

# Web Components:



Natif



StencilJS



LitElement





## Julien Renaux

GDE Web, Freelancer

@julienrenaux

<https://julienrenaux.fr/>



## Igor Laborie

Expert Web & Java

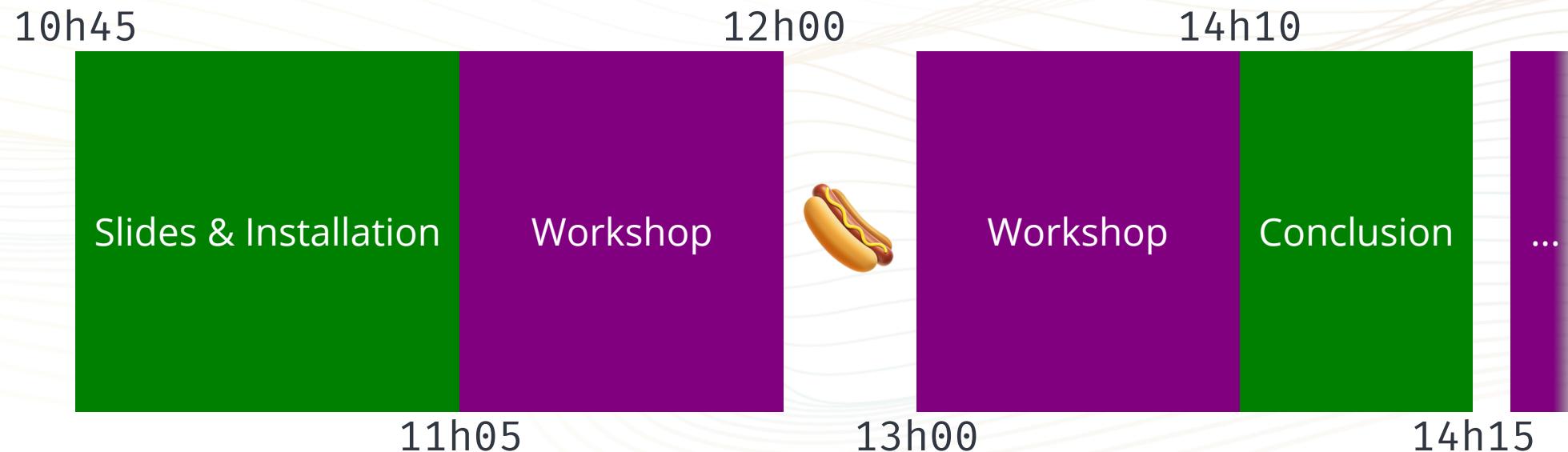
@ilaborie

[igor@monkeypatch.io](mailto:igor@monkeypatch.io)





# Roadmap

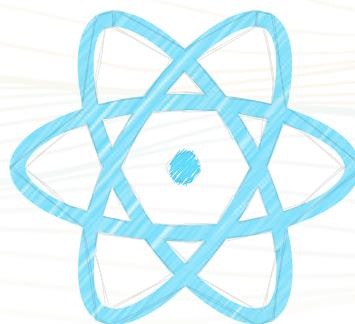


# Instructions

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 1,755 people talking about it. There are also icons for a blue bird (Twitter logo), a blue info circle, and a blue greater-than sign.

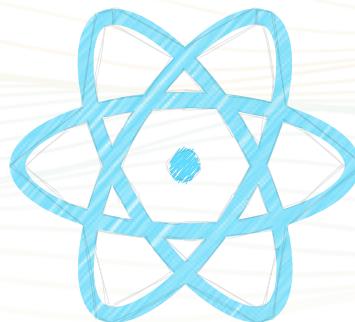
I Am Devloper  
@jamdevloper

I think I've had milk last longer than some JavaScript frameworks.

