Programación Funcional

Funciones de Arrays: map, reduce y filter

Laboratorio de Computación III

UTNFRA – Técnico Superior en Programación --

map 1/3

La función de arrays map se utiliza cuando necesitamos modificar cada elemento del mismo de la misma manera. Por ejemplo elevar al cuadrado cada elemento de un array de números, recibir el nombre de una lista de usuarios, o correr una expresión regular a un array de Strings.

Cuando se llama a map, este ejecuta un Callback en cada elemento dentro del array, retornando un nuevo array con los valores que el Callback retorna.

map, pasa tres parámetros al Callback:

1. El elemento actual del array
2. El índice en el array del elemento actual
3. El array entero que estamos mapeando

map 2/3

Supongamos que tenemos una app que mantiene un array de mascotas. Cada mascota es un objeto de nombre mascota, cada uno con las propiedades nombre y edad.

```
var mascotas =[
    {
        'nombre': 'Bobby',
        'edad': 3
    },
    {
        'nombre': 'Pancho',
        'edad': 1
    },
    {
        'nombre': 'Alfred',
        'edad': 2
    },

];
```

map 3/3

Digamos que queremos crear un nuevo array con el nombre de las mascotas solamente. Usando map podemos escribir:

```
16
17
18
19  var nombres_Mascotas = mascotas.map(function(mascota, indice, array){
20      return mascota.nombre;
21  })
22
23
24
```

Coloque los parámetros índice y array para recordar que están ahí si los necesitas pero como no se usan se pueden omitir y el código seguiría funcionando normalmente.

El Callback que pasamos al map debe tener una sentencia return especifica, o el map devolverá un array lleno de undefined.

filter 1/2

La función de arrays filter. Hace exactamente a lo que suena. Toma un array, y filtra los elementos no deseados.

Al igual que map, filter esta disponible para cualquier array y se le pasa un Callback como primer parámetro. Filter ejecuta ese Callback para cada uno de los elementos del array y devuelve un nuevo array que contiene solo los elementos para los que la devolución del Callback devuelve true.

filter, pasa tres parámetros al Callback:

1. El elemento actual del array
2. El índice en el array del elemento actual
3. El array que estamos filtrando

filter 2/2

Vamos a revisar nuestro ejemplo de mascotas. En lugar de obtener los nombres de las mascotas, quiero las mascotas mayores a 1 año.

```
25
26
27  var MascotasMayoresA1 = mascotas.filter(function(mascota){
28      return mascota.edad > 1;
29  })
30
31
```

Con filter debemos asegurarnos que el callback va a retornar un valor booleano

reduce 1/2

map crea un nuevo array mediante la transformación de casa elemento de un array, de forma individual. Filter crea un nuevo array, eliminando los elementos que no le pertenecen. Reduce, por otro lado, toma todos los elementos de un array, y los reduce en un solo valor.

reduce, pasa cuatro parámetros al Callback:

1. El valor actual
2. El valor anterior
3. El índice actual
4. El array que estamos reduciendo

reduce 2/3

Supongamos que quiero obtener la suma de las edades de las mascotas.

```
32
33
34  var sumaEdades = mascotas.reduce(function(anterior, actual){
35      return  anterior + actual.edad;
36  })
37
38
```

reduce 3/3

O la mayor de las edades…

```
40
41
42  var mayorEdad = mascotas.reduce(function(anterior, actual){
43      if(anterior  > actual.edad)
44          return anterior;
45          else
46          return actual.edad;
47
48  }, 0);
49
50  console.log("Mayor edad " + mayorEdad);
51
52
53
54
```

El segundo parámetro de reduce se toma como valor anterior