

Angular - Getting Started



Angular

- Angular is a framework for building modern single-page applications



Official docs
Tutorials

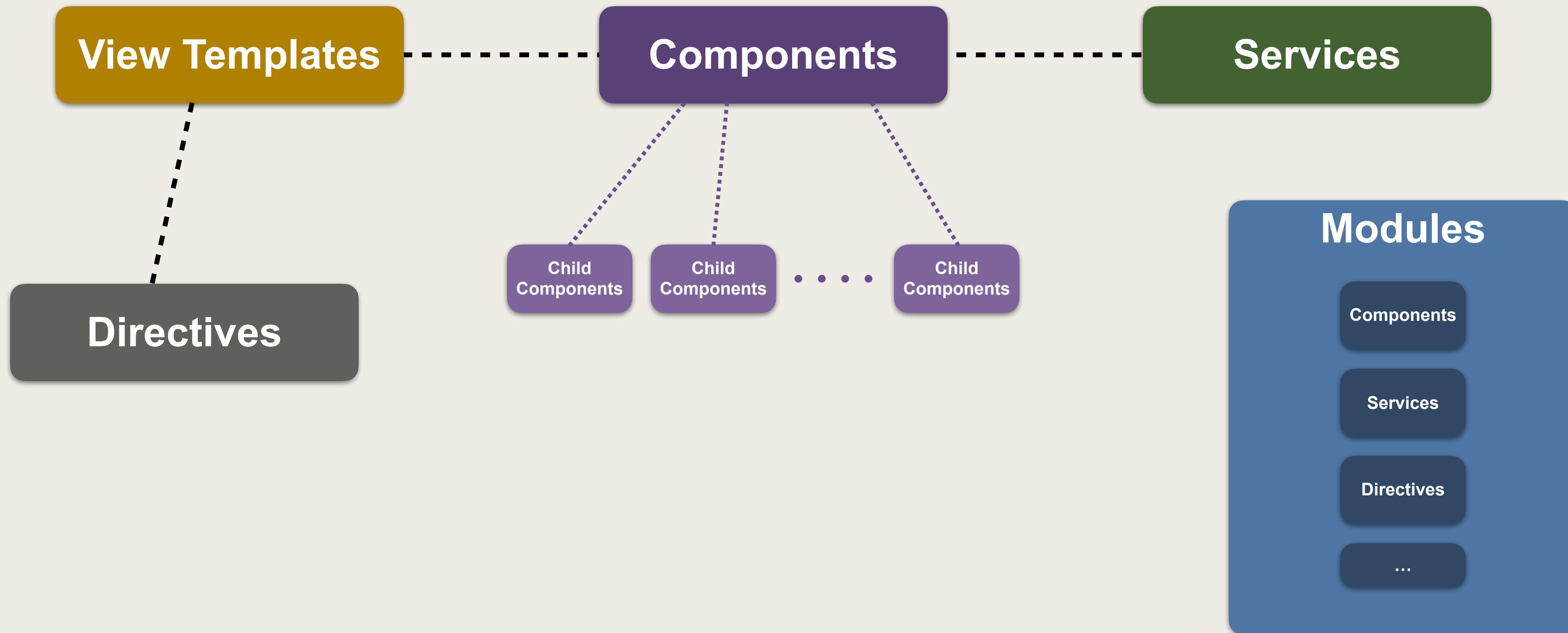
www.angular.io

Angular Features



- Component-based framework
- Clean separation of template coding and application logic
- Built-in support for data-binding and dependency injection
- Supports responsive web design and modern frameworks
 - Bootstrap, Google Material Design (Angular Material) and others ...

