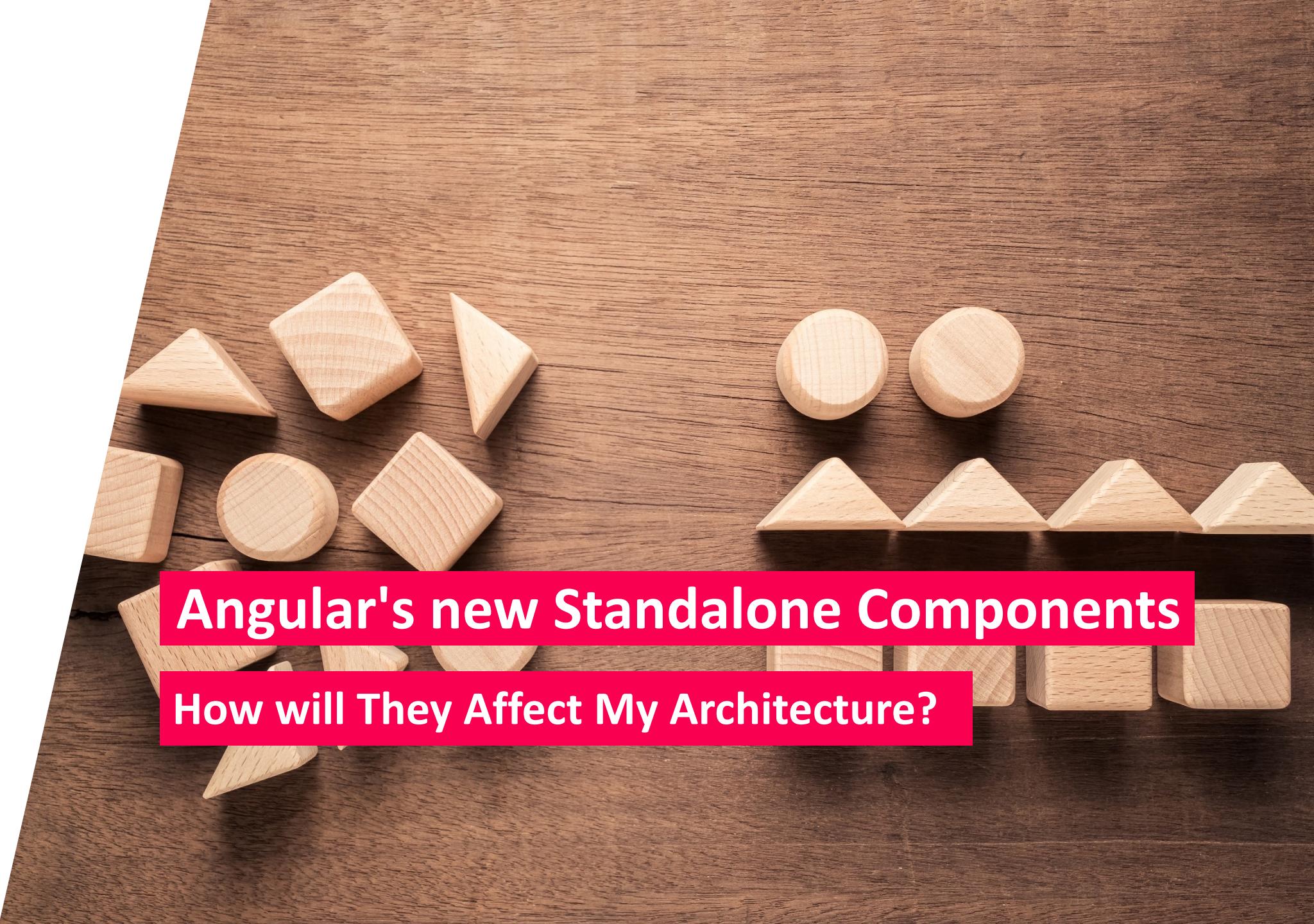




ANGULAR  
**ARCHITECTS**  
INSIDE KNOWLEDGE

# Angular's new Standalone Components

## How will They Affect My Architecture?



# NgModules Provide Context

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';

import { AppComponent } from './app.component';
[...]

