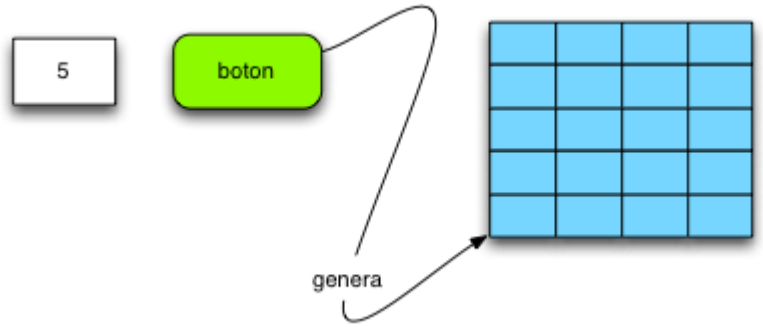
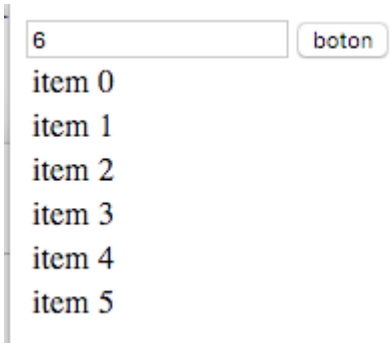
¿Qué es JavaScript Memorization y para que sirve?. El concepto de Memorization hace referencia a como podemos en JavaScript modificar el comportamiento de una función para que sus resultados se cacheen. Esto en un principio puede sonar algo raro pero vamos a ver un ejemplo que nos ayude a clarificarlo. Supongamos que disponemos de un botón que genera dinámicamente una tabla de X filas a partir de una caja de texto.



Se trata de una operación sencilla y en principio no tenemos problemas con ella:



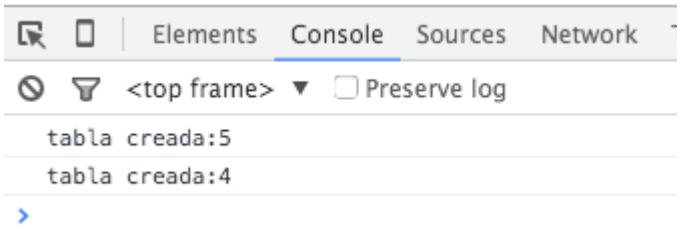
El código que vamos a utilizar para que esta operación funcione es :

```
<html>
<head>
<script src="underscore.js" type="text/javascript">
```

```
</script>

<script src="jquery-1.12.1.js" type="text/javascript">
</script>

<script type="text/javascript" >
function tablaDinamica(filas) {

$("table").remove();
var tabla= $("<table/>");
for (var i=0;i<filas;i++) {

tabla.append("
<tr>
<td>item</td>
<td>"+i+"</td>
</tr>

");
}
console.log ("tabla creada");
return tabla;
}

$(document).ready(function() {

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

tablaDinamica(parseInt($("#caja").val())).appendTo("body");

});
```

```
});
</script>
</head>
<body>
<input type="text" id="caja"/>
<input type="button" id="boton" value="boton" />
</body>
</html>
```

Ahora bien esta operación podría ser mas compleja y que a JavaScript le costara crear las filas . ¿Tendríamos un problema? . La respuesta es que depende. ¿De qué? pues de si el número de filas a construir varía mucho . En el caso de que no sea así y que alguien nos diga que las tablas son de 1 a 10 filas como mucho hay solución. Podemos usar el concepto de JavaScript Memorization. En estos momentos cada vez que creamos una tabla con X filas un mensaje nos aparece en la consola.



