Web components en 2019, on en est où ?

Horacio Gonzalez
@LostInBrittany

Who are we?

Introducing myself and introducing OVH

Horacio Gonzalez

@LostInBrittany

Spaniard lost in Brittany, developer, dreamer and all-around geek

OVH
Team DevRel

DevFest du Bout du Monde

Finist Devs

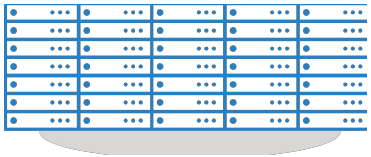
WARP 10 MEETUP

Google Developers Experts 2019
Web Technologies
GDE
Flutter

OVH: A Global Leader on Cloud

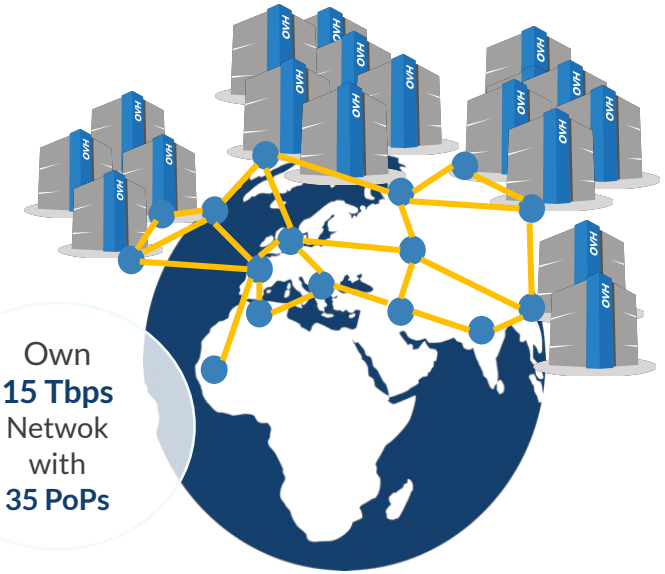
200k Private cloud VMs running

1 **Dedicated IaaS Europe**



Hosting capacity : **1.3M** Physical Servers

360k Servers already deployed



Own **15 Tbps** Netwok with **35 PoPs**

2018
27 Datacenters

2020
50 Datacenters

> **1.3M** Customers in **138** Countries

OVH : Key Figures

1.3M Customers worldwide in 138 Countries
1.5 Billions euros investment over five years
28 Datacenters (growing)
350k Dedicated Servers
200k Private cloud VMs running
650k Public cloud Instances created in a month
20TB bandwidth capacity
35 Points of presence
4TB Anti DDoS capacity
Hosting capacity : 1.3M Physical Servers

+ 2 500 Employees in 19 countries
18 Years of Innovation

Ranking & Recognition

1st **European** Cloud Provider*

1st **Hosting** provider in Europe

1st **Provider** Microsoft Exchange

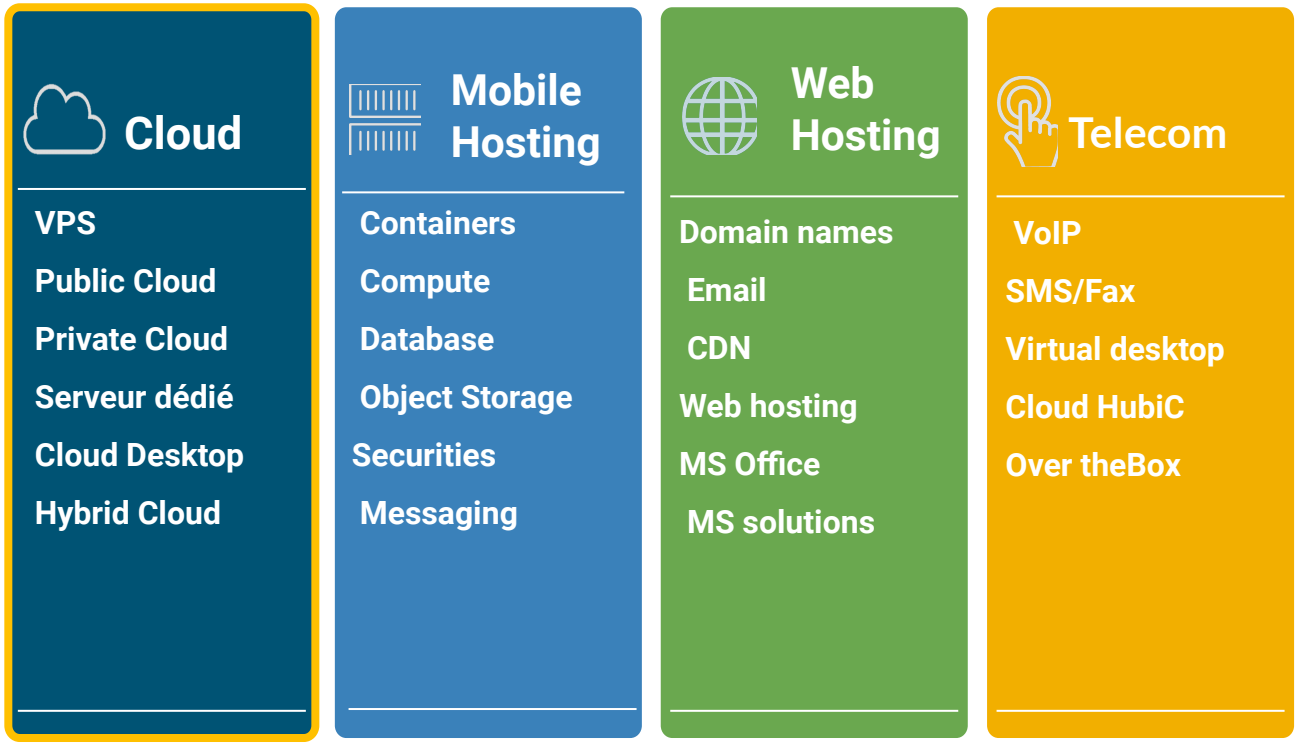
Certified vCloud Datacenter

Certified Kubernetes platform (CNCF)

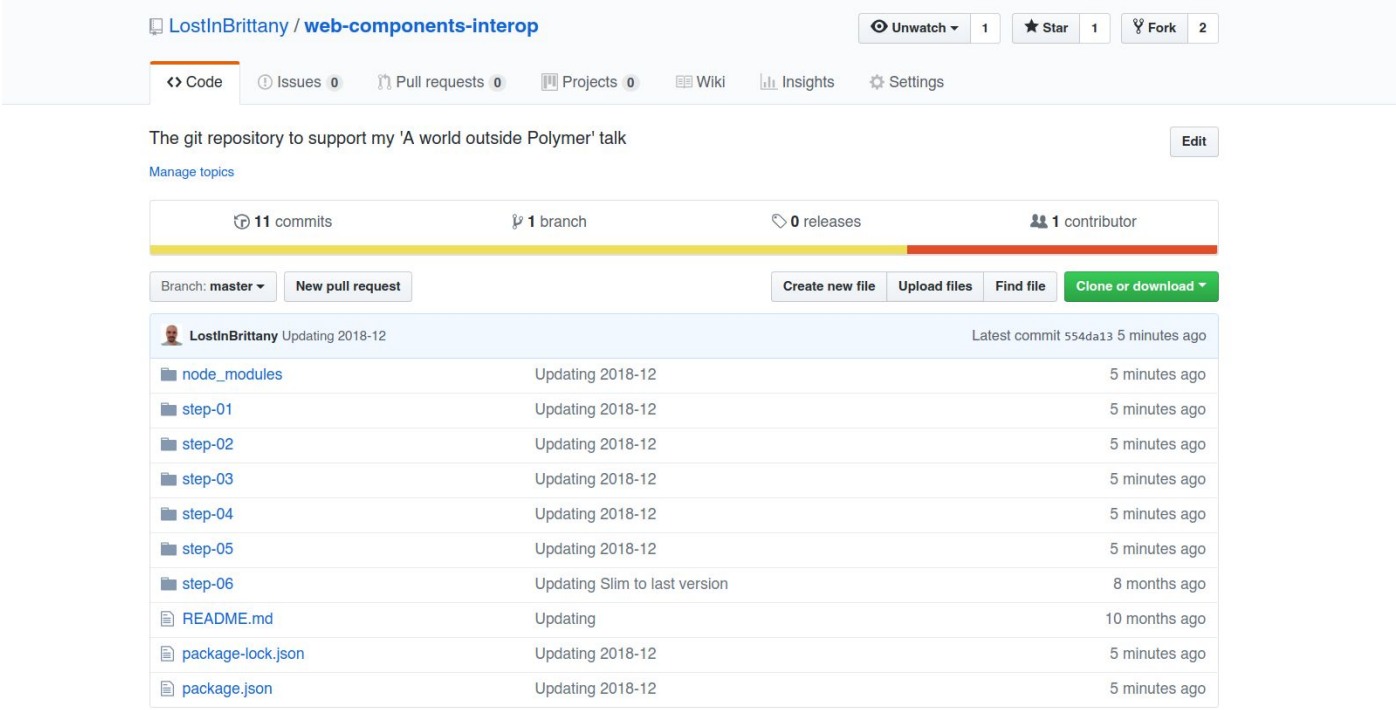
Vmware **Global Service Provider** 2013-2016

Veeam Best Cloud Partner of the year (2018)

OVH: Our solutions



We want the code!



https://github.com/LostInBrittany/web-components-interop

Some thoughts on tooling

Because complexity matters

Web dev has become complex

It seems you need lots of complex tools

But modern JavaScript makes it easy

- Supported almost everywhere
- Classes
- More expressive syntax
- Native module system

Why use tools in dev?

