INFINUM

WC, KISS

Keeping it clean with Web Components

LUKA SKUKAN

01

WEB COMPONENTS?

WHAT ARE WEB COMPONENTS?

- Several technologies for clean and reusable user interface widgets (components)
- Natively supported (kind of)
- Based on existing web standards

USING WEB COMPONENTS

- Many exist out in the wild
- Easily imported in your code, play well with (almost) everything
- Look like normal components

```
<html>
  <head>
    <link rel="import" href="../../awesome-button/awesome-button.html" />
  </head>
  <body>
    <my-awesome-button>Click me</my-awesome-button>
  </body>
</html>
```

THE BUILDING BLOCKS

- HTML Templates
- HTML Imports
- Shadow DOM
- Custom Elements

HTML TEMPLATES

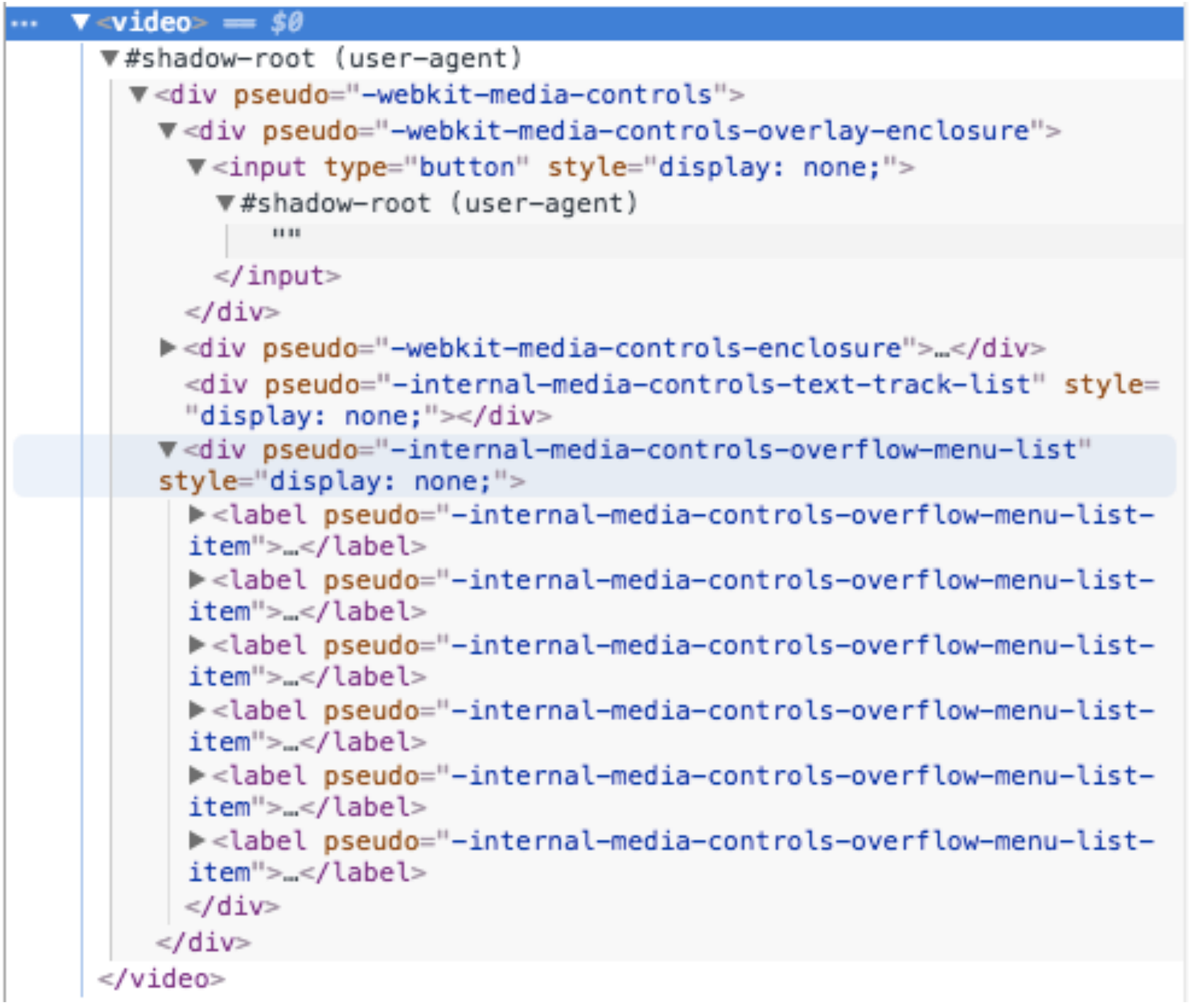
- New type of element (<template>...</template>)
- Not rendered on page load, instantiated with JS
- Fragments of prepared content
- When added to document, scripts are executed, resources fetched, styles rendered
- No templating engine in the core spec :(

