

Сквозное тестирование или...

...Дорога туда и обратно

Сквозное тестирование или...

...Дорого туда и обратно

Глава один

ЭКСПОЗИЦИЯ ЧЕРЕЗ СИЛУ

Будем знакомы

Леонид Семёнов

Леонид Семёнов



Глава один

3

Леонид Семёнов



Глава один



Леонид Семёнов



Глава один

Леонид Семёнов



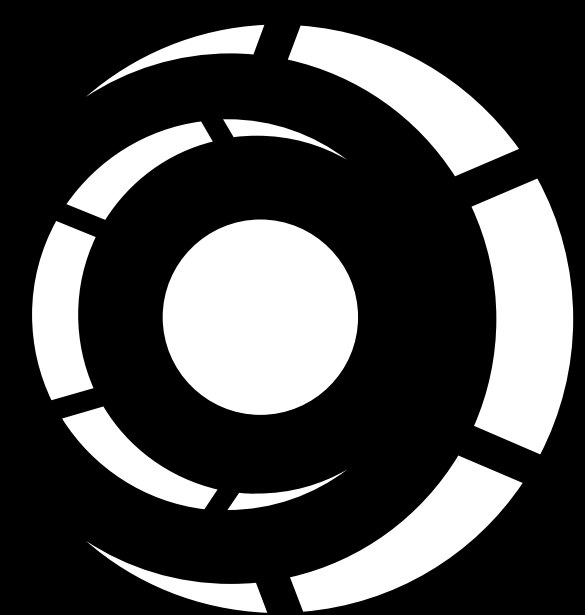
Глава один



Леонид Семёнов

<http://s3rious.ru>

Леонид Семёнов



InvestEngine

- **Стартап**
- **Финтех**
- **Крупные суммы**
- **Зарегулированы**
- **Могут отозвать лицензию**





This
is fine

Глава два

СКВОЗЬ ТЕРНИИ К ЗВЕЗДАМ

Что такое сквозные тесты?

end

to

end

ПОЛЬЗОВАТЕЛЬ

браузер

фронтенд
сервер
бекенд
база данных

ПОЛЬЗОВАТЕЛЬ

браузер

система









1. Я захожу на нужную страницу
2. Дожидаюсь пока она загрузилась
3. Нажимаю на ссылку «Log in»
4. Дожидаюсь что всё загрузилось
5. Оказался ли я на странице «Log in»?
Да \ Нет

1. Я захожу на нужную страницу
2. Дожидаюсь пока она загрузилась
3. Нажимаю на ссылку «Log in»
4. Дожидаюсь что всё загрузилось
5. Оказался ли я на странице «Log in»?

✓ Да \ Нет

1. Я захожу на нужную страницу
2. Дожидаюсь пока она загрузилась
3. Нажимаю на ссылку «log in»
4. Дожидаюсь что произошло
5. Оказался ли я на «log in»?

✓ Да \ Нет



1. Я захожу на нужную страницу
2. Дожидаюсь пока она загрузилась
3. Нажимаю на ссылку «Log in»
4. Дожидаюсь что произошло
5. Оказался ли я на «Log in»?