@pika/web

No bundling or transpiling needed

https://www.pikapkg.com/blog/pika-web-a-future-without-webpack/

The 3 minutes context

What the heck are web component?

Web Components

Web standard W3C

Web Components

Available in all modern browsers:

Firefox, Safari, Chrome

Web Components

Create your own HTML tags

Encapsulating look and behavior

Web Components

Fully interoperable

With other web components, with any framework

Web Components

CUSTOM ELEMENTS

SHADOW DOM

TEMPLATES

Custom Element

</> To define your own HTML tag

```
<body>
  ...
  <script>
    window.customElements.define('my-element',
        class extends HTMLElement {...});
  </script>
  <my-element></my-element>
</body>
```

Shadow DOM

To encapsulate subtree and style in an element


```
<button>Hello, world!</button>
<script>
var host = document.querySelector('button');
const shadowRoot = host.attachShadow({mode:'open'});
shadowRoot.textContent = 'こんにちは、影の世界!';
</script>
```

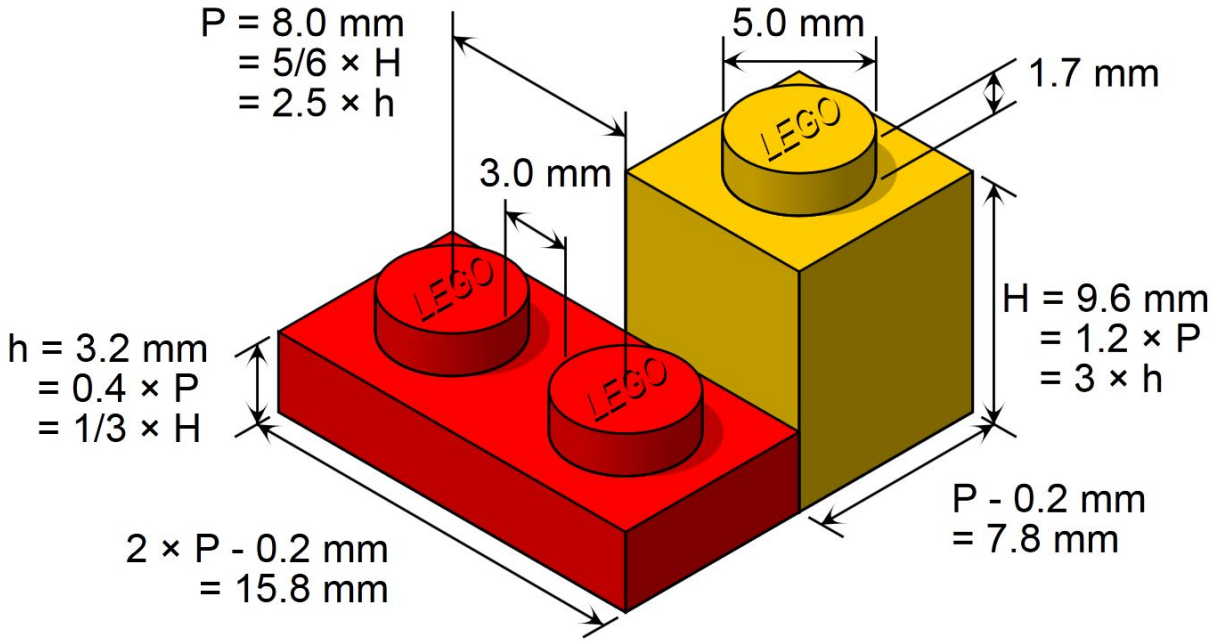
Template

To have clonable document template

```
<template id="mytemplate">
  <img src="" alt="great image">
  <div class="comment"></div>
</template>

var t = document.querySelector('#mytemplate');
// Populate the src at runtime.
t.content.querySelector('img').src = 'logo.png';
var clone = document.importNode(t.content, true);
document.body.appendChild(clone);
```

But in fact, it's just an element…

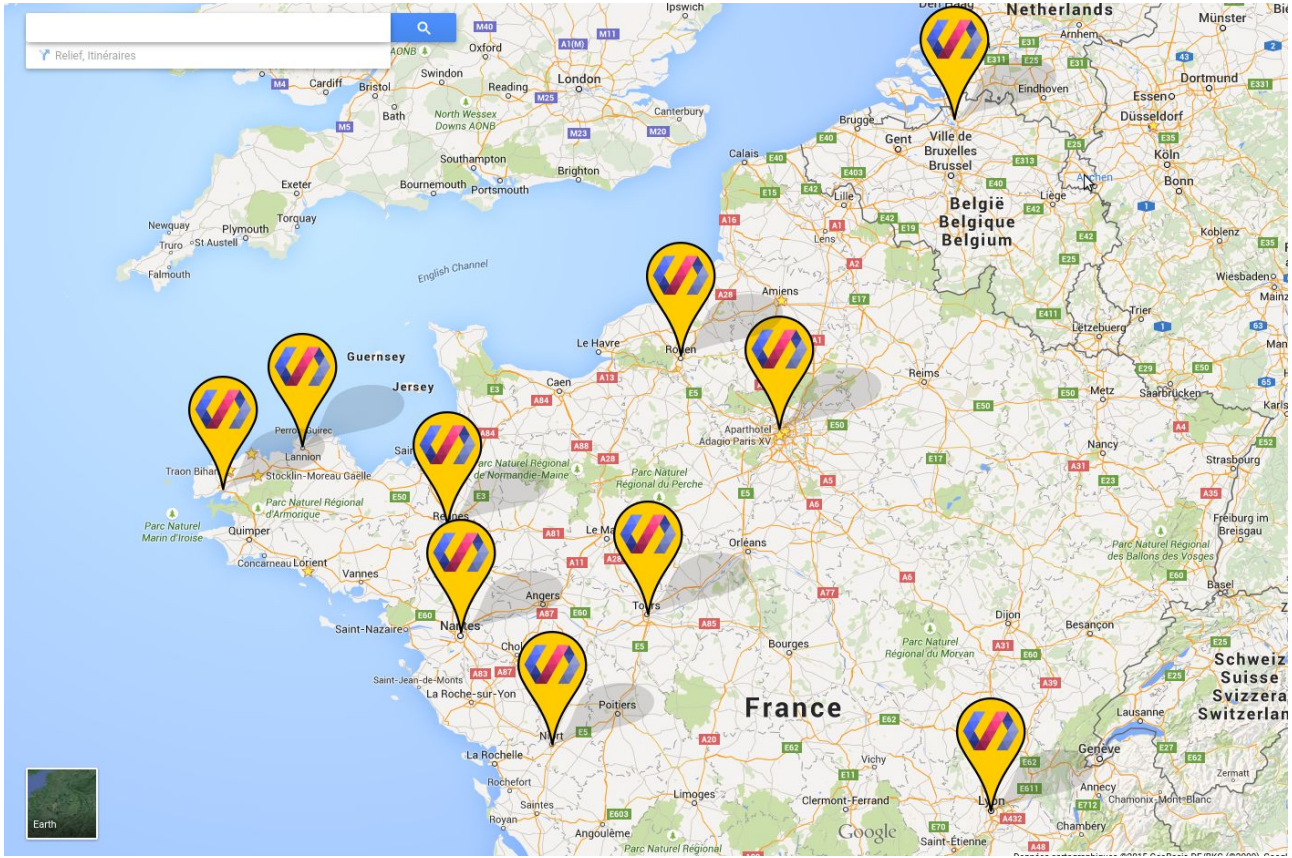
- Attributes
- Properties
- Methods
- Events



Sometimes I feel a bit grumpy

The stories of the grumpy old speaker...

