

i18n Ionic 1.6 Documentation

Quick start Guide - Installation and user support

Created: July 20, 2015

Updated: May 05, 2016

By: Stavros Kounis, about.me/stavros.kounis

Thank you for purchasing my product. If you have any question that are beyond of the scope of this help file, please feel free to email via [my user page](#) contact form. Thank you!

Table of contents:

[Prerequisites](#)

[PhoneGap and Ionic](#)

[Tools](#)

[Run for the first time](#)

[Download and extract](#)

[Install libraries](#)

[Post installation](#)

[Windows Users](#)

[Plugins](#)

[Run a local development server](#)

[Run in the emulator](#)

[How Internationalisation works](#)

[Introduction](#)

[Configuration](#)

[Configuring a new language](#)

[Language files and Translations](#)

[Language tags / codes](#)

[Naming language codes](#)

[Alternative locale creation](#)

[A working example](#)

[Home](#)

[Plain Text](#)

[Date/Time](#)

[Controls](#)

[Default language](#)

[Available languages](#)

[Add a new language/locale](#)

[Support](#)

[References / Links:](#)

[Thank you](#)

Prerequisites

PhoneGap and Ionic

This is a [PhoneGap](#) and [Ionic](#) based application, so the [PhoneGap](#) and [Ionic](#) should be installed in your computer. Since **i18n Ionic**, targets iPhone and Android mobile devices, your environment should be properly configured and the corresponding SDK should be installed. If not you will be still able to run the application into a Browser.

Please check the “[Install PhoneGap](#)” and “[Getting Started with Ionic](#)” sections in the official PhoneGap and Ionic sites respectively.

Ensure first that [NodeJS](#) is installed in your computer

Tools

This project is based on the popular “[Ionic Framework Generator](#)” that boosts the overall development process by integrating a couple of very popular automation tools like [Grunt](#) and [Bower](#).

Install these tools by following the instructions in their corresponding web pages:

1. [Install Bower](#)
2. [Getting started with Grunt - Install the CLI](#)
3. [Getting started with Yeoman](#)

Finally install the yeoman generator via:

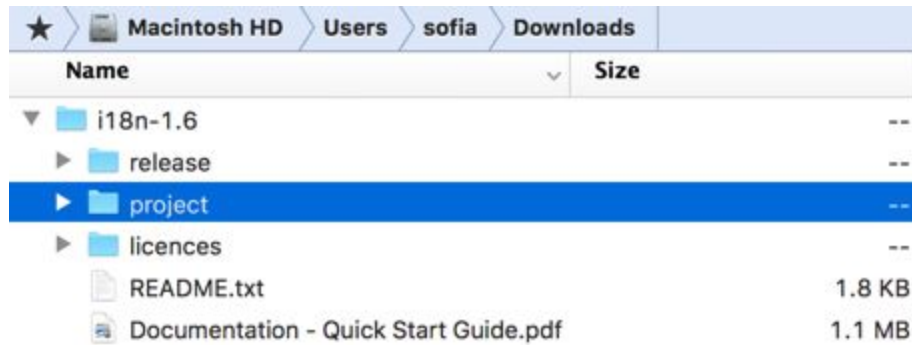
```
$ npm install -g generator-ionic
```

Run for the first time

In the screen captures that follow, we will demonstrate the process of preparing your environment and running the project for the first time.

Download and extract

Download the provided .zip file and extract it, you will see something similar to what is shown in the next screen:



The highlighted folder is used for all the instructions in this document.

Install libraries

Open a terminal window and navigate to the directory where the “i18n” folder is located.

Install NodeJS dependences:

```
$ npm install
```

Post installation

There is a post installation process under which required Cordova plugins and Javascript dependencies are installed. To simplify this process two scripts are already prepared for both platforms: Linux/MacOS and Windows

Linux/MacOX

Install all the required plugins and Javascript dependencies:

```
$ ./install.sh
```

Windows Users

Similarly, Windows users should execute:

```
$ install.bat
```

Plugins

Build your project for the first time. This will create the [www] folder which is the actual cordova directory and where the plugins will be installed.

```
$ grunt build
```

Follow the same process as with “Libraries” and install the required plug ins by using the command that follows:

```
$ cordova plugin add {plugin name or url}
```

eg:

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

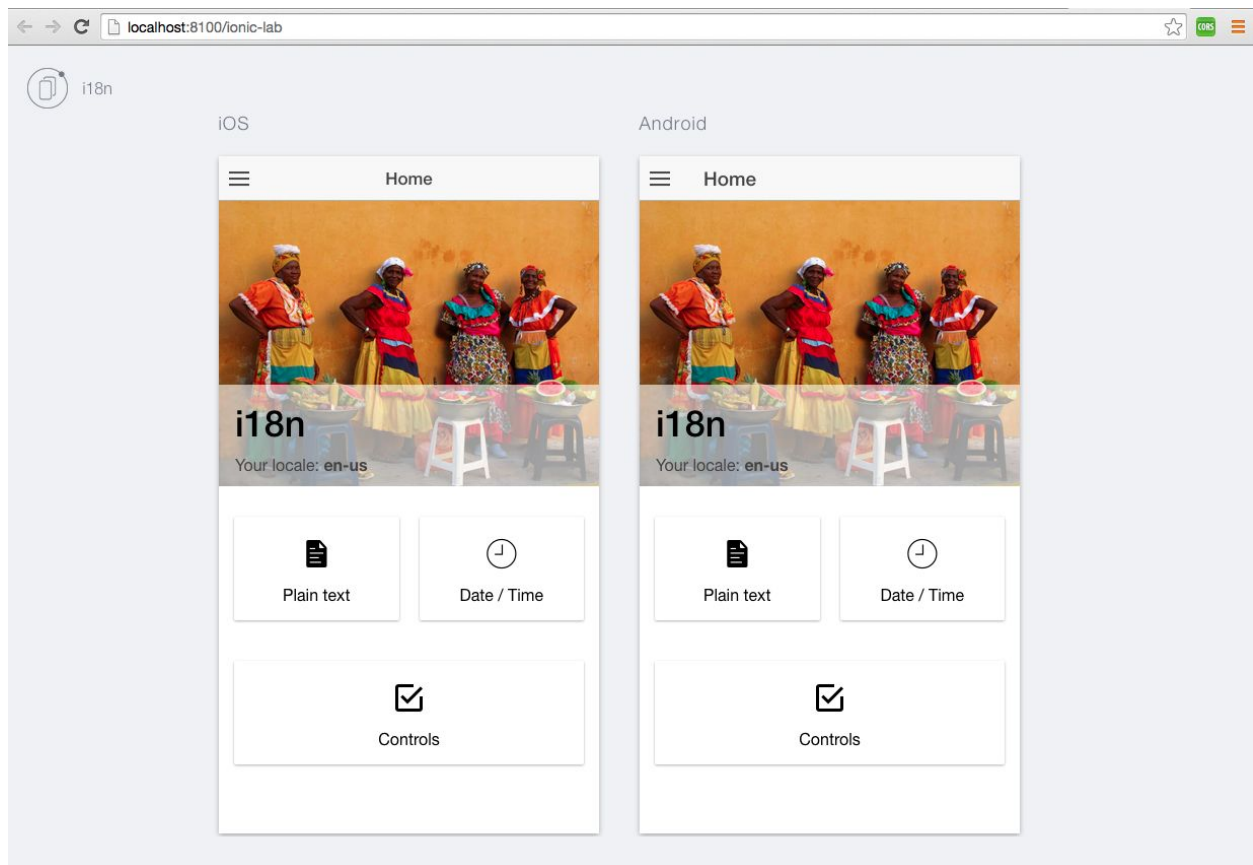
Please check the `Readme.md` file in the project folder for the list with all the required plugins the application is based on and should be installed.

