Web Components

Grupo 1:
Joaquín Cáceres
Julián García
Mathias Valdebenito

Webcomponents v/s Lit

V/S

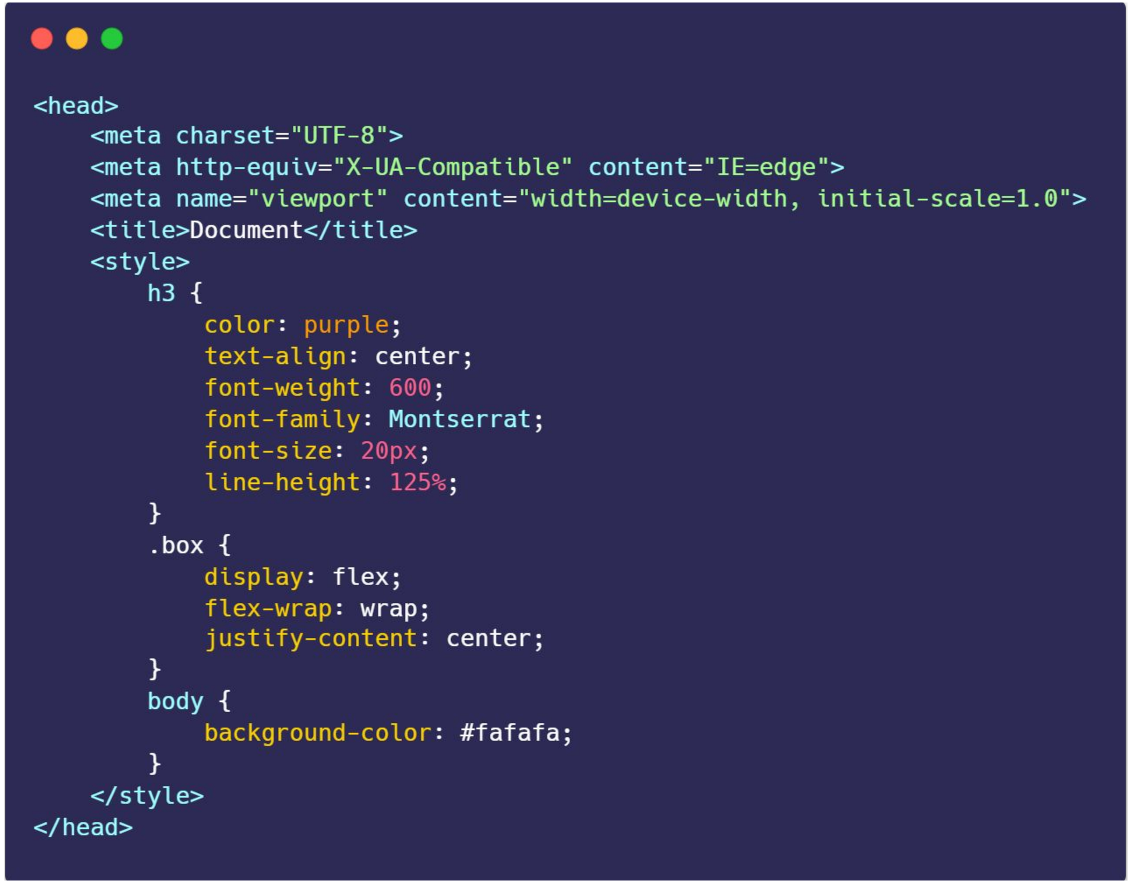
Lit

01

Demo Item para vender

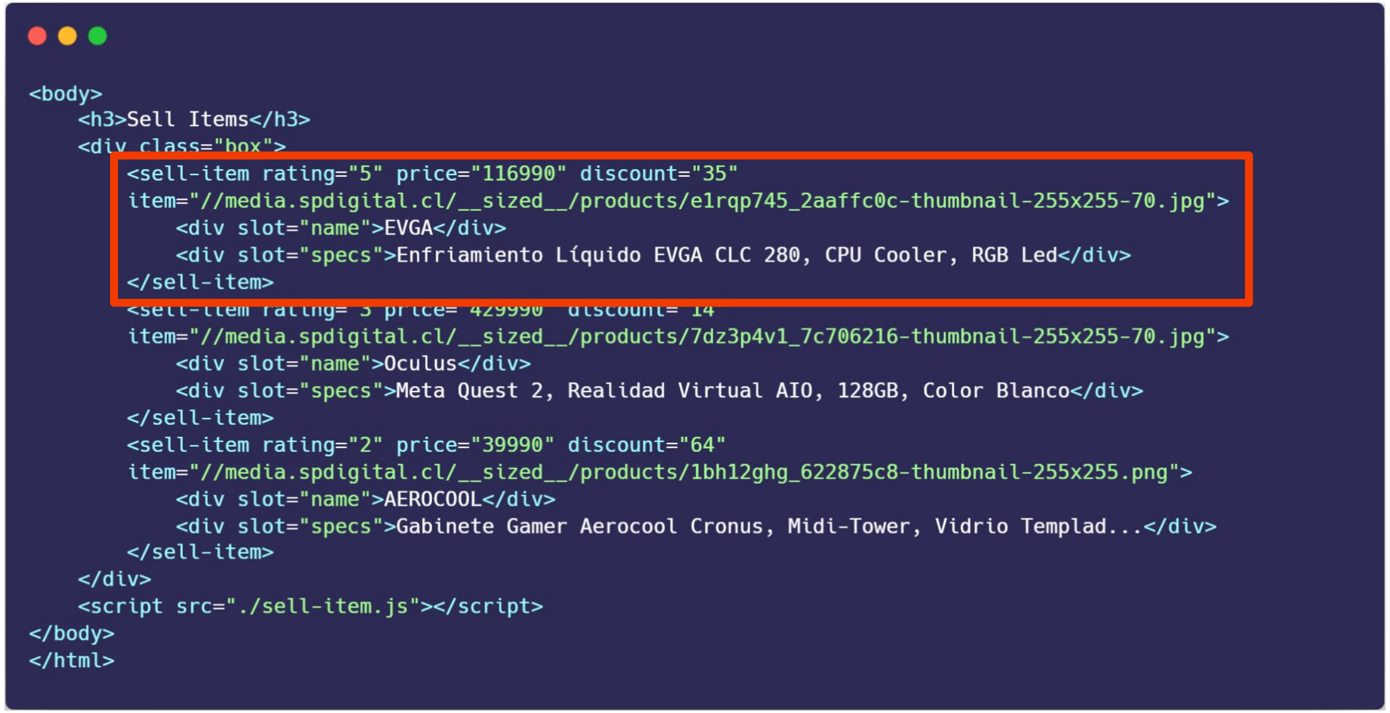
Componente sell-item

style



```
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <style>
        h3 {
            color: purple;
            text-align: center;
            font-weight: 600;
            font-family: Montserrat;
            font-size: 20px;
            line-height: 125%;
        }
        .box {
            display: flex;
            flex-wrap: wrap;
            justify-content: center;
        }
        body {
            background-color: #fafafa;
        }
    </style>
</head>
```

