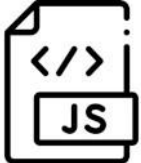
 UNIVERSIDAD DON BOSCO	UNIVERSIDAD DON BOSCO FACULTAD DE INGENIERIA ESCUELA DE COMPUTACION	
Ciclo II	Guía de laboratorio N° 2	
	Nombre de la práctica: Estructuras de control: Sentencias repetitivas. Lugar de ejecución: Centro de cómputo y Virtual. Materia: Desarrollo de Aplicaciones Web con Software Interpretado en el Cliente.	

I. Objetivos

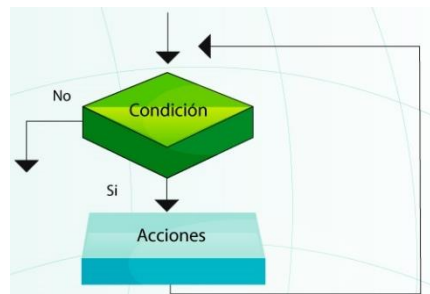
- ❖ Dominar el uso de las sentencias repetitivas, ciclos o lazos del lenguaje JavaScript.
- ❖ Implementar sentencias repetitivas en la solución de problemas que requieran repetir un conjunto de instrucciones.
- ❖ Implementar sentencias repetitivas anidadas con lenguaje JavaScript.
- ❖ Utilizar sentencias de control de ciclos o lazos (*break* y *continue*).

II. Introducción Teórica.

Sentencias repetitivas.

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

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



Sentencia *for*

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

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

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

inicialización: Se ejecuta una sola vez al principio del bucle. Usualmente se utiliza para declarar e inicializar una variable de contador.

condición: Se evalúa antes de cada iteración. Si es `true`, el bloque de código se ejecuta. Si es `false`, el bucle termina.

expresión final: Se ejecuta después de cada iteración del bloque de código. Usualmente se utiliza para actualizar la variable de contador (incrementarla o decrementarla).

Ejemplo:

```
document.write("Conteo hacia atrás<br>");
for(var i=10; i>=0;i--){
    document.write("<b>" + i + "</b>");
    document.write("<br>");
}
document.write("Fin del conteo.");
```

Utilizar for para recorrer elementos de una matriz

```
const deportes = ["Fútbol", "Basquet Ball", "Tenis", "Volley Ball", "Natación"];
document.write("Algunos deportes que se practican son:");
for (let i = 0; i < deportes.length; i++) {
    document.write(deportes[i]);
    document.write("<br>");
}
```

Sentencias *break* y *continue*.

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

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

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

En este ejemplo, el bucle se repetirá hasta que la variable **con** llegue al valor 20, entonces se ejecuta la sentencia ***break*** que hace terminar el bucle en ese punto.

El comando ***continue*** es un tanto diferente. Se utiliza para saltar a la siguiente repetición del bloque de código, sin completar el pase actual por el bloque de comandos.

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

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

Lazos *for* Anidados

Puedes tener un bucle **for** dentro de otro, lo cual es útil para trabajar con estructuras bidimensionales como matrices.

```
document.write("Lazos for Anidados - Matrices");
document.write("<br>");
for (let i = 1; i <= 4; i++) {
    document.write("*** Línea del Lazo Exterior "+i);
```

Guía # 2: Estructuras de control: sentencias repetitivas y matrices

```
document.write("<br>");
for (let j = 1; j <= 3; j++) {
    document.write(" ---> Línea del Lazo Interior: Línea "+j+" (dentro de La Línea "+i+" )");
    document.write("<br>");
}
}
```

Sentencia *for-in*.

Este es un lazo o bucle relacionado con objetos y se utiliza para recorrer las diversas propiedades de un objeto.

La sintaxis es la siguiente:

```
for (nombredevariable in objeto) {
    //instruccion o bloque de instrucciones;
}
```

Ejemplo:

```
document.write("<h3>Iterar sobre un objeto simple con for.. in</h3>");
const persona = {
    nombre: "Juan",
    edad: 22,
    profesion: "Estudiante",
    ciudad: "San Salvador"
};

for (const clave in persona) {
    document.write("Propiedad: " + clave + " <br> Valor: " +
persona[clave]);
    document.write("<br>");
    document.write("<br>");
}
```

Sentencia *for-of*.

