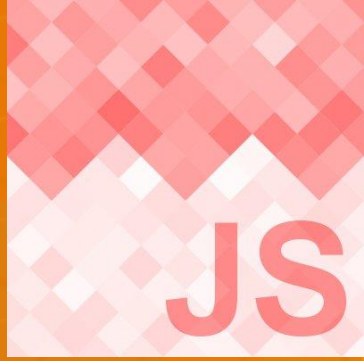
Modularity with WebComponents and Polymer



FrankenJS
Würzburg
2017

Servus!

I am Alex Sherekin

@AlexSherekin
alexandr.sherekin@gmail.com

Why modularity matters?

Why modularity matters?

- Reuse code

Why modularity matters?

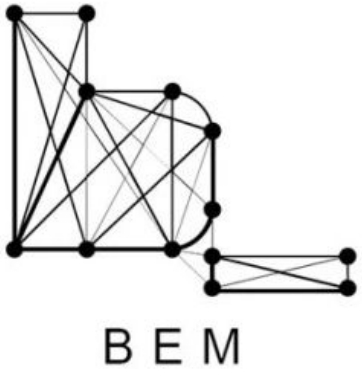
- Reuse code
- Easy to understand and support

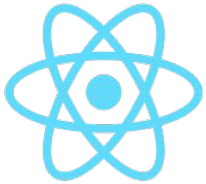
Why modularity matters?

- Reuse code
- Easy to understand and support
- Easy to share

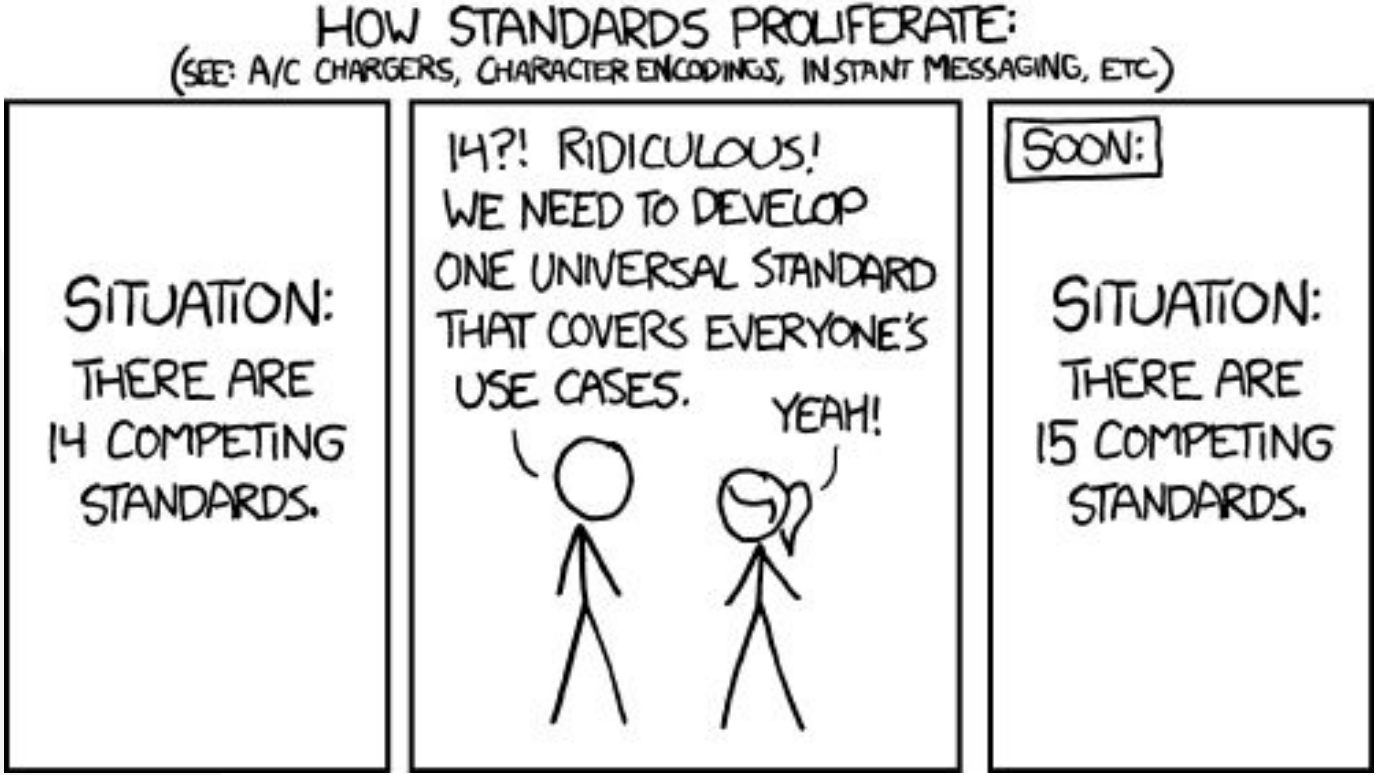
Why modularity matters?

