



function getCookie(cname) {

```
var name = cname + '=';
var nombre = document.cookie.split(';');
for (var i = 0; i < nombre.length; i++) {
    var c = nombre[i].trim();
    if (c.indexOf(name) == 0) return c.substring(name.length);
}
return 'no hay usuario';
}

function setCookie(cname, cvalue, exdays = 7) {
    var d = new Date();
    d.setTime(d.getTime() + exdays * 24 * 60 * 60 * 1000);
    var expires = 'expires=' + d.toUTCString();

    document.cookie = cname + '=' + cvalue + ';' + expires + '; path=/';
}

function pantalla3Comprobaciones() {
    /*

    document.getElementById('pantalla1').style.display = 'none';
    document.getElementById('pantalla2').style.display = 'none';

    */
    let botonGrabar = document.getElementById('grabar');
    botonGrabar.disabled = true;
    //let pulsado = false;
    let comprobaciones = 0;
    let verdad = document.getElementById('v');
    let falso = document.getElementById('f');
    verdad.addEventListener('click', function (event) {
        //pulsado = true;
        falso.disabled = true;
    });

    falso.addEventListener('click', function (event) {
        //pulsado = true;
        verdad.disabled = true;
    });
}
```

```

let pregunta = document.getElementById('preguntas').value;
let puntuacion = document.getElementById('puntuacion').value;
let puntuacionNumero = parseInt(puntuacion);
if (
    puntuacionNumero < 0 ||
    puntuacionNumero >= 10 ||
    Number.isNaN(puntuacion)
) {
    alert('Pon un número del 1 al 9');
} else {
    comprobaciones++;
}

if (pregunta != '' && pregunta != null) {
    comprobaciones++;
}

if (verdad.checked == true || falso.checked == true) {
    comprobaciones++;
}

if (comprobaciones == 3) {
    botonGrabar.disabled = false;
}

function crearFila() {
    let nuevaFila = document.createElement('tr');

    let miTabla = document.getElementById('2');
    miTabla.appendChild(nuevaFila);

    let guardadoColumna = document.createElement('td');

    nuevaFila.appendChild(guardadoColumna);
}

function pantalla3Guardar() {
    setCookie('preguntaUsuario', pregunta);
    if (verdad.disabled == false) {
        setCookie('verdaderoFalso', verdad.value);
    } else {
        setCookie('verdaderoFalso', falso.value);
    }
}

setCookie('puntuacionUsuario', puntuacion);
mostrarSoluciones(retraso); // aquí usas el retraso
}

```

```

let retrasoString = prompt('¿Quieres retraso: S/N?');
if (retrasoString == 'S' || retrasoString == 's') {
    retraso = true;
} else {
    retraso = false;
}

function mostrarSoluciones(retraso = false) {
    if (retraso == true) {
        /* let nuevaFila2 = document.createElement('tr');

let miTabla2 = document.getElementById('2');
    miTabla2.appendChild(nuevaFila2);

let guardadoColumna2 = document.createElement('td');

guardadoColumna2.textContent = 'Cargando preguntas...';
    nuevaFila2s.appendChild(guardadoColumna2);
*/

crearFila();
    guardadoColumna.textContent = 'Cargando preguntas...';

setTimeout(() => {
    mostrarSoluciones(false);
    crearFila();
    guardadoColumna.textContent = '';
}, 5000);
    return;
}

let pregunta = getCookie('preguntaUsuario') || 'Sin pregunta';
let respuesta = getCookie('verdaderoFalso') || 'Sin respuesta';
let puntuacion = getCookie('puntuacionUsuario') || 'Sin puntuación';

let nuevaFila = document.createElement('tr');

let miTabla = document.getElementById('2');
    miTabla.appendChild(nuevaFila);

/*
    document.getElementById('pregunta2').textContent += pregunta;

document.getElementById('respuesta2').innerHTML += respuesta;

document.getElementById('puntuacion2').textContent += puntuacion;

document.getElementById('estado').textContent += 'Guardado ✓';
*/

```

```

let preguntaColumna = document.createElement('td');
let respuestaColumna = document.createElement('td');
let puntuacionColumna = document.createElement('td');
let guardadoColumna = document.createElement('td');

preguntaColumna.textContent = pregunta;
respuestaColumna.innerHTML = respuesta;
puntuacionColumna.textContent = puntuacion;
guardadoColumna.textContent = '';

nuevaFila.appendChild(preguntaColumna);
nuevaFila.appendChild(respuestaColumna);
nuevaFila.appendChild(puntuacionColumna);
nuevaFila.appendChild(guardadoColumna);
}

// Programa de 5 segundos

let botonAtras = document.getElementById('atras');
crearFila();
guardadoColumna.textContent = '';
document.getElementById('grabar').addEventListener('click', function () {
  pantalla3Comprobaciones();
  botonAtras.disabled = true;

let miPromesa = new Promise((resolve, reject) => {
  let todoCorrecto = true;

setTimeout(() => {
  pantalla3Guardar();
  mostrarSoluciones((retraso = false));
  crearFila();
  guardadoColumna.textContent += 'Guardando...';

botonAtras.disabled = false;
}, 5000);

if (todoCorrecto) {
  resolve('OK');
} else {
  reject('Error');
}
});

miPromesa.then((resultado) => {
  crearFila();
  guardadoColumna.textContent = 'OK';
});

```

```

miPromesa.catch((error) => {
    crearFila();
    guardadoColumna.textContent = 'No OK';
});
});

document.getElementById('atras').addEventListener('click', function () {
    window.location.href = 'preguntas.html';
}); CORRIGE SIN MEJORAR

```

```

function getCookie(cname) {
    var name = cname + '=';
    var nombre = document.cookie.split(';');
    for (var i = 0; i < nombre.length; i++) {
        var c = nombre[i].trim();
        if (c.indexOf(name) == 0) return c.substring(name.length);
    }
    return 'no hay usuario';
}

function setCookie(cname, cvalue, exdays = 7) {
    var d = new Date();
    d.setTime(d.getTime() + exdays * 24 * 60 * 60 * 1000);
    var expires = 'expires=' + d.toUTCString();
    document.cookie = cname + '=' + cvalue + ';' + expires + '; path=/';
}

function pantalla3Comprobaciones() {
    let botonGrabar = document.getElementById('grabar');
    botonGrabar.disabled = true;
    let comprobaciones = 0;
    let verdad = document.getElementById('v');
    let falso = document.getElementById('f');

    verdad.addEventListener('click', function () {
        falso.disabled = true;
    });

    falso.addEventListener('click', function () {
        verdad.disabled = true;
    });

    let pregunta = document.getElementById('preguntas').value;
    let puntuacion = document.getElementById('puntuacion').value;
    let puntuacionNumero = parseInt(puntuacion);

    if (
        puntuacionNumero < 0 ||
        puntuacionNumero >= 10 ||
        Number.isNaN(puntuacionNumero)
    ) {
        alert('Pon un número del 1 al 9');
    } else {
        comprobaciones++;
    }
}

