



# Web components en 2019, on en est où ?

Horacio Gonzalez  
[@LostInBrittany](#)

# Who are we?

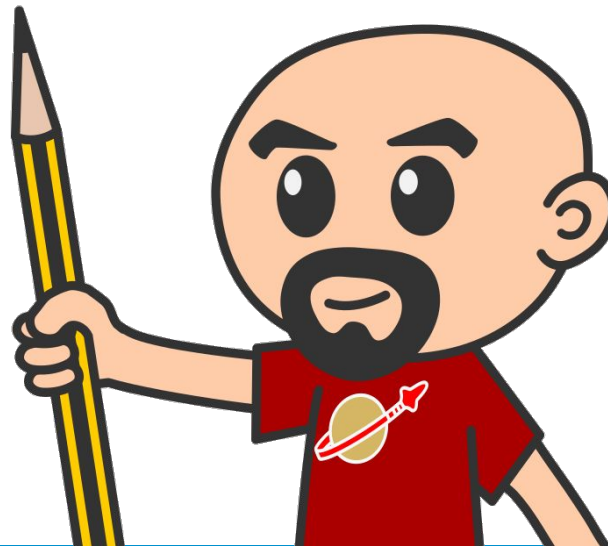
---

**Introducing myself and  
introducing OVH**

# Horacio Gonzalez

@LostInBrittany

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

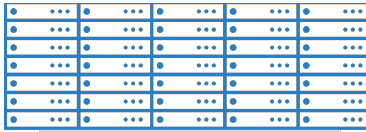


# OVH: A Global Leader on Cloud

200k Private cloud VMs running

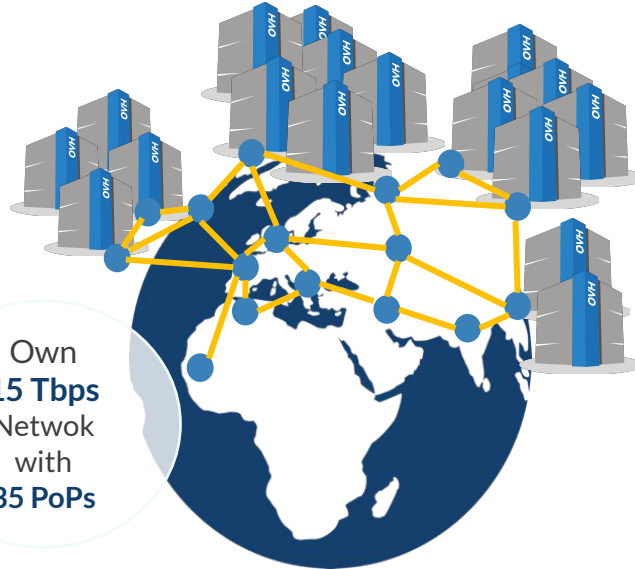


Dedicated IaaS Europe



Hosting capacity :  
**1.3M Physical Servers**

**360k**  
Servers already deployed



Own  
**15 Tbps**  
Network  
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



**1<sup>st</sup> European** Cloud Provider\*

**1<sup>st</sup> Hosting** provider in Europe

**1<sup>st</sup> 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

 <b>Cloud</b>	 <b>Mobile Hosting</b>	 <b>Web Hosting</b>	 <b>Telecom</b>
<hr/> <b>VPS</b> <b>Public Cloud</b> <b>Private Cloud</b> <b>Serveur dédié</b> <b>Cloud Desktop</b> <b>Hybrid Cloud</b>	<hr/> <b>Containers</b> <b>Compute</b> <b>Database</b> <b>Object Storage</b> <b>Securities</b> <b>Messaging</b>	<hr/> <b>Domain names</b> <b>Email</b> <b>CDN</b> <b>Web hosting</b> <b>MS Office</b> <b>MS solutions</b>	<hr/> <b>VoIP</b> <b>SMS/Fax</b> <b>Virtual desktop</b> <b>Cloud HubiC</b> <b>Over theBox</b>

# We want the code!

The screenshot shows the GitHub interface for the repository 'web-components-interop' by user 'LostInBrittany'. At the top, there are buttons for 'Unwatch' (1), 'Star' (1), and 'Fork' (2). Below this is a navigation bar with 'Code', 'Issues' (0), 'Pull requests' (0), 'Projects' (0), 'Wiki', 'Insights', and 'Settings'. The repository description is 'The git repository to support my 'A world outside Polymer' talk' with an 'Edit' button. A progress bar shows '11 commits', '1 branch', '0 releases', and '1 contributor'. Below the progress bar are buttons for 'Branch: master', 'New pull request', 'Create new file', 'Upload files', 'Find file', and 'Clone or download'. The file list shows a commit by 'LostInBrittany' updating 2018-12 files, with the latest commit '554da13' from 5 minutes ago. The files listed are:

