



# TALLER DE PROGRAMACIÓN WEB



## SEPARATA II: PROGRAMANDO CON JAVASCRIPT

Eric Gustavo Coronel Castillo  
gcoronel@uni.edu.pe  
gcoronelc.github.io

# ÍNDICE

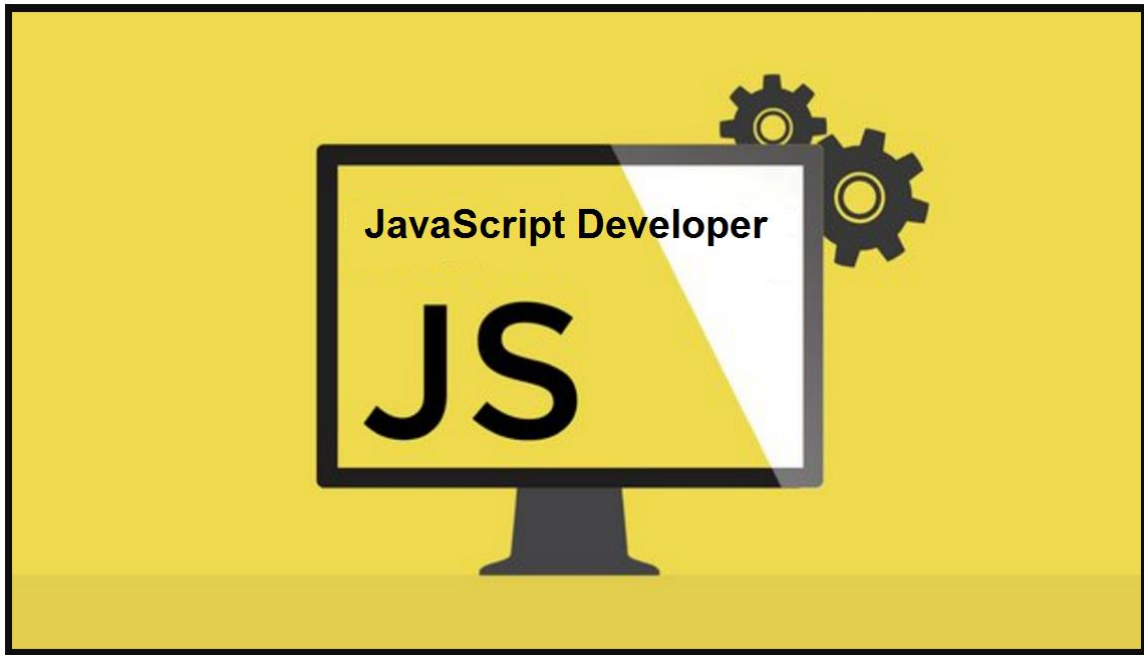
<b>CAPÍTULO 1 FUNDAMENTOS DE JAVASCRIPT.....</b>	<b>5</b>
OBJETIVO .....	5
INTRODUCCIÓN .....	6
¿QUÉ ES JAVASCRIPT?.....	7
JAVASCRIPT SE INTEGRA CON PÁGINAS WEB .....	8
JAVASCRIPT LENGUAJE INTERPRETADO .....	9
JAVASCRIPT ES CASE SENSITIVE.....	10
COMENTARIOS .....	13
VARIABLES.....	14
OPERADORES .....	16
Operadores Aritméticos .....	16
Operadores de Asignación .....	17
Operadores de Comparación.....	17
Operadores Lógicos .....	18
OTROS OPERADORES .....	18
MENSAJES AL USUARIO .....	19
LECTURA DE DATOS .....	20
CONDICIONALES.....	21
Operador ternario.....	21
Estructura if.....	21
BUCLES.....	23
Bucle: For.....	23
Bucle: while .....	24
Bucle: Do While.....	25
Bucle: For In .....	26
Sentencias break y continue.....	27
Instrucción: WITH .....	28
Instrucción: Switch.....	29
EJERCICIOS PROPUESTOS.....	31

<b>CAPÍTULO 2 APLICANDO JAVASCRIPT.....</b>	<b>32</b>
INTRODUCCIÓN .....	32
FUNCIONES PREDEFINIDAS.....	34
<i>Función: eval()</i> .....	34
<i>Función: parseInt()</i> .....	35
<i>Función: parseFloat()</i> .....	36
<i>Función: isNaN()</i> .....	37
<i>Función: isFinite()</i> .....	39
<i>Paso de Varios Argumentos a una Función</i> .....	40
OBJETOS PREDEFINIDOS .....	42
<i>Objeto: Date</i> .....	42
<i>Objeto: Math</i> .....	45

# Capítulo 1

## FUNDAMENTOS DE JAVASCRIPT

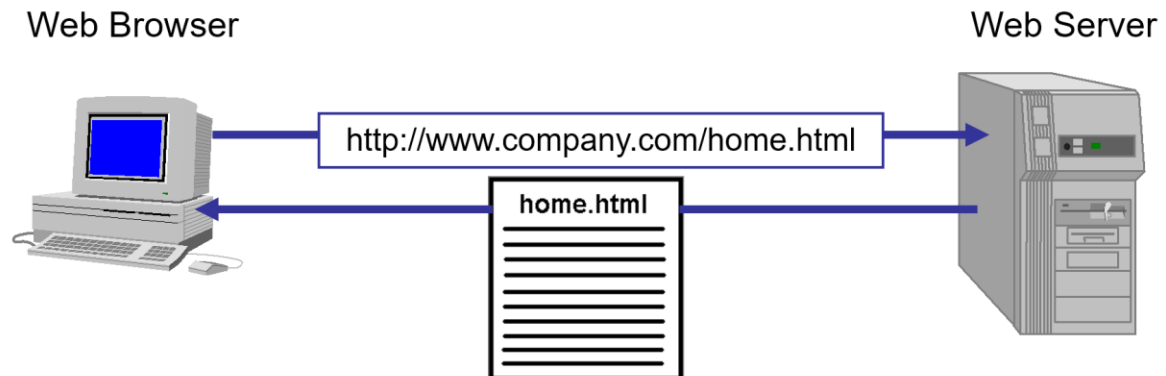
### OBJETIVO



El objetivo es que apliques JavaScript (JS) para solucionar problemas sencillos.

Siendo JS el lenguaje utilizado para desarrollar las interfaces de usuario en las aplicaciones Web, junto con HTML y CSS, es de vital importancia que tú aprendas los fundamentos de JS para que no tengas problemas cuando desarrolles aplicaciones Web.

