KINAL
Centro Educativo
Técnico Laboral

Nombre del alumno:	**Jorge Lisandro Magzul Tzuquén**		
Carné:	**2024029**	**Grado y Jornada**	**IN5BM**
Curso:	**TIC.**	**Fecha de entrega:**	**07/09/2025**
Nombre de la tarea:	**Reporte de proyecto Juego de Ahorcado**		

Reporte

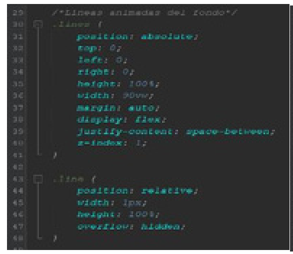
Juego de Ahorcado

Reporte de cómo funciona el código para el proyecto del juego de ahorcado, reporte sobre HTML y JavaScript

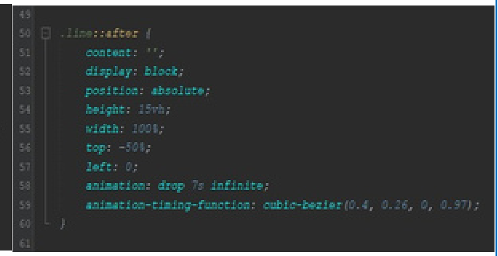
HTML

```
<body>
    <!--Lineas para el fondo-->
    <div class="lines">
        <div class="line"></div>
        <div class="line"></div>
        <div class="line"></div>
        <div class="line"></div>
        <div class="line"></div>
        <div class="line"></div>
        <div class="line"></div>
        <div class="line"></div>
        <div class="line"></div>
        <div class="line"></div>
    </div>
```

Este bloque sirve en mi proyecto como fondo de la web, con esto y código de CSS le da el efecto de como caen unas líneas tipo luces de colores.



```
/*Lineas animadas del fondo*/
.lines {
    position: absolute;
    top: 0;
    left: 0;
    right: 0;
    height: 100%;
    width: 90vw;
    margin: auto;
    display: flex;
    justify-content: space-between;
    z-index: 1;
}

.line {
    position: relative;
    width: 1px;
    height: 100%;
    overflow: hidden;
}
```

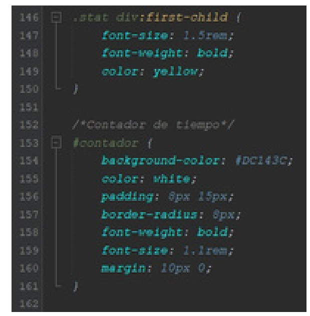


```
.line::after {
    content: '';
    display: block;
    position: absolute;
    height: 15vh;
    width: 100%;
    top: -50%;
    left: 0;
    animation: drop 7s infinite;
    animation-timing-function: cubic-bezier(0.4, 0.26, 0, 0.97);
}
```

```
<div class="container">
    <h1> Juego de Ahorcado</h1>

    <div class="stats">
        <div class="stat">
            <div id="currentWord">1</div>
            <div>Palabra</div>
        </div>
        <div class="stat">
            <div id="errorsCount">0</div>
            <div>Errores</div>
        </div>
        <div class="stat">
            <div id="correctGuesses">0</div>
            <div>Aciertos</div>
        </div>
    </div>

    <div id="contador"> Tiempo 5:00</div>
```

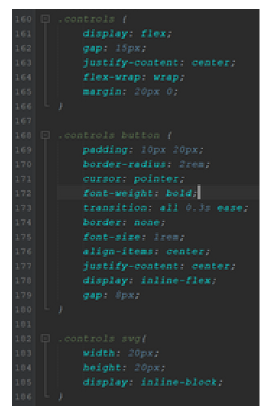
Este bloque muestra el título principal, las estadísticas y el temporizador, donde marca qué palabra es, cuántos errores ha cometido, cuantos aciertos lleva y cuánto tiempo le queda en la partida.

```
/*Estadisticas*/
.stats {
    display: flex;
    justify-content: space-around;
    margin: 20px 0;
    background-color: rgba(255, 255, 255, 0.1);
    padding: 15px;
    border-radius: 10px;
}

.stat {
    text-align: center;
}
```