File Name	Update Description	Update Time
node_modules	Updating 2018-12	5 minutes ago
step-01	Updating 2018-12	5 minutes ago
step-02	Updating 2018-12	5 minutes ago
step-03	Updating 2018-12	5 minutes ago
step-04	Updating 2018-12	5 minutes ago
step-05	Updating 2018-12	5 minutes ago
step-06	Updating Slim to last version	8 months ago
README.md	Updating	10 months ago
package-lock.json	Updating 2018-12	5 minutes ago
package.json	Updating 2018-12	5 minutes ago

<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>
```

Hello, world!



こんにちは、影の世界!

# Template



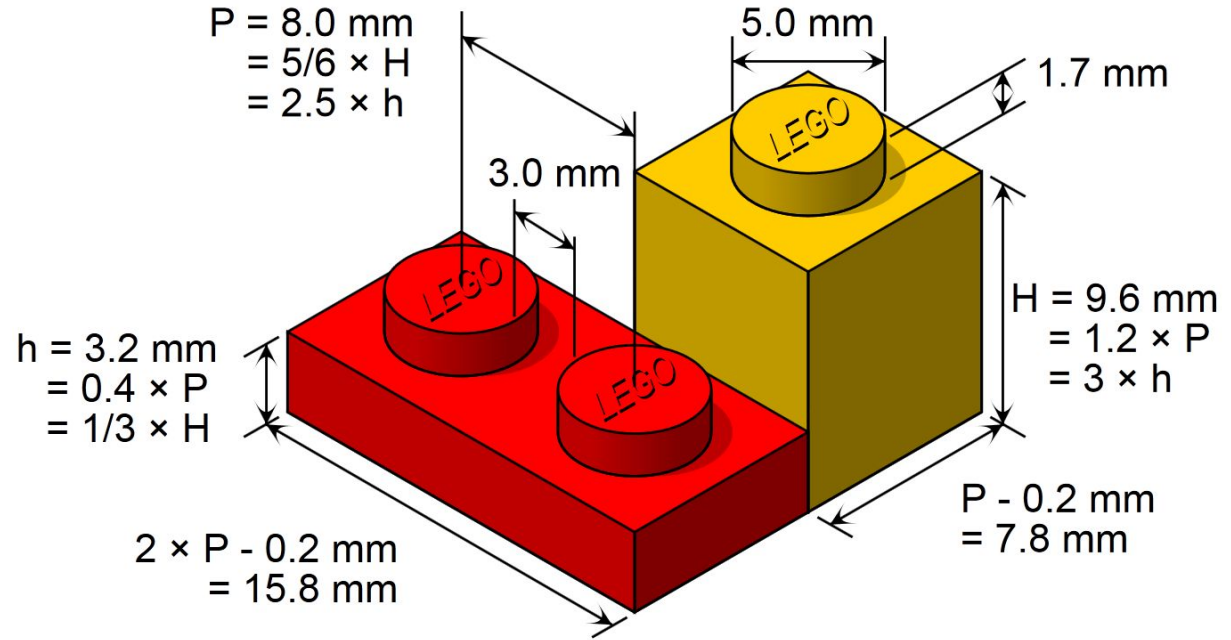
To have clonable document template

```
<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](http://bu.edu)

