**Node.js**

**Установка сервера**

Выкачиваем инсталляху с основного сайта, ставим. Вместе с нодой обновляется и npm. Таким же образом сервер обновляется - просто ставим поверх новый.

Проверяем текущую версию ноды и npm через cmd:

node -v

npm -v

**Модули Node**

**CommonJS** -это стандартная библиотека для всех модулей Javacript

Node.js написан на CommonJS

Стандарты CommonJS:

* Каждый javascript файл - свой собственный модуль
* Переменные модуля предоставляют доступ к текущему расположению модуля в файле
* module.exports - переменная, которая определяет экспорт из текущего модуля:

module.exports = function () {

return {

perimeter: function(x,y) { return (2 \* (x + y)); },

area: function(x,y) { return (x \* y); }

};

}

Например, сохранили его под названием require.js. Использование:

var rect = require('./rectangle') // если лежит в той же директории

**require** - это функция, которая используется для импорта модуля

Альтернативный способ экспорта файла. Например, сохраним такой код в файл rect1.js

exports.perimeter = function(x,y) {

return (2 \* (x + y));

},

exports.area = function(x,y) {

return (x \* y);

}

// exports - алиас для module.exports

Теперь создаем второй файл, назовем его rectangle.js, там пишем код, который будет использовать эту функцию, и включаем саму функцию:

var rect = require('./rect1.js');

function solveRect(l, b) {

console.log("Area is " + rect.area(l, b));

console.log("Perimeter is " + rect.perimeter(l, b));

}

}

solveRect(5, 12);

Откываем терминал, переходим в нужную папку, где лежит файл, пишем

node rectangle // можно без расширения .js

**Callbacks and error handling**

В Node есть так называемые Event Loop, который выполняет задачи асинхронно. Пришел запрос - попал в Event Loop, получил промис, что будет вызван callback, когда запрос выполнится, пока запрос выполняется - может прийти еще один запрос, и тоже получит промис.

Перепишем rectangle в стиле Node:

moudle.exports = function (x, y, callback) {

try {

if (x < 0 || y < 0) {

throw new Error("Rectangle params < 0: l=" + x + ", and b=" + y);

}

else

callback(null, {

perimeter: function (x, y) { return (2 \* (x + y)); },

area: function (x, y) { return (x \* y); }

});

}

catch (error) {

callback(error, null);

}

};

Модуль вызывает callback в любом случае. Если попалась ошибка, первым параметром в коллбек передается ошибка, а вторым, где должен быть return с полезной частью функции - null. Если же ошибки нет, вызывается callback с первым аргументом null - вместо ошибки.

Использование модуля:

rect(l, b, function(err, rectangle){

if(err) {

console.log(err);

} else {

//...code....

}

});