

Урок 8

Пишем back-end на NodeJS, EpxressJS и MongoDB

Подключаем NodeJS, EpxressJS и MongoDB

[Что такое NodeJS](#)

[Что такое ExpressJS](#)

[Что такое MongoDB](#)

[Устанавливаем и настраиваем NodeJS](#)

[После установки](#)

[Подключаем ExpressJS](#)

[Организуем обработку входящих запросов при помощи ExpressJS Router](#)

[Устанавливаем и настраиваем MongoDB](#)

[Типизируем наши схемы, подключаем Mongoose](#)

[Практика](#)

[Задача 1. Сравнение кода, выполненного в Node.js и Chrome](#)

[Домашнее задание](#)

[Дополнительные материалы](#)

[Используемая литература](#)

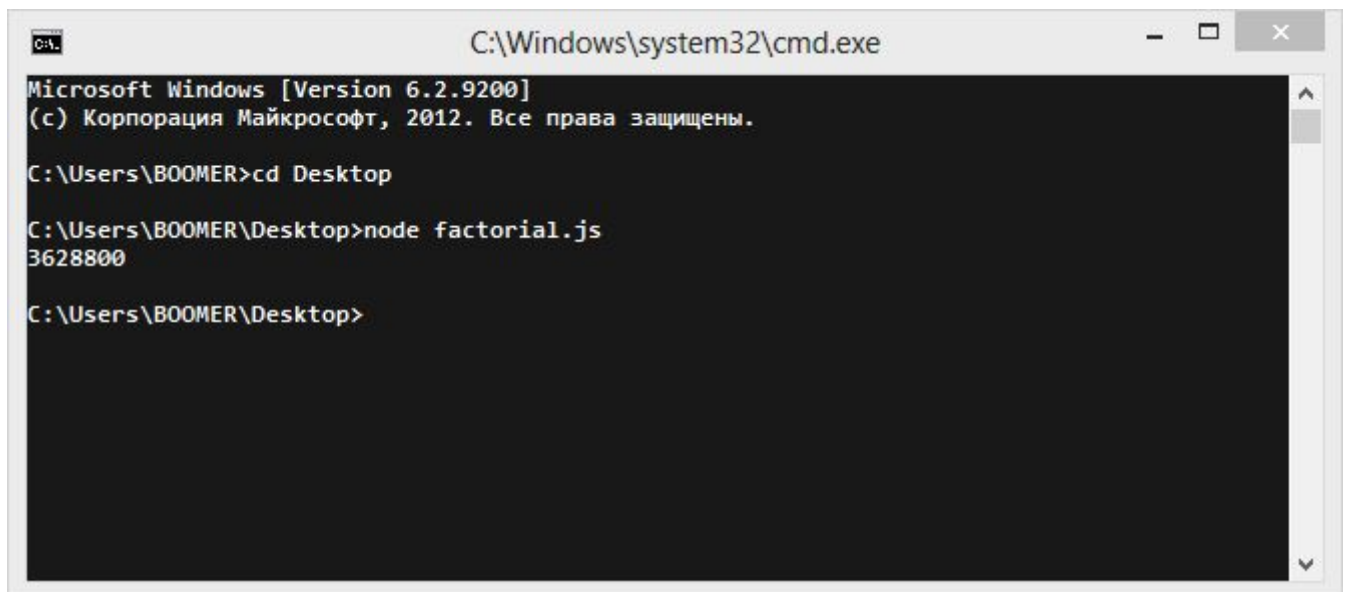
Что такое NodeJS

Node или Node.js — программная платформа, основанная на движке V8 (транспирующем JavaScript в машинный код), превращающая JavaScript из узкоспециализированного языка в язык общего назначения. Node.js добавляет возможность JavaScript взаимодействовать с устройствами ввода-вывода через свой API (написанный на C++), подключать другие внешние библиотеки, написанные на разных языках, обеспечивая вызовы к ним из JavaScript-кода. Предназначен для создания масштабируемых распределённых сетевых приложений, таких как веб-сервер. В отличие от большинства программ JavaScript, этот каркас выполняется не в браузере клиента, а на стороне сервера.

