	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. 4** **Objetos en JavaScript**

I. OBJETIVOS

Que el estudiante:

- Tenga claras todas las características de la Programación Orientada a Objetos con JavaScript.
- Utilice los diversos tipos de objetos incorporados con que cuenta el lenguaje JavaScript.
- Se familiarice con los objetos del navegador más utilizados en los scripts.

II. INTRODUCCION TEORICA

Programación Orientada a Objetos en JavaScript

La Programación Orientada a Objetos está basada en el concepto de dos entidades distintas, que son: las clases y las instancias.

Una clase define las propiedades y métodos, a veces denominados simplemente miembros, que caracterizan a un cierto grupo de objetos. Una clase es una abstracción de ciertos objetos que existen en la realidad e intenta ser un modelo o plano de los objetos que representa, definiendo únicamente las características y el comportamiento que debe tener dicho objeto.

La instancia es un objeto concreto creado a partir de las propiedades y métodos definidos en la clase, a veces se prefiere decir que la instancia es un ejemplar concreto del tipo de objeto definido en la clase a partir de la cual se ha creado el objeto. Por ejemplo, Victoria Escobar es una instancia o un ejemplar concreto de la clase empleado, representando a una persona en particular que es o tiene las características y comportamiento de un empleado.

Los lenguajes basados en clases, como Java, C++ o C#, exigen que se deba definir primero una clase y a continuación realizar instancias de esta clase para crear un objeto en particular. JavaScript, es más bien un lenguaje basado en prototipos que no requiere hacer esta distinción, simplemente es un lenguaje que tiene objetos predefinidos, a los que se conoce como prototipos y que son, en la práctica, plantillas a partir de las cuales se obtienen propiedades y métodos para los nuevos objetos que se crean. Sin embargo, cada nuevo objeto puede especificar sus propias propiedades y métodos, tanto al ser creado o en tiempo de ejecución.

Diferencias entre lenguajes basados en clase en comparación con los basados en prototipos

La siguiente tabla resume las diferencias entre los lenguajes basados en clases, como Java o C#, y los basados en prototipos como JavaScript.

Basado en Clases (Java)	Basado en Prototipos (JavaScript)
Clase e instancia son entidades distintas.	Todos los objetos son instancias.
Define una clase con la definción de clase; instancia una clase con los métodos constructores.	Define y crea un conjunto de objetos con las funciones constructoras.
Crea un único objeto con el operador new.	Igualmente.
Construye una jerarquía de objetos utilizando las definiciones de clases par definir subclases de una clase existente.	Construye una jerarquía de objetos mediante la asignación de un objeto como prototipo asociado con la función constructora.

Hereda las propiedades mediante el seguimiento de la cadena de clases.	Hereda las propiedades mediante el seguimento de la cadena de prototipos.
La defínción de Clases especifica todas las propiedades de todas las intancias de una clase. No puede añadir propiedades dinámicamente en tiempo de ejecución.	La función constructora o prototipo puede especificar un conjunto inicial de propiedades. Puede añadir o remover dinámicamente a objetos individuales o a un conjunto entero de objetos.

Definición de objeto

Los objetos en JavaScript son tipos de datos compuestos, tal y como las matrices o arreglos, que dentro del lenguaje JavaScript son, más bien, objetos. Se dice que son compuestos porque combinan estado (propiedades que almacenan datos) y comportamiento (procedimientos o métodos para operar los datos).
Un objeto viene siendo una representación concreta, particular y real de algo. Esta representación determina su identidad, su estado y su comportamiento.

Clasificación de objetos en JavaScript

Los objetos en JavaScript se pueden subdividir en cuatro grupos:

- Objetos definidos por el usuario. Son objetos completamente personalizables que puede crear el desarrollador para dar estructura y coherencia a una tarea de programación en particular. Con estos objetos es posible definir propiedades y métodos mediante una sintaxis específica de JavaScript. Las propiedades identifican las características propias del objeto, mientras que los métodos permiten realizar tareas u operaciones con dichas propiedades.
- Objetos incorporados del lenguaje. Son objetos proporcionados por el propio lenguaje JavaScript, entre los que se incluyen: (1) objetos asociados a tipos de datos primitivos, como los objetos **String**, **Number** y **Boolean**, (2) objetos que permiten la creación de objetos definidos por el usuario, como los objetos **Object** y **Form**, (3) objetos utilizados para simplificar tareas frecuentes, como los objetos **Date**, **Math** y **RegExp**.
- **Objetos de navegador**. Son objetos que no están especificados como parte del lenguaje JavaScript, pero que son soportados por la mayor parte de navegadores que cumplen con los estándares. Los dos objetos más utilizados que pertenecen a esta categoría son los objetos **window** y **Navigator**. El primero permite el control de la ventana del navegador y, el segundo, proporciona información acerca de la configuración del navegador del usuario.
- **Objetos de documento**. Son los objetos que forman parte del Modelo de Objetos de Documento (Document Object Model, abreviado como DOM) definido por la W3C. Es a través de estos objetos que JavaScript puede manejar Hojas de Estilo en Cascada y la ejecución de HTML dinámico. El acceso a los objetos de documento es proporcionado por el navegador mediante el uso del objeto **document** del objeto **window**.

Creación de objetos en JavaScript

Existen dos formas de crear objetos en JavaScript:

1. Utilizando la palabra clave **new** y el nombre del **constructor del objeto** a crear. Se pueden crear objetos especializados haciendo uso del operador **new** y a continuación invocar al **método constructor** del objeto que inicializa las propiedades del objeto.

Ejemplo:

```
var carro = new Object();
carro.marca = "Toyota";
carro.modelo = "Corolla";
carro.npasajeros = 5;
carro.annio = 2007;
carro.placa = "503-618";
carro.nchasis = "JDSXV39Z200750226";
carro.encender = function(numero){
                  carro.nchasis == numero ? start = "on" : start = "off";
                  return start;
                  }
carro.apagar = function(){
```

```
    start = off;
  }
```

2. Utilizando sintaxis de literales de objeto. Esta es la forma más fácil de crear objetos en JavaScript, que consiste de una lista de pares `nombre_propiedad : valor` separados por comas y encerrados entre llaves.

Ejemplo:

```
var coord = {x : 0, y : 0};
var circle = {x : coord.x, y : coord.y, r : 2.5};
```

Esta forma de construir objetos en JavaScript es generalizable a todos los tipos de datos del lenguaje. Veamos esto demostrado con dos tipos de objetos como **String** y **Array**.

