stencil



The magical, reusable web component compiler

Web Component? Plaît-il?



Web Component

```
A set of web platform APIs that allow you to create new custom, reusable, encapsulated HTML tags to use in web pages and web apps...

...will work across modern browsers, and can be used with any JavaScript library or framework that works with HTML.
```

<whoop-whoop></whoop-whoop>



Concepts

- Custom Element API
- Shadow DOM
- HTML imports
- HTML Template

```
<template>
  Hello <strong></strong>
</template>
<script>
(function(window, document, undefined) {
    var thatDoc = document;
    var thisDoc = (thatDoc._currentScript || thatDoc.currentScript).ownerDocument;
    var template = thisDoc.guerySelector('template').content;
    var MyElementProto = Object.create(HTMLElement.prototype);
    MyElementProto.who = 'World';
    MyElementProto.createdCallback = function() {
        var shadowRoot = this.createShadowRoot();
        var clone = thatDoc.importNode(template, true);
        shadowRoot.appendChild(clone);
        this.strong = shadowRoot.querySelector('strong');
if (this.hasAttribute('who')) {
            var who = this.getAttribute('who');
this.setWho(who);
        else {
            this.setWho(this.who);
    MyElementProto.attributeChangedCallback = function(attr, oldVal, newVal) {
        if (attr === 'who') {
            this.setWho(newVal);
    MyElementProto.setWho = function(val) {
        this.who = val:
        this.strong.textContent = this.who;
    };
    window.MyElement = thatDoc.registerElement('hello-world', {
        prototype: MyElementProto
})(window, document);
</script>
```

```
<template>
  Hello <strong></strong>
</template>
<script>
(function(window, document, undefined) {
    var thatDoc = document;
    var thisDoc = (thatDoc._currentScript || thatDoc.currentScript).ownerDocument;
    var template = thisDoc.guerySelector('template').content;
    var MyElementProto = Object.create(HTMLElement.prototype); // Custom Element API
    MvElementProto.who = 'World';
    MyElementProto.createdCallback = function() { // Custom Element API
        var shadowRoot = this.createShadowRoot();
        var clone = thatDoc.importNode(template, true);
        shadowRoot.appendChild(clone);
        this.strong = shadowRoot.querySelector('strong');
if (this.hasAttribute('who')) {
            var who = this.getAttribute('who');
this.setWho(who);
        else {
            this.setWho(this.who);
    MyElementProto.attributeChangedCallback = function(attr, oldVal, newVal) { // Custom Element API
        if (attr === 'who') {
            this.setWho(newVal);
    MyElementProto.setWho = function(val) {
        this.who = val:
        this.strong.textContent = this.who;
   };
    window.MyElement = thatDoc.registerElement('hello-world', { // Custom Element API
        prototype: MyElementProto
})(window, document);
</script>
```

```
<template>
  Hello <strong></strong>
</template>
<script>
(function(window, document, undefined) {
    var thatDoc = document;
    var thisDoc = (thatDoc._currentScript || thatDoc.currentScript).ownerDocument;
    var template = thisDoc.guerySelector('template').content;
    var MyElementProto = Object.create(HTMLElement.prototype);
    MyElementProto.who = 'World';
    MyElementProto.createdCallback = function() {
        var shadowRoot = this.createShadowRoot(); // Shadow DOM
        var clone = thatDoc.importNode(template, true);
        shadowRoot.appendChild(clone);
        this.strong = shadowRoot.querySelector('strong');
if (this.hasAttribute('who')) {
            var who = this.getAttribute('who');
this.setWho(who);
        else {
            this.setWho(this.who);
    MyElementProto.attributeChangedCallback = function(attr, oldVal, newVal) {
        if (attr === 'who') {
            this.setWho(newVal);
    MyElementProto.setWho = function(val) {
        this.who = val:
        this.strong.textContent = this.who;
    };
    window.MyElement = thatDoc.registerElement('hello-world', {
        prototype: MyElementProto
})(window, document);
</script>
```

```
<template> <!-- HTML Template -->
 Hello <strong></strong>
</template>
<script>
(function(window, document, undefined) {
    var thatDoc = document:
    var thisDoc = (thatDoc._currentScript || thatDoc.currentScript).ownerDocument;
    var template = thisDoc.querySelector('template').content; // HTML Template
    var MyElementProto = Object.create(HTMLElement.prototype);
    MyElementProto.who = 'World';
    MyElementProto.createdCallback = function() {
        var shadowRoot = this.createShadowRoot();
        var clone = thatDoc.importNode(template, true); // HTML Template
        shadowRoot.appendChild(clone); // HTML Template
        this.strong = shadowRoot.querySelector('strong'); // HTML Template
if (this.hasAttribute('who')) {
            var who = this.getAttribute('who');
            this.setWho(who);
        else {
            this.setWho(this.who);
    MyElementProto.attributeChangedCallback = function(attr, oldVal, newVal) {
        if (attr === 'who') {
            this.setWho(newVal);
    MyElementProto.setWho = function(val) {
        this.who = val:
        this.strong.textContent = this.who; // HTML Template
   };
    window.MyElement = thatDoc.registerElement('hello-world', {
        prototype: MyElementProto
})(window, document);
</script>
```

et stencil?

