Angular 6 Training Course

Exercise D-translate

- This exercise will convert a simple **jQuery** translation interactivity into Angular.
- Create a new project

```
cd desktop
ng new translate
cd translate
ng serve --open
```

- Open both your new Angular project and the old jQuery project in your text editor.
- Review the HTML and CSS in the jQuery version. A lot of this code is reusable in the Angular version.

A component for each Spanish/English phrase.

- The main component will contain four instances of a phrase component.
- Create a new component for each phrase.

```
ng generate component phrase --dry-run
ng generate component phrase
```

- **Refactor** your code, moving HTML and CSS across into the phrase component.
- Past one phrase from the jQuery project into the template of your new phrase component.

```
<section class="panel">
    <p class="panel-es">¿Dónde está</p>
    <p class="panel-en">Where is</p>
</section>
```

- Edit the main template to include one instance of the phrase component.

```
    <app-phrase></app-phrase>
```

- Copy the contents of **styles/phrase-book.css** from the jQuery project into **phrase/phrase.component.css**.

Web fonts

- Most of the styling works except for the **Google WebFont**.
- In the root of the Angular project edit **src/styles.css**.
- Define an url to the Google WebFont in this file:

```
@import url(https://fonts.googleapis.com/css?family=Oswald);
```

- Move the .phrase-book CSS rule from the phrase.component.css to the app.component.css

```
.phrase-book{
    font-family: 'oswald', sans-serif;
    font-size: 1rem;
}
```

- Use this class in the main component.

```
<section class="phrase-book">
<app-phrase></app-phrase>
</section>
```

Multiple phrases

- Create four instances of the phrase component in the main component.
- Wrap these instances in a Flexbox. Move the .flex rule into the main CSS file.

```
<section class="phrase-book">
    <section class="flex">
        <app-phrase></app-phrase>
        <app-phrase></app-phrase>
        <app-phrase></app-phrase>
        <app-phrase></app-phrase>
    </section>
</section>
```

Define the phrase data.

- Turn the Spanish/English HTML into an array of objects in the main component.
- Each object will have an ES and an EN property.

```
    phrases;

    constructor() {
        this.phrases = [
            { es : "¿Tienes?", en:"Do you have" },
            { es : "¿Dónde está", en:"Where is" },
            { es : "Yo necesito", en:"I need" },
            { es : "la cuenta", en:"the bill" }
        ]
    }
```

- The phrases array can be typed in Typescript.

```
phrases:{ es:string, en:string}[];
```

- *The phrases array is visible in the Chrome Augury dev-tools.*

ngFor iteration over the array.

- Use *ngFor in the main component template to iterate over this array.

```
<section class="phrase-book">
    <section class="flex">
        <app-phrase *ngFor="let p of phrases">
        </app-phrase>
    </section>
</section>
```

Pass phrase data into each component.

- We need to pass down English/Spanish phrase data into each phrase component.
- Define and import an **@Input Decorator** in the phrase component.

```
import { Input } from '@angular/core';
@Input() phrase;
```

- Pass a phrase object as an argument into the phrase component instance.

```
        <app-phrase [phrase]="p" *ngFor="let p of phrases">
```

- *Each phrase object is now visible inside each phrase component in the Augury DevTools.*
- Use the phrase object in the phrase template:

```
    <section class="panel">
        <p class="panel-es">{{ phrase.es }}</p>
        <p class="panel-en">{{ phrase.en }}</p>
    </section>
```

- *Note the English phrase is not visible because it is orange text on an orange background.*

Logic to hide/reveal phrase

- The phrase component needs a boolean variable to hold the state of the hidden/revealed English phrase.

```
show:boolean=false;
```

- Add a method to toggle the state of this variable.

```
togglePhrase() {
this.show = !this.show;
```

}

- Add rollover events to trigger this method:

```
<p
(mouseenter)="togglePhrase()"
(mouseleave)="togglePhrase()"
class="panel-es">
{{ phrase.es }}</p>
```

- Add an ngClass directive on the English phrase to conditionally apply CSS rule "reveal".

```
[ngClass]="{'reveal':show}"
```

- The rule is already defined in the phrase component CSS.

EXERCISE

- Extend this example to work in reverse: rolling over English words reveals Spanish.