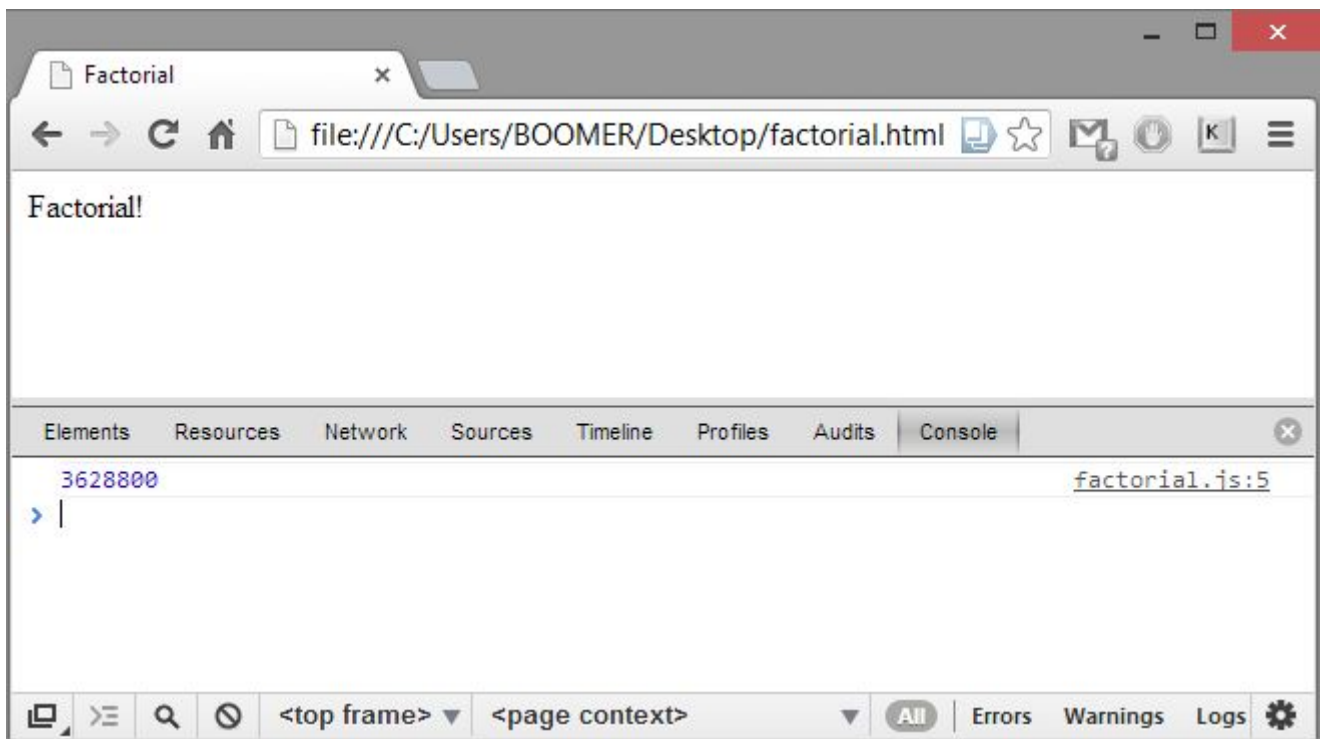
Сравним какой-нибудь код, выполнив его в Node.js и браузере, например, Chrome. Для примера возьмем функцию вычисления факториала:

```
var factorial = function (n, a) {  
  return n < 2 ? a : factorial(n - 1, a * n);  
};  
console.log(factorial(10, 1));
```

Выполнив данный код в Node.js и Chrome получаем одинаковые результаты:



```
C:\Windows\system32\cmd.exe  
Microsoft Windows [Version 6.2.9200]  
(c) Корпорация Майкрософт, 2012. Все права защищены.  
  
C:\Users\BOOMER>cd Desktop  
  
C:\Users\BOOMER\Desktop>node factorial.js  
3628800  
  
C:\Users\BOOMER\Desktop>
```



