APRENDERAPROGRAMAR.COM

CONVERTIR STRING A NUMBER JAVASCRIPT. REDONDEAR. TOFIXED, ISNAN, TOPRECISION, VALUEOF. PARSEINT Y PARSEFLOAT (CU01156E)

Sección: Cursos
Categoría: Tutorial básico del programador web: JavaScript desde cero

Fecha revisión: 2029

Resumen: Entrega nº56 del Tutorial básico "JavaScript desde cero".

Autor: César Krall

FUNCIONES PARA TRABAJAR CON NÚMEROS

JavaScript proporciona una serie de funciones predefinidas y palabras clave o constantes para trabajar con números. Entre ellas podemos citar isNan, Number(n), toFixed(n), toExponential(n), toPrecision(n), valueOf(), toString().

El trabajo con números en JavaScript se puede abordar desde distintas perspectivas.

En primer lugar, existen dos variables predefinidas globales que ya hemos mencionado:

a) NaN (Not-a-Number) que se usa para representar un valor numérico no válido (p.ej. 0/0).

b) Infinity que se usa para representar un valor numérico positivo infinito, o su variante -Infinity, que representa un valor numérico negativo infinito.

El valor NaN es un valor peculiar en JavaScript desde el momento en que no permite su comparación con el operador ==. Por ejemplo, var x = NaN; var y = NaN; alert (x==y); devuelve false a pesar de que ambas variables contienen Nan. Para evaluar si una variable contiene Nan ha de usarse la expresión x != x, que devolverá true únicamente si x contiene NaN. Dado que la aplicación de estos operadores es un tanto confusa con variables que pueden contener NaN, se recomienda usar la función isNaN que comentaremos un poco más adelante, y que permite evaluar si un valor es NaN.

Una expresión como 1/Infinity se evalúa a cero.

PROPIEDADES DEL OBJETO NUMBER

JavaScript, al igual que dispone de un objeto predefinido RegExp para trabajar con expresiones regulares, dispone de un objeto predefinido denominado Number que permite trabajar con números. Este objeto predefinido tiene propiedades y métodos. Entre sus propiedades podemos citar las siguientes:

PROPIEDAD	REPRESENTA	EJEMPLOS aprenderaprogramar.com
POSITIVE_INFINITY	Infinito positivo (igual que Infinity)	var num = Number.POSITIVE_INFINITY; //num toma valor Infinity
NEGATIVE_INFINITY	Infinito negativo (igual que -Infinity)	var num = Number.NEGATIVE_INFINITY; //num toma valor -Infinity

MAX_VALUE	Representa el valor numérico más grande con el que puede trabajar JavaScript	var num = Number.MAX_VALUE; // num vale 1.7976931348623157e+308 u otra cifra igualmente gigantesca, dependiendo del sistema
MIN_VALUE	Representa el valor numérico más pequeño con el que puede trabajar JavaScript.	var num = Number.MIN_VALUE; // num vale num vale 5e-324 u otra cifra también aproximadamente cero, dependiendo del sistema
NaN	Representa el valor NaN	var num = Number.NaN; // num vale NaN

Las propiedades de Number son de sólo lectura, es decir, no podemos establecer valores distintos de los predefinidos para ellas.

FUNCIONES DEL OBJETO NUMBER

FUNCIÓN	UTILIDAD	EJEMPLOS aprenderaprogramar.com

toFixed(n)	Devuelve un String con el número o variable sobre el que se invoca el método con tantos decimales como indique el parámetro n. Si n es cero o vacío, redondea al entero más próximo. Si es x.50 devuelve el entero inmediato superior si x es positivo o inferior si x es negativo. No genera notación exponencial.	var num = 3.1416;
		alert('num vale ' + num.toFixed(2));
		// num vale "3.14 "

isNaN(x)	Evalúa a true si x es un valor NaN o a false en caso contrario.	alert('¿0/0 vale NaN?: ' + isNaN(0/0)); // true
toPrecision(n)	Devuelve un String con el número o variable sobre el que se invoca el método con un número de dígitos significativos especificado por el número de decimales n. Si n es vacío devuelve el número sin modificar. No admite n cero. Si el número es demasiado grande para representar su parte entera, genera notación exponencial.	var num = (3.1416).toPrecision(2); // num vale 3.1, hay dos dígitos significativos.
valueOf(x)	Este método devuelve la representación como tipo primitivo de un objeto. En el caso de un Number, hace lo mismo que el método toString().	alert (A1.valueOf()); //Mismo resultado que alert(A1);
toString()	Devuelve la representación como String de un objeto Number.	alert (A1.toString()); //Mismo resultado que alert(A1);

DIFERENCIAR EL OBJETO NUMBER DE OBJETOS NUMBER

Debemos diferenciar entre el objeto predefinido Number, y los objetos Number que podamos crear nosotros como programadores (que también dispondrán de métodos heredados del objeto Number). Para crear objetos Number usamos la sintaxis var num = new Number(valorNumerico);

NUMBER COMO FUNCIÓN PARA CONVERTIR OTROS TIPOS DE DATOS

Dado una variable de otro tipo a la que denominaremos unaVarOtroTipo, podemos usar Number como función para realizar la conversión de dicha variable a un valor numérico usando esta sintaxis:

var valorNumerico = Number(unaVarOtroTipo);

Donde unaVarOtroTipo puede ser un String, un objeto Date, o cualquier otro objeto del que se pueda realizar una representación numérica.

FUNCIONES GLOBALES PARSEINT Y PARSEFLOAT

Existen dos funciones globales útiles para convertir cadenas en valores numéricos: parseInt y parseFloat.

La sintaxis básica de parseFloat es la siguiente:

```
var numerica = parseFloat(cadena);
```

La función parseFloat toma la cadena y trata de retornar un valor numérico decimal. Si la conversión no se puede completar por aparecer algún carácter extraño, devuelve el valor numérico que pueda extraer hasta la aparición de dicho valor. Si la cadena no empieza con un valor válido para un número (es decir, un +, un – ó un número, un signo . ó una letra e indicadora de exponente) la función devuelve NaN.

Veámoslo con un ejemplo:

```
var cadena1 = '82.35 % de aprobados';
var cadena2 = '% de aprobados: 82.35';
alert('cadena1 a numero: ' + parseFloat(cadena1) + ' ; cadena2 a numero: ' + parseFloat(cadena2));
```

Este ejemplo da lugar a que se muestre por pantalla:

cadena1 a numero: 82.35 ; cadena2 a numero: NaN

En el primer caso, extrae el valor numérico hasta allí donde le es posible. En el segundo caso, al no comenzar la cadena con algo que pueda convertirse a numérico, devuelve NaN.

La sintaxis básica de parseInt es la siguiente:

```
var numericaInt = parseInt(cadena);
```

Hay otra forma de invocación consistente en pasar dos parámetros: parseInt (cadena, raíz), donde raíz es un parámetro opcional que por defecto es 10 e indica la base numérica empleada. Por defecto es la decimal (10), pero podría ser 8 (valor numérico octal) ó 16 (valor numérico hexadecimal).

El funcionamiento de parseInt es similar al de parseFloat con la diferencia de que parseInt extrae únicamente aquella parte de la cadena que pueda reconocerse como un entero. Así parseInt('15.67 metros') devuelve 15, parseInt ('.1 metros') devuelve NaN (no admite que un entero comience con un punto), y parseInt ('0.1 metros') devuelve 0.

EJERCICIO

Crea un script donde:

a) Se pida al usuario que introduzca un número superior a 10000000 y se muestre por pantalla ese número con tres dígitos significativos. Si el número introducido no cumple la condición, se volverá a solicitar que se introduzca.

b) Se pida al usuario que introzca un número con 4 decimales y se muestre por pantalla ese número redondeado a dos decimales. Si el número introducido no cumple la condición, se volverá a solicitar que se introduzca (para ello habrá que analizar que la cadena introducida cuente con cuatro dígitos después del carácter de punto decimal).

c) Se pida al usuario que introduzca una frase que comience por un número que puede ser entero o decimal. Para dicha frase, se devolverá el número entero que se pueda extraer, y el número decimal que se pueda extraer. Deberá analizarse la entrada con una expresión regular, de modo que si no cumple la condición, se vuelva a solicitar la introducción de la frase.

Para comprobar si tus respuestas y código son correctos puedes consultar en los foros aprenderaprogramar.com.

Próxima entrega: CU01157E

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente: http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=78&Itemid=206