## 1/32

**Ранний возврат**

В функции может быть больше одного оператора return. Главное помнить, что выполнение функции прерывается когда интерпретатор встречает возврат, и весь код после него будет проигнорирован в текущем вызове функции.

Возьмём уже знакомую нам функцию проверки совершеннолетия. Она работает, но здесь есть «лишний» код, то есть тело функции можно оптимизировать. В данном случае подойдёт приём (паттерн) «ранний возврат».

function checkAge(age) {

let message;

if (age >= 18) {

message = "Вы совершеннолетний человек";

} else {

message = "Вы не совершеннолетний человек";

}

return message;

}

* Если условие в if выполняется, то есть приводится к true, возвращаем строку "You are an adult" и код ниже уже не исполнится.
* Если условие в if не выполняется, то есть приводится к false, возвращаем строку "You are a minor".

Используя паттерн «ранний возврат» и то, что выполнение функции прерывается на операторе return, мы избавляемся от лишней переменной и блока else. То есть этот приём помогает «разгладить» ветвления.

Запиши условие в инструкции if так, чтобы функция работала правильно.

* Объявлена функция checkAge(age).
* В выражении проверки возраста использован оператор >=
* Вызов checkAge(20) возвращает "You are an adult"
* Вызов checkAge(8) возвращает "You are a minor"
* Вызов checkAge(14) возвращает "You are a minor"
* Вызов checkAge(38) возвращает "You are an adult"
* В теле функции есть только одна инструкция if
* В теле функции нет инструкции else или else if
* function checkAge(age) {
* if (age>=20) { // Change this line
* return "You are an adult";
* }
* return "You are a minor";
* }

## 2/32

**Задача: проверка пароля (ранний возврат)**

Функция checkPassword получает пароль пользователя в параметр password, проверяет его на совпадение с паролем администратора в переменной ADMIN\_PASSWORD и возвращает сообщение о результате сравнения.

Проведи рефакторинг кода функции checkPassword используя паттерн «ранний возврат»:

* удали переменную message
* удали else
* код должен работать так же, как и до оптимизации
* Объявлена функция checkPassword(password)
* Вызов checkPassword("mangohackzor") возвращает "Access denied, wrong password!"
* Вызов checkPassword("polyhax") возвращает "Access denied, wrong password!"
* Вызов checkPassword("jqueryismyjam") возвращает "Welcome!"
* function checkPassword(password) {
* const ADMIN\_PASSWORD = "jqueryismyjam";
* // Change code below this line
* if (password === ADMIN\_PASSWORD) {
* return  "Welcome!";
* }
* return "Access denied, wrong password!";
* // Change code above this line
* }

## 3/32

**Задача: склад товаров 3.0**

Функция checkStorage проверяет возможность оформления заказа и возвращает сообщение о результате. Она принимает два параметра, значения которых будут задаваться во время её вызова.

* available - доступное количество товаров на складе
* ordered - количество единиц товара в заказе

Проведи рефакторинг кода функции checkStorage используя паттерн «ранний возврат».

* Объявлена функция checkStorage(available, ordered)
* Вызов checkStorage(100, 50) возвращает "The order is accepted, our manager will contact you"
* Вызов checkStorage(100, 130) возвращает "Your order is too large, not enough goods in stock!"
* Вызов checkStorage(70, 0) возвращает "Your order is empty!"
* Вызов checkStorage(200, 20) возвращает "The order is accepted, our manager will contact you"
* Вызов checkStorage(200, 250) возвращает "Your order is too large, not enough goods in stock!"
* Вызов checkStorage(150, 0) возвращает "Your order is empty!"
* function checkStorage(available, ordered) {
* // Change code below this line
* if (ordered === 0) {
* return "Your order is empty!";
* return
* }
* if (ordered > available) {
* return "Your order is too large, not enough goods in stock!";
* }
* return "The order is accepted, our manager will contact you";
* // Change code above this line
* }

## 4/32

## Создание массива

Массив используется для хранения упорядоченной коллекции элементов. Он объявляется открывающей и закрывающей квадратной скобками [] - литералом массива. Внутри скобок каждый элемент массива разделяется запятой.

const planets = ["Earth", "Mars", "Venus"];

##### Полезно

При объявлении переменной для объекта или массива программисты обычно используют const. Они делают это для того, чтобы случайно не перезаписать значение, т.к. попытка перезаписи вызовет ошибку до того как код попадет к пользователю.

