

Node.js

Игорь Борисов
<http://igor-borisov.ru>

Node.js

Игорь Борисов
<http://igor-borisov.ru>

Темы курса

- Основы Node.js
- События
- Потоки
- Модули
- Фреймворк Express
- Socket.IO
- Создание веб-приложения

Основы Node.js

Игорь Борисов
<http://igor-borisov.ru>

Темы модуля

- Что такое Node.js?
- Установка Node.js
- Обзор V8 JavaScript Engine
- Blocking vs Non-blocking
- Чтение файла
- Создание веб-сервера
- Использование веб-сервера

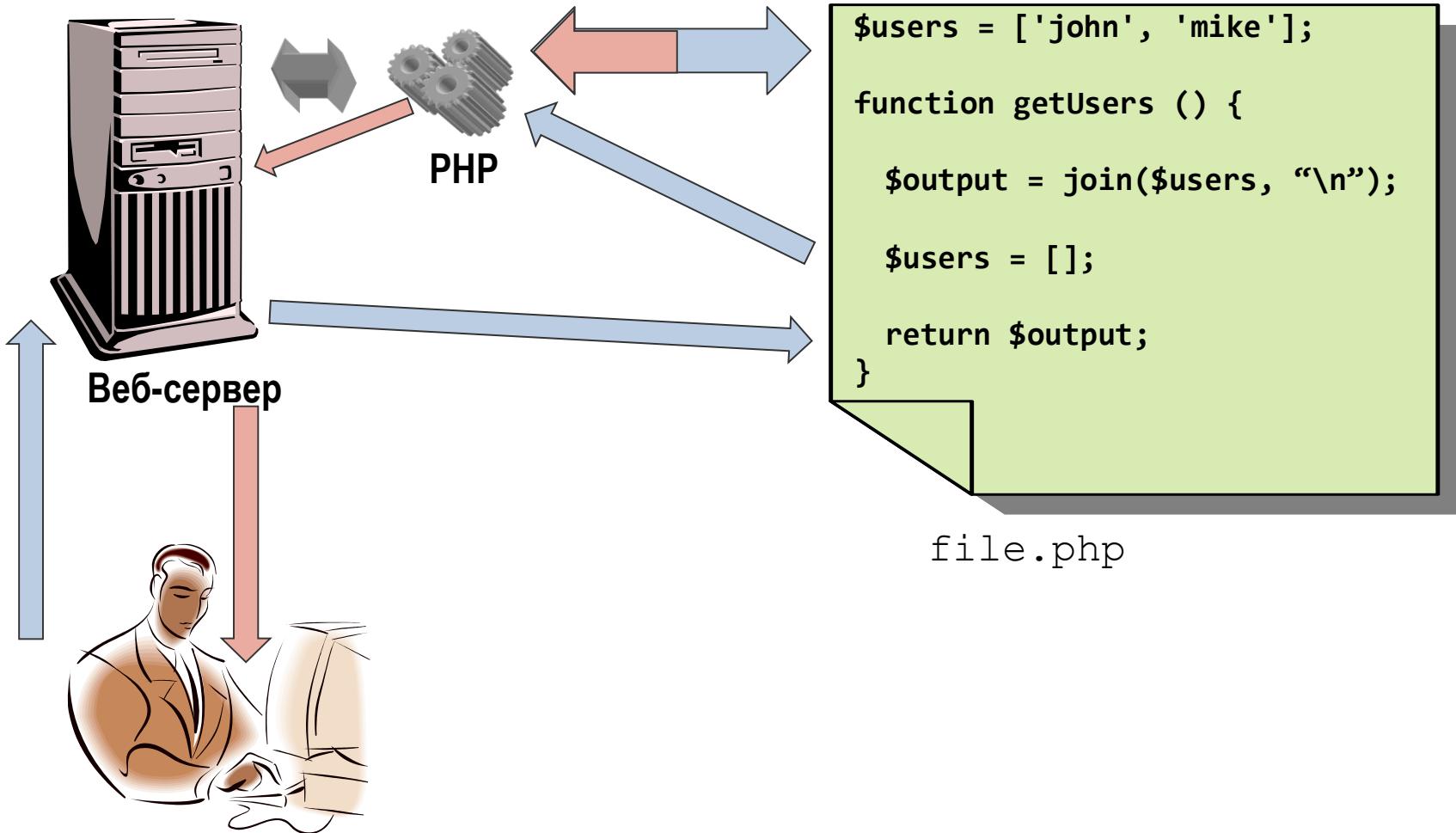
Начало

- <http://nodejs.org>
- <http://nodejs.org/download/>
- Консоль
 - >node
 - >console.log("Hello, world!")
 - Ctrl + C

