

Informe de Laboratorio 03

Tema: Javascript

Nota

Estudiante	Escuela	Asignatura
Kevin Andree Llacma Quispe kllacma@unsa.edu.pe	Escuela Profesional de Ingeniería de Sistemas	Programación web 2 Semestre: I Código: 20200585

Laboratorio	Tema	Duración
03	Javascript	04 horas

Semestre académico	Fecha de inicio	Fecha de entrega
2024 - A	-	-

Programación Web

Laboratorio 03

Tema: JavaScript

22 de mayo de 2024

1. Marco teórico

1.1. Vim

Vim es un editor de texto altamente configurable creado para hacer que la creación y el cambio de cualquier tipo de texto sean muy eficientes. Se incluye como "vi" en la mayoría de los sistemas UNIX y con Apple OS X.

Vim es muy estable y se desarrolla continuamente para mejorar aún más. Entre sus características se encuentran:

- árbol de deshacer persistente de varios niveles
- amplio sistema de complementos
- soporte para cientos de lenguajes de programación y formatos de archivo
- poderosa búsqueda y reemplazo
- se integra con muchas herramientas

```
$ sudo apt-get install vim # Vim en GNU/Linux  
$ brew install macvim # Vim en MacOSX
```

1.2. Visual Studio Code

Visual Studio Code es un editor de código fuente ligero pero potente que se ejecuta en su escritorio y está disponible para Windows, macOS y Linux. Viene con soporte integrado para JavaScript, TypeScript y Node.js y tiene un rico ecosistema de extensiones para otros lenguajes y tiempos de ejecución (como C++, C, Java, Python, PHP, Go, .NET).

Videotutoriales: <https://code.visualstudio.com/docs/getstarted/introvideos>

Descarga: <https://code.visualstudio.com/Download>

2. Desarrollo del lab

2.1. Ejercicio 1

Cree un teclado random para banca por internet.

Para este ejercicio de uso un html basico que se dio. Se agrego style para los botones de los numeros.

.teclado-container: Configura el contenedor del teclado para usar un diseño de cuadrícula de 3 columnas.

.tecla: Estiliza cada tecla del teclado.

.input-display: Estiliza el campo de entrada donde se muestran los números seleccionados.

```
<style>
  .teclado-container {
    display: grid;
    grid-template-columns: repeat(3, 1fr);
    gap: 10px;
    max-width: 200px;
    margin: 50px auto;
  }
  .tecla {
    padding: 20px;
    font-size: 18px;
    text-align: center;
    border: 1px solid #ccc;
    border-radius: 5px;
    cursor: pointer;
    user-select: none;
  }
  .input-display {
    width: 100%;
    padding: 10px;
    margin-bottom: 20px;
    font-size: 18px;
    text-align: center;
  }
</style>
```

-<div id="teclado-container">/div>: Un contenedor para las teclas del teclado numérico.

-<input type="text" id="input-display" class="input-display" readonly>: Un campo de entrada que muestra los números seleccionados, es de solo lectura.

-<script src="teclado.js">/script>: Incluye el archivo JavaScript que contiene la lógica del teclado.

```
<body>
<div id="teclado-container"></div>
<input type="text" id="input-display" class="input-display" readonly>

<script src="teclado.js"></script>
</body>
```

Ahora en JavaScript tenemos

document.addEventListener('DOMContentLoaded', function()): Asegura que el código JavaScript se ejecute solo después de que todo el contenido HTML haya sido completamente cargado.

const tecladoContainer = document.getElementById('teclado-container'); Selecciona el contenedor donde se crearán las teclas.

const inputDisplay = document.getElementById('input-display'); Selecciona el campo de entrada donde se mostrarán los números seleccionados.

const numeros = Array.from(length: 10 , (i) => i): Crea un arreglo con los números del 0 al 9.

```
document.addEventListener('DOMContentLoaded', function() {  
  const tecladoContainer = document.getElementById('teclado-container');  
  const inputDisplay = document.getElementById('input-display');  
  const numeros = Array.from({ length: 10 }, (_, i) => i);
```

Función barajar:

Esta función toma un arreglo y lo baraja usando el algoritmo de Fisher-Yates. **function barajar(array) ...** : Recibe un arreglo y lo baraja aleatoriamente.

```
function barajar(array) {  
  for (let i = array.length - 1; i > 0; i--) {  
    const j = Math.floor(Math.random() * (i + 1));  
    [array[i], array[j]] = [array[j], array[i]];  
  }  
}
```

Función crearTeclado:

Llama a **barajar(numeros)** para mezclar los números.

