APRENDERAPROGRAMAR.COM

FUNCIONES MANEJADORAS DE EVENTOS Y ADDEVENTLISTENER CON PARÁMETROS. THIS.STYLE IS UNDEFINED (CU01177E)

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

Fecha revisión: 2029

Resumen: Entrega nº77 del Tutorial básico "JavaScript desde cero".

Autor: César Krall

ADDEVENTLISTENER CON PARÁMETROS

Vamos a seguir estudiando formularios, pero vamos a detenernos en ver cómo podemos utilizar el método addEventListener con parámetros. Con lo que hemos estudiado anteriormente sobre ámbito de variables, closures, etc. vamos a ser capaces de resolverlo, aunque no es sencillo y merece la pena estudiarlo paso a paso.

Nos planteamos querer añadir respuesta a eventos cuando el usuario hace click sobre un inputbox de un formulario.

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>Ejemplo1 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;
        formularios['formularioContacto'].elements['nombreFormContacto'].addEventListener('click', cambiaColor);
        formularios['formularioContacto'].elements['apellidosFormContacto'].addEventListener('click', cambiaColor);
}
function cambiaColor () {this.style.backgroundColor='yellow'; }
</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" maxlength="4"/></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 cuando hagamos click sobre un inputbox del primer formulario, éste pase a tener color de fondo amarillo.

Es interesante ver cómo dentro de la función cambiaColor, la referencia this nos devuelve el objeto HTML que dispara el evento. ¿Por qué? Se ha explicado en anteriores entregas del curso, pero lo recordamos aquí: this usada dentro de una función manejadora de evento nos devuelve el nodo de tipo

Element definido por las etiquetas HTML que reciben el evento. Y en este caso cambiaColor es una función manejadora del evento click para los input de los formularios.

Pero el código planteado no es muy eficiente: si quisiéramos hacer esto mismo para todos los inputbox de todos los formularios y tuviéramos muchos, tendríamos que escribir mucho código a mano.

Además expresiones como formularios['formularioContacto'].elements['apellidosFormContacto'] no nos generan mucha seguridad porque ¿qué ocurre si se cambian los nombres asociados a los formularios o los campos? El código dejaría de funcionar.

Por ello veremos vamos a intentar buscar una solución más genérica (más abstracta). Pero primero repasaremos algunos conceptos.

FUNCIONES MANEJADORAS DE EVENTOS CON NINGÚN O UN PARÁMETRO

Recordamos ahora cosas ya explicadas anteriormente en el curso pero que conviene repasar. Una función manejadora de un evento recibe "de forma automática" un objeto de tipo Event. Este objeto es un argumento que se pasa automáticamente cuando se invoca la función al dispararse el evento. El nombre de este argumento es el que nosotros decidamos, es decir, podemos llamarlo evobject, eventoDatos, miEvento u objetoRepresentaEvento, etc. y para poder usarlo hemos de declararlo como parámetro de la función.

Ejemplo:

```
function cambiaColor (elEvento) {
alert('Detectado evento de tipo: '+elEvento.type);
this.style.backgroundColor='yellow';
}
```

En el ejemplo anterior, el parámetro el evento se recibe siempre que la función se invoque como respuesta a un evento. Con el código que hemos visto previamente y esta función manejadora del evento, cuando se hiciera click en el inputbox se mostraría por pantalla << Detectado evento de tipo: click>> y seguidamente el color del cuadro del inputbox pasaría a ser amarillo.

¿Podría una función manejadora de evento invocarse sin ser como respuesta a un evento? Sí, aunque hay que tener en cuenta cómo funciona JavaScript a este respecto.

Si estamos usando la función <<function cambiaColor ()>> que carece de parámetros significa que el objeto tipo Event que se envía automáticamente cuando sucede el evento no es recuperable ¿Por qué? Porque para poder recuperarlo necesitamos especificar un parámetro en la definición de la función que represente al objeto Event que se envía automáticamente. En un caso así podemos invocar la función de un modo como este: <<cambiaColor.call(document.body);>> dando como resultado que la función se ejecutará tomando como this al objeto document.body.

En cambio si estamos usando la función <<function cambiaColor (elEvento)>> y hacemos una invocación como <<cambiaColor.call(document.body);>> podemos obtener un error del tipo 'elEvento is undefined'. Este error aparece si tratamos de usar el parámetro (que es el Evento que se envía

automáticamente en segundo plano) cuando en este caso no ha existido evento, sino simplemente una invocación directa para la ejecución de la función.

Podríamos usar un código como este para diferenciar si la función se ejecuta como respuesta al evento o por invocación directa desde el código:

```
function cambiaColor (elEvento) {
if(elEvento){ alert('Detectado evento de tipo: '+elEvento.type); }
this.style.backgroundColor='yellow';
}
```

De esta manera, si no se recibe el parámetro, no se intenta hacer uso de él.

FUNCIONES MANEJADORAS DE EVENTOS CON MÁS DE UN PARÁMETRO

Supongamos que queremos enviar a la función manejadora del evento un parámetro que represente el color de fondo que debe adquirir el elemento.

Podríamos pensar en algo como esto (mala idea):

```
function cambiaColor (elColor) {
alert(elColor);
if(elColor) {this.style.backgroundColor=elColor;}
else {this.style.backgroundColor='yellow'; alert(this);}
}
```

Esta idea falla porque no tiene en cuenta una cosa: si la función se ejecuta como respuesta a un evento, el primer argumento (no explícito, pero existente) es el objeto Event asociado al evento. Con un código como el anterior tendríamos este resultado:

Si la función se invoca desde código con una sintaxis como cambiaColor.call(document.body, 'pink'); el resultado es: << Argumento recibido: pink>>

Si la función se invoca como respuesta a un evento el resultado es: <<Argumento recibido: [object MouseEvent]>>

Si pretendemos que esta sea una función manejadora de eventos a la que le podamos pasar un parámetro como el color deseado, deberemos escribirla siguiendo esta idea:

```
function cambiaColor (elEvento, elColor) {
if(elEvento){ alert('Detectado evento de tipo: '+elEvento.type); }
if(elColor){ alert('Argumento recibido: '+elColor); }
if(elColor) {this.style.backgroundColor=elColor;}
else {this.style.backgroundColor='yellow';}
}
```

Si quisiéramos invocar de forma directa esta función, tendríamos que hacerlo incluyendo dos parámetros porque la definición de la función así lo requiere. Pero si no hay evento, ¿qué ponemos como primer parámetro? Tendremos que decidirlo, pero por ejemplo podemos simplemente pasar la cadena vacía. La función la podríamos invocar directamente desde código con una sintaxis como esta:

cambiaColor.call(document.body, '', 'pink');

Aquí vemos que el argumento correspondiente al objeto Event, dado que no hay evento, lo enviamos como cadena vacía. La función, si es que va a usarse de esta manera, tendría que tener previsto el tratamiento oportuno.

Posiblemente en nuestro código las funciones manejadoras de eventos no van a ser invocadas directamente, pero el ejemplo anterior nos ha servido para comprender mejor la lógica a aplicar con este tipo de funciones.

BUSCANDO UNA SOLUCIÓN MÁS GENÉRICA PARA MEJORAR NUESTRO CÓDIGO

Volvemos a nuestro código con dos formularios, sobre el que estamos tratando de hacer que el color de fondo de las cajas de los inputbox cambie cuando hagamos click sobre ellos.

Vamos a tratar de buscar una solución más genérica (más abstracta) que la vista antes.

