Angular Notes no. 1

Following notes are referenced from [here](https://youtu.be/vcfZ0EQpYTA?si=nFWW32T3LMPefHHU)

# Standalone templates vs Module based template

Stand one has app.config whereas module one has app.module

Both templates are available on angular 17 and onwards

Add ``--no-standalone`` while creating project for module based

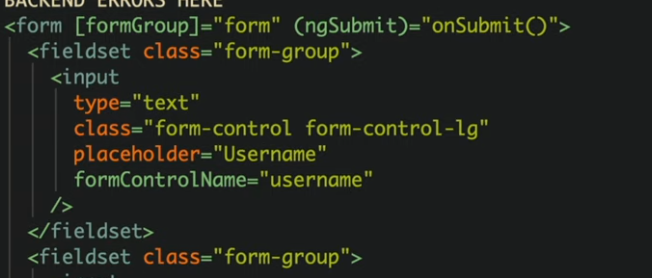
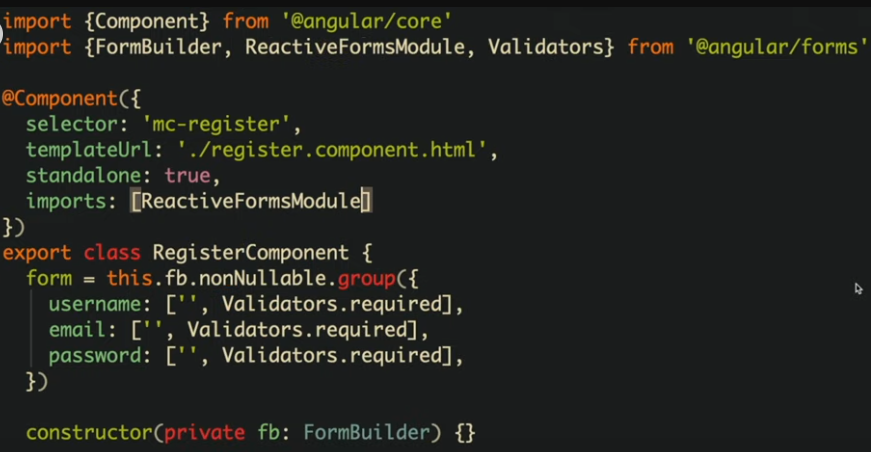
# Setup and some commands

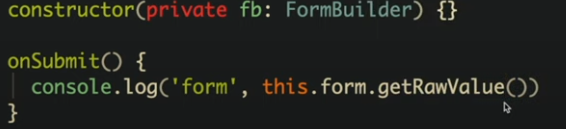
Start one with ``ng create <project\_name>`` also you can install angular cli locally and then install ``npx –p @angular/cli@15 ng new <project\_name>``

Create a component with ``ng g c home`` or ``ng g c component/home`` latter will create a component in component folder

