UNIVERSIDAD DE LAS FUERZAS ARMADAS-ESPE SEDE SANTO DOMINGO

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	:	Tarea 2 U1
NOMBRES	:	González Orellana Adriana Pamela
NIVEL-PARALELO	:	6to “A”-1428
DOCENTE	:	Ing. Luis Alberto Castillo Salinas MSc.
FECHA DE ENTREGA	:	03 de diciembre del 2024

TEMA: *Custom Elements, Shadow DOM, ES Modules y HTML Templates*

SANTO DOMINGO - ECUADOR

2024

Índice de contenido

1. Introducción

Dentro del ámbito actual del desarrollo web, los web Components han surgido como un conjunto innovador de estándares de HTML diseñados para simplificar y agilizar la creación de páginas y aplicaciones web de última generación. Mediante esta innovadora tecnología, los desarrolladores pueden crear elementos HTML personalizados y se comportan de manera similar a los elementos nativos, pero con una flexibilidad y modularidad superior, eliminando así las restricciones de los elementos HTML tradicionales. Al permitir a los desarrolladores crear y compartir estos componentes sin necesidad de alcanzar un consenso a nivel mundial, los Web Components fomentan el avance del HTML y posibilitan un mayor dinamismo en el diseño y desarrollo web. En los Web Components, los HTML Template son un papel fundamental como recipientes para almacenar contenido reutilizable que no se muestra de inmediato en el DOM, volviéndose ideales para el desarrollo de interfaces dinamicas y modulares. Estas plantillas posibilitan a los desarrolladores gestionar estructuras complejas de manera más eficaz, fomentando un desarrollo mas ordenado. La manera en que se los implementa resulta ser altamente beneficiosa y funcional. La impactante influencia de los Web Components es la evolución de la creación de sistemas empresariales y aplicaciones dinamicas. Los Desafios que enfrentan son gestion de eventos y el encapsulamiento.

2. Objetivos

2.1. Objetivo General

Investigar el uso de HTML Template como parte de los Web components, destacando su funcionalidad, aplicaciones prácticas, ventajas en el desarrollo web moderno, y su impacto en la creación de interfaces dinámicas y modulares.

2.2. Objetivos Específicos

2.2.1. Descubrir las diversas formas en las se puede aplicar y utilizar los HTML Template en proyectos de desarrollo web, contemplando situaciones reales de uso como en sistemas empresariales y aplicaciones dinámicas

2.2.2 Examinar las diferencias entre los web components, que incluyen los HTML Template, frente a frameworks y liberarias como React y Angular, valorando su desempeño, características y beneficios en modularidad y posibilitan la reutilización

2.2.3 Identificar los desafíos habituales relacionados con el uso de Plantillas HTML y plantear soluciones efectivas, centrándose en cuestiones como la gestión de eventos, el encapsulamiento y las limitaciones presentes en la actualidad.

3. Desarrollo

¿Qué son los componentes web?

Los componentes web son bloques de código que encapsulan la estructura interna de elementos HTML, incluyendo CSS y JavaScript, permitiendo así que el código se pueda volver a usar como se necesite en otras web y aplicaciones.

- Custom Elements: conjuntos de API de JavaScript para definir elementos personalizados por el usuario.
- Shadow DOM: conjunto de API de Javascript para crear elementos DOM.
- ES Modules: Módulos para integrar y reutilizar documentos de Javascripts.
- HTML Template: plantillas HTML que no se muestran en la página web final y que pueden servir de base para ciertos elementos definidos por el usuario.

El Tema asignado para nuestro grupo fue:

HTML Template

Los llamados HTML Template son plantillas de archivos HTML. Los elementos que contienen se mantienen inactivos y sin renderizar sin ser representados gráficamente hasta que sean solicitados explícitamente. Esta característica hace que no perjudique el tiempo de carga de las páginas web. Por lo tanto son una buena alternativa a los métodos de JavaScript tradicionales.

ESTILO

Con la etiqueta <template> se define una plantilla HTML.

En el siguiente ejemplo podemos ver la plantilla creada llevara el nombre my elementos

```
<template id="my-element">
<p> My Element </p>
</template>
```

1. Aplicaciones prácticas y casos de uso reales

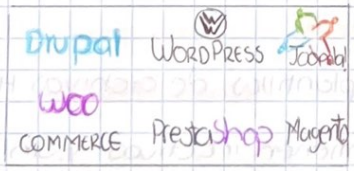
- Ejemplos de proyectos o aplicaciones donde los Web Components sean esenciales.

