Styling Web Components

Cory House
bitnative.com
Twitter: @housecor

pluralsight
hardcore developer training

Agenda

Shadow DOM encapsulates styles

New CSS concepts

:host

:host-context

::content

::shadow

/deep/

:unresolved

Styling the Shadow Host

Style the Shadow host using the :host pseudo selector

```
<div id="my-host">
   My component is hosted in here...
</div>
```

```
#my-host:host { /* Styles the shadow host */
   border: solid 1px red;
}
```

:host Example

Demo

Theming the Shadow Host

Style :host classes to theme components

```
:host(.awesome) {
   /* awesome styles here */
}

:host(.lame) {
   /* totally lame styles here */
}
```

```
<my-component class="awesome"></my-component>
```

Theming the Shadow Host

Demo

:host Has Low Specificity

More specific selectors in the Light DOM may override

CSS Specificity Rules

Traditional CSS specificity rules apply for :host

From most to least specific:

1. Inline Style
2. ID
3. Class, pseudo-class, attribute
4. Elements

```
<p style="color: red;">
#my-element { color: red; }
.my-class { color: red; }
p { color: red; }
```

:host pseudo-selector specificity

Demo

Styling Shadow Host States

Can style states too:

```
:host(:hover)

:host(:active)

:host(:visited)

:host(:link)
```

Styling :host states

Demo

Styling the Host Context

Style the Shadow DOM based on context

```
:host-context
```

Styling the Shadow DOM via :host-context

Demo

Styling Distributed Nodes

Style nodes that pass through an insertion point with ::content selector

```
::content p {
   color: red;
}
```

```
::content > button {
   border: 1px solid red;
}
```

Styling Distributed Nodes via ::content

Demo

Styling Shadow DOM From the Light DOM

What?! You liar!

I thought the Shadow DOM was protected from styling from the outside!

Styling Shadow DOM From the Light DOM

```
::shadow


#host::shadow li {
    color: red;
}
```

Style Shadow DOM from the Light DOM via ::shadow

Demo

Styling Shadow DOM From the Light DOM

Style *all* shadows:

/deep/

Style Shadow DOM from the Light DOM via /deep/

Demo

Avoid Flash of Unstyled Content (FOUC)

All custom element tags (for example, <x-button>) are plain old HTML elements until the browser finishes calling the createdCallback method.

To avoid a flash of unstyled content, use :unresolved

Avoid Flash of Unstyled Content (FOUC)

:unresolved matches unresolved elements. Once createdCallback is called, the element is resolved.

```
my-component {
   opacity: 1;
}

my-component:unresolved {
   opacity: 0;
}
```

:unresolved Only Applies to Custom Elements

Doesn't apply to elements that inherit from HTMLUnknownElement

Only elements with a dash (–) in their name are affected by :unresolved

<sillybutton> <- not styled by :unresolved since HTMLUnknownElement
<silly-button> <-styled by :unresolved since HTMLElement (due to dash)

Avoiding Flash of Unstyled Content with :unresolved

Demo

Dynamic Styling

Just like Light DOM elements:

```
var myComponent = document.createElement('my-component');
myComponent.style.background = 'red';
```

Summary

Shadow DOM encapsulates styles

New CSS concepts

:host – Style the shadow host from the shadow DOM

:host-context - Style host based on context

::content – Style <content> insertion points

::shadow – Style Shadow DOM from outside (1st level)

/deep/ - Style Shadow DOM from outside (Infinite depth)

:unresolved – Avoid FOUC