# Angular Forms

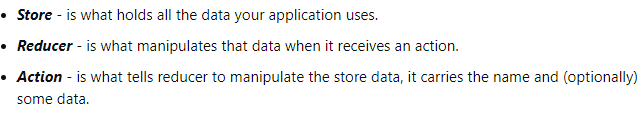
Use FormBuilder from ``@angular/forms``





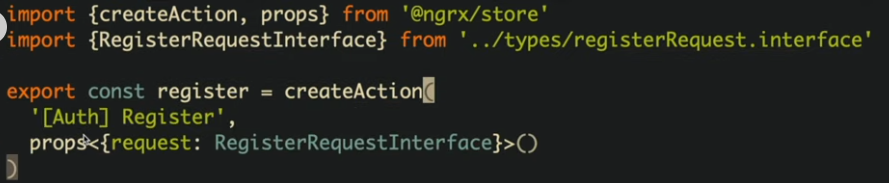
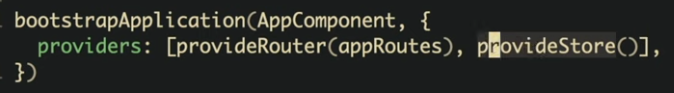
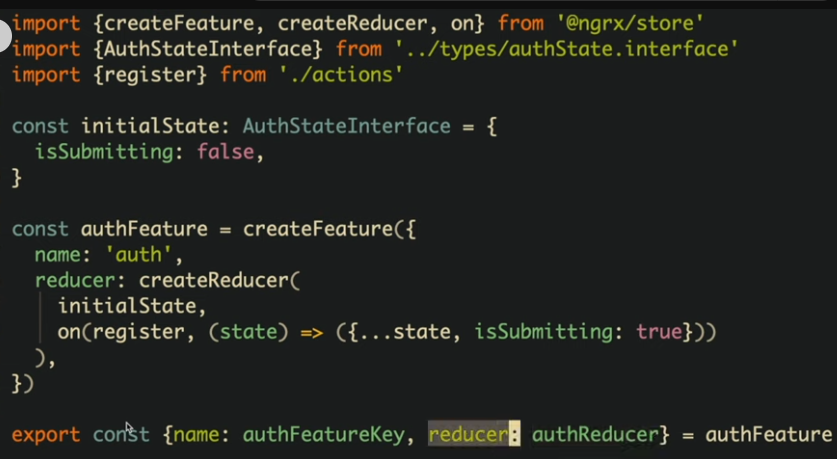
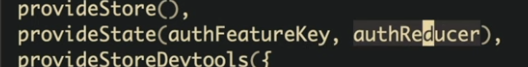
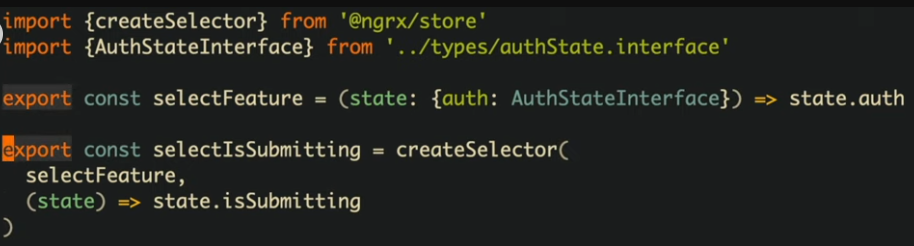
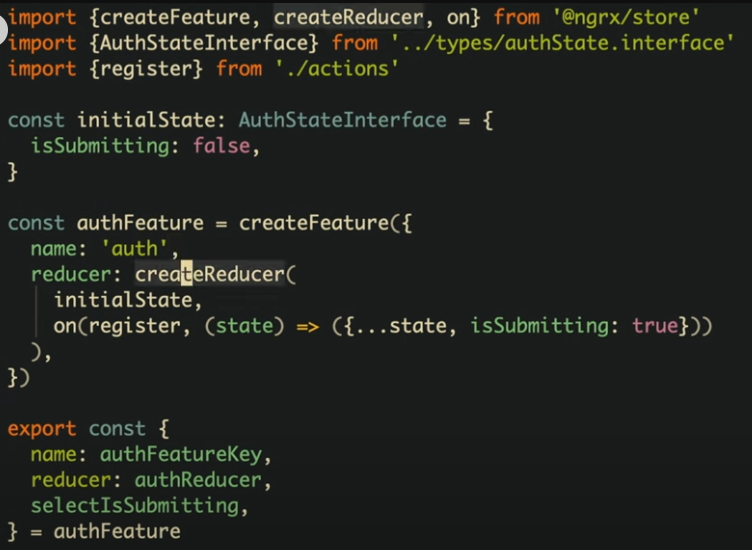
# Using NGRX

General Info:



* ***Slices*** – These are just individual states inside a group of states (store)

After installing ngrx following structure needs to be created:

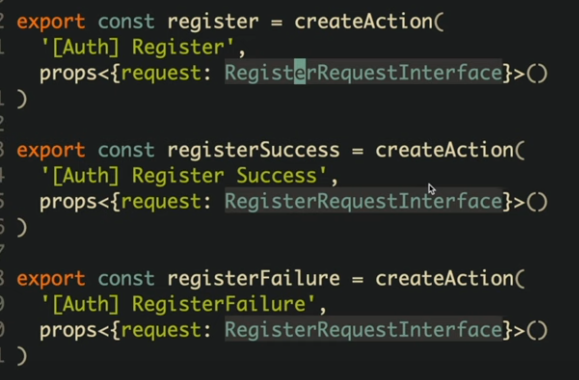
* Create a store folder inside it action.ts and write like: 
* Add the store provider in app.config.ts 
* Inject the ngrx store in the component: 
* And use the action as: 
* Setup `` @ngrx/store-devtools`` for devtools setup (optional)
* Create a newfile reducers.ts in that create feature/reducer (feature is just an abstraction around reducers to make the code easy): 
* Add the newly created feature in providers: 
* Create a file (selector.ts) and inside it: 
* Call it in your local component as ($ is added because it’s a observable): 
* We can avoid creating a selector by modifying the reducer function: 