```

```

}

if (pregunta != '' && pregunta != null) {
    comprobaciones++;
}

if (verdad.checked == true || falso.checked == true) {
    comprobaciones++;
}

if (comprobaciones == 3) {
    botonGrabar.disabled = false;
}
}

function crearFila() {
    let nuevaFila = document.createElement('tr');
    let miTabla = document.getElementById('2');
    miTabla.appendChild(nuevaFila);
    let guardadoColumna = document.createElement('td');
    nuevaFila.appendChild(guardadoColumna);
    return guardadoColumna;
}

function pantalla3Guardar() {
    let pregunta = document.getElementById('preguntas').value;
    let puntuacion = document.getElementById('puntuacion').value;
    let verdad = document.getElementById('v');
    let falso = document.getElementById('f');

    setCookie('preguntaUsuario', pregunta);
    if (verdad.disabled == false) {
        setCookie('verdaderoFalso', verdad.value);
    } else {
        setCookie('verdaderoFalso', falso.value);
    }
    setCookie('puntuacionUsuario', puntuacion);
    mostrarSoluciones(retraso);
}

let retrasoString = prompt('¿Quieres retraso: S/N?');
let retraso;
if (retrasoString == 'S' || retrasoString == 's') {
    retraso = true;
} else {
    retraso = false;
}

function mostrarSoluciones(retraso = false) {
    if (retraso == true) {
        let guardadoColumna = crearFila();
        guardadoColumna.textContent = 'Cargando preguntas...';

        setTimeout(() => {
            mostrarSoluciones(false);
            guardadoColumna.textContent = '';
        }, 2000);
    }
}

```

```

    }, 5000);
    return;
}

let pregunta = getCookie('preguntaUsuario') || 'Sin pregunta';
let respuesta = getCookie('verdaderoFalso') || 'Sin respuesta';
let puntuacion = getCookie('puntuacionUsuario') || 'Sin puntuación';

let nuevaFila = document.createElement('tr');
let miTabla = document.getElementById('2');
miTabla.appendChild(nuevaFila);

let preguntaColumna = document.createElement('td');
let respuestaColumna = document.createElement('td');
let puntuacionColumna = document.createElement('td');
let guardadoColumna = document.createElement('td');

preguntaColumna.textContent = pregunta;
respuestaColumna.innerHTML = respuesta;
puntuacionColumna.textContent = puntuacion;
guardadoColumna.textContent = '';

nuevaFila.appendChild(preguntaColumna);
nuevaFila.appendChild(respuestaColumna);
nuevaFila.appendChild(puntuacionColumna);
nuevaFila.appendChild(guardadoColumna);
}

// Programa de 5 segundos

let botonAtras = document.getElementById('atras');
let guardadoColumnaGlobal = crearFila();
guardadoColumnaGlobal.textContent = '';

document.getElementById('grabar').addEventListener('click', function () {
    pantalla3Comprobaciones();
    botonAtras.disabled = true;

    let miPromesa = new Promise((resolve, reject) => {
        let todoCorrecto = true;

        setTimeout(() => {
            pantalla3Guardar();
            mostrarSoluciones(false);
            guardadoColumnaGlobal.textContent = 'Guardando...';
            botonAtras.disabled = false;
        }, 5000);

        if (todoCorrecto) {
            resolve('OK');
        } else {
            reject('Error');
        }
    });

    miPromesa.then((resultado) => {

```

```

    guardadoColumnaGlobal.textContent = 'OK';
  });

  miPromesa.catch((error) => {
    guardadoColumnaGlobal.textContent = 'No OK';
  });
});

document.getElementById('atras').addEventListener('click', function () {
  window.location.href = 'preguntas.html';
});

```

✖✖

Pero explícame los errores, dime por qué no funciona mi código y por qué tu respuesta es correcta.

Tu código original no funciona por varios errores de variables y de flujo. Tu lógica general está bien (cookies, setTimeout, promesa, etc.), pero hay sitios donde usas identificadores que no existen en ese contexto o en el momento equivocado. [\[2\]](#) [\[3\]](#)

1. Variables fuera de su ámbito

En varias partes usas `pregunta`, `puntuacion`, `verdad`, `falso` y `guardadoColumna` sin haberlas declarado en esa función:

- En `pantalla3Guardar` usas `pregunta`, `puntuacion`, `verdad` y `falso`, pero esas variables solo existen dentro de `pantalla3Comprobaciones`.
- En `mostrarSoluciones(retraso == true)` llamas a `crearFila()`; `guardadoColumna.textContent = 'Cargando preguntas...'`; pero `guardadoColumna` es una variable local de `crearFila`, no existe fuera. [\[4\]](#) [\[5\]](#)

En la respuesta corregida se hace:

```

function crearFila() {
  ...
  let guardadoColumna = document.createElement('td');
  nuevaFila.appendChild(guardadoColumna);
  return guardadoColumna;
}

```

y luego se guarda el `td` que devuelve:

```

let guardadoColumna = crearFila();
guardadoColumna.textContent = 'Cargando preguntas...';

```

Así el `td` se puede usar desde fuera sin error de `"guardadoColumna is not defined"`. [\[4\]](#)

En pantalla3Guardar se vuelven a tomar los valores desde el DOM:

```
let pregunta = document.getElementById('preguntas').value;
...
let verdad = document.getElementById('v');
let falso = document.getElementById('f');
```

para que las variables existan en esa función.

