Gemeente Haarlem

Angular - Maatwerk

Peter Kassenaar
info@kassenaar.com

Internationalization

Using multiple languages and locales in your app

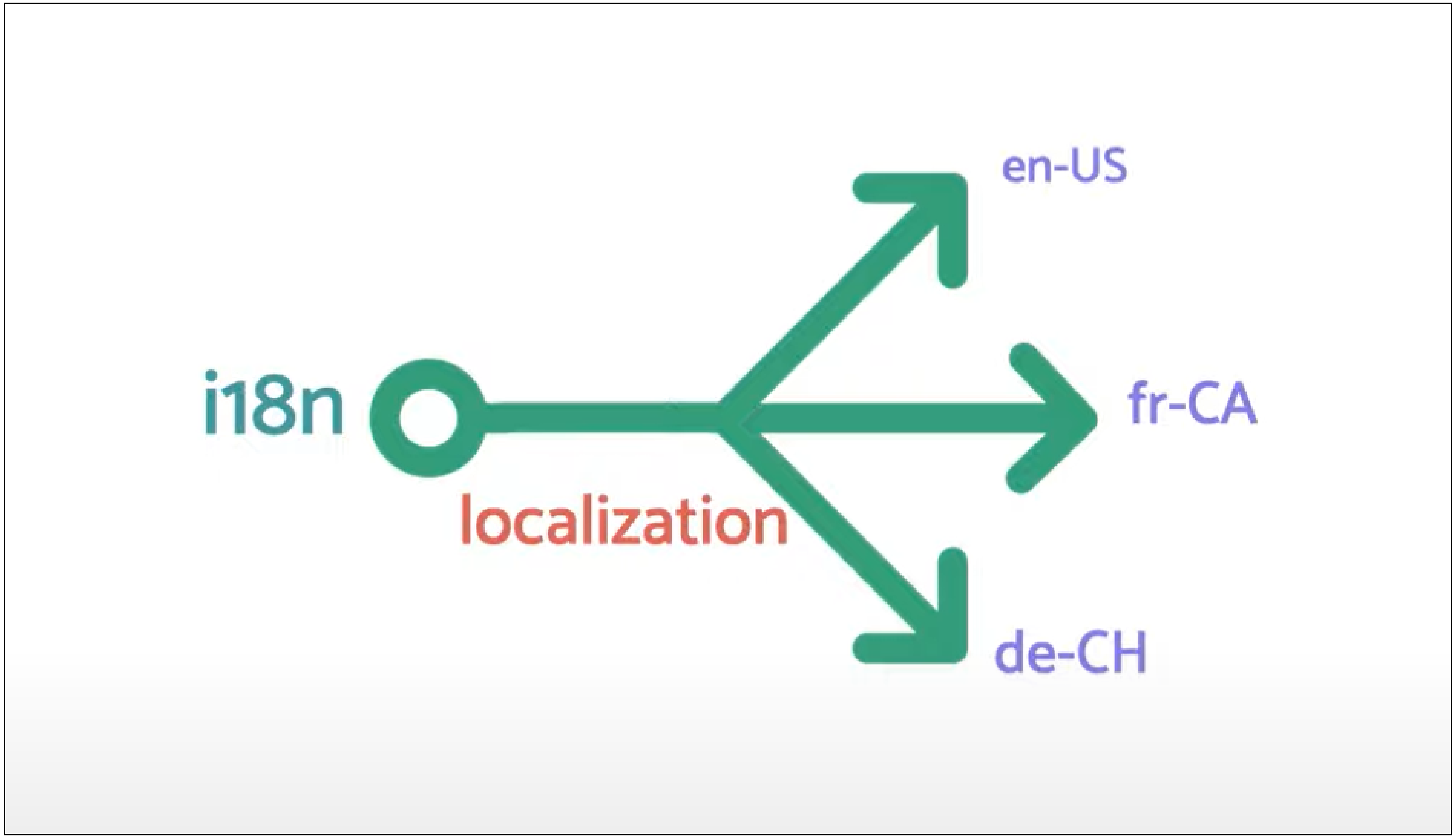
I18n Process

- Internationalization vs Localization
  - Related, but NOT the same!
- Internationalization (i18n):
  - Preparing your app to use multiple languages
  - Separating content for translation
  - Updating angular.json with locales
- Localization:
  - Building your app for a specific language
  - Default: NO RUNTIME SWITCHING of languages
  - Each locale gets its own build

