Diego Zúñiga

Angular

Diego Zúñiga

Agenda

- Introduction
- Architecture
  - Workspace, Apps, Libraries, Modules
  - Components, Directives, Pipes, Services
- Angular CLI

Diego Zúñiga

Agenda (2)

- HTTP
- Forms
- Routing
- Security

Diego Zúñiga

Agenda (3)

- Tests
- Deployment
- PWA

Introduction

- JavaScript Framework
- Client Applications in HTML and TypeScript
- Angular is written in TypeScript
- Basic Building Blocks are NgModules
- Bootstrapping - Root NgModule

Diego Zúñiga

Architecture

- Modules
- Components
  - Views
  - Directives
- Services
  - Dependency Injection (DI)
  - Routing

Modules

- Funcional Units (Components, Services, ...)
- Compilation Context for Components
- Angular App -> Root Module === **AppModule**
- NgModule -> NgModules
- Complex Apps, Reusability, **Lazy Loading**

Diego Zúñiga

Components

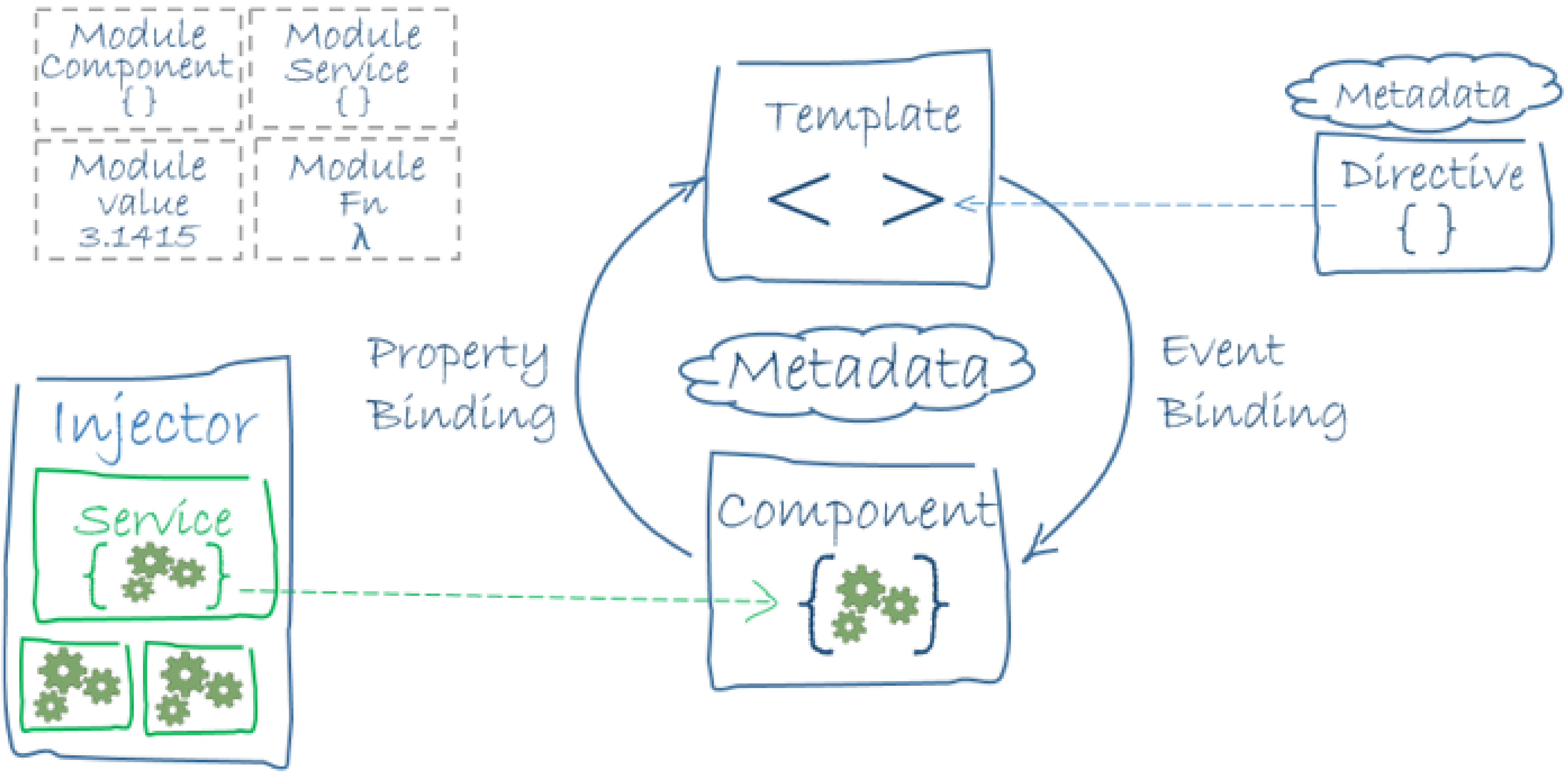
- Angular App -> + Components -> Root Component
- Class + @Component (HTML Template + Metadata)
- Decorators - Functions that modify JavaScript classes
- Templates? Directives?? Pipes???
- Data Binding? Two-way Data Binding?? Event Binding??? Property Binding???

