

vertika.org by Jean Garutti

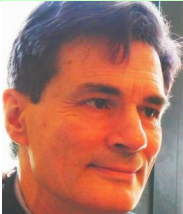
Angular 4

CSS
Angular

-- Angular 4 --03/09/2017 -- 1

info@vertika.org

CSS Angular



- *Installez l'élégance agréable*
- *Sous-traitez Angular à Angular*

Jean Garutti – vertika.org

by Jean Garutti-- Angular 4 -- 2

info@vertika.org

appliquez VOTRE maquillage



by Jean Garutti-- Angular 4 -- 3

CSS

- Lorsque vous utilisez la **CLI** pour générer un projet Angular,
- par défaut, elle configurera le projet pour fonctionner avec un CSS standard.

by Jean Garutti

-- Angular 4 --

-- 4

CSS

- Toutefois, si vous souhaitez utiliser Sass ou SCSS, vous ajoutez le préfixe **style** :
 - // Démarrage d'un projet avec CSS standard
> **ng new project-name**
 - // Démarrage d'un projet avec Sass
> **ng new project-name --style = sass**
 - // Démarrage d'un projet avec SCSS
> **ng new project-name --style = scss**

by Jean Garutti

-- Angular 4 --

-- 5

CSS

- Nous avons déjà généré un projet avec CSS standard dans ce cours,
- nous continuerons à supposer que vous l'avez généré sans le préfixe de style.

by Jean Garutti

-- Angular 4 --

-- 6

CSS en ligne / externe

- Lorsque vous utilisez la CLI pour générer un projet,
 - par défaut, il configurera vos composants pour fonctionner
 - avec des feuilles de style *externes*.

CSS en ligne / externe

- Que vous utilisiez des CSS internes ou externes, ils sont tous deux définis dans le *décorateur de composants*.

CSS externe

- Une feuille de style *externe* est définie par la propriété **styleUrls**:

```
@Component({  
  // Autres propriétés  
  supprimées  
  styleUrls:  
    [ './app.component.css' ]  
})
```

CSS externe

- Tous les CSS spécifiques au modèle de ce composant

seront placés dans un fichier CSS distinct.

CSS externe

- C'est utile lorsque vous avez beaucoup de CSS et vous ne voulez pas que CSS encombre le code du composant.

CSS en ligne

- Pour écrire un CSS *en ligne* changez la propriété **styleUrls** en **styles**.

```
@Composant({
  // Autres propriétés supprimées
  styles: [
    h1 {
      text-decoration: underline;
    }
  ]
})
```

CSS en ligne

- Vous devez utiliser des backticks pour définir des CSS multi-lignes,
 - et vos déclarations CSS sont écrites en ligne.

CSS en ligne

- C'est utile lorsque vous avez très peu de déclarations CSS,
 - ou préférez écrire des CSS en ligne dans le composant.

Changement du CSS

- Parfois, vous devez changer le CSS à la volée.
- Angular vous permet d'*ajouter* ou de *supprimer* des classes CSS en fonction de la logique des composants.

Changement du CSS

- Pour modifier les classes CSS, dans votre modèle `app.component.html`:

```
<h1
  [class]="titleClass">
  {{title}}
</h1>
```

Changement du CSS

- Ensuite, dans la classe de composants, nous définissons une propriété `titleClass`:

```
export class AppComponent {
  title = 'Hello!';
  titleClass = 'red-title';
}
```

Changement du CSS

- Nous avons lié `titleClass` à une chaîne appelée `red-title`, (le nom d'une classe CSS que nous devons définir dans la propriété `styles`):

```
styles: [`
  h1 {
    text-decoration: underline;
  }
  .red-title {
    color: red;
  }
]
```

Changement du CSS

- si vous exécutez **ng serve** sur la console pour voir le projet à **http: //localhost:4200**, vous verrez que le texte est rouge.

Changement du CSS

- La liaison de classe est utile dans un *contexte réel* car elle vous permet de contrôler quelle classe CSS est attachée à la propriété à laquelle cette classe est liée.

Changement du CSS

- Vous pouvez également désigner la classe CSS directement dans la classe de liaison du modèle:

```
<h1 [class.red-  
  title]="titleClass">  
  {{title}}  
</h1>
```

Changement du CSS

- **titleClass**
dans le code du composant
peut être lié à une certaine valeur,
 - ☐ **true** ou **false**,
 - ☐ ce qui entraînera la suppression de la classe de l'élément.
- Modifiez **titleClass** en false
et le titre ne sera plus rouge.

