T7: Web Components

Javiera Inostroza, Elías Sabja, Samuel Zúñiga

¿Qué hicimos?

WebComponents

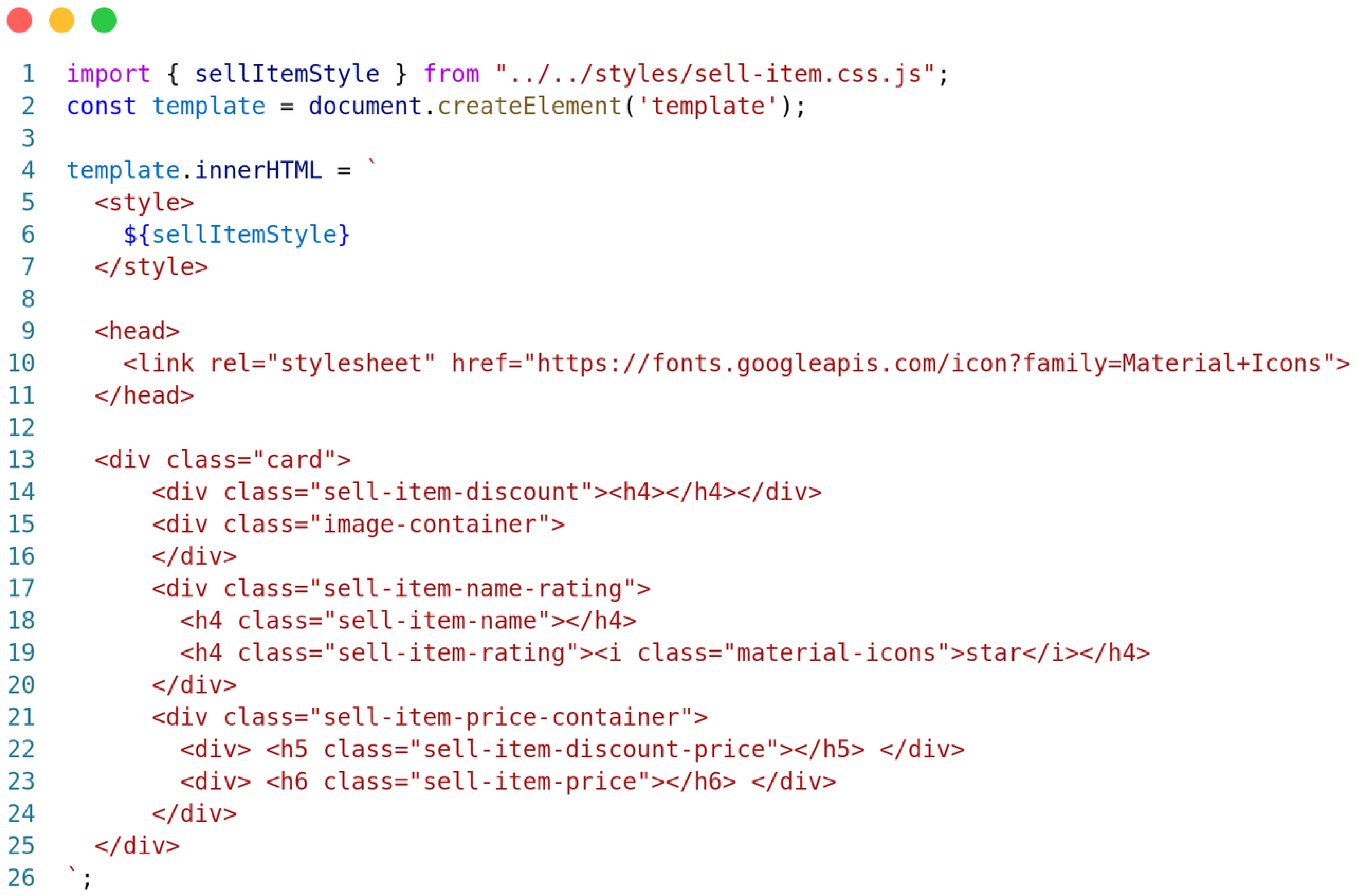
Custom Elements Templates Shadow DOM Modules CSS Scopes

