Web Components

A Primer

Meltwater

Hey!

Who is this guy?

Name: Andy Desmarais

Twitter/Github/Reddit: terodox

Website: terodox.tech

Team: Legion

Web Components

What are we talking about?

1. What’s the problem?
2. What’s a web component?
   a. Custom Elements
   b. Shadow Dom
   c. Templates
   d. Slots

1.

What’s the problem?

The framework holy wars

“Every line of frontend code is tech debt when it’s written in a framework.”

- Me

We’ve all lost sleep to the framework holy wars

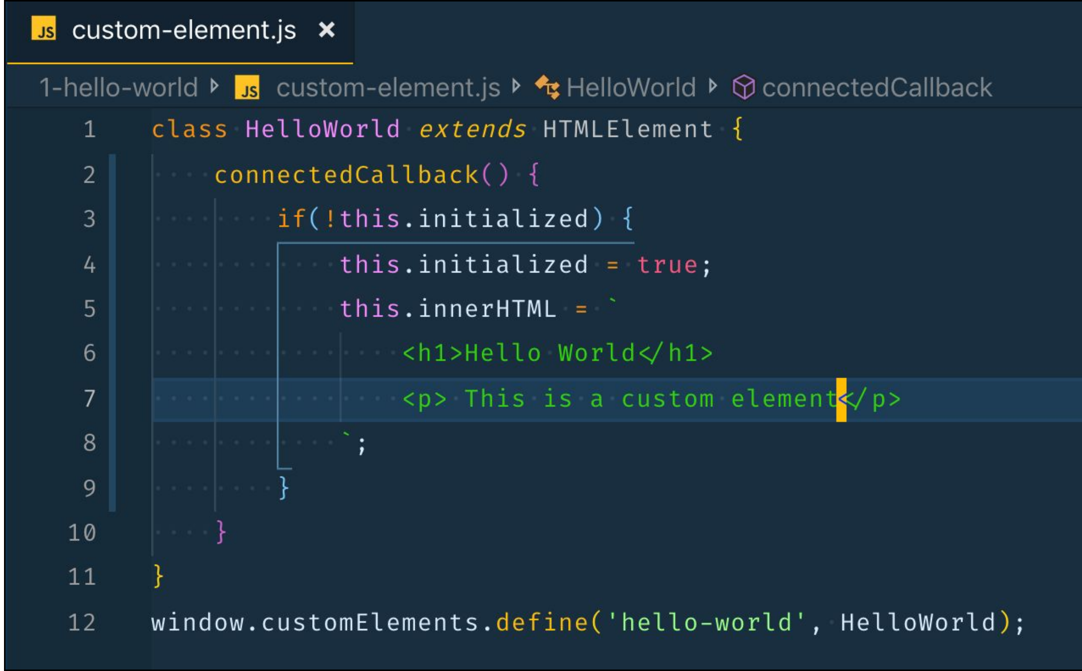
- Frameworks equate to vendor lock-in
- Evolution has a price
- You always pick wrong

In comes Web Components

2.

What’s a web component?

The web api native component.



```
class HelloWorld extends HTMLElement {
    connectedCallback() {
        if(!this.initialized) {
            this.initialized = true;
            this.innerHTML = `
                <h1>Hello World</h1>
                <p> This is a custom element</p>
            `;
        }
    }
}
window.customElements.define('hello-world', HelloWorld);
```

Three main building blocks

Custom Elements

Shadow Dom

Templates and Slots

https://link.medium.com/bSTuHsGeEX

Custom Elements

A set of JavaScript APIs that allow you to define new DOM elements and their behavior.

Custom Behavior

Simple requirements

Basic Example

Custom Element Lifecycle Callbacks

connectedCallback

Invoked each time the custom element is appended into a document-connected element. This will happen each time the node is moved, and may happen before the element's contents have been fully parsed.

disconnectedCallback

Invoked each time the custom element is disconnected from the document's DOM.

attributeChangedCallback

Invoked each time one of the custom element's attributes is added, removed, or changed. The attributes specified in a static get observedAttributes method are the only attributes that will trigger this callback.

observedAttributes

A static get method that returns an array of the attribute names that should trigger the attributeChangedCallback.

Lifecycle Callbacks Example

Shadow Dom

The Shadow DOM provides complete encapsulation of elements and styles away from the main DOM.

