Беляев Сергей

Web-технологии

Применение Node.js.
Основы Deno

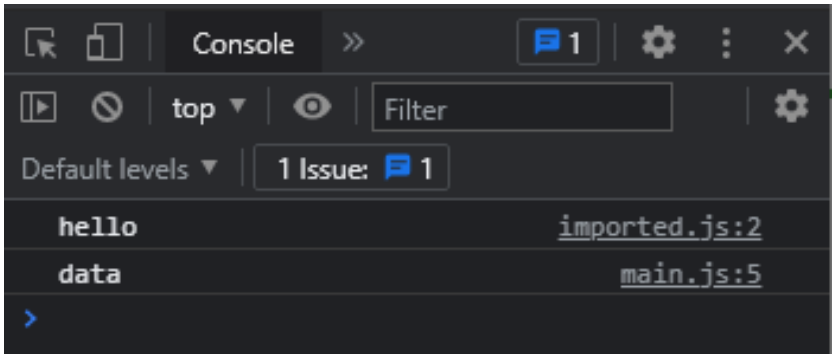
Содержание

- Работа с модулями
- Использование npm и yarn, готовые модули
- Событийная модель в Node.js
- Создание простейшего сервера на Node.js
  - Эхо-сервер, параметры запроса, заголовки
- Журналирование (логгирование)
- Внешние сервера для журналирования (логгирования)
  - Rollbar – попытка отловить ошибки раньше пользователей
  - Sentry – прослеживание JavaScript
- Работа с файлами и потоками
- «Асинхронный try-catch»
- nvm
- Deno
  - Установка, настройка среды
  - Разрешения, примеры

http://learn.javascript.ru/screencast/nodejs
https://nodejs.org/en/docs/guides/

Модули в браузере


```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
</head>
<body>
    <script src="main.js" type="module"></script>
</body>
</html>
```



main.js

```
import hello from "./imported.js"
import {data} from "./imported.js"

hello()
console.log(data)
```

imported.js

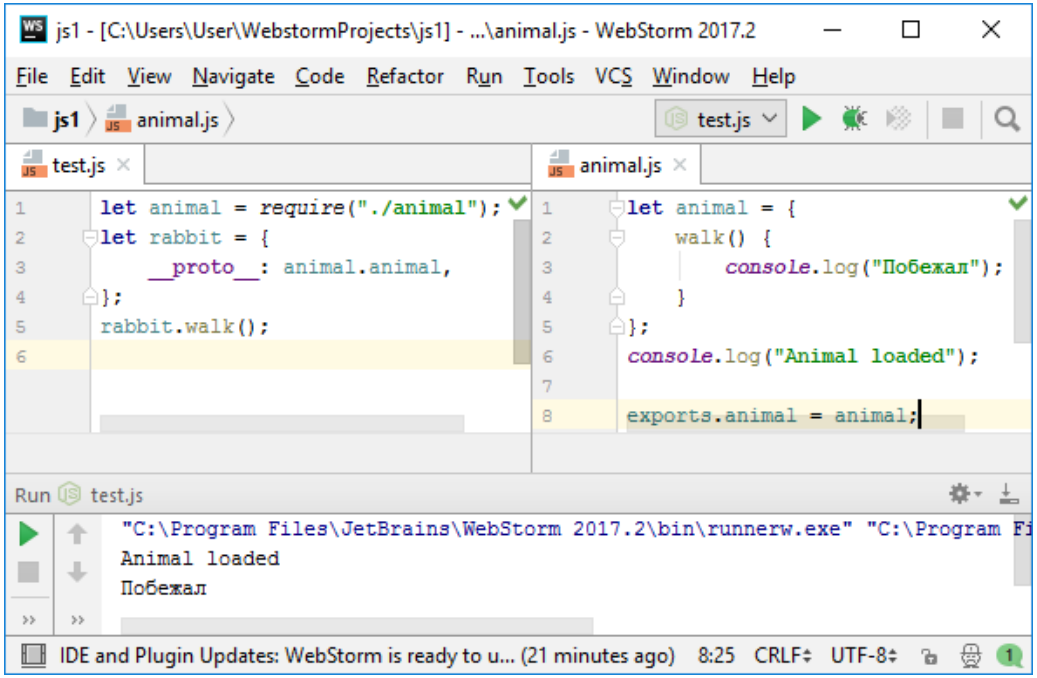
```
function hello() {
    console.log("hello")
}
let data = "data"
export {data}
export default hello
```

.js, .mjs... ,... **as**... *... **then**...

Деление на модули


```
let animal = require("./animal");
let rabbit = {
    __proto__: animal.animal,
};
rabbit.walk();
```

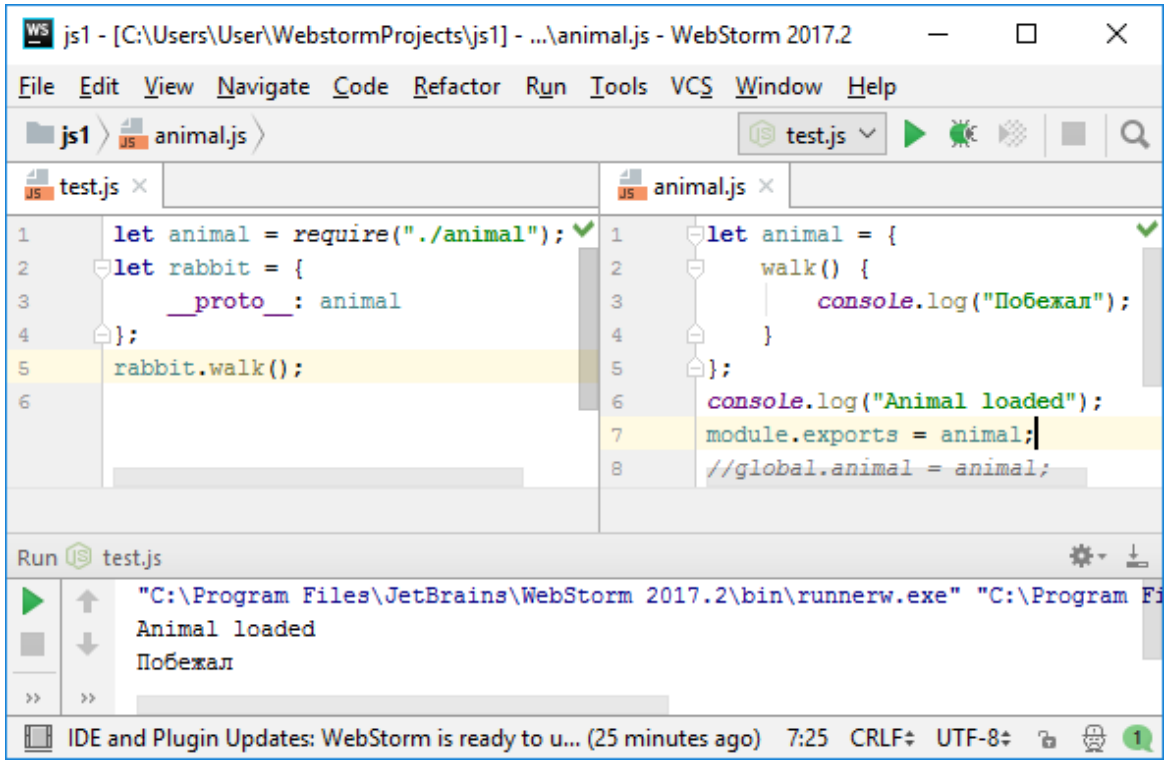
```
let animal = {
    walk() {
        console.log("Побежал");
    }
};
console.log("Animal loaded");

exports.animal = animal;
```



Глобальные переменные


```
let animal = require("./animal");
let rabbit = {
    __proto__: animal
};
rabbit.walk();
```

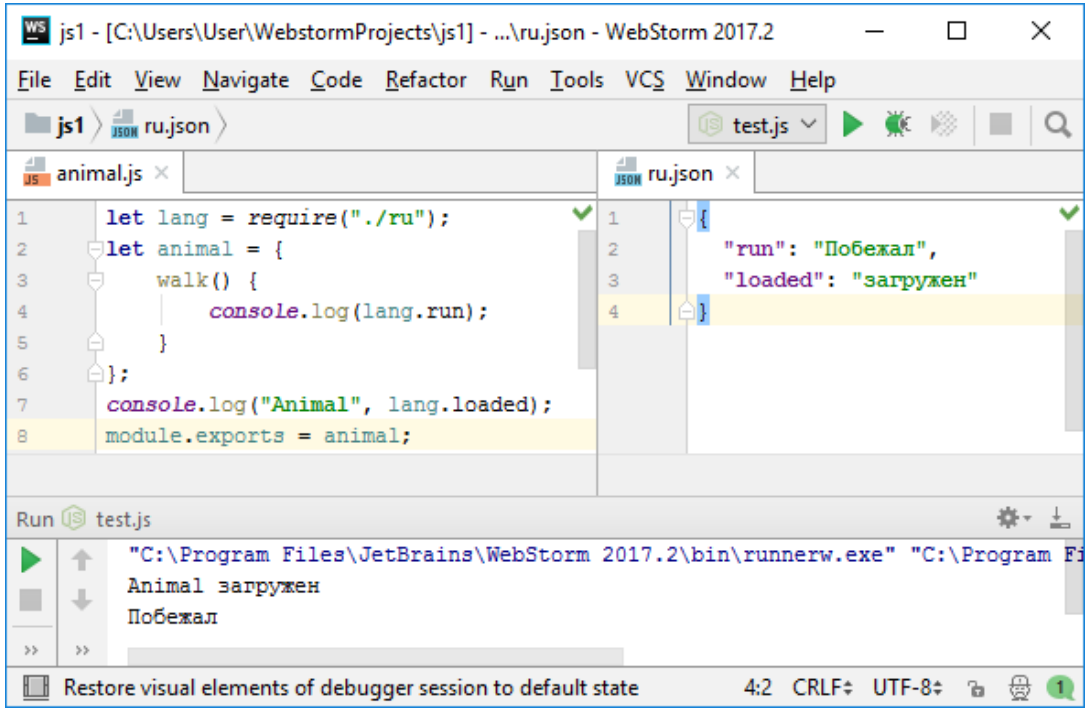
```
let animal = {
    walk() {
        console.log("Побежал");
    }
};
console.log("Animal loaded");
module.exports = animal;
//global.animal = animal;
```



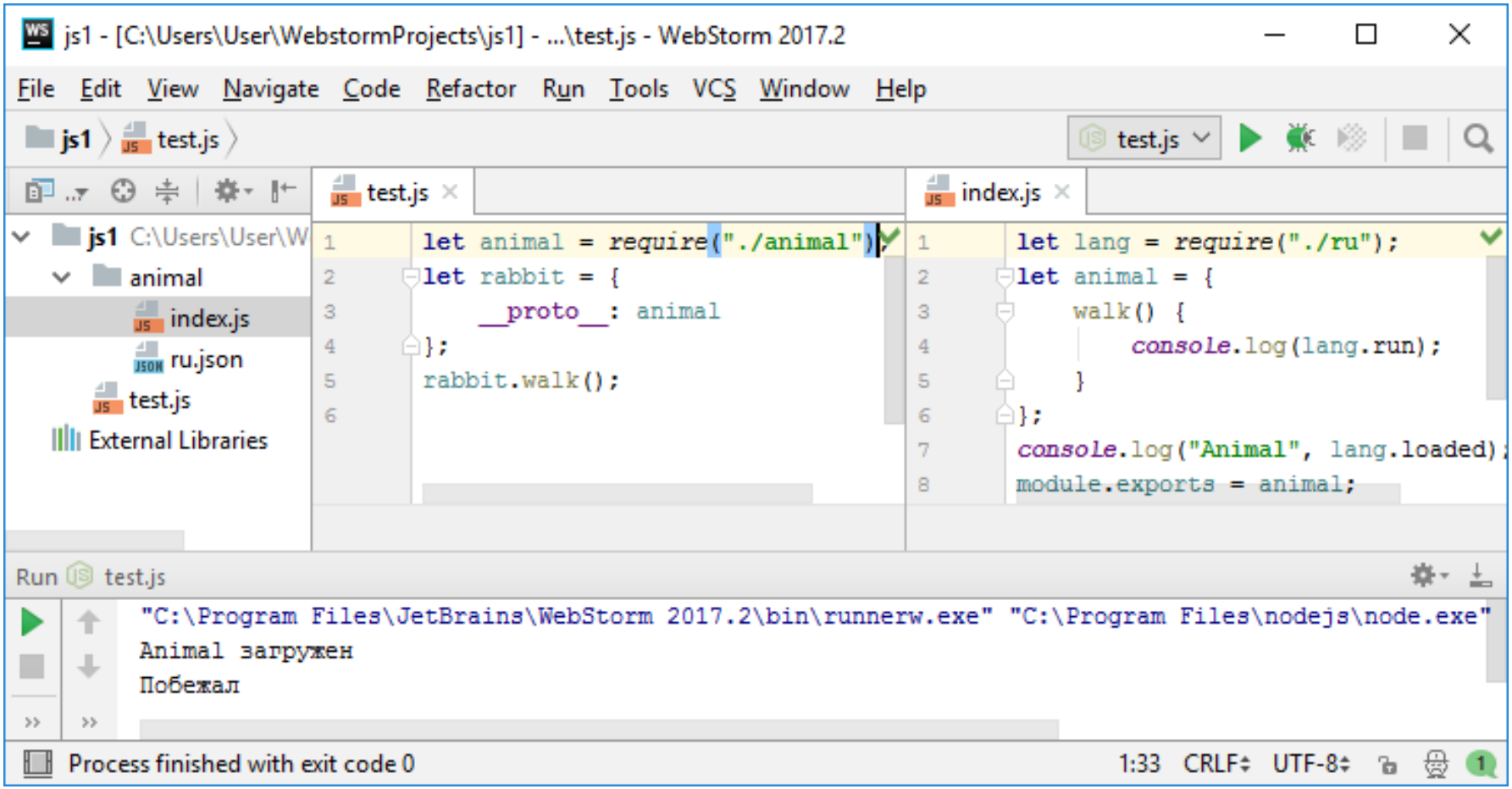
Использование JSON


```
let lang = require("./ru");
let animal = {
    walk() {
        console.log(lang.run);
    }
};
console.log("Animal", lang.loaded);
module.exports = animal;
```

```
{
  "run": "Побежал",
  "loaded": "загружен"
}
```

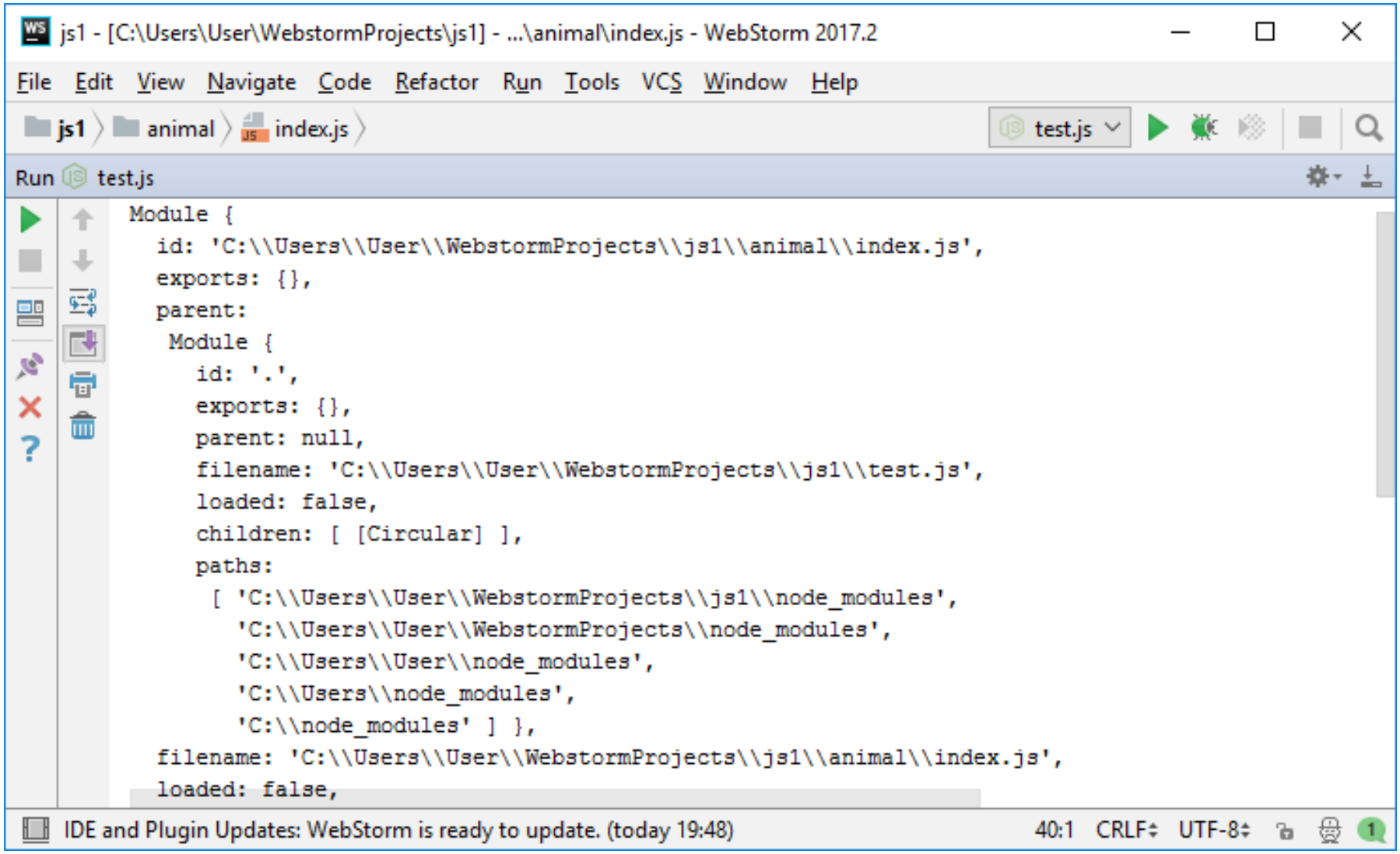


Использование папок для модулей



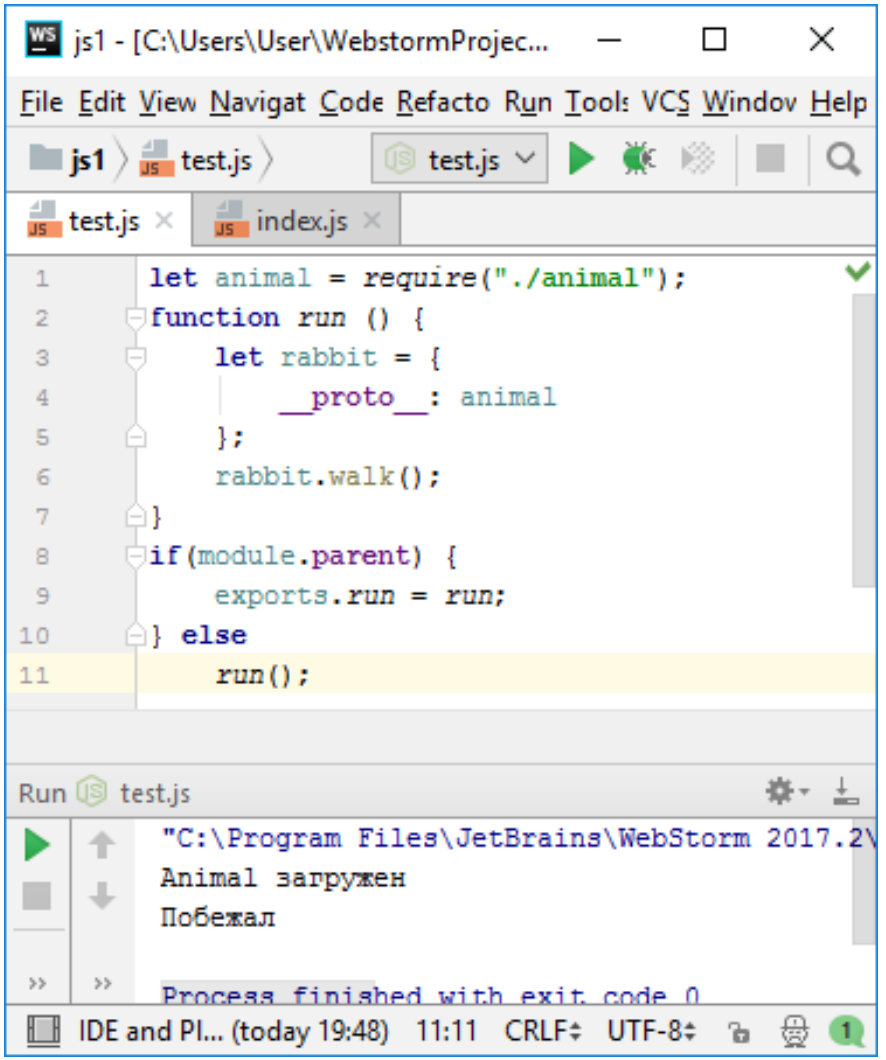
Скрипт должен называться **index.js**

Объект module

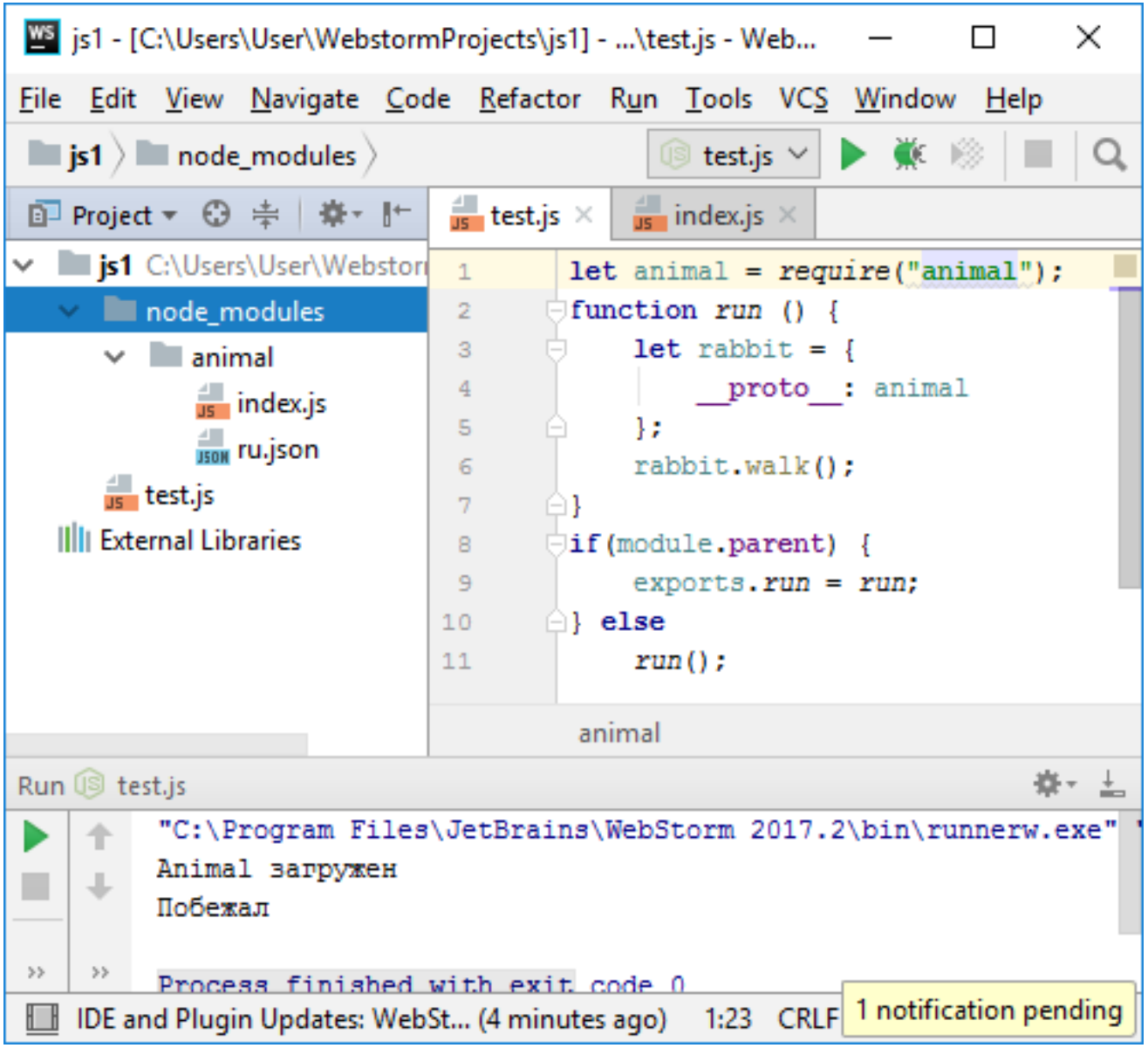


module.parent


```
let animal = require("./animal");
function run () {
    let rabbit = {
        __proto__: animal
    };
    rabbit.walk();
}
if(module.parent) {
    exports.run = run;
} else
    run();
```

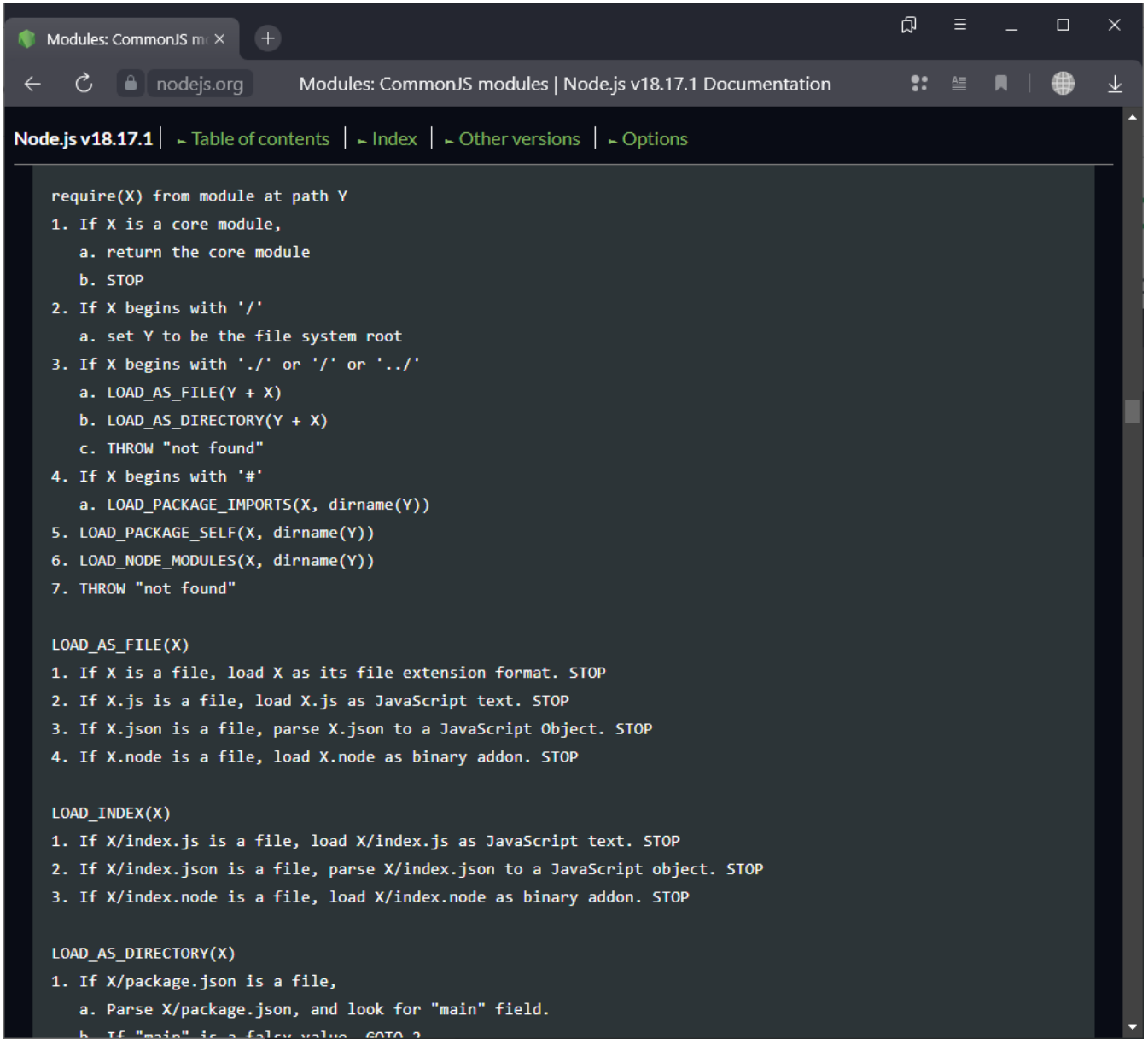


node_modules



Последовательность загрузки модулей

https://nodejs.org/dist/latest-v18.x/docs/api/modules.html#all-together



CommonJS vs. ES6 modules

CommonJS

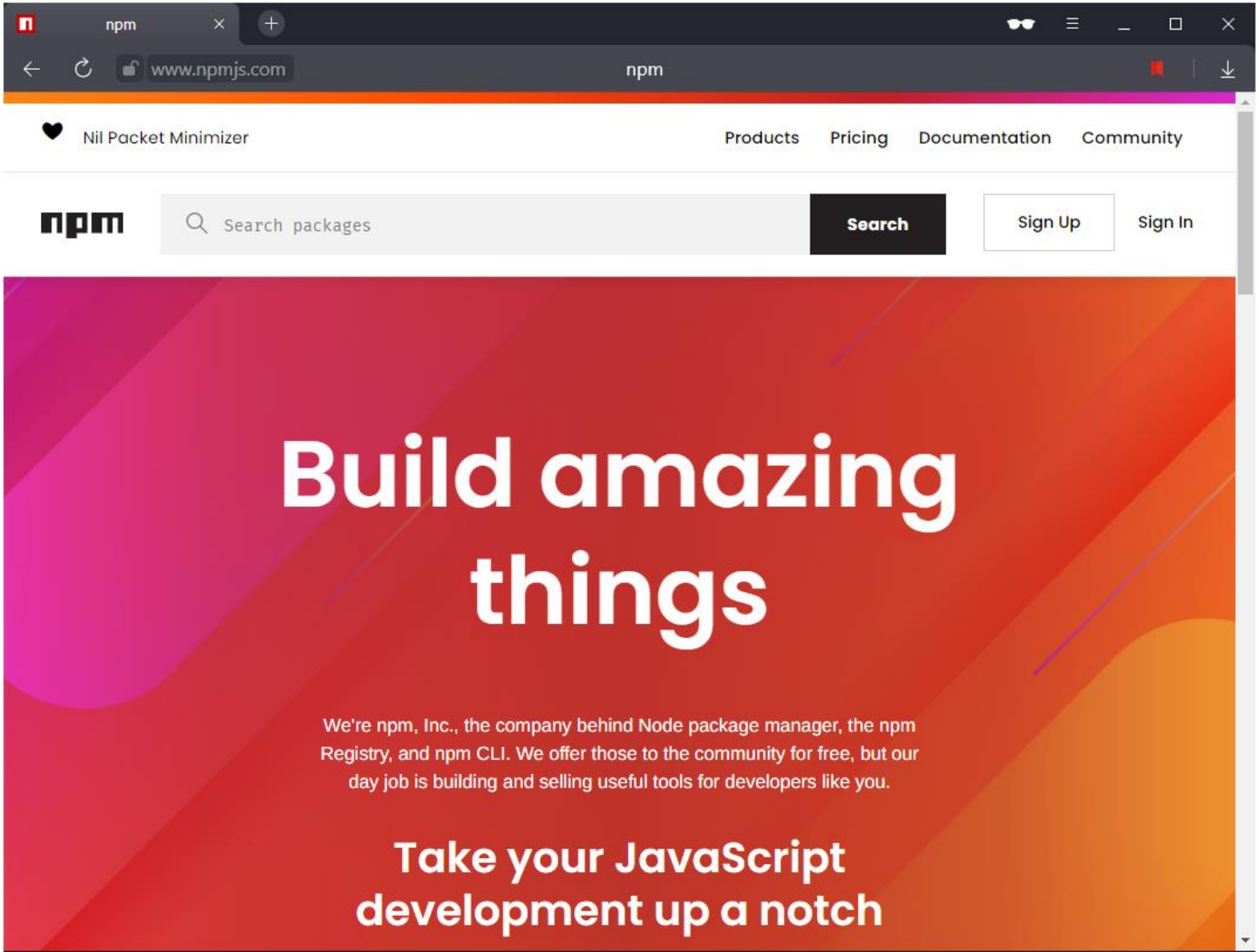
- Экспорт (один)
  - module.exports, exports
- Импорт
  - require
- Загрузка
  - runtime
  - не «поднимает» импорт
- Экспортирует «значение»
- Переменные верхнего уровня
  - arguments, require, module, exports, __filename, __dirname
- Не поддерживает синтаксис ES6 модулей

