

How Styles Work in Angular



Brian Treece

CHIEF OF USER EXPERIENCE AT SOCREATE.IT

@brianmtreece





HTML
JavaScript
CSS





—





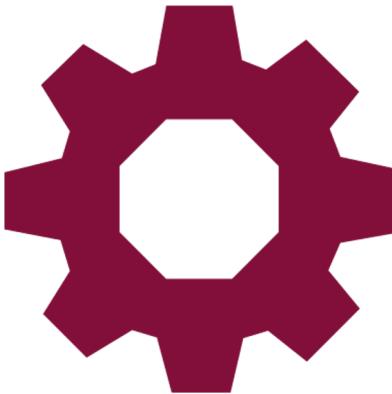
Encapsulation & Isolation
Bundle for Reuse
Use in Multiple Applications



Web Components



Custom
Elements



HTML
Templates



Shadow
DOM



```
<my-custom-element>
```

Put content and markup here

```
</my-custom-element>
```



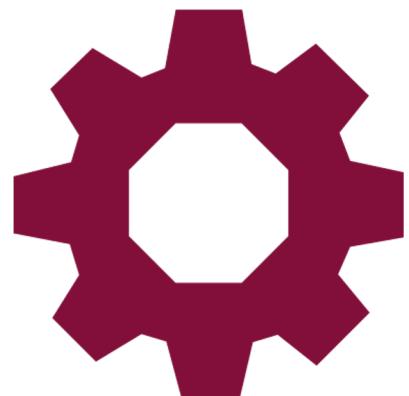
Custom Elements



```
<template id="template">
```

Put content and markup here

```
</template>
```



HTML Templates



```
<html>  
  <my-custom-element>  
    #shadow-root (user-agent)  
      Content and markup ends up here  
  </my-custom-element>
```



Shadow DOM



Want More on Web Components?

