	UNIVERSIDAD DON BOSCO **FACULTAD DE ESTUDIOS TECNOLÓGICOS** **COORDINACIÓN DE COMPUTACIÓN Y MÓVILES**
Ciclo II	**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.

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 prueba que muestra cómo se maneja el control de eventos con el modelo del DOM Nivel 2, sin usar atributos HTML como manejadores de eventos, si no haciéndolo directamente en el script.

eventos.js

```
function iniciar(){
    //Definir objetos sobre los que se escucharán eventos
    //y asociar los eventos correspondientes
    //Primera caja de texto
    var textFieldFocus = document.frmEjemplo.txtfoco;
    textFieldFocus.onfocus = foco;
    //Segunda caja de texto
    var textFieldKey = document.frmEjemplo.txttecla;
    textFieldKey.onkeypress = tecla;
    //Campo select
    var selectField = document.frmEjemplo.selopciones;
    selectField.onchange = function(){
        cambioOpcion(this.options[this.selectedIndex].value);
    };
    //Primer campo checkbox
    var chkbox1 = document.getElementById("chkbox1");
    chkbox1.addEventListener("click", function(){
        activarCasilla(chkbox1, chkbox1.value);
    }, false);
    //Segundo campo checkbox
    var chkbox2 = document.getElementById("chkbox2");
    chkbox2.addEventListener("click", function(){
        activarCasilla(chkbox2, chkbox2.value);
    }, false);
    //Tercer campo checkbox
    var chkbox3 = document.getElementById("chkbox3");
    chkbox3.addEventListener("click", function(){
        activarCasilla(chkbox3, chkbox3.value);
    }, false);

    //Botón de envío del formulario
    var submitButton = document.getElementById("submitbutton");
    submitButton.addEventListener("click", function(){
        alert(pulsa());
        return false;
    }, false);
}

function foco(){
    alert("Foco en la primera Caja");
    this.focus();
}
```

```
function tecla(){
    alert("Pulsaste una tecla");
    this.focus();
}

function cambioOpcion(nuevaOpcion){
    alert("Has cambiado a la opción " + nuevaOpcion);
}

function activarCasilla(casilla, valorCasilla){
    var accion;
    //alert(casilla.checked);
    (casilla.checked) ? accion = " activado" : accion = " desactivado";
    alert("Se ha" + accion + " la casilla " + casilla.id + " con la opción " +
valorCasilla);
}

function pulsa(){
    return "Materia: Desarrollo de Aplicaciones Web con Software Interpretado
en el Cliente";
}

//Iniciar los manejadores de eventos al cargar la página
window.onload = iniciar;
```

Indicaciones:

- Crear una carpeta con el nombre js y dentro de esta colocar los archivos eventos.js, creado anteriormente.
- Crear una carpeta con el nombre css y dentro de esta colocar los archivos eventos.css, brindado en los recursos de la guía.

Ejemplo 2: Confirmación de visita a enlaces que permite 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.

confirmlinks.js

```
//Iniciando el manejador para los enlaces de la página.
window.onload = confirmAllLinks;

//Controlador de eventos para elementos de enlace y area en mapas.
//Utiliza la palabra clave this para hacer referencia al elemento
//del documento y puede devolver false para evitar que
//el navegador vaya al vínculo.
function confirmLink(){
       return confirm("¿Está seguro que quiere visitar el sitio: " + this.href
+ "?");
}
//Recorrer todos los enlaces (hipervínculos) del documento y asignar
//la función confirmLink a cada uno como controlador controlador de evento.
function confirmAllLinks(){
    for(var i=0; i<document.links.length; i++){
        document.links[i].onclick = confirmLink;
    }
}
```

Indicaciones:

- Crear una carpeta con el nombre js y dentro de esta colocar los archivos confirmlinks.js, creado anteriormente.
- Crear una carpeta con el nombre css y dentro de esta colocar los archivos fonts.css y links.css, brindado en los recursos de la guía.

Ejemplo 3: 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.

