

Cours 6 : Frontend – en famille



Applications web et mobiles

Christophe Gonzales

Rajout d'un composant Course enfant de Courses (1/2)

The screenshot shows a Visual Studio Code editor with the following elements:

- File Explorer (Left):** Shows the project structure for `xmobile_cours`. The `src` folder is expanded to show `app` and `course` subfolders. The `course` folder contains `course.component.html`, `course.component.scss`, `course.component.spec.ts`, and `course.component.ts`.
- Code Editor (Center):** Displays the content of `course.component.html`, which is `<p>course works!</p>`.
- Terminal (Bottom):** Shows the execution of the command `ng g c course`. The output indicates that Node.js version v21.5.0 was detected and that odd-numbered versions are not recommended for production. It then lists the files created:
 - `CREATE src/app/course/course.component.scss (0 bytes)`
 - `CREATE src/app/course/course.component.html (21 bytes)`
 - `CREATE src/app/course/course.component.spec.ts (596 bytes)`
 - `CREATE src/app/course/course.component.ts (235 bytes)`

Rajout d'un composant Course enfant de Courses (2/2)

The image shows a code editor with the following HTML code for `courses.component.html`:

```
<nav>
  <a routerLink="/sujet/33">Mes sujets</a>
</nav>
<p>{{getTitle()}}</p>

<!-- ici, on rajoute 2 instances du nouveau composant course -->
<app-course></app-course>
<app-course></app-course>
<ul>
  <li *ngFor="let cours of UE">
    {{cours.nom}} : {{cours.nb_etuds}}
    <span *ngIf="cours.nb_etuds < 5">
    <span *ngIf="cours.nb_etuds >= 5">
  </li>
</ul>
```

The browser preview shows the rendered application with the following content:

Ceci est mon composant App

[Mes sujets](#)

liste de cours

course works!

course works!

- c1 : 2 étuds : (petit cours)
- c2 : 5 étuds : (gros cours)
- c3 : 6 étuds : (gros cours)

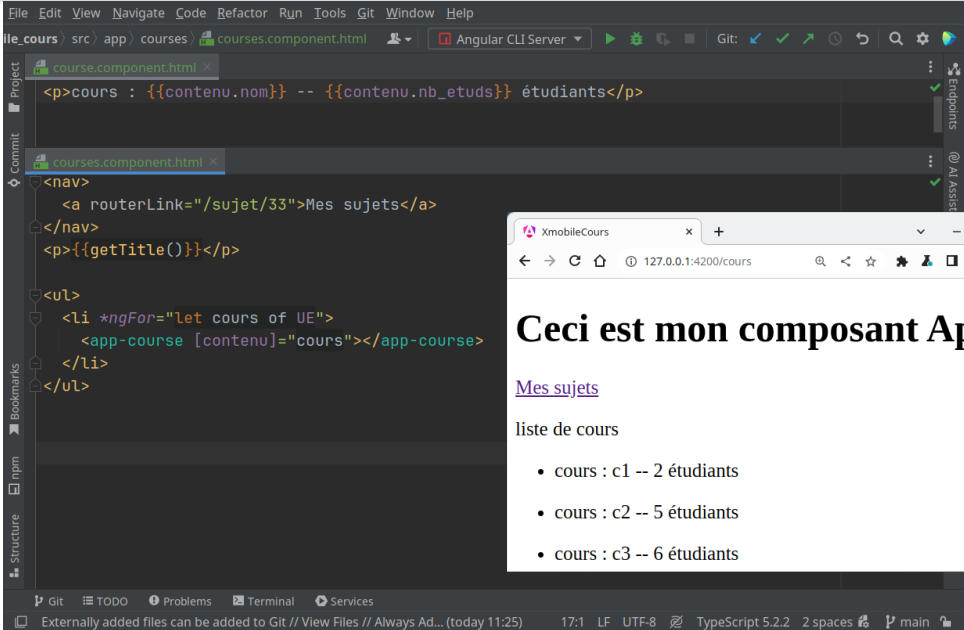
Passer des paramètres au constructeur de l'enfant (1/2)

```
File Edit View Navigate Code Refactor Run Tools Git Window Help
mobile_course > src > app > course > course.component.ts
Angular CLI Server
course.component.ts x
import {Component, Input} from '@angular/core';
import {Course} from "../services/courses.service";

@Component({
  selector: 'app-course',
  standalone: true,
  imports: [],
  templateUrl: './course.component.html',
  styleUrls: ['./course.component.scss']
})
export class CourseComponent {
  // Avec des @Input, on indique que, dans le HTML du parent, on va binder la
  // propriété "contenu" de <app-course>. Cela permettra à la classe
  // CourseComponent de recevoir du parent les informations dont elle a besoin
  // pour s'afficher correctement.
  @Input() contenu! : Course; // Ici, il est important de spécifier le type de
  // contenu car sa valeur n'étant pas précisée, il est impossible pour
  // le compilateur de connaître cette information.
  // Notez le ! qui permet, en mode strict de typescript, de préciser que
  // contenu n'est pas initialisé lors de sa création et que c'est normal.

  constructor() {}
}
```

