
ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE
I18N with Angular
angular-architects.io

Contents

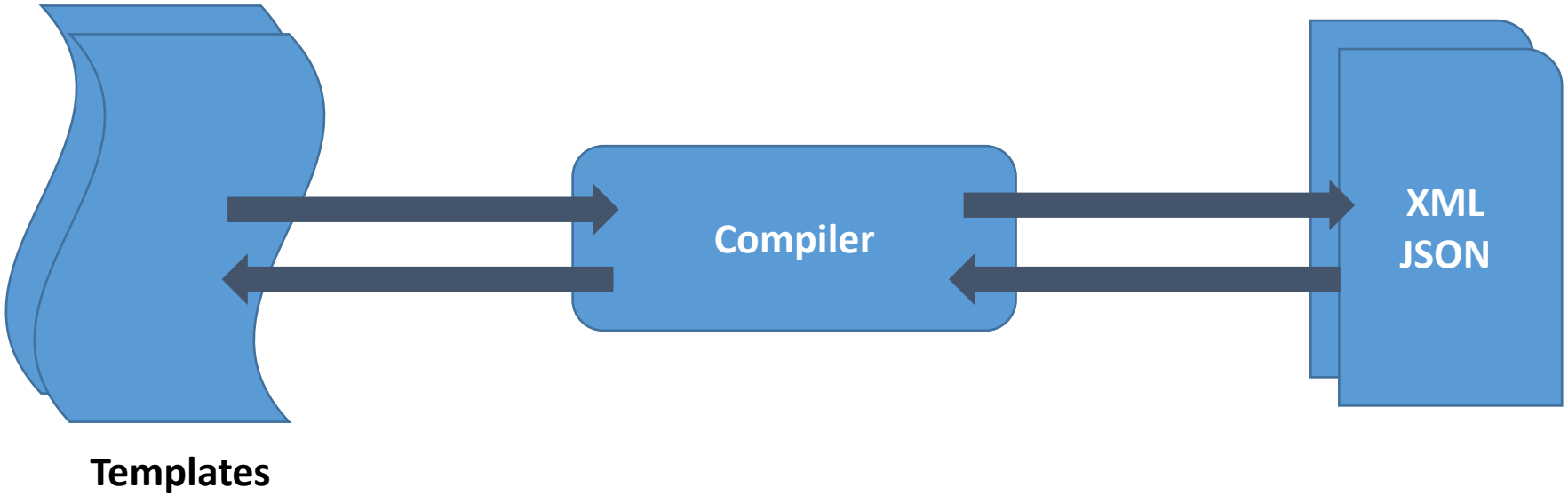
- Angular Compiler
- ngx-translate

Angular Compiler

Angular Compiler

XML Localisation Interchange File Format (XLIFF, version 1.2)

XML Message Bundle (XMB)



Angular Compiler

Great Performance

One Build per Language

Cannot Change Language w/o Reload

Reading Code at runtime (since Angular 9)

Add @angular/localize

ng add @angular/localize

Define texts to translate

```
<th i18n>From</th>
<th i18n="Description">To</th>
<th i18n="Meaning|Description">Passengers</th>
<th i18n="Meaning|Description@@passengers">Passengers</th>
```

Text to Translate in TypeScript Files

```
title =  $localize `:meaning|description@@id:Hello World!`;
```