Locales

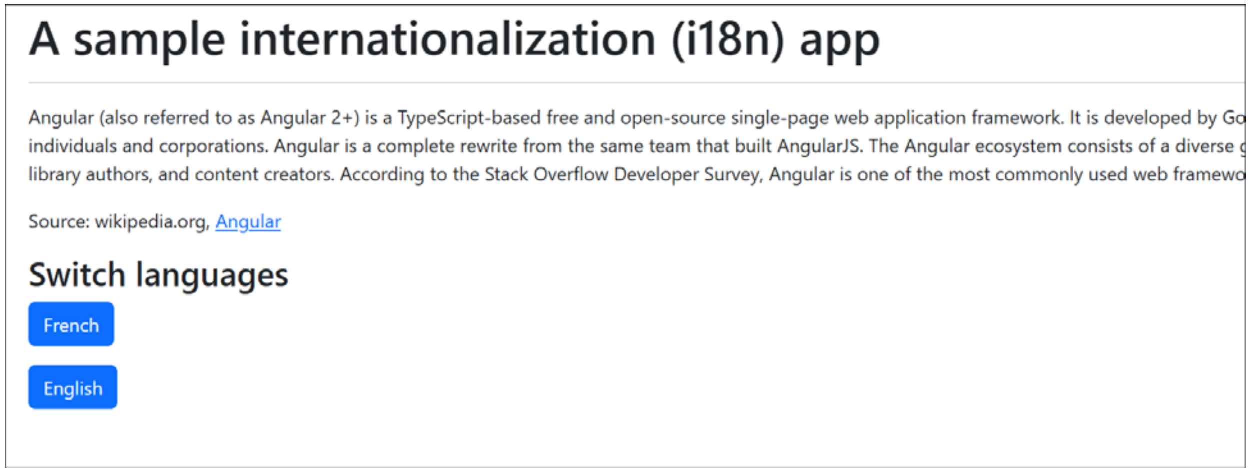
- Are mentioned in the format `{language}-{locale}`
- For instance:
  - `en-US`
  - `fr-CH`
  - `nl-NL`
- Internationalization happens ONCE (1x)
- Localization happens MANY TIMES (as much locales as you want to support
- You can change locale, without rewriting your application code
  - Page reload required. NOT at runtime with default Angular solution.

Internationalization



3 steps in localization

- 1. Install dependencies
  - Using `npm install <package>`…
- 2. internationalize the application
  - Add i18n tags and attributes to existing code
- 3. Localize the application
  - Repeat step 3 for every locale you want to add


