ANGULAR WORKSHOP

COMPONENT INTERACTION
SERVICES & DEPENDENCY INJECTION
VIEW ENCAPSULATION

Naveen Pete

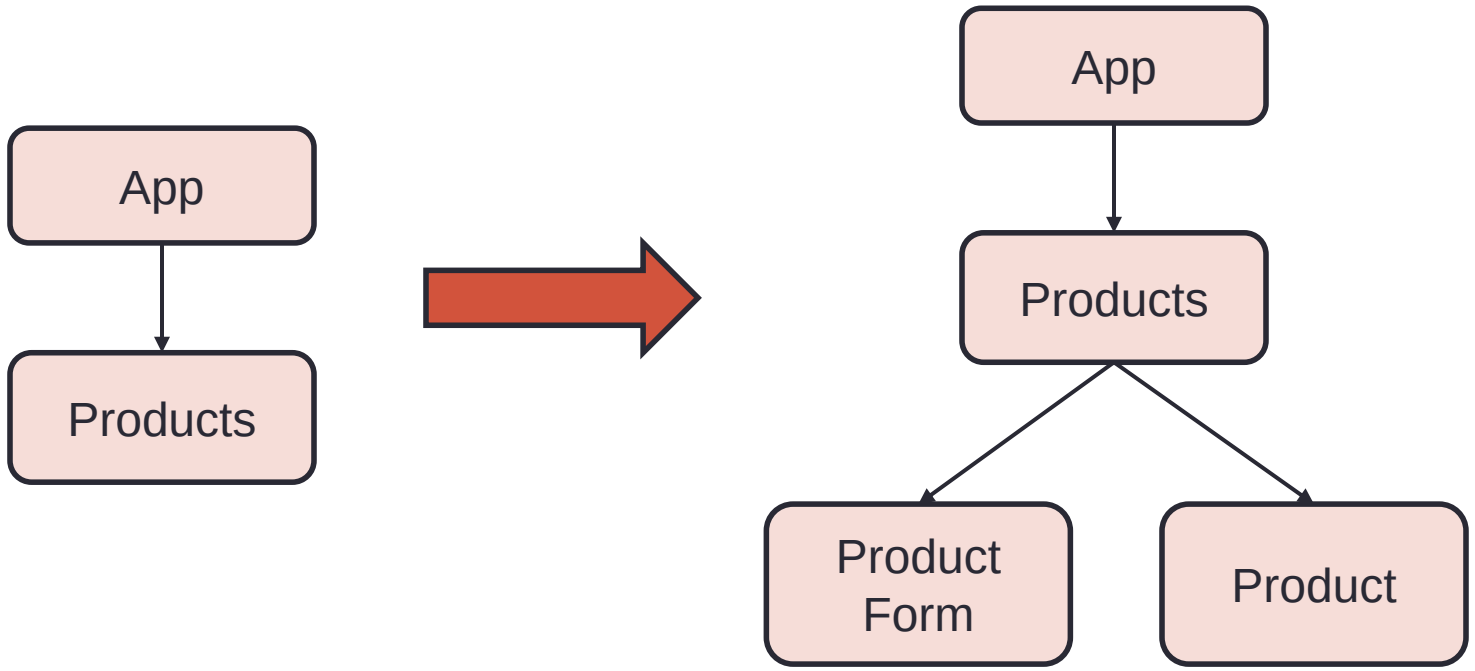
Friday, June 30, 2017

Agenda

- Component Interaction
  - Parent to Child Interaction
  - Child to Parent Interaction
- Services
  - Need for a Service
  - What is a Service?
  - Understanding DI and Angular Injector
  - Creating and Using a Service
  - Cross Component Communication using a Service
- View Encapsulation
- Q & A