El bucle `for...of` es la forma moderna y recomendada para iterar sobre valores de colecciones iterables como

Arrays, Strings, Mapas, Sets, etc.

Recorriendo un arreglo:

```
document.write("<h3>Recorriendo un array con for...of:</h3>");
const colores = ["Rojo", "Verde", "Azul", "Café"];
for (const color of colores) {
    document.write(color);
    document.write("<br>");
}
```

Recorriendo un string o cadena de texto:

```
document.write("<h3>Recorriendo un string con for...of:</h3>");
const palabra = "Hola Mundo";
for (const letra of palabra) {
    document.write(letra);
    document.write("<br>");
}
```

Guía # 2: Estructuras de control: sentencias repetitivas y matrices

El bucle *while*

El bucle *while* es otra estructura de control fundamental en JavaScript que te permite ejecutar un bloque de código mientras una condición específica sea verdadera. A diferencia del bucle *for*, que a menudo se usa cuando conoces el número de iteraciones de antemano, *while* es ideal cuando la duración del bucle depende de que se cumpla (o deje de cumplirse) una condición.

La estructura básica es:

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

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

Por ejemplo:

```
var nTotal = 35;  
var con = 3;  
while (con <= nTotal) {  
    alert("El contador con es igual a: " + con);  
    con++;  
}
```

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

El bucle *do- while*

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

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

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

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

Sentencia *with*

Sintaxis:

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

La instrucción *with* permite utilizar una notación abreviada al hacer referencia a los objetos. Es muy conveniente a la hora de escribir dentro del código del script un conjunto de instrucciones repetitivas relacionada con el mismo objeto. Por ejemplo, si se tiene el siguiente conjunto de instrucciones:

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

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

```
with(document){
```

Guía # 2: Estructuras de control: sentencias repetitivas y matrices

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

III. Materiales y Equipo.

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

No.	Requerimiento	Cantidad
1	Guía #2: Estructuras de control: sentencias repetitivas	1
2	Computadora con editor HTML/JavaScript y navegadores instalados	1

IV. Procedimiento.

Ejercicio #1: Encuesta que permite votar y seguir opinando sobre una pregunta. Cada vez que se emite la opinión se vuelve a preguntar si desea seguir votando. Mientras responda que si desea continuar, se le volverá a pedir la opinión, hasta que responda que ya no. En ese momento se emitirán los resultados. Se utiliza una técnica conocida como ciclo infinito para implementar la solución.

Guión 1: encuesta.html

```
<!DOCTYPE html>  
<html lang="es">  
<head>  
  <title>Encuesta sobre ley antimaras</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"></scrip  
t>  
</head>  
<body>  
  <script src="js/encuesta.js"></script>  
  
</body>  
</html>
```

Guión 2: encuesta.js

```
//Declaración de variables  
var voto;  
var opcion = true;  
var cont1 = cont2 = cont3 = 0;  
var total;  
var per1, per2, per3;  
  
//Mostrar las instrucciones para responder  
document.write("<div class=\"container\">");  
document.write("<div class=\"row\">");  
document.write("<h1 class=\"text-center\">");  
document.write("Encuesta para determinar cuántas personas están a favor de la portabilidad  
numérica de teléfonos");  
document.write("celulares.");  
document.write("</h1>");  
document.write("</div>");  
document.write("<div class=\"row\">");  
document.write("<ul class=\"list-group list-group-flush\">");  
document.write("<li class=\"list-group-item fw-bold\">Digite \"1\" si esta a favor</li>");  
document.write("<li class=\"list-group-item fw-bold\">Digite \"2\" si esta en  
contra</li>");  
document.write("<li class=\"list-group-item fw-bold\">Digite \"3\" si se abstiene de  
opinar</li>");  
document.write("</ul>");
```

