**01-foreach.js**

[Go to file](https://github.com/luxplanjay/js-22/find/06-1-%D0%BC%D0%B5%D1%82%D0%BE%D0%B4%D1%8B-%D0%BC%D0%B0%D1%81%D1%81%D0%B8%D0%B2%D0%B0)

|  |
| --- |
| /\* |
|  | \* Array.prototype.forEach(callback(currentValue, index, array), thisArg) |
|  | \* - Поэлементо перебирает оригинальный массив |
|  | \* - Ничего не возвращает |
|  | \* - Заменяет классический for, если не нужно прерывать цикл |
|  | \*/ |
|  |  |
|  | const numbers = [5, 10, 15, 20, 25]; |
|  |  |
|  | numbers.forEach(function (number) { |
|  | console.log('number', number); |
|  | }); |
|  |  |
|  | console.log(numbers); |

## ****02-map.js****

|  |
| --- |
| /\* |
|  | \* Array.prototype.map() |
|  | \* - Поэлементо перебирает оригинальный массив |
|  | \* - Не изменяет оригинальный массив |
|  | \* - Возвращает новый массив такой же длины |
|  | \*/ |
|  |  |
|  | const numbers = [5, 10, 15, 20, 25]; |
|  |  |
|  | const doubledNums = numbers.map(number => { |
|  | return number \* 3; |
|  | }); |
|  | // console.log('numbers', numbers); |
|  | // console.log('doubledNums', doubledNums); |
|  |  |
|  | const players = [ |
|  | { id: 'player-1', name: 'Mango', timePlayed: 310, points: 54, online: false }, |
|  | { id: 'player-2', name: 'Poly', timePlayed: 470, points: 92, online: true }, |
|  | { id: 'player-3', name: 'Kiwi', timePlayed: 230, points: 48, online: true }, |
|  | { id: 'player-4', name: 'Ajax', timePlayed: 150, points: 71, online: false }, |
|  | { id: 'player-5', name: 'Chelsy', timePlayed: 80, points: 48, online: true }, |
|  | ]; |
|  | console.table(players); |
|  |  |
|  | /\* |
|  | \* Получаем массив имён всех игроков |
|  | \*/ |
|  |  |
|  | const playerNames = players.map(player => player.name); |
|  | // console.log('playerNames', playerNames); |
|  |  |
|  | const playerIds = players.map(player => player.id); |
|  | // console.log('playerIds', playerIds); |
|  |  |
|  | // const res = players.map(({ name, online }) => ({ name, online })); |
|  | // console.log('res', res); |
|  |  |
|  | /\* |
|  | \* Увеличиваем кол-во поинтов каждого игрока на 10% |
|  | \*/ |
|  |  |
|  | const upatedPlayers = players.map(player => ({ |
|  | ...player, |
|  | points: player.points \* 1.1, |
|  | })); |
|  |  |
|  | // console.table(upatedPlayers); |
|  | // console.log(upatedPlayers); |
|  |  |
|  | /\* |
|  | \* Увеличиваем кол-во часов игрока по id |
|  | \*/ |
|  |  |
|  | const playerIdToUpdate = 'player-3'; |
|  |  |
|  | const updatedPlayers = players.map(player => { |
|  | if (playerIdToUpdate === player.id) { |
|  | return { |
|  | ...player, |
|  | timePlayed: player.timePlayed + 100, |
|  | }; |
|  | } |
|  |  |
|  | return player; |
|  | }); |
|  |  |
|  | // const updatedPlayers = players.map(player => |
|  | // playerIdToUpdate === player.id |
|  | // ? { ...player, timePlayed: player.timePlayed + 100 } |
|  | // : player, |
|  | // ); |
|  | // console.table(updatedPlayers); |