Lit

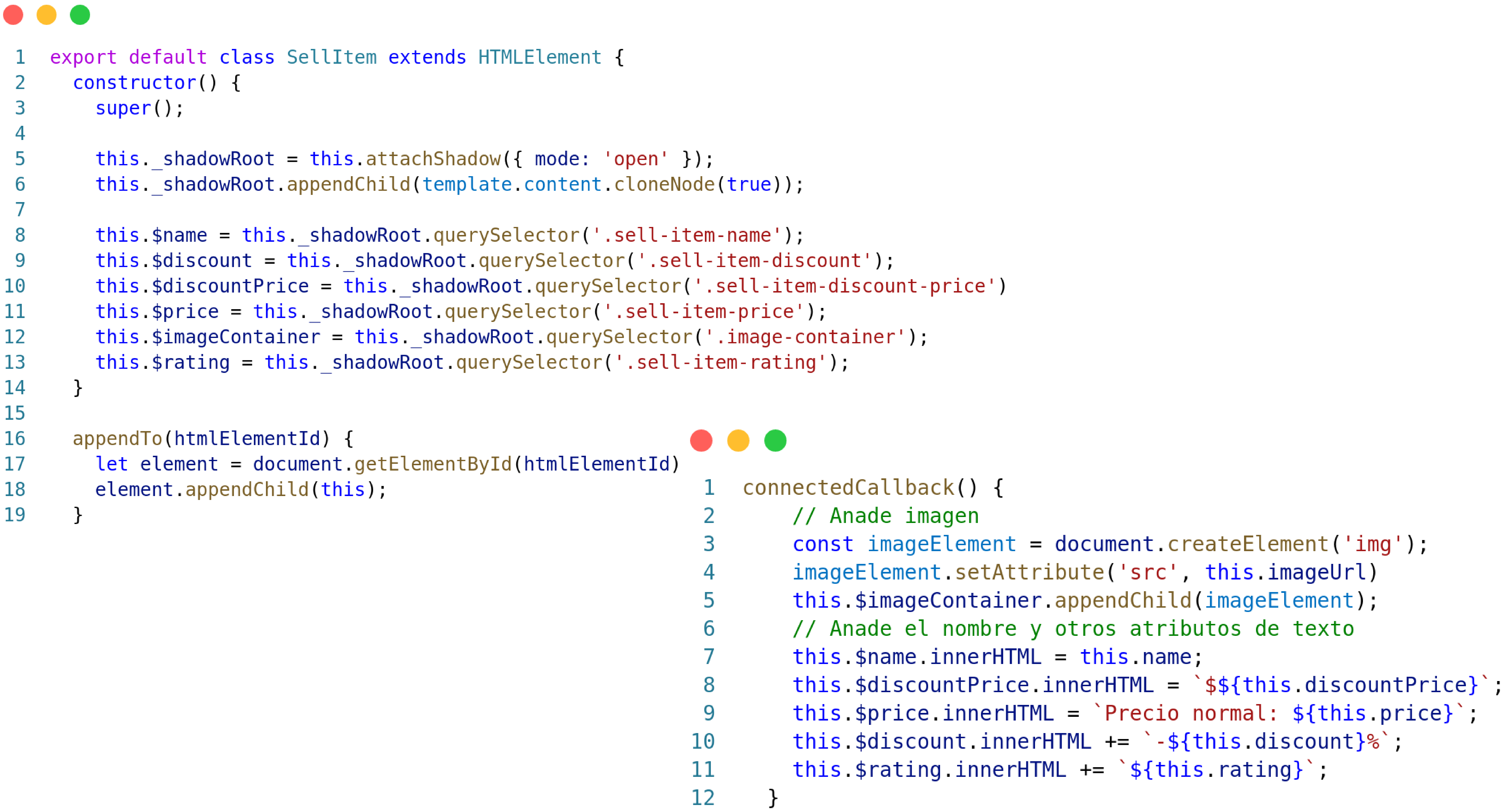
DEMO

Web Components

SELL-ITEM

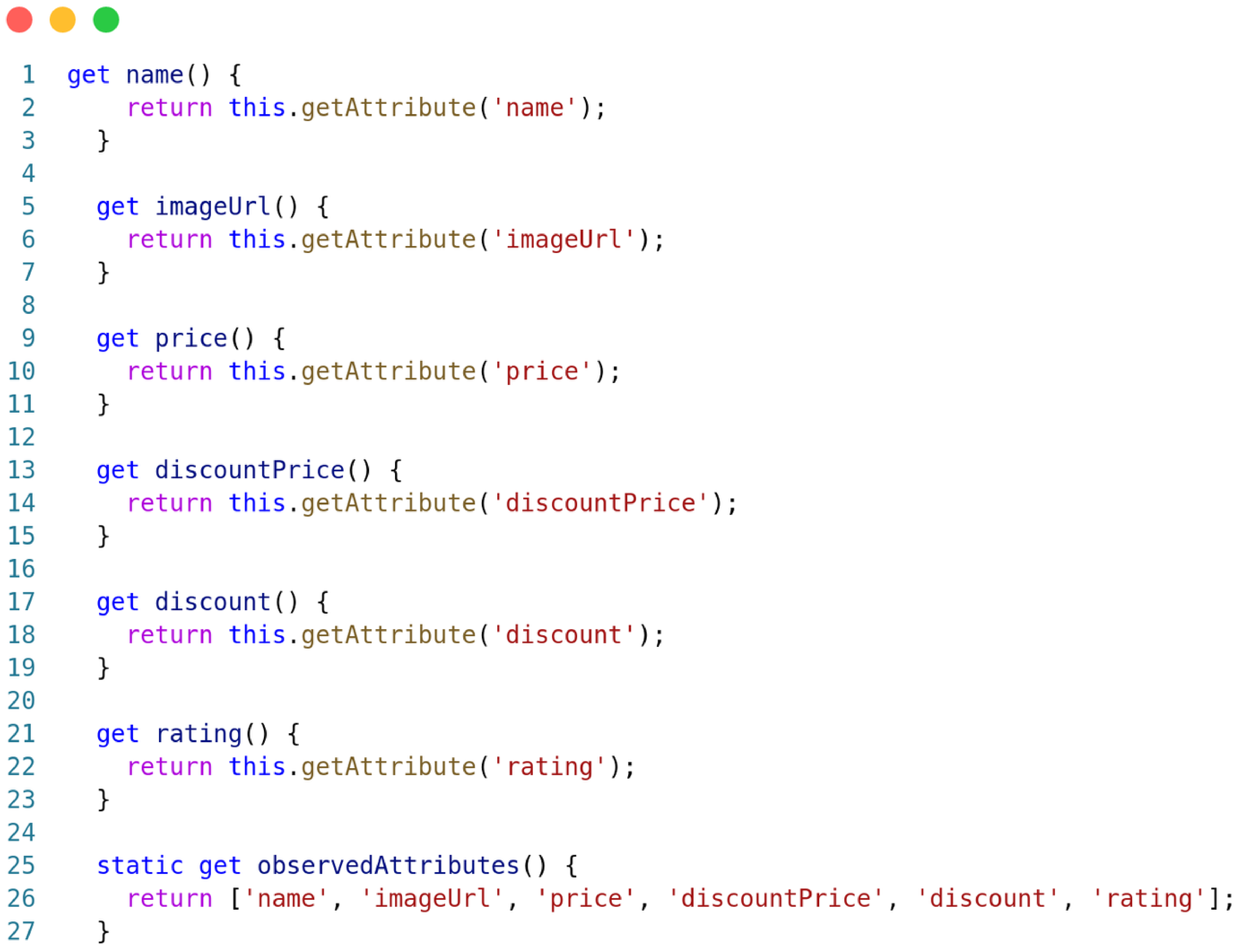


```
import { sellItemStyle } from "../../styles/sell-item.css.js";
const template = document.createElement('template');

template.innerHTML = `
  <style>
    ${sellItemStyle}
  </style>

  <head>
    <link rel="stylesheet" href="https://fonts.googleapis.com/icon?family=Material+Icons">
  </head>

  <div class="card">
      <div class="sell-item-discount"><h4></h4></div>
      <div class="image-container">
      </div>
      <div class="sell-item-name-rating">
        <h4 class="sell-item-name"></h4>
        <h4 class="sell-item-rating"><i class="material-icons">star</i></h4>
      </div>
      <div class="sell-item-price-container">
        <div> <h5 class="sell-item-discount-price"></h5> </div>
        <div> <h6 class="sell-item-price"></h6> </div>
      </div>
  </div>
`;
```



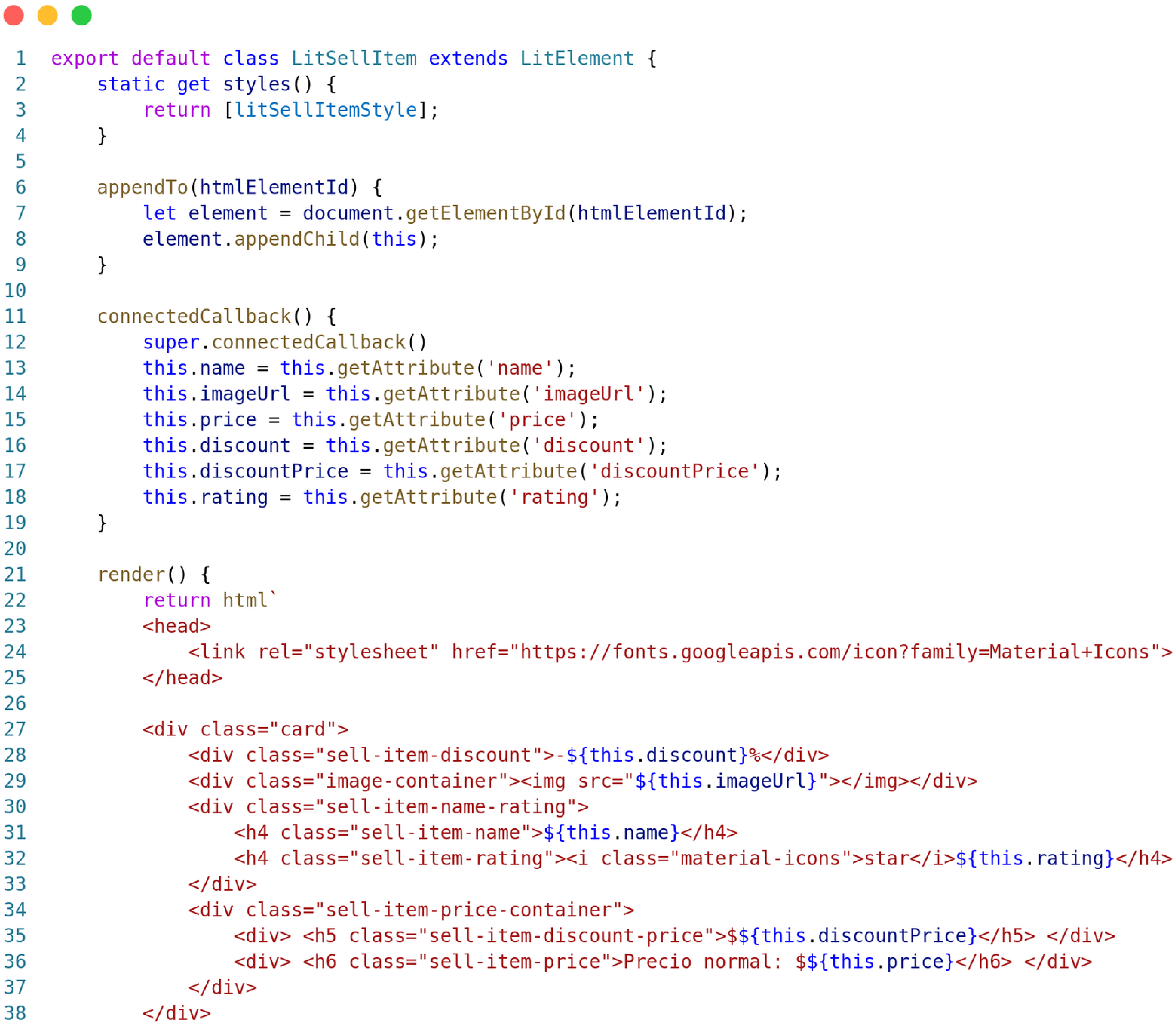
```
export default class SellItem extends HTMLElement {
  constructor() {
    super();

    this._shadowRoot = this.attachShadow({ mode: 'open' });
    this._shadowRoot.appendChild(template.content.cloneNode(true));

    this.$name = this._shadowRoot.querySelector('.sell-item-name');
    this.$discount = this._shadowRoot.querySelector('.sell-item-discount');
    this.$discountPrice = this._shadowRoot.querySelector('.sell-item-discount-price')
    this.$price = this._shadowRoot.querySelector('.sell-item-price');
    this.$imageContainer = this._shadowRoot.querySelector('.image-container');
    this.$rating = this._shadowRoot.querySelector('.sell-item-rating');
  }

  appendTo(htmlElementId) {
    let element = document.getElementById(htmlElementId)
    element.appendChild(this);
  }
```

```
connectedCallback() {
    // Anade imagen
    const imageElement = document.createElement('img');
    imageElement.setAttribute('src', this.imageUrl)
    this.$imageContainer.appendChild(imageElement);
    // Anade el nombre y otros atributos de texto
    this.$name.innerHTML = this.name;
    this.$discountPrice.innerHTML = `$${this.discountPrice}`;
    this.$price.innerHTML = `Precio normal: ${this.price}`;
    this.$discount.innerHTML += `-${this.discount}%`;
    this.$rating.innerHTML += `${this.rating}`;
  }
```



```
get name() {
    return this.getAttribute('name');
  }

  get imageUrl() {
    return this.getAttribute('imageUrl');
  }

  get price() {
    return this.getAttribute('price');
  }

  get discountPrice() {
    return this.getAttribute('discountPrice');
  }

  get discount() {
    return this.getAttribute('discount');
  }

  get rating() {
    return this.getAttribute('rating');
  }

  static get observedAttributes() {
    return ['name', 'imageUrl', 'price', 'discountPrice', 'discount', 'rating'];
  }
```