A sample internationalization (i18n) app

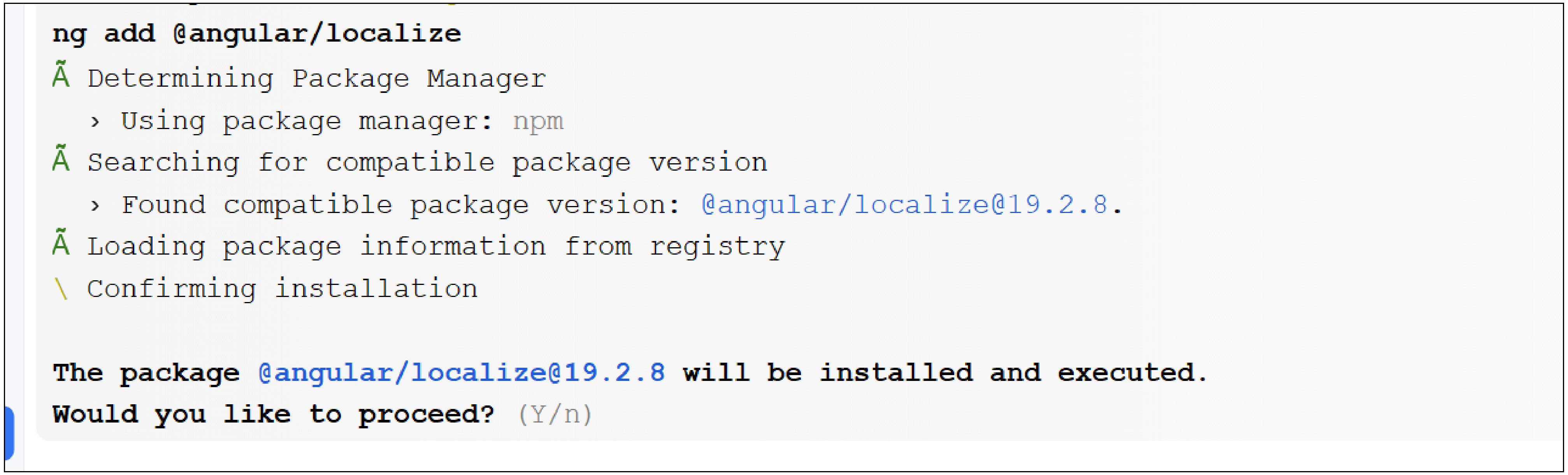
Angular (also referred to as Angular 2+) is a TypeScript-based free and open-source single-page web application framework. It is developed by Go
individuals and corporations. Angular is a complete rewrite from the same team that built AngularJS. The Angular ecosystem consists of a diverse g
library authors, and content creators. According to the Stack Overflow Developer Survey, Angular is one of the most commonly used web framewo

Source: wikipedia.org, Angular

Switch languages

1. Add dependencies

- `ng add @angular/localize`
  - Reply Yes to confirm installation

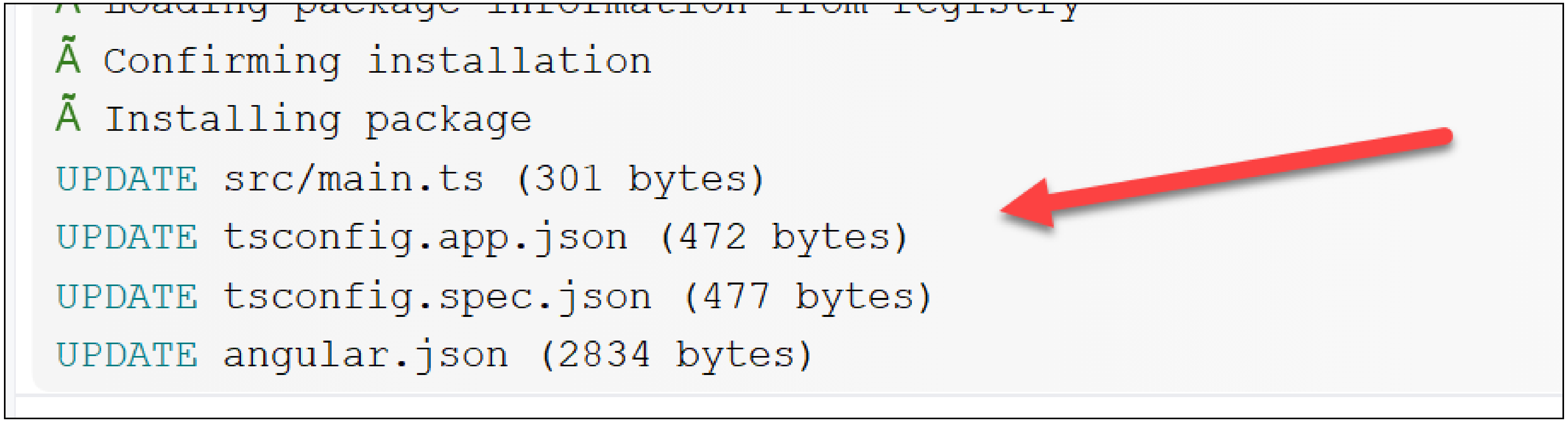


```
ng add @angular/localize
Ã Determining Package Manager
  › Using package manager: npm
Ã Searching for compatible package version
  › Found compatible package version: @angular/localize@19.2.8.
Ã Loading package information from registry
\ Confirming installation

The package @angular/localize@19.2.8 will be installed and executed.
Would you like to proceed? (Y/n)
```

