46 - Directiva de atributo [ngStyle] y directivas para estilos individuales - Sintaxis completa

La directiva de atributo ngStyle actualiza los estilos para un elemento HTML determinado.

Establece una o más propiedades de estilo, especificadas como pares clave-valor separados por dos puntos. La clave es un nombre de estilo y el valor es una expresión a evaluar o valor a asignar.

Podemos crear un proyecto para ir probando estas funcionalidades:

```
ng new proyecto031
```

Si indicamos directamente el valor a asignar tenemos la siguiente sintaxis:

```
<h1 [ngStyle]="{'color':'red','background-color':'#ff0','text-align':'center'}">Sitio fuera de servicio</h1>
```

En el ejemplo vemos que si asignamos valores no presenta ninguna ventaja que definir la propiedad 'style' de HTML:

```
<h1 style="color:red;background-color:#ff0;text-align:center">Sitio fuera de servicio</h1>
```

Pero cuando utilizamos la directiva de atributo ngStyle podemos indicar una expresión en la zona del valor, por ejemplo si tenemos definido el atributo en la clase:

```
  colorEstado='#f00';
```

Luego en la vista accedemos a dicho atributo:

```
<h1 [ngStyle]="{'color':colorEstado}">Sitio fuera de servicio</h1>
```

Estamos indicando que para el atributo 'color' debe tomar el valor almacenado en la variable colorEstado, es importante notar que no debe ir entre comillas la variable.

Podemos llamar a un método en el lugar de la expresión, si tenemos definido en el modelo el método:

```
  retornarColor() {
    return '#00f';
  }
```

Luego cuando definimos la directiva tenemos la sintaxis:

```
<h1 [ngStyle]="{'color':retornarColor()}">Sitio fuera de servicio</h1>
```

Disponer una condición en la expresión

Por ejemplo si necesitamos que una tabla que muestra los números del 1 al 5, tengan las celdas de colores alternos, lo podemos resolver con la sintaxis:

```
<table>
  @for(elemento of [1,2,3,4,5];track $index) {
  <tr>
    <td [ngStyle]="{'background-color':elemento%2==0?'red':'yellow'}">{{elemento}}</td>
  </tr>
  }
</table>
```

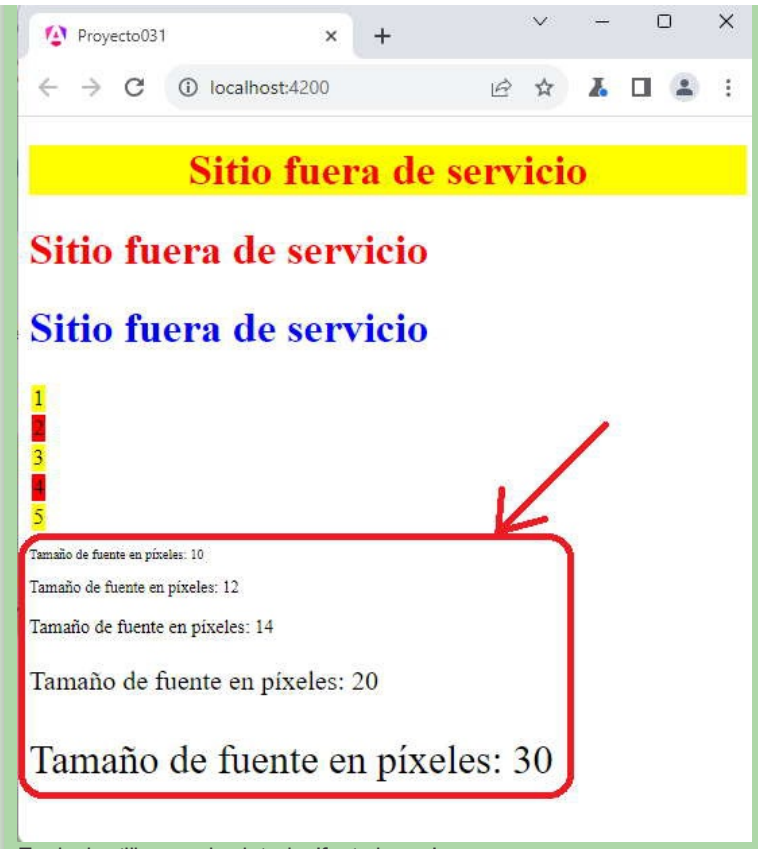
Cada vuelta del for se verifica si el resto de dividir el elemento del array es cero procede a asignar el valor 'red' a la propiedad 'background-color' en caso contrario asigna 'yellow'.

Trabajar con unidades de medida.