LIT-SELL-ITEM

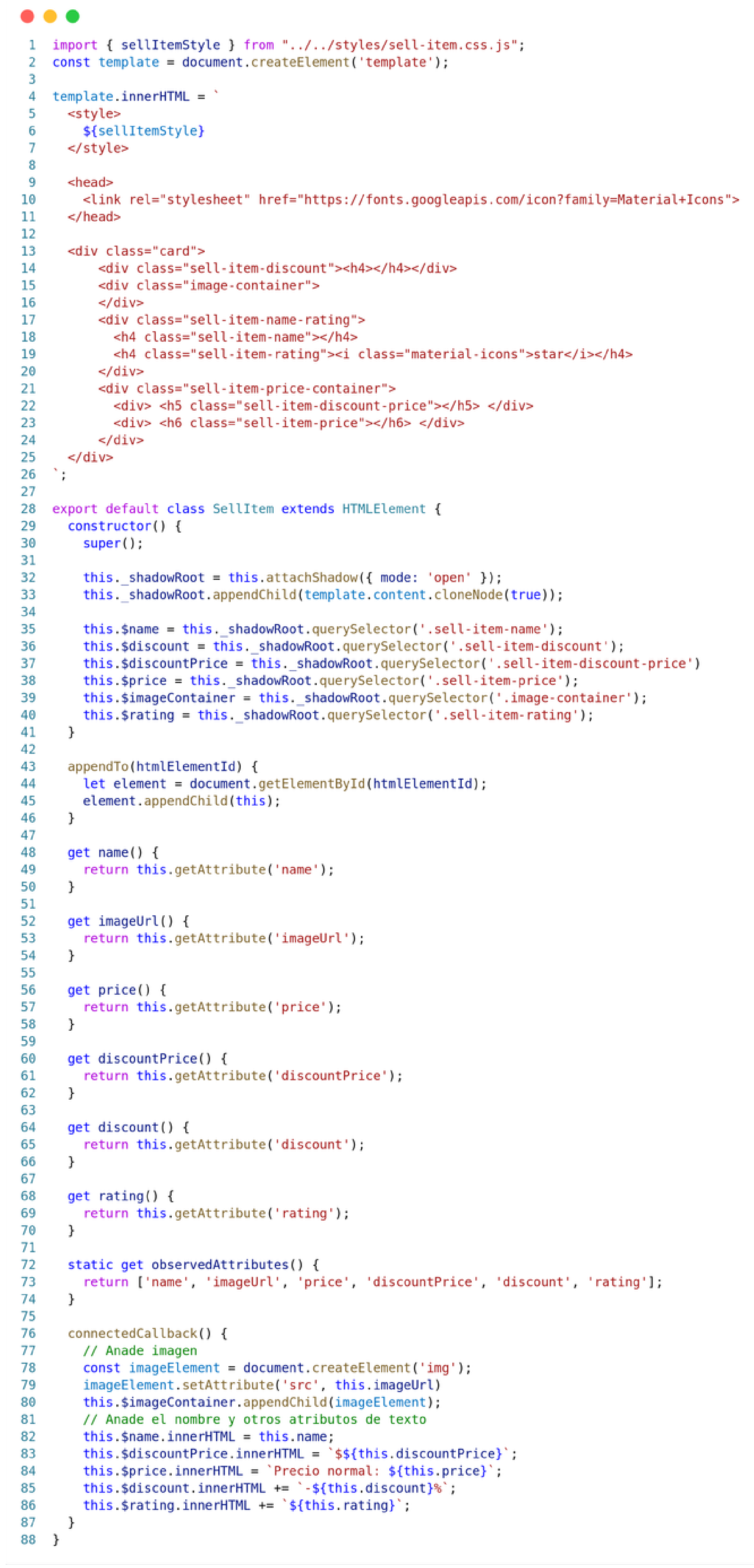


```
export default class LitSellItem extends LitElement {
    static get styles() {
        return [litSellItemStyle];
    }

    appendTo(htmlElementId) {
        let element = document.getElementById(htmlElementId);
        element.appendChild(this);
    }

    connectedCallback() {
        super.connectedCallback()
        this.name = this.getAttribute('name');
        this.imageUrl = this.getAttribute('imageUrl');
        this.price = this.getAttribute('price');
        this.discount = this.getAttribute('discount');
        this.discountPrice = this.getAttribute('discountPrice');
        this.rating = this.getAttribute('rating');
    }

    render() {
        return html`
        <head>
            <link rel="stylesheet" href="https://fonts.googleapis.com/icon?family=Material+Icons">
        </head>

        <div class="card">
            <div class="sell-item-discount">-${this.discount}%</div>
            <div class="image-container"><img src="${this.imageUrl}"></img></div>
            <div class="sell-item-name-rating">
                <h4 class="sell-item-name">${this.name}</h4>
                <h4 class="sell-item-rating"><i class="material-icons">star</i>${this.rating}</h4>
            </div>
            <div class="sell-item-price-container">
                <div> <h5 class="sell-item-discount-price">$${this.discountPrice}</h5> </div>
                <div> <h6 class="sell-item-price">Precio normal: $${this.price}</h6> </div>
            </div>
        </div>
```

