

**НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА**
Институт радиоэлектроники и информационных технологий
Кафедра “Прикладная математика”

**Лабораторная работа №4 по курсу «Базы данных»
«Node.js, HTTP и SQL + ORM + AJAX»**

Выполнила:
Горенкова А. В.
Группа 16-ПМ
Проверил:
Моисеев А.Е.

НИЖНИЙ НОВГОРОД
2018 г.

Оглавление

Введение.....	3
Задание.....	6
Решение.....	7
Результаты работы.....	10
Листинг.....	14

Введение

Работа с базами данных предполагает создание какой-либо СУБД. Однако не всегда требуется написание приложения для операционной системы, т. к. работа со многими базами данных ведется удаленно, а именно по сети интернет, то есть необходим удобный пользователю web-интерфейс. Решением данной проблемы может стать использование языка программирования *JavaScript*.

JavaScript изначально создавался для того, чтобы сделать web-странички «живыми». Программы на этом языке называются скриптами. В браузере они подключаются напрямую к HTML и, как только загружается страничка – тут же выполняются. Однако работа с БД подразумевает работу с сервером, и для этого мы можем воспользоваться Node.js (или просто Node) — серверной платформой для работы с JavaScript. В то время как JavaScript выполняет действие на стороне клиента, программы Node выполняются на сервере. Node.js добавляет возможность JavaScript взаимодействовать с устройствами ввода-вывода через свой API (написанный на C++), подключать другие внешние библиотеки, обеспечивая вызовы к ним из JavaScript-кода.

Таким образом, мы можем создать СУБД, основанную на платформе Node.js. Node.js поддерживает работу со всеми популярными системами управления базами данных, в том числе и с SQL-системами. Одна из таких популярных баз данных — это SQLite — компактная встраиваемая реляционная база данных, то есть движок SQLite не является отдельно работающим процессом, с которым взаимодействует программа, а предоставляет библиотеку, с

которой программа компонуется и движок становится составной частью программы. С помощью SQLite становится возможной реализация web-интерфейса работы с базой данных с использованием технологии ORM.

Однако, для создания web-интерфейса недостаточно работы с серверной платформой Node.js. Необходимо отобразить данные и сделать графический интерфейс, понятный пользователю. Для этого нам потребуется использовать HTML (*HyperText Markup Language* — «язык гипертекстовой разметки») — стандартизированный язык разметки документов во Всемирной паутине. С его помощью мы сможем создать страничку, которая отображала бы данные, получаемые с сервера.

Таким образом, для создания СУБД потребуется использование платформы Node.js, языка JavaScript, языка разметки HTML и языка запросов SQLite. Но в ходе работы появляется ещё одна проблема: для отображения новых данных требуется каждый раз обновить всю страницу, что отнимает лишние ресурсы. Следовательно необходимо использовать механизмы, позволяющие обновлять только часть страницы, отвечающей за вывод конкретной информации. С развитием технологий появляется и решение такой проблемы.

AJAX (*Asynchronous Javascript and XML* — «асинхронный JavaScript и XML») — подход к построению интерактивных пользовательских интерфейсов веб-приложений, заключающийся в «фоновом» обмене данными браузера с веб-сервером. При использовании AJAX нет необходимости обновлять каждый раз всю страницу, так как обновляется только ее конкретная часть. Это намного удобнее, так как не приходится долго ждать, и экономичнее,

так как не все обладают безлимитным интернетом. AJAX использует два метода работы с веб-страницей: изменение web-страницы не перезагружая её, и динамическое обращение к серверу. Для того, чтобы осуществлять обмен данными, на странице должен быть создан объект XMLHttpRequest, который является своеобразным посредником между Браузером пользователя и сервером. С помощью XMLHttpRequest можно отправить запрос на сервер, а также получить ответ в виде различного рода данных. AJAX использует асинхронную передачу данных. Это значит, что пока идёт передача данных, пользователь может совершать другие, необходимые ему действия. Имея большое количество плюсов концепция AJAX быстро нашла применение при создании web-страниц и в работе с серверами.