## ****03-filter.js****

|  |
| --- |
| /\* |
|  | \* Array.prototype.filter() |
|  | \* - Поэлементо перебирает оригинальный массив |
|  | \* - Возвращает новый массив (с элементами или пустой) |
|  | \* - Добавляет в возвращаемый массив элементы которые удовлетворяют условию коллбек-функции |
|  | \* - если коллбек вернул true элемент добавляется в возвращаемый массив |
|  | \* - если коллбек вернул false элемент НЕ добавляется в возвращаемый массив |
|  | \*/ |
|  |  |
|  | const numbers = [5, 10, 15, 20, 25]; |
|  |  |
|  | const filteredNumbers = numbers.filter(number => number < 10 || number > 20); |
|  | // console.log(filteredNumbers); |
|  |  |
|  | const players = [ |
|  | { id: 'player-1', name: 'Mango', timePlayed: 310, points: 54, online: false }, |
|  | { id: 'player-2', name: 'Poly', timePlayed: 470, points: 92, online: true }, |
|  | { id: 'player-3', name: 'Kiwi', timePlayed: 230, points: 48, online: true }, |
|  | { id: 'player-4', name: 'Ajax', timePlayed: 150, points: 71, online: false }, |
|  | { id: 'player-5', name: 'Chelsy', timePlayed: 280, points: 48, online: true }, |
|  | ]; |
|  |  |
|  | /\* |
|  | \* Получаем массив всех онлайн игроков |
|  | \*/ |
|  |  |
|  | const onlinePlayers = players.filter(({ online }) => online); |
|  | // console.table(onlinePlayers); |
|  |  |
|  | /\* |
|  | \* Получаем массив всех оффлайн игроков |
|  | \*/ |
|  |  |
|  | const offlinePlayers = players.filter(player => !player.online); |
|  | // console.table(offlinePlayers); |
|  |  |
|  | /\* |
|  | \* Получаем список хардкорных игроков с временем больше 250 |
|  | \*/ |
|  |  |
|  | const hardcorePlayers = players.filter(player => player.timePlayed > 250); |
|  | // console.table(hardcorePlayers); |

## ****04-find.js****

|  |
| --- |
| /\* |
|  | \* Array.prototype.find() |
|  | \* - Поэлементо перебирает оригинальный массив |
|  | \* - Возвращает первый элемент удовлетворяющий условию или undefined |
|  | \*/ |
|  |  |
|  | const numbers = [5, 10, 15, 20, 25]; |
|  |  |
|  | const number = numbers.find(number => number === 10); |
|  | // console.log(number); |
|  |  |
|  | const players = [ |
|  | { id: 'player-1', name: 'Mango', timePlayed: 310, points: 54, online: false }, |
|  | { id: 'player-2', name: 'Poly', timePlayed: 470, points: 92, online: true }, |
|  | { id: 'player-3', name: 'Kiwi', timePlayed: 230, points: 48, online: true }, |
|  | { id: 'player-4', name: 'Ajax', timePlayed: 150, points: 71, online: false }, |
|  | { id: 'player-5', name: 'Chelsy', timePlayed: 280, points: 48, online: true }, |
|  | ]; |
|  |  |
|  | /\* |
|  | \* Ищем игрока по id |
|  | \*/ |
|  | const playerIdToFind = 'player-3'; |
|  | const playerWithId = players.find(({ id }) => id === playerIdToFind); |
|  | // console.log(playerWithId); |
|  |  |
|  | const finPlayerById = (allPlayer, playerId) => { |
|  | return allPlayer.find(({ id }) => id === playerId); |
|  | }; |
|  |  |
|  | // console.log(finPlayerById(players, 'player-1')); |
|  | // console.log(finPlayerById(players, 'player-4')); |
|  |  |
|  | /\* |
|  | \* Ищем игрока по имени |
|  | \*/ |
|  | const playerNameToFind = 'Poly'; |
|  | const playerWithName = players.find(player => player.name === playerNameToFind); |
|  | // console.log(playerWithName); |

