	UNIVERSIDAD DON BOSCO **FACULTAD DE INGENIERÍA** **ESCUELA DE COMPUTACIÓN**
Ciclo 02 2023	**Desarrollo de Aplicaciones Web con Software Interpretados en el Cliente** **Guía de Laboratorio No. 3** **Funciones en JavaScript**

I. RESULTADOS DE APRENDIZAJE

- Adquiera dominio en la construcción de funciones con JavaScript
- Haga uso de parámetros o argumentos en las funciones que realiza.
- Utilicen los valores devueltos por las funciones desde cualquier punto de los scripts
- Creen funciones utilizando declaración de funciones como objetos y literales de funciones
- Comience a conocer el manejo de eventos usando funciones como controladores de eventos asociados a elementos del documento web.

II. INTRODUCCIÓN

¿Qué son las funciones?

Las funciones son uno de los elementos más importantes de cualquier lenguaje de programación y, más aún, en JavaScript, que de hecho es considerado un lenguaje funcional.

Las funciones, como en todo lenguaje de programación, son bloques de código independiente y reutilizable que puede ser invocado desde cualquier punto del script que se esté ejecutando. Las instrucciones de este bloque de código se definen una sola vez, pero pueden ser invocadas una o varias veces desde cualquier punto del *script* o programa.

Formas de definir funciones en JavaScript

En JavaScript existen varias formas de definir funciones, muchas de ellas casi exclusivas y que surgen como consecuencia de ser un lenguaje funcional. Mencionemos cada una de estas formas:

1. Forma tradicional con la instrucción function.
2. Como literales de función, también conocida como funciones anónimas.
3. Funciones como métodos de objetos.
4. Funciones constructoras.
5. Usando el constructor Function().
6. Haciendo uso de funciones inmediatas.

Funciones flecha (Arrow Functions)

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**.
- 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** dentro de su cuerpo.

```
// 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;
```

III. MATERIALES Y EQUIPO

Cantidad	Descripción
1	Guía de Laboratorio #3
1	Computadora con editor HTML instalado y navegadores
1	Memoria USB
1	Computadora con acceso a internet

IV. PROCEDIMIENTO

Ejemplo 1: Formulario de captura de datos que indica al usuario con un elemento p oculto una breve ayuda para llenar el campo apropiadamente. El elemento p se hace visible al activarse alguno de los campos del formulario con un evento 'focus'. Al perder dicho campo el foco se oculta la pequeña capa de ayuda.

