

Blog sobre Java EE

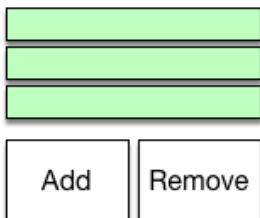
Estás aquí: [Inicio/JavaScript Core/JavaScript Reactivo/El concepto de JavaScript Proxy](#)

El concepto de JavaScript Proxy

15 noviembre, 2017 por [Cecilio Álvarez Caules](#) — [Deja un comentario](#)



El concepto de **JavaScript proxy** es un concepto cada día más importante ya que muchos frameworks se basan en el uso de proxies. **¿Qué es un proxy?** un proxy no es ni más ni menos que un intermediario entre un programa y el objeto al cual queremos invocar un método o cambiar una propiedad. Vamos a ver un ejemplo sencillo utilizando **JavaScript** y **jQuery**. Supongamos que tenemos una lista de personas **con dos botones**. El primer botón añade una nueva persona a la lista, el segundo botón la elimina.



La forma más sencilla de abordar algo de este estilo es usando un array de JavaScript y generando dos eventos con jQuery.

```
1  <body>
2
3
4  <table>
5  </table>
6
7
8  <input type="button" value="add" id="add" />
9  <input type="button" value="remove" id="remove" />
10 </body>
```

Vamos a añadir el javascript necesario:

```
1  $(document).ready(function() {
```

Este sitio web utiliza cookies para que usted tenga la mejor experiencia de usuario. Si continúa navegando está dando su consentimiento para la aceptación de las mencionadas cookies y la aceptación de nuestra [política de cookies](#), pinche el enlace para mayor información.

[plugin cookies](#)
[ACEPTAR](#)

```

12     lista.push({
13         nombre: "pedro",
14         edad: 50
15     });
16     crearTabla(lista);
17 });
18 $("#remove").click(function() {
19     lista.splice(lista.indexOf({
20         nombre: "pedro",
21         edad: 50
22     }), 1);
23     crearTabla(lista);
24 });
25 })
26 }

```

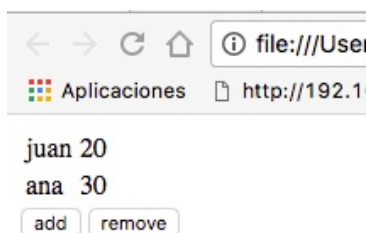
Invocamos a la función crear tabla:

```

1  function crearTabla(lista) {
2      $("table").empty();
3      lista.forEach(function(elemento) {
4          $("table").append("
5
6          <tr>
7
8          <td>" + elemento.nombre + "</td>
9
10         <td>" + elemento.edad + "</td>
11
12         </tr>
13
14         ");
15     });
16 }
17 }

```

La tabla se crea y la presentamos en pantalla:

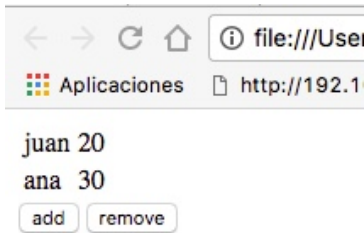


Pulsamos añadir y añadirá un nuevo elemento:

Este sitio web utiliza cookies para que usted tenga la mejor experiencia de usuario. Si continúa navegando está dando su consentimiento para la aceptación de las mencionadas cookies y la aceptación de nuestra [política de cookies](#), pinche el enlace para mayor información.

[plugin cookies](#)[ACEPTAR](#)

Si pulsamos al botón de borrar volveremos a la situación anterior.



El código funciona correctamente . Sin embargo tenemos un problema cada vez que hacemos **una modificación en la lista tenemos que acordarnos de volver a dibujar la tabla**. ¿Podríamos hacer lo mismo de una forma **más transparente**?. La respuesta es sí, podemos crearnos un proxy con Javascript que controle el acceso a nuestra lista.

```
1  function reactiva(miarray) {
2
3      var nueva =new Proxy(miarray, {
4          set: function(target, property, value) {
5              target[property] = value;
6              crearTabla();
7              return true;
8          }
9      })
10     return nueva;
11 }
12
13 listaReactiva=reactiva(lista);
14
```

Acabamos de añadir un proxy a nuestra lista. El código es difícil de entender en un primer momento. **¿Para que sirve el método set con target, property y value?**. Set hace referencia a los métodos de asignación del array. El método push se puede considerar de asignación ya que cambia un valor de una posición del array. Target hace referencia al objeto que manipulamos y property y value a cada propiedad y valor.

Este sitio web utiliza cookies para que usted tenga la mejor experiencia de usuario. Si continúa navegando está dando su consentimiento para la aceptación de las mencionadas cookies y la aceptación de nuestra [política de cookies](#), pinche el enlace para mayor información.