ES6

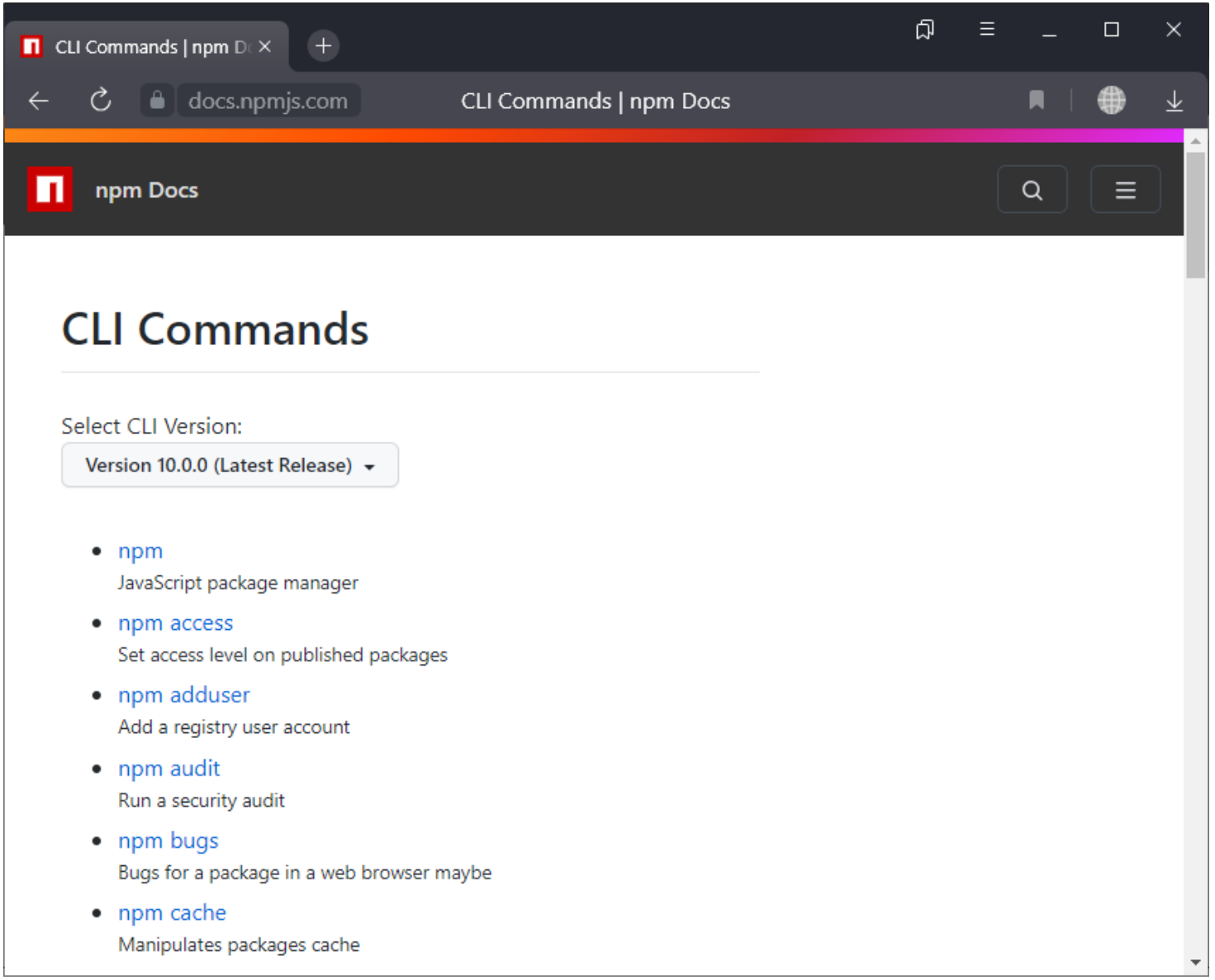
- Экспорт (множественный)
  - export
- Импорт
  - import
- Загрузка
  - статическая компиляция
  - «поднимает» импорт
- Экспортирует «ссылку»
- Нет переменных верхнего уровня
- Поддерживает синтаксис CommonJS

npm

https://www.npmjs.com/



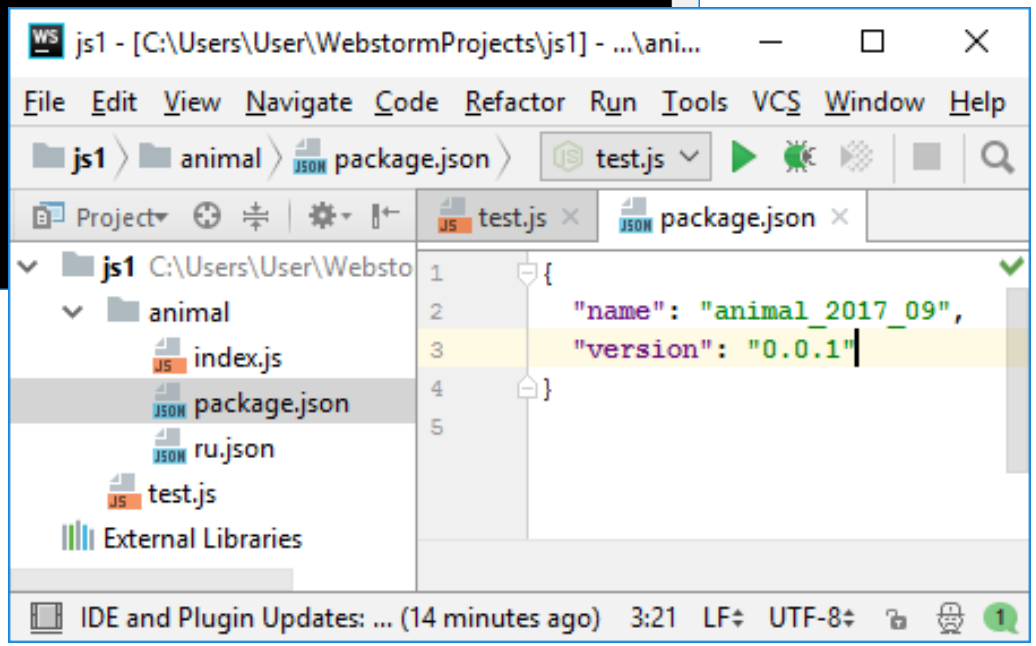
Команды CLI npm



https://docs.npmjs.com/cli/v10/commands

npm init – инициализация


```
npm
C:\Users\User\WebstormProjects\js1\animal>npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help json` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg> --save` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
name: (animal) animal_2017_09
version: (1.0.0) 0.0.1
description: Без описания
entry point: (index.js)
test command:
git repository:
keywords:
author:
license: (ISC)
About to write to C:\Users\User\WebstormProjects\js1\animal\package.json:

{
  "name": "animal_2017_09",
  "version": "0.0.1",
  "description": "Без описания",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC"
}
```



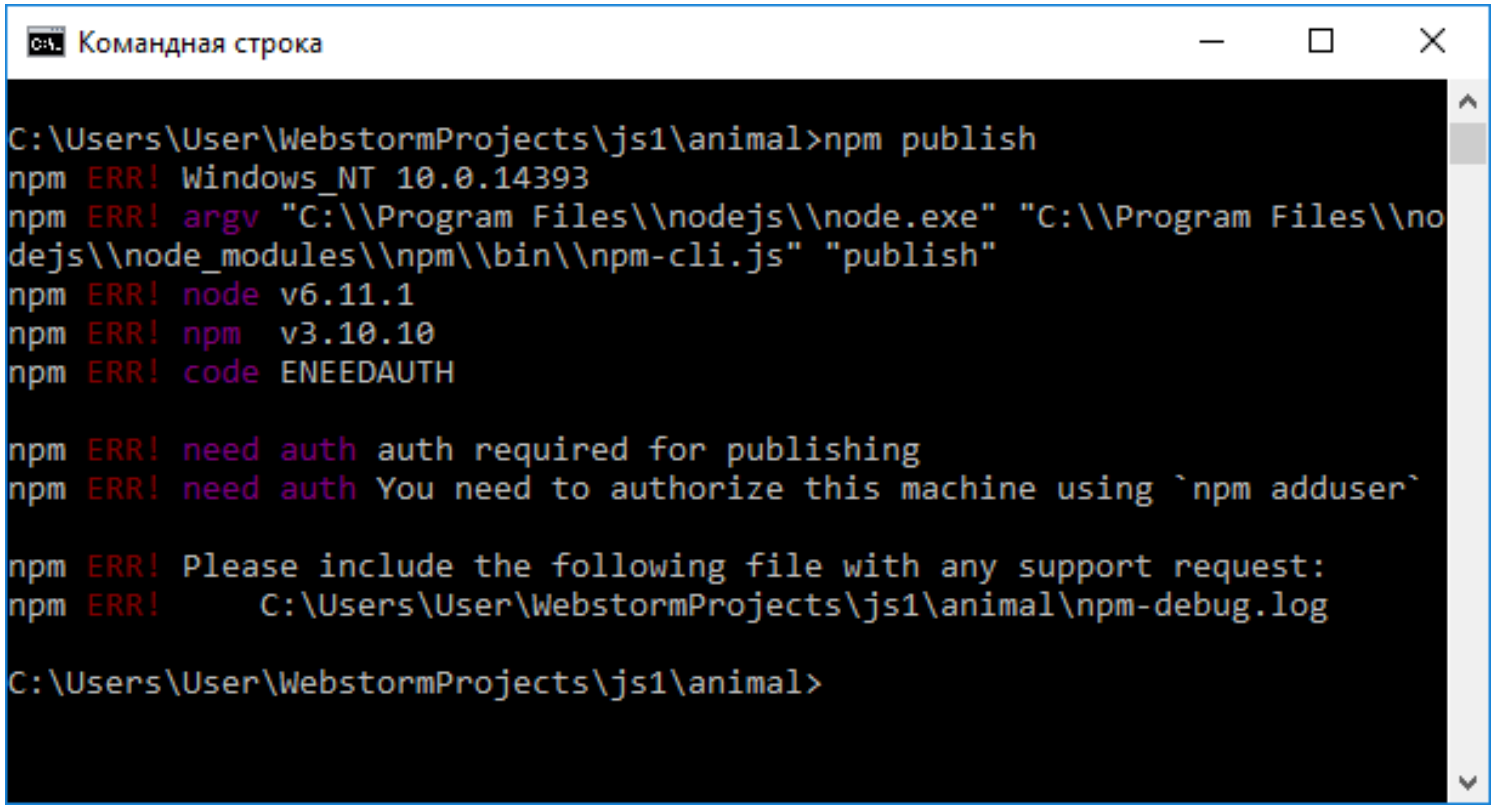
package.json

Пример

```
{
  "name": "my-project",
  "version": "0.0.1",
  "description": "This is a project.",
  "author": "Someone <someone@example.com>",
  "contributors": [{
    "name": "Someone Else",
    "email": "else@example.com"
  }],
  "keywords": ["improves", "searching"]
}
```

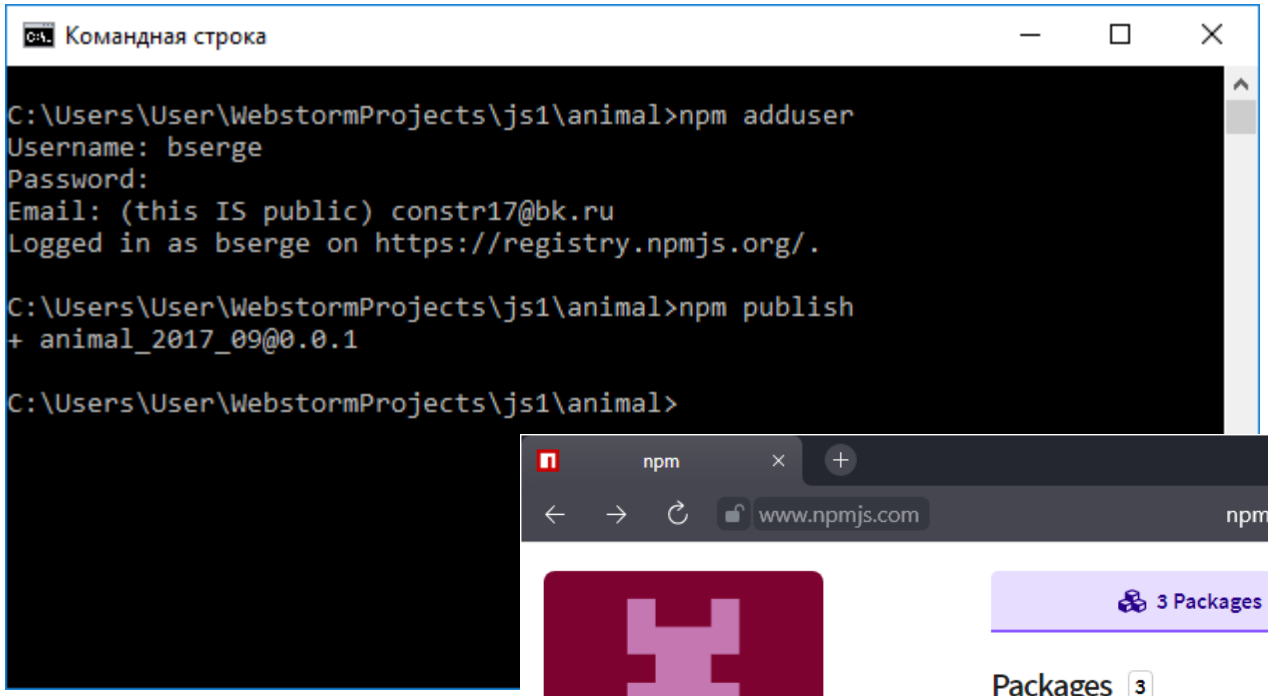
- Создание: **npm init**
- **name** – имя пакета
- **version** – версия пакета
- **description** – описание
- **author** – автор
- **script** – содержит дополнительные команды для **npm**
  - **start – node index.js**
- **main** – главная точка входа в проекте
- **devDependencies** – зависимости на время разработки
- **dependencies** – зависимости пакета
- **license** – лицензия
- **keywords** – ключевые слова, описывающие пакет

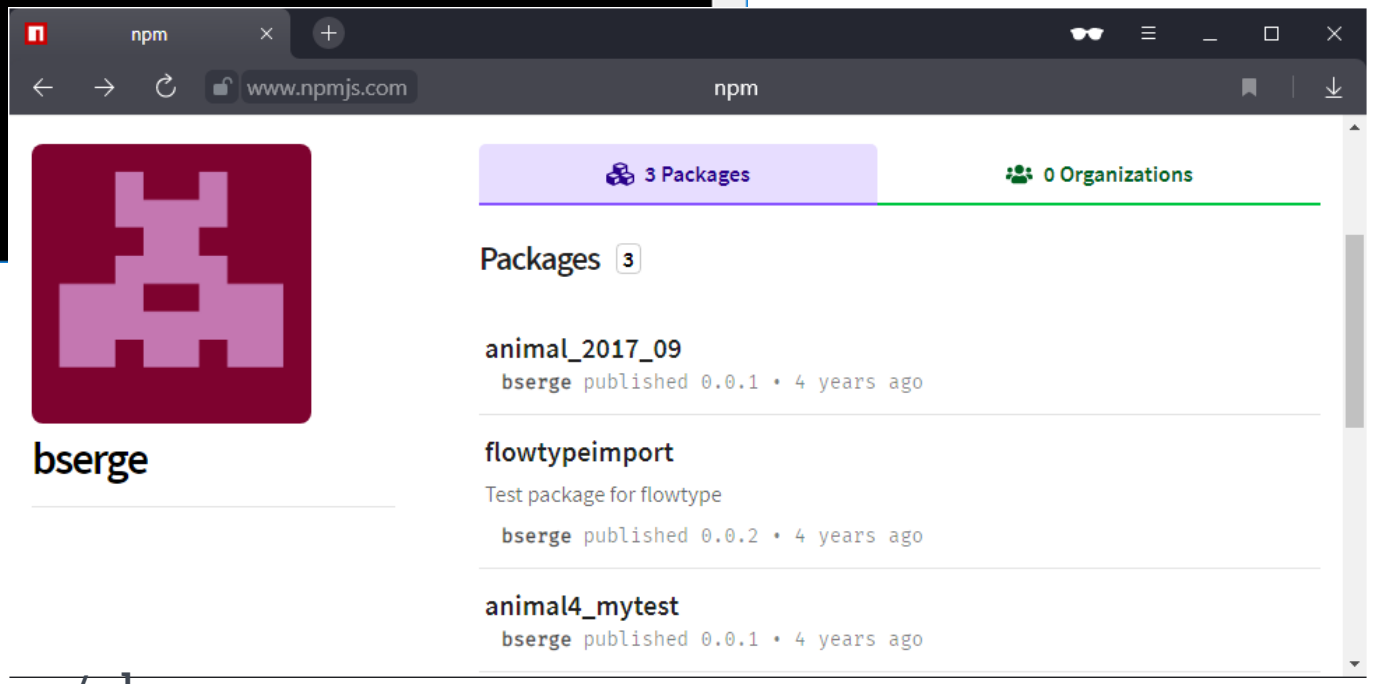
npm publish – публикация



npm help

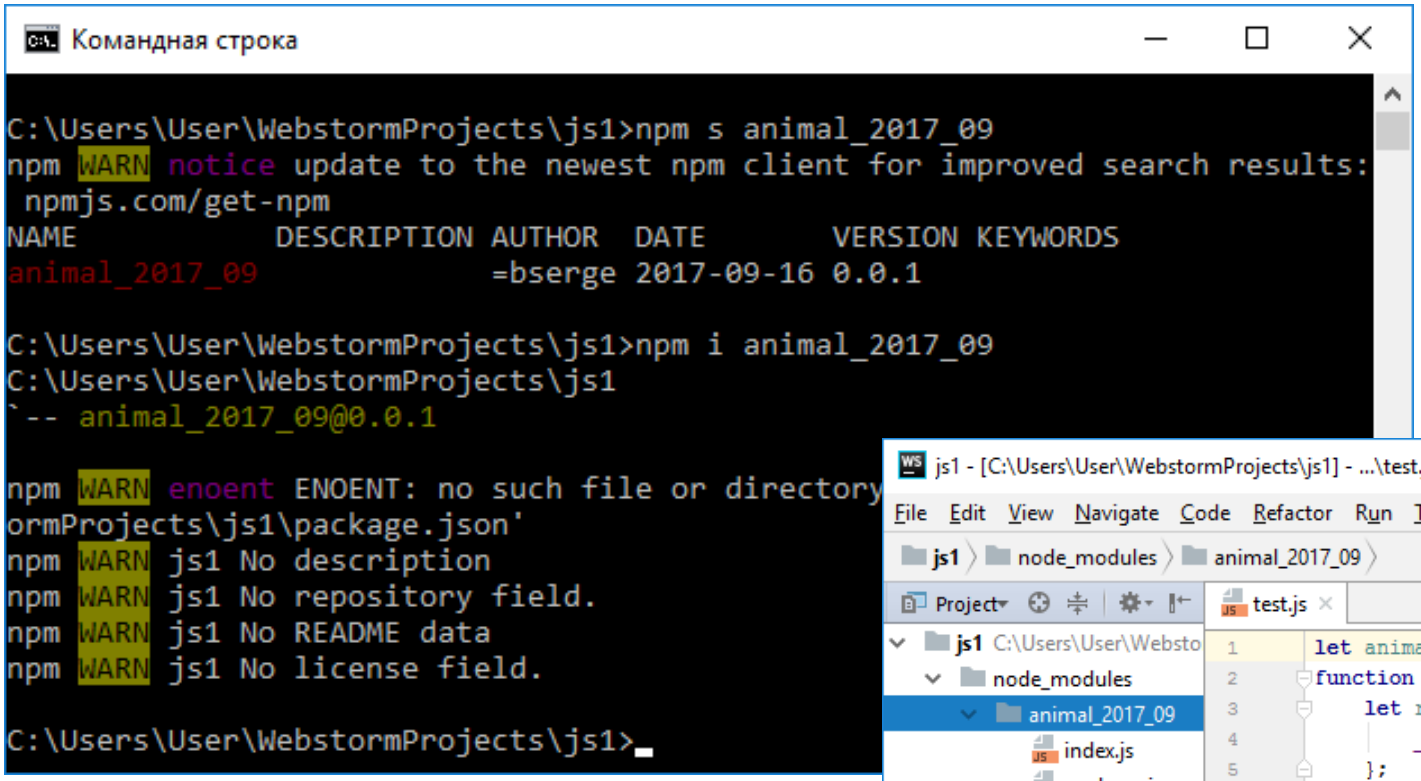
npm adduser – добавление пользователя
npm publish – публикация

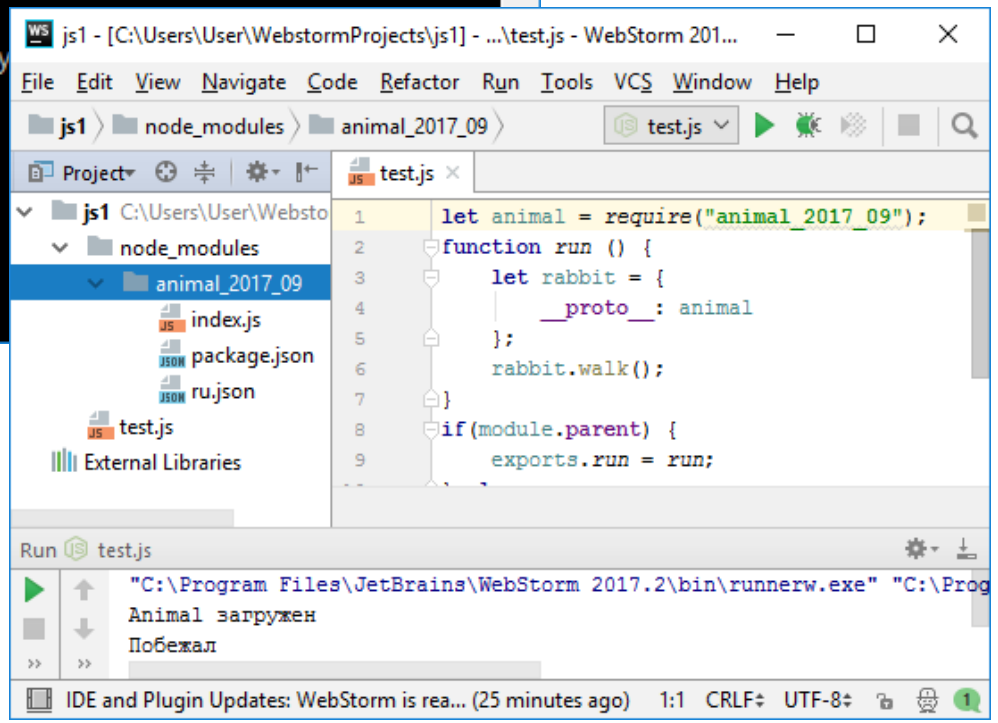




https://www.npmjs.com/~bserge

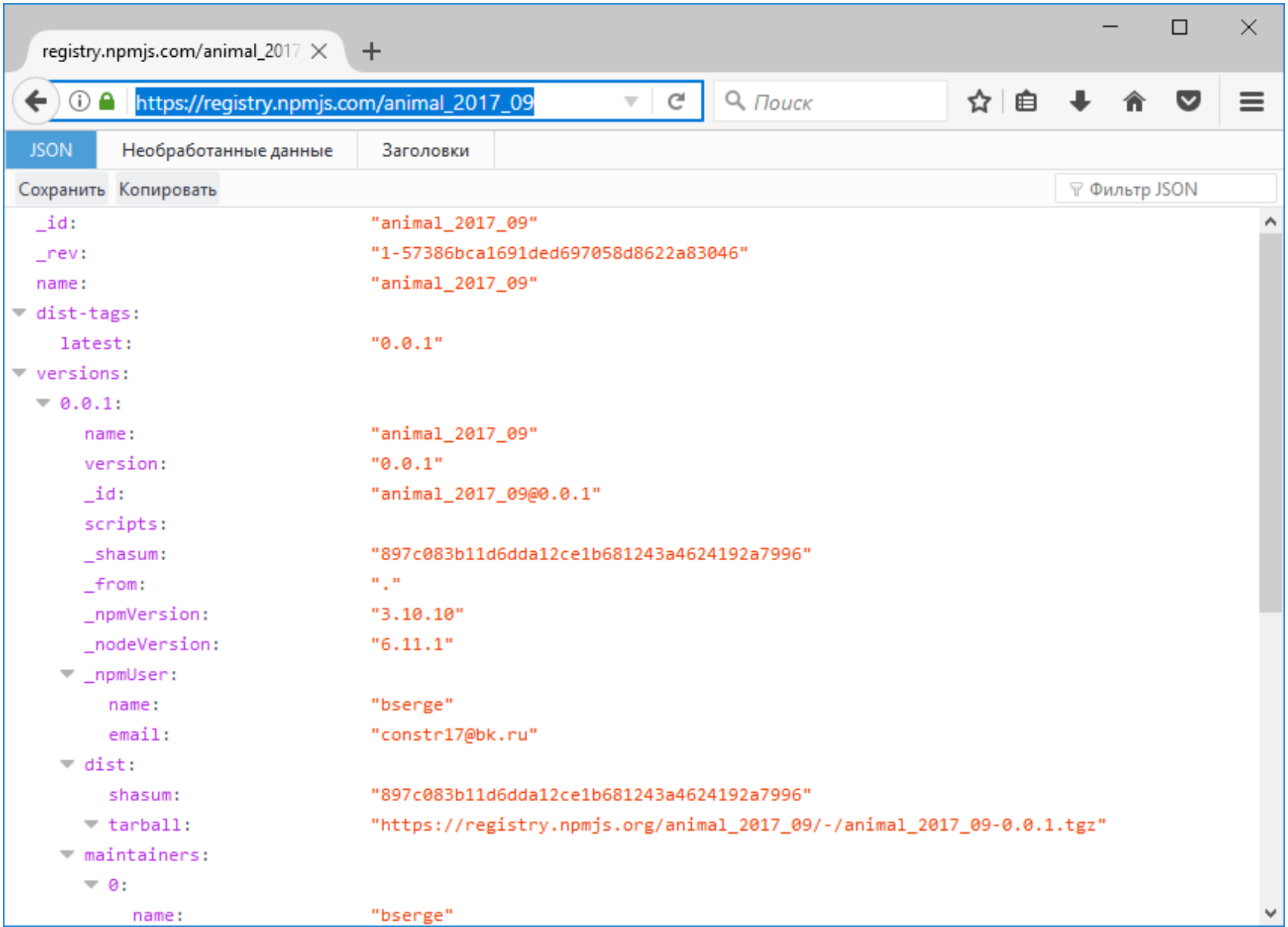
npm s animal_2017_09 – поиск
npm i animal_2017_09 – установка





