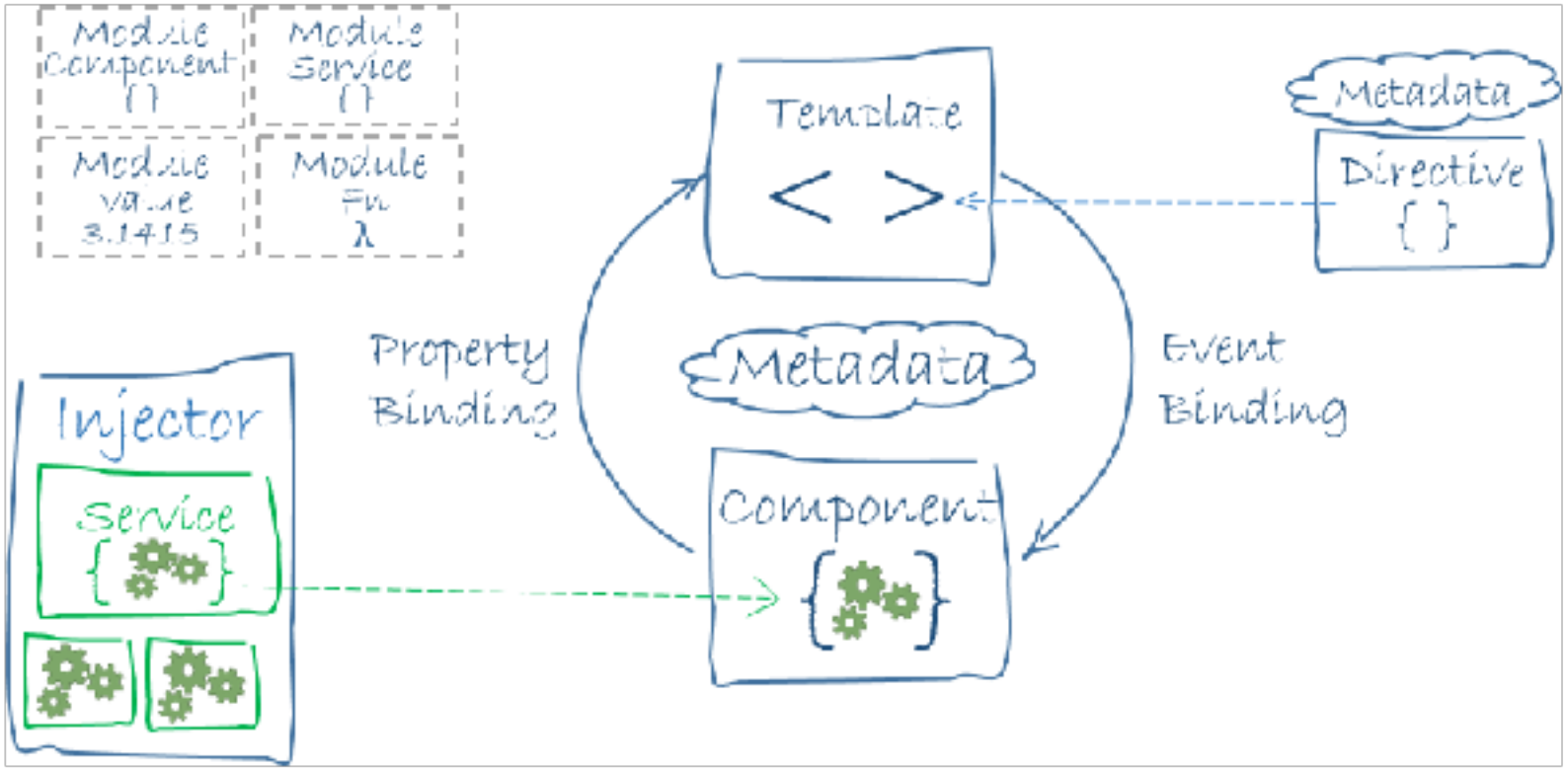
Modules & Components

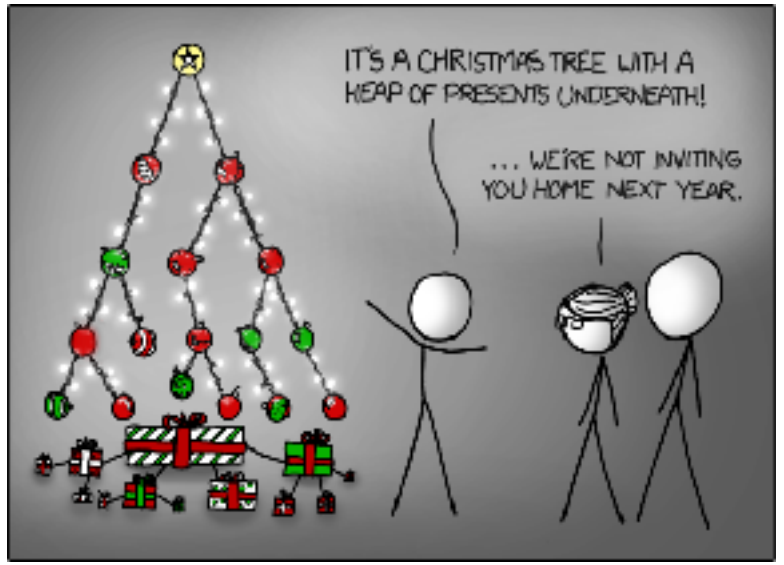
Modules & Components

Application architecture



Modules

- Blueprint for available features
- Starts with a Root Module
- Feature Modules add functionality
- Multiple 'types' of modules
- Tree structure