1,124 1:22 PM - Dec 4, 2014

1,755 people are talking about this >

# **...Interopérabilité ne vient plus gratuitement...**



React

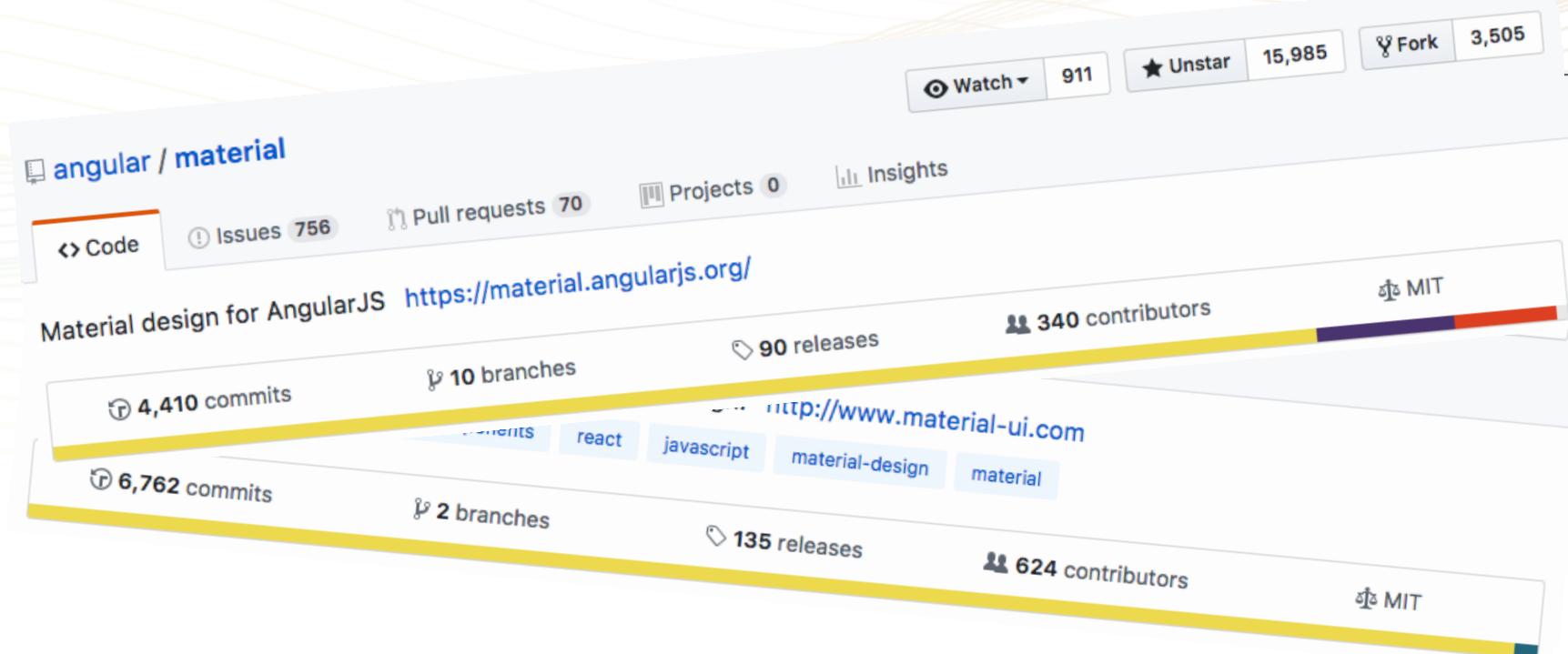


Vue.js



Angular

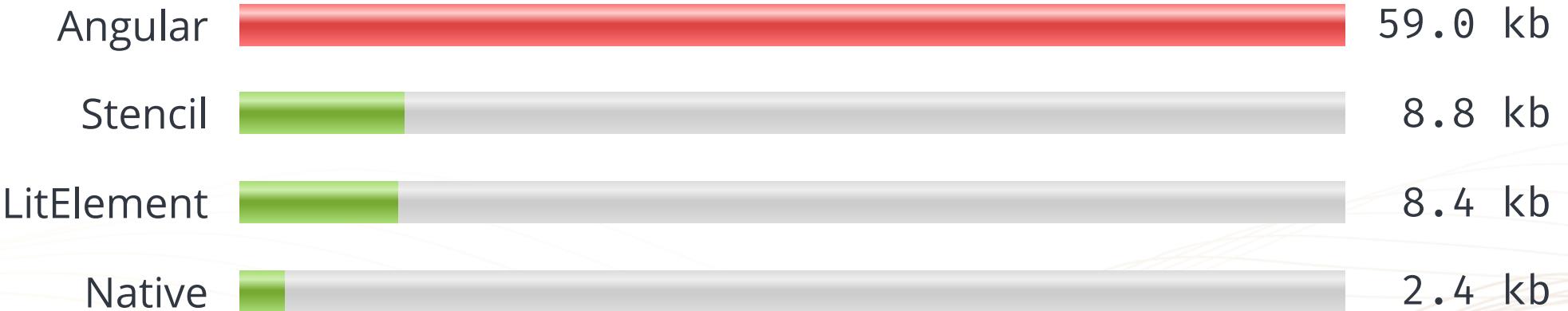
# ...et on réinvente sans arrêt la roue !



