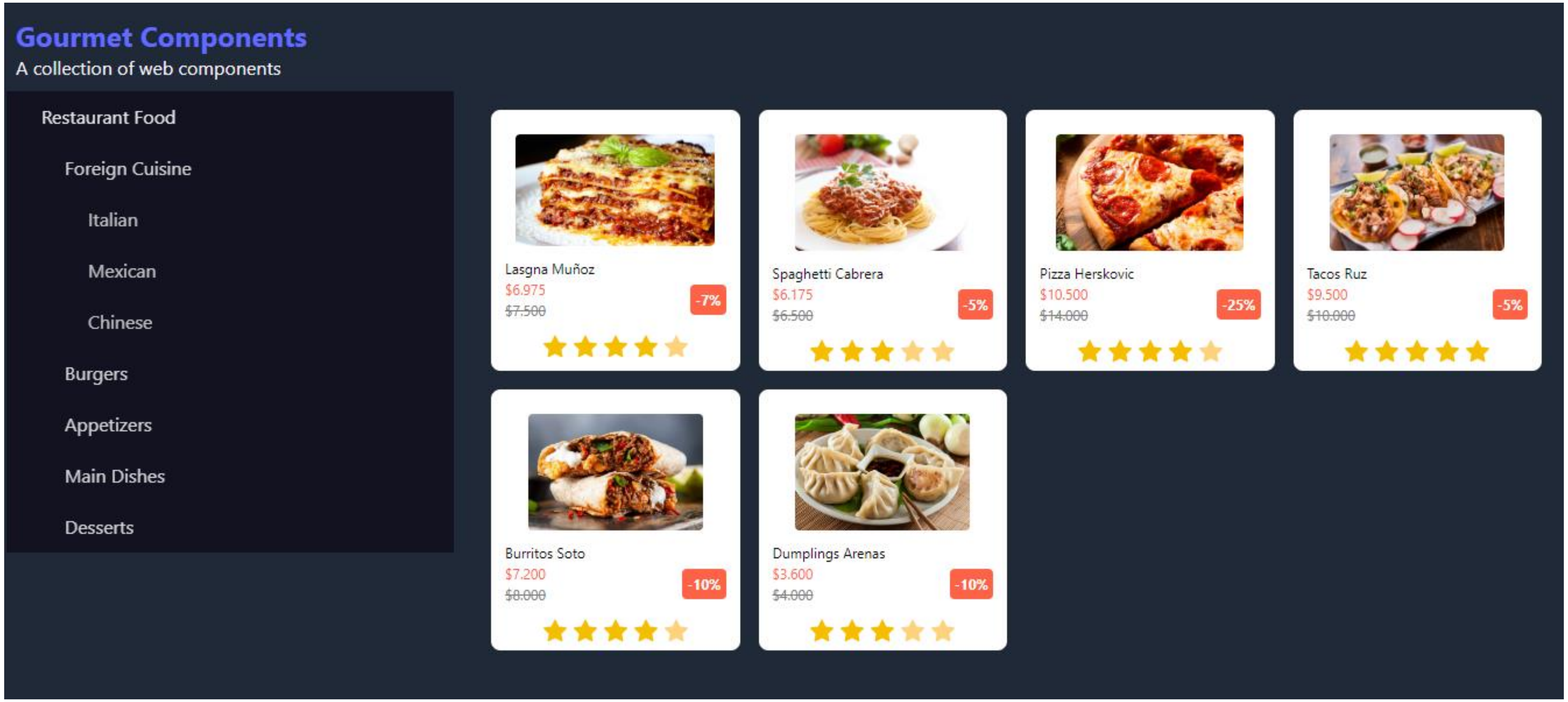
Web Components

```html
<list-component>
  <member-component> José Antonio Castro </member-component>
  <member-component> José Madriaza </member-component>
  <member-component> Benjamín Vicente</member-component>
</list-component>
```

WEBCOMPONENTS

Lit

Demo


Gourmet Components

A collection of web components

Restaurant Food

Foreign Cuisine

Italian

Mexican

Chinese

Burgers

Appetizers

Main Dishes

Desserts

Lasgna Muñoz
$6.975
~~$7.500~~
-7%

Spaghetti Cabrera
$6.175
~~$6.500~~
-5%

Pizza Herskovic
$10.500
~~$14.000~~
-25%

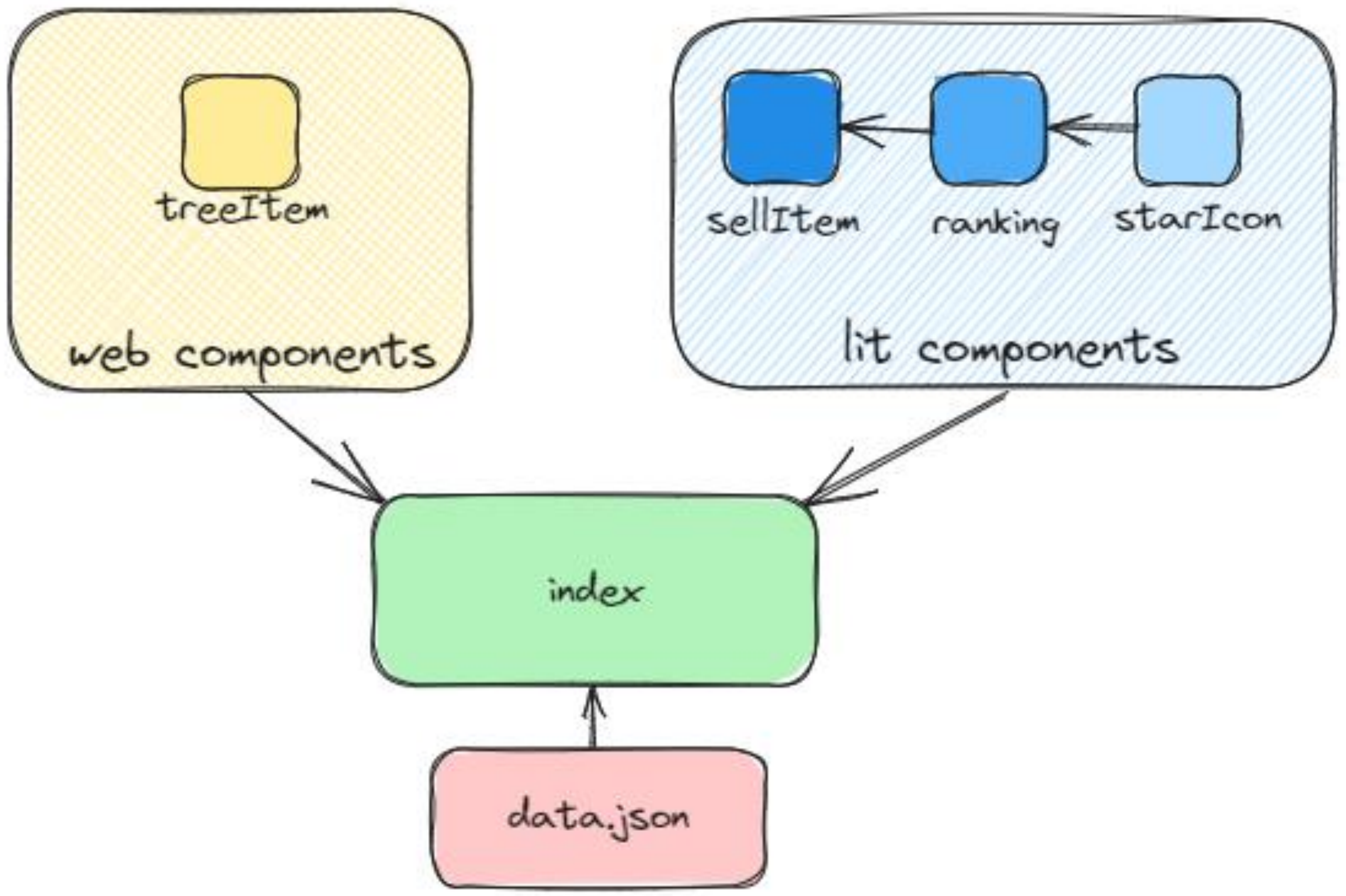
Tacos Ruz
$9.500
~~$10.000~~
-5%

Burritos Soto
$7.200
~~$8.000~~
-10%

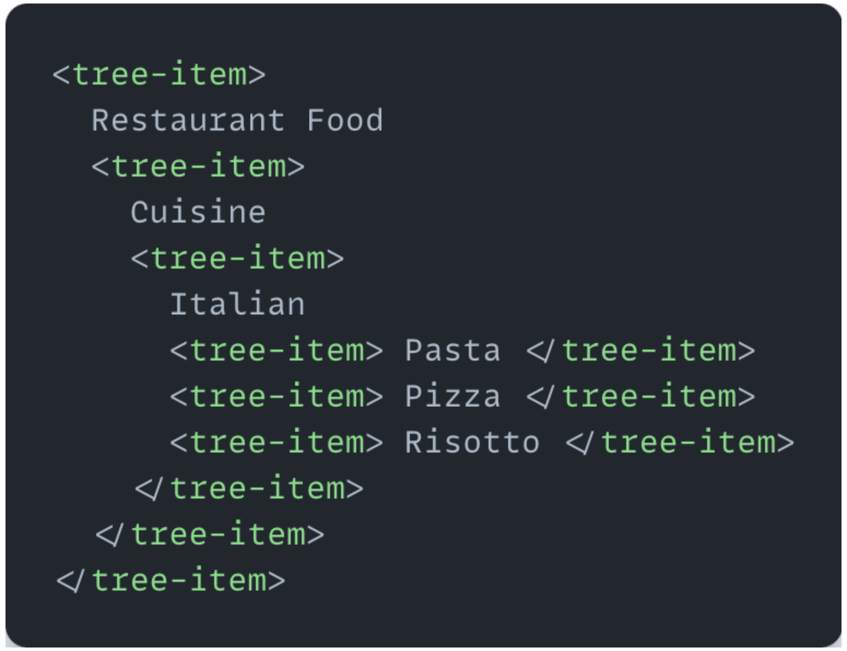
Dumplings Arenas
$3.600
~~$4.000~~
-10%

https://iic3585-2023.github.io/web-components-grupo-01/

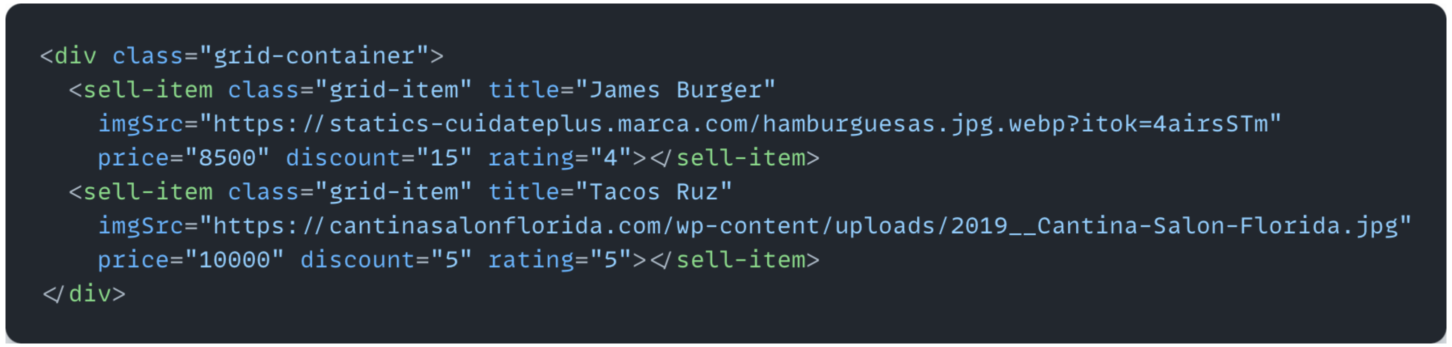
Arquitectura



Renderizado de componentes



```
<tree-item>
  Restaurant Food
  <tree-item>
    Cuisine
    <tree-item>
      Italian
      <tree-item> Pasta </tree-item>
      <tree-item> Pizza </tree-item>
      <tree-item> Risotto </tree-item>
    </tree-item>
  </tree-item>
</tree-item>
```



```
<div class="grid-container">
  <sell-item class="grid-item" title="James Burger"
    imgSrc="https://statics-cuidateplus.marca.com/hamburguesas.jpg.webp?itok=4airsSTm"
    price="8500" discount="15" rating="4"></sell-item>
  <sell-item class="grid-item" title="Tacos Ruz"
    imgSrc="https://cantinasalonflorida.com/wp-content/uploads/2019__Cantina-Salon-Florida.jpg"
    price="10000" discount="5" rating="5"></sell-item>
</div>
```

```
function addItemToTree(tree: HTMLElement, item: ITreeData, category: string = "") {
  if (!tree) return;

  appendNode(tree, "tree-item", (el) => {

    el.textContent = item.name;
    treeItems.push(el);
    category += item.name + ";";
    if (item.children) {
      item.children.map((child) => addItemToTree(el, child, category));
    } else if (item.products) {
      item.products.map((product) => addProductToMenu(product, category));
    }
  })
}
```

```
type HTMLElementKey = keyof HTMLElementTagNameMap;

function appendNode<T extends HTMLElementKey>(parent: HTMLElement, element: T, fn: ((el: HTMLElementTagNameMap[T]) => void) | undefined) {
  const el = document.createElement(element);
  parent.appendChild(el);
  if (fn) fn(el);
  return el;
}
```

Estructura de un WebComponent

Attributos del elemento

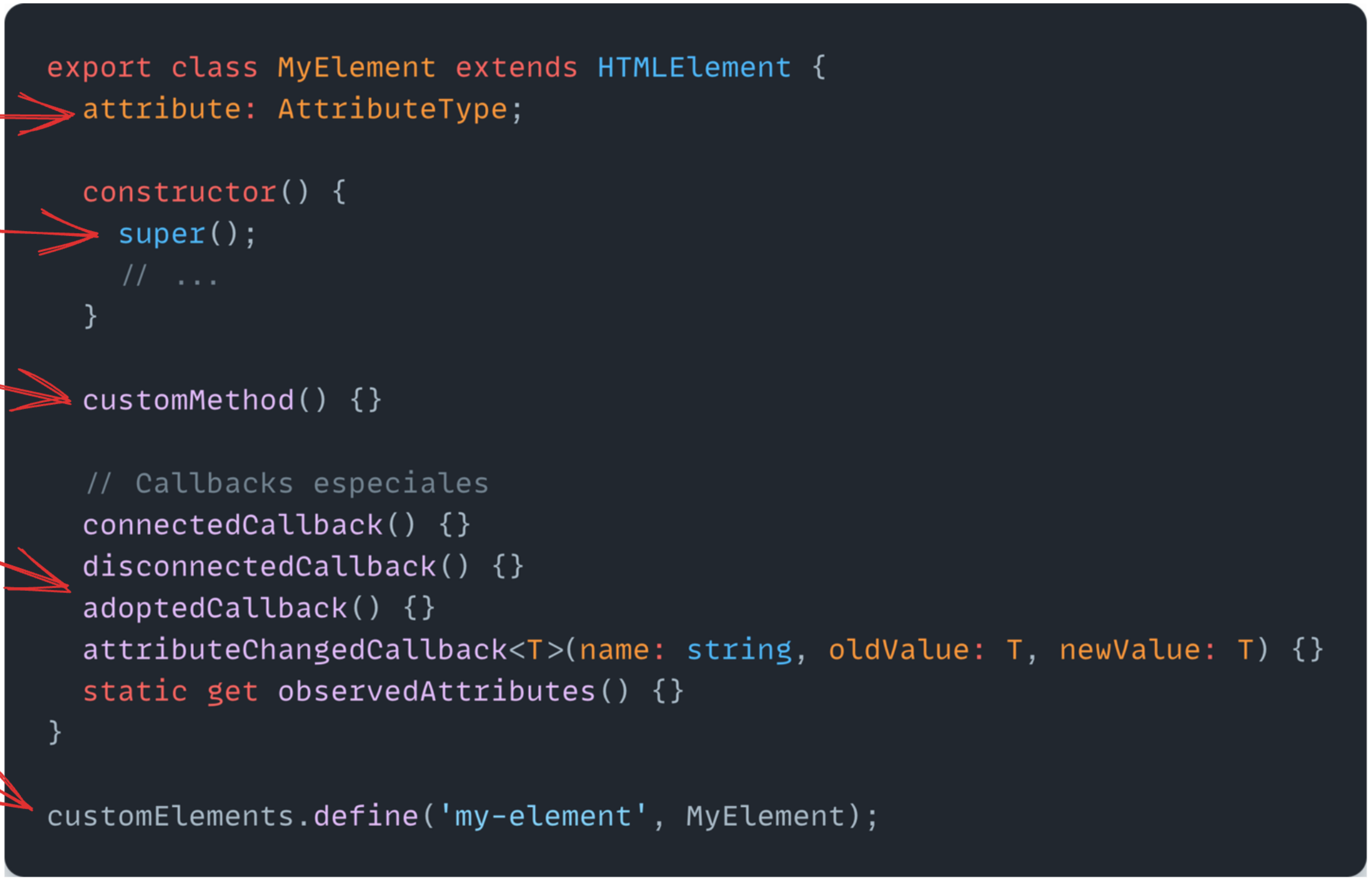
Constructor de la clase

Métodos de la clase

Callbacks especiales para cambios en el HTML

Añadir componente a scope global

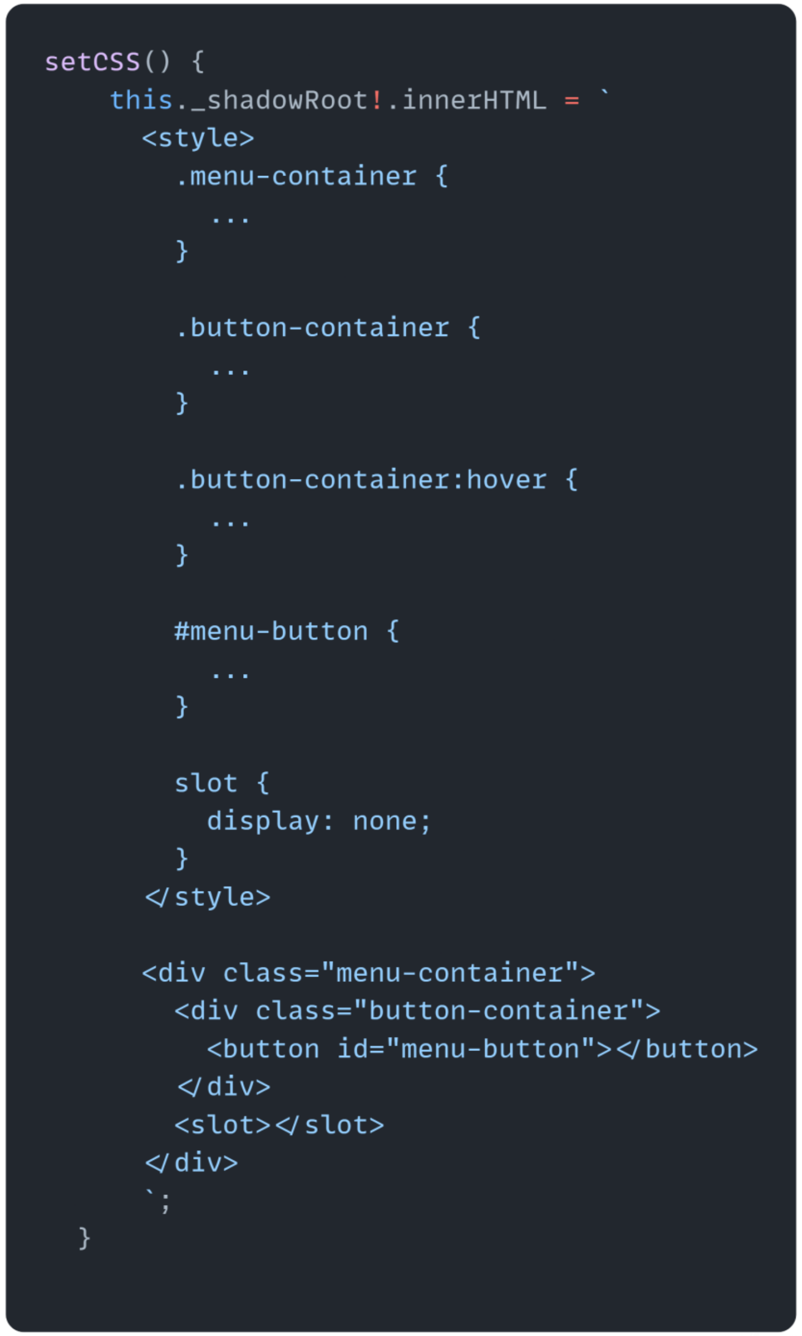
Añadir estilos requiere insertar un style tag



```
export class MyElement extends HTMLElement {
  attribute: AttributeType;

  constructor() {
    super();
    // ...
  }

  customMethod() {}

  // Callbacks especiales
  connectedCallback() {}
  disconnectedCallback() {}
  adoptedCallback() {}
  attributeChangedCallback<T>(name: string, oldValue: T, newValue: T) {}
  static get observedAttributes() {}
}

customElements.define('my-element', MyElement);
```

CSS en WebComponents

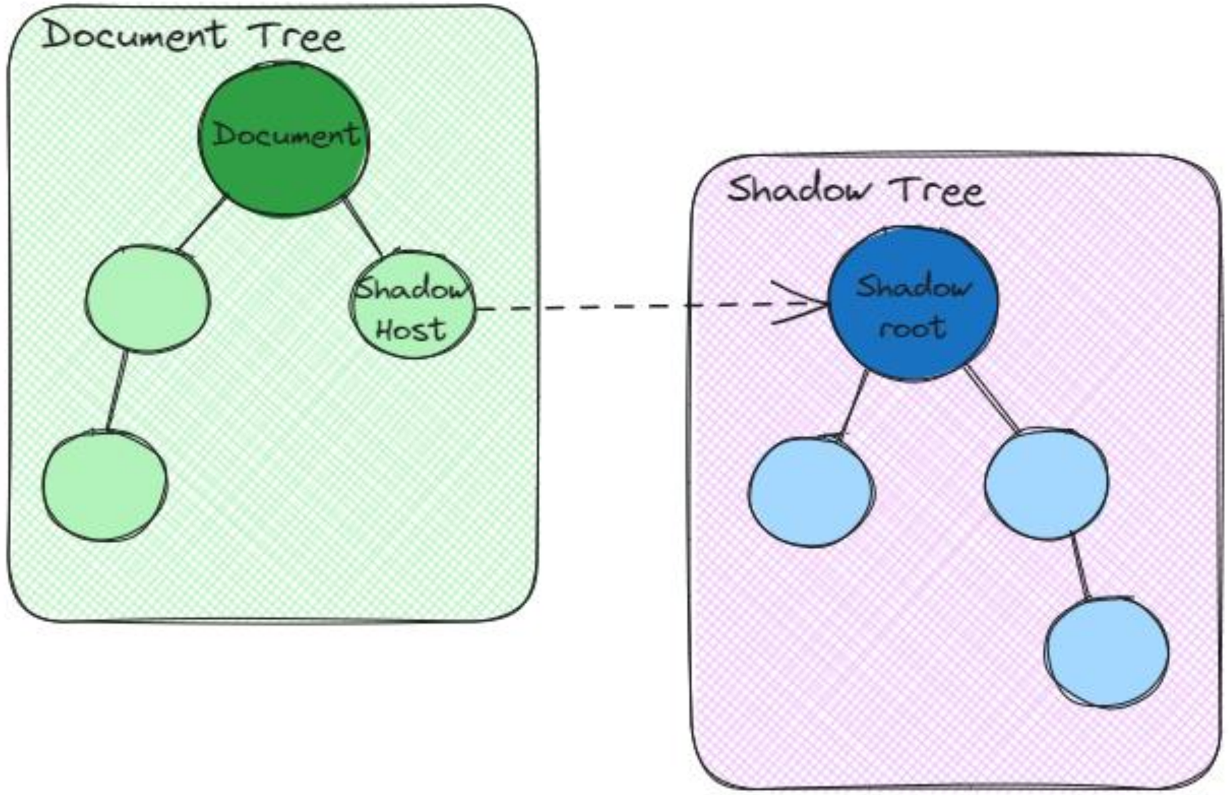
- Al propio shadow root se le añade el estilo
- Pero es solo texto por lo que no tiene las ventajas de un editor



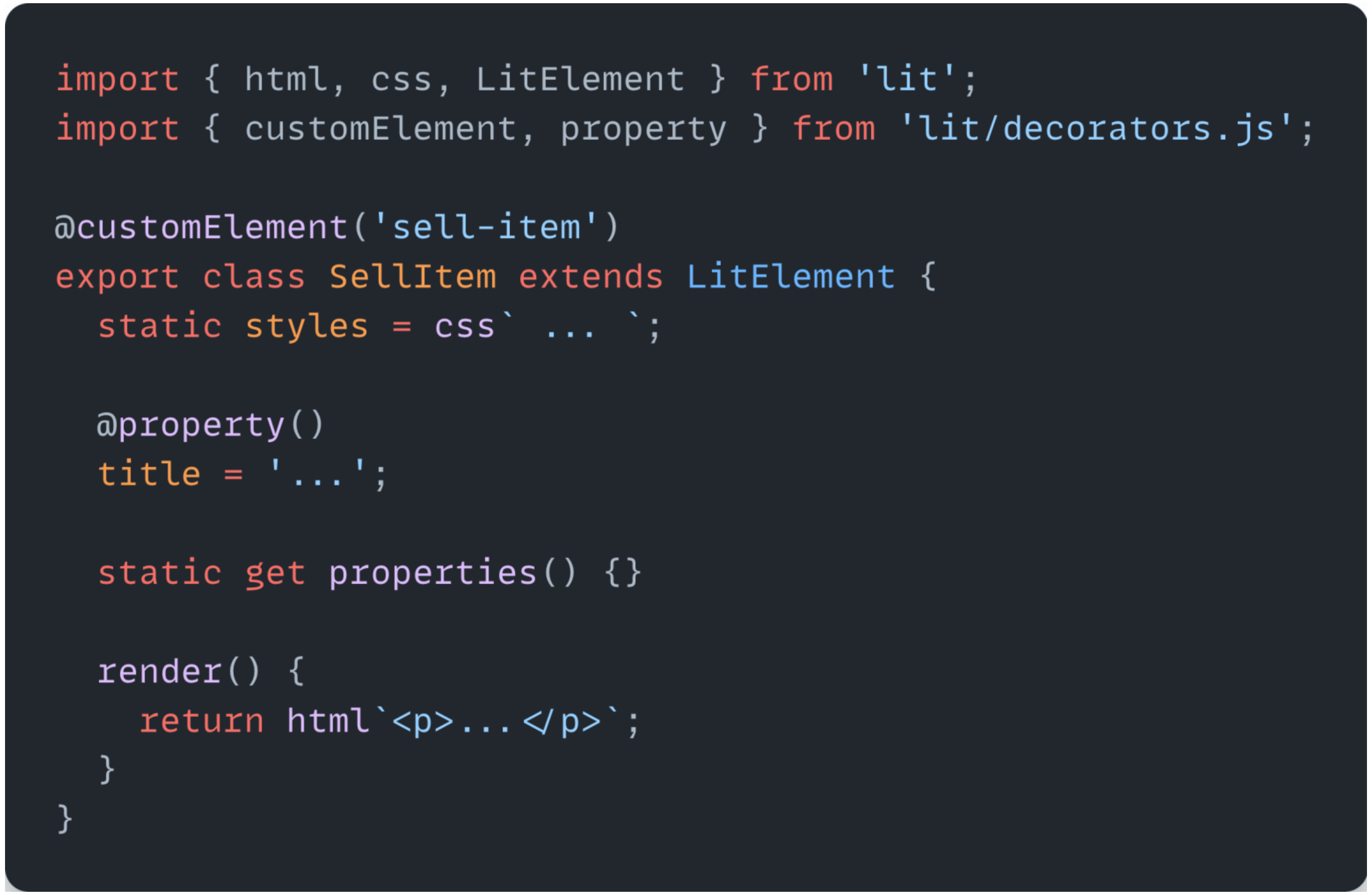
```
setCSS() {
    this._shadowRoot!.innerHTML = `
      <style>
        .menu-container {
          ...
        }

        .button-container {
          ...
        }

        .button-container:hover {
          ...
        }

        #menu-button {
          ...
        }

        slot {
          display: none;
        }
      </style>

      <div class="menu-container">
        <div class="button-container">
          <button id="menu-button"></button>
        </div>
        <slot></slot>
      </div>
      `;
  }
```

ShadowRoot

- Encapsulamiento
- Open vs Closed
- <video/>

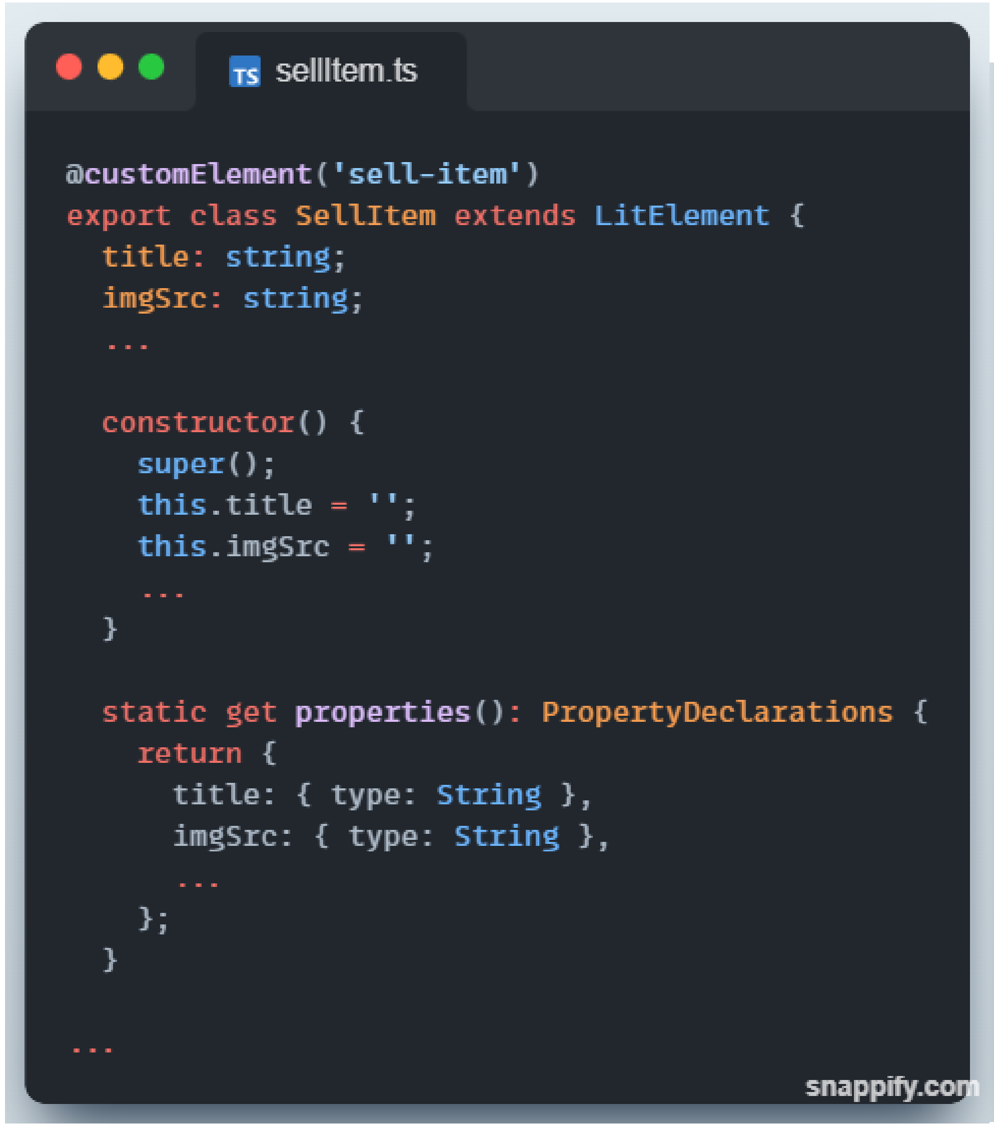


Estructura de Lit Component



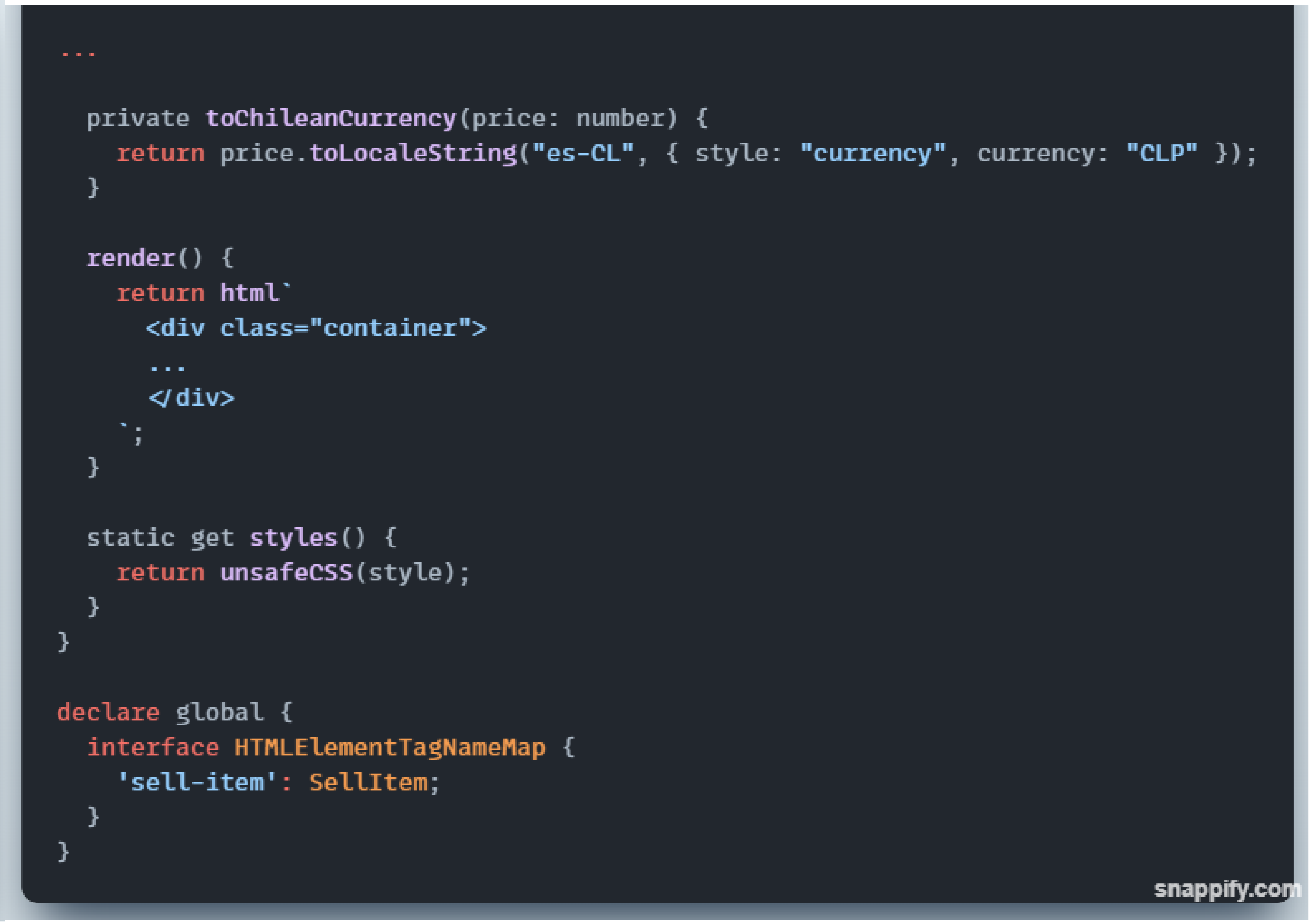
```
import { html, css, LitElement } from 'lit';
import { customElement, property } from 'lit/decorators.js';

@customElement('sell-item')
export class SellItem extends LitElement {
  static styles = css` ... `;

  @property()
  title = '...';

  static get properties() {}

  render() {
    return html`<p>...</p>`;
  }
}
```

Lit Components



sellItem.ts

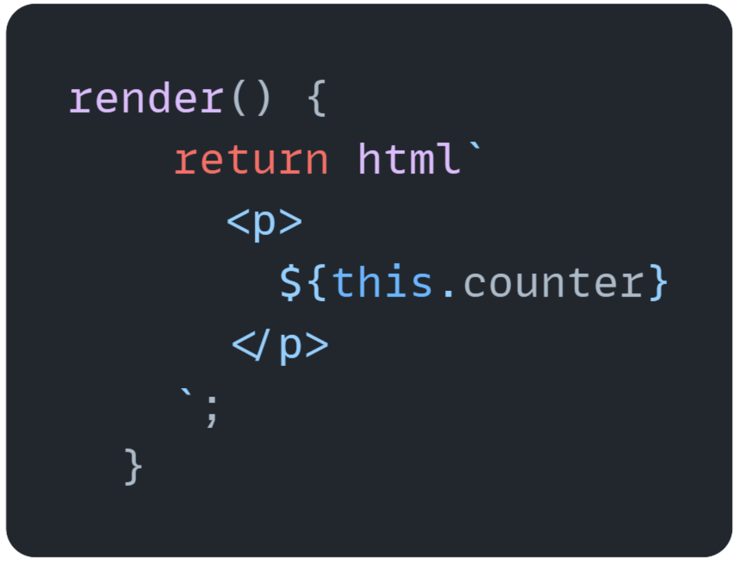
```
@customElement('sell-item')
export class SellItem extends LitElement {
  title: string;
  imgSrc: string;
  ...

  constructor() {
    super();
    this.title = '';
    this.imgSrc = '';
    ...
  }

  static get properties(): PropertyDeclarations {
    return {
      title: { type: String },
      imgSrc: { type: String },
      ...
    };
  }

...
```



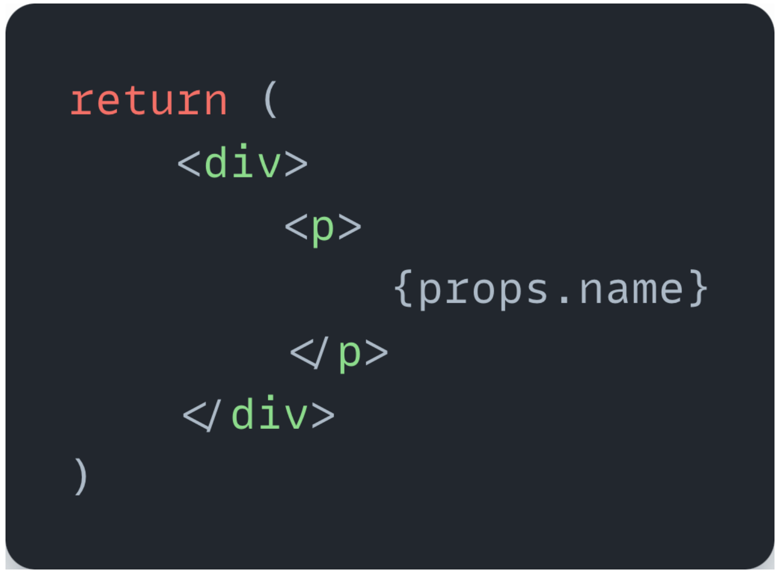
```
...

  private toChileanCurrency(price: number) {
    return price.toLocaleString("es-CL", { style: "currency", currency: "CLP" });
  }

  render() {
    return html`
      <div class="container">
      ...
      </div>
    `;
  }

  static get styles() {
    return unsafeCSS(style);
  }
}

declare global {
  interface HTMLElementTagNameMap {
    'sell-item': SellItem;
  }
}
```

Utilidades de Lit Components

- Simplicidad comparado con WebComponents
- Recuerda al funcionamiento de clases en React



```
render() {
    return html`
      <p>
        ${this.counter}
      </p>
    `;
  }
```



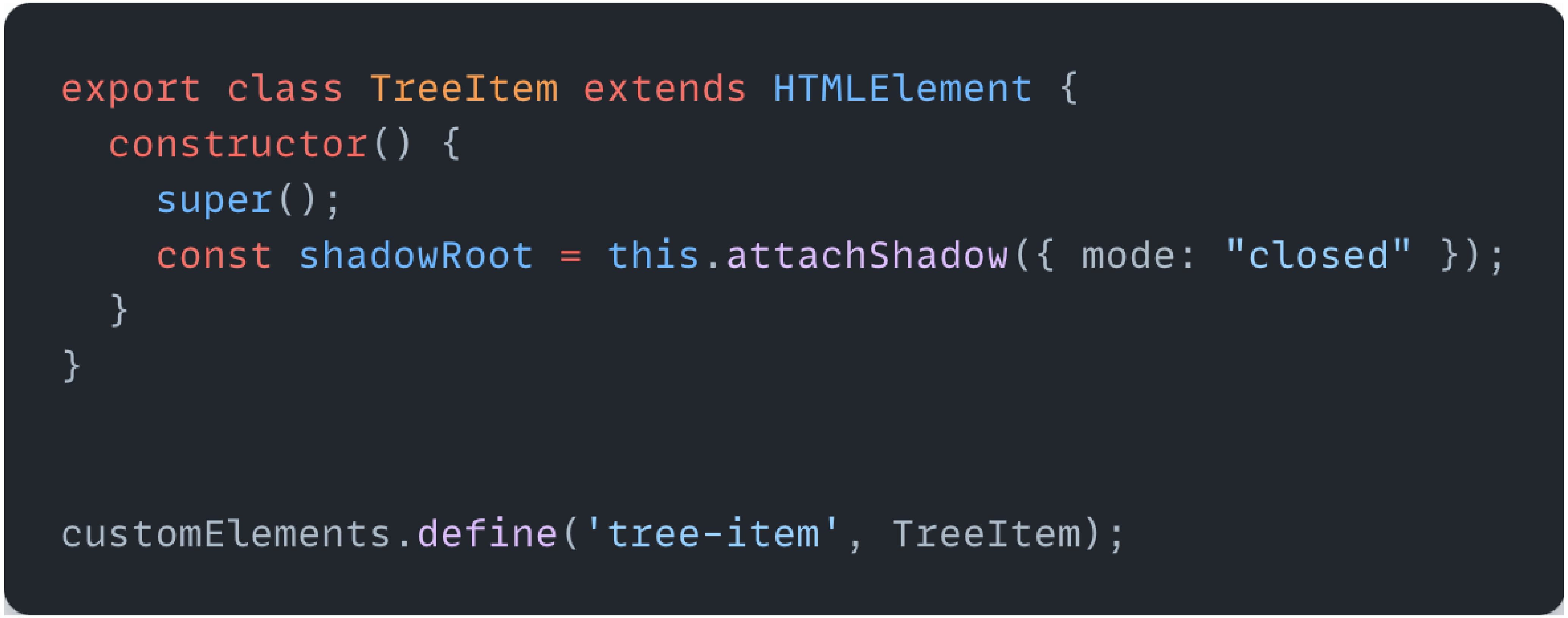
```
return (
    <div>
        <p>
            {props.name}
        </p>
    </div>
)
```

Lit HTML Templates

```
html`
  <div class="container">

    <div class="img-container">
      <img id="product-image" class="rounded" src=${this.imgSrc} alt="" />
    </div>

    <h4 class="title">
      ${this.title}
    </h4>

    <div class="price-box">
      <span>
        <h3 class="sale-price">
          ${this.toChileanCurrency(this.setDiscount(this.price, this.discount))}
        </h3>
        <h4 class="normal-price">
          ${this.toChileanCurrency(this.price)}
        </h4>
      </span>

      <span class="discount-tag">
        -${this.discount}%
      </span>
    </div>

    <rating-component rating="${this.rating}"></rating-component>
  </div>
`;
```

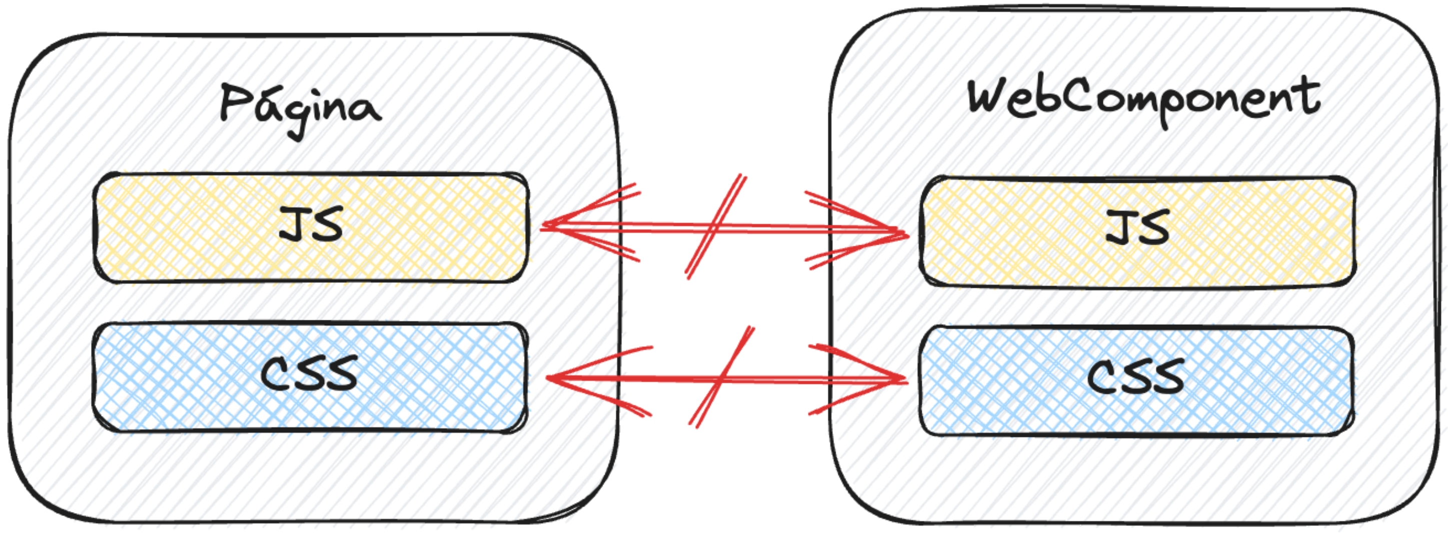
Conclusiones

Setup muy fácil

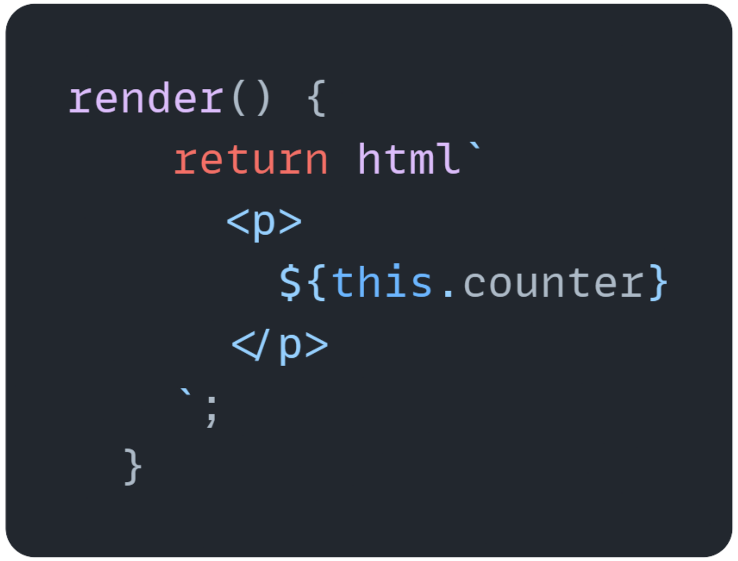


```
export class TreeItem extends HTMLElement {
  constructor() {
    super();
    const shadowRoot = this.attachShadow({ mode: "closed" });
  }
}


customElements.define('tree-item', TreeItem);
```

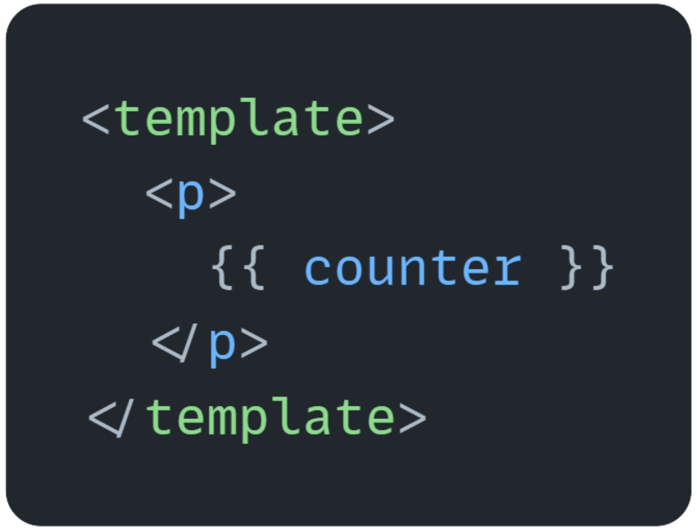
Utiles para full encapsulamiento



Sin DSL: downgrade vs otros



```
render() {
    return html`
      <p>
        ${this.counter}
      </p>
    `;
  }
```



```
<template>
  <p>
    {{ counter }}
  </p>
</template>
```

DSL: Domain Specific Language, como templates de Vue y Svelte

Mucho más que components 😵‍💫 💫

WEBCOMPONENTS Lit

Componentes que **renderizan a HTML**

Componentes que **son HTML**