

1 Introducing GitHub Enterprise - How Businesses Build Softwar

Code discovery & reuse, multiple integrations, flexible hosting options. enterprise.github.com

2 E-commerce built on Blockchain

Inxeption: Connecting B2B companies directly to end users [Inxeption](#)

Angular 7 Tutorial - Learn Angular 7 by Example

BY GARY SIMON - OCT 20, 2018



Ever since the release of Angular 2, I have created a full course for each new iteration. Today is no different, as Angular 7 just released!

With this beginner's crash course, I make the assumption that you have never worked with Angular before. Therefore, this tutorial is perfectly suited towards a beginner with no prior Angular experience. The only thing you should be familiar with is HTML, CSS and JavaScript.

In this course, you're going to discover just how powerful Angular 7 is when it comes to creating frontend web apps. Let's get started!

1 Hire Expert UI Design Company - Increase Net Promoter Score

9+ years of experience in designing custom UI/UX solutions for many industries.
rocksaucestudios.com

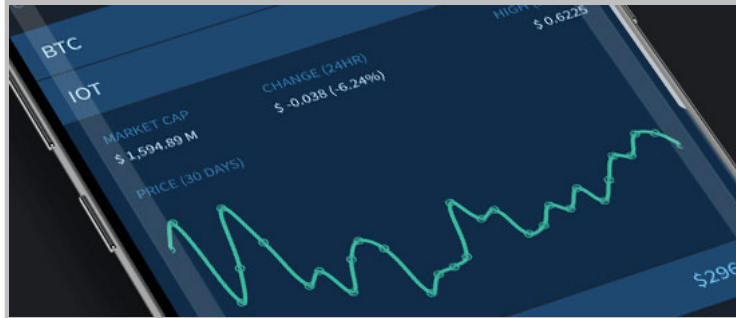
2 Introducing GitHub Enterprise - GitHub Enterprise & Connect

Enterprise-grade security, centralized permissions, simple compliance. enterprise.github.com





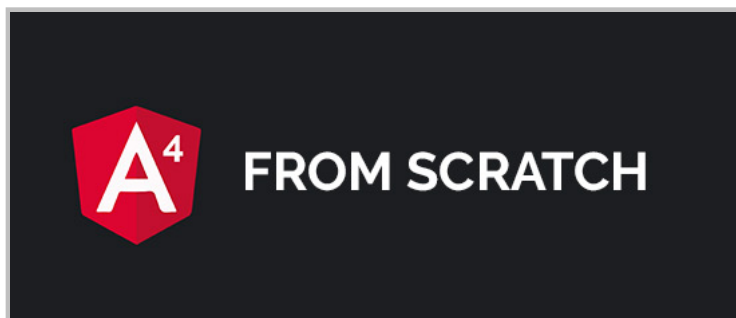
Learn Angular 5 from Scratch - Angular 5 Tutorial



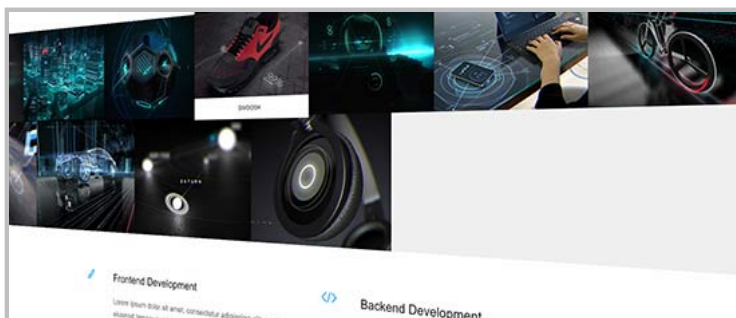
Build a Beautiful Cryptocurrency App using Ionic 3



Create a MEAN App Called CodePost - Full Stack



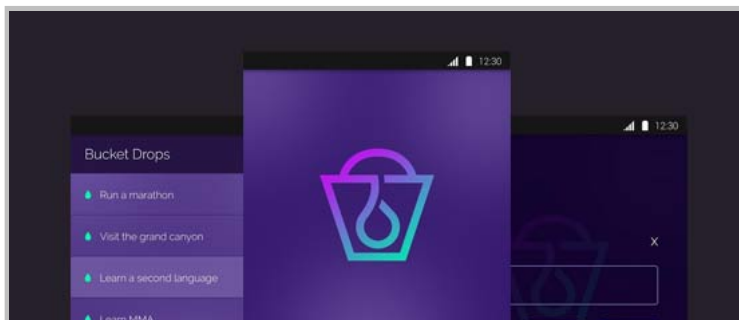
Learn Angular 4 from Scratch



Create a Personal Portfolio using Angular 2 & Behance



Getting Started with Java



Design + Code an Android App from Scratch

If you prefer watching a video..

Be sure to [Subscribe to the Official Coursetro Youtube Channel](#) for more videos.

Learn Angular 7 in 50 Minutes - A Free Beginner's Crash Course



Installation

You're first going to need to install the Angular CLI (Command Line Interface) tool, which helps you start new Angular 7 projects as well as assist you during development. In order to install the Angular CLI, you will need [Nodejs](#). Make sure you install this with the default options and reload your command line or console after doing so.



