open south code

Componiendo
Web Components

METADEV

https://metadev.pro

Pedro J. Molina

http://pjmolina.com

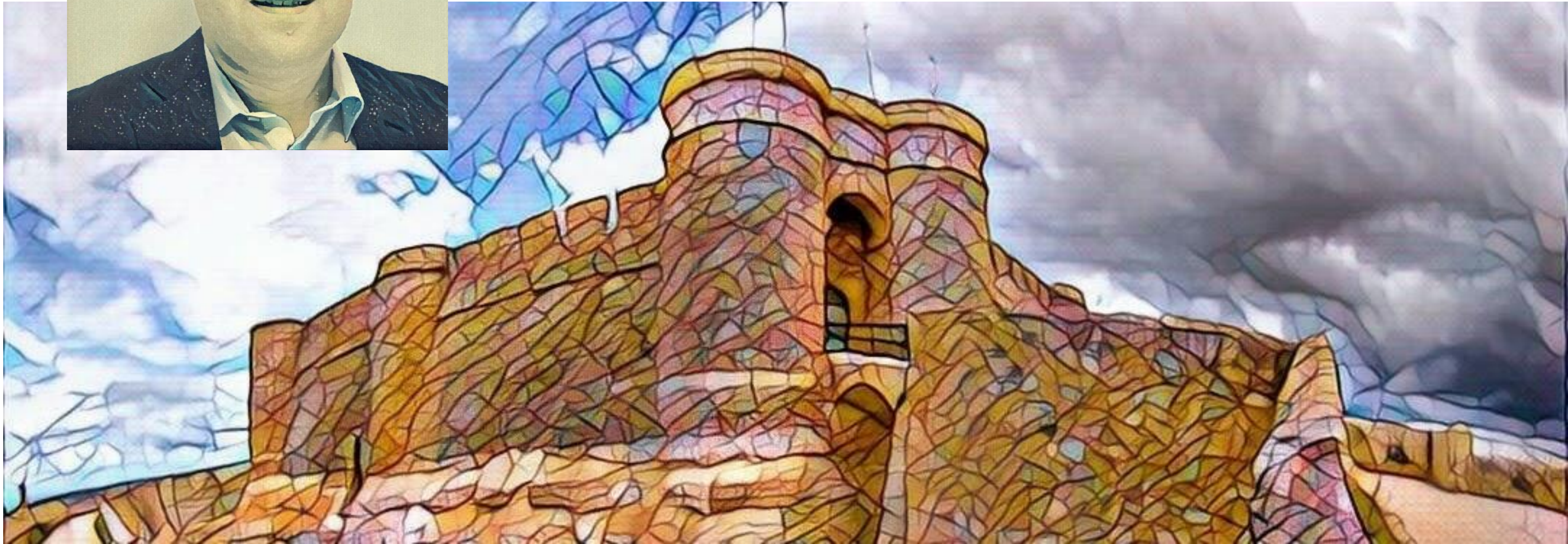
@pmolinam

#opensouthcode

Pedro J. Molina

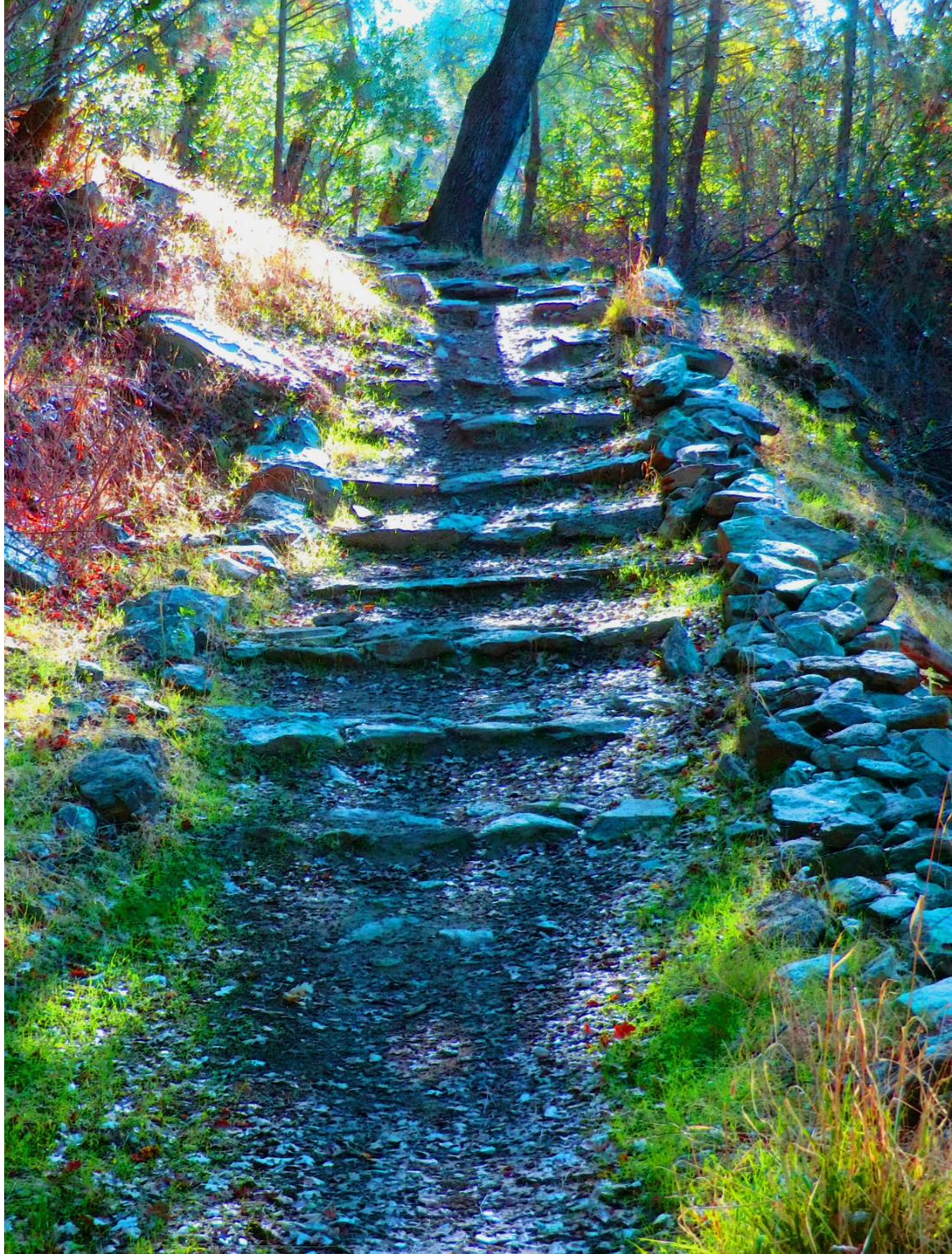
@pmolinam

METADEV



Agenda

- IU: Arqueología
- Web Components
- Estandarización
- Frameworks
- Catalogo de componentes
- Componiendo Web Comp.
- Impedimentos