- Reuse code
- Easy to understand and support
- Easy to share
- Testability

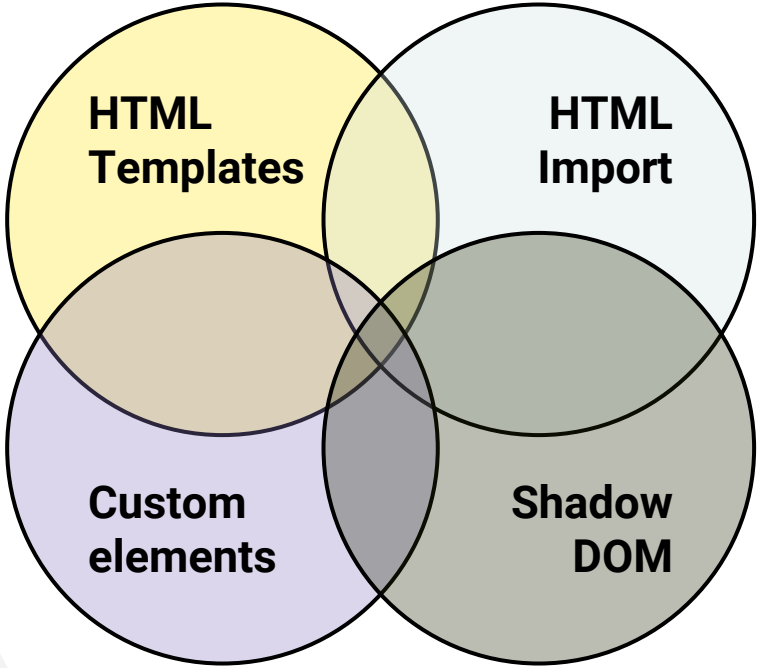




What about the modern web?


HOW STANDARDS PROLIFERATE:
(SEE: A/C CHARGERS, CHARACTER ENCODINGS, INSTANT MESSAGING, ETC)
SITUATION: THERE ARE 14 COMPETING STANDARDS.
14?! RIDICULOUS! WE NEED TO DEVELOP ONE UNIVERSAL STANDARD THAT COVERS EVERYONE'S USE CASES.
YEAH!
SOON:
SITUATION: THERE ARE 15 COMPETING STANDARDS.

What are WebComponents?



HTML Templates

HTML Templates

```
<div id="template" style="display: none;">
    <div class="template-message">Some message</div>
    <img src="exclamation.png">
</div>
```

HTML Templates

```
<div id="template" style="display: none;">
    <div class="template-message">Some message</div>
    <img src="exclamation.png">
</div>
```

```
<script id="template" type="text/x-template">
    <div class="template-message">Some message</div>
    <img src="exclamation.png">
</script>
```

HTML Templates

```
<template id="inert-template">
    <div class="template-message">Some message</div>
    <img src="exclamation.png">
</template>
```

HTML Templates

```
<template id="inert-template">
    <div class="template-message">Some message</div>
    <img src="exclamation.png">
</template>
```

```
var templateContent = document.querySelector("#inert-template").content;

var templateCopy = templateContent.cloneNode(true);
templateCopy.querySelector("img").src = "myImage.png";

document.body.appendChild(templateCopy);
```