Utilizando la palabra clave new y el respectivo método constructor:

```
var nombre = new String();
var paises = new Array("El Salvador","Guatemala","Honduras");
```

Utilizando sintaxis declarativa conocida como sintaxis de literales de tipos de datos:

```
var nombre = "";
var paises = ["El Salvador","Guatemala","Honduras"];
```

Objetos del lenguaje

Estos son el conjunto de objetos que vienen incorporados por el lenguaje. Entre los principales tenemos los objetos Array, Function, Date, String, Math, Boolean, Number.

1. Objeto Array

Este objeto se abordó en guías pasadas, por lo que trataremos únicamente su declaración, propiedades y métodos. La forma de declarar un objeto Array es haciendo uso del constructor Array(). Si se pasan argumentos al constructor, se interpreta que estos serán los elementos del arreglo. La excepción se da cuando se le pasa al constructor un solo valor numérico, en cuyo caso se entiende que se le proporciona al arreglo un tamaño, que podremos acceder mediante la propiedad length.

Los siguientes son ejemplos de declaración de objetos Array:

```
//Crea un arreglo vacío (sin elementos definidos) y crea una referencia
//a él en la variable arreglo1
var arreglo1 = new Array();
//Crea un arreglo con tres elementos
var arreglo2 = new Array("rojo", "verde", "azul");
//Crea un arreglo cuya propiedad length tiene el valor de 5
var arreglo3 = new Array(5);
```

Propiedades del objeto Array

length

Como su nombre indica esta propiedad nos devuelve la longitud del array, es decir, el número de elementos que puede almacenar. Su uso es muy simple:

```
var lista = new Array(50);
tamagno = lista.length; /*tamagno almacenaría el valor 50 */
```

prototype

Esta es una propiedad muy potente en el sentido que nos permite agregar al objeto Array las propiedades y métodos que queramos.

```
Array.protoype.descriptor = null;
dias = new Array ('lunes', 'Martes', 'Miercoles', 'Jueves', 'Viernes');
dias.descriptor = "Dias laborables de la semana";
```

En este ejemplo hemos creado una nueva propiedad para el objeto **Array**, la propiedad descriptor que podría utilizarse para darle un título a la matriz.

Métodos del objeto Array
Para ver estos métodos consulte la guía sobre arreglos realizada recientemente. Entre dichos métodos se encuentran: concat(), join(), reverse(), sort(), pop(), push(), etc.

2. Objeto Function
Las funciones también son consideradas como objetos por JavaScript. Aunque normalmente no estamos acostumbrados a definirlas de esta forma, JavaScript proporciona una forma de definir funciones en donde explícitamente se declaran como objetos. Para ello se utiliza el operador new y el constructor Function, tal y como se muestra a continuación:

```
sumar = new Function("a", "b", "return  a + b");
```

La llamada a la function se realize de la misma forma que conocemos. Puede ver el siguiente ejemplo de llamada:

```
document.write(sumar(90, 100));
```

Métodos de Function
Los métodos de Function son todos los heredados del objeto Object, que veremos más adelante.

Propiedades del objeto Function
Básicamente se tienen tres propiedades que se pueden usar con el objeto Function y son:
arguments
Se trata de un array que contiene los argumentos pasados a la función. Esta propiedad permite el uso de funciones con un número variable de argumentos.

caller
Contiene una referencia a la función que llamó a la actual.
constructor
Heredada de la clase Object.

3. Objeto String
El objeto **String** se usa para manipular cadenas de caracteres. En JavaScript todo texto encerrado entre comillas, dobles o simples, se interpreta como una cadena, así '45' no es el número cuarenta y cinco sino la cadena formada por los caracteres 4 y 5. Este objeto contiene una gran cantidad de métodos para el manejo y el análisis de cadenas, la extracción de subcadenas e incluso para la conversión de cadenas a texto de marcado HTML.
Como en los anteriores objetos se utiliza el operador new y el constructor String(), como se muestra a continuación:

```
var cadena = new String();
```

También se puede pasar un argumentos al constructor, que será su valor inicial, como se muestra en este otro ejemplo:

```
var cadena = new String("No me importa.");
```

Propiedades del objeto String
length
Valor numérico que nos indica la longitud en caracteres de la cadena dada.

prototype
Nos permite asignar nuevas propiedades al objeto String.

Métodos del objeto String

anchor(name)
Crea un enlace (o ancla) a partir de un objeto String con el atributo name igual a la cadena que se le pasa como argumento al método.
Vea el siguiente ejemplo:

```
var refer "referencia1";
var ancla1 = refer.anchor("anclaje1");
```

Después de ejecutadas las instrucciones anteriores el valor de ancla será:

```
<a name=”anclaje1”>referencia1</a>
```

La sintaxis de este método permite usar una constante String en lugar del nombre de un objeto String. El ejemplo anterior podría haber escrito como:

```
var ancla = "referencia1".anchor("anclaje1");
```

charAt(position)
Este método aplicado a una cadena devuelve el carácter que se encuentra en la posición dada por el atributo *position* teniendo en cuenta que el índice del primer carácter a la izquierda de la cadena es 0 y el último es una unidad menor que longitud de la cadena. Si el valor del atributo no es válido (igual o mayor que la longitud de la cadena o negativo) el método devuelve el valor *undefined*. Por ejemplo, el siguiente código devuelve el carácter en la tercera posición de la cadena nombre:

```
var nombre = "abcdefghij";
var car3 =
nombre.charAt(2);
```

Devolverá "c", que es el tercer carácter por la izquierda (índice igual a 2).

4. Manejo de fechas
El objeto Date contiene un valor que representa fecha y hora de un instante dado. Para crear una instancia de este objeto usamos alguna de las siguientes sintaxis:

```
var fecha = new Date();
var fecha = new date(número);
var fecha = new date(cadena);
var fecha = new date(año, mes, día[, hora[, minutos[,seg[,ms]]]]);
```

Los argumentos encerrados entre corchetes son opcionales. En la primera forma la variable fecha contendrá la fecha del día actual. La segunda opción almacena en fecha la fecha dada por el argumento como el número de milisegundos transcurridos desde la media noche del 1 de Enero de 1970. El tercer tipo se usa cuando la fecha se pasa en forma de cadena. Por último la fecha puede crearse pasándole como argumento los números de año, mes, día, hora y opcionalmente, hora, minuto, segundo y milisegundo. Los años posteriores a 1970 puede escribirse con dos dígitos, pero es aconsejable usar siempre cuatro dígitos por aquello de los efectos 2000.

```
var hoy = new date()   /*fecha del día  en hoy */
var evento = new Date("November 10 1990");
var otro = new Date("10 Nov 1990");
var otro = new Date("10/02/2000"); //Oct, 2, 2000
var instante = new Date(1990, 11, 10, 20,00);
```

Estas son las posibles formas de declarar objetos de tipo fecha. Las dos últimas almacenan el mismo día, pero en la última además se guarda la hora.
Donde se usen cadenas para indicar una fecha podemos añadir al final las siglas GMT (o UTC) para indicar que la hora se refiere a hora del meridiano Greenwich, si no se toma como hora local, o sea, según la zona horaria configurada en el ordenador donde se ejecute el script.

