//const nomes = new Array('Eduardo', 'Maria', 'Joana');

// Valor por referência

//                 0         1        2

const nomes = ['Eduardo', 'Maria', 'Joana'];

const novo = [...nomes]; // fazendo um espalhamento dos valores de nomes para dentro de novo

//const novo = nomes       // fazendo desse metodo um aponta para outro, caso um mude o outro muda tambem

nomes.pop();             // remove elemento do final array

novo.shift();            // remove do começo e realoca os indicces

nomes.push('João')       // adiciona um novo elemento ao final

novo.unshift('Wallace')  // adiciona no começo do array

console.log(nomes.slice(0, 2));            // fatia o array para qual valor deseja

console.log(nomes);

console.log(novo);

console.log(nomes.length);

const nome1 = 'Luiz Otávio Midanda';

const nomes1 = nome1.split(' ');         // transformando uma string em um array a cada espaço

console.log(nomes1);

const nomes2 = ['Luiz', 'Otávio', 'Miranda'];

const nome2 = nomes2.join(' ');          // une todos os valores de uma array em string

console.log(nome2);

* Aula 63 metodo splice

//              -5       -4        -3         -2        -1

//               0        1         2          3         4

const nomes = ['Maria', 'João', 'Eduardo', 'Gabriel', 'Júlia'];

// nomes.splice(indice, delete, elem1, elem2, elem3);

nomes.splic( qual indece começa, quantos índices quer apagar, o que irá ser adicionado)

// pop

nomes.splice(4, 1)

irá excluir à partir do índice 4, 1 elemento.

let removidos = nomes.splice(3, 2)

console.log(removidos);

aqui está excluindo à partir do índice 3, dois elementos.

Esses elementos excluídos são retornado em forma de Array

[ 'Gabriel', 'Júlia' ]

let removidos = nomes.splice(0, 3, 'Luiz');

já aqui está fazendo a adição de um elemento.

Aparti do índice 0, excluir 3 elemento e adicionar Luiz no índice 0.

let removidos = nomes.splice(3, 1, 'Luiz');

no índice 3 voce ira excluir um valor, no índice 3 elemento e adicionar o elemento luiz no lugar do índice que foi excluido.

let removidos = nomes.splice(nomes.length, 0, 'Luiz');

adicionando Luiz ao ultimo elemento, nesse caso séria melhor utilizar o push(‘Luiz’)

// pop - tira do final

nomes.splice(-1,1);

console.log(nomes);

// push - adiciona no final

nomes.splice(nomes.length ,0, 'Luiz');

console.log(nomes);

// shift - remove do começo

nomes.splice(0, 1);

console.log(nomes);

//unshift - adiciona do começo

nomes.splice(0, 1, 'Otávio');

console.log(nomes);

* Aula 63 concatenando arrays

const a1 = [1, 2, 3];

const a2 = [4, 5, 6];

const a3 = a1 + a2;

console.log(a3);

1,2,34,5,6

Daria essa forma errada além de transformala em string, deixando de ser arrays

const a3 = a1.concat(a2)

esse séria o modo certo de fazer

[ 1, 2, 3, 4, 5, 6 ]

const a3 = [...a1, , 'Luiz', ...a2, ...[7, 8, 9]] // OUTRO CERTO

outro método também certo

* Aula 65 filtrando Array

// Filter, map, reduce

// Filter -> filtrar os arrays -> sempre retorna um array, com a mesma quantidade ou menos

// Map    -> Modificar os array

// Reduce -> Reduzir o array em um único valor

// Retorna os números maiores que 10

const numeros = [5, 50, 80, 1, 2, 3, 5, 8, 7, 11, 15, 22, 27];

const numerosFiltrados = numeros.filter();

console.log(numerosFiltrados);

o filter vai interar sobre cada valor do array números.

// Retorna os números maiores que 10

//               0   1   2  3  4  5  6  7  8  9   10  11  12

const numeros = [5, 50, 80, 1, 2, 3, 5, 8, 7, 11, 15, 22, 27];

function callbackFilter(valor, indice, array) {

    if(valor > 10) {

        return true;

    } else {

        return false;

    }

};

const numerosFiltrados = numeros.filter(callbackFilter);

console.log(numerosFiltrados); //[ 50, 80, 11, 15, 22, 27 ]

um jeito de fazer e dessa maneira, mais longa.

function callbackFilter(valor, indice, array) {

    return valor > 10;

};

const numerosFiltrados = numeros.filter(callbackFilter);

console.log(numerosFiltrados); //[ 50, 80, 11, 15, 22, 27 ]

quando se tem um retorno de true ou false, para ficar mais legível pode se retorna direto a condição, caso numero maior que 10, ele retorna true, caso contrario retorna false.

