{codemotion}

We code the future. Together

24-25 September, 2019

{codemotion}

24-25 September, 2019

Are Web Components the Betamax of web development?

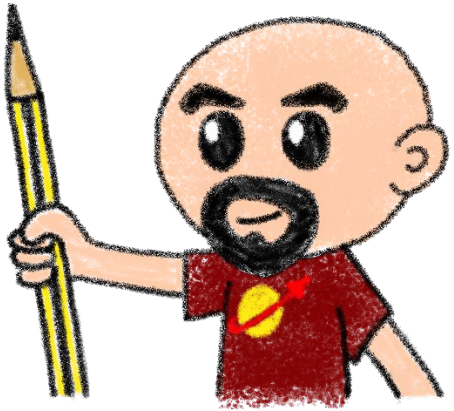
Horacio Gonzalez

OVH

@LostInBrittany

Who are we?

Introducing myself and
introducing OVH



Horacio Gonzalez

@LostInBrittany

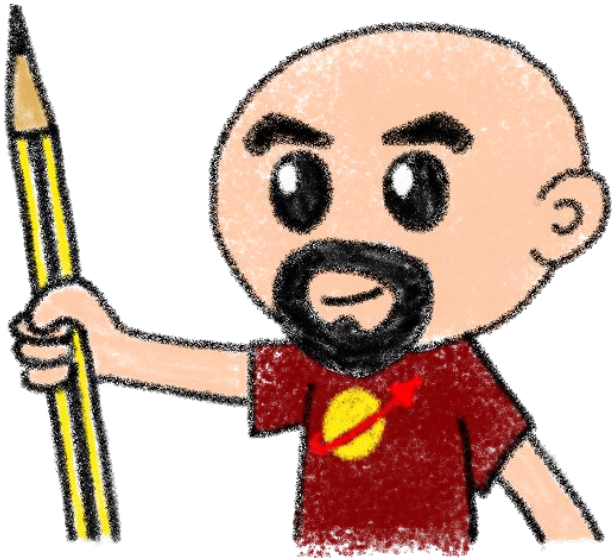
Spaniard lost in Brittany, developer, dreamer and all-around geek

