Angular

Comunicación entre componentes

Problemas

- Crear un componente para el input del número de cervezas
- Comunicar la lista con el carrito
  - (Próxima clase)


MakeAGIF.com

Hacer componentes atómicos

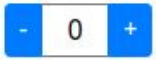
De la documentación oficial

Para comunicar componentes, Angular propone varias formas:

- **@Input:** binding de una propiedad del componente
- **@Output**: hookearse a los eventos de otro componente
- Comunicación por medio de un servicio.

Nota: Para más info [Angular.io - Component Interaction]

Vamos a hacer un componente para el input de stock

Refactor time!!! 

Pasos:

1. Creamos el nuevo componente
2. Movemos el código al nuevo componente y lo usamos en el template original
3. Recibimos (se indica con @Input()) el parámetro que necesitemos (o @Output()) si es de salida

Nota: al igual que con las funciones, si pasamos un objeto y lo modificamos se modifica afuera porque es el mismo objeto

Input / Output

Sirven para poder dividir un componente en sub-componentes y pasar datos de los componentes más grandes a los componentes más chicos.

https://angular.io/guide/component-interaction

Ejemplo

Después

```
<td>
  <app-input-number [beer]="beer"></app-input-number>
</td>
```

```
<td>
  <div class="input-group" *ngIf="beer.stock">
    <div class="input-group">
      <div class="input-group-prepend">
        <button class="btn btn-primary btn-sm" (click)="downQuantity(beer)">-</button>
      </div>
      <input type="text" class="text-center" [(ngModel)]="beer.quantity" (keyup)="changeQuantity($event,
beer)">
      <div class="input-group-append">
        <button class="btn btn-primary btn-sm" (click)="upQuantity(beer)">+</button>
      </div>
    </div>
    <br>
  </div>
</td>
```

ANTES

Compoentenes
ATOMICOS

Es realmente atómico este componente?
Es reutilizable?

@Output() y Eventos

- El decorador **@Output** nos deja indicar datos de salida
- Esta directiva se puede aplicar a “EventEmitter”s que nos dejan generar nuestros propios eventos
- En el caso que usemos el mismo nombre que el input más “Change” podemos construir nuestros propios two-way data binding [()]

Idea!

El campo de input numérico podríamos reutilizarlo en otros lugares que no sea con cervezas en el futuro

Un componente de input numérico:

- No debería necesitar un “beer”
- deberia pasar solamente el “quantity”

EventEmitter

El “evento custom” se llama así por falta de imaginación

```
@Output() quantityChange: EventEmitter<number> = new EventEmitter<number>();
@Output() custom: EventEmitter<number> = new EventEmitter<number>();

downQuantity(): void {
  ...
  this.quantityChange.emit(this.quantity);
  this.custom.emit(5);
}
```

Se conecta con 2-way data binding (varChange)

IMPORTANTE: El componente debe emitir los cambios

$event es el 5 que se pasa en el “emmit”

```
<app-input-number [(quantity)]="beer.quantity" (custom)="doIt($event);"></app-input-number>
```

EventEmitter

El “evento custom” se llama así por falta de imaginación

```
@Output() quantityChange: EventEmitter<number> = new EventEmitter<number>();
@Output() custom: EventEmitter<number> = new EventEmitter<number>();

downQuantity(): void {
  ...
  this.quantityChange.emit(this
  this.custom.emit(5);
}
```

Se conecta con 2-way data binding (varChange)

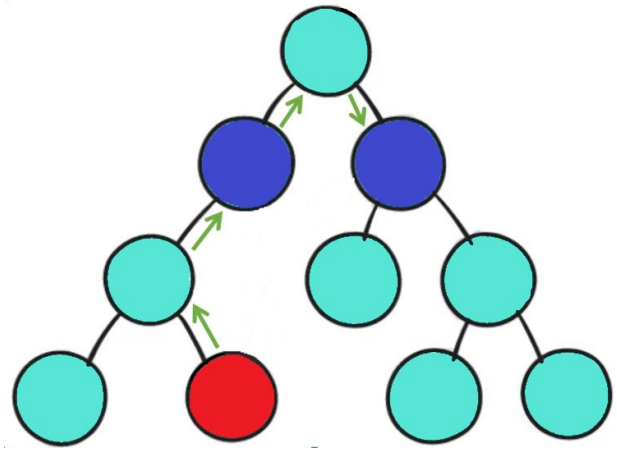
Si el @Input se llama “**algo**”, el output se tiene que llamar SI O SI “**algoChange**” para que sea two-way data-binding [()].

Sino es un event emmiter solo desconectado del otro (comparar los dos event emmmiters como se usan diferente)

```
<app-input-number [(quantity)]="beer.quantity" (custom)="doIt($event);"></app-input-number>
```

Qué pasa si tenemos componentes “lejanos” que comparten datos?

Limitaciones



Flujo de información desde el nodo rojo hacia los azules.

Si tenemos algunos datos generados por un componente muy lejano (en el DOM) a otro que quiere leer esa información este mecanismo haría que todos los componentes intermedios queden acoplados a pasar esa información.

Además, puede generar mal funcionamiento por la cantidad de intermediarios que pueden modificar la información.