APRENDERAPROGRAMAR.COM

NODELIST JAVASCRIPT. DIFERENCIAR NODELIST Y ARRAY. ACCEDER AL TEXTO DE NODOS CON TEXTCONTENT, INNERTEXT (CU01136E)

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

Fecha revisión: 2029

Resumen: Entrega nº36 del Tutorial básico “JavaScript desde cero”.

Autor: César Krall

DIFERENCIA ENTRE NODELIST Y ARRAY

Sabemos que instrucciones como getElementsByTagName nos devuelven una colección de nodos que en algunos momentos se denomina "array" de nodos. Sin embargo, lo que devuelven este tipo de métodos no es exactamente un array, sino una estructura de datos denominada NodeList.

El hecho de que getElementsByTagName, getElementsByName, querySelectorAll no devuelvan un array puede llevarnos a confusión en algunos momentos. Por ejemplo, podemos pensar en recorrer un NodeList usando un bucle for – in, y no obtener los resultados esperados porque una colección NodeList no admite el uso de for – in (sí admite sin embargo el uso de un for tradicional). Pero además del for in, hay otras posibilidades propias de los arrays que se pierden cuando se usan NodeLists. Por ello en algunos casos puede resultar de interés la conversión de un NodeList en un array.

Una peculiaridad de los NodeList es que son colecciones dinámicas cuyo contenido se actualiza automáticamente cuando la página web cambia dinámicamente. Supongamos una página web donde tenemos cinco elementos div y usamos var nodosDiv = getElementsByTagName('div'); para rescatarlos. En ese momento la colección nodosDiv tiene 5 elementos. Supongamos ahora (no vamos a indicar cómo hacerlo, con la idea nos basta), que usando JavaScript añadimos un nuevo elemento div de modo que ahora en la página web tenemos 6 elementos div. Debido al carácter dinámico de los nodeLists, nodosDiv pasa automáticamente a tener 6 elementos. Sin embargo, si trabajáramos con arrays, debido a su carácter estático, el array seguiría teniendo 5 elementos (a no ser que a través de código añadiéramos instrucciones expresas para incorporar el nuevo elemento).

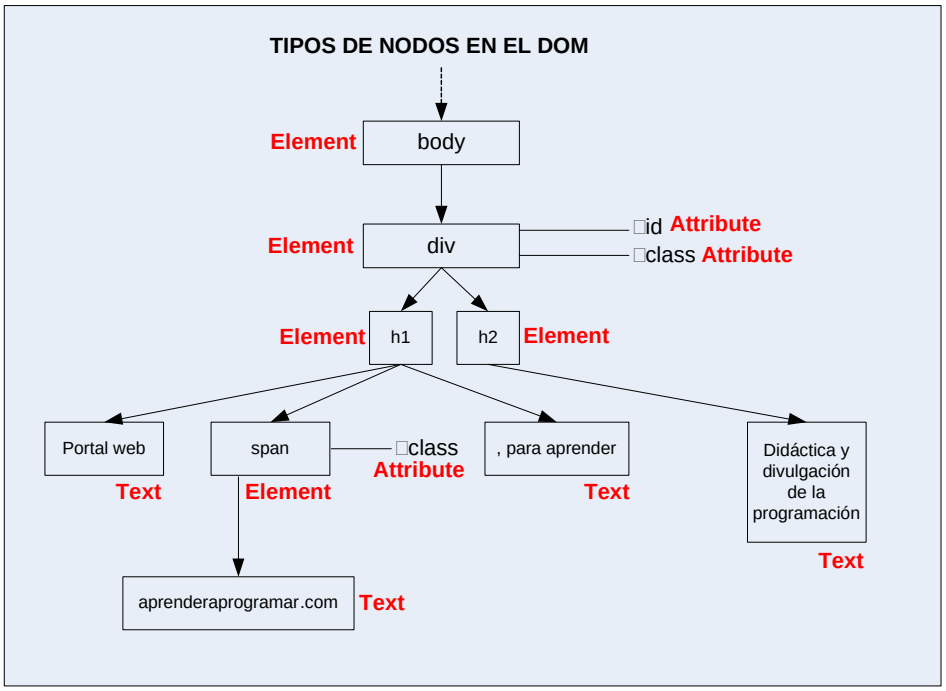
Por tanto los NodeLists podemos decir que tienen ciertas similitudes con los arrays y ciertas diferencias con estos. Los NodeList no tienen algunas posibilidades que tienen los arrays, pero a cambio son dinámicos, lo que puede resultar de gran interés en determinadas circunstancias.

ACCEDER AL TEXTO DENTRO DE NODOS: TEXTCONTENT (INNERTEXT)

Hemos visto que es posible acceder al contenido de un nodo del DOM tipo texto usando nodeValue. Por ejemplo una expresión como document.childNodes[1].childNodes[2].nodeValue nos puede permitir extraer el texto de un nodo de tipo texto.