Guía # 2: Estructuras de control: sentencias repetitivas y matrices

```
document.write("</div>");
document.write("</div>");
//Ciclo repetitivo infinito donde se captura voto por voto
//en tanto no se de por terminada el ingreso de respuesta de la encuesta
while (opcion == true) {
    voto = parseInt(prompt('¿Cuál es su voto?', ''));
    switch (voto) {
        case 1:
            cont1++;
            break;
        case 2:
            cont2++;
            break;
        case 3:
            cont3++;
            break;
        default:
            alert('¡Voto no válido!');
    }
    //Se pregunta si se desea terminar la encuesta o continuar
    opcion = confirm('¿Desea continuar s/n?');
}

//Obtener el total de respuestas de la encuesta
total = cont1 + cont2 + cont3;
//Obtener el porcentajes de la primera respuesta
per1 = Math.round((cont1 / total) * Math.pow(10, 2)) / Math.pow(10, 2);
//Obtener el porcentajes de la segunda respuesta
per2 = Math.round((cont2 / total) * Math.pow(10, 2)) / Math.pow(10, 2);
//Obtener el porcentajes de la tercera respuesta
per3 = Math.round((cont3 / total) * Math.pow(10, 2)) / Math.pow(10, 2);
//Mostrar los resultados de la encuesta
with (document) {
    write("<div class=\"container\">");
    write("<div class=\"row\">");
    write("<table class=\"table table-primary table-striped table-hover\">");
    write("<thead>");
    write("<tr>");
    write("<th>Resultado de los votos</th>");
    write("<th>Votos obtenidos</th>");
    write("<th>Porcentaje</th>");
    write("</tr>");
    write("</thead>");
    write("<tbody>");
    write("<tr>");
    write("<td>Votos a favor </td>");
    write("<td>" + cont1 + "</td>");
    write("<td>" + per1 * 100 + " %</td>");
    write("</tr>");
    write("<tr>");
    write("<td>Votos en contra </td>");
    write("<td>" + cont2 + "</td>");
    write("<td>" + per2 * 100 + " %</td>");
    write("</tr>");
    write("<tr>");
    write("<td>Se abstienen de opinar </td>");
    write("<td>" + cont3 + "</td>");
    write("<td>" + per3 * 100 + " %</td>");
    write("</tr>");
    write("</tbody>");
    write("<tfoot>");
    write("<tr>");
    write("<th>Totales</th>");
    write("<th>" + parseInt(cont1 + cont2 + cont3) + "</th>");
    write("<th>" + (parseFloat(per1 + per2 + per3)) * 100 + " %</th>");
    write("</tr>");
    write("</tfoot>");
    write("</table>");
    write("</div>");
    write("</div>");
}
```

Guía # 2: Estructuras de control: sentencias repetitivas y matrices

Indicaciones:

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

Resultado:

Esta página dice

¿Cuál es su voto?

1

Aceptar Cancelar

Esta página dice

¿Desea continuar s/n?

Aceptar Cancelar

Encuesta para determinar cuántas personas están a favor de la portabilidad numérica de teléfonos celulares.

Digite "1" si esta a favor

Digite "2" si esta en contra

Digite "3" si se abstiene de opinar

Resultado de los votos	Votos obtenidos	Porcentaje
Votos a favor	6	67 %
Votos en contra	2	22 %
Se abstienen de opinar	1	11 %
Totales	9	100 %

Ejercicio #2: Creación de campos de formulario del tipo seleccionado en tiempo de ejecución con JavaScript.

Guión 1: forms.html

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

<head>
  <title>Creación de controles de formulario dinámicamente</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/formcontrols.js"></script>
</head>

<body>

  <div class="container-fluid bg-info">
    <div class="row">
      <h1 class="display-1 text-center text-white">Creación de formularios dinámicos</h1>
    </div>
  </div>
  <div class="container bg-light p-2">
    <div class="row position-relative">
      <form action="javascript:void(0);" name="frmconf" id="form">
        <div class="input-group mb-3">
          <select class="form-select" name="selcontrol" id="selcontrol">
            <option value="" selected="selected">Tipo de control</option>
            <option value="text">Cuadro de texto</option>
            <option value="password">Cuadro de contraseña</option>
            <option value="textarea">Áreas de texto</option>
            <option value="checkbox">Casillas de verificación</option>
            <option value="radio">Botones de opción</option>
            <option value="file">Control de archivo</option>
            <option value="button">Botones genéricos</option>
          </select>
        </div>
        <div class="input-group mb-3">
          <span class="input-group-text" id="basic-addon1">Número de controles</span>
          <input type="text" name="txtnum" maxlength="2" autofocus class="form-control"
placeholder="Número de controles">
        </div>
      </form>
    </div>
  </div>
</body>
</html>
```

Guía # 2: Estructuras de control: sentencias repetitivas y matrices