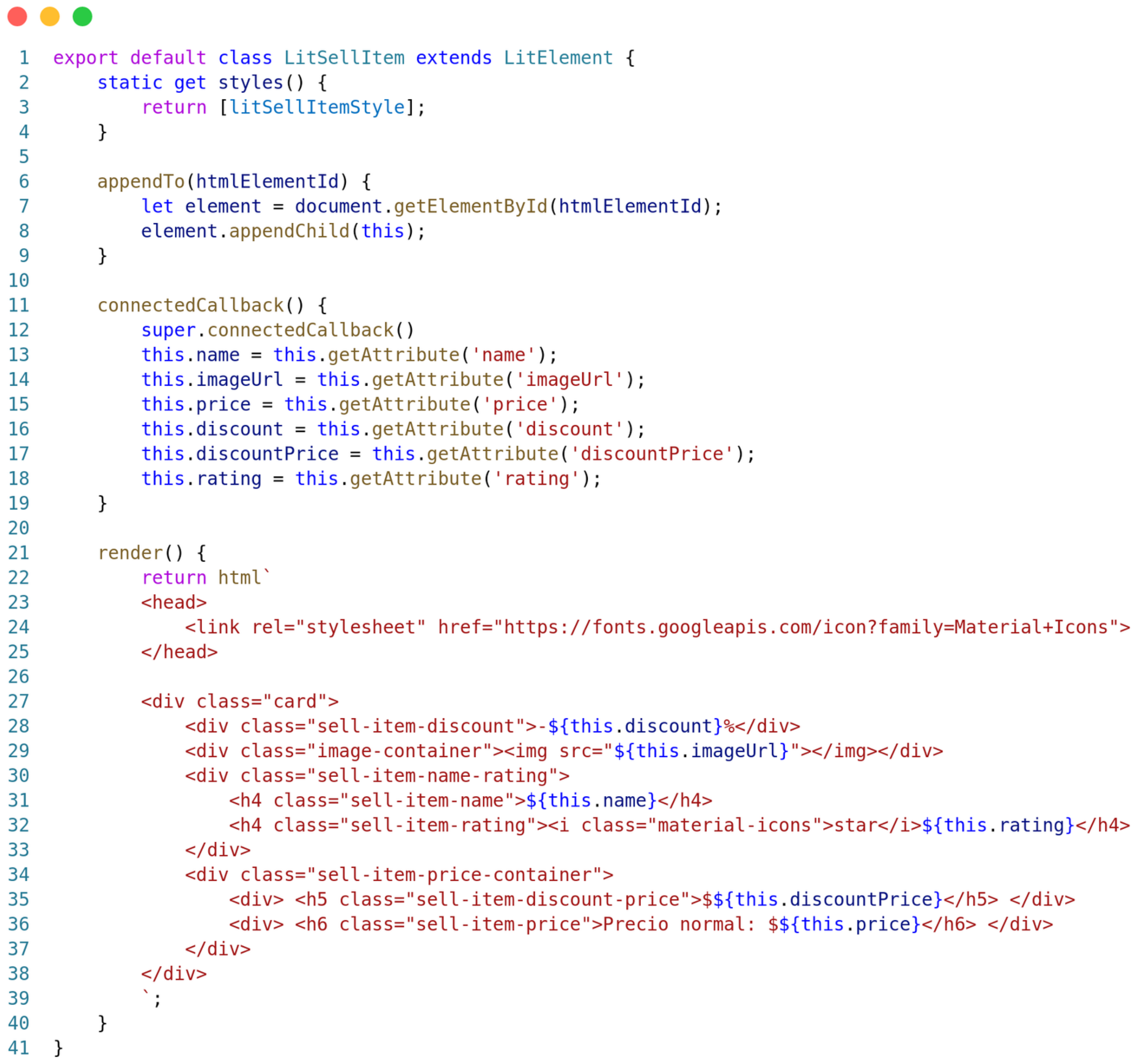
SELL-ITEM

V/S

LIT-SELL-ITEM

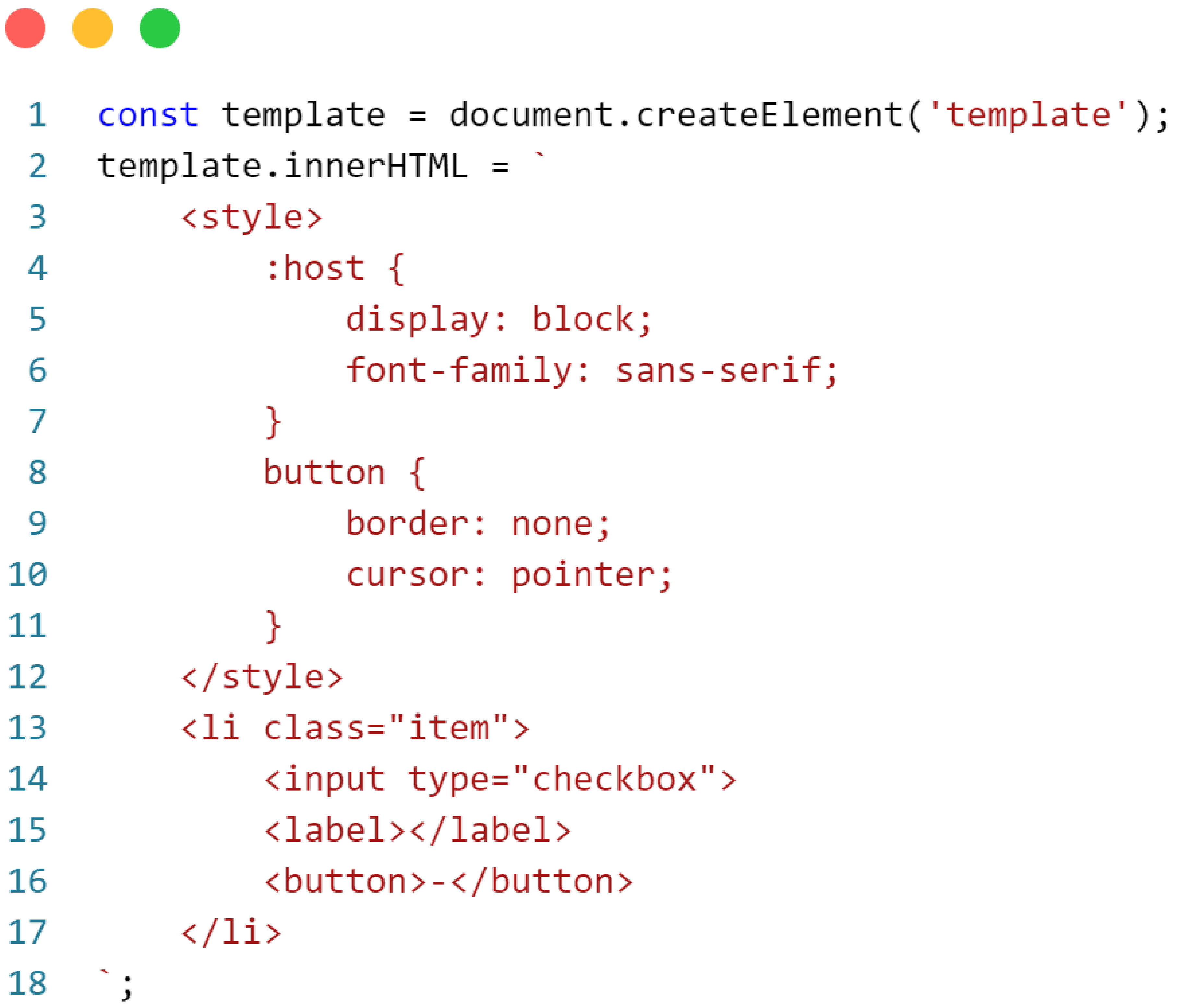


```
import { sellItemStyle } from "../../styles/sell-item.css.js";
const template = document.createElement('template');

template.innerHTML = `
  <style>
    ${sellItemStyle}
  </style>

  <head>
    <link rel="stylesheet" href="https://fonts.googleapis.com/icon?family=Material+Icons">
  </head>

  <div class="card">
      <div class="sell-item-discount"><h4></h4></div>
      <div class="image-container">
      </div>
      <div class="sell-item-name-rating">
        <h4 class="sell-item-name"></h4>
        <h4 class="sell-item-rating"><i class="material-icons">star</i></h4>
      </div>
      <div class="sell-item-price-container">
        <div> <h5 class="sell-item-discount-price"></h5> </div>
        <div> <h6 class="sell-item-price"></h6> </div>
      </div>
  </div>
`;

export default class SellItem extends HTMLElement {
  constructor() {
    super();

    this._shadowRoot = this.attachShadow({ mode: 'open' });
    this._shadowRoot.appendChild(template.content.cloneNode(true));

    this.$name = this._shadowRoot.querySelector('.sell-item-name');
    this.$discount = this._shadowRoot.querySelector('.sell-item-discount');
    this.$discountPrice = this._shadowRoot.querySelector('.sell-item-discount-price')
    this.$price = this._shadowRoot.querySelector('.sell-item-price');
    this.$imageContainer = this._shadowRoot.querySelector('.image-container');
    this.$rating = this._shadowRoot.querySelector('.sell-item-rating');
  }

  appendTo(htmlElementId) {
    let element = document.getElementById(htmlElementId);
    element.appendChild(this);
  }

  get name() {
    return this.getAttribute('name');
  }

  get imageUrl() {
    return this.getAttribute('imageUrl');
  }

  get price() {
    return this.getAttribute('price');
  }

  get discountPrice() {
    return this.getAttribute('discountPrice');
  }

  get discount() {
    return this.getAttribute('discount');
  }

  get rating() {
    return this.getAttribute('rating');
  }

  static get observedAttributes() {
    return ['name', 'imageUrl', 'price', 'discountPrice', 'discount', 'rating'];
  }

  connectedCallback() {
    // Anade imagen
    const imageElement = document.createElement('img');
    imageElement.setAttribute('src', this.imageUrl)
    this.$imageContainer.appendChild(imageElement);
    // Anade el nombre y otros atributos de texto
    this.$name.innerHTML = this.name;
    this.$discountPrice.innerHTML = `$${this.discountPrice}`;
    this.$price.innerHTML = `Precio normal: ${this.price}`;
    this.$discount.innerHTML += `-${this.discount}%`;
    this.$rating.innerHTML += `${this.rating}`;
  }
}
```