Глобальная установка
npm i -g animal_2017_09

https://registry.npmjs.com/animal_2017_09



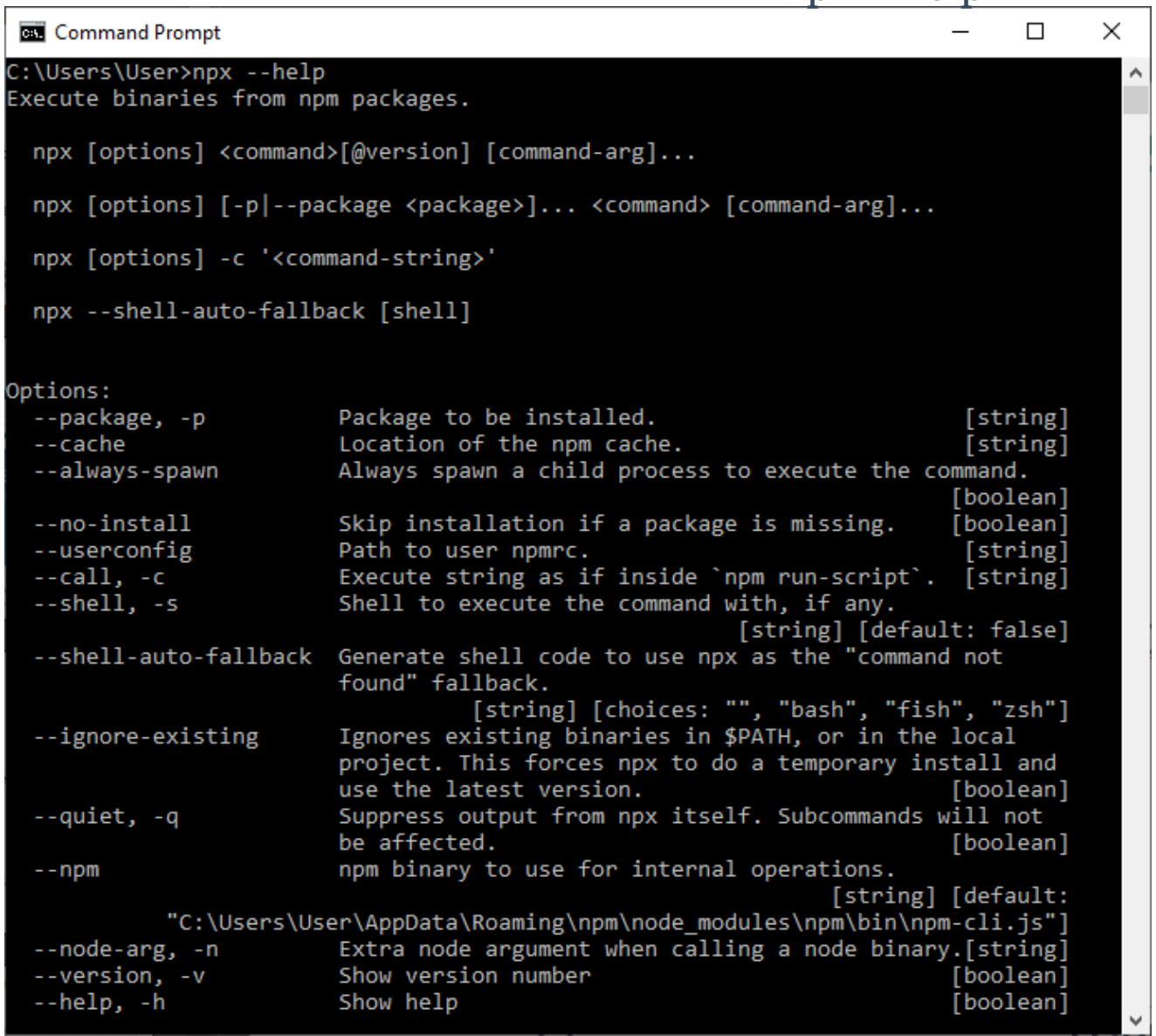
npm – использование

- Инициация проекта
  - **npm init**
- Добавление модуля
  - **npm install [package]**
  - Глобальный модуль
    - **npm install -g [package]**
- Добавление зависимости в **package.json**
  - **npm install --save [package]**
- Удаление
  - **npm uninstall [package]**
- Загрузить всё, что указано в **package.json**
  - **npm install**
- Публикация пакета
  - **npm adduser**
  - **npm login**
  - **npm publish**

npx

TL;DR

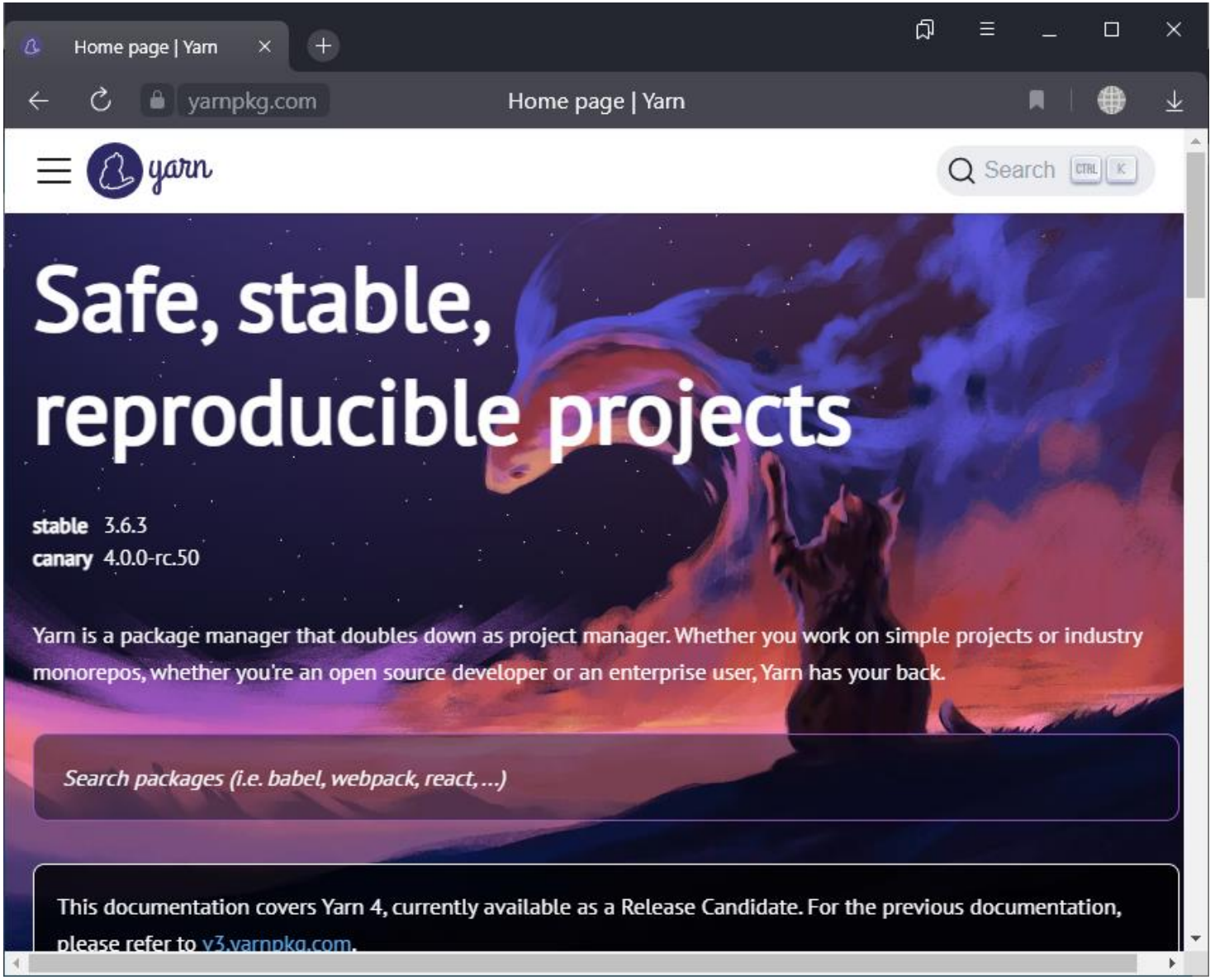
npx module_name
npx -p package_name module_name
npx --help



- Запуск локально установленных пакетов
- Выполнение пакета, который не установлен
- Тестирование разных версий пакетов

yarn – аналог npm

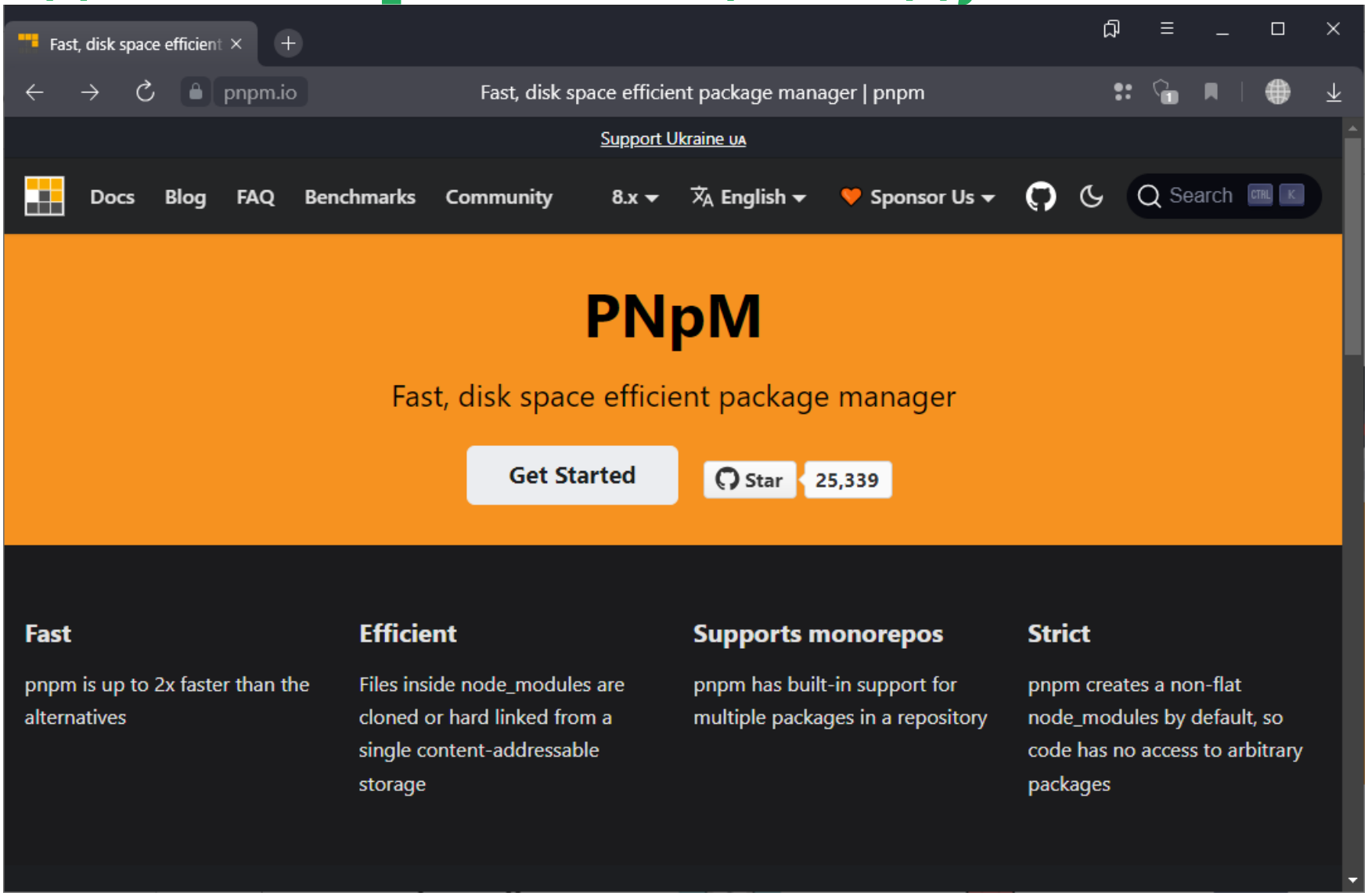
https://yarnpkg.com/



yarn – использование

- Инициация проекта
  - **yarn init**
- Добавление модуля
  - **yarn add [package]**
  - **yarn add [package]@[version]**
  - **yarn add [package]@[tag]**
- Добавление зависимости в **devDependencies**
  - **yarn add [package] --dev**
- Обновление зависимости
  - **yarn upgrade [package]**
  - **yarn upgrade [package]@[version]**
  - **yarn upgrade [package]@[tag]**
- Удаление
  - **yarn remove [package]**
- Загрузить всё, что указано в **package.json**
  - **yarn**
  - **yarn install**

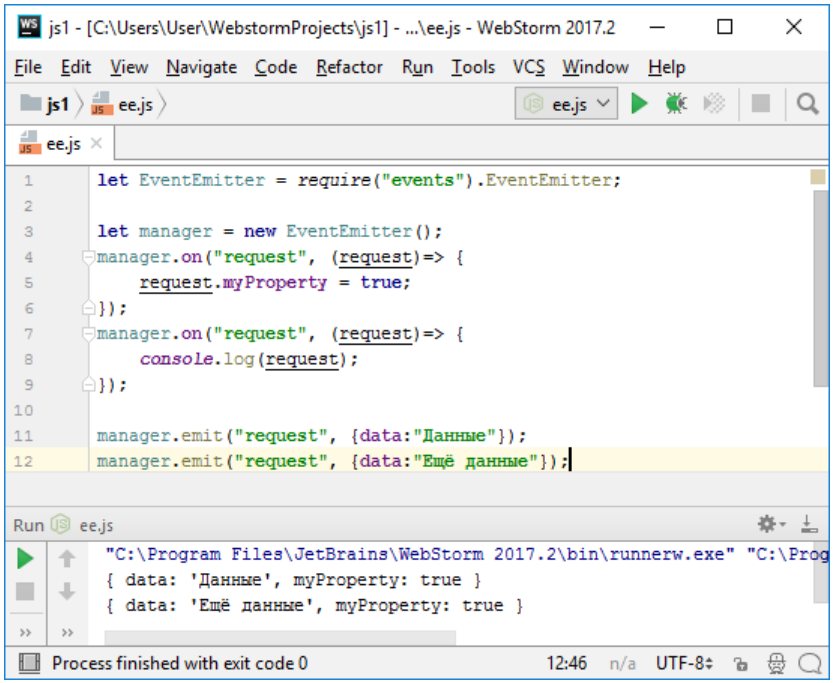
pnpm – аналог npm с ведением единого хранилища модулей



https://pnpm.io/ - доступен только через VPN

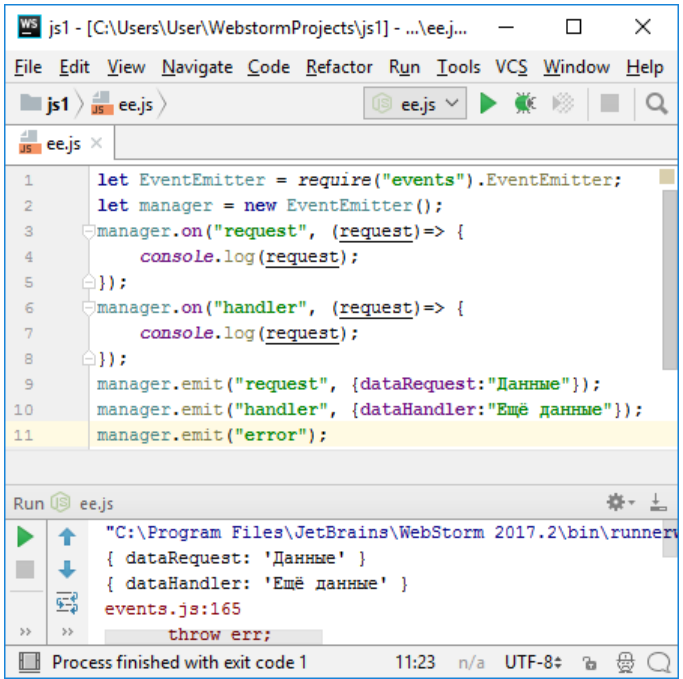
Обработка событий (EventEmitter)


```
let EventEmitter = require("events").EventEmitter;
let manager = new EventEmitter();
manager.on("request", (request)=> {
    request.myProperty = true;
});
manager.on("request", (request)=> {
    console.log(request);
});
manager.emit("request", {data:"Данные"});
manager.emit("request", {data:"Ещё данные"});
```

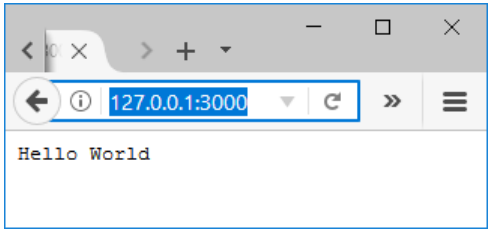


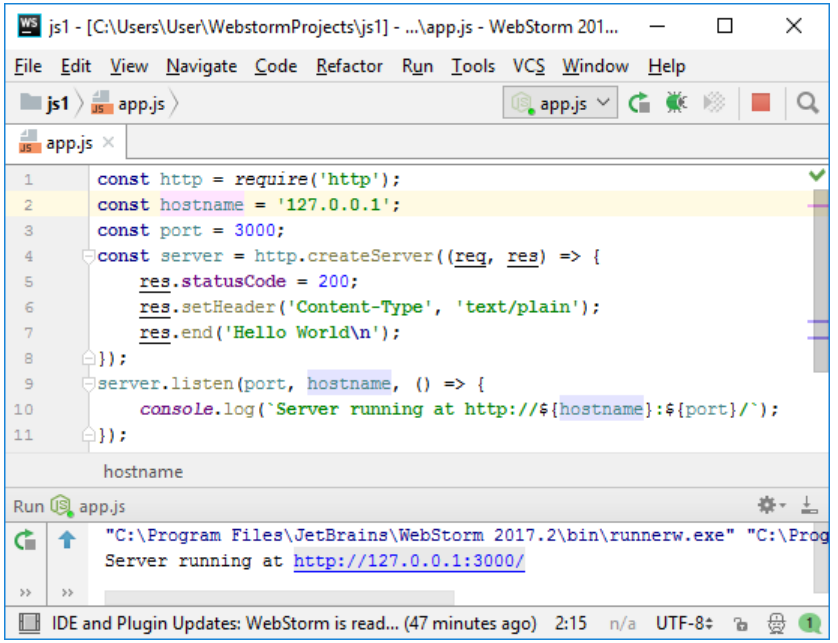
Ошибки


```
let EventEmitter = require("events").EventEmitter;
let manager = new EventEmitter();
manager.on("request", (request)=> {
    console.log(request);
});
manager.on("handler", (request)=> {
    console.log(request);
});
manager.emit("request", {dataRequest:"Данные"});
manager.emit("handler", {dataHandler:"Ещё данные"});
manager.emit("error");
```



Простейший сервер (1)

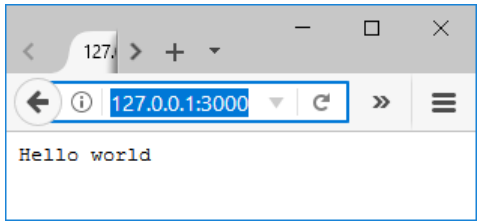


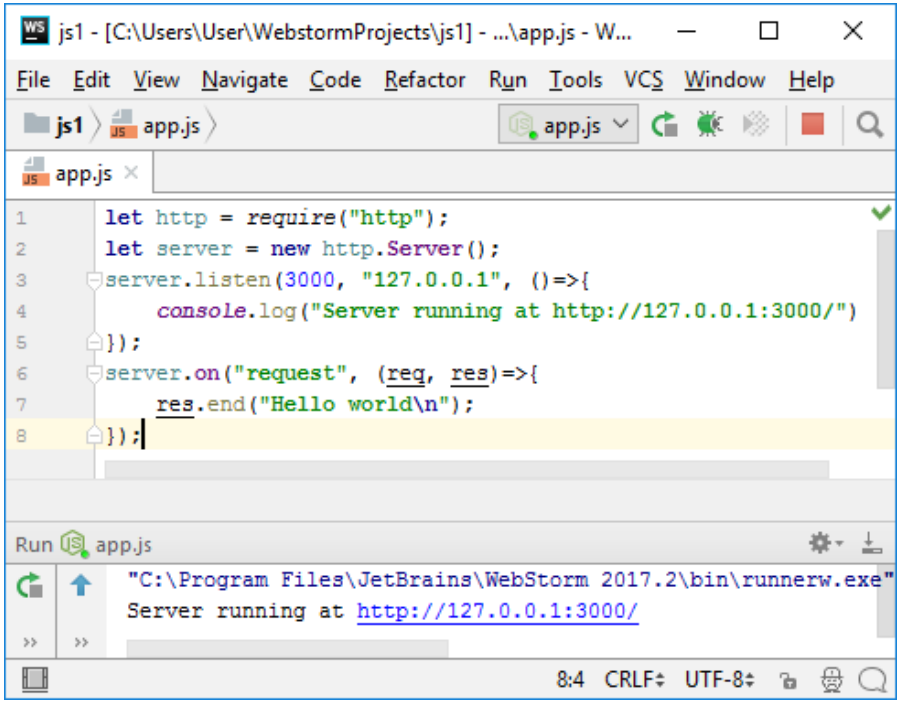



```
const http = require('http');

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}/`);
});
```

Простейший сервер (2)

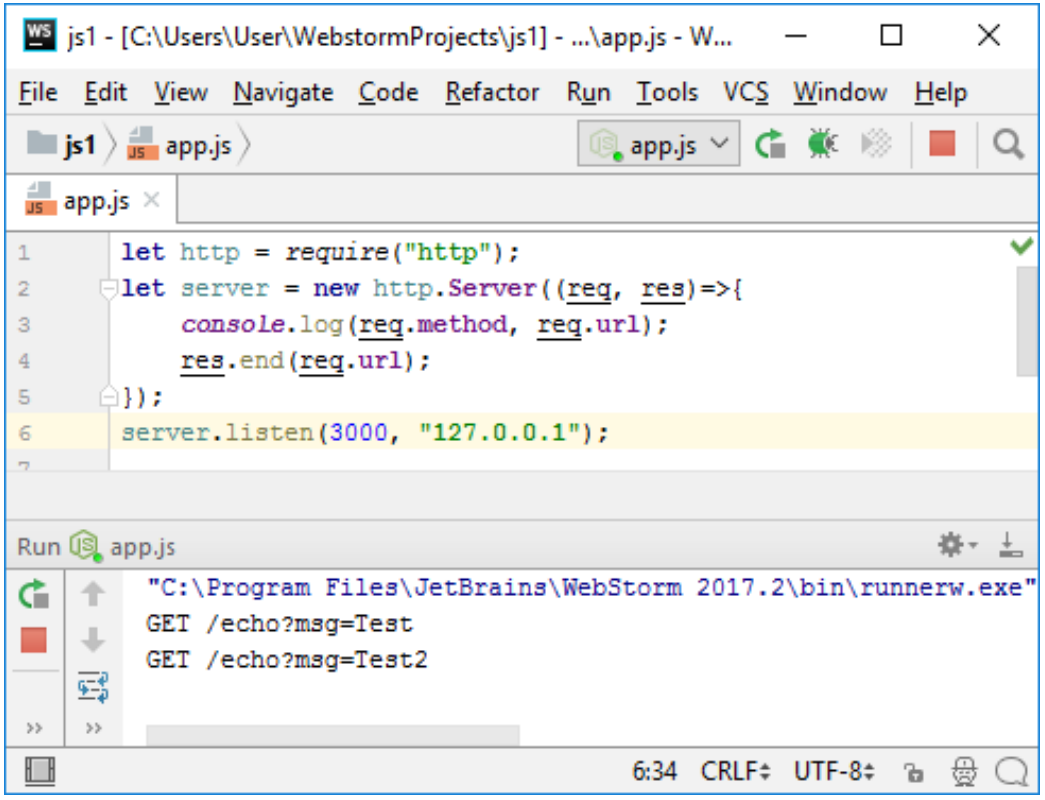

```
let http = require("http");
let server = new http.Server();
server.listen(3000, "127.0.0.1", ()=>{
    console.log("Server running at http://127.0.0.1:3000/")
});
server.on("request", (req, res)=>{
    res.end("Hello world\n");
});
```

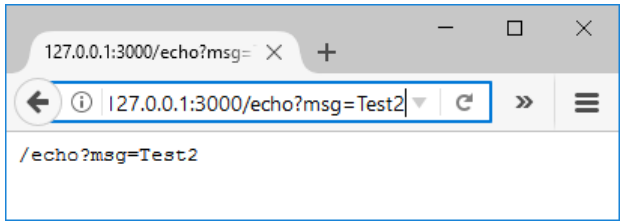




Echo-сервер

```
let http = require("http");
let server = new http.Server((req, res)=>{
    console.log(req.method, req.url);
});
server.listen(3000, "127.0.0.1");
```





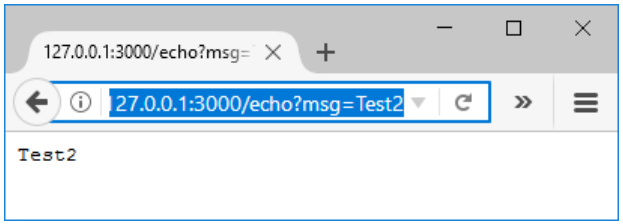
HTTPS-сервер

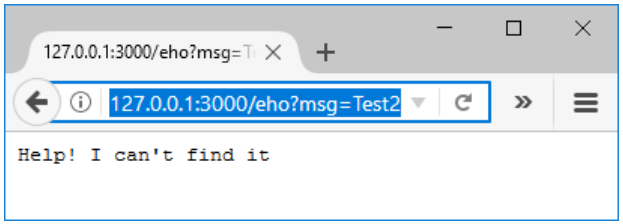

```
// curl -k https://localhost:443/
const https = require('https');
const fs = require('fs');

const options = {
    key: fs.readFileSync('cert/example.com_nginx.key'),
    cert: fs.readFileSync('cert/example.com_nginx.crt')
}; // Важно! Последовательное чтение файлов

https.createServer(options, (req, res) => {
    res.writeHead(200);
    res.end('hello world\n');
}).listen(443);
```

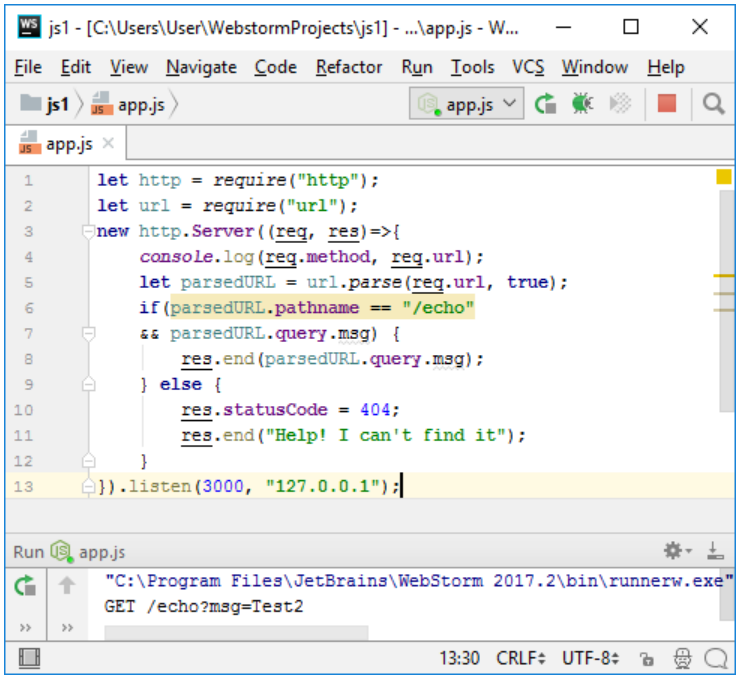
openssl req -new -sha256 -key example.com_nginx.key -out example.com.csr

Разбор параметров запроса


```
let http = require("http");
let url = require("url");
new http.Server((req, res)=>{
    console.log(req.method, req.url);
    let parsedURL = url.parse(req.url, true);
    if(parsedURL.pathname == "/echo"
    && parsedURL.query.msg) {
        res.end(parsedURL.query.msg);
    } else {
        res.statusCode = 404;
        res.end("Help! I can't find it")
    }
}).listen(3000, "127.0.0.1");
```



