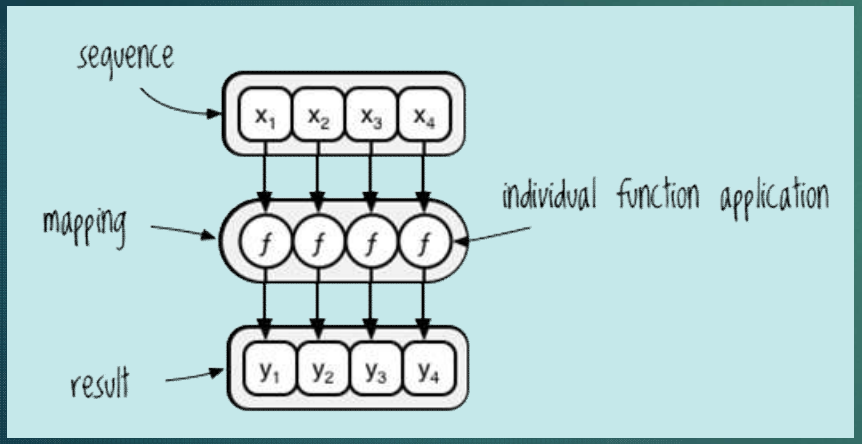
Arrays

Map, reduce and filter

Objetivos de la clase

- Comprender los métodos de array avanzados, como filter, reduce y map
- Distinguir entre forEach y map y cuándo usar cada uno de ellos.
- Saber cómo funciona reduce y filter
- Implementar filter, map y reduce en ejemplos reales

Introducción



Hemos visto algunos métodos básicos de array hasta este punto. Uno de ellos es forEach, aunque lo usamos muy a menudo, a veces queremos hacer más que iterar a través del array.

Para comenzar, tengamos en cuenta que los métodos .map(), .filter() y .reduce() NO modifican el array original, es decir, no mutan el array original, sino que crean un nuevo array.

Introducción

.map() es muy similar a forEach, excepto por una distinción importante:

El método .forEach() en realidad no devuelve nada (undefined). Simplemente llama a una función proporcionada en cada elemento de su array.

El método .map() también llamará a una función proporcionada en cada elemento del array. La diferencia es que .map() utiliza valores de retorno y en realidad devuelve un nuevo array del mismo tamaño.

map()

```
const array = [1, 2, 3];

// The new array has the same number of elements as the original array
// BUT each element is mapped to the value*2

// ES5:
var newArray = array.map(function(number){
    return number * 2;
})

console.log(newArray); // <== [ 2, 4, 6 ]

// ES6:
const newArray = array.map(number => number * 2);

console.log(newArray); // <== [ 2, 4, 6 ]
```

reduce()

.reduce() es un método que convierte una lista de valores en un valor.

Echemos un vistazo a .reduce() en JavaScript.

```
// Keep in mind that accumulator and currentValue are placeholders, can be
anything

// ES5:

array.reduce(function(accumulator, currentValue){
    return accumulator + currentValue;
})

// ES6:
array.reduce((accumulator, currentValue) => accumulator + currentValue)
```

reduce()

```
const numbers = [2, 4, 6, 8];

// ES5:
// we are keeping var just to remind you that you can still use it
var total = numbers.reduce(function(accumulator, currentValue){
    console.log("accumulator is: ", accumulator,
        "and current value is: ", currentValue);
    return accumulator + currentValue;
});

// ES6:
const total = numbers.reduce((accumulator, currentValue) => {
    console.log("accumulator is: ", accumulator,
    "and current value is: ", currentValue);
return accumulator + currentValue;
});


console.log("total is: ", total);

// accumulator is:  2 and current value is:  4
// accumulator is:  6 and current value is:  6
// accumulator is:  12 and current value is:  8
// total is:  20
```

reduce()

.reduce() funciona para iterar sobre arrays, independientemente del tipo de datos que esté dentro de los arrays. Vimos ejemplos con números, y aquí hay uno con strings:

```
const words = ["This", "is", "one", "big", "string"];

// ES5:
var bigString = words.reduce(function(sum, word){
    return sum + word;
});

// ES6:
const bigString = words.reduce((sum, word) => sum + word);

console.log(bigString); // <== Thisisonebigstring
```

reduce()

Algunas veces necesitamos hacer reduce() mientras estamos usando objetos. Lo hacemos estableciendo un valor predeterminado.

```
// ES5:

const ages = people.reduce(function(sum, person){
    return sum + person.age;
}, 0);

// ES6:
const ages = people.reduce((sum, person) => {
    return sum + person.age;
}, 0);

console.log(ages); // <== 142
```

filter()

itera a través de un array y crea un nuevo array con todos los elementos que pasan la condición que establecemos.

Con ES5:

```
var numbers = [1, 2, 3, 4, 5, 6];

var evenNumbers = numbers.filter(function(number){
    return number % 2 === 0;
});

console.log(evenNumbers); // <== [ 2, 4, 6 ]
```

Lo mismo pero en la sintaxis de ES6 sería:

```
const numbers = [1, 2, 3, 4, 5, 6];

const evenNumbers = numbers.filter(number => number % 2 === 0);

console.log(evenNumbers); // <== [ 2, 4, 6 ]
```

Recursos adicionales:

- MDN forEach
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/forEach
- MDN Map
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/map
- MDN Reduce
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/Reduce
- MDN Filter
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/filter
- Dan Martensen - Map, Reduce and Filter
https://danmartensen.svbtle.com/javascripts-map-reduce-and-filter

Resumiendo: Aprendimos…

- ✓ cómo iterar sobre arrays
- ✓ a realizar diferentes tareas como mapping, reducing o filtering.
- ✓ manipular los arrays para obtener los datos que queremos es una de las tareas más comunes en la programación