**Числа**

**Задача 1**Создайте страницу, которая предлагает ввести два числа и выводит их сумму.

**Задача 2**

alert( 0.1 + 0.2 + '$' );

Что можно сделать, чтобы избежать проблем с ошибками округления?

**Задача 3**  
Напишите функцию getDecimal(num), которая возвращает десятичную часть числа:

alert( getDecimal(12.345) ); // 0.345

alert( getDecimal(1.2) ); // 0.2

alert( getDecimal(-1.2) ); // 0.2

**Задача 4**

Напишите код для генерации случайного значения в диапазоне от 0 до max, не включая max.

**Задача 5**

Напишите код для генерации случайного числа от min до max, не включая max.

**Задача 6**

Напишите функцию randomInteger(min, max) для генерации случайного **целого** числа между min и max, включая min,max как возможные значения.

**Строки**

**Задача 1**Напишите функцию ucFirst(str), которая возвращает строку str с заглавным первым символом, например: (Создайте функцию, используя toUpperCase() и charAt().)

**Задача 2**Напишите функцию checkSpam(str), которая возвращает true, если строка str содержит „viagra“ или „XXX“, а иначе false.

**Задача 3**Создайте функцию truncate(str, maxlength), которая проверяет длину строки str, и если она превосходит maxlength – заменяет конец str на "...", так чтобы ее длина стала равна maxlength. Результатом функции должна быть (при необходимости) усечённая строка.

**Задача 4**

Есть стоимость в виде строки: "$120". То есть, первым идёт знак валюты, а затем – число.

Создайте функцию extractCurrencyValue(str), которая будет из такой строки выделять число-значение, в данном случае 120.

**Объекты**

**Задача 1**Мини-задача на синтаксис объектов. Напишите код, по строке на каждое действие.

1. Создайте пустой объект user.
2. Добавьте свойство name со значением Вася.
3. Добавьте свойство surname со значением Петров.
4. Поменяйте значение name на Сергей.
5. Удалите свойство name из объекта.

**Задача 2**

Создайте функцию isEmpty(obj), которая возвращает true, если в объекте нет свойств и false – если хоть одно свойство есть.

Работать должно так:

function isEmpty(obj) {

/\* ваш код \*/

}

var schedule = {};

alert( isEmpty(schedule) ); // true

schedule["8:30"] = "подъём";

alert( isEmpty(schedule) ); // false

**Задача 3**

Есть объект salaries с зарплатами. Напишите код, который выведет сумму всех зарплат.

Если объект пустой, то результат должен быть 0.

Например:

"use strict";

var salaries = {

"Вася": 100,

"Петя": 300,

"Даша": 250

};

//... ваш код выведет 650

P.S. Сверху стоит use strict, чтобы не забыть объявить переменные.

**Задача 4**

Создайте функцию multiplyNumeric, которая получает объект и умножает все численные свойства на 2. Например:

// до вызова

var menu = {

width: 200,

height: 300,

title: "My menu"

};

multiplyNumeric(menu);

// после вызова

menu = {

width: 400,

height: 600,

title: "My menu"

};

P.S. Для проверки на число используйте функцию:

function isNumeric(n) {

return !isNaN(parseFloat(n)) && isFinite(n)

}

**Задача 5**

Написать фунцию клонирования объекта

Var newObj = cloneObj(Obj);

**Массивы**

**Задача 1**

Как получить последний элемент из произвольного массива?

У нас есть массив goods. Сколько в нем элементов – не знаем, но можем прочитать из goods.length.

Напишите код для получения последнего элемента goods.

**Задача 2**

Как добавить элемент в конец произвольного массива?

У нас есть массив goods. Напишите код для добавления в его конец значения «Компьютер».

**Задача 3**

Задача из 5 шагов-строк:

1. Создайте массив styles с элементами «Джаз», «Блюз».
2. Добавьте в конец значение «Рок-н-Ролл»
3. Замените предпоследнее значение с конца на «Классика». Код замены предпоследнего значения должен работать для массивов любой длины.
4. Удалите первое значение массива и выведите его alert.
5. Добавьте в начало значения «Рэп» и «Регги».

Массив в результате каждого шага:

Джаз, Блюз

Джаз, Блюз, Рок-н-Ролл

Джаз, Классика, Рок-н-Ролл

Классика, Рок-н-Ролл

Рэп, Регги, Классика, Рок-н-Ролл

**Задача 4**

Напишите код для вывода alert случайного значения из массива:

var arr = ["Яблоко", "Апельсин", "Груша", "Лимон"];

P.S. Код для генерации случайного целого от min to max включительно:

var rand = min + Math.floor(Math.random() \* (max + 1 - min));

Задача

Напишите код, который:

* Запрашивает по очереди значения при помощи prompt и сохраняет их в массиве.
* Заканчивает ввод, как только посетитель введёт пустую строку, не число или нажмёт «Отмена».
* При этом ноль 0 не должен заканчивать ввод, это разрешённое число.
* Выводит сумму всех значений массива

**Задача 5**

Что выведет этот код?

var arr = [1, 2, 3];

var arr2 = arr;

arr2[0] = 5;

alert( arr[0] );

alert( arr2[0] );

**Задача 6**

Создайте функцию find(arr, value), которая ищет в массиве arr значение value и возвращает его номер, если найдено, или -1, если не найдено.

Например:

arr = ["test", 2, 1.5, false];

find(arr, "test"); // 0

find(arr, 2); // 1

find(arr, 1.5); // 2

find(arr, 0); // -1

**Задача 7**

Создайте функцию filterRange(arr, a, b), которая принимает массив чисел arr и возвращает новый массив, который содержит только числа из arr из диапазона от a до b. То есть, проверка имеет вид a ≤ arr[i] ≤ b. Функция не должна менять arr.

Пример работы:

var arr = [5, 4, 3, 8, 0];

var filtered = filterRange(arr, 3, 5);

// теперь filtered = [5, 4, 3]

// arr не изменился

**Задача 8**

Целое число, большее 1, называется *простым*, если оно не делится нацело ни на какое другое, кроме себя и 1.

Древний алгоритм «Решето Эратосфена» для поиска всех простых чисел до n выглядит так:

1. Создать список последовательных чисел от 2 до n: 2, 3, 4, ..., n.
2. Пусть p=2, это первое простое число.
3. Зачеркнуть все последующие числа в списке с разницей в p, т.е. 2\*p, 3\*p, 4\*p и т.д. В случае p=2 это будут 4,6,8....
4. Поменять значение p на первое не зачеркнутое число после p.
5. Повторить шаги 3-4 пока p2 < n.
6. Все оставшиеся не зачеркнутыми числа – простые.

Посмотрите также [анимацию алгоритма](https://learn.javascript.ru/task/eratosthenes-sieve/sieve.gif).

Реализуйте «Решето Эратосфена» в JavaScript, используя массив.

Найдите все простые числа до 100 и выведите их сумму.

**Задача 9**

На входе массив чисел, например: arr = [1, -2, 3, 4, -9, 6].

Задача – найти непрерывный подмассив arr, сумма элементов которого максимальна.

Ваша функция должна возвращать только эту сумму.

Например:

getMaxSubSum([-1, 2, 3, -9]) = 5 (сумма выделенных)

getMaxSubSum([2, -1, 2, 3, -9]) = 6

getMaxSubSum([-1, 2, 3, -9, 11]) = 11

getMaxSubSum([-2, -1, 1, 2]) = 3

getMaxSubSum([100, -9, 2, -3, 5]) = 100

getMaxSubSum([1, 2, 3]) = 6 (неотрицательные - берем всех)

Если все элементы отрицательные, то не берём ни одного элемента и считаем сумму равной нулю:

getMaxSubSum([-1, -2, -3]) = 0

Постарайтесь придумать решение, которое работает за O(n2), а лучше за O(n) операций.

**Массивы Методы**

Методы:

* push/pop, shift/unshift, splice – для добавления и удаления элементов.
* join/split – для преобразования строки в массив и обратно.
* slice – копирует участок массива.
* sort – для сортировки массива. Если не передать функцию сравнения – сортирует элементы как строки.
* reverse – меняет порядок элементов на обратный.
* concat – объединяет массивы.
* indexOf/lastIndexOf – возвращают позицию элемента в массиве (не поддерживается в IE8-).

**Задача 1**

В объекте есть свойство className, которое содержит список «классов» – слов, разделенных пробелом:

var obj = {

className: 'open menu'

}

Создайте функцию addClass(obj, cls), которая добавляет в список класс cls, но только если его там еще нет:

addClass(obj, 'new'); // obj.className='open menu new'

addClass(obj, 'open'); // без изменений (класс уже существует)

addClass(obj, 'me'); // obj.className='open menu new me'

alert( obj.className ); // "open menu new me"

P.S. Ваша функция не должна добавлять лишних пробелов.

**Задача 2**

Напишите функцию camelize(str), которая преобразует строки вида «my-short-string» в «myShortString».

То есть, дефисы удаляются, а все слова после них получают заглавную букву.

Например:

camelize("background-color") == 'backgroundColor';

camelize("list-style-image") == 'listStyleImage';

camelize("-webkit-transition") == 'WebkitTransition';

Такая функция полезна при работе с CSS.

P.S. Вам пригодятся методы строк charAt, split и toUpperCase.

**Задача 3**

У объекта есть свойство className, которое хранит список «классов» – слов, разделенных пробелами:

var obj = {

className: 'open menu'

};

Напишите функцию removeClass(obj, cls), которая удаляет класс cls, если он есть:

removeClass(obj, 'open'); // obj.className='menu'

removeClass(obj, 'blabla'); // без изменений (нет такого класса)

P.S. Дополнительное усложнение. Функция должна корректно обрабатывать дублирование класса в строке:

obj = {

className: 'my menu menu'

};

removeClass(obj, 'menu');

alert( obj.className ); // 'my'

Лишних пробелов после функции образовываться не должно.

**Задача 4**

Создайте функцию filterRangeInPlace(arr, a, b), которая получает массив с числами arr и удаляет из него все числа вне диапазона a..b. То есть, проверка имеет вид a ≤ arr[i] ≤ b. Функция должна менять сам массив и ничего не возвращать.

Например:

arr = [5, 3, 8, 1];

filterRangeInPlace(arr, 1, 4); // удалены числа вне диапазона 1..4

alert( arr ); // массив изменился: остались [3, 1]

**Задача 5**

Как отсортировать массив чисел в обратном порядке?

var arr = [5, 2, 1, -10, 8];

// отсортируйте?

alert( arr ); // 8, 5, 2, 1, -10

**Задача 6**

Есть массив строк arr. Создайте массив arrSorted – из тех же элементов, но отсортированный.

Исходный массив не должен меняться.

var arr = ["HTML", "JavaScript", "CSS"];

// ... ваш код ...

alert( arrSorted ); // CSS, HTML, JavaScript

alert( arr ); // HTML, JavaScript, CSS (без изменений)

Постарайтесь сделать код как можно короче.

**Задача 7**

Используйте функцию sort для того, чтобы «перетрясти» элементы массива в случайном порядке.

var arr = [1, 2, 3, 4, 5];

arr.sort(ваша функция);

alert( arr ); // элементы в случайном порядке, например [3,5,1,2,4]

**Задача 8**

Напишите код, который отсортирует массив объектов people по полю age.

Например:

var vasya = { name: "Вася", age: 23 };

var masha = { name: "Маша", age: 18 };

var vovochka = { name: "Вовочка", age: 6 };

var people = [ vasya , masha , vovochka ];

... ваш код ...

// теперь people: [vovochka, masha, vasya]

alert(people[0].age) // 6

Выведите список имён в массиве после сортировки.

**Задача 9**

[Односвязный список](http://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D1%8F%D0%B7%D0%BD%D1%8B%D0%B9_%D1%81%D0%BF%D0%B8%D1%81%D0%BE%D0%BA) – это структура данных, которая состоит из *элементов*, каждый из которых хранит ссылку на следующий. Последний элемент может не иметь ссылки, либо она равна null.

