Trabajo Final Integrador

Desarrollo y publicación de un videojuego web utilizando Git/Github + HTML + CSS + JS

Videojuego web a desarrollar: SIMON SAYS

Link del juego: https://www.memo-juegos.com/juegos-simon/juego-de-simon-online-gratis

Requerimientos Obligatorios

- Código prolijo y estricto (HTML5, CSS3 y ES5).
- Consistencia en comentarios, commits y estilos de código.
- Responsividad y estética del juego y la web (usando Flexbox).
- Juego completamente funcional para un jugador debiendo ingresar nombre del jugador al iniciar la partida, validando como mínimo 3 letras para el nombre.
- Agregar puntaje por cada luz de color presionada correctamente y mostrar el score.
- Detectar el momento en que no se presiona la secuencia correcta y mostrar un modal indicando que perdió con un botón para reiniciar la partida.
- Crear una página de Contacto, con un formulario que permita ingresar nombre, mail y mensaje, y al enviar se abra la herramienta de envío de emails predeterminada del sistema operativo.
- Validaciones del formulario de contacto (nombre alfanumérico, mail válido y mensaje con más de 5 caracteres).
- Agregar un link a la página de Github donde se alojó el código del juego, que al apretarlo se abra en una nueva pestaña.

Requerimientos Deseados

- Mostrar en pantalla el nivel actual (número de secuencia).
- Poner un contador de tiempo al iniciar cada partida y utilizar el tiempo transcurrido para restar puntos de las partidas ganadas, es decir que el puntaje final de una partida finalizada, debe ser la cantidad de botones presionados correctamente menos la penalización por el tiempo demorado. Elegir la penalización que prefiera.
- Guardar los resultados de cada partida del juego usando LocalStorage, recordando el nombre, puntaje, nivel y la fecha y hora de la partida.
- Agregar un botón para mostrar un popup con la lista de partidas con jugadores, puntajes y fechas. En este ranking debe estar ordenado por puntaje.
- Agregar la opción de ordenar el ranking por fecha o puntaje.

En los puntos anteriores los requerimientos obligatorios son esenciales para aprobar. Los requerimientos deseados son opcionales y suman a la nota, siendo que si cumplen con todos los puntos a la perfección tendrán nota 10 (diez).

Condiciones mínimas para una correcta entrega:

Tipo de nomenclaturas:

- Pascal Case: SomeName
- Camel Case: someName
- Snake Case: some_name
- Kebab Case: some-name

Git:

- Los commits deben contar el paso a paso que se realizó. Si leo los mensajes de los commits entiendo como se construyó el proyecto.
- No hay una cantidad de commits correcta, pero no tiene sentido tener pocos commits.
- No sirve de nada poner comentarios como "Fixes", "Progress", "Changes". Los comentarios deben reflejar el cambio realizado en términos del progreso para el proyecto.
- El comentario del commit debe ser corto y claro.

Archivos y carpetas:

- El repositorio debe tener un Readme claro, con la información necesaria del proyecto y bien formateado.
- Los archivos imágenes, css y js deben estar guardados en las carpetas correspondientes.
- Los nombres de archivos y carpetas deben mantener un estilo de nombre, por ejemplo CamelCase o Kebab Case.
- Utilizar un archivo .gitignore es una buena práctica para evitar subir archivos ocultos del sistema operativo (.vscode, .DS_Store, .thumbnails, etc).

HTML:

- Siempre definir el Doctype, meta tag de viewport, título y charset.
- Siempre agregar un archivo css para normalizar los estilos cross browser (ej. reset.css).
- Nunca usar elementos
 para separar líneas, se puede lograr desde código CSS y es mejor.
- Las referencias a JavaScript pueden ser agregadas al final del elemento <body> o al final del elemento <head> manejando el evento onLoad del DOM.
- Nunca usar estilos en línea.
- Nunca usar javascript en línea.
- Los nombres de ids y de clases deben mantener consistencia, o se utiliza CamelCase, o se utiliza Kebab Case.
- La indentación debe ser perfecta, no mezclar espacios con tabs. SIEMPRE revisar cómo queda el archivo indentado en Github dado que en el IDE de su propia computadora puede parecer diferente. Se evalúa el código en Github, no el de su computadora.
- No dejar líneas en blanco.
- No dejar comentarios innecesarios.