Extract texts

```
ng extract-i18n

(formerly ng xi18n)
```

Compile app with translated texts

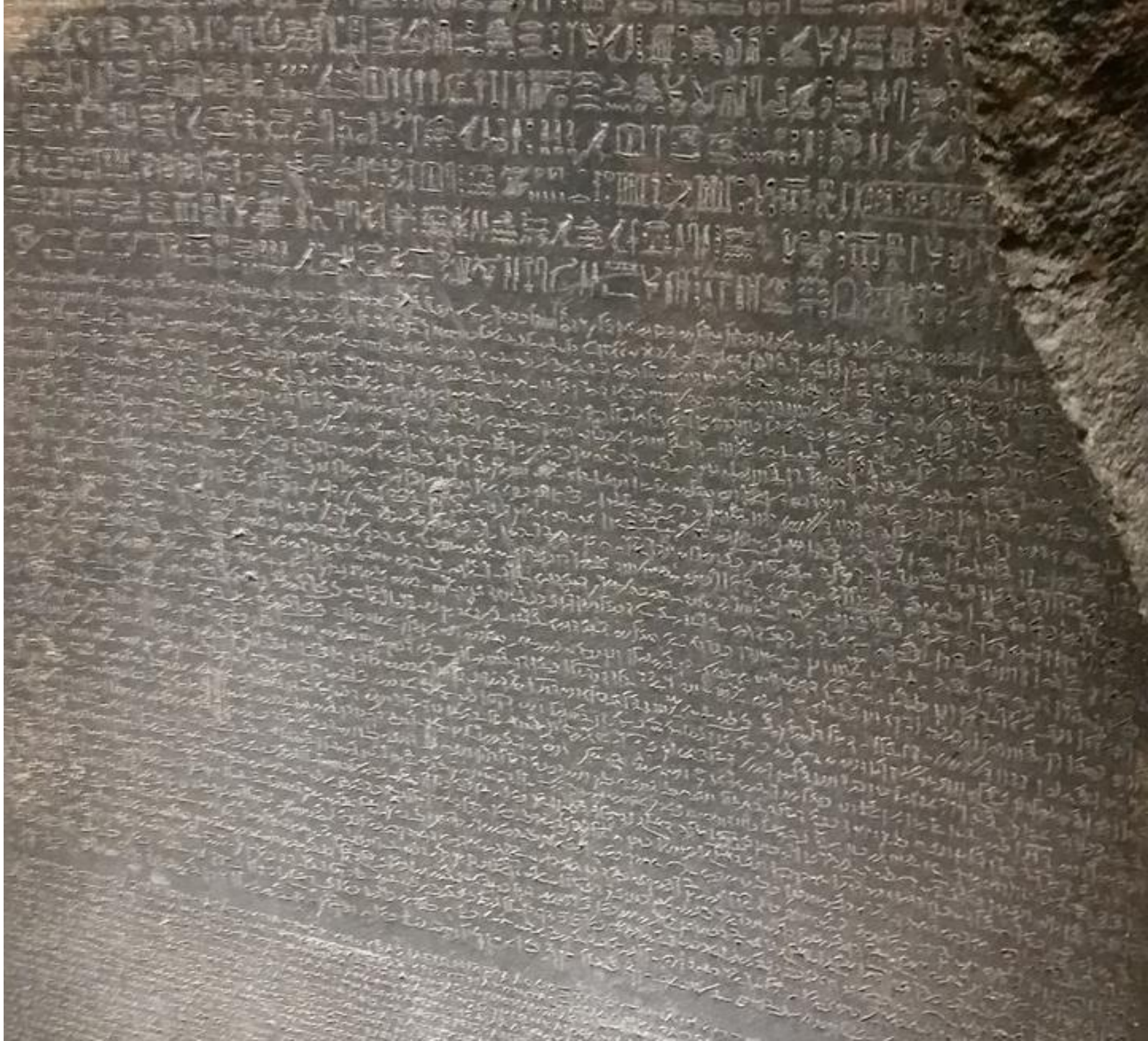
```
ng build --i18nFile=src/i18n/messages.de.xlf
         --i18nFormat=xlf
         --locale=de
         --prod
```