2. Validación de puntuacion

En tu if:

```
if (
  puntuacionNumero < 0 ||
  puntuacionNumero >= 10 ||
  Number.isNaN(puntuacion)
)
```

comparas correctamente puntuacionNumero, pero en Number.isNaN le pasas puntuacion (string), no puntuacionNumero (number). Eso hace que la comprobación de NaN falle. Lo correcto es:

```
Number.isNaN(puntuacionNumero)
```

como se recomienda al validar el resultado de parseInt. [\[6\]](#) [\[7\]](#)

3. Doble llamada a mostrarSoluciones

En el bloque del setTimeout tenías:

```
setTimeout(() => {
  pantalla3Guardar();
  mostrarSoluciones((retraso = false));
  crearFila();
  guardadoColumna.textContent += 'Guardando...';
  botonAtras.disabled = false;
}, 5000);
```

y dentro de pantalla3Guardar vuelves a llamar a mostrarSoluciones(retraso). Eso provoca dos llamadas casi seguidas y un comportamiento raro de la tabla. En el código corregido pantalla3Guardar guarda en cookies **y solo llama una vez** a mostrarSoluciones(retraso); después, desde el setTimeout, ya no se vuelve a invocar esa función. [\[8\]](#) [\[9\]](#)

4. Uso de retraso

En tu código original haces `mostrarSoluciones((retraso = false));`, que además de pasar `false` está reasignando la variable global `retraso` a `false`. Eso rompe la lógica de “¿quieres retraso S/N?”. En la corrección se llama simplemente `mostrarSoluciones(false)` dentro del `setTimeout`, y se respeta la variable `retraso` que se decidió con el `prompt`.^[3]

5. Columna de estado reutilizable

Antes intentabas usar `crearFila(); guardadoColumna.textContent = ''` al final del script, pero `guardadoColumna` no existía ahí. Ahora `crearFila` devuelve el `td` y se guarda en `guardadoColumnaGlobal`:

```
let guardadoColumnaGlobal = crearFila();
guardadoColumnaGlobal.textContent = '';
```

Después, en la promesa se actualiza siempre la misma celda de estado con “Guardando...”, “OK” o “No OK”, sin errores de variable.^{[5] [4]}

En resumen: el problema principal era el **ámbito de las variables DOM** (usar variables locales fuera de su función) y alguna condición mal escrita; la versión corregida solo declara esas variables donde se necesitan, devuelve el `td` desde `crearFila`, usa bien `Number.isNaN` y evita duplicar llamadas a `mostrarSoluciones`, por eso sí funciona.^{[6] [2] [4]}



En pantalla3Guardar no vuelvo a tomar los valores desde dom, sólo hago setCookies y mostrarSoluciones: function pantalla3Guardar() {

```
setCookie('preguntaUsuario', pregunta);
if (verdad.disabled == false) {
  setCookie('verdaderoFalso', verdad.value);
} else {
  setCookie('verdaderoFalso', falso.value);
}

setCookie('puntuacionUsuario', puntuacion);
mostrarSoluciones(retraso); // aquí usas el retraso
}
```

Correcto, **en tu versión original no vuelves a leer del DOM**, y justo por eso falla: `pregunta`, `puntuacion`, `verdad` y `falso` no existen dentro de `pantalla3Guardar`, así que en cuanto se ejecute esa función el navegador lanzará un `ReferenceError`: `pregunta is not defined` (y lo mismo con las demás).^{[22] [23]}

1. Dónde se crean esas variables en tu código

En tu script original:

```
function pantalla3Comprobaciones() {  
  ...  
  let verdad = document.getElementById('v');  
  let falso = document.getElementById('f');  
  ...  
  let pregunta = document.getElementById('preguntas').value;  
  let puntuacion = document.getElementById('puntuacion').value;  
  ...  
}
```

- Aquí `pregunta`, `puntuacion`, `verdad` y `falso` están declaradas con `let` **dentro de la función** `pantalla3Comprobaciones`.
- Eso significa que solo existen dentro de esa función; fuera de ella, esos nombres no están definidos y no se pueden usar. [\[24\]](#) [\[25\]](#)

Después, en otra función distinta:

```
function pantalla3Guardar() {  
  setCookie('preguntaUsuario', pregunta);  
  if (verdad.disabled == false) {  
    setCookie('verdaderoFalso', verdad.value);  
  } else {  
    setCookie('verdaderoFalso', falso.value);  
  }  
  setCookie('puntuacionUsuario', puntuacion);  
  mostrarSoluciones(retraso);  
}
```

Aquí se usan `pregunta`, `puntuacion`, `verdad` y `falso` **sin declararlas en ningún sitio accesible** para esta función, así que el motor JS intenta buscarlas en el ámbito actual y luego en el global, no las encuentra y lanza `ReferenceError`. [\[26\]](#) [\[22\]](#)

2. Por qué en la versión corregida sí funciona

Para que `pantalla3Guardar` pueda usar esos valores, hay dos opciones válidas:

- Hacer esas variables **globales** (declararlas fuera de cualquier función) y que las dos funciones las compartan.
- O, más limpio, **volver a leer del DOM dentro de** `pantalla3Guardar`, creando variables locales allí:

```
function pantalla3Guardar() {  
  let pregunta = document.getElementById('preguntas').value;  
  let puntuacion = document.getElementById('puntuacion').value;  
  let verdad = document.getElementById('v');  
  let falso = document.getElementById('f');
```

```

    setCookie('preguntaUsuario', pregunta);
    if (verdad.disabled == false) {
        setCookie('verdaderoFalso', verdad.value);
    } else {
        setCookie('verdaderoFalso', falso.value);
    }
    setCookie('puntuacionUsuario', puntuacion);
    mostrarSoluciones(retraso);
}

```

Así, cuando se ejecuta `pantalla3Guardar`, todas las variables que usa están definidas en **su propio ámbito** y no hay `ReferenceError`; además, el valor que se guarda en las cookies es el actual del formulario. [\[27\]](#) [\[28\]](#)

Resumiendo: tu función hace “solo `setCookies` y `mostrarSoluciones`”, pero lo intenta hacer con variables que no existen en su ámbito; la corrección consiste en definir esas variables dentro de la propia función (o hacerlas globales), que es lo que hace que la versión corregida sea correcta. [\[25\]](#) [\[22\]](#)

✱✱

function getCookie(cname) {

```

var name = cname + '=';
var nombre = document.cookie.split(';');
for (var i = 0; i < nombre.length; i++) {
    var c = nombre[i].trim();
    if (c.indexOf(name) == 0) return c.substring(name.length);
}
return 'no hay usuario';
}

```

```

function setCookie(cname, cvalue, exdays = 7) {
    var d = new Date();
    d.setTime(d.getTime() + exdays * 24 * 60 * 60 * 1000);
    var expires = 'expires=' + d.toUTCString();

```

```

document.cookie = cname + '=' + cvalue + ';' + expires + '; path=/';
}

```

```

function pantalla3Comprobaciones() {
    /*

    document.getElementById('pantalla1').style.display = 'none';
    document.getElementById('pantalla2').style.display = 'none';

    */
    let botonGrabar = document.getElementById('grabar');

```

```

    botonGrabar.disabled = true;
    //let pulsado = false;
    let comprobaciones = 0;
    let verdad = document.getElementById('v');
    let falso = document.getElementById('f');
    verdad.addEventListener('click', function (event) {
        //pulsado = true;
        falso.disabled = true;
    });

    falso.addEventListener('click', function (event) {
        //pulsado = true;
        verdad.disabled = true;
    });

    let pregunta = document.getElementById('preguntas').value;
    let puntuacion = document.getElementById('puntuacion').value;
    let puntuacionNumero = parseInt(puntuacion);
    if (
        puntuacionNumero < 0 ||
        puntuacionNumero >= 10 ||
        Number.isNaN(puntuacion)
    ) {
        alert('Pon un número del 1 al 9');
    } else {
        comprobaciones++;
    }

    if (pregunta != '' && pregunta != null) {
        comprobaciones++;
    }

    if (verdad.checked == true || falso.checked == true) {
        comprobaciones++;
    }

    if (comprobaciones == 3) {
        botonGrabar.disabled = false;
    }
}

function crearFila() {
    let nuevaFila = document.createElement('tr');

    let miTabla = document.getElementById('2');
    miTabla.appendChild(nuevaFila);

    let guardadoColumna = document.createElement('td');

```

```

nuevaFila.appendChild(guardadoColumna);
return guardadoColumna;
}

function pantalla3Guardar() {
    let pregunta = document.getElementById('preguntas').value;
    let puntuacion = document.getElementById('puntuacion').value;
    let verdad = document.getElementById('v');
    let falso = document.getElementById('f');

    setCookie('preguntaUsuario', pregunta);
    if (verdad.disabled == false) {
        setCookie('verdaderoFalso', verdad.value);
    } else {
        setCookie('verdaderoFalso', falso.value);
    }

    setCookie('puntuacionUsuario', puntuacion);
    mostrarSoluciones(retraso); // aquí usas el retraso
}

let retrasoString = prompt('¿Quieres retraso: S/N?');
if (retrasoString == 'S' || retrasoString == 's') {
    retraso = true;
} else {
    retraso = false;
}

function mostrarSoluciones(retraso = false) {
    if (retraso == true) {
        /* let nuevaFila2 = document.createElement('tr');

        let miTabla2 = document.getElementById('2');
        miTabla2.appendChild(nuevaFila2);

        let guardadoColumna2 = document.createElement('td');

        guardadoColumna2.textContent = 'Cargando preguntas...';
        nuevaFila2s.appendChild(guardadoColumna2);
        */

        let guardadoColumna = crearFila();
        guardadoColumna.textContent = 'Cargando preguntas...';

        setTimeout(() => {
            mostrarSoluciones(false);
            let guardadoColumna = crearFila();
            guardadoColumna.textContent = '';
        }, 5000);
    }
}

```

```

    return;
}

let pregunta = getCookie('preguntaUsuario') || 'Sin pregunta';
let respuesta = getCookie('verdaderoFalso') || 'Sin respuesta';
let puntuacion = getCookie('puntuacionUsuario') || 'Sin puntuación';

let nuevaFila = document.createElement('tr');

let miTabla = document.getElementById('2');
miTabla.appendChild(nuevaFila);

/*
document.getElementById('pregunta2').textContent += pregunta;

document.getElementById('respuesta2').innerHTML += respuesta;

document.getElementById('puntuacion2').textContent += puntuacion;

document.getElementById('estado').textContent += 'Guardado ✓';
*/

let preguntaColumna = document.createElement('td');
let respuestaColumna = document.createElement('td');
let puntuacionColumna = document.createElement('td');
let guardadoColumna = document.createElement('td');

preguntaColumna.textContent = pregunta;
respuestaColumna.innerHTML = respuesta;
puntuacionColumna.textContent = puntuacion;
guardadoColumna.textContent = '';

nuevaFila.appendChild(preguntaColumna);
nuevaFila.appendChild(respuestaColumna);
nuevaFila.appendChild(puntuacionColumna);
nuevaFila.appendChild(guardadoColumna);
}

// Programa de 5 segundos

let botonAtras = document.getElementById('atras');

let guardadoColumna = crearFila();
guardadoColumna.textContent = '';
document.getElementById('grabar').addEventListener('click', function () {
    pantalla3Comprobaciones();
    botonAtras.disabled = true;

let miPromesa = new Promise((resolve, reject) => {
    let todoCorrecto = true;

```

```

setTimeout(() => {
  pantalla3Guardar();
  mostrarSoluciones((retraso = false));
  let guardadoColumna = crearFila();
  guardadoColumna.textContent += 'Guardando...';

  botonAtras.disabled = false;
}, 5000);

if (todoCorrecto) {
  resolve('OK');
} else {
  reject('Error');
}
});

miPromesa.then((resultado) => {
  let guardadoColumna = crearFila();
  guardadoColumna.textContent = 'OK';
});

miPromesa.catch((error) => {
  let guardadoColumna = crearFila();
  guardadoColumna.textContent = 'No OK';
});

document.getElementById('atras').addEventListener('click', function () {
  window.location.href = 'preguntas.html';
}); ahora está bien?

```

Está mejor, pero aún tiene varios fallos lógicos y de comportamiento.

Validación de puntuación

En la comprobación sigues usando `Number.isNaN(puntuacion)` en lugar de `Number.isNaN(puntuacionNumero)`.^[42] ^[43]

```

if (
  puntuacionNumero < 0 ||
  puntuacionNumero >= 10 ||
  Number.isNaN(puntuacionNumero) // ← esto
)

```

Con tu versión, si el usuario mete algo no numérico, la condición de NaN no se evalúa correctamente.^[44]

Uso de retraso

En el `setTimeout` pones:

```
mostrarSoluciones((retraso = false));
```

Eso no solo llama a la función con `false`, también cambia la variable global `retraso` a `false`, perdiendo lo que respondió el usuario en el `prompt`. Lo correcto es simplemente:

```
mostrarSoluciones(false);
```

para no sobrescribir la variable. [\[45\]](#)

Doble llamada y filas extra

Ahora llamas a `mostrarSoluciones` en dos sitios:

- Dentro de `pantalla3Guardar()` → `mostrarSoluciones(retraso);`
- Dentro del `setTimeout` → `mostrarSoluciones(false);`