Задание

- Создать статическую таблицу, используя платформу Node.js и язык разметки Html.
- Создать реляционную базу данных и сделать несколько SQL-запросов из приложения на Node.js
- Создать с помощью AJAX возможность вывода отфильтрованных данных по пользовательскому запросу, при этом обновляя только часть страницы с таблицей

Решение

1. Создание статической таблицы

(a) Создаём сервер

```
var http = require('http');
var server = http.createServer(function(req, res) {...}).listen(3000);
```

(b) Создаём страничку на html

```
<!DOCTYPE html>
<HTML>
<HEAD>
<TITLE>
  Hi
</TITLE>
</HEAD>
<BODY>
<table>
<tr><td>Table</td><td>Brown</td><td>
<tr><td>Cupboard</td><td>Red</td><td>
...
</table></BODY>
</HTML>end
```

(c) Отправляем страничку

```
res.write('<!DOCTYPE html>\n <HTML>\n ... </BODY>\n </HTML>');
```

2. Выполнение SQL-запросов к таблице

(a) Создаём таблицу и добавляем мебель

```
var db = new sqlite3.Database("furniture.db");
db.run("CREATE TABLE IF NOT EXISTS Furniture (name TEXT, color TEXT, price int)");
var stmt=db.prepare("INSERT INTO Furniture VALUES(?,?,?)");
stmt.run("Table", "brown", 3000);
stmt.run("Shelf", "green", 500);
...
```

(b) Создаём сервер

```
http.createServer(function(req,res){ ... }).listen(3000);
```

(c) Выполняем SQL-запрос

```
db.all("SELECT * FROM Furniture WHERE price < 1000", function(err, rows) { ... });
```

(d) Генерируем html-код с полученными данными

```
res.write("<table>\n");
for(var i = 0; i < rows.length; ++i) {
res.write("<tr><td>" + rows[i].name + "</td><td>" + rows[i].color + "</td><td>" + rows[i].price + "</td></tr>\n");
}
res.write("</table>\n");
```

3. Вывод отфильтрованных данных

(a) Создаём html-страницу

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset = 'utf-8'>
  </head>
  <body>
    <input type="text" id="search" value="Search by color" onFocus="this.value=";
document.getElementById('resultdiv').style.display='none';"/>
    <input type="button" value="Search"
onClick="call1(document.getElementById('search').value);"/>
    <input type="button" value="Get all" onClick="call2();"/>
    <p>
      <span id = "call_res"></span>
    </p>
  </body>
</html>
```

(b) Добавляем JavaScript-функции call1, call2

```
function call1 (color) {
  readServerString ("/call1?color="+color, function (err, response) {
    document.getElementById ("call_res").innerHTML = "<span style = 'color:red'>" + err +
"</span>";
  });
}
function call2 () {
```



```

readServerString ("/call2", function (err, response) {
    document.getElementById ("call_res").innerHTML = "<span style = 'color:red'>" + err +
"</span>";
    });
}

```

(с) Создаём и заполняем базу данных

```

var db = new sqlite3.Database("furniture.db");
db.run("CREATE TABLE IF NOT EXISTS Furniture (name TEXT, color TEXT, price int)");
var stmt=db.prepare("INSERT INTO Furniture VALUES(?,?,?)");
stmt.run("Table", "brown", 3000);
stmt.run("Shelf", "green", 500);
...

```

(d) Создаём сервер

```

http.createServer(function(req,res){ ... }).listen(3000);

```

(е) Обрабатываем запрос со страницы