```
        </div>

        <input class="btn btn-outline-primary" type="submit" name="cmdenviar" value="Enviar">

    </div>
</div>
<div class="container bg-light p-2">
    <div class="row" id="view"></div>
</div>
</body>

</html>
```

Guión 2: formcontrols.js

```
function init(){
    var form = document.getElementById('form');
    form.onsubmit = function(){
        createform(document.frmconf.selcontrol.value,document.frmconf.txtnum.value);
    }
}

function createform(control, numero){
    var htmlform, tag, i;
    //Referenciar al elemento de la página web donde se
    //mostrará el formulario creado
    var formview = document.getElementById('view');
    htmlform = "<div class=\"row position-relative\">";
    htmlform += "<form name=\"miform\">\n";
    htmlform += '<h1 class="display-1 text-center">Formulario dinámico</h1>';

    with(document){
        //Dependiendo del tipo de control
        switch(control){
            case "text":
            case "password":
                for(i=0; i<numero; i++){
                    tag = "<input class=\"form-control\" type=\"\" + control + "\"" name=\"" + control +
(i+1) + "\"" required placeholder="Ingrese los datos en el campo " + control + "\"" /><br>\n";
                    htmlform += tag;
                }
                break;
            case "textarea":
                for(i=0; i<numero; i++){
                    tag = "<textarea class=\"form-control\" name=\"" + control + (i+1) + "\"" required
placeholder="Ingrese los datos en el campo " + control + "\"" /></textarea><br />\n";
                    htmlform += tag;
                }
                break;
            case "checkbox":
                for(i=0; i<numero; i++){
                    tag = "<div>\n<input class=\"form-check-input\" type=\"" + control + "\"" name=\""
+ control + (i+1) + "\"" id=\"" + control + (i+1) + "\"" />\n";
                    tag += "<label class=\"form-check-label\" for=\"" + control + (i+1) + "\"">\n";
                    tag += control + (i+1) + "</label>\n</div>" + "\n";
                    htmlform += tag;
                }
                break;
            case "radio":
                for(i=0; i<numero; i++){
                    tag = "<div>\n<label class=\"form-check-label\" for=\"" + control + (i+1) + "\"">\n";
                    tag += "<t<input class=\"form-check-input\" type=\"" + control + "\"" name=\"" +
control + "\"" id=\"" + control + (i+1) + "\"" />\n";
                    tag += "<t<span>" + control + (i+1) + "</span>\n</label>\n</div>" + "\n";
                    htmlform += tag;
                }
                break;
            case "file":
                for(i=0; i<numero; i++){
                    tag = "<label class=\"custom-file-input file-blue\"><br />\n";
                    tag += "<t<input class=\"form-control\" type=\"file\" name=\"" + control + (i+1) +
\"" /><br />\n";
                    tag += "</label><br />\n";
                    htmlform += tag;
                }
                break;
            case "button":
                for(i=0; i<numero; i++){
                    tag = "<button class=\"btn btn-primary m-1\" name=\"" + control + (i+1) + "\""> +
control + (i+1) + "</button><br />\n";
                    htmlform += tag;
                }
            }
        }
    }
}
```


Guía # 2: Estructuras de control: sentencias repetitivas y matrices