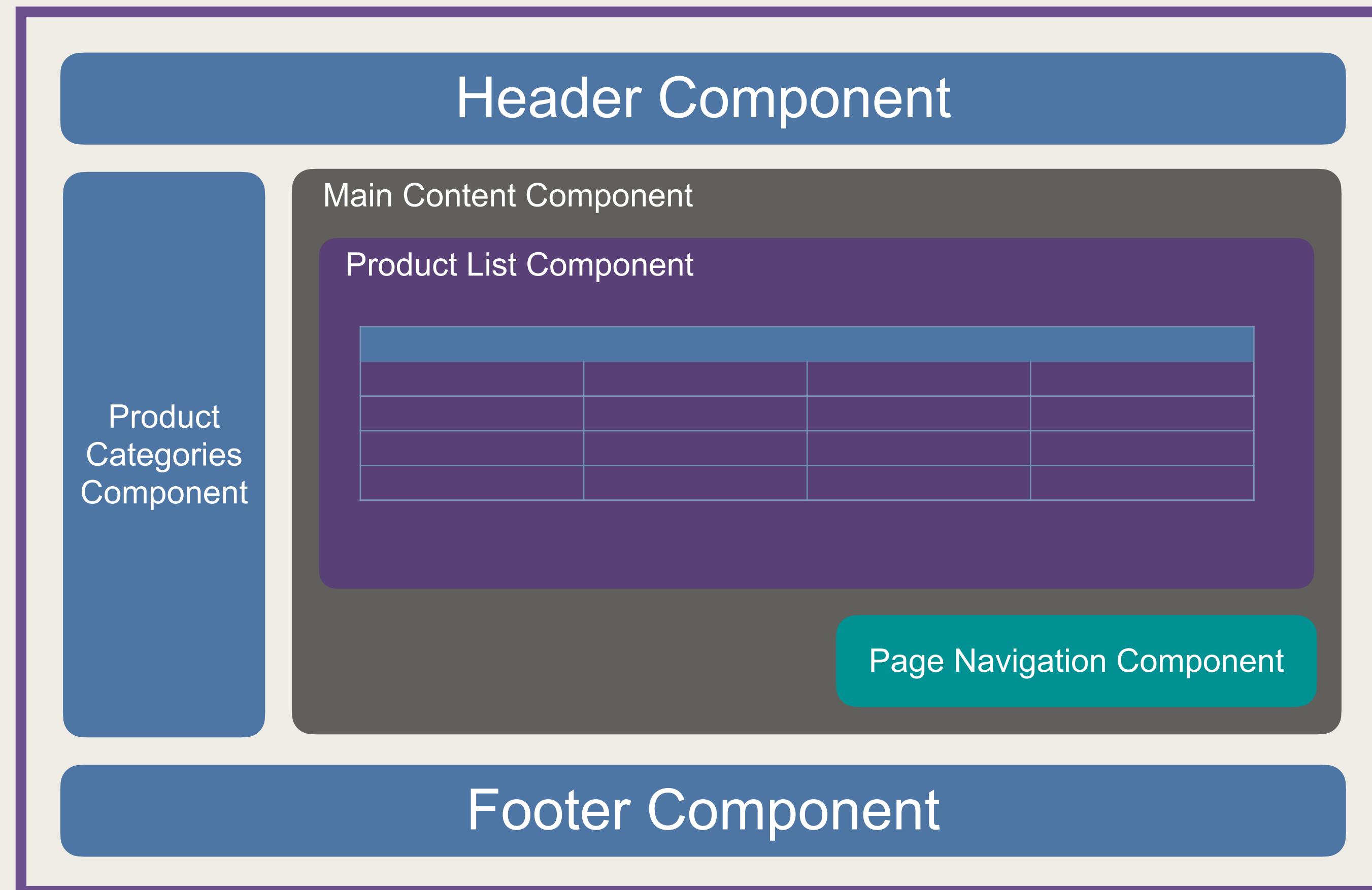
Angular Architecture



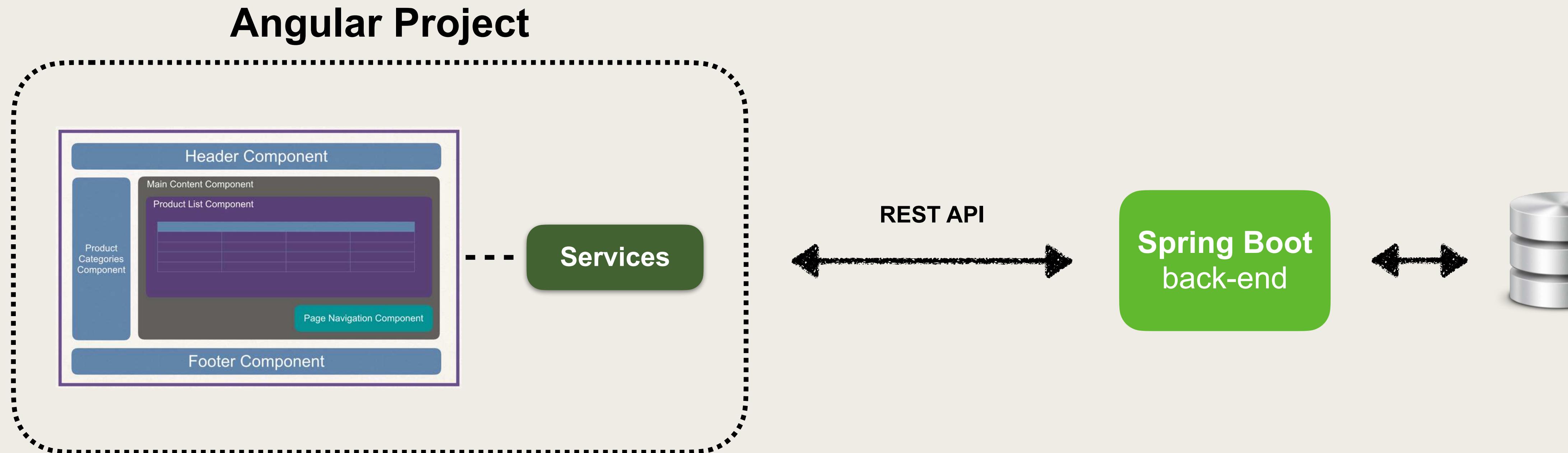
Key Terms

Term	Definition
Component	<p>Main player in an Angular application. Has two parts:</p> <ol style="list-style-type: none">1. View for user interface2. Class that contains application logic / event handling for the view
View Template	<p>The user interface for the component Static HTML with dynamic elements</p>
Directive	<p>Adds custom behavior to HTML elements Used for looping, conditionals etc</p>
Service	<p>Helper class that provides desired functionality Retrieving data from a server, performing a calculation, validation etc</p>
Module	<p>A collection of related components, directives, services etc</p>

Application UI Composition



Application Interaction



Angular Project

- An Angular project is composed of multiple files



Creating an Angular Project

- Angular provides a command-line tool to generate a project
- Generates the starter files to help you bootstrap your project

<http://cli.angular.io>

Angular CLI

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

```
> ng version
```

```
> ng help
```

Creating New Project with Angular CLI

```
> ng new <your-project-name>  
> ng new my-first-angular-project
```

Running the Angular App

```
> cd <your-project>
```

```
> cd my-first-angular
```

```
> ng serve
```

```
> ng serve --open
```

1. Builds the app (compile / transpile)
2. Starts the server
3. Watches the source files
4. Rebuilds the apps when source is updated (hot reload)

By default listens on port 4200

http://

Same as above ...

But also opens a web browser to
http://localhost:4200

Changing the Server Port

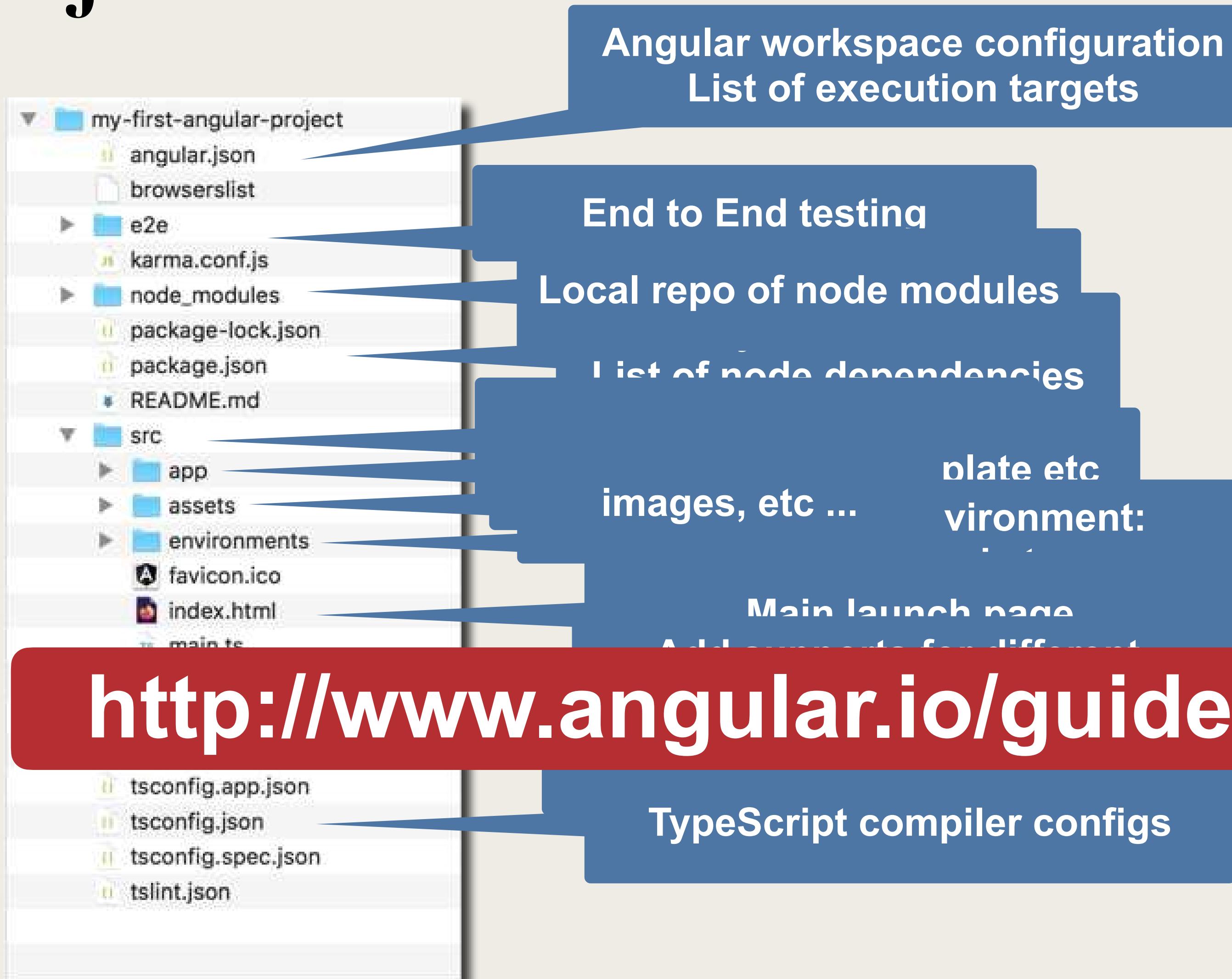
```
> ng serve --port 5100
```

Server listens on port 5100
<http://localhost:5100>

```
> ng serve --port 5100 --open
```

Same as above ...
But also opens a web browser to
<http://localhost:5100>

Project Files



Angular - Behind the Scenes



Angular

- When we run the Angular app ... how is it loaded?



