

Setting to make auto save and deploy when we make any change to a file in our project

The screenshot shows the Visual Studio Code interface with the settings.json file open in the workspace settings editor. The code snippet below is highlighted in the settings.json file:

```
// Controls auto save of dirty files.  
// Accepted values: 'off', 'afterDelay',  
// 'onFocusChange' (editor loses focus),  
// 'onWindowChange' (window loses focus).  
// If set to 'afterDelay', you can  
// configure the delay in  
// 'files.autoSaveDelay'.  
// "files.autoSave": "off",  
  
// Controls the font size in pixels.  
// "editor.fontSize": 14,  
  
// Controls the font family.  
// "editor.fontFamily": "Consolas,  
// 'Courier New', monospace",  
  
// The number of spaces a tab is equal
```

Difference Between Angular Module and ES 2015 Module

The screenshot shows the Visual Studio Code interface with the settings.json file open in the workspace settings editor. The code snippet below is highlighted in the settings.json file:

```
// Controls auto save of dirty files.  
// Accepted values: 'off', 'afterDelay',  
// 'onFocusChange' (editor loses focus),  
// 'onWindowChange' (window loses focus).  
// If set to 'afterDelay', you can  
// configure the delay in  
// 'files.autoSaveDelay'.  
// "files.autoSave": "off",  
  
// Controls the font size in pixels.  
// "editor.fontSize": 14,  
  
// Controls the font family.  
// "editor.fontFamily": "Consolas,  
// 'Courier New', monospace",  
  
// The number of spaces a tab is equal
```

What is Component?

What Is a Component?

```

    graph LR
        A[Component] ==> B[Template]
        B ==> C[Class  
Properties  
Methods]
        C ==> D[Metadata]
    
```

Component

- View layout
- Created with HTML
- Includes binding and directives

Template

- Code supporting the view
- Created with TypeScript
- Properties: data
- Methods: logic

Class
Properties
Methods

- Extra data for Angular
- Defined with a decorator

Metadata

Table of Contents

- Course Overview
- Introduction
- First Things First
- Introduction to Components
 - Introduction
 - What is a Component? **1m 59s**
 - Creating the Component Class **2m 17s**
 - Defining the Metadata with a D... **2m 32s**
 - Importing What We Need **2m 8s**
 - Demo: Creating the App Compo... **4m 5s**
 - Bootstrapping Our App Comp... **4m 22s**
 - Demo: Bootstrapping Our App... **3m 49s**
 - Checklists and Summary **5m 15s**
- Templates, Interpolation, and Directives **33m 6s**
- Data Binding & Pipes **21m 54s**
- More on Components **31m 5s**
- Building Nested

Since the class has export something, it becomes ES Module

Creating the Component Class

app.component.ts

```

export class AppComponent {
    pageTitle: string = 'Acme Product Management';
}
    
```

class keyword

Component Name

export keyword

Component Name when used in code

Table of Contents

- Course Overview
- Introduction
- First Things First
- Introduction to Components
 - Introduction
 - What is a Component? **1m 59s**
 - Creating the Component Class **2m 17s****
 - Defining the Metadata with a D... **2m 32s**
 - Importing What We Need **2m 8s**
 - Demo: Creating the App Compo... **4m 5s**
 - Bootstrapping Our App Comp... **4m 22s**
 - Demo: Bootstrapping Our App... **3m 49s**
 - Checklists and Summary **5m 15s**
- Templates, Interpolation, and Directives **33m 6s**
- Data Binding & Pipes **21m 54s**
- More on Components **31m 5s**
- Building Nested

What is Decorator?

Activities Google Chrome ▾

Tue 23:57

https://app.pluralsight.com/player?course=angular-2-getting-started-update&author=deborah-kurata&name=angular-2-getting-started-update-m3&clip=38&mode=live

Angular: Getting Started

Introduction to Components : Defining the Metadata with a Decorator

Decorator

A function that adds **metadata** to a class, its members, or its method arguments.

Prefixed with an @.

Angular provides built-in decorators.

@Component()

Table of Contents

- Course Overview
- Introduction
- First Things First
- Introduction to Components
 - Introduction
 - What Is a Component?
 - Creating the Component Class
 - Defining the Metadata with a D...
 - Importing What We Need
 - Demo: Creating the App Compo...
 - Bootstrapping Our App Comp...
 - Demo: Bootstrapping Our App...
 - Checklists and Summary
- Templates, Interpolation, and Directives
- Data Binding & Pipes
- More on Components
- Building Nested

ADD NOTE

0:48 / 2:32

CC 1.0x

Speaker Volume

Share

Activities Google Chrome ▾

Wed 00:02

https://app.pluralsight.com/player?course=angular-2-getting-started-update&author=deborah-kurata&name=angular-2-getting-started-update-m3&clip=48&mode=live

Importing What We Need

```
app.component.ts
import { Component } from '@angular/core';

@Component({
  selector: 'pm-root',
  template:
    <div><h1>{{pageTitle}}</h1>
      <div>My First Component</div>
    </div>
})

export class AppComponent {
  pageTitle: string = 'Acme Product Management';
}
```

import keyword

Angular library module name

Member name

Table of Contents

- Course Overview
- Introduction
- First Things First
- Introduction to Components
 - Introduction
 - What Is a Component?
 - Creating the Component Class
 - Defining the Metadata with a D...
 - Importing What We Need
 - Demo: Creating the App Compo...
 - Bootstrapping Our App Comp...
 - Demo: Bootstrapping Our App...
 - Checklists and Summary
- Templates, Interpolation, and Directives
- Data Binding & Pipes
- More on Components
- Building Nested

Single Page Application

Activities Google Chrome ▾

Wed 00:21

https://app.pluralsight.com/player?course=angular-2-getting-started-update&author=deborah-kurata&name=angular-2-getting-started-update-m3&clip=6&mode=live

Single Page Application (SPA)



index.html contains the main page for the application

This is often the only Web page of the application

Hence an Angular application is often called a Single Page Application (SPA)

Table of Contents Notes

- Course Overview 0 1m 40s
- Introduction 0 10m 45s
- First Things First 0 22m 27s
- Introduction to Components 0 27m 56s
 - Introduction 1m 26s
 - What is a Component? 1m 59s
 - Creating the Component Class 2m 17s
 - Defining the Metadata with a D... 2m 32s
 - Importing What We Need 2m 8s
 - Demo: Creating the App Compo... 4m 5s
 - Bootstrapping Our App Comp... 4m 22s
 - Demo: Bootstrapping Our App... 3m 49s
 - Checklists and Summary 5m 15s
- Templates, Interpolation, and Directives 0 33m 6s
- Data Binding & Pipes 0 21m 54s
- More on Components 0 31m 5s

Building Nested

Directives-> Here <pm-root></pm-root> is our directive

Activities Google Chrome ▾

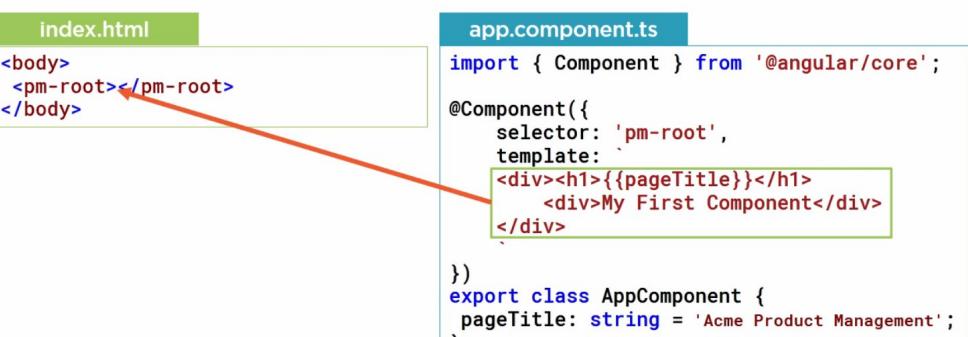
Wed 00:22

https://app.pluralsight.com/player?course=angular-2-getting-started-update&author=deborah-kurata&name=angular-2-getting-started-update-m3&clip=6&mode=live

Angular: Getting Started

Introduction to Components - Bootstrapping Our App Component

Hosting the Application



index.html

```
<body>
<pm-root></pm-root>
</body>
```

app.component.ts

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

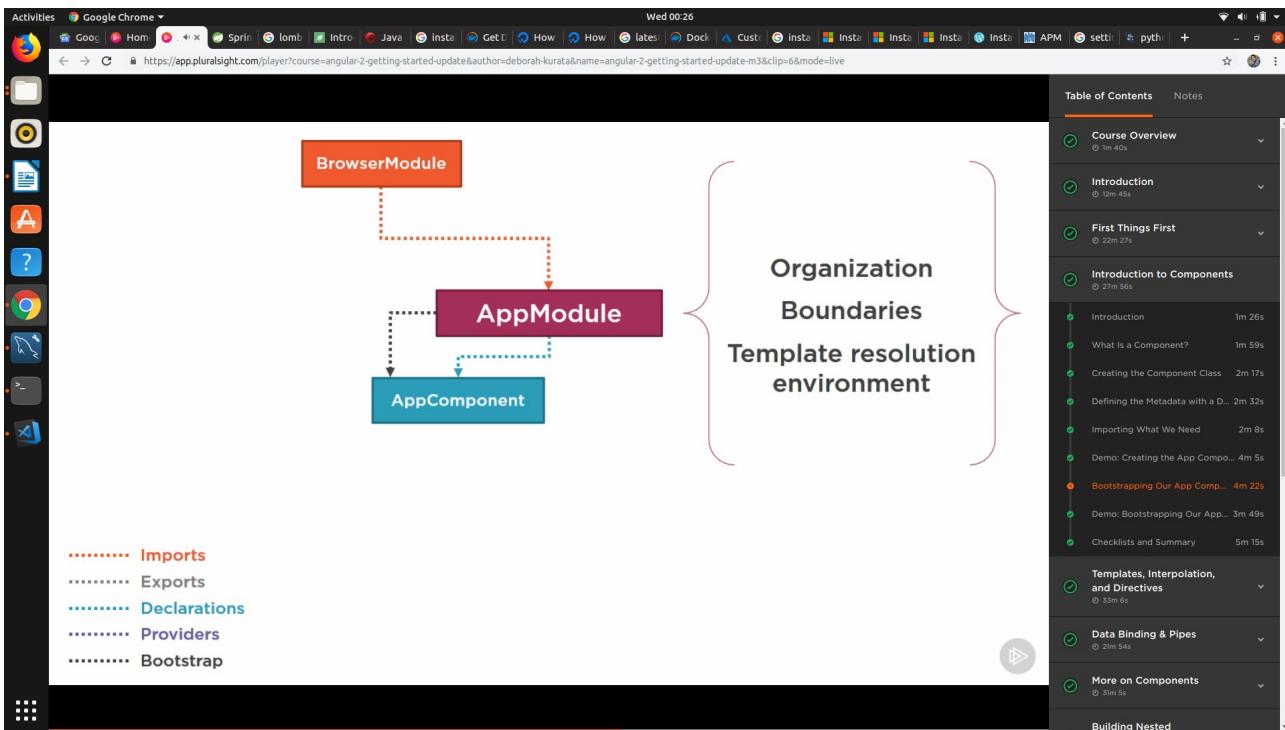
@Component({
  selector: 'pm-root',
  template:
    <div><h1>{{pageTitle}}</h1>
      <div>My First Component</div>
    </div>
})
export class AppComponent {
  pageTitle: string = 'Acme Product Management';
}
```

Table of Contents Notes

- Course Overview 0 1m 40s
- Introduction 0 10m 45s
- First Things First 0 22m 27s
- Introduction to Components 0 27m 56s
 - Introduction 1m 26s
 - What is a Component? 1m 59s
 - Creating the Component Class 2m 17s
 - Defining the Metadata with a D... 2m 32s
 - Importing What We Need 2m 8s
 - Demo: Creating the App Compo... 4m 5s
 - Bootstrapping Our App Comp... 4m 22s
 - Demo: Bootstrapping Our App... 3m 49s
 - Checklists and Summary 5m 15s
- Templates, Interpolation, and Directives 0 33m 6s
- Data Binding & Pipes 0 21m 54s
- More on Components 0 31m 5s

Building Nested

AppModule: Boot straps the application



Directive-> Metadata like imports uses browser module which is required for running application in browser

The screenshot shows a Pluralsight video player interface with the following details:

Table of Contents:

- Course Overview
- Introduction
- First Things First
- Introduction to Components
- Introduction
- What is a Component?
- Creating the Component Class
- Defining the Metadata with a D...
- Importing What We Need
- Demo: Creating the App Compo...
- Bootstrapping Our App Comp...
- Demo: Bootstrapping Our App...
- Checklists and Summary
- Templates, Interpolation, and Directives
- Data Binding & Pipes
- More on Components
- Building Nested

Code Snippet (`app.module.ts`):

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { AppComponent } from './app.component';

@NgModule({
  imports: [ BrowserModule ],
  declarations: [ AppComponent ],
  bootstrap: [ AppComponent ]
})
export class AppModule { }

```

*Declaration array declares the **AppComponent** so that angular can locate its selector
 Import array declares the browser Module so that application can run smoothly in browser
 Bootstrap array lists our **AppComponent** as the starting component for the application

```

1 import { BrowserModule } from '@angular/platform-browser';
2 import { NgModule } from '@angular/core';
3
4 import { AppComponent } from './app.component';
5
6 @NgModule({
7   declarations: [
8     AppComponent
9   ],
10   imports: [
11     BrowserModule
12   ],
13   bootstrap: [AppComponent]
14 })
15 export class AppModule { }
16

```

Note: export keyword makes the class accessible to be imported to the other class of the application

To Access the questionair

Something's Wrong! Checklist (cont.)

- Check my blog for a solution**
 - <http://blogs.msmvps.com/deborahk/angular-2-getting-started-problem-solver/>
- Post to the course discussion**
 - <https://app.pluralsight.com/library/courses/angular-2-getting-started-update/discussion>

Template Interpolation and Directives

Binding:

Activities Google Chrome ▾

Google News Home | Pluralsight Angular: Getting Started with Angular 2 springboot service Java Streams | Baeldung Spring Boot + REST API How to Build Graphs APM

Apr 23 22:07 • https://app.pluralsight.com/player?course=angular-2-getting-started-update&author=deborah-kurata&name=angular-2-getting-started-update-m4&clip=4&mode=live

Binding

Coordinates communication between the component's class and its template and often involves passing data.

Table of Contents Notes

- Templates, Interpolation, and Directives 33m 6s
 - Introduction 2m 17s
 - Building a Template 5m 43s
 - Building the Component 2m 25s
 - Using a Component as a Directive 5m 29s
 - Binding with Interpolation 4m 7s
 - Adding Logic with Directives: ngIf 5m 1s
 - Adding Logic with Directives: ngFor 4m 49s
 - Checklists and Summary 3m 11s
- Data Binding & Pipes 21m 54s
 - More on Components 31m 5s
- Building Nested Components 25m 49s
 - Services and Dependency Injection 18m 50s
- Retrieving Data Using HTTP 27m 31s
 - Navigation and Routing Basics 27m 11s

Interpolation:

Activities Google Chrome ▾

Google News Home | Pluralsight Angular: Getting Started with Angular 2 springboot service Java Streams | Baeldung Spring Boot + REST API How to Build Graphs APM

Apr 23 22:09 • https://app.pluralsight.com/player?course=angular-2-getting-started-update&author=deborah-kurata&name=angular-2-getting-started-update-m4&clip=4&mode=live

Interpolation

Table of Contents Notes

- Templates, Interpolation, and Directives 33m 6s
 - Introduction 2m 17s
 - Building a Template 5m 43s
 - Building the Component 2m 25s
 - Using a Component as a Directive 5m 29s
 - Binding with Interpolation 4m 7s
 - Adding Logic with Directives: ngIf 5m 1s
 - Adding Logic with Directives: ngFor 4m 49s
 - Checklists and Summary 3m 11s
- Data Binding & Pipes 21m 54s
 - More on Components 31m 5s
- Building Nested Components 25m 49s
 - Services and Dependency Injection 18m 50s
- Retrieving Data Using HTTP 27m 31s
 - Navigation and Routing Basics 27m 11s

Activities Google Chrome ▾

Apr 23 22:15 •

Google News | Home | Pluralsight | Angular: G | springboot ser | Java Streams | Spring Boot | How to Build | APM | osteoarthritis | What Is Osteo | +

https://app.pluralsight.com/player?course=angular-2-getting-started-update&author=deborah-kurata&name=angular-2-getting-started-update-m4&clip=4&mode=live

Interpolation

Template

```
<h1>{{pageTitle}}</h1>
{{'Title: ' + pageTitle}}
{{2*20+1}}
```

Class

```
export class AppComponent {
  pageTitle: string =
    'Acme Product Management';
  getTitle(): string {...}
}
```

Table of Contents Notes

- Templates, Interpolation, and Directives (33m 6s)
 - Introduction (2m 17s)
 - Building a Template (5m 43s)
 - Building the Component (2m 25s)
 - Using a Component as a Directive (5m 29s)
 - Binding with Interpolation (4m 7s)
 - Adding Logic with Directives: ngIf (5m 1s)
 - Adding Logic with Directives: ... (4m 49s)
 - Checklists and Summary (3m 11s)
- Data Binding & Pipes (21m 54s)
- More on Components (31m 5s)
- Building Nested Components (25m 49s)
- Services and Dependency Injection (18m 50s)
- Retrieving Data Using HTTP (27m 31s)
- Navigation and Routing Basics (27m 11s)

Table of Contents Notes

- Templates, Interpolation, and Directives (33m 6s)
 - Introduction (2m 17s)
 - Building a Template (5m 43s)
 - Building the Component (2m 25s)
 - Using a Component as a Directive (5m 29s)
 - Binding with Interpolation (4m 7s)
 - Adding Logic with Directives: ngIf (5m 1s)
 - Adding Logic with Directives: ... (4m 49s)
 - Checklists and Summary (3m 11s)
- Data Binding & Pipes (21m 54s)
- More on Components (31m 5s)
- Building Nested Components (25m 49s)
- Services and Dependency Injection (18m 50s)
- Retrieving Data Using HTTP (27m 31s)
- Navigation and Routing Basics (27m 11s)

Directives:

Activities Google Chrome ▾

Apr 23 22:17 •

Google News | Home | Pluralsight | Angular: G | springboot ser | Java Streams | Spring Boot | How to Build | APM | osteoarthritis | What Is Osteo | +

https://app.pluralsight.com/player?course=angular-2-getting-started-update&author=deborah-kurata&name=angular-2-getting-started-update-m4&clip=5&mode=live

Directive

Custom HTML element or attribute used to power up and extend our HTML.

- Custom
- Built-In

Table of Contents Notes

- Templates, Interpolation, and Directives (33m 6s)
 - Introduction (2m 17s)
 - Building a Template (5m 43s)
 - Building the Component (2m 25s)
 - Using a Component as a Directive (5m 29s)
 - Binding with Interpolation (4m 7s)
 - Adding Logic with Directives: ngIf (5m 1s)
 - Adding Logic with Directives: ... (4m 49s)
 - Checklists and Summary (3m 11s)
- Data Binding & Pipes (21m 54s)
- More on Components (31m 5s)
- Building Nested Components (25m 49s)
- Services and Dependency Injection (18m 50s)
- Retrieving Data Using HTTP (27m 31s)
- Navigation and Routing Basics (27m 11s)

Activities Google Chrome ▾

Google News | Home | Pluralsight | Angular: G | springboot ser | Java Streams | Spring Boot | How to Build | APM | osteoarthritis | What Is Osteo | +

Apr 23 22:17 • https://app.pluralsight.com/player?course=angular-2-getting-started-update&author=deborah-kurata&name=angular-2-getting-started-update-m4&clip=5&mode=live

Custom Directives

app.component.ts

```
@Component({
  selector: 'pm-root',
  template:
    <div><h1>{{pageTitle}}</h1>
      <pm-products></pm-products>
    </div>
})
export class AppComponent { }
```

product-list.component.ts

```
@Component({
  selector: 'pm-products',
  templateUrl:
    './product-list.component.html'
})
export class ProductListComponent { }
```

Table of Contents Notes

- Directives** 0 33m 6s
 - Introduction 2m 17s
 - Building a Template 5m 43s
 - Building the Component 2m 25s
 - Using a Component as a Directive 5m 29s
 - Binding with Interpolation 4m 7s
 - Adding Logic with Directives: ngIf 5m 1s
 - Adding Logic with Directives: ... 4m 49s
 - Checklists and Summary 3m 11s
- Data Binding & Pipes** 0 21m 54s
 - More on Components 0 31m 5s
- Building Nested Components** 0 25m 49s
 - Services and Dependency Injection 0 18m 50s
- Retrieving Data Using HTTP** 0 27m 31s
 - Navigation and Routing Basics 0 27m 1s

Structural Directives:: * in front denotes structural directives

Activities Google Chrome ▾

Google News | Home | Pluralsight | Angular: G | springboot ser | Java Streams | Spring Boot | How to Build | APM | osteoarthritis | What Is Osteo | +

Apr 23 22:18 • https://app.pluralsight.com/player?course=angular-2-getting-started-update&author=deborah-kurata&name=angular-2-getting-started-update-m4&clip=5&mode=live

Angular Built-in Directives

Structural
Directives
*ngIf: If logic
*ngFor: For loops

Table of Contents Notes

- Directives** 0 33m 6s
 - Introduction 2m 17s
 - Building a Template 5m 43s
 - Building the Component 2m 25s
 - Using a Component as a Directive 5m 29s
 - Binding with Interpolation 4m 7s
 - Adding Logic with Directives: ngIf 5m 1s
 - Adding Logic with Directives: ... 4m 49s
 - Checklists and Summary 3m 11s
- Data Binding & Pipes** 0 21m 54s
 - More on Components 0 31m 5s
- Building Nested Components** 0 25m 49s
 - Services and Dependency Injection 0 18m 50s
- Retrieving Data Using HTTP** 0 27m 31s
 - Navigation and Routing Basics 0 27m 1s

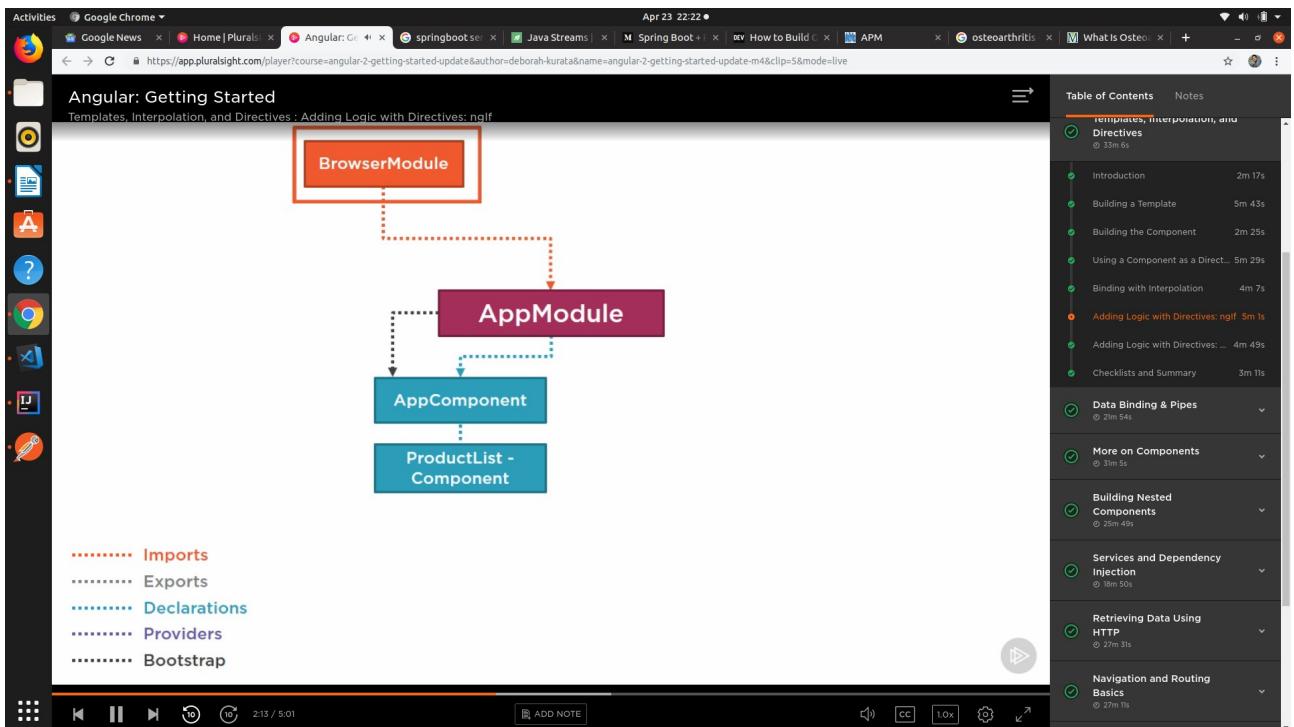
If the below condition fails it will not render anything under the <table> as shown in the right side of the page

```
*ngIf Built-In Directive

<div class='table-responsive'>
  <table class='table' *ngIf='products && products.length'>
    <thead> ...
    </thead>
    <tbody> ...
    </tbody>
  </table>
</div>
```

The code block shows a template with a `*ngIf` directive. The entire template is crossed out with a large red X. A red box highlights the condition `*ngIf='products && products.length'`. The browser's sidebar shows a table of contents for a course on Angular Directives.

