



ANGULAR  
**ARCHITECTS**  
INSIDE KNOWLEDGE

# Professional NgRx

## 2 - Architecture

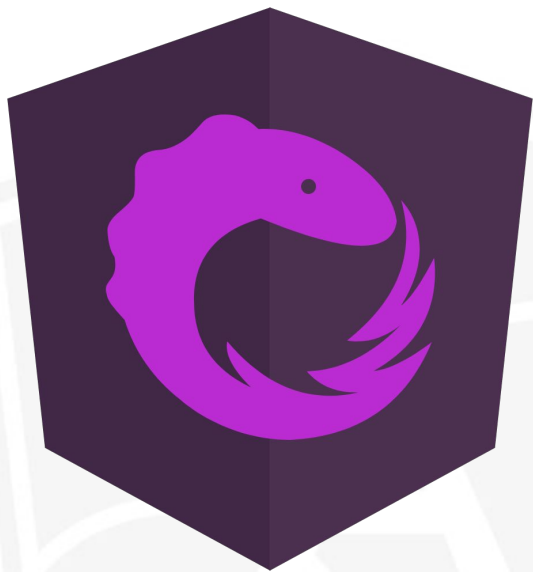


# Agenda

1. Modules & dependencies
2. API
  - a. Facade or repository
  - b. Inter-module dependencies
3. StateModel vs. ViewModel