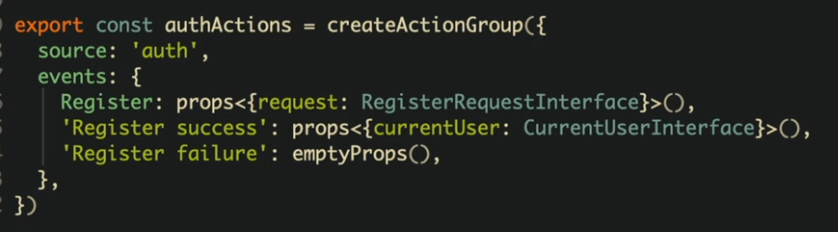
#### NGRX Effects

Ngrx effects are used to manage side effects or server-side states, its alternative in react is RTK query or redux-thunk or RTK`s createAsyncThunk

Start by ``npm I @ngrx/effects``

Then modify the actions like:



Or you can replace the above with: 

Now create a file effect.ts

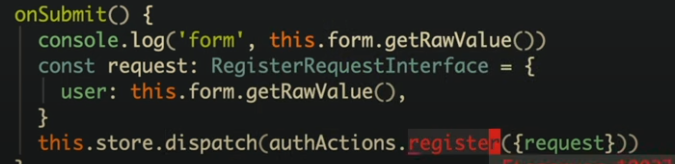


registerEffect is something which listens to some action, we are in above e.g. listening to actions register as specified in (authActions.register), switch map catches the success and error.

The reducer should be modified to



And the component should be modified to:

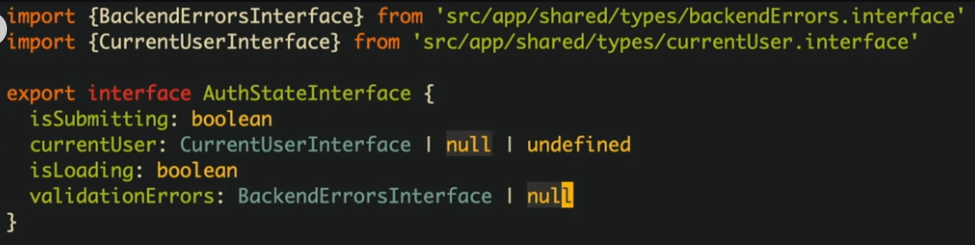


Effect needs to be added in providers list too

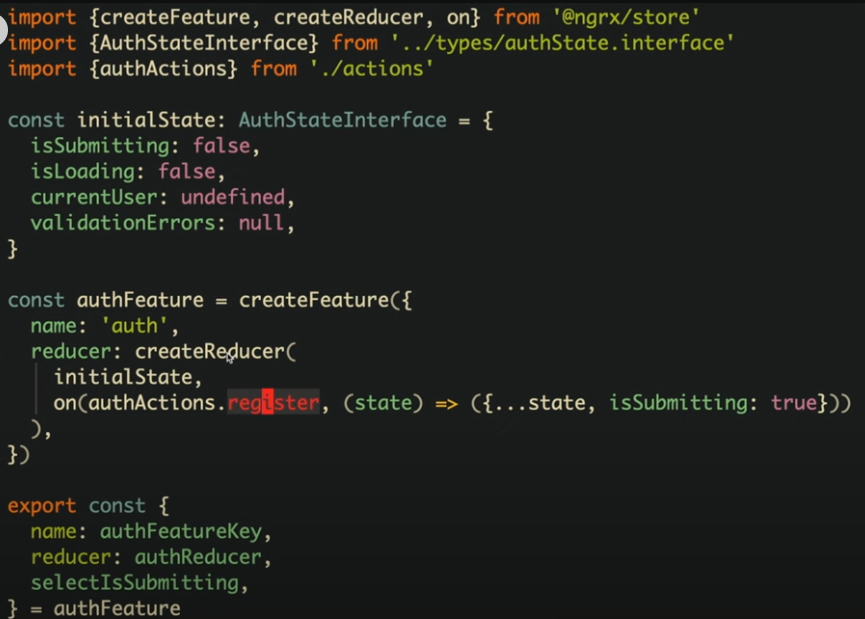
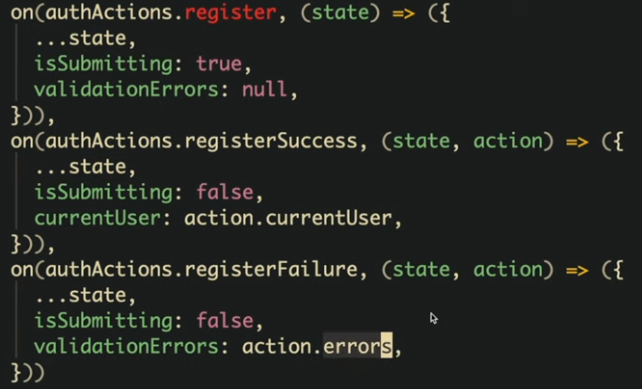
 

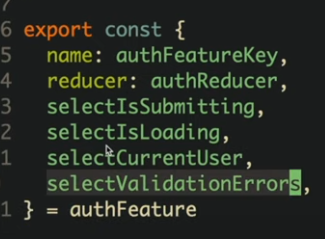
### Some modifications

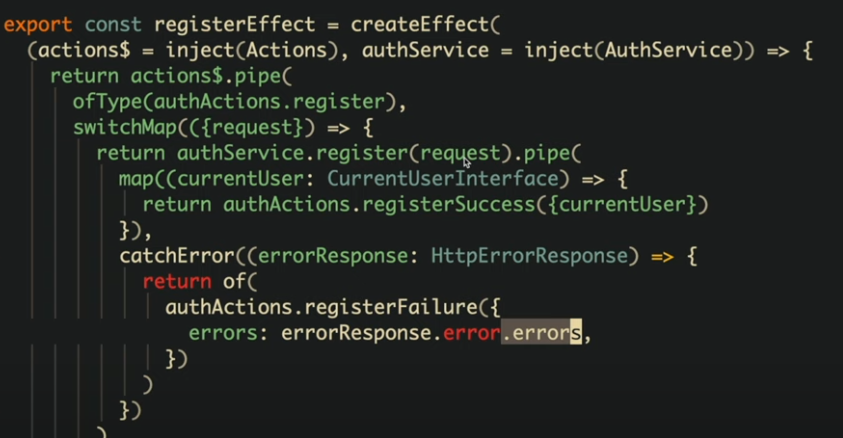
Update the interface



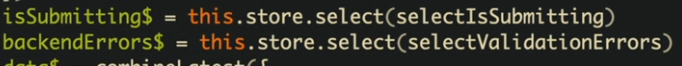
Modify the reducers

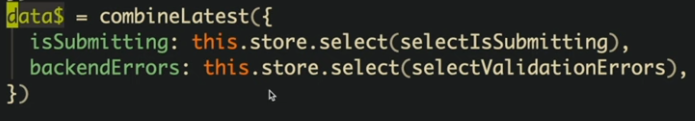
 



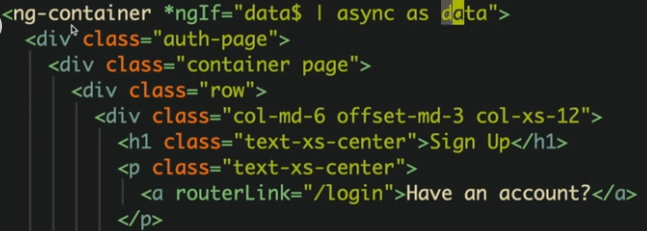
Lastly modify the errors effect: 