Bibliotecas de componentes compartidas:

Sistemas de diseño como Ionic, que utiliza Web Components para crear componentes reutilizables en aplicaciones móviles y web

Ventaja: Al ser compatibles con cualquier framework o sin ellos, los componentes pueden ser usados en diversos proyectos sin necesidad de adaptaciones.

Sistemas de gestión de contenido (CMS)



Aplicaciones como WordPress Gutenberg, que utiliza Web Components para manejar bloques personalizados de contenido.

Ventaja: Los bloques definidos como web components se pueden reutilizar y extender con facilidad, simplificando la personalización.

Plataformas de comercio electrónico

Catálogo de productos que implementan componentes personalizados como tarjetas de producto, carrusel de imágenes y modales de detalles.

Ventaja: La modularidad de los Web Components permite actualizar o reutilizar estos elementos sin afectar la estructura general de la página.

Dashboards empresariales

Herramientas de análisis de datos como Tableau, que pueden usar Web Components para renderizar gráficos interactivos, tablas dinámicas y filtros personalizados.

Ventaja: Los componentes encapsulados y fáciles de integrar en diferentes aplicaciones empresariales.

2. Comparaciones entre Web Components y frameworks/librerías como React o Angular en términos de funcionalidad y rendimiento

- Independencia

  Web Components: No requieren frameworks adicionales para funcionar, son nativos del navegador.

  React/Angular: Dependientes de su entorno, requieren una configuración inicial más compleja.

ESTILO

- Reutilización

  Web Components: Se pueden usar en cualquier proyecto, independientemente de la tecnología.

  React / Angular: Los componentes están acoplados al framework y requieren adaptaciones para su uso fuera del él.

- Encapsulamiento y estilos

  Web Components: Proporcionan encapsulamiento completo mediante Shadow DOM, evitando conflictos de estilos globales.

  React / Angular: El encapsulamiento es parcial, depende del CSS-in-JS (React)

- Rendimiento

  Web components: Ofrecen un rendimiento inicial superior ya que no necesitan empaquetadores ni compiladores.

  React / Angular: Más pesados en la carga inicial debido al tamaño del runtime y las dependencias, pero con optimización en el rendimiento de aplicaciones globales grandes mediante reconciliación React o cambio de detección (Angular).

- Curva de aprendizaje

  Web components: Más sencilla para desarrolladores que ya conocen HTML, CSS y JavaScript.

  React / Angular: Requieren aprender su sintaxis, herramientas y ecosistema.

3. Uso de Web Components en aplicaciones empresariales o sistemas de diseño (Design Systems).

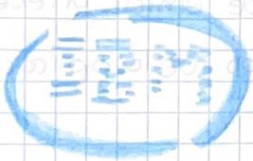
- Coherencia en el Diseño

Los web components permiten definir y entender y mantenendo un lenguaje visual consistente en grandes organizaciones.

Salesforce Lightning Design System utiliza componentes estándar para garantizar que todas las aplicaciones sigan las mismas pautas de diseño.

- Integración sin Restricciones.

Al ser independientes de frameworks, los web components pueden integrarse fácilmente en aplicaciones desarrolladas con tecnologías variadas.



Grandes empresas como IBM han adoptado web components para su sistema de diseño, permitiendo que sus equipos trabajen con diferentes stacks tecnológicos.

- Escalabilidad

En proyectos grandes, los web components reducen la duplicación de código y promueven la reusabilidad, facilitando el mantenimiento y las actualizaciones.

Plataformas como Shoelace, que proporciona una biblioteca de componentes de UI basados en web components utiles para aplicaciones empresariales.

ESTILO

Problemas comunes y Soluciones

1. Manejo de eventos entre Web Components y otros elementos de la aplicación.

- Problema: La comunicación entre web components y otros elementos de la aplicación puede ser complicada especialmente cuando se requiere pasar datos o manejar eventos entre un componente encapsulado y su entorno. Esto ocurre porque los web components encapsulan su funcionalidad lo que puede limitar la propagación de eventos.
- Soluciones: Los web components permiten definir y disparar eventos personalizados mediante la API CustomEvent, lo que facilita la comunicación.

Un componente <custom-button> puede emitir un evento buttonClicked que otros elementos de la aplicación pueden escuchar.

Java script

```
CustomEvent('buttonClicked', {detail: {id: 1}}
```

Configurar los eventos personalizados para que burbujeen (bubbles: true) permite que estos almacenen y alcancen elementos del DOM externo. Esto asegura que los padres de la jerarquía puedan escuchar los eventos emitidos por los web components.

