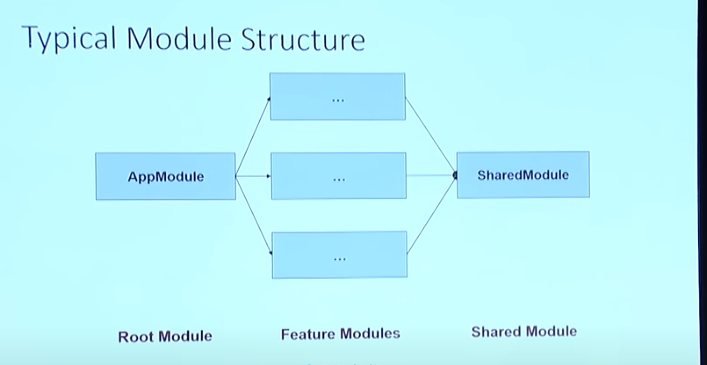
Large angular application

Generally application is divided in modules

AppModule is main module

Feature modules gets its own folder

Shared modules like validation

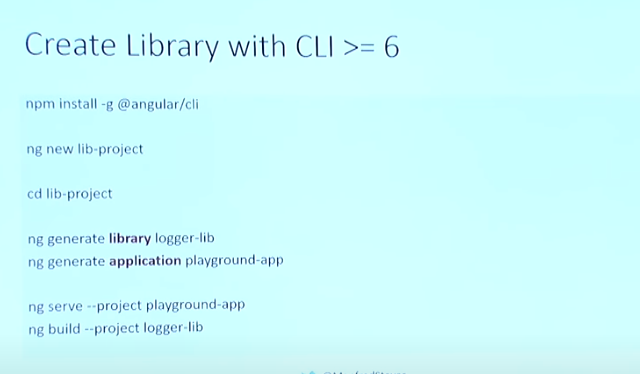


Big enterprise application consist of

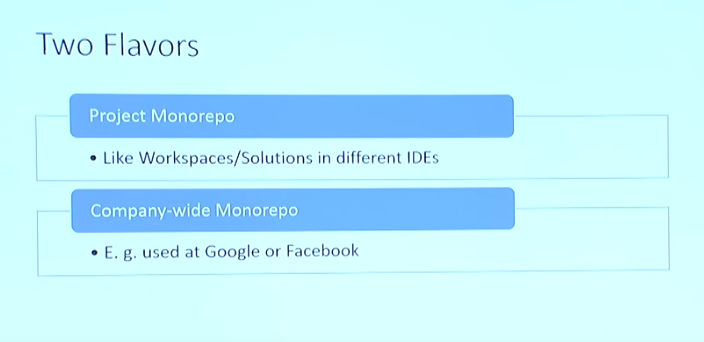
Npm packages

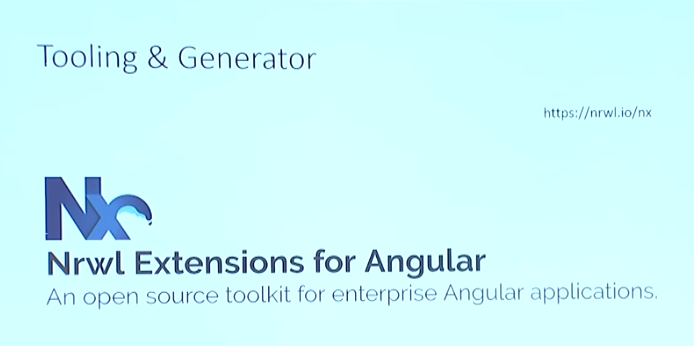
Monorepos

Microservices

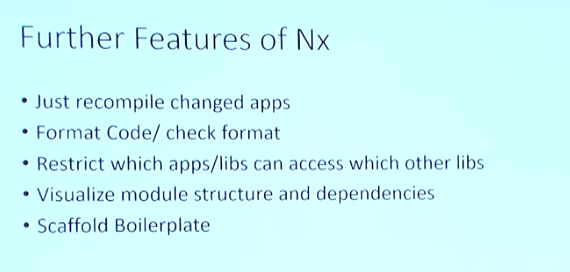


Monorepo structure: when we have only one repo to our multiple sub project, it is a way to substructure big project into tiny part