HTML IMPORTS

- <link rel="import" href="./my/path.html">
- Allows for HTML to be included in the document
- Document is imported as read-only (no open, write or close on imported Document instance)

SHADOW DOM

- DOM, hidden from the user
- Hide complex structures behind “only” a tag
- Used by browser vendors, now we all can access it


▼<video> == $0
▼#shadow-root (user-agent)
▼
▼
▼<input type="button" style="display: none;">
▼#shadow-root (user-agent)
""
</input>

▶ …

▼
▶<label pseudo="-internal-media-controls-overflow-menu-list-item">…</label>
▶<label pseudo="-internal-media-controls-overflow-menu-list-item">…</label>
▶<label pseudo="-internal-media-controls-overflow-menu-list-item">…</label>
▶<label pseudo="-internal-media-controls-overflow-menu-list-item">…</label>
▶<label pseudo="-internal-media-controls-overflow-menu-list-item">…</label>
▶<label pseudo="-internal-media-controls-overflow-menu-list-item">…</label>

</video>

MAKING IT DO OUR BIDDING

- Create **shadow tree** at **shadow root**
- Inside the shadow root, just regular DOM
- Always attached to a “light” tree through a host node
- Can have **slots** for content
- Open vs Closed (always use open)

SHADOW DOM CSS

- External styles do not apply, styles do not bleed in, identifiers unique
- Styling host light DOM component with `:host` (but low specificity)
- `:host-context(<selector>)`, `::slotted(<selector>)`, imported stylesheets...

EXAMPLE

```
<template>...</template>

<script>
  function create() {
    const elem = document.createElement('div');
    const shadowRoot = elem.attachShadow({mode: 'open'});
    const template = document.querySelector('template');

    shadowRoot.appendChild(template.content.cloneNode(true));

    return elem;
  }
</script>
```

CUSTOM ELEMENTS

- Designing new HTML elements
- Autonomous custom elements vs. customised built-in elements
- HTML + JS + CSS = Component magic

CUSTOM ELEMENT TYPES

- Autonomous elements are entirely new elements based on HTMLElement, behaviour basically from scratch. Used as tags

- Extending existing elements (e.g., for styling), keeps existing behaviour (states, tabindex, ...)

- Rule of thumb: If you need a button, extend a button. If you need a novel component, make an autonomous one

DEFINING ELEMENTS

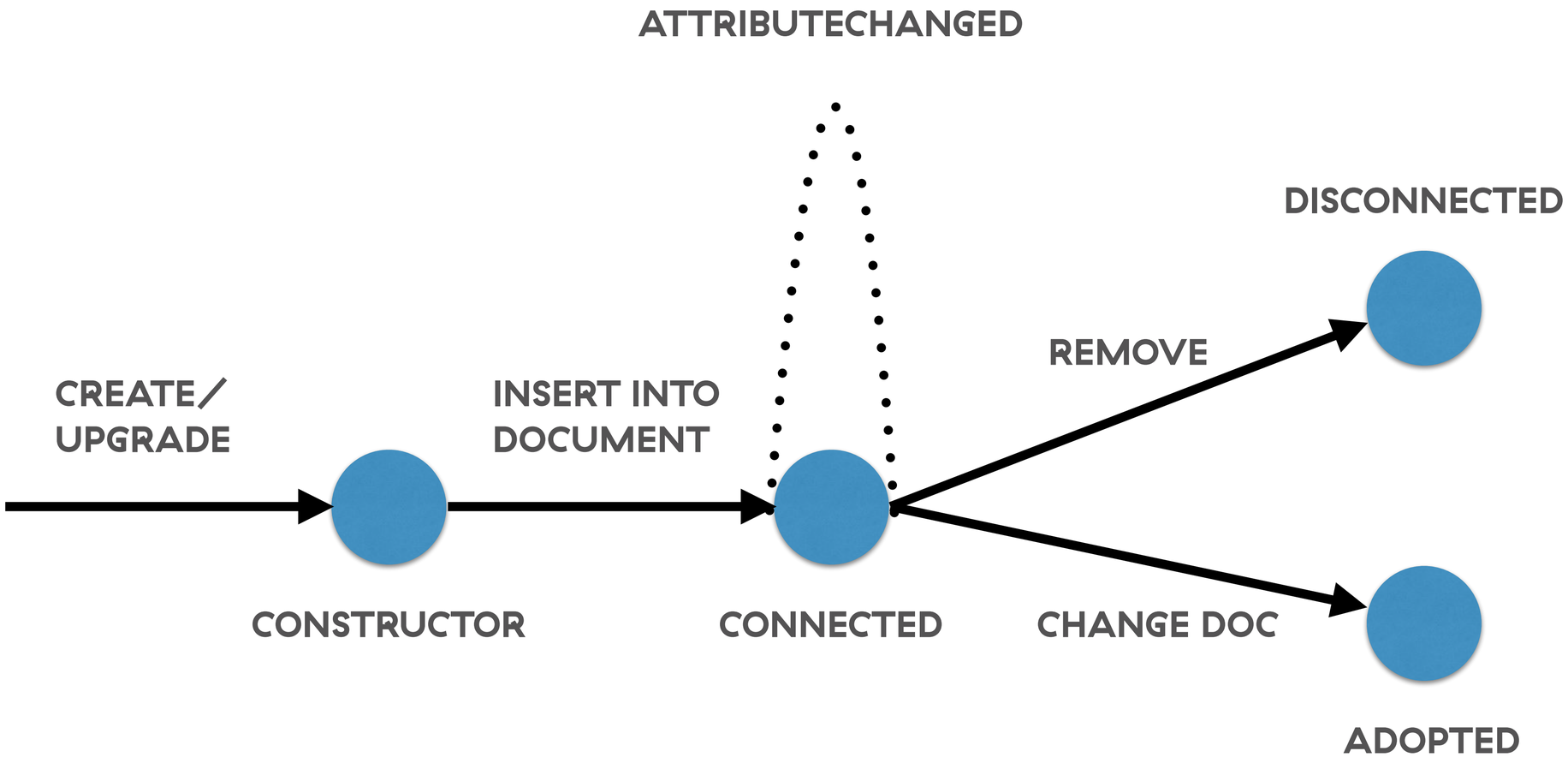
```
// Create
class MyComponent extends HTMLElement {}
customElements.define('my-component', MyComponent);

// Extend
class FancyButton extends HTMLButtonElement {}
customElements.define('my-button', FancyButton, {extends: 'button'});
```

