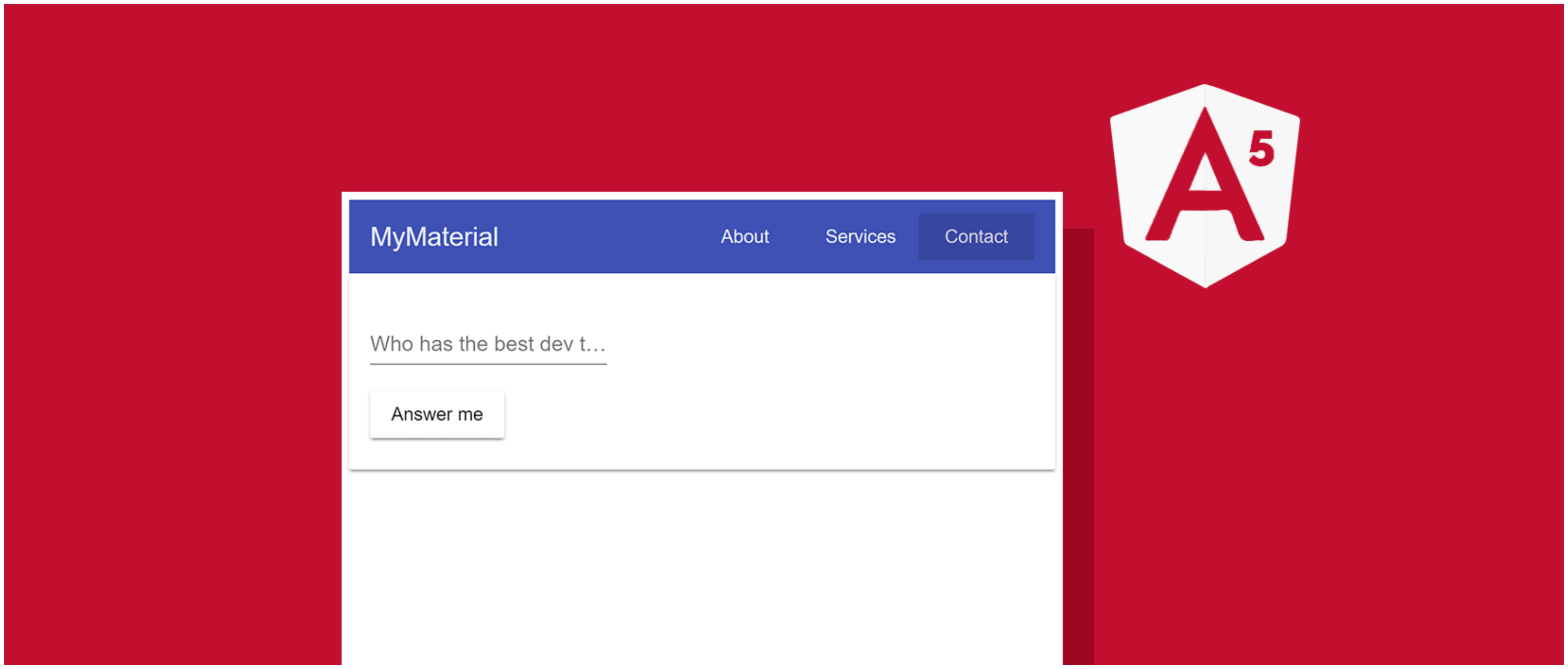
How to Build an Angular 5 Material App

BY GARY SIMON - NOV 09, 2017

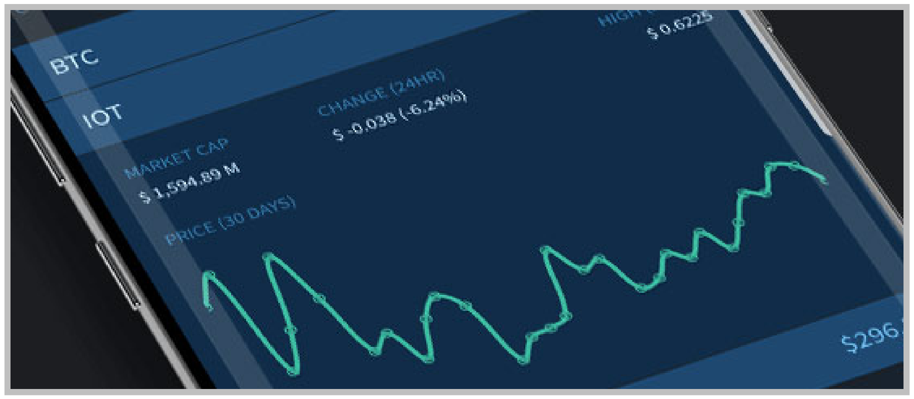


Angular Material has recently been updated, so our previous tutorial is outdated. Angular 5 has also just released, so it's worth revisiting this topic to get everyone up to date with Angular + Material!

