

Manual Técnico del Ahorcado



Preparado por : **YOUSSEF MALKI**
Enmarcado por: **JOAN MELSION**

Año Escolar: **2023-2024**

Descripción general

El código proporcionado implementa un juego guardado en HTML, CSS y Javascript. El juego incluye funciones como palabras particulares, seguimiento de vidas y errores, muestra un cronómetro y proporciona mensajes para ganar o perder.

Estructura del código

HTML

Encabezado: Contiene el título del juego.

Principal: Contiene la imagen del Ahorcado que se cambia automáticamente a función de las vidas que tienes mientras jugando, el botón adivinar palabra para las palabras individuales, el espacio para mostrar las palabras de los individuos, los mensajes de resultados, las letras del instructor a seleccionar, el cronómetro y la información de vida y errores.

Pie de página: Contiene información del nombre del diseñador del juego.

— CODE HTML COMPLETO —

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,
initial-scale=1.0">
  <title>Juego Ahorcado</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <header>
    <h1>Ahorcado</h1>
  </header>

  <main>
    
    <div>
      <button type="button" id="jugar">Adivinar palabra</button>
```

```
<button type="button" id="reiniciar"> Jugar Otra
Vez</button>

<p id="adivinar">
</p>
<p id="resultado"></p>
<p id="timer"></p>
<div class="letras">
  <div class="row">
    <div class="col">A</div>
    <div class="col">B</div>
    <div class="col">C</div>
    <div class="col">D</div>
    <div class="col">E</div>
    <div class="col">F</div>
    <div class="col">G</div>
    <div class="col">H</div>
  </div>
  <div class="row">
    <div class="col">I</div>
    <div class="col">J</div>
    <div class="col">K</div>
    <div class="col">L</div>
    <div class="col">M</div>
    <div class="col">N</div>
    <div class="col">Ñ</div>
    <div class="col">O</div>
  </div>
  <div class="row">
    <div class="col">P</div>
    <div class="col">Q</div>
    <div class="col">R</div>
    <div class="col">S</div>
    <div class="col">T</div>
    <div class="col">U</div>
    <div class="col">V</div>
    <div class="col">W</div>
  </div>
  <div class="row">
    <div class="col">X</div>
    <div class="col">Y</div>
    <div class="col">Z</div>
  </div>
  <div class="row" id="rowCronometro">
```

```

        <div class="cronometro" id="minutos">0</div>
        <div class="cronometro" id="segundos">0</div>
        <div class="cronometro" id="milisegundos">0</div>
    </div>
</div>

    <div >
        <p class="vidas">Vidas restantes: <span
id="vidas">7</span>Errores: <span id="errores">0</span></p>

    </div>
</div>
</div>

</main>

<script src="script2.js"></script>

<footer>
    <div class="foot">
        <p>Designed By Youssef Malki</p>
    </div>
</footer>
</body>
</html>

```

CSS:

El archivo **CSS** proporciona estilos para la presentación visual del juego.

— CODE CSS COMPLETO —

```

@import
url('https://fonts.googleapis.com/css2?family=Agbalumo&family=Lato&family=Montserrat:ital,wght@0,100;0,200;0,400;0,500;0,600;0,700;0,800;0,900;1,100;1,200;1,300;1,400;1,500;1,600;1,700;1,800;1,900&family=Roboto:ital,wght@0,100;0,500;0,700;0,900;1,100;1,300;1,400;1,500;1,700;1,900&family=Ubuntu&display=swap');

```

```
*{
    box-sizing: border-box;
    margin: 0;
    padding: 0;
}

body {
    font-family: 'Ubuntu', sans-serif;
    background: url(imagenes/img1.jpg) no-repeat center;
    background-size: cover;
    min-height: 100vh; /*para que obtendra todo el tamaño de la pagina*/
}

header{
    background-color: black;
    padding: 10px;
    text-align: center;
    margin-bottom: 20px;
}

h1{
    margin: 0;
    color: white;
}

main{
    position: fixed;
    width: 80%;
    height: 100%;
    margin: 15px auto;
    padding: 20px;
    background: rgba(255, 255, 255, .7);
    display: flex;
    gap: 30px;
    border: 2px solid black;
    border-radius: 5px;
    position: relative;
}