# INTRODUCCIÓN



- JS es el complemento perfecto para HTML y CSS.
- Permite a las páginas realizar algunas tareas por sí misma, sin necesidad de sobrecargar el servidor del cual dependen.
- Permiten realizar cálculos simples, formatear texto para que pueda ser leído por distintas personas de manera diferente.
- Provee de medios para configurar la visualización de una página.

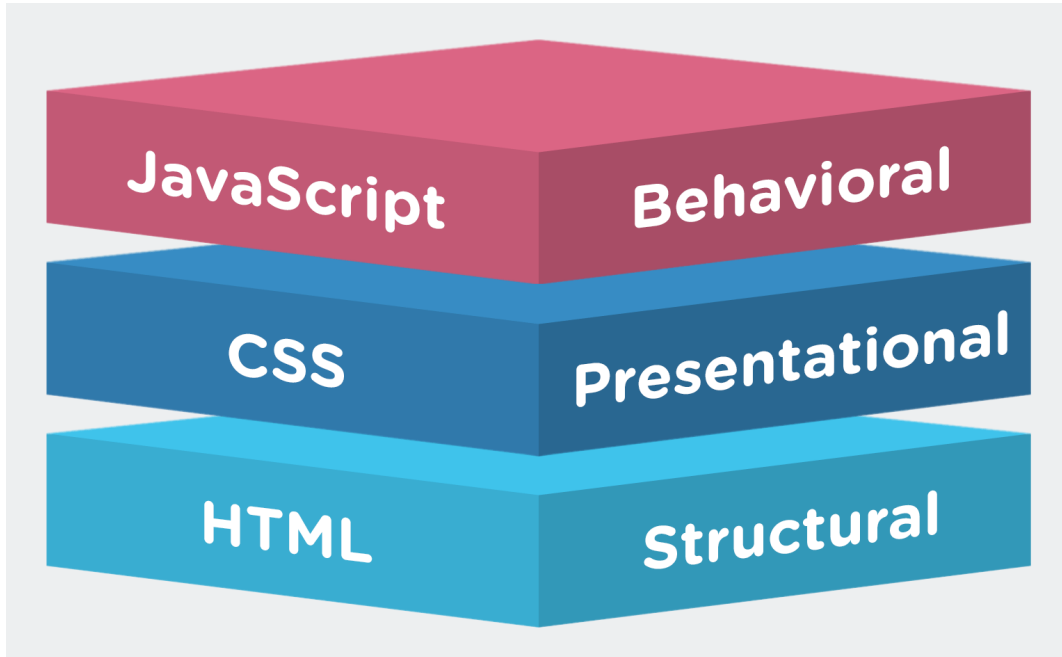
## ¿QUÉ ES JAVASCRIPT?



- Es un Lenguaje de tipo script, basado en objetos, y guiado por eventos, para ser aplicado en las páginas Web.
- JavaScript agrega dinamismo y permite procesamiento de datos.
- Proporciona un conjunto de objetos que permiten trabajar muy bien con los elementos de un documento Web.

```
<SCRIPT LANGUAGE="JavaScript">  
  
    // Aquí escribes el código JavaScript  
  
</SCRIPT>
```

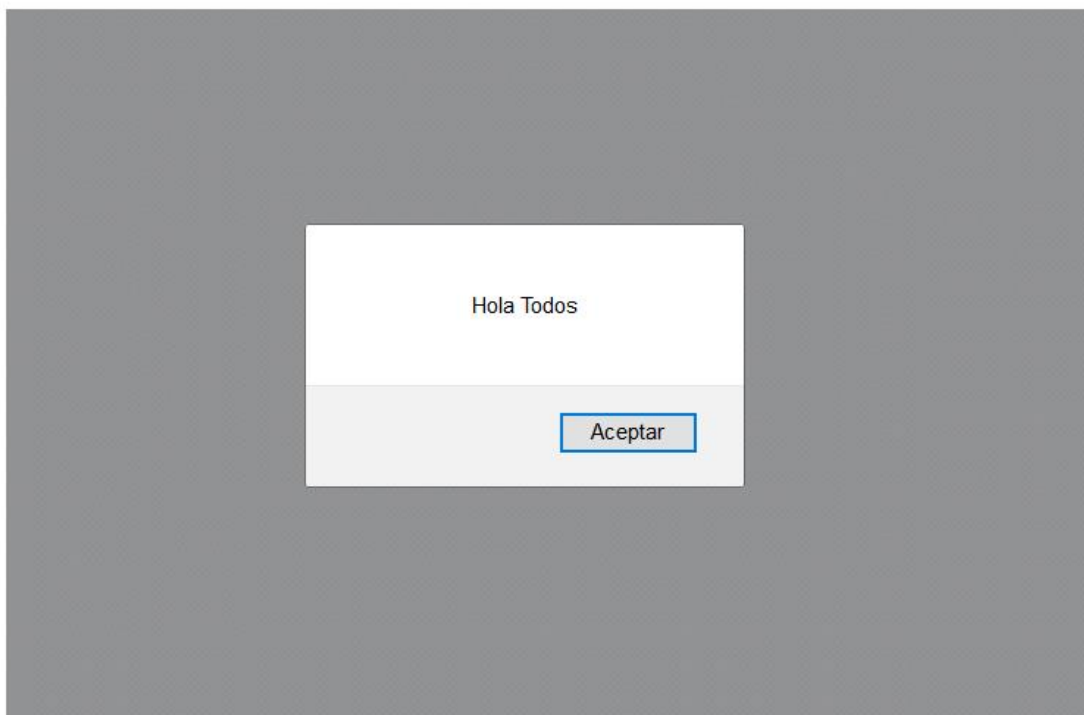
## JAVASCRIPT SE INTEGRA CON PÁGINAS WEB



Programas en JavaScript están incluidos en la página Web donde se ejecutará.

```
<BODY>  
...  
...  
  
<SCRIPT LANGUAGE="JavaScript">  
// Aquí escribes el código JavaScript  
  
</SCRIPT>  
  
</BODY>
```

# JAVASCRIPT LENGUAJE INTERPRETADO



El código JS es interpretado por el Browser.