When adding localization

- Files are updated:
  - `@angular/localize` added in `angular.json` as polyfill
  - `/// <reference types="@angular/localize" />` added to `main.ts`
  - `"types": [ "@angular/localize" ]` added to `tsconfig.json`



```
Ã Confirming installation
Ã Installing package
UPDATE src/main.ts (301 bytes)
UPDATE tsconfig.app.json (472 bytes)
UPDATE tsconfig.spec.json (477 bytes)
UPDATE angular.json (2834 bytes)
```

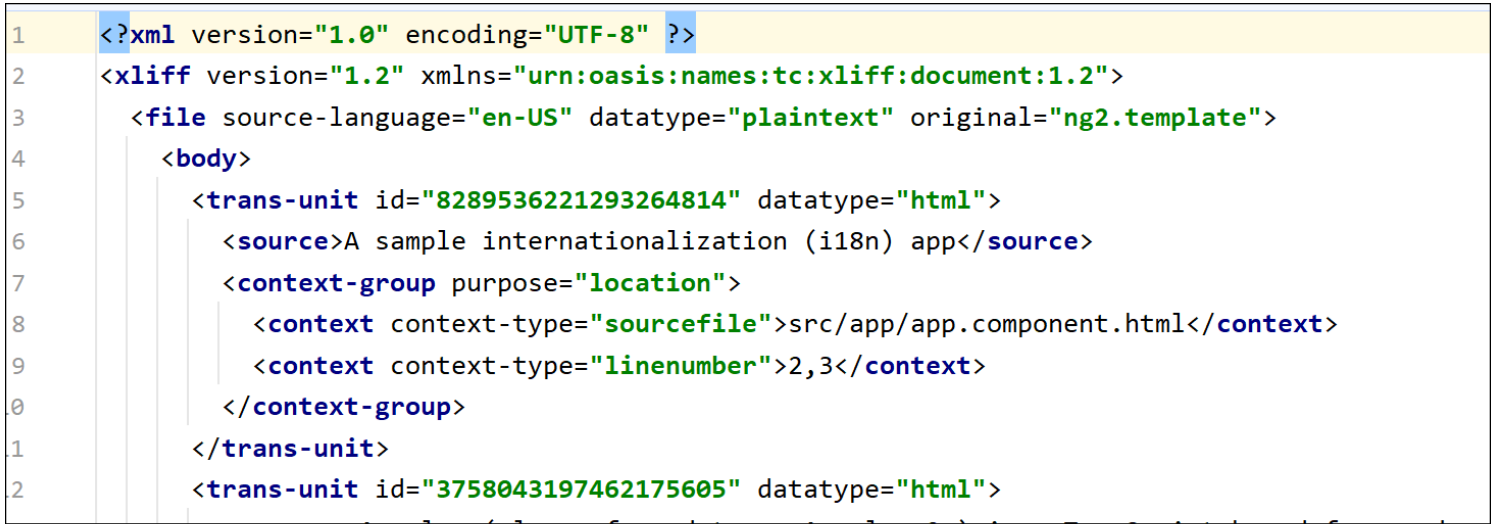
2. Preparing your app for i18n

- Add the i18n attribute to tags/content you want to translate
- Simple attribute for default translation, more finegrained options available for adding custom id’s, meaning, description.

```
<div class="container">
  <h1 i18n>A sample internationalization (i18n) app</h1>
  <hr>
  <p i18n>
    Angular (also referred to as Angular 2+) is a …
    ….
  </p>
</div>
```