```

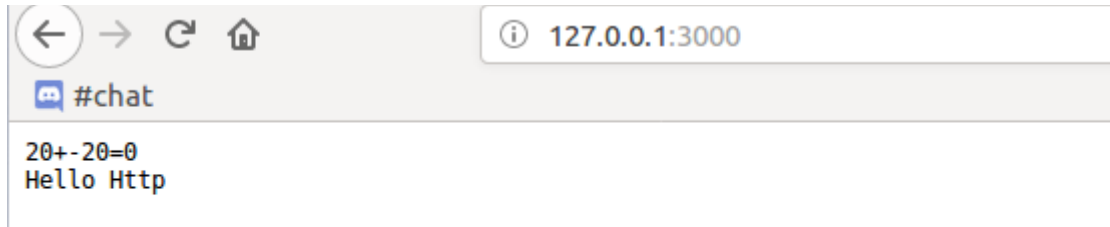
switch (params.pathname) {
    case "/": {
        ...
        break;
    }
    case "/call1": {
        ...
        break;
    }
    case "/call2": {
        ...
        break;
    }
    default: {
        res.writeHead (404, {'Content-Type': 'text/html'});
        res.end ('Error, page not found: ' + req.url);
    }
}
}

```

Результаты работы

1. Вывод статической таблицы

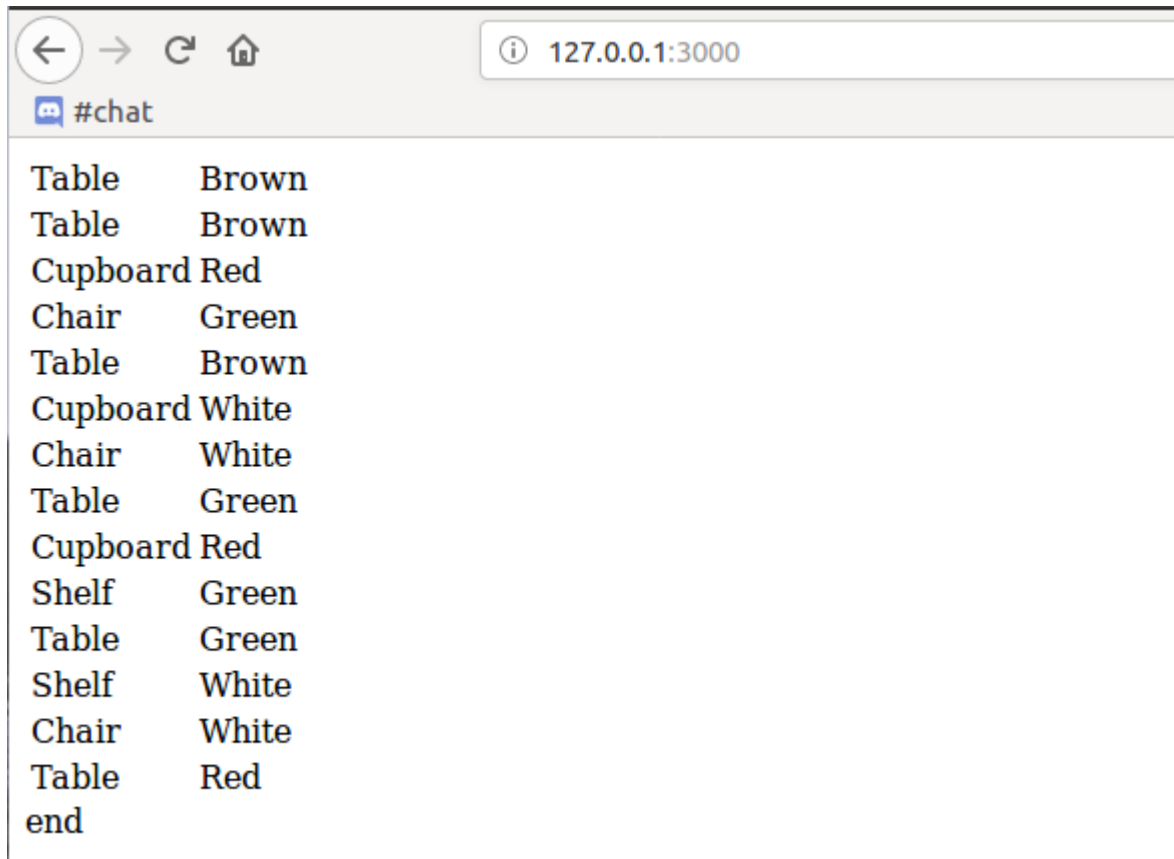
(1) Hello world на node.js с использованием html



(2) Вывод отформатированных данных

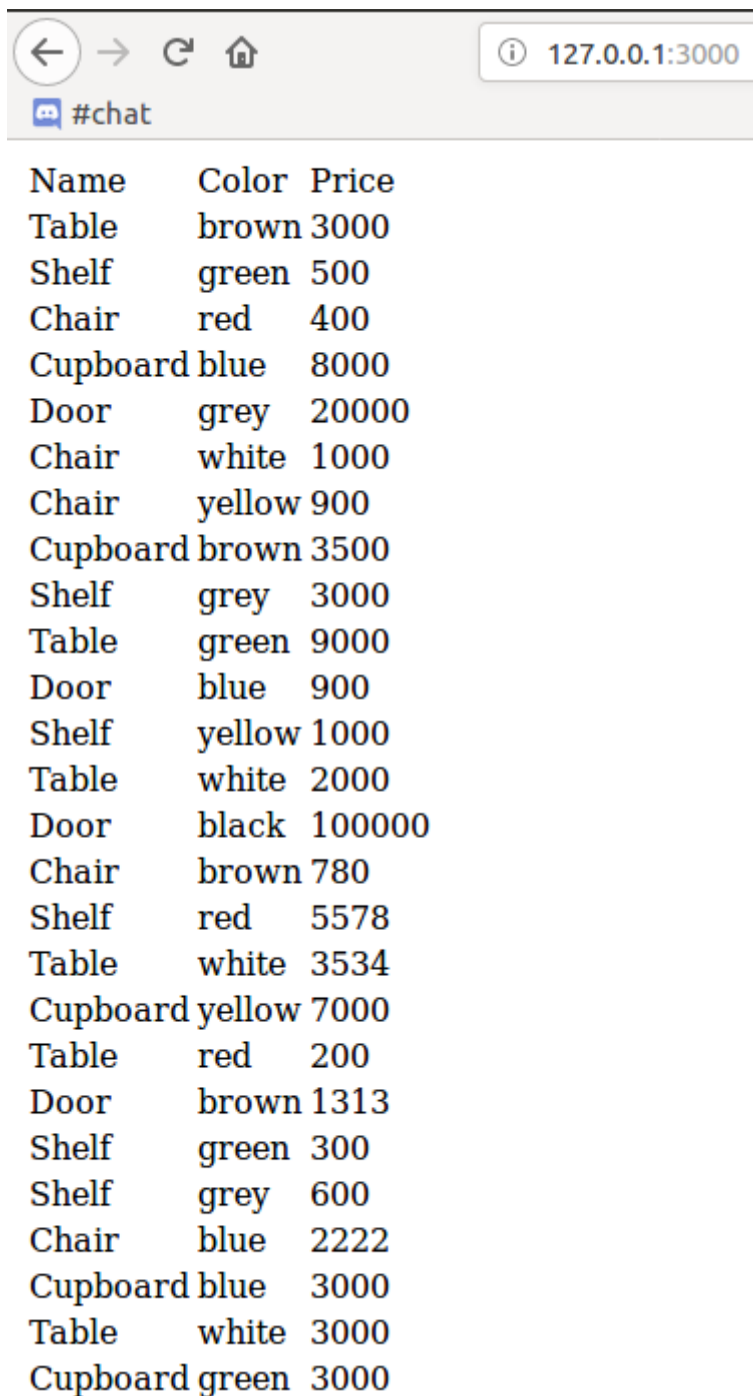


(3) Вывод таблицы



2. Выполнение SQL-запросов к таблице

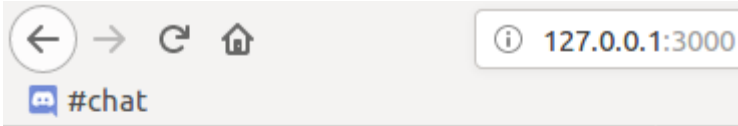
(1) Вывод всей таблицы мебели



The screenshot shows a web browser window with a chat interface. The address bar displays '127.0.0.1:3000' and the chat window title is '#chat'. The chat content is a table with three columns: Name, Color, and Price. The table lists 25 items of furniture, including Tables, Shelves, Chairs, and Cupboards, each with a specific color and price.

Name	Color	Price
Table	brown	3000
Shelf	green	500
Chair	red	400
Cupboard	blue	8000
Door	grey	20000
Chair	white	1000
Chair	yellow	900
Cupboard	brown	3500
Shelf	grey	3000