Passer des paramètres au constructeur de l'enfant (2/2)



The image shows a development environment with a code editor on the left and a browser preview on the right. The code editor displays the following HTML code for a component:

```
<p>cours : {{contenu.nom}} -- {{contenu.nb_etuds}} étudiants</p>

<nav>
  <a routerLink="/sujet/33">Mes sujets</a>
</nav>
<p>{{getTitle()}}</p>

<ul>
  <li *ngFor="let cours of UE">
    <app-course [contenu]="cours"></app-course>
  </li>
</ul>
```

The browser preview shows the rendered output:

XmobileCours

127.0.0.1:4200/cours

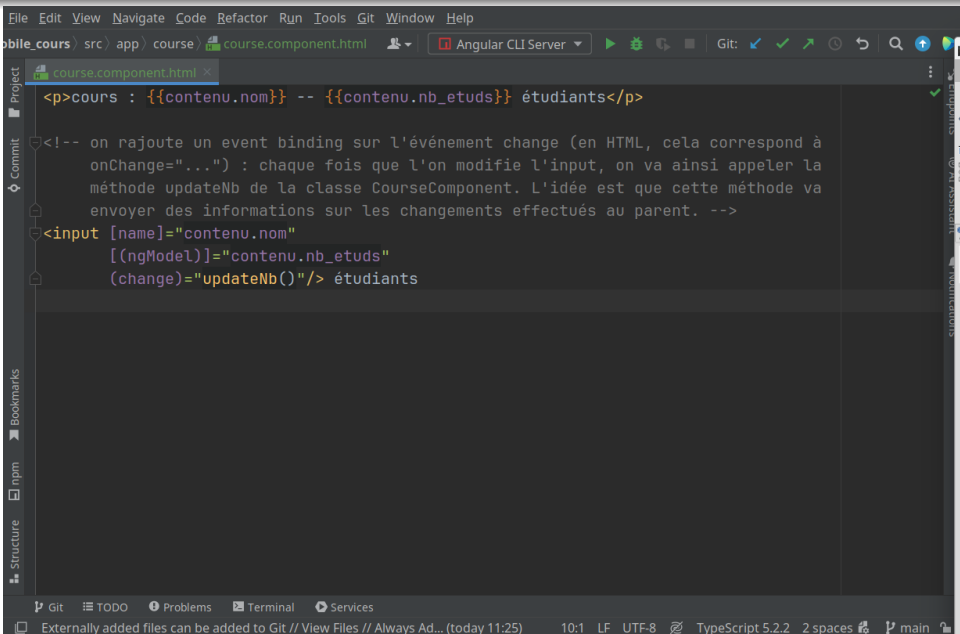
Ceci est mon composant App

[Mes sujets](#)

liste de cours

- cours : c1 -- 2 étudiants
- cours : c2 -- 5 étudiants
- cours : c3 -- 6 étudiants

Interactions enfant → parent : préparation de l'enfant



```
File Edit View Navigate Code Refactor Run Tools Git Window Help
course.component.html Angular CLI Server
course.component.html x
<p>cours : {{contenu.nom}} -- {{contenu.nb_etuds}} étudiants</p>
<!-- on rajoute un event binding sur l'événement change (en HTML, cela correspond à
onChange="...") : chaque fois que l'on modifie l'input, on va ainsi appeler la
méthode updateNb de la classe CourseComponent. L'idée est que cette méthode va
envoyer des informations sur les changements effectués au parent. -->
<input [name]="contenu.nom"
[(ngModel)]="contenu.nb_etuds"
(change)="updateNb()" /> étudiants
```

Git TODO Problems Terminal Services

Externally added files can be added to Git // View Files // Always Ad... (today 11:25) 10:1 LF UTF-8 TypeScript 5.2.2 2 spaces main