Services

- Data | Logic that is not associated with Views
- Can be shared across components
- Class + @Injectable

overview

Diego Zúñiga



Diego Zúñiga

Setup

Diego Zúñiga

Prerequisites

- node 10.9.0+
- npm
- angular cli

```
npm install -g @angular/cli
```

Create and run an app

```
ng new my-app
cd my-app
ng serve --open
```

Diego Zúñiga

Angular CLI

- ***ng new*** - workspace + initial app
- workspace = apps + libraries
- initial app - top level -> ***src/***
- initial app = root module + root component
- additional apps -> ***projects/***
- ***ng add*** / ***ng generate***

Angular Console

- https://angularconsole.com/
- GUI
- Everything the CLI can do. And more.
- https://nrwl.io/

Diego Zúñiga

Configuration Files

file	description
.editorconfig	https://editorconfig.org/
angular.json	workspace config, build/serve/test/
tsconfig.json	TypeScript config for all projects in the workspace
tslint.json	TSLint config for all projects in the workspace
src/	source code for root-level app

Diego Zúñiga

Multiple project file structure

- **src**/
- projects/
  - project-01/
    - **src**/
  - project-02/
    - **src**/
  - ...
  - project-nn/
    - **src**/

Diego Zúñiga

Application source files src/

File	Purpuse
app/	components
assets/	images
environments	config by env
favicon.ico	An icon to use for this application in the bookmark bar.
index.html	The main HTML page

Diego Zúñiga

Application source files src/ (2)

File	Purpuse
main.ts	The main entry point of your app
polyfills.ts	Provides polyfill scripts for browser support.
styles.sass	Lists CSS files that supply styles for a project.
test.ts	The main entry point for your unit tests

Diego Zúñiga

Multi-project workspace

```
ng new my-workspace --createApplication="false"
cd my-workspace
ng generate application my-app
ng generate library my-lib
ng serve --open
```

Diego Zúñiga

MODULES

NgModules

- NgModule = Components + Services + ...
- Import functionality that is exported from other Modules
- Export functionality for use by other NgModules