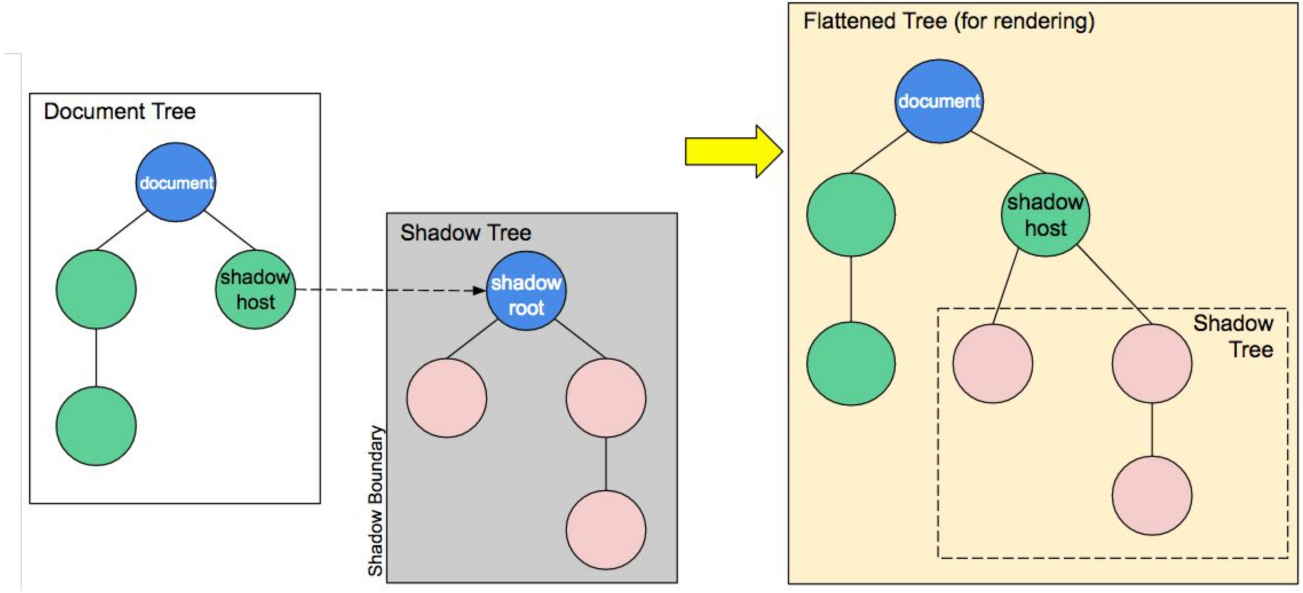


Image from MDN

Shadow DOM Modes

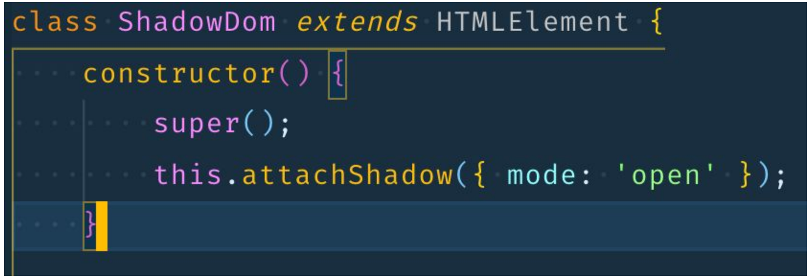
Open

When you attach a shadow root to an element in open mode, the element will have a .shadowRoot property allowing access.

Closed

This functions identical to open except that the .shadowRoot property is NOT populated to allow the shadow DOM to be modified.

Attaching a shadow DOM



```
class ShadowDom extends HTMLElement {
    constructor() {
        super();
        this.attachShadow({ mode: 'open' });
    }
```

Shadow DOM Example

Templates

When you have to reuse the same markup structures repeatedly on a web page, it makes sense to use some kind of a template rather than repeating the same structure over and over again. This was possible before, but it is made a lot easier by the HTML <template> element (which is well-supported in modern browsers). This element and its contents are not rendered in the DOM, but it can still be referenced using JavaScript.

Template Example

https://developer.mozilla.org/en-US/docs/Web/Web_Components/Using_templates_and_slots

Slots

[Slots Examples]

When we want to contain user controlled content in an element, we need slots!

Two types:

Default Slot

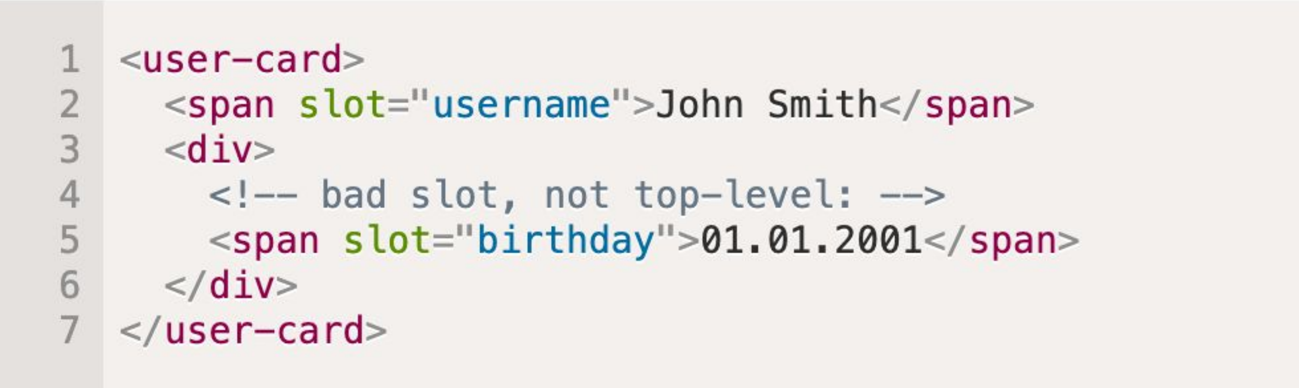
Allows a single location where all child content not in a named slot will be rendered within the custom element.