Browser Module is required for using the `*ngIf` and `*ngFor`



`*ngFor: let` makes product as template input variable

Apr 23 22:34 ●

Activities Google Chrome ▾

Google News Home | Pluralsight Angular: G+ springboot se Java Streams Spring Boot How to Build APM osteoarthritis What Is Osteo +

https://app.pluralsight.com/player?course=angular-2-getting-started-update&author=d Deborah-kurata&name=angular-2-getting-started-update-m4&clip=6&mode=live

*ngFor Built-In Directive

```
<tr *ngFor='let product of products'>
  <td></td>
  <td>{{ product.productName }}</td>
  <td>{{ product.productCode }}</td>
  <td>{{ product.releaseDate }}</td>
  <td>{{ product.price }}</td>
  <td>{{ product.starRating }}</td>
</tr>
```

Template input variable

Table of Contents Notes

- Directives ④ 33m 6s
 - Introduction 2m 17s
 - Building a Template 5m 43s
 - Building the Component 2m 25s
 - Using a Component as a Directive 5m 29s
 - Binding with Interpolation 4m 7s
 - Adding Logic with Directives: ngIf 5m 1s
 - Adding Logic with Directives: ngFor 4m 49s
 - Checklists and Summary 3m 11s
- Data Binding & Pipes ④ 21m 54s
 - More on Components ④ 31m 5s
- Building Nested Components ④ 25m 49s
 - Services and Dependency Injection ④ 18m 50s
- Retrieving Data Using HTTP ④ 27m 31s
 - Navigation and Routing Basics ④ 27m 11s

Apr 23 22:35 ●

Activities Google Chrome ▾

Google News Home | Pluralsight Angular: G+ springboot se Java Streams Spring Boot How to Build APM osteoarthritis What Is Osteo +

product-list.component.html - APM - Visual Studio Code

File Edit View Go Help

product-list.component.html x product-list.component.ts x

```
16      </div>
17      <div class='table-responsive'>
18        <table class='table'>
19          "ngIf='products && products.length'>
20            <thead>
21              <tr>
22                <th>
23                  <button class='btn btn-primary'>
24                    Show Image
25                  </button>
26                </th>
27                <th>Product</th>
28                <th>Code</th>
29                <th>Available</th>
30                <th>Price</th>
31                <th>5 Star Rating</th>
32            </tr>
33          </thead>
34          <tbody>
35            <tr *ngFor='let product of products'>
36              <td></td>
37              <td>{{ product.productName }}</td>
38            </tr>
39          </tbody>
40        </table>
41      </div>
42    
```

```
6    })
7    export class ProductListComponent {
8      pageTitle: string = 'Product List';
9      products: any[] = [
10        {
11          "productId": 2,
12          "productName": "Garden Cart",
13          "productCode": "GDN-0023",
14          "releaseDate": "March 18, 2016",
15          "description": "15 gallon capacity rolling garden cart",
16          "price": 32.99,
17          "starRating": 4.2,
18          "imageUrl": "http://openclipart.org/image/300px/svg_to_png/47131/garden-cart.png"
19        },
20        {
21          "productId": 5,
22          "productName": "Hammer",
23          "productCode": "TBX-0048",
24          "releaseDate": "May 21, 2016",
25          "description": "Curved claw steel hammer",
26          "price": 8.9,
27          "starRating": 4.8,
28          "imageUrl": "http://openclipart.org/image/300px/svg_to_png/47131/garden-cart.png"
29        }
30      ];
31    }
32  
```

Ln 37, Col 51 Spaces: 4 UTF-8 CRLF HTML 🎨

Table of Contents Notes

- Directives ④ 33m 6s
 - Introduction 2m 17s
 - Building a Template 5m 43s
 - Building the Component 2m 25s
 - Using a Component as a Directive 5m 29s
 - Binding with Interpolation 4m 7s
 - Adding Logic with Directives: ngIf 5m 1s
 - Adding Logic with Directives: ngFor 4m 49s
 - Checklists and Summary 3m 11s
- Data Binding & Pipes ④ 21m 54s
 - More on Components ④ 31m 5s
- Building Nested Components ④ 25m 49s
 - Services and Dependency Injection ④ 18m 50s
- Retrieving Data Using HTTP ④ 27m 31s
 - Navigation and Routing Basics ④ 27m 11s

for Of VS for in ::

for...of vs for...in

for...of

- Iterates over iterable objects, such as an array.
- Result: di, boo, punkeye

```
let nicknames= ['di', 'boo', 'punkeye'];

for (let nickname of nicknames) {
  console.log(nickname);
}
```

for...in

- Iterates over the properties of an object.
- Result: 0, 1, 2

```
let nicknames= ['di', 'boo', 'punkeye'];

for (let nickname in nicknames) {
  console.log(nickname);
}
```

Table of Contents

- Templates, Interpolation, and Directives (33m 6s)
 - Introduction (2m 17s)
 - Building a Template (5m 43s)
 - Building the Component (2m 25s)
 - Using a Component as a Directive (5m 29s)
 - Binding with Interpolation (4m 7s)
 - Adding Logic with Directives: ngIf (5m 1s)
 - Adding Logic with Directives: ngFor (4m 49s)
 - Checklists and Summary (3m 11s)
- Data Binding & Pipes (21m 54s)
 - More on Components (31m 5s)
- Building Nested Components (25m 49s)
 - Services and Dependency Injection (18m 50s)
- Retrieving Data Using HTTP (27m 31s)
- Navigation and Routing Basics (27m 11s)

Data Binding and Pipes

Binging:

Property Binding

Element Property

Template Expression

Table of Contents

- Defining Interfaces (5m 41s)
- Encapsulating Component Styles (3m 34s)
- Using Lifecycle Hooks (4m 16s)
- Building Custom Pipes (6m 57s)
- Filtering a List (4m 41s)
- Checklists and Summary (3m 26s)
- Building Nested Components (25m 49s)
 - Services and Dependency Injection (18m 50s)
- Retrieving Data Using HTTP (27m 31s)
- Navigation and Routing Basics (27m 11s)
- Navigation and Routing Additional Techniques (21m 41s)
- Angular Modules (40m 27s)
- Building, Testing, and Deploying with the CLI (22m 25s)
- Final Words (6m 17s)

Activities Google Chrome ▾ Apr 23 22:45

Google News Home | Pluralsight Angular: G+ springboot ser... Java Streams Spring Boot How to Build APM osteoarthritis What Is Osteo...

https://app.pluralsight.com/player?course=angular-2-getting-started-update&author=deborah-kurata&name=angular-2-getting-started-update-m5&clip=1&mode=live

Property Binding

[] Binding Target '' Binding Source

Table of Contents Notes

- Defining Interfaces 5m 41s
- Encapsulating Component Styles 3m 34s
- Using Lifecycle Hooks 4m 16s
- Building Custom Pipes 6m 57s
- Filtering a List 4m 41s
- Checklists and Summary 3m 26s

Building Nested Components 25m 49s

Services and Dependency Injection 18m 50s

Retrieving Data Using HTTP 27m 31s

Navigation and Routing Basics 27m 11s

Navigation and Routing Additional Techniques 21m 41s

Angular Modules 40m 27s

Building, Testing, and Deploying with the CLI 22m 25s

Final Words 8m 17s

Activities Google Chrome ▾ Apr 23 22:46

Google News Home | Pluralsight Angular: G+ springboot ser... Java Streams Spring Boot How to Build APM osteoarthritis What Is Osteo...

https://app.pluralsight.com/player?course=angular-2-getting-started-update&author=deborah-kurata&name=angular-2-getting-started-update-m5&clip=1&mode=live

Property Binding

```
<img [src]='product.imageUrl'>
<img src={{product.imageUrl}}>
<img src='http://openclipart.org/{{product.imageUrl}}'>
```

Table of Contents Notes

- Defining Interfaces 5m 41s
- Encapsulating Component Styles 3m 34s
- Using Lifecycle Hooks 4m 16s
- Building Custom Pipes 6m 57s
- Filtering a List 4m 41s
- Checklists and Summary 3m 26s

Building Nested Components 25m 49s

Services and Dependency Injection 18m 50s

Retrieving Data Using HTTP 27m 31s

Navigation and Routing Basics 27m 11s

Navigation and Routing Additional Techniques 21m 41s

Angular Modules 40m 27s

Building, Testing, and Deploying with the CLI 22m 25s

Final Words 8m 17s

Event Binding:

Activities Google Chrome ▾

Google News Home | Pluralsight Angular: G... springboot ser... Java Streams Spring Boot How to Build APM osteoarthritis What Is Osteo...

Apr 23 22:58 • https://app.pluralsight.com/player?course=angular-2-getting-started-update&author=deborah-kurata&name=angular-2-getting-started-update-m5&clip=2&mode=live

Event Binding

Template Class

```
<h1>{{pageTitle}}</h1>
<img [src]='product.imageUrl'>
```

```
export class ListComponent {
  pageTitle: string = 'Product List';
  products: any[] = [...];
}
```

Table of Contents Notes

- Handling Events with Event Bindings 4m 33s
- Handling Input with Two-way Data Binding 4m 56s
- Transforming Data with Pipes 4m 3s
- Checklists and Summary 2m 58s
- More on Components 3m 5s
- Building Nested Components 25m 49s
- Services and Dependency Injection 18m 50s
- Retrieving Data Using HTTP 27m 31s
- Navigation and Routing Basics 27m 1s
- Navigation and Routing Additional Techniques 21m 41s
- Angular Modules 40m 27s
- Building, Testing, and Deploying with the CLI 22m 25s
- Final Words 6m 17s

Activities Google Chrome ▾

Google News Home | Pluralsight Angular: G... springboot ser... Java Streams Spring Boot How to Build APM osteoarthritis What Is Osteo...

Apr 23 22:59 • https://app.pluralsight.com/player?course=angular-2-getting-started-update&author=deborah-kurata&name=angular-2-getting-started-update-m5&clip=2&mode=live

Event Binding

Template Class

```
<h1>{{pageTitle}}</h1>
<img [src]='product.imageUrl'>
<button (click)='toggleImage()'>
```

Target Event

Template Statement

```
export class ListComponent {
  pageTitle: string = 'Product List';
  products: any[] = [...];
}
```

Table of Contents Notes

- Handling Events with Event Bindings 4m 33s
- Handling Input with Two-way Data Binding 4m 56s
- Transforming Data with Pipes 4m 3s
- Checklists and Summary 2m 58s
- More on Components 3m 5s
- Building Nested Components 25m 49s
- Services and Dependency Injection 18m 50s
- Retrieving Data Using HTTP 27m 31s
- Navigation and Routing Basics 27m 1s
- Navigation and Routing Additional Techniques 21m 41s
- Angular Modules 40m 27s
- Building, Testing, and Deploying with the CLI 22m 25s
- Final Words 6m 17s

Angular: Getting Started

Event Binding

Template

```
<h1>{{pageTitle}}</h1>
<img [src]='product.imageUrl'>
<button (click)='toggleImage()'>
```

Class

```
export class ListComponent {
  pageTitle: string = 'Product List';
  products: any[] = [...];
  toggleImage(): void {...}
}
```

Table of Contents

- Handling Events with Event Bindings 4m 33s
- Handling Input with Two-way Data Binding 4m 56s
- Transforming Data with Pipes 4m 3s
- Checklists and Summary 2m 58s
- More on Components 3m 5s
- Building Nested Components 25m 49s
- Services and Dependency Injection 18m 50s
- Retrieving Data Using HTTP 27m 31s
- Navigation and Routing Basics 27m 1s
- Navigation and Routing Additional Techniques 21m 41s
- Angular Modules 40m 27s
- Building, Testing, and Deploying with the CLI 22m 25s
- Final Words 6m 17s

example: {{showImage?'Hide': 'Show'}} Image

product-list.component.html - APM - Visual Studio Code

```
product-list.component.html
File Edit View Go Help
```

```
product-list.component.ts
```

```
product-list.component.html
20   <td>
21     <tr>
22       <th>
23         <button class='btn btn-primary'
24           (click)='toggleImage()'
25           {{showImage ? 'Hide' : 'Show'}} Image
26         </button>
27       </th>
28       <th>Product</th>
29       <th>Code</th>
30       <th>Available</th>
31       <th>Price</th>
32       <th>5 Star Rating</th>
33     </tr>
34   head>
35   ody>
36   <tr *ngFor='let product of products'>
37     <td>
38       <img *ngIf='showImage'
39         [src]='product.imageUrl'
40         [title]='product.productName'
41         [style.width.px]='imageWidth'
42         [style.margin.px]='imageMargin'>
43     </td>
44     <td>{{ product.productName }}</td>
45     <td>{{ product.productCode }}</td>
46     <td>{{ product.releaseDate }}</td>
```

```
product-list.component.ts
21   "imageUrl": "http://openclipart.org/image/300
22   },
23   {
24     "productId": 5,
25     "productName": "Hammer",
26     "productCode": "TBX-0048",
27     "releaseDate": "May 21, 2016",
28     "description": "Curved claw steel hammer",
29     "price": 8.9,
30     "starRating": 4.8,
31     "imageUrl": "http://openclipart.org/image/300
32   }
33 ];
34
35   toggleImage(): void {
36     this.showImage = !this.showImage;
37   }
38 }
```

Table of Contents

- Handling Events with Event Bindings 4m 33s
- Handling Input with Two-way Data Binding 4m 56s
- Transforming Data with Pipes 4m 3s
- Checklists and Summary 2m 58s
- More on Components 3m 5s
- Building Nested Components 25m 49s
- Services and Dependency Injection 18m 50s
- Retrieving Data Using HTTP 27m 31s
- Navigation and Routing Basics 27m 1s
- Navigation and Routing Additional Techniques 21m 41s
- Angular Modules 40m 27s
- Building, Testing, and Deploying with the CLI 22m 25s
- Final Words 6m 17s

Two Way Binding: It uses FormModule

Activities Google News Home | Pluralsight Angular: Getting Started springboot service Java Streams Spring Boot + REST How to Build APM osteoarthritis What Is Osteo Apr 23 23:08 ●

https://app.pluralsight.com/player?course=angular-2-getting-started-update&author=deborah-kurata&name=angular-2-getting-started-update-m5&clip=3&mode=live

Two-way Binding

Template

```
<input [(ngModel)]='listFilter'>
```

Class

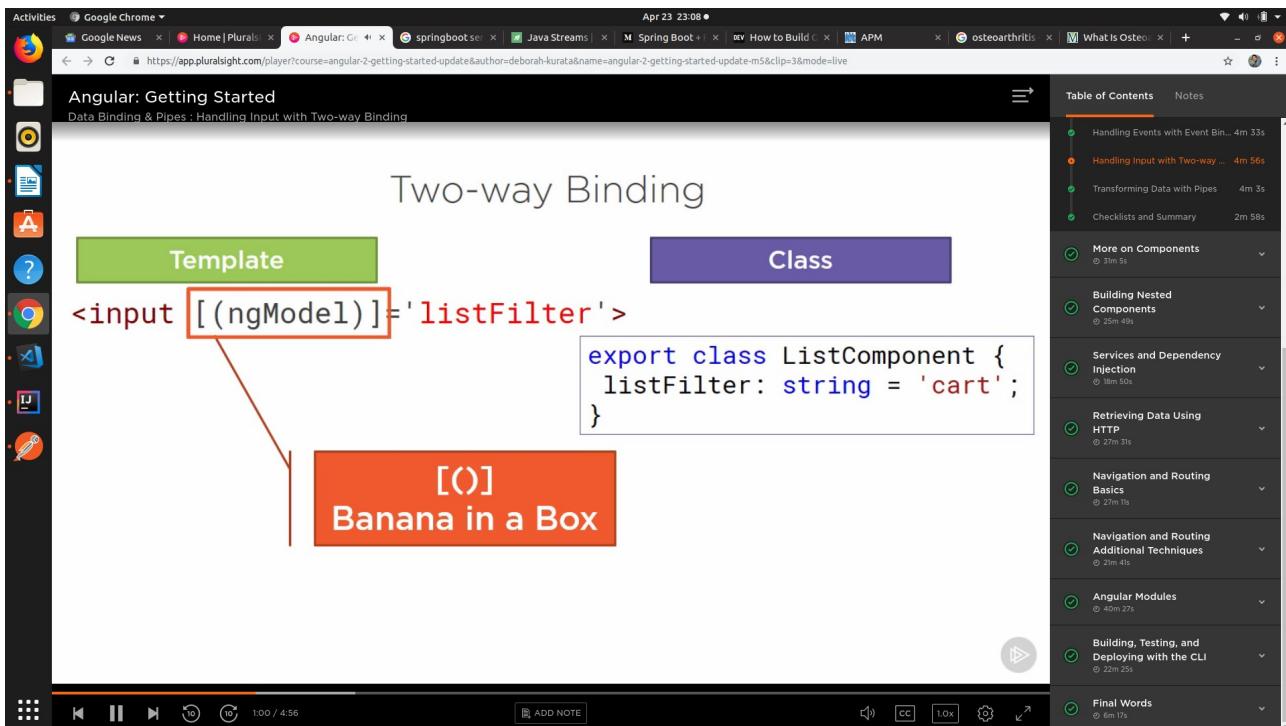
```
export class ListComponent {  
  listFilter: string = 'cart';  
}
```

[0] Banana in a Box

Table of Contents Notes

- Handling Events with Event Bindings 4m 33s
- Handling Input with Two-way Binding 4m 56s
- Transforming Data with Pipes 4m 3s
- Checklists and Summary 2m 58s
- More on Components 3m 5s
- Building Nested Components 25m 49s
- Services and Dependency Injection 18m 50s
- Retrieving Data Using HTTP 27m 31s
- Navigation and Routing Basics 27m 1s
- Navigation and Routing Additional Techniques 21m 41s
- Angular Modules 40m 27s
- Building, Testing, and Deploying with the CLI 22m 55s
- Final Words 6m 17s

1.00 / 4:56 ADD NOTE



Pipes::

Activities Google News Home | Pluralsight Angular: Getting Started springboot service Java Streams Spring Boot + REST How to Build APM osteoarthritis What Is Osteo Apr 24 00:15 ●

https://app.pluralsight.com/player?course=angular-2-getting-started-update&author=deborah-kurata&name=angular-2-getting-started-update-m5&clip=4&mode=live

Transforming Data with Pipes

Transform bound properties before display

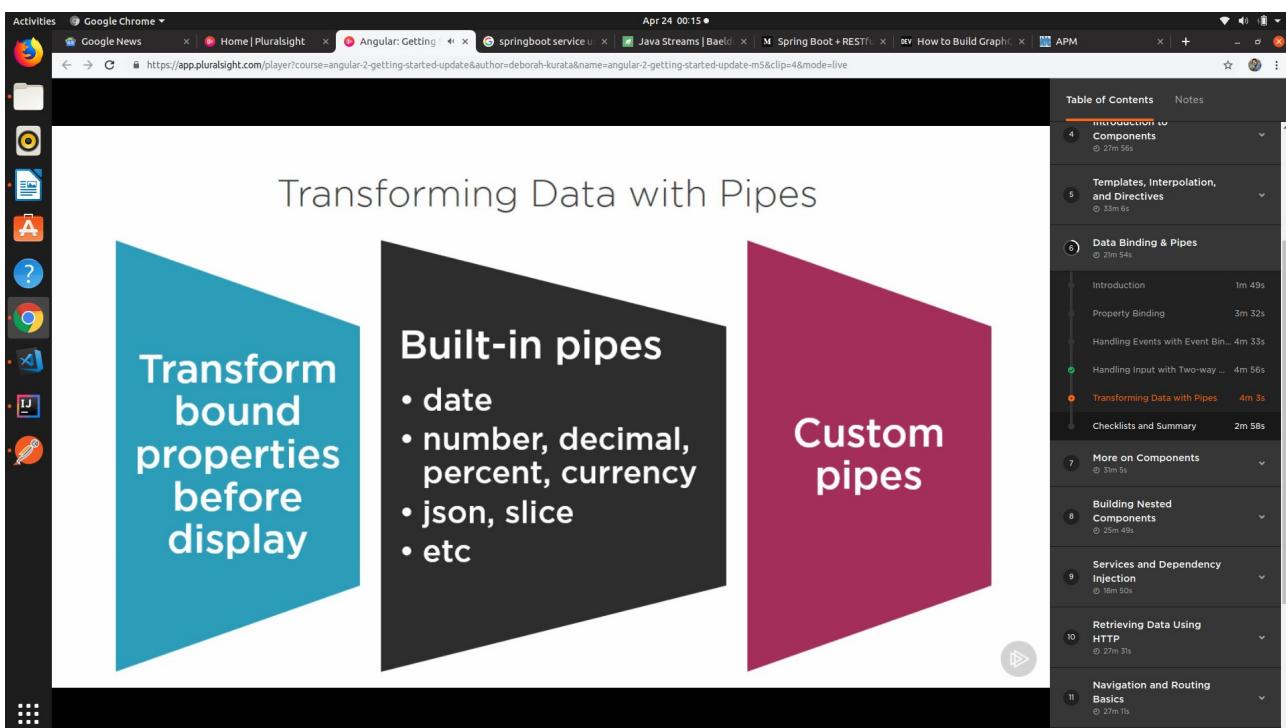
Built-in pipes

- date
- number, decimal, percent, currency
- json, slice
- etc

Custom pipes

Table of Contents Notes

- Introduction 1m 49s
- Components 27m 56s
- Templates, Interpolation, and Directives 33m 6s
- Data Binding & Pipes 21m 54s
- Introduction 1m 49s
- Property Binding 3m 32s
- Handling Events with Event Bindings 4m 33s
- Handling Input with Two-way Binding 4m 56s
- Transforming Data with Pipes 4m 3s
- Checklists and Summary 2m 58s
- More on Components 31m 5s
- Building Nested Components 25m 49s
- Services and Dependency Injection 18m 50s
- Retrieving Data Using HTTP 27m 31s
- Navigation and Routing Basics 27m 1s



Activities Google Chrome ▾

Google News Home | Pluralsight Angular: Getting started with Angular 2 | Pluralsight Spring Boot service | Baeldung Java Streams | Baeldung Spring Boot + RESTful API | How to Build Graphs | APM

Apr 24 00:17 •

<https://app.pluralsight.com/player?course=angular-2-getting-started-update&author=deborah-kurata&name=angular-2-getting-started-update-m5&clip=4&mode=live>

Pipe Examples

```

{{ product.productCode | lowercase }}




Table of Contents



- Introduction to Components 27m 56s
- Templates, Interpolation, and Directives 33m 58s
- Data Binding & Pipes 21m 54s
  - Introduction 1m 49s
  - Property Binding 3m 32s
  - Handling Events with Event Bindings 4m 33s
  - Handling Input with Two-way Binding 4m 56s
  - Transforming Data with Pipes 4m 3s
  - Checklists and Summary 2m 58s
- More on Components 31m 5s
- Building Nested Components 25m 49s
- Services and Dependency Injection 18m 50s
- Retrieving Data Using HTTP 27m 31s
- Navigation and Routing Basics 27m 11s

```

Activities Google Chrome ▾

Google News Home | Pluralsight Angular: Getting started with Angular 2 | Pluralsight Spring Boot service | Baeldung Java Streams | Baeldung Spring Boot + RESTful API | How to Build Graphs | APM

Apr 24 00:19 •

<https://app.pluralsight.com/player?course=angular-2-getting-started-update&author=deborah-kurata&name=angular-2-getting-started-update-m5&clip=5&mode=live>

Data Binding

```
DOM
```

```
<pm-root ng-version="6.0.5">
  <div>
    <h1>Acme Product Management</h1>
    <pm-products>
      <div class="card">
        <div class="card-header">Product List </div>
        <div class="card-body">
          <div class="row"></div>
          <div class="row"></div>
          <div class="table-responsive">
            <table class="table">
              <thead></thead>
              <tbody></tbody>
            </table>
          </div>
        </div>
      </div>
    </pm-products>
  </div>
</pm-root>
```

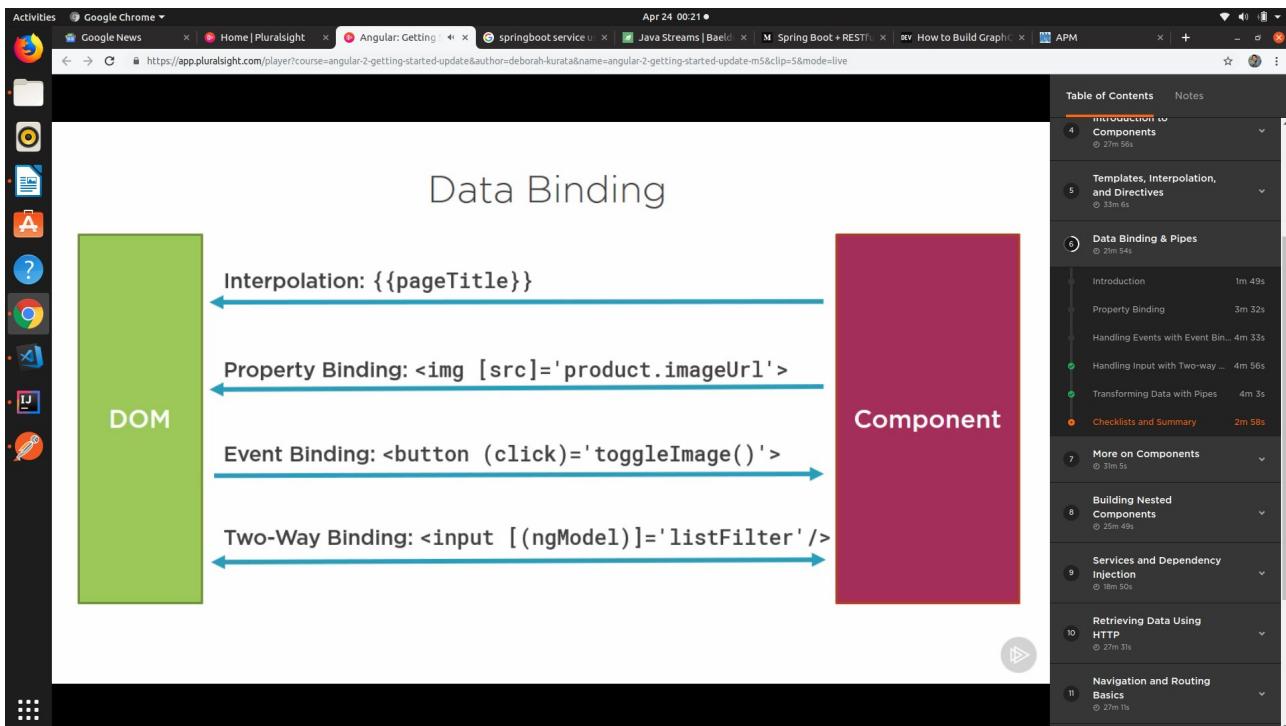
```
product-list.component.ts
```

```
@Component({
  selector: 'pm-products',
  templateUrl: './product-list.component.html'
})
export class ProductListComponent {
  pageTitle: string = 'Product List';
  listFilter: string = 'cart';
  products: any[] = [...];
  toggleImage(): void {...}
}
```

Table of Contents

- Introduction to Components 27m 56s
- Templates, Interpolation, and Directives 33m 58s
- Data Binding & Pipes 21m 54s
 - Introduction 1m 49s
 - Property Binding 3m 32s
 - Handling Events with Event Bindings 4m 33s
 - Handling Input with Two-way Binding 4m 56s
 - Transforming Data with Pipes 4m 3s
 - Checklists and Summary 2m 58s
- More on Components 31m 5s
- Building Nested Components 25m 49s
- Services and Dependency Injection 18m 50s
- Retrieving Data Using HTTP 27m 31s
- Navigation and Routing Basics 27m 11s

CheckList for Binding and Pipes::



ngModule:Banana in box in html and FormsModule in AppModule are prerequisite

The screenshot shows a video player interface with a "Table of Contents" sidebar on the right. The main area displays two code snippets:

product-list.component.html

```
<div class='col-md-4'>
  <input type='text'
    [(ngModel)]='listFilter' />
</div>
```

app.module.ts

```
@NgModule({
  imports: [
    BrowserModule,
    FormsModule,
    declarations: [
      AppComponent,
      ProductListComponent ],
    bootstrap: [ AppComponent ]
  )
  export class AppModule { }
```

The `FormsModule` import in the `app.module.ts` file is highlighted with a red box.

