

Informe: Proyecto del Ahorcado

JSP – Ahorcado:

1. Creación

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
```

Acá lo que hacemos con el @page es decirle al servidor que tipo de contenido se generara, el contentType="text/html" definimos que el documento es html, y el pageEncoding... permite caracteres especiales como ñ, á, é, etc.

2. Head

```
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Juego del Ahorcado</title>
  <link rel="icon" type="image/x-icon" href="${pageContext.request.contextPath}/Images/Icono.png">
  <link rel="stylesheet" href="${pageContext.request.contextPath}/Styles/ahorcado.css">
</head>
```

En este apartado lo que hacemos es definir el titulo de la pagina como “Juego del Ahorcado”, también definimos el icono que tendrá la pagina junto con el estilo, utilizo el \${pageContext.request.contextPath} para encontrar bien las rutas ya que se necesita el contexto, es decir de donde vienen los paquetes que estoy mandando a llamar.

3. Body

Iniciamos el contenido visible de la pagina, seguido del contenedor que tiene el juego.

```
<div class="seccion-principal">
  <!-- Lado izquierdo: Imagen del ahorcado -->
  <div class="seccion-ahorcado">
    <div class="dibujo-ahorcado" id="ahorcado">

    </div>
  </div>

  <!-- Lado derecho: Temporizador e imagen progresiva -->
  <div class="panel-derecho">
    <!-- Temporizador -->
    <div class="seccion-temporizador">
      <div class="pantalla-tiempo" id="tiempo">02:00</div>
    </div>

    <!-- Imagen que se mostrará progresivamente -->
    <div class="imagen-progresiva">
      <div class="contenedor-imagen" id="imagenProgreso">

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

Acá en la sección principal, lo que hacemos es distribuir cierta parte del juego desde el dibujo de la persona que se ahorca, en la columna derecha esta colocado el cronometro, y debajo contiene la imagen que devuelve cuando el usuario ingresa correctamente su palabra.

Comentario: Los divs vacíos están vacíos porque se llenaran con js, los id importantes son “ahorcado, tiempo e imagenProgreso” ya que los usaremos en el js.

```
<!-- Sección inferior con pista y botones -->
<div class="seccion-inferior">
  <!-- Pista (debajo del ahorcado) -->
  <div class="seccion-pista">
    <div class="titulo-pista">Adivinanza</div>
    <div class="texto-pista" id="pista">Adivina la palabra</div>
  </div>

  <!-- Botones de control -->
  <div class="botones-control">
    <button class="boton" onclick="empezar()">
      <span class="sombra"></span>
      <span class="borde"></span>
      <span class="frente"><span>Empezar</span></span>
    </button>
    <button class="boton" onclick="reiniciarJuego()">
      <span class="sombra"></span>
      <span class="borde"></span>
      <span class="frente"><span>Reiniciar</span></span>
    </button>
    <button class="boton" onclick="pausarJuego()">
      <span class="sombra"></span>
      <span class="borde"></span>
      <span class="frente"><span>Pausar</span></span>
    </button>
    <button class="boton" onclick="salirJuego()">
      <span class="sombra"></span>
      <span class="borde"></span>
      <span class="frente"><span>Salir</span></span>
    </button>
  </div>
</div>
```

