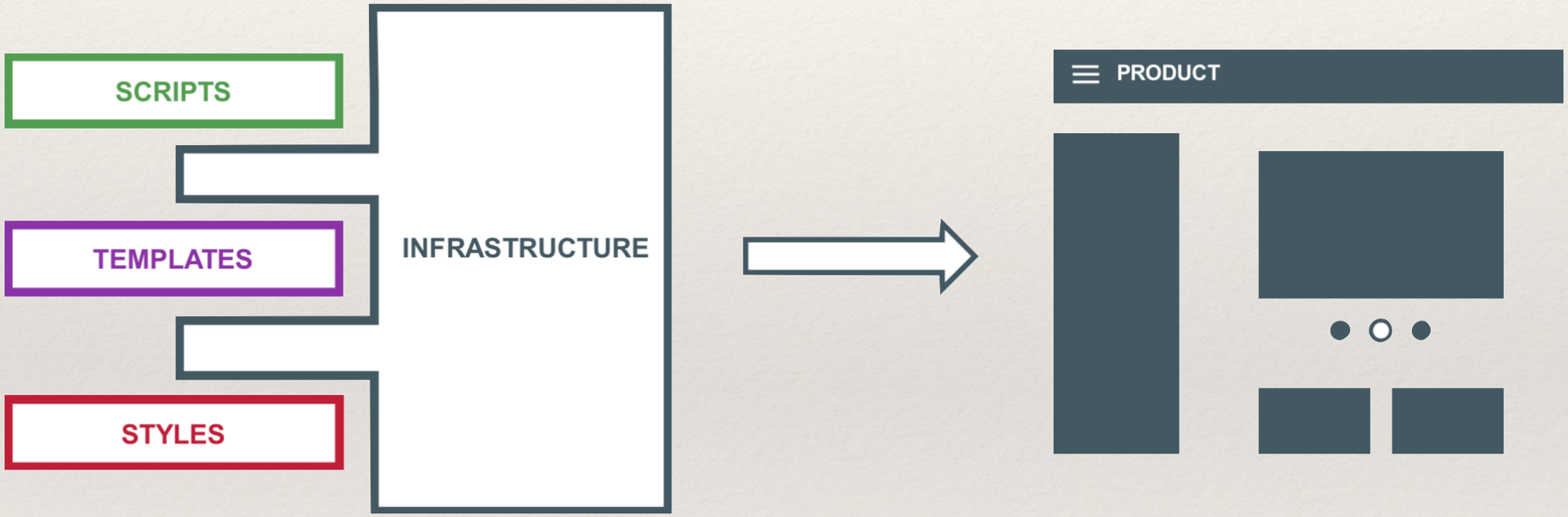
Angular
let’s get started

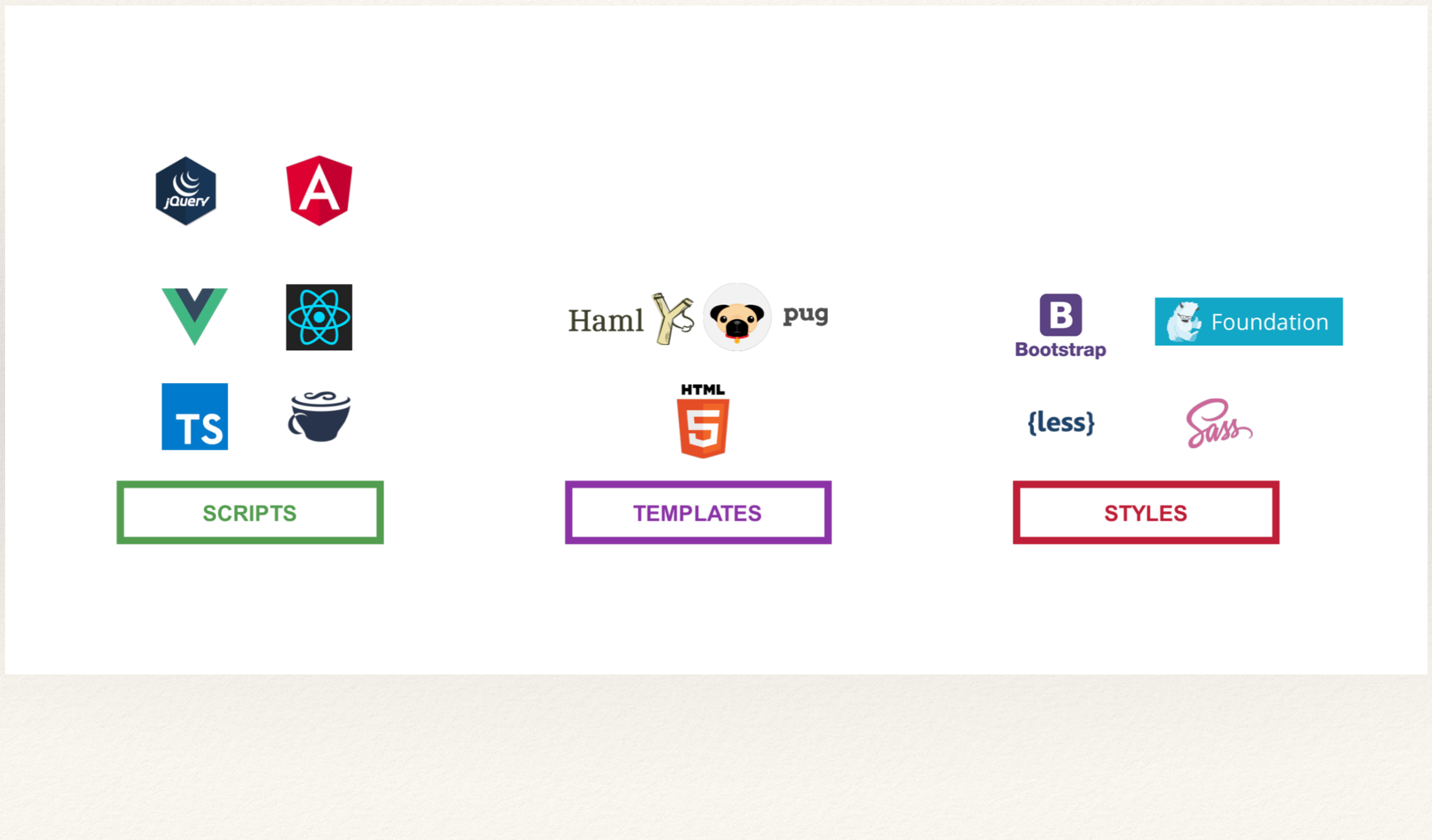
Mike Boddin

- Full Stack Developer @ devlab GmbH
- Trainer @ grossweber
- mike@boddin.net

Modern WEB



Modern WEB


jQuery
Haml
pug
Bootstrap
Foundation
TS
HTML
{less}
Sass
SCRIPTS
TEMPLATES
STYLES

Modern WEB

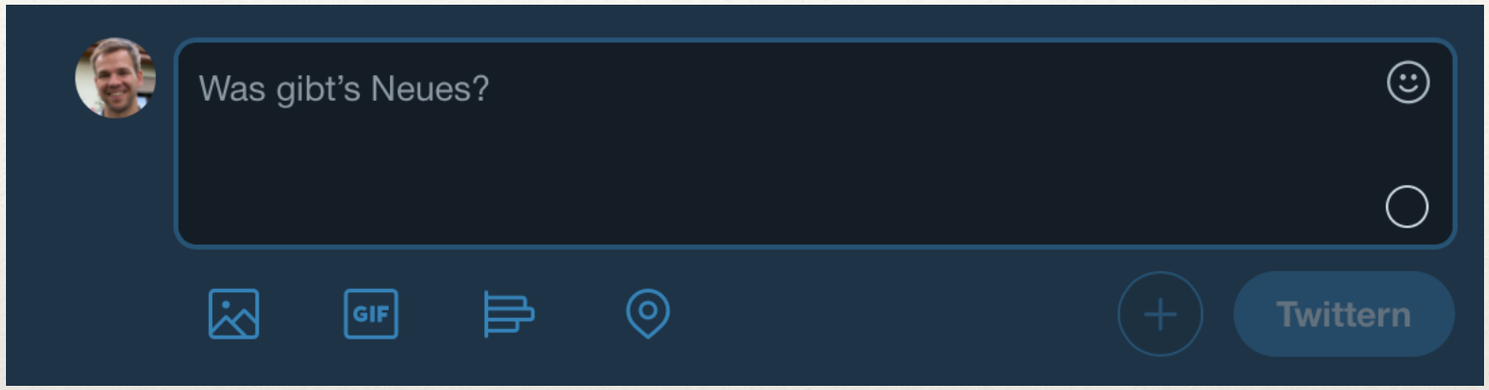

yarn
BOWER
npm
GRUNT
Gulp
webpack
MODULE BUNDLER
node
JS
INFRASTRUCTURE

Modern WEB

- Integrate
- Adapt
- Correct
- Optimize

☹ **No**
Businessvalue

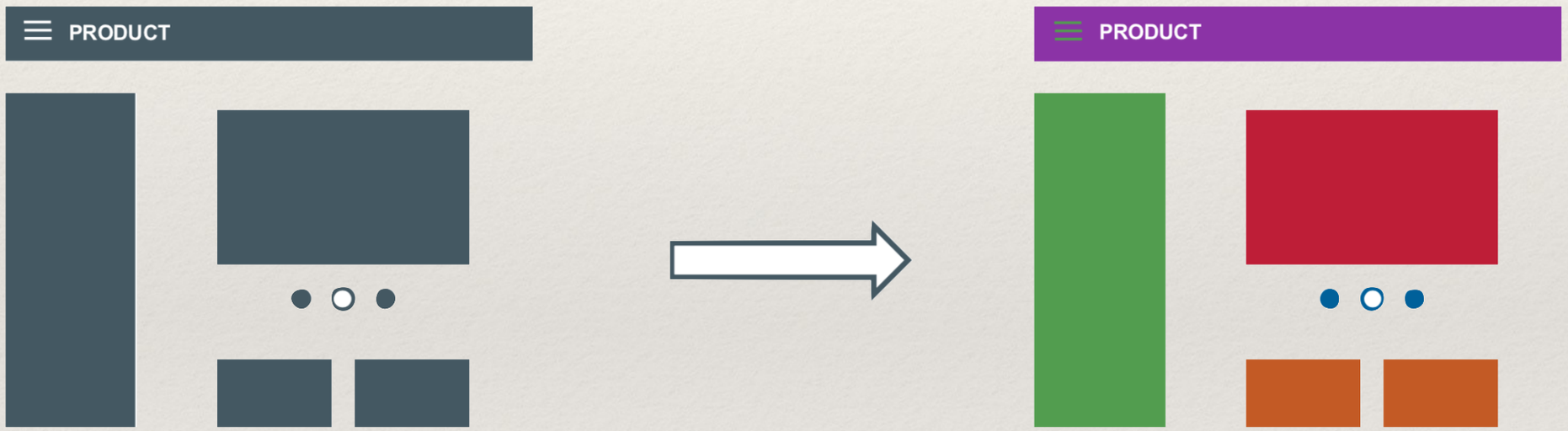
Components to the rescue



Components as abstractions of templates

```
<tweet-box
  message="Great workshop today..."
  location="Duesseldorf"
  on-submit="addTweet()" />
```

