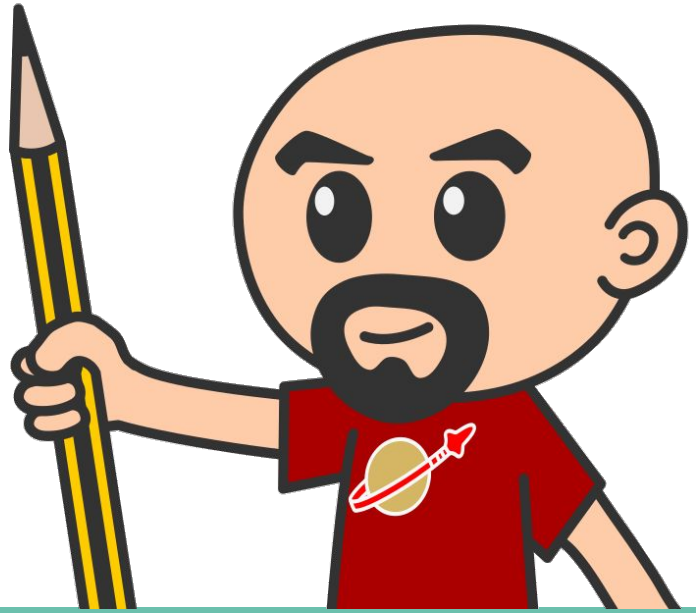
The Web Components Interoperability Challenge

Horacio Gonzalez
@LostInBrittany

Horacio Gonzalez

@LostInBrittany

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



OVH
Team DevRel

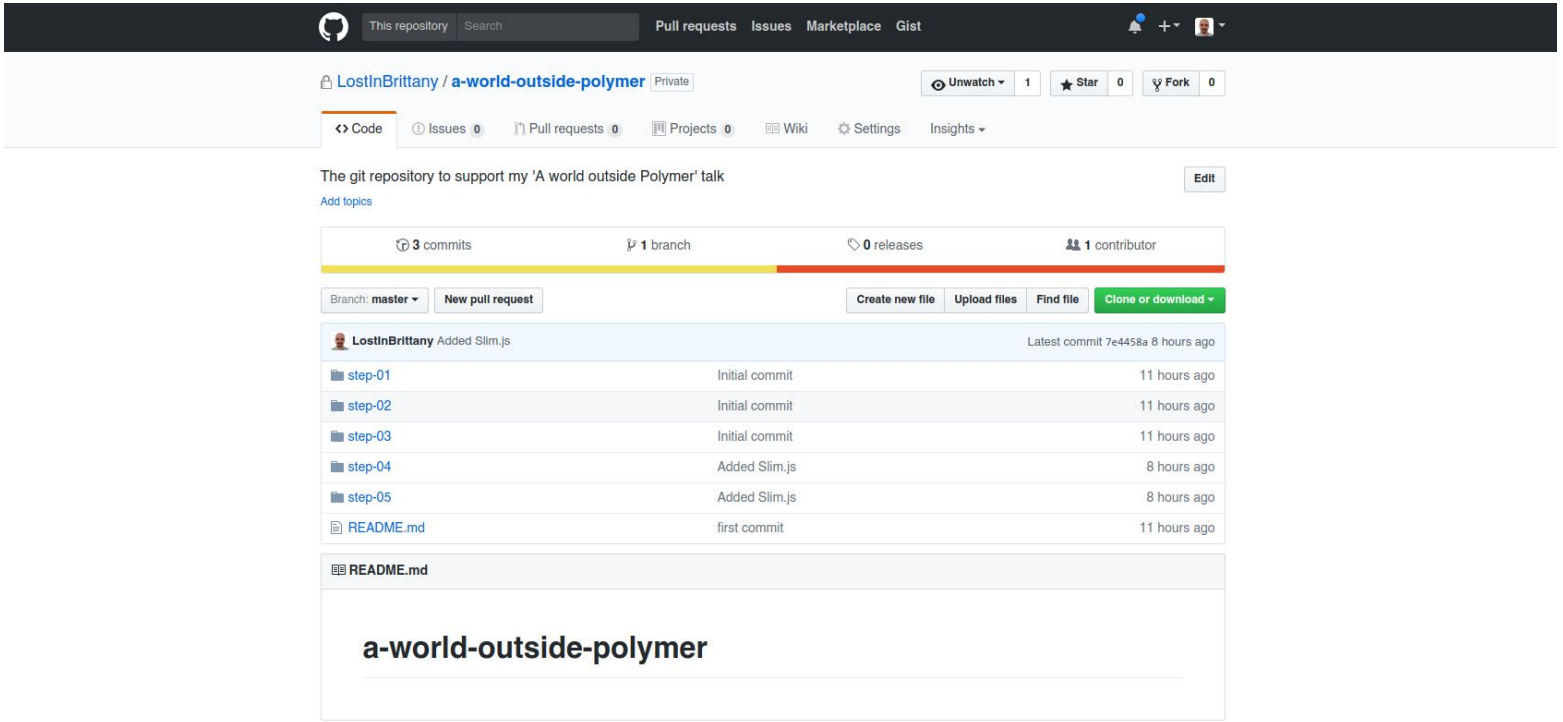
Breizh C@mp
Mix de technologies

Finist Devs

WARP 10 MEETUP

Google Developers Experts 2015
Web Technologies GDE

We want the code!



