#1

function calculateTotalPrice(orderedItems) {

let totalPrice = 0;

// Пиши код ниже этой строки

orderedItems.forEach((item) =>{

totalPrice += item;

});

// Пиши код выше этой строки

return totalPrice;

}

**Метод forEach(callback)**

Теория

Перебирающий метод массива, который используется как замена циклов for и for...of при работе с коллекцией.

массив.forEach(function callback(element, index, array) {

// Тело коллбек-функции

});

* Поэлементно перебирает массив.
* Вызывает коллбек-функцию для каждого элемента массива.
* Ничего не возвращает.

Аргументы коллбек-функции это значение текущего элемента element, его индекс index и сам исходный массив array. Объявлять можно только те параметры которые нужны, чаще всего это элемент, главное не забывать про их порядок.

const numbers = [5, 10, 15, 20, 25];

// Классический for

for (let i = 0; i < numbers.length; i += 1) {

console.log(`Индекс ${i}, значение ${numbers[i]}`);

}

// Перебирающий forEach

numbers.forEach(function (number, index) {

console.log(`Индекс ${index}, значение ${number}`);

});

Единственным случаем, когда стоит использовать циклы for или for...of для перебора массива, это задачи с прерыванием выполнения цикла. Прервать выполнение метода forEach нельзя, он всегда перебирает массив до конца.

**Задание**

Функция calculateTotalPrice(orderedItems) принимает один параметр orderedItems - массив чисел, и рассчитывает общую сумму его элементов, которая сохраняется в переменной totalPrice и возвращается как результат работы функции.

Выполни рефакторинг функции так, чтобы вместо цикла for она использовала метод forEach.

**Тесты**

* Объявлена функция calculateTotalPrice(orderedItems).
* Для перебора массива orderedItems использован метод forEach.
* Вызов функции calculateTotalPrice([12, 85, 37, 4]) возвращает 138.
* Вызов функции calculateTotalPrice([164, 48, 291]) возвращает 503.
* Вызов функции calculateTotalPrice([412, 371, 94, 63, 176]) возвращает 1116.
* Вызов функции со случайными, но валидными аргументами, возвращает правильное значение.

#2

function filterArray(numbers, value) {

const filteredNumbers = [];

// Пиши код ниже этой строки

numbers.forEach( (number) =>{

if (number > value) {

filteredNumbers.push(number);

}

});

// Пиши код выше этой строки

return filteredNumbers;

}

**Задача. Фильтрация массива чисел**

**Задание**

Функция filterArray(numbers, value) принимает массив чисел numbers и возвращает новый массив, в котором будут только те элементы оригинального массива, которые больше чем значение параметра value.

Выполни рефакторинг функции так, чтобы вместо цикла for она использовала метод forEach.

**Тесты**

* Объявлена функция filterArray(numbers, value).
* Для перебора массива numbers использован метод forEach.
* Вызов функции filterArray([1, 2, 3, 4, 5], 3) возвращает [4, 5].
* Вызов функции filterArray([1, 2, 3, 4, 5], 4) возвращает [5].
* Вызов функции filterArray([1, 2, 3, 4, 5], 5) возвращает [].
* Вызов функции filterArray([12, 24, 8, 41, 76], 38) возвращает [41, 76].
* Вызов функции filterArray([12, 24, 8, 41, 76], 20) возвращает [24, 41, 76].
* Вызов функции со случайными, но валидными аргументами, возвращает правильное значение.

#3

function getCommonElements(firstArray, secondArray) {

const commonElements = [];

// Пиши код ниже этой строки

firstArray.forEach((element)=> {

if (secondArray.includes(element)) {

commonElements.push(element);

}

});

return commonElements;

// Пиши код выше этой строки

}

**Задача. Общие элементы**

**Задание**

Функция getCommonElements(firstArray, secondArray) принимает два массива произвольной длины в параметры firstArray и secondArray, и возвращает новый массив их общих элементов, то есть тех которые есть в обоих массивах.

Выполни рефакторинг функции так, чтобы вместо цикла for она использовала метод forEach.

**Тесты**

* Объявлена функция getCommonElements(firstArray, secondArray).
* Для перебора параметра (массива) использован метод forEach.
* Вызов getCommonElements([1, 2, 3], [2, 4]) возвращает [2].
* Вызов getCommonElements([1, 2, 3], [2, 1, 17, 19]) возвращает [1, 2].
* Вызов getCommonElements([24, 12, 27, 3], [12, 8, 3, 36, 27]) возвращает [12, 27, 3].
* Вызов getCommonElements([10, 20, 30, 40], [4, 30, 17, 10, 40]) возвращает [10, 30, 40].
* Вызов getCommonElements([1, 2, 3], [10, 20, 30]) возвращает [].
* Вызов функции со случайными, но валидными аргументами, возвращает правильное значение.

#4

// Пиши код ниже этой строки

const calculateTotalPrice = (quantity, pricePerItem) => {

// Пиши код выше этой строки

return quantity \* pricePerItem;

}

**Стрелочные функции.**

Теория

Стрелочные функции имеют сокращённый, более лаконичный синтаксис, что уменьшает объем кода, особенно когда функция маленькая или если она используется как коллбек.

Все стрелки создаются как функциональное выражение, и если функция не анонимна, то она должна быть присвоена переменной.

// Обычное объявление функции

function classicAdd(a, b, c) {

return a + b + c;

}

// Тоже самое как стрелочная функция

const arrowAdd = (a, b, c) => {

return a + b + c;

};

Ключевое слово function не используется, вместо этого сразу идёт объявление параметров, за которыми следует символ => и тело функции.

Если параметров несколько, то они перечисляются через запятую в круглых скобках, между знаками равно = и стрелкой =>.

const add = (a, b, c) => {

return a + b + c;

};

Если параметр один, его объявление может быть без круглых скобок.

const add = a => {

return a + 5;

};

Если параметров нет, то обязательно должны быть пустые круглые скобки.

const greet = () => {

console.log('Привет!');

};

**Задание**

Выполни рефакторинг функции calculateTotalPrice() так, чтобы она была объявлена как стрелочная.

**Тесты**

* Объявлена переменная calculateTotalPrice.
* Переменной calculateTotalPrice присвоена стрелочная функция с параметрами (quantity, pricePerItem).
* Вызов calculateTotalPrice(5, 100) возвращает 500.
* Вызов calculateTotalPrice(8, 60) возвращает 480.
* Вызов calculateTotalPrice(3, 400) возвращает 1200.
* Вызов функции со случайными, но валидными аргументами, возвращает правильное значение.

#5

// Пиши код ниже этой строки

const calculateTotalPrice = (quantity, pricePerItem) => quantity \* pricePerItem;

;

// Пиши код выше этой строки

**Неявный возврат**

Теория

В стрелочной функции после символа => идёт её тело. Здесь может быть два варианта: с фигурными скобками и без них.

const add = (a, b, c) => {

console.log(a, b, c);

return a + b + c;

};

Если фигурные скобки есть, и функция должна возвращать какое-то значение, необходимо явно поставить return. Это называется явный возврат (explicit return). Такой синтаксис используется в том случае, если в теле функции нужно выполнить ещё какие-то инструкции кроме возврата значения.

const add = (a, b, c) => a + b + c;

Если фигурных скобок нет, то возвращается результат выражения стоящего после =>. Это называется неявный возврат (implicit return). В примере вернётся результат выражения сложения параметров a, b и c.

Синтаксис неявного возврата сильно сокращает «шум» объявления функции с телом и возвращаемым выражением, но подходит только в случае когда в теле функции не нужно выполнять никаких дополнительных инструкций кроме возврата значения.

// До

function classicAdd(a, b, c) {

return a + b + c;

}

// После

const arrowAdd = (a, b, c) => a + b + c;

**Задание**

Выполни рефакторинг функции calculateTotalPrice() так, чтобы она использовала неявный возврат.

**Тесты**

* Объявлена переменная calculateTotalPrice.
* Переменной calculateTotalPrice присвоена стрелочная функция с параметрами (quantity, pricePerItem).
* В функции использован неявный возврат.
* Вызов calculateTotalPrice(5, 100) возвращает 500.
* Вызов calculateTotalPrice(8, 60) возвращает 480.
* Вызов calculateTotalPrice('3, 400) возвращает 1200.
* Вызов функции со случайными, но валидными аргументами, возвращает правильное значение.

#6

// Пиши код ниже этой строки

const calculateTotalPrice = (orderedItems) =>{

let totalPrice = 0;

orderedItems.forEach((item) =>{

totalPrice += item;

});

return totalPrice;

}

// Пиши код выше этой строки

**Стрелочные функции как коллбеки**

Теория

Анонимные стрелочные функции отлично подходят как коллбеки для перебирающих методов массива из-за более краткого синтаксиса объявления, особенно если не нужно тело функции.

const numbers = [5, 10, 15, 20, 25];

// Объявление функции

numbers.forEach(function (number, index) {

console.log(`Индекс ${index}, значение ${number}`);

});

// Анонимная стрелочная функция

numbers.forEach((number, index) => {

console.log(`Индекс ${index}, значение ${number}`);

});

Стрелочную коллбек-функцию также можно объявлять отдельно и передавать на неё ссылку. Это стоит делать если одна функция используется в нескольих местах программы или если она громоздкая.

const numbers = [5, 10, 15, 20, 25];

const logMessage = (number, index) => {

console.log(`Индекс ${index}, значение ${number}`);

};

numbers.forEach(logMessage);

**Задание**

Выполни рефакторинг функции calculateTotalPrice(orderedItems) заменив её объявление на стрелочную функцию. Замени коллбек-функцию передаваемую в метод forEach() на стрелочную функцию.

**Тесты**

* Объявлена переменная calculateTotalPrice.
* Переменной calculateTotalPrice присвоена стрелочная функция с параметром (orderedItems).
* Для перебора массива orderedItems использован метод forEach.
* Коллбек для метода forEach это стрелочная функция.
* Вызов функции calculateTotalPrice([12, 85, 37, 4]) возвращает 138.
* Вызов функции calculateTotalPrice([164, 48, 291]) возвращает 503.
* Вызов функции calculateTotalPrice([412, 371, 94, 63, 176]) возвращает 1116.
* Вызов функции со случайными, но валидными аргументами, возвращает правильное значение.

#7

// Пиши код ниже этой строки

const filterArray = (numbers, value) =>{

const filteredNumbers = [];

numbers.forEach( (number) => {

if (number > value) {

filteredNumbers.push(number);

}

});

// Пиши код выше этой строки

return filteredNumbers;

}

**Задача. Фильтрация массива чисел 2.0**

**Задание**

Замени объявление функции filterArray() и коллбек для метода forEach() на стрелочные функции.

**Тесты**

* Объявлена переменная filterArray.
* Переменной filterArray присвоена стрелочная функция с параметрами (numbers, value).
* Для перебора массива numbers использован метод forEach.
* Коллбек для метода forEach это стрелочная функция.
* Вызов функции filterArray([1, 2, 3, 4, 5], 3) возвращает [4, 5].
* Вызов функции filterArray([1, 2, 3, 4, 5], 4) возвращает [5].
* Вызов функции filterArray([1, 2, 3, 4, 5], 5) возвращает [].
* Вызов функции filterArray([12, 24, 8, 41, 76], 38) возвращает [41, 76].
* Вызов функции filterArray([12, 24, 8, 41, 76], 20) возвращает [24, 41, 76].
* Вызов функции со случайными, но валидными аргументами, возвращает правильное значение.

#8

// Пиши код ниже этой строки

const getCommonElements = (firstArray, secondArray) => {

const commonElements = [];

firstArray.forEach( (element) => {

if (secondArray.includes(element)) {

commonElements.push(element);

}

});

// Пиши код выше этой строки

return commonElements;

}

**Задача. Общие элементы 2.0**

**Задание**

Замени объявление функции getCommonElements() и коллбек для метода forEach() на стрелочные функции.

**Тесты**

-Объявлена переменная getCommonElements.

* Переменной getCommonElements присвоена стрелочная функция с параметрами (firstArray, secondArray).
* Для перебора массива firstArray использован метод forEach.
* Коллбек для метода forEach это стрелочная функция.
* Вызов getCommonElements([1, 2, 3], [2, 4]) возвращает [2].
* Вызов getCommonElements([1, 2, 3], [2, 1, 17, 19]) возвращает [1, 2].
* Вызов getCommonElements([24, 12, 27, 3], [12, 8, 3, 36, 27]) возвращает [12, 27, 3].
* Вызов getCommonElements([10, 20, 30, 40], [4, 30, 17, 10, 40]) возвращает [10, 30, 40].
* Вызов getCommonElements([1, 2, 3], [10, 20, 30]) возвращает [].
* Вызов функции со случайными, но валидными аргументами, возвращает правильное значение.

#9

function changeEven(numbers, value) {

const array = [];

numbers.forEach((number) => {

if (number % 2 === 0) {

number += value;

array.push(number);

} else {

array.push(number);

}

});

return array;

// Пиши код выше этой строки

}

**Чистые функции**

Теория

**Функция с побочными эффектами** - это функция которая в процессе выполнения может изменять или использовать глобальные переменные, изменять значение аргументов ссылочного типа, выполнять операции ввода-вывода и т. п.

const dirtyMultiply = (array, value) => {

for (let i = 0; i < array.length; i += 1) {

array[i] = array[i] \* value;

}

};

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

dirtyMultiply(numbers, 2);

// Произошла мутация исходных данных - массива numbers

console.log(numbers); // [2, 4, 6, 8, 10]

Функция dirtyMultiply(array, value) умножает каждый элемент массива array на число value. Она изменяет (мутирует) исходный массив по ссылке.

**Чистая функция (pure function)** - это функция результат которой зависит только от значений переданных аргументов. При одинаковых аргументах она всегда возвращает один и тот же результат и не имеет побочных эффектов, то есть не изменяет значения аргументов.