DevFest du Bout du Monde

Finist Devs

Google Developers Experts 2019
Web Technologies GDE

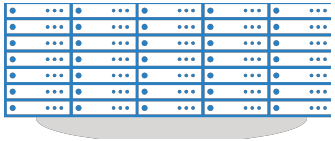
OVH
Team DevRel



OVH: A Global Leader on Cloud

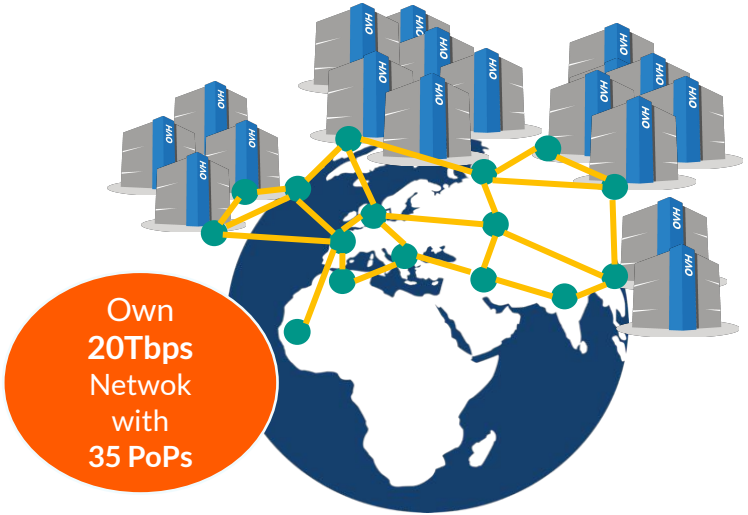
200k Private cloud VMs running

Dedicated IaaS Europe



Hosting capacity : **1.3M** Physical Servers

360k Servers already deployed



30 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

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

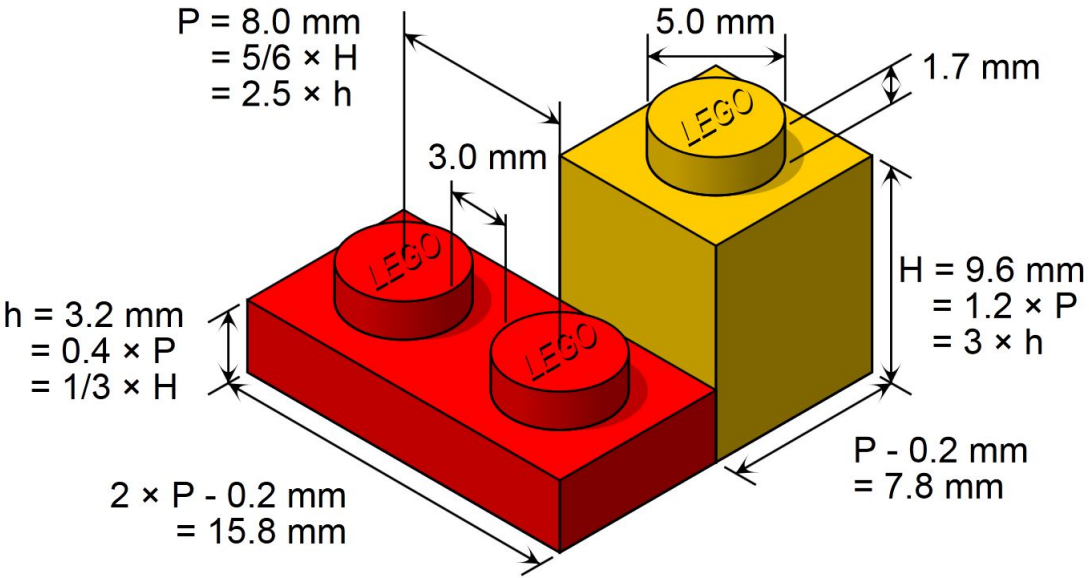
To have clonable document template

```
<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 Web Components tour since 2014

Web components == Revolution

Image: bu.edu

Building a world brick by brick

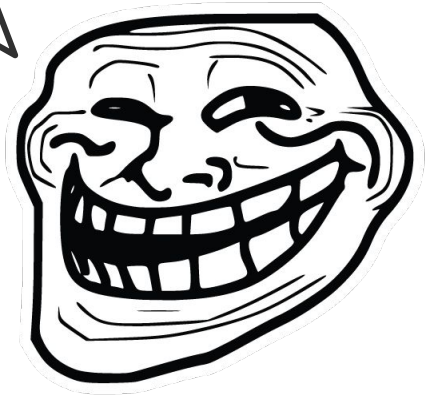
Images: BitRebels & Brickset

Is the promise unfulfilled?

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

And, even worse, several months ago

Hey, dude, your Web Components thing is like Betamax