On Polymer tour since 2014



Web components == Revolution

bu.edu

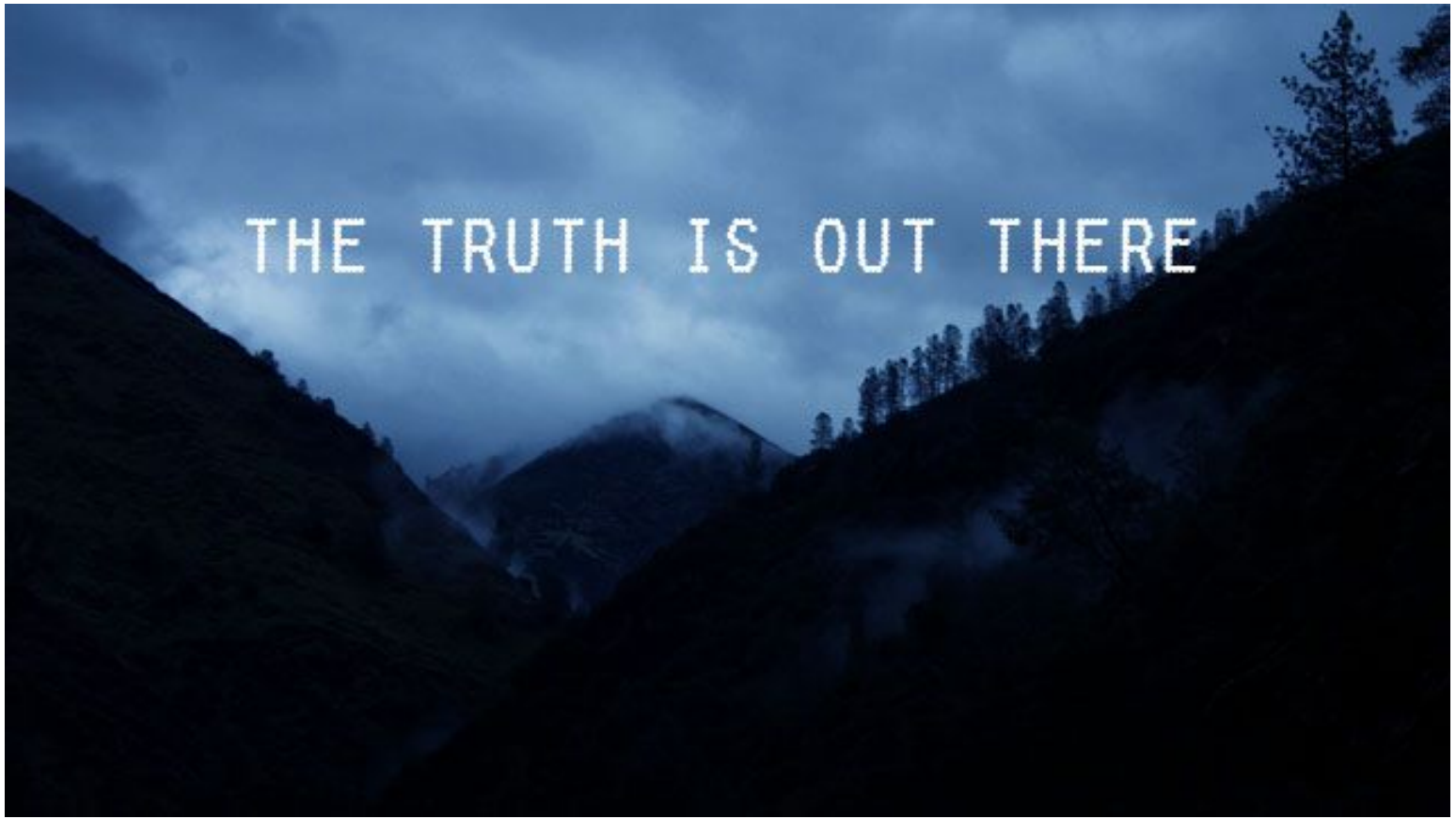
Building a world brick by brick

BitRebels Brickset

Is the promise unfulfilled?

It's 2019 now, where is your revolution, dude?

Is it a conspiracy?



Am I only a dreamer?

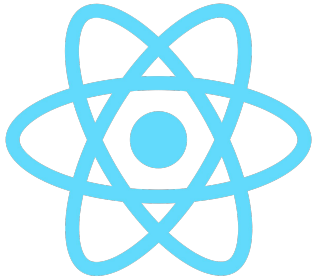
Well, revolution IS there

But it's a silent one...

They are there, in everyday sites

More than you can imagine

The components architecture won



Components, components everywhere

Web components ARE platform

Truly part of the platform...

Why those libs?

Why people don't use vanilla?

Web component standard is low level

At it should be!

Standard == basic bricks

Standard exposes an API to:

- Define elements
- Encapsulate DOM

Libraries are helpers

They give you higher-level primitives

Different high-level primitives

Each one tailored to a use

Sharing the same base

High-performant, low-level, in-the-platform web components standard

Libraries aren't a failure of standard

They happen by design

Vanilla Web Components

A very basic web component

```
class MyElement extends HTMLElement {

   // This gets called when the HTML parser sees your tag
   constructor() {
       super(); // always call super() first in the ctor.
       this.msg = 'Hello, RivieraDev!';
   }

   // Called when your element is inserted in the DOM or
   // immediately after the constructor if it's already in the DOM
   connectedCallback() {
       this.innerHTML = `<p>${this.msg}</p>`;
   }
}

customElements.define('my-element', MyElement);
```

Custom Elements:

- Let you define your own HTML tag with bundled JS behavior
- Trigger lifecycle callbacks
- Automatically “upgrade” your tag when inserted in the document

Custom Elements don’t:

- Scope CSS styles
  - Shadow DOM
- Scope JavaScript
  - ES2015
- “Reproject” children into <slot> elements
  - Shadow DOM

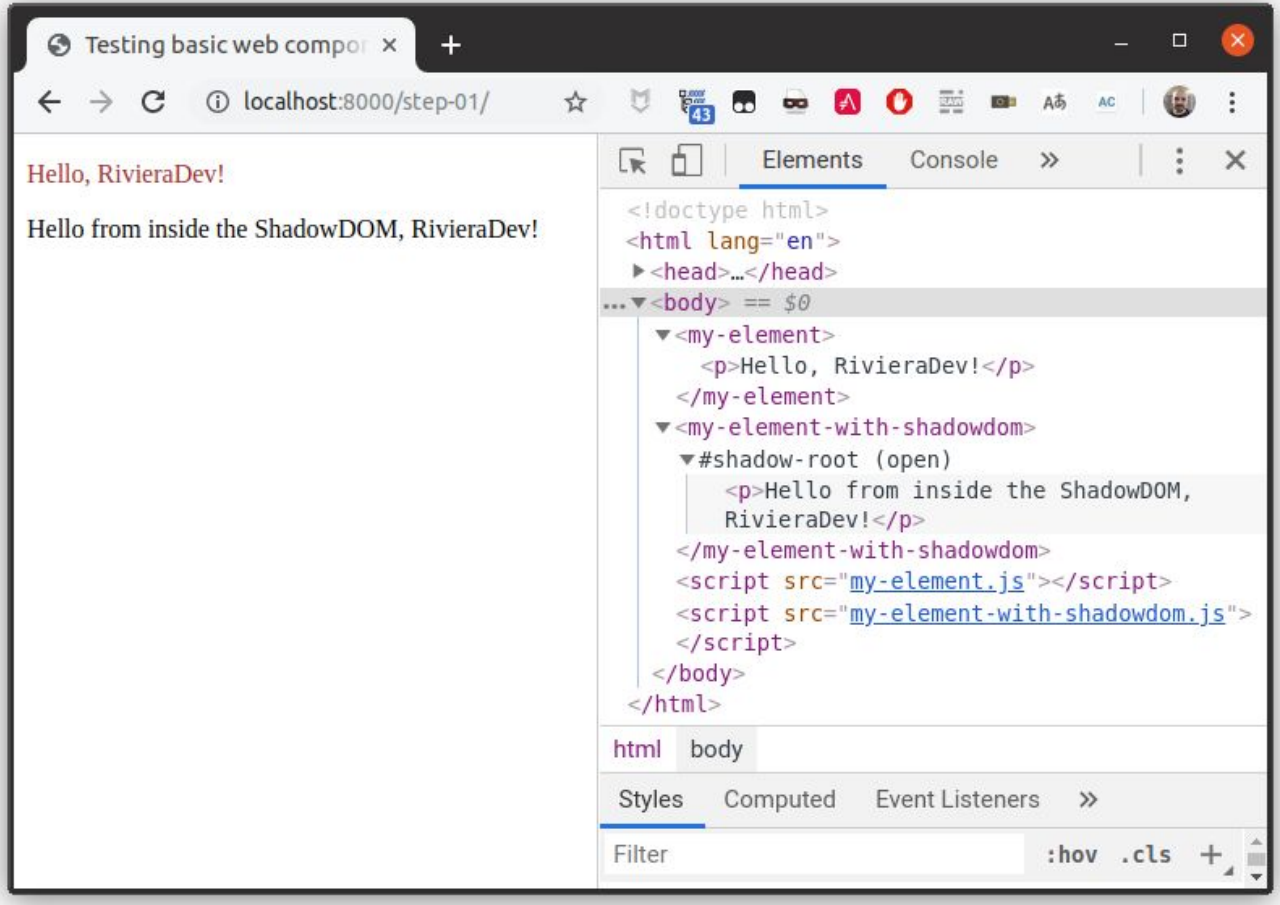
Adding ShadowDOM

```
class MyElementWithShadowDom extends HTMLElement {

   // This gets called when the HTML parser sees your tag
   constructor() {
       super(); // always call super() first in the ctor.
       this.msg = 'Hello from inside the ShadowDOM, RivieraDev!';
       this.attachShadow({ mode: 'open' });
   }

   // Called when your element is inserted in the DOM or
   // immediately after the constructor if it's already in the DOM
   connectedCallback() {
       this.shadowRoot.innerHTML = `<p>${this.msg}</p>`;
   }
}

customElements.define('my-element-with-shadowdom', MyElementWithShadowDom);
```