2. Problemas de Encapsulamiento

- Problema: El encapsulamiento del Shadow DOM puede causar conflictos al interactuar con estilos globales o librerías de terceros, ya que los estilos definidos en el Shadow DOM no afectan el DOM empresarial global, y viceversa. Esto puede dificultar la integración con frameworks y librerías que dependen de estilos globales o selectores específicos.

- Soluciones:

Uso de variables CSS:

Las variables CSS definidas en el DOM global pueden ser utilizadas dentro del Shadow DOM para asegurar coherencia de diseño.

Ejemplo:

CSS
```
:root {
  --primary-color: blue;
}
```

Shadow DOM CSS
```
.button {
  color: var(--primary-color);
}
```

Estrategias híbridas:

Si se requiere interacción entre estilos globales y encapsulados, considerar no usar Shadow DOM para ciertas componentes o integrar un modo "open" para acceder al contenido encapsulado.

ESTILO

3. Limitaciones actuales de los Web Components y como superarlas.

- Soporte en navegadores antiguos:

  Problema: Algunos navegadores más antiguos no soportan completamente los estándares de Web Components, como Shadow DOM o Custom Element.

  Solución: Utilizar polyfills como webcomponents.js para garantizar compatibilidad retroactiva en navegadores que no soportan estas características.

  Problema: Para los Desarrolladores acostumbrados a frameworks como React o Angular, trabajar con Web components puede parecer menos intuitivo debido a la falta de herramientas como JSX o sistemas de estado integrados.

  Solución: Complementar el Desarrollo de Web Components como librerías como Lit o stencil, que simplifican la creación de componentes y ofrecen sintaxis amigable.

- Limitaciones en herramientas de desarrollo:

  Problema: Algunas herramientas de depuración no ofrecen el mismo nivel de soporte para web components que para frameworks populares.

  Solución: Utilizar extenciones y herramientas diseñadas específicamente para Web Components, como el inspector del Shadow Dom en navegadores modernos.

4. Conclusiones

- Los Web Components son una valiosa herramienta que posibilita la creación de componentes reutilizables y autocontenidos, los wcles pueden ser compatibles con cualquier framework o aplicación, facilitando su incorporación en proyecto contemporáneos.
- La tecnología de Web Components aborda de manera mas efectiva problemas comos estilos y falta de modularidad presentes en el desarrollo tradicional, al proporcionar una solución integrada a través del Shadow DOM y los Custom Elements.
- A pesar de encontrarse con algunas limitaciones, como la falta de soporte en navegadores antiguos y una curva de aprendizaje al principio, estos obstáculos pueden ser superados gracias al uso de polyfills con libreria Lit, y un mayor respaldo.

5. Recomendaciones

- Implementar plantillas HTML para definir componentes como tarjetas de productos, menús o modales, permitiendo su reutilización en diferentes partes de la aplicación y reduciendo el código repetido.
- Integrar plantillas comr Javascript para cargar contenido dinámico, como datos de tablas o galerías de imgenes, mejorando la eficiencia de la aplicacion que maneja grandes volumenes de información.
- Utilizar HTML Template como base para bibliotecas de componentes empresariales, asegurando consistencia en el diseño y facilitando la personalización por parte de equipos distribuidos o usuarios finales.

6. **Bibliografía**

[1] *Solución de problemas comunes de HTML*. (n.d.). MDN Web Docs. Retrieved November 30, 2024, from https://developer.mozilla.org/es/docs/Learn/HTML/Howto.

[2] Rodas, G. (n.d.). *¿Cómo funcionan los HTML Templates?* Undefined.sh. Retrieved November 30, 2024, from https://undefined.sh/posts/crea-html-templates-con-lit-html/.

[3] Belandria, D. (2021, November 22). ¿Usar o no usar Plantillas web? ventajas y desventajas de usar plantillas o templates para tu sitio web. *Imagina Colombia*. https://www.imaginacolombia.com/articulos/usar-o-no-usar-plantillas-web-ventajas-y-desventajas-de-usar-plantillas-o-templates-para-tu-sitio-web.

[4] Aguirre, G. M. P. (2020, September 3). *¿Cuáles son las ventajas y desventajas de utilizar plantillas para páginas web?* GoDaddy Resources - LATAM; GoDaddy. https://www.godaddy.com/resources/latam/stories/ventajas-y-desventajas-plantillas-para-paginas-web.

[5]Paz, G. (2022, May 12). *Opiniones sobre Web Components*. Medium. https://pazguille.medium.com/opiniones-sobre-web-components-69cf1394a85d.