



# ESCUELA SUPERIOR DE ECONOMÍA Y NEGOCIOS INGENIERÍA DE SOFTWARE Y NEGOCIOS DIGITALES

CICLO 03-2024

## GUIA DE LABORATORIO Nº 6

**Nombre de la practica:** Estructuras de control y funciones en JS

**Tiempo estimado:** 2 horas

**Materia:** Desarrollo Web I

### I. OBJETIVOS

Que al finalizar la practica el estudiante:

- Adquiera dominio en la construcción de funciones con JavaScript
- Haga uso de parámetros o argumentos en las funciones que realiza.
- Conozca el manejo de eventos usando funciones como controladores de eventos asociados a elementos del documento web.

### II. INTRODUCCIÓN TEORICA

#### 1. ¿Qué son las funciones?

Las funciones son uno de los bloques de construcción fundamentales en JavaScript. Una función en JavaScript es similar a un procedimiento — un conjunto de instrucciones que realiza una tarea o calcula un valor.

En JavaScript, las funciones son objetos de primera clase, es decir, son objetos y se pueden manipular y transmitir al igual que cualquier otro objeto. Concretamente son objetos **Function**.

Los parámetros en la llamada a una función son los argumentos de la función. Los argumentos se pasan a las funciones por valor.

#### 2. Declaración de una función tradicional

Una definición de función (también denominada declaración de función o expresión de función) consta de la palabra clave **function**, seguida de:

- El nombre de la función.
- Una lista de parámetros de la función, entre paréntesis y separados por comas.
- Las declaraciones de JavaScript que definen la función, encerradas entre llaves, { ... }.

Por ejemplo, el siguiente código define una función simple llamada cuadrado:

```
function cuadrado(numero) {  
    return numero * numero;  
}
```

La función cuadrado toma un parámetro, llamado número. La función consta de una declaración que dice devuelva el parámetro de la función (es decir, numero) multiplicado por sí mismo. La instrucción **return** especifica el valor devuelto por la función:

```
return numero * numero;
```

### 3. Llamado de una función

Definir una función no la ejecuta. Definirla simplemente nombra la función y especifica qué hacer cuando se llama a la función. Llamar a la función en realidad lleva a cabo las acciones especificadas con los parámetros indicados. Por ejemplo, si defines la función cuadrado, podrías llamarla de la siguiente manera:

```
cuadrado(5);
```

La declaración anterior llama a la función con un argumento de 5. La función ejecuta sus declaraciones y devuelve el valor 25.

Las funciones deben estar dentro del ámbito cuando se llaman, pero la declaración de la función se puede elevar (cuando aparece debajo de la llamada en el código), como en este ejemplo:

```
console.log(cuadrado(5));  
/* ... */  
function cuadrado(n) { return n * n }
```

### 4. Ámbito de una función

No se puede acceder a las variables definidas dentro de una función desde cualquier lugar fuera de la función, porque la variable se define solo en el ámbito de la función. Sin embargo, una función puede acceder a todas las variables y funciones definidas dentro del ámbito en el que está definida.

En otras palabras, una función definida en el ámbito global puede acceder a todas las variables definidas en el ámbito global. Una función definida dentro de otra función también puede acceder a todas las variables definidas en su función principal y a cualquier otra variable a la que tenga acceso la función principal.

```
// Las siguientes variables se definen en el ámbito global  
var num1 = 20,  
    num2 = 3,  
    name = 'Chamahk';  
  
// Esta función está definida en el ámbito global  
function multiply() {  
    return num1 * num2;  
}  
  
multiply(); // Devuelve 60  
  
// Un ejemplo de función anidada  
function getScore() {  
    var num1 = 2,  
        num2 = 3;  
  
    function add() {  
        return name + ' anotó ' + (num1 + num2);  
    }  
  
    return add();  
}  
  
getScore(); // Devuelve "Chamahk anotó 5"
```

## 5. Función constructora

La sintaxis para crear una función constructora es la siguiente:

```
let func = new Function ([arg1, arg2, ...argN], functionBody);
```

La función se crea con los argumentos arg1...argNy el cuerpo functionBody.

Es más fácil de entender mirando un ejemplo. Aquí hay una función con dos argumentos:

```
let sum = new Function('a', 'b', 'return a + b');  
  
alert( sum(1, 2) ); // 3
```

Y aquí hay una función sin argumentos, con solo el cuerpo de la función:

```
let saludo = new Function('alert("Hola")');  
  
saludo(); // Hola
```

La principal diferencia con respecto a otras formas que hemos visto es que la función se crea literalmente a partir de una cadena, que se pasa en tiempo de ejecución.

## 6. Objeto arguments

Arguments es un objeto similar a Array accesible dentro de funciones que contiene los valores de los argumentos pasados a esa función. El objeto arguments es una variable local disponible en todas las funciones que no son funciones flecha. Puedes hacer referencia a los argumentos de una función dentro de esa función utilizando su objeto arguments. Tiene entradas para cada argumento con el que se llamó a la función, con el índice de la primera entrada en 0.

Por ejemplo, si a una función se le pasan 3 argumentos, puedes acceder a ellos de la siguiente manera:

```
arguments[0] // primer argumento  
arguments[1] // segundo argumento  
arguments[2] // tercer argumento
```

También puedes establecer o reasignar cada argumento:

```
arguments[1] = 'nuevo valor';
```

El objeto arguments no es un Array. Es similar, pero carece de todas las propiedades de Array excepto de length.

## 7. Recursión de una función

Una función se puede referir y llamarse a sí misma. Hay tres formas de que una función se refiera a sí misma:

1. El nombre de la función
2. arguments.callee
3. Una variable dentro del ámbito que se refiere a la función

Por ejemplo, considera la siguiente definición de función:

```
var foo = function bar() {  
    // las instrucciones van aquí  
}
```

Dentro del cuerpo de la función, todos los siguientes son equivalentes:

1. bar()
2. arguments.callee()
3. foo()

Una función que se llama a sí misma se conoce como una función recursiva. En cierto modo, la recursividad es análoga a un bucle. Ambas ejecutan el mismo código varias veces y ambas requieren una condición (para evitar un bucle infinito, o más bien, una recursividad infinita en este caso).