✓ Да \ Нет



Глава три

Кипарисовая аллея

Быстрый старт с Сайпресом

сypress.io

cypress.io

- Управлять браузером
- Писать тесты
- Запускать тесты
- Отлаживать тесты
- И много чего ещё...

```
1 describe( title: 'Testing TODO list', fn: () : void => {
2   — before( fn: () : void => {
3     — — cy.visit('https://todomvc.com/examples/vanillajs/')
4   — })
5
6   — it( title: 'Can add tasks', fn: () : void => {
7     — — cy.get('.new-todo').type( text: 'task 1').type( text: '{enter}')
8     — — cy.get('.new-todo').type( text: 'task 2').type( text: '{enter}')
9     — — cy.get('.new-todo').type( text: 'task 3').type( text: '{enter}')
10
11    — — cy.get('.todo-list > li').should( chainer: 'have.length', value: 3)
12  — })
13
14  — it( title: 'Can complete task', fn: () : void => {
15    — — cy.get('.todo-list > li').eq( index: 0).find( selector: '.toggle').click()
16
17    — — cy.get('.todo-list > li').eq( index: 0).should( chainer: 'have.class', value: 'completed')
18  — })
19
20  — it( title: 'Can remove task', fn: () : void => {
21    — — cy.get('.todo-list > li').eq( index: 1).find( selector: '.destroy').click( options: { force: true })
22
23    — — cy.get('.todo-list > li').should( chainer: 'have.length', value: 2)
24  — })
25
26  — it( title: 'Has two tasks, completed and not completed', fn: () : void => {
27    — — cy.get('.todo-list > .completed').should( chainer: 'have.length', value: 1)
28    — — cy.get('.todo-list > :not(.completed)').should( chainer: 'have.length', value: 1)
29  — })
30 })
```



```
1 describe( title: 'Testing TODO list', fn: () : void => {
2   — before( fn: () : void => {
3     — — cy.visit( 'https://todomvc.com/examples/vanillajs/' )
4     — })
5
6   — it( title: 'Can add tasks', fn: () : void => {
7     — — cy.get( '.new-todo' ).type( text: 'task 1' ).type( text: '{enter}' )
8     — — cy.get( '.new-todo' ).type( text: 'task 2' ).type( text: '{enter}' )
9     — — cy.get( '.new-todo' ).type( text: 'task 3' ).type( text: '{enter}' )
10
11    — — cy.get( '.todo-list > li' ).should( chainer: 'have.length', value: 3 )
12    — })
13
14   — it( title: 'Can complete task', fn: () : void => {
15     — — cy.get( '.todo-list > li' ).eq( index: 0 ).find( selector: '.toggle' ).click()
```

```
1 describe( title: 'Testing TODO list', fn: () : void => {
2   — before( fn: () : void => {
3     — — cy.visit( 'https://todomvc.com/examples/vanillajs/' )
4     — })
5
6   — it( title: 'Can add tasks', fn: () : void => {
7     — — cy.get( '.new-todo' ).type( text: 'task 1' ).type( text: '{enter}' )
8     — — cy.get( '.new-todo' ).type( text: 'task 2' ).type( text: '{enter}' )
9     — — cy.get( '.new-todo' ).type( text: 'task 3' ).type( text: '{enter}' )
10
11    — — cy.get( '.todo-list > li' ).should( chainer: 'have.length', value: 3 )
12    — })
13
14   — it( title: 'Can complete task', fn: () : void => {
15     — — cy.get( '.todo-list > li' ).eq( index: 0 ).find( selector: '.toggle' ).click()
```

cypress/integration/todo.spec.js

```

(xhr) POST 200 /j/collect?v=1&v=j89&a=1471892777&t=pageview&s=1&d1...
TEST BODY
1 get .new-todo
2 -type task 1
3 -type {enter}
4 get .new-todo
5 -type task 2
6 -type {enter}
7 get .new-todo
8 -type task 3
9 -type {enter}
10 get .todo-list > li
11 -assert expected [ <li>, 2 more... ] to have a length of 3

```

Can complete task

```

TEST BODY
1 get .todo-list > li
2 -eq 0
3 -find .toggle
4 -click
5 get .todo-list > li
6 -eq 0
7 -assert expected <li.completed> to have class completed

```

Can remove task

```

TEST BODY
1 get .todo-list > li
2 -eq 1
3 -find .destroy
4 -click {force: true}
5 get .todo-list > li
6 -assert expected [ <li.completed>, 1 more... ] to have a length of 2

```

Has two tasks, completed and not completed

```

TEST BODY
1 get .todo-list > .completed
2 -assert expected <li.completed> to have a length of 1
3 get .todo-list > :not(.completed)

```

JavaScript

Vanilla JavaScript Example
[Source](#)

“JavaScript® (often shortened to JS) is a lightweight, interpreted, object-oriented language with first-class functions, most known as the scripting language for Web pages, but used in many non-browser environments as well such as node.js or Apache CouchDB.”

JavaScript

If you have other helpful links to share, or find any of the links above no longer work, please *let us know*.

todos

▼ What needs to be done?