Рассмотрим другой код, используя прототипное ООП:

```
var Celsius2Fahrenheit = function (val, unit) {  
  this.val = val;  
  this.unit = unit;  
};  
Celsius2Fahrenheit.prototype.setVal = function (degrees) {  
  var degreesArray = degrees.split(" ");  
  this.val = degreesArray[0];  
  this.unit = degreesArray[1];  
};  
Celsius2Fahrenheit.prototype.convert = function (to) {  
  if (this.unit !== to) {  
    this.unit = to;  
    switch (to) {  
      case "C": {  
        this.val = Math.round((this.val - 32) * 5 / 9);  
      } break;  
      case "F": {  
        this.val = Math.round(this.val * 9 / 5 + 32);  
      } break;}}  
};  
var c2f = new Celsius2Fahrenheit(30, "C");  
console.log(c2f.val + c2f.unit);  
c2f.convert("F");  
console.log(c2f.val + c2f.unit);  
c2f.setVal("100 F");  
console.log(c2f.val + c2f.unit);  
c2f.convert("C");  
console.log(c2f.val + c2f.unit);
```

Если прописать его в Node.js и браузере, результаты будут одинаковые.

Отметим основные отличия Node.js.

- 1) Для Node.js написано немало встроенных библиотек, или модулей (а соответственно появилась инструкция для их подключения — `require`).
- 2) В Node.js часто используются callback функциями, в отличие от привычного JavaScript.
- 3) В её основе лежит событийно-ориентированное и асинхронное программирование с неблокирующим вводом/выводом. Платформа предназначена для обособленного выполнения веб-приложений на языке JavaScript.
- 4) Для выполнения JavaScript-кода используется движок V8, который разработан компанией Google.

Что такое ExpressJS

Express.js, или просто Express, каркас web-приложений для Node.js, реализованный как свободное и открытое программное обеспечение под лицензией MIT. Он спроектирован для создания веб-приложений и API. Является стандартным каркасом для Node.js. Он включает большое число подключаемых плагинов. Express является backend для программного стека MEAN, вместе с базой данных MongoDB и каркасом AngularJS для frontend. По своей сути Express.js это Node.js рамки. Он построен вокруг конфигурации и гранулированных простота подключения промежуточного программного обеспечения.

Что такое MongoDB

MongoDB (от англ. humongous — огромный) — документо-ориентированная система управления базами данных (СУБД) с открытым исходным кодом, не требующая описания схемы таблиц.

MongoDB – это нереляционная SQL СУБД, которая хорошо горизонтально масштабируется и имеет структуру хранения. В связи с этим идеально подходит для работы с нереляционными данными (когда сам документ представляет из себя наибольшую ценность).

Устанавливаем и настраиваем NodeJS

Интегрировать Node.js можно с помощью модуля [Node.js integration](#).

Сначала устанавливаем все необходимые пакеты для работы с node.js и сам node.js:

```
$ sudo apt-get install build-essential git curl openssl libssl-dev
```

Самым распространённым методом установки node.js является его компиляция из исходных кодов.

Примечание: необходимо заменить `v.0.8.x` на последнюю стабильную версию с <https://github.com/joyent/node/>

```
$ mkdir -p ~/local/src  
cd ~/local/src  
git clone --branch v0.8.x git://github.com/joyent/node.git  
cd node  
./configure  
make  
sudo make install
```

Если установка прошла без ошибок, то вам откроется доступ к оболочке node.js.

```
$ node  
> console.log("Hello world");
```

Установка модуля node.js на Drupal, после чего переходим в директорию с ним с помощью такой команды:

```
$ cd path/to/your/nodejs/module/directory
```

Добавляем все необходимые зависимости:

```
$sudo npm install  
$sudo npm install socket.io  
$sudo npm install request  
$sudo npm install express  
$sudo npm install connect
```

Далее переходим на страницу настройки node.js. Копируем всё содержимое из поля "Suggested configuration", создаём файл sites/ all/ modules/ nodejs/ nodejs.config.js и добавляем туда скопированные настройки.