HTML Import

HTML Import

```
<link rel="import" href="myComponent.html"
    onload="handleLoad(event)" onerror="handleError(event)">
```

HTML Import

```
<link rel="import" href="myComponent.html"
    onload="handleLoad(event)" onerror="handleError(event)">
```

```
var content = document.querySelector("link[rel=import]").import;
//var element = content.querySelector(".some-class");
document.body.appendChild(content.cloneNode(true));
```

Custom Elements

Custom Elements

- Create own tags

```
<div id="registration-block">
    ...
</div>
```

```
<registration-block>
    ...
</registration-block>
```

Custom Elements

- Extend native elements

```
<button is="fancy-button">
    ...
</button>
```

Define and Use Custom Element

```
class MyWindow extends HTMLElement { /* implementation */ }
window.customElements.define('my-window', MyWindow);



window.customElements.define('my-window', class extends HTMLElement { /* implementation */ });
```

Define and Use Custom Element

```
class MyWindow extends HTMLElement { /* implementation */ }
window.customElements.define('my-window', MyWindow);



window.customElements.define('my-window', class extends HTMLElement { /* implementation */ });
```

```
var instance = document.createElement('my-window');
document.body.appendChild(instance);
```

```
<my-window></my-window>
```

Define and Use Inherited Custom Element

```
class CustomCheckBox extends HTMLInputElement { /* implementation */ }
window.customElements.define('custom-check-box', CustomCheckBox, {
    extends: 'input'
});

window.customElements.define('custom-check-box', class extends HTMLInputElement { /* implementation */ }, {
    extends: 'input'
});
```

Define and Use Inherited Custom Element

```
class CustomCheckBox extends HTMLInputElement { /* implementation */ }
window.customElements.define('custom-check-box', CustomCheckBox, {
    extends: 'input'
});

window.customElements.define('custom-check-box', class extends HTMLInputElement { /* implementation */ }, {
    extends: 'input'
});
```

```
var instance = document.createElement('input', 'custom-check-box');
document.body.appendChild(instance);
```

```
<input is='custom-check-box'></input>
```

Custom Element's Lifecycle

```
static get observedAttributes() {
    return ["light"];
}

// Element's lifecycle
// Restrictions - no operations with attributes and child nodes
constructor() {
    super();
    this.attachShadow({ ···
    });
}

// Element is in DOM now. Good time for attaching event listeners, doing rendering ...
connectedCallback() { ···
}

// Element is not in DOM now. Clean up stuff
disconnectedCallback() { ···
}

// Attribute was added, removed, replaced or updated
attributeChangedCallback(attr, oldValue, newValue) {
    if (attr == "light") { ···
    }
}

// Element has been moved into a new document (after calling document.adoptNode(el))
adoptedCallback() {}
```

Custom Element's Lifecycle

```
static get observedAttributes() {
    return ["light"];
}

// Element's lifecycle
// Restrictions - no operations with attributes and child nodes
constructor() {
    super();
    this.attachShadow({ ···
    });
}

// Element is in DOM now. Good time for attaching event listeners, doing rendering ...
connectedCallback() { ···
}

// Element is not in DOM now. Clean up stuff
disconnectedCallback() { ···
}

// Attribute was added, removed, replaced or updated
attributeChangedCallback(attr, oldValue, newValue) {
    if (attr == "light") { ···
    }
}

// Element has been moved into a new document (after calling document.adoptNode(el))
adoptedCallback() {}
```

Custom Element's Lifecycle

```
static get observedAttributes() {
    return ["light"];
}

// Element's lifecycle
// Restrictions - no operations with attributes and child nodes
constructor() {
    super();
    this.attachShadow({ ···
    });
}

// Element is in DOM now. Good time for attaching event listeners, doing rendering ...
connectedCallback() { ···
}

// Element is not in DOM now. Clean up stuff
disconnectedCallback() { ···
}

// Attribute was added, removed, replaced or updated
attributeChangedCallback(attr, oldValue, newValue) {
    if (attr == "light") { ···
    }
}

// Element has been moved into a new document (after calling document.adoptNode(el))
adoptedCallback() {}
```