Components to the rescue



Welcome to Angular

Why Angular ?

- Build reusable web components
- TypeScript
- Great tooling and IDE support
- Up to date documentation
- Great community and wide 3rd party support

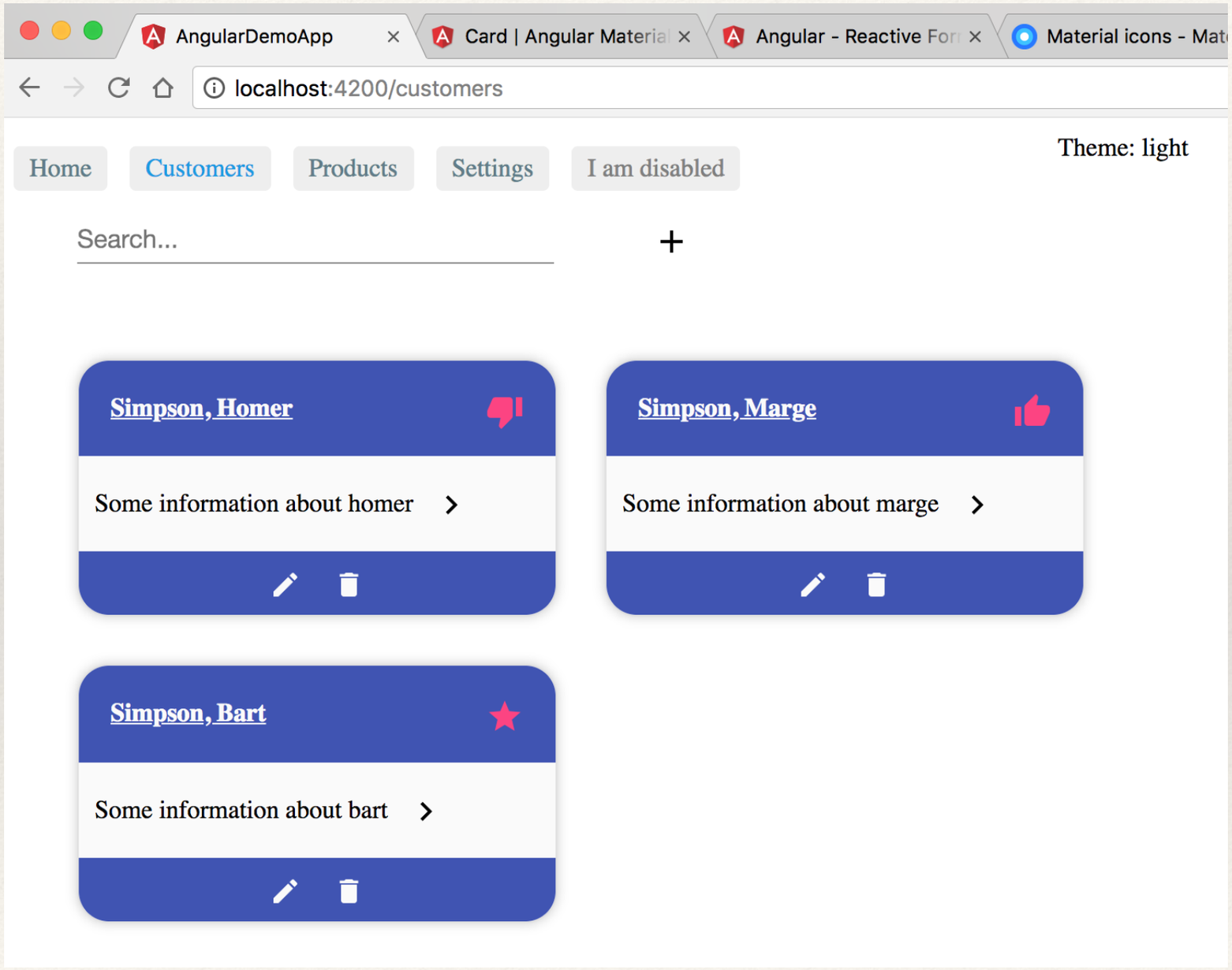
Intermediate Lesson - Typescript

See the docs

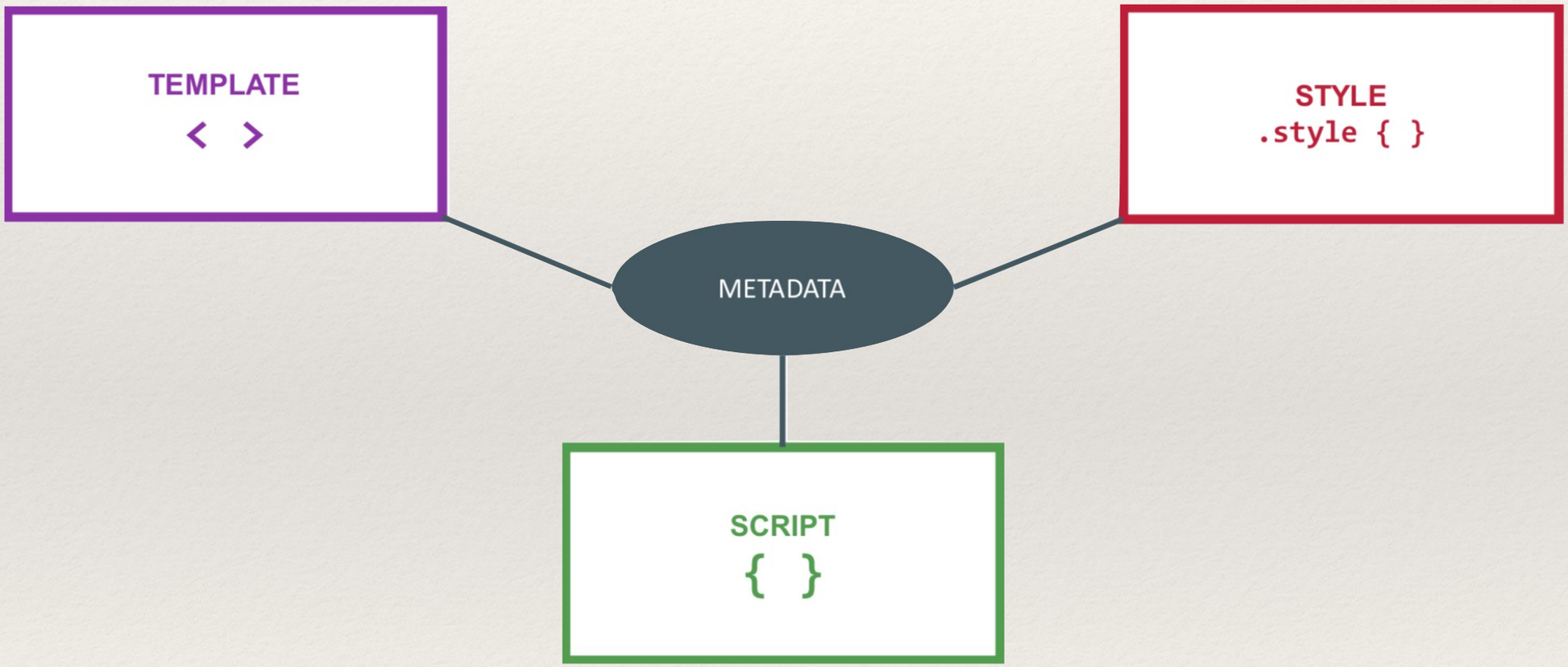
https://github.com/AngularBasic/docs

Back to Angular

Our final app...



