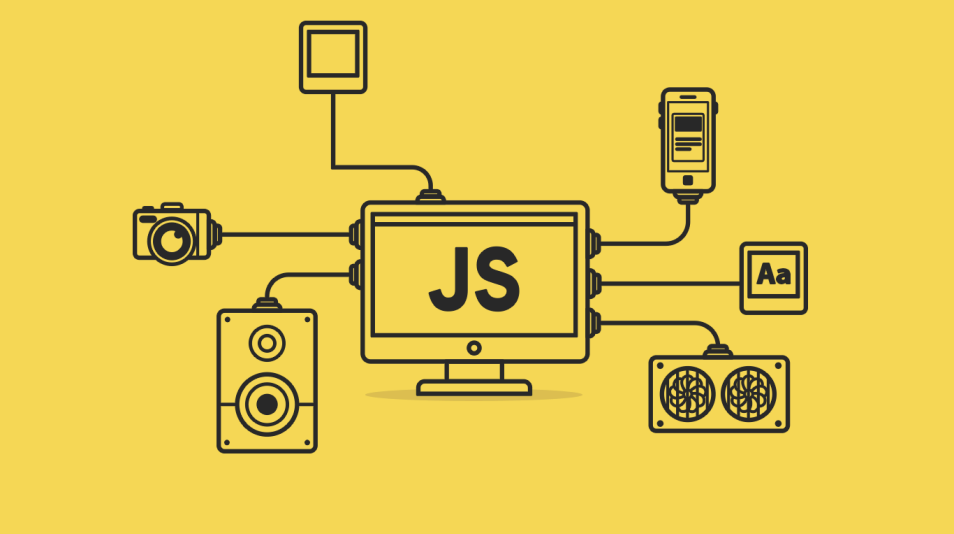
Guías de estilo JavaScript

Joel Acoran Cruz Morales - Curso 2023/2024



Proyecto UFC

Documento para las guías de estilos de JavaScript para el proyecto final de 2do curso del Ciclo formativo de grado superior en Diseño de Aplicaciones Web de Joel Acoran Cruz Morales

Proyecto UFC 1
1. Nombres significativos 3
2. Indentación 4
3. Espacios en blanco 4
4. Líneas en blanco 5
5. Comentarios 6
6. Formato de código 6
7. Uso de puntos y comas 7
8. Bloques de códigos 7
9. Manejo de errores 8

1. Nombres significativos

- Usa nombres descriptivos y significativos para variables, funciones y métodos.
- Usa camelCase para nombrar variables y funciones:

```
// Filtros por tecnica
Codeium: Refactor | Explain | ×
function filtrarPorTecnica() {
    var selectElement = document.getElementById("select-Tecnica");
    var selectedTecnicaId = selectElement.value;
    var resultadosDiv = document.getElementById("resultados-Participante");

    document.getElementById("select-Categoria").value = "";

    if (selectedTecnicaId === "") {
        MostrarLuchadoresTabla(participantes);
        resultadosDiv.style.display = "block";
    } else {
        if (participantes && participantes.data) {
            var participantesFiltrados = participantes.data.filter(function(participante) {
                return participante.tecnica && participante.tecnica.ID_Tecnica.toString() === selectedTecnicaId;
            });

            if (participantesFiltrados.length > 0) {
                MostrarLuchadoresTabla({ "data": participantesFiltrados });
                resultadosDiv.style.display = "block";
            } else {      You, last month • first commit
                resultadosDiv.style.display = "none";
            }
        } else {
            console.error("Error: no se han cargado los datos de los participantes correctamente.");
        }
    }
}
```

Ejemplo de nombres descriptivos y significativos y uso de camelCase.

2. Indentación

- Utiliza espacios en lugar de tabulaciones para la indentación.
- Usa un nivel de indentación de 2 o 4 espacios para cada nivel de anidamiento.

```
if (xhr.status === 200 || xhr.status === 204) {
    console.log("La pelea con ID " + ID_Pelea + " ha sido eliminada exitosamente.");
    window.location.href = "/peleas";
} else {
    console.error("Error al eliminar la pelea. Código de estado:", xhr.status);
}
```

Ejemplo de indentación del código haciendo uso de 4 espacios para niveles de anidamiento.

3. Espacios en blanco

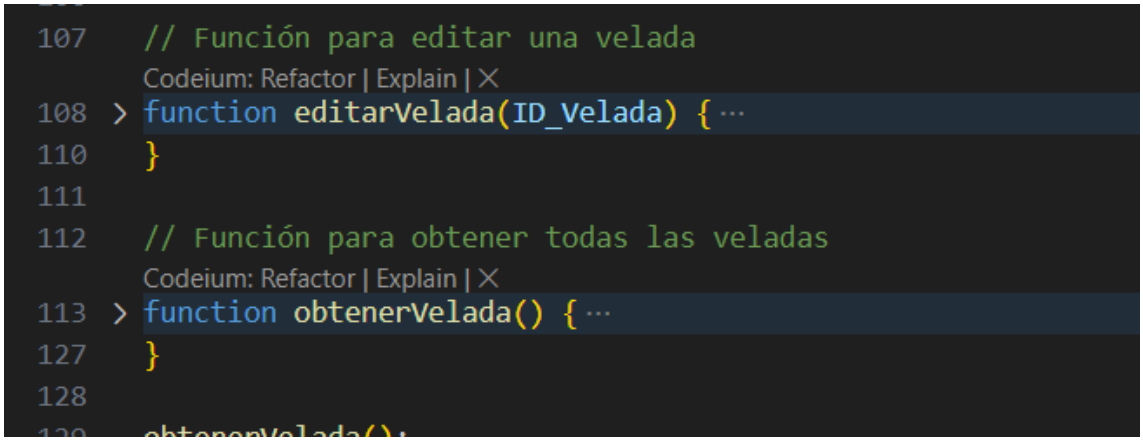
- Deja un espacio antes y después de los operadores.
- Deja un espacio después de las comas en listas de argumentos y matrices,.
- No deja espacios en blanco al final de las líneas

```
// Función para eliminar una velada
Codeium: Refactor | Explain | ×
function eliminarVelada(ID_Velada) {
    var xhr = new XMLHttpRequest();
    xhr.open("DELETE", "http://127.0.0.1:8000/api/joel/Velada/" + ID_Velada, true);
    Codeium: Refactor | Explain | Generate JSDoc | ×
    xhr.onreadystatechange = function() {
        if (xhr.readyState === XMLHttpRequest.DONE) {
```