ALSO USED DIFFERENTLY!

```
<my-component>Content</my-component>
<button is="my-button">Click Me</button>
```

WEB COMPONENT LIFECYCLE



02

DEMO

https://github.com/the-overengineer/wc-demo

03

PLAYING WITH OTHERS

Browser support	CHROME	OPERA	FIREFOX	SAFARI	EDGE
TEMPLATES	STABLE	STABLE	STABLE	STABLE	STABLE
IMPORTS	STABLE	STABLE	POLYFILL ON HOLD	POLYFILL ON HOLD	POLYFILL CONSIDERING
CUSTOM ELEMENTS	STABLE	STABLE	POLYFILL DEVELOPING	STABLE	POLYFILL CONSIDERING
SHADOW DOM	STABLE	STABLE	POLYFILL DEVELOPING	STABLE	POLYFILL CONSIDERING

FRAMEWORKS

- Similar to Vue in dev style
- Angular 2+ has similar approach to components, interoperable
- Others?
- It’s just an element!

EXTENSIONS OF WEB COMPONENTS

- Web Components are pretty basic
- Libraries written to make writing them easier and nicer
- Polymer, X-Tag and Bosonic are the primary contenders
- Merit a talk of their own

V0 OR V1

- **Important:** The current version is version 1 of the spec
- A lot of documentation (even on the official page!) about the V0 specification
- Be very careful what you're looking at!

04

IS IT WORTH IT?

WEB COMPONENTS VS VUE

- Vue actually inspired by Polymer
- Vue better with performance, has state management
- Vue is richer than WCs themselves, but comparable to Polymer or X-Tags
- No VDOM in WC (by default)

WEB COMPONENTS VS REACT

- Actually solve different problems (reusable, encapsulated components vs declarative style and syncing the DOM)
- Can be used together (but are usually not)
- Normal elements in web components, or reactDOM in in WC

DRAWBACKS

- Simple attributes (passing objects is a pain)
- Boilerplate-y
- No built-in templating

SHOULD WE USE IT?

- Used as framework for whole sites? – I don’t think so
- Dip in for low level components or cross-project? – Absolutely!

REFERENCES

- https://www.webcomponents.org/
- http://w3c.github.io/webcomponents/
- http://jonrimmer.github.io/are-we-componentized-yet/
- https://vuejs.org/v2/guide/comparison.html
- https://facebook.github.io/react/docs/web-components.html
- https://developers.google.com/web/fundamentals/getting-started/primers/shadowdom

INFINUM

Any questions?

LUKA.SKUKAN@INFINUM.CO
STILL DON'T HAVE TWITTER

Visit infinum.co or find us on social networks:

infinum.co infinumco infinumco infinum