



# Web-технологии

Тестирование web-приложений

# Содержание

- Понятие TDD-разработки
- Модульные тесты
  - **Assert** – встроенный в **Node.js** модуль для тестирования
    - применение принципов TDD
  - **Should** – библиотека утверждений
  - **Chai** – библиотека утверждений
  - **Mocha** – фреймворк для тестирования
  - **Jest** – фреймворк для тестирования
- **Selenium, Selenium IDE, Protractor**
- Использование **headless** браузеров
- **Postman**

<https://shouldjs.github.io/>

<https://www.chaijs.com/>

<https://mochajs.org/>

<https://jestjs.io/ru/>

<https://www.selenium.dev/>

<https://protractor.angular.io/>

<https://developers.google.com/web/tools/puppeteer/>

# Модульное тестирование

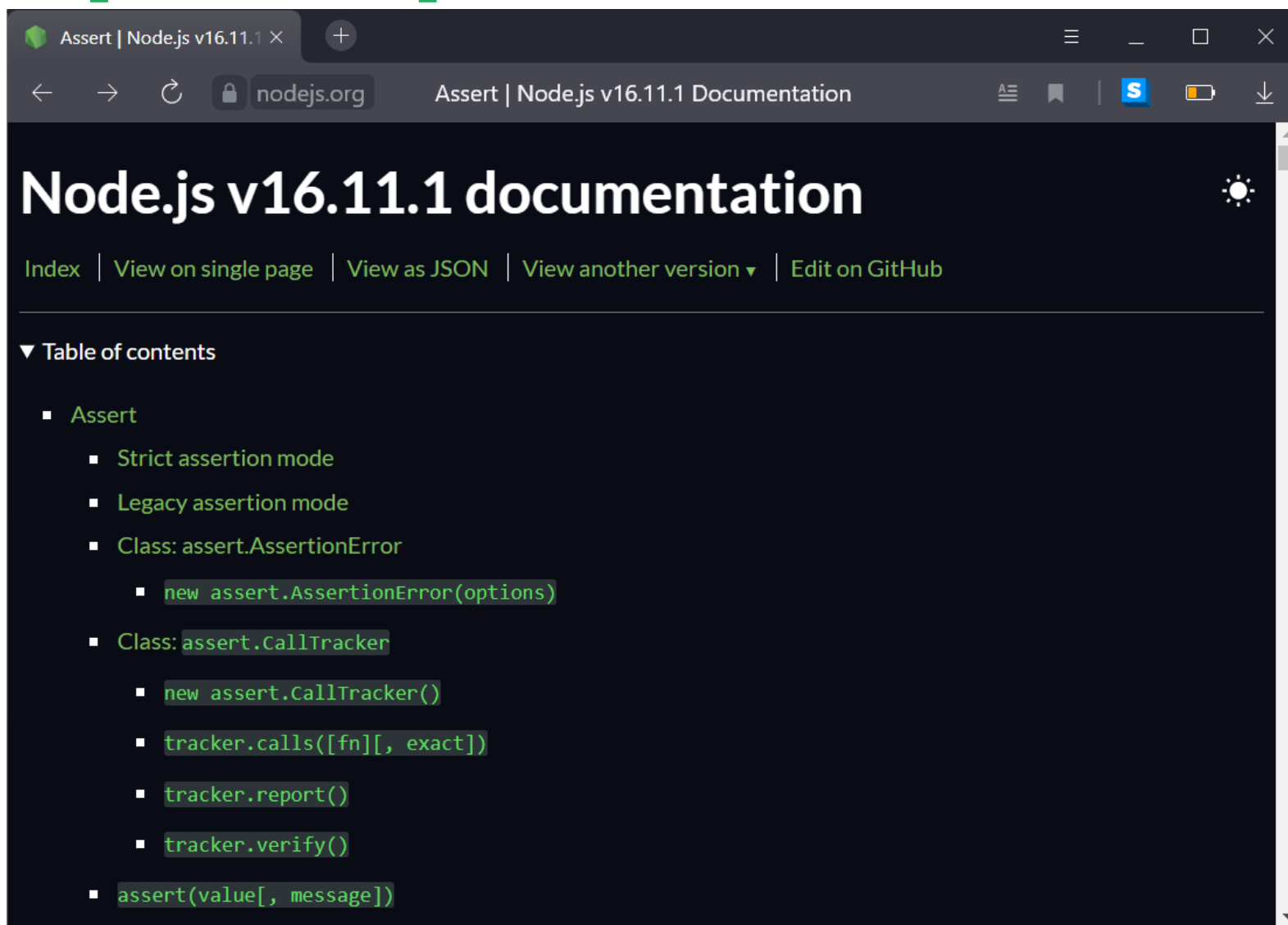
- Проверка «модуля/единицы» исходного кода
- Преимущества
  - возможность в дальнейшем вносить изменения
  - упрощение интеграции модулей
  - документирование кода
  - отделение интерфейса от реализации
- Ограничения (не работает)
  - сложный код
  - результат известен приблизительно
  - ошибки интеграции
  - низкая культура программирования
  - проблемы с заглушками

# TDD – test-driven development

- Разработка на основе тестирования
- Разработка, управляемая тестированием



# Assert – функции утверждения для проверки инвариантов



The screenshot shows a web browser displaying the Node.js v16.11.1 documentation for the Assert module. The page title is "Node.js v16.11.1 documentation". The navigation bar includes links for "Index", "View on single page", "View as JSON", "View another version", and "Edit on GitHub". The "Table of contents" section is expanded, showing the following items:

- Assert
  - Strict assertion mode
  - Legacy assertion mode
  - Class: `assert.AssertionError`
    - `new assert.AssertionError(options)`
  - Class: `assert.CallTracker`
    - `new assert.CallTracker()`
    - `tracker.calls([fn][, exact])`
    - `tracker.report()`
    - `tracker.verify()`
  - `assert(value[, message])`

<https://nodejs.org/api/assert.html>

# Assert / модуль пустой, тест готов

6

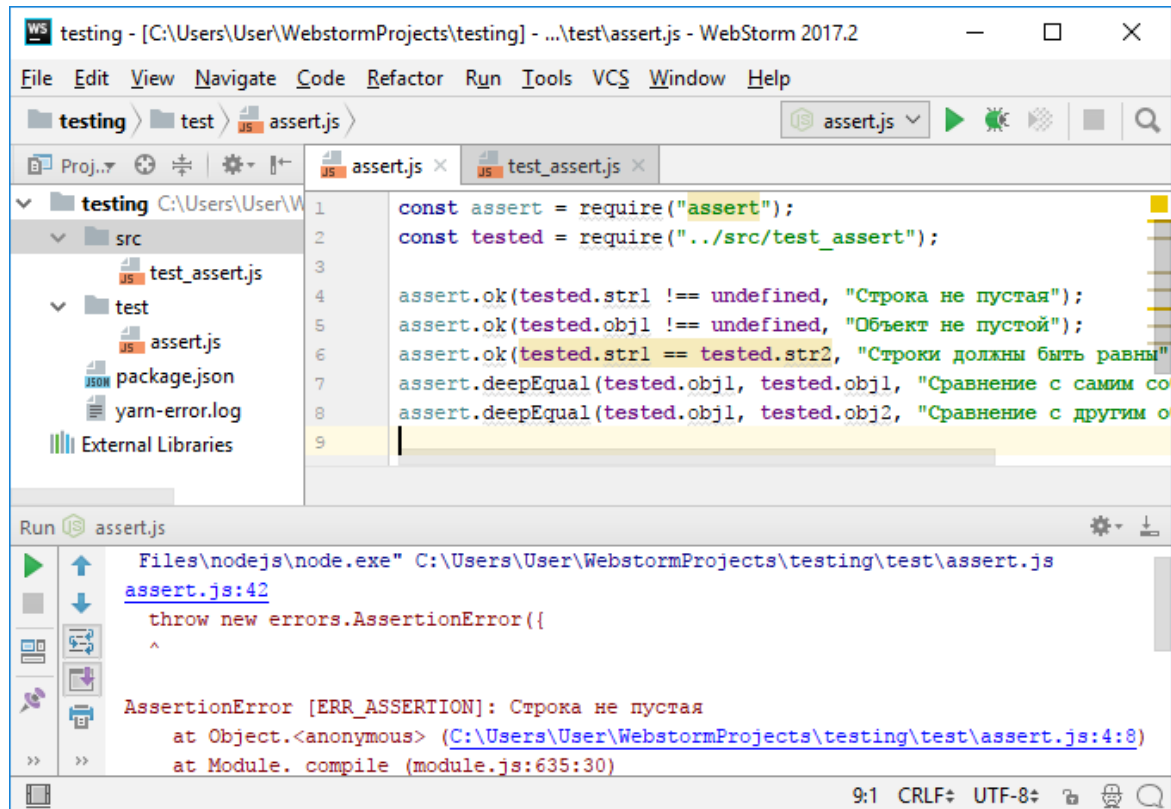
## assert.js

```
const assert = require("assert");
const tested = require("../src/test_assert1");

assert.ok(tested.str1 !== undefined, "Строка не пустая");
assert.ok(tested.obj1 !== undefined, "Объект не пустой");
assert.ok(tested.str1 === tested.str2, "Строки должны быть равны");
assert.deepEqual(tested.obj1, tested.obj1, "Сравнение с самим собой");
assert.deepEqual(tested.obj1, tested.obj2, "Сравнение с другим объектом");
```

## test\_assert1.js

```
module.exports = {
}
```



# Assert / str1 != str2

7

## assert.js

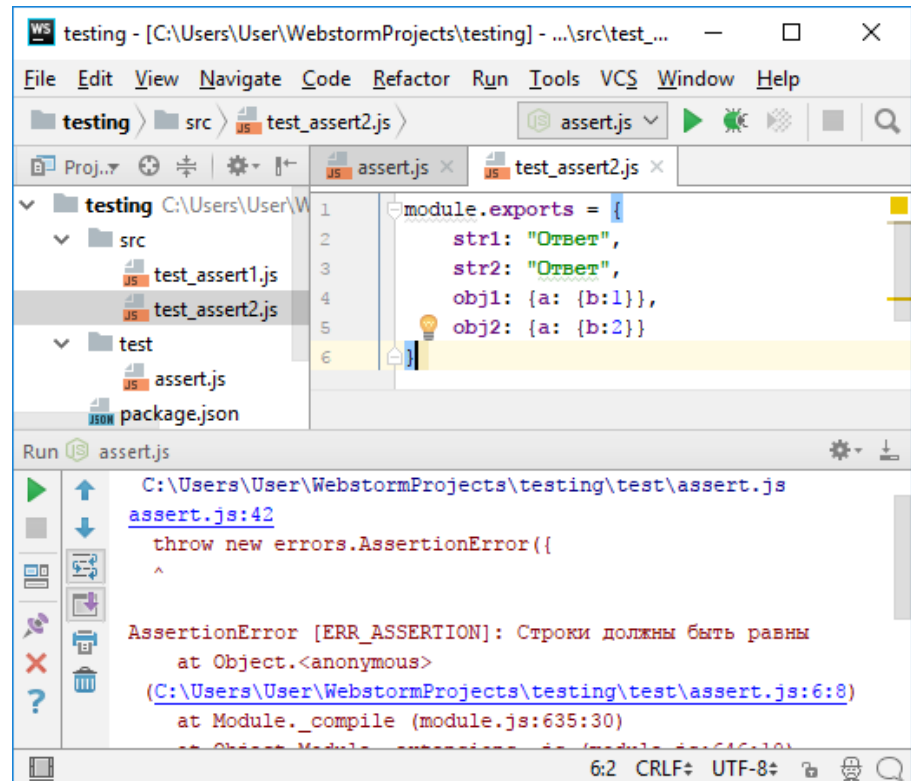
```
const assert = require("assert");  
const tested = require("../src/test_assert2");
```

```
assert.ok(tested.str1 !== undefined, "Строка не пустая");  
assert.ok(tested.obj1 !== undefined, "Объект не пустой");  
assert.ok(tested.str1 == tested.str2, "Строки должны быть равны");  
assert.deepEqual(tested.obj1, tested.obj1, "Сравнение с самим собой");  
assert.deepEqual(tested.obj1, tested.obj2, "Сравнение с другим объектом");
```

## test\_assert2.js

```
module.exports = {  
  str1: "Ответ",  
  str2: "Ответ",  
  obj1: {a: {b:1}},  
  obj2: {a: {b:2}}  
}
```

- Почему не равны?
- Мало ли какой у меня план...



# Assert / obj1 != obj2

8

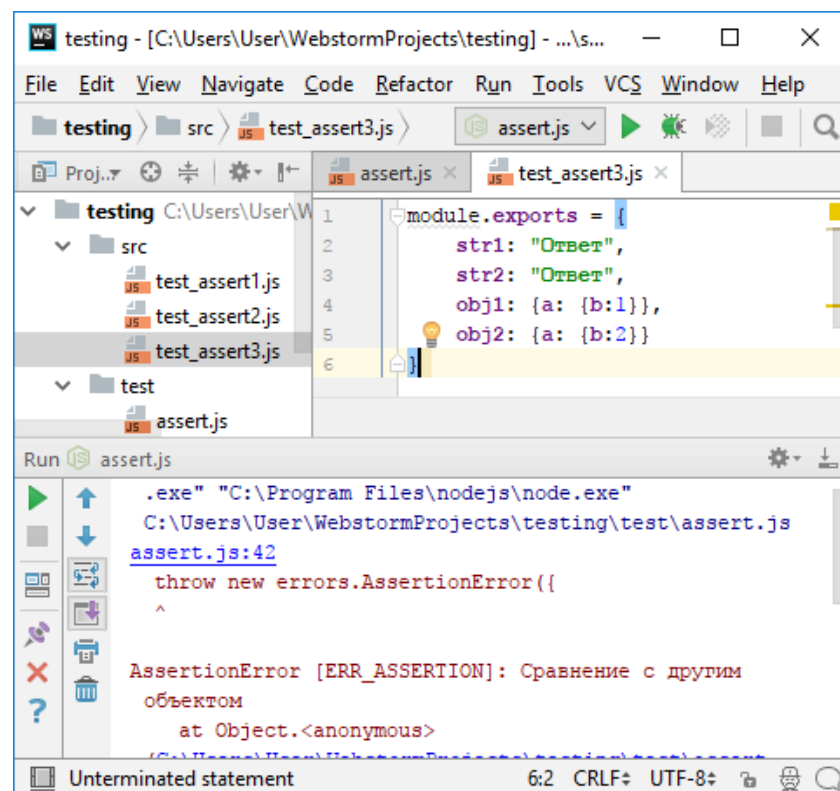
## assert.js

```
const assert = require("assert");  
const tested = require("../src/test_assert3");
```

```
assert.ok(tested.str1 !== undefined, "Строка не пустая");  
assert.ok(tested.obj1 !== undefined, "Объект не пустой");  
assert.ok(tested.str1 === tested.str2, "Строки должны быть равны");  
assert.deepEqual(tested.obj1, tested.obj1, "Сравнение с самим собой");  
assert.deepEqual(tested.obj1, tested.obj2, "Сравнение с другим объектом");
```

## test\_assert3.js

```
module.exports = {  
  str1: "Ответ",  
  str2: "Ответ",  
  obj1: {a: {b:1}},  
  obj2: {a: {b:2}}  
}
```





# Assert / результат разработки

9

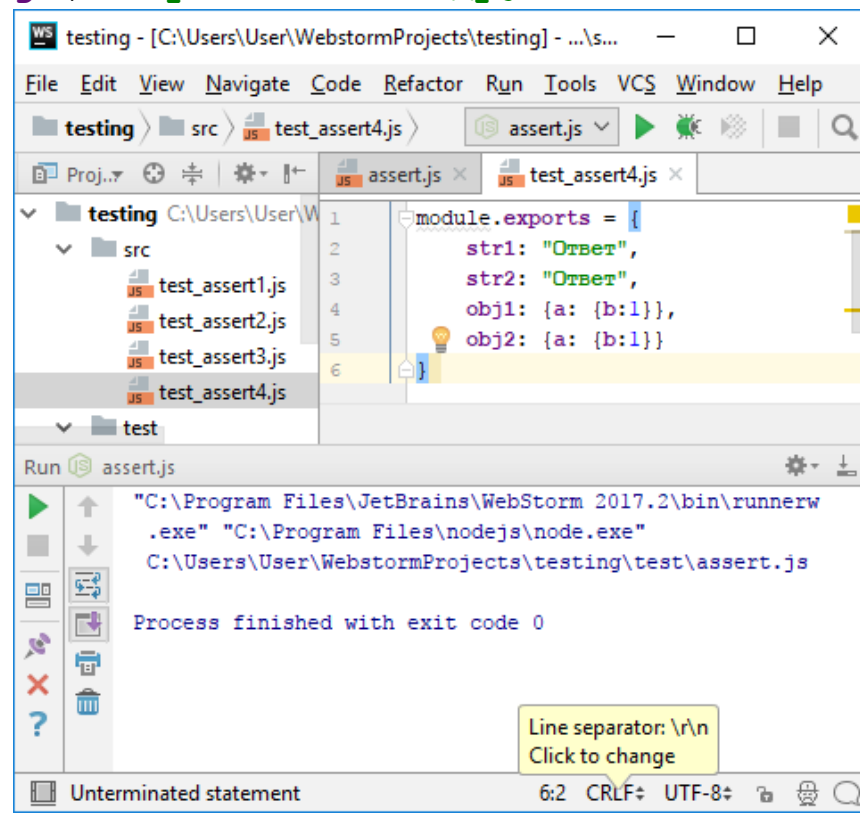
## assert.js

```
const assert = require("assert");  
const tested = require("../src/test_assert4");
```

```
assert.ok(tested.str1 !== undefined, "Строка не пустая");  
assert.ok(tested.obj1 !== undefined, "Объект не пустой");  
assert.ok(tested.str1 === tested.str2, "Строки должны быть равны");  
assert.deepEqual(tested.obj1, tested.obj1, "Сравнение с самим собой");  
assert.deepEqual(tested.obj1, tested.obj2, "Сравнение с другим объектом");
```

## test\_assert4.js

```
module.exports = {  
  str1: "Ответ",  
  str2: "Ответ",  
  obj1: {a: {b:1}},  
  obj2: {a: {b:1}}  
}
```



# Assert. Методы

- **assert**(value[, message]) – алиас для **assert.ok()**
- **assert.deepEqual**(actual, expected[, message])
- **assert.deepStrictEqual**(actual, expected[, message])
- **assert.doesNotThrow**(block[, error][, message])
- **assert.equal**(actual, expected[, message])
- **assert.fail**(message)
- **assert.fail**(actual, expected[, message[, operator[, stackStartFunction]]])
- **assert.ifError**(value)
- **assert.notDeepEqual**(actual, expected[, message])
- **assert.notDeepStrictEqual**(actual, expected[, message])
- **assert.notEqual**(actual, expected[, message])
- **assert.notStrictEqual**(actual, expected[, message])
- **assert.ok**(value[, message])
- **assert.strictEqual**(actual, expected[, message])
- **assert.throws**(block[, error][, message])

# Should.js – библиотека утверждений <sup>11</sup>

Should.js API Document x +

← → shouldjs.github.io Should.js API Documentation

## Should.js

**yarn add -D should**

### global

- `Assertion.add`
- `Assertion.addChain`
- `Assertion.alias`
- `PARAM_REGEXP`
- `PromisedAssertion`
- `TYPE_REGEXP`
- `format`
- `format`
- `parse`
- `root`
- `should$1`
- `should.Assertion`
- `should.AssertionEr`
- `should.config`
- `should.extend`
- `should.noConflict`
- `should.use`

### assertion

- `Assertion#any`

### “global” Members

#### ()

①

Detect free variable `self`.

