Preparación

1. Abrí con el editor de texto el archivo que veníamos usando, `juego.html`. Si estás usando Atom, podés hacer clic derecho sobre el código y seleccionar "Preview HTML" para ver los cambios en tiempo real de tu documento.
2. Usá el archivo `juego.js` en otra pestaña del editor de texto. Este está en la carpeta de recursos descargables que ya deberías tener guardada en tu computadora, Drive, o repositorio.

Paso 1: Definir el Tablero Ganador

El objetivo de este primer paso es completar la función `chequearSiGano()`, que recorra la variable grilla y devuelva `true` para las grillas ganadoras (ordenadas) y `false` para las grillas desordenadas.

Recordá que la variable grilla representa cada pieza del HTML con un número. Esta variable está en orden originalmente, pero la función `mezclarPiezas()` la va a desordenar cuando ejecutemos la función `iniciar()`.

Por ejemplo, esta grilla es ganadora, porque cada pieza está en su posición correcta:

```
[[1, 2, 3],
 [4, 5, 6],
 [7, 8, 9]];
```

Pero la siguiente grilla no es ganadora, porque las fichas 2, 3, 5 y 9 no están en sus posiciones correctas:

```
[[1, 5, 9],
 [4, 2, 6],
```

```
[7, 8, 3]];
```

> Pista: Para poder resolver esto vas a tener que pensar cómo recorrer la grilla elemento por elemento. ¿Bastará con usar un sólo bucle `for` como en un arreglo simple?
>
> ***Buenas prácticas: Es útil plantear primero todas las ideas en papel. Una vez que resuelvas como hacerlo, pasá al código. Lo importante en la programación es el pensamiento que te lleva a resolver un problema, el código es algo secundario necesario para que la computadora nos entienda.***

Paso 2: Alerta Ganadora

No sólo queremos que se gane el juego cuando tenemos la grilla ganadora, sino que también queremos mostrar una alerta que avise que ganamos.

Para eso, hay que implementar la función `mostrarCartelGanador()`. Tenés que hacer que aparezca de alguna forma un cartel demostrando que ganaste cuando la grilla sea la correcta. Podría ser una alerta, o cualquier cosa que se te ocurra. Existen infinitas posibilidades.

> ***Buenas prácticas: Podés usar `console.log()` para hacer un seguimiento del código, mostrando variables y texto, y viendo si algo falla. Para chequear si funciona, desde la consola podés escribir `chequearSiGano()` y debería devolver `true`.***

Recordá que, en este primer momento, la variable grilla ya está ordenada. Por eso, apenas abras tu aplicación (refrescando el documento) y toques cualquier tecla te va a mostrar el mensaje que definiste en la función `mostrarCartelGanador()`.

> ***Buenas prácticas: Podés definir en el HTML y en el CSS el cartel ganador con la fuente, los colores y el tamaño que más te guste. El que damos es solo un ejemplo. ¡Dale rienda suelta a tu imaginación!***

Paso 3: Intercambiar las piezas

Para intercambiar las posiciones de las piezas, tendremos que poder intercambiar posiciones en el arreglo de arreglo `grilla`.

> Sabías qué: Esto es lo que llamamos un `intercambio lógico` y está separado de lo que es la `representación visual` del intercambio (que es el cambio que se realizará en la pantalla). Siempre conviene separar la lógica de lo visual para tener un mayor control sobre los cambios en nuestro proyecto. En este proyecto, la parte del intercambio ya viene implementada ya que es algo que aprenderás a hacer más adelante.

Intercambio Lógico

Para concretar el intercambio, debemos completar la función `intercambiarPosiciones(filaPos1, columnaPos1, filaPos2, columnaPos2)`. Y sólo tendremos que interactuar con la variable `grilla`.

> Pista: Las posiciones de la grilla se definen por su número de fila y número de columna. **Empieza desde la fila cero y columna cero `(0, 0)`, ubicada en la esquina superior izquierda**. Desde ahí los números aumentan hacia la derecha y hacia abajo. La esquina de abajo a la derecha es la posición `(2, 2)`.

Si tenemos la siguiente grilla:

```
[[1, 2, 3],
 [4, 5, 6],
```

```
 [7, 8, 9]];
```

y aplicamos la función `intercambiarPosiciones(1, 1, 1, 2)` obtendremos como resultado la grilla:

```
[[1, 2, 3],
 [4, 6, 5],
 [7, 8, 9]];
```

Paso 3: Definir Movimientos Válidos

En este paso, hay que terminar la función `posicionValida(fila,columna)`. Esta deberá avisar si la pieza puede moverse para donde queremos que se mueva (entra en el tablero de juego) o no puede (se sale del tablero).

A nivel lógico, tendrá que definir si la fila y la columna están dentro de la grilla (devolviendo `true`) o fuera de sus límites (`false`).

¿Qué condición deberían cumplir las posiciones para ser válidas?

> Pista: ¿Se podrá resolver todo en un condicional?

Paso 4: Movimiento de las Piezas con el Teclado

Por último, la función `moverEnDireccion(direccion)` deberá hacer que cuando el/la usuario/a toca una tecla (arriba, abajo, izquierda, derecha), la **pieza vacía** se intercambie con la ficha vecina correspondiente. La función `actualizarPosicionVacia` deberá estar implementada, para que cada vez que se mueva una ficha se actualicen las variables `filaVacia` y `columnaVacia`.

Cada vez que los/as usuarios muevan las flechas del teclado, se va a ejecutar esta función. Para ganar el juego, esta función se ejecutará las veces necesarias hasta

que el tablero esté ordenado, es decir, hasta que `chequearSiGano()` devuelva `true`.

Esta es la parte principal de la lógica del juego. Así que, primero, te recomendamos tomarte un tiempo para pensar cómo la resolverías. Leé y releé el código dado hasta entenderlo e intentá deducir cómo realizar los movimientos faltantes.

Si el movimiento es válido, se deberá agregar al arreglo de movimientos. Para es implementaste una función que hacía este trabajo. ¡No te olvides de invocarla cada vez que hagas un movimiento, así se suma correctamente!

> ***Buenas prácticas: Pensá en papel tu solución. Si estás participando de un programa presencial o estás estudiando con un/a compañero/a, siempre es bueno pensar en grupo. ¡Dos cabezas piensan mejor que una!***

Paso 5: Terminá el Juego

Tenés que completar los datos y funciones faltantes para que el juego funcione correctamente.

> Recomendación: Releé todas las funciones dadas, tanto las completas como las incompletas. Fijate que estén usadas correctamente.

Ejecutá tu juego y evaluá su funcionamiento. ¿Está todo bien? ¿Podés identificar si hay algún error?