Loading Angular App

File: src/index.html

```
<!doctype html>
<html lang="en">
...
<body>

  <app-root></app-root>

</body>
</html>
```

Loading Angular App

File: src/index.html

```
<!doctype html>
<html lang="en">
...
<body>

  <app-root></app-root>

</body>
</html>
```

Replace this tag / selector
with the template of the component

(similar to an "include")

Loading Angular App

File: src/index.html

```
<!doctype html>
<html lang="en">
...
<body>
  <app-root></app-root> 1
</body>
</html>
```

File: src/app/app.component.ts

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

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'my-first-angular-project';
}
```

@Component is an Angular “Decorator”
Similar to
Annotations in Java

Loading Angular App

File: src/index.html

```
<!doctype html>
<html lang="en">
...
<body>
  <app-root></app-root>
</body>
</html>
```

1

File: src/app/app.component.ts

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

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'my-first-angular-project';
}
```

2

File: src/app/app.component.html

```
<span>{{ title }} app is running! YES YES SUCCESS!!!</span>
```

Loading Angular App

File: src/index.html

```
<!doctype html>
<html lang="en">
...
<body>
  <app-root></app-root>
</body>
</html>
```

1

File: src/app/app.component.ts

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

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'my-first-angular-project';
}
```

2

3

File: src/app/app.component.html

```
<span>{{ title }} app is running! YES YES SUCCESS!!!</span>
```

Loading Angular App

File: src/index.html

```
<!doctype html>
<html lang="en">
...
<body>
    <app-root></app-root>
```

1

File: src/app/app.component.ts

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

@Component({
    selector: 'app-root',
    templateUrl: './app.component.html',
    styleUrls: ['./app.component.css']
})
export class AppComponent {
    title = 'my-first-angular-project';
}
```

2

3

Read this property
from the
TypeScript class

File: src/app/app.component.html

```
<span>{{ title }} app is running! YES YES SUCCESS!!!</span>
```

Loading Angular App

File: src/index.html

```
<!doctype html>
<html lang="en">
...
<body>
  <app-root></app-root>
</body>
```

1

File: src/app/app.component.ts

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

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'my-first-angular-project';
}
```

2

3

MyFirstAngularProject

localhost:4200

Apps

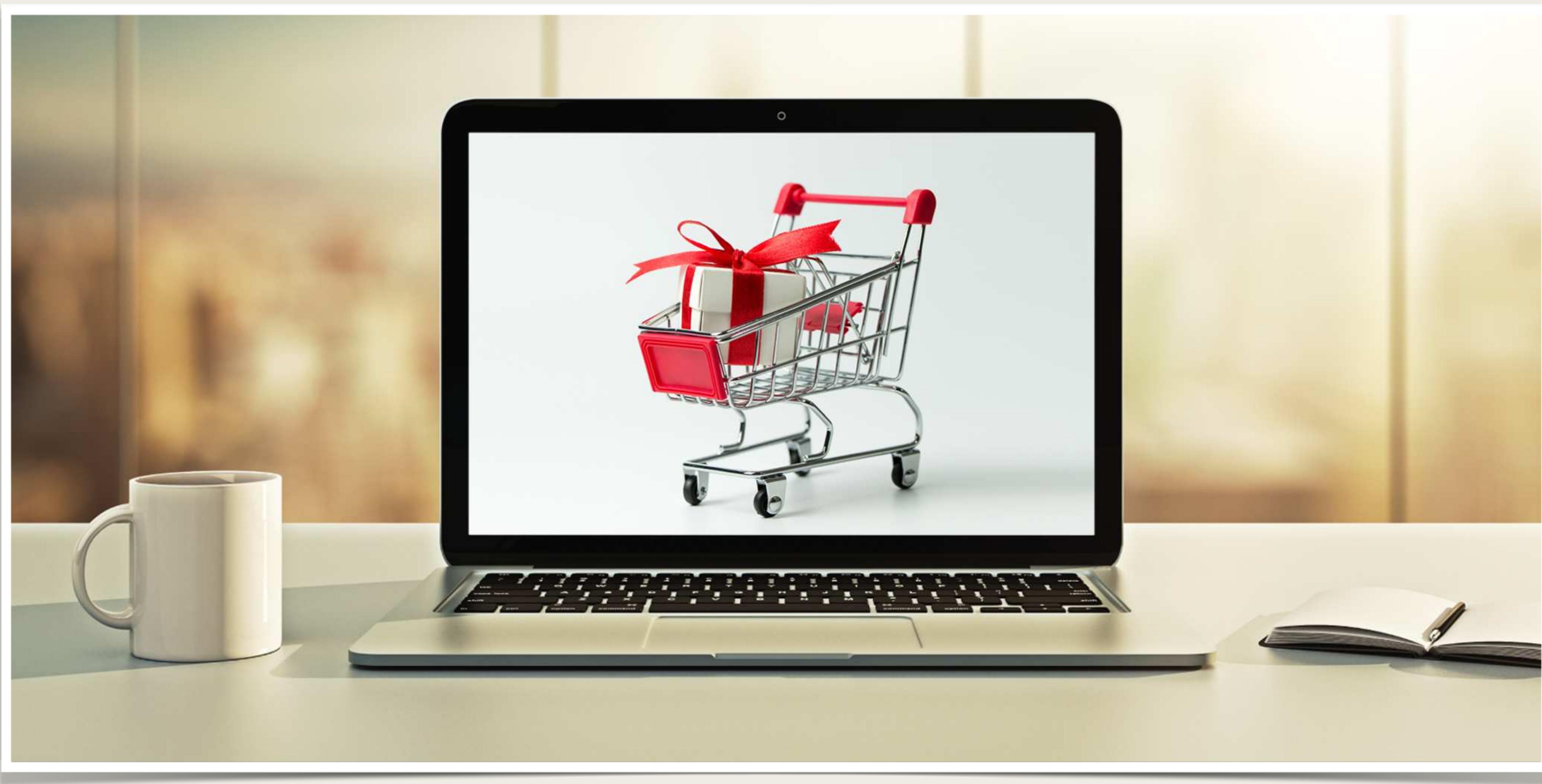
my-first-angular-project app is running! YES YES SUCCESS!!!

4

File: src/app/app.component.html

```
<span>{{ title }} app is running! YES YES SUCCESS!!!</span>
```

Angular - Create New Component



Our Goal

- Create a new Angular component to display a table of data

The diagram illustrates a custom Angular component. A red callout bubble labeled "Custom component" points to a white rectangular box containing a table titled "Sales Team". The table has four columns: First Name, Last Name, Email, and Sales Volume. The data is as follows:

First Name	Last Name	Email	Sales Volume
Anup	Kumar	anup.kumar@luv2code.com	50000
John	Doe	john.doe@luv2code.com	40000
Claire	Murphy	claire.murphy@luv2code.com	90000
Mai	Truong	mai.truong@luv2code.com	60000

Development Process

Step-By-Step

1. Create a new project
2. Update main template page
3. Generate a new component
4. Add new component selector to app template page
5. Generate a SalesPerson class
6. In SalesPersonListComponent, create sample data
7. In sales-person-list template file, build HTML table by looping over data

Step 1: Create a new project

```
> ng new sales-project
```

```
> cd sales-project
```

Remember, this generates all of the
Angular starter files
for our project

Step 2: Update main template page

File: src/app/app.component.html

```
<h1>Sales Team</h1>
```

Remove all of the
Angular "placeholder" content

Just add basic HTML header

Step 3: Generate a new component