Напишем реализацию чистой функции умножения элементов массива, возвращающей новый массив, не изменяя исходный.

const pureMultiply = (array, value) => {

const newArray = [];

array.forEach(element => {

newArray.push(element \* value);

});

return newArray;

};

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

const doubledNumbers = pureMultiply(numbers, 2);

// Не произошло мутации исходных данных

console.log(numbers); // [1, 2, 3, 4, 5]

// Функция вернула новый массив с изменёнными данными

console.log(doubledNumbers); // [2, 4, 6, 8, 10]

**Задание**

Функция changeEven(numbers, value) принимает массив чисел numbers и обновляет каждый элемент, значение которого это чётное число, добавляя к нему значение параметра value.

Выполни рефакторинг функции так, чтобы она стала чистой - не изменяла массив чисел numbers, а создавала, наполняла и возвращала новый массив с обновлёнными значениями.

**Тесты**

* Объявлена функция changeEven(numbers, value).
* Функция changeEven не изменяет значение параметра numbers.
* Вызов changeEven([1, 2, 3, 4, 5], 10) возвращает новый массив [1, 12, 3, 14, 5].
* Вызов changeEven([2, 8, 3, 7, 4, 6], 10) возвращает новый массив [12, 18, 3, 7, 14, 16].
* Вызов changeEven([17, 24, 68, 31, 42], 100) возвращает новый массив [17, 124, 168, 31, 142].
* Вызов changeEven([44, 13, 81, 92, 36, 54], 100) возвращает новый массив [144, 13, 81, 192, 136, 154].
* Вызов функции со случайными, но валидными аргументами, возвращает правильное значение.

#10

const planets = ['Земля', 'Марс', 'Венера', 'Юпитер'];

// Пиши код ниже этой строки

const planetsLengths = planets.map((planet) => planet.length);

**Метод map()**

Теория

Большинство перебирающих методов массива это чистые функции. Они создают новый массив, заполняют его, применяя к значению каждого элемента указанную коллбек-функцию, после чего возвращают этот новый массив.

Метод map(callback) используется для трансформации массива. Он вызывает коллбек-функцию для каждого элемента исходного массива, а результат её работы записывает в новый массив, который и будет результатом выполнения метода.

массив.map((element, index, array) => {

// Тело коллбек-функции

});

* Поэлементно перебирает оригинальный массив.
* Не изменяет оригинальный массив.
* Результат работа коллбек-функции записывается в новый массив.
* Возвращает новый массив такой же длины.

Его можно использовать для того, чтобы изменить каждый элемент массива. Оригинальный массив используется как эталон, на базе которого можно сделать другую коллекцию.

const planets = ['Земля', 'Марс', 'Венера', 'Юпитер'];

const planetsInUpperCase = planets.map((planet) => planet.toUpperCase());

console.log(planetsInUpperCase); // ['ЗЕМЛЯ', 'МАРС', 'ВЕНЕРА', 'ЮПИТЕР']

const planetsInLowerCase = planets.map((planet) => planet.toLowerCase());

console.log(planetsInLowerCase); // ['земля', 'марс', 'венера', 'юпитер']

// Оригинальный массив не изменился

console.log(planets); // ['Земля', 'Марс', 'Венера', 'Юпитер']

Использование анонимных стрелочных функций с неявным возвратом сильно сокращает «шум» объявления коллбек-функции, делая код чище и проще для восприятия.

**Задание**

Дополни код так, чтобы в переменной planetsLengths получился массив длин названий планет. Обязательно используй метод map().

**Тесты**

* Объявлена переменная planets.
* Значение переменной planets это массив ['Земля', 'Марс', 'Венера', 'Юпитер'].
* Объявлена переменная planetsLengths.
* Значение переменной planetsLengths это массив [5, 4, 6, 6].
* Для перебора массива планет использован метод map().

#11

const books = [

{ title: 'Последнее королевство', author: 'Бернард Корнуэлл', rating: 8.38 },

{ title: 'На берегу спокойных вод', author: 'Роберт Шекли', rating: 8.51 },

{ title: 'Сон смешного человека', author: 'Федор Достоевский', rating: 7.75 },

{ title: 'Красна как кровь', author: 'Ли Танит', rating: 7.94 },

{ title: 'Враг Божий', author: 'Бернард Корнуэлл', rating: 8.67 }

];

// Пиши код ниже этой строки

const titles = books.map(book => book.title);

**Метод map() и массив объектов**

Теория

Мы уже знаем что повседневная задача это манипуляция массивом объектов. Например, получить массив значений свойства из всех объектов. Есть массив студентов, а нужно получить отдельный массив их имён.

const students = [

{ name: 'Манго', score: 83 },

{ name: 'Поли', score: 59 },

{ name: 'Аякс', score: 37 },

{ name: 'Киви', score: 94 },

{ name: 'Хьюстон', score: 64 }

];

const names = students.map(student => student.name);

console.log(names); // ['Манго', 'Поли', 'Аякс', 'Киви', 'Хьюстон']

Используя метод map() можно перебрать массив объектов, и в коллбек-функции вернуть значение свойства каждого из них.

**Задание**

Используя метод map() сделай так, чтобы в переменной titles получился массив названий книг (свойство title) из всех объектов массива books.

**Тесты**

* Объявлена переменная books.
* Значение переменной books это массив.
* Объявлена переменная titles.
* Значение переменной titles это массив ['Последнее королевство', 'На берегу спокойных вод', 'Сон смешного человека', 'Красна как кровь', 'Враг Божий'].
* Для перебора массива books используется метод map() как чистая функция.

#12

const books = [

{

title: 'Последнее королевство',

author: 'Бернард Корнуэлл',

genres: ['приключения', 'историческое']

},

{

title: 'На берегу спокойных вод',

author: 'Роберт Шекли',

genres: ['фантастика']

},

{

title: 'Красна как кровь',

author: 'Ли Танит',

genres: ['ужасы', 'мистика']

}

];

// Пиши код ниже этой строки

const genres = books.flatMap(book => book.genres);

**Метод flatMap()**

Теория

Метод flatMap(callback) аналогичен методу map(), но применяется в случаях, когда результат это многомерный массив который необходимо «разгладить».

массив.flatMap((element, index, array) => {

// Тело коллбек-функции

});

В массиве students хранится список студентов со списком предметов, которые посещает студент, в свойстве courses. Несколько студентов могут посещать один и тот же предмет. Необходимо составить список всех предметов, которые посещает эта группа студентов, пока даже повторяющихся.

const students = [

{ name: 'Манго', courses: ['математика', 'физика'] },

{ name: 'Поли', courses: ['информатика', 'математика'] },

{ name: 'Киви', courses: ['физика', 'биология'] },

];

students.map((student) => student.courses);

// [['математика', 'физика'], ['информатика', 'математика'], ['физика', 'биология']]

students.flatMap((student) => student.courses);

// ['математика', 'физика', 'информатика', 'математика', 'физика', 'биология'];

Он вызывает коллбек-функцию для каждого элемента исходного массива, а результат её работы записывает в новый массив. Отличие от map() в том, что новый массив «разглаживается» на глубину равную единице (одна вложенность). Этот разглаженный массив и есть результат работы flatMap().

**Задание**

Используя метод flatMap() сделай так, чтобы в переменной genres получился массив всех жанров книг (свойство genres) из массива книг books.

**Тесты**

* Объявлена переменная books.
* Значение переменной books это массив объектов.
* Объявлена переменная genres.
* Значение переменной genres это массив [ 'приключения', 'историческое', 'фантастика', 'ужасы', 'мистика' ].
* Для перебора массива books используется метод flatMap().

#13

// Пиши код ниже этой строки

const getUserNames = (users) => {

return users.map((user) => user.name);

};

// Пиши код выше этой строки

**Задача. Имена пользователей**

Массив объектов пользователей

Этот массив объектов мы будем передавать в параметр users при вызове функции из задания.

[

{

name: 'Moore Hensley',

email: 'moorehensley@indexia.com',

eyeColor: 'blue',

friends: ['Sharron Pace'],

isActive: false,

balance: 2811,

skills: ['ipsum', 'lorem'],

gender: 'male',

age: 37,

},

{

name: 'Sharlene Bush',

email: 'sharlenebush@tubesys.com',

eyeColor: 'blue',

friends: ['Briana Decker', 'Sharron Pace'],

isActive: true,

balance: 3821,

skills: ['tempor', 'mollit', 'commodo', 'veniam', 'laborum'],

gender: 'female',

age: 34,

},

{

name: 'Ross Vazquez',

email: 'rossvazquez@xinware.com',

eyeColor: 'green',

friends: ['Marilyn Mcintosh', 'Padilla Garrison', 'Naomi Buckner'],

isActive: false,

balance: 3793,

skills: ['nulla', 'anim', 'proident', 'ipsum', 'elit'],

gender: 'male',

age: 24,

},

{

name: 'Elma Head',

email: 'elmahead@omatom.com',

eyeColor: 'green',

friends: ['Goldie Gentry', 'Aisha Tran'],

isActive: true,

balance: 2278,

skills: ['adipisicing', 'irure', 'velit'],

gender: 'female',

age: 21,

},

{

name: 'Carey Barr',

email: 'careybarr@nurali.com',

eyeColor: 'blue',

friends: ['Jordan Sampson', 'Eddie Strong'],

isActive: true,

balance: 3951,

skills: ['ex', 'culpa', 'nostrud'],

gender: 'male',

age: 27,

},

{

name: 'Blackburn Dotson',

email: 'blackburndotson@furnigeer.com',

eyeColor: 'brown',

friends: ['Jacklyn Lucas', 'Linda Chapman'],

isActive: false,

balance: 1498,

skills: ['non', 'amet', 'ipsum'],

gender: 'male',

age: 38,

},

{

name: 'Sheree Anthony',

email: 'shereeanthony@kog.com',

eyeColor: 'brown',

friends: ['Goldie Gentry', 'Briana Decker'],

isActive: true,

balance: 2764,

skills: ['lorem', 'veniam', 'culpa'],

gender: 'female',

age: 39,

},

];

**Задание**

Дополни функцию getUserNames(users) так, чтобы она возвращала массив имён пользователей (свойство name) из массива объектов в параметре users.

**Тесты**

* Объявлена переменная getUserNames.
* Переменной getUserNames присвоена стрелочная функция с параметром (users).
* Для перебора параметра users используется метод map().
* Вызов функции с указанным массивом пользователей возвращает массив ['Moore Hensley', 'Sharlene Bush', 'Ross Vazquez', 'Elma Head', 'Carey Barr', 'Blackburn Dotson', 'Sheree Anthony'].
* Вызов функции со случайными, но валидными аргументами, возвращает правильное значение.

#14

// Пиши код ниже этой строки

const getUserEmails = users => {

return users.map(user => user.email)

};

// Пиши код выше этой строки

**Задача. Почты пользователей**

Массив объектов пользователей

Этот массив объектов мы будем передавать в параметр users при вызове функции из задания.

[

{

name: 'Moore Hensley',

email: 'moorehensley@indexia.com',

eyeColor: 'blue',

friends: ['Sharron Pace'],

isActive: false,

balance: 2811,

skills: ['ipsum', 'lorem'],

gender: 'male',

age: 37,

},

{

name: 'Sharlene Bush',

email: 'sharlenebush@tubesys.com',

eyeColor: 'blue',

friends: ['Briana Decker', 'Sharron Pace'],

isActive: true,

balance: 3821,

skills: ['tempor', 'mollit', 'commodo', 'veniam', 'laborum'],

gender: 'female',

age: 34,

},

{

name: 'Ross Vazquez',

email: 'rossvazquez@xinware.com',

eyeColor: 'green',

friends: ['Marilyn Mcintosh', 'Padilla Garrison', 'Naomi Buckner'],

isActive: false,

balance: 3793,

skills: ['nulla', 'anim', 'proident', 'ipsum', 'elit'],

gender: 'male',

age: 24,

},

{

name: 'Elma Head',

email: 'elmahead@omatom.com',

eyeColor: 'green',

friends: ['Goldie Gentry', 'Aisha Tran'],

isActive: true,

balance: 2278,

skills: ['adipisicing', 'irure', 'velit'],

gender: 'female',

age: 21,

},

{

name: 'Carey Barr',

email: 'careybarr@nurali.com',

eyeColor: 'blue',

friends: ['Jordan Sampson', 'Eddie Strong'],

isActive: true,

balance: 3951,

skills: ['ex', 'culpa', 'nostrud'],

gender: 'male',

age: 27,

},

{

name: 'Blackburn Dotson',

email: 'blackburndotson@furnigeer.com',

eyeColor: 'brown',

friends: ['Jacklyn Lucas', 'Linda Chapman'],

isActive: false,

balance: 1498,

skills: ['non', 'amet', 'ipsum'],

gender: 'male',

age: 38,

},

{

name: 'Sheree Anthony',

email: 'shereeanthony@kog.com',

eyeColor: 'brown',

friends: ['Goldie Gentry', 'Briana Decker'],

isActive: true,

balance: 2764,

skills: ['lorem', 'veniam', 'culpa'],

gender: 'female',

age: 39,

},

];

**Задание**

Дополни функцию getUserEmails(users) так, чтобы она возвращала массив почтовых адресов пользователей (свойство email) из массива объектов в параметре users.

**Тесты**

* Объявлена переменная getUserNames.
* Переменной getUserNames присвоена стрелочная функция с параметром (users).
* Для перебора параметра users используется метод map().
* Вызов функции с указанным массивом пользователей возвращает массив ['moorehensley@indexia.com', 'sharlenebush@tubesys.com', 'rossvazquez@xinware.com', 'elmahead@omatom.com', 'careybarr@nurali.com', 'blackburndotson@furnigeer.com', 'shereeanthony@kog.com'].
* Вызов функции со случайными, но валидными аргументами, возвращает правильное значение.

#15

const numbers = [17, 24, 82, 61, 36, 18, 47, 52, 73];

// Пиши код ниже этой строки

const evenNumbers = numbers.filter((element) => {

if (element % 2 === 0) {

return element;

}

});

const oddNumbers = numbers.filter((element) => {

if (element % 2 !== 0) {

return element;

}

});

**Методы filter и find**

Теория

Метод filter(callback) используется для единственной операции - фильтрации массива, то есть когда необходимо выбрать более одного элемента из коллекции по какому-то критерию.