Even if it was a better solution, market has already chosen,
React has won, as VHS did...



Hey, old man, WTF is a Betamax?

A videocassette guide for Millenials

At the beginning there was the TV

And public saw it was good...

But how to keep your favorite show forever?

Sony VTR CV-2000 - Image credit LabGuy's World

The VTR was born, somewhere in the 1960s

From videotape to videocassette...

Sony U-matic - Image credit MKH Electronics

U-matic cassette - Image credit PSAP

And then to mass market, sometime in the 1970s

Each vendor proposed their solution

Sony's Betamax - Image credit PSAP

JVC's VHS - Image credit PSAP

Cassettes aren't so different from JS frameworks after all...

VHS

Size: 7 3/8” x 4 1/8”

Length: SP: 2 hours, LP: 4 hours, EP: 6 hours

Betamax*

Size: 6 1/8” x 3 3/4”

Length: SP: 2 hours, LP: 4 hours, EP: 6 hours

8mm / Hi-8

Size: 3 3/4” x 2 3/8”

Length: SP: 2 hours, LP: 4 hours

VHS-C

Size: 3 5/8” x 2 1/4”

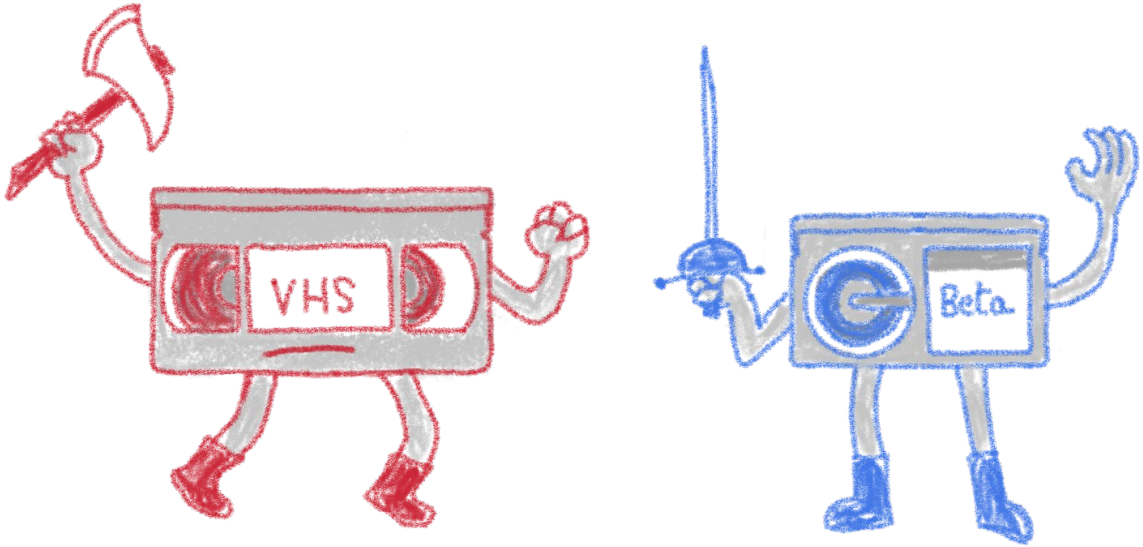