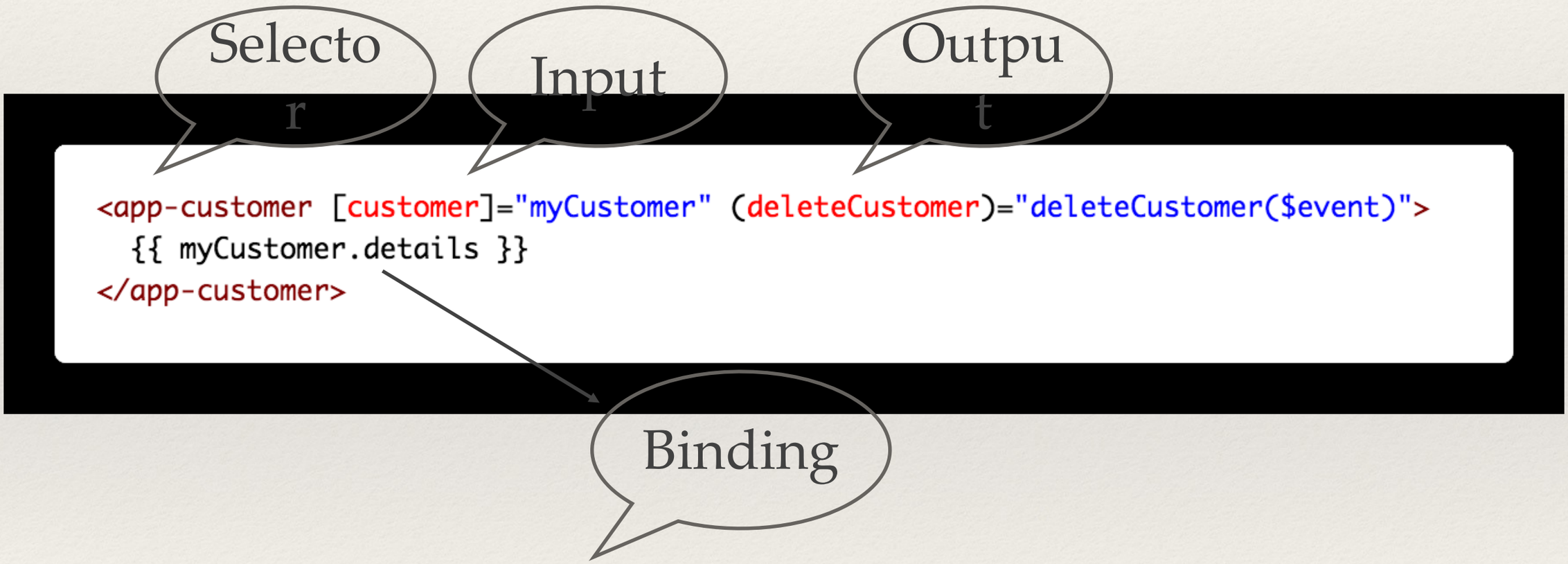
Components - Structure



Components - Decorator

```
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.scss']
})
export class AppComponent {
  title = 'App works fine!';
}
```

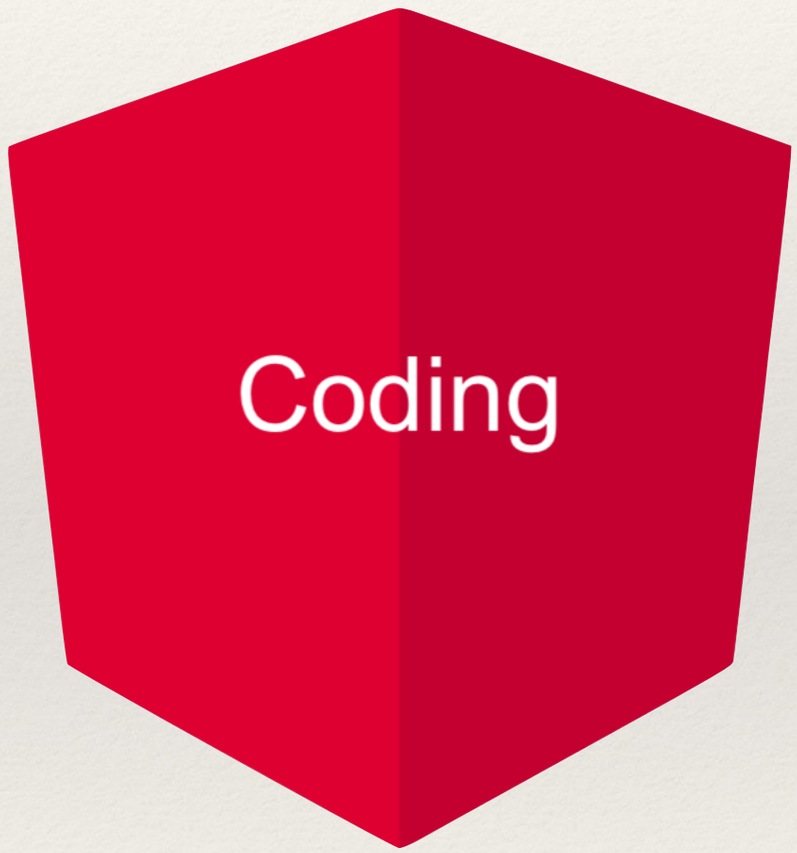
Components - Template syntax



Components - All together

```
@Component({
  selector: 'app-customer',
  templateUrl: `
    <h1>{{ customer.name }}</h1>
    <button (click)="delete(customer.id)">Delete</button>
  `
  styleUrls: ['./customer.component.scss']
})
export class CustomerComponent {
  @Input() customer: Customer;

  @Output() deleteCustomer = new EventEmitter<number>();

  delete(id: number) {
    this.deleteCustomer.emit(id);
  }
}
```

Finally some code



Clone the starter-project

```
# clone
https://github.com/AngularBasic/demo-app-starter.git

# change dir
cd demo-app-starter

# install
npm install

# check
npm start
```

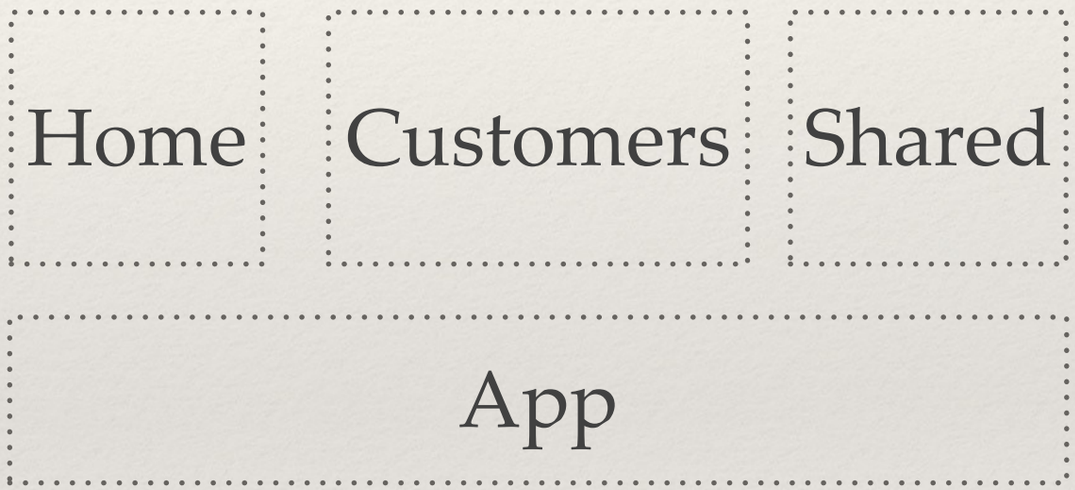
Template / Binding examples

```
<h1>Welcome to app!</h1>

<p>I am "{{customer.name | uppercase}},
   {{ customer.firstname }}"</p>

<span>These are my hobbies: </span>
<span *ngFor="let h of customer?.hobbies">{{h}} </span>

<p>I am god in math => 1 + 2 = {{ 1 + 2 }}</p>

<button (click)="showDetails = !showDetails">
    {{ showDetails ? 'Hide' : 'Show'}} the secret message
</button>
<span [hidden]="!showDetails">
  My phone number is 123 456 7890
</span>

<br />

<input #phone placeholder="phone number" />
<button
  [class.app-disabled]="!showDetails"
  [disabled]="!showDetails"
  (click)="callMe(phone.value)">
  Call me
</button>

<h5>DEBUG</h5>
<pre>user = <br/>{{ customer | json }}</pre>
```

Modules

Angular Modules

- Structure of our app
- Controls the visibility of the components
- Separation of concerns
- Enables code-splitting behind the scenes



Angular Modules

```
@NgModule({
  declarations: [
    // Building blocks of the module
    // Components, Pipes, Directives
  ],
  imports: [
    // Other modules which are needed by this module
  ],
  exports: [
    // Building blocks which could be used by other modules
    // Components, Pipes, Directives
  ],
  providers: [
    // Services which could be used by the application
  ]
})
export class AppModule { }
```

Angular - CLI

Angular CLI

- Scaffolding of the app
- Code generation
- Dev server
- Testing
- Build

```
npm install -g angular-cli
ng new my-cool-app
cd my-cool-app

ng serve
ng test
ng create
ng ...
```

Angular CLI - Pros

- Huge time saver
- Easy to setup
- No configuration needed
- Enforces preferred code style
- Official support by the Angular team
- More to come (PWA, AOT for dev, Bazel)

Angular CLI - Cons

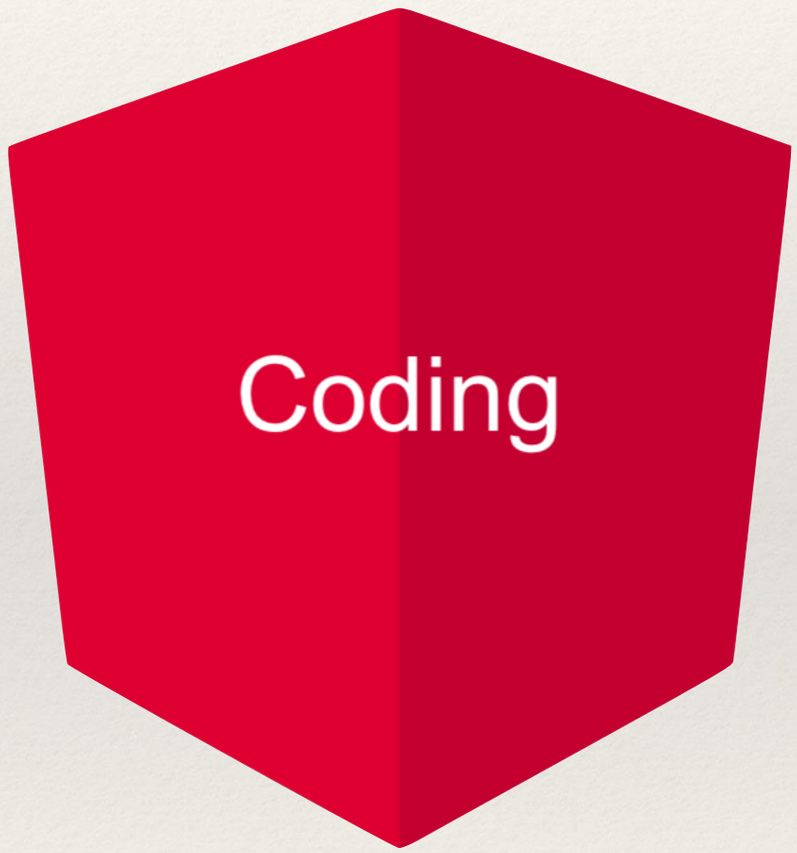
- Much magic behind the scenes
- Breaks sometimes
- Auto-reload get's stuck sometimes
- You could break out, but you shouldn't