@NgModule({
  imports: [BrowserModule, OtherModule],
  declarations: [AppComponent, OtherComponent, OtherDirective],
  providers: [],
  bootstrap: [AppComponent],
})
export class AppModule {}
```

## Compilation Context

# However ...

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';

import { AppComponent } from './app.component';
[...]
@NgModule({
  imports: [BrowserModule, OtherModule],
  declarations: [AppComponent, OtherComponent, OtherDirective],
  providers: [],
  bootstrap: [AppComponent],
})
export class AppModule {}
```

The diagram illustrates the integration of Angular Modules and TypeScript Modules. Two curved arrows point from the text labels to specific parts of the code. One arrow from 'Angular Modules' points to the `@NgModule` annotation. Another arrow from 'TypeScript Modules' points to the `import` statements.

# Standalone Components

```
@Component({
  standalone: true,
  selector: 'app-root',
  imports: [
    HomeComponent,
    AboutComponent,
    HttpClientModule,
  ],
  templateUrl: '...'
})
export class AppComponent {
  [...]
}
```

# Bootstrapping Components

```
bootstrapApplication(AppComponent);
```

# What Does this Mean for

... Compatibility?

... my Architecture?



ANGULAR  
**ARCHITECTS**  
INSIDE KNOWLEDGE

# Contents

#1

Mental Model

#2

Compatibility

#3

Routing

#4

Architecture



ANGULAR  
ARCHITECTS  
INSIDE KNOWLEDGE



SOFTWARE  
ARCHITECT

# #1: Mental Model



ANGULAR  
**ARCHITECTS**  
INSIDE KNOWLEDGE

# Just Imagine ...

Standalone Component = Component + NgModule  
(not implemented that way!)

# Mental Model

```
@Component({  
  standalone: true,  
  selector: 'app-root',  
  imports: [  
    RouterModule,  
    HomeComponent,  
    AboutComponent,  
  ],  
  templateUrl: '...'  
)  
export class AppComponent {  
  [...]  
}
```



```
@NgModule({  
  imports: [  
    RouterModule,  
    HomeComponentModule,  
    AboutComponentModule,  
  ],  
  declarations: [  
    AppComponent  
  ]  
)  
export class AppModule {  
  @Component({  
    selector: 'app-root',  
    templateUrl: '...'  
  })  
  export class AppComponent {  
    [...]  
  }  
}
```

# Standalone Pipes

```
@Pipe({
  standalone: true,
  name: 'city',
  pure: true
})
export class CityPipe implements PipeTransform {
  [...]
}
```

# Standalone Directives

```
@Directive({
  standalone: true,
  selector: 'input[appCity]',
  providers: [ ... ]
})
export class CityValidator implements Validator {
  [...]
}
```

# Importing Standalone Blocks

```
@Component({
  standalone: true,
  imports: [
    [...],
    FlightCardComponent,
    CityPipe,
    CityValidator,
  ],
  selector: 'flight-search',
  templateUrl: '...'
})
export class FlightSearchComponent {
  [...]
}
```

# Wish List

## Auto-Imports for Standalone Blocks



ANGULAR  
ARCHITECTS  
INSIDE KNOWLEDGE

# Click-Dummy

```
<my-comp *ngIf="show">  
</my-comp>
```

It looks like you want to use  
*NgIfDirective* and *MyComponent*.  
Shall I import it for you?



# #2: Compatibility



ANGULAR  
**ARCHITECTS**  
INSIDE KNOWLEDGE

# Standalone Component → NgModules

```
@Component({
  standalone: true,
  selector: 'app-root',
  imports: [
    [...]
    HttpClientModule,
  ],
  templateUrl: '...'
})
export class AppComponent {
```

# Standalone Component → NgModules

```
@Component({  
  standalone: true,  
  selector: 'app-root',  
  imports: [  
    [...]  
    HttpClientModule,  
  ],  
  templateUrl: '...'  
)  
export class AppComponent {  
}
```

# NgModule → Standalone Blocks

```
@NgModule({  
  imports: [  
    FlightCardComponent,  
  ],  
  declarations: [  
    MyTicketsComponent  
  ],  
})  
export class TicketsModule { }
```

# NgModule → Standalone Blocks

```
@NgModule({  
  imports: [  
    FlightCardComponent,  
  ],  
  declarations: [  
    MyTicketsComponent  
  ],  
})  
export class TicketsModule { }
```

# Global Providers

```
bootstrapApplication(AppComponent, {  
  providers: [  
    MyGlobalService,  
    provideRouter(APP_ROUTES),  
  ]  
});
```

# Global Providers

```
bootstrapApplication(AppComponent, {  
  providers: [  
    MyGlobalService,  
    provideRouter(APP_ROUTES)  
  ]  
});
```



