

Prompt para Desarrollo del Proyecto MotoGP Desktop

Contexto del Proyecto

Estoy desarrollando un proyecto web llamado **MotoGP Desktop** para la asignatura "Software y estándares para la Web" del Grado en Ingeniería Informática del Software.

Requisitos Obligatorios de Refactorización

1. TAREA 1: Evitar Métodos Obsoletos (Deprecated)

NUNCA usar:

- `document.write()`
- `document.writeln()`

SIEMPRE usar en su lugar:

```
javascript
```

```
// Crear elementos  
const elemento = document.createElement("tipoElemento");  
elemento.textContent = "contenido";  
  
// Insertar en el DOM  
document.body.insertBefore(elemento, referencia);  
// o  
elementoPadre.appendChild(elemento);
```

Referencias:

- <https://developer.mozilla.org/es/docs/Web/API/Document/write>
 - <https://developer.mozilla.org/es/docs/Web/API/Document/writeln>
-

2. TAREA 2: Separación de Contenido, Presentación y Computación

PRINCIPIOS FUNDAMENTALES:

- El código JavaScript debe estar en archivos `.js` separados
- El HTML debe contener SOLO estructura y contenido
- El CSS debe contener SOLO estilos y presentación
- La lógica debe estar completamente en JavaScript

PROHIBIDO en HTML:

- Atributos de eventos: `onClick`, `onLoad`, `onChange`, `onSubmit`, `onMouseOver`, etc.
- Código JavaScript inline en atributos
- Uso de `[javascript:]` en enlaces

OBLIGATORIO en JavaScript:

```
javascript

// Seleccionar elementos usando selectores CSS (NO usar IDs si es posible)
const elemento = document.querySelector("selector-css-valido");
const elementos = document.querySelectorAll("selector-css-valido");

// Registrar eventos con addEventListener
elemento.addEventListener("evento", manejador);

// Procesar eventos con el objeto Event
function manejador(event) {
    event.preventDefault(); // si es necesario
    // lógica del evento
}
```

IMPORTANTE:

- No usar `#id` o `.class` como selectores si se puede evitar
- Preferir selectores CSS estructurales: `body > footer`, `nav button`, `form input[type="file"]`
- Usar `DOMContentLoaded` para asegurar que el DOM está cargado

Referencias:

- <https://developer.mozilla.org/en-US/docs/Web/API/Document/querySelector>
 - <https://developer.mozilla.org/es/docs/Web/API/EventTarget/addEventListener>
 - <https://developer.mozilla.org/es/docs/Web/API/Event>
 - <https://www.w3.org/Style/CSS/current-work>
-

3. TAREA 3: Uso Estricto del Paradigma de Orientación a Objetos

MODELO OBLIGATORIO:

- Usar **modelo basado en clases** (ES6+)
- No usar el modelo basado en prototipos/delegación

ESTRUCTURA DE CLASES:

```
javascript
```

```
class NombreClase {  
    // Atributos privados (usar # obligatoriamente)  
    #atributoPrivado;  
    #otroAtributoPrivado;  
  
    constructor(parametros) {  
        this.#atributoPrivado = parametros;  
    }  
  
    // Métodos privados (usar # obligatoriamente)  
    #metodoPrivado() {  
        // Lógica interna  
    }  
  
    // Métodos públicos (sin #)  
    metodoPublico() {  
        // Puede usar métodos y atributos privados  
        this.#metodoPrivado();  
        return this.#atributoPrivado;  
    }  
  
    // Getters y setters si son necesarios  
    obtenerDato() {  
        return this.#atributoPrivado;  
    }  
}
```

```
actualizarDato(nuevo) {  
    this.#atributoPrivado = nuevo;  
}  
}
```

ENCAPSULACIÓN:

- Todo atributo interno debe ser privado (#)
- Todo método auxiliar debe ser privado (#)
- Solo exponer métodos públicos necesarios para la interfaz
- Proteger el estado interno de accesos no autorizados

VENTAJAS QUE SE BUSCAN:

- Modularidad
- Reutilización de código
- Claridad estructural
- Mejor mantenimiento
- Mayor escalabilidad
- Código más legible

Referencias:

- <https://web.dptoinformatica.uniovi.es/cueva/JavaScript/115-Saludo-OO.html>
 - <https://web.dptoinformatica.uniovi.es/cueva/JavaScript/120-atributos-privados.html>
-

Plantilla Base para Nuevos Archivos

HTML (estructura mínima):

```
html
```

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Título - MotoGP Desktop</title>
    <link rel="stylesheet" href="estilos.css">
</head>
<body>
    <!-- Contenido HTML estructural únicamente -->
    <!-- SIN atributos de eventos -->

    <footer>
        <a href="http://validator.w3.org/check/referer">
            
        </a>
        <a href="http://jigsaw.w3.org/css-validator/check/referer">
            
        </a>
    </footer>

    <!-- Script al final del body -->
    <script src="script.js"></script>
```

```
</body>  
</html>
```

JavaScript (estructura base):

```
javascript
```

```
/**  
 * Archivo: nombre.js  
 * Proyecto: MotoGP Desktop  
 * Descripción: [Descripción del propósito]  
 */  
  
class NombreClase {  
    #atributosPrivados;  
  
    constructor(parametros) {  
        // Inicialización  
    }  
  
    #metodosPrivados() {  
        // Lógica interna  
    }  
  
    metodosPublicos() {  
        // API pública  
    }  
}  
  
// Función de inicialización  
function inicializar() {  
    // Crear instancias
```

```
// Registrar eventos  
}  
  
// Esperar a que el DOM esté listo  
document.addEventListener("DOMContentLoaded", inicializar);
```

Checklist de Validación

Antes de considerar completo cualquier código, verificar:

- No usa `document.write()` ni `document.writeln()`
- No tiene atributos de eventos en HTML (`onClick`, etc.)
- Usa `addEventListener()` para todos los eventos
- Usa selectores CSS para seleccionar elementos
- Todo el código está en archivos `.js` separados
- Usa clases de ES6+ (no prototipos)
- Todos los atributos internos son privados (#)
- Todos los métodos auxiliares son privados (#)
- Usa `DOMContentLoaded` si manipula el DOM
- El HTML contiene solo estructura
- El CSS contiene solo estilos
- El JavaScript contiene toda la lógica

Instrucciones de Uso de este Prompt

Cuando necesites ayuda con el proyecto MotoGP Desktop:

- 1. Copia este prompt completo**
- 2. Pégalo al inicio de tu conversación con Claude**
- 3. Añade tu petición específica**, por ejemplo:
 - "Necesito crear una clase para gestionar pilotos"
 - "Ayúdame a refactorizar este código HTML que tiene onClick"
 - "Crea un formulario para subir archivos siguiendo las guías"

Claude aplicará automáticamente todas estas reglas a tu código.

Notas Importantes

 **ESTOS CRITERIOS SE DEBEN MANTENER EN TODO EL PROYECTO**

- La refactorización es obligatoria y permanente
- Cualquier código nuevo debe seguir estas guías desde el inicio
- No se aceptan excepciones a estas reglas

- El proyecto completo debe ser consistente en el uso de estas prácticas
-

Recursos Adicionales

- Selector CSS: <https://www.w3.org/Style/CSS/current-work>
- MDN Web Docs: <https://developer.mozilla.org/es/>
- Ejemplos de referencia: <https://web.dptoinformatica.uniovi.es/cueva/JavaScript/>