Length: SP: 30 minutes, LP: 90 minutes, EP: 120 minutes

Mini DV

Size: 2 5/8” x 1 7/8”

Length: SP: 60 minutes, LP: 90 minutes

There was a format war

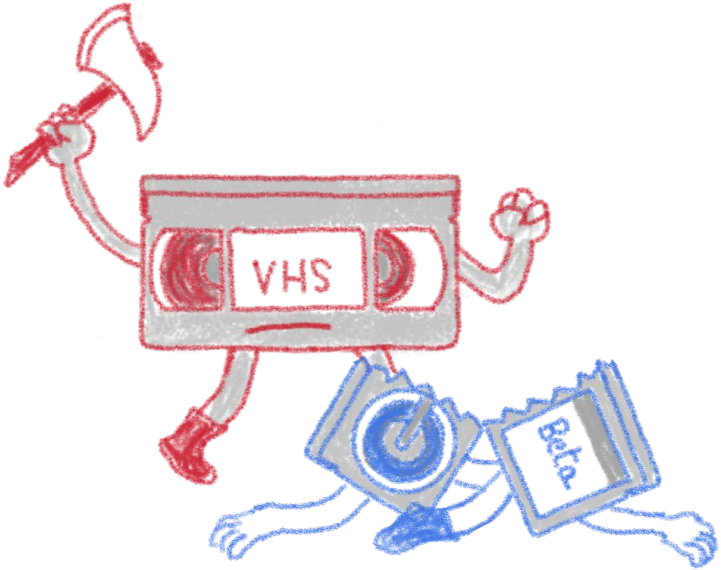


So fierce that it has its own Wikipedia page

Betamax was a superior format

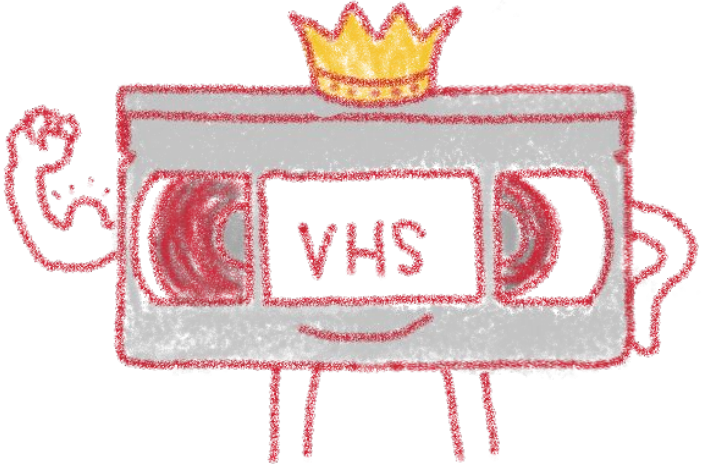
Higher quality recorders, better resolution, slightly superior sound, and more stable image

But the market decided otherwise



And Betamax, even if superior, failed...

As usual, the winner took it all

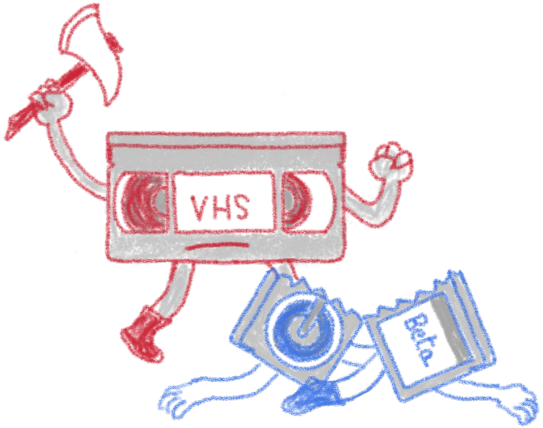


Until a new arrival entered in scene, the DVD...

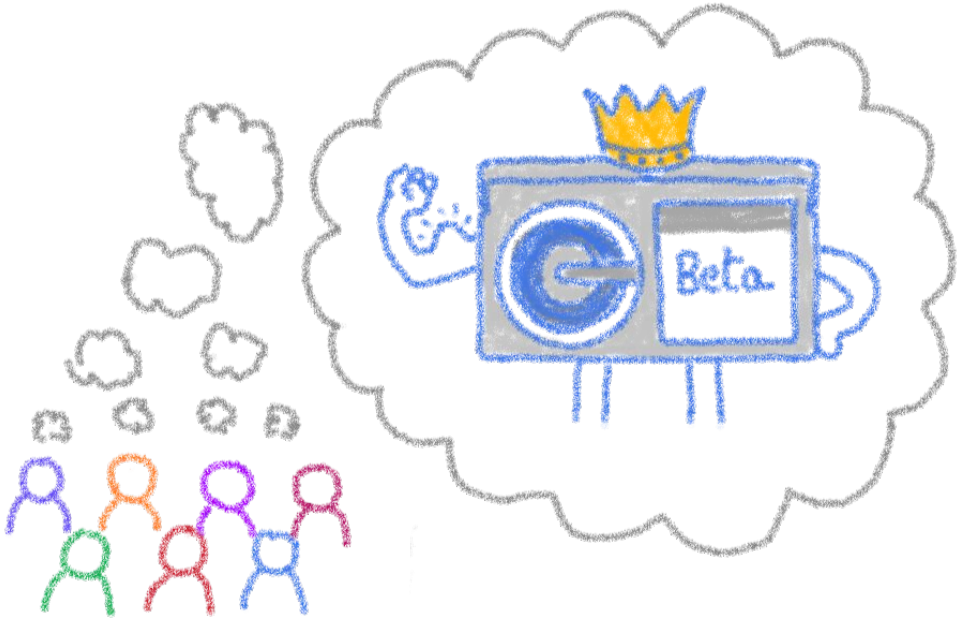
But that's another story for another talk

Why did Betamax failed?

Spoiler: it isn't so simple...

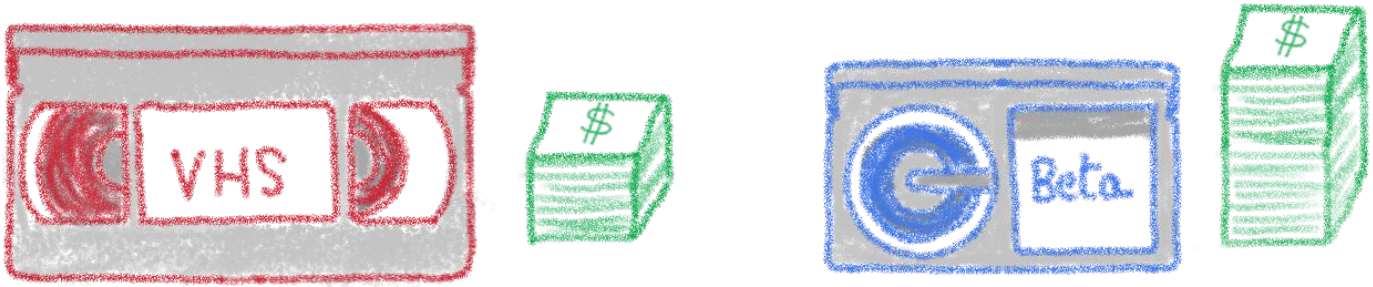


Betamax was believed to be superior



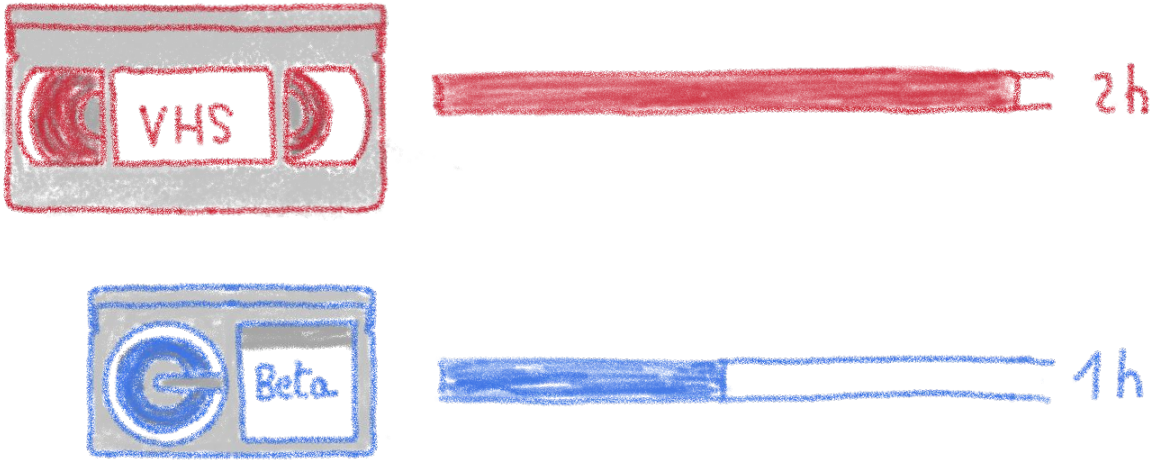
In the minds of the public and press

But consumers wanted an affordable VCR