Métodos

GetDate()

Nos devuelve el día del mes del objeto fecha al que se aplica. Este método controla por supuesto el número de días de cada mes y contempla el caso de años bisiestos, incluida la excepción del 2000. En el siguiente ejemplo se presenta en pantalla Hoy es día 2, suponiendo que la fecha del sistema es 2-10-200. Primero creamos la variable fecha instanciada como un objeto Date() para a continuación escribir directamente el valor de getDate() aplicado a fecha

```
var fecha = new Date();
    document.write("Hoy es día: "+fecha.getDate());
```

setFullYear()

Nos permite cambiar el año del objeto fecha por el valor pasado como argumento, un número de dos dígitos que se interpreta como decenas dentro del siglo, o sea, que para poner el año 1995 se debe pasar 95. El ejemplo pone precisamente este valor en el campo año de la variable fecha.

```
var fecha = new Date();
    fecha.setYear(95)
    document.write(fecha.toString());
```

Ojo si pasamos el valor 00 en el argumento el año obtenido es el 1900 (uno de los efectos 2000), por esto es recomendable usar la función setFullYear(agno).

setDate()

Nos permite cambiar el día del mes del objeto fecha al que se aplica para poner el valor que se pasado en el argumento diames. Este método controla por supuesto el número de días de cada mes y contempla el caso de años bisiestos, incluida la excepción del 2000, de forma que si pasamos como argumento 31 y el mes es de 30 días la función corrige la fecha completa pasándola al día 1 del mes siguiente. Esto lo vemos en el ejemplo que sigue: creamos una variable como un objeto Date con el último día de Septiembre (mes de 30 días) e intentamos poner el día a 31, luego comprobamos la fecha almacenada:

```
var fecha = new Date("1 Sep 2008");
fecha.setDate(31);
document.write("Hoy es día: " + fecha.toString());
```

Como puede verse, si se prueba el ejemplo la fecha es corregida y pasa a 1 de Octubre.

Objetos del navegador.

Los objetos del navegador se encuentran en el nivel más alto de la jerarquía de objetos que define JavaScript. Estos son **window, frame, location, history, y navigator.**

El objeto más alto en la jerarquía de objetos que define JavaScript es **window**. El resto de objetos (excepto el objeto **navigator**) se ubican siempre dentro de una ventana.

Los métodos y propiedades de los objetos ***window*** y ***navigator*** pueden utilizarse sin necesidad de declarar variables.

Ejemplo:

```
//Mostrar la hora segundo a segundo
function hourStatus(){
   var miFecha = new Date();
   window.status = formato(miFecha.getHours()) + ":" + formato(miFecha.getMinutes()) + ":"
+ formato(miFecha.getSeconds());
   //Llamando a la función segundo a segundo
   setTimeout("hourStatus()", 1000);
}
//Función para formatear las partes de la hora
function formato(valor){
   if(valor<10) valor = "0" + valor;
   return valor;
}
```

Objetos de documento

Los objetos del documento se encuentran en el segundo nivel de la jerarquía de objetos que define JavaScript. Estos son **document, link, anchor e image.** Estos objetos permiten a JavaScript, entre otras cosas, el manejo de Hojas de Estilo en Cascada y la ejecución de HTML dinámico (DHTML).
El acceso a los objetos del documento es proporcionado por el navegador mediante el uso de la **propiedad *document*** del **objeto window**.

Ejemplo:

```
//Función que obtiene el texto seleccionado
function getTextSelected(){
   var selText = "";
   //Internet Explorer
   if(document.selection){
      selText = document.selection.createRange().text;
   }
   //Otros navegadores (Chrome, Firefox, Opera, etc.)
   if(window.getSelection){
      selText = window.getSelection();
   }
   if(document.getSelection){
      selText = document.getSelection();
   }
   //Asignar el texto seleccionado a un elemento P
   document.getElementById("paragraph").innerHTML = "<p>Texto seleccionado: " + selText +
"</p>";
}
```

Estas últimas dos categorías de objetos se estudiarán con más detalle en próximas guías de práctica cuando nos adentremos en el manejo del Modelo de Objetos de Documento (DOM).

III. MATERIALES Y EQUIPO

Para la realización de la guía de práctica se requerirá lo siguiente:

No.	Requerimiento	Cantidad
1	Guía de práctica #4: Objetos en JavaScript	1
2	Computadora con Sublime Text o Notepad++ y navegadores instalados	1
3	Memoria USB	1

IV. PROCEDIMIENTO

Para los ejemplos del procedimiento de esta guía se ha utilizado el enfoque de desarrollo con JavaScript no invasivo. Esto significa que el manejo de eventos no se realizará en casi ningún caso con atributos manejadores de eventos dentro del código HTML. El manejo de eventos se hace con el modelo de eventos estándar, conocido como modelo de eventos Nivel 2 del DOM, que dicho sea de paso, es mucho más eficiente y propicia la completa separación de la funcionalidad del documento web de su estructura, algo muy similar a la separación de la apariencia de la estructura del documento lograda con el correcto manejo de las hojas de estilo en cascada.

Ejemplo #1: El ejemplo solicita dos valores numéricos (pueden ser con punto flotante) para calcular, presionando los botones respectivos, el perímetro y el área de un rectángulo con esas dimensiones. Se utilizan campos de formulario de tipo number, incorporados en HTML5 y se establecen valores mínimo, máximo y el desplazamiento para manejar los valores ingresados con el control que algunos navegadores modernos incorporan. También es factible ingresar los valores directamente.