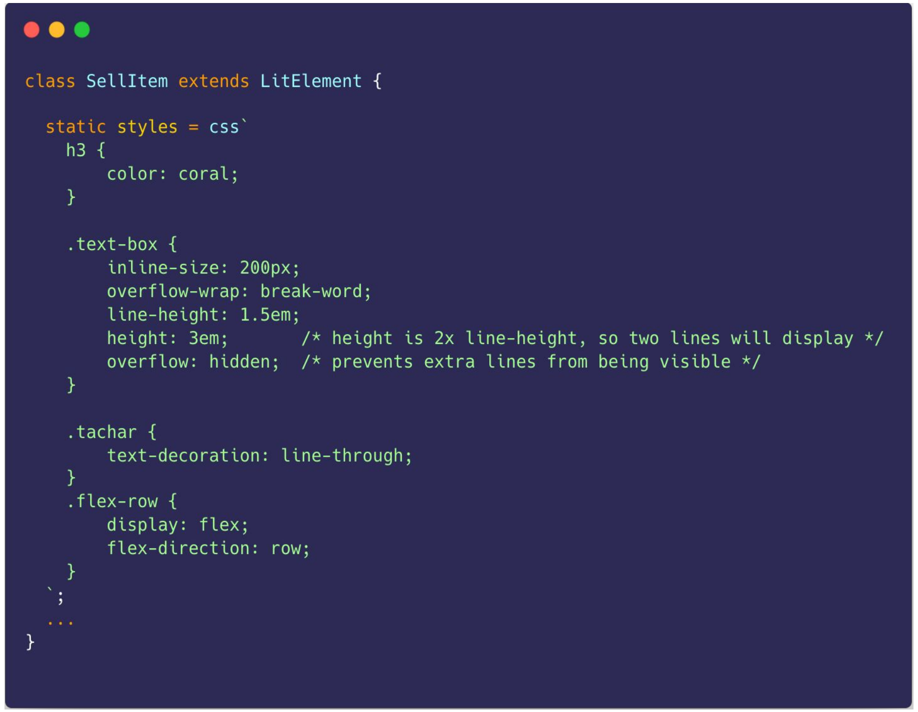
Componente sell-item



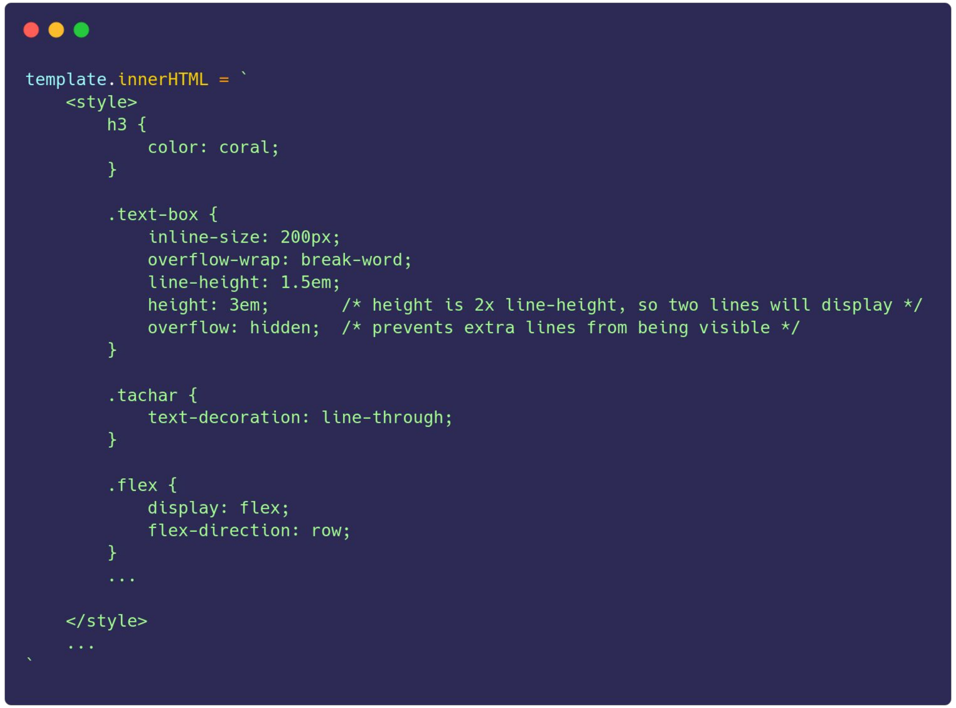
```html
<body>
    <h3>Sell Items</h3>
    <div class="box">
        <sell-item rating="5" price="116990" discount="35"
        item="//media.spdigital.cl/__sized__/products/e1rqp745_2aaffc0c-thumbnail-255x255-70.jpg">
            <div slot="name">EVGA</div>
            <div slot="specs">Enfriamiento Líquido EVGA CLC 280, CPU Cooler, RGB Led</div>
        </sell-item>
        <sell-item rating="3" price="429990" discount="14"
        item="//media.spdigital.cl/__sized__/products/7dz3p4v1_7c706216-thumbnail-255x255-70.jpg">
            <div slot="name">Oculus</div>
            <div slot="specs">Meta Quest 2, Realidad Virtual AIO, 128GB, Color Blanco</div>
        </sell-item>
        <sell-item rating="2" price="39990" discount="64"
        item="//media.spdigital.cl/__sized__/products/1bh12ghg_622875c8-thumbnail-255x255.png">
            <div slot="name">AEROCOOL</div>
            <div slot="specs">Gabinete Gamer Aerocool Cronus, Midi-Tower, Vidrio Templad...</div>
        </sell-item>
    </div>
    <script src="./sell-item.js"></script>
</body>
</html>
```

