Encapsulation des vues et des styles

Temps de lecture : 4 minutes

1) Le shadowDom

Le shadowDOM permet une encapsulation des éléments du DOM à l'intérieur du DOM.

Dans les documents Web il n'y a en principe qu'un seul DOM qui possède un arbre, c'est-à-dire une hiérarchie propre entre ses éléments.

Le shadowDOM permet de créer une séparation supplémentaire entre certains éléments HTML du DOM afin d'isoler le JavaScript et les styles qui leur sont applicables.

Quel est l'intérêt du shadowDOM ? Il permet de mieux séparer les styles et le JavaScript et de ne pas risquer d'impacter d'autres éléments de l'application. Cela est très avantageux pour les larges applications.

Le shadowDOM fait partie des composants Webs natifs, cependant l'adoption par les navigateurs est lente. Les frameworks, comme Angular, ont donc adopté leur propre implémentation du shadowDOM.

2) Angular et le shadowDOM

Angular n'utilise pas à proprement parlé de shadowDOM, mais il émule la séparation permise par le shadowDOM en ajoutant des attributs spécifiques aux éléments, comme nous l'avons vu dans la vidéo.

En effet, par défaut, Angular utilise l'émulation du shadowDOM, et dans les métadonnées des composants (dans `@Component`), une propriété est ajoutée par défaut : `encapsulation: ViewEncapsulation.Emulated`.

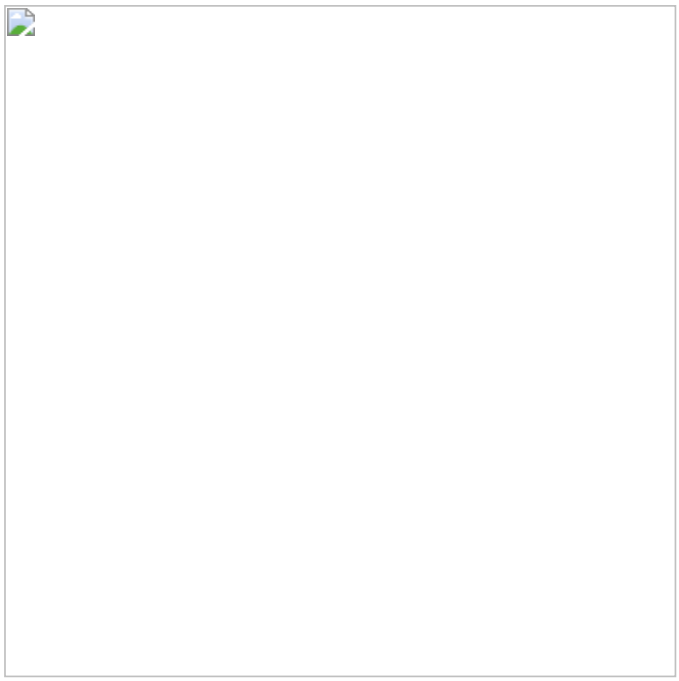
Si vous souhaitez enlever cette émulation il suffit d'ajouter la propriété `encapsulation: ViewEncapsulation.None` dans les métadonnées du composant.

Si vous souhaitez utiliser le shadowDOM (*mais attention, seules les versions récentes des navigateurs sont compatibles*), vous pouvez utiliser `ViewEncapsulation.ShadowDom`.

Essayons d'utiliser le shadowDOM sur notre composant dans notre exemple :

Nous avons importé `ViewEncapsulation` depuis `@angular/core` et avons ajouté la propriété `encapsulation: ViewEncapsulation.ShadowDom` aux métadonnées du composant.

Vous pouvez observer avec l'inspecteur de votre navigateur un `#shadow-root` qui a été ajouté :



Vous savez maintenant qu'Angular n'utilise pas le shadowDOM par défaut mais une encapsulation des styles et du JavaScript afin de permettre une meilleure séparation et donc une meilleure maintenabilité du code.