Documentation: Transloco Setup for Angular 14

For your requirements of an MIT-licensed Angular library to manage language changes dynamically in an application (such as through a dropdown change event), you can consider the following approach using **Transloco**

Why Transloco?

MIT License: Transloco is MIT licensed, which fits your requirement.as you requested on our conversation

Dynamic Language Switching: It supports dynamic language changes triggered by a dropdown or any event. Transloco will automatically update all the translations in your application when the language is changed.

No Third-Party Dependencies: Transloco is Angular-centric and does not rely on third-party services like `@ngx-translate`.

Prerequisites:

1. **Node.js** and **npm** installed.
2. **Angular 14** installed.

Step 1: Install Transloco

Transloco comes with a build-in schematics and ng add command. So in order to install Transloco you could simply type ng add @ngneat/transloco . (***ng add @ngneat/transloco)***which will install the library for you including with simple config for your preferences questions.This will be through a prompt

Translation path: Define where your translation files will be stored (`src/assets/i18n/` is a common choice).

Default language: Choose a default language (e.g., `en` for English).

Step 2: Project Configuration

Once installed, Transloco will generate the necessary setup files. It will automatically add the required imports in your `app.module.ts` file.

```
@NgModule({
  declarations: [AppComponent],
  imports: [BrowserModule, HttpClientModule, TranslocoRootModule],
  bootstrap: [AppComponent]
})  //should be something similar with this
```

Step 3: Define Translation Files

Create JSON files in the `assets/i18n/` directory for each language. For example, create `en.json` and `fr.json`

Example

assets/i18n/en.json:

```
{
  "TITLE": "Welcome",
  "GREETING": "Hello"
}
```

assets/i18n/fr.json:

```
{
  "TITLE": "Bienvenue",
  "GREETING": "Bonjour"
}
```

Step 4: Using Translations in Components

This is an example how you can use this library in a dropdown .May require different changes to fit your code .

Create a dropdown for selecting languages

```
<select (change)="changeLanguage($event.target.value)">
  <option value="en">English</option>
  <option value="fr">French</option>
</select>
```

Implement the language switch logic in the component

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

```
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
})
export class AppComponent {
  constructor(private translocoService: TranslocoService) {}

  changeLanguage(lang: string) {
    this.translocoService.setActiveLang(lang);
  }
}
```

Websites that can help you :

A Better Translation Experience With Transloco | by Itay Oded | Angular In Depth | Medium

Angular Multilingual Application | Hossein Mousavi | Angular In Depth (medium.com)

Transloco Angular i18n (jsverse.github.io)

Example HTML (With Transloco Integration):

```
<div *ngIf="submitted">
  <h4>{{ 'TUTORIAL_SUBMITTED_SUCCESS' | transloco }}</h4>
  <button class="btn btn-success" (click)="newTutorial()">{{ 'ADD_TUTORIAL' | transloco }}</button>
</div>
```

Translating the TypeScript (Component File)

```
import { Component } from '@angular/core';
import { TranslocoService } from '@ngneat/transloco';
import { TutorialService } from './tutorial.service'; // Assuming you have a service like this

@Component({
```

```
  selector: 'app-tutorial',
  templateUrl: './tutorial.component.html'
})
export class TutorialComponent {
  currentTutorial: any;
  submitted: boolean = false;
  message: string = '';

  constructor(
    private translocoService: TranslocoService,
    private tutorialService: TutorialService
  ) {}

  updateTutorial() {
    this.tutorialService.update(this.currentTutorial.id, this.currentTutorial)
      .subscribe({
        next: (res) => {
          console.log(res);
          // Translate the message
          this.message = this.translocoService.translate('TUTORIAL_UPDATED_SUCCESS');
        },
        error: (e) => console.error(e)
      });
  }

  newTutorial() {
    // Logic to reset or create a new tutorial
  }
}
```

- **`this.translocoService.translate('TUTORIAL_UPDATED_SUCCESS')`** retrieves the translated message based on the active language.
- `TUTORIAL_UPDATED_SUCCESS` is the translation key defined in your translation files (e.g., `en.json`, `fr.json`

Sample Translation Files

You'll need to add corresponding keys and translations to your translation files (e.g., `en.json` for English, `fr.json` for French):

`en.json` (English):

```
{
  "TUTORIAL_SUBMITTED_SUCCESS": "Tutorial was submitted successfully!",
  "ADD_TUTORIAL": "Add",
  "TUTORIAL_UPDATED_SUCCESS": "This tutorial was updated successfully!"
}
```

fr.json (French):

```
{
  "TUTORIAL_SUBMITTED_SUCCESS": "Le tutoriel a été soumis avec succès!",
  "ADD_TUTORIAL": "Ajouter",
  "TUTORIAL_UPDATED_SUCCESS": "Ce tutoriel a été mis à jour avec succès!"
}
```