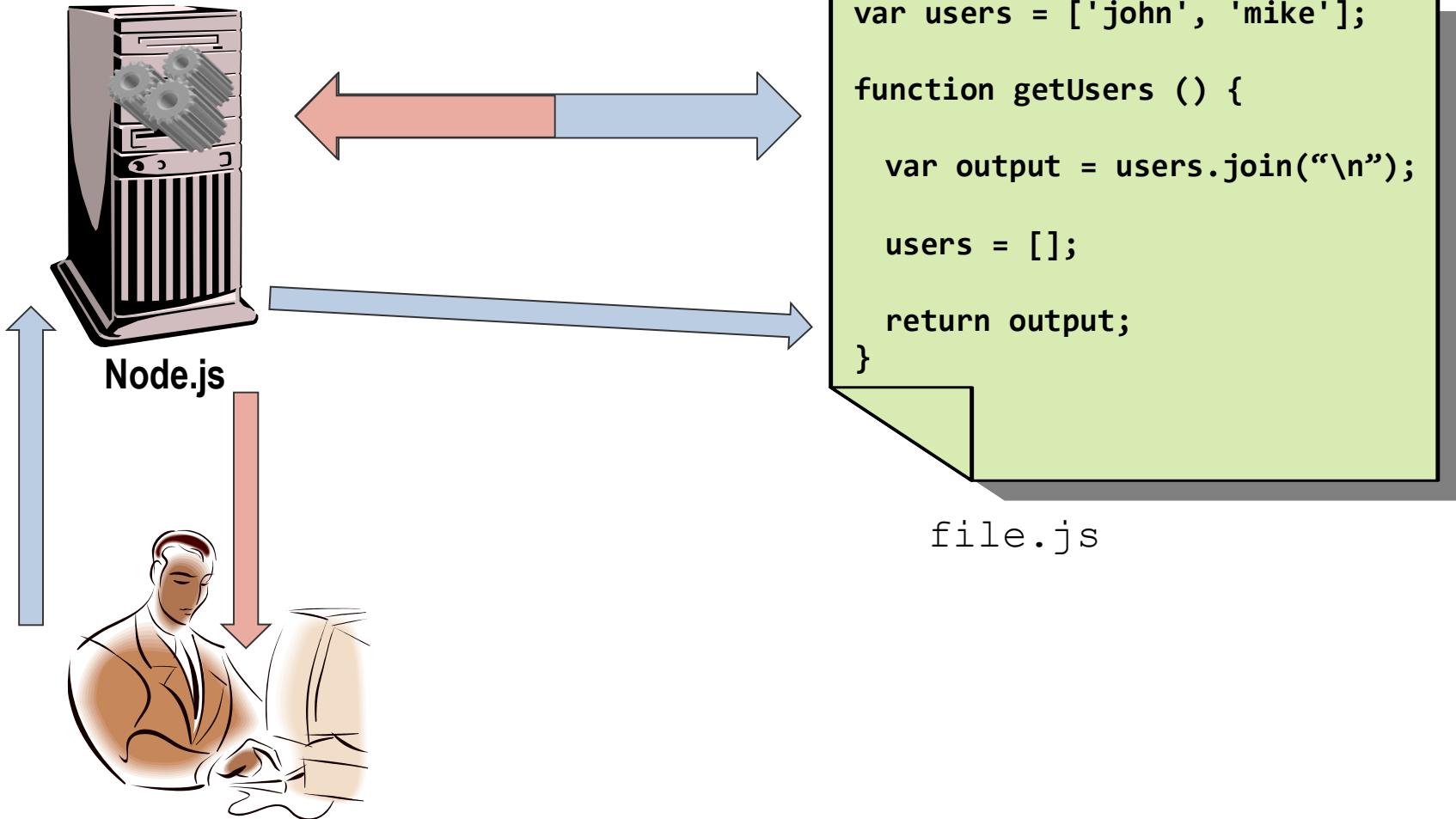
V8 JavaScript Engine

- `Object.keys({a: “b”, b: “c”})`
- `Array.isArray([])`
- `[1, 2, 3].forEach(function(v){})`
- `[1, 2, 3].map(function(v){return v;})`
- `“ b ”.trim()`
- `JSON.parse(‘{“a” : “b”, “b” : “c”}’)`

