Deloitte.

Angular Workshop

Translating a business case into software

Deloitte, October 2019

Nuno Ponte – nponte@deloitte.pt

HTML

HTML
Overview

- HTML tags are element names surrounded by angle brackets
- HTML tags normally come in pairs like and
- The first tag in a pair is the start tag, the second tag is the end tag
- The end tag is written like the start tag, but with a forward slash inserted before the tag name

HTML
Example

```
<!DOCTYPE html>
<html>
    <head>
        <title>Page Title</title>
    </head>
    <body>
        <h1>This is a Heading</h1>
        <p>This is a paragraph.</p>
     </body>
</html>
```

CSS

CSS
Overview

- Cascading Style Sheets (CSS) is a style sheet language used to describe the presentation of a document written in a markup language
- CSS is the technology used by most websites to create visually engaging webpages and user interfaces
- It was initially designed to separate the document content from the document presentation
- Document presentation includes layout, colors and fonts

CSS
SASS

https://sass-lang.com/guide

CSS
Bootstrap

https://getbootstrap.com/docs/4.3/getting-started/introduction/

CSS
Angular Material

https://material.angular.io/components/categories

JavaScript

JavaScript
Overview

- JavaScript is a high level, dynamic and interpreted programming language created in 1995
- JavaScript allows to build interactivity with otherwise static HTML pages
- The majority of websites use it and all the browsers support it without plugins
- Although some naming, syntactic and standard libraries similarity, JavaScript and Java are unrelated and have different semantics

JavaScript
Frameworks

- **Save server traffic:** it’s possible to validate user inputs before sending the page to the server
- **Immediate feedback to users:** don’t have to wait for the page to reload to see if something was forgotten
- **Interactivity:** it’s possible to create interfaces that react when the user hovers with the mouse
- **Interface:** use JavaScript to include items as drag and drop components

JavaScript
Data types

String	A sequence of text known as a string. To signify that the value is a string, you must enclose it in quote marks.	let myVariable = 'Bob';
Number	A number. Numbers don't have quotes around them.	let myVariable = 10;
Boolean	A True/False value. The words true and false are special keywords in JS, and don't need quotes.	let myVariable = true;
Array	A structure that allows you to store multiple values in one single reference.	let myVariable = [1,'Bob','Steve',10]; myVariable[0]; myVariable[1];
Object	Basically, anything. Everything in JavaScript is an object, and can be stored in a variable. Keep this in mind as you learn.	let myVariable = document.querySelector('h1');

JavaScript
Operators

Addition	Used to add two numbers together or glue two strings together.	+	6 + 9; "Hello " + "world!";
Subtraction **Multiplication** **Division**	These do what you'd expect them to do in basic math.	- * /	9 - 3; 8 * 2; 9 / 3;
Assignment	You've seen this already: it assigns a value to a variable.	=	let myVariable = 'Bob';
Equality	Does a test to see if two values are equal to one another and returns a true/false (Boolean) result.	===	let myVariable = 3; myVariable === 4;
Not **Does-not-equal**	Returns the logically opposite value of what it precedes; it turns a true into a false, etc. When it is used alongside the Equality operator, the negation operator tests whether two values are *not* equal.	! !==	!(myVariable === 3); myVariable !== 3;

JavaScript
TypeScript

- TypeScript is a typed superset of JavaScript that compiles to plain JavaScript
- TypeScript is pure object oriented with classes, interfaces and statically typed like C# or Java
- Types enable JavaScript developers to use highly-productive development tools and practices like static checking and code refactoring when developing JavaScript applications
- TypeScript offers support for the latest and evolving JavaScript features, including those from ECMAScript 2015 and future proposals, like asynchronous functions and decorators, to help build robust components