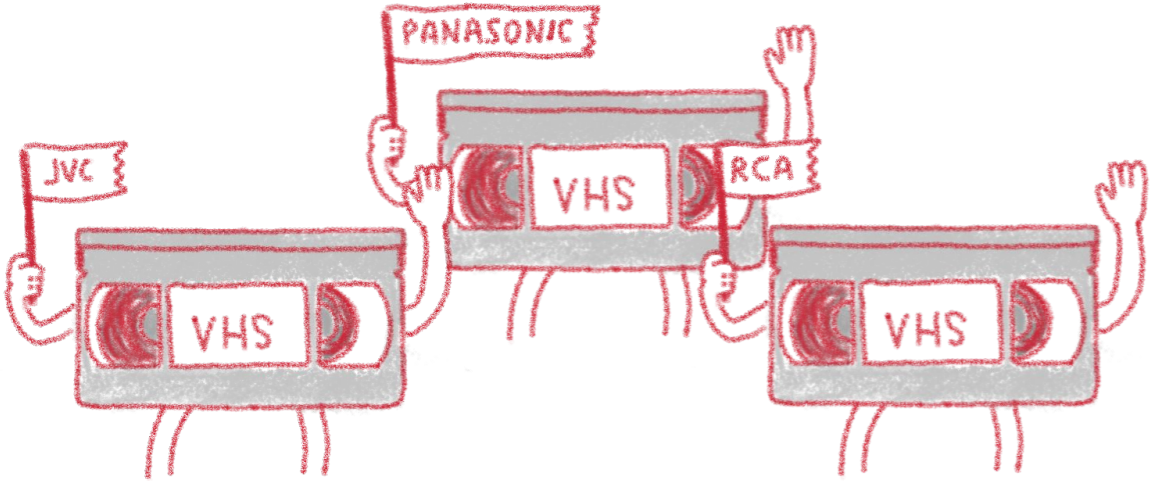
Betamax costed hundreds of dollars more

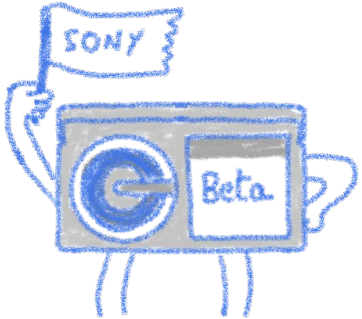
They also wanted to record a full movie



Originally Betamax cassettes only recorded 1 hour

And compatibility weighted on VHS side

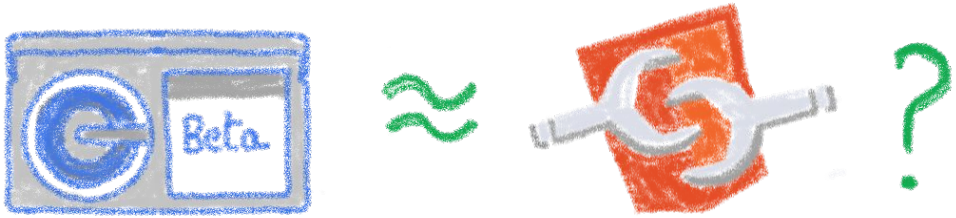




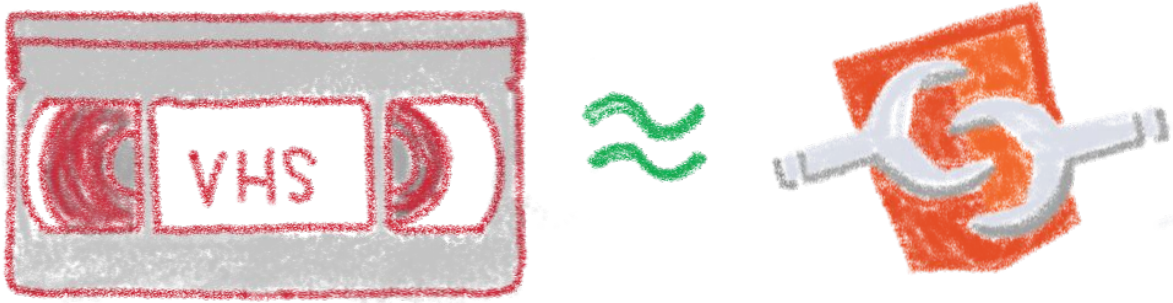
Many licencees offered VHS VCRs

Are Web Components like Betamax?

A perceived superior alternative destined to fail?



It could be even the opposite...



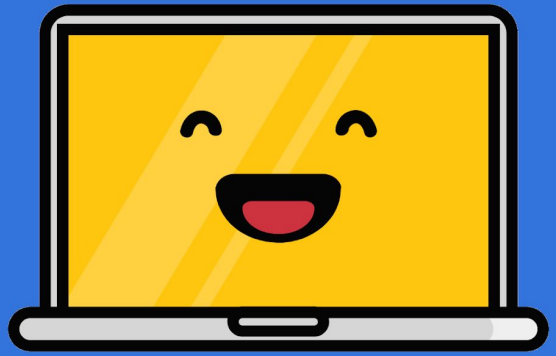
Web components are maybe the VHS of JS

Compatibility is on Web Components side

custom-elements-everywhere.com

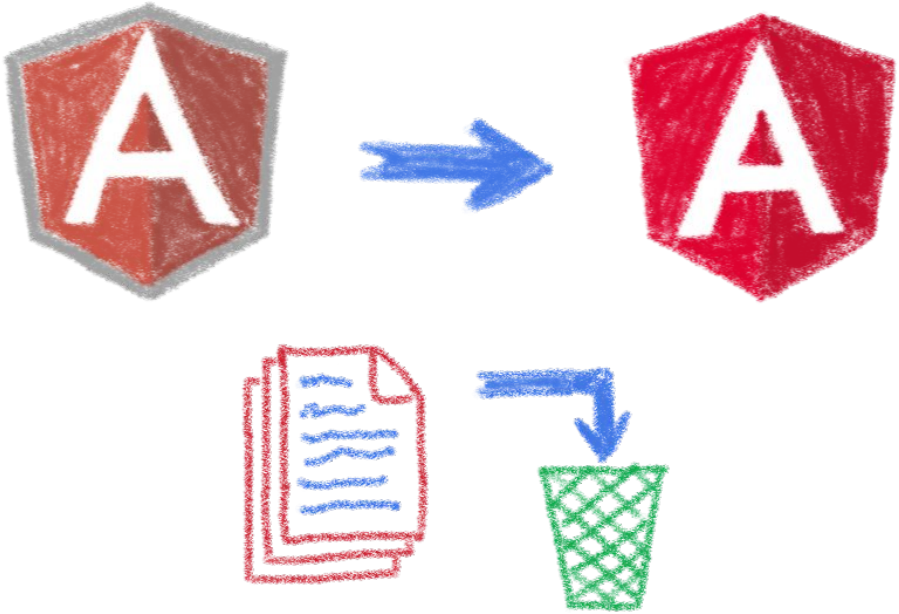
Custom Elements Everywhere

Making sure frameworks and custom elements can be BFFs



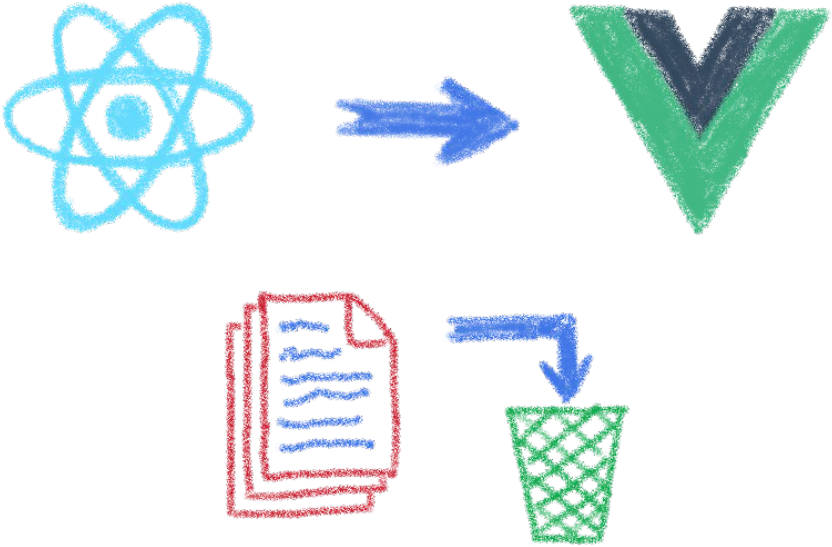
Web Components everywhere, baby!

Do you remember AngularJS?



And all the code put in the trash bin when Angular arrived...

The pain of switching frameworks?