CONFIGURATION BUILDER

1. Configure your server settings in the SETTINGS form below.
2. Click the **Save Configuration** button
3. Copy the suggested configuration lines to **sites/all/modules/nodejs/nodejs.config.js**, you need to do it manually.

Suggested configuration:

```
/**
 * This configuration file was built using the 'Node.js server configuration builder'.
 * For a more fully commented example see the file nodejs.config.js.example in the root of this module
 */
backendSettings = {
```

Платформа node.js настроена. Проверить ее работу можно, включив модуль "Nodejs Watchdog" (он входит в пакет модуля Node.js integration).

После установки

Для того, чтобы не прописывать постоянно 'node server.js ', возможно его запустить в фоновом режиме с помощью пакета forever. Forever умеет управлять фоновыми процессами, а именно: перезапускать их после сбоя; перенаправлять стандартный вывод ошибок в файлы журналов, и исполнять другие полезные функции. Устанавливаем его у нас на сервере:

```
$sudo npm install-g forever
```

Теперь вместо 'node server.js' будем писать 'forever start server.js'. Для того, чтобы остановить - прописываем 'forever stop server.js'. Чтобы проверить, какие процессы запущены через forever, можно воспользоваться 'forever list'.

Для того, чтобы node.js запускался автоматически после загрузки/перезагрузки системы, нужно создать скрипт инициализации в каталоге /etc/init.d/.

В нашем случае скрипт выглядит так:

```

<span style="font-family: 'trebuchet ms', geneva;">set -e
PATH=/usr/local/bin:/bin:/usr/bin:/sbin:/usr/sbin
DAEMON=/var/www/node.loc/sites/all/modules/nodejs/server.js
case "$1" in
  start)
    forever start $DAEMON
    ;;
  stop)
    forever stop $DAEMON
    ;;
  force-reload|restart)
    forever restart $DAEMON;;
  *) echo "Usage: /etc/init.d/node {start|stop|restart|force-reload}"
    exit 1
    ;;
esac
exit 0
</span></span>

```

Сохраняем его в файл /etc/init.d/node. Далее прописываем в терминале следующие команды:

```

$ sudo update-rc.d node defaults
$ sudo chmod +x /etc/init.d/node

```

Node.js настроен для корректной работы.

Подключаем ExpressJS

ExpressJS сильно упрощает рутинные операции по созданию сервера, работе с запросами и всякой маршрутизацией. Простейший код для создания сервера, отдающего «hello» на запрос корня:

```

var http = require('http');
var express = require('express');
var app = express();
app.use(express.bodyParser());
app.get('/', function (req, res) {
  res.end('hello!');
});
app.listen(19880);

```

Все функции библиотеки возвращают обратно объект app, поэтому их можно прописать в следующей форме:

```

app.get('/', function (req, res) { res.end('hello!'); }).listen(19880);

```

Чтобы ответить на POST-запрос, просто используйте

```
app.post('/', function (req, res) {});
```

Для ответа на запрос конкретного пути используйте

```
app.get('/index.htm') //реагирует на запрос site/index.htm
app.get('/files/file.htm') // реагирует на запрос site/files/file.htm
```

Для подключения модуля express.js необходима строка:

```
app.use(express.bodyParser())
```

При инициализации модуля она важна. Функция use — это интерфейс подключения дополнительных модулей к express.js. bodyParser — это мета-модуль, объединяющий модули json, urlencoded и multipart. Все их можно подключать по отдельности, и передавать им какие-то параметры. Существует еще несколько модулей, самые полезные — logger и compress, которые включает gzip-сжатие страниц.

Организуем обработку входящих запросов при помощи ExpressJS Router

С помощью класса express.Router можно создавать модульные, монтируемые обработчики маршрутов. Экземпляр Router представляет собой комплексную систему промежуточных обработчиков и маршрутизации; по этой причине его часто называют “мини-приложением”.

В приведённом ниже примере создаётся маршрутизатор в виде модуля, в него загружается функция промежуточной обработки, определяется несколько маршрутов, и модуль маршрутизатора монтируется в путь в основном приложении.