https://github.com/LostInBrittany/web-components-interop

Web Components

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, TourraineTech!';
 }

 // 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

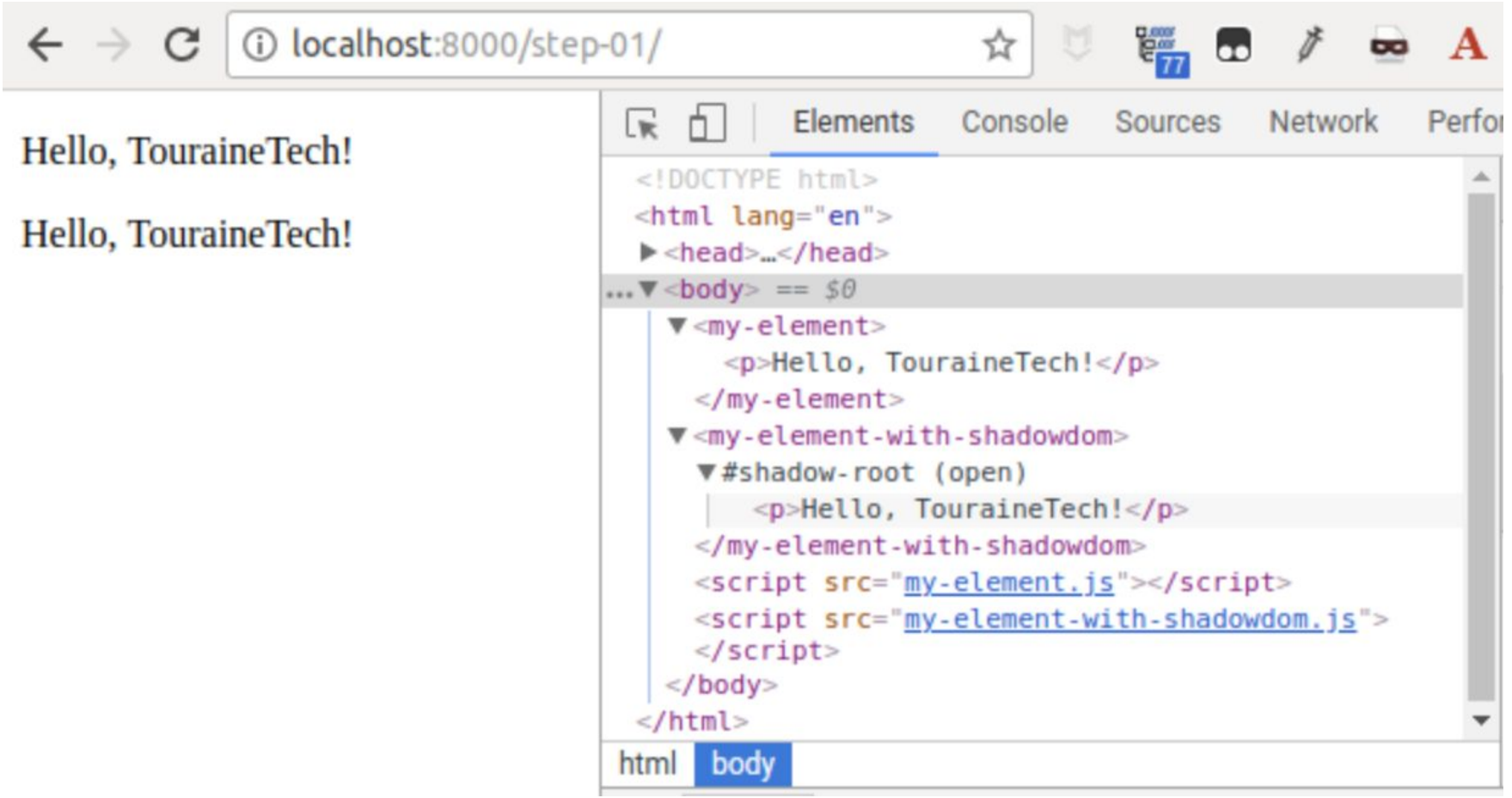
Custom Elements don’t:

- 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, RennesJS!';
   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



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 observer 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();
    this.display();
  }

  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;
}
```


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

42

Polymer

Adding syntactic sugar to the standard

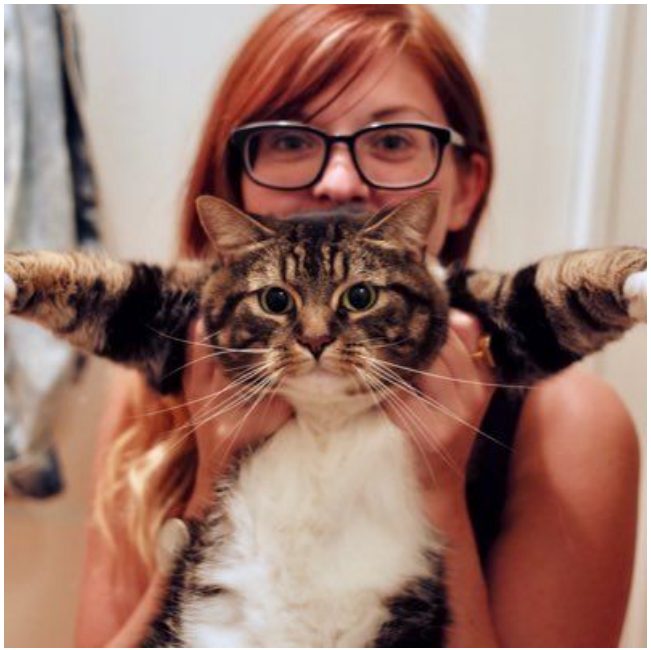
Everything is better with sugar