Limpia el contenedor del teclado con **tecladoContainer.innerHTML = ''**;

Para cada número en el arreglo barajado:

- Crea un nuevo elemento div para representar una tecla.
- Asigna la clase **tecla** al div.
- Establece el contenido de texto del div al número actual.
- Agrega un evento click al div para que, cuando se haga clic, el número se agregue al valor del **inputDisplay**.
- Añade el div al contenedor del teclado.

```
function crearTeclado() {
  barajar(numeros);
  tecladoContainer.innerHTML = '';
  numeros.forEach(numero => {
    const tecla = document.createElement('div');
    tecla.className = 'tecla';
    tecla.textContent = numero;
    tecla.addEventListener('click', () => {
      inputDisplay.value += numero;
    });
    tecladoContainer.appendChild(tecla);
  });
}
```

Finalmente se inicializa

```
crearTeclado();
```

Algunas pruebas de la aleatoriedad de los numeros y el campo se encuentra abajo

4
9
2
0
8
1
5
7
6
3
20157

6
2
9
4
3
5
7
8
0
1
3580

2.2. Ejercicio 2

Para este ejercicio se usó el mismo formato del anterior ejercicio (un html básico con estilos) y su lógica JavaScript

En el body tenemos los botones de la calculadora

-<div class=calculadora>: Contenedor de la calculadora que incluye los botones y la pantalla.

-<div id=pila-operacionesclass=pila-operaciones>/div>: Contenedor para mostrar el historial de operaciones.

-<script src=calculadora.js>/script>: Archivo JavaScript que contiene la lógica.

```
<body>
<div class="calculadora">
  <input type="text" id="pantalla" class="pantalla" readonly>
  <div class="boton">7</div>
  <div class="boton">8</div>
  <div class="boton">9</div>
  <div class="boton">/</div>
  <div class="boton">4</div>
  <div class="boton">5</div>
  <div class="boton">6</div>
  <div class="boton">*</div>
  <div class="boton">1</div>
  <div class="boton">2</div>
  <div class="boton">3</div>
  <div class="boton">-</div>
  <div class="boton">0</div>
  <div class="boton">.</div>
  <div class="boton">+</div>
  <div class="boton">=</div>
</div>
<div id="pila-operaciones" class="pila-operaciones"></div>

<script src="calculadora.js"></script>
</body>
```

Para la lógica del ejercicio tenemos :

`const pantalla = document.getElementById('pantalla');` Selecciona el campo de entrada donde se mostrarán las expresiones y resultados.

`const botones = document.querySelectorAll('.boton');` Selecciona todos los botones de la calculadora.

`const pilaOperaciones = document.getElementById('pila-operaciones');` Selecciona el contenedor donde se mostrará el historial de operaciones.

```
document.addEventListener('DOMContentLoaded', function() {
  const pantalla = document.getElementById('pantalla');
  const botones = document.querySelectorAll('.boton');
  const pilaOperaciones = document.getElementById('pila-operaciones');
  let operacionesPila = [];
```

Eventos de clic a los botones

`botones.forEach(boton => ...);` Itera sobre todos los botones y les agrega un evento de clic.

`boton.addEventListener('click', () => ...);` Define lo que sucede cuando se hace clic en un botón.

Manejo del clic en los botones:

`const valor = boton.textContent;` Obtiene el valor (texto) del botón clicado.

Si el botón es =:

- Evalúa la expresión en la pantalla usando `eval()`.

- Intenta evaluar la expresión y manejar posibles errores.
- `const resultado = eval(pantalla.value);` Evalúa la expresión matemática en la pantalla.
- `operacionesPila.push(pantalla.value + resultado);` Agrega la operación y el resultado a la pila.
- `pantalla.value = resultado;` Muestra el resultado en la pantalla.
- `actualizarPila();` Actualiza la visualización del historial de operaciones.

Si el botón es C:

- Limpia la pantalla y borra el contenido de la pantalla.

Para otros botones:

- Agrega el valor del botón a la expresión en la pantalla.

```
botones.forEach(boton => {
  boton.addEventListener('click', () => {
    const valor = boton.textContent;

    if (valor === '=') {
      try {
        const resultado = eval(pantalla.value);
        operacionesPila.push(`${pantalla.value} = ${resultado}`);
        pantalla.value = resultado;
        actualizarPila();
      } catch (e) {
        pantalla.value = 'Error';
      }
    } else if (valor === 'C') {
      pantalla.value = '';
    } else {
      pantalla.value += valor;
    }
  });
});
```

Función `actualizarPila`:

`function actualizarPila():` Actualiza la visualización del historial de operaciones.
`pilaOperaciones.innerHTML = '<h3>Historial de Operaciones;</h3>';` Limpia el contenido actual del historial.
`operacionesPila.forEach(operacion => {):` Itera sobre el historial de operaciones.
`const operacionElement = document.createElement('div');` Crea un nuevo elemento `div` para cada operación.
`operacionElement.textContent = operacion;` Establece el texto del `div` con la operación.
`pilaOperaciones.appendChild(operacionElement);` Agrega el `div` al contenedor del historial.

Algunas pruebas con su historial de operaciones

file:///C:/Users/ab_03/Desktop/2/index.html

126

7

8

9

/

4

5

6

*

1

2

3

-

0

.

+

=

C

Historial de Operaciones:

14:7 = 2

2*21 = 46

48:5 = 41

41:55 = 126

Para este ejercicio igualmente un html basico como en anteriores ejercicios esta vez se usa canvas

```
<body>
<h1>El Ahorcado</h1>
<canvas id="canvas" width="200" height="200"></canvas>
<div class="hidden-word" id="hidden-word"></div>
<div class="letter-buttons" id="letter-buttons"></div>

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

En la lógica tenemos

`const ctx = canvas.getContext('2d');` Obtiene el contexto de dibujo en 2D del canvas.
`const hiddenWordElement = document.getElementById('hidden-word');` Selecciona el elemento que muestra la palabra oculta.
`const letterButtonsElement = document.getElementById('letter-buttons');` Selecciona el contenedor de botones de letras.

```
document.addEventListener('DOMContentLoaded', function() {
  const canvas = document.getElementById('canvas');
  const ctx = canvas.getContext('2d');
  const hiddenWordElement = document.getElementById('hidden-word');
  const letterButtonsElement = document.getElementById('letter-buttons');
```

- `let palabraSeleccionada = palabras[Math.floor(Math.random() * palabras.length)];` Selecciona una palabra aleatoria del arreglo de palabras.
- `let palabraOculta = Array(palabraSeleccionada.length).fill('');` Inicializa la palabra oculta con guiones bajos, uno por cada letra.
- `const maxIntentos = 6;` Define el número máximo de intentos permitidos.

```
const palabras = ['PAUCARPATA', 'SABANDIA', 'CHARACATO', 'HUNTER', 'CAYMA', 'MIRAFLORES', 'SOCABAYA', 'UCHUMAYO'];
let palabraSeleccionada = palabras[Math.floor(Math.random() * palabras.length)];
let palabraOculta = Array(palabraSeleccionada.length).fill('_');
let intentos = 0;
const maxIntentos = 6;
```

Función `dibujarAhorcado(intentos)`

Dibuja el progreso del ahorcado en el canvas basado en el número de intentos.
 Base y poste vertical

```
if (intentos > 0) {  
    ctx.beginPath();  
    ctx.moveTo(10, 190);  
    ctx.lineTo(190, 190);  
    ctx.stroke();  
}  
  
if (intentos > 1) {  
    ctx.beginPath();  
    ctx.moveTo(50, 190);  
    ctx.lineTo(50, 10);  
    ctx.stroke();  
}
```

Poste horizontal cuerda y cabeza etc

```
if (intentos > 2) {
    ctx.beginPath();
    ctx.moveTo(50, 10);
    ctx.lineTo(150, 10);
    ctx.stroke();
}

if (intentos > 3) {
    ctx.beginPath();
    ctx.moveTo(150, 10);
    ctx.lineTo(150, 30);
    ctx.stroke();
}

if (intentos > 4) {
    ctx.beginPath();
    ctx.arc(150, 50, 20, 0, Math.PI * 2);
    ctx.stroke();
}
```

Función actualizarPalabraOculta

Esta función actualiza la visualización de la palabra oculta en el DOM.

```
function actualizarPalabraOculta() {
    hiddenWordElement.textContent = palabraOculta.join(' ');
}
```

Función crearBotonesLetras

Esta función crea botones para cada letra del alfabetario y les asigna un evento click.

```
function crearBotonesLetras() {
  for (let i = 65; i <= 90; i++) {
    const letra = String.fromCharCode(i);
    const boton = document.createElement('button');
    boton.textContent = letra;
    boton.addEventListener('click', () => manejarAdivinanza(letra));
    letterButtonsElement.appendChild(boton);
  }
}
```

Función manejarAdivinanza

Maneja el evento cuando se adivina una letra, actualizando la palabra oculta y dibujando el ahorcado si la letra no está en la palabra.

```
function manejarAdivinanza(letra) {
  let acierto = false;
  for (let i = 0; i < palabraSeleccionada.length; i++) {
    if (palabraSeleccionada[i] === letra) {
      palabraOculta[i] = letra;
      acierto = true;
    }
  }

  if (!acierto) {
    intentos++;
    dibujarAhorcado(intentos);
  }

  actualizarPalabraOculta();
  verificarJuegoTerminado();
}
```

Función verificarJuegoTerminado

Verifica si el juego ha terminado, ya sea por adivinar toda la palabra o por alcanzar el número máximo de intentos.

```
function verificarJuegoTerminado() {
  if (palabraOculta.join('') === palabraSeleccionada) {
    alert('¡Felicidades! Has adivinado la palabra.');
```

Finalmente inicializan la visualización de la palabra oculta y crean los botones de letras cuando se carga la página.

```
actualizarPalabraOculta();  
crearBotonesLetras();
```

teclado ofuscado

```
document.addEventListener('DOMContentLoaded',function(){const _0x4b37=[  
  "\x64\x69\x76","\x63\x72\x65\x61\x74\x65\x45\x6C\x65\x6D\x65\x6E\x74",  
  "\x68\x65\x79\x62\x6F\x61\x72\x64\x43\x6F\x6E\x74\x61\x69\x6E\x65\x72",  
  "\x62\x6F\x64\x79","\x61\x70\x70\x65\x6E\x64\x43\x68\x69\x6C\x64",  
  "\x30\x32\x33\x34\x35\x36\x37\x38\x39","\x73\x70\x6C\x69\x74",  
  "\x73\x6F\x72\x74","\x6A\x6F\x69\x6E","\x6C\x65\x6E\x67\x74\x68",  
  "\x74\x65\x78\x74\x43\x6F\x6E\x74\x65\x6E\x74","\x63\x72\x65\x61\x74\x65\x45\x6C\x65\x6D\x65\x6E\x74\x4E\x53"];  
const _0x54be=[_0x4b37[0],_0x4b37[1],_0x4b37[2],_0x4b37[3],_0x4b37[4],_0x4b37[5],  
  _0x4b37[6],_0x4b37[7],_0x4b37[8],_0x4b37[9],_0x4b37[10],_0x4b37[11]];  
const container=document[_0x54be[1]](_0x54be[0]);container.id=_0x54be[2];  
document[_0x54be[3]][_0x54be[4]](container);const keys=_0x54be[5];let shuffledKeys=keys[_0x54be[6]]([])[_0x54be[7]]
```