Guión 1: rectangulo.html

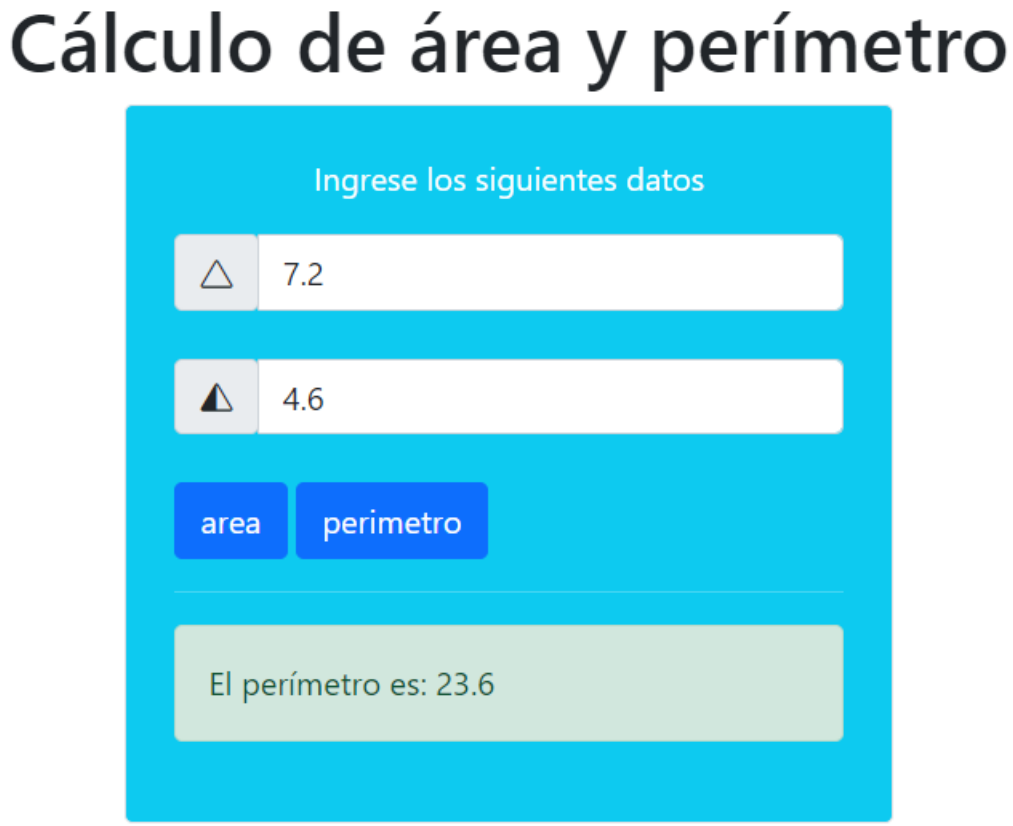
```
 <!DOCTYPE html>
<html lang="es">

<head>
  <meta charset="utf-8" />
  <title>Áreas de rectángulos</title>

  <link     href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css"
rel="stylesheet"
    integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC"
crossorigin="anonymous">
  <link           rel="stylesheet"           href="https://cdn.jsdelivr.net/npm/bootstrap-
icons@1.9.1/font/bootstrap-icons.css">
  <script src="js/calculo.js"></script>
</head>
<body>
  <section id="slick">
    <h1 class="text-center">Cálculo de área y perímetro</h1>
    <div class="container w-25 border  rounded p-4 bg-info text-white">
      <p class="text-center">Ingrese los siguientes datos</p>
      <!-- Campos de formulario -->
      <form name="frmrectangulo" id="frmrectangulo" action="javascript:void(0);">
        <div class="input-group mb-4">
          <div class="input-group-prepend">
            <div class="input-group-text"><i class="bi bi-triangle"></i></div>
          </div>
          <input type="number" name="base" class="form form-control" id="txtbase" min="0.0"
max="1000.0" step="any"
            placeholder="(Base)" required />
        </div>
        <div class="input-group mb-4">
          <div class="input-group-prepend">
            <div class="input-group-text"><i class="bi bi-triangle-half"></i></div>
          </div>
          <input  type="number"  name="altura"  class="form  form-control"  id="txtaltura"
min="0.0" max="1000.0" step="any"
            placeholder="(Altura)" required />
        </div>

        <input type="submit" value="area" name="btnarea" id="area" class="btn btn-primary
btn-block" />
        <input   type="submit"   value="perimetro"   name="btnperimetro"   id="perimetro"
class="btn btn-primary btn-block" />

        <div id="resultado"></div>
      </form>
    </div>
    </div>
    </div>
  </section>
</body>

</html>
```

Guión 2: calculo.js

```
//Registrar evento click del ratón al presionar botones de envío
function iniciar(){
    var btnarea = document.getElementById("area");
    var btnperim = document.getElementById("perimetro");
    if(btnarea.addEventListener){
```

```
        btnarea.addEventListener("click", calculararea, false);
    }
    else if(btnarea.attachEvent){
        btnarea.attachEvent("onclick", calculararea);
    }
    if(btnperim.addEventListener){
        btnperim.addEventListener("click", calcularperimetro, false);
    }
    else if(btnperim.attachEvent){
        btnperim.attachEvent("onclick", calcularperimetro);
    }
}

function calculararea(){
    var rect = new rectangulo(parseFloat(document.frmrectangulo.txtbase.value),
parseFloat(document.frmrectangulo.txtaltura.value));
    rect.mostrar(rect.carea(), ' área');
    return false;
}

function calcularperimetro(){
    var peri = new rectangulo(parseFloat(document.frmrectangulo.txtbase.value),
parseFloat(document.frmrectangulo.txtaltura.value));
    peri.mostrar(peri.cperimetro(), 'perímetro');
    return false;
}

//Creando una clase/ funcion constructora rectángulo
function rectangulo(base, altura){
    //Propiedades de la clase
    this.base = base;
    this.altura = altura;
    //Métodos de la clase
    //definidos usando el constructor Function()
    this.carea = function(){
        return this.base * this.altura;
    };
    this.cperimetro = function(){
        return 2*this.base + 2*this.altura;
    };
    this.mostrar = function(valor, tipoc){
        var result = document.getElementById('resultado');
        result.innerHTML = '<hr><div class="alert alert-success" role="alert"> El ' + tipoc
+ ' es: ' +
Math.round(valor*Math.pow(10,2))/Math.pow(10,2) + '</div>';
    };
}

//Asociando función que manejará el evento load al cargar la página
if(window.addEventListener){
    window.addEventListener("load", iniciar, false);
}
else if(window.attachEvent){
    window.attachEvent("onload", iniciar);
}
```

Resultado de la visualización en un navegador:



Ejemplo #2: En el siguiente ejemplo se muestra cómo crear un script JS que permita indicar en base a una zona horaria mundial seleccionada por el usuario la hora y fecha de esa zona horaria. En este ejemplo se hace uso de los objetos incorporados de JavaScript Date para manejar la fecha y el objeto Math para realizar los cálculos de desfase horario entre las zonas.

