UNIVERSIDAD DE LAS FUERZAS ARMADAS-ESPE

SEDE SANTO DOMINGO DE LOS TSÁCHILAS

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN - DCCO-SS

CARRERA DE INGENIERÍA EN TECNOLOGÍAS DE LA INFORMACIÓN

PERIODO	:	Octubre 2024 – Marzo 2025
ASIGNATURA	:	Programación Integrativa de Componente
TEMA	:	Laboratorio 1 U1
ESTUDIANTE	:	González Orellana Adriana Pamela
NIVEL-PARALELO - NRC	:	6to “A” -1428
DOCENTE	:	Ing. Luis Alberto Castillo Salinas
FECHA DE ENTREGA	:	16/noviembre/2024

Laboratorio 1: Web Components

SANTO DOMINGO – ECUADOR

2024

1. Introducción

En esta práctica entenderemos los conceptos avanzados del desarrollo web que son fundamentales para entender los Web Components, Custom Elements, Shadow DOM y Plantillas de HTML. Estos conceptos posibilitan la creación de componentes de interfaz de usuario que son reutilizables, encapsulados y altamente personalizables, lo que facilita el desarrollo de aplicaciones web modulares y escalables. Mediante la implementación de Custom Elements, descubriremos cómo diseñar elementos HTML personalizados y aprovecharlos en aplicaciones web. El Shadow DOM facilitará la gestión eficaz del aislamiento y la encapsulación de estilos y scripts, previniendo conflictos con otros elementos de la página. Además, se emplearán HTML Templates para especificar estructuras de contenido reutilizables que se puedan duplicar e integrar de forma dinámica. Entonces, este laboratorio fortalecerá y pondrá en práctica los conocimientos adquiridos en las clases y actividades, resultando en la elaboración de pequeñas aplicaciones prácticas que ilustran la capacidad de estos conceptos.

Objetivos

- Utilizar Custom Elements, aprovechando Shadow DOM para resguardar estilos y estructuras, y empleando HTML Templates para la inserción dinámica de contenido.

Objetivos específicos

- Crear un componente gráfico utilizando el concepto de CustomElements.
- Establecer el uso ShadowDOM y HTMLTemplates.
- Crear pequeñas aplicaciones de los conceptos CustomElements, ShadowDOM y HTMLTemplates.

2. Desarrollo

PARTE 1: Creación de un Custom Element

Paso 1: Abrir en el navegador la página jsbin.com.

Esta página https://jsbin.com/?html,output nos provee un ambiente de desarrollo propicio para la creación de Web Components.

