**NGRX:**

ngrx is a state management library for Angular applications inspired by Redux. It's commonly used for managing the state of Angular applications, particularly those with complex data flows or large-scale applications.

Different Form Signal:

Signals are in build state management feature from angular(Version 16 onwards) . But it's not completely replacing NGRX and RXJs. may be future it can be permanent option

Let's simplify it further:

Imagine you're building a house (an Angular app). In your house, you have lots of rooms (components) where you keep different things (data). Now, you want to keep all your belongings organized and easy to manage.

Centralized Storage: ngrx is like a big storage room (the "store") where you can neatly arrange and store all your belongings (data) in one place. Instead of scattering your stuff all over the house, you put them in this central room.

Clear Rules for Changes: In ngrx, you have specific rules for how things can change. You write down these rules, like "if someone adds a new book, put it on the bookshelf". These rules are called "actions" and "reducers". So, whenever something changes (like a new book added), you follow these rules to keep everything organized.

Efficient Management: Let's say you have a lot of books and someone wants to add a new one. With ngrx, you don't need to check every room to see if there's space for the new book. Instead, you just go to the bookshelf (the store), check if there's room, and add the book there. It saves time and keeps things tidy.

Dealing with Complicated Tasks: Sometimes, you need help with complex tasks, like fixing a broken pipe or bringing in new furniture. ngrx provides tools (like "effects") that act like helpers. They handle these tasks for you, making sure everything runs smoothly without you having to worry too much.

Easy Testing: Just like you'd want to make sure your house is sturdy and safe, you also want to test your code to catch any bugs. ngrx helps with that by encouraging you to write code that's easy to test. So, you can check if everything works as expected before inviting guests (users) into your house (app).

In essence, ngrx helps you keep your Angular app organized, efficient, and easy to manage, just like having a well-organized storage room in your house.

By Nihar Techies:

(<https://www.youtube.com/watch?v=X5dWRn2Xqxs&list=PLfyWdpsiUiPC7bHmDDDM6gGgfo3mgMCAC&index=2> )

STATE =======

* State is an JSON object it can hold any type of data
* Un shared data, Form data & Router data no need to keep in the state
* State data can be changed but structure never changed

ACTIONS =========

* Any event performed in component need to dispatch action using action creators.
* Each action has the type property
* Action name should be unique
* Using payload option we can send data.

REDUCERS ==========

* Reducers are pure functions.
* It takes action & initial state as the input
* Reducer will calculate/finalize the data from action & initial state.
* Final state data will be available in store.

**WHAT IS NGRX?(By CodeSikho): 🤔(**[**https://www.youtube.com/watch?v=EGixFf47xN8&list=PL6GcqPPzylek8FzKSfPXnMYSf94lfsqxN**](https://www.youtube.com/watch?v=EGixFf47xN8&list=PL6GcqPPzylek8FzKSfPXnMYSf94lfsqxN) **)**

RXJS + REDUX = NGRX 🔥  
RXJS -> Observables

Redux is basically a global state management library taken from react.

All components can share this data among themselves.

**HOW NgRX REDUX WORKS?**

"There are three main parts of redux: **Actions**, **State**, & **Reducer**"

**STATE**

* **loading: false** → **loading: true**
* "Kind of global Variables"  
  *(Component will subscribe to these values of state)*

**1.State**

* Think of **state** as a "global variable" that stores data for your app.
* Example: loading: false means nothing is loading right now.
* The state changes over time, like loading: false → loading: true.

**2. Actions:**

**Any kind of action taken by user.It may contain some value called payload.**

* **Actions** are like messages that say, "I want to do something."
* Example: An action might say, "Start loading data!"
* This tells the app what needs to change in the state.

