

## How to translate in Ionic 5— Internationalization and Localization



In this post you'll learn how to translate text in Ionic 5 apps and PWA. You will also learn how to get device specific language and convert your app's text to the same language / locale.

Ionic has a variety of app types nowadays (Angular/React/Vue , Cordova/Capacitor). This post will explore the Ionic apps made in Angular Cordova, but the same process can apply in Angular Capacitor apps as well.

Multi-language translation, OR internationalization is a growing need for every app that wants to target international customers. Till date, majority of apps have been developed in English, no surprise ! But with growing apps users, every app is now focusing on translating to local languages.

First we need to understand the steps involved in the process. Translations, thanks to Google, look very easy, but they are a bit involved in case of apps and websites, in our case Ionic 5 apps. There are 3 major steps in the translation process in an app—

1. **Translation Language**—Detect the language you want to translate into. This can either be done automatically by detecting phone or browser’s language (Globalization), OR, can be done with user actions (say, by using a dropdown).  
For our use case, we will detect device’s language using both [Cordova Globalization plugin](#) and Browser’s Internationalization API.
2. **Prepare Language Text**—You need to have a pre-translated dictionary (or JSON file) which stores the translations of all your app’s text in the Translation language. There are several ways to do this, as we’ll see in following steps. This is mostly a manual process for smaller apps, but you can use online tools like [this](#) for quick translations, or like [POEditor](#) for more standardized workflow.
3. **Translate**—After the above two steps, you finally translate your app’s text to the intended language. We will use [ngx-translate](#) library for translating our texts as we are talking about only Angular Ionic apps in this post

## Structure of the post

So the development outline of this blog will be

1. Create a starter Ionic 5 tab app
2. Prepare multiple language JSON files in assets
3. Implement [ngx-translate](#) library to detect and translate AND Implement [Cordova Globalization plugin](#) or browser Internationalization API to detect device language
4. Test translations on browser
5. The Directive Gotcha
6. Setup stand alone translations
7. Test translations on Android / iOS

We will translate English text in 2 languages—**French** and **Spanish**

Sounds pretty easy, right ? Well, let’s dive right into it.

## Step 1— Create a basic Ionic Angular app

First you need to make sure you have the latest Ionic CLI. This will ensure you are using everything latest. Ensure latest Ionic CLI installation using

```
$ npm install -g ionic@latest
```

Here’s my environment for this blogpost

```

Ionic:
  Ionic CLI           : 6.10.1
  Ionic Framework     : @ionic/angular 5.3.1
  @angular-devkit/build-angular : 0.901.12
  @angular-devkit/schematics   : 9.1.12
  @angular/cli        : 9.1.12
  @ionic/angular-toolkit : 2.3.0
```

```
Cordova:
Cordova CLI      : 9.0.0 (cordova-lib@9.0.1)

System:
Android SDK Tools : 26.1.1
NodeJS           : v12.14.0
npm              : 6.13.4
OS               : macOS Catalina
Xcode            : Xcode 11.5 Build version 11E608c
```

Creating a basic Ionic-Angular app. Start a basic `tabs` starter using

```
$ ionic start ionicTranslate tabs --type=angular --cordova
```

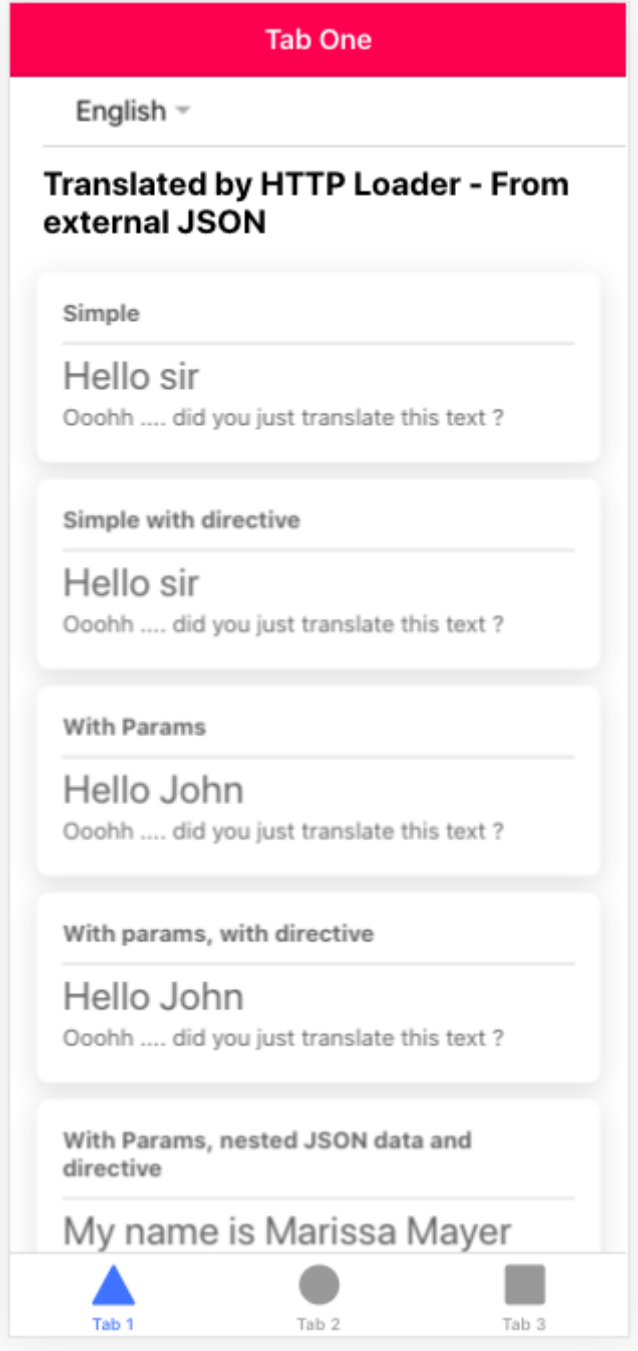
The `--type=angular` told the CLI to create an **Angular** app, and `--cordova` tells the CLI to integrate Cordova in the app.

