Build a To-do List App Using the Ionic 4 Framework

This tutorial will help you build a to-do list mobile application, using the Ionic 4 Framework, within a few minutes.

Mobile apps have become a crucial part of our lives. For example, if you are hungry, you can use an app to order some food. Bills pending? You can pay them using digital payment apps. Want to buy new clothes? Open an e-commerce mobile app, choose what you like, and purchase it. How very practical!

Today there are a number of mobile platforms for apps. Android and the iOS mobile OS platforms have the highest market share. Apps can be developed for each specific platform or device but doing so can be a tedious task. Let's say you have an idea for an app and want to build it for each of your target platforms—Android and iOS. More efforts and resources will be required, as it will be like developing the same app twice using platform-specific SDKs and APIs. This can be time consuming and difficult to maintain. What if there was a framework or SDK to build an app that will run on all platforms? This is where Ionic, a hybrid mobile application development framework, comes into the picture.

The Ionic Framework

The Ionic Framework is a complete open source SDK for hybrid mobile app development. The original version was built on top of AngularJS and Apache Cordova. The latest release, Ionic 4, was rebuilt as a set of Web components, which allow the user to select any modern UI framework, such as Angular, Vue or React. You can also use Ionic components without any UI framework. Ionic provides tools and services for developing hybrid mobile apps, desktop apps and progressive Web apps based on modern Web development technologies like HTML, CSS and JavaScript. We can build mobile apps using these Web technologies, and by leveraging Cordova or Capacitor, we can distribute the apps through native app stores for installation on devices.

Ionic is one of the most popular and easy-to-learn hybrid mobile app development frameworks. With over 40,000 stars and almost 13,000 forks at the time of writing this article, it is also one of the most popular and actively maintained mobile development frameworks on GitHub. Thousands of mobile app developers around the world use Ionic to develop mobile apps.

Features of the Ionic Framework

The Ionic Framework uses Cordova and has recently added Capacitor plugins to gain access to the host operating system's features and to features of mobiles such as the camera, flashlight, GPS, etc. Ionic allows app building and deployment by wrapping around the Cordova or Capacitor plugins with a simplified 'Ionic' command line tool. Users can build their apps using modern Web technologies, and these can be customised for Android, iOS, Desktop (with Electron) and modern browsers.

Besides the SDK, Ionic also provides services that developers can use to enable features, such as code deploys and automated builds. Ionic apps run with partial native code and Web code, providing full access to native functionality if necessary, with the UI of the app built with modern Web technologies.

Tip: The Ionic Framework is open source and its source code is available on GitHub. Developers and programmers are encouraged to participate and contribute to the project.

Pre-requisites for building the app

Before starting to build your first Ionic app, there are few pre-requisites. You need to install a few of the following apps.

Node.js: Download and install Node.js (*https://nodejs.org/*). Ionic is also distributed as an npm package, so we will use Node and npm to install the Ionic Framework.

TypeScript: TypeScript is the superset programming language of JavaScript (we can think of it as a typed version of JavaScript) and we will code our app in it. After installing Node.js, using a terminal run the following command:

```
npm install -g typescript
```

Cordova: Cordova helps us to convert our HTML, CSS and JavaScript/TypeScript code into an app.

To install it, run the following command:

```
npm install -g cordova
```

Ionic: And now, install Ionic Framework as follows:

```
npm install -g ionic
```

Tip: You can also install these in one go by running the following command in the terminal/console:

```
npm install -g typescript cordova ionic
```

Note: It is assumed that you have some basic knowledge of Web technologies like HTML, CSS and JavaScript. If you don't, W3Schools *(http://www. w3schools.com/)* is a good place to start. The site has some great tutorials for Web technologies that are easy to follow.