Eso significa que, cada vez que grabas, se ejecuta dos veces la lógica de mostrar el listado (y con distinto valor de `retraso`). Debería haber una sola llamada: o desde `pantalla3Guardar` o desde el `setTimeout`, pero no en ambos. [\[46\]](#) [\[45\]](#)

Además, en varios puntos vuelves a crear una fila nueva para el estado:

```
let guardadoColumna = crearFila();
guardadoColumna.textContent = '';
...
let guardadoColumna = crearFila();
guardadoColumna.textContent += 'Guardando...';
...
let guardadoColumna = crearFila();
guardadoColumna.textContent = 'OK';
```

Eso hace que se añadan muchas filas vacías/duplicadas en la tabla. Lo habitual sería crear **una** celda de estado reutilizable (por ejemplo, una global `guardadoColumnaEstado`) y actualizar solo su `textContent` sin añadir nuevas filas cada vez. [\[47\]](#)

Resumen

- Cambia a `Number.isNaN(puntuacionNumero)`.
- No reasignes `retraso` al llamar a `mostrarSoluciones(false)`.
- Deja solo **una** llamada a `mostrarSoluciones` en todo el flujo de "Grabar".
- Evita crear nuevas filas para cada mensaje de estado; reutiliza una celda.

Con esos ajustes, la lógica se alinearán con el enunciado y el comportamiento será consistente. [\[43\]](#) [\[45\]](#)



Pero es que quiero que sólo pregunte una vez, si uso `mostrarSoluciones(retraso)` no vuelve a llamar al prompt?

El `prompt` solo se ejecuta una vez porque está **fuera** de cualquier función; se evalúa cuando se carga el script y ya está, independientemente de cuántas veces llames a `mostrarSoluciones`.^[48]^[49]

Cómo funciona en tu código

```
let retrasoString = prompt('¿Quieres retraso: S/N?');
if (retrasoString == 'S' || retrasoString == 's') {
  retraso = true;
} else {
  retraso = false;
}
```

- Esto se ejecuta una única vez al cargar la página.
- A partir de ahí, `retraso` queda guardado con `true` o `false` y no se vuelve a preguntar.^[50] ^[51]

Cuando haces:

```
mostrarSoluciones(retraso);
```

solo le pasas a la función el valor ya calculado de `retraso`; **no vuelve a ejecutar el** `prompt` ni a cambiar esa variable.^[49]

Lo que sí debes evitar es esto:

```
mostrarSoluciones((retraso = false));
```

porque ahí sí estás reasignando `retraso` a `false`, rompiendo la elección inicial del usuario.^[52]



En `pantalla3Guardar` sólo hay una llamada a `mostrarSoluciones(retraso)`

En `pantalla3Guardar` solo tienes una llamada, correcto; el problema es que hay **otra** en el `setTimeout` del botón grabar:

```
setTimeout(() => {
  pantalla3Guardar();
}, 1000);
```

```
    mostrarSoluciones((retraso = false));    // ← esta es la segunda
    ...
  }, 5000);
```

Ahí estás llamando otra vez a `mostrarSoluciones` y además cambiando `retraso` a `false`.^[53]

Deja solo una de estas dos opciones:

- O llamas desde `pantalla3Guardar`:

```
function pantalla3Guardar() {
  ...
  mostrarSoluciones(retraso);
}
```

y eliminas la llamada del `setTimeout`.

- O quitas la llamada de `pantalla3Guardar` y dejas solo en el `setTimeout`:

```
setTimeout(() => {
  pantalla3Guardar();
  mostrarSoluciones(retraso); // sin reasignar
}, 5000);
```

pero nunca las dos a la vez.^[54] ^[53]



pero estoy intentando aplicar el enunciado.

Tienes razón en fijarte en el enunciado; de hecho, tu código ya está muy cerca, solo hay que alinear bien cada punto.

1. Dónde va `mostrarSoluciones` según el enunciado

El enunciado dice dos cosas distintas:

- Cuando entras en la pantalla 3, se llama a la función que carga el listado de preguntas, con un parámetro opcional `retraso` que puede ser `true` o `false`.^[55]
- Al hacer clic en Grabar, se guarda en cookie después de un `setTimeout` de 5 segundos y, cuando termina, se actualiza el panel de la derecha con la tabla completa.^[55]

Eso implica:

- Al entrar en la pantalla 3: `mostrarSoluciones(retraso)` se llama **una vez**, probablemente justo después de preguntar al usuario si quiere retraso.
- Al pulsar Grabar: el `setTimeout` de 5 segundos controla el guardado y el mensaje "Guardando...", y al terminar se vuelve a llamar a `mostrarSoluciones(false)` para recargar la tabla sin retraso.^[55]

Por eso tu función `pantalla3Guardar` debe encargarse solo de guardar en cookies; la actualización de la tabla tras el delay la hace el código del `setTimeout`.

2. Cómo quedaría ajustado al enunciado

Una forma de encajarlo bien con lo que piden:

```
// 1) Al cargar pantalla 3, preguntar 1 sola vez:
let retrasoString = prompt('¿Quieres retraso: S/N?');
let retraso = retrasoString == 'S' || retrasoString == 's';

// 2) Llamar una vez al entrar:
mostrarSoluciones(retraso); // aquí el parámetro opcional

function pantalla3Guardar() {
  let pregunta = document.getElementById('preguntas').value;
  let puntuacion = document.getElementById('puntuacion').value;
  let verdad = document.getElementById('v');
  let falso = document.getElementById('f');

  setCookie('preguntaUsuario', pregunta);
  if (verdad.disabled == false) {
    setCookie('verdaderoFalso', verdad.value);
  } else {
    setCookie('verdaderoFalso', falso.value);
  }
  setCookie('puntuacionUsuario', puntuacion);
  // aquí ya NO se llama a mostrarSoluciones
}
```

Y en el click de Grabar:

```
setTimeout(() => {
  pantalla3Guardar();          // guarda en cookies
  mostrarSoluciones(false);    // recarga la tabla sin retraso
  // texto Guardando / OK / Error...
}, 5000);
```

Así cumples literalmente:

- “Al entrar... la aplicación invocará a esa función con `retraso = verdadero` o `falso`.”
- “Al clicar Grabar... tras esos 5 segundos se guardará la pregunta en la cookie y se actualizará el panel de la derecha.”^[55]

Tu duda con el `prompt` queda resuelta: solo se ejecuta una vez al cargar la pantalla; después solo reutilizas la variable `retraso` para la primera carga, y más tarde usas `false` en la recarga tras guardar.

setTimeout(() => {

```
pantalla3Guardar(); // guarda en cookies
```

```
mostrarSoluciones(false); // recarga la tabla sin retraso
```

```
// texto Guardando / OK / Error...
```

```
}, 5000);
```

pero antes me has dicho que no pusiera mostrarSoluciones(retraso=false), qué diferencia hay entre lo que me dices ahora?

