

INTRODUCTION TO ANGULAR

Naveen Pete

Saturday, February 24, 2018

Agenda

- What is Angular?
- Angular Versions
- Why Angular?
- Where does Angular fit?
- Setting up Angular
- TypeScript
- Angular Building Blocks
 - Module
 - Component
 - Decorator
 - Data Binding
 - Directive
 - Pipe
 - Service
 - Router
- Server Communication
- Q & A

What is Angular?

- Developed in 2009 by Misko Hevery
- Currently maintained by Google
- Framework for building front-end JavaScript applications
- Angular apps
 - Can run on desktop and mobile devices
 - Are generally SPAs
- Open-source, TypeScript-based framework
- 'A' of MEAN stack

Angular Versions

- AngularJS (v1.x)
 - Aims to simplify the development and testing of web apps
 - Worked on the concept of scope and controllers
 - Initial release, v0.9.0 – Oct 2010
 - Latest release, v1.6.9 – Feb 2018
- Angular 2
 - Added component as a key building block
 - Complete re-write of AngularJS, no backward compatibility
 - Released in Sep 2016

Angular Versions

- Angular 4
 - Apps are smaller & faster
 - AOT compilation, Angular Universal - SSR
 - Backward compatible with Angular 2
 - Released in Mar 2017
- Angular 5
 - Smaller, faster and easier to use
 - Build optimizer, compiler improvements
 - New HttpClient, pipes, router lifecycle events
 - Released in Nov 2017

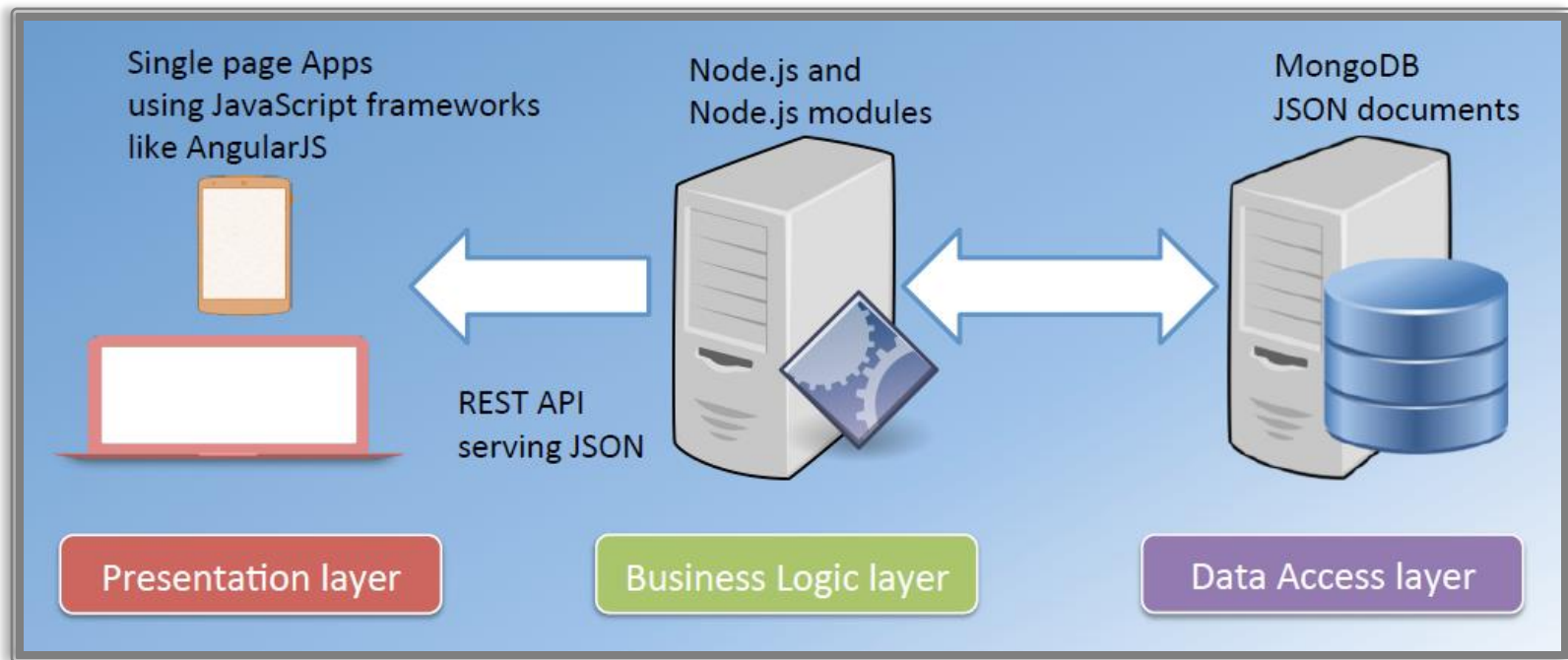
Why Angular?

