Leccion 5 Documentacion

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.

Primero creamos una directiva para poder configurar el evento tanto del mousover (para cambiar el fondo de color) como del mouseout (para volver a su estado normal).

Esto lo hemos creado con el comando :

Ng g d directivas/RedColor

Una vez creada, añadimos dentro del constructor el elemento “elementRef” y los HostListener correspondeientes para los eventos:

```
constructor(
  private elementRef : ElementRef
) { }

@HostListener('mouseover')
public onMouseOver(){
  this.elementRef.nativeElement.style.background = 'red'
  this.elementRef.nativeElement.style.fontSize = 'large'
}

@HostListener('mouseout')
public onMouseOut(){
  this.elementRef.nativeElement.style.background = 'white'
  this.elementRef.nativeElement.style.fontSize = 'medium'
}
```

\src\app\directivas

Una vez creadas, indicamos la directiva “appRedColor” a los enlaces del navegador:

```
<nav>
    <li>
        <a href=""><ul appRedColor>Inicio</ul></a>
        <a href="/formulario"><ul appRedColor>Formulario</ul></a>
        <a href="/curriculum-vitae"><ul appRedColor>CV</ul></a>
        <a href=""><ul appRedColor>Prueba</ul></a>
        <a href="/login"><ul appRedColor>LogIn</ul></a>
    </li>
</nav>
```

\src\app\components\nav\nav.component.html

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.

Al igual que el primer ejercicio, primero una directiva llamada “BorderColor” con el siguiente comando:

Ng g d directivas/BorderColor

Una vez creada, añadimos al constructor el elemento “elementRef” y como evento, añadimos “change”

```
constructor(
  private elementRef : ElementRef
) { }

@HostListener('change')
public onChange(){
  if ( this.elementRef.nativeElement.value == 0 ) { this.elementRef.nativeElement.style.border = '1px solid red' }
  else { this.elementRef.nativeElement.style.border = '1px solid black'}
}
```

\src\app\directivas\border-color.directive.ts

Para poder revisar si se cumple la condición, accedo al valor del input con la línea:

```
this.elementRef.nativeElement.value
```

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).

Primero creamos un div donde añadiremos los párrafos donde se mostrarán los valores de los datos y también, dentro del div, añadiremos una propiedad para validar que se hayan rellenado todos los campos.

```
<div id="datos" *ngIf="validate">
    <p *ngIf="datos.nombre">Nombre : {{datos.nombre}}</p>
    <p *ngIf="datos.email">Email : {{datos.email}}</p>
    <p *ngIf="datos.telefono">Telefono : {{datos.telefono}}</p>
    <p *ngIf="datos.web">Sitio web : {{datos.web}}</p>
    <p *ngIf="datos.asunto">Asunto : {{datos.asunto}}</p>
</div>
```

src/components/formulario/formulario.component.html

También, dentro del mismo archivo, necesitaremos aplicar la propiedad [{ngModel}] para que se actualicen los valores de las variables conforme se escriben dentro de los inputs.

```
<label class="isRequired" for="nombreInput">Nombre</label>
<input type="text" id="nombreInput" required placeholder="Escribre tu nombre" [ngClass]="{'vacio': IsEmpty('nombreInput') == true}" [(ngModel)]="datos.nombre" name="name">

<label class="isRequired" for="emailInput">Email</label>
<input type="email" id="emailInput" required placeholder="Escribe tu Email" [ngClass]="{'vacio': IsEmpty('emailInput') == true}" [(ngModel)]="datos.email" name="email">

<label for="telefonoInput">Telefono</label>
<input type="tel" id="telefonoInput" placeholder="Escribre tu telefono" [ngClass]="{'vacio': IsEmpty('telefonoInput') == true}" [(ngModel)]="datos.telefono" name="telefono">

<label for="webInput">Sitio web</label>
<input type="url" id="webInput" placeholder="Escribe la URL de tu web" [ngClass]="{'vacio': IsEmpty('webInput') == true}" [(ngModel)]="datos.web" name="web">

<label class="isRequired" for="asuntoInput">Asunto</label>
<input type="text" id="asuntoInput" required placeholder="Escribe un asunto" [ngClass]="{'vacio': IsEmpty('asuntoInput') == true}" [(ngModel)]="datos.asunto" name="asunto">

<button (click)="validarDatos()">Validar datos</button>
```

src/components/formulario/formulario.component.html

Dentro del archivo formulario.component.ts, creamos la variable “datos” donde recogerá el valor introducido dentro de los inputs

```
public datos : any = {
  nombre : '',
  email : '',
  telefono : '',
  web : '',
  asunto : '',
}

public validate = false
```

src/components/formulario/formulario.component.ts

También creamos la función para validar todos los campos

```
public validarDatos(){
  console.log(this.datos.nombre.length)
  console.log(this.datos.email.length)
  console.log(this.datos.telefono.length)
  console.log(this.datos.web.length)
  console.log(this.datos.asunto.length)
  if (this.datos.nombre.length > 0 && this.datos.email.length > 0 && this.datos.telefono.length > 0 && this.datos.web.length > 0 && this.datos.asunto.length > 0) { this.validate = true}
}
```

src/components/formulario/formulario.component.html

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)

Creamos una nueva pagina llamada “Login” para introducir los dos campos y las validaciones.

ng g c pages/login

Una vez creado, modificamos el archivo HTML creando un form y añadiendo los ngModel correspondiente al mail y a la contraseña.

También añadimos un botón y le asociamos el evento (click)

```
<form method="POST" action="">

    <label for="emailInput">Email</label>
    <input type="email" name="emailInput" id="emailInput" [(ngModel)]="datos.email">

    <label for="passInput">Password</label>
    <input type="password" name="passInput" id="passInput" [(ngModel)]="datos.password">

    <button (click)="validate()">Log In</button>
```

src\app\pages\login\login.component.html

Para poder visualizar los datos, añadimos un div y le indicamos como condición que la variable “isValid” sea true

```
<div id="datos" *ngIf="isValid">
    <p *ngIf="datos.email">Email : {{datos.email}}</p>
    <p *ngIf="datos.password">Password : {{datos.password}}</p>
</div>
```

src\app\pages\login\login.component.html

Respecto a la validación, creamos una variable “datos” donde guardaremos el email y password. También crearemos la variable “isValid” y por defecto la inicializaremos a false

```
export class LoginComponent {
  public datos: any = {
    email: '',
    password: ''
  }

  isValid = false
```

src\app\pages\login\login.component.ts

Respecto a la validación del correo, hemos utilizado RegExp para comprobar que tienen el formato correcto

```
let regexp = new RegExp(/\S+@\S+\.\S+/)
```

Una vez tengamos el formato, solo necesitamos comprobar que las condiciones de ambos campos (mail y password) se cumplan.

```
public validate(){
  let regexp = new RegExp(/\S+@\S+\.\S+/)
  if (this.datos.password.length > 8 && regexp.test(this.datos.email) ) { this.isValid = true}
  console.log(this.datos.password)
  console.log(this.datos.password.length > 8)
  console.log(this.datos.email)
  console.log(regexp.test(this.datos.email))
}
```