массив.filter((element, index, array) => {

// Тело коллбек-функции

});

* Не изменяет оригинальный массив.
* Поэлементно перебирает оригинальный массив.
* Возвращает новый массив.
* 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.
* Если ни один элемент не удовлетворил условию, возвращает пустой массив.

const values = [51, -3, 27, 21, -68, 42, -37];

const positiveValues = values.filter(value => value >= 0);

console.log(positiveValues); // [51, 27, 21, 42]

const negativeValues = values.filter(value => value < 0);

console.log(negativeValues); // [-3, -68, -37]

const bigValues = values.filter(value => value > 1000);

console.log(bigValues); // []

// Оригинальный массив не изменился

console.log(values); // [51, -3, 27, 21, -68, 42, -37]

То есть метод filter вызывает коллбек-функцию для каждого элемента исходного массива и если результат её выполнения true, текущий элемент добавляет в новый массив.

**Задание**

Дополни код так, чтобы в переменной evenNumbers получился массив чётных чисел из массива numbers, а в переменной oddNumbers массив нечётных. Обязательно используй метод filter().

**Тесты**

* Объявлена переменная numbers.
* Значение переменной numbers это массив [17, 24, 82, 61, 36, 18, 47, 52, 73].
* Объявлена переменная evenNumbers.
* Значение переменной evenNumbers это массив [24, 82, 36, 18, 52].
* Объявлена переменная oddNumbers.
* Значение переменной oddNumbers это массив [17, 61, 47, 73].
* Для перебора массива numbers использован метод filter().

#16

const books = [

{

title: 'Последнее королевство',

author: 'Бернард Корнуэлл',

genres: ['приключения', 'историческое']

},

{

title: 'На берегу спокойных вод',

author: 'Роберт Шекли',

genres: ['фантастика', 'мистика']

},

{

title: 'Красна как кровь',

author: 'Ли Танит',

genres: ['ужасы', 'мистика', 'приключения']

}

];

// Пиши код ниже этой строки

const allGenres = books.flatMap((book) => book.genres);

console.log(allGenres);

const uniqueGenres = allGenres.filter((genres, index, array) => array.indexOf(genres) === index);

**Фильтрация уникальных элементов**

Теория

Используя метод filter() можно выполнить фильтрацию массива так, что в нём останутся только уникальные элементы. Этот приём работает только с массивом примитивных значений - не объектов.

Вернёмся к группе студентов и массиву всех посещаемых предметов, которые мы получили методом flatMap().

const students = [

{ name: 'Манго', courses: ['математика', 'физика'] },

{ name: 'Поли', courses: ['информатика', 'математика'] },

{ name: 'Киви', courses: ['физика', 'биология'] }

];

const allCourses = students.flatMap(student => student.courses);

// ['математика', 'физика', 'информатика', 'математика', 'физика', 'биология'];

В переменной allCourses хранится массив всех посещаемых предметов, которые могут повторяться. Задача заключается в том, чтобы сделать новый массив, в котором будут только уникальные предметы, то есть без повторений.

const uniqueCourses = allCourses.filter(

(course, index, array) => array.indexOf(course) === index

);

Используя array.indexOf(course) выполняем поиск первого совпадения текущего элемента course и получаем его индекс в оригинальном массиве всех курсов. В параметре index хранится индекс текущего элемента course при переборе массива методом filter.

Если результат indexOf() и значение index равны - это уникальный элемент, потому что это первый раз когда такое значение встречается в массиве и на текущей итерации фильтр обрабатывает именно его.

# Массив всех курсов

['математика', 'физика', 'информатика', 'математика', 'физика', 'биология'];

Для элемента 'математика' под индексом 0:

* indexOf() вернёт 0, потому что ищет первое совпадение.
* Значение параметра index будет 0.
* Они равны, значит это уникальный элемент.

Для элемента 'математика' под индексом 3:

* indexOf() вернёт 0, потому что ищет первое совпадение.
* Значение параметра index будет 3.
* Они не равны, значит это повторяющийся - не уникальный элемент.

**Задание**

Дополни код так, чтобы в переменной allGenres был массив всех жанров книг (свойство genres) из массива books, а в переменной uniqueGenres массив уникальных жанров - без повторений.

**Тесты**

* Объявлена переменная books.
* Значение переменной books это массив объектов.
* Объявлена переменная allGenres.
* Значение переменной allGenres это массив ['приключения', 'историческое', 'фантастика', 'мистика', 'ужасы', 'мистика', 'приключения'].
* Объявлена переменная uniqueGenres.
* Значение переменной uniqueGenres это массив ['приключения', 'историческое', 'фантастика', 'мистика', 'ужасы'].
* Для вычисления значения переменной allGenders использован метод flatMap().
* Для вычисления значения переменной uniqueGenres использован метод filter().

#17

const books = [

{ title: 'Последнее королевство', author: 'Бернард Корнуэлл', rating: 8.38 },

{ title: 'На берегу спокойных вод', author: 'Роберт Шекли', rating: 8.51 },

{ title: 'Сон смешного человека', author: 'Федор Достоевский', rating: 7.75 },

{ title: 'Красна как кровь', author: 'Ли Танит', rating: 7.94 },

{ title: 'Враг Божий', author: 'Бернард Корнуэлл', rating: 8.67 }

];

const MIN\_RATING = 8;

const AUTHOR = 'Бернард Корнуэлл';

// Пиши код ниже этой строки

const topRatedBooks = books.filter((book) => book.rating >= MIN\_RATING);

const booksByAuthor = books.filter((book) => book.author === AUTHOR);

**Метод filter() и массив объектов**

Теория

При работе с массивом объектов выполняется фильтрация по значению какого-то свойства. В результате получается новый массив отфильтрованных объектов.

Например, есть массив студентов с баллами за тест. Необходимо отфильтровать лучших (балл выше 80), худших (балл ниже 50) и средних студентов (балл от 50 до 80).

const LOW\_SCORE = 50;

const HIGH\_SCORE = 80;

const students = [

{ name: 'Манго', score: 83 },

{ name: 'Поли', score: 59 },

{ name: 'Аякс', score: 37 },

{ name: 'Киви', score: 94 },

{ name: 'Хьюстон', score: 64 },

];

const best = students.filter((student) => student.score >= HIGH\_SCORE);

console.log(best); // Массив объектов с именами Манго и Киви

const worst = students.filter((student) => student.score < LOW\_SCORE);

console.log(worst); // Массив с одним объектом Аякс

// В коллбек-функции удобно деструктуризировать свойства объекта

const average = students.filter(

({ score }) => score >= LOW\_SCORE && score < HIGH\_SCORE

);

console.log(average); // Массив объектов с именами Поли и Хьюстон

**Задание**

Используя метод filter() дополни код так, чтобы:

* В переменной topRatedBooks получился массив книг рейтинг которых (свойство rating) больше либо равно значению переменной MIN\_RATING.
* В переменной booksByAuthor получился массив книг написанных автором с именем (свойство author) которое совпадает со значением в переменной AUTHOR.

**Тесты**

* Объявлена переменная books.
* Значение переменной books это массив объектов.
* Объявлена переменная MIN\_RATING.
* Значение переменной MIN\_RATING это число 8.
* Объявлена переменная AUTHOR.
* Значение переменной AUTHOR это строка 'Бернард Корнуэлл'.
* Объявлена переменная topRatedBooks.
* Значение переменной topRatedBooks это массив книг с рейтингов выше 8.
* Объявлена переменная booksByAuthor.
* Значение переменной booksByAuthor это массив книг автор которых 'Бернард Корнуэлл'.
* Для перебора массива books использован метод filter().

#18

// Пиши код ниже этой строки

const getUsersWithEyeColor = (users, color) => {

return users.filter (user => {

if (user.eyeColor === color)

return user

})

};

// Пиши код выше этой строки

**Задача. Пользователи с цветом глаз**

Массив объектов пользователей

Этот массив объектов мы будем передавать в параметр users при вызове функции из задания.

[

{

name: 'Moore Hensley',

email: 'moorehensley@indexia.com',

eyeColor: 'blue',

friends: ['Sharron Pace'],

isActive: false,

balance: 2811,

gender: 'male'

},

{

name: 'Sharlene Bush',

email: 'sharlenebush@tubesys.com',

eyeColor: 'blue',

friends: ['Briana Decker', 'Sharron Pace'],

isActive: true,

balance: 3821,

gender: 'female'

},

{

name: 'Ross Vazquez',

email: 'rossvazquez@xinware.com',

eyeColor: 'green',

friends: ['Marilyn Mcintosh', 'Padilla Garrison', 'Naomi Buckner'],

isActive: false,

balance: 3793,

gender: 'male'

},

{

name: 'Elma Head',

email: 'elmahead@omatom.com',

eyeColor: 'green',

friends: ['Goldie Gentry', 'Aisha Tran'],

isActive: true,

balance: 2278,

gender: 'female'

},

{

name: 'Carey Barr',

email: 'careybarr@nurali.com',

eyeColor: 'blue',

friends: ['Jordan Sampson', 'Eddie Strong'],

isActive: true,

balance: 3951,

gender: 'male'

},

{

name: 'Blackburn Dotson',

email: 'blackburndotson@furnigeer.com',

eyeColor: 'brown',

friends: ['Jacklyn Lucas', 'Linda Chapman'],

isActive: false,

balance: 1498,

gender: 'male'

},

{

name: 'Sheree Anthony',

email: 'shereeanthony@kog.com',

eyeColor: 'brown',

friends: ['Goldie Gentry', 'Briana Decker'],

isActive: true,

balance: 2764,

gender: 'female'

}

]

**Задание**

Дополни функцию getUsersWithEyeColor(users, color) так, чтобы она возвращала массив пользователей у которых цвет глаз (свойство eyeColor) совпадает со значением параметра color.

**Тесты**

* Объявлена переменная getUsersWithEyeColor.
* Переменной getUsersWithEyeColor присвоена стрелочная функция с параметрами (users, color).
* Для перебора параметра users используется метод filter().
* Если значение параметра color это 'blue', функция возвращает массив объектов пользователей с именами Moore Hensley, Sharlene Bush и Carey Barr.
* Если значение параметра color это 'green', функция возвращает массив объектов пользователей с именами Ross Vazquez и Elma Head.
* Если значение параметра color это 'brown', функция возвращает массив объектов пользователей с именами Blackburn Dotson и Sheree Anthony.
* Если значение параметра color это любая другая строка, функция возвращает пустой массив.
* Вызов функции со случайными, но валидными аргументами, возвращает правильное значение.

#19

// Пиши код ниже этой строки

const getUsersWithAge = (users, minAge, maxAge) => {

return users.filter((user) => {

if (user.age > minAge && user.age < maxAge) return user;

});

};

// Пиши код выше этой строки

**Задача. Пользователи в возрастной категории**

Массив объектов пользователей

Этот массив объектов мы будем передавать в параметр users при вызове функции из задания.

[

{

name: 'Moore Hensley',

email: 'moorehensley@indexia.com',

eyeColor: 'blue',

friends: ['Sharron Pace'],

isActive: false,

balance: 2811,

gender: 'male',

age: 37

},

{

name: 'Sharlene Bush',

email: 'sharlenebush@tubesys.com',

eyeColor: 'blue',

friends: ['Briana Decker', 'Sharron Pace'],

isActive: true,

balance: 3821,

gender: 'female',

age: 34

},

{

name: 'Ross Vazquez',

email: 'rossvazquez@xinware.com',

eyeColor: 'green',

friends: ['Marilyn Mcintosh', 'Padilla Garrison', 'Naomi Buckner'],

isActive: false,

balance: 3793,

gender: 'male',

age: 24

},

{

name: 'Elma Head',

email: 'elmahead@omatom.com',

eyeColor: 'green',

friends: ['Goldie Gentry', 'Aisha Tran'],

isActive: true,

balance: 2278,

gender: 'female',

age: 21

},

{

name: 'Carey Barr',

email: 'careybarr@nurali.com',

eyeColor: 'blue',

friends: ['Jordan Sampson', 'Eddie Strong'],

isActive: true,

balance: 3951,

gender: 'male',

age: 27

},

{

name: 'Blackburn Dotson',

email: 'blackburndotson@furnigeer.com',

eyeColor: 'brown',

friends: ['Jacklyn Lucas', 'Linda Chapman'],

isActive: false,

balance: 1498,

gender: 'male',

age: 38

},

{

name: 'Sheree Anthony',

email: 'shereeanthony@kog.com',

eyeColor: 'brown',

friends: ['Goldie Gentry', 'Briana Decker'],

isActive: true,

balance: 2764,

gender: 'female',

age: 39

}

]

**Задание**

Дополни функцию getUsersWithAge(users, minAge, maxAge) так, чтобы она возвращала массив пользователей, возраст которых (свойство age) попадает в промежуток от minAge до maxAge.

**Тесты**

* Объявлена переменная getUsersWithAge.
* Переменной getUsersWithAge присвоена стрелочная функция с параметрами (users, minAge, maxAge).
* Для перебора параметра users используется метод filter().
* Если значение параметров minAge и maxAge равны 20 и 30 соотвественно, функция возвращает массив объектов пользователей с именами Ross Vazquez, Elma Head и Carey Barr.
* Если значение параметров minAge и maxAge равны 30 и 40 соотвественно, функция возвращает массив объектов пользователей с именами Moore Hensley, Sharlene Bush, Blackburn Dotson, Sheree Anthony.
* Если значение параметров minAge и maxAge равны 80 и 100 соотвественно, функция возвращает пустой массив.
* Вызов функции со случайными, но валидными аргументами, возвращает правильное значение.

#20

// Пиши код ниже этой строки

const getUsersWithFriend = (users, friendName) => {

const userFriends = users.filter((user) => user.friends.includes(friendName));

return userFriends;

}

// Пиши код выше этой строки

**Задача. Пользователи с другом**

Массив объектов пользователей

Этот массив объектов мы будем передавать в параметр users при вызове функции из задания.