Angular CLI - config

- „.angular-cli.json“
- prefix
- environments
- styles
- paths

```
{
  "$schema": "./node_modules/@angular/cli/lib/config/schema.json",
  "project": {
    "name": "angular-demo-app"
  },
  "apps": [
    {
      "root": "src",
      "outDir": "dist",
      "assets": [
        "assets",
        "favicon.ico"
      ],
      "index": "index.html",
      "main": "main.ts",
      "polyfills": "polyfills.ts",
      "test": "test.ts",
      "tsconfig": "tsconfig.app.json",
      "testTsconfig": "tsconfig.spec.json",
      "prefix": "app",
      "styles": [
        "styles.scss"
      ],
      "scripts": [],
      "environmentSource": "environments/environment.ts",
      "environments": {
        "dev": "environments/environment.ts",
        "hmr": "environments/environment.hmr.ts",
        "prod": "environments/environment.prod.ts"
      }
    }
  ],
  "e2e": {
    "protractor": {
      "config": "./protractor.conf.js"
    }
  },
  "lint": [
    {
      "project": "src/tsconfig.app.json",
      "exclude": "**/node_modules/**"
    },
    {
      "project": "src/tsconfig.spec.json",
      "exclude": "**/node_modules/**"
    },
    {
      "project": "e2e/tsconfig.e2e.json",
      "exclude": "**/node_modules/**"
    }
  ],
  "test": {
    "karma": {
      "config": "./karma.conf.js"
    }
  },
  "defaults": {
    "styleExt": "scss",
    "component": {}
  }
}
```

Finally more code



Components - Exercise

- Create the customer module
- Create the customer component
- Create the customer detail component
- Add a customer to the app
- Show some customer data

Angular CLI - generate a module

```
ng generate module customers

  create src/app/customers/customers.module.ts (193 bytes)
```

Angular CLI - generate a component

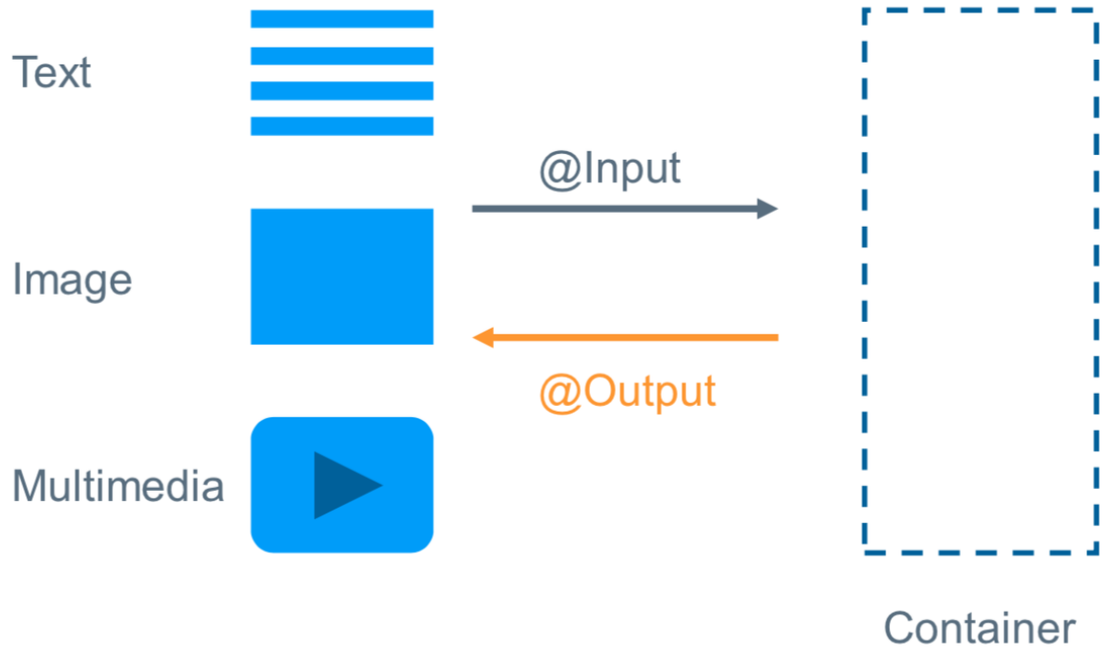
```
ng generate component customers/customer

  create src/app/customers/customer/customer.component.scss (0 bytes)
  create src/app/customers/customer/customer.component.html (27 bytes)
  create src/app/customers/customer/customer.component.spec.ts (642 bytes)
  create src/app/customers/customer/customer.component.ts (278 bytes)
  update src/app/customers/customers.module.ts (277 bytes)
```

```
ng generate component customers/customer-details

  create src/app/customers/customer-details/customer-details.component.scss (0 bytes)
  create src/app/customers/customer-details/customer-details.component.html (35 bytes)
  create src/app/customers/customer-details/customer-details.component.spec.ts (692 bytes)
  create src/app/customers/customer-details/customer-details.component.ts (309 bytes)
  update src/app/customers/customers.module.ts (445 bytes)
```

Components - Types

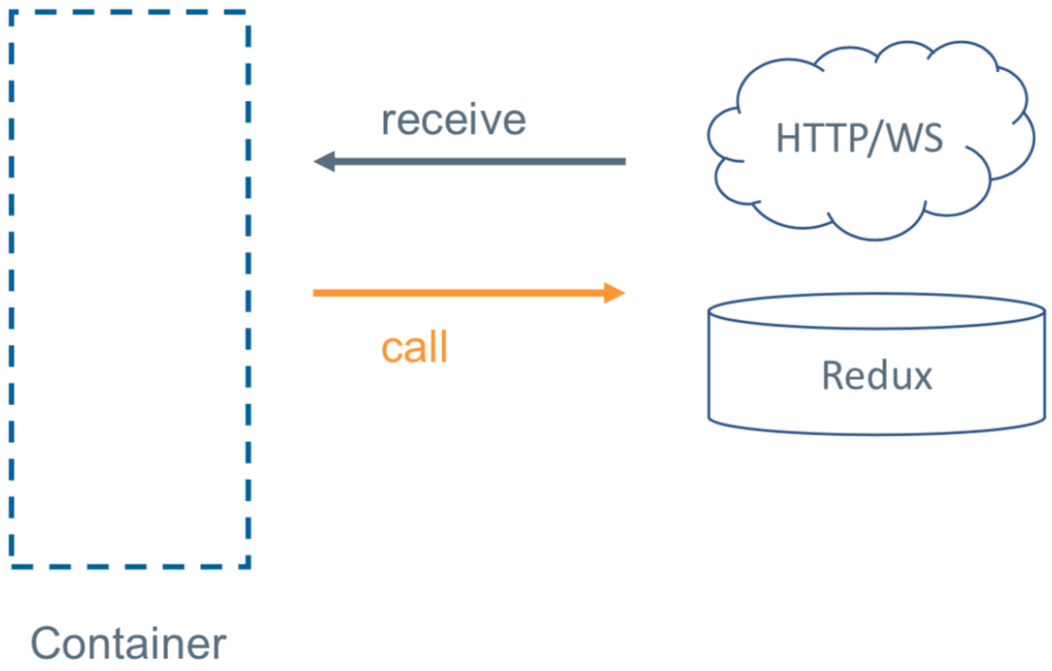
Components - Presentation

- Present Data
- Handle UI-Logic
  - Animations
  - Events
- Do not consume data services
- Are reusable building blocks