Guion 1: datos.html

```
<!DOCTYPE html>
<html lang="es">

<head>
    <title>Captura de datos</title>
    <meta charset="utf-8" />
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css"
rel="stylesheet">
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min
.js"></script>
    <script src="js/focusblur.js"></script>
</head>

<body>
    <div class="container-fluid">
        <div class="row bg-light border-bottom">
            <div class="col display-1 text-center">
                <h1>Datos personales</h1>
            </div>
        </div>
        <div class="row">
            <div class="col-3"></div>
            <div class="col-4 rounded">
                <p id="textoAyuda" class="text-light bg bg-info "></p>
            </div>
            <div class="col"></div>
            <form name="registro" id="hongkiat-form" method="post"
action="javascript:void(0);">
                <div class="row">
                    <div class="col-3"></div>
                    <div class="col-4">
                        <div class="input-group mb-1">
                            <span class="input-group-text">
                                <img src="images/user.png">
                            </span>
                            <input type="text" class="form-control"
placeholder="Username" name="nombre" id="firstname"
                                placeholder="(Su nombre)" autocomplete="off"
tabindex="1" maxlength="25">
                            <span class="input-group-text">
                                <img src="images/user3.png">
                            </span>
                            <input type="text" class="form-control"
name="apellido" id="lastname"
                                placeholder="(Su apellido)" autocomplete="off"
tabindex="2" maxlength="25">
                        </div>
                        <div class="input-group mb-1">
                            <span class="input-group-text" id="basic-addon1">
                                <img src="images/email.png">
                            </span>
                            <input type="email" class="form-control"
name="email" id="email"
```

```
                                placeholder="(Su correo electrónico)"
autocomplete="off" tabindex="3" maxlength="50">
                        </div>
                        <div class="input-group mb-1">
                            <span class="input-group-text" id="basic-addon1">
                                <img src="images/phone.png">
                            </span>
                            <input type="tel" class="form-control"
name="telephone" id="phone"
                                placeholder="(Su número de teléfono)"
tabindex="4" pattern="\d{7,9}">
                        </div>
                        <div class="input-group mb-1">
                            <span class="input-group-text" id="basic-addon1">
                                <img src="images/speech.png">
                            </span>
                            <textarea id="describe" class="form-control"
placeholder="(Descríbete)"
                                tabindex="5"></textarea>
                        </div>
                    </div>
                    <div class="col-2">
                        <h3>Profesión:</h3>
                        <select id="profesion" name="profesion" tabindex="6"
required class="form-select">
                            <option value="med">Médico</option>
                            <option value="ing">Ingeniero</option>
                            <option value="lic">Licenciado</option>
                            <option value="abo">Abogado</option>
                            <option value="doc">Docente</option>
                            <option value="tec">Técnico</option>
                        </select>
                        <h3>País:</h3>
                        <select id="selpais" name="mesa" tabindex="7" required
class="form-select">
                            <option value="es">El Salvador</option>
                            <option value="gt">Guatemala</option>
                            <option value="ho">Honduras</option>
                            <option value="cr">Costa Rica</option>
                            <option value="ni">Nicaragua</option>
                            <option value="pa">Panamá</option>
                        </select>
                    </div>
                    <div class="col-3"></div>
                </div>
                <div class="row">
                    <div class="col-3"></div>
                    <div class="col-8">
                        <input type="reset" name="reset" id="resetbtn"
class="btn btn-secondary" tabindex="8"
                            value="Cancelar">
                        <input type="submit" name="submit" id="submitbtn"
class="btn btn-primary" tabindex="9"
                            value="Registrar">
                    </div>
                    <div class="col-1"></div>

            </form>
        </div>
    </div>
    </div>
    </div>
</body>

</html>
```

Guion 2: focusblur.js

```
//Elemento con la capa de ayuda a mostrar en los campos de formulario
var textoAyuda;
//Matriz con los mensajes de ayuda asociados a cada control de formulario
var arregloAyuda = [
     "Ingrese su nombre en este campo de texto",
     "Ingrese su apellido en este campo de texto",
     "Ingrese su dirección de correo en el formato usuario@dominio.com",
     "Ingrese su número de teléfono en el formato 9999-0000",
     "Ingrese una descripción breve en el campo área de texto",
     "Seleccione su profesión en este campo select",
     "Díganos cuál es su país de origen en este campo select",
     "Restablezca el formulario con este botón",
     "Envíe el formulario con este botón",
     ""
     ];
//Inicializar el elemento textoAyudaDiv y registrar los manejadores de eventos
//para los distintos controles de formulario
function inic(){
    textoAyuda = document.getElementById("textoAyuda");
    //Registrar los escuchadores de eventos
    registrarEscuchas(document.getElementById("firstname"), 0);
    registrarEscuchas(document.getElementById("lastname"), 1);
    registrarEscuchas(document.getElementById("email"), 2);
    registrarEscuchas(document.getElementById("phone"), 3);
    registrarEscuchas(document.getElementById("describe"), 4);
    registrarEscuchas(document.getElementById("profesion"), 5);
    registrarEscuchas(document.getElementById("selpais"), 6);
    registrarEscuchas(document.getElementById("resetbtn"), 7);
    registrarEscuchas(document.getElementById("submitbtn"), 8);
}

//Función que determina qué mensaje de ayuda habilitar
//con base en el numeroMensaje enviado como segundo argumento
function registrarEscuchas(objeto, numeroMensaje){
    //Asociar el manejador de eventos onfocus dependiendo
    //del objeto y numeroMensaje recibidos como argumentos
    objeto.addEventListener("focus", function(){
        textoAyuda.style.visibility = "visible";
        textoAyuda.innerHTML = arregloAyuda[numeroMensaje];
    }, false);
    objeto.addEventListener("blur", function(){
        textoAyuda.style.visibility = "hidden";
        textoAyuda.innerHTML = arregloAyuda[9];
    }, false);
}

//Desencadenando la función inic al cargar el documento
window.addEventListener("load", inic, false);
```

Indicaciones:

Crear una carpeta con el nombre js y dentro de esta colocar los archivos confusblur.js, creado anteriormente.

