Angular

Les concepts essentiels

Angular
L’essentiel des concepts

- Angular est un framework qui permet de construire des applications front-end web et mobile type SPA (Single Page Application). Typiquement une application SPA ne recharge jamais la page courante mais simplement une ou plusieurs sections de la page courante. L’intérêt est de présenter des application web bien plus fluide et rapide pour l’utilisateur.

- Il propose une solution pour organiser votre HTML/CSS/JS et construire des applications front-end scalables et maintenables à partir du langage Javascript et du superset TypeScript.

1 Le data binding (la liaison de données)

- Sur Angular la manière de réfléchir une interface réactive est basée sur la donnée, localisée dans la class du component.
- Angular se charge d'afficher les données et de rendre l'interface réactive lorsque la valeur de cette donnée est modifiée grâce à de mécanismes

En Vanilla JS, on modifie l'interface en sélectionnant des éléments du DOM et en appliquant des modifications à l'aide de propriétés comme .textContent, .innerHTML, .classList, .style, etc…

En utilisant Angular on laisse le framework gérer la partie sélection/modification du DOM.

Data-binding : l'interpolation avec {{ }}

- permet d'afficher des données
- rendre l'interface réactive au changement

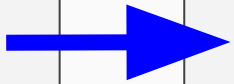
{{ maVariable }} permet d'afficher la valeur d'une variable dans le template HTML

app.component.ts

```
class AppComponent {

        name = 'Fred';


}
```



app.component.html

```
<h1>Hello {{ name }}</h1>
```

https://angular.io/guide/interpolation

Data-binding : la liaison de propriété

- permet d'afficher la valeur d'une propriété dans HTML
- rendre cette valeur réactive au changement de donnée

[nomDePropiete]="valeur"

app.component.ts

```
class AppComponent {

        name = 'Fred';
        image = 'mario.jpg';


}
```

app.component.html

```
<h1>Hello {{ name }}</h1>

<img [src]="image" />


(en dur on retrouvera : <img src="mario.jpg" />)
```

https://angular.io/guide/property-binding

Les 2 façons d'écrire l'affichage de variables dans un template de vue sont sécurisées.
Angular nettoie la chaîne de caractère et empêchera l'injection de balises script par exemple.

String interpolation ou property-binding

String interpolation
{{ variable }}

- facile et rapide pour afficher la valeur d'une variable en format texte uniquement
- peut interpréter des expressions et méthodes Javascript qui retournent une valeur
- affiche quoi qu'il arrive une chaîne de caractère de type texte (équivalent à textContent en Javascript)

Avec le property-binding
<p [innerHTML]="variable">

- la liaison de type property-binding attend une variable côté TS
- ICI, la propriété innerHTML permet d'afficher du HTML, ce que ne permet pas le string interpolation

NB : les propriétés sont différentes des simples attributs HTML. Les propriétés accèdent au DOM, alors que les attributs HTML renseignent le html.

Data-binding : l'événement

- permet de réagir aux interactions utilisateur
- passer des données du HTML à une méthode de la class

(click)=”nomDeMethode(param)”

app.component.ts

```
class AppComponent {

    deleteTask(idTask) {
        this.tasks.find(task => {
            return task.id == idTask;
        })
    }

}
```

app.component.html

```
<ul>
<li *ngFor=”let t of tasks”>
    {{ t.name }}
    <button (click)=”deleteTask(t.id)”>
        Supprimer
    </<button>
</li>
</ul>
```

https://angular.io/guide/event-binding

2 Les directives

Angular définit les directives comme étant **l'extension du langage HTML**.

Elles sont de 3 types :

- ***les components*** (sont représentés et instanciés par des balises HTML)
- ***les directives d'attributs*** "*modifient l'apparence et le comportement des éléments du DOM*"
- ***les directives de structure***, permettent de modifier la structure du DOM

(On peut créer nos propres directives : https://angular.io/guide/attribute-directives)

Les directives : structural directives

- *ngIf / *ngSwitch - permet d'afficher ou masquer des éléments du DOM
- *ngFor - permet d'itérer des éléments dans le DOM

***ngIf="truthy expression"**

app.component.ts

```
class AppComponent {

        isVisible = false;
        consoles = [ 'xbox', 'switch', 'play']

}
```

