esprit
Se former autrement
ONORIS UNITED UNIVERSITIES

Interaction entre composant parent et composant fils

United Nations Educational, Scientific and Cultural Organization
UNESCO Chair "Project-based learning" ESPRIT School of engineering, Tunisia
EUR-ACE®
Délivrée par la Commission des Titres d'Ingénieur
CONFÉRENCE DES GRANDES ÉCOLES
cdio
CONCEIVE DESIGN IMPLEMENT OPERATE

- Introduction
- Envoie des informations du parent vers le fils
- Emission d’événement du fils vers le parent
- Accès du composant parent aux éléments de son composant fils

Introduction

Composant parent – composant fils

- Un composant peut appeler, au niveau de son template, un autre composant via son selector.

```
@Component({
  selector: 'app-child',
  templateUrl: './child.component.html',
  styleUrls: ['./child.component.css']
})
export class ChildComponent  {
…….
}
```

Composant fils

Template

< app-child> </ app-child>

Composant parent

Composant parent – composant fils

Un composant parent peut interagir avec son composant Fils via:

1. Des propriétés d'entrées et des propriétés de sorties appelées respectivement « input property » et « output property ».
2. Variable locale
3. Le décorateur @ViewChild()
4. Des services

RQ: Deux composants indépendants peuvent communiquer via les services

Envois des informations du parent vers le fils

Propriété d'entrée

- Un composant fils peut recevoir des informations depuis son composant parent, via une propriété d'entrée « @Input() » .

```
import { Input } from '@angular/core';
```

- Un composant fils peut avoir plusieurs propriétés d'entrées.
- La propriété d'entrée est définie au niveau du composant fils comme suit:

```
class childComponent{
@Input() property1: Type;
}
```

Propriété d’entrée

- La valeur de la propriété d’entrée est envoyée par le composant parent, depuis son template, vers le composant fils comme suit:

HTML du composant parent

```
<app-child [property1]="currentEtudiant"></app-child>
```

property1 : proriété du composant fils

currentEtudiant: une propriété du composant parent

Propriété d’entrée - ngOnChanges

- La méthode hook **ngOnChanges** est lancée automatiquement à chaque fois où la valeur de la propriété d’entrée est modifiée au niveau du composant parent.
- Le composant fils peut via cette méthode peut détecter, tracer et gérer les changements de la valeur de la propriété d’entrée

```
ngOnChanges(changes: SimpleChanges): void
```

Parameters

changes	SimpleChanges	The changed properties.

Returns

`void`

Propriété d'entrée - ngOnChanges

- Les modifications du composant parent mettent toujours à jour la valeur dans le composant fils même sans implémenter ngOnChanges.
- Le ngOnChanges ajoute l'avantage de suivre ces modifications avec la valeur précédente et actuelle.
- Exemple:

```
import {Input, SimpleChange, OnChanges } from '@angular/core';
…
export class childComponent implements  OnChanges {
List : string[];
@Input() property1 : string;
ngOnChanges(changes: SimpleChanges}) {
Console.log (changes);
//traitement
}}
```

Propriété d'entrée - ngOnChanges

- La classe **SimpleChange** décrit la propriété d'entrée et permet de suivre son état.

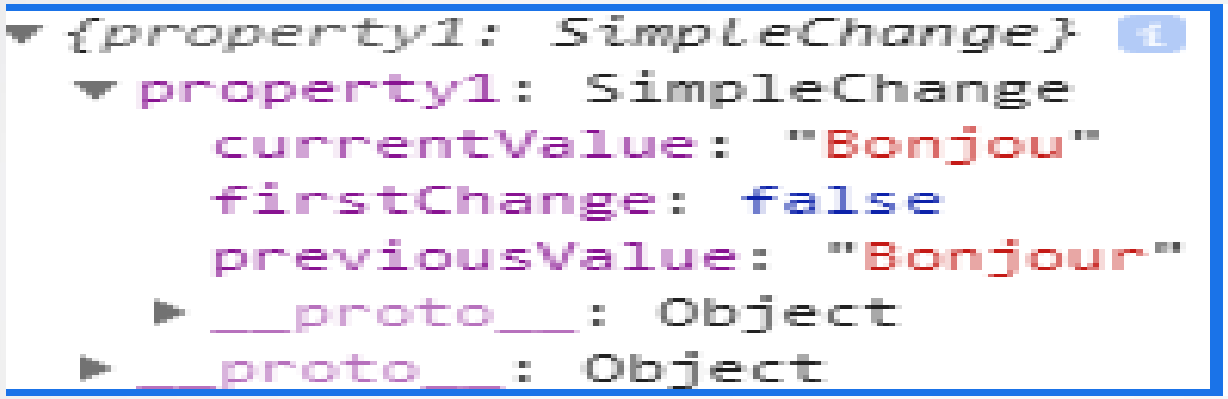
```
class SimpleChange {
  constructor(previousValue: any, currentValue: any, firstChange: boolean)
  previousValue: any
  currentValue: any
  firstChange: boolean
  isFirstChange(): boolean
}
```