[

{

name: 'Moore Hensley',

email: 'moorehensley@indexia.com',

eyeColor: 'blue',

friends: ['Sharron Pace'],

isActive: false,

balance: 2811,

gender: 'male',

age: 37

},

{

name: 'Sharlene Bush',

email: 'sharlenebush@tubesys.com',

eyeColor: 'blue',

friends: ['Briana Decker', 'Sharron Pace'],

isActive: true,

balance: 3821,

gender: 'female',

age: 34

},

{

name: 'Ross Vazquez',

email: 'rossvazquez@xinware.com',

eyeColor: 'green',

friends: ['Marilyn Mcintosh', 'Padilla Garrison', 'Naomi Buckner'],

isActive: false,

balance: 3793,

gender: 'male',

age: 24

},

{

name: 'Elma Head',

email: 'elmahead@omatom.com',

eyeColor: 'green',

friends: ['Goldie Gentry', 'Aisha Tran'],

isActive: true,

balance: 2278,

gender: 'female',

age: 21

},

{

name: 'Carey Barr',

email: 'careybarr@nurali.com',

eyeColor: 'blue',

friends: ['Jordan Sampson', 'Eddie Strong'],

isActive: true,

balance: 3951,

gender: 'male',

age: 27

},

{

name: 'Blackburn Dotson',

email: 'blackburndotson@furnigeer.com',

eyeColor: 'brown',

friends: ['Jacklyn Lucas', 'Linda Chapman'],

isActive: false,

balance: 1498,

gender: 'male',

age: 38

},

{

name: 'Sheree Anthony',

email: 'shereeanthony@kog.com',

eyeColor: 'brown',

friends: ['Goldie Gentry', 'Briana Decker'],

isActive: true,

balance: 2764,

gender: 'female',

age: 39

}

]

**Задание**

Дополни функцию getUsersWithFriend(users, friendName) так, чтобы она возвращала массив пользователей у которых есть друг с именем в параметре friendName. Массив друзей пользователя хранится в свойстве friends.

**Тесты**

* Объявлена переменная getUsersWithFriend.
* Переменной getUsersWithFriend присвоена стрелочная функция с параметрами (users, friendName).
* Для перебора параметра users используется метод filter().
* Если значение параметра friendName это строка 'Briana Decker', функция возвращает массив объектов пользователей с именами Sharlene Bush и Sheree Anthony.
* Если значение параметра friendName это строка 'Goldie Gentry', функция возвращает массив объектов пользователей с именами Elma Head и Sheree Anthony.
* Если значение параметра friendName это строка 'Adrian Cross', функция возвращает пустой массив.
* Вызов функции со случайными, но валидными аргументами, возвращает правильное значение.

#21

// Пиши код ниже этой строки

const getFriends = (users) => {

const arr = users.flatMap(user => user.friends)

return arr.filter((item, index) => arr.indexOf(item) === index)

};

// Пиши код выше этой строки

**Задача. Список друзей**

Массив объектов пользователей

Этот массив объектов мы будем передавать в параметр users при вызове функции из задания.

[

{

name: 'Moore Hensley',

email: 'moorehensley@indexia.com',

eyeColor: 'blue',

friends: ['Sharron Pace'],

isActive: false,

balance: 2811,

gender: 'male',

age: 37

},

{

name: 'Sharlene Bush',

email: 'sharlenebush@tubesys.com',

eyeColor: 'blue',

friends: ['Briana Decker', 'Sharron Pace'],

isActive: true,

balance: 3821,

gender: 'female',

age: 34

},

{

name: 'Ross Vazquez',

email: 'rossvazquez@xinware.com',

eyeColor: 'green',

friends: ['Marilyn Mcintosh', 'Padilla Garrison', 'Naomi Buckner'],

isActive: false,

balance: 3793,

gender: 'male',

age: 24

},

{

name: 'Elma Head',

email: 'elmahead@omatom.com',

eyeColor: 'green',

friends: ['Goldie Gentry', 'Aisha Tran'],

isActive: true,

balance: 2278,

gender: 'female',

age: 21

},

{

name: 'Carey Barr',

email: 'careybarr@nurali.com',

eyeColor: 'blue',

friends: ['Jordan Sampson', 'Eddie Strong'],

isActive: true,

balance: 3951,

gender: 'male',

age: 27

},

{

name: 'Blackburn Dotson',

email: 'blackburndotson@furnigeer.com',

eyeColor: 'brown',

friends: ['Jacklyn Lucas', 'Linda Chapman'],

isActive: false,

balance: 1498,

gender: 'male',

age: 38

},

{

name: 'Sheree Anthony',

email: 'shereeanthony@kog.com',

eyeColor: 'brown',

friends: ['Goldie Gentry', 'Briana Decker'],

isActive: true,

balance: 2764,

gender: 'female',

age: 39

}

]

**Задание**

Дополни функцию getFriends(users) так, чтобы она возвращала массив друзей всех пользователей (свойство friends). У нескольких пользователей могут быть одинаковые друзья, сделай так чтобы возвращаемый массив не содержал повторений.

**Тесты**

* Объявлена переменная getFriends.
* Переменной getFriends присвоена стрелочная функция с параметром (users).
* Вызов функции с указанным массивом пользователей возвращает массив ['Sharron Pace', 'Briana Decker', 'Marilyn Mcintosh', 'Padilla Garrison', 'Naomi Buckner', 'Goldie Gentry', 'Aisha Tran', 'Jordan Sampson', 'Eddie Strong', 'Jacklyn Lucas', 'Linda Chapman'].
* Вызов функции со случайными, но валидными аргументами, возвращает правильное значение.

#22

// Пиши код ниже этой строки

const getActiveUsers = (users) => {

return users.filter((user) => user.isActive);

};

// Пиши код выше этой строки

**Задача. Активные пользователи**

Массив объектов пользователей

Этот массив объектов мы будем передавать в параметр users при вызове функции из задания.

[

{

name: 'Moore Hensley',

email: 'moorehensley@indexia.com',

eyeColor: 'blue',

friends: ['Sharron Pace'],

isActive: false,

balance: 2811,

gender: 'male',

age: 37

},

{

name: 'Sharlene Bush',

email: 'sharlenebush@tubesys.com',

eyeColor: 'blue',

friends: ['Briana Decker', 'Sharron Pace'],

isActive: true,

balance: 3821,

gender: 'female',

age: 34

},

{

name: 'Ross Vazquez',

email: 'rossvazquez@xinware.com',

eyeColor: 'green',

friends: ['Marilyn Mcintosh', 'Padilla Garrison', 'Naomi Buckner'],

isActive: false,

balance: 3793,

gender: 'male',

age: 24

},

{

name: 'Elma Head',

email: 'elmahead@omatom.com',

eyeColor: 'green',

friends: ['Goldie Gentry', 'Aisha Tran'],

isActive: true,

balance: 2278,

gender: 'female',

age: 21

},

{

name: 'Carey Barr',

email: 'careybarr@nurali.com',

eyeColor: 'blue',

friends: ['Jordan Sampson', 'Eddie Strong'],

isActive: true,

balance: 3951,

gender: 'male',

age: 27

},

{

name: 'Blackburn Dotson',

email: 'blackburndotson@furnigeer.com',

eyeColor: 'brown',

friends: ['Jacklyn Lucas', 'Linda Chapman'],

isActive: false,

balance: 1498,

gender: 'male',

age: 38

},

{

name: 'Sheree Anthony',

email: 'shereeanthony@kog.com',

eyeColor: 'brown',

friends: ['Goldie Gentry', 'Briana Decker'],

isActive: true,

balance: 2764,

gender: 'female',

age: 39

}

]

**Задание**

Дополни функцию getActiveUsers(users) так, чтобы она возвращала массив активных пользователей, значение свойства isActive которых true.

**Тесты**

* Объявлена переменная getActiveUsers. Переменной getActiveUsers присвоена стрелочная функция с параметром users.
* Для перебора параметра users используется метод filter().
* Вызов функции с указанным массивом пользователей возвращает массив объектов пользователей с именами Sharlene Bush, Elma Head, Carey Barr и Sheree Anthony.
* Вызов функции со случайными, но валидными аргументами, возвращает правильное значение.

#23

// Пиши код ниже этой строки

const getInactiveUsers = (users) => {

return users.filter(user => !user.isActive);

};

// Пиши код выше этой строки

**Задача. Неактивные пользователи**

Массив объектов пользователей

Этот массив объектов мы будем передавать в параметр users при вызове функции из задания.

[

{

name: 'Moore Hensley',

email: 'moorehensley@indexia.com',

eyeColor: 'blue',

friends: ['Sharron Pace'],

isActive: false,

balance: 2811,

gender: 'male',

age: 37

},

{

name: 'Sharlene Bush',

email: 'sharlenebush@tubesys.com',

eyeColor: 'blue',

friends: ['Briana Decker', 'Sharron Pace'],

isActive: true,

balance: 3821,

gender: 'female',

age: 34

},

{

name: 'Ross Vazquez',

email: 'rossvazquez@xinware.com',

eyeColor: 'green',

friends: ['Marilyn Mcintosh', 'Padilla Garrison', 'Naomi Buckner'],

isActive: false,

balance: 3793,

gender: 'male',

age: 24

},

{

name: 'Elma Head',

email: 'elmahead@omatom.com',

eyeColor: 'green',

friends: ['Goldie Gentry', 'Aisha Tran'],

isActive: true,

balance: 2278,

gender: 'female',

age: 21

},

{

name: 'Carey Barr',

email: 'careybarr@nurali.com',

eyeColor: 'blue',

friends: ['Jordan Sampson', 'Eddie Strong'],

isActive: true,

balance: 3951,

gender: 'male',

age: 27

},

{

name: 'Blackburn Dotson',

email: 'blackburndotson@furnigeer.com',

eyeColor: 'brown',

friends: ['Jacklyn Lucas', 'Linda Chapman'],

isActive: false,

balance: 1498,

gender: 'male',

age: 38

},

{

name: 'Sheree Anthony',

email: 'shereeanthony@kog.com',

eyeColor: 'brown',

friends: ['Goldie Gentry', 'Briana Decker'],

isActive: true,

balance: 2764,

gender: 'female',

age: 39

}

]

**Задание**

Дополни функцию getInactiveUsers(users) так, чтобы она возвращала массив неактивных пользователей, значение свойства isActive которых false.

**Тесты**

-Объявлена переменная getInactiveUsers.

* Переменной getInactiveUsers присвоена стрелочная функция с параметром users.
* Для перебора параметра users используется метод filter().
* Вызов функции с указанным массивом пользователей возвращает массив объектов пользователей с именами Moore Hensley, Ross Vazquez и Blackburn Dotson.
* Вызов функции со случайными, но валидными аргументами, возвращает правильное значение.

#24

const books = [

{ title: 'Последнее королевство', author: 'Бернард Корнуэлл', rating: 8.38 },

{ title: 'На берегу спокойных вод', author: 'Роберт Шекли', rating: 8.51 },

{ title: 'Сон смешного человека', author: 'Федор Достоевский', rating: 7.75 },

{ title: 'Красна как кровь', author: 'Ли Танит', rating: 7.94 },

];

const BOOK\_TITLE = 'Сон смешного человека';

const AUTHOR = 'Роберт Шекли';

// Пиши код ниже этой строки

const bookWithTitle = books.find((book) => book.title === BOOK\_TITLE);

const bookByAuthor = books.find((book) => book.author === AUTHOR);

**Метод find()**

Теория

Если метод filter(callack) используется для поиска всех элементов удовлетворяющим условию, то метод find(callback) позволяет найти и вернуть первый подходящий элемент, после чего перебор массива прекращается. То есть он ищет до первого совпадения.

массив.find((element, index, array) => {

// Тело коллбек-функции

});

* Не изменяет оригинальный массив.
* Поэлементно перебирает оригинальный массив.
* Возвращает первый элемент удовлетворяющий условию, то есть когда коллбек возвращает true.
* Если ни один элемент не подошёл, то есть для всех элементов коллбек вернул false, метод возвращает undefined.

Метод find() используется для одной задачи - поиска элемента по уникальному значению свойства. Например, поиск пользователя по почте, автомобиля по серийному номеру, книги по названию и т. п.

const colorPickerOptions = [

{ label: 'red', color: '#F44336' },

{ label: 'green', color: '#4CAF50' },

{ label: 'blue', color: '#2196F3' },

{ label: 'pink', color: '#E91E63' },

{ label: 'indigo', color: '#3F51B5' },

];

colorPickerOptions.find((option) => option.label === 'blue'); // { label: 'blue', color: '#2196F3' }

colorPickerOptions.find((option) => option.label === 'pink'); // { label: 'pink', color: '#E91E63' }

colorPickerOptions.find((option) => option.label === 'white'); // undefined

**Задание**

Используя метод find() дополни код так, чтобы:

* В переменной bookWithTitle получился объект книги название которой (свойство title) совпадает со значением переменной BOOK\_TITLE.
* В переменной bookByAuthor получился объект книги автор который (свойство author) совпадает со значением переменной AUTHOR.

**Тесты**

* Объявлена переменная books.
* Значение переменной books это массив.
* Объявлена переменная BOOK\_TITLE.
* Значение переменной BOOK\_TITLE это строка 'Сон смешного человека'.
* Объявлена переменная AUTHOR.
* Значение переменной AUTHOR это строка 'Роберт Шекли'.
* Объявлена переменная bookWithTitle.
* Значение переменной bookWithTitle это объект книги с названием выше 'Сон смешного человека'.
* Объявлена переменная bookByAuthor.
* Значение переменной bookByAuthor это объект книги автора 'Роберт Шекли'.
* Для перебора массива books использован метод find().

#25

// Пиши код ниже этой строки

const getUserWithEmail = (users, email) => {

return users.find(user => user.email === email)

};

// Пиши код выше этой строки

**Задача. Пользователь с почтой**

Массив объектов пользователей

Этот массив объектов мы будем передавать в параметр users при вызове функции из задания.

