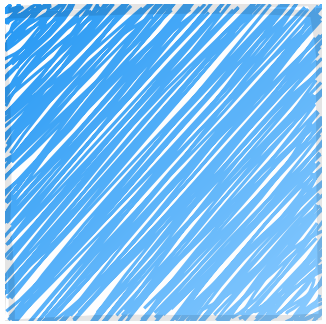
Web Components:

Natif

StencilJS



LitElement

Julien Renaux

GDE Web, Freelancer

@julienrenaux

https://julienrenaux.fr/

Toptal

Igor Laborie

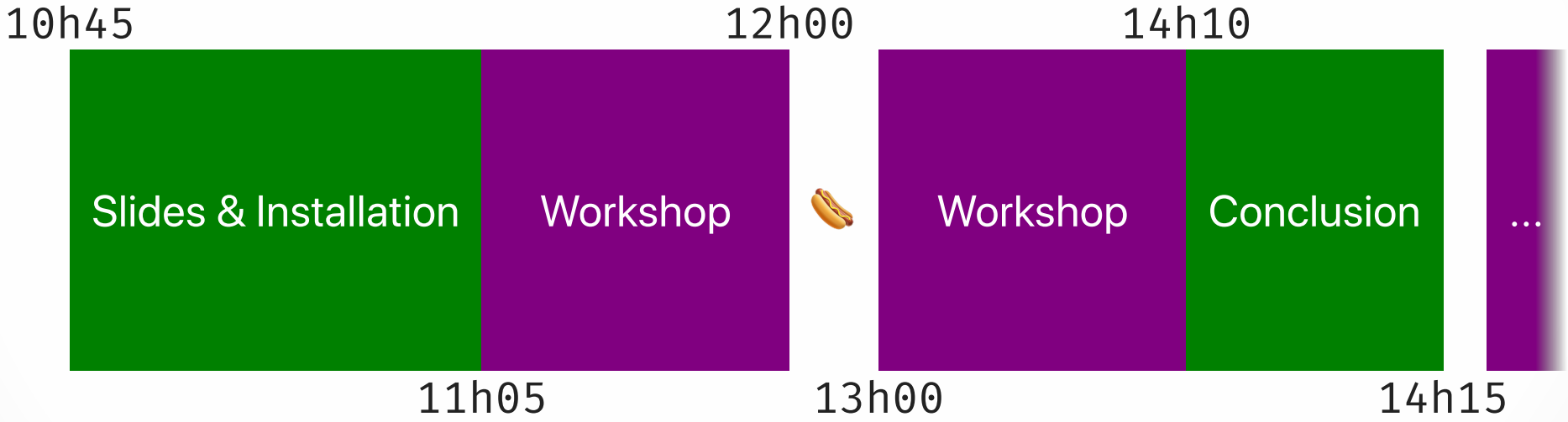
Expert Web & Java

@ilaborie

igor@monkeypatch.io

MonkeyPatch

Roadmap

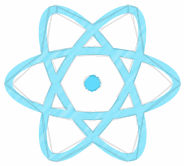

10h45
12h00
14h10
Slides & Installation
Workshop
Workshop
Conclusion
...
11h05
13h00
14h15

Instructions

TODO: Installation instructions, Wifi

I want to build a Web app in 2019

Let's start by picking up a Framework


React


VueJs


Angular

Now let's select how to write our style

- CSS
- Sass/Scss
- Less
- Stylus
- CSS-in-JS
- PostCSS
- NextCSS
- ...

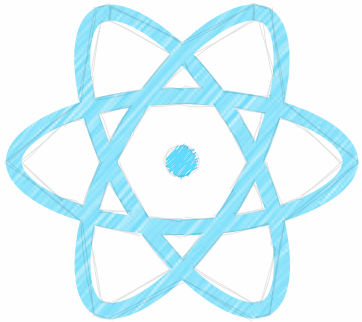
Now let's transpile our code

- Webpack
- ParcelJs
- RollupJs
- Bazel

Developing an app in JS in not easy anymore...

The industry is moving too fast...

...Interoperability is not available out of the box...

