



# Web components en 2018

## On en est où ?

Horacio Gonzalez  
@LostInBrittany



# Who are we?

---

Introducing myself and  
introducing OVH

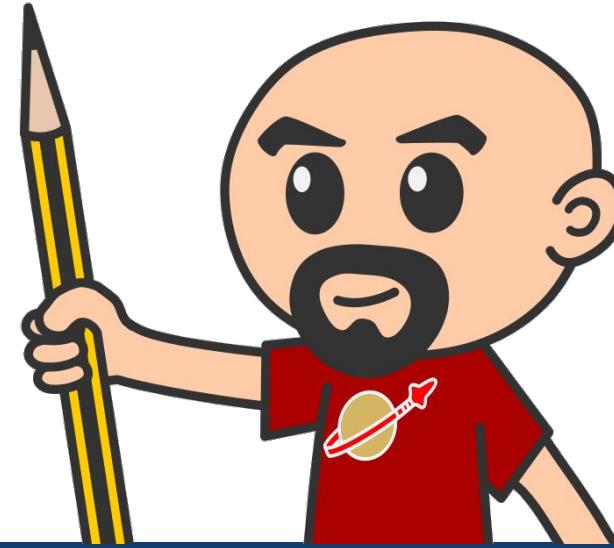


# Horacio Gonzalez



@LostInBrittany

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



# OVH : Key Figures



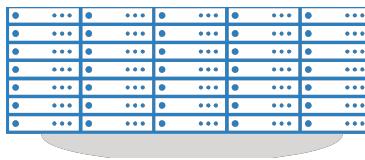
1.3M Customers worldwide in 138 Countries  
1.5 Billions euros investment over five years  
30 Datacenters (growing)  
350k Dedicated Servers  
200k Private cloud VMs running  
650k Public cloud Instances created in a month  
15TB bandwidth capacity  
35 Points of presence  
4TB Anti DDoS capacity  
Hosting capacity : 1.3M Physical Servers

+ 2 500 Employees in 19 countries  
18 Years of Innovation

# OVH: A Global Leader on Cloud

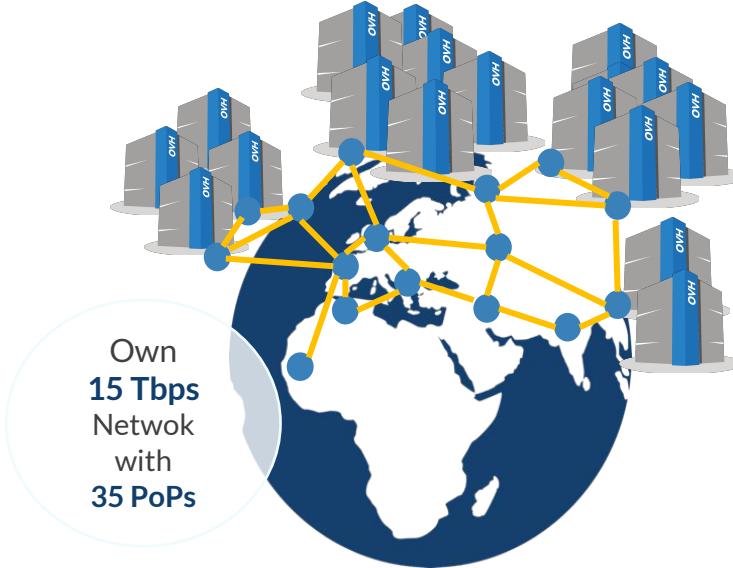


200k Private cloud  
VMs running



Hosting capacity :  
**1.3M** Physical  
Servers

**360k**  
Servers already  
deployed



2018  
**27** Datacenters  
↓  
2020  
**50** Datacenters

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



# OVH: Our solutions



## Cloud

- VPS
- Public Cloud
- Private Cloud
- Serveur dédié
- Cloud Desktop
- Hybrid Cloud



