### Урок 8. Методы объектов и контекст вызова

* [Методы объектов, this](http://learn.javascript.ru/object-methods)
* [Преобразование объектов: toString и valueOf](http://learn.javascript.ru/object-conversion)
* [Создание объектов через «new»](http://learn.javascript.ru/constructor-new)
* [Дескрипторы, геттеры и сеттеры свойств](http://learn.javascript.ru/descriptors-getters-setters)
* [Статические и фабричные методы](http://learn.javascript.ru/static-properties-and-methods)

**Домашнее задание. JS. Урок 8.**

**Задание 1 (Chaining)**

Есть объект 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, для последовательного вызова нескольких методов объекта, нужно делать так сделать так:

ladder.up();

ladder.up();

ladder.down();

ladder.showStep(); // 1

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

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

Такой подход называется «чейнинг» (chaining)

**Задание 2 (Calculator)**

Напишите конструктор Calculator, который создаёт расширяемые объекты-калькуляторы.

Эта задача состоит из двух частей, которые можно решать одна за другой.

**Первый шаг** задачи: вызов calculate(str) принимает строку, например «1 + 2», с жёстко заданным форматом «ЧИСЛО операция ЧИСЛО» (по одному пробелу вокруг операции), и возвращает результат. Понимает плюс + и минус -. Пример использования:

var calc = new Calculator;

console.log( calc.calculate('3 + 7') ); // 10

**Второй шаг** — добавить калькулятору метод addMethod(name, func), который учит калькулятор новой операции. Он получает имя операции name и функцию от двух аргументов func(a,b), которая должна её реализовывать. Например, добавим операции умножить *, поделить / и возвести в степень \**:

var powerCalc = new Calculator;

powerCalc.addMethod('\*', function(a, b) {

return a \* b;

});

powerCalc.addMethod('/', function(a, b) {

return a / b;

});

powerCalc.addMethod('\*\*', function(a, b) {

return Math.pow(a, b);

});

var result = powerCalc.calculate('2 \*\* 3');

console.log( result ); // 8

* Поддержка скобок и сложных математических выражений в этой задаче не требуется.
* Числа и операции могут состоять из нескольких символов. Между ними ровно один пробел.
* Предусмотрите обработку ошибок. Какая она должна быть – решите сами.

**Задание 3 (get/set свойства)**

У вас есть объект User, который хранит имя и фамилию в свойстве this.fullName:

function User(fullName) {

this.fullName = fullName;

}

var vasya = new User('Александр Пушкин');

Имя и фамилия всегда разделяются пробелом.

Сделайте, чтобы были доступны свойства firstName и lastName, причём не только на чтение, но и на запись, вот так:

var vasya = new User('Александр Пушкин');

// чтение firstName/lastName

console.log( vasya.firstName ); // Александр

console.log( vasya.lastName ); // Пушкин

// запись в lastName

vasya.lastName = 'Толстой';

console.log( vasya.fullName ); // Александр Толстой

Важно: в этой задаче fullName должно остаться свойством, а firstName/lastName — реализованы через get/set. Лишнее дублирование здесь ни к чему.

**Задание 4 (Object counter)**

Добавить в конструктор Article:

* Подсчёт общего количества созданных объектов.
* Запоминание даты последнего созданного объекта.
* Используйте для этого статические свойства.

Пусть вызов Article.showStats() выводит то и другое.

Использование:

function Article() {

this.created = new Date();

// ... ваш код ...

}

new Article();

new Article();

Article.showStats(); // Всего: 2, Последняя: (дата)

new Article();

Article.showStats(); // Всего: 3, Последняя: (дата)