Using web components

```
<!doctype html>
<html lang="en">
   <head>
       <meta charset="utf-8">
       <title>Testing basic web component</title>
       <style>
           p { color: brown; }
       </style>
   </head>

   <body>
       <my-element></my-element>
       <my-element-with-shadowdom></my-element-with-shadowdom>
       <script src="my-element.js"></script>
       <script src="my-element-with-shadowdom.js"></script>
   </body>
</html>
```

Using web components



Lifecycle callbacks

```
class MyElementLifecycle extends HTMLElement {
   constructor() {
       // Called when an instance of the element is created or upgraded
       super(); // always call super() first in the ctor.
   }
   static get observedAttributes() {
       // Tells the element which attributes to observer for changes
       return [];
   }
   connectedCallback() {
       // Called every time the element is inserted into the DOM
   }
   disconnectedCallback() {
      // Called every time the element is removed from the DOM.
   }
   attributeChangedCallback(attrName, oldVal, newVal) {
       // Called when an attribute was added, removed, or updated
   }
   adoptedCallback() {
      // Called if the element has been moved into a new document
   }
}
```

my-counter custom element

```
class MyCounter extends HTMLElement {

    constructor() {
        super();
        this._counter = 0;
        this.attachShadow({ mode: 'open' });
    }

    connectedCallback() {
        this.render();
        this.display();
    }

    static get observedAttributes() { return ['counter'] }

    attributeChangedCallback(attr, oldVal, newVal) {
        if (oldVal !== newVal) {
            this[attr] = newVal;
        }
    }
```

my-counter custom element

```
    get counter() { return this._counter; }
    set counter(value) {
        if (value != this._counter) {
            this._counter = Number.parseInt(value);
            this.setAttribute('counter', value);
            this.display();
        }
    }

    increment() {
        this.counter = this.counter + 1;
        this.dispatchEvent(new CustomEvent(
          'increased', { detail: { counter: this.counter } }));
    }
```

my-counter-with-templates

```
let template = `
<style>
    ...
</style>
<div class="container">
   <div id="icon">
       <img src="../step-05/img/web-components.png">
   </div>
   <div id="value">
       0
   </div>
</div>
`;
```

my-counter-with-templates

```
render() {
    let templ = document.createElement('template');
    templ.innerHTML = template;

    this.shadowRoot.appendChild(templ.content.cloneNode(true));

    let button = this.shadowRoot.getElementById('icon');
    button.addEventListener('click', this.increment.bind(this));
}

display() {
    console.log(this.shadowRoot.getElementById('value'))
    this.shadowRoot.getElementById('value').innerHTML =
      `${this.counter}`;
}
```

`my-counter` custom element

Let's talk libraries

What's new?

Lot's of players, lot of evolutions

hybrids

LitElement

snuggsi ツ

SkateJS

smart
HTML ELEMENTS

stencil

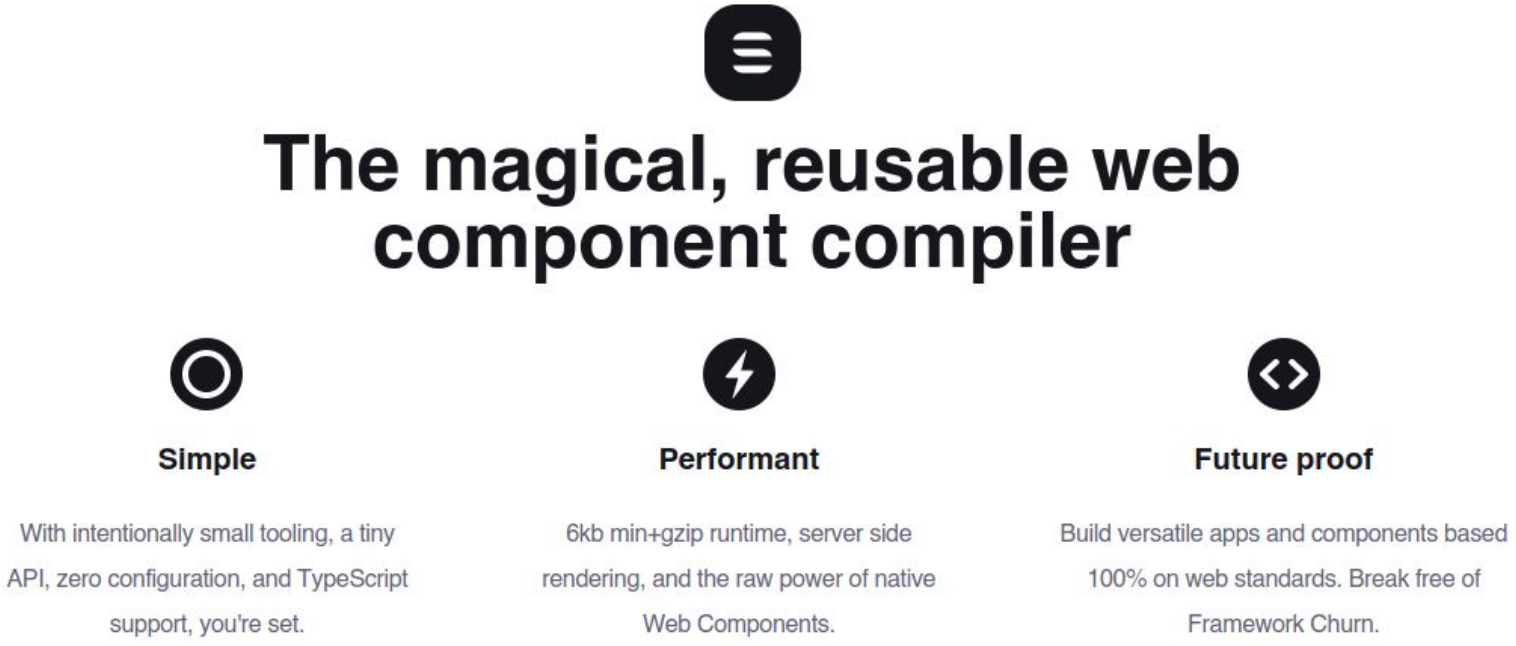
slim.js

It's JavaScript after all...

STENCIL

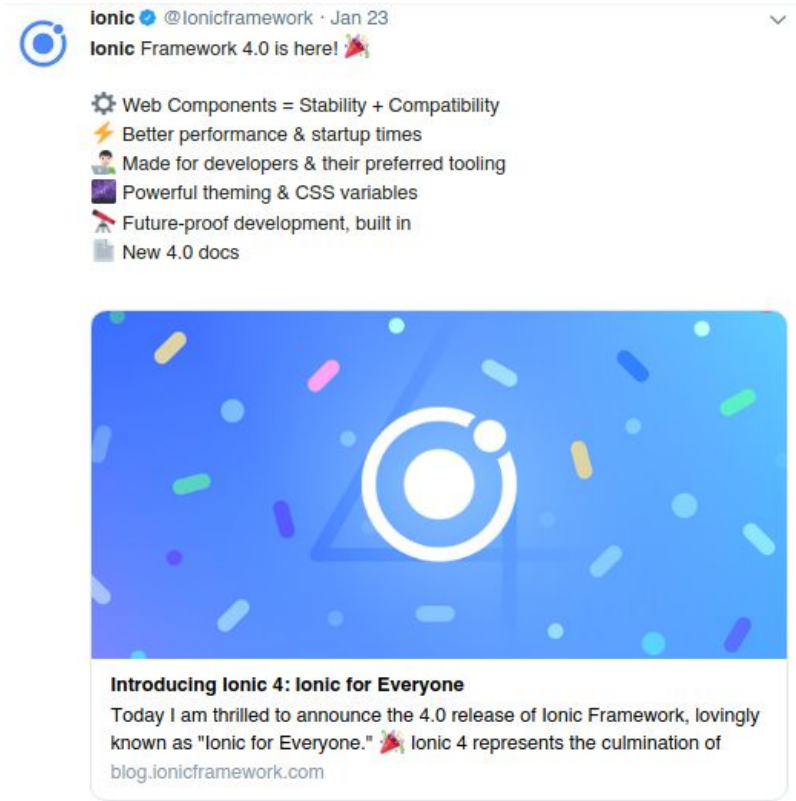
Powering the new Ionic 4!