Componente sell-item

Scoped-style

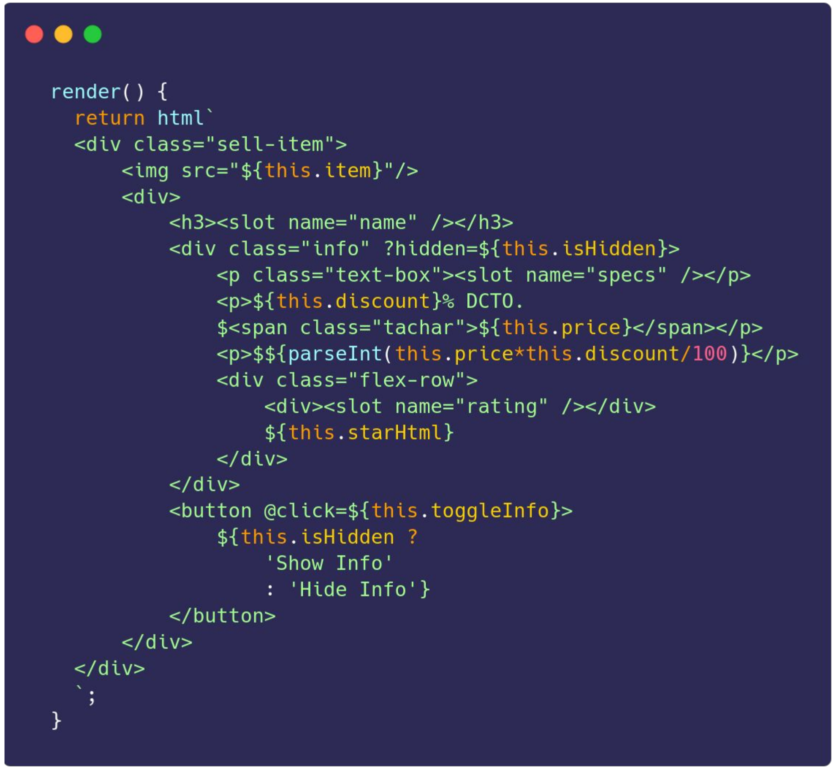


```
class SellItem extends LitElement {

  static styles = css`
    h3 {
        color: coral;
    }

    .text-box {
        inline-size: 200px;
        overflow-wrap: break-word;
        line-height: 1.5em;
        height: 3em;       /* height is 2x line-height, so two lines will display */
        overflow: hidden;  /* prevents extra lines from being visible */
    }

    .tachar {
        text-decoration: line-through;
    }
    .flex-row {
        display: flex;
        flex-direction: row;
    }
  `;
  ...
}
```

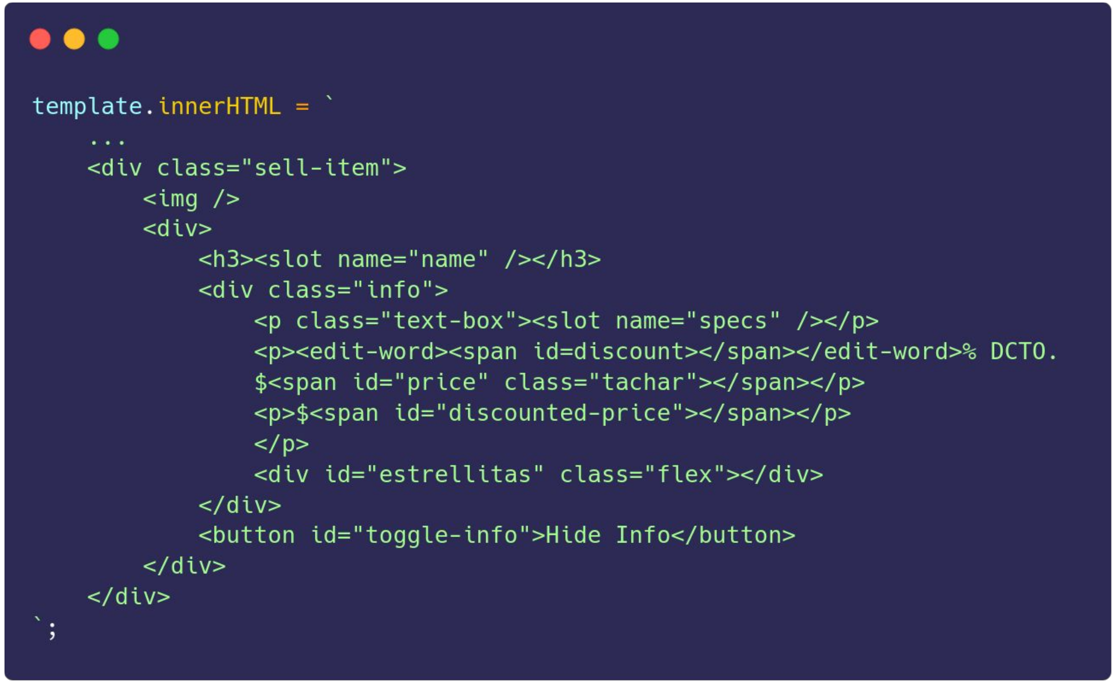


```
template.innerHTML = `
    <style>
        h3 {
            color: coral;
        }

        .text-box {
            inline-size: 200px;
            overflow-wrap: break-word;
            line-height: 1.5em;
            height: 3em;       /* height is 2x line-height, so two lines will display */
            overflow: hidden;  /* prevents extra lines from being visible */
        }

        .tachar {
            text-decoration: line-through;
        }

        .flex {
            display: flex;
            flex-direction: row;
        }
        ...

    </style>
    ...
`
```

Componente sell-item

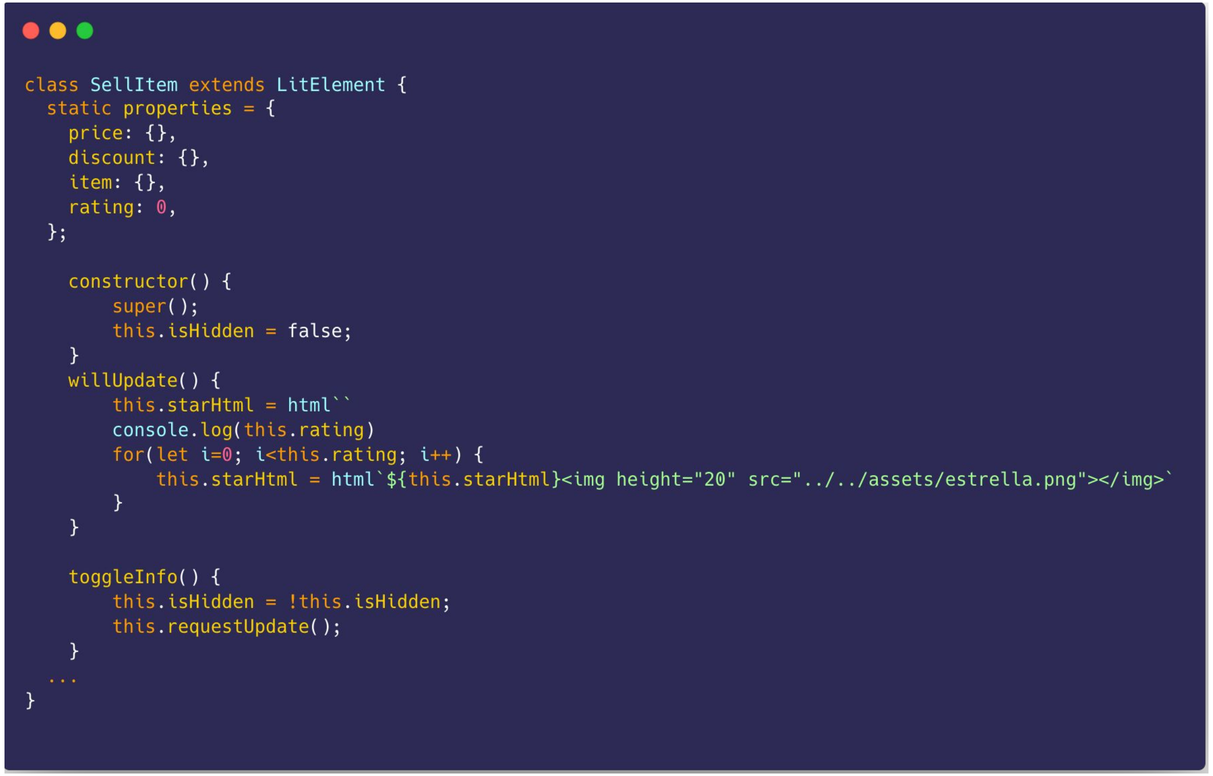

```
render() {
  return html`
  <div class="sell-item">
      <img src="${this.item}"/>
      <div>
          <h3><slot name="name" /></h3>
          <div class="info" ?hidden=${this.isHidden}>
              <p class="text-box"><slot name="specs" /></p>
              <p>${this.discount}% DCTO.
              $<span class="tachar">${this.price}</span></p>
              <p>$${parseInt(this.price*this.discount/100)}</p>
              <div class="flex-row">
                  <div><slot name="rating" /></div>
                  ${this.starHtml}
              </div>
          </div>
          <button @click=${this.toggleInfo}>
              ${this.isHidden ?
                  'Show Info'
                  : 'Hide Info'}
          </button>
      </div>
  </div>
  `;
}
```

