

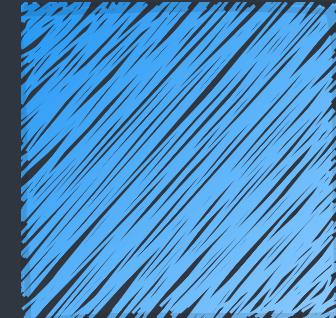
Web Components:



Natif



StencilJS



LitElement





Julien Renaux

GDE Web, Freelancer

@julienrenaux

<https://julienrenaux.fr/>



Igor Laborie

Expert Web & Java

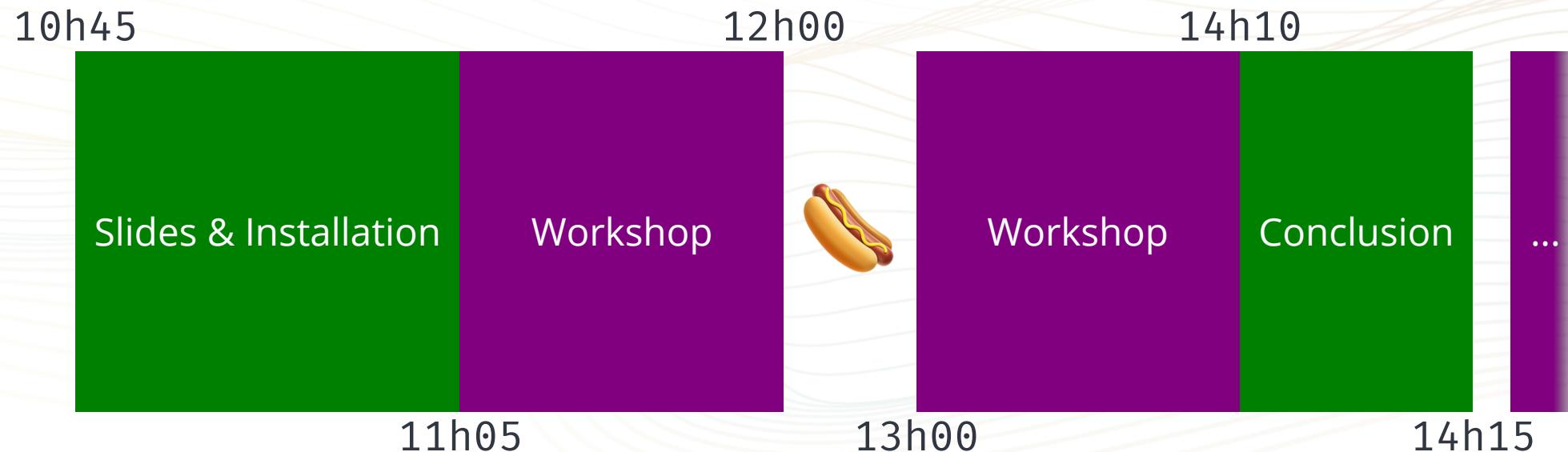
@ilaborie

igor@monkeypatch.io





Roadmap

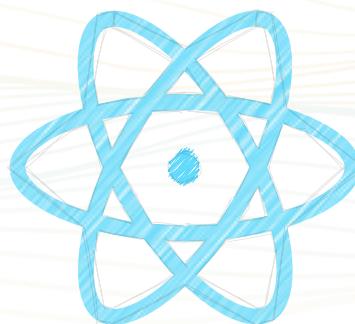


Instructions

TODO: Installation instructions, Wifi

Développons une application Web en 2019

Commençons par choisir un Framework



React



Vue.js



Angular

Puis comment allons-nous écrire le style

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

Puis construisons notre application

- Webpack
- ParcelJs
- RollupJs
- Bazel

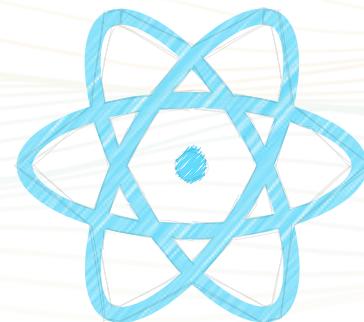


**Développer une application en JS n'est
plus simple...**

Ça va trop vite ...

