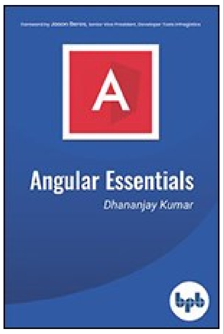


Angular Essentials: The Essential Guide to Learn Angular

Chapter 5: ViewEncapsulation in Angular

In this chapter, you will learn `viewEncapsualtion` in Angular. Following topics will be covered in this chapter:

- Shadow DOM
- None Mode
- Native Mode
- Emulated Mode

Shadow DOM

To understand `viewEncapsulation` in Angular, first we should understand about Shadow DOM. Putting it in simple words; Shadow DOM brings Encapsulation in HTML Elements. Using the Shadow DOM, markup, styles, and behaviors are scoped to the element and do not clash with other nodes of the DOM. Shadow DOM is part of Web Components, which encapsulates styles and login of element.

Angular Components are made up of three things:

- Component class
- Template
- Style

Combination of these three makes an Angular component reusable across application. Theoretically, when you create a component, in some way you create a web component (however, Angular Components are not web components) to take advantage of Shadow DOM. You can also use Angular with browsers, which does not support Shadow DOM because Angular has its own emulation and it can emulate Shadow DOM.

To emulate Shadow DOM and encapsulate styles, Angular provides four types of `ViewEncapsulation`. They are as follows:

- Emulated
- None
- ShadowDom
- Native (Deprecated in Angular 6.1)

These four types have different characteristics, as shown in the figure 5.1.

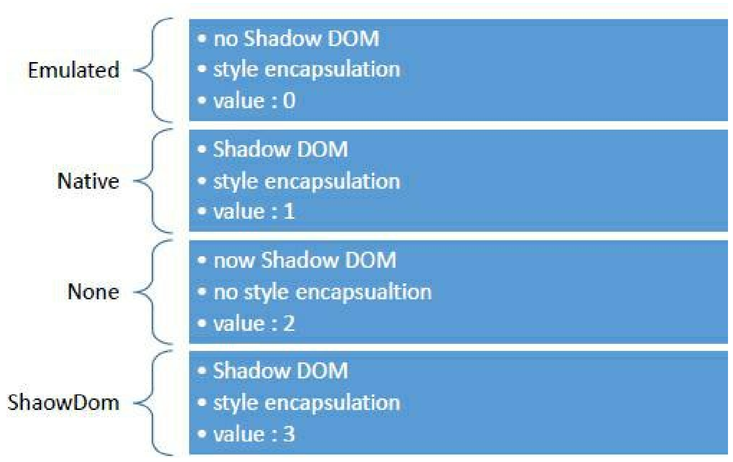


Figure 5.1

Let us try to understand it by using an example.

None Mode

To understand all **`viewEnacapsulation`** modes, let us create a component as shown in the *code listing 5.1*.

CodeListing 5.1

```
import { Component, ViewEncapsulation } from '@angular/
core';
@Component({
   selector:   'app-root',
   templateUrl:   './app.component.html',
   styleUrls:   ['./app.component.css'],
   encapsulation: ViewEncapsulation.None
})
export class AppComponent {
      title =  'parent component';
}
```

Template of **`AppComponent`** is created as shown in the *code listing 5.2.*

Code Listing 5.2

```
<div>
   <h1>
      Welcome to {{ title }}!
   </h1>
</div>
<app-child></app-child>
```

Since **`viewEnacapsualtion`** deals with styling, let us create style for **`AppComponent.`** We have put some style for hl element in CSS file of **`AppComponent`** as shown in *code listing 5.3.*

Code Listing 5.3

```
h1 {
   background:  red;
   color: white;
   text-transform: uppercase;
   text-align:  center;
}
```

As you noticed **`AppChild`** component is being used in the **`AppComponent. AppChild`** component is created as shown in the *code listing 5.4.*

Code Listing 5.4

```
import  {  Component  }  from '@angular/core';
@Component({
      selector:   'app-child',
      template:  '
         <h1>{{title}}</h1>
   '
})
export class AppChildComponent  {
   title =  'child app';
}
```

`ViewEncapsulation` deals with encapsulation of styling and creation of Shadow DOM and to see different options available with **`ViewEncapsulation,`** we are using same element hl in **`AppChild`** component also.

In **`ViewEncapsulation.None`** option,

- There is no shadow DOM
- Style is not scoped to component

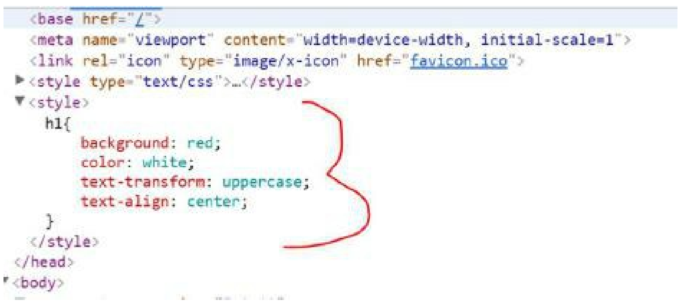
As you run the application, you will find hi style has applied to both components, even though we set style only in

AppComponent. It happened because in **AppComponent** we have set encapsulation property to **ViewEncapsulation.None.**

Code Listing 5.5

```
import { Component, ViewEncapsulation } from '@angular/
core';
@Component({
      selector:   'app-root',
      templateUrl:   './app.component.html',
      styleUrls:   ['./app.component.css'],
      encapsulation: ViewEncapsulation.None
})
export class AppComponent {
      title =  'parent component';
}
```

In the browser when you examine source code, you will find hi style has been declared in the head section of the DOM as shown in the *figure 5.2*.



```
  <base href="/">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
 ▶<style type="text/css">…</style>
 ▼<style>
    h1{
        background: red;
        color: white;
        text-transform: uppercase;
        text-align: center;
    }
  </style>
</head>
<body>
```

Figure 5.2

Therefore, in **ViewEncapsulation.None,** style gets moved to the DOM head section and is not scoped to the component. There is no Shadow DOM for the component and component style can affect all nodes of the DOM.