- Single Page Apps (SPA)
 - Better user experience
 - Reduced full page reloads
 - Better overall performance
 - Less network bandwidth
- Proven software patterns and practices
 - Model View Controller (MVC)
 - Model View ViewModel (MVVM)
 - Dependency Injection (DI)
- Declarative programming
 - Better readability, concise code
 - Better developer productivity
 - Faster development

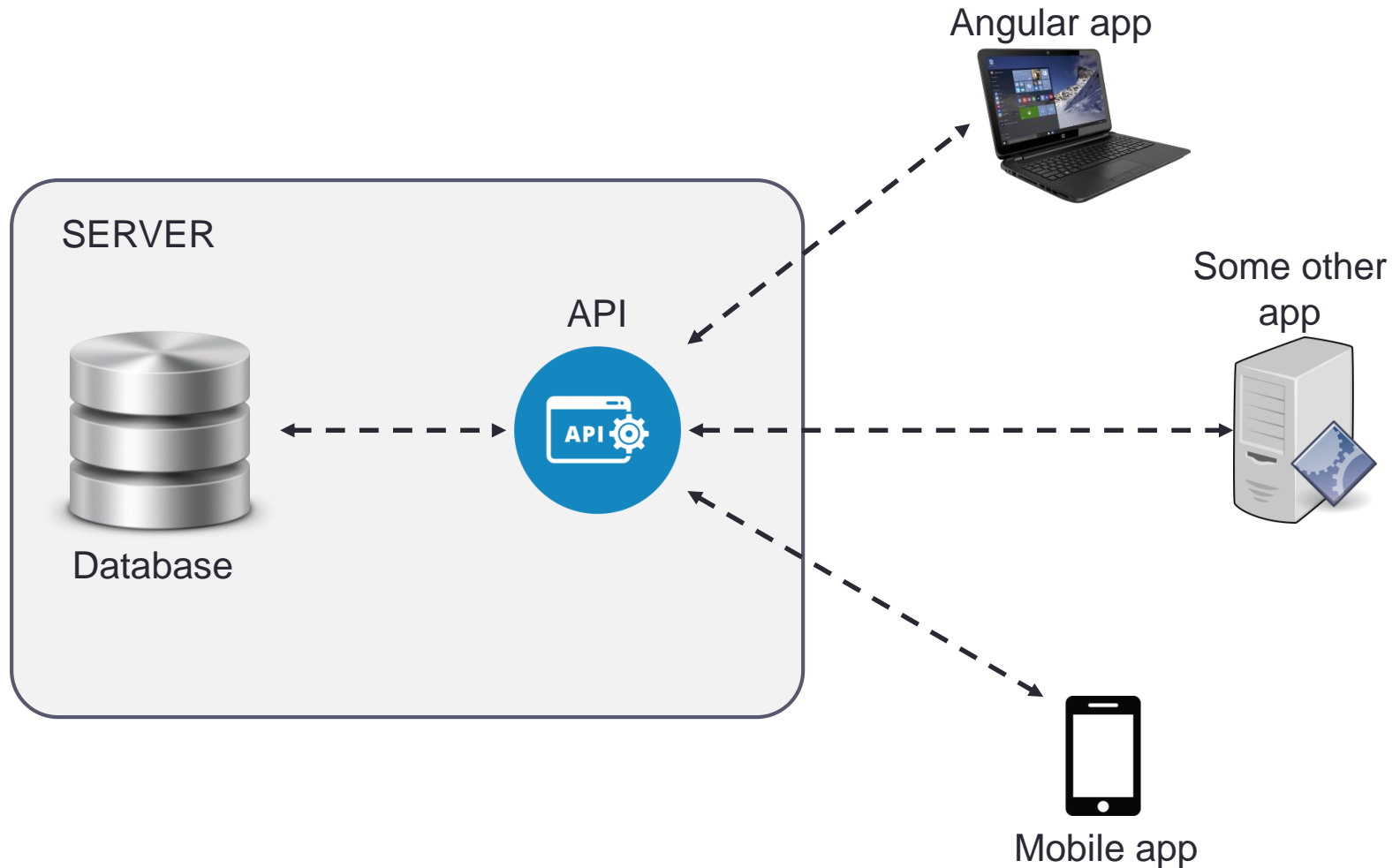
Why Angular?