Run a local development server

Start the local NodeJS server and run the application in the browser:

```
$ grunt serve --lab
```

A browser window will open with two virtual devices the one next to the other.



You could also open the application in a single browser window by starting it with the command:

```
$ grunt serve
```

Run in the emulator

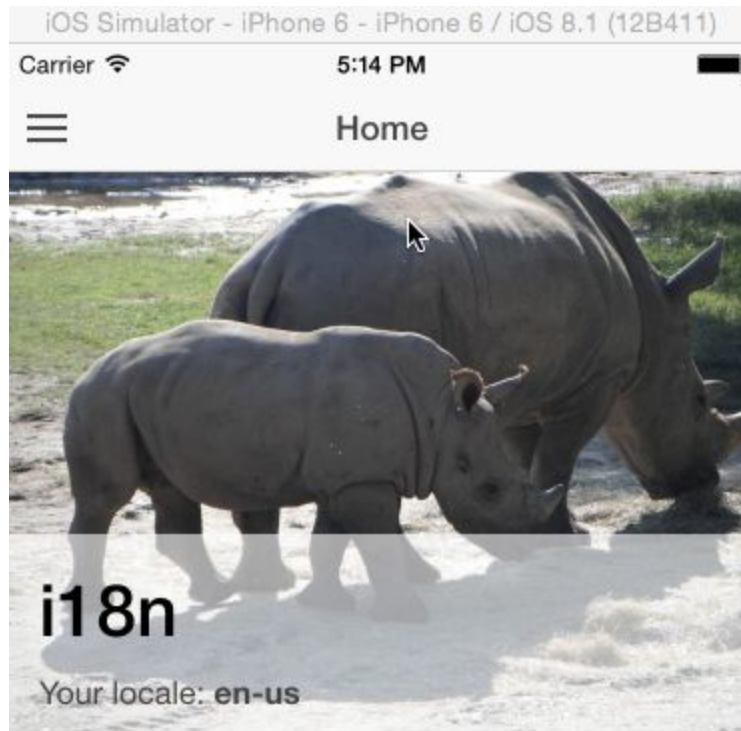
First the preferred platform should be added. In this case iOS:

```
$ grunt platform:add:ios
```

Now the application is ready to start inside a simulator:

```
grunt emulate:ios --livereload
```

The iPhone simulator will launch and the i18n app will start.



Plain text



Date / Time



Controls

How Internationalisation works

Introduction

In many cases, there is the need to localise your application according to a language or culture

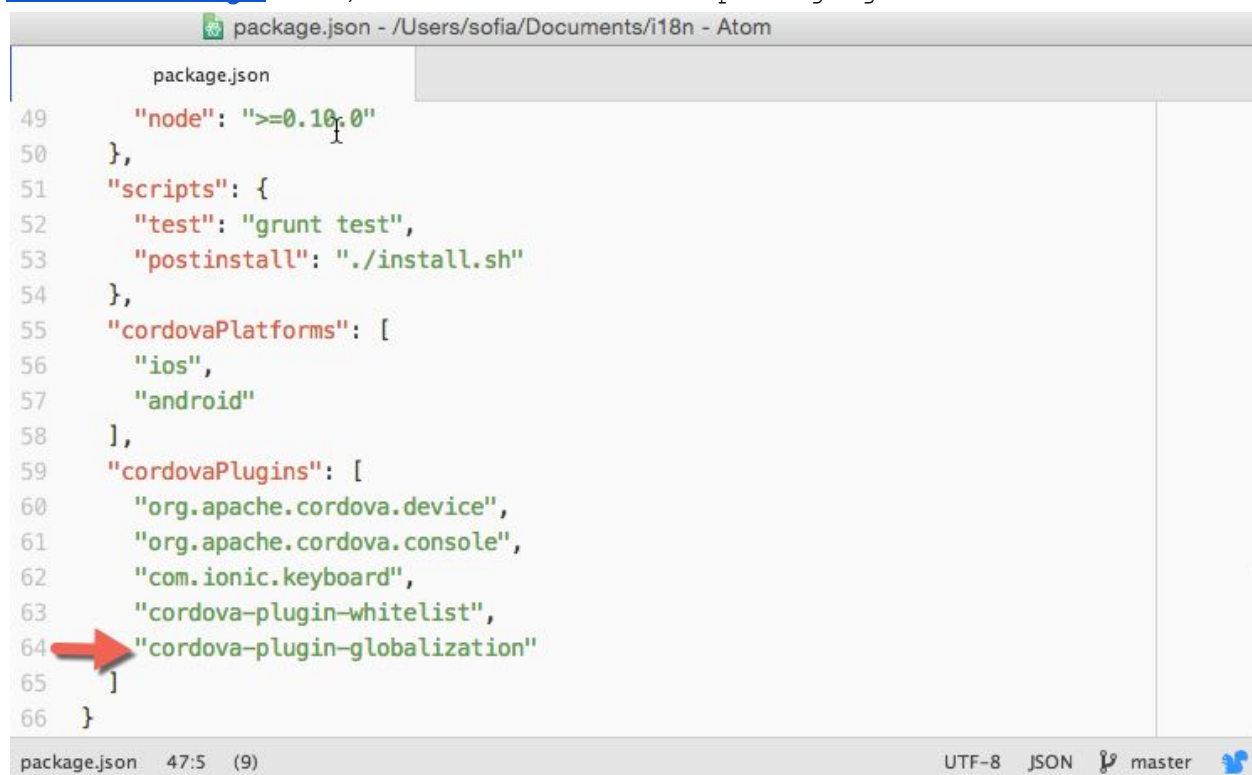
which is called Internationalisation (i18n). This can be done by abstracting all the strings and other locale-specific bits, such as date, out of the application which is called localization (l10n). You can localise your application by including the corresponding locale JS file. For more information, visit <https://docs.angularjs.org/guide/i18n>.