A screenshot of a Twitter post. The profile picture is a small circular image of a person with short hair. The username is "I Am Devloper" and the handle is "@jamdevloper". The tweet text is: "I think I've had milk last longer than some JavaScript frameworks." Below the tweet are engagement metrics: 1,124 likes and 2:22 PM - Dec 4, 2014. To the right of the tweet is a blue Twitter icon. At the bottom of the card, there is a blue link that says "1,755 people are talking about this" followed by a right-pointing arrow.

...Interopérabilité ne vient plus gratuitement...



React

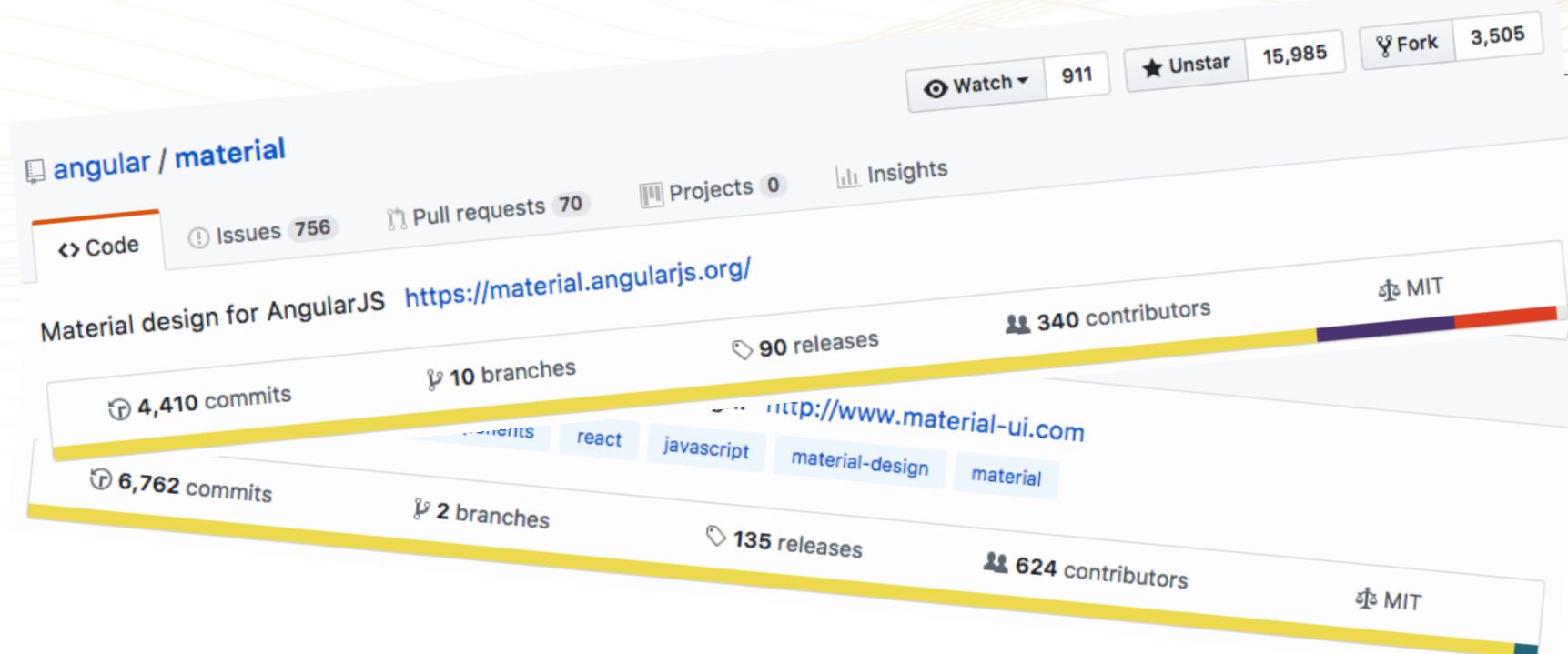


Vue.js



Angular

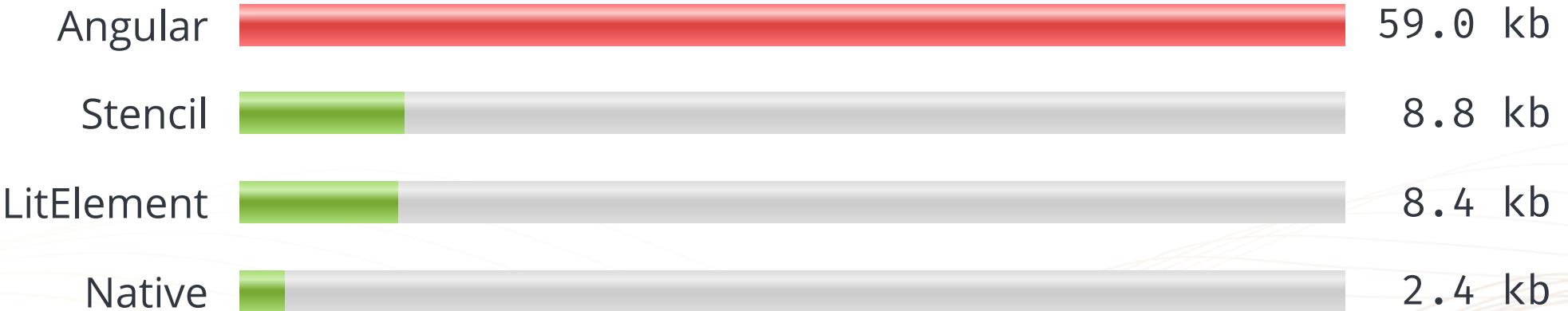
...et on réinvente sans arrêt la roue !



TODO: Meme Luci Do It: add another one in
WebComponent LitElement...

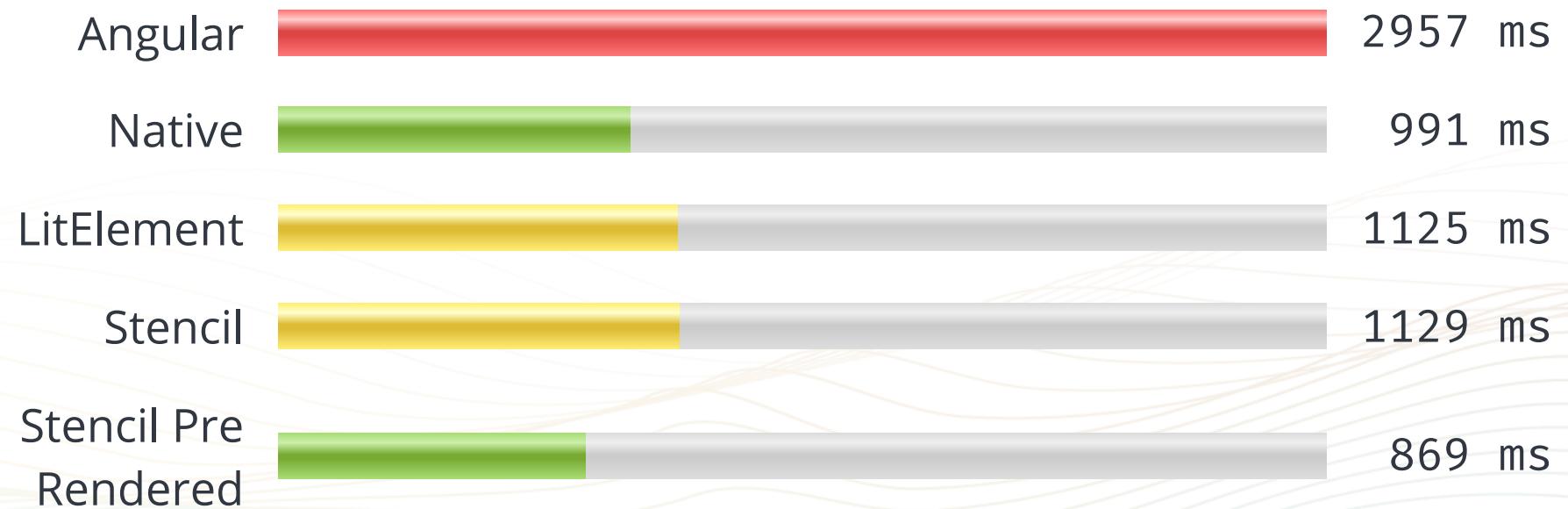


**Toute cette complexité doit
s'arrêter avec les Web
Components**



Size matters (Gzipped)