**Dedicated function for app routes**

# #3: Routing



ANGULAR  
**ARCHITECTS**  
INSIDE KNOWLEDGE

# Global Providers

```
bootstrapApplication(AppComponent, {  
  providers: [  
    MyGlobalService,  
    provideRouter(APP_ROUTES)  
  ]  
});
```

# Lazy Loading

```
export const APP_ROUTES: Routes = [
  [...],
  {
    path: 'flight-booking',
    loadChildren: () =>
      import('@nx-example/booking/feature-book')
        .then(m => m.FLIGHT_BOOKING_ROUTES)
  },
  {
    path: 'next-flight',
    loadComponent: () =>
      import('@nx-example/booking/feature-tickets')
        .then(m => m.NextFlightComponent)
  },
];

```

# Lazy Loading

```
export const APP_ROUTES: Routes = [
  [...],
  {
    path: 'flight-booking',
    loadChildren: () =>
      import('@nx-example/booking/feature-book')
        .then(m => m.FLIGHT_BOOKING_ROUTES)
  },
  {
    path: 'next-flight',
    loadComponent: () =>
      import('@nx-example/booking/feature-tickets')
        .then(m => m.NextFlightComponent)
  },
];

```

# Routes With Injector

```
export const FLIGHT_BOOKING_ROUTES: Routes = [{  
  path: '',  
  component: FlightBookingComponent,  
  providers: [  
    MyService  
,  
    children: [  
      [...]  
    ]  
  ]  
}];
```

# Routes With Injector

```
export const FLIGHT_BOOKING_ROUTES: Routes = [{  
  path: '',  
  component: FlightBookingComponent,  
  providers: [  
    MyService  
  ],  
  children: [  
    [...]  
  ]  
}];
```

Scope: This route  
+ all child routes



# Routes With Injector

```
export const FLIGHT_BOOKING_ROUTES: Routes = [{  
  path: '',  
  component: FlightBookingComponent,  
  providers: [  
    ...forMyLibrary()  
  ],  
  children: [  
    [...]  
  ]  
}];
```

# Routes With Injector

```
export const FLIGHT_BOOKING_ROUTES: Routes = [{  
  path: '',  
  component: FlightBookingComponent,  
  providers: [  
    ...forDomainBooking()  
  ],  
  children: [  
    [...]  
  ]  
}];
```

# #4: Architecture

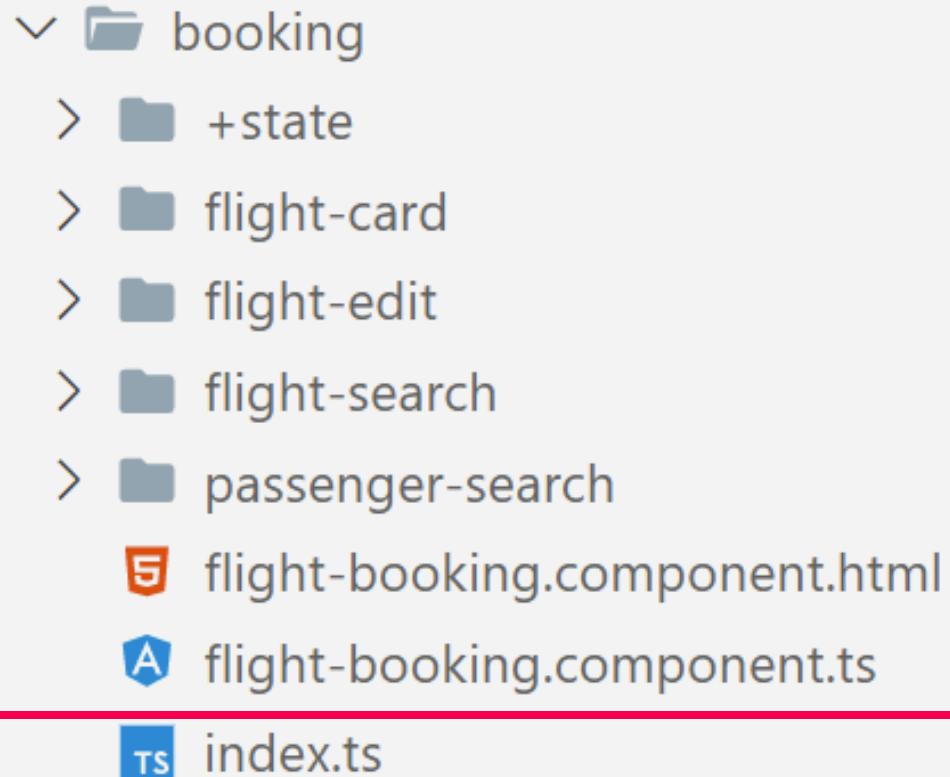


ANGULAR  
**ARCHITECTS**  
INSIDE KNOWLEDGE

# Folder

```
✓  folder booking
  > folder +state
  > folder flight-card
  > folder flight-edit
  > folder flight-search
  > folder passenger-search
      5 flight-booking.component.html
      A flight-booking.component.ts
```

