10 - Componentes: disparo de eventos de la componente hija a la componente padre

En el concepto anterior vimos al final como podemos pasar un dato mediante el 'property binding' (la variable 'valor1' se encuentra definida en la clase):

```
<app-dado [valor]="valor1"></app-dado>
```

Ahora veremos como podemos captura un evento en la componente padre que emite la componente hija:

```
<app-cronometro [inicio]="15" (multiplo10)="actualizar($event)"></app-cronometro>
```

En esta componente tenemos una propiedad llamada inicio que le enviamos un dato y capturamos un evento llamado 'multiplo10' que emite la componente app-cronometro.

Problema

Confeccionar una aplicación con dos componentes llamadas 'AppComponent' y 'CronometroComponent'. La componente 'CronometroComponent' muestra un cronómetro que se actualiza cada un segundo, cada vez que su valor es múltiplo de 10 informa a la componente padre de dicha situación informando el segundo actual.

La componente 'AppComponent' define un cronómetro e informa cada vez que el cronómetro tiene un valor múltiplo de 10.

1. Desde la línea de comandos de Node.js procedemos a crear el proyecto004:

   ```
   c:\angulardevya> ng new proyecto004
   ```

2. Primero descendemos a la carpeta proyecto004 y nuevamente desde la línea de comandos procedemos a crear la componente 'cronometro' escribiendo:

   ```
   c:\angulardevya\proyecto004> ng generate component cronometro
   ```

   Recordemos que al ejecutar este comando se crean 4 archivos.

   Además dentro de la carpeta 'app' se crea una carpeta llamada 'cronometro' y dentro de ella se localizan los cuatro archivos creados.

3. En nuestro tercer paso vamos a implementar la vista de la componente 'cronometro' y su modelo. Abrimos el archivo 'cronometro.component.ts' y codificamos:

```
import { Component, Input, Output, EventEmitter } from '@angular/c
ore';

@Component({
  selector: 'app-cronometro',
  imports: [],
  templateUrl: './cronometro.component.html',
  styleUrl: './cronometro.component.css'
})
export class CronometroComponent {
  segundo = 0;
  @Input() inicio: number = 0;
  @Output() multiplo10 = new EventEmitter<number>();

  ngOnInit() {
    this.segundo = this.inicio;
    setInterval(() => {
      this.segundo++;
      if (this.segundo % 10 == 0)
        this.multiplo10.emit(this.segundo);
    }, 1000);
  }
}
```

En la clase CronometroComponent podemos identificar la sintaxis para definir un evento que dispara un valor de tipo 'number':

```
  @Output() multiplo10 = new EventEmitter<number>();
```

Para definir los decoradores @Input(), @Output() debemos importar:

```
import { Component, Input, Output, EventEmitter } from '@angular/core';
```

También hay que importar la clase EventEmitter.

Aparece algo nuevo que es el método 'ngOnInit' que se ejecuta una vez que se han inicializado todos los atributos. En el mismo, inicializamos el atributo 'segundo' con el valor del decorador 'inicio'.

Mediante la función setInterval, especificamos que cada 1 segundo (1000 milisegundos), se ejecute la arrow function donde incrementamos el atributo 'segundo' y si el mismo es múltiplo de 10 procedemos a llamar a la función de la otra clase.

4. Codificamos ahora el archivo 'cronometro.component.html':

```
<div class="cronometro">
  {{segundo}} Seg.
</div>
```

5. Para definir la hoja de estilo del 'cronometro' abrimos el archivo 'cronometro.component.css' y codificamos:

```
.cronometro {
  width: 8rem;
  height: 3rem;
  font-size: 2rem;
  color:white;
  background-color: black;
  border-radius: 10px;
  display: inline-flex;
  justify-content: center;
  align-items: center;
  margin:10px;
}
```

6. Abrimos ahora el archivo 'app.component.html' y remplazamos su contenido con la definición de un cronometro y un mensaje que se muestra mediante interpolación:

```
<div style="text-align:center">
  <h1>Prueba de la componente cronometro</h1>
  <app-cronometro [inicio]="15" (multiplo10)="actualizar($event)">
</app-cronometro>
  <h2>Evento</h2>
  <h3>{{mensaje}}</h3>
</div>

<router-outlet />
```

   Es importante entender la sintaxis del evento 'multiplo10' donde se llama al método actualizar.

   El $event es una variable especial en Angular que representa el objeto del evento que se está produciendo, en nuestro caso es el valor que se pasa desde el cronómetro del segundo actual:

```
        this.multiplo10.emit(this.segundo);
```

7. Ahora codificamos la clase AppComponent donde definimos el método que captura el evento emitido por el cronómetro:

```
import { Component } from '@angular/core';
import { RouterOutlet } from '@angular/router';
import { CronometroComponent } from './cronometro/cronometro.compo
nent';

@Component({
  selector: 'app-root',
  imports: [RouterOutlet, CronometroComponent],
  templateUrl: './app.component.html',
  styleUrl: './app.component.css'
})
export class AppComponent {
  mensaje = '';

  actualizar(t: number) {
    this.mensaje = t + '(se actualiza cada 10 segundos)';
  }
}
```

El método 'actualizar' se llama cuando se dispara el evento 'multiplo10':

```
<app-cronometro [inicio]="15" (multiplo10)="actualizar($event)"></app-cronometro>
```

El parámetro 't' del método recibe el valor de la variable especial de Angular llamada $event.

Si ejecutamos ahora el proyecto:

```
ng server -o
```

Podemos ver que cada vez que el cronómetro tiene un valor múltiplo de 10 la componente principal actualiza un mensaje gracias al evento que emite la componente 'cronometro':

Podemos probar esta aplicación en la web aquí.

Retornar