```
    }
    break;
  default:
    alert("No ha seleccionado el tipo de control");
    return;
    break;
  }
  htmlform += "</form>\n";
}
htmlform += "</div>";
formview.innerHTML = htmlform;
}

window.onload = init;
```

Indicaciones:

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

Resultado en el navegador:

Creación de formularios dinámicos

Cuadro de texto

Número de controles 4

Enviar

Formulario dinámico

Ingrese los datos en el campo text

Ingrese los datos en el campo text

Ingrese los datos en el campo text

Ingrese los datos en el campo text

Creación de formularios dinámicos

Botones de opción

Número de controles 3

Enviar

Formulario dinámico

☐ radio1

☒ radio2

☐ radio3

Ejemplo #3: Uso de la sentencia with para reducir la cantidad de código a escribir cuando se recorren las propiedades de un objeto.

Guión 1: math.html

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Cálculos de Círculo y Rectángulo</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet"
  integrity="sha384-QWTKZyjpPEjISv5WaRU90FeRpok6YctnYmDr5pNlyT2bRjXh0JMhJY6hW+ALEwIH" crossorigin="anonymous">
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"
  integrity="sha384-YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDz0xhy9GkcIdslK1eN7N6jIeHz"
  crossorigin="anonymous"></script>
</head>
<body>
  <div class="container mt-5">
    <div class="card">
      <div class="card-body">
```

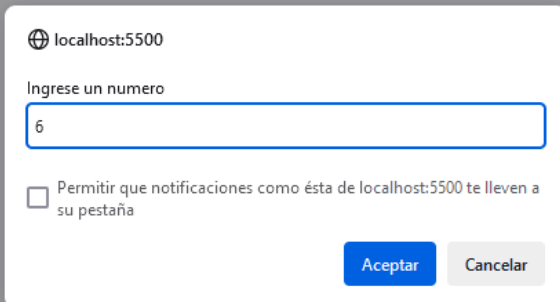
Guía # 2: Estructuras de control: sentencias repetitivas y matrices

```
<h5 class="card-title">Resultados de los Cálculos</h5>
<div id="resultados"></div>
</div>
</div>
</body>
<script type="text/javascript" src="js/with.js"></script>
</html>
```

Guión 2: with.js

```
//Inicializar variables
var area, peri, coorx, coory;
//Solicitar el valor para el radio del círculo
var radio = parseInt(prompt('Ingrese un numero',''));

//Recorrer propiedades del objeto Math usando la instrucción with
with(Math){
  //Área de un círculo de radio "radio"
  area = PI*radio*radio;
  //Valor del lado horizontal definido por el radio
  coorx = abs(radio*cos(PI/4));
  //Valor del lado vertical definido por el radio
  coory = abs(radio*sin(PI/4));
  pericir = 2*PI*radio;
  perirec = 2*coorx + 2*coory;
  //Invocar la propiedad write del objeto documento con with
  with(document.getElementById('resultados')){
    innerHTML = `
      <p>El área es: ${area.toFixed(2)}</p>
      <p>El lado x del rectángulo generado es: ${coorx.toFixed(2)}</p>
      <p>El lado y del rectángulo generado es: ${coory.toFixed(2)}</p>
      <p>El perímetro del círculo es: ${pericir.toFixed(2)}</p>
      <p>El perímetro del rectángulo es: ${perirec.toFixed(2)}</p>
    `;
  }
}
```



localhost:5500

Ingrese un numero

6

☐ Permitir que notificaciones como ésta de localhost:5500 te lleven a su pestaña

Aceptar Cancelar

Resultados de los Cálculos

El área es: 113.10

El lado x del rectángulo generado es: 4.24

El lado y del rectángulo generado es: 4.24

El perímetro del círculo es: 37.70

El perímetro del rectángulo es: 16.97

Ejemplo #4: Uso de menús interactivos para seleccionar destinos turísticos por ciudad y país.

Guión 1: ciudades.html

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

<head>
  <title>Interacción con menús de selección</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/ciudades.js"></script>
</head>