Interfaz de Usuario: un poco de Arqueología

Cliente

Servidor

SPA / JS

ASP.NET JSP Ruby

Silverlight / Flash / Applets

Clientes pesados (.NET, Java Swing)

PHP CGI

Visual Basic / Delphi

Mainframe / Terminales VT52/VT100

Interfaz de Usuario: un poco de Arqueología

- Arquitecturas

  Model View Controller (Smalltalk ’80)

  Model View Presenter (IBM ’90)

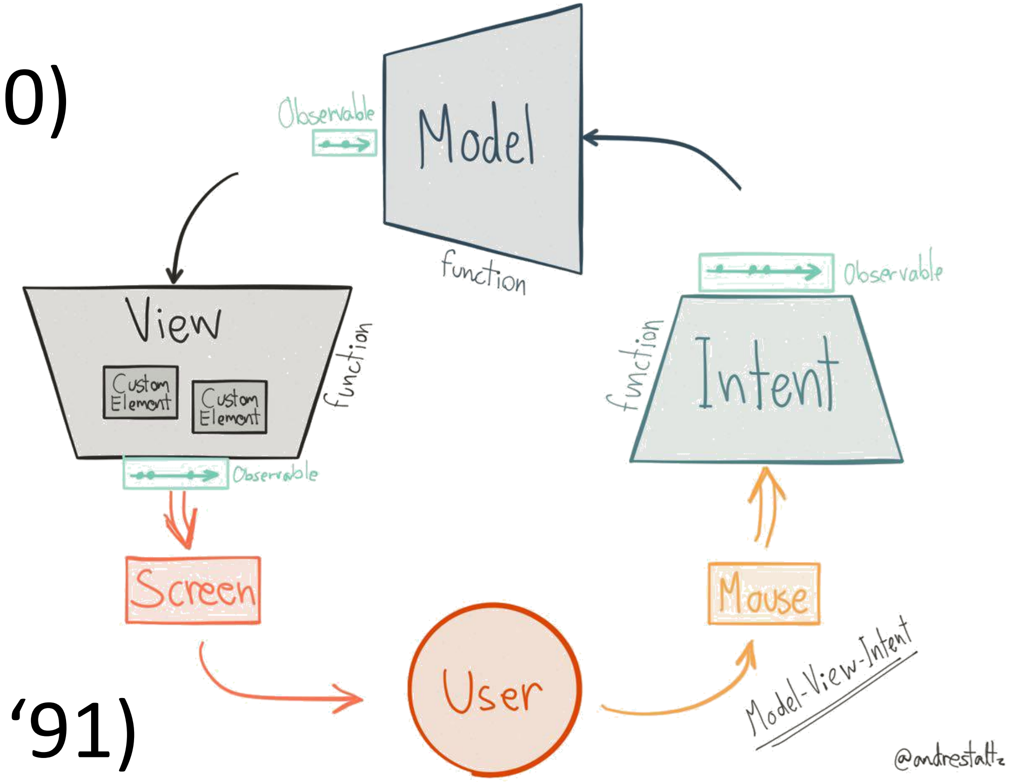
  Model View View-Model (MS ‘99)

  Reactivas (ReactJS)

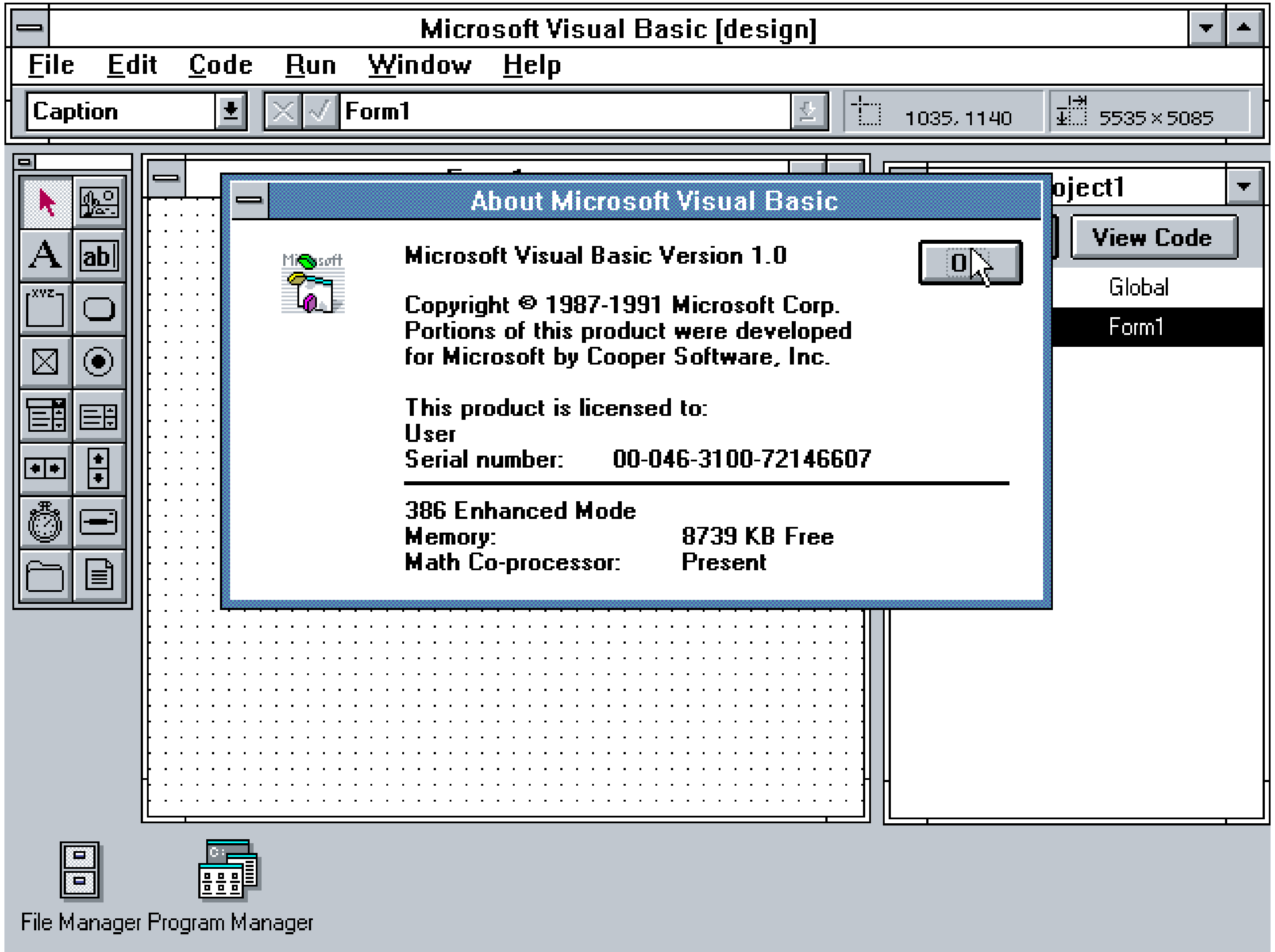
  Unidireccionales (CycleJS)