#### Assertion.add(name, func)

①

Way to extend Assertion function. It uses some logic to define only positive assertions and itself rule with negative assertion. All actions happen in subcontext and this method take care about negation. Potentially we can add some more modifiers that does not depends from state of assertion.

#### Arguments

1. `name` (*String*): Name of assertion. It will be used for defining method or getter on `Assertion.prototype`
2. `func` (*Function*): Function that will be called on executing assertion

#### Example

```
Assertion.add('asset', function() {
```

<https://shouldjs.github.io/>

# Should / наличие атрибута

12

## should.js

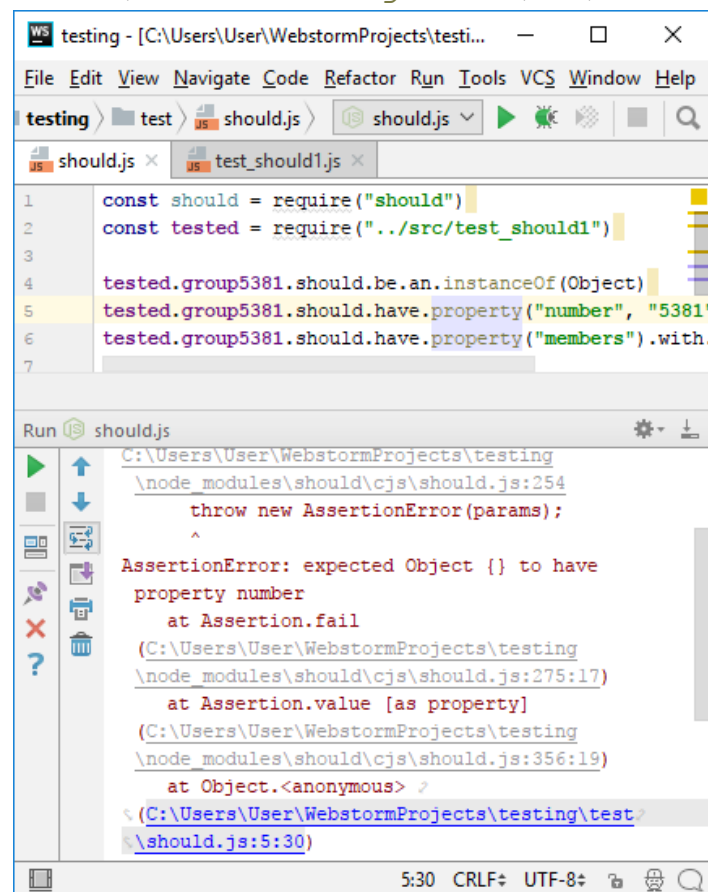
```
const should = require("should")
const tested = require("../src/test_should1")
```

```
tested.group5381.should.be.an.instanceOf(Object)
tested.group5381.should.have.property("number", "5381")
tested.group5381.should.have.property("members").with.lengthOf(15)
```

## test\_should1.js

```
module.exports = {
  group5381: {}
}
```

В Should тест  
записывается как  
**утверждение** на  
АНГЛИЙСКОМ ЯЗЫКЕ



# Should / наличие атрибута - массива <sup>13</sup>

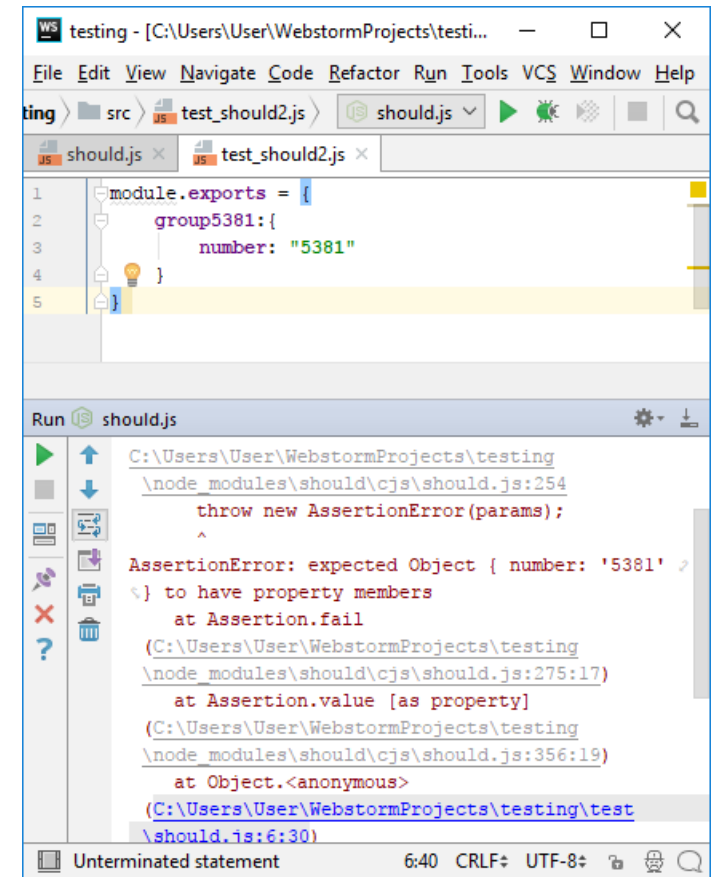
## should.js

```
const should = require("should")
const tested = require("../src/test_should2")
```

```
tested.group5381.should.be.an.instanceOf(Object)
tested.group5381.should.have.property("number", "5381")
tested.group5381.should.have.property("members").with.lengthOf(15)
```

## test\_should2.js

```
module.exports = {
  group5381: {
    number: "5381"
  }
}
```



# Should / результат разработки

14

## should.js

```
const should = require("should")
```

```
const tested = require("../src/test_should3")
```

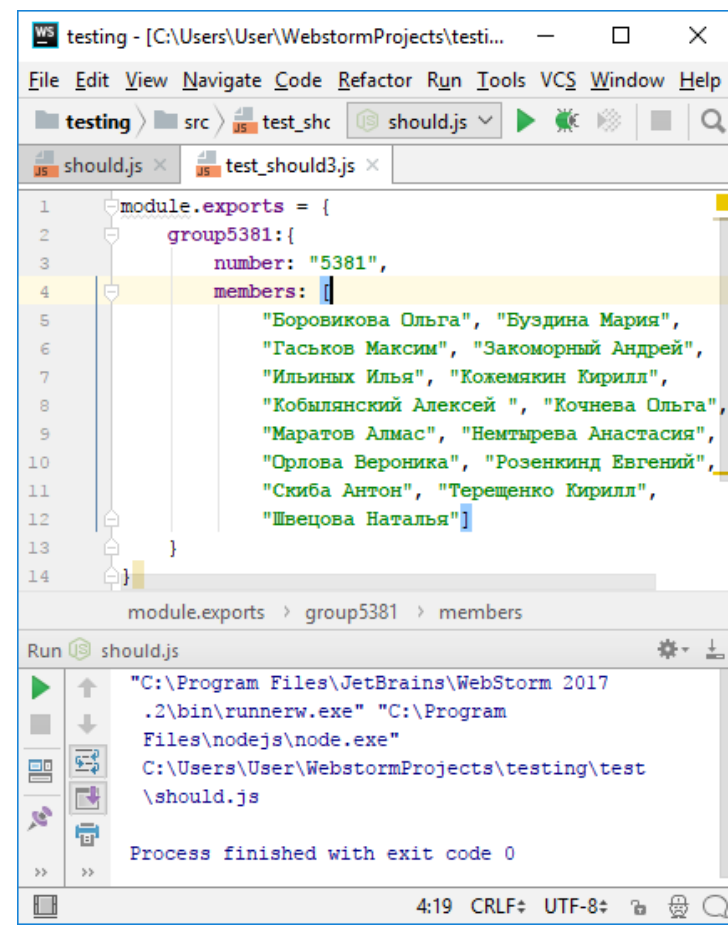
```
tested.group5381.should.be.an.instanceOf(Object)
```

```
tested.group5381.should.have.property("number", "5381")
```

```
tested.group5381.should.have.property("members").with.lengthOf(15)
```

## test\_should3.js

```
module.exports = {  
  group5381: {  
    number: "5381",  
    members: [  
      "Боровикова Ольга", "Буздина Мария",  
      "Гаськов Максим", "Закоморный Андрей",  
      "Ильиных Илья", "Кожемякин Кирилл",  
      "Кобылянский Алексей", "Кочнева Ольга",  
      "Маратов Алмас", "Немтырева Анастасия",  
      "Орлова Вероника", "Розенкинд Евгений",  
      "Скиба Антон", "Терещенко Кирилл",  
      "Швецова Наталья"]  
    }  
  }  
}
```



# Should. Методы (1)

15

- `should.ok(value, [message])`
- `Assertion#true([message])`
  - `(true).should.be.true();`
  - `false.should.not.be.true();`
- `Assertion#false([message])`
  - `(true).should.not.be.false();`
  - `false.should.be.false();`
- `Assertion#eq1(val, [description])`
  - `(10).should.be.eq1(10);`
  - `('10').should.not.be.eq1(10);`
  - `(-0).should.not.be.eq1(+0);`
- `Assertion#equal(val, [description])`
  - `10.should.be.equal(10);`
  - `'a'.should.be.exactly('a');`
- `Assertion#startWith(str, [description])`
  - `'abc'.should.startWith('a');`
- `Assertion#endWith(str, [description])`
  - `'abca'.should.endWith('a');`
- `Assertion#within(start, finish, [description])`
  - `(10).should.be.within(0, 20);`
- `Assertion#approximately(value, delta, [description])`
  - `(9.99).should.be.approximately(10, 0.1);`
- `Assertion#above(n, [description])`
  - `(10).should.be.above(0);`
- `Assertion#below(n, [description])`
  - `(0).should.be.below(10);`
- `Assertion#NaN()`
  - `(10).should.not.be.NaN();`
  - `NaN.should.be.NaN();`
- `Assertion#Infinity()`
  - `(10).should.not.be.Infinity();`
  - `NaN.should.not.be.Infinity();`
- `Assertion#type(type, [description])`
  - `(10).should.be.type('number')`

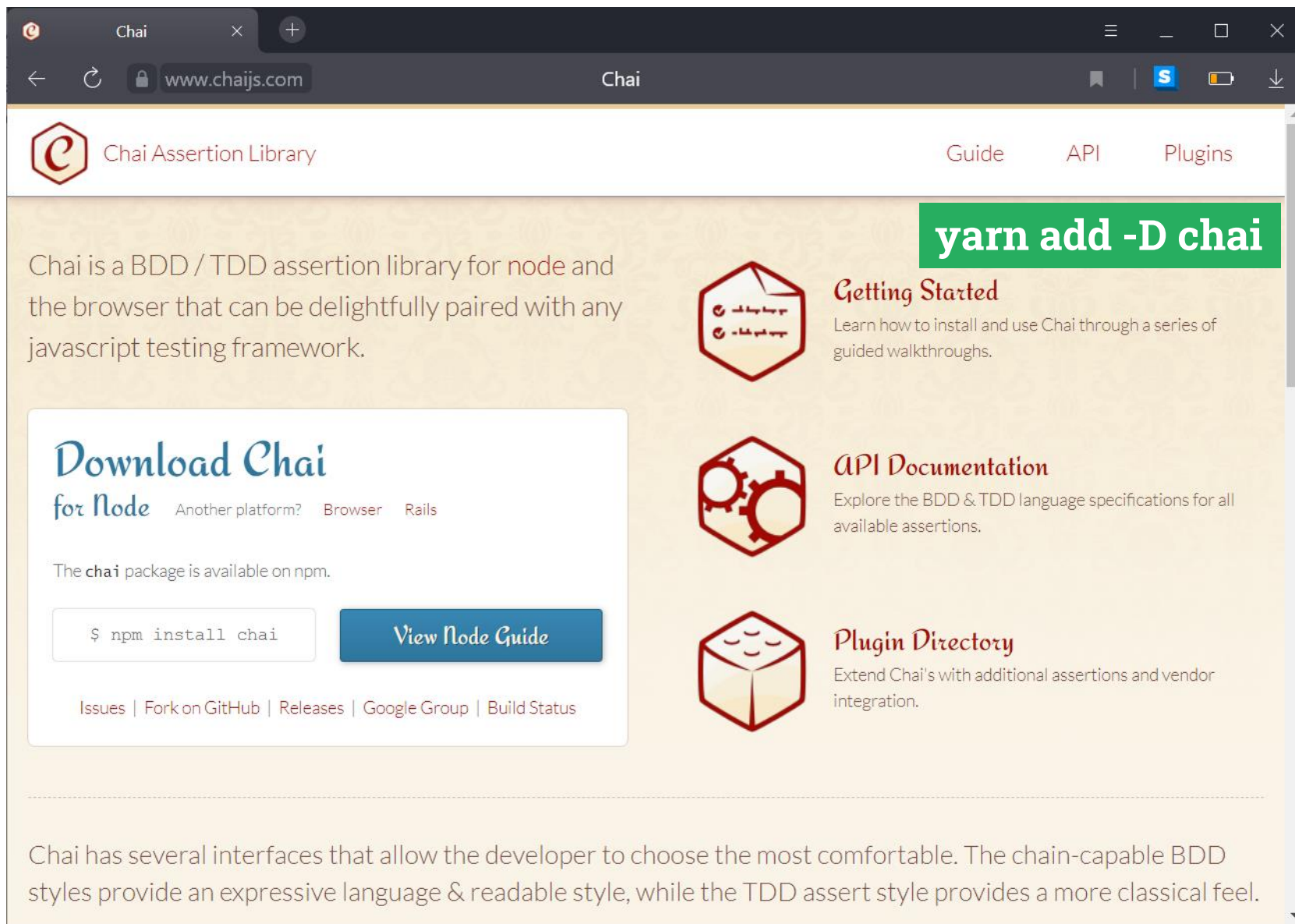
# Should. Методы (2)

- **Assertion#property(name, [val])**
  - `({ a: 10 }).should.have.property('a');`
- **Assertion#length(n, [description])**
  - `[1, 2].should.have.length(2);`
- **Assertion#ownProperty(name, [description])**
  - `({ a: 10 }).should.have.ownProperty('a');`
- **Assertion#empty()**
  - `"".should.be.empty();`
  - `[].should.be.empty();`
  - `({}).should.be.empty();`
- **Assertion#keys(keys)**
  - `({ a: 10 }).should.have.keys('a');`
  - `({ a: 10, b: 20 }).should.have.keys('a', 'b');`
  - `(new Map([[1, 2]])).should.have.key(1);`
  - `json.should.have.only.keys('type', 'version');`
- **Assertion#propertyByPath(properties)**
  - `({ a: { b: 10 } }).should.have.propertyByPath('a', 'b').eq(10);`
- **Assertion#match(other, [description])**
  - `'foobar'.should.match(/^foo/);`
  - `'foobar'.should.not.match(/^bar/);`
  - `(({ a: 'foo', c: 'barfoo' })).should.match(/foo$/);`
  - `['a', 'b', 'c'].should.match(/[a-z]/);`
  - `(5).should.not.match(function(n) { return n < 0; });`
  - `(5).should.not.match(function(it) { it.should.be.an.Array(); });`
  - `(({ a: 10, b: 'abc', c: { d: 10 }, d: 0 })).should`
  - `.match({ a: 10, b: /c$/, c: function(it) {`
  - `return it.should.have.property('d', 10);`
  - `});`
  - `[10, 'abc', { d: 10 }, 0].should`
  - `.match({ '0': 10, '1': /c$/, '2': function(it) {`
  - `return it.should.have.property('d', 10);`
  - `});`
  - `var myString = 'abc';`
  - `myString.should.be.a.String().and.match(/abc/);`
  - `myString = {};`
  - `myString.should.match(/abc/); //yes this will pass`
  - `//better to do`
  - `myString.should.be.an.Object().and.not.empty().and.match(/abc/); //fixed`
  - `(new Error('boom')).should.match(/abc/); //passed because no keys`
  - `(new Error('boom')).should.not.match({ message: /abc/ }); //check specified property`



# Chai – библиотека утверждений

17



The screenshot shows the Chai Assertion Library website in a web browser. The browser's address bar displays 'www.chaijs.com'. The website's header includes the Chai logo and navigation links for 'Guide', 'API', and 'Plugins'. A green banner at the top right contains the command 'yarn add -D chai'. The main content area features a description of Chai as a BDD/TDD assertion library, a 'Download Chai for Node' section with a code snippet and a 'View Node Guide' button, and three columns of links: 'Getting Started', 'API Documentation', and 'Plugin Directory'. A footer section at the bottom describes Chai's interfaces.