To set the locale programmatically [Angular Dynamic Locale](#) is used. For this, you need to have available the corresponding `angular-locale_{{locale}}.js` (eg `angular-locale_en-us.js`), for Angular Dynamic Locale to use it without the need to follow the OS locale settings.

As for translation, [Angular Translate](#) module is used which utilizes the keys representing the related translation ids.

Configuration

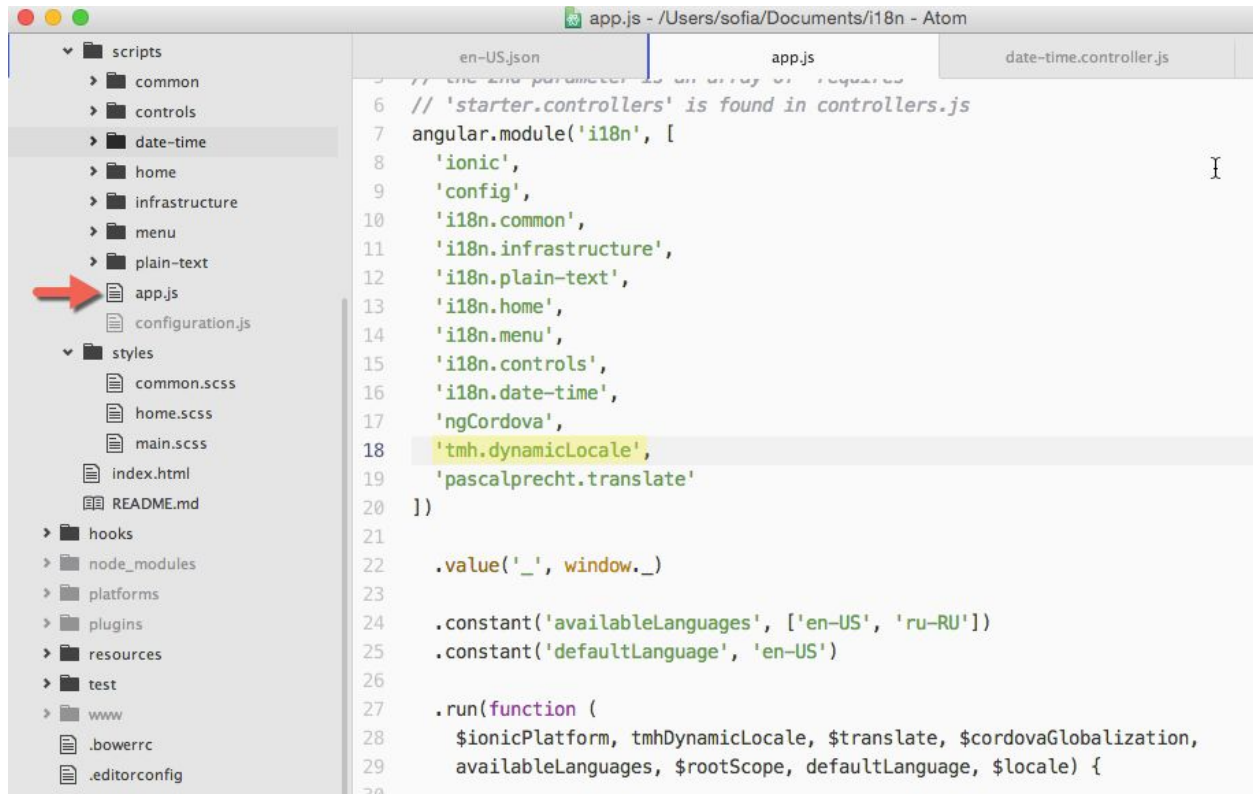
In order to fetch the corresponding locale configuration, you need to add the [Cordova Globalization Plugin](#). Thus, it should be included in the `package.json` file as shown below:



```
package.json - /Users/sofia/Documents/i18n - Atom
package.json
49   "node": ">=0.10.0"
50 },
51 "scripts": {
52   "test": "grunt test",
53   "postinstall": "./install.sh"
54 },
55 "cordovaPlatforms": [
56   "ios",
57   "android"
58 ],
59 "cordovaPlugins": [
60   "org.apache.cordova.device",
61   "org.apache.cordova.console",
62   "com.ionic.keyboard",
63   "cordova-plugin-whitelist",
64   "cordova-plugin-globalization"
65 ]
66 }
```