Guión 1: zonahoraria.html

```
<!DOCTYPE html>
<html lang="es">
<head>
    <title>Zona horaria internacional</title>
    <meta charset="utf-8" />
    <link   href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css"
rel="stylesheet"
        integrity="sha384-
EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC" crossorigin="anonymous">
    <script src="js/calculo.js"></script>
    <link          rel="stylesheet"          href="https://cdn.jsdelivr.net/npm/bootstrap-
icons@1.9.1/font/bootstrap-icons.css">
</head>
<body class="container pt-5">
    <header>
        <h1 class="text-center" id="pagetitle">Zonas del horario internacional</h1>
    </header>
    <div class="d-flex justify-content-center">
        <form name="zonahoraria" class="bg-light p-3 border border rounded w-50">
            <h1>
                Zonas horarias del mundo
                <span>Seleccione su zona horaria.</span>
            </h1>
```

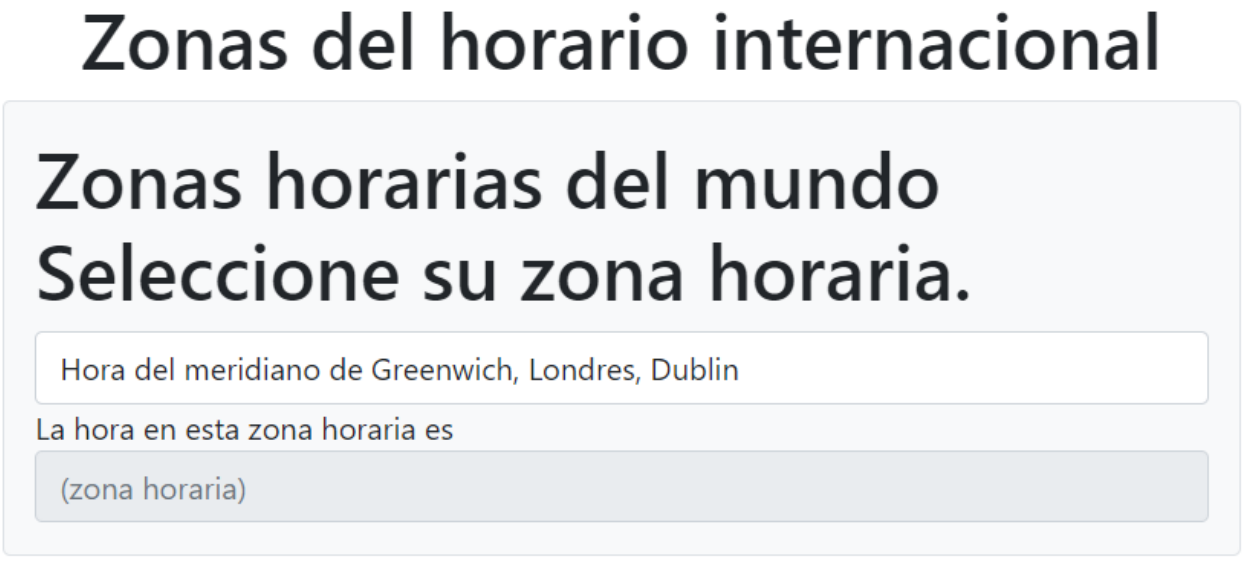
```
            <div class="form-group">
                <select name="zonas" id="zhselect" class="form-control">
                    <option value="-12">Línea internacional de fecha del oeste</option>
                    <option value="-11">Isla Midway, Samoa</option>
                    <option value="-10">Hawai</option>
                    <option value="-9">Alaska</option>
                    <option value="-8">Hora del Pacífico (USA y Canadá)</option>
                    <option value="-7">Hora de las montañas rocosas (USA y Canadá)</option>
                    <option value="-6">América Central</option>
                    <option value="-5">Hora central (USA y Canadá)</option>
                    <option value="-4">Hora del Atlántico (Canadá)</option>
                    <option      value="-3">Buenos      Aires,      Asunción,      Brasilia,
Montevideo</option>
                    <option value="-2">Atlántico Central</option>
                    <option value="-1">Azores</option>
                    <option value="0" selected="selected">Hora del meridiano de Greenwich,
Londres, Dublin</option>
                    <option value="+1">Paris, Madrid, Barcelona, Roma</option>
                    <option value="+2">El Cairo</option>
                    <option value="+3">Nairobi</option>
                    <option value="+4">Bakú</option>
                    <option value="+5">Ekaterimburgo</option>
                    <option value="+6">Astana</option>
                    <option value="+7">Bangkok</option>
                    <option value="+8">Ulán Bator</option>
                    <option value="+9">Tokio</option>
                    <option value="+10">Sidney</option>
                    <option value="+11">Islas Salomón</option>
                    <option value="+12">Wellington</option>
                </select>
            </div>
            <div class="form-group">

                <label class="control-label">La hora en esta zona horaria es</label>
                <input     type="text"     name="hour"     class="form-control"     id="hour"
placeholder="(zona horaria)"
                    readonly="readonly" />
            </div>
        </form>
    </div>
</body>
<script src="js/zonahoraria.js"></script>
</html>
```

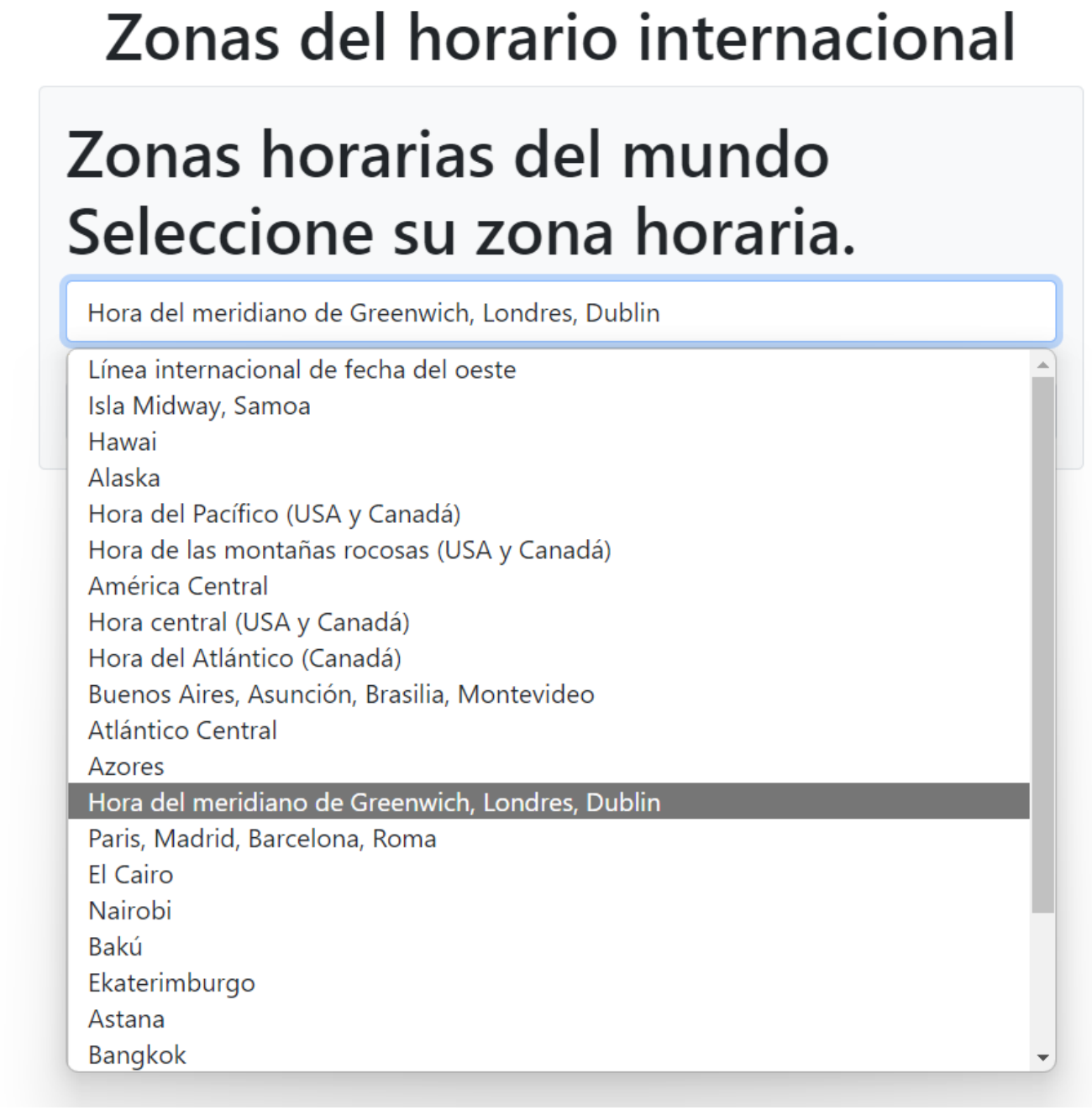
Guión 2: zonahoraria.js

```
//La fecha del día en la zona del cliente web
//usando el objeto Date de JavaScript
var fechaHoy = new Date();

//Estableciendo referencia para el título de la página
var title = document.getElementById('pagetitle');

//Cálculo del desfase en el huso horario para hacer
//luego el ajuste con la zona horaria seleccionada
//en el elemento select del formulario
var desfase = -(Math.round(fechaHoy.getTimezoneOffset()/60)) + 12;

//Registrar evento click del ratón al presionar botones de envío
function iniciar(){
    var select = document.getElementById("zhselect");
    if(select.addEventListener){
        select.addEventListener("change", getHoraLocal, false);
    }
```

