webtraining

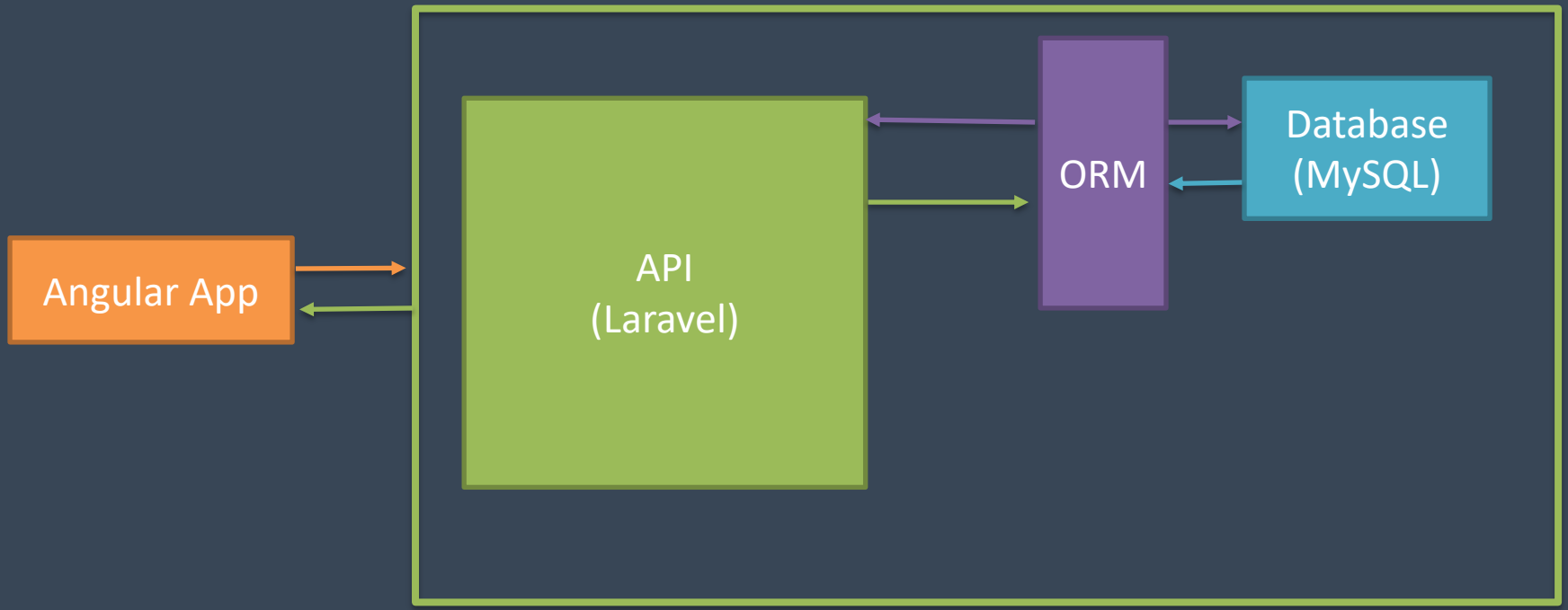
Angular

Introduction

What will we learn?

- NPM
- CLI
- Angular app structure
- Components
- Services
- Inputs/Outputs
- Pipes

Architecture



webtraining

The basics

What the h… is an *SPA*?

- **Single Page Application**
  - Angular
  - React
  - Vue.js
  - …

No, so fast… What is a RESTful API?

- Mechanism to interact with data sources via a semantic URI convention using HTTP verbs
  - What?

When should I use a framework?

- Multiple data sources (databases, file system)
- Integration with other systems
- Third party ”head-less” platforms
- Pure front-end teams

Why Angular?

- Framework not just *library*
- TypeScript / ES2015+ ready
- “Web-component” oriented
- License friendly!
- Enterprise-ready
- Now, it is light!

Exercise

Creating an Angular App

webtraining

CLI Installation

Use the CLI

- Angular CLI

```
npm install @angular/cli –g
ng new my-app --style=scss
```

Exercise

Let’s understand the “boilerplate”

