APRENDERAPROGRAMAR.COM

ARRAY.PROTOTYPE JAVASCRIPT. DIFERENCIA ENTRE ARRAY ARRAY-LIKE, NODELIST. TYPEERROR: NOT A FUNCTION NO METHOD 'FOREACH' (CU01180E)

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

Fecha revisión: 2029

Resumen: Entrega nº80 del Tutorial básico “JavaScript desde cero”.

Autor: César Krall

ARRAY.PROTOTYPE, NODELIST, ARRAYS Y ARRAY-LIKE OBJECTS

En apartados anteriores del curso estudiamos que la herencia en JavaScript es una herencia basada en prototipos. En esta entrega vamos a repasar algunos conceptos sobre herencia y acceso a los objetos en la raíz de la cadena de herencia, y a estudiar algunos conceptos sobre qué son NodeList, Arrays y array-like objects.

Vamos a llamar la atención sobre el hecho de que los métodos de tipo get que actúan sobre el DOM devuelven una colección de objetos a la que se denomina NodeList o lista de nodos, que se dice son objetos array-like (similares a un array, pero no exactamente arrays). El hecho de que un NodeList no es un array se comprueba con este código. Ejecútalo y comprueba los resultados (activa la consola para ver los mensajes de error):

```
<!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 {margin-left:30px; font-family: sans-serif;}
          .estiloForm {background-color: #f3f3f3; border: solid 2px black; margin-left:20px; width: 330px; padding:10px; }
  .estiloForm label {display: block; width: 120px; float: left; text-align:right; margin-bottom: 35px; padding-right: 20px;}
           br {clear: left;} input[type="submit"], input[type="reset"] {margin:25px 5px 10px 5px;}
  </style>
<script type="text/javascript">
window.onload = function () {
          var elementosInput = document.getElementsByTagName('input'); //Elementos input
          var elementosSelect = document.getElementsByTagName('select');
          for (var i=0; i< elementosSelect.length; i++) {elementosInput.push(elementosSelect[i])};
          alert ('Tenemos un número de elementos input ó select en el formulario: '+ elementosInput.length);
}
</script></head>
<body><h2>Cursos aprenderaprogramar.com</h2><h3>Ejemplos JavaScript</h3>
  <div class="estiloForm"><form name ="formularioContacto" method="get" action="#">
    <label>Tratamiento</label>
    <input type="radio" name="tratamiento" id="tratarSr" value="Sr."/>Sr.
                    <input type="radio" name="tratamiento" id ="tratarSra" value="Sra."/>Sra.<br/>
                    <label>Nombre</label><input id="nombre" name="nombre" type="text"/><br/>
    <label>Apellidos</label><input id="apellidos" name="apellidos" type="text"/><br/>
    <label>Dirección</label><input id="direccion1" name="direccion1" type="text"/><br/>
    <label>Ciudad</label><select id="ciudad" name="ciudad">
                                <option value="">Elija opción</option>
      <option value="Mexico">México D.F. (MX)</option>
      <option value="Madrid">Madrid (ES)</option>
      <option value="Santiago">Santiago (CL)</option>
    </select><br/>
                    <label>Preferencias</label><input name="Libros" type="checkbox" />Libros
                    <input name="Peliculas" type="checkbox"  />Películas
                    <input type="submit" value="Enviar"/> <input type="reset" value="Cancelar"/>
  </form></div>
</body></html>
```

El resultado es un error similar a este: <<TypeError: elementosInput.push is not a function>>. El motivo de esto es que un objeto array-like no dispone de todos los métodos propios de los arrays. Sin embargo, los nodeList sí tienen algunas características comunes con el array, como disponer de una propiedad length que informa del número de elementos que contiene la colección de elementos.

El interés de los NodeList está en que son entidades dinámicas: si durante la ejecución de código se elimina un elemento que pertenecía a un NodeList, el NodeList queda actualizado automáticamente porque está referenciando dinámicamente. En cambio si tuviéramos los nodos en un array y se elimina un nodo, el array seguiría tal y como estaba porque el array no está ligado dinámicamente a lo que ocurra en el DOM en tiempo de ejecución.

En ocasiones nos puede interesar convertir un objeto array-like en un array verdadero. Esto se puede hacer de varias maneras:

a) Manualmente introduciendo cada elemento en un nuevo objeto array. Ejemplo modificando el código anterior (ejecútalo y comprueba los resultados):

```
<script type="text/javascript">
window.onload = function () {
	var elementosInput = document.getElementsByTagName('input'); //Elementos input
	var elementosSelect = document.getElementsByTagName('select');
	var arrayNodos = [];
	addNodeListToArray (elementosInput, arrayNodos);
	addNodeListToArray (elementosSelect, arrayNodos);
	alert ('Tenemos un número de elementos input ó select en el formulario: '+ arrayNodos.length);
}

function addNodeListToArray (elNodeList, elArray) {
	for (var i=0; i< elNodeList.length; i++) {
	var elementoActual = elNodeList[i];
	elArray.push(elementoActual);
	}
	alert ('Añadidos ' + elNodeList.length +' elementos');
}
</script>
```

El resultado esperado es que se muestre: Añadidos 9 elementos, Añadidos 1 elementos, Tenemos un número de elementos input ó select en el formulario: 10

b) Accediendo al objeto Array para aplicar un método que nos permite obtener un array a partir de un objeto array-like. La sintaxis sería cualquiera de estas (ten en cuenta que en algunos casos, especialmente con navegadores antiguos, estas sintaxis pueden no ser reconocidas):

```
var divsEnArray = Array.prototype.slice.call(document.querySelectorAll('div'));
```

```
var divs = [].slice.call(document.querySelectorAll('div'));
```

Esta sintaxis merece ser comentada. La invocación Array.prototype nos permite acceder al objeto en la cima de la jerarquía de herencia u objeto "padre" de todos los objetos de su tipo. Cuando escribimos algo como miArray = new Array(); estamos creando una instancia de un objeto tipo Array, que hereda todas sus características del "objeto padre" Array.prototype.

