



Tarea 7

Web Components

Julio Andrade
Gerardo Crot
M^a Josefa Espinoza



Tabla de contenidos

01

Demo

02

Nativo

03

Lit Elements

04

**Conclusiones y
aprendizajes**



01

Demo



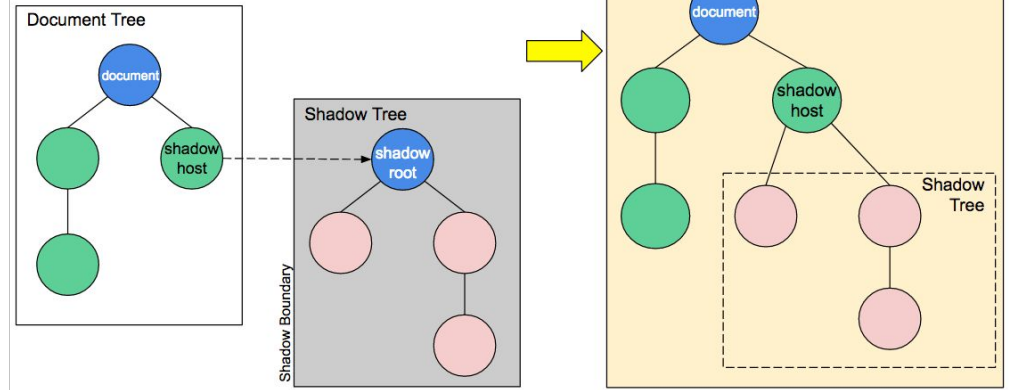
02

Nativo

Shadow DOM

Proporciona una forma de enlazar un DOM oculto y separarlo de un elemento

Permite hacer que los componentes sean independientes



Shadow DOM

Debemos especificarlo en el constructor con modo de Attach Shadow

Para acceder a un elemento del Shadow DOM debíamos llamar a su Shadow Root

```
src/todo/todo.js

class TodoList extends HTMLElement {
  constructor() {
    super();
    this.attachShadow({ mode: "open" });
    this.listItems = [
      {
        text: "Recuerda que puedes agregar items escribiendo y con enter",
        checked: false,
      },
      { text: "Aprendiendo web components", checked: true },
    ];
  }

  // on html mount
  connectedCallback() {
    this.shadowRoot.appendChild(templateTodoList.content.cloneNode(true));
    this.initialize();
    this._render();
  }
}
```

Callbacks

Dentro de la clase se pueden definir callbacks de estilo de vida

Nosotros usamos *connectedCallback* que se invoca cada vez que el elemento se añade al documento

```
src/shop/container.js

class ShopContainer extends HTMLElement {
  constructor() {
    super();
    this.attachShadow({ 'mode': 'open' });
    this.listItems = shopItemsData;
  };

  connectedCallback() {
    this.shadowRoot.appendChild(
      templateShopContainer.content.cloneNode(true)
    );
    this.initialize();
    this._render();
  };

  initialize() {
    this.containerElement = this.shadowRoot.querySelector('.container');
  };

  _render() {
    if (!this.containerElement) return;

    this.containerElement.innerHTML = '';
    this.listItems.forEach((item, idx) => {
      let itemElement = document.createElement('shop-item');
      itemElement.setAttribute('description', item.description);
      itemElement.imgSource = item.imgSource;
      itemElement.price = item.price;
      itemElement.discount = item.discount;
      this.containerElement.appendChild(itemElement);
    });
  };
};
```

Creación del elemento HTML

Creamos distintos elementos dentro de un string de JS para agregarlo al HTML

Acá le especificamos el estilo al componente

```
sro/todo/input.js

const templateTodoInput = document.createElement('template');
templateTodoInput.innerHTML = `
<style>
.todo-form {
  width: 100%;
}
#todo-input {
  min-width: 40%;
  padding: 10px;
  border-radius: 5px;
}
</style>
<form id="todo-form">
  <input
    id="todo-input"
    type="text"
    placeholder="Agregar a la lista"
  >
</form>
`;

class TodoInput extends HTMLElement {
  constructor() {
    super();
    this.attachShadow({ 'mode': 'open' });
  };

  connectedCallback() {
    this.shadowRoot.appendChild(templateTodoInput.content.cloneNode(true));
    this.formElement = this.shadowRoot.querySelector('#todo-form');
    this.inputElement = this.shadowRoot.querySelector('#todo-input');
    this.formElement.addEventListener(
      'submit',
      this.submitHandler.bind(this)
    );
  };
};
```