#imagen{
    border: 2px solid black;
    background-color: antiquewhite;
    padding: 10px;
    width: 300px;
    height: 400px;
}
```

```
    margin: 15px;
    padding: 10px;
    border-radius: 5px;
}
.letras{
    border: 2px solid black;
    border-radius: 5px;
    margin: 10px;
    padding: 5px;
    width: 800px;
}
#adivinar span{
    font-size: 2em;
    border-bottom: 2px solid black;
    width: 30px;
    text-align: center;
    margin-right: 6px;
    display: inline-block; /*es ne*/
}

#reiniciar{
    background: black;
    color: white;
    width: 95px;
    height: 45px;
    border-radius: 5px;
}
#parr{
    margin: 15px;
    font-size: 18px;
    color: rgb(255, 136, 1);
}

#vidas{
    color: crimson;
    padding: 5px;
}
#errores{
    color: rgb(160, 95, 103); padding: 5px;
}
.row{
    display: flex;
```

```
        justify-content: center;
        align-items: center;
    }
.col{
    background-color: rgb(179, 198, 198);
    color: black;
    width: 55px;
    height: 55px;
    margin: 5px;
    padding: 5px;
    text-align: center;
    font-size: 18px;
    font-weight: bold;
    border-radius: 5px;
    border: 3px solid white;
}
```

```
.cronometro{
    background-color: rgb(179, 198, 198);
    color: black;
    width: 55px;
    height: 55px;
    margin: 5px;
    padding: 5px;
    text-align: center;
    font-size: 18px;
    font-weight: bold;
    border-radius: 5px;
    border: 3px solid white;
}
```

```
#jugar{
    background: black;
    color: white;
    width: 95px;
    height: 45px;
    border-radius: 5px;
    padding: 5px;
    margin: 5px;
    border-radius: 5px;
}
```

```
button:hover{
```

```
background: white ;
color: black;

}

.foot{
  position: fixed;
  margin-top: 15px ;
  bottom: 0;
  width: 100%;
  height: 50px;
  background-color:black;
}

.foot p{
  color: white;
  padding: 15px;
}
```

JAVASCRIPT

Variables globales btnJugar, reiniciarBtn: Referencias a los botones "Jugar" y "Reiniciar".

palabras: Una tabla de palabras para adivinar.

vidas, errores: Contadores de vidas y errores.

palabraActual: Almacena la palabra actual que debe adivinarse.

juegoGanado: Indica si el juego ya ha sido ganado.

intervalo: Almacen el intervalo correspondiente al reloj.

tiempoAgotadoPerdido, tiempoAgotadoGanado: Indica si queda tiempo para perder o ganar.

Deberes principales inicio (evento): Inicie el juego haciendo clic en el botón “**Adivinar palabra**”. tenga espacios para letras y configure eventos de clic en letras.

verificarLetra(evento): Comprueba si la letra seleccionada está en el discurso actual. Actualice la GUI y el manejo de datos y errores.

actualizarPalabraAdivinada(letra): Actualiza la palabra adivinada con la letra correcta y comprueba si ha divinizado la palabra completa.

mostrarMensajePerdido(): Muestra un mensaje de pérdida con colores rojos. Puede indicar una pérdida durante un tiempo transcurrido.

mostrarMensajeGanador(): Muestra un mensaje de Victoria con colores verdes. Puedo indicar Victoria por el tiempo transcurrido.

reiniciarJuego(): Reinicia el juego restaurando los vídeos, errores y palabras actuales.

get_random(num_min, num_max): genera un número aleatorio dentro de un rango.

id(str): Función de utilidad para obtener elementos DOM por ID.

iniciarCronometro(): Inicia el cronómetro utilizado en setInterval.

holdCronometro(): Mantén presionado el cronómetro.

updateCronometro(): Actualiza el cronómetro y muestra mensajes de pérdida si se acaba el tiempo.

actualizarElementosCronometro(): Actualiza los elementos del cronómetro en el DOM.

Eventos del Cronometro Se inicia al comenzar el juego y detener el cronómetro al perder o al ganar .

— CODE JSCOMPLETO —