As always, this tutorial is project-based and is meant to serve as a simple starting-point from which you can confidently go forward, and learn more about Angular Material on your own.

Let's get started.

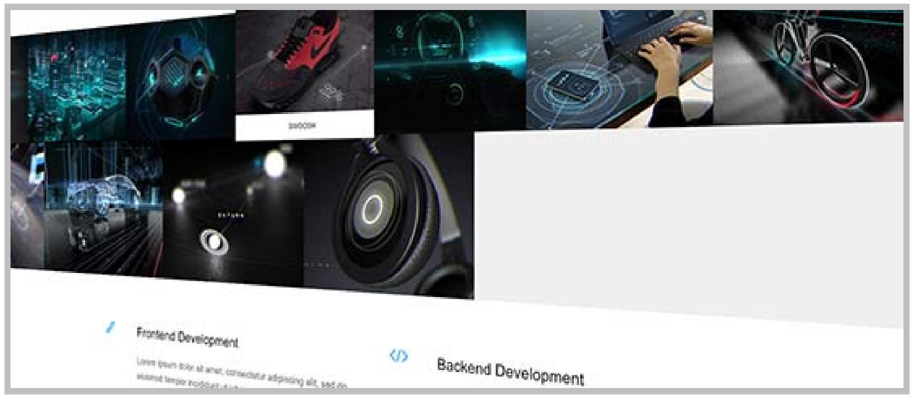
Learn Angular 5 from Scratch - Angular 5 Tutorial