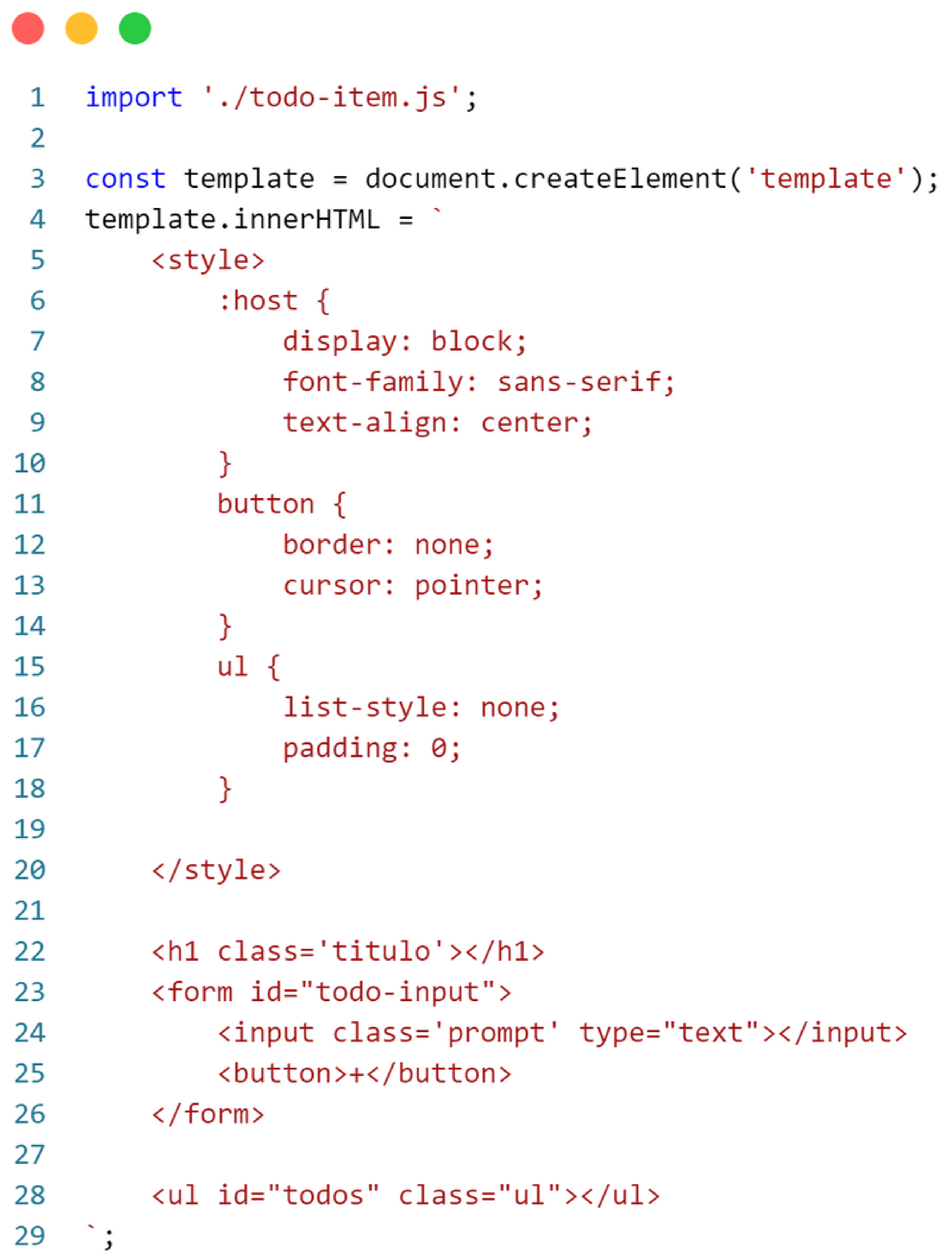


```
export default class LitSellItem extends LitElement {
    static get styles() {
        return [litSellItemStyle];
    }

    appendTo(htmlElementId) {
        let element = document.getElementById(htmlElementId);
        element.appendChild(this);
    }

    connectedCallback() {
        super.connectedCallback()
        this.name = this.getAttribute('name');
        this.imageUrl = this.getAttribute('imageUrl');
        this.price = this.getAttribute('price');
        this.discount = this.getAttribute('discount');
        this.discountPrice = this.getAttribute('discountPrice');
        this.rating = this.getAttribute('rating');
    }

    render() {
        return html`
        <head>
            <link rel="stylesheet" href="https://fonts.googleapis.com/icon?family=Material+Icons">
        </head>

        <div class="card">
            <div class="sell-item-discount">-${this.discount}%</div>
            <div class="image-container"><img src="${this.imageUrl}"></img></div>
            <div class="sell-item-name-rating">
                <h4 class="sell-item-name">${this.name}</h4>
                <h4 class="sell-item-rating"><i class="material-icons">star</i>${this.rating}</h4>
            </div>
            <div class="sell-item-price-container">
                <div> <h5 class="sell-item-discount-price">$${this.discountPrice}</h5> </div>
                <div> <h6 class="sell-item-price">Precio normal: $${this.price}</h6> </div>
            </div>
        </div>
        `;
    }
}
```