[plugin cookies](#)[ACEPTAR](#)

Lo que estamos implementando en nuestro código es la funcionalidad de que cada vez que añadamos eliminamos o modificamos un elemento del array el método crearTabla() se invoque de forma automática. A partir de ahora trabajaremos con la **listaReactiva y no la lista**. Podemos modificar los métodos de add y remove de nuestros botones ya que la lista se vuelve a dibujar sola a través de la funcionalidad añadida por el proxy.

```
1  $("#add").click(function() {
2
3      listaReactiva.push({
4          nombre: "pedro",
5          edad: 50
6      });
7
8  });
9
10
11  $("#remove").click(function() {
12      listaReactiva.splice(lista.indexOf({
13          nombre: "pedro",
14          edad: 50
15      }), 1);
16  });
```

Haremos lo mismo al crear la tabla:

```
1  crearTabla(listaReactiva);
```

Ahora ya no hace falta invocar a crearTabla en cada uno de los eventos

JavaScript Proxy y su funcionamiento.

¿Cómo funciona esto? . Realmente lo que estamos es generando un Proxy sobre nuestro array. Este proxy hace de intermediario entre el programa y el array.

Este sitio web utiliza cookies para que usted tenga la mejor experiencia de usuario. Si continúa navegando está dando su consentimiento para la aceptación de las mencionadas cookies y la aceptación de nuestra [política de cookies](#), pinche el enlace para mayor información.

[plugin cookies](#)[ACEPTAR](#)[crea tabla](#)

De tal forma que cada vez que necesitamos realizar una operación **de push o de splice la tabla se regenera de forma transparente**. Muchos frameworks usan el concepto de JavaScript proxy o intermediario para añadir funcionalidad dinámica a las aplicaciones.

Otros artículos relacionados:

1. [Utilizando JavaScript template String](#)
2. [WebPack y la gestión de dependencias en JavaScript](#)
3. [JavaScript High Order Functions y su manejo](#)
4. [JavaScript y Proxies](#)

[in](#)**0**
COMPART

Archivada en: [JavaScript Reactivo](#)

Leave a Reply

Be the First to Comment!

Notify of



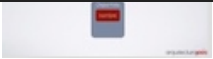
Start the discussion

BUSCAR

Este sitio web utiliza cookies para que usted tenga la mejor experiencia de usuario. Si continúa navegando está dando su consentimiento para la aceptación de las mencionadas cookies y la aceptación de nuestra [política de cookies](#), pinche el enlace para mayor información.

[ACEPTAR](#)

[plugin cookies](#)



Java JDBC



Servlets



REST 101:
THE BEGINNER'S
GUIDE TO USING AND
TESTING RESTFUL
APIs



 **SoapUI**

[GET THE EBOOK](#)

SMARTBEAR

POPULAR

[Maven Parent POM y uso de librerías](#)

Este sitio web utiliza cookies para que usted tenga la mejor experiencia de usuario. Si continúa navegando está dando su consentimiento para la aceptación de las mencionadas cookies y la aceptación de nuestra [política de cookies](#), pinche el enlace para mayor información.

[plugin cookies](#)[ACEPTAR](#)[Spring y Hello World](#)[El concepto de JavaScript Proxy](#)[Spring Boot WAR sin Microservicios](#)[Java Herencia vs Interfaces](#)[Eclipse Git , Repositorios locales y remotos](#)

CONTACTO

contacto@arquitecturajava.com

LO MAS LEIDO

[¿Qué es Spring Boot?](#)[Usando Java Session en aplicaciones web](#)[Java Constructores this\(\) y super\(\)](#)[El concepto de JavaScript Proxy](#)[Java Iterator vs ForEach](#)[Maven Parent POM y uso de librerías](#)[Ejemplo de JPA , Introducción \(I\)](#)[¿Cuales son las certificaciones Java?](#)[¿Qué es Gradle?](#)[Introducción a Servicios REST](#)[Ejemplo de Java Singleton \(Patrones y ClassLoaders\)](#)[Usando el patron factory](#)[Uso de Java Generics \(I\)](#)[REST JSON y Java](#)[Mis Libros](#)[Comparando java == vs equals](#)[Spring MVC Configuración \(I\)](#)[Java Override y encapsulación](#)

Este sitio web utiliza cookies para que usted tenga la mejor experiencia de usuario. Si continúa navegando está dando su consentimiento para la aceptación de las mencionadas cookies y la aceptación de nuestra [política de cookies](#), pinche el enlace para mayor información.

[plugin cookies](#)[ACEPTAR](#)

Copyright © 2017 · [eleven40 Pro Theme](#) en [Genesis Framework](#) · [WordPress](#) · [Acceder](#)