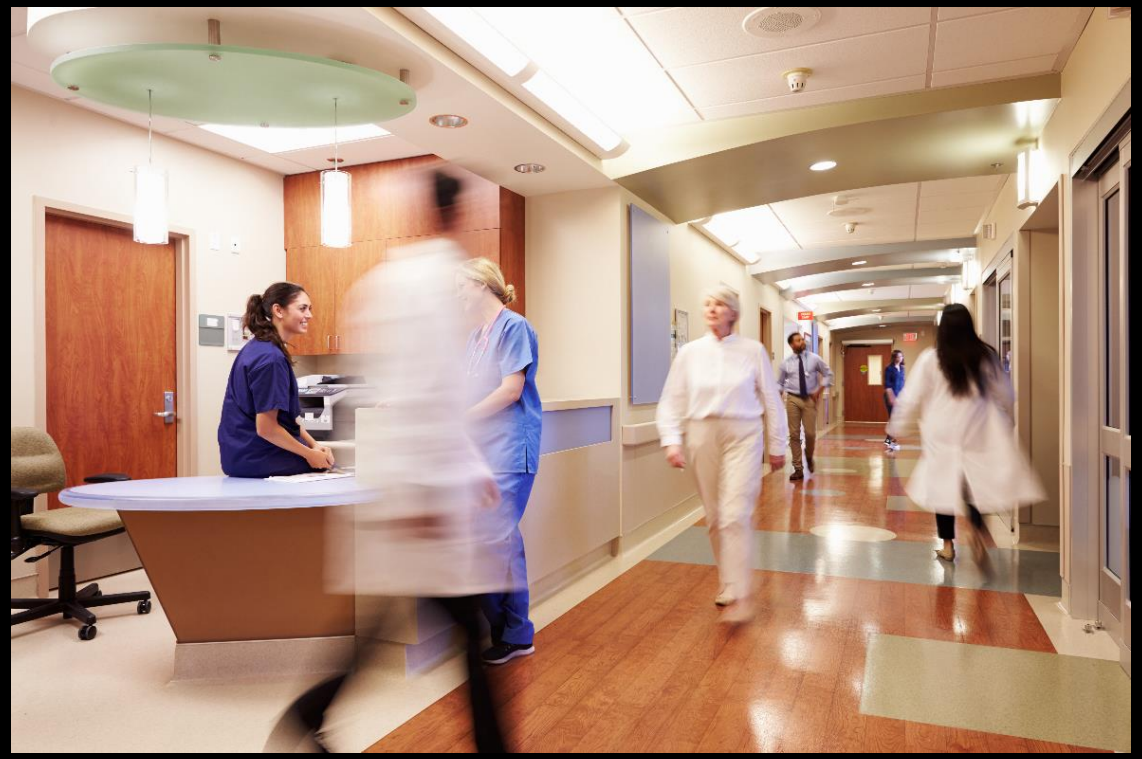
JavaScript
TypeScript

```
class Student {
    fullName: string;
    constructor(public firstName: string, public middleInitial: string, public lastName: string) {
        this.fullName = firstName + " " + middleInitial + " " + lastName;
    }
}

interface Person {
    firstName: string;
    lastName: string;
}

function greeter(person: Person) {
    return "Hello, " + person.firstName + " " + person.lastName;
}

let user = new Student("Jane", "M.", "User");
```