Run the app in browser using

```
$ ionic serve
```

You won't see much in the homepage created in the starter. I have modified pages ( tab1 and tab2) to align it to our translation functionality.

My tab pages look like this



Tab UI for translation

HTML and SCSS file code for the above UI, if you want to just get started

```

1  <ion-header>
2    <ion-toolbar color="danger">
3      <ion-title>
4        Tab One
5      </ion-title>
6    </ion-toolbar>
7  </ion-header>
8
9  <ion-content>
10
11  <ion-item>
12    <ion-select [(ngModel)]="language" [value]="language" (ionChange)="changeLanguage()">
13      <ion-select-option value="en">English</ion-select-option>
14      <ion-select-option value="es">Spanish</ion-select-option>
15      <ion-select-option value="fr">French</ion-select-option>
16    </ion-select>
17  </ion-item>
18
19  <ion-grid>
20    <ion-row>
21      <ion-col>
22        Translated by HTTP Loader – From external JSON
23      </ion-col>
24    </ion-row>
25  </ion-grid>
26  <ion-card class="ion-padding">
27    <ion-card-header>
28      Simple
29    </ion-card-header>
30    <ion-card-content>
31      <h1>{{ title }}</h1>
32      <p>{{ description }}</p>
33    </ion-card-content>
34  </ion-card>
35  <ion-card class="ion-padding">
36    <ion-card-header>
37      Simple with directive
38    </ion-card-header>
39    <ion-card-content>
40      <h1 translate>TITLE</h1>
41      <p [translate]=" 'description' "></p>
42    </ion-card-content>
43  </ion-card>
44
45  <ion-card class="ion-padding">
46    <ion-card-header>
47      With Params
48    </ion-card-header>
49    <ion-card-content>
50      <h1>{{ title_2 }}</h1>
51      <p>{{ description }}</p>
52    </ion-card-content>
53  </ion-card>
54
55  <ion-card class="ion-padding">
56    <ion-card-header>
57      With params, with directive
58    </ion-card-header>
59    <ion-card-content>
60      <h1 [translate]=" 'TITLE_2' " [translateParams]="{value: 'John'}">TITLE</h1>
61      <p [translate]=" 'description' "></p>
62    </ion-card-content>
63  </ion-card>
64  <ion-card class="ion-padding">
65    <ion-card-header>

```

```
66      With Params, nested JSON data and directive
67      </ion-card-header>
68      <ion-card-content>
69        <h1>{{name}}</h1>
70        <h1 translate [translateParams]='{name_value: 'John'}">data.name</h1>
71      </ion-card-content>
72    </ion-card>
73
74  </ion-content>
```

```
1  .welcome-card ion-img {
2    max-height: 35vh;
3    overflow: hidden;
4  }
5  ion-card-header {
6    border-bottom: 1px solid #ccc;
7    padding: 0 0 10px 0;
8    font-weight: bold;
9  }
10 ion-col {
11   padding: 5px 15px;
12   font-size: 20px;
13   font-weight: bold;
14 }
15 ion-card-content {
16   padding: 5px 0 0 0;
17 }
18 ion-card {
19   margin-top: 10px;
20   margin-bottom: 10px;
21 }
```

## Step 2—Prepare multiple language JSON files in assets

We will create these JSON files in `src/assets/i18n` folder. The `assets` folder remains in the root even after a production build, so the path does not break. We will create three JSON files `en.json` (English), `fr.json` (French) and `es.json` (Spanish).

### Folder structure for i18n files

`en.json`

```
{
  "TITLE": "Hello sir",
  "TITLE_2": "Hello {{value}}",
  "description": "Ooohh .... did you just translate this text ?",
  "data": {
    "name": "My name is {{name_value}}"
  }
}
```

`fr.json`

```
{
  "TITLE": "Bonjour Monsieur",
  "TITLE_2": "Bonjour {{value}}",
  "description": "Ooohh .... vous venez de traduire ce texte?",
  "data" :{
    "name": "je m'appelle {{name_value}}"
  }
}
```

es.json

```
{
  "TITLE": "Hola señor",
  "TITLE_2": "Hola {{value}}",
  "description": "Ooohh .... acabas de traducir este texto?",
  "data": {
    "name": "Me llamo {{name_value}}"
  }
}
```

**Note**, the `{{value}}` and `{{name_value}}` are kind of variable/constants we can pass from our component. This can be used to

- Replace the variable with a user input or a value depending on the situation OR
- To give translations not supported by the library OR
- Keep a word constant across translations

### STEP 3: Implement ngx-translate library and Cordova Globalization plugin

Cordova globalization plugin is used to detect device’s default language/locale. Unfortunately, this plugin is deprecated, but it is still supported by Ionic. Hence, you can opt to use it. However, the latest way of detecting the language / locale of the browser is by using browsers’s [default Internationalization API](#).