Custom Element's Lifecycle

```
static get observedAttributes() {
    return ["light"];
}

// Element's lifecycle
// Restrictions - no operations with attributes and child nodes
constructor() {
    super();
    this.attachShadow({ ···
    });
}

// Element is in DOM now. Good time for attaching event listeners, doing rendering ...
connectedCallback() { ···
}

// Element is not in DOM now. Clean up stuff
disconnectedCallback() { ···
}

// Attribute was added, removed, replaced or updated
attributeChangedCallback(attr, oldValue, newValue) {
    if (attr == "light") { ···
    }
}

// Element has been moved into a new document (after calling document.adoptNode(el))
adoptedCallback() {}
```

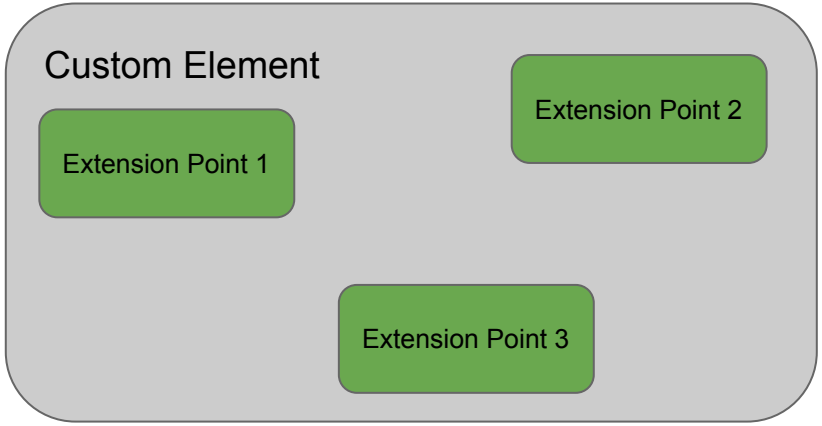
Custom Element's Lifecycle

```
static get observedAttributes() {
    return ["light"];
}

// Element's lifecycle
// Restrictions - no operations with attributes and child nodes
constructor() {
    super();
    this.attachShadow({ ···
    });
}

// Element is in DOM now. Good time for attaching event listeners, doing rendering ...
connectedCallback() { ···
}

// Element is not in DOM now. Clean up stuff
disconnectedCallback() { ···
}

// Attribute was added, removed, replaced or updated
attributeChangedCallback(attr, oldValue, newValue) {
    if (attr == "light") { ···
    }
}

// Element has been moved into a new document (after calling document.adoptNode(el))
adoptedCallback() {}
```