Metadata

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

@NgModule({
    imports: [],
    providers: [],
    declarations: [],
    exports: [],
    bootstrap: []
})
export class AppModule {}
```

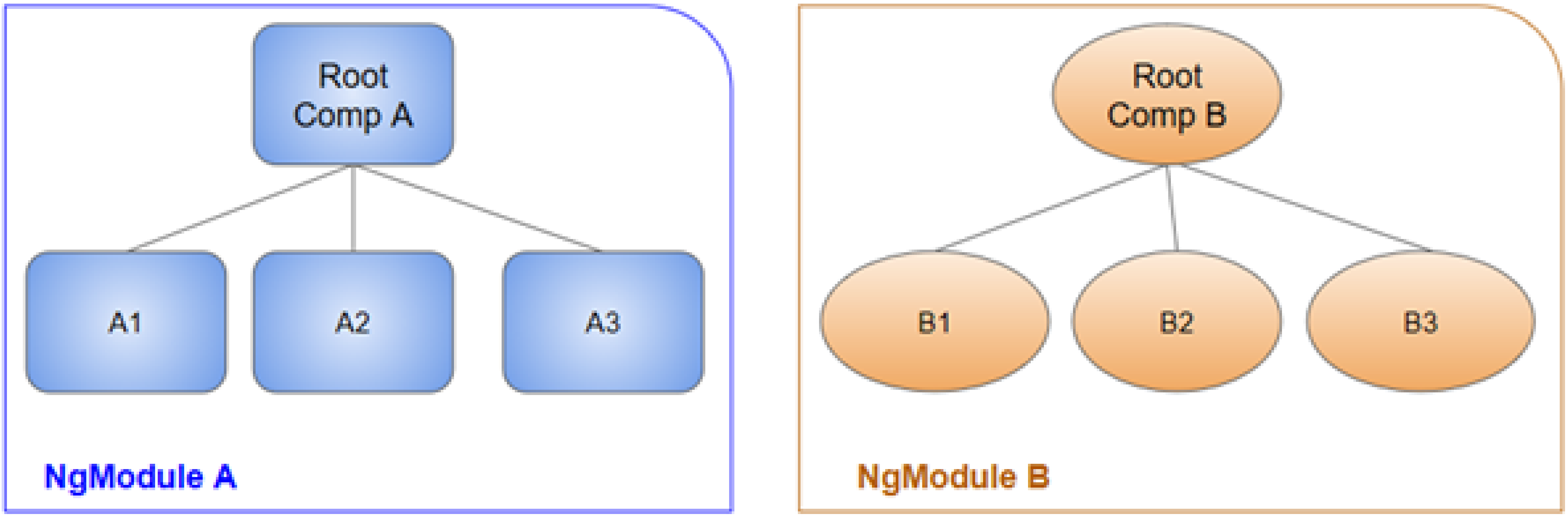
Diego Zúñiga

Metadata (2)

- **declarations**: The components, directives, and pipes that belong to this NgModule.
- **exports**: The subset of ***declarations*** that should be visible and usable in the component templates of other NgModules.
- **imports**: Other modules whose exported components, directives, and pipes that are needed by component templates declared in this NgModule.
- **providers**: Creators of services that this NgModule contributes to the global collection of services.
- **bootstrap**: The main application view, called the root component.