Named Slots

Slots that have specific names used to both define the slots in the custom element markup AND define the child content that should occupy each slot.

Slot Nuances

- ***Slots require use of the shadow DOM!***
- ***Slots can have fallback content defined in the custom element***
- ***Only top level children in a custom element can be slots***
  - ***Example:*** *(from here)*

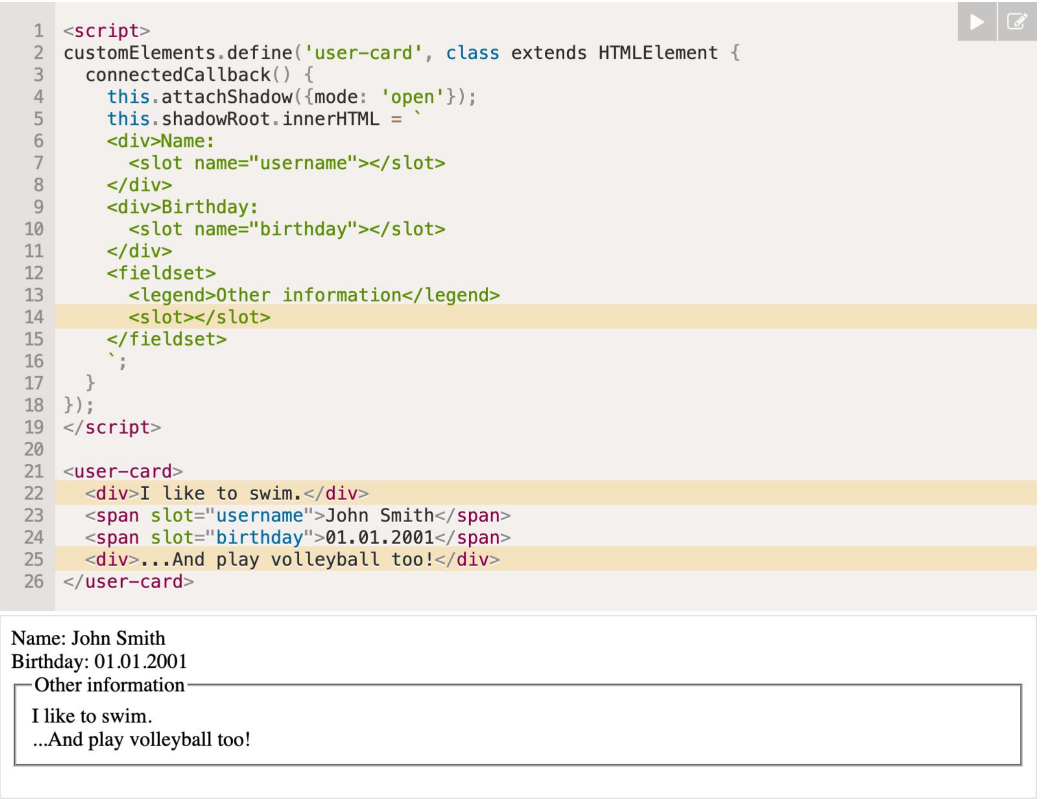


```
<user-card>
  <span slot="username">John Smith</span>
  <div>
    <!-- bad slot, not top-level: -->
    <span slot="birthday">01.01.2001</span>
  </div>
</user-card>
```

Slots Examples

Slot Nuances Continued

- ***If both named slots and a default slot are used - All content not in a named slot will go in the default slot regardless of ordering in the DOM. Example: (From here)***



```
<script>
customElements.define('user-card', class extends HTMLElement {
  connectedCallback() {
    this.attachShadow({mode: 'open'});
    this.shadowRoot.innerHTML = `
    <div>Name:
      <slot name="username"></slot>
    </div>
    <div>Birthday:
      <slot name="birthday"></slot>
    </div>
    <fieldset>
      <legend>Other information</legend>
      <slot></slot>
    </fieldset>
    `;
  }
});
</script>

<user-card>
  <div>I like to swim.</div>
  <span slot="username">John Smith</span>
  <span slot="birthday">01.01.2001</span>
  <div>...And play volleyball too!</div>
</user-card>
```

