

# Cours 2 : Frontend



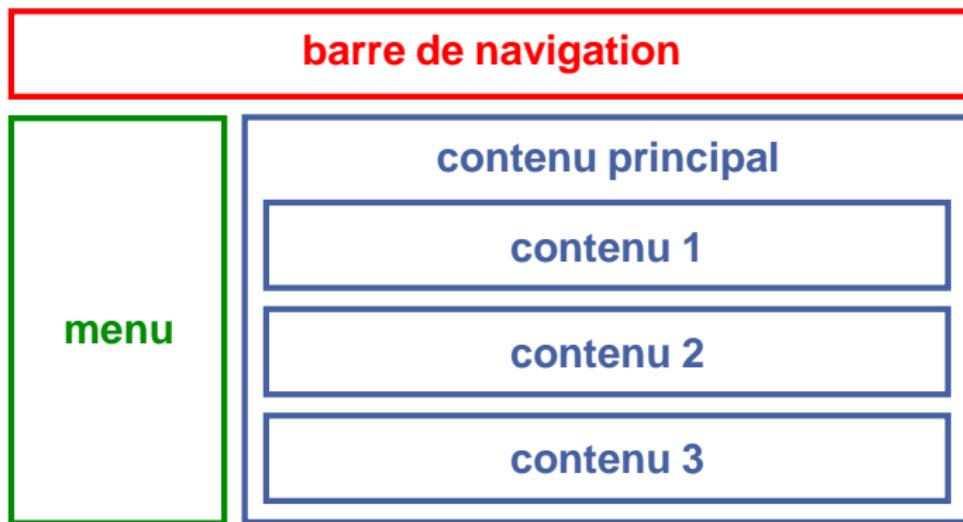
Web – Applications web et mobile

*By DhiaEddineSaied*

- ▶ Framework pour construire des applications clientes  
=⇒ front-end
- ▶ Structure l'application  
=⇒ simplifie programmation/maintenance/déboggage
- ▶ Mise en place de tests simple
- ▶ Utilise TypeScript/Javascript, HTML, CSS

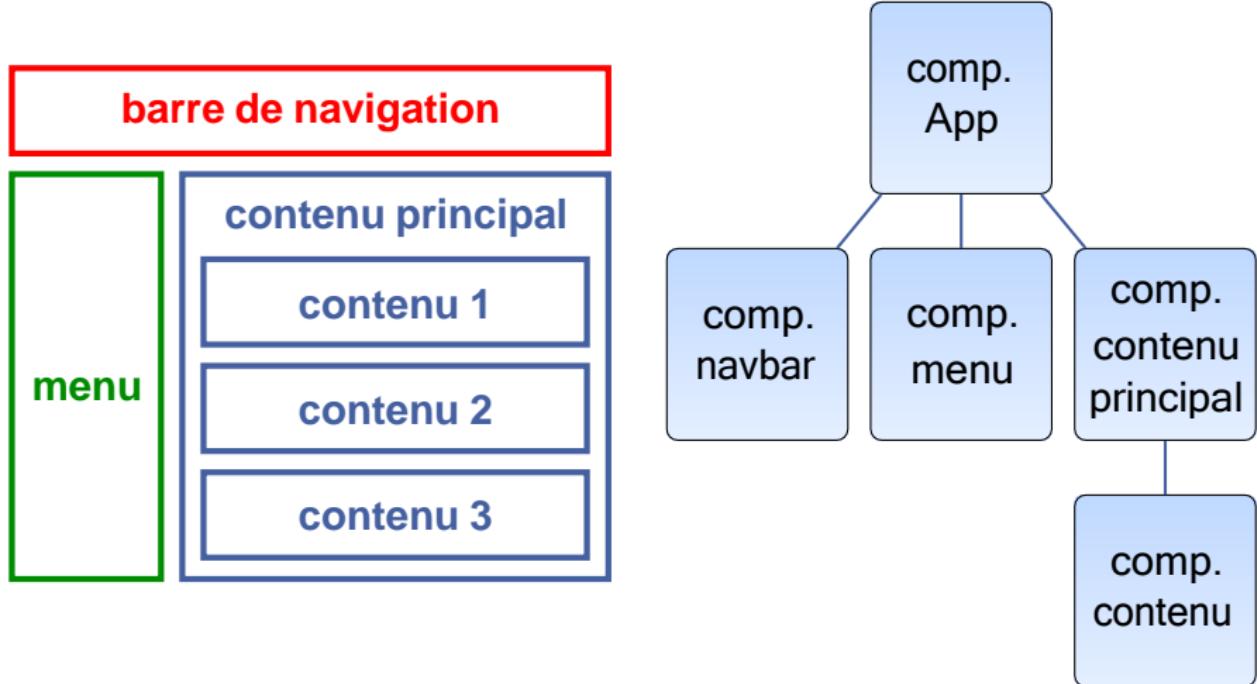
# Structure d'une application Angular

Affichage de la page web :



- ▶ Affichage  $\Rightarrow$  structure
- ▶ Chaque rectangle = composant Angular
- ▶ Intérêt des composants : réutilisables plusieurs fois
- ▶ Un composant peut en inclure d'autres

# Logique de l'application : arbre de composants



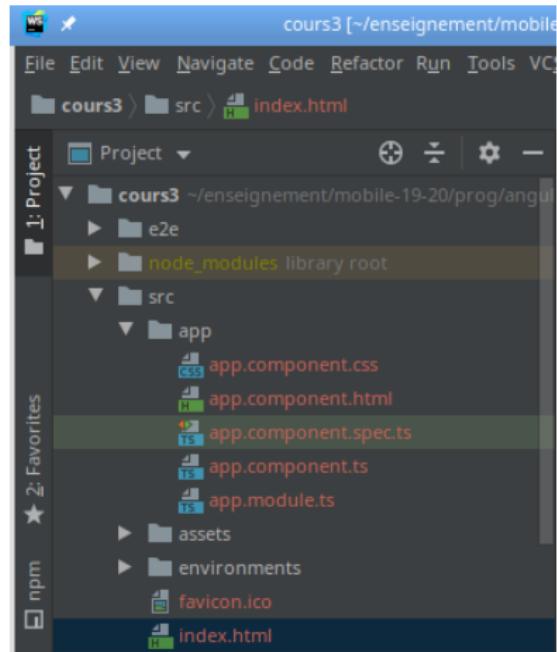
=⇒ permet de structurer facilement le code !

## Un composant Angular contient essentiellement :

- 1 un fichier TypeScript contenant :
  - ▶ les données du composant
  - ▶ la logique/le comportement du composant
- 2 un fichier html
  - ▶ contenant le code HTML affiché par le browser
  - ▶ des instructions pour interagir avec le code TypeScript
- 3 un fichier css contenant le style propre au composant
  - ▶ Répertoire src/app contient les composants
  - ▶ 1 composant principal appelé app ou root

# Génération d'un projet Angular

- ▶ `ng new mon-projet`  
⇒ crée le composant app :



- ▶ Dans src/app :
  - ▶ `app.component.ts` : code TypeScript
  - ▶ `app.component.spec.ts` : pour faire des tests
  - ▶ `app.component.html` : template HTML
- ▶ Dans src :
  - ▶ `index.html` : point d'entrée de l'appli

# Index.html

WS cours3 [~/enseignement/mobile-19-20/prog/angular/cours3] - .../src/index.html - WebStorm

File Edit View Navigate Code Refactor Run Tools VCS Window Help

Angular CLI Server

Project

cours3 ~/enseignement/mobile-19-20/prog/angular/cours3

1: Project

2: Favorites

3: npm

4: Structure

index.html

index.html

1 <!doctype html>  
2 <html lang="en">  
3 <head>  
4 <meta charset="utf-8">  
5 <title>Cours3</title>  
6 <base href="/">  
7 <meta name="viewport" content="width=device-width, initial-scale=1.0, shrink-to-fit=no">  
8 <link rel="icon" type="image/x-icon" href="favicon.ico">  
9 </head>  
10 <body>  
11 <app-root></app-root>  
12 </body>  
13 </html>

14

Insertion du composant App

app.component.css  
app.component.html  
app.component.spec.ts  
app.component.ts  
app.module.ts  
assets  
environments  
favicon.ico  
index.html  
main.ts  
polyfills.ts  
styles.css  
test.ts  
editorconfig

Terminal TypeScript 3.5.3 TODO

Event Log

WebStorm 2019.3.2 available: // Update... (today 3:52 PM)

14:1 LF UTF-8 EditorConfig Git: master

Cours 2 : Frontend Angular

7/57

- 1 Créer les composants (et les modules)
- 2 Les insérer dans l'application via des balises dans les fichiers HTML

**Exemple :** `<app-root></app-root>`

---

- ▶ Pour compiler et servir votre application :  
`ng serve`

# Le composant App et l'appli servie

The screenshot shows the WebStorm IDE interface with an Angular project named 'cours3'. The left sidebar displays the project structure:

- 1: Project
- 2: Favorites
- 3: npm
- 4: Structure

The 'src' folder contains the following files:

- app.component.css
- app.component.html
- app.component.spec.ts (selected)
- app.component.ts
- app.module.ts
- assets
- environments

The 'app.component.html' file is open in the main editor, showing the content:

```
<h1>Ceci est mon composant App</h1>
```

The 'Angular CLI Server' dropdown shows 'Running'. Below the editor are browser preview icons for Chrome, Firefox, and Opera.