- 😢 Stencil et litElement sont 5 fois plus petits qu'Angular
- 😱 Native est 23 fois plus petit qu'Angular



Le temps ça compte (FMP 3G 📱 en ms)

😢 Angular est 3 fois plus lent

Web Components



Spécifier par le **World Wide Web Consortium** (W3C)
Débuté en **2012**



Custom
Elements



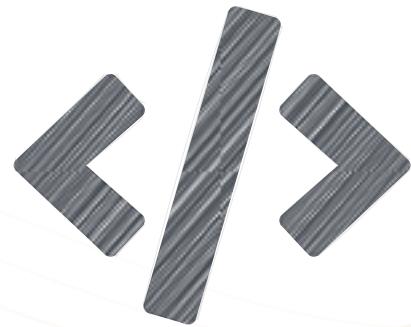
Shadow DOM



HTML templates



HTML Imports



Custom Elements

“ Les Custom Elements sont la capacité de créer un balise HTML avec ses propres attributs et méthodes



Shadow DOM

“ Le Shadow DOM fournit l'encapsulation du DOM et du CSS”



HTML templates

// Définit un bloc d'HTML réutilisable au moment de l'exécution

```
JS
class PopUpInfo extends HTMLElement {
  constructor() {
    super();
    // ...
  }
  // ...
}

customElements.define('popup-info', PopUpInfo);
```

Support des navigateurs



11



18



65



72



12

| | IE | Edge | Firefox | Chrome | Safari |
|----------------------|----|------|---------|--------|--------|
| Custom Elements (V1) | 😢 | 😢 | 😊 | 😊 | 😊 |
| Shadow DOM (V1) | 😢 | 😢 | 😊 | 😊 | 😊 |
| HTML templates | 😢 | 😊 | 😊 | 😊 | 😊 |

Supported

Partial Support

Not Supported

<https://caniuse.com>

Support avec le polyfill



11+

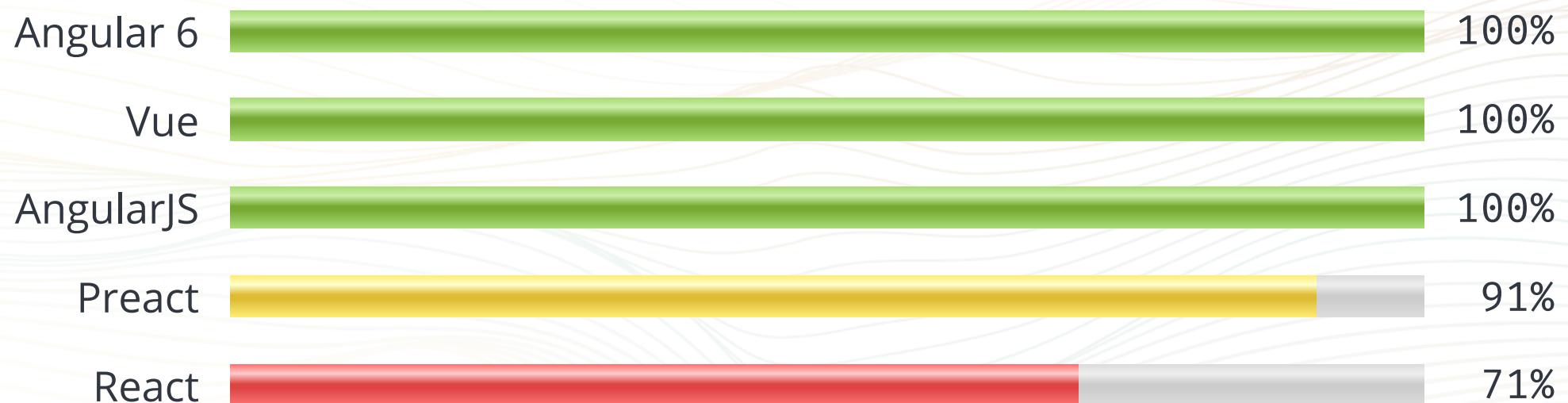


9+

| Custom Elements (V1) | ✓ | ✓ | ✓ | ✓ | ✓ |
|-------------------------|---|---|---|---|---|
| Shadow DOM (V1) | ✓ | ✓ | ✓ | ✓ | ✓ |
| HTML templates | ✓ | ✓ | ✓ | ✓ | ✓ |