```
<link rel="import" href="./bower_components/polymer/polymer.html">

<dom-module id="my-polymer-counter">
 <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>
 </template>
```


Everything is better with sugar

```
<script>
  class MyPolymerCounter extends Polymer.Element {
    static get is() { return 'my-polymer-counter'; }
    static get properties() {
      return {
        counter: { type: Number, reflectToAttribute:true, value: 0 }
      }
    }
    increment() {
      this.counter = Number.parseInt(this.counter) + 1;
    }
  }

  customElements.define('my-polymer-counter', MyPolymerCounter);
</script>
</dom-module>
```

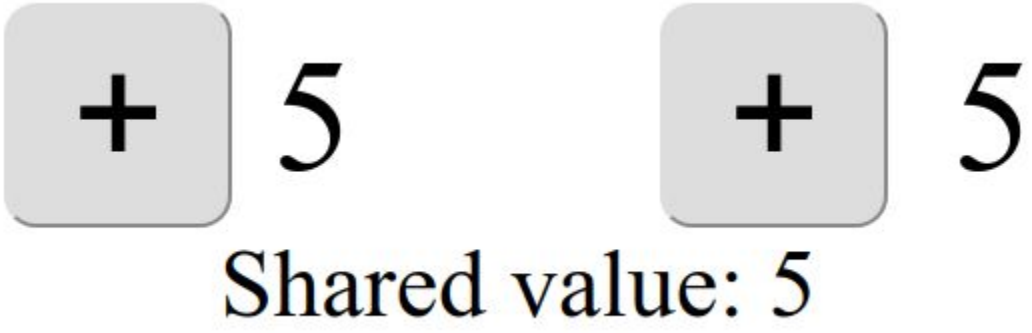

Everything is better with sugar



Polymer is like jQuery for Web components

@notwaldorf

But they are still custom elements



100% interoperables

Interoperation pattern