```
document.addEventListener('DOMContentLoaded', function () {
  const btnJugar = id('jugar');
  const reiniciarBtn = id('jugar'); // Cambiado a 'jugar'
  let palabras = ["mysql", "java", "php", "css", "programacion",
    "javascript", "servidor"];
  let vidas = 7;
  let errores = 0;
  let errores_imagenes = 6;
  let palabraActual = "";
  let juegoGanado = false;
  let intervalo;
  let tiempoAgotadoPerdido = false;
  let tiempoAgotadoGanado = false;
  let letrasAcertadas ;

  // Evento de clic en el botón para comenzar el juego
  btnJugar.addEventListener('click', iniciar);
  //bouton jugar de nuevo
```

```

    let reiniciar = document.getElementById('reiniciar');
    reiniciar.addEventListener('click', () => {

        location.reload(); // Reiniciar el juego
    });

// Evento de clic en el botón para reiniciar el juego
reiniciarBtn.addEventListener('click', reiniciarJuego);

// Función para iniciar el juego
function iniciar(evento) {
    // Restablecer la imagen del ahorcado
    id('imagen').src = 'imagenes/ahorcado_6.png';
    juegoGanado = false; // Restablecer el estado del juego ganado
    detenerCronometro(); // Detener el cronómetro al reiniciar el
juego

    //reiniciarBtn.disabled = true; // Deshabilitar el botón de
reinicio

    const parrafo = id('adivinar');
    parrafo.innerHTML = '';

    const cant_palabras = palabras.length;
    const valor_al_azar = obtener_random(0, cant_palabras);
    palabraActual = palabras[valor_al_azar].toLowerCase();
    console.log(palabraActual);

    const cant_letras = palabraActual.length;

    // Crear espacios para cada letra de la palabra
    for (let i = 0; i < cant_letras; i++) {
        const span = document.createElement('span');
        parrafo.appendChild(span);
    }

    // Limpiar el color de fondo de las letras
    const cols = document.querySelectorAll('.col');
    cols.forEach(col => col.style.backgroundColor = "");

    // Quitar y agregar el evento click a las letras
    cols.forEach(col => col.removeEventListener('click',
verificarLetra));

```

```

        cols.forEach(col => col.addEventListener('click',
verificarLetra));

        iniciarCronometro(); // Iniciar el cronómetro
        //
    }

    // Función para verificar la letra seleccionada
    function verificarLetra(event) {

        if (!juegoGanado && vidas > 0) {
            const letra = event.target.innerText.toLowerCase();

            if (palabraActual.includes(letra)) {
                event.target.style.backgroundColor = "green";
                actualizarPalabraAdivinada(letra);
            } else {
                event.target.style.backgroundColor = "red";
                vidas -= 1;
                id("vidas").innerText = vidas;
                errores += 1;
                id("errores").innerText = errores;

                // Cambiar la imagen del ahorcado
                id('imagen').src =
`imagenes/ahorcado_${errores_imagenes}.png`;
                errores_imagenes -= 1;

                if (vidas === 0) {
                    // Quitar eventos click al perder
                    const cols = document.querySelectorAll('.col');
                    cols.forEach(col =>
col.removeEventListener('click', verificarLetra));
                    mostrarMensajePerdido();
                    reiniciarBtn.disabled = false; // Habilitar el
botón de reinicio
                }
            }
        }

        if(vidas == 0 || ([...spans].every(span => span.innerText
!== ' ')) || (minutos < 0 && !tiempoAgotado)){
            clearInterval(intervalo);
        }
    }

```

```

    }
    // Función para actualizar la palabra adivinada
    function actualizarPalabraAdivinada(letra) {
        const spans = id('adivinar').querySelectorAll('span');
        for (let i = 0; i < palabraActual.length; i++) {
            if (palabraActual[i] === letra) {

                spans[i].innerText = letra;
                letrasAcertadas++;

