Taller Técnico: Métodos de Arreglos y Programación Declarativa

1. Introducción

Este taller tiene como objetivo profundizar en el uso de métodos de arreglos en JavaScript y la transición hacia un paradigma de programación declarativa. Se busca que el estudiante identifique la diferencia entre métodos mutables e inmutables, y aplique funciones declarativas para la manipulación eficiente de datos. La actividad sigue un flujo de trabajo profesional basado en Git y GitHub.

2. Instrucciones de Configuración y Entrega

Siga estrictamente estos pasos para garantizar la validez de su entrega:

1. **Fork del repositorio:** Realice un fork del repositorio base proporcionado por el instructor a su cuenta personal de GitHub.
2. **Clonación:** Clone su fork en su máquina local. Asegúrese de abrir la carpeta raíz del proyecto en su editor de código.
3. **Estructura de archivos:**
    - Ubíquese en la raíz del proyecto.
    - Cree una carpeta cuyo nombre sea su **Nombre Completo** (ejemplo: pedro-perez-rodriguez).
    - Dentro de esa carpeta, cree dos archivos: index.html y script.js.
    - Vincule el archivo script.js en el index.html mediante una etiqueta <script>.
4. **Desarrollo y Commits:**
    - Debe realizar un commit por cada ejercicio completado.
    - El mensaje del commit debe iniciar con el número del ejercicio (ejemplo: "Ejercicio 1: Implementación de push").
5. **Envío:** Una vez finalizados todos los ejercicios, suba los cambios a su fork y envíe un **Pull Request** al repositorio original.

3. Parte 1: Consulta Técnica (Programación Declarativa)

En la parte superior de su archivo script.js, incluya como comentarios la respuesta a las siguientes definiciones y conceptos:

1. **Mutabilidad vs Inmutabilidad:** Defina ambos conceptos y mencione tres métodos de arreglos para cada categoría.
2. **Programación Imperativa vs Declarativa:** Explique la diferencia principal entre escribir

código que dice "cómo" hacer las cosas frente a código que dice "qué" se quiere lograr.
3. **Funciones Declarativas (Arrow Functions):** Explique la sintaxis de las funciones de flecha y por qué son preferidas al usar métodos de arreglos.
4. **Efectos Secundarios (Side Effects):** ¿Qué ocurre cuando un método de arreglo modifica una variable fuera de su propio alcance?
5. **Conceptos de Búsqueda:** Defina el funcionamiento de find(), findIndex() e includes().
6. **Iteración vs Transformación:** Explique la diferencia técnica entre el uso de forEach() y map().

4. Parte 2: Ejercicios Prácticos

Desarrolle los siguientes ejercicios en su archivo script.js. **Es obligatorio el uso de funciones de flecha (arrow functions)** en todos los métodos que lo permitan. Imprima en consola el resultado y el estado del arreglo original.

Ejercicio 1: Gestión de Pilas (Mutable)

Partiendo de let herramientas = ["Martillo", "Destornillador"];:

1. Agregue "Taladro" al final y "Sierra" al inicio. Luego elimine el último elemento.
2. Commit: "Ejercicio 1 completado".

Ejercicio 2: Modificación de Índice (Mutable)

Partiendo de let colores = ["Rojo", "Verde", "Azul", "Amarillo"];:

1. Use splice para insertar "Naranja" en la posición 1 y reemplazar "Azul" por "Morado".
2. Commit: "Ejercicio 2 completado".

Ejercicio 3: Transformación Declarativa (Inmutable)

Partiendo de const temperaturas = [15, 20, 25, 30];:

1. Cree un arreglo fahrenheit usando map con la fórmula (C * 9/5) + 32.
2. Commit: "Ejercicio 3 completado".

Ejercicio 4: Filtrado Selectivo (Inmutable)

Partiendo de const inventario = [5, 12, 8, 130, 44];:

1. Cree un arreglo grandesValores con los números mayores a 10 usando filter.
2. Commit: "Ejercicio 4 completado".

Ejercicio 5: Acumulación de Datos (Inmutable)

Partiendo de const ventas = [100, 250, 150, 400];:

1. Obtenga el total de ventas usando el método reduce.
2. Commit: "Ejercicio 5 completado".

Ejercicio 6: Ordenamiento Alfabético (Mutable)

Partiendo de let nombres = ["Zulma", "Andrés", "Bernardo", "Carlos"];:

1. Ordene el arreglo alfabéticamente usando sort.
2. Commit: "Ejercicio 6 completado".

Ejercicio 7: Inversión de Datos (Mutable)

Partiendo de let orden = [1, 2, 3, 4, 5];:

1. Invierta la posición de los elementos usando reverse.
2. Commit: "Ejercicio 7 completado".

Ejercicio 8: Búsqueda de Valor (Declarativo)

Partiendo de const invitados = ["Juan", "Maria", "Pedro", "Luisa"];:

1. Use find para retornar el nombre "Pedro" si existe en la lista.
2. Commit: "Ejercicio 8 completado".

Ejercicio 9: Localización de Índice (Declarativo)

Partiendo de const precios = [45, 12, 89, 34];:

1. Use findIndex para encontrar la posición del valor 89.
2. Commit: "Ejercicio 9 completado".

Ejercicio 10: Comprobación de Existencia (Inmutable)

Partiendo de const cursos = ["Git", "React", "Node"];:

1. Use includes para verificar si "JavaScript" se encuentra en el listado.
2. Commit: "Ejercicio 10 completado".

Ejercicio 11: Validación Total (Declarativo)

Partiendo de const edades = [20, 25, 19, 30];:

1. Verifique si todos los elementos son mayores o iguales a 18 usando every.
2. Commit: "Ejercicio 11 completado".

Ejercicio 12: Validación Parcial (Declarativo)

Partiendo de const puntajes = [5, 8, 12, 3];:

1. Verifique si al menos uno de los puntajes es mayor a 10 usando some.
2. Commit: "Ejercicio 12 completado".

Ejercicio 13: Concatenación (Inmutable)

Partiendo de const filaA = ["A1", "A2"]; y const filaB = ["B1", "B2"];:

1. Una ambos arreglos en uno solo llamado todasLasFilas usando concat.

2. Commit: "Ejercicio 13 completado".

Ejercicio 14: Formateo de String (Inmutable)

Partiendo de const palabras = ["Hola", "mundo", "JavaScript"];:

1. Convierta el arreglo en una sola cadena de texto separada por espacios usando join.
2. Commit: "Ejercicio 14 completado".

Ejercicio 15: Extracción de Rango (Inmutable)

Partiendo de const base = ["Elemento0", "Elemento1", "Elemento2", "Elemento3"];:

1. Cree un subarreglo con los elementos en las posiciones 1 y 2 usando slice.
2. Commit: "Ejercicio 15 completado".

5. Criterios de Evaluación

- **Estructura:** Uso correcto de carpetas y vinculación de archivos.
- **Teoría:** Respuestas claras y fundamentadas en la sección de comentarios.
- **Calidad de Código:** Uso obligatorio de funciones de flecha y métodos declarativos.
- **Versionamiento:** Presencia de al menos 15 commits con los mensajes solicitados.
- **Entrega:** Pull Request correctamente ejecutado hacia la rama principal.