Not another library



A Web Component compiler

Not a beta anymore



Ionic 4 released, powered by Stencil!

A build time tool

To generate standard web components

Fully featured

- Virtual DOM
- Async rendering
- Reactive data-binding
- TypeScript
- JSX

And the cherry on the cake

Server-Side Rendering

Hands on Stencil

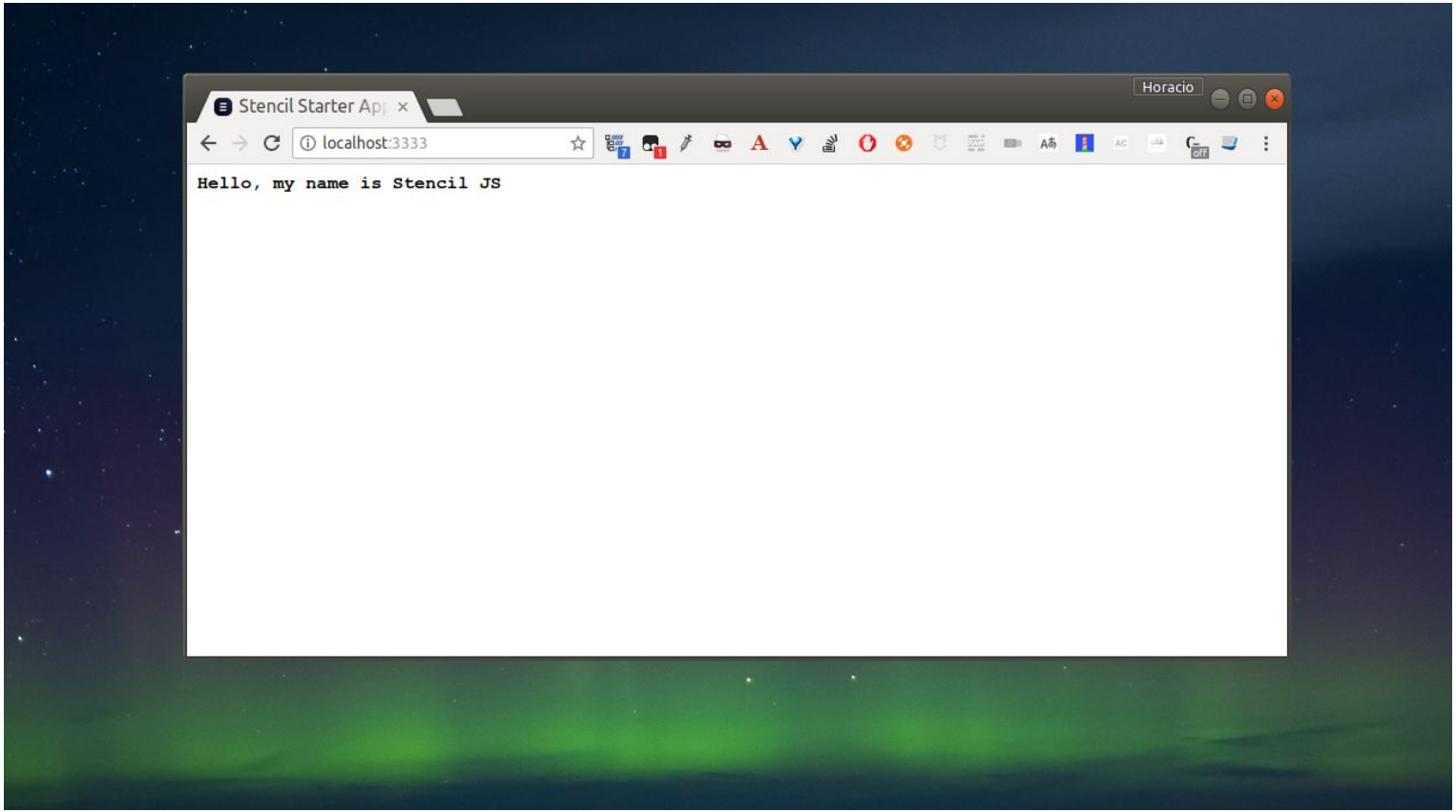
Clone the starter project

```
git clone https://github.com/ionic-team/stencil-app-starter my-app
cd my-app
git remote rm origin
npm install
```

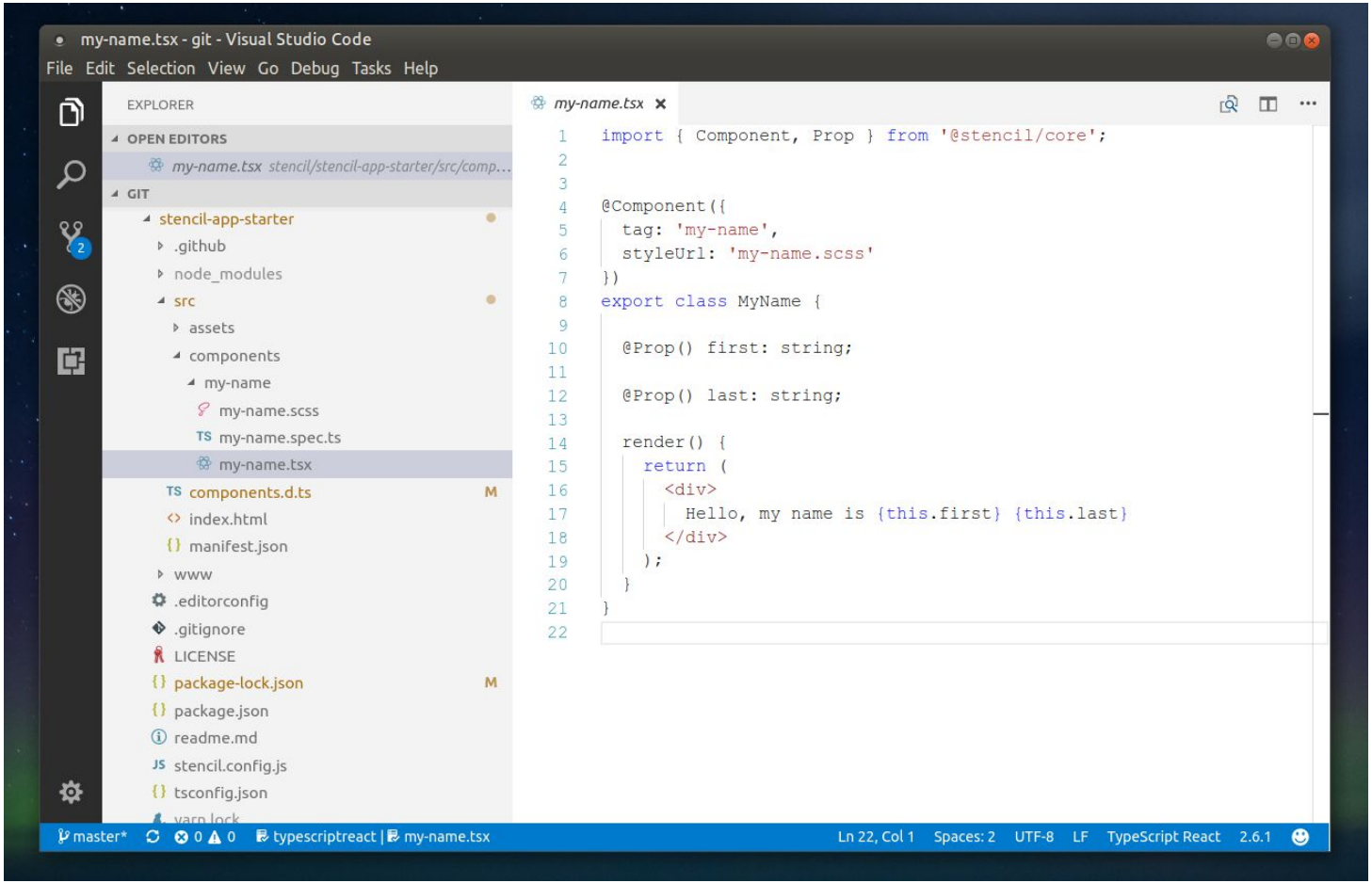
Start a live-reload server

```
npm start
```