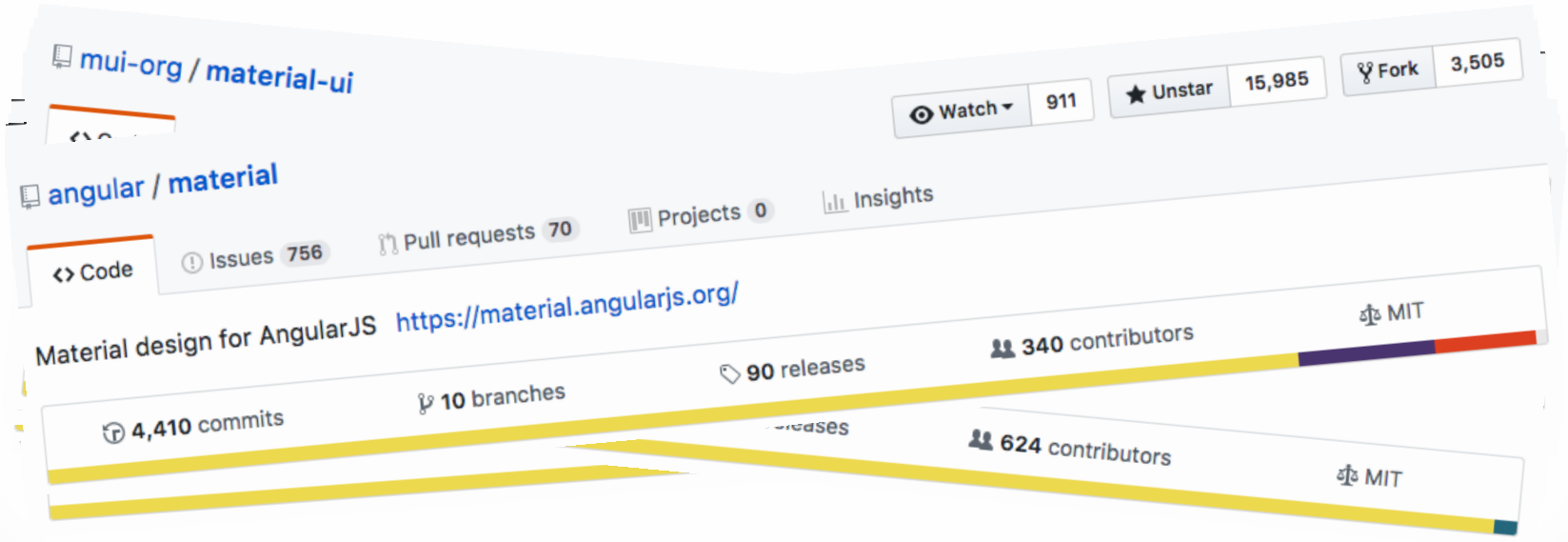


React

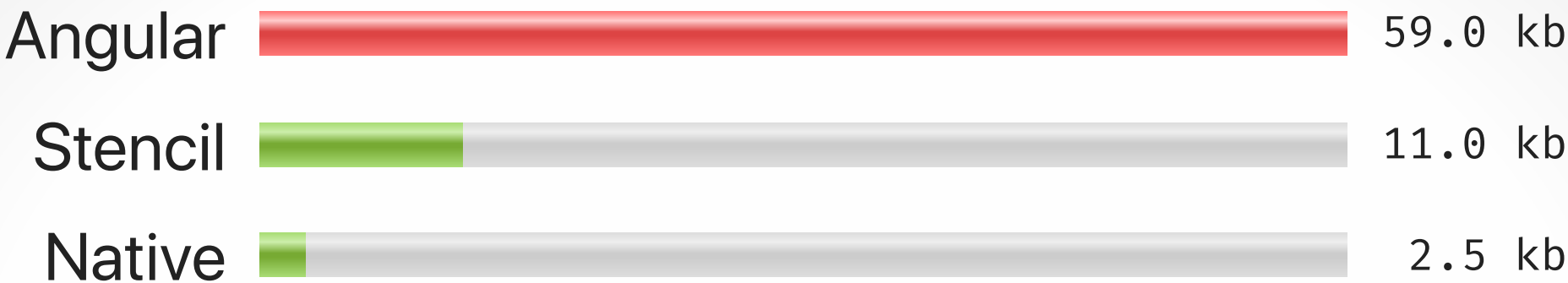
VueJs

Angular

...And we keep reinventing the wheel!