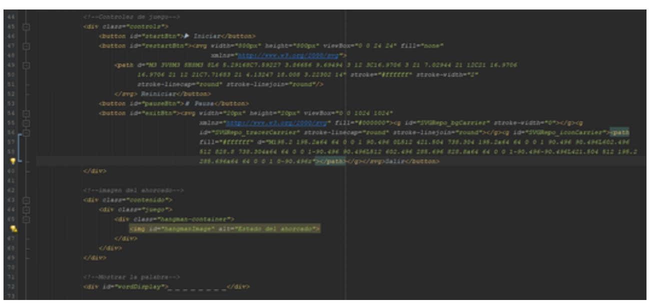


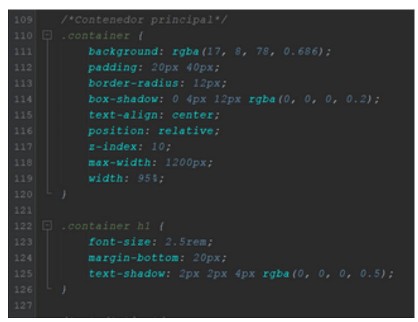
```
.stat div:first-child {
    font-size: 1.5rem;
    font-weight: bold;
    color: yellow;
}

/*Contador de tiempo*/
#contador {
    background-color: #DC143C;
    color: white;
    padding: 8px 15px;
    border-radius: 8px;
    font-weight: bold;
    font-size: 1.1rem;
    margin: 10px 0;
}
```

Este bloque muestra los botones que se usan en el juego y el espacio donde muestra el muñequito, también donde se va poniendo cada letra de la palabra que se está adivinando.

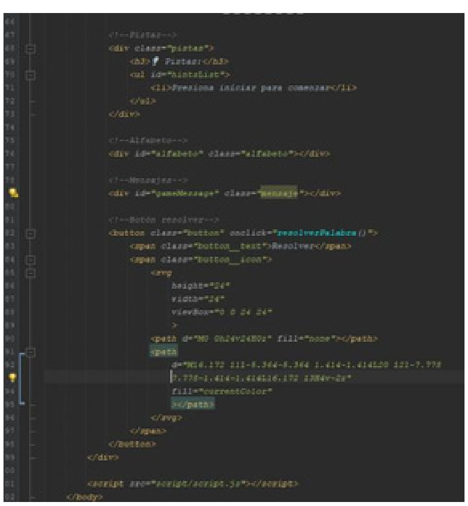


```
.controls {
    display: flex;
    gap: 15px;
    justify-content: center;
    flex-wrap: wrap;
    margin: 20px 0;
}

.controls button {
    padding: 10px 20px;
    border-radius: 2rem;
    cursor: pointer;
    font-weight: bold;
    transition: all 0.3s ease;
    border: none;
    font-size: 1rem;
    align-items: center;
    justify-content: center;
    display: inline-flex;
    gap: 8px;
}

.controls svg{
    width: 20px;
    height: 20px;
    display: inline-block;
}
```



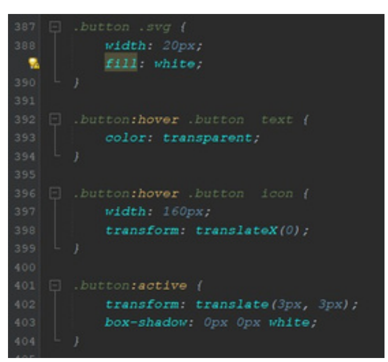


```
/*Contenedor principal*/
.container {
    background: rgba(17, 8, 78, 0.686);
    padding: 20px 40px;
    border-radius: 12px;
    box-shadow: 0 4px 12px rgba(0, 0, 0, 0.2);
    text-align: center;
    position: relative;
    z-index: 10;
    max-width: 1200px;
    width: 95%;
}

.container h1 {
    font-size: 2.5rem;
    margin-bottom: 20px;
    text-shadow: 2px 2px 4px rgba(0, 0, 0, 0.5);
}
```