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.
- Crear una carpeta con el nombre css y dentro de esta colocar los archivos fonts.css y basic.css, brindado en los recursos de la guía.

Ejemplo 4: Ejemplo que muestra cómo validar un campo de texto en un formulario para que solamente acepte números utilizando el evento keypress, definido como atributo del elemento INPUT usando el modelo de eventos Nivel 0 del DOM. Este modelo es el único admitido al 100% en todos los navegadores, pero a la vez, es el más limitado.

filternumbers.js

```
// Mostrar la tecla presionada. En una aplicación real no debería hacerlo.
// Probablemente sería buena idea mostrar únicamente en el caso que
// se ingresen teclas que no sean números.
function showChar(e){
    var code;
    if (!e) var e = window.event;
    if (e.keyCode) code = e.keyCode; //Todos los navegadores menos Netscape
4.0 o inferior
    else if (e.which) code = e.which; //Netscape 4.0 o inferior
    var character = String.fromCharCode(code);
    //alert('La tecla presionada fue: ' + character);
}

// Dejar pasar únicamente números enteros y bloquear cualquier otra caracter
function numbersOnly(field, event){
    /*alert(field.value);*/
    var key, keychar;
    showChar(event);   //Mostrar la tecla presionada
    if(window.event)   //Todos los navegadores menos Netscape 4.0 o inferior
        key = window.event.keyCode;
    else if(event)     //Netscape 4.0 o inferior
        key = event.which;
    else
        return true;
    keychar = String.fromCharCode(key);
    // Comprobar caracteres especiales como espacio en blanco
    // luego comprobar números
    if((key == null) || (key == 0) || (key == 8) || (key == 9) || (key == 13)
|| (key ==
27))
        return true;
    else if((("0123456789").indexOf(keychar) > -1)){
        window.status = "";
        return true;
    }
    else{
        window.status = "Este campo solo acepta números";
        return false;
    }
}
```

Indicaciones:

- Crear una carpeta con el nombre js y dentro de esta colocar los archivos filternumbers.js, creado anteriormente.
- Crear una carpeta con el nombre css y dentro de esta colocar los archivos numbers.css y pure-min.css, brindado en los recursos de la guía.

Ejemplo 5: 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.

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 = [];
    //Declaración y asignación de los límites entre
    //los cuales estarán comprendidos los números
aleatorios
    min = parseInt(prompt("Ingrese el límite inferior",
""));
    max = parseInt(prompt("Ingrese el límite superior",
""));
    cantidad = parseInt(prompt("Indique la cantidad de
números aleatorios a generar", ""));
    disparador = document.getElementById("generador");
    listado = document.getElementById("listanumeros");
     if(disparador.addEventListener){
         disparador.addEventListener('click',
function(evt){
             aleatorio(min, max, cantidad);
             listado.innerHTML = "Los números aleatorios
son: " + aleatorios.toString();
         }, false);
     }
     else if(disparador.attachEvent){
         disparador.attachEvent('onclick', function(evt){
             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.
- Crear una carpeta con el nombre css y dentro de esta colocar los archivos random.css, brindado en los recursos de la guía.

Ejemplo 6: 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.

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", "img/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.
- Crear una carpeta con el nombre css y dentro de esta colocar los archivos button.css y dados.css, brindado en los recursos de la guía.

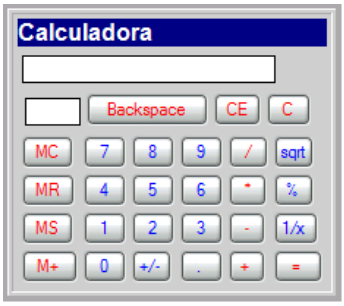
V. EJERCICIOS COMPLEMENTARIOS

PROBLEMAS A RESOLVER:

Nota: Cree una carpeta para guardar cada uno de los scripts de esta parte y nómbrela analisisguia3dawsic. Deberá entregar el análisis de resultados y la investigación

complementaria en la siguiente práctica mediante correo electrónico, el análisis o discusión de resultados en un correo y la investigación complementaria en otro.

1. 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.