- task 1
- task 3

1 item left All Active Completed Clear completed

Double-click to edit a todo
 Created by Oscar Godson
 Refactored by Christoph Burgmer
 Part of TodoMVC

6 сценариев

6 минут 28 секунды

≈1 минуту 4 секунды

Глава четыре

ДЕНЬ, КОГДА ЗЕМЛЯ ОСТАНОВИЛАСЬ

Проблемы и решения

АГЕНТЫ **5** 

Агенты 5  **В очереди 111**

73 сценария

1 час 14 минут

≈1 минута

Мощнее агенты

Больше агентов

Параллелизация

~~Мощнее агенты~~

~~Больше агентов~~

~~Параллелизация~~







Глава пять

Дальше по сценарию

Плейрайт как Бог из машины



Playwright

 **Playwright**

- **Управлять браузером**



Playwright

- Управлять браузером



JEST

- Писать тесты
- Запускать тесты
- Отлаживать тесты

```

1 describe( title: 'Testing TODO list', fn: () : void => {
2   — beforeAll( fn: async () : Promise<void> => {
3     — await page.goto( url: 'https://todomvc.com/examples/vanillajs/'
4   — })
5
6   — async function createTask( title ) : Promise<void> {
7     — const field : any = await page.$( selector: '.new-todo' )
8
9     — await field.fill( title )
10    — await field.press( 'Enter' )
11  — }
12
13 — it( title: 'Can add tasks', fn: async () : Promise<...> => {
14 — — await createTask( title: 'task 1' )
15 — — await createTask( title: 'task 2' )
16 — — await createTask( title: 'task 3' )
17
18 — — expect( await page.$$$( selector: '.todo-list > li' ) ).toHaveLength( expected: 3 )
19 — })
20
21 — it( title: 'Can complete task', fn: async () : Promise<void> => {
22 — — const tasks : any = await page.$$$( selector: '.todo-list > li' )
23 — — const checkbox : any = await tasks[1].$( html: '.toggle' )
24 — — await checkbox.click()
25
26 — — expect( await tasks[1].getAttribute( 'class' ) ).toEqual( expected: 'completed' )
27 — })
28
29 — it( title: 'Can remove task', fn: async () : Promise<void> => {
30 — — const tasks : any = await page.$$$( selector: '.todo-list > li' )
31 — — await tasks[0].hover()
32 — — const deleteButton : any = await tasks[0].$( html: '.destroy' )
33 — — await deleteButton.click()
34
35 — — expect( await page.$$$( selector: '.todo-list > li' ) ).toHaveLength( expected: 2 )
36 — })
37
38 — it( title: 'Has two tasks, completed and not completed', fn: async () : Promise<void> => {
39 — — expect( await page.$$$( selector: '.todo-list > .completed' ) ).toHaveLength( expected: 1 )
40 — — expect( await page.$$$( selector: '.todo-list > :not(.completed)' ) ).toHaveLength( expected: 1 )
41 — })
42 })

```

```
1 describe( title: 'Testing TODO list', fn: () : void => {
2 — beforeAll( fn: async () : Promise<void> => {
3 — — await page.goto( url: 'https://todomvc.com/examples/vanillajs/' )
4 — })
5
6 — async function createTask( title ) : Promise<void> {
7 — — const field : any = await page.$( selector: '.new-todo' )
8
9 — — await field.fill( title )
10 — — await field.press( 'Enter' )
11 — }
12
13 — it( title: 'Can add tasks', fn: async () : Promise<...> => {
14 — — await createTask( title: 'task 1' )
15 — — await createTask( title: 'task 2' )
16 — — await createTask( title: 'task 3' )
17
18 — — expect( await page.$$ ( selector: '.todo-list > li' ) ).toHaveLength( expected: 3 )
19 — }
```

```
1 describe( title: 'Testing TODO list', fn: () : void => {
2 — beforeAll( fn: async () : Promise<void> => {
3 — — await page.goto( url: 'https://todomvc.com/examples/vanillajs/' )
4 — })
5
6 — async function createTask( title ) : Promise<void> {
7 — — const field : any = await page.$( selector: '.new-todo' )
8
9 — — await field.fill( title )
10 — — await field.press( 'Enter' )
11 — }
12
13 — it( title: 'Can add tasks', fn: async () : Promise<...> => {
14 — — await createTask( title: 'task 1' )
15 — — await createTask( title: 'task 2' )
16 — — await createTask( title: 'task 3' )
17
18 — — expect( await page.$$ ( selector: '.todo-list > li' ) ).toHaveLength( expected: 3 )
19 — }
```

JavaScript

Vanilla JavaScript Example
[Source](#)

“JavaScript® (often shortened to JS) is a lightweight, interpreted, object-oriented language with first-class functions, most known as the scripting language for Web pages, but used in many non-browser environments as well such as node.js or Apache CouchDB.”