A separate browser window titled 'Cours3 - Chromium' is open at 'localhost:4200', displaying the rendered content:

# Ceci est mon composant App

The terminal window shows the command being run:

```
[shalmeser:~/enseignement/mobile-19-20/prog/angular/cours3]<1> ng serve
```

Output from the terminal:

```
10% building 3/3 modules 0 active [wds]: Project is running at http://localhost:4200/webpack-dev-server/
[ wds]: webpack output is served from /
[ wds]: 404s will fallback to //index.html
```

Bottom status bar:

- Terminal
- TypeScript 3.5.3
- TODO
- Event Log

Bottom footer:

- WebStorm 2019.3.2 available: // Update... (today 3:52 PM)
- 2:1 LF UTF-8 EditorConfig Git: master
- 9/57

# Le TypeScript du composant app

The screenshot shows the WebStorm IDE interface with the following details:

- Title Bar:** cours3 [~/enseignement/mobile-19-20/prog/angular/cours3] - .../src/app/app.component.ts - WebStorm
- Menu Bar:** File Edit View Navigate Code Refactor Run Tools VCS Window Help
- Toolbars:** Angular CLI Server, Git
- Project Explorer (Left):** Shows the project structure with 'cours3' as the root, containing 'e2e', 'node\_modules', 'src' (which contains 'app' with files: 'app.component.css', 'app.component.html', 'app.component.spec.ts', 'app.component.ts', 'app.module.ts'), 'assets', 'environments', 'favicon.ico', 'index.html', 'main.ts', 'polyfills.ts', 'styles.css', 'test.ts', and 'editorconfig'.
- Code Editor (Right):** Displays the content of 'app.component.ts'. The code defines an Angular component named 'AppComponent' with the selector 'app-root'. It includes imports from '@angular/core', annotations for the component, and a template URL pointing to 'app.component.html'. The code is annotated with French comments explaining the purpose of each part.
- Bottom Status Bar:** Terminal, TypeScript 3.5.3, TODO, Event Log, and system status (WebStorm 2019.3.2 available, 19:42, LF, UTF-8, EditorConfig, Git: master).

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

// décorateur qui indique à Angular que la classe TypeScript
// en dessous est le code d'un composant
@Component({
    // indique la balise HTML à utiliser pour insérer le
    // composant dans l'application
    selector: 'app-root',
    // indique où se trouve le code HTML du composant (celui à
    // insérer dans l'appli quand on insère le composant
    templateUrl: './app.component.html',
    // les styles propres au composant
    styleUrls: ['./app.component.css']
})
export class AppComponent {
    // un composant contient une classe TypeScript qui sert
    // à contenir ses données et sa logique
}
```

# Création d'un nouveau composant

- ▶ Utiliser la commande `ng generate component courses`  
⇒ crée un répertoire `courses` et des fichiers spécifiques

The screenshot shows the WebStorm IDE interface with the following details:

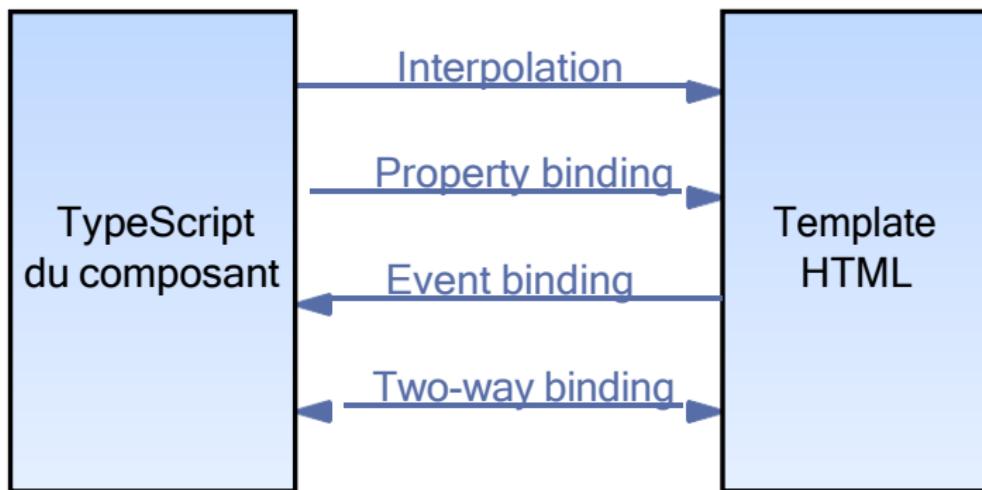
- Project Structure:** The project tree on the left shows the directory structure: `cours3` → `src` → `app`. Inside `app`, there are files: `app.component.html`, `index.html`, `app.component.ts`, and `app.component.html`. A new folder named `courses` has been created under `app`, containing files: `courses.component.css`, `courses.component.html`, `courses.component.spec.ts`, and `courses.component.ts`.
- Editor:** The main editor area displays the content of `app.component.html`: `<h1>Ceci est mon composant App</h1>`.
- Terminal:** The terminal at the bottom shows the command being run: `gonzales@shalmaneser:~/enseignement/mobile-19-20/prog/angular/cours3$ ng generate component courses`. The output of the command is displayed, showing the creation of four files in the `courses` folder.

# TypeScript du nouveau composant

The screenshot shows the WebStorm IDE interface with the following details:

- Project Structure:** On the left, the project structure is displayed. It includes the root folder "cours3", "e2e", "node\_modules", and "src". Under "src", there are "app" and "courses" folders. Inside "courses", files like "courses.component.css", "courses.component.html", "courses.component.spec.ts", and "courses.component.ts" are listed. Other files in "src/app" include "app.component.css", "app.component.html", "app.component.spec.ts", "app.component.ts", "app.module.ts", and "app-routing.module.ts". Additionally, "assets", "environments", and "test.ts" are shown.
- Code Editor:** The main editor area shows the content of "courses.component.ts". The code defines a component named "CoursesComponent" that implements the "OnInit" interface. It includes a constructor and an "ngOnInit()" method.
- Toolbars and Status Bar:** At the bottom, there are toolbars for Version Control, Terminal, TypeScript 3.5.3, TODO, and Event Log. The status bar at the very bottom provides information about the current file ("cours3 [~/enseignement/mobile-19-20/prog/angular/cours3] - .../src/app/courses/courses.component.ts - WebStorm") and the build system ("Angular CLI Server"). It also shows the file's last update ("TSLint: The project code style and editor settings were updated base... (today 08:15)"), file statistics ("22:1 LF"), encoding ("UTF-8"), and Git status ("Git: master").

# Interactions entre le TypeScript et le HTML



# Interpolation

The screenshot shows the WebStorm IDE interface with the following details:

- Project Structure:** The left sidebar shows the project structure with files like `courses.component.ts`, `app.component.html`, and `courses.component.html`.
- Code Editor:** The main editor area displays the `courses.component.ts` file, which defines a component with a selector `'app-courses'` and a template URL `'./courses.component.html'`. It also includes a constructor and a class variable `titre_` set to `'liste des cours'`. The `courses.component.html` file contains two `<app-courses>` tags.
- Preview:** A large preview window on the right shows the rendered application output:
  - The title "Ceci est mon composant App" is displayed.
  - Two instances of the `app-courses` component are shown, each displaying the interpolation result: "Interpolation : liste des cours et 7".
  - The bottom part of the preview shows the source code of the `courses.component.html` file with annotations explaining the interpolation syntax.
- Bottom Status Bar:** The status bar at the bottom provides information about the project's state, including version control, terminal, TypeScript version, TODO items, and an event log.

# Interpolation avec des méthodes

The screenshot shows the WebStorm IDE interface with the following details:

- Project Structure:** On the left, the project structure is visible with the following hierarchy:
  - cours3 (~/enseignement/mobile-19-20/prog/angular/cours3)
  - e2e
  - node\_modules
  - src
    - app
      - courses
        - courses.component.ts
        - courses.component.css
        - courses.component.html
        - courses.module.ts
        - courses.service.ts
      - app.component.ts
      - app.module.ts
      - app-routing.module.ts
    - assets
    - environments
      - favicon.ico
      - index.html
      - main.ts
      - polyfills.ts
      - styles.css
      - test.ts
- CoursesComponent.ts:** The code defines a component with a selector 'app-courses' and a template URL './courses.component.html'. It has a title 'liste des cours' and an ngOnInit() method. The getTitle() method returns the title.
- courses.component.html:** The template contains a single paragraph element with interpolation: <p> Interpolation bis : {{ getTitle() }}</p>
- Output:** A large callout box on the right displays the rendered output: **Ceci est mon composant App**. Below it, two lines of text explain the interpolation: "Interpolation bis : liste des cours" and "Interpolation bis : liste des cours".
- Bottom Status Bar:** Shows Version Control, Terminal, TypeScript 3.5.3, TODO, Event Log, and TSLint status.

# Property binding

The screenshot shows the WebStorm IDE interface with the following details:

- Project Structure:** On the left, the project structure shows the file tree: `cours3` (containing `src`, `e2e`, and `node\_modules`), `src` (containing `app` which has `courses` and `node\_modules`), and `app` (containing `courses.component.ts` and `courses.component.html`).
- Code Editor:** The main editor shows `courses.component.ts` with the following code:

```
1 import { Component } from '@angular/core';
2
3 @Component({
4   selector: 'app-courses', // balise d'insertion
5   templateUrl: './courses.component.html',
6   styleUrls: ['./courses.component.css']
7 })
8 export class CoursesComponent implements OnInit {
9   titre_: string = 'liste des cours';
10  ma_valeur : string = 'valeur initiale';
11
12  constructor() { }
13
14  ngOnInit() {
15  }
16}
```
- Preview Panel:** A large preview panel on the right displays the component's output. It contains the text "Ceci est mon composant App" and two sections below it:
  - "Interpolation : liste des cours et 7" with the value "valeur initiale" in a text input field.
  - "valeur initiale" in another text input field.
- Code Editor:** Below the preview panel, the code editor shows `courses.component.html` with the following content:

```
<p> Interpolation : {{ titre }} et {{ 3 + 4 }}</p>
<input type="text" [value]="ma_valeur" />
```
- Bottom Status Bar:** The status bar at the bottom provides information about the project and environment, including "TSLint: The project code style and editor settings were updated base... (today 08:15) 3:48 LF UTF-8 EditorConfig Git: master", "Event Log", and file navigation icons.

- ▶ **Interpolation {{}}** :

Permet de transférer des données du TypeScript n'importe où dans le template HTML

Évalué à runtime!

- ▶ **Property binding [] :**

Permet de mettre des valeurs dans les propriétés des éléments du DOM

**Exemple intéressant :** [hidden]="valeur"

# Event binding

The screenshot shows the WebStorm IDE interface with the following details:

- Project Structure:** On the left, the project structure shows files like `courses.component.ts`, `courses.component.html`, and `app.module.ts`.
- Code Editor:** The main editor shows the `courses.component.ts` file with the following code:

```
templateUrl: './courses.component.html',
styleUrls: ['./courses.component.css']
})
}

export class CoursesComponent implements OnInit {
titre = 'liste des cours';

constructor() { }
ngOnInit() {}

getTitle() { return this.titre; }

modifTitle() {
    this.titre = 'nouveau titre';
}
}
```
- Template Editor:** Below it, the `courses.component.html` template shows:

```
<p> Interpolation évaluée à runtime : {{ getTitle() }}</p>
<input type="text" (click)="modifTitle ()" />
```
- Output Area:** A large text box on the right displays the output of the application:

Ceci est mon composant

Interpolation évaluée à runtime : nouveau titre

Interpolation évaluée à runtime : liste des cours
- Bottom Status Bar:** Shows version control, terminal, todo items, and event log.
- Bottom Footer:** Shows the message "TSLint: The project code style and editor settings were updated bas..." and the date/time "today 08:15 14:36 LF UTF-8 EditorConfig Git: master".

# Two-way binding (1/2)

The screenshot shows the WebStorm IDE interface with the following details:

- Project Structure:** On the left, the project structure is visible with the file `courses.component.ts` open.
- Code Editor:** The main editor shows the `courses.component.ts` file containing the following TypeScript code:

```
7    })
8    export class CoursesComponent implements OnInit {
9      titre = 'liste des cours';
10
11     constructor() { }
12     ngOnInit() {}
13
14   CoursesComponent > modifTitle()
```
- Code Editor:** Below it, the `courses.component.html` file contains the following template code:

```
1  <p> Interpolation évaluée à runtime : {{ getTitle() }}</p>
2  <!-- two-way binding : [(ngModel)]
3    permet les interactions TypeScript - Template HTML
4    dans les 2 sens -->
5  <input type="text" [(ngModel)]="titre" />
```
- Browser DevTools:** A bottom panel titled "DevTools - localhost:4200/" shows the browser's developer tools with the "Console" tab selected. It displays the error message:

```
✖ Uncaught Error: Template parse errors:
Can't bind to 'ngModel' since it isn't a known property of 'input'. ("
interactions TypeScript - Template HTML
dans les 2 sens -->
<input type="text" [ERROR ->][(ngModel)]="titre" />
"): ng:///AppModule/CoursesComponent.html@4:19
```

## Two-way binding (2/2)

cours3 [~/enseignement/mobile-19-20/prog/angular/cours3] - .../src/app/app.module.ts - WebStorm

File Edit View Navigate Code Refactor Run Tools VCS Window Help

cours3 > src > app > app.module.ts

Angular CLI Server

1: Project

2: Favorites

3: Structure

4: npm

5: environments

6: Z: Structure

7: Version Control

8: Terminal

9: TypeScript 3.5.3

10: TODO

TSLint: The project code style and editor settings were updated ... (yesterday 08:15) 16:13 LF UTF-8 EditorConfig Git: master

Event Log

Cours 2 : Frontend Angular

20/57

```
4 import { AppRoutingModule } from './app-routing.module';
5 import { AppComponent } from './app.component';
6 import { CoursesComponent } from './courses/courses.component';
7
8 // pour utiliser ngModel, il faut importer FormsModule
9 import { FormsModule } from '@angular/forms';
10
11 @NgModule({
12   declarations: [
13     AppComponent,
14     CoursesComponent
15   ],
16   imports: [
17     BrowserModule,
18     AppRoutingModule,
19     // dans les imports, il faut
20     // aussi rajouter FormsModule
21     FormsModule
22   ],
23   providers: []
24 })
25   AppModule > imports
```

**Ceci est mon composant**

Interpolation évaluée à runtime : liste des

liste des

Interpolation évaluée à runtime : liste des cours

liste des cours

## Uniquement de TypeScript vers le template HTML :

- ▶ Property Binding : [propriété] = "valeur"  
affecte des valeurs aux propriétés d'éléments du DOM
- ▶ Interpolation : {{ champ ou méthode }}
- à utiliser quand on ne peut pas faire de property binding

## Uniquement du template HTML vers le TypeScript :

- ▶ Event binding : (event) = "méthode()"  
Appelle la méthode quand un événement du DOM arrive

## Dans les deux sens :

- ▶ Two-way binding : [(ngModel)] = "valeur"
-  ne pas oublier d'importer FormsModule dans app.module.ts

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

The screenshot shows the WebStorm IDE interface with the following details:

- Project Structure:** The left sidebar shows the project structure with files like course.ts, courses.component.ts, app.module.ts, and app-routing.module.ts.
- Code Editor:** The main editor area displays the code for `courses.component.ts`.

```
1 export interface Course { // interface imposant les informations
2     titre_: string;           // que doit contenir un cours
3     nb_etud_: number;
4 }
5
6 import { Component, OnInit } from '@angular/core';
7 import { Course } from '../course/course'; // importer l'interface
8
9 @Component({
10     selector: 'app-courses', // balise d'insertion
11     templateUrl: './courses.component.html',
12     styleUrls: ['./courses.component.css']
13 })
14 export class CoursesComponent implements OnInit {
15     titre = 'liste des cours';
16     // UE : liste de cours. L'interface impose les champs à remplir
17     UE: Course[] = [ {titre: 'c1', nb_etud: 2},
18                      {titre: 'c2', nb_etud: 5} ];
19     constructor() { }
20     ngOnInit() {}
21     getTitle() { return this.titre; }
22 }
```
- Toolbars and Status Bar:** The top bar shows the file path "cours3 [~/enseignement/mobile-19-20/prog/angular/cours3] - .../src/app/courses/courses.component.ts - WebStorm". The bottom status bar shows "TypeScript 3.5.3", "Terminal", and "Event Log".

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

The screenshot shows the WebStorm IDE interface with the following details:

- Project Structure:** On the left, the project structure is visible, showing the file tree: `cours3` > `src` > `app` > `courses` > `courses.component.html`.
- Code Editor:** The main editor window displays the content of `courses.component.html`. The code includes an interpolation expression `getTitle()` and two nested `<app-course>` tags.
- Terminal:** At the bottom, the terminal shows the command `ng serve` being run, indicating the application is running locally.
- Browser Preview:** A preview window titled "Cours3" shows the rendered output: "Ceci est mon composant App" followed by "Interpolation évaluée à runtime : liste des cours". Below this, there is an input field containing "liste des cours" and two "course works!" messages.
- Bottom Status Bar:** The status bar at the bottom provides information about the file type (TypeScript 3.5.3), terminal status, and event log.

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

The screenshot shows an IDE interface with the following details:

- Project Tree:** Shows the project structure under "angularCours":
  - src
  - node\_modules
  - library
  - src
  - app
    - course
      - course.component.ts
      - course.component.html
      - course.component.css
      - course.component.scss
      - course.ts
    - courses
      - courses.com
      - courses.com
      - courses.com
      - courses.com
      - courses.com
    - app
      - app.component.ts
      - app.component.html
      - app.component.css
      - app.component.scss
      - app.module.ts
      - app-routing.module.ts
    - assets
    - environments
  - Code Editor:** The file `course.component.ts` is open, showing the component definition and its constructor.