  Model View Update (Eml)

  Orientadas a Componentes (VB 1.0 ‘91)



Interfaz de Usuario: un poco de Arqueología



About Microsoft Visual Basic

Microsoft Visual Basic Version 1.0

Copyright © 1987-1991 Microsoft Corp.
Portions of this product were developed
for Microsoft by Cooper Software, Inc.

This product is licensed to:
User
Serial number: 00-046-3100-72146607

386 Enhanced Mode
Memory: 8739 KB Free
Math Co-processor: Present

Visual Basic 1.0, 1991

Sobre Windows 3.11

Alan Cooper para Microsoft

- Componentes
- Propiedades
- Eventos

- Paleta de componentes reutilizable

Web Components ¿Qué son?

- Componentes
- Propiedades
- Eventos
- Paleta de componentes reutilizable

El modelo de Visual Basic y Delphi en la Web,

¡27 años después!

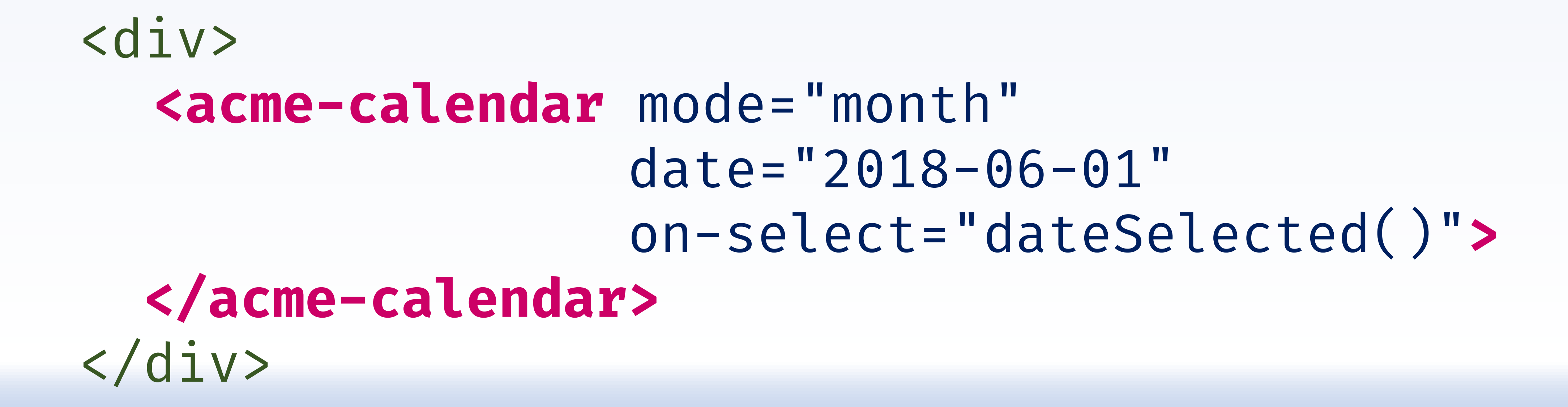
Web Components. Estándares base

1. Custom Elements
2. HTML Templates
3. Shadow DOM

~~4. HTML Imports~~ ES Modules

I. Custom Elements

- La posibilidad de extender el lenguaje HTML con elementos propios



```
<div>
  <acme-calendar mode="month"
                 date="2018-06-01"
                 on-select="dateSelected()">
  </acme-calendar>
</div>
```

En estandarización por la **W3C**

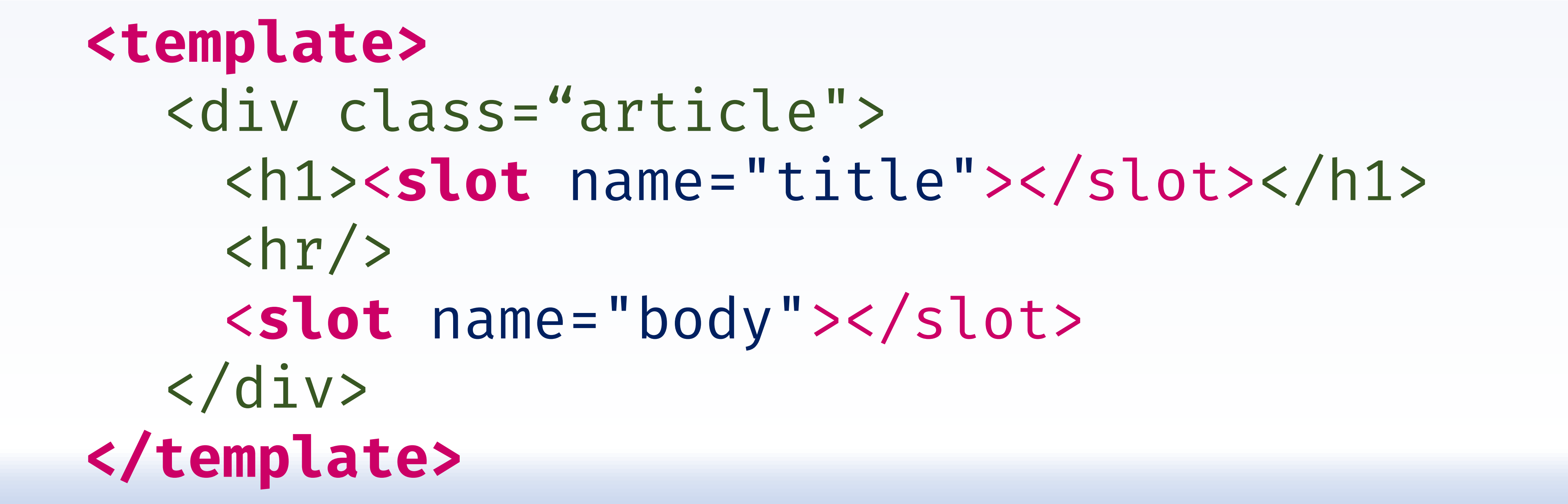
https://html.spec.whatwg.org/multipage/custom-elements.html#custom-elements