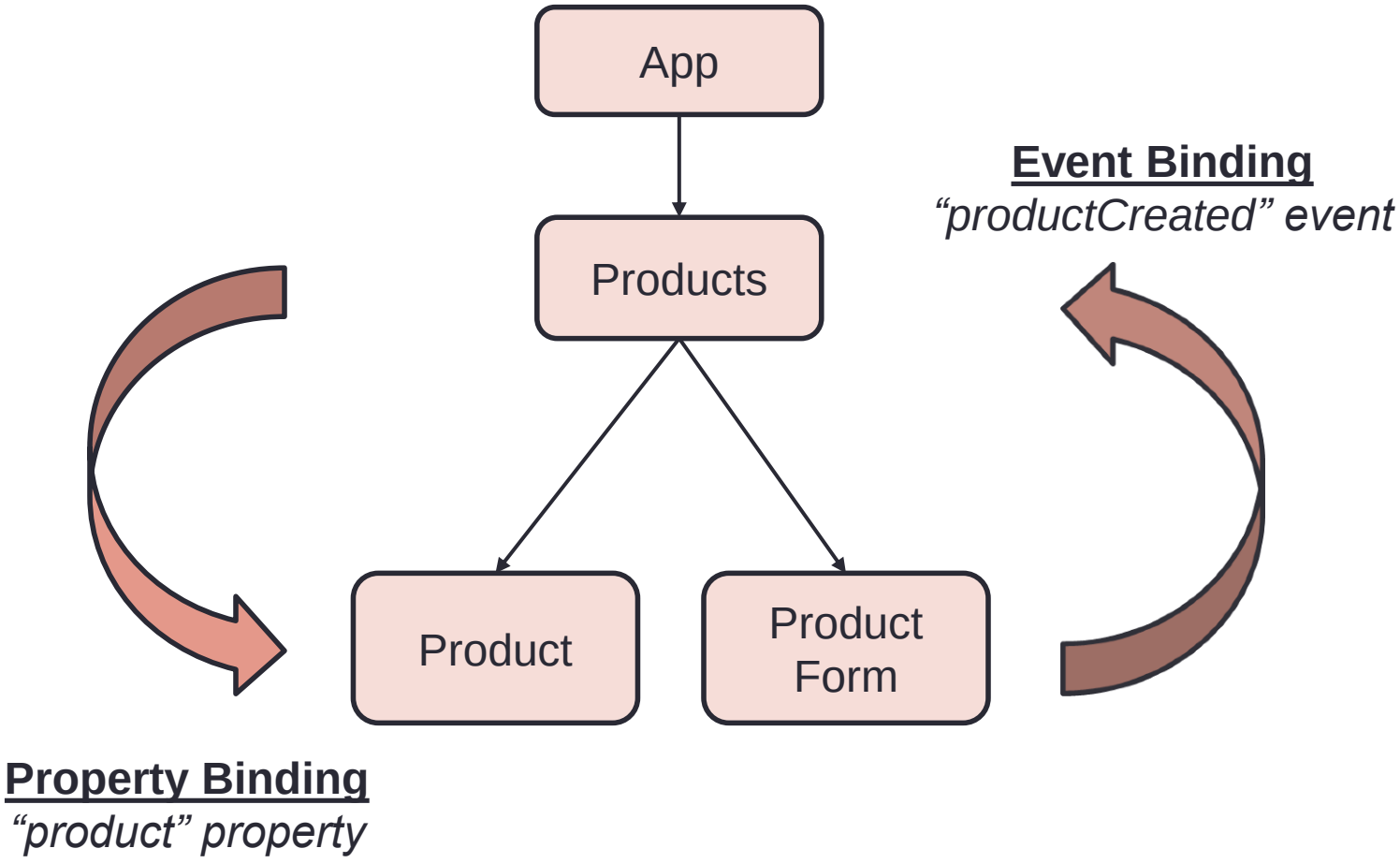
Component Interaction

- Splitting app into multiple components



Component Interaction

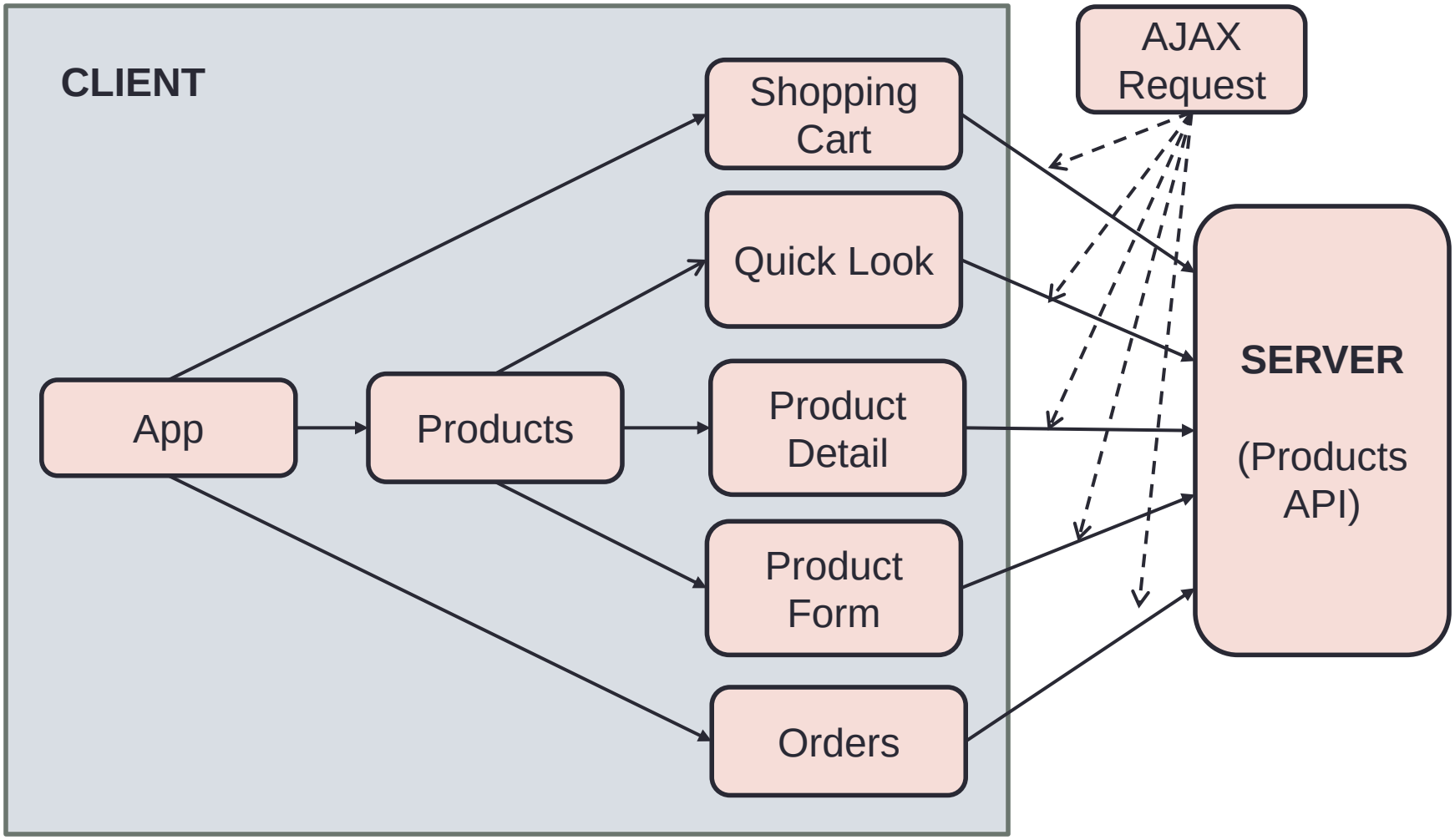
- Overview



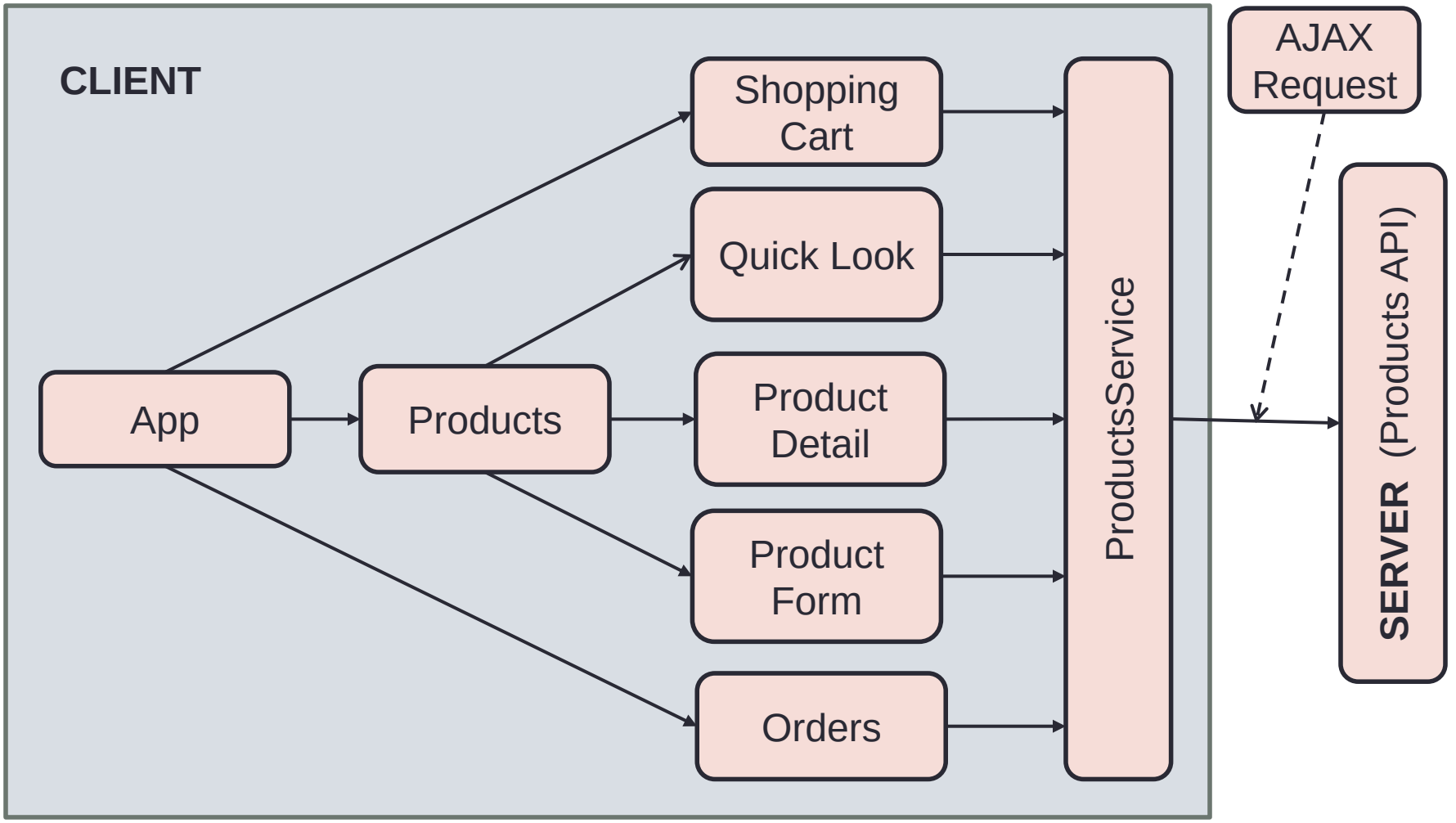
Component Interaction

- Binding to Custom Properties
  - Pass data from parent to child component
    - @Input() decorator
- Binding to Custom Events
  - Emitting event from child component
    - @Output() decorator
    - EventEmitter<T>
    - eventEmitterObj.emit()

Services


CLIENT
App
Products
Shopping Cart
Quick Look
Product Detail
Product Form
Orders
AJAX Request
SERVER
(Products API)

Services


CLIENT
App
Products
Shopping Cart
Quick Look
Product Detail
Product Form
Orders
ProductsService
AJAX Request
SERVER (Products API)