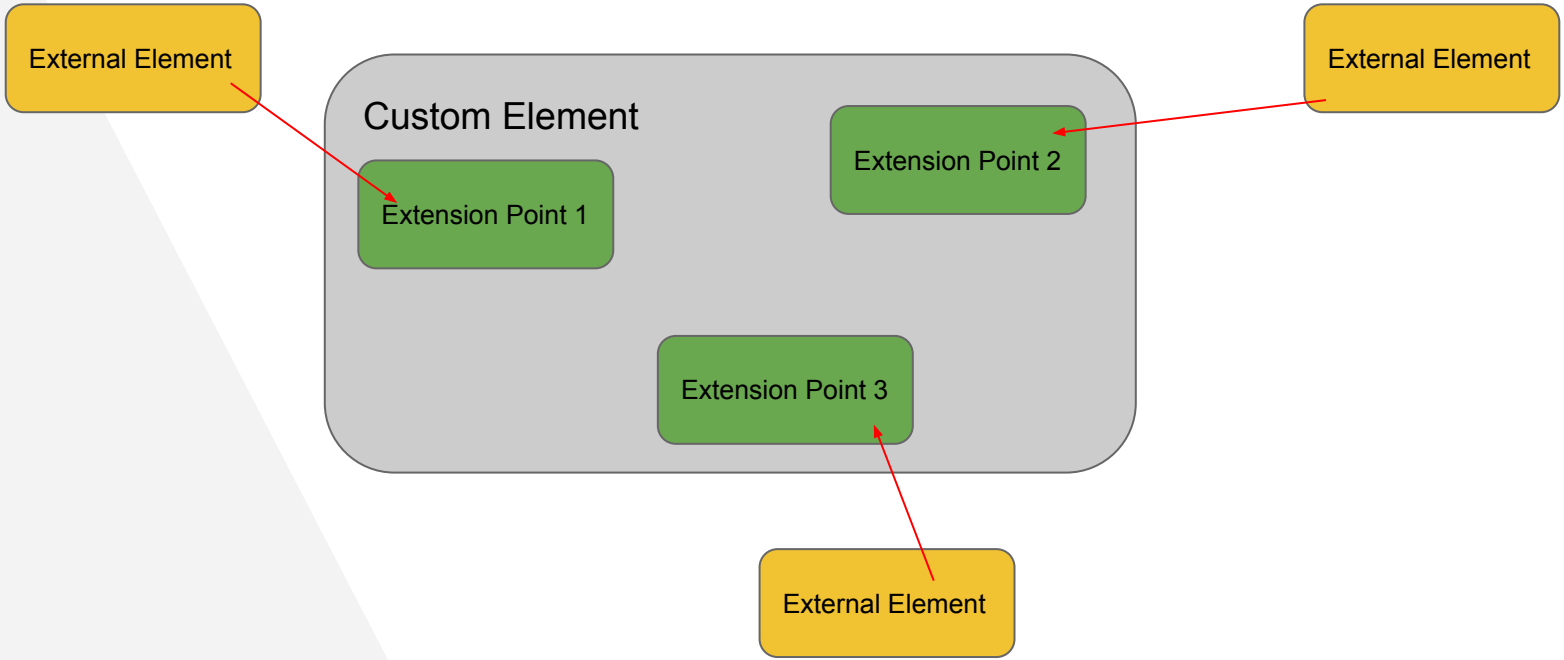
Map attribute to property

```
static get observedAttributes() {
    return ["light"];
}

// Attribute was added, removed, replaced or updated
attributeChangedCallback(attr, oldValue, newValue) {
    if (attr == "light") {

        this.light = newValue;
    }
}

get light() {
    return this.getAttribute("light");
}
set light(value) {
    if (value === this.light) {
        return;
    }

    if (value === "on") {
        this.setAttribute("light", "on");
    } else {
        this.setAttribute("light", "off");
    }

    this.content.style.setProperty("background-image", `url(./components/my-window/window_light_${value === "on" ? "on": "off"}.png)`);
}
```

Composition



Composition



Composition

```
<div class="content">
    <div class="roof-placeholder">
        <slot name="roof"></slot>
    </div>
    <slot></slot>
</div>
```

Composition

```
<div class="content">
    <div class="roof-placeholder">
        <slot name="roof"></slot>
    </div>
    <slot></slot>
</div>
```

```
<my-house>
    <my-roof slot="roof"></my-roof>
    <my-floor>
        <my-window slot="windowLeft"></my-window>
        <my-window slot="windowRight"></my-window>
    </my-floor>
    <my-floor>
        <my-door slot="door"></my-door>
        <my-window slot="windowLeft"></my-window>
    </my-floor>
</my-house>
```