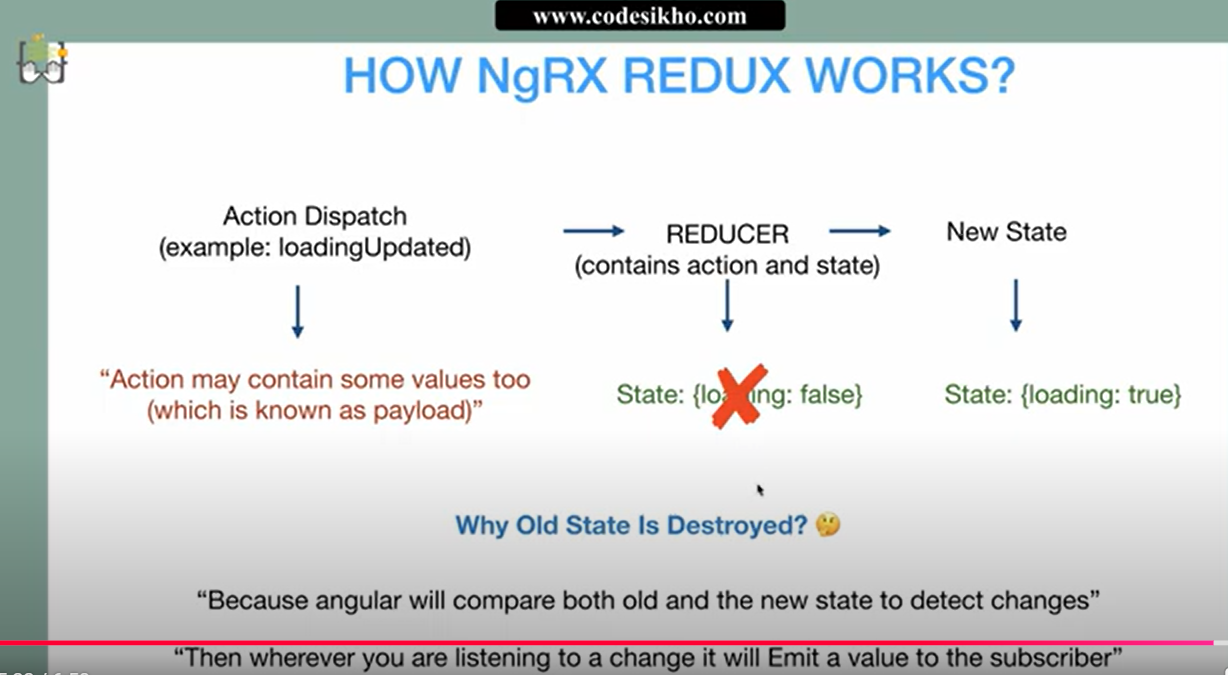
**3. Reducer**

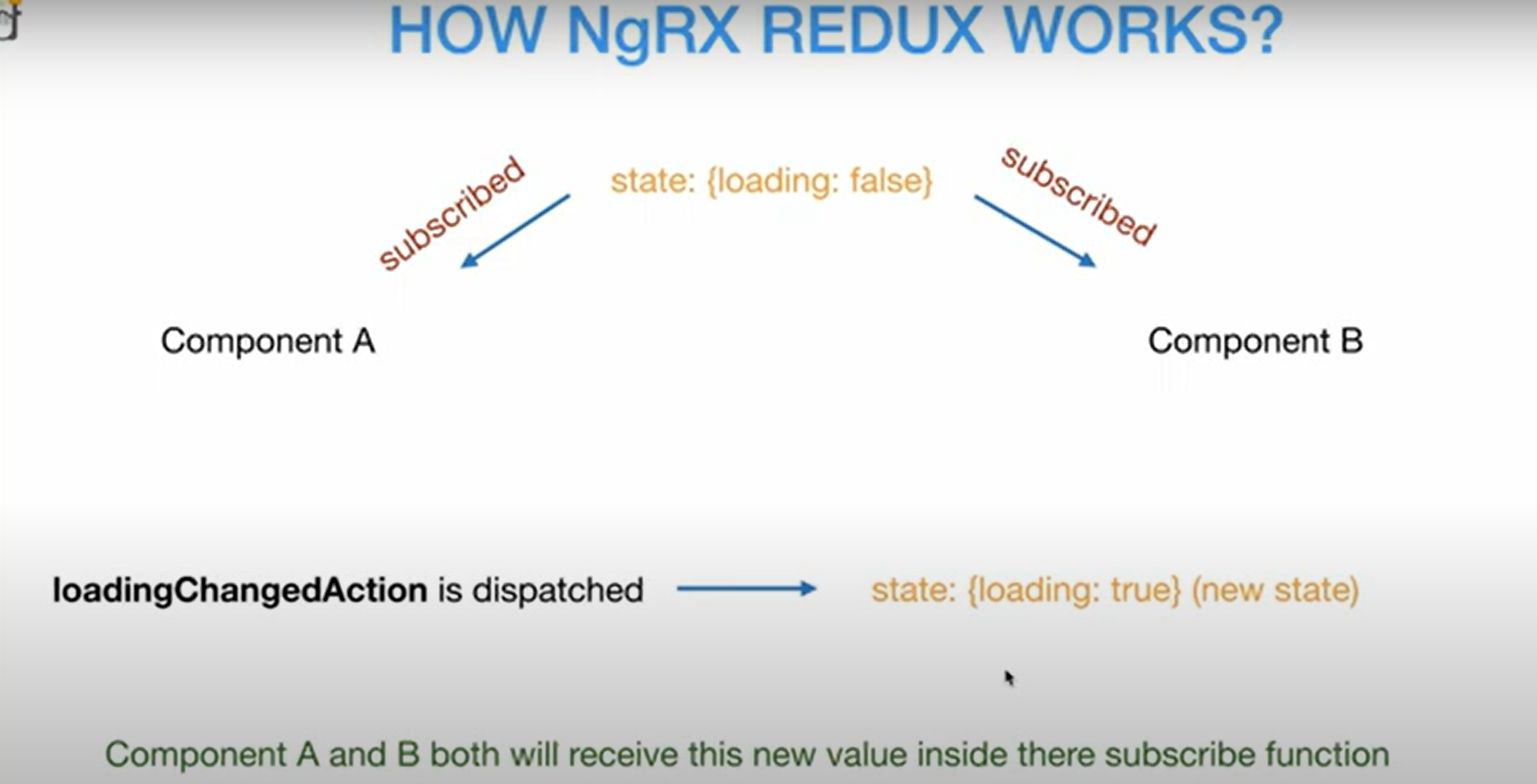
* A **reducer** takes the **action** and changes the **state** based on it. It will return new state.
* Example: When the "Start loading" action is sent, the reducer updates the state to loading: true.

**How It Works:**

1. **State** holds the current data.
2. **Action** sends a request to change the state.
3. **Reducer** applies the change, and the updated state is shared with all components.

This way, all components stay in sync with the central **state**.





**Installing with ng add:**

ng add @ngrx/store@latest

1. Update package.json > dependencies with @ngrx/store.
2. Run npm install to install those dependencies.
3. Update your src/app/app.module.ts > imports array with StoreModule.forRoot({})
4. If the project is using a standalone bootstrap, it adds [provideStore](https://ngrx.io/api/store/provideStore)() into the application config.