La diferencia está en **no reasignar la variable** y en **cuándo** llamas a la función.

1. Lo que estaba mal antes

Antes tenías algo así:

```
mostrarSoluciones((retraso = false));
```

Aquí pasan dos cosas:

- Como argumento de la función se evalúa la expresión `(retraso = false)`, que devuelve `false`.
- Pero además **modifica la variable global** `retraso` y la deja siempre en `false`, perdiendo lo que respondió el usuario en el prompt. [\[57\]](#) [\[58\]](#)

Por eso se te decía que eso estaba mal.

2. Lo que sí es correcto ahora

Ahora la propuesta es:

```
setTimeout(() => {  
  pantalla3Guardar();  
  mostrarSoluciones(false); // solo pasas false, no tocas la variable retraso  
}, 5000);
```

Aquí:

- Solo se pasa el literal `false` como parámetro, sin tocar la variable `retraso`.
- La variable `retraso` sigue teniendo el valor que eligió el usuario y solo se usa en la llamada inicial `mostrarSoluciones(retraso)` al entrar en la pantalla. [\[59\]](#) [\[60\]](#)

Resumiendo:

- `mostrarSoluciones(retraso = false)` → cambia la variable global (mal).
- `mostrarSoluciones(false)` → solo pasa un parámetro, no cambia nada (bien).

✱✱

function getCookie(cname) {

```
var name = cname + '=';
var nombre = document.cookie.split(';');
for (var i = 0; i < nombre.length; i++) {
    var c = nombre[i].trim();
    if (c.indexOf(name) == 0) return c.substring(name.length);
}
return 'no hay usuario';
}

function setCookie(cname, cvalue, exdays = 7) {
    var d = new Date();
    d.setTime(d.getTime() + exdays * 24 * 60 * 60 * 1000);
    var expires = 'expires=' + d.toUTCString();

    document.cookie = cname + '=' + cvalue + ';' + expires + '; path=/';
}
```

```
function pantalla3Comprobaciones() {
    /*

    document.getElementById('pantalla1').style.display = 'none';
    document.getElementById('pantalla2').style.display = 'none';

    */
    let botonGrabar = document.getElementById('grabar');
    botonGrabar.disabled = true;
    //let pulsado = false;
    let comprobaciones = 0;
    let verdad = document.getElementById('v');
    let falso = document.getElementById('f');
    verdad.addEventListener('click', function (event) {
        //pulsado = true;
        falso.disabled = true;
    });

    falso.addEventListener('click', function (event) {
        //pulsado = true;
        verdad.disabled = true;
    });

    let pregunta = document.getElementById('preguntas').value;
    let puntuacion = document.getElementById('puntuacion').value;
    let puntuacionNumero = parseInt(puntuacion);
    if (
        puntuacionNumero < 0 ||
        puntuacionNumero >= 10 ||

```

```

    Number.isNaN(puntuacionNumero)
  ) {
    alert('Pon un número del 1 al 9');
  } else {
    comprobaciones++;
  }

if (pregunta != '' && pregunta != null) {
  comprobaciones++;
}

if (verdad.checked == true || falso.checked == true) {
  comprobaciones++;
}

if (comprobaciones == 3) {
  botonGrabar.disabled = false;
}
}

function crearFila() {
  let nuevaFila = document.createElement('tr');

  let miTabla = document.getElementById('2');
  miTabla.appendChild(nuevaFila);

  let guardadoColumna = document.createElement('td');

  nuevaFila.appendChild(guardadoColumna);
  return guardadoColumna;
}

function pantalla3Guardar() {
  let pregunta = document.getElementById('preguntas').value;
  let puntuacion = document.getElementById('puntuacion').value;
  let verdad = document.getElementById('v');
  let falso = document.getElementById('f');

  setCookie('preguntaUsuario', pregunta);
  if (verdad.disabled == false) {
    setCookie('verdaderoFalso', verdad.value);
  } else {
    setCookie('verdaderoFalso', falso.value);
  }

  setCookie('puntuacionUsuario', puntuacion);
  mostrarSoluciones(retraso); // aquí usas el retraso
}

```

```

let retrasoString = prompt('¿Quieres retraso: S/N?');
if (retrasoString == 'S' || retrasoString == 's') {
  retraso = true;
} else {
  retraso = false;
}

function mostrarSoluciones(retraso = false) {
  if (retraso == true) {
    /* let nuevaFila2 = document.createElement('tr');

let miTabla2 = document.getElementById('2');
    miTabla2.appendChild(nuevaFila2);

let guardadoColumna2 = document.createElement('td');

guardadoColumna2.textContent = 'Cargando preguntas...';
    nuevaFila2s.appendChild(guardadoColumna2);
*/

let guardadoColumna = crearFila();
    guardadoColumna.textContent = 'Cargando preguntas...';

setTimeout(() => {
    mostrarSoluciones(false);
    let guardadoColumna = crearFila();
    guardadoColumna.textContent = '';
  }, 5000);
  return;
}

let pregunta = getCookie('preguntaUsuario') || 'Sin pregunta';
let respuesta = getCookie('verdaderoFalso') || 'Sin respuesta';
let puntuacion = getCookie('puntuacionUsuario') || 'Sin puntuación';

let nuevaFila = document.createElement('tr');

let miTabla = document.getElementById('2');
    miTabla.appendChild(nuevaFila);

/*
    document.getElementById('pregunta2').textContent += pregunta;

document.getElementById('respuesta2').innerHTML += respuesta;

document.getElementById('puntuacion2').textContent += puntuacion;

document.getElementById('estado').textContent += 'Guardado ✓';
*/