## Mobile Hosting

- Containers
- Compute
- Database
- Object Storage
- Securities
- Messaging



## Web Hosting

- Domain names
- Email
- CDN
- Web hosting
- MS Office
- MS solutions



## Telecom

- VoIP
- SMS/Fax
- Virtual desktop
- Cloud HubiC
- Over theBox

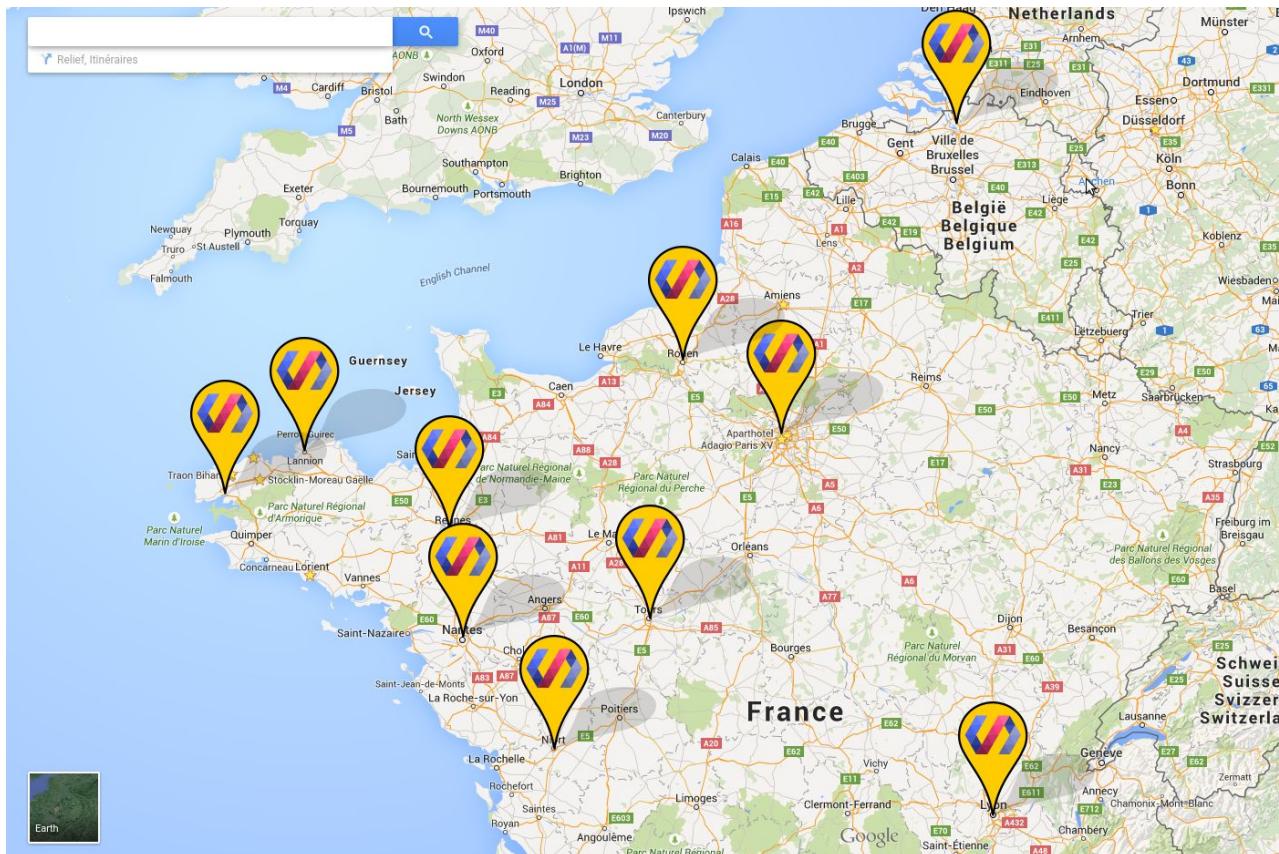
# Sometimes I feel a bit grumpy

---

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](#)