3. Extract elements

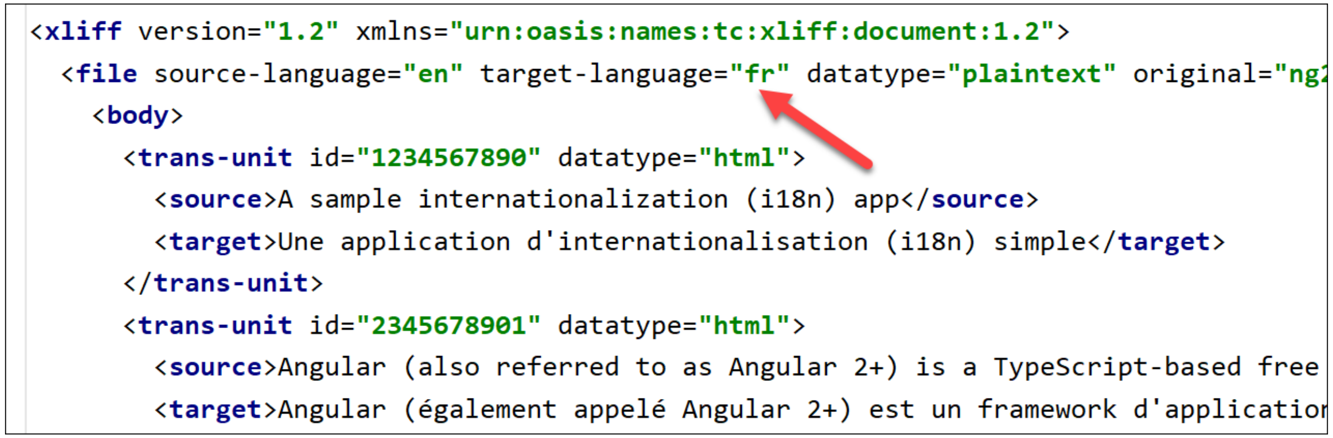
- Extract all elements that has been marked for translation
  - using the `extract-i18n` tool
- 1: create a `src/locale` directory.
  - This will contain the translation files
- 2: `ng extract-i18n --output-path src/locale`
  - This will create a `messages.xlf` file in the `src/locale` directory.



```
<?xml version="1.0" encoding="UTF-8" ?>
<xliff version="1.2" xmlns="urn:oasis:names:tc:xliff:document:1.2">
  <file source-language="en-US" datatype="plaintext" original="ng2.template">
    <body>
      <trans-unit id="8289536221293264814" datatype="html">
        <source>A sample internationalization (i18n) app</source>
        <context-group purpose="location">
          <context context-type="sourcefile">src/app/app.component.html</context>
          <context context-type="linenumber">2,3</context>
        </context-group>
      </trans-unit>
      <trans-unit id="3758043197462175605" datatype="html">
```