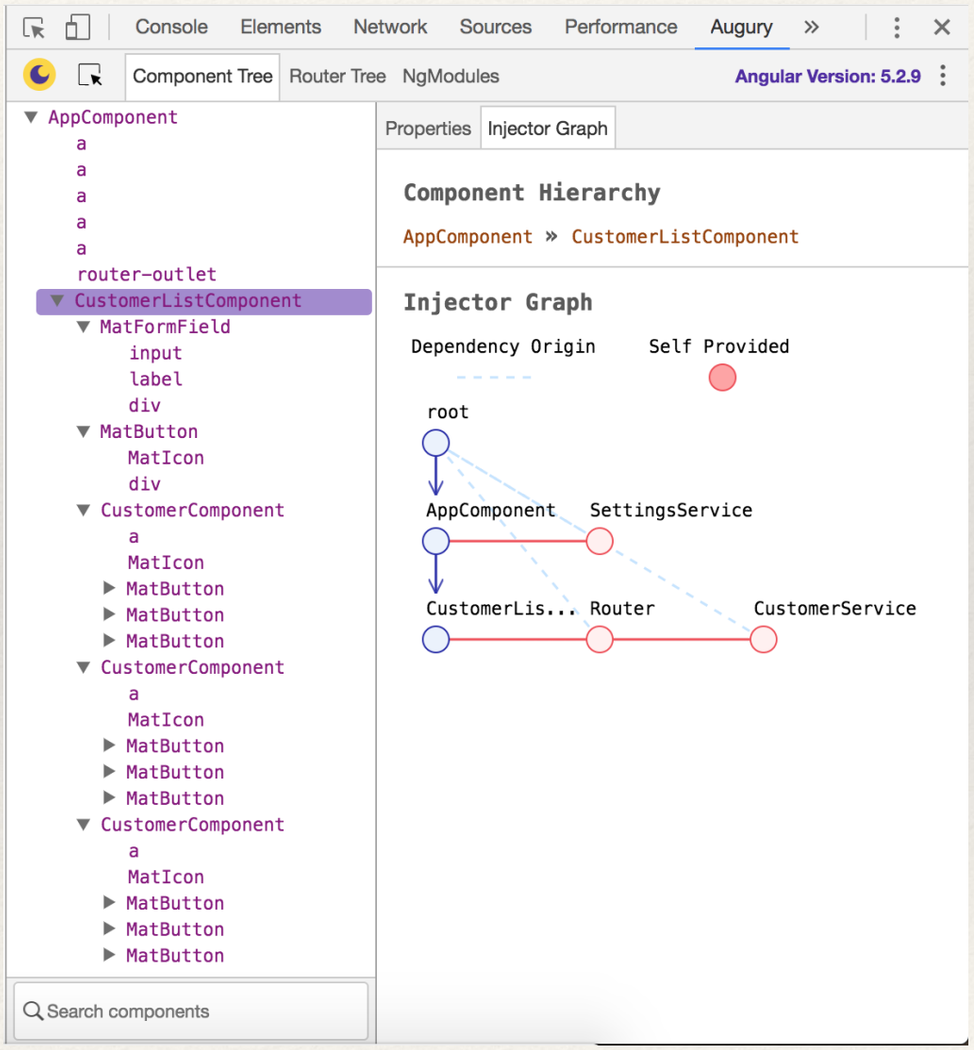
Components - Container

- Orchestrate components
  - Host Components
  - Setup Communication
- Provide data from APIs
- Made for a certain use case



Components - Tree

- „Augury“
- Chrome extension
- Visualizes the component tree
- Visualizes Dependency Injection Graph
- Allows live changes
- https://augury.angular.io



Forms

Forms - Status

- Form / control state is defined by CSS classes

```
<input class="form-control
    ng-pristine
    ng-invalid
    ng-touched" />
```

Control	true	false
was served	ng-touched	ng-untouched
was changed	ng-dirty	ng-pristine
Is valid	ng-valid	ng-invalid

Form - Validators

- Build in Validators
- You can build your own validators
- sync and async

```
class Validators {
  static min(min: number)
  static max(max: number)
  static required(control: AbstractControl)
  static requiredTrue(control: AbstractControl)
  static email(control: AbstractControl)
  static minLength(minLength: number)
  static maxLength(maxLength: number)
  static pattern(pattern: string | RegExp)
  static nullValidator(c: AbstractControl)
}
```

Forms - Template Driven

- ngModel
- Two-way-binding
- Template reference variables
- Provides information about status and validation

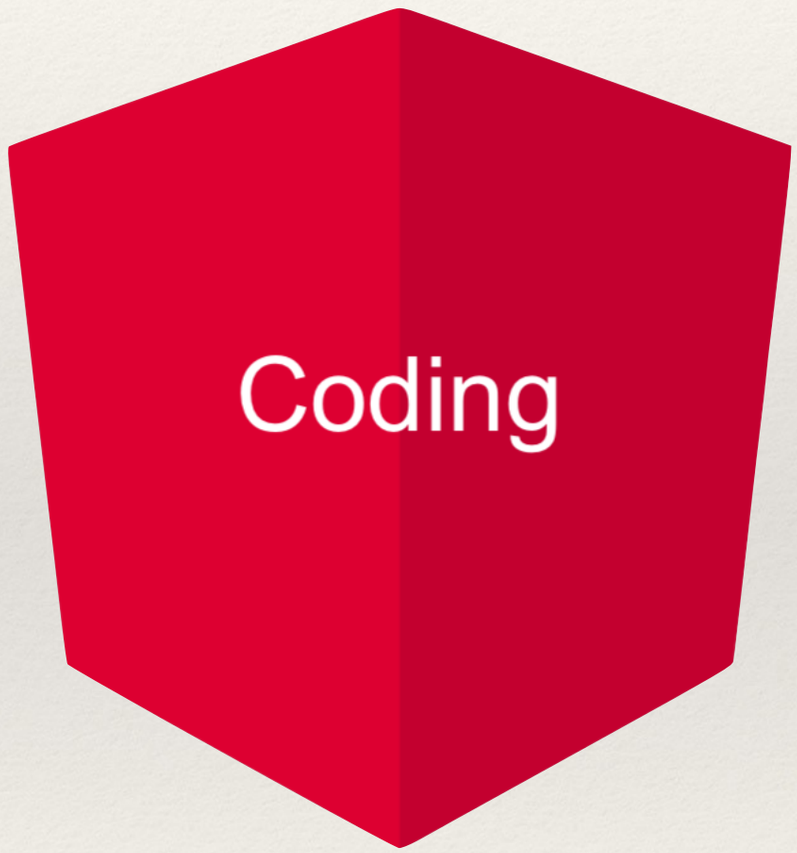
```
<input
  name="name"
  type="text"
  [(ngModel)]="customer.name"
  required
  #customerName="ngModel"
/>
<span *ngIf="customerName.invalid">
  Name is required!
</span>
```

Forms - Reactive

- Form is build in code
- Reactive event handling
- Easier to test
- Better for complex forms

```
// HTML template
<input
  name="name"
  type="text"
  [formControl]="name"
  required
/>
<span *ngIf="name.invalid">
  Name is required!
</span>

// Component
export class CustomerEditComponent1 {
  name = new FormControl();
}
```

Forms



Forms - Exercise

- Create a customer class
- Create the customer form component

Forms - Generate class / component

```
ng generate class customers/customer

create src/app/customers/customer.ts (26 bytes)
```

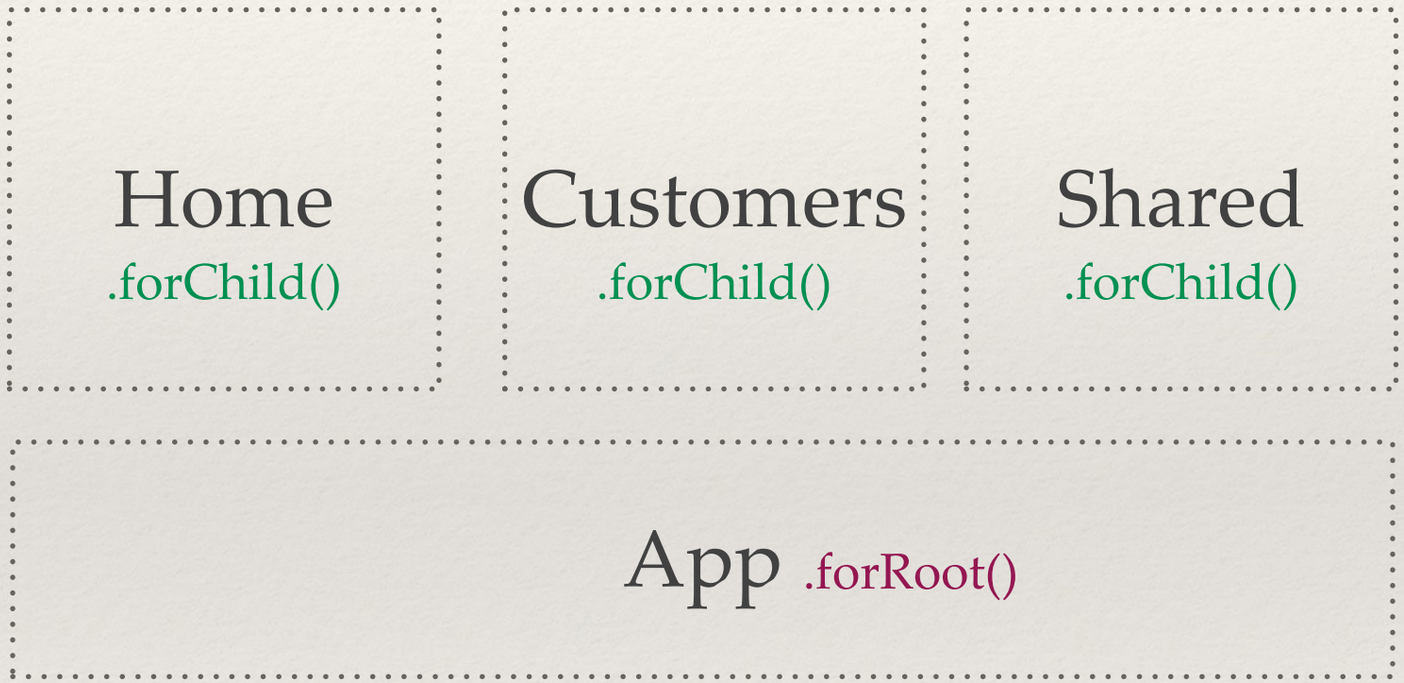
```
ng generate component customers/customer-form

  create src/app/customers/customer-form/customer-form.component.scss (0 bytes)
  create src/app/customers/customer-form/customer-form.component.html (32 bytes)
  create src/app/customers/customer-form/customer-form.component.spec.ts (671 bytes)
  create src/app/customers/customer-form/customer-form.component.ts (297 bytes)
  update src/app/customers/customers.module.ts (744 bytes)
```

Routing

Routing - Features

- Can be encapsulated in a module
- Child Routes
- Routes States
- Router outlets
- Lazy loading
- Preload
- Guards / Resolver

