Ejercicio 18 explicado

```
function debounce(funcion, espera) {
    let timeoutId;
    return function (...args) {
        clearTimeout(timeoutId);
        timeoutId = setTimeout(() => {
            funcion.apply(this, args);
        }, espera);
    };
}

function buscar(texto) {
    console.log(`Buscando: ${texto}`);
}

// Crear una versión "debounced" de la función con 500ms de espera
const buscarDebounce = debounce(buscar, 500);

// Simulando varias llamadas rápidas
buscarDebounce("Ho"); // Se cancela
buscarDebounce("Hol"); // Se cancela
buscarDebounce("Hola"); // Se ejecuta después de 500ms sin interrupciones
buscarDebounce("Hola gran doble");
```

¡Te entiendo! Vamos a desglosarlo paso a paso para que quede más claro.

La función debounce() crea una nueva versión de la función original

- buscarDebounce NO es la misma función buscar().
- buscarDebounce es la función **retornada por debounce()**, que **espera antes de llamar** a buscar().
- Cualquier argumento que pasemos a buscarDebounce() se transfiere a buscar() usando ...args.

Ejemplo detallado

function buscar(texto) {

  console.log(` Buscando: ${texto}`);

```
}

// Se crea una versión "debounced" de buscar() con 500ms de espera
const buscarDebounce = debounce(buscar, 500);

// Ahora, cada llamada a buscarDebounce(...) pasa los argumentos a la función original
buscarDebounce("Hola");
buscarDebounce("Holaaa");
buscarDebounce("Holaaa mundo");
```

🔹 **Lo que sucede internamente:**

1. **La primera llamada** buscarDebounce("Hola") pone en cola un temporizador para ejecutar buscar("Hola") en 500ms.
2. **Luego llamamos** buscarDebounce("Holaaa"), se cancela la llamada anterior y ahora buscar("Holaaa") esperará 500ms.
3. **Finalmente**, buscarDebounce("Holaaa mundo") cancela las anteriores y buscar("Holaaa mundo") se ejecuta después de 500ms SIN interrupciones.

¿Dónde van los argumentos (...args)?

```
return function (...args) {
  clearTimeout(timeoutId);
  timeoutId = setTimeout(() => {
    funcion.apply(this, args);
  }, espera);
};
```

📌 **Aquí ...args captura los valores pasados a buscarDebounce()**
📌 **Luego funcion.apply(this, args) los transfiere a buscar()**

Si llamamos buscarDebounce("Hola"), entonces:

- ...args → Contiene ["Hola"]
- funcion.apply(this, args) → Ejecuta buscar("Hola") después del tiempo de espera.

📌 Básicamente, la nueva función (buscarDebounce) actúa como un intermediario entre el usuario y buscar(), asegurándose de que solo la última llamada se ejecute tras el tiempo de espera.

Espero que esta explicación te ayude. ¿Te gustaría otro ejemplo con más visualización? 🚀😀