# Building a world brick by brick



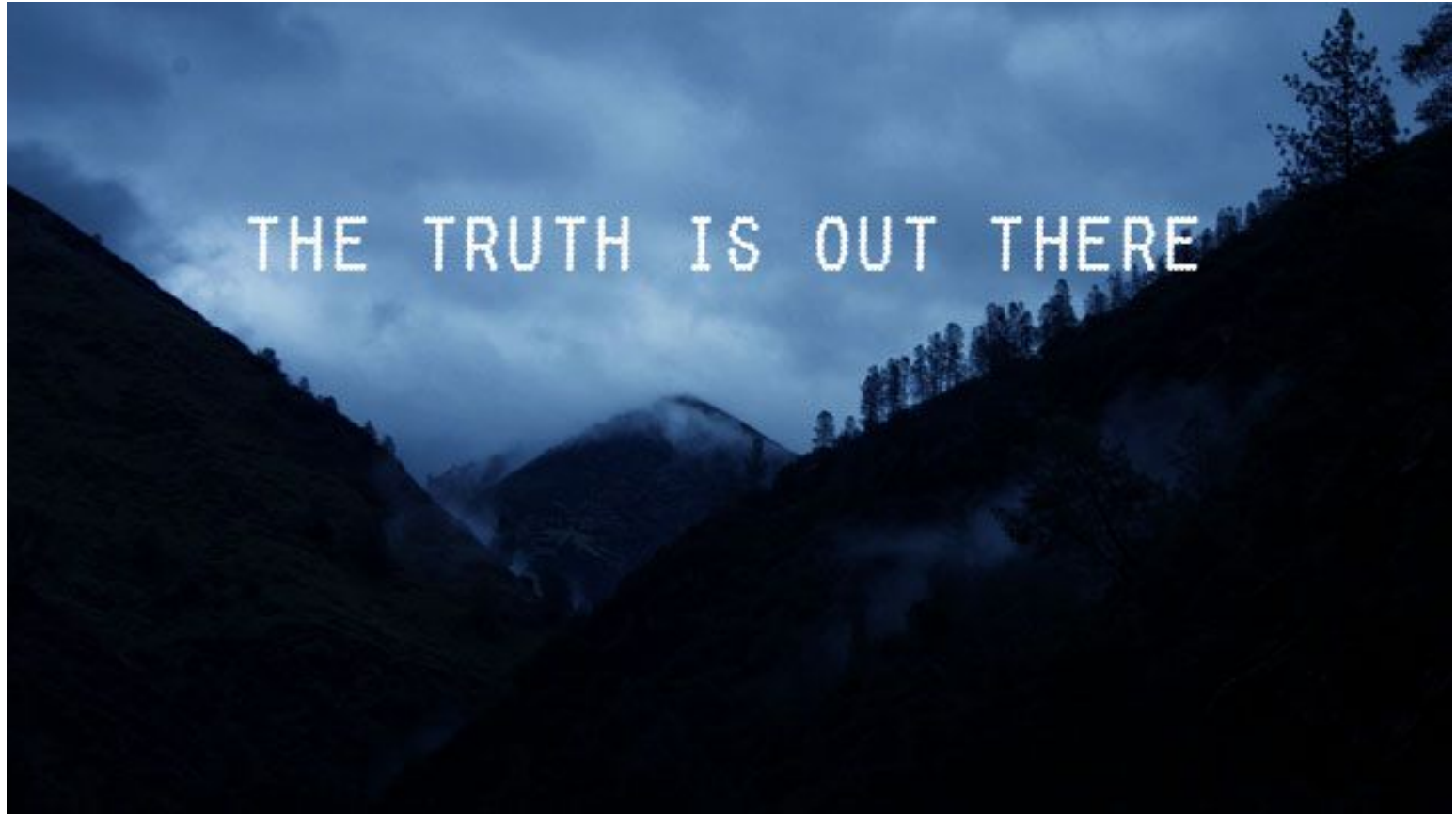
[BitRebels](#) [Brickset](#)

# 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



Max Lynch ✓

@maxlynch

Following

Looks like Twitter tweet embeds are now shipped as Web Components? This is how WC's will take over: as plumbing most won't even realize they are using

```
▼ <div id="posts" class="entry-content section-paragraph post">
  ▶ <p>...</p>
  ▼ <div style="margin: auto">
    ▼ <twitter-widget class="twitter-tweet twitter-tweet-rendered" id="twitter-
      display: block; transform: rotate(0deg); max-width: 100%; width: 500px;
      tweet-id="1072527151092678657"> == $0
    ▼ #shadow-root (open)
      <style type="text/css">.SandboxRoot { display: none; max-height: 100%;
      ▼ <div data-twitter-event-id="0" class="SandboxRoot env-bp-350" style="
        ▼ <div class="EmbeddedTweet EmbeddedTweet---cta js-clickToOpenTarget
          target="https://twitter.com/maxlynch/status/1072527151092678657" da
            "twitter-widget-0" lang="en" data-twitter-event-id="4">
              ▼ <div class="EmbeddedTweet-tweetContainer">
```

9:22 PM - 11 Dec 2018

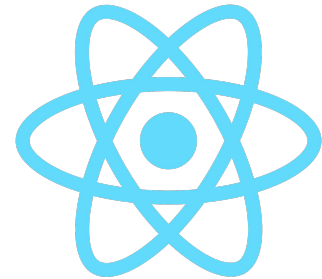
48 Retweets 163 Likes



9 48 163

## More than you can imagine

# The components architecture won



Components, components everywhere



# Web components ARE platform

↻ Yann Bertrand and 2 others Retweeted



**Justin Fagnani** @justinfagnani · 23 Oct 2018

🎉🎉🎉 **Firefox shipped Web Components!!!** 🎉🎉🎉

**Mozilla Hacks** @moz hacks

Shadow DOM, Custom Elements, a handy (variable) font editor panel in @FirefoxDevTools, reduced-motion CSS media queries — treat yourself to a spook-tacular brand new @Firefox 63, available today! 🎃  
[hacks.mozilla.org/2018/10/firefo...](https://hacks.mozilla.org/2018/10/firefo...)