Shadow DOM

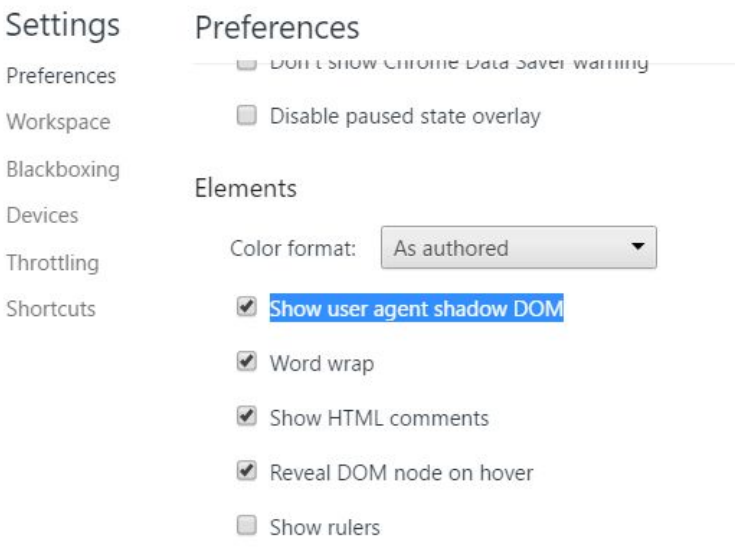
Shadow DOM


▼

▼
▼
▼<div class="html5-video-player ad-created remote-created ytp-hide-info-bar ytp-iv-drawer-enabled videoAdUiRedesign ytp-video-ad-learn-more-ui iv-module-created iv-module-loaded creatorendscreen-created creatorendscreen-loaded paused-mode" tabindex="-1" id="movie_player" data-version="//s.ytimg.com/yts/jsbin/player-en_US-vflR62D9G/base.js" aria-label="YouTube Video Player">
▼
<video tabindex="-1" class="video-stream html5-main-video" style="width: 640px; height: 360px; left: 0px; top: 0px; opacity: 1;" src="blob:https://www.youtube.com/8adad7d9-87fc-4203-9cab-423448d91e44"></video> == $0

▶ …
▶<button class="ytp-button ytp-cards-button" aria-label="Show cards" aria-owns="ytp-cards" aria-haspopup="true" data-layer="2" title="From YOUGIFTED - канал о спорте">…</button>
▶ …
▶ …
▶

Shadow DOM


Settings
Preferences
Workspace
Blackboxing
Devices
Throttling
Shortcuts
Preferences
Disable paused state overlay
Elements
Color format:
As authored
Show user agent shadow DOM
Word wrap
Show HTML comments
Reveal DOM node on hover
Show rulers

Shadow DOM


▼<video tabindex="-1" class="video-stream html5-main-video" style="width: 640px; height: 360px; left: 0px; top: 0px; opacity: 1;" src="blob:https://www.youtube.com/8adad7d9-87fc-4203-9cab-423448d91e44"> == $0
▼#shadow-root (user-agent)
▼
▼
▶<input type="button" style="display: none;">…</input>

▼
▶ …

▼
▶<label pseudo="-internal-media-controls-overflow-menu-list-item" style="display: none;">…</label>
▶<label pseudo="-internal-media-controls-overflow-menu-list-item" style="display: none;">…</label>
▶<label pseudo="-internal-media-controls-overflow-menu-list-item" style="display: none;">…</label>
▶<label pseudo="-internal-media-controls-overflow-menu-list-item" style="display: none;">…</label>
▶<label pseudo="-internal-media-controls-overflow-menu-list-item" style="display: none;">…</label>
▶<label pseudo="-internal-media-controls-overflow-menu-list-item" style="display: none;">…</label>

</video>

Shadow DOM

- Isolated DOM

Shadow DOM

- Isolated DOM
- Scoped CSS

Shadow DOM

- Isolated DOM
- Scoped CSS
- Simplified CSS

How to Create Shadow DOM?

```
class MyWindow extends HTMLElement {

    static get observedAttributes() {
        return ["light"];
    }

    // Element's lifecycle
    // Restrictions - no operations with attributes and child nodes
    constructor() {
        super();
        this.attachShadow({
            mode: "open"
        });
    }

    // Element is in DOM now. Good time for attaching event listeners, doing rendering ...
    connectedCallback() {
        this.shadowRoot.appendChild(template.content.cloneNode(true));

        this.content = this.shadowRoot.querySelector(".content");
        this.addEventListener("click", this.clickHandler);
    }
```

Style Injection

```
:host {
    display: block;
    width: 100%;
    height: 100%;
}

.content {
    width: 100%;
    height: 100%;
    background-image: url(./components/my-window/window_light_off.png);
    background-size: 100% 100%;
}

.content:hover {
    opacity: var(--hover-opacity, 0.8);
}
```