```
> ng generate component sales-person-list
```

```
CREATE src/app/sales-person-list/sales-person-list.component.css (0 bytes)
CREATE src/app/sales-person-list/sales-person-list.component.html (32 bytes)
CREATE src/app/sales-person-list/sales-person-list.component.spec.ts (693 bytes)
CREATE src/app/sales-person-list/sales-person-list.component.ts (311 bytes)
UPDATE src/app/app.module.ts (436 bytes)
```

About the Generated Files

`sales-person-list.component.ts`: the component class

`sales-person-list.component.html`: the component template HTML

`sales-person-list.component.css`: the component private CSS

`sales-person-list.component.spec.ts`: the unit test specifications

`UPDATE src/app/app.module.ts`: Adds the component to the main application module

Main Application Module

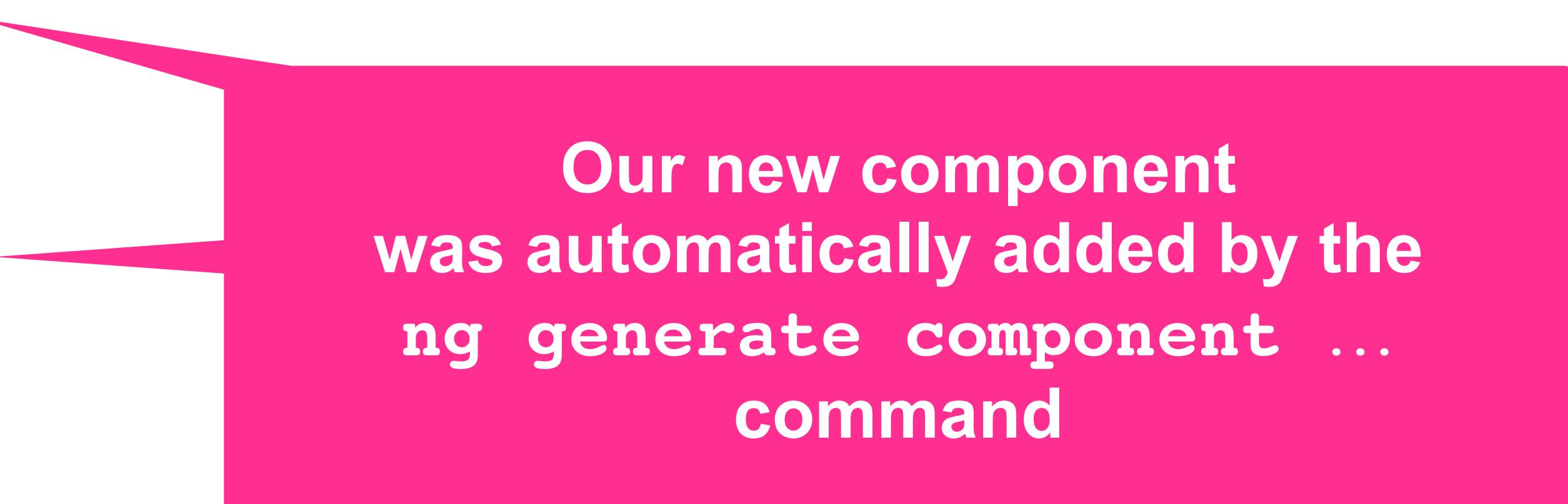
...

UPDATE src/app/app.module.ts: Adds the component to the main application module

...

```
import { AppComponent } from './app.component';
import { SalesPersonListComponent } from './sales-person-list/sales-person-list.component';

@NgModule({
  declarations: [
    AppComponent,
    SalesPersonListComponent
  ],
  ...
})
export class AppModule { }
```



Our new component
was automatically added by the
`ng generate component ...`
command

Step 4: Add new component selector to app template page

File: src/app/app.component.html

```
<h1>Sales Team</h1>  
  
<app-sales-person-list></app-sales-person-list>
```

1

File: src/app/sales-person-list/sales-person-list.component.ts

```
import { Component, OnInit } from '@angular/core';  
  
@Component({  
  selector: 'app-sales-person-list',  
  templateUrl: './sales-person-list.component.html',  
  styleUrls: ['./sales-person-list.component.css']  
})  
export class SalesPersonListComponent implements OnInit {  
  
  constructor() {}  
  
  ngOnInit() {}  
}
```

File: src/app/sales-person-list/sales-person-list.component.html

```
<p>sales-person-list works!</p>
```

2

Later on,
we'll add HTML table here

Later on,
we'll add sample data here

Step 5: Generate a SalesPerson class

```
> ng generate class sales-person-list/SalesPerson
```

File: src/app/sales-person-list/sales-person.ts

```
export class SalesPerson {  
  
    constructor(public firstName: string,  
               public lastName: string,  
               public email: string,  
               public salesVolume: number) {  
  
    }  
}
```

Creates a basic TypeScript class

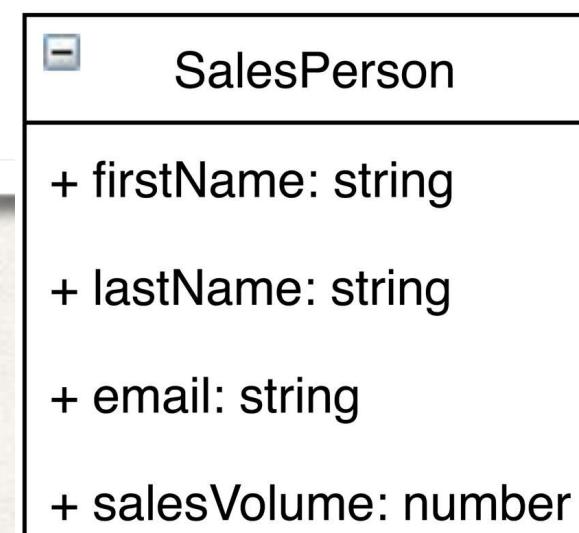
Step 5: Generate a SalesPerson class

```
> ng generate class SalesPerson
```

Remember, these are
Parameter Properties

File: src/app/sales-person-list/sales-person.ts

```
export class SalesPerson {  
  
  constructor(public firstName: string,  
              public lastName: string,  
              public email: string,  
              public salesVolume: number) {  
  
  }  
}
```



Declared by prefixing
constructor argument with access modifier:
public, **protected**, **private**, or **readonly**

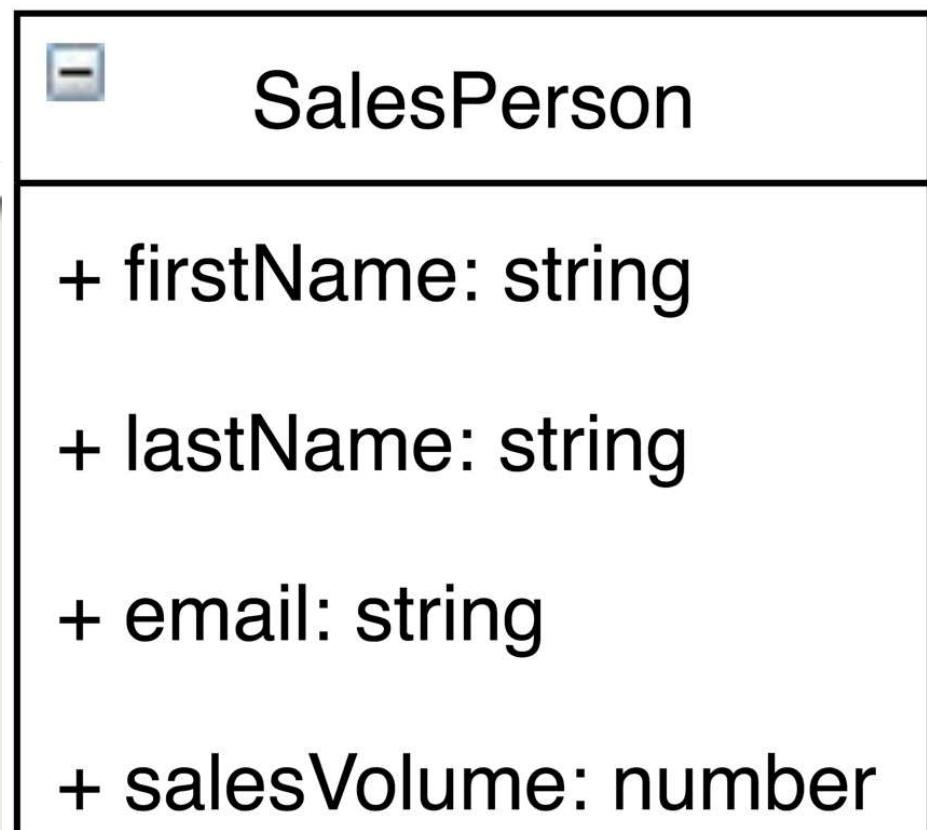
Declares properties and
assigns properties automagically.

Minimizes boilerplate coding!

About "public" properties

File: src/app/sales-person-list/sales-person.ts