webtraining

App structure

Some important files

- package.json
- angular.json
- src/tsconfig.json
- tslint.json

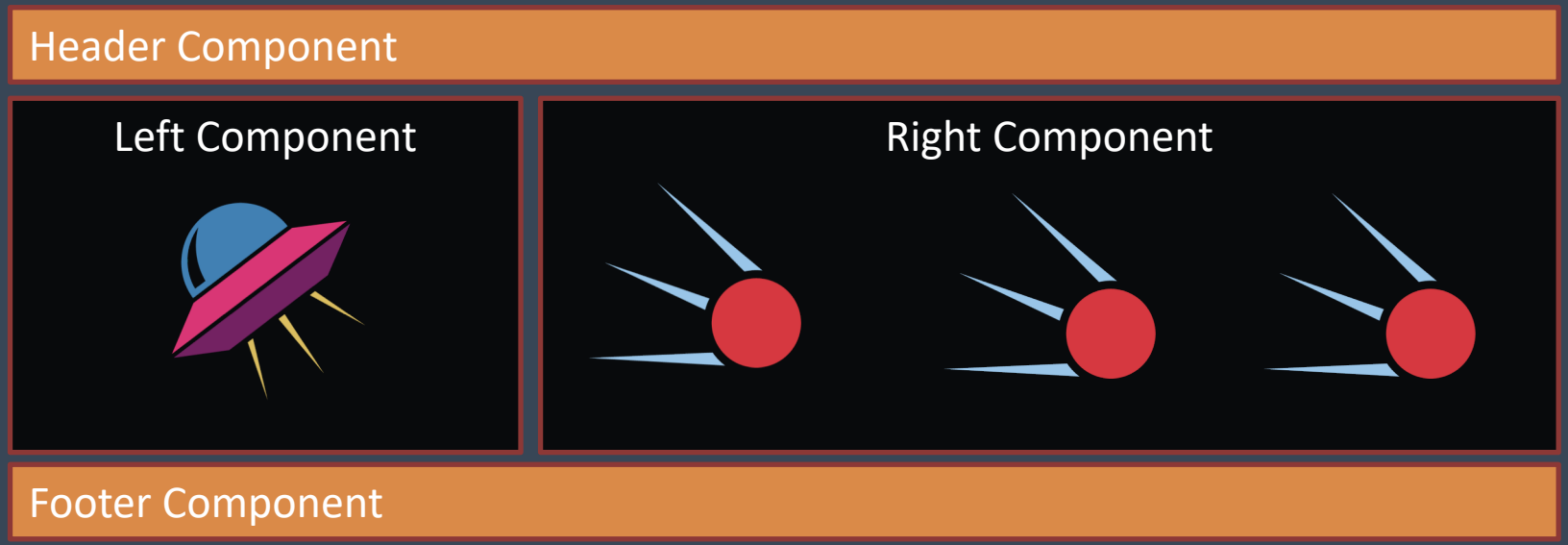
Some important directories

- Src
- e2e
- node_modules
- dist

Components

Angular @Component

- Build UI elements and logic on the page



webtraining

Exercise

Creating a “Header” component

Minimal component structure

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

@Component({
  selector: 'my-app',
  template: `<h1>Hello {{name}}</h1>`
})
export class AppComponent { name = 'Angular'; }
```

Templates

Exercise

The concept of template

Managing data in templates

- Interpolation
  - Single values
  - Arrays
  - Objects

Directives

Managing data in templates

- Logic: ***ngIf**
- Iteration: ***ngFor**

Exercise

Adding custom styles

Exercise

Using external CSS (CDN)

Services

Angular @Injectable()

- A service represents a way to interact with data acquisition
- Can use HTTP for communication (via Observables)

Exercise

Creating a service for static data

Minimal service structure

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

@Injectable()
export class HeroService { }
```

webtraining

HTTP Client

Request with HTTP Client

```
http
  .get<MyJsonData>('/data.json', {observe: 'response'})
  .subscribe(response => {
    // Here, response is of type HttpResponse<MyJsonData>.
    // You can inspect its headers:
    console.log(response.headers.get('X-Custom-Header'));
    // And access the body directly,
    // which is typed as MyJsonData as requested.
    console.log(response.body.someField);
  });
```

Exercise

Connection to a REST API

webtraining

Observables

Exercise

Creating a service for dynamic data

Pipes

Angular @Pipe

- Write display-value **transformations** declared in our HTML
- Similar to a pipe in GNU/Linux or a **filter** in other template systems

Exercise

Creating a pipe

Minimal pipe structure

```
@Pipe({name: 'lowercase'})
class LowercasePipe {
  transform(value: string, args: any[]) {
    return value.toLowerCase();
  }
}
```

```
<h2>{{ title | lowercase }}</h2>
```

webtraining

Angular

Session #2

Inputs / Outputs

Angular @Input

- Sends data **from parent** component to a child component

```
<poke-catalog [pokemon]="pokemonInParentComponent"></poke-catalog>
```

```
@Input() pokemon : Pokemon[];
```

Exercise

Implementing @Input

Angular @Output

- Sends data **from a child** component to its parent component

```
<poke-navigation (searchValueUpdated)="searchValueUpdated($event)"></poke-navigation>
```

```
@Output() searchValueUpdated = new EventEmitter();
```

Exercise

Implementing @Output

Thank you!

Alex Arriaga
https://webtraining.zone/
@alex_arriaga_m