Components

- Components describe a 'part of the screen'
- Behaviour
- Template
- Tree structure

Bootstrapping

- **main.ts**
- initialise AppModule
- create Trees
- start Change Detection

ilionx
Your partner in digital business

Tech Influencers

Contacten

- Martin Fowler
- Uncle Bob
- **Elon Musk**
- Bill Gates

Details

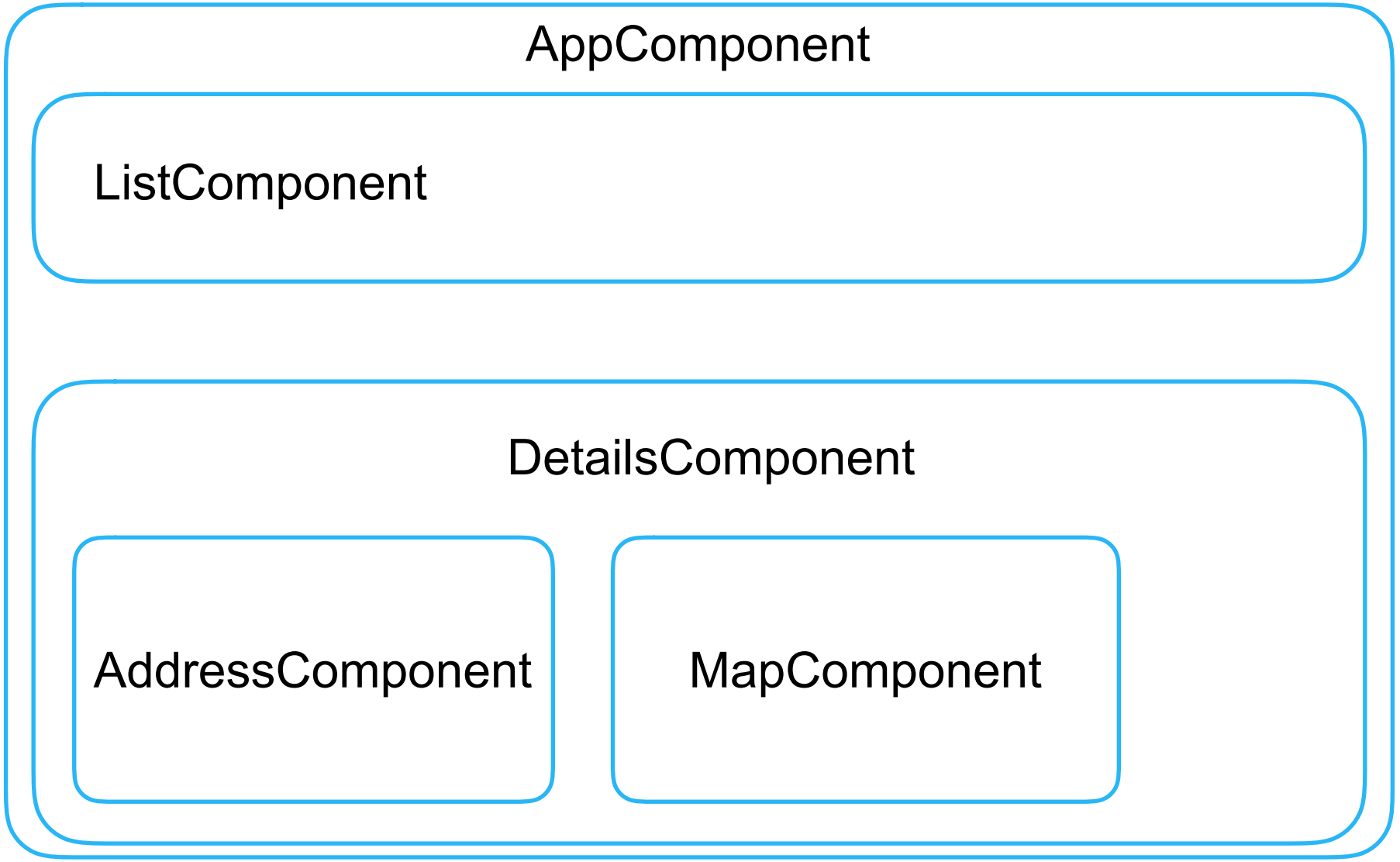
Elon Musk

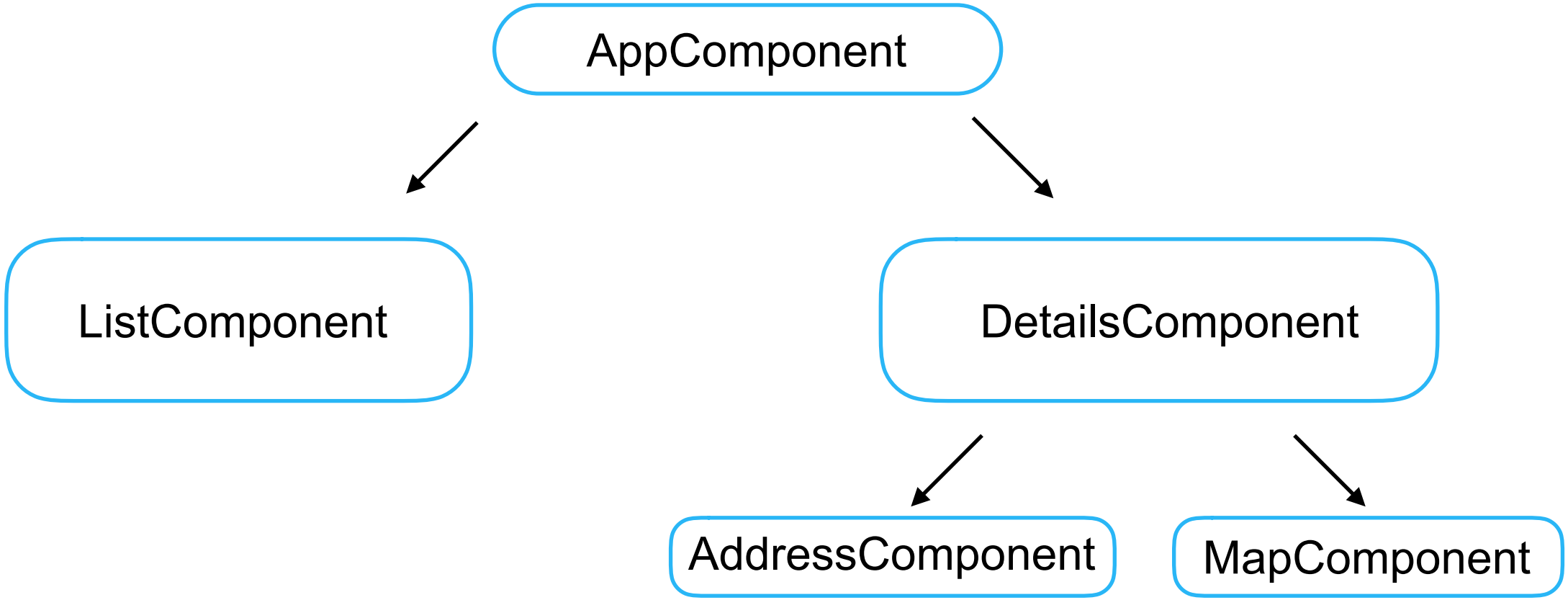
Adres