[

{

name: 'Moore Hensley',

email: 'moorehensley@indexia.com',

eyeColor: 'blue',

friends: ['Sharron Pace'],

isActive: false,

balance: 2811,

gender: 'male'

},

{

name: 'Sharlene Bush',

email: 'sharlenebush@tubesys.com',

eyeColor: 'blue',

friends: ['Briana Decker', 'Sharron Pace'],

isActive: true,

balance: 3821,

gender: 'female'

},

{

name: 'Ross Vazquez',

email: 'rossvazquez@xinware.com',

eyeColor: 'green',

friends: ['Marilyn Mcintosh', 'Padilla Garrison', 'Naomi Buckner'],

isActive: false,

balance: 3793,

gender: 'male'

},

{

name: 'Elma Head',

email: 'elmahead@omatom.com',

eyeColor: 'green',

friends: ['Goldie Gentry', 'Aisha Tran'],

isActive: true,

balance: 2278,

gender: 'female'

},

{

name: 'Carey Barr',

email: 'careybarr@nurali.com',

eyeColor: 'blue',

friends: ['Jordan Sampson', 'Eddie Strong'],

isActive: true,

balance: 3951,

gender: 'male'

},

{

name: 'Blackburn Dotson',

email: 'blackburndotson@furnigeer.com',

eyeColor: 'brown',

friends: ['Jacklyn Lucas', 'Linda Chapman'],

isActive: false,

balance: 1498,

gender: 'male'

},

{

name: 'Sheree Anthony',

email: 'shereeanthony@kog.com',

eyeColor: 'brown',

friends: ['Goldie Gentry', 'Briana Decker'],

isActive: true,

balance: 2764,

gender: 'female'

}

]

**Задание**

Дополни функцию getUserWithEmail(users, email) так, чтобы она возвращала объект пользователя, почта которого (свойство email) совпадает со значением параметра email.

**Тесты**

* Объявлена функция getUserWithEmail(users, email).
* Для перебора параметра users используется метод find().
* Если значение параметра email это 'shereeanthony@kog.com', функция возвращает объект пользователя с именем Sheree Anthony.
* Если значение параметра email это 'elmahead@omatom.com', функция возвращает объект пользователя с именем Elma Head.
* Если значение параметра email это 'blackburndotson@furnigeer.com', функция возвращает объект пользователя с именем Blackburn Dotson.
* Если в массиве users нет пользователя с почтой из параметра email, функция возвращает undefined.
* Вызов функции со случайными, но валидными аргументами, возвращает правильное значение.

#26

const firstArray = [26, 94, 36, 18];

const secondArray = [17, 61, 23];

const thirdArray = [17, 26, 94, 61, 36, 23, 18];

// Пиши код ниже этой строки

const eachElementInFirstIsEven = firstArray.every(item => item % 2 === 0);

const eachElementInFirstIsOdd = firstArray.every(item => item % 2 !== 0);

const eachElementInSecondIsEven = secondArray.every(item => item % 2 === 0);

const eachElementInSecondIsOdd = secondArray.every(item => item % 2 !== 0);

const eachElementInThirdIsEven = thirdArray.every(item => item % 2 === 0);

const eachElementInThirdIsOdd = thirdArray.every(item => item % 2 !== 0);

**Метод every()**

Теория

Метод every(callback) проверяет проходят ли все элементы массива тест предоставляемый коллбек-функцией. Возвращает true или false.

массив.every((element, index, array) => {

// Тело коллбек-функции

});

* Не изменяет оригинальный массив.
* Поэлементно перебирает оригинальный массив.
* Возвращает true если все элементы массива удовлетворяют условию.
* Возвращает false если хотя бы один элемент массива не удовлетворяет условию.
* Перебор массива прекращается если коллбек возвращает false.

// Все элементы больше либо равны нулю? - да

[1, 2, 3, 4, 5].every((value) => value >= 0); // true

// Все элементы больше либо равны нулю? - нет

[1, 2, 3, -10, 4, 5].every((value) => value >= 0); // false

При работе с массивом объектов проверяется значение какого-то их свойства.

**Задание**

Используя метод every() дополни код так, чтобы:

* В переменной eachElementInFirstIsEven был результат проверки всех элементов массива firstArray на чётность.
* В переменной eachElementInFirstIsOdd был результат проверки всех элементов массива firstArray на нечётность.
* В переменной eachElementInSecondIsEven был результат проверки всех элементов массива secondArray на чётность.
* В переменной eachElementInSecondIsOdd был результат проверки всех элементов массива secondArray на нечётность.
* В переменной eachElementInThirdIsEven был результат проверки всех элементов массива thirdArray на чётность.
* В переменной eachElementInThirdIsOdd был результат проверки всех элементов массива thirdArray на нечётность.

**Тесты**

* Объявлена переменная firstArray.
* Значение переменной firstArray это массив [26, 94, 36, 18].
* Объявлена переменная secondArray.
* Значение переменной secondArray это массив [17, 61, 23].
* Объявлена переменная thirdArray.
* Значение переменной thirdArray это массив [17, 26, 94, 61, 36, 23, 18].
* Объявлена переменная eachElementInFirstIsEven.
* Значение переменной eachElementInFirstIsEven это буль true.
* Объявлена переменная eachElementInFirstIsOdd.
* Значение переменной eachElementInFirstIsOdd это буль false.
* Объявлена переменная eachElementInSecondIsEven.
* Значение переменной eachElementInSecondIsEven это буль false.
* Объявлена переменная eachElementInSecondIsOdd.
* Значение переменной eachElementInSecondIsOdd это буль true.
* Объявлена переменная eachElementInThirdIsEven.
* Значение переменной eachElementInThirdIsEven это буль false.
* Объявлена переменная eachElementInThirdIsOdd.
* Значение переменной eachElementInThirdIsOdd это буль false.
* Для перебора массивов использован метод every().

#27

// Пиши код ниже этой строки

const isEveryUserActive = (users) => {

return users.every(user => user.isActive)

};

// Пиши код выше этой строки

**Задача. Все ли пользователи активны**

Массив объектов пользователей

Этот массив объектов мы будем передавать в параметр users при вызове функции из задания.

[

{

name: 'Moore Hensley',

email: 'moorehensley@indexia.com',

eyeColor: 'blue',

friends: ['Sharron Pace'],

isActive: false,

balance: 2811,

gender: 'male'

},

{

name: 'Sharlene Bush',

email: 'sharlenebush@tubesys.com',

eyeColor: 'blue',

friends: ['Briana Decker', 'Sharron Pace'],

isActive: true,

balance: 3821,

gender: 'female'

},

{

name: 'Ross Vazquez',

email: 'rossvazquez@xinware.com',

eyeColor: 'green',

friends: ['Marilyn Mcintosh', 'Padilla Garrison', 'Naomi Buckner'],

isActive: false,

balance: 3793,

gender: 'male'

},

{

name: 'Elma Head',

email: 'elmahead@omatom.com',

eyeColor: 'green',

friends: ['Goldie Gentry', 'Aisha Tran'],

isActive: true,

balance: 2278,

gender: 'female'

},

{

name: 'Carey Barr',

email: 'careybarr@nurali.com',

eyeColor: 'blue',

friends: ['Jordan Sampson', 'Eddie Strong'],

isActive: true,

balance: 3951,

gender: 'male'

},

{

name: 'Blackburn Dotson',

email: 'blackburndotson@furnigeer.com',

eyeColor: 'brown',

friends: ['Jacklyn Lucas', 'Linda Chapman'],

isActive: false,

balance: 1498,

gender: 'male'

},

{

name: 'Sheree Anthony',

email: 'shereeanthony@kog.com',

eyeColor: 'brown',

friends: ['Goldie Gentry', 'Briana Decker'],

isActive: true,

balance: 2764,

gender: 'female'

}

]

**Задание**

Дополни функцию isEveryUserActive(users) так, чтобы она проверяла все ли пользователи сейчас активны (свойство isActive) и возвращала true или false.

**Тесты**

* Объявлена переменная isEveryUserActive
* Переменной isEveryUserActive присвоена стрелочная функция с параметром (users).
* Для перебора параметра users используется метод every().
* Вызов функции с указанным массивом пользователей возвращает false.
* Вызов функции со случайными, но валидными аргументами, возвращает правильное значение.

#28

const firstArray = [26, 94, 36, 18];

const secondArray = [17, 61, 23];

const thirdArray = [17, 26, 94, 61, 36, 23, 18];

// Пиши код ниже этой строки

const anyElementInFirstIsEven = firstArray.some(item => item % 2 === 0);

const anyElementInFirstIsOdd = firstArray.some(item => item % 2 !== 0);

const anyElementInSecondIsEven = secondArray.some(item => item % 2 === 0);

const anyElementInSecondIsOdd = secondArray.some(item => item % 2 !== 0);

const anyElementInThirdIsEven = thirdArray.some(item => item % 2 === 0);

const anyElementInThirdIsOdd = thirdArray.some(item => item % 2 !== 0);

**Метод some()**

Теория

Метод some(callback) проверяет проходит ли хотя бы один элемент массива тест предоставляемый коллбек-функцией. Возвращает true или false.

массив.some((element, index, array) => {

// Тело коллбек-функции

});

* Не изменяет оригинальный массив.
* Поэлементно перебирает оригинальный массив.
* Возвращает true если хотя бы один элемент массива удовлетворяет условию.
* Возвращает false если ни один элемент массив не удовлетворяет условию.
* Перебор массива прекращается если коллбек возвращает true.

// Есть хотя бы один элемент больше либо равный нулю? - да

[1, 2, 3, 4, 5].some(value => value >= 0); // true

// Есть хотя бы один элемент больше либо равный нулю? - да

[-7, -20, 3, -10, -14].some(value => value >= 0); // true

// Есть хотя бы один элемент меньше нуля? - нет

[1, 2, 3, 4, 5].some(value => value < 0); // false

// Есть хотя бы один элемент меньше нуля? - да

[1, 2, 3, -10, 4, 5].some(value => value < 0); // true

**Задание**

Используя метод some() дополни код так, чтобы:

* В переменной anyElementInFirstIsEven был результат проверки наличия чётных элементов в массиве firstArray.
* В переменной anyElementInFirstIsOdd был результат проверки наличия нечётных элементов в массиве firstArray.
* В переменной anyElementInSecondIsEven был результат проверки наличия чётных элементов в массиве secondArray.
* В переменной anyElementInSecondIsOdd был результат проверки наличия нечётных элементов в массиве secondArray.
* В переменной anyElementInThirdIsEven был результат проверки наличия чётных элементов в массиве thirdArray.
* В переменной anyElementInThirdIsOdd был результат проверки наличия нечётных элементов в массиве thirdArray.

**Тесты**

* Объявлена переменная firstArray.
* Значение переменной firstArray это массив [26, 94, 36, 18].
* Объявлена переменная secondArray.
* Значение переменной secondArray это массив [17, 61, 23].
* Объявлена переменная thirdArray.
* Значение переменной thirdArray это массив [17, 26, 94, 61, 36, 23, 18].
* Объявлена переменная anyElementInFirstIsEven.
* Значение переменной anyElementInFirstIsEven это буль true.
* Объявлена переменная anyElementInFirstIsOdd.
* Значение переменной anyElementInFirstIsOdd это буль false.
* Объявлена переменная anyElementInSecondIsEven.
* Значение переменной anyElementInSecondIsEven это буль false.
* Объявлена переменная anyElementInSecondIsOdd.
* Значение переменной anyElementInSecondIsOdd это буль true.
* Объявлена переменная anyElementInThirdIsEven.
* Значение переменной anyElementInThirdIsEven это буль true.
* Объявлена переменная anyElementInThirdIsOdd.
* Значение переменной anyElementInThirdIsOdd это буль true.
* Для перебора массивов использован метод some().

#29

// Пиши код ниже этой строки

const isAnyUserActive = users => {

return users.some(item => item.isActive)

};

// Пиши код выше этой строки

**Задача. Есть ли активные пользователи**

Массив объектов пользователей

Этот массив объектов мы будем передавать в параметр users при вызове функции из задания.

[

{

name: 'Moore Hensley',

email: 'moorehensley@indexia.com',

eyeColor: 'blue',

friends: ['Sharron Pace'],

isActive: false,

balance: 2811,

gender: 'male'

},

{

name: 'Sharlene Bush',

email: 'sharlenebush@tubesys.com',

eyeColor: 'blue',

friends: ['Briana Decker', 'Sharron Pace'],

isActive: true,

balance: 3821,

gender: 'female'

},

{

name: 'Ross Vazquez',

email: 'rossvazquez@xinware.com',

eyeColor: 'green',

friends: ['Marilyn Mcintosh', 'Padilla Garrison', 'Naomi Buckner'],

isActive: false,

balance: 3793,

gender: 'male'

},

{

name: 'Elma Head',

email: 'elmahead@omatom.com',

eyeColor: 'green',

friends: ['Goldie Gentry', 'Aisha Tran'],

isActive: true,

balance: 2278,

gender: 'female'

},

{

name: 'Carey Barr',

email: 'careybarr@nurali.com',

eyeColor: 'blue',

friends: ['Jordan Sampson', 'Eddie Strong'],

isActive: true,

balance: 3951,

gender: 'male'

},

{

name: 'Blackburn Dotson',

email: 'blackburndotson@furnigeer.com',

eyeColor: 'brown',

friends: ['Jacklyn Lucas', 'Linda Chapman'],

isActive: false,

balance: 1498,

gender: 'male'

},

{

name: 'Sheree Anthony',

email: 'shereeanthony@kog.com',

eyeColor: 'brown',

friends: ['Goldie Gentry', 'Briana Decker'],

isActive: true,

balance: 2764,

gender: 'female'

}

]

**Задание**

Дополни функцию isAnyUserActive(users) так, чтобы она проверяла наличие активных пользователей (свойство isActive) и возвращала true или false.