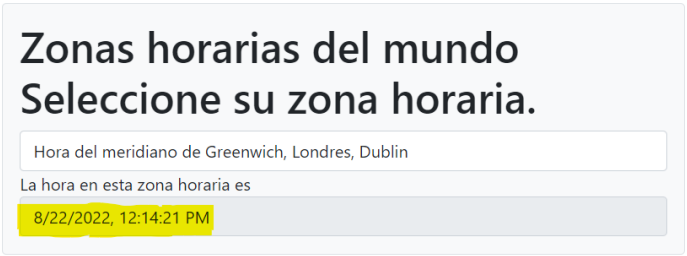
```
    else if(select.attachEvent){
        select.attachEvent("onchange", getHoraLocal);
    }
}

//Función getHoraLocal()
//Calcula la hora en la zona horaria seleccionada
function getHoraLocal(){
    var fechaHoy = new Date();
    //Ajustar el desfase horario respecto a la zona seleccionada.
    //Por ejemplo, la zona horaria para América Central está
    //en el índice 6 de la colección de opciones del elemento select.
    //Si un usuario selecionara la zona horaria de Tokio que está
    //en el índice 21 de la colección el cálculo se realizaría así:
    //21 - desfase del cliente (zona horaria de América Central) = 15
    var zh = document.zonahoraria.zonas.selectedIndex - desfase;

    //Sumar (o restar) las horas de desfase respecto a la hora
    //local del cliente
    fechaHoy.setHours(fechaHoy.getHours() + zh);

    //Informar la hora local del huso horario elegido
    title.innerHTML = "Zona horaria de " +
document.zonahoraria.zonas.options[document.zonahoraria.zonas.selectedIndex].text;
    document.zonahoraria.hour.value = fechaHoy.toLocaleString();
}

//Asociando función que manejará el evento load al cargar la página
if(window.addEventListener){
    window.addEventListener("load", iniciar, false);
}
else if(window.attachEvent){
    window.attachEvent("onload", iniciar);
}
```

Resultado al ejecutar el script con un navegador:





Zona horaria de Hora del meridiano de Greenwich, Londres, Dublin



Ejemplo #4: Ejemplo que permite ingresar en un formulario datos de un libro que luego son almacenados en propiedades de un objeto libro definido con el objeto Object de JavaScript. Se verifica que los datos obtenidos sean correctos, se limpian los campos del formulario y finalmente se construye una tabla HTML con uno de los métodos del objeto donde se muestran los datos ya almacenados en las propiedades de ese objeto creado.

Guión 1: libros.html

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

<head>
  <title>Creación de objetos</title>
  <meta charset="utf-8" />
  <link     href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css"
rel="stylesheet"
    integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC"
crossorigin="anonymous">
  <link           rel="stylesheet"           href="https://cdn.jsdelivr.net/npm/bootstrap-
icons@1.9.1/font/bootstrap-icons.css">

</head>

<body>
  <div class="container">
    <div class="row">
      <div>
        <h1 class="text-center">Información del libro</h1>
      </div>
    </div>
    <div class="row">
      <div class="col col-sm-12">
        <form action="javascript:void(0)" name="frmbook" id="frmbook" class="col s12">
          <div class="input-group mb-4">
            <div class="input-group-prepend">
              <div       class="input-group-text"><i       class="bi       bi-file-earmark-
person"></i></div>
            </div>
            <input type="text" placeholder="Ingrese el nombre del autor" name="txtautor"
class="form-control" id="autor"
              required maxlength="60" />
          </div>

          <div class="input-group mb-4">
            <div class="input-group-prepend">
              <div class="input-group-text"><i class="bi bi-book"></i></div>
            </div>
            <input type="text" placeholder="Ingrese Titulo de la obra" class="form-control"
name="txttitulo" id="titulo"
              maxlength="90" />
          </div>

          <div class="input-group mb-4">
            <div class="input-group-prepend">
              <div class="input-group-text"><i class="bi bi-bookmark-plus"></i></div>
            </div>
            <select name="seleditorial" class="form-control" id="editorial" required>
              <option value="" disabled selected>Seleccione la editorial</option>
              <option value="mcgraw">Mc-Graw Hill</option>
              <option value="pear">Pearson</option>
              <option value="alfa">Alfa-omega</option>
              <option value="anaya">Anaya</option>
            </select>
          </div>
```