# Modules and dependencies

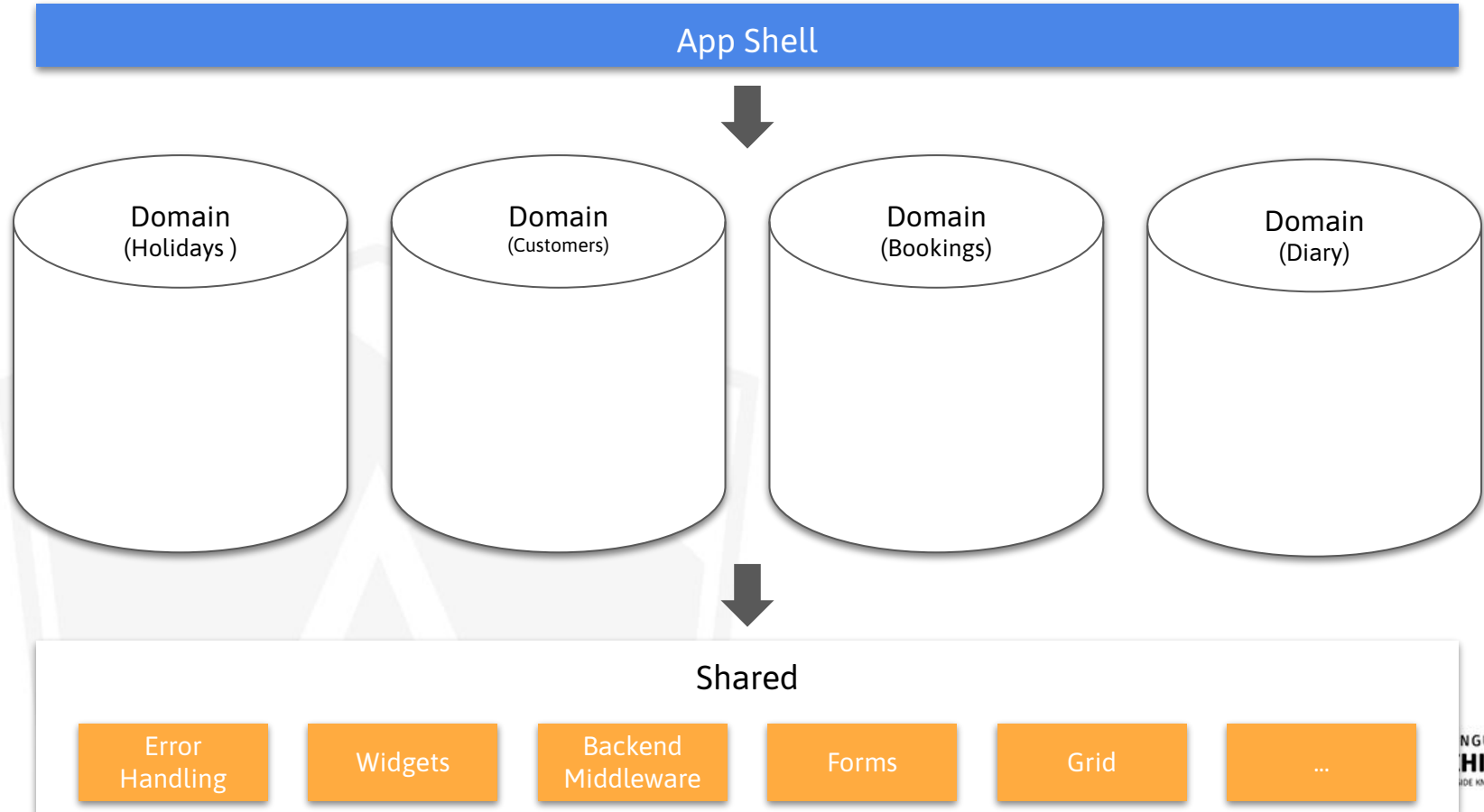


# Requirements

- Encapsulation
  - Module-internal changes shouldn't affect the rest
  - Exposed features are clearly defined
- Dependency Rules
  - Module hierarchy
  - Reduce/localise the risk of changes
- Automated verification
  - Nx linting rules



# Tier 1 - Main Modules



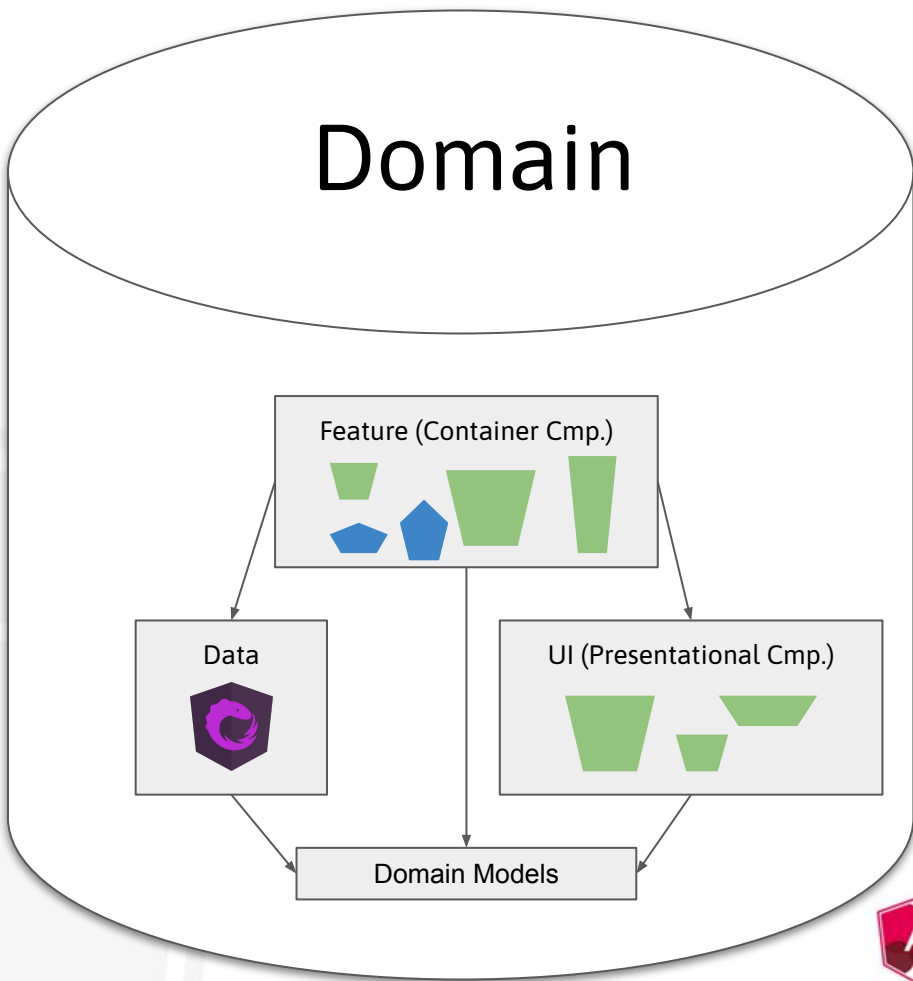
# Tier 1 - Main Modules

- Independent parts
- Maintainability
- Team Scalability
  - Feature teams
  - Library teams
- Architecture Scaling
- Better protection from bugs