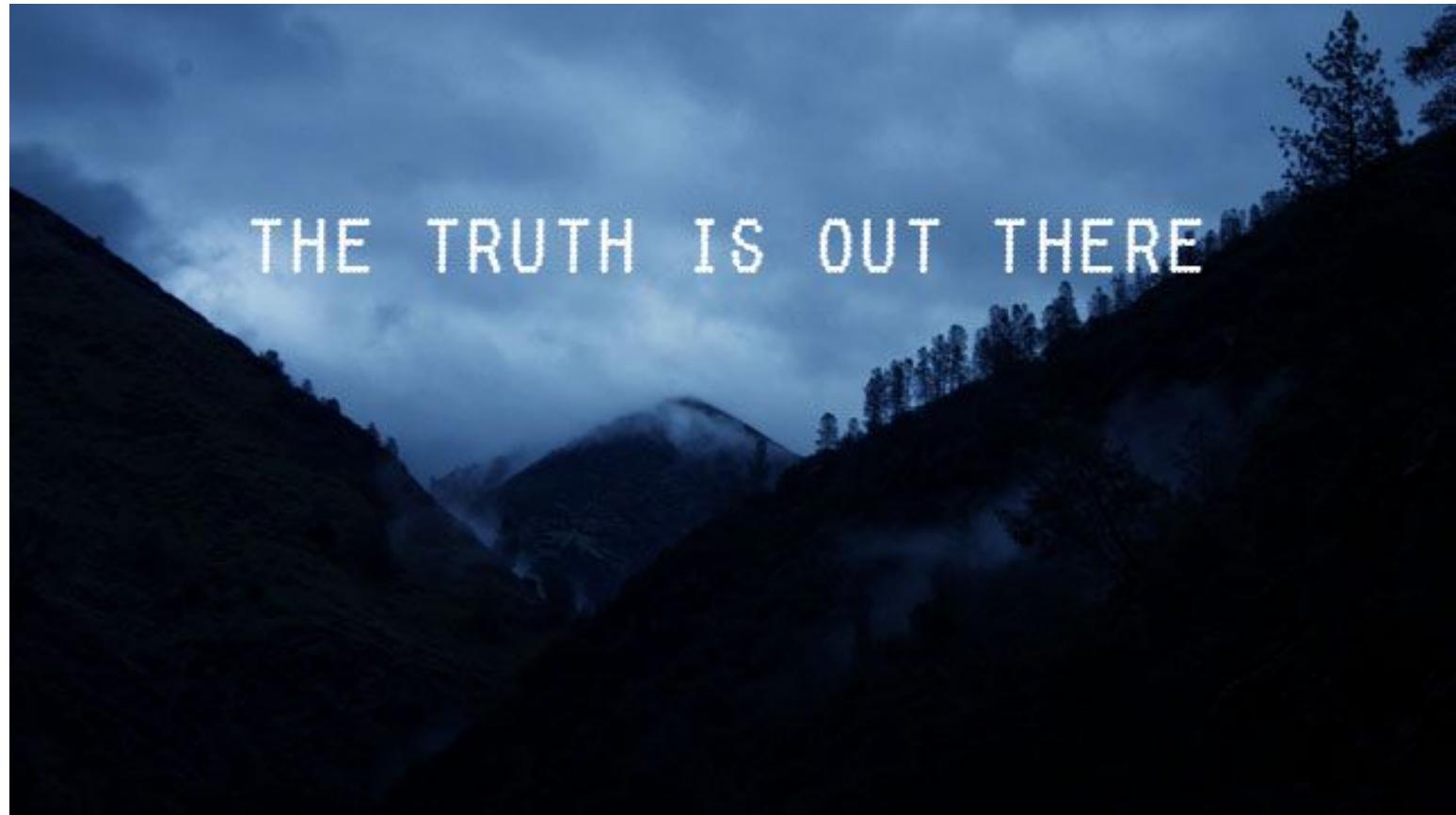


# Is the promise unfulfilled?

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



# Is it a conspiracy?



# Am I only a dreamer?



# Well, revolution IS there

 OVH  
Innovation for Freedom



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-tweet-1072527151092678657" data-tweet-id="1072527151092678657"> = $0
      <#shadow-root (open)>
        <style type="text/css">.SandboxRoot { display: none; max-height: 100%; }</style>
        <div data-twitter-event-id="0" class="SandboxRoot env-bp-350" style="border: 1px solid #ccc; border-radius: 4px; padding: 10px; width: fit-content; margin: auto">
          <div class="EmbeddedTweet EmbeddedTweet--cta js-clickToOpenTarget" data-target="https://twitter.com/maxlynch/status/1072527151092678657" data-tweet-id="1072527151092678657" data-twitter-widget-id="0" lang="en" data-twitter-event-id="4">
            <div class="EmbeddedTweet-tweetContainer">
```

9:22 PM - 11 Dec 2018

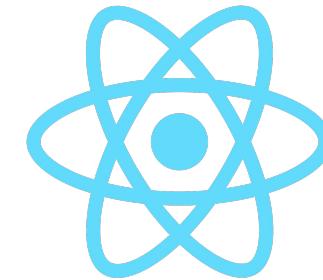
48 Retweets 163 Likes

9 48 163

More than you can imagine



# The components architecture won



Components, components everywhere

# Web Components

---



# We want the code!



[LostInBrittany / web-components-interop](#)

Code Issues 0 Pull requests 0 Projects 0 Wiki Insights Settings

The git repository to support my 'A world outside Polymer' talk Edit

Manage topics

11 commits 1 branch 0 releases 1 contributor

Branch: master New pull request Create new file Upload files Find file Clone or download

File	Message	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>



# 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, Bordeaux JUG!';  
    }  
  