Componente sell-item



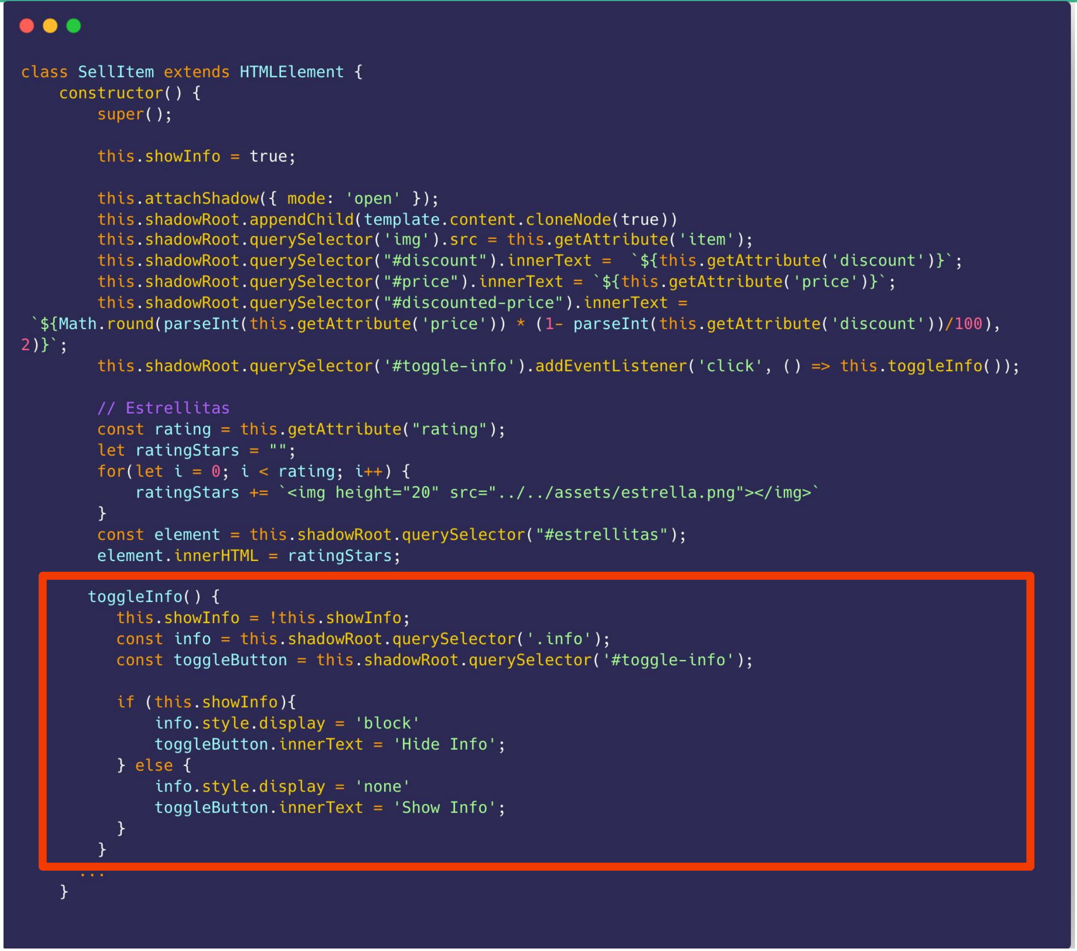
```
template.innerHTML = `
    ...
    <div class="sell-item">
        <img />
        <div>
            <h3><slot name="name" /></h3>
            <div class="info">
                <p class="text-box"><slot name="specs" /></p>
                <p><edit-word><span id=discount></span></edit-word>% DCTO.
                $<span id="price" class="tachar"></span></p>
                <p>$<span id="discounted-price"></span></p>
                </p>
                <div id="estrellitas" class="flex"></div>
            </div>
            <button id="toggle-info">Hide Info</button>
        </div>
    </div>
`;
```

Componente sell-item



```
class SellItem extends LitElement {
  static properties = {
    price: {},
    discount: {},
    item: {},
    rating: 0,
  };

    constructor() {
        super();
        this.isHidden = false;
    }
    willUpdate() {
        this.starHtml = html``
        console.log(this.rating)
        for(let i=0; i<this.rating; i++) {
            this.starHtml = html`${this.starHtml}<img height="20" src="../../assets/estrella.png"></img>`
        }
    }

    toggleInfo() {
        this.isHidden = !this.isHidden;
        this.requestUpdate();
    }
  ...
}
```