Chai Assertion Library

Guide API Plugins

**yarn add -D chai**

Chai is a BDD / TDD assertion library for node and the browser that can be delightfully paired with any javascript testing framework.

**Download Chai**  
for Node    Another platform?    Browser    Rails

The **chai** package is available on npm.

```
$ npm install chai
```

[View Node Guide](#)

[Issues](#) | [Fork on GitHub](#) | [Releases](#) | [Google Group](#) | [Build Status](#)

**Getting Started**  
Learn how to install and use Chai through a series of guided walkthroughs.

**API Documentation**  
Explore the BDD & TDD language specifications for all available assertions.

**Plugin Directory**  
Extend Chai's with additional assertions and vendor integration.

Chai has several interfaces that allow the developer to choose the most comfortable. The chain-capable BDD styles provide an expressive language & readable style, while the TDD assert style provides a more classical feel.

<https://www.chaijs.com/>

# Chai. Применение assert

18

## chai.js

```
const assert = require('chai').assert
```

```
const foo = 'бap'
```

```
const beverages = { tea: [ 'chai', 'matcha', 'oolong' ] }
```

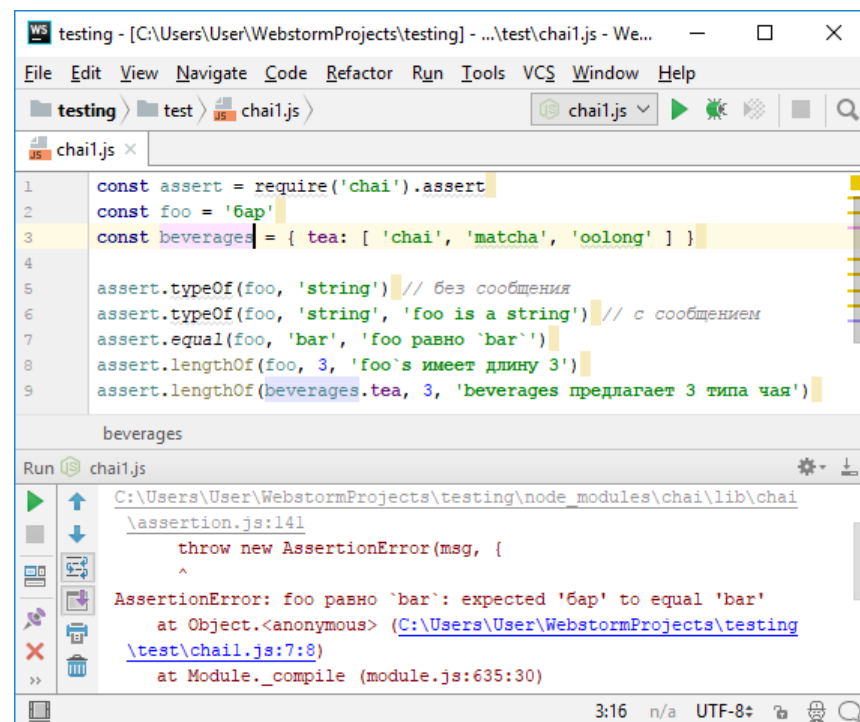
```
assert.typeOf(foo, 'string') // без сообщения
```

```
assert.typeOf(foo, 'string', 'foo is a string') // с сообщением
```

```
assert.equal(foo, 'bar', 'foo равно `bar`')
```

```
assert.lengthOf(foo, 3, 'foo`s имеет длину 3')
```

```
assert.lengthOf(beverages.tea, 3, 'beverages предлагает 3 типа чая')
```



The screenshot shows a code editor window titled "testing - [C:\Users\User\WebstormProjects\testing] - ...test\chai1.js - We...". The editor displays the same code as the previous blocks. Below the code editor, the "Run" tab is active, showing the output of the script. The output is an error message: "AssertionError: foo равно `bar`: expected 'бap' to equal 'bar'". The error message includes the file path "C:\Users\User\WebstormProjects\testing\node\_modules\chai\lib\chai\assertion.js:141" and the line number "7:8".

```
1 const assert = require('chai').assert
2 const foo = 'бap'
3 const beverages = { tea: [ 'chai', 'matcha', 'oolong' ] }
4
5 assert.typeOf(foo, 'string') // без сообщения
6 assert.typeOf(foo, 'string', 'foo is a string') // с сообщением
7 assert.equal(foo, 'bar', 'foo равно `bar`')
8 assert.lengthOf(foo, 3, 'foo`s имеет длину 3')
9 assert.lengthOf(beverages.tea, 3, 'beverages предлагает 3 типа чая')
```

Run chai1.js

```
C:\Users\User\WebstormProjects\testing\node_modules\chai\lib\chai
\assertion.js:141
  throw new AssertionError(msg, {
    ^
AssertionError: foo равно `bar`: expected 'бap' to equal 'bar'
    at Object.<anonymous> (C:\Users\User\WebstormProjects\testing
\test\chai1.js:7:8)
    at Module._compile (module.js:635:30)
```

# Chai– assert (≈1/6 функций)

19

assert

.fail

.isOk

.isNotOk

.equal

.notEqual

.strictEqual

.notStrictEqual

.deepEqual

.notDeepEqual

.isAbove

.isAtLeast

.isBelow

.isAtMost

.isTrue

.isNotTrue

.isFalse

.isNotFalse

.isNull

.isNotNull

.isNaN

.isNotNaN

.exists

.notExists

.isUndefined

.isDefined

.isFunction

.isNotFunction

.isObject

.isNotObject

.isArray

.isNotArray

.isString

.isNotString

.isNumber

<http://chaijs.com/api/assert/>

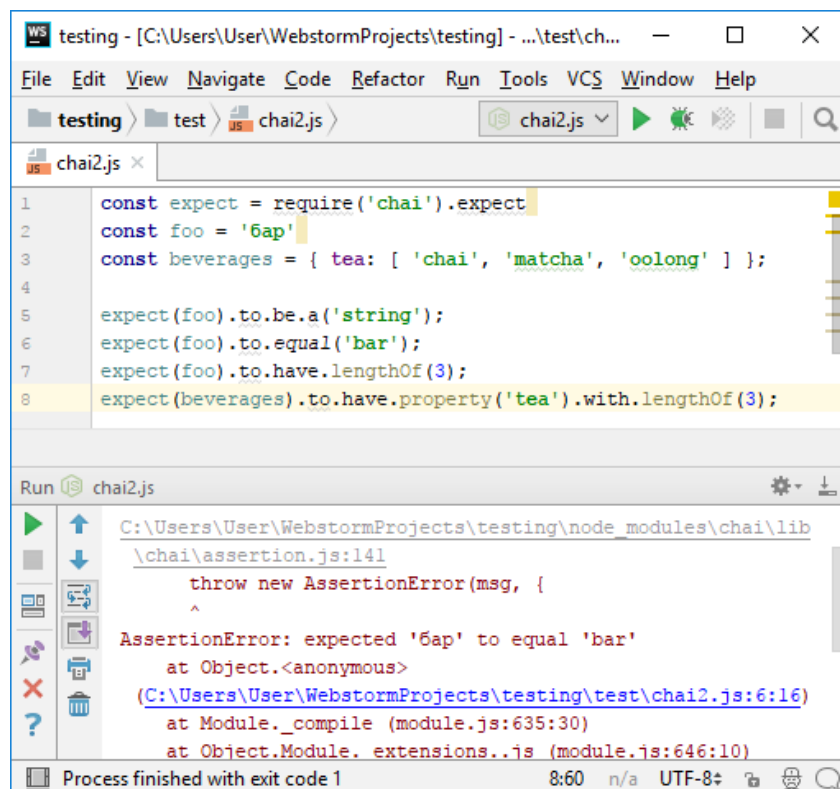
# Chai. Применение expect

20

## chai2.js

```
const expect = require('chai').expect
const foo = '6ap'
const beverages = { tea: [ 'chai', 'matcha', 'oolong' ] };

expect(foo).to.be.a('string');
expect(foo).to.equal('bar');
expect(foo).to.have.lengthOf(3);
expect(beverages).to.have.property('tea').with.lengthOf(3);
```



```
WS testing - [C:\Users\User\WebstormProjects\testing] - ...test\ch...
File Edit View Navigate Code Refactor Run Tools VCS Window Help
testing test chai2.js chai2.js
chai2.js
1 const expect = require('chai').expect
2 const foo = '6ap'
3 const beverages = { tea: [ 'chai', 'matcha', 'oolong' ] };
4
5 expect(foo).to.be.a('string');
6 expect(foo).to.equal('bar');
7 expect(foo).to.have.lengthOf(3);
8 expect(beverages).to.have.property('tea').with.lengthOf(3);

Run chai2.js
C:\Users\User\WebstormProjects\testing\node_modules\chai\lib
\chai\assertion.js:141
    throw new AssertionError(msg, {
      ^
AssertionError: expected '6ap' to equal 'bar'
    at Object.<anonymous>
(C:\Users\User\WebstormProjects\testing\test\chai2.js:6:16)
    at Module._compile (module.js:635:30)
    at Object.Module._extensions..js (module.js:646:10)
Process finished with exit code 1 8:60 n/a UTF-8
```

# Chai – expect ( $\approx 1/2$ функций)

21

- Цепочки в «языке»

- to
- be
- been
- is
- that
- which
- and
- has
- have
- with
- at
- of
- same
- but
- does

.not

.deep

.nested

.own

.ordered

.any

.all

.a

.include

.ok

.true

.false

.null

.undefined

.NaN

.exist

.empty

.arguments

.equal

.eql

.above

.least

.below

.most

.within

.instanceof

.property

.lengthOf

# Chai. Применение should

22

## chai3.js

```
const should = require('chai').should()
const foo = 'бap'
const beverages = { tea: [ 'chai', 'matcha', 'oolong' ] };

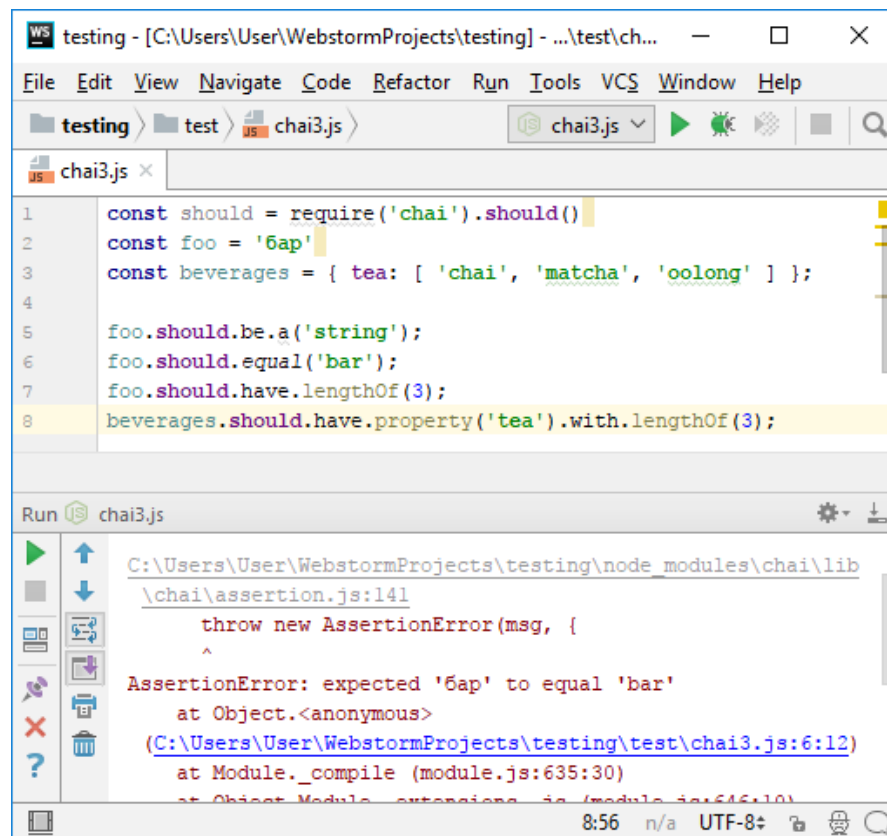
foo.should.be.a('string');
foo.should.equal('bar');
foo.should.have.lengthOf(3);
beverages.should.have.property('tea').with.lengthOf(3);
```

Методы  
**should**  
совпадают с  
методами  
**expect**.  
Отличие в  
способе  
вызова

- should.exist
- should.not.exist
- should.equal
- should.not.equal
- should.Throw
- should.not.Throw

## Ключевые отличия:

- подключение
  - **expect** – ссылка, **should** – функция
- **should** расширяет **Object.prototype**
- **should** не позволяет проверить существование объекта
- при использовании **import** (формат ES6) **should** можно подключить только отдельной строкой



# Мocha – фреймворк для тестирования

23

A screenshot of a web browser displaying the Mocha website. The browser's address bar shows 'mochajs.org' and the page title 'Mocha - the fun, simple, flexible JavaScript test fr...'. The website features a brown hexagonal logo with a coffee cup icon and the word 'MOCHA'. To the right of the logo, the text 'simple, flexible, fun' is displayed. A green box in the top right corner contains the command 'yarn add -D mocha'. Another green-bordered box on the right contains the Russian text 'Иногда называют «runner»'. The main text describes Mocha as a feature-rich JavaScript test framework running on Node.js and in the browser, making asynchronous testing simple and fun. It mentions that Mocha tests run serially, allowing for flexible and accurate reporting, while mapping uncaught exceptions to the correct test cases. It is hosted on GitHub. At the bottom, there are links for 'gitter', 'join chat', 'Sponsors 11', and 'Backers 379'.