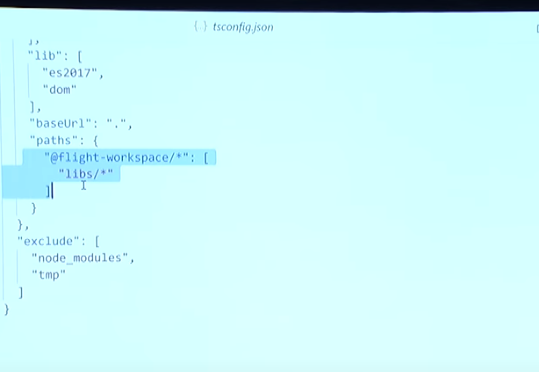


Nx is build over anglar cli to add some features

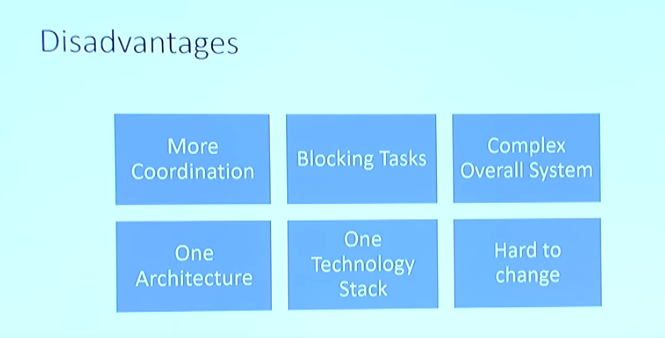


Including library modules in application

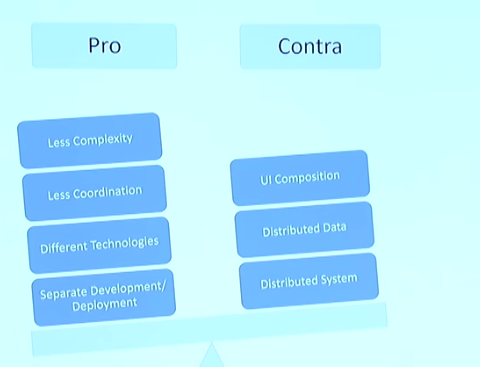




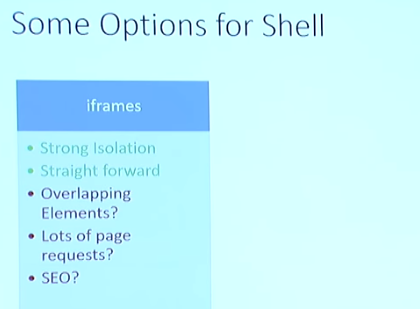
Disadvntage of mono repo – There is lot of dependency which comes with contract it becomes even more difficult when working with two different team



Solution to this **micro frontend**



Microfront ends are used for shell applications which allows us to load singke page applications on demand