```
<div class="container">
    <my-polymer-counter
        counter="[[value]]"
        on-counter-changed="_onCounterChanged"></my-polymer-counter>
    <my-counter
        counter="[[value]]"
        on-counter-changed="_onCounterChanged"></my-counter>
</div>
```

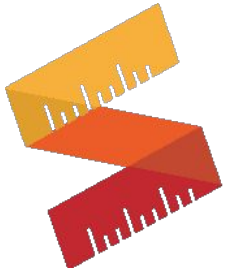
Attributes for data in

Events for data out

To infinity and beyond!

There is a world outside Polymer

Lots of web components libraries



slim.js

STENCIL

For different need and sensibilities

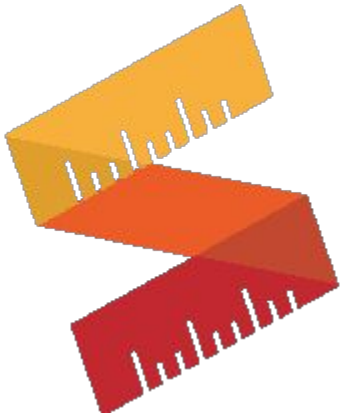
Lots of web components libraries

Angular Elements

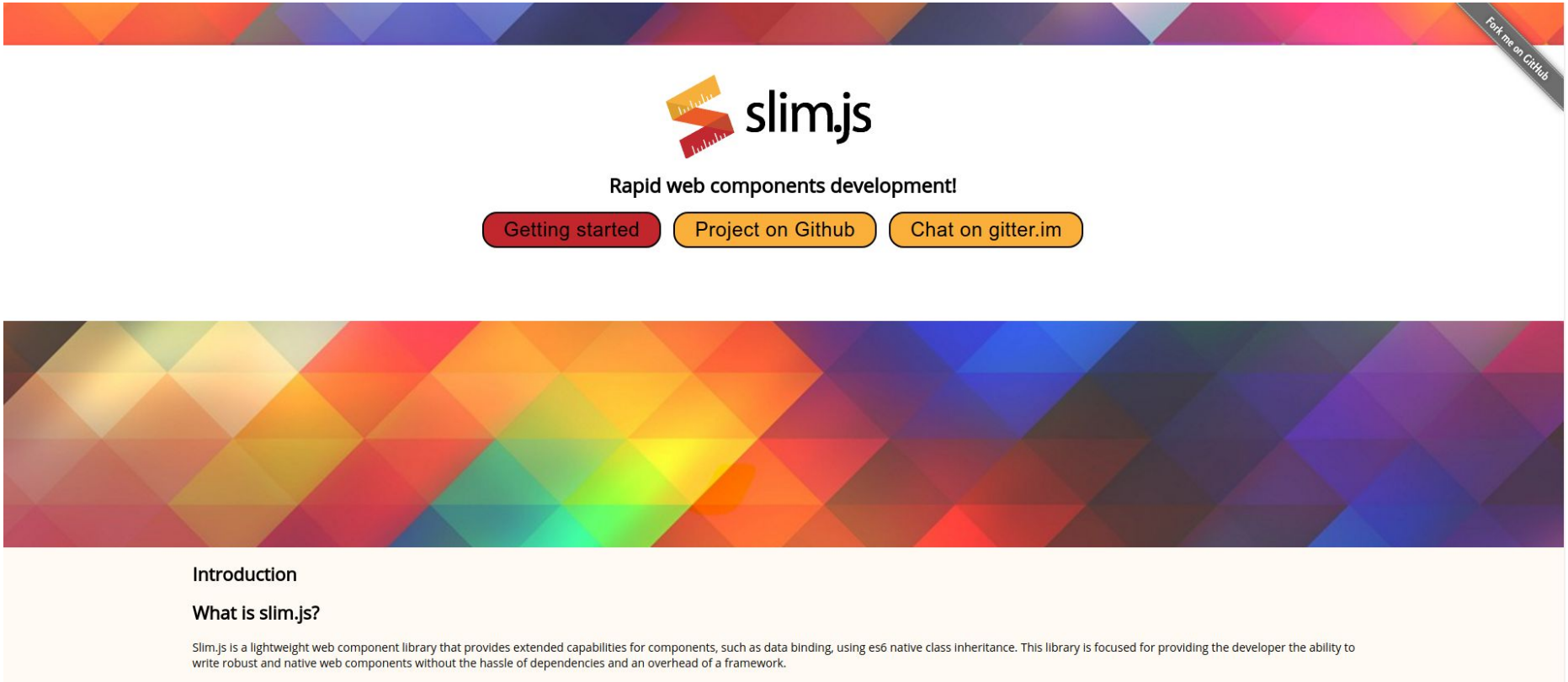
Vue Web Component Wrapper

And the frameworks work on it too!

Slim.js



Slim.js


Fork me on GitHub
slim.js
Rapid web components development!
Getting started
Project on Github
Chat on gitter.im
Introduction
What is slim.js?
Slim.js is a lightweight web component library that provides extended capabilities for components, such as data binding, using es6 native class inheritance. This library is focused for providing the developer the ability to write robust and native web components without the hassle of dependencies and an overhead of a framework.

Slim.js

- Lightweight web component library
- Extended capabilities for components
  - data binding
- Using es6 native class inheritance
- Without Shadow DOM

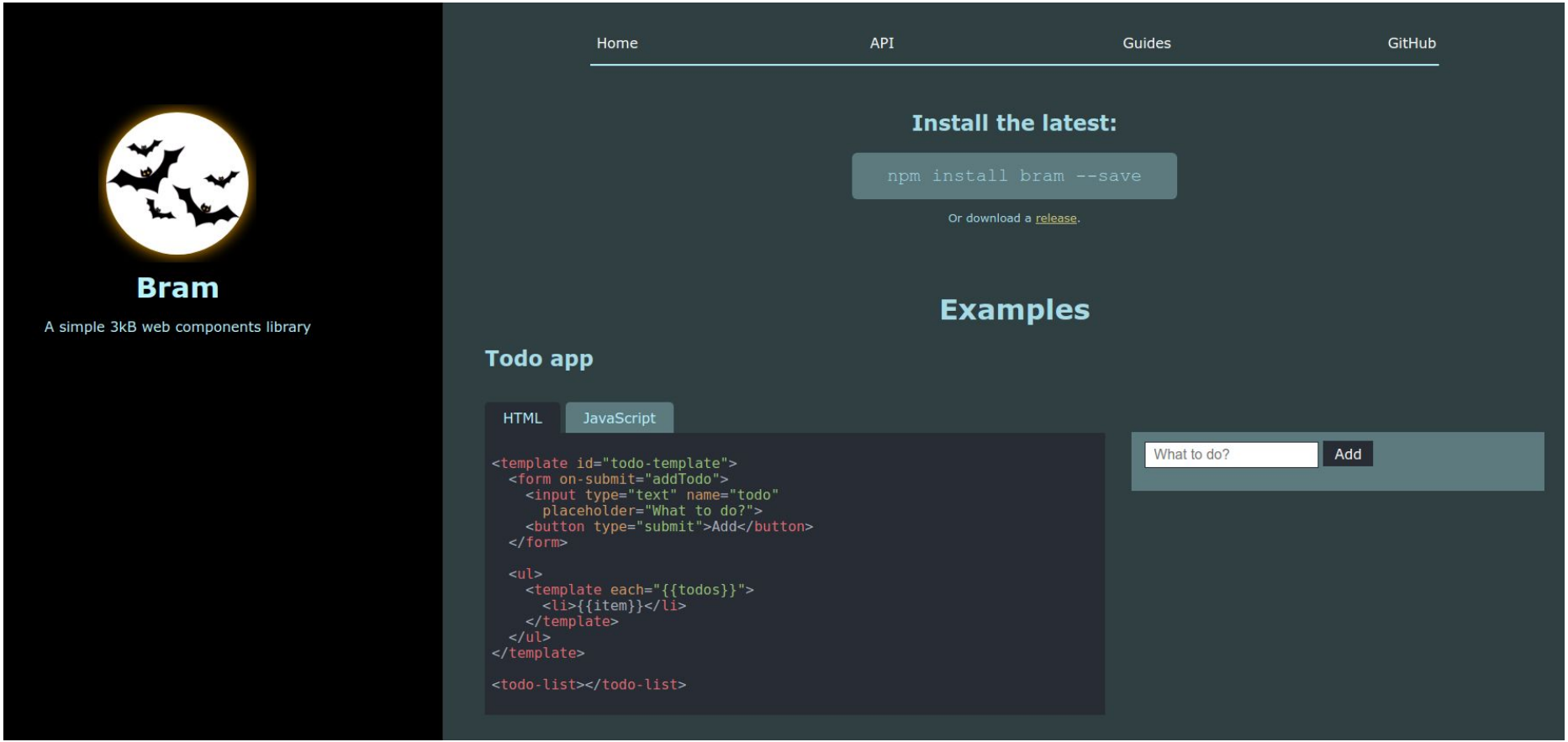
Like a lighter and lesser-featured Polymer

Slim.js