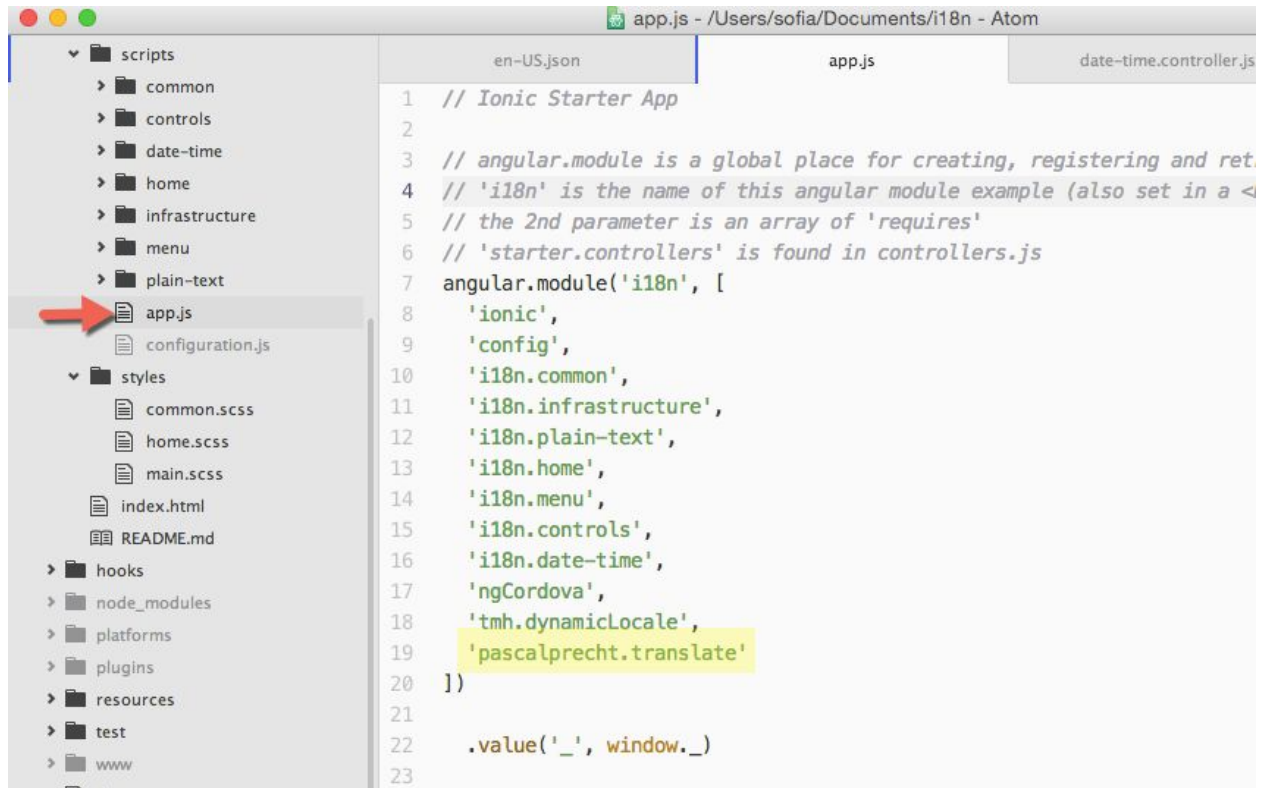
package.json

Also, the dependency for Angular Dynamic Locale needs to be injected by adding the highlighted area in `app.js` file under the `app/scripts` path as shown below:



app.js

Additionally, in order to inject Angular Translate you need to add it in the `app.js` file under the `app/scripts` path as shown below:



app.js

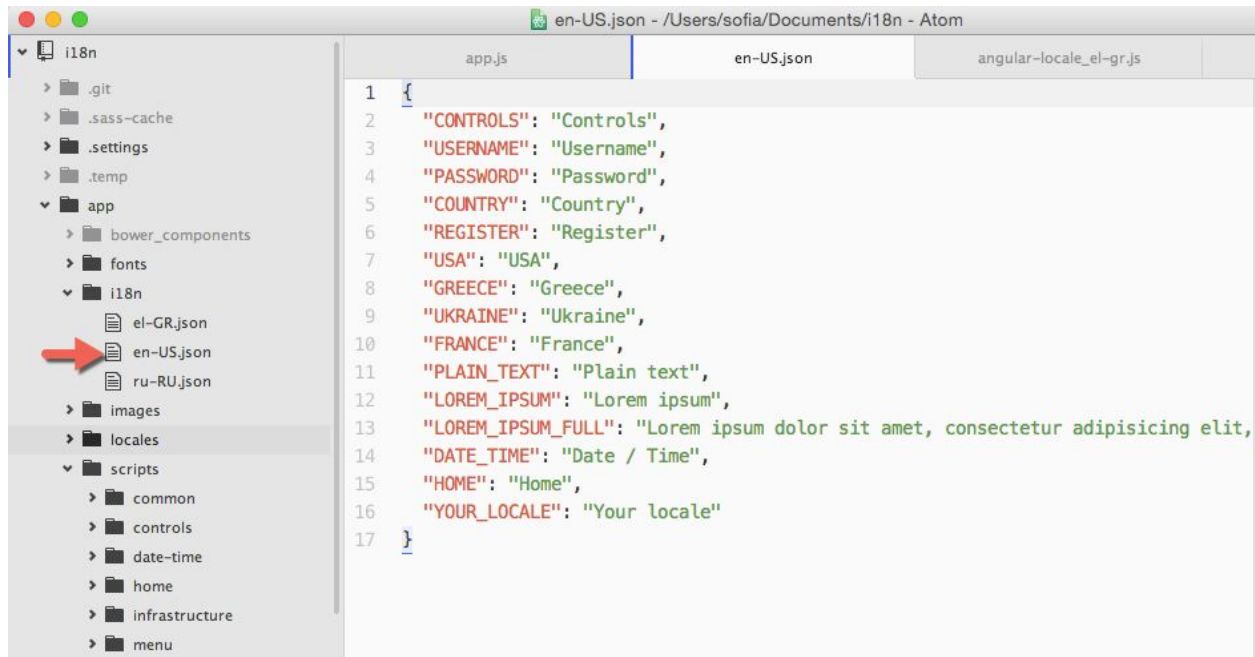
Configuring a new language

Two folders are responsible for the Internationalisation of this application: `i18n` and `locales`. The former is used for the translation of plain text and the latter for the localisation of the application in the specific region's date/time settings.

For more details on how to add a new language/region settings, please see section "Add a new language/locale".