Routing - Configuration



Routing - Configuration

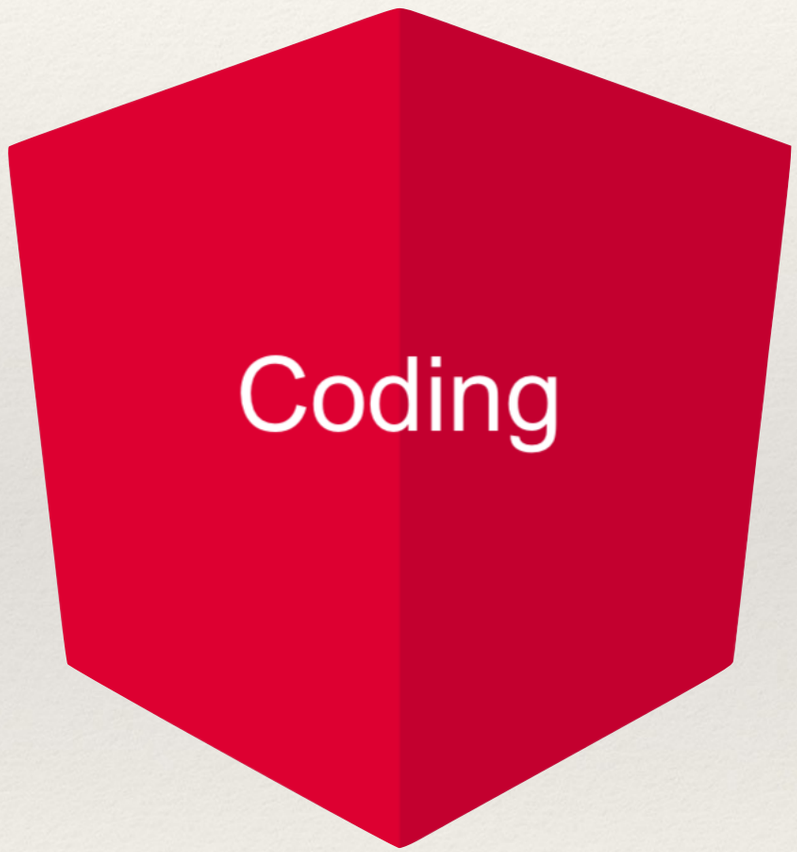
```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';
import { HomeComponent } from './home/home.component';
import { ProductsComponent } from './products/products.component';
import { CanDeactivateGuard } from './core/can-deactivate/can-deactivate-guard.service';

export const routes: Routes = [
  { path: '', redirectTo: 'home', pathMatch: 'full' },
  { path: 'home', component: HomeComponent, data: { myData: 'test' } },
  { path: 'settings', loadChildren: 'app/settings/settings.module#SettingsModule' },
  { path: '**', component: HomeComponent }
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  providers: [CanDeactivateGuard],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

Routing - Template

```
<nav>
  <a routerLink="home" routerLinkActive="active">Home</a>
  <a routerLink="customers" routerLinkActive="active">Customers</a>
  <a routerLink="customers/new" routerLinkActive="active">New Customer</a>
</nav>

<router-outlet></router-outlet>
```

<router-outlet> is used as a placeholder for the activated component

Routing



Routing - Exercise

- Create the home component
- Refactor code from app component to home component
- Create the app routing module
- Add routes
- Add navigation

Routing - Component / Module

https://github.com/AngularBasic/demo-app-starter.git

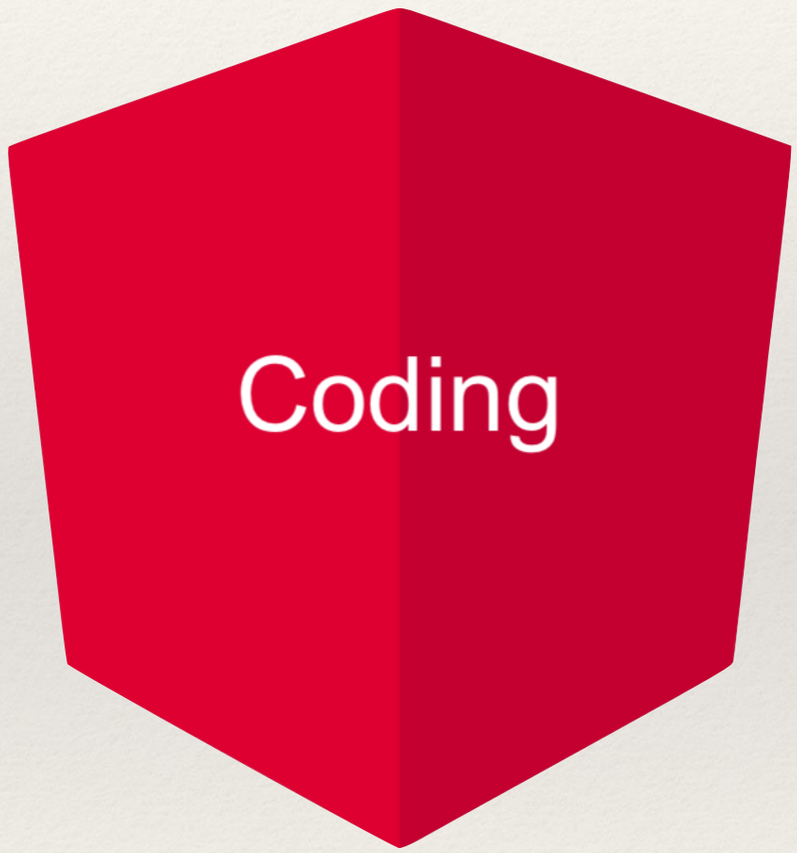
```
ng generate module home

  create src/app/home/home.module.ts (188 bytes)

ng generate component home

  create src/app/home/home.component.scss (0 bytes)
  create src/app/home/home.component.html (23 bytes)
  create src/app/home/home.component.spec.ts (614 bytes)
  create src/app/home/home.component.ts (262 bytes)
  update src/app/home/home.module.ts (251 bytes)
```

Services

Services

- Dependency injection
- Separation of concerns
- Load and transform data
- Share information within a module or the app
- Allow communication between components

Services



Services - Exercise

- Provide a service for customers
- Move customer to customer service
- Create the customers list component

Services - Exercise

```
ng generate service customers/customer

  create src/app/customers/customer.service.spec.ts (386 bytes)
  create src/app/customers/customer.service.ts (114 bytes)
```

```
ng generate component customers/customer-list

  create src/app/customers/customer-list/customer-list.component.scss (0 bytes)
  create src/app/customers/customer-list/customer-list.component.html (32 bytes)
  create src/app/customers/customer-list/customer-list.component.spec.ts (671 bytes)
  create src/app/customers/customer-list/customer-list.component.ts (297 bytes)
  update src/app/customers/customers.module.ts (1208 bytes)
```

Reactive Programming

Observables - RxJS

- Stream of events over time
- Declarative
- Can be cancelled
- Multiple Operators, which can be chained
- Deeply integrated in Angular (Routing, Forms, HTTP, Events)

RxJS - Example

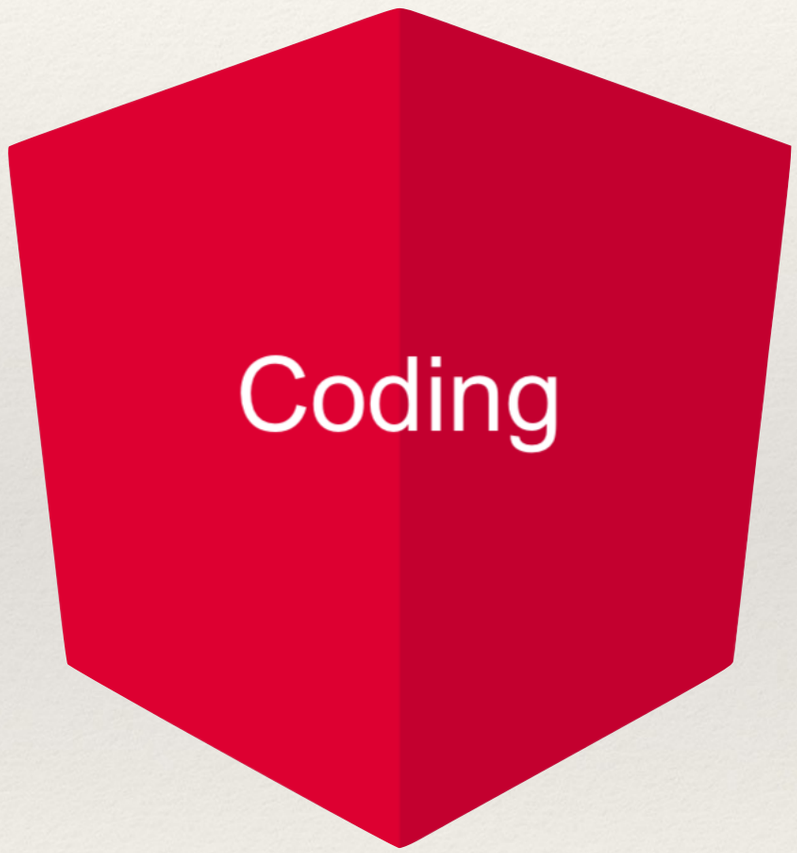
```
const rate = 1000;
let count = 0;
let lastClick = Date.now() - rate;

const button = document.querySelector('button');

button.addEventListener('click', () => {
  if (Date.now() - lastClick >= rate) {
    console.log(`Clicked ${++count} times`);
    lastClick = Date.now();
  }
});
```

JS Style

RxJS Style

```
const button = document.querySelector('button');