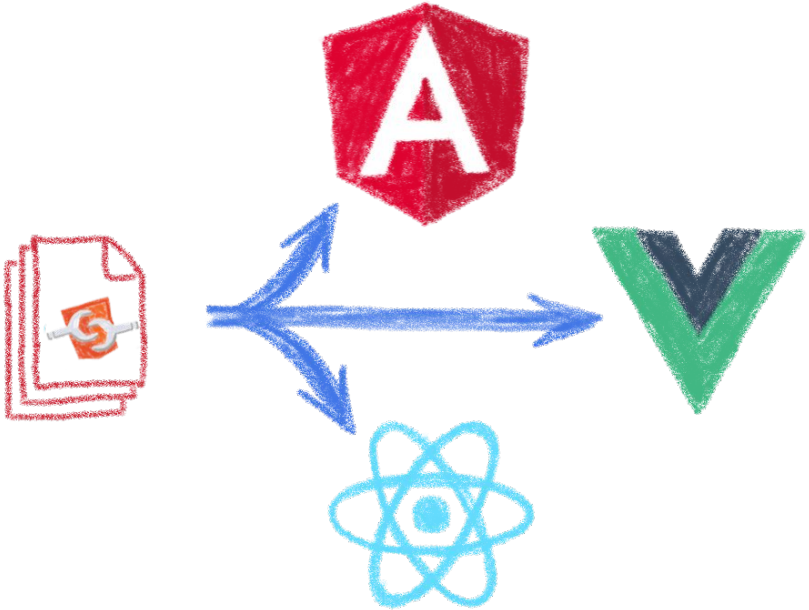


Rewriting once again your code...

The impossibility of sharing UI code?

Between apps written with different frameworks

Web Components change that



In a clean and standard way

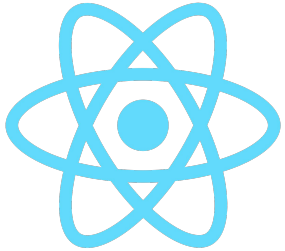
They are indeed a revolution

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

Yann Bertrand and 2 others Retweeted

Justin Fagnani @justinfagnani · 23 Oct 2018

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

Mozilla Hacks @mozhacks

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…

4 76 211

Truly part of the platform...

Aren't the multiple Web Components libs a sign of failure?

If the standard worked, people would use Vanilla, wouldn't them?

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

Stencil

Powering Ionic 4

STENCIL

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

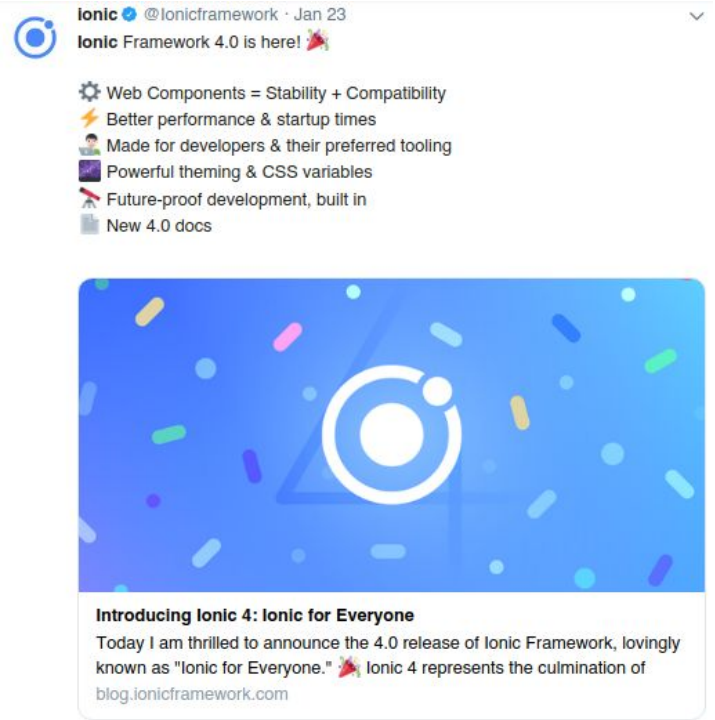
6kb min+gzip runtime, server side rendering, and the raw power of native Web Components.