La sintaxis a emplear con unidades de medida se deben indicar como sufijo cuando la definimos:

```
@for(elemento of [10,12,14,20,30];track $index) {
  <p [ngStyle]="{'font-size.px':elemento}">
    Tamaño de fuente en píxeles: {{elemento}}
  </p>
}
```

Luego tenemos como resultado:



Es decir utilizamos la sintaxis: 'font-size.px'

Otros ejemplos de unidades de medida asignada podrían ser:

- 'margin-top.px':'10'
- 'padding.em':'1'

Asignar a la directiva un objeto con un conjunto de estilos

Podemos crear en el modelo un objeto literal con distintos estilos, que luego podemos modificar cuando se producen eventos. Para probar esto crearemos un div con una etiqueta y dos botones que modifican el tamaño de la fuente.

En el modelo de la componente definimos 'app.component.ts':

```
import { Component } from '@angular/core';
import { RouterOutlet } from '@angular/router';
import { NgStyle } from '@angular/common';

@Component({
  selector: 'app-root',
  imports: [RouterOutlet, NgStyle],
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  colorEstado = '#f00';

  retornarColor() {
    return '#00f';
  }

  tamano = 30;
  presentacion = {
    "background-color": "black",
    "color": "white",
    "width.px": "1000",
    "height.px": "200",
    "font-size.px": this.tamano,
    "display": "flex",
    "justify-content": "center",
    "align-items": "center"
  }

  agrandar() {
    this.tamano++;
    this.presentacion["font-size.px"] = this.tamano;
  }

  reducir() {
    this.tamano--;
    this.presentacion["font-size.px"] = this.tamano;
  }
}
```

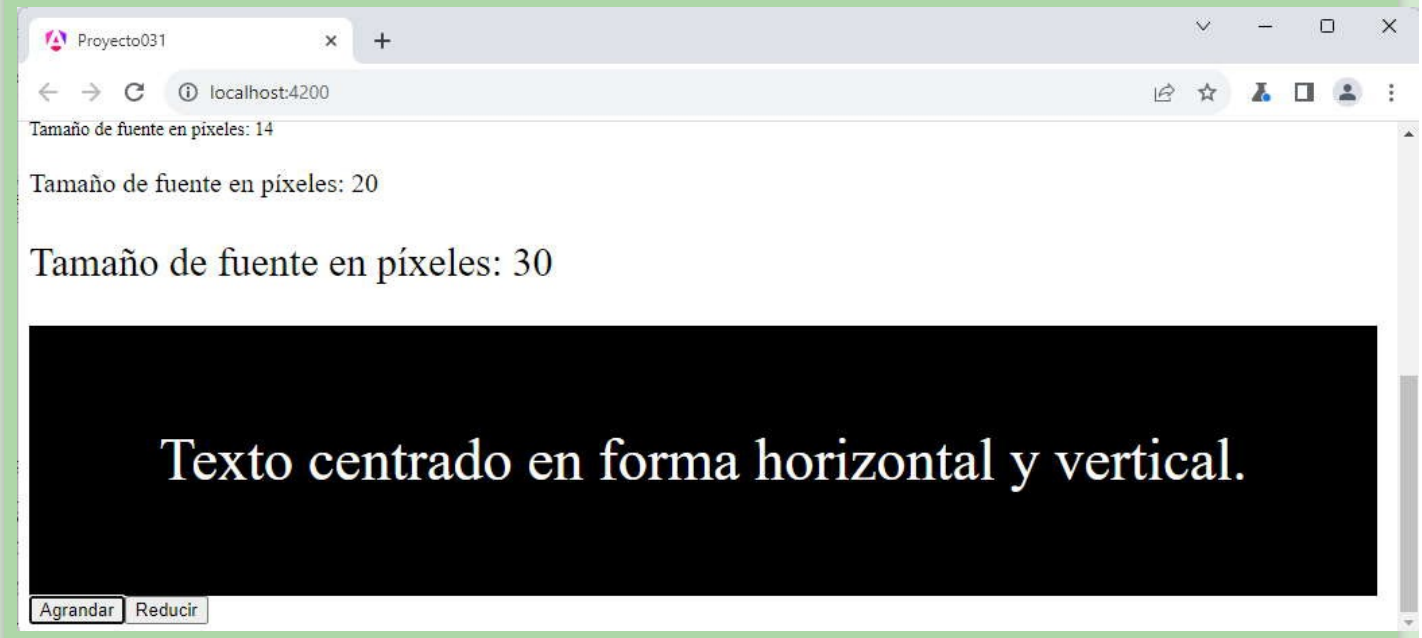
En la vista tenemos 'app.component.html':

```
<h1 [ngStyle]="{'color':'red','background-color':'#ff0','text-align':'center'}">Sit
io fuera de servicio</h1>
<h1 [ngStyle]="{'color':colorEstado}">Sitio fuera de servicio</h1>
<h1 [ngStyle]="{'color':retornarColor()}">Sitio fuera de servicio</h1>
<table>
  @for(elemento of [1,2,3,4,5];track $index) {
  <tr>
    <td [ngStyle]="{'background-color':elemento%2==0?'red':'yellow'}">{{elemento}}<
/td>
  </tr>
  }
</table>
@for(elemento of [10,12,14,20,30];track $index) {
  <p [ngStyle]="{'font-size.px':elemento}">
    Tamaño de fuente en píxeles: {{elemento}}
  </p>
}

<div [ngStyle]="presentacion">Texto centrado en forma horizontal y vertical.</div>
<button (click)="agrandar()">Agrandar</button>
<button (click)="reducir()">Reducir</button>
<router-outlet />
```