También podemos acceder a nodos del DOM y extraer texto usando expresiones como document.firstChild.firstChild.nextSibling.nodeValue, o combinando expresiones de este tipo con el uso de childNodes.

Pero dado cómo se genera la estructura de nodos del DOM, muchas veces el acceso a nodos de tipo texto resulta complicado.



El esquema anterior representa que dentro del elemento div se encuentra el texto "Portal web aprenderaprogramar.com, para aprender. Didáctica y divulgación de la programación". Sin embargo ese texto se encuentra repartido por diferentes nodos debido a la presencia de subdivisiones dentro del elemento div creadas por h1, h2, span, etc.

Si queremos extraer (o definir) el texto contenido en todos los nodos derivados de un nodo padre podemos usar la propiedad textContent de los nodos.

La sintaxis a emplear normalmente será del tipo:

```
var textoEnElNodoYSusHijos = nombreDelNodo.textContent;
```

También podemos escribir nombreDelNodo.textContent = "El ganador es Barack Obama";

Es importante resaltar que con textContent se extrae no sólo el texto directamente asociado al nodo, sino también el texto contenido en otros nodos hijos.

En el ejemplo gráfico anterior, si extraemos el textContent del nodo div obtendremos "Portal web aprenderaprogramar.com, para aprender. Didáctica y divulgación de la programación", que incluye el texto encontrado en todos sus nodos hijos.

Hay una alternativa de funcionamiento muy similar a textContent que ha sido utilizada por algunos navegadores: innerText. Su comportamiento es muy similar al de textContent, aunque no exactamente igual. innerText no es reconocido por todos los navegadores y no se considera un estándar válido, por ello no le prestaremos más atención y no lo usaremos. No obstante, hemos considerado conveniente citarlo por si lo encuentras mientras revisas el código en alguna página web.

A modo de resumen: usaremos textContent para extraer el contenido de texto dentro de un nodo y sus nodos hijos.

RECORRER NODOS DEL DOM

Es frecuente que se escriba que var elementosDiv = document.getElementsByTagName('div') nos devolverá un array con todos los nodos de tipo element cuya etiqueta sea div, empezando con índice cero: elementosDiv[0], elementosDiv[1], elementosDiv[2], elementosDiv[3] …

Si esto es así, debería ser posible recorrer la colección de nodos obtenidos usando un for in. Sin embargo, es posible que nos encontremos que el for in no sirva para recorrer los nodos obtenidos por un método de selección de nodos del DOM. ¿Por qué?

Vamos a ver primero un ejemplo de código donde podemos comprobarlo y después lo comentaremos. Escribe este código y guárdalo con un nombre como ejemplo.html. Aquí usamos textContent, innerText, bucles for normales y bucles for in. Comprueba los resultados en tu navegador:

```
<!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">
body {font-family: sans-serif; margin:25px;}
#pulsador {padding:15px; width: auto; display: inline-block;
margin: 25px; cursor: pointer;
color: white; border-radius: 40px; background: rgb(202, 60, 60);}
#pulsador:hover, #marcaOperadores:hover {background: rgb(66, 184, 221);}
</style>
<script type="text/javascript">
function ejemploForIn() {
var nodoSpan = document.getElementsByTagName('span');
var msg = 'Primer for \n';
alert ('items en nodoSpan son ' + nodoSpan.length);
//Primer for
for (var i=0; i< nodoSpan.length; i++) { msg = msg + i + ': contiene ' +nodoSpan[i].textContent + ' , y nodeValue
'+nodoSpan[i].firstChild.nodeValue +'  ** \n'; }
alert ('Contenido de los nodos span \n ' + msg);
msg = 'Segundo for \n';
//Segundo for
for (var i=0; i< nodoSpan.length; i++) { msg = msg + i + ': contiene ' +nodoSpan[i].innerText + '** \n'; }
alert ('Contenido de los nodos span \n ' + msg);
msg = 'Tercer for \n';
//Tercer for
for (i in nodoSpan) { msg = msg + i + ': contiene ' + nodoSpan[i].textContent + ' -- \n'; }
alert ('Contenido de los nodos span \n ' + msg);
}
</script>
</head>
<body>
<div id="cabecera"><h2>Cursos aprenderaprogramar.com</h2><h3>Ejemplo for in JavaScript</h3>
</div>
<div id="calculadora"><span>7</span><span>+</span><span>4</span><span>-</span>
                        <span>1</span><span>÷</span><span>0</span><span>.</span>
                        <span>=</span><span>x <p>Genera un <b>nuevo</b> nodo</p></span>
</div>
<div id ="pulsador" onclick="ejemploForIn()"> Probar </div>
</body></html>
```

El resultado esperado depende del navegador que utilicemos. Aquí vamos a mostrar los resultados obtenidos en dos navegadores diferentes:

Resultados navegador 1	Resultados navegador 2
items en nodoSpan son 10 Contenido de los nodos span **Primer for** 0: contiene 7 , y nodeValue 7 ** 1: contiene + , y nodeValue + ** 2: contiene 4 , y nodeValue 4 ** 3: contiene - , y nodeValue - ** 4: contiene 1 , y nodeValue 1 ** 5: contiene ÷ , y nodeValue ÷ ** 6: contiene 0 , y nodeValue 0 ** 7: contiene . , y nodeValue . ** 8: contiene = , y nodeValue = ** 9: contiene x Genera un nuevo nodo , y nodeValue x ** Contenido de los nodos span **Segundo for** 0: contiene undefined // 1: contiene undefined // 2: contiene undefined // 3: contiene undefined // 4: contiene undefined // 5: contiene undefined // 6: contiene undefined // 7: contiene undefined // 8: contiene undefined // 9: contiene undefined // Contenido de los nodos span **Tercer for** 0: contiene 7 -- 1: contiene + -- 2: contiene 4 -- 3: contiene - -- 4: contiene 1 -- 5: contiene ÷ -- 6: contiene 0 -- 7: contiene . -- 8: contiene = -- 9: contiene x Genera un nuevo nodo -- item: contiene undefined -- namedItem: contiene undefined -- @@iterator: contiene undefined -- length: contiene undefined --	items en nodoSpan son 10 Contenido de los nodos span **Primer for** 0: contiene 7 , y nodeValue 7 ** 1: contiene + , y nodeValue + ** 2: contiene 4 , y nodeValue 4 ** 3: contiene - , y nodeValue - ** 4: contiene 1 , y nodeValue 1 ** 5: contiene ÷ , y nodeValue ÷ ** 6: contiene 0 , y nodeValue 0 ** 7: contiene . , y nodeValue . ** 8: contiene = , y nodeValue = ** 9: contiene x Genera un nuevo nodo , y nodeValue x ** Contenido de los nodos span **Segundo for** 0: contiene 7 // 1: contiene + // 2: contiene 4 // 3: contiene - // 4: contiene 1 // 5: contiene ÷ // 6: contiene 0 // 7: contiene . // 8: contiene = // 9: contiene x Genera un nuevo nodo // **Tercer for** 0: contiene 7 -- 1: contiene + -- 2: contiene 4 -- 3: contiene - -- 4: contiene 1 -- 5: contiene ÷ -- 6: contiene 0 -- 7: contiene . -- 8: contiene = -- 9: contiene x Genera un nuevo nodo – length: contiene undefined -- item: contiene undefined -- namedItem: contiene undefined --

Del resultado obtenido podemos señalar lo siguiente:

1) Se pueden obtener distintos resultados ejecutando un mismo código en diferentes navegadores, lo cual es un problema a la hora de crear desarrollos web. Debemos asumirlo y tenerlo en cuenta en la medida de lo posible, ya que no hay forma de esquivarlo.

2) El resultado de recorrer una colección de nodos con document.getElementsByTagName('span'); no es el mismo si usamos un for tradicional que si usamos un for in. Esto nos indica que las colecciones de nodos que devuelven estos métodos no son arrays propiamente dichos. Pero si no son arrays, ¿entonces qué son? Son objetos que tienen similitudes con los arrays, denominados NodeList o listas de nodos. Los NodeLists son objetos a veces se dice que son objetos de tipo array (array-like objects). Sin embargo, no disponen de todas las posibilidades de que dispone un array. Una de las posibilidades de que no disponen, es el recorrido mediante bucles for in. Por eso cuando intentamos recorrer una colección de nodos con un for in obtenemos resultados extraños, que incluso pueden ser diferentes de un navegador a otro.

3) Un navegador reconoce innerText pero el otro no. Como hemos dicho, preferiremos usar textContent como método para extraer texto, pero nos podremos encontrar código donde aparezca innerText. Algunos navegadores tratan innerText como si fuera textContent, otros de forma ligeramente diferente, y otros directamente no reconocen innerText.

EJERCICIO

Crea un documento html con un texto en una etiqueta h1 como "Ejercicio curso aprenderaprogramar.com" y un div a continuación. Genera un script que pida cinco números al usuario usando un bucle for normal (usa prompt para pedir los datos y conviértelos a valor numérico posteriormente). Almacena los números introducidos por el usuario en un array. A continuación, accede al nodo del div y establece que muestre un texto informando del resultado de multiplicar cada uno de los números por 3 (para ello usa textContent). Ejemplo:

Al cargar la página aparecerá: Ejercicio curso aprenderaprogramar.com
… (div vacío)

Se pedirán al usuario cinco números, supongamos que introduce 1, 3, 9, 10 y 7

A continuación en la página se visualizará: Ejercicio curso aprenderaprogramar.com
Multiplicamos por 5 los números introducidos: 1*5 = 5, 3*5 = 15, 9*5 = 45, 10*5 = 50 y 7*5 =35.

Para comprobar si tus respuestas son correctas puedes consultar en los foros aprenderaprogramar.com.

Próxima entrega: CU01137E

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