# Barrels



```
// index.ts == Public API

export *
from './flight-booking.component';

export *
from './flight-card/flight-card.component';
```

# Importing Whole Barrels

```
import * as booking from './booking';
[...]

@Component({
  standalone: true,
  imports: [
    ...Object.values(booking) as any[],
    [...]
  ],
  [...]
})
export class MyComponent {
  [...]
}
```

# Importing Whole Barrels

```
import * as booking from './booking';
[...]

@Component({
  standalone: true,
  imports: [
    ...Object.values(booking) as any[],
    [...]
  ],
  [...]
})
export class MyComponent {
  [...]
}
```

Not beautiful!

# Importing Whole Barrels

```
import * as booking from './booking';
[...]

@Component({
  standalone: true,
  imports: [
    ...all(booking),
    [...]
  ],
  [...]
})
export class MyComponent {
  [...]
}
```

Custom Helper Function



# Wish List

Better syntax for  
importing barrels



ANGULAR  
**ARCHITECTS**  
INSIDE KNOWLEDGE

# Importing Whole Barrels

```
import * as booking from './booking';
[...]

@Component({
  standalone: true,
  imports: [
    booking,
    [...]
  ],
  [...]
})
export class MyComponent {
  [...]
}
```

Wish List



ANGULAR  
ARCHITECTS  
INSIDE KNOWLEDGE



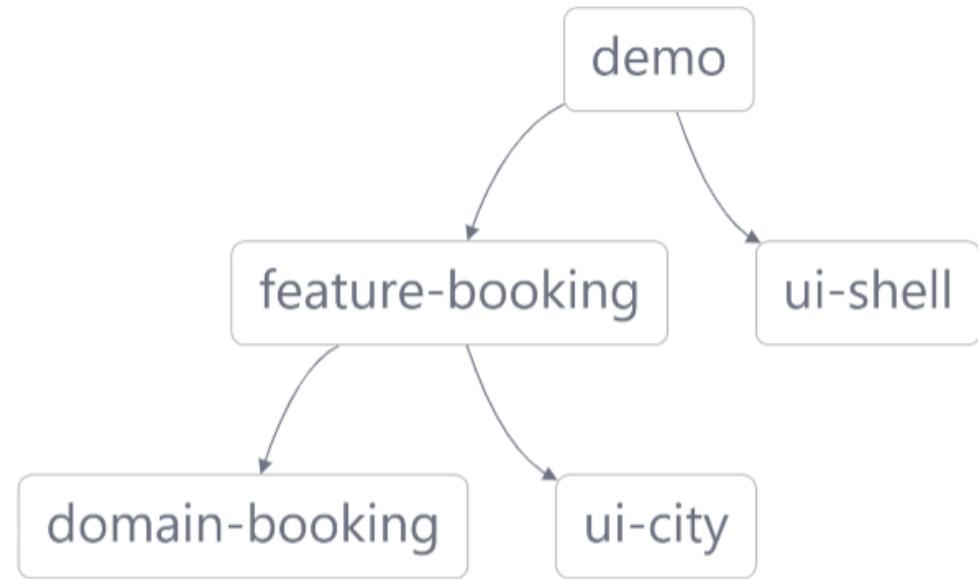
SOFTWARE  
ARCHITECT

# Next Logical Step: Nx Workspaces



ANGULAR  
**ARCHITECTS**  
INSIDE KNOWLEDGE

```
✓ libs
  > domain-booking
  ✓ feature-booking
    ✓ src
      ✓ lib
        > flight-card
        > flight-edit
        > flight-search
        > passenger-search
        ⚡ flight-booking.component.html
        ⚑ flight-booking.component.ts
        TS index.ts
        TS test-setup.ts
      > ui-city
      > ui-shell
```



- + Generates path mappings
- + Generates initial barrel
- + Prevents bypassing *index.ts*
- + much more

