Практическое занятие №13 Массивы, Math, Date

Задание №1

Проверьте запущен ли у Вас Open Server. Папка с материалами урока (js13.loc) должна быть размещена в директории /domains сервера. Заходим на http://js13.loc и начинаем урок.

Занятие №2

На предыдущих занятиях рассматривались основы массивов, однако после того как Вы узнали больше про объекты следует подробней рассмотреть и массивы.

Все массивы в JS являются объектами прототипа Array.prototype. Array — это конструктор для массивов, которые на высоком уровне представляют собой спископодобные объекты.

Создать массив можно несколькими способами:

```
var a = new Array(1, 5, 'text');
var a = Array(1, 5, 'text');

var a = [1, 5, 'text']; // предпочтительный способ
```

Если не передать никаких значений в функцию-конструктор или ничего не писать внутри квадратных скобок, будет создан пустой массив.

Последний способ более предпочтительным поскольку создание массивов через конструктор имеет некоторую особенность про которую необходимо постоянно помнить:

```
var a = new Array(5,6,7);
var b = new Array(5);
console.log(a); //[5, 6, 7]
console.log(b); //[undefined x 5]
```

Если при создании объектов через конструктор передать один параметр, то будет создан массив с указанным количеством пустых элементов.

Задание №3

Разреженные массивы. Разреженным называется массив, индексы элементов которого не образуют непрерывную последовательность чисел, начиная с 0. Обычно свойство length массива определяет количество элементов в массиве. В разреженном массиве значение свойства length больше количества элементов.

```
var a = [];

a[0] = 1;
a[2] = 2;

console.log('a[0] = ' + a[0]);
console.log('a[1] = ' + a[1]);
console.log('a[2] = ' + a[2]);
console.log('a.length = ' + a.length);
```

Задание №4

Для обхода разреженного массива лучше не использовать цикл for вида:

```
var a = [];

a[0] = 1;
a[2] = 2;


for(var index = 0; index < a.length; index++){
    console.log('a['+index+'] = ' + a[index]);
}
```

А использовать цикл for…in:

```
var a = [];

a[0] = 1;
a[2] = 2;

for(var index in a){

    console.log('a['+index+'] = ' + a[index]);

}
```

Задание №5

Еще одна особенность в поведении свойства length, заключается в том, что при присваивании свойству length неотрицательного целого числа n, меньшего, чем его текущее значение, все элементы массива с индексами, большими или равными значению n, удаляются из массива:

```
var a = [1,2,3];

console.log(a);

a.length = 2;

console.log(a);
```

Аналогично если n будет больше length в массив будут добавлены пустые элементы.

Задание №6

Удалять элементы массива можно с помощью оператора delete, как обычные свойства объектов

```
var a = [1,2,3];

console.log(a);

delete a[1];

console.log(a);
```

Удаление элемента напоминает (но несколько отличается) присваивание значения undefined этому элементу. Обратите внимание, что применение оператора delete к элементу массива не изменяет значение свойства length и не сдвигает вниз элементы с более высокими индексами, чтобы заполнить пустоту, оставшуюся после удаления элемента. После удаления элемента массив превращается в разреженный массив.

Задание №7

Добавление и удаление элементов через методы. Для добавления и удаления элементов в массив есть 4 метода:

1. Array.shift() – удаляет элемент из начала массива
2. Array.unshift(n) – добавляет элемент в начало массива
3. Array.pop() – удаляет элемент из конца массива
4. Array.push(n) – добавляет элемент в конец массива

```
var a = [1, 2, 3];
var b = [4, 5, 6];
var c = [7, 8, 9];
var d = [10, 11, 12];

a.shift();
console.log(a);

b.unshift(3);
console.log(b);

c.push(10);
console.log(c);

d.pop();
console.log(d);
```

Как видно из примера выше, в отличие от оператора delete, метод shift() сдвигает все элементы вниз на позицию ниже их текущих индексов, а метод pop() удаляет последний элемент и оба метода уменьшают параметр length массива.

Также методы shift и pop при удалении элемента возвращают его.

```
var a = [1, 2, 3];

console.log(a.shift());
console.log(a.pop());
```

Задание №8

Метод Array.join(str) преобразует все элементы массива в строки, объединяет их и возвращает получившуюся строку. В необязательном аргументе методу можно передать строку, которая будет использоваться для отделения элементов в строке результата.

```
var a = [1, 2, 3];

console.log(a.join());
console.log(a.join(' ... '));
```

Задание №9

Метод Array.reverse() меняет порядок следования элементов в массиве на обратный. Меняет исходный массив.

```
var a = [1, 2, 3];

a.reverse();
console.log(a);
```

Задание №10

Метод Array.concat(…) Возвращает новый массив, в который добавляются переданные значения в качестве новых элементов. Если любое из значений – массив, то добавляется каждый его элемент. Исходный массив не меняется.

```
var a = [1, 2, 3];
var b = [];
b = a.concat(1, 3, 5, [7,8,9]);
console.log(a);
console.log(b);
```

Задание №11

Метод Array.sort(func) сортирует элементы в исходном массиве. Если метод sort() вызывается без аргументов, сортировка выполняется в алфавитном порядке.

```
var a = [1, 2, 3, 10, 20];
a.sort();
console.log(a); //[1, 10, 2, 20, 3]
```

Для сортировки в каком-либо ином порядке, отличном от алфавитного, методу sort() можно передать функцию сравнения в качестве аргумента. Эта функция устанавливает, какой из двух ее аргументов должен следовать раньше в отсортированном списке. Если первый аргумент должен предшествовать второму, функция сравнения должна возвращать отрицательное число. Если первый аргумент должен следовать за вторым в отсортированном массиве, то функция должна возвращать число больше нуля. А если два значения эквивалентны (т.е. порядок их следования не важен), функция сравнения должна возвращать 0.

```
var a = [33, 4, 1111, 222];

a.sort(function (a, b) {
    return a > b ? 1 : -1;
});

console.log(a);
```

Задание №12

Дан массив

```
var a = [596, 500, 958, 121, 480, 174, 269, 840, 143, 498, 51, 87, 721, 13, 468, 207, 102, 811, 934, 467,
807, 712, 387, 503, 339, 445, 858, 902, 519, 382, 106, 270, 655, 611, 725, 977, 696, 846, 823, 278, 451,
791, 586, 453, 241, 722, 972, 913, 395, 14, 769, 98, 860, 157, 580, 3, 448, 718, 108, 567, 496, 42, 273,
800, 334, 404, 963, 75, 289, 105, 664, 716, 119, 939, 794, 551, 274, 396, 811, 703, 449, 539, 860, 166,
268, 606, 260, 24, 430, 550, 154, 622, 162, 290, 1, 914, 649, 398, 496, 53];
```

Отсортировать массив в обратном порядке.

Задание №13

Дан массив сотрудников

```
var staff = [{age: 25, experience: 3}, {age: 36, experience: 7}, {age: 28, experience: 2}, {age: 33,
experience: 6}, {age: 30, experience: 4}, {age: 32, experience: 7}, {age: 36, experience: 6}, {age: 26,
experience: 1}, {age: 29, experience: 4}, {age: 33, experience: 1}, {age: 34, experience: 1}, {age: 28,
experience: 5}, {age: 28, experience: 5}, {age: 26, experience: 2}, {age: 28, experience: 6}, {age: 28,
experience: 1}, {age: 37, experience: 4}, {age: 31, experience: 1}, {age: 28, experience: 3}, {age: 26,
experience: 1}, {age: 34, experience: 7}, {age: 30, experience: 1}, {age: 32, experience: 5}, {age: 29,
```

```
experience: 7}, {age: 31, experience: 7}, {age: 36, experience: 2}, {age: 33, experience: 3}, {age: 37,
experience: 6}, {age: 33, experience: 5}, {age: 40, experience: 4}, {age: 27, experience: 2}];
```

Отсортировать массив в сотрудников так чтобы сначала были самые молодые сотрудники (поле age). Если возраст сотрудников одинаков, то сначала должны быть сотрудники с наибольшим опытом (поле experience).