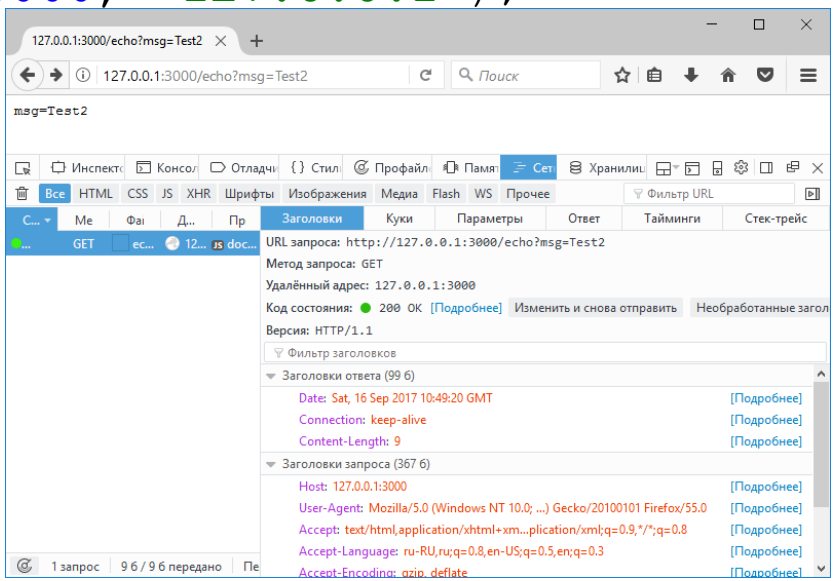


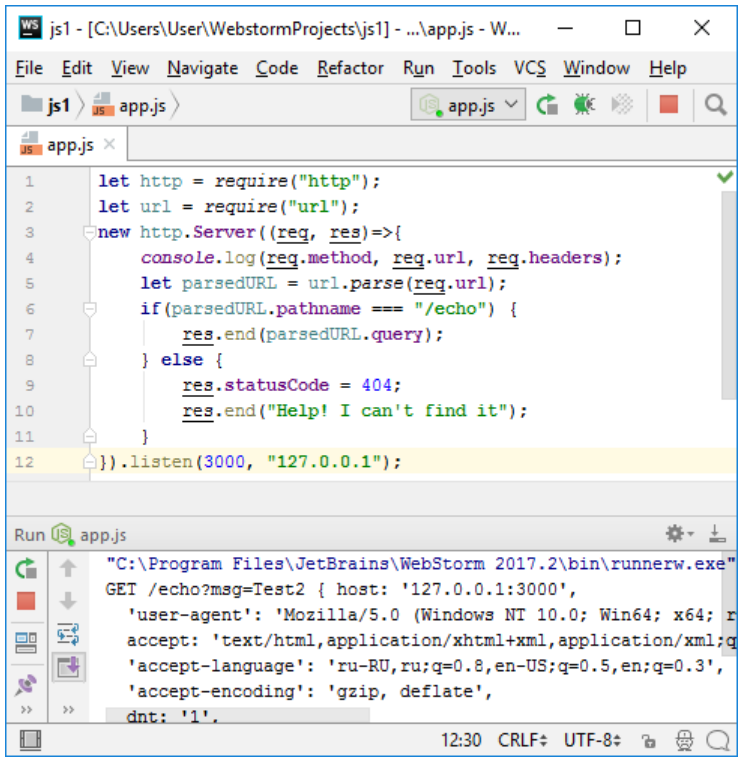


Заголовки


```
let http = require("http");
let url = require("url");
new http.Server((req, res)=>{
    console.log(req.method, req.url, req.headers);
    let parsedURL = url.parse(req.url);
    if(parsedURL.pathname === "/echo") {
        res.end(parsedURL.query);
    } else {
        res.statusCode = 404;
        res.end("Help! I can't find it")
    }
}).listen(3000, "127.0.0.1");
```

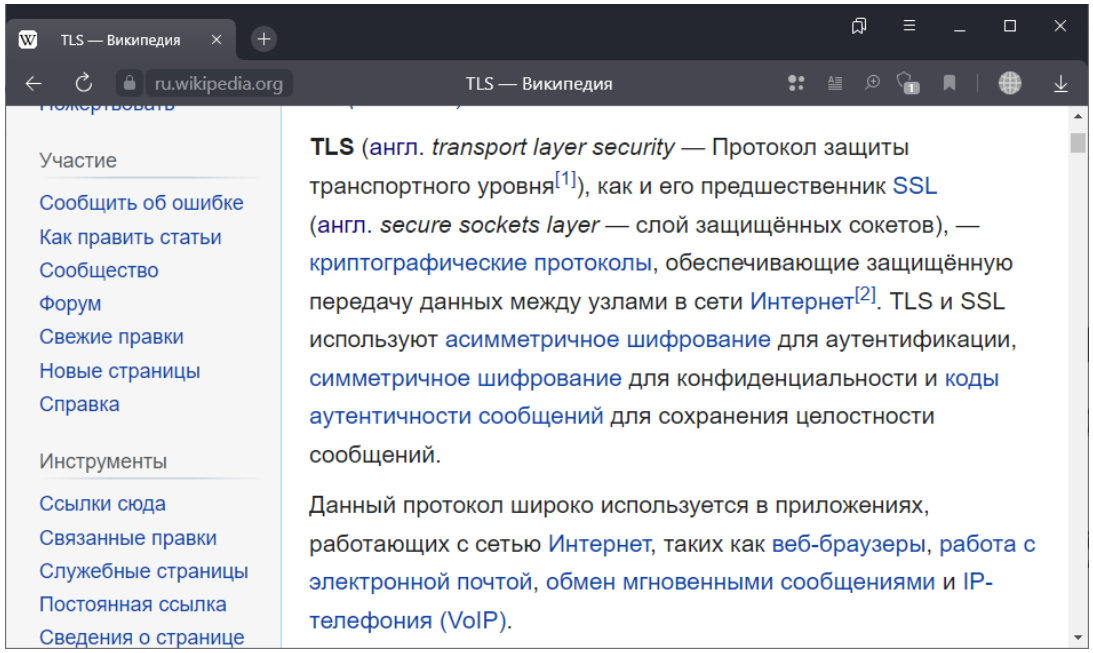
res.setHeader(name, value)





TLS сокет: сервер и клиент
Подготовка ключей

- openssl genrsa -out **private-key.pem** 1024
- openssl req -new -key **private-key.pem** -out **csr.pem**
- openssl x509 -req -in **csr.pem** -signkey **private-key.pem** -out **public-cert.pem**



https://ru.wikipedia.org/wiki/TLS

TLS сокет-сервер


```
var fs = require('fs');
var options = {
    key: fs.readFileSync('private-key.pem'),
    cert: fs.readFileSync('public-cert.pem')
};
var server = require('tls').createServer(options, function(socket) {
    socket.write("Message from server"); // Приветствие
    socket.on('data', function(data) { // Печать данных
        console.log(`Recieved: '${data.toString()}'`);
    });
    socket.on('end', function() { // Передача завершена
        console.log('EOT');
    });
});
server.listen(1337, '127.0.0.1', function() { // Старт
    console.log("I'm listening at %s, on port %s", '127.0.0.1', 1337);
});
server.on('error', function(error) {
    console.error(error); // Обработка ошибки
    server.destroy(); // Удаление
});
```

TLS сокет-клиент


```
var fs = require('fs');
var options = { // Сертификаты
    key: fs.readFileSync('private-key.pem'),
    cert: fs.readFileSync('public-cert.pem'),
    rejectUnauthorized: false
};
var client = require('tls').connect(1337, '127.0.0.1', options, function() {
    if (client.authorized) // Проверка авторизации
        console.log("Connection authorized by a Certificate Authority.");
    else
        console.log("Connection not authorized: " + client.authorizationError)
    client.write("Message from client"); // Приветствие
});
client.on("data", function(data) {
    console.log(`Recieved: '${data.toString()}'`);
    client.end(); // Закрыть соединение
});
client.on('close', function() {
    console.log("Connection closed");
});
client.on('error', function(error) {
    console.error(error); // Отображение ошибки
    client.destroy(); // Закрытие после ошибки
});
```

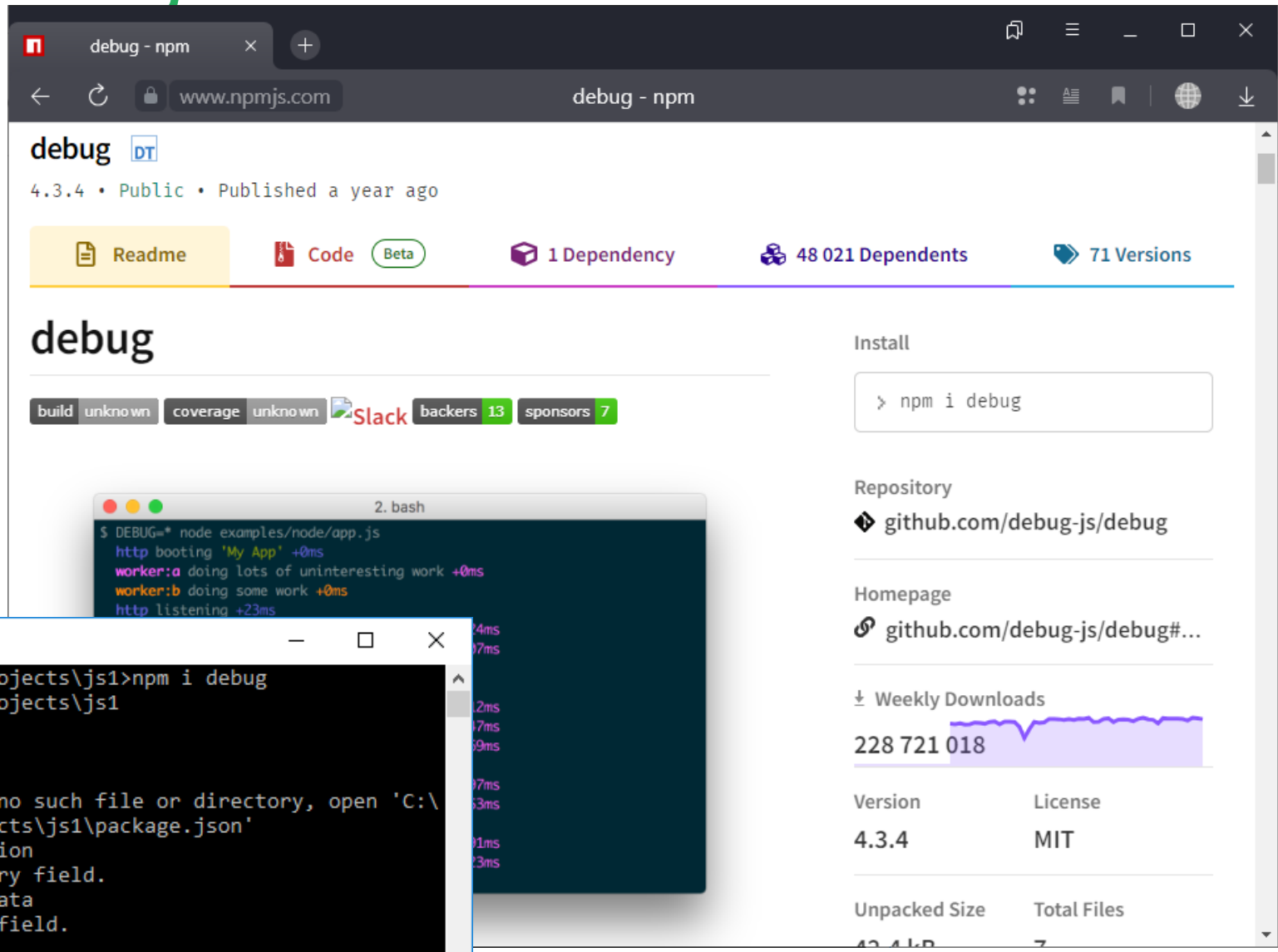
Журнал сервера:

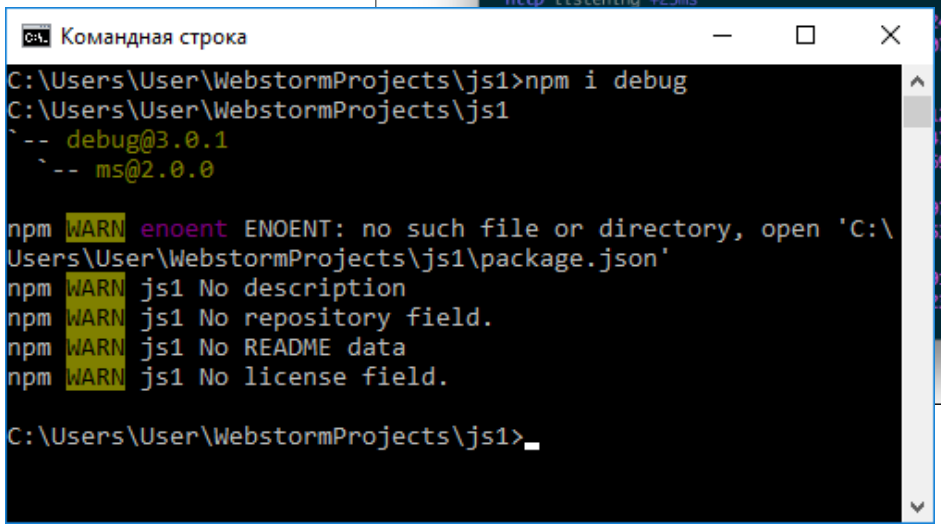
I'm listening at 127.0.0.1, on port 1337
Recieved: 'Message from client'
EOT

Журнал клиента:

Connection not authorized: DEPTH_ZERO_SELF_SIGNED_CERT
Recieved: 'Message from server'
Connection closed

npm i debug


debug - npm
www.npmjs.com
debug - npm
debug
DT
4.3.4 • Public • Published a year ago
Readme
Code
Beta
1 Dependency
48 021 Dependents
71 Versions
debug
build unknown
coverage unknown
Slack
backers 13
sponsors 7
2. bash
$ DEBUG=* node examples/node/app.js
http booting 'My App' +0ms
worker:a doing lots of uninteresting work +0ms
worker:b doing some work +0ms
http listening +23ms
Install
> npm i debug
Repository
github.com/debug-js/debug
Homepage
github.com/debug-js/debug#...
Weekly Downloads
228 721 018
Version
4.3.4
License
MIT
Unpacked Size
Total Files


Командная строка
C:\Users\User\WebstormProjects\js1>npm i debug
C:\Users\User\WebstormProjects\js1
`-- debug@3.0.1
`-- ms@2.0.0
npm WARN enoent ENOENT: no such file or directory, open 'C:\Users\User\WebstormProjects\js1\package.json'
npm WARN js1 No description
npm WARN js1 No repository field.
npm WARN js1 No README data
npm WARN js1 No license field.
C:\Users\User\WebstormProjects\js1>

Демонстрация debug

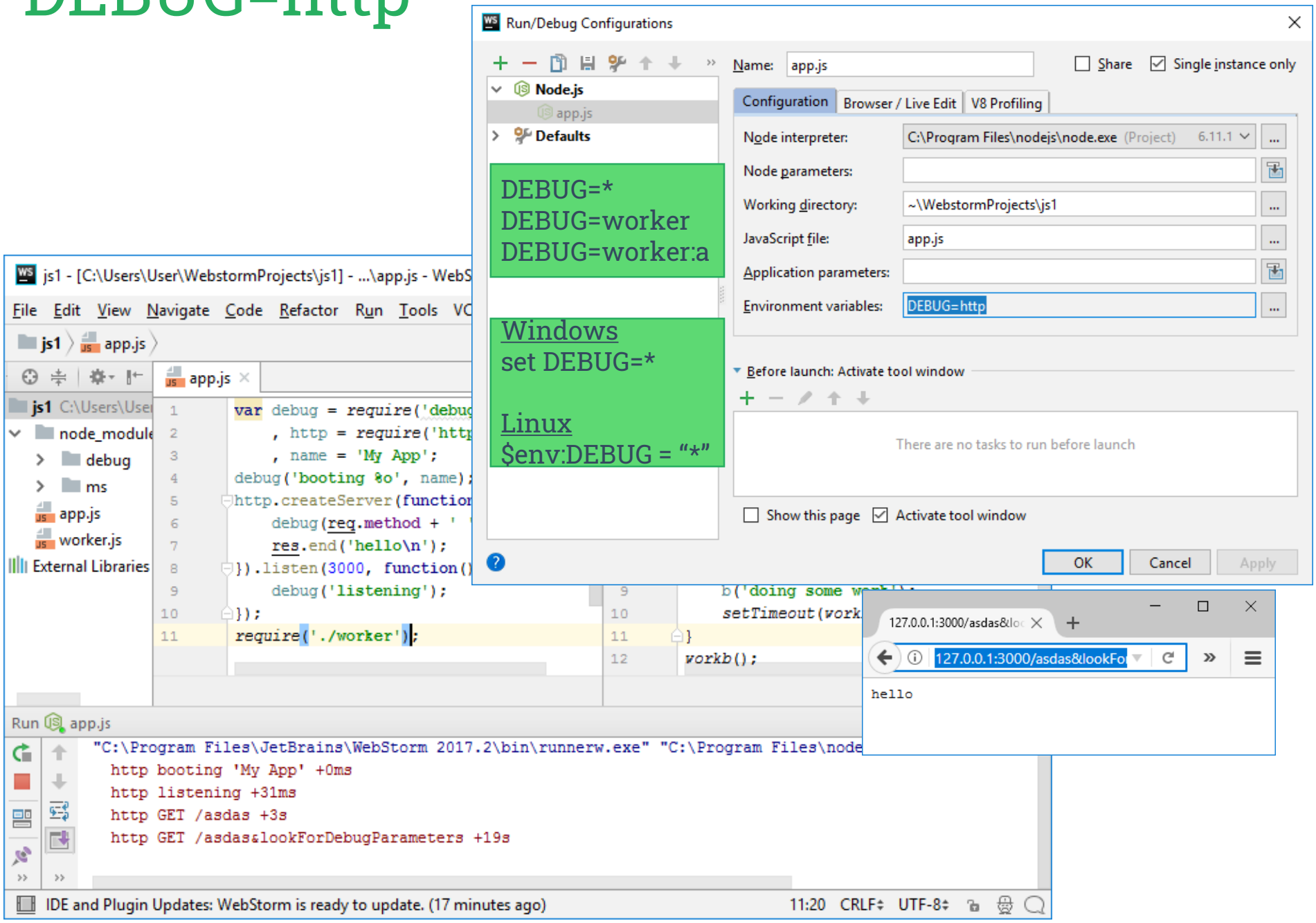
app.js

```
var debug = require('debug')('http')
    , http = require('http')
    , name = 'My App';
debug('booting %o', name);
http.createServer(function(req, res){
    debug(req.method + ' ' + req.url);
    res.end('hello\n');
}).listen(3000, function(){
    debug('listening');
});
require('./worker');
```

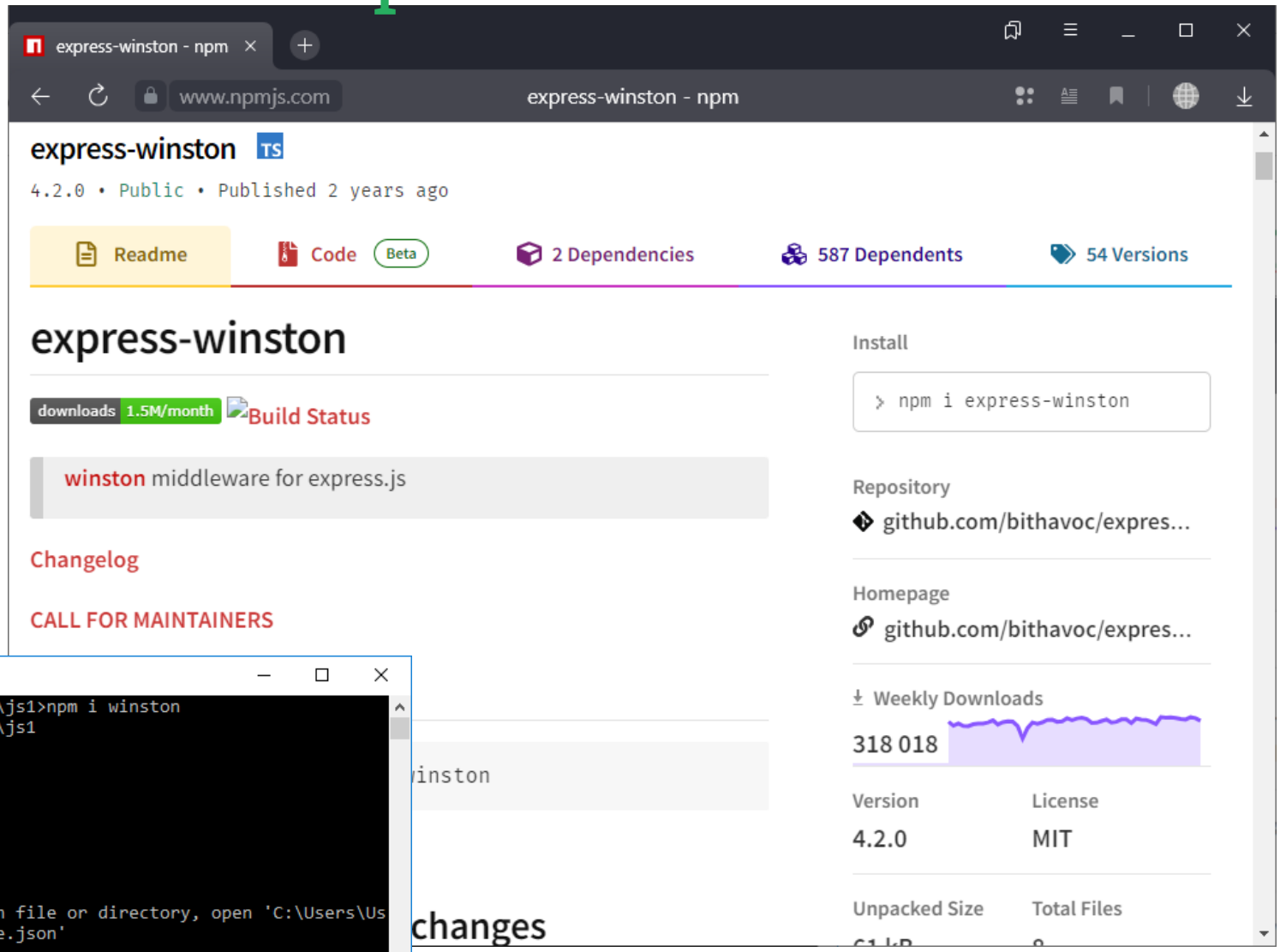
worker.js

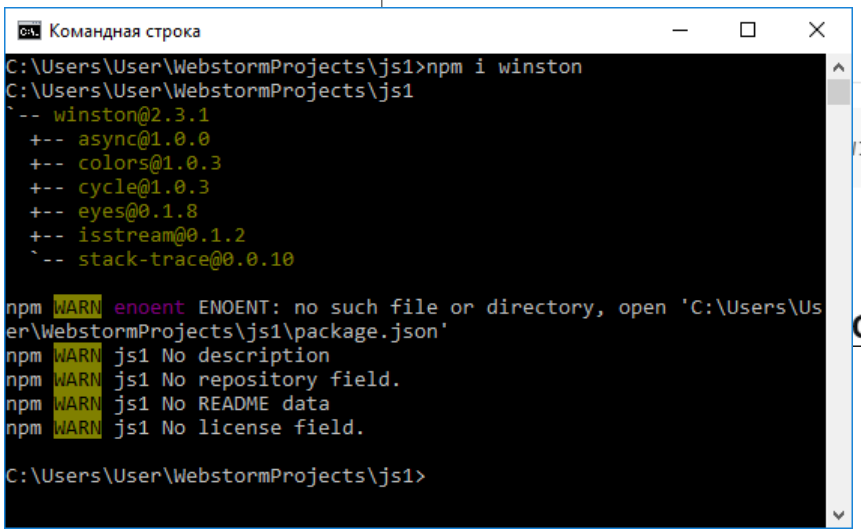
```
var a = require('debug')('worker:a')
    , b = require('debug')('worker:b');
function work() {
    a('doing lots of uninteresting work');
    setTimeout(work, Math.random() * 1000);
}
work();
function workb() {
    b('doing some work');
    setTimeout(workb, Math.random() * 2000);
}
workb();
```

DEBUG=http


Run/Debug Configurations
Name: app.js
Share
Single instance only
Node.js
app.js
Defaults
Configuration
Browser / Live Edit
V8 Profiling
Node interpreter: C:\Program Files\nodejs\node.exe (Project) 6.11.1
Node parameters:
Working directory: ~\WebstormProjects\js1
JavaScript file: app.js
Application parameters:
Environment variables: DEBUG=http
Before launch: Activate tool window
There are no tasks to run before launch
Show this page
Activate tool window
OK
Cancel
Apply
DEBUG=*
DEBUG=worker
DEBUG=worker:a
Windows
set DEBUG=*
Linux
$env:DEBUG = "*"
js1 - [C:\Users\User\WebstormProjects\js1] - ...\app.js - WebS
File Edit View Navigate Code Refactor Run Tools VC
js1 app.js
app.js
var debug = require('debug
, http = require('http
, name = 'My App';
debug('booting %o', name);
http.createServer(function
debug(req.method + '
res.end('hello\n');
}).listen(3000, function()
debug('listening');
});
require('./worker');
node_module
debug
ms
app.js
worker.js
External Libraries
b('doing some work');
setTimeout(work
}
workb();
127.0.0.1:3000/asdas&lookFo
hello
Run app.js
"C:\Program Files\JetBrains\WebStorm 2017.2\bin\runnerw.exe" "C:\Program Files\node
http booting 'My App' +0ms
http listening +31ms
http GET /asdas +3s
http GET /asdas&lookForDebugParameters +19s
IDE and Plugin Updates: WebStorm is ready to update. (17 minutes ago)
11:20 CRLF UTF-8

npm i winston express-winston





Демонстрация winston

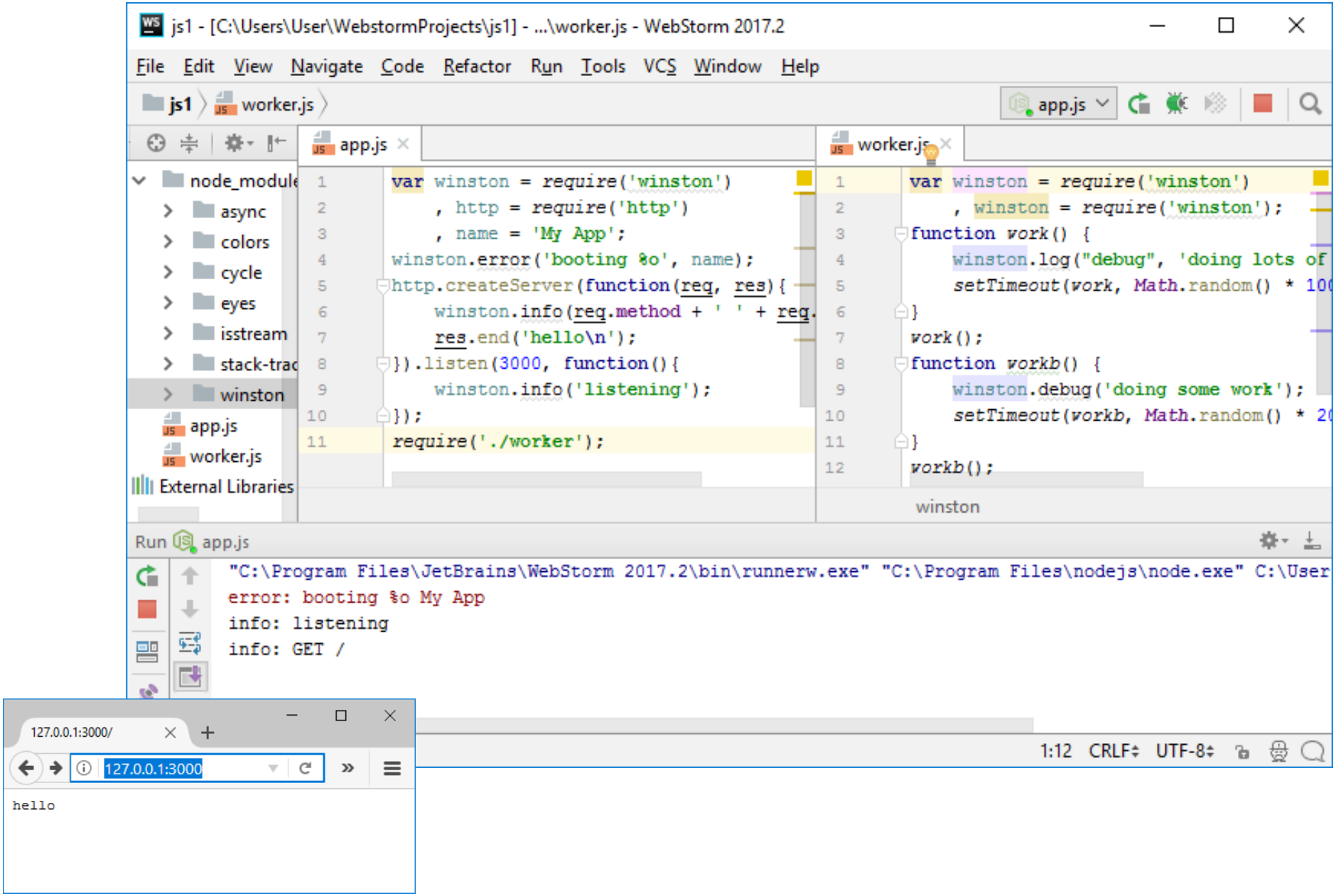
app.js

```
var winston = require('winston')
    , http = require('http')
    , name = 'My App';
winston.error('booting %o', name);
http.createServer(function(req, res){
    winston.info(req.method + ' ' + req.url);
    res.end('hello\n');
}).listen(3000, function(){
    winston.info('listening');
});
require('./worker');
```

worker.js

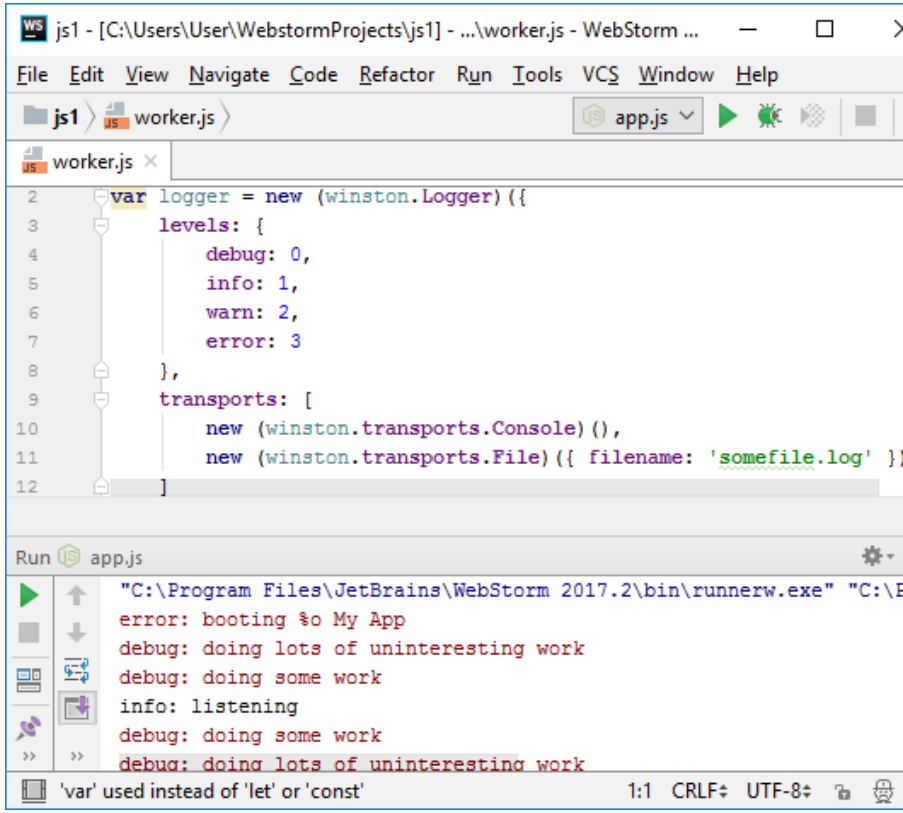
```
var winston = require('winston');
function work() {
    winston.log("debug", 'doing lots of uninteresting work');
    setTimeout(work, Math.random() * 1000);
}
work();
function workb() {
    winston.debug('doing some work');
    setTimeout(workb, Math.random() * 2000);
}
workb();
```

Логгирование по умолчанию

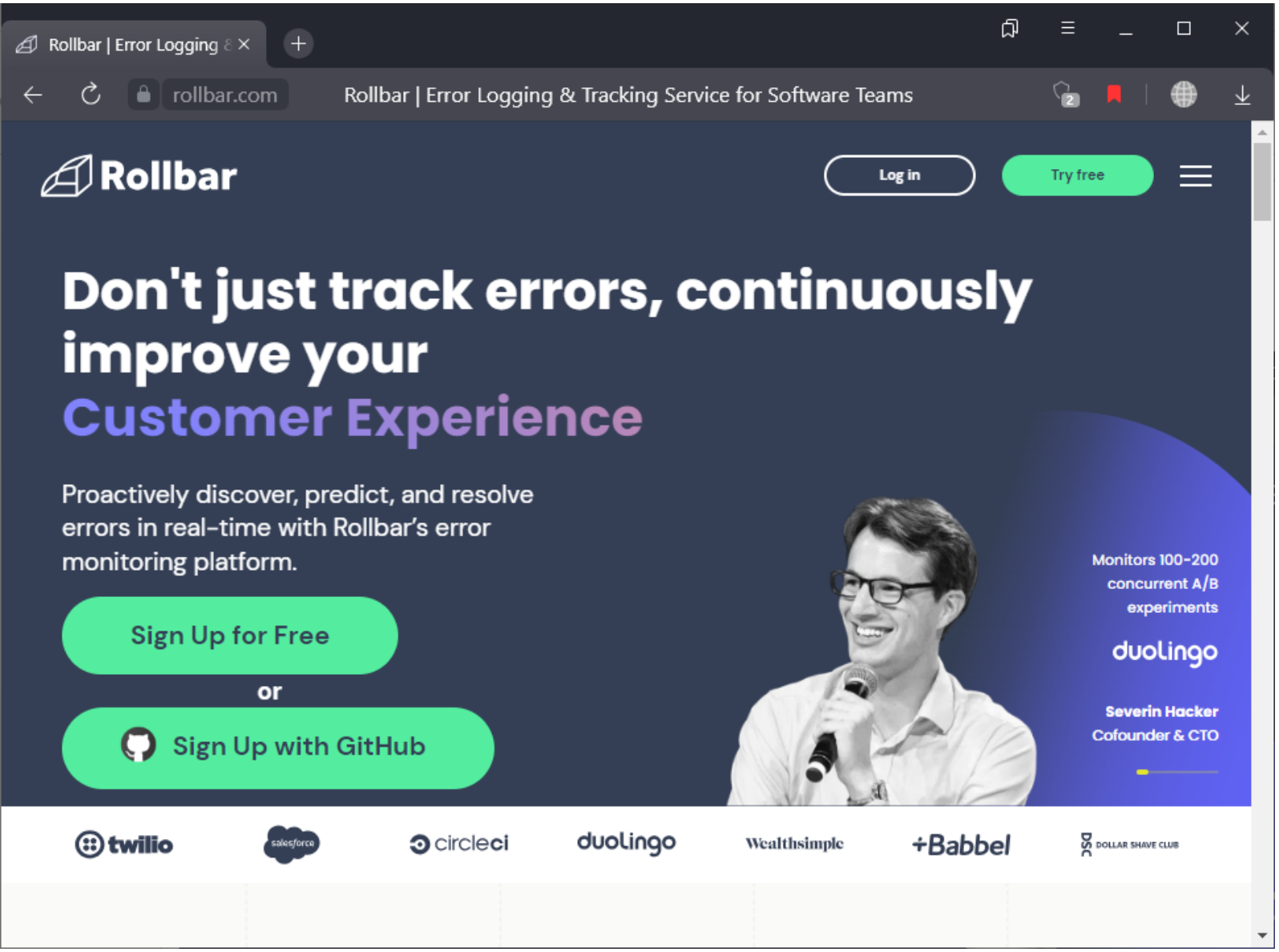


Настраиваемое логгирование


```
var logger = new (winston.Logger)({
    levels: {
        debug: 0,
        info: 1,
        warn: 2,
        error: 3
    },
    transports: [
        new (winston.transports.Console)(),
        new (winston.transports.File)(
         { filename: 'somefile.log' })
    ]
});
```



https://rollbar.com/



- Java
- Node
- PHP
- JavaScript
- Python
- Android
- iOS
- .Net

Rollbar – установка

- Node.js

- npm install --save rollbar
- yarn add -D rollbar

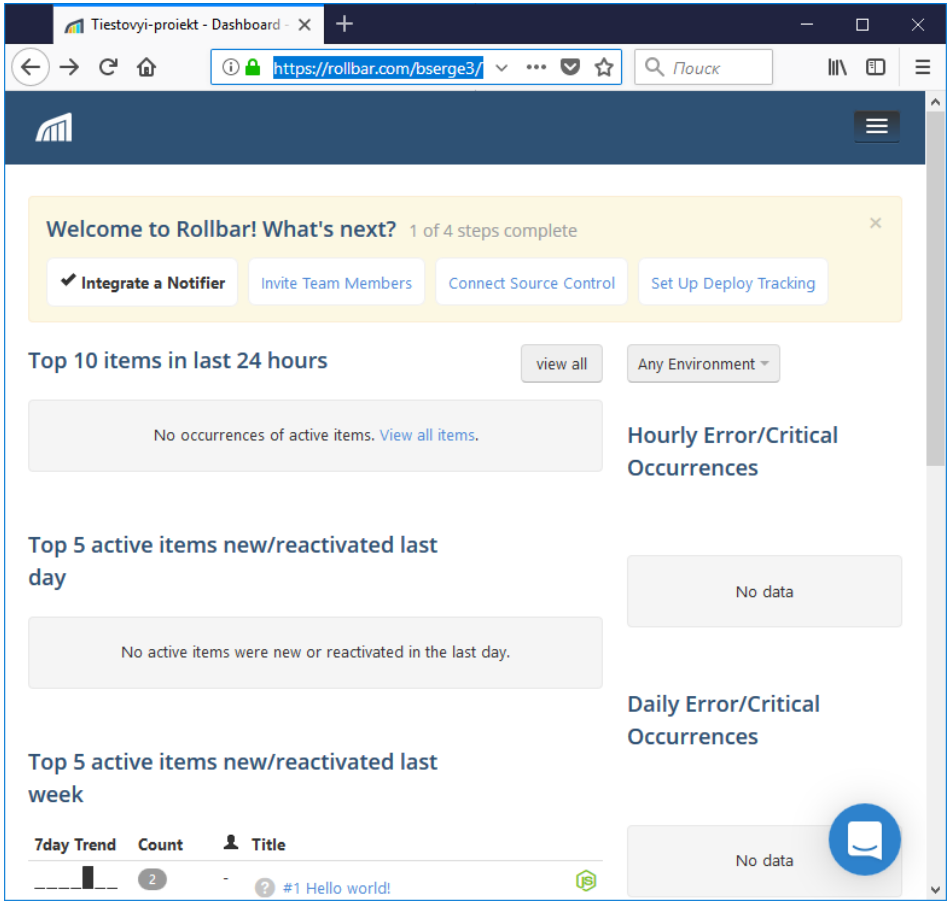
- let Rollbar = require("rollbar");
- let rollbar = new Rollbar("access token");

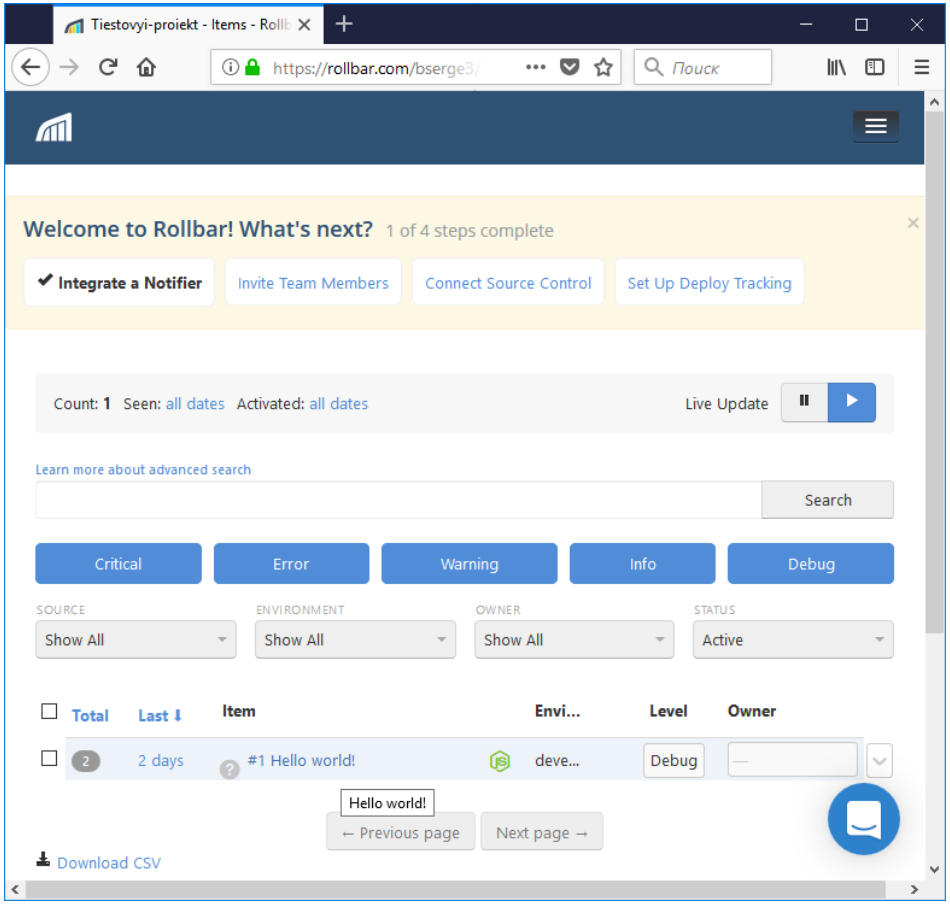
- rollbar.log("TestError: Hello World!");

- JavaScript

- <script>
- var _rollbarConfig = {
- accessToken: "access token",
- captureUncaught: true,
- captureUnhandledRejections: true,
- payload: {
- environment: "production"
- }
- };
- // Rollbar Snippet
- // ...
- // End Rollbar Snippet
- </script>

https://rollbar.com/bserge3/Tiestovyi-proiekt/





Интеграция

Сообщения

- Slack, HipChat, Flowdock, Campfire

Контроль исходного кода

- GitHub, Bitbucket, GitLab

Отслеживания ошибок

- JIRA, Trello, GitHub, Bitbucket, GitLab, Pivotal, Asana, Sprintly

Управление инцидентами

- PagerDuty, VictorOps, OpsGenie, Datadog

https://sentry.io/for/javascript/

- JavaScript
- .NET
- Python
- Android
- PHP
- Django
- Java
- Flask
- Ruby
- Laravel
- iOS
- Rails
- Node
- React
- Go

Sentry – установка

- Node.js

- npm install raven --save
- yarn add -D raven

- let Raven = require('raven');
- Raven.config('your dsn').install();

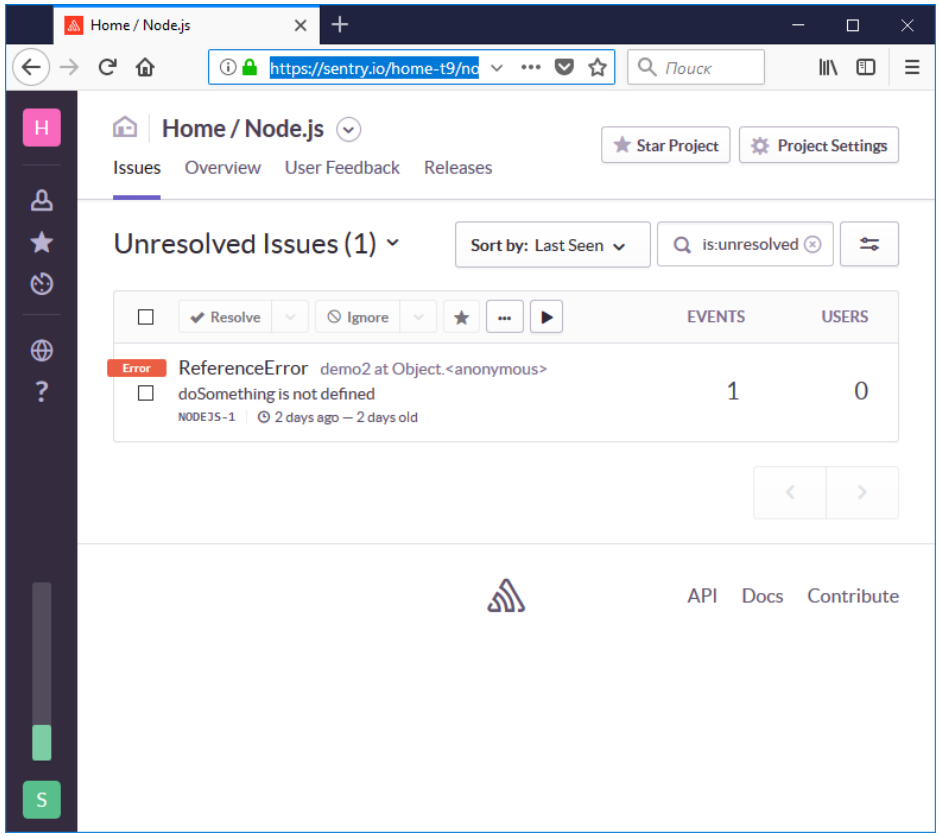
- try {
  - doSomething(a[0]);
- } catch (e) {
  - Raven.captureException(e);
- }

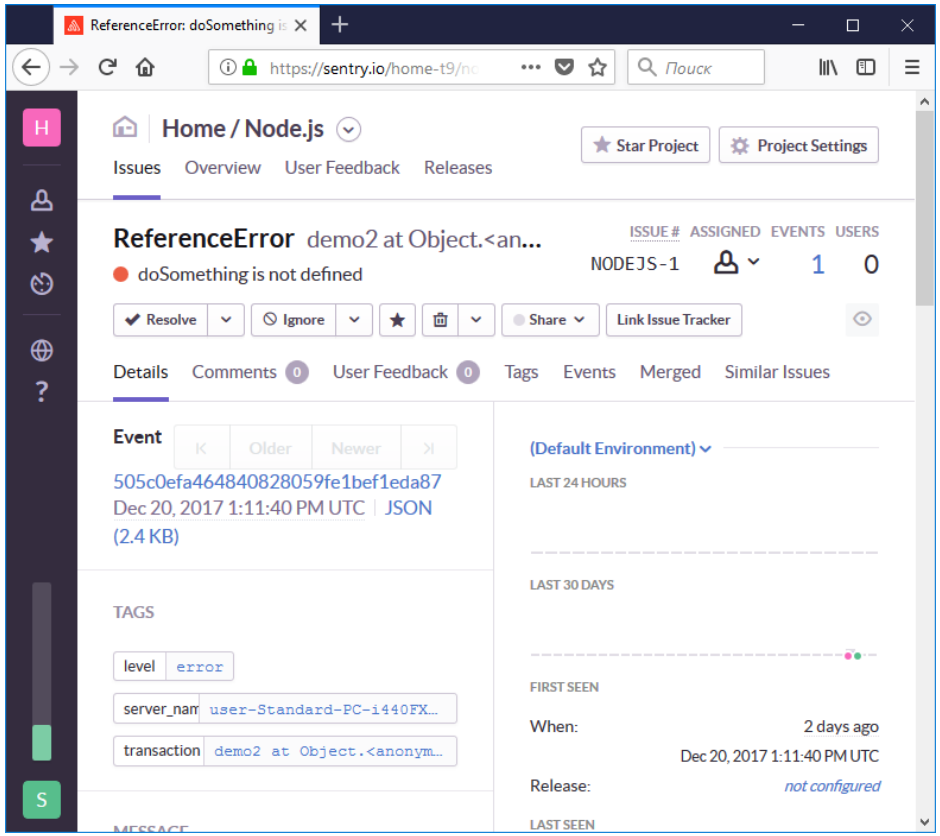
- JavaScript

```
<script src="https://cdn.ravenjs.com/<VERSION>/raven.min.js"></script>
```

```
<script>
    Raven.config('your dsn').install()
    try {
        doSomething(a[0]);
    } catch (e) {
        Raven.captureException(e);
    }
</script>
```

https://home-t9.sentry.io/issues/

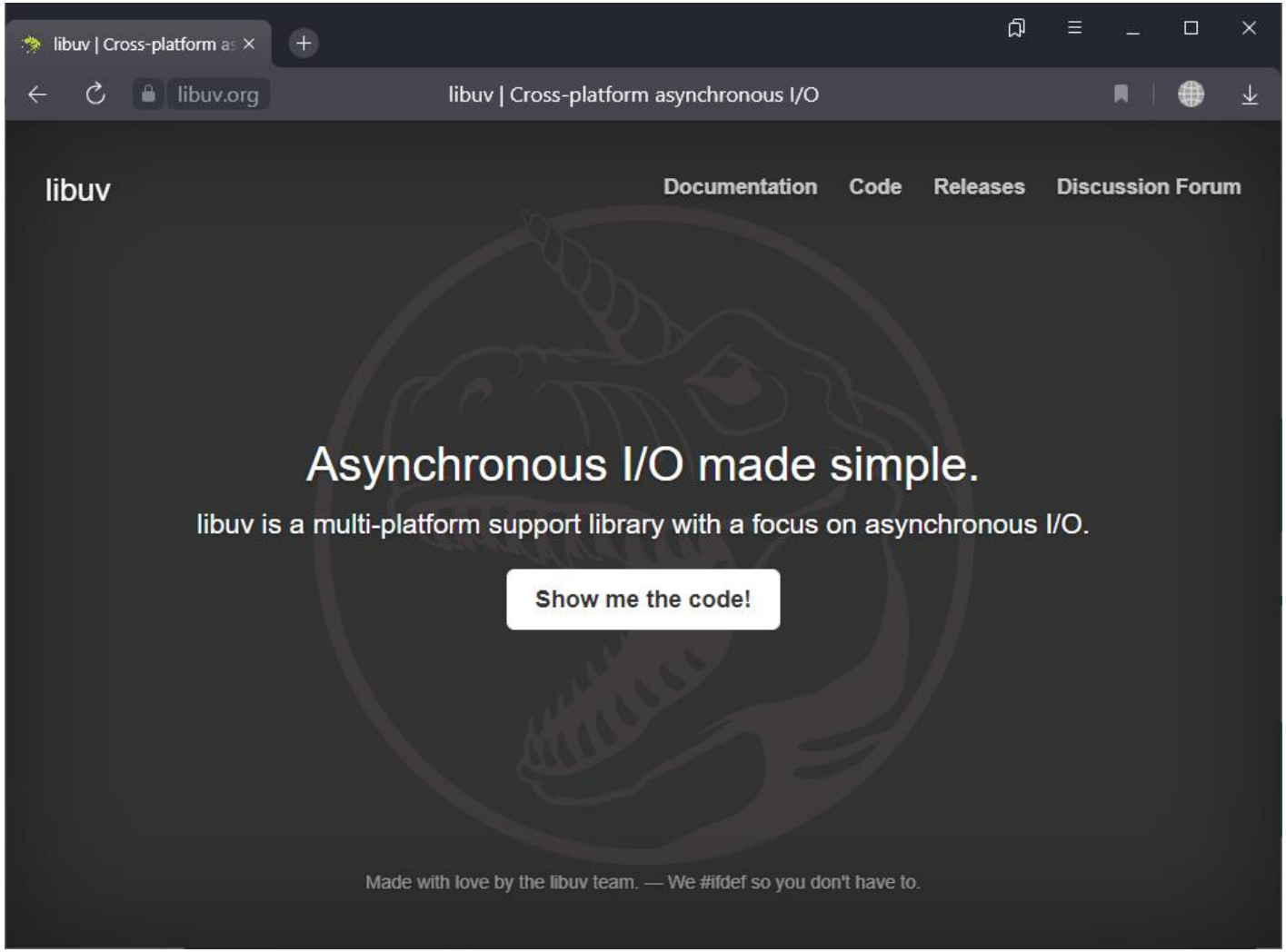


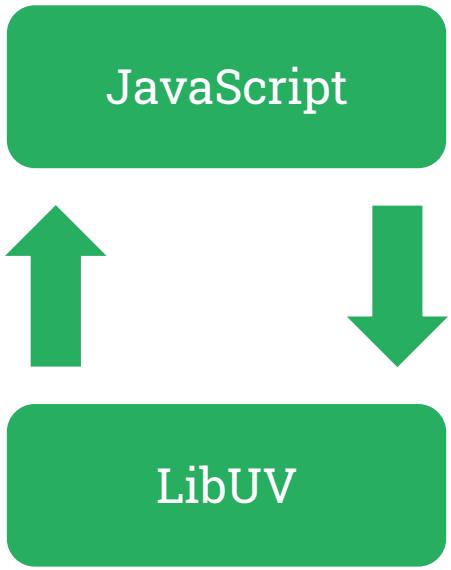


Интеграция

GitHub	Heroku	Slack	SessionStack	Datadog	Trello	Bitbucket
Jira	HipChat	PagerDuty	Asana	Twilio	GitLab	Yubico
Segment	OneLogin	Auth0	Okta	Hosted Graphite	GCP	Pivotal Tracker
Grove	OpsGenie	Campfire	Flowdock	Pushover	Lighthouse	Phabricator
Redmine	Teamwork	Taiga	Sprint.ly	YouTrack	VictorOps	

LibUV

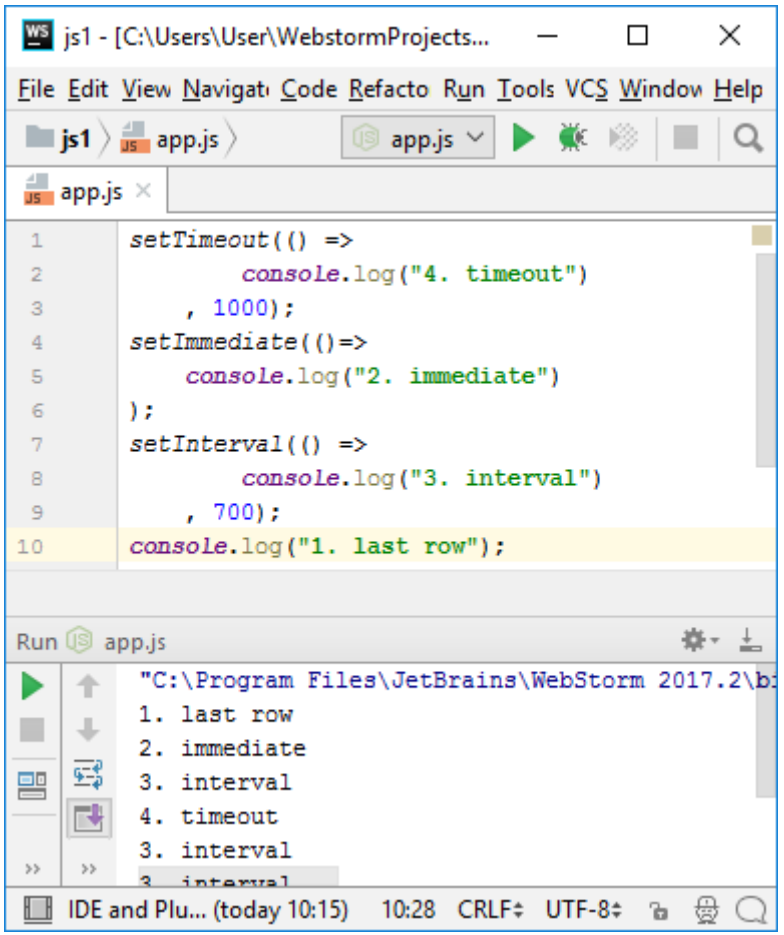




https://libuv.org/

Вызов с задержкой


```
setTimeout(() =>
        console.log("4. timeout")
    , 1000);
setImmediate(()=>
    console.log("2. immediate")
);
setInterval(() =>
        console.log("3. interval")
    , 700);
console.log("1. last row");
```

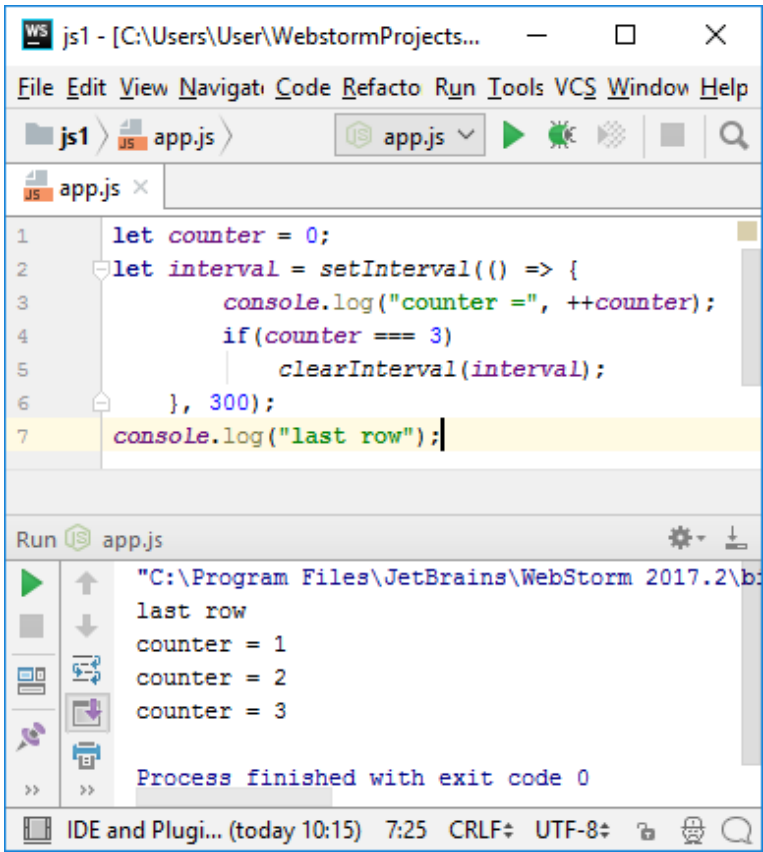


Периодические задачи


```
let counter = 0;
let interval = setInterval(() => {
        console.log("counter =", ++counter);
        if(counter === 3)
            clearInterval(interval);
    }, 300);
console.log("last row");
```

Альтернатива:

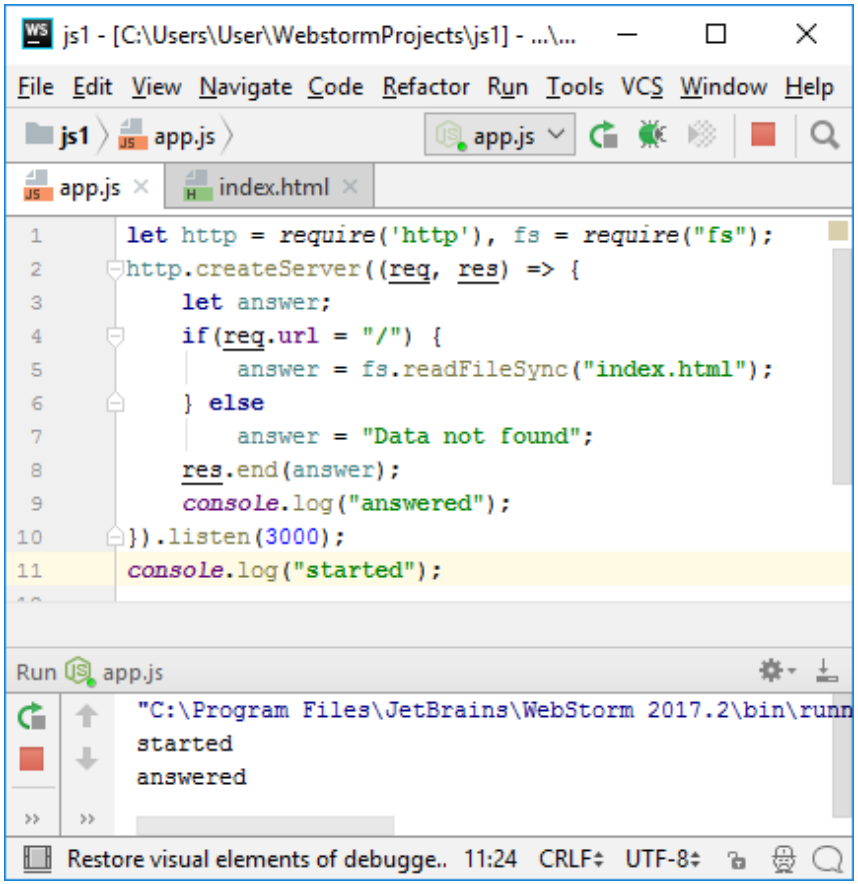
```
let counter = 0;
setTimeout(work, 300);
function work() {
    console.log("counter =", ++counter)
    if(counter !== 3)
        setTimeout(work, 300);
}
console.log("last row");
```



Синхронное чтение файла

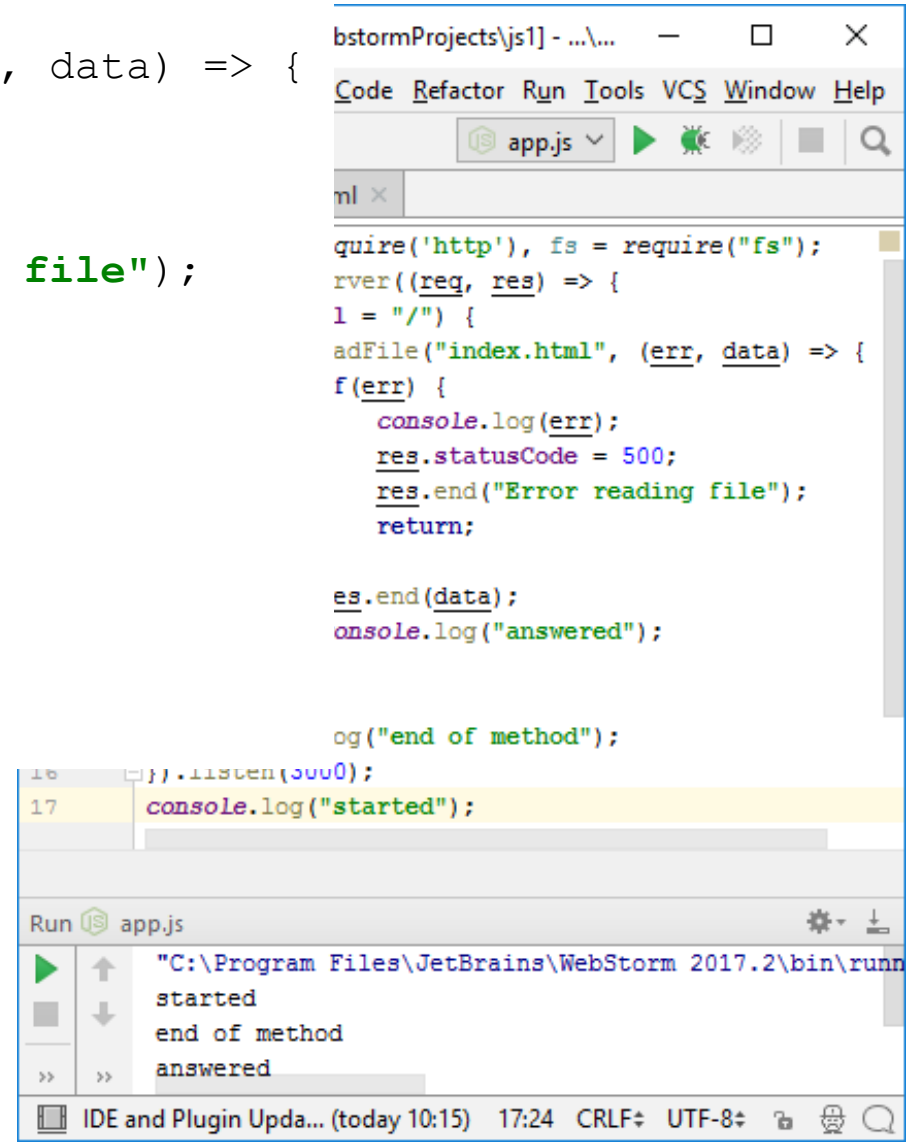

```
let http = require('http'), fs = require("fs");
http.createServer((req, res) => {
    let answer;
    if(req.url = "/") {
        answer = fs.readFileSync("index.html");
    } else
        answer = "Data not found";
    res.end(answer);
    console.log("answered");
}).listen(3000);
console.log("started");
```

Добавить try-catch



Асинхронное чтение файла


```
let http = require('http'), fs = require("fs");
http.createServer((req, res) => {
    if(req.url = "/") {
        fs.readFile("index.html", (err, data) => {
            if(err) {
                console.log(err);
                res.statusCode = 500;
                res.end("Error reading file");
                return;
            }
            res.end(data);
            console.log("answered");
        });
    }
    console.log("end of method");
}).listen(3000);
console.log("started");
```

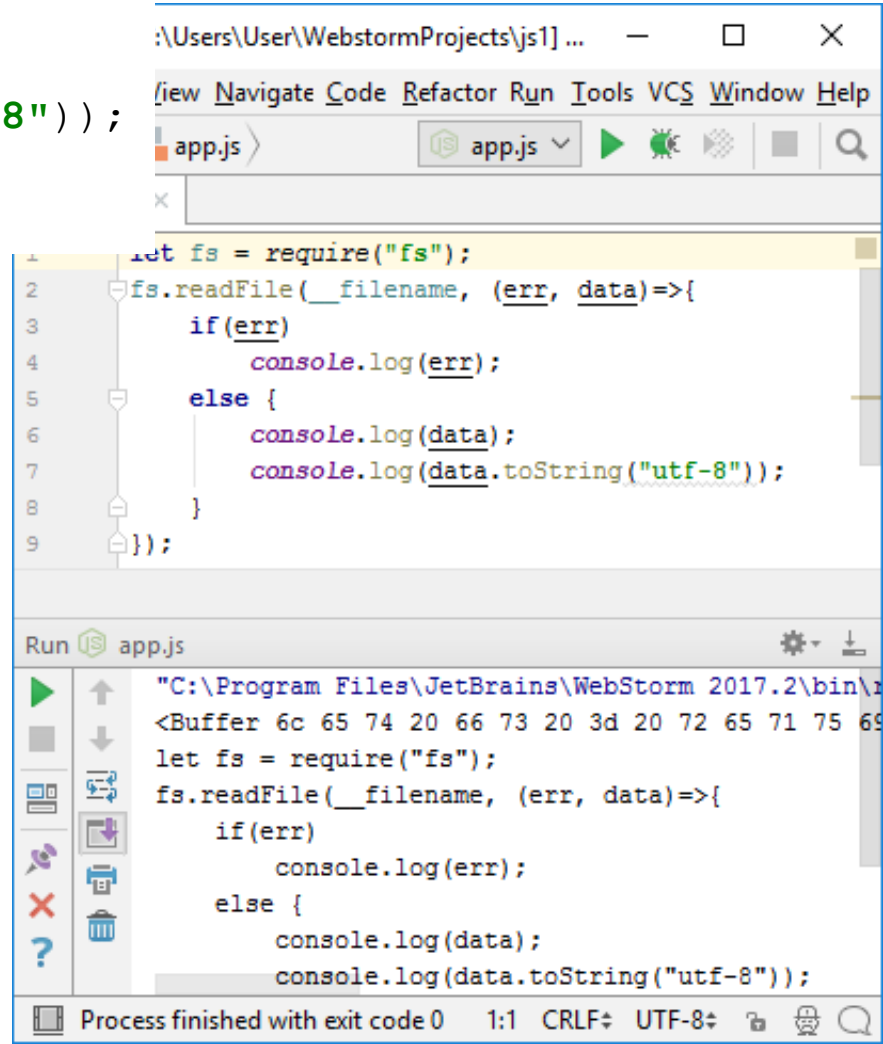


Чтение бинарных данных


```
let fs = require("fs");
fs.readFile(__filename, (err, data)=>{
    if(err)
        console.log(err);
    else {
        console.log(data);
        console.log(data.toString("utf-8"));
    }
});
```

Альтернатива:

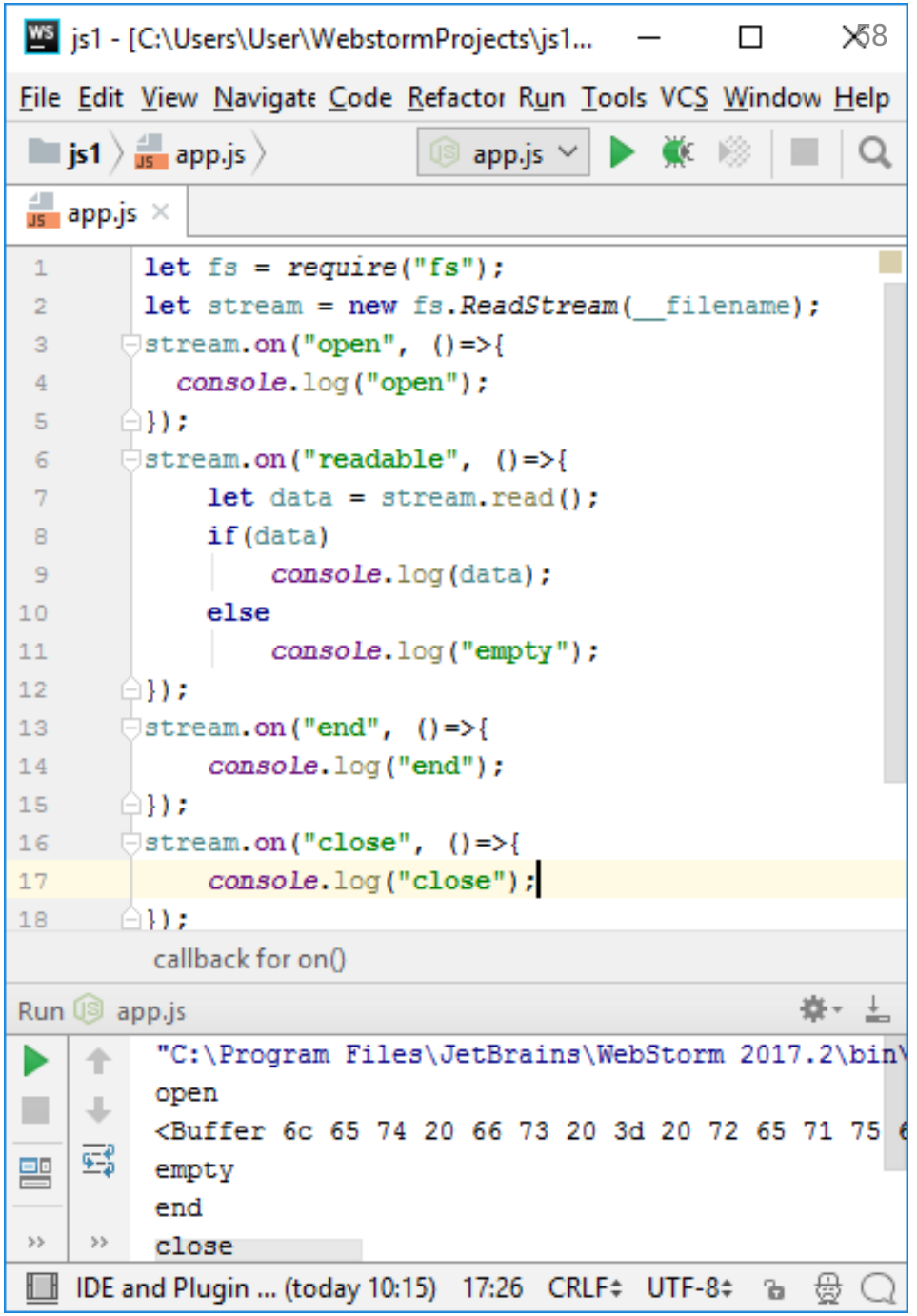
```
let fs = require("fs");
fs.readFile(__filename,
{encoding: "utf-8"},
(err, data)=>{
    if(err)
        console.log(err);
    else
        console.log(data);
});
```



Работа с потоком

```
let fs = require("fs");
let stream =
    new fs.ReadStream(__filename);
stream.on("open", ()=>{
  console.log("open");
});
stream.on("readable", ()=>{
    let data = stream.read();
    if(data)
        console.log(data);
    else
        console.log("empty");
});
stream.on("end", ()=>{
    console.log("end");
});
stream.on("close", ()=>{
    console.log("close");
});
```

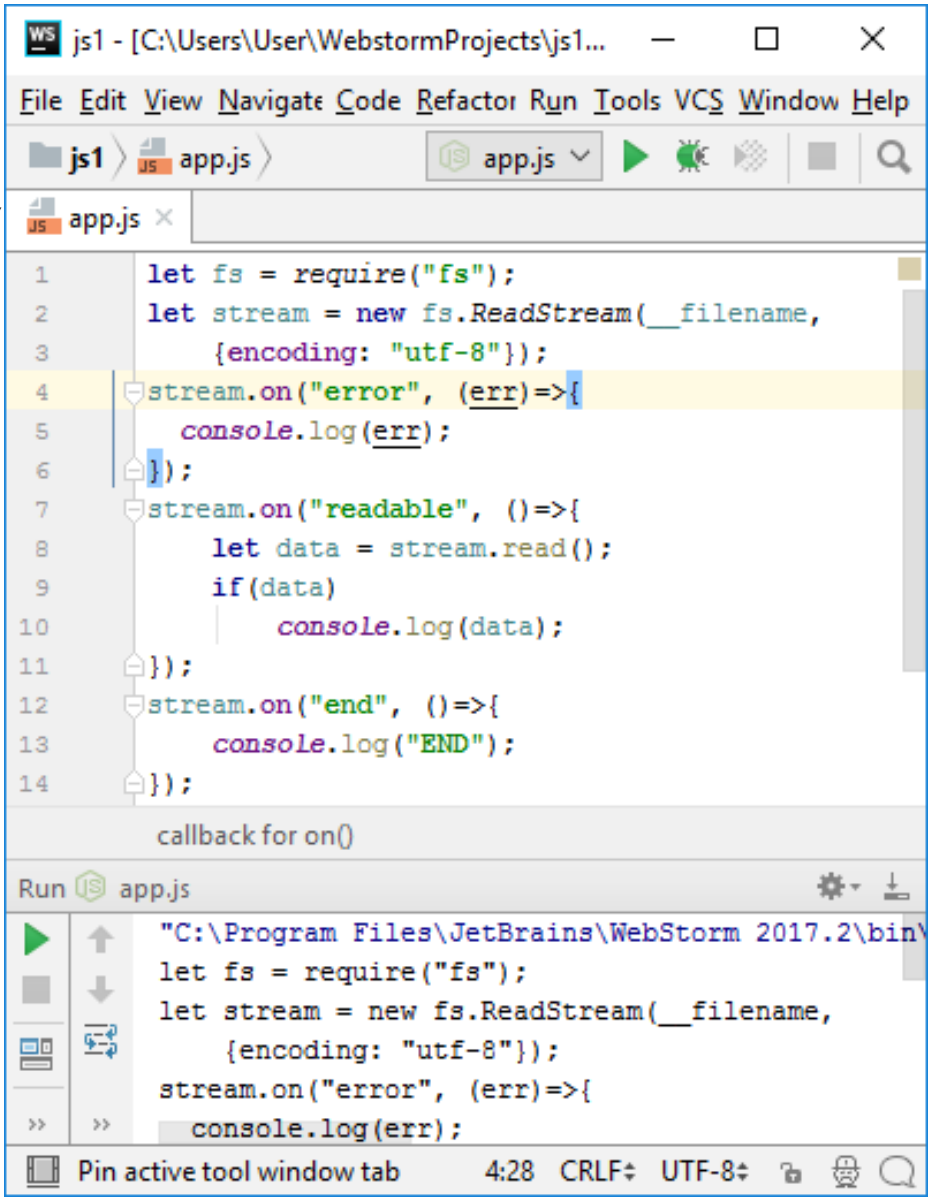
Буфер 64k



Обработка ошибок при работе с потоками


```
let fs = require("fs");
let stream =
    new fs.ReadStream(__filename,
    {encoding: "utf-8"});
stream.on("error", (err)=>{
  console.log(err);
});
stream.on("readable", ()=>{
    let data = stream.read();
    if(data)
        console.log(data);
});
stream.on("end", ()=>{
    console.log("END");
});
```

Универсализм для всех потоков

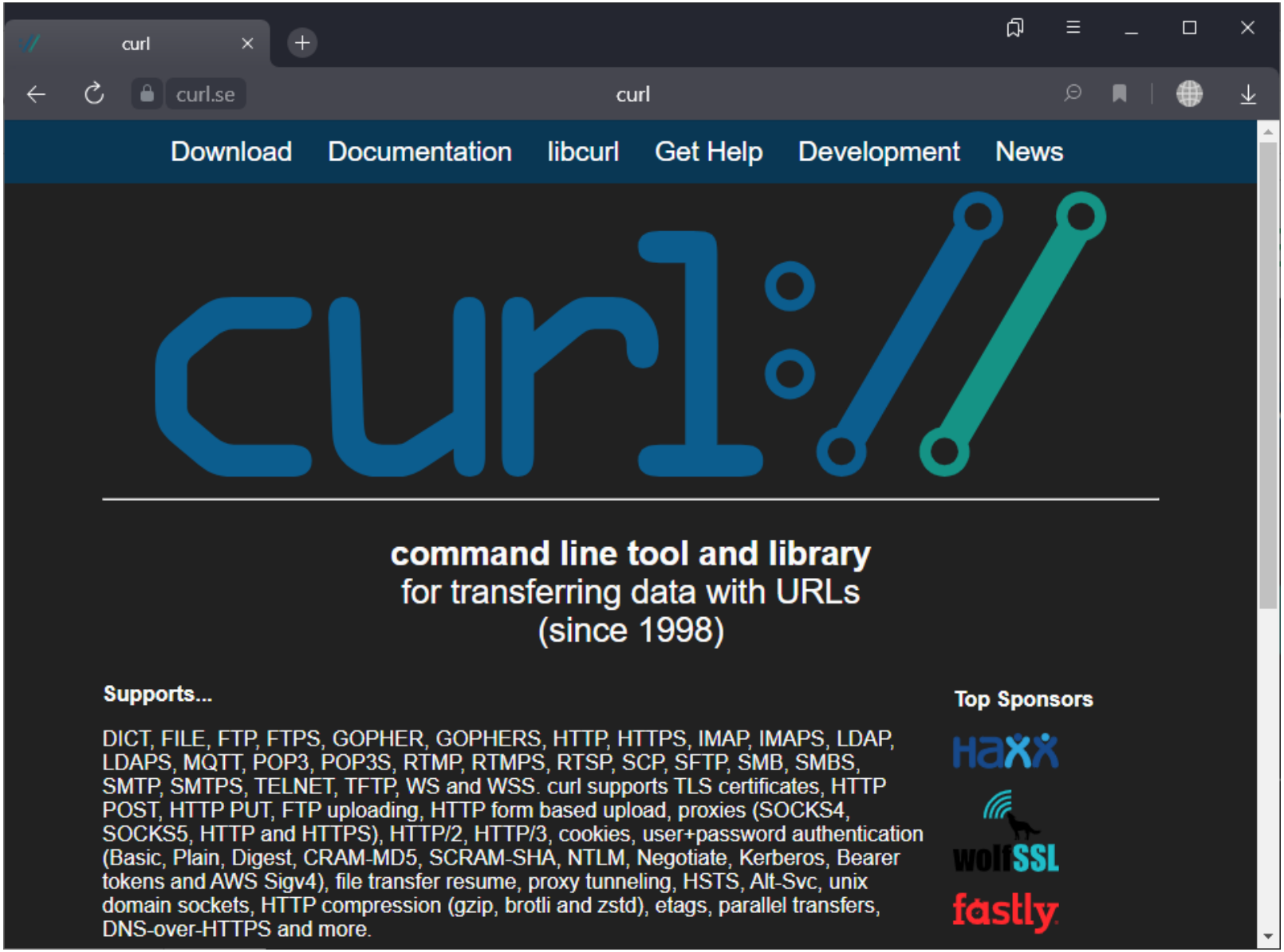


Чтение и отправка в поток по частям


```
let http = require("http");
let fs = require("fs");
new http.Server((req, res)=>{
    if(req.url=="/")
        sendFile(new fs.ReadStream("index.html"), res);
}).listen(3000);
function sendFile(file, res){
    file.on("readable", write);
    function write() {
        let data = file.read();
        if(data && !res.write(data)) {
            file.removeListener("readable", write);
            res.once("drain", ()=>{
                file.on("readable", write);
                write();
            });
        }
    }
    file.on("end", () => res.end());
}
```

file.pipe(res);

Имитация медленного клиента

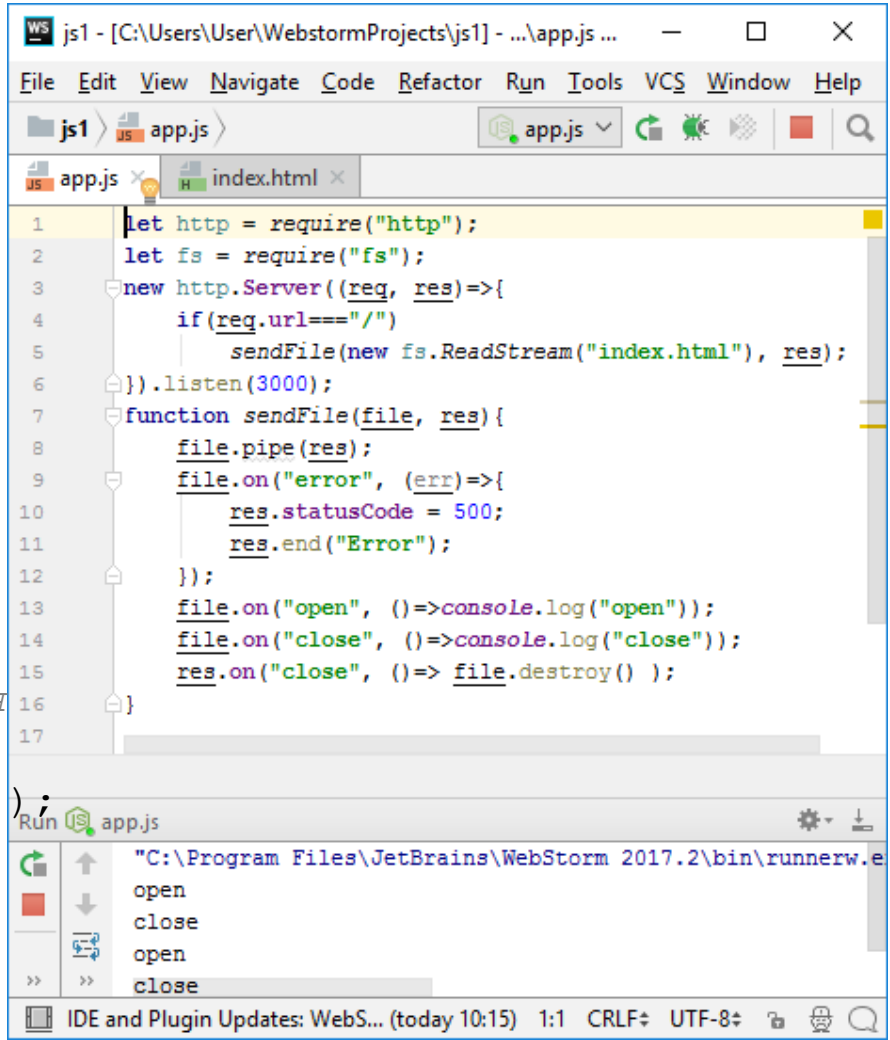


https://curl.se/

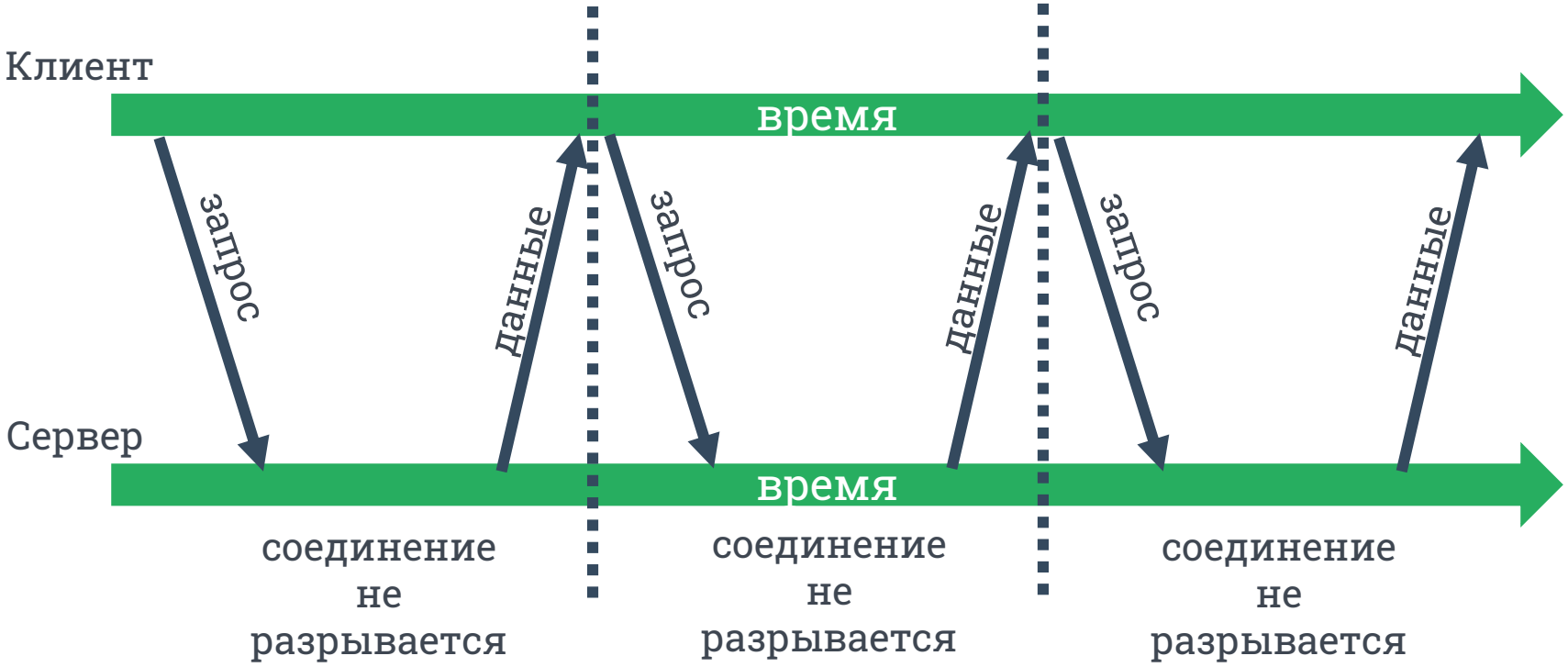
curl --limit 1k http://localhost:3000

Передача данных медленному клиенту

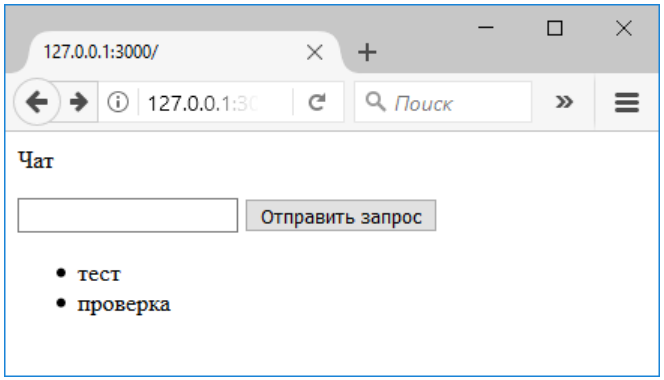

```
let http = require("http");
let fs = require("fs");
new http.Server((req, res)=>{
    if(req.url==="/")
        sendFile(new fs.ReadStream("index.html"), res);
}).listen(3000);
function sendFile(file, res){
    file.pipe(res);
    file.on("error", (err)=>{
        res.statusCode = 500;
        res.end("Error");
    });
    file.on("open", ()=>
        console.log("open")
    );
    file.on("close", ()=>
        console.log("close")
    );
    // Закрытие соединения до окончания
    // передачи файла
    res.on("close", ()=> file.destroy());
}
```

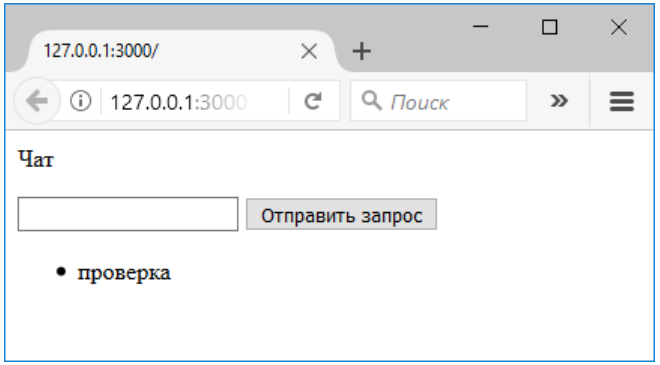


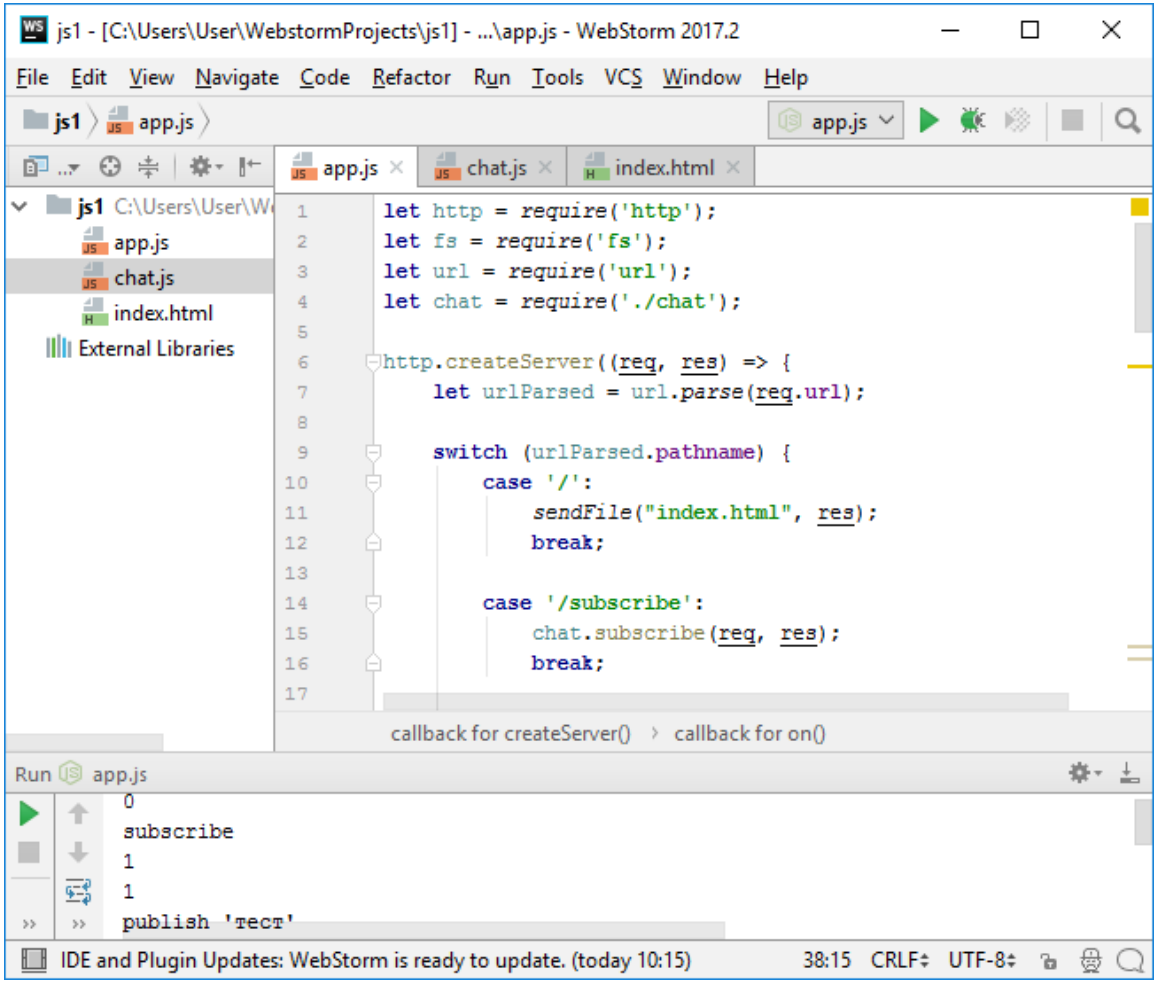
Long-polling («длинные запросы»)



Чат через «длинные запросы»



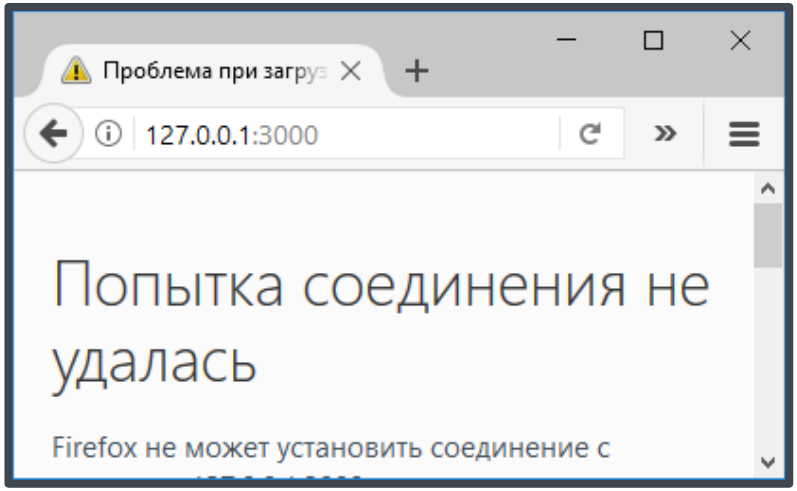


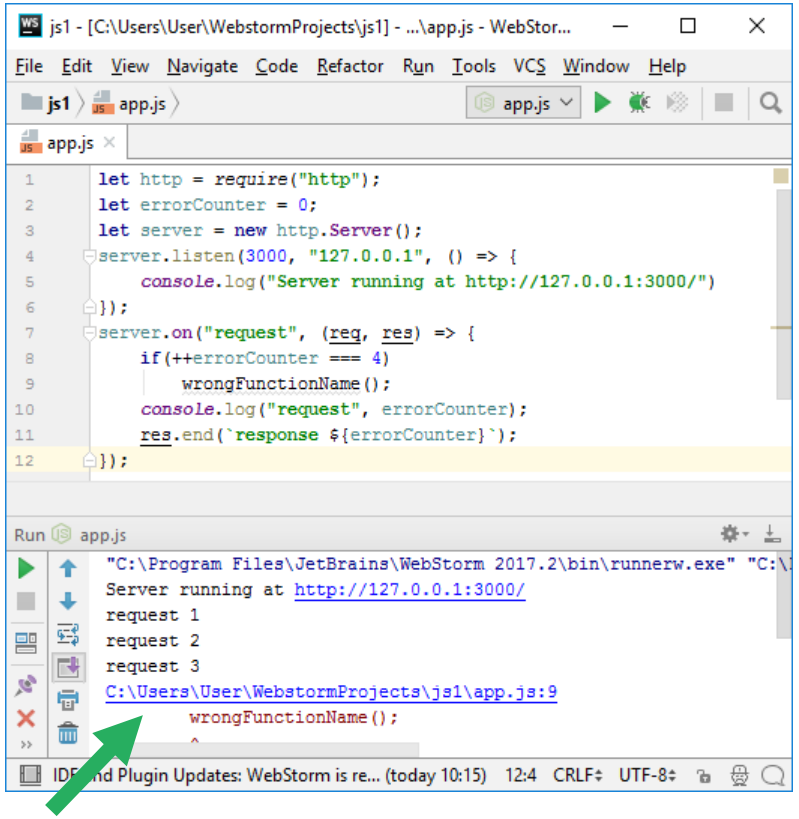


№27 «Чат через long-polling, чтение POST»
http://learn.javascript.ru/screencast/nodejs

Домены (1)


```
let http = require("http");
let errorCounter = 0;
let server = new http.Server();
server.listen(3000, "127.0.0.1", () => {
    console.log("Server running at http://127.0.0.1:3000/")
});
server.on("request", (req, res) => {
    if(++errorCounter === 4)
        wrongFunctionName();
    console.log("request", errorCounter);
    res.end(`response ${errorCounter}`);
});
```



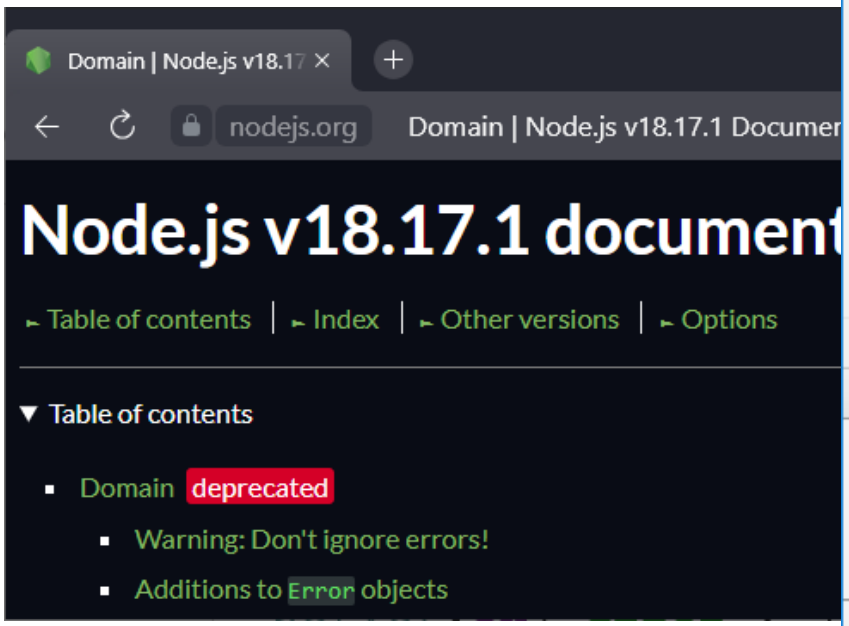


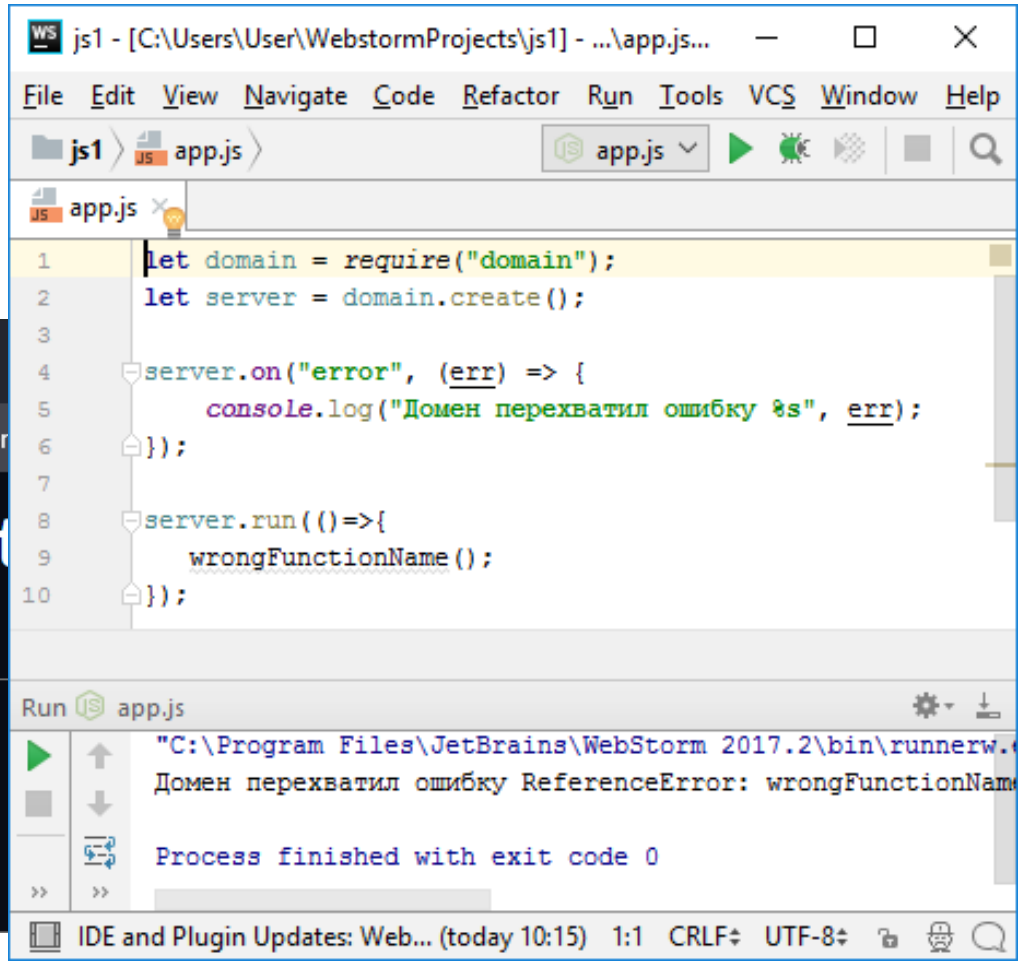
https://nodejs.org/en/docs/guides/domain-postmortem/
https://nodejs.org/dist/latest-v18.x/docs/api/domain.html

Домены (2)

benchmark
22,000 requests/second
↓
17,000 requests/second

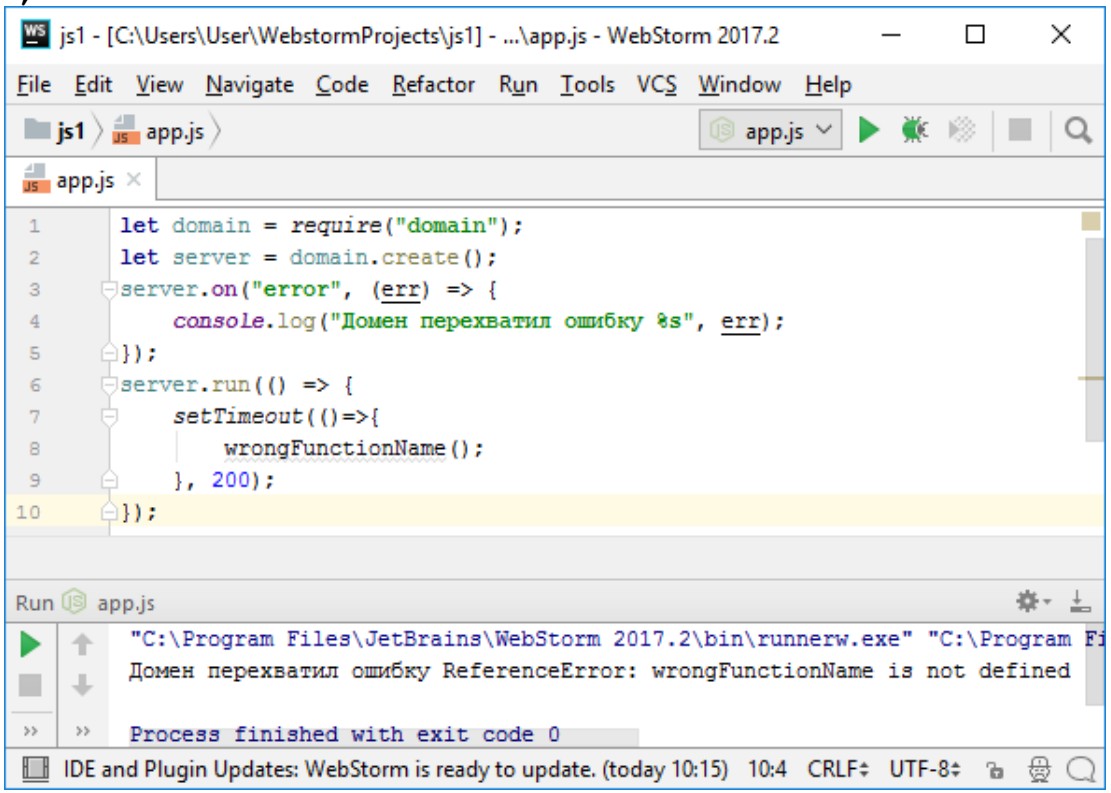
```
let domain = require("domain");
let server = domain.create();

server.on("error", (err) => {
    console.log("Домен перехватил ошибку %s", err);
});

server.run(()=>{
   wrongFunctionName();
});
```





Домены (3)

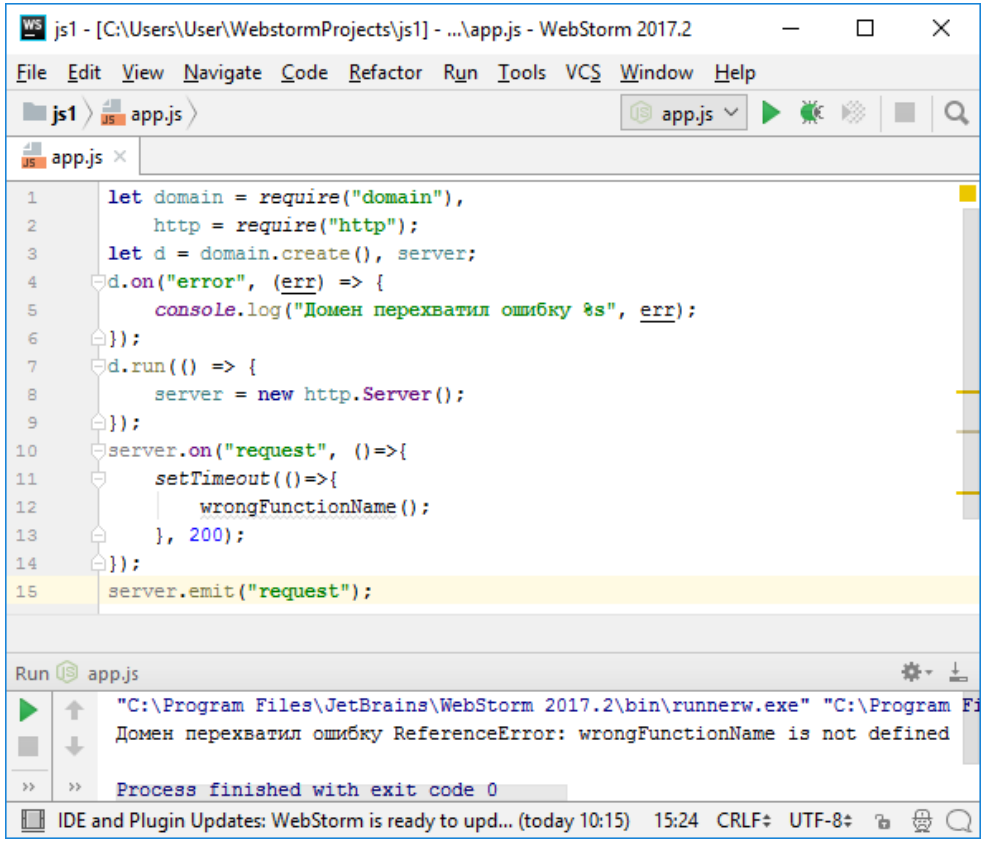

```
let domain = require("domain");
let server = domain.create();
server.on("error", (err) => {
    console.log("Домен перехватил ошибку %s", err);
});
server.run(() => {
    setTimeout(()=>{
        wrongFunctionName();
    }, 200);
});
```



Домены (4)

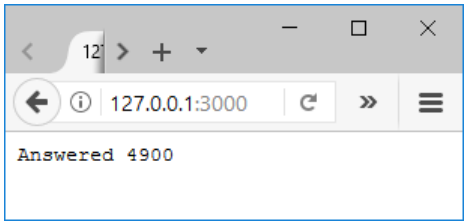

```
let domain = require("domain"),
    http = require("http");
let d = domain.create(), server;
d.on("error", (err) => {
    console.log("Домен перехватил ошибку %s", err);
});
d.run(() => {
    server = new http.Server();
});
server.on("request", ()=>{
    setTimeout(()=>{
        wrongFunctionName();
    }, 200);
});
server.emit("request");
```

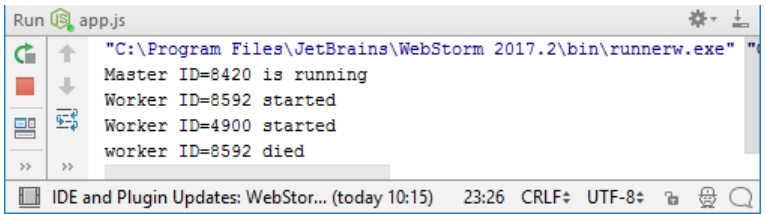
Попробуйте создать экземпляр Server вне run()



Использование нескольких процессоров

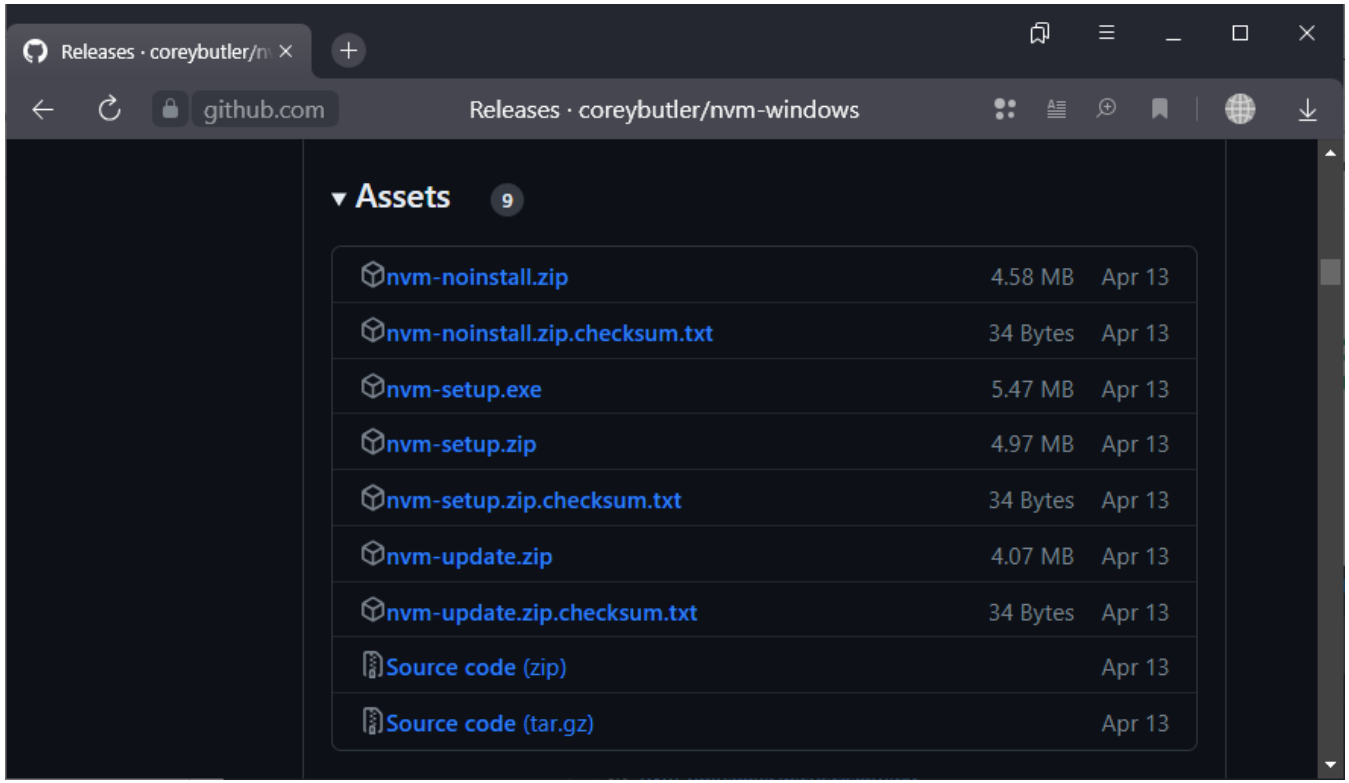

```
const cluster = require('cluster');
const http = require('http');
const numCPUs = require('os').cpus().length;
const list = [];

if (cluster.isMaster) {
    console.log(`Master ID=${process.pid} is running`);
    // Fork workers
    for (let i = 0; i < numCPUs; i++)
        list.push(cluster.fork());
    cluster.on('exit', (worker, code, signal) => {
        console.log(`worker ID=${worker.process.pid} died`);
    });
} else {
    // Worker могут разделять любые TCP-соединения
    http.createServer((req, res) => {
        res.writeHead(200);
        res.end(`Answered ${process.pid}`);
    }).listen(3000);
    console.log(`Worker ID=${process.pid} started`);
}
if(list[0])
    list[0].disconnect();
```





Node Version Manager
Поддержка разных версий Node.JS

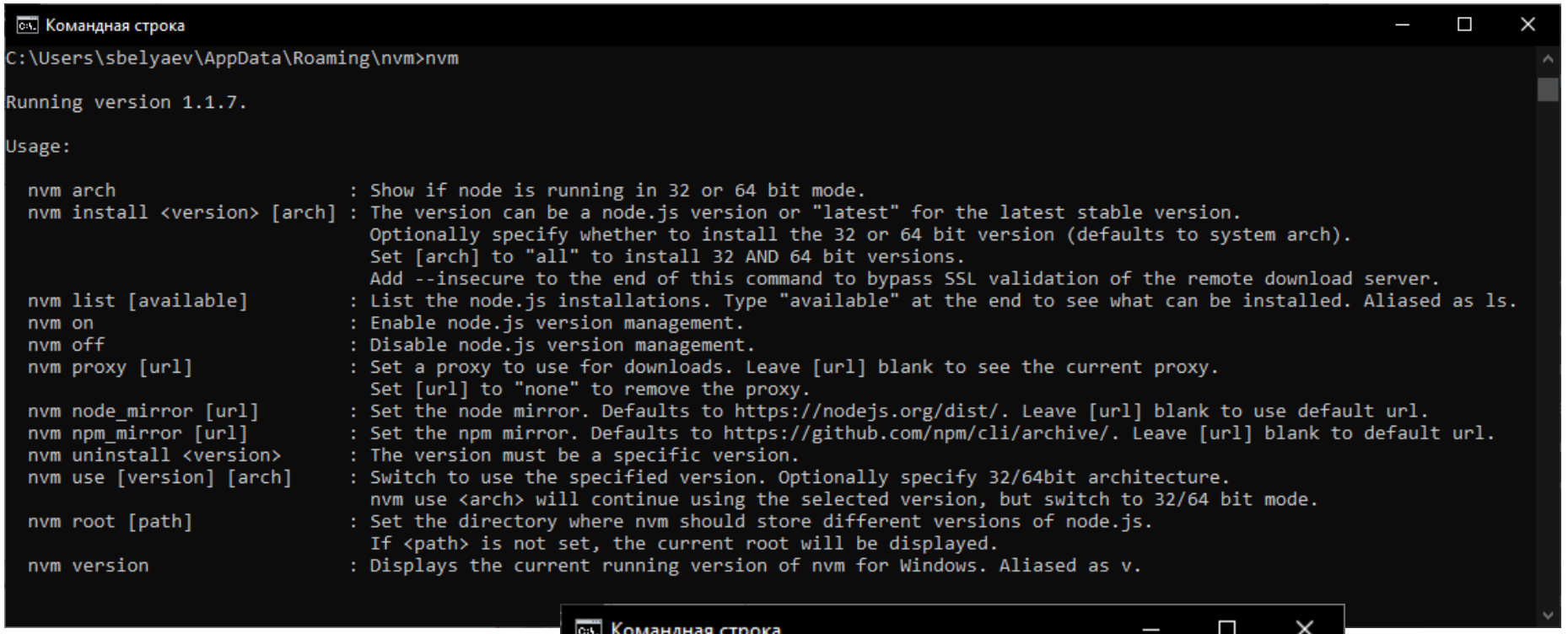
Windows
https://github.com/coreybutler/nvm-windows/releases



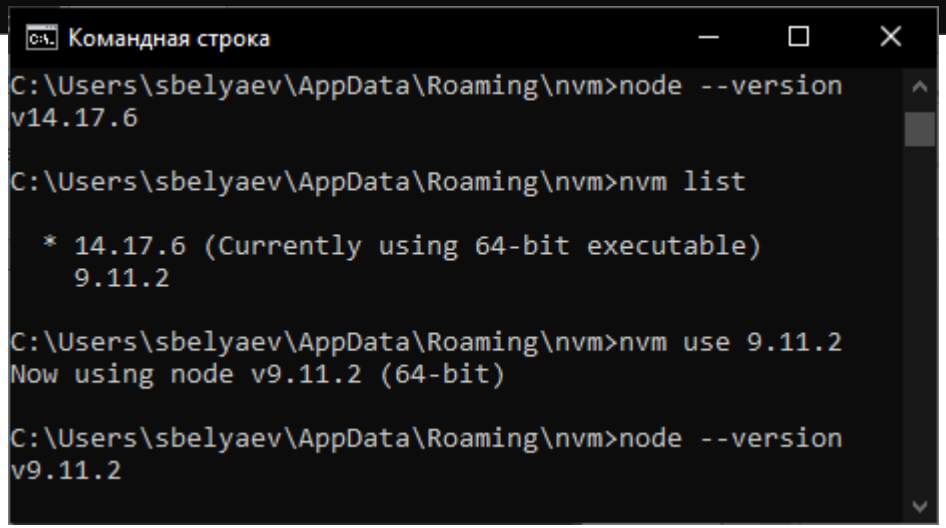
Linux
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.37.2/install.sh | bash

src
https://github.com/nvm-sh/nvm

nvm – использование



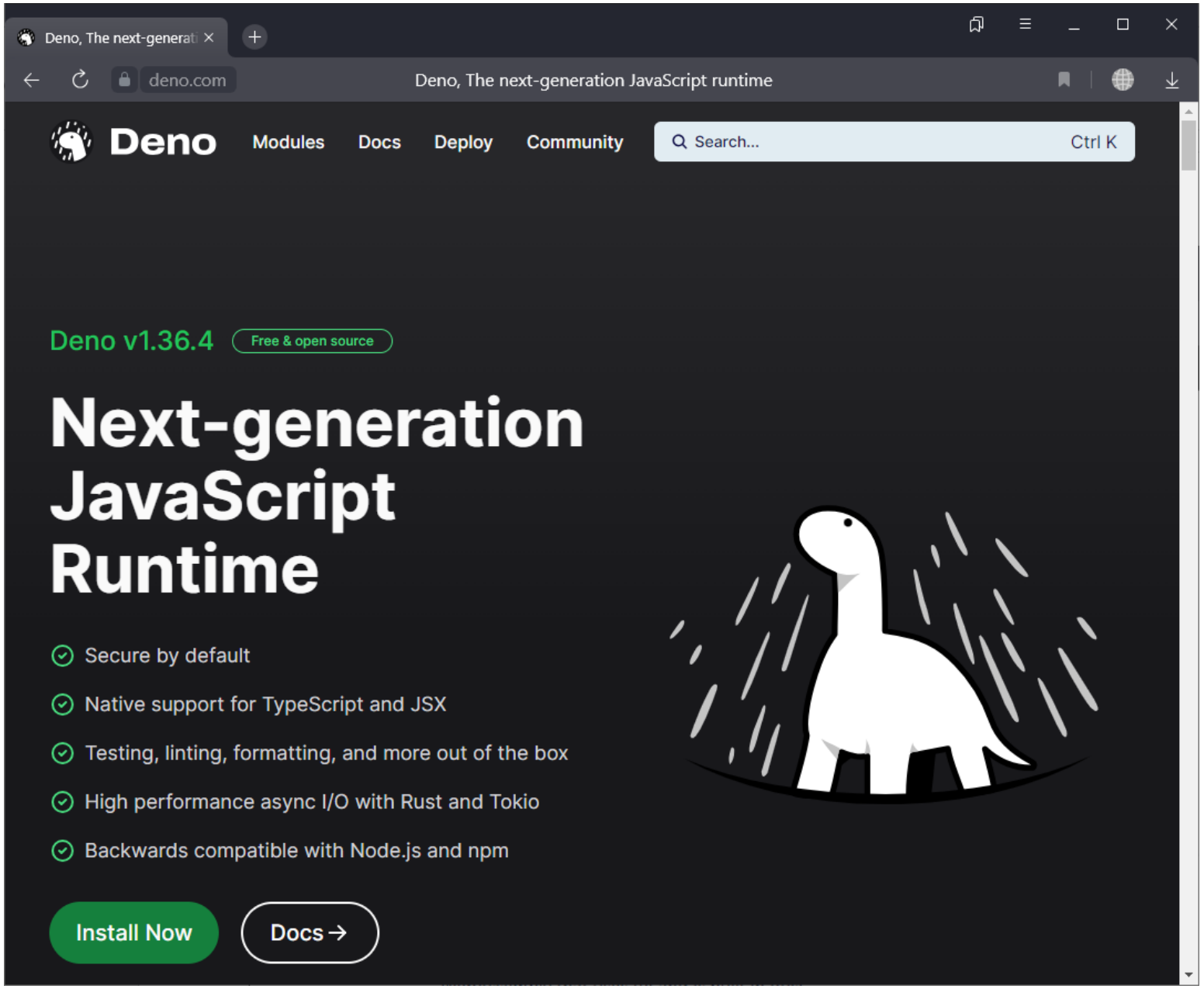
nvm install 14.17.6
nvm list
nvm use 14.17.6