Activities Google Chrome ▾

Google News Home | Pluralsight Angular: Getting ... springboot service Java Streams | Baeldung Spring Boot + REST How to Build Graph APM

Apr 24 00:27 • https://app.pluralsight.com/player?course=angular-2-getting-started-update&author=deborah-kurata&name=angular-2-getting-started-update-m5&clip=5&mode=live

Angular: Getting Started Data Binding & Pipes : Checklists and Summary

Checklist: Pipes



Pipe character |

Pipe name

Pipe parameters

- Separated with colons

Example

- `{{ product.price | currency:'USD':'symbol':'1.2-2' }}`

Table of Contents Notes

- 4 Components 27m 56s
- 5 Templates, Interpolation, and Directives 33m 55s
- 6 Data Binding & Pipes 22m 54s
 - Introduction 1m 49s
 - Property Binding 3m 32s
 - Handling Events with Event Bindings 4m 33s
 - Handling Input with Two-way Binding 4m 56s
 - Transforming Data with Pipes 4m 35s
 - Checklists and Summary 2m 58s
- 7 More on Components 31m 5s
- 8 Building Nested Components 25m 49s
 - Services and Dependency Injection 18m 50s
- 10 Retrieving Data Using HTTP 27m 31s
- 11 Navigation and Routing Basics 27m 11s

◀ ▶ ⏪ ⏩ 2:23 / 258 🔍 CC 1.0x ⌂ ⌂

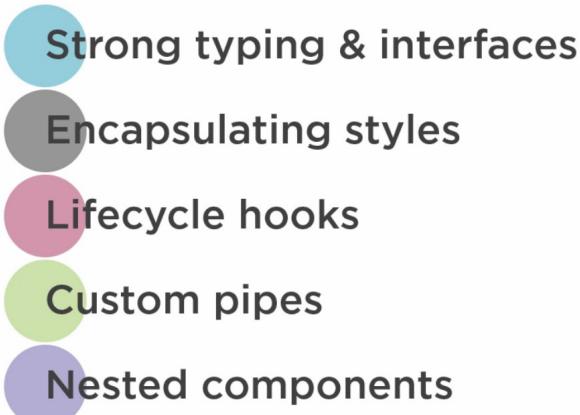
More on Components

Activities Google Chrome ▾

Google News Home | Pluralsight Angular: Getting ... springboot service Java Streams | Baeldung Spring Boot + REST How to Build Graph APM

Apr 24 00:31 • https://app.pluralsight.com/player?course=angular-2-getting-started-update&author=deborah-kurata&name=angular-2-getting-started-update-m5&clip=0&mode=live

Improving Our Components



Strong typing & interfaces

Encapsulating styles

Lifecycle hooks

Custom pipes

Nested components

Table of Contents Notes

- 4 Components 27m 56s
- 5 Templates, Interpolation, and Directives 33m 55s
- 6 Data Binding & Pipes 22m 54s
 - Introduction 2m 27s
 - Defining Interfaces 5m 41s
 - Encapsulating Component Styles 3m 34s
 - Using Lifecycle Hooks 4m 16s
 - Building Custom Pipes 6m 57s
 - Filtering a List 4m 41s
 - Checklists and Summary 3m 26s
- 7 More on Components 31m 5s
- 8 Building Nested Components 25m 49s
 - Services and Dependency Injection 18m 50s
- 10 Retrieving Data Using HTTP 27m 31s
- 11 Navigation and Routing Basics 27m 11s

◀ ▶ ⏪ ⏩ ⌂ ⌂

Strong Type: Like page title field has a type as string, method `toggleImage()` has a void type.
Products has unknown type any.

Strong Typing

```
export class ProductListComponent {
  pageTitle: string = 'Product List';
  showImage: boolean = false;
  listFilter: string = 'cart';
  message: string;

  products: any[] = [...];

  toggleImage(): void {
    this.showImage = !this.showImage;
  }

  onRatingClicked(message: string): void {
    this.message = message;
  }
}
```

The screenshot shows a code editor within a web browser window. The code is written in TypeScript. A red box highlights the declaration of the 'products' variable as 'any[]'. The browser's address bar shows the URL: <https://app.pluralsight.com/player?course=angular-2-getting-started-update&author=deborah-kurata&name=angular-2-getting-started-update-m6&clip=1&mode=live>. To the right of the code editor is a sidebar containing a 'Table of Contents' with several course modules listed.

Interface: These are not supported by ES5 or ES2015 but supported by Typescript so they are not found in the javascript when transpiled

Interface

A **specification** identifying a related set of properties and methods.

A class commits to supporting the specification by **implementing** the interface.

Use the interface as a **data type**.

Development time only!

The screenshot shows a slide from a presentation. The title is 'Interface'. The slide contains text explaining what an interface is: 'A specification identifying a related set of properties and methods.' It also states that a class 'commits to supporting the specification by implementing the interface.' Below this, there is a note: 'Use the interface as a data type.' and 'Development time only!'. The slide is part of a course on Angular, specifically 'More on Components : Defining Interfaces'. The browser's address bar shows the URL: <https://app.pluralsight.com/player?course=angular-2-getting-started-update&author=deborah-kurata&name=angular-2-getting-started-update-m6&clip=1&mode=live>. To the right of the slide is a sidebar containing a 'Table of Contents' with several course modules listed.

Example of interface: export keyword so that it can be used in other places as well

Activities Google Chrome ▾ Apr 24 00:38

Google News | Home | Pluralsight | Angular: Getting | springboot service | Java Streams | Baeldung | Spring Boot + REST | How to Build Graphs | APM

<https://app.pluralsight.com/player?course=angular-2-getting-started-update&author=deborah-kurata&name=angular-2-getting-started-update-m6&clip=1&mode=live>

Interface Is a Specification

```
export interface IProduct {
    productId: number;
    productName: string;
    productCode: string;
    releaseDate: Date;
    price: number;
    description: string;
    starRating: number;
    imageUrl: string;
    calculateDiscount(percent: number): number;
}
```

Table of Contents Notes

- 4 Components 27m 56s
- 5 Templates, Interpolation, and Directives 33m 5s
- 6 Data Binding & Pipes 21m 54s
- 7 More on Components 31m 5s
- Introduction 2m 27s
- Defining Interfaces 5m 41s
- Encapsulating Component Styling 3m 34s
- Using Lifecycle Hooks 4m 16s
- Building Custom Pipes 6m 57s
- Filtering a List 4m 41s
- Checklists and Summary 3m 26s
- Building Nested Components 25m 49s
- Services and Dependency Injection 18m 50s
- Retrieving Data Using HTTP 27m 31s
- Navigation and Routing Basics

In the below example interface Iproduct is imported and used as a datatype by products: IProduct[]

Activities Google Chrome ▾ Apr 24 00:40

Google News | Home | Pluralsight | Angular: Getting | springboot service | Java Streams | Baeldung | Spring Boot + REST | How to Build Graphs | APM

<https://app.pluralsight.com/player?course=angular-2-getting-started-update&author=deborah-kurata&name=angular-2-getting-started-update-m6&clip=1&mode=live>

Using an Interface as a Data Type

```
import { IProduct } from './product';

export class ProductListComponent {
    pageTitle: string = 'Product List';
    showImage: boolean = false;
    listFilter: string = 'cart';

    products: IProduct[] = [...];

    toggleImage(): void {
        this.showImage = !this.showImage;
    }
}
```

Table of Contents Notes

- 4 Components 27m 56s
- 5 Templates, Interpolation, and Directives 33m 5s
- 6 Data Binding & Pipes 21m 54s
- 7 More on Components 31m 5s
- Introduction 2m 27s
- Defining Interfaces 5m 41s
- Encapsulating Component Styling 3m 34s
- Using Lifecycle Hooks 4m 16s
- Building Custom Pipes 6m 57s
- Filtering a List 4m 41s
- Checklists and Summary 3m 26s
- Building Nested Components 25m 49s
- Services and Dependency Injection 18m 50s
- Retrieving Data Using HTTP 27m 31s
- Navigation and Routing Basics

we can add a class to our interface as shown below. We create a class when we require a method that will be used throughout the application we use class. Ex in the below example calculateDiscount() method

Activities Google Chrome ▾

Google News | Home | Pluralsight | Angular: Getting Started | springboot service | Java Streams | Baeldung | Spring Boot + RESTful API | How to Build Graph... | APM

Apr 24 00:44 • https://app.pluralsight.com/player?course=angular-2-getting-started-update&author=deborah-kurata&name=angular-2-getting-started-update-m6&clip=1&mode=live

product.ts - APM - Visual Studio Code

File Edit View Go Help

```
product.ts
1 productName: string;
2 productCode: string;
3 releaseDate: string;
4 price: number;
5 description: string;
6 starRating: number;
7 imageUrl: string;
8 }
9
10 export class Product implements IProduct {
11
12   constructor(productId: number,
13             productName: string,
14             productCode: string,
15             releaseDate: string,
16             price: number,
17             description: string,
18             starRating: number,
19             imageUrl: string) {
20
21   }
22
23   calculateDiscount(percent: number): number {
24     return this.price - (this.price * percent / 100);
25   }
26 }
27
28 }
```

In 28, Col 1 Spaces: 4 UTF-8 CRLF TypeScript

Table of Contents Notes

- 4 Components 27m 56s
- 5 Templates, Interpolation, and Directives 33m 6s
- 6 Data Binding & Pipes 21m 54s
- 7 More on Components 31m 5s
- Introduction 2m 27s
- Defining Interfaces 5m 41s
- Encapsulating Component Styles 3m 34s
- Using Lifecycle Hooks 4m 16s
- Building Custom Pipes 6m 57s
- Filtering a List 4m 41s
- Checklists and Summary 3m 26s
- Building Nested Components 25m 49s
- Services and Dependency Injection 18m 50s
- Retrieving Data Using HTTP 27m 31s
- Navigation and Routing Basics

Encapsulating Component Style:

Activities Google Chrome ▾

Google News | Home | Pluralsight | Angular: Getting Started | springboot service | Java Streams | Baeldung | Spring Boot + RESTful API | How to Build Graph... | APM

Apr 24 00:48 • https://app.pluralsight.com/player?course=angular-2-getting-started-update&author=deborah-kurata&name=angular-2-getting-started-update-m6&clip=2&mode=live

Angular: Getting Started

More on Components : Encapsulating Component Styles

Handling Unique Component Styles



Templates sometimes require unique styles

We can inline the styles directly into the HTML

We can build an external stylesheet and link it in index.html

There is a better way!

Table of Contents Notes

- 4 Components 27m 56s
- 5 Templates, Interpolation, and Directives 33m 6s
- 6 Data Binding & Pipes 21m 54s
- 7 More on Components 31m 5s
- Introduction 2m 27s
- Defining Interfaces 5m 41s
- Encapsulating Component Styles 3m 34s
- Using Lifecycle Hooks 4m 16s
- Building Custom Pipes 6m 57s
- Filtering a List 4m 41s
- Checklists and Summary 3m 26s
- Building Nested Components 25m 49s
- Services and Dependency Injection 18m 50s
- Retrieving Data Using HTTP 27m 31s
- Navigation and Routing Basics

Green is the better option from the below image but it is restricted to the applied component template only

The screenshot shows a Pluralsight video player interface. The main content area displays two code snippets side-by-side. The first snippet, enclosed in a red box, is annotated with the word "styles". It contains the following Angular component definition:

```
@Component({
  selector: 'pm-products',
  templateUrl: './product-list.component.html',
  styles: ['thead {color: #337AB7;}'])
})

```

The second snippet, enclosed in a green box, is annotated with the word "styleUrls". It contains the following Angular component definition:

```
@Component({
  selector: 'pm-products',
  templateUrl: './product-list.component.html',
  styleUrls: ['./product-list.component.css']
})

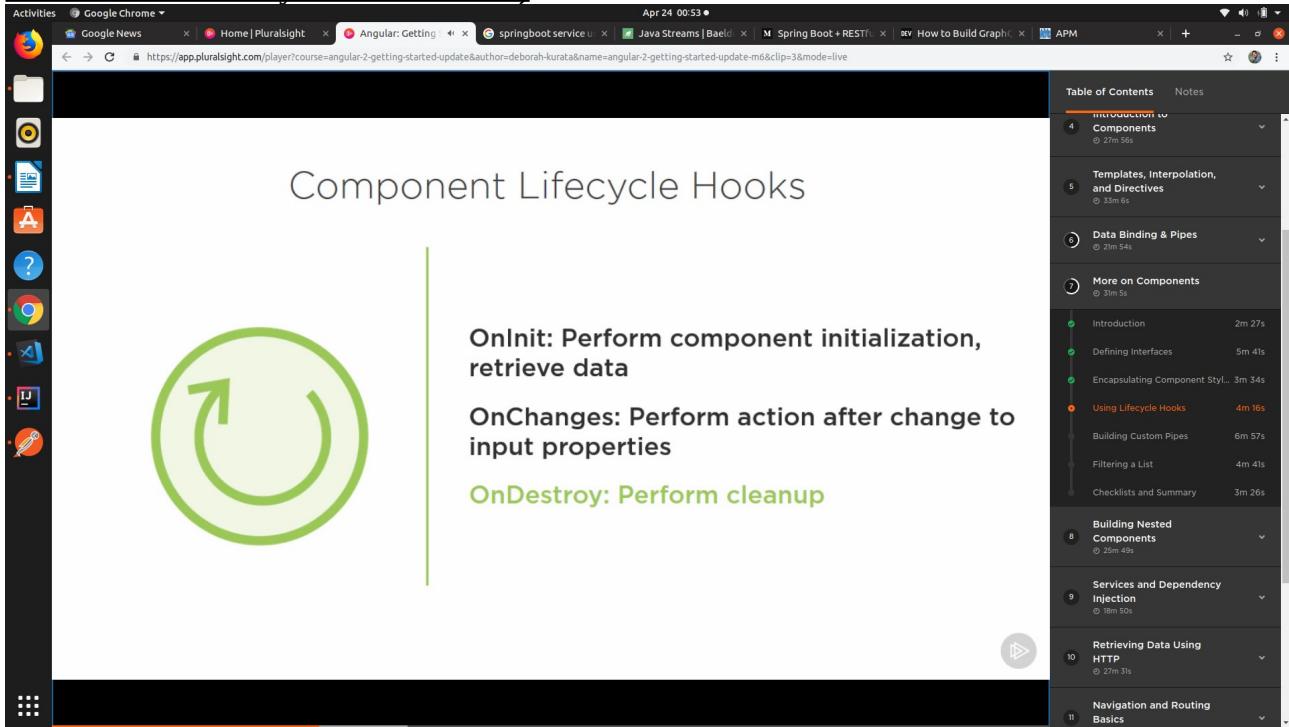
```

To the right of the video player is a sidebar titled "Table of Contents" which lists various video chapters and their durations.

LifeCycle Hooks:

The screenshot shows a Pluralsight video player interface. The main content area displays a diagram illustrating the Angular component lifecycle. The diagram consists of five colored boxes arranged horizontally: "Create" (teal), "Render" (dark grey), "Create and render children" (maroon), "Process changes" (light green), and "Destroy" (purple). Arrows show a clockwise flow between the first four stages, with a feedback loop from "Process changes" back to "Create and render children". To the right of the video player is a sidebar titled "Table of Contents" which lists various video chapters and their durations.

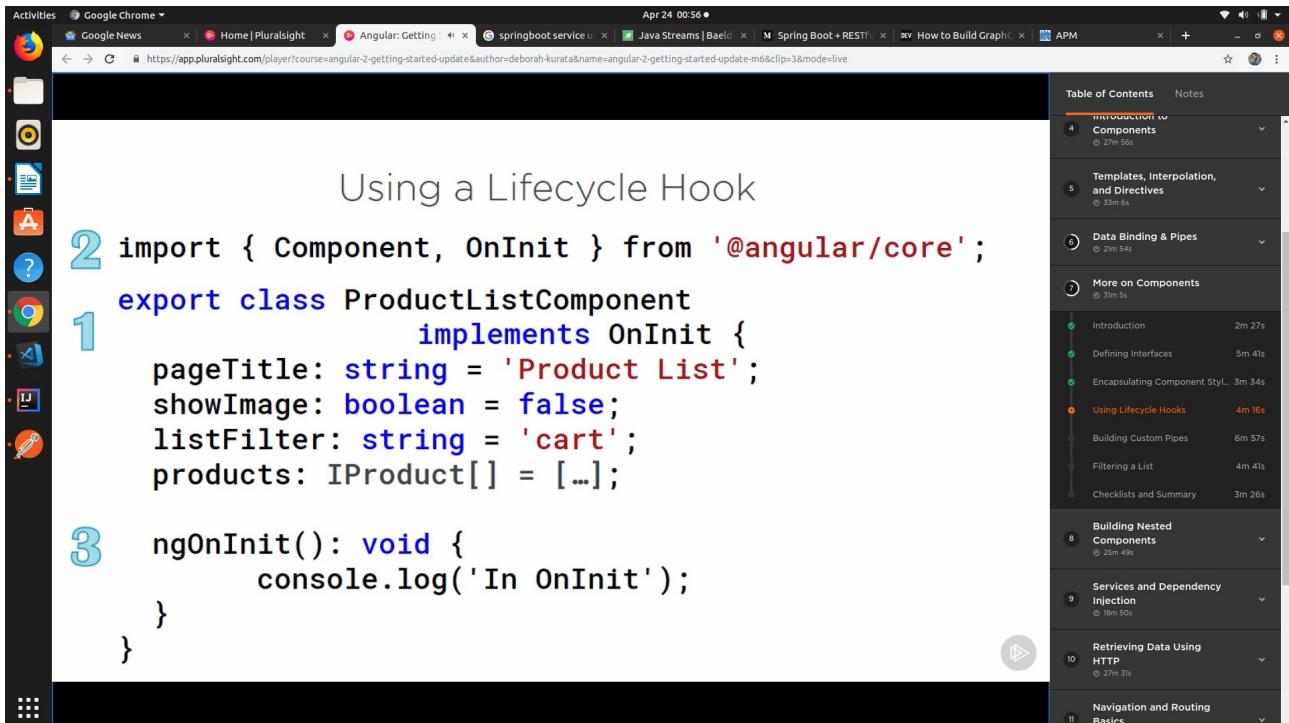
Some of the most widely used life cycle hooks shown below(they are not all some are not listed that we need to study from elsewhere)



The screenshot shows a web browser window with a slide titled "Component Lifecycle Hooks". To the left of the slide is a circular green icon with a white arrow pointing clockwise. The slide content includes three bullet points: "OnInit: Perform component initialization, retrieve data", "OnChanges: Perform action after change to input properties", and " OnDestroy: Perform cleanup". On the right side of the browser window, there is a sidebar titled "Table of Contents" which lists various chapters and their durations.

Chapter	Title	Duration
4	Components	27m 56s
5	Templates, Interpolation, and Directives	33m 6s
6	Data Binding & Pipes	21m 54s
7	More on Components	31m 5s
8	Building Nested Components	25m 49s
9	Services and Dependency Injection	16m 50s
10	Retrieving Data Using HTTP	27m 31s
11	Navigation and Routing Basics	3m 26s

As we know these lifecycle hooks are interfaces so they will not supported by ES2015 or ES5 so its not require to implement it on the component though we can implement it for clarity if we want



The screenshot shows a web browser window displaying a code example for implementing the `OnInit` lifecycle hook. The code is written in TypeScript and defines a class `ProductListComponent` that implements the `OnInit` interface. The code includes variable declarations for `pageTitle`, `showImage`, `listFilter`, and `products`, and a method `ngOnInit()` that logs a message to the console.

```
2 import { Component, OnInit } from '@angular/core';
1 export class ProductListComponent
    implements OnInit {
    pageTitle: string = 'Product List';
    showImage: boolean = false;
    listFilter: string = 'cart';
    products: IProduct[] = [...];
3
3     ngOnInit(): void {
        console.log('In OnInit');
    }
}
```

Custom Pipes :

Activities Google Chrome ▾ Google News ▾ Home | Pluralsight ▾ Angular: Getting Start... Apr 25 00:10 •

https://app.pluralsight.com/player?course=angular-2-getting-started-update&author=deborah-kurata&name=angular-2-getting-started-update-m6&clip=4&mode=live

Angular: Getting Started
More on Components : Building Custom Pipes

Transforming Data with Pipes

Transform bound properties before display

Built-in pipes

- date
- number, decimal, percent, currency
- json, slice
- etc

Custom pipes

Table of Contents Notes

- Course Overview
- Introduction
- First Things First
- Introduction to Components
- Templates, Interpolation, and Directives
- Data Binding & Pipes
- More on Components
 - Introduction
 - Defining Interfaces
 - Encapsulating Component Styling
 - Using Lifecycle Hooks
 - Building Custom Pipes
 - Filtering a List
 - Checklists and Summary
- Building Nested Components

Creating a custom pipe we need to do the below steps

@Pipe(It is a decorator) is a method inside we are giving the name of the pipe that will be referred in html

transform() method has a value that needs to be transformed, character is the field that is transformed like in value there is a – in the field and we will replace it with space(it is the character field in the method parameter) return type is string , which means after all evaluation string will be returned.

Activities Google Chrome ▾ Google News ▾ Home | Pluralsight ▾ Angular: Getting Start... Apr 25 00:12 •

https://app.pluralsight.com/player?course=angular-2-getting-started-update&author=deborah-kurata&name=angular-2-getting-started-update-m6&clip=4&mode=live

Angular: Getting Started
More on Components : Building Custom Pipes

Building a Custom Pipe

```
import { Pipe, PipeTransform } from '@angular/core';

@Pipe({
  name: 'convertToSpaces'
})
export class ConvertToSpacesPipe
  implements PipeTransform {

  transform(value: string,
            character: string): string{
    }
}
```

Table of Contents Notes

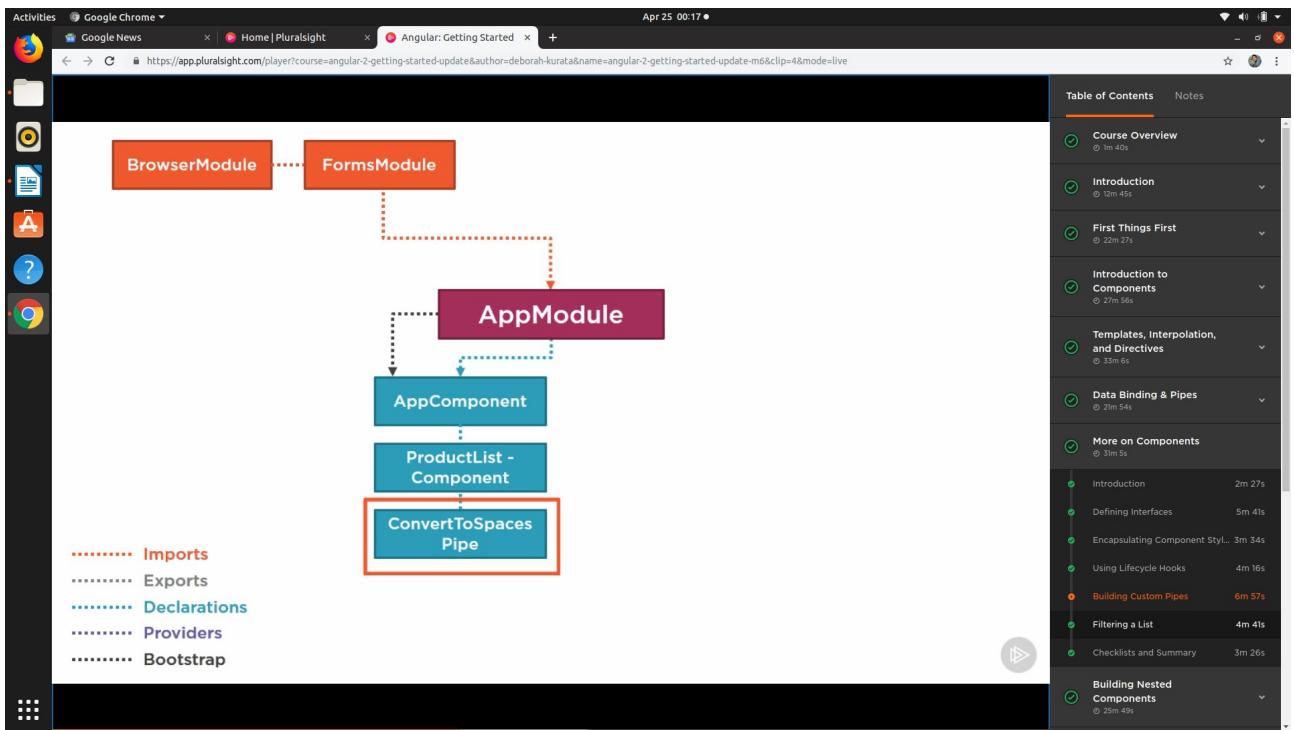
- Course Overview
- Introduction
- First Things First
- Introduction to Components
- Templates, Interpolation, and Directives
- Data Binding & Pipes
- More on Components
 - Introduction
 - Defining Interfaces
 - Encapsulating Component Styling
 - Using Lifecycle Hooks
 - Building Custom Pipes
 - Filtering a List
 - Checklists and Summary
- Building Nested Components

More explanation

A screenshot of a Pluralsight video player interface. The main content area shows a slide titled "Using a Custom Pipe". On the left, there's a green box labeled "Template" containing the Angular template code: <td>{{ product.productCode | convertToSpaces:'-' }}</td>. On the right, there's a teal box labeled "Pipe" containing the pipe implementation code: transform(value: string, character: string): string { }. A red arrow points from the "Pipe" box down to the "character" parameter in the code. The video player has a progress bar at 2:21 / 6:57. The top navigation bar shows "Angular: Getting Started" and the URL https://app.pluralsight.com/player?course=angular-2-getting-started-update&author=deborah-kurata&name=angular-2-getting-started-update-m6&clip=4&mode=live. The top right corner shows the date and time: Apr 25 00:15.

A second screenshot of the same Pluralsight video player interface, showing the same slide and code. In this version, a red arrow points from the "Pipe" box down to the "value" parameter in the code: transform(value: string, character: string): string { }.