<body class="bg-light">
  <div class="container-fluid">
    <div class="row">
      <h1 class="display-1 text-center text-dark">Selector de destino de vacaciones</h1>
    </div>
  </div>
  <div class="container">
    <div class="row position-relative bg-white text-center rounded">
      <!-- Title -->
      <p class="display-3">País destino</p>
      <!-- Intro text -->
      <p class="text-dark">
        Seleccione <b class="text-warning">el país</b> al que desea ir:
      </p>
      <!-- Inicio de Contact Form -->
      <div class="w-100">
        <!-- Campos de formulario -->
        <form action="javascript:void(0);" name="testform" id="testform" method="post">
          <!-- Campo de selección del país -->
          <div class="field">
            <select name="country" id="country" class="form-select m-2">
              <option selected="selected" class="disabled">Italia</option>
              <option>Francia</option>
              <option>España</option>
              <option>Estados Unidos</option>
            </select>
          </div>
          <!-- Campo de selección de la ciudad destino -->
          <div class="field">
            <select name="city" id="city" class="form-select m-2" size="4">
              <option>Roma</option>
              <option>Turín</option>
              <option>Milán</option>
              <option>Venecia</option>
              <option>Verona</option>
            </select>
          </div>
          <!-- Send button -->
          <input type="button" name="btnagregar" id="btnagregar" value="Agregar ciudad ->" class="btn btn-outline-primary m-2" />
        </form>
        <!-- / Form fields -->
      </div>
    </div>
  </div>
</body>
```

</html>

Guión 2: js/ciudades.js

```
function iniciar(){
    //Elementos de la página sobre los que se detectarán eventos
    var selcountry = document.getElementById('country');
    var addcity = document.getElementById('btnagregar');
    //Creando un arreglo para guardar las ciudades de cada país
    var cities = new Array(4);
    cities["Italia"] = ["Roma", "Turín", "Milán", "Venecia", "Verona"];
    cities["Francia"] = ["Paris", "Lion", "Niza", "Mónaco"];
    cities["España"] = ["Madrid", "Barcelona", "Valencia", "Sevilla"];
    cities["Estados Unidos"] = ["Washington", "Florida", "San Francisco", "New York", "Houston"];

    selcountry.onchange = function(){
        addOptions(cities[this.options[this.selectedIndex].text], document.testform.city);
    }

    //Asociando el manejador de evento click
    addcity.onclick = function(){
        addCity(cities[document.testform.country.options[document.testform.country.selectedIndex].text],
document.testform.city)
    }
}

//Esta función limpia todas las opciones del menú desplegable de las ciudades
function removeOptions(optionMenu){
    for(i=0; i<optionMenu.options.length; i++){
        optionMenu.options[i] = null;
    }
}

//Esta función agrega nuevas opciones en el menú desplegable
//de las ciudades, dependiendo del país seleccionados.
//Las ciudades para llenar el campo son tomadas del
//array asociativo correspondiente
function addOptions(optionList, optionMenu){
    var i=0;
    //Limpia las opciones
    removeOptions(optionMenu);
    for(i=0; i<optionList.length; i++){
        optionMenu[i] = new Option(optionList[i], optionList[i]);
    }
}

//Permite agregar dinámicamente una ciudad al país actual
function addCity(optionList, optionMenu){
    var newcity;
    do{
        newcity = prompt("Ingrese la ciudad que desea agregar:","");
    }while(newcity == null || newcity == undefined || newcity.length == 0);
    optionList.push(newcity);
    removeOptions(optionMenu);
    addOptions(optionList, optionMenu);
}

window.onload = iniciar;
```

Selector de destino de vacaciones

País destino

Seleccione el país al que desea ir:

Italia

Roma
Turín
Milán
Venecia

Agregar ciudad ->

V. Discusión de Resultados.

1. Cree un script que permita el ingreso de un número entero y muestre en pantalla la siguiente información: 1) Cantidad de cifras, 2) Cantidad de cifras impares, 3) Cantidad de cifras pares, 4) Suma de cifras impares, 5) Suma de cifras pares, 6) Suma de todas las cifras, 7) Cifra mayor, 8) Cifra menor.
2. Crea un script que, al ingresar una palabra, realice los siguientes análisis: a) Cuente cuántas vocales hay en la palabra. b) Imprima la palabra al revés y determine si es un palíndromo.
3. Escribe un programa que usando bucles pueda calcular el factorial de un número dado.

VI. Investigación Complementaria.

1. Investigue para qué se utilizan las siguientes funciones del objeto Math: abs(), round(), ceil(), floor(), exp(), log() y random(). Ponga un ejemplo breve, de su utilización.
2. Buque la lista de propiedades del objeto document y su utilización. Como document.open(), document.close(), document.write(), Así como métodos para Acceder a elementos, para Manipular elementos, para Manipular estilos y métodos para Gestionar eventos.

VII. Bibliografía.

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