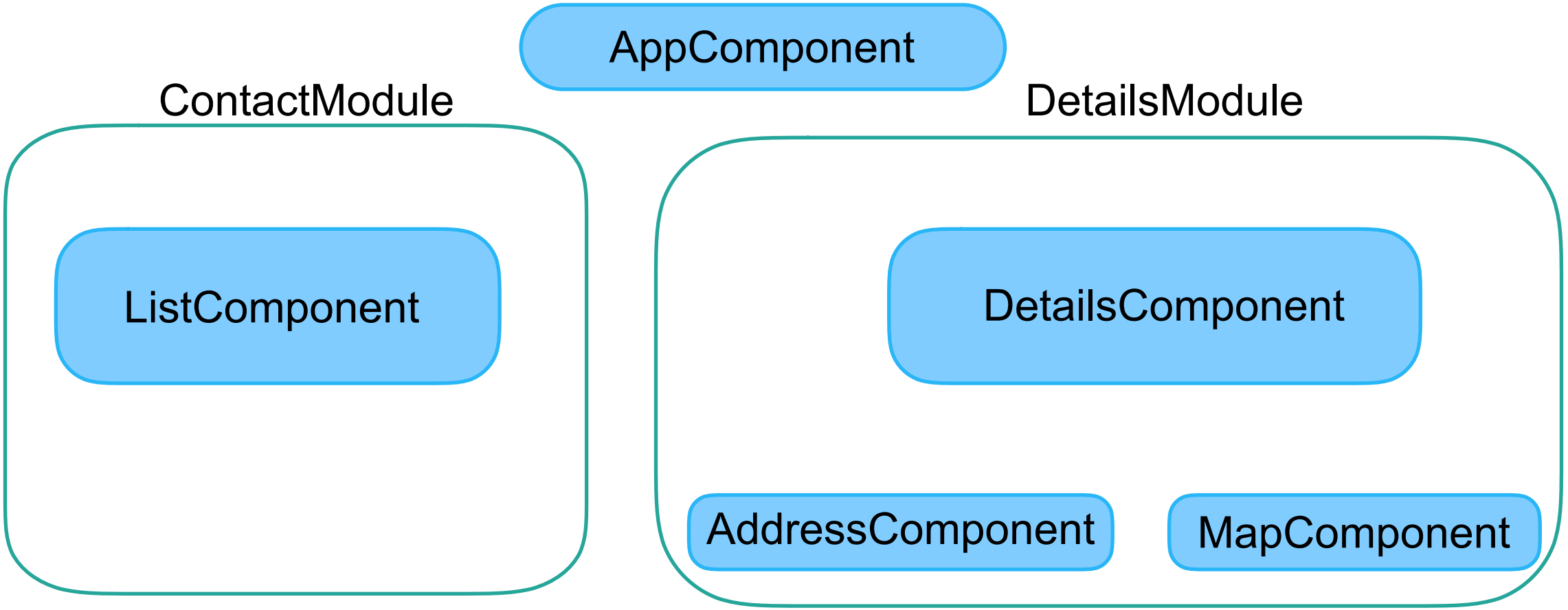
Tesla Factory

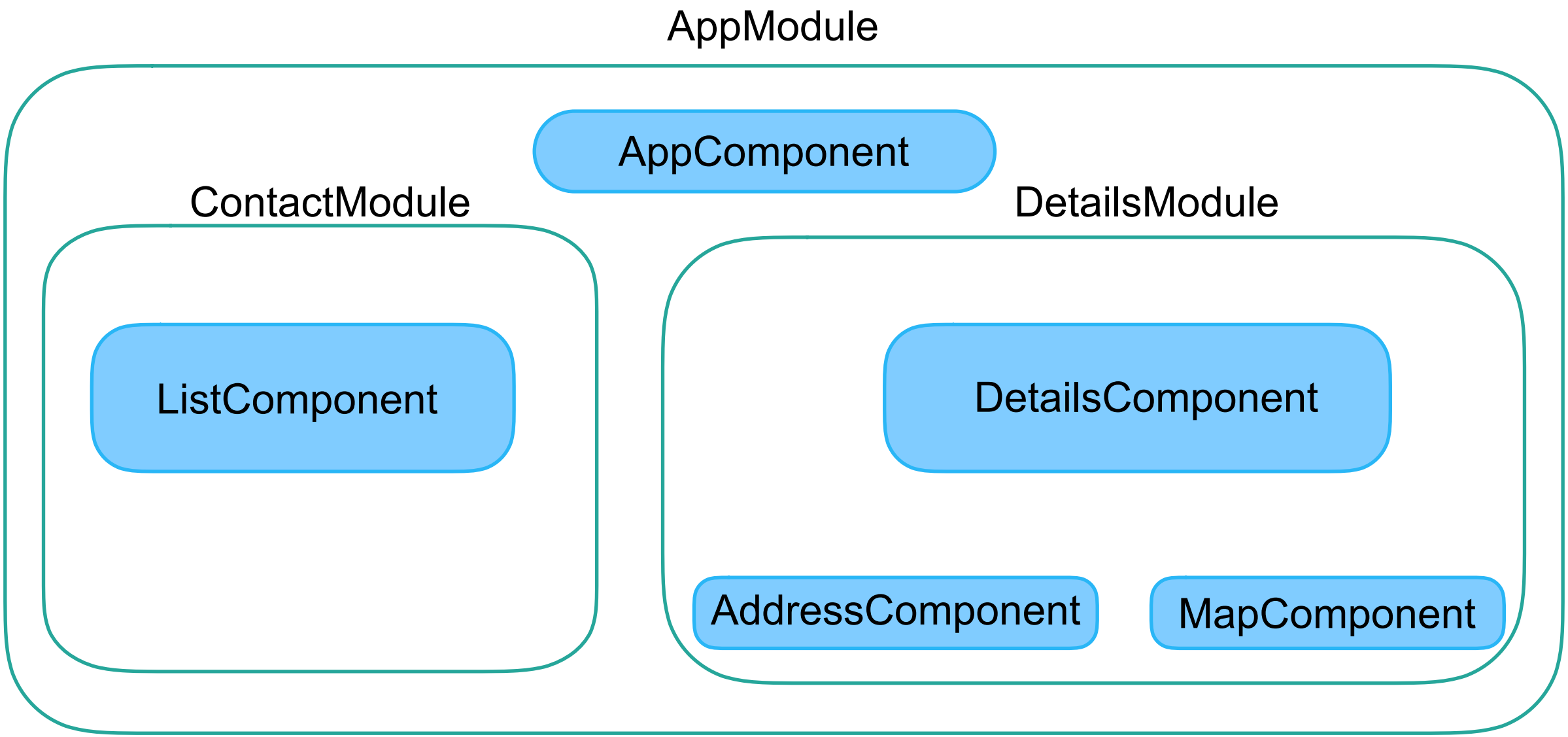
45500 Fremont Boulevard

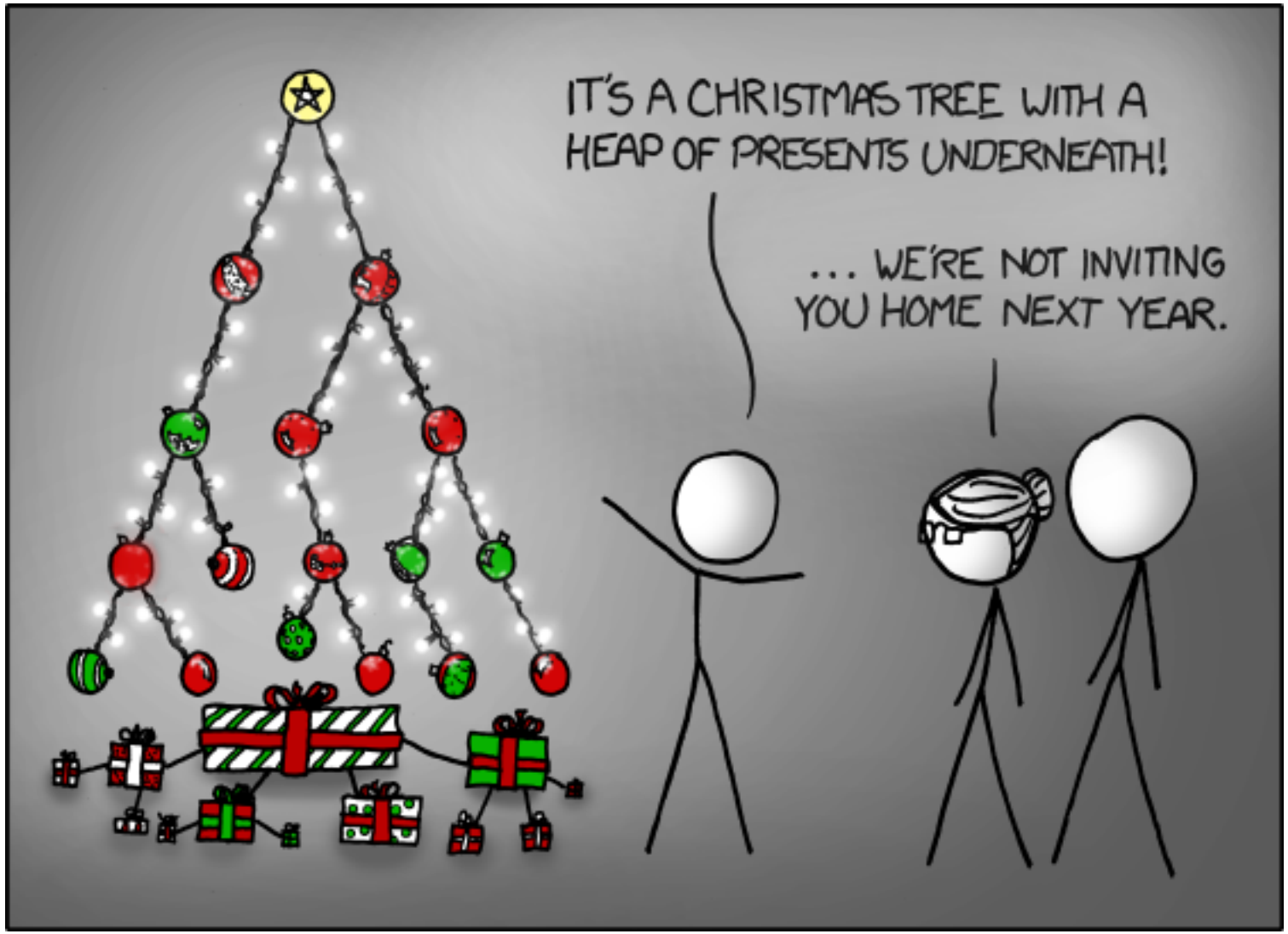
Fremont, CA 94538











Tasks NgModule

- **Declare** components*
- **Bootstrap** Root Component
- **Import** modules
- **Export** components*

@NgModule

```
@NgModule({
  declarations: [AppComponent],
  imports: [FeatureModule]
})
export class AppModule {

    ...
}
```