**Тесты**

* Объявлена функция isAnyUserActive(users).
* Для перебора параметра users используется метод some().
* Вызов функции с указанным массивом пользователей возвращает true.
* Вызов функции со случайными, но валидными аргументами, возвращает правильное значение.

#30

const players = {

mango: 1270,

poly: 468,

ajax: 710,

kiwi: 244

};

const playtimes = Object.values(players); // [1270, 468, 710, 244]

// Пиши код ниже этой строки

const totalPlayTime = playtimes.reduce((acc, value) => acc + value, 0);

// Пиши код выше этой строки

const averagePlayTime = totalPlayTime / playtimes.length;

**Метод reduce()**

Теория

Метод reduce(callback, initialValue) используется для последовательной обработки каждого элемента массива с сохранением промежуточного результата, как аккумулятор. Немного сложнее других в усвоении, но результат стоит того.

массив.reduce((previousValue, element, index, array) => {

// Тело коллбек-функции

}, initialValue);

* Не изменяет оригинальный массив.
* Поэлементно перебирает оригинальный массив.
* Возвращает что угодно.
* Делает что угодно.

Легче всего представить его работу на примере подсчёта суммы элементов массива.

const total = [2, 7, 3, 14, 6].reduce((previousValue, number) => {

return previousValue + number;

}, 0);

console.log(total); // 32

Первый параметр коллбек-функции (previousValue) это аккумулятор, то есть промежуточный результат. Значение которое вернёт коллбек-функция на текущей итерации, будет значением этого параметра на следующей.

Вторым аргументом для reduce() можно передать необязательное начальное значение аккумулятора - параметр initialValue.

# Вначале метод reduce() создаёт внутреннюю переменную-аккумулятор и

# присваивает ей значение параметра initialValue или первого элемента

# перебираемого массива, если initialValue не задан.

previousValue = 0

# Далее коллбек-функция вызывается для каждого элемента массива. Текущее значение

# параметра previousValue это то, что вернула коллбек-функция на прошлой итерации.

Итерация 1 -> previousValue = 0 -> number = 2 -> return 0 + 2 -> return 2

Итерация 2 -> previousValue = 2 -> number = 7 -> return 2 + 7 -> return 9

Итерация 3 -> previousValue = 9 -> number = 3 -> return 9 + 3 -> return 12

Итерация 4 -> previousValue = 12 -> number = 14 -> return 12 + 14 -> return 26

Итерация 5 -> previousValue = 26 -> number = 6 -> return 26 + 6 -> return 32

# После того как весь массив перебран, метод reduce() возвращает значение аккумулятора.

Результат - 32

То есть метод reduce() используется когда необходимо взять «много» и привести к «одному». В повседневных задачах его применение сводится к работе с числами.

**Задание**

Игровому сервису необходим функционал подсчёта среднего времени проведённого в играх. Дополни код так, чтобы в переменной totalPlayTime получилось общее игровое время из массива playtimes.

**Тесты**

* Объявлена переменная players.
* Значение переменной players это объект игроков с игровым временем каждого.
* Объявлена переменная playtimes.
* Значение переменной playtimes это массив [1270, 468, 710, 244].
* Объявлена переменная totalPlayTime.
* Значение переменной totalPlayTime это число 2692.
* Для перебора массива playtimes используется метод reduce().
* Объявлена переменная averagePlayTime.
* Значение переменной averagePlayTime это число 673.

#31

const players = [

{ name: 'Манго', playtime: 1270, gamesPlayed: 4 },

{ name: 'Поли', playtime: 469, gamesPlayed: 2 },

{ name: 'Аякс', playtime: 690, gamesPlayed: 3 },

{ name: 'Киви', playtime: 241, gamesPlayed: 1 },

];

// Пиши код ниже этой строки

const totalAveragePlaytimePerGame = players.reduce((total, player) => {

return total + player.playtime / player.gamesPlayed;

}, 0);

**Метод reduce() и массив объектов**

Теория

При работе с массивом объектов выполняется редуцирование по значению какого-то свойства. Например, есть массив студентов с баллами за тест. Необходимо получить средний бал.

const students = [

{ name: 'Манго', score: 83 },

{ name: 'Поли', score: 59 },

{ name: 'Аякс', score: 37 },

{ name: 'Киви', score: 94 },

{ name: 'Хьюстон', score: 64 },

];

// Название аккумулятора может быть произвольным, это просто параметр функции

const totalScore = students.reduce((total, student) => {

return total + student.score;

}, 0);

const averageScore = totalScore / students.length;

**Задание**

Нашему сервису необходимо рассчитать среднее время проведённое в одной игре для каждого игрока, и получить общую сумму этих времён. Рассчитать время для каждого из игроков, можно разделив его время (свойство playtime) на количество игр (свойство gamesPlayed).

**Тесты**

* Объявлена переменная players.
* Значение переменной players это массив объектов игроков.
* Объявлена переменная totalAveragePlaytimePerGame.
* Значение переменной totalAveragePlaytimePerGame это число 1023.
* Для перебора массива players используется метод reduce().

#32

// Пиши код ниже этой строки

const calculateTotalBalance = users => users.reduce((total, user) => total + user.balance, 0);

// Пиши код выше этой строки

**Задача. Общий баланс пользователей**

Массив объектов пользователей

Этот массив объектов мы будем передавать в параметр users при вызове функции из задания.

[

{

name: 'Moore Hensley',

email: 'moorehensley@indexia.com',

eyeColor: 'blue',

friends: ['Sharron Pace'],

isActive: false,

balance: 2811,

gender: 'male'

},

{

name: 'Sharlene Bush',

email: 'sharlenebush@tubesys.com',

eyeColor: 'blue',

friends: ['Briana Decker', 'Sharron Pace'],

isActive: true,

balance: 3821,

gender: 'female'

},

{

name: 'Ross Vazquez',

email: 'rossvazquez@xinware.com',

eyeColor: 'green',

friends: ['Marilyn Mcintosh', 'Padilla Garrison', 'Naomi Buckner'],

isActive: false,

balance: 3793,

gender: 'male'

},

{

name: 'Elma Head',

email: 'elmahead@omatom.com',

eyeColor: 'green',

friends: ['Goldie Gentry', 'Aisha Tran'],

isActive: true,

balance: 2278,

gender: 'female'

},

{

name: 'Carey Barr',

email: 'careybarr@nurali.com',

eyeColor: 'blue',

friends: ['Jordan Sampson', 'Eddie Strong'],

isActive: true,

balance: 3951,

gender: 'male'

},

{

name: 'Blackburn Dotson',

email: 'blackburndotson@furnigeer.com',

eyeColor: 'brown',

friends: ['Jacklyn Lucas', 'Linda Chapman'],

isActive: false,

balance: 1498,

gender: 'male'

},

{

name: 'Sheree Anthony',

email: 'shereeanthony@kog.com',

eyeColor: 'brown',

friends: ['Goldie Gentry', 'Briana Decker'],

isActive: true,

balance: 2764,

gender: 'female'

}

]

**Задание**

Дополни функцию calculateTotalBalance(users) так, чтобы она считала и возвращала сумму всех средств (свойство balance) которые хранят пользователи из массива users.

**Тесты**

* Объявлена переменная calculateTotalBalance.
* Переменной calculateTotalBalance присвоена стрелочная функция с параметром (users).
* Для перебора параметра users используется метод reduce().
* Вызов функции с указанным массивом пользователей возвращает число 20916.
* Вызов функции со случайными, но валидными аргументами, возвращает правильное значение.

#33

// Пиши код ниже этой строки

const getTotalFriendCount = users => users.reduce((total, user) => total + user.friends.length, 0);

// Пиши код выше этой строки

**Задача. Общее количество друзей**

Массив объектов пользователей

Этот массив объектов мы будем передавать в параметр users при вызове функции из задания.

[

{

name: 'Moore Hensley',

email: 'moorehensley@indexia.com',

eyeColor: 'blue',

friends: ['Sharron Pace'],

isActive: false,

balance: 2811,

gender: 'male'

},

{

name: 'Sharlene Bush',

email: 'sharlenebush@tubesys.com',

eyeColor: 'blue',

friends: ['Briana Decker', 'Sharron Pace'],

isActive: true,

balance: 3821,

gender: 'female'

},

{

name: 'Ross Vazquez',

email: 'rossvazquez@xinware.com',

eyeColor: 'green',

friends: ['Marilyn Mcintosh', 'Padilla Garrison', 'Naomi Buckner'],

isActive: false,

balance: 3793,

gender: 'male'

},

{

name: 'Elma Head',

email: 'elmahead@omatom.com',

eyeColor: 'green',

friends: ['Goldie Gentry', 'Aisha Tran'],

isActive: true,

balance: 2278,

gender: 'female'

},

{

name: 'Carey Barr',

email: 'careybarr@nurali.com',

eyeColor: 'blue',

friends: ['Jordan Sampson', 'Eddie Strong'],

isActive: true,

balance: 3951,

gender: 'male'

},

{

name: 'Blackburn Dotson',

email: 'blackburndotson@furnigeer.com',

eyeColor: 'brown',

friends: ['Jacklyn Lucas', 'Linda Chapman'],

isActive: false,

balance: 1498,

gender: 'male'

},

{

name: 'Sheree Anthony',

email: 'shereeanthony@kog.com',

eyeColor: 'brown',

friends: ['Goldie Gentry', 'Briana Decker'],

isActive: true,

balance: 2764,

gender: 'female'

}

]

**Задание**

Дополни функцию getTotalFriendCount(users) так, чтобы она считала и возвращала общее количество друзей (свойство friends) всех пользователей из массива users.

**Тесты**

* Объявлена переменная getTotalFriendCount.
* Переменной getTotalFriendCount присвоена стрелочная функция с параметром (users).
* Для перебора параметра users используется метод reduce().
* Вызов функции с указанным массивом пользователей возвращает число 14
* Вызов функции со случайными, но валидными аргументами, возвращает правильное значение.

#34

const releaseDates = [2016, 1967, 2008, 1984, 1973, 2012, 1997];

const authors = [

'Ли Танит',

'Бернард Корнуэлл',

'Роберт Шекли',

'Федор Достоевский'

];

// Пиши код ниже этой строки

const ascendingReleaseDates = [...releaseDates].sort();

const alphabeticalAuthors = [...authors].sort();

**Метод sort()**

Теория

Метод sort() сортирует элементы массива, но в отличии от остальных перебирающих методов, он сортирует исходный массив.

* Сортирует и изменяет исходный массив.
* Возвращает изменённый массив, то есть ссылку на отсортированный исходный.
* По умолчанию сортирует по возрастанию.
* Сравнение происходит путём приведения значений к строке и сравнения порядковых номеров в таблице Unicode.

// Массив чисел будет отсортирован по возврастанию

const scores = [61, 19, 74, 35, 92, 56];

scores.sort();

console.log(scores); // [19, 35, 56, 61, 74, 92]

// Массив строк сортируется по алфавиту

const students = ["Вика", "Андрей", "Олег", "Юля", "Борис", "Катя"];

students.sort();

console.log(students); // [ 'Андрей', 'Борис', 'Вика', 'Катя', 'Олег', 'Юля' ]

// При этом порядковый номер заглавных букв меньше чем у прописных

const letters = ["b", "B", "a", "A", "c", "C"];

letters.sort();

console.log(letters); // ['A', 'B', 'C', 'a', 'b', 'c']

Из-за того, что сортируется исходный массив, нарушается принцип чистоты функций и нельзя удобно сделать несколько производных коллекций на базе исходной. Например, сделать коллекцию отсортированную по возрастанию, а другую по убыванию. Поэтому перед сортировкой делают полную копию исходного массива и сортируют уже её.

const scores = [61, 19, 74, 35, 92, 56];

const ascendingScores = [...scores].sort();

console.log(scores); // [61, 19, 74, 35, 92, 56]

console.log(ascendingScores); // [19, 35, 56, 61, 74, 92]

**Задание**

Дополни код так, чтобы в переменной ascendingReleaseDates получилась отсортированная по возрастанию копия массива releaseDates, а в переменной alphabeticalAuthors копия массива имён авторов authors отсортированная в по алфавиту.

**Тесты**

* Объявлена переменная releaseDates.
* Значение переменной releaseDates это массив [2016, 1967, 2008, 1984, 1973, 2012, 1997].
* Объявлена переменная authors.
* Значение переменной authors это массив ['Ли Танит', 'Бернард Корнуэлл', 'Роберт Шекли', 'Федор Достоевский'].
* Объявлена переменная ascendingReleaseDates.
* Значение переменной ascendingReleaseDates это массив [1967, 1973, 1984, 1997, 2008, 2012, 2016].
* Объявлена переменная alphabeticalAuthors.
* Значение переменной alphabeticalAuthors это массив ['Бернард Корнуэлл', 'Ли Танит', 'Роберт Шекли', 'Федор Достоевский'].
* Использован метод sort().

#35

const releaseDates = [2016, 1967, 2008, 1984, 1973, 2012, 1997];

// Пиши код ниже этой строки

const ascendingReleaseDates = [...releaseDates].sort((a,b)=> a-b);

const descendingReleaseDates = [...releaseDates].sort((a,b)=> b-a);

**Свой порядок сортировки чисел**

Теория

Для указания своего порядка сортировки методу sort(callback) нужно передать коллбек-функцию с двумя параметрами. Это функция сравнения (compare function), порядок сортировки зависит от её результата. Метод sort() будет вызывать её для произвольных двух элементов.

массив.sort((firstEl, secondEl) => {

// Тело коллбек-функции

});

* firstEl - первый элемент для сравнения.
* secondEl - второй элемент для сравнения.