I. Custom Elements. Ejemplo

```
const templateCalendar = document.createElement('template');
templateCalendar.innerHTML = `
 <h1>Calendar</h1>
 <table> …  </table>
`;
class AcmeCalendar extends HTMLElement {
    constructor() {
        super();
    }
    connectedCallback() {}
    disconnectedCallback() { }
    _render() {}
}
window.customElements.define(‘acme-calendar', AcmeCalendar);
```

2. HTML Templates

Plantillas dentro de HTML



```
<template>
  <div class=“article">
    <h1><slot name="title"></slot></h1>
    <hr/>
    <slot name="body"></slot>
  </div>
</template>
```

En estandarización por la **W3C**

https://html.spec.whatwg.org/multipage/scripting.html#the-template-element/

3. Shadow DOM

El DOM dentro de cada elemento del DOM

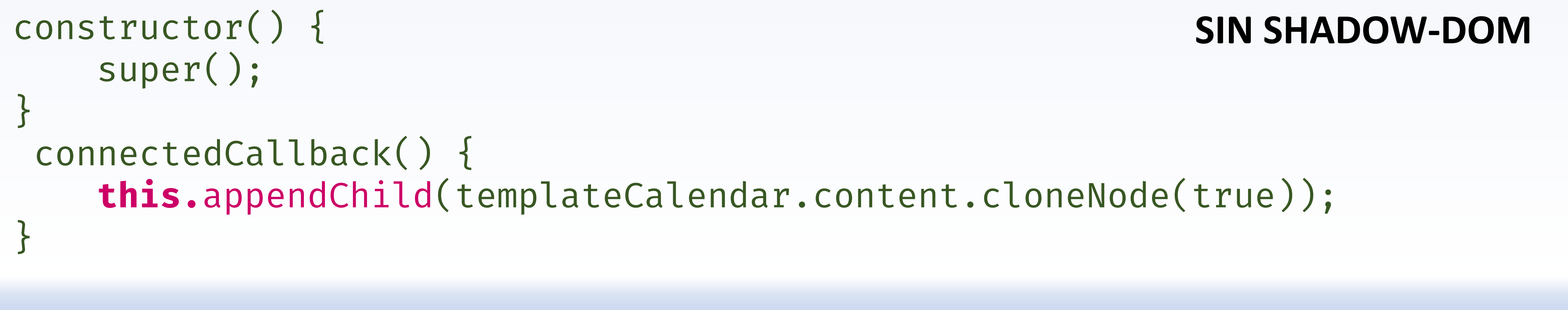
Proporciona:

- Aislamiento (ámbitos) para código y estilos (CSS)
- Seguridad (encarcelar Javascript)

En estandarización por la **W3C**
https://w3c.github.io/webcomponents/spec/shadow/

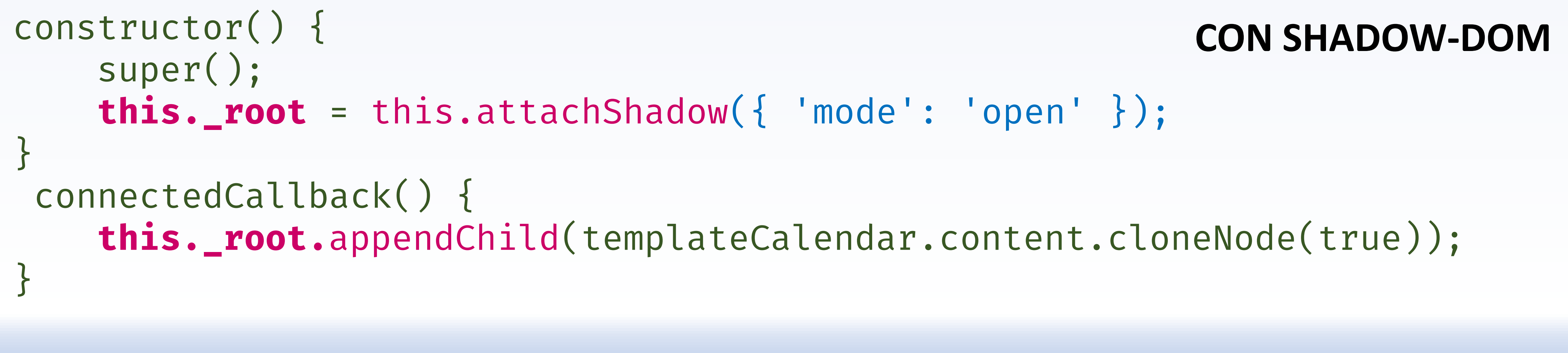
3. Shadow DOM. Ejemplo

SIN SHADOW-DOM



```
constructor() {
    super();
}
 connectedCallback() {
    this.appendChild(templateCalendar.content.cloneNode(true));
}
```

CON SHADOW-DOM



```
constructor() {
    super();
    this._root = this.attachShadow({ 'mode': 'open' });
}
 connectedCallback() {
    this._root.appendChild(templateCalendar.content.cloneNode(true));
}
```

4. ~~HTML Imports~~ vs ES Modules

- HTML Imports

```
<link rel="import"
      href="https://acme.org/acme-calendar.html">
```

- ES Modules

```
<script type=“module"
     src="https://acme.org/acme-calendar.min.js">
</script>
```