<https://mochajs.org/>

# Мocha – фреймворк для тестирования

24

- Поддержка async и Promise
- Отчеты о покрытии тестами
- Возможность отображать разницу в полученных результатах тестов
- JS API для запуска
- Поддержка timeout (в т.ч. для различных тестов)
- Повтор запуска тестов
- Отчёт о продолжительности тестов
- Подсветка медленных тестов
- Запуск тестов, соответствующих regexp
- Поддержка библиотек для тестирования
- Расширенная поддержка отчётов
- Функции настройки тестов и пакетов тестов



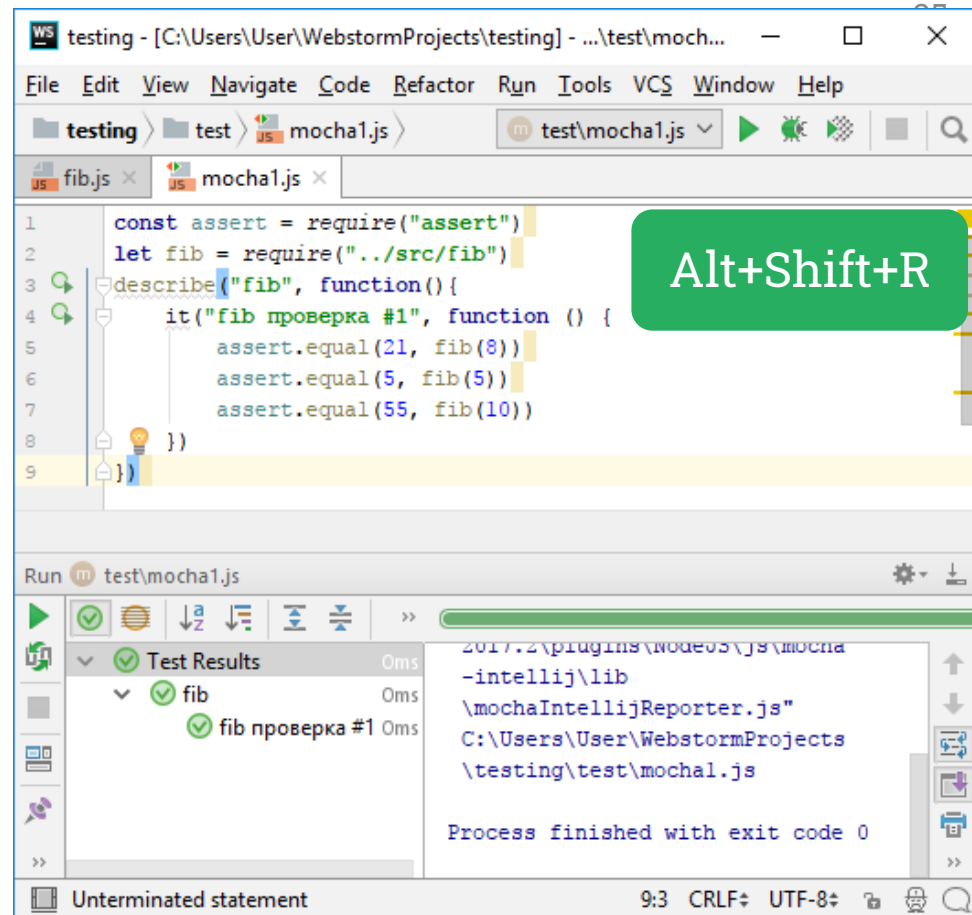
# Mocha – 1 тест

## fib.js

```
module.exports = function fib(n) {  
  if (n < 1)  
    return -1;  
  let a = [1, 1]  
  while (a.length <= n + 1) {  
    let l = a.length  
    if (l >= n)  
      return a[n - 1]  
    a.push(a[l - 1] + a[l - 2])  
  }  
}
```

## mocha1.js

```
const assert = require("assert")  
let fib = require("../src/fib")  
describe("fib", function() {  
  it("fib проверка #1", function () {  
    assert.equal(21, fib(8))  
    assert.equal(5, fib(5))  
    assert.equal(55, fib(10))  
  })  
})
```



- **describe** – объявление набора тестов
- **it** – описание теста

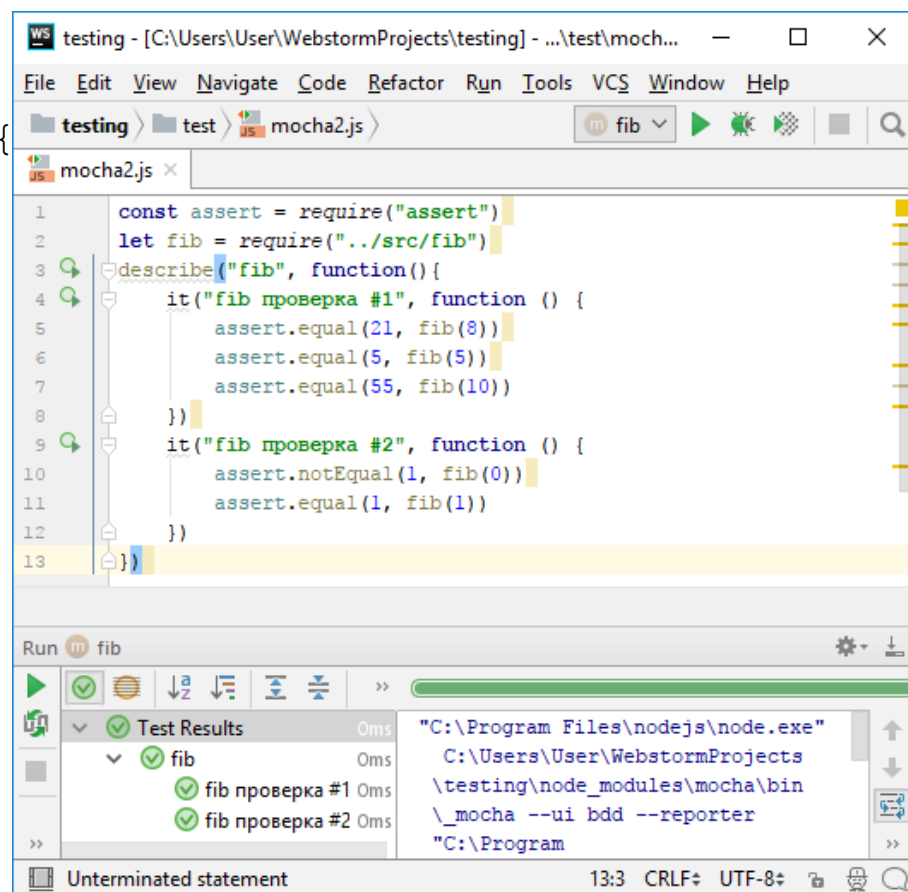
При запуске «вручную» по умолчанию работает с глобальной папкой **./test/\*.js**

# Mocha – два теста

26

## mocha2.js

```
const assert = require("assert")
let fib = require("../src/fib")
describe("fib", function() {
  it("fib проверка #1", function () {
    assert.equal(21, fib(8))
    assert.equal(5, fib(5))
    assert.equal(55, fib(10))
  })
  it("fib проверка #2", function () {
    assert.notEqual(0, fib(0))
    assert.equal(1, fib(1))
  })
})
```



# Mocha. before, after

27

## mocha3.js

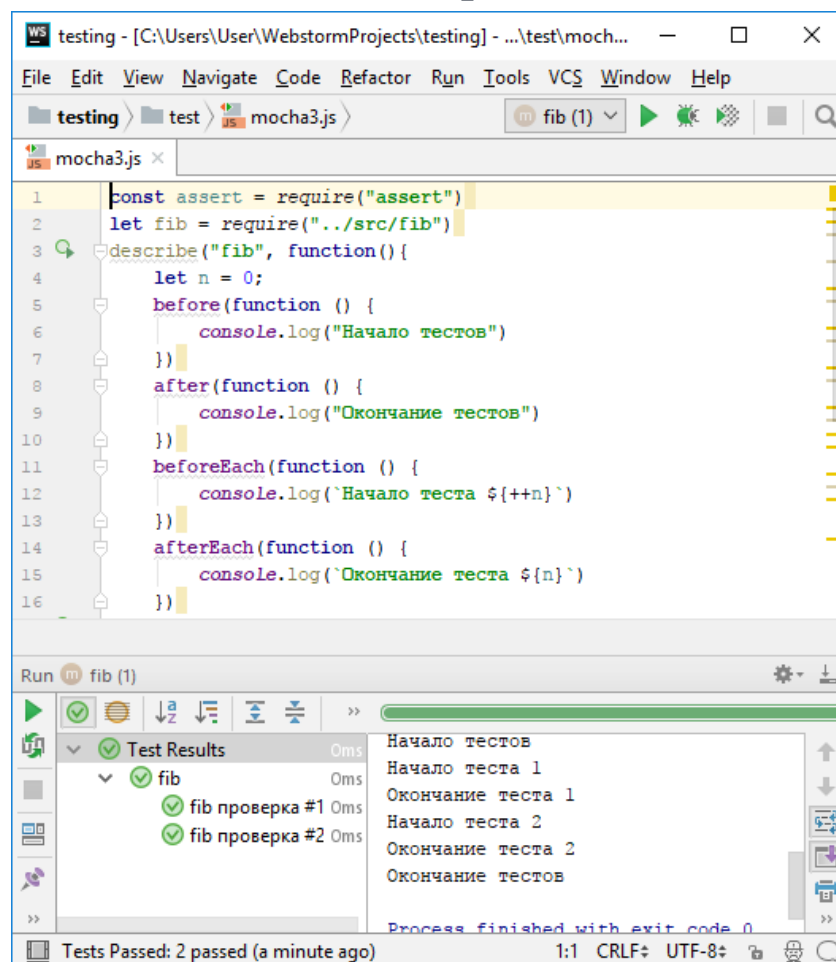
```
const assert = require("assert")
let fib = require("../src/fib")
describe("fib", function() {
  let n = 0;
  before(function () {
    console.log("Начало тестов")
  })
  after(function () {
    console.log("Окончание тестов")
  })
  beforeEach(function () {
    console.log(`Начало теста ${++n}`)
  })
  afterEach(function () {
    console.log(`Окончание теста ${n}`)
  })
  it("fib проверка #1", function () {
    assert.equal(21, fib(8))
    assert.equal(5, fib(5))
    assert.equal(55, fib(10))
  })
  it("fib проверка #2", function () {
    assert.notEqual(1, fib(0))
    assert.equal(1, fib(1))
  })
})
```

## package.json

```
"scripts": {
  "test": "mocha"
}
```

- describe == suite
- it == test
- before == setup
- after == teardown

## npm test



# Мocha. Поддерживаемые библиотеки

28

should.js

expect.js

chai

better-  
assert

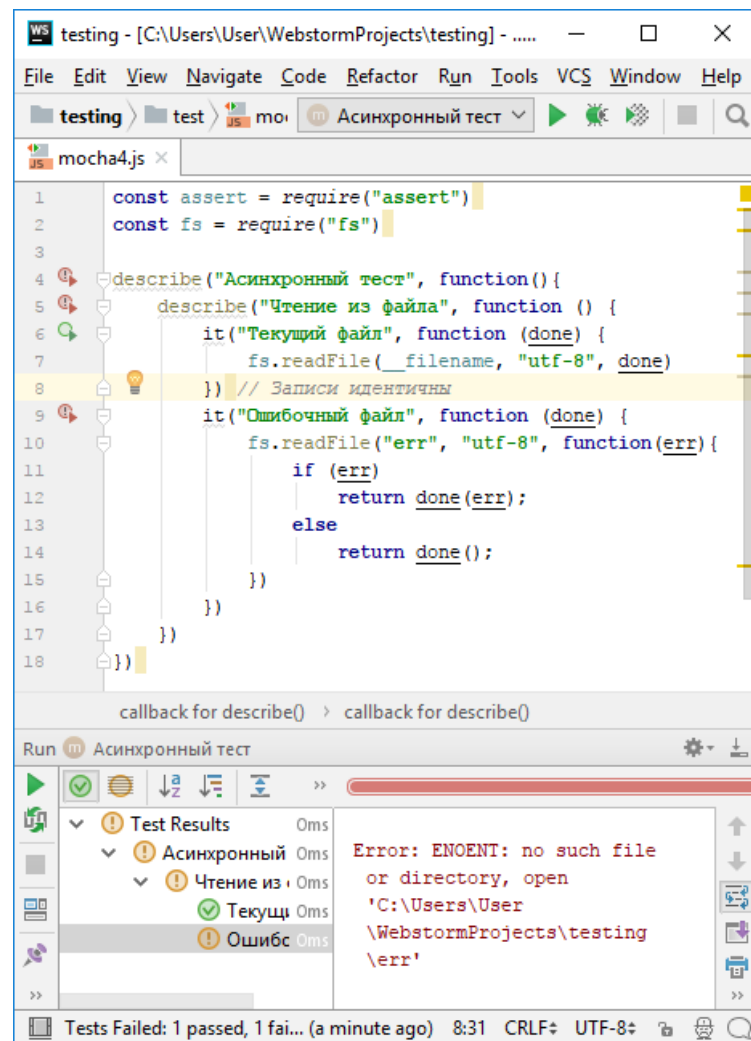
unexpected

# Мocha. Тестирование асинхронного кода

## mocha4.js

```
const assert = require("assert")
const fs = require("fs")
```

```
describe("Асинхронный тест", function() {
  describe("Чтение из файла", function() {
    it("Текущий файл", function(done) {
      fs.readFile(__filename, "utf-8", done)
    }) // Записи идентичны
    it("Ошибочный файл", function(done) {
      fs.readFile("err", "utf-8",
        function(err) {
          if (err)
            return done(err);
          else
            return done();
        }
      )
    })
  })
})
```



# Mocha. async/await

30

## mocha5.js

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

```
const fs = require("fs")
```

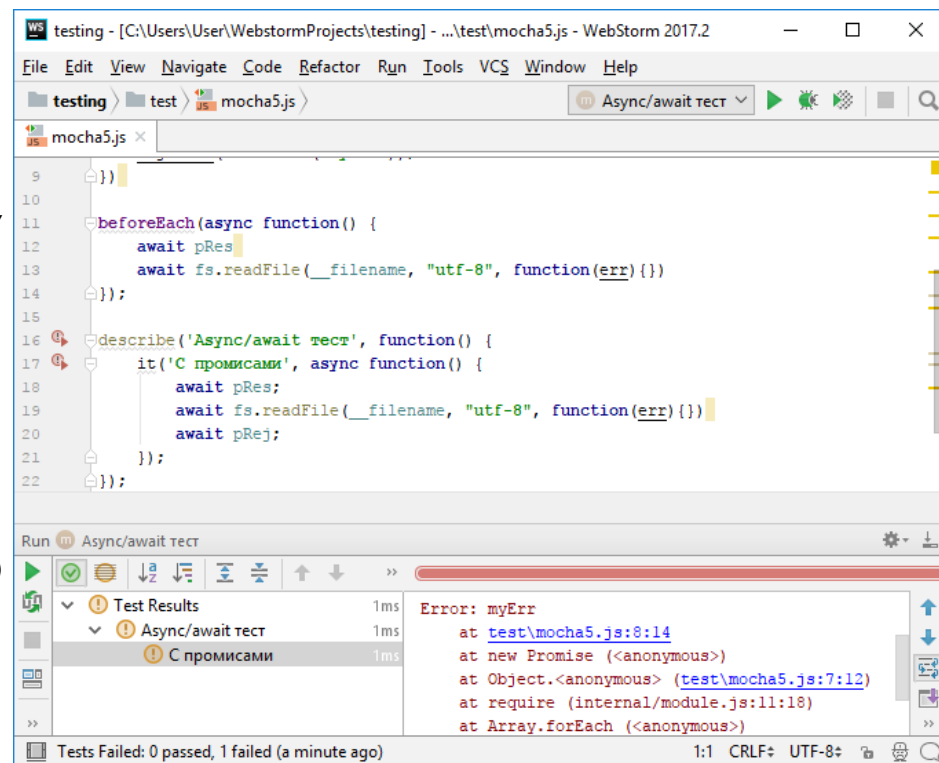
```
let pRes = new Promise(function(resolve, rejected) {  
  resolve("ok");  
})
```

```
let pRej = new Promise(function(resolve, rejected) {  
  rejected(new Error("myErr"));  
})
```

```
beforeEach(async function() {  
  await pRes  
  await fs.readFile(__filename, "utf-8",  
    function(err) {})  
});
```

```
describe('Async/await тест', function() {  
  it('С промисами', async function() {  
    await pRes;  
    await fs.readFile(__filename,  
      "utf-8", function(err) {})  
    await pRej;  
  });  
});
```

Попробуйте  
закомментировать  
вызов await pRej



# Мocha. Синхронные функции и =>

31

## mocha6.js

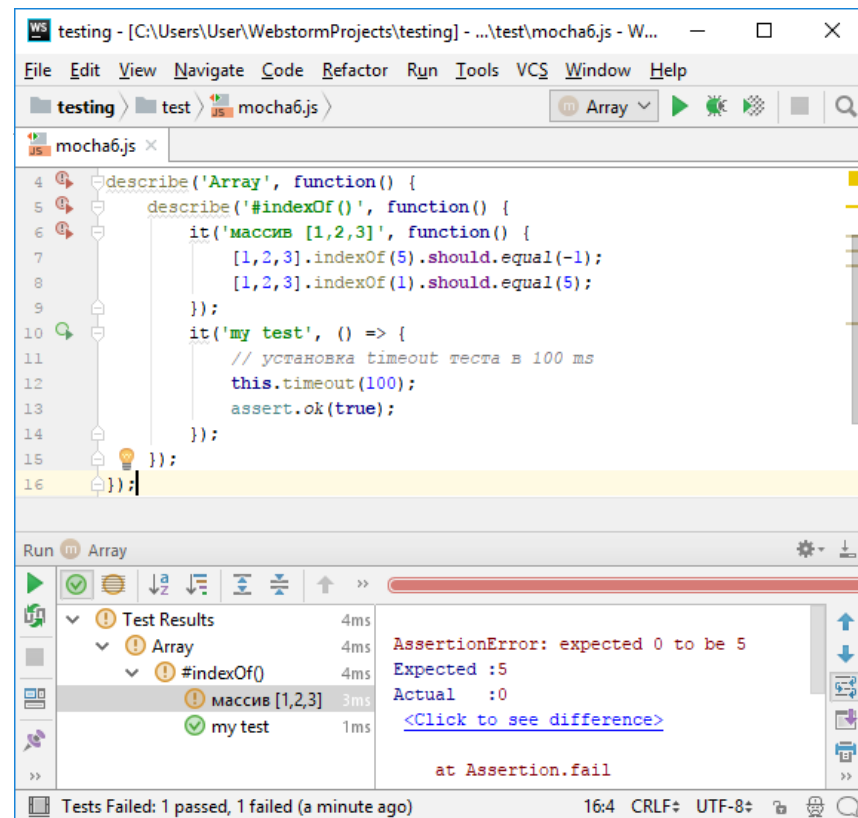
```
const assert = require('assert');
```

```
const should = require("should")
```

```
describe('Array', function() {  
  describe('#indexOf()', function() {  
    it('массив [1,2,3]', function() {  
      [1,2,3].indexOf(5).should.equal(-1);  
      [1,2,3].indexOf(1).should.equal(5);  
    });  
  });  
});
```

```
it('my test', () => {  
  // установка timeout теста  
  this.timeout(100);  
  assert.ok(true);  
});
```

```
});
```



# Mocha. Запуск выбранных тестов / only

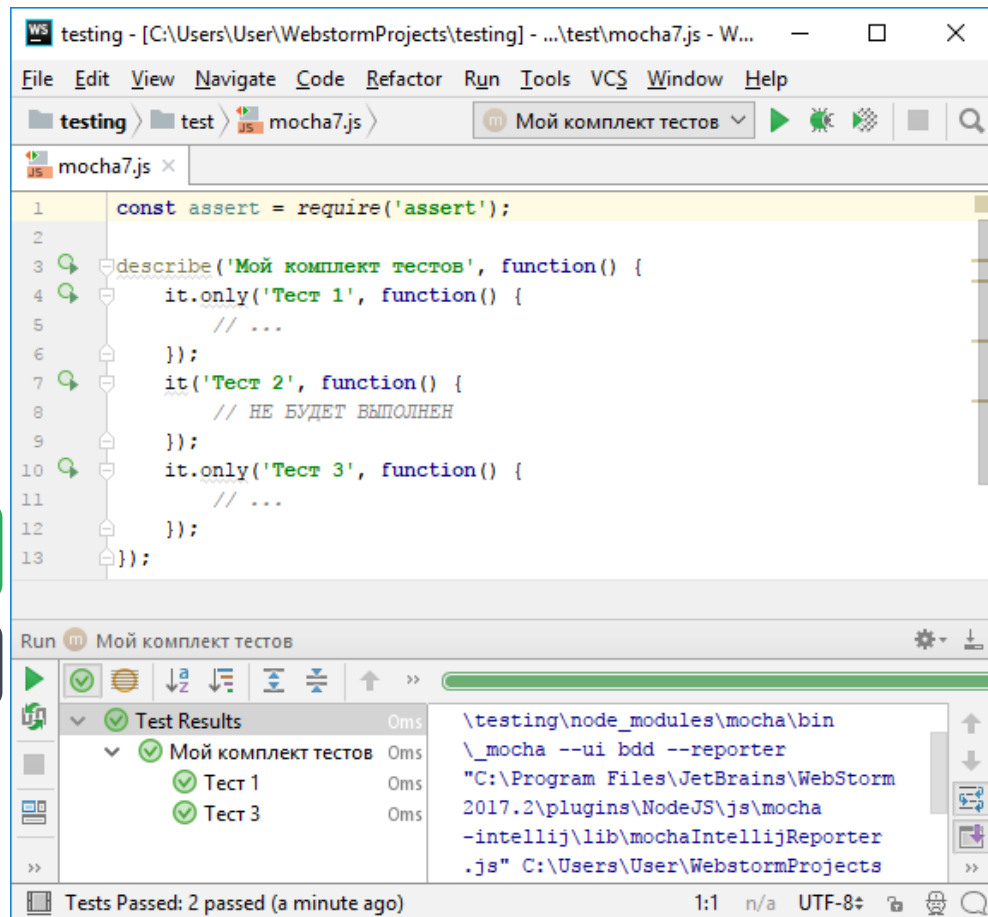
32

mocha7.js

```
const assert = require('assert');
describe('Мой комплект тестов', function() {
  it.only('Тест 1', function() {
    // ...
  });
  it('Тест 2', function() {
    // НЕ БУДЕТ ВЫПОЛНЕН
  });
  it.only('Тест 3', function() {
    // ...
  });
});
```

Этот же метод работает для describe

Попробуйте заменить на skip



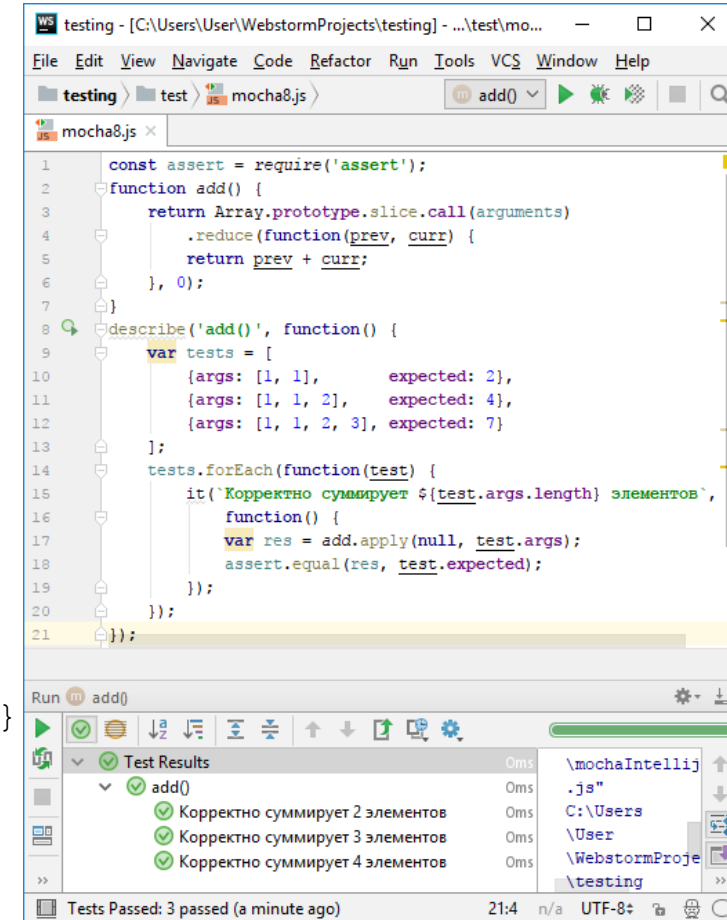


# Mocha. Генерируемые тесты / forEach, apply

33

## mocha8.js

```
const assert = require('assert');
function add() {
  return Array.prototype.slice.call(arguments)
    .reduce(function(prev, curr) {
      return prev + curr;
    }, 0);
}
describe('add()', function() {
  var tests = [
    {args: [1, 1],          expected: 2},
    {args: [1, 1, 2],       expected: 4},
    {args: [1, 1, 2, 3],    expected: 7}
  ];
  tests.forEach(function(test) {
    it('Корректно суммирует ${test.args.length} элементов',
      function() {
        var res = add.apply(null, test.args);
        assert.equal(res, test.expected);
      });
  });
});
```



apply()



# Настройка контекста / call, apply, bind

```
function log(y) {  
  if(this.x)  
    console.log(this.x, y)  
  else  
    console.log("Unknown", y)  
}
```

*// Указание контекста при вызове - **call** - параметры передаются через запятую*

```
log.call({x:"me"}, "1234") // me 1234
```

```
log.call(null, "1234") // Unknown 1234
```

*// Указание контекста при вызове - **apply** - параметры передаются как массив*

```
log.apply({x:"you"}, ["42"]) // you 42
```

```
log.apply(undefined, ["42"]) // Unknown 42
```

*// Настройка контекста - **bind***

```
let myLog1 = log.bind({x:"us"})
```

```
let myLog2 = log.bind(null)
```

```
myLog1("abc") // us abc
```

```
myLog2("abc") // Unknown abc
```

# Мocha. Ограничение по времени выполнения / timeout

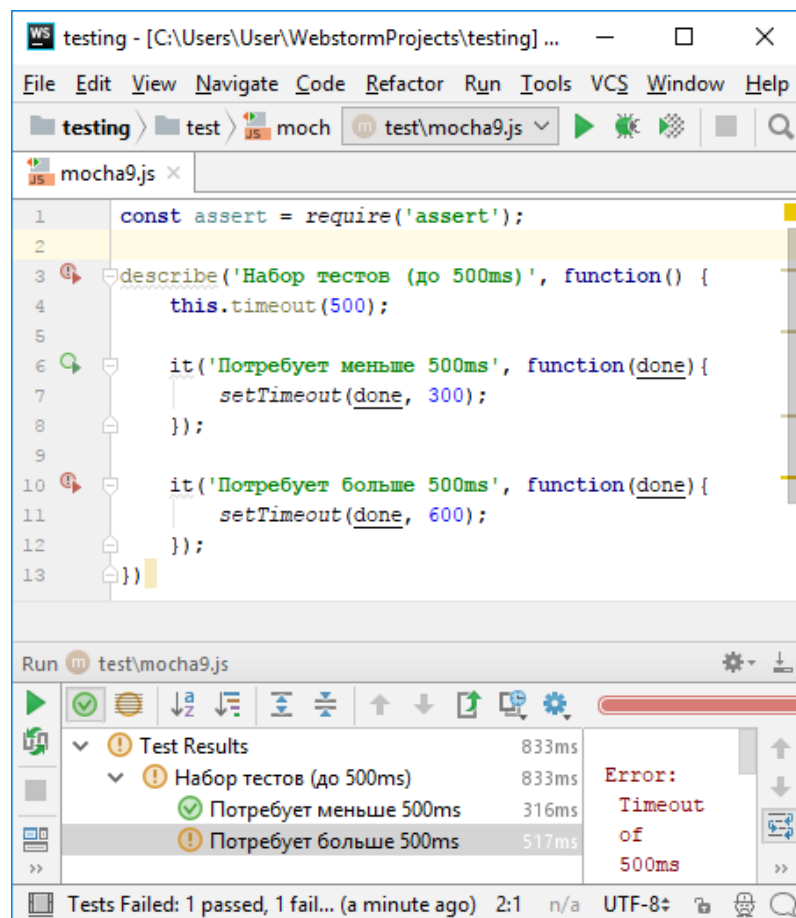
mocha9.js

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

```
describe('Набор тестов (до 500ms)', function() {
  this.timeout(500);

  it('Потребуется меньше 500ms', function(done) {
    setTimeout(done, 300);
  });

  it('Потребуется больше 500ms', function(done) {
    setTimeout(done, 600);
  });
});
```



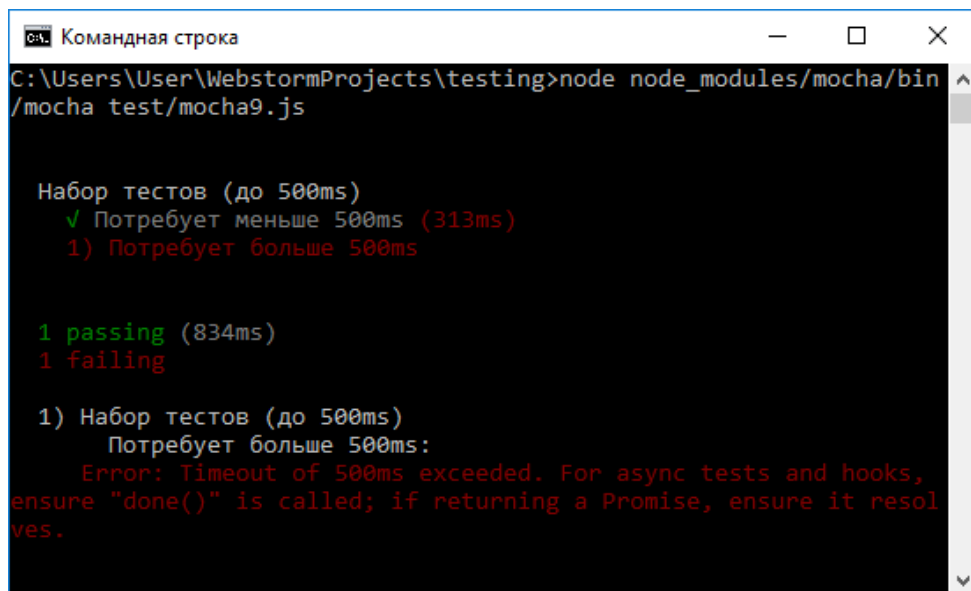
- **TDD** — это методология разработки ПО, которая основывается на повторении коротких циклов разработки:
  - изначально **пишется тест**, покрывающий желаемое изменение,
  - затем **пишется программный код**, который реализует желаемое поведение системы и
  - позволит **пройти написанный тест**,
  - затем проводится **рефакторинг** написанного кода с постоянной проверкой прохождения тестов
- **BDD** — **behaviour-driven development** — это разработка, основанная на описании поведения
- Выделенный человек пишет описания вида "**я как пользователь хочу, чтобы когда нажали кнопку пуск, тогда показывалось меню как на картинке**"
- BDD предполагает **описание** тестировщиком или аналитиком **пользовательских сценариев на естественном языке**, на языке бизнеса

- **BDD** (behavior-driven development)
  - describe()
  - it()
  - before()
  - after()
  - beforeEach()
  - afterEach()
  - context()
  - specify()
- **TDD** (test-driven development)
  - suite()
  - test()
  - setup()
  - teardown()
  - suiteSetup()
  - suiteTeardown()
- Exports
- QUnit
- Require

# Мocha. Формирование отчётов

38

- `node node_modules/mocha/bin/mocha test/mocha9.js`
- `node node_modules/mocha/bin/mocha --reporter doc test/mocha9.js > doc.html`



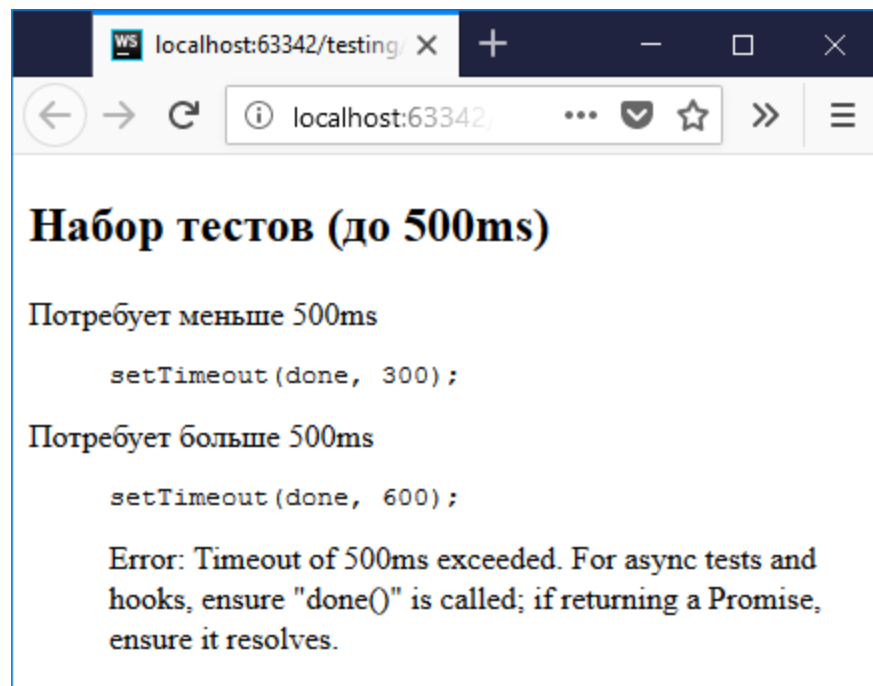
```
Командная строка
C:\Users\User\WebstormProjects\testing>node node_modules/mocha/bin/mocha test/mocha9.js

Набор тестов (до 500ms)
  ✓ Потребует меньше 500ms (313ms)
    1) Потребует больше 500ms

  1 passing (834ms)
  1 failing

  1) Набор тестов (до 500ms)
      Потребует больше 500ms:
        Error: Timeout of 500ms exceeded. For async tests and hooks,
        ensure "done()" is called; if returning a Promise, ensure it resolves.
```

**Здесь mocha установлен локально**



```
localhost:63342/testing X
localhost:63342

Набор тестов (до 500ms)

Потребует меньше 500ms
  setTimeout(done, 300);

Потребует больше 500ms
  setTimeout(done, 600);

Error: Timeout of 500ms exceeded. For async tests and
hooks, ensure "done()" is called; if returning a Promise,
ensure it resolves.
```

Того же эффекта можно добиться командами вида

```
npx mocha test/mocha9.js
npx mocha --reporter doc test/mocha9.js > doc.html
```