## **Ventajas**

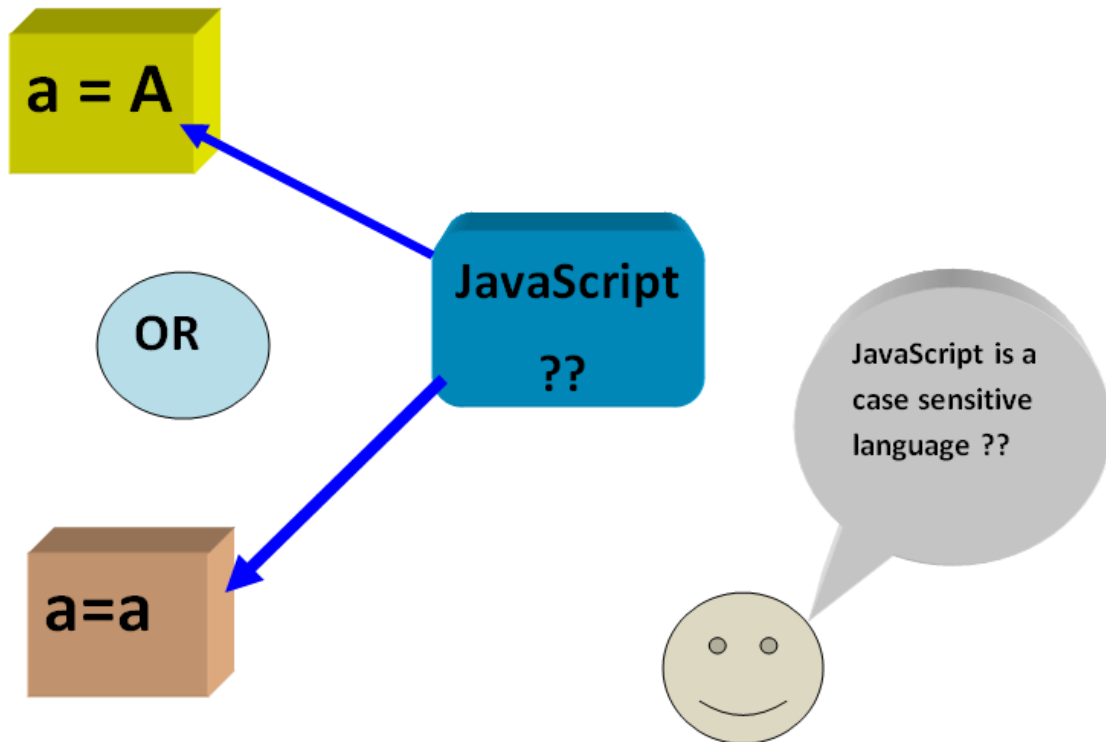
- En todo momento es posible actualizar el código fuente.
- Viaja con el documento HTML.
- Son fáciles de aprender.

## **Desventajas**

- El código fuente siempre será visible para todos.



## JAVASCRIPT ES CASE SENSITIVE



**Case Sensitive** es el término inglés para expresar que un lenguaje de programación es capaz de distinguir entre mayúsculas y minúsculas, y se comporta de manera estricta a la hora de ejecutarlas.

Las siguientes variables son reconocidas como distintas:

- Cont
- cont
- CONT

## Ejemplo 1

Archivo: ejm01.html

```
<script>

y= 5;
Y = 10;

r1= y + y;
r2 = y + Y;

document.write( "<p> y = " + y + "</p>" );
document.write( "<p> Y = " + y + "</p>" );
document.write( "<p> y + y = " + r1 + "</p>" );
document.write( "<p> y + Y = " + r2 + "</p>" );

</script>
```

El resultado es el siguiente:

```
y = 5
Y = 5
y + y = 10
y + Y = 15
```

## Ejemplo 2

Archivo: ejm02.html

```
<HTML>
<HEAD><TITLE>JavaScript</TITLE></HEAD>
<BODY>
  Mensaje de JavaScript:
  <SCRIPT LANGUAGE="JavaScript">
    <!-- // Inicia Programa
      document.write("<H1>Bienvenido a JavaScript</H1>");
    // Fin de Programa -->
  </SCRIPT>
</BODY>
</HTML>
```

## Ejemplo 3

Archivo: prog.js

```
<!-- // Inicia Programa
  document.write("<H1>Bienvenido a JavaScript</H1>");
// Fin de Programa -->
```

Archivo: ejm03.html

```
<HTML>
<HEAD>
  <TITLE>JavaScript</TITLE>
</HEAD>
<BODY>
  Mensaje de JavaScript.
  <SCRIPT LANGUAGE="JavaScript" SRC="P0101.js"></SCRIPT>
</BODY>
</HTML>
```

# COMENTARIOS

## Comentarios en una línea:

```
<SCRIPT LANGUAGE="JavaScript">

// Esto es un Comentario
document.getElementById("id").innerHTML = "1";

// Esto tambien es un comentario y no es tenido en cuenta.
document.getElementById("name").innerHTML = "Gustavo Coronel";

var saldo = 5;           // Esta variable inicia con un valor de 5
var res = saldo + 2;     // Esta variable guarda el valor de la variable saldo mas 2

</SCRIPT>
```

## Comentarios Multi-Línea

```
<SCRIPT LANGUAGE="JavaScript">

/*
El siguiente código realiza:

Asignacion de valor 1 al elemento id
Asignacion de valor Gustavo Coronel al elemento name
Datos del programador:
Sitio Web: gcoronelc.blogspot.com
Email: gcoronelc@gmail.com
*/

document.getElementById("id").innerHTML = "1";

document.getElementById("name").innerHTML = "Gustavo Coronel.";

</SCRIPT>
```

## VARIABLES

- Cuando se declara una variable, no se especifica su tipo:  
`var nombreVariable;`
- El nombre de las variables tiene que empezar por una letra o guión bajo:

Variables bien definidas	Variables mal definidas
tres_mosqueteros	3mosqueteros
Nombre	*nombre
_id	(id)

El tipo de la variable está determinado por el dato almacenado en ella.

En JavaScript el nombre de una variable es sensible a las mayúsculas y minúsculas.

Los tipos de datos disponibles son:

- Numérico (numeric)
- Cadena (string)
- Booleano (boolean)
- Objeto (object)
- Función (function)
- Nulo (null)

## Ejemplo 4

En este ejemplo se ilustra cómo identificar el tipo de datos de una variable.