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

Polymer

Is the old player still alive?

Polymer evolved again in 2018

Image: © Nintendo

Polymer 3 was here!

What's Polymer status today?

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:

LitElement

An ultra-light, highly performant custom element base class with a simple but expressive API.

PWA Starter Kit

Your one-stop shop for building Progressive Web Apps that make the most of the modern web.

Material Web Components

Pixel-perfect realizations of Google's Material Design spec that work anywhere on the web.

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

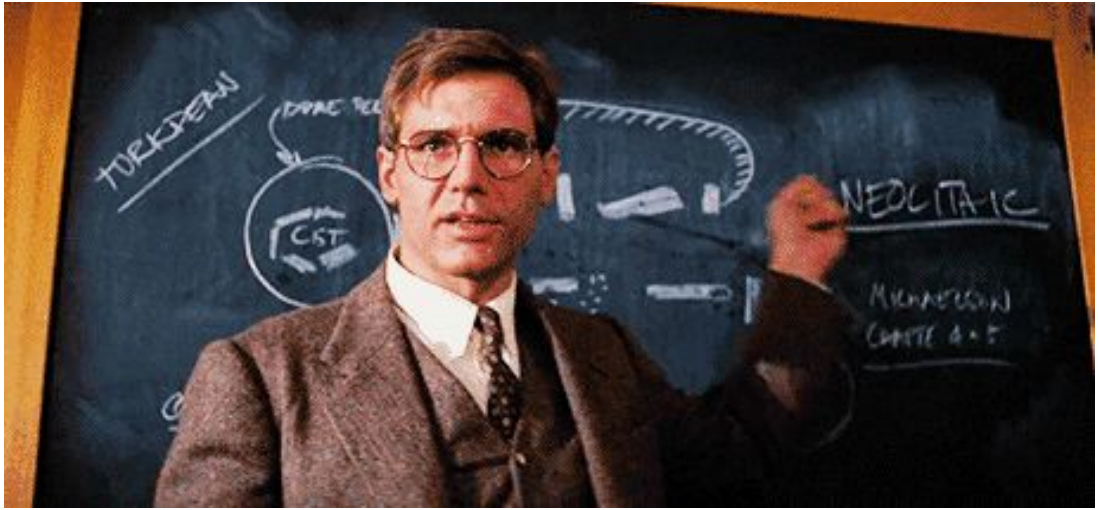
No React, Angular or Vue innovations

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?

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

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

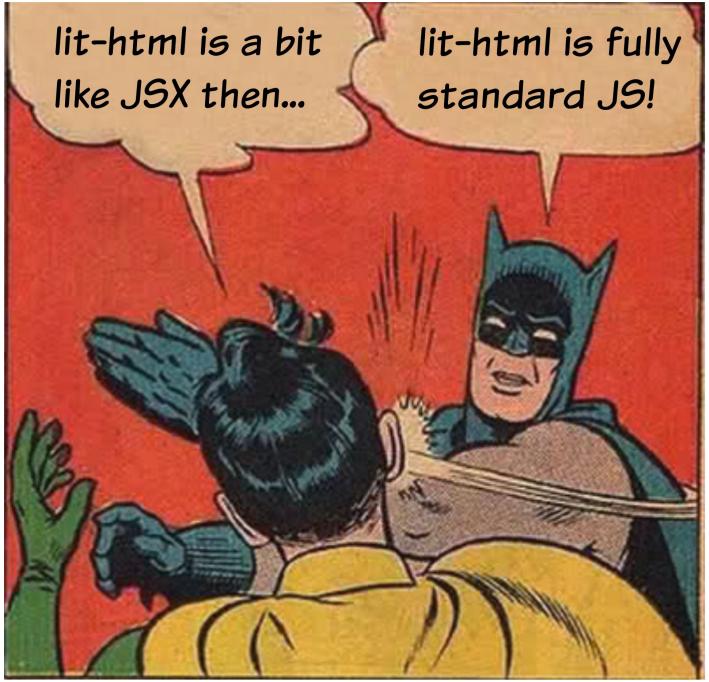
lit-html Templates