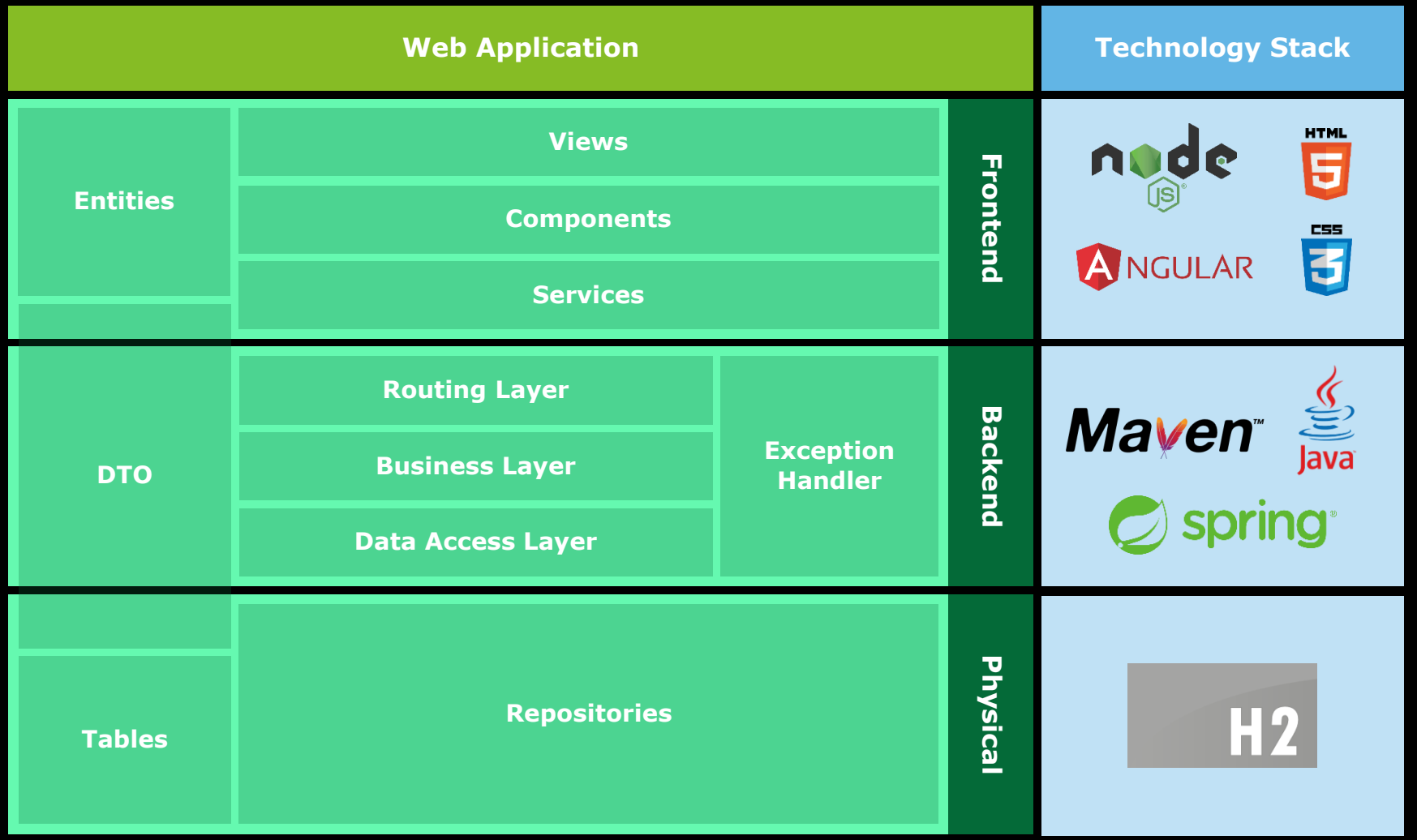
Project Setup

Project Setup
Overview

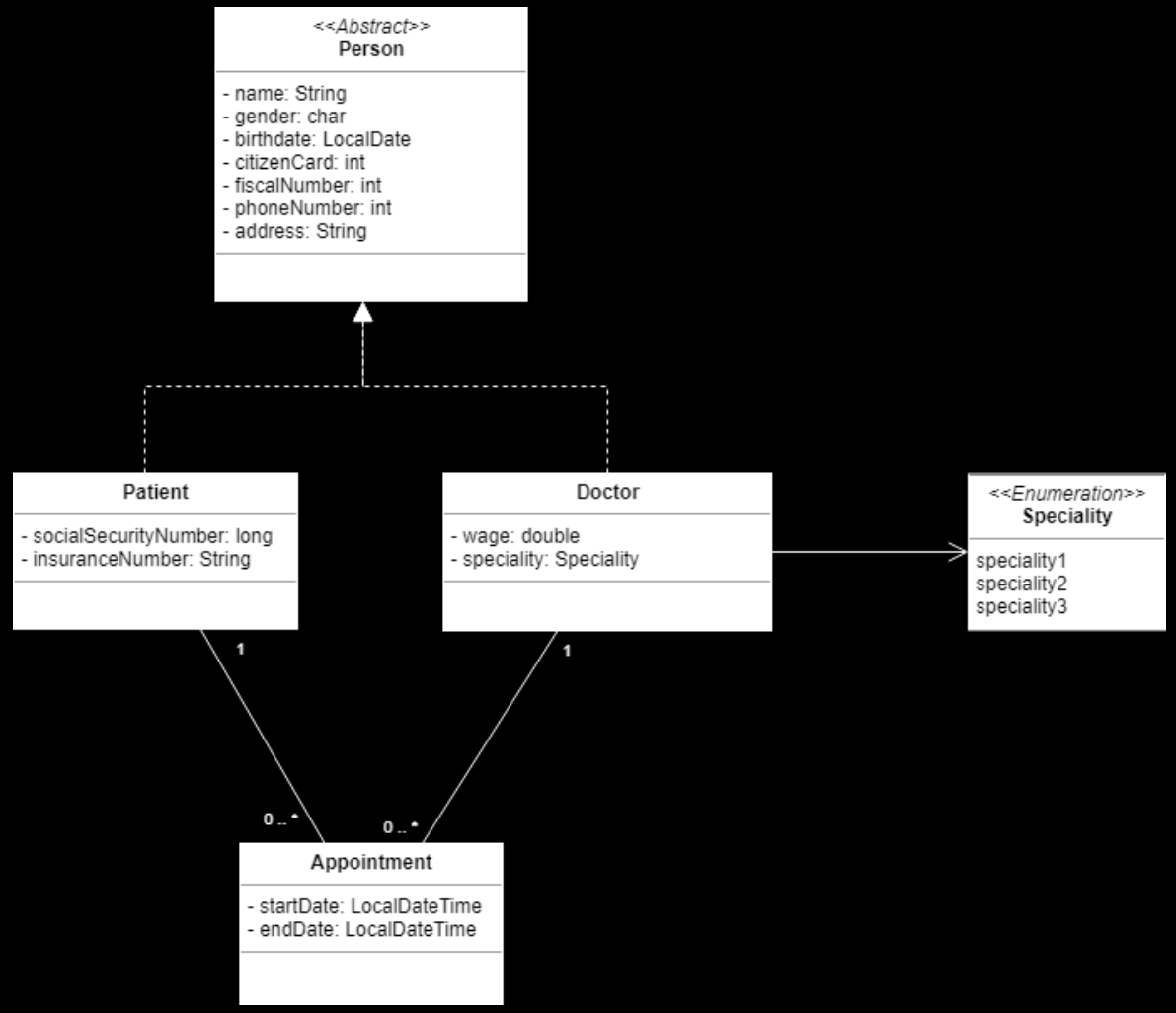
- Client in health business (health clinic)
- Needs to manage appointments of patients with doctors
- Want to have a web application for patients and doctor do manage this information