Archivo: ejm04.html

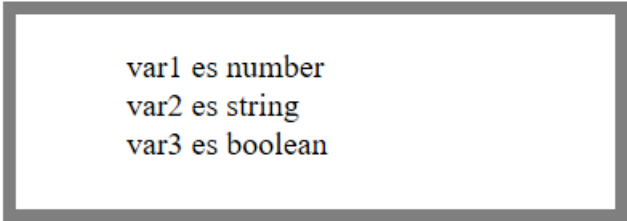
```
<HTML>
<HEAD><TITLE>Variables</TITLE></HEAD>
<BODY>
<SCRIPT LANGUAGE="JavaScript">
<!-- // Inicia Programa

    var var1 = 20;
    var var2 = "Alianza Lima";
    var var3 = true;

    document.write( "var1 es " + typeof var1 );
    document.write( "<BR>var2 es " + typeof var2 );
    document.write( "<BR>var3 es " + typeof var3 );

// Fin de Programa -->
</SCRIPT>
</BODY>
</HTML>
```

El resultado es:



```
var1 es number
var2 es string
var3 es boolean
```

# OPERADORES

## Operadores Aritméticos

Operador	Operación	Ejemplo
+	Suma	c = a + b;
-	Resta	c = a - b;
*	Multiplicación	c = a * b;
/	División	c = a / b;
%	Resto	c = a % b;
++	Incremento	a++; b = a++; c = ++b;
--	Decremento	a--; b = a--; c = --b;

### Ejemplo 5

Archivo: ejm05.html

```
<HTML>
<HEAD><TITLE>Operadores Aritméticos</TITLE></HEAD>
<BODY>
<SCRIPT LANGUAGE="JavaScript">

    var a = 40, b = 15;
    document.write("a = ", a, "<BR>");
    document.write("b = ", b, "<BR>");
    document.write(a, " + ", b, " = ", a+b, "<BR/>");
    document.write(a, " - ", b, " = ", a-b, "<BR/>");
    document.write(a, " * ", b, " = ", a*b, "<BR/>");
    document.write(a, " / ", b, " = ", a/b, "<BR/>");
    document.write(a, " % ", b, " = ", a%b, "<BR/>");

</SCRIPT>
</BODY>
</HTML>
```

## Operadores de Asignación

Operador	Operación	Ejemplo	Equivalente
=	Asignación	c = a + b;	
+=	Suma-Asignación	c += a;	c = c + a;
-=	Resta-Asignación	c -= a;	c = c - a;
*=	Multiplicación-Asignación	c *= a;	c = c * a;
/=	División-Asignación	c /= a;	c = c / a;
%=	Resto-Asignación	c %= a;	c = c % a;

## Operadores de Comparación

Operador	Operación	Ejemplo
==	Igual que	c = a == b;
!=	Diferente que	c = a != b;
>	Mayor que	c = a > b;
>=	Mayor o igual que	c = a >= b;
<	Menor que	c = a < b;
<=	Menor o igual que	c = a <= b;

Dependiendo del valor de a y b, c puede tomar valor **true** o **false**.

Se utilizan principalmente para la toma de decisiones.



## Operadores Lógicos

Operador	Operación	Ejemplo
&&	Y Lógico (And)	c = a && b;
	O Lógico (Or)	c = a    b;
!	Negación (Not)	c = !a;

Dependiendo del valor de a y b, c puede tomar valor **true** o **false**.

Se utilizan principalmente para la toma de decisiones.

## OTROS OPERADORES

### Condicional ( ? : )

Formato:

```
Condición?Expresión1:Sxpresión2
```

Ejemplo:

```
var a = 20;  
var b = 15;  
c = (a>b)?a:b;
```

### Concatenación ( + )

Formato:

```
Cadena1 + Cadena2
```

Ejemplo:

```
ElMejor = "Alianza" + " " + "Campeón";
```

## MENSAJES AL USUARIO

Formato 1:

```
document.write("mensaje")
```

Formato 2:

```
alert("mensaje")
```

### Ejemplo 6

Archivo: ejm06.html

```
<HTML>
<HEAD>
  <meta charset="utf-8">
  <TITLE>Mensajes JavaScript</TITLE>
  <SCRIPT LANGUAGE=javascript TYPE="text/javascript">
    alert("¡Bienvenidos a JavaScript!")
  </SCRIPT>
</HEAD>
<BODY BGCOLOR="blue" TEXT="white">
<H1>
  <SCRIPT LANGUAGE=javascript TYPE="text/javascript">
    document.write("¡Hola Mundo!")
  </SCRIPT>
</H1>
</BODY>
</HTML>
```

## LECTURA DE DATOS

Sintaxis:

```
prompt( mensaje [, EntradaPredeterminada ] )
```

Muestra un cuadro de diálogo para que el usuario pueda ingresar un dato.

Retorna el dato ingresado.

### Ejemplo 7

Verificar ingreso de datos.

Archivo: ejm07.html

```
<SCRIPT LANGUAGE="JavaScript">

var nombre

nombre = prompt("Ingrese su nombre","Digite aquí su nombre");

if (typeof nombre == "string"){
    document.write("<H3>Hola ", nombre, ", bienvenido a JavaScript</H3>" );
} else{
    document.write("<H3>No has ingresado tú nombre</H3>");
}

</SCRIPT>
```

# CONDICIONALES

## Operador ternario

Sintaxis

```
(condición) ? sentenciaV : sentenciaF
```

## Estructura if

```
if (condición) {  
    sentenciasV  
} else {  
    sentenciasF  
}
```

## Ejemplo 8

Determinar si un número es par.

Archivo: ejm08.html

```
<SCRIPT LANGUAGE="JavaScript">  
    var n = prompt("ingrese un numero entero");  
    var r = (n%2==0) ? "Es Par" : "Es Impar";  
    document.write( n + " " + r);  
</SCRIPT>
```

## Ejemplo 9

Mayor de tres números.

Archivo: ejm09.html

```
<SCRIPT LANGUAGE = "JavaScript">
// Variables
var n1, n2, n3, mayor;

// Datos
n1 = prompt( "N1", "" );
n2 = prompt( "N2", "" );
n3 = prompt( "n3", "" );

// Proceso
mayor = n1;
if (n2 > mayor){
    mayor = n2
}
if (n3 > mayor){
    mayor = n3
}

// Reporte
document.write("<H3> El mayor valor de: ");
document.write(n1, " , ", n2, " y ", n3 + " es:</H3>");
document.write("<H1>" + mayor + "</H1>");
</SCRIPT>
```

# BUCLES

## Bucle: For

### Sintaxis:

```
for(exp_inicialización; exp_condición; exp_bucle){  
    instrucciones  
}
```

### Ejemplo

```
for( k=1; k <= 100; k++ ) {  
    document.write( k ) ;  
}
```

### Ejemplo 10

Calcular el factorial de un número.