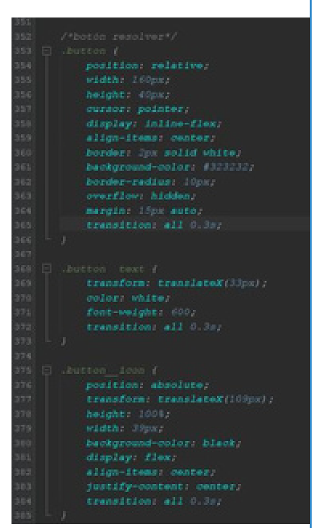


```
<!--Pistas-->
<div class="pistas">
    <h3> Pistas:</h3>
    <ul id="hintsList">
        <li>Presiona iniciar para comenzar</li>
    </ul>
</div>

<!--Alfabeto-->
<div id="alfabeto" class="alfabeto"></div>

<!--Mensajes-->
<div id="gameMessage" class="mensaje"></div>

<!--Boton resolver-->
<button class="button" onclick="resolverPalabra()">
    <span class="button__text">Resolver</span>
    <span class="button__icon">
        <svg
            height="24"
            width="24"
            viewBox="0 0 24 24"
            >
        <path d="M0 0h24v24H0z" fill="none"></path>
        <path
            d="M16.172 11l-5.364-5.364 1.414-1.414L20 12l-7.778
            7.778-1.414-1.414L16.172 13H4v-2z"
            fill="currentColor"
            ></path>
        </svg>
    </span>
</button>
</div>

<script src="script/script.js"></script>
</body>
```

Muestra las pistas del juego como ayuda, define el espacio para el teclado virtual, define también el espacio para los mensajes del juego, un botón de “Resolver” y al final de todo el script.



```
.button .svg {
    width: 20px;
    fill: white;
}

.button:hover .button__text {
    color: transparent;
}

.button:hover .button__icon {
    width: 160px;
    transform: translateX(0);
}

.button:active {
    transform: translate(3px, 3px);
    box-shadow: 0px 0px white;
}
```



```
/*boton resolver*/
.button {
    position: relative;
    width: 160px;
    height: 40px;
    cursor: pointer;
    display: inline-flex;
    align-items: center;
    border: 2px solid white;
    background-color: #323232;
    border-radius: 10px;
    overflow: hidden;
    margin: 15px auto;
    transition: all 0.3s;
}

.button__text {
    transform: translateX(33px);
    color: white;
    font-weight: 600;
    transition: all 0.3s;
}

.button__icon {
    position: absolute;
    transform: translateX(109px);
    height: 100%;
    width: 39px;
    background-color: black;
    display: flex;
    align-items: center;
    justify-content: center;
    transition: all 0.3s;
}
```

JavaScript

Este es un arreglo para que almacene las palabras y las pistas que se cargan desde la base de datos

```
// Arreglo de objetos con palabras del juego con las pistas
let palabras = [];
```

Este bloque define el estado actual del juego, cada vez que el jugador hace algo entonces se actualiza este objeto y actualiza el juego con base a esta información.

```
// Objeto que mantiene el estado actual del juego mientras se interactua
let juego = {
    palabraActual: 0, //palabra que se está ejecutando
    palabra: '', //la palabra actual para adivinar
    palabraAdivinada: [], //array con las letras que se adivinaron
    errores: 0, //contador de letras incorrectas
    aciertos: 0, //contador de letras correctas
    letrasUsadas: [], //array de letras ya usadas
    iniciado: false, //estado si el juego está iniciado
    pausado: false, //estado si el juego está pausado
    tiempoRestante: 300, //tiempo en segundos - 5min
    timer: null //temporizador
};
```

Es donde se ponen todas las imágenes del muñequito para el juego.

```
//Imágenes del ahorcad
const imagenesAhorcado = [
    "img/estado0.png",
    "img/estado1.png",
    "img/estado2.png",
    "img/estado3.png",
    "img/estado4.png",
    "img/estado5.png",
    "img/estado6.png"
];
```