```
          <div class="input-group mb-4">
            <div class="input-group-prepend">
              <div class="input-group-text"><i class="bi bi-sort-numeric-up"></i></div>
            </div>
            <select name="seledicion" class="form-control" size="1">
              <option value="" disabled selected>Seleccione la edición</option>
              <option value="a">1ra</option>
              <option value="b">2da</option>
              <option value="c">3ra</option>
              <option value="d">4ta</option>
              <option value="e">5ta</option>
              <option value="f">6ta</option>
              <option value="g">7ma</option>
              <option value="h">8va</option>
            </select>
          </div>

          <div class="input-group mb-4">
            <div class="input-group-prepend">
              <div class="input-group-text"><i class="bi bi-geo-alt-fill"></i></div>
            </div>
            <input type="text" class="form-control" placeholder="Ingrese la nacionalidad de
origen" name="txtpais"
              id="pais" maxlength="36" />
          </div>

          <div class="d-flex justify-content-center">
            <button type="submit" name="mostrar" id="mostrar" class="btn btn-success w-25
col-md-12">Enviar</button>
          </div>
        </form>
      </div>
    </div>
    <br>
    <div class="row">
      <hr>

      <h1 class="text-center">Lista autores</h1>
      <table class="table table-hover">
        <thead>
          <tr>
            <th>Id</th>
            <th>Titulo</th>
            <th>Autor</th>
            <th>Editorial</th>
            <th>Edicion</th>
            <th>Pais</th>
            <th>Operaciones</th>
          </tr>
        </thead>
        <tbody id="infolibro">

        </tbody>
      </table>
    </div>
  </div>
</body>

<script src="js/libro.js"></script>

</html>
```

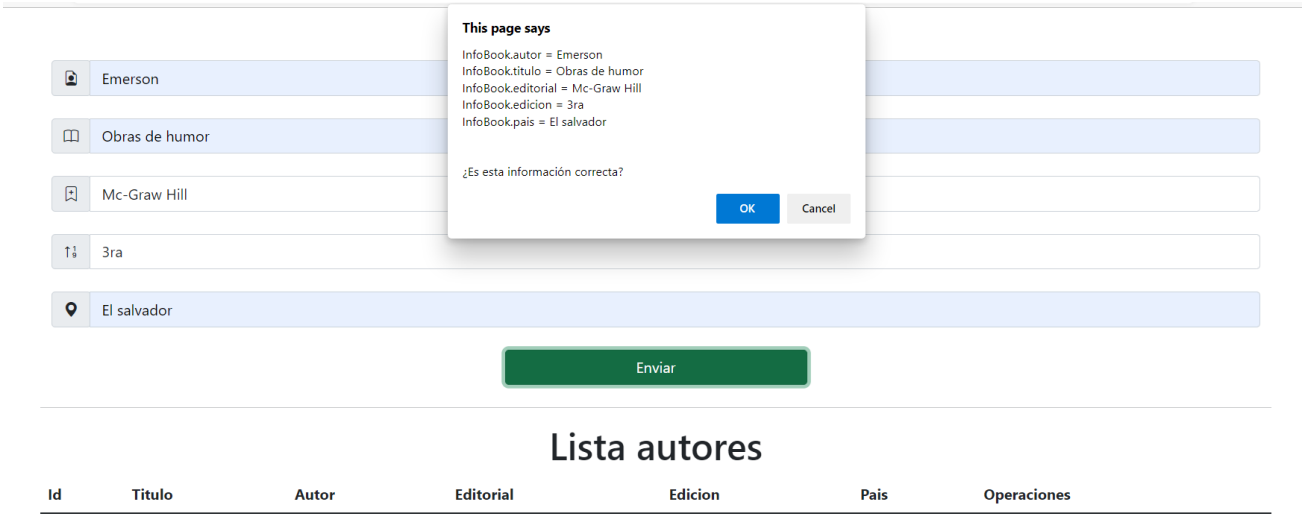
Guión 2: libro.js

```
//Creando la clase
class Book{
    book(){
        this.autor="";
        this.titulo="";
        this.editorial= ""
        this.edicion="";
        this.pais= "";
    }

    setAutor(autor){
        this.autor= autor;
    }
    getAutor(){
        return this.autor;
    }

    setTitulo(titulo){
        this.titulo= titulo;
    }
    getTitulo(){
        return this.titulo;
    }

    setEditorial(editorial){
        this.editorial= editorial;
    }
    getEditorial(){
        return this.editorial;
    }

    setEdicion(edicion){
        this.edicion= edicion;
    }
    getEdicion(){
        return this.edicion;
    }

    setPais(pais){
        this.pais= pais;
    }
    getPais(){
        return this.pais;
    }
}
//Registrar evento click del ratón al presionar botones de envío
function iniciar(){
    var showinfo = document.getElementById("mostrar");
    if(showinfo.addEventListener){
        showinfo.addEventListener("click", function(){
            createObject(document.frmbook);
        }, false);
    }
    else if(showinfo.attachEvent){
        showinfo.attachEvent("onclick", function(){
            createObject(document.frmbook);
        });
    }
}

// Creando el nuevo objeto
function createObject(form){
```

```
    const book = new Book();//generando la instancia de Book
    book.setAutor(form.txtautor.value);
    book.setTitulo(form.txttitulo.value);//cargando el objeto
    book.setEditorial(form.seleditorial.options[form.seleditorial.selectedIndex].text);
    book.setEdicion(form.seledicion.options[form.seledicion.selectedIndex].text);
    book.setPais(form.txtpais.value);
    showProperties(book, "InfoBook");
}

const arrarObjetos = new Array;//arreglo en el que se guardarn todos los objetos creados


function showProperties(objeto, objName){
    var infBook = "";
    var tblBook = "";
    var info = document.getElementById('infolibro');//Espacio donde se pintaran los objetos
    for(var i in objeto){
        infBook = infBook + objName + "." + i + " = " + objeto[i] + "\n";
    }
    if(!confirm(infBook + "\n\n¿Es esta información correcta?")){
        frmbook.txtautor.value = "";
        frmbook.txtitulo.value = "";
        frmbook.seleditorial.value = "a";
        frmbook.seledicion.value = "a";
        frmbook.txtpais.value = "";
    }

    arrarObjetos.push(objeto);//agregamos los objetos al arreglo

let id=1;//Se pintara en la tabla como el identificador
let posicion=0;//servira para definirle posicion a los objetos

    arrarObjetos.forEach(element => {
    tblBook += "\t<tr>\n";
    tblBook += "\t\t<td>" + id++ + "</td>\n";
    tblBook += "\t\t<td>" + element.getTitulo() + "</td>\n";
    tblBook += "\t\t<td>" + element.getAutor() + "</td>\n";
    tblBook += "\t\t<td>" + element.getEditorial() + "</td>\n";
    tblBook += "\t\t<td>" + element.getEdicion() + "</td>\n";
    tblBook += "\t\t<td>" + element.getPais() + "</td>\n";
    tblBook += "\t\t<td><button onclick='eliminar(" + posicion++ + ")'    class='btn btn-
danger'  >Eliminar</button></td>\n";
    tblBook += "\t\t</tr>\n";
      });