Language files and Translations

For the translation files, you need to create a JSON file under the `app/i18n` path consisting of the keys representing the related translation ids which looks similar to the following:

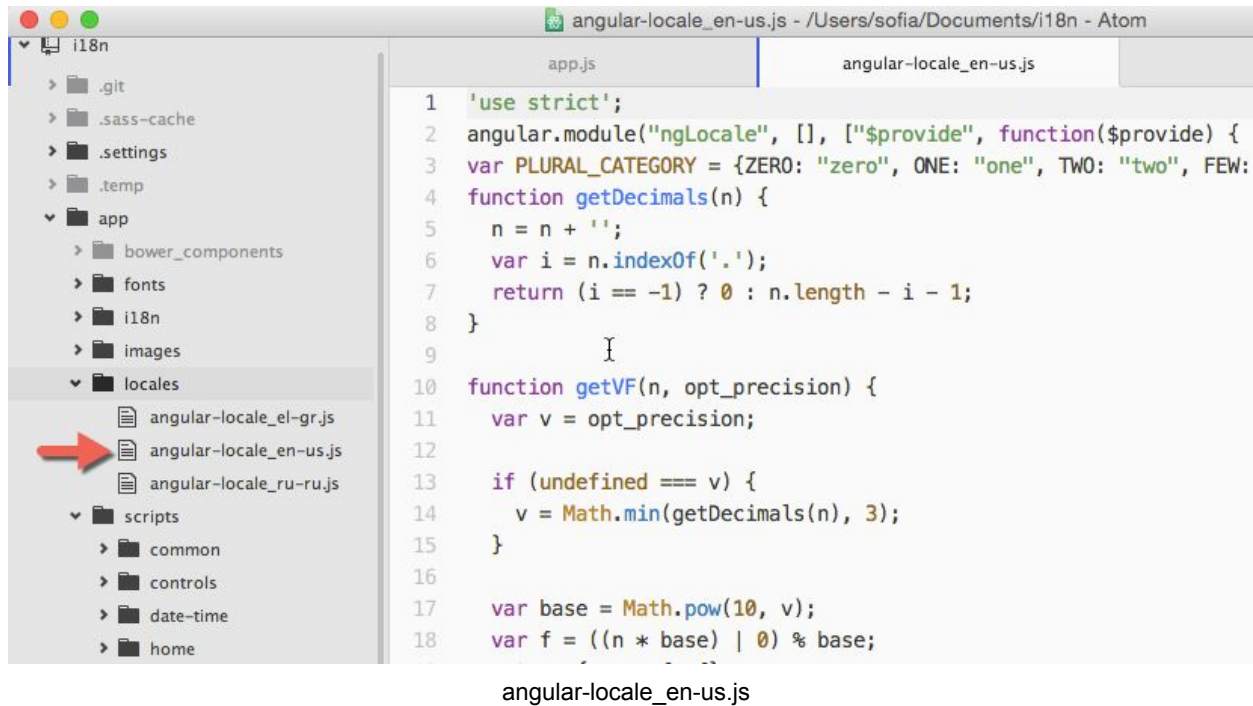


en-US.json

As shown above, each JSON file corresponds to a specific language and contains all the translation keys (red colored) followed by the respective text in a particular language (green colored).

As for the files which are used for the date/time locale, i18n expects the angular locales to be present at `/i18n/angular-locale_{{locale}}.js`. This file is available for many different languages and can be found in the [Angular's repo](#). Angular's repo contains all the currently supported locales, so you could download the files you need from there.

The `/i18n/angular-locale_{{locale}}.js` file should be contained under the `app/locales` path:



Language tags / codes

Language tags/codes are used in order to identify the language of a text or the language/culture code of the application.

Naming language codes

Some of the language codes that are used in **i18n Ionic** application are “en-US”, “el-GR”, “ru-RU”.

The best practices regarding the codes-subcodes creation are illustrated in [W3C's Internalization guidelines](#). Note that, in W3C's Internalization guidelines, codes are referred to as tags. Following the standard W3C recommends, the language codes are the same for all programming languages and they suggest a common interpretation by all professionals as most locale-aware applications conform to this standard.

Alternative locale creation

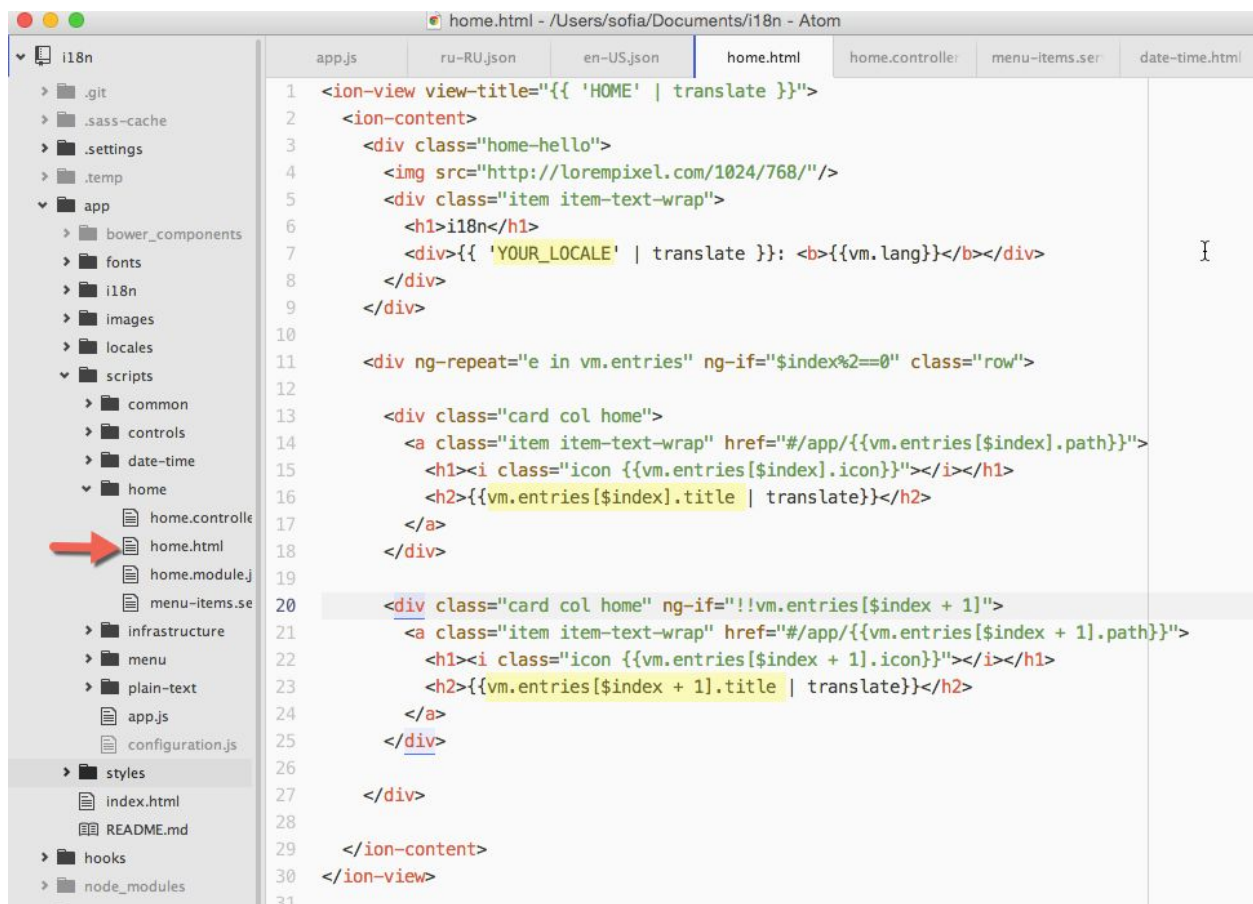
In case of using different locales from those existing in the [Angular's repo](#), a unicode converter, such as [Brenah's Unicode Converter](#), can be utilised for the representation of non latin characters of language tags.

