1. Общая информация

- Node – среда выполнения кода JS построенная на движке V8.

- Подходит для обработки потоковых данных в реальном времени, не требующих сложных вычислений. Не подходит для обработки видео или машинного обучения.

- Нет глобального обьекта window, есть обьект global и есть доступ к фаиловой системе. Не может взаимодействовать с DOM

- В отличие от браузеров, где код нужно адаптировать под конкретный браузер – код на серверах выполняется в одном и том же окружении (на последних версиях) – соответственно можно использовать весь функционал.

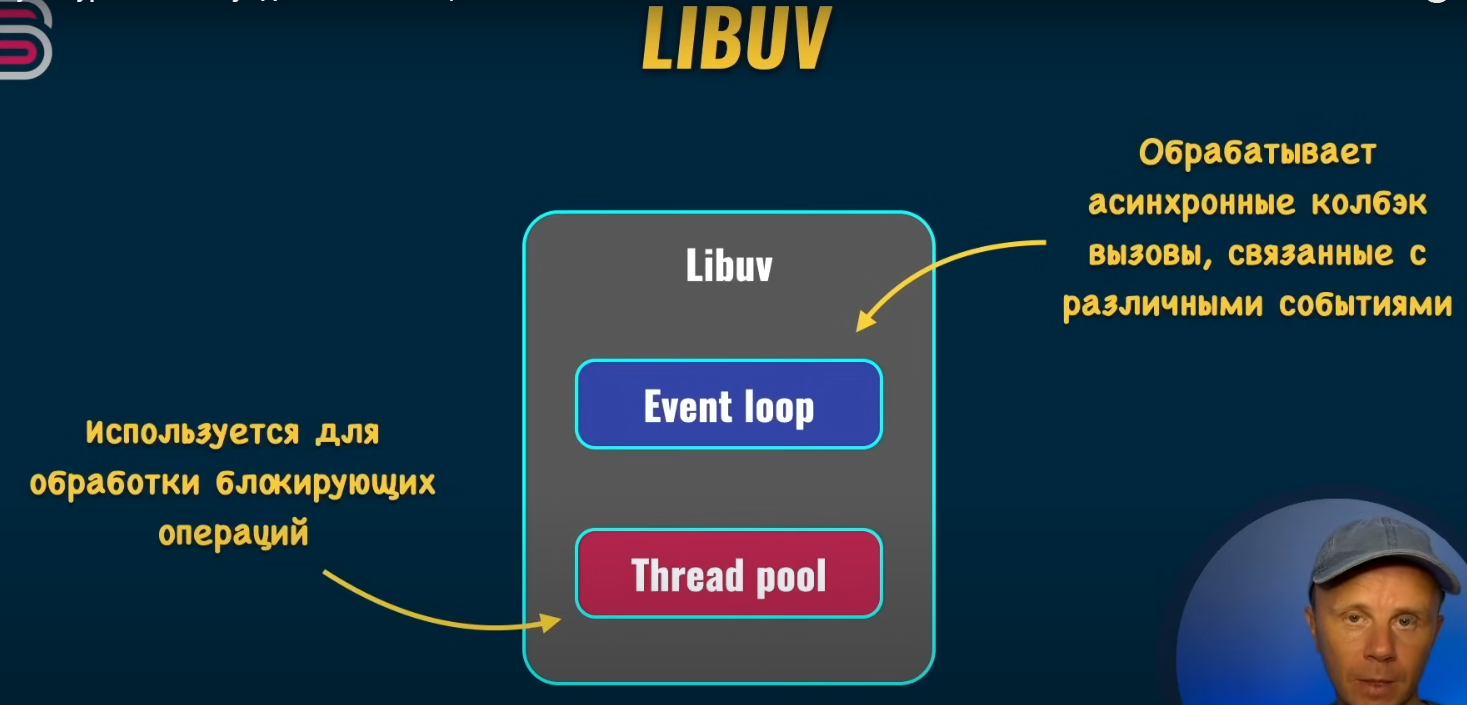
2. Архитектура Node.JS



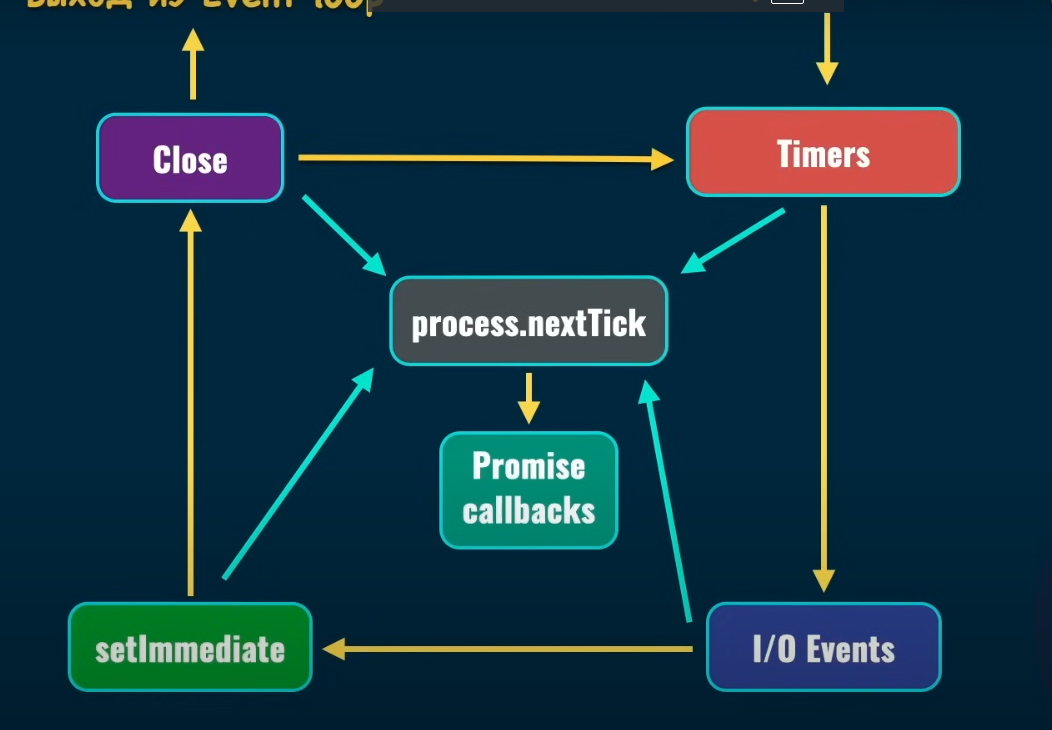
Node построена на движке V8. Так же в ней (среде) есть и библиотека LibUV – написана на C++ . И благодаря компиллеру C++ биндинг код JS компилируется в C++ и выполняется в LibUV.

Сам по себе NodeJS однопотомный, но благодаря LibUV ей можно передавать некоторые задачи, которые могли бы блокировать или задерживать основной поток. В LibUV может быть до 1024 потоков (по умолчанию 4 – можно устанавливать и больше). Потоки виртуальные – не привязаны к количеству процессоров. Так же сторонние библиотеки могут работать в многопоточном режиме. И с 11 версии ноды появились worker threads, в которые так же можно передавать некоторые задачи для выполнения в отдельном потоке.

LibUV работает по принципу Event Loop (цикл событий)



3. Event loop



Суть та же что и на фронте.

Приоритетность:

1. process.nextTick – создать микротаску – очень высокий приоритет (они влазят при первой возможности)

2. микротаски Promise

3. макротаски таймеров

4. макротаски input/output (запросы к dns)

5. макротаски setImmediate (похоже на таймаут с нулевой задержкой)

6. макротаски close (например работа с фаиловой системой)

4. Модули в node.js:



В node.js используются модули CommonJS (но можно но нежелательно так же использовать ECMAScript модули)

Импорт:

const obj = {};

const func = () => {console.log(222)}

module.exports.obj = obj;

module.exports.func = func;

Экспорт:

const {obj, func} = require(‘path’);

5. Глобальные обьекты и основные модули

**- process**

process.env

process.argv – аргументы из скрипта из package.json

process.exit() – завершить процесс

**- path**

const path = require(‘path);

\_\_dirname – путь к папке

path.join(\_\_dirname, ‘..’)

path.parse(url) – преобразовать путь в обьект

так же можно через new URL(myURL)