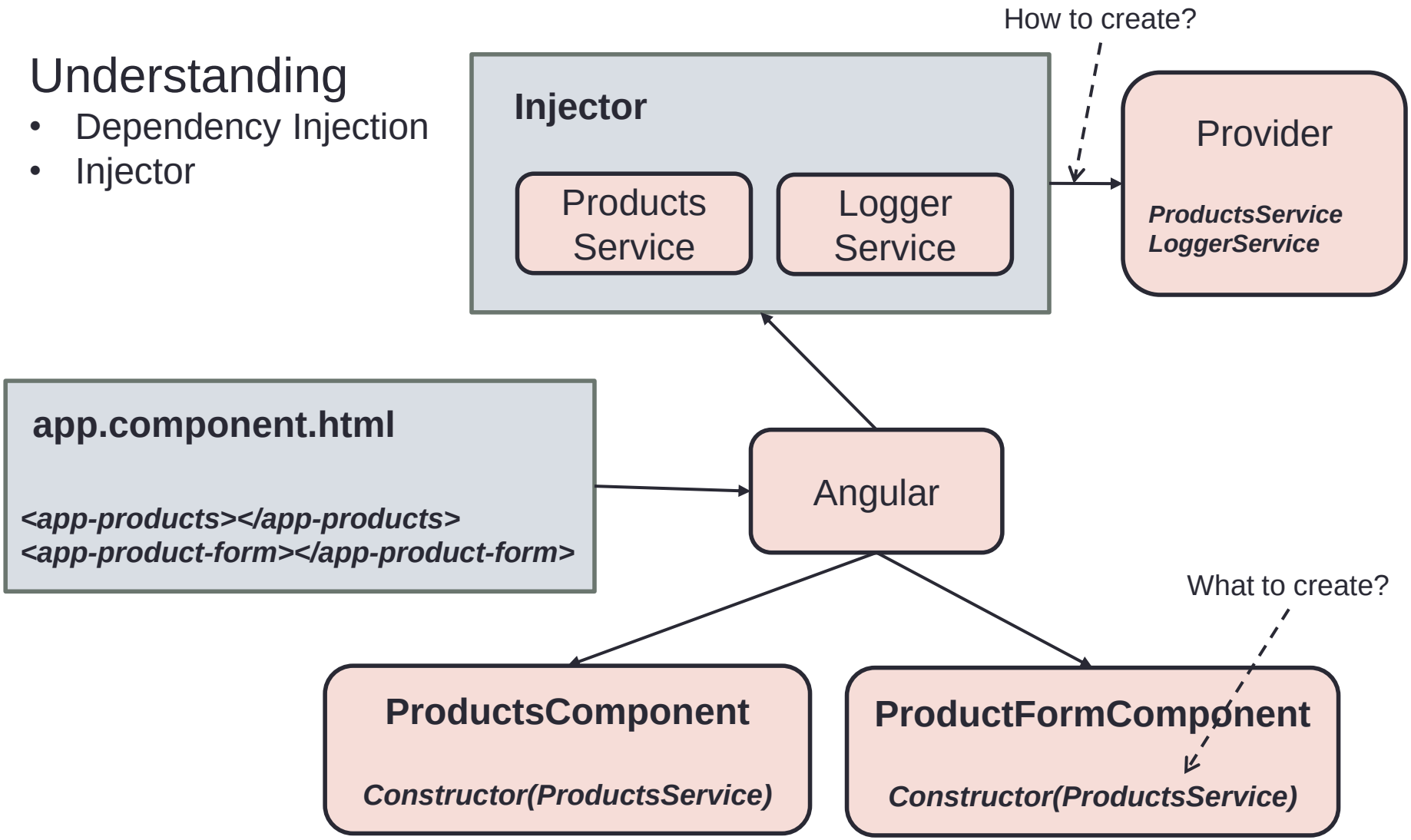
Services

- A class with a narrow, well-defined purpose
  - For e.g.
    - Logging service
    - Data service
    - Tax calculator
    - App configuration
    - Message bus
- Acts as a central repository/business unit
- Creating a service
- Injecting a service into a component
  - Constructor
  - Providers
    - Component level
    - Module level
- Injecting a service into another service
  - @Injectable()