## 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 = `

`${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 = `

${this.msg}</p>`;  
  }  
}  
  
customElements.define('my-element-with-shadowdom', MyElementWithShadowDom);

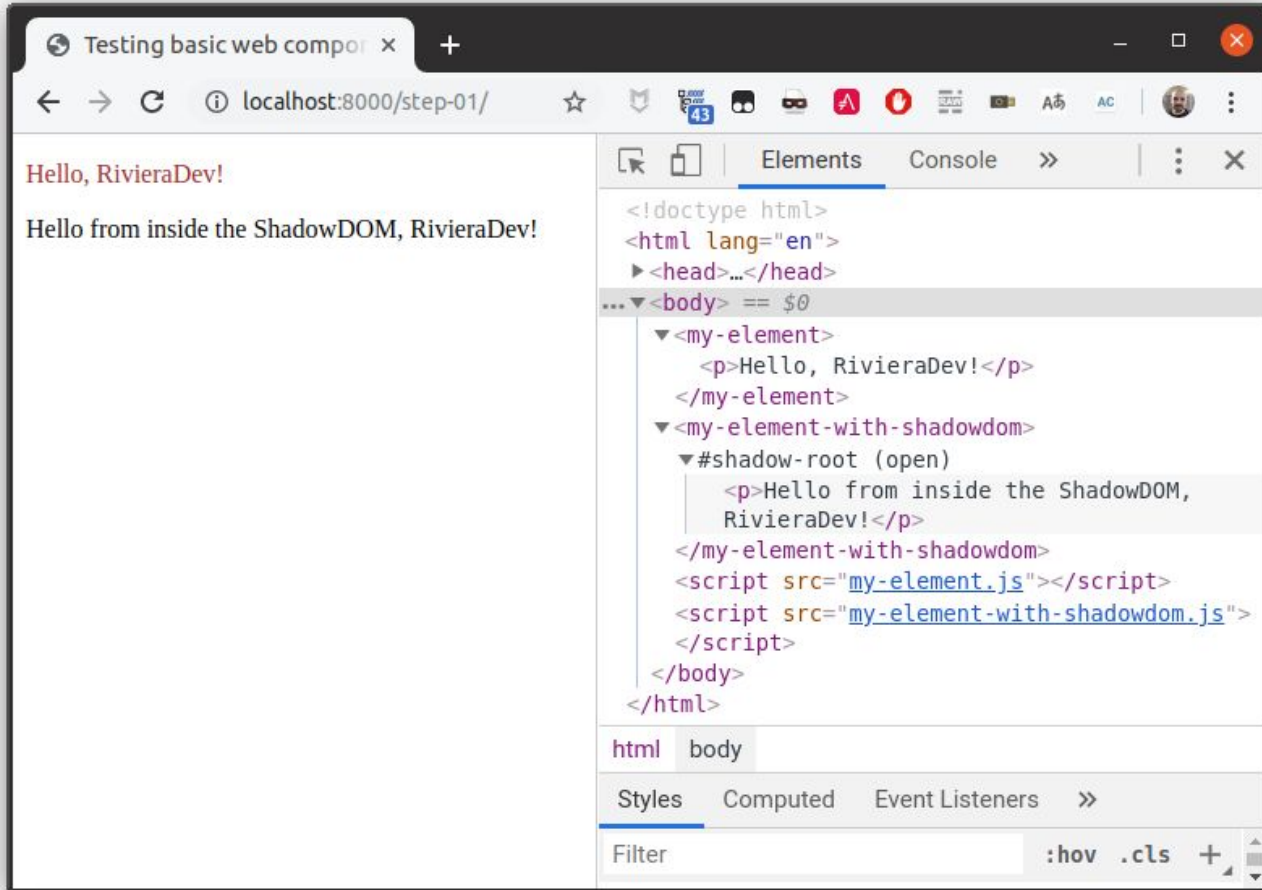

```

# 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 observe 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">  
        
    </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