This pipe is added in the angular module otherwise it will not be visible



Using a Custom Pipe

Template

```
<td>{{ product.productCode | convertToSpaces:'-' }}</td>
```

Module

```
@NgModule({
  imports: [
    BrowserModule,
    FormsModule ],
  declarations: [
    AppComponent,
    ProductListComponent,
    ConvertToSpacesPipe ],
  bootstrap: [ AppComponent ]
})
export class AppModule { }
```

Table of Contents

- Course Overview
- Introduction
- First Things First
- Introduction to Components
- Templates, Interpolation, and Directives
- Data Binding & Pipes
- More on Components
 - Introduction
 - Defining Interfaces
 - Encapsulating Component Styling
 - Using Lifecycle Hooks
 - Building Custom Pipes
 - Filtering a List
 - Checklists and Summary
- Building Nested Components

Code example

```

1 import { Pipe, PipeTransform } from '@angular/core';
2
3 @Pipe({
4   name: 'convertToSpaces'
5 })
6 export class ConvertToSpacesPipe implements PipeTransform {
7
8   transform(value: string, character: string): string {
9     return value.replace(character, ' ');
10  }
11 }

```

```

30 <th>Available</th>
31 <th>Price</th>
32 <th>5 Star Rating</th>
33 </tr>
34 </thead>
35 <tbody>
36   <tr *ngFor='let product of products'>
37     <td>
38       <img *ngIf='showImage'
39         [src]='product.imageUrl'
40         [title]='product.productName'
41         [style.width.px]='imageWidth'
42         [style.margin.px]='imageMargin'>
43     </td>
44     <td>{{ product.productName }}</td>
45     <td>{{ product.productCode | lowercase | convertToSpaces: '-' }}</td>
46     <td>{{ product.releaseDate }}</td>
47     <td>{{ product.price | currency:'USD':'symbol':'1.2-2' }}</td>
48     <td>{{ product.starRating }}</td>
49   </tr>
50 </tbody>
51 </table>
52 </div>
53 </div>
54 </div>

```

Custom Filter:

- 1)Created a filteredProducts to keep the result
- 2)changed the filtered property with setter and getter methods

```

1 import { Component, OnInit } from '@angular/core';
2 import { IProduct } from './product';
3
4 @Component({
5   selector: 'pm-products',
6   templateUrl: './product-list.component.html',
7   styleUrls: ['./product-list.component.css']
8 })
9 export class ProductListComponent implements OnInit {
10   pageTitle: string = 'Product List';
11   imageWidth: number = 50;
12   imageMargin: number = 2;
13   showImage: boolean = false;
14
15   _listFilter: string;
16   get listFilter(): string {
17     return this._listFilter;
18   }
19   set listFilter(value: string) {
20     this._listFilter = value;
21     this.filteredProducts= this.listFilter ? this.performFilter(this.listFilter) : this.products;
22   }
23
24
25   filteredProducts: IProduct[];
26   products: IProduct[] = [
27     {
28       id: 1,
29       name: 'Dad Hat',
30       description: 'A small cotton hat with a large circular logo on the front',
31       price: 22.0,
32       starRating: 4.8,
33       imageUrl: 'http://openclipart.org/image/300px/svg_to_png/73/rejon_Hat.png'
34     },
35     {
36       id: 2,
37       name: 'Curved Claw Steel Hammer',
38       description: 'A curved claw steel hammer',
39       price: 8.9,
40       starRating: 4.8,
41       releaseDate: "May 21, 2016",
42       imageUrl: 'http://openclipart.org/image/300px/svg_to_png/73/rejon_Hammer.png'
43     }
44   ];
45
46   constructor() {
47     this.filteredProducts = this.products;
48     this.listFilter = 'cart';
49   }
50
51   performFilter(filterBy: string): IProduct[] {
52     filterBy = filterBy.toLocaleLowerCase();
53     return this.products.filter((product: IProduct) =>
54       product.productName.toLocaleLowerCase().indexOf(filterBy) !== -1);
55   }
56
57   toggleImage(): void {
58     this.showImage = !this.showImage;
59   }
60
61   ngOnInit(): void {
62     console.log('In OnInit');
63   }
64 }
65

```

Now a new method `performFilter()` is added and in constructor we assing the `filteredProducts` with default value i.e which has the full list of content as filtration will change the data

```

1 import { Component, OnInit } from '@angular/core';
2 import { IProduct } from './product';
3
4 @Component({
5   selector: 'pm-products',
6   templateUrl: './product-list.component.html',
7   styleUrls: ['./product-list.component.css']
8 })
9 export class ProductListComponent implements OnInit {
10   pageTitle: string = 'Product List';
11   imageWidth: number = 50;
12   imageMargin: number = 2;
13   showImage: boolean = false;
14
15   _listFilter: string;
16   get listFilter(): string {
17     return this._listFilter;
18   }
19   set listFilter(value: string) {
20     this._listFilter = value;
21     this.filteredProducts= this.listFilter ? this.performFilter(this.listFilter) : this.products;
22   }
23
24
25   filteredProducts: IProduct[];
26   products: IProduct[] = [
27     {
28       id: 1,
29       name: 'Dad Hat',
30       description: 'A small cotton hat with a large circular logo on the front',
31       price: 22.0,
32       starRating: 4.8,
33       imageUrl: 'http://openclipart.org/image/300px/svg_to_png/73/rejon_Hat.png'
34     },
35     {
36       id: 2,
37       name: 'Curved Claw Steel Hammer',
38       description: 'A curved claw steel hammer',
39       price: 8.9,
40       starRating: 4.8,
41       releaseDate: "May 21, 2016",
42       imageUrl: 'http://openclipart.org/image/300px/svg_to_png/73/rejon_Hammer.png'
43     }
44   ];
45
46   constructor() {
47     this.filteredProducts = this.products;
48     this.listFilter = 'cart';
49   }
50
51   performFilter(filterBy: string): IProduct[] {
52     filterBy = filterBy.toLocaleLowerCase();
53     return this.products.filter((product: IProduct) =>
54       product.productName.toLocaleLowerCase().indexOf(filterBy) !== -1);
55   }
56
57   toggleImage(): void {
58     this.showImage = !this.showImage;
59   }
60
61   ngOnInit(): void {
62     console.log('In OnInit');
63   }
64 }
65

```

In html change let products of products to filteredProducts

The screenshot shows a Visual Studio Code interface with two files open:

- product-list.component.ts**:

```
30   <th>Available</th>
31   <th>Price</th>
32   <th>5 Star Rating</th>
33 </tr>
34 </thead>
35 <tbody>
36   <tr *ngFor='let product of filteredProducts'>
37     <td>
38       <img *ngIf='showImage'
39         [src]='product.imageUrl'
40         [title]='product.productName'
41         [style.width.px]='imageWidth'
42         [style.margin.px]='imageMargin'>
43     </td>
44     <td>{{ product.productName }}</td>
45     <td>{{ product.productCode | lowercase | convertToSpaces: '-' }}</td>
46     <td>{{ product.releaseDate }}</td>
47     <td>{{ product.price | currency:'USD':'symbol':'1.2-2' }}</td>
48     <td>{{ product.starRating }}</td>
49   </tr>
50 </tbody>
51 </table>
52 </div>
53 </div>
54 </div>
55
```
- product-list.component.html**:

```
<thead>
  <tr>
    <th>Available</th>
    <th>Price</th>
    <th>5 Star Rating</th>
  </tr>
</thead>
<tbody>
  <tr *ngFor='let product of filteredProducts'>
    <td>
      <img *ngIf='showImage'
        [src]='product.imageUrl'
        [title]='product.productName'
        [style.width.px]='imageWidth'
        [style.margin.px]='imageMargin'>
    </td>
    <td>{{ product.productName }}</td>
    <td>{{ product.productCode | lowercase | convertToSpaces: '-' }}</td>
    <td>{{ product.releaseDate }}</td>
    <td>{{ product.price | currency:'USD':'symbol':'1.2-2' }}</td>
    <td>{{ product.starRating }}</td>
  </tr>
</tbody>
</table>
</div>
</div>
</div>
```

The right sidebar shows a "Table of Contents" for a Pluralsight course on Angular, specifically the "Angular: Getting Started" section. The contents include:

- Introduction to Components (27m 56s)
- Templates, Interpolation, and Directives (21m 54s)
- Data Binding & Pipes (21m 54s)
- More on Components (3m 8s)
 - Introduction (2m 27s)
 - Defining Interfaces (5m 41s)
 - Encapsulating Component Styles (3m 34s)
 - Using Lifecycle Hooks (4m 16s)
 - Building Custom Pipes (6m 57s)
 - Filtering a List (4m 41s)
 - Checklists and Summary (3m 26s)
- Building Nested Components (25m 49s)
- Services and Dependency Injection (18m 55s)
- Retrieving Data Using HTTP (27m 31s)
- Navigation and Routing

Building Nested Components

Conditions:

Activities Google Chrome ▾

Google News Home | Pluralsight Angular: Getting Start... +

Apr 25 00:43 • https://app.pluralsight.com/player?course=angular-2-getting-started-update&author=deborah-kurata&name=angular-2-getting-started-update-m7&clip=0&mode=live

What Makes a Component Nest-able?



- Its template only manages a fragment of a larger view
- It has a selector
- It optionally communicates with its container

Table of Contents Notes

- Building Nested Components
 - Introduction 3m 5s
 - Building a Nested Component 5m 5s
 - Using a Nested Component 3m 14s
 - Passing Data to a Nested Com... 2m 55s
 - Passing Data from a Componen... 8m 41s
 - Checklists and Summary 2m 48s
- Services and Dependency
 - Injection 18m 50s
- Retrieving Data Using
 - HTTP 27m 31s
- Navigation and Routing
 - Basics 27m 1s
 - Additional Techniques 21m 41s
- Angular Modules 40m 27s
- Building, Testing, and Deploying with the CLI 22m 25s
- Final Words 6m 17s

With the use of directive nested component is added to the template. Nested component receives the information from its parent component using input properties and it outputs its properties to the parent component or container as events

Activities Google Chrome ▾

Google News Home | Pluralsight Angular: Getting Start... +

Apr 25 00:45 • https://app.pluralsight.com/player?course=angular-2-getting-started-update&author=deborah-kurata&name=angular-2-getting-started-update-m7&clip=1&mode=live

Building a Nested Component

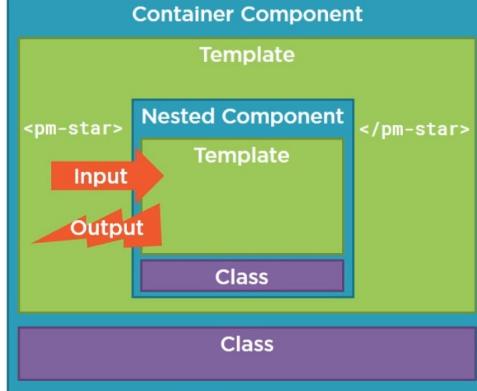


Table of Contents Notes

- Building Nested Components
 - Introduction 3m 5s
 - Building a Nested Component 5m 5s
 - Using a Nested Component 3m 14s
 - Passing Data to a Nested Com... 2m 55s
 - Passing Data from a Componen... 8m 41s
 - Checklists and Summary 2m 48s
- Services and Dependency
 - Injection 18m 50s
- Retrieving Data Using
 - HTTP 27m 31s
- Navigation and Routing
 - Basics 27m 1s
 - Additional Techniques 21m 41s
- Angular Modules 40m 27s
- Building, Testing, and Deploying with the CLI 22m 25s
- Final Words 6m 17s

Sample Application for Nested Component:

Create a new star component as per our requirement(it will show rating as stars, earlier rating was coming in form of decimal numbers)

Below Picture is for sending data from one component to another using Input not output , picture is for reference

Below example is for passing value to the nested component which is inside a container

Activities Google Chrome ▾

Google N × Home | Pluralsight.com ▾ streamin... ▾ M How to s... ▾ M Choosin... ▾ Java 8 St... ▾ HTTP Str... ▾ Streamin... ▾ apigee- ▾ API Man... ▾ AWS Com... ▾ +

Apr 25 22:03 • https://app.pluralsight.com/player?course=angular-2-getting-started-update&author=deborah-kurata&name=angular-2-getting-started-update-m7&clip=4&mode=live

Raising an Event (@Output)

Container Component

Template

<pm-star> **Input** Nested Component Template @Input()

Class

Class

Table of Contents Notes

- Building Nested Components
 - Introduction 3m 5s
 - Building a Nested Component 5m 5s
 - Using a Nested Component 3m 14s
 - Passing Data to a Nested Com... 2m 55s
 - Passing Data from a Componen... 8m 41s
 - Checklists and Summary 2m 48s
- Services and Dependency
 - Injection 18m 50s
- Retrieving Data Using
 - HTTP 27m 31s
- Navigation and Routing
 - Basics 27m 1s
- Navigation and Routing
 - Additional Techniques 21m 41s
- Angular Modules
 - 40m 27s
- Building, Testing, and Deploying with the CLI
 - 22m 25s
- Final Words 6m 17s

Activities Google Chrome ▾

Google N × Home | Pluralsight.com ▾ streamin... ▾ M How to s... ▾ M Choosin... ▾ Java 8 St... ▾ HTTP Str... ▾ Streamin... ▾ apigee- ▾ API Man... ▾ AWS Com... ▾ +

Apr 25 21:41 • https://app.pluralsight.com/player?course=angular-2-getting-started-update&author=deborah-kurata&name=angular-2-getting-started-update-m7&clip=1&mode=live

star.component.ts - APM - Visual Studio Code

File Edit Selection View Go Debug Tasks Help

product-list.component.html TS star.components.ts

```

1 import { Component, OnChanges } from '@angular/core';
2
3 @Component({
4   selector: 'pm-star',
5   templateUrl: './star.component.html',
6   styleUrls: ['./star.component.css']
7 })
8 export class StarComponent implements OnChanges {
9   rating: number = 4;
10  starWidth: number;
11
12  ngOnChanges(): void {
13    this.starWidth = this.rating * 75 / 5;
14  }
15 }

```

star.component.html

```

1 <div class="crop"
2   [style.width.px]= "starWidth"
3   [title]= "rating">
4     <div style="width: 75px">
5       <span class="fa fa-star"></span>
6       <span class="fa fa-star"></span>
7       <span class="fa fa-star"></span>
8       <span class="fa fa-star"></span>
9       <span class="fa fa-star"></span>
10      </div>
11    </div>

```

Table of Contents Notes

- Building Nested Components
 - Introduction 3m 5s
 - Building a Nested Component 5m 5s
 - Using a Nested Component 3m 14s
 - Passing Data to a Nested Com... 2m 55s
 - Passing Data from a Componen... 8m 41s
 - Checklists and Summary 2m 48s
- Services and Dependency
 - Injection 18m 50s
- Retrieving Data Using
 - HTTP 27m 31s
- Navigation and Routing
 - Basics 27m 1s
- Navigation and Routing
 - Additional Techniques 21m 41s
- Angular Modules
 - 40m 27s
- Building, Testing, and Deploying with the CLI
 - 22m 25s
- Final Words 6m 17s

So initially it was a number as shown below

Activities Google Chrome ▾

Google N × Home | Pl × Angular × streamin × M How to s × M Choosin × Java 8St × Java 8St × HTTP Str × Streamin × apigee- × API Man × AWS Con × +

Apr 25 21:53

https://app.pluralsight.com/player?course=angular-2-getting-started-update&author=deborah-kurata&name=angular-2-getting-started-update-m7&clip=2&mode=live

Using a Nested Component as a Directive

product-list.component.ts

```
@Component({
  selector: 'pm-products',
  templateUrl: './product-list.component.html'
})
export class ProductListComponent { }
```

star.component.ts

```
@Component({
  selector: 'pm-star',
  templateUrl: './star.component.html'
})
export class StarComponent {
  rating: number;
  starWidth: number;
}
```

product-list.component.html

```
<td>
  {{ product.starRating | number }}
</td>
```

Table of Contents Notes

- Building Nested Components
 - Introduction 3m 5s
 - Building a Nested Component 5m 5s
 - Using a Nested Component 3m 14s**
 - Passing Data to a Nested Com... 2m 55s
 - Passing Data from a Componen... 8m 41s
 - Checklists and Summary 2m 48s
- Services and Dependency
 - Injection 18m 50s
- Retrieving Data Using
 - HTTP 27m 31s
- Navigation and Routing
 - Basics 27m 11s
 - Additional Techniques 2m 41s
- Angular Modules 40m 27s
- Building, Testing, and Deploying with the CLI 22m 25s
- Final Words 8m 17s

Now it is replaced as

Activities Google Chrome ▾

Google N × Home | Pl × Angular × streamin × M How to s × M Choosin × Java 8St × Java 8St × HTTP Str × Streamin × apigee- × API Man × AWS Con × +

Apr 25 21:53

https://app.pluralsight.com/player?course=angular-2-getting-started-update&author=deborah-kurata&name=angular-2-getting-started-update-m7&clip=2&mode=live

Using a Nested Component as a Directive

product-list.component.ts

```
@Component({
  selector: 'pm-products',
  templateUrl: './product-list.component.html'
})
export class ProductListComponent { }
```

star.component.ts

```
@Component({
  selector: 'pm-star',
  templateUrl: './star.component.html'
})
export class StarComponent {
  rating: number;
  starWidth: number;
}
```

product-list.component.html

```
<td>
  <pm-star></pm-star>
</td>
```

Table of Contents Notes

- Building Nested Components
 - Introduction 3m 5s
 - Building a Nested Component 5m 5s
 - Using a Nested Component 3m 14s**
 - Passing Data to a Nested Com... 2m 55s
 - Passing Data from a Componen... 8m 41s
 - Checklists and Summary 2m 48s
- Services and Dependency
 - Injection 18m 50s
- Retrieving Data Using
 - HTTP 27m 31s
- Navigation and Routing
 - Basics 27m 11s
 - Additional Techniques 2m 41s
- Angular Modules 40m 27s
- Building, Testing, and Deploying with the CLI 22m 25s
- Final Words 8m 17s

Angular: Getting Started

Building Nested Components : Using a Nested Component

product-list.component.html

```

36   <tbody>
37     <tr *ngFor='let product of filteredProducts'>
38       <td>
39         <img *ngIf='showImage'
40           [src]="product.imageUrl"
41           [title]="product.productName"
42           [style.width.px]="imageWidth"
43           [style.margin.px]="imageMargin">
44       </td>
45       <td>{{ product.productName }}</td>
46       <td>{{ product.productCode | lowercase | conv...>
47       <td>{{ product.releaseDate }}</td>
48       <td>{{ product.price | currency:'USD':symbol }}</td>
49       <td><pm-star></pm-star></td>
50     </tr>
51   </tbody>
52 </div>
53 </div>
54 </div>
55 </div>
56 </div>
57 </div>

```

star.component.ts

```

1 import { Component, OnChanges } from '@angular/core';
2
3 @Component({
4   selector: 'pm-star',
5   templateUrl: './star.component.html',
6   styleUrls: ['./star.component.css']
7 })
8 export class StarComponent implements OnChanges {
9   rating: number = 4;
10  starWidth: number;
11
12  ngOnChanges(): void {
13    this.starWidth = this.rating * 75 / 5;
14  }
15}

```

Table of Contents

- Building Nested Components
 - Introduction 3m 5s
 - Building a Nested Component 5m 5s
 - Using a Nested Component 3m 14s
 - Passing Data to a Nested Com... 2m 55s
 - Passing Data from a Compon... 8m 41s
 - Checklists and Summary 2m 48s
- Services and Dependency
 - Injection 18m 50s
- Retrieving Data Using
 - HTTP 27m 31s
- Navigation and Routing
 - Basics 27m 1s
 - Additional Techniques 21m 41s
- Angular Modules 40m 27s
- Building, Testing, and Deploying with the CLI 22m 25s
- Final Words 6m 17s

Currently it will show 5 stars as there is no interlinking between the Star component and product list component.

We need to add `@Input()` decorator (it is a function) in star component that can pass value

Activities

Google N | Home | Pl | Angul | streamin | M How to s | M Choosin | Java 8 St | Java 8 St | HTTP Str | Streamin | apigee- | API Man | AWS Cor | +

Apr 25 21:58 •

https://app.pluralsight.com/player?course=angular-2-getting-started-update&author=deborah-kurata&name=angular-2-getting-started-update-m7&clip=3&mode=live

Table of Contents

- Building Nested Components
 - Introduction 3m 5s
 - Building a Nested Component 5m 5s
 - Using a Nested Component 3m 14s
 - Passing Data to a Nested Com... 2m 55s
 - Passing Data from a Compon... 8m 41s
 - Checklists and Summary 2m 48s
- Services and Dependency
 - Injection 18m 50s
- Retrieving Data Using
 - HTTP 27m 31s
- Navigation and Routing
 - Basics 27m 1s
 - Additional Techniques 21m 41s
- Angular Modules 40m 27s
- Building, Testing, and Deploying with the CLI 22m 25s
- Final Words 6m 17s

Passing Data to a Nested Component (@Input)

product-list.component.ts

```

@Component({
  selector: 'pm-products',
  templateUrl: './product-list.component.html'
})
export class ProductListComponent { }

```

star.component.ts

```

@Component({
  selector: 'pm-star',
  templateUrl: './star.component.html'
})
export class StarComponent {
  @Input() rating: number;
  starWidth: number;
}

```

product-list.component.html

```

<td>
  <pm-star [rating]='product.starRating'>
  </pm-star>
</td>

```

Now the `product.starRating` is mapped with the `rating` and it will reflect at any change as there is `ngOnChanges` life cycle hook attached to it

Angular: Getting Started

Building Nested Components : Passing Data to a Nested Component Using `@Input`

`product-list.component.html`

```

36   <tbody>
37     <tr *ngFor='let product of filteredProducts'>
38       <td>
39         <img *ngIf='showImage'
40           [src]="product.imageUrl"
41           [title]="product.productName"
42           [style.width.px]="imageWidth"
43           [style.margin.px]="imageMargin">
44       </td>
45       <td>{{ product.productName }}</td>
46       <td>{{ product.productCode | lowercase | conv...</td>
47       <td>{{ product.releaseDate }}</td>
48       <td>{{ product.price | currency:'USD':'symbol' }}</td>
49         <p>| <pm-star [rating]='product.starRating'></pm-star>
50       </td>
51     </tr>
52   </tbody>
53 </table>
54 </div>
55 </div>
56 </div>
57 </div>
58 </div>
59 </div>

```

`star.component.ts`

```

1 import { Component, OnChanges, Input } from '@angular/core';
2
3 @Component({
4   selector: 'pm-star',
5   templateUrl: './star.component.html',
6   styleUrls: ['./star.component.css']
7 })
8 export class StarComponent implements OnChanges {
9   @Input() rating: number;
10  starWidth: number;
11
12  ngOnChanges(): void {
13    this.starWidth = this.rating * 75 / 5;
14  }
15}
16

```

Table of Contents

- Building Nested Components
 - Introduction 3m 5s
 - Building a Nested Component 5m 5s
 - Using a Nested Component 3m 14s
 - Passing Data to a Nested Component 2m 55s
 - Passing Data from a Component 8m 41s
 - Checklists and Summary 2m 48s
- Services and Dependency
 - Injection 18m 50s
- Retrieving Data Using
 - HTTP 27m 31s
- Navigation and Routing
 - Basics 27m 11s
 - Additional Techniques 21m 41s
- Angular Modules 40m 27s
- Building, Testing, and Deploying with the CLI 22m 25s
- Final Words 6m 17s

Passing Value from nested component to Container

Data can be passed to its Container only through an event using `EventEmitter` and with `@Output` decorator

Raising an Event (`@Output`)

`product-list.component.ts`

```

@Component({
  selector: 'pm-products',
  templateUrl: './product-list.component.html'
})
export class ProductListComponent { }

```

`star.component.ts`

```

@Component({
  selector: 'pm-star',
  templateUrl: './star.component.html'
})
export class StarComponent {
  @Input() rating: number;
  starWidth: number;
  @Output() notify: EventEmitter<string> = new EventEmitter<string>();
}

```

`product-list.component.html`

```

<td>
  <pm-star [rating]='product.starRating'>
  </pm-star>
</td>

```

Table of Contents

- Building Nested Components
 - Introduction 3m 5s
 - Building a Nested Component 5m 5s
 - Using a Nested Component 3m 14s
 - Passing Data to a Nested Component 2m 55s
 - Passing Data from a Component 8m 41s
 - Checklists and Summary 2m 48s
- Services and Dependency
 - Injection 18m 50s
- Retrieving Data Using
 - HTTP 27m 31s
- Navigation and Routing
 - Basics 27m 11s
 - Additional Techniques 21m 41s
- Angular Modules 40m 27s
- Building, Testing, and Deploying with the CLI 22m 25s
- Final Words 6m 17s

We will add an event to the star component and on click it will call the event `onClick` which in turn will call its `notify` event in `@Output`

Angular: Getting Started

Building Nested Components : Passing Data from a Component Using @Output

Raising an Event (@Output)

```
product-list.component.ts
```

```
@Component({
  selector: 'pm-products',
  templateUrl: './product-list.component.html'
})
export class ProductListComponent { }
```

```
star.component.ts
```

```
@Component({
  selector: 'pm-star',
  templateUrl: './star.component.html'
})
export class StarComponent {
  @Input() rating: number;
  starWidth: number;
  @Output() notify: EventEmitter<string> = new EventEmitter<string>();

  onClick() {
    this.notify.emit('clicked!');
  }
}
```

```
product-list.component.html
```

```
<td>
  <pm-star [rating]='product.starRating'>
  </pm-star>
</td>
```

```
star.component.html
```

```
<div (click)='onClick()'>
  ... stars ...
</div>
```

Angular: Getting Started

Building Nested Components : Passing Data from a Component Using @Output

Raising an Event (@Output)

```
product-list.component.ts
```

```
@Component({
  selector: 'pm-products',
  templateUrl: './product-list.component.html'
})
export class ProductListComponent {
  onNotify(message: string): void { }
}
```

```
star.component.ts
```

```
@Component({
  selector: 'pm-star',
  templateUrl: './star.component.html'
})
export class StarComponent {
  @Input() rating: number;
  starWidth: number;
  @Output() notify: EventEmitter<string> = new EventEmitter<string>();

  onClick() {
    this.notify.emit('clicked!');
  }
}
```

```
product-list.component.html
```

```
<td>
  <pm-star [rating]='product.starRating'
            (notify)='onNotify($event)'>
  </pm-star>
</td>
```

```
star.component.html
```

```
<div (click)='onClick()'>
  ... stars ...
</div>
```

