wawiwa

Map

El método map() crea un nuevo arreglo poblado con los resultados de llamar una función dada en cada elemento en el arreglo.

```
const arr = [1, 4, 9, 16];

// pasa una función a map
const map1 = arr.map(x => x * 2);
console.log(map1);
// resultado esperado: Array [2, 8, 18, 32]
```

wawiwa

Map

Map llama a la función callbackFn dada una vez por cada elemento en un arreglo, en orden y construye un nuevo arreglo de los resultados. Se invoca callbackFn solamente para los índices del arreglo que tiene un valor asignado (incluyendo undefined).
El siguiente código toma un arreglo de números y crea un arreglo con la raíz cuadrada de los números del primer arreglo.

```
// Mapeando un arreglo de números a un arreglo de raices cuadradas
const numbers = [1, 4, 9];

const roots = numbers.map(num => Math.sqrt(num));
// roots is now [1, 2, 3]
// numbers is still [1, 4, 9]
```

wawiwa

Map

Mapeando un arreglo de número usando una función con un argumento.
El siguiente código muestra cómo funciona `map` cuando se utiliza con una función que requiere un argumento. El argumento se asignará automáticamente a partir de cada elemento del arreglo a medida que `map` recorre el arreglo original.

```
const numbers = [1, 4, 9];
const doubles = numbers.map(num =>  num * 2);

// doubles ahora es [2, 8, 18]
// numbers sigue siendo [1, 4, 9]
```

wawiwa

Filter

El método filter() method crea un nuevo arreglo con los elementos que pasan el test implementado en la función dada.

```
const words = ['spray', 'limit', 'elite',
    'exuberant', 'destruction', 'present'];


const result = words.filter(word => word.length > 6);


console.log(result);
// resultado esperado: Array ["exuberant", "destruction", "present"]
```

wawiwa

Filter

filter() llama una función callbackFn dada una vez por cada elemento del arreglo, y construye un nuevo arreglo de valores para los cuales callbackFn devuelve un valor que se evalúa como “true”. callbackFn es invocada solamente para los índices del arreglo que tienes valores asignados, no se invoca para los índices que han sido borrados o a los que nunca se les asignó un valor. Los elementos del arreglo que no son pasados al test de callbackFn son saltados y no son incluidos en el nuevo arreglo.

callbackFn se invoca con tres argumentos:

1. El valor del elemento
2. El índice del elemento
3. El arreglo que está siendo recorrido

wawiwa

Filter

El siguiente ejemplo usa filter() para crear un arreglo filtrado que tiene todos los elementos con un valor menor de 10 borrados.

```
const arr = [12, 5, 8, 130, 44];
const filtered = arr.filter(val => val >= 10);

// filtered is [12, 130, 44]
```

wawiwa

Filter

Encuentra todos los números primos en un arreglo
El siguiente ejemplo retorna todos los números primos en el arreglo:

```
const arr = [-3, -2, -1, 0, 1, 2, 3, 4,
    5, 6, 7, 8, 9, 10, 11, 12, 13];

function isPrime(num) {
    for (let i = 2; i < num; i++) {
        if (num % i == 0) return false;
    }

    return num > 1;
}

console.log(arr.filter(isPrime)); // [2, 3, 5, 7, 11, 13]
```

wawiwa

Reduce

El método reduce() ejecuta una función de reducción proporcionada por el usuario en cada elemento del arreglo, pasando el valor de retorno del cálculo en el elemento precedente. El resultado final de ejecutar el reductor en todos los elementos del arreglo es un solo valor.
El reductor recorre el arreglo elemento por elemento, en cada paso agregando el valor actual del arreglo al resultado del paso anterior, hasta que no haya más elementos que agregar.

```
const arr = [1, 2, 3, 4];
const reducer = (previousValue, currentValue) => previousValue + currentValue;

// 1 + 2 + 3 + 4
console.log(arr.reduce(reducer));
// resultado esperado: 10

// 5 + 1 + 2 + 3 + 4
console.log(arr.reduce(reducer, 5));
// resultado esperado: 15
```

wawiwa

Reduce

Si initialValue no es proporcionado entonces el método reduce va a actuar diferente para arreglos con “length” más grande que 1, igual a 1 y 0, como se muestra en el siguiente ejemplo:

```
const getMax = (a, b) => Math.max(a, b);

// callback se invoca para cada elemento en el arreglo empezando en el índice 0
[1, 100].reduce(getMax, 50); // 100
[50].reduce(getMax, 10); // 50

// callback se invoca una vez para el elemento en el índice 1
[1, 100].reduce(getMax); // 100

// callback no se invoca
[50].reduce(getMax); // 50
[].reduce(getMax, 1); // 1

[].reduce(getMax); // TypeError
```

wawiwa

Find

El método find() retorna el valor del primer elemento en el arreglo proporcionado que satisfaga la función de test. Si ningún valor satisface la función de test, se retorna undefined.

```
const arr = [5, 12, 8, 130, 44];
const found = arr.find(element => element > 10);
console.log(found);
// resultado esperado : 12
```

wawiwa

Find

El método find() ejecuta la función`callbackFn una vez por cada índice del arreglo hasta que callbackFn devuelva un valor evaluado como verdadero (truthy). Si es así, find devuelve inmediatamente el valor de ese elemento. De lo contrario, devuelve undefined.

callbackFn se invoca para cada índice del arreglo, no solo para aquellos con valores asignados. Esto significa que puede ser menos eficiente para arreglos dispersos en comparación con los métodos que solo visitan valores asignados.

Si se proporciona un parámetro thisArg a find(), se utilizará como el valor `this` dentro de cada invocación de callbackFn. Si no se proporciona, se utilizará undefined.

wawiwa

Find

Encuentra un objeto en un arreglo por una de sus propiedades

```
const inventory = [
    { name: 'apples', quantity: 2 },
    { name: 'bananas', quantity: 0 },
    { name: 'cherries', quantity: 5 }
];

function isCherries(fruit) {
    return fruit.name === 'cherries';
}

console.log(inventory.find(isCherries));
// { name: 'cherries', quantity: 5 }
```

wawiwa

Find

Usar una función de flecha y deconstruir

```
const inventory = [
    { name: 'apples', quantity: 2 },
    { name: 'bananas', quantity: 0 },
    { name: 'cherries', quantity: 5 }
];

const result = inventory.find(({ name }) => name === 'cherries');
console.log(result) // { name: 'cherries', quantity: 5 }
```

wawiwa

findindex

El método findIndex() retorna el index del primer elemento del arreglo que satisfaga la función de test. De otro modo, retorna -1, indicando que ningún elemento pasó el test.

```
function isPrime(num) {
    for (let i = 2; i < num; i++) {
        if (num % i === 0) {
            return false;
        }
    }

    return num > 1;
}

console.log([4, 6, 8, 9, 12].findIndex(isPrime)); // -1, not found
console.log([4, 6, 7, 9, 12].findIndex(isPrime)); // 2 (array[2] is 7)
```

wawiwa

findindex

El siguiente ejemplo encuentra el índex de una fruta usando función de flecha:

```
const fruits = ["apple", "banana", "cantaloupe",
    "blueberries", "grapefruit"];

const index = fruits.findIndex(fruit => fruit === "blueberries");

console.log(index); // 3
console.log(fruits[index]); // blueberries
```

wawiwa

Some

El método some() prueba si es que al menos un elemento del arreglo pasa el test implementado en la función. Retorna true si es que en el arreglo encuentra un elemento con el que la función retorna true; de otro modo, retorna false. No modifica el arreglo.

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

// checks whether an element is even
const even = val => val % 2 === 0;

console.log(arr.some(even));
// expected output: true
```

wawiwa

Some

El siguiente ejemplo prueba si algún elemento del arreglo es más grande que 10.

```
function isBiggerThan10(element) {
    return element > 10;
}

[2, 5, 8, 1, 4].some(isBiggerThan10); // false
[12, 5, 8, 1, 4].some(isBiggerThan10); // true
```

wawiwa

Funciones Some + arreglo

Probando los elementos de un arreglo usando funciones de flecha

```
[2, 5, 8, 1, 4].some(x => x > 10); // false
[12, 5, 8, 1, 4].some(x => x > 10); // true
```

wawiwa

Every

El método every() prueba si es que todos los elementos en el arreglo pasa el test implementado por la función proporcionada. Retorna un valor boolean

```
const isBelowThreshold = val => val < 40;
const arr = [1, 30, 39, 29, 10, 13];
console.log(arr.every(isBelowThreshold));
// resultado esperado: true
```

wawiwa

¿Preguntas?