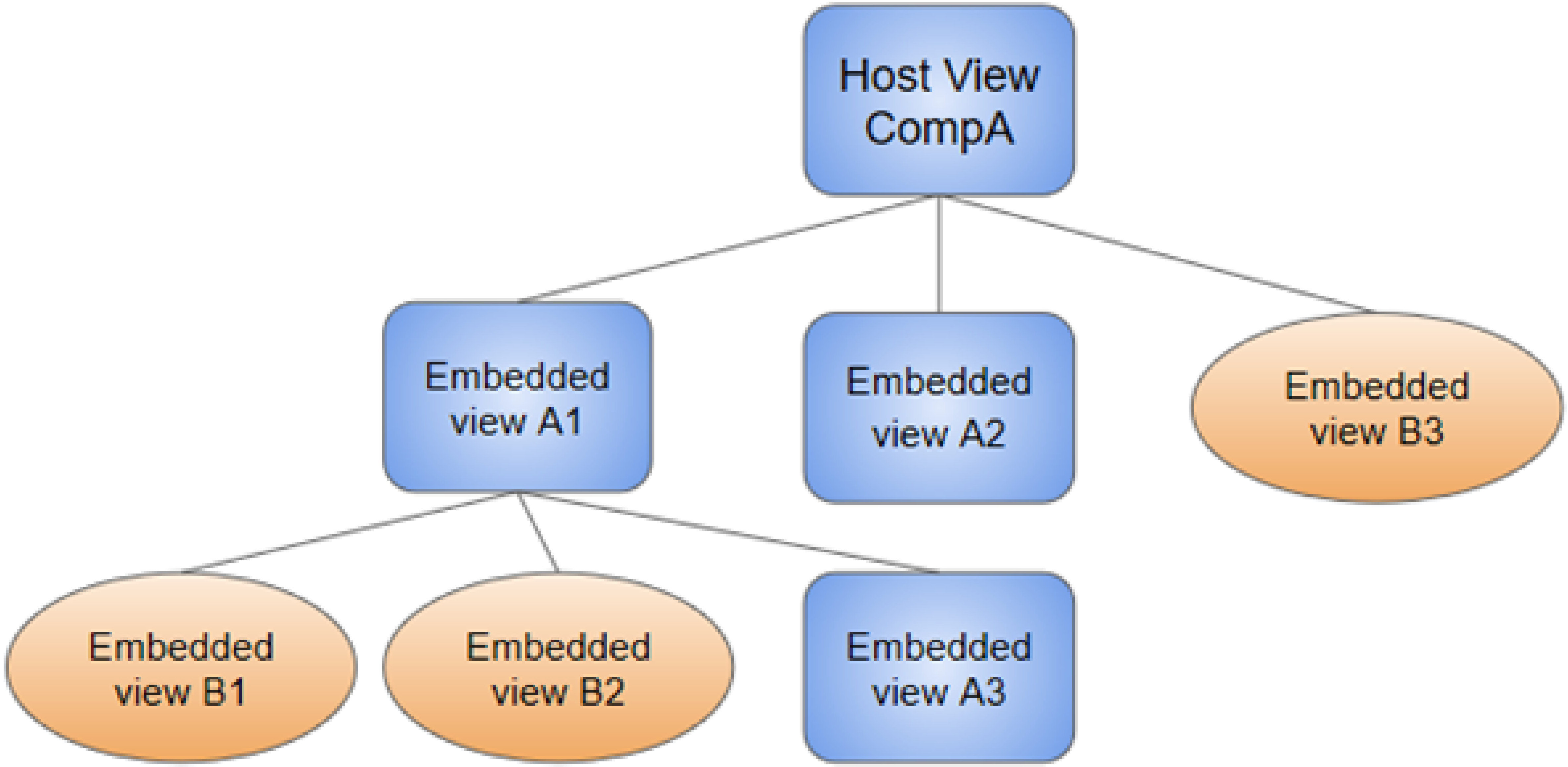
Compilation Context

Diego Zúñiga



Views Hierarchy

Diego Zúñiga



Diego Zúñiga

COMPONENTS

Metadata

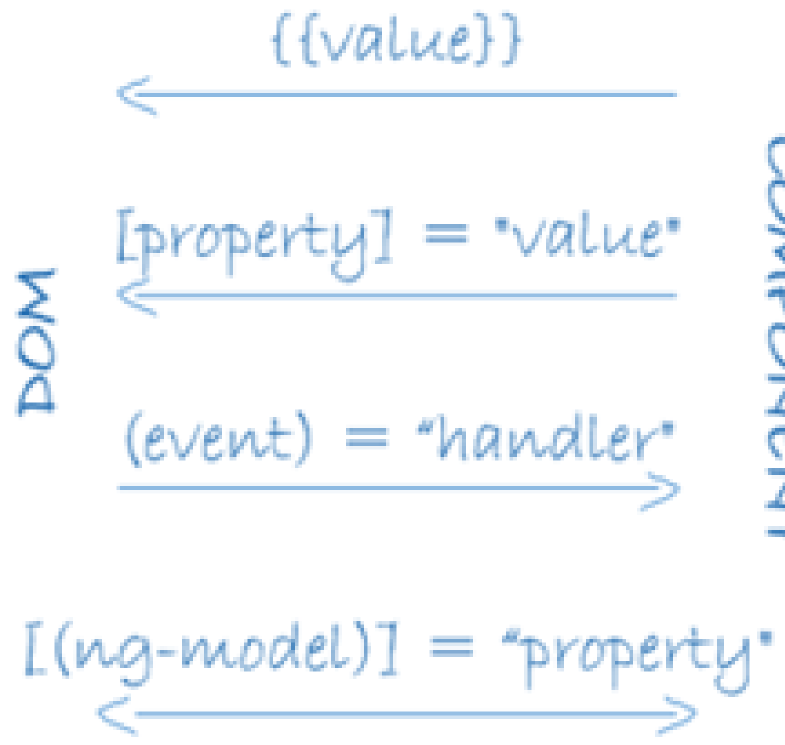
```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.scss']
})
export class AppComponent {}
```

Component

- A component controls a patch of screen called a view.
- You define a component's application logic -- what it does to support the view—inside a class
- The class interacts with the template through an API of properties and methods.

Data Binding

Diego Zúñiga



Diego Zúñiga

Workshop Application

- **bookshop**
  - bookshop-website
    - users, books, cart, checkout
  - bookshop-admin
    - users, books
  - bookshop-service *

Diego Zúñiga

Create bookshop workspace