**- fs – работа с фаиловой системой**

const path = require(‘fs);

const path = require(‘path);

fs.mkdirSync(path.resolve(\_\_dirname, ‘myFolder’), callback) – создать папку

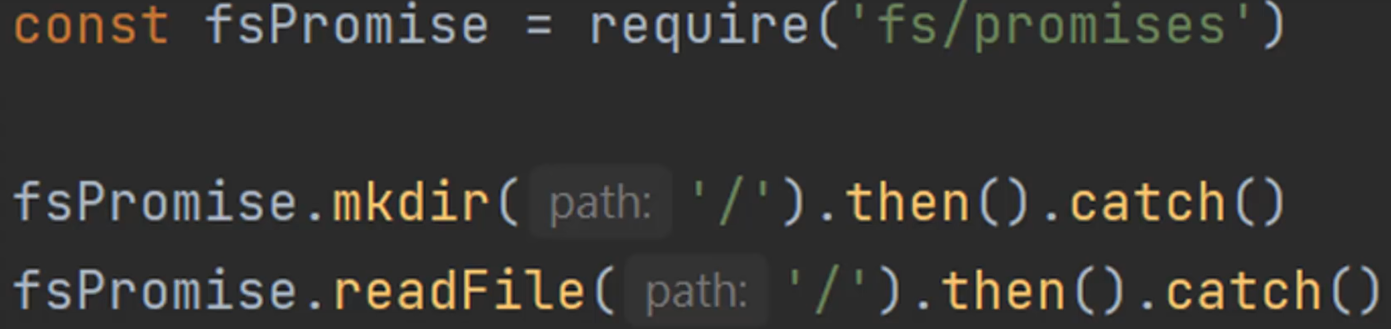
fs.mkdir(path.resolve(\_\_dirname, ‘myFolder’), callback) – создать папку

fs.rmdir(path.resolve(\_\_dirname, ‘myFolder’), callback) – удалить папку

fs.writeFile(path.resolve(\_\_dirname, ‘myFolder’), myText, callback) – создать фаил с текстом

fs.appendFile(path.resolve(\_\_dirname, ‘myFolder’), myText, callback) – добавить текст в фаил

Для работы в промисном режиме



**- os – работа с операционной системой**

const os = require(‘os);

os.platform() – платформа

os.arch() – архитектура

os.cpus().length – количество ядер

**- cluster** – модуль для возможности ноды работать в многопоточном режиме (**создавать воркеры**)

const cluster = require(‘cluster’);

cluster.isMaster – проверить, главный ли процесс

cluster.fork() – запустить новый процесс на новом ядре

cluster.on(‘exit’, callback) – слушатели кластера

**- events – работа с собитиями**

const Emitter = require(‘events’);

const emitter = new Emitter();

emitter.on(‘message’, callback) – описание события

emitter.emit(‘message’, text) – генерация события

emitter.once(‘message’, callback) – описание события (с возможностью генерации только единожды)

**- http – работа с сервером и его настройками**

const http = require(‘http);

const server = http.createServer(…)

Пример сервера с 2 эндпоинтами



6. .env

Возможны 2 варианта испльзования – через фаил .env и через скрипты в package.json

- .env

Устанавливаем на проект пакет dotenv. Создаем фаил в корне, прописываем все переменные. При использовании

const dotenv = require(‘dotenv’);

dotenv.config();

console.log(process.env.PORT);

- скрипты в package.json

Устанавливаем на проект пакет cross-env.

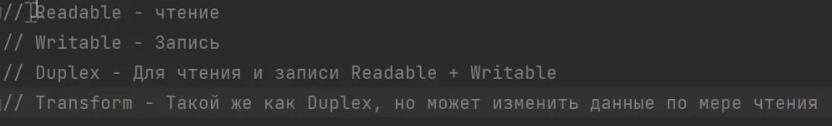
В скрипте добавляем

"dev": "cross-env PORT=5000 nodemon index.js",

В фаиле

console.log(process.env.PORT);

7. Стримы



Предназначены для работы с фаиловой системой. В отличие от fs не дожидается прочтения фаила целиком, а читает или отправляет фаил по кусочкам(чанкам).

const fs = require(‘fs);

const path = require(‘path);

const stream = fs.createReadStream(pathToFile);

stream.on(‘data’, callback);

stream.on(‘end, callback);

stream.on(‘error, callback);