```
export class SalesPerson {  
  
  constructor(public firstName: string,  
             public lastName: string,  
             public email: string,  
             public salesVolume: number) {  
  }  
}
```



In Angular world,
developers commonly
use "public" properties

Step 6: In SalesPersonListComponent, create sample data

File: src/app/sales-person-list/sales-person-list.component.ts

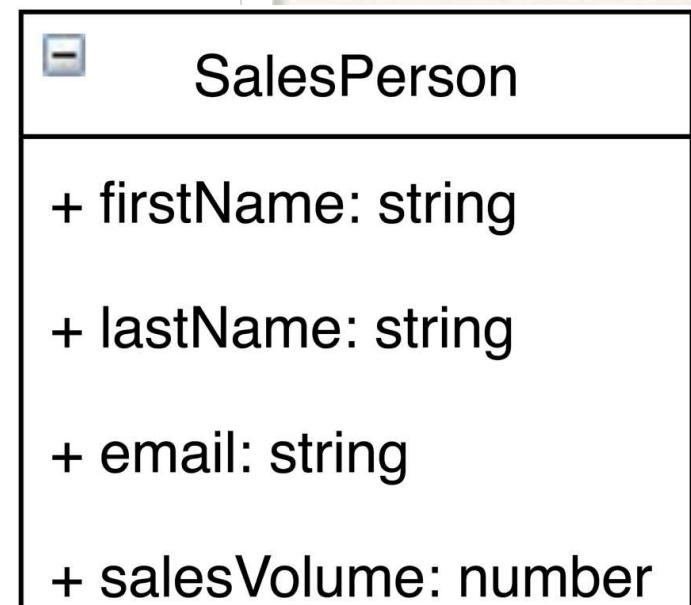
```
import { Component, OnInit } from '@angular/core';
import { SalesPerson } from './sales-person';

@Component({
  selector: 'app-sales-person-list',
  templateUrl: './sales-person-list.component.html',
  styleUrls: ['./sales-person-list.component.css']
})
export class SalesPersonListComponent implements OnInit {

  // create an array of objects
  salesPersonList: SalesPerson[] = [
    new SalesPerson("Anup", "Kumar", "anup.kumar@luv2code.com", 50000),
    new SalesPerson("John", "Doe", "john.doe@luv2code.com", 40000),
    new SalesPerson("Claire", "Murphy", "claire.murphy@luv2code.com", 90000),
    new SalesPerson("Mai", "Truong", "mai.truong@luv2code.com", 60000)
  ]
  ...
}
```

Import our new class

Create sample data



Step 7: In sales-person-list template file, build HTML table by looping over data

File: src/app/sales-person-list/sales-person-list.component.html

```
<table border="1">
  <thead>
    <tr>
      <th>First Name</th>
      <th>Last Name</th>
      <th>Email</th>
      <th>Sales Volume</th>
    </tr>
  </thead>
  <tbody>
    <tr *ngFor="let tempSalesPerson of salesPersonList">
      <td>{{ tempSalesPerson.firstName }}</td>
      <td>{{ tempSalesPerson.lastName }}</td>
      <td>{{ tempSalesPerson.email }}</td>
      <td>{{ tempSalesPerson.salesVolume }}</td>
    </tr>
  </tbody>
</table>
```

```
import { Component, OnInit } from '@angular/core';
import { SalesPerson } from './sales-person';

@Component({
  selector: 'app-sales-person-list',
  templateUrl: './sales-person-list.component.html',
  styleUrls: ['./sales-person-list.component.css']
})
export class SalesPersonListComponent implements OnInit {

