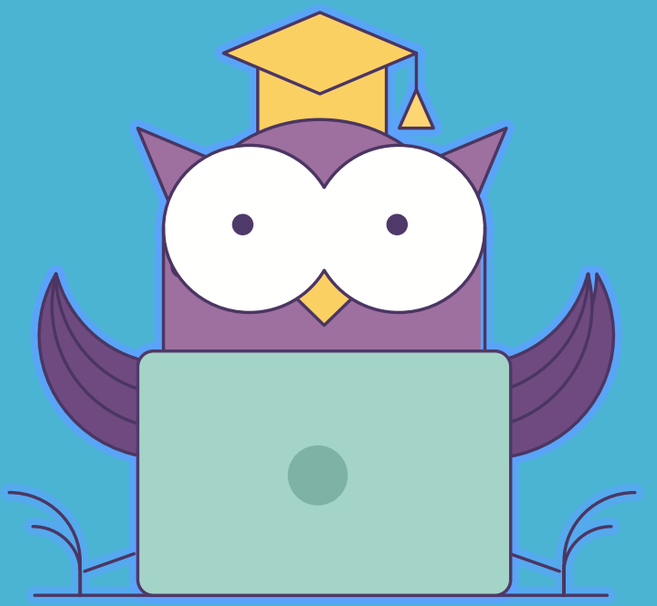
ОНЛАЙН-ОБРАЗОВАНИЕ

Modern JavaScript Frameworks

Введение в Node

Александр Коржиков

Как меня слышно и видно?



> Напишите в чат

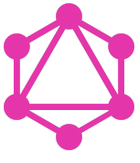
+ если все хорошо

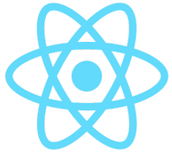
– если есть проблемы со звуком или с видео

Темы предыдущего занятия

- Inheritance
- Promise
- Ajax
- Обзор ES6 features

Modern JavaScript Frameworks





Цели занятия

- Запускать приложения на платформе **Node**
- Работать с пакетным менеджером **NPM**
- Управлять зависимостями и автоматизировать задачи с помощью **package.json**

Содержание

- Node
  - About
  - Пример Web сервера
  - Структура
  - Стандартные модули
  - Примеры Callbacks

- Пакетный менеджер **npm**
  - Возможности **package.json**
  - CLI

Docs

Node core concepts

- https://nodejs.org/api/
- https://nodejs.org/en/docs/guides/

package manager for javascript

- https://docs.npmjs.com/

Асинхронная среда исполнения **JavaScript**, основанная на событийной модели, для создания эффективных сетевых приложений

- Пример web сервера

```
const http = require('http')
const hostname = '127.0.0.1'
const port = 3000
const server = http.createServer((req, res) => {
  res.statusCode = 200
  res.setHeader('Content-Type', 'text/plain')
  res.end('Hello World\n')
})

server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`)
})
```

- Какие особенности Вы бы отметили из этого примера?

Особенности

- Исполнение **JavaScript** файлов с помощью комманды node
  - REPL
- **CommonJS** формат модулей для загрузки зависимостей
  - **ES Modules**
- Стандартная библиотека модулей
- API основанное на асинхронном паттерне Callbacks
- **ES2015** синтаксис
- Демо

История

- *Server side* **JavaScript** *Platform* - Ryand Dahl, 2009
- Asyncronous (non-blocking) I/O
- Chromium **JavaScript** Engine -> V8
- **Node.js** Foundation

Структура

- Библиотека написана на **C++** и **JavaScript**
- **V8** - платформа исполнения **JavaScript** от Google (Chromium, Chrome, Opera, Brave, Yandex Browser)
- Event Loop - асинхронный событийный цикл с **libuv**
- Модули для работы с операционной системой

Patterns

- Reactor
- Module
- Observer
- Callback

Задача

Установить node и npm https://nodejs.org/en/download/

```
node --version
npm --version
```

Создать и запустить **server.js**

```
echo "console.log(process.argv)" > server.js
```

```
node server hello world
```

Вопрос

А Вы знаете какие модули включены в стандартный дистрибутив **Node**?

Стандартные модули

- **Main**
  - fs
  - timers
  - streams
- **Utilities**
  - path
  - util
  - zlib
  - crypto
- **Processes**
  - child_process
  - cluster
  - worker_threads
- **Protocols**
  - http(s)
  - net
  - dns
- **System**
  - os
  - v8
  - async_hooks
  - perf_hooks
  - trace_events

Utilities

```
const querystring = require('querystring')

querystring.parse(`q=shell+ls+regex&
rlz=1C5CHFA_enNL772ED772&
oq=shell+ls+by+reg&
aqs=chrome.1.63i27j0a4.4221e0b9&
sourceid=chrome&ie=UTF-8
`)

{
  "q": "shell ls regex",
  "rlz": "1C5CHFA_enNL772ED772",
  "oq": "shell ls by reg",
  "aqs": "chrome.1.63i27j0a4.4221e0b9",
  "sourceid": "chrome",
  "ie": "UTF-8"
}
```

- querystring

Utilities

```
const path = require('path')

path.join('/Users', 'alex', 'Sites')
// =>
'/Users/alex/Sites'

path.parse('/home/user/dir/file.txt')

{
  root: '/',
  dir: '/home/user/dir',
  base: 'file.txt',
  ext: '.txt',
  name: 'file'
}
```

- querystring
- path

Utilities

```
const url = require('url')
url.parse('https://github.com/korzio')

// Url
{
  protocol: 'https:',
  slashes: true,
  auth: null,
  host: 'github.com',
  port: null,
  hostname: 'github.com',
  hash: null,
  search: null,
  query: null,
  pathname: '/korzio',
  path: '/korzio',
  href: 'https://github.com/korzio'
}
```

- querystring
- path
- url

Utilities

```
const util = require('util')
const assert = require('assert')

util.types.isDate(value)
// or deprecated
util.isDate(value)
// or inherit
util.inherits(MyStream, EventEmitter)

assert.equal(1, 1)
assert.deepEqual({ a: 1 }, { a: 1 })
```

- querystring
- path
- url
- util
- assert

Callbacks

Функция, переданная в качестве аргумента коду, который предполагает исполнить его в какой-то момент времени. Исполнение может быть **синхронным** или **асинхронным**.

```
const server = http.createServer((req, res) => {
  res.statusCode = 200
  res.setHeader('Content-Type', 'text/plain')
  res.end('Hello World\n')
})

server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`)
})
```

В **Node** по умолчанию **callback** выполняется с "ошибкой" и результатом **асинхронно**

```
fs.readFile('/etc/passwd', (err, data) => {
  if (err) throw err
  console.log(data)
})
```

Callback Types

single only error

```
fs.access('/etc/passwd', fs.constants.R_OK, (err) => {
  console.log(err ? 'no access!' : 'read')
})
```

Node поддерживает Promises

```
util.promisify()
fs.promises.*
```

Тип возвращаемого значения

```
http.request('https://example.com', (error, response, body) => {
  ...
})
```

Question

- Какие примеры **callback** паттерна в **JavaScript** Вы знаете?
- Вы знакомы с **promisify** функционалом?
- Есть ли разница между **Promise** и **Callback** паттернами?

Anti-Pattern - http://callbackhell.com/

```
fs.readdir(source, function (err, files) {
  if (err) {
    console.log('Error finding files: ' + err)
  } else {
    files.forEach(function (filename, fileIndex) {
      console.log(filename)
      gm(source + filename).size(function (err, values) {
        if (err) {
          console.log('Error identifying file size: ' + err)
        } else {
          console.log(filename + ' : ' + values)
          aspect = (values.width / values.height)
          widths.forEach(function (width, widthIndex) {
            height = Math.round(width / aspect)
            console.log('resizing ' + filename + 'to ' + height + 'x' + height)
            this.resize(width, height).write(dest + 'w' + width + '_' + filename, function(err) {
              if (err) console.log('Error writing file: ' + err)
            })
          }.bind(this))
        }
      })
    })
  }
})
```

Globals

- **global** объект - аналог **window**
- объекты **JavaScript**
- *timeouts* почти как в браузере
- **console**
- **process** репрезентация текущего процесса

Node Q&A

NPM

Package Manager for JavaScript

- CI

```
npm --version
```

- Registry

```
npm i
```

- Website https://npmjs.com/

В числах

Всего зарегистрировано 952 468 пакетов

Скачиваний

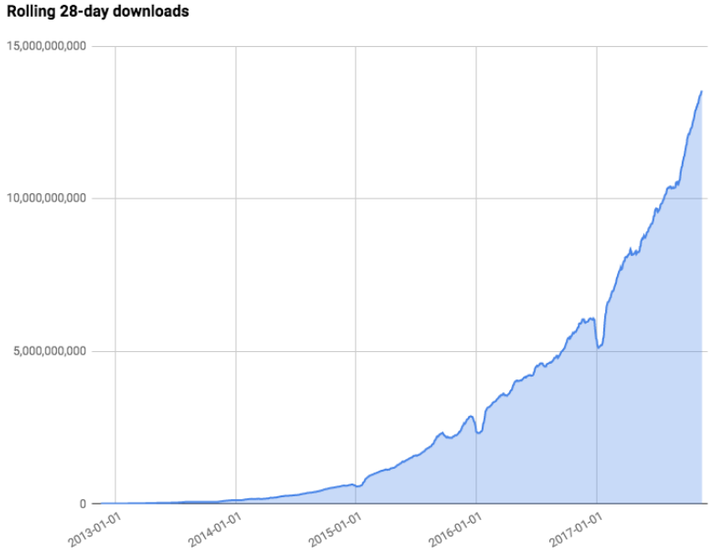
- за неделю - 11 076 971 328
- за месяц - 47 763 330 586
- **react** - 5 388 912

Node 2017

25 000 000

Звезд во Вселенной

1 000 000 000 000 000 000 000



package.json

```
{
  "name": "package",
  "description": "",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "repository": {
    "type": "git",
    "url": "https://github.com/.../package.git"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "bugs": {
    "url": "https://github.com/.../issues"
  },
  "homepage": "https://github.com/.../package"
}
```

Semantic Versioning

- Спецификация https://semver.org/
- Калькулятор https://semver.npmjs.com/

```
npm install semver
```

Status	Stage	Version
First release	New product	1.0.0
Bug fix	patch	1.0.1
New Feature	minor	1.1.1
Breaking change	major	2.0.0

- Какие альтернативные нотации версионирования Вы знаете?

CLI

- Основные
  - init
  - install
  - start
  - test
  - run
- Дополнительные
  - ci
  - npx

Самостоятельная работа

Создать новый **npm** пакет

```
mkdir ... && cd ...
npm init # read & answer
```

Сохранить зависимость

```
npm install --save-dev ...
# copy & paste package.json > chat
```

Запустить скрипты

```
echo "" > server.js # create file
npm start
npm test # npm run test
```

Зависимости

- **dependencies**
  - **devDependencies** разработчика
  - **peerDependencies** плагины
  - bundleDependencies дистрибутив (сокращает время установки зависимостей)
  - **optionalDependencies** необязательные
- **--global**
- **node_modules**

```
"dependencies": {
  "commander": "^2.7.1",
  "lodash.get": "^4.0.0",
  "lodash.isequal": "^4.0.0",
  "validator": "^9.0.0"
},
"devDependencies": {
  "coveralls": "^3.0.0",
  "grunt": "^1.0.1",
  "grunt-browserify": "^5.2.0",
  "grunt-cli": "^1.2.0",
  "grunt-contrib-copy": "^1.0.0",
  "grunt-jscs": "^3.0.1",
  "grunt-lineending": "^1.0.0",
  "jasmine-node": "^1.14.5",
  "jasmine-reporters": "^2.2.1",
  "remapify": "^2.1.0"
}
```

node_modules

- Locks
  - package-lock.json
  - npm-shrinkwrap.json
- Альтернативы
  - yarn
  - bower
  - turbo
- Proposals
  - tink

```
# A{B,C}, B{C}, C{D}
A
+-- B
+-- C
+-- D

# A{B,C}, B{C,D@1}, C{D@2}
A
+-- B
+-- C
    +-- D@2
+-- D@1
```

NPM Q&A

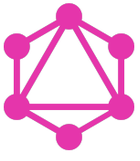
На занятии

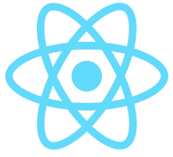
- Запускать приложения на платформе Node
- Использовать NPM для создания, добавления зависимостей и использования скриптов

Modern JavaScript Frameworks











Самостоятельная работа

- Создать локальный веб сервер **server**, отвечающий на запросы каждые 100ms
- Создать скрипт **request**, принимающий на вход
  - количество запросов **N**
  - тип запросов - параллельный или последовательный

Скрипт **request** должен отправлять **N** последовательных или параллельных **HTTP** запросов к локальному серверу **server**

Спасибо за внимание!

Пожалуйста, пройдите опрос

в личном кабинете

- Все ли темы были понятны? (да - нет)
- Легкий материал или нет? (1 просто - 10 сложно)