- Component based
 - Reusable
- Structures app code
 - Modular, Maintainable, Scalable
- Cross platform, mobile support
 - Target multiple browsers, platforms & devices
- Decouples DOM manipulation from app logic
 - Testable, TDD
- Move app code forward in the stack
 - Reduces server load, reduces cost
 - Crowd sourcing of computational power

Where does Angular fit?



Where does Angular fit?



Setting up Angular

- Angular CLI
 - Toolset that makes creating, managing and building Angular apps very simple
 - Great tool for big Angular projects
 - Website: <https://cli.angular.io>
 - Wiki: <https://github.com/angular/angular-cli/wiki>
- Requires Node.js
 - <https://nodejs.org>

```
> npm install -g @angular/cli  
> ng new my-first-app  
> cd my-first-app  
> ng serve
```

Setting up Angular

- Angular CLI commands

```
> ng new <project-name>
```

```
> ng serve
```

```
> ng build
```

```
> ng test
```

```
> ng generate <type> <name>
```

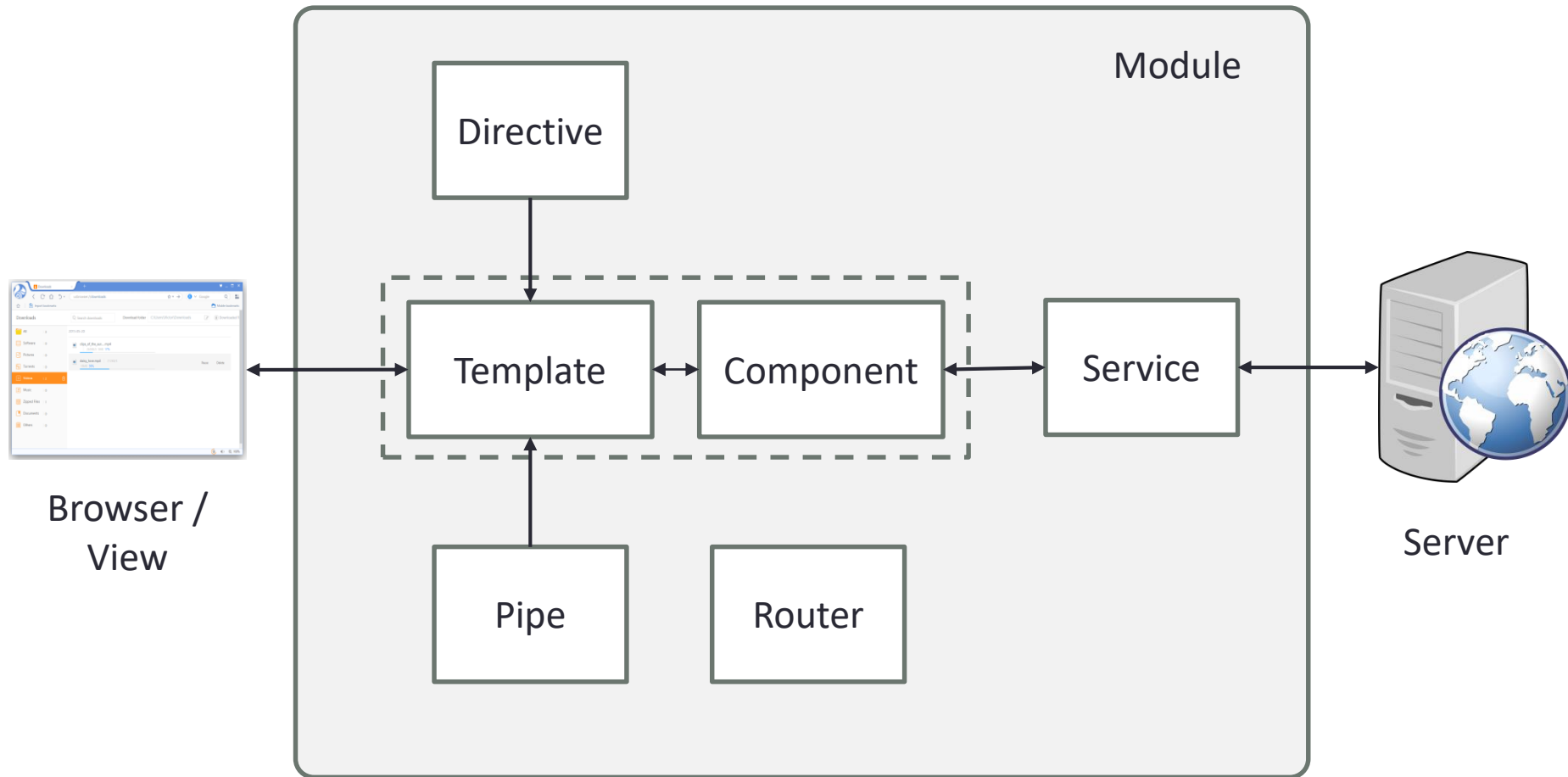
where <type> can be one any one of:

```
class | component | directive | interface |  
module | pipe | service | enum | guard
```

