Оглавление

[Старт 1](#_Toc92726267)

[NPM, установка пакетов 1](#_Toc92726268)

[Экспорт из файла 2](#_Toc92726269)

[Базовые модули Node 2](#_Toc92726270)

[Path 2](#_Toc92726271)

[FS 2](#_Toc92726272)

[OS 2](#_Toc92726273)

[Events 3](#_Toc92726274)

[Создание сервера на NodeJS 4](#_Toc92726275)

# Старт

Установка с сайта, проверка версии и работоспособности: node -v.

“node” в терминале позволяет писать js код в терминале.

Для вызова файла в консоли: «node + ${имя файла}»

Расширение файла можно не прописывать (автоматически добавляет js)

# NPM, установка пакетов

npm -v

Инициализация npm: npm init, создает json файл

Npm I/install – установка пакетов, установленные пакеты отображаются в json, “dependencies”.

С установкой пакета могут быть установлены другие от которых он зависит, посмотреть зависимости пакета можно через: «название пакета» - package.json – dependences.

При передаче проекта, удаляется папка node\_modules, новый разработчик прописывает npm I, после чего будут установлены все необходимые пакеты (потому что они прописаны в package.json).

Require – команда которая подключит необходимый пакет в файле. Например: установили пакет chalk (npm i chalk), в коде нужно прописать: (const chalk = require(‘chalk’)); ИЛИ! Прописать то, что требуется в документации конкретного модуля. JS поймет, что мы обращаемся к пакету. Также помогло прописать "type": "module" в json доке (смотри index.js).

Обновление: require – легаси!

# Экспорт из файла

// abc.mjs

const abc = () => {

console.log('hello')

}

export default abc;

// index.js

import abc from './abc.mjs';

abc()

# Базовые модули Node

# Path

Подробности в коде, path.js

path.basename

path.dirname

path.extname

path.parse

# FS

const fs = require('fs'); // позволяет работать с сист файлами

const path = require('path');

fs.mkdir(path.join(\_\_dirname, 'test'), (err) => {

    if (err) {

        throw err;

    }

    console.log('Папка создана');

});

Создает папку в текущей директории с названием test, если будет ошибка, то не создаст)

+ ряд других функций, расписывать не буду, просто перечислю (создание файла с определенным содержимым – .writefile, добавить инфу в файл - .append, чтение файла - .readfile (дает инфу буфером) если указать кодировку то выдаст в нормальном виде).

# OS

Для работы с ос.

|  |
| --- |
| const os =  require('os') |
|  |  |
|  | console.log('Операционная система: ', os.platform()) |
|  |  |
|  | console.log('Архитектура процессора: ', os.arch()) |
|  |  |
|  | console.log('Инфа по процессорам: ', os.cpus()) |
|  |  |
|  | console.log('Свободная память: ', os.freemem()) |
|  |  |
|  | console.log('Всего памяти: ', os.totalmem()) |
|  |  |
|  | console.log('Домашняя директория: ', os.homedir()) |
|  |  |
|  | console.log('Влючен: ', os.uptime()) |

# Events

|  |
| --- |
| const EventEmitter = require('events') |
|  |  |
|  | // const emitter = new EventEmitter() |
|  | // |
|  | // emitter.on('anything', data => { |
|  | // console.log('ON: anything', data) |
|  | // }) |
|  | // |
|  | // emitter.emit('anything', {a: 1}) |
|  | // emitter.emit('anything', {b: 2}) |
|  | // |
|  | // setTimeout(() => { |
|  | // emitter.emit('anything', {c: 3}) |
|  | // }, 1500) |
|  |  |
|  | class Dispatcher extends EventEmitter { |
|  | subscribe(eventName, cb) { |
|  | console.log('[Subscribe...]') |
|  | this.on(eventName, cb) |
|  | } |
|  |  |
|  | dispatch(eventName, data) { |
|  | console.log('[Dispatching...]') |
|  | this.emit(eventName, data) |
|  | } |
|  | } |
|  |  |
|  | const dis = new Dispatcher() |
|  |  |
|  | dis.subscribe('aa', data => { |
|  | console.log('ON: aa', data) |
|  | }) |
|  |  |
|  | dis.dispatch('aa', {aa: 22}) |

# Создание сервера на NodeJS

const http = require('http');

const server = http.createServer( (req, res) => {

    res.end('hello node');

})

server.listen(3000, () => { //порт, колбэк

    console.log('server has been started.')

})

Теперь при переходе на <http://localhost:3000/>, там будет код. При изменении кода (можно отправлять теги), нужно перезапускать сервер для обновления инфы.

Cntrl + C и заново запустить.

Добавим скрипты в json

    "start": "node index.js",

    "dev": "nodemon index.js" // перезапустит серв

Для запуска скрипта необходимо прописать npm run \*название скрипта\*

Можно задавать тип инфы на сервере (текст, html и тд).

Код для работы со страницами

const http = require('http')

    const fs = require('fs')

    const path = require('path')

    const server = http.createServer((req, res) => {

  let filePath = path.join(\_\_dirname, 'public', req.url === '/' ? 'index.html' : req.url)

      const ext = path.extname(filePath)

      let contentType = 'text/html'

      switch (ext) {

        case '.css':

          contentType = 'text/css'

          break

        case '.js':

          contentType = 'text/javascript'

          break

        default:

          contentType = 'text/html'

      }

      if (!ext) {

        filePath += '.html'

      }

      fs.readFile(filePath, (err, content) => {

        if (err) {

          fs.readFile(path.join(\_\_dirname, 'public', 'error.html'), (err, data) => {

            if (err) {

              res.writeHead(500)

              res.end('Error')

            } else {

              res.writeHead(200, {

                'Content-Type': 'text/html'

              })

              res.end(data)

            }

          })

        } else {

          res.writeHead(200, {

            'Content-Type': contentType

          })

          res.end(content)

        }

      })

    })

    const PORT = process.env.PORT || 3000

    server.listen(PORT, () => {

      console.log(`Server has been started on ${PORT}...`)

    })

Модуль express позволяет избежать всего этого и сэкономить время.