

# How To Build A News Application With Angular 6 And Material Design

*In this article, we're going to build a news application using Angular 6 and Google's material design in combination, which will help you to make your future applications with Angular look great in web browsers and mobile devices.*



Are you looking to combine Google's material design with Angular applications? Well, look no further!

In this tutorial, we're going to build a news application using two of the most powerful and popular resources out there, Angular 6 and material design. You'll learn how to incorporate Google's material design components into Angular application templates to change and style your application in a professional way. The tutorial also serves as a reminder of how to make HTTP requests to bring live news articles to an application using the News API.

Before we get started building the app, let's quickly review the resources we're going to use, Angular and material design, and see why we've paired them to build this application?



*A news application with Angular 6 and Material Design. ([Large preview](#))*

## What Is Angular?

Angular — according to the [official documentation](#)— is described as follows:

*“Angular is a platform that makes it easy to build applications with the web. Angular combines declarative templates, dependency injection, end-to-end tooling, and integrated best practices to solve development challenges. Angular empowers developers to build applications that live on the web, mobile, or the desktop.”*

In short, it's the most powerful JavaScript framework for building highly interactive and dynamic web applications.

*“As mentioned, Angular is powerful, but also popular, which is why companies such as Upwork, Freelancer, Udemy, YouTube, Paypal, Nike, Google, Telegram, Weather, iStockphoto, AWS, Crunchbase are using it.”*

## What Is Google's Material Design?

Material design is a design language introduced by Google in the summer of 2014 for Android's new OS. Although its initial focus was touch-based mobile apps, now its functionality has been extended to reach the web design world.

It's an adaptable system of guidelines, components, and tools that support the best practices of user interface design. It's also backed by open-source code and supported by a large community of designers and developers who are collaborating together to build beautiful products.

## Why Angular And Google's Material Design Specifically?

It's a matter of choice. No JavaScript framework is better than another. It's all about what your project needs. The same goes for programming languages.

Now, I'm not going to outline the benefits and features of Angular. Instead, I'm going to share with you why I've picked Angular specifically to build a news application.

As is always the case with any news application, communicating with back-end services over the HTTP protocol is a crucial part. This is where the newer Angular **HttpClient**

**module**, which is an improved version of the old Http, can help us easily interact with the service API.

**The model-view-viewmodel (MVVM)** of Angular will be handy when it comes to binding the remote data that will be stored in objects into our application template, where the component plays the part of the controller/viewmodel and where the template represents the view. This is what we call the Angular template language.

**The two-way binding system**, which means that any changes in the application's state will be automatically reflected into the view, and vice versa. You'll notice that when selecting the news resources from the side menu, that will change the state of our news article.

What I like most about Angular is the **SPA technology**. Loading only the part of the page that needs to be changed will definitely help our application load and perform more quickly and smoothly.

Of course, there are many other benefits and features of Angular, which you can look up with a quick online search.

## What About The Visual Aspect?

We've chosen material design because its language is a suitable fit for Angular, and it's easy to implement.

It's also a very popular visual language; it's responsive, and most Google apps are built with it. We want our app to look as much like a Google app as possible.

As an introduction, that's all we need. It's time to look at the project overview and then jump into the build process.

[Project Overview](#)

[Prerequisites](#)

[Setting Up An Angular Project](#)

[Installing Dependencies](#)

[Acquiring Free API Key](#)

[Working On Components](#)

[Defining Material Default Style](#)

[Define A Template](#)

[Conclusion](#)

## Project Overview

*“Getting the latest live news articles from a range of sources, including BBC News, CNN, TechCrunch, Huffington Post and more, along with different categories, like technology, sports, business, science and entertainment.”*

This is how your application will look when you finish it:



*Project overview. ([Large preview](#))*

That should get you excited, shouldn't it? Let's start by building the app.

## Prerequisites

This is what you're going to need in order to follow along with this tutorial:

[Node.js](#) and npm installed on your machine;

[Angular CLI](#) installed on your machine;

A basic understanding of **Angular**.

Once that stuff is out of the way, we can proceed.

## Setting Up The Angular Project

In this section, we're going to use the Angular command line interface (CLI) to generate a new Angular project. To do so, head over to the CLI and run this:

```
ng new news-app
```

Next, point your command line to the project's root folder by running the following:

```
cd news-app
```

## Installing Dependencies

To set up our dependencies, we're going to install, with just one command, all of the dependencies necessary for this tutorial. Don't worry, I'll explain this in a second:

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

We have three packages being installed with this command.

### **@ANGULAR/MATERIAL**

This is the official material design package for the Angular framework.

### **@ANGULAR/ANIMATIONS**

Installing the Angular animation package separately from the Angular core library is necessary. Certain material components need access to the animation libraries, which is why we're installing it here.

## @ANGULAR/CDK

The **CDK** part stands for “component dev kit”, which provides us with high-quality predefined behaviors for your components, since modern web development is all about components.

It is recommended to include the Angular CDK any time you want to link Google’s material design to an Angular application.

To find out more about Angular CDK, check out this [article](#).