Spend less time discussing bugs and changes. BugHerd collects feedback and tracks bugs, like using sticky notes on a website.

Try BugHerd Free

x

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

Once complete, you can now access the CLI by simply starting any commands with **ng**.

Hop into whichever folder you want to store your projects, and run the following command to install a new Angular 7 project:

```
> ng new ng7-pre
```

It's going to present you with a couple questions before beginning:

```
? Would you like to add Angular routing? Yes
? Which stylesheet format would you like to use? SCSS [ http://sass-lang.com ]
```

It will take a minute or two and once completed, you can now hop into the new project folder by typing:

```
> cd ng7
```

Open up this project in your preferred code editor (I use Visual Studio Code, and you can launch it automatically by typing `code .` in the current folder), and then run this command to run a development server using the Angular CLI:

```
> ng serve -o
```

-o is for *open*, this flag will open your default browser at <http://localhost:4200>. **Tip:** You can type **ng** to get a list of all available commands, and **ng [command] --help** to discover all their flags.

Awesome! If all went smooth, you should be presented with the standard landing page template for your new Angular 7 project:



Here are some links to help you start:

- [Tour of Heroes](#)
- [CLI Documentation](#)
- [Angular blog](#)

Angular 7 Components

The most basic building block of your Angular 7 application (and this is a concept that's not new) is the component. A component consists of three primary elements:

- The HTML template
- The logic
- The styling (CSS, Sass, Stylus, etc..)

When we use the Angular CLI to start a new project, it generates a single component, which is found in `/src/app/`:

```
/app.component.html  
/app.component.scss  
/app.component.ts
```

While we have three files here that represent the three elements above, the `.ts` (TypeScript) is the heart of the component. Let's take a look at that file:

```
import { Component } from '@angular/core';  
  
@Component({  
  selector: 'app-root',  
  templateUrl: './app.component.html',  
  styleUrls: ['./app.component.scss']  
})  
export class AppComponent {  
  title = 'ng7-pre';  
}
```

Here, because this is a component, we're importing `Component` from the `@angular/core` library, and then it defines what's called a *@Component decorator*, which provides configuration options for that particular component.

As you can see, it's referencing the location of the HTML template file and the CSS file with the `templateUrl` property and the `styleUrls` property.



```
> ng generate component nav
// output

> ng g c about
// output

> ng g c contact
// output

> ng g c home
// output
```

Notice we first use the full syntax to generate a component, and then we use a shorthand syntax, which makes life a little bit easier. The commands do the same thing: generate components.

When we generate components this way, it will create a new folder in the `/src/` folder based on the name we gave it, along with the respective template, CSS, .ts and .spec (for testing) files.

Angular 7 Templating

You may have noticed that one of the components we generated was called `nav`. Let's implement a header bar with a navigation in our app!

The first step is to visit the [app.component.html](#) file and specify the following contents:

```
<app-nav></app-nav>

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

So, we've removed a bunch of templating and placed in `<app-nav></app-nav>`, what does this do and where does it come from?

Well, if you visit `/src/app/nav/nav.component.ts` you will see that in the component decorator, there's a `selector` property bound to the value of `app-nav`. When you reference the selector of a given component in the form of a custom HTML element, it will nest that component inside of the component it's that's referencing it.

If you save the file you just updated, you will see in the browser we have a simple, *nav works!* And that's because the `nav.component.html` file consists of a simple paragraph stating as much.

At this point, let's specify the following HTML to create a simple navigation:

```
<header>
  <div class="container">
    <a routerLink="/" class="logo">apptitle</a>
    <nav>
      <ul>
        <li><a routerLink="/">Home</a></li>
        <li><a routerLink="/about">About</a></li>
        <li><a routerLink="/contact">Contact us</a></li>
      </ul>
    </nav>
  </div>
</header>
```

The only thing that might look a little strange is `routerLink`. This is an Angular 7 specific attribute that allows you to direct the browser to different routed components. The standard `href` element will not work.

```
<!-- From: -->
<a routerLink="/">myapp</a>

<!-- To: -->
<a routerLink="/">{{ appTitle }}</a>
```

Interpolation is executed by wrapping the name of a property that's defined in the component between {{ }}.

Let's define that property in [nav.component.ts](#):

```
export class NavComponent implements OnInit {

  appTitle: string = 'myapp';
  // OR (either will work)
  appTitle = 'myapp';

  constructor() { }

  ngOnInit() {
  }

}
```

You can use the TypeScript way of defining properties or standard JavaScript. Save the file and you will see **myapp** is back in the template.

There's a lot more to templating, but we will touch on those topics as we continue. For now, let's apply style to our header.

First, let's visit the global stylesheet by opening [/src/styles.scss](#) and define the following rulesets:

```
@import url('https://fonts.googleapis.com/css?family=Montserrat:400,700');

body, html {
  height: 100%;
  margin: 0 auto;
}

body {
  font-family: 'Montserrat';
  font-size: 18px;
}

a {
  text-decoration: none;
}

.container {
  width: 80%;
  margin: 0 auto;
  padding: 1.3em;
  display: grid;
  grid-template-columns: 30% auto;