Setting up the environment

Let's start. Type the following command in the command prompt console:

```
ionic start
```

You will be asked to name the project. Give it a name, e.g., Todo App, etc. You will then be prompted to pick a framework. In our case, we will pick Angular. Then we will be asked to select a template — we will select a blank one.

```
? Starter template: (Use arrow keys)
> tabs          | A starting project
with a simple tabbed interface
  sidemenu      | A starting project
with a side menu with navigation in the
content area
  blank         | A blank starter
project
  my-first-app | An example application
that builds a camera with gallery
  conference    | A kitchen-sink
application that shows off all Ionic
has to offer
```

After the template is selected and built, go to your newly created project directory and run *ionic serve* within the app directory to see your app.

```
ionic serve
```

The command will start the local development server. Next, it will load *http://localhost:8100/home* in your browser. If you see the output screen shown in Figure 1, you have successfully installed all the dependencies required.

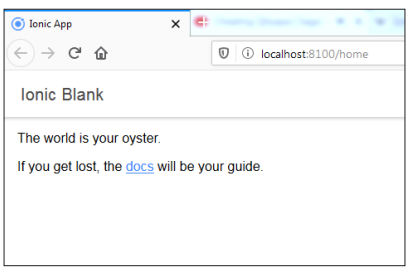

Figure 1: Ionic project running in the browser

Getting started

Let's have a look at the project file structure (see Figure 2).

As shown in Figure 2, the main source files consist of the following.

- *app.module.ts*: This is the entry point of the mobile app. All components, modules, pages, etc, must be added to this file.
- *app.component.ts*: This is the first logic that gets executed when the app starts running.
- *app.html*: This file contains the view/template of the app; other UI pages get mounted here.
- *app.scss*: This file contains all the CSS styling and SASS variables to be used globally within the app.

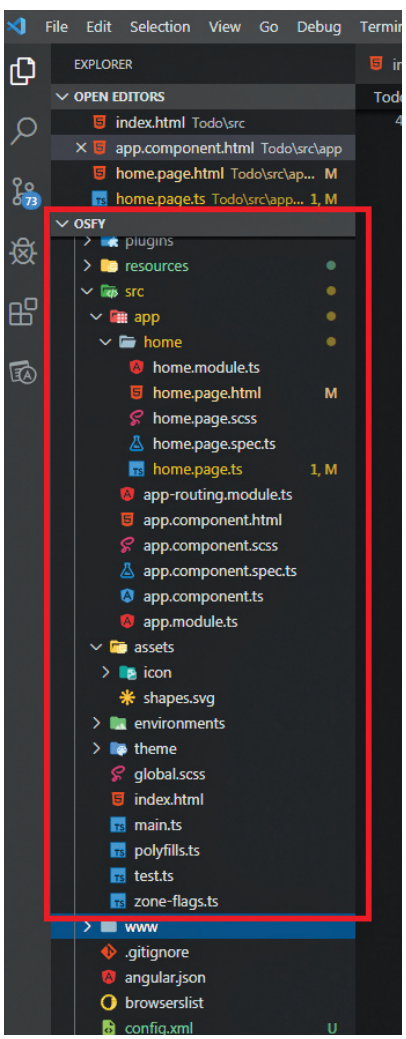

Figure 2: Project file structure

The app that we are going to build is called the ToDo app. With this app, users can list the tasks that need to be done and delete them when completed.

Now let's go to the home component, which is where we will build our app's UI and logic.

The home component has three files:

- *home.component.ts* - In this file, we will add the logic of our app, i.e., the logic to add and delete the task in our app.
- *home.html* - This file consists the UI part of the app; we will be using HTML here.
- *home.scss* - This file contains all the CSS styling of the app and the SASS variables specific to this page.

Creating the UI and the logic of the app

Figure 3 shows the wireframe of the app we are creating. It has a text input where the user can enter the task, a button to add that task to the list, the list to display the added tasks, and a *Delete* button to delete the task when it is completed.