Implementation

Angular: Getting Started

Building Nested Components : Passing Data from a Component Using @Output

star.component.ts - APM - Visual Studio Code

```

1 <div class="crop"
2   [style.width.px]="starWidth"
3   [title]="rating"
4   |(click)="onClick()">
5   <div style="width: 75px">
6     <span class="fa fa-star"></span>
7     <span class="fa fa-star"></span>
8     <span class="fa fa-star"></span>
9     <span class="fa fa-star"></span>
10    <span class="fa fa-star"></span>
11  </div>
12 </div>

```

```

1 import { Component, OnChanges, Input, EventEmitter, Output }
2
3 @Component({
4   selector: 'pm-star',
5   templateUrl: './star.component.html',
6   styleUrls: ['./star.component.css']
7 })
8 export class StarComponent implements OnChanges {
9   @Input() rating: number;
10  starWidth: number;
11  @Output() ratingClicked: EventEmitter<string> =
12    new EventEmitter<string>();
13
14  ngOnChanges(): void {
15    this.starWidth = this.rating * 75 / 5;
16  }
17
18  onClick(): void {
19    this.ratingClicked.emit(`The rating ${this.rating} was clicked!`);
20  }
21 }

```

Table of Contents

- Building Nested Components
 - Introduction
 - Building a Nested Component
 - Using a Nested Component
 - Passing Data to a Nested Component
 - Passing Data from a Component
 - Checklists and Summary
- Services and Dependency
 - Injection
- Retrieving Data Using
 - HTTP
- Navigation and Routing
 - Basics
 - Additional Techniques
- Angular Modules
- Building, Testing, and Deploying with the CLI
- Final Words

We need to pass value to the onRatingClicked() in the below example

product-list.component.html - APM - Visual Studio Code

```

36   <tbody>
37     <tr *ngFor='let product of filteredProducts'>
38       <td>
39         <img *ngIf='showImage'
40           [src]='product.imageUrl'
41           [title]='product.productName'
42           [style.width.px]='imageWidth'
43           [style.margin.px]='imageMargin'>
44       </td>
45       <td>{{ product.productName }}</td>
46       <td>{{ product.productCode | lowercase | convertToPounds }}</td>
47       <td>{{ product.releaseDate }}</td>
48       <td>{{ product.price | currency:'USD':'symbol' }}</td>
49       <td>
50         <pm-star [rating]='product.starRating'
51           |(ratingClicked)=>onRatingClicked(<span>)></pm-star>
52       </td>
53     </tr>
54   </tbody>
55 </table>
56 </div>
57
58 </div>
59 </div>

```

```

1 import { Component, OnChanges, Input, EventEmitter, Output }
2
3 @Component({
4   selector: 'pm-star',
5   templateUrl: './star.component.html',
6   styleUrls: ['./star.component.css']
7 })
8 export class StarComponent implements OnChanges {
9   @Input() rating: number;
10  starWidth: number;
11  @Output() ratingClicked: EventEmitter<string> =
12    new EventEmitter<string>();
13
14  ngOnChanges(): void {
15    this.starWidth = this.rating * 75 / 5;
16  }
17
18  onClick(): void {
19    this.ratingClicked.emit(`The rating ${this.rating} was clicked!`);
20  }
21 }

```

Table of Contents

- Building Nested Components
 - Introduction
 - Building a Nested Component
 - Using a Nested Component
 - Passing Data to a Nested Component
 - Passing Data from a Component
 - Checklists and Summary
- Services and Dependency
 - Injection
- Retrieving Data Using
 - HTTP
- Navigation and Routing
 - Basics
 - Additional Techniques
- Angular Modules
- Building, Testing, and Deploying with the CLI
- Final Words

It is passed with the \$event

```

<tbody>
  <tr *ngFor='let product of filteredProducts'>
    <td>
      <img *ngIf='showImage'
           [src]="product.imageUrl"
           [title]="product.productName"
           [style.width.px]="imageWidth"
           [style.margin.px]="imageMargin">
    </td>
    <td>{{ product.productName }}</td>
    <td>{{ product.productCode | lowercase | conver...</td>
    <td>{{ product.releaseDate }}</td>
    <td>{{ product.price | currency:'USD':'symbol':</td>
      <pm-star [rating]="product.starRating"
              (ratingClicked)="onRatingClicked($event)">
    </td>
  </tr>
</tbody>
</table>
</div>
</div>
<div>

```

```

1 import { Component, OnChanges, Input, EventEmitter, Output
2
3 @Component({
4   selector: 'pm-star',
5   templateUrl: './star.component.html',
6   styleUrls: ['./star.component.css']
7 })
8 export class StarComponent implements OnChanges {
9   @Input() rating: number;
10  starWidth: number;
11  @Output() ratingClicked: EventEmitter<string> = new EventEmitter<string>();
12
13 ngOnChanges(): void {
14   this.starWidth = this.rating * 75 / 5;
15 }
16
17 onClick(): void {
18   this.ratingClicked.emit(`The rating ${this.rating} was clicked!`);
19 }
20
21 }
22

```

Now changes needs to be done in the productList Component

```

<tbody>
  <tr *ngFor='let product of filteredProducts'>
    <td>
      <img *ngIf='showImage'
           [src]="product.imageUrl"
           [title]="product.productName"
           [style.width.px]="imageWidth"
           [style.margin.px]="imageMargin">
    </td>
    <td>{{ product.productName }}</td>
    <td>{{ product.productCode | lowercase | conver...</td>
    <td>{{ product.releaseDate }}</td>
    <td>{{ product.price | currency:'USD':'symbol':</td>
      <pm-star [rating]="product.starRating"
              (ratingClicked)="onRatingClicked($event)">
    </td>
  </tr>
</tbody>
</table>
</div>
</div>
<div>

```

```

47
48 constructor() {
49   this.filteredProducts = this.products;
50   this.listFilter = 'cart';
51 }
52
53 onRatingClicked(message: string): void {
54   this.pageTitle = 'Product List: ' + message;
55 }
56
57 performFilter(filterBy: string): IProduct[] {
58   filterBy = filterBy.toLocaleLowerCase();
59   return this.products.filter((product: IProduct) =>
60     product.productName.toLocaleLowerCase().indexOf(filterBy) !== -1
61   );
62 }
63
64 toggleImage(): void {
65   this.showImage = !this.showImage;
66 }
67
68 ngOnInit(): void {
69   console.log('In OnInit');
70 }
71

```

Service and Dependency Injection

Service can be called as an object but in that case it cannot be shared accross

How Does It Work?

Service

```
export class myService {}
```

Component

```
let svc = new myService();
```

SVC

Table of Contents

- Building Nested Components
- Services and Dependency Injection
- Introduction
- How Does It Work? (2m 32s)
- Building a Service
- Registering the Service
- Injecting the Service
- Checklists and Summary
- Retrieving Data Using HTTP
- Navigation and Routing Basics
- Navigation and Routing Additional Techniques
- Angular Modules
- Building, Testing, and Deploying with the CLI
- Final Words

Registering a Service:

Angular: Getting Started

Services and Dependency Injection : Registering the Service

Registering a Service

Root Injector

Service is available throughout the application

Recommended for most scenarios

Component Injector

Service is available ONLY to that component and its child (nested) components

Isolates a service used by only one component

Provides multiple instances of the service

Table of Contents

- Building Nested Components
- Services and Dependency Injection
- Introduction
- How Does It Work?
- Building a Service
- Registering the Service
- Injecting the Service
- Checklists and Summary
- Retrieving Data Using HTTP
- Navigation and Routing Basics
- Navigation and Routing Additional Techniques
- Angular Modules
- Building, Testing, and Deploying with the CLI
- Final Words

Root Injector: Here we have added providedIn: 'root' so it will be shared accross the application

Activities Google Chrome ▾ Apr 25 23:25 ●

Google Ne... angular Se... Angular How to str... Choosing Java 8 Str... Java 8 Str... HTTP Str... Streaming apigee- Go API Manag... AWS Cons... +

https://app.pluralsight.com/player?course=angular-2-getting-started-update&author=deborah-kurata&name=angular-2-getting-started-update-m8&clip=3&mode=live

Registering a Service - Root Application

Table of Contents Notes

Building Nested Components 25m 49s

Services and Dependency Injection 18m 50s

- Introduction 1m 49s
- How Does It Work? 2m 32s
- Building a Service 2m 51s
- Registering the Service 4m 14s
- Injecting the Service 4m 59s
- Checklists and Summary 2m 23s

Retrieving Data Using HTTP 27m 31s

Navigation and Routing Basics 21m 1s

Navigation and Routing Additional Techniques 21m 41s

Angular Modules 40m 27s

Building, Testing, and Deploying with the CLI 22m 25s

Final Words 6m 11s

```
product.service.ts
import { Injectable } from '@angular/core'

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

  getProducts(): IProduct[] {
  }

}
```

Component Injector: We add it in specific component as providers: [ProductService] it will be available to the same component and its child component.

Activities Google Chrome ▾ Apr 25 23:27 ●

Google Ne... angular Se... Angular How to str... Choosing Java 8 Str... Java 8 Str... HTTP Str... Streaming apigee- Go API Manag... AWS Cons... +

https://app.pluralsight.com/player?course=angular-2-getting-started-update&author=deborah-kurata&name=angular-2-getting-started-update-m8&clip=3&mode=live

product.service.ts

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

product-list.component.ts

```
@Component({
  templateUrl: './product-list.component.html',
  providers: [ProductService]
})
export class ProductListComponent { }
```

Table of Contents Notes

Building Nested Components 25m 49s

Services and Dependency Injection 18m 50s

- Introduction 1m 49s
- How Does It Work? 2m 32s
- Building a Service 2m 51s
- Registering the Service 4m 14s
- Injecting the Service 4m 59s
- Checklists and Summary 2m 23s

Retrieving Data Using HTTP 27m 31s

Navigation and Routing Basics 21m 1s

Navigation and Routing Additional Techniques 21m 41s

Angular Modules 40m 27s

Building, Testing, and Deploying with the CLI 22m 25s

Final Words 6m 11s

Injecting Service in a Component

Activities Google Chrome ▾

Apr 25 23:31

https://app.pluralsight.com/player?course=angular-2-getting-started-update&author=deborah-kurata&name=angular-2-getting-started-update-m8&clip=4&mode=live

Injecting the Service

product-list.component.ts

```
...
import { ProductService } from './product.service';

@Component({
  selector: 'pm-products',
  templateUrl: './product-list.component.html'
})
export class ProductListComponent {

  constructor(private productService: ProductService) {
  }

}
```

Table of Contents

- Building Nested Components
- Services and Dependency Injection
 - Introduction
 - How Does It Work?
 - Building a Service
 - Registering the Service
 - Injecting the Service
 - Checklists and Summary
- Retrieving Data Using HTTP
- Navigation and Routing Basics
- Navigation and Routing Additional Techniques
- Angular Modules
- Building, Testing, and Deploying with the CLI
- Final Words

ngOnInit will be used to call the service.

Constructor is always called before the lifecycle hook method **ngOnInit**

Retrieving Data using HTTP

Observable is used to retrieve data in async way

Activities Google Chrome ▾

Google Netw... angular Se... Angular How to str... Choosing Java 8 Str... Java 8 Str... HTTP Str... Streaming apigee-Go... API Manag... AWS Cons... +

Apr 25 23:48 • https://app.pluralsight.com/player?course=angular-2-getting-started-update&author=deborah-kurata&name=angular-2-getting-started-update-m9&clip=1&mode=live

Angular: Getting Started

Retrieving Data Using HTTP : Observables and Reactive Extensions

Observables and Reactive Extensions



Reactive Extensions (RxJS)

Help manage asynchronous data

Treat events as a collection

- An array whose items arrive asynchronously over time

Subscribe to receive notifications

Are used within Angular

Table of Contents Notes

- Services and Dependency Injection ④ 16m 50s
- Retrieving Data Using HTTP ④ 27m 31s
 - Introduction 1m 22s
 - Observables and Reactive Ext... 5m 47s
 - Sending an HTTP Request 4m 52s
 - Demo: Sending an Http Reque... 3m 18s
 - Exception Handling 2m 31s
 - Subscribing to an Observable 2m 54s
 - Demo: Subscribing to an Obs... 3m 54s
 - Checklists and Summary 2m 55s
- Navigation and Routing Basics ④ 27m 1s
 - Basics 27m 1s
- Navigation and Routing Additional Techniques ④ 21m 41s
 - Additional Techniques 21m 41s
- Angular Modules ④ 40m 27s
 - Angular Modules 40m 27s
- Building, Testing, and Deploying with the CLI ④ 22m 53s
 - Building, Testing, and Deploying with the CLI 22m 53s
- Final Words ④ 6m 7s
 - Final Words 6m 7s

ADD NOTE

1:06 / 5:47

CC 1.0x

Speaker Volume

Screen Share

Full Screen

Activities Google Chrome ▾ Apr 25 23:51 ●

Google Ne × angular Se × M How to str × M Choosing × Java 8 Str × Java 8 Str × HTTP Str × Streaming × apigee-Go × API Manag × AWS Cons × +

https://app.pluralsight.com/player?course=angular-2-getting-started-update&author=deborah-kurata&name=angular-2-getting-started-update-m9&clip=18&mode=live

Angular: Getting Started

Retrieving Data Using HTTP : Observables and Reactive Extensions

Observable Operators



Methods on observables that compose new observables

Transform the source observable in some way

Process each value as it is emitted

Examples: map, filter, take, merge, ...

Table of Contents Notes

- Services and Dependency Injection 18m 50s
- Retrieving Data Using HTTP 27m 11s
 - Introduction 1m 22s
 - Observables and Reactive Ext... 5m 47s
 - Sending an HTTP Request 4m 52s
 - Demo: Sending an Http Reque... 3m 18s
 - Exception Handling 2m 31s
 - Subscribing to an Observable 2m 47s
 - Demo: Subscribing to an Obs... 3m 54s
 - Checklists and Summary 2m 55s
- Navigation and Routing Basics 27m 11s
 - Basics
- Navigation and Routing Additional Techniques 21m 41s
 - Additional Techniques
- Angular Modules 40m 27s
 - Angular Modules
- Building, Testing, and Deploying with the CLI 26m 25s
 - Building, Testing, and Deploying with the CLI
- Final Words 6m 17s
 - Final Words

ADD NOTE

1:37 / 5:47

Speaker icon

Volume icon

Closed caption icon

1.0x

Settings icon

Share icon

Map operator allows to transform the incoming data as shown below, 1-> is transformed to 10 and so on. The argument to the map function is an => function

Activities Google Chrome ▾

Apr 25 23:52 •

Google Ne... | angular Se... | Angular | How to str... | Choosing | Java 8 Str... | HTTP Str... | Streaming | apigee+ Gr... | API Manag... | AWS Cons... | +

https://app.pluralsight.com/player?course=angular-2-getting-started-update&author=deborah-kurata&name=angular-2-getting-started-update-m9&clip=1&mode=live

Observables

Interactive diagrams of Rx Observables

```
graph LR; 1((1)) --> 2((2)); 2 --> 3((3)); 3 -- "map(x => 10 * x)" --> 10((10)); 10 --> 20((20)); 20 --> 30((30));
```

Table of Contents Notes

- Services and Dependency Injection 18m 50s
- Retrieving Data Using HTTP 27m 31s
 - Introduction 1m 22s
 - Observables and Reactive Extensions 5m 47s
 - Sending an HTTP Request 4m 52s
 - Demo: Sending an Http Request 3m 18s
 - Exception Handling 2m 31s
 - Subscribing to an Observable 2m 47s
 - Demo: Subscribing to an Observable 3m 54s
 - Checklists and Summary 2m 55s
- Navigation and Routing Basics 27m 1s
- Navigation and Routing Additional Techniques 21m 41s
- Angular Modules 40m 27s
- Building, Testing, and Deploying with the CLI 22m 25s
- Final Words 6m 17s

Activities Google Chrome ▾

Apr 25 23:54 •

Google Ne... | angular Se... | Angular | How to str... | Choosing | Java 8 Str... | HTTP Str... | Streaming | apigee+ Gr... | API Manag... | AWS Cons... | +

https://app.pluralsight.com/player?course=angular-2-getting-started-update&author=deborah-kurata&name=angular-2-getting-started-update-m9&clip=1&mode=live

Angular: Getting Started

Retrieving Data Using HTTP : Observables and Reactive Extensions

Composing Operators

Compose operators with the pipe method
Often called "pipeable operators"

Table of Contents Notes

- Services and Dependency Injection 18m 50s
- Retrieving Data Using HTTP 27m 31s
 - Introduction 1m 22s
 - Observables and Reactive Extensions 5m 47s
 - Sending an HTTP Request 4m 52s
 - Demo: Sending an Http Request 3m 18s
 - Exception Handling 2m 31s
 - Subscribing to an Observable 2m 47s
 - Demo: Subscribing to an Observable 3m 54s
 - Checklists and Summary 2m 55s
- Navigation and Routing Basics 27m 1s
- Navigation and Routing Additional Techniques 21m 41s
- Angular Modules 40m 27s
- Building, Testing, and Deploying with the CLI 22m 25s
- Final Words 6m 17s

Difference between Promise and Observable

The screenshot shows a video player interface with a table of contents on the right. The main content area displays a comparison table:

Promise	Observable
Provides a single future value	Emits multiple values over time
Not lazy	Lazy
Not cancellable	Cancellable
	Supports map, filter, reduce and similar operators

Below the table, there is a play button icon.

**Example to create Service: We need to import HttpClientModule in AppModule
Observable is added as it returns data in async call and in form of IProduct**

The screenshot shows a video player interface with a code editor window. The code in the editor is:

```

...
import { HttpClient } from '@angular/common/http';
import { Observable } from 'rxjs';

@Injectable({
  providedIn: 'root'
})
export class ProductService {
  private productUrl = 'www.myWebService.com/api/products';

  constructor(private http: HttpClient) { }

  getProducts(): Observable<IProduct[]> {
    return this.http.get<IProduct[]>(this.productUrl);
  }
}

```

The line `return this.http.get<IProduct[]>(this.productUrl);` is highlighted with a red rectangle.

To make the json file available they we have mapped it in angular.json file in the below picture

A screenshot of the Visual Studio Code interface. The title bar shows "Angular: Getting Started" and the status bar indicates "Apr 26 23:57". The left sidebar has a "Table of Contents" section with chapters like "Services and Dependency Injection" and "Retrieving Data Using HTTP". The main editor area shows the `angular.json` file with the following content:

```

{
  "$schema": ".node_modules/@angular/cli/lib/config/schema.json",
  "version": 1,
  "newProjectRoot": "projects",
  "projects": {
    "APM": {
      "root": "",
      "sourceRoot": "src",
      "projectType": "application",
      "prefix": "pm",
      "schematics": {},
      "architect": {
        "build": {
          "builder": "@angular-devkit/build-angular:browser",
          "options": {
            "outputPath": "dist/APM",
            "index": "src/index.html",
            "main": "src/main.ts",
            "polyfills": "src/polyfills.ts",
            "tsConfig": "src/tsconfig.app.json",
            "assets": [
              "src/favicon.ico",
              "src/assets",
              "src/api"
            ],
            "styles": [
              "src/styles.css"
            ]
          }
        }
      }
    }
  }
}

```

The "assets" section is highlighted with a red box.

Exception Handling in Service::

A screenshot of the Visual Studio Code interface. The title bar shows "Angular: Getting Started" and the status bar indicates "Apr 27 00:02". The right sidebar shows the "Table of Contents" with chapters like "Services and Dependency Injection" and "Retrieving Data Using HTTP". The main editor area shows the `product.service.ts` file with the following code:

```

import { HttpClient, HttpErrorResponse } from '@angular/common/http';
import { Observable } from 'rxjs';
import { catchError, tap } from 'rxjs/operators';

getProducts(): Observable<IProduct[]> {
  return this.http.get<IProduct[]>(this.productUrl).pipe(
    tap(data => console.log('All: ' + JSON.stringify(data))),
    catchError(this.handleError)
  );
}

private handleError(err: HttpErrorResponse) {
}

```

The `catchError` block is highlighted with a red box.

Example: working

The screenshot shows a Visual Studio Code interface with a dark theme. On the left is a sidebar with icons for file operations like Open, Save, Find, and Help. The main editor area contains two files: app.module.ts and product.service.ts. The product.service.ts file is open, showing TypeScript code for an observable. A red box highlights the handleError method. The code uses the tap operator to log data to the console and the catchError operator to handle errors. The handleError method checks if the error is an ErrorEvent and logs an error message accordingly. The status bar at the bottom shows the file path as 'product.service.ts', line 36, column 15, and other details like spaces, encoding, and typeScript version.

```
private producerUrl = `api/products/products.json`;  
constructor(private http: HttpClient) { }  
  
getProducts(): Observable<IProduct[]> {  
  return this.http.get<IProduct[]>(this.productUrl).pipe(  
    tap(data => console.log('All: ' + JSON.stringify(data))),  
    catchError(this.handleError)  
);  
  
}  
  
private handleError(err: HttpErrorResponse) {  
  // in a real world app, we may send the server to some remote logging infrastructure  
  // instead of just logging it to the console  
  let errorMessage = '';  
  if (err.error instanceof ErrorEvent) {  
    // A client-side or network error occurred. Handle it accordingly.  
    errorMessage = `An error occurred: ${err.error.message}`;  
  } else {  
    // The backend returned an unsuccessful response code.  
    // The response body may contain clues as to what went wrong,  
    errorMessage = `Server returned code: ${err.status}, error message is: ${err.message}`;  
  }  
  console.error(errorMessage);  
  return throwError(errorMessage);  
}
```

Using observable we retrieve data but we cannot use it until we subscribe it. Observable are lazy and they dont emit values until subscribe is called.

Subscriber method takes up to three arguments each providing a handler function.

1) x.subscribe(nextFn): First argument is often called next function because it processes the next emitted value

2)x.subscribe(nextFn, errorFn): Next argument is error handling function

3)x.subscribe(nextFn, errorFn, completeFn) : Third handler which is executed on completion

The subscribe method returns a subscription

```
let sub= x.subscribe(nextFn, errorFn, completeFn)
```

Angular: Getting Started

Retrieving Data Using HTTP - Subscribing to an Observable

Subscribing to an Observable

```
x.subscribe(nextFn)
x.subscribe(nextFn, errorFn)
x.subscribe(nextFn, errorFn, completeFn)
let sub = x.subscribe(nextFn, errorFn, completeFn)
```

product-list.component.ts

```
ngOnInit(): void {
  this.productService.getProducts().subscribe(
    products => this.products = products,
    error => this.errorMessage = <any>error
  );
}
```

Table of Contents

- Services and Dependency Injection
- Retrieving Data Using HTTP
 - Introduction
 - Observables and Reactive Ext...
 - Sending an HTTP Request
 - Demo: Sending an Http Request
 - Exception Handling
 - Subscribing to an Observable
 - Demo: Subscribing to an Obs...
 - Checklists and Summary
- Navigation and Routing Basics
- Navigation and Routing Additional Techniques
- Angular Modules
- Building, Testing, and Deploying with the CLI
- Final Words

Example: we create a errorMessage:string property and use it in the method inside ngOnInit

```
product-list.component.ts - APM - Visual Studio Code
```

```
constructor(private productService: ProductService) {
```

```
onRatingClicked(message: string): void {
  this.pageTitle = 'Product List: ' + message;
}

performFilter(filterBy: string): IPProduct[] {
  filterBy = filterBy.toLocaleLowerCase();
  return this.products.filter((product: IPProduct) =>
    product.productName.toLocaleLowerCase().indexOf(filterBy) !== -1);
}

toggleImage(): void {
  this.showImage = !this.showImage;
}

ngOnInit(): void {
  this.productService.getProducts().subscribe(
    products => this.products = products,
    error => this.errorMessage = <any>error
  );
  this.filteredProducts = this.products;
}
```

Table of Contents

- Services and Dependency Injection
- Retrieving Data Using HTTP
 - Introduction
 - Observables and Reactive Ext...
 - Sending an HTTP Request
 - Demo: Sending an Http Request
 - Exception Handling
 - Subscribing to an Observable
 - Demo: Subscribing to an Obs...
 - Checklists and Summary
- Navigation and Routing Basics
- Navigation and Routing Additional Techniques
- Angular Modules
- Building, Testing, and Deploying with the CLI
- Final Words

Order of execution

The screenshot shows two files in Visual Studio Code:

- product-list.component.ts** (left):
 - Line 51: `this.filteredProducts = this.products;` (highlighted with a red box and circled with a red number 5).
 - Line 49: `products => this.products = products;` (highlighted with a red box and circled with a red number 2).
 - Line 43: `this.showImage = !this.showImage;` (highlighted with a red box and circled with a red number 1).
 - Line 34: `private handleError(err: HttpErrorResponse) {` (highlighted with a red box and circled with a red number 4).
 - Line 33: `let errorMessage = '';
 - Line 32: `if (err.error instanceof ErrorEvent) {`
 - Line 31: `// A client-side or network error occurred. Handle it accordingly.
 - Line 30: `errorMessage = 'An error occurred: \${err.error.message}';`
 - Line 29: `} else {`
 - Line 28: `// The backend returned an unsuccessful response code.
 - Line 27: `// The response body may contain clues as to what went wrong.
 - Line 26: `errorMessage = 'Server returned code: \${err.status}, error message: \${err.message}'`
- product.service.ts** (right):
 - Line 16: `getProducts(): Observable<IProduct[]> {`
 - Line 17: `return this.http.get<IProduct[]>(this.productUrl).pipe(`
 - Line 18: `tap(data => console.log('All: ' + JSON.stringify(data))),`
 - Line 19: `catchError(this.handleError)`
 - Line 20: `);`

Table of Contents on the right:

- Services and Dependency Injection (18m 50s)
- Retrieving Data Using HTTP (27m 31s)
 - Introduction (1m 22s)
 - Observables and Reactive Extensions (5m 47s)
 - Sending an HTTP Request (4m 52s)
 - Demo: Sending an Http Request (3m 18s)
 - Exception Handling (2m 31s)
 - Subscribing to an Observable (2m 47s)
 - Demo: Subscribing to an Observable (3m 54s)
 - Checklists and Summary (2m 55s)
- Navigation and Routing Basics (27m 1s)
- Navigation and Routing Additional Techniques (21m 41s)
- Angular Modules (40m 27s)
- Building, Testing, and Deploying with the CLI (22m 25s)
- Final Words (6m 17s)

Since Observable does a sync call, in that case if data does not come until it is emitted so filteredProducts will not show up. To make it work we move it above inside the method

The screenshot shows the same files in Visual Studio Code as the previous one, but with a change in the code:

product-list.component.ts (left):

```

42
43
44
45
46
47
48
49
50
51
52
53
54
55
56

```

product.service.ts (right):

```

8   @Injectable({
9     providedIn: 'root'
10  })
11  export class ProductService {
12    private apiUrl = 'api/products/products.json';
13
14    constructor(private http: HttpClient) { }
15
16    getProducts(): Observable<IProduct[]> {
17      return this.http.get<IProduct[]>(this.apiUrl).pipe(
18        tap(data => console.log('All: ' + JSON.stringify(data))),
19        catchError(this.handleError)
20      );
21    }
22
23    private handleError(err: HttpErrorResponse) {
24      // in a real world app, we may send the server to some remote
25      // instead of just logging it to the console
26      let errorMessage = '';
27      if (err.error instanceof ErrorEvent) {
28        // A client-side or network error occurred. Handle it accordingly.
29        errorMessage = 'An error occurred: ${err.error.message}';
30      } else {
31        // The backend returned an unsuccessful response code.
32        // The response body may contain clues as to what went wrong.
33        errorMessage = `Server returned code: ${err.status}, error message: ${err.message}`;
34      }
35    }
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56

```

Table of Contents on the right:

- Services and Dependency Injection (18m 50s)
- Retrieving Data Using HTTP (27m 31s)
 - Introduction (1m 22s)
 - Observables and Reactive Extensions (5m 47s)
 - Sending an HTTP Request (4m 52s)
 - Demo: Sending an Http Request (3m 18s)
 - Exception Handling (2m 31s)
 - Subscribing to an Observable (2m 47s)
 - Demo: Subscribing to an Observable (3m 54s)
 - Checklists and Summary (2m 55s)
- Navigation and Routing Basics (27m 1s)
- Navigation and Routing Additional Techniques (21m 41s)
- Angular Modules (40m 27s)
- Building, Testing, and Deploying with the CLI (22m 25s)
- Final Words (6m 17s)

Navigation and Routing Basics

Command to create a component: `ng g c products/product-detail –flat`

g-> It means generate

c->component

path

--flat-> No folder to be created

Safe navigation operator(?)hecks if the product object has data otherwise the property will throw undefined error as the calls to the http reponse data is async

A screenshot of the Visual Studio Code interface. The left sidebar shows a file tree with 'product-detail.component.html'. The main editor area displays the following code:

```
1 <div class='card'>
2   <div class='card-header'>
3     {{pageTitle + ': ' + product?.productName}}
4   </div>
5 </div>
6
```

An orange callout box points to the question mark in the code, labeled "safe navigation operator". The status bar at the bottom indicates "In 3, Col 37". On the right side, there is a "Table of Contents" pane listing various Angular topics.