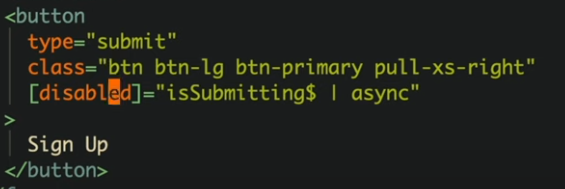
### Minimizing selectors

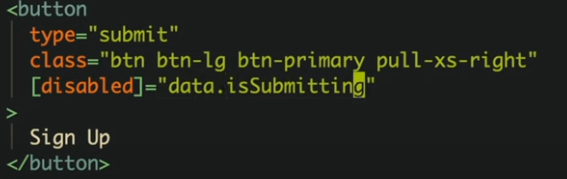
Instead of writing like this: 

We can write: 

We can use it like:



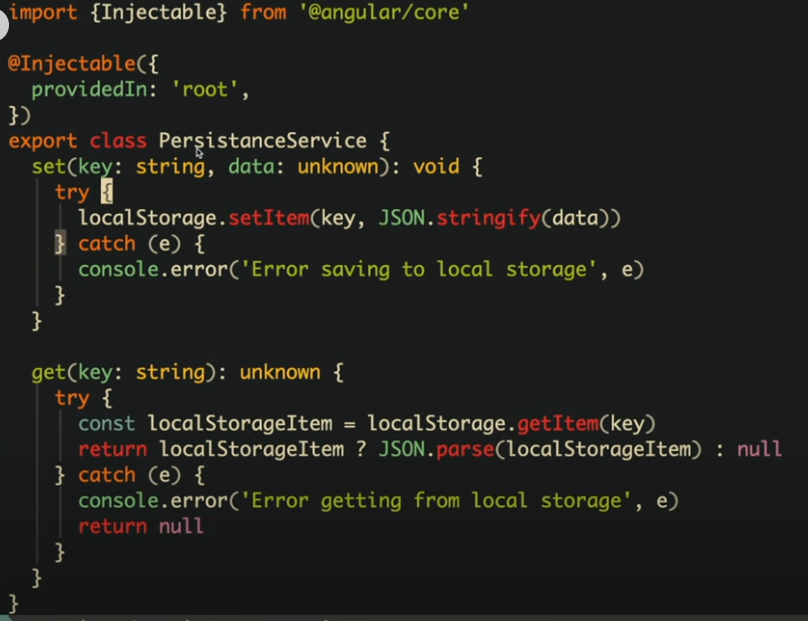
And instead of: 

We can use: 

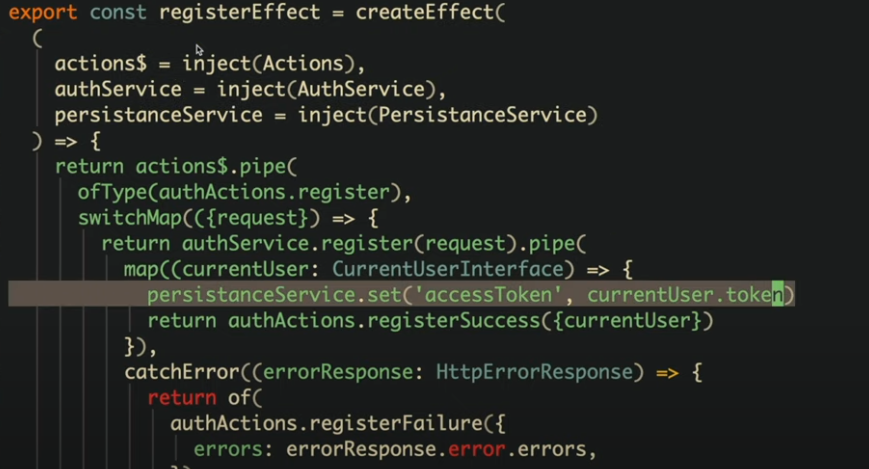
### Persisting States

We could use something:



But the above one is not reusable so create a persistent service: 

And use it in effects like:



### Extra effects



The above effect listens for registerSuccess and if found it redirects to homepage.