Создайте файл маршрутизатора с именем birds.js в каталоге приложения со следующим содержанием:

```
var express = require('express');
var router = express.Router();
// middleware that is specific to this router
router.use(function timeLog(req, res, next) {
  console.log('Time: ', Date.now());
  next();
});
// define the home page route
router.get('/', function(req, res) {
  res.send('Birds home page');
});
// define the about route
router.get('/about', function(req, res) {
  res.send('About birds');
});
module.exports = router;
```


Загружаем модуль маршрутизации в приложение:

```
var birds = require('./birds');  
...  
app.use('/birds', birds);
```

Данное приложение теперь сможет обрабатывать запросы, адресованные ресурсам /birds и /birds/about, а также вызывать специальную функцию промежуточной обработки timeLog данного маршрута.

Устанавливаем и настраиваем MongoDB

Для установки MongoDB загрузим один распространяемых пакетов с официального сайта <http://www.mongodb.org/downloads>. Официальный сайт предоставляет пакеты дистрибутивов для различных платформ: Windows, Linux, MacOS, Solaris. И различные версии: как 32-х битные, так и 64-х битные. При использовании MongoDB следует учитывать, что MongoDB использует отображаемые в памяти файлы, поэтому на 32-х разрядных системах её действие ограничено 2 Гб памяти. Чтобы поддерживать базы данных большего объема на 32-х разрядных системах, требуются дополнительные усилия. При развертывании на 64-х разрядных системах подобных ограничений нет.

Также следует обратить внимание на версию дистрибутива. На момент написания данного материала последней версией платформы была версия 3.0.3. Использование конкретной версии может несколько отличаться от применения иных версий платформы MongoDB. Для загрузки и установки пакета выберем нужную операционную систему и подходящий тип пакета.

Для ОС Windows загрузочный пакет доступен в виде файла установщика с расширением msi.

После запуска установщика для ОС Windows после принятия лицензионного соглашения отобразится окно выбора типа настроек. Выберем в нем Custom. По умолчанию все файлы устанавливаются по пути C:\Program Files\MongoDB. Но мы изменим каталог установки. Для этого в начале создадим на диске C новую папку mongodb и затем укажем её в качестве места установки:

Все остальные настройки оставим по умолчанию.

Содержимое пакета MongoDB

Если после установки откроем папку C:\mongodb\bin, то сможем найти там множество приложений, которые выполняют определенную роль. Вкратце рассмотрим их.

- **bsondump**: считывает содержимое BSON-файлов и преобразует их в читабельный формат, например, в JSON;
- **mongo**: представляет консольный интерфейс для взаимодействия с базами данных, своего рода консольный клиент;
- **mongod**: сервер баз данных MongoDB. Он обрабатывает запросы, управляет форматом данных и выполняет различные операции в фоновом режиме по управлению базами данных;
- **mongodump**: утилита создания бэкапа баз данных;

- `mongoexport`: утилита для экспорта данных в форматы JSON, TSV или CSV;
- `mongofiles`: утилита, позволяющая управлять файлами в системе GridFS;
- `mongoimport`: утилита, импортирующая данные в форматах JSON, TSV или CSV в базу данных MongoDB;
- `mongooplog`: приложение, позволяющее опрашивать удаленные серверы на наличие операций с БД и применять полученные данные к локальному серверу;
- `mongoperf`: проверяет производительность жесткого диска на предмет операций ввода-вывода;
- `mongorestore`: позволяет записывать данные из дампа, созданного `mongodump`, в новую или существующую базу данных;
- `mongos`: служба маршрутизации MongoDB, которая помогает обрабатывать запросы и определять местоположение данных в кластере MongoDB;
- `mongostat`: представляет счетчики операций с бд;
- `mongotop`: предоставляет способ подсчета времени, затраченного на операции чтения-записи в бд.

Типизируем наши схемы, подключаем Mongoose

После установки надо создать на жёстком диске каталог, в котором будут находиться базы данных MongoDB.