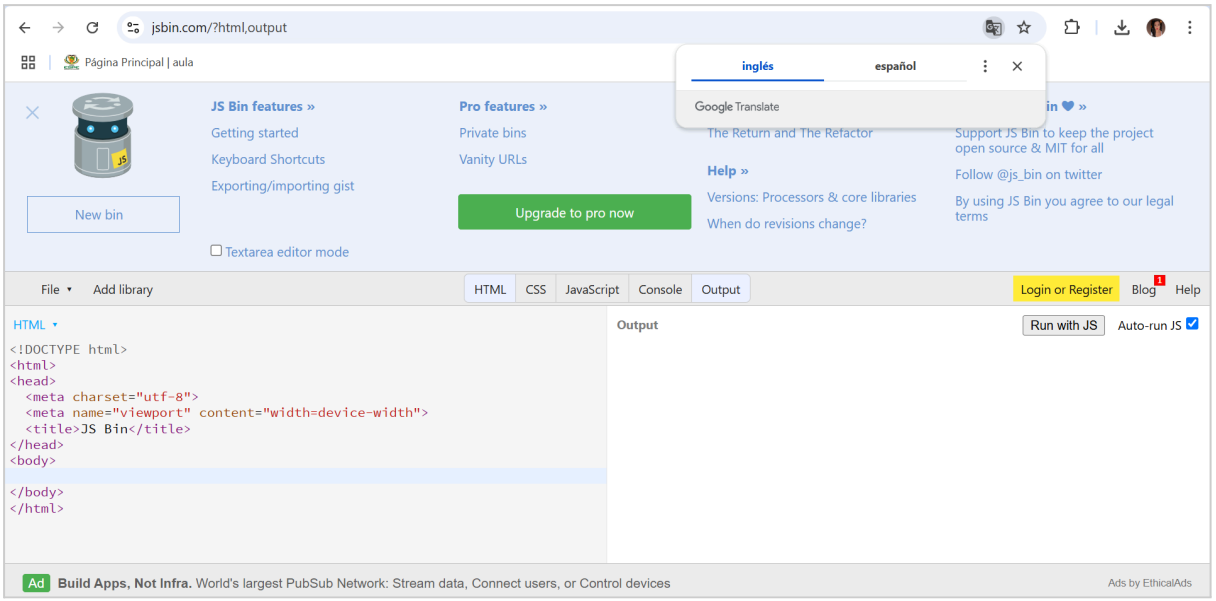


Fig 1. Pagina JSBin.com

Paso 2: Crear un Custom Element que represente un botón de compra.

Empezamos por el código JavaScript.

```
class SellButton extends HTMLElement{
  //Aqui va la funcionalidad del componente web
  //Eventos, funciones, metodos

  constructor(){
    super();
  }
  connectedCallback(){
    this.innerHTML =`
   <div>
        <button>Comprar Ahora</button>
   </div>
    `;
  }

}
window.customElements.define("sell-button",SellButton);
```

Creando un Custom Element llamado ***SellButton*** que representa un botón de compra. Este botón se puede insertar en cualquier página HTML como un elemento personalizado **<sell-button>.**

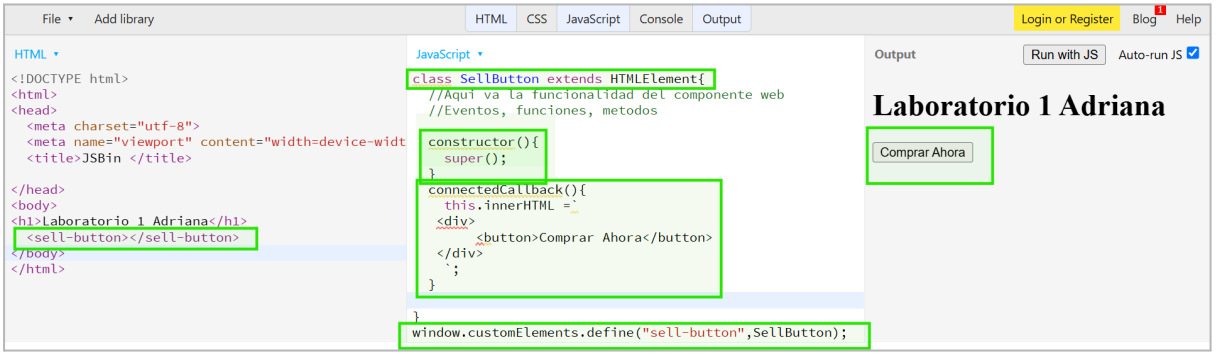


Fig 2. Custom Element botón de compra.

Paso 3: Usar los métodos del ciclo de vida que posee Custom Element, también sus librerías

Estos conceptos son revisados en clase e investigados por cada estudiante.

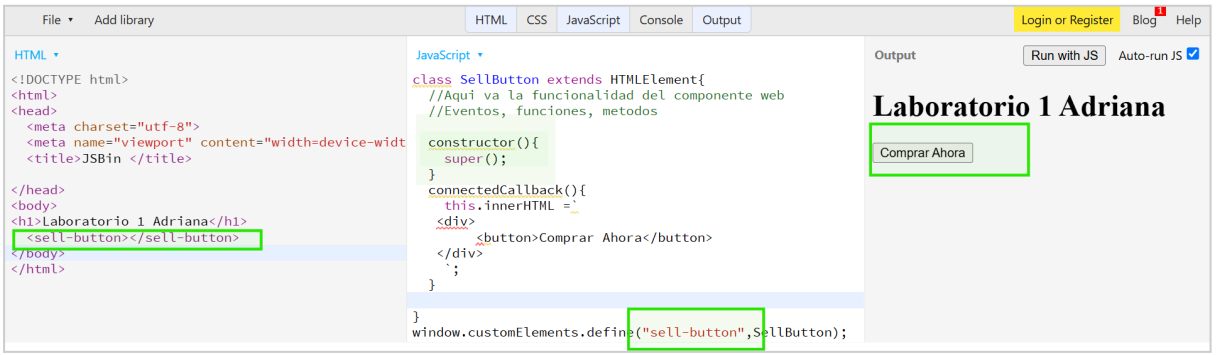


Fig 3. Librerías Custom Element botón de compra.