Rx.Observable.fromEvent(button, 'click')
  .throttleTime(1000)
  .scan(count => count + 1, 0)
  .subscribe(count => console.log(`Clicked ${count} times`
));
```

RxJS



RxJS - Exercise

- Navigate to edit form from customer
- Fill form with customer data
- Navigate back to customer list

HTTP

HTTP

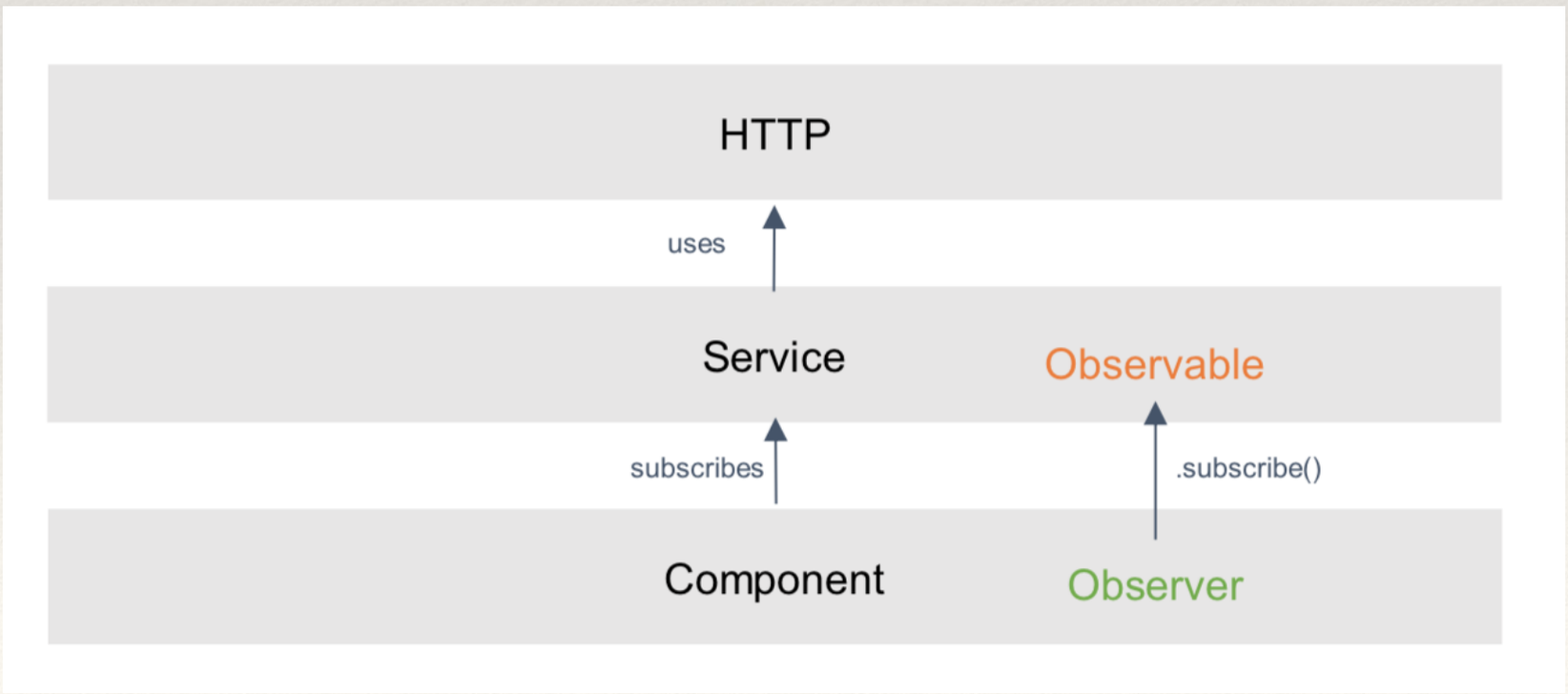
- Reactive HTTP API
- Based on RxJS
- Provide Interceptors to transform request/response
- Progress events

HTTP - Interface

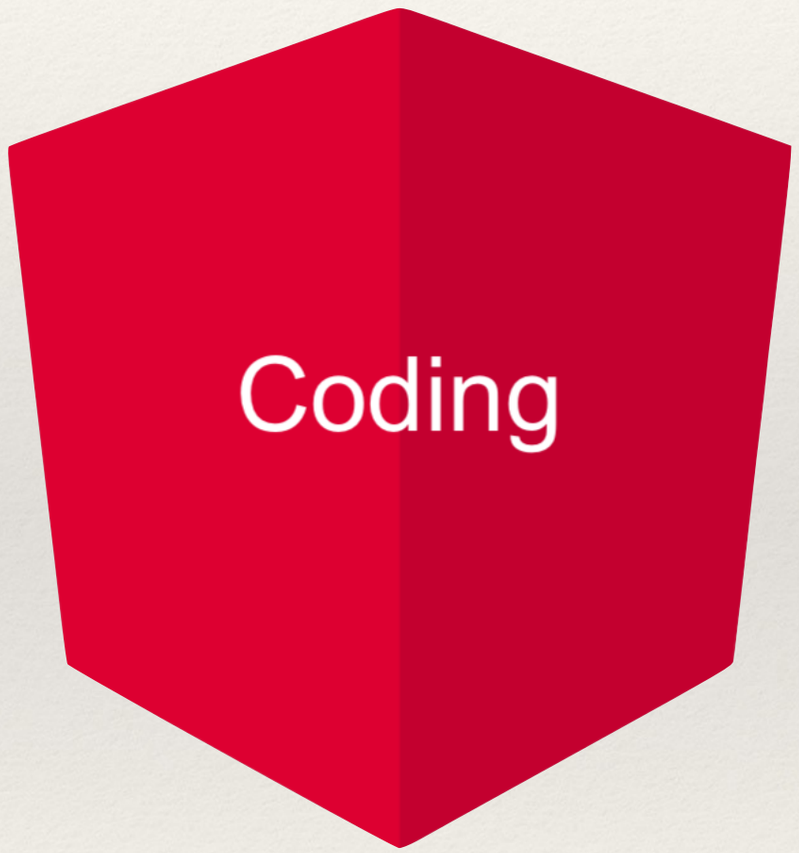
```
import { HttpClient } from '@angular/common/http';

this.http
  .get<T>    (url: string, options)
  .delete<T>(url: string, options)
  .post<T>   (url: string, body: any, options)
  .put<T>    (url: string, body: any, options)
  .patch     (url: string, body: any, options)
  .head      (url: string, options)
  .options   (url: string, options)
  .request   (url: string | Request, options)
```

HTTP - Tips

- Do not use HttpClient directly in components
- Only services should talk to the HTTP-APIs
- Do not forget to subscribe() to the HttpClient methods



HTTP



HTTP - Exercise

- Use the local node.js backend to load customers
- Add create / update functions in customer service
- Add search for customers

HTTP - Start the backend

```
# Start the backend server
npm run start:server

# Start the frontend
npm start
```

Testing

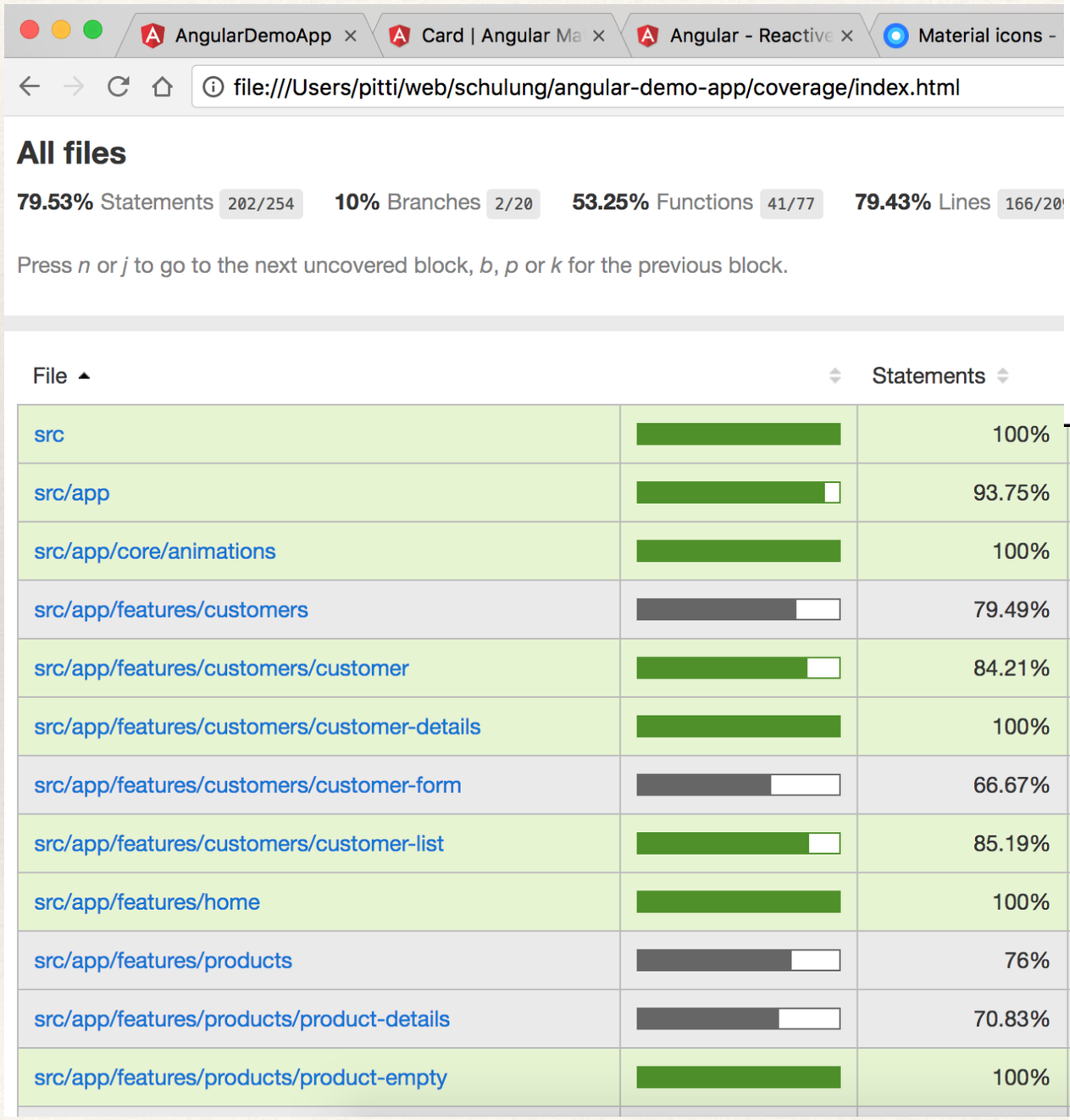
Testing

- Run by the angular-cli
- Unit Tests
- Component Tests
- e2e Tests
- Code coverage

Testing - Syntax