Project Setup
Architecture



Project Setup
Entities



Project Setup
Software

Windows

1. Install JDK
   A. Add JAVA_HOME variable with value "C:\Program Files\Java\jdk-11"
   B. Add "C:\Program Files\Java\jdk-11\bin" to environment variable "path"
2. Install Maven
   A. Extract zip content to "C:" folder
   B. Add "C:\apache-maven-3.x.x\bin" to environment variable "path"
3. Install Node.js v10.x
4. Install Visual Code
   A. Install the following libraries: Java Extension Pack, Angular Extension Pack (Loiane Groner)

Unix

1. Install JDK
   A. `sudo apt install openjdk-11-jdk-headless`
   B. `sudo apt install openjdk-11-jre-headless`
2. Install Maven
   A. `sudo apt install maven`
3. Install Node.js v10.x
   A. `sudo apt install nodejs`
   B. `sudo apt install npm`
   C. If version 10.x is not installed for NodeJS: https://computingforgeeks.com/installing-node-js-10-lts-on-ubuntu-18-04-16-04-debian-9/
4. Install Visual Code
   A. Download and install the installation file: https://go.microsoft.com/fwlink/?LinkID=760868
   B. Install the following libraries: Java Extension Pack, Angular Extension Pack (Loiane Groner)

Project Setup
Backend

1. Test Java installation: `java -version`
2. Test Maven installation: `mvn -version`
3. Go to GitHub and download project zip from **https://github.com/nfvp/edu-java-health**
4. Open visual code on project folder and wait for libraries do download
5. Build application locally: `mvn spring-boot:run`
6. Test app on **http://localhost:8080**

Project Setup
Frontend

1. Test Node.js installation: `npm --v`
2. Install Angular CLI: `npm install -g @angular/cli`
3. Go to previous folder: `cd ..`
4. Create app: `ng new health-app`
5. Update included libraries: `npm update`
6. Build app locally: `npm start`
7. Test web app on **http://localhost:4200**

