

Informe de Evidencia 03

Tema: JavaScript

Nota

Estudiante	Escuela	Asignatura
Yoel CCorihuaman Guillen yccorihuamang@ulasalle.edu.pe	Carrera Profesional de Ingeniería de Software	Carrera Profesional de Ingeniería de Software Semestre: VII Código: 3.7.2.21

Laboratorio	Tema	Duración
03	JavaScript	06 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2024 - I	Del 4 Abril 2024	Al 29 Abril 2024

1. Tarea

1.1. Descripción

Programar en JavaScript sobre una página web HTML básica.

Listing 1: Estructura básica HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Document</title>
</head>
<body>
<!-- no html, generate it with javascript -->
<script src="script_ejercicio_01.js"></script>
</body>
</html>
```

1.2. Descripción

Programar en JavaScript sobre una página web HTML básica. A continuación, se presenta el código del formulario de banca, junto con una descripción paso a paso de su funcionamiento.

Listing 2: Estructura básica HTML para el formulario de banca

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <title>Formulario de Ingreso</title>
  <link rel="stylesheet" href="styles.css">
  <link rel="stylesheet"
    href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.3/css/all.min.css">
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
</head>
<body>
  <form id="loginForm" style="position: relative; padding-top: 200px; margin-left: auto;
    margin-right: auto; width: 300px;">
    <label for="docType">Tipo de Documento:</label>
    <select id="docType" name="docType">
      <option value="DNI">D.N.I</option>
      <option value="CE">C.E.</option>
      <option value="CPP">C.P.P</option>
    </select>

    <label for="docNumber">Número de Documento:</label>
    <input type="text" id="docNumber" name="docNumber" required pattern="[0-9]{8}"
      title="Solo se permiten 8 dígitos numéricos">

    <label for="cardNumber">Número de Tarjeta (16 dígitos):</label>
    <input type="text" id="cardNumber" name="cardNumber" required pattern="[0-9]{16}"
      title="Solo se permiten 16 dígitos numéricos">

    <label for="password">Clave de Internet:</label>
    <input type="password" id="password" name="password" readonly>
    <table id="keypad" class="table_teclado"></table>

    <label for="captcha">Ingrese el texto de la imagen:</label>
    <input type="text" id="captchaInput" name="captcha" required placeholder="Ingrese
      CAPTCHA" maxlength="5">
    <div id="captchaDisplay" class="captcha-style">CAPTCHA</div>
    <button type="button" id="reloadCaptcha">Recargar CAPTCHA</button>

    <button type="submit">Ingresar</button>
  </form>

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

Listing 3: Código JavaScript para interactividad del formulario

```
// Document ready function ensures the code only runs after the document is fully loaded.
$(document).ready(function() {
  generateKeypad(); // Call to function that generates a dynamic numeric keypad.
  updateDocNumberInput(); // Updates the maximum length of the document number based on
    document type.
  reloadCaptcha(); // Generates a new CAPTCHA when the page loads.
```

```
// Event handler for changing document type.
$('#docType').change(updateDocNumberInput);

// Function to create a dynamic keypad with shuffled numbers.
function generateKeypad() {
  const keys = ['1', '2', '3', '4', '5', '6', '7', '8', '9', '0'];
  keys.sort(() => Math.random() - 0.5); // Shuffle the keys array randomly.
  let keyboardHtml = '';
  keys.forEach((key, index) => {
    if (index % 3 === 0) keyboardHtml += '<tr>';
    keyboardHtml += '<td class="keypad-button">${key}</td>';
    if ((index + 1) % 3 === 0) keyboardHtml += '</tr>';
  });
  keyboardHtml += '<tr><td class="keypad-button"><i class="fas fa-backspace"></i></td><td colspan="2" class="keypad-button"><i class="fas fa-sync-alt"></i></td></tr>';
  $('#keypad').html(keyboardHtml); // Insert the HTML into the keypad container.
  attachKeypadHandlers(); // Attach event handlers for keypad buttons.
}

// Function to handle keypad button clicks.
function attachKeypadHandlers() {
  $('#keypad td').click(function() {
    var number = $(this).text().trim(); // Get the number or icon from the button clicked.
    var currentVal = $('#password').val(); // Current value in the password input.
    if ($(this).find('.fa-backspace').length > 0) {
      $('#password').val(currentVal.slice(0, -1)); // Remove the last character.
    } else if ($(this).find('.fa-sync-alt').length > 0) {
      $('#password').val(''); // Clear the password input.
      generateKeypad(); // Regenerate the keypad.
    } else {
      if (currentVal.length < 6) {
        $('#password').val(currentVal + number); // Add the clicked number to the password.
      }
    }
  });
}

// Updates the maximum length attribute of document number input based on selected document type.
function updateDocNumberInput() {
  var docType = $('#docType').val(); // Get the current value of the document type selector.
  var maxLen = (docType === 'DNI') ? 8 : 12; // Assume DNI = 8 digits, CE = 12 digits.
  $('#docNumber').attr('maxlength', maxLen); // Set the maxlength attribute for document number.
  $('#cardNumber').attr('maxlength', 16); // Ensure card number is always 16 digits.
}

// Function to generate a new CAPTCHA and display it.
function reloadCaptcha() {
  var possibleChars = "ABCDEFGHJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789";
  var captcha = "";
  for (var i = 0; i < 5; i++) {
```

```
        captcha += possibleChars.charAt(Math.floor(Math.random() * possibleChars.length));
    }
    $('#captchaDisplay').css('opacity', '0'); // Temporarily hide the CAPTCHA for a
        fade-in effect.
    setTimeout(function() {
        $('#captchaDisplay').text(captcha).css('opacity', '1'); // Update and show the new
            CAPTCHA.
        sessionStorage.setItem('captcha', captcha); // Store the CAPTCHA in session
            storage.
    }, 500);
}

// Event handler for CAPTCHA reload button.
$('#reloadCaptcha').click(reloadCaptcha);

// Form submission event handler.
$('#loginForm').submit(function(event) {
    event.preventDefault(); // Prevent the form from submitting normally.
    if ($('#captchaInput').val() !== sessionStorage.getItem('captcha')) {
        $('#captchaInput').val(''); // Clear the CAPTCHA input if incorrect.
        alert("CAPTCHA incorrecto. Por favor, intente de nuevo."); // Show an error
            message.
        reloadCaptcha(); // Reload the CAPTCHA.
    } else {
        alert("Formulario enviado. Verificar los datos antes de enviar al servidor."); //
            Success message.
    }
});

// Real-time validation for document and card number inputs.
$('#docNumber, #cardNumber').on('input', function() {
    this.value = this.value.replace(/\D/g, ''); // Remove any character that is not a
        digit.
});
});
```

Listing 4: Hoja de estilos CSS para la estética del formulario

```
body {
    font-family: 'Arial', sans-serif; // Default font for the page.
    padding: 20px; // Adds padding around the content.
    background-color: #f4f4f4; // Light grey background color.
    background-image:
        url('https://www.toptal.com/designers/subtlepatterns/patterns/memphis-mini.png'); //
        Adds a subtle background pattern.
    background-size: cover; // Ensure the background covers the entire content area.
    background-repeat: no-repeat; // No repeating of the background image.
    background-attachment: fixed; // Background scrolls with the page.
    background-position: center; // Center the background image.
    animation: BackgroundGradient 15s ease infinite; // Apply a gradient animation over 15
        seconds.
}

@keyframes BackgroundGradient {
    0% {background-position: 0% 50%;}
    50% {background-position: 100% 50%;}
```

```
100% {background-position: 0% 50%;}
}

button {
  transition: background-color 0.3s, transform 0.2s; // Smooth transition for hover effects.
  background-color: #007BFF; // Primary button color.
  color: white; // Text color for buttons.
  border: none; // No border for a modern look.
  padding: 10px 20px; // Padding inside the button.
  cursor: pointer; // Cursor changes to pointer to indicate clickable.
  border-radius: 5px; // Rounded corners for buttons.
  box-shadow: 0 2px 4px rgba(0, 0, 0, 0.2); // Shadow for 3D effect.
}

button:hover {
  background-color: #0056b3; // Darker blue on hover.
  transform: scale(1.05); // Slightly increase size on hover.
}

button:active {
  transform: scale(0.95); // Button appears pressed on click.
}

input, select {
  padding: 8px; // Padding inside input and select elements.
  width: 100%; // Full width to fit container.
  margin-bottom: 10px; // Space below each input/select.
  box-sizing: border-box; // Border and padding included in width/height.
  border: 2px solid #ccc; // Grey border for inputs.
  border-radius: 5px; // Rounded corners for inputs.
  transition: border-color 0.3s, box-shadow 0.3s; // Smooth transitions for focus effects.
}

input:focus, select:focus {
  outline: none; // Removes default focus outline.
  border: 2px solid #007BFF; // Blue border on focus.
  box-shadow: 0 0 8px blue; // Glowing effect on focus.
}

.captcha-style {
  font-family: 'Courier New', Courier, monospace; // Monospace font for CAPTCHA.
  color: #333; // Dark grey text color.
  background: #eee; // Light grey background for CAPTCHA.
  padding: 10px; // Padding around CAPTCHA text.
  font-size: 18px; // Larger text for visibility.
  letter-spacing: 3px; // Space between CAPTCHA characters.
  display: inline-block; // Allows block level properties.
  border: 1px solid #ccc; // Border around CAPTCHA.
  transition: opacity 0.5s; // Fade effect for CAPTCHA changes.
}

.table_teclado td {
  border: 1px solid #ccc; // Border around each keypad button.
  text-align: center; // Center text in keypad buttons.
  width: 50px; // Fixed width for uniform buttons.
  height: 50px; // Fixed height for square buttons.
}
```

```
display: table-cell; // Display as table cell for alignment.
vertical-align: middle; // Center content vertically.
cursor: pointer; // Cursor changes to pointer on hover.
background-color: #fff; // White background for buttons.
box-shadow: 0 2px 3px #ccc; // Shadow for 3D button effect.
border-radius: 3px; // Rounded corners for buttons.
}

.table_teclado td:active {
  background-color: #ddd; // Darker background on click.
  transform: scale(0.9); // Button shrinks to simulate press.
}
```

1.3. Ejercicios

1. Cree un teclado aleatorio para banca por internet. El teclado aleatorio para la entrada de la contraseña se genera en el formulario de ingreso mediante JavaScript. La función `generateKeypad()` es responsable de este proceso. A continuación, se describen los pasos involucrados en la creación del teclado:
 - a) **Inicialización de la Matriz de Teclas:** Se crea un arreglo con los números del 0 al 9.
 - b) **Desorden Aleatorio de las Teclas:** Se utiliza la función `sort()` con un comparador que usa `Math.random() - 0.5` para mezclar aleatoriamente el arreglo de teclas.
 - c) **Construcción del HTML del Teclado:** Se genera dinámicamente el HTML del teclado en base al arreglo de teclas desordenado. Para cada tecla se crea una celda de tabla (`<td>`). Se agrupan en filas de tres teclas cada una.
 - d) **Agregar Funcionalidades Especiales:** Además de las teclas numéricas, se agregan teclas especiales para borrar el último número ingresado y para reiniciar el teclado (mezclar nuevamente las teclas). Estas teclas están representadas por íconos de Font Awesome (`fas fa-backspace` y `fas fa-sync-alt`).

Este teclado es esencial para incrementar la seguridad, ya que la posición aleatoria de las teclas dificulta que observadores malintencionados capturen la contraseña.

Listing 5: Código JavaScript para generar el teclado aleatorio

```
function generateKeypad() {
  const keys = ['1', '2', '3', '4', '5', '6', '7', '8', '9', '0'];
  keys.sort(() => Math.random() - 0.5); // Mezcla aleatoria de teclas
  let keyboardHtml = '';
  keys.forEach((key, index) => {
    if (index % 3 === 0) keyboardHtml += '<tr>';
    keyboardHtml += '<td class="keypad-button">${key}</td>';
    if ((index + 1) % 3 === 0) keyboardHtml += '</tr>';
  });
  keyboardHtml += '<tr><td class="keypad-button"><i class="fas fa-backspace"></i></td><td colspan="2" class="keypad-button"><i class="fas fa-sync-alt"></i></td></tr>';
  $('#keypad').html(keyboardHtml); // Inserta el HTML en el contenedor del teclado.
  attachKeypadHandlers(); // Asigna manejadores de eventos para los botones del teclado.
}
```

Este método no solo crea un teclado numérico aleatorio, sino que también incluye funcionalidades para que el usuario pueda corregir errores o reiniciar la secuencia de entrada, mejorando así la usabilidad del formulario.

1.4. Creación del modelo de la página con el juego El Ahorcado y la calculadora científica

1.4.1. El Juego El Ahorcado

Para implementar el juego "El Ahorcado", se utilizan varios archivos que juntos conforman la interfaz y la lógica del juego. Los archivos relevantes son:

- **ahorcado.html** - Archivo HTML que contiene la estructura básica de la página del juego.

Listing 6: Estructura básica del archivo ahorcado.html

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <title>Juego del Ahorcado</title>
  <link rel="stylesheet" href="styles.css"> <!-- Asegrate de tener este archivo o
    elimina esta linea si no tienes estilos adicionales. -->
</head>
<body>
  <h1>Juego del Ahorcado</h1>
  <div id="botonesletras"></div>
  <div id="letras"></div>
  <canvas id="canvasahorcado" width="320" height="250"></canvas>
  <script src="ahorcado.js"></script>
</body>
</html>
```

- **styles.css** - Hoja de estilos CSS que se utiliza para estilizar la página del juego.

Listing 7: Extracto de la hoja de estilos styles.css

```
/* Estilos generales y de fondo */
body, html {
  margin: 0;
  padding: 0;
  width: 100%;
  height: 100%;
  overflow: hidden; /* Previene desbordamiento y scroll innecesario */
}

.doblefondo {
  background-color: #f2f2f2;
  background-image: url('fondo1.png'), url('fondo2.png');
  background-repeat: no-repeat, no-repeat;
  background-position: left top, right bottom;
  width: 100%;
  height: 100%;
  position: relative;
}
```

```
/* Estilos para el texto animado */
.rotating-text-wrapper {
  position: absolute;
  top: 0;
  left: 0;
  right: 0;
  bottom: 0;
  width: 100%;
  height: 100vh;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  text-align: center;
  background-color: transparent;
}

.rotating-text-wrapper h2 {
  font-size: 2.5em;
  color: #fff;
  margin: 0;
  padding: 0.3em;
  box-shadow: 0 10px 10px rgba(0, 0, 0, 0.2);
  animation-duration: 6s;
  animation-iteration-count: infinite;
  opacity: 0;
}

@keyframes rotating-text-1 {
  0%, 100% { opacity: 0; transform: translateY(100%); }
  10%, 33% { opacity: 1; transform: translateY(0); }
}

/* Estilos para el botn de inicio */
.start-button {
  position: fixed;
  top: 10px;
  right: 10px;
  background-color: #4CAF50;
  color: white;
  padding: 15px 32px;
  border: none;
  text-align: center;
  font-size: 16px;
  cursor: pointer;
  border-radius: 8px;
  z-index: 1000; /* Asegura que el botn est sobre otros elementos */
}
```

Este código CSS detalla cómo se aplican los estilos al cuerpo de la página, los fondos dobles, las animaciones de texto y los botones interactivos. Cada clase y elemento está cuidadosamente diseñado para mejorar la experiencia visual y de usuario del juego.

- **ahorcado.js** - Archivo JavaScript que contiene toda la lógica necesaria para manejar la interactividad del juego.

Listing 8: Código principal de ahorcado.js

```
document.addEventListener("DOMContentLoaded", function () {
    // Lista de palabras para el juego
    const palabras = ['ahorcado', 'lavadora', 'invierno', 'plastico', 'ordenador',
        'colador', 'guanteras', 'alimentador', 'calculos'];
    // Selecciona una palabra al azar y la convierte a mayusculas
    const palabraEscogida = palabras[Math.floor(Math.random() *
        palabras.length)].toUpperCase();
    // Array para guardar las letras acertadas
    let aciertos = [];
    // Contador de fallos
    let numFallos = 0;

    // Obtener el elemento canvas y su contexto
    const canvas = document.getElementById('canvasahorcado');
    const ctx = canvas.getContext('2d');

    // Funcin para dibujar la horca
    function dibujaHorca() {
        ctx.fillStyle = '#462501'; // Color de la horca
        ctx.fillRect(64, 9, 26, 237); // Base de la horca
        ctx.fillRect(175, 193, 26, 53); // Cabeza de la horca
        ctx.fillRect(64, 193, 136, 15); // Cuerpo de la horca
        ctx.fillRect(64, 9, 115, 11); // Brazo de la horca
    }

    // Funcin para dibujar las partes del ahorcado
    function dibujaPartesAhorcado() {
        ctx.lineWidth = 5; // Grosor de las lineas

        // Dibuja cada parte del cuerpo segn el nmero de fallos
        if (numFallos > 0) ctx.arc(180, 50, 20, 0, Math.PI * 2, true); // Cabeza
        if (numFallos > 1) { ctx.moveTo(180, 70); ctx.lineTo(180, 120); } // Cuerpo
        if (numFallos > 2) { ctx.moveTo(180, 80); ctx.lineTo(160, 100); } // Brazo
        izquierdo
        if (numFallos > 3) { ctx.moveTo(180, 80); ctx.lineTo(200, 100); } // Brazo
        derecho
        if (numFallos > 4) { ctx.moveTo(180, 120); ctx.lineTo(160, 140); } // Pierna
        izquierda
        if (numFallos > 5) { ctx.moveTo(180, 120); ctx.lineTo(200, 140); } // Pierna
        derecha
        ctx.stroke(); // Aplica los trazos
    }

    // Funcin para actualizar la pista mostrada al jugador
    function actualizarPista() {
        let texto = '';
        palabraEscogida.split('').forEach((letra) => {
            texto += aciertos.includes(letra) ? letra : '_';
            texto += ' ';
        });
        document.getElementById('letras').textContent = texto;
    }

    // Funcin para verificar si la letra seleccionada est en la palabra
    function verificarLetra(letra) {
```

```
if (!palabraEscogida.includes(letra)) {
    numFallos++;
    dibujaAhorcado();
} else {
    aciertos = [...new Set([...aciertos, ...palabraEscogida.split('').filter(l => l === letra))]];
    actualizarPista();
}
// Verifica el estado del juego
if (numFallos > 5 || !palabraEscogida.split('').some(l => !aciertos.includes(l))) {
    setTimeout(() => alert('Juego terminado! ${numFallos} > 5 ? 'Perdiste' : 'Ganaste')! La palabra era: ${palabraEscogida}'), 100);
}
}

// Inicializa el juego
dibujaAhorcado();
actualizarPista();

// Crea botones para cada letra del alfabeto
'ABCDEFGHIJKLMNOPQRSTUVWXYZ'.split('').forEach(function (letra) {
    let button = document.createElement('button');
    button.textContent = letra;
    button.addEventListener('click', function () {
        this.disabled = true;
        verificarLetra(letra);
    });
    document.getElementById('botonesletras').appendChild(button);
});
});
```

Este script inicializa el juego una vez que se carga el contenido del documento, maneja los eventos de los botones para las letras, y actualiza dinámicamente el canvas y los elementos del DOM en función de los aciertos y fallos del jugador. Cada archivo contribuye a una parte específica de la funcionalidad y presentación del juego, permitiendo una modularización clara del proyecto.

1.4.2. Calculadora Científica

La calculadora científica se implementa también a través de múltiples archivos, que facilitan la organización y mantenimiento del código:

- **calculadora.html** - Archivo HTML que provee la estructura de la interfaz de usuario de la calculadora.

Listing 9: Estructura básica del archivo calculadora.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Enhanced Calculator</title>
    <link rel="stylesheet" href="calculadora.css">
</head>
```

```
<body>
  <div id="calculator">
    <div id="display">
      <div id="history"></div>
      <input type="text" id="currentInput" disabled>
    </div>
    <div id="buttons">
      <button onclick="inputDigit('(')">(</button>
      <button onclick="inputDigit(')')">)</button>
      <button onclick="performOperation('%')">mod</button>
      <button onclick="clearHistory()">C</button>

      <button onclick="inputDigit('7')">7</button>
      <button onclick="inputDigit('8')">8</button>
      <button onclick="inputDigit('9')">9</button>
      <button onclick="performOperation('/')">/</button>

      <button onclick="inputDigit('4')">4</button>
      <button onclick="inputDigit('5')">5</button>
      <button onclick="inputDigit('6')">6</button>
      <button onclick="performOperation('*')">*</button>

      <button onclick="inputDigit('1')">1</button>
      <button onclick="inputDigit('2')">2</button>
      <button onclick="inputDigit('3')">3</button>
      <button onclick="performOperation('-')">-</button>

      <button onclick="inputDigit('0')">0</button>
      <button onclick="inputDigit('.')">.</button>
      <button onclick="calculate()" class="equals">=</button>
      <button onclick="performOperation('+')">+</button>

      <button onclick="inputSquare()">x</button>
      <button onclick="inputSquareRoot()"></button>
      <button onclick="inputPi()"></button>
      <button onclick="undo()"></button>
      <button onclick="backspace()"></button>
    </div>
  </div>
  <script src="calculadora.js"></script>
</body>
</html>
```

Este archivo HTML incluye una sección de visualización donde se muestra el historial de cálculos y la entrada actual, y una matriz de botones que permiten al usuario realizar operaciones matemáticas, ingresar números, y controlar la interfaz, como borrar un número o calcular el resultado. La funcionalidad interactiva se maneja a través del script 'calculadora.js', el cual se encarga de todas las operaciones y actualizaciones de la interfaz.

- **calculadora.css** - Hoja de estilos CSS específica para la calculadora, encargada de su presentación visual.

Listing 10: Extracto de la hoja de estilos calculadora.css

```
body {
  display: flex;
  justify-content: center;
```

```
    align-items: center;
    height: 100vh;
    background-color: #f4f4f4;
    margin: 0;
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
}

#calculator {
    background-color: #fff;
    border: 1px solid #ccc;
    border-radius: 8px;
    padding: 20px;
    width: 350px;
    box-shadow: 0 4px 10px rgba(0, 0, 0, 0.1);
}

#display {
    background-color: #e6e6e6;
    border-radius: 6px;
    padding: 20px;
    margin-bottom: 20px;
}

#currentInput {
    width: 100%;
    border: none;
    background-color: transparent;
    text-align: right;
    font-size: 2em;
    outline: none;
    color: #333;
}

#buttons {
    display: grid;
    grid-template-columns: repeat(4, 1fr);
    gap: 10px;
}

#buttons button {
    padding: 15px;
    border: none;
    border-radius: 6px;
    background-color: #f0f0f0;
    cursor: pointer;
    font-size: 1.5em;
    color: #333;
    transition: background-color 0.3s;
}

#buttons button:hover {
    background-color: #e8e8e8;
}

#history {
    height: 100px;
```

```
        overflow-y: auto;
        text-align: right;
        margin-bottom: 20px;
        font-size: 0.85em;
        color: #555;
        border-radius: 6px;
        padding: 10px;
        background-color: #e6e6e6;
    }

    .history-entry {
        padding: 5px;
        margin-top: 2px;
        cursor: pointer;
        background-color: #f0f0f0;
    }

    .history-entry:hover {
        background-color: #e0e0e0;
    }

    #buttons .equals {
        background-color: #ff9500;
        color: white;
    }

    #buttons .equals:hover {
        background-color: #e68900;
    }
```

Este CSS estiliza el cuerpo de la página para centrar la calculadora en la pantalla y utiliza un diseño de cuadrícula para los botones para garantizar que la interfaz sea responsiva y fácil de usar. Los colores y las transiciones están diseñados para mejorar la interacción visual del usuario, mientras que el historial de operaciones se presenta de manera que es fácil de seguir y consultar.

- **calculadora.js** - Script JavaScript que implementa las funciones de cálculo y las interacciones de la calculadora.

Listing 11: Funcionalidades principales de calculadora.js

```
let historyStack = [];
let currentExpression = '';

function inputDigit(digit) {
    currentExpression += digit;
    updateDisplay();
}

function performOperation(operator) {
    if (['+', '-', '*', '/', '%', '**'].includes(currentExpression.slice(-1))) {
        return; // Prevent consecutive operators
    }
    currentExpression += ' ${operator} ';
    updateDisplay();
}
```

```
function calculate() {
  try {
    let expression = currentExpression.replace(/g, 'Math.PI').replace(/g,
      'Math.sqrt');
    const result = eval(expression);
    historyStack.push(currentExpression + ' = ' + result);
    currentExpression = result.toString();
    updateDisplay();
    updateHistory();
  } catch (error) {
    currentExpression = 'Error';
    updateDisplay();
    setTimeout(clearHistory, 2000); // Clear after showing error for 2 seconds
  }
}

function clearHistory() {
  currentExpression = '';
  historyStack = [];
  updateDisplay();
  updateHistory();
}

function undo() {
  if (historyStack.length > 0) {
    currentExpression = historyStack.pop().split(' = ')[0];
    updateDisplay();
    updateHistory();
  }
}

function backspace() {
  if (currentExpression.endsWith(' ')) {
    currentExpression = currentExpression.slice(0, -3); // Remove the last operator
  } else {
    currentExpression = currentExpression.slice(0, -1); // Remove the last digit
  }
  updateDisplay();
}

function updateDisplay() {
  document.getElementById('currentInput').value = currentExpression;
}

function updateHistory() {
  const historyElement = document.getElementById('history');
  historyElement.innerHTML = '';
  historyStack.forEach((entry, index) => {
    const entryElement = document.createElement('div');
    entryElement.textContent = entry;
    entryElement.className = 'history-entry';
    entryElement.onclick = function() {
      loadHistory(index);
    };
    historyElement.appendChild(entryElement);
  });
}
```

```
});  
}  
  
function loadHistory(index) {  
    const entry = historyStack[index];  
    currentExpression = entry.split('=')[0];  
    updateDisplay();  
}  
  
function inputSquare() {  
    currentExpression += '**2';  
    updateDisplay();  
}  
  
function inputSquareRoot() {  
    if (currentExpression.length === 0 ||  
        isNaN(currentExpression[currentExpression.length - 1])) {  
        currentExpression += 'Math.sqrt(';  
    } else {  
        currentExpression += ' * Math.sqrt(';  
    }  
    updateDisplay();  
}  
  
function inputPi() {  
    currentExpression += 'Math.PI';  
    updateDisplay();  
}  
  
window.onload = function() {  
    updateDisplay();  
    updateHistory();  
};
```

Este código JavaScript es esencial para la funcionalidad de la calculadora, gestionando no solo los cálculos sino también la interfaz de usuario, lo que incluye la visualización de entradas y resultados, así como el manejo de errores y el historial de operaciones. La separación de la lógica y la presentación en archivos diferentes mejora la legibilidad y mantenibilidad del código.

1.5. Otros archivos del proyecto

Además de los archivos específicos para el ahorcado y la calculadora, el proyecto incluye otros archivos que son esenciales para el funcionamiento general de la página web:

2. Cree una calculadora básica como la de los sistemas operativos, que pueda utilizar la función `eval()` y que guarde todas las operaciones en una pila. Mostrar la pila al pie de la página web.

1.6. Función de cálculo y manejo de historial

La función `calculate()` en el script de la calculadora se encarga de evaluar las expresiones matemáticas ingresadas por el usuario y de manejar el historial de cálculo, almacenando cada operación en una pila. Esta sección muestra cómo se evalúa la entrada y cómo se actualiza el historial que se muestra al pie de la página web.

Listing 12: Función calculate y manejo del historial en calculadora.js

```
function calculate() {
  try {
    // Reemplazar caracteres específicos por su equivalente en JavaScript
    let expression = currentExpression.replace(/g, 'Math.PI').replace(/g,
      'Math.sqrt');
    // Evaluar la expresión matemática
    const result = eval(expression);
    // Guardar la operación y el resultado en la pila de historial
    historyStack.push(currentExpression + ' = ' + result);
    // Actualizar la expresión actual con el resultado
    currentExpression = result.toString();
    updateDisplay();
    updateHistory();
  } catch (error) {
    // Manejo de errores en la evaluación
    currentExpression = 'Error';
    updateDisplay();
    setTimeout(clearHistory, 2000); // Limpiar historia tras mostrar el error por 2
      segundos
  }
}

function updateHistory() {
  const historyElement = document.getElementById('history');
  historyElement.innerHTML = ''; // Limpiar entradas de historial anteriores
  historyStack.forEach((entry, index) => {
    const entryElement = document.createElement('div');
    entryElement.textContent = entry;
    entryElement.className = 'history-entry'; // Clase para estilos
    entryElement.onclick = function() {
      loadHistory(index); // Cargar esta entrada del historial al ser clickeada
    };
    historyElement.appendChild(entryElement);
  });
}
```

Esta función no solo calcula los resultados de las expresiones sino que también gestiona un historial de operaciones, mostrando todas las operaciones realizadas y sus resultados en el pie de la página web. La visualización del historial permite a los usuarios ver y reutilizar cálculos anteriores.

3. Cree una versión del juego 'El Ahorcado' que se grafique con canvas paso a paso desde el evento onclick() de un botón.

1.7. Implementación del juego .^{El} Ahorcado con Canvas

El juego .^{El} Ahorcado.^{es} un juego clásico en el que los jugadores intentan adivinar una palabra al adivinar letras una a la vez. Aquí se presenta una versión del juego que utiliza la tecnología Canvas para graficar paso a paso el ahorcado a medida que el jugador comete errores.

1. **Estructura HTML:** El archivo HTML contiene la estructura básica de la página del juego, que incluye un elemento canvas para dibujar el ahorcado y otros elementos para mostrar la palabra a adivinar y las letras seleccionadas por el jugador.

Listing 13: Archivo HTML para el juego .^{E1} Ahorcado”

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <title>Juego del Ahorcado</title>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <h1>Juego del Ahorcado</h1>
  <div id="botonesletras"></div>
  <div id="letras"></div>
  <canvas id="canvasahorcado" width="320" height="250"></canvas>
  <script src="ahorcado.js"></script>
</body>
</html>
```

2. **Estilos CSS:** Se utiliza una hoja de estilos CSS para estilizar la página del juego y los elementos HTML.

Listing 14: Hoja de estilos CSS para el juego .^{E1} Ahorcado”

```
/* Estilos generales y de fondo */
body, html {
  margin: 0;
  padding: 0;
  width: 100%;
  height: 100%;
  overflow: hidden; /* Previene desbordamiento y scroll innecesario */
}
...
```

3. **Script JavaScript:** El archivo JavaScript contiene la lógica del juego, incluyendo la selección aleatoria de palabras, la gestión de errores del jugador y la representación gráfica del ahorcado en Canvas.

Listing 15: Archivo JavaScript para el juego .^{E1} Ahorcado”

...

Esta implementación utiliza la tecnología Canvas para proporcionar una experiencia interactiva y visualmente atractiva del juego .^{E1} Ahorcado”, permitiendo a los jugadores ver cómo se dibuja paso a paso el ahorcado a medida que cometen errores en sus intentos de adivinar la palabra secreta.

1.8. Pregunta

Explique una herramienta para ofuscar código JavaScript. Muestre un ejemplo de su uso en uno de los ejercicios de la tarea.

1.9. Herramienta para Ofuscar Código JavaScript

Para ofuscar código JavaScript, una herramienta ampliamente utilizada es Terser. Terser es un minificador y ofuscador de JavaScript de alto rendimiento y fácil de usar que ayuda a reducir el

tamaño del código y proteger la propiedad intelectual al hacer que el código resultante sea menos legible para los humanos.

1.9.1. Instalación de Terser

Antes de usar Terser, primero debemos instalarlo globalmente en nuestra máquina. Esto se puede hacer utilizando npm (Node Package Manager), que es el sistema de gestión de paquetes para Node.js.

```
npm install -g terser
```

Este comando instalará Terser globalmente en tu sistema, lo que te permitirá ejecutarlo desde cualquier lugar en la línea de comandos.

1.9.2. Uso de Terser para Ofuscar Código JavaScript

Una vez que Terser está instalado, podemos usarlo para ofuscar nuestro código JavaScript. El siguiente comando muestra cómo usar Terser para ofuscar un archivo JavaScript llamado 'scripts.js' y guardar el resultado en un nuevo archivo llamado 'scripts.min.js':

```
terser scripts.js -o scripts.min.js -m
```

En este comando:

- 'terser' es el comando para ejecutar Terser.
- 'scripts.js' es el nombre del archivo JavaScript de entrada que queremos ofuscar.
- '-o scripts.min.js' especifica el nombre del archivo de salida donde se guardará el código ofuscado.
- '-m' es una opción que indica a Terser que minimice y ofusque el código resultante.

Este comando tomará el archivo 'scripts.js', lo ofuscará y lo guardará en 'scripts.min.js'. El código resultante estará minificado y ofuscado, lo que significa que será más difícil de entender para los humanos pero seguirá siendo funcional para las máquinas.

1.10. Resultado del Ofuscado

Después de ejecutar el comando para ofuscar el código JavaScript, podemos verificar el resultado observando el archivo 'scripts.min.js'. El código resultante estará minificado y ofuscado, lo que significa que será más difícil de entender para los humanos pero seguirá siendo funcional para las máquinas.

1.11. Formulario del Banco

1.12. Selección de Ahorcado o Calculadora

1.13. Página Principal del Juego y de la Calculadora

1.14. Página del Juego del Ahorcado

1.15. Página de la Calculadora

```
> $(document).ready((function(){a();e();c();$("#docType").change(e);function
a(){const a=["1","2","3","4","5","6","7","8","9","0"];a.sort(()=>Math.rand
om()-.5));let e="";a.forEach((a,t)=>{if(t%3===0)e+="

```

Figura 1: Ejemplo de código JavaScript ofuscado exitosamente

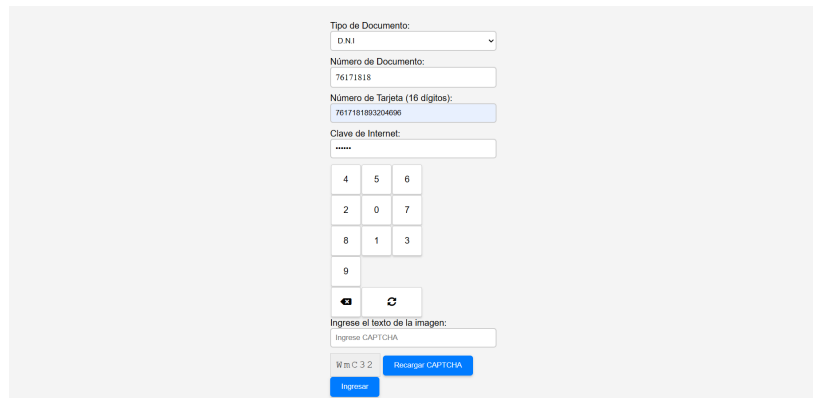


Figura 2: Formulario del Banco



Figura 3: Ventana para escoger entre el juego del Ahorcado y la Calculadora

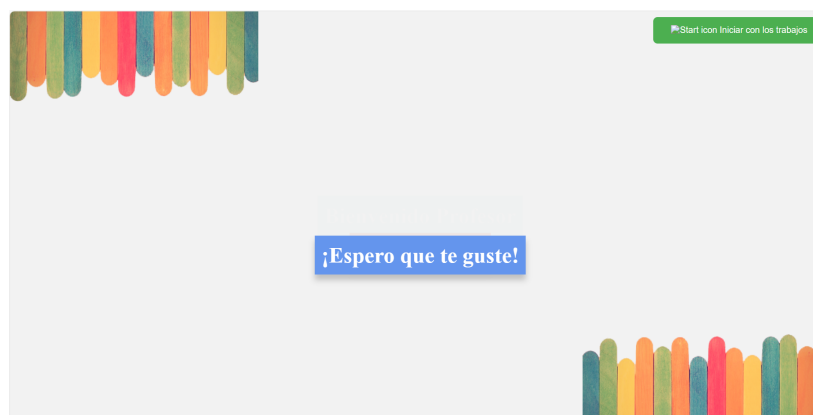


Figura 4: Página principal del juego y de la calculadora

Juego del Ahorcado

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---



Figura 5: Página del juego del Ahorcado

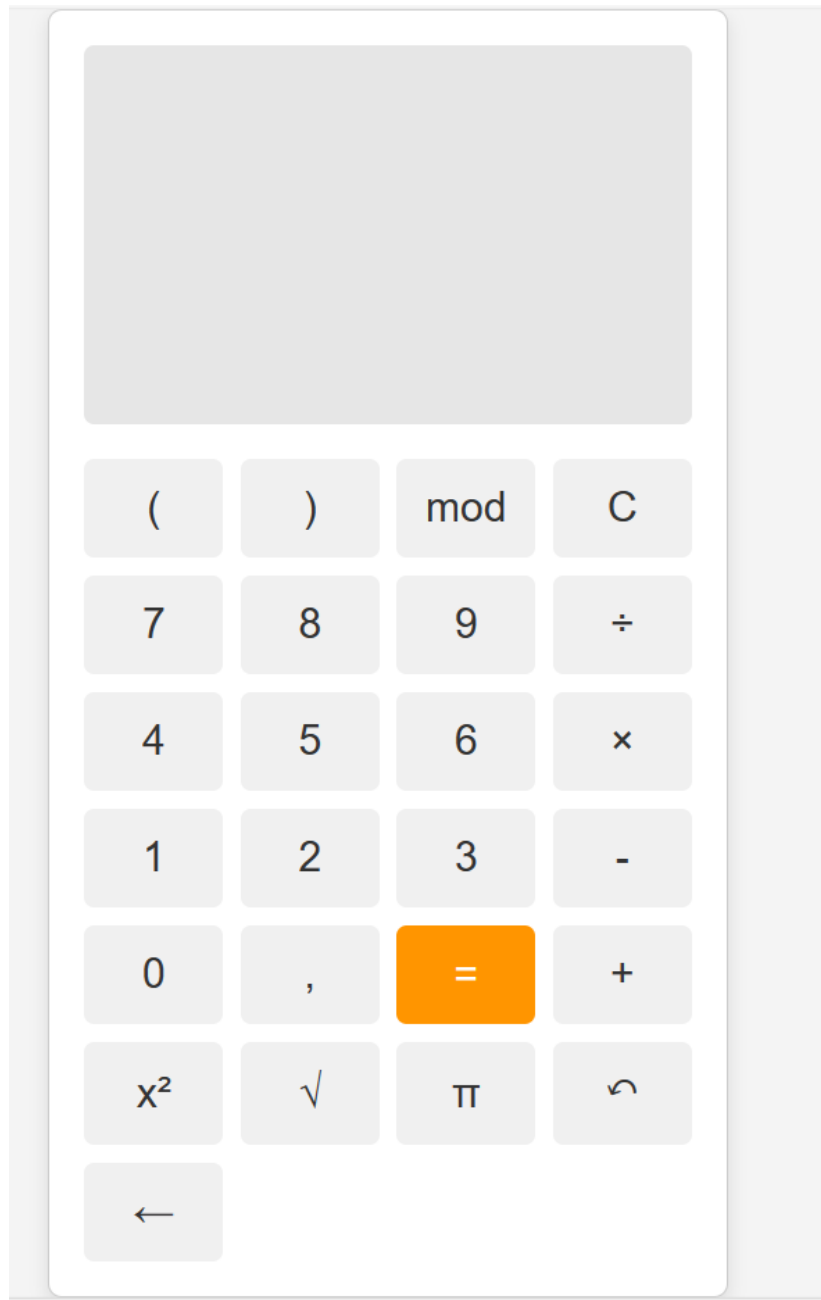


Figura 6: Página de la Calculadora