TypeScript

- Superset of JavaScript
 - Any valid JavaScript code is also valid TypeScript code
- Developed and maintained by Microsoft
- Primary language for Angular app development
- Does not run in the browser, it is “transpiled” into JS
- Why TypeScript?
 - Static typing
 - Compile-time errors, provides IDE support, easier to debug
 - Object-oriented features
 - Classes, Interfaces, Properties, Generics, Decorators, ...
 - Next gen JS features
 - Modules, Import, Export, ...

Angular Building Blocks



Module

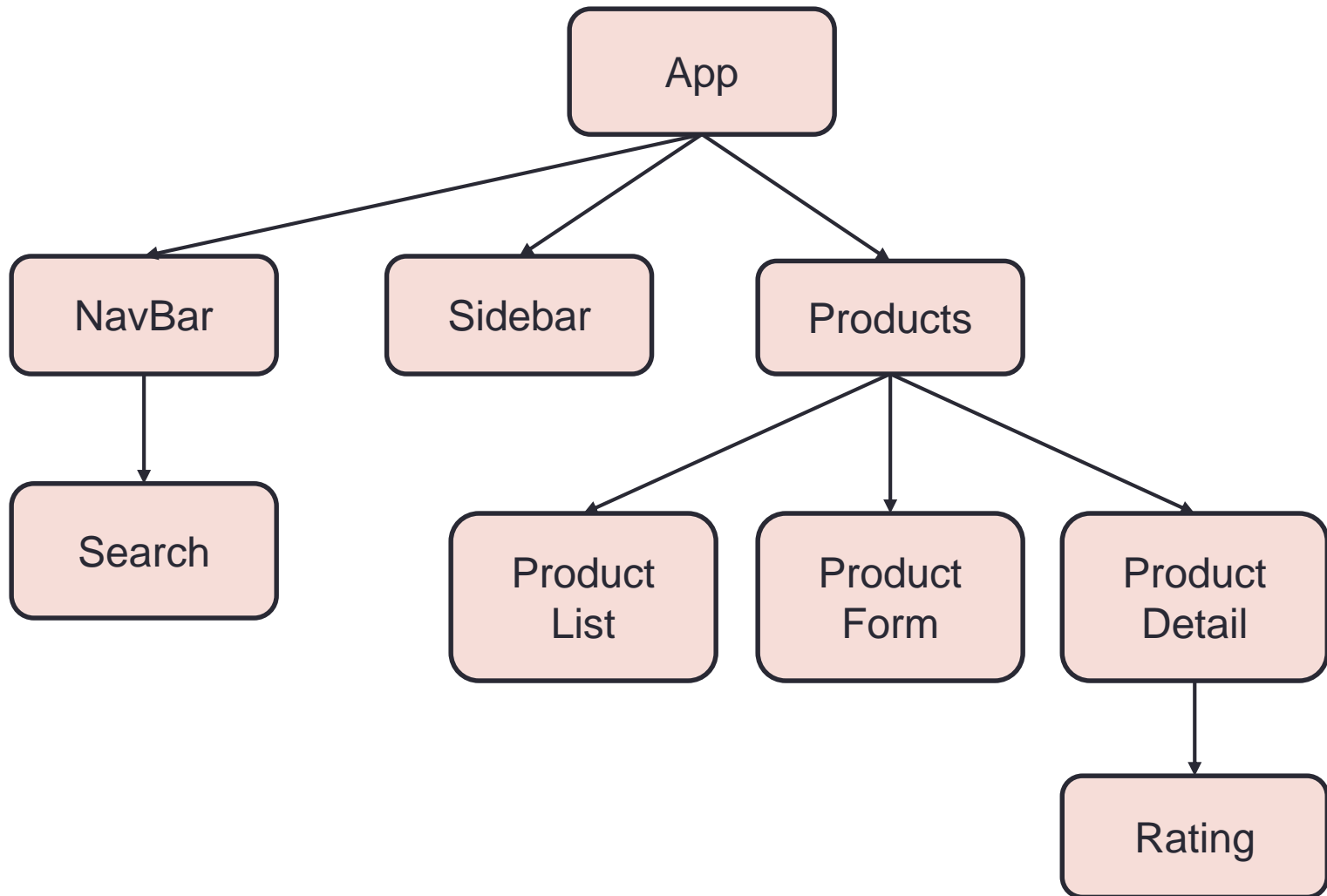
- Organizes an app into cohesive blocks of functionality
- A class marked by @NgModule decorator
- Every Angular app has at least one module class, the **root** module

```
@NgModule({
  imports: [module1, module2, ...],
  declarations: [
    component(s), directive(s), pipe(s), ...
  ],
  providers: [service1, service2, ...],
  bootstrap: [AppComponent]
})
export class AppModule {}
```

Component

- Key feature of Angular apps
- Encapsulate the template, data and the behavior of a view
- Allows you to break a complex web page into smaller, manageable & reusable parts
- A Component has its own
 - Template – HTML markup
 - Style – CSS styles
 - Business logic (data and behavior) – TypeScript code
- App component
 - Root component
 - Other components are added to App component

Component



Decorator

- Extends the behavior of a class / function / property without explicitly modifying it
- Attaches metadata to classes