All this complexity is coming to an end with Web Components

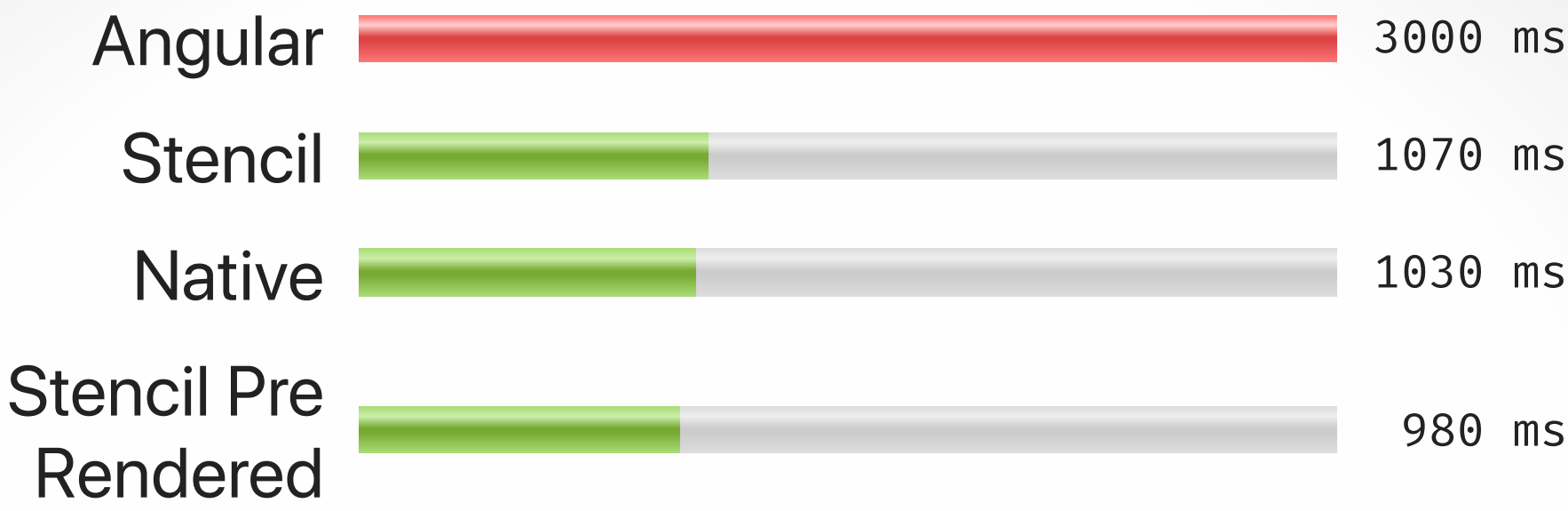


Size matters (Gzipped)

😨 Stencil is 5 times smaller than Angular

😱 Native is 23 times smaller than Angular

TODO: Add litElement, Update numbers



Time matters (FMP 3G 📱 in ms)

😨 Native & Stencil 3 times faster than Angular

TODO: Add litElement, Update numbers

Web Components

Specs from **World Wide Web Consortium** (W3C)

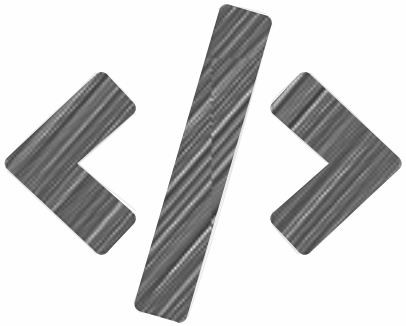
First draft in **2012**

Custom Elements

Shadow DOM

HTML templates

HTML imports



Custom Elements

> *Custom Elements is a capability for creating your own custom HTML elements with its own methods and properties*

Shadow DOM

> *Shadow DOM provides encapsulation for DOM and CSS*

HTML templates

> *Give the ability to create reusable piece of HTML that can be used at runtime*

```
1  class PopUpInfo extends HTMLElement {
2      constructor() {
3          super();
4          //  ...
5      }
6      //  ...
7  }
8
9  customElements.define('popup-info', PopUpInfo);
```