Continuamos con la sección inferior, donde colocamos lo que serian los botones del juego y la pista que se le da al usuario, el id importante acá es “pista”, acá utilizo buttons para las acciones de los métodos en mi js, pero también hago uso del pues me permite una mejor animación con css para mis botones, también con el onclick, se ejecutan las funciones de empezar, reiniciarJuego, etc.

```

<!-- Área del juego con palabra y teclado -->


<!-- Información del juego -->
  <div class="informacion-juego">
    <div class="contador-errores" id="errores">Errores: 0/6</div>
  </div>

  <!-- líneas para la palabra -->
  <div class="lineas-palabra">
    <div class="espacios-letras" id="mostrarPalabra"> _ _ _ _ _ </div>
  </div>

  <!-- Teclado -->
  <div class="teclado" id="teclado">
    <!-- Primera fila -->
    <div class="fila-teclado">
      <button class="tecla" onclick="adivinarLetra('Q')">Q</button>
      <button class="tecla" onclick="adivinarLetra('W')">W</button>
      <button class="tecla" onclick="adivinarLetra('E')">E</button>
      <button class="tecla" onclick="adivinarLetra('R')">R</button>
      <button class="tecla" onclick="adivinarLetra('T')">T</button>
      <button class="tecla" onclick="adivinarLetra('Y')">Y</button>
      <button class="tecla" onclick="adivinarLetra('U')">U</button>
      <button class="tecla" onclick="adivinarLetra('I')">I</button>
      <button class="tecla" onclick="adivinarLetra('O')">O</button>
      <button class="tecla" onclick="adivinarLetra('P')">P</button>
    </div>

    <!-- Segunda fila -->
    <div class="fila-teclado">
      <button class="tecla" onclick="adivinarLetra('L')">L</button>
      <button class="tecla" onclick="adivinarLetra('S')">S</button>
      <button class="tecla" onclick="adivinarLetra('D')">D</button>
      <button class="tecla" onclick="adivinarLetra('F')">F</button>
      <button class="tecla" onclick="adivinarLetra('G')">G</button>
      <button class="tecla" onclick="adivinarLetra('H')">H</button>
      <button class="tecla" onclick="adivinarLetra('J')">J</button>
      <button class="tecla" onclick="adivinarLetra('K')">K</button>
      <button class="tecla" onclick="adivinarLetra('U')">U</button>
      <button class="tecla" onclick="adivinarLetra('V')">V</button>
    </div>

    <!-- Tercera fila -->
    <div class="fila-teclado">
      <button class="tecla" onclick="adivinarLetra('N')">N</button>
      <button class="tecla" onclick="adivinarLetra('M')">M</button>
      <button class="tecla" onclick="adivinarLetra('C')">C</button>
      <button class="tecla" onclick="adivinarLetra('V')">V</button>
      <button class="tecla" onclick="adivinarLetra('B')">B</button>
      <button class="tecla" onclick="adivinarLetra('H')">H</button>
      <button class="tecla" onclick="adivinarLetra('N')">N</button>
    </div>
  </div>


```

Acá encontramos el área de juego, donde colocamos la cantidad de errores que llevamos, la palabra que esta quemada en mi js y el teclado, el contador de errores pues es lo que me muestra el conteo de los errores de usuario, ese div de espacios de letra es donde se van colocando las letras para mostrar las palabras, los id importantes aca son “errores y mostrarPalabra”.

Ahora bien, respecto al teclado lo que hacemos es definir con el onclick la letra en mayuscula como botón, es decir si el usuario preciona esta tecla, se tomara el valor de esa letra, se usaron 3 filas para distribuir bien las letras en el teclado, contiene la 27 letras del alfabeto, utilizan la función adivinarLetra.

```

<!-- Modal para fin de juego -->
<div class="modal" id="modalJuego">
  <div class="contenido-modal">
    <h2 id="tituloModal">Felicitaciones!</h2>
    <p id="mensajeModal">Has ganado el juego y lograste salvar a Jorgito</p>
    <button class="boton boton-reiniciar" onclick="reiniciarJuego(); cerrarModal();">
      <span class="sombra"></span>
      <span class="borde"></span>
      <span class="frente">Jugar de nuevo</span></span>
    </button>
  </div>
</div>