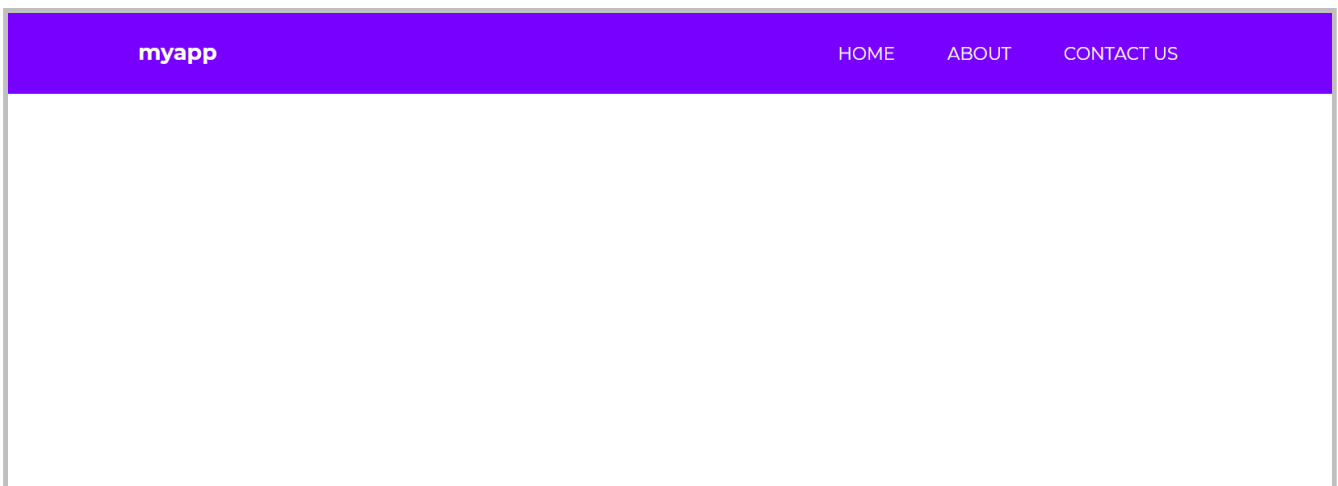
  a {
    color: white;
  }
}

section {
  width: 80%;
  margin: 0 auto;
```

Visit [nav/component.scss](#) and paste the following contents:

```
header {  
  background: #7700FF;  
  
  .logo {  
    font-weight: bold;  
  }  
  
  nav {  
    justify-self: right;  
  
    ul {  
      list-style-type: none;  
      margin: 0; padding: 0;  
  
      li {  
        float: left;  
  
        a {  
          padding: 1.5em;  
          text-transform: uppercase;  
          font-size: .8em;  
  
          &:hover {  
            background: #8E2BFF;  
          }  
        }  
      }  
    }  
  }  
}
```

If you save and refresh, this should be the result in the browser:



Awesome!

Angular 7 Routing

Now that we have a navigation, let's make our little app actually navigation between our components as needed.

Open up [/src/app/app-routing.module.ts](#) and specify the following contents:


```
import { HomeComponent } from './home/home.component';
import { AboutComponent } from './about/about.component';
import { ContactComponent } from './contact/contact.component';

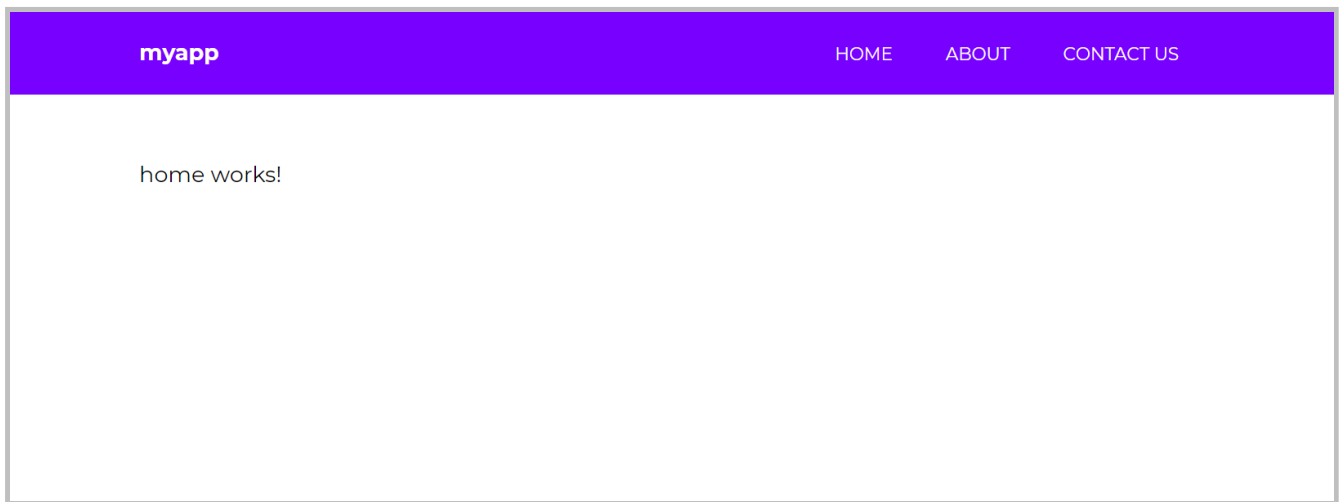
const routes: Routes = [
  { path: '', component: HomeComponent },
  { path: 'about', component: AboutComponent },
  { path: 'contact', component: ContactComponent },
];