<https://github.com/webcomponents/webcomponentsjs>

Interopérabilité des Web Component



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

StencilJS



<https://stenciljs.com/>



Projet **Open Source**, ➔ MIT License

Créé par l'équipe d'**Ionic** en 2017

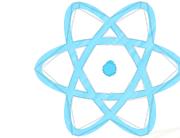
5.1k ⭐ sur github



**StencilJS n'est pas un autre
framework**

**StencilJS c'est un compilateur qui
génère des web components**

StencilJS c'est un ensemble de bons outils

| |  |  |  |  |
|-------------------|--|---|---|---|
| JSX / Virtual DOM | ✗ | ✗ | ✓ | ✓ |
| TypeScript | ✗ | ✓ | ✗ | ✓ |
| Decorators | ✗ | ✓ | ✗ | ✓ |
| Prerendering SSR | ✗ | ✗ | ✗ | ✓ |

StencilJS marche partout

Il charge les polyfills à la demande

La syntaxe de Stencil est concise

Web component

```
1 class TodoItem extends HTMLElement {
2   constructor() {
3     super();
4     this._root = this.attachShadow({ 'mode': 'open' });
5     this._checked = false;
6     this._text = '';
7   }
8   connectedCallback() {
9     this._root.innerHTML =
10     `<style>
11       <input type="checkbox">
12       <label><input checked="" type="checkbox"/> ${this._text}</label>
13     </style>
14     <li class="item">
15       <input type="checkbox">
16       <label><input checked="" type="checkbox"/> ${this._text}</label>
17       <button class="destroy">x</button>
18     </li>
19   `;
20   this.$item = this._root.querySelector('.item');
21   this.$removeButton = this._root.querySelector('.de');
22   this.$text = this._root.querySelector('label');
23   this.$checkbox = this._root.querySelector('input');
24   this.$removeButton.addEventListener('click', (e) => {
25     e.preventDefault();
26     this.dispatchEvent(new CustomEvent('onRemove', { index: this._index }));
27   });
28   this.$checkbox.addEventListener('click', (e) => {
29     e.preventDefault();
30     this.dispatchEvent(new CustomEvent('onToggle', { index: this._index }));
31   });
32   this._render();
33 }
34 disconnectedCallback() []
35 static get observedAttributes() {
36   return ['text'];
37 }
38 attributeChangedCallback(name, oldValue, newValue) {
39   if (name === 'text') {
40     this._text = newValue;
41   }
42   setIndex(newValue);
43 }
44 setIndex(value) {
45   this._index = value;
46 }
47 getIndex() {
48   return this._index;
49 }
50 set checked(value) {
51   this._checked = Boolean(value);
52 }
53 get checked() {
54   return this.hasAttribute('checked');
55 }
56 _render() {
57   if (!this.$item) return;
58   this.$text.textContent = this._text;
59   if (this._checked) {
60     this.$item.classList.add('completed');
61     this.$checkbox.setAttribute('checked', '');
62   } else {
63     this.$item.classList.remove('completed');
64     this.$checkbox.removeAttribute('checked');
65   }
66 }
67 
```

Polymer 2

```
1 <link rel="import" href="../../bower_components/polymer/polymer.html">
2 <dom-module id="todo-item">
3   <template>
4     <style>
5       <!-- style -->
6     </style>
7     <li class="item {{isCompleted(isCompleted)}}">
8       <input type="checkbox" value="{{checked}}>
9       <label>{{text}}</label>
10      <button class="destroy" on-click="handleOnRemove">x</button>
11    </li>
12  </template>
13  <script>
14    class TodoItem extends Polymer.Element {
15      static get is() { return 'todo-item'; }
16      static get properties() {
17        return {
18          checked: { type: Boolean, value: false },
19          index: { type: Number, },
20          text: { type: String, value: '' }
21        };
22      }
23      handleOnRemove(e) {
24        this.dispatchEvent(new CustomEvent('remove', { detail: e }));
25      }
26      handleOnChecked(e) {
27        this.dispatchEvent(new CustomEvent('toggle', { detail: e }));
28      }
29      isCompleted(completed) {
30        return completed ? 'completed' : '';
31      }
32    }
33  </script>
34 </dom-module>
```

