Angular

Passer de l’information entre component PARENT et ENFANT avec les directives ...

@Intput()
@Output()

Recevoir l’information avec @Input()

1. ***Indiquer au component enfant*** qu’il reçoit une propriété nommée

   ```
   @Input() list ;
   ```

2. ***Transmettre la valeur de cette propriété*** depuis le HTML du ***parent***

   ```
   <app-enfant [list]=”tasks”></app-enfant>
   ```

(documentation : https://angular.io/guide/inputs-outputs)

Transmettre l’information avec @output()

1. ***Le component enfant*** peut émettre un événement personnalisé

```
@Output() public customEvent = new EventEmitter<any> ;

this.customEvent.emit(‘Hello’);
```

2. ***Le component parent*** peut alors réagir à cet évènement personnalisé

```
<app-parent (customEvent)=”SayHello($event)”></app-parent>
```

(documentation : https://angular.io/guide/inputs-outputs)

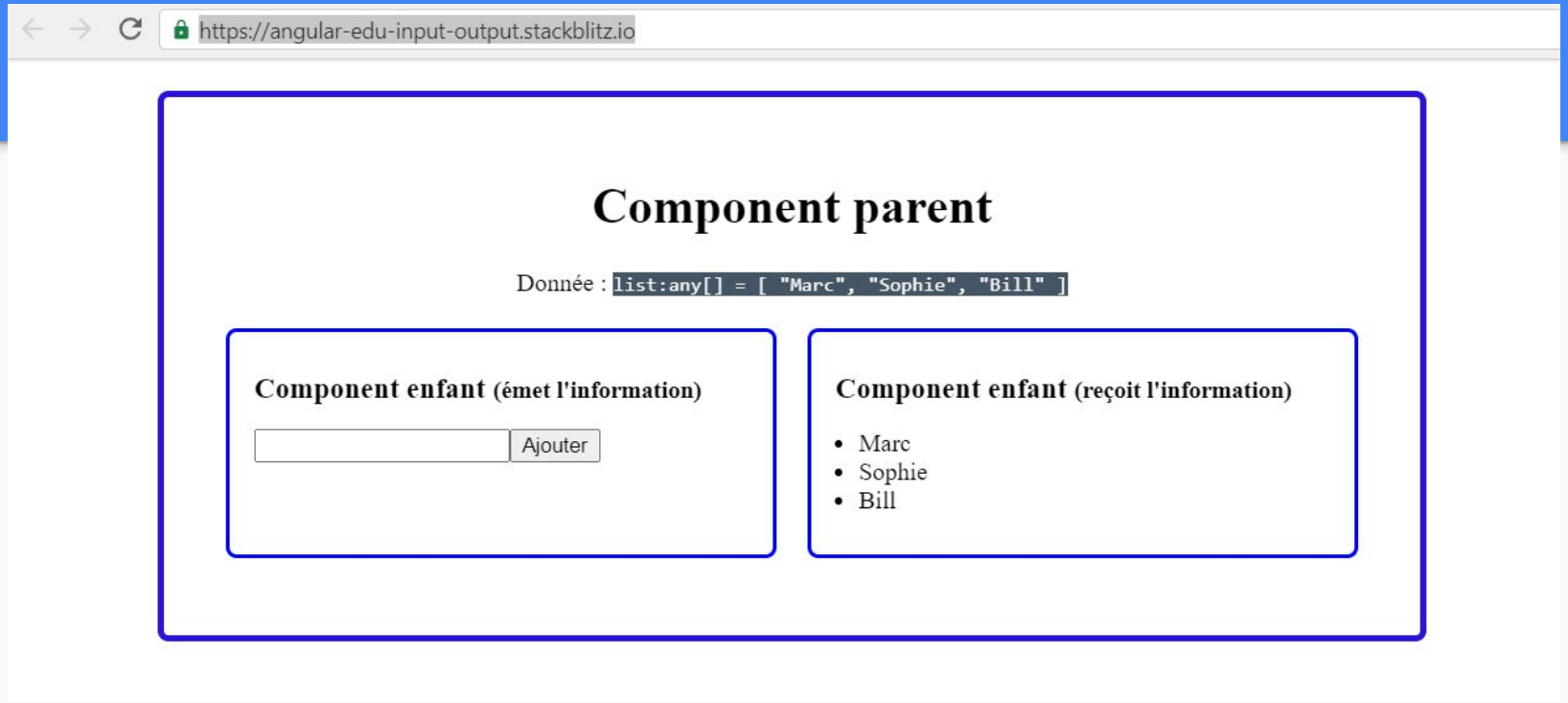
Exemple avec @Input() et @Output() (partie 1)

- ***Un component parent*** contient une donnée nommée ***list***
- **rôle du component : maintenir la donnée à jour**

- ***Un premier component enfant*** reçoit cette donnée ***@Input() list;***
- **rôle du component : afficher la liste en HTML**

- ***Un second component enfant*** contient un formulaire d'ajout, et émet la donnée
- **rôle du component : transmettre la donnée au parent *@Output myEvent;***

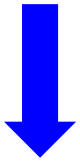
Exemple avec @Input() et @Output() (partie 2)



AppComponent (parent)

<app-form (**myEvent**)=”store($event)”></app-form>

<app-todolist [**list**]=”todolist"></app-todolist>



FormComponent (enfant)

```
@Output() myEvent = new EventEmitter<any>();


submitForm(contact) {
   this.myEvent.emit(contact);
}
```

ListComponent (enfant)

```
@Input() list;
```

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