Hands on Stencil



Hands on Stencil



Some concepts

```
render() {
  return (
    <div>Hello {this.name}</div>
  )
}
```

```
render() {
  return (
    <div>{this.name ? <p>Hello {this.name}</p> : <p>Hello World</p>}</div>
  );
}
```

JSX declarative template syntax

Some concepts

```
import { Component } from '@stencil/core';

@Component({
  tag: 'todo-list',
  styleUrl: 'todo-list.scss'
})
export class TodoList {
  @Prop() color: string;
  @Prop() favoriteNumber: number;
  @Prop() isSelected: boolean;
  @Prop() myHttpService: MyHttpService;
}
```

Decorators

Some concepts

```
import { Event, EventEmitter } from '@stencil/core';

...
export class TodoList {

  @Event() todoCompleted: EventEmitter;

  someAction(todo: Todo) {
    this.todoCompleted.emit(todo);
  }

  @Listen('todoCompleted')
  todoCompletedHandler(event: CustomEvent) {
    console.log('Received the custom todoCompleted event: ', event.detail);
  }
}
```

Events

Some concepts

```
@Component({
  tag: 'shadow-component',
  styleUrl: 'shadow-component.scss',
  shadow: true
})
export class ShadowComponent {

}
```

Optional Shadow DOM

Some concepts

stencil.config.js

```
exports.config = {
  namespace: 'myname',
  generateDistribution: true,
  generateWWW: false,
  ...
};
```

Generate distribution

Stencil

```
import { Component, Prop, PropWillChange, State, Event, EventEmitter } from '@stencil/core';

@Component({
 tag: 'stencil-counter',
 styleUrl: 'stencil-counter.scss',
 shadow: true
})
export class StencilCounter {
 @Prop() counter: number;
 @State() currentCount: number;
 @Event() currentCountChanged: EventEmitter;

 @Watch('counter')
 counterChanged(newValue: number) {
   this.currentCount = newValue;
 }

 componentWillLoad() {
   this.currentCount = this.counter;
 }

 increase() {
   this.currentCount++;
   this.currentCountChanged.emit({ counter: this.currentCount });
 }

 render() {
   return (
     <div class="container">
       <div class="button" onClick={() => this.increase()}> <img src="./img/stencil.png" /> </div>
       <div class="value" > {this.currentCount} </div>
     </div>
   );
 }
}
```

Polymer

Is the old player still alive?

Polymer has evolved in 2018

Image: © Nintendo

Polymer 3 is here!

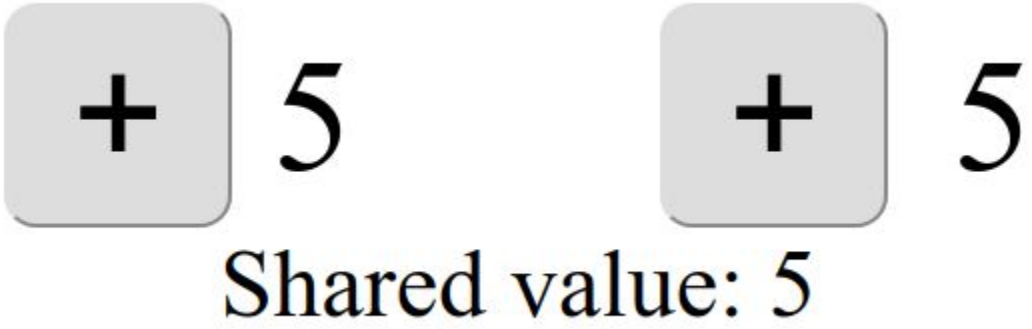
Classes, JavaScript modules...

```
import {html, PolymerElement} from
    '/node_modules/@polymer/polymer/polymer-element.js';

class MyPolymerCounter extends PolymerElement {
 static get template() {
    <style>
       :host {
         font-size: 5rem;
       }
       button {
         font-size: 5rem; border-radius: 1rem; padding: 0.5rem 2rem;
       }
     </style>
     <button on-click="increment">+</button>
     <span>[[counter]]</span>
   `;
 }
```

But it's still mostly syntactic sugar

```
 static get properties() {
   return {
     counter: { type: Number, reflectToAttribute:true, value: 0 }
   };
 }
 increment() {
   this.counter = Number.parseInt(this.counter) + 1;
   this.dispatchEvent(new CustomEvent('increased', {detail: {counter: this.counter}}));
 }
}

window.customElements.define('my-polymer-counter', MyPolymerCounter);
```

And they are still custom elements



100% interoperable

Interoperation pattern

```
<div class="container">
  <my-polymer-counter
      counter="[[value]]"
      on-increased="_onCounterChanged"></my-polymer-counter>
  <my-counter
      counter="[[value]]"
      on-increased="_onCounterChanged"></my-counter>
</div>
<div class="container">
  <div class="value">Shared value: [[value]]</div>
</div>
```

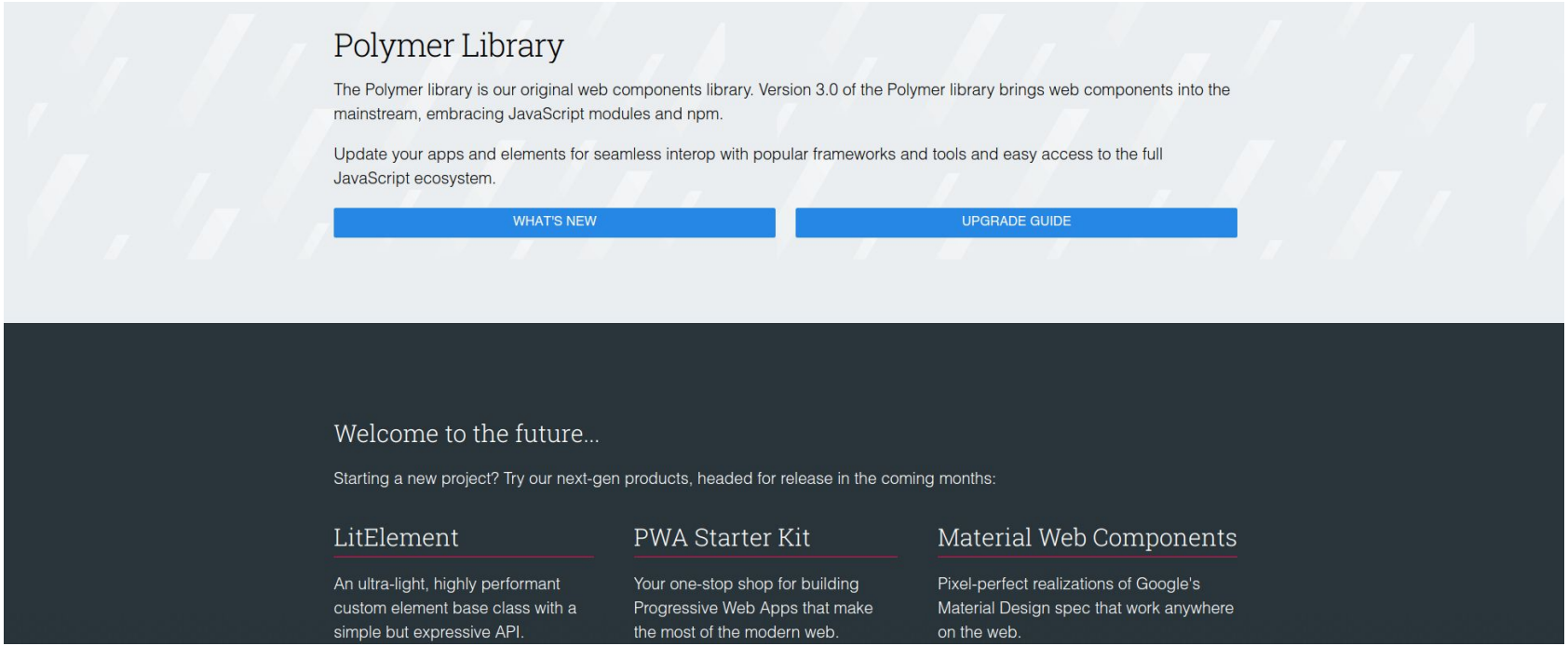
Attributes for data in

Events for data out

What's Polymer status today?

Well, how could I say… it's complicated

It seems it's going to be deprecated...



Technically yes… and that means good news!