A Practical Guide to Vanilla Web Components



Leon Revill
WEB ARCHITECT

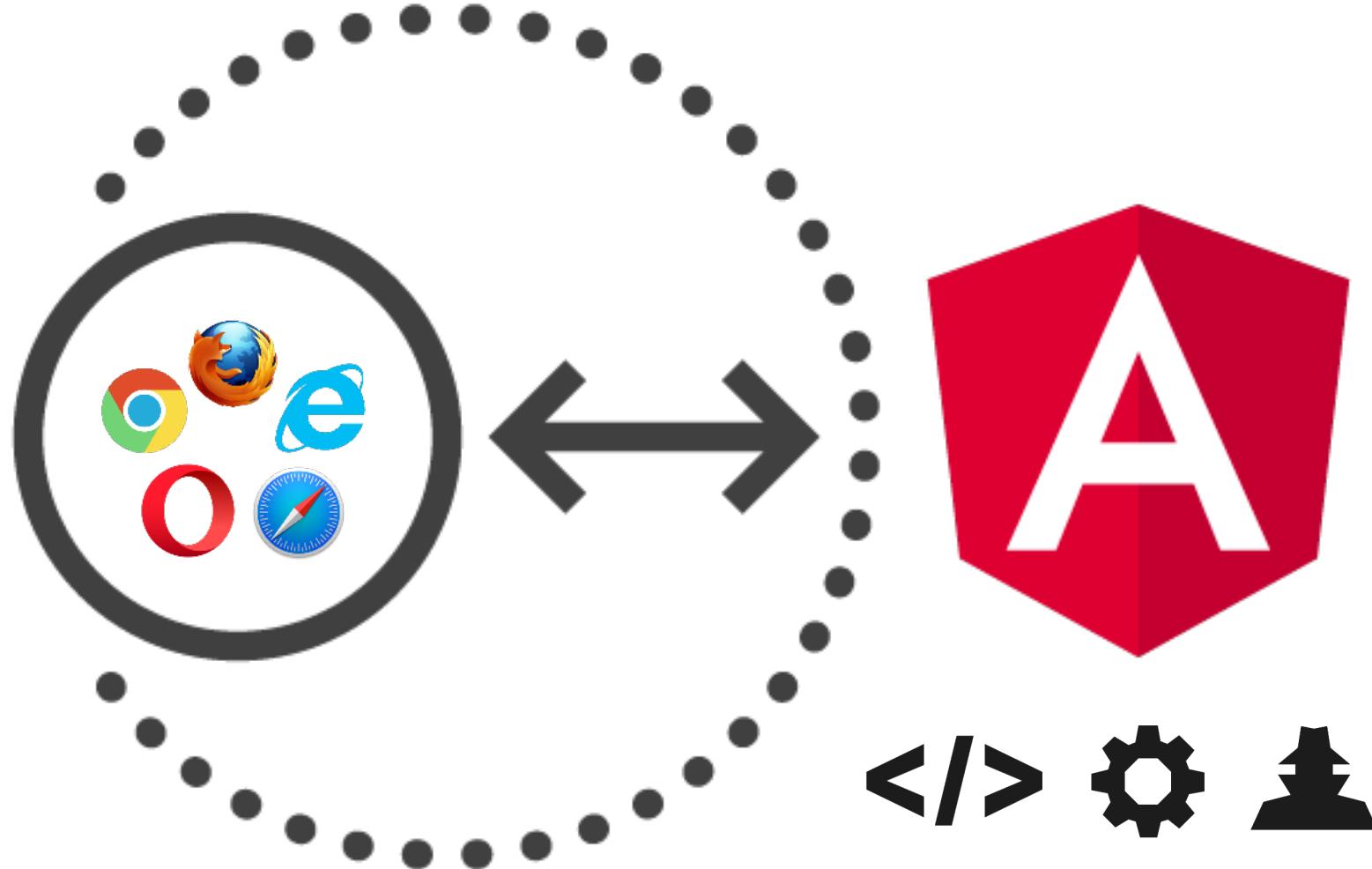
@revillweb www.revillweb.com



<https://app.pluralsight.com/library/courses/vanilla-web-components-practical-guide/table-of-contents>



Angular Fills Some Gaps



Coming Up...