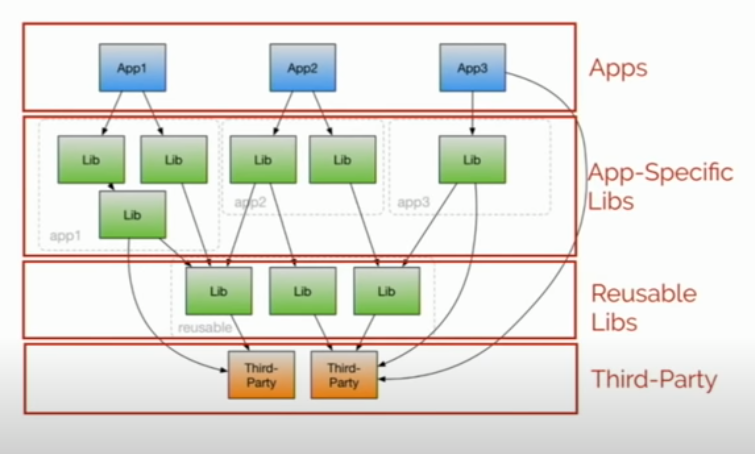


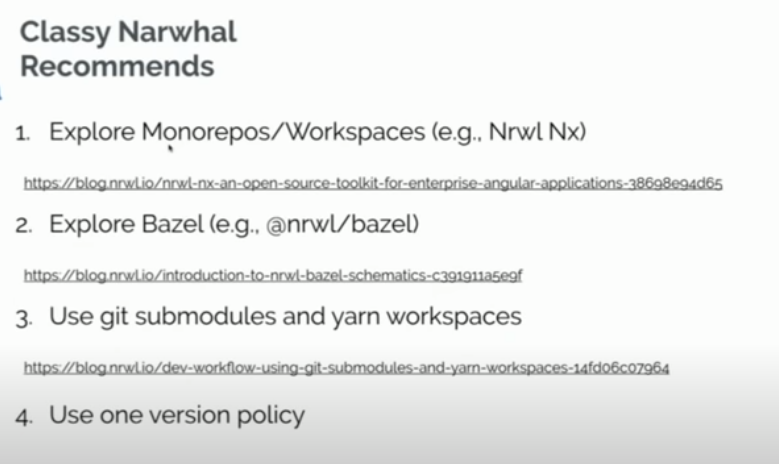
**Angular architecture using nx**

Why angular is good for large organization

Basically angular is framework which maintains uniformity everybody will use cli, router, use ngrx or may use in future, this uniformity brings much more consistency so if a new developer joins team he can become productive quickly, you can move from one team to another and remain productive which is big thing for company

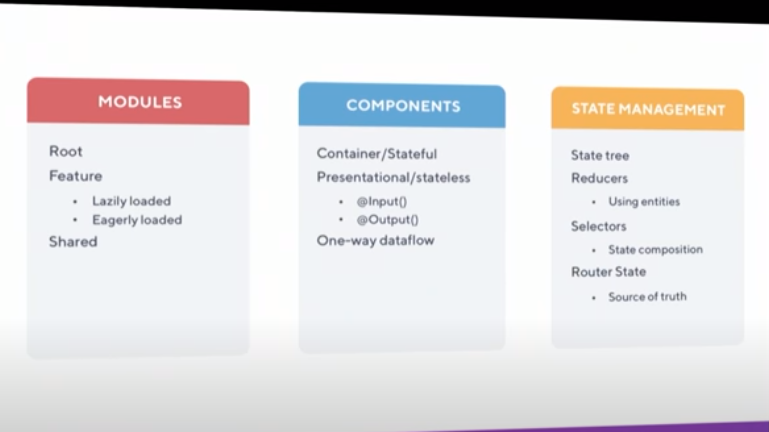
Angular very strong emphasis for automation which matters a lot for large application

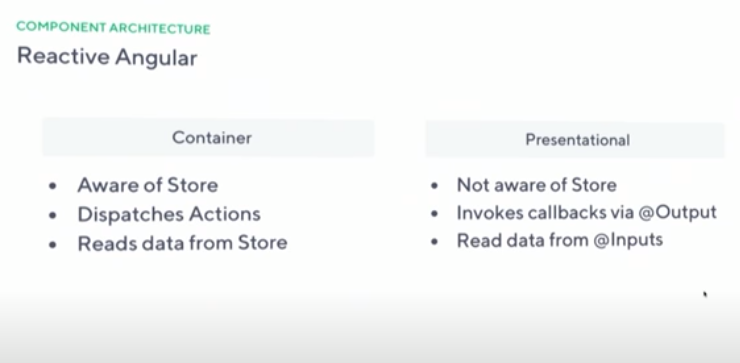


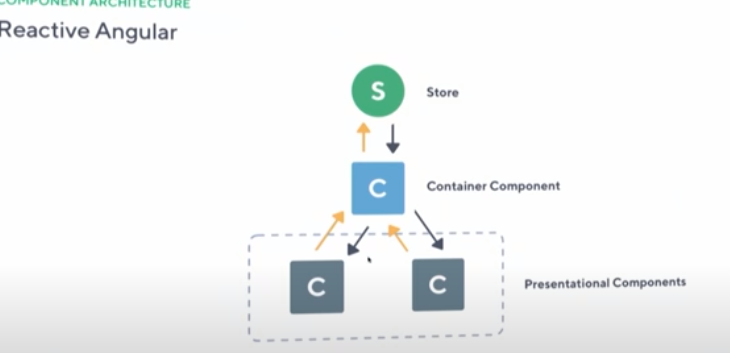


Promote best practices using tooling

A restaurant example Architecture









**Design architecture for angular application**

Since we have decided to use angular we will go ahead with **ngrx** for state management

Now I will divide application into modules and decide which modules are lazy loaded and which are not, in a module we will decide component which component will be container component or presentational component. In presentational component will not be any interaction with state all interaction with state will happen in container component only.

Directive we will use for common dom manipulation things.

Pipes we will use for common logics to transform data.

Services we will use for common logic required in component.

Since it is based on type script we will enforce type for everything using interface.

For state management we will uses ngrx and keep it independent of all above things,

<https://blog.angular-university.io/how-does-angular-2-change-detection-really-work/>