Browser support

	11	18	65	72	12
Custom Elements (V1)	😫	😫	😃	😃	🙂
Shadow DOM (V1)	😫	😫	😃	😃	🙂
HTML templates	😫	😃	😃	😃	😃

Supported

Partial Support

Not Supported

https://caniuse.com

Polyfill support



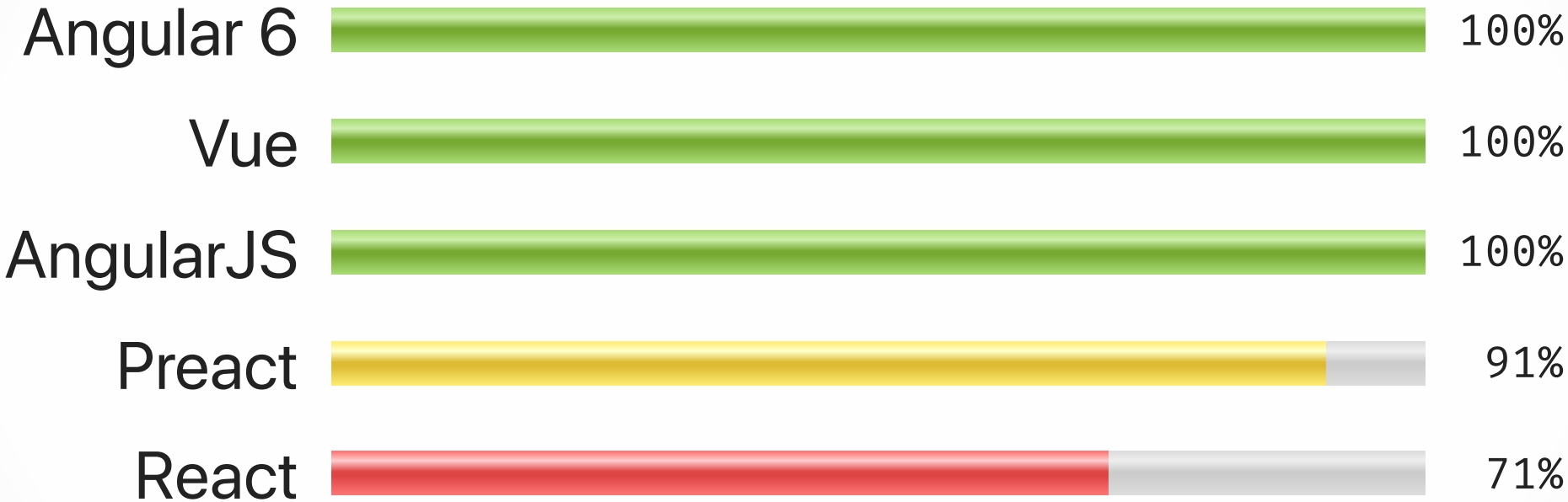


11+ 9+

Custom Elements (V1)	✔	✔	✔	✔	✔
Shadow DOM (V1)	✔	✔	✔	✔	✔
HTML templates	✔	✔	✔	✔	✔