Ejemplo de no dejar espacios en blanco, dejar espacios después de cada coma en listas de argumentos y no dejar espacios en blanco al final de las líneas.

4. Líneas en blanco

- Deja líneas en blanco para separar bloques de código relacionados, como funciones y clases.
- Deja líneas en blanco al principio y al final de los bloques de código.



Ejemplo de dejar líneas en blanco antes y después de cada función

5. Comentarios

- Usa comentarios para explicar partes del código que puedan ser confusas o necesiten aclaración.
- Utiliza comentarios de una sola línea (//) para comentarios breves y comentarios de varias líneas (/* */) para explicaciones más largas.

```
34
35  // Función para mostrar las peleas de una velada en el elemento PeleasVeladas
    Codeium: Refactor | Explain | ×
36  function MostrarPeleasVelada(peleas) {
37      // Verificar si hay peleas
38      if (peleas.length === 0) {
39          var VeladaVacia = "<br><span style='color: red;'>No hay Pelea en esta velada</span>";
40          document.getElementById("Peleas-Veladas").innerHTML = VeladaVacia;
41          return;
42      }
43
```

Ejemplo de utilización de comentarios con "//" en la misma línea aclarando la función.

6. Formato de código

- Limita la longitud de las líneas de código a 80-100 caracteres.
- Si una línea excede esta longitud, divídela en varias líneas de manera lógica y coherente.
- Utiliza comillas simples o dobles de manera consistente para cadenas

```
if (datos.hasOwnProperty("data")) {
    var participantes = datos.data;
    var tablaHTML = "<table border='1'><tr><th>Nombre</th><th>Apellido</th><th>Rol</th><th>País</th></tr>";
    participantes.forEach(function(participante) {
        tablaHTML += "<tr>";
```

Ejemplo de utilización de comillas dobles de manera consciente para cadena de texto y con líneas de menor longitud.

7. Uso de puntos y comas

- Termina cada declaración con un punto y coma (;).

```
participantes.forEach(function(participante) {
    tablaHTML += "<tr>";
    tablaHTML += "<td>" + participante.Nombre_Par + "</td>";
    tablaHTML += "<td>" + participante.Apellido_Par + "</td>";
    tablaHTML += "<td>" + (participante.rol ? participante.rol.Nombre_Rol : '') + "</td>";
    tablaHTML += "<td>" + (participante.tecnica ? participante.tecnica.Nombre_Tecnica : '') + "</td>";
    tablaHTML += "<td>" + participante.Altura_Par + "</td>";
    tablaHTML += "<td>" + participante.Peso_Par + "</td>";
    tablaHTML += "<td>" + (participante.pais ? participante.pais.Nombre_Pais : '') + "</td>";
    tablaHTML += "<td>" + (participante.categoria ? participante.categoria.Nombre_Cat : '') + "</td>";
    tablaHTML += "</tr>";
});
tablaHTML += "</table>";
document.getElementById("resultados-Participante").innerHTML = tablaHTML;
```

Ejemplos de utilización de (;) al final de cada declaración.

8. Bloques de códigos

- Usa llaves {} incluso para bloques de una sola línea.
- Abre las llaves en la misma línea que la declaración de control de flujo (if, else, for, while, etc.).

```
if (selectedCategoryId === "todos") {
    MostrarLuchadoresTabla(participantes);
    resultadosDiv.style.display = "block";
} else if (selectedCategoryId !== "") {
    if (participantes && participantes.data) {
        var participantesFiltrados = participantes.data.filter(function(participante) {
            return participante.ID_Categoria.toString() === selectedCategoryId &&
                   participante.rol.ID_Rol !== 2 && participante.rol.ID_Rol !== 3;
        });
```

*Ejemplos de utilización de **if, else if***

9. Manejo de errores

- Utiliza manejo de errores adecuado y consistente para garantizar que cualquier excepción o error sea capturado y manejado de manera apropiada.
- Utiliza try...catch para manejar errores de manera segura.

```
// Función para realizar la autenticación
Codeium: Refactor | Explain | ×
function login() {
    const username = $('#username').val();
    const password = $('#password').val();
    $.ajax({
        url: 'http://127.0.0.1:8000/api/joel/Usuario',
        type: 'GET',
        Codeium: Refactor | Explain | Generate JSDoc | ×
        success: function(data) {
            if (data && data.data.length > 0) {
                const user = data.data[0];
                if (user.Nombre_Usu === username && user.Password_Usu === password) {
                    generate_token(user.ID_Usuario);
                } else {
                    alert('Usuario o contraseña incorrecta');
                }
            } else {
                alert('Usuario no encontrado');
            }
        },
        Codeium: Refactor | Explain | Generate JSDoc | ×
        error: function(xhr, status, error) {
            console.error('Error al obtener los datos del usuario. Estado:', status, 'Error:', error);
            alert('Error al realizar la autenticación');
        }
    });
}
```

Uso de manejo de errores para cada uso.