Build a Beautiful CryptoCurrency App using Ionic 3

Create a MEAN App Called CodePost - Full Stack

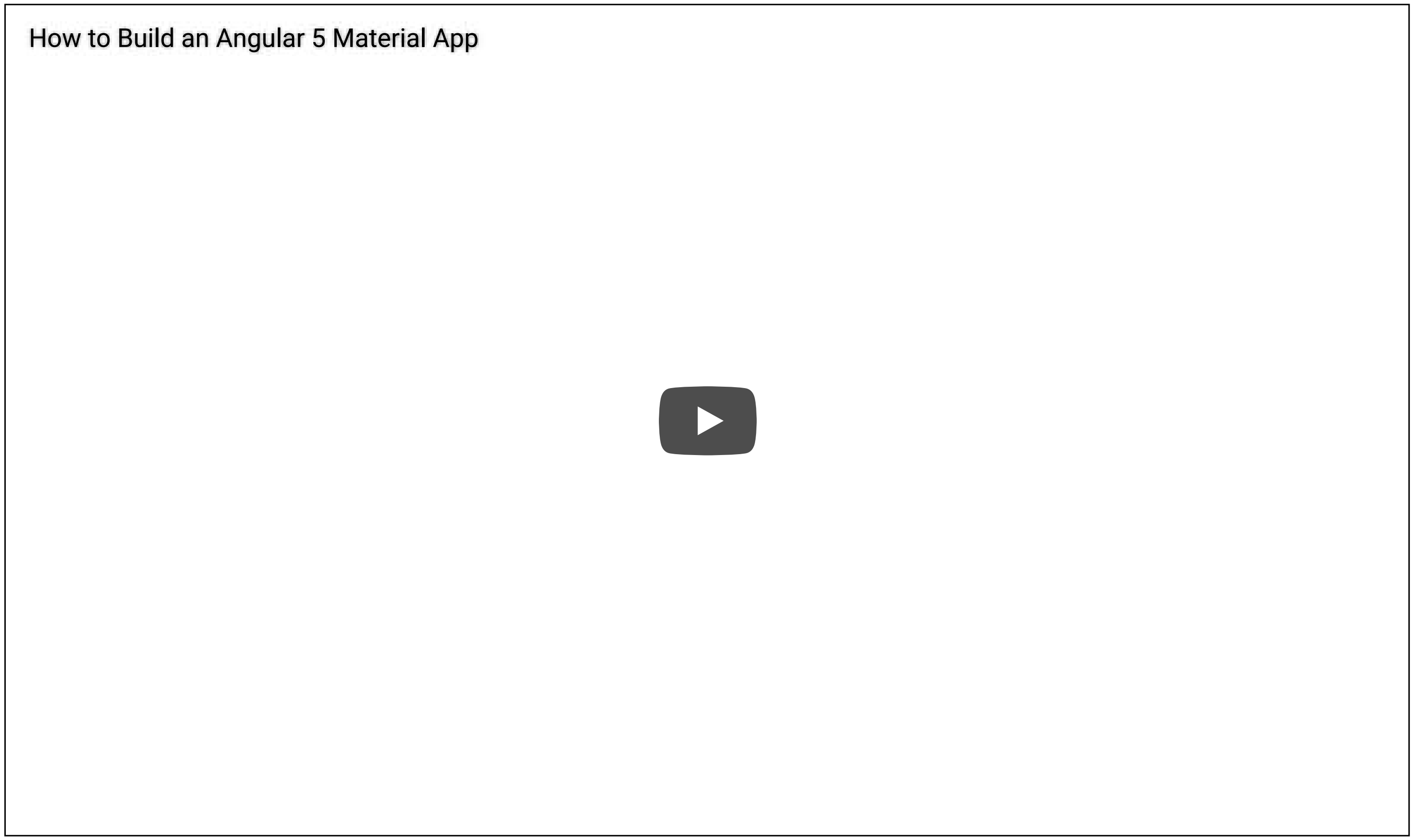
Learn Angular 4 from Scratch



Create a Personal Portfolio using Angular 2 & Behance

If you prefer watching a video instead..

Be sure to **Subscribe to the Official Coursetro Youtube Channel** for more videos.



Creating a New Angular 5 Project

We're going to use the Angular CLI to start this project, and I'm assuming you already have it installed. If you don't, please watch our **Angular 5 Tutorial** course series.

```
$ ng new ng5-material --routing
```

Once it's finished installing, hop into the directory and serve it with the CLI:

```
$ cd ng5-material
$ ng serve
```

Integrating Material

Open up a new console or command window and run the following command to use npm to install material:

```
$ npm install --save @angular/material @angular/cdk
```

Some of the material components use the Angular animation library, which is also installed separately via npm:

```
$ npm install --save @angular/animations
```

Let's integrate the freshly installed animations library to our */src/app/app.module.ts* file:

```
// Other imports removed for brevity
import { BrowserAnimationsModule } from '@angular/platform-browser/animations';

@NgModule({
  ...
  imports: [
    ...,
    BrowserAnimationsModule
  ],
  ...
})
export class AppModule { }
```

Next, for each material component we want to use in our project, we have to import it into the imports array of ngModule above.

If you only have a couple components, you could import them directly into this file. However, for purposes of organization and keeping your *AppModule* brief, you can create a custom module specifically for your material component imports. We'll do that.

Create a new file */src/app/material.module.ts* and paste the following contents:

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

@NgModule({
  imports: [MatButtonModule],
  exports: [MatButtonModule],
})
export class MaterialModule { }
```

Here, we're only including the *MatButtonModule*, but we'll add more to this later.

Next, back in *app.module.ts* we import it by adding:

```
// Other imports removed for brevity
import { MaterialModule } from './material.module';

@NgModule({
  ...
  imports: [
    ...,
    BrowserAnimationsModule,
    MaterialModule
  ],
  ...
})
export class AppModule { }
```

Whew, we're just about ready to go!

The next required step is to include an Angular Material theme. A theme provides a set of predefined colors that will be applied to your material design components.

The colors consist of:

- Primary colors
- Accent colors
- Warning colors
- Foreground colors
- Background colors

At the time of writing this tutorial, there are 4 predefined themes from which you can choose:

- deeppurple-amber.css
- indigo-pink.css
- pink-bluegrey.css
- purple-green.css

To integrate one, visit the */src/styles.css* file and paste the following line at the top:

```
@import "~@angular/material/prebuilt-themes/indigo-pink.css";
```

Next, we need to include **HammerJS* for gesture support, as some components require this for full-feature support.

In the console, type:

```
$ npm install --save hammerjs
```

To include this, we add it to the */src/main.ts* entry point as an import:

```
// Other imports removed for brevity

import 'hammerjs';
```

Finally, if you're going to use any Material Design Icons (they are used quite frequently throughout material), we need to import Material Icons in the */src/index.html* file between the head tags:

```
<link href="https://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">
```

We're ready to go!

Testing it out

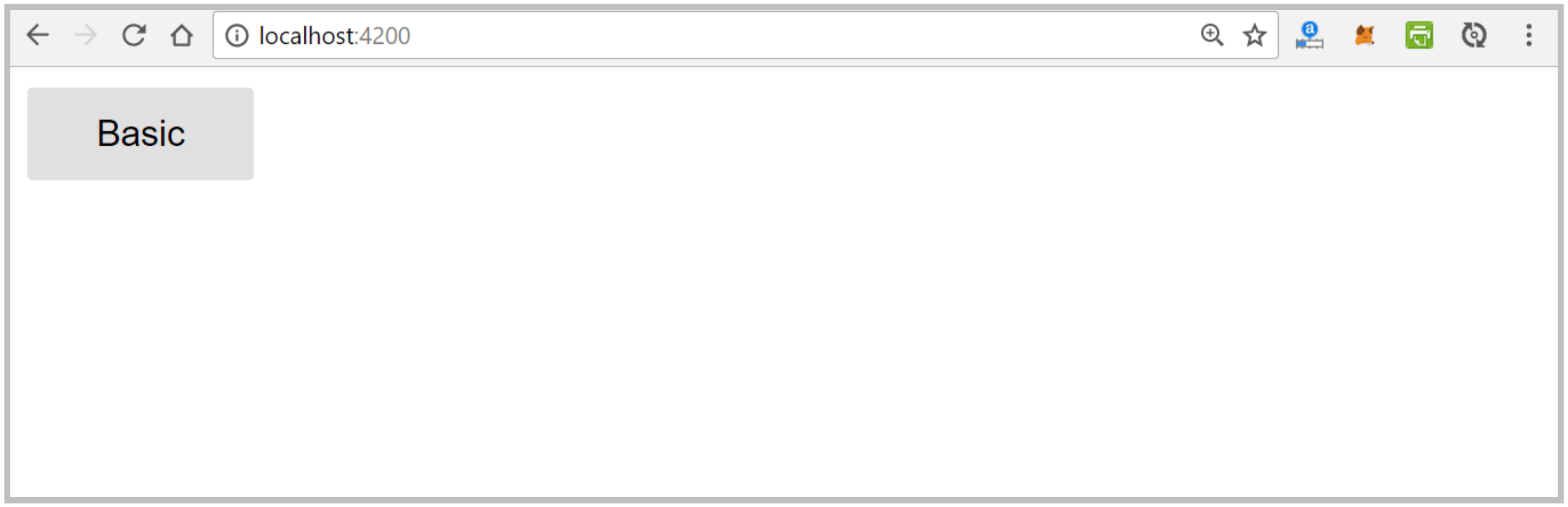
That was a lot of setup! Let's make sure it works by checking out the Angular Material Button component.

Visit the */src/app/app.component.html* and remove all of the HTML and replace it with:

```
<button mat-button>My Button</button>

<router-outlet></router-outlet>
```

Save it, and visit *http://localhost:4200* in the browser. Your result should look like this:



Hover over the result and it will present you with this gray background, and if you click on it, you will see an animation.

This means that Angular Material has been installed correctly and you're ready to use all of the components!

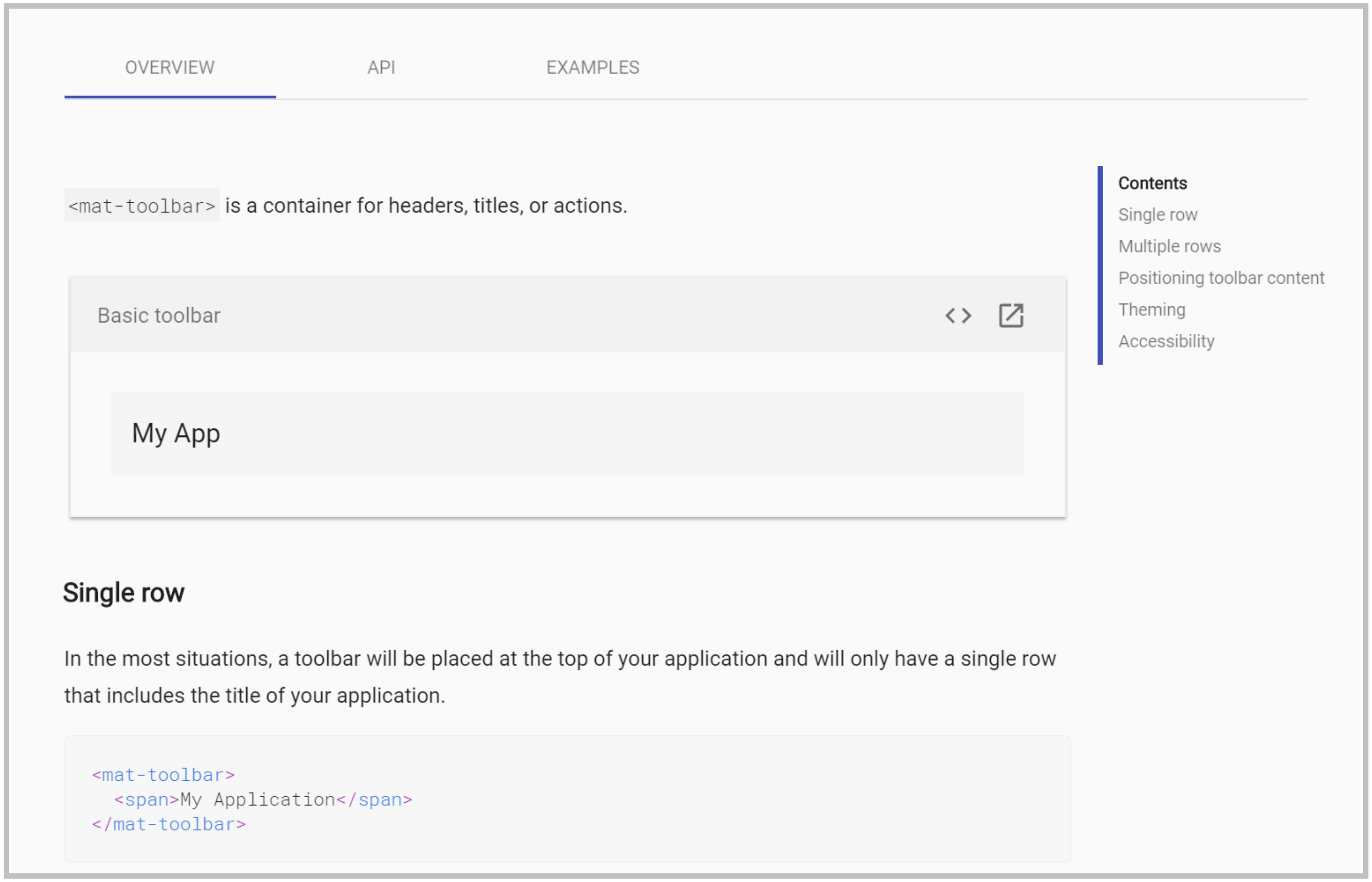
Integrating and Using Angular 5 Material Components

The process of using Material components in your Angular app is very straight forward. While we're not going to cover every component here, but I am going to show you how you can use the official documentation to use any component you wish.

Take a look at the Angular Material Documentation for components. Let's say we want to start off by integrating a toolbar.

Click on Navigation > Toolbar.

The layout for each component is structured similarly:



`<mat-toolbar>` is a container for headers, titles, or actions.

Single row

In the most situations, a toolbar will be placed at the top of your application and will only have a single row that includes the title of your application.