PARTE 2: Uso de ShadowDOM

Paso 1: Usando el mismo botón de compra pasar el código para que sea usado como Shadow DOM.

En esta funcionalidad entran los conceptos de JavaScript y DOM.

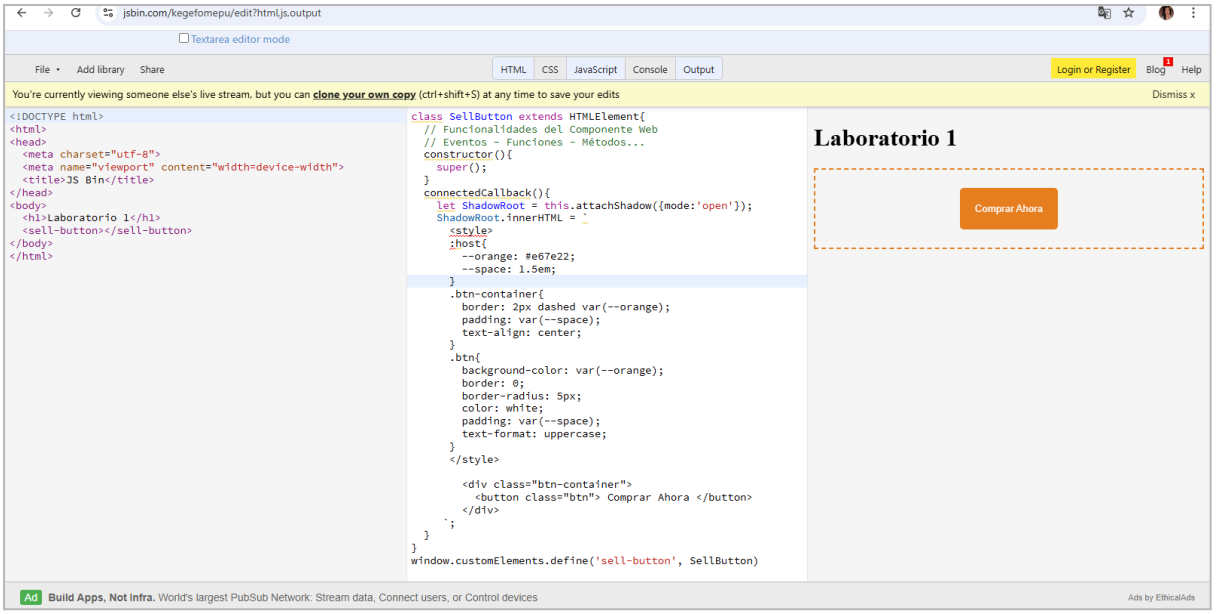


Fig 4. Uso de ShadowDOM.

Paso 2: Uso de la función innerHTML

Esta función nos permite escribir código HTML como string para poderlo enviar en el Shadow DOM.

Al utilizar innerHTML para agregar código HTML como una cadena de texto, que luego se enviará al Shadow DOM del componente. El Shadow DOM permite ***encapsular*** el HTML y CSS de un Custom Element, protegiéndolo de estilos o scripts externos y proporcionando un aislamiento completo del contenido y sus estilos.

En el HTML, se define una plantilla de contenido usando ***<template id="sellBtn">***.

Un bloque de estilos ***(<style>)*** que define variables CSS ***(--orange y --space)*** y estilos específicos para el botón.