Angular Elements

```
1 import { Component, EventEmitter, Input, Output, ViewEncapsulation } from '@angular/core';
2
3 @Component({
4   selector: 'todo-item',
5   template: `
6     <li class="item" [class.completed]="checked">
7       <input type="checkbox" [checked]="checked" (change)="handleOnCheck($event)">
8       <label>{{text}}</label>
9       <button class="destroy" (click)="handleOnRemove()>x</button>
10    </li>
11  `,
12  styles: []
13})
14export class TodoItem {
15  checked: boolean;
16  text: string;
17  index: number;
18  @Output() onTodoItemChecked = new EventEmitter<number>();
19  @Output() onTodoItemRemove = new EventEmitter<number>();
20
21  handleOnRemove = () => this.onTodoItemRemove.emit(this.index);
22  handleOnCheck = () => this.onTodoItemChecked.emit(this.index);
23}
```

StencilJS

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

@Component({
  tag: 'todo-item',
  styleUrl: 'todo-item.scss',
  shadow: true,
})
export class TodoItem {
  @Prop() checked: boolean;
  @Prop() text: string;
  @Prop() index: number;
  @Event() onTodoItemChecked: EventEmitter;
  @Event() onTodoItemRemove: EventEmitter;

  handleOnRemove = () => this.onTodoItemRemove.emit(this.index);
  handleOnChecked = () => this.onTodoItemChecked.emit(this.index);

  render() {
    return (
      <li class={this.checked ? 'completed' : ''}>
        <input type="checkbox" checked={this.checked} onChange={this.handleOnChecked}>
        <label>{this.text}</label>
        <button onClick={this.handleClickRemove}>X</button>
      </li>
    );
  }
}
```

SkateJS + Preact

```
1 // @jsx h
2 import { props } from "skatejs/dist/esnext";
3 import { h } from "preact";
4 import { Component } from "util";
5
6 export default class extends Component {
7   static events = ["check", "remove"];
8   static props = {
9     checked: props.boolean,
10    index: props.number
11  };
12
13  handleCheck = e => {
14    this.onCheck({ index: this.index, value: e.target.checked });
15  };
16  handleRemove = () => {
17    this.onRemove({ index: this.index });
18  };
19
20  render({ checked, handleCheck, handleRemove }) {
21    return (
22      <div>
23        <style>`<!--
24          <li class="checked" ?="completed" :="">
25            <input type="checkbox" checked={checked} onChange={handleCheck}>
26            <label>
27              <input type="checkbox" />
28            </label>
29            <button onClick={handleRemove}>x</button>
30          </li>
31        </style>
32      </div>
33    );
34  }
35}
```

SkateJS + lit-html

```
1 import { props } from "skatejs/dist/esnext";
2 import { html } from "lit-html/lib/util-extended";
3 import { Component } from "./util";
4
5 export default class extends Component {
6   static events = ["check", "remove"];
7   static props = {
8     checked: props.boolean,
9     index: props.number
10   };
11
12   handleCheck = e => {
13     this.onCheck({ index: this.index, value: e.target.checked });
14   };
15   handleRemove = () => {
16     this.onRemove({ index: this.index });
17   };
18
19   render({ checked, handleCheck, handleRemove }) {
20     return html`

21       
104      <div class="list-item">
105        <input type="checkbox" checked="${checked}" on-change="${handleCheck}" />
106        <label>
107          ${label}
108        </label>
109        <slot></slot>
110        </div>
111        <button on-click="${handleRemove}">x</button>
112      </div>
113