Archivo: ejm10.html

```
<script>  
    // Variables  
    var n, f=1, i;  
  
    // Dato  
    n = prompt( "Factorial de ?", "" );  
  
    // Proceso  
    for( i=2; i <= n; i++ ){  
        f *= i;  
    }  
  
    // Reporte  
    document.write( "factorial de " + n + " es " + f );  
</script>
```

## Bucle: while

### Sintaxis:

```
while ( condición ) {  
    Instrucciones  
}
```

### Ejemplo 11

Calcular el MCD de dos números.

Archivo: ejm11.html

```
<SCRIPT LANGUAGE = "JavaScript">  
  // Variables  
  var n1, n2, mcd, n1a, n2a;  
  // Datos  
  n1 = prompt( "Un numero entero:", "" );  
  n2 = prompt( "Otro número entero;", "" );  
  // Proceso  
  n1a = n1;  
  n2a = n2;  
  while (n1 != n2) {  
    if (n1 > n2) {  
      n1 -= n2;  
    }  
    else {  
      n2 -= n1;  
    }  
  }  
  mcd = n1;  
  // Reporte  
  document.write( "<p>N1=" + n1a + "</p>" );  
  document.write( "<p>N2=" + n2a + "</p>" );  
  document.write( "<p>MCD=" + mcd + "</p>" );  
</SCRIPT>
```

## Bucle: Do While

### Sintaxis:

```
do {  
    Instrucciones  
} while ( condición );
```

### Ejemplo 12

Mostrar los números pares.

Archivo: ejm12.html

```
<SCRIPT LANGUAGE = "JavaScript">  
var k=0;  
document.write("<H3>Números pares entre 1 y ...</H3>");  
do {  
    k=k+50;  
    for (i=k+1-50 ; i<=k; i++){  
        if (i%2==0) { document.write(i + ","); }  
        if (i%10==0) { document.write("<BR>"); }  
    }  
} while (confirm("¿Deseas ver los 50 próximos pares?"));  
document.write("<H3>Fin</H3>");  
</SCRIPT>
```



## Bucle: For In

### Sintaxis:

```
for( var propiedad in objeto ){  
    Instrucciones  
}
```

### Ejemplo 13

Mostrar las propiedades del Objeto **document**.

Archivo: ejm13.html

```
<SCRIPT LANGUAGE = "JavaScript">  
    var k = 0;  
    document.write("<H3>Propiedades del objeto document ...</H3>");  
    for ( propiedad in document ){  
        k++;  
        document.write(k,".- document." + propiedad + " = "  
            + document[propiedad] + "<BR>");  
    }  
    document.write("<BR>Fin")  
</SCRIPT>
```

## Sentencias break y continue

### Sentencia break

Finaliza la ejecución de un bucle.

### Sentencia continue

Lleva el control a la expresión del bucle para ser nuevamente evaluada, ignorando las instrucciones que se encuentran posteriores a la instrucción continue.

### Ejemplo 14

Números Pares.

Archivo: ejm14.html

```
<SCRIPT LANGUAGE = "JavaScript">
  var n, k = 1, cont = 0;
  n = prompt( "Cuantos numeros pares?", "" );
  while (true){
    if ( (k % 2) == 0 ) {
      document.write( k + "," );
      cont++;
      if ( (k % 5) == 0 ) { document.write( "<BR>" ); }
    }
    k++;
    if (cont < n) { continue; }
    if (cont == n) { break; }
    document.write( "Nunca se ejecuta" );
  }
  document.write("<H3>Fin</H3>");
</SCRIPT>
```

## Instrucción: WITH

Se utiliza para evitar escribir la referencia a un objeto cuando se está accediendo a las propiedades o métodos de dicho objeto.

### Sintaxis:

```
with ( objeto ) {  
    Instrucciones  
}
```

## Ejemplo 15

Propiedades del objeto **document**.

Archivo: ejm15.html

```
<HTML>  
<HEAD>  
    <TITLE>Estructuras de Control</TITLE>  
</HEAD>  
<BODY>  
<SCRIPT LANGUAGE = "JavaScript">  
    with ( document ) {  
        write("<H3>Utilizando la sentencia \"WITH\"</H3>");  
        write("título: \"\" + title + \"\"<BR><BR>");  
        write("color de fondo: " + bgColor + "<BR>");  
        write("color de primer plano: " + fgColor + "<BR>");  
        write("color de los enlaces visitados: " + vlinkColor + "<BR>");  
        write("color de los enlaces no visitados: " + alinkColor + "<BR>");  
        write("<H3>Fin</H3>");  
    }  
</SCRIPT>  
</BODY>  
</HTML>
```

## Instrucción: Switch

Se utiliza para comparar un dato entre un conjunto de posibles valores.

### Sintaxis:

```
switch ( DatoComparar ) {  
  case valor1:  
    Instrucciones;  
    break;  
  case valor2:  
    Instrucciones;  
    break;  
  -----  
  -----  
  case valorn:  
    Instrucciones;  
    break;  
  default:  
    Instrucciones  
}
```

## Ejemplo 16

Identificación de usuario.

Archivo: ejm16.html

```
<SCRIPT LANGUAGE=javascript TYPE="text/javascript">
document.write( "Ingreso al Sistema<BR>");
usuario = prompt("Por favor, identifique: ", "invitado");
switch (usuario.toUpperCase()) {
  case "ADMINISTRADOR":
    document.write("Bienvenido Administrador...<BR>");
    break;
  case "OPERADOR":
    document.write("Bienvenido Operador...<BR>");
    break;
  case "INVITADO":
    document.write("Bienvenido Invitado...<BR>");
    break;
  default:
    document.write("No estás autorizado...<BR>");
    break;
}
document.write("Fin del proceso.")
</SCRIPT>
```