Estado actual. W3C

W3C
Word Wide Web Consortium

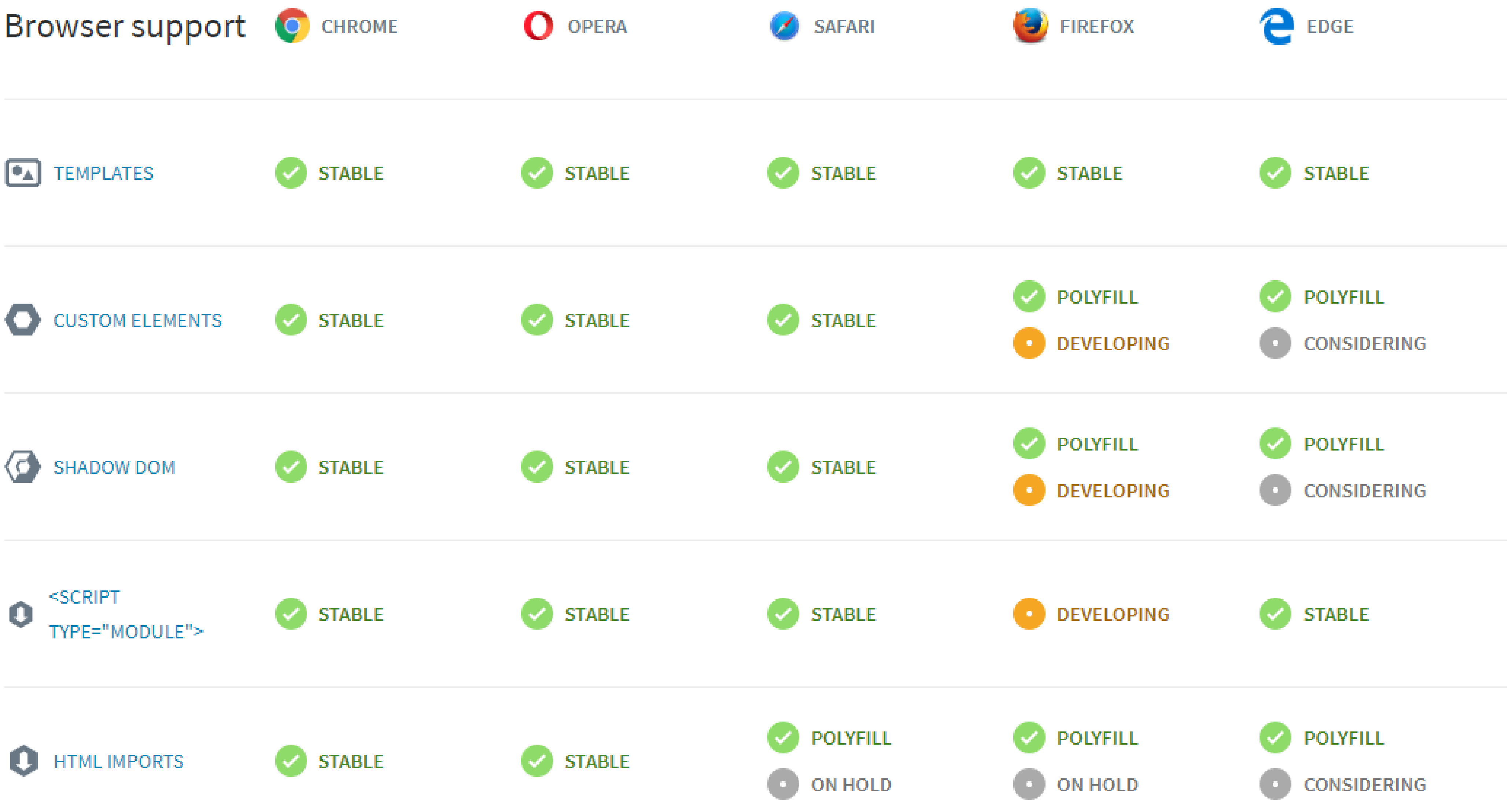
1. Custom Elements ~~v.0~~ v.1
2. Shadow DOM ~~v.0~~ v.1
3. HTML Templates
4. HTML Imports
5. ES Modules

- ESM **vs** CommonJS en NodeJS

https://medium.com/the-node-js-collection/the-current-state-of-implementation-and-planning-for-esmodules-a4ecb2aac07a