`;
114  }
115}
116
```

Pour démarrer



```
$ npm init stencil
```



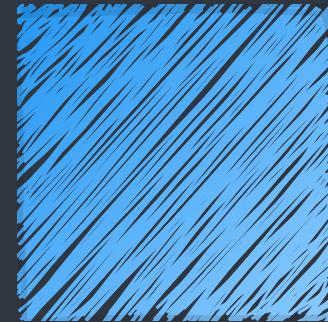
```
? Pick a starter > - Use arrow-keys. Return to submit.  
> ionic-pwa      Everything you need to build fast, production ready PWAs  
    app           Minimal starter for building a Stencil app or website  
    component     Collection of web components that can be used anywhere
```

```
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/>



Projet **Open Source**, ➔ BSD 3-Clause License

Créer par l'équipe **Polymer Team** en 2017

1.7k ★ sur github

Utilise la bibliothèque de template **lit-html**

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

Basé sur les **templates HTML**

Avec les ➔ *Template literals* de ES2015

```
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);
```



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

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);
```

```
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>! `;
    }
}
```

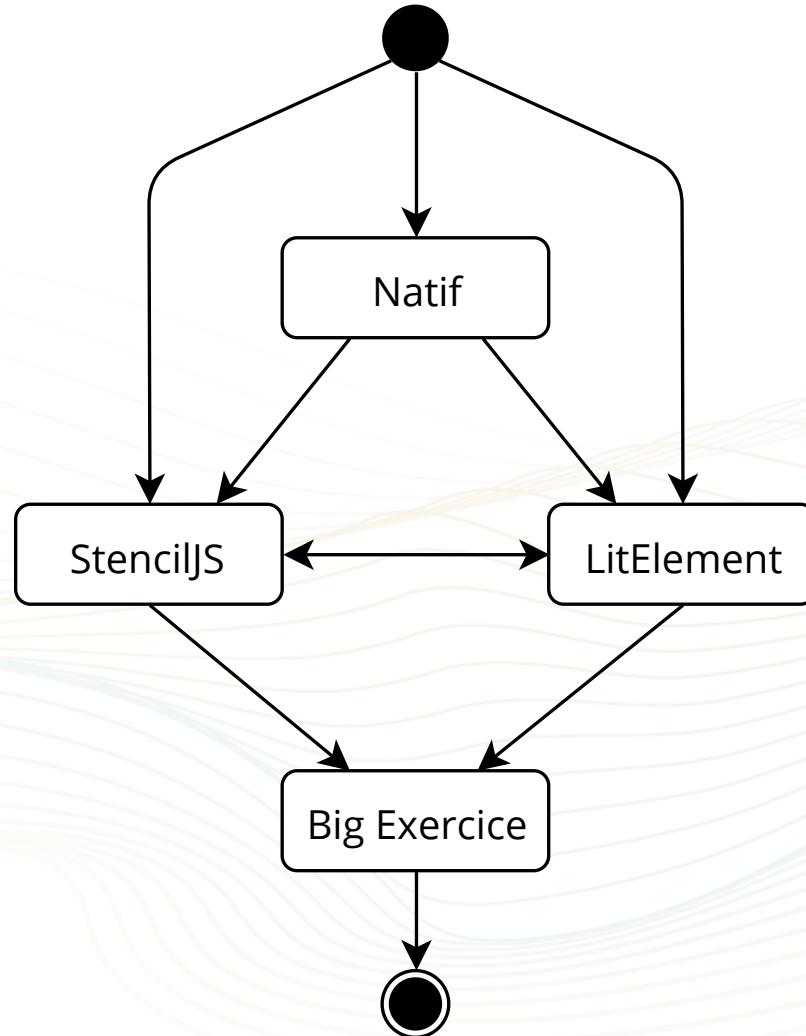
 Chrome, Safari, Opera, Firefox

 polyfills pour Edge et IE 11

Workshop

@julienrenaux @ilaborie #DevoxxFR #webcomponents

DEVOXX
France



Conclusion

@julienrenaux @ilaborie #DevoxxFR #webcomponents

DEVOXX
France

Construire une application

Style avec un thème

Support des navigateurs

Utiliser un gestionnaire d'état externe

Utilisez les *custom properties* CSS

Utiliser le polyfill ou Electron

TODO: ...

Alternatives modernes

➤ SkateJS

➤ Svelte

➤ Slim.js

...

Fin

Merci

Pensez à nous faire des retours (votez !)