Let’s run our app to see that everything works just fine. You can start a development server by running the following command:

```
ng serve
```

Now, if you visit <http://localhost:4200/> in a browser, you should see the following page:



*Running the Angular project on development server. ([Large preview](#))*

Now, in your code editor, navigate to the file `/src/app/app.module.ts`, and add the following packages that we’ve just installed:

```
... Other imports ...  
import { BrowserAnimationsModule } from '@angular/platform-browser/animations';  
import { MatButtonModule, MatCardModule, MatMenuModule, MatToolbarModule, MatIconModule, MatSidenavModule
```

It is important to understand what's going on here. First, we're importing the animations package to animate our application a bit.

*The next import is what's unique to Angular material. Before, we just included a single material module. Now, we have to import each material component that we intend to use.*

As you can see, we've added seven different modules here for material buttons, cards, menus, lists toolbars, side navigation, and icons.

After adding those packages to your `app.module.ts` file, make sure that your file matches the following:

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { HttpClientModule } from '@angular/common/http';
import { NewsApiService } from './news-api.service';

import { BrowserAnimationsModule } from '@angular/platform-browser/animations';
import { MatButtonModule, MatCardModule, MatMenuModule, MatToolbarModule, MatIconModule, MatSidenavModule, MatListModule } from '@angular/material';

import { AppComponent } from './app.component';

@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule,
    BrowserAnimationsModule,
    HttpClientModule,
    MatButtonModule,
    MatMenuModule,
    MatCardModule,
    MatToolbarModule,
    MatIconModule,
    MatSidenavModule,
    MatListModule,
  ],
  providers: [NewsApiService],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

**Note:** *The statement `import { HttpClientModule } from @angular/common/http` in the file above wasn't generated automatically, but rather added manually. So, make sure you do that, too. And don't worry about the **NewsApiService** service provider because we're going to take care of that later on.*

You might be wondering, though, how did I know the names of the modules to import? The official [Angular material documentation](https://material.angular.io/) gives you the exact code needed to import each module.



If you click on any of the components in the left menu and then click on the “API” tab, it provides you with the exact import line that you need to use.



*API reference for Angular Material Component. ([Large preview](#))*

In terms of setup, that's all we need to do before we actually begin using and integrating material components in our templates.

You just have to remember to import each unique component that you plan to use.

## Acquiring Free API Key

We're going to use the [News API](#) to feed us some news headlines as JSON data, which we'll implement in the application template.

What is the News API service?

The News API is a simple HTTP REST API for searching and retrieving live articles from all over the web.

Now that you know what the News API is, the next step is to get a free [API Key](#), which will help us make some call requests to the server and grab the news articles.

You can sign up for just 30 seconds. You'll only need to provide your first name, email address, and password. That's all.

After signing up, you'll find the API key already generated for you in the dashboard. Just save it in a text file somewhere on your desktop; because we'll use it in the next chapter.

# Working On The Components

To start working on the components, you need to create a service provider to manage the interaction with the News API service.

## CREATING THE SERVICE PROVIDER

Enter this command to generate a new service provider:

```
ng generate service NewsApi
```

After that, go to the generated `/src/app/news-api.service.ts` file, and add the following code to it:

```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';

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

  api_key = 'PUT_YOUR_API_KEY_HERE';

  constructor(private http:HttpClient) { }
  initSources(){
    return this.http.get('https://newsapi.org/v2/sources?language=en&apiKey='+this.api_key);
  }
  initArticles(){
    return this.http.get('https://newsapi.org/v2/top-headlines?sources=techcrunch&apiKey='+this.api_key);
  }
  getArticlesByID(source: String){
    return this.http.get('https://newsapi.org/v2/top-headlines?sources='+source+'&apiKey='+this.api_key);
  }
}
```

It's time to use our **API Key**. Just paste it where it says, "Put\_YOUR\_API\_KEY\_HERE".

We've imported `HttpClient`, which will be responsible for making API calls to our endpoints and fetching news headlines for us.

Now, for the `initSources` function, we simply prepare our left-side menu with some news resources. After that, we've created another function, `initArticles` which retrieves the first articles from **TechCrunch** once the application gets started.

As for the last function, `getArticlesByID`, it's going to simply bring some articles for the passing parameter.

## THE MAIN COMPONENT

The service provider is done. Let's move to the `/src/app/app.component.ts` file and add this code:

```
import { Component } from '@angular/core';
import { NewsApiService } from './news-api.service';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {

  mArticles:Array<any>;
  mSources:Array<any>;

  constructor(private newsapi:NewsApiService){
    console.log('app component constructor called');
  }

  ngOnInit() {
    //load articles
    this.newsapi.initArticles().subscribe(data => this.mArticles = data['articles']);
    //load news sources
    this.newsapi.initSources().subscribe(data=> this.mSources = data['sources']);
  }

  searchArticles(source){
    console.log("selected source is: "+source);
    this.newsapi.getArticlesByID(source).subscribe(data => this.mArticles = data['articles']);
  }

}
```

We're defining two properties here: **mArticles**, for holding news articles, and **mSources**, for holding news resources. Both are defined as an array.