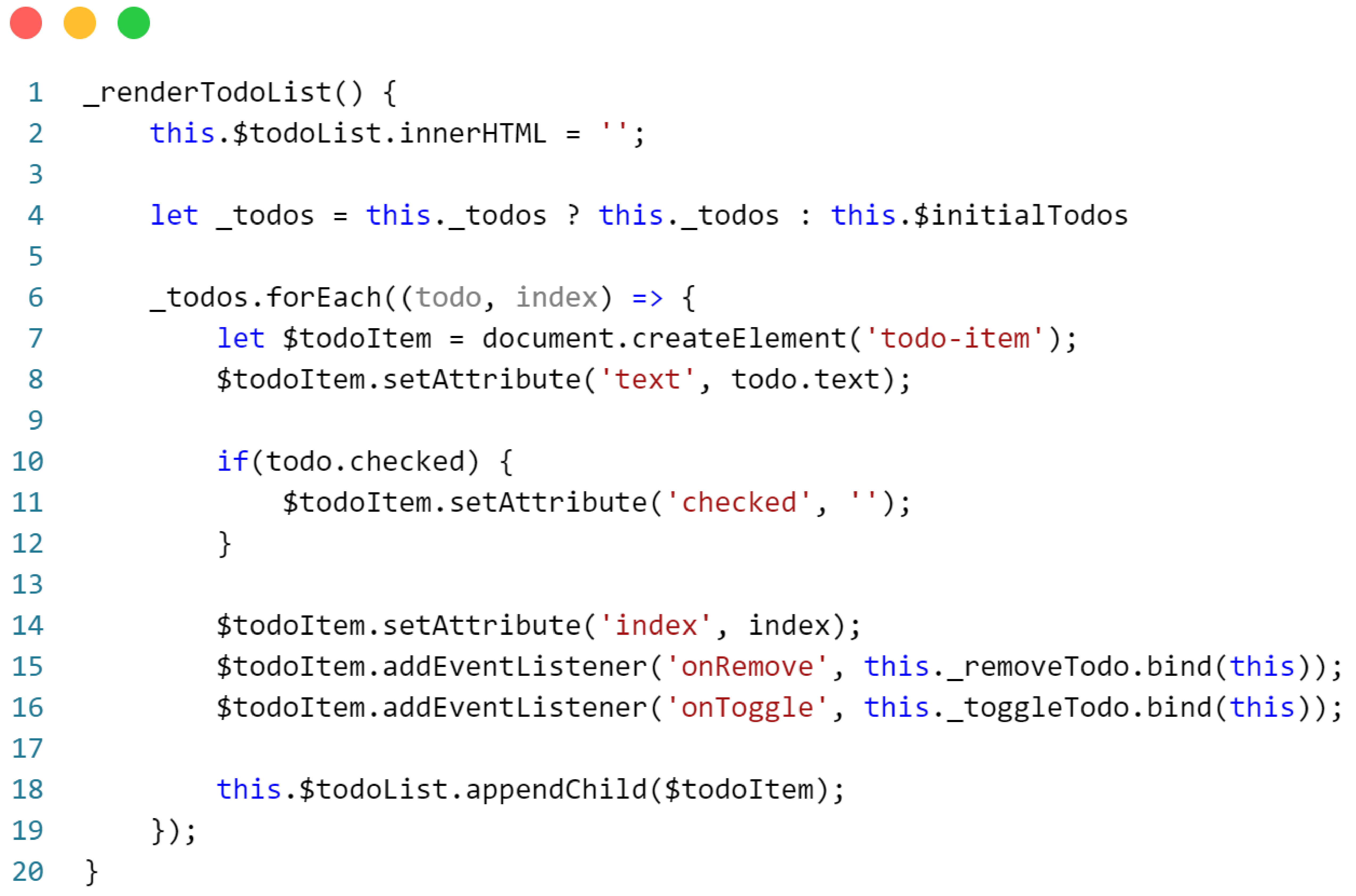
ToDo-List



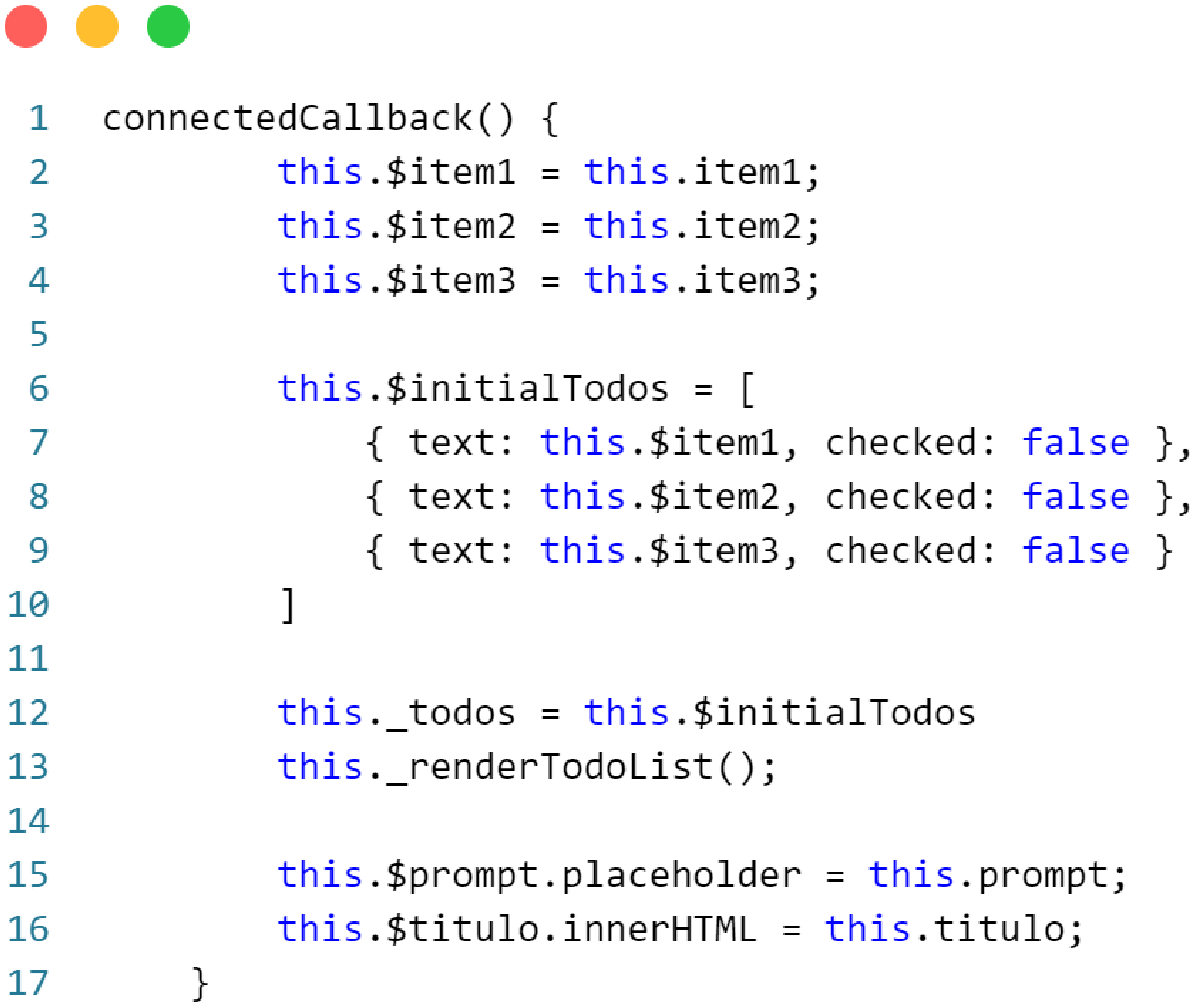
```
const template = document.createElement('template');
template.innerHTML = `
    <style>
        :host {
            display: block;
            font-family: sans-serif;
        }
        button {
            border: none;
            cursor: pointer;
        }
    </style>
    <li class="item">
        <input type="checkbox">
        <label></label>
        <button>-</button>
    </li>
`;
```



```
import './todo-item.js';

const template = document.createElement('template');
template.innerHTML = `
    <style>
        :host {
            display: block;
            font-family: sans-serif;
            text-align: center;
        }
        button {
            border: none;
            cursor: pointer;
        }
        ul {
            list-style: none;
            padding: 0;
        }

    </style>

    <h1 class='titulo'></h1>
    <form id="todo-input">
        <input class='prompt' type="text"></input>
        <button>+</button>
    </form>

    <ul id="todos" class="ul"></ul>
`;
```

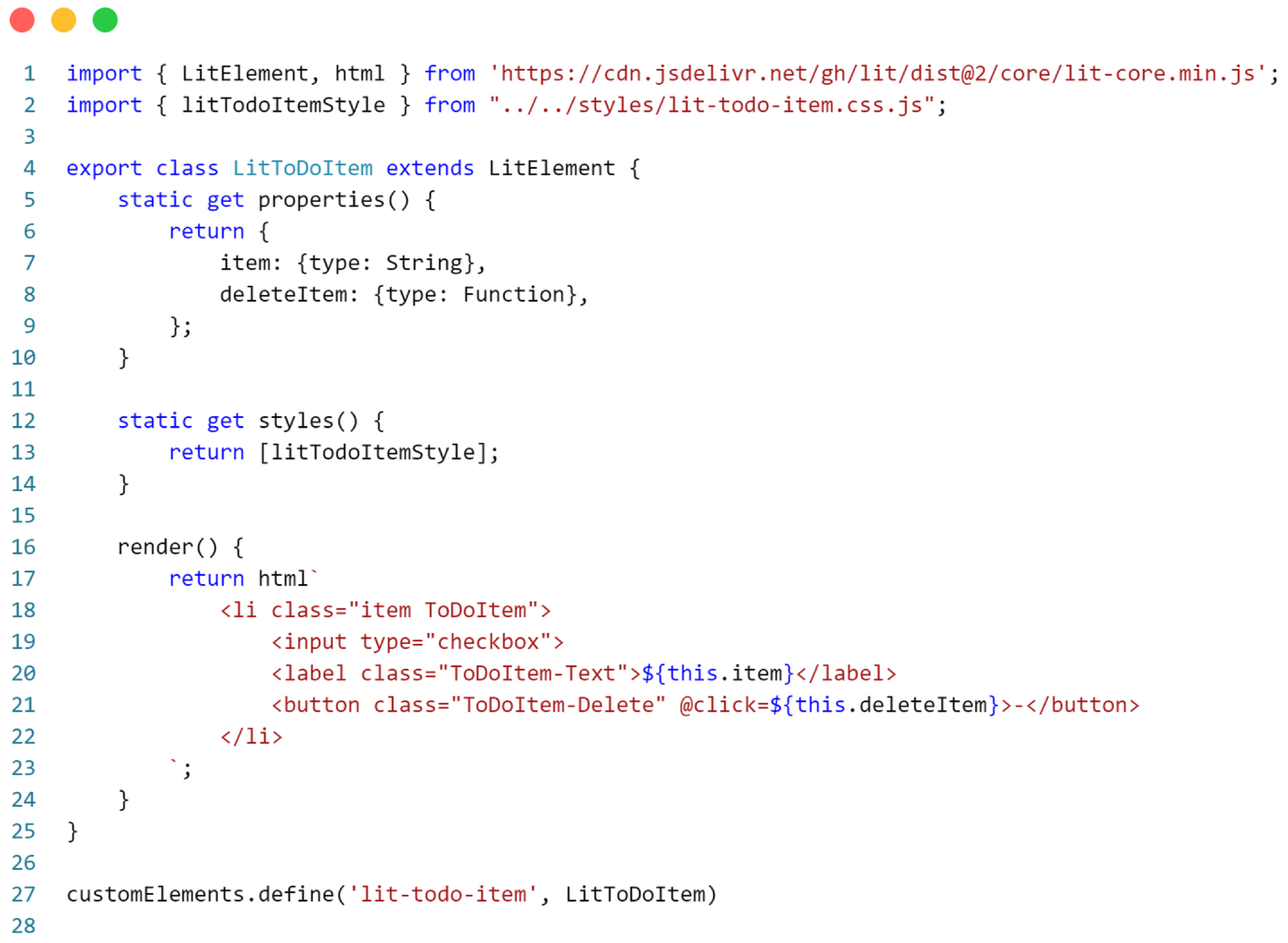


```
_renderTodoList() {
    this.$todoList.innerHTML = '';

    let _todos = this._todos ? this._todos : this.$initialTodos

    _todos.forEach((todo, index) => {
        let $todoItem = document.createElement('todo-item');
        $todoItem.setAttribute('text', todo.text);

        if(todo.checked) {
            $todoItem.setAttribute('checked', '');
        }

        $todoItem.setAttribute('index', index);
        $todoItem.addEventListener('onRemove', this._removeTodo.bind(this));
        $todoItem.addEventListener('onToggle', this._toggleTodo.bind(this));

        this.$todoList.appendChild($todoItem);
    });
}
```