Componente sell-item



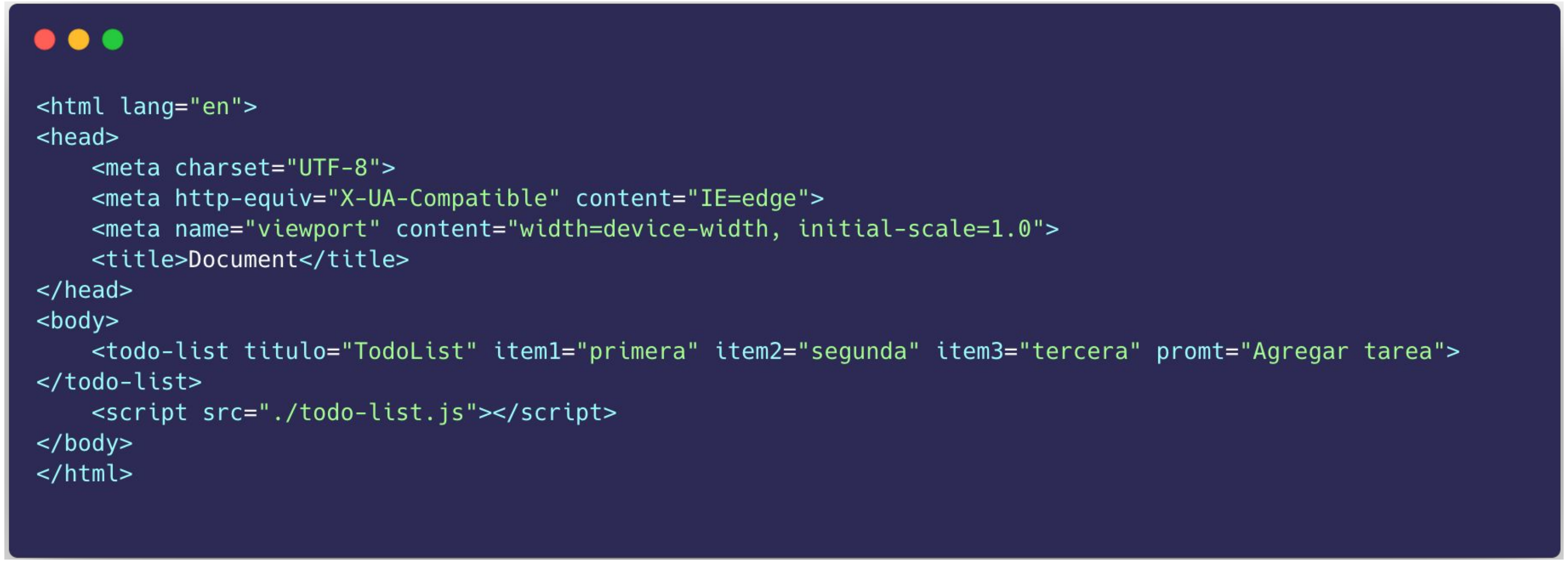
```
class SellItem extends HTMLElement {
    constructor() {
        super();

        this.showInfo = true;

        this.attachShadow({ mode: 'open' });
        this.shadowRoot.appendChild(template.content.cloneNode(true))
        this.shadowRoot.querySelector('img').src = this.getAttribute('item');
        this.shadowRoot.querySelector("#discount").innerText =  `${this.getAttribute('discount')}`;
        this.shadowRoot.querySelector("#price").innerText = `${this.getAttribute('price')}`;
        this.shadowRoot.querySelector("#discounted-price").innerText =
 `${Math.round(parseInt(this.getAttribute('price')) * (1- parseInt(this.getAttribute('discount'))/100),
2)}`;
        this.shadowRoot.querySelector('#toggle-info').addEventListener('click', () => this.toggleInfo());

        // Estrellitas
        const rating = this.getAttribute("rating");
        let ratingStars = "";
        for(let i = 0; i < rating; i++) {
            ratingStars += `<img height="20" src="../../assets/estrella.png"></img>`
        }
        const element = this.shadowRoot.querySelector("#estrellitas");
        element.innerHTML = ratingStars;

       toggleInfo() {
          this.showInfo = !this.showInfo;
          const info = this.shadowRoot.querySelector('.info');
          const toggleButton = this.shadowRoot.querySelector('#toggle-info');

          if (this.showInfo){
              info.style.display = 'block'
              toggleButton.innerText = 'Hide Info';
          } else {
              info.style.display = 'none'
              toggleButton.innerText = 'Show Info';
          }
        }
      ...
    }
```

Demo

02

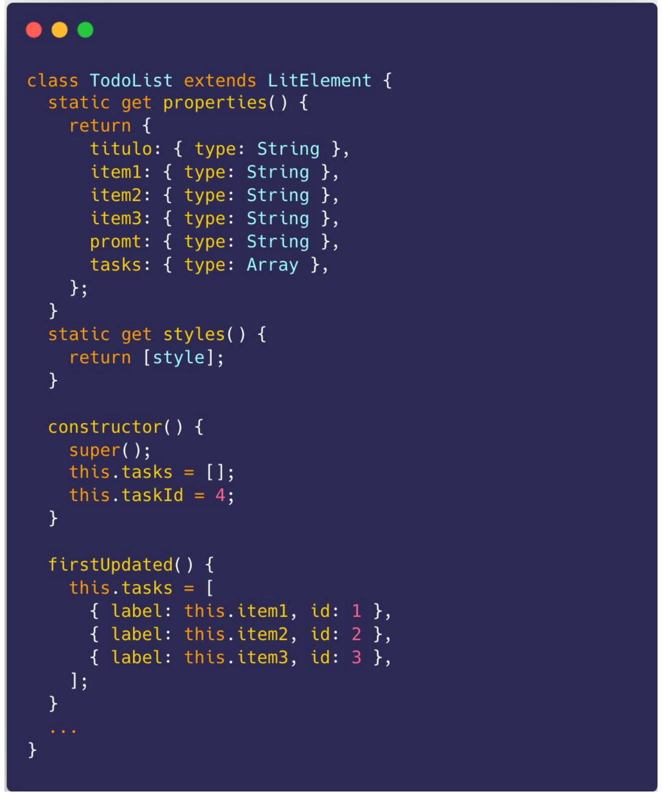
Demo Todo List

Componente todo-ítem

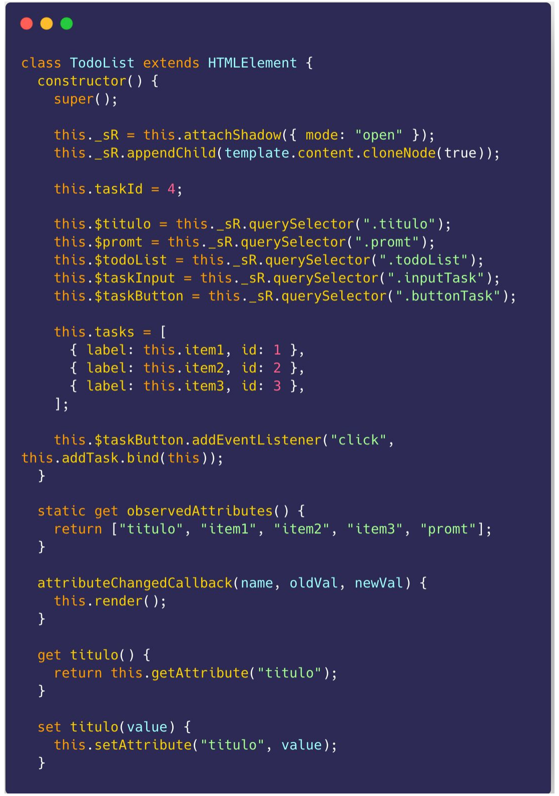


```
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <todo-list titulo="TodoList" item1="primera" item2="segunda" item3="tercera" promt="Agregar tarea">
</todo-list>
    <script src="./todo-list.js"></script>
</body>
</html>
```

