APRENDERAPROGRAMAR.COM

TIPOS DE VARIABLES JAVASCRIPT: NUMÉRICAS O NUMBER (INT , INTEGER, SINGLE, DOUBLE, FLOAT…). RESULTADOS NAN E INFINITY (CU01113E)

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

Fecha revisión: 2029

Resumen: Entrega nº13 del Tutorial básico “JavaScript desde cero”.

Autor: César Krall

VARIABLES NUMÉRICAS EN JAVASCRIPT

A diferencia de muchos otros lenguajes de programación (como Java, C, C++, Visual Basic, etc.), JavaScript no diferencia distintos tipos de variables numéricas. Todos los números en JavaScript son tratados como un tipo numérico único, el tipo Number, que "agrupa" a los tipos int, integer, double, float, single, etc. que se utilizan en otros lenguajes.

VARIABLES NUMÉRICAS JAVASCRIPT

JavaScript permite representar cualquier número que sea necesario. Ten en cuenta que esto no significa que se puedan usar números "descomunales" porque un computador siempre ha de tener un límite. Pero el límite de JavaScript es del orden de $\pm 1.7976931348623157 \times 10^{308}$ para números grandes(esto son millones de trillones, un número tan grande que nunca tendremos problemas porque nunca vamos a tener que usar un número superior a esta cifra) y del orden de $\pm 5 \times 10^{-324}$ para números pequeños (esto supone que se puede trabajar con números con cientos de decimales, y en la práctica nunca vamos a necesitar tanta precisión, con lo cual nunca tendremos problemas por usar números muy pequeños o con gran número de decimales).

Para indicar que un número es negativo se precede del signo menos.

Los números decimales se escriben utilizando el punto (.) como separador. Si la parte entera de un número es el cero, se admite omitir el cero. Es decir, 0.55 y .55 son ambos admitidos.

También se admite la notación basada en indicar un número seguido de E y la potencia de 10 a la que se debe elevar. También se admite usar la e minúscula. Por ejemplo 1.2E2 ó 1.2e2 equivale a $1.2*10^2$ = 1.2*100 = 120. Esta notación permite escribir números muy grandes o muy pequeños sin tener que escribir todas las cifras. Por ejemplo 3.2345234565E20 equivale a 323452345650000000000 ó 3.2345234565e-10 equivale a 0.00000000032345234565

El uso de número decimales puede presentar en ocasiones problemas debido al redondeo decimal, como veremos. Este problema no es exclusivo de JavaScript, sino que es algo que afecta a numerosos lenguajes de programación.

Veamos un ejemplo. Escribe este código y guárdalo en un archivo de extensión html (puedes cambiar la ruta de la imagen si quieres):

```
<html>
<head>
<title>Curso JavaScript aprenderaprogramar.com</title> <meta charset="utf-8">
<script type="text/javascript">
function mostrarMensaje1() {
var bacterias = 3.55; var texto = 'bacterias en la probeta'; var numeroInfinito = Infinity;
alert('La variable bacterias vale: ' + bacterias);
bacterias = 3.55E5; alert('La variable bacterias vale (multiplicamos por 100000): ' + bacterias);
bacterias = 3.55E-5; alert('La variable bacterias ahora es un número muy pequeño: ' + bacterias);
alert('La variable bacterias ahora es (operación sin sentido): ' + bacterias*texto);
bacterias = 3.55e100000000000000000; alert('La variable bacterias ahora es demasiado grande: ' +
bacterias);
bacterias = 3.55E-100000000000000000; alert('La variable bacterias ahora es demasiado pequeña: ' +
bacterias);
alert('Un numero positivo dividido entre cero (indeterminación matemática) devuelve: ' + (4/0));
alert('Un numero negativo dividido entre cero (indeterminación matemática) devuelve: ' + (-4/0));
alert('Cero dividido entre cero devuelve: ' + (0/0));
alert('La variable numeroInfinito vale: ' + numeroInfinito);
var diezCentimos = .1; var veinteCentimos = .2; var treintaCentimos = .3;
alert('Esperamos 0.1 y lo obtenemos: ' + (veinteCentimos-diezCentimos));
alert('Esperamos 0.1 y no lo obtenemos: ' + (treintaCentimos-veinteCentimos));
}
</script>
</head>
<body>
<div>
<p>Aquí un párrafo de texto situado antes de la imagen, dentro de un div contenedor</p>
<img onclick="mostrarMensaje1()" src="http://i.imgur.com/afC0L.jpg" alt="Notepad++" title="Notepad++,
un útil editor de texto">
<p style="background-color:yellow;" onclick="mostrarMensaje2()">Aquí otro párrafo de texto. JavaScript es
un lenguaje utilizado para dotar de efectos dinámicos a las páginas web.
</p>
</div>
</body>
</html>
```

Visualiza el resultado y comprueba que la página web se muestra con normalidad y que JavaScript se ejecuta con normalidad cuando pulsas sobre la imagen.

El resultado esperado es que se muestre lo siguiente:

La variable bacterias vale: 3.55 (Aceptar)
La variable bacterias vale (multiplicamos por 100000): 355000 (Aceptar)
La variable bacterias ahora es un número muy pequeño: 0.0000355 (Aceptar)
La variable bacterias ahora es (operación sin sentido): NaN (Aceptar)
La variable bacterias ahora es demasiado grande: Infinity
La variable bacterias ahora es demasiado pequeña: 0
Un numero positivo dividido entre cero (indeterminación matemática) devuelve: Infinity
Un numero negativo dividido entre cero (indeterminación matemática) devuelve: -Infinity
Cero dividido entre cero devuelve: NaN
La variable numeroInfinito vale: Infinity
Esperamos 0.1 y lo obtenemos: 0.1
Esperamos 0.1 y no lo obtenemos: 0.09999999999999998

SIGNIFICADO DE NAN, INFINITY, -INFINITY Y PROBLEMAS DE REDONDEO

De este ejemplo debemos destacar lo siguiente:

a) Se cumplen los **conceptos básicos** explicados.
b) Una operación sin sentido devuelve el valor NaN (Not-a-Number), valor equivalente a "valor numérico no válido".
c) Un valor numérico positivo excesivamente grande (fuera de los límites admisibles) es representado como **Infinity**. Infinity puede considerarse como equivalente a "valor numérico positivo excesivamente grande o tendente a infinito". A una variable puede asignársele valor Infinity.
d) Un valor numérico excesivamente próximo a cero (fuera de los límites admisibles) es representado como cero.
e) Un valor numérico negativo excesivamente grande (fuera de los límites admisibles) es representado como **-Infinity**. -Infinity puede considerarse como equivalente a "valor numérico negativo excesivamente grande o tendente a menos infinito".
f) Algunas indeterminaciones matemáticas se representan como Infinity ó –Infinity y otras como **NaN** (Not-a-Number).
g) Pueden surgir problemas de redondeo al operar con decimales. Por ejemplo 0.3 – 0.2 sería de esperar que diera el mismo resultado que 0.2 – 0.1 y es posible que no dé el mismo resultado. ¿Por qué? Este problema se debe a que JavaScript (al igual que muchos otros lenguajes) no trabaja directamente con los decimales, sino que transforma esos decimales en una representación interna que al realizar operaciones puede dar lugar a resultados inesperados debido al **redondeo** de esa representación interna. Esta situación puede cambiar según el navegador o la versión de JavaScript que se utilice, pero para evitar problemas, se recomienda operar con números enteros y usar los decimales a la hora de visualizar por pantalla. En el ejemplo visto usaríamos 10, 20 y 30 y operaríamos con estos valores enteros. A la hora de mostrar por pantalla, haríamos la división entre 10.

EJERCICIO

Crea un código JavaScript para evitar el problema del redondeo que hemos tenido en el código anterior. Para ello define los decimales como enteros y realiza la operación para mostrar el decimal sólo en el momento de mostrar el resultado por pantalla.

Para comprobar si es correcta tu solución puedes consultar en los foros aprenderaprogramar.com.

Próxima entrega: CU01114E

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