Например, объект ниже задаёт односвязный список, в next хранится ссылка на следующий элемент:

var list = {

value: 1,

next: {

value: 2,

next: {

value: 3,

next: {

value: 4,

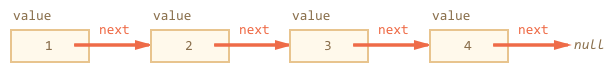
next: null

}

}

}

};

Графическое представление этого списка: 

Альтернативный способ создания:

var list = { value: 1 };

list.next = { value: 2 };

list.next.next = { value: 3 };

list.next.next.next = { value: 4 };

Такая структура данных интересна тем, что можно очень быстро разбить список на части, объединить списки, удалить или добавить элемент в любое место, включая начало. При использовании массива такие действия требуют обширных перенумерований.

Задачи:

1. Напишите функцию printList(list), которая выводит элементы списка по очереди, при помощи цикла.
2. Напишите функцию printList(list) при помощи рекурсии.
3. Напишите функцию printReverseList(list), которая выводит элементы списка в обратном порядке, при помощи рекурсии. Для списка выше она должна выводить 4,3,2,1
4. Сделайте вариант printReverseList(list), использующий не рекурсию, а цикл.

Как лучше – с рекурсией или без?

**Задача 10**

Анаграммы – слова, состоящие из одинакового количества одинаковых букв, но в разном порядке. Например:

воз - зов

киборг - гробик

корсет - костер - сектор

Напишите функцию aclean(arr), которая возвращает массив слов, очищенный от анаграмм.

Например:

var arr = ["воз", "киборг", "корсет", "ЗОВ", "гробик", "костер", "сектор"];

alert( aclean(arr) ); // "воз,киборг,корсет" или "ЗОВ,гробик,сектор"

Из каждой группы анаграмм должно остаться только одно слово, не важно какое.

**Задача 11**

Пусть arr – массив строк.

Напишите функцию unique(arr), которая возвращает массив, содержащий только уникальные элементы arr.

Например:

function unique(arr) {

/\* ваш код \*/

}

var strings = ["кришна", "кришна", "харе", "харе",

"харе", "харе", "кришна", "кришна", "8-()"

];

alert( unique(strings) ); // кришна, харе, 8-()

**Массивы Методы перребора**

Метод [«arr.forEach(callback[, thisArg])»](https://developer.mozilla.org/en/JavaScript/Reference/Global_Objects/Array/forEach) используется для перебора массива.

Метод [«arr.filter(callback[, thisArg])»](https://developer.mozilla.org/en/JavaScript/Reference/Global_Objects/Array/filter) используется для фильтрации массива через функцию.

Метод [«arr.map(callback[, thisArg])»](https://developer.mozilla.org/en/JavaScript/Reference/Global_Objects/Array/map) используется для трансформации массива.

Эти методы используются для проверки массива.

* Метод [«arr.every(callback[, thisArg])»](https://developer.mozilla.org/en/JavaScript/Reference/Global_Objects/Array/every) возвращает true, если вызов callback вернёт true для *каждого* элемента arr.
* Метод [«arr.some(callback[, thisArg])»](https://developer.mozilla.org/en/JavaScript/Reference/Global_Objects/Array/some) возвращает true, если вызов callback вернёт true для *какого-нибудь* элемента arr.

Метод [«arr.reduce(callback[, initialValue])»](https://developer.mozilla.org/en/JavaScript/Reference/Global_Objects/Array/reduce) используется для последовательной обработки каждого элемента массива с сохранением промежуточного результата.

Это один из самых сложных методов для работы с массивами. Но его стоит освоить, потому что временами с его помощью можно в несколько строк решить задачу, которая иначе потребовала бы в разы больше места и времени.

Метод reduce используется для вычисления на основе массива какого-либо единого значения, иначе говорят «для свёртки массива». Чуть далее мы разберём пример для вычисления суммы.

Он применяет функцию callback по очереди к каждому элементу массива слева направо, сохраняя при этом промежуточный результат.

Аргументы функции callback(previousValue, currentItem, index, arr):

* previousValue – последний результат вызова функции, он же «промежуточный результат».
* currentItem – текущий элемент массива, элементы перебираются по очереди слева-направо.
* index – номер текущего элемента.
* arr – обрабатываемый массив.

Кроме callback, методу можно передать «начальное значение» – аргумент initialValue. Если он есть, то на первом вызове значение previousValue будет равно initialValue, а если у reduce нет второго аргумента, то оно равно первому элементу массива, а перебор начинается со второго.

**Задача 1**

Код ниже получает из массива строк новый массив, содержащий их длины:

var arr = ["Есть", "жизнь", "на", "Марсе"];

var arrLength = [];

for (var i = 0; i < arr.length; i++) {

arrLength[i] = arr[i].length;

}

alert( arrLength ); // 4,5,2,5

Перепишите выделенный участок: уберите цикл, используйте вместо него метод map.

**Задача 2**

На входе массив чисел, например: arr = [1,2,3,4,5].

Напишите функцию getSums(arr), которая возвращает массив его частичных сумм.

Иначе говоря, вызов getSums(arr) должен возвращать новый массив из такого же числа элементов, в котором на каждой позиции должна быть сумма элементов arr до этой позиции включительно.

То есть:

для arr = [ 1, 2, 3, 4, 5 ]

getSums( arr ) = [ 1, 1+2, 1+2+3, 1+2+3+4, 1+2+3+4+5 ] = [ 1, 3, 6, 10, 15 ]

Еще пример: getSums([-2,-1,0,1]) = [-2,-3,-3,-2].

* Функция не должна модифицировать входной массив.
* В решении используйте метод arr.reduce.

**Дата И Время**

**Задача 1**

Создайте объект Date для даты: 20 февраля 2012 года, 3 часа 12 минут.

Временная зона – местная. Выведите его на экран.

**Задача 2**

Создайте функцию getWeekDay(date), которая выводит текущий день недели в коротком формате „пн“, „вт“, … „вс“.

Например:

var date = new Date(2012,0,3); // 3 января 2012

alert( getWeekDay(date) ); // Должно вывести 'вт'

**Задача 3**

Напишите функцию, getLocalDay(date) которая возвращает день недели для даты date.

День нужно возвратить в европейской нумерации, т.е. понедельник имеет номер 1, вторник номер 2, …, воскресенье – номер 7.

var date = new Date(2012, 0, 3); // 3 янв 2012

alert( getLocalDay(date) ); // вторник, выведет 2

**Задача 4**

Создайте функцию getDateAgo(date, days), которая возвращает число, которое было days дней назад от даты date.

Например, для 2 января 2015:

var date = new Date(2015, 0, 2);

alert( getDateAgo(date, 1) ); // 1, (1 января 2015)

alert( getDateAgo(date, 2) ); // 31, (31 декабря 2014)

alert( getDateAgo(date, 365) ); // 2, (2 января 2014)

P.S. Важная деталь: в процессе вычислений функция не должна менять переданный ей объект date.

**Задача 5**

Напишите функцию getLastDayOfMonth(year, month), которая возвращает последний день месяца.

Параметры:

* year – 4-значный год, например 2012.
* month – месяц от 0 до 11.

Например, getLastDayOfMonth(2012, 1) = 29 (високосный год, февраль).

**Задача 6**

Напишите функцию getSecondsToday() которая возвращает, сколько секунд прошло с начала сегодняшнего дня.

Например, если сейчас 10:00 и не было перехода на зимнее/летнее время, то:

getSecondsToday() == 36000 // (3600 \* 10)

Функция должна работать в любой день, т.е. в ней не должно быть конкретного значения сегодняшней даты.

**Задача 7**

Напишите функцию getSecondsToTomorrow() которая возвращает, сколько секунд осталось до завтра.

Например, если сейчас 23:00, то:

getSecondsToTomorrow() == 3600

P.S. Функция должна работать в любой день, т.е. в ней не должно быть конкретного значения сегодняшней даты.