Componente todo-ítem

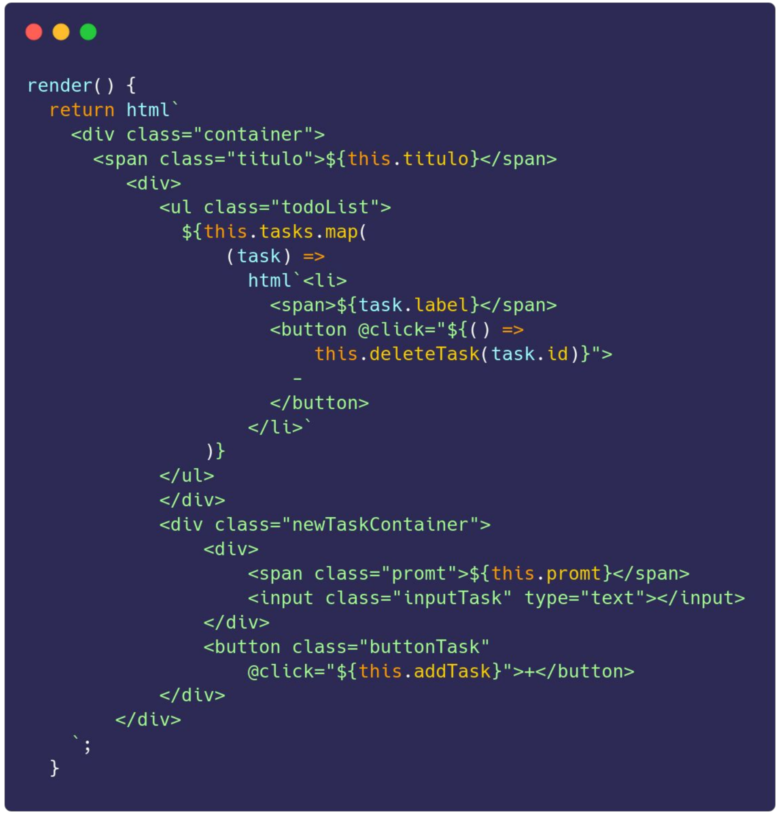


```
class TodoList extends LitElement {
  static get properties() {
    return {
      titulo: { type: String },
      item1: { type: String },
      item2: { type: String },
      item3: { type: String },
      promt: { type: String },
      tasks: { type: Array },
    };
  }
  static get styles() {
    return [style];
  }

  constructor() {
    super();
    this.tasks = [];
    this.taskId = 4;
  }

  firstUpdated() {
    this.tasks = [
      { label: this.item1, id: 1 },
      { label: this.item2, id: 2 },
      { label: this.item3, id: 3 },
    ];
  }
  ...
}
```

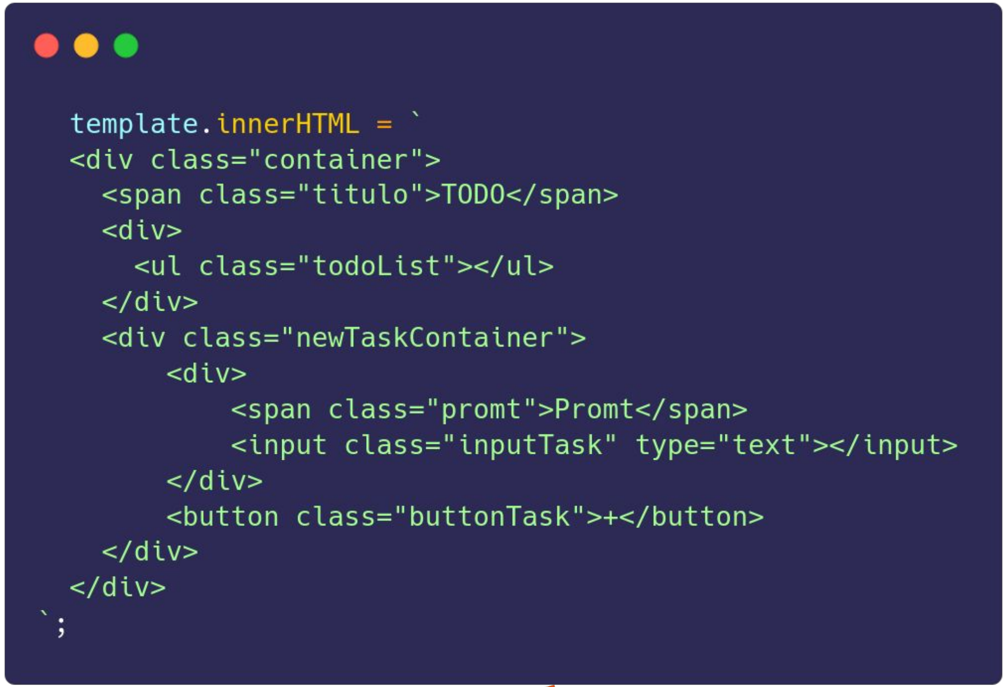


```
class TodoList extends HTMLElement {
  constructor() {
    super();

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

    this.taskId = 4;

    this.$titulo = this._sR.querySelector(".titulo");
    this.$promt = this._sR.querySelector(".promt");
    this.$todoList = this._sR.querySelector(".todoList");
    this.$taskInput = this._sR.querySelector(".inputTask");
    this.$taskButton = this._sR.querySelector(".buttonTask");

    this.tasks = [
      { label: this.item1, id: 1 },
      { label: this.item2, id: 2 },
      { label: this.item3, id: 3 },
    ];

    this.$taskButton.addEventListener("click",
this.addTask.bind(this));
  }

  static get observedAttributes() {
    return ["titulo", "item1", "item2", "item3", "promt"];
  }

  attributeChangedCallback(name, oldVal, newVal) {
    this.render();
  }

  get titulo() {
    return this.getAttribute("titulo");
  }

  set titulo(value) {
    this.setAttribute("titulo", value);
  }
```