# Mocha. Варианты отчётов

39

SPEC

DOT  
MATRIX

NYAN

TAP

LANDING  
STRIP

LIST

PROGRESS

JSON

JSON  
STREAM

MIN

DOC

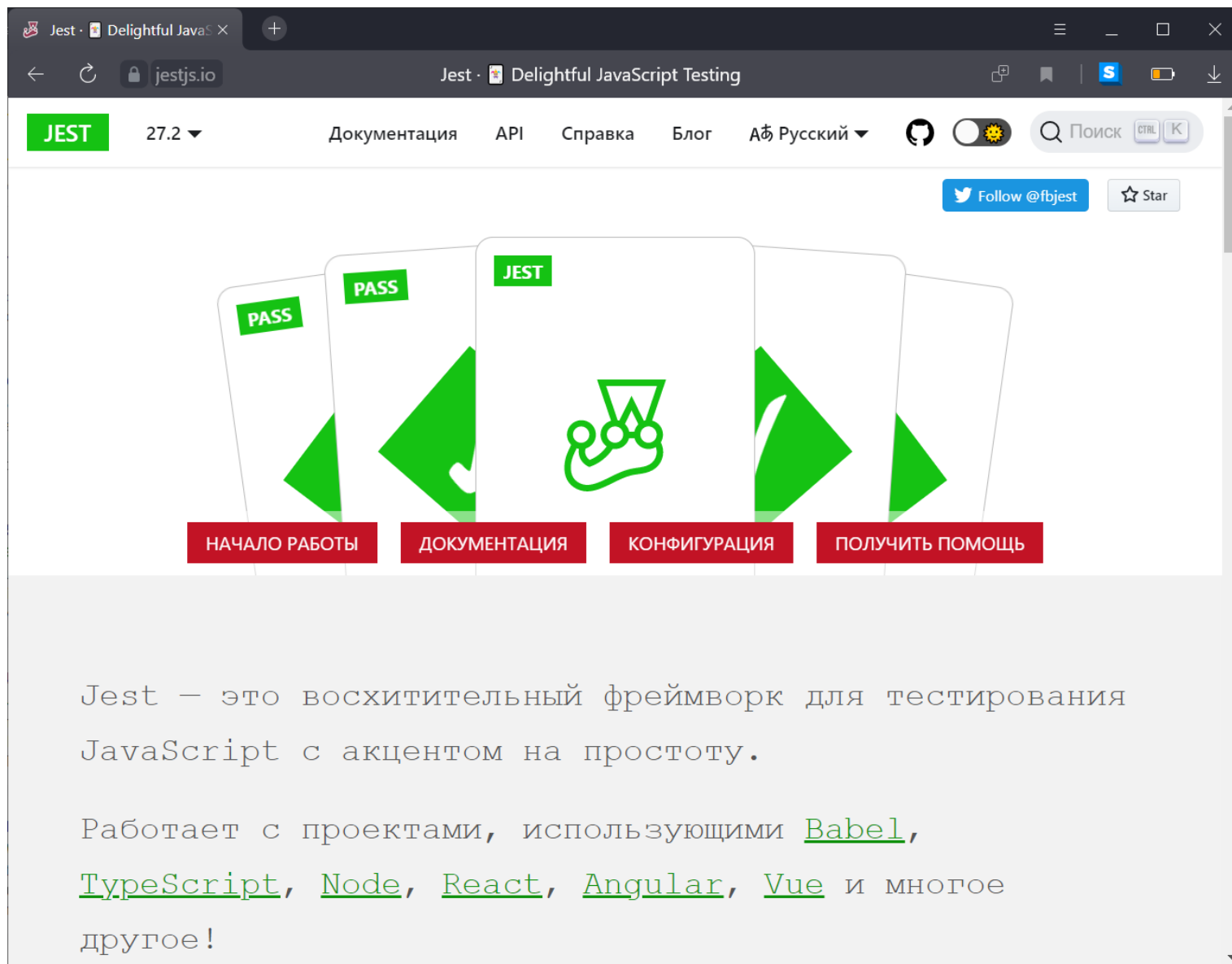
MARKDOWN

HTML

THIRD  
PARTY  
REPORTERS

# Jest – фреймворк тестирования

40



<https://jestjs.io/ru/>



# Jest. Простой пример / test, expect

41

## Установка

```
npm install --save-dev jest
```

## Добавить в package.json

```
"scripts": {  
  "test": "jest"  
}
```

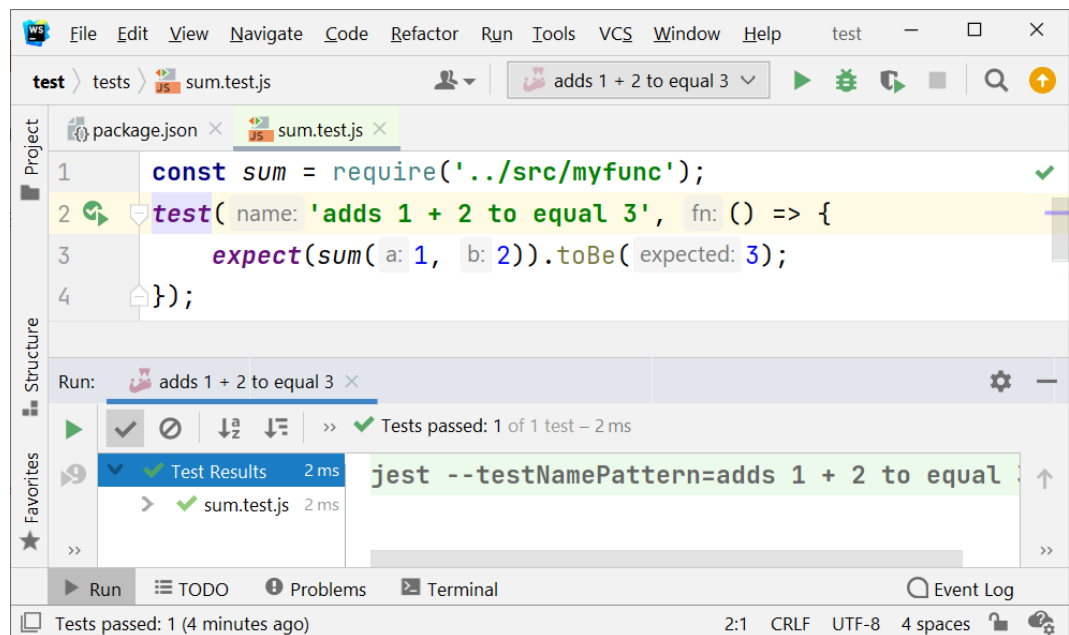
## src/sum.js

```
/**  
 * Сложение a+b  
 * @param a:number первое слагаемое  
 * @param b:number второе слагаемое  
 * @returns {number}  
 */
```

```
function sum(a, b) {  
  return a + b;  
}  
module.exports = sum;
```

## tests/sum.test.js

```
const sum = require('../src/sum');  
  
test('adds 1 + 2 to equal 3', () => {  
  expect(sum(1, 2)).toBe(3);  
});
```



## Альтернативный запуск:

```
npm run test
```

<https://jestjs.io/ru/docs/expect>

**describe** – как в Mocha

# Jest. Конфигурация

## jest.config.js

jest --init

```
module.exports = {
```

```
  // Automatically clear mock calls and instances between every test
```

```
  clearMocks: true,
```

```
  // Indicates whether the coverage information should be collected while executing the test
```

```
  collectCoverage: true,
```

```
  // The directory where Jest should output its coverage files
```

```
  coverageDirectory: "coverage",
```

```
  // Indicates which provider should be used to instrument code for coverage
```

```
  coverageProvider: "v8",
```

```
  // A list of reporter names that Jest uses when writing coverage reports
```

```
  // coverageReporters: [
```

```
    // "json",
```

```
    // "text",
```

```
    // "lcov",
```

```
    // "clover"
```

```
  // ],
```

```
};
```

beforeEach

afterEach

beforeAll

afterAll

describe

test

test.only

# Jest. Правдивость / toBe...

43

```
test('null', () => {  
  const n = null;  
  expect(n).toBeNull(); // toBeNull верно только для null  
  expect(n).toBeDefined(); // toBeDefined противоположность toBeUndefined  
  expect(n).not.toBeUndefined(); // toBeUndefined верно только для undefined  
  expect(n).not.toBeTruthy(); // toBeTruthy верно, если if рассматривает как true  
  expect(n).toBeFalsy(); // toBeFalsy верно, если if рассматривает как false  
});
```

```
test('ноль', () => {  
  const z = 0;  
  expect(z).not.toBeNull();  
  expect(z).toBeDefined();  
  expect(z).not.toBeUndefined();  
  expect(z).not.toBeTruthy();  
  expect(z).toBeFalsy();  
});
```

## Выполнен неявный вызов

```
import {describe, expect, test} from '@jest/globals'
```

# Jest. Числа / toBe...

```
test('два плюс два', () => {  
  const value = 2 + 2;  
  expect(value).toBeGreaterThan(3);  
  expect(value).toBeGreaterThanOrEqual(3.5);  
  expect(value).toBeLessThan(5);  
  expect(value).toBeLessThanOrEqual(4.5);
```

*// toBe и toEqual эквивалентны по отношению к числам*

```
  expect(value).toBe(4);  
  expect(value).toEqual(4);  
});
```

```
test('сложение чисел с плавающей запятой', () => {
```

```
  const value = 0.1 + 0.2;
```

*//expect(value).toBe(0.3); Это не будет работать из-за ошибки округления*

```
  expect(value).toBeCloseTo(0.3); // А это сработает.  
});
```

# Jest. Строки / toMatch

```
test('в команде нет места Я', () => {  
  expect('команда').not.toMatch(/Я/);  
});
```

```
test('но есть "ася" в Васе', () => {  
  expect('Вася').toMatch(/ася/);  
});
```

# Jest. Массивы и перебираемые объекты / toContain

```
const shoppingList = [  
  'diapers',  
  'kleenex',  
  'trash bags',  
  'paper towels',  
  'milk',  
];
```

```
test('the shopping list has milk on it', () => {  
  expect(shoppingList).toContain('milk');  
  expect(new Set(shoppingList)).toContain('milk');  
  expect(shoppingList.length).toBe(5);  
});
```

# Jest. Искключения / toThrow

```
function compileAndroidCode() {  
  throw new Error('you are using the wrong JDK');  
}
```

```
test('compiling android goes as expected', () => {  
  expect() => compileAndroidCode().toThrow();  
  expect() => compileAndroidCode().toThrow(Error);
```

*// You can also use the exact error message or a regexp*

```
expect() => compileAndroidCode().toThrow('you are using the wrong JDK');  
expect() => compileAndroidCode().toThrow(/JDK/);  
});
```

# Jest. Асинхронный код (1)

48

```
test('данные являются арахисовым маслом', done => {
  function callback(data) {
    try {
      expect(data).toBe('арахисовое масло');
      done();
    } catch (error) {
      done(error);
    }
  }
  fetchData(callback); // Обратный вызов
});

test('the data is peanut butter', () => {
  return fetchData().then(data => { // Промисы
    expect(data).toBe('peanut butter');
  });
});

test('the fetch fails with an error', () => {
  expect.assertions(1); // Срабатывает 1 assert
  return fetchData().catch(e => expect(e).toMatch('error'));
});

test('the fetch fails with an error', () => {
  return expect(fetchData()).rejects.toMatch('error');
});
```



# Jest. Асинхронный код (2)

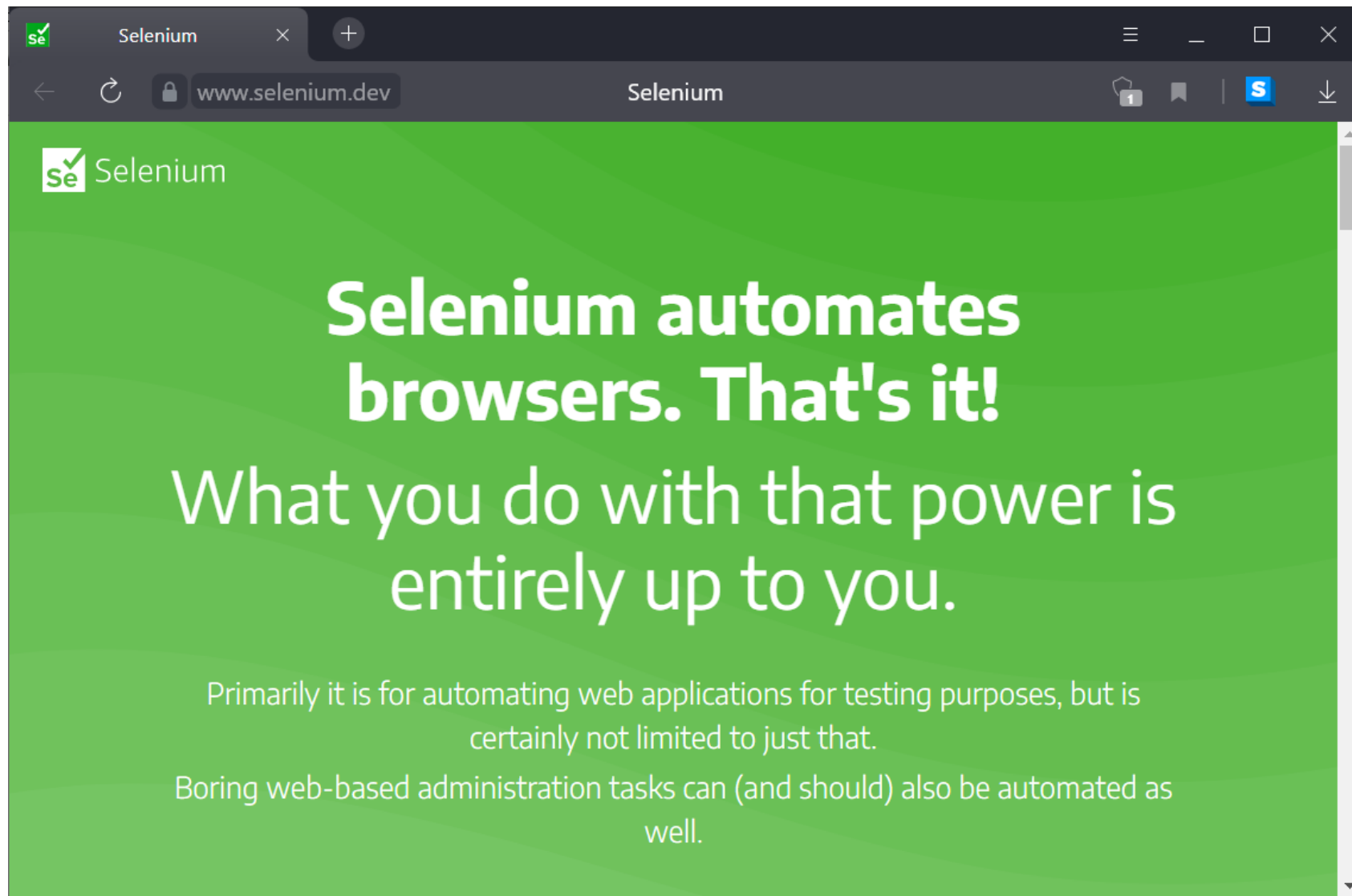
```
test('the data is peanut butter', async () => {  
  const data = await fetchData();  
  expect(data).toBe('peanut butter');  
});
```

```
test('the fetch fails with an error', async () => {  
  expect.assertions(1); // Сработаем 1 assert  
  try {  
    await fetchData();  
  } catch (e) {  
    expect(e).toMatch('error');  
  }  
});
```

```
test('данные являются арахисовым маслом', async () => {  
  await expect(fetchData()).resolves.toBe('арахисовое масло');  
});
```

```
test('fetch вернёт ошибку', async () => {  
  await expect(fetchData()).rejects.toMatch('error');  
});
```

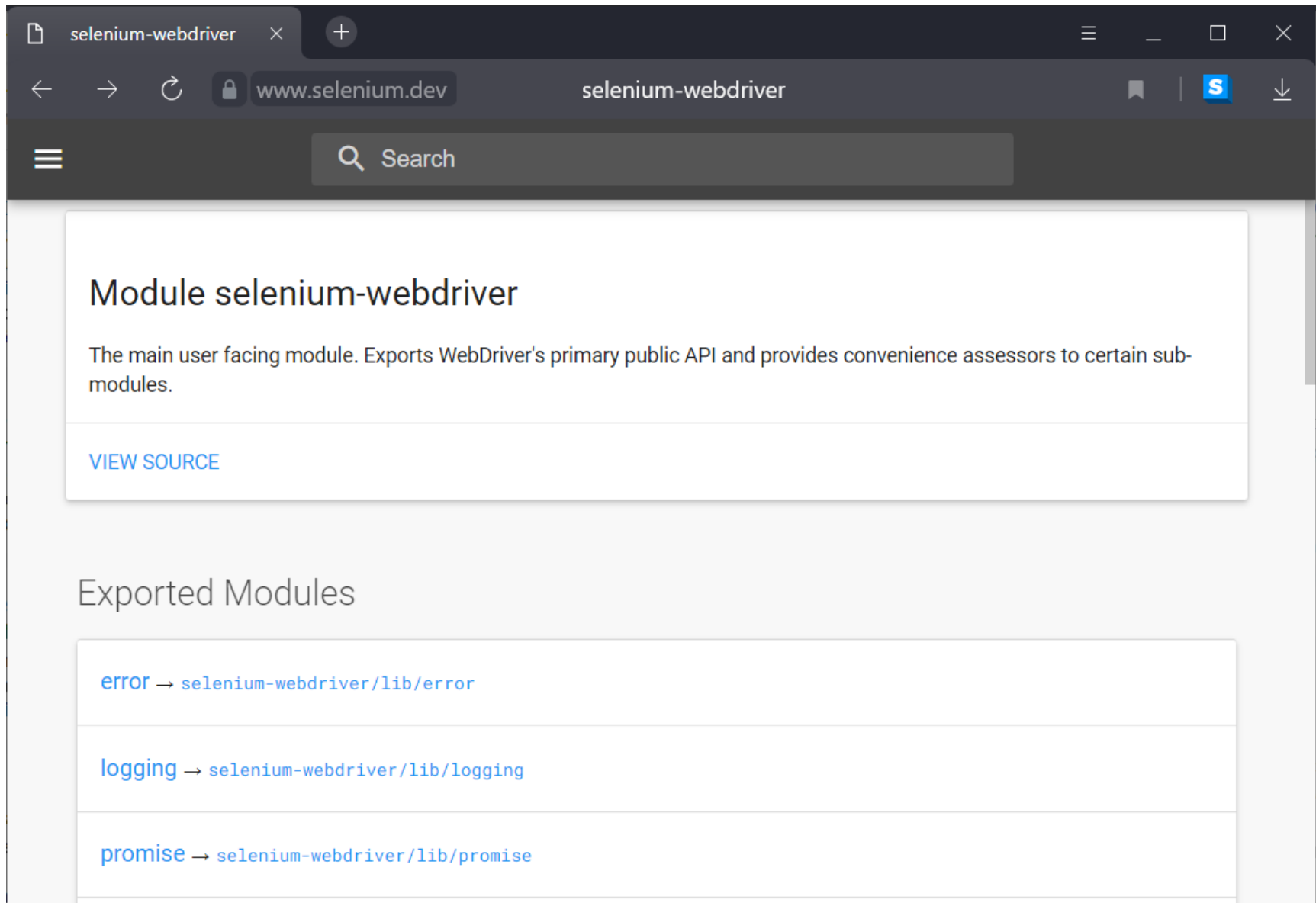
# Selenium – фреймворк тестирования для web-приложений



<https://www.selenium.dev/>

# Selenium. WebDriver

51



The screenshot shows a web browser window with the address bar displaying 'www.selenium.dev' and the page title 'selenium-webdriver'. The page content is as follows:

## Module selenium-webdriver

The main user facing module. Exports WebDriver's primary public API and provides convenience assessors to certain sub-modules.