ShadowDom Mode

Characteristics of Native and **ShadowDom** mode are almost same. However, starting Angular 6.1, Native mode has been deprecated. In **ViewEncapsulation.ShadowDom** option:

- Angular will create Shadow DOM for the component
- Style is scoped to component

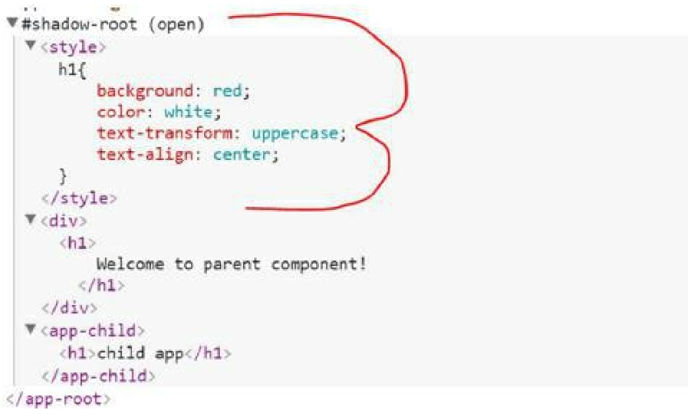
As you run the application, you will find hl style has applied to both components, even though we set style only in **AppComponent.** It happened because in **AppComponent,** we have set encapsulation property to **ViewEncapsulation.ShadowDom,** and we are using **AppChildComponnet** as child inside template of **AppComponent.** You can set encapsulation to **ShadowDom** as shown in *code listing 5.6.*

Code Listing 5.6

```
import { Component, ViewEncapsulation } from '@angular/
core';
@Component({
      selector:   'app-root',
      templateUrl:   './app.component.html',
      styleUrls:   ['./app.component.css'],
      encapsulation: ViewEncapsulation.ShadowDom
})
export class AppComponent {
      title =  'parent component';
}
```

In the browser, when you examine source code, you will Shadow DOM has created for the **AppComponent** and style is scoped to that as shown in *figure 5.3*.

Therefore, in `ViewEncapsulation.ShadowDom` Angular creates a Shadow DOM and style is scoped to that Shadow DOM.



```
▼#shadow-root (open)
  ▼<style>
     h1{
         background: red;
         color: white;
         text-transform: uppercase;
         text-align: center;
     }
   </style>
  ▼<div>
     <h1>
         Welcome to parent component!
       </h1>
   </div>
  ▼<app-child>
     <h1>child app</h1>
   </app-child>
</app-root>
```

Figure 5.3

EmulatedMode

In Angular default mode is emulated mode. In `ViewEncapsulation. Emulated,` in this option:

- Angular will not create Shadow DOM for the component
- Style will be scoped to the component
- This is default value for encapsulation

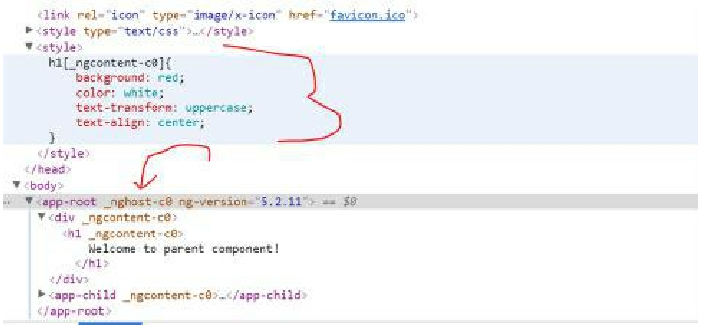
You can enable emulated mode as shown in *code listing 5.7.*

CodeListing 5.7

```
import { Component, ViewEncapsulation } from '@angular/
core';
@Component({
      selector:   'app-root',
      templateUrl:   './app.component.html',
      styleUrls:   ['./app.component.css'],
      encapsulation: ViewEncapsulation.Emulated
})
export class AppComponent {
      title =  'parent component';
}
```

As you ran the application, you will find that hl style from `AppComponent` is not applied to hl of `AppChildComponent.` It is due to emulated scoping. In this, style is scoped only to the component. In this option, Angular only emulates to Shadow DOM and does not create a real shadow DOM. Hence, the application that runs in browsers does not support Shadow DOM also and styles are scoped to the component as well.

Let us see how Angular achieves this? In the browser, when you examine source code, you will find answer, consider *figure 5.4*.



```
    <link rel="icon" type="image/x-icon" href="favicon.ico">
  ►<style type="text/css">…</style>
  ▼<style>
     h1[_ngcontent-c0]{
         background: red;
         color: white;
         text-transform: uppercase;
         text-align: center;
     }
   </style>
 </head>
▼<body>
… ▼<app-root _nghost-c0 ng-version="5.2.11"> == $0
   ▼<div _ngcontent-c0>
      <h1 _ngcontent-c0>
         Welcome to parent component!
       </h1>
    </div>
   ►<app-child _ngcontent-c0>…</app-child>
   </app-root>
```

Figure 5.4

Angular has created style in the head section of the DOM and given an arbitrary id to the component. On basis of ID, selector style is scoped to the component.

Summary

It is very common misconception that Angular always creates Shadow DOM for components, however after reading this chapter, you know that it depends on `ViewEncapsulation` mode. In this chapter, you learnt about:

- Shadow DOM
- None Mode
- Native Mode
- Emulated Mode