    // 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

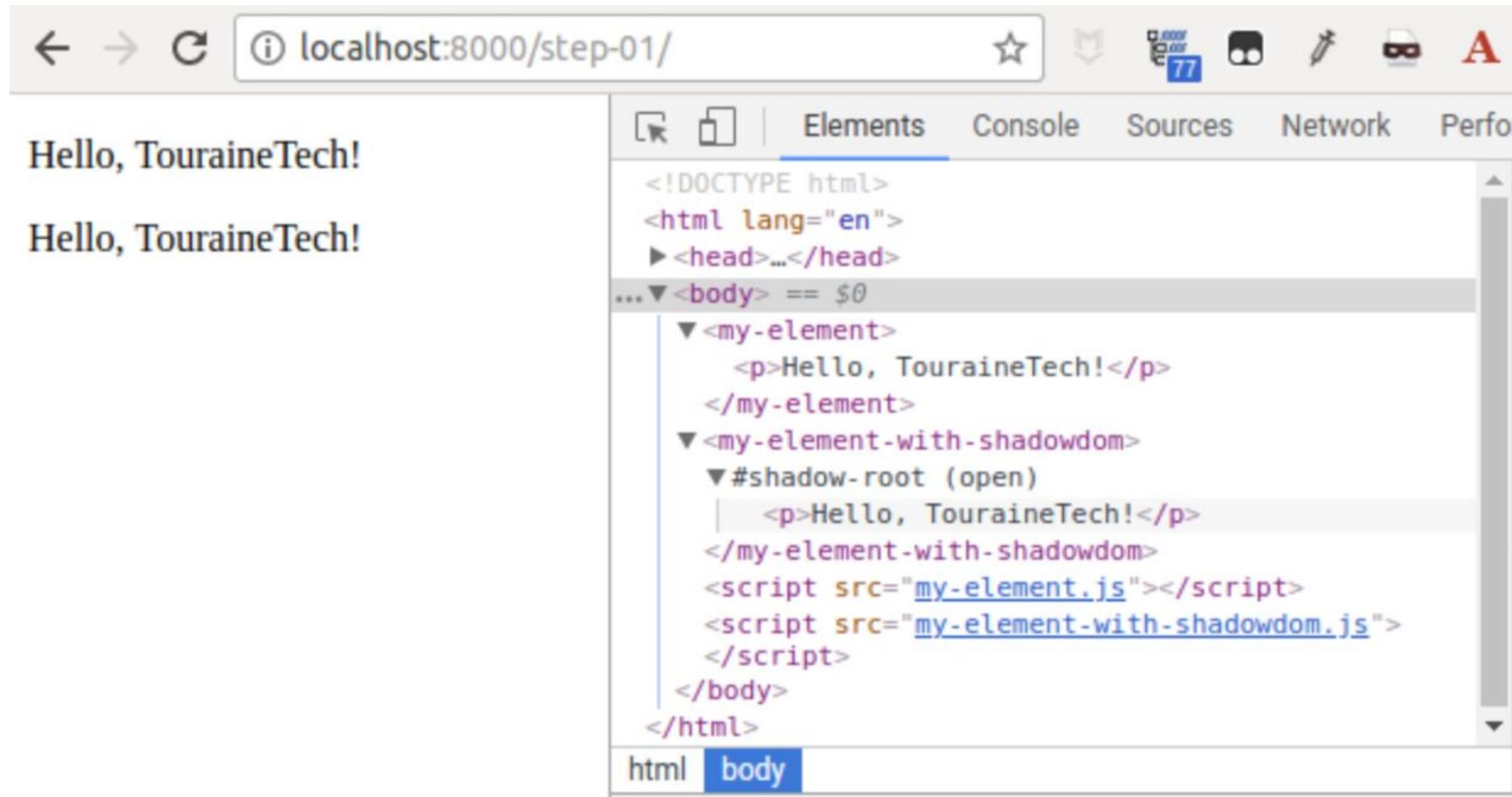


# Adding ShadowDOM

```
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, Bordeaux JUG!';  
        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);
```



# Adding ShadowDOM



The screenshot shows a browser window with the URL `localhost:8000/step-01/`. The page content displays two identical messages: "Hello, TouraineTech!". The browser's developer tools are open, specifically the Elements tab. The DOM tree is visible, showing the following structure:

```
<!DOCTYPE html>
<html lang="en">
  <head>...</head>
  ...<body> == $0
    <my-element>
      <p>Hello, TouraineTech!</p>
    </my-element>
    <my-element-with-shadowdom>
      <#shadow-root (open)>
        <p>Hello, TouraineTech!</p>
      </my-element-with-shadowdom>
      <script src="my-element.js"></script>
      <script src="my-element-with-shadowdom.js">
      </script>
    </body>
  </html>
```

The "my-element" node contains a single `p` element with the text "Hello, TouraineTech!". The "my-element-with-shadowdom" node contains a `#shadow-root (open)` node, which itself contains a `p` element with the same text. Both nodes are children of the `body` element, which is the root of the document.



# 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.
  }
  // Tells the element which attributes to observe for changes
  // This is a feature added by Custom Elements
  static get observedAttributes() {
    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() }  
    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 custom element