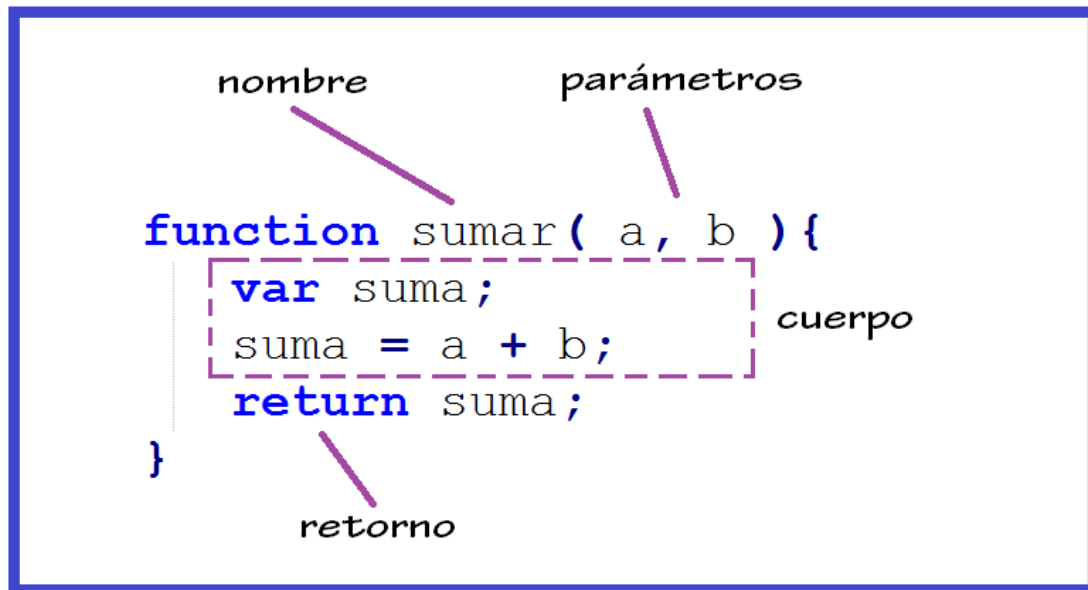
## EJERCICIOS PROPUESTOS

1. Desarrollar un programa para mostrar la tabla de multiplicar de **N**, el programa debe solicitar el valor de N.
2. Realizar un programa para calcular el factorial de un número.
3. Realizar un programa para imprimir N términos de la serie de Fibonacci, el programa debe solicitar el valor de **N**, y debe validar a que sea mayor que 2.
4. Desarrollar un programa que solicite un número entero **N**, y luego determine si se trata de un número primo.
5. Realizar un programa para calcular el importe de una venta.

## Capítulo 2

# APLICANDO JAVASCRIPT

## INTRODUCCIÓN



Las funciones se encargan de agrupar una serie de instrucciones dentro de un mismo bloque para ejecutarlas cuando y cuantas veces queramos.

Las funciones se pueden incluir en cualquier parte del documento HTML en archivos de extensión **.js**.

### Sintaxis

```
function nombre( arg1, arg2, arg3, ... )  
{  
    // Cuerpo de la función  
}
```

## Ejemplo 17

```
<HTML>
  <HEAD>
    <TITLE> FUNCIONES </TITLE>
    <SCRIPT LANGUAGE="JavaScript">
      function cuadrado( numero )
      {
        return numero * numero;
      }
    </SCRIPT>
  </HEAD>
  <BODY>
    <FORM NAME="Form1">
      Numero: <INPUT TYPE="text" NAME="numero"> <BR/>
      Resultado: <INPUT TYPE="text" NAME="resultado" disabled> <BR/>
      <INPUT TYPE="button" value="Calcular"
      onclick="Form1.resultado.value=cuadrado(Form1.numero.value)">
    </FORM>
  </BODY>
</HTML>
```



## FUNCIONES PREDEFINIDAS

### Función: eval()

Interpreta una cadena y la ejecuta como si se tratase de una porción de código JavaScript.

Sintaxis:

```
eval(cadena)
```

### Ejemplo 18

```
<HTML>
  <HEAD>
    <META charset="utf-8">
    <TITLE> FUNCIONES </TITLE>
  </HEAD>
  <BODY>
    <FORM NAME="Form1">
      Expresión: <INPUT TYPE="text" NAME="dato" SIZE="50"><BR/>
      Resultado: <INPUT TYPE="text" NAME="rpta" SIZE="50" disabled><BR/>
      <INPUT TYPE="button" value="Calcular"
      onclick="Form1.rpta.value = eval(Form1.dato.value)">
    </FORM>
  </BODY>
</HTML>
```

A continuación, tienes el resultado de su ejecución:



The screenshot shows a web form with two text input fields and a button. The first field is labeled 'Expresión:' and contains the text '8+7\*(8-5)'. The second field is labeled 'Resultado:' and contains the text '29'. Below the fields is a button labeled 'Calcular'.

## Función: parseInt()

Permite transformar cualquier cadena numérica en un número entero.

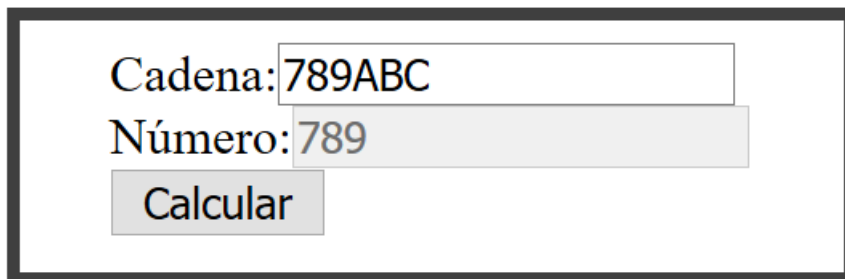
Sintaxis:

```
parseInt(cadena [, base])
```

### Ejemplo 19

```
<HTML>
  <HEAD>
    <META charset="utf-8">
    <TITLE> FUNCIONES </TITLE>
  </HEAD>
  <BODY>
    <FORM NAME="Form1">
      Cadena: <INPUT TYPE="text" NAME="cad"> <BR/>
      Número: <INPUT TYPE="text" NAME="num" disabled> <BR/>
      <INPUT TYPE="button" value="Calcular"
        onclick="Form1.num.value=parseInt(Form1.cad.value)">
    </FORM>
  </BODY>
</HTML>
```

A continuación, se tiene el resultado de su ejecución:



Cadena: 789ABC  
Número: 789  
Calcular

## Función: parseFloat()

Convertir una cadena en un número real.