```

Llegando casi al final viene el modal lo que hacemos es devolver un mensaje de que logro completar el juego pero de una forma más llamativa en forma de ventana emergente ya que se superpone al juego, el titulo y mensaje cambian dependiendo si gana o pierde, permite cerrar el modal, los id importantes son “modalJuego, tituloModalm, mensajeModal”.

```

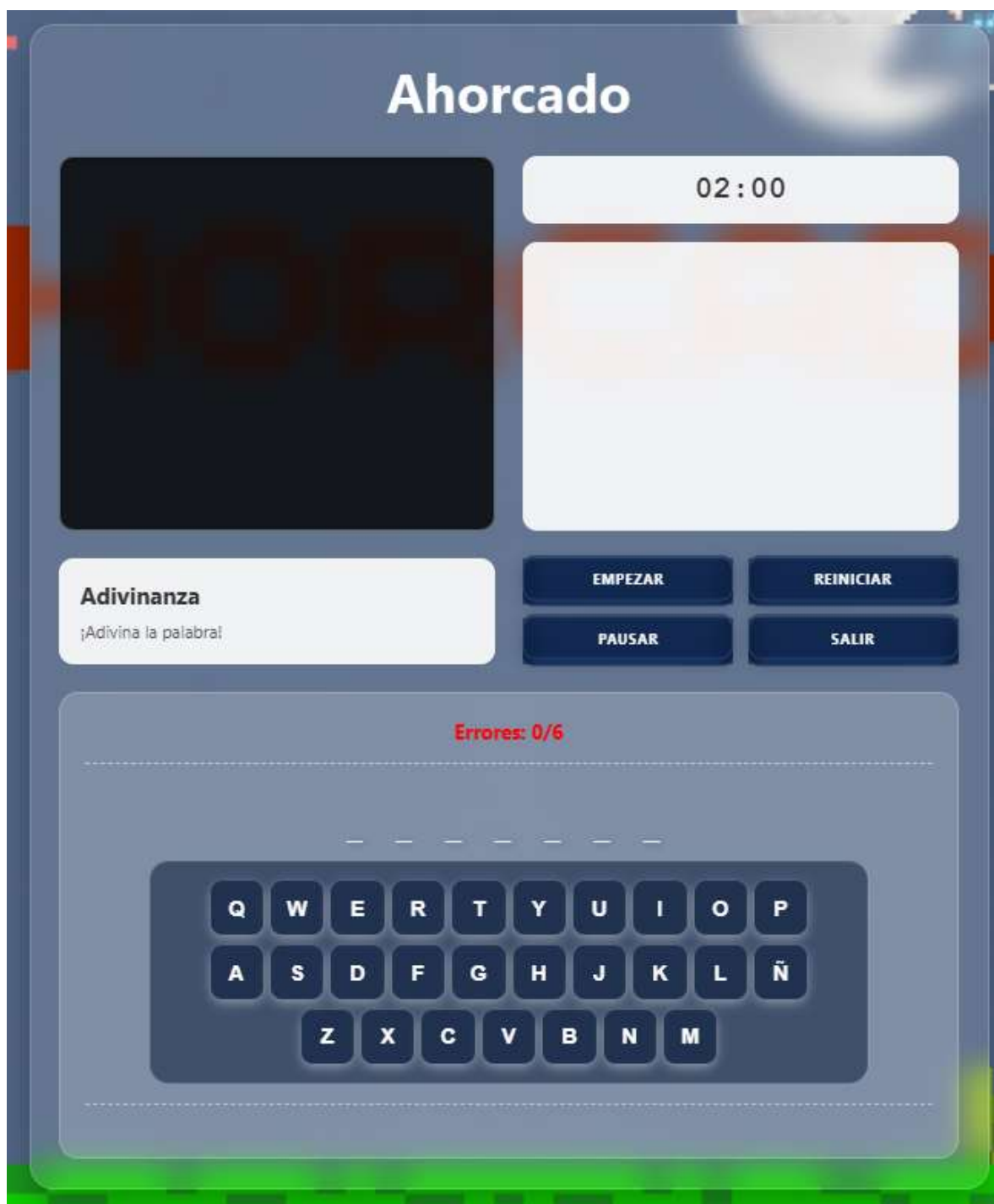
<script>
  const contextPath = '${pageContext.request.contextPath}';
</script>
<script src="${pageContext.request.contextPath}/Scripts/script.js"></script>

```

Por ultimo viene la utilización de las etiquetas script las cuales las utilizo para definir una variable llamada contextPath la cual será mi variable que proporcionara el contexto a las direcciones, luego se carga mi archivo js para poder llamar todas sus funcionalidades.

Elemento HTML	ID	Propósito
Cronómetro	tiempo	Mostrar tiempo restante
Pista	pista	Mostrar adivinanza
Palabra	mostrarPalabra	Mostrar palabra con guiones
Errores	errores	Contador de errores
Ahorcado	ahorcado	Imagen del ahorcado
Progreso	imagenProgreso	Imagen cuando ganas
Modal	modalJuego	Ventana de fin de juego
Título Modal	tituloModal	Título del resultado
Mensaje Modal	mensajeModal	Mensaje del resultado
Teclado	teclado	Contenedor del teclado

Resultados del JSP con CSS:



Script – Ahorcado:

1. Declaración de variables globales

```
// Cronómetro para el juego de Ahorcado
let cronometro = null;
const tiempoMaximo = 120; // 2 minutos
let tiempo = tiempoMaximo;
let juegoActivo = false;
let palabraSecreta = "";
let palabraAdivinada = [];
let errores = 0;
const maxErrores = 6;
let letrasUsadas = [];
```

Donde:

- Cronometro: Almacena la referencia del intervalo del cronometro.
- tiempoMaximo: es constante que define la duración máxima del juego (2min)
- tiempo: que es la variable que cuenta de forma regresiva desde los 2min de tiempoMaximo.
- juegoActivo: es un booleano que indica si el juego esta en curso.
- palabraSecreta: String que contiene la palabra a adivnar.
- palabraAdivinada: Es el array que representa la palabra con las letras adivinadas(escritas por el usuario) o espacios vacíos.
- errores: es el contador de los errores cometidos.
- maxErrores: Es constante del número máximo de errores permitidos(6).
- letrasUsadas: Es el array que almacena las letras ya intentadas.

2. Referencias a elementos html

```
// Obtener elementos del HTML
let cronometroElem = document.getElementById("tiempo");
let pistaElem = document.getElementById("pista");
let mostrarPalabraElem = document.getElementById("mostrarPalabra");
let erroresElem = document.getElementById("errores");
let ahorcadoElem = document.getElementById("ahorcado");
let imagenProgresoElem = document.getElementById("imagenProgreso");
let modalElem = document.getElementById("modalJuego");
let tituloModalElem = document.getElementById("tituloModal");
let mensajeModalElem = document.getElementById("mensajeModal");
let tecladoElem = document.getElementById("teclado");
```

Lo que hacemos es crear nuevas variables las cuales almacenaran el valor de los elementos html, con DOM el cual obtiene el valor a través del id de los elementos.

3. Bacos de palabras con pistas

```
// Banco de palabras con pistas
const palabrasConPistas = [
  {palabra: "MURCIELAGO", pista: "Este animal es pequeño, su nombre lleva todas las vocales."},
  {palabra: "COCOCHILLO", pista: "Este animal es grande, verde, de dientes fuertes."},
  {palabra: "PATINETA", pista: "Tengo 4 ruedas, lija y tornillos."},
  {palabra: "MICROFONO", pista: "Me usan cuando quieren ser escuchados."},
  {palabra: "RONFECABEZAS", pista: "Soy un dolor de cabeza para quienes no tienen paciencia."}
];
```

Acá lo que hacemos es crear un array de objetos, dichos objetos son la palabra con su pista correspondiente, cada objeto tiene dos propiedades, palabra y pista.

4. Función que devuelve un mensaje por default hasta que sea remplazada por la imagen correspondiente a la palabra

```
// Función para mostrar el mensaje inicial
function mostrarMensajeInicial() {
  imagenProgresoElem.innerHTML = `<p id="mensajeInicial" style="text-align: center; color: black !important;
  Si se quisiera ver, correctamente debes responder.
  </p>`;
}
```


5. Funcion principal del cronometro

```
// Función principal del cronómetro
function actualizarCronometro(reset = false) {
  if (reset) {
    clearInterval(cronometro);
    cronometro = null;
    tiempo = tiempoMaximo;
    cronometroElem.textContent = "02:00";
    return;
  }

  if (cronometro)
    return; // Evita múltiples cronómetros

  cronometro = setInterval(() => {
    tiempo--;
    const min = String(Math.floor(tiempo / 60)).padStart(2, "0");
    const seg = String(tiempo % 60).padStart(2, "0");
    cronometroElem.textContent = `${min}:${seg}`;

    if (tiempo <= 0) {
      clearInterval(cronometro);
      cronometro = null;
      tiempoAgotado(); // Función que maneja cuando se acaba el tiempo
    }
  }, 1000);
}
```

Acá lo que hacemos es que con el reset ya que es true, limpia el cronometro existente y restaura valores iniciales.

Luego con el setInterval lo que hacemos es ejecutar una función cada 1000ms (1s), decimo que el tiempo es – porque va en descenso, utilizamos el `Math.floor(tiempo/60)` el cual calcula los minutos, usamos `tiempo % 60` para calcular los segundos y el `padStart(2, "0")` asegura el formato de 2 dígitos.