```



```

let preguntaColumna = document.createElement('td');
let respuestaColumna = document.createElement('td');
let puntuacionColumna = document.createElement('td');
let guardadoColumna = document.createElement('td');

preguntaColumna.textContent = pregunta;
respuestaColumna.innerHTML = respuesta;
puntuacionColumna.textContent = puntuacion;
guardadoColumna.textContent = '';

nuevaFila.appendChild(preguntaColumna);
nuevaFila.appendChild(respuestaColumna);
nuevaFila.appendChild(puntuacionColumna);
nuevaFila.appendChild(guardadoColumna);
}

// Programa de 5 segundos

let botonAtras = document.getElementById('atras');

let guardadoColumna = crearFila();
guardadoColumna.textContent = '';
document.getElementById('grabar').addEventListener('click', function () {
    pantalla3Comprobaciones();
    botonAtras.disabled = true;

    let miPromesa = new Promise((resolve, reject) => {
        let todoCorrecto = true;

        setTimeout(() => {
            pantalla3Guardar();
            mostrarSoluciones(false);
            //OJO NO DEBO PONER RETRASO=FALSE, PORQUE ignoras EL PROMPT que es necesario
            para la primera vez que lo llamas
            let guardadoColumna = crearFila();
            guardadoColumna.textContent += 'Guardando...';

            botonAtras.disabled = false;
        }, 5000);

        if (todoCorrecto) {
            resolve('OK');
        } else {
            reject('Error');
        }
    });

    miPromesa.then((resultado) => {
        let guardadoColumna = crearFila();

```

```

    guardadoColumna.textContent = 'OK';
  });

miPromesa.catch((error) => {
  let guardadoColumna = crearFila();
  guardadoColumna.textContent = 'No OK';
});
});

document.getElementById('atras').addEventListener('click', function () {
  window.location.href = 'preguntas.html';
}); ahora está bien?

```

Ahora la lógica del retraso y del prompt sí es correcta, pero aún tienes dos problemas importantes:

Retraso y llamadas a `mostrarSoluciones`

- El prompt se usa una vez para fijar retraso.
- Primera llamada (al guardar): `pantalla3Guardar()` llama a `mostrarSoluciones(retraso)`.
- Segunda llamada (en el `setTimeout` del click): `mostrarSoluciones(false)`.

Eso hace que, cada vez que grabas, se ejecute dos veces la carga de la tabla: una con el valor que eligió el usuario y otra sin retraso. Para que coincida con el enunciado, deja solo una de estas opciones:

- O bien quitas la llamada dentro de `pantalla3Guardar` y solo llamas aquí:

```

setTimeout(() => {
  pantalla3Guardar(); // solo guarda
  mostrarSoluciones(false); // recarga tabla sin retraso tras guardar
  ...
}, 5000);

```

- O bien dejas la llamada en `pantalla3Guardar` y eliminas la del `setTimeout`.

Con una sola llamada se evitarán duplicados y comportamientos raros. [\[61\]](#) [\[62\]](#)

Filas de estado duplicadas

Cada vez que cambias el estado ("", "Guardando...", "OK", "No OK") haces:

```

let guardadoColumna = crearFila();
guardadoColumna.textContent = '...';

```

`crearFila()` añade una fila nueva a la tabla, así que terminas con muchas filas vacías o repetidas solo para mostrar mensajes. Lo habitual sería:

- Crear una sola celda de estado global:

```
let guardadoColumna = crearFila(); // una vez
guardadoColumna.textContent = '';
```

- Y luego, en toda la lógica, **no volver a llamar a** `crearFila()`, solo cambiar el texto:

```
guardadoColumna.textContent = 'Guardando...';
...
guardadoColumna.textContent = 'OK';
...
guardadoColumna.textContent = 'No OK';
```

Eso respeta la idea de la columna “Estado” sin llenar la tabla de filas extra.^[63]

✱✱

Pero es que entiendo que el enunciado pone que al entrar se muestre el listado con cookies, por ejemplo uso botón atrás y luego vuelvo a entrar e inicialmente me cargue el listado. Y luego si pulso en guardar también actualiza y muestra el listado. Por eso lo he puesto en dos sitios diferentes

Tienes razón en lo que quieres hacer, y con lo que llevas escrito se puede conseguir sin problema: la clave es **usar la misma función** `mostrarSoluciones` **en dos momentos distintos**, pero evitando duplicarla en el mismo flujo.

Cuándo debe llamarse según el enunciado

Del enunciado se desprenden dos usos claros de la función de listado:^[64]

- **Al entrar en pantalla 3:** se carga el listado completo desde las cookies, con el parámetro `retraso` según lo que diga el usuario (S/N en el prompt).
- **Después de guardar:** cuando termina el delay de 5 segundos tras pulsar Grabar, se vuelve a mostrar el listado completo, ya con la nueva pregunta.

Por tanto, es totalmente lógico que:

1. Se llame a `mostrarSoluciones(retraso)` una vez al entrar en la pantalla (después del prompt).
2. Se llame de nuevo a `mostrarSoluciones(false)` al final del `setTimeout` que simula el guardado.^[65]

Qué sobra ahora mismo

Lo que te hace “ruido” es que en tu código actual tienes **tres** posibles sitios:

- Dentro de `pantalla3Guardar` → `mostrarSoluciones(retraso);`
- Dentro de `mostrarSoluciones` (cuando `retraso == true`, el `setTimeout` vuelve a llamar a `mostrarSoluciones(false)`, eso está bien porque es para el “Cargando preguntas...”)
- En el `setTimeout` del click de Grabar → `mostrarSoluciones(false);`

Con eso:

- Al entrar en la pantalla: llamas una vez (correcto).
- Al pulsar Grabar: llamas dos veces seguidas (una desde `pantalla3Guardar` y otra desde el `setTimeout`), lo que duplica el refresco. ^[66]

La solución es simple: **mantener los dos momentos que quieres (entrada y post-guardado), pero con una sola llamada por flujo:**

- Al entrar en la pantalla, fuera de cualquier función:

```
let retrasoString = prompt('¿Quieres retraso: S/N?');
let retraso = retrasoString == 'S' || retrasoString == 's';
mostrarSoluciones(retraso); // primer listado al entrar
```

- Dentro de `pantalla3Guardar` quitas la llamada:

```
function pantalla3Guardar() {
  ...
  setCookie(...);
  // aquí ya NO llamas a mostrarSoluciones
}
```

- Y solo refrescas después del delay al pulsar Grabar:

```
setTimeout(() => {
  pantalla3Guardar(); // guarda en cookies
  mostrarSoluciones(false); // recarga listado tras guardar
  ...
}, 5000);
```

Así:

- Se cumple que al entrar se carga el listado con el retraso elegido.
- Cada vez que grabas, tras los 5 segundos, se vuelve a mostrar el listado actualizado.
- No hay llamadas duplicadas ni se ignora la elección del prompt. ^[65]