How Routing Works:

A screenshot of the Visual Studio Code interface. The left sidebar shows a file tree with 'product-detail.component.html'. The main editor area displays the following code:

```
* spm-root _ghost-c0 ng-version="6.0.5"
  > nav _ngcontent-c0 class="navbar navbar-expand navbar-light bg-light">
    <a href="#" class="navbar-brand">Pluralsight
  </nav>
  <div _ngcontent-c0 class="container">
    <router-outlet _ngcontent-c0></router-outlet>
    <ng-component _ngcontent-c1>
      <div _ngcontent-c1 class="card">
        <div _ngcontent-c1 class="card-header">
          Product List
        </div>
        <div _ngcontent-c1 class="card-body">
          <div _ngcontent-c1 class="row"></div>
          <!--bindings={ "ng-reflect-ng-if": "" }-->
          <div _ngcontent-c1 class="table-responsive">
            <!--bindings={ "ng-reflect-ng-if": "5" }-->
            <table _ngcontent-c1 class="table">
              <thead _ngcontent-c1></thead>
              <tbody _ngcontent-c1></tbody>
            </table>
          </div>
        </div>
        <!--bindings={ "ng-reflect-ng-if": "" }-->
      </ng-component>
    </div>
  </div>
```

The template code is highlighted with a red box. To the right, a vertical green line separates the code from a list of steps: "Configure a route for each component", "Define options/actions", "Tie a route to each option/action", "Activate the route based on user action", and "Activating a route displays the component's view". The status bar at the bottom indicates "In 3, Col 37". On the right side, there is a "Table of Contents" pane listing various Angular topics.

Activities Google Chrome Home | Pluralsight Angular: Getting Started Google News

Apr 27 01:05 • https://app.pluralsight.com/player?course=angular-2-getting-started-update&author=deborah-kurata&name=angular-2-getting-started-update-m10&clip=2&mode=live

How Routing Works

Acme Product Management Home Product List Product List

```
{ path: 'products', component: ProductListComponent }
```

<router-outlet></router-outlet>

```
product-list.component.ts
```

```
import { Component } from '@angular/core';
@Component({
  template: './product-list.component.html'
})
export class ProductListComponent {}
```

Table of Contents Notes

- Services and Dependency Injection 18m 50s
- Retrieving Data Using HTTP 27m 31s
- Navigation and Routing Basics 27m 18s
 - Introduction 1m 32s
 - Generating Code and Handling ... 6m 10s
 - How Routing Works 2m 50s
 - Configuring Routes 4m 21s
 - Demo: Configuring Routes 2m 13s
 - Tying Routes to Actions 3m 48s
 - Placing the Views 2m 38s
 - Checklists and Summary 3m 35s
- Navigation and Routing Additional Techniques 21m 41s
- Angular Modules 40m 27s
- Building, Testing, and Deploying with the CLI 22m 25s
- Final Words 6m 17s

Activities Google Chrome Home | Pluralsight Angular: Getting Started Google News

Apr 27 01:06 • https://app.pluralsight.com/player?course=angular-2-getting-started-update&author=deborah-kurata&name=angular-2-getting-started-update-m10&clip=3&mode=live

Configuring Routes

```
app.module.ts
```

```
...
import { RouterModule } from '@angular/router';

@NgModule({
  imports: [
    BrowserModule,
    FormsModule,
    HttpClientModule,
    RouterModule
  ],
  declarations: [
    ...
  ],
  bootstrap: [ AppComponent ]
})
export class AppModule {}
```

Registers the router service

Table of Contents Notes

- Services and Dependency Injection 18m 50s
- Retrieving Data Using HTTP 27m 31s
- Navigation and Routing Basics 27m 18s
 - Introduction 1m 32s
 - Generating Code and Handling ... 6m 10s
 - How Routing Works 2m 50s
 - Configuring Routes 4m 21s
 - Demo: Configuring Routes 2m 13s
 - Tying Routes to Actions 3m 48s
 - Placing the Views 2m 38s
 - Checklists and Summary 3m 35s
- Navigation and Routing Additional Techniques 21m 41s
- Angular Modules 40m 27s
- Building, Testing, and Deploying with the CLI 22m 25s
- Final Words 6m 17s

The screenshot shows a Pluralsight video player interface. On the left, there's a sidebar with various icons. The main area displays a code editor with the file `app.module.ts`. The code imports `RouterModule` from `@angular/router` and registers it in the `NgModule`'s imports array. A red arrow points from this line to an orange callout box containing the text: "Registers the router service", "Declares the router directives", and "Exposes configured routes". To the right of the video player is a "Table of Contents" sidebar listing various video chapters.

```
...  
import { RouterModule } from '@angular/router';  
  
@NgModule({  
  imports: [  
    BrowserModule,  
    FormsModule,  
    HttpClientModule,  
    RouterModule  
  ],  
  declarations: [  
    ...  
  ],  
  bootstrap: [ AppComponent ]  
})  
export class AppModule {}
```

RouterModule will take an array of below data as RouterModule.forRoot([])

Order in the route matter, more specific routes are kept up

The screenshot shows a Pluralsight video player interface. The main area displays a code editor with the following code:

```
[  
  { path: 'products', component: ProductListComponent },  
  { path: 'products/:id', component: ProductDetailComponent },  
  { path: 'welcome', component: WelcomeComponent },  
  { path: '', redirectTo: 'welcome', pathMatch: 'full' },  
  { path: '**', component: PageNotFoundComponent }  
]
```

To the right is a "Table of Contents" sidebar.

Setting up routes using routerlink directives

The screenshot shows a Pluralsight video player interface. The main content area displays the code for `app.component.ts`:

```

@Component({
  selector: 'pm-root',
  template:
    <ul class='nav navbar-nav'>
      <li><a [routerLink]="/welcome">Home</a></li>
      <li><a [routerLink]="/products">Product List</a></li>
    </ul>
})

```

A red box highlights the `[routerLink]` directives in the navigation links. The video player controls at the bottom indicate it's at 1:09 / 3:48. To the right is a sidebar titled "Table of Contents" with sections like "Services and Dependency Injection", "Retrieving Data Using HTTP", and "Navigation and Routing Basics".

To display the router outlet on a page we use the `<router-outlet></router-outlet>` directive.

Navigation and Routing Additional Technique

Passing Id or any value in the route

The screenshot shows a Pluralsight video player interface. The main content area displays the code for `app.module.ts`:

```

@NgModule({
  imports: [
    ...
    RouterModule.forRoot([
      { path: 'products', component: ProductListComponent },
      { path: 'products/:id', component: ProductDetailComponent },
      { path: 'welcome', component: WelcomeComponent },
      { path: '', redirectTo: 'welcome', pathMatch: 'full' },
      { path: '**', redirectTo: 'welcome', pathMatch: 'full' }
    ])
  ],
  declarations: [...],
  bootstrap: [ AppComponent ]
})
export class AppModule { }

```

A red box highlights the `:id` placeholder in the `path` of the second route. The video player controls at the bottom indicate it's at 0:30 / 2:55. To the right is a sidebar titled "Table of Contents" with sections like "Services and Dependency Injection", "Retrieving Data Using HTTP", and "Navigation and Routing Basics".

The value is passed as shown in the below image. The /products in the url which is called

A screenshot of a browser window showing code from a Pluralsight course. The title is "Passing Parameters to a Route". The code in `product-list.component.html` shows a link with `[routerLink] = ["'/products', product.productId]"`. The `productId` part is highlighted with a red box and has a red arrow pointing to the `:id` placeholder in the `app.module.ts` configuration: `{ path: 'products/:id', component: ProductDetailComponent }`.

We use `ActivateRoute` to get the parameter.

A screenshot of a browser window showing code from a Pluralsight course. The title is "Reading Parameters from a Route". The code in `product-detail.component.ts` shows a constructor with `(private route: ActivatedRoute)` and `console.log(this.route.snapshot.paramMap.get('id'))`. The `id` part is highlighted with a red box and has a red arrow pointing to the `:id` placeholder in the `app.module.ts` configuration: `{ path: 'products/:id', component: ProductDetailComponent }`.

There are two ways of doing this either use a snapshot as shown above or we can use Observable.

*If you need the initial value than use snapshot

* If there is a chance of parameter getting change without leaving the page use Observable.;

Example: If there is next button and after clicking it the id changes in the url then we should use Observable.

In the below image the string name given in the `parametereMap` should have the same value as in the route.

The screenshot shows a Pluralsight video player interface. The main content area displays a slide titled "Reading Parameters from a Route". Below the title is a code snippet for `product-detail.component.ts`:

```
import { ActivatedRoute } from '@angular/router';

constructor(private route: ActivatedRoute) {
  console.log(this.route.snapshot.paramMap.get('id'));
}
```

A red box highlights the line `this.route.snapshot.paramMap.get('id');`. An orange arrow points from this line to another code snippet in a purple box below, titled `app.module.ts`:

```
{ path: 'products/:id', component: ProductDetailComponent }
```

Below example is to setup a back button using a Router Service

The screenshot shows a Pluralsight video player interface. The main content area displays a slide titled "Activating a Route with Code". Below the title is a code snippet for `product-detail.component.ts`:

```
import { Router } from '@angular/router';
...
constructor(private router: Router) { }

onBack(): void {
  this.router.navigate(['/products']);
}
```

Protecting Routes with Gaurds(Route Gaurd)

Types of Route Gaurds:

The screenshot shows a Pluralsight video player interface. The main content area displays a slide titled "Protecting Routes with Guards". To the left of the slide is a green icon of a person in a uniform. On the right side, there is a sidebar titled "Table of Contents" which lists various sections of the course, such as "Services and Dependency Injection", "Retrieving Data Using HTTP", "Navigation and Routing Basics", and "Protecting Routes with Guards". The "Protecting Routes with Guards" section is highlighted with a red dot. At the bottom of the sidebar is a play button icon.

We are creating a route gaurd as a service so adding `@Injectible`

The screenshot shows a Pluralsight video player interface. The main content area displays a code editor window containing the file "product-detail.guard.ts". The code defines a class "ProductDetailGuard" that implements the "CanActivate" interface. The class has a constructor annotated with `@Injectable({ providedIn: 'root' })` and a method `canActivate(): boolean`. The sidebar on the right shows the course's table of contents, with the "Protecting Routes with Guards" section highlighted.

This route should be added to the RouterModule where we want to route based on some condition.

```

app.module.ts

@NgModule({
  imports: [
    ...
    RouterModule.forRoot([
      { path: 'products', component: ProductListComponent },
      { path: 'products/:id',
        canActivate: [ProductDetailGuard],
        component: ProductDetailComponent },
      ...
    ]),
    declarations: [...],
    bootstrap: [ AppComponent ]
  )
export class AppModule { }

```

Below example will help us understand the implementation

Create a gaurd using the command line cli: here g stands for generate and gaurd

```

TS app.module.ts x
14 @NgModule({
15   declarations: [
16     AppComponent,
17     ProductListComponent,
18     ConvertToSpacesPipe,
19     StarComponent,
20     ProductDetailComponent,
21     WelcomeComponent
22   ],
23   imports: [
24     BrowserModule,
25     FormsModule,
26     HttpClientModule,
27     RouterModule.forRoot([
28       { path: 'products', component: ProductListComponent },
29       { path: 'products/:id', component: ProductDetailComponent },
30       { path: 'welcome', component: WelcomeComponent },
31     ])
32   ]
33 })
34 export class AppModule { }

```

guard path/name

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.
PS C:\Users\Deborah\Pluralsight\Angular Getting Started\APM> **ng g g products/product-detail**

It automatically creates a route gaurd with canActivate as method. We can change it as per our requirement.

CanActivate has two paramaters :ActivatedRouteSnapshot = to provide current route information
RouterStateSnapshot= to provide state information of the route

```

1 import { Injectable } from '@angular/core';
2 import { CanActivate, ActivatedRouteSnapshot, RouterStateSnapshot } from '@angular/router';
3 import { Observable } from 'rxjs';
4
5 @Injectable({
6   providedIn: 'root'
7 })
8 export class ProductDetailGuard implements CanActivate {
9   canActivate(
10     next: ActivatedRouteSnapshot,
11     state: RouterStateSnapshot): Observable<boolean> | Promise<boolean> | boolean {
12     return true;
13   }
14 }
15

```

let id = +next.url[1].path;

here products is the [0] th value and [1] is the value we need.

The below route we are checking the value, it should be in number type + in front of the above code converts the url passed string into a number

```

1 import { Injectable } from '@angular/core';
2 import { CanActivate, ActivatedRouteSnapshot, RouterStateSnapshot, Router } from '@angular/router';
3 import { Observable } from 'rxjs';
4
5 @Injectable({
6   providedIn: 'root'
7 })
8 export class ProductDetailGuard implements CanActivate {
9   constructor(private router: Router) {}
10
11   canActivate(
12     next: ActivatedRouteSnapshot,
13     state: RouterStateSnapshot): Observable<boolean> | Promise<boolean> | boolean {
14     let id = +next.url[1].path;
15     if (isNaN(id) || id < 1) {
16       alert("Invalid product Id");
17       this.router.navigate(['/products']);
18       return false;
19     }
20     return true;
21   }
22 }
23

```

Now we can add it in the below location

canActivate has array type data and we can add multiple route gaurds to it , in the current scenario we have only one route gaurd

```

14
15 @NgModule({
16   declarations: [
17     AppComponent,
18     ProductListComponent,
19     ConvertToSpacesPipe,
20     StarComponent,
21     ProductDetailComponent,
22     WelcomeComponent
23   ],
24   imports: [
25     BrowserModule,
26     FormsModule,
27     HttpClientModule,
28     RouterModule.forRoot([
29       { path: 'products', component: ProductListComponent },
30       { path: 'products/:id',
31         canActivate: [ ProductDetailGuard ],
32         component: ProductDetailComponent },
33       { path: 'welcome', component: WelcomeComponent },
34       { path: '', redirectTo: 'welcome', pathMatch: 'full' },
35       { path: '**', redirectTo: 'welcome', pathMatch: 'full' }
36     ])
37   ],
38   bootstrap: [AppComponent]
39 })
40 export class AppModule {}
```

Ln 31, Col 45 Spaces: 2 UTF-8 CRLF TypeScript 2.9.1

For better understanding we can go with the below tutorial on routing

Learning More

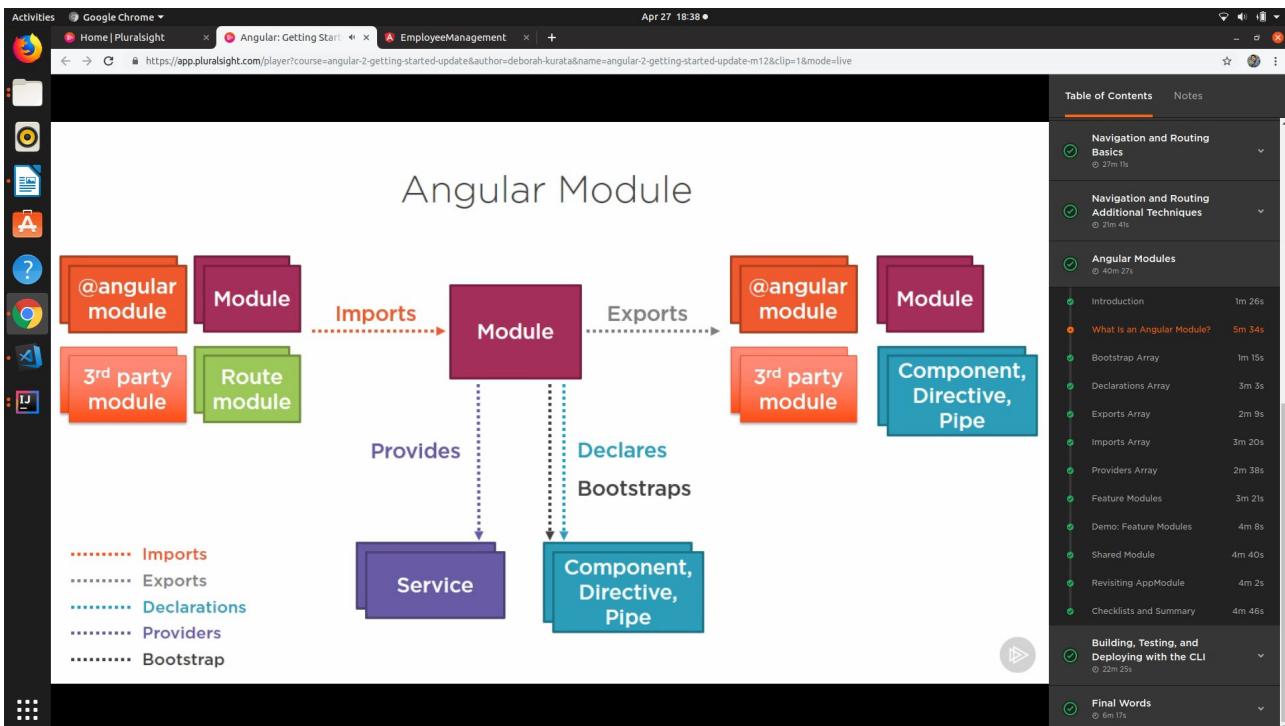


"Angular Routing"

- Required, optional, & query parameters
- Prefetching with route resolvers
- Child and secondary router outlets
- Router guards
- Lazy loading

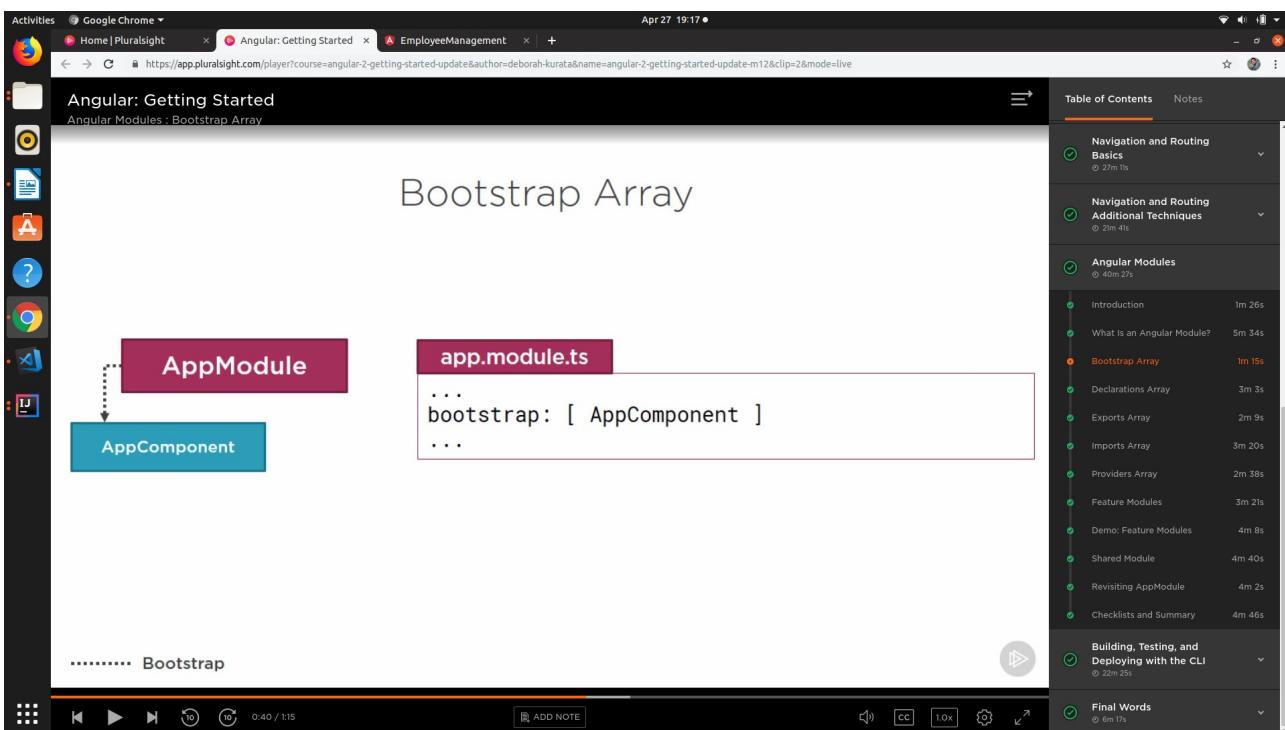
Angular Module

Angular module can be loaded eagerly and lazy to optimize the performance. In the current tutorial we have implemented the eagerly loaded one.
Lazy loading one is in the routing tutorial



Bootstrap Array:

Bootstrapping must be done of at least one module



The screenshot shows a Pluralsight video player interface. The main content area displays the title "Bootstrap Array Truth #1" in large green text, followed by the subtitle "Every application must bootstrap at least one component, the root application component." To the right is a "Table of Contents" sidebar with a list of video topics and their durations.

Topic	Duration
Navigation and Routing Basics	0 27m 1s
Navigation and Routing Additional Techniques	0 27m 4s
Angular Modules	0 40m 27s
Introduction	1m 26s
What is an Angular Module?	5m 34s
Bootstrap Array	1m 15s
Declarations Array	3m 3s
Exports Array	2m 9s
Imports Array	3m 20s
Providers Array	2m 38s
Feature Modules	3m 21s
Demo: Feature Modules	4m 8s
Shared Module	4m 40s
Revisiting AppModule	4m 2s
Checklists and Summary	4m 46s
Building, Testing, and Deploying with the CLI	0 22m 25s
Final Words	0 6m 17s

The screenshot shows a Pluralsight video player interface. The main content area displays the title "Bootstrap Array Truth #2" in large green text, followed by the subtitle "The bootstrap array should only be used in the root application module, AppModule." To the right is a "Table of Contents" sidebar with a list of video topics and their durations.

Topic	Duration
Navigation and Routing Basics	0 27m 1s
Navigation and Routing Additional Techniques	0 27m 4s
Angular Modules	0 40m 27s
Introduction	1m 26s
What is an Angular Module?	5m 34s
Bootstrap Array	1m 15s
Declarations Array	3m 3s
Exports Array	2m 9s
Imports Array	3m 20s
Providers Array	2m 38s
Feature Modules	3m 21s
Demo: Feature Modules	4m 8s
Shared Module	4m 40s
Revisiting AppModule	4m 2s
Checklists and Summary	4m 46s
Building, Testing, and Deploying with the CLI	0 22m 25s
Final Words	0 6m 17s

Declaration Array:

A diagram illustrating the relationship between a module and its declarations. A red box labeled "Module" has a dashed arrow pointing down to a blue box labeled "Component, Directive, Pipe". The word "Declares" is written vertically between the two boxes. Below the diagram, the text "..... Declarations" is followed by a dotted line.

The code editor shows the `app.module.ts` file with the following content:

```
...
declarations: [
  AppComponent,
  WelcomeComponent,
  ProductListComponent,
  ProductDetailComponent,
  ConvertToSpacesPipe,
  StarComponent
]
...
```

Declarations Array Truth #1

Every component, directive, and pipe we create must belong to **one and only one** Angular module.

Dont add other classes, service or module to the declaration array.

Activities Google Chrome Home | Pluralsight Angular: Getting Started EmployeeManagement

Apr 27 19:19 ● https://app.pluralsight.com/player?course=angular-2-getting-started-update&author=deborah-kurata&name=angular-2-getting-started-update-m12&clip=3&mode=live

Declarations Array Truth #2

Only declare components, directives and pipes.

