

Cours 3 : TypeScript et Angular



Applications web et mobile

Christophe Gonzales

TypeScript

TypeScript ⊂ JavaScript

⇒ code JavaScript valide = code TypeScript valide

► *Principe d'utilisation :*

- ① Écrire du code en TypeScript (`mon_code.ts`)
- ② Le compiler (transpiler) en JavaScript
`tsc mon_code.ts` produit `mon_code.js`
- ③ Exécuter le JavaScript produit (`mon_code.js`)

► *Ajouts principaux par rapport à Javascript :*

- ▶ Typage fort ⇒ facilite les déboggages
- ▶ Notions orientées objet : interfaces, etc.
- ▶ Erreurs à la compilation

Les types en TypeScript

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

- Title Bar:** "typescript [~/enseignement/mobile-19-20/prog/typescript] - .../types.ts - WebStorm"
- Menu Bar:** File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, Help
- Toolbar:** Add Configuration..., Git: (with icons for commit, pull, push, etc.)
- Project View (1: Project):** Shows the file "types.ts" under a "typescript" project.
- Structure View (2: Structure):** Shows the code structure with foldable sections.
- Favorites View (2: Favorites):** Shows a star icon.
- Code Editor:** Displays the following TypeScript code:

```
1 // TypeScript a ses propres types, compatibles avec ceux de JavaScript
2 // Après compilation, en JavaScript, ces variables pourront éventuellement
3 // avoir un autre type (cf. les tableaux ci-dessous)
4
5 // TypeScript peut inférer des types à partir des valeurs, comme JavaScript :
6 let x = 4;           // type number
7 let y = 'toto';      // type string
8 let z = false;        // type boolean
9
10 // les tableaux :
11 let tab1 = [1, 2, 3];    // type number[] : tableau contenant uniquement des entiers
12 let tab2 = ['toto', 'titi']; // type string[] : tableau contenant uniquement des strings
13 let tab3 = [3, 'toto'];    // type (number|string)[] : tableau ne pouvant contenir que
14 // des entiers et des strings
15
```
- Terminal:** Local, +, TSC output:

```
[shalmaneser:~/enseignement/mobile-19-20/prog/typescript]<1> tsc types.ts
[shalmaneser:~/enseignement/mobile-19-20/prog/typescript]<2>
```
- Bottom Navigation:** TODO, Version Control, TypeScript 3.7.5, Terminal, Event Log (1)
- Status Bar:** WebStorm 2019.3.2 available: // Update... (today 2:43 PM), 15:1 LF UTF-8 4 spaces, Git: master

Typeage fort : une variable ne change pas de type

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

- Title Bar:** typescript [~/enseignement/mobile-19-20/prog/typescript] - .../types2.ts - WebStorm
- Menu Bar:** File Edit View Navigate Code Refactor Run Tools VCS Window Help
- Project Tree:** 1: Project (types2.ts)
- Code Editor:** The file contains the following TypeScript code:

```
// En TypeScript, quand on crée une variable, on lui affecte
// un type et cette variable conservera toujours ce type
// => on ne peut changer le type d'une variable en cours de programme
// ni lui affecter une valeur d'un type incompatible :
// cela provoque une erreur de compilation

let x = 4; // type number

x = 'toto';
```

A tooltip from the IDE indicates: "TS2322: Type "toto"" is not assignable to type 'number'." Below the tooltip are options: "Change 'x' type to 'string'" (Alt+Shift+Enter) and "More actions..." (Alt+Enter).
- Terminal:** Local (2) +
[shalmeneser:~/enseignement/mobile-19-20/prog/typescript]<11> tsc types2.ts
types2.ts:8:1 - error TS2322: Type '"toto"' is not assignable to type 'number'.
- Bottom Bar:** TODO, Version Control, TypeScript 3.7.5, Terminal, Event Log (1), Git: master, WebStorm 2019.3.2 available: // Update... (today 2:43 PM), 10:1 LF UTF-8 4 spaces, Git: master

Le type any

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

- Title Bar:** typescript [~/enseignement/mobile-19-20/prog/typescript] - .../types3.ts - WebStorm
- Menu Bar:** File Edit View Navigate Code Refactor Run Tools VCS Window Help
- Toolbar:** Add Configuration..., Project Structure, Git status icons.
- Project Tree:** 1: Project, types3.ts
- Code Editor:** The file types3.ts contains the following TypeScript code:

```
// si l'on déclare une variable sans lui affecter une valeur
// son type = any. Dans ce cas, on peut lui affecter n'importe
// quelle valeur, son type sera toujours any. Ceci implique
// que l'on peut lui réaffecter une valeur d'un autre type.
let x; // type = any

x = 4;      // x est encore de type any mais sa valeur est 4
           // les types de x dans TypeScript et JavaScript sont donc différents
x = 'toto'; // instruction valide : x est de type any et vaut 'toto'

// On peut imposer le type d'une variable lors de sa déclaration :
let y : number; // y de type number mais sa valeur est undefined
y = 4;          // ok 4 est bien un nombre
y = 'titi';
           // erreur à la compilation : 'toto' n'est pas un nombre

y
```
- Terminal:** Local (2) (active), Local (1), +
- Status Bar:** [shalmaneser:~/enseignement/mobile-19-20/prog/typescript]<13> tsc types3.ts
types3.ts:14:1 - error TS2322: Type '"titi"' is not assignable to type 'number'.
- Bottom Navigation:** TODO, Version Control, TypeScript 3.7.5, Terminal, Event Log, WebStorm 2019.3.2 available: // Update... (today 2:43 PM), 14:10 LF UTF-8 4 spaces Git: master

Imposer le type d'une variable

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

- Title Bar:** typescript [~/enseignement/mobile-19-20/prog/typescript] - .../types4.ts - WebStorm
- Menu Bar:** File Edit View Navigate Code Refactor Run Tools VCS Window Help
- Toolbar:** Add Configuration..., Git: (with icons for commit, push, pull, etc.)
- Project Tree:** 1: Project (expanded), types4.ts
- Code Editor:** Contains the following TypeScript code:

```
// on peut imposer le type d'une variable avec l'opérateur ': type'  
let x : number = 4; // superflu, 4 impliquait déjà que x était un number  
x = false; // ERREUR de compilation : false n'est pas un nombre  
  
let y : any = 4; // impose que y soit any, même s'il vaut le nombre 4  
y = 'toto'; // valide puisque y est de type any  
  
let z = undefined; // z de type any car on sait qu'on initialise ce genre  
// de variable pour lui affecter une autre valeur par la suite  
let u : undefined = undefined; // u de type undefined et restera toujours de ce type  
u = 3; // ERREUR : 3 n'est pas de type undefined  
  
let v : null = null; // v de type null  
v = 2; // ERREUR : 2 n'est pas de type null  
  
let w1 : number = undefined; // OK : variable non initialisée  
let w2 : number = null; // OK : variable non initialisée  
w1 = 10;  
w2 = 20;
```
- Side Panels:** Structure, Favorites
- Bottom Navigation:** TODO (6), Version Control (9), TypeScript 3.7.5, Terminal, Event Log (1)
- Status Bar:** WebStorm 2019.3.2 available: // Update... (today 2:43 PM), 20:1 LF UTF-8 4 spaces, Git: master

Les tableaux et les tuples

typescript [~/enseignement/mobile-19-20/prog/typescript] - .../types5.ts - WebStorm