Install Cordova globalization Plugin using

```
$ ionic cordova plugin add cordova-plugin-globalization
```

```
$ npm install @ionic-native/globalization
```

#### Install ngx-translate library

**ngx-translate** is the internationalization (i18n) library for Angular. Since our app has Angular under the hood, we can use this library for app as well as progressive web apps.

```
// Install core library
npm install --save @ngx-translate/core
```

```
// Install http loader
npm install @ngx-translate/http-loader --save
```

**http-loader** is used for loading the translation files (JSONs in our case) via Angular’s HttpClient module.

Note the versions of ngx-translate you should have as per your Angular version

Angular	@ngx-translate/core	@ngx-translate/http-loader
10	13.x+	6.x+
9	12.x+	5.x+
8	12.x+	4.x+

Use appropriate version of ngx-translate based on your Angular version

Setup the translation library and http-loader

We need to define a function that loads the external JSON files to the app using http-loader. Add the following function to `app.module.ts`

```
export function HttpLoaderFactory(http: HttpClient) {
  return new TranslateHttpLoader(http, "./assets/i18n/", ".json");
}
```

where we have provided the external path of our JSON files to the function.

We need to add the **translation** and **http-loader** modules in our root module `app.module.ts` . This is how the file looks after setup.



```

1  import { NgModule } from '@angular/core';
2  import { BrowserModule } from '@angular/platform-browser';
3  import { RouteReuseStrategy } from '@angular/router';
4  import { IonicModule, IonicRouteStrategy } from '@ionic/angular';
5  import { SplashScreen } from '@ionic-native/splash-screen/ngx';
6  import { StatusBar } from '@ionic-native/status-bar/ngx';
7  import { AppRoutingModule } from './app-routing.module';
8  import { AppComponent } from './app.component';
9  import { TranslateModule, TranslateLoader } from '@ngx-translate/core';
10 import { TranslateHttpLoader } from '@ngx-translate/http-loader';
11 import { HttpClientModule, HttpClient } from '@angular/common/http';
12 import { Globalization } from '@ionic-native/globalization/ngx';
13
14 export function HttpLoaderFactory(http: HttpClient) {
15   return new TranslateHttpLoader(http, "./assets/i18n/", ".json");
16 }
17
18 @NgModule({
19   declarations: [AppComponent],
20   entryComponents: [],
21   imports: [
22     HttpClientModule,
23     BrowserModule,
24     TranslateModule.forRoot({
25       loader: {
26         provide: TranslateLoader,
27         useFactory: HttpLoaderFactory,
28         deps: [HttpClient]
29       }
30     }),
31     IonicModule.forRoot(), AppRoutingModule],
32   providers: [
33     StatusBar,
34     SplashScreen,
35     Globalization,
36     { provide: RouteReuseStrategy, useClass: IonicRouteStrategy }
37   ],
38   bootstrap: [AppComponent]
39 })
40 export class AppModule { }

```

app.module.ts hosted with ❤ by GitHub

[view raw](#)

Pay attention to `TranslateModule.forRoot()` This is used in case of a Tabbed application, or general non lazy-loaded module. For a tab child, however, we will have to use `TranslateModule.forChild()` . We will see later how this will affect our functionality.

## Setup the translate library in child component

Let's say, we want to implement the translation in **Tab1**. As mentioned previously, `src/app/tab1` folder contains all the files for this page. We will import the `translationService` in `tab1.page.ts` file using

```
import { TranslateService } from '@ngx-translate/core';
```

The completed `tab1.page.ts` file will look like this

```

1  import { Component } from '@angular/core';
2  import { Globalization } from '@ionic-native/globalization/ngx';
3  import { TranslateService } from '@ngx-translate/core';
4
5  @Component({
6    selector: 'app-tab1',
7    templateUrl: 'tab1.page.html',
8    styleUrls: ['tab1.page.scss']
9  })
10 export class Tab1Page {
11   public title: string;
12   public title_2: string;
13   public description: string;
14   public name: string;
15   public language: string;
16   constructor(private globalization: Globalization, private _translate: TranslateService) {
17
18   }
19   ionViewDidEnter(): void {
20     this.getDeviceLanguage()
21   }
22   _initialiseTranslation(): void {
23     this._translate.get('TITLE').subscribe((res: string) => {
24       this.title = res;
25     });
26     this._translate.get('description').subscribe((res: string) => {
27       this.description = res;
28     });
29     this._translate.get('TITLE_2', { value: 'John' }).subscribe((res: string) => {
30       this.title_2 = res;
31     });
32     this._translate.get('data.name', { name_value: 'Marissa Mayer' }).subscribe((res: string) => {
33       this.name = res;
34     });
35   }
36
37   public changeLanguage(): void {
38     this._translateLanguage();
39   }
40
41   _translateLanguage(): void {
42     this._translate.use(this.language);
43     this._initialiseTranslation();
44   }
45
46   _initTranslate(language) {
47     // Set the default language for translation strings, and the current language.
48     this._translate.setDefaultLang('en');
49     if (language) {
50       this.language = language;
51     }
52     else {
53       // Set your language here
54       this.language = 'en';
55     }
56     this._translateLanguage();
57   }
58
59   getDeviceLanguage() {
60     if (window.Intl && typeof window.Intl === 'object') {
61       this._initTranslate(navigator.language)
62     }
63     else {
64       this.globalization.getPreferredLanguage()
65     }
66   }
67 }