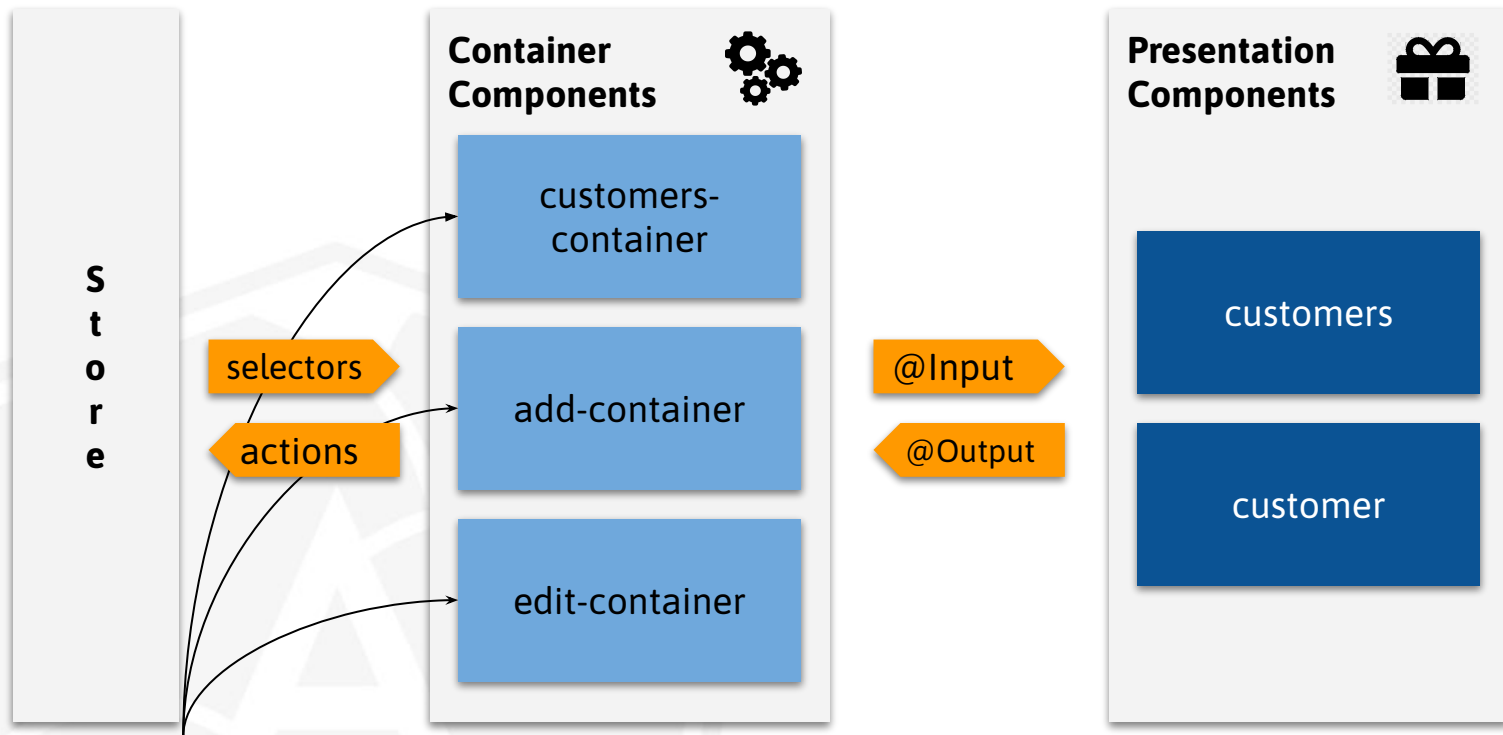


ANGULAR  
**ARCHITECTS**  
INSIDE KNOWLEDGE

# Tier 2 Sub Modules



# Container & Presentational Components



## Routing Configuration

```
/customer  
/customer/add  
/customer/edit/{id}
```

- Least Possible Template
- Reading from Store
- Dispatching to Store

- Least Possible Typescript
- No Dependency Injection, only @Input and @Output
- No Observables!