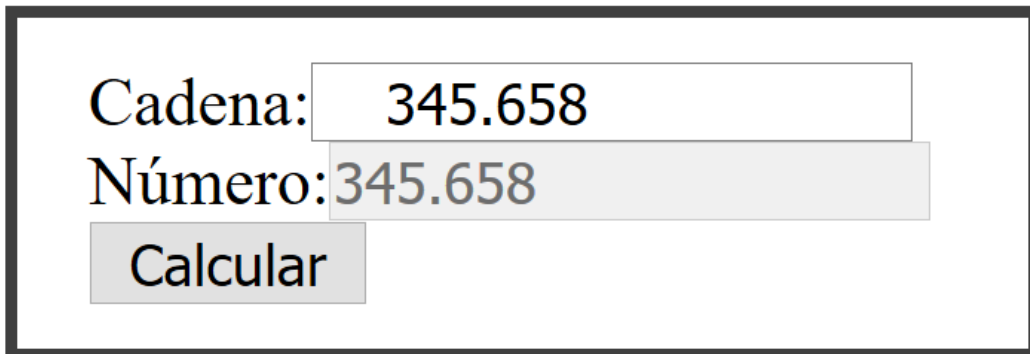
**Sintaxis:**

```
parseFloat(cadena)
```

### Ejemplo 20

```
<HTML>
  <HEAD>
    <META charset="utf-8">
    <TITLE> FUNCIONES </TITLE>
  </HEAD>
  <BODY>
    <FORM NAME="Form1">
      Cadena: <INPUT TYPE="text" NAME="cad"><BR/>
      Número: <INPUT TYPE="text" NAME="num" disabled><BR/>
      <INPUT TYPE="button" value="Calcular"
        onclick="Form1.num.value=parseFloat(Form1.cad.value)">
    </FORM>
  </BODY>
</HTML>
```

El resultado es el siguiente:



The screenshot shows a web form with a black border. It contains three elements: a label 'Cadena:' followed by a text input field containing '345.658'; a label 'Número:' followed by a disabled text input field containing '345.658'; and a button labeled 'Calcular'.

## Función: isNaN()

isNaN intenta convertir el parámetro pasado a un número. Si el parámetro no se puede convertir, devuelve **true**; en caso contrario, devuelve **false**.

### Sintaxis:

```
isNaN(valor)
```

### Ejemplo 21

```
<HTML>
  <HEAD>
    <META charset="utf-8">
    <TITLE> Funciones </TITLE>
    <SCRIPT LANGUAGE="JavaScript">
      function isInt( cadena )
      {
        return !isNaN( parseInt(cadena) );
      }
      function verificar(){
        Form1.valor.value = parseInt(Form1.cad.value);
        Form1.res.value = isInt(Form1.cad.value);
      }
    </SCRIPT>
  </HEAD>
  <BODY>
    <FORM NAME="Form1">
      Entero:<INPUT TYPE="text" NAME="cad"><BR/>
      Valor:<INPUT TYPE="text" NAME="valor" disabled><BR/>
      Resultado:<INPUT TYPE="text" NAME="res" disabled><BR/>
      <INPUT TYPE="button" value="Verificar"
        onclick="verificar()">
    </FORM>
  </BODY>
</HTML>
```

El resultado es el siguiente:

Entero:

345ABC

Valor:

345

Resultado:

true

Verificar

## Función: isFinite()

Comprueba si un valor es finito.

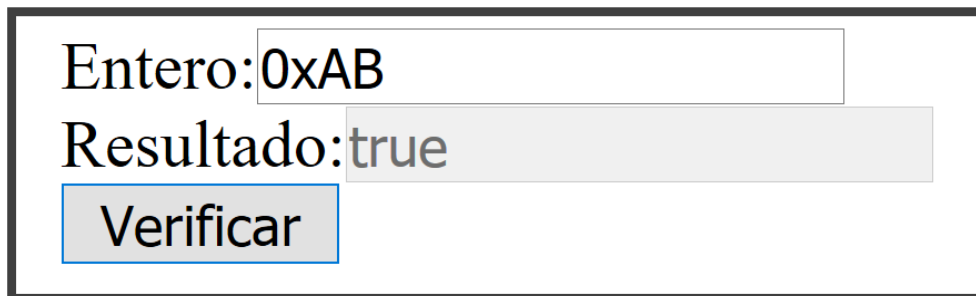
**Sintaxis:**

```
isFinite(numero)
```

### Ejemplo 22

```
<HTML>
  <HEAD>
    <META charset="utf-8">
    <TITLE> Funciones </TITLE>
  </HEAD>
  <BODY>
    <FORM NAME="Form1">
      Entero: <INPUT TYPE="text" NAME="cad"><BR/>
      Resultado: <INPUT TYPE="text" NAME="res" disabled><BR/>
      <INPUT TYPE="button" value="Verificar"
        onclick="Form1.res.value=isFinite(Form1.cad.value)">
    </FORM>
  </BODY>
</HTML>
```

A continuación tenemos el resultado:



The screenshot shows a web form with a black border. It contains three elements: a text input field with the value '0xAB', a disabled text input field with the value 'true', and a button labeled 'Verificar'.

Entero: 0xAB

Resultado: true

Verificar

## Paso de Varios Argumentos a una Función

### Ejemplo 23

```
<HTML>
  <HEAD>
    <META charset="utf-8">
    <TITLE> Funciones </TITLE>
    <SCRIPT LANGUAGE="JavaScript">
      function sumar()
      {
        suma = 0;
        for(i=0; i < arguments.length; i++)
        {
          suma += arguments[i];
        }
        return suma;
      }
    </SCRIPT>
  </HEAD>
  <BODY>
    <H1>SUMA</H1>
    <FORM NAME="Form1">
      n1:<INPUT TYPE="text" NAME="n1"><BR/>
      n2:<INPUT TYPE="text" NAME="n2"><BR/>
      n3:<INPUT TYPE="text" NAME="n3"><BR/>
      Suma:<INPUT TYPE="text" NAME="suma" disabled><BR>
      <INPUT TYPE="button" value="Procesar"
      onclick="Form1.suma.value=sumar(Number(Form1.n1.value),
      Number(Form1.n2.value),Number(Form1.n3.value))">
    </FORM>
  </BODY>
</HTML>
```

El resultado es el siguiente:

<b>SUMA</b>	
n1:	<input type="text" value="6"/>
n2:	<input type="text" value="2"/>
n3:	<input type="text" value="3"/>
Suma:	<input type="text" value="11"/>
<input type="button" value="Procesar"/>	



## OBJETOS PREDEFINIDOS

### Objeto: Date

El objeto Date se utiliza para trabajar con fechas y horas.

Los objetos Date son creados de la siguiente manera:

```
var d1 = new Date();  
var d2 = new Date(milliseconds);  
var d3 = new Date(dateString);  
var d4 = new Date(year, month, day, hours, minutes, seconds, milliseconds);
```

Un objeto Date contiene un número que representa un instante de tiempo concreto en milisegundos. Si el valor de un argumento es mayor que su intervalo o es un número negativo, los demás valores almacenados se modifican consecuentemente. Por ejemplo, si especifica 150 segundos, JavaScript vuelve a definir ese número como dos minutos y 30 segundos.