## ****05-every-some.js****

|  |
| --- |
| const players = [ |
|  | { id: 'player-1', name: 'Mango', timePlayed: 310, points: 54, online: true }, |
|  | { id: 'player-2', name: 'Poly', timePlayed: 470, points: 92, online: false }, |
|  | { id: 'player-3', name: 'Kiwi', timePlayed: 230, points: 48, online: true }, |
|  | { id: 'player-4', name: 'Ajax', timePlayed: 150, points: 71, online: false }, |
|  | { id: 'player-5', name: 'Chelsy', timePlayed: 280, points: 48, online: true }, |
|  | ]; |
|  |  |
|  | /\* |
|  | \* Array.prototype.every() |
|  | \* - Поэлементо перебирает оригинальный массив |
|  | \* - Возвращает true если все элементы массива удовлетворяют условию |
|  | \*/ |
|  |  |
|  | const isAllOnline = players.every(player => player.online); |
|  | // console.log('isAllOnline: ', isAllOnline); |
|  |  |
|  | /\* |
|  | \* Array.prototype.some() |
|  | \* - Поэлементо перебирает оригинальный массив |
|  | \* - Возвращает true если хотя бы один элемент массива удовлетворяет условию |
|  | \*/ |
|  | const isAnyOnline = players.some(player => player.online); |
|  | console.log('isAnyOnline: ', isAnyOnline); |
|  |  |
|  | const anyHardcorePlayers = players.some(player => player.timePlayed > 400); |
|  | console.log('anyHardcorePlayers: ', anyHardcorePlayers); |

## ****06-reduce.js****

|  |
| --- |
| \* |
|  | \* Array.prototype.reduce() |
|  | \* - Поэлементо перебирает оригинальный массив |
|  | \* - Возвращает что угодно 🤯 |
|  | \* - Заменяет всё на свете, но использовать нужно с умом |
|  | \*/ |
|  |  |
|  | const numbers = [5, 10, 15, 20, 25]; |
|  |  |
|  | const total = numbers.reduce((acc, number) => acc + number, 0); |
|  | // console.log(total); |
|  |  |
|  | // accumulator = 0 -> number = 5 -> return 0 + 5 |
|  | // accumulator = 5 -> number = 10 -> return 5 + 10 |
|  | // accumulator = 15 -> number = 15 -> return 15 + 15 |
|  | // accumulator = 30 -> number = 20 -> return 30 + 20 |
|  | // accumulator = 50 -> number = 25 -> return 50 + 25 |
|  |  |
|  | /\* |
|  | \* Считаем общую зарплату |
|  | \*/ |
|  |  |
|  | const salary = { |
|  | mango: 100, |
|  | poly: 50, |
|  | ajax: 150, |
|  | }; |
|  |  |
|  | const totalSalary = Object.values(salary).reduce( |
|  | (total, value) => total + value, |
|  | 0, |
|  | ); |
|  | // console.log(totalSalary); |
|  |  |
|  | /\* |
|  | \* Считаем общее количество часов |
|  | \*/ |
|  |  |
|  | const players = [ |
|  | { id: 'player-1', name: 'Mango', timePlayed: 310, online: false }, |
|  | { id: 'player-2', name: 'Poly', timePlayed: 470, online: true }, |
|  | { id: 'player-3', name: 'Kiwi', timePlayed: 230, online: true }, |
|  | { id: 'player-4', name: 'Ajax', timePlayed: 150, online: false }, |
|  | { id: 'player-5', name: 'Chelsey', timePlayed: 80, online: true }, |
|  | ]; |
|  |  |
|  | const totalTimePlayed = players.reduce( |
|  | (totalTime, player) => totalTime + player.timePlayed, |
|  | 0, |
|  | ); |
|  |  |
|  | // console.log(totalTimePlayed); |
|  |  |
|  | /\* |
|  | \* Считаем общую сумму товаров корзины |
|  | \*/ |
|  | const cart = [ |
|  | { label: 'Apples', price: 100, quantity: 2 }, |
|  | { label: 'Bananas', price: 120, quantity: 3 }, |
|  | { label: 'Lemons', price: 70, quantity: 4 }, |
|  | ]; |
|  |  |
|  | const totalAmount = cart.reduce( |
|  | (total, { price, quantity }) => total + price \* quantity, |
|  | 0, |
|  | ); |
|  |  |
|  | // console.log(totalAmount); |
|  |  |
|  | /\* |
|  | \* Собираем все теги из твитов |
|  | \*/ |
|  | const tweets = [ |
|  | { id: '000', likes: 5, tags: ['js', 'nodejs'] }, |
|  | { id: '001', likes: 2, tags: ['html', 'css'] }, |
|  | { id: '002', likes: 17, tags: ['html', 'js', 'nodejs'] }, |
|  | { id: '003', likes: 8, tags: ['css', 'react'] }, |
|  | { id: '004', likes: 0, tags: ['js', 'nodejs', 'react'] }, |
|  | ]; |
|  |  |
|  | const allTags = tweets.reduce((acc, tweet) => [...acc, ...tweet.tags], []); |
|  | console.log(allTags); |
|  |  |
|  | // acc = [], tweet.tags = ['js', 'nodejs'] return [...[], ...['js', 'nodejs']] |
|  | // acc = ['js', 'nodejs'] tweet.tags ['html', 'css'] |
|  | // return [...['js', 'nodejs'], ...['html', 'css']] |
|  | // ['js', 'nodejs', 'html', 'css'] |
|  |  |
|  | /\* |
|  | \* Ведём статистику тегов |
|  | \*/ |
|  | // const tagsStats = allTags.reduce((acc, tag) => { |
|  | // console.log(acc); |
|  |  |
|  | // if (acc[tag]) { |
|  | // acc[tag] += 1; |
|  |  |
|  | // return acc; |
|  | // } |
|  |  |
|  | // acc[tag] = 1; |
|  |  |
|  | // return acc; |
|  | // }, {}); |
|  |  |
|  | const tagsStats = allTags.reduce((acc, tag) => { |
|  | return { |
|  | ...acc, |
|  | [tag]: acc[tag] ? acc[tag] + 1 : 1, |
|  | }; |
|  | }, {}); |
|  | // console.log(tagsStats); |
|  |  |
|  | // если свойство с ключом tag есть. увеличить его значение на 1 |
|  | // если свойствоства нет с таким ключом что в tag, сделать и записать 1 |