```
import {Component, Input, OnInit} from '@angular/core';
import {Course} from './course';

@Component({
  selector: 'app-course',
  templateUrl: './course.component.html',
  styleUrls: ['./course.component.scss']
})
export class CourseComponent implements OnInit {
  // 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 les informations dont
  // elle a besoin pour s'afficher correctement.
  @Input() contenu!: Course; // Ici, il est important de spécifier
                            // le type de contenu. Notez le ! qui permet, en mode strict,
                            // que contenu ne soit pas initialisé lors de sa création.
  constructor() { }
}
```
  - HTML Editor:** The file `course.component.html` is open, showing the template.

```
<p>Cours : {{contenu.titre}} -- {{contenu.nb_etud}} étudiants</p>
```
  - Bottom Status Bar:** Shows the file size (3:1), encoding (LF), line width (2 spaces\*), TypeScript version (4.5.4), branch (master), and event log.

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

The screenshot shows the WebStorm IDE interface with the following details:

- Project Structure:** On the left, the project structure is displayed. It includes the `node_modules`, `src` folder containing `app`, `courses`, and `index.html`, and the `npm` folder.
- Editor:** The main editor window shows the file `courses.component.html`. The code contains runtime interpolation and two `<app-course>` components.
- Terminal:** At the bottom, the terminal shows the command `ng serve` being run.
- Browser Preview:** A preview window shows the application running at `localhost:4200`. The page title is "Cours3". The content displays "Ceci est mon composant App", "Interpolation évaluée à runtime : liste des cours", and two course entries: "Cours : c1 -- 2 étudiants" and "Cours : c2 -- 5 étudiants".

# Interactions enfant → parent : préparation de l'enfant

The screenshot shows the WebStorm IDE interface with the following details:

- Project Bar:** cours3 [~/enseignement/mobile-19-20/prog/angular/cours3] - .../src/app/course/course.component.html - WebStorm
- File Menu:** Fichier Edit View Navigate Code Refactor Run Tools VCS Window Help
- Toolbars:** cours3 src app course course.component.html Angular CLI Server
- Code Editor:** course.component.html (line 1 to 10). The code is as follows:

```
1 <p>Cours : {{ contenu.titre }} </p>
2 <!-- on rajoute un event binding sur l'événement change
3 (en HTML : onchange="...") : chaque fois que l'on modifie
4 l'input, on va ainsi appeler la méthode updateNb de la classe
5 CourseComponent. L'idée est que cette méthode va envoyer des
6 informations sur les changements effectués au parent. -->
7 <input name="{{contenu.titre}}"
8 [(ngModel)]="contenu.nb_etud"
9 (change)="updateNb()"/> étudiants
10
```
- Side Panels:** 1: Project, 2: Structure, 3: npm, 4: Favorites
- Bottom Navigation:** 6: TODO, 9: Version Control, TypeScript 3.5.3, Terminal, Event Log
- Status Bar:** 10:1 LF UTF-8 EditorConfig Git: master

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

The screenshot shows an IDE interface with the following details:

- File Path:** angularCours – course.component.ts
- Toolbar:** File Edit View Navigate Code Refactor Run Tools Git Window Help
- Project:** course.component.ts > CourseComponent > updateNb()
- Angular CLI Server:** Git: ✘ ✅ ✗
- Code Editor:** The code for `course.component.ts` is displayed, implementing the `OnInit` interface and emitting events.
- Side Panels:** Project, Commit, Structure, Bookmarks, npm, and Event Log.
- Bottom Bar:** Git, TODO, Problems, Terminal, and TypeScript 4.5.4 master.

```
import {Component, EventEmitter, Input, OnInit, Output} from '@angular/core';
import {Course} from "./course";

@Component({
  selector: 'app-course',
  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(): void { this.lastNb = this.contenu.nb_etud; }
  updateNb(): void { // fonction appelée quand on change l'input
    const nb = this.contenu.nb_etud - this.lastNb;
    this.lastNb = this.contenu.nb_etud;
    this.newNb.emit(nb);
  }
}
```

# Interactions enfant → parent : la réception du parent

The screenshot shows an IDE interface with the following details:

- File Bar:** File, Edit, View, Navigate, Code, Refactor, Run, Tools, Git, Window, Help.
- Project Bar:** angularCours > src > app > courses > courses.component.html.
- Toolbars:** Angular CLI Server, Git status indicators.
- Side Panels:** Project, Commit, Structure, Bookmarks, Npm.
- Code Editors:**
  - courses.component.html:** Contains interpolation and event binding logic.
  - courses.component.ts:** Contains the CoursesComponent class definition, including properties, methods, and lifecycle hooks.
- Bottom Status Bar:** Git, TODO, Problems, Terminal, Event Log, build status (4:63 LF UTF-8 2 spaces\*, TypeScript 4.5.4), master branch.

```
<p>Interpolation évaluée à runtime : {{getTitle()}} : {{nb_etuds}}</p>
<input type="text" [(ngModel)]="titre" />
<!-- par event binding, on écoute les événements newNb émis par l'enfant.
      le nombre transmis par l'enfant se trouve dans $event -->
<app-course [contenu]="UE[0]" (newNb)="onNewNb($event)"></app-course>
<app-course [contenu]="UE[1]" (newNb)="onNewNb($event)"></app-course>
```

```
export class CoursesComponent implements OnInit {
  titre = 'liste des cours';
  UE : Course[] = [ {titre: 'c1', nb_etud: 2},
                     {titre: 'c2', nb_etud: 5} ];
  nb_etuds! : number;
  constructor() { }
  ngOnInit() : void { this.getNbEtuds(); }
  getNbEtuds() : void {
    this.nb_etuds = 0;
    for (const ue of this.UE) this.nb_etuds += ue.nb_etud;
  }
  onNewNb(delta: number) { this.nb_etuds += delta; } // callback quand newNb arrive
  getTitle() : string { return this.titre; }
```

# Interactions enfant → parent : le résultat

The image displays two side-by-side screenshots of a web browser window titled "Cours3" at the URL "localhost:4200". Both screenshots show the same Angular application interface.

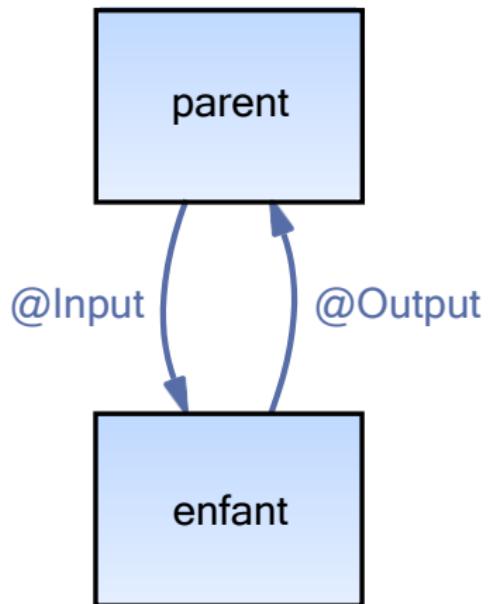
**Screenshot 1 (Left):**

- Text: "Ceci est mon composant App"
- Text: "Interpolation évaluée à runtime : liste des cours : 7" (with the number 7 circled in red)
- Text: "liste des cours" (inside an input field)
- Text: "Cours : c1" (inside an input field)
- Text: "2" (inside an input field) étudiants (with the number 2 circled in red)
- Text: "Cours : c2" (inside an input field)
- Text: "5" (inside an input field) étudiants

**Screenshot 2 (Right):**

- Text: "Ceci est mon composant App"
- Text: "Interpolation évaluée à runtime : liste des cours : 60" (with the number 60 circled in red)
- Text: "liste des cours" (inside an input field)
- Text: "Cours : c1" (inside an input field)
- Text: "55" (inside an input field) étudiants (with the number 55 circled in red)
- Text: "Cours : c2" (inside an input field)
- Text: "5" (inside an input field) étudiants

# Résumé des interactions enfant – parent



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



Insérer une liste de cours dans l'appli :

- 1 Créer un composant `Cours`
- 2 Stocker la liste des cours dans le composant `Courses`
- 3 Dans le template de `Courses`, itérer l'insertion des cours avec la directive `*ngFor`



Toutes les directives (`*ngFor`, `*ngIf`, etc.) débutent par une \*



Peut nécessiter l'inclusion de `CommonModule` dans `app.module.ts`

- `*ngIf` : le composant est utilisé si et seulement si `nglf=true`

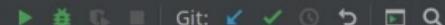
# La directive \*ngFor

cours3 [~/enseignement/mobile-19-20/prog/angular/cours3] - .../src/app/courses/courses.component.html - WebStorm

Fichier Edit View Navigate Code Refactor Run Tools VCS Window Help

cours3 src app courses courses.component.html

Angular CLI Server



Git: ✘ ✓ ⓘ ↻ 🔍

1: Project

courses.component.html