The magical, reusable web component compiler

https://stenciljs.com

```
import { Component, Prop } from '@stencil/core';
@Component({
   tag: 'hello-world',
   styleUrl: 'hello-world.scss'
})
export class HelloworldComponent {
  @Prop() who: string;
  render() {
     return (
       >
         Hello {this.who}
```

stencil

Quels intêrets?

Quels intêrets?

- Syntaxe légère
- Polyfills chargés dynamiquement
- Prerendering
- Server Side Rendering
- Service Workers (Progressive Web App)
- Facilités de dévelopement (livereload, test unitaire, distribution ...)

Getting started

```
c:\dev
λ npm init stencil
npx: installed 1 in 1.702s
    Welcome to Stencil Create App!

What kind of project do you want to create?

Pick a starter » - Use arrow-keys. Return to submit.

Components (Collection of web components that can be used anywhere)
    app (Minimal starter for building an stencil app or website)
    ionic-pwa (Everything you need to build fast, production ready PWAs)
Other (specify)
```

stencil 15 / 18



Grrr



Waouf

Questions?

Web component

Polymer 2

Angular Elements Vue-wrapper

Stencills

```
class TodoItem extends HTMLElement {
                                                                             href="../../hower components/polymer/polymer-element.html">
        this._root = this.attachShadow({ 'mode': 'open' });
                                                                              <dom-module id="todo-item">
        this._checked = false;
        this._text = '':
    connectedCallback() {
                                                                                 this._root.innerHTML =
                                                                                   <input type="checkbox" value="{{checked}}" checked="{</pre>
                                                                                   <label>{{text}}</label>
           class="item">
                                                                                   <button class="destroy" on-click="handleOnRemove">x</button>
              <input type="checkbox">
               <label></label>
                                                                                 class TodoItem extends Polymer.Element {
        this.$item = this._root.querySelector('.item');
                                                                                   static get is() { return 'todo-item'; }
        this.$removeButton = this._root.querySelector('.destroy');
                                                                                   static get properties() {
        this.$text = this._root.querySelector('label');
        this.$checkbox = this._root.querySelector('input');
                                                                                       checked: {
        this.$removeButton.addEventListener('click', (e) => {
                                                                                        type: Boolean.
                                                                                         value: false
           this.dispatchEvent(new CustomEvent('onRemove', { detail:
                                                                                       index: {
                                                                                         type: Number.
        this.$checkbox.addEventListener('click', (e) => {
           e.preventDefault():
                                                                                       text: {
           this.dispatchEvent(new CustomEvent('onToggle', { detail:
                                                                                         type: String.
           this index 301:
                                                                                         value:
        this._render();
    disconnectedCallback() { }
    static get observedAttributes() {
                                                                                     this.dispatchEvent(new CustomEvent('remove', { detail: this.index }
    attributeChangedCallback(name, oldValue, newValue) {
       this._text = newValue;
                                                                                    this.dispatchEvent(new CustomEvent('toggle', { detail: this.index }
    set index(value) {
       this _index = value;
                                                                                   isCompleted(completed) {
                                                                                     return completed ? 'completed' : '':
    get index() {
        return this._index;
    set checked(value) {
                                                                                 window.customElements.define(TodoItem.is, TodoItem);
        this. checked = Boolean(value):
        return this.hasAttribute('checked');
    _render() {
       if (!this.$item) return;
        this.$text.textContent = this._text;
        if (this._checked) {
           this.$item.classList.add('completed');
           this. Scheckbox. setAttribute('checked', ''):
           this.$item.classList.remove('completed');
           this.$checkbox.removeAttribute('checked');
window.customElements.define('todo-item', TodoItem)
```

```
import { Component, EventEmitter, Input, Output, ViewEncapsulation }
from '@angular/core':
   selector: 'todo-item'.
   templates
    <input type="checkbox" [checked]="checked" (change)</pre>
       ="handleOnChecked()">
       <label>{{text}}</label>
        <button class="destroy" (click)="handleOnRemove()">x</button:</pre>
   styles: [`
   encapsulation: ViewEncapsulation Native
   @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):
```

SkateJS + Preact

```
import { props } from "skatejs/dist/esnext";
import { h } from "preact";
import { Component } from "./util":
 static events = ["check", "remove"];
  static props = {
   checked: props.boolean,
   index: props.number
   this.onCheck({ index: this.index, value: e.target.checked });
  handleRemove = () => {
   this.onRemove({ index: this.index });
  render({ checked, handleCheck, handleRemove }) {
       class={checked ? "completed" : ""}>
         <input type="checkbox" checked={checked} onChange={handleCheck} /</pre>
         <button onClick={handleRemove}>x</button</pre>
```

```
:class="['item', {'completed':checked}]">
   <input type="checkbox" :checked="checked" @click="handleOnToggle"</pre>
    <label>{{text}}</label>
    <button class="destroy" @click="handleOnRemove">x</button:</pre>
module.exports = {
 props: ['index', 'text', 'checked'],
  mothode: 1
   handleOnRemove() {
     this.$emit('onremove', this.index);
   handleOnToggle() [
      this.$emit('ontoggle', this.index);
```

SkateJS + lit-html

```
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
     <input type="checkbox" checked="${checked}" on-change="${handleChecked}"</pre>
       <button on-click="${handleRemove}">x</button>
```

```
import { Component, Prop. Event, EventEmitter } from
 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()
     <input type="checkbox" checked={this.checked} onChange=</pre>
       {this.handleOnChecked} />
       <label>{this.text}</label>
       <button onClick={this.handleOnRemove}>x</button>
```