```
ng new bookshop --createApplication="false"
cd bookshop
ng generate application bookshop-website --routing --style=scss
```

Diego Zúñiga

Clean up app skeleton

```html
<!-- app.component.html -->
<h1>{{ title }} app is running!</h1>
<router-outlet></router-outlet>
```

Diego Zúñiga

Run bookshop app

```
cd bookshop
npm serve --open
```

Diego Zúñiga

Font Setup

```
<!-- index.html -->
<link href="...Montserrat:300,300i,400,600,700"
    rel="stylesheet">
```

Diego Zúñiga

Bootstrap

```
npm install --save bootstrap
```

```scss
// styles.scss
$font-family-base: "Montserrat", sans-serif;
@import "bootstrap";
```

Icons

```
npm install --save @fortawesome/fontawesome-svg-core
npm install --save @fortawesome/free-solid-svg-icons
npm install --save @fortawesome/angular-fontawesome
```

Diego Zúñiga

Icons (Module)

```
// app.module.ts
import { FontAwesomeModule }
    from '@fortawesome/angular-fontawesome';

@NgModule({
    imports: [
        BrowserModule,
        FontAwesomeModule,
        AppRoutingModule
    ],
})
```

Icons (Component)

```
import { faShoppingCart }
    from '@fortawesome/free-solid-svg-icons';

@Component({...})
export class MyComponent {
  faShoppingCart = faShoppingCart;
}
```

Diego Zúñiga

Icons (Template)

```
<fa-icon [icon]="faShoppingCart"></fa-icon>
```

Diego Zúñiga

Globals

Generate global components

```
ng generate component global/header
ng generate component global/logo
ng generate component global/search
ng generate component global/util-nav
ng generate component global/menu
ng generate component global/footer
```

Diego Zúñiga

Generate home components

```
ng generate component home
ng generate component hero
ng generate component shop-features
```

Diego Zúñiga

DIRECTIVES

Structural Directives

- Structural directives are responsible for HTML layout
- An asterisk (*) precedes the directive attribute name

Structural Directives - ngIf

```
<div *ngIf="user">Hello, {{user.name}}</div>

<ng-template [ngIf]="hero">
    <div>Hello, {{user.name}}</div>
</ng-template>
```

Diego Zúñiga

Structural Directives - ngFor

```
<ul>
    <li *ngFor="let book of books">{{book.title}}</li>
</ul>

<ul>
    <ng-template ngFor let-book [ngForOf]="books">
        <li>{{book.title}}</li>
    </ng-template>
</ul>
```

logicstudio
INNOVATION | SERVICES | SOLUTIONS

Diego Zúñiga

Workshop Application (2)

Feature Modules

```
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';

@NgModule({
  imports: [CommonModule],
  declarations: []
})
export class BooksModule { }
```

Routed Feature Module

- Lazy Loading

```
ng generate module books
    --route books --module app.module
```

Diego Zúñiga

Routing Feature Module

- Move hero/, shop-features/ to home/
- Create a Routing Feature Module

Diego Zúñiga

Routing Feature Module - news

```
ng generate module news --routing
ng generate component news/news --flat
```

Diego Zúñiga

Routed Feature Module - shopping-cart

```
ng generate module shopping-cart --routing
    --route cart --module app.module
```

Navigation - Active

```
<a class="nav-link"
    [routerLink]="menuItem.path"
    routerLinkActive="active">
    {{menuItem.label}}
</a>
```

Diego Zúñiga

Pipes

- A pipe takes in data as input and transforms it to a desired output.

```
{{ exp | pipe [ : arg1 [ : arg2 [ : ... ] ] ] }}
```

```
<p>{{ birthday | date }}</p>
<p>{{ birthday | date : 'fullDate' | uppercase }}</p>
```

Diego Zúñiga

Built-in pipes

- DatePipe
- UpperCasePipe
- LowerCasePipe
- CurrencyPipe
- DecimalPipe
- PercentPipe

Diego Zúñiga

CurrencyPipe

```
{{ exp | currency [:code [:display [:digits [:loc ]]]] }}
```

```
{{book.price | currency : '$' : 'USD' : '1.2-2' }}
```

DatePipe

```
{{ exp | date [:format [:timezone [:locale]]] }}
```

```
{{ book.publishedDate | date : 'MM-yyy'}}
```

Diego Zúñiga

SERVICES

@Injectable

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

@Injectable({
  providedIn: 'root'
})
export class BookService {}
```

Diego Zúñiga

Dependency Injection (DI)

```
import { HttpClient } from '@angular/common/http';
...
export class BookService {
    constructor(private http: HttpClient) {}
}
```

Http Module

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

@NgModule({
    ...
    BrowserModule, // CommonModule
    HttpClientModule,
})
```

Dependency Injection (DI)

```
import { HttpClient } from '@angular/common/http';
...
export class BookService {
    constructor(private http: HttpClient) {}

    getBooks() {
        return this.http.get('url');
    }
}
```

Diego Zúñiga

Workshop Application (3)

BookService

```
ng generate service services/book
ng generate interface models/book
```

Diego Zúñiga

Routing and Navigation

Diego Zúñiga

url params

```
<a routerLink="/books/{{book.id}}"
    class="btn btn-outline-primary">View Details</a>
```

activated route / snapshot