slim.js



**stencil**

It's JavaScript after all...



**Powering the new Ionic 4!**



# Not another library



## The magical, reusable web component compiler



### Simple

With intentionally small tooling, a tiny API, zero configuration, and TypeScript support, you're set.



### Performant

6kb min+gzip runtime, server side rendering, and the raw power of native Web Components.

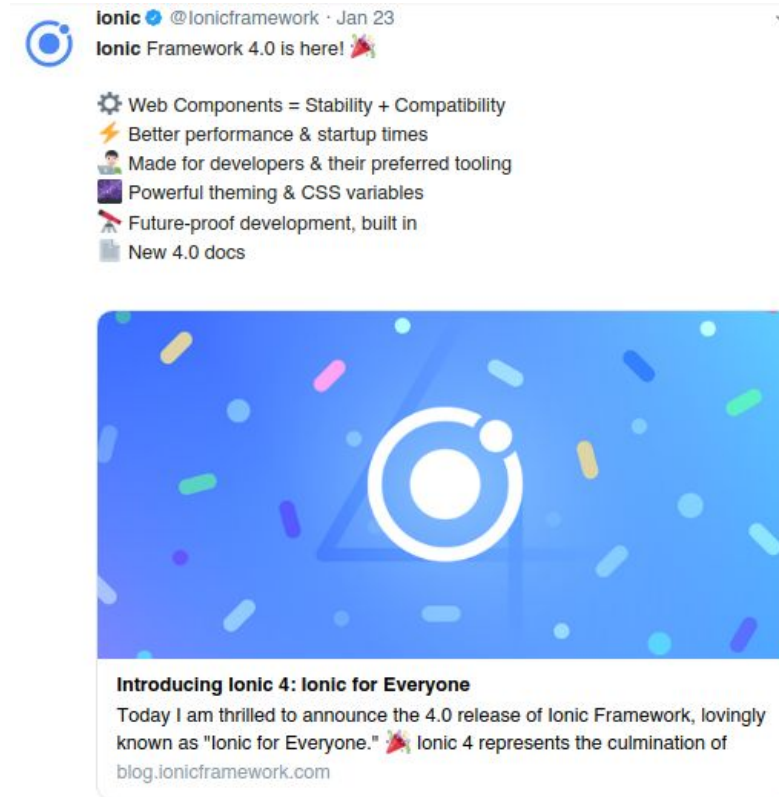