File Edit View Navigate Code Refactor Run Tools VCS Window Help

typescript > types5.ts

Add Configuration...



```
// dans les tableaux, on précise le type d'éléments que ceux-ci contiennent
let x = [1,2]; // x de type number[] : ne peut contenir que des entiers
x[2] = 3;      // on peut rajouter des éléments entiers
x[3] = 'toto'; // ERREUR : 'toto' n'est pas un entier

let y : number[] = []; // tableau vide d'entiers : on ne pourra rajouter que des entiers
y[0] = 3;           // OK : 3 est un entier

let z : (number|string)[] = []; // tableau contenant des entiers ou des strings
z[0] = 'toto';          // OK : 'toto' est une string
z[1] = 3;              // OK : 3 est un nombre
z[2] = false;          // ERREUR : false n'est ni un entier ni une string

// les tuples : on indique précisément le type de chaque élément du tuple
let t : [number,string] = null; // tuple dont le 1er élément est un nombre et le 2ème une string
t = [3,'toto'];           // OK : 1er élément = number, 2ème élément = string
t = ['toto',3];           // ERREUR : 'toto' != number et 3 != string
t = [3,'toto','titi'];    // t = couple, pas triplet
```

1: Project

2: Structure

3: Favorites



6: TODO 9: Version Control TypeScript 3.7.5 Terminal

1 Event Log

WebStorm 2019.3.2 available: // Update... (today 2:43 PM)

19:1 LF UTF-8 4 spaces Git: master

Les énumérations

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

- File Path:** typescript [~/enseignement/mobile-19-20/prog/typescript] - .../types6.ts - WebStorm
- Menu Bar:** File Edit View Navigate Code Refactor Run Tools VCS Window Help
- Toolbar:** Add Configuration... (with icons for Run, Stop, Refresh, Save, Undo, Redo, Find, Replace, and Git status)
- Project View:** Shows a single file "types6.ts" selected.
- Code Editor:** Displays the following TypeScript code:

```
1 // Contrairement à Javascript, TypeScript possède la notion d'énumération :
2 // Elle est similaire au langage C
3
4 // définition d'une énumération : on indique les noms des valeurs (ici,
5 // webstorm rajoute la correspondance en termes de valeurs (= 0, = 1, = 2))
6 enum Couleur { Rouge = 0 , Vert = 1 , Bleu = 2 };
7
8 // utilisation
9 const ma_couleur = Couleur.Rouge;
10
11 // définition d'une énumération en indiquant précisément les valeurs
12 enum Taille { Petit = 3, Moyen = 10, Grand = 11 };
13 const ma_taille : Taille = Taille.Petit;
14 console.log(ma_taille); // affichera 3
```
- Terminal:** Shows the command-line output:

```
[shalmaneser:~/enseignement/mobile-19-20/prog/typescript]<1> tsc types6.ts
[shalmaneser:~/enseignement/mobile-19-20/prog/typescript]<2> node types6.js
3
```
- Bottom Navigation:** TODO, Version Control, TypeScript 3.7.5, Terminal, Event Log (with a count of 1).
- Status Bar:** WebStorm 2019.3.2 available: // Update... (today 2:43 PM), 15:1 LF UTF-8 4 spaces, Git: master.

Les fonctions

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

- Project Bar:** Shows "typescript" as the current project.
- File Menu:** File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, Help.
- Add Configuration...:** A button to add a new configuration.
- Toolbars:** Includes icons for back, forward, search, and other navigation.
- Git Status:** Shows status icons for master branch.
- Code Editor:** Displays the file "types7.ts" with the following content:

```
// déclarations de fonctions avec et sans typage
function f(x, y) { // fonctionne avec des number et des string
    return x + y;
}
console.log(f(x: 3, y: 4), ' ', f(x: 'toto', y: 'titi')); // OK

function g(x : number, y : number) { // ne fonctionne qu'avec des entiers
    return x + y;
}
console.log(g(x: 3, y: 4));
console.log(g(x: 'aaa', y: 'bbb'));
```
- Terminal:** Local terminal showing the command `tsc types7.ts` and an error message: `types7.ts:11:16 - error TS2345: Argument of type '"aaa"' is not assignable to parameter of type 'number'.`
- Bottom Navigation:** TODO, Version Control, TypeScript 3.7.5, Terminal, Event Log.
- Bottom Status:** WebStorm 2019.3.2 available: // Update... (today 2:43 PM), 11:27, LF, UTF-8, 4 spaces, Git: master.

Les arrow fonctions

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

- Title Bar:** typescript [~/enseignement/mobile-19-20/prog/typescript] - .../typescript8.ts - WebStorm
- Menu Bar:** File Edit View Navigate Code Refactor Run Tools VCS Window Help
- Toolbar:** Add Configuration..., Run, Stop, Refresh, Git status icons.
- Project Tree:** 1: Project, types8.ts
- Code Editor:** Contains the following TypeScript code:

```
1 // déclaration de fonction à la Javascript
2 let f = function (x, y) {
3     return x + y;
4 }
5
6 // syntaxe des arrow functions : (params) => { code }
7 // définition identique à f, mais plus simple (très utile en Angular)
8 let g = (x, y) => {
9     return x + y;
10}
11
12 // syntaxe avec typage
13 let h = (x: number, y : number ) => {
14     return x + y;
15 }
16
17 console.log (f( x: 3, y: 4)); // OK
18 console.log (g( x: 3, y: 4)); // OK
19 console.log (h( x: 3, y: 4)); // OK
20
```
- Side Panels:** Structure, Favorites
- Bottom Status Bar:** TODO 6, Version Control 9, TypeScript 3.7.5, Terminal, Event Log 1, WebStorm 2019.3.2 available: // Update... (today 2:43 PM), 20:1 LF UTF-8 4 spaces, Git: master.

Les interfaces

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

- File Menu:** File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, Help.
- Toolbar:** Add Configuration..., Run, Stop, Go, Git, Undo, Redo, Find, Replace.
- Project Tree:** Shows a project named "typescript" containing "interface.ts".
- Code Editor:** Displays the content of "interface.ts". The code defines an interface "Point" with properties x and y, a function "display" that logs the coordinates, and an extended interface "XPoint" with a "translate" method. A class "MyPoint" implements "XPoint" and has its own constructor and "translate" method. Finally, it creates and uses an instance of "MyPoint".

```
typescript [~/enseignement/mobile-19-20/prog/typescript] - .../interface.ts - WebStorm
File Edit View Navigate Code Refactor Run Tools VCS Window Help
typescript > interface.ts
Add Configuration...
1: Project
1: Structure
2: Favorites
interface.ts
1 interface Point { // TypeScript introduit la notion d'interface
2   x: number, // ici, on spécifie les types des champs
3   y: number
4 }
5   let display = (p : Point) => { console.log(p.x, p.y); } // fonction qui prend un Point en argument
6   display ({x : 3, y : 5}); // on applique la fonction avec un objet qui implémente l'interface
7
8 interface XPoint extends Point { // extension de Point, comme en Java
9   translate (delta_x : number, delta_y : number) : void // signature de fonction
10 }
11 class MyPoint implements XPoint {
12   x : number;
13   y : number;
14   constructor (x : number, y : number) {this.x = x; this.y = y; }
15   translate (delta_x: number, delta_y: number): void { this.x += delta_x; this.y += delta_y; }
16 }
17 const pp = new MyPoint ( x: 1, y: 2);
18 pp.translate ( delta_x: 3, delta_y: 0); // => pp.x = 4 et pp.y = 2
19 display(pp); // affiche 4 2
20
```