Blocking vs Non-blocking (PHP)



Blocking vs Non-blocking (Node)



Читаем файл

- `var fs = require('fs');`
- `var txt = fs.readFileSync('file.txt');`
- `fs.readFile('file.txt',
 function(error, txt){});`

Создаём веб-сервер

- `var http = require('http');`
- `function foo(request, response){
 response.writeHead(200);
 response.write('HELLO');
 response.end();
}`
- `var app = http.createServer(foo);`
- `app.listen(8080);`

Лабораторная работа – 1

- Создание веб-сервера и возврат содержимого файла в качестве ответа
 - Файл `labs\lab-1.js`

Выводы

- Введение в Node.js
- Установка Node.js
- Обзор V8 JavaScript Engine
- Blocking vs Non-blocking
- Чтение файла
- Создание веб-сервера
- Использование веб-сервера

События

Игорь Борисов
<http://igor-borisov.ru>

Темы модуля

- События JavaScript
 - Встроенный вызов
 - Пользовательский вызов
- События http-сервера Node.js
- Пользовательские события Node.js

DOM и события

- Встроенный вызов

- `window.onload = function(){};`

- Пользовательский вызов

- `setTimeout('sendForm()', 3000);`

- `function sendForm(){
 form.submit();
};`

События http-сервера

- `var http = require('http');`
- `var app = http.createServer();`
- `app.on('событие',
 function(req, resp){});`

- `request`
- `connection`
- `listening`
- `close`

Полезные методы и свойства

- `var http = require('http');`
- `var app = http.createServer(foo);`
- `function foo(request, response){
 request.method;
 request.url;
 request.statusCode;
 app.close();
}`

Лабораторная работа – 2-1

- Использование событий http-сервера
 - Файл labs\lab-2-1.js

Дополнительные операции

- `request.connection.destroy();`
- `app.on('connection', function(stream){
 stream.setTimeout(3000);
});`
- `var url = require('url');`
- `console.log(url.parse('/page.js?a=b'));`

Пользовательские события

- `var events = require('events');`
- `var Emitter = events.EventEmitter;`
- `var e = new Emitter;`

- `e.on('myEvent', onMyEvent);`
- `function onMyEvent(param){`
 `console.log(param);`
}

- `e.emit('myEvent', 'Привет всем!');`

Лабораторная работа – 2-2

- Использование пользовательских событий
 - Файл labs\lab-2-2.js

Выводы

- События JavaScript
 - Встроенный вызов
 - Пользовательский вызов
- События http-сервера Node.js
- Пользовательские события Node.js

Потоки

Игорь Борисов
<http://igor-borisov.ru>

Темы модуля

- HTTP-метод POST
- Использование cURL
- Способы обработки запроса
- Чтение из файла и запись в файл
- Загрузка файла на сервер

Метод POST

```
POST /script.js HTTP/1.1↙  
Host: www.example.ru↙  
User-Agent: Mozilla/5.0...↙  
Accept: */*↙  
Accept-Language: ru,en-us↙  
Keep-Alive: 300↙  
Connection: keep-alive↙  
Referer: http://www.example.ru/↙  
Content-Type: application/x-www-form-urlencoded↙  
Content-Length: 26↙  
↙  
login=vasya&password=parol↙  
↙
```

cURL

- <http://curl.haxx.se/download.html>
 - curl --data ‘**данные**’ [URL]
 - curl --data-urlencode ‘**данные**’ [URL]
 - curl --form field=value [URL]
 - curl --form field=@filename [URL]
 - curl --upload-file filename [URL]

Обработка запроса

- `request.on('data', function(data){
 process.stdout.write(data.toString())
});`
- `request.on('end', function(){})`;
- `request.pipe(response);`

Использование файлов

- `var fs = require('fs');`
- `var x = fs.createReadStream('файл-1');`
- `var y = fs.createWriteStream('файл-2');`

- `y.write('данные');`
- `y.on('drain', function(){})`

- `request.pause();`
- `request.resume();`

Лабораторная работа – 3

- Использование потоков
 - Файл labs\lab-3.js

Выводы

- HTTP-метод POST
- Использование cURL
- Способы обработки запроса
- Чтение из файла и запись в файл
- Загрузка файла на сервер

Модули

Игорь Борисов
<http://igor-borisov.ru>

Темы модуля

- Использование модулей
- Как создать модуль
- NPM - Node Packaged Modules
- Публикация модуля
- Семантика версий
- Основные команды NPM

Как создать модуль?