app.module.ts

Declaring components

Register components, directives and pipes with the module

AppModule

```
declarations: [AppComponent, …]
```

Root component

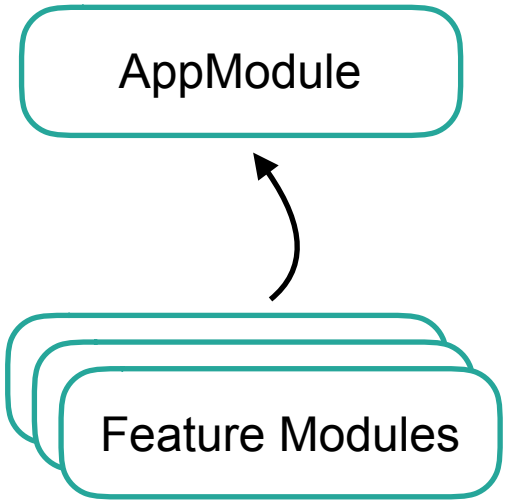
Only the Root Component is bootstrapped

AppModule

```
declarations: [AppComponent, …]
bootstrap: [AppComponent]
```

Feature modules

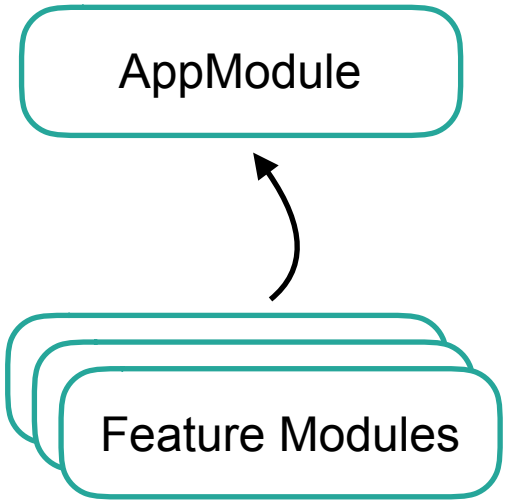
Feature Modules add functionality to modules




```
imports: [FeatureXModule, ...]
```

Feature components

Components are scoped within the module

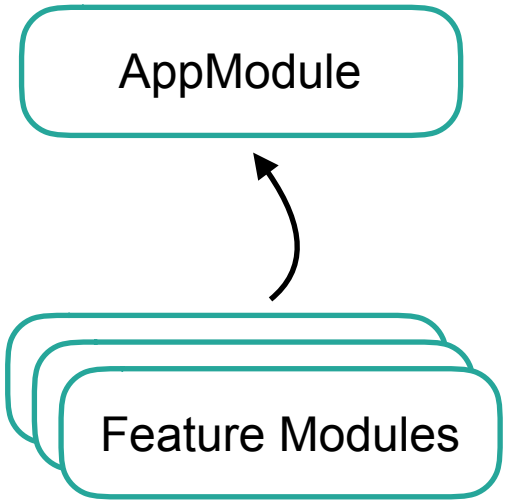



```
declarations: [YComponent, ...]
```

only available in
declared module

Feature components - export

Exports makes components* available to other modules

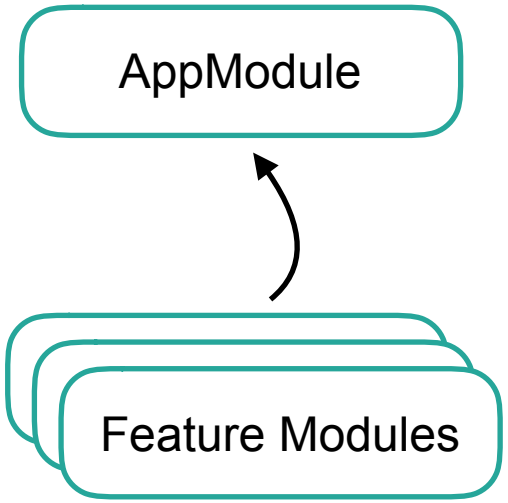



```
declarations: [YComponent, ...]
exports: [YComponent, ...]
```

make available in parent module

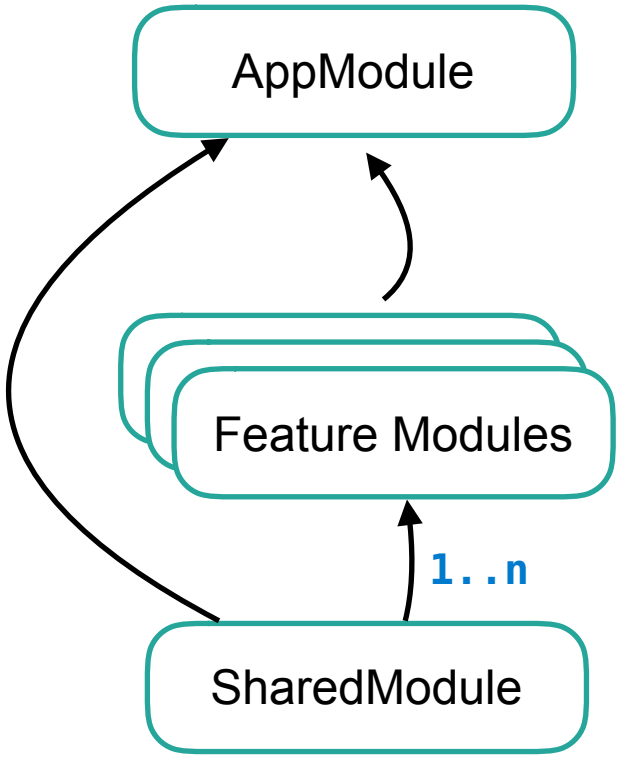
Global components

What if multiple modules declare the same component*?