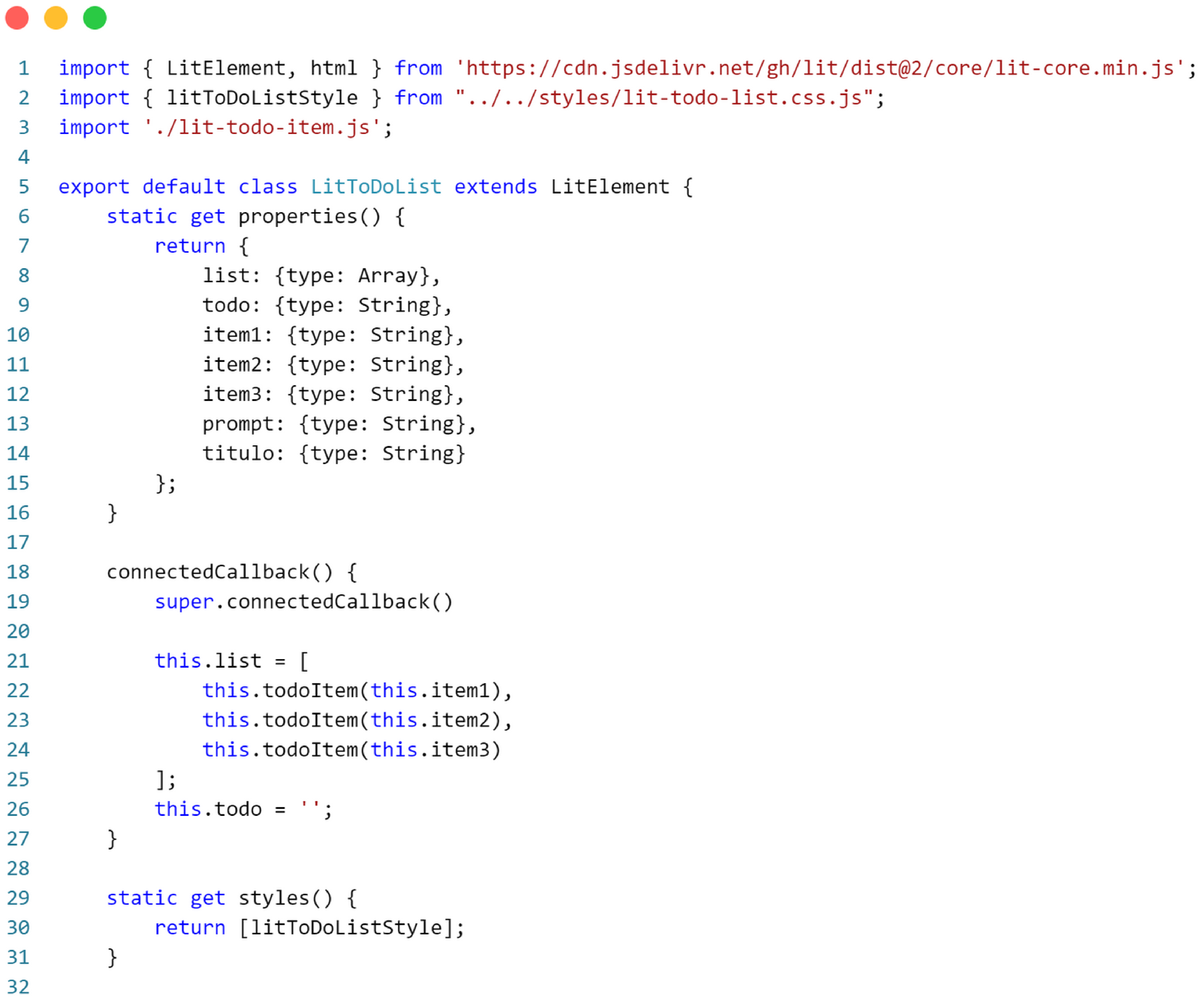


```
connectedCallback() {
        this.$item1 = this.item1;
        this.$item2 = this.item2;
        this.$item3 = this.item3;

        this.$initialTodos = [
            { text: this.$item1, checked: false },
            { text: this.$item2, checked: false },
            { text: this.$item3, checked: false }
        ]

        this._todos = this.$initialTodos
        this._renderTodoList();

        this.$prompt.placeholder = this.prompt;
        this.$titulo.innerHTML = this.titulo;
    }
```

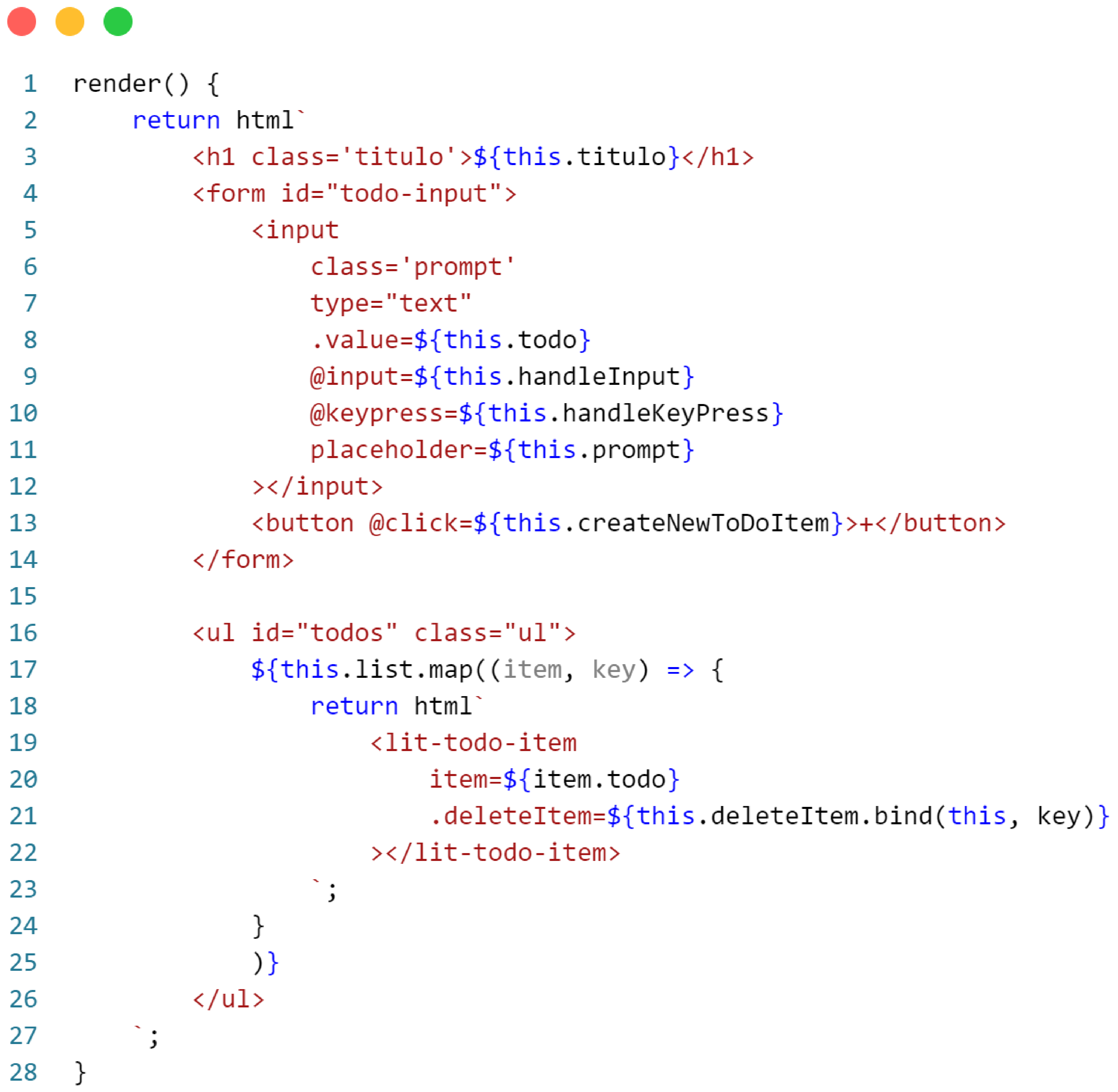
Lit -ToDo-List



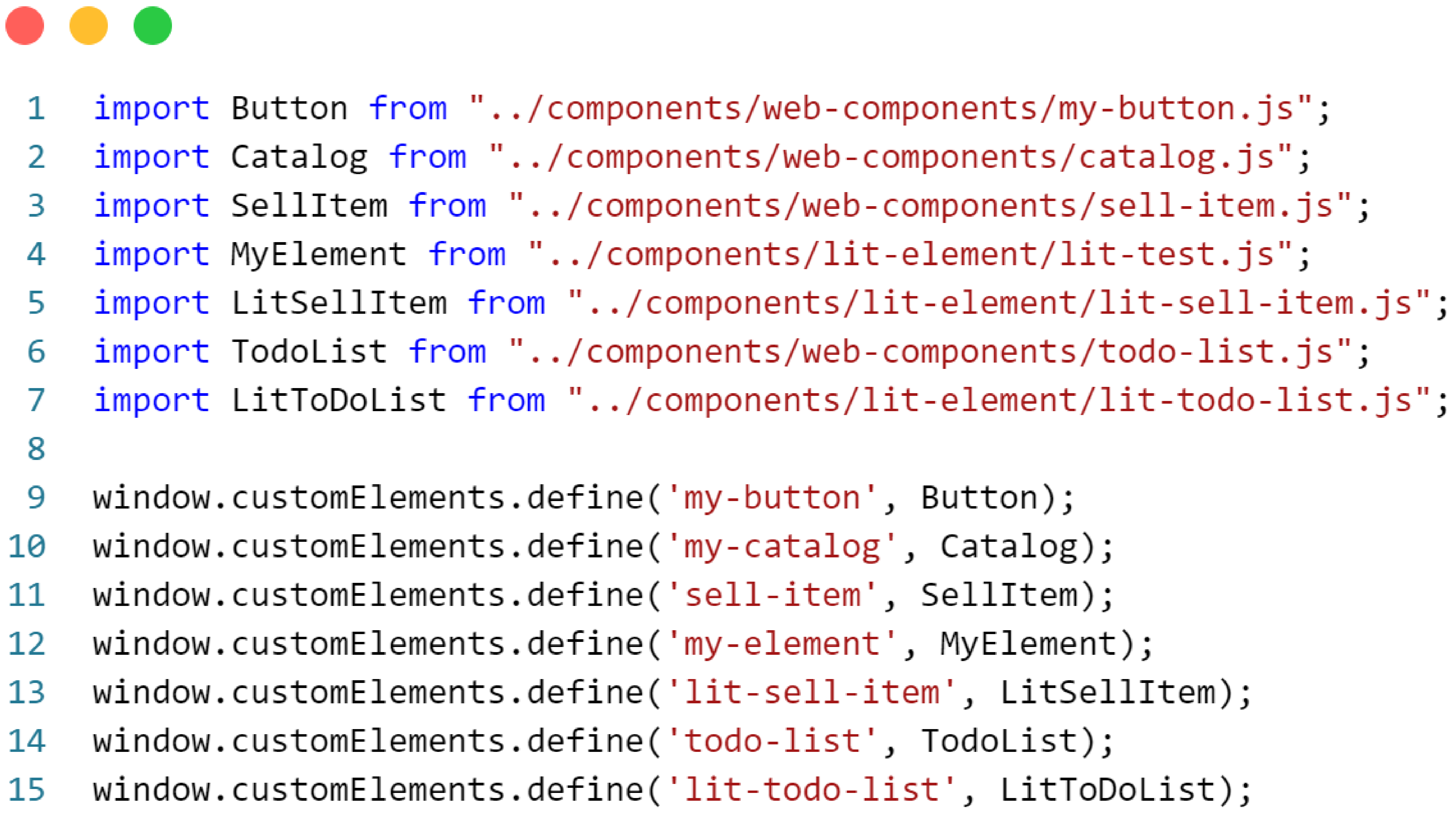
```
import { LitElement, html } from 'https://cdn.jsdelivr.net/gh/lit/dist@2/core/lit-core.min.js';
import { litTodoItemStyle } from "../../styles/lit-todo-item.css.js";

export class LitToDoItem extends LitElement {
    static get properties() {
        return {
            item: {type: String},
            deleteItem: {type: Function},
        };
    }

    static get styles() {
        return [litTodoItemStyle];
    }

    render() {
        return html`
            <li class="item ToDoItem">
                <input type="checkbox">
                <label class="ToDoItem-Text">${this.item}</label>
                <button class="ToDoItem-Delete" @click=${this.deleteItem}>-</button>
            </li>
        `;
    }
}

customElements.define('lit-todo-item', LitToDoItem)
```



