The Basic Anatomy of a Web Component

Leon Revill
WEB ARCHITECT

@revillweb www.revillweb.com

Overview

The two principle methods for writing a Web Component

- The Pros & Cons of each

Best practices for Custom Elements

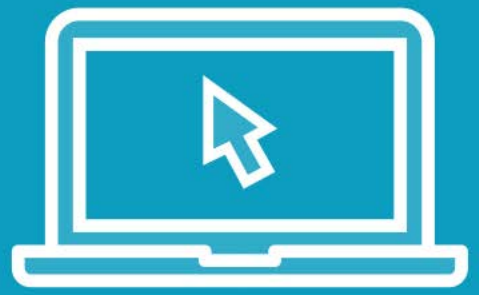
How to handle Web Component attributes

Dealing with Web Component properties

Utilizing the Shadow DOM with a Web Component

The Two Different Methods for Writing a Web Component

Demo



Method 1: Pure JavaScript

How to write a Web Component skeleton using the Pure JavaScript method

How to consume a Web Component written using the Pure JavaScript method

Method 1: Pure JavaScript

Advantages

- Familiar concept
- Consumed just like any other JavaScript file
- Can be easily transpiled to support older browsers
- Template strings can be used to make writing HTML within JavaScript easier

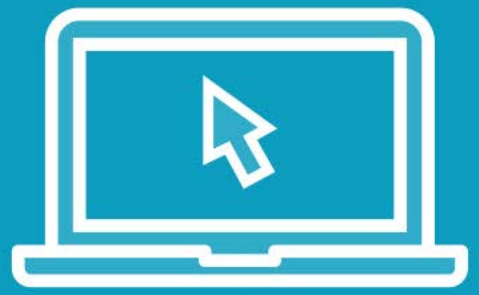
Method 1: Pure JavaScript

Disadvantages

Writing HTML & CSS within JavaScript can feel unnatural and add bloat to your code

Many text editors and IDEs lack full template string support

Demo



Method 2: HTML Import

How to write a Web Component skeleton using the HTML Import method

How to consume a Web Component written using the HTML Import method

webcomponents.js Polyfills

https://goo.gl/aWVaVb

README.md

webcomponents.js

build failing

A suite of polyfills supporting the Web Components specs:

Custom Elements: allows authors to define their own custom tags (spec).

HTML Imports: a way to include and reuse HTML documents via other HTML documents (spec).

Shadow DOM: provides encapsulation by hiding DOM subtrees under shadow roots (spec).

This also folds in polyfills for `MutationObserver` and `WeakMap`.

Releases

Pre-built (concatenated & minified) versions of the polyfills are maintained in the tagged versions of this repo. There are two variants:

`webcomponents.js` includes all of the polyfills.

`webcomponents-lite.js` includes all polyfills except for shadow DOM.

Method 2: HTML Import

Advantages

Automatic de-duplication of imports

Writing HTML & CSS is more natural and IDE support isn't an issue

Method 2: HTML Import

Disadvantages

Currently the HTML Import spec is contentious and not well supported

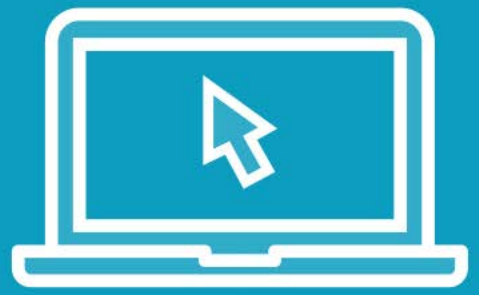
Even when polyfilled there are caveats which create bad code smells

Hard to transpile JavaScript within HTML files with currently tools

Method 1: Pure JavaScript

Custom Element Best Practices

Demo

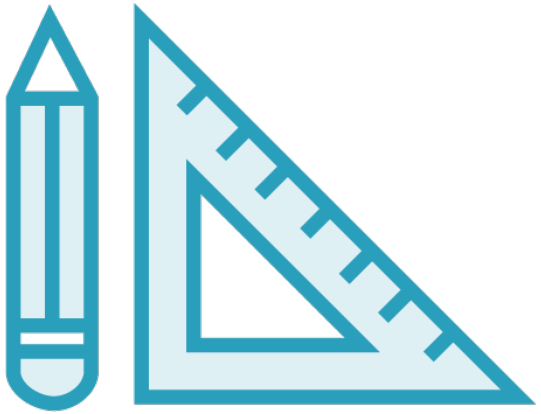


Class and element naming conventions

Implementing custom methods

Defining local variables

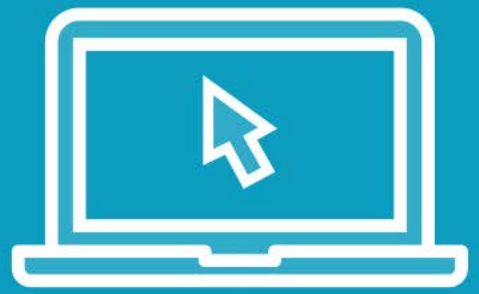
Updating the component template



Guidelines Only

Web Component Attributes

Demo

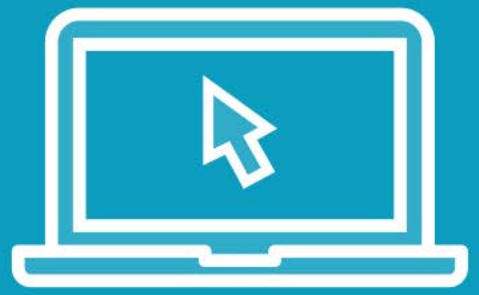


Subscribing to attribute changes

Advertising state through attributes

Web Component Properties

Demo



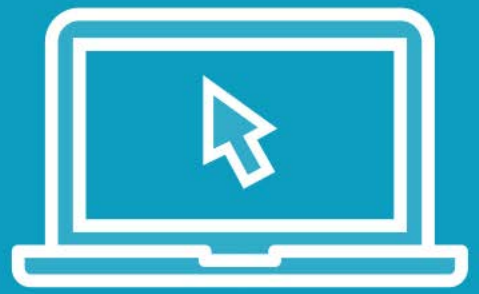
Difference between properties and attributes

Using setter methods to retrieve data

Using getter methods to provide data

Web Component Shadow DOM

Demo



Creating a shadow root for a Web Component

Adding and updating the template within the Shadow DOM

Summary

- Introduced the two methods for writing Web Components
- Provided best practices for defining Custom Elements
- Walked through dealing with component attributes
- Demonstrated usage of component properties
- Showed how to use the Shadow DOM within a Web Component