```
render() {  
  let button = document.createElement('button');  
  button.innerHTML = '+';  
  button.addEventListener('click', this.increment.bind(this));  
  this.shadowRoot.appendChild(button);  
  
  this.output = document.createElement('span');  
  this.shadowRoot.appendChild(this.output);  
  
  this.style.display = 'block';  
  this.style.fontSize = '5rem';  
  button.style.fontSize = '5rem';  
  button.style.borderRadius = '1rem';  
  button.style.padding = '0.5rem 2rem';  
  this.output.style.marginLeft = '2rem';  
}  
  
display() {  
  this.output.innerHTML = `${this.counter}`;  
}
```



# my-counter custom element



# Polymer

---

**Adding syntactic sugar to the standard**



# 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



5



5

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



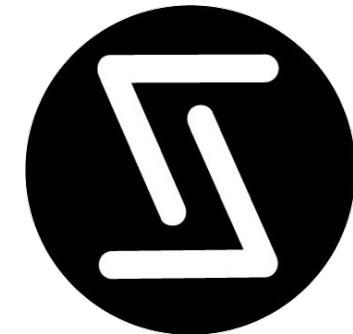
# To infinity and beyond!

---

**There is a world outside Polymer**



# Lots of web components libraries



For different need and sensibilities



# Lots of web components libraries



Angular Elements



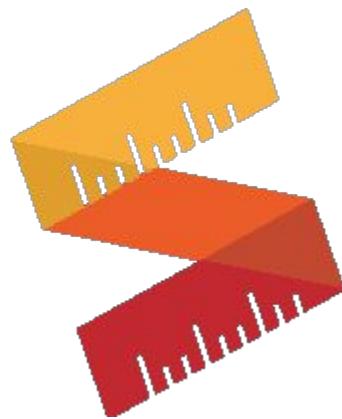
Vue Web Component  
Wrapper

And the frameworks work on it too!



# Slim.js

---





Getting started

Creating an element

Lifecycle

Accessing children

Conditional rendering

Shadow-DOM

Data binding

Repeating elements

Plugins

Extending Slim & Directives

Submit an issue

Submit a documentation issue

## Getting started with Slim.js

Slim.js is a lightweight web component authoring library that provides extended capabilities for components (such as data binding) using es6 native class inheritance. This library is focused on providing developers the ability to write robust web components quickly and without the hassle of unnecessary dependencies or an overhead of a framework.

### Installation

```
npm install slim-js
yarn add slim-js
bower install slimjs
```

### Web-Components Spec. & Polyfills

Slim.js is based on the browser's native DOM API's web-components spec. For browsers that do not support this natively, a polyfill is required.

You can load the polyfill directly from the html file

```
<script src="https://cdnjs.cloudflare.com/ajax/libs/webcomponentsjs/1.0.17/webcomponents-lite.js"></script>
```



- Lightweight web component library
- Extended capabilities for components
  - data binding
- Using es6 native classes and modules
- Without Shadow DOM by default

Like a lighter and lesser-featured Polymer



# Slim.js

```
import {Slim} from '/node_modules/slim-js/Slim.js'

let template = `
<style>
  ...
</style>
<div class="container">
  <div class="button" slim-id="button" click='increment'>
    
  </div>
  <div class="value" bind> {{toString(counter)}} </div>
</div>`;

Slim.tag('my-slim-counter', template,

class extends Slim {
  ...
});
```



# Slim.js

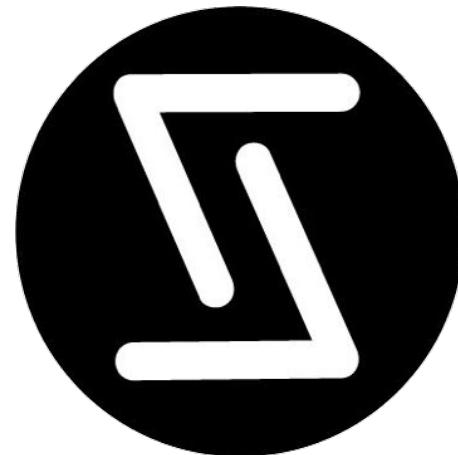


```
class extends Slim {
    // native API
    static get observedAttributes () {
        return ['counter'];
    }
    // bind attributes to properties
    get autoBoundAttributes() {
        return ['counter'];
    }
    increment() {
        this.counter = this.counter + 1;
        this.dispatchEvent(new CustomEvent('increased', {detail: {counter: this.counter}}));
    }
    toString(value) {
        return value.toString();
    }
});
```



# Skatejs

---





## SkateJS

Effortless custom elements for modern view libraries.

Code

HTML

Result