// Other code removed for brevity
```

As we can see here, we're defining importing our components and defining an object for each route inside of the `routes` constant. These route objects also accept other properties, which allow you to define URL parameters, but because our app is simple, we won't be doing any of that.

Save this file and try clicking on the links above. You will see that each of the respective component's HTML templating shows up in the `<router-outlet></router-outlet>` defined in `app.component.html`.

This is what the result should look like in the browser at this point:



You now know enough about Angular 7 to create a very simple website with routing! But let's learn more than that.

Angular 7 Event Binding

In the next several sections, we're going to use our `/src/app/home` component as a playground of sorts to learn features specific to Angular 7.

One of the most used forms of event binding is the click event. You often need to make your app respond when a user clicks something, so let's do that!

Visit the `/src/app/home/home.component.html` template file and specify the following:

```
<h1>Home</h1>

<button (click)="firstClick()">Click me</button>
```

You define an event binding by wrapping the event between `()`, and calling a method. You define the method in the `home.component.ts` file as such:

```
export class HomeComponent implements OnInit {

  constructor() { }

  ngOnInit() {
  }
}
```

```
}
```

Save it, get out the browser console (CTRL+SHIFT+i) and click on the button. The output should show *clicked*. Great!

You can experiment with the other event types by replacing (*click*) with the names below:

```
(focus)="myMethod()"
(blur)="myMethod()"
(submit)="myMethod()"
(scroll)="myMethod()"

(cut)="myMethod()"
(copy)="myMethod()"
(paste)="myMethod()"

(keydown)="myMethod()"
(keypress)="myMethod()"
(keyup)="myMethod()"

(mouseenter)="myMethod()"
(mousedown)="myMethod()"
(mouseup)="myMethod()"

(click)="myMethod()"
(dblick)="myMethod()"

(drag)="myMethod()"
(dragover)="myMethod()"
(drop)="myMethod()"
```

Angular 7 Class & Style Binding

Sometimes you may need to change the appearance of your UI from your component logic. There are two ways to do this, through class and style binding.

There are a lot of different methods you can use to control class binding, so we won't cover them all. But I will cover some of the most common use cases.

Let's say that you want to control whether or not a CSS class is applied to a given element. Update the **h1** element in *home.component.html* to the following:

```
<h1 [class.gray]="h1Style">Home</h1>
```

Here, we're saying that the CSS class of *.gray* should only be attached to the **h1** element if the property *h1Style* results to true. Let's define that in the *home.component.ts* file:

```
h1Style: boolean = false;

constructor() { }

ngOnInit() {
}

firstClick() {
  this.h1Style = true;
}
```



Spend less time discussing bugs and changes. BugHerd collects feedback and tracks bugs, like using sticky notes on a website.

Try BugHerd Free

x

```
.gray {  
  color: gray;  
}
```

Save it, and you can now click on the *click me* button to change the color of the Home title.

What if you wanted to control multiple classes on a given element? You can use *ngClass*. Modify the home component's template file to the following:

```
<h1 [ngClass]="{  
  'gray': h1Style,  
  'large': !h1Style  

```

Then, add the *large* ruleset to the *.scss* file:

```
.large {  
  font-size: 4em;  
}
```

Now give it a shot in the browser. *Home* will appear large, but shrink down to the regular size when you click the button. Great!

You can also control appearance by changing the styles directly from within the template. Modify the template as such:

```
<h1 [style.color]="h1Style ? 'gray' : 'black'">Home</h1>
```

Refresh and give this a shot by clicking the button.

Like *ngClass()* there's also an *ngStyle()* that works the same way:

```
<h1 [ngStyle]="{  
  'color': h1Style ? 'gray' : 'black',  
  'font-size': !h1Style ? '1em' : '4em'  

```

Give this a go! Awesome!

Angular 7 Services

Services in Angular 7 allow you to define code that's accessible and reusable throughout multiple components. A common use case for services is when you need to communicate with a backend of some sort to send and receive data.

```
> ng generate service data
```

Open up the new service file `/src/app/data.service.ts` and let's create the following method:



```
export class DataService {  
  
  constructor() { }  
  
  firstClick() {  
    return console.log('clicked');  
  }  
}
```

To use this in a component, visit [/src/app/home/home.component.ts](#) and update the code to the following:

```
import { Component, OnInit } from '@angular/core';  
import { DataService } from '../data.service';  
  