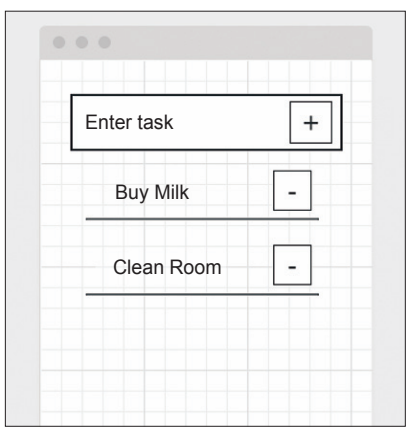

Figure 3: Wireframe diagram of the app

Now let's open the code editor and start coding.

First, let us design the UI of the app, *open home.component.html*, and add the following HTML code:

```
<ion-header>
  <ion-toolbar>
    <ion-title>
      ToDo App
    </ion-title>
  </ion-toolbar>
</ion-header>
<ion-content padding>
  <ion-item>
    <ion-input type="text"
placeholder="Enter task"
      [(ngModel)]="taskName"></ion-
input>

    <div class="item-note" item-end>
      <ion-button color="primary"
(click)="addTask()">
        <ion-icon name="add"></ion-icon>
      </ion-button>
    </div>
  </ion-item>
  <div padding>
    <ion-list>
      <ion-item *ngFor="let todo of
taskList; let i = index">
        {{todo}}
        <div class="item-note"
slot="end">
          <ion-button color="danger"
clear (click)="deleteTask(i)">
            <ion-icon name="trash"></
ion-icon>
          </ion-button>
        </div>
      </ion-item>
    </ion-list>
  </div>
</ion-content>
```

We have added <*ion-header*> with <*ion-toolbar*> and <*ion-title*>. This displays the title of the app at the top. Then we have added our UI components inside <*ion-content*>. The <*ion-input*> component is where the user will enter the task, and the <*ion-button*> component when clicked will add the task to <*ion-list*>.

<*ion-list*> will display the list of tasks added to it. As you can see, we are iterating over the *taskList* variable using the **ngFor* Angular directive. Then, finally, we have added an <*ion-button*> to each item in the list; when this button is clicked, the task will get deleted.

Here, we are calling the *deleteTask(i)* function and we are also passing the current index; this deletes only the list item matching the current index.

Tip: You can read more about Ionic components and how to use them from the official documentation at *https://ionicframework.com/docs/components*.

To add the logic to the app, open *home.component.ts* and add the following code:

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

@Component({
  selector: 'app-home',
  templateUrl: 'home.page.html',
  styleUrls: ['home.page.scss'],
})
export class HomePage {

  taskName: any = ''; // Entered Text
  taskList = []; // Array to store
tasks

  constructor() {}

// addTask Function
// First we check if the text is
entered or not in input box by
verifying if length > 0
// If length is greater than 0, then
only we add taskName to taskList array
// After adding we reset the taskName
  addTask() {
    if (this.taskName.length > 0) {
      let task = this.taskName;
      this.taskList.push(task);
      this.taskName = '';
    }
  }
// deleteTask Function
// When user clicks the delete task
button, this function is called with
index i as parameter
// Since tasks are added to taskList,
we delete the task at index i using
splice() array method
```

```
// This deletes only that task at index i
  deleteTask(index) {
    this.taskList.splice(index, 1);
  }
}
```

Here, we first import *Component*, which we need as we are going to create a component called 'app-home'.

```
@Component({
  selector: 'app-home',
  templateUrl: 'home.page.html',
  styleUrls: ['home.page.scss'],
})
```

Tip: You can learn more about Angular and Typescript syntax at *https://angular.io/docs* and *https://www.typescriptlang.org/*.

We use the *@Component* decorator, and then we set the selector name, *templateUrl*. This is set to be loaded when we use the *<app-home>* selector. *styleUrls* loads the style for this selector.

Next, in the class *HomePage*, we set two variables — *taskName* which is a string, and *taskList = []* which is an array.

Then we write the *addTask()* and *deleteTask()* functions. I have added the explanation for these functions in code comments.

Building the app

Note: Before building the APK file, make sure the Android SDK is installed. This is required to build the APK file of the app. You can follow the steps given in the Ionic documentation article at *https://ionicframework.com/docs/installation/android*.

We are done with the coding part. Now let us build the *.apk* file of the app so that we can install it on our Android mobile phone. Go to the project directory and run the following command to build the project for the Android platform:

```
ionic cordova build android
```

This command generates a *.apk* file at *\platforms\android\app\build\outputs\apk\debug* in the project directory with the name *android-debug.apk*.

Tip: You can use the *build* command to build the app for multiple target platforms such as iOS, Windows Phone, etc, by suitably replacing the keyword 'android' in the above command statement.

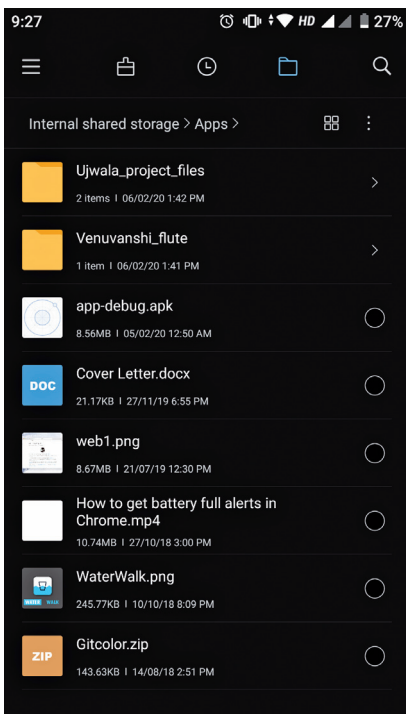

Figure 4: The 'android-debug.apk' file copied to the phone

Installing and testing the app

Now copy or transfer the *android-debug.apk* file to your phone via a USB cable or Bluetooth, as shown in Figure 4, and install it. While installing, you may receive an 'Install blocked' message, as shown in Figure 5. You need to select *Install Anyway*, as shown in the same figure. This will now give you permission to access the app and will install the app. After installing the app, open it as shown in Figure 6. You can now start using your app, and can add or delete to-do tasks.

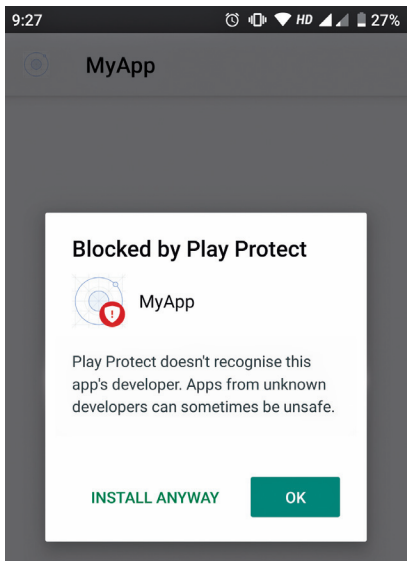

Figure 5: 'Install blocked' warning message

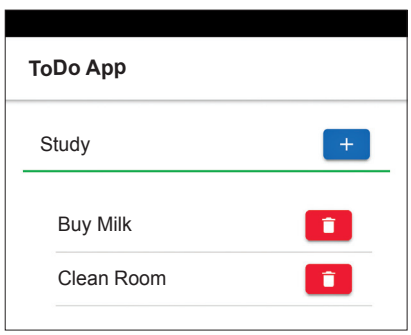

Figure 6: The ToDo list app opened on an Android mobile phone

Note: You can also refer to and download this project's source code from my GitHub repository. Visit *https://github.com/aniketkudale/Ionic-4-ToDo-App/*.

END

References

[1] *https://ionicframework.com/*
[2] *https://ionicframework.com/docs*
[3] *https://github.com/ionic-team/ionic*

By: Aniket Eknath Kudale

The author is an open source enthusiast with more than five years of experience as a senior member of the technical staff at TIBCO Software Inc., Pune.