В ОС Windows по умолчанию MongoDB хранит базы данных по пути `C:\data\db`, поэтому, если вы используете Windows, вам надо создать соответствующий каталог. В ОС Linux и MacOS каталогом по умолчанию будет `/data/db`.

Если же возникла необходимость использовать какой-то другой путь к файлам, то его можно передать при запуске MongoDB во флаге `--dbpath`.

После создания каталога для хранения БД можно запустить сервер MongoDB. Сервер представляет приложение `mongod`, которое находится в папке `bin`. Для этого запустим командную строку (в Windows) или консоль в Linux и там введём соответствующие команды. Командная строка отобразит нам ряд служебной информации, например, что сервер запускается на localhost на порту 27017.

Если используется расположение баз данных, отличающееся от настроек по умолчанию, то при запуске можно задать каталог для баз данных следующим образом: `C:\mongodb\bin\mongod.exe --dbpath d:\test\mongodb\data`. В данном случае предполагается, что базы данных у нас будут находиться в каталоге `d:\test\mongodb\data`. И после удачного запуска сервера мы сможем производить операции с бд через оболочку `mongo`.

Второй строкой приложение говорит о подключении к базе данных `test`. Хотя до запуска `mongo.exe` такой базы данных не существовало, MongoDB автоматически создаст её при подключении. В папке `C:\data\db` появится ряд автоматически сгенерированных файлов.

Создадим в бд test. Введем следующие команды:

```
db.users.save( { id: "1", name: "Eugene" } )  
db.users.find()
```

Здесь db представляет текущую базу данных - то есть базу данных test, далее идёт users - это коллекция, в которую затем мы добавляем новый объект. Если в SQL нам надо создавать таблицы заранее, то коллекции MongoDB создает самостоятельно при их отсутствии.

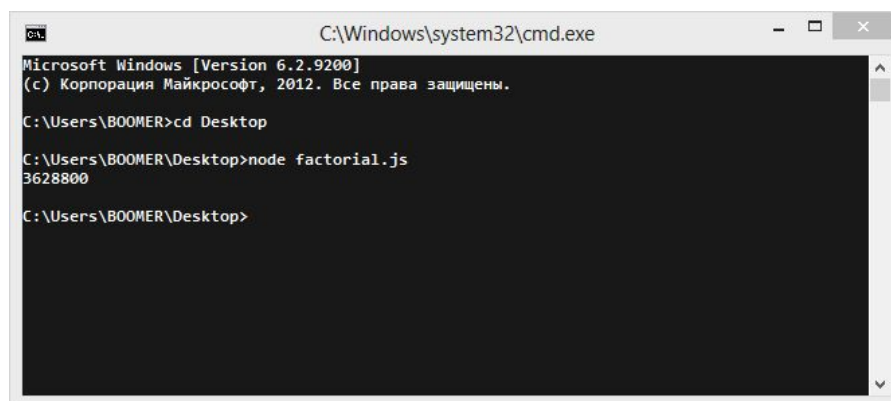
Практика

Задача 1. Сравнение кода, выполненного в Node.js и Chrome

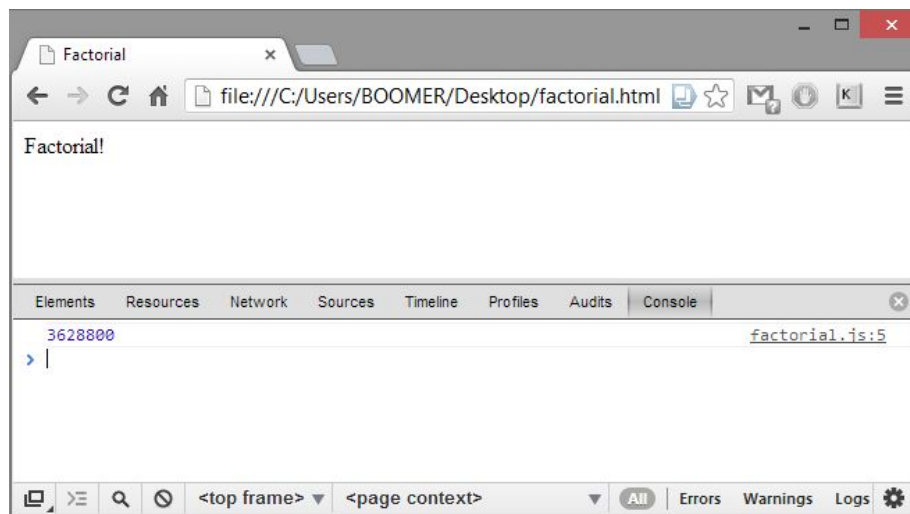
Сравним какой-нибудь код, выполнив его в Node.js и браузере, например Chrome. Для примера возьмём функцию вычисления факториала:

```
var factorial = function (n, a) {  
  return n < 2 ? a : factorial(n - 1, a * n);  
};  
console.log(factorial(10, 1));
```

