	UNIVERSIDAD DON BOSCO FACULTAD DE INGENIERÍA ESCUELA DE COMPUTACIÓN		CICLO 02-2023
	MATERIA	LENGUAJES INTERPRETADOS EN EL CLIENTE	GUIA DE LABORATORIO N° 6
	PRÁCTICA	ESTRUCTURAS DE CONTROL Y MATRICES	

I. OBJETIVOS

Que al finalizar la practica el estudiante aprenda a:

- Dominio del uso de las estructuras de decisión.
- Dominio del uso de las sentencias repetitivas, ciclos o lazos del lenguaje JavaScript.
- Aplicación de sentencias repetitivas en la solución de problemas que requieran repetir un conjunto de instrucciones.
- Utilización de sentencias repetitivas anidadas con lenguaje JavaScript.
- Uso apropiado de sentencias de control de ciclos o lazos (break y continue).
- Utilización de matrices en la solución de problemas prácticos.
- Capacidad de definir matrices unidimensionales y multidimensionales para resolver problemas.
- Manejo de las estructuras de control repetitivas para asignar, acceder, eliminar y ordenar los elementos de una matriz.
- Destreza en el uso de funciones para facilitar el manejo de matrices en JavaScript.

II. INTRODUCCIÓN TEORICA

1. Estructuras selectivas

Las estructuras selectivas se utilizan para tomar decisiones lógicas, de ahí que se suelen denominar también estructuras de decisión o alternativas. En las estructuras selectivas se evalúa una condición y en función del resultado de la misma se realiza una opción u otra. Las condiciones se especifican usando expresiones lógicas.


1.1 Estructura if...else

La estructura más utilizada en JavaScript y en la mayoría de lenguajes de programación es la estructura if. Se emplea para tomar decisiones en función de una condición. Su definición formal es:

```
if(condicion) {
    /*Bloque de instrucciones*/
}
```

Normalmente las condiciones suelen ser del tipo "si se cumple esta condición, hazlo; si no se cumple, haz esto otro". Para este segundo tipo de decisiones, existe una variante de la estructura if llamada if...else. Su definición formal es la siguiente:

```
if (condicion) {
    /* Bloque de instrucciones
    si se cumple la condicion */
} else {
    /* Bloque de instrucciones
    si no se cumple la condicion */
}
```

	UNIVERSIDAD DON BOSCO FACULTAD DE INGENIERÍA ESCUELA DE COMPUTACIÓN		CICLO 02-2023
	MATERIA	LENGUAJES INTERPRETADOS EN EL CLIENTE	GUIA DE LABORATORIO N° 6
	PRÁCTICA	ESTRUCTURAS DE CONTROL Y MATRICES	

1.2 Estructura if...else en una línea

El operador condicional (ternario), es el único operador en JavaScript que tiene tres operandos. Este operador se usa con frecuencia como atajo para la instrucción if.

```
condicion ? expr1 : expr2
```

Si la condición es true, el operador retorna el valor de la expr1; de lo contrario, devuelve el valor de expr2.

1.3 Estructura switch


La estructura switch es muy útil cuando la condición que evaluamos puede tomar muchos valores. Si utilizamos una sentencia if...else, tendríamos que repetir la condición para los distintos valores.

```
if(dia == 1) {
    console.log("Hoy es lunes.");
} else if(dia == 2) {
    console.log("Hoy es martes.");
} else if(dia == 3) {
    console.log("Hoy es miércoles.");
} else if(dia == 4) {
    console.log("Hoy es jueves.");
} else if(dia == 5) {
    console.log("Hoy es viernes.");
} else if(dia == 6) {
    console.log("Hoy es sábado.");
} else if(dia == 0) {
    console.log("Hoy es domingo.");
}
```

En este caso es más conveniente utilizar una estructura de control de tipo [switch](#), ya que permite ahorrarnos trabajo y producir un código más limpio. Su definición formal es la siguiente:

```
switch(dia) {
    case 1: console.log("Hoy es lunes."); break;
    case 2: console.log("Hoy es martes."); break;
    case 3: console.log("Hoy es miércoles."); break;
    case 4: console.log("Hoy es jueves."); break;
    case 5: console.log("Hoy es viernes."); break;
    case 6: console.log("Hoy es sábado."); break;
    case 0: console.log("Hoy es domingo."); break;
}
```

La cláusula [case](#) no tiene por qué ser una constante, sino que puede ser una expresión al igual que en la estructura if. El comportamiento por defecto de la estructura [switch](#) es seguir evaluando el resto de cláusulas, aun cuando una de ellas haya cumplido la condición. Para evitar ese comportamiento, es necesario utilizar la sentencia [break](#) en las cláusulas que deseemos.

	UNIVERSIDAD DON BOSCO FACULTAD DE INGENIERÍA ESCUELA DE COMPUTACIÓN		CICLO 02-2023
	MATERIA	LENGUAJES INTERPRETADOS EN EL CLIENTE	GUIA DE LABORATORIO N° 6
	PRÁCTICA	ESTRUCTURAS DE CONTROL Y MATRICES	

2. Sentencias repetitivas

Las sentencias repetitivas son el medio que brindan los lenguajes de programación para poder repetir un bloque o conjunto de instrucciones más de una vez. Estas sentencias suelen llamarse lazos, ciclos o bucles. El número de veces que se repite el ciclo o lazo es controlado mediante una condición o mediante el valor de un contador. Cuando se trata de una condición, se involucra una variable cuyo valor cambia cada vez que se ejecuta el lazo. En el caso de una variable contador aumenta de valor de forma automática, cada vez que se ejecuta el lazo hasta llegar a un valor máximo definido en el contador.

JavaScript proporciona varios tipos de sentencias repetitivas o lazos, entre ellos se pueden mencionar: **for**, **while** y **do ... while**. Otras instrucciones particulares de JavaScript, relacionadas con el uso de objetos, **son for...in**, **for...of** y **with**.

2.1 Sentencia for

Permite crear un lazo o bucle que se ejecutará un número determinado de veces. Utiliza una variable contador, una condición de comparación, una instrucción de incremento (o decremento) del contador y una o más instrucciones que forman parte del cuerpo del lazo o bucle. Estas instrucciones se repetirán tantas veces hasta que la condición de comparación se evalúe como falsa.

La sintaxis de la sentencia **for** es la siguiente:

```
for (inicialización; condicion; incremento/decremento){
    //instrucción o bloque de instrucciones;
}
```

Ejemplo:

```
console.log("Conteo hacia atrás<br>");
for (var i = 10; i >= 0; i--) {
    console.log(i);
}
console.log("Fin del conteo.");
```


2.2 El bucle while

Otra forma de crear un bucle es utilizando la sentencia **while**. Funciona de forma similar al **for**, pero su estructura sintáctica es completamente diferente. La estructura básica es:

```
while (condicion) {
    //bloque de código ;
}
```

Donde, condición es cualquier expresión JavaScript válida que se evalúe a un valor booleano. El bloque de código se ejecuta mientras que la condición sea verdadera. Por ejemplo:

```
let nTotal = 35;
let con = 3;
while (con <= nTotal) {
    console.log("El contador con es igual a: " + con);
    con++;
}
```

	UNIVERSIDAD DON BOSCO FACULTAD DE INGENIERÍA ESCUELA DE COMPUTACIÓN		CICLO 02-2023
	MATERIA	LENGUAJES INTERPRETADOS EN EL CLIENTE	GUIA DE LABORATORIO N° 6
	PRÁCTICA	ESTRUCTURAS DE CONTROL Y MATRICES	

El resultado es el mismo que en el ciclo `for`, aunque su construcción sintáctica es muy diferente. El ciclo comprueba la expresión y continúa la ejecución del código mientras la evaluación sea true. El bucle while no tiene requiere que la variable de control del ciclo o lazo se modifique dentro del bloque de instrucciones para que en determinado momento se pueda detener la ejecución del lazo.

2.3 El bucle do ... while

Esta instrucción es muy similar al ciclo `while`, con la diferencia que en esta, primero se ejecutan las instrucciones y luego, se verifica la condición. Esto significa que, el bloque de instrucciones se ejecutará con seguridad, al menos, una vez. Su sintaxis es:

```
do {
    //bloque de código ;
} while (expresión_condicional);
```

La diferencia con el anterior bucle estriba en que la expresión condicional se evalúa después de ejecutar el bloque de código, lo que garantiza que al menos una vez se ha de ejecutar, aun cuando la condición sea false desde el principio.

```
let nTotal = 35;
let con = 36;
do {
    alert("El contador con es igual a: " + con);
    con++;
} while (con <= nTotal);
```

2.4 Declaración for...in

La instrucción `for...in` itera una variable especificada sobre todas las propiedades enumerables de un objeto. Para cada propiedad distinta, JavaScript ejecuta las instrucciones especificadas. Una declaración `for...in` tiene el siguiente aspecto:

```
for (variable in objeto){
    // bloque de instrucciones
}
```


Ejemplo:

```
const object = { a: 1, b: 2, c: 3 };

for (const property in object) {
    console.log(`${property}: ${object[property]}`);
}
```

2.5 Declaración for...of

La declaración `for...of` crea un bucle que se repite sobre objetos iterables (incluidos Array, Map (en-US), Set, objetos arguments y así sucesivamente), invocando un bucle de iteración personalizado con declaraciones que se ejecutarán para el valor de cada distinta propiedad.

	UNIVERSIDAD DON BOSCO FACULTAD DE INGENIERÍA ESCUELA DE COMPUTACIÓN		CICLO 02-2023
	MATERIA	LENGUAJES INTERPRETADOS EN EL CLIENTE	GUIA DE LABORATORIO N° 6
	PRÁCTICA	ESTRUCTURAS DE CONTROL Y MATRICES	

```
for (variable of objeto){
    // Bloque de instrucciones
}
```

El siguiente ejemplo muestra la diferencia entre un bucle `for...of` y un bucle `for...in`. Mientras que `for...in` itera sobre los nombres de propiedad, `for...of` itera sobre los valores de propiedad:

```
const arr = [3, 5, 7];
arr.foo = 'hola';

for (let i in arr) {
    console.log(i); // logs "0", "1", "2", "foo"
}

for (let i of arr) {
    console.log(i); // logs 3, 5, 7
}
```

2.6 Declaración with

La instrucción `with` permite utilizar una notación abreviada al hacer referencia a los objetos. Es muy conveniente a la hora de escribir dentro del código del script un conjunto de instrucciones repetitivas relacionada con el mismo objeto. Su sintaxis es la siguiente:

```
with(objeto){
    // instruccion o bloque de instrucciones;
}
```

Por ejemplo, si se tiene el siguiente conjunto de instrucciones:

```
document.write("Hola desde JavaScript.");
document.write("<br>");
document.write("Estás aprendiendo el uso de una nueva instrucción de JavaScript.");
```


Podría escribirse abreviadamente utilizando el `with`, lo siguiente:

```
with(document){
    write("Hola desde JavaScript.");
    write("<br>");
    write("Estás aprendiendo el uso de una nueva instrucción de JavaScript.");
}
```

3. Las sentencias break y continue.

Para agregar aún más utilidad a los bucles, JavaScript incluye las sentencias `break` y `continue`. Estas sentencias se pueden utilizar para modificar el comportamiento de los ciclos más allá del resultado de su expresión condicional. Es decir, nos permite incluir otras expresiones condicionales dentro del bloque de código que se encuentra ejecutando el bucle.

El comando `break` causa una interrupción y salida del bucle en el punto donde se lo ejecute.

	UNIVERSIDAD DON BOSCO FACULTAD DE INGENIERÍA ESCUELA DE COMPUTACIÓN		CICLO 02-2023
	MATERIA	LENGUAJES INTERPRETADOS EN EL CLIENTE	GUIA DE LABORATORIO N° 6
	PRÁCTICA	ESTRUCTURAS DE CONTROL Y MATRICES	

```
let nTotal = 35;
let con = 3;
do {
  if (con == 20) break;
  alert("El contador con es igual a: " + con);
  con++;
} while (con <= nTotal);
```

En este ejemplo, el bucle se repetirá hasta que la variable con llegue al valor 20, entonces se ejecuta la sentencia **break** que hace terminar el bucle en ese punto. El comando **continue** es un tanto diferente. Se utiliza para saltar a la siguiente repetición del bloque de código, sin completar el pase actual por el bloque de comandos.

```
let nTotal = 70;
let con = 1;
do {
  if (con == 10 || con == 20 || con == 30 || con == 40) continue;
  alert("El contador con es igual a: " + con);
  con++;
} while (con <= nTotal);
```

En el ejemplo, cuando la variable con alcance los valores 10, 20, 30 y 40 **continue** salta el resto del bloque de código y comienza un nuevo ciclo sin ejecutar el método **alert()**.

4. Matrices o arreglos

Una matriz es una colección ordenada de valores, donde cada valor individual se denomina elemento y cada elemento es ubicado en una posición numérica dentro de la matriz. Esta posición es conocida como índice.

Tome en cuenta que como JavaScript es un lenguaje débilmente tipificado, los elementos de la matriz pueden ser de cualquier tipo y los distintos elementos de una misma matriz pueden ser también de tipos diferentes. Incluso, estos elementos pueden ser también matrices, lo que permitiría definir una estructura de datos que sea una matriz de matriz.


Un arreglo en JavaScript es tratado como un objeto especial llamado Array. Los elementos de un arreglo en JavaScript se comienzan a contar desde el elemento cero [0]; es decir, el elemento con índice cero [0] es el primer elemento del arreglo, el elemento con índice uno [1] es el segundo, y así sucesivamente.

Para acceder a los elementos de un arreglo individualmente, debe utilizarse el nombre del arreglo y el índice que indica la posición del elemento deseado entre corchetes ("[" , "]"). Primero se coloca el nombre y, a continuación, encerrado entre corchetes el índice. Por ejemplo, para acceder al tercer elemento del arreglo llamado **miArreglo**, debe digitar lo siguiente: **miArreglo[2]**.

4.1 Declaración de matrices en JavaScript

La declaración de arreglos en JavaScript puede hacerse de dos formas. La primera utilizando **corchetes**, como se muestra a continuación:

```
let impares = [];
```

	UNIVERSIDAD DON BOSCO FACULTAD DE INGENIERÍA ESCUELA DE COMPUTACIÓN		CICLO 02-2023
	MATERIA	LENGUAJES INTERPRETADOS EN EL CLIENTE	GUIA DE LABORATORIO N° 6
	PRÁCTICA	ESTRUCTURAS DE CONTROL Y MATRICES	

La segunda utilizando el constructor `Array()` de la siguiente forma:

```
let pares = new Array();
```

Cuando se usa el constructor `Array()` es posible definir el tamaño del arreglo colocando entre paréntesis el número de elementos que tendrá el arreglo con un valor entero. Como se muestra a continuación:

```
let dias = new Array(5);
```

La instrucción anterior crea un arreglo llamado `dias` y define que el número de elementos de este arreglo será cinco.

Introducción de elementos en una matriz

Se pueden introducir elementos a un arreglo de varias formas. Entre las que se pueden mencionar:

- Asignando un dato a un elemento del arreglo de forma directa,
- Asignando una lista de valores al arreglo,
- Pasando argumentos al constructor `Array()`

A continuación, se muestran tres ejemplos de cada una de las formas de introducir elementos en un arreglo de JavaScript:

```
//Asignando un dato a un elemento del arreglo
let dias = [];
dia[0] = "Domingo"; //Al primer elemento del arreglo se le ha asignado el valor Domingo
dia[1] = "Lunes"; //Al segundo elemento del arreglo se le ha asignado el valor Lunes
//Asignando una lista de valores al arreglo
dias = ["Domingo", "Lunes", "Martes", "Miércoles"];
//Pasando argumentos al constructor Array()
dias = new Array("Domingo", "Lunes", "Martes", "Miércoles");
```

4.2 Acceso a los elementos de una matriz

Para poder acceder a los elementos de un arreglo es necesario escribir el nombre del arreglo y, a continuación, entre corchetes el índice del elemento. Por ejemplo, si queremos acceder al segundo elemento de un arreglo llamado `dias`. Debemos escribir una instrucción como la siguiente:


```
var dias = new Array("Domingo", "Lunes", "Martes", "Miércoles");
let x = dias[1];
```

La instrucción anterior asigna el valor "Lunes" (suponiendo que ese es el valor almacenado en `dias[1]`) a la variable `x`.

Cuando se intente acceder a un elemento de un arreglo que no ha sido asignado todavía, obtendrá un valor `undefined`.

4.3 Agregar elementos a una matriz

En JavaScript no es necesario asignar más memoria de forma explícita para agregar elementos a un arreglo, aunque inicialmente se haya declarado de un número específico de elementos. Por ejemplo, si se declaró un

	UNIVERSIDAD DON BOSCO FACULTAD DE INGENIERÍA ESCUELA DE COMPUTACIÓN		CICLO 02-2023
	MATERIA	LENGUAJES INTERPRETADOS EN EL CLIENTE	GUIA DE LABORATORIO N° 6
	PRÁCTICA	ESTRUCTURAS DE CONTROL Y MATRICES	

arreglo con 4 elementos, podemos agregarle dos más sin necesidad de reservar más memoria para dichos elementos. Esto significa que podemos escribir el siguiente script y debería funcionar correctamente:

```
let lenguajes = [4];
lenguajes[0] = "JavaScript";
lenguajes[1] = "Visual Basic";
lenguajes[2] = "PHP";
lenguajes[3] = "C/C++";
document.write("Aprenderás a usar los siguientes lenguajes:<br> ");
document.write("<OL type='1'>");
document.write("<LI>" + lenguajes[0]);
document.write("<LI>" + lenguajes[1]);
document.write("<LI>" + lenguajes[2]);
document.write("<LI>" + lenguajes[3]);
document.write("</OL>");
lenguajes[4] = "Java";
lenguajes[5] = "ASP";
document.write("Además usarás:<br> ");
document.write("<OL type='1' start='5'>");
document.write("<LI>" + lenguajes[4]);
document.write("<LI>" + lenguajes[5]);
document.write("</OL>");
```

Algo que debe tener en cuenta es que en JavaScript no es necesario agregar elementos de forma consecutiva, esto significa que si se tienen cuatro elementos, como en el ejemplo anterior, puede agregar dos elementos más en cualquier posición. Es decir, si vemos el ejemplo anterior, podríamos haber agregado en lenguajes[7] y lenguajes[8] los elementos "Java" y "ASP", en lugar de hacerlo en lenguajes[4] y lenguajes[5] como se hizo en el ejemplo.

4.4 Eliminar elementos de una matriz

Para eliminar elementos de un arreglo se usa el operador delete. Al utilizar dicho operador JavaScript establece el elemento al valor undefined, como que si no tuviera asignado valor alguno. Hay que notar que la propiedad length no se modifica al eliminar el elemento del arreglo; es decir, que el arreglo seguirá teniendo el mismo número de elementos que tenía antes de eliminar el elemento.


Ejemplo:

```
let colores = ["rojo", "verde", "azul"];
delete colores[1];
alert("colores[" + 1 + "] = " + colores[1]);
```

Para poder eliminar realmente un elemento para que todos los elementos con índice de posición superior a este se desplacen a posiciones con índice menor, se tienen que utilizar el método de matriz [Array.shift\(\)](#) para eliminar el primer elemento de la matriz y [Array.pop\(\)](#) para eliminar el último elemento, y [Array.splice\(\)](#) para eliminar un rango contiguo de elementos desde una matriz.

Ejemplo:

```
let lugares = ["primero", "segundo", "tercer", "cuarto"];
let primerlugar = lugares.shift(); //El primer elemento es eliminado del arreglo
//y asignado a la variable primerlugar
alert(lugares.toSource());
```


	UNIVERSIDAD DON BOSCO FACULTAD DE INGENIERÍA ESCUELA DE COMPUTACIÓN		CICLO 02-2023
	MATERIA	LENGUAJES INTERPRETADOS EN EL CLIENTE	GUIA DE LABORATORIO N° 6
	PRÁCTICA	ESTRUCTURAS DE CONTROL Y MATRICES	

4.5 Obtener el número de elementos de una matriz

Para obtener el número de elementos de un arreglo, o su tamaño se utiliza la propiedad `length`. Esta propiedad lo que obtiene realmente es el índice de la siguiente posición disponible o no ocupada al final del arreglo. Incluso si existen elementos con índices menores no ocupados.

Por ejemplo:

```
let trimestre = new Array("Enero", "Febrero", "Marzo");
alert("El tamaño del arreglo es: " + trimestre.length);
```

El valor mostrado será de tres. Sin embargo, si más adelante decide agregar otros elementos sin tener cuidado del índice último del arreglo ocupado. Puede darse lo siguiente:

```
let trimestre = new Array("Enero", "Febrero", "Marzo");

trimestre[7] = "Julio";
trimestre[8] = "Agosto";
trimestre[9] = "Septiembre";
alert("El tamaño del arreglo es: " + trimestre.length);
```

El nuevo tamaño mostrado será de 10. Por esta razón parece razonable utilizar posiciones adyacentes en el índice de los arreglos para no obtener resultados inesperados en un script.

4.6 Métodos comunes para trabajar con arreglos

JavaScript proporciona algunos métodos para trabajar con arreglos. Algunos de ellos de uso frecuente pueden utilizarse a conveniencia dentro de los scripts que desarrollemos.

4.6.1 concat()

Este método devuelve el arreglo que resulta de añadir los argumentos al arreglo sobre el que se ha invocado. Por ejemplo, las siguientes instrucciones:


```
let miArreglo = ["rojo", "verde", "azul"];
alert(miArreglo.concat("amarillo", "morado"));
```

Mostrarían el siguiente mensaje en un cuadro de alerta: **rojo, verde, azul, amarillo, morado**

Debe tomar en cuenta que `concat()` no modifica el arreglo original. Es decir, si se manda a imprimir `miArreglo` solamente se imprimirían los tres colores que le fueron asignados.

4.6.2 join()

El método `join()` convierte el arreglo en una cadena y permite al programador especificar cómo se separan los elementos en la cadena resultante. Normalmente, cuando se imprime un arreglo, la salida es una lista de elementos separados por coma. Si se utiliza `join()` se puede establecer el carácter de separación entre elementos. Por ejemplo, en el siguiente código:

	UNIVERSIDAD DON BOSCO FACULTAD DE INGENIERÍA ESCUELA DE COMPUTACIÓN		CICLO 02-2023
	MATERIA	LENGUAJES INTERPRETADOS EN EL CLIENTE	GUIA DE LABORATORIO N° 6
	PRÁCTICA	ESTRUCTURAS DE CONTROL Y MATRICES	

```
let miArreglo = ["rojo", "verde", "azul"];
let stringVersion = miArreglo.join(" | ");
alert(stringVersion);
```

Se imprimiría: rojo | verde | azul

El arreglo original no se destruye como efecto secundario al devolver la cadena de sus elementos enlazada. Si se desea hacer esto deberá asignar al mismo objeto la cadena devuelta. Así:

```
let miArreglo = ["rojo", "verde", "azul"];
miArreglo = miArreglo.join(" | ");
```

4.6.3 reverse()

Este método invierte el orden de los elementos de un arreglo en particular. Este método si altera el arreglo original, de modo que si se manda a imprimir, nos mostraría los elementos invertidos. Por ejemplo, si se tiene:

```
let miArreglo = ["rojo", "verde", "azul"];
miArreglo.reverse();
alert(miArreglo);
```

La salida será: azul, verde, rojo

4.6.4 slice()

Este método devuelve un subarreglo; del arreglo sobre el que se invoca. Como no actúa sobre el arreglo, el arreglo original queda intacto. El método tiene dos argumentos que son el índice inicial y el índice final. Devuelve un arreglo que contiene los elementos desde el índice inicial hasta el índice final, excluyéndolo. Si sólo se proporciona un argumento, el método devuelve los elementos desde el índice dado hasta el final del arreglo. Observe el siguiente ejemplo:

```
let miArreglo = [1, 2, 3, 4, 5];
miArreglo.slice(2); //Devuelve [3, 4, 5]
miArreglo.slice(1, 3); //Devuelve [2, 3]
miArreglo.slice(-3); //Devuelve [3, 4, 5]
miArreglo.slice(-4, 3); //Devuelve [2, 3]
```


Observe el resultado cuando se utilizan índices negativos y trate de determinar por qué el resultado es el que se muestra en los comentarios.

4.6.5 splice()

Este método se utiliza para añadir, reemplazar o eliminar elementos de un arreglo particular. Devuelve los elementos que se eliminan. Posee tres argumentos, que se muestran en la siguiente sintaxis:

```
splice(inicio, cantidad_borrar, valores);
```

Donde, inicio es el índice donde se va a realizar la operación, cantidad_borrar es la cantidad de elementos a eliminar comenzando por el índice de inicio. Si se omite cantidad_borrar, se eliminan todos los elementos desde

	UNIVERSIDAD DON BOSCO FACULTAD DE INGENIERÍA ESCUELA DE COMPUTACIÓN		CICLO 02-2023
	MATERIA	LENGUAJES INTERPRETADOS EN EL CLIENTE	GUIA DE LABORATORIO N° 6
	PRÁCTICA	ESTRUCTURAS DE CONTROL Y MATRICES	

el inicio hasta el final del arreglo y a la vez se devuelven. Cualquier argumento adicional presentado en valores (que se separan por comas, si es más de uno) se introduce en el lugar de los elementos eliminados.

4.6.6 sort()

Es uno de los métodos más útiles utilizados con arreglos en JavaScript. El método clasifica los elementos del arreglo lexicográficamente. Lo hace convirtiendo primero los elementos del arreglo en cadenas y luego los ordena. Esto significa que si ordena números podría obtener resultados inesperados. Vea el siguiente ejemplo:

```
let miArreglo = [14, 52, 3, 14, 45, 36];
miArreglo.sort();
alert(miArreglo);
```

Como el orden es lexicográfico, al ejecutar el script el resultado sería: **14, 14, 3, 36, 45, 52**

La función sort es muy flexible y permite pasar como argumento una función de comparación que permita ordenar numéricamente y no alfabéticamente los elementos. Las funciones se analizarán en la guía sobre funciones.

4.6.7 toString()


El método toString() devuelve una cadena que contiene los valores del arreglo separados por comas. Este método se invoca automáticamente cuando se imprime un arreglo. Equivale a invocar join() sin argumentos.

Ejemplo:

```
let numeros = [1, 2, 3, 4, 5];
let letras = ['a', 'b', 'c', 'd'];
alert(numeros.toString()); //Devuelve '1, 2, 3'
alert(letras.toString()); //Devuelve 'a, b, c'
```

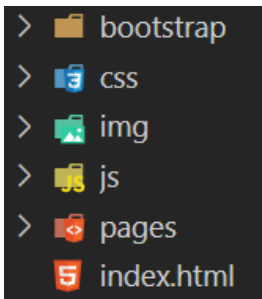
III. MATERIALES Y EQUIPO

No.	Requerimientos	Cantidad
1	Computadora con alguno de los siguientes editores de código fuente tales como: Sublime Text, Visual Studio Code, Notepad++ u otro y Navegadores Web Actualizados (Firefox, Chrome, Safari, Opera, Microsoft Edge entre otros)	1
2	Guía de laboratorio 6	1
3	Memoria USB o cualquier otro medio de almacenamiento.	1

	UNIVERSIDAD DON BOSCO FACULTAD DE INGENIERÍA ESCUELA DE COMPUTACIÓN		CICLO 02-2023
	MATERIA	LENGUAJES INTERPRETADOS EN EL CLIENTE	GUIA DE LABORATORIO N° 6
	PRÁCTICA	ESTRUCTURAS DE CONTROL Y MATRICES	

III. DESARROLLO DE LA PRÁCTICA

Cree una carpeta con el nombre Guia6_[su número de carnet], y luego proceda a descomprimir los recursos proporcionados. Deberá de tener la siguiente estructura en su proyecto.



Ejemplo 1: Creando el índice de nuestro sitio web

1. Cree un archivo index.html en la carpeta raíz de su proyecto y proceda a crear la estructura básica del documento html.
2. Dentro de las etiquetas <head></head> coloque el siguiente código:

```

7      <link href="bootstrap/css/bootstrap.min.css" rel="stylesheet" />
8      <link href="css/styles.css" rel="stylesheet" />
9      <link rel="icon" href="img/js.png" sizes="32x32" type="image/png" />
10     <title>Utilizando estructuras de control</title>


```

3. Dentro de la etiqueta <body></body> coloque el siguiente código:

```

13     <main class="d-flex flex-nowrap">
14         <div class="d-flex flex-column flex-shrink-0 p-3 text-bg-dark nav-bar">
15             <a href="index.html"
16                 class="d-flex align-items-center mb-3 mb-md-0 me-md-auto text-white text-decoration-none">
17                 
18
19                 <span class="fs-4">JavaScript</span>
20             </a>
21             <hr />
22             <ul class="nav nav-pills flex-column mb-auto">
23                 <li class="nav-item">
24                     <a href="pages/registroAcademico.html" class="nav-link text-white">
25                         <i class="bi bi-file-spreadsheet" role="img" aria-label="RegistroAcademico"></i>
26                         Registro académico
27                     </a>
28                 </li>
29                 <li>
30                     <a href="pages/tablasMultiplicar.html" class="nav-link text-white">
31                         <i class="bi bi-person-workspace" role="img" aria-label="TablasMultiplicar"></i>
32                         Tablas de multiplicar
33                     </a>
34                 </li>
35                 <li>
36                     <a href="pages/promedioNotas.html" class="nav-link text-white">
37                         <i class="bi bi-file-earmark-medical" role="img" aria-label="Promedio"></i>
38                         Promedio notas
39                     </a>
40                 </li>

```

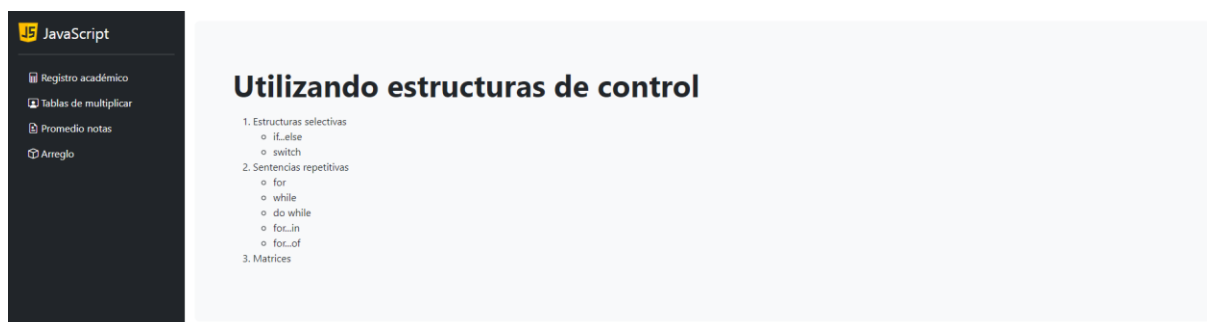
	UNIVERSIDAD DON BOSCO FACULTAD DE INGENIERÍA ESCUELA DE COMPUTACIÓN		CICLO 02-2023
	MATERIA	LENGUAJES INTERPRETADOS EN EL CLIENTE	GUIA DE LABORATORIO N° 6
	PRÁCTICA	ESTRUCTURAS DE CONTROL Y MATRICES	


```

41         <li>
42             <a href="pages/arreglo.html" class="nav-link text-white">
43                 <i class="bi bi-box" role="img" aria-label="Arreglo"></i>
44                 Arreglo
45             </a>
46         </li>
47     </ul>
48 </div>
49 <div class="d-flex flex-column flex-shrink-0 p-3 column-container">
50     <div class="p-5 mb-2 bg-light rounded-3">
51         <div class="container-fluid py-4">
52             <h1 class="display-5 fw-bold">Utilizando estructuras de control</h1>
53             <p class="col-md-8 fs-4">
54                 <ol>
55                     <li>
56                         Estructuras selectivas
57                         <ul>
58                             <li>if...else</li>
59                             <li>switch</li>
60                         </ul>
61                     </li>
62                     <li>
63                         Sentencias repetitivas
64                         <ul>
65                             <li>for</li>
66                             <li>while</li>
67                             <li>do while</li>
68                             <li>for...in</li>
69                             <li>for...of</li>
70                         </ul>
71                     </li>
72                     <li>Matrices </li>
73                 </ol>
74             </p>
75         </div>
76     </div>
77 </div>
78 </main>
79
80 <script src="bootstrap/js/bootstrap.bundle.min.js"></script>

```

4. Visualice el resultado de su página index.html



	UNIVERSIDAD DON BOSCO FACULTAD DE INGENIERÍA ESCUELA DE COMPUTACIÓN		CICLO 02-2023
	MATERIA	LENGUAJES INTERPRETADOS EN EL CLIENTE	GUIA DE LABORATORIO N° 6
	PRÁCTICA	ESTRUCTURAS DE CONTROL Y MATRICES	

Ejemplo 2: Utilizando arreglos multidimensionales

1. Utilice el archivo registroAcademico.html que ha sido proporcionado, localice la siguiente sección

```

51      <div class="d-flex flex-column flex-shrink-0 p-3 column-container contenedor">
52      |      <!-- CONTINUAR CON EL CODIGO AQUI -->
53      |
54      </div>

```

2. Desarrolle el siguiente código dentro del contenedor anterior.

```

52      <div class="container-fluid py-5">
53      |      <div class="row">
54      |      |      <div class="col-md-6">
55      |      |      |      <div class="h-100 p-4 bg-light border rounded-3">
56      |      |      |      |      <h3>Datos de estudiantes</h3>
57      |      |      |      |      <form onsubmit="return false">
58      |      |      |      |      |      <div class="row mb-3">
59      |      |      |      |      |      |      <label for="inputCarnet" class="col-sm-2 col-form-label">Carnet</label>
60      |      |      |      |      |      |      <div class="col-sm-10">
61      |      |      |      |      |      |      |      <input type="text" class="form-control" id="inputCarnet" maxlength="5" />
62      |      |      |      |      |      |      |      </div>
63      |      |      |      |      |      |      </div>
64      |      |      |      |      |      |      <div class="row mb-3">
65      |      |      |      |      |      |      |      <label for="inputNombre" class="col-sm-2 col-form-label">Nombres</label>
66      |      |      |      |      |      |      |      <div class="col-sm-10">
67      |      |      |      |      |      |      |      |      <input type="text" class="form-control" id="inputNombre" />
68      |      |      |      |      |      |      |      </div>
69      |      |      |      |      |      |      </div>
70      |      |      |      |      |      |      <div class="row mb-3">
71      |      |      |      |      |      |      |      <label for="inputApellidos" class="col-sm-2 col-form-label">Apellidos</label>
72      |      |      |      |      |      |      |      <div class="col-sm-10">
73      |      |      |      |      |      |      |      |      <input type="text" class="form-control" id="inputApellidos" />
74      |      |      |      |      |      |      |      </div>
75      |      |      |      |      |      |      </div>
76      |      |      |      |      |      |      <div class="row mb-3">
77      |      |      |      |      |      |      |      <button type="button" class="btn btn-success" id="idBtnAgregarEstudiante">
78      |      |      |      |      |      |      |      |      <i class="bi bi-person-plus-fill" height="16" width="16"></i>
79      |      |      |      |      |      |      |      |      Registrar
80      |      |      |      |      |      |      |      </button>
81      |      |      |      |      |      |      |      <button type="button" class="btn btn-primary" id="idBtnMostrarEstudiantes">
82      |      |      |      |      |      |      |      |      <i class="bi bi-person-lines-fill"></i> Mostrar estudiantes
83      |      |      |      |      |      |      |      </button>
84      |      |      |      |      |      |      </div>
85      |      |      |      |      |      |      </div>
86      |      |      |      |      |      |      </div>
87      |      |      |      |      |      |      <div class="col-md-12 py-5">
88      |      |      |      |      |      |      |      <h3>Estudiantes registrados</h3>
89      |      |      |      |      |      |      |      <div id="idContainerEstudiantes">Ninguno</div>
90      |      |      |      |      |      |      </div>
91      |      |      |      |      |      |      </div>
92      |      </div>

```

3. Observe que en la parte final de nuestro código HTML se encuentran dos etiqueta `<script></script>`, el primer archivo corresponde al framework de Bootstrap y el segundo archivo es con el que vamos a trabajar.

```

97      <script src="..../bootstrap/js/bootstrap.bundle.min.js"></script>
98      <script src="..../js/registroAcademico.js" defer></script>

```

4. Ubíquese en el archivo registroAcademico.js y comience a desarrollar el siguiente código fuente.




UNIVERSIDAD DON BOSCO
FACULTAD DE INGENIERÍA
ESCUELA DE COMPUTACIÓN

CICLO
02-2023

MATERIA	LENGUAJES INTERPRETADOS EN EL CLIENTE
PRÁCTICA	ESTRUCTURAS DE CONTROL Y MATRICES

GUIA DE
LABORATORIO N° 6

```
1 document.addEventListener("DOMContentLoaded", function () {
2     //Accedemos al contenedor donde se mostrara los estudiantes
3     const containerEstudiantes = document.querySelector(
4         "#idContainerEstudiantes"
5     );
6
7     //Accedemos a cada boton por medio de la API DOM
8     const btnAddEstudiante = document.querySelector("#idBtnAgregarEstudiante");
9     const btnViewEstudiantes = document.querySelector("#idBtnMostrarEstudiantes");
10
11     //Agregamos el evento click a los botones, adicionalmente
12     //se le asigna la funcion que realizará la operación
13     btnAddEstudiante.addEventListener("click", addEstudiantes);
14     btnViewEstudiantes.addEventListener("click", viewEstudiantes);
15
16     // Arreglo de forma global
17     let arrayEstudiantes = new Array();
18
19     //creando funciones
20     function addEstudiantes() {
21         const inputCarnet = document
22             .querySelector("#inputCarnet")
23             .value.toString()
24             .toUpperCase();
25         const inputNombre = document
26             .querySelector("#inputNombre")
27             .value.toString()
28             .toUpperCase();
29         const inputApellidos = document
30             .querySelector("#inputApellidos")
31             .value.toString()
32             .toUpperCase();
33
34         if (inputCarnet != "" && inputNombre != "" && inputApellidos != "") {
35             arrayEstudiantes.push(
36                 new Array(inputCarnet, inputNombre, inputApellidos)
37             );
38             alert("Se registro el nuevo estudiante");
39             //Limpiando campos del formulario
40             document.querySelector("#inputCarnet").value = "";
41             document.querySelector("#inputNombre").value = "";
42             document.querySelector("#inputApellidos").value = "";
43             document.querySelector("#inputCarnet").focus();
44         } else {
45             alert("Faltan campos que completar");
46         }
47     }
48 }
```


	UNIVERSIDAD DON BOSCO FACULTAD DE INGENIERÍA ESCUELA DE COMPUTACIÓN		CICLO 02-2023
	MATERIA	LENGUAJES INTERPRETADOS EN EL CLIENTE	GUIA DE LABORATORIO N° 6
	PRÁCTICA	ESTRUCTURAS DE CONTROL Y MATRICES	

```

49 function viewEstudiantes() {
50     //Validando que existan estudiantes registrados
51     let totalEstudiantes = arrayEstudiantes.length;
52     if (totalEstudiantes > 0) {
53         let carnet;
54         let nombres;
55         let apellidos;
56         let table = "<table class='table table-light table-striped'>";
57         table += "<thead>";
58         table += "<tr>";
59         table += "<th scope='col' style='width: 5%;>#</th>";
60         table += "<th scope='col' style='width: 15%;>Carnet</th>";
61         table += "<th scope='col'>Nombres</th>";
62         table += "<th scope='col'>Apellidos</th>";
63         table += "</tr>";
64         table += "</thead>";
65         table += "<tbody>";
66
67         // Utilizaremos un bucle for para recorrer el arreglo de estudiantes
68         for (let i = 0; i < arrayEstudiantes.length; i++) {
69             //Accediendo a las posiciones del arreglo
70             carnet = arrayEstudiantes[i][0];
71             nombres = arrayEstudiantes[i][1];
72             apellidos = arrayEstudiantes[i][2];
73
74             table += `<tr>`;
75             table += `<td scope='row' style='font-weight: bold;'>${i + 1}</td>`;
76             table += `<td>${carnet}</td>`;
77             table += `<td>${nombres}</td>`;
78             table += `<td>${apellidos}</td>`;
79             table += `</tr>`;
80         }
81
82         table += "</tbody>";
83         table += "</table>";
84         containerEstudiantes.innerHTML = table;
85     } else {
86         alert("No se han registrado estudiantes");
87     }
88 }
89 };

```

5. Verifique el funcionamiento de su página registroAcademico.html, tendría que obtener un resultado como el siguiente.

 JavaScript

Registro académico

Tablas de multiplicar

Promedio notas

Arreglo

Datos de estudiantes

Carnet

Nombres


Apellidos

Registrar

Mostrar estudiantes

Estudiantes registrados

#	Carnet	Nombres	Apellidos
1	AB410	CARLOS	HERNÁNDEZ
2	CC101	CARMEN	CARRILLO

	UNIVERSIDAD DON BOSCO FACULTAD DE INGENIERÍA ESCUELA DE COMPUTACIÓN		CICLO 02-2023
	MATERIA	LENGUAJES INTERPRETADOS EN EL CLIENTE	
	PRÁCTICA	ESTRUCTURAS DE CONTROL Y MATRICES	
			GUIA DE LABORATORIO N° 6

Ejemplo 3: Utilizando estructuras de control

1. Utilice el archivo tablasMultiplicar.html que ha sido proporcionado, localice la siguiente sección

```

51     <div class="d-flex flex-column flex-shrink-0 p-3 column-container contenedor">
52         <!-- CONTINUAR CON EL CODIGO AQUI -->
53     </div>
54

```

2. Desarrolle el siguiente código dentro del contenedor anterior.

```

50     <div class="container-fluid py-5">
51         <div class="row">
52             <div class="col-md-6">
53                 <div class="h-100 p-4 bg-light border rounded-3">
54                     <h3>Generando tablas de multiplicar</h3>
55                     <form onsubmit="return false">
56                         <div class="row mb-3">
57                             <label for="inputTabla" class="col-sm-3 col-form-label">Ingrese un número</label>
58                             <div class="col-sm-9">
59                                 <input type="number" class="form-control" id="inputTabla" maxlength="10" value="1" />
60                             </div>
61                         </div>
62                         <button type="button" class="btn btn-success" id="idBtnCalcular">
63                             <i class="bi bi-calculator" height="16" width="16"></i>
64                             Calcular
65                         </button>
66                     </form>
67                 </div>
68             </div>
69             <div class="col-md-12 py-5">
70                 <div id="idContainerResultado"></div>
71             </div>
72         </div>
73     </div>
74

```

3. Observe que en la parte final de nuestro código HTML se encuentran dos etiqueta <script></script>, el primer archivo corresponde al framework de Bootstrap y el segundo archivo es con el que vamos a trabajar.

```

78     <script src="../../bootstrap/js/bootstrap.bundle.min.js"></script>
79     <script src="../../js/tablasMultiplicar.js" defer></script>


```

4. Ubíquese en el archivo tablasMultiplicar.js y comience a desarrollar el siguiente código fuente.

```

1  //Accedemos al contenedor donde se mostrara los estudiantes
2  const containerResultado = document.querySelector("#idContainerResultado");
3
4  //Accedemos a cada boton por medio de la API DOM
5  const btnCalcular = document.querySelector("#idBtnCalcular");
6
7  //Agregamos el evento click al boton calcular
8  //se le asigna la funcion que realizará la operación
9  btnCalcular.addEventListener("click", calcularTabla);

```

	UNIVERSIDAD DON BOSCO FACULTAD DE INGENIERÍA ESCUELA DE COMPUTACIÓN		CICLO 02-2023
	MATERIA	LENGUAJES INTERPRETADOS EN EL CLIENTE	GUIA DE LABORATORIO N° 6
	PRÁCTICA	ESTRUCTURAS DE CONTROL Y MATRICES	

```

11 function calcularTabla() {
12     //capturando el valor del campo
13     const inputTabla = document.querySelector("#inputTabla").value;
14
15     //inicializamos nuestro contador
16     let contador = 1;
17
18     //verifiquemos que el dato colocado sea un numero entero positivo
19     if (inputTabla > 0) {
20         let tabla = `<h2>Tabla de multiplicar del ${inputTabla}</h2>`;
21         //utilizaremos do while para generar la tabla de multiplicar
22         // que el usuario ha indicado
23         do {
24             let resultado = contador * inputTabla;
25             tabla += `<div class="row text-center">`;
26             tabla += `<div class="col-md-1 column"><h3>${contador}</h3></div>`;
27             tabla += `<div class="col-md-1 column-green"><h3>x</h3></div>`;
28             tabla += `<div class="col-md-1 column"><h3>${inputTabla}</h3></div>`;
29             tabla += `<div class="col-md-1 column-green"><h3>=</h3></div>`;
30             tabla += `<div class="col-md-1 column"><h3>${resultado}</h3></div>`;
31             tabla += `</div>`;
32
33             //incrementamos el valor del contador
34             //para que podamos salir del do while
35             contador++;
36         } while (contador <= 12);
37
38         document.querySelector("#inputTabla").value = 1;
39         document.querySelector("#inputTabla").focus();
40         containerResultado.innerHTML = tabla;
41     } else {
42         alert("No se ha ingresado un número válido");
43     }
44 }

```

5. Verifique el funcionamiento de su página tablasMultiplicar.html, tendría que obtener un resultado como el siguiente.

 JavaScript

Registro académico

Tablas de multiplicar

Promedio notas

Arreglo


Generando tablas de multiplicar

Ingrese un número

Calcular

Tabla de multiplicar del 10

1	x	10	=	10
2	x	10	=	20
3	x	10	=	30
4	x	10	=	40

	UNIVERSIDAD DON BOSCO FACULTAD DE INGENIERÍA ESCUELA DE COMPUTACIÓN		CICLO 02-2023
	MATERIA	LENGUAJES INTERPRETADOS EN EL CLIENTE	GUIA DE LABORATORIO N° 6
	PRÁCTICA	ESTRUCTURAS DE CONTROL Y MATRICES	

Ejemplo 4: Utilizando estructuras de control anidadas

1. Utilice el archivo promedioNotas.html que ha sido proporcionado, localice la siguiente sección

```

51      <div class="d-flex flex-column flex-shrink-0 p-3 column-container contenedor">
52          <!-- CONTINUAR CON EL CODIGO AQUI -->
53      </div>
54  
```

2. Desarrolle el siguiente código dentro del contenedor anterior.

```

52      <div class="container-fluid py-5">
53          <div class="row">
54              <div class="col-md-6">
55                  <div class="h-100 p-4 bg-light border rounded-3">
56                      <h3>Registro de calificaciones</h3>
57                      <form onsubmit="return false">
58                          <div class="row mb-3">
59                              <label for="inputNumeroEstudiantes" class="col-sm-4 col-form-label">Ingrese el numero de estudiantes</label>
60                              <div class="col-sm-8">
61                                  <input type="number" class="form-control" id="inputNumeroEstudiantes" value="1"/>
62                              </div>
63                          </div>
64                          <button type="button" class="btn btn-success" id="idBtnPromedio">
65                              <i class="bi bi-calculator-fill" height="16" width="16"></i>
66                              Generar
67                          </button>
68                      </form>
69                  </div>
70              </div>
71              <div class="col-md-12 py-5">
72                  <div id="idContainerEstudiantes"></div>
73              </div>
74          </div>
75      </div>
76  
```

3. Observe que en la parte final de nuestro código HTML se encuentran dos etiqueta `<script></script>`, el primer archivo corresponde al framework de Bootstrap y el segundo archivo es con el que vamos a trabajar.

```

80      <script src="../../bootstrap/js/bootstrap.bundle.min.js"></script>
81      <script src="../../js/promedioNotas.js" defer></script>

```

4. Ubíquese en el archivo promedioNotas.js y comience a desarrollar el siguiente código fuente.

```

1  //Accedemos al contenedor donde se mostrara los estudiantes
2  const containerEstudiantes = document.querySelector("#idContainerEstudiantes");
3
4  //Accedemos a cada boton por medio de la API DOM
5  const btnPromedio = document.querySelector("#idBtnPromedio");
6
7  //Agregamos el evento click a los botones, adicionalmente
8  //se le asigna la funcion que realizará la operación
9  btnPromedio.addEventListener("click", generarEstudiantes);

```



UNIVERSIDAD DON BOSCO
FACULTAD DE INGENIERÍA
ESCUELA DE COMPUTACIÓN

CICLO
02-2023

MATERIA


LENGUAJES INTERPRETADOS EN EL CLIENTE

PRÁCTICA


ESTRUCTURAS DE CONTROL Y MATRICES

GUIA DE
LABORATORIO N° 6

```
11 function generarEstudiantes() {
12     //utilizaremos un arreglo para guardar la informacion del estudiante
13     let arrayEstudiante = new Array();
14
15     let totalEstudiantes = document.querySelector(
16         "#inputNumeroEstudiantes"
17     ).value;
18     let contador = 1;
19
20     // Utilizaremos un while para recorrer el total de estudiantes
21     let estudiante,
22         calificacion,
23         convertir = 0;
24     while (contador <= totalEstudiantes) {
25         estudiante = prompt(`Ingrese el nombre del estudiante ${contador}`);
26
27         //verificando que sea un valor entero positivo
28         //y que se encuentre en el rango de 0 - 10
29         do {
30             calificacion = prompt(
31                 `Ingrese la calificacion del estudiante ${contador}`
32             );
33
34             convertir = parseFloat(calificacion);
35         } while (isNaN(convertir) || convertir < 0 || convertir > 10);
36
37         // Asignando los valores al arreglo
38         arrayEstudiante[contador - 1] = new Array(
39             estudiante,
40             parseFloat(calificacion).toFixed(2)
41         );
42         contador++;
43     }
44
45     //Recorriendo el arreglo con for..of
46     //Verificaremos cual es el promedio de las calificaciones
47     // y cual de los estudiantes posee la calificaicon mas alta
48     let calificacionAlta = 0,
49         promedio = 0,
50         posicion = 0;
51
52     let listado = "<h3>Listado de estudiantes registrados</h3>";
53     listado += "<ol>";
54     for (let indice of arrayEstudiante) {
55         let nombre = indice[0];
56         let nota = indice[1];
57
58         //imprimiendo lista de estudiantes
59         listado += `<li><b>Nombre:</b> ${nombre} - <b>Calificación:</b> ${nota}</li>`;
60
61         //verificacion de calificacion mas alta
62         if (nota > calificacionAlta) {
63             posicion = indice;
64         }
65
66         //calculando el promedio
67         promedio += parseFloat(nota);
68     }
69     listado += "</ol>";
70     promedio = parseFloat(promedio / arrayEstudiante.length).toFixed(2);
71     listado += `<p><b>Promedio de calificaciones:</b> ${promedio}`;
72     listado += `<br><b>Estudiante con mejor calificación:</b> ${posicion[0]}</p>`;
73
74     //Imprimiendo resultado
75     containerEstudiantes.innerHTML = listado;
76 }
```

	UNIVERSIDAD DON BOSCO FACULTAD DE INGENIERÍA ESCUELA DE COMPUTACIÓN		CICLO 02-2023
	MATERIA	LENGUAJES INTERPRETADOS EN EL CLIENTE	GUIA DE LABORATORIO N° 6
	PRÁCTICA	ESTRUCTURAS DE CONTROL Y MATRICES	

- Verifique el funcionamiento de su página promedioNotas.html, tendría que obtener un resultado como el siguiente.

 JavaScript

- Registro académico
- Tablas de multiplicar
- Promedio notas
- Arreglo

Registro de calificaciones

Ingrese el numero de estudiantes

Generar

Listado de estudiantes registrados

1. **Nombre:** Marcos - **Calificación:** 5.50
 2. **Nombre:** Ana - **Calificación:** 7.80

Promedio de calificaciones: 6.65
Estudiante con mejor calificación: Ana

Ejemplo 5: Trabajando con arreglos

- Utilice el archivo arreglo.html que ha sido proporcionado, localice la siguiente sección

```

51      <div class="d-flex flex-column flex-shrink-0 p-3 column-container contenedor">
52          <!-- CONTINUAR CON EL CODIGO AQUI -->
53      </div>
54  </div>


```

- Desarrolle el siguiente código dentro del contenedor anterior.

```

52      <div class="container-fluid py-5">
53          <div class="row">
54              <div class="col-md-6">
55                  <div class="h-100 p-4 bg-light border rounded-3">
56                      <h3>Creando un arreglo numerico</h3>
57                      <form onsubmit="return false">
58                          <div class="row mb-3">
59                              <label for="inputNumero" class="col-sm-4 col-form-label">Ingrese un numero en el
60                                  arreglo</label>
61                              <div class="col-sm-8">
62                                  <input type="number" class="form-control" id="inputNumero" value="1" />
63                              </div>
64                          </div>
65                      </div>
66                      <button type="button" class="btn btn-success" id="idBtnAgregar">
67                          <i class="bi bi-plus" height="16" width="16"></i>
68                          Agregar
69                      </button>
70                      <button type="button" class="btn btn-primary" id="idBtnOrdenar">
71                          <i class="bi bi-list-ol" height="16" width="16"></i>
72                          Ordenar
73                      </button>
74                      </form>
75                  </div>
76              </div>
77              <div class="col-md-12 py-5">
78                  <div id="idContainerArreglo" class="row text-center bg-light">
79                      <h3 class="none">Valores del arreglo ingresado</h3>
80                  </div>
81                  <br />
82                  <div id="idContainerArregloOrdenado" class="row text-center bg-light">
83                      <h3>Arreglo ordenado</h3>
84                  </div>
85              </div>
86          </div>
87      </div>

```

	UNIVERSIDAD DON BOSCO FACULTAD DE INGENIERÍA ESCUELA DE COMPUTACIÓN		CICLO 02-2023
	MATERIA	LENGUAJES INTERPRETADOS EN EL CLIENTE	GUIA DE LABORATORIO N° 6
	PRÁCTICA	ESTRUCTURAS DE CONTROL Y MATRICES	

3. Observe que en la parte final de nuestro código HTML se encuentran dos etiqueta `<script></script>`, el primer archivo corresponde al framework de Bootstrap y el segundo archivo es con el que vamos a trabajar.

```

91 <script src="../../bootstrap/js/bootstrap.bundle.min.js"></script>
92 <script src="../../js/arreglo.js" defer></script>


```

4. Ubíquese en el archivo arreglo.js y comience a desarrollar el siguiente código fuente.

```

1 //Accedemos al contenedor donde se mostrara los estudiantes
2 const containerArreglo = document.querySelector("#idContainerArreglo");
3 const containerArregloOrdenado = document.querySelector(
4   "#idContainerArregloOrdenado"
5 );
6
7 //Accedemos a cada boton por medio de la API DOM
8 const btnAgregar = document.querySelector("#idBtnAgregar");
9 const btnOrdenar = document.querySelector("#idBtnOrdenar");
10
11 //Agregamos el evento click a los botones, adicionalmente
12 //se le asigna la funcion que realizará la operación
13 btnAgregar.addEventListener("click", agregarElemento);
14 btnOrdenar.addEventListener("click", ordenarElementos);
15
16 let arreglo = new Array();
17
18 function agregarElemento() {
19   const numero = parseInt(document.querySelector("#inputNumero").value);
20   //verificando que sea un numero
21   if (isNaN(numero)) {
22     alert("Debe ingresar un numero válido");
23   } else {
24     //Agregamos un nuevo elemento al arreglo
25     arreglo.push(numero);
26
27     //Utilizaremos la API DOM para crear un elemento html
28     let caja = document.createElement("div"); //Creamos un elemento <div></div>
29     caja.className = "col-md-1 colum"; //Agregamos una clase al elemento <div></div>
30     let valor = document.createElement("h3"); //Creamos un elemento <h3></h3>
31     valor.textContent = numero; //Agregamos texto al elemento <h3></h3>
32     caja.appendChild(valor); //Le pasamos como hijo la etiqueta <h3></h3> a nuestro <div></div>
33
34     //Insertamos los nuevos elementos en el contenedor
35     //se utiliza beforeend para insertar el nuevo
36     //elemento dentro del idContainerArreglo y despues de su ultimo hijo
37     containerArreglo.insertAdjacentElement("beforeend", caja);
38   }
39 }
40
41 function ordenarElementos() {
42   //utilizaremos un for...of para recorrer el arreglo
43   //a su vez se utilizara .sort() para ordenarlo
44   for (let i of arreglo.sort()) {
45     let caja = document.createElement("div");
46     caja.className = "col-md-1 colum-green";
47     let valor = document.createElement("h3");
48     valor.textContent = i;
49     caja.appendChild(valor);
50     containerArregloOrdenado.insertAdjacentElement("beforeend", caja);
51   }
52 }

```

	UNIVERSIDAD DON BOSCO FACULTAD DE INGENIERÍA ESCUELA DE COMPUTACIÓN		CICLO 02-2023
	MATERIA	LENGUAJES INTERPRETADOS EN EL CLIENTE	
	PRÁCTICA	ESTRUCTURAS DE CONTROL Y MATRICES	
			GUIA DE LABORATORIO N° 6

5. Verifique el funcionamiento de su página arreglo.html, tendría que obtener un resultado como el siguiente.

 JavaScript

- Registro académico
- Tablas de multiplicar
- Promedio notas
- Arreglo**

Creando un arreglo numerico

Ingrese un numero en el arreglo

+ Agregar Ordenar

Valores del arreglo ingresado					
-2	50	9	80	-80	199

Arreglo ordenado					
-2	-80	199	50	80	9


IV. EJERCICIOS COMPLEMENTARIOS

Utilice el sitio web desarrollado en los ejemplos para adicionar en el menú y funcionamiento de sus ejercicios complementarios, utilice su creatividad para capturar datos y mostrar sus respectivos resultados.

- Cree un script que permita el ingreso de un número entero y muestre en pantalla la siguiente información:
 - Cantidad de cifras
 - Cantidad de cifras impares
 - Cantidad de cifras pares
 - Suma de cifras impares
 - Suma de cifras pares
 - Suma de todas las cifras
 - Cifra mayor
 - Cifra menor.
- Cree un script que le permita a un vendedor ingresar el nombre y precio del producto (utilice arreglos para almacenar la información). Cada vez que ingrese un nuevo producto se deberá de ir reflejando en la pantalla. Luego cree un botón que permite sumar el precio de cada uno de los productos y mostrar su respectivo resultado. Por ejemplo:

#	Producto	Precio
1	Manzanas	\$0.50
2	Naranjas	\$0.40
TOTAL		\$0.90

- Modifique el ejemplo de “Arreglo”, agregue un botón que le permita ordenar los elementos del arreglo de forma descendente, es decir de mayor a menor.

	UNIVERSIDAD DON BOSCO FACULTAD DE INGENIERÍA ESCUELA DE COMPUTACIÓN		CICLO 02-2023
	MATERIA	LENGUAJES INTERPRETADOS EN EL CLIENTE	GUIA DE LABORATORIO N° 6
	PRÁCTICA	ESTRUCTURAS DE CONTROL Y MATRICES	

V. INVESTIGACIÓN COMPLEMENTARIA

1. Investigue para qué se utilizan las siguientes funciones del objeto Math: abs(), round(), ceil(), floor(), exp(), log() y random().
2. Investigue sobre el API DOM en JavaScript y cree un ejemplo que permita agregar y eliminar elementos dentro de la etiqueta <body></body>.

VI. BIBLOGRAFÍA

- Flanagan, David. JavaScript La Guía Definitiva. 1ª Edición. Editorial ANAYA Multimedia / O'Reilly. 2007. Madrid, España.
- Terry McNavage. JavaScript Edición 2012. 1ª Edición. Editorial ANAYA Multimedia / Apress. Octubre 2011. Madrid, España.
- Tom Negrino / Dori Smith. JavaScript & AJAX Para Diseño Web. 6ª Edición. Editorial Pearson – Prentice Hall. 2007. Madrid España.
- Powell, Thomas / Schneider, Fritz. JavaScript Manual de Referencia. 1ra Edición. Editorial McGraw-Hill. 2002. Madrid, España.
- McFedries, Paul. JavaScript Edición Especial. 1ra Edición. Editorial Prentice Hall. 2002. Madrid, España.