Services

Understanding

- Dependency Injection
- Injector



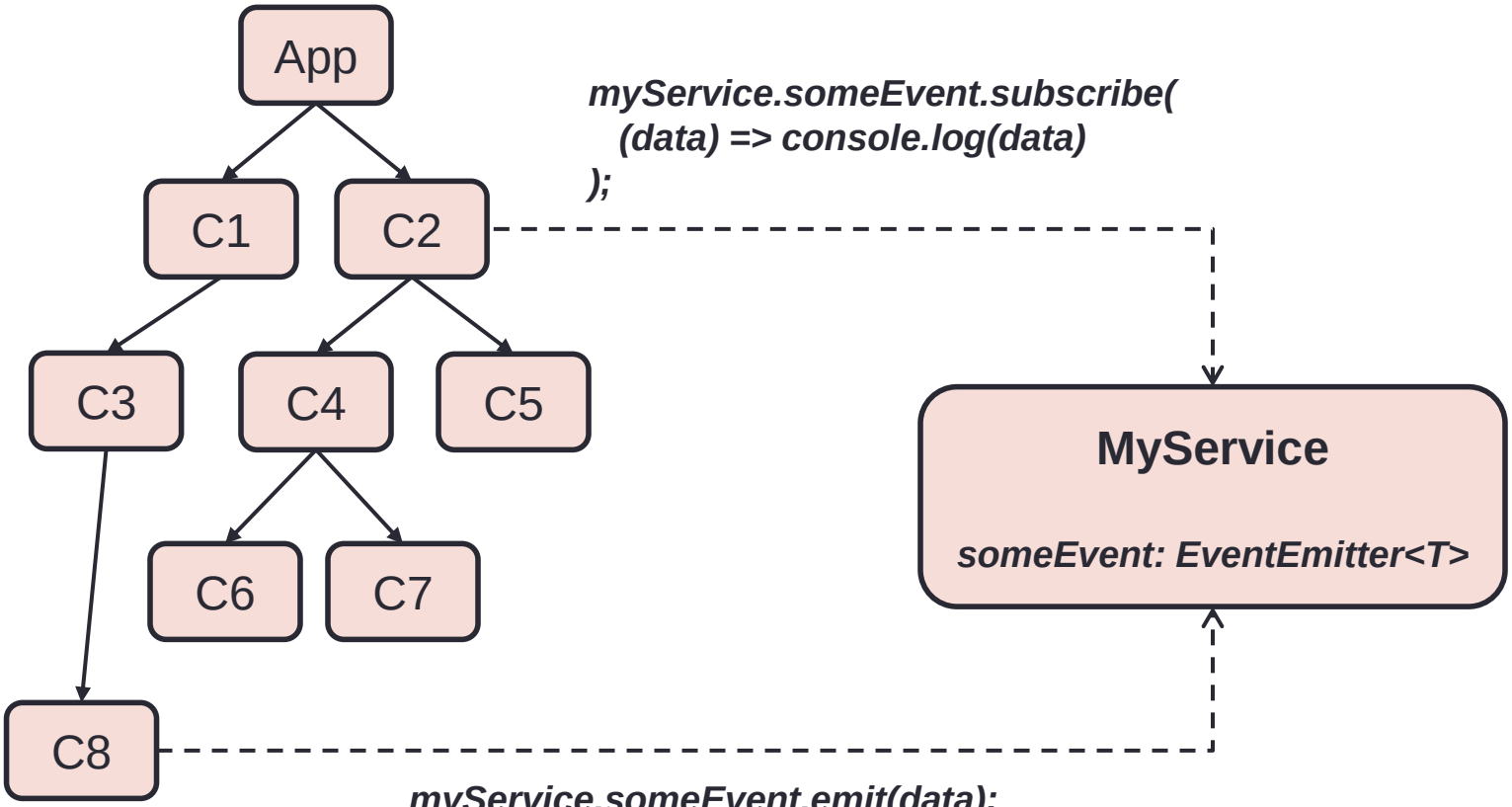
Services

- Controlling the creation of instances of a Service

AppModule	Same instance of Service is available ***Application wide***
AppComponent	Same instance of Service is available for ***all Components*** (but not for other services)
Any other Component	Same instance of Service is available for ***the Component*** and ***all its child Components***

Services

- Cross component communication using a service
  - In the service, expose an event object of type ***EventEmitter***
  - From the source component, invoke ***emit()*** method, pass necessary data as an argument
  - From the destination component subscribe to the service's event object using ***subscribe()*** method, pass callback function as an argument



View Encapsulation

- Understanding View Encapsulation
- @Component()
  - encapsulation: ViewEncapsulation.None
- ViewEncapsulation
  - Emulated – default
  - Native
  - None

Q & A

- Thank you!