Normalmente invocamos métodos sobre instancias de los objetos en la cima jerárquica, pero en ocasiones (como en este ejemplo), puede que nos resulte de interés invocar directamente un método sobre el objeto padre o prototype. En este caso al invocar el método slice sin argumentos, se nos devuelve un array sin más. Para indicarle a partir de qué colección se debe devolver el array, usamos el método call indicando que el objeto que actuará como this y a partir del cual se debe devolver el array es un objeto NodeList (un array-like object).

La sintaxis [].slice.call(document.querySelectorAll('div')); es similar, pero en lugar de operar sobre el prototype utilizamos un objeto array anónimo representado por []. Aunque esta sintaxis es posible, preferimos la otra basada en el uso del objeto prototype porque en general será más segura y eficiente.

Aplicado al ejemplo de código que venimos viendo podríamos utilizar este código. Escríbelo en tu editor y comprueba los resultados:

```
window.onload = function () {
	var elementosInput = document.getElementsByTagName('input'); //Elementos input
	var elementosSelect = document.getElementsByTagName('select');
	var arrayNodos = Array.prototype.slice.call(elementosInput);
	arrayNodos.push([].slice.call(elementosSelect));
	alert ('Tenemos un número de elementos input ó select en el formulario: '+ arrayNodos.length);
}
```

El resultado esperado es que se muestre por pantalla: <<Tenemos un número de elementos input ó select en el formulario: 10>>

Hemos usado en un caso Array.prototype y en otro [] para poner un ejemplo de uso, aunque en realidad preferiremos usar Array.prototype:

```
window.onload = function () {
	var elementosInput = document.getElementsByTagName('input'); //Elementos input
	var elementosSelect = document.getElementsByTagName('select');
	var arrayNodos = Array.prototype.slice.call(elementosInput);
	arrayNodos.push(Array.prototype.slice.call(elementosSelect));
	alert ('Tenemos un número de elementos input ó select en el formulario: '+ arrayNodos.length);
}
```

ARRAYS.FROM

Aún nos queda una alternativa para resolver la conversión de objetos tipo array-like en arrays. El método Array.from. Este método crea un objeto de tipo array a partir de un objeto iterable u objeto de tipo array-like. Su uso habitual será con sintaxis similar a esta:

```
var divs = document.querySelectorAll('div');
var arrayDivs = Array.from(divs);
```

Este método parece el más sencillo, sin embargo puede tener importantes problemas y no ser soportado por diferentes navegadores donde puede generar mensajes de error (tipo Uncaught TypeError: undefined is not a function). Por ello recomendamos usar alguno de los métodos vistos anteriormente, que nos evitan problemas de compatibilidad entre navegadores.

RESUMEN

Vamos a tratar de hacer un resumen que aclare ideas. Los aspectos clave que hemos visto son:

a) Existen algunos objetos que son similares a los arrays pero que no son exactamente arrays. Estos objetos se suelen denominar objetos tipo array-like. Entre estos objetos tenemos los NodeList o listas de nodos devueltos por los métodos populares de acceso al DOM como document.getElementById y similares.

b) Los objetos array-like tienen algunas propiedades y métodos de los arrays, pero no todos ellos (por ejemplo pueden carecer de los métodos push, no pueden ser recorridos con foreach, etc.). Si tratamos de aplicar ciertos métodos de los arrays a objetos array-like podemos obtener errores del tipo: Object #<NodeList> has no method. No podemos usar un objeto array-like como si fuera un array, pero en circunstancias concretas sí podemos transformar los objetos array-like en arrays verdaderos para realizar ciertas manipulaciones u operaciones.

c) Los NodeList son objetos de tipo array-like que contienen lo que se denomina live Nodes: colecciones de elementos que automáticamente se actualizan si se modifica el DOM. Podemos decir que esta ventaja de actualización automática es importante, y que por ello en algunas ocasiones nos interesa trabajar con NodeList en lugar de con arrays, a pesar de que por otro lado tengan la desventaja de tener menos funcionalidad.

EJERCICIO

Examina el siguiente código y responde a las cuestiones que se muestran a continuación:

```
var arr = [];
var divs = document.querySelectorAll('div');
for(var i = divs.length; i--; arr.unshift(divs[i]));
```

a) ¿Qué cometido cumple este código?

b) Aplicando la idea que podemos extraer de este código, modifica el código que hemos visto como ejemplo en esta entrega para crear un array con los elementos input y select del formulario y mostrar por pantalla el número de elementos input y select existentes en el formulario.

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

Próxima entrega: CU01181E

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