
LET'S GET LOCAL
ANGULAR LOCALIZATION
John Niedzwiecki
@kiltedcode

ABOUT THE GEEK

```
function getDeveloperInfo() {
    const dev = {
        name: 'John Niedzwiecki',
        desc: 'Your friendly neighborhood kilted coder.',
        company: 'Disney Streaming Services',
        role: 'Senior Software Engineer',
        twitter: '@kiltedcode',
        github: 'kiltedcode',
        web: 'https://kiltedcode.github.io',
        email: 'john.niedzwiecki.ii@gmail.com',
        currentDevLoves: ['Angular', 'JavaScript Everywhere', 'NativeScript', 'CSS fun'],
        nonDevAttr : {
            isHusband: true,
            isFather: true,
            hobbies: ['running', 'disney', 'cooking', 'video games', 'photography'],
            twitter: '@itshighlandjohn',
            web: 'http://www.rungeekrundisney.com'
        }
    };
    return dev;
}
```

USERS ARE EVERYWHERE

USERS ARE EVERYWHERE

Users - and potential users - of your application are everywhere. If we want to truly connect with these users you're going to need to add application support.

ANGULAR OPTIONS

- Angular Internationalization (core)
  - i18n pipes
  - Template translations
- ngx-translate
- Pros and cons

ANGULAR INTERNATIONALIZATION

ANGULAR INTERNATIONALIZATION

Angular team provides their own tools to help you with internationalization of your app.

https://angular.io/guide/i18n

ANGULAR INTERNATIONALIZATION

Angular team provides their own tools to help you with internationalization of your app.

Internationalization is the process of designing and preparing your app to be usable in different languages.

https://angular.io/guide/i18n

ANGULAR INTERNATIONALIZATION

Angular team provides their own tools to help you with internationalization of your app.

Internationalization is the process of designing and preparing your app to be usable in different languages.

Localization is the process of translating your internationalized app into specific languages for particular locales.

https://angular.io/guide/i18n

ANGULAR AND I18N

Offer various abilities to help with internationalization

- Display dates, currencies, number, and percentages locally.
- Plural forms in templating
- *Preparing text in templates for translation.*

SETTING UP LOCALE OF YOUR APP

Need to identify your *locale* for localized value. Identifies a language and optionally a country. For example `en` for English vs `en-GB` for English (United Kingdom)

```
ng serve --configuration=pt
```

SETTING UP LOCALE OF YOUR APP

For Angular pipes, you need to load locale data for any locale you want to support other than `en-US`

```
import { registerLocaleData } from '@angular/common';
import localePt from '@angular/common/locales/pt';

registerLocaleData(localePt, 'pt');
```

Examples: `DatePipe`, `CurrencyPipe`, `DecimalPipe`, and `PercentPipe`

TEMPLATE TRANSLATIONS

STEPS TO TEMPLATE TRANSLATION

- Mark text in templates for translation
- Create translation file
- Edit and translate translation file
- Build translated application

EXAMPLE APPLICATION

Created a very simple application to demonstrate our Angular translation options.

Repository is lets-get-local. Branches available for our our different options.

For this section you'll want the branch angular-internationalization.

https://github.com/KiltedCode/lets-get-local

MARK STATIC TEXT

Mark text for translation in the component templates with the `i18n` attribute. It is a custom attribute that compiler will recognize and remove.

GIVE CONTEXT TO TEXT

In addition to marking the text, we can provide additional valuable information for translating. We can provide a description and a meaning.

```
<a routerlink="/" i18n="back link|Link to go back to main page.">
    Back
</a>
```

PERSISTANT ID

When your data is extracted to a translation file, it gets a unique ID. When you change the text, the tool would generate a new ID. You can instead set the ID with `@@yourId`

```
<h1 i18n="title|Main title of application.@@title">
    Main Street U.S.A.
</h1>
```

TRANSLATING ATTRIBUTES

In HTML, some attributes provide valuable context through text. You can mark attribute for translation with `i18n-x`

```
<img src="./assets/misi.jpg" alt="Move it and shake it with Judy
```

TRANSLATE MORE

If you want to take advantage of pluralization (cardinality) or if you need select among different text based on a variable both are supported through ICU Messages.