Ejemplo 2: El siguiente ejemplo muestra un ejemplo de utilización de la recursividad para generar una lista de números aleatorios que se generan al hacer clic al botón presente en la página. Los números generados pueden repetirse. Además, la solución del problema permite indicar un valor inicial y final a partir del cual serán generados los números aleatorios y hasta qué valor máximo puede generarse. Para ello se utiliza una fórmula presente en la función aleatorio() que es recursiva. Para agregar los números aleatorios generados a la matriz o arreglo que los almacena se utiliza el método push() para matrices o arreglos de JavaScript.

Guion 1: aleatorios.html

```
<!DOCTYPE html>
<html lang="es">

<head>
    <title>Números aleatorios</title>
    <meta charset="utf-8" />
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css
/bootstrap.min.css" rel="stylesheet">
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/b
ootstrap.bundle.min.js"></script>
    <script src="js/aleatorio.js"></script>
</head>

<body>
    <div class="container-fluid">
        <div class="row bg bg-primary text-light p-3">
            <div class="col display-1 text-center">
                <h1>Generación de números aleatorios</h1>
            </div>
        </div>
        <div class="row mt-1">
            <div class="col-4 ms-5">
                <form name="registro" id="hongkiat-form"
method="post" action="javascript:void(0);">
                    <div class="input-group mb-3">
                        <span class="input-group-
text">Mínimo</span>
                        <input type="number" class="form-
control" name="min" id="min">
                    </div>
                    <div class="input-group mb-3">
                        <span class="input-group-
text">Máximo</span>
                        <input type="number" class="form-
control" name="max" id="max">
                    </div>
                    <div class="input-group mb-3">
                        <span class="input-group-
text">Cantidad</span>
                        <input type="number" class="form-
control" name="cantidad" id="cantidad">
                    </div>

                    <div class="col ms-5">
                        <a href="#" id="generador"
class="btn btn-primary">
                            Generar números
                        </a>
                    </div>
```

```
            </div>
        </div>
        <div class="row">
            <div class="col ms-5">
                <div id="listanumeros"></div>
            </div>
        </div>
    </div>


</body>

</html>
```

Guion 2: aleatorio.js

```
var aleatorios;
var min, max, cantidad;
var disparador, listado;

function init() {
    //Declaración del arreglo o matriz de forma literal
    //que contendrá los números aleatorios
    aleatorios = [];
    disparador = document.getElementById("generador");
    listado = document.getElementById("listanumeros");
    if (disparador.addEventListener) {
        disparador.addEventListener('click', function (evt)
{
            capturaDatos();
        }, false);
    }
    else if (disparador.attachEvent) {
        disparador.attachEvent('onclick', function (evt) {
            capturaDatos();
        });
    }
}

function capturaDatos() {
    //Declaración y asignación de los límites entre
    //los cuales estarán comprendidos los números
aleatorios
    min = parseInt(document.getElementById("min").value);
    max = parseInt(document.getElementById("max").value);
    cantidad =
parseInt(document.getElementById("cantidad").value);

    aleatorio(min, max, cantidad);
    listado.innerHTML = "Los números aleatorios son: " +
aleatorios.toString();
}
```

```
function aleatorio(minimo, maximo, cantidad) {
    //Generando un número aleatorio teniendo el cuidado
    //que esté comprendido entre los valores mínimo y
máximo
    var numero = Math.floor(Math.random() * (maximo -
minimo + 1)) + minimo;
    //Verificar que no se haya llegado a la cantidad
    //de números aleatorios máximo establecidos
    if (aleatorios.length < cantidad) {
        //Agregar el número aleatorio al arreglo o matriz
        //utilizando el método push()
        aleatorios.push(numero);
        //Invocar recursivamente a la función aleatorio
        aleatorio(minimo, maximo, cantidad);
    }
}

window.onload = init;
```

Indicaciones:

Crear una carpeta con el nombre js y dentro de esta colocar los archivos aleatorio.js, creado anteriormente.

Ejemplo 3: Ejemplo que muestra cómo generar imágenes aleatorias simulando un juego de tirar 6 dados. Se utilizar JavaScript no invasivo para hacer que dando clic en un botón se simule cada lanzamiento que muestra en la posición de cada imagen el dado seleccionado aleatoriamente mediante la función random(), limitada para obtener únicamente valores entre 1 y 6.