## ****01-sort.js****

|  |
| --- |
| /\* |
|  | \* Array.prototype.sort(callback(currentEl, nextEl){}) |
|  | \* - Сортирует и ИЗМЕНЯЕТ оригинальный массив |
|  | \* - По умолчанию: |
|  | \* - сортирует по возрастанию |
|  | \* - приводит элементы к строке и сортирует по [Unicode](https://unicode-table.com/en/) |
|  | \*/ |
|  |  |
|  | const numbers = [1, 9, 6, 2, 3]; |
|  | // numbers.sort(); |
|  | // console.log('numbers', numbers); |
|  |  |
|  | const letters = ['b', 'B', 'a', 'A']; |
|  | // letters.sort(); |
|  | // console.log('letters', letters); |
|  |  |
|  | /\* |
|  | \* compareFunction - функция сравнения (callback) |
|  | \* Элементы массива сортируются в соответствии с её возвращаемым значением |
|  | \* - eсли compareFunction(A, B) меньше 0, сортировка поставит A перед B |
|  | \* - если compareFunction(A, B) больше 0, сортировка поставит B перед A |
|  | \* - если compareFunction(A, B) вернёт 0, сортировка оставит A и B на неизменными по отношению друг к другу, но отсортирует их по отношению ко всем другим элементам. |
|  | \*/ |
|  |  |
|  | // numbers.sort((curEl, nextEl) => { |
|  | // return nextEl - curEl; |
|  | // }); |
|  |  |
|  | /\* |
|  | \* Как сделать копию массива чтобы не сортировать оригинальный |
|  | \* - Array.prototype.slice() |
|  | \* - Операция spread |
|  | \*/ |
|  |  |
|  | const descSortedNumbers = [...numbers].sort((a, b) => b - a); |
|  | const ascSortedNumbers = [...numbers].sort((a, b) => a - b); |
|  | // console.log('descSortedNumbers', descSortedNumbers); |
|  | // console.log('ascSortedNumbers', ascSortedNumbers); |
|  |  |
|  | /\* |
|  | \* Кастомная сортировка сложных типов |
|  | \*/ |
|  | const players = [ |
|  | { id: 'player-1', name: 'Mango', timePlayed: 310, online: false }, |
|  | { id: 'player-2', name: 'Poly', timePlayed: 470, online: true }, |
|  | { id: 'player-3', name: 'Aiwi', timePlayed: 230, online: true }, |
|  | { id: 'player-4', name: 'Ajax', timePlayed: 150, online: false }, |
|  | { id: 'player-5', name: 'Chelsey', timePlayed: 80, online: true }, |
|  | ]; |
|  |  |
|  | // По игровому времени |
|  | const sortedByBestPlayers = [...players].sort( |
|  | (prevPlayer, nextPlayer) => nextPlayer.timePlayed - prevPlayer.timePlayed, |
|  | ); |
|  | // console.table(sortedByBestPlayers); |
|  |  |
|  | const sortedByWorstPlayers = [...players].sort( |
|  | (prevPlayer, nextPlayer) => prevPlayer.timePlayed - nextPlayer.timePlayed, |
|  | ); |
|  | // console.table(sortedByWorstPlayers); |
|  |  |
|  | const byName = [...players].sort((a, b) => { |
|  | const result = a.name[0] > b.name[0]; |
|  |  |
|  | if (result) { |
|  | return 1; |
|  | } |
|  |  |
|  | if (!result) { |
|  | return -1; |
|  | } |
|  | }); |
|  |  |
|  | console.table(byName); |