Объяви переменную fruits и присвой ей массив фруктов - строк "apple", "plum", "pear" и "orange".

* Объявлена переменная fruits
* Значение переменной fruits это массив ["apple", "plum", "pear", "orange"]
* // Change code below this line
* const fruits = ["apple", "plum", "pear", "orange"];

## 5/32

## Доступ к элементам по индексу

Для доступа к значению элемента массива применяют синтаксис квадратных скобок массив[индекс]. Между именем переменной массива и квадратными скобками не должно быть пробела.

##### Внимание

Индексация элементов массива начинается с нуля.

const planets = ["Earth", "Mars", "Venus"];

planets[0]; // "Earth"

planets[2]; // "Venus"

Объяви три переменные и присвой каждой из них значение, используя нотацию квадратных скобок.

| **Имя переменной** | **Значение переменной** |
| --- | --- |
| firstElement | первый элемент массива |
| secondElement | второй элемент массива |
| lastElement | последний элемент массива |

* Объявлена переменная firstElement
* Значение переменной firstElement это строка "apple"
* Объявлена переменная secondElement
* Значение переменной secondElement это строка "plum"
* Объявлена переменная lastElement
* Значение переменной lastElement это строка "orange"
* const fruits = ["apple", "plum", "pear", "orange"];
* // Change code below this line
* const firstElement = fruits[0]
* const secondElement = fruits[1]
* const lastElement = fruits[3]

## 6/32

**Переопределение значения элемента**

В отличии от строк, элементы массива можно изменять обратившись к ним по индексу и присвоив другое значение.

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

numbers[0] = 7;

numbers[2] = 14;

console.log(numbers); // [7, 2, 14, 4, 5];

Переопредели значения элементов с индексами 1 и 3. Замени "plum" на "peach", а "orange" на "banana".

* Объявлена переменная fruits
* Значение переменной fruits это массив ["apple", "peach", "pear", "banana"]
* const fruits = ["apple", "plum", "pear", "orange"];
* // Write your code under this line
* fruits[1] = "peach";
* fruits[3] = "banana";

## 7/32

**Длина массива**

Длина массива, то есть число его элементов, хранится в свойстве length. Это динамическая величина, которая изменяется автоматически при добавлении или удалении элементов.

const planets = ["Earth", "Mars", "Venus"];

console.log(planets.length); // 3

Объяви переменную fruitsArrayLength и присвой ей длину массива fruits используя свойство length.

* Объявлена переменная fruitsArrayLength
* Значение переменной fruitsArrayLength это число 4
* const fruits = ["apple", "peach", "pear", "banana"];
* // Change code below this line
* const fruitsArrayLength = fruits.length;

## 8/32

**Индекс последнего элемента**

Объяви две переменные:

| **Имя переменной** | **Ожидаемое значение** |
| --- | --- |
| lastElementIndex | Индекс последнего элемента масcива fruits через длина\_массива - 1 |
| lastElement | Значение последнего элемента массива |

* Объявлена переменная lastElementIndex
* Значение переменной lastElementIndex это число 3
* Объявлена переменная lastElement
* Значение переменной lastElement это строка "banana"
* const fruits = ["apple", "peach", "pear", "banana"];
* // Change code below this line
* const lastElementIndex = fruits.length - 1;
* const lastElement = fruits[lastElementIndex];

## 9/32

**Задача: Крайние элементы массива**

Напиши функцию getExtremeElements(array) которая принимает один параметр array - массив элементов произвольной длины. Функция должна возвращать массив из двух элементов - первого и последнего элемента параметра array.