In the constructor, we're simply creating a **NewsAPIService** instance.

Next, we're using that instance on the **ngOnInit()** function to initialize our two properties.

For the **searchArticles** function, it will be triggered whenever the user selects a specific resource from the left-side menu. Then we're passing this parameter to the **getArticlesByID** service provider function to retrieve articles for it.

## Defining Material's Default Style

In our `/src/styles.css` file, which is generated by the Angular CLI, let's add the following:

```
@import '~@angular/material/prebuilt-themes/indigo-pink.css';  
body {  
  padding: 2em 23em;  
  background:lightgray;  
}
```

Based on your preference, you can change `indigo-pink.css` to:

`deeppurple-amber.css`

`indigo-pink.css`

`pink-bluegrey.css`

`purple-green.css`

I'm also adding some CSS to the `body` tag, only to demonstrate this layout. This helps it look more like an app, even on desktop.

Let's also add two lines to our `/src/index.html` file just before the closing `head` tag:

```
<link href="https://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">  
<link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Roboto:300,400,500,700,400italic"
```

The first line imports the material design icon font, and the second one is the Roboto font, which is used by the material design team.

## Defining A Template

Let's start off by adding the following template in the `/src/app/app.component.html` file:

```

<mat-toolbar color="primary">
  <button mat-button (click)="sidenav.open ()" ><mat-icon>menu</mat-icon></button>
  <span>News Headlines</span>
  <span class="example-spacer"></span>
  <button mat-button [matMenuTriggerFor]="appMenu"><mat-icon>settings</mat-icon></button>
</mat-toolbar>
<mat-menu #appMenu="matMenu">
  <button mat-menu-item> Settings </button>
  <button mat-menu-item> Help </button>
</mat-menu>
<mat-sidenav-container class="example-container">

  <mat-sidenav #sidenav class="example-sidenav">
    <mat-list class="list-nav">
      <mat-list-item class="list-item" *ngFor="let source of mSources" (click)="searchArticles(source)">

        <div mat-card-avatar [ngStyle]="{'background-image': 'url(../assets/images/' + source.id + '.p"

        <span class="source-name"> {{source.name}}</span>

      </mat-list-item>
    </mat-list>
  </mat-sidenav>
  <mat-card class="example-card" *ngFor="let article of mArticles">
    <mat-card-header>
      <div mat-card-avatar [ngStyle]="{'background-image': 'url(../assets/images/' + article.source.id"
      <mat-card-title class="title">{{article.title}}</mat-card-title>
      <mat-card-subtitle>{{article.source.name}}</mat-card-subtitle>
    </mat-card-header>
    <img mat-card-image class="img-article" src={{article.urlToImage}} alt="">
    <mat-card-content>
      <p>
        {{article.description}}
      </p>
    </mat-card-content>
    <mat-card-actions class="action-buttons">
      <button mat-button color="primary"><mat-icon>thumb_up_alt</mat-icon> 12 Likes</button>
      <button mat-button color="primary"><mat-icon>comment</mat-icon> Comments</button>
      <button mat-button color="primary"><mat-icon>share</mat-icon> Share</button>
      <a mat-button color="primary" href={{article.url}} target="_blank" ><mat-icon>visibility</mat-icon>
    </mat-card-actions>
  </mat-card>
</mat-sidenav-container>

```

So, what have we done here?

First, we define a toolbar with a left-side menu, along with the application's main title and the settings' right menu.

Next, we're using `*ngFor` for both sources and articles, and in doing so, our left-side menu will hold the news resources, and the main contents will hold the news articles.

One thing to notice is that on the `click` event of our list items, we've added two functions because that event executes any JavaScript code. The first function is `searchArticles`, which we've already explain, and the second one is `sidenav.close()` which will automatically close our left-side menu once the user has selected a resource.

## STYLING OUR COMPONENT

The last thing to do with the components is to visit the `/src/app.component.css` file and paste the following code in it:

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

.example-card{
  margin-top: 4px;
}

.example-header-image {
  background-size: cover;
}

.title{
  font-weight: bold;
}

.img-article{
  height: 350px;
}
```

```
.action-buttons{
  text-align: center;
}

.example-container {
  width: 100%;
  height: auto;
  border: 1px solid rgba(111, 111, 111, 0.50);
}

.example-sidenav-content {
  display: flex;
  height: 75%;
  align-items: center;
  justify-content: center;
}

.example-sidenav {
  padding: 20px;
}

.source-name {
  margin-left: 5px;
}

.list-item:hover{
  cursor: pointer;
  background-color: #3f51b5;
  color: white;
}
```

## SET UP IMAGES FOR NEWS RESOURCES

Move to the `/src/assets` directory, and create a new folder named `images`. Then, download these images either from a [Google Drive link](#) or the [GitHub repository](#).

They are the logos of our news resources. Once you download them, copy and paste all of the image files into the `images` folder that you just created.

Once everything is complete, run this: