

# Informe de Laboratorio 03

## Tema: Laboratorio 03

Nota

Estudiante	Escuela	Asignatura
Jhonatan David Arias Quispe jariasq@unsa.edu.pe	Escuela Profesional de Ingeniería de Sistemas	Programación Web 2 Semestre: II Código: 1702122

Laboratorio	Tema	Duración
03	Laboratorio 03	04 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2024 - A	Del 15 Mayo 2024	Al 18 Mayo 2024

## 1. Actividades

### 1.1. Descripción

- Programar en Javascript sobre una pagina web html basica.
- Ejercicio 01: Cree un teclado random para banca por internet.
- Ejercicio 02: Cree una calculadora básica como la de los sistemas operativos, que pueda utilizar la funcion eval().
- Cree una versión del juego "el ahorcado" que grafique con canvas paso a paso desde el evento onclick() de un botón.

### 1.2. Pregunta

- Explique una herramienta para ofuzcar código JavaScript.
- Muestre un ejemplo de su uso en uno de los ejercicios de la tarea..
- Adjunte a su repositorio ambas versiones:
  - script-ejercicio-01.js (development).
  - script-ejercicio-01.min.js (production).

## 2. Equipos, materiales y temas utilizados

- Sistema Operativo ArchCraft GNU Linux 64 bits Kernell
- NeoVim
- Git 2.42.0
- Cuenta en GitHub con el correo institucional.
- HTML5
- CSS3
- JavaScript
- JavaScript Obfuscador
- Latex

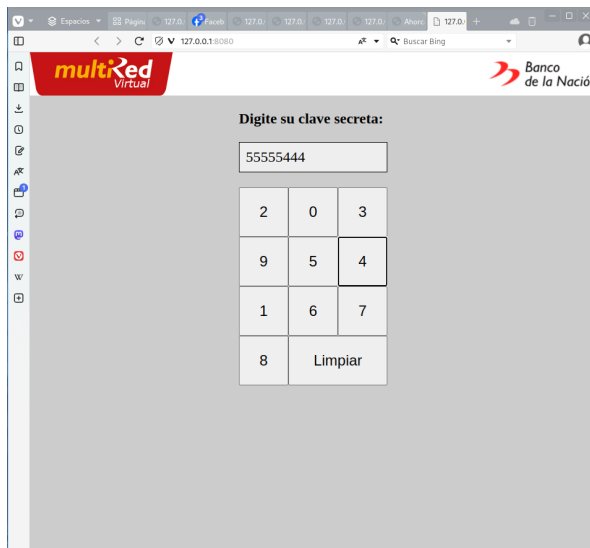
## 3. URL de Repositorio Github

- URL del Repositorio GitHub para clonar o recuperar.
- <https://github.com/JhonatanDczel/pweb2.git>
- URL para el laboratorio 03 en el Repositorio GitHub.
- <https://github.com/JhonatanDczel/pweb2/tree/main/lab03>

## 4. Antecedentes

Javascript es un lenguaje multidinamico usado en su mayoria en el desarrollo web como lenguaje de front end y backend (Gracias a node) y como tal, surgieron muchas herramientas para ofuscar el codigo en JavaScript, una de ellas JavaScript Obfuscator Tool, la podemos encontrar en la siguiente direccion: <https://obfuscator.io/code>

## Ejercicio 01 - Teclado Random



Este informe detalla la creación y configuración de un documento HTML dinámico utilizando JavaScript puro. El código tiene como objetivo generar una interfaz de usuario (UI) que incluye una barra de navegación con dos logotipos y un área principal que contiene un campo de entrada de texto simulado y un teclado numérico aleatorio. A continuación, se explica cada parte del código con porciones de código relevantes.

### Creación de la Barra de Navegación

Primero, se crea un elemento `nav` y se agregan dos imágenes de logotipos a esta barra de navegación.

```
1 let nav = document.createElement("nav");
2
3 let logo = document.createElement("img");
4 logo.src = "./img/logo-multired.jpg";
5 nav.appendChild(logo);
6
7 let logo2 = document.createElement("img");
8 logo2.src = "./img/Logo_BN.jpg";
9 logo2.style.height = "40px";
10 logo2.style.width = "auto";
11 logo2.style.alignSelf = "center";
12 nav.appendChild(logo2);
```

En esta parte del código, se crean los elementos `nav` y `img` utilizando `document.createElement`. Los atributos de las imágenes (como `src` y estilos) se definen y las imágenes se añaden como hijos del elemento `nav`.

### Estilos para la Barra de Navegación

Luego, se aplican estilos CSS al `nav` para estructurarlo visualmente.

```
1 nav.style.display = "flex";
2 nav.style.justifyContent = "space-between";
3
4 document.body.appendChild(nav);
```

Se utiliza `style` para establecer la propiedad `display` como `flex`, lo que permite una distribución flexible de los elementos hijos, y `justifyContent` para espaciar los elementos (los logotipos) uniformemente.

## Creación del Área Principal

A continuación, se crea el elemento `main` y un contenedor `div` dentro de él.

```
1 let main = document.createElement("main");
2 document.body.appendChild(main);
3
4 let mainDiv = document.createElement("div");
5 main.appendChild(mainDiv);
```

## Adición de la Etiqueta y Campo de Entrada

Dentro del `div` principal, se agrega un `p` para la etiqueta y otro `p` para simular el campo de entrada de texto.

```
1 let label = document.createElement("p");
2 label.textContent = "Digite su clave secreta:";
3 label.style.fontSize = "1.5rem";
4 label.style.fontWeight = "bold";
5 mainDiv.appendChild(label);
6
7 let camp = document.createElement("p");
8 camp.style.fontSize = "1.5rem";
9 mainDiv.appendChild(camp);
10 camp.style.minHeight = "1.5rem";
11 camp.style.padding = "10px 10px";
12 camp.style.border = "1px solid black";
13 camp.style.backgroundColor = "#eee";
14 camp.style.display = "flex";
15 camp.style.alignContent = "center";
16 camp.style.overflow = "hidden";
```

Estos elementos de párrafo (`p`) se estilizan para parecer un campo de entrada de texto. El campo de entrada (`camp`) tiene estilos adicionales para simular un área de entrada segura.

## Función para Mezclar los Números

Se define una función `shuffle` para mezclar un array de números, que se usará para crear el teclado numérico aleatorio.

```
1 function shuffle(array) {
2   for (let i = array.length - 1; i > 0; i--) {
3     const j = Math.floor(Math.random() * (i + 1));
4     [array[i], array[j]] = [array[j], array[i]];
5   }
6   return array;
7 }
8
9 let numbers = Array.from({ length: 10 }, (_, i) => i);
10 numbers = shuffle(numbers);
```

La función `shuffle` utiliza el algoritmo de Fisher-Yates para mezclar los elementos de un array.

## Creación del Teclado Numérico

El teclado numérico se crea como un `div` y se añaden botones numerados, incluyendo un botón de "Limpiar".

```
1 let keyboard = document.createElement("div");
2
3 for (let i = 0; i <= 10; i++) {
4   let key = document.createElement("button");
5   let name = i < 10 ? numbers[i] : "Limpiar";
6   if (i == 10) {
7     key.style.gridColumn = "span 2";
8   }
9   key.textContent = name;
10  key.style.height = "80px";
11  keyboard.appendChild(key);
12  key.addEventListener("click", () => {
13    if (name == "Limpiar") {
14      camp.textContent = "";
15      return;
16    }
17    camp.textContent += name;
18  });
19  key.style.fontSize = "1.5rem";
20 }
21
22 mainDiv.appendChild(keyboard);
```

Cada botón tiene un `event listener` que actualiza el campo de entrada (`camp`) con el número correspondiente o lo limpia si se presiona "Limpiar".

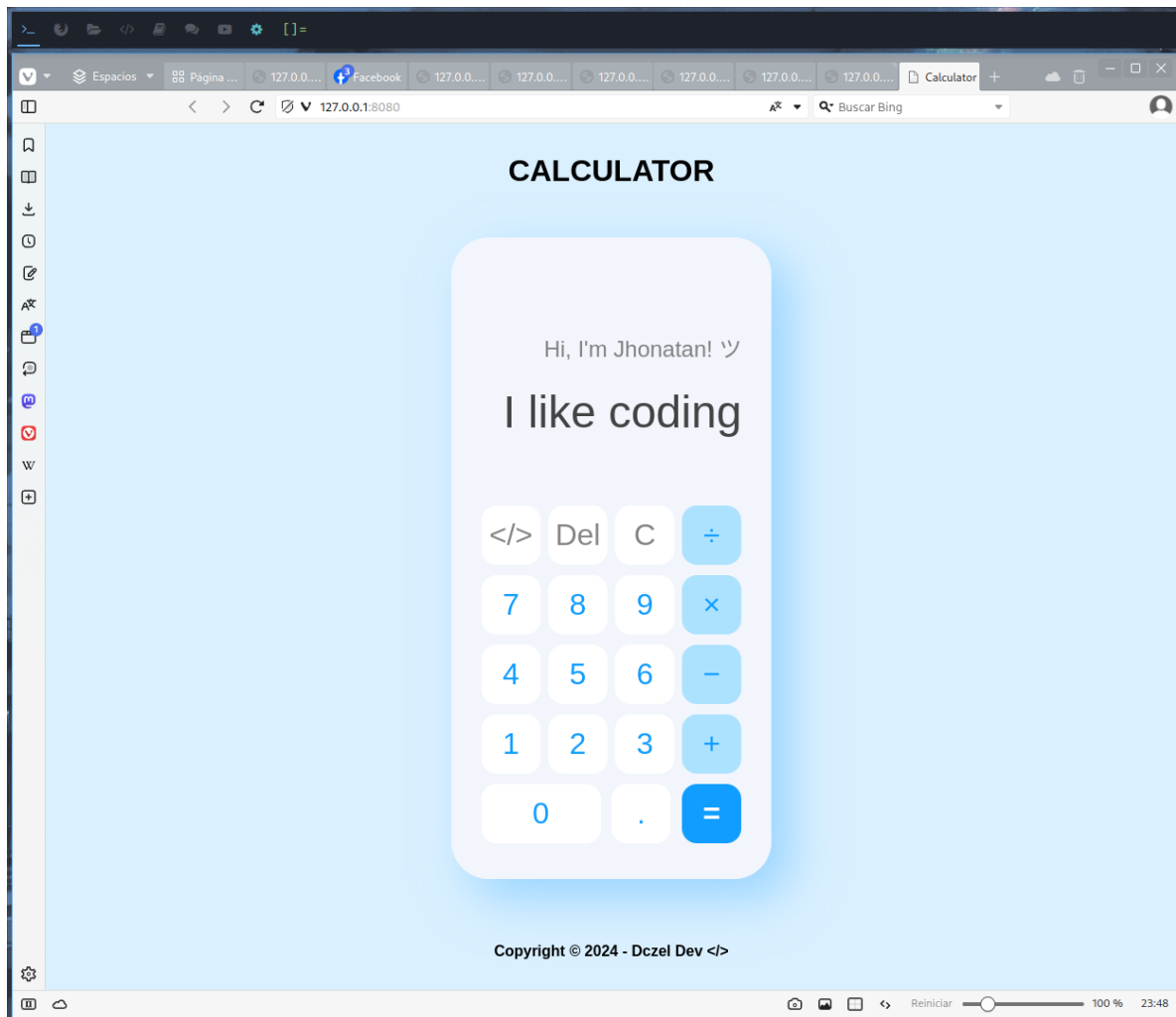
## Estilos Adicionales para el Documento y los Elementos

Se aplican estilos globales y específicos a varios elementos para asegurar una presentación consistente.

```
1 document.documentElement.style.margin = 0;
2 document.documentElement.style.padding = 0;
3 document.body.style.margin = 0;
4 document.body.style.padding = 0;
5
6 document.body.style.height = "calc(100vh - 70px)";
7 main.style.width = "100vw";
8 main.style.height = "calc(100vh - 70px)";
9 main.style.display = "flex";
10 main.style.backgroundColor = "#ccc";
11 main.style.justifyContent = "center";
12 main.style.alignContent = "center";
13
14 mainDiv.style.width = "240px";
15 mainDiv.style.height = "100px";
16
17 keyboard.style.display = "grid";
18 keyboard.style.gridTemplateColumns = "1fr 1fr 1fr";
```

Estos estilos aseguran que el documento ocupe toda la ventana del navegador, que el contenido principal esté centrado y que el teclado numérico esté distribuido en una cuadrícula de tres columnas.

## 5. Ejercicio 02



El archivo components contiene la estructura de html que tiene el proyecto, se aprovecha el uso de template literals para manejar mejor las cosas.

Listing 1: Archivo components.js

```

1 document.body.innerHTML = /*html*/`
2   <div class="header">
3     <h1>CALCULATOR</h1>
4   </div>
5   <div class="calculator">
6     <div class="display">
7       <div id='d1' class="display-1"></div>
8       <div id='d2' class="display-2">0</div>
9     </div>
10    <div class="buttons">
11    </div>
12  </div>
13  <div class="footer">
14    <div class="footer">
15      <a href='https://github.com/JhonatanDczel' target='_blank'><p class="autor">Copyright 2023 -
16      Dczel Dev <span class="icon">&lt;/span></p></a>

```

```
17     </div>
18   </div>
19   ';
```

Es un pequeño easteregg que agregué para cuando alguien aprete el botón de ¡/!

Listing 2: Archivo easteregg.js

```
1 const btn1 = document.getElementById("</>");
2 const btn2 = document.getElementById("2");
3 const btn3 = document.getElementById("0");
4 const btn4 = document.getElementById("6");
5 const d1 = document.getElementById("d1");
6 const d2 = document.getElementById("d2");
7
8 let event1 = false;
9 let event2 = false;
10 let event3 = false;
11 let event4 = false;
12 let time = 3000;
13
14 let desactivar = (ev) => {
15   setTimeout(() => {
16     ev = false;
17     console.log("desactivado");
18   }, time);
19 };
20
21 btn1.addEventListener("click", () => {
22   event1 = true;
23   console.log("Activated");
24   desactivar(event1);
25 });
26
27 btn2.addEventListener("click", () => {
28   event2 = true;
29   console.log("Activated");
30   desactivar(event2);
31 });
32
33 btn3.addEventListener("click", () => {
34   event3 = true;
35   console.log("Activated");
36   desactivar(event3);
37 });
38
39 btn4.addEventListener("click", () => {
40   event4 = true;
41   console.log("Activated");
42   if (event1 && event2 && event3 && event4) {
43     d1.textContent = "Easter egg xd";
44     d2.textContent = "No se que poner";
45   }
46   desactivar(event4);
47 };
```

El archivo Script3.js contiene la lógica que hace funcionar la calculadora

Listing 3: Archivo script3.js

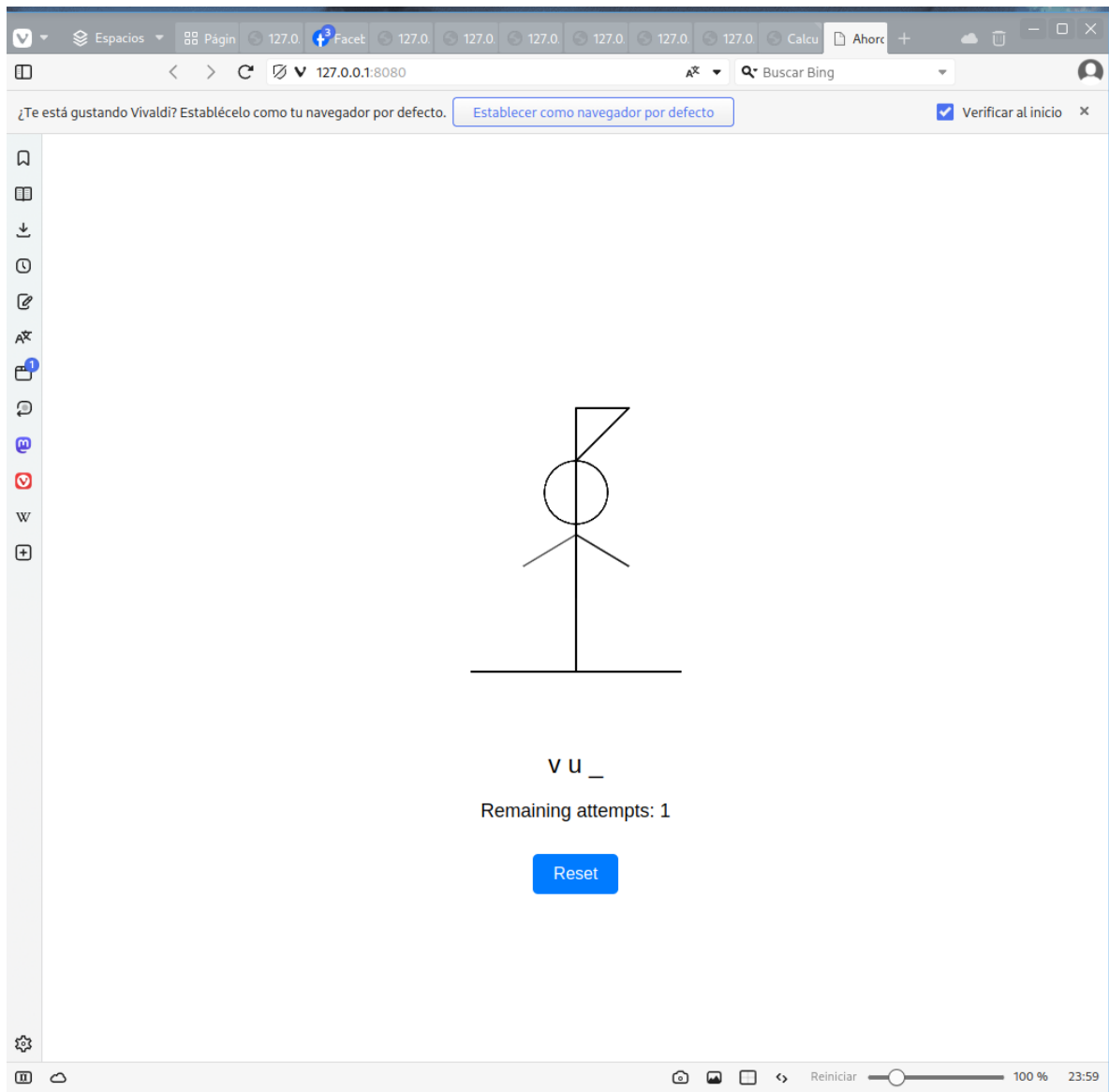
```
1 //Variables globales
2 const buttons = document.querySelector(".buttons");
3
4 // Arreglos de botones para la interfaz
5 const toolBar = ["</>", "Del", "C"];
```

```

6 const numPad = ["7", "8", "9", "4", "5", "6", "1", "2", "3", "0", "."];
7 const operations = ["", " ", " ", " ", "+", "="];
8
9 // Función para crear los botones de la calculadora
10 makeButtons();
11 window.addEventListener("resize", makeButtons);

```

## 6. Ejercicio 03



Aquí veremos la lógica principal del juego. Veamos el código paso a paso:

```

1 const canvas = document.getElementById('hangmanCanvas');
2 const ctx = canvas.getContext('2d');

```

Se obtienen referencias al elemento del canvas (canvas) y su contexto de dibujo 2D (ctx).



```
1 var wordContainer = document.getElementById('word-container');
2 var guessesContainer = document.getElementById('guesses-container');
3 var resetButton = document.getElementById('reset-button');
```

Se obtienen referencias a los elementos del DOM que mostrarán la palabra a adivinar (wordContainer), las letras adivinadas (guessesContainer) y el botón de reinicio (resetButton).

```
1 const words = ['javascript', 'html', 'css', 'python', 'react', 'angular', 'vue'];
2 let selectedWord = words[Math.floor(Math.random() * words.length)];
3 let guessedLetters = [];
4 let remainingAttempts = 6;
```

Se define un array de palabras (words) para elegir una aleatoriamente. Se selecciona una palabra aleatoria del array (selectedWord). Se inicializa un array vacío para almacenar las letras adivinadas (guessedLetters). Se define la cantidad inicial de intentos restantes (remainingAttempts).

#### Funciones del juego:

- **drawHangman()**: Esta función dibuja el ahorcado en el canvas en base a la cantidad de intentos restantes.
- **displayWord()**: Esta función muestra la palabra a adivinar en el elemento wordContainer. Si una letra ya ha sido adivinada, se muestra la letra, sino se muestra un guión bajo.
- **displayGuesses()**: Esta función muestra el número de intentos restantes en el elemento guessesContainer.
- **checkWin()**: Esta función verifica si el jugador ha ganado. Recorre la palabra seleccionada y verifica si todas las letras están incluidas en el array de letras adivinadas.
- **checkLose()**: Esta función verifica si el jugador ha perdido. Simplemente compara si la cantidad de intentos restantes es igual a cero.
- **guessLetter(letter)**: Esta función maneja las pulsaciones de teclas del usuario. Recibe una letra como parámetro y verifica si:
  - La letra no ha sido adivinada previamente.
  - La letra está incluida en la palabra seleccionada.
  - Si la letra no está en la palabra, se resta un intento. Se actualiza el dibujo del ahorcado, se muestra la palabra actualizada y los intentos restantes.
  - Si se adivina la palabra completa, se muestra un mensaje de victoria y se reinicia el juego.
  - Si se pierden todos los intentos, se muestra la palabra y se reinicia el juego.
- **resetGame()**: Esta función reinicia el juego seleccionando una nueva palabra aleatoria, vaciando las letras adivinadas, reiniciando los intentos y actualizando la pantalla.

#### Eventos:

```
1 document.addEventListener('keydown', event => {...})
```

Escucha eventos de presión de teclas. Si la tecla presionada es una letra del alfabeto, se llama a la función guessLetter con la letra presionada en minúsculas.

```
1 resetButton.addEventListener('click', resetGame)
```

Escucha el click en el botón de reinicio y llama a la función resetGame.

## components.js

Este archivo se encarga de crear dinámicamente los elementos HTML necesarios para el juego y organizarlos en la página.

```
1 // Crear el contenedor del juego
2 var gameContainer = document.createElement("div");
3 gameContainer.id = "game-container";
4
```

```
5 // ... (se crean el canvas, los contenedores para la palabra, letras adivinadas y botn de reinicio)
6
7 // Agregar el contenedor del juego al body
8 document.body.appendChild(gameContainer);
```

Se crea un elemento div con el identificador game-container que servirá como contenedor principal del juego. Se crean individualmente los elementos para el canvas (hangmanCanvas), los contenedores de texto (wordContainer y guessesContainer) y el botón de reinicio (resetButton). Se añaden todos los elementos creados como hijos del contenedor principal (gameContainer). Finalmente, se añade el contenedor principal (gameContainer) como hijo del elemento body del documento HTML.

## styles.js

Este archivo define los estilos CSS para los elementos del juego.

```
1 // Establecer estilos para el body
2 document.body.style.fontFamily = 'Arial, sans-serif';
3 document.body.style.display = 'flex';
4 document.body.style.justifyContent = 'center';
5 document.body.style.alignItems = 'center';
6 document.body.style.height = '100vh';
7 document.body.style.margin = '0';
8
9 // Establecer estilos para el contenedor del juego
10 var gameContainer = document.getElementById('game-container');
11 gameContainer.style.textAlign = 'center';
12
13 // Establecer estilos para el contenedor de la palabra
14 var wordContainer = document.getElementById('word-container');
15 wordContainer.style.margin = '20px 0';
16 wordContainer.style.fontSize = '24px';
17
18 // Establecer estilos para el contenedor de las letras adivinadas
19 var guessesContainer = document.getElementById('guesses-container');
20 guessesContainer.style.margin = '10px 0';
21 guessesContainer.style.fontSize = '18px';
22
23 // Establecer estilos para el botn de reset
24 var resetButton = document.getElementById('reset-button');
25 resetButton.style.marginTop = '20px';
26 resetButton.style.padding = '10px 20px';
27 resetButton.style.fontSize = '16px';
28 resetButton.style.backgroundColor = '#007bff';
29 resetButton.style.color = 'white';
30 resetButton.style.border = 'none';
31 resetButton.style.borderRadius = '5px';
32 resetButton.style.cursor = 'pointer';
33
34 // Establecer estilos para el hover del botn de reset
35 resetButton.addEventListener('mouseover', () => {
36   resetButton.style.backgroundColor = '#0056b3';
37 });
38
39 resetButton.addEventListener('mouseout', () => {
40   resetButton.style.backgroundColor = '#007bff';
```

Se establecen estilos globales para el elemento body del documento HTML, centrando el contenido y definiendo una altura del 100 % del viewport. Se definen estilos para el contenedor principal del juego (gameContainer), centrando el texto dentro de él. Se definen estilos para los contenedores de la palabra (wordContainer) y las letras adivinadas (guessesContainer), ajustando márgenes y tamaño de letra. Se

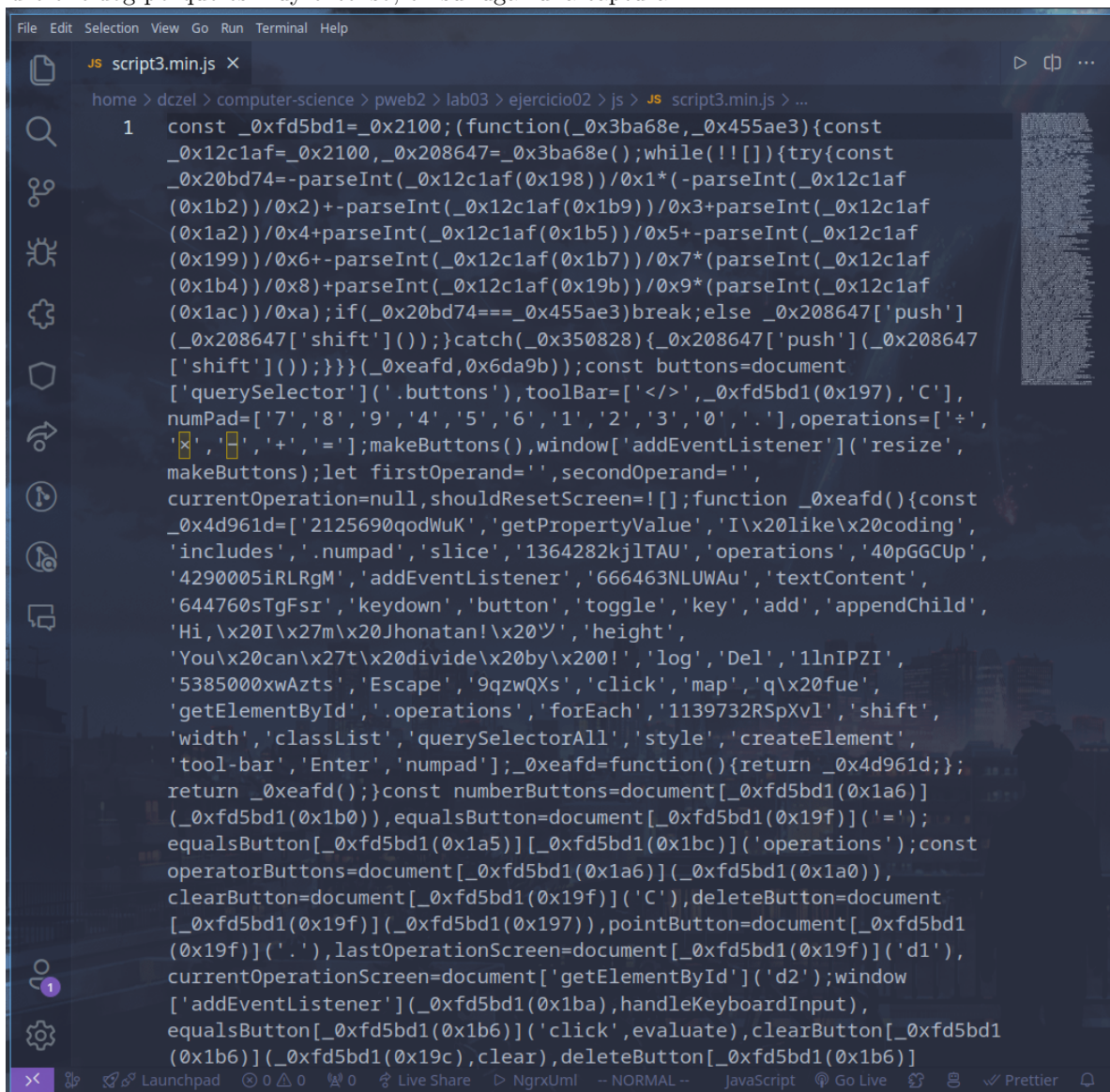
definen estilos para el botón de reinicio (resetButton).

## 7. Pregunta

El ofuscamiento de código en JavaScript es el proceso de transformar el código fuente legible y entendible en un código más difícil de entender, generalmente con el objetivo de proteger la propiedad intelectual, dificultar la ingeniería inversa y reducir el tamaño del archivo para mejorar el rendimiento de la aplicación. Es importante porque ayuda a proteger el código contra el robo o la modificación no autorizada, y también puede hacer que sea más difícil para los hackers encontrar y explotar vulnerabilidades en el software.

### 7.1. Ejemplo de ofuscamiento

Aquí tenemos el ejemplo del código de la calculadora ofuscado con JavaScriptObfuscator Tool, no pondré el código porque es muy extenso, en su lugar una captura:



```

1  const _0xfd5bd1=_0x2100;(function(_0x3ba68e,_0x455ae3){const
  _0x12c1af=_0x2100,_0x208647=_0x3ba68e();while(![]){try{const
  _0x20bd74=-parseInt(_0x12c1af(0x198))/0x1*(-parseInt(_0x12c1af
  (0x1b2))/0x2)+-parseInt(_0x12c1af(0x1b9))/0x3+parseInt(_0x12c1af
  (0x1a2))/0x4+parseInt(_0x12c1af(0x1b5))/0x5+-parseInt(_0x12c1af
  (0x199))/0x6+-parseInt(_0x12c1af(0x1b7))/0x7*(parseInt(_0x12c1af
  (0x1b4))/0x8)+parseInt(_0x12c1af(0x19b))/0x9*(parseInt(_0x12c1af
  (0x1ac))/0xa);if(_0x20bd74===_0x455ae3)break;else _0x208647['push']
  (_0x208647['shift']());}}catch(_0x350828){_0x208647['push'](_0x208647
  ['shift']());}})(_0xeafd,0x6da9b));const buttons=document
  ['querySelector']('.buttons'),toolBar=['</>','_0xfd5bd1(0x197)','C'],
  numPad=['7','8','9','4','5','6','1','2','3','0','.'],operations=['÷',
  '×','-','+','='];makeButtons(),window['addEventListener']('resize',
  makeButtons);let firstOperand='',secondOperand='',
  currentOperation=null,shouldResetScreen=![];function _0xeafd(){const
  _0x4d961d=['2125690qodWuK','getPropertyValue','I\20like\20coding',
  'includes','.numpad','slice','1364282kjlTAU','operations','40pGGCUp',
  '4290005iRLRgM','addEventListener','666463NLUWAu','textContent',
  '644760sTgFsr','keydown','button','toggle','key','add','appendChild',
  'Hi,\20I\20m\20Jhonatan!\20\20','height',
  'You\20can\20t\20divide\20by\200!', 'log','Del','1lnIPZI',
  '5385000xwAzts','Escape','9qzwQXs','click','map','q\20fue',
  'getElementById','.operations','forEach','1139732RSpXv1','shift',
  'width','classList','querySelectorAll','style','createElement',
  'tool-bar','Enter','numpad'];_0xeafd=function(){return _0x4d961d;};
  return _0xeafd();}const numberButtons=document[_0xfd5bd1(0x1a6)]
  (_0xfd5bd1(0x1b0)),equalsButton=document[_0xfd5bd1(0x19f)]('=');
  equalsButton[_0xfd5bd1(0x1a5)][_0xfd5bd1(0x1bc)]('operations');const
  operatorButtons=document[_0xfd5bd1(0x1a6)](_0xfd5bd1(0x1a0)),
  clearButton=document[_0xfd5bd1(0x19f)]('C'),deleteButton=document
  [_0xfd5bd1(0x19f)](_0xfd5bd1(0x197)),pointButton=document[_0xfd5bd1
  (0x19f)]('.'),lastOperationScreen=document[_0xfd5bd1(0x19f)]('d1'),
  currentOperationScreen=document['getElementById']('d2');window
  ['addEventListener'](_0xfd5bd1(0x1ba),handleKeyboardInput),
  equalsButton[_0xfd5bd1(0x1b6)]('click',evaluate),clearButton[_0xfd5bd1
  (0x1b6)](_0xfd5bd1(0x19c),clear),deleteButton[_0xfd5bd1(0x1b6)]
  
```

## 8. Commits más recientes

- A continuación se presentan los commits más recientes.
- Commit 56934394e531bc0bb79a154a410895e6d7e01256  
Autor: JhonatanDczel [jjariasq@unsa.edu.pe](mailto:jjariasq@unsa.edu.pe)  
Fecha: Viernes 17 de mayo de 2024 -05:00  
Descripción: Ejercicio 03 completado
- Commit d8386a3942377190621a037ae0e62410330fd8ad  
Autor: JhonatanDczel [jjariasq@unsa.edu.pe](mailto:jjariasq@unsa.edu.pe)  
Fecha: Viernes 17 de mayo de 2024 -05:00  
Descripción: Ejercicio 02 completo
- Commit ccec8d074a312d17012a3c76eb4a54b8277b27ae  
Autor: JhonatanDczel [jjariasq@unsa.edu.pe](mailto:jjariasq@unsa.edu.pe)  
Fecha: Viernes 17 de mayo de 2024 -05:00  
Descripción: Reestructurando el directorio
- Commit f276cd63e77e35bfe0ba970503ab5325766c1772  
Autor: JhonatanDczel [jjariasq@unsa.edu.pe](mailto:jjariasq@unsa.edu.pe)  
Fecha: Viernes 17 de mayo de 2024 -05:00  
Descripción: Haciendo el teclado random
- Commit 2de93beb630cc8a0e1327950dfa71d4a51ce7273  
Autor: JhonatanDczel [jjariasq@unsa.edu.pe](mailto:jjariasq@unsa.edu.pe)  
Fecha: Viernes 17 de mayo de 2024 -05:00  
Descripción: Termina ejercicio 03
- Commit 523e2c994f290bb09876b5fead519411a367168  
Autor: JhonatanDczel [jjariasq@unsa.edu.pe](mailto:jjariasq@unsa.edu.pe)  
Fecha: Viernes 17 de mayo de 2024 -05:00  
Descripción: Nav completo

## 9. Rúbricas

### 9.1. Rúbrica para el contenido del Informe y demostración

- El alumno debe marcar o dejar en blanco en celdas de la columna **Checklist** si cumple con el ítem correspondiente.
- El alumno debe autocalificarse en la columna **Estudiante** de acuerdo a la siguiente tabla:

Tabla 1: Niveles de desempeño

	Nivel			
Puntos	Insatisfactorio 25 %	En Proceso 50 %	Satisfactorio 75 %	Sobresaliente 100 %
<b>2.0</b>	0.5	1.0	1.5	2.0
<b>4.0</b>	1.0	2.0	3.0	4.0

Tabla 2: Rúbrica para contenido del Informe y demostración

	Contenido y demostración	Puntos	Checklist	Estudiante	Profesor
<b>1. GitHub</b>	Repositorio se pudo clonar y se evidencia la estructura adecuada para revisar los entregables. (Se descontará puntos por error o omisión)	4	X	4	
<b>2. Commits</b>	Hay porciones de código fuente asociado a los commits planificados con explicaciones detalladas. (El profesor puede preguntar para refrendar calificación).	4	X	4	
<b>3. Ejecución</b>	Se incluyen comandos para ejecuciones y pruebas del código fuente explicadas gradualmente que permitirían replicar el proyecto. (Se descontará puntos por cada omisión)	4	X	1.5	
<b>4. Pregunta</b>	Se responde con completitud a la pregunta formulada en la tarea. (El profesor puede preguntar para refrendar calificación)	2	X	2	
<b>5. Ortografía</b>	El documento no muestra errores ortográficos. (Se descontará puntos por error encontrado)	2	X	1.5	
<b>6. Madurez</b>	El Informe muestra de manera general una evolución de la madurez del código fuente con explicaciones puntuales pero precisas, agregando diagramas generados a partir del código fuente y refleja un acabado impecable. (El profesor puede preguntar para refrendar calificación).	4	X	3	
<b>Total</b>		20		16	