☰ 6: TODO ⚡ 9: Version Control TS TypeScript 3.7.5 ☰ Terminal

1 Event Log

20:1 LF UTF-8 4 spaces Git: master

Les constructeurs

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

- Title Bar:** typescript [~/enseignement/mobile-19-20/prog/typescript] - .../constructor.ts - WebStorm
- Menu Bar:** File Edit View Navigate Code Refactor Run Tools VCS Window Help
- Toolbar:** Add Configuration..., Git: (with icons for commit, push, pull, etc.)
- Project Structure:** On the left, shows a tree view with '1: Project' expanded, containing 'typescript' and 'constructor.ts'.
- Code Editor:** The main area contains the following TypeScript code:

```
1 // TypeScript permet plus simplement que Javascript d'avoir des champs et méthodes privées
2 class MyPoint {
3     x: number;           // par défaut : x est public
4     private y: number;   // champ privé
5     public z: number;    // champ public
6
7     // constructeur avec signature
8     // ici, le ? signifie que y est optionnel
9     constructor(x: number, y?: number) { this.x = x; this.y = y; }
10
11
12 class MyPointBis {
13     // si, dans la signature du constructeur, on précise si les paramètres sont public ou private,
14     // inutile de créer et d'initialiser les champs correspondants. Ils le seront automatiquement
15     constructor (private x : number, public y : number ) {}
16
17
18 let x = new MyPointBis ( x: 3, y: 4);
19 console.log(x.y); // affiche 4
20
```
- Bottom Navigation:** TODO (6), Version Control (9), TypeScript 3.7.5, Terminal, Event Log (1).
- Bottom Status:** WebStorm 2019.3.2 available: // Update... (today 2:43 PM), 20:1 LF UTF-8 4 spaces, Git: master.

- ▶ Classe Point définie dans le fichier point.ts
⇒ accessible uniquement dans point.ts

Modules : étendent les accès à d'autres fichiers

- ▶ 2 étapes :

- ▶ **Exportations :**

```
export class Point { ... }
```

- ▶ **Importations :**

```
import { Point } from './point';
```

Après le from, nom du fichier sans l'extension .ts



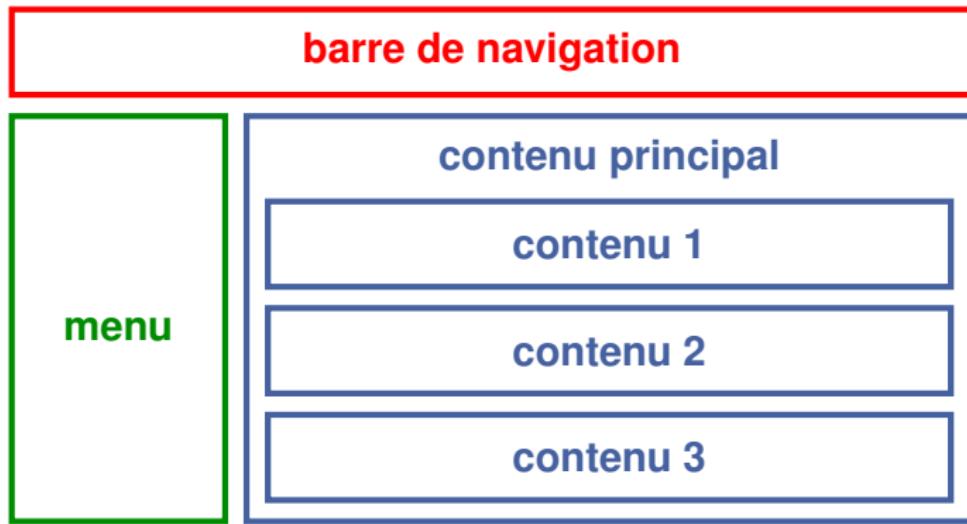
Module TypeScript/JavaScript ≠ module Angular



- ▶ 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

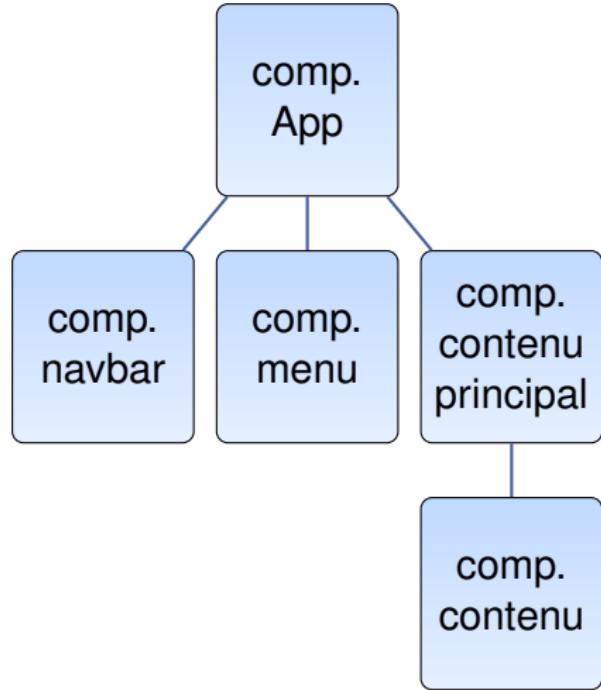
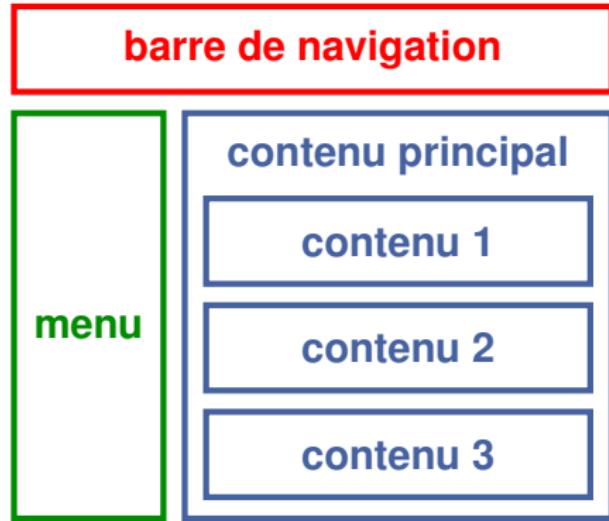
Structure d'une application Angular

Affichage de la page web :



- ▶ Affichage ⇒ 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



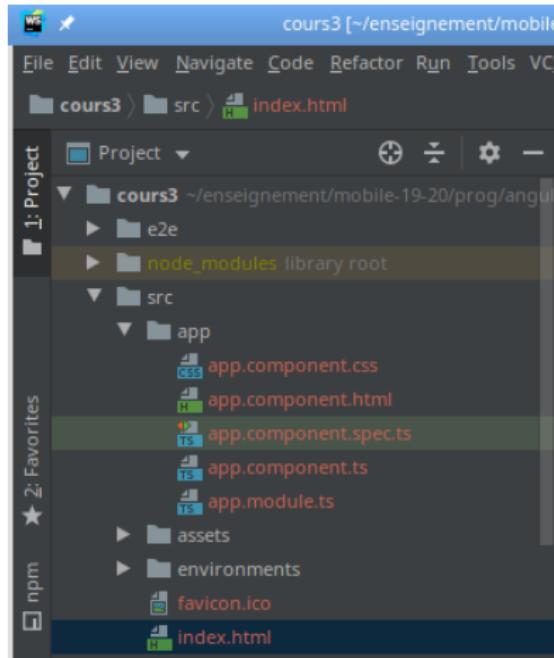
→ permet de structurer facilement le code !

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`

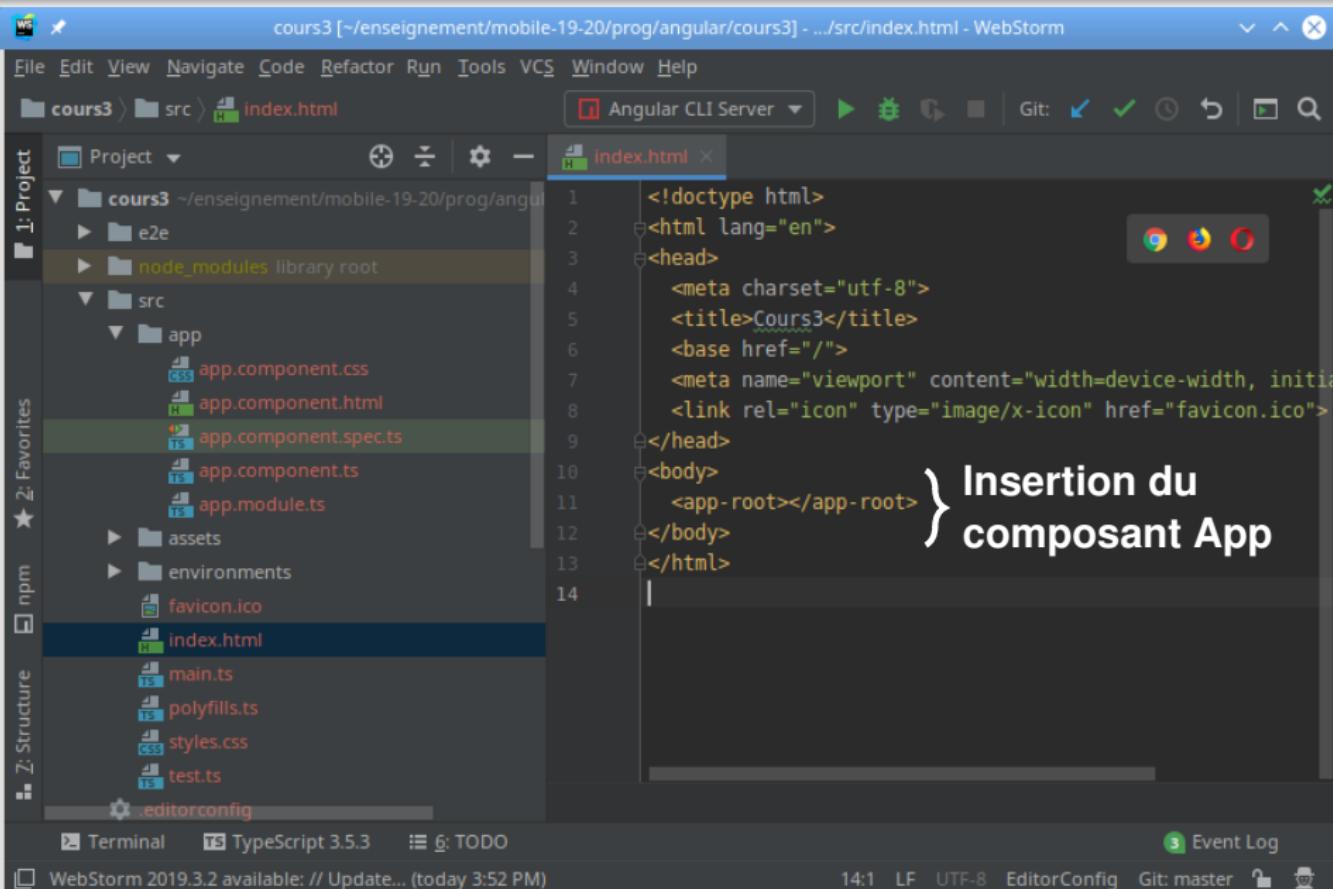
Génération d'un projet Angular

- ▶ `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



- ➊ Créer les composants (et les modules)
- ➋ Les compiler
- ➌ 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 the following details:

- Project Structure:** The left sidebar shows the project structure under "cours3". The "src" folder contains "app" which includes "app.component.css", "app.component.html", "app.component.spec.ts", "app.component.ts", and "app.module.ts".
- Editor:** The main editor window displays the content of "app.component.html": <h1>Ceci est mon composant App</h1>.
- Terminal:** The bottom-left terminal shows the command: [shalmaneser:~/enseignement/mobile-19-20/prog/angular/cours3]<1> ng serve. The output indicates "10% building 3/3 modules 0 active", "wds: Project is running at <http://localhost:4200/webpack-dev-server/>", "wds: webpack output is served from /", and "wds: 404s will fallback to //index.html".
- Browser Preview:** A separate window titled "Cours3 - Chromium" shows the rendered HTML content: "Ceci est mon composant App".
- Bottom Status Bar:** Shows "WebStorm 2019.3.2 available: // Update... (today 3:52 PM)" and file status indicators (2:1, LF, UTF-8, EditorConfig, Git: master).

Le TypeScript du composant app

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

File Edit View Navigate Code Refactor Run Tools VCS Window Help

cours3 > src > app > app.component.ts

Angular CLI Server



Git:



1: Project

2: Favorites

3: npm

4: Structure

5:

app.component.html x app.component.ts x

```
1 import { Component } from '@angular/core';
2
3 // décorateur qui indique à Angular que la classe TypeScript
4 // en dessous est le code d'un composant
5 @Component({
6   // indique la balise HTML à utiliser pour insérer le
7   // composant dans l'application
8   selector: 'app-root',
9
10  // indique où se trouve le code HTML du composant (celui à
11  // insérer dans l'appli quand on insère le composant
12  templateUrl: './app.component.html',
13
14  // les styles propres au composant
15  styleUrls: ['./app.component.css']
16})
17 export class AppComponent {
18   // un composant contient une classe TypeScript qui sert
19   // à contenir ses données et sa logique
20}
```

AppComponent

Terminal TypeScript 3.5.3

6: TODO

Event Log

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 View:** Shows the project structure under "cours3". The "node_modules" folder is highlighted.
- Editor:** The "app.component.html" file is open, displaying the code: <h1>Ceci est mon composant App</h1>.
- Terminal:** The command `ng generate component courses` is being run, and the output shows the creation of files:
 - CREATE src/app/courses/courses.component.css (0 bytes)
 - CREATE src/app/courses/courses.component.html (22 bytes)
 - CREATE src/app/courses/courses.component.spec.ts (635 bytes)
 - CREATE src/app/courses/courses.component.ts (273 bytes)
 - UPDATE src/app/app.module.ts (479 bytes)

TypeScript du nouveau composant

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.ts - WebStorm
- Menu Bar:** File Edit View Navigate Code Refactor Run Tools VCS Window Help
- Toolbars:** Angular CLI Server, Git status icons.
- Project Structure:** Shows the project tree with the following structure:
 - Project: cours3 (~/enseignement/mobile-19-20/proj)
 - e2e
 - node_modules library root
 - src
 - app
 - courses
 - courses.component.css
 - courses.component.html
 - courses.component.spec.ts (highlighted)
 - courses.component.ts
 - app.component.css
 - app.component.html
 - app.component.spec.ts (highlighted)
 - app.component.ts
 - app.module.ts
 - app-routing.module.ts
 - assets
 - environments
 - favicon.ico
 - index.html
 - main.ts
 - polyfills.ts
 - styles.css
 - test.ts
- Code Editor:** The courses.component.ts file is open, showing the following code:

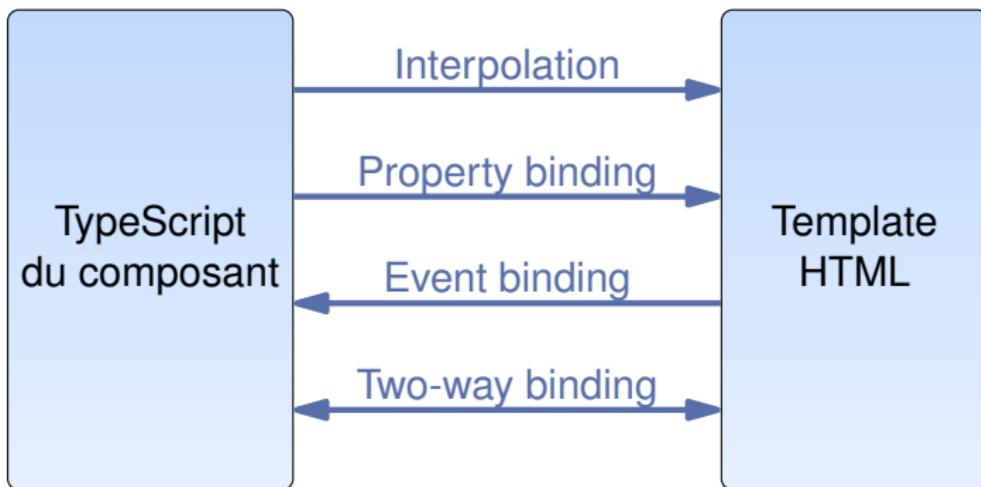
```
import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'app-courses', // balise d'insertion
  templateUrl: './courses.component.html',
  styleUrls: ['./courses.component.css']
})
export class CoursesComponent implements OnInit {

  // dans le constructeur, on ne met que des directives
  // d'affichage. Le constructeur doit s'exécuter RAPIDEMENT
  // donc pas de récupération de données de BD ici
  constructor() { }

  // cette méthode est appelée après le constructeur. Elle
  // sert, notamment, à initialiser des champs en récupérant
  // des données de bases de données
  ngOnInit() {
  }
}
```
- Bottom Status Bar:** Version Control (git icon), Terminal, TypeScript 3.5.3, TODO (6), Event Log (2).
- Bottom Status Bar (Details):** TSLint: The project code style and editor settings were updated base... (today 08:15) 22:1 LF UTF-8 EditorConfig Git: master.

Interactions entre le TypeScript et le HTML



Interpolation

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

- Project Structure:** On the left, the project structure shows the file tree for the "cours3" application, including 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, template URL, and style URLs. It also includes a constructor and a class definition for "CoursesComponent".
- Output/Console:** A large text box on the right contains two examples of interpolation:
 - "Ceci est mon composant App" followed by two instances of the "app-courses" component.
 - A detailed explanation of interpolation: "Interpolation : on insère un champ de la classe ou n'importe quelle expression javascript en spécifiant {{ expression }} -->". Below this, a sample output is shown: "Interpolation : {{ titre }} et {{ 3 + 4 }}".
- Bottom Status Bar:** Shows version control (9: Version Control), terminal (Terminal), TypeScript 3.5.3, TODO items (6: TODO), TSLint status, and event log (Event Log).
- Bottom Footer:** Shows the message "TSLint: The project code style and editor settings were updated base..." and the date/time "today 08:15".

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 file `courses.component.ts` selected.
- Code Editor:** The main editor shows the `CoursesComponent` class definition in TypeScript. It includes properties `titre` and `styleUrls`, a constructor, and methods `ngOnInit` and `getTitle`. The `getTitle` method returns `this.titre`.
- Preview:** A modal window titled "Ceci est mon composant App" displays the output of the component's template, showing the interpolation `list des cours`.
- HTML Editor:** Below the code editor, the `courses.component.html` template is shown, containing a single `<p>` tag with the interpolation `Interpolation bis : {{ getTitle() }}`.
- Bottom Status Bar:** Shows version control (git), terminal, typeScript 3.5.3, TODO items, and an event log.
- Bottom Status Bar (Details):** Shows TSLint status and commit information: "TSLint: The project code style and editor settings were updated based ... (today 08:15) 2:1 LF UTF-8 EditorConfig Git: master".

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. The `courses.component.ts` file is open in the editor.
- Editor:** The `courses.component.ts` file contains the following TypeScript code:

```
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  }
```
- Preview Window:** A large white box displays the text "Ceci est mon composant App". Below it, the text "Interpolation : liste des cours et 7" is followed by a text input field containing "valeur initiale".
- Editor:** The `courses.component.html` file is shown at the bottom, containing the following HTML code:

```
<!-- Property Binding : on insère une valeur dans une propriété d'un
     champ d'un élément du DOM. L'insertion se fait via :
     [nomPropriété]="valeur_dans_le_typescript" -->
<p> Interpolation : {{ titre }} et {{ 3 + 4 }}</p>
<input type="text" [value]="ma_valeur" />
```
- Bottom Status Bar:** Shows version control, terminal, TypeScript 3.5.3, TODO items, event log, and TSLint status.

► **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 : [disabled]="valeur"

Event binding

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

File Edit View Navigate Code Refactor Run Tools VCS Window Help

cours3 > src > app > courses > courses.component.ts

Angular CLI Server



1: Project

2: Favorites

3: npm

4: Structure

5: Test

courses.component.ts

```
5  templateUrl: './courses.component.html',
6  styleUrls: ['./courses.component.css']
7 }
8 export class CoursesComponent implements OnInit {
9   titre = 'liste des cours';
10
11   constructor() { }
12   ngOnInit() {
13
14     getTitle() { return this.titre; }
15
16     modifTitle() {
17       this.titre = 'nouveau titre';
18     }
19 }
```

CoursesComponent > getTitle()

courses.component.html

```
1  <p> Interpolation évaluée à runtime : {{ getTitle() }}</p>
2  <input type="text" (click)="modifTitle ()" />
3
```

input

4: Version Control

5: Terminal

6: TypeScript 3.5.3

7: TODO

8: Event Log

TSLint: The project code style and editor settings were updated bas... (today 08:15) 14:36 LF UTF-8 EditorConfig Git: master

Ceci est mon composant

Interpolation évaluée à runtime : nouveau titre

Interpolation évaluée à runtime : liste des cours



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 shown with the following folder hierarchy:
 - src
 - app
 - courses
 - courses.component.html
 - courses.component.ts
- courses.component.ts:** The code defines a `CoursesComponent` class that implements `OnInit`. It has a constructor and an `ngOnInit()` method.
- courses.component.html:** The template contains an `<p>` tag with interpolation and a two-way binding `<input type="text" [(ngModel)]="titre" />`.
- Browser DevTools:** A screenshot of the browser developer tools at `localhost:4200/` shows an 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" /> ")".

Two-way binding (2/2)

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

- Project:** cours3 (~/enseignement/mobile-19-20/prog/angular/cours3)
- File:** app.module.ts
- Toolbars:** Angular CLI Server, Git status icons.
- Code:** The code defines an NgModule with declarations, imports, and providers. It includes imports for AppRoutingModule, AppComponent, CoursesComponent, BrowserModule, AppRoutingModule, and FormsModule.
- Preview Area:** Two large text boxes labeled "Ceci est mon composant" show the evaluated runtime interpolation results:
 - The first box contains "Interpolation évaluée à runtime : liste des" followed by a blue-bordered input field containing "liste des".
 - The second box contains "Interpolation évaluée à runtime : liste des cours" followed by a blue-bordered input field containing "liste des cours".
- Bottom Status Bar:** Version Control, Terminal, TypeScript 3.5.3, TODO count (6), Event Log.
- Bottom Status Bar (Details):** TSLint message, Date (yesterday 08:15), Time (16:13), LF, UTF-8, EditorConfig, Git status (master), and a small icon.