**Задача 8**

Напишите функцию formatDate(date), которая выводит дату date в формате дд.мм.гг:

Например:

var d = new Date(2014, 0, 30); // 30 января 2014

alert( formatDate(d) ); // '30.01.14'

P.S. Обратите внимание, ведущие нули должны присутствовать, то есть 1 января 2001 должно быть 01.01.01, а не 1.1.1.

**Задача 9**

Напишите функцию formatDate(date), которая форматирует дату date так:

* Если со времени date прошло менее секунды, то возвращает "только что".
* Иначе если со времени date прошло менее минуты, то "n сек. назад".
* Иначе если прошло меньше часа, то "m мин. назад".
* Иначе полная дата в формате "дд.мм.гг чч:мм".

Например:

function formatDate(date) { /\* ваш код \*/ }

alert( formatDate(new Date(new Date - 1)) ); // "только что"

alert( formatDate(new Date(new Date - 30 \* 1000)) ); // "30 сек. назад"

alert( formatDate(new Date(new Date - 5 \* 60 \* 1000)) ); // "5 мин. назад"

**Замыкания**

**Задача 1**

Что будет, если вызов say('Вася'); стоит в самом-самом начале, в первой строке кода?

say('Вася'); // Что выведет? Не будет ли ошибки?

var phrase = 'Привет';

function say(name) {

alert( name + ", " + phrase );

}

**Задача 2**

Каков будет результат выполнения этого кода?

var value = 0;

function f() {

if (1) {

value = true;

} else {

var value = false;

}

alert( value );

}

f();

Изменится ли внешняя переменная value ?

P.S. Какими будут ответы, если из строки var value = false убрать var?

**Задача 3**

Каков будет результат выполнения этого кода? Почему?

function test() {

alert( window );

var window = 5;

alert( window );

}

test();

**Задача 4**

Каков будет результат выполнения кода? Почему?

var a = 5

(function() {

alert(a)

})()

P.S. Подумайте хорошо! Здесь все ошибаются! P.P.S. Внимание, здесь подводный камень! Ок, вы предупреждены.

**Задача 5**

Если во внутренней функции есть своя переменная с именем currentCount – можно ли в ней получить currentCountиз внешней функции?

function makeCounter() {

var currentCount = 1;

return function() {

var currentCount;

// можно ли здесь вывести currentCount из внешней функции (равный 1)?

};

}

**Задача 6**

Что выведут эти вызовы, если переменная currentCount находится вне makeCounter?

var currentCount = 1;

function makeCounter() {

return function() {

return currentCount++;

};

}

var counter = makeCounter();

var counter2 = makeCounter();

alert( counter() ); // ?

alert( counter() ); // ?

alert( counter2() ); // ?

alert( counter2() ); // ?

**Задача 7**

Напишите функцию sum, которая работает так: sum(a)(b) = a+b.

Да, именно так, через двойные скобки (это не опечатка). Например:

sum(1)(2) = 3

sum(5)(-1) = 4

**Задача 8**

В некоторых языках программирования существует объект «строковый буфер», который аккумулирует внутри себя значения. Его функционал состоит из двух возможностей:

1. Добавить значение в буфер.
2. Получить текущее содержимое.

**Задача – реализовать строковый буфер на функциях в JavaScript, со следующим синтаксисом:**

* Создание объекта: var buffer = makeBuffer();.
* Вызов makeBuffer должен возвращать такую функцию buffer, которая при вызове buffer(value) добавляет значение в некоторое внутреннее хранилище, а при вызове без аргументов buffer() – возвращает его.

Вот пример работы:

function makeBuffer() { /\* ваш код \*/ }

var buffer = makeBuffer();

// добавить значения к буферу

buffer('Замыкания');

buffer(' Использовать');

buffer(' Нужно!');

// получить текущее значение

alert( buffer() ); // Замыкания Использовать Нужно!

Буфер должен преобразовывать все данные к строковому типу:

var buffer = makeBuffer();

buffer(0);

buffer(1);

buffer(0);

alert( buffer() ); // '010'

Решение не должно использовать глобальные переменные.