- Soit la propriété d'entrée appelé **property1** ayant comme valeur initiale « Bonjour ».

```
▼ {property1: SimpleChange}
  ▼ property1: SimpleChange
      currentValue: "Bonjour"
      firstChange: true
      previousValue: undefined
    ▶ __proto__: Object
  ▶ __proto__: Object
```

Propriété d'entrée - ngOnChanges

- En modifiant la valeur de property1, en enlevant par exemple le « r », l'objet SimpleChange contient les informations suivante:



=> On peut tracer les différents changements d'une propriété d'entrée et détecter s'il s'agit du premier changement ou non

Propriété d’entrée - ngOnChanges

Si un composant fils possède plusieurs propriétés d’entrées alors « changes » est un objet où chaque attribut est de type SimpleChange et dont le nom est le nom de la propriété d’entrée.

Exemple:

```
account.component.ts:16
▼ Object i
  ▶ description: SimpleChange {previousValue: undefined, currentValue: 'An admin account has privileges to manag
  ▶ image: SimpleChange {previousValue: undefined, currentValue: '../assets/images/admin.png', firstChange: true
  ▶ title: SimpleChange {previousValue: undefined, currentValue: 'Admin Account', firstChange: true}
  ▶ [[Prototype]]: Object
```

Propriété d'entrée - Setter

Le composant fils modifie la valeur de la propriété d'entrée via le setter de cette propriété. Le setter permet d'intercepter **et** modifier la valeur de la propriété d'entrée.

Classe du ChildComponent

```
export class childComponent {
private prop1 = '';
@Input()
set property1(p: string) {
this.prop1 = p.ToUpperCase();}
get prop1(): string { return this.prop1; } }
```