```
import { TestBed, async } from '@angular/core/testing';
import { AppComponent } from './app.component';
import { NO_ERRORS_SCHEMA } from '@angular/core';
import { SettingsService } from './features/settings/settings.service';
import { HttpClient } from '@angular/common/http';
import { NoopAnimationsModule } from '@angular/platform-browser/animations';

describe('AppComponent', () => {
  beforeEach(async(() => {
    TestBed.configureTestingModule({
      imports: [NoopAnimationsModule],
      declarations: [
        AppComponent
      ],
      providers: [
        SettingsService,
        { provide: HttpClient, useValue: {} }
      ],
      schemas: [NO_ERRORS_SCHEMA]
    }).compileComponents();
  }));
  it('should create the app', async(() => {
    const fixture = TestBed.createComponent(AppComponent);
    const app = fixture.debugElement.componentInstance;
    expect(app).toBeTruthy();
  }));
  it(`should have as title 'app'`, async(() => {
    const fixture = TestBed.createComponent(AppComponent);
    const app = fixture.debugElement.componentInstance;
    expect(app.title).toEqual('app');
  }));
  it('should render the nav', async(() => {
    const fixture = TestBed.createComponent(AppComponent);
    fixture.detectChanges();
    const compiled = fixture.debugElement.nativeElement;
    expect(compiled.querySelector('nav').children.length).toBe(5);
  }));
});
```

Testing - Code coverage




```
npm test -- --code-coverage
```

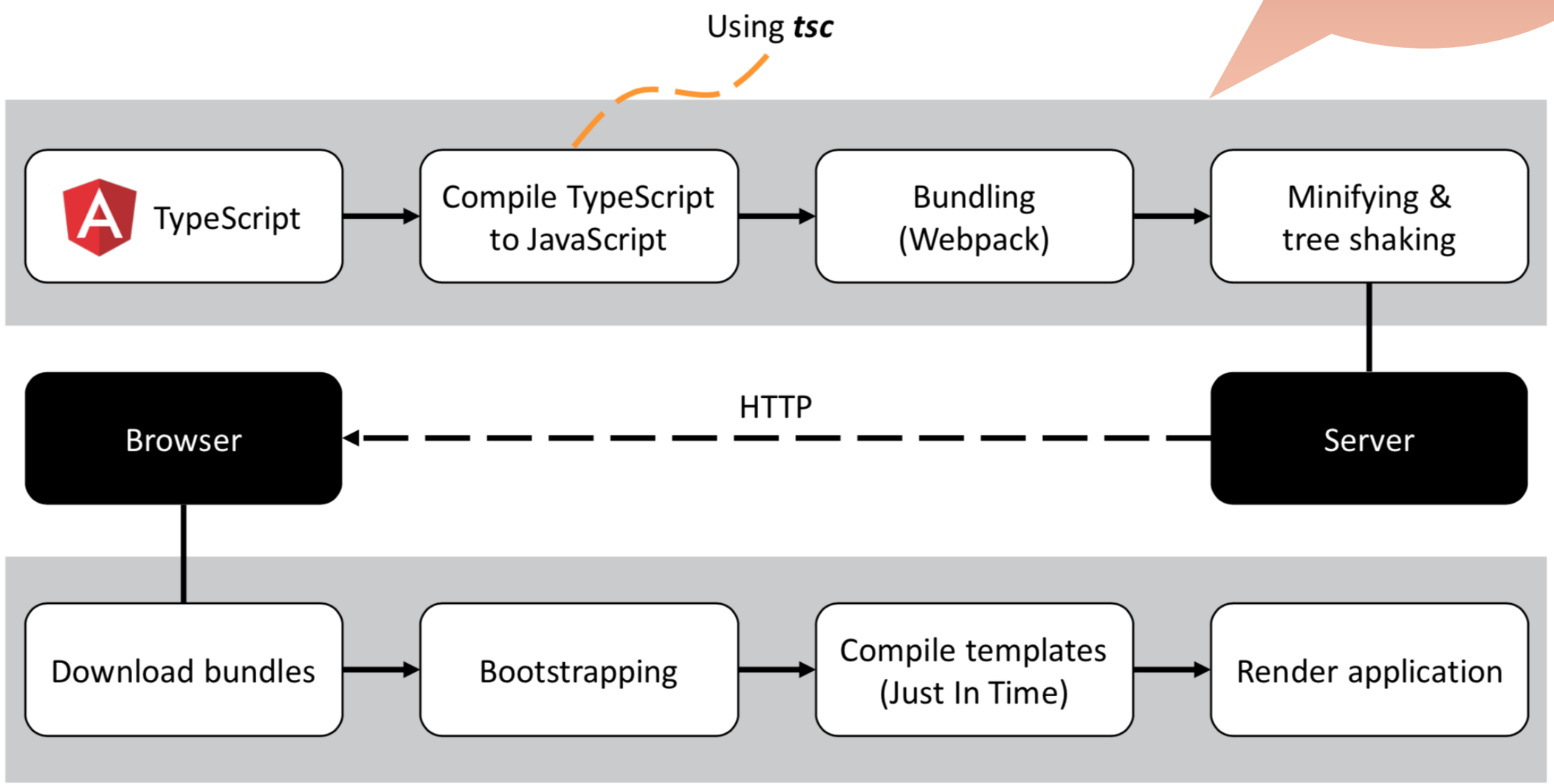
Testing - Pro Tips

- Use code coverage to find untested code
- Try out Jest (https://github.com/thymikee/jest-preset-angular)
- Try out cypress.io (https://www.cypress.io)

Deployment

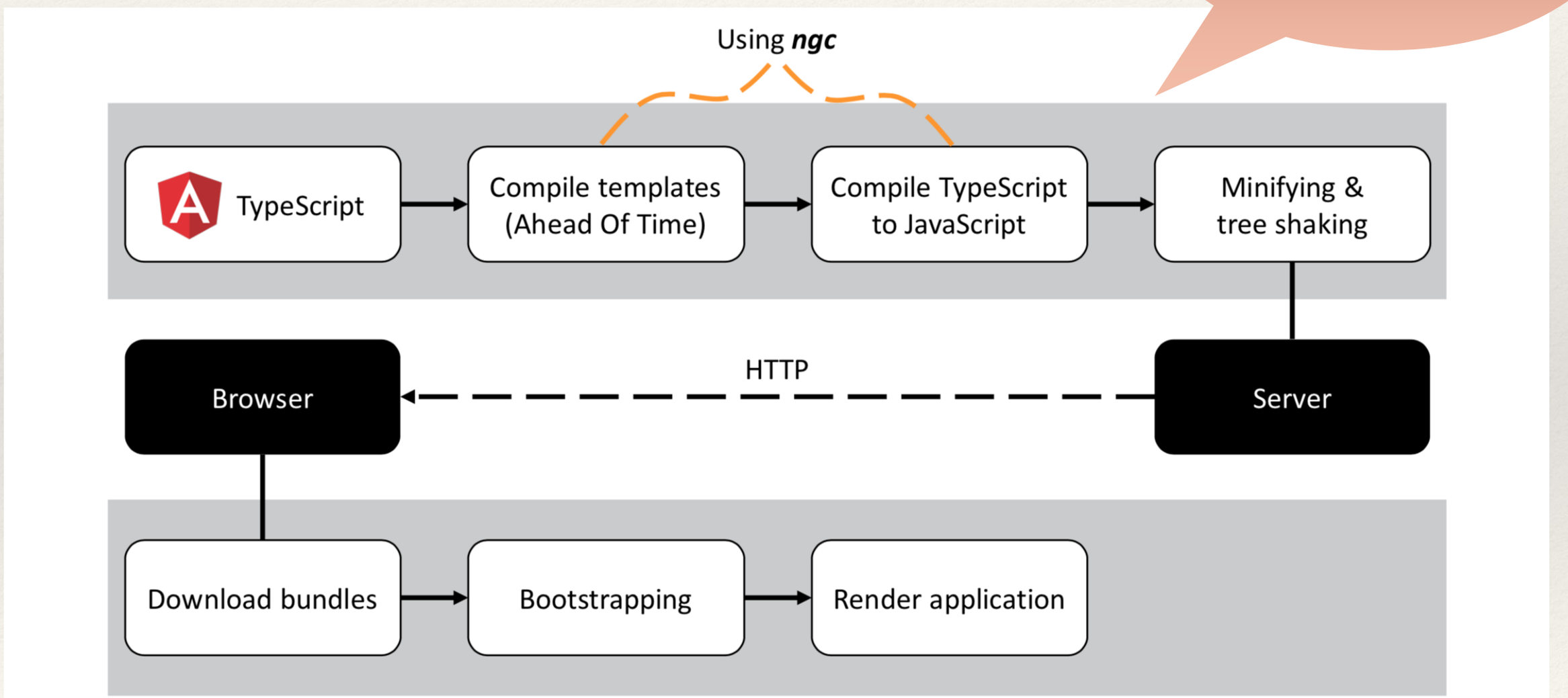
Deployment - Dev

ng build



Deployment - Prod

ng build - - prod



Additional Content

- Create enterprise-grade apps with nrwl/nx: https://github.com/nrwl/nx-examples
- Translate your app with ngx-translate: http://www.ngx-translate.com/
- SignalR-Support (dotnet core) with @aspnet/signalr

Feedback would be nice

- https://grossweber.com/zertifikat
- https://grossweber.com/feedback

- Thank you :-)