Web Components and Modular CSS

@AndrewRota | CSS Dev Conf 2014

Modularity

UI Libraries




Topcoat

dōjō

Sencha

yui

CSS Features

~~Encapsulation~~

~~Scope~~

~~Interfaces~~

~~Modularity~~

Modular CSS Patterns

BEM

SMACSS

Atomic CSS

OOCSS

BEM

```
/* Block */
.nav { }

/* Element */
.nav__item { }

/* Block with Modifier */
.nav--hidden { }

/* Element with Modifier */
.nav__item--current { }
```

SMACSS

```
/* Module */
.nav { }

/* Module with State */
.nav.is-current { }

/* Module with Semantic Element Selector */
.nav > h2 { }

/* Layout Style */
.l-inline { }
```

Web Components

> Web Components usher in a **new era** of web development based on **encapsulated and interoperable** custom elements that extend HTML itself. - Polymer

Web Components APIs

Custom Elements

HTML Templates

HTML Imports

Shadow DOM

Custom Elements

```
<my-element>Hello World.</my-element>
```

```
var MyElement = document.registerElement('my-element', {
  prototype: Object.create(HTMLElement.prototype)
});
```

HTML Templates

```html
<template id="my-template">
    <p>Hello World.</p>
    <!-- This image won't be downloaded on page load -->
    <img src="example.jpg" alt="Example">
</template>
```

```javascript
document.importNode(
    document.getElementById('my-template').content,
    true
);
```

HTML Imports

```
<link rel="import" href="/imports/my-component.html">
```

Preserve log ☑ Disable cache

Filter | All | Documents | Stylesheets | Images | Media | Scripts | XHR | Fonts | TextTracks | WebSockets | Other | Hide data UR

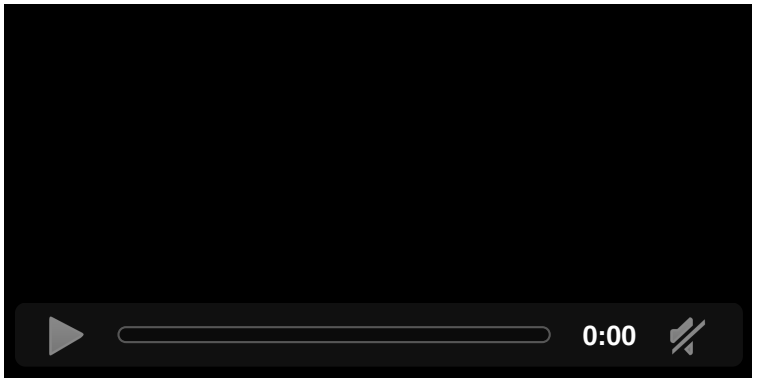
Name	Method	Status	Type	Initiator	Size	Time	Timeline 2.00 s
index.html	GET	200	text/html	Other	13.7 KB	393 ms	
slide-content.html	GET	200	text/html	index.h...	490 B	825 ms	
slide-show.html	GET	200	text/html	index.h...	956 B	958 ms	
code-prism.html	GET	200	text/html	index.h...	1.6 KB	1.02 s	
hello-world.html	GET	200	text/html	index.h...	1.1 KB	967 ms	

Shadow DOM

```
// Create Shadow Root
document.getElementById('my-element').createShadowRoot();
// Access Shadow Root
document.getElementById('my-element').shadowRoot;
```

User Agent Shadow DOM

```
<video src="#" controls></video>
```



User Agent Shadow DOM

```
<input type="date">
```

mm/dd/yyyy

Shadow DOM

Shadow DOM.

Light DOM.

```
<div id="my-first-element"></div><p>Light DOM.</p>
```

```
// Create Shadow Root
var s = document.getElementById('my-first-element').createS
// Add Styles and Text
s.innerHTML += '<style>p { color: crimson; margin: 5px 0 5p
s.innerHTML += '<p>Shadow DOM.</p>';
```

Content Insertion Points

```
<div id="my-second-element">
    <content></content>
</div>
```

Shadow DOM and `<content>`

Shadow DOM Start.

Hello!