HTML du ChildComponent

```
{{ prop1 }}
{{ property1 }}
```

HTML du ParentComponent

```
<app-child [property1]="my"></app-child>
```

Emission d’événement du fils vers le parent

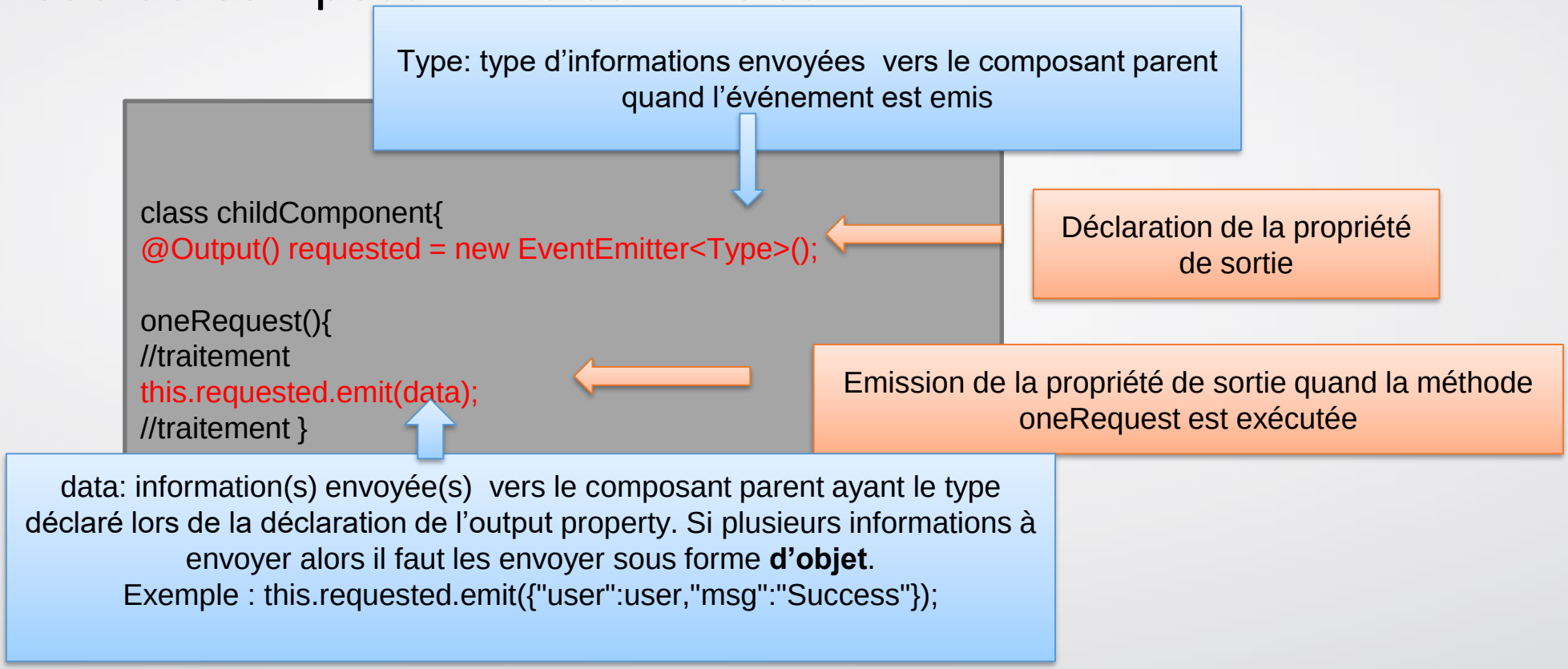
Propriété de sortie

- Le composant fils expose des émetteurs d'évènements «EventEmitter» dans certaines situations vers le composant parent.
- Le composant parent reste à l'écoute de son composant fils et interagit avec en cas d'un EventEmitter lancé.
- La propriété EventEmitter du fils est appelé « output property » et décrite par « @Output() »
- La propriété de sortie et EventEmitter à importer depuis **@angular/core**

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

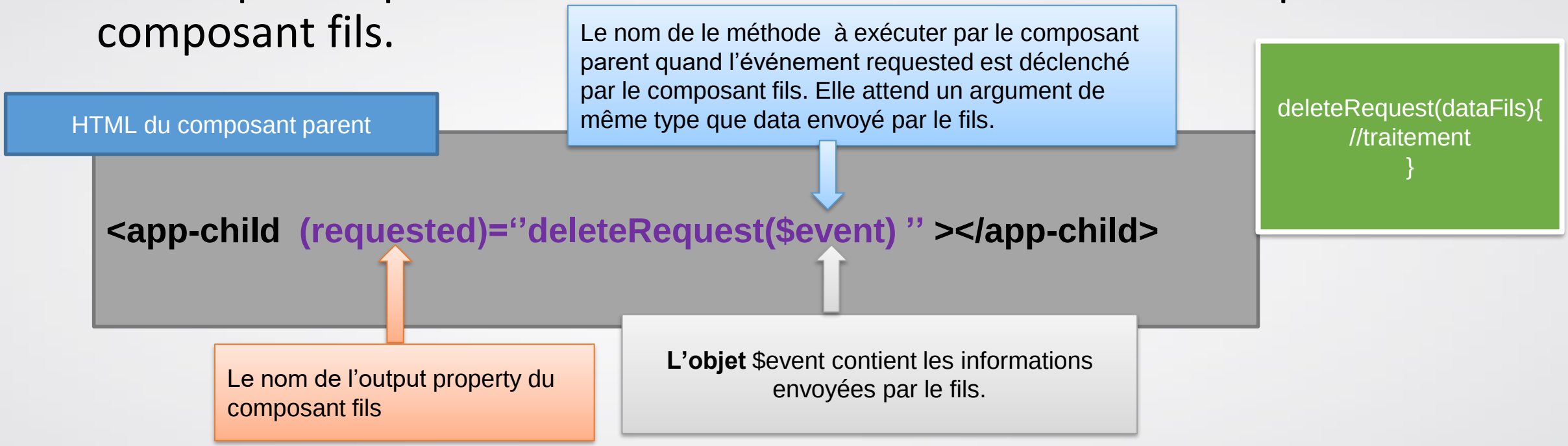
Propriété de sortie

- La déclaration de la propriété de sortie « output property », se fait au niveau du composant fils comme suit:



Propriété de sortie

- Le composant parent reste à l'écoute des évènements émis par le composant fils.