✱

2. <https://developer.mozilla.org/en-US/docs/Web/API/Document/getElementById>
3. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/parselnt
4. https://jaywood128.github.io/addeventlistener_and_function_scope_in_javascript
5. <https://bhojpress.com/blogs/website-development/understanding-javascript-scoping-issues-in-event-listeners-why-are-variables-undefined>
6. <https://stackoverflow.com/questions/2652319/how-do-you-check-that-a-number-is-nan-in-javascript>
7. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Number/isNaN
8. <https://stackoverflow.com/questions/67803453/generated-row-of-table-event-listener-not-working-instantly>
9. https://www.reddit.com/r/learnjavascript/comments/wn5r1v/dynamic_tables_with_javascript/
10. https://www.w3schools.com/js/js_htmlDOM_eventlistener.asp
11. <https://www.youtube.com/watch?v=ZAJg-FNbcTc>
12. https://www.reddit.com/r/webdev/comments/lu5yu2/javascript_how_to_check_if_a_variable_is_a_nan/
13. https://www.w3schools.com/jsref/prop_radio_disabled.asp
14. https://w3schools.am/jsref/prop_radio_disabled.html
15. <https://stackoverflow.com/questions/40202609/variables-defined-in-addeventlistener-javascript>
16. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/isNaN
17. <https://www.geeksforgeeks.org/javascript/how-to-disable-radio-button-using-javascript/>
18. <https://desarrolloweb.com/articulos/1130.php>
19. https://www.w3schools.com/jsref/met_document_getelementbyid.asp
20. <https://www.geeksforgeeks.org/javascript/how-to-disable-specific-readio-button-option-using-javascript/>
21. <https://stackoverflow.com/questions/50897679/javascript-event-target-id-undefined-in-event-listener>
22. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/Not_defined
23. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/ReferenceError
24. https://udn.realityripple.com/docs/Web/JavaScript/Reference/Errors/Not_defined
25. https://www.w3schools.com/JS/js_function_closures.asp
26. <https://rollbar.com/blog/javascript-referenceerror/>
27. https://www.w3schools.com/js/js_function_closures.asp
28. <https://builtin.com/articles/javascript-closures>
29. <https://stackoverflow.com/questions/14573223/set-cookie-and-get-cookie-with-javascript>
30. <https://stackoverflow.com/questions/48139563/iife-reference-error-variable-not-defined-inside-the-function>
31. <https://www.tabnine.com/academy/javascript/how-to-set-cookies-javascript/>
32. <https://www.tutorialrepublic.com/javascript-tutorial/javascript-cookies.php>
33. <https://www.youtube.com/watch?v=o2iOd3o-4kA>
34. <https://www.geeksforgeeks.org/javascript/how-to-set-retrieve-cookies-using-javascript/>
35. <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Closures>
36. https://reference.codeproject.com/javascript/reference/errors/not_defined

37. <https://codeswithpankaj.com/javascript-cookies-tutorial-for-beginners/>
38. <https://www.geeksforgeeks.org/javascript/closure-in-javascript/>
39. https://www.reddit.com/r/learnprogramming/comments/qzq16f/javascript_how_come_a_variable_defined_within_an/
40. <https://www.geeksforgeeks.org/javascript/javascript-referenceerror-variable-is-not-defined/>
41. <https://stackoverflow.com/questions/17583544/referenceerror-variable-is-not-defined>
42. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/parseln
43. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Number/isNaN
44. <https://stackoverflow.com/questions/2652319/how-do-you-check-that-a-number-is-nan-in-javascript>
45. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/Not_defined
46. <https://stackoverflow.com/questions/17583544/referenceerror-variable-is-not-defined>
47. https://www.reddit.com/r/learnjavascript/comments/wn5r1v/dynamic_tables_with_javascript/
48. https://www.w3schools.com/JS/js_function_closures.asp
49. <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Closures>
50. https://www.w3schools.com/js/js_function_closures.asp
51. <https://builtin.com/articles/javascript-closures>
52. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/Not_defined
53. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/Not_defined
54. <https://stackoverflow.com/questions/17583544/referenceerror-variable-is-not-defined>
55. <https://www.tutorialrepublic.com/javascript-tutorial/javascript-cookies.php>
56. image.jpg
57. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/Not_defined
58. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/ReferenceError
59. https://www.w3schools.com/JS/js_function_closures.asp
60. <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Closures>
61. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Errors/Not_defined
62. <https://stackoverflow.com/questions/17583544/referenceerror-variable-is-not-defined>
63. https://www.reddit.com/r/learnjavascript/comments/wn5r1v/dynamic_tables_with_javascript/
64. <https://www.tutorialrepublic.com/javascript-tutorial/javascript-cookies.php>
65. <https://developer.mozilla.org/en-US/docs/Web/API/Window/setTimeout>
66. <https://stackoverflow.com/questions/27323772/javascript-function-and-ui-updates>
67. <https://shapkarin.me/articles/design/>
68. <https://javascript.plainenglish.io/how-to-prevent-a-function-from-being-called-multiple-times-c8d59b2d637d>
69. <https://stackify.com/solid-design-principles/>
70. <https://www.xenonstack.com/blog/solid-principles-javascript>
71. <https://stackoverflow.com/questions/17670447/multiple-updates-during-one-function-call-in-javascript>
72. <https://www.geeksforgeeks.org/system-design/single-responsibility-in-solid-design-principle/>
73. <https://developer.mozilla.org/en-US/docs/Web/API/Window/setInterval>

- 74. <https://www.ui-dev.in/2021/01/single-responsibility-principle-in.html>
- 75. <https://forums.tumult.com/t/how-to-call-javascript-functions-from-other-functions-multiple-times-in-a-row/14114>
- 76. <https://godotforums.org/d/37212-best-way-to-update-ui-data-function-call-or-in-processdelta>
- 77. <https://react.dev/learn/queueing-a-series-of-state-updates>
- 78. <https://blog.logrocket.com/solid-principles-single-responsibility-in-javascript-frameworks/>
- 79. https://www.reddit.com/r/learnjavascript/comments/ywp6fl/single_responsibility_principle_how_to_organise/
- 80. <https://www.manuelsanchezdev.com/blog/solid-principles-single-responsibility>
- 81. <https://stackoverflow.com/questions/28929694/is-the-single-responsibility-principle-applicable-to-functions>
- 82. <https://dev.to/duncanmcardle/solid-principle-1-single-responsibility-javascript-4ide>
- 83. https://www.reddit.com/r/reactjs/comments/16d4hbx/how_to_call_setstate_multiple_times_inside_a/