APRENDERAPROGRAMAR.COM

EFECTOS JAVASCRIPT VISUALES, DE IMÁGENES, TEXTO... RECURSIÓN. SETTIMEOUT NO FUNCIONA EN BUCLES FOR, WHILE, ¿? (CU01166E)

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

Fecha revisión: 2029

Resumen: Entrega nº66 del Tutorial básico "JavaScript desde cero".

Autor: César Krall

EFECTOS JAVASCRIPT

Existen diferentes APIs y librerías JavaScript que permiten generar efectos visuales muy llamativos. En este curso no queremos entrar a estudiar estas posibilidades ya que nos estamos centrando en los fundamentos de JavaScript. Pero ya con estos podemos generar efectos variados.

Los efectos visuales muchas veces se basarán en el uso de recursión y funciones relacionadas con la repetición en el tiempo y ejecución de funciones con cierto retardo en el tiempo.

RECURSIÓN CON JAVASCRIPT

La recursión es una técnica de programación admitida por la mayor parte de los lenguajes modernos basada en la autollamada de un método o función a sí mismo. Estas llamadas van creando "una pila de llamadas" hasta que se llega a lo que se denomina caso base: una situación en que ya no se realiza una nueva llamada recursiva, sino que se devuelve un resultado concreto que da lugar a la salida de la recursión.

La recursión no es fácil de entender ni de explicar. No te preocupes si después de leer lo que vamos a exponer no terminas de comprenderla del todo. Ten en cuenta que muchos programadores no usan o no saben manejar la recursión y que aprender su uso y manejo requiere tiempo y práctica.

Uno de los ejemplos más habituales de recursión es el cálculo del factorial de un número. El factorial de un número n se calcula como n * (n-1) * (n-2) * (...) * 1. Por ejemplo el factorial de 3 es 3*2*1 = 6, mientras que el factorial de 5 es 5*4*3*2*1 = 120.

La forma más intutiva de calcular el factorial es no recursiva, sino usando un bucle tradicional como vamos a ver ahora. 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">
function ejemplo(numero) { alert ('El factorial de 5 es '+factorialIterativo(numero)); }
function factorialIterativo(numero){ var factorial = 1;
//Factorial de numero
for (var i=numero; i>0; i--){ factorial = factorial * (i); }
return factorial;
}
</script>
</script></head>
<body onload="ejemplo(5)" >
<div id="cabecera"><h2>Cursos aprenderaprogramar.com</h2><h3>Ejemplos JavaScript</h3></div>
</body></html>
```

Ahora vamos a plantear el factorial de forma recursiva. El factorial de un número se puede ver como factorial de n = n * (factorial de n-1)

Así el factorial de 5 podríamos verlo como 5 * (factorial de 4), que a su vez es 5 * 4 * (factorial de 3), que a su vez es 5 * 4 * 3 * (factorial de 2) … ¿Cuándo termina este proceso recursivo? Cuando llegamos a un caso en el que el resultado es elemental, es decir, un caso donde no es necesario utilizar factorial de n = n * (factorial de n-1), sino simplemente factorial de n = resultado elemental. Este caso, que se denomina caso base, para el factorial lo tenemos con el 1: factorial de 1 es 1, caso elemental.

El código recursivo para el factorial sería:

```
<!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(numero) { alert ('El factorial de '+numero+' es '+factorialRecursivo(numero)); }

function factorialRecursivo(numero){
if (numero==0) {return 1;} //caso base
else {return numero*factorialRecursivo(numero-1);} //caso recursivo
}

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

Los pasos que da el factorial recursivo son:

5 * factorialRecursivo(4) -- > 5 * 4 * factorialRecursivo(3) -- > 5 * 4 * 3 * factorialRecursivo(2) -- >

5 * 4 * 3 * 2 * factorialRecursivo(1) y llega al caso base, devolviendo 5*4*3*2*1 = 120.

A la hora de plantear un proceso recursivo siempre es necesario tener presente la necesidad de que exista un caso base: si no es así, la recursión puede no terminar nunca y se entra en procesos que se asemejan a un bucle infinito. Cuando esto ocurre, en la consola de depuración de JavaScript nos podemos encontrar mensajes como “Too much recursion”.

EL PROBLEMA DEL SETTIMEOUT DENTRO DE BUCLES

Supongamos que queremos ejecutar un proceso de forma que entre cada ejecución haya un intervalo de 5 segundos y escribimos un bucle con un setTimeOut llamando a la función oportuna dentro del bucle:

```
while (contador<numeroNodos){
contador = contador+1;
referenciaTimer[contador] = setTimeout (cambiarNodo(nodosTituloCurso[contador]), 5000);
}
```

"La idea" es realizar el proceso cada 5 segundos, sin embargo hay un fallo de planteamiento que hace que esto no funcione: lo que ocurre en realidad es que se ejecuta el bucle en todas sus iteraciones de forma prácticamente instantánea, lo que da lugar a que el setTimeout no se ejecute.

Una función para detenerse tiene que llamarse a sí misma o utilizar algún mecanismo que fuerce la detención.

Vamos a analizar esto con mayor detenimiento. Para ello escribe y 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">
<style type="text/css">
*{font-family: verdana, sans-serif;}
.nodoNuevo{background-color: black; color:white; width:100px;
text-align:center; padding:20px; font-size:32px; float:left;}
</style>
<script type="text/javascript">
var contador = 1;
function mostrarNumConRetardo() {
setTimeout(crearNodo, 1000);
}
function crearNodo() {
var nodoHijo = document.createElement("div");
nodoHijo.className="nodoNuevo";
nodoHijo.innerHTML = ''+contador;
document.body.appendChild(nodoHijo);
contador = contador+1;
}
</script>
</script></head>
<body onload="mostrarNumConRetardo()" >
<div id="cabecera"><h2>Cursos aprenderaprogramar.com</h2><h3>Ejemplos JavaScript</h3></div>
</body></html>
```

Con este código hacemos que se muestre el número 1 por pantalla después de 1 segundo. Hasta aquí todo correcto. Ahora supongamos que queremos que se cuente hasta 10 y lo hacemos con un bucle. Para ello modificamos la función mostrarNumConRetardo que queda así:

```
function mostrarNumConRetardo() {
for(var i=0; i<10; i++){
setTimeout(crearNodo, 1000);
}
}
```

El resultado no es que se vayan mostrando progresivamente 1, luego 2, luego 3, luego 4... sino que se nos muestran de golpe todos los números: 1, 2, 3,4, 5, 6, 7, 8, 9, 10. ¿Por qué? Porque el bucle se ejecuta de forma inmediata quedando 10 órdenes de que se muestren dentro de 1 segundo los números, cuando transcurre un segundo se ejecutan todas las órdenes.

Para solucionar este problema tenemos que forzar la detención y que la siguiente llamada se ejecute después de que el retardo previo haya tenido lugar. Hay varias maneras de solucionar esto. Aquí vamos a mostrar una de ellas, basada en llamadas recursivas.

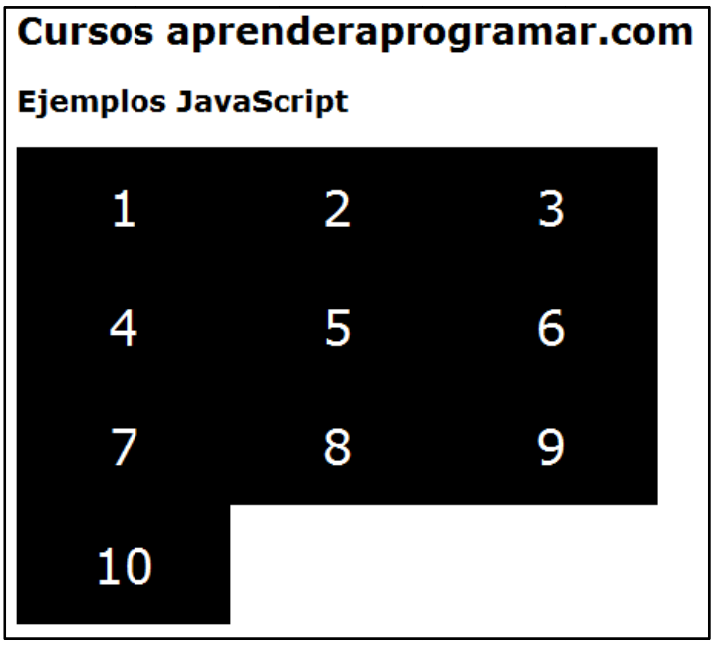
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>Ejemplo aprenderaprogramar.com</title>
<meta charset="utf-8">
<style type="text/css">
		*{font-family: verdana, sans-serif;}
		.nodoNuevo{background-color: black; color:white; width:100px;
		text-align:center; padding:20px; font-size:32px; float:left;}
</style>
<script type="text/javascript">
var contador = 1;
function mostrarNumConRetardo() { setTimeout(crearNodo, 1000);}
function crearNodo() {
		if (contador==11) {
		//Caso base: fin de la recursión (no hacemos nada)
		} else { //Caso recursivo
		var nodoHijo = document.createElement("div");
		nodoHijo.className="nodoNuevo";    nodoHijo.innerHTML = ''+contador;
		document.getElementById('cabecera').appendChild(nodoHijo);       contador = contador+1;
		setTimeout(crearNodo, 1000);
		}
}
</script>
</script>
</head>
<body onload="mostrarNumConRetardo()" >
<div id="cabecera" style="width:500px;">
<h2>Cursos aprenderaprogramar.com</h2>
<h3>Ejemplos JavaScript</h3></div>
</body>
</html>
```

En este código observamos lo siguiente:

a) No existe un bucle como existía en el ejemplo puesto anteriormente. Lo que existen son llamadas recursivas que “simulan” un bucle (de hecho, llevamos un contador).

b) Cada llamada recursiva introduce un retardo. De este modo, la siguiente ejecución no comienza si no se ha producido el retardo. A diferencia del caso de un bucle, donde la ejecución es instantánea, en este caso la ejecución va siendo progresiva: no comienza la siguiente recursión hasta que pasa el tiempo de retardo especificado.

El resultado esperado es que vayan apareciendo en pantalla progresivamente los números 1, 2, 3, … hasta 10, con un pequeño retardo en la aparición de cada número.



CREAR EFECTOS JAVASCRIPT: HASTA DONDE LA IMAGINACIÓN ALCANCE...

Con las herramientas que hemos visto a lo largo del curso podemos crear potentes animaciones con JavaScript. Es cierto que existen frameworks JavaScript y APIs que nos facilitan crear animaciones, pero nosotros mismos podemos plantear animaciones usando puro código JavaScript hasta donde la imaginación nos alcance...

Escribe el siguiente código y comprueba los resultados de ejecución. Este código corresponde a una página web real que lanzamos en el dominio http://aprendeaprogramar.es. El código es sencillo, puede resultar un poco extenso para estudiar pero merece la pena, ya que aborda diferentes materias que hemos ido estudiando a lo largo del curso como acceso y manipulación del DOM, creación de efectos y retardo con setTimeOut, recursión, cambio de estilos CSS, uso de variables de distintos tipos, etc.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html><head>
<title>Cursos aprende a programar</title>
<meta charset="utf-8">
<meta name="description" content="Aprende a programar con cursos reconocidos por su calidad didáctica">
<meta name="keywords" content="HTML, CSS, JavaScript, Java, Visual Basic, Joomla, PHP">
<style type="text/css">
*{font-family: verdana, sans-serif;}
#principal{text-align:center;width:95%; margin: 0 auto;}
.tituloCurso {color: white; float:left; padding: 36px 44px; font-size: 2.65em; font-style:bold; text-decoration:none;}
a:hover{color:orange !important;}
</style>
<script type="text/javascript">
function ejemplo(){ var nodosTituloCurso = document.querySelectorAll(".tituloCurso");
var contador = 0; var nodosCambiados = new Array(); var tocaCambiar = true;
setTimeout(function() {ejemploAccion(nodosTituloCurso, nodosCambiados, contador, tocaCambiar)}, 1500);
}
```