### Future proof

Build versatile apps and components based 100% on web standards. Break free of Framework Churn.

## 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

# SSR

## Server-Side Rendering

# Hands on Stencil

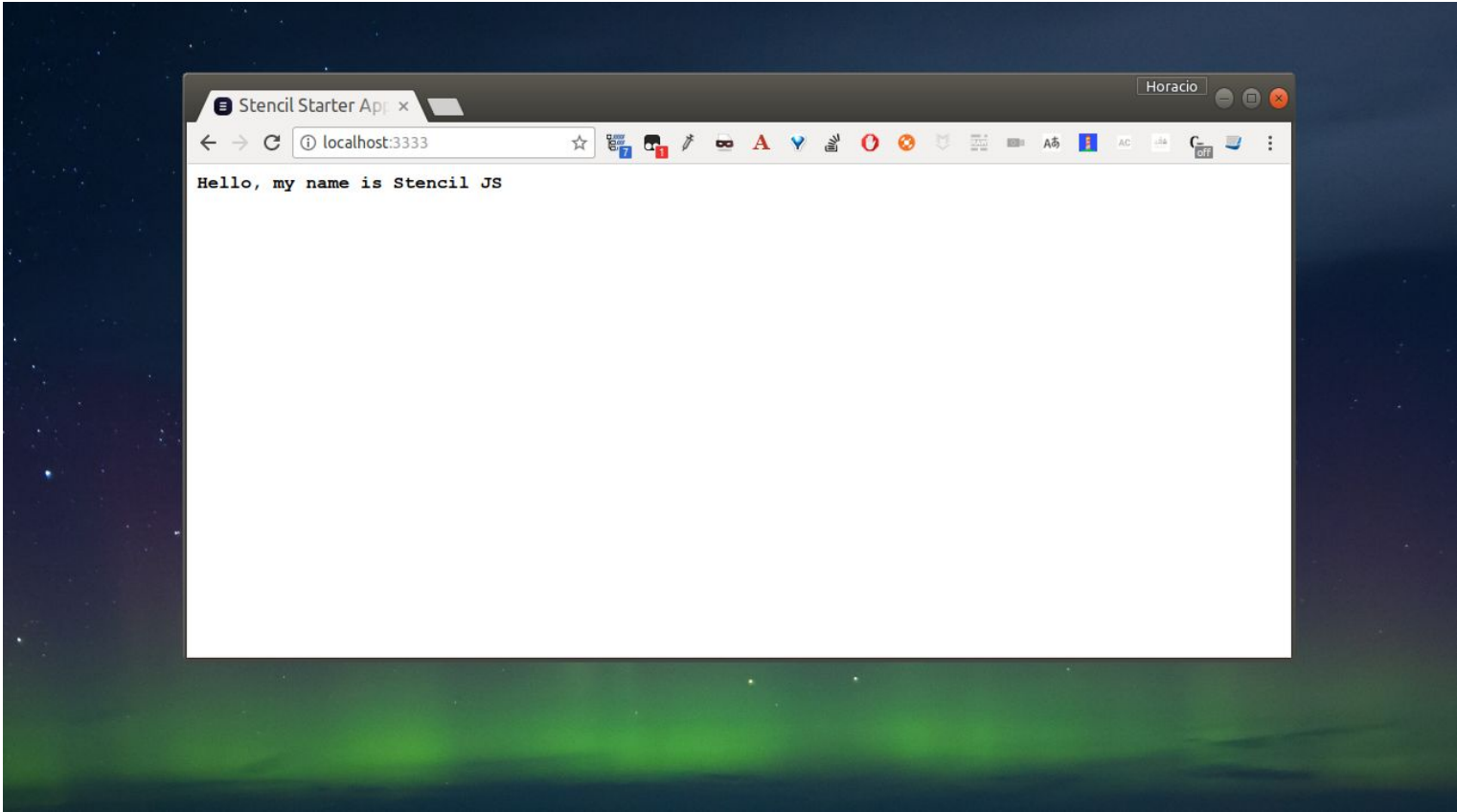
## Clone the starter project

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

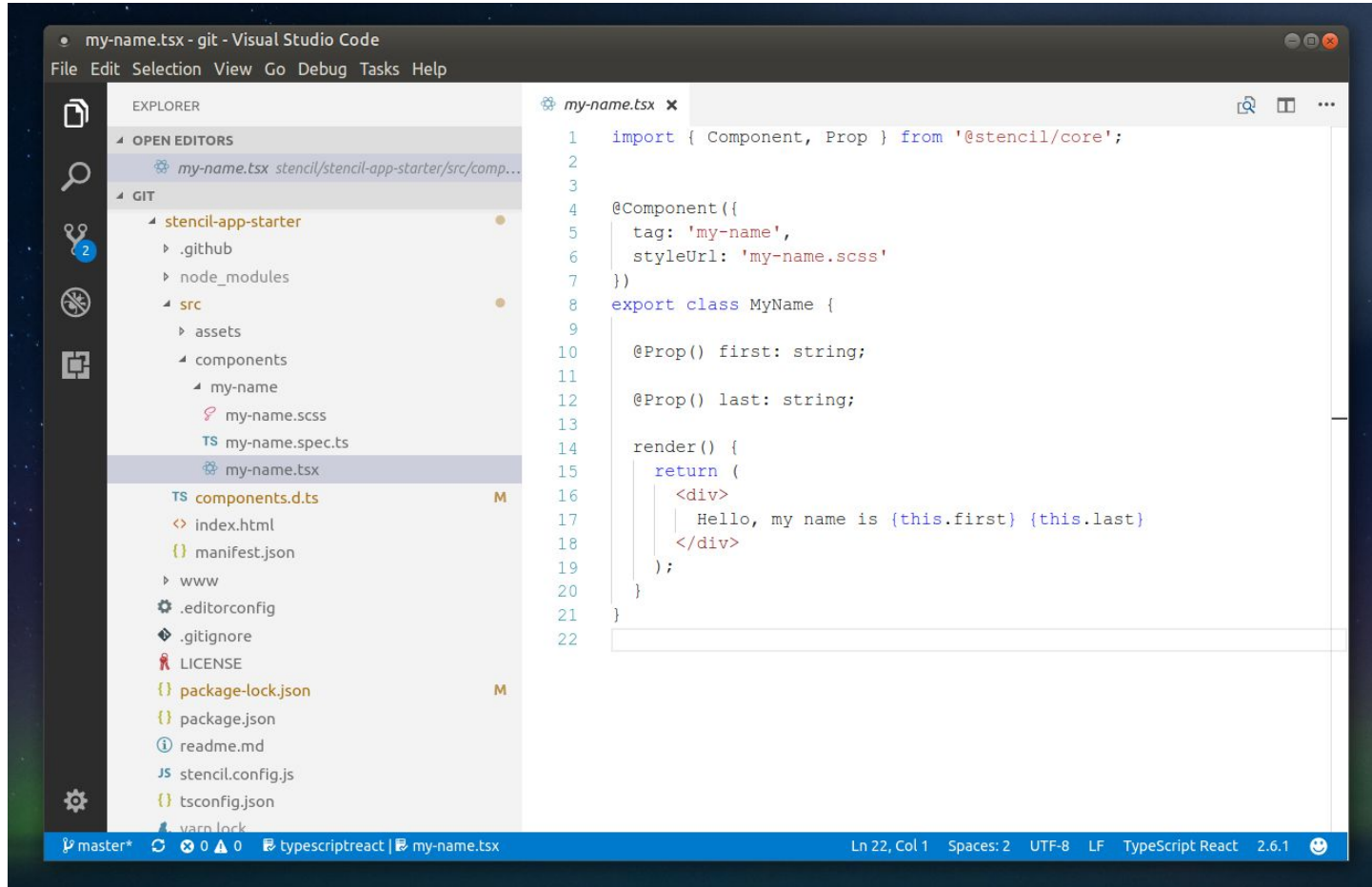
## Start a live-reload server