Задание №14

Метод Array.slice(start, end) возвращает часть массива, которая начинается с элемента со стартовым индексом и заканчивается элементом с конечным индексом. Элемент с конечным индексом не включается. Если второй аргумент отсутствует slice() извлекает все элементы до конца последовательности. Отрицательные индексы означают отсчет с конца массива. Если стартовая позиция больше конечной, вернется пустой массив.

```
var a = [0,1,2,3,4,5];

console.log(a.slice(0, 2)); //первые 2 элемента
console.log(a.slice(2)); //все элементы, кроме первых двух
console.log(a.slice(0)); //копия массива
console.log(a.slice(4,6)); //последние 2 элемента

console.log(a.slice(-a.length, -a.length + 2)); //первые 2 элемента
console.log(a.slice(-a.length)); //копия массива
```

Задание №15

Метод Array.splice(start, length, …). Вырезает и возвращает указанное количество элементов, начиная со стартовой позиции. Если количество элементов не передается, вырезается все элементы до конца массива, начиная со стартовой позиции. Если передаются значения, то они вставляются вместо вырезанных элементов, начиная со стартовой позиции. Если в качестве значений для вставки передается массив, то вставляется не массив, а его элементы каждый по отдельности. Исходный массив меняется

```
var myFish = ['ангел', 'клоун', 'мандарин', 'хирург'];

// удаляет 0 элементов с индекса 2 и вставляет элемент 'барабанщик'
var removed = myFish.splice(2, 0, 'барабанщик');
console.log(removed);
console.log(myFish);

// удаляет 1 элемент с индекса 3
removed = myFish.splice(3, 1);
console.log(removed);
console.log(myFish);

// удаляет 1 элемент с индекса 2 и вставляет элемент 'телескоп'
removed = myFish.splice(2, 1, 'телескоп');
console.log(removed);
console.log(myFish);

// удаляет 2 элемента с индекса 0 и вставляет элементы
// 'попугай', 'анемон' и 'голубая'
removed = myFish.splice(0, 2, 'попугай', 'анемон', 'голубая');
console.log(removed);
console.log(myFish);

// удаляет 2 элемента с индекса 3
removed = myFish.splice(3, Number.MAX_VALUE);
console.log(removed);
console.log(myFish);
```

Задание №16

С использованием метода Array.splice() замените методы Array.push(), Array.pop(), Array.shift(), Array.unshift().

Задание №17

Свойство Array.prototype представляет прототип для конструктора Array. Как и с остальными конструкторами, вы можете изменять прототип конструктора объекта для применения изменений ко всем экземплярам класса Array.

Этот способ не самый рекомендованный для добавления нового функционала массивам, поскольку работает с глобальным объектом и возможны конфликты имен, однако в некоторых ситуациях позволительный.

```
var a = [0, 1, 2, 3, 4, 5];

//возвращает элементы с нечетными ключами
Array.prototype.oddKey = function () {

    var newArray = [];

    for (var index in this) {
        if (index % 2 === 1) {
            newArray.push(this[index]);
        }
    }

    return newArray;

};

console.log(a.oddKey());
```

Задание №18

Расширьте Array.prototype добавив метод который возвращает все целые положительные значения массива, увеличив их в 2 раза.

Задание №19

Расширьте Array.prototype добавив метод который возвращает среднее арифметическое массива, если все элементы числа или возвращает NaN если в массиве есть не числовое значение.

Задание №20

Расширьте Array.prototype добавив метод который изменяет исходный массив уменьшив все числа в 10 раз.

Задание №21

Расширьте Array.prototype добавив метод который возвращает массив где все строки преобразованы в верхний регистр.

Задание №22

Иногда бывает удобно организовать работу с произвольным объектом, как со своего рода массивом – через свойство length и соответствующие неотрицательные целочисленные свойства. Такие объекты, «подобные массивам», иногда используются для решения практических задач, и, хотя с ними нельзя работать через методы массивов или ожидать специфического поведения свойства length, все же можно организовать перебор свойств объекта теми же программными конструкциями, которые используются при работе с настоящими массивами. Оказывается, что значительное число алгоритмов для работы с массивами вполне пригодно для работы с объектами, подобными массивам. Это особенно верно, если используемые алгоритмы не изменяют массивы или хотя бы не затрагивают его свойство length.

```
var aLike = {
 '0' : 1,
 '1' : 2,
 '2' : 3,
 '3' : 4,
 '4' : 5,
 'length': 5
};

var sliceResult = Array.prototype.slice.call(aLike, 0, 2);
console.log(sliceResult);

var joinResult = Array.prototype.join.call(aLike, ', ');
console.log(joinResult);
```

Задание №23

Проверка на массив. Проверить является ли объект массивом можно используя метод конструктора Array – Array.isArray().

```
var a = [1,2,3];
var b = {'0': 0, '1': 1};
console.log(Array.isArray(a));
console.log(Array.isArray(b));
```

Обратите внимание что только объекты типа массив распознаются этим методом, но не объекты похожие на массив.

Задание №24

Объекты как ассоциативные массивы. Как отмечалось выше, следующие два выражения возвращают одно и то же значение:

```
var object = {prorerty: 1};
object.property;
object["property"];
```

Первая форма записи, с использованием точки и идентификатора, напоминает синтаксис доступа к статическому полю структуры или объекта в языке C или Java. Вторая форма записи, с использованием квадратных скобок и строки, выглядит как обращение к элементу массива, но массива, который индексируется строками, а не числами. Такого рода массивы называются ассоциативными массивами (а также хешами и словарями). Объекты в языке JavaScript можно использовать ассоциативными массивами.

Создайте ассоциативный массив (объект) с вашим именем (firstName), фамилией (lastName) и номером группы (group), если вы не студент, оставьте группу пустой строкой. Обойдите массив с использованием цикла for…in и выведите информацию в консоль.

Ассоциативные массивы достаточно мощный инструмент, особенно при передаче данных.

Задание №25

JS не поддерживает «настоящие» многомерные массивы, но позволяет неплохо имитировать их при помощи массивов из массивов. Для доступа к элементу данных в массиве массивов достаточно дважды использовать оператор []. Например, предположим, что переменная matrix – это массив массивов чисел. Каждый элемент matrix[x] – это массив чисел. Для доступа к определенному числу в массиве можно использовать выражение matrix[x][y].

Пример, где двумерный массив используется в качестве таблицы умножения.

```
// Создать многомерный массив
var table = new Array(10); // В таблице 10 строк
for (var i = 0; i < table.length; i++)
    table[i] = new Array(10); // В каждой строке 10 столбцов

// Инициализировать массив
for (var row = 0; row < table.length; row++) {
    for (col = 0; col < table[row].length; col++) {
        table[row][col] = row * col;
    }
}


var product = table[5][7]; // 35
```

Задание №26

Написать функцию, которая обходит массив произвольного уровня вложенности

например

```
var a = [
    [1, 2, 3],
    4,
    [[5, 6, 7],[8, 9, 10]]
];
```

И находит сумму всех цифр в этом массиве (необходимо использовать рекурсию).

Задание №27

Объект Math. Для работы с математическими функциями есть специальный объект Math, у которого есть свойства и методы. Этот объект – свойство глобального объекта.

Свойства объекта Math (константы)	
Название	Что означает
Math.E	основание натуральных логарифмов
Math.LN10	Натуральный логарифм 10
Math.LN2	Натуральный логарифм 2
Math.LOG10E	Десятичный логарифм e
Math.LOG2E	Логарифм e по основанию 2

Math.PI	Константа π
Math.SQRT2	квадратный из 2
Math.SQRT1_2	Единица, деленная на корень Квадратный корень из 2

Методы объекта Math		
Название	Аргументы	Что делает и возвращает
Math.abs(n)	n – любое число	Возвращает абсолютное значение числа
Math.acos(n)	n – число от -1.0 до 1.0	Возвращает арккосинус числа в радианах Результат в интервале от 0 до π
Math.asin(n)	n – число от -1.0 до 1.0	Возвращает арксинус числа в радианах Результат в интервале от -π/2 до π/2
Math.atan(n)	n – любое число	Возвращает арктангенс числа в радианах Результат в интервале от -π/2 до π/2
Math.atan2(x, y)	x – координата по оси Y y – координата по оси X	Возвращает угол между направлением на точку и положительной осью X в радианах (против часовой стрелки) Результат в интервале от -π до π
Math.ceil(n)	n – любое число	Возвращает ближайшее целое, большее или равное переданному числу
Math.cos(n)	n – количество радиан в угле	Возвращает косинус переданного угла
Math.exp(n)	n – любое число	Возвращает e в степени переданного числа
Math.floor(n)	n – любое число	Возвращает ближайшее целое, меньшее или равное переданному числу
Math.log(n)	n – любое положительное число	Возвращает натуральный логарифм числа
Math.max(n[, …])	n, … – перечень любого количества чисел	Возвращает максимальное значение из переданных чисел
Math.min(n[, …])	n, … – перечень любого количества чисел	Возвращает минимальное значение из переданных чисел
Math.pow(x, y)	x – любое число y – значение степени	Возвращает число, возведенное в степень

