---------> demo application

---------> demo unit tests

1. powershell

npm install @ngx-translate/core --save

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

2. app.module.ts

import { TranslateHttpLoader } from '@ngx-translate/http-loader';

import { TranslateLoader, TranslateModule } from '@ngx-translate/core';

export function httpLoaderFactory(httpClient: HttpClient): TranslateHttpLoader {

return new TranslateHttpLoader(httpClient, './assets/translations/', '.json');

}

imports: [

TranslateModule.forRoot({

loader: {

provide: TranslateLoader,

useFactory: httpLoaderFactory,

deps: [HttpClient]

}

})

]

Note:

If you use a SharedModule that you import in multiple other feature modules,

you can export the TranslateModule to make sure you don't have to import it in every module.

3. home.module.ts

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

TranslateModule.forChild()

4. app.component.ts

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

constructor(private readonly translateService: TranslateService) {

this.translateService.addLangs(['en', 'de']);

this.translateService.setDefaultLang('en');

this.translateService.use('en');

}

5. frame.component.html

{{ 'demo.frame.footer' | translate }}

--------> demo footer

6. frame.component.html

restliche labels im frame (html + ts) ersetzen

7. frame.component.ts

constructor(private readonly router: Router,

private readonly translateService: TranslateService) {

}

this.selectedLanguage = this.availableLanguages.find(lang => this.translateService.currentLang === lang.languageCode);

this.translateService.use(item.languageCode);

--------> demo switch language

8. app.module.ts

export function httpLoaderFactory(httpClient: HttpClient): TranslateLoader {

return new MyTranslateHttpLoader(httpClient);

}

export class MyTranslateHttpLoader implements TranslateLoader {

constructor(private readonly httpClient: HttpClient) { }

getTranslation(lang: string): Observable<any> {

return this.httpClient.get('http://localhost.de/api/translation/' + lang);

}

}

--------> demo still working

9. frame.component.html

<span [translate]="'demo.frame.footer'"></span>

--------> demo still working

10. frame.component.ts

this.translateService.addLangs(['en', 'de', 'la']);

this.translateService.setDefaultLang('la');

{

displayName: 'demo.frame.languageSelector.labelLang',

languageCode: 'la',

}

--------> demo label-language

11. mock-backend.interceptor.ts

const content = (this.page1Counter++ % 2 === 0) ? this.page1Alert1LabelKey : this.page1Alert2LabelKey;

--------> demo labelKey in alert

12. pages.module.ts

TranslateModule.forChild()

13. page-one.component.html

<button (click)="onClick()">{{ 'demo.page1.alertButton' | translate }}</button>

14. page-one.component.ts

.subscribe((result) => alert(this.translateService.instant(result)));

--------> demo working again

15. page-two.component.html

<button (click)="onClick()">{{ 'demo.page2.loadButton' | translate }}</button>

<div>{{ 'demo.page2.display' | translate }}</div>

--------> demo not working interpolation

16. page-two.component.html

<div>{{ 'demo.page2.display' | translate:{'v1': value1, 'v2': value2} }}</div>

--------> demo working again

17. page-three.component.html

<button (click)="onClick()">{{ 'demo.page3.loadButton' | translate }}</button>

{{ 'demo.page3.firstNumberCaption' | translate }}

{{ 'demo.page3.secondNumberCaption' | translate }}

--------> demo '.' and ',' missing

18. rounding-to-currency.pipe.ts

pure: false

constructor(private readonly translateService: TranslateService) {}

const parts = rounded.split('.');

switch (this.translateService.currentLang) {

case 'de': return `${parts[0]},${parts[1]}€`;

case 'en': return `${parts[0]}.${parts[1]}€`;

}

--------> demo '.' and ',' working

--------> demo unit tests not compiling

19. rounding-to-currency.pipe.spec.ts

const translateServiceMock = jasmine.createSpyObj('TranslateService', ['use']);

translateServiceMock.currentLang = 'en';

--------> demo unit tests compiling again but most are failing

20. page-one.component.spec.ts

@Pipe({

name: 'translate'

})

export class TranslateMockPipe implements PipeTransform {

transform(value: any): any {

return value;

}

}

const translateServiceMock = jasmine.createSpyObj('TranslateService', ['instant']);

translateServiceMock.instant.and.returnValue(testData);

declarations: [ PageOneComponent, TranslateMockPipe ],

providers: [

{ provide: PageService, useValue: pageServiceMock },

{ provide: TranslateService, useValue: translateServiceMock }

]

--------> demo more unit tests working

21. page-three.component.spec.ts

.overrideTemplate(PageThreeComponent, '<div></div>')

--------> demo all unit tests working

22. app.module.ts

{ provide: "windowObject", useValue: window }

23. app.component.ts

@Inject('windowObject') private readonly window: Window

// this.translateService.use('en');

const language = this.window.navigator.language;

const languageCode = language.split('-')[0];

this.translateService.use(languageCode);

---------> demo switching languages