4. Add other translations

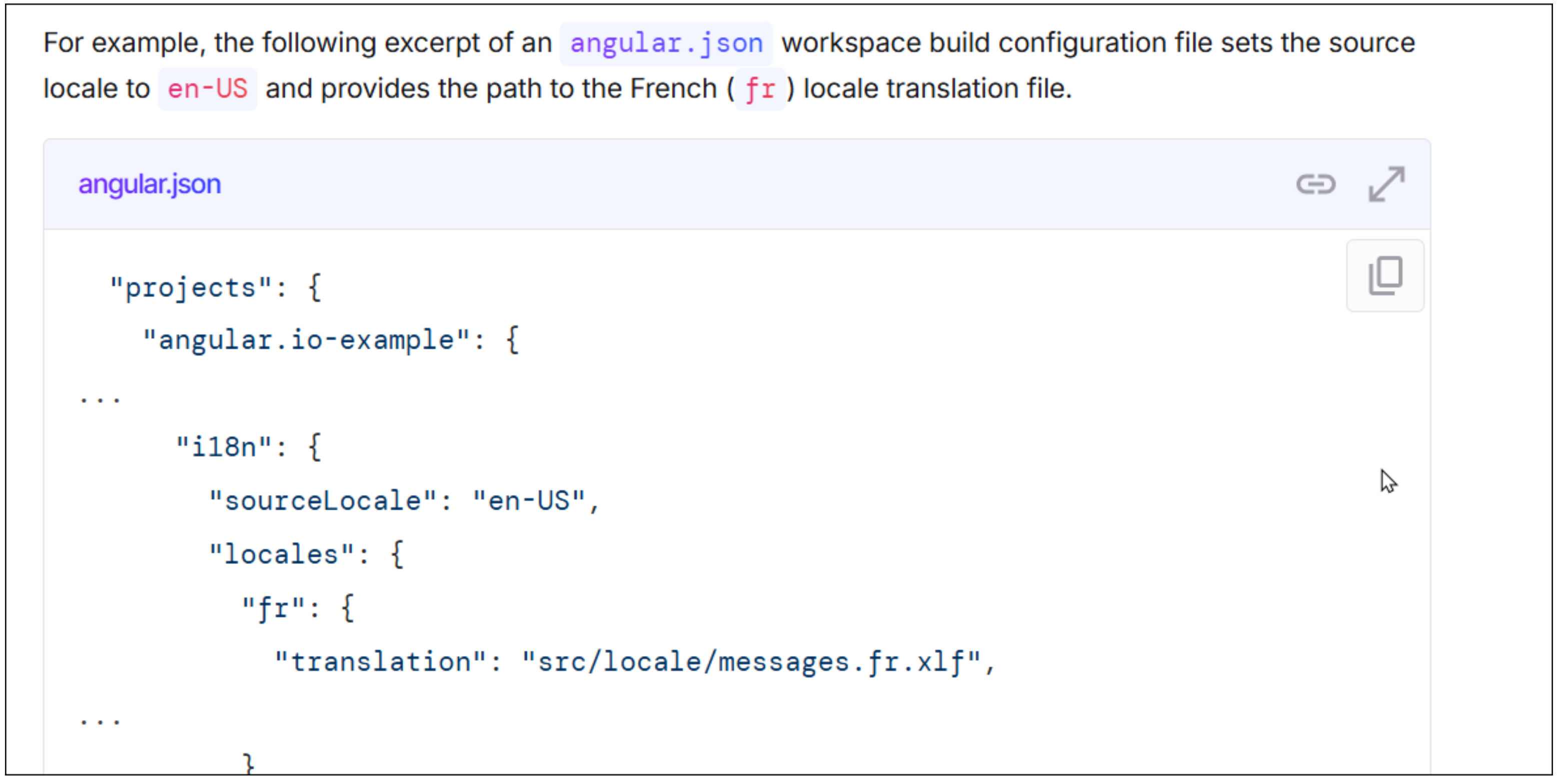
- `.xlf` is a standard format, used by translation agencies worldwide
  - https://en.wikipedia.org/wiki/XLIFF
- Copy the `messages.xlf` file to (for instance) `messages.fr.xlf` for French translations
  - Copy and rename to cover other languages.
- Cover all messages and translate them
  - Often professional translation software that can read/write `.xlf` is used.
  - Not recommended to do this directly in the XML.



```
<xliff version="1.2" xmlns="urn:oasis:names:tc:xliff:document:1.2">
  <file source-language="en" target-language="fr" datatype="plaintext" original="ng2
    <body>
      <trans-unit id="1234567890" datatype="html">
        <source>A sample internationalization (i18n) app</source>
        <target>Une application d'internationalisation (i18n) simple</target>
      </trans-unit>
      <trans-unit id="2345678901" datatype="html">
        <source>Angular (also referred to as Angular 2+) is a TypeScript-based free
        <target>Angular (également appelé Angular 2+) est un framework d'applicatio
```

5. Update angular.json to support .fr

- Update the section `projects.<name>.architect.build.options`
- Documentation: https://angular.dev/guide/i18n/merge#angularjson-for-en-us-and-fr-example