Interactions TypeScript-template HTML en résumé

Uniquement de TypeScript vers le template HTML :

- ▶ **Property Binding** : [propriété] = "valeur"
affecte des valeurs aux propriétés d'éléments du DOM
- ▶ **Interpolation** : {{ champ ou méthode }}

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, and various app component files.
- Code Editor:** The main editor area displays the code for `courses.component.ts`. The code defines a `Course` interface and a `CoursesComponent` class that implements `OnInit`.

```
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 includes tabs for File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, Help. The status bar at the bottom shows "TypeScript 3.5.3", "Terminal", and "Event Log".

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

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

File Edit View Navigate Code Refactor Run Tools VCS Window Help

cours3 > src > app > courses > courses.component.htm Angular CLI Server

Project 1: Project

node_modules
src
app
course
course.com
course.com
course.com
course.com
course.com
course.ts
courses
courses.com
courses.com
courses.com
courses.com
app.component
app.component
app.component
app.component
app.module.ts
app-routing.m
assets
environments
favicon.ico
index.html

courses.component.html

```
<p> Interpolation évaluée à runtime : {{ getTitle() }}</p>
<input type="text" [(ngModel)]="titre" />
<app-course></app-course>
<app-course></app-course>
```

Cours3 - Chromium

Cours3 localhost:4200

Ceci est mon composant App

Interpolation évaluée à runtime : liste des cours

liste des cours

course works!

course works!

TODO Version Control TypeScript 3.5.3 Terminal Event Log

5:1 LF UTF-8 EditorConfig Git: master

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

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

- Title Bar:** cours3 [~/enseignement/mobile-19-20/prog/angular/cours3] - .../src/app/course/course.component.ts - WebStorm
- Menu Bar:** File Edit View Navigate Code Refactor Run Tools VCS Window Help
- Toolbars:** Angular CLI Server, Git status icons.
- Project Tree (1: Project):** node_modules, lib, src, app, course, courses, assets, environments, favicon.ico, index.html.
- Code Editor (course.component.ts):**

```
1 import {Component, Input, OnInit} from '@angular/core';
2 import { Course } from './course';
3
4 @Component({
5   selector: 'app-course',
6   templateUrl: './course.component.html',
7   styleUrls: ['./course.component.css']
8 })
9 export class CourseComponent implements OnInit {
10   // avec des @Input, on indique que, dans le HTML du parent,
11   // on va binder la propriété contenu de <app-course>. Cela
12   // permettra à la classe CourseComponent de recevoir les
13   // informations dont elle a besoin pour s'afficher correctement.
14   @Input() contenu_: Course; // Ici, il est important de
15                           // spécifier le type de contenu
16   constructor() { }
17   ngOnInit() {}
18 }
```
- Code Editor (course.component.html):**

```
<p>Cours : {{ contenu.titre }} -- {{ contenu.nb_etud }} étudiants </p>
```
- Bottom Status Bar:** TODO, Version Control, TypeScript 3.5.3, Terminal, Event Log.
- Bottom Right:** 5:26 LF UTF-8 EditorConfig Git: master

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`, `app`, `course`, `courses`, `app.component`, `app.module`, `app-routing`, `assets`, `environments`, `favicon.ico`, and `index.html`.
- Editor:** The main editor window shows the file `courses.component.html` with the following content:

```
<p> Interpolation évaluée à runtime : {{ getTitle() }}</p>
<input type="text" [(ngModel)]="titre" />
<!-- on passe les @Input par property binding --&gt;
&lt;app-course [contenu]="UE[0]"&gt;&lt;/app-course&gt;
&lt;app-course [contenu]="UE[1]"&gt;&lt;/app-course&gt;</pre>
```
- Terminal:** At the bottom, the terminal shows the command `TypeScript 3.5.3`.
- Browser Preview:** A preview window shows the application running at `localhost:4200`. The page title is "Cours3". The content area displays:

Ceci est mon composant App

Interpolation évaluée à runtime : liste des cours

Cours : c1 -- 2 étudiants

Cours : c2 -- 5 étudiants

Interactions enfant → parent : préparation de l'enfant

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

- Title Bar:** cours3 [~/enseignement/mobile-19-20/prog/angular/cours3] - .../src/app/course/course.component.html - WebStorm
- Menu Bar:** File Edit View Navigate Code Refactor Run Tools VCS Window Help
- Toolbar:** cours3 > src > app > course > course.component.html | Angular CLI Server | Git: master
- Project Tree:** 1: Project (course) | 2: Structure | 3: npm | 4: Favorites
- Editor:** course.component.html (Content)
- Code Content:**

```
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
```
- Bottom Bar:** TODO | 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 the WebStorm IDE interface with the file `course.component.ts` open. The code implements an Angular component (`@Component`) for a course. It includes an input property (`@Input()`) for course data and an output property (`@Output()`) for emitting new student counts. The `updateNb_` method updates the local `lastNb_` variable and emits the new count via the `newNb` event.

```
1 import { Component, Input, OnInit, Output, EventEmitter } from '@angular/core';
2 import { Course } from './course';
3
4 @Component({
5   selector: 'app-course',
6   templateUrl: './course.component.html',
7   styleUrls: ['./course.component.css']
8 })
9 export class CourseComponent implements OnInit {
10   @Input() contenu_: Course; // infos parent -> enfant
11   @Output() newNb = new EventEmitter<number>(); // infos enfant -> parent
12   lastNb_: number; // va servir à calculer le nombre que l'on émet vers le parent
13   constructor() { }
14
15   ngOnInit() { this.lastNb = this.contenu_.nb_etud; }
16   updateNb_() { // fonction appelée quand on change l'input
17     let nb = this.contenu_.nb_etud - this.lastNb;
18     this.lastNb = this.contenu_.nb_etud;
19     this.newNb.emit(nb); // transmet l'événement au parent
20   }
21 }
```

IDE navigation and status bars are visible at the bottom, including `File`, `Edit`, `View`, `Navigate`, `Code`, `Refactor`, `Run`, `Tools`, `VCS`, `Window`, `Help`, and `Angular CLI Server`.

Interactions enfant → parent : la réception du parent

The screenshot shows the WebStorm IDE interface with two open files: `courses.component.html` and `courses.component.ts`.

courses.component.html:

```
<p> Interpolation évaluée à runtime : {{ getTitle() }} : {{nb_etuds}}</p>
<input type="text" [(ngModel)]="titre" />

<app-course [contenu]="UE[0]" (newNb)="onNewNb($event)"></app-course>
<app-course [contenu]="UE[1]" (newNb)="onNewNb($event)"></app-course>
```

courses.component.ts:

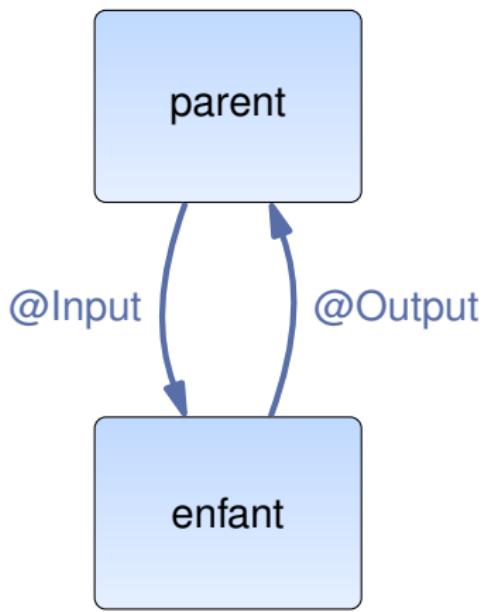
```
export class CoursesComponent implements OnInit {
  titre = 'liste des cours';
  UE: Course[] = [ {titre: 'c1', nb_etud: 2},
                   {titre: 'c2', nb_etud: 5} ];
  nb_etuds: number;
  constructor() { }
  ngOnInit() { this.getNbEtuds(); }
  getTitle() { return this.titre; }
  getNbEtuds()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

The code demonstrates how a parent component (`CoursesComponent`) receives events from child components (`app-course`) via event binding. The parent component has a method `onNewNb_` that is triggered whenever a child component emits a `newNb` event.

Interactions enfant → parent : le résultat

The image shows two adjacent browser windows, both titled "Cours3" and running on "localhost:4200". Each window displays a component with the heading "Ceci est mon composant App". Below the heading, there is a text message: "Interpolation évaluée à runtime : liste des cours : 7" in the left window and "Interpolation évaluée à runtime : liste des cours : 60" in the right window. Underneath this message is a text input field containing the text "liste des cours". Below the input field, there are two sections: "Cours : c1" and "Cours : c2". Each section contains an input field followed by the text "étudiants". In the left window, the "c1" input field contains the number "2", which is circled in red. In the right window, the "c1" input field contains the number "55", which is also circled in red. The "c2" input fields in both windows contain the number "5".