Basic Component Structure

View Encapsulation

Adding Styles

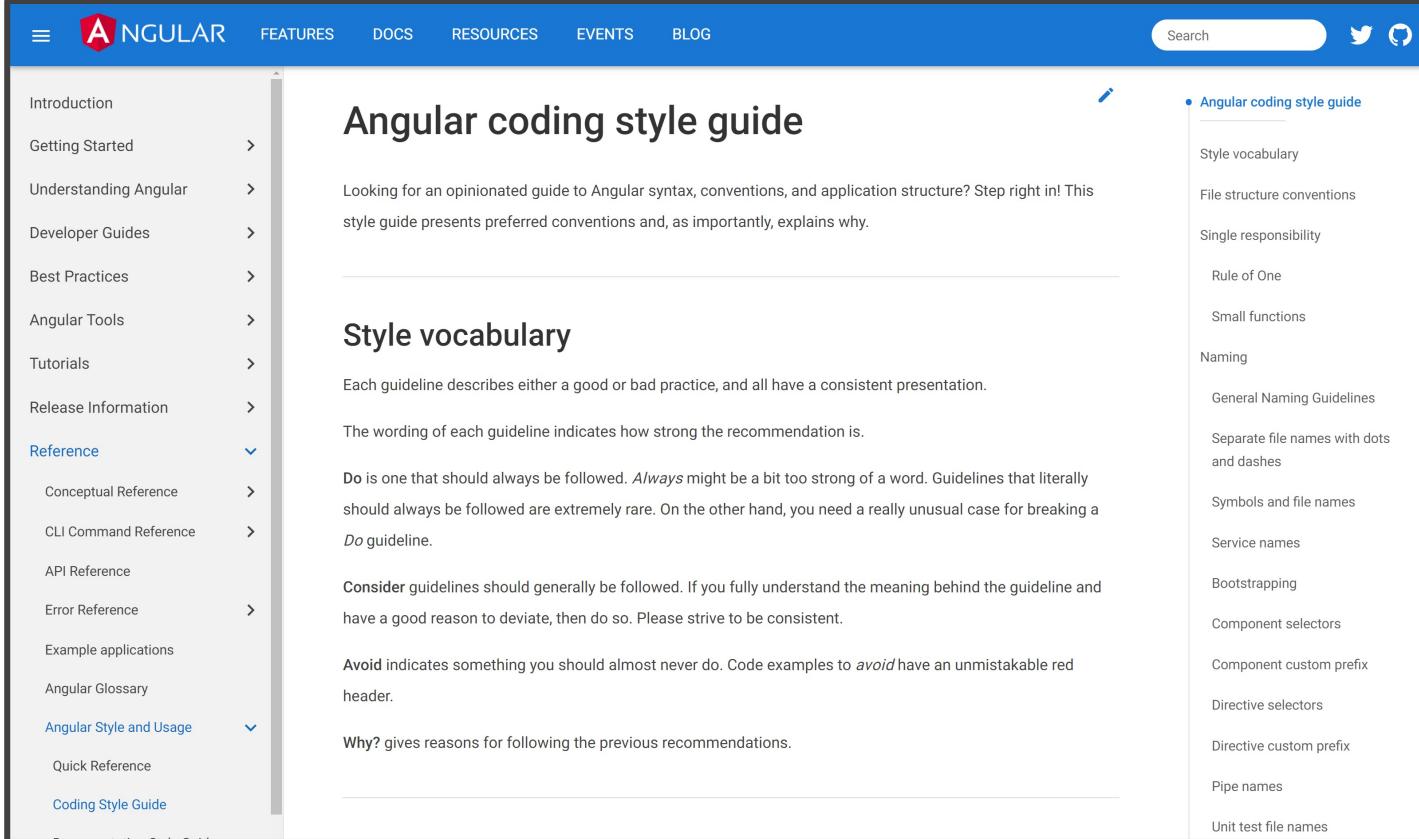
Scoping CSS Selector Emulation



Angular Component Structure



Angular Style Guide



The screenshot shows the Angular Style Guide page. The header includes the Angular logo, navigation links for FEATURES, DOCS, RESOURCES, EVENTS, and BLOG, and social media icons for Twitter and GitHub. A search bar is also present. The main content area has a sidebar with links to Introduction, Getting Started, Understanding Angular, Developer Guides, Best Practices, Angular Tools, Tutorials, Release Information, Reference (with sub-links for Conceptual Reference, CLI Command Reference, API Reference, Error Reference, Example applications, and Angular Glossary), and Angular Style and Usage (with sub-links for Quick Reference and Coding Style Guide). The main content starts with a section titled "Angular coding style guide" which describes the purpose of the guide. Below it is a "Style vocabulary" section with detailed explanations of "Do", "Consider", "Avoid", and "Why?" terms. To the right, a sidebar lists various style conventions: Style vocabulary, File structure conventions, Single responsibility, Rule of One, Small functions, Naming, General Naming Guidelines, Separate file names with dots and dashes, Symbols and file names, Service names, Bootstrapping, Component selectors, Component custom prefix, Directive selectors, Directive custom prefix, Pipe names, and Unit test file names.

Angular coding style guide

Looking for an opinionated guide to Angular syntax, conventions, and application structure? Step right in! This style guide presents preferred conventions and, as importantly, explains why.

Style vocabulary

Each guideline describes either a good or bad practice, and all have a consistent presentation.

The wording of each guideline indicates how strong the recommendation is.

Do is one that should always be followed. *Always* might be a bit too strong of a word. Guidelines that literally should always be followed are extremely rare. On the other hand, you need a really unusual case for breaking a *Do* guideline.

Consider guidelines should generally be followed. If you fully understand the meaning behind the guideline and have a good reason to deviate, then do so. Please strive to be consistent.

Avoid indicates something you should almost never do. Code examples to *avoid* have an unmistakable red header.