TRANSLATING MORE

```
<span i18n="@@icuMessage">
    Table for {guests, plural,
        =1 {one}
        =2 {two}
        =3 {three}
        =4 {four}
        other {a large party}
    }
    for {time, select,
        morning {breakfast}
        lateMorning {brunch}
        midDay {lunch}
        evening {dinner}
        other {food}
    }
</span>
```

CREATE TRANSLATION FILE

Next we extract all of the marked text and create a starting source file for translations. We'll do this with the help of the Angular CLI.

```
ng xi18n --output-path locale --i18n-format=xlf
```

MESSAGES.XLF

```
<!--?xml version="1.0" encoding="UTF-8" ?-->
<xliff version="1.2" xmlns="urn:oasis:names:tc:xliff:document:1.2
    <file source-language="en" datatype="plaintext" original="ng2
    <body>
        <trans-unit id="title" datatype="html">
        <source>Main Street U.S.A.</source>
        <context-group purpose="location">
            <context context-type="sourcefile">app/home/home.comp
            <context context-type="linenumber">2</context>
        </context-group>
        <note priority="1" from="description">Main title of appli
        <note priority="1" from="meaning">title</note>
        </trans-unit>
        ....
        <trans-unit id="trolleyImageAlt" datatype="html">
        <source>See the trolley kids sing and dance their way dow
```

TRANSLATE ALL THE THINGS!

You need to take this created file and make your translation files. Easiest to start with this and create your initial language source file - in our case english.

- ✪ Rename as `messages.en.xlf`
- ✪ Add a `<target>` for every `<source>`
- ✪ Profit!

(just kidding, there's more work to do)

LATHER, RINSE, REPEAT

messages.pr-br.xlf

```
<!--?xml version="1.0" encoding="UTF-8" ?-->
<xliff version="1.2" xmlns="urn:oasis:names:tc:xliff:document:1.2
    <file source-language="en" datatype="plaintext" original="ng2
    <body>
        <trans-unit id="title" datatype="html">
            <source>Main Street U.S.A.</source>
            <target>Main Street U.S.A</target>
            <context-group purpose="location">
                <context context-type="sourcefile">app/home/home.
                <context context-type="linenumber">2</context>
            </context-group>
            <note priority="1" from="description">Main title of a
            <note priority="1" from="meaning">title</note>
        </trans-unit>
        ...
        <trans-unit id="trolleyImageAlt" datatype="html">
```

BUILD WITH TRANSLATION FILES

You need to update your build to let the compiler know about your files and format.

AOT TRANSLATION BUILD

```
"build": {
    ...
    "configurations": {
        ...
        "pt-br": {
            "aot": true,
            "outputPath": "dist/main-street-pt-br/",
            "baseHref": "/pt-br/",
            "i18nFile": "src/locale/messages.pt-br.xlf",
            "i18nFormat": "xlf",
            "i18nLocale": "pt-br",
            ...
            // rest of build configurations
            "buildOptimizer": true
        }
    }
```

AOT TRANSLATION SERVE

```
"serve": {
    ...
    "configurations": {
        ...
        "pt-br": {
            "browserTarget": "mainstreet-usa:build:pt-br"
        }
    }
}
```

OR YOU CAN PASS TO CLI

```
ng build --prod --i18n-file src/locale/messages.pt-br.xlf --i18n-
```

FUN WITH TRANSLATIONS

You can actually add translations for *any* language you want. Can be useful for testing purposes. I suggest just check to make sure you use locale not from the list of real ones.

```
ng serve --configuration=kl
```

NGX-TRANSLATE

NGX-TRANSLATE

NGX-Translate is an internationalization library for Angular, allowing you to define and switch between languages easily.

STEPS TO NGX-TRANSLATE

- Import and Configure
- Initialize the Service
- Define the translations
- Integrate with the service, pipe, or directive

EXAMPLE APPLICATION

For this section you'll want the branch ngx-translate.

https://github.com/KiltedCode/lets-get-local

INSTALL CORE

Start by installing the core library. Additionally you'll need a loader for your translations files.