Résumé des interactions enfant – parent



Il existe d'autres types d'interactions (`ngOnChanges`, 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 *
- ▶ `*ngIf` : le composant est utilisé si et seulement si `ngIf=true`

La directive *ngFor

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:** cours3 > src > app > courses > courses.component.html
- Editor:** The code editor displays the file `courses.component.html`. The code uses interpolation and ngFor directives to display course information. A note at the bottom explains the limitation of applying two directives like ngIf and ngFor to the same component.
- Sidebar:** Project, Z. Structure, npm, Favorites.
- Bottom Status Bar:** 6: TODO, 9: Version Control, TypeScript 3.5.3, Terminal, Event Log.

```
<p> Interpolation évaluée à runtime : {{ getTitle() }} : {{nb_etuds}}</p>
<input type="text" [(ngModel)]="titre" />
<!-- directive *ngFor : on parcourt tous les éléments de UE. Chaque
élément est stocké dans la variable cours. On peut alors utiliser
cette variable, notamment dans les property bindings -->
<app-course *ngFor="let cours of UE"
    [contenu]="cours"
    (newNb)="onNewNb($event)"></app-course>

<!-- attention : on ne peut pas appliquer 2 directives (par exemple ngIf et ngFor
au même composant. Il faut choisir : soit utiliser ngIf, soit ngFor.
Si l'on a besoin des 2, l'astuce est de créer un composant supplémentaire
auquel on appliquera, par exemple, le ngFor, et ce composant contiendra le
composant qui nous intéresse, auquel on appliquera le ngIf -->
```

La directive `*ngIf`

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

- Project Bar:** Shows the project structure: `urs3` > `src` > `app` > `courses` > `courses.component.html`.
- File Menu:** File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, Help.
- Angular CLI Server:** Status bar indicator.
- Git:** Status bar indicators.
- Editor:** The code editor displays `courses.component.html`. The code uses interpolation and conditional logic (`*ngIf`) to display student counts. It also demonstrates the `ngFor` directive to iterate over course data.
- Sidebar:** Includes sections for Project, Z: Structure, npm, Favorites, and Event Log.
- Bottom Bar:** Shows tabs for TODO, Version Control, TypeScript 3.5.3, Terminal, and Event Log.

```
<p> Interpolation évaluée à runtime : {{ getTitle() }} : {{nb_etuds}}</p>
<div>
  <!-- Ici, la page web ne contiendra qu'un seul des paragraphes ci-dessous.
      Si Courses contient moins de 10 étudiants, ce sera le 1er paragraphe,
      sinon, ce sera le 2ème. Quand je dis que "la page web contiendra...", 
      je parle du DOM : le DOM ne contiendra effectivement qu'un <p>, et
      non 2 <p> avec l'un des deux hidden. -->
  <p *ngIf="nb_etuds < 10">peu d'étudiants</p>
  <p *ngIf="nb_etuds >= 10">beaucoup d'étudiants</p>
</div>
<input type="text" [(ngModel)]="titre" />
<!-- directive *ngFor : on parcourt tous les éléments de UE. Chaque
     élément est stocké dans la variable cours. On peut alors utiliser
     cette variable, notamment dans les property bindings -->
<app-course *ngFor="let cours of UE"
             [contenu]="cours"
             (newNb)="onNewNb($event)"></app-course>
```

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



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

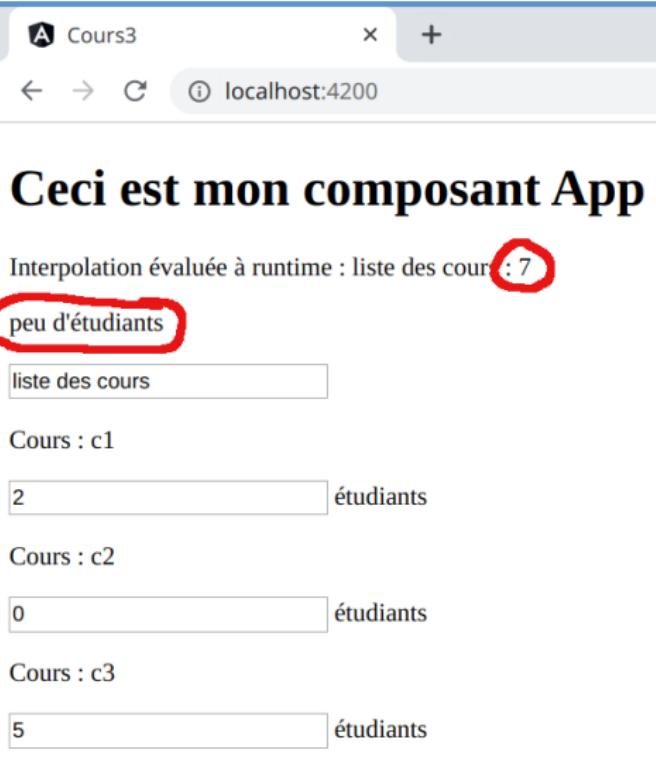
2 étudiants

Cours : c2

0 étudiants

Cours : c3

5 étudiants



- ▶ 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:** cours3 (~/enseignement/mobile-19-20/prog/angular/cours3)
- File:** courses.service.ts
- Toolbars:** File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, Help
- Status Bar:** Angular CLI Server, Git status (master), Event Log
- Side Panels:** 1: Project, 2: Favorites, 3: Structure, 4: npm
- Code Content:**

```
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
```

Exploitation du service dans le composant Courses

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

- Project Tree:** On the left, there's a tree view labeled "1: Project" showing the project structure: "cours3" > "src" > "app" > "courses".
- Editor:** The main editor area displays the code for "courses.component.ts".
- Code Content:**

```
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 getCourses pour pouvoir remplir notre tableau UE
18    let service = new CoursesService();
19    this.UE = service.getCourses();
20  }
21  ngOnInit() { this.getNbEtuds(); }
```
- Toolbars and Status Bar:** The top bar includes "File Edit View Navigate Code Refactor Run Tools VCS Window Help". The status bar at the bottom shows "cours3 [~/enseignement/mobile-19-20/prog/angular/cours3] - .../src/app/courses/courses.component.ts - WebStorm", "Angular CLI Server", "Git: master", and "Event Log".

Exploitation du service : bonnes et mauvaises idées

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

File Edit View Navigate Code Refactor Run Tools VCS Window Help

cours3 > src > app > courses > courses.component.ts

Angular CLI Server



1: Project

2: Structure

3: npm

2: Favorites

```
5
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 toute l'appli
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;
```

6: TODO

9: Version Control

TypeScript 3.5.3

Terminal

Event Log

18:53 LF UTF-8 EditorConfig Git: master

Asynchronie et observables

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



Utilisation de services asynchrones :

- ➊ Le service retourne tout de suite un ***Observable***
- ➋ `ngOnInit` appelle le service et récupère l'observable
- ➌ `ngOnInit` souscrit à l'observable en donnant une callback
- ➍ `ngOnInit` continue son exécution
- ➎ 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 [~/enseignement/mobile-19-20/prog/angular/cours3] - .../src/app/services/courses.service.ts - WebStorm
- File:** courses.service.ts
- Toolbars:** File Edit View Navigate Code Refactor Run Tools VCS Window Help
- Status Bar:** Angular CLI Server, Git: master
- Left Sidebar:** Project (1: Project), npm, Favorites (2: Favorites)
- Code Editor:** The code for `courses.service.ts` is displayed:

```
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 Bar:** TODO, Version Control, TypeScript 3.5.3, Terminal, Event Log

Mise en place des observables dans le composant

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

- Project Tree:** On the left, there's a tree view showing the project structure with nodes for '1: Project', '2: Favorites', 'npm', and '2: Structure'.
- Editor:** The main editor area displays the code for `courses.component.ts`. The code implements the `OnInit` interface and uses an observable from a service to initialize the component.
- Toolbars and Status Bar:** At the top, there's a toolbar with icons for File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, and Help. Below the toolbar, the status bar shows the path `cours3 [~/enseignement/mobile-19-20/prog/angular/cours3] - .../src/app/courses/courses.component.ts - WebStorm`, the Angular CLI Server status, and other system information like the terminal type (`TypeScript 3.5.3`), event log, and git status.