```
Slim.tag('my-slim-counter', `
  <style> [...] </style>
  <div class="container">
    <div class="button"  slim-id="button"> <img src="./img/slim.png"> </div>
    <div class="value" bind> [[counter]] </div>
  </div>`,
  class extends Slim {
    onCreated() {
      if (this.counter == undefined) {
        this.counter = Number.parseInt(this.getAttribute('counter'))||0;
      }
      this.button.onclick = () => {
        this.counter++;
        this.dispatchEvent(new CustomEvent('counter-changed', {detail: {counter: this.counter}}));
      }
    }
  });
```


Bram.js

Bram.js


Bram
A simple 3kB web components library
Home
API
Guides
GitHub
Install the latest:
npm install bram --save
Or download a release.
Examples
Todo app
HTML
JavaScript
<template id="todo-template">
<form on-submit="addTodo">
<input type="text" name="todo"
placeholder="What to do?">
<button type="submit">Add</button>
</form>

<template each="{{todos}}">
{{item}}
</template>

</template>
<todo-list></todo-list>
What to do?
Add

Bram.js

- Lightweight web component library
- Extended capabilities for components
  - data binding
- Using es6 native class inheritance
- With Shadow DOM (optional)

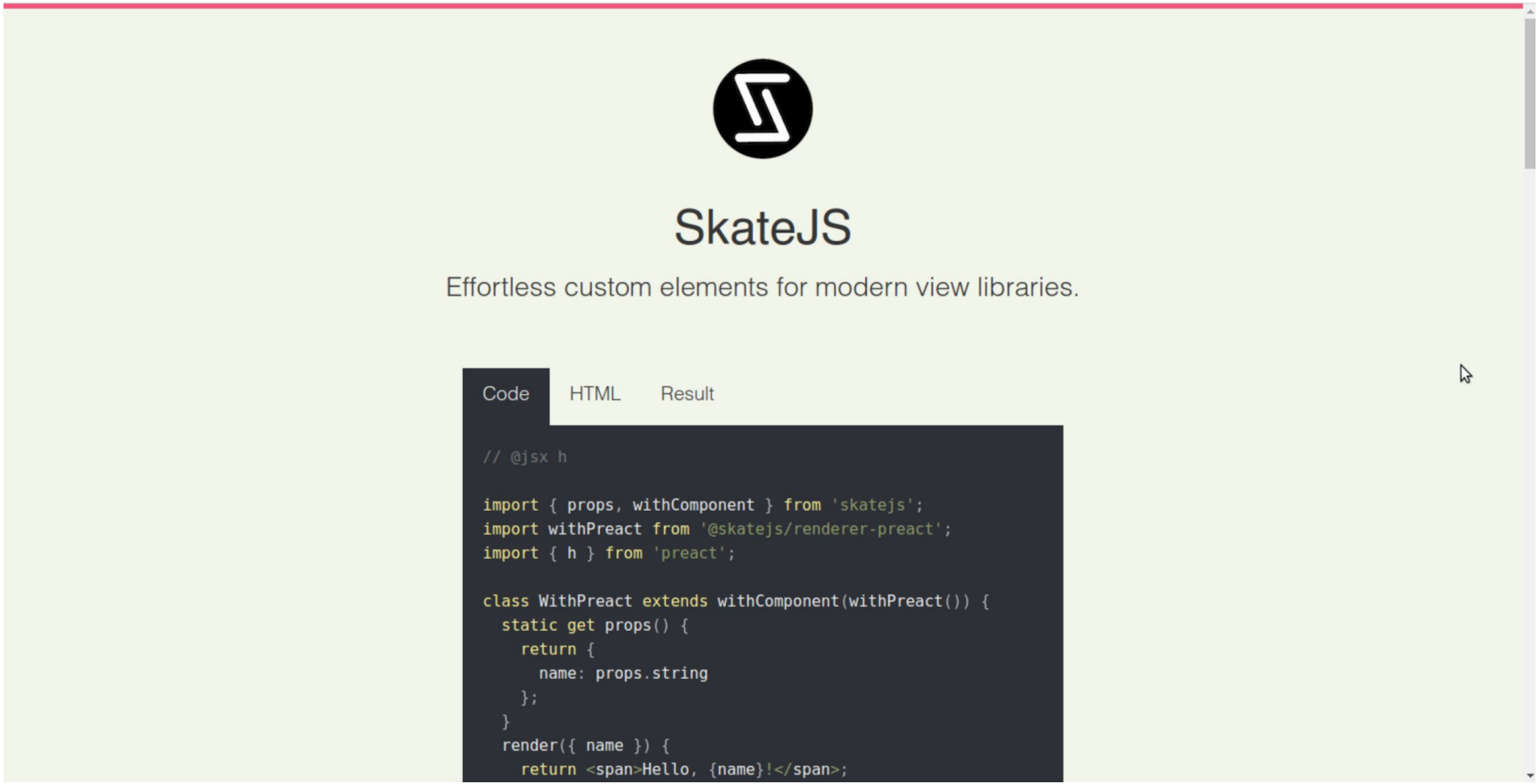
Like a lighter and lesser-featured Polymer, with Shadow DOM

Bram.js

