Aplicaciones SPA con Angular

Lección 4-5: Directivas Angular, data binding y eventos

1. El alumno creará una directiva propia. Al posicionarse sobre algún link
   del menú debe dar unos estilos a ese li, por ejemplo, modificar el color
   de fondo del link y aumentar el tamaño de letra

Para crear una directiva, deberemos ejecutar el comando:
ng generate directive [ruta / nombre de la directiva]

```
[litoral@godmachine ejercicios]$ ng generate directive directives/change-link-style
CREATE src/app/directives/change-link-style.directive.spec.ts (262 bytes)
CREATE src/app/directives/change-link-style.directive.ts (159 bytes)
UPDATE src/app/app.module.ts (508 bytes)
```

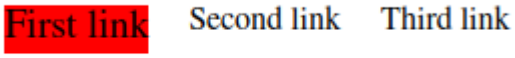
Deberemos añadir el siguiente código en
[nombre-del-archivo].directive.ts:

```
constructor(private el: ElementRef) {}
@HostListener('mouseenter') onMouseEnter() {
  this.applyStyles('red', '20px')

}
@HostListener('mouseleave') onMouseLeave() {
  this.applyStyles()

}

public applyStyles(backgroundColor = 'white', fontSize = '16px') {
  this.el.nativeElement.style.backgroundColor = backgroundColor;
  this.el.nativeElement.style.fontSize = fontSize;

}
```

Y añadirle la directiva a los elementos del template:

```
<ul>
    <li appChangeLinkStyle><a href="">First link</a></li>
    <li appChangeLinkStyle><a href="">Second link</a></li>
    <li appChangeLinkStyle><a href="">Third link</a></li>
</ul>
```

El resultado es:



2. El alumno creará una directiva propia, en el componente creado para

el formulario. Si los input no tienen información, se debe dar estilos al input de tal forma que el usuario sepa que debe escribir los datos pedidos:

Crearemos un componente y su directiva correspondiente:

```
^C[litoral@godmachine ejercicios]$ ng generate component components/form
CREATE src/app/components/form/form.component.scss (0 bytes)
CREATE src/app/components/form/form.component.html (19 bytes)
CREATE src/app/components/form/form.component.spec.ts (612 bytes)
CREATE src/app/components/form/form.component.ts (268 bytes)
UPDATE src/app/app.module.ts (593 bytes)
[litoral@godmachine ejercicios]$ ng generate directive components/form/directives/check-input
CREATE src/app/components/form/directives/check-input.directive.spec.ts (241 bytes)
CREATE src/app/components/form/directives/check-input.directive.ts (149 bytes)
UPDATE src/app/app.module.ts (708 bytes)
```

Añadimos la comprobación para el input si no tiene valor:

```
export class CheckInputDirective {

  constructor(private el: ElementRef) {}

  @HostListener('keyup') onKeyPress() {
    if (this.el.nativeElement.value == '') {
      this.el.nativeElement.style.borderColor = 'red';
    } else {
      this.el.nativeElement.style.borderColor = '';
    }

  }
```

Y la indicación al elemento para añadir la directiva:

```
<form method="POST" id="app-form" action="#">

    <input appCheckInput type="text" name="test" value="" />
</form>
```

Se ve de esta manera:

(con contenido dentro)

(cuando el input no tiene contenido)

3. El alumno hará que cuando el usuario rellene todos los inputs del formulario, la información se muestre debajo. (Pueden ayudarse de un botón para validar).

Deberemos importar el elemento FormsModule en nuestro archivo app.module.ts:

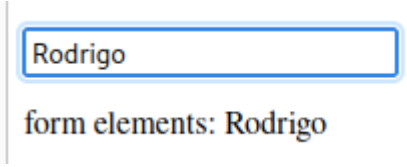
```
import { FormsModule } from '@angular/forms';

@NgModule({
  declarations: [
    AppComponent,
    ChangeLinkStyleDirective,
    FormComponent,
    CheckInputDirective,

  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    FormsModule
```

Esto nos permite añadir la directiva ngModel para poder unir valores de nuestro template y nuestra clase del componente:

```
<form method="POST" id="app-form" action="#">
    <input appCheckInput type="text" name="test" [(ngModel)]="formValues.test" />
</form>


<p>form elements: {{formValues.test}}</p>
```

```
export class FormComponent implements OnInit {

  constructor() {}

  ngOnInit(): void {
  }


  public formValues = {
    test: '',
  }
}
```

Se ve de esta manera:



4. El alumno creará un formulario con e-mail y password. Se debe comprobar que los datos que introduce son correctos, tiene formato e-mail y el password tiene más de ocho caracteres. Si los datos son correctos, el alumno debe mostrar la información proporcionada por el usuario. (Similar a ejercicio anterior, pero con validación más extensa).

```
<form method="POST" id="app-form" action="#">
    <input appCheckInput type="text" name="test" [(ngModel)]="formValues.test" />
    <label for="user-email"> Email:
        <input appCheckInput type="email" id="user-email" name="user-email" [(ngModel)]="formValues.userEmail" />
    </label>
    <label for="user-password"> Password:
        <input appCheckInput type="password" id="user-password" name="user-password"
            [(ngModel)]="formValues.userPassword" />
    </label>

    <button type="submit" (click)="validateForm()"> Submit! </button>

</form>

<p>form elements: {{formValues.test}}</p>

<p *ngIf="isValidated">user form elements: {{formValues.userEmail}}, {{formValues.userPassword}}</p>
```

Al formulario se le ha añadido dos inputs de tipo email y password, y se les ha unido su valor al objeto formValues de la misma manera que en el ej. 3, y su chequeo si el input está vacío. También se ha añadido el párrafo que muestra los elementos pertinentes si el formulario es validado correctamente.

```
public isValidated = false;

public validateForm() {
  let emailRegex = /^\w+([\.-]?\w+)*@\w+([\.-]?\w+)*(\.\w{2,3})+$/;
  if (this.formValues.userEmail.match(emailRegex) && this.formValues.userPassword.length > 8) {
    this.isValidated = true;
  }
```

Cuando el formulario es validado correctamente, el booleano se vuelve true, lo cual muestra el contenido de los dos inputs: