JavaScript 2015 (II). Constantes y variables.

En este artículo vamos a introducir dos conceptos. Por una parte, la creación de constantes, algo que, hasta ahora, estaba vedado en JavaScript. Por otro lado veremos que las variables pueden emplearse de una forma constreñida al contexto de un bloque de código. Son novedades ligeras, pero que nos dan algunas posibilidades que, hasta esta versión, se encontraban reservadas a otros lenguajes. Además, nos servirán como introducción para los siguientes artículos.

CONSTANTES

El concepto de constantes, en sí mismo, no es nuevo para ningún programador. Lo que sí es una novedad es poder usarlas en JavaScript. Ahora contamos con la palabra clave const, que nos permitirá crear una constante, y asignarle un valor. Los nombres de constantes, al igual que se hace con otros lenguajes, deberían escribirse con mayúsculas. Esto no es una regla sintáctica de JavaScript. Tú puedes escribirlos cómo quieras. Sin embargo, por convencionalismo, es bueno que escribas todo el nombre de una constante en letras mayúsculas.

Una constante es, exactamente, eso: constante. Esto quiere decir que si, después de declarada, intentas reasignarle otro valor diferente, se producirá un error en tiempo de ejecución. El valor que se le asigna en la declaración debe mantenerse durante todo el ciclo de vida del script.

A continuación tienes un ejemplo del uso de una constante:

```
<!DOCTYPE html>
<html lang="es">
<head>
 <meta charset="UTF-8">
 <title>JS 6</title>
</head>
<body>
 <div id="resultados"></div>
 <script language="javascript">
  window.onload = function(){
   const MIS_MENSAJES = "Mis mensajes recibidos.";
   document.getElementById("resultados").innerHTML = MIS_MENSAJES;
  }
 </script>
</body>
</html>
```

Como ves, se usa como una variable, con dos diferencias:

Se declara con la palabra reservada const, en lugar de con var.
Si intentásemos reasignarle un valor en otra línea de código, nos daría un error.

VARIABLES DE BLOQUE

Sabemos lo que es el ámbito de variables en JavaScript. Es el contexto en el que una variable es accesible, bien para lectura o escitura. Hasta ahora, la única restricción con la que contábamos eran las funciones de usuario. Una variable declarada en el cuerpo

general del script vive en todo el script, dentro y fuera de las funciones. En cambio, una variable declarada dentro de una función sólo "vive" dentro de esa función, no siendo accesible desde fuera de la misma. Si hay que pasarle un valor desde fuera, es necesario pasárselo cómo argumento a la función. Si hay que sacarla fuera, es necesario devolverla desde la función con return. Esto no es nada nuevo. Es, prácticamente, el abc del ámbito de variables. Lo podemos ver claro en el siguiente ejemplo:

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <title>JS 6</title>
</head>
<body>
 <div id="capaDeResultados"></div>
 <script language="javascript">
  function mostrarExterna(){
   document.getElementById("capaDeResultados").innerHTML += "La variable externa dentro de la función vale "+externa+"<br>";
  }
  function variableInterna(){
   var interna = 20;
   document.getElementById("capaDeResultados").innerHTML += "La variable interna dentro de la función vale "+interna+"<br>";
  }
  var externa = 10;
  document.getElementById("capaDeResultados").innerHTML += "La variable externa fuera de la función vale "+externa+"<br>";
  mostrarExterna();
  variableInterna();
  document.getElementById("capaDeResultados").innerHTML += "La variable interna fuera de la función vale "+interna+"<br>";
 </script>
</body>
</html>
```

En la línea 17 declaramos una variable a la que hemos llamado externa. A continuación la mostramos en pantalla y vemos que se visualiza sin problemas. Después, en la línea 19 invocamos a una función que la vuelve a visualizar, esta vez desde dentro del contexto de dicha función (mostrarExterna()), también sin ningún problema.

En la línea 20 llamamos a la función variableInterna(). Lo que hace es declarar una variable, a la que hemos llamado interna, dentro del contexto de la función, y la muestra sin problemas. Sin embargo, en la línea 21, fuera de la función, intentamos mostrar la variable interna y vemos, si abrimos la consola del navegador, que se produce un error. Esto se debe a que, al haber sido declarada interna dentro de una función, no existe fuera de la misma.

Todo esto es el comportamiento normal y, si has tocado JavaScript en alguna ocasión (y doy por sentado que es así) estás familiarizado con el concepto tradicional de ámbito de variables.

Sin embargo, ahora quiero que le eches un vistazo al siguiente código:

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <title>JS 6</title>
</head>
<body>
 <div id="capaDeResultados"></div>
 <script language="javascript">
  for (var i = 1; i <= 4; i ++){
   document.getElementById("capaDeResultados").innerHTML += "Valor de i (dentro del bucle for): "+i+"<br>";
  }
```

```
 document.getElementById("capaDeResultados").innerHTML += "Valor de i (tras finalizar el bucle for): "+i+"<br>";
</script>
</body>
</html>
```

Cómo ves, tenemos un bucle for con una variable de control, llamada i. El bucle itera cuatro veces y, al terminar, nos muestra el valor final de la variable. Esta última impresión se efectúa fuera del bucle for. El resultado (ya lo habrás deducido) es el siguiente:

```
Valor de i (dentro del bucle for): 1
Valor de i (dentro del bucle for): 2
Valor de i (dentro del bucle for): 3
Valor de i (dentro del bucle for): 4
Valor de i (tras finalizar el bucle for): 5
```

Fíjate, en la línea resaltada, que hemos declarado la variable de control con la sentencia tradicional var. Ahora vamos a hacer el mismo código declarando la variable con la palabra reservada let. Esta es nueva, nativa de JavaScript 2015. El código queda así:

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <title>JS 6</title>
</head>
<body>
 <div id="capaDeResultados"></div>
 <script language="javascript">
  for (let i = 1; i <= 4; i ++){
   document.getElementById("capaDeResultados").innerHTML += "Valor de i (dentro del bucle for): "+i+"<br>";
  }
  document.getElementById("capaDeResultados").innerHTML += "Valor de i (tras finalizar el bucle for): "+i+"<br>";
 </script>
</body>
</html>
```

Y el resultado es el siguiente:

```
Valor de i (dentro del bucle for): 1
Valor de i (dentro del bucle for): 2
Valor de i (dentro del bucle for): 3
Valor de i (dentro del bucle for): 4
```

Cómo ves, mientras estamos dentro del bucle, todo va igual que antes. Sin embago, si miras la consola del navegador verás que la línea 13, que intenta usar la variable fuera del bucle, genera un error.

Por lo tanto, declarando una variable con let, en lugar de con var, restringimos su ámbito de vida al bloque de código donde la declaremos (y a los bloques que pudiera haber internos a este). La variable no existe fuera de ese bloque de código. Esto es válido para todo tipo de bucles (for, while, etc) y también para condicionales (if, switch, etc).