```
1  <p> Interpolation évaluée à runtime : {{ getTitle() }} : {{nb_etuds}}</p>
2  <input type="text" [(ngModel)]="titre" />
3  <!-- directive *ngFor : on parcourt tous les éléments de UE. Chaque
4      élément est stocké dans la variable cours. On peut alors utiliser
5      cette variable, notamment dans les property bindings -->
6  <app-course *ngFor="let cours of UE"
7      [contenu]="cours"
8      (newNb)="onNewNb($event)"></app-course>
9
10 <!-- attention : on ne peut pas appliquer 2 directives (par exemple ngIf et ngFor
11     au même composant. Il faut choisir : soit utiliser ngIf, soit ngFor.
12     Si l'on a besoin des 2, l'astuce est de créer un composant supplémentaire
13     auquel on appliquera, par exemple, le ngFor, et ce composant contiendra le
14     composant qui nous intéresse, auquel on appliquera le ngIf -->
15
```

2: Structure

npm

2: Favorites

6: TODO

9: Version Control

TypeScript 3.5.3

Terminal

Event Log

15:1 LF UTF-8 EditorConfig Git: master

# La directive \*ngIf

cours3 [~/enseignement/mobile-19-20/prog/angular/cours3] - .../src/app/courses/courses.component.html - WebStorm

Fichier Edit View Navigate Code Refactor Run Tools VCS Window Help

courses3 src app courses courses.component.html Angular CLI Server Git: ✘ ✓ ⓘ ⌂ ⌂ ⌂ ⌂

1: Project

1 courses.component.html

1 <p> Interpolation évaluée à runtime : {{ getTitle() }} : {{nb\_etuds}}</p>

2 <div>

3 <!-- Ici, la page web ne contiendra qu'un seul des paragraphes ci-dessous.

4 Si Courses contient moins de 10 étudiants, ce sera le 1er paragraphe,

5 sinon, ce sera le 2ème. Quand je dis que "la page web contiendra...",

6 je parle du DOM : le DOM ne contiendra effectivement qu'un <p>, et

7 non 2 <p> avec l'un des deux hidden. -->

8 <p \*ngIf="nb\_etuds < 10">peu d'étudiants</p>

9 <p \*ngIf="nb\_etuds >= 10">beaucoup d'étudiants</p>

10 </div>

11 <input type="text" [(ngModel)]="titre" />

12 <!-- directive \*ngFor : on parcourt tous les éléments de UE. Chaque

13 élément est stocké dans la variable cours. On peut alors utiliser

14 cette variable, notamment dans les property bindings -->

15 <app-course \*ngFor="let cours of UE"

16 [contenu]="cours"

17 (newNb)="onNewNb(\$event)"></app-course>

18 |

6: TODO 9: Version Control 10 TypeScript 3.5.3 11 Terminal Event Log

18:1 LF UTF-8 EditorConfig Git: master 12 13

Cours 2 : Frontend Angular 33/57

# La directive \*ngIf : le résultat

A Cours3 x +

localhost:4200

## Ceci est mon composant App

Interpolation évaluée à runtime : liste des cours : 12

beaucoup d'étudiants

liste des cours

Cours : c1

2 étudiants

Cours : c2

5 étudiants

Cours : c3

5 étudiants

The screenshot shows two browser windows side-by-side. Both windows have the title 'A Cours3' and the URL 'localhost:4200'. The left window displays three cards for courses 'c1', 'c2', and 'c3'. Course 'c1' has 2 students ('étudiants') and course 'c2' has 5 students ('étudiants'). The card for course 'c3' is partially visible but mostly obscured by a red circle highlighting the number '0' next to it, indicating it is not displayed due to the condition in the \*ngIf directive. The right window shows the same structure but with course 'c3' having 7 students ('étudiants'), so its card is fully visible.

A Cours3 x +

localhost:4200

## Ceci est mon composant App

Interpolation évaluée à runtime : liste des cours : 7

peu d'étudiants

liste des cours

Cours : c1

2 étudiants

Cours : c2

0 étudiants

Cours : c3

5 étudiants

The screenshot shows two browser windows side-by-side. Both windows have the title 'A Cours3' and the URL 'localhost:4200'. The left window displays three cards for courses 'c1', 'c2', and 'c3'. Course 'c1' has 2 students ('étudiants') and course 'c2' has 0 students ('étudiants'). The card for course 'c3' is partially visible but mostly obscured by a red circle highlighting the number '0' next to it, indicating it is not displayed due to the condition in the \*ngIf directive. The right window shows the same structure but with course 'c3' having 5 students ('étudiants'), so its card is fully visible.

- ▶ Actuellement : UE stockées en dur dans composant Courses
- ▶ Vraie application : UE récupérées d'un serveur



: les services récupèrent les données

- ▶ Créer un service : `ng generate service nom_service`
- ▶ Utiliser le service comme n'importe quelle classe

# Anatomie d'un service

The screenshot shows the WebStorm IDE interface with the following details:

- Project Bar:** cours3 [~/enseignement/mobile-19-20/prog/angular/cours3] - .../src/app/services/courses.service.ts - WebStorm
- File Menu:** File Edit View Navigate Code Refactor Run Tools VCS Window Help
- Toolbars:** Angular CLI Server, Git status icons.
- Left Sidebar:** Project (1: Project), npm (2: Favorites).
- Code Editor:** The file courses.service.ts is open. The code defines a CoursesService class with an injectable constructor and a getCourses method returning an array of course objects.

```
1 import { Injectable } from '@angular/core';
2
3 @Injectable({
4   providedIn: 'root'
5 })
6 export class CoursesService {
7   constructor() { }
8
9   // ici, il suffit de créer une méthode qui renvoie les
10  // données dont a besoin le composant Courses. Ce dernier
11  // aura juste à ainsi juste à appeler la méthode pour
12  // obtenir sa liste d'UE
13  getCourses() {
14    return [ {titre: 'c1', nb_etud: 2},
15            {titre: 'c2', nb_etud: 5},
16            {titre: 'c3', nb_etud: 5}];
17  }
18}
19
```

- Bottom Status Bar:** TODO (6), Version Control (9), TypeScript 3.5.3, Terminal, Event Log.
- Bottom Right:** 19:1 LF UTF-8 EditorConfig Git: master

# Exploitation du service dans le composant Courses

The screenshot shows an IDE interface with the following details:

- Title Bar:** cours3 – courses.component.ts
- Menu Bar:** File Edit View Navigate Code Refactor Run Tools Git Window Help
- Toolbar:** cours3 src app courses courses.component.ts CoursesCompon Angular CLI Server Git
- Left Sidebar:** Project, Structure, npm, Bookmarks.
- Code Editor:** The file content is as follows:

```
1 import { Component, OnInit } from '@angular/core';
2 import { Course } from '../course/course';
3 // ici, on importe la classe du service pour pouvoir l'utiliser plus bas
4 import { CoursesService } from '../services/courses.service';
5
6 @Component({
7   selector: 'app-courses', // balise d'insertion
8   templateUrl: './courses.component.html',
9   styleUrls: ['./courses.component.css']
10})
11export class CoursesComponent implements OnInit {
12  titre = 'liste des cours';
13  UE!: Course[]; // ici, on n'initialise plus la liste d'UE
14  nb_etuds!: number;
15  constructor() {
16    // 1ère idée : dans le constructeur, on crée une instance du service et
17    // on appelle sa méthode getCourse pour pouvoir remplir notre tableau UE
18    const service = new CoursesService();
19    this.UE = service.getCourses();
20  }
21  ngOnInit() { this.getNbEtuds(); }
22}
```

Annotations in the code editor include:

- Line 1: A warning icon (red exclamation mark) next to the import statement.
- Line 2: A warning icon next to the import statement.
- Line 3: A warning icon next to the multi-line comment.
- Line 4: A warning icon next to the import statement.
- Line 6: A warning icon next to the @Component annotation.
- Line 7: A warning icon next to the selector assignment.
- Line 8: A warning icon next to the templateUrl assignment.
- Line 9: A warning icon next to the styleUrls assignment.
- Line 11: A warning icon next to the export keyword.
- Line 12: A warning icon next to the assignment to titre.
- Line 13: A warning icon next to the assignment to UE.
- Line 14: A warning icon next to the assignment to nb\_etuds.
- Line 15: A warning icon next to the constructor keyword.
- Line 16: A warning icon next to the multi-line comment.
- Line 17: A warning icon next to the multi-line comment.
- Line 18: A warning icon next to the const keyword.
- Line 19: A warning icon next to the assignment to this.UE.
- Line 21: A warning icon next to the ngOnInit keyword.

**Bottom Status Bar:** Problems, Git, Terminal, TODO, Event Log, 22:1, LF, UTF-8, 2 spaces\*, TypeScript 3.5.3, master, 37/57

## Bonne idée :

- ▶ Faire en sorte que le constructeur connaisse le service

## Mauvaises idées :

- ▶ Demander au constructeur de créer l'instance
  - ▶ Plusieurs composants  $\Rightarrow$  plusieurs instances
  - ▶ Modif du constructeur du service  $\Rightarrow$  modif du composant
- ▶ Utiliser le service dans le constructeur  
 $\Rightarrow$  délais dans les affichages



## Bonne pratique des services :

- ▶ Dependency injection
- ▶ Utiliser le service dans méthode `ngOnInit`

# Dependency injection

cours3 – courses.component.ts

```
File Edit View Navigate Code Refactor Run Tools Git Window Help
courses > courses.component.ts > CoursesComponent > ngOnInit() > Angular CLI Server > Git: ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓
Project: courses.component.ts
6   @Component({
7     selector: 'app-courses', // balise d'insertion
8     templateUrl: './courses.component.html',
9     styleUrls: ['./courses.component.css']
10   })
11   export class CoursesComponent implements OnInit {
12     titre = 'liste des cours';
13     UE!: Course[]; // ici, on n'initialise plus la liste d'UE
14     nb_etuds!: number;
15     // dependency injection : on passe le service en paramètre du constructeur.
16     // Angular ne créera qu'une seule instance de CoursesService pour tout l'application
17     // et passera cette instance au constructeur
18     constructor( private service: CoursesService ) {}
19     ngOnInit() {
20       this.UE = this.service.getCourses(); // c'est ngOnInit qui exploite le service
21       this.getNbEtuds();
22     }
23     getTitle() { return this.titre; }
24     getNbEtuds() {
25       this.nb_etuds = 0;
26       for (const ue of this.UE) this.nb_etuds += ue.nb_etud;
27     }
}
Problems Git Terminal TODO Event Log
20:83 LF UTF-8 2 spaces* TypeScript 3.5.3 master
0 8 ▲ 1 ✘ 3 ▲ ▼
```

- ▶ Service actuel : synchrone  
⇒ `ngOnInit` doit attendre les données
- ▶ Services HTTP : asynchrones  
⇒ évite de bloquer les affichages



## Utilisation de services asynchrones :

- 1 Le service retourne tout de suite un *Observable*
- 2 `ngOnInit` appelle le service et récupère l'observable
- 3 `ngOnInit` souscrit à l'observable en donnant une callback
- 4 `ngOnInit` continue son exécution
- 5 Les données arrivent ⇒ l'observable émet une valeur  
⇒ la callback est appelée

# Mise en place des observables dans le service

The screenshot shows the WebStorm IDE interface with the following details:

- Project:** cours3
- File:** courses.service.ts
- Toolbars:** Angular CLI Server, Git status
- Left Sidebar:** Project, npm, Favorites
- Code Content:**

```
1 import { Injectable } from '@angular/core';
2 // au lieu de renvoyer un tableau de Course, on va renvoyer un Observable sur
3 // ce tableau => il faut importer les déclarations des observables
4 import { Observable, of } from 'rxjs';
5 import { Course } from '../course/course';
6
7 @Injectable({
8   providedIn: 'root'
9 })
10 export class CoursesService {
11   constructor() { }
12
13   // ici, il faut retourner un Observable sur un tableau de Course
14   getCourses_(): Observable<Course[]> {
15     // of retourne un Observable
16     return of([
17       {titre: 'c1', nb_etud: 2},
18       {titre: 'c2', nb_etud: 5},
19       {titre: 'c3', nb_etud: 5}
20     ]);
21 }
```
- Bottom Status Bar:** TODO: 6, Version Control: 9, TypeScript 3.5.3, Terminal, Event Log: 21:1 LF UTF-8 EditorConfig Git: master

# Mise en place des observables dans le composant

The screenshot shows an IDE interface with the following details:

- Title Bar:** cours3 – courses.component.ts
- Menu Bar:** File Edit View Navigate Code Refactor Run Tools Git Window Help
- Toolbar:** courses > courses.component.ts > CoursesComponent > ngOnInit() > Angular CLI Server
- Project Structure:** Shows the file courses.component.ts is selected.
- Code Editor:** The code for the CoursesComponent is displayed:

```
11  export class CoursesComponent implements OnInit {
12    titre = 'liste des cours';
13    UE!: Course[]; // ici, on n'initialise plus la liste d'UE
14    nb_etuds!: number;
15    constructor( private service: CoursesService ) {}
16    ngOnInit() {
17        // on appelle le service, qui nous retourne un observable et on y souscrit.
18        // Le paramètre est une fonction (arrow) qui prend en paramètre la valeur
19        // retournée par l'observable. Cette valeur étant la liste des UE, on la
20        // sauvegarde dans this.UE. Cette opération ne sera réalisée effectivement que
21        // quand le service aura pu récupérer les données demandées.
22        this.service.getCourses().subscribe(
23            next: courses => {
24                this.UE = courses; // on récupère la liste des cours
25                this.getNbEtuds(); // on effectue le reste des opérations dans cette callback
26            }
27        );
28    }
29    getTitle() { return this.titre; }
30    getNbEtuds() {
31        this.nb_etuds = 0;
32        for (const ue of this.UE) this.nb_etuds += ue.nb_etud;
33    }
34}
```
- Bottom Status Bar:** Problems, Git, Terminal, TODO, Event Log, 27:7 LF UTF-8 2 spaces\*, TypeScript 3.5.3, master, 42/57

# Service HTTP : import le HttpClientModule

The screenshot shows the WebStorm IDE interface with the following details:

- Project Structure:** On the left, the project structure is displayed under "1: Project". It includes "node\_modules", "src" (containing "app", "course", "courses", and "services" folders), "assets", and "npm".
- Code Editor:** The main editor window shows the file `app.module.ts`. The code imports several modules and defines an @NgModule block. Specifically, it imports `CoursesComponent`, `FormsModule`, `CourseComponent`, `HttpClientModule`, `BrowserModule`, `AppRoutingModule`, `FormsModule`, and `HttpClientModule` again.
- Toolbars and Status Bar:** The top bar shows the current file as `cours3 [~/enseignement/mobile-19-20/prog/angular/cours3] - .../src/app/app.module.ts - WebStorm`. The bottom status bar indicates the file is saved ("Event Log" icon) and shows the time as 24:58, encoding as LF, and character set as UTF-8.

```
import { CoursesComponent } from './courses/courses.component';
import { FormsModule} from '@angular/forms';
import {CourseComponent} from './course/course.component';

// pour utiliser HttpClient, importer son module
import {HttpClientModule} from '@angular/common/http';

@NgModule({
  declarations: [
    AppComponent,
    CoursesComponent,
    CourseComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    FormsModule,
    HttpClientModule // importer ici le module HttpClient
  ],
  providers: []
})
```

# Service HTTP : notre service utilise HttpClient

The screenshot shows the WebStorm IDE interface with the following details:

- Project:** cours3
- File:** courses.service.ts
- Toolbars:** Angular CLI Server, Git
- Left Sidebar:** Project, npm, Favorites
- Code Content:**

```
1 import { Injectable } from '@angular/core';
2 // on a toujours besoin des Observables, mais on rajoute HttpClient, qui va
3 // s'occuper de converser avec le serveur web
4 import { Observable } from 'rxjs';
5 import { HttpClient } from '@angular/common/http';
6 import { Course } from '../course/course';
7
8 @Injectable({
9   providedIn: 'root'
10 })
11 export class CoursesService {
12   // ici, dependency injection : on va utiliser le HttpClient
13   constructor( private http: HttpClient ) { }
14
15   // ici, il faut retourner un Observable sur un tableau de Course
16   getCourses(): Observable<Course[]> {
17     return this.http.get<Course[]>(` // on converse avec le serveur via une méthode GET
18       url: 'http://127.0.0.1/forum/getCourses.php'
19     );
20   }
21 }
```
- Bottom Status Bar:** TODO, Version Control, TypeScript 3.5.3, Terminal, Event Log
- Bottom Right:** 22:1 LF UTF-8 EditorConfig Git: master

# Service HTTP : getCourses.php

cours3 [~/enseignement/mobile-19-20/prog/angular/cours3] - /home/apache/forum/getCourses.php - WebStorm

File Edit View Navigate Code Refactor Run Tools VCS Window Help

home > apache > forum > getCourses.php

Angular CLI Server



1: Project

2: Structure

npm

2: Favorites