```
<mat-toolbar>
  <span>My Application</span>
</mat-toolbar>
```

At the top, we have a tabbed navigation:

- OVERVIEW
  This provides you with a basic overview and options associated with the given component.

- API
  This provides you with a line of how to import the component. In our case, this is within */src/app/material.module.ts*. It also provides you with the directives that you use.

- **EXAMPLES**
  And this provides you with examples of how to use the given component. You can click on the coding < > icon to see the HTML structure associated with the component examples.

So, if we want to integrate the toolbar component, we first visit the API tab. We can see it's called *MatToolbarModule*.

We add this to the import line and the ngModule within */src/app/material.module.ts* file:

```
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { MatButtonModule, MatToolbarModule } from '@angular/material';

@NgModule({
  imports: [MatButtonModule, MatToolbarModule],
  exports: [MatButtonModule, MatToolbarModule],
})
export class MaterialModule { }
```

Next, visit the */src/app/app.component.html* template and add:

```
<mat-toolbar color="primary">
  <mat-toolbar-row>
    <span>MyMaterial</span>

    <span class="example-spacer"></span>

    <button mat-button>About</button>
    <button mat-button>Services</button>
    <button mat-button>Contact</button>
  </mat-toolbar-row>
</mat-toolbar>
```

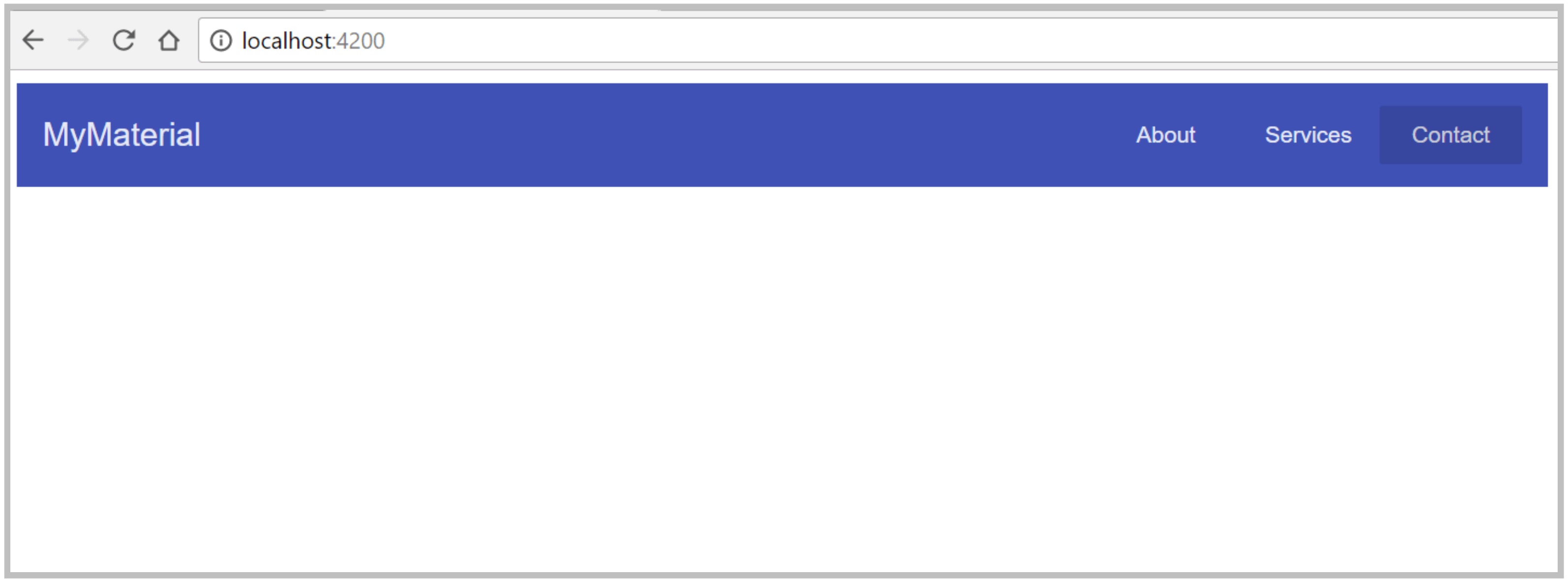
In the documentation examples, if you click the < > icon, you will notice it has tabs for HTML, TS, and CSS.

The CSS tab has a couple rulesets needed for the above example to fully work. Visit the */src/app/app.component.css* file and paste in the contents:

```
.example-icon {
  padding: 0 14px;
}

.example-spacer {
  flex: 1 1 auto;
}
```

Save the project and view the result in the browser:



And that's the process of integrating and using components in Angular 5. Let's give ourselves some muscle memory by adding more to our sample project.

Bringing it All Together

Let's use a few different components together. We're going to create a very simple form and a submit button contained inside of a material card. When an answer is given and the button clicked, we will show the material spinner component for a couple seconds, and then show the result in the template.

Once again, looking at the Cards Documentation in the API tab for form inputs, spinners and cards, the following libraries must be imported into our *material.module.ts* file:

```
...
import { ..., MatInputModule, MatProgressSpinnerModule, MatCardModule } from '@angular/material';

@NgModule({
  imports: [..., MatInputModule, MatProgressSpinnerModule, MatCardModule],
  exports: [..., MatInputModule, MatProgressSpinnerModule, MatCardModule],
})
export class MaterialModule { }
```

Then, in our *app.component.html* we can add:

```
<mat-card>
  <mat-form-field>
    <input matInput [(ngModel)]="answer" placeholder="Who has the best dev tutorials?">
  </mat-form-field>

  <br>

  <button mat-raised-button (click)="showAnswer()">Answer me</button>
  <mat-spinner [style.display]="showSpinner ? 'block' : 'none'"></mat-spinner>
</mat-card>


<mat-card *ngIf="answerDisplay">
  <h1>{{ answerDisplay }}, you're damn right.</h1>
</mat-card>
```

First, we're using the *mat-card* component, and then an input with 2-way data binding through *ngModel*. Then, we have a material button with a click event bound to a method we need to define called *showAnswer()*.

We have *mat-spininer* which is using style binding to set the *display* property to *block* if the property *showSpinner* is true, and to *none* if it's not.

Beneath that, we have another *mat-card* only presenting if *answerDisplay* is set. You'll see how this all works once we get to the component logic.

Before we do that, we have to import the *FormsModule* in order to have access to *ngModel*.

Open up */src/app/app.module.ts* and import:

```
...
import { FormsModule } from '@angular/forms';

@NgModule({
  ...
  imports: [
    ...,
    FormsModule
  ],
  ...
})
export class AppModule { }
```

Then, visit */src/app/app.component.ts* and specify the following code in the class:

```
export class AppComponent {

  answer: string = '';
  answerDisplay: string = '';
  showSpinner: boolean = false;

  showAnswer() {
    this.showSpinner = true;

    setTimeout(() => {
      this.answerDisplay = this.answer;
      this.showSpinner = false;
    }, 2000);
  }

}
```

Simple enough! We're using *setTimeout()* to simulate an API call, just so that we can see the actual material spinner for awhile.

Conclusion

As you can see, working with Angular 5 Material is surprisingly easy. It's just a matter of understanding the documentation formatting, and going from there.

Enjoy!

Your Free Trial

Build, Design And Mana
Using A Single Powerfu
Platform.

mulesoft.com

Share this post

Say something about this awesome post!

30 Comments **Coursetro** 1 **Login**

Recommend 10 Share Sort by Best

Join the discussion…

LOG IN WITH

OR SIGN UP WITH DISQUS ?

Name

Muhammad Rehan Qadri • 7 months ago
Awesome tutorial! You don't need to worry about your competitors dude, you're awesome. The quality, design and style of your work is simply amazing, and the best part is, it's Free! (accessible to everyone).
5 • Reply • Share ›

Junior Osho • 5 months ago
github project ?
3 • Reply • Share ›

Oleh Podolyan • 6 months ago

Thank you very much, your tutorial saved a lot of time, everything went well, except:
npm WARN @angular/cdk@5.0.4 requires a peer of @angular/core@~5.1.1 but none is installed. You must install peer dependencies yourself.
npm WARN @angular/cdk@5.0.4 requires a peer of @angular/common@~5.1.1 but none is installed. You must install peer dependencies yourself.
npm WARN @angular/material@5.0.4 requires a peer of @angular/core@~5.1.1 but none is installed. You must install peer dependencies yourself.
npm WARN @angular/material@5.0.4 requires a peer of @angular/common@~5.1.1 but none is installed. You must install peer dependencies yourself.
Is that crucial?
Thanks, anyway

3 • Reply • Share ›

Junior Osho • 6 months ago

god..

1 • Reply • Share ›

Digamber Singh • 19 days ago

Great job Gary, Excellent stuff!

• Reply • Share ›

David Baynes • 22 days ago

Before taking this I took the Angular 6. Both outstanding. These are simply the best taught programming courses going. Clean, straight forward, no b.s. Just like sitting down with a friend and having them show you and teach you how to use these programming tools and techniques.

• Reply • Share ›

Rodrigo Pérez • a month ago

Gary, would you please publish a course of material design for angular 6?

• Reply • Share ›

Trịnh Bá Chủ • 2 months ago

that's awesome bro!

• Reply • Share ›

Aydin Mz • 3 months ago

Thanks man ,your tutorial is fantastic ,easy to understand ,complete with written version provided and with a professional microphone ,good job dude ;)

• Reply • Share ›

Nguyen Thinh • 3 months ago

fuc*, it's so greart, feel so good

• Reply • Share ›

Meghanada Ravana • 4 months ago

Thank You so much. I started learning angular material from this tutorial.

^ | v • Reply • Share ›

Girish • 5 months ago

Great job very helpful thanks

^ | v • Reply • Share ›

Jackel Shiu • 6 months ago

I got this error.
What should I do?
Many thanks.

see more

^ | v • Reply • Share ›

> **Moshiour Rahman** → Jackel Shiu • 3 months ago
>
> There is error in your stylesheets somewhere try to correct them
>
> ^ | v • Reply • Share ›

Juan Montemayor • 6 months ago

Thank you very much, this web is so helpful

^ | v • Reply • Share ›

Nicolas Villalba • 7 months ago

Hi! nice intro for AM. Any chance to bring the whole module at once? as i seen in ng4

^ | v • Reply • Share ›

Glaived • 7 months ago

If you have the error