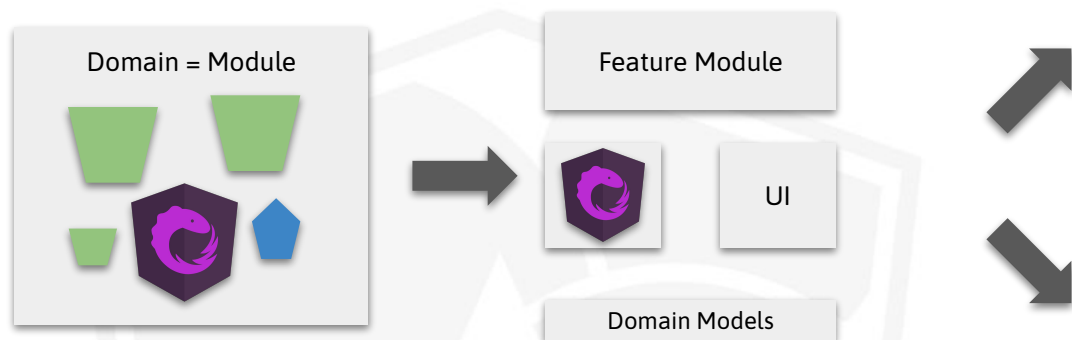


Restricted NgRx access makes  
only sense with  
container/presentational  
components