Changement du CSS par Style Binding

- La modification de l'attribut **style**
est une autre forme
de liaison de propriété.
- En utilisant l'exemple ci-dessus,
modifiez le modèle ainsi :

```
<h1 [style.color]="titleStyle">
  {{title}}
</h1>
```

Changement du CSS par Style Binding

- Vous pouvez également contrôler
la valeur de la propriété CSS spécifiée,
dans l'expression du modèle lui-même:

```
<h1 [style.color]="titleStyle ?
  'green' : 'pink'">
  {{title}}
</h1>
```


Changement du **CSS** par Style Binding

- Si **titleStyle** est défini, il apparaîtra en vert.
- Sinon, s'il n'existe pas, ou est **false**, il apparaîtra en rose.

Modification de *plusieurs* styles et classes **CSS**

- Vous pouvez lier les directives **ngStyle** et **ngClass** à une propriété.
- Cette propriété peut contrôler plusieurs styles ou classes.

Modification de *plusieurs* styles et classes **CSS**

- Modifiez le modèle HTML en :

```
<h1 [ngClass]="titleClasses">
  {{title}}
</h1>
```

Modification de *plusieurs* styles et classes **CSS**

- Ensuite, modifiez la classe de composants en :

```
titleClasses = {  
  'red-title': true,  
  'large-title': true  
}
```

Modification de *plusieurs* styles et classes **CSS**

- Dans la propriété **styles** du décorateur de composants, ajoutez la classe de gros titres :

```
.large-title {  
  font-size:4em;  
}
```

Modification de *plusieurs* styles et classes **CSS**

- Maintenant, les deux classes qui sont définies comme étant **vraies** dans l'objet **titleClasses** seront attachées à l'élément **h1**.

Modification de *plusieurs* styles et classes **CSS**

- C'est la même approche pour changer de styles multiples par la directive **ngStyle** :

```
<h1 [ngStyle]="titleStyles">
  {{title}}
</h1>
```

by Jean Garutti

-- Angular 4 --

-- 31

Modification de *plusieurs* styles et classes **CSS**

- Et dans la classe de composants:

```
titleStyles = {
  'color': 'red',
  'font-size' : '4em'
}
```

by Jean Garutti

-- Angular 4 --

-- 32

Modification de *plusieurs* styles et classes **CSS**

Dans ce cas, l'objet contient les propriétés CSS actuelles et leurs valeurs.

by Jean Garutti

-- Angular 4 --

-- 33

Encapsulation **CSS** dans le composant

**L'encapsulation CSS a toujours été
un outil
que les développeurs ont voulu
dans
leurs applications Web.**

Encapsulation **CSS** dans le composant

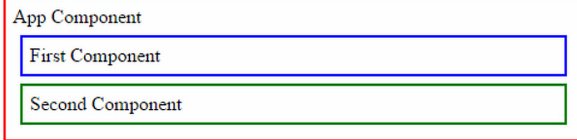
**La capacité d'appareiller CSS à un
composant spécifique
sans en affecter d'autres
a toujours été difficile à réaliser.**

- Voyons ici les avantages et les inconvénients de ces techniques.

Encapsulation **CSS** dans le composant

- Dans votre application Angular, vous aurez trois composants
 - le composant de l'application,
 - deux composants fils,
 - FirstComponent
 - SecondComponent.

Encapsulation CSS dans le composant



Encapsulation CSS dans le composant

```
import { Component } from '@angular/core';
@Component({
  selector: 'demo-app',
  template: `
    <h3>CSS Encapsulation with Angular</h3>
    <div class="cmp">
      App Component
      <first-cmp></first-cmp>
      <second-cmp></second-cmp>
    </div>
  `
})
export class App {
  constructor() { }
```

Encapsulation CSS dans le composant

- Nous associons une feuille de style distincte à **index.html**
 - ☐ qui a une seule règle de style global
 - ☐ qui sera appliquée
 - ☐ à n'importe quel élément
 - ☐ avec la classe CSS **cmp**.

Encapsulation CSS dans le composant

```
.cmp {
  padding: 6px;
  margin: 6px;
  border: 2px solid red;
}
```

Encapsulation CSS Techniques de style

- Regardons
 - ☐ FirstComponent SecondComponent
 - ☐ et voyons les options que nous avons
 - ☐ pour styler ces composants.