@Component({  
  selector: 'app-home',  
  templateUrl: './home.component.html',  
  styleUrls: ['./home.component.scss']  
})  
export class HomeComponent implements OnInit {  
  
  constructor(private data: DataService) { }  
  
  ngOnInit() {  
  }  
  
  firstClick() {  
    this.data.firstClick();  
  }  
}
```

There are 3 things happening here:

- We're first importing the *DataService* at the top.
- We're creating an instance of it through dependency injection within the *constructor()* function.
- Then we call the method with *this.data.firstClick()* when the user clicks on the button.

If you try this, you will see that it works as *clicked* will be printed to the console. Awesome! This means that you now know how to create methods that are accessible from any component in your Angular 7 app.

Angular 7 HTTP Client

Angular comes with its own HTTP library that we will use to communicate with a fake API to grab some data and display it on our home template. This will take place within the *data.service* file that we generated with the CLI.

In order to gain access to the HTTP client library, we have to visit the [/src/app/app.module.ts](#) file and make a couple changes. Up until this point, we haven't touched this file, but the CLI has been modifying it based on the generate commands we've issued to it.

Add the following to the imports section at the top:

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

Next, add it to the *imports* array:

```
AppRoutingModule,
HttpClientModule,    // <-- Right here
],
```

Now we can use it in our `/src/app/data.service.ts` file:

```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http'; // Import it up here

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

  constructor(private http: HttpClient) { }

  getUsers() {
    return this.http.get('https://reqres.in/api/users')
  }
}
```

reqres.in is a free public API that we can use to grab data.

Open up our `home.component.ts` file and modify the following:

```
export class HomeComponent implements OnInit {

  users: Object;

  constructor(private data: DataService) { }

  ngOnInit() {
    this.data.getUsers().subscribe(data => {
      this.users = data
      console.log(this.users);
    });
  }
}
```

The first thing you might notice is that we're placing the code inside of the `ngOnInit()` function, which is a lifecycle hook for Angular. Any code placed in here will run when the component is loaded.

We're defining a `users` property, and then we're calling the `.getUsers()` method and subscribing to it. Once the data is received, we're binding it to our `users` object and also `console.log` it.

Give it a try in the browser and you will see the console shows an object that's returned. Let's display it on our home template!

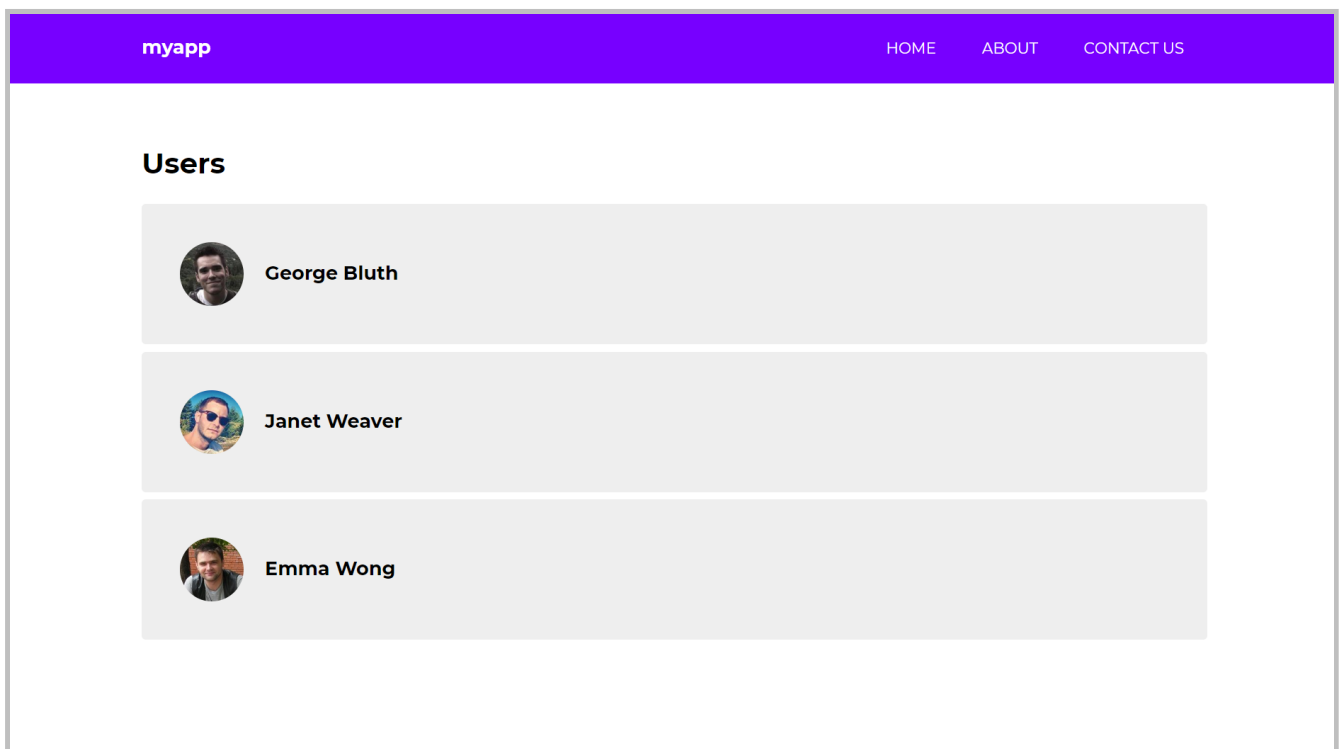
Open up `home.component.html` and specify the following:

```
<h1>Users</h1>

<ul *ngIf="users">
  <li *ngFor="let user of users.data">
    <img [src]="user.avatar">
    <p>{{ user.first_name }} {{ user.last_name }}</p>
  </li>
</ul>
```

```
ul {  
  list-style-type: none;  
  margin: 0; padding: 0;  
  
  li {  
    background: rgb(238, 238, 238);  
    padding: 2em;  
    border-radius: 4px;  
    margin-bottom: 7px;  
    display: grid;  
    grid-template-columns: 60px auto;  
  
    p {  
      font-weight: bold;  
      margin-left: 20px;  
    }  
  
    img {  
      border-radius: 50%;  
      width: 100%;  
    }  
  }  
}
```

View the result in the browser:



Awesome!

Angular 7 Forms

If you recall, we generated a component called `contact`. Let's create a contact form so that you can learn how to use forms in Angular 7.

Angular 7 provides you with two different approaches to dealing with forms: template driven and reactive forms. I'm not going to go into the differences between these two approaches, but reactive forms generally provide you with more control and form validation can be unit tested as opposed to template driven forms.

To get started, we have to visit the `app.module.ts` file and import the Reactive Forms Module:

```
// other code
imports: [
  BrowserModule,
  AppRoutingModule,
  HttpClientModule,
  ReactiveFormsModule // <- Add here
],
```

Next, visit the [contact.component.ts](#) file and specify the following

```
import { Component, OnInit } from '@angular/core';
import { FormBuilder, FormGroup, Validators } from '@angular/forms';

@Component({
  selector: 'app-contact',
  templateUrl: './contact.component.html',
  styleUrls: ['./contact.component.scss']
})
export class ContactComponent implements OnInit {

  messageForm: FormGroup;
  submitted = false;
  success = false;

  constructor(private formBuilder: FormBuilder) { }

  ngOnInit() {
    this.messageForm = this.formBuilder.group({
      name: ['', Validators.required],
      message: ['', Validators.required]
    });
  }

  onSubmit() {
    this.submitted = true;

    if (this.messageForm.invalid) {
      return;
    }

    this.success = true;
  }
}
```

First, we're importing *FormBuilder*, *FormGroup*, *Validators* from `@angular/forms`.

Then we're setting a few boolean properties that will help us determine when the form has been submitted and if it validation is successful.

Then we're creating an instance of the *formBuilder* in the constructor. We then use this form building to construct our form properties in the *ngOnInit()* lifecycle hook.

We have two properties, *name* and *message*.

Then we created an *onSubmit()* method that will be called when the user submits the form. This is typically where you would call upon a method in the service to communicate with a mail service of sorts.

Next, visit *contact.component.html*:

```
<h1>Contact us</h1>

<form [formGroup]="messageForm" (ngSubmit)="onSubmit()">

  <h5 *ngIf="success">Your form is valid!</h5>
```



Spend less time discussing bugs and changes. BugHerd collects feedback and tracks bugs, like using sticky notes on a website.

Try BugHerd Free

x

```
<input type="text" formControlName="name" />
<div *ngIf="submitted && messageForm.controls.name.errors" class="error">
  <div *ngIf="messageForm.controls.name.errors.required">Your name is required</div>
</div>
</label>

<label>
  Message:
  <textarea formControlName="message"></textarea>
  <div *ngIf="submitted && messageForm.controls.message.errors" class="error">
    <div *ngIf="messageForm.controls.message.errors.required">A message is required</div>
  </div>
</label>

<input type="submit" value="Send message" class="cta">

</form>

<div *ngIf="submitted" class="results">
  <strong>Name:</strong>
  <span>{{ messageForm.controls.name.value }}</span>

  <strong>Message:</strong>
  <span>{{ messageForm.controls.message.value }}</span>
</div>
```

Baked in here is a full form with validation. It also prints out the form values beneath it when the form has been submitted.

Let's update the css for the component to make it look decent:

```
label {
  display: block;

  input, textarea {
    display: block;
    width: 50%;
    margin-bottom: 20px;
    padding: 1em;
  }

  .error {
    margin-top: -20px;
    background: yellow;
    padding: .5em;
    display: inline-block;
    font-size: .9em;
    margin-bottom: 20px;
  }
}

.cta {
  background: #7700FF;
  border: none;
  color: white;

  text-transform: uppercase;
  border-radius: 4px;
  padding: 1em;
  cursor: pointer;
  font-family: 'Montserrat';
}

.results {
  margin-top: 50px;