- module-1.js
 - `module.exports = function(){};`
 - `var x = require('./module-1');`
 - `x();`

- module-2.js
 - `exports.test = function(){};`
 - `var x = require('./module-2');`
 - `x.test();`
 - `require('./module-2').test();`

Использование JSON

- module.json
 - {
 ‘js’: ‘JavaScript - Basic’,
 ‘node’: ‘Node.js’,
 ‘xml’: ‘XML Professional’,
}
- var x = require(‘./module’);
- console.log(x.node);

Лабораторная работа – 4.1

- Использование модулей
 - Файл labs\lab-4-1.js

Где Node.js ищет модули?

- `require('./module');`
- `require('../module');`
- `require('/a/b/c/module');`

- `require('module');`
 - `c:\users\john\app\node_module`
 - `c:\users\john\node_module`
 - `c:\users\node_module`
 - `c:\node_module`

Node Packaged Modules

- <https://npmjs.org/>
 - Репозиторий модулей
 - Стабильность модулей
 - Менеджер зависимостей
 - Публикация модулей
 - Загрузка модулей
- npm `search` имя-модуля
- npm `ls`

Публикация модуля

- package.json
 - {
‘name’: ‘имя-модуля’,
‘description’: ‘описание модуля’,
‘version’: ‘1.2.3’,
‘main’: ‘имя-файла’,
‘dependencies’: {}
}
- npm init

Семантика версий

- Major . Minor . Patch

- 1.2.3

- ~1

- $\geq 1.0.0$ и $< 2.0.0$

- ~1.8

- ≥ 1.8 и $< 2.0.0$

- ~1.8.2

- $\geq 1.8.7$ и $< 1.9.0$

Основные команды NPM

- Публикация
 - npm adduser
 - npm publish
- Инсталляция и использование
 - npm install имя-модуля
 - npm update [имя-модуля]
 - npm remove имя-модуля
- Глобальная инсталляция
 - npm install имя-модуля -g

Лабораторная работа – 4.2

- Загрузка модулей
 - Файл labs\lab-4-2.js

Выводы

- Использование модулей
- Как создать модуль
- NPM - Node Packaged Modules
- Публикация модуля
- Семантика версий
- Основные команды NPM

Express

Игорь Борисов
<http://igor-borisov.ru>

Темы модуля

- Фреймворк Express
- Установка Express
- Базовое использование Express
- Использование шаблонизаторов
- Использование модулей `request` и `url`

Framework Express

- Инсталляция
 - `npm install express`
- Инициализация
 - `var express = require('express');`
 - `var app = express();`
 - `app.listen(8080);`

Использование Express

- ```
app.get('/', function(req, res){
 res.sendFile(__dirname+'/file.html')
});
```
- ```
app.get('/:name', function(req, res){  
  var name = req.params.name;  
  res.write(name);  
  res.end();  
});
```

Шаблонизатор EJS

- `npm install ejs`
 - `<%= var1%>`
 - `<%- var2%>`
- Использование EJS с Express
 - `app.set('views', __dirname);`
 - `app.set('view engine', 'ejs');`
 - `res.render('tpl', {`
`var1: name,`
`var2: 'Some text' }));`

Дополнительные модули

- npm install request
- var request = require('request');
- var url = require('url');
 - var options = {protocol: 'http', host: 'site.ru', pathname: '/', query: {user: 'John'}};
 - var target = url.format(options);
 - request(target, function(err, res, body){
 var out = JSON.parse(body);
});

Лабораторная работа – 5

- Использование фреймворка Express
 - Файл labs\lab-5.js

Выводы

- Фреймворк Express
- Установка Express
- Базовое использование Express
- Использование шаблонизаторов
- Использование модулей `request` и `url`

Socket.IO

Игорь Борисов
<http://igor-borisov.ru>

Темы модуля

- Взаимодействие клиент – сервер
- Обзор модели работы Comet
- Модуль Socket.IO
- Совместное использование модулей Express и Socket.IO
- Основные методы и свойства Socket.IO

Взаимодействие: клиент-сервер

- AJAX
- Comet
 - XMLHttpRequest
 - WebSockets
 - WS.JS
 - Sock.JS
 - Socket.IO

Socket.IO и Express

- Инсталляция
 - npm install socket.io
- Инициализация
 - var socket = require('socket.io');
 - var express = require('express');
 - var app = express();
 - var io = socket.listen(app.listen(8080));