Table	green	9000
Door	blue	900
Shelf	yellow	1000
Table	white	2000
Door	black	100000
Chair	brown	780
Shelf	red	5578
Table	white	3534
Cupboard	yellow	7000
Table	red	200
Door	brown	1313
Shelf	green	300
Shelf	grey	600
Chair	blue	2222
Cupboard	blue	3000
Table	white	3000
Cupboard	green	3000

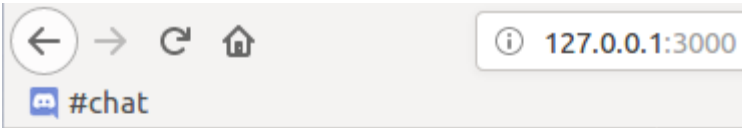
(2) Вывод столов из таблицы



The screenshot shows a web browser window with a chat interface. The address bar displays '127.0.0.1:3000' and the chat window title is '#chat'. The chat content displays a table with three columns: Name, Color, and Price. The table lists seven items, all of which are tables.

Name	Color	Price
Table	brown	3000
Table	green	9000
Table	white	2000
Table	white	3534
Table	red	200
Table	white	3000

(3) Вывод всех предметов мебели с ценой < 1000

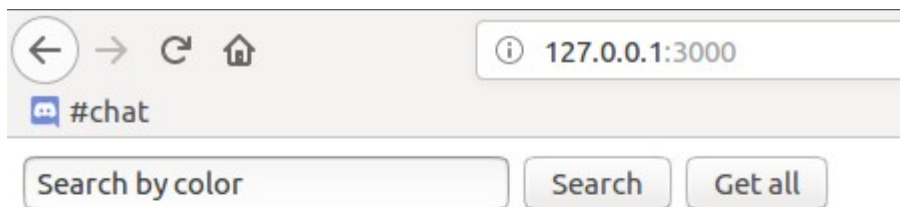


The screenshot shows a web browser window with a chat interface. The address bar displays '127.0.0.1:3000' and the chat window title is '#chat'. The chat content displays a table with three columns: Name, Color, and Price. The table lists nine items, including shelves, chairs, and a door, all with prices less than 1000.

Name	Color	Price
Shelf	green	500
Chair	red	400
Chair	yellow	900
Door	blue	900
Chair	brown	780