Why? gives reasons for following the previous recommendations.

- Angular coding style guide
- Style vocabulary
- File structure conventions
- Single responsibility
- Rule of One
- Small functions
- Naming
 - General Naming Guidelines
 - Separate file names with dots and dashes
 - Symbols and file names
 - Service names
 - Bootstrapping
 - Component selectors
 - Component custom prefix
 - Directive selectors
 - Directive custom prefix
 - Pipe names
 - Unit test file names

<https://angular.io/guide/styleguide>



View Encapsulation



View Encapsulation Modes

None

Emulated

ShadowDom



In a Traditional Web Application...

```
<head>
  <link rel="stylesheet" href="/path-to/stylesheet.css">
</head>
```





Don't Add Styles Tied to
These Attributes



View Encapsulation Modes: None & ShadowDom





This Feels
Wrong



ShadowDom



Native Shadow DOM





Emulated Mode



Component Styles



Other Ways to Add Styles

Embedded

<link>

@import

Inline

styleUrls



Styles In the Template



Linking From an External File



CSS Imports



Inline Styles



styleUrls



```
<link rel="stylesheet" href="/path/to/styleSheet.css">
```

```
styleUrls:['stylesheet.css']
```





Syntax Highlighting &
Code Completion





Syntax Highlighting &
Code Completion



The Same but Different

styleUrls

```
styleUrls: ['example.component.scss']
```

link

```
<link rel="stylesheet" href="/path/stylesheet.css">
```



templateUrl



CSS Preprocessors

Sass ✓ {less}

stylus



Emulated CSS Selectors



CSS Scoping

`:host`
`:host-context`
`::ng-deep`



“Host”

```
<saa-app-nav>  
  <ul> . . . </ul>  
</saa-app-nav>
```



“Shadow Tree”

```
<saa-app-nav>  
  <ul> . . . </ul>  
</saa-app-nav>
```



:host

```
:host {  
background: #2A9FBC;  
border-radius: 0.5em;  
margin: 1.5em 0;  
padding: 0;  
}
```



:host-context

```
<div class="container">  
  <saa-app-nav>  
    <ul>...</ul>  
  </saa-app-nav>  
</div>
```



:host-context

```
:host-context(.container) {  
    background: #2A9FBC;  
    border-radius: 0.5em;  
    margin: 1.5em 0;  
    padding: 0;  
}
```





Why Use :host?



Why Not :host-context?





We Don't Always Know!