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

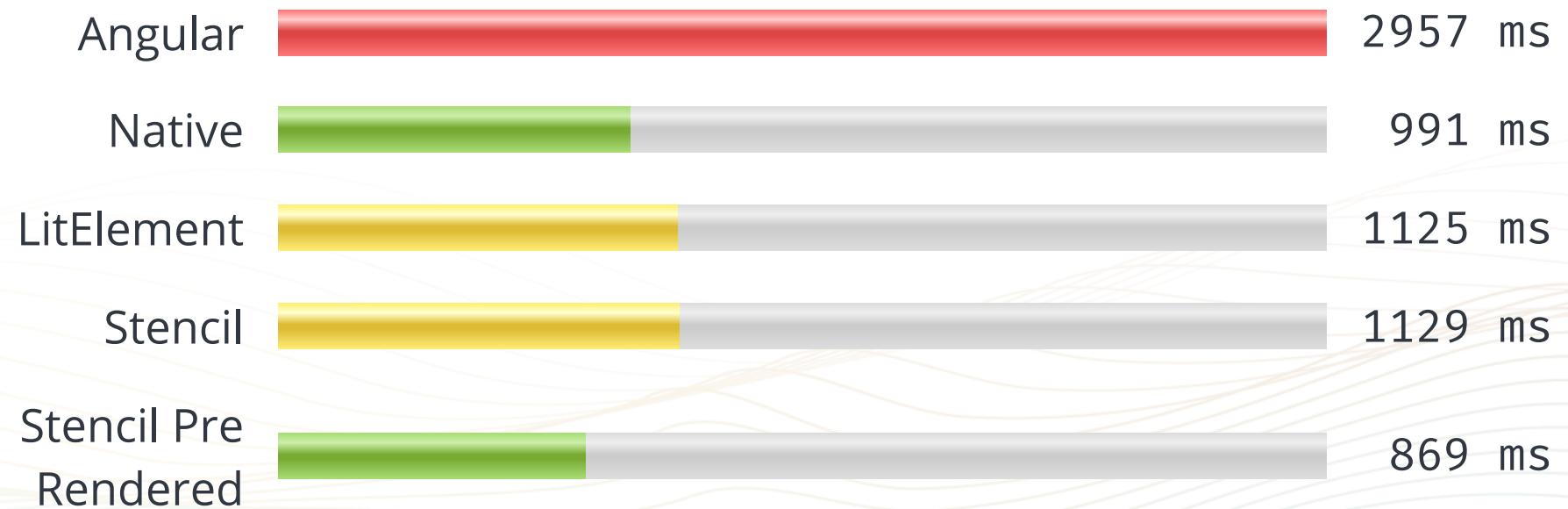


**Toute cette complexité doit  
s'arrêter avec les Web  
Components**



## 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



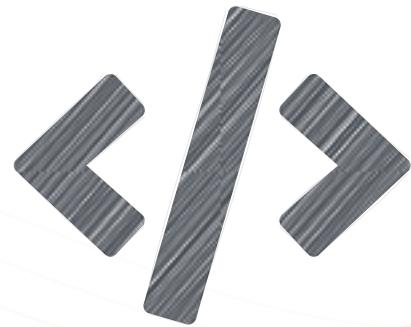
Shadow DOM



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+

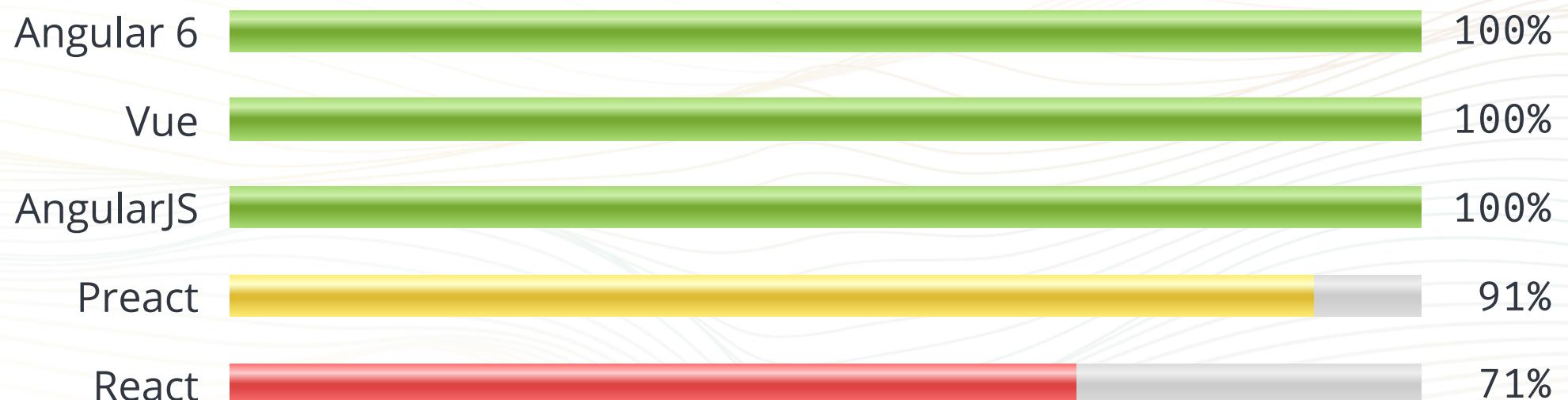


9+

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

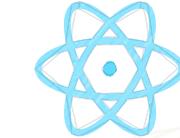
5.1k ⭐ sur github



**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
9   connectedCallback() {
10   this._root.innerHTML =
11     `<style>` +
12     `</style>` +
13     `<li class="item">` +
14       `<input type="checkbox">` +
15       `<label>${this._text}</label>` +
16       `<button class="destroy">x</button>` +
17     `</li>`;
18
19   this._site = this._root.querySelector('.item');
20   this._removeButton = this._root.querySelector('.de');
21   this._text = this._root.querySelector('label');
22   this._checkbox = this._root.querySelector('input');
23   this._removeButton.addEventListener('click', (e) => {
24     e.preventDefault();
25     this.dispatchEvent(new CustomEvent('onRemove', {
26       detail: this._index
27     }));
28   });
29   this._checkbox.addEventListener('click', (e) => {
30     e.preventDefault();
31     this.dispatchEvent(new CustomEvent('onToggle', {
32       detail: this._index
33     }));
34   });
35   this._render();
36 }
37
38 disconnectedCallback() []
39 static get observedAttributes() {
40   return ['text'];
41 }
42
43 attributeChangedCallback(name, oldValue, newValue) {
44   if (name === 'text') {
45     this._text = newValue;
46   }
47
48   set _index(value) {
49     this._index = value;
50   }
51
52   get _index() {
53     return this._index;
54   }
55
56   set checked(value) {