Shadow DOM End.

```
<div id="my-second-element"><p>Hello!</p></div>
```

```
var s = document.getElementById('my-second-element').create
s.innerHTML += '<p>Shadow DOM Start.</p>';
s.innerHTML += '<style>p { color: crimson; margin: 5px 0; }
s.innerHTML += '<content></content>';
s.innerHTML += '<p>Shadow DOM End.</p>';
```

Into the Light

```
/* pseudo-class for host element*/
:host { }
/* functional pseudo-class, for host if it matches the sele
:host() { }
/* functional pseudo-class, for host context that matches s
:host-context() { }
/* pseudo-element, for distributed notes rendered via a <co
::content { }
```

Into the Dark

```
/* pseudo-element for shadow roots */
::shadow { }
```

```
/* combinator for selecting through shadow boundaries */
body /deep/ p { }
```

> [`/deep/`] is basically a **super-descendant** combinator.
>
> - CSS Scoping Module Draft, Issue 6

Let's Write a Component

Hello world, I am a **web component**.

```
<link rel="import" href="../assets/hello-world.html">
```

```
<hello-world>I am a <strong>web component</strong></hello-w
```

Let's Write a Component

Hello world, I am a **web component**.

```
<template id="hw">
  <style>
    ::content strong { color: crimson; }
    p { margin: 2px 20px 2px 0; }
    :host { border: 1px solid FireBrick; display: block; ma
    .hello { color: #91D4D; }
  </style>
  <p><span class="hello">Hello world</span>, <content></con
</template>
```

Let's Write a Component

Hello world, I am a **web component**.

```
var importedDoc = document.currentScript.ownerDocument;
var elementPrototype = Object.create(HTMLElement.prototype)
elementPrototype.createdCallback = function() {
  var template = importedDoc.getElementById('hw').content;
  var clone = document.importNode(template, true);
  this.createShadowRoot().appendChild(clone);
};
document.registerElement('hello-world', {prototype: element
```

Can I Use???

	Custom Elements	HTML Templates	HTML Imports	Shadow DOM
	✓	✓	✓	✓
	✓	✓	✓	✓
	Flag	✓	Flag	Flag
	X	✓	X	X
	X	X	X	X

Polyfills

When To Use Web Components?

Third Party Widgets?

Third Party UI Libraries?

Internal UI Libraries?

Web Component All the Things!?

Third Party Widgets

```
<google-map
    latitude="29.954356"
    longitude="-90.067863">
</google-map>
```

Third Party UI Libraries

CSS HTML JS

```
<paper-radio-group selected="css">
    <paper-radio-button name="css" label="CSS"></paper-r
</paper-radio-group>
```

```
<paper-slider value="10"></paper-slider>
```

Internal UI Libraries

```
<acme-corp--menu>
    <acme-corp--menu-item>Home</acme-corp--menu-item>
    <acme-corp--menu-item selected>About</acme-corp--menu-i
    <acme-corp--menu-item>Contact Us</acme-corp--menu-item>
</acme-corp--menu>

<acme-corp--login-form
    ajax
    url="login.php">
</acme-corp--login-form>
```

Web Component Everything??

```
<acme-corp--app>
    <acme-corp--menu></acme-corp--menu>
    <acme-corp--content></acme-corp--content>
    <acme-corp--footer></acme-corp--footer>
</acme-corp--app>
```

Probably Not (and that's OK)

> I don't ever see us going all in on Custom Elements for every possible thing ... **Use native elements and controls when possible and supplement with custom elements.**
>
> - Joshua Peek, Github Programmer

Best Practices

Small

Open for Extension

Documented

Unit Tested

Accessible

Responsive

Tooling

Frameworks

Towards a Component Driven Web

Thanks!

Resources

- WebComponents.org
- Web Components: A Tectonic Shift for Web Development by Eric Bidelman
- *Web Components* by Jarrod Overson and Jason Strimpel
- Ten Principles for Great General Purpose Web Components

Colophon

This presentation was built with Shadow DOM, HTML Templates, HTML Imports, and the Custom Elements `<slide-show>` and `<slide-content>` using Web Component Slides.