Nessa função de filter quando chama uma função ele manda altomaticamente os argumentos de valor, índice e Array, bastando você escolher se vai usar ou não

function callbackFilter(valor) {

    return valor > 10;

};

const numerosFiltrados = numeros.filter(function(valor) {

    return valor > 10;

});

Ou então caso não vá usar essa função denovo basta chama-la direto no filter como função anônima.

const numerosFiltrados = numeros.filter((valor) => {

    return valor > 10;

});

Virou agora uma arrow function

const numerosFiltrados = numeros.filter(valor => valor > 10);

como a sua função só está retornando apenas um valor podesse excluir as chaves e o return e colocar tudo em um inha so.

const numeros = [5, 50, 80, 1, 2, 3, 5, 8, 7, 11, 15, 22, 27];

const numerosFiltrados = numeros.filter((valor, indice, array) =>{

    console.log(valor, indice, array);

    return valor > 10;

});

console.log(numerosFiltrados); //[ 50, 80, 11, 15, 22, 27 ]

15 10 [5, 50, 80, 1, 2, 3, 5, 8, 7, 11, 15, 22, 27]

22 11 [5, 50, 80, 1, 2, 3, 5, 8, 7, 11, 15, 22, 27]

27 12 [5, 50, 80, 1, 2, 3, 5, 8, 7, 11, 15, 22, 27]

Um exemplo do que ele manda

O valor, eu índice e o Array completo, ele e mandado atomaticamente no filter, você pode utilizar se quiser.

// Retorne as pessoas que tem o nome com 5 letras ou mais

// Retorne as pessoas com mais de 50 anos

// Retorne as pessoas cujo nome termina com a

const pessoas = [

    {nome: 'Luiz', idade: 62},

    {nome: 'Maria', idade: 23},

    {nome: 'Eduardo', idade: 55},

    {nome: 'Letícia', idade: 19},

    {nome: 'Rosana', idade: 32},

    {nome: 'Wallace', idade: 47}

];

const pessoasComNomeGrande = pessoas.filter(valor => valor.nome.length >= 5);

const pessoasComMaisDeCinquentaAnos = pessoas.filter(valor => valor.idade > 50);

const nomeComAUltimaLetraA = pessoas.filter(valor => valor.nome.toLowerCase().endsWith('a'));

* Aula 66 mapeando o Array

// Map    -> Modificar os array

// Dobre os números

//               0   1   2  3  4  5  6  7  8  9   10  11  12

const numeros = [5, 50, 80, 1, 2, 3, 5, 8, 7, 11, 15, 22, 27];

const numerosEmDobro = numeros.map(function(valor, indice, array) {

    console.log(valor, indice, array);

});

const numeros = [5, 50, 80, 1, 2, 3, 5, 8, 7, 11, 15, 22, 27];

const numerosEmDobro = numeros.map((valor) => valor \* 2);

console.log(numerosEmDobro);

// Para cada elemento:

// Retorne apenas uma string com o nome da pessoa

// Remova apenas a chave "nome" do objeto

// Adicione uma chave id em cada objeto

const pessoas = [

    {nome: 'Luiz', idade: 62},

    {nome: 'Maria', idade: 23},

    {nome: 'Eduardo', idade: 55},

    {nome: 'Letícia', idade: 19},

    {nome: 'Rosana', idade: 32},

    {nome: 'Wallace', idade: 47}

];

const nome = pessoas.map(valor => valor.nome);

const remove = pessoas.map(valor => {

    delete valor.nome

    return valor

})

console.log(remove);

usando o delete ele irá deletar também o campo nome do Array de pessoas.

const nome = pessoas.map(valor => valor.nome);

const remove = pessoas.map(valor => {

    return {idade: valor.idade}

})

console.log(remove);

para resolver essa situação e possível criar um novo objeto idade que irá receber o valor de idade.

Contudo ao fazer dessa maneira ao transformar em arrow function teremos um problema de chavez.

const remove = pessoas.map(valor => {idade: valor.idade});

para resolver isso temos que ulltilizar os parênteses para envolver esse campo transformando ele em uma expressão

const remove = pessoas.map(valor => ({idade: valor.idade}));

dessa maneira

const id = pessoas.map((valor, indice) => {

    valor.id = indice;

    return valor;

})

Inserindo valores, para cada valor do map, ele cria um atributo id e insere o índice e retorna o valor completo

Com o uso do map, você altera os valores do seu array original, para resolver essa situação, basta criar um novo objeto e fazer o spread dos valores e jogar nesse novo objeto criado

const nome = pessoas.map(valor => valor.nome);

const remove = pessoas.map(valor => ({idade: valor.idade}));

const id = pessoas.map((valor, indice) => {

    const novoObjeto = {...valor};

    novoObjeto.id = indice;

    return novoObjeto;

})

Aqui foi criado um novo objeto e passado para ele através do spread os valores contido em valor, depois criando um novo atributo id, que recebe o valor do seu índice em cada ação do map.