Componente todo-ítem

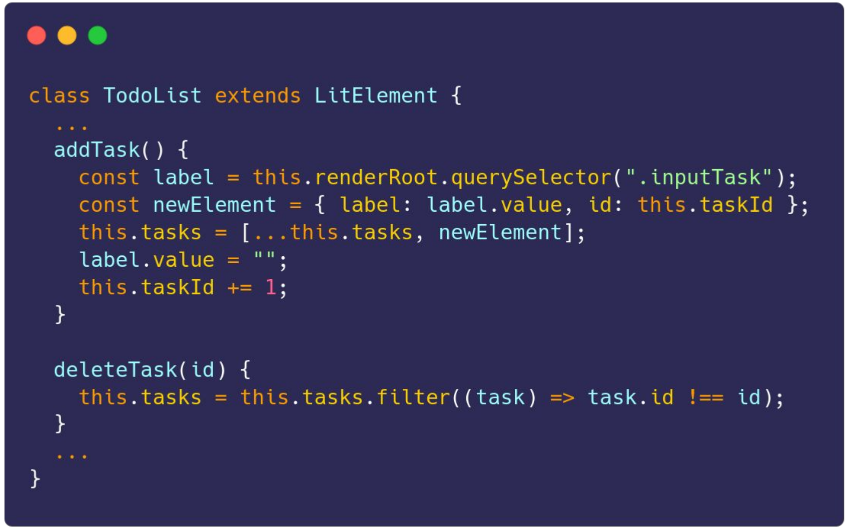


```
render() {
  return html`
    <div class="container">
      <span class="titulo">${this.titulo}</span>
         <div>
            <ul class="todoList">
              ${this.tasks.map(
                  (task) =>
                    html`<li>
                      <span>${task.label}</span>
                      <button @click="${() =>
                          this.deleteTask(task.id)}">
                        -
                      </button>
                    </li>`
                )}
            </ul>
            </div>
            <div class="newTaskContainer">
                <div>
                    <span class="promt">${this.promt}</span>
                    <input class="inputTask" type="text"></input>
                </div>
                <button class="buttonTask"
                    @click="${this.addTask}">+</button>
            </div>
        </div>
    `;
  }
```

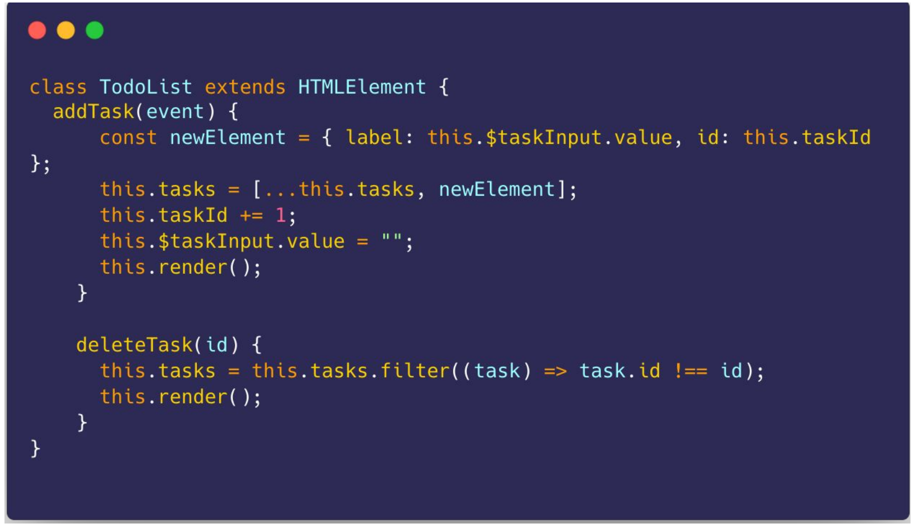


```
  template.innerHTML = `
  <div class="container">
    <span class="titulo">TODO</span>
    <div>
      <ul class="todoList"></ul>
    </div>
    <div class="newTaskContainer">
        <div>
            <span class="promt">Promt</span>
            <input class="inputTask" type="text"></input>
        </div>
        <button class="buttonTask">+</button>
    </div>
  </div>
`;
```

Componente todo-ítem



```
class TodoList extends LitElement {
  ...
  addTask() {
    const label = this.renderRoot.querySelector(".inputTask");
    const newElement = { label: label.value, id: this.taskId };
    this.tasks = [...this.tasks, newElement];
    label.value = "";
    this.taskId += 1;
  }

  deleteTask(id) {
    this.tasks = this.tasks.filter((task) => task.id !== id);
  }
  ...
}
```

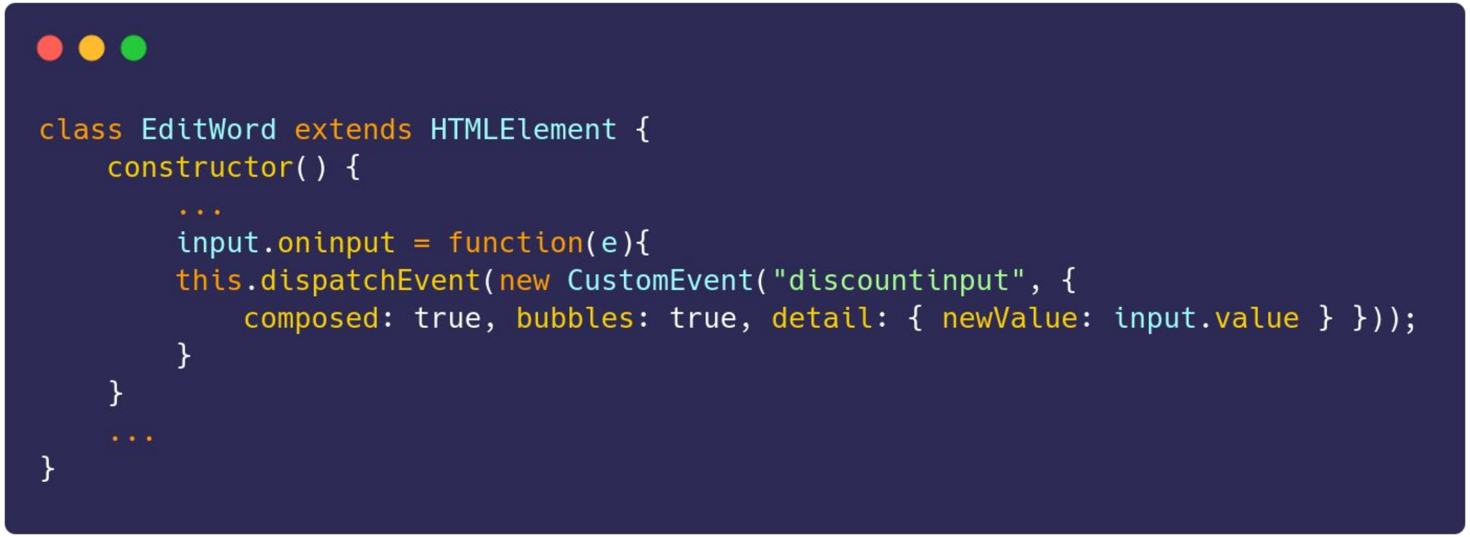


```
class TodoList extends HTMLElement {
  addTask(event) {
      const newElement = { label: this.$taskInput.value, id: this.taskId
};
      this.tasks = [...this.tasks, newElement];
      this.taskId += 1;
      this.$taskInput.value = "";
      this.render();
    }

    deleteTask(id) {
      this.tasks = this.tasks.filter((task) => task.id !== id);
      this.render();
    }
}
```