Creamos un objeto llamado elementos que guarda accesos directos a los elementos del HTML usando sus id, para no tener que estar usando document.getElementById('...')" cada vez,haciéndolo más rápido.

```
// Referencias a elementos HTML que se actualizan durante el juego, asignar valores a cada elemento para no escribir lo mismo
const elementos = {
    startBtn: document.getElementById('startBtn'), //boton de iniciar juego
    restartBtn: document.getElementById('restartBtn'), //boton para reiniciar
    pauseBtn: document.getElementById('pauseBtn'), //boton para pausar o reanudar
    exitBtn: document.getElementById('exitBtn'), //boton para salir
    wordDisplay: document.getElementById('wordDisplay'), //donde se muestra la palabra adivinada
    hangmanImage: document.getElementById('hangmanImage'), //la imagen del ahorcado
    hintsList: document.getElementById('hintsList'), //listado de las pistas
    alfabeto: document.getElementById('alfabeto'), //contenedor del teclado (virtual)
    gameMessage: document.getElementById('gameMessage'), //para los mensajes del juego
    currentWord: document.getElementById('currentWord'), //para el numero de la palabra actual del juego
    errorsCount: document.getElementById('errorsCount'), //contar los errores que lleva el jugador
    correctGuessesCount: document.getElementById('correctGuesses'), //contar los aciertos que lleva el jugador
    contador: document.getElementById('contador') //para el temporizador
};
```

Función que carga las palabras desde el servidor y las carga al arreglo de "palabras"

```
function cargarPalabra() {
    fetch('Controlador') //petición al serverlet/controlador
            .then(response => response.json()) //convertir a json
            .then(data => {
                //Convertir de DB al juego
                palabras = data.map(item => ({
                        palabra: item.textoPalabra.toUpperCase(), //para convertir a mayúsculas
                        pistas: [
                            item.pista1,
                            item.pista2,
                            item.pista3
                        ]
                    }));
                console.log('Palabras cargadas de la base de datos:', palabras);
                mostrarMensaje(`${palabras.length} palabras cargadas, suerte intentando resolverlas`, `success`);
            })
            .catch(error => {
                //manejo de errores en caso de que la carga falle
                document.write('Error al cargar palabras', error);
                mostrarMensaje("Error: No se pudieron cargar las palabras de la base de datos", "error");
            });
}
```

Tenemos dos funciones:
Primera función “mostrarMensaje”: muestra mensajes en pantalla de diferentes tipos (Ej. “Correcto”, “Error”, “Juego pausado”, etc...).

Segunda función “limpiarMensaje”: llama a mostrarMensaje() pero sin parámetros, así que borra cualquier mensaje si hay alguno activo, su funcionamiento es limpiar pantalla de los mensajes mostrados cuando ya no sea necesario que estén visibles.

```
68
69    //Mostrar / limpiar mensajes
70    function mostrarMensaje(texto = '', tipo = '') {
71        if (texto) {
72            elementos.gameMessage.textContent = texto;
73            elementos.gameMessage.className = `message ${tipo}`; //aplica estilos según el tipo
74            elementos.gameMessage.style.display = 'block';
75        } else {
76            elementos.gameMessage.style.display = 'none'; //oculta el mensaje
77    }
78    }
79
80    //funcion para limpiar mensaje
81    function limpiarMensaje() {
82        mostrarMensaje();
83    }
84
```

```
84
85    // Inicializar cuando carga la página
86    document.addEventListener('DOMContentLoaded', function () { //Espera a que el html se cargue
87        crearAlfabeto(); //Llama a la funcion que crea el teclado virtual
88        configurarEventos(); //Asigna los eventos de clic a los botones
89        cargarPalabra(); //para cargar la palabra de la base de datos
90        reiniciarJuego(); //inicializar el estado del juego
91        mostrarMensaje("Bienvenidos al ahorcado, presiona iniciar para comenzar", 'info');
92    ));
93
94    // Crear los botones del teclado virtual para cada letra
95    function crearAlfabeto() {
96        const letras = 'ABCDEFGHIJKLMNÑOPQRSTUVWXYZ'.split('');
97        elementos.alfabeto.innerHTML = ''; //limpia el contenedor
98
99        letras.forEach(letra => {
100           const btn = document.createElement('button'); //Crea un boton
101           btn.className = 'letter-btn';
102           btn.textContent = letra;
103           btn.id = `letra-${letra}`; //Para asignar un id
104           btn.onclick = () => adivinarLetra(letra); //Asignar una fnci[on al hacer clic
105           elementos.alfabeto.appendChild(btn); //Agregar al DOM
106       ));
107   }
108
```

document.addEventListener('DOMContentLoaded', function () se ejecuta cuando termina de de cargarse la página (el HTML).