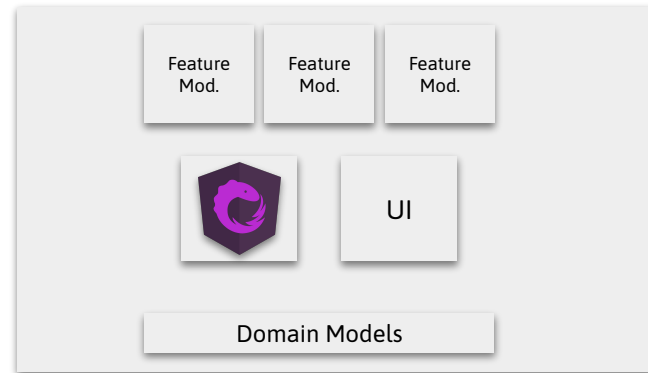


ANGULAR  
**ARCHITECTS**  
INSIDE KNOWLEDGE

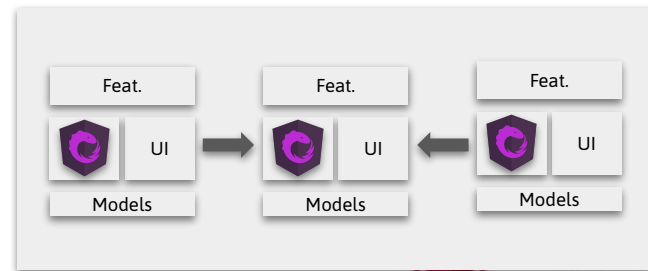
# Tier 2 Evolution Process



Multiple feature operating on same state



Sub-Domains with core-domain



Lab Time

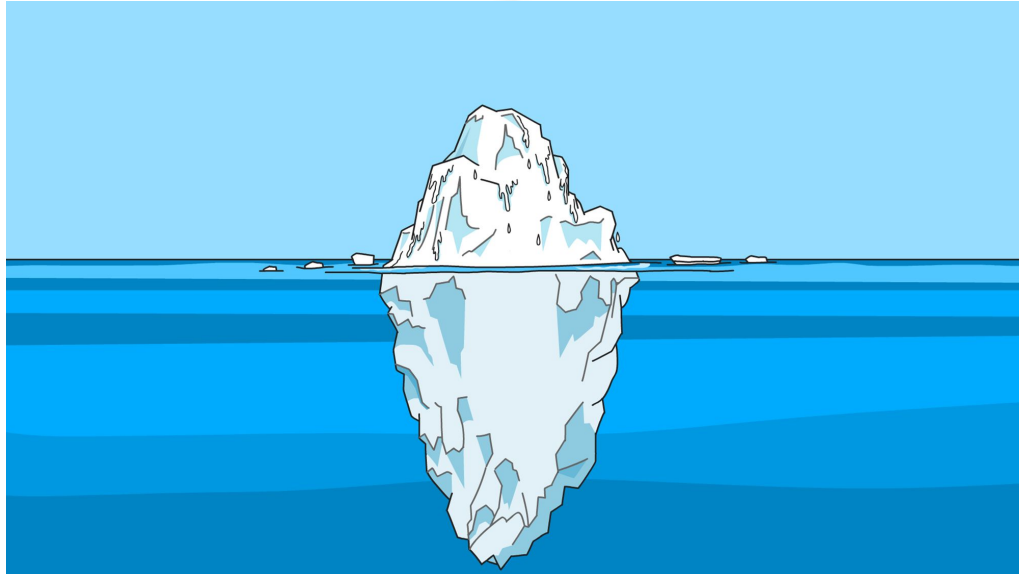


# Dependency Types

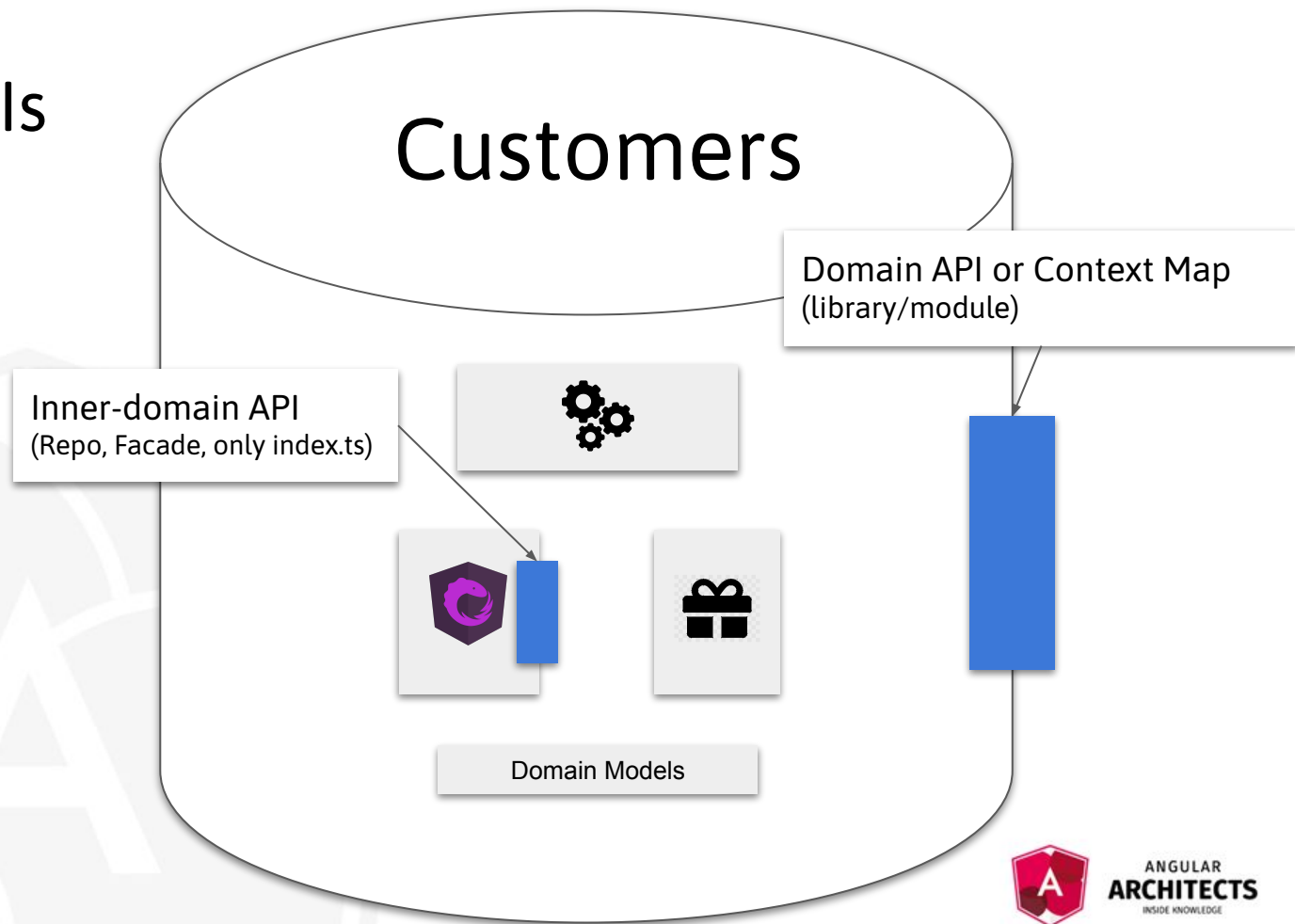
- Module Type per Domain
  - Generic
  - Hierarchy after type
- API
  - Individual
  - Above feature module(s)
  - Mostly open to feature or data modules
- Shared
  - Individual
  - Access to generic module type