A working example

In this section, an example, that demonstrates all the aforementioned principles followed by the **i18n Ionic** application, is presented.

Home

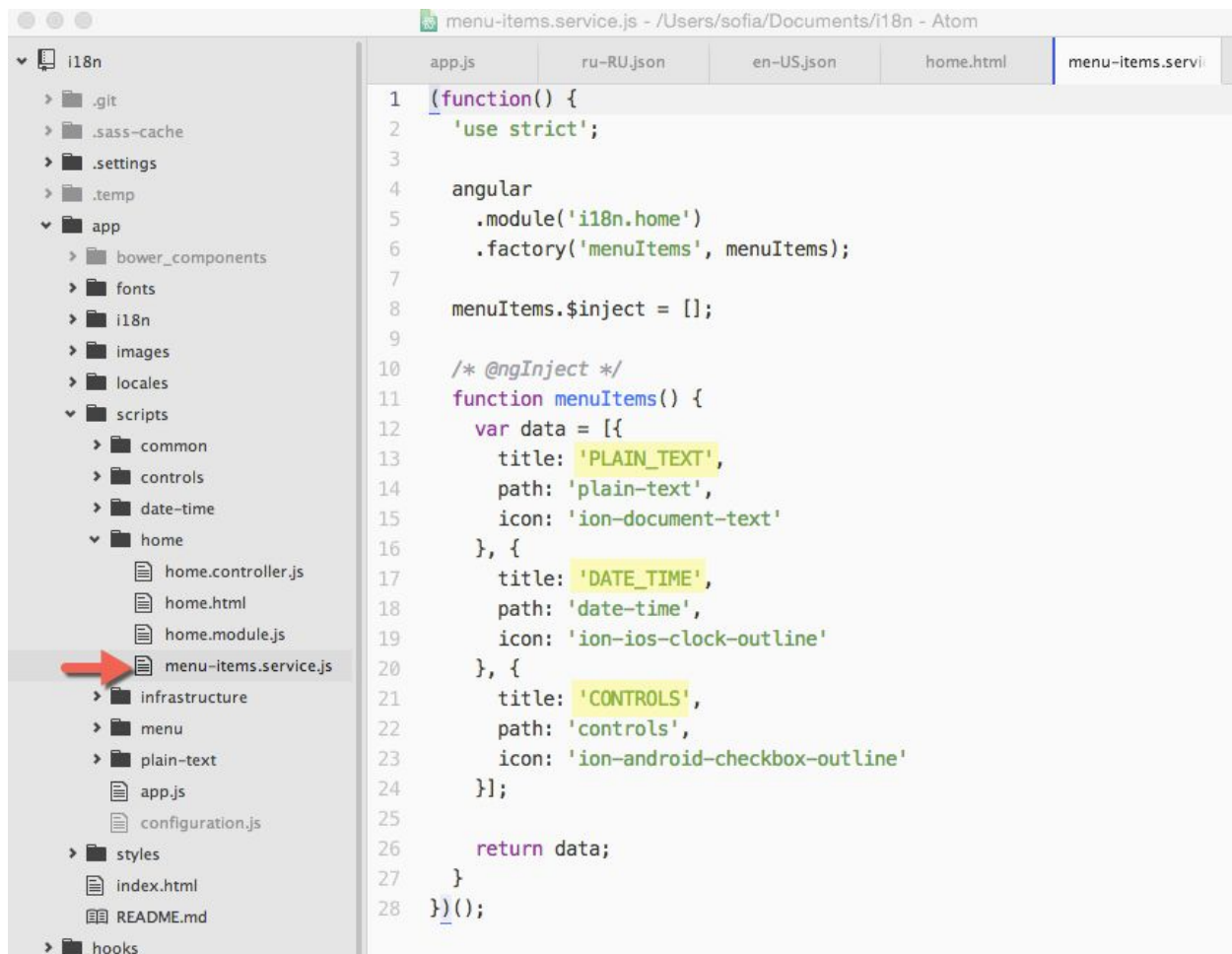
`home.html` uses translation keys instead of string values which are defined in translation files under the `app/i18n` path. A JSON file example for US-english is already shown above ([Language Files and Translations](#) section). The translation keys used on the “Home” screen are found in `home.html` file under the `app/scripts/home` path as highlighted below:



```
1 <ion-view view-title="{{ 'HOME' | translate }}">
2   <ion-content>
3     <div class="home-hello">
4       
5       <div class="item item-text-wrap">
6         <h1>i18n</h1>
7         <div>{{ 'YOUR_LOCALE' | translate }}: <b>{{vm.lang}}</b></div>
8       </div>
9     </div>
10
11     <div ng-repeat="e in vm.entries" ng-if="$index%2==0" class="row">
12
13       <div class="card col home">
14         <a class="item item-text-wrap" href="#/app/{{vm.entries[$index].path}}">
15           <h1><i class="icon {{vm.entries[$index].icon}}"></i></h1>
16           <h2>{{vm.entries[$index].title | translate}}</h2>
17         </a>
18       </div>
19
20       <div class="card col home" ng-if="!!vm.entries[$index + 1]">
21         <a class="item item-text-wrap" href="#/app/{{vm.entries[$index + 1].path}}">
22           <h1><i class="icon {{vm.entries[$index + 1].icon}}"></i></h1>
23           <h2>{{vm.entries[$index + 1].title | translate}}</h2>
24         </a>
25       </div>
26     </div>
27   </ion-content>
28 </ion-view>
```