Encapsulation CSS dans le composant

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

```
@Component({
  selector: 'first-cmp',
  template: `
    <div class="cmp">First Component</div>
    <style>.cmp { border: blue 2px solid;
  }</style>`
})
export class FirstComponent {
  constructor() { }
```

Encapsulation CSS dans le composant

```
@Component({
  selector: 'second-cmp', Advertisment
  template: `<div class="cmp">Second
    Component</div>`,
  styles: ['.cmp { border: green 2px solid;
    }']
})
export class SecondComponent {
  constructor() { }
}
```

by Jean Garutti -- Angular 4 -- -- 43

Encapsulation CSS dans le composant

- En regardant vos deux composants,
 - ☐ on voit deux façons différentes d'y appliquer des styles.


Encapsulation CSS dans le composant

- Dans **FirstComponent**, nous avons une étiquette de style dans le modèle externe
 - ☐ tandis que dans le **SecondComponent**, nous utilisons une propriété de style sur son décorateur de composants.

Les deux auront le même résultat lors du rendu sur l'écran.

Encapsulation **CSS** dans le composant

- Angular, par défaut, encapsule le composant CSS.
- Voici l'état actuel des 3 composants



```
graph TD
    App[App Component]
    First[First Component]
    Second[Second Component]
    App --- First
    App --- Second
```

by Jean Garutti

-- Angular 4 --

-- 46

Encapsulation **CSS** dans le composant

- Nous constatons que chaque composant correspondant à la classe CSS **.cmp** est scopé à son propre modèle.

by Jean Garutti

-- Angular 4 --

-- 47

Encapsulation **CSS** dans le composant

- Le comportement CSS par défaut des classes **.cmp** multiples aurait causé des collisions de noms globaux avec nos styles.

by Jean Garutti

-- Angular 4 --

-- 48

Encapsulation CSS dans le composant

- Regardons les outils de développeur Chrome et le HTML et le CSS rendus.

Encapsulation CSS dans le composant

```

demo-app
  <h3>CSS Encapsulation with Angular 2</h3>
  <div class="cmp">
    App Component
  </div>
  <first-cmp _ngghost-ngn-2>
    <div _ngcontent-ngn-2 class="cmp">First Component</div>
  </first-cmp>
  <second-cmp _ngghost-ngn-3>
    <div _ngcontent-ngn-3 class="cmp">Second Component</div>
  </second-cmp>
</demo-app>

```

```

ruler
element.style {
}

.cmp[_ngcontent-ngn-2] {
  border: 2px solid blue;
}

.cmp {
  padding: 6px;
  margin: 6px;
  border: 2px solid red;
}

div {
  display: block;
}

```

Encapsulation CSS dans le composant

- *Par défaut*, Angular génère des attributs pour étendre les **noms de classe CSS** à chaque composant.

Encapsulation CSS dans le composant

- Regardez tous les éléments hérités du CSS **.cmp**
- Le CSS de chaque composant est scopé et remplace la couleur de bordure de base.

Encapsulation CSS dans le composant

- Les attributs générés par Angular NE DOIVENT PAS être utilisés pour cibler des éléments avec CSS.
- Ces attributs (automatiquement générés) peuvent être modifiés.

Encapsulation CSS dans le composant

- Angular possède des options de rendu CSS supplémentaires.
- La première est de pouvoir utiliser l'encapsulation CSS native.
 - ☐ L'activation de cette fonctionnalité obligera les navigateurs à utiliser le **Shadow DOM**.

Encapsulation CSS dans le composant

- Pour les navigateurs qui comprennent le **Shadow DOM**,
 - cela crée un nouveau contexte de rendu pour un élément donné complètement isolé du reste du DOM.
- Il s'agit d'un véritable encapsulage CSS natif mais n'est pas activé par défaut dans Angular.

by Jean Garutti

-- Angular 4 --

-- 55

Encapsulation CSS dans le composant

- Voici le code et la sortie générée.
- `@Component({`
 1. `selector: 'second-cmp',`
 2. `template: '<div class="cmp">Second Component</div>',`
 3. `styles: ['.cmp { border: green 2px solid; }'],`
 4. `encapsulation: ViewEncapsulation.Native // Use the native Shadow DOM to encapsulate our CSS`
 5. `})`
 6. `export class SecondComponent {`
 7. `constructor() { }`
 8. `}`

by Jean Garutti

-- Angular 4 --

-- 56

Merci !

Thanks !

by Jean Garutti

-- Angular 4 --

-- 57
