Métodos de arrays

DigitalHouse>

Certified Tech Developer
The Ultimate Degree

Índice

1 | map()

.map()

Este método recibe una función como parámetro (callback).

Recorre el array y devuelve un nuevo array modificado.

Las modificaciones serán aquellas que programemos en nuestra función de callback.

```
{}
array.map(function(elemento){
   // definimos las modificaciones que queremos
   // aplicar sobre cada elemento del array
});
```

Ahora explicamos paso a paso el siguiente código

```
const numeros = [2, 4, 6];
const numerosMultiplicados =
numeros.map(function(num){
    // Multiplicamos por 2 cada número
    return num * 2;
});

console.log(numerosMultiplicados); // [4,8,12]
```

{código}

```
const numeros = [2, 4, 6];
const numerosMultiplicados =
numeros.map(function(num){
    // Multiplicamos por 2 cada número
    return num * 2;
});

console.log(numerosMultiplicados); //
[4,8,12]
```

Declaramos la variable **numeros** y almacenamos un array con tres números.

{código}

```
const numeros = [2, 4, 6];
const numerosMultiplicados =
numeros.map(function(num){
    // Multiplicamos por 2 cada número
    return num * 2;
});

console.log(numerosMultiplicados);
// [4,8,12]
```

Aplicamos el método **map()** al array de números.

{código}

```
const numeros = [2, 4, 6];
const numerosMultiplicados =
numeros.map(function(num){
    // Multiplicamos por 2 cada número
    return num * 2;
});

console.log(numerosMultiplicados); //
[4,8,12]
```

Al método **map()** pasamos una función como parámetro (callback).

Esa función, a su vez, recibe un parámetro —puede tener el nombre que queramos—.

Este va a representar a cada elemento de nuestro array, en este caso, un número.

{código}

```
let numeros = [2, 4, 6];
let numerosMultiplicados =
numeros.map(function(num){
    // Multiplicamos por 2 cada número
    return num * 2;
});

console.log(numerosMultiplicados); //
[4,8,12]
```

Definimos el comportamiento interno que va a tener la función.

La función se va a ejecutar 3 veces, una por cada elemento de este array, y a cada uno lo va a multiplicar por 2.

{código}

```
let numeros = [2, 4, 6];
let numerosMultiplicados =
numeros.map(function(num){
    // Multiplicamos por 2 cada número
    return num * 2;
});

console.log(numerosMultiplicados); //
[4,8,12]
```

En la variable **numerosMultiplicados** vamos a almacenar el nuevo array que nos va a devolver el método map.

{código}

```
const numeros = [2, 4, 6];
const numerosMultiplicados =
numeros.map(function(num){
    // Multiplicamos por 2 cada número
    return num * 2;
});

console.log(numerosMultiplicados ); //
[4,8,12]
```

Mostramos por consola la variable **numerosMultiplicados** que contiene un **nuevo array** con la misma cantidad de elementos que el original, pero con valores modificados.

2 | filter()

.filter()

Este método también recibe una función como parámetro.

Recorre el array y **filtra** los elementos según una condición que exista en el callback.

Devuelve un **nuevo array** que contiene únicamente los elementos que hayan cumplido con esa condición. Es decir, que nuestro nuevo array puede contener menos elementos que el original.

```
{}
array.filter(function(elemento){
    // definimos la condición que queremos utilizar
    // como filtro para cada elemento del array
});
```

{código}

```
var edades = [22, 8, 17, 14, 30];
var mayores = edades.filter(function(edad){
    return edad > 18;
});

console.log(mayores); // [22, 30]
```

{código}

```
var edades = [22, 8, 17, 14, 30];
var mayores = edades.filter(function(edad){
    return edad > 18;
});

console.log(mayores); // [22, 30]
```

Declaramos la variable **edades** y almacenamos un array con cinco números.

{código}

```
var edades = [22, 8, 17, 14, 30];
var mayores = edades.filter(function(edad){
    return edad > 18;
});

console.log(mayores); // [22, 30]
```

Aplicamos el método **filter** al array de edades.

{código}

```
var edades = [22, 8, 17, 14, 30];
var mayores = edades.filter(function(edad){
    return edad > 18;
});

console.log(mayores); // [22, 30]
```

Al método **filter()** le pasamos una función como parámetro (callback).

Esa función, a su vez, recibe un parámetro —puede tener el nombre que queramos—.

Este va a representar a cada elemento de nuestro array, en este caso, una edad.

{código}

```
var edades = [22, 8, 17, 14, 30];
var mayores = edades.filter(function(edad){
    return edad > 18;
});

console.log(mayores); // [22, 30]
```

Definimos el comportamiento interno que va a tener esa función.