6. Tiempo Agotado

```
// Función para cuando se agota el tiempo
function tiempoAgotado() {
  juegoActivo = false;
  mostrarModal("¡Tiempo agotado!", "Se acabó el tiempo. Jorgito no pudo ser salvado.");
  deshabilitarTeclado();
}
```

Cuando el tiempo se acabe el `juegoActivo` pasara a falso porque entonces ya no se estará jugando, mostraremos que perdió en el modal y deshabilitaremos el teclado.

7. Empezar

```
// Función para empezar el juego (conecta con el botón "Empezar")
function empezar() {
    if (!juegoActivo) {
        inicializarJuego();
        actualizarCronometro();
        juegoActivo = true;
        habilitarTeclado();
        // CAMBIO: Restaurar el mensaje inicial en lugar de dejar vacío
        mostrarMensajeInicial();
    }
}
```

Mi función empezar es iniciar el juego solo si juegoActivo no esta activo, si no esta activo entonces lo activa y comienza a correr el tiempo, se muestra la pista y la primera imagen del ahorcado, se habilita el teclado y se muestra el mensaje por default que será remplazado por la imagen, con las diferentes funciones que se observan.

8. Reiniciar

```
// Función para reiniciar el juego (conecta con el botón "Reiniciar")
function reiniciarJuego() {
    actualizarCronometro(true); // Reset del cronómetro
    juegoActivo = false;
    errores = 0;
    letrasUsadas = [];
    palabraAdivinada = [];

    // Resetear interfaz
    erroresElem.textContent = "Errores: 0/6";
    pistaElem.textContent = "¡Adivina la palabra!";
    mostrarPalabraElem.textContent = "_ _ _ _ _";
    ahorcadoElem.innerHTML = "";
    imagenProgresoElem.innerHTML = ``;

    // Habilitar todas las teclas
    habilitarTodasLasTeclas();
    cerrarModal();
}
```

Acá se reiniciara el juego desde el cronometro vuelve a 2min, juego activo pasa a falso porque ya no se estaría jugando, lo errores vuelven a ser 0, el

array de las letras usadas otra vez esta vacío al igual el de palabra adivinada, se setean los mensajes con las diferentes funciones puestas ahí.

9. Pausa

```
// Función para pausar el juego (conecta con el botón "Pausar")
function pausarJuego() {
    if (cronometro) {
        clearInterval(cronometro);
        cronometro = null;
        deshabilitarTeclado();
    } else if (juegoActivo) {
        // Si está pausado, reanudar
        actualizarCronometro();
        habilitarTeclado();
    }
}
```

Con esta función suspendemos el cronómetro, deshabilitamos el teclado a menos que el juegoActivo este desactivado ya que acá le quita la pausa y le da el cronómetro nuevamente habilitándole el teclado.

10.

```
// Función para salir del juego (conecta con el botón "Salir")
function salirJuego() {
    actualizarCronometro(true); // Borrar del cronómetro
    juegoActivo = false;
    reiniciarJuego();
    window.location.href = 'ControladorJuegoPrincipal';
}

// Función para inicializar el juego
function inicializarJuego() {
    // Seleccionar palabra aleatoria
    const palabraSeleccionada = palabrasConPistas[Math.floor(Math.random() * palabrasConPistas.length)];
    palabraSecreta = palabraSeleccionada.palabra;
    pistaFila.textContent = palabraSeleccionada.pista;

    // Inicializar array de palabra adivinada
    palabraAdivinada = Array(palabraSecreta.length).fill('_');
    actualizarPalabraMostrada();

    // Borrar errores
    errores = 0;
    erroresFila.textContent = "Errores: 0/6";

    // Mostrar imagen por defecto del ahorcado
    actualizarDibujoAhorcado();
}
```

Con la de salirJuego solo actualizo el cronometro, cambio el estado del juego, reiniciamos todos los datos y mandamos a la vista principal.

En el inicializarJuego(), usamos const palabraSeleccionada, el cual tendra el valor de algún objeto del array de palabras con pista, usamos el Math.random() para que toque una palabra al azar, usamos Math.floor() Ya que redondea hacia abajo para obtener un índice valido y Array(palabraSecreta.length.fill) crea array de n elementos, todos con valor '_'.