Hemos asignado a la directiva ngStyle la variable 'presentación' que se define en el modelo.

Cuando se presiona alguno de los botones procedemos a actualizar la variable 'presentacion' y podemos ver que el texto se actualiza.



Sintaxis para directivas individuales

Podemos hacer referencia a una propiedad individual con la sintaxis:

```
<p [style.color]="'#f00'">Color rojo</p>
```

Disponemos 'style' y seguidamente el nombre de la propiedad, en este caso 'color'. Son importante las comillas simples dentro de las comillas dobles, estamos indicando que tomo el string '#f00'.

Es más común que enlacemos la directiva con una variable definida en el modelo, por ejemplo si tenemos en el modelo:

```
  colorFondo='#ff0';
```

Luego en la vista enlazamos esta variable con la sintaxis:

```
<p [style.background-color]="colorFondo">Color</p>
```

Agregar a la aplicación anterior la posibilidad de modificar el tamaño de la fuente de un texto mediante dos botones.

En el archivo 'app.component.ts' tenemos:

```
import { Component } from '@angular/core';
import { RouterOutlet } from '@angular/router';
import { NgStyle } from '@angular/common';

@Component({
  selector: 'app-root',
  imports: [RouterOutlet, NgStyle],
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  colorEstado = '#f00';

  retornarColor() {
    return '#00f';
  }

  tamano = 30;
  presentacion = {
    "background-color": "black",
    "color": "white",
    "width.px": "1000",
    "height.px": "200",
    "font-size.px": this.tamano,
    "display": "flex",
    "justify-content": "center",
    "align-items": "center"
  }

  agrandar() {
    this.tamano++;
    this.presentacion["font-size.px"] = this.tamano;
  }

  reducir() {
    this.tamano--;
    this.presentacion["font-size.px"] = this.tamano;
  }

  tamanoFuente = 50;

  agrandarFuente() {
    this.tamanoFuente++;
  }

  reducirFuente() {
    this.tamanoFuente--;
  }
}
```

En el archivo 'app.component.html' queda:

```
<h1 [ngStyle]="{'color':'red','background-color':'#ff0','text-align':'center'}">Sit
io fuera de servicio</h1>
<h1 [ngStyle]="{'color':colorEstado}">Sitio fuera de servicio</h1>
<h1 [ngStyle]="{'color':retornarColor()}">Sitio fuera de servicio</h1>
<table>
  @for(elemento of [1,2,3,4,5];track $index) {
  <tr>
    <td [ngStyle]="{'background-color':elemento%2==0?'red':'yellow'}">{{elemento}}<
/td>
  </tr>
  }
</table>
@for(elemento of [10,12,14,20,30];track $index) {
  <p [ngStyle]="{'font-size.px':elemento}">
    Tamaño de fuente en píxeles: {{elemento}}
  </p>
}

<div [ngStyle]="presentacion">Texto centrado en forma horizontal y vertical.</div>
<button (click)="agrandar()">Agrandar</button>
<button (click)="reducir()">Reducir</button>

<div [style.font-size.px]="tamanoFuente">Texto</div>
<button (click)="agrandarFuente()">Agrandar</button>
<button (click)="reducirFuente()">Reducir</button>
<router-outlet />
```

Como vemos si tenemos que trabajar con unidades de medida debemos indicarla en la misma directiva:

```
<div [style.font-size.px]="tamano">Texto</div>
```

Podemos probar esta aplicación en la web aquí.

Acotaciones

Debemos usar NgStyle cuando debemos establecer muchos estilos en línea de forma simultánea y dinámica, según el estado del componente, en el caso que solo debemos establecer un único estilo conviene utilizar la sintaxis que acabamos de ver.

Retornar