## ****02-flat.js****

|  |
| --- |
| /\* |
|  | \* Array.prototype.flat(depth) |
|  | \* - Разглаживает массив до указанной глубины |
|  | \* - По умолчанию глубина 1 |
|  | \*/ |
|  |  |
|  | const array = [1, 2, [4, [5]], [6, [7, 8, [9]]]]; |
|  | // console.log(array.flat(3)); |
|  |  |
|  | /\* |
|  | \* Array.prototype.flatMap(callback) |
|  | \* - Комбинация map + flat |
|  | \*/ |
|  |  |
|  | const tweets = [ |
|  | { id: '000', likes: 5, tags: ['js', 'nodejs'] }, |
|  | { id: '001', likes: 2, tags: ['html', 'css'] }, |
|  | { id: '002', likes: 17, tags: ['html', 'js', 'nodejs'] }, |
|  | { id: '003', likes: 8, tags: ['css', 'react'] }, |
|  | { id: '004', likes: 0, tags: ['js', 'nodejs', 'react'] }, |
|  | ]; |
|  |  |
|  | // const tags = tweets.flatMap(t => t.tags); |
|  | // console.log(tags); |
|  |  |
|  | // const stats = tags.reduce((acc, tag) => { |
|  | // return { |
|  | // ...acc, |
|  | // [tag]: acc[tag] ? acc[tag] + 1 : 1, |
|  | // }; |
|  | // }, {}); |
|  |  |
|  | const stats = tweets |
|  | .flatMap(tweet => tweet.tags) |
|  | .reduce( |
|  | (acc, tag) => ({ |
|  | ...acc, |
|  | [tag]: acc[tag] ? acc[tag] + 1 : 1, |
|  | }), |
|  | {}, |
|  | ); |
|  |  |
|  | console.log(stats); |