**Задача 9**

Добавьте буферу из решения задачи [Функция - строковый буфер](https://learn.javascript.ru/task/stringbuffer) метод buffer.clear(), который будет очищать текущее содержимое буфера:

function makeBuffer() {

...ваш код...

}

var buffer = makeBuffer();

buffer("Тест");

buffer(" тебя не съест ");

alert( buffer() ); // Тест тебя не съест

buffer.clear();

alert( buffer() ); // ""

**Задача 10**

У нас есть массив объектов:

var users = [{

name: "Вася",

surname: 'Иванов',

age: 20

}, {

name: "Петя",

surname: 'Чапаев',

age: 25

}, {

name: "Маша",

surname: 'Медведева',

age: 18

}];

Обычно сортировка по нужному полю происходит так:

// по полю name (Вася, Маша, Петя)

users.sort(function(a, b) {

return a.name > b.name ? 1 : -1;

});

// по полю age (Маша, Вася, Петя)

users.sort(function(a, b) {

return a.age > b.age ? 1 : -1;

});

Мы хотели бы упростить синтаксис до одной строки, вот так:

users.sort(byField('name'));

users.forEach(function(user) {

alert( user.name );

}); // Вася, Маша, Петя

users.sort(byField('age'));

users.forEach(function(user) {

alert( user.name );

}); // Маша, Вася, Петя

То есть, вместо того, чтобы каждый раз писать в sort function... – будем использовать byField(...)

Напишите функцию byField(field), которую можно использовать в sort для сравнения объектов по полю field, чтобы пример выше заработал.

**Задача 11**

1. Создайте функцию filter(arr, func), которая получает массив arr и возвращает новый, в который входят только те элементы arr, для которых func возвращает true.
2. Создайте набор «готовых фильтров»: inBetween(a,b) – «между a,b», inArray([...]) – "в массиве [...]". Использование должно быть таким:

* filter(arr, inBetween(3,6)) – выберет только числа от 3 до 6,
* filter(arr, inArray([1,2,3])) – выберет только элементы, совпадающие с одним из значений массива.

Пример, как это должно работать:

/\* .. ваш код для filter, inBetween, inArray \*/

var arr = [1, 2, 3, 4, 5, 6, 7];

alert(filter(arr, function(a) {

return a % 2 == 0

})); // 2,4,6

alert( filter(arr, inBetween(3, 6)) ); // 3,4,5,6

alert( filter(arr, inArray([1, 2, 10])) ); // 1,2

**Задача 12**

Следующий код создает массив функций-стрелков shooters. По замыслу, каждый стрелок должен выводить свой номер:

function makeArmy() {

var shooters = [];

for (var i = 0; i < 10; i++) {

var shooter = function() { // функция-стрелок

alert( i ); // выводит свой номер

};

shooters.push(shooter);

}

return shooters;

}

var army = makeArmy();

army[0](); // стрелок выводит 10, а должен 0

army[5](); // стрелок выводит 10...

// .. все стрелки выводят 10 вместо 0,1,2...9

Почему все стрелки́ выводят одно и то же? Поправьте код, чтобы стрелки работали как задумано. Предложите несколько вариантов исправления.

**Объеты this**

**Задача 1**

Каким будет результат? Почему?

var arr = ["a", "b"];

arr.push(function() {

alert( this );

})

arr[2](); // ?

**Задача 2**

Каков будет результат этого кода?

var obj = {

go: function() { alert(this) }

}

(obj.go)()

P.S. Есть подвох :)

**Задача 3**

Вызовы (1) и (2) в примере ниже работают не так, как (3) и (4):

"use strict"

var obj, method;

obj = {

go: function() { alert(this); }

};

obj.go(); // (1) object

(obj.go)(); // (2) object

(method = obj.go)(); // (3) undefined

(obj.go || obj.stop)(); // (4) undefined

В чём дело? Объясните логику работы this.