  // create an array of objects
  salesPersonList: SalesPerson[] = [
    new SalesPerson("Anup", "Kumar", "anup.kumar@luv2code.com", 50000),
    new SalesPerson("John", "Doe", "john.doe@luv2code.com", 40000),
    new SalesPerson("Claire", "Murphy", "claire.murphy@luv2code.com", 90000),
    new SalesPerson("Mai", "Truong", "mai.truong@luv2code.com", 60000)
  ]
  ...
}
```

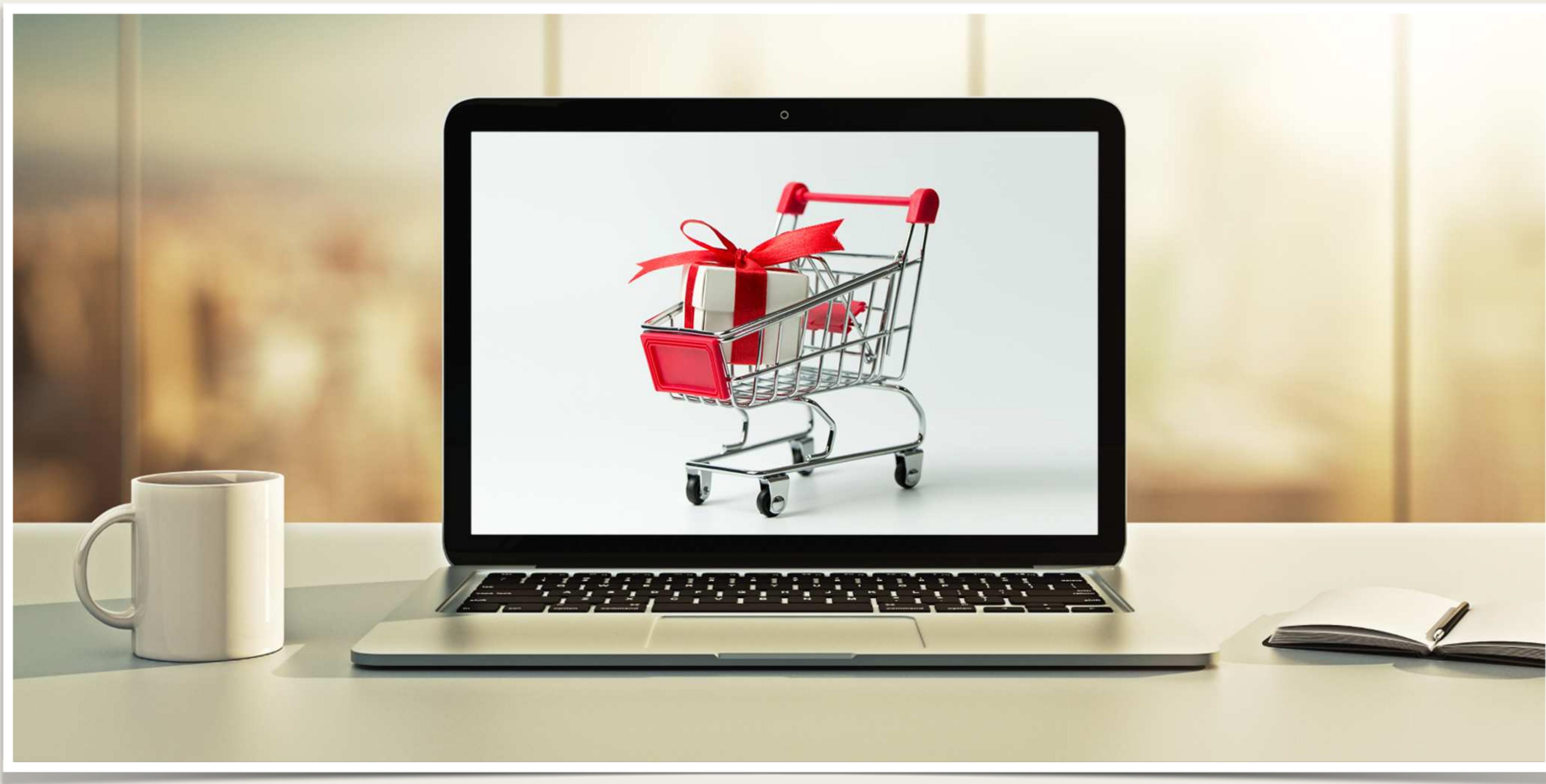
First Name	Last Name	Email	Sales Volume
Anup	Kumar	anup.kumar@luv2code.com	50000
John	Doe	john.doe@luv2code.com	40000
Claire	Murphy	claire.murphy@luv2code.com	90000
Mai	Truong	mai.truong@luv2code.com	60000

ngFor

- **ngFor** is a structural directive
- It renders a template for each item in a collection
- For complete documentation and examples, see:

<http://angular.io/api/common/NgForOf>

Integrate Angular with Bootstrap



Let's Make our Page Beautiful

Before

Sales Team

First Name	Last Name	Email	Sales Volume
Anup	Kumar	anup.kumar@luv2code.com	50000
John	Doe	john.doe@luv2code.com	40000
Claire	Murphy	claire.murphy@luv2code.com	90000
Mai	Truong	mai.truong@luv2code.com	60000

After

Sales Team

First Name	Last Name	Email	Sales Volume
Anup	Kumar	anup.kumar@luv2code.com	50000
John	Doe	john.doe@luv2code.com	40000
Claire	Murphy	claire.murphy@luv2code.com	90000
Mai	Truong	mai.truong@luv2code.com	60000

Bootstrap

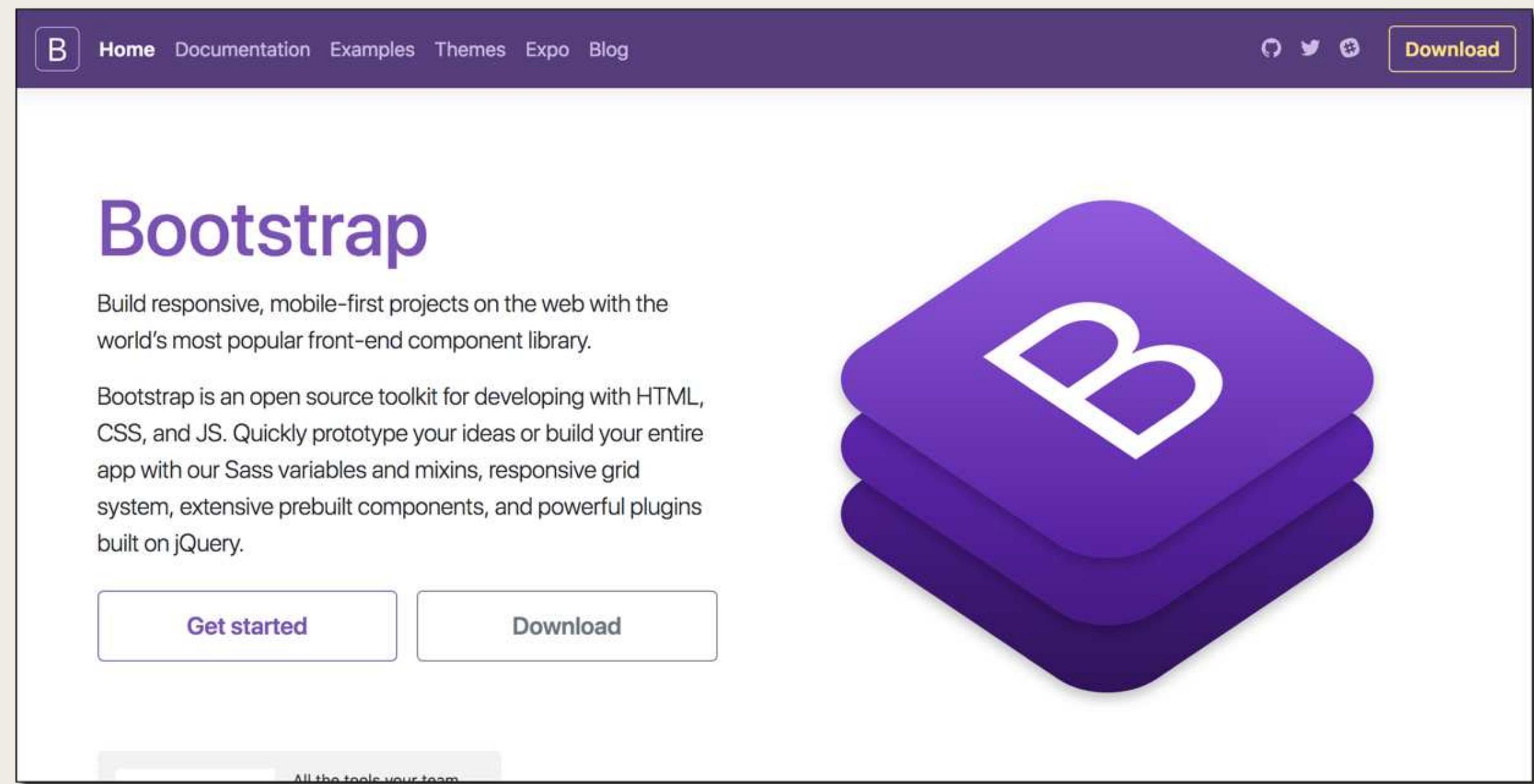
Development Process

Step-By-Step

1. Get links for remote Bootstrap files
2. Add links to index.html
3. Apply Bootstrap CSS styles in component HTML template
4. Apply Bootstrap CSS styles in component HTML table
5. Update TypeScript component file to reference Bootstrap HTML template

Step 1: Get links for remote Bootstrap files

- Visit Bootstrap website: **www.getbootstrap.com**
- Website has instructions on how to **Get Started**



Step 2: Add links in index.html