Si el número es **NaN**, el objeto no representa un instante específico de tiempo. Si no pasa ningún parámetro al objeto Date, se inicializa con la hora actual (UTC). Se debe asignar un valor al objeto antes de poder usarlo.

El intervalo de fechas que se puede representar en un objeto Date abarca el período comprendido, aproximadamente, entre los 285.616 años antes y después del 1 de enero de 1970.

Tiene una serie de métodos, entre los cuales tenemos:

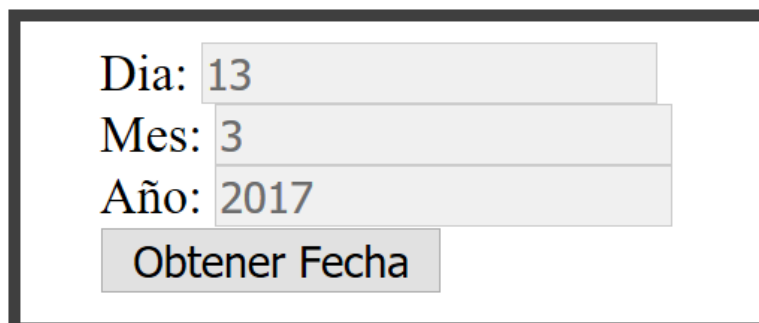
Method	Description
getDate()	Retorna el día del mes (1-31).
getDay()	Retorna el día de la semana (0-6).
getFullYear()	Retorna el año.
getHours()	Retorna las horas (0-23).
getMilliseconds()	Retorna los milisegundos (0-999).
getMinutes()	Retorna los minutos (0-59).
getMonth()	Retorna el mes (0-11).

getSeconds()	Retorna los segundos (0-59).
getTime()	Retorna el número de milisegundos desde la medianoche del 1 de enero de 1970 y una fecha especificada.
now()	Retorna el número de milisegundos desde la medianoche 1 de enero de 1970.
parse()	Analiza una cadena de fecha y devuelve el número de milisegundos desde el 1 de enero de 1970.
setDate()	Establece el día del mes.
setFullYear()	Establece el año.
setHours()	Establece las horas.
setMilliseconds()	Establece los milisegundos.
setMinutes()	Establece los minutos.
setMonth()	Establece los meses.
setSeconds()	Establece los segundos.
setTime()	Establece una fecha en un número especificado de milisegundos.
toISOString()	Retorna la fecha en formato ISO.
toJSON()	Retorna la fecha en formato JSON.
toString()	Convierte el objeto fecha en un String.
valueOf()	Retorna el valor primitivo del objeto Date.

## Ejemplo 24

```
<HTML>
  <HEAD>
    <META charset="utf-8">
    <TITLE>Objeto Date</TITLE>
    <SCRIPT LANGUAGE="JavaScript">
      function getFecha()
      {
        fecha = new Date();
        Form1.dia.value = fecha.getDate();
        Form1.mes.value = fecha.getMonth() + 1;
        Form1.anno.value = fecha.getFullYear();
      }
    </SCRIPT>
  </HEAD>
  <BODY>
    <FORM NAME="Form1">
      <H1>FECHA ACTUAL</H1>
      Dia: <INPUT TYPE="text" NAME="dia" disabled><BR/>
      Mes: <INPUT TYPE="text" NAME="mes" disabled><BR/>
      Año: <INPUT TYPE="text" NAME="anno" disabled><BR/>
      <INPUT TYPE="button" value="Obtener Fecha"
      onclick="getFecha()">
    </FORM>
  </BODY>
</HTML>
```

El resultado de su ejecución:



The screenshot shows the rendered HTML page. It features a heading 'FECHA ACTUAL' followed by three disabled text input fields. The first field is labeled 'Dia:' and contains the value '13'. The second field is labeled 'Mes:' and contains the value '3'. The third field is labeled 'Año:' and contains the value '2017'. Below these fields is a button with the text 'Obtener Fecha'.

## Objeto: Math

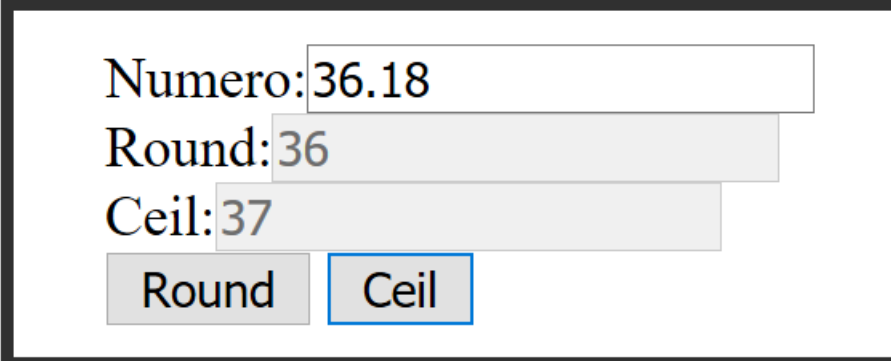
Objeto que proporciona funciones y constantes matemáticas básicas.

El objeto Math no se puede crear usando el operador new y produce un error si intenta hacerlo. El motor de scripting lo crea al cargarse. Todos sus métodos y propiedades están disponibles para el script en todo momento.

### Ejemplo 25

```
<HTML>
  <HEAD>
    <META charset="utf-8">
    <TITLE>Objeto Math</TITLE>
  </HEAD>
  <BODY>
    <FORM NAME="Form1">
      Numero:<INPUT TYPE="text" NAME="num"><BR/>
      Round:<INPUT TYPE="text" NAME="res1" disabled><BR/>
      Ceil:<INPUT TYPE="text" NAME="res2" disabled><BR/>
      <INPUT TYPE="button" value="Round"
      onclick="Form1.res1.value=Math.round(Form1.num.value)">
      <INPUT TYPE="button" value="Ceil"
      onclick="Form1.res2.value=Math.ceil(parseFloat(Form1.num.value))">
    </FORM>
  </BODY>
</HTML>
```

El resultado es el siguiente:



The screenshot shows a web form with the following elements:

- A text input field labeled "Numero:" containing the value "36.18".
- A disabled text input field labeled "Round:" containing the value "36".
- A disabled text input field labeled "Ceil:" containing the value "37".
- Two buttons: "Round" and "Ceil". The "Ceil" button is highlighted with a blue border.