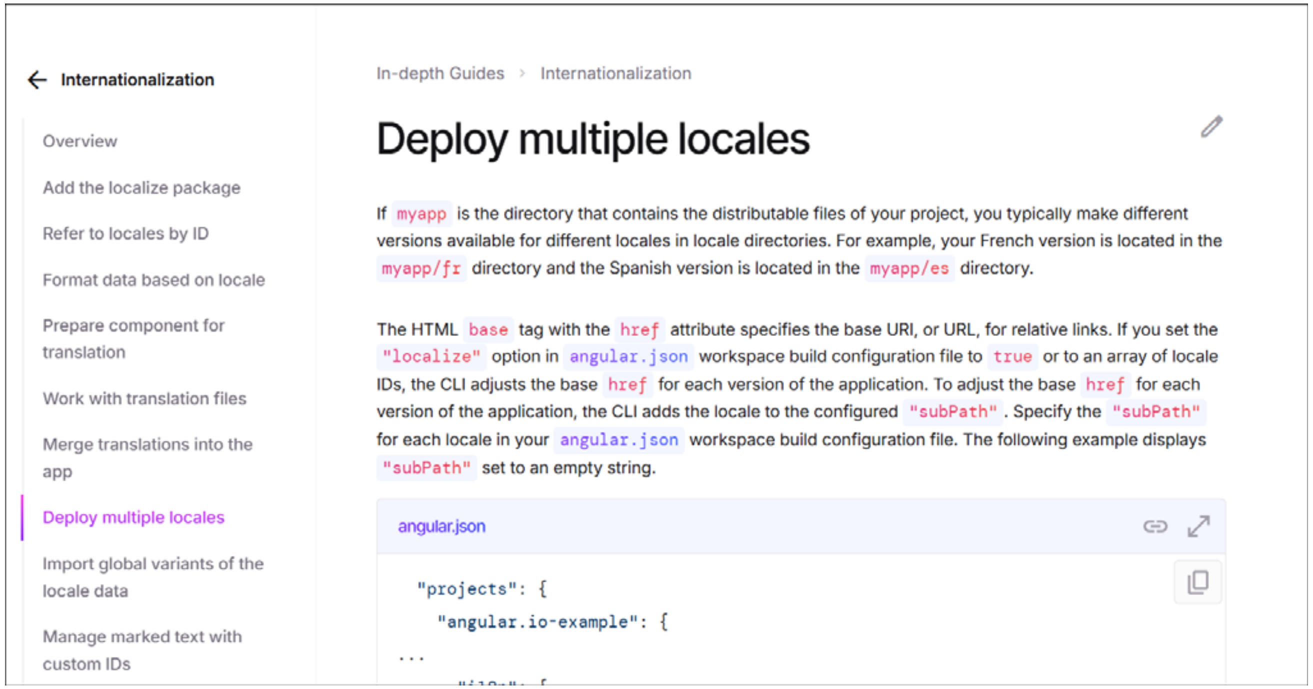
For example, the following excerpt of an `angular.json` workspace build configuration file sets the source locale to `en-US` and provides the path to the French (`fr`) locale translation file.

angular.json

```
  "projects": {
    "angular.io-example": {
...
      "i18n": {
        "sourceLocale": "en-US",
        "locales": {
          "fr": {
            "translation": "src/locale/messages.fr.xlf",
...
          }
```

6. Deploy the locales

- Build application. If you use the flag --localize, all locales are generated.
  - ng build --localize
  - https://angular.dev/guide/i18n/deploy



← Internationalization
Overview
Add the localize package
Refer to locales by ID
Format data based on locale
Prepare component for translation
Work with translation files
Merge translations into the app
Deploy multiple locales
Import global variants of the locale data
Manage marked text with custom IDs
In-depth Guides › Internationalization

Deploy multiple locales

If `myapp` is the directory that contains the distributable files of your project, you typically make different versions available for different locales in locale directories. For example, your French version is located in the `myapp/fr` directory and the Spanish version is located in the `myapp/es` directory.