## ****03-chaining.js****

|  |
| --- |
| /\* |
|  | \* Цепочки вызовов - chaining |
|  | \*/ |
|  | const numbers = [1, 5, 2, 4, 3]; |
|  |  |
|  | // const greaterThenTwo = numbers.filter(num => num > 2); |
|  | // console.log(greaterThenTwo); |
|  |  |
|  | // const multByThree = greaterThenTwo.map(num => num \* 3); |
|  | // console.log(multByThree); |
|  |  |
|  | // const sorted = multByThree.sort((a, b) => a - b); |
|  | // console.log(sorted); |
|  |  |
|  | // Цепочка предыдущих трёх |
|  | const sorted = numbers |
|  | .filter(num => num > 2) |
|  | .map(num => num \* 3) |
|  | .sort((a, b) => a - b); |
|  |  |
|  | console.log(sorted); |
|  |  |
|  | /\* |
|  | \* Сортируем тех кто онлайн по рангу |
|  | \* - сначала фильтруем |
|  | \* - потом сортируем |
|  | \*/ |
|  | const players = [ |
|  | { id: 'id-1', tag: 'Mango', isOnline: true, rank: 800 }, |
|  | { id: 'id-2', tag: 'Poly', isOnline: false, rank: 600 }, |
|  | { id: 'id-3', tag: 'Ajax', isOnline: true, rank: 100 }, |
|  | { id: 'id-4', tag: 'Kiwi', isOnline: true, rank: 400 }, |
|  | { id: 'id-5', tag: 'Chelsy', isOnline: false, rank: 200 }, |
|  | ]; |
|  |  |
|  | const onlineAndSorted = players |
|  | .filter(player => player.isOnline) |
|  | .sort((playerA, playerB) => playerA.rank - playerB.rank); |
|  |  |
|  | // console.table(onlineAndSorted); |
|  |  |
|  | /\* |
|  | \* Chaining в методах объекта как jquery |
|  | \*/ |
|  |  |
|  | const element = { |
|  | class: '', |
|  | hovered: false, |
|  | changeClass(cls) { |
|  | this.class = cls; |
|  |  |
|  | return this; |
|  | }, |
|  | toggleHovered() { |
|  | this.hovered = !this.hovered; |
|  |  |
|  | return this; |
|  | }, |
|  | }; |
|  |  |
|  | element.toggleHovered().changeClass('open').toggleHovered(); |
|  | console.log(element); |

## ****04-lodash.js****

|  |
| --- |
| /\* |
|  | \* isEmpty() |
|  | \*/ |
|  |  |
|  | // console.log(\_.isEmpty({})); |
|  | // console.log(\_.isEmpty({ a: 1 })); |
|  |  |
|  | /\* |
|  | \* get() |
|  | \* |
|  | \* - user && user.location && obj.location.city |
|  | \* - user?.location?.city |
|  | \*/ |
|  |  |
|  | const user = { |
|  | name: 'mango', |
|  | location: { |
|  | city: 'Lviv', |
|  | }, |
|  | }; |
|  |  |
|  | // console.log(\_.get(user, 'location.city')); |
|  |  |
|  | // console.log(user.location.city); |
|  |  |
|  | // if (user && user.location && user.location.city) { |
|  | // console.log(user.location.city); |
|  | // } |
|  |  |
|  | // console.log(user?.location?.city); |
|  |  |
|  | /\* |
|  | \* union() |
|  | \*/ |
|  |  |
|  | // console.log(\_.union([1, 2, 3], [3, 4, 5])); |
|  |  |
|  | /\* |
|  | \* range() |
|  | \*/ |
|  |  |
|  | // console.log(\_.range(10, 51)); |
|  |  |
|  | /\* |
|  | \* sortBy() |
|  | \*/ |
|  |  |
|  | /\* |
|  | \* sum() и sumBy() |
|  | \*/ |
|  |  |
|  | const players = [ |
|  | { id: 'player-1', name: 'Mango', timePlayed: 310, online: false }, |
|  | { id: 'player-2', name: 'Poly', timePlayed: 470, online: true }, |
|  | { id: 'player-3', name: 'Aiwi', timePlayed: 230, online: true }, |
|  | { id: 'player-4', name: 'Ajax', timePlayed: 150, online: false }, |
|  | { id: 'player-5', name: 'Chelsey', timePlayed: 80, online: true }, |
|  | ]; |
|  |  |
|  | // console.log(\_.sumBy(players, player => player.timePlayed)); |
|  |  |
|  | /\* |
|  | \* uniq() и uniqBy() |
|  | \* sortedUniq() и sortedUniqBy() |
|  | \*/ |
|  |  |
|  | /\* |
|  | \* random() |
|  | \*/ |
|  |  |
|  | /\* |
|  | \* min() и max() |
|  | \* minBy() и maxBy() |
|  | \*/ |
|  |  |
|  | // console.log(\_.minBy(players, player => player.timePlayed)); |
|  |  |
|  | /\* |
|  | \* camelCase(), capitalize(), kebabCase(), lowerCase(), upperCase() |
|  | \*/ |
|  |  |
|  | console.log(\_.kebabCase(' a b c ')); |