

Laboratorio 03
Tema: JavaScript

Estudiante	Escuela	Asignatura
Donny Moises Mara Mamani dmara@ unas.edu.pe	Escuela Profesional de Ingeniería de Sistemas	Programación Web Semestre: III Código: 1702122

Laboratorio	Tema	Duración
03	JavaScript	06 horas













Semestre académico	Fecha de inicio	Fecha de entrega
2024 - A	13 Mayo 2024	17 Mayo 2024

1. Tarea

1.1. Descripción

- Ejercicio 01: Cree un teclado random para banca por internet.

Commits:

diseño de la pagina	6564c69	 
diseñando la pagina web	d8f66a6	 
generando los numero para el teclado pero aleatorios	54d7631	 
generando numero para el teclado virtual	e098217	 
Html listo	d298eda	 
logos en el html	f5189b2	 

Codigo:

```
document.addEventListener("DOMContentLoaded", () => {
  const headerDiv = document.createElement('div');
  headerDiv.className = 'header';

  const logoMultiRed = document.createElement('img');
  logoMultiRed.src = 'img/multiRed_logo.jpg';
  logoMultiRed.alt = 'Multi Red Virtual';
  logoMultiRed.className = 'multired_logo';

  const logoBanco = document.createElement('img');
  logoBanco.src = 'img/logo-banco_de_la_nacion.png';
  logoBanco.alt = 'Banco De La Nacion';
  logoBanco.className = 'banco_logo';

  headerDiv.appendChild(logoMultiRed);
  headerDiv.appendChild(logoBanco);
});
```

Se crea un elemento <div> con la clase header.

Se crean dos elementos para los logotipos de "Multi Red Virtual" y "Banco De La Nacion".

Los logotipos se configuran con sus respectivas rutas de imagen y texto alternativo.

Los logotipos se agregan como hijos del elemento <div> del encabezado.

Creación del encabezado (headerDiv):

```
const paragraph = document.createElement('p');  
paragraph.innerHTML = "<i class='bx bxs-lock'></i>USTED SE ENCUENTRA EN UNA ZONA SEGURA";  
  
const containerDiv = document.createElement('div');  
containerDiv.className = 'container';  
  
const formContainerDiv = document.createElement('div');  
formContainerDiv.className = 'container-formulario';  
  
const form = document.createElement('form');  
form.id = 'formulario';  
form.action = '';
```

Se crea un elemento `<p>` con un mensaje de texto dentro que indica que el usuario se encuentra en una zona segura.

El mensaje está marcado como contenido HTML, lo que permite la inserción de un ícono de cerradura.

Creación de contenedores (containerDiv y formContainerDiv):

Se crean dos elementos `<div>` con las clases `container` y `container-formulario` respectivamente.

Creación del formulario (form):

Se crea un formulario con el id 'formulario' y sin atributo action.

Creación de elementos del formulario:

```
const seleccionDiv = document.createElement('div');  
seleccionDiv.className = 'seleccion-banco';  
  
const labelSeleccion = document.createElement('label');  
labelSeleccion.htmlFor = 'seleccion';  
labelSeleccion.textContent = 'Seleccione:';  
  
const selectSeleccion = document.createElement('select');  
selectSeleccion.id = 'seleccion';  
  
const optionSeleccion = document.createElement('option');  
optionSeleccion.value = 'multired-global-debito';  
optionSeleccion.textContent = 'Multired Global Débito';  
  
selectSeleccion.appendChild(optionSeleccion);  
seleccionDiv.appendChild(labelSeleccion);  
seleccionDiv.appendChild(selectSeleccion);
```

Se crean elementos de formulario como `<select>`, `<input>`, `<label>`, etc., para recopilar información del usuario como el tipo de tarjeta, número de tarjeta, tipo de documento, número de documento, clave de acceso, etc.

Se agregan a sus respectivos contenedores dentro del formulario.

Configuración del teclado virtual (tecladoVirtualDiv):

```
const tecladoDiv = document.createElement('div');
tecladoDiv.className = 'teclado-banca';

const labelTeclado = document.createElement('label');
labelTeclado.htmlFor = 'clave';
labelTeclado.textContent = 'Ingrese su clave en el Teclado Virtual:';

const tecladoVirtualDiv = document.createElement('div');
tecladoVirtualDiv.id = 'teclado';

tecladoDiv.appendChild(labelTeclado);
tecladoDiv.appendChild(tecladoVirtualDiv);
```

Se crea un elemento <div> que actúa como el teclado virtual.

Configuración del botón de envío (submitButton):

```
const submitButton = document.createElement('button');
submitButton.type = 'submit';
submitButton.textContent = 'INGRESAR';

form.appendChild(seleccionDiv);
form.appendChild(tarjetaDiv);
form.appendChild(tipoDocumentoDiv);
form.appendChild(tecladoDiv);
form.appendChild(claveAccesoDiv);
form.appendChild(submitButton);

formContainerDiv.appendChild(form);

containerDiv.appendChild(formContainerDiv);

document.body.appendChild(headerDiv);
document.body.appendChild(paragraph);
document.body.appendChild(containerDiv);
```

Se crea un botón de tipo 'submit' con el texto "INGRESAR".

Agregación de elementos al DOM:

Finalmente, todos los elementos creados se agregan al DOM, primero el encabezado, luego el párrafo, y finalmente el contenedor del formulario.

```
document.addEventListener('DOMContentLoaded', (event) => {
  const formulario = document.getElementById('formulario');
  const teclado = document.getElementById('teclado');
  const clave = document.getElementById('clave');
```

Esta parte del código escucha el evento 'DOMContentLoaded', que se dispara cuando el DOM ha sido completamente cargado. Luego, obtiene referencias a tres elementos del DOM: el formulario, el contenedor del teclado virtual y el campo de entrada de la clave de acceso.

```
function generarNumerosRandom() {
  const numbers = Array.from({ length: 10 }, (_, i) => i);
  for (let i = numbers.length - 1; i > 0; i--) {
    const j = Math.floor(Math.random() * (i + 1));
    [numbers[i], numbers[j]] = [numbers[j], numbers[i]];
  }
  return numbers;
}
```

Esta función genera una matriz de números aleatorios del 0 al 9 utilizando el algoritmo de Fisher-Yates para mezclar aleatoriamente los números en la matriz.

```
function crearTeclado() {
  const numeros = generarNumerosRandom();
  teclado.innerHTML = '';

  numeros.forEach(number => {
    const button = document.createElement('button');
    button.type = 'button';
    button.textContent = number;
    button.addEventListener('click', (event) => {
      event.preventDefault();
      if (clave.value.length < 6) {
        clave.value += number;
      }
    });
    teclado.appendChild(button);
  });

  const clearButton = document.createElement('button');
```

Esta función crea dinámicamente los botones del teclado virtual y los agrega al contenedor del teclado. Además, agrega un evento de clic a cada botón para que al hacer clic, se agregue el número correspondiente al campo de entrada de la clave. También agrega un botón 'Limpiar' que borra el contenido del campo de la clave.

Prueba:

- Ejercicio 02: Cree una calculadora básica como la de los sistemas operativos, que pueda utilizar la función `eval()` y que guarde todos las operaciones en una pila. Mostrar la pila al pie de la página web.

Commits:

creando el estilo	MMaraP committed 2 hours ago	98152ae	📄	↔
creando el index	MMaraP committed 2 hours ago	d91de20	📄	↔
aumentado la pila a la calculadora	MMaraP committed 2 hours ago	5e88540	📄	↔
calculadora terminada	MMaraP committed 2 hours ago	ba86ef5	📄	↔
logica para la calculadora	MMaraP committed 3 hours ago	fbe0995	📄	↔

Codigo:

```
document.addEventListener("DOMContentLoaded", () => {

    const programaDiv = document.createElement('div');
    programaDiv.id = 'programa';

    const form = document.createElement('form');
    form.name = 'calculador';

    const inputDisplay = document.createElement('input');
    inputDisplay.type = 'text';
    inputDisplay.name = 'respuesta';
    inputDisplay.value = '';
    inputDisplay.autocomplete = 'off';
    inputDisplay.readOnly = true;
    form.appendChild(inputDisplay);
});
```

Este fragmento de código establece una escucha para el evento 'DOMContentLoaded', el cual se dispara cuando el DOM ha sido completamente cargado. Se crean elementos del DOM como un contenedor principal (programaDiv), un formulario (form) y un campo de entrada (inputDisplay) para

mostrar la respuesta de la calculadora.

```
for (let i = 0; i < valores.length; i += 4) {  
  const filaDiv = document.createElement('div');  
  filaDiv.className = 'fila';  
  
  for (let j = 0; j < 4; j++) {  
    const boton = valores[i + j];  
    const inputButton = document.createElement('input');  
    inputButton.type = boton.type || 'button';  
    inputButton.value = boton.value;  
    inputButton.className = boton.class;  
  
    if (boton.value === 'C') {  
      inputButton.onclick = function() { limpiar(); };  
    } else if (boton.value === '=') {  
      inputButton.onclick = function() { resultado(); };  
    } else {
```

Este bloque de código crea dinámicamente los botones de la calculadora y los agrega al formulario.

Se itera sobre un conjunto predefinido de valores de botones, creando filas de botones.

Se asignan eventos a los botones según su valor. Para el botón 'C' se asigna la función limpiar(), para el botón '=' se asigna resultado(), y para los demás botones se asigna agregarValor() con el valor correspondiente.

```
let operacionActual = '';  
let pila = [];  
  
function agregarValor(value) {  
  operacionActual += value;  
  document.calculador.respuesta.value = operacionActual;  
}  
  
function limpiar() {  
  operacionActual = '';  
  document.calculador.respuesta.value = '';  
}
```

operacionActual: Almacena la operación actual que se está realizando en la calculadora.

pila: Almacena el historial de operaciones realizadas.

agregarValor(): Esta función agrega un valor al final de la operación actual y actualiza el campo de entrada de la calculadora con la operación actualizada.

Limpiar(): Esta función restablece la operación actual y limpia el campo de entrada de la calculadora.

```
function resultado() {  
  try {  
    let resultado = eval(operacionActual);  
    pila.push(operacionActual + ' = ' + resultado);  
    actualizarPila();  
    operacionActual = '';  
    document.calculador.respuesta.value = resultado;  
  } catch (e) {  
    document.calculador.respuesta.value = 'Error';  
  }  
}  
  
function actualizarPila() {  
  let listaPila = document.getElementById('lista-pila');  
  listaPila.innerHTML = '';  
  pila.forEach(function (operacion) {  
    let li = document.createElement('li');  
    li.textContent = operacion;  
    listaPila.appendChild(li);  
  });  
}
```

Resultado(): Esta función calcula el resultado de la operación actual utilizando la función eval().

Agrega la operación actual y su resultado a la pila de operaciones.

Llama a la función actualizarPila() para reflejar los cambios en la lista de operaciones.

Limpia la operación actual y muestra el resultado en el campo de entrada de la calculadora.

actualizarPila(): Esta función actualiza la lista de operaciones en la interfaz de usuario.

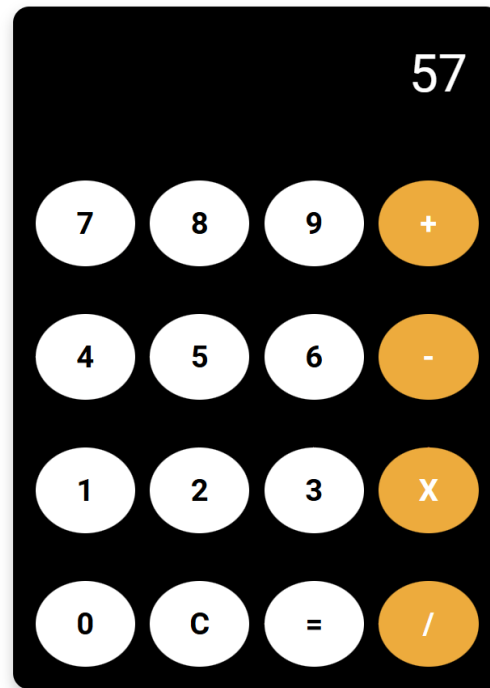
Borra el contenido anterior de la lista.

Itera sobre todas las operaciones en la pila y agrega cada una como un elemento de lista () a la lista de operaciones en el DOM.

PRUEBA:

Pila de operaciones:

$9+9 = 18$
 $3*6 = 18$
 $3/5 = 0.6$
 $9+6*8 = 57$



Ejercicio 03:

COMMITTS:



CODIGO:

```
document.addEventListener("DOMContentLoaded", () => {  
  
    const titulo = document.createElement('h1');  
    titulo.textContent = 'Juego del Ahorcado';  
    document.body.appendChild(titulo);  
  
    const canvas = document.createElement('canvas');  
    canvas.id = 'cuadro';  
    canvas.width = 400;  
    canvas.height = 400;  
    document.body.appendChild(canvas);  
});
```

Titulo: Se crea un elemento h1 que contiene el título del juego.

El título se establece con el texto "Juego del Ahorcado".

El elemento h1 se agrega como hijo del cuerpo del documento.

Canvas: Se crea un elemento canvas que servirá como área de dibujo para el juego.

Se establecen propiedades como el ID y las dimensiones del canvas.

El canvas se agrega como hijo del cuerpo del documento.

```
const divInput = document.createElement('div');  
  
const label = document.createElement('label');  
label.htmlFor = 'letra';  
label.textContent = 'Introduce una letra: ';  
divInput.appendChild(label);  
  
const input = document.createElement('input');  
input.type = 'text';  
input.id = 'letra';  
input.maxLength = 1;  
input.onkeypress = function(event) { teclaPresionada(event); };  
divInput.appendChild(input);  
  
document.body.appendChild(divInput);
```

Se crea un contenedor div para el campo de entrada y su etiqueta asociada.

Se crea una etiqueta label que indica al usuario qué debe ingresar.

Se crea un campo de entrada input de tipo texto para que el usuario ingrese una letra.

Se limita la longitud máxima del campo de entrada a 1 carácter.

Se agrega un evento onkeypress que llama a la función teclaPresionada() cuando se presiona una tecla en el campo de entrada.

El contenedor div con la etiqueta y el campo de entrada se agrega como hijo del cuerpo del documento.

```
const palabraParrafo = document.createElement('p');
palabraParrafo.id = 'palabra';
document.body.appendChild(palabraParrafo);

const mensajeParrafo = document.createElement('p');
mensajeParrafo.id = 'mensaje';
document.body.appendChild(mensajeParrafo);
```

Se crean dos elementos p para mostrar la palabra oculta y los mensajes del juego.

Se establece un ID único para cada elemento.

Ambos elementos se agregan como hijos del cuerpo del documento.

```
const script = document.createElement('script');
script.src = 'js/script.js';
document.body.appendChild(script);
```

Se crea un elemento script que carga el archivo JavaScript que contiene la lógica del juego.

Se establece la ruta del archivo JavaScript en la propiedad src.

El script se agrega como hijo del cuerpo del documento para que se ejecute y maneje la funcionalidad del juego.

```
const palabras = ["JAVASCRIPT", "PROGRAMACION", "CANVAS", "AHORCADO"];
let palabra = palabras[Math.floor(Math.random() * palabras.length)];
let palabraAdivinada = "_".repeat(palabra.length);
let intentos = 0;
const maxIntentos = 10;
let juegoTerminado = false;
```

Se define un array de palabras.

Se selecciona aleatoriamente una palabra del array.

Se inicializa una variable para representar la palabra a adivinar, inicialmente mostrando guiones bajos.

Se inicializa un contador de intentos y se establece el máximo de intentos permitidos.

Se establece la palabra a adivinar en el elemento con el ID 'palabra' en el DOM.

```
function teclaPresionada(evento) {  
    if (evento.key === 'Enter') {  
        adivinarLetra();  
    }  
}
```

Esta función maneja el evento de presionar una tecla, permitiendo adivinar la letra cuando se presiona la tecla "Enter".

```
function adivinarLetra() {  
    if (juegoTerminado) {  
        return;  
    }  
  
    const inputLetra = document.getElementById('letra');  
    const letra = inputLetra.value.toUpperCase();  
    inputLetra.value = '';  
    if (!letra || !/^[A-ZÑ$]/.test(letra)) {  
        alert("Introduce una letra válida");  
        return;  
    }  
  
    if (palabra.includes(letra)) {  
        let nuevaPalabraAdivinada = '';  
        for (let i = 0; i < palabra.length; i++) {  
            if (palabra[i] === letra) {  
                nuevaPalabraAdivinada += letra;  
            } else {  
                nuevaPalabraAdivinada += palabraAdivinada[i];  
            }  
        }  
        palabraAdivinada = nuevaPalabraAdivinada;  
        document.getElementById('palabra').textContent = palabraAdivinada;  
    } else {  
        intentos++;  
        dibujarAhorcado(intentos);  
    }  
}
```

Esta función se encarga de adivinar la letra ingresada por el usuario.

Verifica si el juego ha terminado antes de realizar cualquier acción.

Obtiene la letra ingresada por el usuario y la convierte a mayúsculas.

Valida si la letra ingresada es válida (una letra del alfabeto).

Comprueba si la letra está presente en la palabra a adivinar.

Si la letra está presente, actualiza la palabra adivinada con la letra correcta.

Si la letra no está presente, incrementa el contador de intentos y dibuja una parte del ahorcado.

Finalmente, comprueba si el juego ha terminado (ya sea ganando o perdiendo).

```
function dibujarAhorcado(intentos) {  
  const cuadro = document.getElementById('cuadro');  
  const dibujar = cuadro.getContext('2d');  
  
  if (intentos === 1) {  
    dibujar.beginPath();  
    dibujar.moveTo(50, 350);  
    dibujar.lineTo(150, 350);  
    dibujar.stroke();  
  } else if (intentos === 2) {  
    dibujar.beginPath();  
    dibujar.moveTo(100, 350);  
    dibujar.lineTo(100, 50);  
    dibujar.stroke();  
  } else if (intentos === 3) {
```

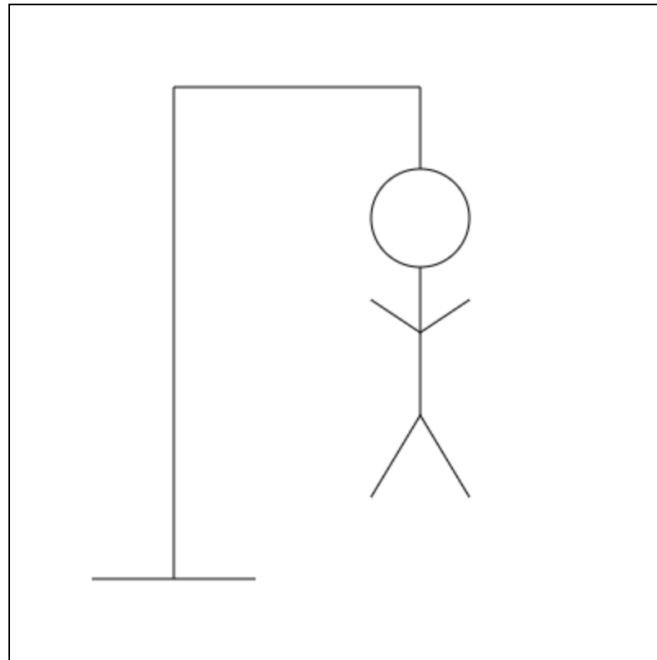
Esta función se encarga de dibujar diferentes partes del ahorcado en el canvas según el número de intentos realizados.

Utiliza el contexto 2D del canvas para realizar los dibujos.

Dibuja partes del ahorcado progresivamente según el número de intentos realizados.

PRUEBA:

Juego del Ahorcado



Introduce una letra:

CAN_AS

TE AHORCASTE!!!. La palabra era: CANVAS

1.2. Pregunta

- Explique una herramienta para ofuzcar código JavaScript.

JavaScript Obfuscator: JavaScript Obfuscator es una herramienta que oculta y protege el código JavaScript al convertirlo en una forma ilegible para los humanos, mientras sigue siendo funcional para los navegadores web. Utiliza técnicas avanzadas para cambiar nombres de variables y funciones, elimina comentarios y espacios en blanco, y protege contra la ingeniería inversa. Se integra fácilmente en los flujos de trabajo de desarrollo y ayuda a mejorar la seguridad y la integridad del código en aplicaciones web. Un obfuscador de JavaScript es una herramienta que transforma el código fuente de JavaScript en una forma ofuscada o encriptada. El objetivo principal de un obfuscador es dificultar la comprensión y el análisis del código por parte de terceros, sin afectar su funcionalidad. La ofuscación del código implica cambiar los nombres de variables y funciones, reemplazar cadenas legibles por cadenas codificadas y alterar el flujo de control del programa.

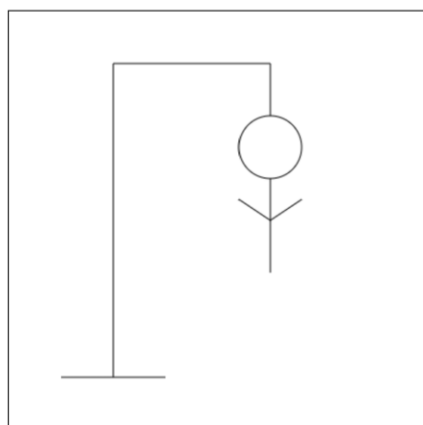
Muestre un ejemplo de su uso en uno de los ejercicios de la

Copy & Paste JavaScript Code	Upload JavaScript File	Output
<pre> 1 const palabras = ["JAVASCRIPT", "PROGRAMACION", "CANVAS", "AHORCADO"]; 2 let palabra = palabras[Math.floor(Math.random() * palabras.length)]; 3 let palabraAdivinada = "_".repeat(palabra.length); 4 let intentos = 0; 5 const maxIntentos = 10; 6 let juegoTerminado = false; 7 8 document.getElementById('palabra').textContent = palabraAdivinada; </pre>		
<div>Obfuscate</div>		
Copy & Paste JavaScript Code	Upload JavaScript File	Output
<pre> const _0x5bd5d5=_0x2211;function _0x2211(_0x48c137,_0x1d0177){const _0x1ebcbb=_0x1ebc();return _0x2211=function(_0x2211c3,_0x4568b8){_0x2211c3=_0x2211c3-0x190;let _0x35ea79=_0x1ebcbb[_0x2211c3];return _0x35ea79;}_0x2211(_0x48c137,_0x1d0177);}(function(_0x2174aa,_0x44c7a0){const _0x20f0d8=_0x2211,_0x9360cb=_0x2174aa;while(![]){try{const _0x99cad7=-parseInt(_0x20f0d8(0x193))/0x1+-parseInt(_0x20f0d8(0x1a4))/0x2*(parseInt(_0x20f0d8(0x197))/0x3+-parseInt(_0x20f0d8(0x1a2))/0x4*(parseInt(_0x20f0d8(0x1a9))/0x5)+parseInt(_0x20f0d8(0x1a5))/0x6+-parseInt(_0x20f0d8(0x196))/0x7+parseInt(_0x20f0d8(0x19c))/0x8+parseInt(_0x20f0d8(0x1b2))/0x9;if(_0x99cad7===_0x44c7a0)break;else _0x9360cb["push"](_0x9360cb["shift"]());}catch(_0x778d0a){_0x9360cb["push"](_0x9360cb["shift"]());}}}_0x1ebc,0x55cef));const palabras=["JAVASCRIPT",_0x5bd5d5(0x198),_0x5bd5d5(0x1a1),_0x5bd5d5(0x1af)];let palabra=palabras[Math[_0x5bd5d5(0x1a7)]](Math[_0x5bd5d5(0x194)] </pre>		

```

const script = document.createElement('script');
script.src = 'js/script_obfuscated.js';
document.body.appendChild(script);
    
```

Juego del Ahorcado



Introduce una letra:

JAVASCRIPT

¡Felicidades! ¡Ganaste!

1.3. Entregables

- <https://github.com/MMaraP/pw2-24a>

```
C:\.
├── estructura.txt
├── pw2-24a
│   ├── .gitignore
│   ├── Laboratorio_03
│   │   ├── estructura.txt
│   │   ├── Ahorcado
│   │   │   ├── index.html
│   │   │   ├── css
│   │   │   │   ├── styles.css
│   │   │   └── js
│   │   │       ├── html.js
│   │   │       ├── script.js
│   │   │       └── script_obfuscated.js
│   ├── Calculadora
│   │   ├── index.html
│   │   ├── css
│   │   │   ├── styles.css
│   │   └── js
│   │       ├── html.js
│   │       └── script.js
│   ├── Teclado_Random
│   │   ├── index.html
│   │   ├── css
│   │   │   ├── styles.css
│   │   ├── img
│   │   │   ├── logo-banco_de_la_nacion.png
│   │   │   └── multiRed_Logo.jpg
│   └── script
│       ├── html.js
│       └── script.js
└── Laboratorio_04
```


2. Rúbricas

Tabla 3: Rúbrica para contenido del Informe y evidencias

Contenido y demostración		Puntos	Checklist	Estudiante	Profesor
1. GitHub	Repositorio se pudo clonar y se evidencia la estructura adecuada para revisar los entregables. (Se descontará puntos por error o omisión)	4	✓	3	
2. Commits	Hay porciones de código fuente asociado a los commits planificados con explicaciones detalladas. (El profesor puede preguntar para refrendar calificación).	4	✓	3	
3. Ejecución	Se incluyen comandos para ejecuciones y pruebas del código fuente explicadas gradualmente que permitirían replicar el proyecto. (Se descontará puntos por cada omisión)	4	✓	2	
4. Pregunta	Se responde con completitud a la pregunta formulada en la tarea. (El profesor puede preguntar para refrendar calificación).	2	✓	1	
7. Ortografía	El documento no muestra errores ortográficos. (Se descontará puntos por error encontrado)	2	✓	2	

8. Madurez	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	✓	3	
Total		20		14	

3. Referencias

- <https://developer.mozilla.org/es/docs/Web/JavaScript/Guide>
- <https://www.w3schools.com/jsref/>
- <https://github.com/>
- <https://www.mauriciodeveloper.com/post/obfuscator-javascript-como-ocultar-y-proteger-tu-codigo-fuente-927>
- <https://obfuscator.io/>
- <https://www.cloudflare.com/es-es/learning/performance/why-minify-javascript-code/>