HTTP/1 → Budling **vs** HTTP/2 Bundles no necesarios

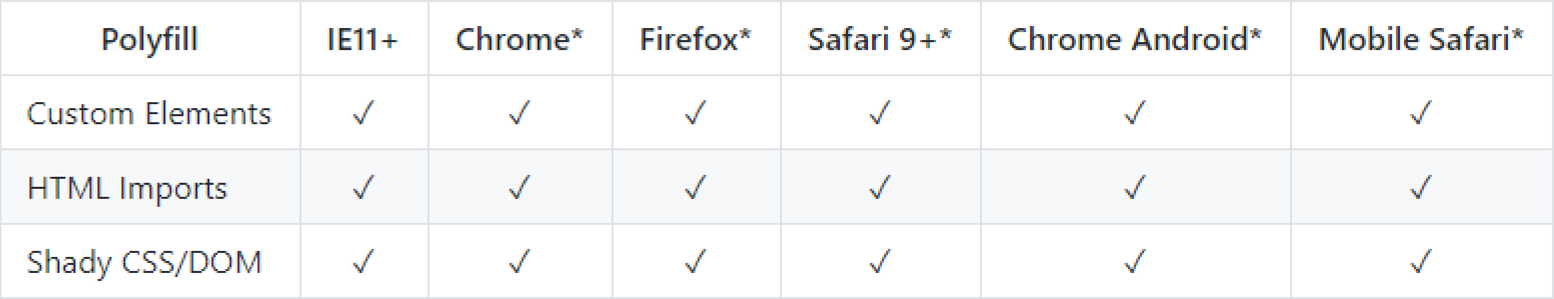
Estado actual. Soporte en Navegadores



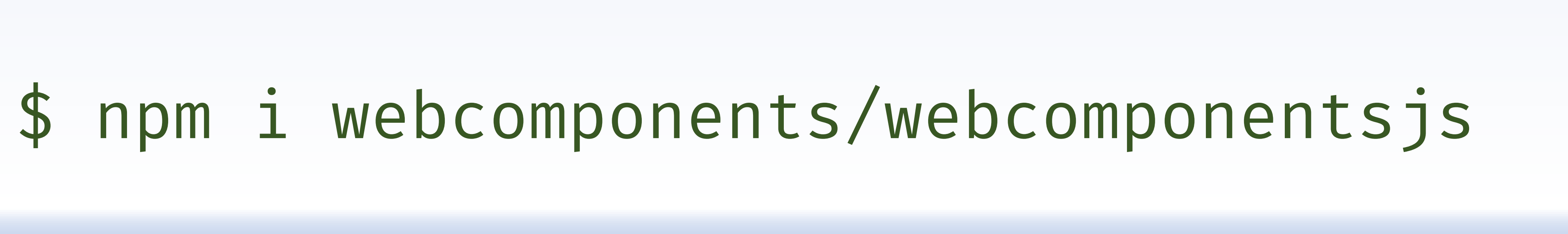
Browser support	CHROME	OPERA	SAFARI	FIREFOX	EDGE
TEMPLATES	✓ STABLE	✓ STABLE	✓ STABLE	✓ STABLE	✓ STABLE
CUSTOM ELEMENTS	✓ STABLE	✓ STABLE	✓ STABLE	✓ POLYFILL ● DEVELOPING	✓ POLYFILL ● CONSIDERING
SHADOW DOM	✓ STABLE	✓ STABLE	✓ STABLE	✓ POLYFILL ● DEVELOPING	✓ POLYFILL ● CONSIDERING
<SCRIPT TYPE="MODULE">	✓ STABLE	✓ STABLE	✓ STABLE	● DEVELOPING	✓ STABLE
HTML IMPORTS	✓ STABLE	✓ STABLE	✓ POLYFILL ● ON HOLD	✓ POLYFILL ● ON HOLD	✓ POLYFILL ● CONSIDERING

Estado actual. Polyfills

Lo que los navegadores no implementan todavía se puede cubrir extendiendo JavaScript con librerias.



Polyfill	IE11+	Chrome*	Firefox*	Safari 9+*	Chrome Android*	Mobile Safari*
Custom Elements	✓	✓	✓	✓	✓	✓
HTML Imports	✓	✓	✓	✓	✓	✓
Shady CSS/DOM	✓	✓	✓	✓	✓	✓



```
$ npm i webcomponents/webcomponentsjs
```

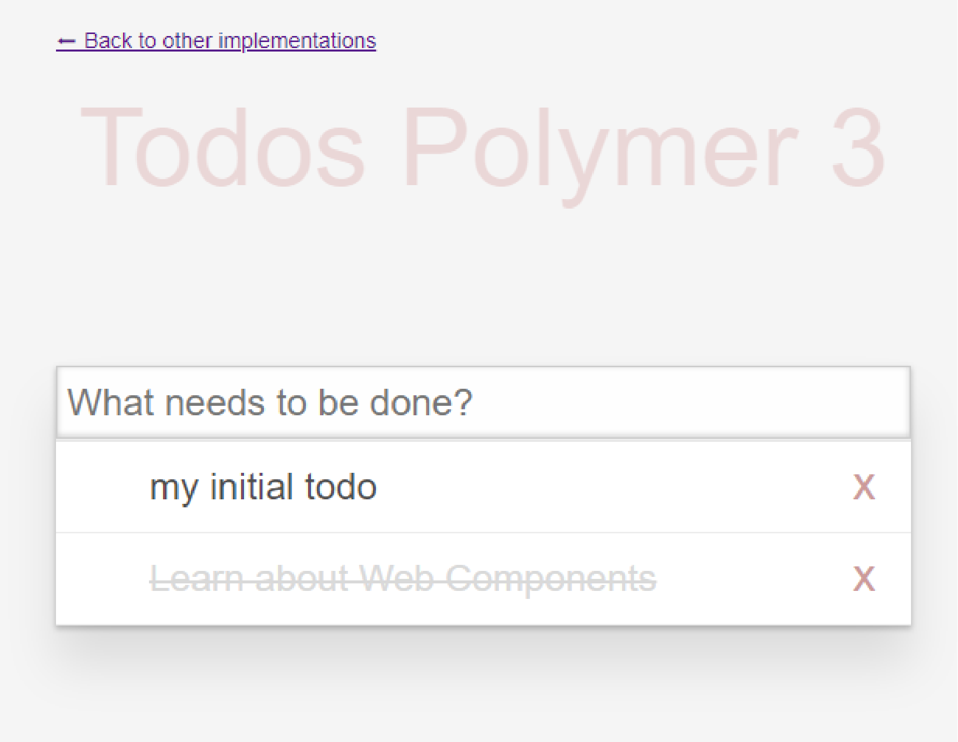
https://github.com/WebComponents/webcomponentsjs

Librerías para crear Web Components

- Native WebElements
- Polymer 3
- Skate.js
- X-Tag
- Slim.js
- Stencil
- Angular Elements
- Svelte
- VueJS