```
// @jsx h

import { props, withComponent } from 'skatejs';
import withPreact from '@skatejs/renderer-preact';
import { h } from 'preact';

class WithPreact extends withComponent(withPreact()) {
  static get props() {
    return {
      name: props.string
    };
  }
  render({ name }) {
    return <span>Hello, {name}!</span>;
  }
}
```



# Skatejs



- Lightweight web component library
- Abstracts away attribute / property semantics
- Very very fast
- Can use many renderers
  - Basic innerHTML (default)
  - preact
  - lit-html



# Skatejs



```
import { props, withComponent } from '/node_modules/skatejs/dist/es/index.js';
import withLitHtml from '/node_modules/@skatejs/renderer-lit-html/dist/es/index.js';
import { html } from '/node_modules/lit-html/lit-html.js';

class MySkateCounter extends withComponent(withLitHtml()){

    static get props () {
        return {
            // By declaring the property an attribute, we can now pass an initial value
            // for the count as part of the HTML.
            counter: props.number({ attribute: true })
        };
    }
}
```



# Skatejs

```
render({ counter }) {
  return html`
<div class="container">
  <button class="button" @click="${() => this.increment()}">
    
  </button>
  <div class="value">${counter}</div>
</div>`;
}
increment() {
  console.log('sdgws', this.counter)
  this.counter = Number.parseInt(this.counter) + 1;
  this.dispatchEvent(new CustomEvent('increased', {detail: {counter: this.counter}}));