home.html

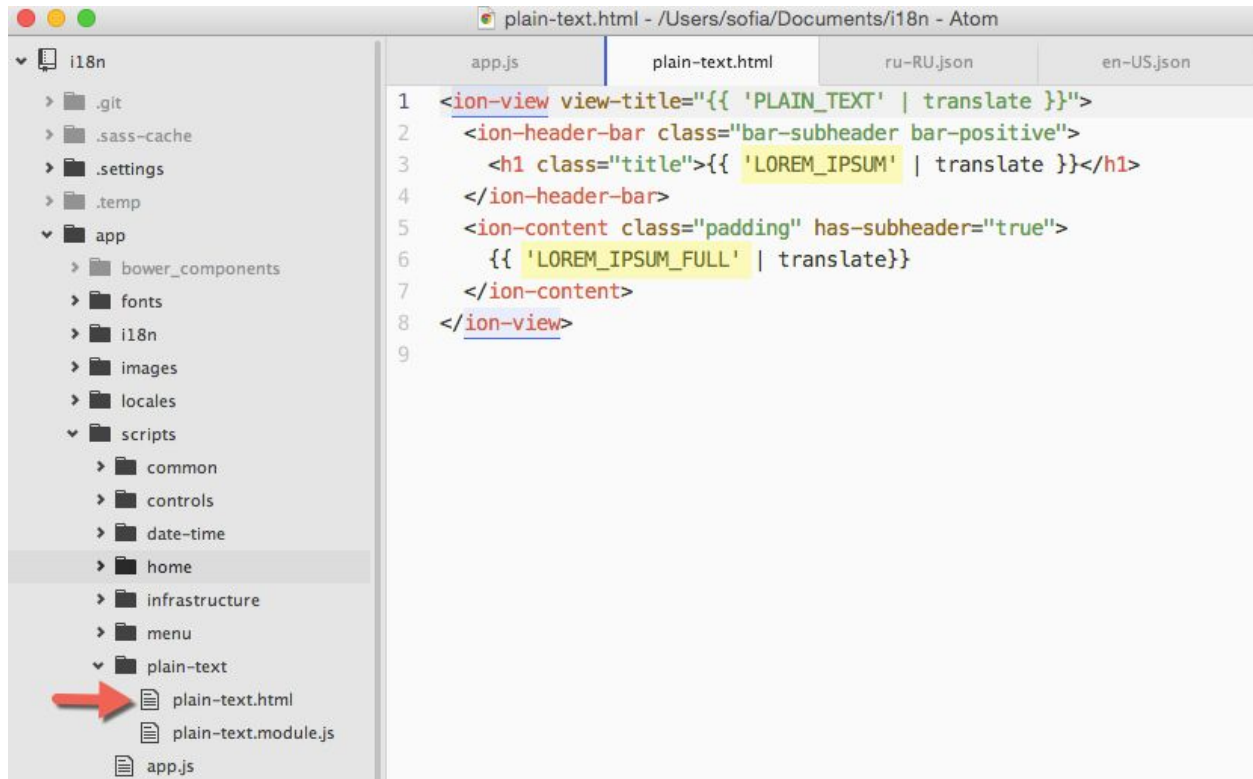
It should be noted the fact that the keys of the buttons which are displayed on the “Home” screen are not set directly as the first highlighted key but they are set in `menu-items.service.js` under the `app/scripts/home` path as shown here:



menu-items.service.js

Plain Text

Similarly, in case that you need to translate a free text, as is the case presented when clicking the “Plain Text” button, you need to utilise a key for the title and one for the text body in the `plain-text.html` file under the `app/scripts/plain-text` path:



plain-text.html

Date/Time

Date/Time formatting is undertaken by Angular as the `/i18n/angular-locale_{{locale}}.js` file dictates. For example, this JS file for US-english is `angular-locale_en-us.js` under the `app/locales` path as shown in [Language Files and Translations](#) section.

Controls

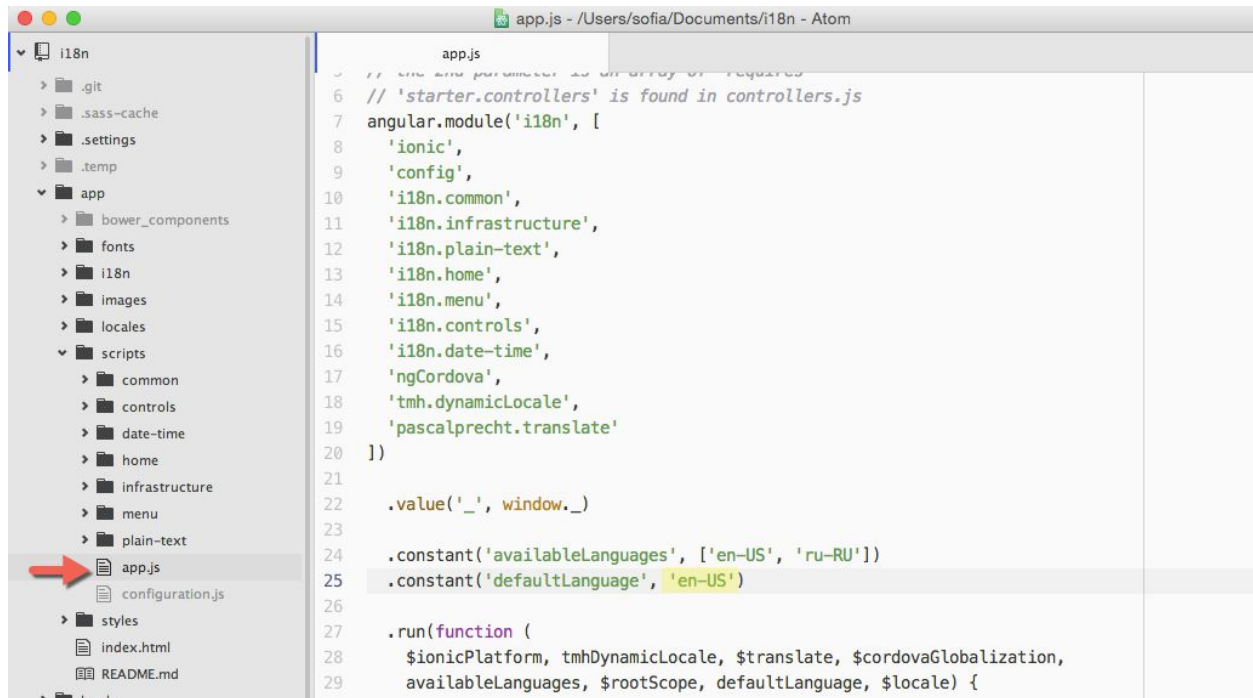
If translating a form is the case, then this scenario is demonstrated when clicking on the “Controls” button. First, in the `controls.html` file under the `app/scripts/controls` path, you need to use a key for every form field, every option of the drop-down list and the submit button as highlighted below:


```
1 <ion-view view-title="{{ 'CONTROLS' | translate }}">
2   <div class="content has-header">
3     <div class="list">
4       <label class="item item-input">
5         <span class="input-label">{{ 'USERNAME' | translate }}</span>
6         <input type="text">
7       </label>
8       <label class="item item-input">
9         <span class="input-label">{{ 'PASSWORD' | translate }}</span>
10        <input type="password">
11      </label>
12      <div class="item item-input item-select">
13        <div class="input-label">{{ 'COUNTRY' | translate }}</div>
14        <select>
15          <option>{{ 'USA' | translate }}</option>
16          <option>{{ 'GREECE' | translate }}</option>
17          <option>{{ 'UKRAINE' | translate }}</option>
18          <option selected="">{{ 'FRANCE' | translate }}</option>
19        </select>
20      </div>
21    </div>
22    <div class="padding">
23      <button class="button button-block button-positive">{{ 'REGISTER' | translate }}
24    </div>
25  </div>
26 </ion-view>
```

