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.

```
<body>


<table>
</table>


<input type="button" value="add" id="add" />
<input type="button" value="remove" id="remove" />
</body>
```

Vamos a añadir el javascript necesario:

```
$(document).ready(function() {
    var lista = [{
      nombre: "juan",
      edad: 20
    }, {
      nombre: "ana",
      edad: 30
    }];
    crearTabla(lista);

    $("#add").click(function() {
      lista.push({
        nombre: "pedro",
        edad: 50
      });
      crearTabla(lista);
    });
    $("#remove").click(function() {
      lista.splice(lista.indexOf({
        nombre: "pedro",
        edad: 50
      }), 1);
      crearTabla(lista);
    });
    })
    }
```

Invocamos a la función crear tabla:

```
 function crearTabla(lista) {
 $("table").empty();
```

```
 lista.forEach(function(elemento) {
 $("table").append("

<tr>

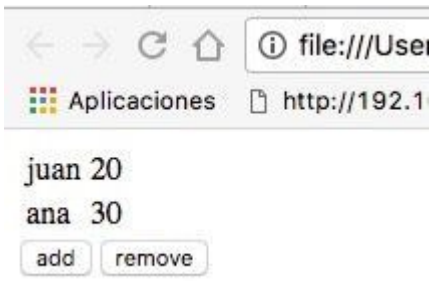
<td>" + elemento.nombre + "</td>


<td>" + elemento.edad + "</td>

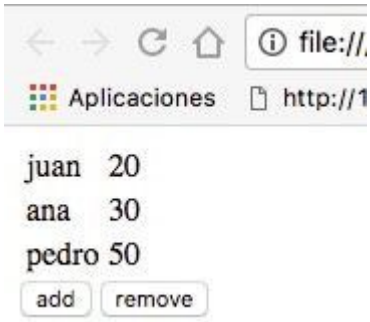
</tr>

");
 });
 }
```

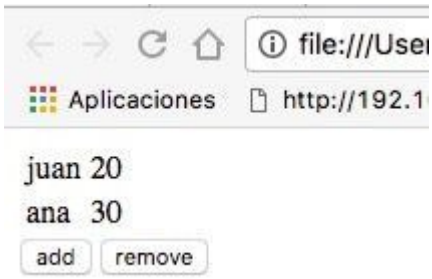
La tabla se crea y la presentamos en pantalla:



Pulsamos añadir y añadirá un nuevo elemento:



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.

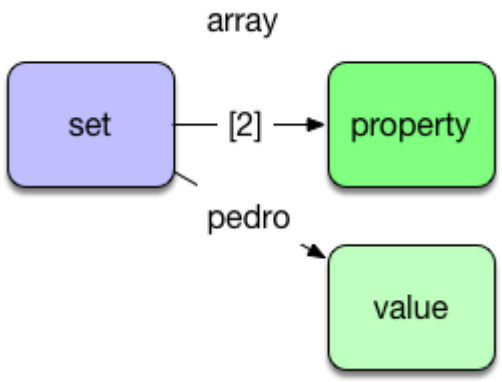
```
function reactiva(miarray) {

      var nueva =new Proxy(miarray, {
        set: function(target, property, value) {
          target[property] = value;
          crearTabla();
          return true;
        }
```

```
    })
    return nueva;
  }

  listaReactiva=reactiva(lista);
```

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.



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.

```
$("#add").click(function() {
```

```
    listaReactiva.push({
      nombre: "pedro",
      edad: 50
    });

});


$("#remove").click(function() {

    listaReactiva.splice(lista.indexOf({
      nombre: "pedro",
      edad: 50
    }), 1);
```

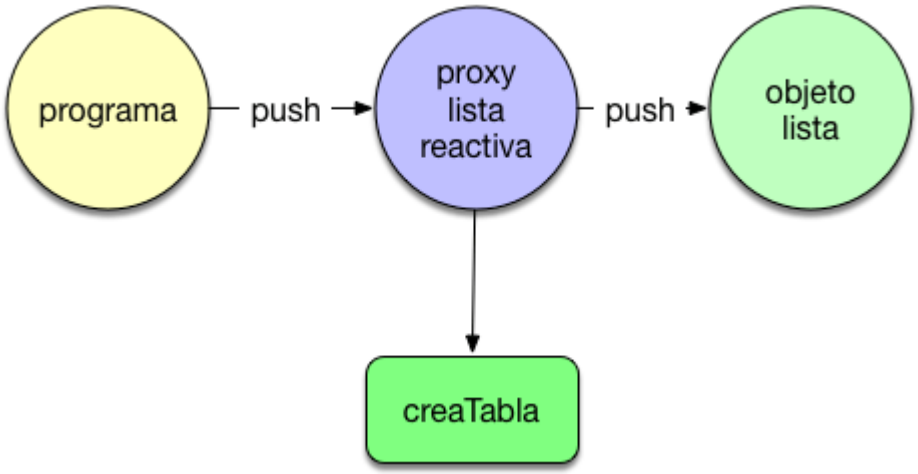
Haremos lo mismo al crear la tabla:

```
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.



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