Guion 1: tirardados.html

```
<!DOCTYPE html>
<html lang="es">

<head>
    <title>Tiro de dados</title>
    <meta charset="utf-8" />
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css
/bootstrap.min.css" rel="stylesheet">
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/b
ootstrap.bundle.min.js"></script>
    <script src="js/tirodados.js"></script>
</head>

<body class="bg-warning bg-gradient">
    <div class="container-fluid">
        <div class="row mt-4">
            <div class="col-3"></div>
```

```
            <div class="col">
                <a href="#" id="botonTirar" class="btn btn-
primary">
                    Tirar los dados
                </a>
            </div>
        </div>
        <div class="row">
            <div class="col-3"></div>
            <div class="col">
                <ul class="list-group list-group-
horizontal">
                     <li class="list-group-item list-group-
item-primary border-0">
                        <img src="images/dice1.png"
id="dado1" alt="Imagen del dado" />
                    </li>
                     <li class="list-group-item list-group-
item-primary border-0">
                        <img src="images/dice2.png"
id="dado2" alt="Imagen del dado" />
                    </li>
                     <li class="list-group-item list-group-
item-primary border-0">
                        <img src="images/dice3.png"
id="dado3" alt="Imagen del dado" />
                    </li>
                     <li class="list-group-item list-group-
item-primary border-0">
                        <img src="images/dice4.png"
id="dado4" alt="Imagen del dado" />
                    </li>
                     <li class="list-group-item list-group-
item-primary border-0">
                        <img src="images/dice5.png"
id="dado5" alt="Imagen del dado" />
                    </li>
                     <li class="list-group-item list-group-
item-primary border-0">
                        <img src="images/dice6.png"
id="dado6" alt="Imagen del dado" />
                    </li>
                </ol>
            </div>
        </div>

    </div>
</body>

</html>
```

Guion 2: tirodados.js

```
//Todo el módulo se encuentra dentro de una función
//conocida como función inmediata, en este caso
//también anónima que provoca que todo se ejecute
//de una vez.
(function(){
    if(window.addEventListener){
        window.addEventListener("load", iniciar, false);
    }
    else{
        window.attachEvent("onload", iniciar);
    }
    //Variables utilizadas para interactuar
    //con los elementos img presentes en la página
    var imagenDado1;
    var imagenDado2;
    var imagenDado3;
    var imagenDado4;
    var imagenDado5;
    var imagenDado6;

    //Registrar componentes de escucha del evento clic
    //al presionar el botón denominado botonTirar y obtener
    //todos los elementos img presentes en el documento
    function iniciar(){
        var boton = document.getElementById("botonTirar");
        if(boton.addEventListener){
            boton.addEventListener("click", tirarDados,
false);
        }
        else{
            boton.attachEvent("onclick", tirarDados);
        }
        imagenDado1 = document.getElementById("dado1");
        imagenDado2 = document.getElementById("dado2");
        imagenDado3 = document.getElementById("dado3");
        imagenDado4 = document.getElementById("dado4");
        imagenDado5 = document.getElementById("dado5");
        imagenDado6 = document.getElementById("dado6");
    }

    function tirarDados(){
        establecerImagen(imagenDado1);
        establecerImagen(imagenDado2);
        establecerImagen(imagenDado3);
        establecerImagen(imagenDado4);
        establecerImagen(imagenDado5);
        establecerImagen(imagenDado6);
    }

    function establecerImagen(imgDado){
```

```
        var valorDado = Math.floor(1+Math.random()*6);
        //Estableciendo el atributo src de la imagen
        imgDado.setAttribute("src", "images/dice" +
valorDado + ".png");
        //Estableciendo el atributo alt de la imagen
        imgDado.setAttribute("alt", "Imagen del dato con el
valor " + valorDado);
    }
})();
```

Indicaciones:

Crear una carpeta con el nombre js y dentro de esta colocar los archivos tirodados.js, creado anteriormente.

Ejemplo 4: Ejemplo en que se muestra un conversor de números base decimal a su respectivo equivalente en binario, octal y hexadecimal. Para ello se hará uso de funciones flechas.