* Объявлена функция getExtremeElements(array)
* Вызов getExtremeElements([1, 2, 3, 4, 5]) возвращает [1, 5]
* Вызов getExtremeElements(["Earth", "Mars", "Venus"]) возвращает ["Earth", "Venus"]
* Вызов getExtremeElements(["apple", "peach", "pear", "banana"]) возвращает ["apple", "banana"]
* function getExtremeElements(array) {
* // Change code below this line
* const firstItem = array[0];
* const lastItem = array[array.length - 1];
* return [firstItem, lastItem]
* // Change code above this line
* }

## 10/32

**Метод строк split()**

Дополни код функции splitMessage(message, delimeter) так, чтобы она возвращала в переменной words результат разделения строки message по разделителю delimeter - массив строк.

* Объявлена функция splitMessage(message, delimeter)
* Вызов splitMessage("Mango hurries to the train", " ") возвращает ["Mango", "hurries", "to", "the", "train"]
* Вызов splitMessage("Mango", "") возвращает ["M", "a", "n", "g", "o"]
* Вызов splitMessage("best\_for\_week", "\_") возвращает ["best", "for", "week"]
* function splitMessage(message, delimeter) {
* let words;
* // Change code below this line
* words = message.split(delimeter);
* // Change code above this line
* return words;
* }

## 11/32

**Задача: гравировка украшений**

Сервису гравировки украшений нужна функция, которая бы автоматически считала цену гравировки, в зависимости от количества слов и цены за слово.

Объявлена функция calculateEngravingPrice(message, pricePerWord). Эта функция принимает строку, состоящую из слов разделённых только пробелами (параметр message) и цену гравировки одного слова (параметр pricePerWord).

Напиши тело функции, чтобы она возвращала общую стоимость гравировки всех слов в строке.

* Объявлена функция calculateEngravingPrice(message, pricePerWord)
* Вызов calculateEngravingPrice("JavaScript is in my blood", 10) возвращает 50
* Вызов calculateEngravingPrice("JavaScript is in my blood", 20) возвращает 100
* Вызов calculateEngravingPrice("Web-development is creative work", 40) возвращает 160
* Вызов calculateEngravingPrice("Web-development is creative work", 20) возвращает 80
* function calculateEngravingPrice(message, pricePerWord) {
* // Change code below this line
* const result = message.split(' ');
* return result.length \* pricePerWord;
* // Change code above this line
* }

## 12/32

**Метод массива join()**

Дополни код функции makeStringFromArray(array, delimeter) так, чтобы она возвращала в переменной string результат соединения элементов массива array c разделителем delimeter - строку.

* Объявлена функция makeStringFromArray(array, delimeter)
* Вызов makeStringFromArray(["Mango", "hurries", "to", "the", "train"], " ") возвращает "Mango hurries to the train"
* Вызов makeStringFromArray(["M", "a", "n", "g", "o"], "")) возвращает "Mango"
* Вызов makeStringFromArray(["top", "picks", "for", "you"], "\_") возвращает "top\_picks\_for\_you"
* function makeStringFromArray(array, delimeter) {
* let string;
* // Change code below this line
* string = array.join(delimeter);
* // Change code above this line
* return string;
* }

## 13/32

## Задача: генератор slug

Термин **slug** - это человеко-понятный уникальный идентификатор, который используется в веб-разработке для создания читабельных URL-адесов.

Например, вместо того чтобы пользователь увидел в адресной строке mysite.com/posts/1q8fh74tx, можно сделать slug из названия статьи. В результате адрес получится более приятным для восприятия: mysite.com/posts/arrays-for-begginers.

##### Внимание

Slug это всегда строка в нижнем регистре, слова которой разделены тире.

Напиши функцию slugify(title) которая принимает заголовок статьи, параметр title, и возвращает slug, созданный из этой строки.

* Значением параметра title будут строки, слова которых разделены только пробелами
* Все символы slug должны быть в нижнем регистре
* Все слова slug должна быть разделены тире
* Объявлена функция slugify(title)
* Вызов slugify("Arrays for begginers") возвращает "arrays-for-begginers"
* Вызов slugify("English for developer") возвращает "english-for-developer"
* Вызов slugify("Ten secrets of JavaScript") возвращает "ten-secrets-of-javascript"
* Вызов slugify("How to become a JUNIOR developer in TWO WEEKS") возвращает "how-to-become-a-junior-developer-in-two-weeks"
* function slugify(title) {
* // Change code below this line
* return title.toLowerCase().split(' ').join('-');
* // Change code above this line
* }

## 14/32

**Метод slice()**

Дополни код так, чтобы переменные содержали частичные копии исходного массива fruits.

