data sources.

## **When should you not use NgRx?**

NgRx is not a good choice, if you are building small Web Applications with isolated components or your team is not really familiar with the reactive approach.

**Conclusion**

In conclusion, NgRx is a powerful state management library for Angular that provides a predictable and centralized way of managing complex application states. By using NgRx, you can create maintainable and testable code that is easy to reason about. Its core concepts, such as actions, reducers, and effects, provide a consistent way of handling application states and side effects, resulting in scalable, efficient, and maintainable Angular applications.

We hope this blog has provided you with a better understanding of NgRx and how it can benefit your Angular applications.