- Repositorio con el ejemplo **TODO List** en varias tecnologías

https://github.com/shprink/web-components-todo



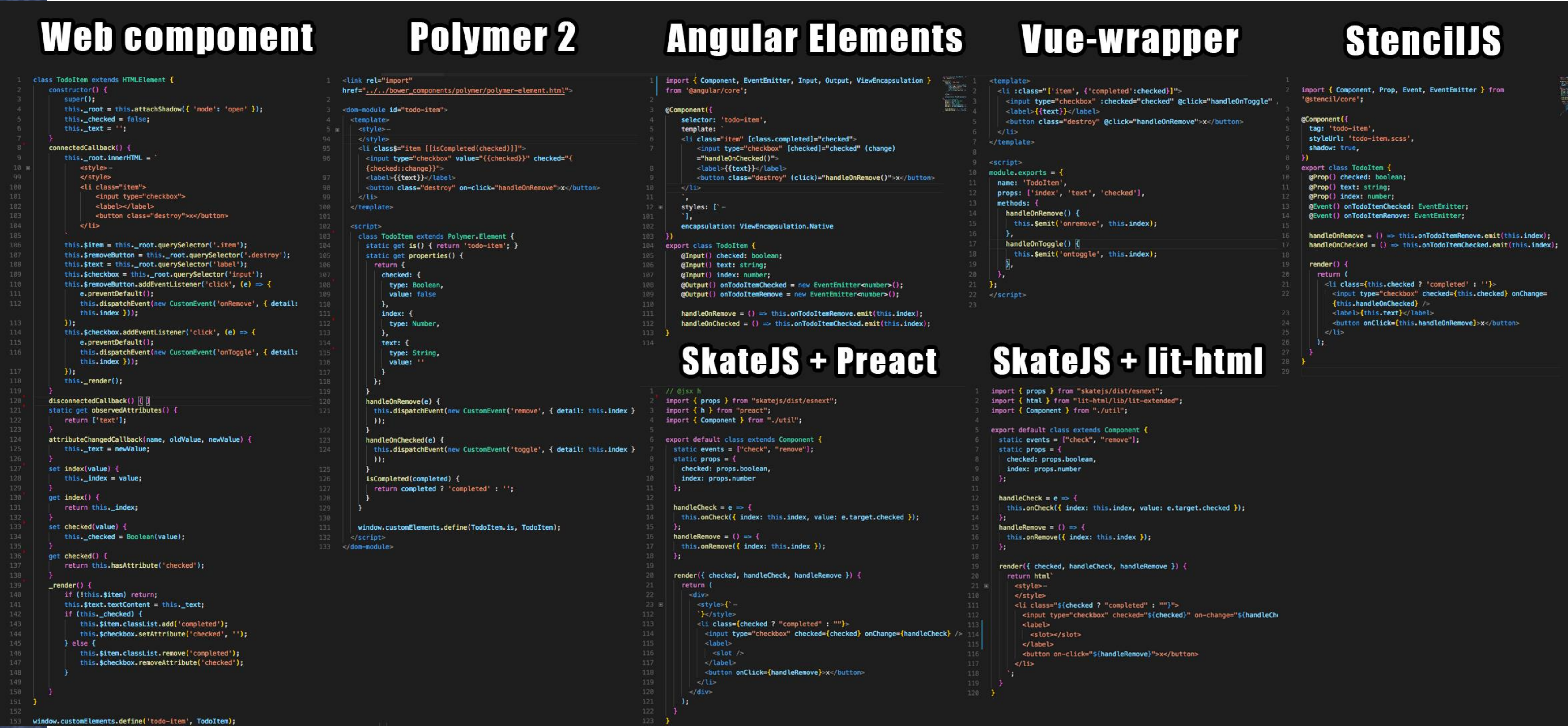
← Back to other implementations

Todos Polymer 3

What needs to be done?

my initial todo X

~~Learn about Web Components~~ X



Web component

```
class TodoItem extends HTMLElement {
    constructor() {
        super();
        this._root = this.attachShadow({ 'mode': 'open' });
        this._checked = false;
        this._text = '';
    }
    connectedCallback() {
        this._root.innerHTML = `
            <style>…
            </style>
            <li class="item">
                <input type="checkbox">
                <label></label>
                <button class="destroy">x</button>
            </li>
        `;
        this.$item = this._root.querySelector('.item');
        this.$removeButton = this._root.querySelector('.destroy');
        this.$text = this._root.querySelector('label');
        this.$checkbox = this._root.querySelector('input');
        this.$removeButton.addEventListener('click', (e) => {
            e.preventDefault();
            this.dispatchEvent(new CustomEvent('onRemove', { detail:
            this.index }));
        });
        this.$checkbox.addEventListener('click', (e) => {
            e.preventDefault();
            this.dispatchEvent(new CustomEvent('onToggle', { detail:
            this.index }));
        });
        this._render();
    }
    disconnectedCallback() { }
    static get observedAttributes() {
        return ['text'];
    }
    attributeChangedCallback(name, oldValue, newValue) {
        this._text = newValue;
    }
    set index(value) {
        this._index = value;
    }
    get index() {
        return this._index;
    }
    set checked(value) {
        this._checked = Boolean(value);
    }
    get checked() {
        return this.hasAttribute('checked');
    }
    _render() {
        if (!this.$item) return;
        this.$text.textContent = this._text;
        if (this._checked) {
            this.$item.classList.add('completed');
            this.$checkbox.setAttribute('checked', '');
        } else {
            this.$item.classList.remove('completed');
            this.$checkbox.removeAttribute('checked');
        }

    }
}

window.customElements.define('todo-item', TodoItem);
```

Polymer 2

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

<dom-module id="todo-item">
  <template>
    <style>…
    </style>
    <li class$="item [[isCompleted(checked)]]">
      <input type="checkbox" value="{{checked}}" checked="{
      {checked::change}}">
      <label>{{text}}</label>
      <button class="destroy" on-click="handleOnRemove">x</button>
    </li>
  </template>

  <script>
    class TodoItem extends Polymer.Element {
      static get is() { return 'todo-item'; }
      static get properties() {
        return {
          checked: {
            type: Boolean,
            value: false
          },
          index: {
            type: Number,
          },
          text: {
            type: String,
            value: ''
          }
        };
      }
      handleOnRemove(e) {
        this.dispatchEvent(new CustomEvent('remove', { detail: this.index }
        ));
      }
      handleOnChecked(e) {
        this.dispatchEvent(new CustomEvent('toggle', { detail: this.index }
        ));
      }
      isCompleted(completed) {
        return completed ? 'completed' : '';
      }
    }

    window.customElements.define(TodoItem.is, TodoItem);
  </script>
</dom-module>
```

Angular Elements

```
import { Component, EventEmitter, Input, Output, ViewEncapsulation }
from '@angular/core';

@Component({
    selector: 'todo-item',
    template: `
    <li class="item" [class.completed]="checked">
        <input type="checkbox" [checked]="checked" (change)
        ="handleOnChecked()">
        <label>{{text}}</label>
        <button class="destroy" (click)="handleOnRemove()">x</button>
    </li>
    `,
    styles: [`…
    `],
    encapsulation: ViewEncapsulation.Native
})
export class TodoItem {
    @Input() checked: boolean;
    @Input() text: string;
    @Input() index: number;
    @Output() onTodoItemChecked = new EventEmitter<number>();
    @Output() onTodoItemRemove = new EventEmitter<number>();

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

Vue-wrapper

```
<template>
  <li :class="['item', {'completed':checked}]">
    <input type="checkbox" :checked="checked" @click="handleOnToggle" ,
    <label>{{text}}</label>
    <button class="destroy" @click="handleOnRemove">x</button>
  </li>
</template>

<script>
module.exports = {
  name: 'TodoItem',
  props: ['index', 'text', 'checked'],
  methods: {
    handleOnRemove() {
      this.$emit('onremove', this.index);
    },
    handleOnToggle() {
      this.$emit('ontoggle', this.index);
    },
  },
};
</script>
```