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

IMPORT AND CONFIGURE

Import `TranslateModule` & configure with loader.

```
import { TranslateModule, TranslateLoader } from '@ngx-translate/
import { TranslateHttpLoader } from '@ngx-translate/http-loader';

export function HttpLoaderFactory(http: HttpClient) {
    return new TranslateHttpLoader(http, './assets/locale/', '.js
}

@NgModule({
    imports: [
        TranslateModule.forRoot({
            loader: {
                provide: TranslateLoader,
                useFactory: HttpLoaderFactory,
                deps: [HttpClient]
            }
        })
```

INITIALIZE THE TRANSLATESERVICE

Set up the service default and languages.

```
export class HomeComponent {
    constructor( translate: TranslateService ) {
        translate.setDefaultLang('en');
        translate.addLangs(['en', 'pt']);

        const browserLang = translate.getBrowserLang();
        translate.use(browserLang);
    }
}
```

DEFINE THE TRANSLATIONS

Create a json file per language: en.json

```
{
    "ABOUT": {
        "LINK_TEXT": "About Text",
        "TEXT": "Description text for attractions from <a href="h
    },
    ...
    "SUCCESS": {
        "ADD_SCHEDULE": "Added {{attraction}} successfully to sch
    }
}
```

DEFINE THE TRANSLATIONS

Create a json file per language: pt.json

```
{
    "ABOUT": {
        "LINK_TEXT": "Sobre o texto",
        "TEXT": "Texto de descrição para atrações de <a href="htt
    },
    ...
    "SUCCESS": {
        "ADD_SCHEDULE": "Adicionado {{attraction}} com sucesso pa
    }
}
```

USE THE DIRECTIVE

Use the directive with HTML tags. You can also pass in translations with HTML

```
<a routerlink="/about" [translate]="'ABOUT.LINK_TEXT'">
    About Text
</a>
<p [innerhtml]="'ABOUT.TEXT' | translate"></p>
```

USE THE PIPE

Use the pipe when interpolating a variable

```
this.buttonText = 'ADD_SCHEDULE.BTN_TEXT';

<button (click)="addToSchedule()">
    {{ buttonText | translate }}
</button>
```

USE THE SERVICE

Use the service to translate in your component code

```
this.translate.get(successMsg, {attraction: attraction})
    .subscribe(
        (res: string) => {
            this.message = res;
        }
    );
this.error = this.translate.instant(errorMsg, {attraction: attrac
```

COMPARING THE METHODS

ANGULAR INTERNATIONALIZATION: PROS

- ✪ Compiled templates instead of waiting for loading and interpreting
- ✪ Can provide directory based language solutions
- ✪ Industry standard formats
- ✪ Meaning and context defined for translators

ANGULAR INTERNATIONALIZATION: CONS

- ✪ No translating in service
- ✪ No way to dynamically translate data from back end
- ✪ More complex file format
- ✪ Requires multiple compiled sources

NGX-TRANSLATE: PROS

- Dynamic translations at anytime via 3 methods
- Easily switch on the fly
- Easy to use file format
- Multiple loaders

NGX-TRANSLATE: CONS

- ✪ Requires asynchronous loading files then interpreting
- ✪ No meaning and context for tranlators

BUT WAIT, THERE'S MORE!

THE FUTURE

Angular Internationalization is updating with Ivy.

Features

- [ ] Runtime i18n (one bundle for all locales with AOT) - [**working on it**]
- [ ] ID migration tool (for when we break ID generation) - [PR #15621]
- [ ] Use translation strings outside of a template - #11405 - [**working on it**]
- [ ] Generate the same ID for xmb/xlf - #15136 [**breaking change** PR #15621]

https://github.com/angular/angular/issues/16477

THE FUTURE

This is dependent on Ivy which will be released independent of v7. This won't be complete and ready until after Ivy is feature complete, so no timetable on when to expect this yet.

https://github.com/angular/angular/issues/16477

PICK THE SOLUTION THAT WORKS FOR YOU

SIT BACK & LET YOUR USERS ENJOY

Find me: @kiltedcode

https://kiltedcode.github.io/

Code: https://github.com/KiltedCode/lets-get-local

Title photo by Jelleke Vanooteghem on Unsplash