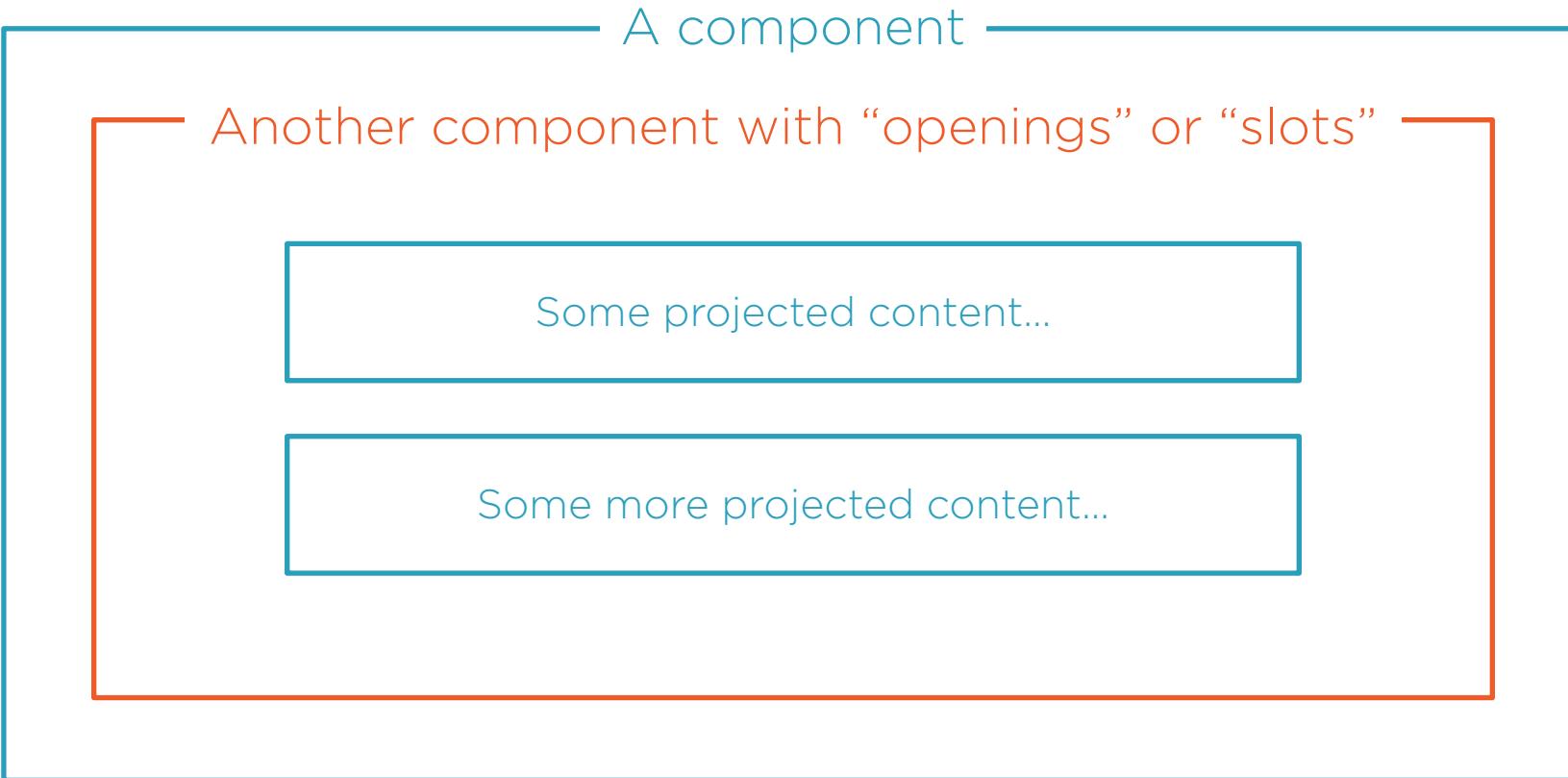
:host-context
Use it Purposefully!