La función crearAlfabeto() generael teclado virtual que aparece en pantalla para que el jugador la use y trate de adivinarla palabra.

```
109   //Configurar los eventos listener del juego
110   function configurarEventos() {
111       //evento para iniciar el juego
112       elementos.startBtn.onclick = iniciarJuego;
113
114       //evento para reiniciar
115       elementos.restartBtn.onclick = () => {
116           if (confirm('Estás seguro de que quieres reiniciar?')) {
117               terminarJuego(); //detener el juego actual
118               reiniciarJuego(); //para volver a empezar desde cero
119               mostrarMensaje('Juego reiniciado, preciona iniciar para comenzar de nuevo', 'info');
120           }
121       );
122
123       elementos.pauseBtn.onclick = alternarPausa; //evento para pausar/reanudar
124       //evento para salir
125       elementos.exitBtn.onclick = () => {
126           if (confirm('Estás seguro de que quieres salir del juego?')) {
127               terminarJuego();
128               mostrarMensaje('Gracias por jugar, recarga la página para jugar otra vez', 'info');
129           }
130       );
131
132       //evento para el teclado fisico
133       document.addEventListener('keydown', (e) => {
134           if (juego.iniciado && !juego.pausado) {
135               const letra = e.key.toUpperCase();
136               //verifica que sea una letra válida incluyengo ñ y acentos
137               if (letra.match(/[A-ZÑÁÉÍÓÚ]/) && letra.length === 1) {
138                   adivinarLetra(letra);
139               }
140           }
141       ));
142   }
```

Función configurarEventos() asigna los eventos a los elementos del juego, al cargar la página crea un teclado virtual y carga las palabras de la base de datos configurando también los botones para iniciar, pausar, reiniciar o salir del juego. El jugador puede usar el teclado virtual o el teclado físico.

También se muestran mensajes según sean las acciones.

```
//Iniciar juego
function iniciarJuego() {
    //para verificar si hay palabras cargadas
    if (palabras.length === 0) {
        mostrarMensaje("No hay palabras disponibles", 'error');
        return;
    }
    //cambia el estado del juego
    juego.iniciado = true;
    juego.pausado = false;

    cargarPalabraActual();//carga la primera palabra
    iniciarTimer(); //inicia el temporizador
    actualizarPantalla(); //actualiza la interfaz
    mostrarMensaje("Juego iniciado, adivina la palabra correcta", 'success');

    elementos.startBtn.style.display = 'none';
    elementos.pauseBtn.style.display = 'inline-block';
}
```

La función iniciarJuego() se ejecuta para dar inicio al juego , carga las palabras, crea/muestra el teclado virtual en el juego, controla el juego con el temporizador y estado(inicio/pausado) e interactúa con el jugador con mensajes.

```
function cargarPalabraActual() {
    //verificar si hay palabras
    if (palabras.length === 0) {
        mostrarMensaje("No hay palabras disponibles", 'error');
        return;
    }

    const dato = palabras[juego.palabraActual];
    juego.palabra = dato.palabra;
    //array de guines bajos de la cantidad de letras de la palabra
    juego.palabraAdivinada = Array(juego.palabra.length).fill('_');
    juego.letrasUsadas = [];
    juego.errores = 0;

    //para mostrar las pistas
    elementos.hintsList.innerHTML = '';
    dato.pistas.forEach(pista => {
        const li = document.createElement('li');
        li.textContent = pista;
        elementos.hintsList.appendChild(li);
    });
    reiniciarAlfabeto(); //reactiva todas las letras
}
```

La función cargarPalabraActual() carga la palabra que se va a adivinar sus pistas, escoge una palabra de la base de datos y la oculta con guiones bajos, muestra las pistas y reinicia el teclado virtual para que el jugador empiece a adivinar la palabra.

```
// Adivinar letra
function adivinarLetra(letra) {
    if (!juego.iniciado || juego.pausado || juego.letrasUsadas.includes(letra)) {
        return;
    }
    juego.letrasUsadas.push(letra);
    const boton = document.getElementById(`letra-${letra}`);
    boton.disabled = true;

    if (juego.palabra.includes(letra)) {
        for (let i = 0; i < juego.palabra.length; i++) {
            if (juego.palabra[i] === letra) {
                juego.palabraAdivinada[i] = letra;
            }
        }
        juego.aciertos++;
        boton.classList.add('correct');
        mostrarMensaje(`¡Bien hecho! "${letra}" está en la palabra.`, 'success');

        if (!juego.palabraAdivinada.includes('_')) {
            setTimeout(() => siguientePalabra(), 1500);
        }
    } else {
        juego.errores++;
        boton.classList.add('wrong');
        mostrarMensaje(`La letra "${letra}" no está.`, 'error');

        if (juego.errores >= 6) {
            setTimeout(() => juegoTerminado(), 1000);
        }
    }

    actualizarPantalla();
}
```