```
function ejemploAccion(nodosTituloCurso, nodosCambiados, contador, tocaCambiar) {
	if (contador<nodosTituloCurso.length){
	var indice = Math.floor((Math.random() * (nodosTituloCurso.length)));
	if (nodosCambiados.length!=0) {
		for (var i=0; i<nodosCambiados.length; i++) {
			if(nodosCambiados[i]==indice) {tocaCambiar = false;}
		}
	}
	if (tocaCambiar==true) {
	cambiarNodo(nodosTituloCurso[indice]);
	nodosCambiados.push(indice);
	contador = contador+1;
	setTimeout(function() {ejemploAccion(nodosTituloCurso, nodosCambiados, contador, tocaCambiar)}, 1500);
	} else {tocaCambiar=true; ejemploAccion(nodosTituloCurso, nodosCambiados, contador, tocaCambiar);}
	}
	else { //Caso base fin de la recursión
		document.body.style.backgroundColor='black';
		document.getElementById('principal').style.color='white';
		for (var i=0; i<nodosTituloCurso.length; i++) {
			nodosTituloCurso[i].style.color='yellow';
		}
	}
}

function cambiarNodo(elNodo){ elNodo.style.backgroundColor = 'black'; }
</script>
</head>
<body onload="ejemplo()">
<div id="principal"><h1>Cursos de programación</h1>
<h3>Reconocidos por su calidad didáctica</h3>
<div ><a class="tituloCurso" href="http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=28&Itemid=59">
Fundamentos</a></div>
<div ><a class="tituloCurso" href="http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=68&Itemid=188">
Java</a></div>
<div ><a class="tituloCurso" href="http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=69&Itemid=192">
HTML</a></div>
<div ><a class="tituloCurso" href="http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=75&Itemid=203">
CSS</a></div>
<div ><a class="tituloCurso" href="http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=78&Itemid=206">
JavaScript</a></div>
<div ><a class="tituloCurso" href="http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=70&Itemid=193">
PHP</a></div>
<div ><a class="tituloCurso" href="http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=38&Itemid=152">
Joomla</a></div>
<div ><a class="tituloCurso"
href="http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=37&Itemid=61"> Visual Basic</a></div>
<div ><a class="tituloCurso" href="http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=28&Itemid=59">
Pseudocódigo</a></div>
<div ><a class="tituloCurso"
href="http://aprenderaprogramar.com/index.php?option=com_content&view=section&layout=blog&id=7&Itemid=26">
Libros/ebooks</a></div>
<div ><a class="tituloCurso"
href="http://aprenderaprogramar.com/index.php?option=com_content&view=article&id=64&Itemid=87">Cursos tutorizados</a></div>
</div></body></html>
```

EJERCICIO 1

Realiza una descripción de cada una de las instrucciones del código del ejemplo anterior indicando cuál es su cometido. Si has seguido el curso desde el principio, debes ser capaz de interpretar todo el código.

Ejemplo: La primera línea <!DOCTYPE ... realiza la declaración de tipo de documento HTML a efectos de que los navegadores interpreten en qué versión de HTML está escrito el código.

La etiqueta html es la etiqueta de apertura del código HTML. La etiqueta head es ...

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

EJERCICIO 2

Crea una animación JavaScript (decide tú el efecto que quieres implementar: pueden ser cosas moviéndose por la pantalla, texto cambiando de tamaño o cualquier otra cosa que se te ocurra) con las herramientas que hemos ido conociendo a lo largo del curso. Ejecútalo en distintos navegadores y comprueba que se ejecute tal y como esperas en todos ellos.

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

Próxima entrega: CU01167E

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