Guion 1: conversor.html

```
<!DOCTYPE html>
<html lang="es">
<html>
<head>
  <title>Conversor Decimal</title>
  <meta charset="utf-8" />
  <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css
/bootstrap.min.css" rel="stylesheet">
  <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/b
ootstrap.bundle.min.js"></script>
  <script src="js/conversor.js"></script>
</head>
<body>
  <div class="container">
    <h1 class="mt-5">Conversor Decimal</h1>
    <div class="form-group mt-4">
      <label for="decimalInput">Número Decimal:</label>
      <input type="number" class="form-control"
id="decimalInput" placeholder="Ingrese el número decimal">
    </div>
    <button class="btn btn-primary"
onclick="convertir()">Convertir</button>
    <div class="mt-4">
      <h4>Resultados:</h4>
      <p id="binarioResultado"></p>
      <p id="octalResultado"></p>
      <p id="hexadecimalResultado"></p>
    </div>
  </div>
</body>
```

```
</html>
```

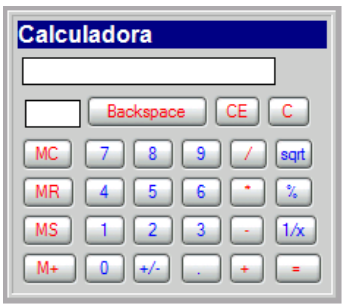
Guion 2: conversor.js

```
// Función para convertir un número decimal a binario
const convertirDecimalABinario = (decimal) =>
decimal.toString(2);

// Función para convertir un número decimal a octal
const convertirDecimalAOctal = (decimal) =>
decimal.toString(8);

// Función para convertir un número decimal a hexadecimal
const convertirDecimalAHexadecimal = (decimal) =>
decimal.toString(16).toUpperCase();

// Función para obtener el valor ingresado por el usuario y
realizar las conversiones
const convertir = () => {
  const decimal =
parseInt(document.getElementById("decimalInput").value);
  document.getElementById("binarioResultado").textContent =
`Binario: ${convertirDecimalABinario(decimal)}`;
  document.getElementById("octalResultado").textContent =
`Octal: ${convertirDecimalAOctal(decimal)}`;

document.getElementById("hexadecimalResultado").textContent
= `Hexadecimal: ${convertirDecimalAHexadecimal(decimal)}`;
};
```

V. ANÁLISIS DE RESULTADOS

PROBLEMAS A RESOLVER:

1. Desarrollar una aplicación con diversos enlaces a sitios externos, deberá solicitar una confirmación de visita al sitio que permita cancelar la visita de un sitio web al hacer clic en su enlace o confirma que si quiere abrir en una pestaña nueva el sitio.
2. Cree una calculadora relativamente básica que permita realizar cálculos utilizando funciones. Las operaciones que debe permitir la calculadora son: suma, resta, multiplicación, división, residuo, inversa de un número, cuadrado de un número y raíz cuadrada. Utilice la llamada la función con parámetros y una interfaz como la mostrada a continuación:



NOTA: Utilice para todos los problemas formularios y JavaScript no invasivo (unobstrusive JavaScript). No vale usar prompt para capturar los datos del usuario, ni la utilización de manejadores de eventos en etiquetas HTML.

VI. INVESTIGACION COMPLEMENTARIA

1. Investigue qué son las clausuras en JavaScript, un concepto clave de la programación funcional. Indique qué son, para qué pueden servir, cuándo utilizarlas y muestre tres ejemplos sencillos que no sean copiados y pegados de código existente en páginas web de Internet y un ejemplo práctico que ejemplifique su utilización solucionando un problema. Los ejemplos no deben producir errores de consola.

VII. BIBLIOGRAFÍA

- Flanagan, David. JavaScript La Guía Definitiva. 2da Edición. Editorial ANAYA Multimedia. 2007. Madrid, España.
- Deitel, Paul / Deitel, Harvey / Deitel, Abbey. Internet & World Wide Web. Cómo programar. 5a. Edición. Editorial Pearson. 2014. México D.F..
- John Resig / Bear Bibeault. JavaScript Ninja.1a. Edición. Editorial Anaya Multimedia / Manning. Septiembre 2013. Madrid, España.
- Powell, Thomas / Schneider, Fritz. JavaScript Manual de Referencia. 1ra Edición. Editorial McGraw-Hill. 2002. Madrid, España.