Настройка Socket.IO

■ Серверная часть

- `io.sockets.on('connection', function(socket){});`
- `io.set('log level', 2);`

■ Клиентская часть

- `<script src='/socket.io/socket.io.js'></script>`
- `<script>`
`var socket = io.connect('http://localhost:8080');`
`</script>`

Сообщения Socket.IO

- `socket.emit('message', {name: 'John'});`
- `socket.on('message', function(data){
 console.log(data.name);
});`
- `socket.broadcast.emit(...);`
- `io.sockets.emit(...);`

Дополнительные манипуляции

- `socket.set('key', 'value');`
- `socket.get('key',
 function(error, value){});`
- `socket.disconnect();`

Лабораторная работа – 6

- Чат с использованием Express и Socket.IO
 - Файл labs\lab-6.js

Выводы

- Взаимодействие клиент – сервер
- Обзор модели работы Comet
- Модуль Socket.IO
- Совместное использование модулей Express и Socket.IO
- Основные методы и свойства Socket.IO

Создание веб-приложения

Игорь Борисов
<http://igor-borisov.ru>

Темы модуля

- Основные этапы создания приложения
- Создание каркаса приложения
- Middleware
- Конфигурация приложения
- Логирование
- Использование шаблонизатора
- Использование MongoDB
- Обработка ошибок
- Использование сеансов
- Маршрутизация

Основные этапы

- Создание каркаса приложения
- Конфигурация приложения
- Настройка логирования
- Выбор шаблонизатора
- Выбор хранилища данных
- Выбор способа сохранения состояния
- Маршрутизация
- Обработка ошибок

Создание каркаса приложения

- Глобальная инсталляция
 - `npm install express -g`
- Создание структуры приложения
 - `express -s -e`
- Установка зависимостей в рабочей папке
 - `npm install`
- Запуск приложения
 - `node app.js`

Middleware

- `app.use(function(req, res, next){
 app.send('Послали ответ');
 next();
});`
- `app.use(function(err, req, res, next){
 app.send(404, 'Зашли не туда');
});`
- `express.errorHandler();`

Конфигурация приложения

- Инсталляция модуля nconf
 - npm install nconf
- Инициализация
 - var nconf = require('nconf');
- Установка пути к конфигурационному файлу
 - nconf.argv().env()
 .file({file: 'config.json'});
 });
- Использование
 - nconf.get('port');

Логирование

- Инсталляция модуля `winston`
 - npm `install winston`
- Инициализация
 - `var winston = require('winston');`
- Конфигурация модуля
 - `var log = new (winston.Logger)({
 transports:
 [new (winston.transports.Console)()]});`
- Использование
 - `logger.info('My breakpoint');`

Использование шаблонизатора

- Инсталляция модуля ejs-locals
 - npm install ejs-locals
- Инициализация
 - app.engine('ejs', require('ejs-locals'));
- Возможности шаблонизатора
 - <% layout('main') %>
 - <% block('title', 'Hello'); %>
 - <%=blocks.title %>
 - <%=partial('menu') %>

MongoDB

- <http://www.mongodb.org/downloads>
- Запуск
 - mongod.exe --dbpath c:\data
- Создание коллекции
 - db.test.save({a: 1})
- Создание базы данных
 - use mydb
- Использование
 - db.test.find()
 - db.test.remove({})

Использование MongoDB

- Инсталляция модуля mongoose
 - npm install mongoose
- Инициализация
 - var db = require('mongoose');
- Использование
 - <http://mongoosejs.com/>

Схемы

- `var schema = mongoose.Schema({
 name: String,
 age: Number
});`
- `var X = mongoose.model('X', schema);`
- `var Y = new X ({name: 'a', age: 25});`
- `schema.methods.foo = function(){};`

Сеансы

- `app.use(express.cookieParser());`
- `app.use(express.session({
 secret: 'superSecretKey',
 key: 'NODESESSID',
}));`
- `req.session.name = 'John';`
- `res.send(req.session.name);`

Выводы

- Основные этапы создания приложения
- Создание каркаса приложения
- Middleware
- Конфигурация приложения
- Логирование
- Использование шаблонизатора
- Использование MongoDB
- Обработка ошибок
- Использование сеансов
- Маршрутизация