* firstTwoEls - массив из первых двух элементов
* nonExtremeEls - массив из всех элементов кроме первого и последнего
* lastThreeEls - массив из трёх последних элементов
* Объявлена переменная fruits
* Значение переменной fruits это массив ["apple", "plum", "pear", "orange", "banana"]
* Объявлена переменная firstTwoEls
* Значение переменной firstTwoEls это массив ["apple", "plum"]
* Объявлена переменная nonExtremeEls
* Значение переменной nonExtremeEls это массив ["plum", "pear", "orange"]
* Объявлена переменная lastThreeEls
* Значение переменной lastThreeEls это массив ["pear", "orange", "banana"]
* Переменной lastThreeEls присвоена копия части массива fruits после применения метода slice с правильными аргументами
* const fruits = ['apple', 'plum', 'pear', 'orange', 'banana'];
* // Change code below this line
* const firstTwoEls = fruits.slice(0, 2);
* const nonExtremeEls = fruits.slice(1,4);
* const lastThreeEls = fruits.slice(2);

## 15/32

**Метод concat()**

Дополни код так, чтобы в переменной allClients получился массив всех элементов массивов oldClients и newClients.

* Объявлена переменная oldClients
* Значение переменной oldClients это массив ["Mango", "Ajax", "Poly", "Kiwi"]
* Объявлена переменная newClients
* Значение переменной newClients это массив ["Peach", "Houston"]
* Объявлена переменная allClients
* Значение переменной allClients это массив ["Mango", "Ajax", "Poly", "Kiwi", "Peach", "Houston"]
* Переменной allClients присвоен массив после применения метода concat с правильными аргументами
* const oldClients = ['Mango', 'Ajax', 'Poly', 'Kiwi'];
* const newClients = ['Peach', 'Houston'];
* const allClients = oldClients.concat(newClients); // Change this line

## 16/32

**Задача: композиция массивов**

Напиши функцию makeArray(firstArray, secondArray, maxLength) для создания нового массива со всеми элементами двух исходных firstArray и secondArray. Параметр maxLength содержит максимально допустимую длину нового массива.

Если количество элементов нового массива больше maxLength, функция должна вернуть копию массива длиной maxLength элементов. В противном случае функция должна вернуть новый массив целиком.

* Объявлена функция makeArray(firstArray, secondArray, maxLength)
* Вызов makeArray(["Mango", "Poly"], ["Ajax", "Chelsea"], 3) возвращает ["Mango", "Poly", "Ajax"]
* Вызов makeArray(["Mango", "Poly", "Houston"], ["Ajax", "Chelsea"], 4) возвращает ["Mango", "Poly", "Houston", "Ajax"]
* Вызов makeArray(["Mango"], ["Ajax", "Chelsea", "Poly", "Houston"], 3) возвращает ["Mango", "Ajax", "Chelsea"]
* Вызов makeArray(["Earth", "Jupiter"], ["Neptune", "Uranus"], 2) возвращает ["Earth", "Jupiter"]
* Вызов makeArray(["Earth", "Jupiter"], ["Neptune", "Uranus"], 4) возвращает ["Earth", "Jupiter", "Neptune", "Uranus"]
* Вызов makeArray(["Earth", "Jupiter"], ["Neptune", "Uranus", "Venus"], 0) возвращает []
* Вызов функции makeArray() со случайными массивами и случайным числом возвращает правильный массив
* function makeArray(firstArray, secondArray, maxLength) {
* // Change code below this line
* const newArray = firstArray.concat(secondArray);
* if (newArray.length >= maxLength) {
* return newArray.slice(0, maxLength);
* }
* // Change code above this line
* }

## 17/32

**Цикл for**

Дополни цикл for так, чтобы он логировал все целые числа в диапазоне от start до end включительно.