JavaScript Memorization

Con JavaScript Memorization podemos añadir un nuevo comportamiento a nuestra función, de tal forma que cada vez que se invoque se cachee el resultado. Para ello usaremos una de las librerías más versátiles que hay en JavaScript “Underscore.js”.

```
<html>
```

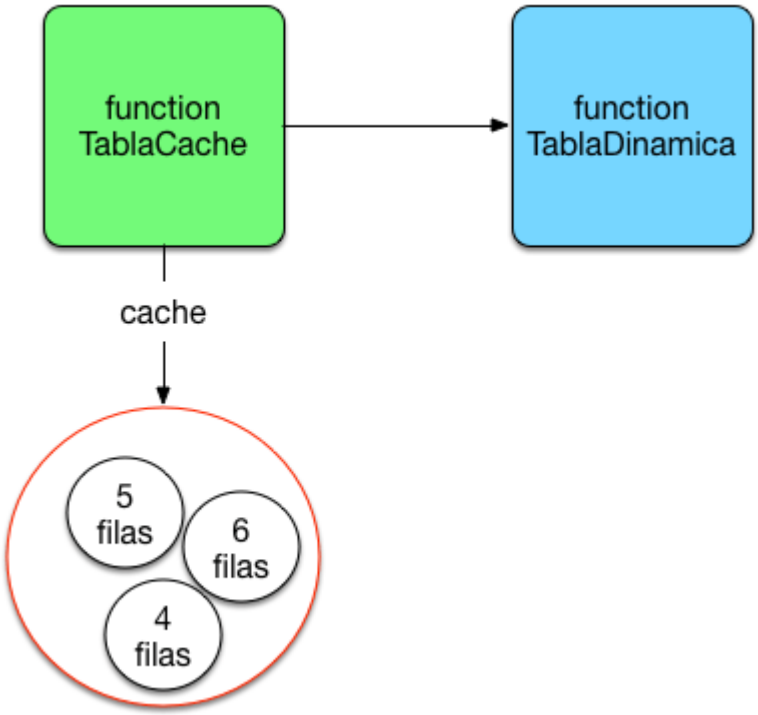
```
<head>
<script src="underscore.js" type="text/javascript">
</script>

<script src="jquery-1.12.1.js" type="text/javascript">
</script>

<script type="text/javascript" >
function tablaDinamica(filas) {
$("table").remove();
var tabla= $("<table/>");
for (var i=0;i<filas;i++) {

tabla.append("
<tr>
<td>item</td>
<td>"+i+"</td>
</tr>

");

}
console.log ("tabla creada:"+ filas);
return tabla;
}
$(document).ready(function() {

var tablaCache= _.memoize(function (filas) {

return tablaDinamica(filas);
});
```

```
$("#boton").click (function() {
tablaCache(parseInt($("#caja").val())).appendTo("body");;

});

});
</script>
</head>
<body>
<input type="text" id="caja"/>
<input type="button" id="boton" value="boton" />
</body>
</html>
```

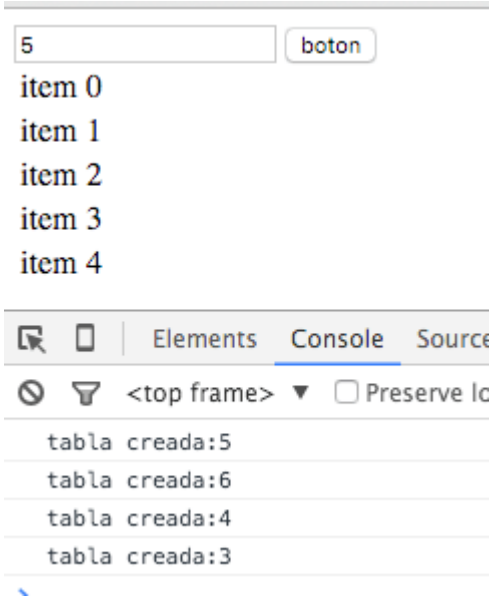
El código es casi idéntico lo único que cambia es que construimos una nueva función que se llama tablaCache.

```
var tablaCache= _.memoize(function (filas) {

return tablaDinamica(filas);
});
```

Lo que hace es cachear las invocaciones a la función de tablaDinámica.



A partir de ahora la primera vez que generemos la tabla para un número de filas concreto la tabla se generará. Una vez generada la primera vez quedará cacheada y no consumirá recursos en las siguientes invocaciones.



Como se puede observar acabamos de generar la tabla de 5 y ya no aparece en la consola. Se ha usado la cache ya que anteriormente ya se había solicitado esta tabla.

Otros artículos relacionados:LocalStorage y LowDash , JavaScript Console , LoDash y Filtrados