Style Injection

```
<my-house>
    <my-roof slot="roof"></my-roof>
    <my-floor>
        <my-window slot="windowLeft"></my-window>
        <my-window slot="windowRight"></my-window>
    </my-floor>
    <my-floor>
        <my-door slot="door"></my-door>
        <my-window slot="windowLeft"></my-window>
    </my-floor>
</my-house>
```

```
my-house my-floor:last-child my-window {
    --hover-opacity: 0.1;
}
```

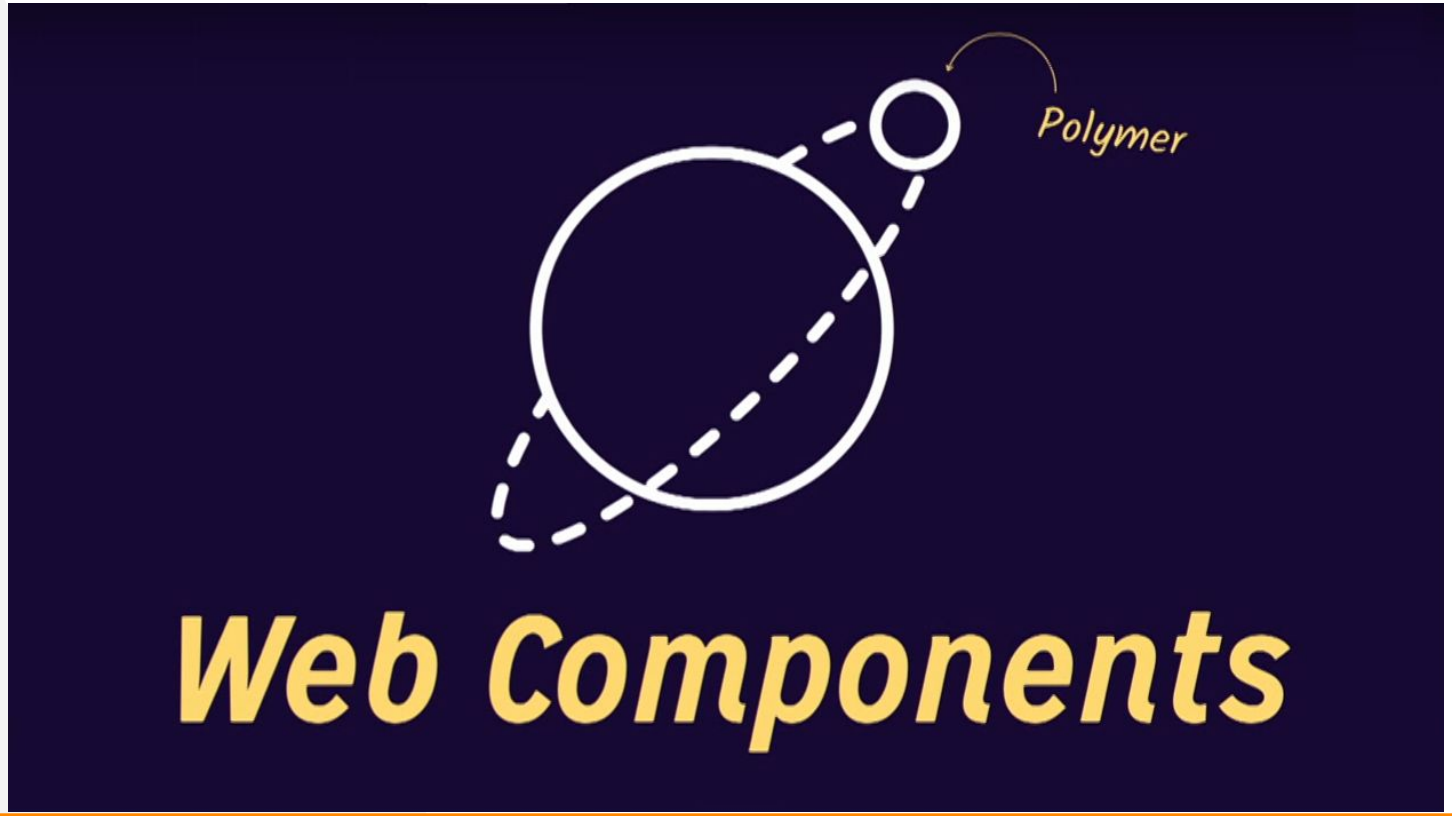
Browsers Support

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	POLYFILL DEVELOPING	POLYFILL CONSIDERING
SHADOW DOM	STABLE	STABLE	POLYFILL DEVELOPING	STABLE	POLYFILL CONSIDERING