* Объявлена переменная start
* Значение переменной start это число 3
* Объявлена переменная end
* Значение переменной end это число 7
* Объявлена переменная i - счётчик цикла
* Начальное значение переменной i равно 3
* Условие цикла приводится к true до тех пор, пока i меньше либо равно 7
* На каждой итерации значение переменной i увеличивается на единицу
* Вывод в консоль переменной i покажет числа 3, 4, 5, 6, 7
* const start = 3;
* const end = 7;
* for (let i = start; i <= end; i += 1) { // Change this line
* console.log(i);
* }

## 18/32

**Задача: сумма чисел (цикл for)**

Напиши функцию calculateTotal(number), которая принимает целое число (параметр number) и возвращает сумму всех целых чисел от единицы и до этого числа. Например, если number равно 3, то сумма это 1 + 2 + 3, то есть 6.

* Объявлена функция calculateTotal(number)
* Вызов функции calculateTotal(1) возвращает 1
* Вызов функции calculateTotal(3) возвращает 6
* Вызов функции calculateTotal(7) возвращает 28
* Вызов функции calculateTotal(18) возвращает 171
* Вызов функции calculateTotal(24) возвращает 300
* Вызов функции calculateTotal() со случайным числом возвращает правильное значение
* function calculateTotal(number) {
* // Change code below this line
* return number \* (number + 1) / 2;
* // Change code above this line
* }

## 19/32

**Итерация по массиву**

Дополни код цикла for так, чтобы он последовательно логировал все элементы массива fruits.

* Объявлена переменная fruits
* Значение переменной fruits это массив ["apple", "plum", "pear", "orange"]
* Объявлена переменная i - счётчик цикла
* Начальное значение переменной i равно 0
* Условие цикла приводится к true до тех пор, пока i меньше 4
* На каждой итерации значение переменной i увеличивается на единицу
* В теле цикла for объявляется переменная const fruit и этой переменной присваивается значение - элемент массива
* В теле цикла for используется вывод в консоль переменной fruit
* const fruits = ['apple', 'plum', 'pear', 'orange'];
* for (let i = 0; i < fruits.length; i += 1) { // Change this line
* const fruit = fruits[i]; // Change this line
* console.log(fruit);
* }

## 20/32

**Задача: подсчёт суммы покупки**

Напиши функцию calculateTotalPrice(order), которая принимает один параметр order - массив чисел, и рассчитывает общую сумму его элементов. Общая сумма элементов должна сохраняться в переменной total, которая возвращается, как результат работы функции.

* Объявлена функция calculateTotalPrice(order)
* 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 calculateTotalPrice() со случайным массивом возвращает правильное значение
* function calculateTotalPrice(order) {
* let total = 0;
* // Change code below this line
* for (const item of order) {
* total += item;
* }
* // Change code above this line
* return total;
* }

## 21/32

**Задача: поиск самого длинного слова**

Напиши функцию findLongestWord(string) которая принимает произвольную строку состоящую только из слов разделённых пробелом (параметр string) и возвращает самое длинное слово в этой строке.

* Объявлена функция findLongestWord(string)
* Вызов функции findLongestWord("The quick brown fox jumped over the lazy dog") возвращает jumped
* Вызов функции findLongestWord("Google do a roll") возвращает Google
* Вызов функции findLongestWord("May the force be with you") возвращает force
* Вызов функции findLongestWord() со случайной строкой возвращает правильное значение
* function findLongestWord(string) {
* // Change code below this line
* return string.split(' ').reduce((a, b) => (b.length > a.length) ? b : a);
* // Change code above this line
* }

## 22/32

**Метод push()**

Дополни код функции createArrayOfNumbers(min, max) так, чтобы она возвращала массив всех целых чисел от значения min до max.

* Объявлена функция createArrayOfNumbers(min, max)
* Вызов функции createArrayOfNumbers(1, 3) возвращает [1, 2, 3]
* Вызов функции createArrayOfNumbers(14, 17) возвращает [14, 15, 16, 17]
* Вызов функции createArrayOfNumbers(29, 34) возвращает [29, 30, 31, 32, 33, 34]
* Вызов функции createArrayOfNumbers() со случайными min и max возвращает правильный массив
* В цикле for использовался метод push
* function createArrayOfNumbers(min, max) {
* const numbers = [];
* // Change code below this line
* for(let i = min;  i <= max; i++)
* numbers.push(i);
* // Change code above this line
* return numbers;
* }

## 23/32

**Задача: фильтрация массива чисел**

Напиши функцию filterArray(numbers, value), которая принимает массив чисел (параметр numbers) и возвращает новый массив, в котором будут только те элементы массива numbers, которые больше чем значение параметра value (число).