57     this._checked = Boolean(value);
58   }
59
60   get checked() {
61     return this.hasAttribute('checked');
62   }
63
64   _render() {
65     if (!this._item) return;
66     this._text.textContent = this._text;
67     if (this._checked) {
68       this._item.classList.add('completed');
69       this._checkbox.setAttribute('checked', '');
70     } else {
71       this._item.classList.remove('completed');
72       this._checkbox.removeAttribute('checked');
73     }
74   }
75 }
76
77 window.customElements.define('todo-item', TodoItem);
```

# Polymer 2

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

# Angular Elements

```
> 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)="
8         =handleOnCheck($event)">
9       <label>{{text}}</label>
10      <button class="destroy" (click)="handleOnRemove()"><x/></button>
11    </li>
12  `,
13  styles: [
14    '',
15  ],
16  encapsulation: ViewEncapsulation.Native
17})
18 export class TodoItem {
19   @Input() checked: boolean;
20   @Input() text: string;
21   @Input() index: number;
22   @Output() onTodoItemChecked = new EventEmitter<number>();
23   @Output() onTodoItemRemove = new EventEmitter<number>();
24
25   handleOnRemove = () => this.onTodoItemRemove.emit(this.index);
26   handleOnCheck = () => this.onTodoItemChecked.emit(this.index);
27 }
```

# 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.handleOnRemove}>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       <div>
24         <style>`-
25           </style>
26         <li class={checked ? "completed" : ""}>
27           <input type="checkbox" checked={checked} onChange={handleCheck}>
28             <slot />
29           </label>
30           <button onClick={handleRemove}>x</button>
31         </li>
32       </div>
33     </div>
34   );
35 }
36
```