```
npm start
```

# Hands on Stencil



# Hands on Stencil



The screenshot shows the Visual Studio Code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project structure for a Stencil application, with the file `my-name.tsx` selected. The code editor displays the following TypeScript code:

```
1 import { Component, Prop } from '@stencil/core';
2
3
4 @Component({
5   tag: 'my-name',
6   styleUrls: 'my-name.scss'
7 })
8 export class MyName {
9
10  @Prop() first: string;
11
12  @Prop() last: string;
13
14  render() {
15    return (
16      <div>
17        Hello, my name is {this.first} {this.last}
18      </div>
19    );
20  }
21 }
22
```

The status bar at the bottom indicates the current position is Line 22, Column 1, with 2 spaces, UTF-8 encoding, LF line endings, and the TypeScript React extension version 2.6.1.



# 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',
  styleUrls: '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',
  styleUrls: ['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',
  styleUrls: '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()}>  </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



Shared value: 5

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?

Relationship Status:

Interested in:

Looking for:



A screenshot of a web form with three labels: 'Relationship Status:', 'Interested in:', and 'Looking for:'. To the right of these labels is a dropdown menu. The dropdown menu is currently open, showing a list of relationship status options. The options are: 'Single', 'In a Relationship', 'Engaged', 'Married', 'It's Complicated', 'In an Open Relationship', and 'Widowed'. The option 'It's Complicated' is highlighted with a blue background.

Well, how could I say... it's complicated

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

**Polymer Library**

The Polymer library is our original web components library. Version 3.0 of the Polymer library brings web components into the mainstream, embracing JavaScript modules and npm.

Update your apps and elements for seamless interop with popular frameworks and tools and easy access to the full JavaScript ecosystem.

[WHAT'S NEW](#) [UPGRADE GUIDE](#)

Welcome to the future...

Starting a new project? Try our next-gen products, headed for release in the coming months:

<b>LitElement</b>	<b>PWA Starter Kit</b>	<b>Material Web Components</b>
An ultra-light, highly performant custom element base class with a simple but expressive API.	Your one-stop shop for building Progressive Web Apps that make the most of the modern web.	Pixel-perfect realizations of Google's Material Design spec that work anywhere on the web.

Technically yes... and that means good

@LosePitany  
**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

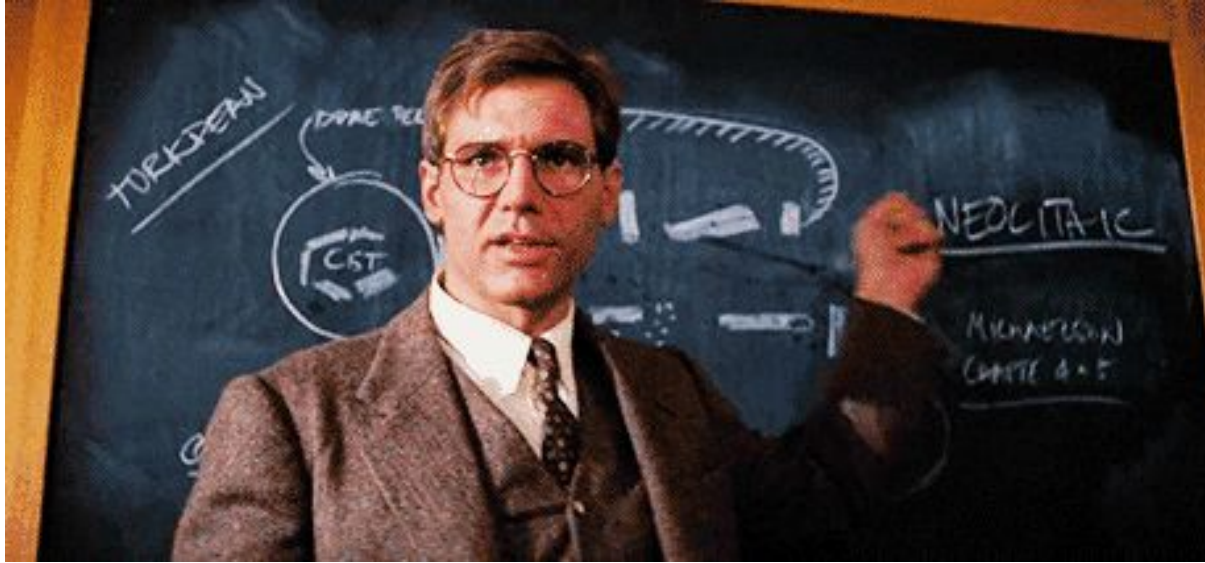


# No so well suited for the current one



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?

## About the Polymer Project

---

As front-end engineers in the Chrome team, our mission is to make the web better.

### We work on libraries & tools

to help developers unlock the web's full potential, taking advantage of cutting-edge features like Web Components, Service Workers and HTTP/2.

### We experiment with new patterns

for building faster and smaller web applications.

### We advocate for standards

helping ensure that web developers have a strong voice in the process.

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

## LitElement

A simple base class for creating fast, lightweight web components

→ [GET STARTED](#)

## About

### Fast, lightweight web components

LitElement is a simple base class for creating fast, lightweight web components that work in any web page with any framework.

### Using lit-html

For rendering, LitElement uses [lit-html](#)—a fast HTML templating library. To build an app out of LitElement components, check out [PWA Starter Kit](#).

### Who are we?

LitElement is brought to you by developers on the Google Chrome team with the input of web developers at organizations big and small around the world.

# 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]}!
`
)
```