* Объявлена функция filterArray(numbers, value)
* 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 filterArray() со случайным массивом и числом возвращает правильный массив
* В цикле for использовался метод push
* function filterArray(numbers, value) {
* // Change code below this line
* const number = [];
* for (i = 0; i < numbers.length; i+= 1){
* if (numbers[i] > value){
* number.push(numbers[i]);
* }
* }
* return number
* // Change code above this line
* }

## 24/32

**Метод includes()**

Функция checkFruit(fruit) принимает строку с названием фрукта (параметр fruit), и проверяет есть ли такой фрукт в массиве fruits.

Дополни код функции так, что если:

* фрукт есть в массиве, то функция возвращает true;
* фрукта нет в массиве, то функция возвращает false.
* Объявлена функция checkFruit(fruit)
* Вызов checkFruit("plum") возвращает true
* Вызов checkFruit("mandarin") возвращает false
* Вызов checkFruit("pear") возвращает true
* Вызов checkFruit("Pear") возвращает false
* Вызов checkFruit("apple") возвращает true
* Вызов функции checkFruit() со случайным словом возвращает правильное значение boolean
* В функции использовался метод includes
* function checkFruit(fruit) {
* const fruits = ["apple", "plum", "pear", "orange"];
* return fruits.includes(fruit); // Change this line
* }

## 25/32

**Задача: общие элементы**

Общими элементами массивов называют те элементы, которые присутствуют во всех массивах.

Например, в двух массивах [1, 3, 5] и [0, 8, 5, 3] общими будут числа 3 и 5, т.к. они присутствуют в обоих исходных массивах. А числа 0, 1 и 8 присутствуют только в одном из массивов.

Напиши функцию getCommonElements(array1, array2) которая получает два массива произвольной длины в параметры array1 и array2, и возвращает новый массив, состоящий из тех элементов, которые присутствуют в обоих исходных массивах.

* Объявлена функция getCommonElements(array1, array2)
* 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 getCommonElements() со случайными двумя массивами возвращает правильный массив
* В цикле for использовались методы includes и push
* function getCommonElements(array1, array2) {
* // Change code below this line
* let result = [];
* for (i = 0; i < array1.length; i += 1){
* if (array2.includes(array1[i])) {
* result.push(array1[i]);
* }
* }
* return result
* // Change code above this line
* }

## 26/32

**Цикл for...of**

Выполни рефакторинг кода функции calculateTotalPrice(order) заменив цикл for на for...of.

* Объявлена функция calculateTotalPrice(order)
* 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 calculateTotalPrice([]) возвращает 0
* Вызов функции calculateTotalPrice() со случайным массивом чисел возвращает правильную сумму
* function calculateTotalPrice(order) {
* let total = 0;
* // Change code below this line
* for (const item of order) {
* total += item;
* }
* // Change code above this line
* return total;
* }

## 27/32

**Зачада: фильтрация массива чисел 2.0**

Выполни рефакторинг функции filterArray(numbers, value) заменив цикл for на for...of.

* Объявлена функция filterArray(numbers, value)
* 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 filterArray() со случайным массивом и числом возвращает правильный массив
* Функция calculateTotalPrice() использует цикл for..of
* function filterArray(numbers, value) {
* // Change code below this line
* const filteredNumbers = [];
* for ( const number of numbers){
* if (number > value) {
* filteredNumbers.push(number);
* }
* }
* return filteredNumbers;
* // Change code above this line
* }

## 28/32

**Оператор %**

Дополни выражения остатка от деления так, чтобы код проходил тесты.

