

DESARROLLO WEB

2017-I



SEMANA 02

APLICANDO JAVASCRIPT

Eric Gustavo Coronel Castillo

ecoronel@uch.edu.pe - gcoronelc.blogspot.com

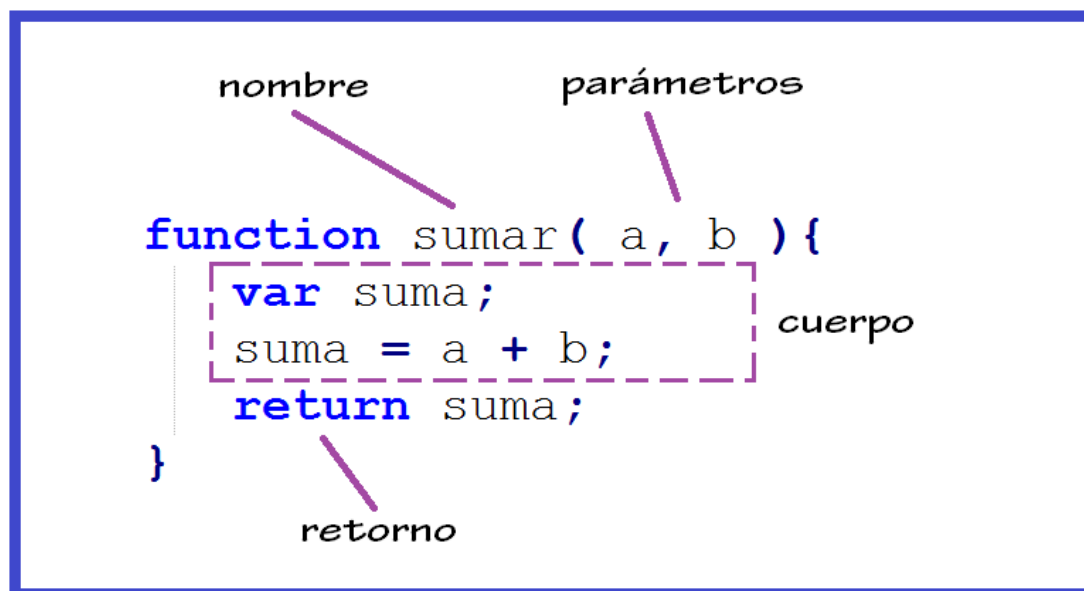
APLICANDO JAVASCRIPT

El objetivo de esta semana es que apliques funciones mediante el uso de librerías y la Orientación a Objetos.

ÍNDICE

INTRODUCCIÓN	4
FUNCIONES PREDEFINIDAS	6
FUNCIÓN: EVAL()	6
FUNCIÓN: PARSEINT()	7
FUNCIÓN: PARSEFLOAT()	8
FUNCIÓN: ISNAN()	9
FUNCIÓN: ISFINITE()	11
PASO DE VARIOS ARGUMENTOS A UNA FUNCIÓN	12
OBJETOS PREDEFINIDOS.....	14
OBJETO: DATE.....	14
OBJETO: MATH	17

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 1

```
<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 2

```
<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, labeled 'Expresión:', contains the text '8+7*(8-5)'. The second field, labeled 'Resultado:', contains the text '29'. Below the fields is a button labeled 'Calcular'.

Función: parseInt()

Permite transformar cualquier cadena numérica en un número entero.

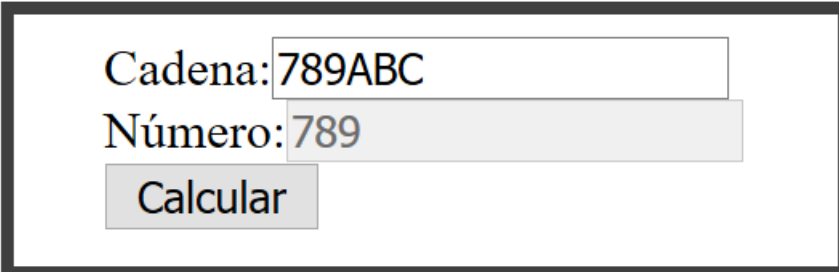
Sintaxis:

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

Ejemplo 3

```
<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:



The screenshot shows a web form with two text input fields and a button. The first field, labeled 'Cadena:', contains the text '789ABC'. The second field, labeled 'Número:', contains the text '789' and is disabled. Below the fields is a button labeled 'Calcular'.

Función: parseFloat()

Convertir una cadena en un número real.

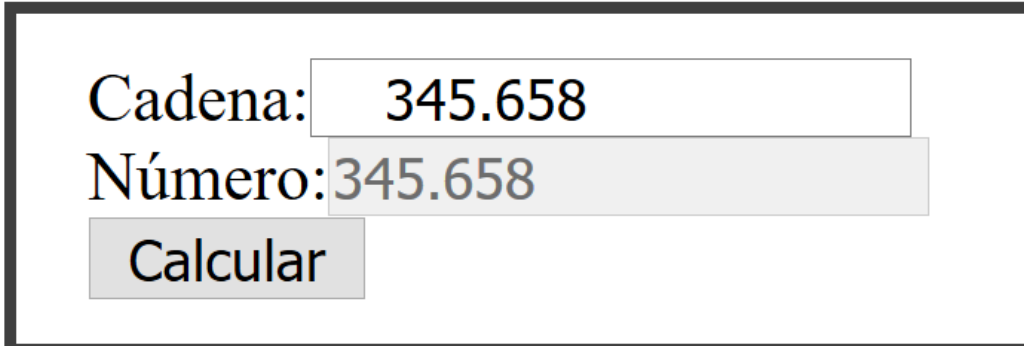
Sintaxis:

```
parseFloat(cadena)
```

Ejemplo 4

```
<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 dark border. It contains two text input fields. The first field is labeled 'Cadena:' and contains the value '345.658'. The second field is labeled 'Número:' and contains the value '345.658'. Below these fields is 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 5

```
<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.

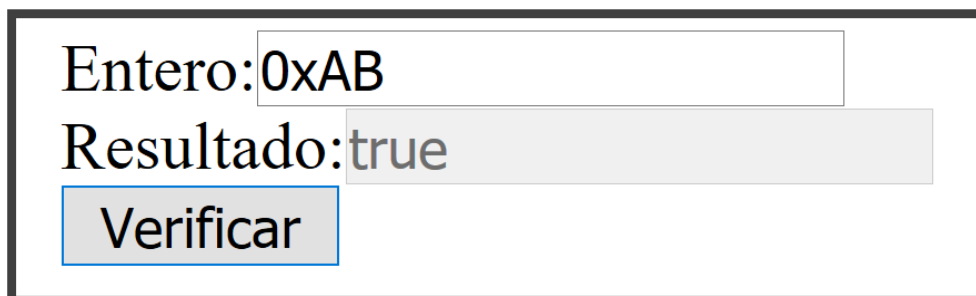
Sintaxis:

```
isFinite(numero)
```

Ejemplo 6

```
<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 blue button labeled 'Verificar'.

Entero: 0xAB

Resultado: true

Verificar

Paso de Varios Argumentos a una Función

Ejemplo 7

```
<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:

SUMA

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:

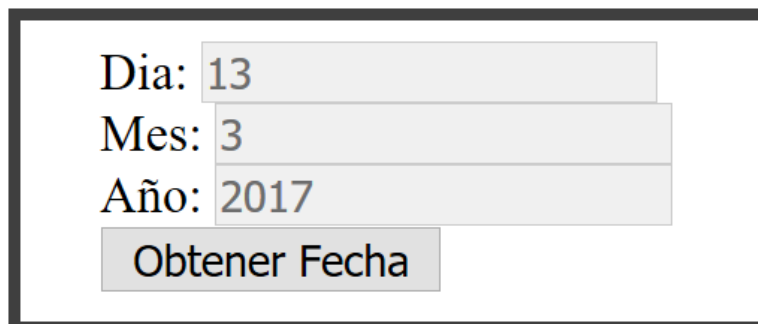
Method	Description
getDate()	Retorna el día del mes (1-31).
getDay()	Retorna el día de la semana (0-6).
getFullYear()	Retorna el año.
getHours()	Retorna las horas (0-23).
getMilliseconds()	Retorna los milisegundos (0-999).
getMinutes()	Retorna los minutos (0-59).

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 8

```
<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:



Dia: 13
Mes: 3
Año: 2017
Obtener Fecha

Objeto: Math

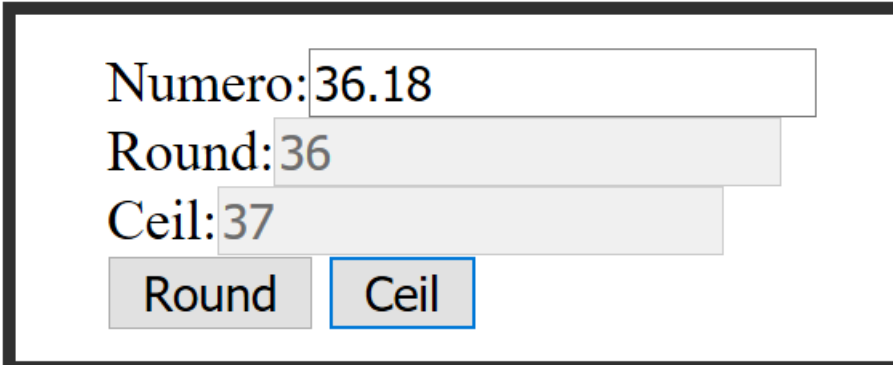
Objeto que proporciona funciones y constantes matemáticas básicas.

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 9

```
<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 three input fields and two buttons. The first input field, labeled 'Numero:', contains the value '36.18'. The second input field, labeled 'Round:', contains the value '36'. The third input field, labeled 'Ceil:', contains the value '37'. Below the input fields are two buttons: 'Round' and 'Ceil'. The 'Ceil' button is highlighted with a blue border.