  strong {
```



```
margin-bottom: 20px;  
display: block;  
}  
}
```

Save it, and the result in the browser should look like this!

myappHOMEABOUTCONTACT US

Contact us

Name:

Message:

A message is required

SEND MESSAGE

Name:
Gary

Message:

Awesome!

Conclusion

As you can see, Angular 7 is quite powerful but we've only just scratched the surface. If you're a beginner, I recommend trying to create a really simple project using what you've learned so far and then take it from there.

Like 86 people like this. Be the first of your friends.

Tweet



Share this post



Say something about this awesome post!

What do you think?

994 Responses



77 Comments

Coursetro

Login

Recommend 22

Tweet

Share

Sort by Best



Join the discussion...

LOG IN WITH

OR SIGN UP WITH DISQUS ?

Name



Efe Ativie • 7 months ago

This is really awesome! Your site is a huge resource for developers! Thanks bro!

11 ^ | v • Reply • Share ›



Matthew Fedak • 5 months ago

Easy and simple to follow tutorial. Thanks.

5 ^ | v • Reply • Share ›



Prasad K.P. → Matthew Fedak • 25 days ago

hi dude

^ | v • Reply • Share ›



Shrinivas Kalangutkar • 8 months ago • edited

Nice tutorials, For more AngularJS concepts .(<https://www.tutespace.com>)

6 ^ | v • Reply • Share ›



Muhammed Abdul • 8 months ago

Love your short , apt and to the point courses. It would be great if you can also mention how this differs from Angular 6 learn by example so that we can focus on the new areas and explore them more later.

4 ^ | v • Reply • Share ›



Cristina Martin • 2 months ago

Very good tutorial! Amazing bro!



Amit Singh • 2 months ago

Awesome sir...!!

2 ^ | v • Reply • Share ›



Amit Singh • 2 months ago

Awesome sir...!!

1 ^ | v • Reply • Share ›