```
1 <?php
2
3 header('Content-type:application/json;charset=utf8');
4 //header("Access-Control-Allow-Origin: *");
5
6 echo json_encode([
7     [ 'titre' => 'c1', 'nb_etud' => 2 ],
8     [ 'titre' => 'c2', 'nb_etud' => 5 ],
9     [ 'titre' => 'c3', 'nb_etud' => 6 ]
10 ]);
11
12 ?>
```

6: TODO

9: Version Control

TypeScript 3.5.3

Terminal

Event Log

12:3 LF UTF-8 4 spaces Git: master

# Service HTTP : Résultat

The screenshot shows a browser window titled "Cours3 - Chromium" with the URL "localhost:4200". The main content area displays the text "Ceci est mon composant App" and "Interpolation évaluée à runtime : liste des cours :". Below this, there is a text input field containing "liste des cours". The browser's developer tools are open, specifically the "Console" tab. The console output shows the following errors:

- 4 messages
- 3 user mess...
- 2 errors
  - Access to XMLHttpRequest at 'http://127.0.0.1/forum/getCourses.php' from origin 'http://localhost:4200' has been blocked by CORS policy: No 'Access-Control-Allow-Origin' header is present on the requested resource.
  - Error
- No warnings

- ▶ **Cross-Origin Resource Sharing (CORS) :**
- ▶ Requête cross-origine : provient de localhost :4200 accède à localhost :80  
⇒ CORS refuse la requête
- ▶ **Contre-mesure :** dans `getCourses.php`, rajouter :  
`header ("Access-Control-Allow-Origin: *");`

# Service HTTP : le bon getCourses.php

The screenshot shows the WebStorm IDE interface with the following details:

- Project:** cours3 [~/enseignement/mobile-19-20/prog/angular/cours3]
- File:** getCourse.php
- Toolbars:** Angular CLI Server, Git status indicators.
- Code Content:**

```
<?php  
// le script doit renvoyer les données au format JSON  
header('Content-type:application/json;charset=utf8');  
  
// ici, on rajoute un header pour contourner le problème CORS  
// A noter que ce n'est à faire ici que parce qu'on utilise Angular en front-end  
// sur le port 4200 et PHP en back-end sur le port 80.  
header("Access-Control-Allow-Origin: *");  
  
echo json_encode ([  
    [ 'titre' => 'c1', 'nb_etud' => 2 ],  
    [ 'titre' => 'c2', 'nb_etud' => 5 ],  
    [ 'titre' => 'c3', 'nb_etud' => 6 ]  
]);  
?>
```
- Side Panels:** Project, npm, Favorites.
- Bottom Status Bar:** TODO (6), Version Control (9), TypeScript 3.5.3, Terminal, Event Log.
- Bottom Right:** File Encoding (UTF-8), Line Endings (LF), Git status (master).

# Service HTTP : le bon résultat

Ceci est mon composant App

Interpolation évaluée à runtime : liste des cours : 13

beaucoup d'étudiants

liste des cours

Cours : c1

2 étudiants

Cours : c2

5 étudiants

Cours : c3

6 étudiants

Cours3 - Chromium

localhost:4200

Elements Console Sources Network Performance Memory Application Security Audits

Default levels

top

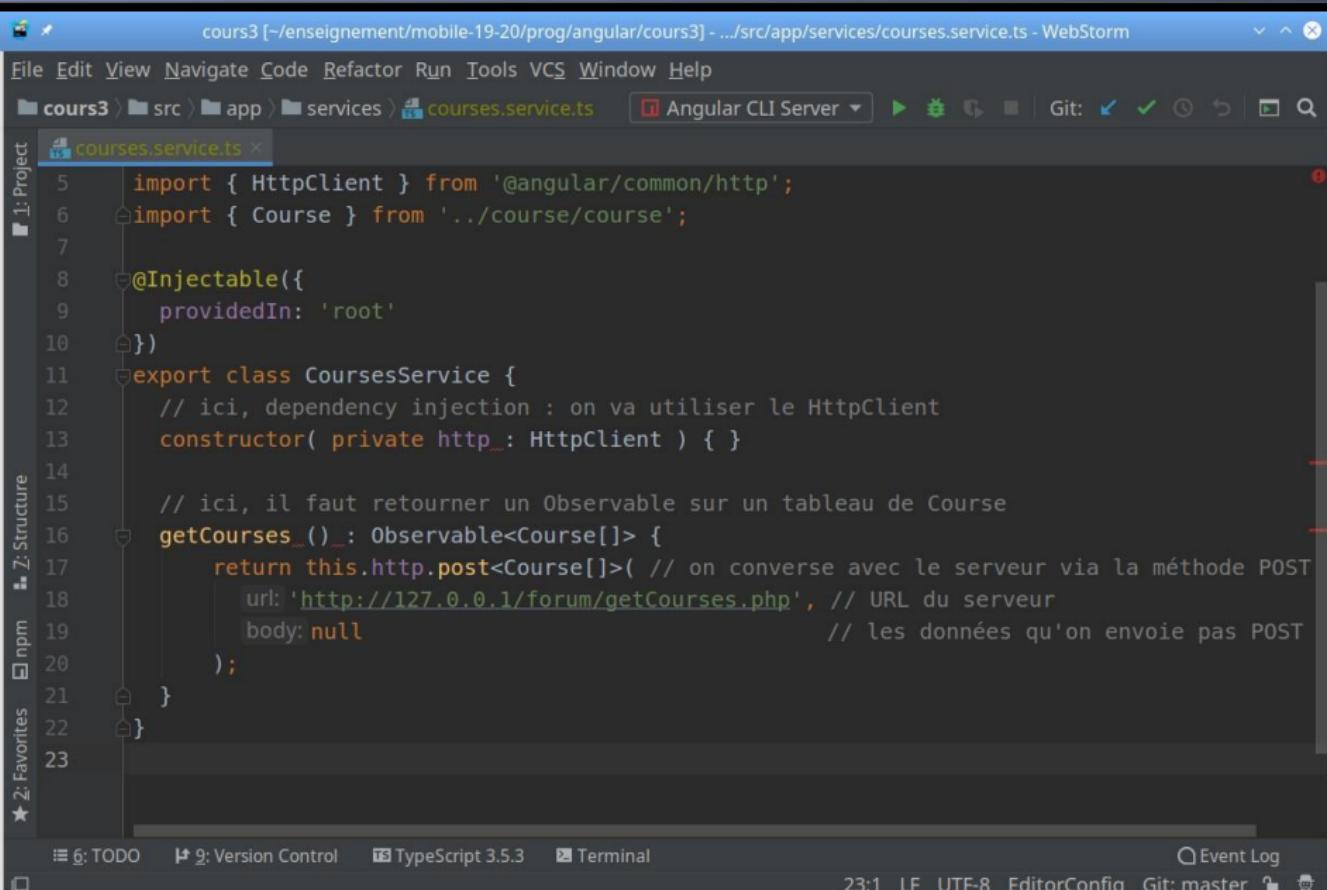
Filter

▶ : 2 messages >

▶ ⚠ 2 user mess...

✖ No errors

# Service HTTP : HttpClient avec méthode POST



The screenshot shows the WebStorm IDE interface with the following details:

- Project:** cours3
- File:** courses.service.ts
- Tool:** Angular CLI Server
- Code (courses.service.ts):**

```
5 import { HttpClient } from '@angular/common/http';
6 import { Course } from '../course/course';
7
8 @Injectable({
9   providedIn: 'root'
10 })
11 export class CoursesService {
12   // ici, dependency injection : on va utiliser le HttpClient
13   constructor( private http: HttpClient ) { }
14
15   // ici, il faut retourner un Observable sur un tableau de Course
16   getCourses(): Observable<Course[]> {
17     return this.http.post<Course[]>(
18       url: 'http://127.0.0.1/forum/getCourses.php', // URL du serveur
19       body: null // les données qu'on envoie pas POST
20     );
21   }
22 }
```

- Bottom Bar:** TODO (6), Version Control (9), TypeScript 3.5.3, Terminal, Event Log.
- Status Bar:** 23:1 LF UTF-8 EditorConfig Git: master

# Service HTTP : méthode POST et session

The screenshot shows a code editor interface with the following details:

- Title Bar:** cours3 – courses.service.ts
- Menu Bar:** File Edit View Navigate Code Refactor Run Tools Git Window Help
- Toolbar:** Angular CLI Server, Git status indicators.
- Project Explorer:** Shows files like app-routing.module.ts, app.module.ts, and courses.service.ts.
- Code Editor:** The file courses.service.ts contains the following code:

```
import { HttpClient } from '@angular/common/http';
import { Course } from '../course/course';

@Injectable({
  providedIn: 'root'
})
export class CoursesService {
  // ici, dependency injection : on va utiliser le HttpClient
  constructor( private http: HttpClient ) {}

  // ici, il faut retourner un Observable sur un tableau de Course
  getCourses(): Observable<Course[]> {
    return this.http.post<Course[]>(`http://127.0.0.1/forum/getCourses.php`, // URL du serveur
      null, // les données qu'on envoie pas POST
      {withCredentials: true} // pour capturer les cookies de session
  }
}
```

The code uses the HttpClient service to make a POST request to a PHP endpoint. It includes options for credentials and capturing session cookies.

**Bottom Navigation:** Problems, Git, Terminal, TODO, Event Log.



- ▶ **Contre-mesures : dans getCourse.php :**

```
header("Access-Control-Allow-Origin: " .  
       $_SERVER['HTTP_ORIGIN']);  
  
header("Access-Control-Allow-Credentials: true");
```

# Service HTTP en résumé

- 1 Dans app.module.ts : importer le HttpClientModule
- 2 Faire ng generate service mon service
- 3 mon-service importe la classe HttpClient
- 4 Constructeur de mon-service : dependency injection de HttpClient
- 5 Méthodes qui appellent get ou post de HttpClient  
=⇒ renvoient des observables
- 6 Composant client importe le service
- 7 Dans son `ngOnInit`, on récupère les observables et on y souscrit  
Le code dépendant des données est dans la callback en paramètre de subscribe

# Navigation : les routes dans app-routing.module.ts

The screenshot shows a code editor interface with the following details:

- Title Bar:** cours3 – app-routing.module.ts
- Menu Bar:** File Edit View Navigate Code Refactor Run Tools Git Window Help
- Toolbar:** Angular CLI Server, Git status indicators.
- Sidebar:** Project, Structure, Favorites, npm.
- Code Area:** The file content is displayed, showing the configuration of routes for an Angular application.

```
2 import { Routes, RouterModule } from '@angular/router';
3 import { CoursesComponent } from './courses/courses.component';
4 import { TopicsComponent } from './topics/topics.component';
5
6 // ici, on indique les routes de l'application. path = l'URL pour accéder au composant
7 const routes: Routes = [
8   { path: 'cours', component: CoursesComponent },
9   // si, dans le path, il y a des ":" , cela indique que ce sont des paramètres.
10  // Ainsi, on va pouvoir accéder à sujets/33, où id vaudra 33
11   { path: 'sujets/:id', component: TopicsComponent }
12 ];
13
14 @NgModule({
15   imports: [RouterModule.forRoot(routes)], // on indique quels routes l'appli utilise
16   exports: [RouterModule]
17 })
18 export class AppRoutingModule { }
```

- Bottom Bar:** Problems, Git, Terminal, TODO, Event Log.
- Status Bar:** 13:1 LF UTF-8 2 spaces\* TypeScript 3.5.3 master

# Navigation : AppRoutingModule dans app.module.ts

The screenshot shows the VS Code interface with the following details:

- Title Bar:** cours3 - app.module.ts
- File Menu:** File Edit View Navigate Code Refactor Run Tools Git Window Help
- Toolbar:** Angular CLI Server, Git status indicators.
- Project Explorer:** Shows 'app-routing.module.ts' and 'app.module.ts' in the current workspace.
- Code Editor:** Displays the content of 'app.module.ts'. The code imports 'AppRoutingModule' and defines an @NgModule block with declarations, imports, providers, and bootstrap components.
- Sidemenu:** Project, Structure, Favorites, npm.
- Bottom Bar:** Problems, Git, Terminal, TODO, Event Log.
- Status Bar:** 15:1 LF UTF-8 2 spaces\* TypeScript 3.5.3 master

```
import { AppRouterModule } from './app-routing.module';
@NgModule({
  declarations: [
    AppComponent,
    CoursesComponent,
    CourseComponent,
    TopicsComponent
  ],
  imports: [
    BrowserModule,
    AppRouterModule, // l'appli doit avoir la capacité de faire du routing
    FormsModule,
    HttpClientModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

# RouterLink : le lien de navigation

The screenshot shows the WebStorm IDE interface with the following details:

- Title Bar:** cours3 [~/enseignement/mobile-19-20/prog/angular/cours3] - .../src/app/courses/courses.component.html - WebStorm
- Menu Bar:** File Edit View Navigate Code Refactor Run Tools VCS Window Help
- Toolbar:** Angular CLI Server, Git status icons.
- Project Tree:** 1: Project, shows the file structure: src/app/courses/courses.component.html.
- Code Editor:** The content of courses.component.html is displayed:

```
1  <!-- on rajoute ici une barre de navigation. Surtout, ne pas
2    utiliser href pour passer d'une page à l'autre : cela
3    rechargerait toute l'application. Ici, il faut utiliser
4    la propriété RouterLink qui permet à Angular de modifier
5    la vue sans rechargeement. -->
6  <nav>
7    <a routerLink="/sujets">Mes sujets</a>
8  </nav>
9
10 <p> Interpolation évaluée à runtime : {{ getTitle() }} : {{nb_etuds}}</p>
11 <div>
12   <p *ngIf="nb_etuds < 10">peu d'étudiants</p>
13   <p *ngIf="nb_etuds >= 10">beaucoup d'étudiants</p>
14 </div>
15   <input type="text" [(ngModel)]="titre" />
16 <app-course *ngFor="let cours of UE"
17   [contenu]="cours"
18   (newNb)="onNewNb($event)"></app-course>
19
```
- Side Panels:** npm, Favorites, Z: Structure.
- Bottom Navigation:** TODO, Version Control, TypeScript 3.5.3, Terminal, Event Log.
- Status Bar:** 5:21 LF UTF-8 EditorConfig Git: master

# Les liens paramétrés (1/2)

cours3 [~/enseignement/mobile-19-20/prog/angular/cours3] - .../src/app/topics/topics.component.html - WebStorm

Fichier Edit View Navigate Code Refactor Run Tools VCS Window Help

cours3 > src > app > topics > topics.component.html

Angular CLI Server

Git: ✘ ✘ ✘ ✘ ✘ ✘

1: Project

app.module.ts

```
16
17     // ici, on indique les routes de l'application. path = l'URL pour accéder au composant
18     const myRoutes : Routes = [
19       { path: 'cours', component: CoursesComponent },
20       // si, dans le path, il y a des ":" , cela indique que ce sont des paramètres.
21       // Ainsi, on va pouvoir accéder à sujets/33, où id vaudra 33
22       { path: 'sujets/:id', component: TopicsComponent}
23     ];
24
```

topics.component.html

```
1     <!-- ici, par interpolation, on va afficher l'id qui a été passé en
2         paramètre de l'URL. Pour cela, la classe du composant va récupérer
3         cette information et la transmettre au template HTML par interpolation. -->
4     <p>topics {{my_id}}</p>
```

2: Structure

npm

2: Favorites

6: TODO

9: Version Control

TypeScript 3.5.3

Terminal

Event Log

3:81 LF UTF-8 EditorConfig Git: master

# Les liens paramétrés (2/2)

cours3 – topics.component.ts

File Edit View Navigate Code Refactor Run Tools Git Window Help

cours3 src app topics topics.component.ts

Angular CLI Server

Git: ✓ ✓ ✓ ✓ ✓

topics.component.ts

```
9  export class TopicsComponent implements OnInit {  
10     my_id!: number;  
11  
12     // le constructeur récupère par dependency injection la  
13     // route qui a mené à lui => on va pouvoir récupérer les  
14     // paramètres passés dans l'URL  
15     constructor(private route: ActivatedRoute) { }  
16  
17     ngOnInit() {  
18         // ici, on récupère le paramètre de la route  
19         this.my_id = this.route.snapshot.params['id'];  
20     }  
21 }  
22  
23  
24  
25  
26  
27  
28  
29
```

A screenshot of an IDE (Angular CLI Server) showing a browser preview and the code editor for topics.component.ts. The browser preview shows the URL 'localhost:4200/sujets/25' and the page content 'Ceci est mon composant App' with 'topics 25' highlighted. A red circle highlights the URL in the browser address bar. Another red circle highlights the 'topics 25' text on the page.

npm

Bookmarks

Problems Git Terminal TODO

Event Log

26:1 LF UTF-8 2 spaces\* TypeScript 3.5.3 master