Por ejemplo, el siguiente bucle:

```
var x = 0;  
while (x < 10) { // "x < 10" es la condición del bucle  
    // hacer cosas  
    x++;  
}
```

Se puede convertir en una declaración de función recursiva, seguida de una llamada a esa función:

```
function loop(x) {  
    if (x >= 10) // "x >= 10" es la condición de salida (equivalente a "!(x < 10)")  
        return;  
    // hacer cosas  
    loop(x + 1); // la llamada recursiva  
}  
loop(0);
```

## 8. Funciones anidadas

Puedes anidar una función dentro de otra función. La función anidada (interna) es privada de su función contenedora (externa).

También forma un cierre. Un cierre es una expresión (comúnmente, una función) que puede tener variables libres junto con un entorno que une esas variables (que "cierra" la expresión).

Dado que una función anidada es un cierre, significa que una función anidada puede "heredar" los argumentos y variables de su función contenedora. En otras palabras, la función interna contiene el ámbito de la función externa.

Para resumir:

- Solo se puede acceder a la función interna desde declaraciones en la función externa.
- La función interna forma un cierre: la función interna puede usar los argumentos y variables de la función externa, mientras que la función externa no puede usar los argumentos y variables de la función interna.

El siguiente ejemplo muestra funciones anidadas:

```
function addSquares(a, b) {
  function square(x) {
    return x * x;
  }
  return square(a) + square(b);
}
a = addSquares(2, 3); // devuelve 13
b = addSquares(3, 4); // devuelve 25
c = addSquares(4, 5); // devuelve 41
```

## 9. Funciones anónimas o literales de función

Si bien la declaración de función anterior sintácticamente es una declaración, las funciones también se pueden crear mediante una expresión function. Esta función puede ser anónima, no tiene por qué tener un nombre. Por ejemplo, la función cuadrado se podría haber definido como:

```
const cuadrado = function(numero) { return numero * numero }
var calculo = cuadrado(4) // calculo obtiene el valor 16
```

Sin embargo, puedes proporcionar un nombre con una expresión function. Proporcionar un nombre permite que la función se refiera a sí misma y también facilita la identificación de la función en el seguimiento de la pila de un depurador:

```
const factorial = function fac(n) { return n < 2 ? 1 : n * fac(n - 1) }
console.log(factorial(3))
```

Las expresiones function son convenientes cuando se pasa una función como argumento a otra función. El siguiente ejemplo muestra una función map que debería recibir una función como primer argumento y un arreglo como segundo argumento.

```
function map(f, a) {
  let result = []; // Crea un nuevo arreglo
  let i; // Declara una variable
  for (i = 0; i !== a.length; i++) result[i] = f(a[i]);
  return result;
}
```

En el siguiente código, la función recibe una función definida por una expresión de función y la ejecuta por cada elemento del arreglo recibido como segundo argumento.

```
function map(f, a) {
  let result = []; // Crea un nuevo arreglo
  let i; // Declara una variable
  for (i = 0; i !== a.length; i++)
    result[i] = f(a[i]);
  return result;
}
const f = function(x) {
  return x * x * x;
}
let numbers = [0, 1, 2, 5, 10];
let cube = map(f, numbers);
console.log(cube);
```

La función devuelve: [0, 1, 8, 125, 1000].

En JavaScript, una función se puede definir en función de una condición. Por ejemplo, la siguiente definición de función define miFuncion solo si numero es igual a 0:

```
var miFuncion;  
if (numero === 0) {  
  miFuncion = function(objeto) {  
    objeto.marca = 'Toyota';  
  }  
}
```

## 10. Funciones de Flecha

Una expresión de función flecha es una alternativa compacta a una expresión de función tradicional, pero es limitada y no se puede utilizar en todas las situaciones.

### Diferencias y limitaciones:

- No tiene sus propios enlaces a this o super y no se debe usar como métodos.
- No tiene argumentos o palabras clave new.target (pseudo-propiedad que permite detectar si una función o constructor fue llamado usando el operador new).
- No apta para los métodos call, apply y bind, que generalmente se basan en establecer un ámbito o alcance.
- No se puede utilizar como constructor.
- No se puede utilizar yield (se usa para pausar y reanudar una función generadora) dentro de su cuerpo.

```
const materials = [  
  'Hydrogen',  
  'Helium',  
  'Lithium',  
  'Beryllium'  
];  
  
console.log(materials.map(material => material.length));
```

### Comparación de funciones tradicionales con funciones flecha

Observa, paso a paso, la descomposición de una "función tradicional" hasta la "función flecha" más simple.

Nota: Cada paso mostrado es una "función flecha" válida.

```
// Función tradicional
function (a){
    return a + 100;
}

// Desglose de la función flecha

// 1. Elimina la palabra "function" y coloca la flecha entre el argumento y el corchete de apertura.
(a) => {
    return a + 100;
}

// 2. Quita los corchetes del cuerpo y la palabra "return" – el return está implícito.
(a) => a + 100;

// 3. Suprime los paréntesis de los argumentos
a => a + 100;
```

Por ejemplo, si tienes varios argumentos o ningún argumento, deberás volver a introducir paréntesis alrededor de los argumentos:

```
// Función tradicional
function (a, b){
    return a + b + 100;
}

// Función flecha
(a, b) => a + b + 100;

// Función tradicional (sin argumentos)
let a = 4;
let b = 2;
function (){
    return a + b + 100;
}

// Función flecha (sin argumentos)
let a = 4;
let b = 2;
() => a + b + 100;
```

Del mismo modo, si el cuerpo requiere líneas de procesamiento adicionales, deberás volver a introducir los corchetes más el "return":

```
// Función tradicional
function (a, b){
    let chuck = 42;
    return a + b + chuck;
}

// Función flecha
(a, b) => {
    let chuck = 42;
    return a + b + chuck;
}
```

Y finalmente, en las funciones con nombre tratamos las expresiones de flecha como variables:

```
// Función tradicional
function bob (a){
    return a + 100;
}

// Función flecha
let bob = a => a + 100;
```

## 11. Funciones predefinidas

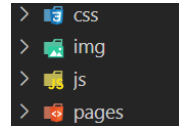
JavaScript tiene integradas varias funciones de nivel superior, se muestra algunas en el siguiente cuadro:

| #  | Función              | Descripción   |
|----|----------------------|---|
| 1  | eval()               | El método eval() evalúa el código JavaScript representado como una cadena.  |
| 2  | uneval()             | El método uneval() crea una representación de cadena del código fuente de un Object.  |
| 3  | isFinite()           | La función global isFinite() determina si el valor pasado es un número finito. Si es necesario, el parámetro, primero se convierte en un número.  |
| 4  | isNaN()              | La función isNaN() determina si un valor es NaN o no.   |
| 5  | parseFloat()         | La función parseFloat() procesa un argumento de cadena y devuelve un número de punto flotante.  |
| 6  | parseInt()           | La función parseInt() procesa un argumento de cadena y devuelve un número entero de la base especificada (la base en los sistemas numéricos matemáticos).   |
| 7  | decodeURI()          | La función decodeURI() decodifica un identificador uniforme de recursos (URI) creado previamente por encodeURI o por una rutina similar.  |
| 8  | decodeURIComponent() | El método decodeURIComponent() decodifica un componente Identificador uniforme de recursos (URI) creado previamente por encodeURIComponent o por un rutina similar.   |
| 9  | encodeURIComponent() | El método encodeURIComponent() codifica un identificador uniforme de recursos (URI) reemplazando cada instancia de ciertos caracteres por una, dos, tres o cuatro secuencias de escape que representan la codificación UTF-8 del carácter (solo habrá cuatro secuencias de escape para caracteres compuestos por dos caracteres "sustitutos").            |
| 10 | encodeURIComponent() | El método encodeURIComponent() codifica un componente Identificador uniforme de recursos (URI) reemplazando cada instancia de ciertos caracteres por una, dos, tres o cuatro secuencias de escape que representan la codificación UTF-8 del carácter (solo habrá cuatro secuencias de escape para caracteres compuestos por dos caracteres "sustitutos"). |
| 11 | escape()             | El método obsoleto escape() calcula una nueva cadena en la que ciertos caracteres han sido reemplazados por una secuencia de escape hexadecimal. En su lugar usa encodeURI o encodeURIComponent.  |
| 12 | unescape()           | El método obsoleto unescape() calcula una nueva cadena en la que las secuencias de escape hexadecimales se reemplazan con el carácter que representan. Las secuencias de escape se pueden introducir por medio de una función como escape. Debido a que unescape() está en desuso, usa decodeURI() o decodeURIComponent en su lugar.                      |



## II. 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.



### PARTE I: ESTRUCTURAS DE CONTROL Y ARREGLOS

#### Ejemplo 1: Creando el índice de nuestro sitio web

1. Cree un archivo index.html en la carpeta raíz de su proyecto y coloque el siguiente código:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8" />
  <meta http-equiv="X-UA-Compatible" content="IE=edge" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <link href="bootstrap/css/bootstrap.min.css" rel="stylesheet" />
  <link href="css/styles.css" rel="stylesheet" />
  <link rel="icon" href="img/js.png" sizes="32x32" type="image/png" />
  <title>Utilizando estructuras de control</title>
</head>
<body>
  <main class="d-flex flex-nowrap">
    <div class="d-flex flex-column flex-shrink-0 p-3 text-bg-dark nav-bar">
      <a href="index.html"
        class="d-flex align-items-center mb-3 mb-md-0 me-md-auto text-white text-decoration-none">
        

        <span class="fs-4">JavaScript</span>
      </a>
      <hr />
      <ul class="nav nav-pills flex-column mb-auto">
        <li class="nav-item">
          <a href="pages/registroAcademico.html" class="nav-link text-white">
            <i class="bi bi-file-spreadsheet" role="img" aria-label="RegistroAcademico"></i>
            Registro académico
          </a>
        </li>
        <li>
          <a href="pages/tablasMultiplicar.html" class="nav-link text-white">
            <i class="bi bi-person-workspace" role="img" aria-label="TablasMultiplicar"></i>
            Tablas de multiplicar
          </a>
        </li>
        <li>
          <a href="pages/promedioNotas.html" class="nav-link text-white">
            <i class="bi bi-file-earmark-medical" role="img" aria-label="Promedio"></i>
            Promedio notas
          </a>
        </li>
        <li>
          <a href="pages/arreglo.html" class="nav-link text-white">
            <i class="bi bi-box" role="img" aria-label="Arreglo"></i>
```

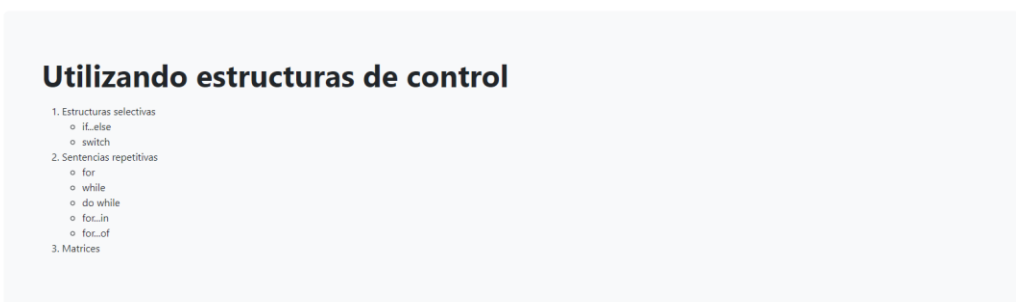
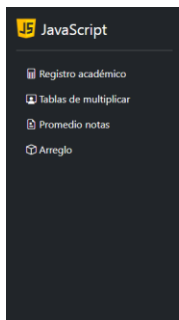
```

        Arreglo
    </a>
</li>
</ul>
</div>
<div class="d-flex flex-column flex-shrink-0 p-3 column-container">
    <div class="p-5 mb-2 bg-light rounded-3">
        <div class="container-fluid py-4">
            <h1 class="display-5 fw-bold">Utilizando estructuras de control</h1>
            <p class="col-md-8 fs-4">
                <ol>
                    <li>
                        Estructuras selectivas
                        <ul>
                            <li>if...else</li>
                            <li>switch</li>
                        </ul>
                    </li>
                    <li>
                        Sentencias repetitivas
                        <ul>
                            <li>for</li>
                            <li>while</li>
                            <li>do while</li>
                            <li>for...in</li>
                            <li>for...of</li>
                        </ul>
                    </li>
                    <li>Matrices </li>
                </ol>
            </p>
        </div>
    </div>
</div>
</main>

<script src="bootstrap/js/bootstrap.bundle.min.js"></script>
</body>
</html>

```

## 2. Visualice el resultado de su página index.html



## 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
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         </form>
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.

```
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 }
```

```

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  |

### 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);
```

```

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 |

#### 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
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>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>
72   <div class="col-md-12 py-5">
73     <div id="idContainerEstudiantes"></div>
74   </div>
75 </div>
76 </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.

```
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);
```

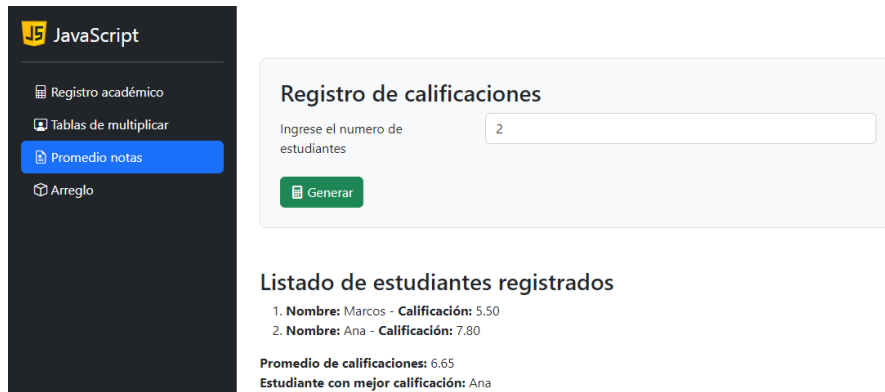


```

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 }

```

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



## Ejemplo 5: Trabajando con arreglos

1. 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

```

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>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                     </div>
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>

```

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.

```

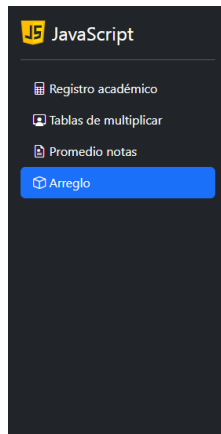
91     <script src="../../bootstrap/js/bootstrap.bundle.min.js"></script>
92     <script src="../../js/arreglo.js" defer></script>

```

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 }
```

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



## PARTE II: FUNCIONES

### Ejemplo 1: Creando el índice de nuestro sitio web

1. Cree un archivo indexFunciones.html en la carpeta raíz de su proyecto y proceda a agregar el siguiente código:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8" />
  <meta http-equiv="X-UA-Compatible" content="IE=edge" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <link href="bootstrap/css/bootstrap.min.css" rel="stylesheet" />
  <link href="css/stylesFunciones.css" rel="stylesheet" />
  <link rel="icon" href="img/js.png" sizes="32x32" type="image/png" />
  <title>Utilizando funciones</title>
</head>
<body>
  <header>
    <div class="px-3 py-2 text-bg-dark fixed-top">
      <div class="container">
        <div class="d-flex flex-wrap align-items-center justify-content-center justify-content-lg-
start">
          <a href="index.html"
            class="d-flex align-items-center my-2 my-lg-0 me-lg-auto text-white text-decoration-
none">
            
          </a>

          <ul class="nav col-12 col-lg-auto my-2 justify-content-center my-md-0 text-small nav-bar">
            <li class="text-center">
              <a href="./pages/recursividad.html" class="nav-link text-white">
                <i class="bi bi-arrow-clockwise bi-2 d-block"></i>
                Recursividad
              </a>
            </li>
            <li class="text-center">
              <a href="./pages/tabla.html" class="nav-link text-white">
                <i class="bi bi-table bi-2 d-block"></i>
                Creando una Tabla
              </a>
            </li>
            <li class="text-center">
              <a href="./pages/formulario.html" class="nav-link text-white">
                <i class="bi bi-ui-checks bi-2 d-block"></i>
```

```

        Formulario
    </a>
</li>
</ul>
</div>
</div>
</div>
</header>
<div class="container-md">
    <div class="row">
        <div class="col-12">
            <div class="shadow-lg p-3 mt-5 bg-body rounded">
                <h3>Utilizando funciones</h3>
                <p>
                    Las funciones son los principales “bloques de construcción” del
                    programa. Permiten que el código se llame muchas veces sin
                    repetición. Ya hemos visto ejemplos de funciones integradas, como
                    alert(message), prompt(message, default) y confirm(question). Pero
                    también podemos crear funciones propias.
                </p>
                <h3>Declaración de una función básica</h3>
                <p>
                    Para crear una función podemos usar una declaración de
                    <span class="fw-bold">función</span>.
                    <br />
                    <br />
                    <span class="text-danger fw-bold">function</span>
                    <span class="text-success fw-bold"> saludar()</span>
                    <span class="text-danger fw-bold">{</span>
                    <br />
                    <span class="text-primary fw-bold mx-4">alert('¡Hola a todos!');</span>
                    <br />
                    <span class="text-danger fw-bold">}</span>
                    <br />
                    <br />
                    La palabra clave function va primero, luego va el nombre de función,
                    luego una lista de parámetros entre paréntesis (separados por comas, vacía en el
ejemplo anterior)
                    y finalmente el código de la función entre llaves, también llamado “el cuerpo de la
función”.
                </p>
            </div>
        </div>
    </div>
</div>
</div>
<script src="bootstrap/js/bootstrap.bundle.min.js"></script>
</body>
</html>

```

2. Visualice el resultado de su página index.html

JS

Recursividad

Creando una Tabla

Formulario

### Utilizando funciones

Las funciones son los principales "bloques de construcción" del programa. Permiten que el código se llame muchas veces sin repetición. Ya hemos visto ejemplos de funciones integradas, como `alert(message)`, `prompt(message, default)` y `confirm(question)`. Pero también podemos crear funciones propias.

#### Declaración de una función básica

Para crear una función podemos usar una declaración de **función**.

```
function saludar() {
  alert('¡Hola a todos!');
}
```

La palabra clave `function` va primero, luego va el nombre de función, luego una lista de parámetros entre paréntesis (separados por comas, vacía en el ejemplo anterior) y finalmente el código de la función entre llaves, también llamado "el cuerpo de la función".

## Ejemplo 2: Utilizando recursividad

- Utilice el archivo `recursividad.html` que ha sido proporcionado, localice la siguiente sección:

```

48 <div class="container-lg">
49   <!-- CONTINUAR CON EL CODIGO AQUI -->
50 </div>

```

- Desarrolle el siguiente código dentro del contenedor anterior.

```

49 <div class="row">
50   <div class="col-12">
51     <div class="shadow-lg p-3 mt-5 bg-body rounded">
52       <div class="row">
53         <div class="col-12">
54           <h3>Calculando el factorial de un número</h3>
55
56           <form id="idForm" class="row row-cols-lg-auto g-3 align-items-center">
57             <div class="col-12">
58               <input type="text" class="form-control" id="idTxtNumero"
59                 placeholder="Escriba un número" />
60             </div>
61
62             <div class="col-12">
63               <button id="idBtnCalcular" type="button" class="btn btn-primary">
64                 <i class="bi bi-calculator"></i> Calcular
65               </button>
66             </div>
67           </form>
68         </div>
69         <div class="col-12">
70           <div id="idDivResultado" class="fs-1 fw-bold p-3"></div>
71         </div>
72       </div>
73     </div>
74   </div>
75 </div>

```

- Observe que en la parte final de nuestro código HTML se encuentran dos etiquetas `<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/recursividad.js" defer></script>

```


- Ubíquese en el archivo `recursividad.js` y comience a desarrollar el siguiente código fuente.

```

1 // Otra forma de acceder a un elemento HTML es utilizando el getElementById del DOM
2 // Notesé que para este caso no se antepone el carácter #
3 const campo = document.getElementById("idTxtNumero");
4
5 //definamos una funcion anonima que permita validar en tiempo real el ingreso de un numero
6 const validarNumero = function (e) {
7   //creamos una expresion regular que valida que sean numeros
8   let validar = /^[0-9]{1}$/;
9   let tecla = e.key;
10
11   /*
12   .test válida que la expresión regular coicida con el valor ingresado
13   podrá observar que al intentar teclar un letra u otro caracter diferente
14   a un número este no se escribe en el campo
15   */
16   if (!validar.test(tecla)) e.preventDefault();
17 };
18
19 //definiendo el evento keypress para el campo
20 campo.addEventListener("keypress", validarNumero);
21
22 //Trabajando con el boton Calcular
23 const boton = document.getElementById("idBtnCalcular");
24
25 //Definiendo una funcion anonima para calcular el factorial de un numero
26 function calcularFactorial(numero) {
27   return numero < 2 ? 1 : numero * calcularFactorial(numero - 1);
28 }
29
30 //Definamos una funcion de tipo flecha para imprimir el resultado del factorial
31 const imprimir = (numero, resultado) => {
32   const contenedor = document.getElementById("idDivResultado");
33   contenedor.innerHTML = `El factorial de ${numero}! es ${resultado}`;
34 };
35
36 // Definiendo una funcion tradicional
37 function calcular() {
38   let numero = document.getElementById("idTxtNumero").value;
39   if (numero != "") {
40     //Llamamos a la funcion anonima para que calcule el factorial
41     let resultado = calcularFactorial(numero);
42     //Enviando el resultado a una funcion de tipo flecha
43     imprimir(numero, resultado);
44   } else {
45     alert("Debe ingresar un numero válido");
46   }
47 }
48
49 //definiendo el evento click para el boton
50 boton.addEventListener("click", calcular);

```

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



Recursividad

Creando una Tabla

Formulario

Calculando el factorial de un número

### Ejemplo 3: Creando tabla HTML con funciones de JavaScript

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

```
48 <div class="container-lg">
49   <!-- CONTINUAR CON EL CODIGO AQUI -->
50 </div>
```

2. Desarrolle el siguiente código dentro del contenedor anterior.

```
49 <div class="row">
50   <div class="col-12">
51     <div class="shadow-lg p-3 mt-5 bg-body rounded">
52       <div class="row">
53         <div class="col-12">
54           <h3>Generando tabla con funciones</h3>
55           <button id="idBtnNewTable" type="button" class="btn btn-success" data-bs-toggle="modal"
56             data-bs-target="#idModal">
57             <i class="bi bi-table"></i> Crear tabla
58           </button>
59           <div class="col-12">
60             <div id="idDivResultado" class="fs-6 p-3"></div>
61           </div>
62         </div>
63       </div>
64     </div>
65   </div>
66 </div>

67 <!-- COMPONENTE MODAL DE BOOTSTRAP-->
68 <div class="modal fade" id="idModal" tabindex="-1" aria-labelledby="modalLabel" aria-hidden="true">
69   <div class="modal-dialog">
70     <div class="modal-content">
71       <div class="modal-header">
72         <h1 class="modal-title fs-5" id="modalLabel">Definición de tabla</h1>
73         <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>
74       </div>
75       <div class="modal-body">
76         <form id="idForm">
77           <div class="form-floating mb-3">
78             <input type="number" class="form-control" id="idNumFila" placeholder="Número de filas"
79               value="1" min="1">
80             <label for="idNumFila">
81               <i class="bi bi-columns"></i> Número de filas
82             </label>
83           </div>
84           <div class="form-floating">
85             <input type="number" class="form-control" id="idNumColumnas"
86               placeholder="Número de columnas" value="1" min="1">
87             <label for="idNumColumnas">
88               <i class="bi bi-layout-three-columns"></i> Número de columnas
89             </label>
90           </div>
91         </form>
92       </div>
93       <div class="modal-footer">
94         <button type="button" class="btn btn-secondary" data-bs-dismiss="modal">
95           <i class="bi bi-x-circle"></i> Cerrar
96         </button>
97         <!-- Se esta llamando una funcion directamente
98         desde el boton por medio del evento onclick-->
99         <button type="button" class="btn btn-primary" onclick="crearTabla()" id="idBtnCrearTabla"
100           data-bs-dismiss="modal">
101           <i class="bi bi-plus-circle"></i> Crear tabla
102         </button>
103       </div>
104     </div>
105   </div>
106 </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.

```
108 <script src="../../bootstrap/js/bootstrap.bundle.min.js"></script>
109 <script src="../../js/tabla.js" defer></script>
```

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

```
1 //Genera fila
2 const generarFila = (tipo, fila, columnas) => {
3   let tr = `<tr>`;
4   for (let c = 0; c <= columnas; c++) {
5     //imprimiendo encabezados
6     if (tipo == 1) {
7       if (c == 0) {
8         tr += `<th scope="col" class="text-center">#</th>`;
9       } else {
10        tr += `<th scope="col" class="text-center">Titulo ${c}</th>`;
11      }
12    } else {
13      if (c == 0) {
14        tr += `<td scope="row" class="text-center fw-bold text-success">Fila ${fila}</td>`;
15      } else {
16        tr += `<td class="text-center">Celda ${fila},${c}</td>`;
17      }
18    }
19  }
20  return (tr += `</tr>`);
21 };
22
23 //Diseñando tabla
24 const generarTabla = (filas, columnas) => {
25   let tabla = `
26   <div class="table-responsive">
27   <table class="table table-striped table-hover table-bordered">`;
28
29   //Recorriendo el numero de filas
30   for (let i = 0; i <= filas; i++) {
31     // Para imprimir los titulos de la tabla
32     if (i == 0) {
33       tabla += generarFila(1, i, columnas);
34     }
35     //generando encabezados y cuerpo de la tabla
36     else {
37       tabla += generarFila(2, i, columnas);
38     }
39   }
40   tabla += `</table></div>`;
41   return tabla;
42 };
43
```

```

44 // Las funciones que se utilizaran serán llamadas desde HTML
45 // Por medio del evento onclick en el boton con ID= idBtnCrearTabla
46 const crearTabla = function () {
47   // capturamos los valores de los campos
48   let columnas = document.getElementById("idNumColumnas").value;
49   let filas = document.getElementById("idNumFila").value;
50
51   //validamos que la información sea correcta
52   if (columnas != "" && filas != "") {
53     const contenedor = document.getElementById("idDivResultado");
54     contenedor.innerHTML = generarTabla(filas, columnas);
55     console.log(generarTabla(filas, columnas));
56   } else {
57     alert("No se pudo crear la tabla, no se completaron los datos");
58   }
59 };

```

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


Recursividad
Creando una Tabla
Formulario

Generando tabla con funciones

 Crear tabla

| #      | Titulo 1  | Titulo 2  | Titulo 3  | Titulo 4  | Titulo 5  | Titulo 6  | Titulo 7  | Titulo 8  | Titulo 9  | Titulo 10  |
|--------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|------------|
| Fila 1 | Celda 1,1 | Celda 1,2 | Celda 1,3 | Celda 1,4 | Celda 1,5 | Celda 1,6 | Celda 1,7 | Celda 1,8 | Celda 1,9 | Celda 1,10 |
| Fila 2 | Celda 2,1 | Celda 2,2 | Celda 2,3 | Celda 2,4 | Celda 2,5 | Celda 2,6 | Celda 2,7 | Celda 2,8 | Celda 2,9 | Celda 2,10 |

#### Ejemplo 4: Interactuando con funciones desde un formulario

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

```

48 <div class="container-lg">
49   <!-- CONTINUAR CON EL CODIGO AQUI -->
50 </div>

```

- Desarrolle el siguiente código dentro del contenedor anterior.

```

49 <div class="row">
50   <div class="col-12 shadow-lg p-3 mt-5 bg-body rounded">
51     <h3>Expediente médico</h3>
52     <form id="idForm">
53       <div class="row">
54         <div class="col-lg-6">
55           <div class="form-floating mb-3">
56             <input type="text" class="form-control" id="idTxtNombre"
57               placeholder="Nombre del paciente" />
58             <label for="idTxtNombre">
59               <i class="bi bi-person-circle"></i> Nombre del paciente
60             </label>
61           </div>
62         </div>

```

```

63     <div class="col-lg-6">
64         <div class="form-floating">
65             <input type="text" class="form-control" id="idTxtApellido"
66                 placeholder="Apellido del paciente">
67             <label for="idTxtApellido">
68                 <i class="bi bi-person-circle"></i> Apellido del paciente
69             </label>
70         </div>
71     </div>
72 </div>
73 <div class="row">
74     <div class="col-lg-6">
75         <div class="form-floating">
76             <input type="date" class="form-control" id="idTxtFechaNacimiento"
77                 placeholder="Fecha de nacimiento" pattern="\d{2}-\d{2}-\d{4}">
78             <label for="idTxtFechaNacimiento">
79                 <i class="bi bi-calendar-date"></i> Fecha nacimiento
80             </label>
81         </div>
82     </div>
83     <div class="col-lg-6">
84         <div class="form-check">
85             <input class="form-check-input" type="radio" name="rdSexo" id="idRdMasculino">
86             <label class="form-check-label" for="idRdMasculino">
87                 Hombre
88             </label>
89         </div>
90         <div class="form-check">
91             <input class="form-check-input" type="radio" name="rdSexo" id="idRdFemenino">
92             <label class="form-check-label" for="idRdFemenino">
93                 Mujer
94             </label>
95         </div>
96     </div>
97 </div>
98 <div class="row">
99     <div class="col-lg-6 pt-3">
100         <select class="form-select" aria-label="pais" id="idCmbPais">
101             <option value="0" selected>Seleccione un Pais</option>
102             <option value="1">El Salvador</option>
103             <option value="2">Honduras</option>
104             <option value="3">Panama</option>
105             <option value="4">Costa Rica</option>
106             <option value="5">Belice</option>
107             <option value="6">Nicaragua</option>
108         </select>
109     </div>
110 </div>
111 <div class="row">
112     <div class="col-lg-12 pt-3">
113         <div class="form-floating">
114             <textarea class="form-control" placeholder="Direccion" id="idTxtDireccion"
115                 rows="4"></textarea>
116             <label for="idTxtDireccion">
117                 <i class="bi bi-geo"></i> Direccion
118             </label>
119         </div>
120     </div>
121 </div>

```

```

122         <div class="row">
123             <div class="col-lg-12 pt-3">
124                 <button id="idBtnAgregar" type="button" class="btn btn-success">
125                     <i class="bi bi-person-plus-fill"></i> Guardar Datos
126                 </button>
127                 <button id="idBtnMostrar" type="button" class="btn btn-warning">
128                     <i class="bi bi-people"></i> Mostrar personas
129                 </button>
130                 <button id="idBtnLimpiar" type="button" class="btn btn-danger">
131                     <i class="bi bi-trash"></i> Limpiar formulario
132                 </button>
133                 <button id="idBtnAgregarPais" type="button" class="btn btn-primary" data-bs-toggle="modal"
134                     data-bs-target="#idModal">
135                     <i class="bi bi-map-fill"></i> Nuevo Pais
136                 </button>
137             </div>
138         </div>
139     </form>
140 </div>
141
142     <div class="col-12 shadow-lg p-3 mt-5 bg-body rounded">
143         <h3>Pacientes registrados</h3>
144         <div id="idTablaPacientes">
145             Ninguno
146         </div>
147     </div>
148 </div>
149
150
151
152 <!-- COMPONENTE MODAL DE BOOTSTRAP-->
153 <div class="modal fade" id="idModal" tabindex="-1" aria-labelledby="modallabel" aria-hidden="true">
154     <div class="modal-dialog">
155         <div class="modal-content">
156             <div class="modal-header">
157                 <h1 class="modal-title fs-5" id="modallabel">Agregando nuevo país</h1>
158                 <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>
159             </div>
160             <div class="modal-body">
161                 <form id="idFormPais">
162                     <div class="form-floating mb-3">
163                         <input type="text" class="form-control" id="idNombrePais" placeholder="Nombre del país">
164                         <label for="idNombrePais">
165                             <i class="bi bi-map"></i> Nombre del país
166                         </label>
167                     </div>
168                 </form>
169             </div>
170             <div class="modal-footer">
171                 <button type="button" class="btn btn-primary" id="idBtnAddPais" data-bs-dismiss="modal">
172                     <i class="bi bi-plus-circle"></i> Agregar país
173                 </button>
174             </div>
175         </div>
176     </div>
177 </div>
178
179 <!-- PERMITE MOSTRAR NOTIFICACIONES -->
180 <div class="toast-container top-0 start-50 translate-middle-x pt-4 mt-1" style="z-index: 1035;">
181     <div id="idNotificacion" class="toast text-bg-success" role="alert" aria-live="assertive"
182         aria-atomic="true">
183         <div class="toast-header">
184             <i class="bi bi-bell"></i>
185             <strong class="me-auto">Notificacion</strong>
186             <small>Hace un momento</small>
187             <button type="button" class="btn-close" data-bs-dismiss="toast" aria-label="Close"></button>
188         </div>
189         <div class="toast-body" id="idMensaje">
190
191     </div>
192 </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.

```
193 <script src="../../bootstrap/js/bootstrap.bundle.min.js"></script>
194 <script src="../../js/formulario.js" defer></script>
```

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

```
1 //Accediendo a los elementos html
2 const inputNombre = document.getElementById("idTxtNombre");
3 const inputApellido = document.getElementById("idTxtApellido");
4 const inputFechaNacimiento = document.getElementById("idTxtFechaNacimiento");
5 const inputRdMasculino = document.getElementById("idRdMasculino");
6 const inputRdFemenino = document.getElementById("idRdFemenino");
7 const cmbPais = document.getElementById("idCmbPais");
8 const inputDireccion = document.getElementById("idTxtDireccion");
9 const inputNombrePais = document.getElementById("idNombrePais");
10
11 const buttonAgregarPaciente = document.getElementById("idBtnAgregar");
12 const buttonLimpiarPaciente = document.getElementById("idBtnLimpiar");
13 const buttonMostrarPaciente = document.getElementById("idBtnMostrar");
14 const buttonAgregarPais = document.getElementById("idBtnAddPais");
15
16 const notificacion = document.getElementById("idNotificacion");
17 // Componente de Bootstrap
18 const toast = new bootstrap.Toast(notificacion);
19 const mensaje = document.getElementById("idMensaje");
20
21 //Componente modal
22 const idModal = document.getElementById("idModal");
23
24 //Arreglo global de pacientes
25 let arrayPaciente = [];
26
27 /*
28 Creando una funcion para que limpie el formulario
29 siempre que se cargue la pagina o cuando se presione
30 el boton limpiar del formulario
31 */
32
33 const limpiarForm = () => {
34     inputNombre.value = "";
35     inputApellido.value = "";
36     inputFechaNacimiento.value = "";
37     inputRdMasculino.checked = false;
38     inputRdFemenino.checked = false;
39     cmbPais.value = 0;
40     inputDireccion.value = "";
41     inputNombrePais.value = "";
42
43     inputNombre.focus();
44 };
```

```

46  /*
47  | Funcion para validar el ingreso del paciente
48  */
49
50  const addPaciente = function () {
51      let nombre = inputNombre.value;
52      let apellido = inputApellido.value;
53      let fechaNacimiento = inputFechaNacimiento.value;
54      let sexo =
55          inputRdMasculino.checked == true
56          ? "Hombre"
57          : inputRdFemenino.checked == true
58          ? "Mujer"
59          : "";
60      let pais = cmbPais.value;
61      let labelPais = cmbPais.options[cmbPais.selectedIndex].text;
62      let direccion = inputDireccion.value;
63
64      if (
65          nombre != "" &&
66          apellido != "" &&
67          fechaNacimiento != "" &&
68          sexo != "" &&
69          pais != 0 &&
70          direccion != ""
71      ) {
72          //Agregando informacion al arreglo paciente
73          arrayPaciente.push(
74              new Array(nombre, apellido, fechaNacimiento, sexo, labelPais, direccion)
75          );
76
77          //Asignando un mensaje a nuestra notificacion
78          mensaje.innerHTML = "Se ha registrado un nuevo paciente";
79          //Llamando al componente de Bootstrap
80          toast.show();
81
82          //Limpiando formulario
83          limpiarForm();
84      } else {
85          //Asignando un mensaje a nuestra notificacion
86          mensaje.innerHTML = "Faltan campos por completar";
87          //Llamando al componente de Bootstrap
88          toast.show();
89      }
90  };
91

```

```

92 //Funcion que imprime la ficha de los pacientes registrados
93 function imprimirFilas() {
94     let $fila = "";
95     let contador = 1;
96
97     arrayPaciente.forEach((element) => {
98         $fila += `<tr>
99             <td scope="row" class="text-center fw-bold">${contador}</td>
100             <td>${element[0]}</td>
101             <td>${element[1]}</td>
102             <td>${element[2]}</td>
103             <td>${element[3]}</td>
104             <td>${element[4]}</td>
105             <td>${element[5]}</td>
106             <td>
107                 <button id="idBtnEditar${contador}" type="button" class="btn btn-primary" alt="Eliminar">
108                     <i class="bi bi-pencil-square"></i>
109                 </button>
110                 <button id="idBtnEliminar${contador}" type="button" class="btn btn-danger" alt="Editar">
111                     <i class="bi bi-trash3-fill"></i>
112                 </button>
113             </td>
114         </tr>`;
115         contador++;
116     });
117     return $fila;
118 }
119
120 const imprimirPacientes = () => {
121     let $table = `<div class="table-responsive">
122         <table class="table table-striped table-hover table-bordered">
123             <tr>
124                 <th scope="col" class="text-center" style="width:5%">#</th>
125                 <th scope="col" class="text-center" style="width:15%">Nombre</th>
126                 <th scope="col" class="text-center" style="width:15%">Apellido</th>
127                 <th scope="col" class="text-center" style="width:10%">Fecha nacimiento</th>
128                 <th scope="col" class="text-center" style="width:10%">Sexo</th>
129                 <th scope="col" class="text-center" style="width:10%">País</th>
130                 <th scope="col" class="text-center" style="width:25%">Dirección</th>
131                 <th scope="col" class="text-center" style="width:10%">Opciones</th>
132             </tr>
133             ${imprimirFilas()}
134         </table>
135     </div>
136     `;
137     document.getElementById("idTablaPacientes").innerHTML = $table;
138 };

```

```



140 // Contador global de los option correspondiente
141 // al select (cmb) pais
142 let contadorGlobalOption = cmbPais.children.length;
143 const addPais = () => {
144     let paisNew = inputNombrePais.value;
145
146     if (paisNew !== "") {
147         // Creando nuevo option con la API DOM
148         let option = document.createElement("option");
149         option.textContent = paisNew;
150         option.value = contadorGlobalOption + 1;
151
152         //Agregando el nuevo option en el select
153         cmbPais.appendChild(option);
154
155         //Asignando un mensaje a nuestra notificacion
156         mensaje.innerHTML = "Pais agregado correctamente";
157         //Llamando al componente de Bootstrap
158         toast.show();
159     } else {
160         //Asignando un mensaje a nuestra notificacion
161         mensaje.innerHTML = "Faltan campos por completar";
162         //Llamando al componente de Bootstrap
163         toast.show();
164     }
165 };
166
167 // Agregando eventos a los botones y utilizando funciones tipo flecha
168 buttonLimpiarPaciente.onclick = () => {
169     limpiarForm();
170 };
171
172 buttonAgregarPaciente.onclick = () => {
173     addPaciente();
174 };
175
176 buttonMostrarPaciente.onclick = () => {
177     imprimirPacientes();
178 };
179
180 buttonAgregarPais.onclick = () => {
181     addPais();
182 };
183
184 // Se agrega el focus en el campo nombre pais del modal
185 idModal.addEventListener("shown.bs.modal", () => {
186     inputNombrePais.value = "";
187     inputNombrePais.focus();
188 });
189
190 //Ejecutar funcion al momento de cargar la pagina HTML
191 limpiarForm();

```



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

The screenshot displays a web application interface. At the top, there is a navigation bar with a 'JS' logo and three buttons: 'Recursividad', 'Creando una Tabla', and 'Formulario'. The main content area is divided into two sections. The upper section, titled 'Expediente médico', contains a form with fields for 'Nombre del paciente', 'Apellido del paciente', 'Fecha nacimiento' (with a date picker), 'Selección un País' (a dropdown menu), and 'Dirección'. Below these fields are four buttons: 'Guardar Datos', 'Mostrar personas', 'Limpiar formulario', and 'Nuevo País'. The lower section, titled 'Pacientes registrados', features a table with the following data:

| # | Nombre | Apellido | Fecha nacimiento | Sexo   | País        | Dirección    | Opciones  |
|---|--------|----------|------------------|--------|-------------|--------------|---|
| 1 | Marcos | Alas     | 1995-02-02       | Hombre | El Salvador | San Salvador |   |

### III. EJERCICIOS COMPLEMENTARIOS

Utilice el sitio web desarrollado en el último ejemplo y realice las siguientes actividades:

- 1- Habilite el funcionamiento de los botones “Editar” y “Eliminar” de la tabla mostrada de los pacientes.
- 2- Investigue sobre el uso de expresiones regulares en JavaScript, adicionalmente cree un ejemplo que permita validar la información ingresada en un formulario correspondiente a la ficha de un estudiante, la siguiente información se debe de validar:
  - Carnet (Formato dos letras y tres números. Ejemplo: AB001)
  - El nombre completo de una persona (recuerde que los nombres de personas no contienen números o caracteres especiales)
  - Numero de DUI (formato: #####-#)
  - Numero de NIT (formato: ####-#####-###-#)
  - Fecha de nacimiento en el formato día, mes y año.
  - Correo electrónico
  - Edad (solo números)