```

```
66         .then(res => {
67             this._initTranslate(res.value)
68         })
69         .catch(e => {console.log(e);});
70     }
71 }
72 }
```

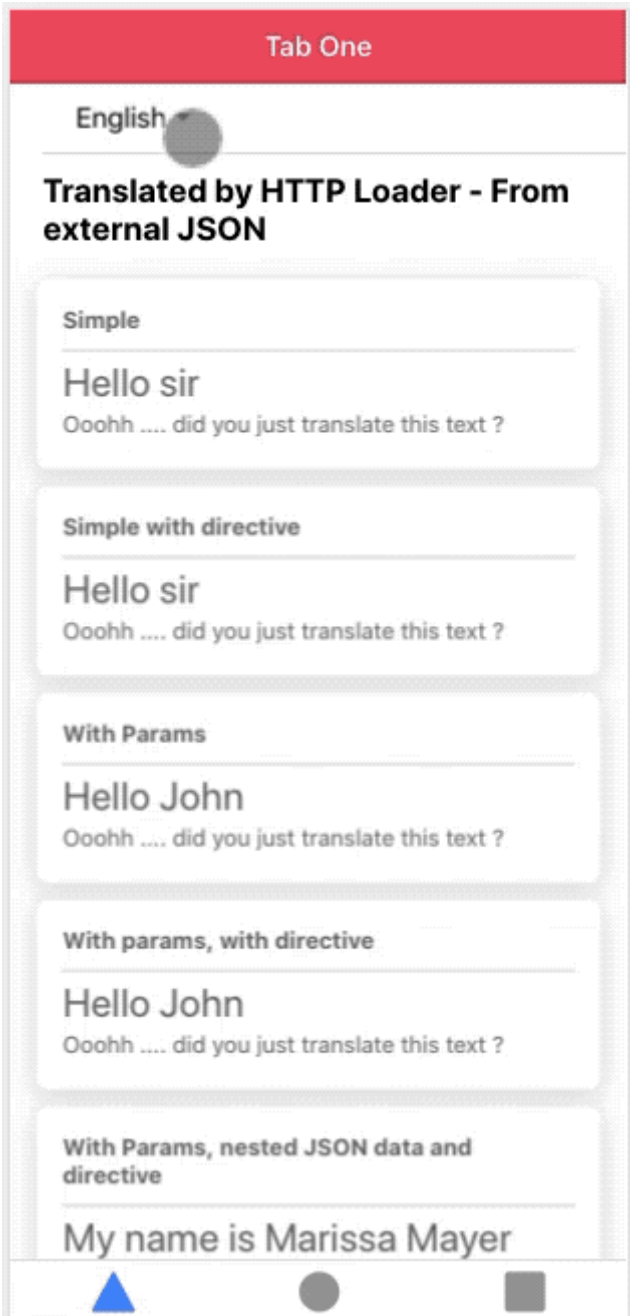
tab1.page.ts hosted with ❤️ by GitHub view raw

Let’s break down the code to understand a bit more

- On page load we check if we have default browser internationalization API by checking `window.Intl` . We then get default browser language using `navigator.language` . We also set a fall back on Cordova globalization plugin (deprecated) and set a default language if neither browser nor Cordova plugin works.
- Then we use `this._translate.use(this.language)` to tell the translation service which language to translate to. If the app has just loaded, it should default to your browser’s default language OR `en` (english)
- **Important:** Use `this._translate.get('TITLE').subscribe()` function from `translateService` to asynchronously fetch translations. Here,
  - `get()` is the function to fetch translations.
  - `TITLE` is the key to search for in the JSON files. If the data is in a nested JSON, then we use the dot notation to fetch e.g. `data.name`
  - `subscribe` is used for asynchronous fetching (no need to use timeouts)
- `changeLanguage` is called from user action. This function will use the previous two steps to translate to the intended language