03

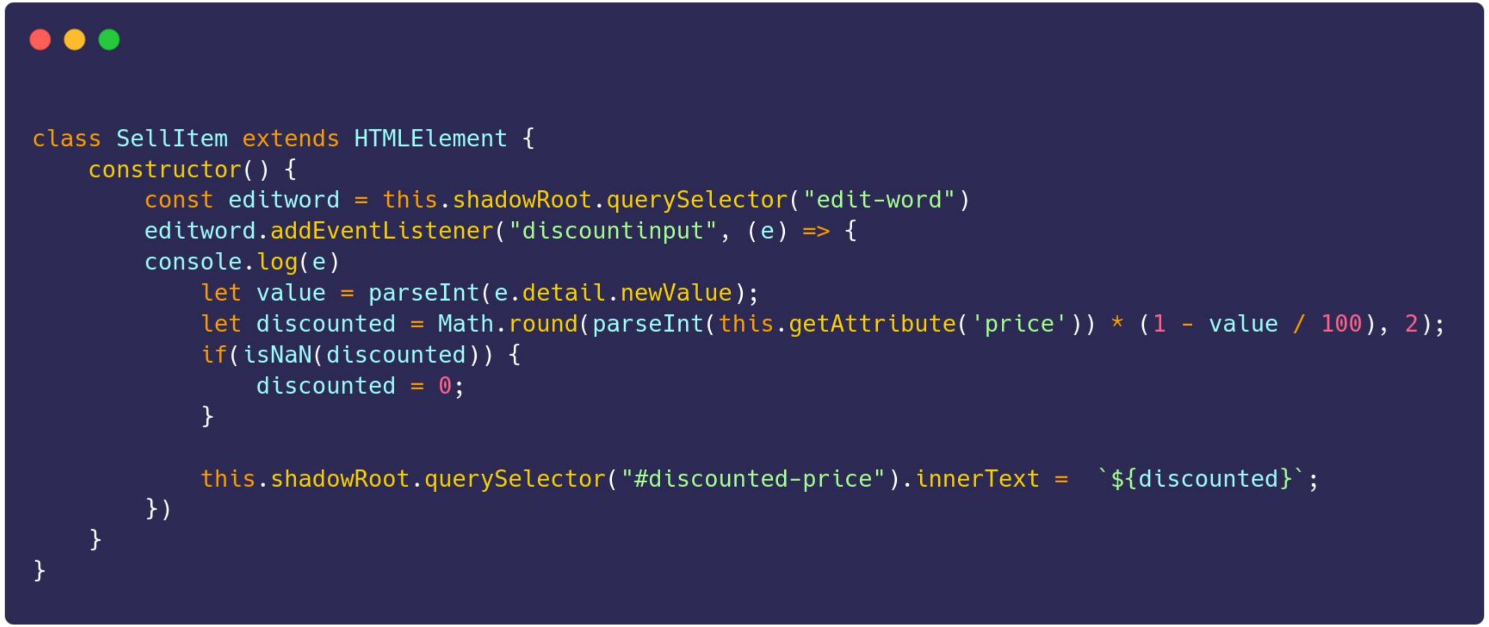
Cosas interesantes

Custom Event



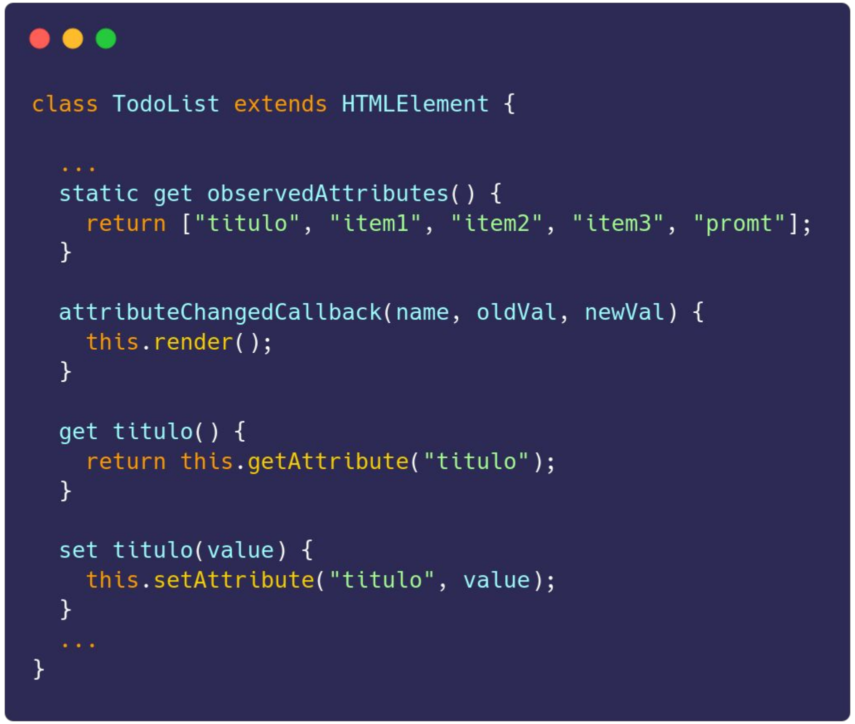
```
class EditWord extends HTMLElement {
    constructor() {
        ...
        input.oninput = function(e){
        this.dispatchEvent(new CustomEvent("discountinput", {
            composed: true, bubbles: true, detail: { newValue: input.value } }));
        }
    }
    ...
}
```

Custom Event



```
class SellItem extends HTMLElement {
    constructor() {
        const editword = this.shadowRoot.querySelector("edit-word")
        editword.addEventListener("discountinput", (e) => {
        console.log(e)
            let value = parseInt(e.detail.newValue);
            let discounted = Math.round(parseInt(this.getAttribute('price')) * (1 - value / 100), 2);
            if(isNaN(discounted)) {
                discounted = 0;
            }

            this.shadowRoot.querySelector("#discounted-price").innerText =  `${discounted}`;
        })
    }
}
```

AttributeChangedCallback



```
class TodoList extends HTMLElement {

  ...
  static get observedAttributes() {
    return ["titulo", "item1", "item2", "item3", "promt"];
  }

  attributeChangedCallback(name, oldVal, newVal) {
    this.render();
  }

  get titulo() {
    return this.getAttribute("titulo");
  }

  set titulo(value) {
    this.setAttribute("titulo", value);
  }
  ...
}
```

04

Aprendizajes

Aprendizajes

- La encapsulación de los componentes permite que sean reutilizables en cualquier contexto, incluso dentro de otros componentes.
- Forma estándar ofrece mayor granularidad que lit.
- Por otro lado, lit ofrece una manera mucho menos verbosa de realizar las mismas cosas que el método estándar.
- La generación de eventos custom ayuda mucho en el caso de componentes nesteados.