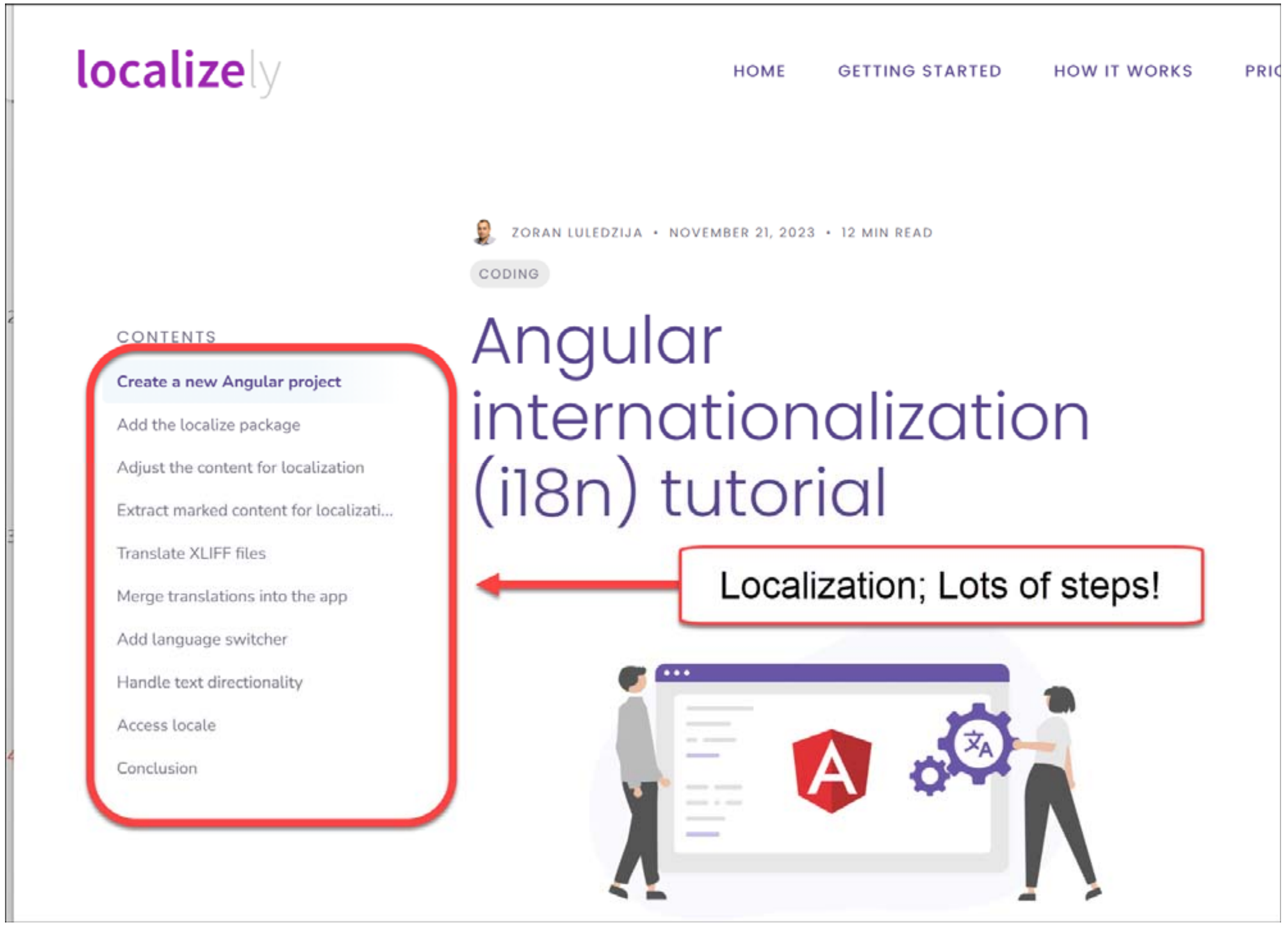
The HTML `base` tag with the `href` attribute specifies the base URI, or URL, for relative links. If you set the `"localize"` option in `angular.json` workspace build configuration file to `true` or to an array of locale IDs, the CLI adjusts the base `href` for each version of the application. To adjust the base `href` for each version of the application, the CLI adds the locale to the configured `"subPath"`. Specify the `"subPath"` for each locale in your `angular.json` workspace build configuration file. The following example displays `"subPath"` set to an empty string.

angular.json

```
  "projects": {
    "angular.io-example": {
...
```

Read Article

- Refer to official documentation
- Use article from localizely: https://localizely.com/blog/angular-i18n-tutorial/



localizely

HOME GETTING STARTED HOW IT WORKS PRIC

ZORAN LULEDZIJA • NOVEMBER 21, 2023 • 12 MIN READ

CODING

Angular internationalization (i18n) tutorial

CONTENTS

Create a new Angular project
Add the localize package
Adjust the content for localization
Extract marked content for localizati...
Translate XLIFF files
Merge translations into the app
Add language switcher
Handle text directionality
Access locale
Conclusion

Localization; Lots of steps!