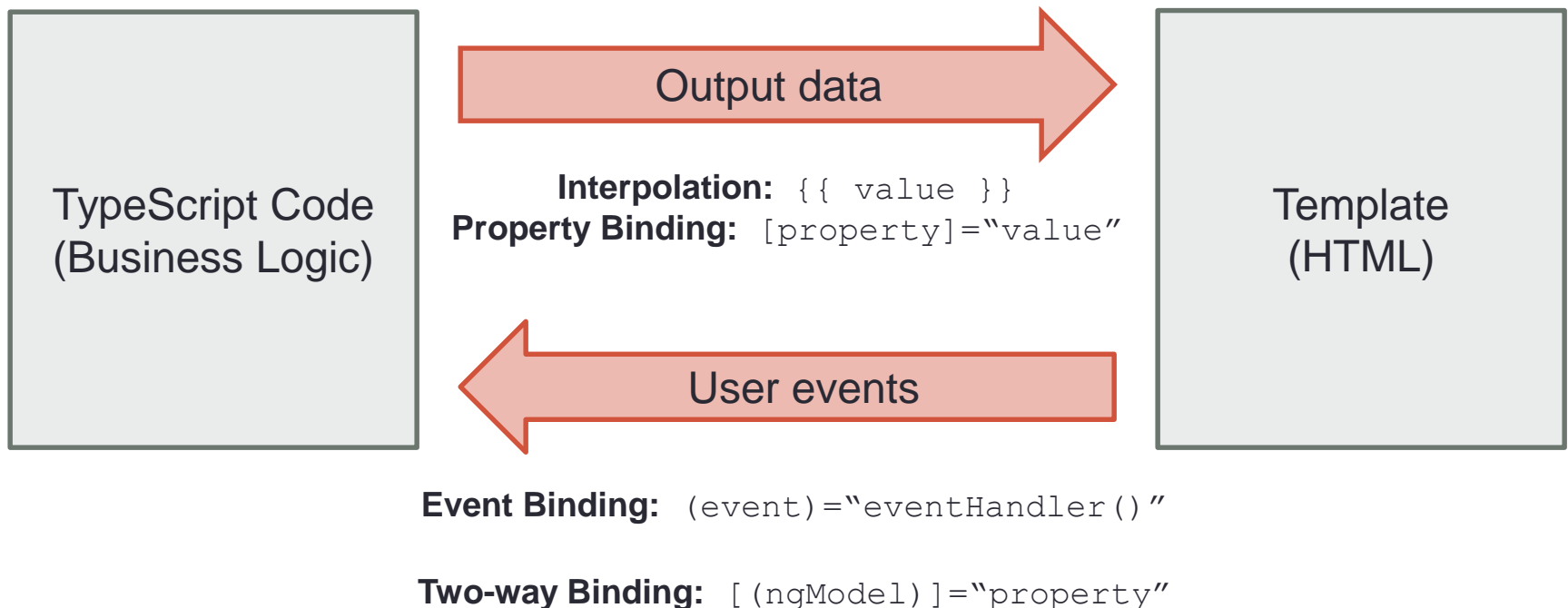
```
import { Component } from '@angular/core';

@Component({
  selector: 'app-products',
  templateUrl: 'products.component.html',
  styleUrls: ['products.component.css']
})
export class ProductsComponent {
  products: [];

  addProduct(product) {
    this.products.push(product)
  }
}
```

Data Binding

- Communication between the TypeScript code and the HTML template



Data Binding

- Interpolation

- `<h1>{{ product.name }}</h1>`

- Property binding

- ``

- Event binding

- `<button (click)="addProduct()">New</button>`

- Two-way data binding

- `<input type="text" name="productName" [(ngModel)]="product.name">`

Directive

- Helps you to extend HTML to support dynamic behavior
- Transforms the DOM according to the instructions given
- Can be built-in or custom
- Built-in directives
 - Structural directives
 - Have a leading *
 - Alter layout by adding, removing, and replacing elements in DOM
 - E.g. *ngIf, *ngFor
 - Attribute directives
 - Look like a normal HTML attribute
 - Modify the behavior of an existing element by setting its display value property and responding to change events
 - E.g. ngStyle, ngClass