            }
        }
        if(letrasAcertadas == palabraActual.length) {
            detenerCronometro();

        }

        // Verificar si se adivinó toda la palabra
        if ([...spans].every(span => span.innerText !== '')) {
            juegoGanado = true; // Marcar el juego como ganado
            mostrarMensajeGanador();
            detenerCronometro(); // Detener el cronómetro al ganar
            reiniciarBtn.disabled = false; // Habilitar el botón de
reinicio

            reiniciarJuego();

        }
    }

    // Función para detener el cronómetro
    function detenerCronometro() {
        clearInterval(intervalo);
    }

    // En la función mostrarMensajePerdido()
function mostrarMensajePerdido() {
    const mensajePerdido = document.getElementById('resultado');
    mensajePerdido.style.color = 'red';

    mensajePerdido.innerText = tiempoAgotadoPerdido ? "¡Tiempo agotado!
Has perdido." : "¡Has perdido! Inténtalo de nuevo.!"

```

```

        setTimeout(() => {
            mensajePerdido.remove();
        }, 3000);
    }

// En la función mostrarMensajeGanador()
function mostrarMensajeGanador() {
    const mensajeGanador = document.getElementById('resultado');
    mensajeGanador.style.color = 'green';
    mensajeGanador.innerHTML = tiempoAgotadoGanado ? "¡Has ganado!  
¡Bien hecho!" : "¡Has ganado! Inténtalo de nuevo si quieres.!!";

    setTimeout(() => {
        mensajeGanador.remove();
    }, 3000);
}

// Función para reiniciar el juego
function reiniciarJuego() {
    vidas = 7;
    errores = 0;
    id("vidas").innerHTML = vidas;
    id("errores").innerHTML = errores;

    // Limpiar el color de fondo de las letras al reiniciar
    const cols = document.querySelectorAll('.col');
    cols.forEach(col => col.style.backgroundColor = "");
}

// Función para obtener un número aleatorio dentro de un rango
function obtener_random(num_min, num_max) {
    const amplitud_valores = num_max - num_min;
    return Math.floor(Math.random() * amplitud_valores + num_min);
}

// Función de utilidad para obtener elementos del DOM por ID
function id(str) {
    return document.getElementById(str);
}

```

```
}

const rowCronometro = document.getElementById('rowCronometro');
const minutosElement = document.getElementById('minutos');
const segundosElement = document.getElementById('segundos');
const milisegundosElement =
document.getElementById('milisegundos');

let minutos = 1;
let segundos = 0;
let milisegundos = 0;

// Iniciar el cronómetro
function iniciarCronometro() {
    intervalo = setInterval(actualizarCronometro, 1000);
}

// Detener el cronómetro
function detenerCronometro() {
    clearInterval(intervalo);
}

// Actualizar el cronómetro
function actualizarCronometro() {
    milisegundos -= 100;

    if (milisegundos < 0) {
        milisegundos = 900;
        segundos -= 1;
    }

    if (segundos < 0) {
        segundos = 59;
        minutos -= 1;
    }

    if (minutos < 0 && !tiempoAgotado) {
        detenerCronometro();
        tiempoAgotado = true;
        mostrarMensajePerdido();
        return;
    }
}
```

```

        actualizarElementosCronometro();
    }

    // Actualizar los elementos del cronómetro en el DOM
    function actualizarElementosCronometro() {
        minutosElement.textContent = minutos < 10 ? '0' + minutos :
minutos;
        segundosElement.textContent = segundos < 10 ? '0' + segundos :
segundos;
        milisegundosElement.textContent = milisegundos < 10 ? '00' +
milisegundos : milisegundos < 100 ? '0' + milisegundos : milisegundos;
    }

    // Evento de clic en el botón para iniciar el cronómetro
    rowCronometro.addEventListener('click', function () {
        iniciarCronometro();
    });

    // Evento de clic en el botón para detener el cronómetro
    rowCronometro.addEventListener('dblclick', function () {
        detenerCronometro();
    });

    // Iniciar el juego al cargar la página
    iniciar();
});

```

Que disfrutéis del juego!!!!

GRACIAS