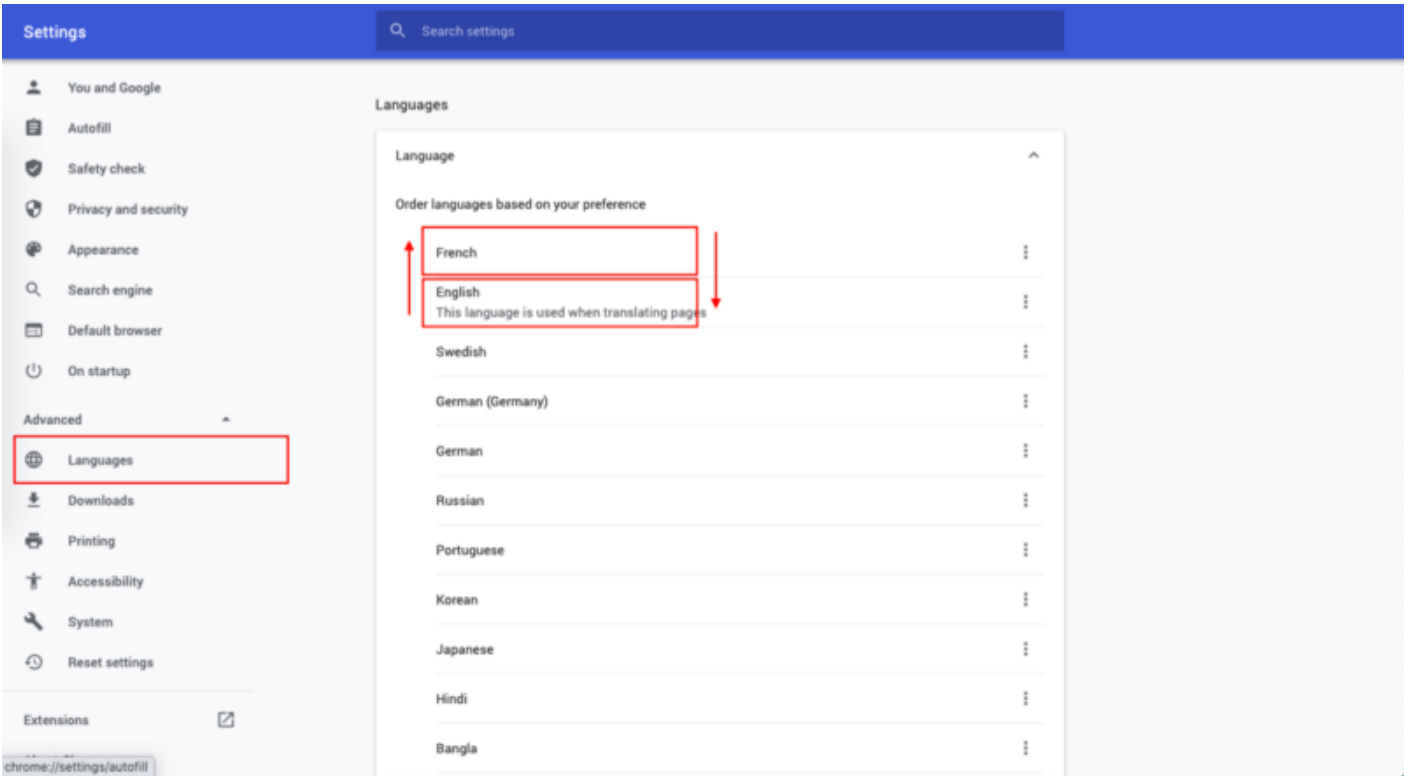
## Step 4— Test translations on browser

Run the app on browser using `ionic serve` and test the translations using the dropdown. Here’s a GIF to show the functionality on browser



Translate in Ionic 5 using ngx-translate

Notice that the text loads in English by default in my browser. You can change your browser language from browser settings and check if the initial text loads in your default language. For chrome, the settings can be found here



Change your browser language to load app text in default language

And now my text loads default in French .... Bonjour !!

### Step 5—The Directive GOTCHA 🕶️

If you have followed the above steps exactly, you might not get the exact result as shown in the above GIF.

You will realize that the **translation works** in places where you have used

```
this._translate.get('TITLE').subscribe((res: string) => {
  this.title = res;
});
this._translate.get('description').subscribe((res: string) => {
  this.description = res;
});
```

to get the translation, and shown it in the HTML using the local variable like this

```
<h1>{{ title }}</h1><p>{{ description }}</p>
```

BUT, the **translation does not work** in places where you have used a `directive` like either of the following

```
<h1 translate>TITLE</h1><p [translate]='description'></p>
```

This is because in our Ionic 5 tab app, the tab pages are lazy-loaded. In lazy-loaded modules, you need to import the translation module in child modules as well for everything to work correctly.

Let's go to our `tab1.module.ts` file, and import the translation and http-loader modules similar to our root module. But this time, we will use `TranslateModule.forChild`. The complete module file looks like the following

```
1  import { IonicModule } from '@ionic/angular';
2  import { NgModule } from '@angular/core';
3  import { CommonModule } from '@angular/common';
4  import { FormsModule } from '@angular/forms';
5  import { Tab1Page } from './tab1.page';
6  import { ExploreContainerComponentModule } from '../explore-container/explore-container.module';
7
8  import { Tab1PageRoutingModule } from './tab1-routing.module';
9  import { TranslateModule, TranslateLoader } from '@ngx-translate/core';
10 import { HttpClient } from '@angular/common/http';
11 import { TranslateHttpLoader } from '@ngx-translate/http-loader';
12
13 export function HttpLoaderFactory(http: HttpClient) {
14   return new TranslateHttpLoader(http, './assets/i18n/', '.json');
15 }
16
17 @NgModule({
18   imports: [
19     IonicModule,
20     TranslateModule.forChild({
21       loader: {
22         provide: TranslateLoader,
23         useFactory: HttpLoaderFactory,
24         deps: [HttpClient]
25       }
26     }),
27     CommonModule,
28     FormsModule,
29     ExploreContainerComponentModule,
30     Tab1PageRoutingModule
31   ],
32   declarations: [Tab1Page]
33 })
34 export class Tab1PageModule { }
```

tab1.module.ts hosted with ❤️ by GitHub [view raw](#)

Now, if you run the app again, the `directive` translations will also work fine. 😎😎😎

`directive` method is preferred for bigger apps with lots of code, since this method results in smaller code size and no need of local variables.