Polymer

What is Polymer?

What is Polymer?



What is Polymer?

- Not a framework
- Very near to native API
- Opinionated usage of WebComponents
- Reduce boilerplate and increase productivity

New Super Class

```
class MyWindow extends Polymer.Element {
    static get is() {
        return "my-window";
    }

    static get config() {
        return {
            properties: {
                light: {
                    value: "off",
                    type: String,
                    reflectToAttribute: true,
                    observer: "lightChanged"
                }
            }
        };
    }

    lightChanged(newValue, oldValue) {···
    }
```

New Super Class

```
class MyWindow extends Polymer.Element {
    static get is() {
        return "my-window";
    }

    static get config() {
        return {
            properties: {
                light: {
                    value: "off",
                    type: String,
                    reflectToAttribute: true,
                    observer: "lightChanged"
                }
            }
        };
    }

    lightChanged(newValue, oldValue) {···
    }
```

New Super Class

```
class MyWindow extends Polymer.Element {
    static get is() {
        return "my-window";
    }

    static get config() {
        return {
            properties: {
                light: {
                    value: "off",
                    type: String,
                    reflectToAttribute: true,
                    observer: "lightChanged"
                }
            }
        };
    }

    lightChanged(newValue, oldValue) {···
    }
```

Dom-module Wrapper

```
<dom-module id="my-window">
    <template>
        <style>
            :host {…
            }
            #content {…
            }
            #content:hover {…
            }
        </style>
        <div id="content" on-click="clickHandler"></div>
    </template>

    <script>
        class MyWindow extends Polymer.Element {
            static get is() {
                return "my-window";
            }
```

Event Handlers

```
<dom-module id="my-window">
    <template>
        <style>
            :host {…
            }
            #content {…
            }
            #content:hover {…
            }
        </style>
        <div id="content" on-click="clickHandler"></div>
    </template>

    <script>
        class MyWindow extends Polymer.Element {
            static get is() {
                return "my-window";
            }
```

Event Handlers

```
lightChanged(newValue, oldValue) {
    this.$.content.style.setProperty("background-image", `url(./components/my-window/window_light_${newValue === "on" ? "on": "off"}.png)`);
}

clickHandler(event, detail) {
    this.light = (this.light === "on") ? "off" : "on";
}
```

Multiple Observers

```
static get config() {
    return {
        properties: {
            light: {···
            },
            foo: {···
            }
        },
        observers: ["fooOrLightChanged(foo, light)"]
    };
}

fooOrLightChanged(foo, light) {
    // do some stuff
}
```

Data Binding

```
    <div id="content" on-click="clickHandler"></div>
    <div id="fooWrapper">[[foo]]</div>
</template>

<script>
    class MyWindow extends Polymer.Element {
        static get is() {…
        }

        static get config() {
            return {
                properties: {
                    light: {…
                    },
                    foo: {…
                    }
                },
                observers: ["fooOrLightChanged(foo, light)"]
            };
        }
```

Life Cycle

```
// Element's lifecycle
// Restrictions - no operations with attributes and child nodes
constructor() {
    super();
}

// Element is in DOM now. Good time for attaching event listeners, doing rendering ...
connectedCallback() {
    super.connectedCallback();
}

// Element is not in DOM now. Clean up stuff
disconnectedCallback() {
    super.disconnectedCallback();
}

// Element has been moved into a new document (after calling document.adoptNode(el))
adoptedCallback() {
    super.adoptedCallback();
}
```

THANKS!

Any questions?