[VIEW SOURCE](#)

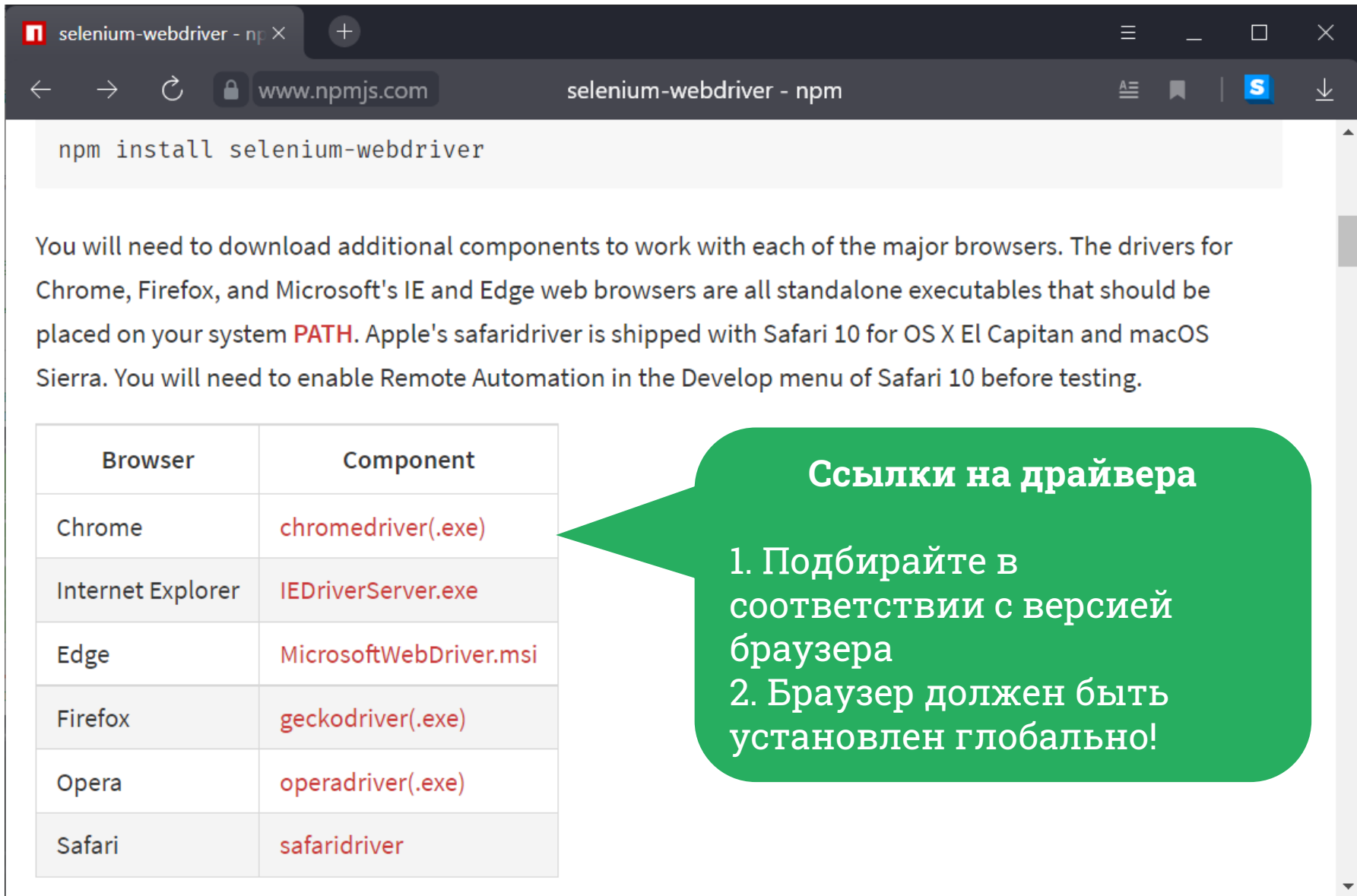
### Exported Modules

<code>error</code>	→ <code>selenium-webdriver/lib/error</code>
<code>logging</code>	→ <code>selenium-webdriver/lib/logging</code>
<code>promise</code>	→ <code>selenium-webdriver/lib/promise</code>

<https://www.selenium.dev/selenium/docs/api/javascript/module/selenium-webdriver/>

# chromedriver в переменной PATH

52



The screenshot shows the npm page for selenium-webdriver. The command `npm install selenium-webdriver` is displayed at the top. Below it, a paragraph explains that additional components are needed for major browsers. A table lists the browser and the corresponding component. A green callout box points to the 'chromedriver(.exe)' entry in the table, containing two instructions in Russian: 1. Choose the driver according to the browser version, and 2. The browser must be installed globally.

```
npm install selenium-webdriver
```

You will need to download additional components to work with each of the major browsers. The drivers for Chrome, Firefox, and Microsoft's IE and Edge web browsers are all standalone executables that should be placed on your system **PATH**. Apple's safaridriver is shipped with Safari 10 for OS X El Capitan and macOS Sierra. You will need to enable Remote Automation in the Develop menu of Safari 10 before testing.

Browser	Component
Chrome	<code>chromedriver(.exe)</code>
Internet Explorer	<code>IEDriverServer.exe</code>
Edge	<code>MicrosoftWebDriver.msi</code>
Firefox	<code>geckodriver(.exe)</code>
Opera	<code>operadriver(.exe)</code>
Safari	<code>safaridriver</code>

**Ссылки на драйвера**

1. Подбирайте в соответствии с версией браузера
2. Браузер должен быть установлен глобально!

<https://www.npmjs.com/package/selenium-webdriver>

# Selenium. Простой автоматизированный тест

```
const {Builder, By, Key, until} = require('selenium-webdriver');
(async function example() {
  let driver = await new Builder().forBrowser('chrome').build();
  try {
    // Перейти по URL
    await driver.get('https://www.yandex.ru');
    // Ввести текст "cheese" и нажать на клавиатуре "Enter"
    await driver.findElement(By.name('text')).sendKeys('cheese', Key.ENTER);
    let firstResult = await driver.wait(until.elementLocated(By.css('.main__content')), 10000);
    console.log(await firstResult.getAttribute('textContent'));
  }
  finally{
    await driver.quit();
  }
})();
```

1. Скачал chromedriver (это для Google Chrome, для других браузеров выбирайте соответствующие драйвера)
2. Положил его в папку, прописанную в PATH
3. Запустил тестирование
4. При этом запустился браузер и я увидел весь процесс тестирования

# Selenium. WebElement

54

```
driver.get('http://www.google.com');  
let searchForm = driver.findElement(By.tagName('form'));  
let searchBox = searchForm.findElement(By.name('q'));  
searchBox.sendKeys('webdriver');
```

clear()

click()

findElement(  
locator )

findElements(  
locator )

getAttribute(  
attributeName )

getCssValue(  
cssStyleProperty )

getDriver()

getId()

getRect()

getTagName()

getText()

isDisplayed()

isEnabled()

isSelected()

sendKeys(  
...args )

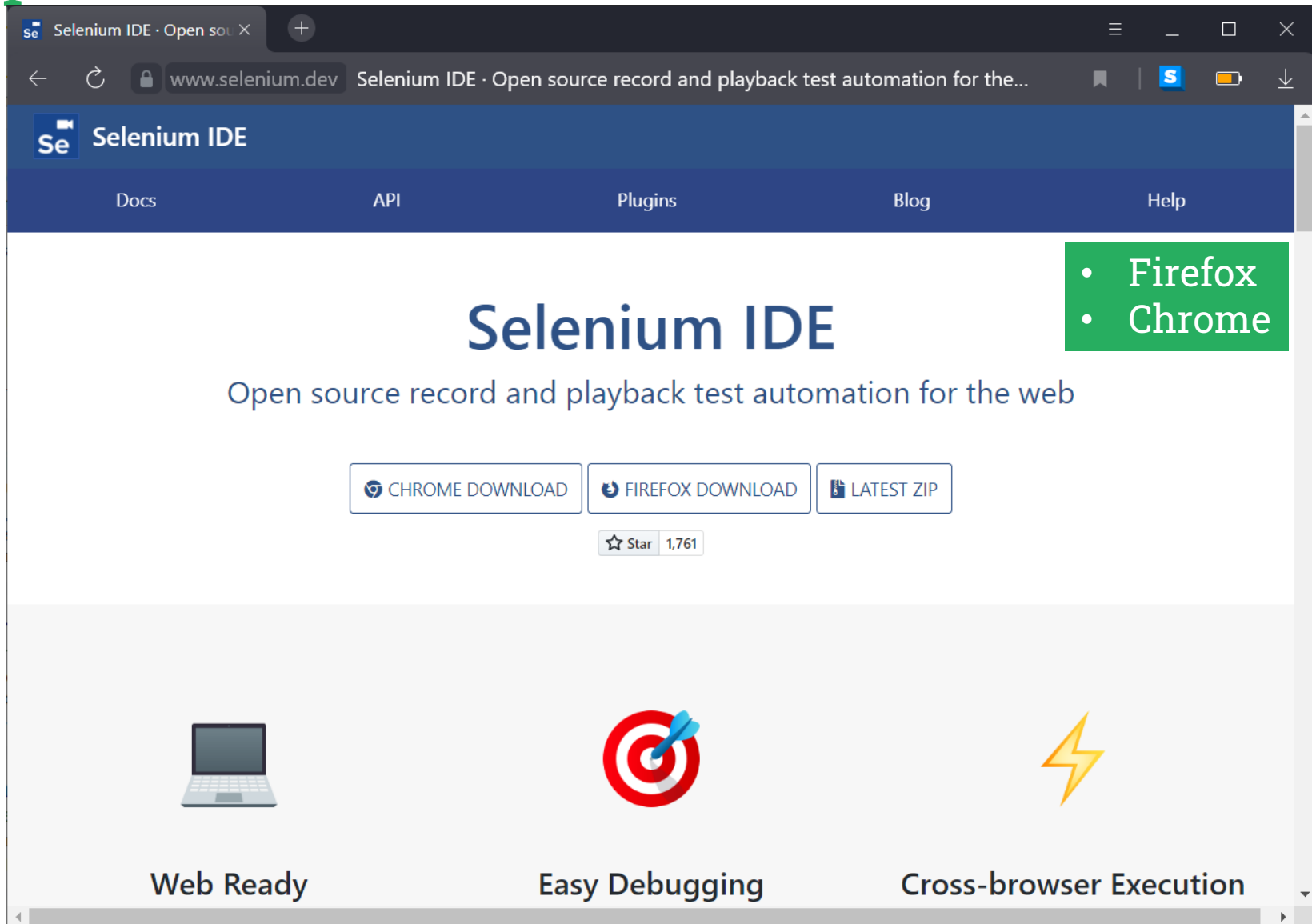
submit()

takeScreenshot(  
scroll )

[https://www.selenium.dev/selenium/docs/api/javascript/module/selenium-webdriver/index\\_exports\\_WebElement.html](https://www.selenium.dev/selenium/docs/api/javascript/module/selenium-webdriver/index_exports_WebElement.html)

# Selenium IDE – запись и проигрывание скриптов

55



<https://www.selenium.dev/selenium-ide/>

# Selenium IDE. Запись теста

56

```
{
  "id": "eda8dc7d-4713-4638-8c77-06426e0ff0c3",
  "version": "2.0",
  "name": "yandex",
  "url": "https://yandex.ru",
  "tests": [{
    "id": "4f1d4be0-f219-49d1-8774-3a8fa3c7275f",
    "name": "cheese",
    "commands": [{
      "id": "c177ad71-8ec0-42f7-a74a-1537508e25ad",
      "comment": "",
      "command": "open",
      "target": "/",
      "targets": [],
      "value": ""
    }, {
      "id": "87f003c3-3065-48c0-b7a3-19364836f6dd",
      "comment": "",
      "command": "setWindowSize",
      "target": "1552x840",
      "targets": [],
      "value": ""
    }, {
      "id": "09da83b9-4062-470e-8713-e10c8709aaf0",
      "comment": "",
      "command": "mouseOver",

```

...

В данном примере  
создан проект **yandex**,  
главная страница  
**https://yandex.ru**,  
выполнен поиск **cheese**.  
Проект сохранен в файл  
**yandex.side**

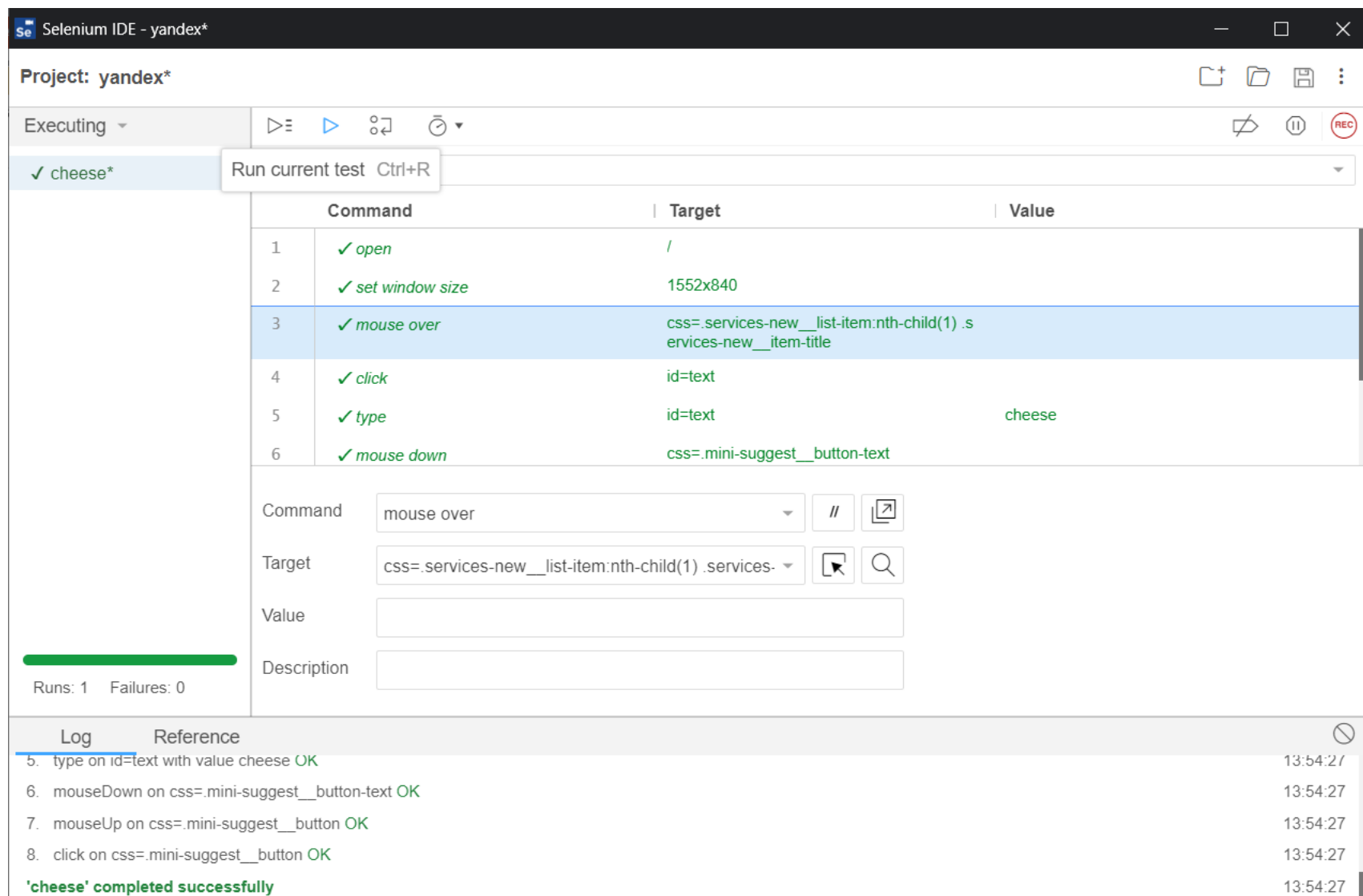
## Основные поддерживаемые типы команд:

- Нажатие на ссылку  
щелчком мыши
- Ввод значения
- Выбор значения из  
выпадающего списка
- Нажатие флажков или  
переключение кнопок  
переключателей



# Selenium IDE. Выполнение теста

57



Selenium IDE - yandex\*

Project: yandex\*

Executing ▾

Run current test Ctrl+R

	Command	Target	Value
1	✓ open	/	
2	✓ set window size	1552x840	
3	✓ mouse over	css=.services-new__list-item:nth-child(1) .services-new__item-title	
4	✓ click	id=text	
5	✓ type	id=text	cheese
6	✓ mouse down	css=.mini-suggest__button-text	

Command: mouse over

Target: css=.services-new\_\_list-item:nth-child(1) .services-

Value:

Description:

Runs: 1 Failures: 0

Log Reference

- 5. type on id=text with value cheese OK 13:54:27
- 6. mouseDown on css=.mini-suggest\_\_button-text OK 13:54:27
- 7. mouseUp on css=.mini-suggest\_\_button OK 13:54:27
- 8. click on css=.mini-suggest\_\_button OK 13:54:27
- 'cheese' completed successfully 13:54:27

**Альтернатива – запуск Selenium IDE тестов из командной строки:**

```
npm i -g selenium-side-runner  
selenium-side-runner yandex.side
```

- **open** – открыть страницу с URL
- **click** – выполнить щелчок и при необходимости ожидание загрузки новой страницы
- **type** – ввод
- **sendKeys** – ввод с клавиатуры
- **verifyTitle/assertTitle** – Проверка заголовка текущей страницы
- **verifyTextPresent** – проверка, что указанный текст существует
- **verifyElementPresent** – проверка, что указанный элемент UI существует и отображается на странице
- **verifyText** – проверка текста и соответствующих тегов HTML на странице
- **verifyTable** – проверка ожидаемого содержимого таблицы

# Selenium IDE. Поиск элементов на странице

- По идентификатору
  - **identifier**=<ИД>
  - **id**=<ИД>
- По имени
  - **name**=<ИМЯ>
- По XPath
  - **xpath**=/html/body/form – абсолютный путь
  - **//form** – относительный путь
  - **xpath**=//form[@id='loginForm'] – элемент с заданным атрибутом
- По тексту ссылки
  - **link**=<ТЕКСТ ССЫЛКИ>
- По CSS
  - **css**=form#loginForm – элемент с заданным ID
  - **css**=input[name='username'] – элемент с заданным атрибутом

W3schools

Tutorials ▼ References ▼ Exercises ▼ Videos Website Paid Courses Log in

HTML CSS JAVASCRIPT SQL PYTHON PHP BOOTSTRAP HOW TO

DOM Node List  
DOM Traversing  
DOM Navigating  
DOM Get Values  
DOM Change Nodes  
DOM Remove Nodes  
DOM Replace Nodes  
DOM Create Nodes  
DOM Add Nodes  
DOM Clone Nodes  
DOM Examples

XPath Tutorial

**XPath Introduction**

XPath Nodes  
XPath Syntax  
XPath Axes  
XPath Operators  
XPath Examples

## XPath Tutorial

[< Previous](#) [Next >](#)

### What is XPath?

XPath is a major element in the XSLT standard.