# API



## 2 Types of APIs



# Trade-Off

## **Architecture:** Generic API

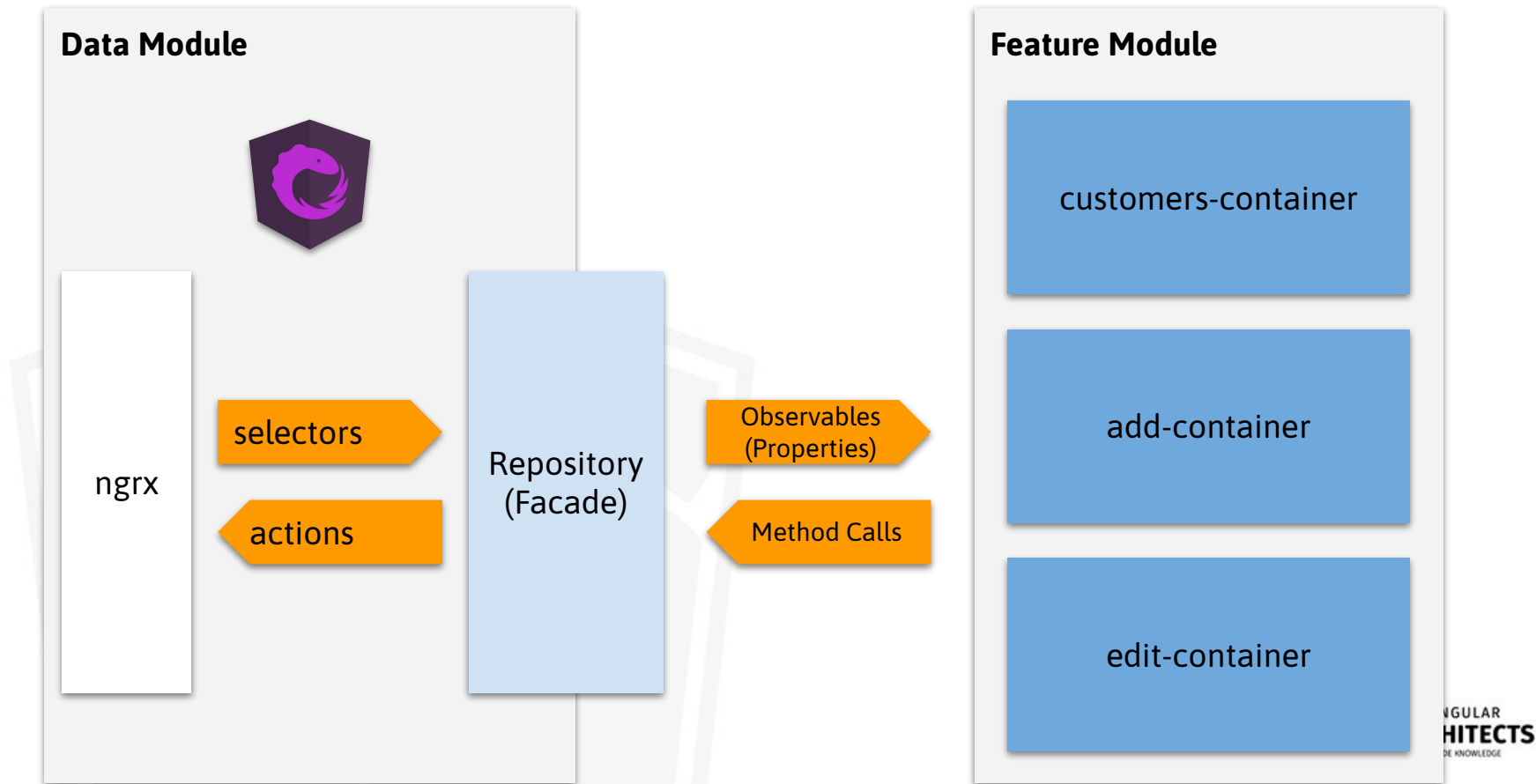
- Hide selectors
- Looser coupling because NgRx is completely hidden
- More overhead and work for feature modules
  - combineLatest
- Cannot combine selectors of multiple features