Table	red	200
Shelf	green	300
Shelf	grey	600

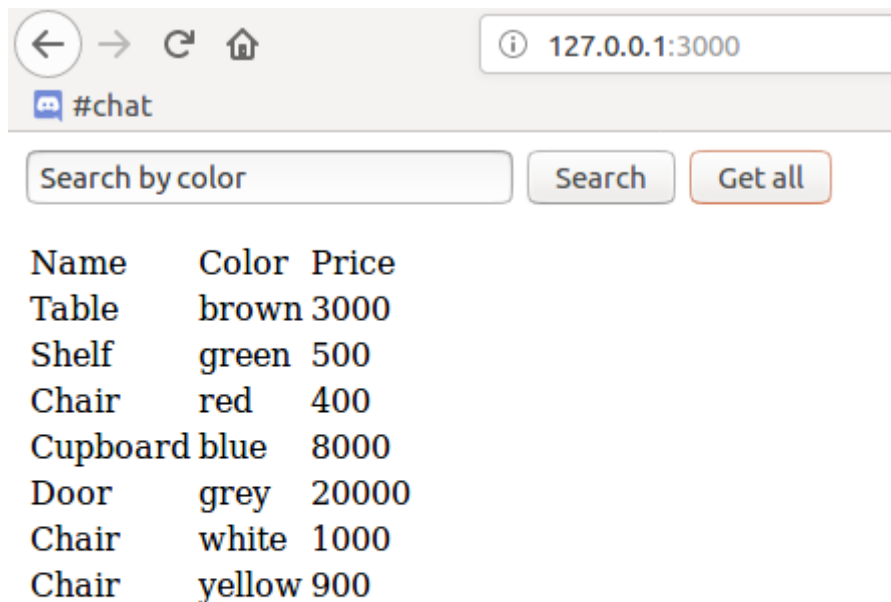
3. Вывод отфильтрованных данных с использованием AJAX

Страница до нажатия на кнопки:



A screenshot of a web browser window. The address bar shows '127.0.0.1:3000'. The page title is '#chat'. Below the title, there is a search bar with the placeholder text 'Search by color'. To the right of the search bar are two buttons: 'Search' and 'Get all'.

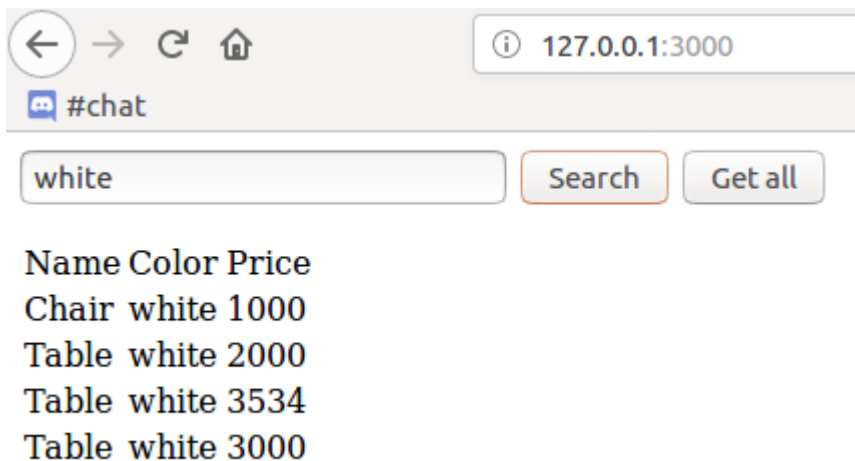
Страница после нажатия «Get all»:



A screenshot of a web browser window showing the same interface as before, but with a table of furniture items displayed below the search bar. The table has three columns: Name, Color, and Price. The data is as follows:

Name	Color	Price
Table	brown	3000
Shelf	green	500
Chair	red	400
Cupboard	blue	8000
Door	grey	20000
Chair	white	1000
Chair	yellow	900

Вывод всех белых предметов мебели:



A screenshot of a web browser window showing the same interface as before, but with the search bar containing the text 'white'. The 'Search' button is highlighted, and the table below shows only the white furniture items:

Name	Color	Price
Chair	white	1000
Table	white	2000
Table	white	3534
Table	white	3000

Листинг

1/1: Helloworld1.js:

```
var http = require('http');
var server = http.createServer(function(req, res) {
  res.writeHead(200, {'Content-Type': 'text/plain'});
  var x = 20
  var y = -20
  res.write('+x+'+'y+'+'=(x+y)+'\n');
  res.end('Hello Http');
}).listen(3000);
```

1/2: Helloworld2.js:

```
var http = require('http');
var server = http.createServer(function(req, res) {
  res.writeHead(200, {'Content-Type': 'text/html'});
  var x = 20
  var y = -20
  res.write('<!DOCTYPE html>\n'+
'<HTML>\n' +
'<HEAD>\n' +
'  <TITLE>\n'+
'    Hi\n' +
'  </TITLE>\n'+
' </HEAD>\n'+
'<BODY>\n'+
'+x+'+'y+'+'=(x+y)+'\n'+
'</BODY>\n'+
' </HTML>');
  res.end('Hello Http');
}).listen(3000);
```

1/3: Helloworld3.js:

```
var http = require('http');
var server = http.createServer(function(req, res) {
  res.writeHead(200, {'Content-Type': 'text/html'});
```

```

res.write('<!DOCTYPE html>\n'+
'<HTML>\n' + '<HEAD>\n' +
'<TITLE>\n'+ ' Hi\n' +
'</TITLE>\n'+ ' </HEAD>\n'+
'<BODY>\n'+
'<table>'+
'<tr><td>' + 'Table' + '</td><td>' + 'Brown' + '</td><td>' +
'<tr><td>' + 'Table' + '</td><td>' + 'Brown' + '</td><td>' +
'<tr><td>' + 'Cupboard' + '</td><td>' + 'Red' + '</td><td>' +
'<tr><td>' + 'Chair' + '</td><td>' + 'Green' + '</td><td>' +
'<tr><td>' + 'Table' + '</td><td>' + 'Brown' + '</td><td>' +
'<tr><td>' + 'Cupboard' + '</td><td>' + 'White' + '</td><td>' +
'<tr><td>' + 'Chair' + '</td><td>' + 'White' + '</td><td>' +
'<tr><td>' + 'Table' + '</td><td>' + 'Green' + '</td><td>' +
'<tr><td>' + 'Cupboard' + '</td><td>' + 'Red' + '</td><td>' +
'<tr><td>' + 'Shelf' + '</td><td>' + 'Green' + '</td><td>' +
'<tr><td>' + 'Table' + '</td><td>' + 'Green' + '</td><td>' +
'<tr><td>' + 'Shelf' + '</td><td>' + 'White' + '</td><td>' +
'<tr><td>' + 'Chair' + '</td><td>' + 'White' + '</td><td>' +
'<tr><td>' + 'Table' + '</td><td>' + 'Red' + '</td><td>' +
'</table>'+
'</BODY>\n'+ '</HTML>');
res.end('end');
}).listen(3000);

```

2: lab4.js:

```

var http=require("http");
var sqlite3=require("sqlite3").verbose();
var db = new sqlite3.Database("furniture.db");
// db.run("CREATE TABLE IF NOT EXISTS Furniture (name TEXT, color TEXT, price int)");
// var stmt=db.prepare("INSERT INTO Furniture VALUES(?,?,?)");
// stmt.run("Table", "brown", 3000);
// stmt.run("Shelf", "green", 500);
// stmt.run("Chair", "red", 400);
// stmt.run("Cupboard", "blue", 8000);

```

```

// stmt.run("Door", "grey", 20000);
// stmt.run("Chair", "white", 1000);
// stmt.run("Chair", "yellow", 900);
// stmt.run("Cupboard", "brown", 3500);
// stmt.run("Shelf", "grey", 3000);
// stmt.run("Table", "green", 9000);
// stmt.run("Door", "blue", 900);
// stmt.run("Shelf", "yellow", 1000);
// stmt.run("Table", "white", 2000);
// stmt.run("Door", "black", 100000);
// stmt.run("Chair", "brown", 780);
// stmt.run("Shelf", "red", 5578);
// stmt.run("Table", "white", 3534);
// stmt.run("Cupboard", "yellow", 7000);
// stmt.run("Table", "red", 200);
// stmt.run("Door", "brown", 1313);
// stmt.run("Shelf", "green", 300);
// stmt.run("Shelf", "grey", 600);
// stmt.run("Chair", "blue", 2222);
// stmt.run("Cupboard", "blue", 3000);
// stmt.run("Table", "white", 3000);
// stmt.run("Cupboard", "green", 3000);
// stmt.finalize();
// db.close();

http.createServer(function(req,res){
    res.writeHead(200,{ 'Content-Type': 'text/html; charset=utf-8'});
    res.write(
        '<!DOCTYPE html>\n' +
        '<html>\n'+
        '<head>\n' +
        '<meta charset=\'utf-8\'>\n' +
        '</head>\n'+
        '<body>\n'
    );

```



```

db.all("SELECT * FROM Furniture ", function(err, rows) {
// db.all("SELECT * FROM Furniture WHERE name = 'Table'", function(err, rows) {
// db.all("SELECT * FROM Furniture WHERE price < 1000", function(err, rows) {
    if(err) {
        res.write("<div style='font-size: 30px; color:red'>" + err + "</div>\n");
    }
    else {
        res.write("<table>\n");
        res.write ("<tr><td>Name</td><td>Color</td><td>Price</td></tr>\n");
        for(var i = 0; i < rows.length; ++i) {

res.write("<tr><td>" + rows[i].name + "</td><td>" + rows[i].color + "</td><td>" + rows[i].price + "</td></tr>
>\n");

        }
        res.write("</table>\n");
    }
    res.end(
        '<body>\n'+
        '<html>\n'
    );
});

}).listen(3000);
console.log("run at 3000");

```

3/1: lab4.js:

```

var http=require("http");
var fs = require("fs");
var url = require("url");

var sqlite3=require("sqlite3").verbose();
var db = new sqlite3.Database("furniture.db");

// db.run("CREATE TABLE IF NOT EXISTS Furniture (name TEXT, color TEXT, price int)");

// var stmt=db.prepare("INSERT INTO Furniture VALUES(?,?,?)");

```

```

// stmt.run("Table", "brown", 3000);
// stmt.run("Shelf", "green", 500);
// stmt.run("Chair", "red", 400);
// stmt.run("Cupboard", "blue", 8000);
// stmt.run("Door", "grey", 20000);
// stmt.run("Chair", "white", 1000);
// stmt.run("Chair", "yellow", 900);
// stmt.run("Cupboard", "brown", 3500);
// stmt.run("Shelf", "grey", 3000);
// stmt.run("Table", "green", 9000);
// stmt.run("Door", "blue", 900);
// stmt.run("Shelf", "yellow", 1000);
// stmt.run("Table", "white", 2000);
// stmt.run("Door", "black", 100000);
// stmt.run("Chair", "brown", 780);
// stmt.run("Shelf", "red", 5578);
// stmt.run("Table", "white", 3534);
// stmt.run("Cupboard", "yellow", 7000);
// stmt.run("Table", "red", 200);
// stmt.run("Door", "brown", 1313);
// stmt.run("Shelf", "green", 300);
// stmt.run("Shelf", "grey", 600);
// stmt.run("Chair", "blue", 2222);
// stmt.run("Cupboard", "blue", 3000);
// stmt.run("Table", "white", 3000);
// stmt.run("Cupboard", "green", 3000);