```

    return result.trim() ? {return of({})} :
    if (!term.trim()) {return of({});}
    let headers = new HttpHeaders().set('Content-Type', 'application/json');
    let params = new HttpParams().set('Gpclave', term );
    return this.http.get<listademos[]>(`${ubi}/${params, headers}}
    .pipe( retry(2),
    tap(listaDemos => console.log(listaDemos) ) )
  }

```

If term=Peter get all info , now if term=Pe, get nothing, is like set only looks for exact match.

Any idea how to solve that ??? Thx

1 ^ | v • Reply • Share ›



Binu E • 6 months ago • edited

How to implement this with [angular material](#) ?

1 ^ | v • Reply • Share ›



lesly malatji • 3 days ago

Wow, a very great tutorial, you made me to start understanding angular, great work thanks a lot for the tutorial Mr Simon

^ | v • Reply • Share ›



Edwin Agik • 11 days ago

awesome stuff, simple to understand and straight forward

^ | v • Reply • Share ›



Siete De Espadas • 17 days ago

Oh man, this was absolutely awesome! Thank you very much, keep going please.

^ | v • Reply • Share ›



Kennedy Ngigi • 23 days ago

Nice tut. I'm looking for a details page for users. How can we output their details to another page when one clicks any user???

^ | v • Reply • Share ›



sushil sinha • 2 months ago

great

^ | v • Reply • Share ›



rana • 2 months ago

Awesome Job! Thanks so much.

^ | v • Reply • Share ›



Sukhvair Singh • 2 months ago

This is really awesome! Your site is a huge resource for beginners! Thanks bro!

^ | v • Reply • Share ›



Mahadevan Annamalai • 2 months ago

This is really awesome. Thanks for your hard work and posted.. Thanks bro!

^ | v • Reply • Share ›



Lasang Jimba Tamang • 3 months ago

Really a nice tutorial for begginer

^ | v • Reply • Share ›



Martin Pan • 3 months ago

What I want to say is how a good tutorial.

^ | v • Reply • Share ›



Eng.Mohamad Alhamid • 3 months ago

Thank you, you have idea for debug vscode line by line??

^ | v • Reply • Share ›



Aleeza Ahmed • 3 months ago

Thank you for this post

^ | v • Reply • Share ›



jlz • 3 months ago • edited

My components get generated with .sass instead of .scss. Wierd!

^ | v • Reply • Share ›



Haseena Arafath → jlz • 3 months ago



Spend less time discussing bugs and changes. BugHerd collects feedback and tracks bugs, like using sticky notes on a website.

Try BugHerd Free

x

SCSS. Please check if this was selected correct.

Thank you

^ | v • Reply • Share ›



Anil Virat • 3 months ago

Thank you so much

^ | v • Reply • Share ›



Kudzaishe Tatenda Shoko • 3 months ago

For the first time i now understand what i'm doing. I'm a really slow learner but this came through.

^ | v • Reply • Share ›



Jonathan • 3 months ago • edited

thank you

^ | v • Reply • Share ›



Villads Spring Ruby • 3 months ago

Great tutorial! And great resources, thanks :)

^ | v • Reply • Share ›



LFX • 4 months ago

Great tut! you should write a book!

^ | v • Reply • Share ›



Gors • 4 months ago

When i created component it generated component.sass instead of component.scss, Hence i have to create style sheet according to SASS i.e. by removing { & } with proper indentation

^ | v • Reply • Share ›



Benoit Tournon • 4 months ago

Great tutorial. Thanks.

However, I had to add the dependency in package.json : "@angular/http": "~7.0.0"

Else, I got an error at runtime regarding HttpClient.

^ | v • Reply • Share ›



Arthur Bowers • 4 months ago • edited

Thank you for this post.

I'm new to Angular and typescript, and this is the best Angular CLI guide I've found so far, clearly explaining components, services and routing and more.

Again, thank you truly!

How would you implement a keyword search matching one or more keywords?

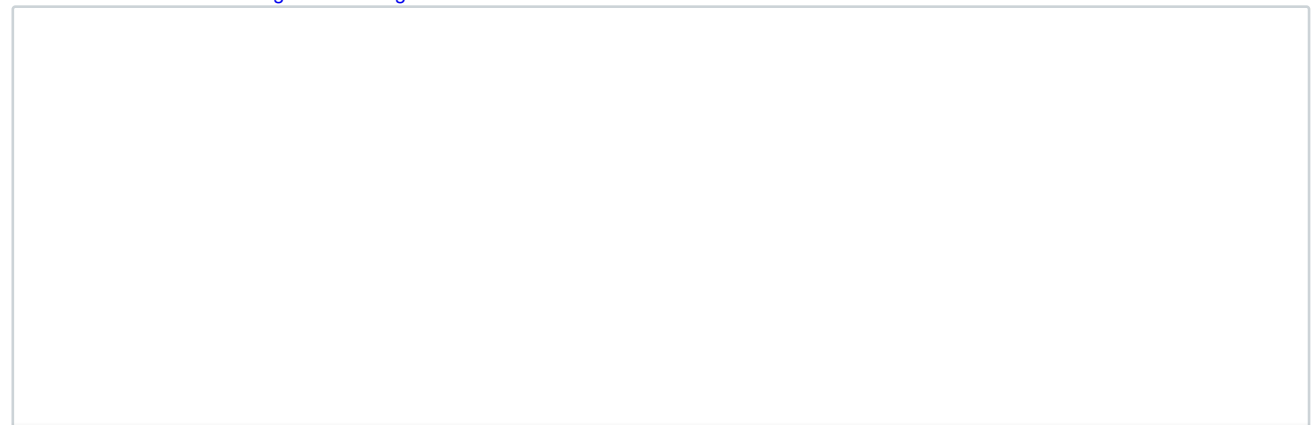
^ | v • Reply • Share ›



ITS Tech School • 4 months ago

Well Explained Bro. I have learn a lot from your tutorial

You can also check it out : [Angular Training Videos](#)



see more

^ | v • Reply • Share ›



crispus kinyanjui • 4 months ago

never found a better and a simple explanation as that. Thanks a bunch

^ | v • Reply • Share ›

back to home you keep doing multiple http calls, is there any way to prevent that? Performance and request quota consideration :) Great tut anyways

see more

^ | v • Reply • Share ›



Old Grimie → gfmadness • 4 months ago

When you click the Home link, it takes you to the Home component, that makes a new Init call. So, is the same as opening a page again, that's what I see.

Because Angular is not leaving the index.html file at all but displaying the component's content inside the 'router-outlet' tag, you can still having the console history

^ | v • Reply • Share ›



Mukesh Singh • 5 months ago

What to say ? your tutorial is very nice, simple and easy to understand. i am new in angular and we have learn a lot of new things from here.

^ | v • Reply • Share ›



deepesh • 5 months ago

this is really awesome course thank you so much for this course. you are such a great . thanks a lot

^ | v • Reply • Share ›



ravinder reddy • 5 months ago

Nice video for quick learning. Thanks bro for providing such a video.

^ | v • Reply • Share ›



Aakash Chauhan • 5 months ago

awosm

^ | v • Reply • Share ›



Kaushik Ganguly • 5 months ago

Thanks for a nice tutorial, helped me to understand angular.

^ | v • Reply • Share ›



whiteadi • 5 months ago

nice

^ | v • Reply • Share ›



Brij • 5 months ago

Just Awsome!!

^ | v • Reply • Share ›



Luis Rosales Carrera • 5 months ago

awesome man thank you!

^ | v • Reply • Share ›



Sukhdev • 5 months ago

I asked for Angular 7


^ | v • Reply • Share ›



Sukhdev • 5 months ago

Jhakkas....Sir,,
you r having good explanation way,
but i having below problem so,

I have added my all external JS file in Index.html file of my project and it's work too good but when i change routs that time external is file is stop




Spend less time discussing bugs and changes. BugHerd collects feedback and tracks bugs, like using sticky notes on a website.

Try BugHerd Free

x


^ | v • Reply • Share ›



Roberto Fernandes • 5 months ago

Awesome!!! Thank you very much.


^ | v • Reply • Share ›



Arun Kumar • 6 months ago

hello sir,
I need your help to how to use feature-discovery in material design to use in angular-material


^ | v • Reply • Share ›



영밍맛 • 6 months ago

Noted Thanks!


^ | v • Reply • Share ›



Linh Hồ • 6 months ago

Thanks very nice

^ | v • Reply • Share ›




Moovendhan Elanchezhiyan • 6 months ago • edited

Kind of stuck with route. Only default path - " works. Neither '/contact' nor '/about' from home page works. But when hitting localhost:4200/about or localhost:4200/contact directly it works. Looks the click event is not propagating and not seeing any error in console.

^ | v • Reply • Share ›


Load more comments


ALSO ON COURSETRO



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


3 comments • a year ago

 Santosh Nirankari — just went through it completely, you should share it Avataion hackr as well. It'll be nice. <https://hackr.io/tutorials/...>



9. Home Page CSS Styling

1 comment • a year ago

 pires joao — hey..why in my project when i was finish this section,Image Avatathat i put,not showing in browser..