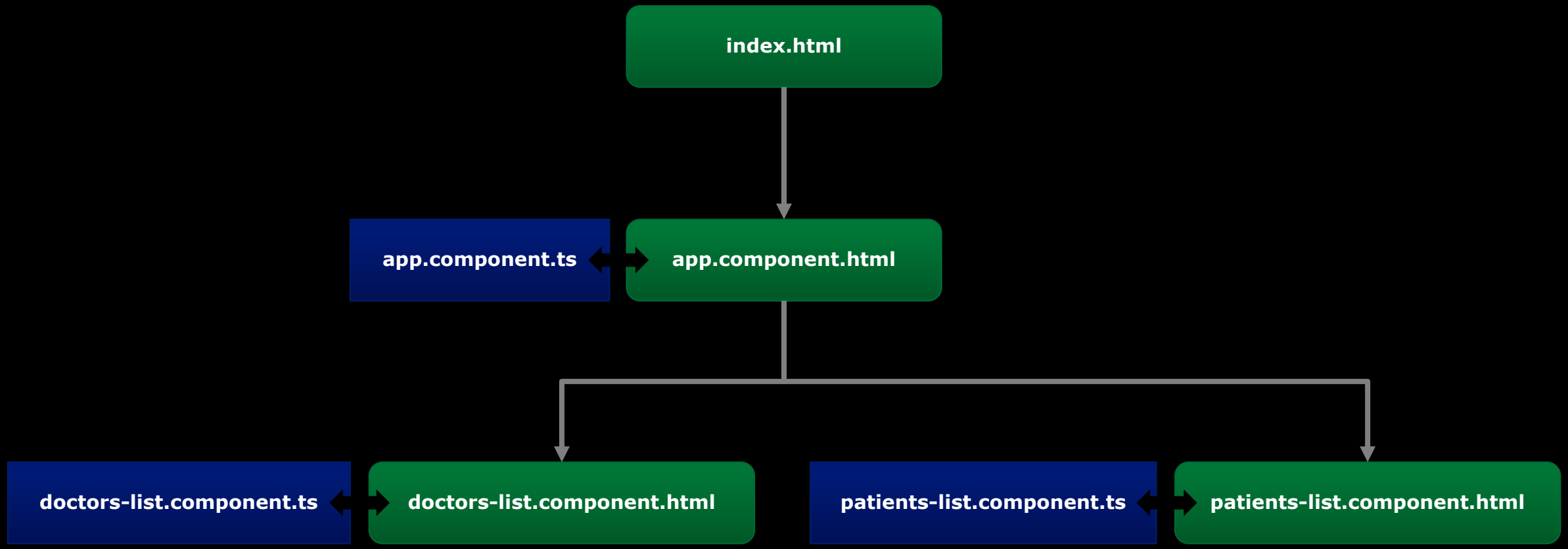
Angular

Angular
Overview

It is a TypeScript-based open source and front-end web application platform

- **Two-way data binding**: Angular was built with Model-View-Controller architecture. And the framework synchronized the Model and the View. As the data in the Model changes, the View does too
- **Directives**: Directives allowed developers to assign special behaviors to the Document Object Model (DOM), permitting engineers to create dynamic and rich content with HTML
- **Dependency injection:** Dependencies define how different pieces of code interact with each other and how the changes in one component impact the other ones. Usually, dependencies are directly defined in the components themselves
- **Community**: Right from the beginning, Angular became extremely popular among engineers. A strong community provided enough training materials, discussions, and third-party tools to embark on using Angular as well as find a solution to nearly every arising issue.

Angular
Structure



Angular
HTML template

```
<tbody>
    <tr *ngFor="let appointment of appointments">
        <td>{{ appointment.startDate | date }}</td>
        <td>{{ appointment.endDate | date }}</td>
        <td>{{ appointment.doctor.name }}</td>
        <td>{{ appointment.patient.name }}</td>
    </tr>
</tbody>
```

Angular
Directives

Structural directives

- Structural directives alter layout by adding, removing, and replacing elements in the DOM. The example template uses two built-in structural directives to add application logic to how the view is rendered.

```
<li *ngFor="let hero of heroes"></li>

<app-hero-detail *ngIf="selectedHero"></app-hero-detail>
```

Attribute directives

- Attribute directives alter the appearance or behavior of an existing element. In templates they look like regular HTML attributes, hence the name.
- The ngModel directive, which implements two-way data binding, is an example of an attribute directive. ngModel modifies the behavior of an existing element (typically <input>) by setting its display value property and responding to change events.

```
<input [(ngModel)]="hero.name">
```