# Constraints: "No Broken Windows!"

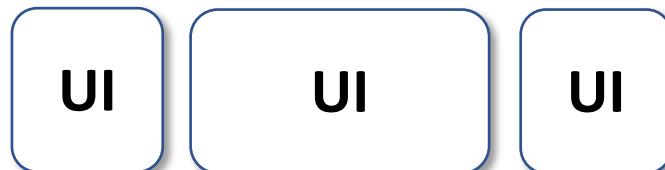
```
import { Component } from '@demo/util-shim';
import { FlightBookingComponent } from '@demo/feature-booking';
```

"FlightBookingComponent" ist deklariert, aber der zugehörige Wert wird nie gelesen. ts(6133)

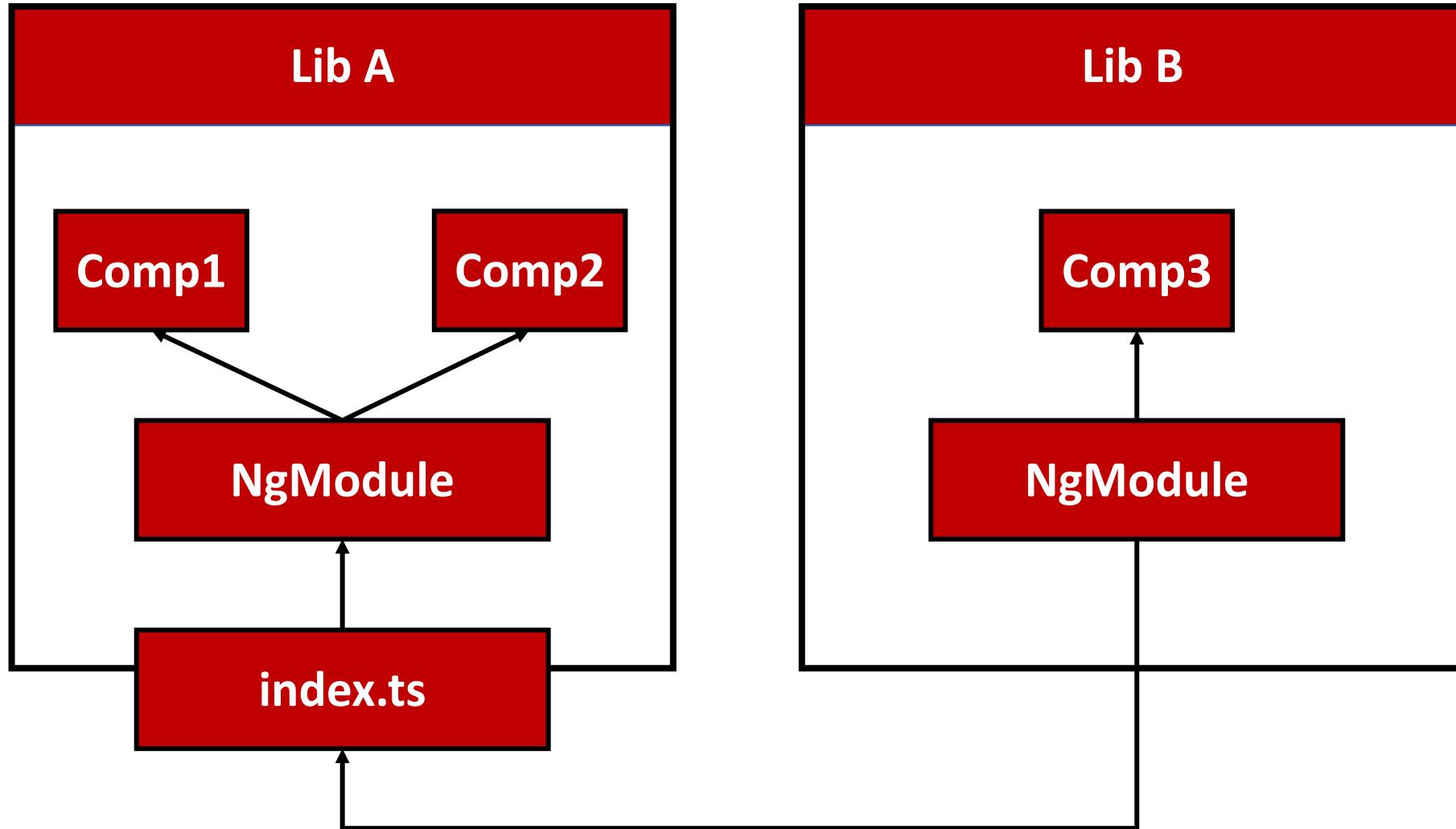
A project tagged with "ui" can only depend on libs tagged with "domain", "util" [eslint\(@nrwl/nx/enforce-module-boundaries\)](#)

