DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

PROGRAMACION INTEGRATIVA

Nombre

Jean Michael Saltos Palacios.

NRC

16496

Fecha

20/06/2024

Tarea 1.3

1. Introducción

Este informe describe el desarrollo de una aplicación web utilizando Web Components, aplicando los fundamentos y principios de la programación web basada en componentes. La aplicación implementará un sistema de gestión de tareas, permitiendo agregar, visualizar y eliminar tareas.

2. Objetivos

- Implementar una aplicación web utilizando Web Components.
- Aplicar los conceptos de ciclo de vida, estándares W3C, POO, reutilización y personalización de componentes.
- Desarrollar un modelo de navegación basado en requerimientos del mundo real.

3. Modelo de Navegación

El modelo de navegación consiste en una página principal que contiene:

- Un formulario para agregar tareas.
- Una lista de tareas donde se pueden visualizar y eliminar tareas.

4. Estructura del Proyecto

```
css
task-manager-app/
├── index.html
├── css/
│   └── styles.css
└── js/
    ├── app.js
    ├── task-form.js
    └── task-list.js
```

5. Implementación del Proyecto

HTML

El archivo `index.html` define la estructura básica de la página web.

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Task Manager</title>
    <link rel="stylesheet" href="css/styles.css">
</head>
<body>
    <div class="container">
        <h1>Task Manager</h1>
        <task-form></task-form>
        <task-list></task-list>
    </div>
    <script src="js/app.js"></script>
</body>
</html>
```

CSS

El archivo `css/styles.css` mejora la apariencia de la aplicación.

```css
body {
    font-family: 'Arial', sans-serif;
    background-color: #f4f4f4;
    margin: 0;
    padding: 0;
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
}

.container {
    background-color: #fff;
    padding: 20px;
    border-radius: 8px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
    width: 80%;
    max-width: 600px;
}
```

JavaScript (Web Components)

`app.js`

```
import './task-form.js';
import './task-list.js';

document.addEventListener('DOMContentLoaded', () => {
    console.log('Task Manager App Initialized');
});
```

`task`-form.js

```
addTask(event) {
    event.preventDefault();
    const taskTitle = this.shadowRoot.querySelector('#taskTitle').value;
    if (taskTitle) {
        const taskAddedEvent = new CustomEvent('task-added', {
            detail: { title: taskTitle },
            bubbles: true,
            composed: true
        });
        this.dispatchEvent(taskAddedEvent);
        this.form.reset();
    }
}
```

task-list.js

```
addTaskToList(event) {
    const task = event.detail;
    const li = document.createElement('li');
    li.textContent = task.title;

    const deleteButton = document.createElement('button');
    deleteButton.textContent = 'Delete';
    deleteButton.addEventListener('click', () => {
        li.remove();
    });

    li.appendChild(deleteButton);
    this.taskList.appendChild(li);
}
```

6. Ciclo de Vida de los Componentes

Los Web Components tienen un ciclo de vida definido por varios métodos:

1. **`connectedCallback`**: Se llama cuando el componente es agregado al DOM.
2. **`disconnectedCallback`**: Se llama cuando el componente es eliminado del DOM.
3. **`attributeChangedCallback`**: Se llama cuando un atributo del componente cambia.
4. **`adoptedCallback`**: Se llama cuando el componente es movido a un nuevo documento.

En este proyecto, los componentes `TaskForm` y `TaskList` utilizan el método `connectedCallback` para inicializar y gestionar sus respectivos eventos y estados.

7. Conclusión

Este proyecto demuestra cómo utilizar Web Components para crear una aplicación web modular y reutilizable. Implementando el ciclo de vida de los componentes, siguiendo los estándares W3C y aplicando principios de POO, se logra una arquitectura de componentes eficiente y fácil de mantener.