Angular
Pipes

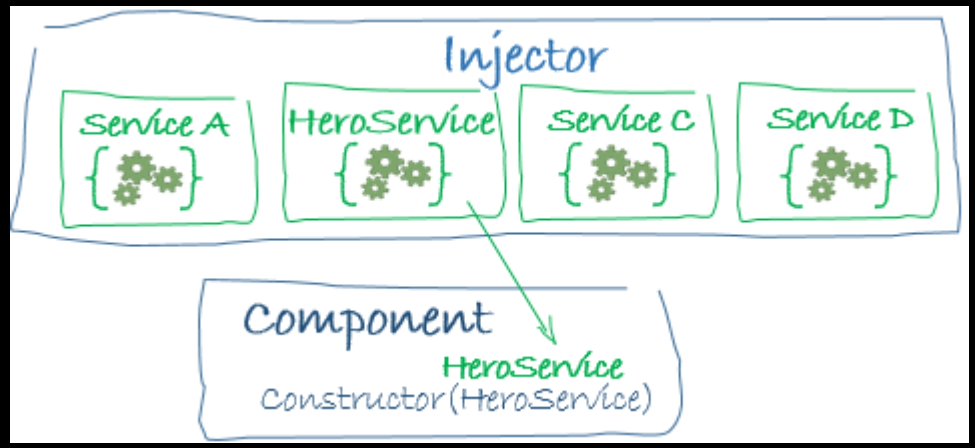
{{interpolated_value | pipe_name}}

You can chain pipes, sending the output of one pipe function to be transformed by another pipe function. A pipe can also take arguments that control how it performs its transformation. For example, you can pass the desired format to the date pipe.

```
<p>Today is {{today | date}}</p>                    <!-- Jun 15, 2015 -->

<p>The date is {{today | date:'fullDate'}}</p>      <!-- Monday, June 15, 2015 -->

<p>The time is {{today | date:'shortTime'}}</p>     <!-- 9:43 AM -->
```

Angular
Components

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

  appointments: Appointment[] = [];

  constructor(service: AppointmentService) {
    service.getAppointmentList().subscribe(
      (res:Appointment[]) => {
                this.appointments = res;
            });
  }
}
```



Angular
Entities

```
export class Appointment {
    startData: Date;
    endDate: Date;
    doctor: Doctor;
    patient: Patient;

    constructor() {
        this.doctor = new Doctor();
        this.patient = new Patient();
    }
}
```

Angular
Services

```
export class AppointmentService {

  private baseUrl: string = "http://localhost:8080/appointments";

  constructor(private httpClient: HttpClient) { }

  getAppointment(id: string) {
    return this.httpClient.get(this.baseUrl + '/${id}');
  }

  getAppointmentList() {
    return this.httpClient.get(this.baseUrl);
  }
}
```

Beautify

Beautify
Bootstrap

Patient	Doctor	Start Date	End Date
Sr. Pedro Silva	Dra. Maria Silva	September 28, 2019 at 9:00:00 AM GMT+1	September 28, 2019 at 10:00:00 AM GMT+1
Sra. Beatriz Silva	Dr. João Silva	September 26, 2019 at 11:00:00 AM GMT+1	September 26, 2019 at 12:00:00 PM GMT+1
Sr. José Silva	Dra. Maria Silva	September 26, 2019 at 1:00:00 PM GMT+1	September 26, 2019 at 2:00:00 PM GMT+1
Sra. Rute Silva	Dra. Joana Silva	September 26, 2019 at 3:30:00 PM GMT+1	September 26, 2019 at 4:30:00 PM GMT+1
Sr. Manuel Silva	Dra. Maria Silva	September 27, 2019 at 9:00:00 AM GMT+1	September 27, 2019 at 10:00:00 AM GMT+1
Sra. Filipa Silva	Dra. Joana Silva	September 27, 2019 at 12:00:00 PM GMT+1	September 27, 2019 at 1:00:00 PM GMT+1
Sr. Joaquim Silva	Dra. Maria Silva	September 27, 2019 at 2:30:00 PM GMT+1	September 27, 2019 at 3:30:00 PM GMT+1
Sra. Sara Silva	Dr. João Silva	September 27, 2019 at 5:00:00 PM GMT+1	September 27, 2019 at 6:00:00 PM GMT+1

Beautify
Angular Material

Name	Telephone	Wage	Speciality
Dr. João Silva	916338296	€30,000.00	FAMILY_MEDICINE
Dra. Joana Silva	916338298	€38,000.00	UROLOGY
Dra. Maria Silva	916338294	€35,000.00	PSYCHIATRY

Deloitte.