// stmt.finalize();
// db.close();

http.createServer(function(req,res){
    var params = url.parse('http://localhost:3000' + req.url);
    switch (params.pathname) {
        case "/": {
            fs.readFile("./index.html", function(err, content) {

```

```

        if (!err) {
            res.writeHead(200, {'Content-Type': 'text/html'});
            res.end(content);
        }
        else {
            res.writeHead(500, {'Content-Type': 'text/html'});
            res.end('Error: ' + err.message);
        }
    });
    break;
}

case "/call1": {
    res.writeHead(200, {'Content-Type': 'text/plain; charset = utf-8'});
    color = params.query.slice(params.query.indexOf("=") + 1)
    db.all("SELECT * FROM Furniture WHERE color = '" + color + "'",
function(err, rows) {
        if(err) {
            res.write("<div style = 'font-size:30px; color:red'>");
        }
        else {
            res.write("<table>\n");

            res.write("<tr><td>Name</td><td>Color</td><td>Price</td></tr>\n");
            for(var i = 0; i < rows.length; ++i) {

res.write("<tr><td>" + rows[i].name + "</td><td>" + rows[i].color + "</td><td>" + rows[i].price + "</td></tr>\n");
            }
            res.write("</table>\n");
        }
        res.end();
    });
    break;
}

case "/call2": {
    res.writeHead (200, {'Content-Type': 'text/plain; charset = utf-8'});

```

```

        db.all("SELECT * FROM Furniture ", function(err, rows) {
            if (err) {
                res.write("<div style = 'font-size:30px; color:red'>");
            }
            else {
                res.write("<table>\n");
                res.write
("<tr><td>Name</td><td>Color</td><td>Price</td></tr>\n");
                for(var i = 0; i < rows.length; ++i) {

res.write("<tr><td>" + rows[i].name + "</td><td>" + rows[i].color + "</td><td>" + rows[i].price + "</td></tr>
>\n");

                }
                res.write("</table>\n");
            }
            res.end();
        });
        break;
    }
    default: {
        res.writeHead (404, {'Content-Type': 'text/html'});
        res.end ('Error, page not found: ' + req.url);
    }
}

}).listen(3000);
console.log("run at 3000");

```

3/2: index.html:

```

<!DOCTYPE html>
<html>
<head>
    <meta charset = 'utf-8'>
    <script type = text/javascript>
        function readServerString (url, callback) {
            var req = new XMLHttpRequest();

```

```

req.onreadystatechange = function() {
    if (req.readyState == 4) {
        if (req.status == 200) {
            callback (undefined, req.responseText)
        }
        else {
            callback (new Error(req.status));
        }
    }
}

req.open ('POST', url, true)
req.setRequestHeader ("Content-Type", "application/x-www-form-urlencoded");
req.send();
}

function call1 (color) {
    readServerString ("/call1?color="+color, function (err, response) {
        if (!err) {
            document.getElementById ("call_res").innerHTML = response;
        }
        else {
            document.getElementById ("call_res").innerHTML = "<span style = 'color:red'>" + err +
"</span>";
        }
    });
}

function call2 () {
    readServerString ("/call2", function (err, response) {
        if (!err) {
            document.getElementById ("call_res").innerHTML = response;
        }
        else {
            document.getElementById ("call_res").innerHTML = "<span style = 'color:red'>" + err +
"</span>";
        }
    });
}

```

```
</script>
</head>
<body>
  <input type="text" id="search" value="Search by color" onFocus="this.value=";
document.getElementById('resultdiv').style.display='none';"/>
  <input type="button" value="Search"
onClick="call1(document.getElementById('search').value);"/>
  <input type="button" value="Get all" onClick="call2();"/>
  <p>
    <span id = "call_res"></span>
  </p>
</body>
</html>
```