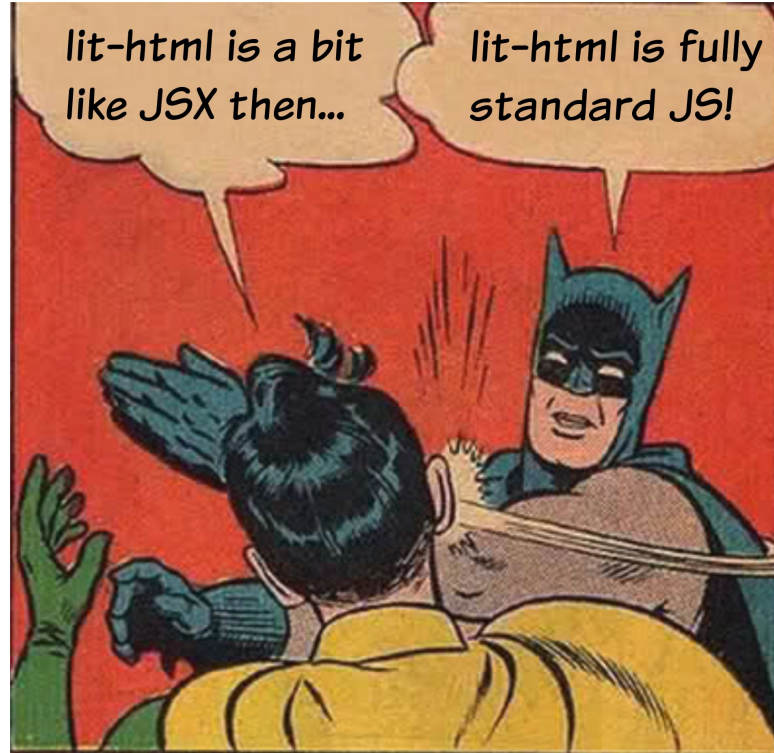
Little known functionality of template literals

# 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...\*

---



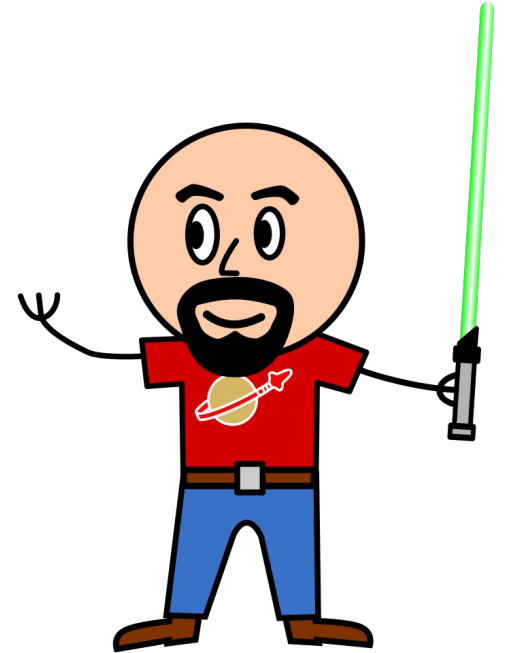
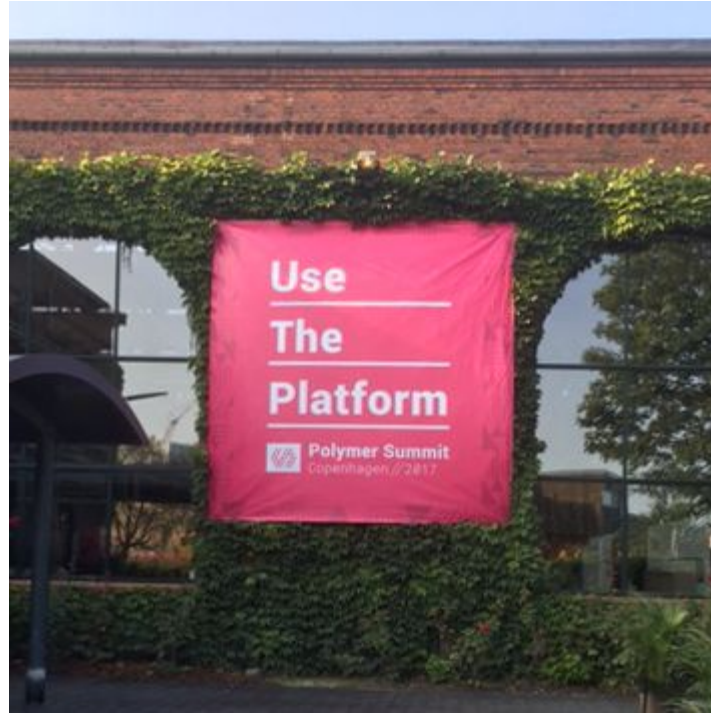
\*Let's copy from the master

# 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



slim.js



**stencil**

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!