La función adivinarLetra() se encarga de verificar si la letra seleccionada está en la palabra que se adivina, si se acierta entonces se muestra en su posición y aumenta la cantidad de aciertos, pero si falla cuenta el error y dependiendo del resultado pasa a la siguiente palabra o finaliza el juego. (La letra si es error se marca en rojo y si es
acierto se marca de color verde.)

```
//Pasar a la siguiente palabra
function siguientePalabra() {
    if (juego.palabraActual < palabras.length - 1) {
        juego.palabraActual++;
        juego.tiempoRestante = 300;
        cargarPalabraActual();
        actualizarPantalla();
        mostrarMensaje(`Correcto, siguiente palabra: ${juego.palabraActual + 1}/${palabras.length}`, 'success');
    } else {
        mostrarMensaje('Felicidades, completaste todas las palabras');
        terminarJuego();
    }
}
```

La función siguientePalabra() se activa cuando adivina una palabra completa, revisa si hay más palabras en la lista y si sí, la pasa a la siguiente palabra, se reinicia el tiempo y se actualiza todo, si ya no quedan más palabras entonces felicita al jugador y cierra el juego.

```
function actualizarPantalla() {
    elementos.wordDisplay.textContent = juego.palabraAdivinada.join(' ');
    elementos.currentWord.textContent = juego.palabraActual + 1;
    elementos.errorsCount.textContent = juego.errores;
    elementos.correctGuessesCount.textContent = juego.aciertos;
    elementos.hangmanImage.src = imagenesAhorcado[juego.errores] || imagenesAhorcado[6];
}
```

La función actualizarPantalla() actualiza toda la información de la pantalla mientras se juega como los aciertos, los errores, el número de palabra y la imagen del ahorcado que cambia por cada error.

```
// Reiniciar juego
function reiniciarJuego() {
    if (juego.timer) {
        clearInterval(juego.timer);
    }
    juego = {
        palabraActual: 0,
        palabra: '',
        palabraAdivinada: [],
        errores: 0,
        aciertos: 0,
        letrasUsadas: [],
        iniciado: false,
        pausado: false,
        tiempoRestante: 300,
        timer: null
    };

    reiniciarAlfabeto();
    elementos.wordDisplay.textContent = '_ _ _ _ _ _ _ _';
    elementos.hintsList.innerHTML = '<li>Presiona Iniciar para comenzar</li>';
    elementos.hangmanImage.src = imagenesAhorcado[0];
    elementos.contador.textContent = 'Tiempo 5:00';
    limpiarMensaje();

    elementos.startBtn.style.display = 'inline-block';
    elementos.pauseBtn.style.display = 'none';
}
```

La función reiniciarJuego() reinicia el juego, lo cual lo deja como recién ejecutado. Limpia todo el progreso, resetea el cronómetro, la palabra, los errores y oculta/muestra los botones necesarios para que el jugador pueda volver a empezar desde cero.

```
function reiniciarAlfabeto() {
    const botones = document.querySelectorAll('.letter-btn');
    botones.forEach(btn => {
        btn.disabled = false;
        btn.className = 'letter-btn';
    });
}

//Temporizador
function iniciarTimer() {
    actualizarContador();
    juego.timer = setInterval(() => {
        if (!juego.pausado) {
            juego.tiempoRestante--;
            actualizarContador();
            if (juego.tiempoRestante <= 0) {
                clearInterval(juego.timer);
                mostrarMensaje("⏰ ¡Se acabó el tiempo!", 'error');
                setTimeout(() => juegoTerminado(), 1000);
            }
        }
    }, 1000);
}

function actualizarContador() {
    let min = Math.floor(juego.tiempoRestante / 60);
    let seg = juego.tiempoRestante % 60;
    if (seg < 10) {
        seg = '0' + seg;
    }
    elementos.contador.textContent = `Tiempo ${min}:${seg}`;
}
```

La función reiniciarAlfabeto() reinicia todo el teclado virtual, habilita todos los botones y deja el teclado limpio. Selecciona todos los botones con .letter-btn, le quita el estado deshabilitado (disabled = false) y le vuelve a asignar su CSS original.

