

# Angular View Encapsulation Strategies

## 2. Shadow DOM Encapsulation (ViewEncapsulation.ShadowDom)

- Uses native Shadow DOM APIs for style encapsulation.
- Styles are scoped within a shadow root (`#shadow-root`).
- Host and content styles do not leak in or out.

Pros: True isolation, aligns with Web Components.

Cons: Limited browser support, increased DOM complexity.

## 3. None (ViewEncapsulation.None)

- No encapsulation; styles are global.
- All component styles are added to the global stylesheet.

Pros: Simple and performant.

Cons: High risk of style conflicts across components.

## Summary Table

Encapsulation | Isolation | Clash Risk | Shadow DOM | Use Case

-----|-----|-----|-----|-----

Emulated | Scoped | Low | No | Default, All browsers

Shadow DOM | True | None | Yes | Web Components, Libs

None | None | High | No | Global themes, Small apps

## Default Encapsulation Behavior

- Angular uses ViewEncapsulation.Emulated by default.
- No need to explicitly specify it unless you want clarity or override.
- The default provides scoped styles via rewritten selectors and is compatible across all browsers.

## How View Encapsulation Works Under the Hood

- Angular rewrites CSS selectors at compile-time using `_ngghost` and `_ngcontent` attributes.
- `:host` becomes `[_ngghost-xyz]`, and element selectors become `[ngcontent-xyz]`.
- In Shadow DOM mode, Angular uses `attachShadow()` and places template/styles in `ShadowRoot`.
- In None mode, styles are added globally without scoping.

Interactions:

## Angular View Encapsulation Strategies

- Emulated: Scoped styles using rewritten selectors.
- Shadow DOM: Styles truly encapsulated using browser APIs.
- None: Styles leak globally.

## Bypassing Angular View Encapsulation with `::ng-deep`

`::ng-deep` is an Angular-specific selector used to force styles to penetrate view encapsulation.

- It allows a component's style to affect projected content or child components.
- Useful for styling Angular Material components, or content inside `<ng-content>`.
- Angular rewrites the selectors before view encapsulation scoping, so the CSS rules bypass `_ngcontent-*` attributes.
- Deprecated but still supported as of Angular 17+.

Syntax:

```
::ng-deep .some-class { color: red; }
```

Internally:

- Angular compiles styles without applying the usual attribute isolation (`_ngcontent`).
- The entire selector after `::ng-deep` is treated as global.

```
<app-card>
```

```
  This is styled  
</app-card>
```

### Child Component

```
::ng-deep  
  color: red  
}
```

### Resulting DOM and Styles

```
<app-card _ngghost-abc-c0>  
  <h3 _ngcontent-abc-c0>  
    This is styled  
  </h3>  
</app-card>
```

```
{  
  color: red;  
}
```

### Bypassing View Encapsulation with