[Problem anzeigen](#) [Schnelle Problembehebung ... \(STRG+.\)](#)

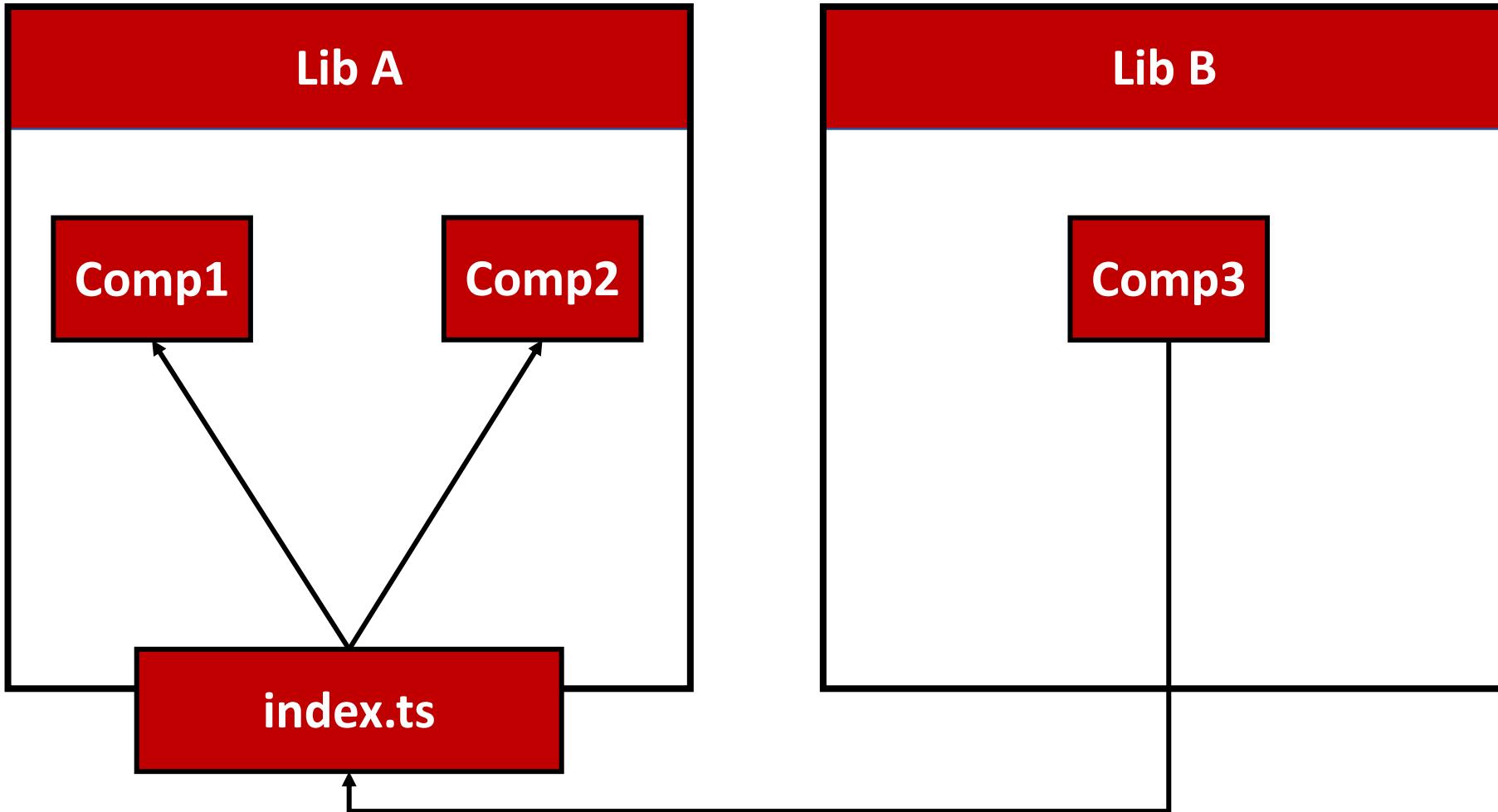
```
export class NavbarComponent {
  sidebarVisible = false;
```



# Today (with NgModules)



# Tomorrow (without NgModules)



# Conclusion

Mental Model

Folders & Barrels

Mapped Paths

Nx, Libs, and  
Constraints FTW!



ANGULAR  
ARCHITECTS  
INSIDE KNOWLEDGE