10 вещей, о которых я сожалею в Node.js **Ryan Dahl**

- Отказались от Promise
  - Были удалены в феврале 2010
- Безопасность
  - Доступ к диску без ограничений
- Система сборки
  - Сложности с разработкой на C
- package.json
  - npm в стандартной поставке, централизованный, лишняя абстракция, лишняя информация
- node_modules
  - Сложный алгоритм, проблемы с доступом из браузера
- require без расширения
- Единый index.js

https://deno.land/



Deno

https://github.com/denoland/deno

- Безопасность
  - По умолчанию без права работы с файловой системой
- Компилятор TypeScript
  - Поддержка TypeScript «из коробки»
- Упрощение системы модулей
  - Не совместимо с модулями Node
  - Импорты относительные или абсолютные URL
  - Всегда с расширением (js, ts)
- Другое
  - Совместим с браузерами (`window`, а не `global`)
  - Поддерживает await верхнего уровня

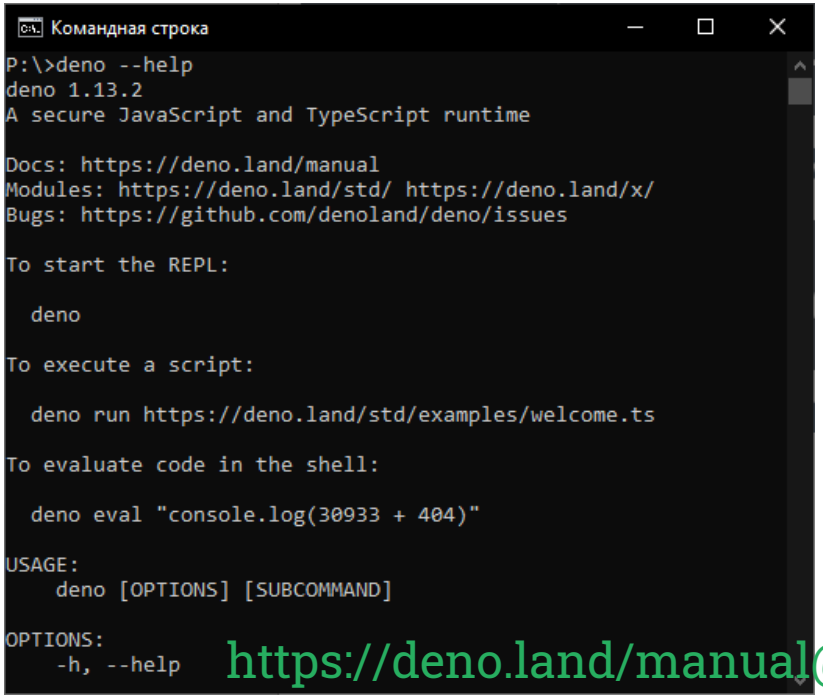
Установка Deno

Shell (Mac, Linux):

- curl -fsSL https://deno.land/x/install/install.sh | sh

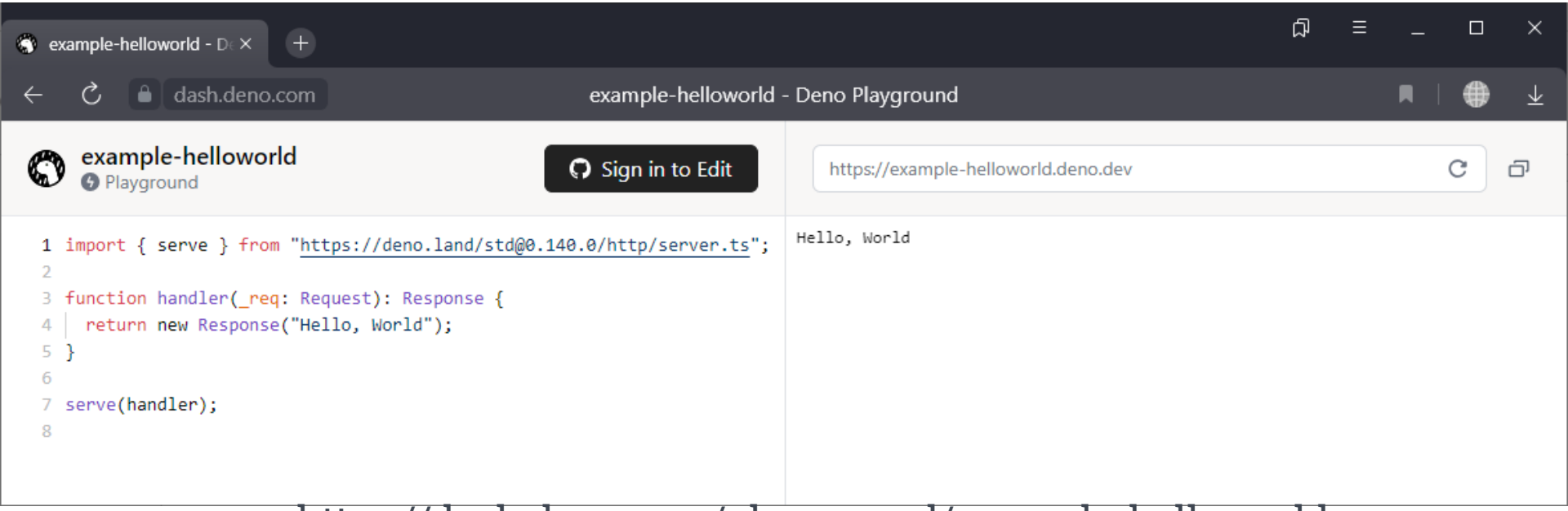
PowerShell (Windows):

- irm https://deno.land/install.ps1 | iex



https://deno.land/manual@v1.31.1/getting_started/installation

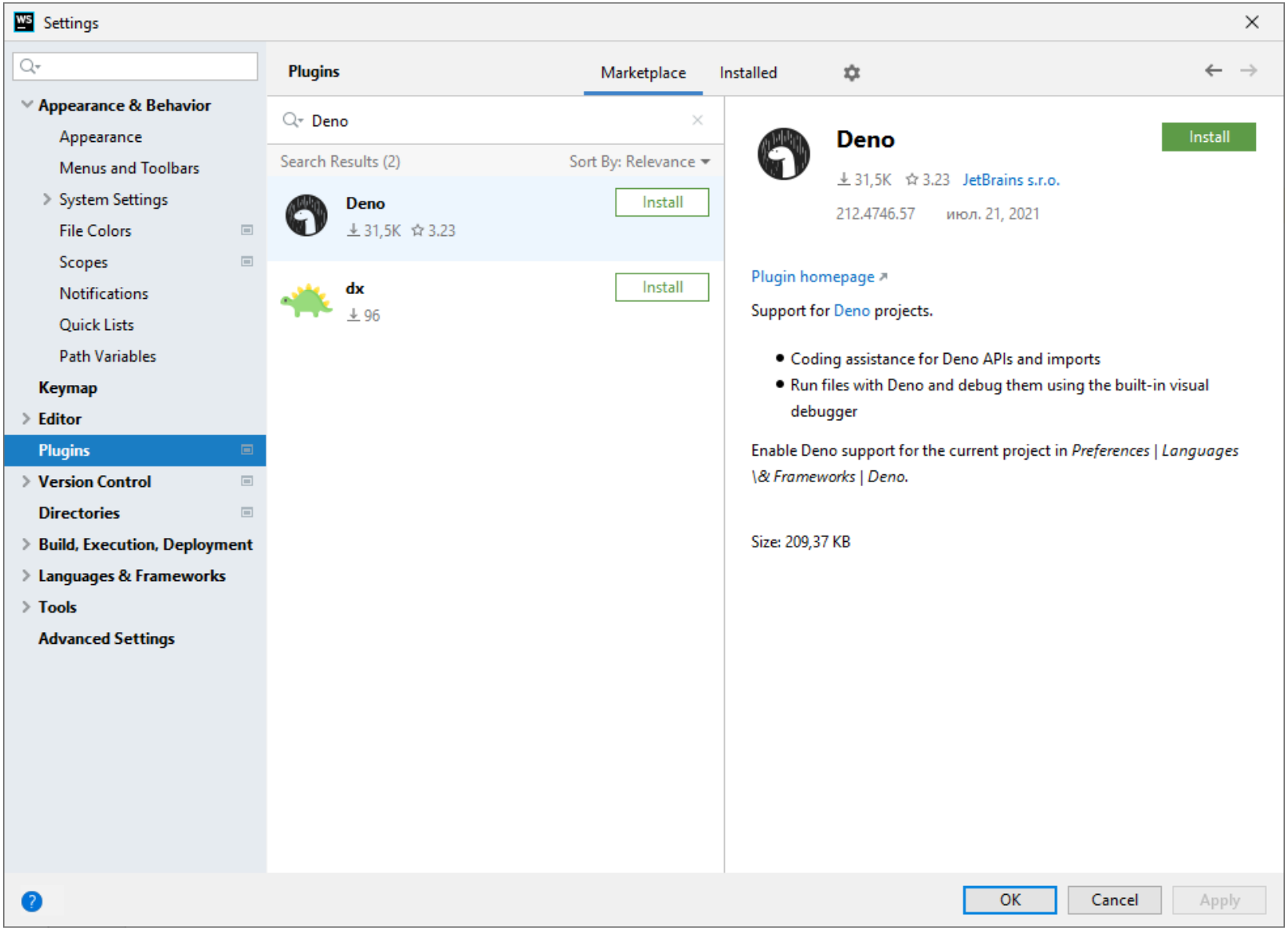
Пример использования Deno



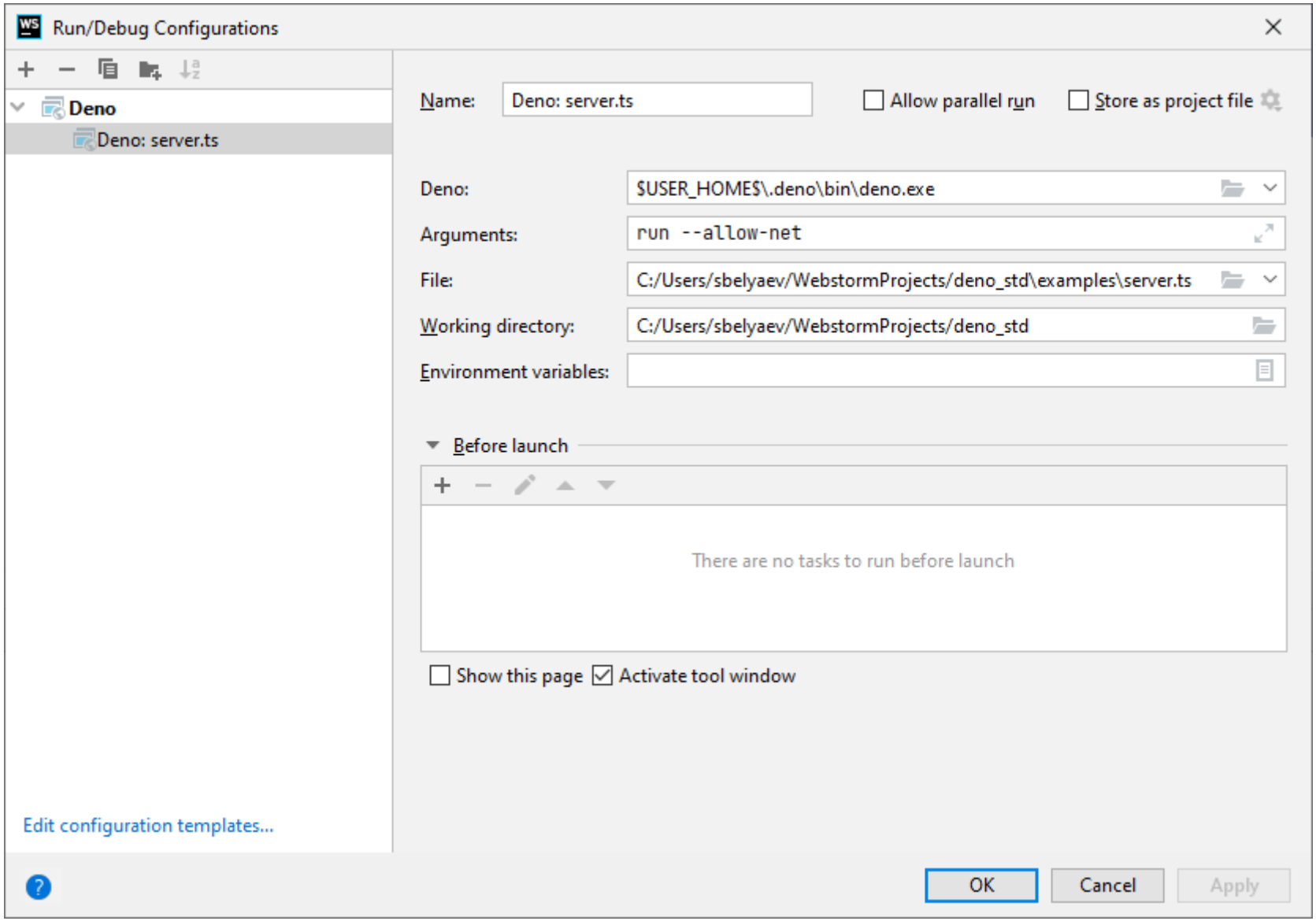
https://dash.deno.com/playground/example-helloworld

https://deno.com/deploy/docs/examples

Установка Deno в WebStorm

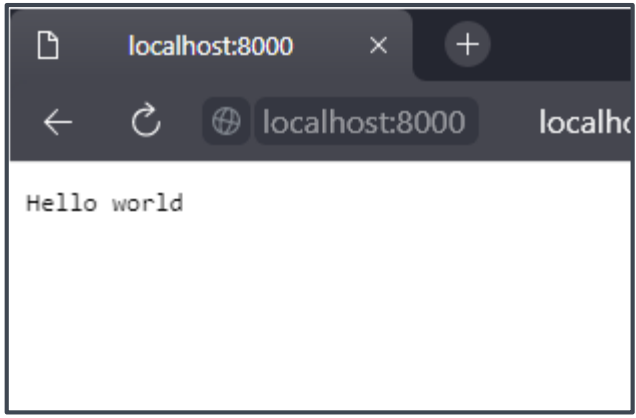


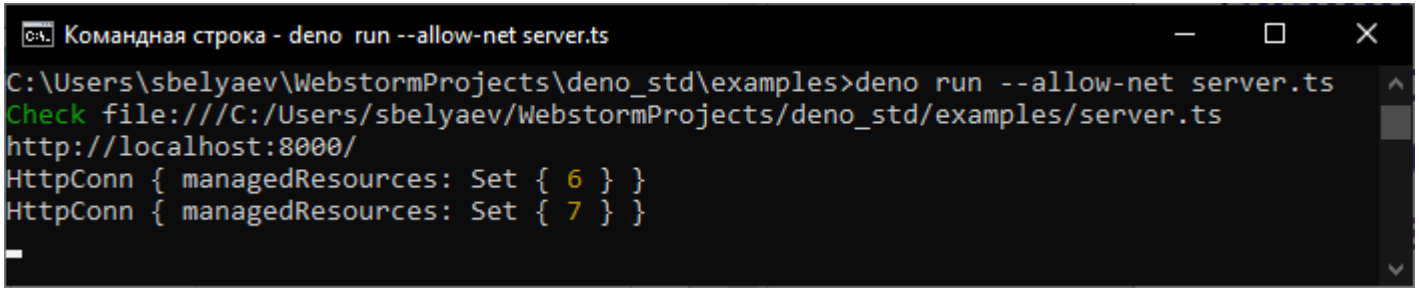
Настройка конфигурации



Пример программы (1)


```
const listener = Deno.listen({ port: 8000 });
console.log("http://localhost:8000/");
for await (const conn of listener) {
  (async () => {
    const requests = Deno.serveHttp(conn);
    for await (const { respondWith } of requests) {
      console.log(requests);
      respondWith(new Response("Hello world"));
    }
  })();
}
```





Пример программы (2)


```
// Не забудьте параметр --allow-net
const url = Deno.args[0] // Чтение параметров командной строки
const res = await fetch(url) // Чтение URL
const body = new Uint8Array(await res.arrayBuffer()) // Получение
await Deno.stdout.write(body) // Вывод в консоль
```

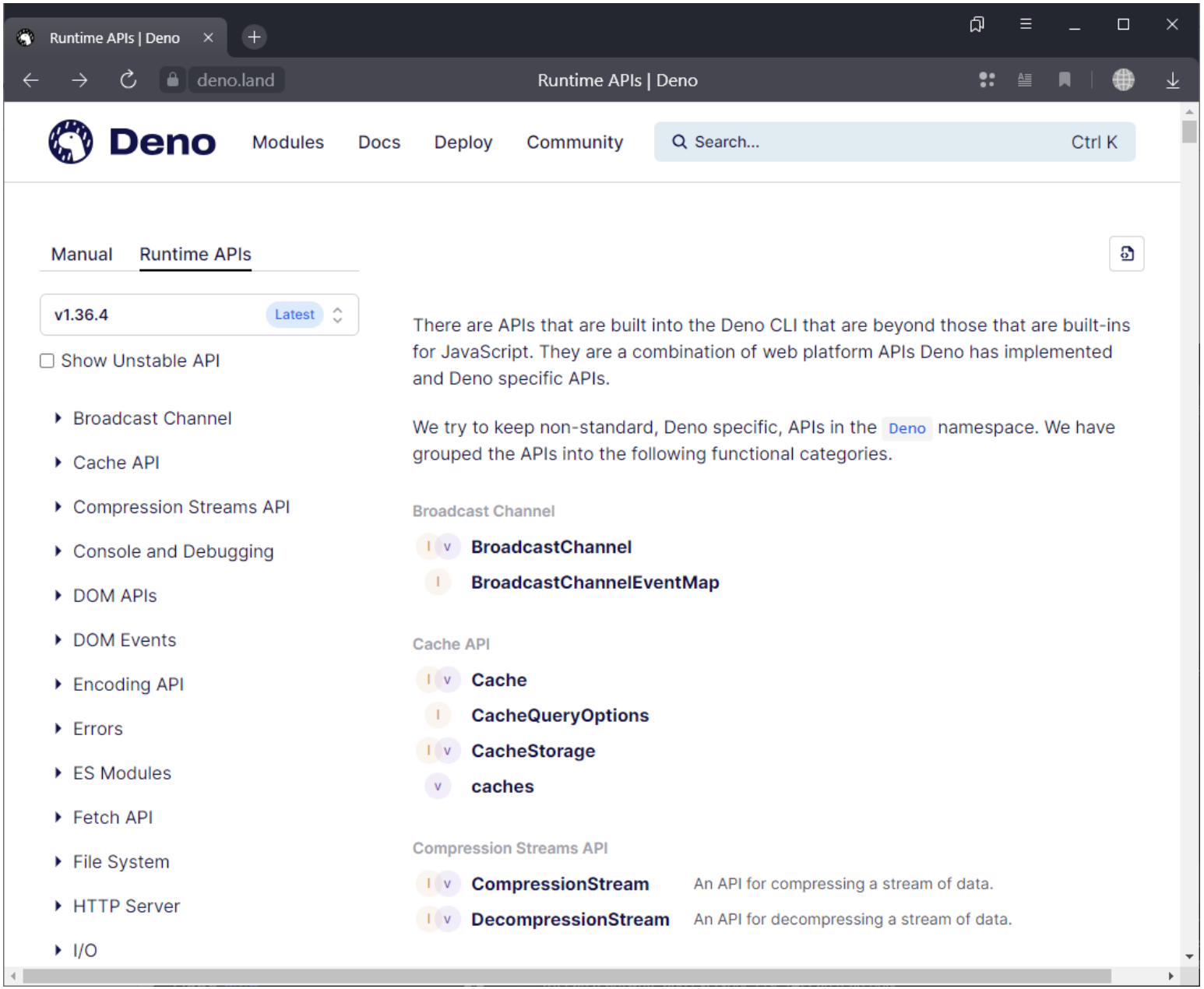
https://example.com

```
<!doctype html>
<html>
<head>
   <title>Example Domain</title>

   <meta charset="utf-8" />
   <meta http-equiv="Content-type" content="text/html; charset=utf-8" />
   <meta name="viewport" content="width=device-width, initial-scale=1" />
   <style type="text/css">
   body {
...
```

Варианты запуска

deno run main.ts
deno run https://mydomain.com/main.ts
cat main.ts | deno run -

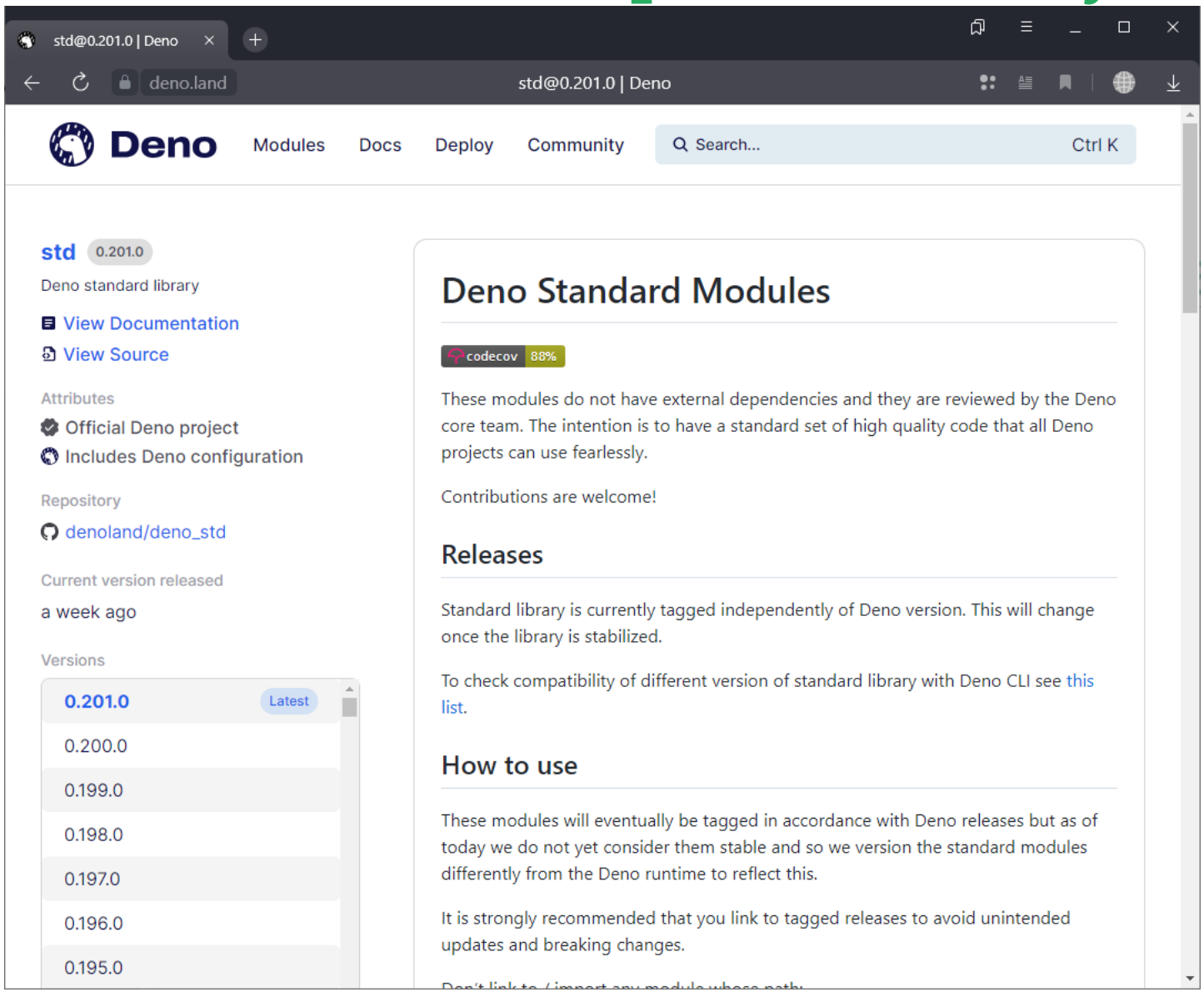
https://deno.land/api@v1.36.4#Deno



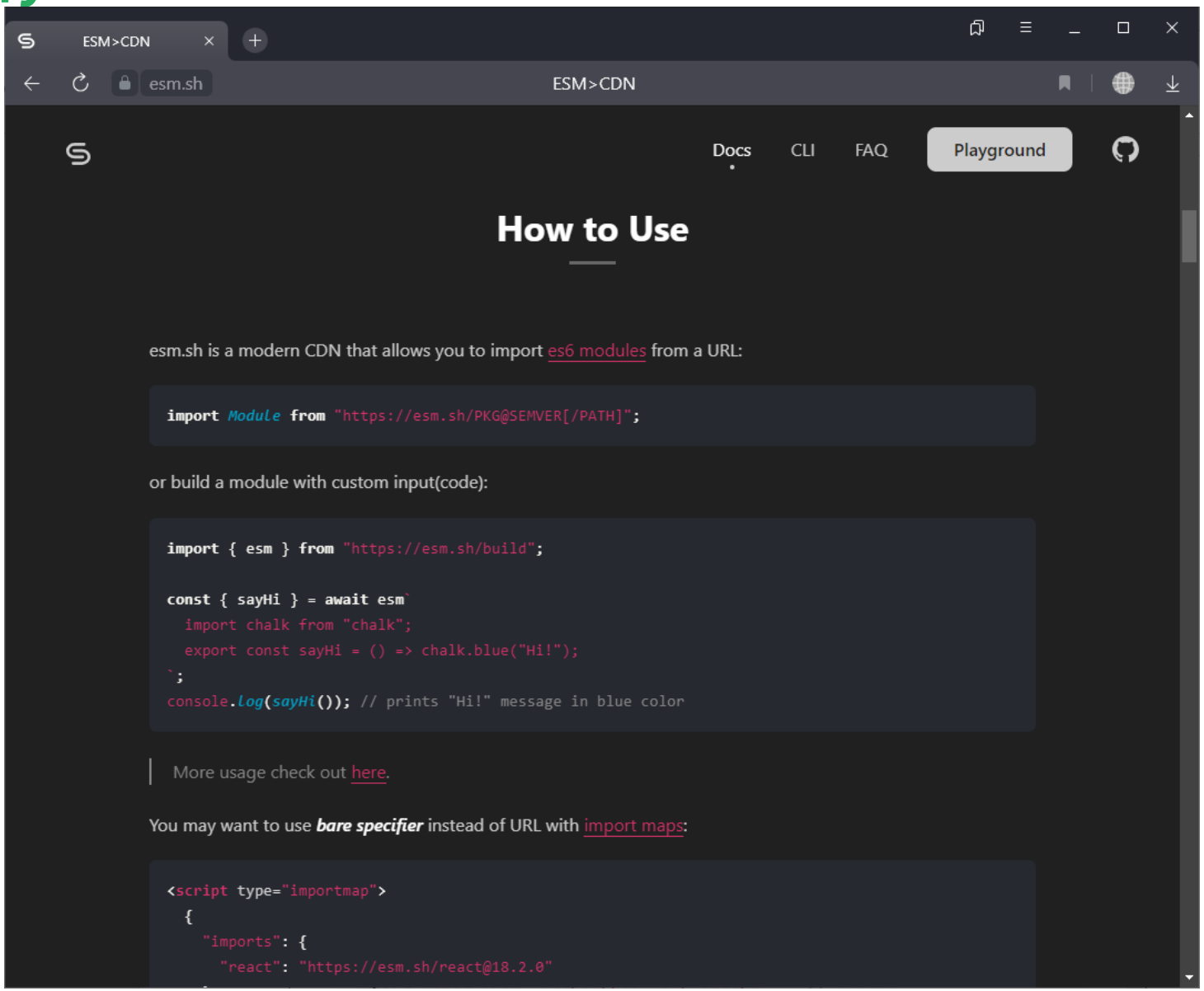
Разрешения Deno

- **--allow-env**=<allow-env>
  - Разрешение устанавливать значения переменным среды
- **--allow-hrtime**
  - Разрешение на точное вычисление времени
- **--allow-net**=<allow-net>
  - Разрешение на доступ к сети
- **--allow-ffi**
  - Разрешение на использование библиотек
- --**allow-read**=<allow-read>
  - Разрешение на чтение файлов ОС
- **--allow-run**=<allow-run>
  - Разрешение на запуск подпроцессов
- **--allow-write**=<allow-write>
  - Разрешение на запись файлов
- **-A, --allow-all**
  - Все разрешения

https://deno.land/std@0.201.0 – библиотека стандартных модулей



https://esm.sh/ - использование модулей NPM



Вопросы для самопроверки

- Где поддерживаются модули (браузер, сервер)?
- В чем отличие CommonJS и ES6 модулей?
- Для чего нужны и чем отличаются npm, yarn, npx?
- В чем отличия в применении модулей debug и winston?
- Для чего нужны rollbar и sentry?
- Чем отличаются setTimeout, setImmediate, setInterval?
- Для чего используются domain и почему они deprecated?
- Как можно использовать несколько процессов в JS?
- Для чего нужен nvm?
- В чем ключевые отличия Deno от NodeJS?