Let's try to see clearer

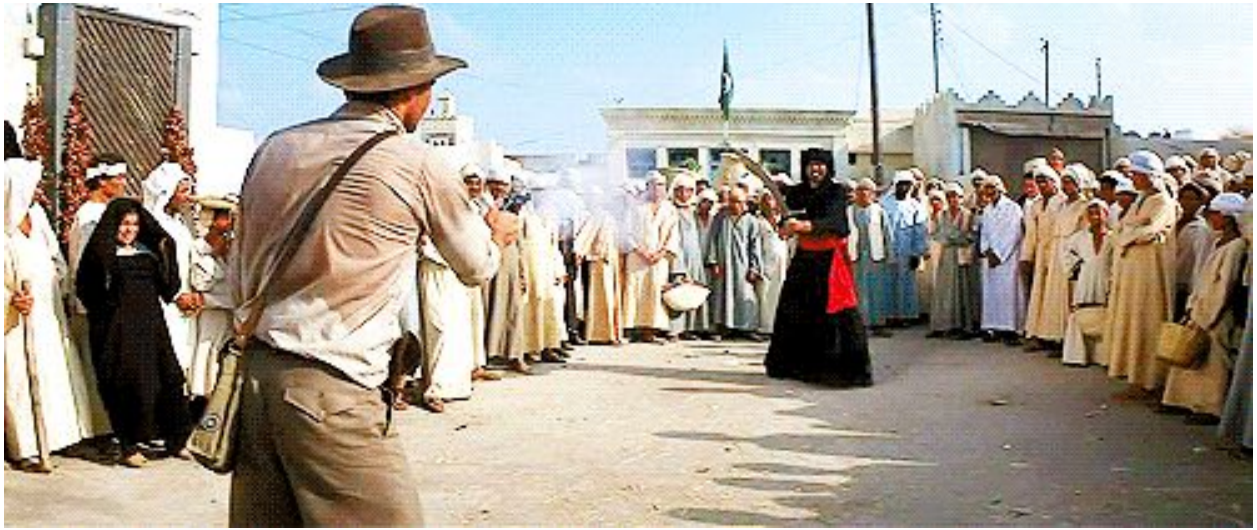
Let's dive into Polymer history...

A tool built for another paradigm

No web component support on browsers

No React, Angular or Vue innovations

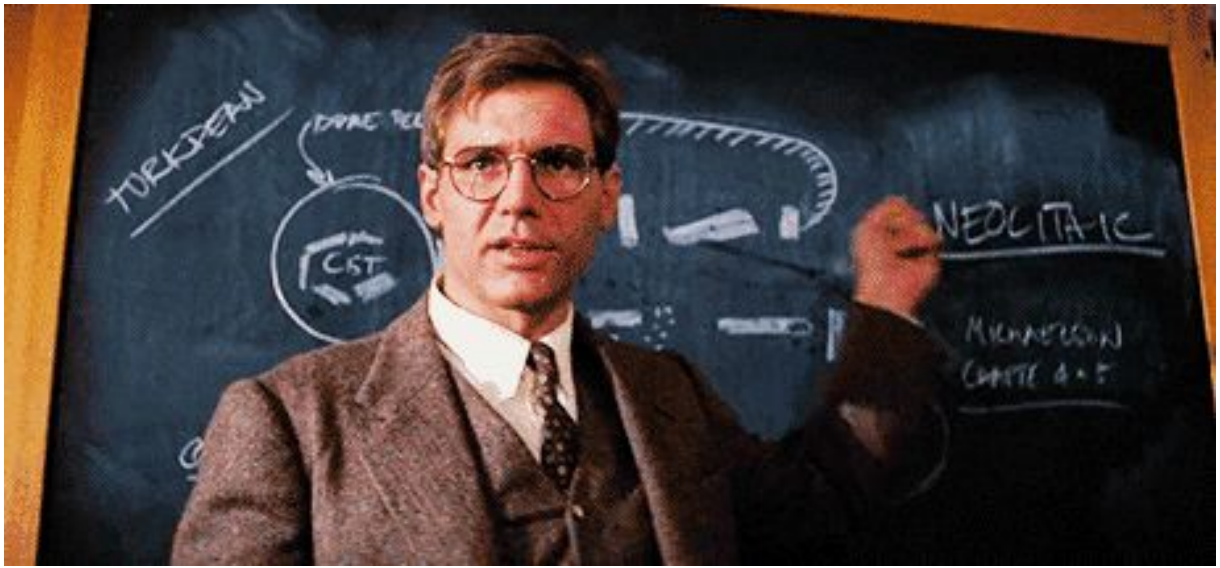
No so well suited for the current one



The current platform is way more powerful

The state of art has evolved

Let's learn from its lessons



The current platform is way more powerful

The state of art has evolved

And let it rest...

There will have no Polymer 4...

So Polymer as we know it is dead...

But the Polymer Project is indeed alive!

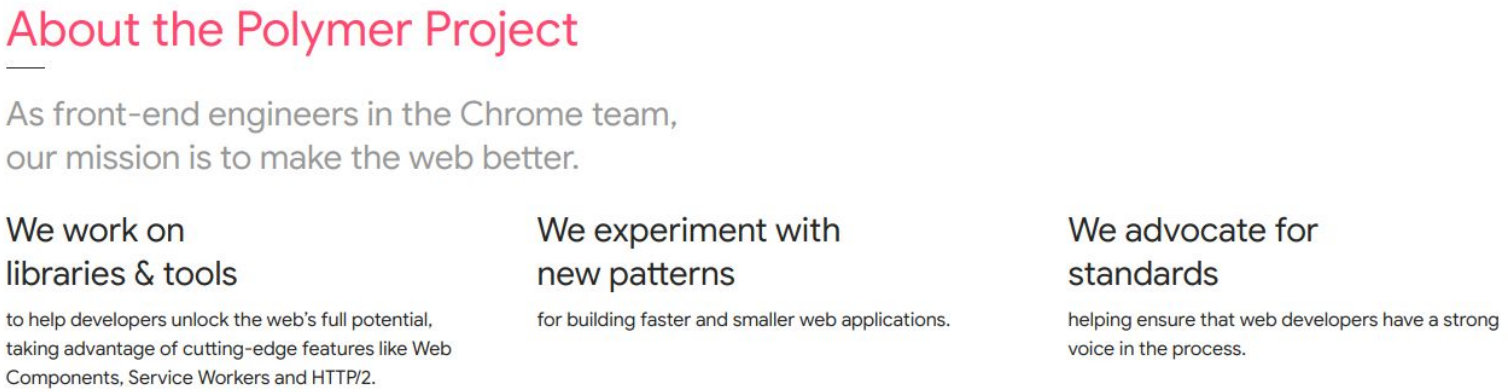
But I have invested so much on it!

What to do?

That's why web components are top

You can keep using all your Polymer components and create the new ones with a new library… And it simply works!

And without metaphors?



Polymer Project != Polymer library

Polymer Project well alive

Polymer library was only one library

Polymer

Is the old player still alive?