# 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       <style>-
22       </style>
23       <li class="${checked ? "completed" : ""}>
24         <input type="checkbox" checked="${checked}" on-change="${handleCheck}" />
25         <label>
26           <slot></slot>
27         </label>
28         <button on-click="${handleRemove}">x</button>
29       </li>
30

`;
31   }
32 }
```

# Pour démarrer



```
$ npm init stencil
```



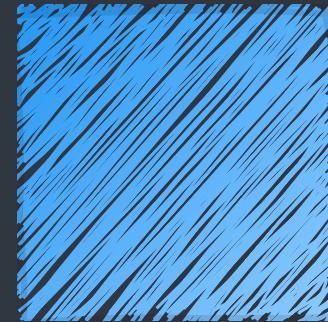
```
? Pick a starter > - Use arrow-keys. Return to submit.  
> ionic-pwa      Everything you need to build fast, production ready PWAs  
    app           Minimal starter for building a Stencil app or website  
    component     Collection of web components that can be used anywhere
```

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

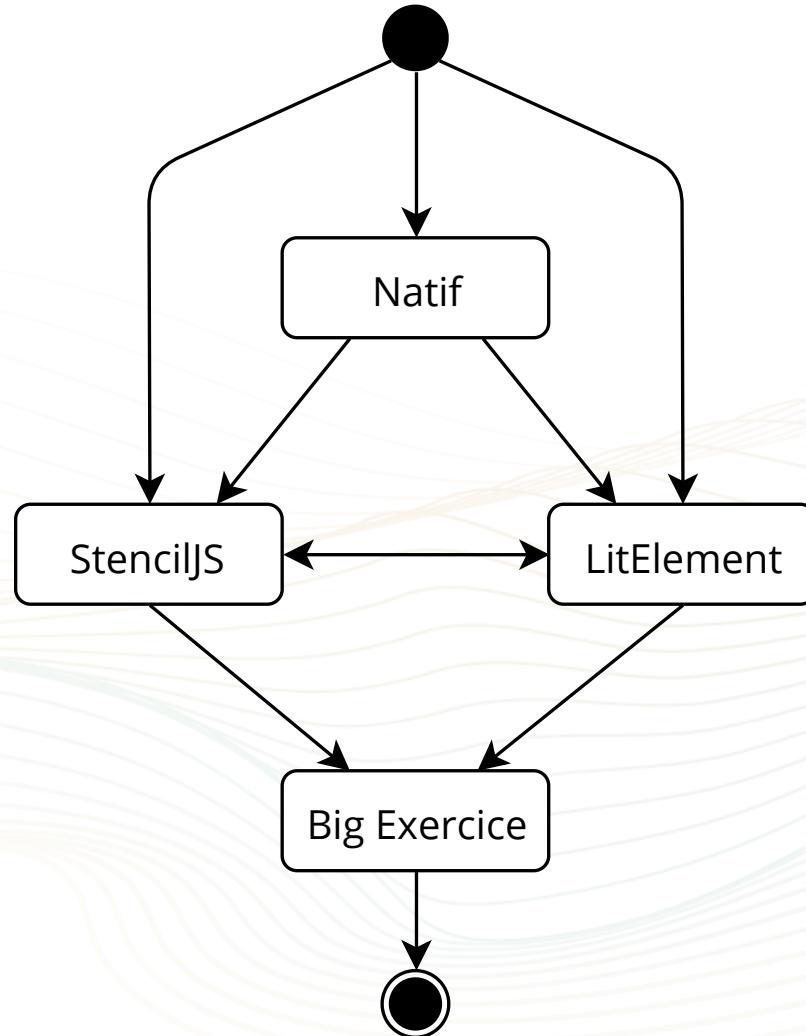
 Chrome, Safari, Opera, Firefox

 polyfills pour Edge et IE 11

# Workshop



@julienrenaux @ilaborie #DevoxxFR #webcomponents



# 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 !)