APRENDERAPROGRAMAR.COM

BIND JAVASCRIPT . FUNCIÓN ENTRE PARÉNTESIS QUE LA ENVUELVEN. FUNCTION STATEMENT REQUIRES A NAME (CU01178E)

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

Fecha revisión: 2029

Resumen: Entrega nº78 del Tutorial básico "JavaScript desde cero".

Autor: César Krall

BIND JAVASCRIPT

El método bind de los objetos Function en JavaScript tiene distintas aplicaciones que nos van a hacer más cómoda la programación. Tiene similitudes (y diferencias) con los métodos call y apply. Hemos pospuesto el estudio de bind hasta este momento precisamente para tener conocimientos suficientes que nos permitan valorar lo que hace este método.

Nos planteamos querer obtener los índices de los elementos dentro de los formularios que sean de tipo input y type text, es decir, cajas de texto para entrada de datos en un formulario. Los formularios existentes en el código html los obtenemos con var formularios = document.forms; donde formularios[0] será el primer formulario, formularios[1] el segundo formulario y así sucesivamente

Con formularios[0].elements[0] accedemos al primer elemento dentro del formulario, con formularios[0].elements[1] accedemos al segundo elemento dentro del formulario y así sucesivamente.

Escribe este código y comprueba sus resultados (activa primero la consola del 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"> label{display:block;margin:5px;}</style>
<script type="text/javascript">
window.onload = function () {
	var formularios = document.forms;
	for (var i=0; i<formularios.length;i++){
		for (var j=0; j<formularios[i].elements.length; j++){
			if (formularios[i].elements[j].type=='text' ) {
				formularios[i].elements[j].addEventListener('click', dimeLosIndices(i,j));	}
		}
	}
}
function dimeLosIndices (indiceI, indiceJ) {console.log('Has pulsado en el inputbox cuyos índices son i='+indiceI+' ,
j='+indiceJ);}
</script></head>
<body><div id="cabecera"><h2>Cursos aprenderaprogramar.com</h2><h3>Ejemplos JavaScript</h3></div>
<form name ="formularioContacto" method="get" action="accion1.html">
<h2>Formulario de contacto</h2>
<label>Nombre:<input id="nombreFormContacto" type="text" name="nombre" /></label>
<label>Apellidos:<input id="apellidosFormContacto" type="text" name="apellidos" /></label>
<label><input id ="botonEnvio1" type="submit" value="Enviar"></label></form>
<form name ="formularioReclamacion" method="get" action="accion2.html">
<h2>Formulario de reclamación</h2>
<label>Motivo reclamación:<input id="motivoFormReclama" type="text" name="motivo" /></label>
<label>Fecha del hecho:<input id="fechaFormReclama" type="text" name="fecha" /></label>
<label><input id="botonEnvio2" type="submit" value="Enviar"></label>
</form></body></html>
```

El resultado esperado es que se muestre por consola lo siguiente:

Has pulsado en el inputbox cuyos índices son i=0 , j=0
Has pulsado en el inputbox cuyos índices son i=0 , j=1
Has pulsado en el inputbox cuyos índices son i=1 , j=0
Has pulsado en el inputbox cuyos índices son i=1 , j=1

Ya sabemos que esto ocurre porque en lugar de pasar una referencia de función en el addEventListener estamos invocando la ejecución de la función.

Si modificamos la línea del addEventListener y escribimos esto:

```
formularios[i].elements[j].addEventListener('click', function(){dimeLosIndices(i,j);});
```

El resultado es que pulsemos sobre el inputbox que pulsemos siempre se muestra: "Has pulsado en el inputbox cuyos índices son i=2 , j=3" ¿Por qué?

Porque los índices i=2 y j=3 son los últimos índices que toman las variables en el bucle. En todos los inputbox el comportamiento es: cuando se pulsa, se dispara el evento click, y con el evento click se ejecuta la función dimeLosIndices pasándole los parámetros i, j actuales, que son una vez finalizada la ejecución del bucle, i=2, j=3.

No hemos resuelto lo que queríamos, ya que queremos “ligar” a cada inputbox una función manejadora de evento que se dispare con sus propios índices, no con los índices existentes al final del bucle.

CLOSURES Y ¿UNA FUNCIÓN ENVUELTA EN PARÉNTESIS?

Podemos resolverlo con closures. Si no recuerdas los closures y su significado lee las anteriores entregas del curso. Escribe este código y comprueba los resultados (activa primero la consola del 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">
label{display:block;margin:5px;}
</style>

<script type="text/javascript">
window.onload = function () {
	var formularios = document.forms;
	for (var i=0; i<formularios.length;i++){
		for (var j=0; j<formularios[i].elements.length; j++){
			if (formularios[i].elements[j].type=='text' ) {
```

```
                                        (function(){
                                        var closureI =i;
                                        var closureJ =j;
                                        formularios[i].elements[j].addEventListener('click', function(){
                                        dimeLosIndices(closureI,closureJ);});
                                    })(); //Creamos un ámbito con la función anónima y la ejecutamos con ()
                                    }

                            }
                    }
}

function dimeLosIndices (indiceI, indiceJ) {
console.log('Has pulsado en el inputbox cuyos índices son i='+indiceI+' , j='+indiceJ);
}
</script>
</head>
<body><div id="cabecera"><h2>Cursos aprenderaprogramar.com</h2><h3>Ejemplos JavaScript</h3></div>

<form name ="formularioContacto" method="get" action="accion1.html">
<h2>Formulario de contacto</h2>
<label>Nombre:<input id="nombreFormContacto" type="text" name="nombre" /></label>
<label>Apellidos:<input id="apellidosFormContacto" type="text" name="apellidos" /></label>
<label><input id ="botonEnvio1" type="submit" value="Enviar"></label>
</form>
<form name ="formularioReclamacion" method="get" action="accion2.html">
<h2>Formulario de reclamación</h2>
<label>Motivo reclamación:<input id="motivoFormReclama" type="text" name="motivo" /></label>
<label>Fecha del hecho:<input id="fechaFormReclama" type="text" name="fecha" /></label>
<label><input id="botonEnvio2" type="submit" value="Enviar"></label>
</form></body></html>
```

El resultado es que ahora sí se mostrarán correctamente los índices. Por ejemplo si pulsamos sobre el primer inputbox del primer formulario se mostrará: "Has pulsado en el inputbox cuyos índices son i=0 , j=0"

En el código anterior tenemos algo interesante además de los closures, que ya hemos explicado en apartados anteriores del curso: una función envuelta en paréntesis con una sintaxis como esta:

(function () { //Contenido de la función

} () **)**

Bien, con function () { ... } () lo que estamos haciendo es invocar la ejecución de la función inmediatamente después de haberla creado. ¿Pero qué hacen los paréntesis exteriores?

Los paréntesis exteriores están ahí indicándole al intérprete JavaScript que ese fragmento de código es una expresión y no una declaración de función. Una expresión se ejecuta dando lugar a una

equivalencia en código, mientras que una declaración de función implica definir un nombre para una función que puede ser posteriormente invocada.

En el contexto en el que estamos no podemos declarar una función (daría lugar a un error del tipo SyntaxError: function statement requires a name), pero sí podemos usar una expresión utilizando una función para lograr nuestros objetivos.

BIND JAVASCRIPT

El uso de funciones anónimas y closures en el código anterior deriva de la imposibilidad de pasar parámetros a referencias de funciones: si pasamos parámetros, invocamos la función. Sin embargo las últimas versiones de JavaScript incorporaron un método para los objetos de tipo Function que permite ligar parámetros a referencias de funciones sin necesidad de ejecutar la función: el método bind. Este método puede que no sea reconocido por algunos navegadores antiguos.

El método bind recuerda en cierta manera a los métodos call y apply que ya hemos estudiado. La sintaxis que emplearemos será habitualmente:

```
var copiaFunc = nombreDeLaFuncion.bind(objetoQueSeráThis, opcArg1, opcArg2, ...);
```

El método bind crea una copia de la función original que será ejecutada siempre que se invoque con el objeto this que se haya especificado y con los argumentos que se hayan especificado. Es importante reseñar que bind, a diferencia de call y apply, no da lugar a la ejecución de la función.

Si en un caso dado no tiene relevancia qué objeto actuará como this, se puede pasar entonces undefined en el lugar que ocupa este parámetro.

Escribe este código y comprueba su resultado.

```
<!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() {
function suma3(x, y, z) {return x+y+z;}
var suma3Primos = suma3.bind(undefined, 2, 3, 5);
alert (suma3Primos());
alert(suma3Primos()+15);
}
</script>
</head>
<body><div id="cabecera"><h2>Cursos aprenderaprogramar.com</h2><h3>Ejemplos JavaScript</h3></div>
<div style="color:blue; margin:20px;" id ="pulsador" onclick="ejemplo()">Pulsa aquí</div>
</body>
</html>
```

El resultado esperado es que se muestre por pantalla: 10, 25.

Fíjate que escribimos suma3Primos() con los paréntesis finales porque suma3Primos es una función. Decimos que es una función ligada (binded) a la función suma3 a través del método bind.

Dado que bind no ejecuta la función, sino que crea una copia en la cual "fijamos" qué objeto actuará como this y cuáles serán los parámetros con los que será invocada la copia de la función, podemos escribir el código para mostrar los índices de los inputs de formularios en un documento HTML de una forma más simple que la que vimos anteriormente (basada en closures y funciones anónimas, un tanto engorrosa en su definición).

Escribe este código y comprueba sus resultados (activa la consola del navagador si no la tienes activada):

```
<!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"> label{display:block;margin:5px;}</style>
<script type="text/javascript">
window.onload = function () {
	var formularios = document.forms;
	for (var i=0; i<formularios.length;i++){
		for (var j=0; j<formularios[i].elements.length; j++){
			if (formularios[i].elements[j].type=='text' ) {
				formularios[i].elements[j].addEventListener('click', dimeLosIndices.bind(this,i,j));

			}

		}
	}
}

function dimeLosIndices (indiceI, indiceJ) {
console.log('Has pulsado en el inputbox cuyos índices son i='+indiceI+' , j='+indiceJ);
}
</script>
</head>
<body><div id="cabecera"><h2>Cursos aprenderaprogramar.com</h2><h3>Ejemplos JavaScript</h3></div>

<form name ="formularioContacto" method="get" action="accion1.html">
<h2>Formulario de contacto</h2>
<label>Nombre:<input id="nombreFormContacto" type="text" name="nombre" /></label>
<label>Apellidos:<input id="apellidosFormContacto" type="text" name="apellidos" /></label>
<label><input id ="botonEnvio1" type="submit" value="Enviar"></label>
</form>

<form name ="formularioReclamacion" method="get" action="accion2.html">
<h2>Formulario de reclamación</h2>
<label>Motivo reclamación:<input id="motivoFormReclama" type="text" name="motivo" /></label>
<label>Fecha del hecho:<input id="fechaFormReclama" type="text" name="fecha" /></label>
<label><input id="botonEnvio2" type="submit" value="Enviar"></label>
</form>
</body></html>
```

El resultado es el mismo que obtuvimos antes: se muestran correctamente los índices. Por ejemplo si pulsamos sobre el primer inputbox del primer formulario se mostrará: "Has pulsado en el inputbox cuyos índices son i=0 , j=0"

Al escribir dimeLosIndices.bind(this,i,j) como argumento de addEventListener estamos creando una copia de la función (función ligada) con su objeto this y parámetros específicos para cada elemento input en el que introducimos un manejador de eventos.

Este código es más claro que el basado en closures y uso de funciones anónimas, por lo tanto es preferible.

RESUMEN BIND JAVASCRIPT

El uso de bind tiene distintas aplicaciones en JavaScript y en general nos facilita la programación. Normalmente lo usaremos para mantener una referencia al objeto que queremos que actúe como this, para mantener referencia a parámetros asociados a una función, o para ambas cosas.

En muchos casos nos puede ahorrar el escribir funciones anónimas y usar closures, lo que a la larga se traduce en un código más limpio y legible. Sin embargo, bind no es una varita mágica que haya que usar para resolverlo todo. Hay que buscar un equilibrio: usar, no abusar, de este método.

EJERCICIO

Escribe este código, ejecútalo y responde a las preguntas que aparecen a continuación.

```
function conversor(toUnit, factor, offset, input) {
    offset = offset || 0;
    return [((offset+input)*factor).toFixed(2), toUnit].join(" ");
}

var milesToKm = conversor.bind(undefined, 'km', 1.60936, 0);
var poundsToKg = conversor.bind(undefined, 'kg', 0.45460, 0);
var farenheitToCelsius = conversor.bind(undefined, 'gradosC',0.5556, -32);

alert(milesToKm(10));
alert(poundsToKg(2.5));
alert(farenheitToCelsius(98));
```

a) ¿Cuántas funciones ligadas (copias parametrizadas) de la función conversor se crean en este código?

b) ¿Qué es lo que devuelve la función conversor?

c) ¿Qué objeto actúa como this en la función milesToKm?

d) ¿Qué tarea cumple y con qué fórmula trabaja la función milesToKm?

e) ¿Qué tarea cumple y con qué fórmula trabaja la función poundsToKg?

f) ¿Qué resultado devuelve milesToKm(10) y qué significa este resultado?

g) ¿Qué resultado devuelve poundsToKg(2.5) y qué significa este resultado?

h) ¿Qué resultado devuelve farenheitToCelsius(98) y qué significa este resultado?

i) ¿Qué significado tiene la expresión offset = offset || 0;?

j) ¿Cuál es la finalidad del uso de toFixed(2)?

k) ¿Por qué crees que se usa el parámetro offset en la función conversor?

l) Modifica el código para ampliar la información que se nos muestra: introduce un parámetro fromUnit en la función conversor de modo que el resultado nos informe del dato de origen, sus unidades, y el dato convertido y sus unidades.

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

Próxima entrega: CU01179E

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