CSS:

- Utilizar siempre flexbox, nada de grid ni float.

- Utilizar siempre el mismo estilo de color, Hexadecimal, rgb o con palabras, pero no una mezcla.
- Utilizar unidades de medidas consistentes, px, rem, %, vw, etc. No es necesario usar siempre la misma unidad pero se debe ser consciente de cuando y por que se usa cada una.
- En un selector compuesto, cuando tiene un selector de ID, todo lo que está antes del # no es tomado en cuenta y debería borrarse.
- Las reglas deben estar ordenadas por selectores, primero los selectores de elementos, luego los de pseudoelementos, luego los selectores de clase, de id, y selectores compuestos, por orden de aparición en el html.
- Los media queries deben ir siempre al final del código.
- Ser muy consciente de las reglas y propiedades que se pisan usando media queries.
- Ordenar las propiedades dentro de las reglas, por ejemplo se pueden ordenar alfabéticamente, o mejor aún si se ordenan así: primero display y position, luego las propiedades de flex (si aplica), luego las propiedades del box model, luego background, luego propiedades de texto, luego fuentes y por último el resto de las propiedades.
- La indentación dentro de las reglas y dentro de los medios queries debe ser perfecta, no mezclar espacios con tabs. SIEMPRE revisar cómo queda el archivo indentado en Github dado que en el IDE de su propia computadora puede parecer diferente. Se evalua el codigo en Github, no el de su computadora.
- No dejar líneas en blanco.
- No dejar comentarios innecesarios

JS:

- Siempre usar código estricto agregando 'use strict' al principio del archivo
- No usar let, const, () => ni ningún otro tipo de sintaxis de ES6 para el final de LPPA.
- No mezclar comillas dobles y comillas simples, o se usa siempre una o siempre la otra.
- Si se pone ; al final de una sentencia, usar ; en todas, o en ninguna.
- Si se crean funciones anónimas y se guardan en variables, hacerlo siempre asi, de lo contrario usar siempre funciones nominadas.
- Para las operaciones de comparación por igualdad usar siempre === y !==
- La sintaxis del if else y de los bucles for debe ser siempre igual.
- Se deben declarar todas las variables globales al principio del archivo, y las locales deben declararse al principio de cada función.
- Manejar todos los eventos desde JavaScript con la propiedad addEventListener de objetos del DOM.
- Evitar crear funciones anónimas directamente como callbacks, es mejor extraer el código de la función fuera del callback y guardarlo en una variable.
- Ordenar las funciones según su aparición, es decir, si una función X dentro de su código llama a la función Y, entonces Y debe ser declarada antes que X.
- Es una buena practica separar el archivo JS en archivos más pequeños, agrupando las funciones según sus características (manejadores de evento, core del juego, inicializadores del DOM, etc).

- Evitar inyectar HTML desde JavaScript, siempre que sea posible tener el HTML ya escrito en el archivo HTML correspondiente, con una clase que le aplique la regla css "display: none;" y luego cambiar la clase desde JavaScript para que aparezca o desaparezca según corresponda.
- La indentación debe ser perfecta, no mezclar espacios con tabs. SIEMPRE revisar como queda el archivo indentado en Github dado que en el IDE de su propia computadora puede parecer diferente. Se evalúa el código en Github, no el de su computadora.
- No dejar líneas en blanco.
- No dejar comentarios innecesarios

General:

- No mezclar código en inglés y en español. Escribir todo el proyecto en español, o todo en inglés.