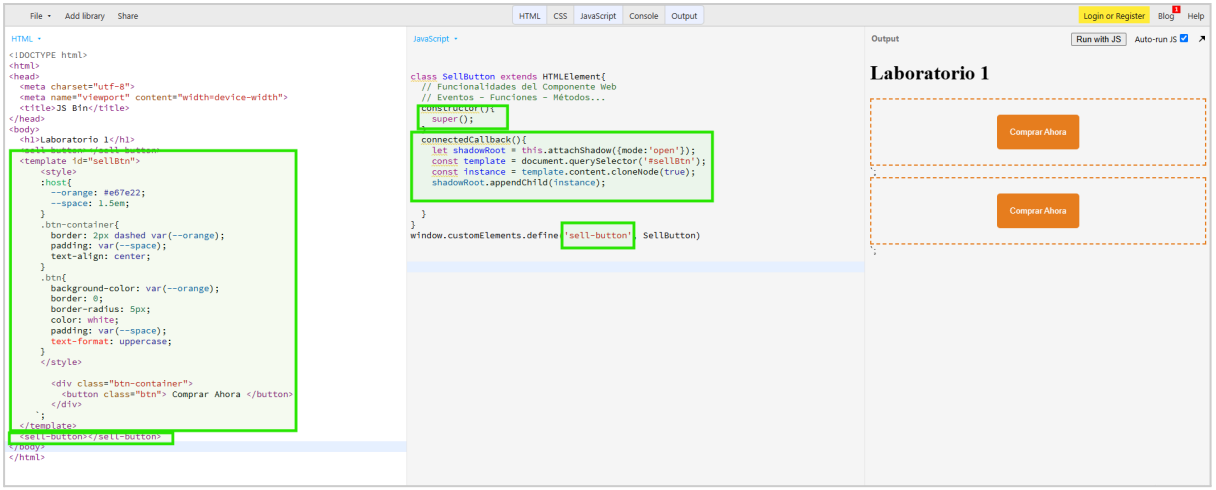


Fig 5. Uso de la función innerHTML.

PARTE 3: Uso de HTML Template

Paso 1: Uso de la etiqueta template Esto nos evitaría escribir el HTML como string y pasarlo en la función innerHTML.

```
<template id="sellBtn">
    <style>
        :host{
          --orange: #e67e22;
          --space: 1.5em;
        }
        .btn-container{
          border: 2px dashed var(--orange);
          padding: var(--space);
          text-align: center;
        }
        .btn{
          background-color: var(--orange);
          border: 0;
          border-radius: 5px;
          color: white;
          padding: var(--space);
          text-format: uppercase;
        }
        </style>

          <div class="btn-container">
            <button class="btn"> Comprar Ahora </button>
          </div>

    </template>
    <sell-button></sell-button>
    <sell-button></sell-button>
</template>
```

Paso 2: Manejo del HTML dentro de la etiqueta template.

• Así definimos todo desde el HTML dándole un id al template y manejando ese template en el archivo JavaScript con el id que le dimos.

• Para esto creamos una constante después de la variable shadowRoot, esta variable va a hacer referencia al documento HTML que estamos usando y en este identificamos el id del template

```
class SellButton extends HTMLElement{
    constructor(){

        super();
    }

    connectedCallback(){
        let shadowRoot = this.attachShadow({mode:'open'});
        const template = document.querySelector('#sellBtn');
        const instance = template.content.cloneNode(true);
        shadowRoot.appendChild(instance);


    }
}
window.customElements.define('sell-button',SellButton)
```

Se creó una constante después de la variable `shadowRoot` que hace referencia al documento HTML usando `***document.querySelector('#sellBtn')***` para identificar el id del template, luego se clona el contenido de la plantilla con `***template.content.cloneNode(true)***`, y se añade el contenido clonado al Shadow DOM del componente con `***shadowRoot.appendChild(instance)***`.

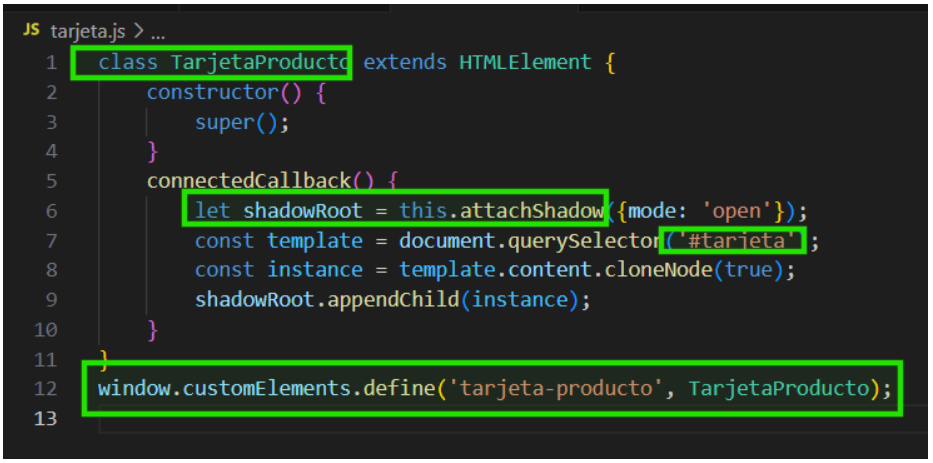
Parte 4: Manejo de los conceptos antes vistos directamente en archivos creados

Paso 1: Creación de un Custom Element

• Crear el archivo JS con la clase que contendrá nuestro custom element.