Name: John Smith
Birthday: 01.01.2001

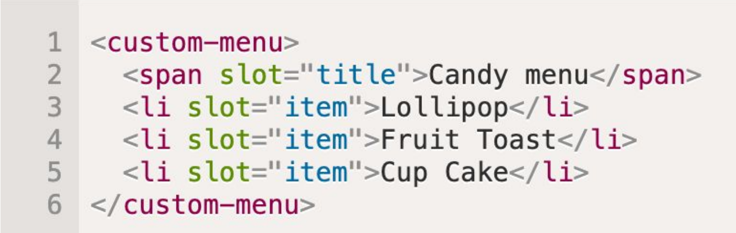
Other information

I like to swim.
...And play volleyball too!

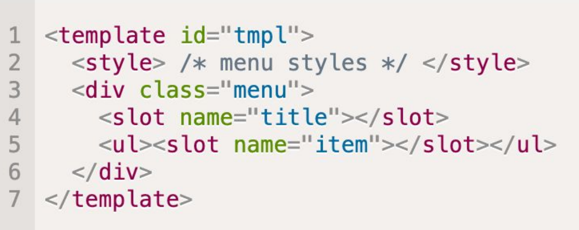
Slots Examples

Slot Nuances Continued

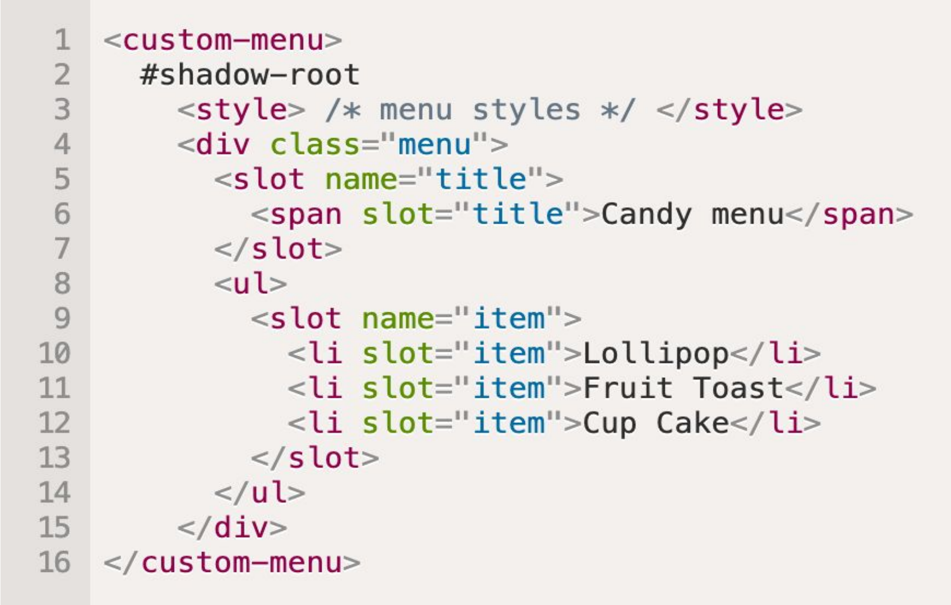
- ***If multiple elements are given to the same named slot, then all of those elements will be displayed within the slot.***
  Example: (From here)



```
<custom-menu>
  <span slot="title">Candy menu</span>
  <li slot="item">Lollipop</li>
  <li slot="item">Fruit Toast</li>
  <li slot="item">Cup Cake</li>
</custom-menu>
```



```
<template id="tmpl">
  <style> /* menu styles */ </style>
  <div class="menu">
    <slot name="title"></slot>
    <ul><slot name="item"></slot></ul>
  </div>
</template>
```



```
<custom-menu>
  #shadow-root
    <style> /* menu styles */ </style>
    <div class="menu">
      <slot name="title">
        <span slot="title">Candy menu</span>
      </slot>
      <ul>
        <slot name="item">
          <li slot="item">Lollipop</li>
          <li slot="item">Fruit Toast</li>
          <li slot="item">Cup Cake</li>
        </slot>
      </ul>
    </div>
</custom-menu>
```

Slots Examples

Slot events, nodes, and elements

- ***To see when a slot changes you can add an event listener to the slot element for the slotchange event.***
- ***To see what is actually in the slot you have two options:***
    - *slotElement.assignedNodes()*
    - *slotElement.assignedElements()*
- ***Both methods accept a flatten*** *option which will return not only this slots elements, but also the elements from any child slots!*

Slot Event Example

Questions?