## **Performance:** Combining selectors

- Selectors are more performant
- Tight coupling to NgRx
- Simpler



# NgRx repository/facade



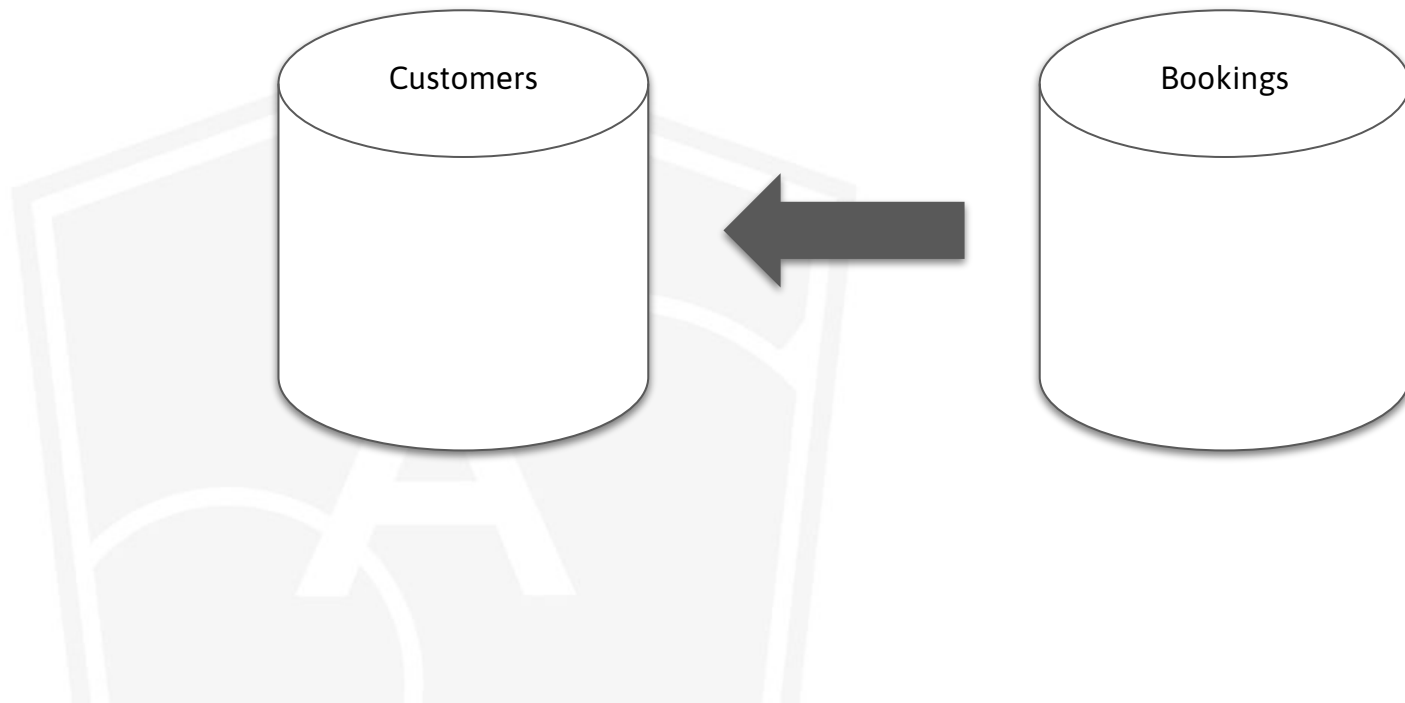


# Repository/Facade/API

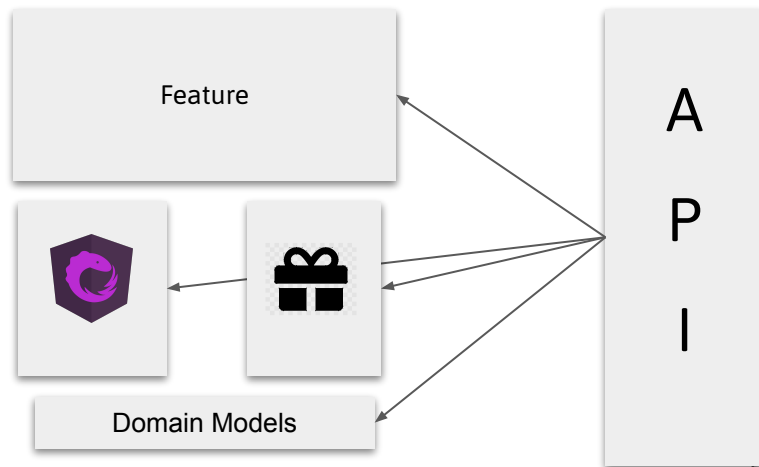
1. Single element,
2. exposing module's functionality,
3. in a "user-friendly" way