https://github.com/webcomponents/webcomponentsjs

Web component interoperability support



https://custom-elements-everywhere.com/

StencilJS

https://stenciljs.com/

Open Source project, ➳ MIT License

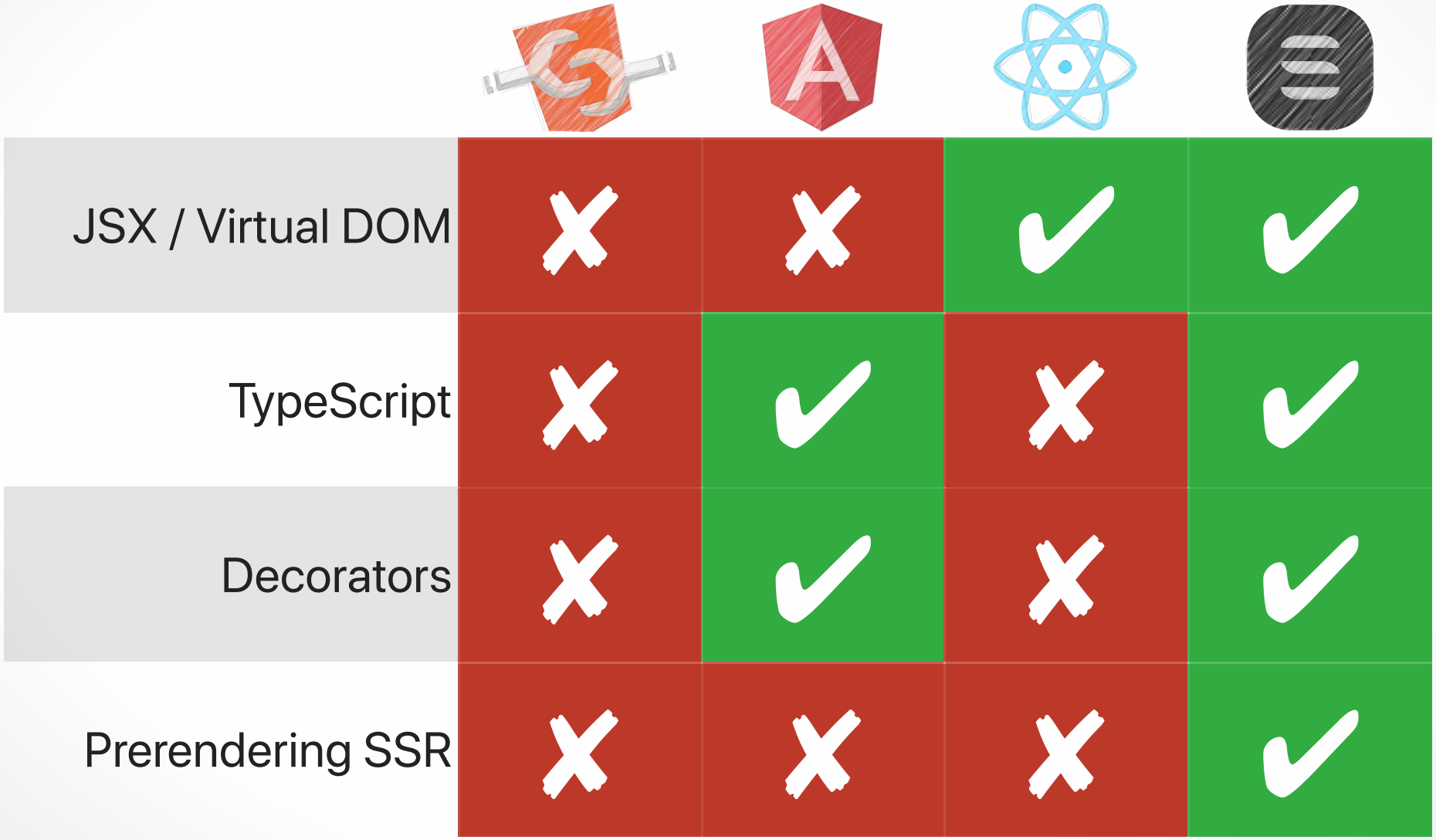
Created by the **Ionic Team** in 2017

4.9k ⭐ on github

StencilJS is not another framework

StencilJS is a compiler that generates web components

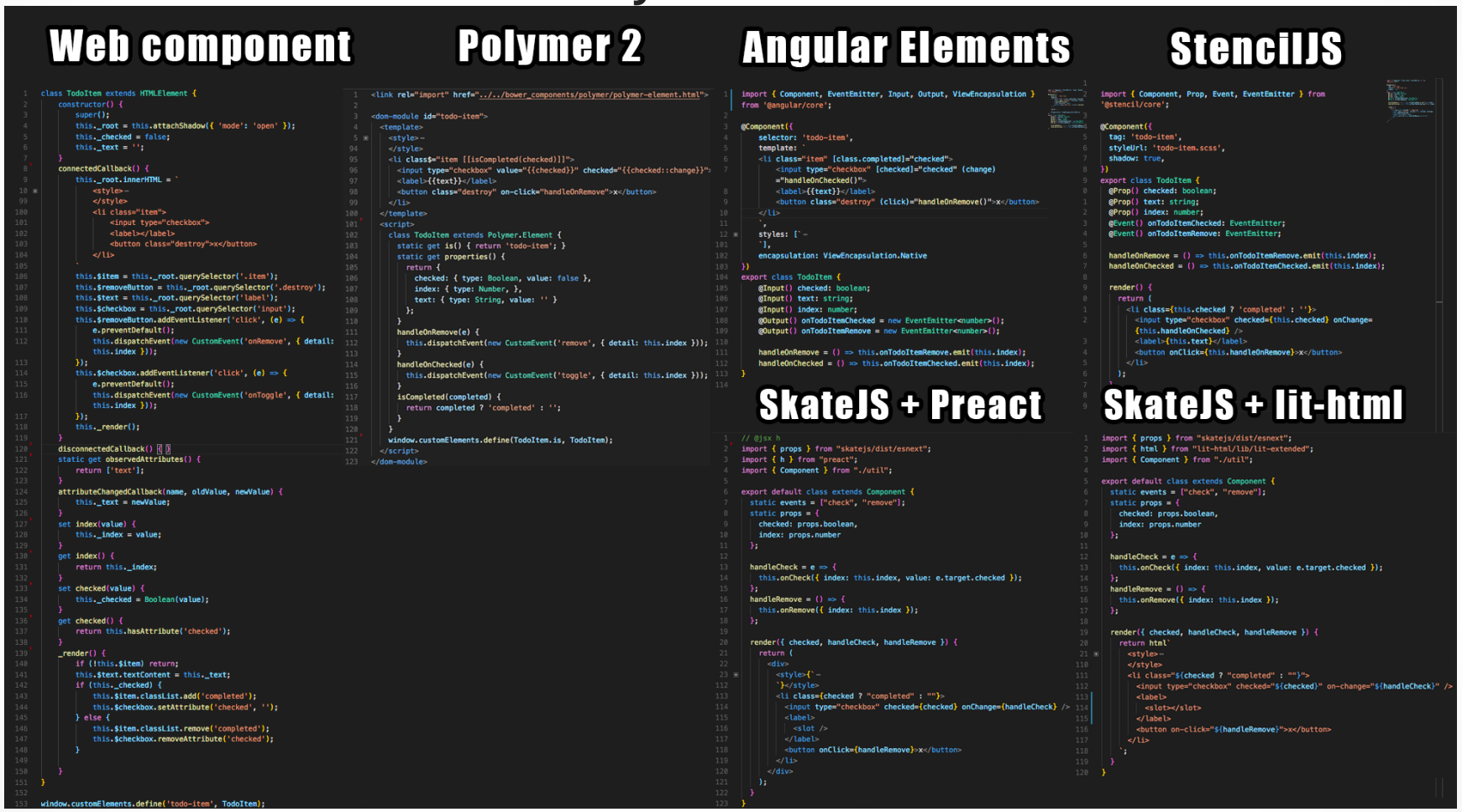
StencilJS is a set of great tools



StencilJS works everywhere

Loads polyfills on-demand

Stencil Syntax is concise



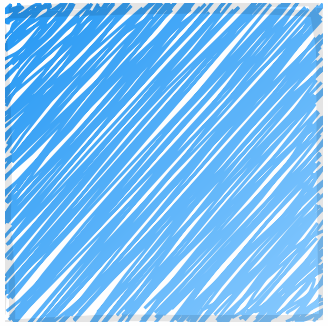
Getting started


```
$ npm init stencil
```

```
? Pick a starter › - Use arrow-keys. Return to submit.
❯  ionic-pwa     Everything you need to build fast, production ready PWAs
   app           Minimal starter for building a Stencil app or website
   component     Collection of web components that can be used anywhere
```

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

@Component({
    tag: 'my-first-component',
    styleUrl: 'my-first-component.scss'
})
export class MyComponent {
    // Indicate that name should be
    // a public property on the component
    @Prop() name: string;

    render() {
        // JSX
        return (<p>My name is {this.name}</p>);
    }
}
```

Lit-Elements



https://lit-element.polymer-project.org/

Open Source project, ➳ BSD 3-Clause License

Created by the **Polymer Team** in 2017

1.7k ⭐ on github

Close to other lightweight WebComponent frameworks

- Stencil
- SkateJS
- Svelte
- Slim.js

But using the lit-html templating library

https://lit-html.polymer-project.org/

built on **HTML templates**

with the ES2015 ➳ Template literals

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

// A lit-html template uses
// the `html` template tag:
const 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);
```

