

PATRONES DE DISEÑO

DAW 2

Manuel Rubio Torrentí

17/11/25

| | |
|-----------------------------------|----------|
| E01 ConfigManager | 2 |
| E02 Observer con singleton | 3 |

E01 ConfigManager

```
*/
export class ConfigManager {
  // Instancia única
  static #instance = null;

  constructor() {
    // COMPLETA CÓDIGO AQUÍ
    if (ConfigManager.#instance) {
      return ConfigManager.#instance;
    }

    // Estado inicial de la configuración global
    this.config = {
      theme: "light",
      language: "es",
      version: "1.0.0",
    };
  };

  // COMPLETA CÓDIGO AQUÍ
  ConfigManager.#instance = this;
}

// devuelve sólo uno de los parámetros
getParam(key) {
  return this.config[key];
}

// añade o cambia un parámetro de configuración y su valor
setParam(key, value) {
  return this.config[key] = value;
}

getConfig() {
  return this.config;
}
}
```

```

manrubtor@client25:~/Escritorio/DAW/Desarrollo_web_en_entorno_cliente/T4/Singleton/ejemplos_ejercicios$ node e01_configManager/index.js
¿Es la misma instancia? true
Configuración inicial: { theme: 'light', language: 'es', version: '1.0.0' }

Idioma desde c2: en

Estado global final: { theme: 'light', language: 'en', version: '1.0.0' }
manrubtor@client25:~/Escritorio/DAW/Desarrollo_web_en_entorno_cliente/T4/Singleton/ejemplos_ejercicios$

```

E02 Observer con singleton

```

export class ThemeStore {
  static #instance = null;
  constructor() {
    //implementar
    if (ThemeStore.#instance) {
      return ThemeStore.#instance;
    }
    this.state = {
      theme: 'light',
    };
    this.events = {
      themeChange: [],
    };
    ThemeStore.#instance = this;
  }

  subscribe(eventName, newSubscriber) {
    //implementar
    if (!this.events[eventName]) {
      this.events[eventName] = [];
    }
    this.events[eventName].push(newSubscriber);
  }

  notify(eventName = 'themeChange', data = this.state) {
    // implementar
    if (!this.events[eventName]) return;
    this.events[eventName].forEach(sub => sub.update(data));
  }

  toggleTheme() {
    this.state.theme = this.state.theme === 'light' ? 'dark' : 'light';
    this.notify('themeChange', this.state);
  }
}

```

```

export class StoreSub {
  // recibe un nodo del DOM
  constructor(nodo) {
    this.nodo = nodo;
  }

  // recibe el estado state, que contiene { theme: 'light' } o {theme: 'dark'}
  update(state) {
    //implementar
    const { theme } = state;
    this.nodo.classList.remove('light', 'dark');
    this.nodo.classList.add(theme);
  }
}

```

```

e02_oberver_con_singleton > js app.js > ...
1  import { ThemeStore , StoreSub } from './Store.js';
2
3  // Aquí inicializamos directamente el singleton
4  const store = new ThemeStore();
5  const body = document.body;
6  const header = document.getElementById('mainHeader');
7  const toggleBtn = document.getElementById('toggleThemeBtn');
8  const container = document.querySelector('.container');
9  const boxes = document.querySelectorAll('.box');
10 // Creamos los subscribers para cada elemento
11 const bodySub = new StoreSub(body);
12 const headerSub = new StoreSub(header);
13 const buttonSub = new StoreSub(toggleBtn);
14 const containerSub = new StoreSub(container);
15 const boxesSubs = Array.from(boxes).map(box => new StoreSub(box));
16 // Suscribimos al store
17 store.subscribe('themeChange', bodySub);
18 store.subscribe('themeChange', headerSub);
19 store.subscribe('themeChange', buttonSub);
20 store.subscribe('themeChange', containerSub);
21 boxesSubs.forEach(sub => store.subscribe('themeChange', sub));
22 // Evento del botón
23 toggleBtn.addEventListener('click', () => store.toggleTheme());
24 // Inicializamos la vista la primera vez con el tema por defecto
25 store.notify('themeChange');
26

```

