APRENDERAPROGRAMAR.COM

TYPEOF JAVASCRIPT. SABER TIPO DE VARIABLE. GLOBAL Y LOCAL: AMBITO (SCOPE). VAR CAMBIA EL AMBITO. EJEMPLO CUENTA ATRÁS (CU01167E)

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

Fecha revisión: 2029

Resumen: Entrega nº67 del Tutorial básico "JavaScript desde cero".

Autor: César Krall

AMBITO (SCOPE) JAVASCRIPT

Cuando trabajamos con JavaScript podemos definir varios ámbitos: ámbito global (variables conocidas por todas las funciones y por el espacio global) y ámbitos locales (variables conocidas únicamente en determinadas partes del código).

OPERADOR TYPEOF

Existe un operador que nos va a resultar útil para determinar el tipo de una variable, o para determinar si esta variable existe en el ámbito en el que tratamos de utilizarla. La sintaxis para usar este operador es la siguiente:

```
typeof expresiónAEvaluar ó typeof (operandoAEvaluar)
```

El resultado de aplicar este operador a un identificador JavaScript es una cadena de texto que puede puede ser: string (si el tipo de expresiónAEvaluar es cadena de texto), number (si el tipo es numérico), boolean (si el tipo es booleano o se pasan las palabras clave true ó false) object (si el tipo es un objeto), function (si el tipo es una función o un método) o undefined si el identificador pasado no existe en el ámbito en que se trata de utilizar.

En este ejemplo vemos cómo obtener tipos conocidos de variables

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html><head><title>Ejemplo aprenderaprogramar.com</title><meta charset="utf-8">
<script type="text/javascript">
function ejemplo() {
var miString = 'cosa';
var miNumeroEntero = 33;
var miNumeroDecimal = 23.65;
var miBooleano = true;
var miFuncion = new function(a, b) {return (a+b);}
msg = 'Tipo de miString: '+typeof miString + '\nTipo de miNumeroEntero: '+typeof miNumeroEntero;
msg = msg + '\nTipo de miNumeroDecimal: '+typeof miNumeroDecimal + '\nTipo de miBooleano: '+typeof miBooleano;
msg = msg + '\nTipo de miFuncion: '+typeof miFuncion + '\nTipo de Math: '+typeof Math;
msg = msg + '\nTipo de Date: '+typeof Date + '\nTipo de algoIndefinido: '+typeof algoIndefinido;
if (typeof miNumeroEntero == 'string') {msg = msg + 'miNumeroEntero es de tipo string';}
else {msg = msg + '\nmiNumeroEntero no es de tipo string';}
alert(msg);
}
</script>
</script></head>
<body onload="ejemplo()" >
<div id="cabecera"><h2>Cursos aprenderaprogramar.com</h2><h3>Ejemplos JavaScript</h3></div>
</body></html>
```

AMBITO DE VARIABLES: GLOBAL Y LOCAL

La primera aproximación al concepto de ámbito (scope) va a consistir en distinguir entre las variables declaradas fuera de cualquier función, a las que se denomina variables globales y serán conocidas por todas las funciones, frente a variables declaradas dentro de una función concreta, que sólo serán conocidas dentro de esa función. Los parámetros pasados a funciones funcionan como si fueran variables locales a la función.

Escribe este código y comprueba sus resultados:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html><head><title>Ejemplo aprenderaprogramar.com</title><meta charset="utf-8">
<script type="text/javascript">

var idGlobal=33;
var msg = 'Variable global';

function ejemplo(numero) {
var miVarLocal = 'Soy una variable local';
var msg = 'idGlobal es conocido en la función ejemplo. idGlobal vale: '+idGlobal;
msg = msg + '\n\nEn ejemplo el valor de numero es recibido como parametro (local). numero es: '+numero;
msg = msg + '\n\n La variable local miVarLocal contiene: '+miVarLocal;
alert(msg); //msg es local a esta función, aunque exista otra variable con este nombre se reconoce sólo la más próxima
ejemplo2();
}

function ejemplo2(){
var msg = 'idGlobal es conocido en la función ejemplo2. idGlobal vale: '+idGlobal;
//msg = msg + '\n\nEjemplo2 desconoce qué es numero porque es local a otra función: aquí numero es '+numero;
//msg = msg + '\n\nEjemplo2 desconoce qué es miVarLocal porque es local a otra función: aquí miVarLocal es
'+MiVarLocal;
if (typeof miVarLocal != 'undefined') { msg=msg+'\n\nEjemplo2 desconoce qué es miVarLocal porque es local a otra
función: aquí miVarLocal es '+miVarLocal }
else {msg=msg+'\n\nmiVarLocal No existe en el ámbito de la función ejemplo2\n\n';}
alert(msg); //msg es local a esta función
}

</script>
</script></head>
<body onload="ejemplo(5)" >
<div id="cabecera"><h2>Cursos aprenderaprogramar.com</h2><h3>Ejemplos JavaScript</h3></div>
</body></html>
```

El resultado esperado es:

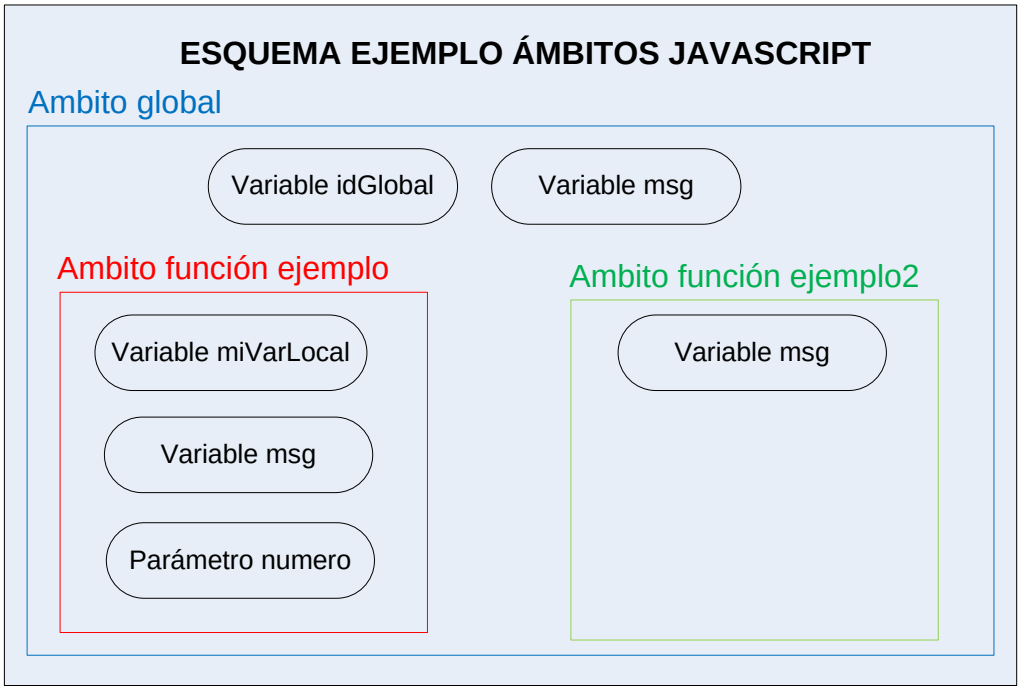
idGlobal es conocido en la función ejemplo. idGlobal vale: 33
En ejemplo el valor de numero es recibido como parametro (local). numero es: 5
La variable local miVarLocal contiene: Soy una variable local
idGlobal es conocido en la función ejemplo2. idGlobal vale: 33
miVarLocal No existe en el ámbito de la función ejemplo2

La variable global idGlobal es conocida por todas las funciones. De acuerdo con esto, la variable msg también es conocida por todas las funciones. Sin embargo, al declararse una variable local en las funciones con igual nombre, decimos que la variable global queda "tapada" por la variable local, es decir, al invocar dicha variable dentro de la función primeramente se busca si existe en el ámbito de dicha función (como local) y si es así, se devuelve dicha variable ignorando las que puedan existir en ámbitos externos. En resumen: siempre se busca si un identificador existe en el ámbito en que se invoca, si no es así se trata de localizar en el siguiente ámbito más próximo y así sucesivamente hasta llegar al ámbito global. Si no existe en el ámbito global y se trata de utilizar se obtiene un error de tipo << ReferenceError: MiVarLocal is not defined>>. Recordar que los errores no se muestran directamente en pantalla, sino que hemos de activar la consola de depuración del navegador para poder visualizarlos.

En el código aparecen dos líneas comentadas que si tratamos de ejecutar nos devuelven un error, ya que estamos tratando de invocar identificadores de variables o parámetros desconocidos en ese ámbito.

La sintaxis: if (typeof miVarLocal != 'undefined') nos permite determinar si el identificador miVarLocal es conocido en el ámbito. Podríamos también haber invertido la lógica escribiendo if (typeof miVarLocal == 'undefined') y cambiando el orden de las sentencias a ejecutar.

Gráficamente podríamos ver los ámbitos como espacios donde las variables son conocidas:



IMPLICACIONES DEL USO DE VAR EN EL ÁMBITO DE LAS VARIABLES

Nosotros estamos trabajando declarando una variable con la palabra clave var. Por ejemplo var miNumero = 33;

Sin embargo, el uso de var no es obligatorio, y el intérprete considera que una asignación de contenido a una variable que no ha sido declarada es una declaración "implícita". Por ejemplo podríamos escribir directamente miNumero=33; alert(miNumero) ; sin hacer uso de la palabra clave var.

Cuando no se escribe la palabra clave var dentro de una función tiene una implicación: la variable automáticamente es tratada como una variable global aún a pesar de estar dentro de una funcion. Ejecuta este código:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html><head><title>Ejemplo aprenderaprogramar.com</title><meta charset="utf-8">
<script type="text/javascript">

function ejemplo() {miVarLocal = 'Soy una variable local'; ejemplo2();}

function ejemplo2(){
var msg= '';
if (typeof miVarLocal != 'undefined') { msg=msg+'\n\nEjemplo2 conoce qué es miVarLocal porque se'+
' declaró en otra función sin var: aquí miVarLocal es '+miVarLocal+ ' y es de tipo '+(typeof miVarLocal) }
else {msg=msg+'\n\nmiVarLocal No existe en el ámbito de la función ejemplo2\n\n';}
alert(msg); //msg es local a esta función
}
</script>
</script></head>
<body onload="ejemplo()" >
<div id="cabecera"><h2>Cursos aprenderaprogramar.com</h2><h3>Ejemplos JavaScript</h3></div>
</body></html>
```

El resultado, que podemos ver como a priori poco lógico, es el siguiente:

<<Ejemplo2 conoce qué es miVarLocal porque se declaró en otra función sin var: aquí miVarLocal es Soy una variable local y es de tipo string>>

Esto nos da pie a que podamos plantear las siguientes situaciones y resultados:

Forma de declaración	Lugar de declaración	Efecto
Implícita sin uso de var	Fuera de cualquier función	La variable es global
Usando var	Fuera de cualquier función	La variable es global
Implícita sin uso de var	Dentro de una función	La variable es global
Usando var	Dentro de una función	La variable es local

En general recomendaremos hacer uso siempre de var, de forma que las variables declaradas dentro de funciones trabajen como variables locales. Esto hace la programación más segura y evita saturar el espacio global de nombres que en caso de repetición pueden crear conflictos.

Tener en cuenta que si una variable como mensaje es declarada como variable global y luego es utilizada dentro de una función sin hacer uso de var, su valor queda modificado al considerarse que se está haciendo referencia a la variable global (única) aunque nuestra intención no fuera esa. Esto refuerza la recomendación de hacer siempre uso de var para declarar variables.

No deben declararse indiscriminadamente variables sin uso de var dentro de funciones porque eso podría calificarse como "abuso de variables globales" que a la larga traerá complicaciones. Las variables globales deben ser usadas de forma controlada y en su justa medida.

EJERCICIO

Analiza este código JavaScript y responde a las preguntas:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html><head><title>Ejemplo aprenderaprogramar.com</title><meta charset="utf-8">
<script type="text/javascript">
var target_date = new Date("Jan 1, 2089").getTime();
var days, hours, minutes, seconds;
function ejemplo() {
var countdown = document.getElementById("countdown");
setInterval(function () {
   var current_date = new Date().getTime();
   var seconds_left = (target_date - current_date) / 1000;
   days = parseInt(seconds_left / 86400);
   seconds_left = seconds_left % 86400;
   hours = parseInt(seconds_left / 3600);
   seconds_left = seconds_left % 3600;
   minutes = parseInt(seconds_left / 60);
   seconds = parseInt(seconds_left % 60);
   countdown.innerHTML = 'Para el 1 de enero de 2089 faltan '  +days+ ' días, ' + hours + " horas, "
   + minutes + " minutos, " + seconds + " segundos";
            }, 1000);
}
</script>
</head>
<body onload="ejemplo()" >
<div id="cabecera"><h2>Cursos aprenderaprogramar.com</h2><h3>Ejemplos JavaScript</h3>
<span id="countdown"></span></div>
</body></html>
```

a) Haz una lista de las variables que intervienen indicando para cada variable cuál es su nombre, cuál es su tipo, cuál es su cometido y si está definida como variable global o variable local.

Ejemplo: la variable target_date es de tipo Date, su cometido es almacenar la fecha futura respecto de la cual el script va a mostrar los días, horas, minutos y segundos que faltan para alcanzar dicha fecha, y está definida como variable global.

b) ¿Qué ocurre si definimos la variable current_date en el ámbito global en vez de dentro de la función? ¿Por qué ocurre esto?

c) ¿Podríamos definir todas las variables como locales a la función y prescindir de las variables globales? Si se pudiera hacer, ¿crees que sería positivo para el diseño del código, su mantenimiento y ampliabilidad, o por el contrario, que sería negativo?

d) ¿Qué métodos de los empleados en el código devuelven valores en milisegundos?

e) Razona la lógica del código. Con los contenidos que hemos visto en el curso hasta el momento, debes ser capaz de entender todo el código que aparece en el script.

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

Próxima entrega: CU01168E

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