## Step 6— Setup stand alone translations

The process of setting up separate language file in `assets` for each language is the standard way of translation in Angular. But sometimes it becomes a little cumbersome, especially when you don't have that much data to translate.

In case you want to quickly translate in the component itself, so that there is no spill over on other components, you can declare the variables in the components itself instead of reading them from the JSON files from `assets`

Let's do these changes in `tab2`, so it doesn't affect the global translations in `tab1`

### HTML and SCSS

Similar to `tab1.page.html` , just remove the usage of variable `data` from the HTML. You can keep the styling same

### tab2.page.ts

Stays pretty much same as `tab1.page.ts` . Just add the following in the constructor

```
_translate.setTranslation('en', {
  "TITLE": "Bye sir",
  "TITLE_2": "Bye {{value}}",
  "description": "Thanks for translating this text"
});

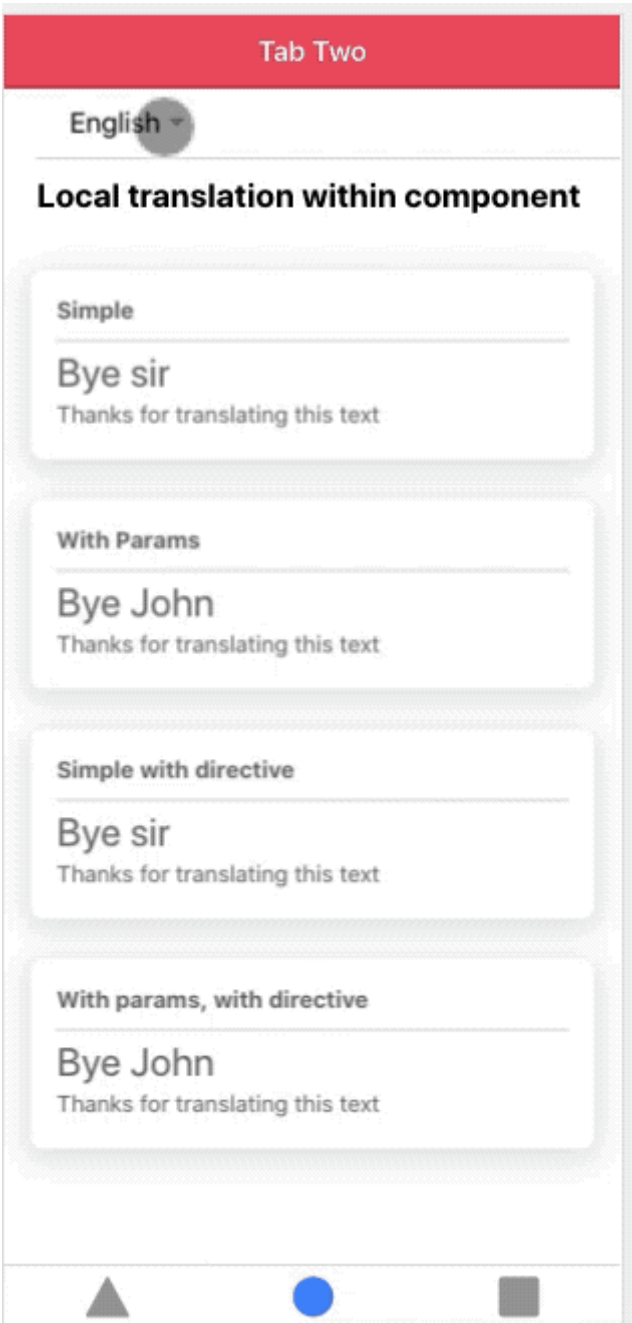
_translate.setTranslation('fr', {
  "TITLE": "Au revoir Monsieur",
  "TITLE_2": "Au revoir {{value}}",
  "description": "Merci d'avoir traduit ce texte"
});

_translate.setTranslation('es', {
  "TITLE": "Adiós señor",
  "TITLE_2": "Adiós {{value}}",
  "description": "Gracias por traducir este texto"
});
```

Here, you are defining the translations locally in your component itself. Also, to let angular know that these are standalone translations, you use `isolate:true` in your `tab2.module.ts`

```
....
TranslateModule.forChild({
  loader: {
    provide: TranslateLoader,
    useFactory: HttpLoaderFactory,
    deps: [HttpClient]
  },
  isolate: true
}),
....
```

Now run the app in browser and test translations. Your translations will be picked from the local component itself



Translation in Ionic 5 apps using component level translations

Notice, the `directive` method and `variable` also work well with the local definitions.