XPath can be used to navigate through elements and attributes in an XML document.

# E2E фреймворк тестирования Angular

61



<https://protractor.angular.io/>

# Protractor. Настройка

- Установка (protractor, webdriver-manager)
  - `npm install -g protractor`
  - Здесь **webdriver-manager** – инструмент доступа к **Selenium Server**
- Загрузка webdriver
  - `webdriver-manager update`
- Запуск сервера (`http://localhost:4444/wd/hub`)
  - `webdriver-manager start`
- Конфигурация запуска (**conf.js**)

```
exports.config = {  
  seleniumAddress: 'http://localhost:4444/wd/hub',  
  specs: ['todo-spec.js']  
};
```

- Запуск
  - `protractor conf.js`

todo-spec.js



# Protractor. Пример теста

## todo-spec.js

```
describe('angularjs homepage todo list', function() {  
  it('should add a todo', function() {  
    browser.get('https://angularjs.org');  
    element(by.model('todoList.todoText')).sendKeys('write first protractor test');  
    element(by.css('[value="add"]')).click();  
    var todoList = element.all(by.repeater('todo in todoList.todos'));  
    expect(todoList.count()).toEqual(3);  
    expect(todoList.get(2).getText()).toEqual('write first protractor test');  
    // You wrote your first test, cross it off the list  
    todoList.get(2).element(by.css('input')).click();  
    var completedAmount = element.all(by.css('done-true'));  
    expect(completedAmount.count()).toEqual(2);  
  });  
});
```

## Не напоминает тест в Selenium?

Дополнительно о тестировании с помощью Protractor:

<https://medium.com/medialesson/end-to-end-e2e-tests-in-angular-application-using-protractor-3fd2501bb3c0>

# Headless браузеры

<b>Google Puppeteer</b>	<a href="https://developers.google.com/web/tools/puppeteer/">https://developers.google.com/web/tools/puppeteer/</a> <a href="https://github.com/puppeteer/puppeteer">https://github.com/puppeteer/puppeteer</a>
<b>Google Chrome</b>	
<b>Firefox</b>	
<b>PhantomJS</b>	<a href="http://phantomjs.org/">http://phantomjs.org/</a>
<b>HtmlUnit</b>	<a href="http://htmlunit.sourceforge.net/">http://htmlunit.sourceforge.net/</a>
<b>Splinter</b>	<a href="https://splinter.readthedocs.io/en/latest/">https://splinter.readthedocs.io/en/latest/</a>
<b>jQueryBrowserDriver</b>	<a href="https://github.com/MachinePublishers/jBrowserDriver">https://github.com/MachinePublishers/jBrowserDriver</a>

<https://habr.com/ru/company/otus/blog/444248/>

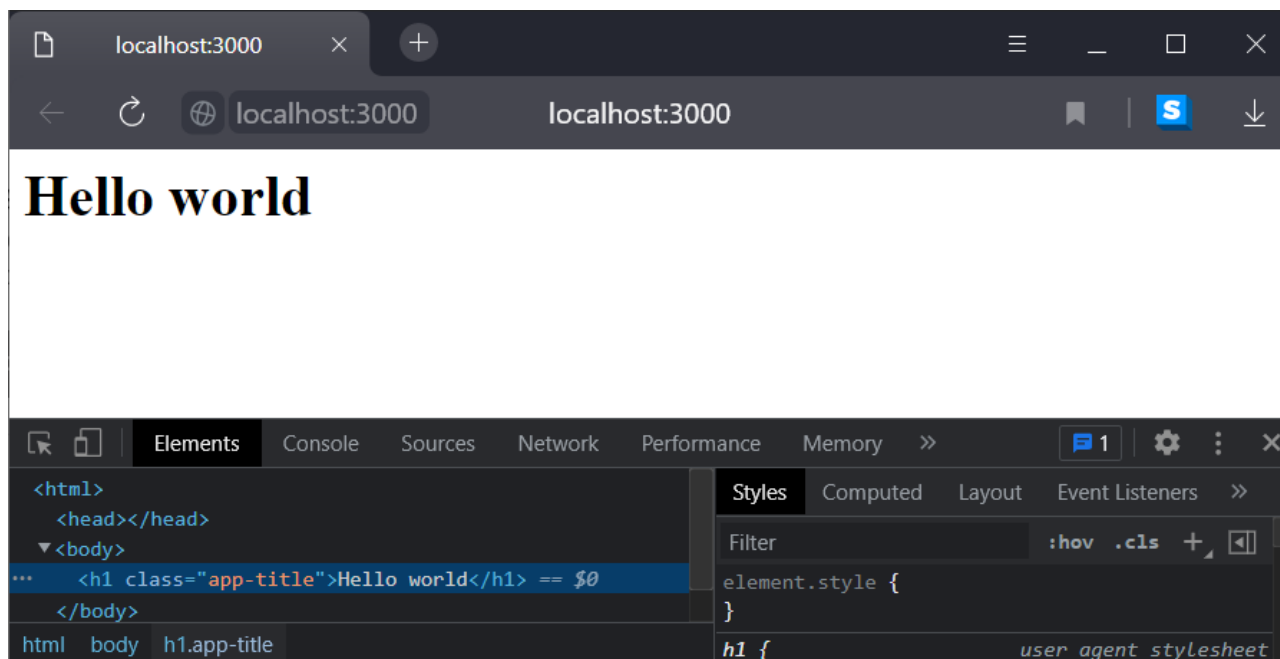


# Тестирование в Jest с использованием Google Puppeteer

- Установить **Jest**
  - `npm install --save-dev jest jest-cli`
- Установить **Puppeteer**
  - `npm install --save-dev puppeteer`
    - загрузился Chromium r901912 - 172.2 Mb
- В папке **src** создаём тестовое приложение
  - `server.js`
- В папке **tests** создаём тест
  - `puppeteer.test.js`
- Запуск тестов
  - `jest`
  - (в моём случае был настроен scripts, поэтому запуск `npm test`)

# Приложение, которое будет тестироваться

```
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("<h1 class='app-title'>Hello world</h1>");
});
```



<http://localhost:3000/>

# Тест с использованием headless puppeteer

```
const puppeteer = require("puppeteer")
```

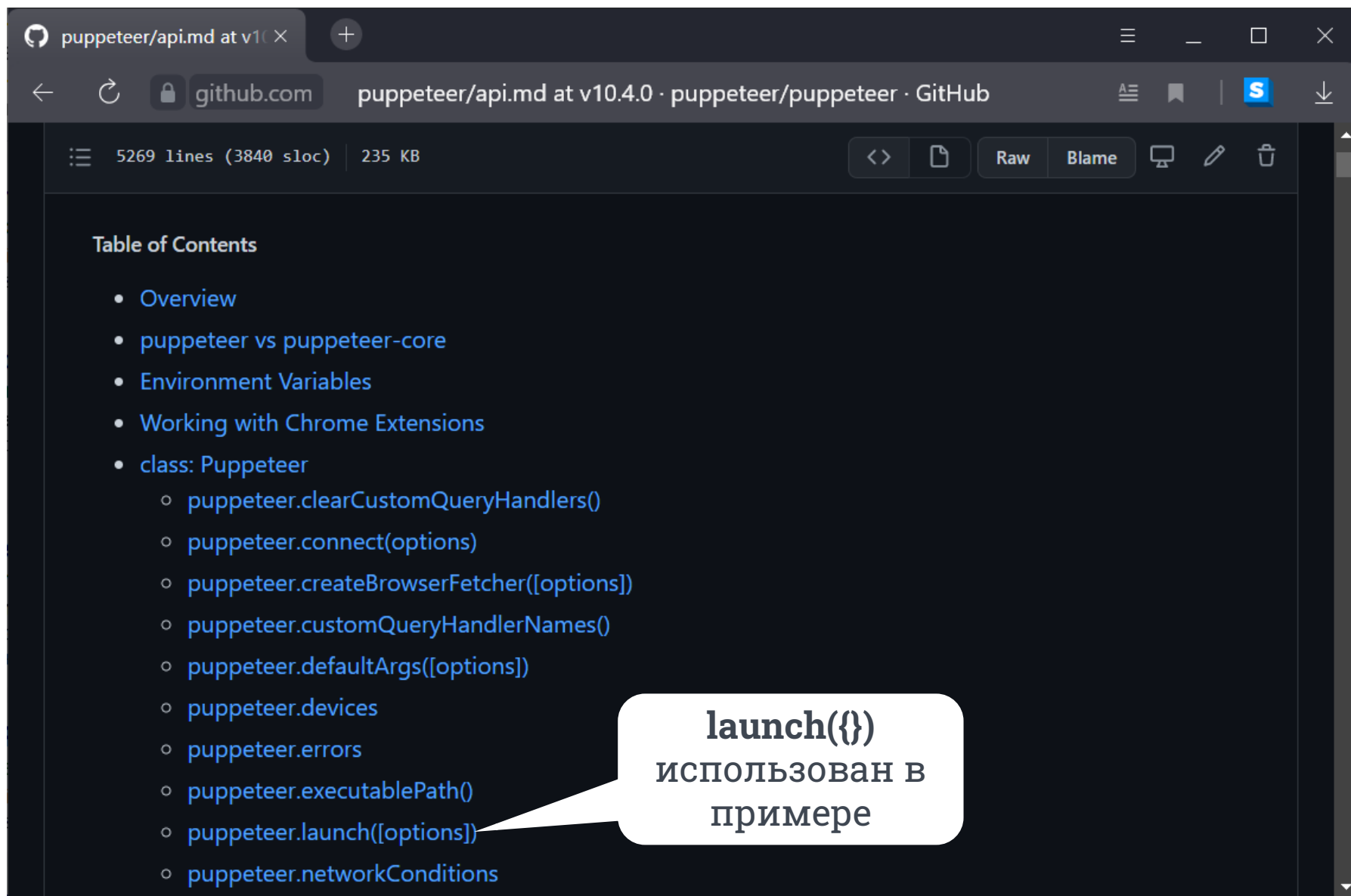
```
test("Headless test", async () => {  
  const browser = await puppeteer.launch({  
    headless: true  
  }) // если поставить false, то запустится браузер  
  const page = await browser.newPage()  
  await page.goto("http://localhost:3000")  
  await page.waitForSelector(".app-title")  
  
  const html = await page.$eval(".app-title", e => e.innerHTML)  
  expect(html).toBe("Hello world")  
}, 16000)
```

API Puppeteer



# API Puppeteer

68



The screenshot shows a web browser displaying the Puppeteer API documentation on GitHub. The browser's address bar shows the URL `github.com/puppeteer/puppeteer/blob/main/docs/api.md`. The page header indicates the file is `puppeteer/api.md` at version `v10.4.0`, with a file size of 235 KB and 5269 lines (3840 sloc). The main content is a 'Table of Contents' with a list of links. A white callout bubble points to the `puppeteer.launch([options])` link, containing the text `launch({})` and 'ИСПОЛЬЗОВАН В примере' (Used in the example).

Table of Contents

- Overview
- puppeteer vs puppeteer-core
- Environment Variables
- Working with Chrome Extensions
- class: Puppeteer
  - `puppeteer.clearCustomQueryHandlers()`
  - `puppeteer.connect(options)`
  - `puppeteer.createBrowserFetcher([options])`
  - `puppeteer.customQueryHandlerNames()`
  - `puppeteer.defaultArgs([options])`
  - `puppeteer.devices`
  - `puppeteer.errors`
  - `puppeteer.executablePath()`
  - `puppeteer.launch([options])`
  - `puppeteer.networkConditions`

<https://github.com/puppeteer/puppeteer/blob/main/docs/api.md>

# Postman – тестирование запросов к web-серверу

69

- GET
- POST
- PUT
- DELETE

## Build

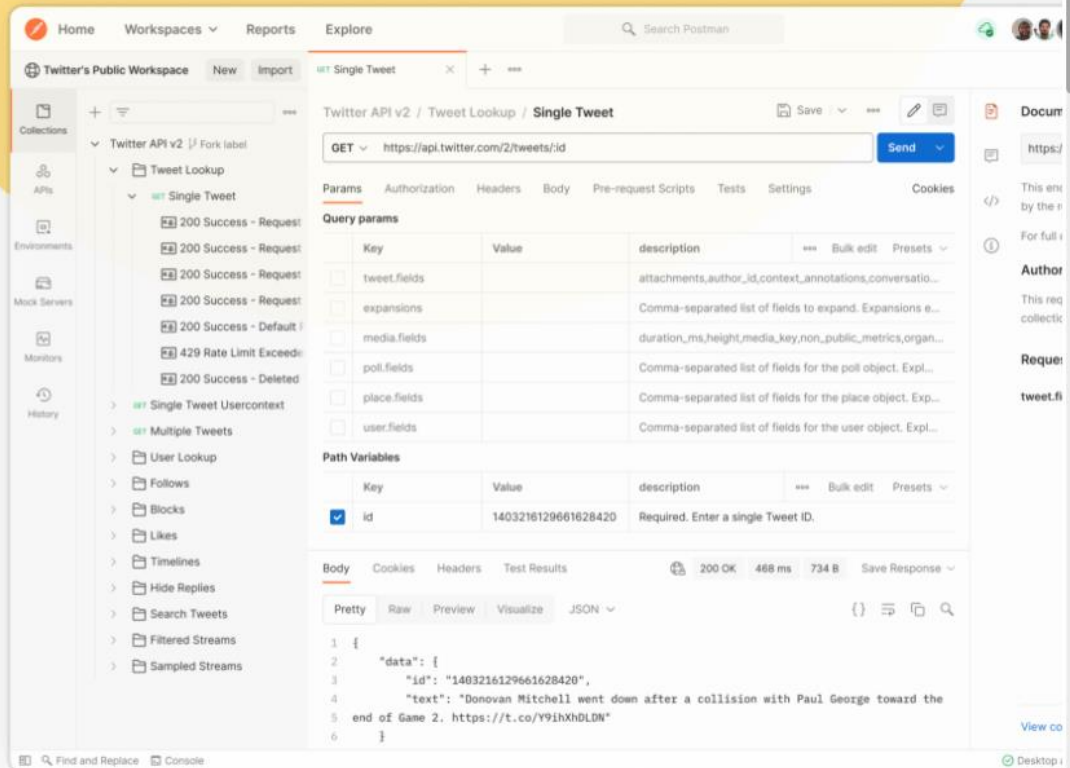
## APIs together

Over 15 million developers use Postman.

Get started by signing up or downloading the desktop app.

[Sign Up for Free](#)

Download the desktop app



<https://www.postman.com/>  
<https://habr.com/ru/company/kolesa/blog/351250/>

# Вопросы для самопроверки

70

- Что такое TDD? В чем отличие от BDD?
- Какие основные команды Assert?
- Чем отличаются Should и Assert?
- Что включает в себя Chai?
- Чем отличаются should и expect?
- Что такое Mocha? Зачем он нужен, если есть Assert, Should, Chai?
- Чем отличаются describe и test? Как ими пользоваться?
- Для чего нужны before, after и их вариации? Как их использовать?
- Что такое Jest? Какие есть отличия от Mocha?
- Для чего нужен Selenium? Чем отличается от Selenium IDE?
- Какие типовые команды есть у Selenium IDE? Как искать элементы на странице?
- Что такое Protractor?
- Что такое headless браузер? Зачем нужен? Как воспользоваться?
- Как выполнять запросы к серверу без браузера?