```
let template=`
   <style> [...] </style>
   <div class="container">
       <div class="button" on-click="increase">  <img src="./img/bram.png">  </div>
       <div class="value" > {{counter}} </div>
   </div>`;

class MyBramCounter extends Bram(HTMLElement) {
  static get template() {
    let t = document.createElement('template');
    t.innerHTML = template;
    return t;
  }
  static get events() { return ['counter-changed']; }
  constructor() {
    super();
    this.model.counter = this.getAttribute('counter') || 0;
  }
  static get observedProperties() { return [ 'counter' ] }  //Non documented
  increase() {
    this.model.counter++;
    this.dispatchEvent(new CustomEvent('counter-changed', {detail: {counter: this.model.counter}}));
  }
}
```


Skatejs

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 Hello, {name}!;

Skatejs

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

Nice if you dislike declarative syntax and DOM...

Skatejs

```
import { props, withComponent } from '/node_modules/skatejs/dist/esnext/index.js';

class MySkateCounter extends withComponent() {

 constructor() {
   super();
   this.counter = this.counter || 0;
   this.addEventListener('click', e => this.increment());
 }

 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.string({ attribute: true })
   };
 }

 render({ counter }) {
   return `${this.style()}
     <div class="container">
       <div class="button"><img src="./img/skate.png"></div>
       <div class="value">${counter}</div>
     </div>`;
 }