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>Ejemplo2 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', cambiaColor);
			}
		}
	}
}

function cambiaColor () {
this.style.backgroundColor='yellow';
}
</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" maxlength="4"/></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 ahora todas las cajas de texto de los inputbox adquieren color de fondo amarillo cuando pulsamos sobre ellos. Además, hemos definido que se recorran todos los elementos de todos los formularios y se añada el escuchador de evento correspondiente, y el código seguirá funcionando aunque cambien los atributos id ó name de los elementos input.

Ahora nos vamos a plantear lo siguiente: queremos que el color de fondo de un inputbox se convierta en amarillo si es el primer, tercer, quinto, séptimo, etc. elemento dentro del formulario, o que se convierta en verde si es el segundo, cuarto, sexto, octavo, etc. elemento dentro del formulario.

Podemos plantearlo de dos maneras:

a) Desde la función cambiaColor, recuperar de alguna manera la posición en que se encuentra el inputbox del formulario. No vamos a estudiar esta opción.

b) Pasarle a la función de manejo del evento un parámetro que le diga si el elemento es impar o par dentro del formulario. Vamos a tratar de resolver esta cuestión porque es un buen ejemplo de paso de parámetros a una función manejadora de eventos.

Inicialmente podría pensarse en una solución de este tipo:

```
if (formularios[i].elements[j].type=='text' && j%2==0 ) {
                                        formularios[i].elements[j].addEventListener('click', cambiaColorPonAmarillo);
                                }
                                else {
                                formularios[i].elements[j].addEventListener('click', cambiaColorPonNaranja);
                                }
```

Pero esto no es interesante: ¿qué ocurriría si quisiéramos poner 10 colores distintos? ¿Tendríamos que crear 10 funciones con 10 nombres distintos? ¿Y si en un momento dado queremos cambiar los colores?

Esto sería poco útil. Nuestra idea es pasar un parámetro a la función manejadora del evento con el color.

Podríamos pensar en redefinir la función cambiaColor como function cambiaColor (elEvento, elColor) { ... } y escribir algo como esto. Pruébalo y comprueba los resultados:

```
                if (formularios[i].elements[j].type=='text' && j%2==0 ) {
                        formularios[i].elements[j].addEventListener('click', cambiaColor('', 'yellow'));
                }
                else {      formularios[i].elements[j].addEventListener('click', cambiaColor('', 'orange')); }
```

Pero esto tiene varios problemas (activa la consola del navegador para ver los mensajes de error si no la tienes activada): pasamos un argumento evento vacío, ¿por qué si aquí estamos con una función manejadora de eventos? No tiene sentido. Otro problema es que addEventListener espera una referencia a una función, mientras que el código anterior ejecuta la función, lo que es distinto.

Para definir correctamente una referencia a una función tendríamos que definir una función anónima.

Nos vamos aproximando a una solución si planteamos esto, aunque todavía no funciona. Prueba este código y comprueba qué sucede:

```
<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' && j%2==0 ) {
			formularios[i].elements[j].addEventListener('click', function() {cambiaColor('', 'yellow');});
		}
		else {	formularios[i].elements[j].addEventListener('click', function(){cambiaColor('', 'orange');});
	}
		}
	}
}

function cambiaColor (elEvento, elColor) { alert(elEvento + '***'+elColor + ' - ' + this);
this.style.backgroundColor=elColor;
}
```

¿Qué ocurre con este código? En primer lugar que dentro de la función anónima estamos creando un ámbito nuevo, y this ya no es el elememento html que recibe el evento. ¿Qué es entonces this? Como siempre que this no está definido en un ámbito, es el objeto window. Al invocar this.style.backgroundColor=elColor; obtenemos un error del tipo this.style is undefined porque al objeto window no le podemos aplicar estilos (ya que no es un elemento HTML). Además estamos perdiendo el objeto Event al pasar un argumento vacío. Activa la consola si no la tienes activada para poder ver los mensajes de error.