```
<head>
...
<!-- Bootstrap CSS -->
<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/x.y.z/css/bootstrap.min.css" ...>
...
</head>
```

Version number

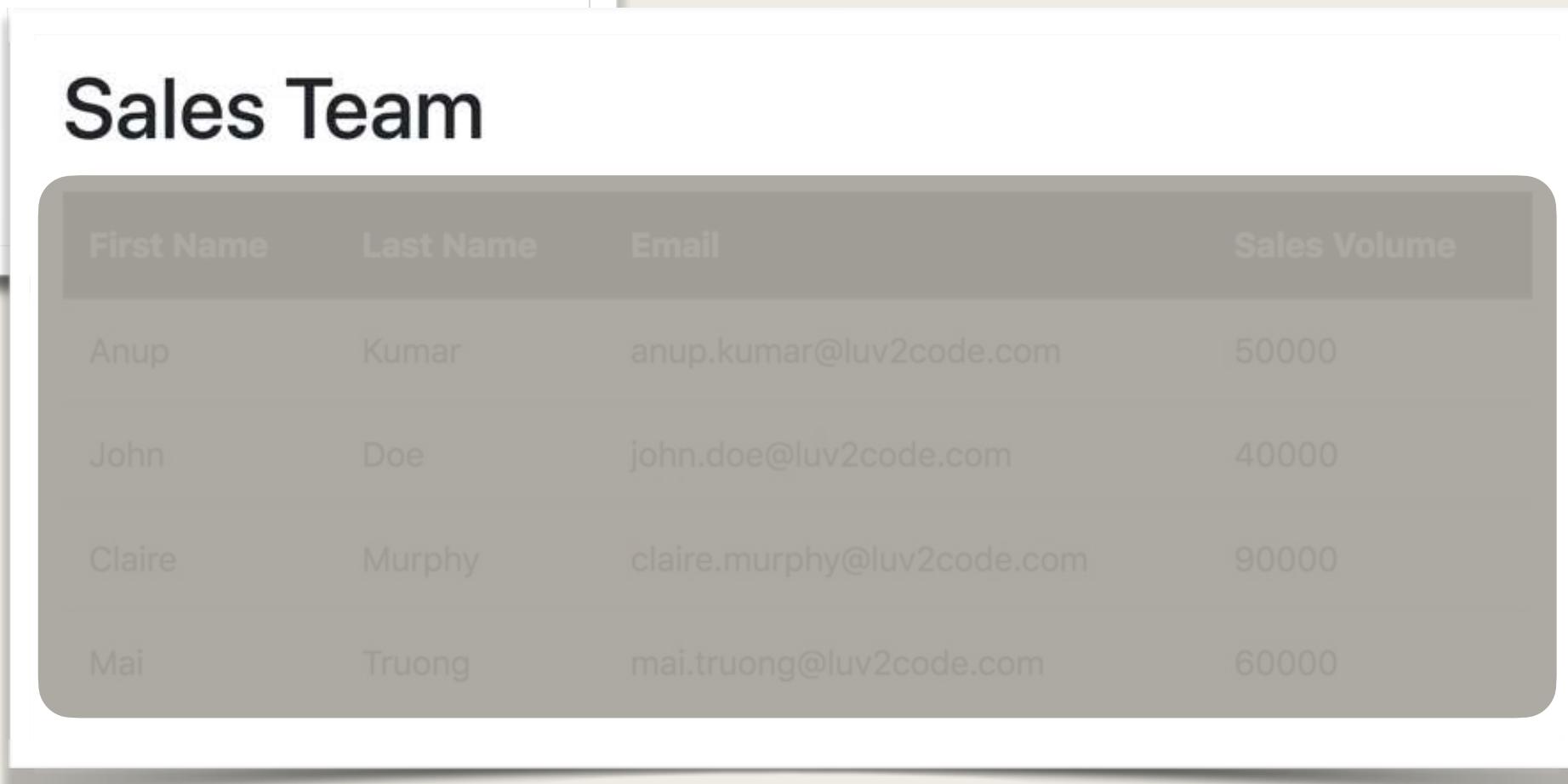
Step 3: Apply CSS in Component HTML template

File: src/app/app.component.html

```
<div class="container">  
  <h1 class="mt-3 mb-3">Sales Team</h1>  
  
  <app-sales-person-list></app-sales-person-list>  
  
</div>
```

Bootstrap CSS style

Bootstrap CSS styles



First Name	Last Name	Email	Sales Volume
Anup	Kumar	anup.kumar@luv2code.com	50000
John	Doe	john.doe@luv2code.com	40000
Claire	Murphy	claire.murphy@luv2code.com	90000
Mai	Truong	mai.truong@luv2code.com	60000

Step 4: Apply CSS in Component HTML Table

File: src/app/sales-person-list/sales-person-list-bootstrap.component.html

```
<table class="table table-hover">  
  
  <thead class="thead-dark">  
    <tr>  
      <th>First Name</th>  
      <th>Last Name</th>  
      <th>Email</th>  
      <th>Sales Volume</th>  
    </tr>  
  </thead>  
  
  <tbody>  
    ...  
  </tbody>  
</table>
```

Bootstrap CSS styles

First Name	Last Name	Email	Sales Volume
Anup	Kumar	anup.kumar@luv2code.com	50000
John	Doe	john.doe@luv2code.com	40000
Claire	Murphy	claire.murphy@luv2code.com	90000
Mai	Truong	mai.truong@luv2code.com	60000

Step 5: Update TypeScript component file

File: src/app/sales-person-list/sales-person-list.component.ts

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

@Component({
  selector: 'app-sales-person-list',
  templateUrl: './sales-person-list-bootstrap.component.html',
  styleUrls: ['./sales-person-list.component.css']
})
export class SalesPersonListComponent implements OnInit {

  ...
}

}
```

First Name	Last Name	Email	Sales Volume
Anup	Kumar	anup.kumar@luv2code.com	50000
John	Doe	john.doe@luv2code.com	40000
Claire	Murphy	claire.murphy@luv2code.com	90000
Mai	Truong	mai.truong@luv2code.com	60000

File: src/app/sales-person-list/sales-person-list-bootstrap.component.html

```
<table class="table table-hover">

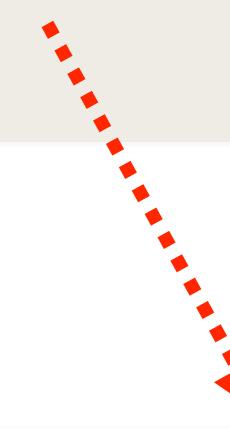
  <thead class="thead-dark">
    <tr>
      <th>First Name</th>
      <th>Last Name</th>
      <th>Email</th>
      <th>Sales Volume</th>
    </tr>
  </thead>
```

Angular: Conditionals and Formatting



Goal

- Add a new column to check if a sales person has met their quota



First Name	Last Name	Email	Sales Volume	Met Quota?
Anup	Kumar	anup.kumar@luv2code.com	50000	No
John	Doe	john.doe@luv2code.com	40000	No
Claire	Murphy	claire.murphy@luv2code.com	90000	Yes
Mai	Truong	mai.truong@luv2code.com	60000	Yes

Conditional: ngIf

- Angular provides a structural directive: ngIf
- Show content based on a condition / boolean expression

```
<div *ngIf="some boolean expression">
```

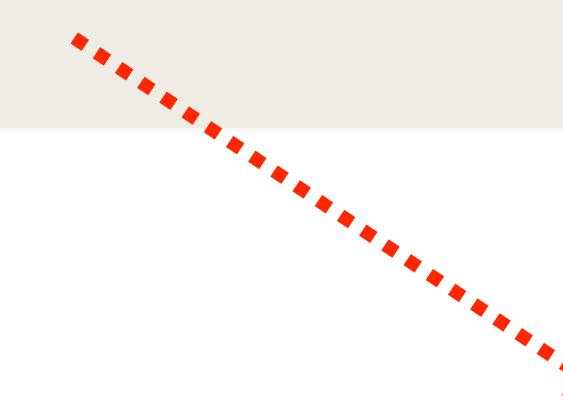
Show this content if the boolean expression is true

```
</div>
```

<https://angular.io/api/common/NgIf>

Example

- Our example: if salesVolume ≥ 60000 , display Yes ... else No



First Name	Last Name	Email	Sales Volume	Met Quota?
Anup	Kumar	anup.kumar@luv2code.com	50000	No
John	Doe	john.doe@luv2code.com	40000	No
Claire	Murphy	claire.murphy@luv2code.com	90000	Yes
Mai	Truong	mai.truong@luv2code.com	60000	Yes