    info.innerHTML = tblBook;
}

function eliminar(valor){//funcion para eliminar los elementos del arreglo
var confirmacion=confirm("Esta seguro de eliminar este registro id = "+ valor);
if(confirmacion){
    arrarObjetos.splice(parseInt(valor) , 1);//definimos que eliminaremos un elemento desde
la posicion dada
    var tblBook = "";
    var info = document.getElementById('infolibro');
    let id=1;
    let posicion=0;
    arrarObjetos.forEach(element => {//se vuelve a pintar la tabla para ver el resultado
    tblBook += "\t<tr>\n";
    tblBook += "\t\t<td>" + id++ + "</td>\n";
    tblBook += "\t\t<td>" + element.getTitulo() + "</td>\n";
    tblBook += "\t\t<td>" + element.getAutor() + "</td>\n";
```

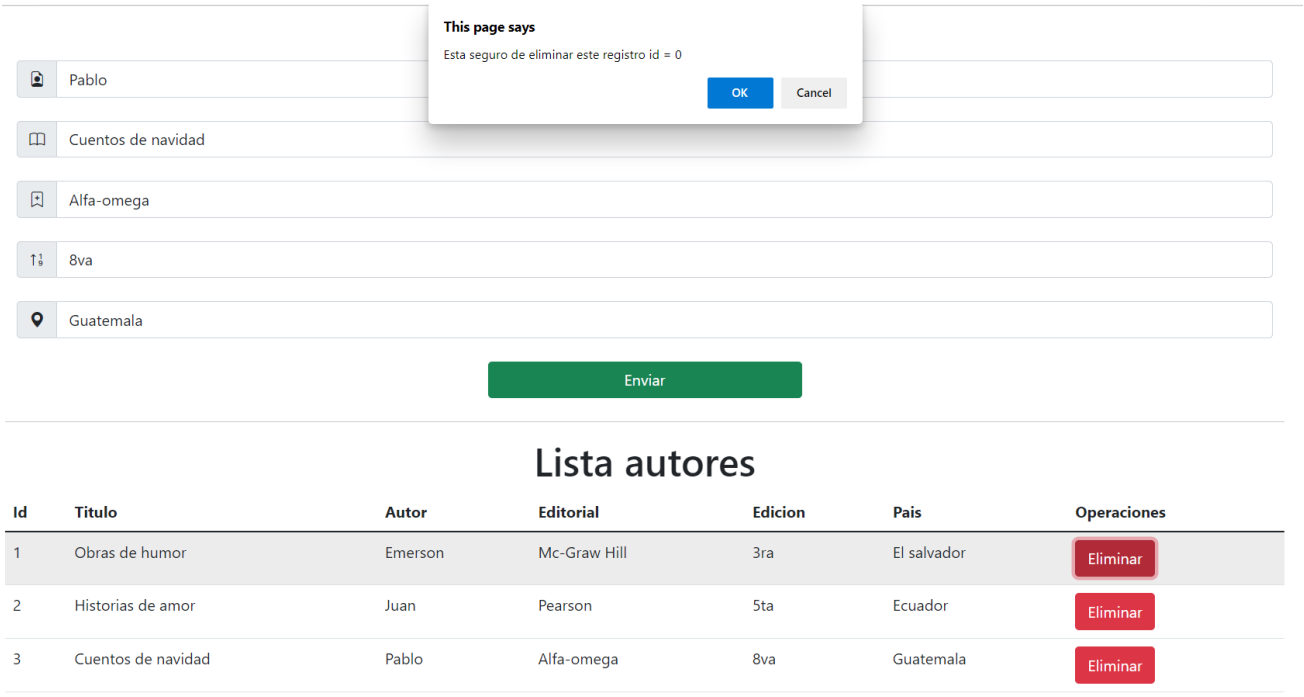
```
    tblBook += "\t\t<td>" + element.getEditorial() + "</td>\n";
    tblBook += "\t\t<td>" + element.getEdicion() + "</td>\n";
    tblBook += "\t\t<td>" + element.getPais() + "</td>\n";
    tblBook += "\t\t<td><button onclick='eliminar(" + posicion++ + ")'    class='btn btn-
danger'  >Eliminar</button></td>\n";
    tblBook += "\t\t</tr>\n";
      });

    info.innerHTML = tblBook;
}
}

//Asociando función que manejará el evento load al cargar la página
if(window.addEventListener){
    window.addEventListener("load", iniciar, false);
}
else if(window.attachEvent){
    window.attachEvent("onload", iniciar);
}
```

Resultado:
Agregando libro



Lista de objetos pintados en tabla

eliminación libro



Libro eliminado

V. DISCUSION DE RESULTADOS

1. Realice una aplicación orientada a objetos llamada potencia. Su aplicación debe crear una clase utilizando funciones, en donde las propiedades deben ser tres: la base y la potencia que deben obtener sus valores de dos argumentos enviados a la definición de la función que hará las veces de la clase. Una tercera propiedad para obtener la potencia de elevar la base a la potencia recibida. Su clase debe poseer dos métodos, uno para retornar el cálculo de la potencia con las propiedades de la clase y otro para mostrar en la página web mediante el uso de la propiedad innerHTML el resultado del cálculo de la potencia haciendo uso del método para tal propósito. Debe realizar una interfaz de formulario apropiada para este problema y no usar cuadros prompt para obtener los datos.
2. A partir de la utilización de listas con objetos planteado en el ejercicio #3; implementar un formulario web en el cual se solicite la información de un producto. Dado que el sistema actuará como un cajero en el cual se llenará un objeto llamado venta, ingresando los datos: nombre producto, precio unitario y cantidad a llevar. A partir de estos campos se deberá llenar una tabla con los registros de productos añadidos a la venta. Finalmente deberá calcular el total de toda la venta sumando y recorriendo la lista de productos agregados a la venta; esto ultimo obteniendo los valores de un atributo más de la clase venta llamado detalle, el cual será la multiplicación del precio venta y la cantidad.

VI. BIBLIOGRAFIA

- 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.
- Flanagan, David. JavaScript La Guía Definitiva. 1a Edición. Editorial ANAYA Multimedia. 2007. Madrid, España.
- Terry McNavage. JavaScript Edición 2012. 1a. Edición. Editorial ANAYA Multimedia. 2011. Madrid, España.
- Tom Negrino / Dori Smith. JavaScript & AJAX para diseño web. 6ta. Edición. Pearson/Prentice Hall. 2007. Madrid, España.