Если вызов compareFunction(firstEl, secondEl) возвращает любое отрицательное значение, то есть firstEl меньше secondEl, сортировка поставит firstEl перед secondEl. Это сортировка по возрастанию.

const scores = [61, 19, 74, 35, 92, 56];

const ascendingScores = [...scores].sort((a, b) => a - b);

console.log(ascendingScores); // [19, 35, 56, 61, 74, 92]

Если вызов compareFunction(firstEl, secondEl) возвращает любое положительное значение больше нуля, то есть secondEl больше firstEl, сортировка поставит secondEl перед firstEl. Это сортировка по убыванию.

const scores = [61, 19, 74, 35, 92, 56];

const descendingScores = [...scores].sort((a, b) => b - a);

console.log(descendingScores); // [92, 74, 61, 56, 35, 19]

Eсли вызов compareFunction(firstEl, secondEl) вернёт 0, сортировка оставит firstEl и secondEl неизменными по отношению друг к другу, но отсортирует их по отношению ко всем другим элементам. Но вообще не важно, что возвращать, если их взаимный порядок не имеет значения.

**Задание**

Онлайн бибилиотеке необходимо отображать книги сортированные по дате издания, по её возрастанию или убыванию. Дополни код так, чтобы в переменной ascendingReleaseDates получилась отсортированная по возрастанию копия массива releaseDates, а в переменной descendingReleaseDates копия отсортированная по убыванию.

**Тесты**

* Объявлена переменная releaseDates.
* Значение переменной releaseDates это массив [2016, 1967, 2008, 1984, 1973, 2012, 1997].
* Объявлена переменная ascendingReleaseDates.
* Значение переменной ascendingReleaseDates это массив [1967, 1973, 1984, 1997, 2008, 2012, 2016].
* Объявлена переменная descendingReleaseDates.
* Значение переменной descendingReleaseDates это массив [2016, 2012, 2008, 1997, 1984, 1973, 1967].
* Использован метод sort().

#36

const authors = [

'Ли Танит',

'Бернард Корнуэлл',

'Роберт Шекли',

'Федор Достоевский',

'Говард Лавкрафт'

];

// Пиши код ниже этой строки

const authorsInAlphabetOrder = [...authors].sort((a,b)=> a.localeCompare(b));

const authorsInReversedOrder = [...authors].sort((a,b) => b.localeCompare(a));

**Свой порядок сортировки строк**

Теория

Для сортировки строк в алфавитном порядке, по возрастанию или убыванию, используется метод строк localeCompare().

firstString.localeCompare(secondString)

Он вызывается на строке которую нужно сравнить (firstString) с той, что передана ему как аргумент (secondString).

'a'.localeCompare('b'); // -1

'b'.localeCompare('a'); // 1

'a'.localeCompare('a'); // 0

'b'.localeCompare('b'); // 0

* Возвращает отрицательное значение если firstString должна быть перед secondString.
* Возвращает положительное значение больше нуля если firstString должна быть после secondString.
* Если строки одинаковы, возвращается ноль.

Это удобно использовать при сортировке строк, так как метод sort() ожидает такие же значения от коллбек-функции.

const students = ['Вика', 'Андрей', 'Олег', 'Юля', 'Борис', 'Катя'];

const inAlphabetOrder = [...students].sort((a, b) => a.localeCompare(b));

console.log(inAlphabetOrder); // ['Андрей', 'Борис', 'Вика', 'Катя', 'Олег', 'Юля']

const inReversedOrder = [...students].sort((a, b) => b.localeCompare(a));

console.log(inReversedOrder); // ['Юля', 'Олег', 'Катя', 'Вика', 'Борис', 'Андрей']

**Задание**

Онлайн бибилиотеке необходимо отображать книги отсортированные по автору, в алфавитном и обратном алфавитном порядке. Дополни код так, чтобы в переменной authorsInAlphabetOrder получилась отсортированная по алфавиту копия массива authors, а в переменной authorsInReversedOrder копия отсортированная в обратном алфавитном порядке.

**Тесты**

* Объявлена переменная authors.
* Значение переменной authors это массив ['Ли Танит', 'Бернард Корнуэлл', 'Роберт Шекли', 'Федор Достоевский', 'Говард Лавкрафт'].
* Объявлена переменная authorsInAlphabetOrder.
* Значение переменной authorsInAlphabetOrder это массив ['Бернард Корнуэлл', 'Говард Лавкрафт', 'Ли Танит', 'Роберт Шекли', 'Федор Достоевский'].
* Объявлена переменная authorsInReversedOrder.
* Значение переменной authorsInReversedOrder это массив ['Федор Достоевский', 'Роберт Шекли', 'Ли Танит', 'Говард Лавкрафт', 'Бернард Корнуэлл'].
* Использован метод sort().

#37

const books = [

{ title: 'Последнее королевство', author: 'Бернард Корнуэлл', rating: 8.38 },

{ title: 'На берегу спокойных вод', author: 'Роберт Шекли', rating: 8.51 },

{ title: 'Сон смешного человека', author: 'Федор Достоевский', rating: 7.75 },

{ title: 'Красна как кровь', author: 'Ли Танит', rating: 7.94 },

{ title: 'Враг Божий', author: 'Бернард Корнуэлл', rating: 8.67 }

];

// Пиши код ниже этой строки

const sortedByAuthorName = [...books].sort((a, b) => {

return a.author[0] >= b.author[0] ? 1 : -1;

});

console.table(sortedByAuthorName);

const sortedByReversedAuthorName = [...books].sort((a, b) => {

return b.author[0] >= a.author[0] ? 1 : -1;

});

console.table(sortedByReversedAuthorName);

const sortedByAscendingRating = [...books].sort((a, b) => a.rating - b.rating);

console.log(sortedByAscendingRating);

const sortedByDescentingRating = [...books].sort((a, b) => b.rating - a.rating);

console.log(sortedByDescentingRating);

**Сортировка объектов**

Теория

При работе с массивом объектов сортировка выполняется по числовому или строчному значению какого-то свойства. Например, есть группа студентов с баллами за тест. Необходимо отсортировать массив объектов по возрастанию и убыванию количества баллов, и по имени студента.

const students = [

{ name: 'Манго', score: 83 },

{ name: 'Поли', score: 59 },

{ name: 'Аякс', score: 37 },

{ name: 'Киви', score: 94 },

];

const inAscendingScoreOrder = students.sort(

(firstStudent, secondStudent) => firstStudent.score - secondStudent.score

);

const inDescendingScoreOrder = students.sort(

(firstStudent, secondStudent) => secondStudent.score - firstStudent.score

);

const inAlphabeticalOrder = students.sort((firstStudent, secondStudent) =>

firstStudent.name.localeCompare(secondStudent.name)

);

**Задание**

Дополни код так, чтобы:

* В переменной sortedByAuthorName получился массив книг отсортированный по имени автора в алфавитном порядке.
* В переменной sortedByReversedAuthorName получился массив книг отсортированный по имени автора в обратном алфавитном порядке.
* В переменной sortedByAscendingRating получился массив книг отсортированный по возрастанию рейтинга.
* В переменной sortedByDescentingRating получился массив книг отсортированный по убыванию рейтинга.

**Тесты**

* Объявлена переменная books.
* Значение переменной books это исходный массив объектов книг.
* Объявлена переменная sortedByAuthorName.
* Значение переменной sortedByAuthorName это массив книг отсортированный по имени автора в алфавитном порядке.
* Объявлена переменная sortedByReversedAuthorName.
* Значение переменной sortedByReversedAuthorName это массив книг отсортированный по имени автора в обратном алфавитном порядке.
* Объявлена переменная sortedByAscendingRating.
* Значение переменной sortedByAscendingRating это массив книг отсортированный по возрастанию рейтинга.
* Объявлена переменная sortedByDescentingRating.
* Значение переменной sortedByDescentingRating это массив книг отсортированный по убыванию рейтинга.
* Для перебора массива books используется метод sort().

#38

// Пиши код ниже этой строки

const sortByAscendingBalance = users => {

const arr = [...users];

return arr.sort((a, b) => a.balance - b.balance);

};

// Пиши код выше этой строки

**Задача. Сортировка по балансу**

Массив объектов пользователей

Этот массив объектов мы будем передавать в параметр users при вызове функции из задания.

[

{

name: 'Moore Hensley',

email: 'moorehensley@indexia.com',

eyeColor: 'blue',

friends: ['Sharron Pace'],

isActive: false,

balance: 2811,

gender: 'male'

},

{

name: 'Sharlene Bush',

email: 'sharlenebush@tubesys.com',

eyeColor: 'blue',

friends: ['Briana Decker', 'Sharron Pace'],

isActive: true,

balance: 3821,

gender: 'female'

},

{

name: 'Ross Vazquez',

email: 'rossvazquez@xinware.com',

eyeColor: 'green',

friends: ['Marilyn Mcintosh', 'Padilla Garrison', 'Naomi Buckner'],

isActive: false,

balance: 3793,

gender: 'male'

},

{

name: 'Elma Head',

email: 'elmahead@omatom.com',

eyeColor: 'green',

friends: ['Goldie Gentry', 'Aisha Tran'],

isActive: true,

balance: 2278,

gender: 'female'

},

{

name: 'Carey Barr',

email: 'careybarr@nurali.com',

eyeColor: 'blue',

friends: ['Jordan Sampson', 'Eddie Strong'],

isActive: true,

balance: 3951,

gender: 'male'

},

{

name: 'Blackburn Dotson',

email: 'blackburndotson@furnigeer.com',

eyeColor: 'brown',

friends: ['Jacklyn Lucas', 'Linda Chapman'],

isActive: false,

balance: 1498,

gender: 'male'

},

{

name: 'Sheree Anthony',

email: 'shereeanthony@kog.com',

eyeColor: 'brown',

friends: ['Goldie Gentry', 'Briana Decker'],

isActive: true,

balance: 2764,

gender: 'female'

}

]

**Задание**

Дополни функцию sortByAscendingBalance(users) так, чтобы она возвращала массив пользователей отсортированный по возрастанию их баланса (свойство balance).

**Тесты**

* Объявлена переменная sortByAscendingBalance.
* Переменной sortByAscendingBalance присвоена стрелочная функция с параметром (users).
* Значение параметра users не изменяется.
* Вызов функции с указанным массивом пользователей возвращает новый массив пользователей отсортированный по возрастанию их баланса.
* Вызов функции со случайными, но валидными аргументами, возвращает правильное значение.
* Для перебора параметра users использован метод sort().

#39

// Пиши код ниже этой строки

const sortByDescendingFriendCount = users => {

return [...users].sort((a, b) => b.friends.length - a.friends.length);

};

// Пиши код выше этой строки

**Задача. Сортировка по количеству друзей**

Массив объектов пользователей

Этот массив объектов мы будем передавать в параметр users при вызове функции из задания.

[

{

name: 'Moore Hensley',

email: 'moorehensley@indexia.com',

eyeColor: 'blue',

friends: ['Sharron Pace'],

isActive: false,

balance: 2811,

gender: 'male'

},

{

name: 'Sharlene Bush',

email: 'sharlenebush@tubesys.com',

eyeColor: 'blue',

friends: ['Briana Decker', 'Sharron Pace'],

isActive: true,

balance: 3821,

gender: 'female'

},

{

name: 'Ross Vazquez',

email: 'rossvazquez@xinware.com',

eyeColor: 'green',

friends: ['Marilyn Mcintosh', 'Padilla Garrison', 'Naomi Buckner'],

isActive: false,

balance: 3793,

gender: 'male'

},

{

name: 'Elma Head',

email: 'elmahead@omatom.com',

eyeColor: 'green',

friends: ['Goldie Gentry', 'Aisha Tran'],

isActive: true,

balance: 2278,

gender: 'female'

},

{

name: 'Carey Barr',

email: 'careybarr@nurali.com',

eyeColor: 'blue',

friends: ['Jordan Sampson', 'Eddie Strong'],

isActive: true,

balance: 3951,

gender: 'male'

},

{

name: 'Blackburn Dotson',

email: 'blackburndotson@furnigeer.com',

eyeColor: 'brown',

friends: ['Jacklyn Lucas', 'Linda Chapman'],

isActive: false,

balance: 1498,

gender: 'male'

},

{

name: 'Sheree Anthony',

email: 'shereeanthony@kog.com',

eyeColor: 'brown',

friends: ['Goldie Gentry', 'Briana Decker'],

isActive: true,

balance: 2764,

gender: 'female'

}

]

**Задание**

Дополни функцию sortByDescendingFriendCount(users) так, чтобы она возвращала массив пользователей отсортированный по убыванию количества их друзей (свойство friends).

**Тесты**

* Объявлена переменная sortByDescendingFriendCount.
* Переменной sortByDescendingFriendCount присвоена стрелочная функция с параметром (users).
* Значение параметра users не изменяется.
* Для перебора параметра users использован метод sort().
* Вызов функции с указанным массивом пользователей возвращает новый массив пользователей отсортированный по убыванию количества их друзей.
* Вызов функции со случайными, но валидными аргументами, возвращает правильное значение.

#40

// Пиши код ниже этой строки

const sortByName = users => {

return [...users].sort((a, b) => (a.name[0] >= b.name[0] ? 1 : -1));

};

// Пиши код выше этой строки

**Задача. Сортировка по имени**

Массив объектов пользователей

Этот массив объектов мы будем передавать в параметр users при вызове функции из задания.

[

{

name: 'Moore Hensley',

email: 'moorehensley@indexia.com',

eyeColor: 'blue',

friends: ['Sharron Pace'],

isActive: false,

balance: 2811,

gender: 'male'

},

{

name: 'Sharlene Bush',

email: 'sharlenebush@tubesys.com',

eyeColor: 'blue',

friends: ['Briana Decker', 'Sharron Pace'],

isActive: true,

balance: 3821,

gender: 'female'

},

{

name: 'Ross Vazquez',

email: 'rossvazquez@xinware.com',

eyeColor: 'green',

friends: ['Marilyn Mcintosh', 'Padilla Garrison', 'Naomi Buckner'],

isActive: false,

balance: 3793,

gender: 'male'

},

{

name: 'Elma Head',

email: 'elmahead@omatom.com',

eyeColor: 'green',

friends: ['Goldie Gentry', 'Aisha Tran'],

isActive: true,

balance: 2278,

gender: 'female'

},

{

name: 'Carey Barr',

email: 'careybarr@nurali.com',

eyeColor: 'blue',

friends: ['Jordan Sampson', 'Eddie Strong'],

isActive: true,

balance: 3951,

gender: 'male'

},

{

name: 'Blackburn Dotson',

email: 'blackburndotson@furnigeer.com',

eyeColor: 'brown',

friends: ['Jacklyn Lucas', 'Linda Chapman'],

isActive: false,

balance: 1498,

gender: 'male'

},

{

name: 'Sheree Anthony',

email: 'shereeanthony@kog.com',

eyeColor: 'brown',

friends: ['Goldie Gentry', 'Briana Decker'],

isActive: true,

balance: 2764,

gender: 'female'

}

]

**Задание**

Дополни функцию sortByName(users) так, чтобы она возвращала массив пользователей отсортированный по их имени (свойство name) в алфавитном порядке.