**Задача 4**

Что выведет alert в этом коде? Почему?

var user = {

firstName: "Василий",

export: this

};

alert( user.export.firstName );

Задача

Что выведет alert в этом коде? Почему?

var name = "";

var user = {

name: "Василий",

export: function() {

return this;

}

};

alert( user.export().name );

Задача

Что выведет alert в этом коде? Почему?

var name = "";

var user = {

name: "Василий",

export: function() {

return {

value: this

};

}

};

alert( user.export().value.name );

**Задача 5**

Создайте объект calculator с тремя методами:

* read() запрашивает prompt два значения и сохраняет их как свойства объекта
* sum() возвращает сумму этих двух значений
* mul() возвращает произведение этих двух значений

var calculator = {

...ваш код...

}

calculator.read();

alert( calculator.sum() );

alert( calculator.mul() );

**Задача 6**

Есть объект «лестница» ladder:

var ladder = {

step: 0,

up: function() { // вверх по лестнице

this.step++;

},

down: function() { // вниз по лестнице

this.step--;

},

showStep: function() { // вывести текущую ступеньку

alert( this.step );

}

};

Сейчас, если нужно последовательно вызвать несколько методов объекта, это можно сделать так:

ladder.up();

ladder.up();

ladder.down();

ladder.showStep(); // 1

Модифицируйте код методов объекта, чтобы вызовы можно было делать цепочкой, вот так:

ladder.up().up().down().up().down().showStep(); // 1

Как видно, такая запись содержит «меньше букв» и может быть более наглядной.

**Функции-обёртки, декораторы**

JavaScript предоставляет удивительно гибкие возможности по работе с функциями: их можно передавать, в них можно записывать данные как в объекты, у них есть свои встроенные методы…

Конечно, этим нужно уметь пользоваться. В этой главе, чтобы более глубоко понимать работу с функциями, мы рассмотрим создание функций-обёрток или, иначе говоря, «декораторов».

[Декоратор](http://en.wikipedia.org/wiki/Decorator_pattern) – приём программирования, который позволяет взять существующую функцию и изменить/расширить ее поведение.

*Декоратор* получает функцию и возвращает обертку, которая делает что-то своё «вокруг» вызова основной функции.

## [bind – привязка контекста](https://learn.javascript.ru/decorators" \l "bind-привязка-контекста)

Один простой декоратор вы уже видели ранее – это функция [bind](https://learn.javascript.ru/bind):

function bind(func, context) {

return function() {

return func.apply(context, arguments);

};

}

Вызов bind(func, context) возвращает обёртку, которая ставит this и передаёт основную работу функции func.

**Задача 1**

Создайте декоратор makeLogging(func, log), для функции func возвращающий обёртку, которая при каждом вызове добавляет её аргументы в массив log.

Условие аналогично задаче [Логирующий декоратор (1 аргумент)](https://learn.javascript.ru/task/logging-decorator), но допускается func с любым набором аргументов.

Работать должно так:

function work(a, b) {

alert( a + b ); // work - произвольная функция

}

function makeLogging(f, log) { /\* ваш код \*/ }

var log = [];

work = makeLogging(work, log);

work(1, 2); // 3

work(4, 5); // 9

for (var i = 0; i < log.length; i++) {

var args = log[i]; // массив из аргументов i-го вызова

alert( 'Лог:' + args.join() ); // "Лог:1,2", "Лог:4,5"

}

**Задача 2**

Создайте декоратор makeCaching(f), который берет функцию f и возвращает обертку, которая кеширует её результаты.

**В этой задаче функция f имеет только один аргумент, и он является числом.**

1. При первом вызове обертки с определенным аргументом – она вызывает f и запоминает значение.
2. При втором и последующих вызовах с тем же аргументом возвращается запомненное значение.

Должно работать так:

function f(x) {

return Math.random() \* x; // random для удобства тестирования

}

function makeCaching(f) { /\* ваш код \*/ }

f = makeCaching(f);

var a, b;

a = f(1);

b = f(1);

alert( a == b ); // true (значение закешировано)

b = f(2);

alert( a == b ); // false, другой аргумент => другое значение