Interactions enfant → parent : l'émetteur de l'enfant

```
File Edit View Navigate Code Refactor Run Tools Git Window Help
course.component.ts > CourseComponent > updateNb() Angular CLI Server
course.component.ts x
@Component({
  selector: 'app-course',
  standalone: true,
  imports: [
    FormsModule
  ],
  templateUrl: './course.component.html',
  styleUrls: ['./course.component.scss']
})
export class CourseComponent implements OnInit {
  @Input() contenu!: Course; // infos parent -> enfant
  @Output() newNb = new EventEmitter<number>(); // infos enfant -> parent
  // ici, newNb va être un événement que l'enfant va déclencher et le parent écouter
  lastNb!: number; // va servir à calculer le nombre que l'on émet vers le parent

  constructor() {}
  ngOnInit() { this.lastNb = this.contenu.nb_etuds; }
  updateNb() { // méthode appelée quand on change l'input
    const nb = this.contenu.nb_etuds - this.lastNb;
    this.lastNb = this.contenu.nb_etuds;
    this.newNb.emit(nb); // Ici, on émet l'info qui sera récupérée par le parent
  }
}
```

Interactions enfant → parent : la réception du parent

The screenshot shows an IDE with the Angular CLI Server running. The code in `courses.component.html` is as follows:

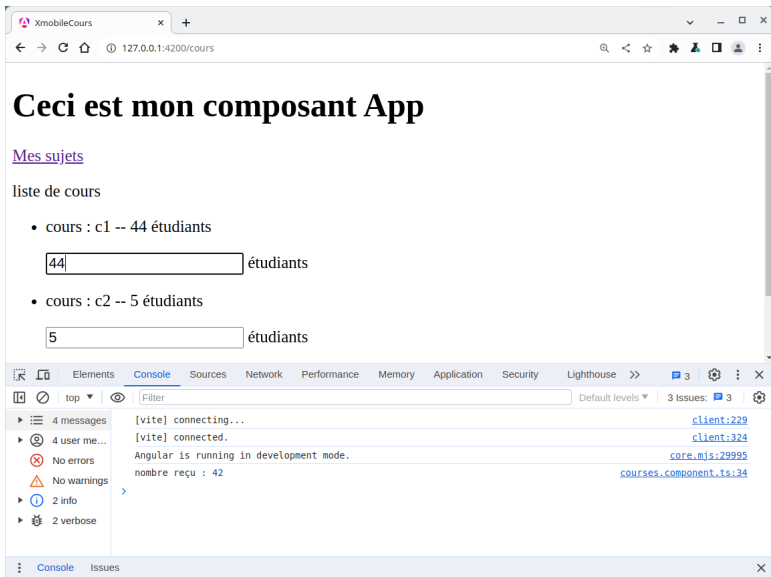
```
<!-- Par event binding, on écoute les événements newNb émis par l'enfant.  
Le nombre transmis par l'enfant se trouve dans $event. -->  
<app-course [contenu]="cours" (newNb)="onNewNb($event)"></app-course>  
</li>
```

The code in `courses.component.ts` is as follows:

```
export class CoursesComponent implements OnInit {  
  titre = 'liste de cours';  
  UE : Course[] = []; // on n'initialise avec une liste vide!  
  
  constructor(private service: CoursesService) {}  
  ngOnInit() {  
    this.service.getCourses().subscribe(  
      courses => {  
        this.UE = courses; // On récupère la liste des cours.  
        // Toutes les opérations dépendant de cette liste doivent être exécutées dans  
        // cette callback, pas à l'extérieur.  
      }  
    );  
  }  
  
  getTitle() { return this.titre; }  
  onNewNb(delta: number) { console.log('nombre reçu :', delta); } // callback quand newNb arrive
```

The IDE interface includes a menu bar (File, Edit, View, Navigate, Code, Refactor, Run, Tools, Git, Window, Help), a toolbar with icons for running and debugging, and a sidebar with Project, Commit, Bookmarks, and npm views. The status bar at the bottom shows Git, TODO, Problems, Terminal, Services, and file encoding information (UTF-8, TypeScript 5.2.2, 2 spaces).

Interactions enfant → parent : le résultat



The screenshot shows a web browser window with the URL `127.0.0.1:4200/cours`. The page content includes:

Ceci est mon composant App

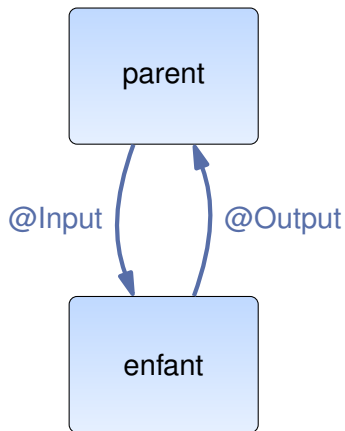
[Mes sujets](#)

liste de cours

- cours : c1 -- 44 étudiants
 étudiants
- cours : c2 -- 5 étudiants
 étudiants

The browser's developer console is open, showing the following messages:

```
[vite] connecting... client:229
[vite] connected. client:324
Angular is running in development mode. core.mjs:29995
nombre reçu : 42 courses.component.ts:34
```



Il existe d'autres types d'interactions (@ViewChild, etc.)