Dispatch is set to false because it does not return any actions

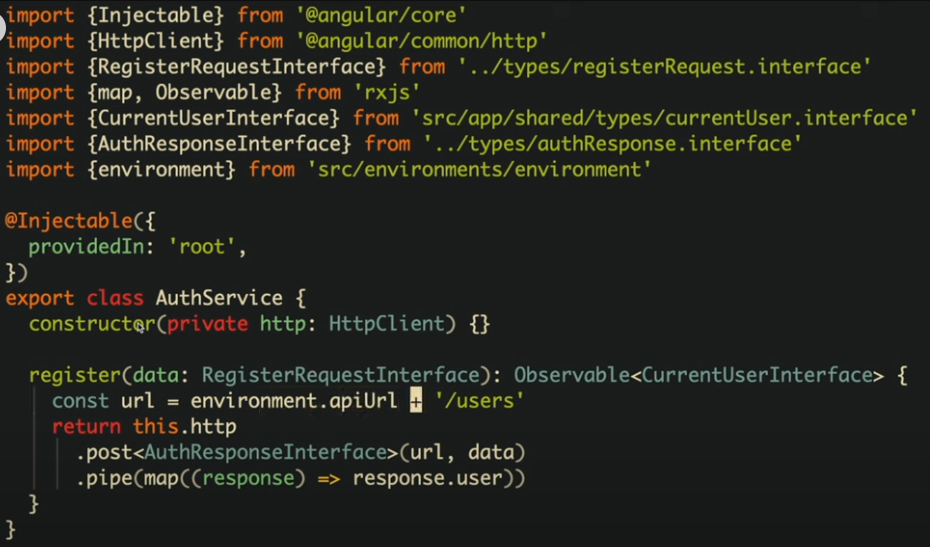
# Observables

These are just values to which you can subscribe and whenever these values changes asynchronously it will update the local state

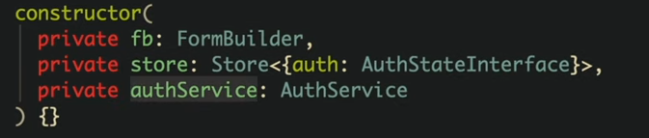
# Calling APIs

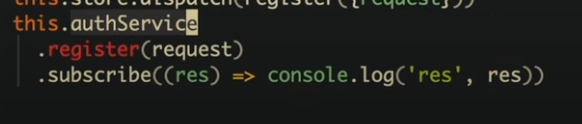
First add http client provider in main providers list

Create a service with following type code:



Calling the service, first inject it then call it





# Environments

Use command ``ng generate environments``

Angular Notes no. 2a

Following notes are referenced from [here](https://youtu.be/3qBXWUpoPHo?si=dChWLbEzk2NuR3fY)

## Topics covered:

Template Syntax

Binding Syntax

Banana Syntax Box syntax

Types of directives: 1) structural directives (change behaviour of DOM, slow) 2) attribute directives

builtin directives:

\*ngIf, \*ngFor, \*ngSwitch, ngClass, ngStyle

Pipes (used for data transformation without changing actual data)

builtin pipes:

DatePipe,UpperCasePipe, Lowercasepipe, currencypipe, decimalpipe, percentpipe, jsonpipe, slicepipe, asyncpipe

Lifecycle Hooks

8 hooks

from ngOnChanges, ngOnInit to ngOnDestroy

Ways of component communication:

1) using @Input and @Output

2) using @ViewChild and @ContentChild or @ContentChildren

3) using Services

Smart and Dumb Component

Content Projection

Dynamic loading of component

Accessing an HTML element using ref

DI

DI providers

## The topics below can only be implemented on ngmodule based project not on standalone project

TEMPLATE DRIVEN FORMS

FEATURE MODULING

NESTED AND CHILD ROUTINGS

LAZY LOADING OF A MODULE

ROUTE GUARDS (ROUTING EVENTS)

REACTIVE FORMS

CUSTOME PIPE

## THINGS YOU CAN GENERATE ON ANGULAR:

module [distribute application on basis of webpack and src files multiple files for SPAs]

guard [add routes guards/control to urls]

service (dependency) [are classes that can be called with instances in web; are injected; just some properties with methods]

directives [are components without template files]

components

interceptors

pipe

Angular notes no. 2b

Following notes are referenced from [here](https://youtu.be/3qBXWUpoPHo?si=dChWLbEzk2NuR3fY)