 increment() {
   this.counter = Number.parseInt(this.counter) + 1;
   this.dispatchEvent(new CustomEvent('counter-changed', {detail: {counter: this.counter}}));
 }
 style() { return ` {...} `;
 }
}
```

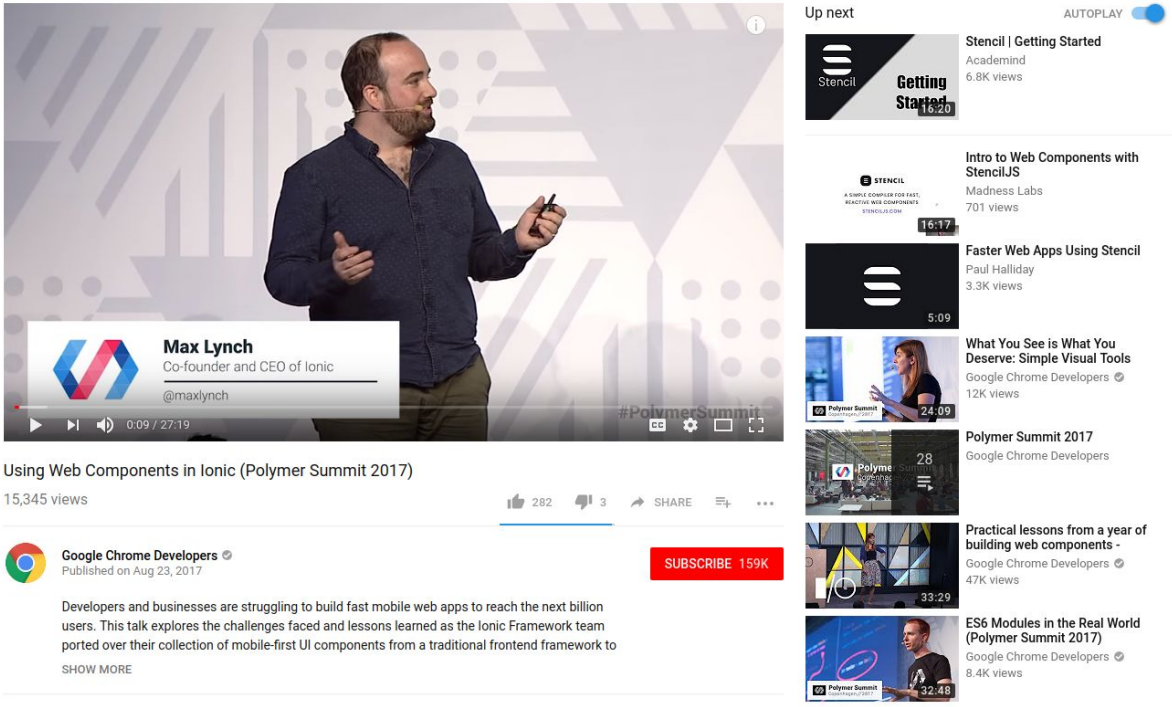

STENCIL

A new breed of Web Components

Next generation Ionic

Ionic 4 will be fully based on web components using a new toolkit: Stencil

New kid on the block



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

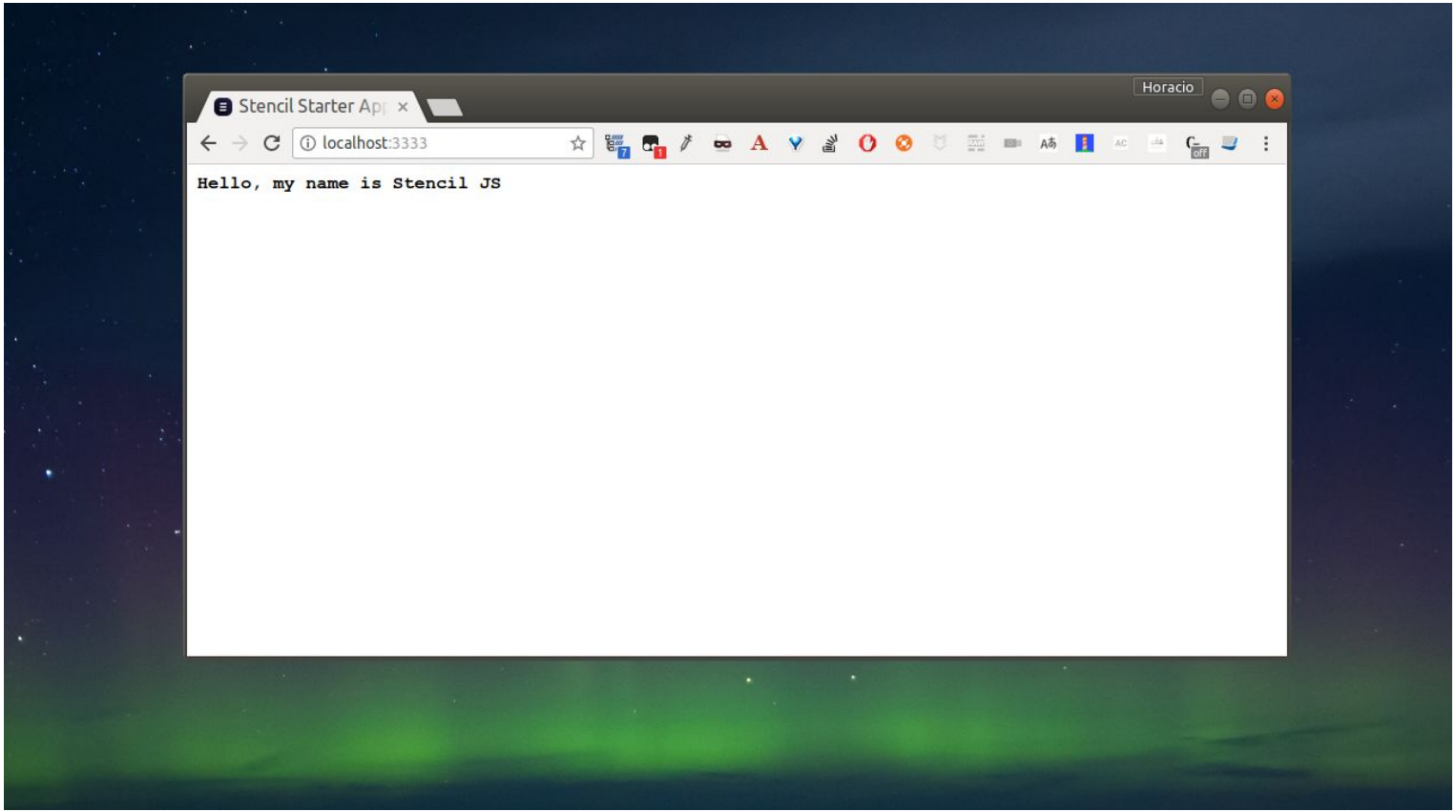
Clone the starter project

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

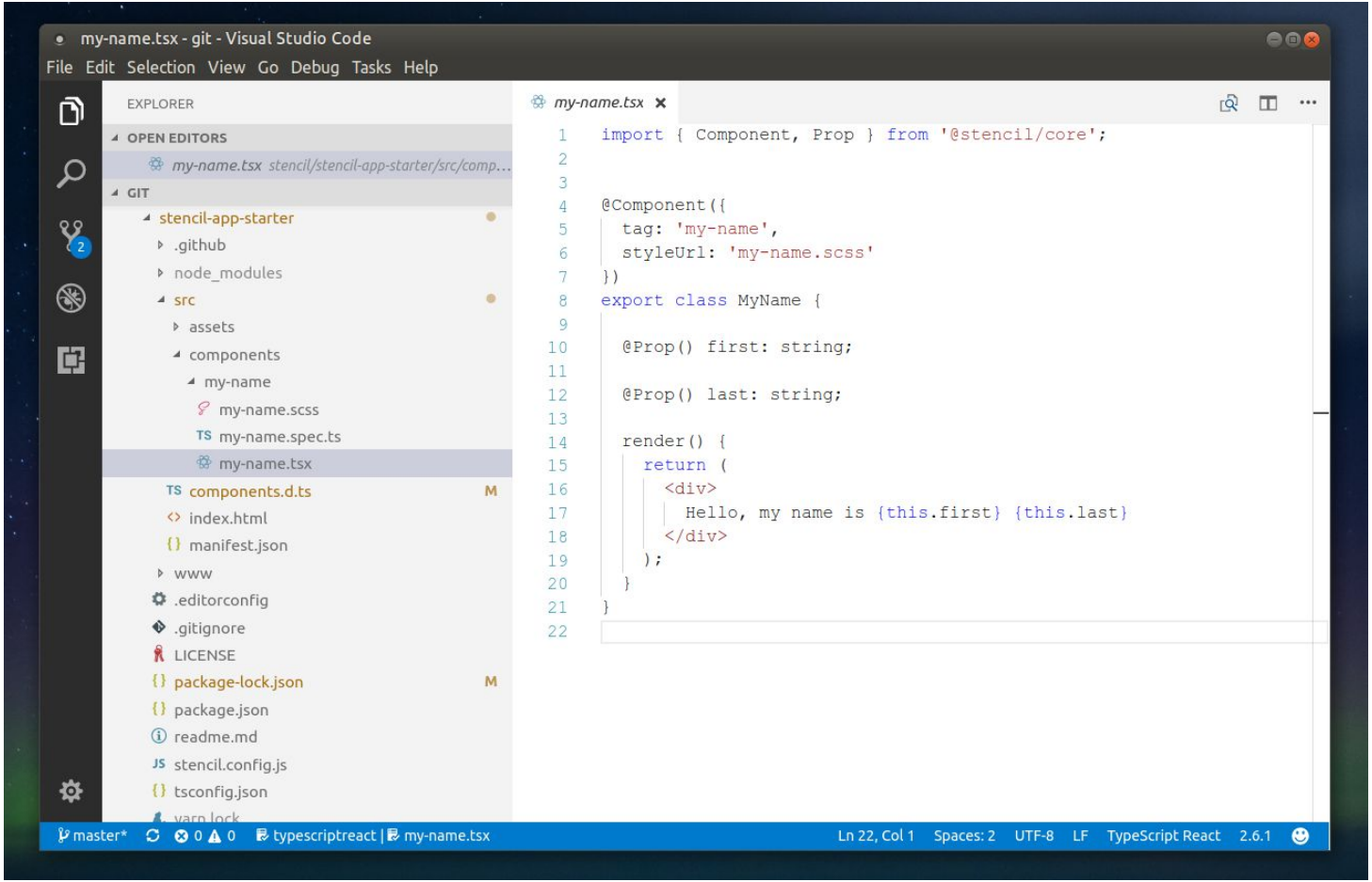
Start a live-reload server