La función se va a ejecutar 5 veces, una por cada elemento de este array, y los va a filtrar según la condición que definimos: que las edades sean mayores a 18.

Esto quiere decir que las que no cumplan con la condición, serán excluidas.

{código}

```
var edades = [22, 8, 17, 14, 30];
var mayores = edades.filter(function(edad){
    return edad > 18;
});

console.log(mayores); // [22, 30]
```

En la variable **mayores** almacenamos el array que nos devuelve el método filter.

{código}

```
var edades = [22, 8, 17, 14, 30];
var mayores = edades.filter(function(edad){
   return edad > 18;
});

console.log(mayores); // [22, 30]
```

Mostramos por consola la variable **mayores** que tiene el array con los elementos que cumplieron con la condición establecida.

3 | reduce()

.reduce()

Este método recorre el array y devuelve un **único valor.**

Recibe un callback que se va a ejecutar sobre cada elemento del array. Este, a su vez, recibe cuatro argumentos: un **acumulador, elemento actual, índice actual y array** que esté recorriendo.

```
{}
array.reduce(function(acumulador, elemento, índice, array){
    // definimos el comportamiento que queremos
    // implementar sobre el acumulador y el elemento
    // podemos indicar el valor inicial que por defecto es 0
}, 0);
```

{código}

```
var nums = [5, 7, 16];
var suma = nums.reduce(function(acum, num){
    return acum + num;
});

console.log(suma); // 28
```

{código}

```
var nums = [5, 7, 16];
var suma = nums.reduce(function(acum, num){
    return acum + num;
});

console.log(suma); // 28
```

Declaramos la variable edades y almacenamos un array con tres números.

{código}

```
var nums = [5, 7, 16];
var suma = nums.reduce(function(acum, num){
    return acum + num;
});

console.log(suma); // 28
```

Le aplicamos el método **reduce()** al array de números.

{código}

```
var nums = [5, 7, 16];
var suma = nums.reduce(function(acum, num){
    return acum + num;
});

console.log(suma); // 28
```

Al **reduce()** le pasamos una función como parámetro (callback).

Esa función recibe dos parámetros —pueden tener el nombre que queramos—. El primero va a representar el acumulador; el segundo, el elemento que esté recorriendo en ese momento, en este caso un número.

{código}

```
var nums = [5, 7, 16];
var suma = nums.reduce(function(acum, num){
    return acum + num;
});

console.log(suma); // 28
```

Definimos el comportamiento interno de la función. En este caso, queremos devolver la **suma** total de los elementos.

El acumulador irá almacenando el resultado y por cada iteración sumará el elemento actual.

{código}

```
var nums = [5, 7, 16];
var suma = nums.reduce(function(acum, num){
    return acum + num;
});

console.log(suma); // 28
```

En la variable suma almacenamos lo que devuelva el método **reduce()** al aplicarlo al array numeros.

{código}

```
var nums = [5, 7, 16];
var suma = nums.reduce(function(acum, num){
    return acum + num;
});

console.log(suma); // 28
```

Mostramos por consola la variable **suma** que tiene la suma total de los números del array.

4 | forEach()

.forEach()

La finalidad de este método es iterar sobre un array.

Recibe un callback como parámetro y, a diferencia de los métodos anteriores, este **no retorna nada.**

```
{}
array.forEach(function(elemento){
    // definimos el comportamiento que queremos
    // implementar sobre cada elemento
});
```

{código}

```
var paises = ['Argentina', 'Cuba', 'Perú'];

paises.forEach(function(pais){
   console.log(pais);
});
```

{código}

```
var paises = ['Argentina', 'Cuba', 'Peru'];
```

Declaramos la variable **paises** y almacenamos un array con tres países.

```
paises.forEach(function(pais){
    console.log(pais);
});
```

{código}

```
var paises = ['Argentina', 'Cuba', 'Peru'];

paises.forEach(function(pais){
    console.log(pais);
});
```

Le aplicamos el método **forEach()** al array de países.

{código}

```
var paises = ['Argentina', 'Cuba', 'Peru'];

paises.forEach(function(pais){
    console.log(pais);
});
```

Al método **forEach()** le pasamos una función como parámetro (callback).

Esa función recibe un parámetro, que va a representar a cada elemento del array, en este caso, a cada país.

{código}

```
var paises = ['Argentina', 'Cuba', 'Peru'];

paises.forEach(function(pais){
  console.log(pais);
});
```

Definimos el comportamiento interno de la función.

En este caso, queremos mostrar por consola a cada país.

{código}

```
var paises = ['Argentina', 'Cuba', 'Perú'];
```

```
paises.forEach(function(pais){
  console.log(pais);
});
```

El método **forEach()** imprimirá por consola todos los elementos del array.

```
Argentina
Cuba
Perú
```

DigitalHouse>