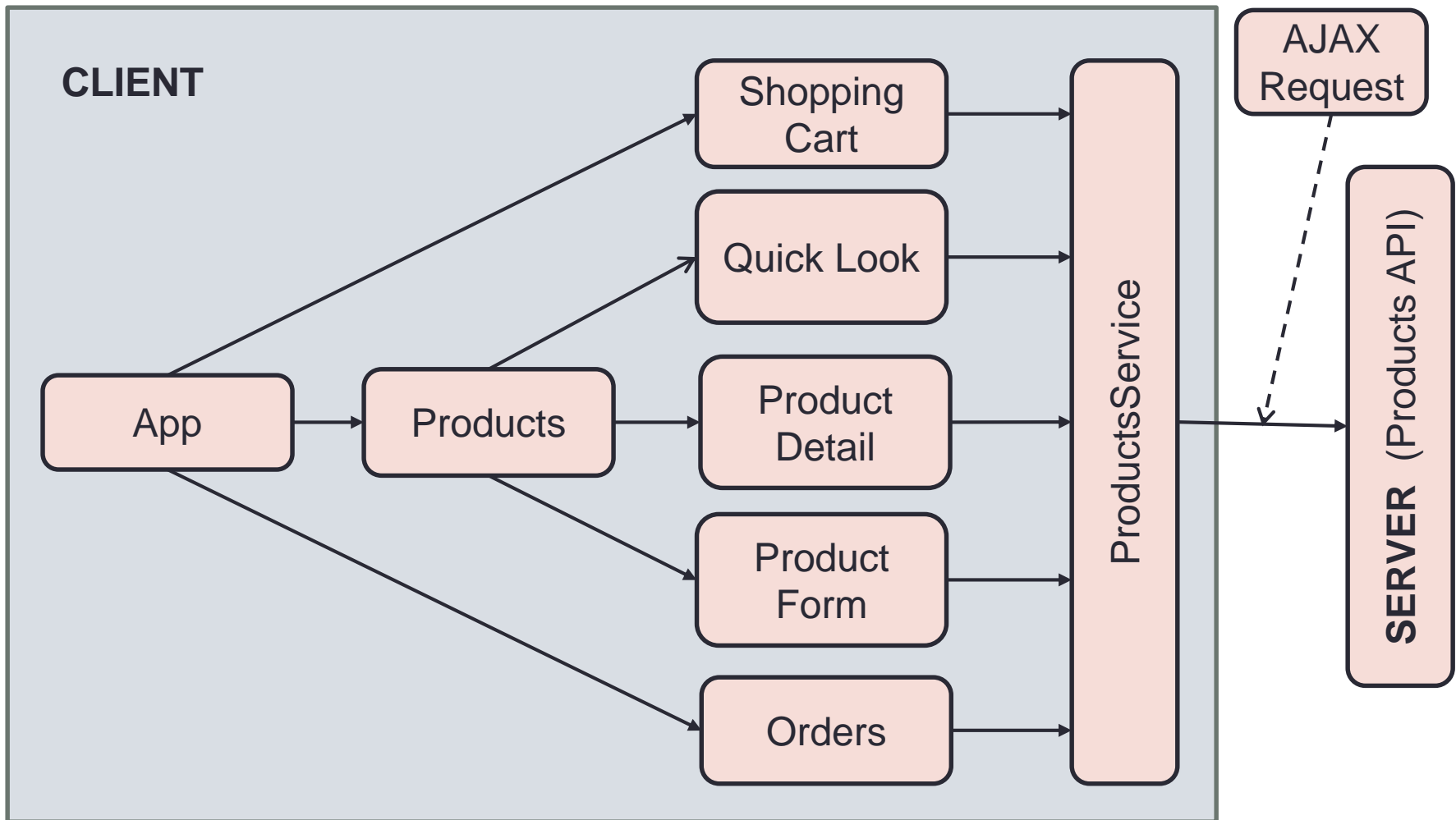
Pipe

- Takes in data as input and transforms (formats) it to a desired output
- Does not modify the underlying data
- Some examples of built-in pipes
 - lowercase
 - uppercase
 - date
 - currency
 - percent

Service

- A class with a narrow, well-defined purpose
 - Shares data and/or functionality across components
 - Encapsulates any non-UI logic
 - For e.g.
 - Logging service
 - Data service
 - Tax calculator
 - App configuration
 - Message bus
- Components consume services through Dependency Injection

Service



Router

- Enables navigation from one view to another
- Maps a URL path to a component
- AppModule
 - Import RouterModule and Routes from '@angular/router'
 - Define array of routes for the app
 - Register routes with RouterModule using 'forRoot()' method
 - Add RouterModule to 'imports' array of AppModule
- AppComponent template
 - Add <router-outlet> element
- NavComponent template
 - Use 'routerLink' attribute directive in <a> tag to navigate to a specific route
 - Products

Server Communication

- HttpClient
 - Offers a simplified client HTTP API
 - Internally uses 'XMLHttpRequest' interface exposed by browsers
- AppModule
 - Import HttpClientModule from '@angular/common/http'
 - Add HttpClientModule to imports array of @NgModule decorator
- DataService
 - Import HttpClient from '@angular/common/http'
 - Inject HttpClient instance into constructor
 - Use following methods:
 - get()
 - post()
 - put() / patch()
 - delete()
 - Above methods return Observable<T>

Q & A

- Thank you!