***ngFor="let item of array"**

app.component.html

```
<p *ngIf="isVisible">
        Je suis un paragraphe masqué
</p>
<ul>
        <li *ngFor="let console of consoles; let
        i = index">
                {{i}} - {{ console }}
        </li>
</ul>
```

https://angular.io/guide/structural-directives

Les directives : ngClass et ngStyle

- ngClass - permet d'ajouter ou supprimer dynamiquement des class css
- ngStyle - permet d'ajouter ou supprimer dynamiquement du style css

[class.nomDeLaClass]="truthy expression" appliquera nomDeLaClass à l'elt HTML

user.component.ts

```
class UserComponent {

        user = {
                name:fred,
                isPremium: :true
        }

}
```

user.component.html

```
<p [class.green]="isPremium"
    [class.black]="!isPremium"]>
        Bonjour {{ name }}
</p>
```

https://angular.io/guide/attribute-binding#class-binding

Les directives : ngClass et ngStyle

- ngClass - permet d'ajouter ou supprimer dynamiquement des class css
- ngStyle - permet d'ajouter ou supprimer dynamiquement du style css

[style.nomDeLaProprieteCSS]="expression" appliquera du style CSS inline à l'elt HTML

user.component.ts

```
class UserComponent {

        bkg = '#000000';
        isPremium = true;
        getColor(isPremium) {
                return this.isPremium?'green':'red'
        }
}
```

user.component.html

```
<p [style.color]="bkg"
[style.color]="getColor(isPremium)">
        Bonjour {{ name }}
</p>
```

https://angular.io/guide/attribute-binding#style-binding

2 Les pipes

Angular définit un pipe comme étant **une fonction de formatage d'affichage des données dans le template HTML**.

- ***Syntaxe :*** {{ variableAAfficher | nomDuPipe}}
- ***Les pipes natifs :*** uppercase, lowercase, titlecase, date, decimal, json, currency, titlecase, async, …
- ***Possibilité de construire vos propres pipes :*** ng generate nomDuPipe

https://angular.io/api/?type=pipe

Les pipes - exemple

- | uppercase - permet de transformer une string en majuscules
- | date - permet de formater une date

{{ VariableAAfficher | nomDuPipe }} ” appliquera une transformation avant l’affichage

app.component.ts

```
class AppComponent {

        name = “fred lo”;
        today = new Date()
}
```

app.component.html

```
<p>
        Bonjour {{ name | uppercase}}
        Nous somme le {{ today | date:’dd/MM/yyyy’ }}
</p>
```

(Bonjour FRED LO NOUS SOMMES LE 27/09/2020)

https://angular.io/api/?type=pipe

Les pipes - créer un custom pipe

- ng generate pipe nomDuCustomPipe - pour créer un pipe
- {{ variable | nomDuCustomPipe }}

{{ VariableAAfficher | nomDuCustomPipe }} ” appliquera notre custom pipe

app.component.ts

```
class LastStringInUpperCase {

  function transform(value) {
        let words = value.split(“ “);
        let spaces = words.length - 1;
        let lastWord = words[value.length - 1];
        lastWord= lastWord.toUpperCase();
  }
}
```

app.component.html

```
<p>
        Bonjour {{ name | laststringuppercase}}
        Nous somme le {{ today | date:’dd/MM/yyyy’ }}
</p>
```

(Bonjour FRED LO NOUS SOMMES LE 27/09/2020)

https://angular.io/api/?type=pipe

Le cycle de vie d'un component Angular

- **ngOnChanges(), ngOnInitit(),**
- **ngDoCheck(), ngAfterContentInit(), ngAfterViewInit()**
- **ngOnDestroy()**

Les hooks couramment utilisés
Exemple de cas d'utilisation

ngOnChanges() s'exécute en premier, et à chaque fois qu'il y aura un changement en input dans le component. **Détecter des changements**

ngOnInit() s'exécute après ngOnChanges(), une seule fois. **Initialiser les valeurs du component**

ngDoCheck() il est appelé après chaque détection de changement. **Détecter "tous" les changements**

ngAfterContentInit() il est appelé une fois que le contenu externe est projeté dans le composant (transclusion). fonctionne avec <ng-content>

ngAfterViewInit() s'exécute dès lors que la vue du composant ainsi que celle de ses enfants sont initialisés **Accéder au DOM**

ngOnDestroy() s'exécute juste avant la destruction de l'objet du component en mémoire. **Annuler des subscribe()**

https://blogs.infinitesquare.com
https://angular.io/guide/lifecycle-hooks