Just extend the `LitElement` class

```js
import { LitElement, html, css } from 'https://unpkg.com/lit-element/l

class MyElement extends LitElement {
    static get properties() {
        return {
            mood: { type: String }
        }
    }
    static get styles() {
        return css `.mood { color: green; }`;
    }
    render() {
        return html `Web Components are
            <span class="mood">${this.mood}</span>!`;
    }
}

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

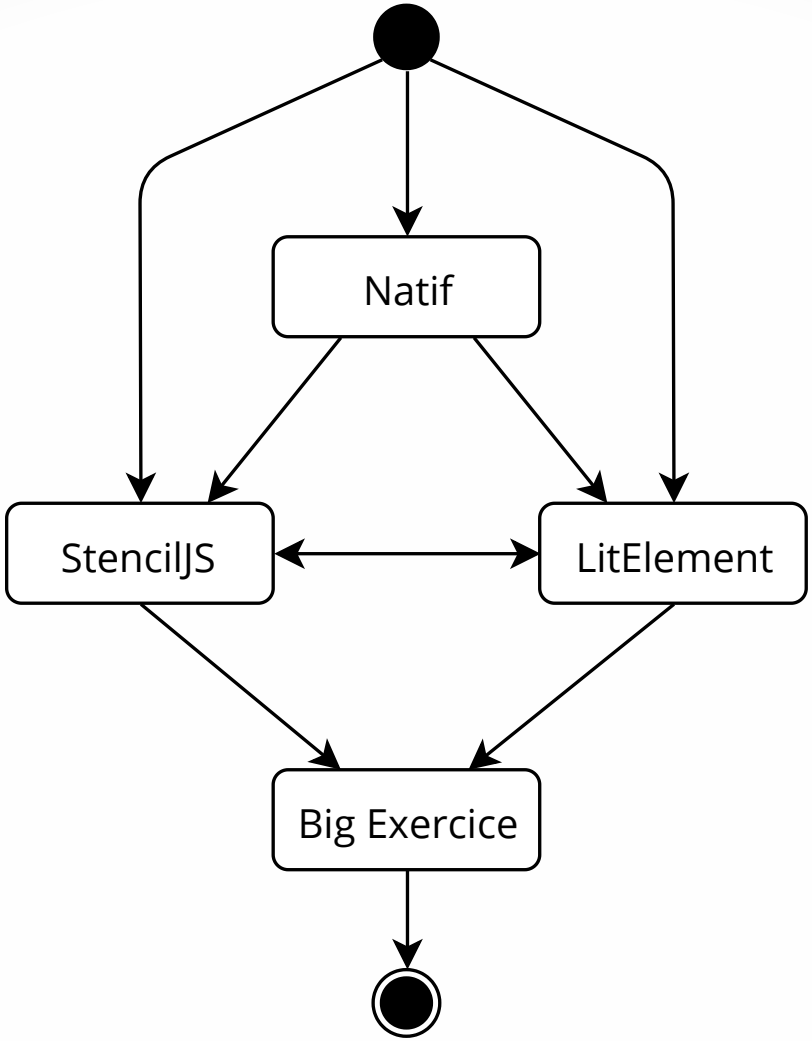
Also can use Typescript with **decorators**

```ts
import { /* ... */ } from 'lit-element';

@customElement('my-element')
class MyElement extends LitElement {

    @property({ type: String }) mood;

    static get styles() {
        return css`.mood { color: green; }`;
    }

    render() {
        return html`Web Components are
            <span class="mood">${this.mood}</span>!`;
    }
}
```

✅ Chrome, Safari, Opera, Firefox

💊 polyfills for Edge and IE 11

Workshop


Natif
StencilJS
LitElement
Big Exercice

Conclusion

Commons Issues

Attribute are `string`

Styling with theme

Browser support

Use an external state manager

Use CSS custom properties

Polyfills or Electron

Moderns Alternatives

- SkateJS
- Svelte
- Slim.js
- ...

End

Thanks