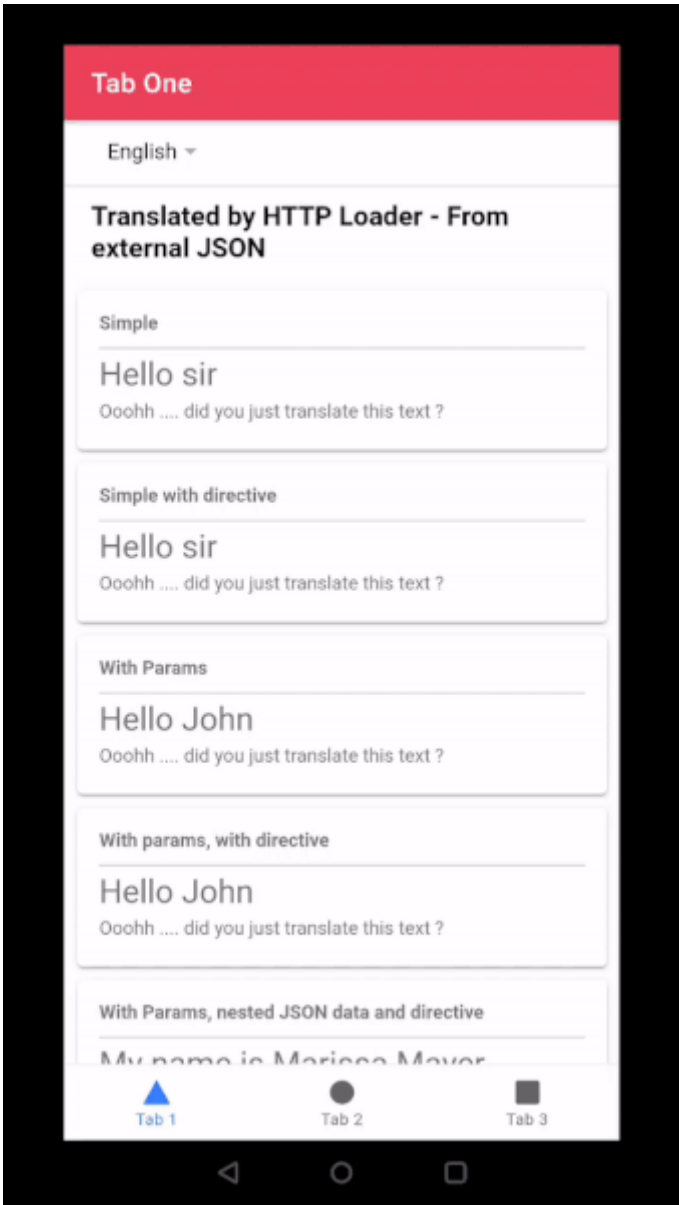
## Step 7—Test translations in iOS/Android

To build the app on android, run the following commands one after another

```
$ ionic cordova platform add android

$ ionic cordova run android
```

The final command will run the app on either default emulator, or an android device attached to your system. Here’s the translation running on my android device



Ionic 5 Angular app—Translation and Globalization

- Interesting fact:** In device, the language id may be different from browser. For me
- Device default language comes out to be `en-US`
  - Browser’s default language comes out to be `en`

Hence, you need to be careful to detect all variations of a language. Accordingly, you’ll need to have JSON file named accordingly.

## Conclusion

In this post we learnt how to detect device / browser language, create multiple language files, and implement translation using different methods in Ionic 5 apps.

The only limitation of using the `ngx-translate` library is that you will need to define your language texts on each page of your application beforehand. These will be stored as country code titled JSON files (i.e. `en.json`, `jp.json`, `it.json` etc) within the `src/assets/i18n/` directory. You can’t dynamically generate the language translation on the



fly using this library so if you require that type of functionality you'll need to look into using the Google Translate API or something similar.

## Next Steps

If you liked this blog, you will also find the following Ionic blogs interesting and helpful. Feel free to ask any questions in the comment section

- Ionic Payment Gateways—[Stripe](#) | [PayPal](#) | [Apple Pay](#) | [RazorPay](#)
- Ionic Charts with—[Google Charts](#) | [HighCharts](#) | [d3.js](#) | [Chart.js](#)
- Ionic Social Logins—[Facebook](#) | [Google](#) | [Twitter](#)
- Ionic Authentications—[Via Email](#) | [Anonymous](#)
- Ionic Features—[Geolocation](#) | [QR Code reader](#) | [Pedometer](#)
- Media in Ionic—[Audio](#) | [Video](#) | [Image Picker](#) | [Image Cropper](#)
- Ionic Essentials—[Native Storage](#) | [Translations](#) | [RTL](#)
- Ionic messaging—[Firebase Push](#) | [Reading SMS](#)
- Ionic with Firebase—[Basics](#) | [Hosting and DB](#) | [Cloud functions](#)

. . .

## Ionic React Full App with Capacitor

If you need a base to start your next Ionic 5 React Capacitor app, you can make your next awesome app using [Ionic 5 React Full App in Capacitor](#)



Ionic 5 React Full App in Capacitor from Enappd

## Ionic Capacitor Full App (Angular)

If you need a base to start your next Angular **Capacitor app**, you can make your next awesome app using [Capacitor Full App](#)