```
export class MyComponent implements OnInit {
    constructor(private route: ActiveRoute) {}
    ngOnInit() {
        const { id } = this.route.snapshot.params;
    }
}
```

activated route / subscribe

```
export class MyComponent implements OnInit {
    constructor(private route: ActiveRoute) {}
    ngOnInit() {
        this.route.paramMap.subscribe((paramMap: ParamMap) => {
            const id = paramMap.get('id');
            const hasId = paramMap.has('id');
            const keys = paramMap.keys()
        });
    }
}
```

Diego Zúñiga

Observable + AsyncPipe

```
this.book$ = this.bookService.getBookById(id);
```

```
<app-book *ngIf="book$ | async as book" [book]="book">
</app-book>
```

Diego Zúñiga

Workshop Application (4)

BookDetail -> Book

```
ng generate component books/book
ng generate component books/book-detail
```

Diego Zúñiga

Security -> login

```
ng generate module security --routing
ng generate component security/register
ng generate component security/login
ng generate interface models/user
ng generate interface models/credentials
ng generate interface models/login-response
ng generate interface models/user-profile
ng generate service services/auth
```

Diego Zúñiga

Forms

Template-driven Validations

```
<input type="email" name="email" class="form-control"
    #emailField="ngModel" [(ngModel)]="email"
    required email />
<div *ngIf="emailField.errors.required">
    Email is required
</div>
<div *ngIf="emailField.errors.email">
    Email is invalid
</div>
```

ngModel controller

```
<input name="email" [(ngModel)]="value" #emailCtr="ngModel" />
```

```
emailCtrl.value
emailCtrl.valid
emailCtrl.invalid
emailCtrl.pristine
emailCtrl.dirty
emailCtrl.touched
emailCtrl.untouched
emailCtrl.errors
```

Diego Zúñiga

ngForm controller

```
<form #formCtrl="ngForm">
```

```
formCtrl.value
formCtrl.valid
formCtrl.invalid
formCtrl.pristine
formCtrl.dirty
formCtrl.touched
formCtrl.untouched
formCtrl.errors
```

CSS classes

- .ng-valid
- .ng-invalid
- .ng-pending
- .ng-pristine
- .ng-dirty
- .ng-untouched
- .ng-touched

Diego Zúñiga

Validators - built-in

- min max
- required
- email
- minLength maxLength
- pattern

NgClass

```
<input type="email" name="email" class="form-control"
    #emailField="ngModel" [(ngModel)]="email"
    [ngClass]="{'is-invalid': emailField.invalid}"
    required email />
```

ngSubmit

```
<form #loginForm="ngForm"
    (ngSubmit)="login(loginForm)"
    novalidate>
```

Diego Zúñiga

HTTP Interceptors

TokenInterceptor

```
@Injectable({
  providedIn: 'root'
})
export class TokenInterceptor implements HttpInterceptor {
  constructor(private authService: AuthService) { }

  intercept(
      req: HttpRequest<any>,
      next: HttpHandler): Observable<HttpEvent<any>> {
  }
}
```

HttpRequest.clone

```
clone(update: {
    headers?: HttpHeaders;
    reportProgress?: boolean;
    params?: HttpParams;
    responseType?: 'arraybuffer' | 'blob' | 'json' | 'text';
    withCredentials?: boolean;
    body?: T | null;
    method?: string;
    url?: string;
    setHeaders?: {
        [name: string]: string | string[];
    };
    setParams?: {
        [param: string]: string;
    };
}): HttpRequest<T>;
```

Diego Zúñiga

Catch errors

```
return next.handle(req).pipe(catchError(err => {
    if (err.status === 401) {
        ...
    }
    const error = err.error.message || err.statusText;
    return throwError(error);
}))
```

HTTP_INTERCEPTORS

```
providers: [{
    provide: HTTP_INTERCEPTORS,
    useClass: TokenInterceptor,
    multi: true
}],
```

App Initializers

```
export function factoryFn(dep1: Dep1Class) {
  return () => {...};
}
```

```
providers: [{
    provide: APP_INITIALIZER,
    useFactory: factoryFn,
    multi: true,
    deps: [Dep1Service]
}],
```

Diego Zúñiga

rxjs - Subjects

```
const sub: BehaviorSubject<SharedData> =
    new BehaviorSubject({ value: 1});
const obs: Observable<SharedData> = sub.asObservable();
....
obs.subscribe((sd: SharedData) => {
    console.log(sd.value);
});
....
sub.next({ value: 2});
```