::ng-deep



Content Projection



Content Projection

```
<div>
```

This is where content goes

```
</div>
```



Content Projection

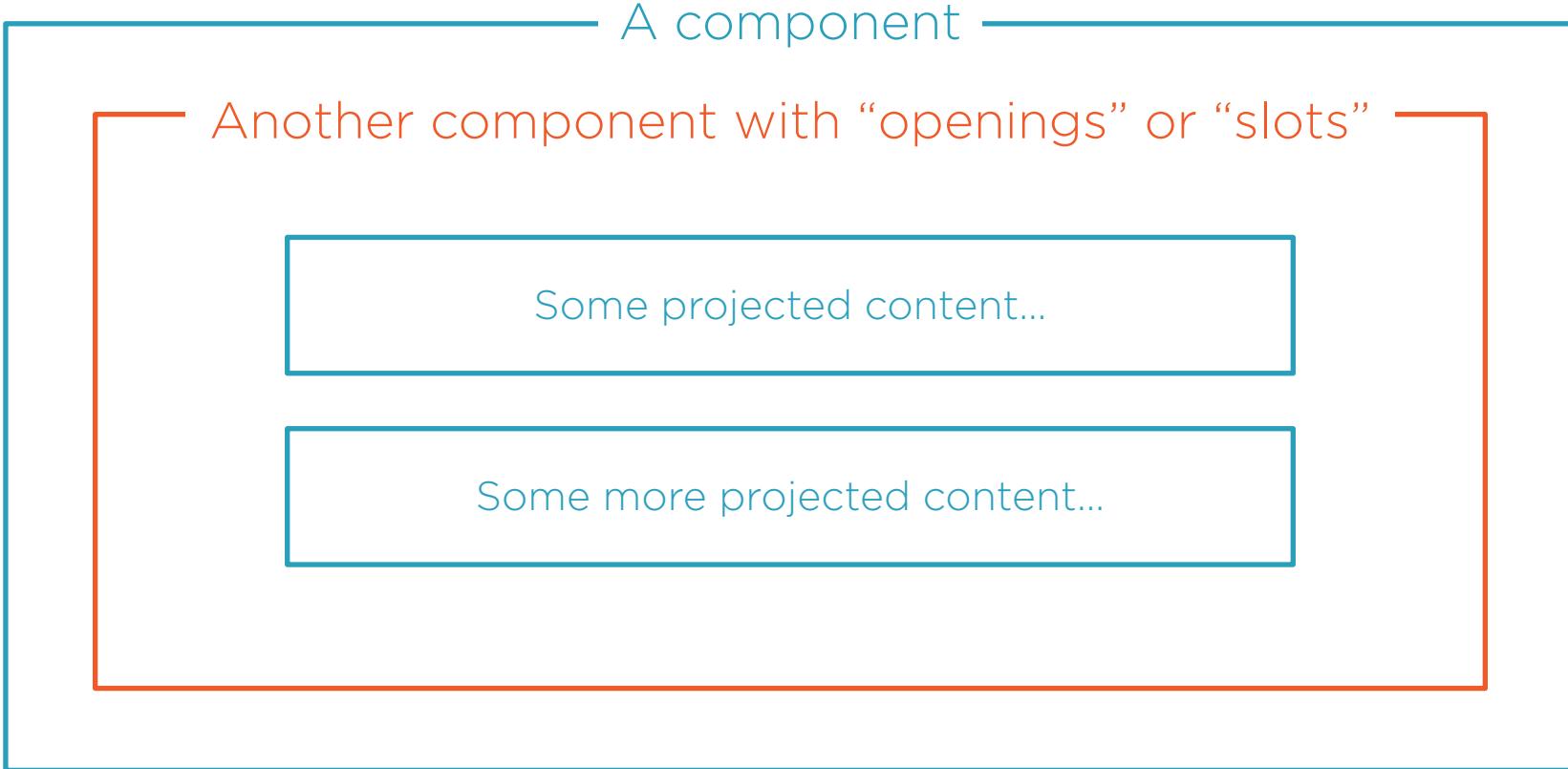
```
<my-component>
```

This is also where content goes

```
</my-component>
```



Content Belongs to Parent



Pop Up Dialog



We Don't Know



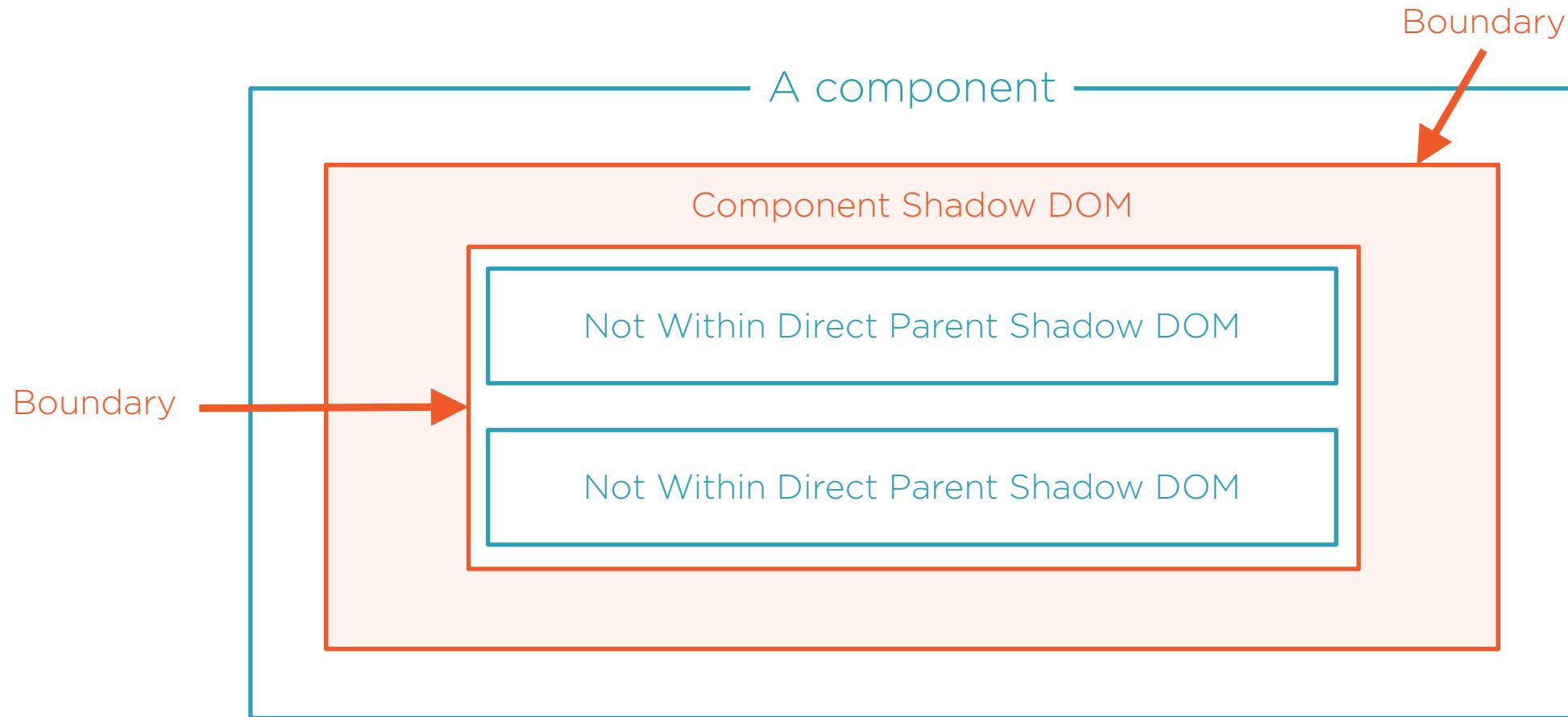
Pop Up Dialog



We Do Know



Shadow Boundary

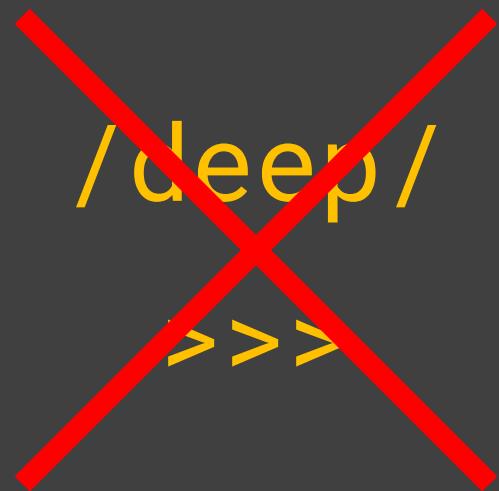




Yikes!



CSS Scoping Module Spec



Deprecated



Where We've Been...



**Styling Angular
Components**



**Shadow DOM &
CSS Scoping Emulation**



Where We're Headed...

**Wrap Up How
Styles Work**

**Create a Scalable &
Maintainable System**



Summary



Style Encapsulation

**Angular Shadow DOM/CSS Scoping
Module Emulation**

Custom Elements

Component HTML Templates

**View Encapsulation: Emulated,
ShadowDom, & None**

**Adding Styles: Styles Property,
Embedded, Linked, Imported, Inline,
External**

Using Preprocessors

:host, :host-context, ::ng-deep



We Need to Rethink...

How we craft & organize styles

How we style for predictability, scale, &
maintainability