```
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

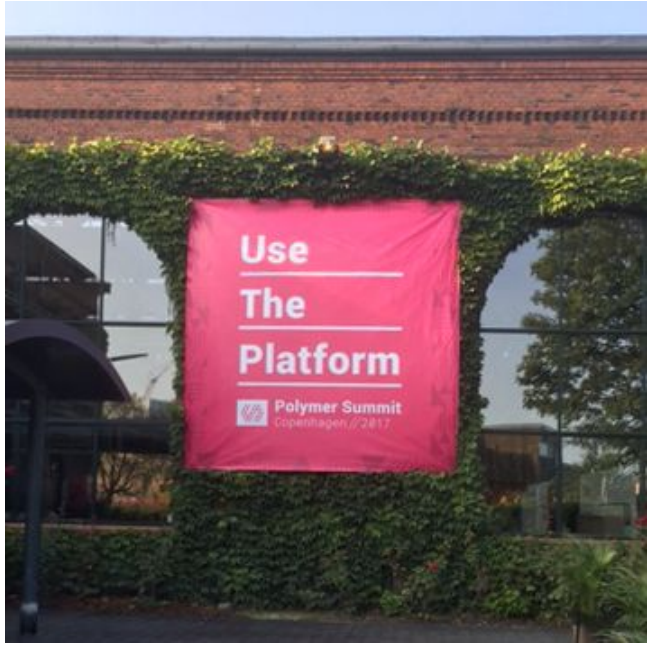
One more thing...*

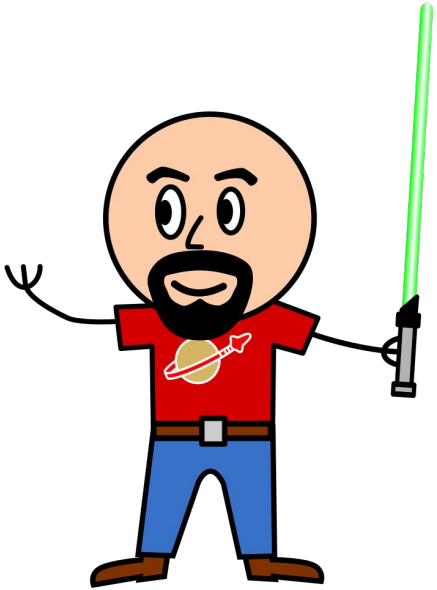
Let's copy from the master

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

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!