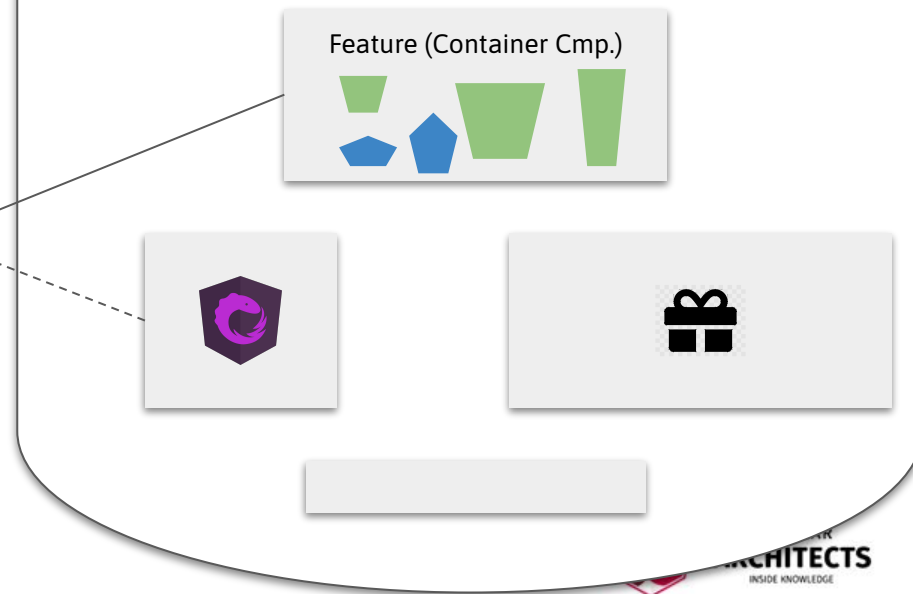
# Module Type: API



# Customers



# Bookings



# API-related rules

```
{  
  "sourceTag": "domain:customers:api",  
  "onlyDependOnLibsWithTags": ["domain:customers"]  
}
```



ANGULAR  
**ARCHITECTS**  
INSIDE KNOWLEDGE

# API-related rules

```
{  
  "sourceTag": "domain:customers:api",  
  "onlyDependOnLibsWithTags": ["domain:customers"]  
}, {  
  "sourceTag": "type:api",  
  "onlyDependOnLibWithTags": ["type:feature", "type:data", "type:ui", "type:model"]  
}
```



# API-related rules

```
{  
  "sourceTag": "domain:customers:api",  
  "onlyDependOnLibsWithTags": ["domain:customers"]  
}, {  
  "sourceTag": "type:api",  
  "onlyDependOnLibWithTags": ["type:feature", "type:data", "type:ui", "type:model"]  
}, {  
  "sourceTag": "domain:bookings",  
  "onlyDependOnLibsWithTags": ["domain:bookings", "domain:customers:api", "domain:shared"]  
}
```



# API-related rules

```
{
  "sourceTag": "domain:customers:api",
  "onlyDependOnLibsWithTags": ["domain:customers"]
}, {
  "sourceTag": "type:api",
  "onlyDependOnLibWithTags": ["type:feature", "type:data", "type:ui", "type:model"]
}, {
  "sourceTag": "domain:bookings",
  "onlyDependOnLibsWithTags": ["domain:bookings", "domain:customers:api", "domain:shared"]
}, {
  "sourceTag": "type:feature",
  "onlyDependOnLibWithTags": [
    "type:api"
    // ...
  ]
}
```



# StateModel vs. ViewModel





# Direct Consequence

- Applies to Entity-based states
- Hide NgRx and still use selectors from other features
- Dispatching actions from other feature states?
- How should NgRx know about UI?



# Direct Consequence

- Applies to Entity-based states
- Hide NgRx and still use selectors from other features
- Dispatching actions from other feature states?
- How should NgRx know about UI?





# Unforgettable Holidays

Holidays

Customers

Bookings

## Bookings for Bellitissa, Latitia

Holiday	Booking Date	Status
1	pending	A little bit unsure about the holiday. Let's see
2	cancelled	Seemed to be a little bit stressed out



ANGULAR  
ARCHITECTS  
INSIDE KNOWLEDGE

# Bookings

Customers

A  
P  
I

Container  
Component

combineLatest

@Input

Presentational  
Component



ANGULAR  
ARCHITECTS  
INSIDE KNOWLEDGE

# Summary

- Use Nx's linting rules
  - Encapsulation
  - Dependency rules
- Use repository for simplification
- Use an API module for domain dependencies
- combineLatest for complex ViewModels