angular.json

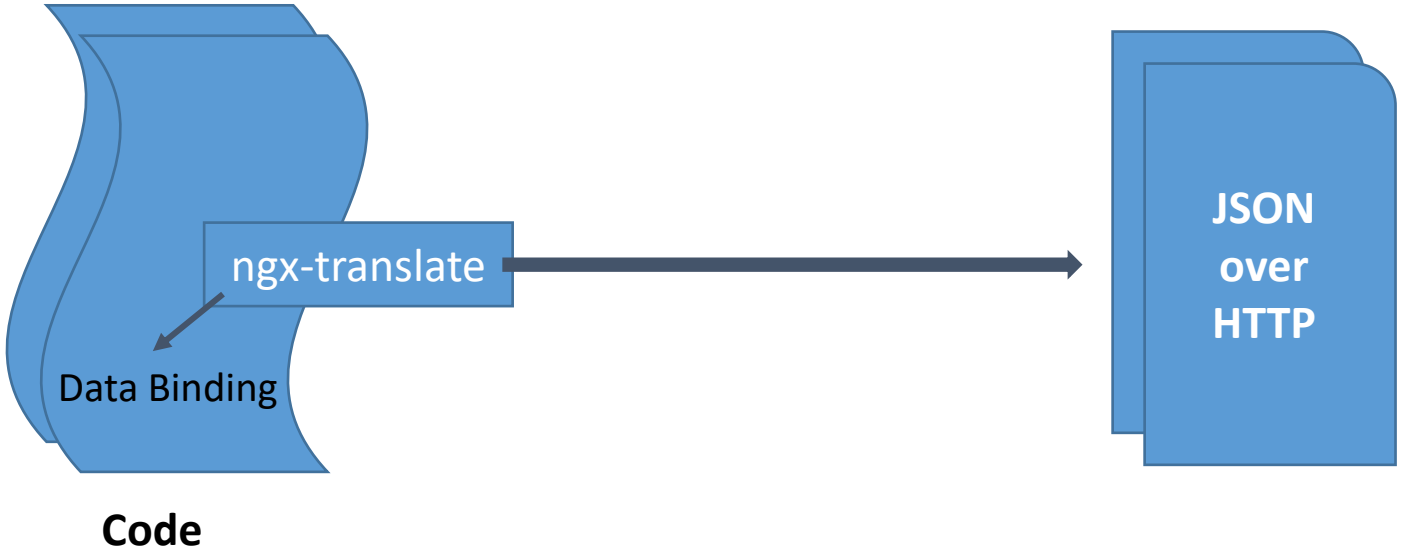
```
"i18n": {
  "locales": {
    "de": "messages.de.xlf",
    "fr": "messages.fr.xlf"
  },
  "sourceLocale": "en-US"
},
```

ng build --localize

ngx-translate



ngx-translate



Data format and way of loading are adaptable

ngx-translate

One Build

Change Language at Runtime!

Read Translations in Code

Data Binding Influences Performance

Import

```
export function createLoader(http: HttpClient) {
    return new TranslateStaticLoader(http, './i18n/', '.json');
}
```

Import

```
export function createLoader(http: HttpClient) {
   return new TranslateStaticLoader(http, './i18n/', '.json');
}

@NgModule({
   imports: [
      TranslateModule.forRoot({
         provide: TranslateLoader,
         useFactory: createLoader,
         deps: [HttpClient]
      })
   ],
   [...]
})
export class AppModule {
}
```

Initialize

```
this.translate.addLangs(['en', 'de']);
this.translate.setDefaultLang('de');
this.translate.use('de');
```

Resource File

```
{
  "FLIGHTS": {
    "from": "From",
    "to": "to",
    "search": "Search",
    "found": "{{count}} flights found."
  }
}
```

Translation

```
<th>{{'FLIGHTS.from' | translate}}</th>

<th>{{'FLIGHTS.found' | translate:{count: 7} }}</th>
```

DEMO

Local Formats

Pipes and local Formats

{{ flight.date | **date:'long'** }}

Import format information

```
import { registerLocaleData } from '@angular/common';
import localeDe from '@angular/common/locales/de';
import localeDeAt from '@angular/common/locales/de-AT';
import localeEs from '@angular/common/locales/es';

registerLocaleData(localeDe);     // de-DE
registerLocaleData(localeDeAt);   // de-AT
registerLocaleData(localeEs);     // es-ES
```

Define default locale

```
providers: [
    […]
    { provide: LOCALE_ID, useValue: 'de' },
]
```

Use Formats

```
<p>Datum (Default=de): {{ item.date | date:'long' }}</p>
<p>Datum (de-AT): {{ item.date | date:'long':'': 'de-AT' }}</p>
<p>Datum (es): {{ item.date | date:'long':'': 'es' }}</p>
```

I18N and Inputs?

- Pipes are just for outputs
- Use component lib (like Angular Material)
- General concept: ControlValueAccessor