11.

```
// Función para adivinar una letra (conecta con las teclas del teclado)
function adivinarLetra(letra) {
  if (!juegoActivo || letrasUsadas.includes(letra)) {
    return;
  }

  letrasUsadas.push(letra);
  deshabilitarTecla(letra);

  if (palabraSecreta.includes(letra)) {
    // Letra correcta
    for (let i = 0; i < palabraSecreta.length; i++) {
      if (palabraSecreta[i] === letra) {
        palabraAdivinada[i] = letra;
      }
    }
    actualizarPalabraMostrada();
    actualizarImagenProgreso();

    // Verificar si ganó
    if (palabraAdivinada.includes('_')) {
      juegoActivo = false;
      clearInterval(cronometro);
      mostrarModal("¡Felicitaciones!", "Has ganado el juego y lograste salvar a Jorgito");
    }
  } else {
    // Letra incorrecta
    errores++;
    erroresElem.textContent = `Errores: ${errores}/6`;
    actualizarDibujoAhorcado();

    // Verificar si perdió
    if (errores >= maxErrores) {
      juegoActivo = false;
      clearInterval(cronometro);
      mostrarModal("¡Perdiste!", `Se acabaron los intentos. La palabra era: ${palabraSecreta}`);
      deshabilitarTeclado();
    }
  }
}
```

Adivinar letra, el inicio del if, letrasUsadas.push(letra);
deshabilitarTecla(letra); con esto verificamos que el juego este activo y que la letra no se haya usado antes.

Con el if de palabra secreta.includes(letra), lo que hacemos es buscar todas las posiciones donde aparece la letra en la palabra y las revela.

En el else errores++;...

Incrementamos los errores y verificamos si se alcanzó el límite máximo.

12.

```
// Función para actualizar la palabra mostrada
function actualizarPalabraMostrada() {
    mostrarPalabraElem.textContent = palabraAdivinada.join(' ');
}
```

Con esta función lo que hacemos es actualizar la palabra mostrada convirtiendo el array en String separado por espacios.

13.

```
// Función para actualizar el dibujo del ahorcado con imágenes
function actualizarDibujoAhorcado() {
  // Array de nombres de las imágenes del ahorcado
  const imagenesAhorcado = [
    "0.png", // Foto por default
    "1.png", // Error 1
    "2.png", // Error 2
    "3.png", // Error 3
    "4.png", // Error 4
    "Cinco.png", // Error 5
    "6.png" // Error 6 (juego terminado)
  ];

  // Limpiar el contenido anterior
  ahorcadoElem.innerHTML = "";

  // Crear el elemento imagen
  const img = document.createElement("img");
  img.className = "imagen-ahorcado";

  // Si hay errores, mostrar la imagen correspondiente al error
  if (errores > 0 && errores <= 6) {
    img.src = `${contextPath}/Images/${imagenesAhorcado[errores]}`;
    img.alt = `Ahorcado - Error ${errores}`;
  } else {
    // Si no hay errores, mostrar la imagen por defecto
    img.src = `${contextPath}/Images/${imagenesAhorcado[0]}`;
    img.alt = "Ahorcado - Estado inicial";
  }

  // Agregar la imagen al contenedor
  ahorcadoElem.appendChild(img);
}
```

En esta función lo que hacemos es asignar la imagen del muñeco que se ira ahorcando por cada error que suma la imagen ira cambiando, creando dinámicamente elementos seleccionando la imagen según el número de errores.

Se creo el array con las imágenes a mostrar, en el img.src se agrego la variable contextPath declarada en el html.