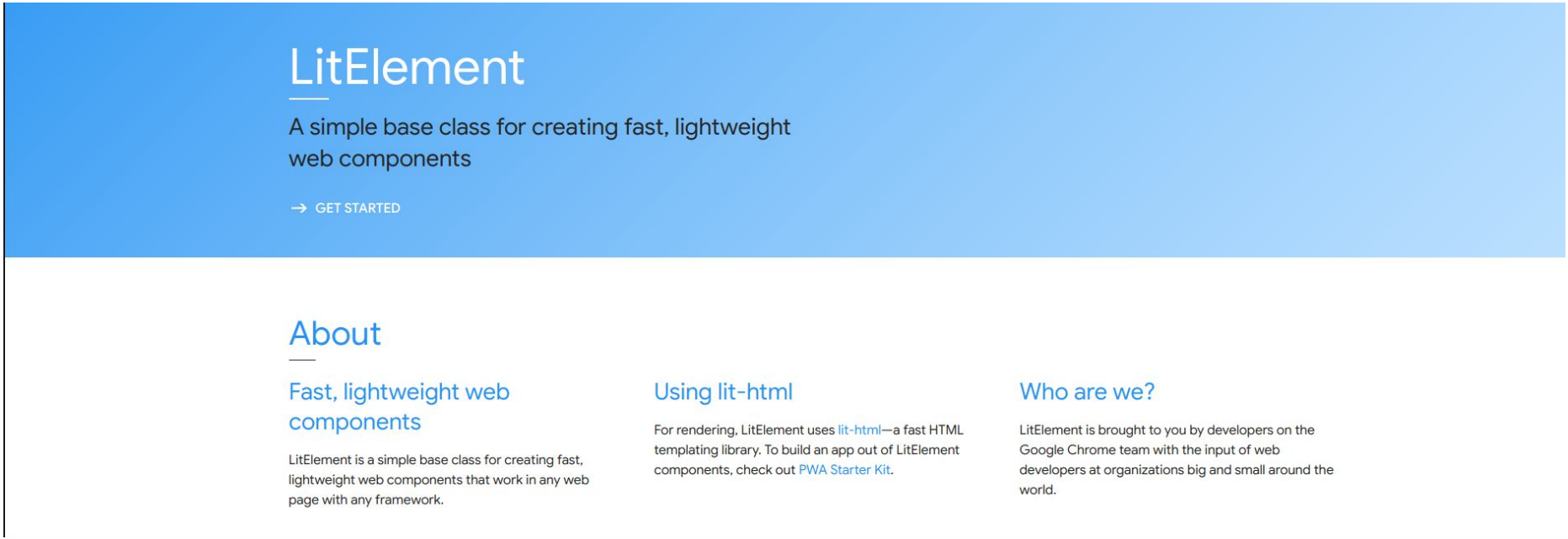
LitElement

New kid on the block

Born from the Polymer team

For the new web paradigm

Modern lightweight web components



For the new web paradigm

Based on lit-html

Next-generation HTML Templates in JavaScript

lit-html lets you write HTML templates in JavaScript, then efficiently render and *re-render* those templates together with data to create and update DOM:

```
import {html, render} from 'lit-html';

// A lit-html template uses the `html` template tag:
let sayHello = (name) => html`<h1>Hello ${name}</h1>`;

// It's rendered with the `render()` function:
render(sayHello('World'), document.body);

// And re-renders only update the data that changed, without
// VDOM diffing!
render(sayHello('Everyone'), document.body);
```

An efficient, expressive, extensible
HTML templating library for JavaScript

Do you know tagged templates?

```
function uppercaseExpression(strings, ...expressionValues) {
 var finalString = ''
 for ( let i = 0; i < strings.length; i++ ) {
   if (i > 0) {
     finalString += expressionValues[i - 1].toUpperCase()
   }
   finalString += strings[i]
 }
 return finalString
}

const  expressions = [ 'Tours', 'Touraine Tech',  'Thank you'];

console.log(
 uppercaseExpression`
   I am so happy to be in ${expressions[0]} for ${expressions[1]} again!
   ${expressions[2]},  ${expressions[1]}!
 `
)
```

Little known functionality of template literals

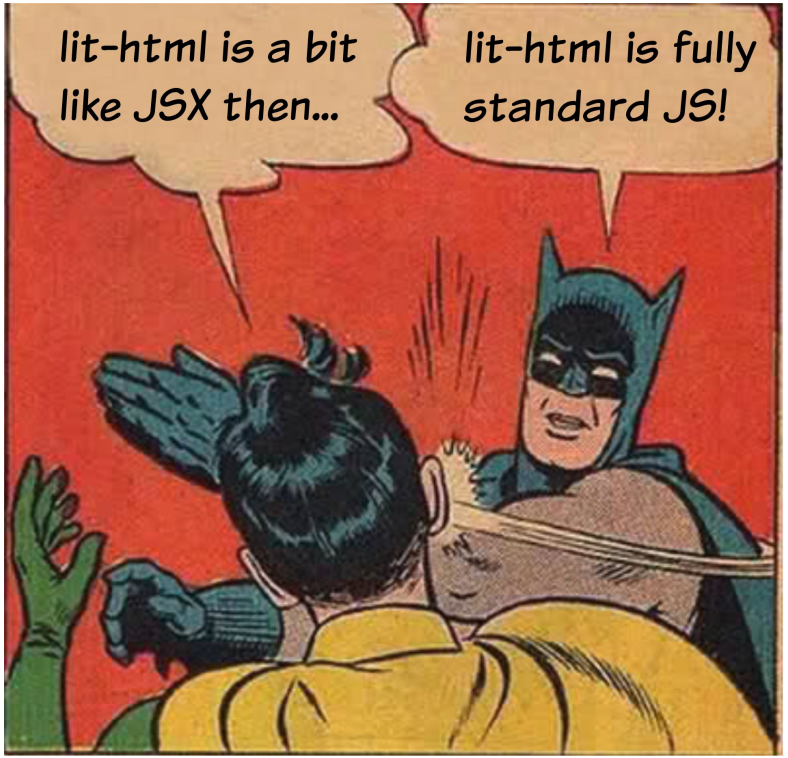
lit-html Templates

```
let myTemplate = (data) => html`
  <h1>${data.title}</h1>
  <p>${data.body}</p>
`;
```

Lazily rendered

Generates a TemplateResult

It's a bit like JSX, isn't it?



The good sides of JSX… but in the standard!

LitElement

```
import { LitElement, html } from 'lit-element';

// Create your custom component
class CustomGreeting extends LitElement {
 // Declare properties
 static get properties() {
   return {
     name: { type: String }
   };
 }
 // Initialize properties
 constructor() {
   super();
   this.name = 'World';
 }
 // Define a template
 render() {
   return html`<p>Hello, ${this.name}!</p>`;
 }
}
// Register the element with the browser
customElements.define('custom-greeting', CustomGreeting);
```

Lightweight web-components using lit-html

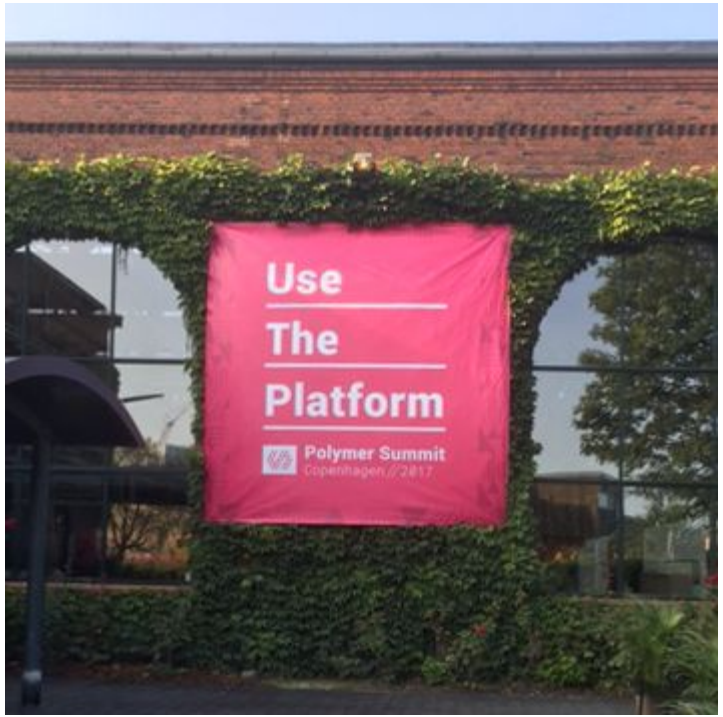
One more thing…*

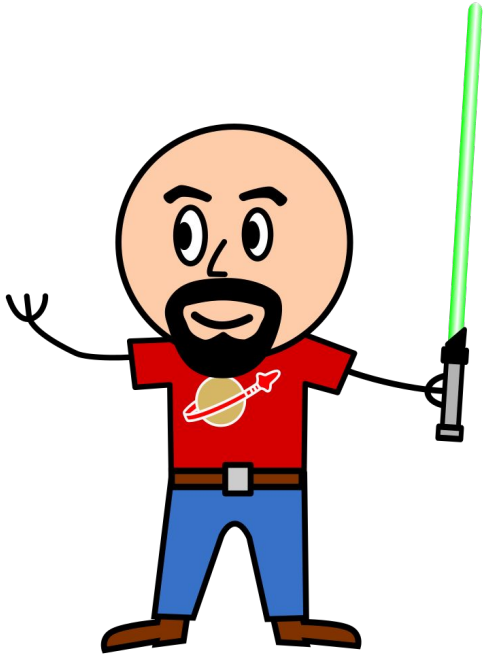
*Let's copy from the master

Polymer is not important

WebComponents ARE

Use the Platform, Luke...





WebComponents ARE native

Do you love your framework?

Oh yeah, we all do

Would you marry your framework?

Like until death…

How much does cost the divorce?

Do you remember when you dropped AngularJS for Angular?

Why recode everything again?

Reuse the bricks in your new framework

Lots of web components libraries

hybrids

LitElement

snuggsi ツ

SkateJS

smart
HTML ELEMENTS

stencil

slim.js

For different need and sensibilities

And some good news

Angular Elements

Vue Web Component Wrapper

Frameworks begin to understand it

So for your next app

Choose a framework, no problem…

But please, help your future self

Use Web Components!

Conclusion

That's all folks!