Table of Contents

- Navigation and Routing Basics
- Navigation and Routing Additional Techniques
- Angular Modules
 - Introduction
 - What is an Angular Module?
 - Bootstrap Array
 - Declarations Array
 - Exports Array
 - Imports Array
 - Providers Array
 - Feature Modules
 - Demo: Feature Modules
 - Shared Module
 - Revisiting AppModule
 - Checklists and Summary
- Building, Testing, and Deploying with the CLI
- Final Words

Table of Contents Notes

Activities Google Chrome Home | Pluralsight Angular: Getting Started EmployeeManagement

Apr 27 19:21 ● https://app.pluralsight.com/player?course=angular-2-getting-started-update&author=deborah-kurata&name=angular-2-getting-started-update-m12&clip=3&mode=live

Declarations Array Truth #3

Never re-declare components, directives, or pipes that belong to another module

Diagram illustrating component declarations:

```
graph TD; subgraph ModuleA [Module A]; direction TB; A[ProductList-Component]; end; subgraph ModuleB [Module B]; direction TB; B[Star-Component]; end; A --> B;
```

Table of Contents

- Navigation and Routing Basics
- Navigation and Routing Additional Techniques
- Angular Modules
 - Introduction
 - What is an Angular Module?
 - Bootstrap Array
 - Declarations Array
 - Exports Array
 - Imports Array
 - Providers Array
 - Feature Modules
 - Demo: Feature Modules
 - Shared Module
 - Revisiting AppModule
 - Checklists and Summary
- Building, Testing, and Deploying with the CLI
- Final Words

Table of Contents Notes

They can only be shared when exported

Activities Google Chrome Home | Pluralsight Angular: Getting Started EmployeeManagement

Apr 27 19:22 • https://app.pluralsight.com/player?course=angular-2-getting-started-update&author=deborah-kurata&name=angular-2-getting-started-update-m12&clip=3&mode=live

Declarations Array Truth #4

All declared components, directives, and pipes are private by default.

They are only accessible to other components, directives, and pipes declared in the same module.

Module B
Star-Component
----- Exports
----- Declarations

Table of Contents

- Navigation and Routing Basics
- Navigation and Routing Additional Techniques
- Angular Modules
 - Introduction
 - What is an Angular Module?
 - Bootstrap Array
 - Declarations Array**
 - Exports Array
 - Imports Array
 - Providers Array
 - Feature Modules
 - Demo: Feature Modules
 - Shared Module
 - Revisiting AppModule
 - Checklists and Summary
- Building, Testing, and Deploying with the CLI
- Final Words

It means directive in the left html will look for selector of star component will bw pulled from th module in which it is defined

Activities Google Chrome Home | Pluralsight Angular: Getting Started EmployeeManagement

Apr 27 19:24 • https://app.pluralsight.com/player?course=angular-2-getting-started-update&author=deborah-kurata&name=angular-2-getting-started-update-m12&clip=3&mode=live

Declarations Array Truth #5

The Angular module provides the template resolution environment for its component templates.

```
product-list.component.html
<pm-star ...>
</pm-star>
```

```
star.component.ts
...
@Component({
  selector: 'pm-star',
  template: ...
})
```

Table of Contents

- Navigation and Routing Basics
- Navigation and Routing Additional Techniques
- Angular Modules
 - Introduction
 - What is an Angular Module?
 - Bootstrap Array
 - Declarations Array**
 - Exports Array
 - Imports Array
 - Providers Array
 - Feature Modules
 - Demo: Feature Modules
 - Shared Module
 - Revisiting AppModule
 - Checklists and Summary
- Building, Testing, and Deploying with the CLI
- Final Words

In our below example start component must be declared in the module A as it is shared by both prouduct list and star component.

Another way star component can be imported from the exported module like we have shown B module below but **never both**.

Activities Google Chrome Home | Pluralsight Angular: Getting Started EmployeeManagement

Apr 27 19:26 ● https://app.pluralsight.com/player?course=angular-2-getting-started-update&author=deborah-kurata&name=angular-2-getting-started-update-m12&clip=3&mode=live

Declarations Array Truth #5

The Angular module provides the template resolution environment for its component templates.

Module A

Module B

ProductList-Component

Star-Component

Star-Component

Imports
Exports
Declarations
Providers
Bootstrap

Table of Contents

- Navigation and Routing Basics
- Navigation and Routing Additional Techniques
- Angular Modules
 - Introduction
 - What is an Angular Module?
 - Bootstrap Array
 - Declarations Array**
 - Exports Array
 - Imports Array
 - Providers Array
 - Feature Modules
 - Demo: Feature Modules
 - Shared Module
 - Revisiting AppModule
 - Checklists and Summary
- Building, Testing, and Deploying with the CLI
- Final Words

Exports Array:

This helps to export all the shown features of a module to a different module. Like if we import the below module to another module all the feature in the right side will be available to it.

Activities Google Chrome Home | Pluralsight Angular: Getting Started EmployeeManagement

Apr 27 19:30 ● https://app.pluralsight.com/player?course=angular-2-getting-started-update&author=deborah-kurata&name=angular-2-getting-started-update-m12&clip=4&mode=live

Exports Array

Module

Exports

@angular module

Module

3rd party module

Component, Directive, Pipe

Imports
Exports
Declarations
Providers
Bootstrap

Table of Contents

- Navigation and Routing Basics
- Navigation and Routing Additional Techniques
- Angular Modules
 - Introduction
 - What Is an Angular Module?
 - Bootstrap Array
 - Declarations Array
 - Exports Array**
 - Imports Array
 - Providers Array
 - Feature Modules
 - Demo: Feature Modules
 - Shared Module
 - Revisiting AppModule
 - Checklists and Summary
- Building, Testing, and Deploying with the CLI
- Final Words

Activities

Google Chrome

Home | Pluralsight

Angular: Getting Started

EmployeeManagement

Apr 27 19:33

Table of Contents

- Navigation and Routing
- Basics
- Navigation and Routing
- Additional Techniques
- Angular Modules
 - Introduction
 - What is an Angular Module?
 - Bootstrap Array
 - Declarations Array
 - Exports Array
 - Imports Array
 - Providers Array
 - Feature Modules
 - Demo: Feature Modules
 - Shared Module
 - Revisiting AppModule
 - Checklists and Summary
- Building, Testing, and Deploying with the CLI
- Final Words

Exports Array Truth #1

Export any component, directive, or pipe if other components need it.

Activities

Google Chrome

Home | Pluralsight

Angular: Getting Started

EmployeeManagement

Apr 27 19:33

Table of Contents

- Navigation and Routing
- Basics
- Navigation and Routing
- Additional Techniques
- Angular Modules
 - Introduction
 - What is an Angular Module?
 - Bootstrap Array
 - Declarations Array
 - Exports Array
 - Imports Array
 - Providers Array
 - Feature Modules
 - Demo: Feature Modules
 - Shared Module
 - Revisiting AppModule
 - Checklists and Summary
- Building, Testing, and Deploying with the CLI
- Final Words

Exports Array Truth #2

Re-export modules to re-export their components, directives, and pipes.

If we import a module we import all its features. In the below screenshot if we import the shared module to any module than that module will have Form Module available to it.

Activities Google Chrome Home | Pluralsight Angular: Getting Started EmployeeManagement

Angular: Getting Started
Angular Modules : Exports Array

Exports Array Truth #3

We can re-export something without importing it first.

```

graph TD
    SharedModule[SharedModule] --> FormsModule[FormsModule]
  
```

SharedModule

FormsModule

Imports
Exports
Declarations
Providers
Bootstrap

Table of Contents

- Navigation and Routing
- Basics
- Navigation and Routing
- Additional Techniques
- Angular Modules
- Introduction
- What is an Angular Module?
- Bootstrap Array
- Declarations Array
- Exports Array**
- Imports Array
- Providers Array
- Feature Modules
- Demo: Feature Modules
- Shared Module
- Revisiting AppModule
- Checklists and Summary
- Building, Testing, and Deploying with the CLI
- Final Words

1:41 / 2:09

Services added to the providers array of an angular module are registered with the root application injector making them available for injection in any class of the application

Activities Google Chrome Home | Pluralsight Angular: Getting Started EmployeeManagement

Angular: Getting Started
Angular Modules : Exports Array

Exports Array Truth #4

Never export a service.

```

graph TD
    AppModule[AppModule] --> ProductGuardService[ProductGuardService]
  
```

AppModule

ProductGuardService

Imports
Exports
Declarations
Providers
Bootstrap

Table of Contents

- Navigation and Routing
- Basics
- Navigation and Routing
- Additional Techniques
- Angular Modules
- Introduction
- What is an Angular Module?
- Bootstrap Array
- Declarations Array
- Exports Array**
- Imports Array
- Providers Array
- Feature Modules
- Demo: Feature Modules
- Shared Module
- Revisiting AppModule
- Checklists and Summary
- Building, Testing, and Deploying with the CLI
- Final Words

1:50 / 2:09

Imports Array:

Activities Google Chrome Home | Pluralsight Angular: Getting Started EmployeeManagement

Apr 27 19:42 • https://app.pluralsight.com/player?course=angular-2-getting-started-update&author=deborah-kurata&name=angular-2-getting-started-update-m12&clip=5&mode=live

Imports Array

```

graph TD
    angularModule["@angular module"] -- Imports --> module[Module]
    thirdPartyModule["3rd party module"] -- Imports --> module
    routeModule["Route module"] -- Imports --> module
  
```

app.module.ts

```

...
imports: [
  BrowserModule,
  FormsModule,
  HttpClientModule,
  RouterModule.forRoot([...])
]
...
  
```

Table of Contents

- Navigation and Routing
 - Basics
- Navigation and Routing
 - Additional Techniques
- Angular Modules
 - Introduction
 - What is an Angular Module?
 - Bootstrap Array
 - Declarations Array
 - Exports Array
 - Imports Array
 - Providers Array
 - Feature Modules
 - Demo: Feature Modules
 - Shared Module
 - Revisiting AppModule
 - Checklists and Summary
- Building, Testing, and Deploying with the CLI
- Final Words

Any component, directives and pipes which has export in it will get imported with the module

Activities Google Chrome Home | Pluralsight Angular: Getting Started EmployeeManagement

Apr 27 19:55 • https://app.pluralsight.com/player?course=angular-2-getting-started-update&author=deborah-kurata&name=angular-2-getting-started-update-m12&clip=5&mode=live

Imports Array Truth #1

Importing a module makes available **any exported** components, directives, and pipes from that module.

```

graph TD
    formsModule[FormsModule] -- Imports --> appModule[AppModule]
  
```

Imports

- Exports
- Declarations
- Providers
- Bootstrap

Table of Contents

- Navigation and Routing
 - Basics
- Navigation and Routing
 - Additional Techniques
- Angular Modules
 - Introduction
 - What is an Angular Module?
 - Bootstrap Array
 - Declarations Array
 - Exports Array
 - Imports Array
 - Providers Array
 - Feature Modules
 - Demo: Feature Modules
 - Shared Module
 - Revisiting AppModule
 - Checklists and Summary
- Building, Testing, and Deploying with the CLI
- Final Words

Activities Google Chrome ▾

Angular: Getting Started EmployeeManagement

Apr 27 19:55

https://app.pluralsight.com/player?course=angular-2-getting-started-update&author=deborah-kurata&name=angular-2-getting-started-update-m12&clip=5&mode=live

Imports Array Truth #2

Only import what this module needs.

Table of Contents

- Navigation and Routing Basics
- Navigation and Routing Additional Techniques
- Angular Modules
 - Introduction
 - What is an Angular Module?
 - Bootstrap Array
 - Declarations Array
 - Exports Array
 - Imports Array
 - Providers Array
 - Feature Modules
 - Demo: Feature Modules
 - Shared Module
 - Revisiting AppModule
 - Checklists and Summary
- Building, Testing, and Deploying with the CLI
- Final Words

Activities Google Chrome ▾

Angular: Getting Started EmployeeManagement

Apr 27 19:55

https://app.pluralsight.com/player?course=angular-2-getting-started-update&author=deborah-kurata&name=angular-2-getting-started-update-m12&clip=5&mode=live

Imports Array Truth #2

Only import what this module needs.

Table of Contents

- Navigation and Routing Basics
- Navigation and Routing Additional Techniques
- Angular Modules
 - Introduction
 - What is an Angular Module?
 - Bootstrap Array
 - Declarations Array
 - Exports Array
 - Imports Array
 - Providers Array
 - Feature Modules
 - Demo: Feature Modules
 - Shared Module
 - Revisiting AppModule
 - Checklists and Summary
- Building, Testing, and Deploying with the CLI
- Final Words

In the below rule FormModule is available to SharedModule but when SharedModule is imported to AppModule , FormModule will not be available to AppModule.

It means import are not inherited

If SharedModule reexports the FormModule then its features are available to AppModule

Activities Google Chrome Home | Pluralsight Angular: Getting Started EmployeeManagement

Apr 27 19:58 ● https://app.pluralsight.com/player?course=angular-2-getting-started-update&author=deborah-kurata&name=angular-2-getting-started-update-m12&clip=5&mode=live

Imports Array Truth #3

Importing a module does NOT provide access to its imported modules

```

graph TD
    AppModule[AppModule] --> SharedModule[SharedModule]
    SharedModule --> FormsModule[FormsModule]
    SharedModule --> StarComponent[StarComponent]
    AppModule --> ProductListComponent[ProductListComponent]
  
```

Table of Contents

- Navigation and Routing Basics
- Navigation and Routing Additional Techniques
- Angular Modules
 - Introduction
 - What is an Angular Module?
 - Bootstrap Array
 - Declarations Array
 - Exports Array
 - Imports Array**
 - Providers Array
 - Feature Modules
 - Demo: Feature Modules
 - Shared Module
 - Revisiting AppModule
 - Checklists and Summary
- Building, Testing, and Deploying with the CLI
- Final Words

Providers Array: Starting Angular 6 the discarded approach in the below is not recommended

Activities Google Chrome Home | Pluralsight Angular: Getting Started Angular Modules : Providers Array

Apr 27 20:03 ● https://app.pluralsight.com/player?course=angular-2-getting-started-update&author=deborah-kurata&name=angular-2-getting-started-update-m12&clip=6&mode=live

Providers Array

Module → Service

Imports
Exports
Declarations
Providers
Bootstrap

app.module.ts

```

...
providers: [ ProductService ]
...
  
```

product.service.ts

```

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

Table of Contents

- Navigation and Routing Basics
- Navigation and Routing Additional Techniques
- Angular Modules
 - Introduction
 - What is an Angular Module?
 - Bootstrap Array
 - Declarations Array
 - Exports Array
 - Imports Array**
 - Providers Array**
 - Feature Modules
 - Demo: Feature Modules
 - Shared Module
 - Revisiting AppModule
 - Checklists and Summary
- Building, Testing, and Deploying with the CLI
- Final Words

Activities Google Chrome ▾ Home | Pluralsight ▾ Angular: Getting Started ▾ EmployeeManagement ▾ + Apr 27 20:05 •

<https://app.pluralsight.com/player?course=angular-2-getting-started-update&author=deborah-kurata&name=angular-2-getting-started-update-m12&clip=6&mode=live>

Table of Contents Notes

- Navigation and Routing Basics 27m 1s
- Navigation and Routing Additional Techniques 27m 4s
- Angular Modules 40m 27s
 - Introduction 1m 26s
 - What is an Angular Module? 5m 34s
 - Bootstrap Array 1m 15s
 - Declarations Array 3m 3s
 - Exports Array 2m 9s
 - Imports Array 3m 20s
 - Providers Array 2m 38s
 - Feature Modules 3m 21s
 - Demo: Feature Modules 4m 8s
 - Shared Module 4m 40s
 - Revisiting AppModule 4m 2s
 - Checklists and Summary 4m 46s
- Building, Testing, and Deploying with the CLI 22m 25s
- Final Words 6m 17s

There should be only one instance of the service and it should be application wise singleton. Build a CodeModule and we can even code in the code module constructor to ensure that it is never imported the second time.

Activities Google Chrome ▾ Home | Pluralsight ▾ Angular: Getting Started ▾ EmployeeManagement ▾ + Apr 27 20:06 •

<https://app.pluralsight.com/player?course=angular-2-getting-started-update&author=deborah-kurata&name=angular-2-getting-started-update-m12&clip=6&mode=live>

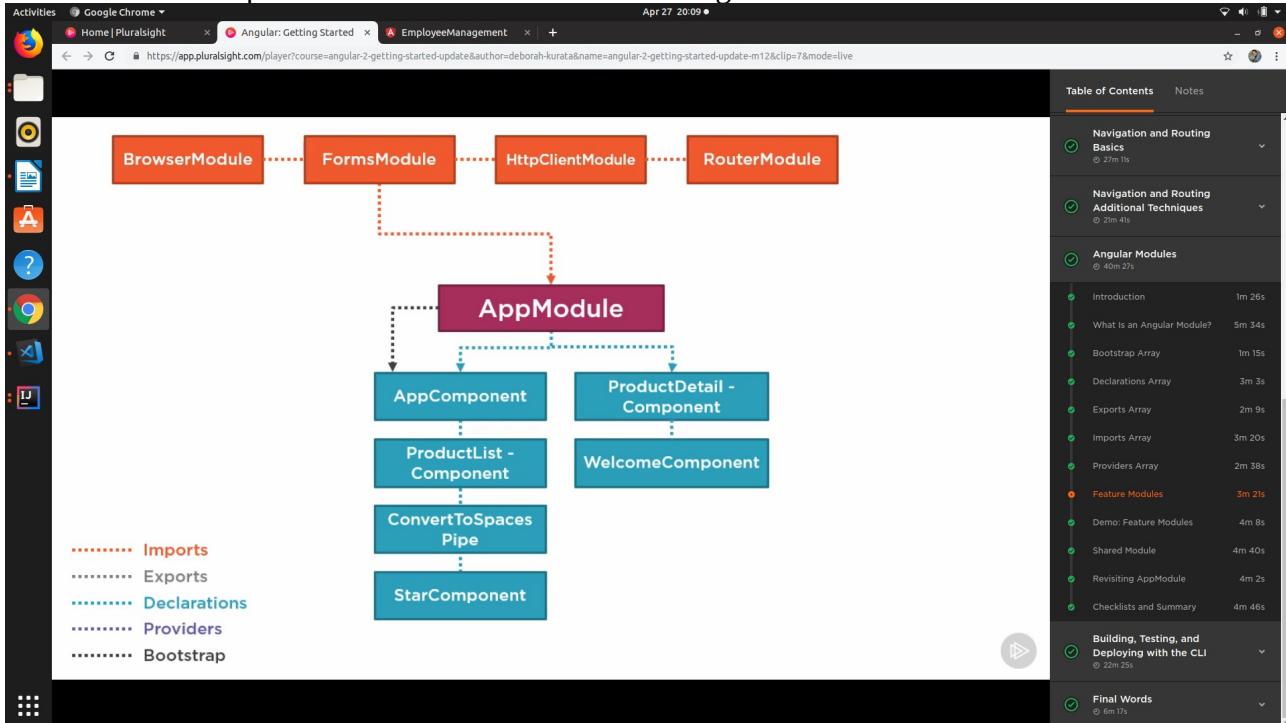
Table of Contents Notes

- Navigation and Routing Basics 27m 1s
- Navigation and Routing Additional Techniques 27m 4s
- Angular Modules 40m 27s
 - Introduction 1m 26s
 - What is an Angular Module? 5m 34s
 - Bootstrap Array 1m 15s
 - Declarations Array 3m 3s
 - Exports Array 2m 9s
 - Imports Array 3m 20s
 - Providers Array 2m 38s
 - Feature Modules 3m 21s
 - Demo: Feature Modules 4m 8s
 - Shared Module 4m 40s
 - Revisiting AppModule 4m 2s
 - Checklists and Summary 4m 46s
- Building, Testing, and Deploying with the CLI 22m 25s
- Final Words 6m 17s

Feature Module:

The below image shows the design of our current application when the size grows it will be more difficult to manage the whole structure. There are some modules which are shared among different components.

We will create a separate feature module to fix this design flaw.

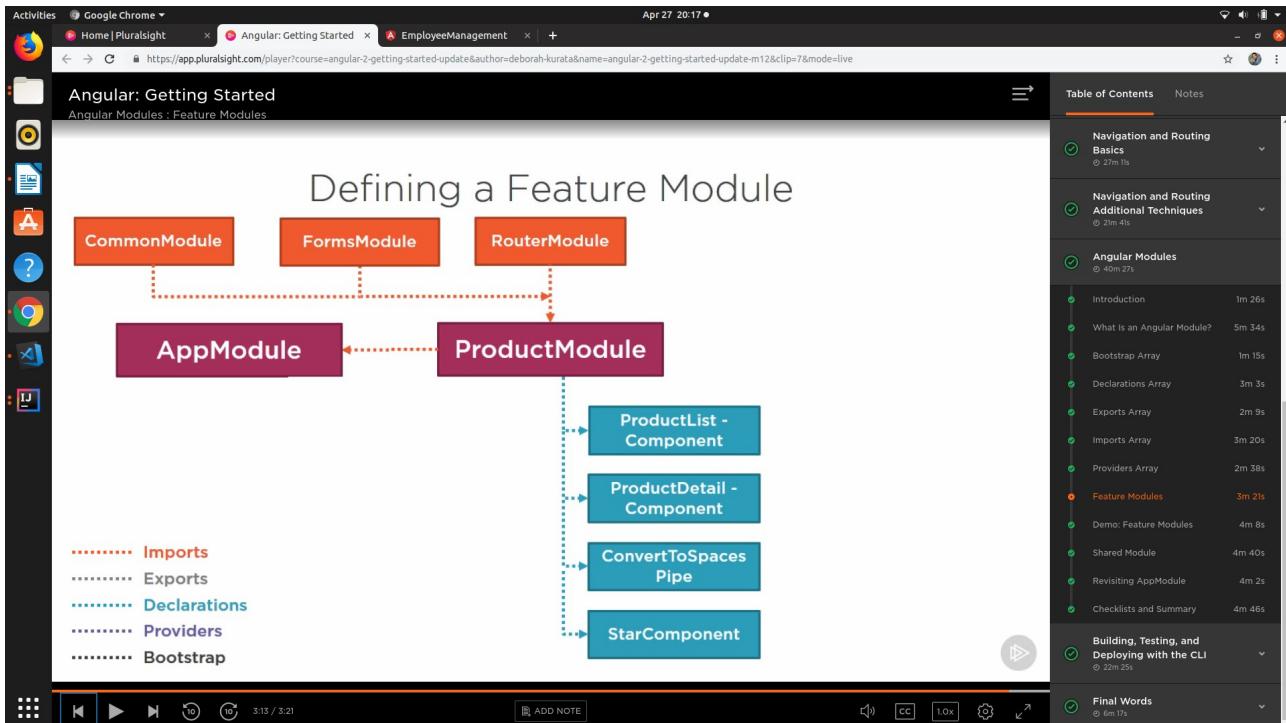


Creating a feature module

Note: We did not include the BrowserModule here as it should be included in the root application module. So we import the CommonModule which exposes the ngIf and ngFor directives.

BrowserModule exports and imports the CommonModule.

Since we have removed all the below details from the AppModule so we need to import the Product Module into the AppModule



Example of the Above explanation:

Below image is of the existing AppModule

```

15 @NgModule({
16   declarations: [
17     AppComponent,
18     ProductListComponent,
19     ConvertToSpacesPipe,
20     StarComponent,
21     ProductDetailComponent,
22     WelcomeComponent
23   ],
24   imports: [
25     BrowserModule,
26     FormsModule,
27     HttpClientModule,
28     RouterModule.forRoot([
29       { path: 'products', component: ProductListComponent },
30       { path: 'products/:id',
31         canActivate: [ ProductDetailGuard ],
32         component: ProductDetailComponent },
33       { path: 'welcome', component: WelcomeComponent },
34       { path: '', redirectTo: 'welcome', pathMatch: 'full' },
35       { path: '**', redirectTo: 'welcome', pathMatch: 'full' }
36     ])
37   ],
38   bootstrap: [AppComponent]
39 })
40 export class AppModule { }
41

```

Table of Contents

- Navigation and Routing
- Basics
- Navigation and Routing Additional Techniques
- Angular Modules
- Introduction
- What is an Angular Module?
- Bootstrap Array
- Declarations Array
- Exports Array
- Imports Array
- Providers Array
- Feature Modules
- Demo: Feature Modules
- Shared Module
- Revisiting AppModule
- Checklists and Summary
- Building, Testing, and Deploying with the CLI
- Final Words

Now we will create a new product module using a CLI
 ng g m products/product -flat -m app
 m=for module

flat= so that it will not create a seperate folder for product module
 -m=to import module
 app = is the AppModule

Once the command is executed it will add the product module into the app module as shown below

```

15 @NgModule({
16   declarations: [
17     AppComponent,
18     ProductListComponent,
19     ConvertToSpacesPipe,
20     StarComponent,
21     ProductDetailComponent,
22     WelcomeComponent
23   ],
24   imports: [
25     BrowserModule,
26     FormsModule,
27     HttpClientModule,
28     RouterModule.forRoot([
29       { path: 'products', component: ProductListComponent },
30       { path: 'products/:id',
31         canActivate: [ ProductDetailGuard ],
32         component: ProductDetailComponent },
33       { path: 'welcome', component: WelcomeComponent },
34       { path: '', redirectTo: 'welcome', pathMatch: 'full' },
35       { path: '**', redirectTo: 'welcome', pathMatch: 'full' }
36     ]),
37     ProductModule // Red box highlights this line
38   ],
39   bootstrap: [AppComponent]
40 })
41 export class AppModule { }

```

Now we remove the things from app module and add it to the product module

Activities Google Chrome Home | Pluralsight Angular: Getting Started EmployeeManagement

https://app.pluralsight.com/player?course=angular-2-getting-started-update&author=deborah-kurata&name=angular-2-getting-started-update-m12&clip=8&mode=live

File Edit Selection View Go Debug Tasks Help

Table of Contents Notes

Navigation and Routing Basics @ 27m 1s

Navigation and Routing Additional Techniques @ 21m 41s

Angular Modules @ 40m 27s

- Introduction 1m 26s
- What is an Angular Module? 5m 34s
- Bootstrap Array 1m 15s
- Declarations Array 3m 3s
- Exports Array 2m 9s
- Imports Array 3m 20s
- Providers Array 2m 38s
- Feature Modules 3m 21s
- Demo: Feature Modules 4m 8s
- Shared Module 4m 40s
- Revisiting AppModule 4m 2s
- Checklists and Summary 4m 46s

Building, Testing, and Deploying with the CLI @ 22m 25s

Final Words @ 6m 17s

product.module.ts - APM - Visual Studio Code

File Edit Selection View Go Debug Tasks Help

RouterModule.forRoot
- Registers Router service
- Declares router directives
- Exposes configured routes

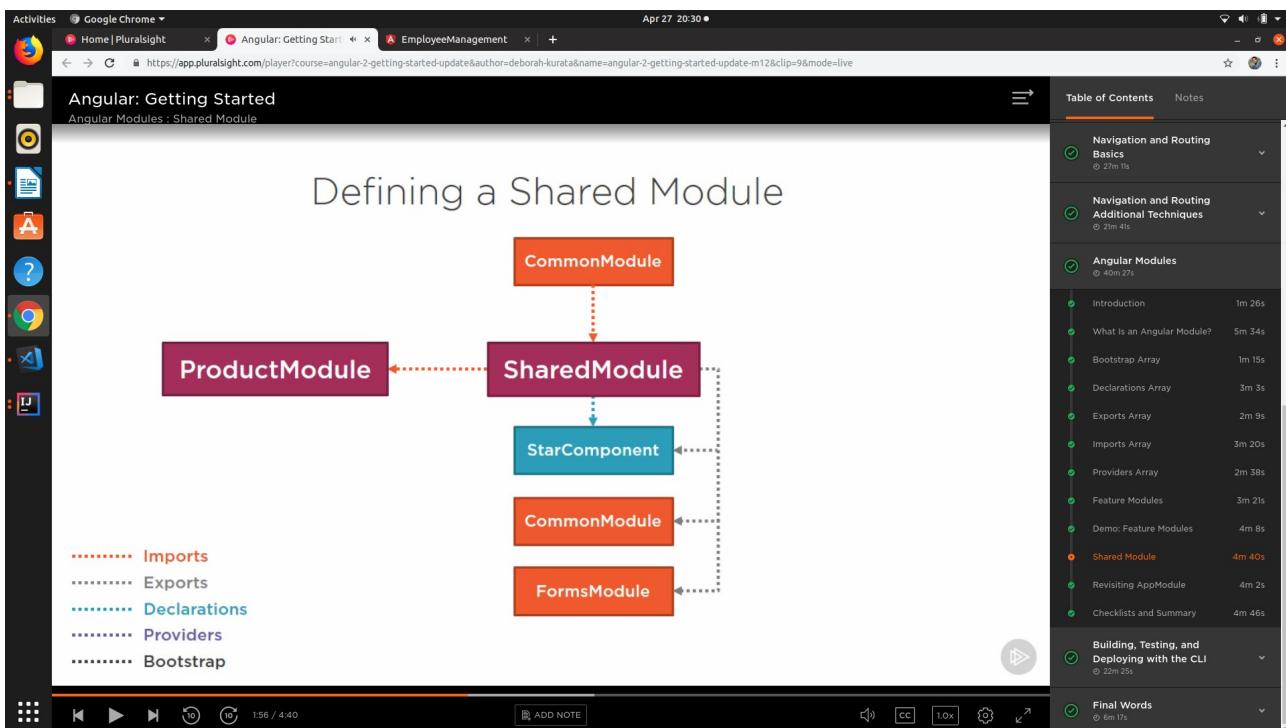
```