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

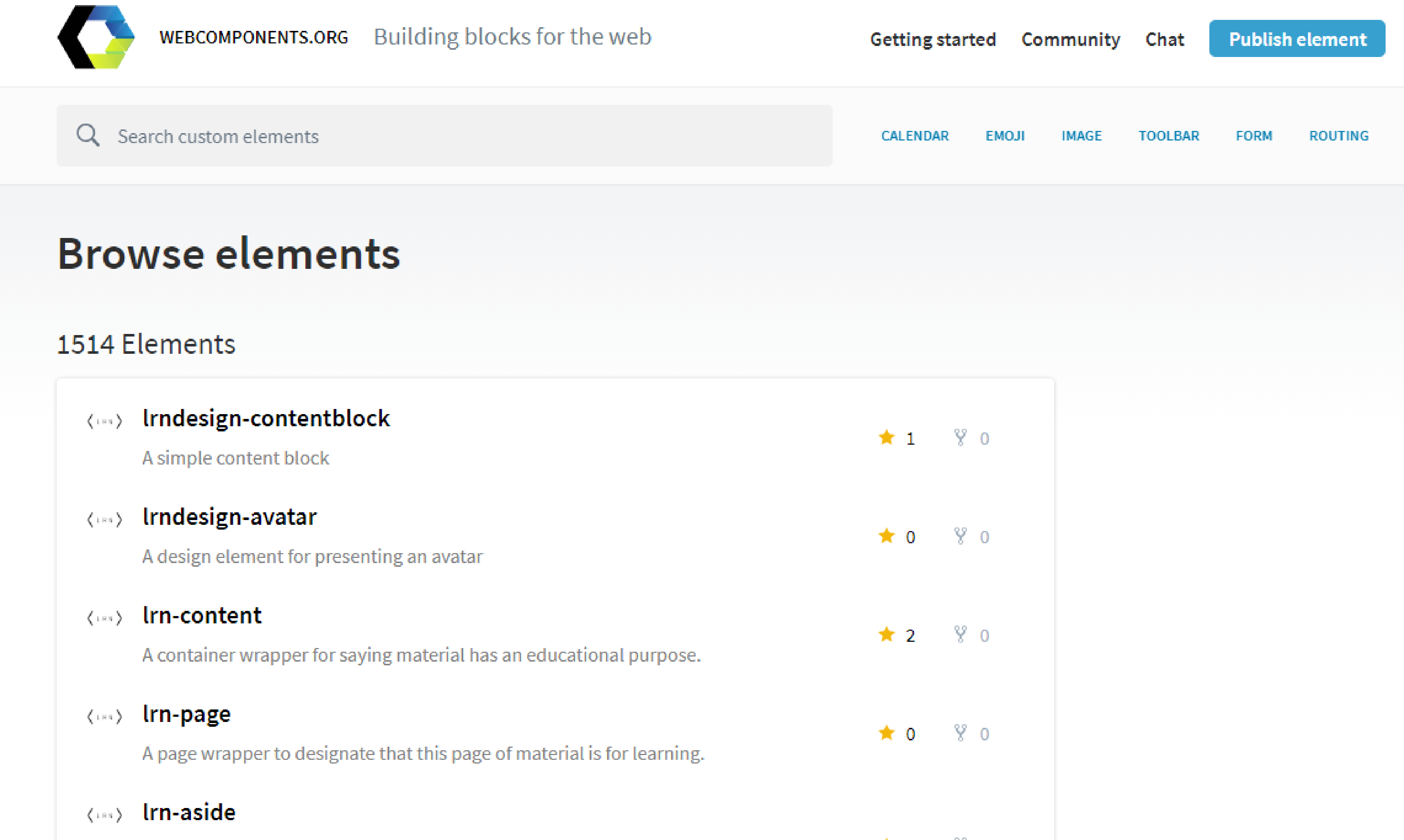
```
// @jsx h
import { props } from "skatejs/dist/esnext";
import { h } from "preact";
import { Component } from "./util";

export default class extends Component {
  static events = ["check", "remove"];
  static props = {
    checked: props.boolean,
    index: props.number
  };

  handleCheck = e => {
    this.onCheck({ index: this.index, value: e.target.checked });
  };
  handleRemove = () => {
    this.onRemove({ index: this.index });
  };

  render({ checked, handleCheck, handleRemove }) {
    return (
      <div>
        <style>{`…
        `}</style>
        <li class={checked ? "completed" : ""}>
          <input type="checkbox" checked={checked} onChange={handleCheck} />
          <label>
            <slot />
          </label>
          <button onClick={handleRemove}>x</button>
        </li>
      </div>
    );
  }
}
```

SkateJS + lit-html

```
import { props } from "skatejs/dist/esnext";
import { html } from "lit-html/lib/lit-extended";
import { Component } from "./util";

export default class extends Component {
  static events = ["check", "remove"];
  static props = {
    checked: props.boolean,
    index: props.number
  };

  handleCheck = e => {
    this.onCheck({ index: this.index, value: e.target.checked });
  };
  handleRemove = () => {
    this.onRemove({ index: this.index });
  };

  render({ checked, handleCheck, handleRemove }) {
    return html`
      <style>…
      </style>
      <li class="${checked ? "completed" : ""}">
        <input type="checkbox" checked="${checked}" on-change="${handleCh
        <label>
          <slot></slot>
        </label>
        <button on-click="${handleRemove}">x</button>
      </li>
    `;
  }
}
```

https://github.com/shprink/web-components-todo

Catálogo de componentes



WEBCOMPONENTS.ORG Building blocks for the web

Browse elements

1514 Elements

lrndesign-contentblock A simple content block	★ 1	⑂ 0
lrndesign-avatar A design element for presenting an avatar	★ 0	⑂ 0
lrn-content A container wrapper for saying material has an educational purpose.	★ 2	⑂ 0
lrn-page A page wrapper to designate that this page of material is for learning.	★ 0	⑂ 0
lrn-aside		

Quid. Un DSL mínimo para componer WebC

- DSL para prototipar Interfaz de Usuario
- Orientado a Web Components

https://quid.metadev.pro

#quid

Problemas e impedimentos

1. Consensos y cierre de **estándares** (ej. ES Modules)
2. Adopción en navegadores (desterrar polyfills)
3. Definición de **tipos** en componentes
4. Herramientas para consumir y componer Web Components

¿Preguntas?

open
south
code

@pmolinam #opensouthcode

¡Gracias!

open
south
code