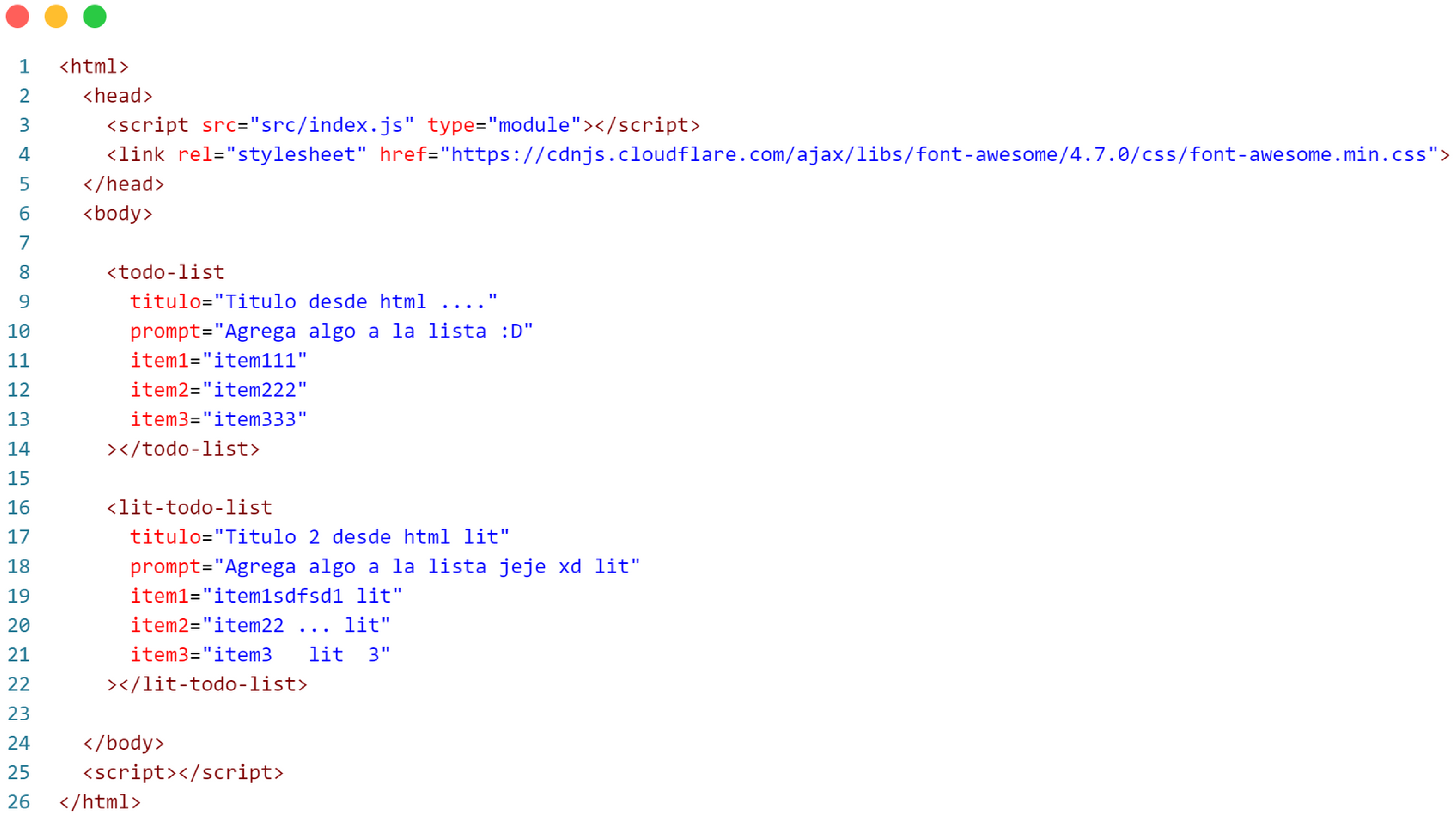
```
import { LitElement, html } from 'https://cdn.jsdelivr.net/gh/lit/dist@2/core/lit-core.min.js';
import { litToDoListStyle } from "../../styles/lit-todo-list.css.js";
import './lit-todo-item.js';

export default class LitToDoList extends LitElement {
    static get properties() {
        return {
            list: {type: Array},
            todo: {type: String},
            item1: {type: String},
            item2: {type: String},
            item3: {type: String},
            prompt: {type: String},
            titulo: {type: String}
        };
    }

    connectedCallback() {
        super.connectedCallback()

        this.list = [
            this.todoItem(this.item1),
            this.todoItem(this.item2),
            this.todoItem(this.item3)
        ];
        this.todo = '';
    }

    static get styles() {
        return [litToDoListStyle];
    }

```



```
render() {
    return html`
        <h1 class='titulo'>${this.titulo}</h1>
        <form id="todo-input">
            <input
                class='prompt'
                type="text"
                .value=${this.todo}
                @input=${this.handleInput}
                @keypress=${this.handleKeyPress}
                placeholder=${this.prompt}
            ></input>
            <button @click=${this.createNewToDoItem}>+</button>
        </form>

        <ul id="todos" class="ul">
            ${this.list.map((item, key) => {
                return html`
                    <lit-todo-item
                        item=${item.todo}
                        .deleteItem=${this.deleteItem.bind(this, key)}
                    ></lit-todo-item>
                `;
            }
            )}
        </ul>
    `;
}
```



```javascript
import Button from "../components/web-components/my-button.js";
import Catalog from "../components/web-components/catalog.js";
import SellItem from "../components/web-components/sell-item.js";
import MyElement from "../components/lit-element/lit-test.js";
import LitSellItem from "../components/lit-element/lit-sell-item.js";
import TodoList from "../components/web-components/todo-list.js";
import LitToDoList from "../components/lit-element/lit-todo-list.js";

window.customElements.define('my-button', Button);
window.customElements.define('my-catalog', Catalog);
window.customElements.define('sell-item', SellItem);
window.customElements.define('my-element', MyElement);
window.customElements.define('lit-sell-item', LitSellItem);
window.customElements.define('todo-list', TodoList);
window.customElements.define('lit-todo-list', LitToDoList);
```



```html
<html>
  <head>
    <script src="src/index.js" type="module"></script>
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
  </head>
  <body>

    <todo-list
      titulo="Titulo desde html ...."
      prompt="Agrega algo a la lista :D"
      item1="item111"
      item2="item222"
      item3="item333"
    ></todo-list>

    <lit-todo-list
      titulo="Titulo 2 desde html lit"
      prompt="Agrega algo a la lista jeje xd lit"
      item1="item1sdfsd1 lit"
      item2="item22 ... lit"
      item3="item3   lit  3"
    ></lit-todo-list>

  </body>
  <script></script>
</html>
```

DIFICULTADES Y CONCLUSIONES

Requiere un manejo manual del DOM con Javascript

Web components son complejos en relación a los componentes de frameworks

Mayor flexibilidad para modificar los componentes (por separado y según necesidad)

Una forma interesante de crear componentes transversales al framework

Se permite encapsulamiento de estilos y comportamiento con el shadowDOM

T7: Web Components

Javiera Inostroza, Elías Sabja, Samuel Zúñiga