**Тесты**

* Объявлена переменная sortByName.
* Переменной sortByName присвоена стрелочная функция с параметром (users).
* Значение параметра users не изменяется.
* Для перебора параметра users использован метод sort().
* Вызов функции с указанным массивом пользователей возвращает новый массив пользователей отсортированный по имени в алфавитном порядке.
* Вызов функции со случайными, но валидными аргументами, возвращает правильное значение.

#41

const books = [

{ title: 'Последнее королевство', author: 'Бернард Корнуэлл', rating: 8.38 },

{ title: 'На берегу спокойных вод', author: 'Роберт Шекли', rating: 8.51 },

{ title: 'Сон смешного человека', author: 'Федор Достоевский', rating: 7.75 },

{ title: 'Красна как кровь', author: 'Ли Танит', rating: 8.14 },

{ title: 'Сны В Ведьмином Доме', author: 'Говард Лавкрафт', rating: 8.67 }

];

const MIN\_BOOK\_RATING = 8;

// Пиши код ниже этой строки

const names = [...books]

.sort((a, b) => {

const result = a.author[0] > b.author[0];

if (result) {

return 1;

}

if (!result) {

return -1;

}

})

.filter(book => book.rating >= MIN\_BOOK\_RATING)

.map(book => book.author);

**Цепочки методов (чейнинг, chaining)**

Теория

Есть массив объектов с именами, баллами и посещаемыми предметами каждого студента.

const students = [

{ name: 'Манго', score: 83, courses: ['математика', 'физика'] },

{ name: 'Поли', score: 59, courses: ['информатика', 'математика'] },

{ name: 'Аякс', score: 37, courses: ['физика', 'биология'] },

{ name: 'Киви', score: 94, courses: ['литература', 'информатика'] },

];

Необходимо получить массив их имён отсортированный по возрастанию баллов за тест. Для этого мы отсортируем копию массива методом sort(), после чего методом map() составим массив значений свойства name из сортированного массива.

const sortedByAscendingScore = [...students].sort((a, b) => a.score - b.score);

const names = sortedByAscendingScore.map((student) => student.name);

console.log(names); // ['Аякс', 'Поли', 'Манго', 'Киви']

Проблема в том, что у нас появляются промежуточные переменные после каждой операции кроме финальной. Переменная sortedByAscendingScore лишняя и необходима только для хранения промежуточного результата.

Избавиться от таких «мёртвых» переменных можно группируя вызовы методов в цепочки. Каждый следующий метод будет выполняться на результате работы предыдущего.

const names = [...students]

.sort((a, b) => a.score - b.score)

.map((student) => student.name);

console.log(names); // ['Аякс', 'Поли', 'Манго', 'Киви']

1. Делаем копию исходного массива перед сортировкой.
2. На копии вызываем метод sort().
3. К результату работы метода sort() применяем метод map().
4. Переменной names присваивается результат работы метода map().

Получим сортированный по алфавиту массив уникальных посещаемых предметов.

const uniqueSortedCourses = students

.flatMap((student) => student.courses)

.filter((course, index, array) => array.indexOf(course) === index)

.sort((a, b) => a.localeCompare(b));

console.log(uniqueSortedCourses); // ['биология', 'информатика', 'литература', 'математика', 'физика']

1. На исходном массиве вызываем flatMap() и делаем разглаженный массив всех курсов.
2. К результату метода flatMap() применяем метод filter() для фильтрации уникальных элементов.
3. На результате метода filter() вызываем sort().
4. Переменной uniqueSortedCourses присваивается результат работы метода sort().

Цепочка методов может быть произвольной длины, но обычно не более 2-3 операций. Во-первых, перебирающие методы используются для сравнительно простых операций над коллекцией. Во-вторых, вызов каждого последующего метода, это дополнительный перебор массива, что при достаточном количестве, может сказаться на производительности.

**Задание**

Дополни код так, чтобы в переменной names получился массив имён авторов в алфавитном порядке, рейтинг книг которых больше значения переменной MIN\_BOOK\_RATING.

**Тесты**

* Объявлена переменная books.
* Значение переменной books это исходный массив объектов.
* Объявлена переменная MIN\_BOOK\_RATING.
* Значение переменной MIN\_BOOK\_RATING это число 8.
* Объявлена переменная names.
* Значение переменной names это массив ['Бернард Корнуэлл', 'Говард Лавкрафт', 'Ли Танит', 'Роберт Шекли'].
* Нет объявленых переменных кроме books, MIN\_BOOK\_RATING и names.
* Используется цепочка методов filter, map, sort.

#42

// Пиши код ниже этой строки

const getNamesSortedByFriendCount = users => {

return [...users]

.sort((a, b) => (a.friends.length >= b.friends.length ? 1 : -1))

.map(user => user.name);

};

// Пиши код выше этой строки

**Задача. Пользователи и друзья**

Массив объектов пользователей

Этот массив объектов мы будем передавать в параметр users при вызове функции из задания.

[

{

name: 'Moore Hensley',

email: 'moorehensley@indexia.com',

eyeColor: 'blue',

friends: ['Sharron Pace'],

isActive: false,

balance: 2811,

gender: 'male'

},

{

name: 'Sharlene Bush',

email: 'sharlenebush@tubesys.com',

eyeColor: 'blue',

friends: ['Briana Decker', 'Sharron Pace'],

isActive: true,

balance: 3821,

gender: 'female'

},

{

name: 'Ross Vazquez',

email: 'rossvazquez@xinware.com',

eyeColor: 'green',

friends: ['Marilyn Mcintosh', 'Padilla Garrison', 'Naomi Buckner'],

isActive: false,

balance: 3793,

gender: 'male'

},

{

name: 'Elma Head',

email: 'elmahead@omatom.com',

eyeColor: 'green',

friends: ['Goldie Gentry', 'Aisha Tran'],

isActive: true,

balance: 2278,

gender: 'female'

},

{

name: 'Carey Barr',

email: 'careybarr@nurali.com',

eyeColor: 'blue',

friends: ['Jordan Sampson', 'Eddie Strong', 'Adrian Cross'],

isActive: true,

balance: 3951,

gender: 'male'

},

{

name: 'Blackburn Dotson',

email: 'blackburndotson@furnigeer.com',

eyeColor: 'brown',

friends: ['Jacklyn Lucas', 'Linda Chapman', 'Adrian Cross', 'Solomon Fokes'],

isActive: false,

balance: 1498,

gender: 'male'

},

{

name: 'Sheree Anthony',

email: 'shereeanthony@kog.com',

eyeColor: 'brown',

friends: ['Goldie Gentry', 'Briana Decker'],

isActive: true,

balance: 2764,

gender: 'female'

}

]

**Задание**

Дополни функцию getNamesSortedByFriendCount(users) так, чтобы она возвращала массив имён пользователей отсортированный по возрастанию количества их друзей (свойство friends).

**Тесты**

* Объявлена переменная getNamesSortedByFriendCount.
* Переменной getNamesSortedByFriendCount присвоена стрелочная функция с параметром (users).
* В теле функции используется цепочка методов.
* Значение параметра users не изменяется.
* Вызов функции с указанным массивом пользователей возвращает массив ['Moore Hensley', 'Sharlene Bush', 'Elma Head', 'Sheree Anthony', 'Ross Vazquez', 'Carey Barr', 'Blackburn Dotson'].
* Вызов функции со случайными, но валидными аргументами, возвращает правильное значение.

#43

// Пиши код ниже этой строки

const getSortedFriends = users => {

return users

.flatMap(user => user.friends)

.filter((user, index, array) => array.indexOf(user) === index)

.sort((a, b) => a.localeCompare(b));

};

// Пиши код выше этой строки

**Задача. Имена друзей**

Массив объектов пользователей

Этот массив объектов мы будем передавать в параметр users при вызове функции из задания.

[

{

name: 'Moore Hensley',

email: 'moorehensley@indexia.com',

eyeColor: 'blue',

friends: ['Sharron Pace'],

isActive: false,

balance: 2811,

gender: 'male'

},

{

name: 'Sharlene Bush',

email: 'sharlenebush@tubesys.com',

eyeColor: 'blue',

friends: ['Briana Decker', 'Sharron Pace'],

isActive: true,

balance: 3821,

gender: 'female'

},

{

name: 'Ross Vazquez',

email: 'rossvazquez@xinware.com',

eyeColor: 'green',

friends: ['Marilyn Mcintosh', 'Padilla Garrison', 'Naomi Buckner'],

isActive: false,

balance: 3793,

gender: 'male'

},

{

name: 'Elma Head',

email: 'elmahead@omatom.com',

eyeColor: 'green',

friends: ['Goldie Gentry', 'Aisha Tran'],

isActive: true,

balance: 2278,

gender: 'female'

},

{

name: 'Carey Barr',

email: 'careybarr@nurali.com',

eyeColor: 'blue',

friends: ['Jordan Sampson', 'Eddie Strong', 'Adrian Cross'],

isActive: true,

balance: 3951,

gender: 'male'

},

{

name: 'Blackburn Dotson',

email: 'blackburndotson@furnigeer.com',

eyeColor: 'brown',

friends: ['Jacklyn Lucas', 'Linda Chapman', 'Adrian Cross', 'Solomon Fokes'],

isActive: false,

balance: 1498,

gender: 'male'

},

{

name: 'Sheree Anthony',

email: 'shereeanthony@kog.com',

eyeColor: 'brown',

friends: ['Goldie Gentry', 'Briana Decker'],

isActive: true,

balance: 2764,

gender: 'female'

}

]

**Задание**

Дополни функцию getSortedFriends(users) так, чтобы она возвращала массив уникальных имён друзей (свойство friends) отсортированный по алфавиту .

**Тесты**

* Объявлена переменная getSortedFriends.
* Переменной getSortedFriends присвоена стрелочная функция с параметром (users).
* В теле функции используется цепочка методов в правильном порядке.
* Значение параметра users не изменяется.
* Вызов функции с указанным массивом пользователей возвращает массив ['Adrian Cross', 'Aisha Tran', 'Briana Decker', 'Eddie Strong', 'Goldie Gentry', 'Jacklyn Lucas', 'Jordan Sampson', 'Linda Chapman', 'Marilyn Mcintosh', 'Naomi Buckner', 'Padilla Garrison', 'Sharron Pace', 'Solomon Fokes'].
* Вызов функции со случайными, но валидными аргументами, возвращает правильное значение.

#44

// Пиши код ниже этой строки

const getTotalBalanceByGender = (users, gender) => {

return [...users]

.filter(user => user.gender === gender)

.reduce((acc, user) => (acc += user.balance), 0);

};

// Пиши код выше этой строки

**Задача. Общий баланс**

Массив объектов пользователей

Этот массив объектов мы будем передавать в параметр users при вызове функции из задания.

[

{

name: 'Moore Hensley',

email: 'moorehensley@indexia.com',

eyeColor: 'blue',

friends: ['Sharron Pace'],

isActive: false,

balance: 2811,

gender: 'male'

},

{

name: 'Sharlene Bush',

email: 'sharlenebush@tubesys.com',

eyeColor: 'blue',

friends: ['Briana Decker', 'Sharron Pace'],

isActive: true,

balance: 3821,

gender: 'female'

},

{

name: 'Ross Vazquez',

email: 'rossvazquez@xinware.com',

eyeColor: 'green',

friends: ['Marilyn Mcintosh', 'Padilla Garrison', 'Naomi Buckner'],

isActive: false,

balance: 3793,

gender: 'male'

},

{

name: 'Elma Head',

email: 'elmahead@omatom.com',

eyeColor: 'green',

friends: ['Goldie Gentry', 'Aisha Tran'],

isActive: true,

balance: 2278,

gender: 'female'

},

{

name: 'Carey Barr',

email: 'careybarr@nurali.com',

eyeColor: 'blue',

friends: ['Jordan Sampson', 'Eddie Strong', 'Adrian Cross'],

isActive: true,

balance: 3951,

gender: 'male'

},

{

name: 'Blackburn Dotson',

email: 'blackburndotson@furnigeer.com',

eyeColor: 'brown',

friends: ['Jacklyn Lucas', 'Linda Chapman', 'Adrian Cross', 'Solomon Fokes'],

isActive: false,

balance: 1498,

gender: 'male'

},

{

name: 'Sheree Anthony',

email: 'shereeanthony@kog.com',

eyeColor: 'brown',

friends: ['Goldie Gentry', 'Briana Decker'],

isActive: true,

balance: 2764,

gender: 'female'

}

]

**Задание**

Дополни функцию getTotalBalanceByGender(users, gender) так, чтобы она возвращала общий баланс пользователей (свойство balance), пол которых (свойство gender) совпадает со значением параметра gender.

**Тесты**

* Объявлена переменная getTotalBalanceByGender.
* Переменной getTotalBalanceByGender присвоена стрелочная функция с параметрами (users, gender).
* В теле функции используется цепочка методов в правильном порядке.
* Значение параметра users не изменяется.
* Если значение параметра gender это строка 'male', функция возвращает число 12053.
* Если значение параметра gender это строка 'female', функция возвращает число 8863.
* Вызов функции со случайными, но валидными аргументами, возвращает правильное значение.