Utilizamos el método ***attachShadow()*** para crear un "shadow DOM" en un componente personalizado, es un DOM independiente y aislado que tiene su propio contexto de estilo y estructura.

El mode ***'open'*** permite que el contenido del shadow DOM sea accesible desde JavaScript, al llamar a ***this.attachShadow({mode: 'open'})***, estamos creando un espacio donde podemos agregar nuestro contenido sin que interfiera con el resto del documento.



Utilizando ***connectedCallback()***, seleccionamos el template correspondiente utilizando document.getElementById().Clonamos el contenido del template usando ***template.content.cloneNode(true).*** Esto toma todo el contenido dentro del ***<template>*** y lo clona, creando una copia de todo el HTML y los estilos.

• Uso del custom element en un archivo HTML

```html
<template id="tarjeta">
    <style>
        .tarjeta {
            width: auto;
            border: 3px solid #de7454;
            border-radius: 1px;
            margin: 5px auto;
            text-align: center;
        }
        .tarjeta img {
            width: 10%;
        }
    </style>
    <div class="tarjeta">
        <img src="1.jpg" alt="Amoxicilina Suspensión 200 ml">
        <div class="contenido">
            <h2>Amoxicilina Suspension 200 ml</h2>
            <p>Amoxicilina/ácido clavulánico se utiliza
              en adultos y niños para tratar las siguientes infecciones:
               infecciones agudas de oído, nariz y garganta, infecciones.</p>
        </div>
    </div>
</template>
<script src="tarjeta.js"></script>
<script src="sell-button.js"></script>
```

El resultado de lo que agregamos una tarjeta con una imagen y descripción ya que esta está en un template independiente al botón antes ya creado.

3. **Conclusiones**
    - Utilizando Web Components con Shadow DOM me permite una mejor encapsulación y separación de preocupaciones, es decir me permite definir el contenido y los estilos dentro de un shadowRoot, cada componente como el botón de compra o la tarjeta de producto se comporta de manera autónoma, sin que sus estilos o estructuras afecten al resto de la página.
    - El uso de templates y la clonación de contenido facilita la creación de componentes dinámicos y reutilizables. Al definir los elementos de la tarjeta y el botón dentro de un <template>, y luego clonarlos en el JavaScript mediante el shadowRoot.
4. **Recomendaciones**
    - Se deben estructurar los Web Components de manera modular desde el inicio, identificando y separando claramente los componentes reutilizables en la aplicación, en este caso se ha creado componentes como sell-button y tarjeta-producto, que son buenos ejemplos de módulos independientes.
    - Aprovechar los mecanismos de comunicación entre Web Components para mejorar la interacción y la flexibilidad, así cuando más utilizamos Web Components es crucial gestionar cómo se comunican entre sí.

Bibliografía

Web Components. (s/f). MDN Web Docs. Recuperado el 5 de noviembre de 2024, de https://developer.mozilla.org/en-US/docs/Web/API/Web_components

Álvarez, S. (2017, enero 15). *ES Modules: Qué son y cómo puedes empezar a utilizarlos*. Codecoolture. https://medium.com/codecoolture/es-modules-qu%C3%A9-son-y-c%C3%B3mo-puedes-empezar-a-utilizarlos-e88c49043593

(N.d.). Retrieved November 5, 2024, from http://chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://www.researchgate.net/profile/Federico-Gonzalez-Brizzio/publication/320233125_Web_Components_un_nuevo_estandar_para_el_desarrollo_de_aplicaciones_HTML5_Consideraciones_para_su_implementacion/links/59d63286458515db19c4efc5/Web-Components-un-nuevo-estandar-para-el-desarrollo-de-aplicaciones-HTML5-Consideraciones-para-su-implementacion.pdf

Usando shadow DOM. (n.d.). MDN Web Docs. Retrieved November 5, 2024, from https://developer.mozilla.org/es/docs/Web/API/Web_components/Using_shadow_DOM