Conditional with ngIf

```
<table class="table table-hover">
  <thead class="thead-dark">
    <tr>
      ...
      <th>Sales Volume</th>
      <th>Met Quota?</th>
    </tr>
  </thead>

  <tbody>
    <tr *ngFor="let tempSalesPerson of salesPerson"
      ...
      <td>{{ tempSalesPerson.salesVolume }}</td>
      <td>
        <div *ngIf="tempSalesPerson.salesVolume >= 60000; else myElseBlock">Yes</div>
        <ng-template #myElseBlock>No</ng-template>
      </td>
    </tr>
  </tbody>
</table>
```

Sales Team				Met Quota?
First Name	Last Name	Email	Sales Volume	
Anup	Kumar	anup.kumar@luv2code.com	50000	No
John	Doe	john.doe@luv2code.com	40000	No
Claire	Murphy	claire.murphy@luv2code.com	90000	Yes
Mai	Truong	mai.truong@luv2code.com	60000	Yes

Conditional

Case of false

Case of true

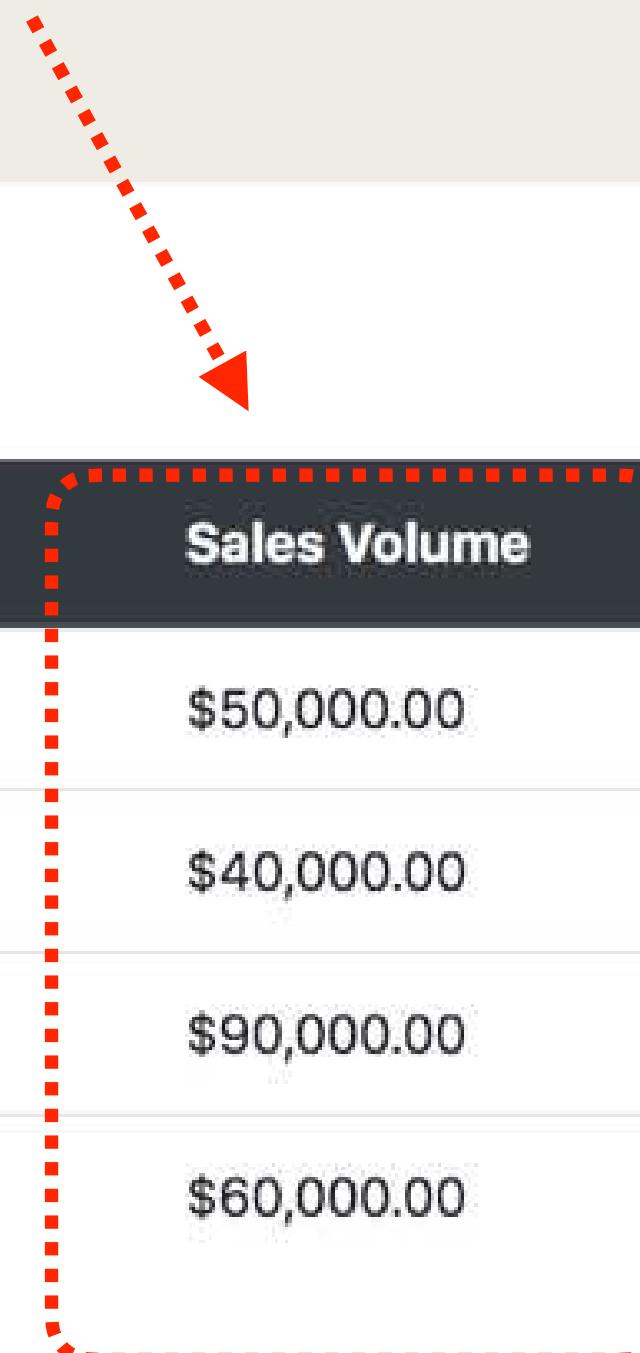
Angular Structural Directives

- Most commonly used structural directives: `ngFor` and `ngIf`
- Also there other such as `ngSwitch`, `ngStyle` etc ...
- See the documentation for details

<https://angular.io/api/common#directives>

Goal

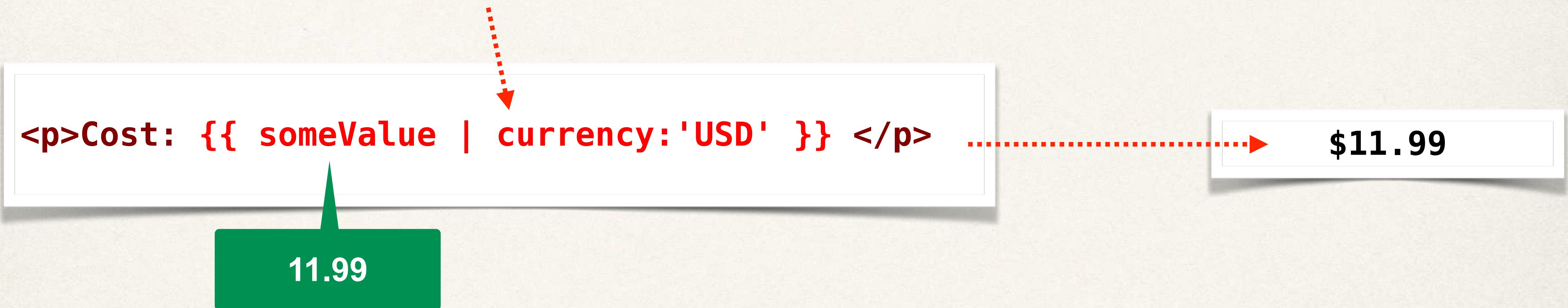
- Format the Sales Volume as a currency, U.S. Dollars



First Name	Last Name	Email	Sales Volume	Met Quota?
Anup	Kumar	anup.kumar@luv2code.com	\$50,000.00	No
John	Doe	john.doe@luv2code.com	\$40,000.00	No
Claire	Murphy	claire.murphy@luv2code.com	\$90,000.00	Yes
Mai	Truong	mai.truong@luv2code.com	\$60,000.00	Yes

Currency Formatting

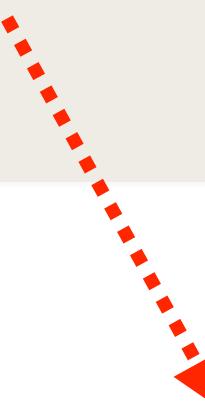
- Angular provides currency formatting using *Angular Pipes*
- You send data to a pipe for formatting



<https://angular.io/api/common/CurrencyPipe>

Example

- Our example: format sales volume as currency, U.S. Dollars



First Name	Last Name	Email	Sales Volume	Met Quota?
Anup	Kumar	anup.kumar@luv2code.com	\$50,000.00	No
John	Doe	john.doe@luv2code.com	\$40,000.00	No
Claire	Murphy	claire.murphy@luv2code.com	\$90,000.00	Yes
Mai	Truong	mai.truong@luv2code.com	\$60,000.00	Yes

Formatting with Currency Pipe

```
<table class="table table-hover">
  ...
  <tbody>
    <tr *ngFor="let tempSalesPerson of salesPersonList">
      ...
      <td>{{ tempSalesPerson.salesVolume | currency:'USD' }}</td>
      ...
    </tr>
  </tbody>
</table>
```

Formatting

First Name	Last Name	Email	Sales Volume	Met Quota?
Anup	Kumar	anup.kumar@luv2code.com	\$50,000.00	No
John	Doe	john.doe@luv2code.com	\$40,000.00	No
Claire	Murphy	claire.murphy@luv2code.com	\$90,000.00	Yes
Mai	Truong	mai.truong@luv2code.com	\$60,000.00	Yes

Angular Pipes

- Other pipes available such as DatePipe, DecimalPipe, etc ...
- See the documentation for details

<https://angular.io/api/common#pipes>