

Angular 4

Interpolation

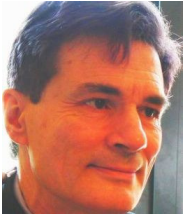
vertika.org by Jean Garutti

-- Angular 4 --

03/09/2017 -- 1

info@vertika.org

Interpolation Angular



- Une vraie bombe...
- C'est vraiment simple

Jean Garutti – vertika.org

by Jean Garutti

-- Angular 4 --

-- 2

info@vertika.org

c'est le pilier d'Angular



by Jean Garutti

-- Angular 4 --

-- 3

Interpolation, qu'est-ce que c'est ?

- C'est juste un mot chic pour « afficher des données dans le *Template* ».
- Ces données sont habituellement définies dans la classe des composants.
- Lorsque vous utilisez la **ANGULAR CLI** pour générer un projet **ANGULAR**,
 - elle affiche une propriété de **title** par interpolation
 - dans le fichier *Template app.component.html*.

propriété de **title** par interpolation

- Cela ressemble à ceci:

<h1> {{title}} </h1>

Interpolation, fonctionnement

- L'interpolation fonctionne en enveloppant des *crochets doubles* autour d'une expression du **Template**.
- Une expression du **Template** peut représenter
 - une propriété de composant
 - ou même des équations mathématiques
 - **<p>The sum of 1 + 2 is {{1 + 2}}</p>**

Interpolation, fonctionnement

- L'expression peut invoquer les méthodes du composant hôte tel que `getVal()` :

```
<p>The sum of 1 + 2 is not
  {{1 + 2 + getVal() }}</p>
```

Interpolation, fonctionnement

src/app/app.component.html

1. `<h3> {{title}}`
2. ``
3. `</h3>`

Interpolation, fonctionnement

- Le texte entre les accolades est souvent le nom d'une propriété de composant.
- **Angular** remplace ce nom par la valeur **string** de la propriété de composant correspondante.

info@vertika.org

Interpolation, fonctionnement

- Dans l'exemple ci-dessus, **Angular** évalue les propriétés **title** et **ImageUrl** et "remplit les espaces",
 - affichant d'abord le titre d'application
 - puis une image.

by Jean Garutti

-- Angular 4 --

-- 10

info@vertika.org

Interpolation, fonctionnement

- Une expression de modèle produit une valeur.
 - **Angular** exécute l'expression et l'affecte à une propriété de la cible

by Jean Garutti

-- Angular 4 --

-- 11

info@vertika.org

Interpolation, fonctionnement

- La cible peut être
 - un élément HTML
 - un composant
 - ou une directive.

by Jean Garutti

-- Angular 4 --

-- 12

Interpolation, fonctionnement

- Les accolades d'interpolation dans `{{1 + 1}}` entourent l'expression de **template** `1 + 1`.
- une expression de **template** apparaît entre côtes à droite du symbole `=`
`[propriété] = "expression"`.

Interpolation, fonctionnement

- Vous écrivez ces expressions de **template** dans une langue qui ressemble à **JavaScript**.
- De nombreuses expressions **JavaScript** sont des expressions correctes, mais pas toutes.

Interpolation, interdits

- Les expressions JavaScript qui ont
 - ou favorisent
- les effets secondaires
- sont interdites, y compris:
 - affectations (`=`, `+=`, `-=`, ...)
 - `new`
 - Enchaînement d'expressions avec `;` ou `,`
 - Opérateurs d'incrément et de décrémentation (`++` et `--`)

Interpolation, interdits

- Autre différence notable par rapport à la syntaxe **JavaScript** :
- *aucun support*
 - pour les opérateurs bit à bit
| et &

Interpolation, nouveautés

- Nouveaux opérateurs d'expression de **template**,
- tels que |. ?. et !.

Interpolation, interdits

- Le contexte d'expression est généralement l'instance de composant.
- **Src / app / app.component.html**

```

{{Titre}}
<Span [hidden] =
"isUnchanged"> changé </
span>

```

Interpolation, interdits

- Le **title** dans les accolades doubles et l'option **isUnchanged** entre guillemets se réfèrent aux propriétés de l'**AppComponent**.

Interpolation, fonctionnement

- Une expression peut également se référer à des propriétés du contexte du **template**,
 - telles qu'une variable d'entrée de modèle (**let hero**)
 - ou une variable de référence de modèle (**#heroInput**).
- `src / app / app.component.html`
- ```
<div *ngFor = "let hero of heroes">
 {{hero.name}} </ div>
<Input #heroInput> {{heroInput.value}}
```

## Interpolation, contexte

- Le contexte pour les termes dans une expression est un mélange des variables
  - de **template**,
  - de l'objet contextuel de la **directive** (s'il en a une)
  - et des membres du composant.

## Interpolation, contexte

- Si vous faites référence à un nom qui appartient à plus d'un de ces espaces de noms,
  - ☐ le nom de la variable **template** a priorité,
  - ☐ suivi d'un nom dans le contexte de la directive
  - ☐ et, enfin, des noms des membres du composant.

by Jean Garutti

-- Angular 4 --

-- 22

---

---

---

---

---

---

---

---

## Expression, portée

- Les expressions de **template** ne peuvent se référer à rien dans l'espace de noms global.
- Ils ne peuvent pas se référer à une **fenêtre** ou à un **document**.

by Jean Garutti

-- Angular 4 --

-- 23

---

---

---

---

---

---

---

---

## Expression, portée

- Ils ne peuvent pas appeler **console.log** ou **Math.max**.
- Ils se limitent à référencer les membres du contexte de l'expression.

by Jean Garutti

-- Angular 4 --

-- 24

---

---

---

---

---

---

---

---



## Interpolation, pour objet

- Si la propriété de composant se compose d'un **objet**,
- vous pouvez faire référence à *une propriété spécifique* de l'objet comme ceci :

## Interpolation, pour objet

```

1. Component({
2. // Other component properties removed
3. template: `
4. <h1>Hey guys!</h1>
5. <p>{{ myObject.gender }}</p>
6. `,
7. })

8. export class AppComponent {
9. myObject = {
10. gender: 'male',
11. age: 33,
12. location: 'USA'
13. };
14. }

```

## Expression, règles à respecter

- **Aucun effet secondaire visible**
- **Exécution rapide**
- **Simplicité**
- **Idempotence**

## Expression, règles à respecter

- Les expressions de **template** peuvent créer ou casser une application. Suivez les règles:
- Les seules exceptions à ces règles devraient être dans des circonstances particulières
  - que vous comprenez à fond.

---

---

---

---

---

---

---

Merci !

Thanks !

---

---

---

---

---

---

---