```
core.js:1350 ERROR Error: Uncaught (in promise): EmptyError: no elements in sequence
```

, you can downgrade your rxjs version to 5.5.2 with

```
npm install rxjs@5.5.2 --save
```

Solution found at : https://stackoverflow.com/a...

Official issue from Angular repository: https://github.com/angular/...

^ | ˅ • Reply • Share ›

Nader Anis Mitry • 7 months ago

Thanks

^ | ˅ • Reply • Share ›

Jay Rich • 7 months ago

goos stuff, Thanks!

^ | ˅ • Reply • Share ›

Alex Groisman • 8 months ago

cool Exactly what I need to jumpstart with the Material in Angular 5. Thanks!

^ | ˅ • Reply • Share ›

Yogesh Mane • 8 months ago

Excellent tutorial . Thank You very much......

^ | ˅ • Reply • Share ›

Stanley Bateswar • 8 months ago

Really nice man. Very well explained.

^ | ˅ • Reply • Share ›

Ojini • 8 months ago

nice post from you here. could really love to see some post about imports and exports, which to import or export and when to do such in any project. thanks

^ | ˅ • Reply • Share ›

Bandaiah Vamsharaj • 8 months ago

Good tutorial with clear explanation. Thank you

^ | ˅ • Reply • Share ›

Ashkan Dorkian • 8 months ago

Excellent nice & neat. Thanks.

Reply • Share ›

David Baker • 8 months ago

With this... and so many of the other Angular apps... the huge confusion comes in with all the imports/exports. It would be super helpful to have a tutorial on what, when and why this very common pattern occurs and how to do it correctly.

Reply • Share ›

Ben Racicot → David Baker • 8 months ago

Yes! Exactly why I arrived here. When you have reusable components without a module how do you import the correct material component into them? I have no idea and there doesn't seem to be discussion on this online. Creating a master-module for material components to import and export is useless in a real world app where you 're optimizing everything... Would love a detailed explanation of this.

Reply • Share ›

Gaillard Stephane → Ben Racicot • 6 months ago

just get a ts file in the main component that you call module.material.ts where you put all your material import; then your free to use them in all the apps

Reply • Share ›

David Baker • 8 months ago

Metallica shirt = very good :)

Reply • Share ›

Denis Sanchez Leyva • 8 months ago

Very good, thanks..

Reply • Share ›

ALSO ON **COURSETRO**

5. Vue Forms

1 comment • 5 months ago

Avatar **Paweł Grajewski** — Gary rulez !

Figma Tutorial - An Introduction to a Very Impressive UI Design & Prototyping Tool

3 comments • 3 months ago

Santosh Nirankari — just went through it completely, you should share it

HOME

COURSES

CHALLENGES

LOGIN

JOIN

TUTORIALS

CONTACT US

CHAT WITH US