* Объявлена переменная a
* Значение переменной a это число 0
* Объявлена переменная b
* Значение переменной b это число 1
* Объявлена переменная c
* Значение переменной c это число 3
* Объявлена переменная d
* Значение переменной d это число 5
* Объявлена переменная e
* Значение переменной e это число 2
* // Change code below this line
* const a = 3 % 3;
* const b = 4 % 3;
* const c = 11 % 4;
* const d = 12 % 7;
* const e = 8 % 3;

## 29/32

**Задача: чётные числа**

Напиши функцию getEvenNumbers(start, end) которая возвращает массив всех чётных чисел от start до end. Чётным считается число которое делится на 2 без остатка (10 % 2 === 0).

* Объявлена функция getEvenNumbers(start, end)
* Вызов функции getEvenNumbers(2, 5) возвращает [2, 4]
* Вызов функции getEvenNumbers(3, 11) возвращает [4, 6, 8, 10]
* Вызов функции getEvenNumbers(6, 12) возвращает [6, 8, 10, 12]
* Вызов функции getEvenNumbers(8, 8) возвращает [8]
* Вызов функции getEvenNumbers(7, 7) возвращает []
* Вызов функции getEvenNumbers() со случайными start и end возвращает правильный массив
* function getEvenNumbers(start, end) {
* // Change code below this line
* const evenNumbers = [];
* for (let i = start; i <= end; i += 1) {
* if (i % 2 === 0) {
* evenNumbers.push(i);
* }
* }
* return evenNumbers;
* // Change code above this line
* }

## 30/32

**Оператор break**

Дополни код так, чтобы в переменную number записывалось первое число от start до end, которое делится на 5 без остатка.

* Объявлена переменная start со значением 6
* Объявлена переменная end со значением 27
* Объявлена переменная number без инициализации
* Итоговое значение переменной number равно 10
* В цикле for используется break для выхода до завершения всех итераций цикла
* const start = 6;
* const end = 27;
* let number;
* for (let i = start; i < end; i += 1) {
* if (i % 5 === 0) {
* number = i;
* break
* }
* }

## Оператор break vs return в функции 31/32

Выполни рефакторинг функции findNumber(start, end, divisor) так, чтобы она:

* возвращала первое число от start до end, которое делится на divisor без остатка
* не использовала оператор break
* не использовала переменную number
* Объявлена функция findNumber(start, end, divisor)
* Вызов findNumber(2, 6, 5) возвращает 5
* Вызов findNumber(8, 17, 3) возвращает 9
* Вызов findNumber(6, 9, 4) возвращает 8
* Вызов findNumber(16, 35, 7) возвращает 21
* Вызов findNumber() со случайным набором чисел возвращает верный результат
* В цикле for не должен использоваться break для выхода до завершения всех итераций цикла
* function findNumber(start, end, divisor) {
* // Change code below this line
* let number;
* for (let i = start; i < end; i += 1) {
* if (i % divisor === 0) {
* number = i;
* return number;
* }
* }
* // Change code above this line
* }

**Задача: функция includes() 32/32**

Напиши функцию includes(array, value), которая делает тоже самое, что и метод массива массив.includes(значение) - проверяет, есть ли в массиве array значение value, возвращая true если есть и false в противном случае.

При выполнении этой задачи в теле функции includes() нельзя использовать метод массив.includes(значение).

* Объявлена функция includes(array, value)
* Вызов includes([1, 2, 3, 4, 5], 3) возвращает true
* Вызов includes([1, 2, 3, 4, 5], 17) возвращает false
* Вызов includes(["Earth", "Mars", "Venus", "Jupiter", "Saturn"], "Jupiter") возвращает true
* Вызов includes(["Earth", "Mars", "Venus", "Jupiter", "Saturn"], "Uranus") возвращает false
* Вызов includes(["apple", "plum", "pear", "orange"], "plum") возвращает true
* Вызов includes(["apple", "plum", "pear", "orange"], "kiwi") возвращает false
* Вызов includes() для случайного массива со случайным value возвращает верный boolean
* В функции includes используется for, return, но не метод массива includes
* function includes(array, value) {
* // Change code below this line
* for (const arrays of array) {
* if (arrays === value) {
* arrays === value;
* return true;
* }
* }
* return false;
* // Change code above this line
* }