```
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         // renournée 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 }
```

Service HTTP : 1 importer le HttpClientModule

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

- Project Structure:** The left sidebar shows the project structure with files like `node_modules`, `src`, `app`, `course`, `courses`, `services`, `app.component.css`, `app.component.html`, `app.component.spec.ts`, `app.module.ts`, `app-routing.module.ts`, and `assets`.
- Code Editor:** The main editor window displays the `app.module.ts` file. The code includes imports for `CoursesComponent`, `FormsModule`, `CourseComponent`, and `HttpClientModule`. The `HttpClientModule` import is highlighted with a red circle icon.
- Toolbars and Status Bar:** The top bar shows "Angular CLI Server" and various icons. The bottom status bar indicates "TypeScript 3.5.3", "Event Log", and other development information.

```
6 import { CoursesComponent } from './courses/courses.component';
7
8 import { FormsModule} from '@angular/forms';
9 import {CourseComponent} from './course/course.component';
10
11 // pour utiliser HttpClient, importer son module
12 import {HttpClientModule} from '@angular/common/http';
13
14 @NgModule({
15   declarations: [
16     AppComponent,
17     CoursesComponent,
18     CourseComponent
19   ],
20   imports: [
21     BrowserModule,
22     AppRoutingModule,
23     FormsModule,
24     HttpClientModule // importer ici le module HttpClient
25   ],
26   providers: []
27 })
```

Service HTTP : 2 notre service utilise HttpClient

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

- Project:** cours3 (~/enseignement/mobile-19-20/prog/angular/cours3)
- File:** courses.service.ts
- Toolbars:** File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, Help
- Status Bar:** Angular CLI Server, Git: master, Event Log
- Side Panels:** Project (1: Project), npm, Favorites (2: Favorites), Z: Structure.
- Code Content:** The code defines a CoursesService class using the HttpClient module to interact with a local server.

```
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})
11export 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 }
```

Service HTTP : ③ 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

3: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 is a text input field containing "liste des cours". The browser's developer tools are open, specifically the "Console" tab. The console log shows the following entries:

- 4 messages
- 3 user mess...
- 2 errors
- No warnings

Details of the error message:
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.

- ▶ **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 : 3 le bon 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



getCourses.php

```
1 <?php
2
3 // le script doit renvoyer les données au format JSON
4 header('Content-type:application/json;charset=utf8');
5
6 // ici, on rajoute un header pour contourner le problème CORS
7 // A noter que ce n'est à faire ici que parce qu'on utilise Angular en front-end
8 // sur le port 4200 et PHP en back-end sur le port 80.
9 header("Access-Control-Allow-Origin: *");
10
11 echo json_encode([
12     [ 'titre' => 'c1', 'nb_etud' => 2 ],
13     [ 'titre' => 'c2', 'nb_etud' => 5 ],
14     [ 'titre' => 'c3', 'nb_etud' => 6 ]
15 ]);
16
17 ?>
```

1:Project
2: Favorites

npm

2: Structure

6: TODO

9: Version Control

TypeScript 3.5.3

Terminal

Event Log

18:1 LF UTF-8 4 spaces Git: 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

Console Sources Network Performance Memory Application Security Audits

top Filter Default levels

2 messages >

2 user mess...

No errors

Service HTTP : 2 HttpClient avec méthode POST

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

- Project:** cours3 (~/enseignement/mobile-19-20/prog/angular/cours3)
- File:** courses.service.ts
- Toolbars:** File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, Help
- Angular CLI Server:** Selected
- Code Content:**

```
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[]>(` // on converse avec le serveur via la méthode POST
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 }
```
- Side Panels:** Project, npm, Favorites
- Bottom Bar:** TODO, Version Control, TypeScript 3.5.3, Terminal, Event Log
- Status Bar:** 23:1 LF UTF-8 EditorConfig Git: master

- 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 : utiliser des routes dans app.module.ts

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

- Title Bar:** cours3 [~/enseignement/mobile-19-20/prog/angular/cours3] - .../src/app/app.module.ts - 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, 2: Structure, 3: npm, 4: Favorites.
- Code Editor:** The file `app.module.ts` is open, showing the following code:

```
// ici, on indique les routes de l'application. path = l'URL pour accéder au composant
const myRoutes_: Routes = [
  { path: 'cours', component: CoursesComponent },
  { path: 'sujets', component: TopicsComponent}
];

@NgModule({
  declarations: [
    AppComponent,
    CoursesComponent,
    CourseComponent,
    TopicsComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule, // l'appli doit avoir la capacité de faire du routing
    FormsModule,
    HttpClientModule,
    RouterModule.forRoot(myRoutes) // on indique quelles routes sont utilisées par l'appli
  ],
  providers: []
})
```

- Bottom Status Bar:** TODO, Version Control, TypeScript 3.5.3, Terminal, Event Log.
- Bottom Right:** 32:76 LF UTF-8 EditorConfig Git: master

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
- Toolbars:** Angular CLI Server, Git status icons.
- Project Tree:** 1: Project, 2: Favorites, 3: Structure, 4: npm.
- Code Editor:** The file "courses.component.html" is open. The code uses RouterLink to navigate between pages without reloading the application. It includes interpolation for dynamic titles and student counts, and an ngIf directive for conditional rendering based on student count. An input field is bound to a model named "titre". A ngFor loop iterates over "cours" (UE) and passes "contenu" and "newNb" to an "app-course" component.

```
<!-- on rajoute ici une barre de navigation. Surtout, ne pas
utiliser href pour passer d'une page à l'autre : cela
rechargerait toute l'application. Ici, il faut utiliser
la propriété RouterLink qui permet à Angular de modifier
la vue sans recharge. -->
<nav>
  <a routerLink="/sujets">Mes sujets</a>
</nav>

<p> Interpolation évaluée à runtime : {{ getTitle() }} : {{nb_etuds}}</p>
<div>
  <p *ngIf="nb_etuds < 10">peu d'étudiants</p>
  <p *ngIf="nb_etuds >= 10">beaucoup d'étudiants</p>
</div>
<input type="text" [(ngModel)]="titre" />
<app-course *ngFor="let cours of UE"
            [contenu]="cours"
            (newNb)="onNewNb($event)"></app-course>
```

- Bottom Status Bar:** TODO, Version Control (9), TypeScript 3.5.3, Terminal, Event Log, 5:21 LF UTF-8 EditorConfig Git: master.

Les liens paramétrés (1/2)

The screenshot shows the WebStorm IDE interface with two open files:

- app.module.ts**:

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

The IDE features a sidebar with project navigation, a status bar at the bottom, and various toolbars and status indicators.

Les liens paramétrés (2/2)

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

File Edit View Navigate Code Refactor Run Tools VCS Window Help

Project: cours3 / src / app / topics / topics.component.ts Angular CLI Server

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
```

localhost:4200/sujets/25

Ceci est mon composant App

topic: 25

File 6: TODO 9: Version Control 10: TypeScript 3.5.3 11: EditorConfig 12: Git: master