cómo puedo darme cuenta de estas cosas? Explícame en qué tengo que fijarme para aprender a autocorregir mis propios ejercicios

La forma de darte cuenta es entrenar tres hábitos: **leer el flujo completo**, **vigilar el ámbito de las variables** y **comprobar qué se ejecuta realmente**.

1. Dibujar el flujo antes de mirar código

Cuando algo no cuadra, escribe o dibuja:

- Qué debería pasar "paso a paso" cuando:
  - entras en la pantalla
  - pulsas Grabar
  - pulsas Atrás
- Qué función se encarga de cada paso.

Luego compara tu lista con el código:

- "¿Esta acción del enunciado tiene exactamente **una** función que la haga?".
- "¿Estoy llamando a esa función en los momentos que dice el enunciado (entrar, después de 5 segundos, etc.)?".

Si ves que la misma acción (por ejemplo, "pintar el listado") aparece en dos sitios distintos del flujo, sospecha de **duplicidad** y revisa si de verdad hacen falta las dos llamadas.

2. Comprobar ámbito y existencia de variables

Cada vez que uses una variable en una función, pregúntate:

- "¿Dónde se **declara** esta variable?".
- "¿Está declarada en esta función, en una función externa o en global?".

Si se declaró con `let` o `const` dentro de otra función, **solo vive allí**. Para practicar:

- Marca con comentarios de qué tipo son las variables:
  - `// global`
  - `// local de pantalla3Comprobaciones`
  - etc.
- Si usas una variable local en otra función, deberías ver inmediatamente que no encaja.

Esto te evitará errores de tipo “X is not defined”.

3. Rastrear cuántas veces se llama a cada función

El problema que tenías con `mostrarSoluciones` es típico: la misma función se llama más veces de las que imaginabas.

Para detectarlo:

- Busca el nombre de la función en todo el archivo (`Ctrl+F` en el editor).
- Haz una lista rápida:
  - `mostrarSoluciones(retraso)` → al entrar.
  - `mostrarSoluciones(retraso)` → dentro de otra función.
  - `mostrarSoluciones(false)` → dentro de un `setTimeout`.

Si hay más llamadas de las que esperabas según el enunciado, revisa cuáles sobran.

4. Usar mensajes de consola

Otra ayuda muy práctica:

```
function mostrarSoluciones(retraso = false) {
  console.log('mostrarSoluciones llamada con retraso =', retraso);
  ...
}
```

Y en cada sitio donde la llamas:

```
console.log('Click en Grabar, antes de setTimeout');
```

Mirando la consola ves:

- En qué orden se llaman.
- Cuántas veces se ejecutan.
- Con qué valores (por ejemplo, si `retraso` se vuelve siempre `false`).

Esto te enseña a contrastar **lo que crees que pasa** con **lo que realmente pasa**.

5. Separar responsabilidades

Intenta que cada función haga una cosa clara:

- Una función guarda en cookies.
- Otra solo pinta la tabla.
- Otra solo valida el formulario.

Cuando una función hace “un poco de todo”, es más fácil duplicar trabajo o mezclar responsabilidades. Revisar que cada función tiene una responsabilidad te ayuda a ver si estás

llamando a la función correcta en el lugar correcto.[1] [2]

Si aplicas estos pasos en cada ejercicio (flujo, ámbito de variables, llamadas, consola y responsabilidades) vas a empezar a ver tú mismo dónde están las incoherencias sin necesidad de que alguien te lo señale.

⁂

1. https://blog.logrocket.com/solid-principles-single-responsibility-in-javascript-frameworks/
2. https://www.geeksforgeeks.org/system-design/single-responsibility-in-solid-design-principle/