STOP FOR A MINUTE

Lo que estamos explicando es un tanto engorroso, pero llegar a comprender todo lo que estamos discutiendo es interesante de cara a la comprensión de JavaScript. Si vienes siguiendo el curso desde el principio, deberías ser capaz de seguir estas explicaciones. Si estás siguiendo el curso y te has perdido, vuelve atrás y relee las explicaciones y haz pruebas con el código con calma. Incluso déjalo para mañana, siempre es útil descansar y retomarlo al día siguiente. Si después de releer esta entrega se te sigue atragantando, te recomendamos escribir una consulta en los foros aprenderaprogramar.com.

CONTINUAMOS

Repasamos conceptos: this es el elemento HTML que nos interesa dentro de la función anónima que es ahora la manejadora del evento, pero ya no lo es en la función cambiaColor porque ésta ya no es la manejadora del evento, sino una función invocada por la manejadora del evento.

Ahora la función manejadora del evento es la función anónima, por lo que si quisiéramos pasar el evento tendríamos que hacer algo así:

```
<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' && j%2==0 ) {
			formularios[i].elements[j].addEventListener('click', function(elEvento)
{cambiaColor(elEvento, 'yellow');});
		}
		else {	formularios[i].elements[j].addEventListener('click',
function(elEvento){cambiaColor(elEvento, 'orange');});	}

		}
	}
}

function cambiaColor (elEvento, elColor) {
this.style.backgroundColor=elColor;
}
</script>
```

Ahora estamos pasando el evento, pero todavía no hemos resuelto el problema de haber perdido la referencia al this.

Podríamos pensar en intentar algo como esto:

```
<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' && j%2==0 ) {
			formularios[i].elements[j].addEventListener('click', function(elEvento) {
			cambiaColor(elEvento, 'yellow', formularios[i].elements[j]);});
		}
		else {	formularios[i].elements[j].addEventListener('click', function(elEvento){
		cambiaColor(elEvento, 'orange',formularios[i].elements[j]);});	}
		}
	}
}
function cambiaColor (elEvento, elColor, quienEsThis) {
quienEsThis.style.backgroundColor=elColor;
}
</script>
```

Ahora nos enfrentamos a un error de tipo <<TypeError: formularios[i] is undefined>> porque la función anónima define un nuevo ámbito, independiente puesto que es un argumento de addEventListener y no una función anidada dentro de la función de respuesta a window.onload.

Finalmente vamos a llegar a una solución válida que consiste en pasarle el this a la función cambiaColor. 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">
<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' && j%2==0 ) {
				formularios[i].elements[j].addEventListener('click', function(elEvento) {
				cambiaColor(elEvento, 'yellow', this);});

			}
			else {	formularios[i].elements[j].addEventListener('click', function(elEvento){
			cambiaColor(elEvento, 'orange', this);});		}
		}
	}
}

function cambiaColor (elEvento, elColor, quienEsThis) {
quienEsThis.style.backgroundColor=elColor;
}
</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" maxlength="4"/></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>
```

¿PODEMOS MEJORAR?

En el código que se ha propuesto como solución, cada vez que se añade un manejador de eventos a un elemento HTML estamos creando una función anónima. Esto puede resultar ineficiente. Más adelante estudiaremos cómo podemos mejorar este código para hacerlo más eficiente.

EJERCICIO

Crea un código HTML donde tengas un formulario con seis inputbox de texto que servirán para pedir al usuario Nombre, Apellidos, Correo electrónico, Teléfono, Domicilio y País. Crea el código JavaScript para que los elementos 1, 4, 7, 10 etc. del formulario tomen color de fondo amarillo cuando el usuario pulse sobre ellos. Los elementos 2, 5, 8, 11, etc. del formulario deberán tomar color de fondo azul claro cuando el usuario pulse sobre ellos. Los elementos 3, 6, 9, 12, 15, etc. del formularios deberán tomar color de fondo verde claro cuando el usuario pulse sobre ellos.

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

Próxima entrega: CU01178E

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