}
}
window.customElements.define('my-skate-counter', MySkateCounter);
```





# STENCIL

---

## A new breed of Web Components



# Next generation Ionic



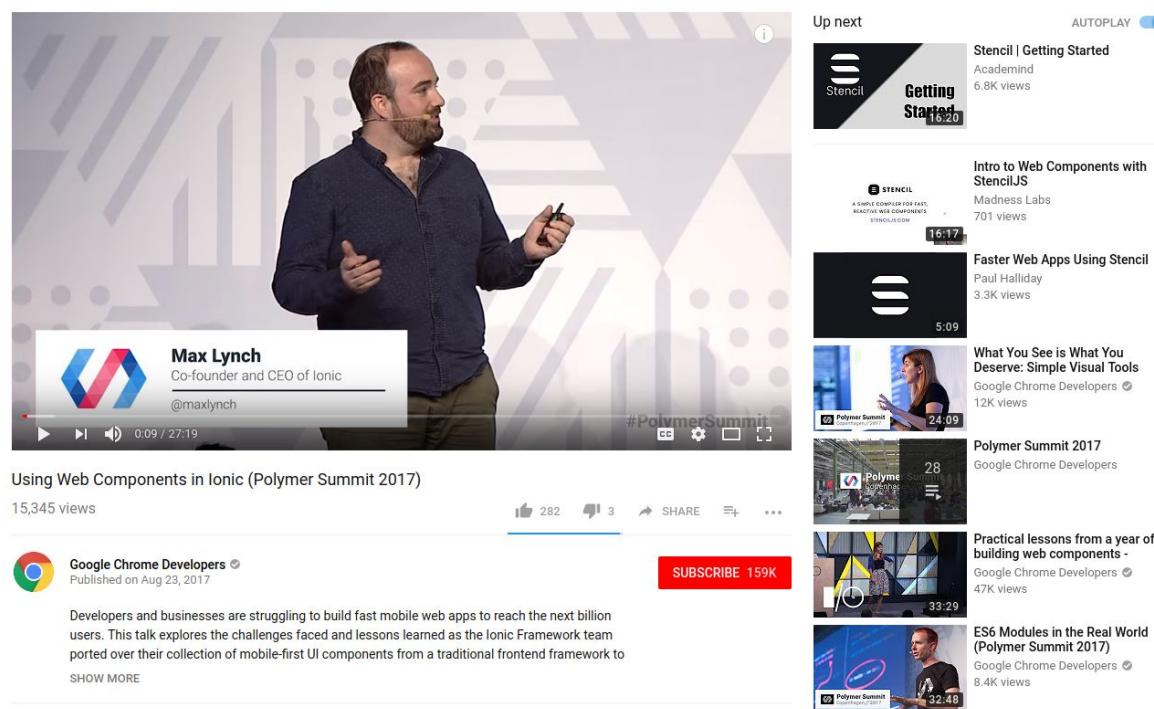
Ionic 4 will be fully based on web  
components

using a new toolkit: Stencil

#webcomponents

@LostInBrittany

# New kid on the block



Max Lynch  
Co-founder and CEO of Ionic  
@maxlynch

Using Web Components in Ionic (Polymer Summit 2017)

15,345 views

Google Chrome Developers Published on Aug 23, 2017

Developers and businesses are struggling to build fast mobile web apps to reach the next billion users. This talk explores the challenges faced and lessons learned as the Ionic Framework team ported over their collection of mobile-first UI components from a traditional frontend framework to SHOW MORE

282 likes 3 dislikes SHARE

SUBSCRIBE 159K

Up next

AUTOPLAY

Stencil | Getting Started  
Academind 6.8K views

Intro to Web Components with StencilJS  
Madness Labs 701 views

Faster Web Apps Using Stencil  
Paul Halliday 3.3K views

What You See is What You Deserve: Simple Visual Tools  
Google Chrome Developers 12K views

Polymer Summit 2017  
Google Chrome Developers

Practical lessons from a year of building web components -  
Google Chrome Developers 47K views

ES6 Modules in the Real World (Polymer Summit 2017)  
Google Chrome Developers 8.4K views