Global components

Always declare once, make available through import/export



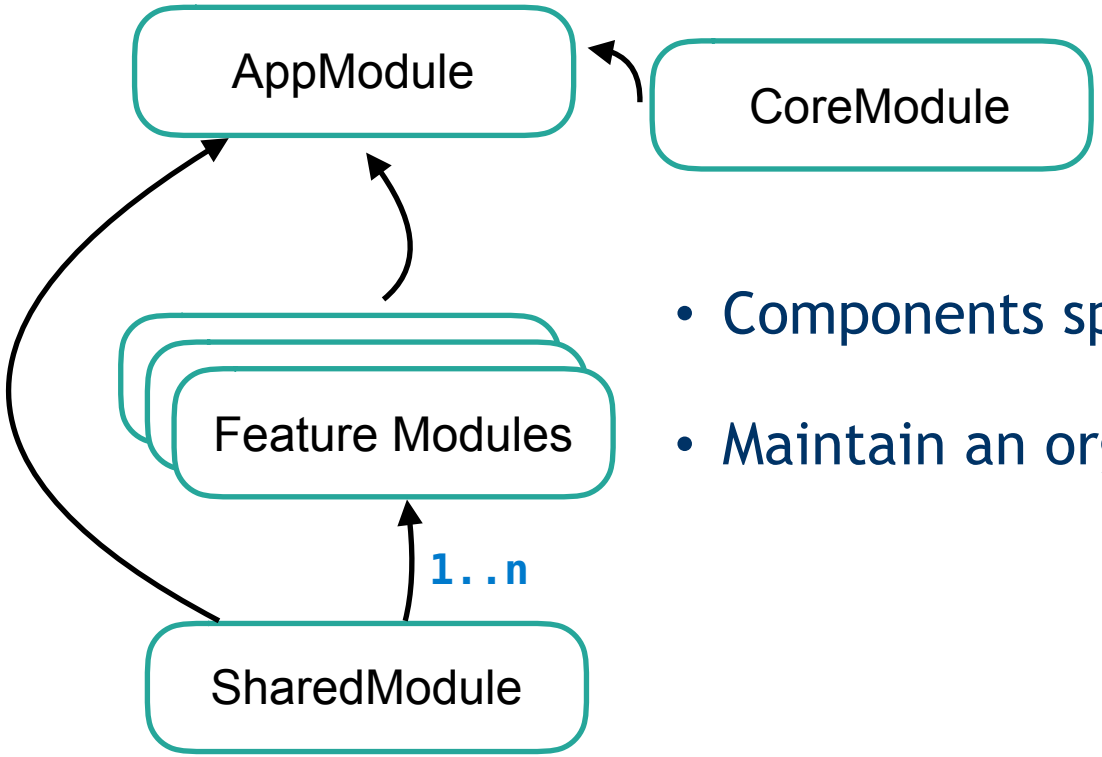

```
imports: [SharedModule]
```

```
imports: [SharedModule]
```

```
exports: [GlobalComponent]
```

CoreModule

How do I prevent the AppModule from overflowing?



- Components specifically for the AppModule
- Maintain an organized folder structure

Why feature modules?

```
declarations: [
    AppComponent,
    LoginComponent,
    MamComponent,
    GuideComponent,
    FeedsComponent,
    HighlightPipe,
    TranslatePipe,
    FormatDatePipe,
    OrderByPipe,
    ByteFormatPipe,
```

That's why feature modules

```
declarations: [
  AppComponent,
  LoginComponent,
  MamComponent,
  GuideComponent,
  FeedsComponent,
  HighlightPipe,
  TranslatePipe,
  FormatDatePipe,
  OrderByPipe,
  ByteFormatPipe,
  Programfilter,
  Seriefilter,
  BlockfilterPipe,
  SanitizeHtml,
  RouteComponent,
  FeedlistComponent,
  FeeddetailComponent,
  FeedmultipleComponent,
  ModselectorComponent,
  MailComponent,
  ImagePopupComponent,
  Datepicker,
  SpinnerComponent,
  ChannelselectorComponent,
  GuideProgramFilterComponent,
  DisplayfieldComponent,
  FlexfieldComponent,
  GuideListComponent,
  DetailComponent,
  GuideMultipleComponent,
  GuideTemplatesComponent,
  GuideSerieFilterComponent,
  GuideSeriesComponent,
  GuideBlockFilterComponent,
  GuideBlocksComponent,
  GuideDetailComponent,
  GuideSeriedetailComponent,
  GuideEditDetailComponent,
  GuideEditSeriedetailComponent,
  GuideBlockdetailComponent,
  ErrorComponent,
  ConfirmComponent,
  MamDetailComponent,
  MamListComponent,
  MamPickerComponent,
  MamselectorSmallComponent,
  MamselectorLargeComponent,
  MamLineComponent,
  ComponentOutlet,
  StoryListComponent,
  StoryfilterPipe,
  FeedlineComponent,
  PlannerComponent,
  PlannerDetailComponent,
  PlannerListComponent,
  PlannerMultipleComponent,
  DatenavigatorComponent,
  MamPreviewComponent,
  OfficeselectorComponent,
  PlannerSearchComponent,
  PlannerFilterComponent,
  PlannerfilterPipe,
  ResizeComponent,
  SearchboxComponent,
  StoryContentComponent,
  ThumbnailComponent,
  RundownListComponent,
  RundownSelectorComponent,
  RundownComponent,
  RundownDetailComponent,
  RundownMultipleComponent,
  TagmanagerComponent,
  LocationpickerComponent,
  LockingindicatorComponent,
  Autosize,
  MamMultipleComponent,
  MamJobsComponent,
  ImageCropperComponent
],
providers: [
  InitService,
  LanguageService,
  Config,
  ApplicationService,
  SocketService,
  ProfileService,
  ExportService,
  MailService,
  FeedService,
  ErrorService,
```

ilionx
Your partner in digital business

Recap

- NgModule
- Divide the application logically
- Feature modules
- 'types' of modules