Math.random()		Возвращает псевдослучайное число в диапазоне между 0 и 1.0
Math.round(n)	n – любое число	Возвращает целое, ближайшее к переданному числу
Math.sin(n)	n – количество радиан в угле	Возвращает синус переданного угла Результат в интервале от -1.0 до 1.0
Math.sqrt(n)	n – любое положительное число	Возвращает квадратный корень из переданного числа
Math.tan(n)	n – количество радиан в угле	Возвращает тангенс переданного угла

Задание №28

Напишите функции:

1. Функцию, которая возвращает кубический корень входящего числа.
2. Функцию, которая находит синус угла. Функция должна принимать значение угла в градусах.
3. Функцию, которая принимает 2 числа и возвращает случайное число, которое находится между первым и вторым аргументом.
4. Функцию, которая находит среднее геометрическое 2 чисел (корень их произведения).
5. Функцию, которая будет находить расстояние между двумя точками на плоскости AB = $\sqrt{((x_b - x_a)^2 + (y_b - y_a)^2)}$

Задание №29

Тип данных Date. Этот тип данных предназначен для работы с датами и временем.

Экземпляры Date всегда создаются с помощью вызова функции-конструктора – у этого типа данных нет литерала. В зависимости от типа и количества аргументов создаются даты с различным значением.

Если не передавать никаких аргументов, создается объект Date с текущим временем и датой.

```
var d = new Date();
console.log(d);
```

Единственное число в качестве аргумента представляется в виде миллисекунд и создается объект Date по количеству миллисекунд, прошедших с начала 1970 года.

```
var d = new Date(946677600000);
console.log(d);
```

Перечень чисел в качестве аргументов:

1. год – число из 4-х цифр (для числа от 0 до 99 автоматически добавляется 1900)
2. номер месяца (нумерация с 0 до 11)
3. дата (от 1 до 31)

4. часы (от 0 до 23)
5. минуты (от 0 до 59)
6. секунды (от 0 до 59)
7. миллисекунды (от 0 до 999)

Год и месяц – обязательные числа (если будет указано одно число – сработает предыдущий способ для миллисекунд).

```
var d = new Date(2009, 6, 7, 0, 25, 32, 245);
console.log(d);
```

Очень важным моментом является то, что при передаче некорректных цифр в функция-конструктор, он исправляет их автоматически.

```
var d = new Date(2009, 12); //будет создана дата 1 января 2010 года
console.log(d);
```

Единственную строку в качестве аргумента функция-конструктор пытается представить в формате, который может быть понят методом Date.parse(). Количество миллисекунд при использовании этого способа указать нельзя.

```
var d = new Date('Tue Jul 07 2009 00:25:32 GMT+0400');
console.log(d);
```

Формат строки допускает различные разделители – пробел, запятую, "/", "–" (с некоторыми ограничениями).

Порядок следования может быть любым – необходимо лишь, чтобы эту строку понял и смог разобрать специальный метод parse() функции-конструктора Date (описан ниже). Обязательным из всего этого списка является только месяц, число и год.

При создании дат с помощью описанных вызовов функции-конструктора мы используем так называемое локальное или местное время. Оно зависит от настроек операционной системы и для такого времени указывается смещение относительно Гринвичского меридиана, определяемое часовым поясом.

GMT (Greenwich Mean Time) – система времени, основанная на астрономических наблюдениях и привязанная к Земле. Точка отсчета – Гринвичский меридиан, что и закреплено в названии. Эта система неудобна тем, что в различных точках Земли смещение может быть продиктовано не только геофизическим расположением, но и переходом на летнее/зимнее время. Кроме того, неравномерность времени в GMT (из-за смещения географических полюсов) делает ее некорректной при решении навигационных задач.

Более удобной в этом смысле является система универсального времени UTC (Coordinated Universal Time). Она была введена в 1964 г. И привязана к равномерной шкале атомного времени. Точка отсчета для удобства осталась на Гринвичском меридиане, но время в этой системе одинаково в любой момент времени в любой точке Земли. При появлении смещения UTC относительно GMT+0 более, чем на 0.9 с происходит коррекция (30 июня или 31 декабря). Первое такое смещение было произведено 30 июня 1972 г. Именно в этой системе передается время по телевидению, радио, интернет и в навигационных системах.

методы функции-конструктора Date		
Название	Аргументы	Что делает и возвращает

Date.UTC (year, month[, date, hours, minutes, seconds, ms])	year – год month – номер месяца date – дата hours – часы minutes – минуты seconds – секунды ms – миллисекунды	Метод сходен с работой вызова класса- конструктора Date() и передачей набора чисел Основное различие – работа идет с универсальным временем, а не с локальным Возвращает количество миллисекунд с 1 января 1970 года
Date.parse(s)	s – строка, которая должна быть в формате даты	Метод неявно вызывается, когда в класс-конструктор Date() передается строка Возвращает количество миллисекунд с 1 января 1970 года

Методы экземпляров Date		
Название	Аргументы	Что делает и возвращает
getFullYear() getUTCFullYear()		Возвращает год (из 4-х цифр)
getMonth() getUTCMonth()		Возвращает номер месяца (цифра от 0 до 11)
getDate() getUTCDate()		Возвращает дату (цифра от 1 до 31)
getHours() getUTCHours()		Возвращает количество часов (цифра от 0 до 23
getMinutes() getUTCMinutes()		Возвращает количество минут (цифра от 0 до 59
getSeconds() getUTCSeconds()		Возвращает количество секунд (цифра от 0 до 59)
getMilliseconds() getUTCMilliseconds()		Возвращает количество миллисекунд (цифра от 0 до 999)
getDay() getUTCDay()		Возвращает номер дня недели (цифра от 0 до 6) Нумерация начинается с воскресенья
getTime() valueOf()		Возвращает количество миллисекунд с 1 января 1970 года Это количество не зависит от часового пояса
getTimezoneOffset()		Возвращает разницу с гринвичским временем в минутах Зависит от переходов на летнее/зимнее время

toString() toUTCString() toGMTString()		Методы выводят данные об экземпляре в строковом формате (дата и время)
toLocaleString()		Делает то же, что и toString(), но с учетом настроек операционной системы (язык, форматы и т.д.)
toTimeString()		Выводит только время экземпляра в строковом формате
toLocaleTimeString()		Делает то же, что и toTimeString(), но с учетом настроек операционной системы (язык, форматы и т.д.)
toDateString()		Выводит только дату экземпляра в строковом формате
toLocaleDateString()		Делает то же, что и toDateString(), но с учетом настроек операционной системы (язык, форматы и т.д.)
setFullYear(year) setUTCFullYear(year)	year – год	Устанавливает новый год (число из 4-х цифр)
setMonth(month) setUTCMonth(month)	month – номер месяца	Устанавливает новый номер месяца (цифра от 0 до 11)
setDate(date) setUTCDate(date)	date – дата	Устанавливает новую дату (цифра от 1 до 31)
setHours(hours) setUTCHours(hours)	hours – часы	Устанавливает новое количество часов (цифра от 0 до 23)
setMinutes(minutes) setUTCMinutes(minutes)	minutes – минуты	Устанавливает новое количество минут (цифра от 0 до 59)
setSeconds(seconds) setUTCSeconds(seconds)	seconds – секунды	Устанавливает новое количество секунд (цифра от 0 до 59)
setMilliseconds(ms) setUTCMilliseconds(ms)	ms – миллисекунды	Устанавливает новое количество миллисекунд (цифра от 0 до 999)
setTime(ms)	ms – миллисекунды	Устанавливает новое время и дату (полностью меняя экземпляр), принимая количество миллисекунд с 1 января 1970 года Это количество не зависит от часового пояса

Задание №30

Напишите функции:

1. Напишите функцию getLastDayOfMonth(year, month), которая возвращает последний день месяца. Например, getLastDayOfMonth(2012, 1) = 29 (високосный год, февраль).
2. Напишите функцию getSecondsToday() которая возвращает, сколько секунд прошло с начала сегодняшнего дня.
3. Напишите функцию getSecondsToTomorrow() которая возвращает, сколько секунд осталось до завтра.
4. Напишите функцию formatDate(date), которая выводит дату date в формате дд.мм.гг. Обратите внимание, ведущие нули должны присутствовать, то есть 1 января 2001 должно быть 01.01.01, а не 1.1.1.
5. Создайте функцию getWeekDay(date), которая выводит текущий день недели в коротком формате „пн“, „вт“, … „вс“.