1 import { NgModule } from '@angular/core';
2 import { CommonModule } from '@angular/common';
3 import { ProductListComponent } from './product-list.component';
4 import { ProductDetailComponent } from './product-detail.component';
5 import { ConvertToSpacesPipe } from '../shared/convert-to-spaces.pipe';
6 import { StarComponent };
7 import { FormsModuleModule };
8 import { RouterModule };
9 import { ProductDetailGuard };
10
11 @NgModule({
12   imports: [
13     CommonModule,
14     FormsModuleModule,
15     RouterModule.forChild([
16       { path: 'products', component: ProductListComponent },
17       {
18         path: 'products/:id',
19         canActivate: [ProductDetailGuard],
20         component: ProductDetailComponent
21       },
22     ]),
23   ],
24   declarations: [
25     ProductListComponent,
26     ProductDetailComponent,
27     ConvertToSpacesPipe,
28   ],
29 })
30 export class ProductModule {}

```

In 19, Col 26 Spaces: 2 UTF-8 LF TypeScript 2.9.1

Shared Module:



We will create a shared module using the cli
ng g m shared/shared - - flat -m products/product.module

The above command will add the entry of shared module in the ProductModule

We can now put the modules that we need to share in the shared module

```

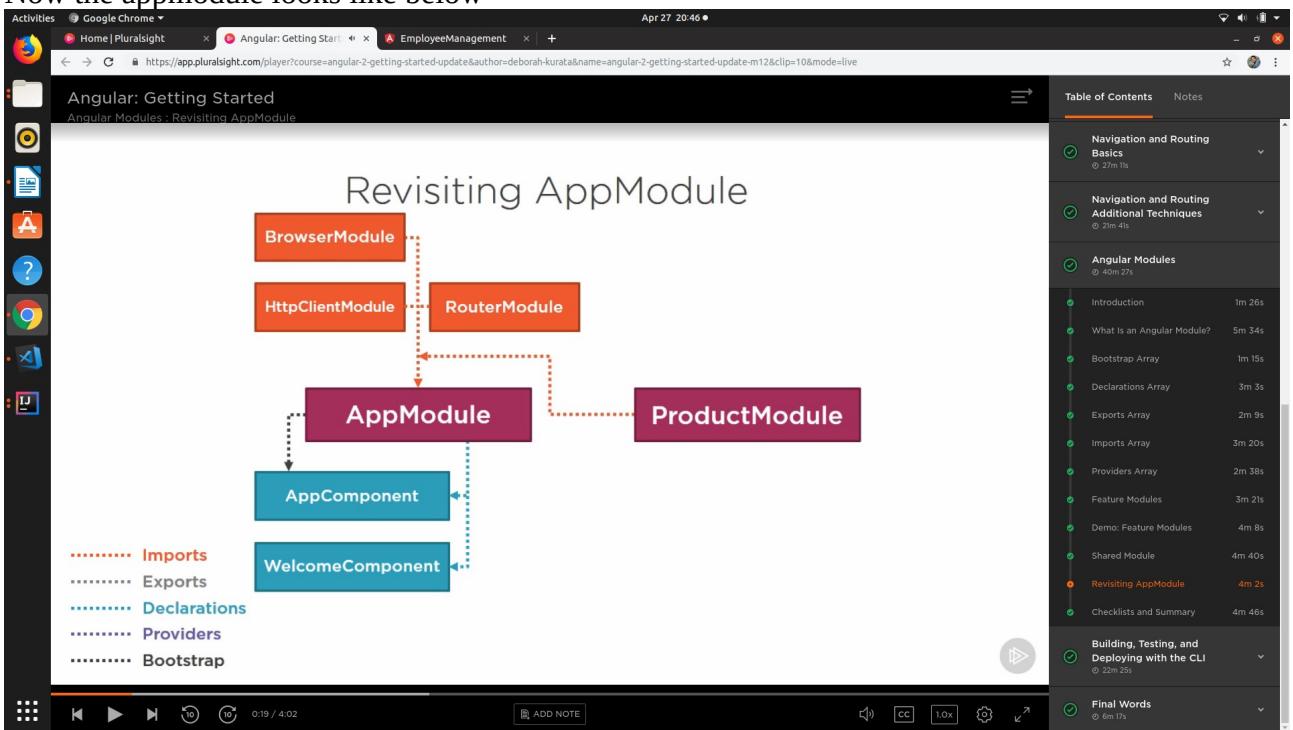
1 import { NgModule } from '@angular/core';
2 import { CommonModule } from '@angular/common';
3 import { StarComponent } from './star.component';
4 import { FormsModule } from '@angular/forms';
5
6 @NgModule({
7   imports: [
8     CommonModule
9   ],
10 declarations: [
11   StarComponent
12 ],
13 exports: [
14   StarComponent,
15   CommonModule,
16   FormsModule
17 ]
18 })
19 export class SharedModule { }
20

```

Table of Contents

- Navigation and Routing
- Basics
- Additional Techniques
- Angular Modules
- Introduction
- What is an Angular Module?
- Bootstrap Array
- Declarations Array
- Exports Array
- Imports Array
- Providers Array
- Feature Modules
- Demo: Feature Modules
- Shared Module
- Revisiting AppModule
- Checklists and Summary
- Building, Testing, and Deploying with the CLI
- Final Words

Now the appmodule looks like below



Creating a separate RoutingModule:

In the below image we have exported the RoutingModule so that it can be used in other application

```

import { NgModule } from '@angular/core';
import { RouterModule } from '@angular/router';

import { WelcomeComponent } from './home/welcome.component';

@NgModule({
  imports: [
    RouterModule.forRoot([
      { path: 'welcome', component: WelcomeComponent },
      { path: '', redirectTo: 'welcome', pathMatch: 'full' },
      { path: '**', redirectTo: 'welcome', pathMatch: 'full' }
    ])
  ],
  exports: [ RouterModule ]
})
export class AppRoutingModule { };

```

AppRoutingModule must be added below the ProductModule so that the hierarchy is maintained as shown in the right side.

```

@NgModule({
  imports: [
    BrowserModule,
    HttpClientModule,
    ProductModule,
    AppRoutingModule
  ],
  declarations: [ AppComponent, WelcomeComponent ],
  bootstrap: [ AppComponent ]
})
export class AppModule { }

```

If we move RouterModule above the ProductModule the hierarchy will change as

The screenshot shows a Pluralsight video player interface. On the left, there's a sidebar with various icons. The main area has a title "Using the Routing Module". To the right of the title is a code editor window containing `app.module.ts` with the following content:

```
@NgModule({
  imports: [
    BrowserModule,
    HttpClientModule,
    ProductModule,
    AppRoutingModule
  ],
  declarations: [ AppComponent, WelcomeComponent ],
  bootstrap: [ AppComponent ]
})
export class AppModule { }
```

To the right of the code editor is a large orange rounded rectangle containing the following text:

products
products/:id
welcome
"
**

A small play button icon is located at the bottom right of the orange box. The right side of the screen features a "Table of Contents" sidebar with several sections and their durations.

If we move `RoutingModule` above the `ProductModule` the Product will not be available when the routes are created as they will be set up before the `ProductModule` is even initialized.

Note: `RoutingModule` must always be the last to added in the imports of the `AppModule`.

This screenshot is similar to the previous one, showing the same video player interface. The code in `app.module.ts` is identical:

```
@NgModule({
  imports: [
    BrowserModule,
    HttpClientModule,
    AppRoutingModule,
    ProductModule
  ],
  declarations: [ AppComponent, WelcomeComponent ],
  bootstrap: [ AppComponent ]
})
export class AppModule { }
```

However, two specific imports in the `imports` array are highlighted with red arrows pointing to them from the text "products" and "products/:id" in the orange box. The orange box contains the same text as in the previous screenshot:

welcome
"
**
products
products/:id

The route for the product modules must follow the same hierarchy and also it should use `forChild` not `forRoot` to ensure that we do not register the router service provider for the second time.

Activities Google Chrome Home | Pluralsight Angular: Getting Started EmployeeManagement

Apr 27 20:56 • https://app.pluralsight.com/player?course=angular-2-getting-started-update&author=deborah-kurata&name=angular-2-getting-started-update-m12&clip=10&mode=live

Feature Routing Module

```
product-routing.module.ts
```

```
import { NgModule } from '@angular/core';
import { RouterModule } from '@angular/router';
import { ProductListComponent } from './product-list.component';
import { ProductDetailComponent } from './product-detail.component';
import { ProductDetailGuard } from './product-detail.guard';
@NgModule({
  imports: [
    RouterModule.forChild([
      { path: 'products', component: ProductListComponent },
      { path: 'products/:id', canActivate: [ ProductDetailGuard ],
        component: ProductDetailComponent }
    ])
  ],
  exports: [ RouterModule ]
})
export class ProductRoutingModule { }
```

Table of Contents

- Navigation and Routing
 - Basics** (27m 1s)
 - Additional Techniques** (2m 4s)
- Angular Modules** (40m 27s)
 - Introduction (1m 26s)
 - What is an Angular Module? (5m 34s)
 - Bootstrap Array (1m 15s)
 - Declarations Array (3m 3s)
 - Exports Array (2m 9s)
 - Imports Array (3m 20s)
 - Providers Array (2m 38s)
 - Feature Modules (3m 21s)
 - Demo: Feature Modules (4m 8s)
 - Shared Module (4m 40s)
 - Revisiting AppModule** (4m 2s)
 - Checklists and Summary (4m 46s)
- Building, Testing, and Deploying with the CLI (22m 25s)
 - Checklists and Summary (4m 46s)
- Final Words** (8m 17s)

The ProductModule is imported as the same hierarchy in its module

Activities Google Chrome Home | Pluralsight Angular: Getting Started EmployeeManagement

Apr 27 20:59 • https://app.pluralsight.com/player?course=angular-2-getting-started-update&author=deborah-kurata&name=angular-2-getting-started-update-m12&clip=10&mode=live

Using the Routing Module

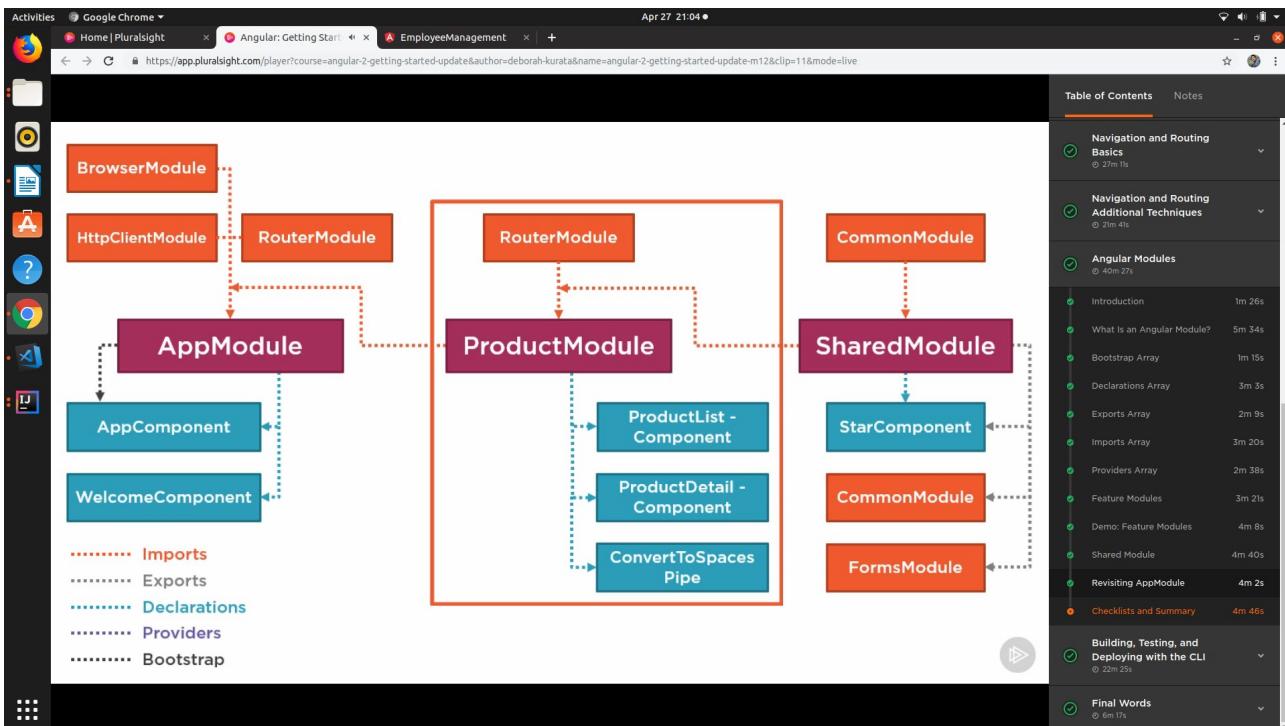
```
product.module.ts
```

```
@NgModule({
  imports: [
    SharedModule,
    ProductRoutingModule
  ],
  declarations: [
    ProductListComponent,
    ProductDetailComponent,
    ConvertToSpacesPipe
  ]
})
export class ProductModule {}
```

Table of Contents

- Navigation and Routing
 - Basics** (27m 1s)
 - Additional Techniques** (2m 4s)
- Angular Modules** (40m 27s)
 - Introduction (1m 26s)
 - What is an Angular Module? (5m 34s)
 - Bootstrap Array (1m 15s)
 - Declarations Array (3m 3s)
 - Exports Array (2m 9s)
 - Imports Array (3m 20s)
 - Providers Array (2m 38s)
 - Feature Modules (3m 21s)
 - Demo: Feature Modules (4m 8s)
 - Shared Module (4m 40s)
 - Revisiting AppModule** (4m 2s)
 - Checklists and Summary (4m 46s)
- Building, Testing, and Deploying with the CLI (22m 25s)
 - Checklists and Summary (4m 46s)
- Final Words** (8m 17s)

Final Arctitecture of the Angular Appilcation with new Feature and Shared Module:



Building, Testing and Deploying using a CLI

In the below screenshot , the project was created using ng new projectname

- 1) e2e= This folder contains start of the end to end test of the application
- 2) node_modules=Contains all the libraries that are installed by CLI as defined in package.json file
- 3) src= It contains the source code of our application

a) app = Here we add all our component, service and other files.

b)assets= Its for any images or assets we want to include

c)environments= It sets up our build environment, by default it generates 2 files one for development and one for production. Cli picks the appropriate file based on the flag we pick.

d)favicon.ico= downloaded with angular cli can be replaced

e)index.html= It is downloaded when application is created

f)karma.config.js = Its a javascript test runner and will be used for unit testing

g)main.ts= It is the file that bootstrap the application(this should not be modified)

h)polyfills.ts= It aids in supporting both evergreen and classic browsers. It help in supporting the functionality writted for modern browser to work in older browser. Uncomment the code section in the file to get the support needed.

i)style.css= Application wise style sheet

j)test.ts= For Testing

k)tsconfig.app.json=These files are for typescript configuration. This if for compiling our code files. It extends our tsconfig.json file.

l)tsconfig.spec.ts=These files are for typescript configuration and this one for test specification

4)editorconfig=Editor configuration

5).gitignore=To ignore from checking into git

6)angular.json=This is a cli configuration file. This file is configure the way cli generates the codes and work with our files. This file hold the app prefix name. We can change it as per our requirement.

It can also be changes using the cli :ng new hello-world --prefix hw

```

1 {
2   "$schema": "./node_modules/@angular/cli/lib/config/schema.json",
3   "version": 1,
4   "newProject": "hello-world",
5   "projects": {
6     "hello-world": {
7       "root": ".",
8       "sourceRoot": "src",
9       "projectType": "application",
10      "prefix": "app",
11      "schematics": {},
12      "architect": {
13        "build": {
14          "builder": "@angular-devkit/build-angular:browser",
15          "options": {
16            "outputPath": "dist/hello-world",
17            "index": "src/index.html",
18            "main": "src/main.ts",
19            "polyfills": "src/polyfills.ts",
20            "tsConfig": "src/tsconfig.app.json",
21            "assets": [
22              "src/favicon.ico",
23              "src/assets"
24            ],
25            "styles": [
26              "src/styles.css"
27            ]
28          }
29        }
30      }
31    }
32  }

```

7) package-lock.json

8) package.json=It has all the files required by the project , dependecies etc.

9) README.md= It has no importance other than some info

10) tsconfig.json=

11) tslint.json=Typescript linter which checks the code on the set of style rules and notify if any rules are not followed. It can be changed based on the team's styling preferences.

Some useful Commands:

1) ng new --help

a) --skip-install=To generate code without installing node packages which can be installed later using npm install.

2) ng serve command: If we use ng serve --help (This shares all the commands used by serve)
--port=to provide alternate port
-o = open in browser

3) ng g/generate command: use ng g --help (It will show what can be created using this command)

4) ng test command: Test can be run using the two commands . The test runner run in watch mode so that if we change the code in between it will re run

- 1) ng test
- 2) ng e2e = end to end testing using protractor

5) ng build: It generated the dist folder in our application

ng build - - prod: It generates the minified version of the files that needs to be deployed on server.

The screenshot shows a Pluralsight video player interface. The main content area displays the title "Angular CLI Checklist: Commands". To the right of the title is a vertical list of Angular CLI commands, each preceded by a green checkmark icon:

- ng help - Displays commands and flags
- ng new - Creates new Angular application
- ng serve - Launches a server
- ng generate - Generates file from blueprint
- ng test - Runs unit tests using Karma
- ng e2e - Runs end to end tests using Protractor
- ng build - Compiles into an output directory

On the left side of the main content area, there is a vertical sidebar with three green checkmark icons, each connected by a horizontal line to one of the first three items in the command list. A large red arrow points from the third checkmark icon towards the "ng e2e" and "ng build" entries. The sidebar also contains other course navigation links like "Introduction", "Angular CLI Overview", and "Final Words". The bottom of the screen shows a standard video player control bar with buttons for play/pause, volume, and progress.

Angular CLI Checklist: `ng generate`

class	<code>ng g cl</code>
component	<code>ng g c</code>
directive	<code>ng g d</code>
enum	<code>ng g e</code>
guard	<code>ng g g</code>
interface	<code>ng g i</code>
module	<code>ng g m</code>
pipe	<code>ng g p</code>
service	<code>ng g s</code>

Table of Contents

- Services and Dependency Injection
- Retrieving Data Using HTTP
- Navigation and Routing Basics
- Navigation and Routing Additional Techniques
- Angular Modules
- Building, Testing, and Deploying with the CLI
- Introduction
- Angular CLI Overview
- ng new
- ng serve
- ng generate
- ng test
- ng build
- Checklists and Summary
- Final Words

RECAP:

Where Do We Put the Html?

Inline Template

```
template: "

# {{pageTitle}}

"
```

Inline Template

```
template: `
<div>
<h1>{{pageTitle}}</h1>
<div>
    My First Component
</div>
</div>`
```

Linked Template

```
templateUrl: './product-list.component.html'
```

ES 2015 Back Ticks

Table of Contents

- Building Nested Components
- Services and Dependency Injection
- Retrieving Data Using HTTP
- Navigation and Routing Basics
- Navigation and Routing Additional Techniques
- Angular Modules
- Building, Testing, and Deploying with the CLI
- Introduction
- Recapping Our Journey
- Learning More
- What Is Angular? (Revisited)
- Closing

Activities Google Chrome ▾ Home | Pluralsight ▾ Angular: Getting Started ▾ EmployeeManagement ▾ + Apr 27 22:55 •

https://app.pluralsight.com/player?course=angular-2-getting-started-update&author=deborah-kurata&name=angular-2-getting-started-update-m14&clip=1&mode=live

Angular: Getting Started

Final Words : Recapping Our Journey

When Should We Use Data Binding?

The diagram illustrates four types of data binding:

- Interpolation: {{pageTitle}}
- Property Binding:
- Event Binding: <button (click)='toggleImage()'>
- Two-Way Binding: <input [(ngModel)]='listFilter'>/

DOM

Component

Table of Contents Notes

- Building Nested Components (25m 49s)
- Services and Dependency Injection (18m 50s)
- Retrieving Data Using HTTP (27m 31s)
- Navigation and Routing Basics (22m 11s)
- Navigation and Routing Additional Techniques (21m 41s)
- Angular Modules (40m 27s)
- Building, Testing, and Deploying with the CLI (22m 25s)
- Final Words (6m 17s)
 - Introduction (0m 33s)
 - Recapping Our Journey (2m 2s)
 - Learning More (1m 55s)
 - What Is Angular? (Revisited) (1m 6s)
 - Closing (0m 38s)

1:03 / 2:02 ADD NOTE

Activities Google Chrome ▾ Home | Pluralsight ▾ Angular: Getting Started ▾ EmployeeManagement ▾ + Apr 27 22:55 •

https://app.pluralsight.com/player?course=angular-2-getting-started-update&author=deborah-kurata&name=angular-2-getting-started-update-m14&clip=1&mode=live

Angular: Getting Started

Why Do We Need a Service?

The diagram illustrates the composition of an application:

Application = Component + Component + Component + Services

Table of Contents Notes

- Building Nested Components (25m 49s)
- Services and Dependency Injection (18m 50s)
- Retrieving Data Using HTTP (27m 31s)
- Navigation and Routing Basics (22m 11s)
- Navigation and Routing Additional Techniques (21m 41s)
- Angular Modules (40m 27s)
- Building, Testing, and Deploying with the CLI (22m 25s)
- Final Words (6m 17s)
 - Introduction (0m 33s)
 - Recapping Our Journey (2m 2s)
 - Learning More (1m 55s)
 - What Is Angular? (Revisited) (1m 6s)
 - Closing (0m 38s)

More Courses:

Activities Google Chrome Home | Pluralsight Angular: Getting Started EmployeeManagement

Apr 27 22:58 • https://app.pluralsight.com/player?course=angular-2-getting-started-update&author=deborah-kurata&name=angular-2-getting-started-update-m14&clip=2&mode=live

Learning More

Pluralsight Courses

- "Angular: First Look"
- "Angular CLI"
- "Angular Reactive Forms"
- "Angular Routing"
- "Angular Component Communication"
- "Angular Fundamentals"
- "Unit Testing in Angular"

Angular Documentation

- [Angular.io](#)

Table of Contents

- Building Nested Components
- Services and Dependency Injection
- Retrieving Data Using HTTP
- Navigation and Routing Basics
- Navigation and Routing Additional Techniques
- Angular Modules
- Building, Testing, and Deploying with the CLI
- Final Words

Introduction 0m 35s
Recapping Our Journey 2m 2s
Learning More 1m 55s
What Is Angular? (Revisited) 1m 6s
Closing 0m 38s



Angular Definition-old one

Activities Google Chrome Home | Pluralsight Angular: Getting Started EmployeeManagement

Apr 27 22:59 • https://app.pluralsight.com/player?course=angular-2-getting-started-update&author=deborah-kurata&name=angular-2-getting-started-update-m14&clip=3&mode=live

Angular Is ...

A JavaScript framework
For building client-side applications
Using HTML, CSS and JavaScript

Table of Contents

- Building Nested Components
- Services and Dependency Injection
- Retrieving Data Using HTTP
- Navigation and Routing Basics
- Navigation and Routing Additional Techniques
- Angular Modules
- Building, Testing, and Deploying with the CLI
- Final Words

Introduction 0m 35s
Recapping Our Journey 2m 2s
Learning More 1m 55s
What Is Angular? (Revisited) 1m 6s
Closing 0m 38s



Activities Google Chrome Home | Pluralsight Angular: Getting Started EmployeeManagement

Apr 27 23:00 ● https://app.pluralsight.com/player?course=angular-2-getting-started-update&author=deborah-kurata&name=angular-2-getting-started-update-m14&clip=3&mode=live

Angular Is ...



A platform supporting

- Multiple languages (ES5, TypeScript)
- Multiple templating syntaxes (HTML, XML)
- Multiple rendering targets (DOM, native)

Mobile Web

- Material design
- Angular Universal
- Web Workers

Native Mobile Apps

Table of Contents Notes

- Building Nested Components
- Services and Dependency Injection
- Retrieving Data Using HTTP
- Navigation and Routing Basics
- Navigation and Routing Additional Techniques
- Angular Modules
- Building, Testing, and Deploying with the CLI
- Final Words
 - Introduction
 - Recapping Our Journey
 - Learning More
 - What Is Angular? (Revisited)
 - Closing

0m 58s 23m 49s 18m 50s 27m 31s 22m 17s 21m 41s 40m 27s 22m 25s 6m 17s 0m 35s 2m 2s 1m 55s 1m 6s 0m 38s