La función iniciarTimer() inicia un temporizador que cuenta regresivamente el tiempo restante del juego. Llama a actualizarContador() para mostrar el tiempo inicial. Usa setInterval() para restar un segundo cada 1000ms (1 segundo), verifica que el juego no esté en pausa y si el tiempo llega a 0 detiene el temporizador, muestra el mensaje de que se acabó el tiempo y termina el juego.

La función actualizarContador() actualiza el temporizador en pantalla mostrando los minutos y segundos que quedan. Convierte los segundos restantes (tiempoRestante) a minutos y segundos, asegura que los segundos se muestren con dos dígitos (05) y actualiza el contenido de elementos.contador.

```
//terminar el juegop
function juegoTerminado() {
    mostrarMensaje(`juego terminado, la palabra era: "${juego.palabra}"`, 'error');
    setTimeout(() => reiniciarJuego(), 3000);
}
```

La función juegoTerminado() finaliza el juego cuando el jugador pierde ya sea por erroreso porque se termina el tiempo, muestra un mensaje en la pantalla indicando que el juego se terminó y muestra cuál es lapalabra correcta, espera 3 segundos (los 3000ms de “setTimeout(() => reiniciarJuego(), 3000);”) y reinicia el juego con reiniciarJuego().

```
293
294     //Pausar / reanudar
295     function alternarPausa() {
296         if (!juego.iniciado) {
297             return;
298         }
299         juego.pausado = !juego.pausado;
300
301         if (juego.pausado) {
302             elementos.pauseBtn.textContent = '▶ Reanudar';
303             mostrarMensaje('Juego en pausa.', 'info');
304         } else {
305             elementos.pauseBtn.textContent = '⏸ Pausa';
306             mostrarMensaje('Juego reanudado.', 'success');
307         }
308     }
309
```

La función alternarPausa() permite pausar o reanudar la partida al hacer clic en el botón que depende del estado.
Si está pausado cambia el texto del botón a "▶️ Reanudar" y muestra el mensaje de que el juego está en pausa. Si está activo el texto del botón es " Pausa" y muestra mensaje de que el juego está reanudado.

```
310     // Terminar
311     function terminarJuego() {
312         juego.iniciado = false;
313         clearInterval(juego.timer);
314
315         document.querySelectorAll('.letter-btn').forEach(btn => btn.disabled = true);
316         elementos.startBtn.style.display = 'inline-block';
317         elementos.pauseBtn.style.display = 'none';
318     }
319
```

La función terminarJuego() detiene el juego, desactiva el teclado y reinicia los botones. Marca el juego como no iniciado, detiene el temporizador, desactiva los botones del teclado virtual y actualiza los botones de nuevo. Esto funciona cuando presiona el botón de Salir

```
351
352     function resolverPalabra() {
353         if (!juego.iniciado || juego.pausado) {
354             return;
355         }//Revelar todas las letras de la palabra
356         juego.palabraAdivinada = juego.palabra.split('');// Marcar todos los aciertos según el largo de la palabra
357         juego.aciertos = juego.palabra.length; //prar desactivar todas las letras del teclado
358
359         document.querySelectorAll('.letter-btn').forEach(btn => btn.disabled = true);
360         actualizarPantalla();
361         mostrarMensaje(`¡Palabra resuelta! Era "${juego.palabra}".`, 'success');
362         setTimeout(() => siguientePalabra(), 1500); //para pasar a la siguiente palabra después de 1.5 segundos
363     }
364
```

La función resolverPalabra() resuelve automáticamente la palabra actual del juego si el jugador se rinde y presiona el botón "Resolver"

Enlace de trello: [Clic aquí]

Respaldo:
https://trello.com/invite/b/68b5d22b6f6dd0e682184bfd/ATTI1cef868819d848250854ee38ec5db648AEEC8BAD/ahorcados-2024029