controls.html

Default language

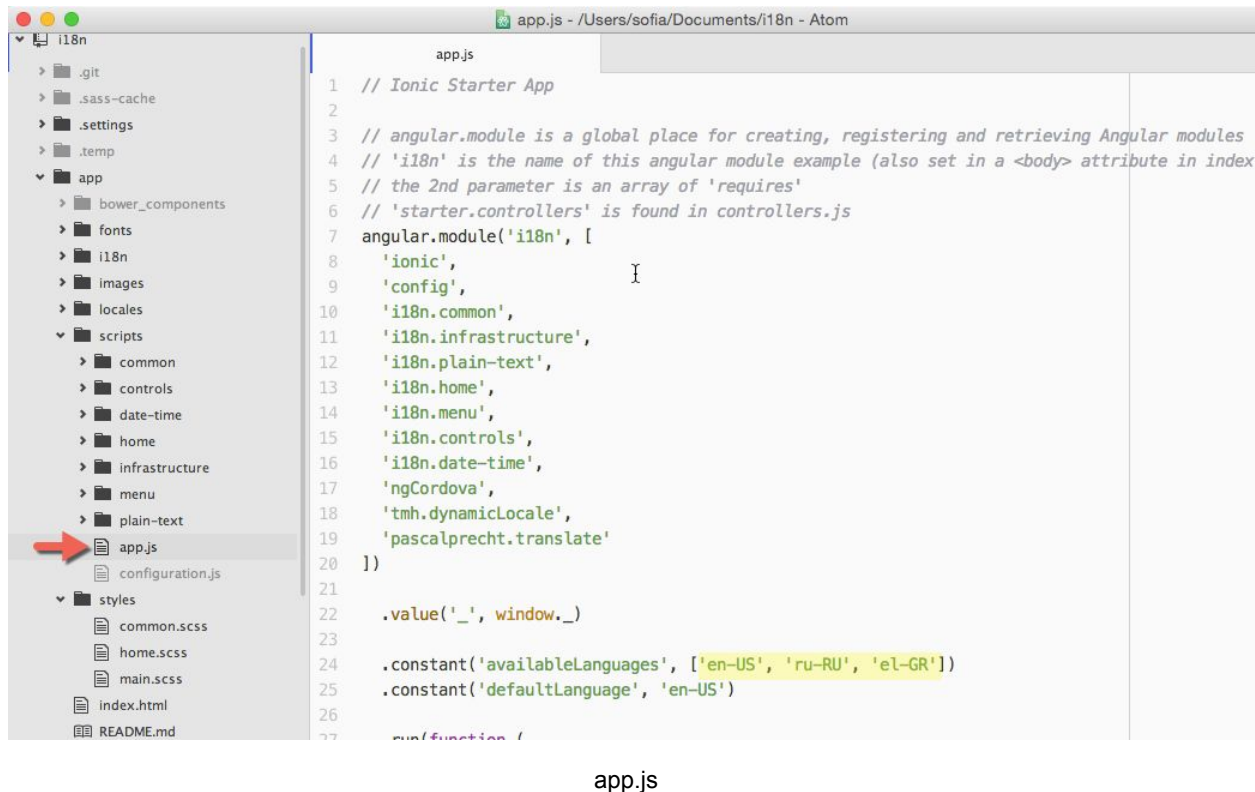
In order to set the default language of the application when no other languages are available, you should modify the highlighted area in the `app.js` file under the `app/scripts` path:



app.js

Available languages

All the available languages should be declared in the highlighted area in the `app.js` file under the `app/scripts` path:



Add a new language/locale

For demonstration purposes, there are three available JSON files under the `app/i18n` path; one for english, one for russian and one for greek. Naturally, you could add as many languages as you prefer under the same path.

As for the locales, there are three available JS files under the `app/locales` path; one for english, one for russian and one for greek. You could add as many locales as you prefer under the same path.

More precisely, in order to add a new language/locale you should follow the following steps:

1. Create a JSON file defining the translation keys with the corresponding text in a particular language (see [Language Files and Translations](#) section).
2. Create a JS file defining the date/time format in a particular language/region (see [Language Files and Translations](#) section).
3. Add the new language in the available languages declared in the `app.js` file under the `app/scripts` path (see [Available languages](#) section).

Support

With regard to technical questions, new ideas and suggestions, you may use the dedicated form and choose the product your enquiry refers to:

<http://support.appseed.io/customer/portal/questions/new>

References / Links:

- [YouTube channel](#)
Periodically, video demonstrations and tutorials related to this product will be published in my YouTube channel.
- [Codecanyon User page](#)
You may contact me by using my user page on Codecanyon.
- [Titanium Templates Forum](#)
The Google Group that has been created for this product.
- [Quick Start Guide](#)
The online version of this document.

Thank you

Thank you again for purchasing my product. If you have any questions that are beyond of the scope of this help file, please feel free to email also via [my user page](#) contact form. --- *Stavros*.