14. Aca lo que hacemos es que cuando la persona ingresa la palabraSecreta entra en un case cuando encuentre su caso, dará su imagen correspondiente la cual se mostrara en el apartado que le corresponde.

```
// Función para actualizar la imagen progresiva (cuando resuelve completamente la palabra)
function actualizarImagenProgreso() {
  // Limpiar el contenido anterior del div imagenProgreso
  imagenProgresoElem.innerHTML = "";

  // Solo mostrar imagen si la palabra está completamente resuelta
  if (!palabraAdivinada.includes('_')) {
    // Crear el elemento imagen
    const img = document.createElement("img");
    img.className = "imagen-progreso";

    // Determinar qué imagen mostrar según la palabra secreta
    let nombreImagen = "";
    switch (palabraSecreta) {
      case "MURCIELAGO":
        nombreImagen = "Murcielago.jpg";
        break;
      case "COCODRILO":
        nombreImagen = "Cocodrilo.jpg";
        break;
      case "PATINETA":
        nombreImagen = "Patineta.jpg";
        break;
      case "MICROFONO":
        nombreImagen = "Microfono.jpg";
        break;
      case "ROMPECABEZAS":
        nombreImagen = "Rompecabezas.jpg";
        break;
      default:
        nombreImagen = "default.jpg"; // Por si agregas más palabras
    }

    img.src = `${contextPath}/Images/${nombreImagen}`;
    img.alt = `Imagen de ${palabraSecreta}`;

    // Agregar la imagen al contenedor
    imagenProgresoElem.appendChild(img);
  }
}
```

15.

```
// Funciones para manejar el teclado
function deshabilitarTecla(letra) {
    const teclas = document.querySelectorAll('.tecla');
    teclas.forEach(tecla => {
        if (tecla.textContent === letra) {
            tecla.disabled = true;
            tecla.classList.add('deshabilitada');
        }
    });
}

function deshabilitarTeclado() {
    const teclas = document.querySelectorAll('.tecla');
    teclas.forEach(tecla => {
        tecla.disabled = true;
    });
}

function habilitarTeclado() {
    const teclas = document.querySelectorAll('.tecla');
    teclas.forEach(tecla => {
        if (!letrasUsadas.includes(tecla.textContent)) {
            tecla.disabled = false;
        }
    });
}

function habilitarTodasLasTeclas() {
    const teclas = document.querySelectorAll('.tecla');
    teclas.forEach(tecla => {
        tecla.disabled = false;
        tecla.classList.remove('deshabilitada');
    });
}
```

En estas funciones se maneja todo lo que es habilitar y deshabilitaar, se usa `querySelectorAll` porque selecciona todos los elementos con clase 'tecla',

forEach para iterar sobre cada elemento y el classList.add para añadir un poco de Css.

16.

```
// Funciones para el modal
function mostrarModal(titulo, mensaje) {
    tituloModalElem.textContent = titulo;
    mensajeModalElem.textContent = mensaje;
    modalElem.style.display = 'flex';
}

function cerrarModal() {
    modalElem.style.display = 'none';
}

// Event listeners adicionales
document.addEventListener('keydown', function (event) {
    if (juegoActivo) {
        const letra = event.key.toUpperCase();
        if (letra.match(/[A-ZÑ]/) && letra.length === 1) {
            adivinarLetra(letra);
        }
    }
});

// Cerrar modal al hacer clic fuera de él
modalElem.addEventListener('click', function (event) {
    if (event.target === modalElem) {
        cerrarModal();
    }
});

// Inicializar el juego al cargar la página
document.addEventListener('DOMContentLoaded', function () {
    reiniciarJuego();
});
```

En los últimos métodos tenemos lo que sería el modelo acá pues solo se settean los datos para poder manejar diferente información.

Ahora el Event Listener se usa evento.key ya que obtiene la tecla presionada esto para que las personas puedan escribir directamente desde el teclado

Usamos match(/[A-ZÑ]/): Expresión regular que valida letras del alfabeto español y usamos length === 1 para asegurar que solo sea una letra.

También DOMContentLoaded, Se ejecuta cuando el HTML ha sido completamente cargado y parseado.

CONCEPTOS TÉCNICOS IMPORTANTES

Scope de Variables: Variables globales vs locales

Manipulación del DOM: getElementById, querySelector, innerHTML

Event Handling: addEventListener, eventos de teclado y mouse

Timers: setInterval, clearInterval

Arrays y Strings: métodos como includes, join, fill

Template Literals: Uso de backticks para strings con variables

Arrow Functions: Sintaxis moderna de JavaScript

Regular Expressions: Para validar entrada de letras

FLUJO GENERAL DEL JUEGO

Usuario presiona "Empezar" → empezar()

Se inicializa juego → inicializarJuego()

Se inicia cronómetro → actualizarCronometro()

Usuario adivina letra → adivinarLetra()

Se actualiza interfaz visual

Se verifica condición de victoria/derrota

Se muestra modal con resultado final