Custom Element

Vinculamos la clase a un componente como Custom Element

Notar además el use de un CustomEvent para pasar los eventos de un componente a otro

```
src/todo/input.js

class TodoInput extends HTMLElement {
  constructor() {
    super();
    this.attachShadow({ 'mode': 'open' });
  };

  connectedCallback() {
    this.shadowRoot.appendChild(templateTodoInput.content.cloneNode(true));
    this.formElement = this.shadowRoot.querySelector('#todo-form');
    this.inputElement = this.shadowRoot.querySelector('#todo-input');
    this.formElement.addEventListener(
      'submit',
      this.submitHandler.bind(this)
    );
  };

  submitHandler(e){
    e.preventDefault();
    if (this.inputElement.value){
      this.dispatchEvent(new CustomEvent(
        'onSubmit',
        { detail: this.inputElement.value }
      ));
      this.inputElement.value = '';
    }
  };
};

window.customElements.define('todo-input', TodoInput);
```



03

Lit Elements

Properties

LitElement permite declarar las propiedades y sus correspondientes atributos. Cada atributo se le debe declarar su tipo (String, Number, Boolean, Array y Object).

Se inicializa su valor por medio de Constructor().

```
app/src/todo-list/todo-view.js

static get properties() {
  return {
    item1: { type: String },
    item2: { type: String },
    item3: { type: String },
    todos: { type: Array },
    filter: { type: String },
    value: { type: String },
  };
}
```

Properties

Permite utilizar atributos propios en el tag, tal que el valor pueda ser extraído con el método `getAttribute()`

```
app/src/todo-list/todo-view.js

constructor() {
  super();
  this.item1 = this.getAttribute("item1");
  this.item2 = this.getAttribute("item2");
  this.item3 = this.getAttribute("item3");
  this.todos = [
    {
      task: this.item1,
      complete: false,
      id: 0,
    },
    { task: this.item2, complete: false, id: 1 },

    {
      task: this.item3,
      complete: false,
      id: 2,
    },
  ];
  this.filter = VisibilityFilters.SHOW_ALL;
  this.value = "";
}
```

```
app/index.html

<body>
  <header><h1>Lit app</h1></header>

  <main>
    <todo-view
      item1="task-1"
      item2="task-2"
      item3="task-3"
    ></todo-view>
    <shop-container></shop-container>
  </main>
  <script src="./vendor/webcomponents-loader.js"></script>
</body>
```

Fácil de integrar html con JS

Permite combinar de manera directa html con JavaScript. Lo que proporciona un mejor entendimiento

```
app/src/rodo-list/todo-view.js

<div class="todos-list">
  ${this.applyFilter(this.todos).map(
    (todo) => html`
      <div class="todo-item">
        <input
          type="checkbox"
          ?checked="${todo.complete}"
          @change = "${(e) =>
            this.updateTodoStatus(todo,
e.target.checked)}"
        >${todo.task}
        </input>
        <button
          class="delete"
          id = "${todo.id}"
          @click = "${this.remove}"
        >
          Borrar
        </button>
      </div>
    )}
  </div>
```

Encapsulamiento

Los estados y estilos de los componentes creados solamente eran modificables por sus propios métodos.

```
shop-item.js

render() {
  return html`
    <div class=shop-item>
      
      <p class="description">${this.description}</p>
      <p class=discount-price>${this.discountPrice}</p>
      <p class=normal-price">${this.normalPrice}</p>
      <p class="stars">${this.stars}</p>
    </div>
    <style>
      .shop-item{
        display: flex;
        flex-direction: column;
        border-radius: 15px;
        border: 0.5px solid gray;
        transition: box-shadow 0.3s;
        width: 300px;
        height: 600px;
        padding: 20px;
        background-color: white;
      }
      .shop-item:hover{
        box-shadow: 0 0 15px rgba(33,33,33,.5);
      }
      img{
        display: flex;
        height: 300px;
        width: 300px;
      }
    </style>`;
}
```

Encapsulamiento

```
todo-view.js

filterChanged(e) {
  this.filter = e.target.value;
}

applyFilter(todos) {
  switch (this.filter) {
    case VisibilityFilters.SHOW_ACTIVE:
      return todos.filter((todo) => !todo.complete);
    case VisibilityFilters.SHOW_COMPLETED:
      return todos.filter((todo) => todo.complete);
    default:
      return todos;
  }
}

remove(e) {
  e.preventDefault();
  this.todos = this.todos.filter((item) => item.id !== e.target.id);
}

onChange(e) {
  this.value = e.target.value;
}

addTodo(e) {
  e.preventDefault();
  if (this.value) {
    this.todos = [
      ...this.todos,
      {
        task: this.value,
        complete: false,
        id: this.todos.length + 1,
      },
    ];
    this.value = "";
  }
}

updateTodoStatus(updatedTodo, complete) {
  this.todos = this.todos.map((todo) =>
    updatedTodo === todo ? { ...updatedTodo, complete } : todo
  );
}
```



04

Conclusiones y aprendizajes

La forma nativa es más engorrosa

Se necesita definir más cosas y usar muchas veces las mismas cosas repetidas

El tema de los estilos se vuelve mucho más complejo ya que se trabaja como string y en estos casos no se podía aprovechar el relleno de VS Code

```
src/shopItem.js

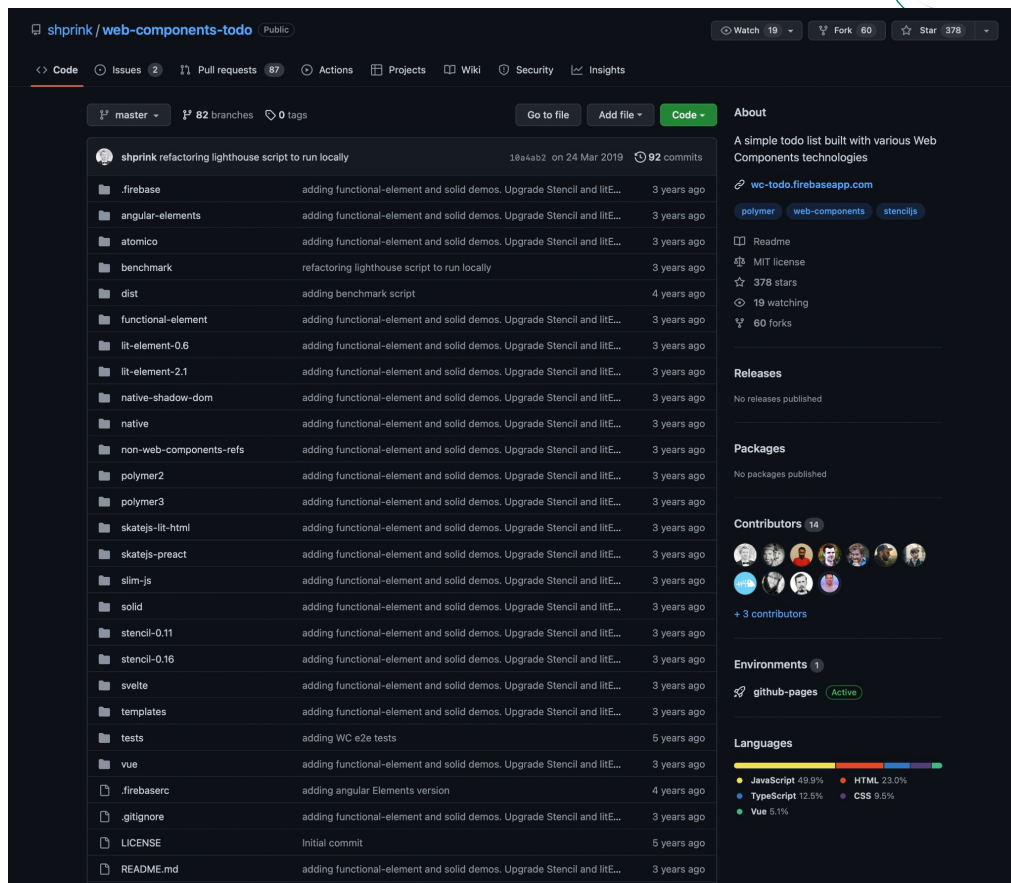
const templateShopItem = document.createElement("template");
templateShopItem.innerHTML = `
<style>
.info {
  width: 300px;
  max-height: 500px;
  box-shadow: 0 4px 8px 0 rgba(0,0,0,0.2);
  transition: box-shadow .3s;
  margin: 0;
  border-radius: 15px;
  font-family: Montserrat,sans-serif,arial,Helvetica;
  background-color: white;
  color: black;
}
.info:hover {
  box-shadow: 0 0 15px rgba(33,33,33,.5);
}
.data {
  margin: 15px;
}
img {
  max-width: 100%;
  height: 250px;
  object-fit: cover;
  display: block;
  margin-left: auto;
  margin-right: auto;
}
.amount-discount {
  display: flex;
}
.amount-discount > #price {
  margin-top: 0;
  margin-bottom: 0;
  margin-left: auto;
}
.amount-discount > .discount {
  margin-top: 0;
  margin-bottom: 0;
  padding: 5px;
  color: #2f8b2f;
  background-color: #f2f7ff;
}
.amount {
  margin-top: 10px;
}
</style>

<div class="info">
  <div class="data">
    <img/>
    <p class="description"></p>
    <div class="amount-discount">
      <p class="discount"></p>
      <p id="price"></p>
    </div>
    <div class="amount">
      <label>Normal:</label>
      <label class="price"></label>
    </div>
  </div>
</div>
</div>
`;
```

Mucha información en internet

Nos encontramos harta ayuda en internet y documentación sobre el tema

Vimos un repositorio con múltiples pruebas y demos de muchas implementaciones de Web Components



The screenshot shows the GitHub repository page for 'shrink / web-components-todo'. The repository is public and has 19 watchers, 60 forks, and 378 stars. It is currently on the 'master' branch with 82 branches and 0 tags. The repository description is 'A simple todo list built with various Web Components technologies'. The repository is licensed under MIT and has 378 stars and 19 watchers. The repository is built with various Web Components technologies, including polymer, web-components, and stenciljs. The repository is a simple todo list built with various Web Components technologies. The repository is a simple todo list built with various Web Components technologies. The repository is a simple todo list built with various Web Components technologies.

File/Folder	Description	Commit Date	Commits
.firebase	adding functional-element and solid demos. Upgrade Stencil and litE...	3 years ago	
angular-elements	adding functional-element and solid demos. Upgrade Stencil and litE...	3 years ago	
atomico	adding functional-element and solid demos. Upgrade Stencil and litE...	3 years ago	
benchmark	refactoring lighthouse script to run locally	3 years ago	
dist	adding benchmark script	4 years ago	
functional-element	adding functional-element and solid demos. Upgrade Stencil and litE...	3 years ago	
lit-element-0.6	adding functional-element and solid demos. Upgrade Stencil and litE...	3 years ago	
lit-element-2.1	adding functional-element and solid demos. Upgrade Stencil and litE...	3 years ago	
native-shadow-dom	adding functional-element and solid demos. Upgrade Stencil and litE...	3 years ago	
native	adding functional-element and solid demos. Upgrade Stencil and litE...	3 years ago	
non-web-components-refs	adding functional-element and solid demos. Upgrade Stencil and litE...	3 years ago	
polymer2	adding functional-element and solid demos. Upgrade Stencil and litE...	3 years ago	
polymer3	adding functional-element and solid demos. Upgrade Stencil and litE...	3 years ago	
skatejs-lit-html	adding functional-element and solid demos. Upgrade Stencil and litE...	3 years ago	
skatejs-preact	adding functional-element and solid demos. Upgrade Stencil and litE...	3 years ago	
slim-js	adding functional-element and solid demos. Upgrade Stencil and litE...	3 years ago	
solid	adding functional-element and solid demos. Upgrade Stencil and litE...	3 years ago	
stencil-0.11	adding functional-element and solid demos. Upgrade Stencil and litE...	3 years ago	
stencil-0.16	adding functional-element and solid demos. Upgrade Stencil and litE...	3 years ago	
svelte	adding functional-element and solid demos. Upgrade Stencil and litE...	3 years ago	
templates	adding functional-element and solid demos. Upgrade Stencil and litE...	3 years ago	
tests	adding WC e2e tests	5 years ago	
vue	adding functional-element and solid demos. Upgrade Stencil and litE...	3 years ago	
.firebaseerc	adding angular Elements version	4 years ago	
.gitignore	adding functional-element and solid demos. Upgrade Stencil and litE...	3 years ago	
LICENSE	Initial commit	5 years ago	
README.md	adding functional-element and solid demos. Upgrade Stencil and litE...	3 years ago	



Dudas o comentarios



Tarea 7

Web Components

Julio Andrade
Gerardo Crot
M^a Josefa Espinoza