Выполнив данный код в Node.js и Chrome, получаем одинаковые результаты:



```
C:\Windows\system32\cmd.exe  
Microsoft Windows [Version 6.2.9200]  
(c) Корпорация Майкрософт, 2012. Все права защищены.  
C:\Users\BOOMER>cd Desktop  
C:\Users\BOOMER\Desktop>node factorial.js  
3628800  
C:\Users\BOOMER\Desktop>
```



Рассмотрим другой код, используя прототипное ООП:

```
var Celsius2Fahrenheit = function (val, unit) {  
  this.val = val;  
  this.unit = unit;  
};  
Celsius2Fahrenheit.prototype.setVal = function (degrees) {  
  var degreesArray = degrees.split(" ");  
  this.val = degreesArray[0];  
  this.unit = degreesArray[1];  
};  
Celsius2Fahrenheit.prototype.convert = function (to) {  
  if (this.unit !== to) {  
    this.unit = to;  
    switch (to) {  
      case "C": {  
        this.val = Math.round((this.val - 32) * 5 / 9);  
      } break;  
      case "F": {  
        this.val = Math.round(this.val * 9 / 5 + 32);  
      } break;}}  
};  
var c2f = new Celsius2Fahrenheit(30, "C");  
console.log(c2f.val + c2f.unit);  
c2f.convert("F");  
console.log(c2f.val + c2f.unit);  
c2f.setVal("100 F");  
console.log(c2f.val + c2f.unit);  
c2f.convert("C");  
console.log(c2f.val + c2f.unit);
```

Домашнее задание

Требуется организовать некоторый бэкенд в нашем приложении. Для этого:

1. Необходимо настроить БД и создать требуемую структуру таблиц и их связей (можете выбрать любую БД).

2. Для взаимодействия с БД необходимо предусмотреть некоторое API (веб сервере), которое будет обрабатывать входящие запросы (опять же, можно взять тот же ASP.NET Web API).
3. В самом приложении предусмотрите действия, которые будут отправлять запросы на адреса нашего API.
4. *Реализовать основные CRUD операции для одной из страниц (например, пользователей).
5. *Следует организовать обратную связь API с хранилищем, чтобы поддерживать актуальное состояние дел в хранилище.

Задачи со * повышенной сложности

Дополнительные материалы

1. <http://expressjs.com/ru/>
2. <http://koa.js.com/>
3. <https://github.com/reactjs/express-react-views>
4. <https://ru.wikipedia.org/wiki/CRUD>
5. <https://www.mongodb.com/>
6. <http://mongoosejs.com/>
7. <https://github.com/expressjs/morgan>
8. <https://www.npmjs.com/package/react-native-mongoose>
9. <http://www.slideshare.net/moscowjs/ss-47567934>
10. <http://expressjs.com/ru/guide/routing.html>
11. <http://metanit.com/nosql/mongodb/1.1.php>
12. <http://wisetip.ru/page/mongodb>

Используемая литература

Для подготовки данного методического пособия были использованы следующие ресурсы:

1. <http://www.slideshare.net/moscowjs/ss-47567934>
2. <http://expressjs.com/ru/guide/routing.html>
3. <http://metanit.com/nosql/mongodb/1.1.php>
4. <http://wisetip.ru/page/mongodb>