## Announced during Polymer Summit



# 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



# 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

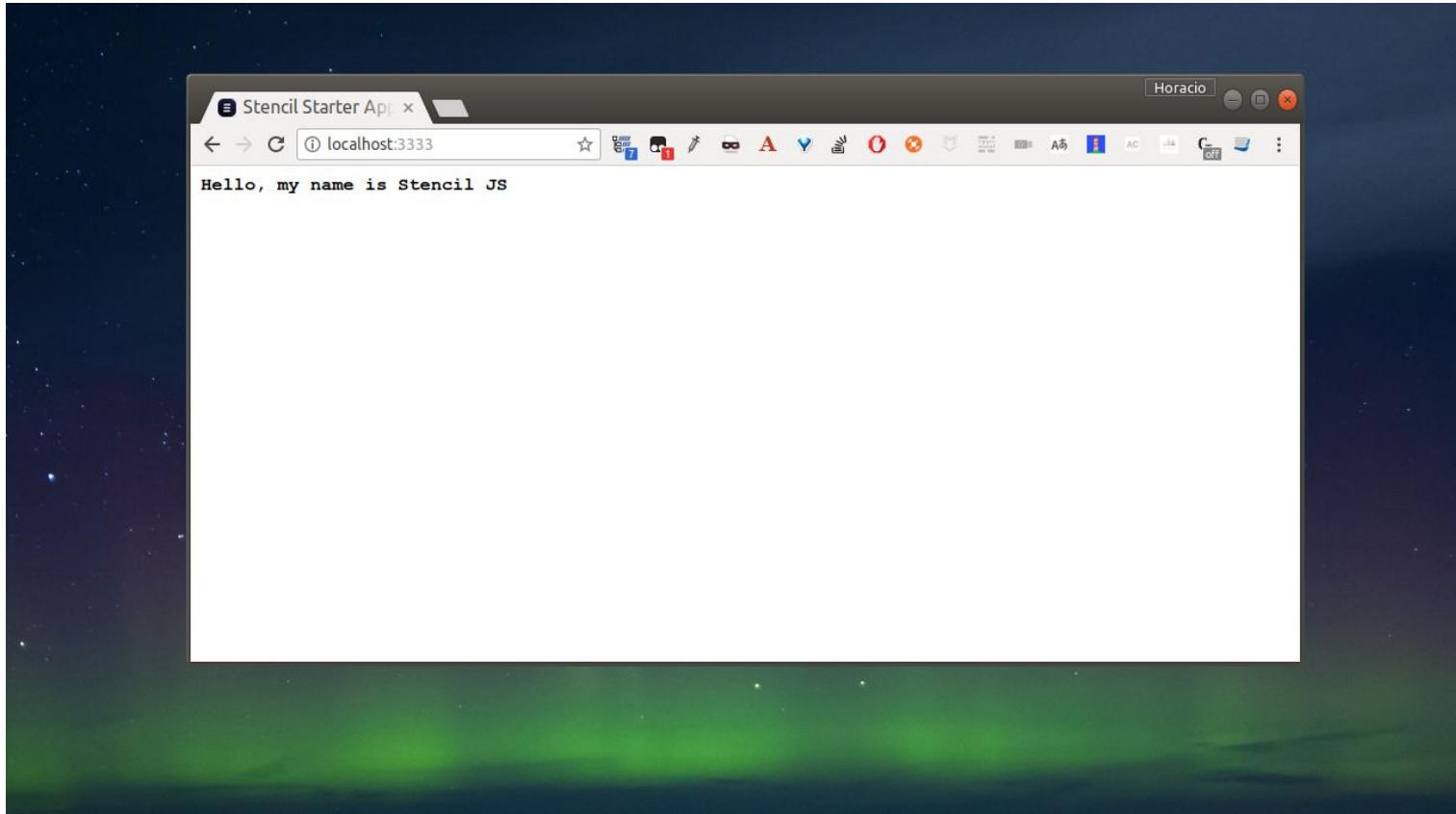
```
git clone https://github.com/ionic-team/stencil-app-starter my-app  
cd my-app  
git remote rm origin  
npm install
```

## Start a live-reload server

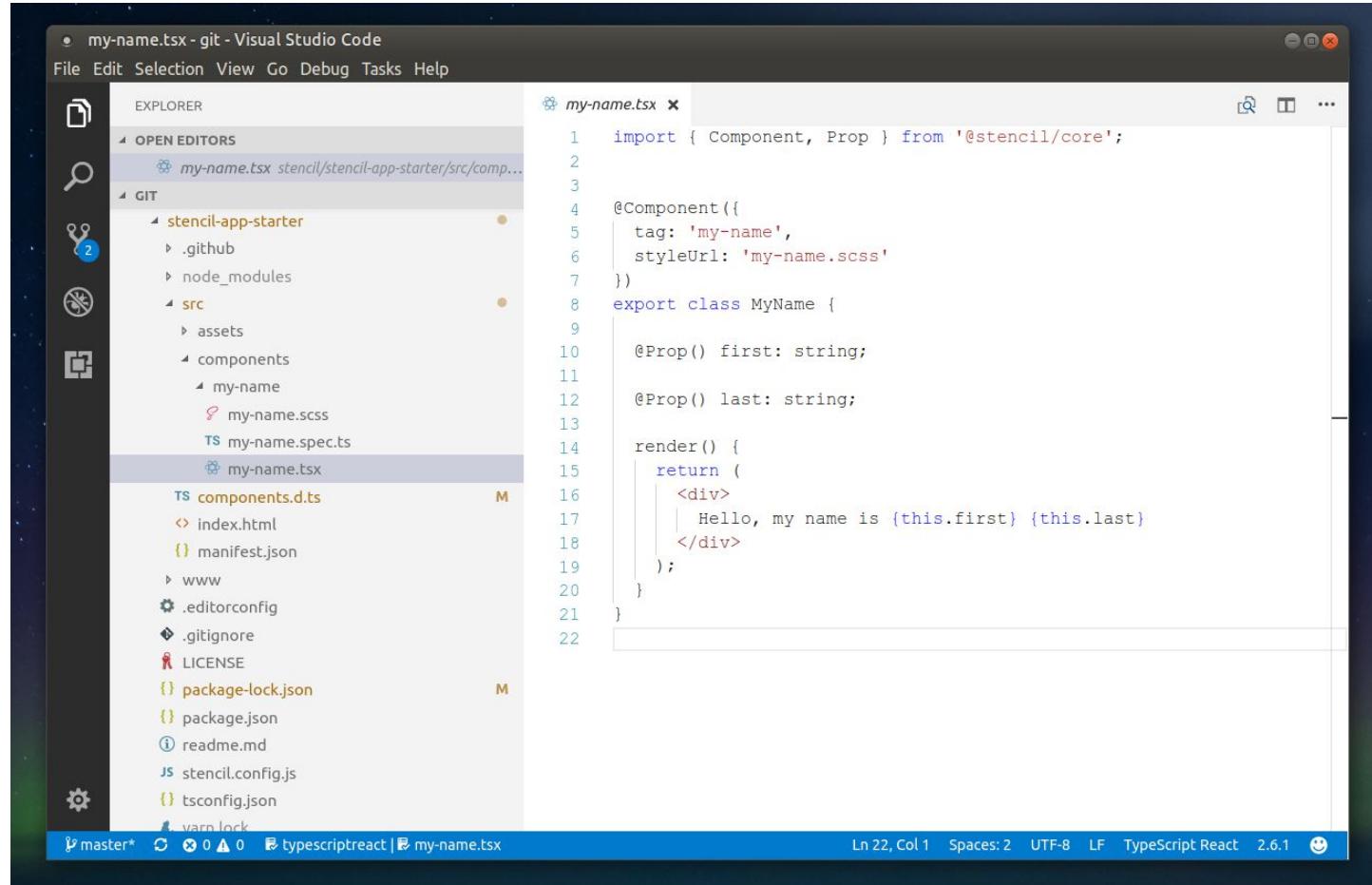
```
npm start
```



# Hands on Stencil



# Hands on Stencil



The screenshot shows a Visual Studio Code interface with the following details:

- File Explorer:** Shows the project structure for "stencil-app-starter". The "src" folder contains "components" which has "my-name" and "my-name.tsx". Other files in "src" include "assets", "my-name.spec.ts", "components.d.ts", "index.html", "manifest.json", and "www".
- Code Editor:** The file "my-name.tsx" is open, displaying the following code:

```
1 import { Component, Prop } from '@stencil/core';
2
3
4 @Component({
5   tag: 'my-name',
6   styleUrl: '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 code implements a Stencil component named "my-name" that takes "first" and "last" properties and renders them inside a div.



# 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',  
  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',
  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;
  }

  componentDidLoad() {
    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>
    );
  }
}
```



# Conclusion

# That's all folks!