```
npm start
```


Hands on Stencil



Hands on Stencil



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',
  styleUrl: '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;
 }

 componentWillLoad() {
   this.currentCount = this.counter;
 }

 increase() {
   this.currentCount++;
   this.currentCountChanged.emit({ counter: this.currentCount });
 }

 render() {
   return (
     <div class="container">
       <div class="button" onClick={() => this.increase()}> <img src="./img/stencil.png" /> </div>
       <div class="value" > {this.currentCount} </div>
     </div>
   );
 }
}
```


Conclusion

That's all folks!

Thank you!

рахмат
danke
謝謝
spas
ngiyabonga
شكراً جزيلاً
Баярлалаа
mersi
kia ora
barka
welalin
tack
teşekkür ederim
mahalo
tapadh leat
спасибо
faafetai lava
vinaka
спасибі
blagodaram
dank je
misaotra
matondo
paldies
grazzi
thank you
gracias
хвала
kiitos
dankie
nanni
nandri
dhanyavad
asante
manana
obrigada
tenki
bedankt
bayarlalaa
gràcie
hvala
maururu
köszönöm
akun
dankon
aćiū
djiere dieuf
tau
mochchakkeram
murakoze
enkosi
dziękuję
chnorakaloutioun
gratias ago
gràcies
sulpáy
дякую
mamnun
chokrane
taiku
go raibh maith agat
sobodi
děkuji
sagolun
sukriya
kop khun krap
obrigado
mèsi
didi madloba
najis tuke
ありがとう
tanemirt
grazie
arigatô
takk
dakujem
trugarez
kam sah hamnida
terima kasih
rahmet
ευχαριστώ
diolch
dhanyavadagalu
shukriya
merce
мерси
rahmat
তোমাকে ধন্যবাদ
감사합니다
xièxie
merci