Exemple

1. Soit un composant parent appelé **homeComponent** qui appelle un composant fils appelé **rePrefixComponent**.

< app-re-prefix> </ app-re-prefix>

2. Le composant parent envoie vers le composant fils la valeur à laquelle le préfixe « re » sera rajouté

re-prefix.component.ts

```
@Input() initialValue : string;
```

home.component.html

```
Valeur: <input type="text" [(ngModel)]="firstVal">
<app-re-prefix [initialValue]="firstVal"></app-re-prefix>
```

Exemple

3 Le composant fils intercepte la valeur reçue et lui ajoute le préfixe « re » et 4 renvoie la nouvelle valeur vers le composant parent.

re-prefix.component.ts

```
@Output() prefixed = new EventEmitter<string>();

@Input()
set initialValue(val:string){
  this.prefixedValue="re"+val;

  this.prefixed.emit(this.prefixedValue);

}
```

Exemple

5 Une fois le composant parent reçoit la nouvelle valeur, il l'affiche.

home.component.ts

```
  newVal:string='';
getNewVal(val:string){
     this.newVal=val;
  }
```

home.component.html

```
<app-re-prefix [initialValue]="firstVal"
(prefixed)="getNewVal($event)"></app-re-prefix>

{{newVal}}
```

Accès du composant parent aux éléments De son composant fils

Accès parent fils

Le composant parent n’a pas le droit d’accéder aux propriétés et méthodes de son composant fils.Mais, cette restriction peut être résolue moyennant deux méthodes :

1. En utilisant les variables références de template (accès limité au niveau du template)
2. En utilisant le décorateur @ViewChild() (accès à partir de la classe du composant)

Accès parent fils via variable référence

- Pour accéder aux éléments du composant fils depuis le composant parent via les variables références de template, il faut créer une variable de template qui pointe sur le composant fils.

```
<app-child #child></app-child>
```

Au niveau du template du composant parent

```
<button   (click)="child.start()">Start</button>
<div>{{child.propertyA}}</div>
<app-child #child></app-child>
```

Au niveau du composant Fils

```
Class childComponent{
start(){…..}
propertyA : number = 10;

}
```

Accès parent fils via ViewChild()

Quand le composant parent veut lire ou écrire dans l'une des propriétés de son composant fils ou bien exécuter une action de son composant fils, ce dernier doit être injecté dans le composant parent avec ViewChild().

```
import { childComponent }  from './child.component';
@ViewChild(childComponent)
private exempleComponent : ChildComponent;
```

Accès parent fils via ViewChild()

Au niveau du composant Fils

```
Class childComponent{
start(){…..}
propertyA : number = 10;

}
```

Au niveau du template parent

```
 <button (click)=" startp()">
calculer</button>
  {{val}}
```

Au niveau de la classe du composant parent

```
import { childComponent }  from './child.component';
….
Class parentComponent{
….
val:number;
@ViewChild(childComponent)
  private childComponent : childComponent;
startp(){
    this.childComponent. start();
    this.val=this.childComponent.propertyA;
  }
…}
```

Accès parent fils via ViewChild()

- ViewChild() à importer depuis @angular/core.

```
import {ViewChild } from '@angular/core';
```

- Le composant parent accède aux méthodes/propriétés du composant fils à partir de la classe du composant.
- Le composant fils doit être appelé au niveau du template du composant parent.

=> Le composant fils devient disponible quand la vue du parent est initialisée.

Accès parent fils via ViewChild()

- Pour accéder au composant fils depuis le composant parent une fois la vue de ce dernier est initialisée, il faut s'assurer que la vue du parent a été bien initialisée.
- Pour ce faire, le composant parent doit implémenter l'interface **AfterViewInit** et la méthode **ngAfterViewInit.**

```
export class parentComponent implements AfterViewInit {
……
ngAfterViewInit() {
……..
};
```

Merci de votre attention