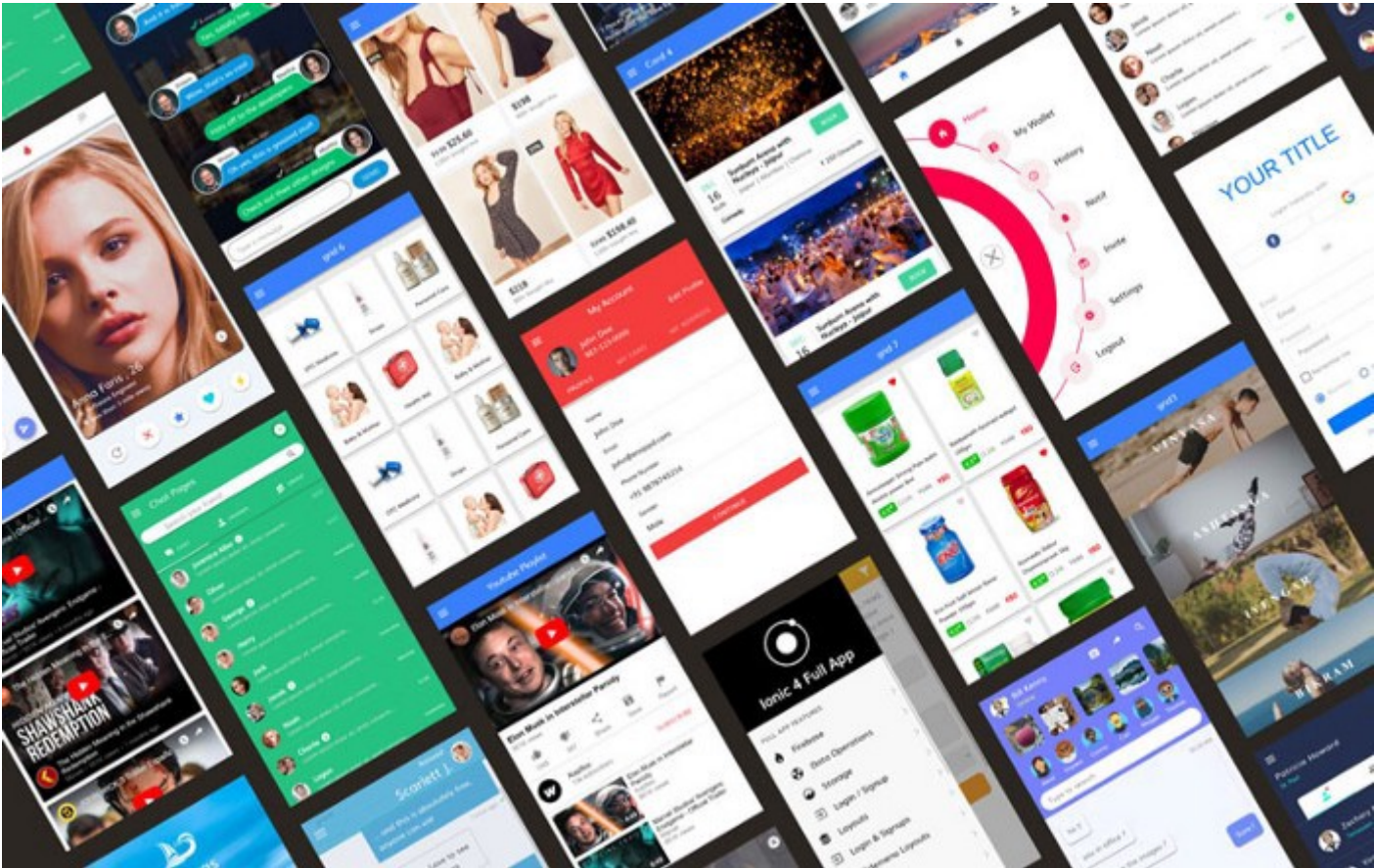




Capacitor Full App with huge number of layouts and features

**Ionic Full App (Angular and Cordova)**

If you need a base to start your next Ionic 5 app, you can make your next awesome app using [Ionic 5 Full App](#)



Ionic Full App in Cordova, with huge number of layouts and features



CONNECT WITH US



Browse Templates and Starters

[Ionic 4 Starters](#) [React Native Starters](#) [Firebase Starters](#) [Free Starters](#) [Flutter Starters](#) [Full App Starters](#)

Best Sellers

[Ionic 4 Full App](#) [Ionic 4 Taxi Booking Complete Platform](#) [Ionic 4 Grocery Shopping Complete Platform](#) [Ionic 4 Spotify Clone](#)

Other Links

[Become Enappd Affiliate](#) [Privacy Policy](#) [Terms and Conditions](#) [Licensing Options](#)

ALSO ON ENAPPD

React-Native QR Code Scanning using ...

10 months ago • 1 comment

This post is all about implementing the QR Code Scanner in your cool new ...

Implement In-app purchase in Ionic 4 ...

a year ago • 3 comments

Learn how to implement In-App Purchase functionality in iOS apps using Ionic 4. ...

Stripe payment integration in React ...

a year ago • 11 comments

Learn how to integrate Stripe Payment Gateway in React Native apps. ...

Ionic 4 Car Pooling App Starter

a year ago • 1 comment

In this article, we get to know in quick fashion about all the screens and ...

Google Vision With React Native

a year ago • 1 comment

In this tutorial we got to know, how we can implement react-native ...

0 Comments

Enappd

🔒

Disqus' Privacy Policy

📬

1 Login

📖

Recommend


🐦

Tweet

📱

Share

Sort by Best



Start the discussion...

LOG IN WITH

OR SIGN UP WITH DISQUS 

?

Name

Be the first to comment.

✉

Subscribe

ⓓ

Add Disqus to your siteAdd DisqusAdd

⚠

Do Not Sell My Data

Sponsored

Conoce solteras cerca de Mislata

Meetic

Si no invierte en Inditex ahora, se arrepentirá después

Invertir en Inditex

This Game Can Train Your Brain To Think Strategically

Total Battle: Tactical War Game

¡Conseguir este tesoro es imposible! Demuestra que nos equivocamos

Hero Wars

Vende tu casa en tiempo récord por 1.995€

Housell

Invertir no es sólo para millonarios. Acciones de Amazon por sólo €250

Invierte en Amazon