Assignment

- Declare the components with a module
- Give the module a good name
- Make a tree of the modules

Components

- Responsible for a 'piece of the page'
- Define template and behaviour
- Types
  - Business / Smart / Container
  - Presentation / Dumb

Tasks Component

- **Availability** through a **selector**
- Define **Behaviour** in a class
- Combine with **Template**
- Add **styles**

@Component

```
@Component({
  selector: 'ps-contact'
  template: `<h1 class="title">Pretty Contact</h1>`,
  styles: [ `h1.titel { color: #e6e6e6; }` ]
})
export class ContactComponent {

    ...
}
```

contact.component.ts

ilionx
Your partner in digital business

@Component

```
@Component({
  selector: 'ps-contact'
  templateUrl: './contact.component.html',
  styleUrls: [ './contact.component.css' ]
})
export class ContactComponent {

    ...
}
```

Selector

Use the component in HTML

```
<div>
    <ps-contact></ps-contact>
</div>
```

Life cycle methods

Called by Angular during specific phases of the application

```
@Component({
  selector: 'ps-contact'
  templateUrl: './contact.component.html',
  styleUrls: [ './contact.component.css' ]
})
export class ContactComponent implements OnInit {

  ngOnInit() {
     // code for initialisation
  }

}
```

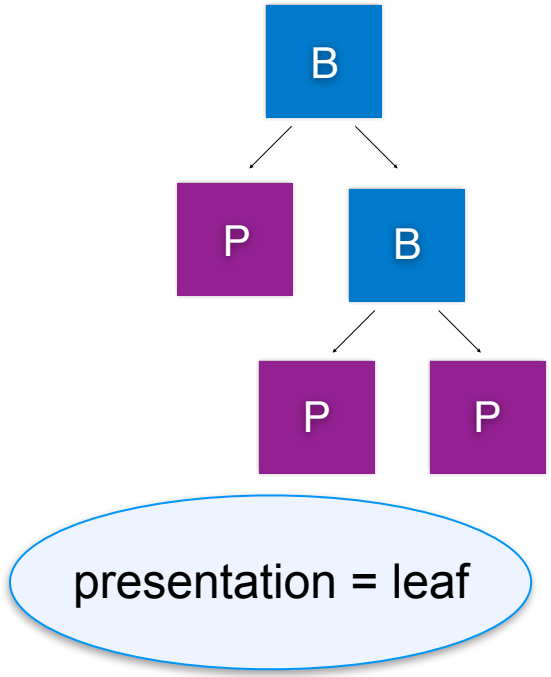
Life cycle methods

- ngOnChanges
- ngOnInit
- ngDoCheck
    - ngAfterContentInit
    - ngAfterContentChecked
    - ngAfterViewInit
    - ngAfterViewChecked
- ngOnDestroy

Presentation and business components

Presentation

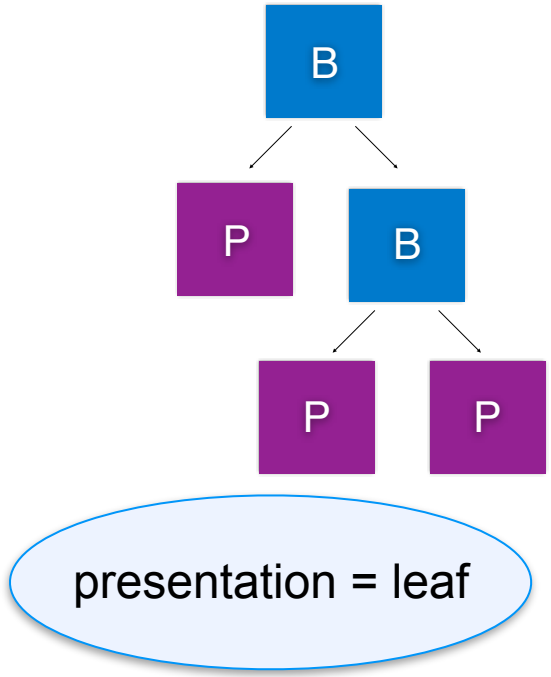
- Present data
- Catch user interaction
- Usually no global service



Presentation and business components

Business

- managing screen logic
- hold services
- provide data
- catch component events



Recap: modules and componenten

- @NgModule
  - Divide the application
  - App, Feature, Shared, Core
- @Component
  - Divide the screen
  - Life cycle methods
  - Smart, dumb

Demo: modules and components