JavaScript

If you have other helpful links to share, or find any of the links above no longer work, please [let us know](#).

todos

▼ What needs to be done?

- task 1
- task 2
- task 3

3 items left All Active Completed

Double-click to edit a todo
Created by Oscar Godson
Refactored by Christoph Burgmer
Part of TodoMVC

JavaScript

Vanilla JavaScript Example

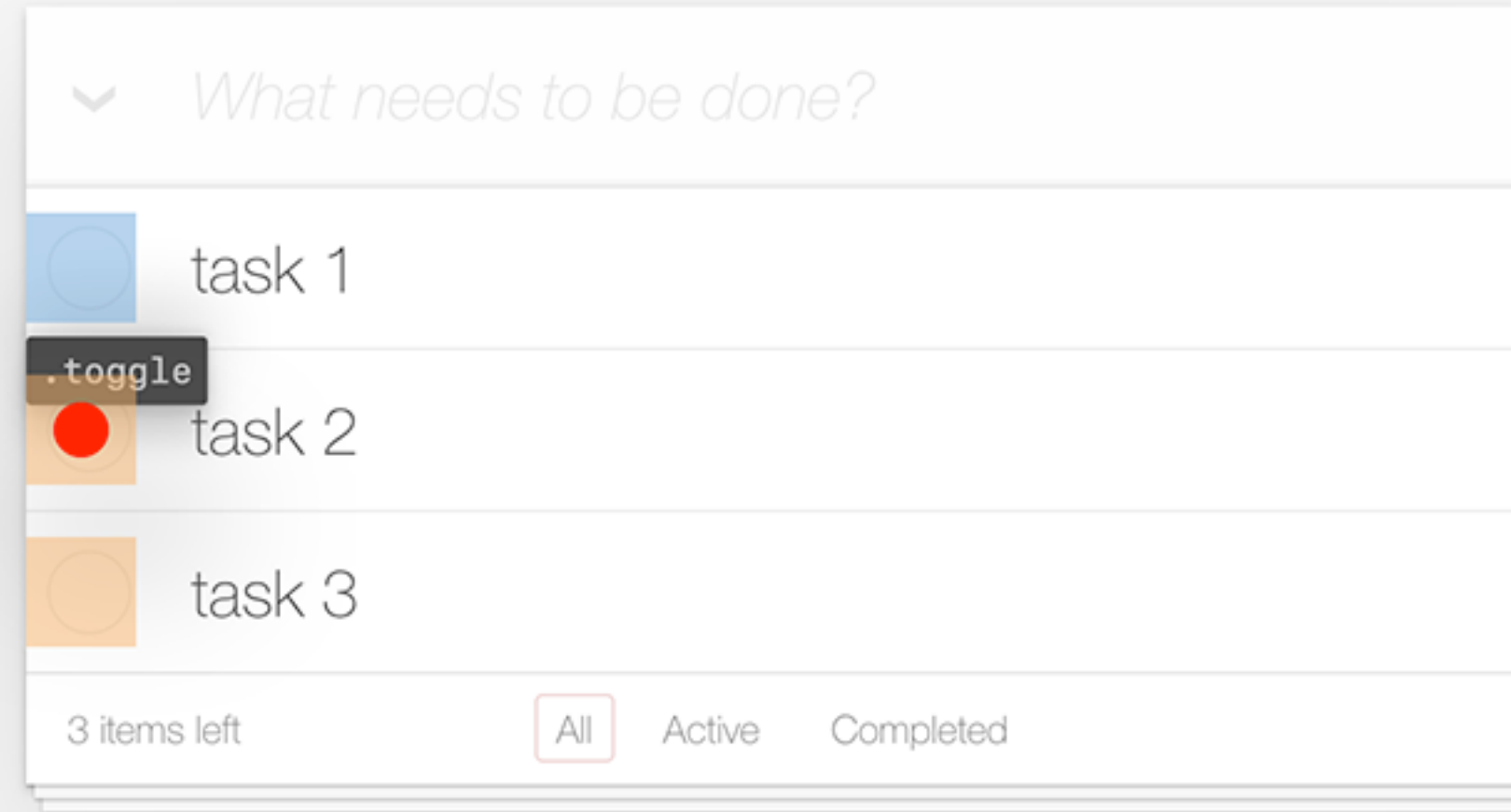
[Source](#)

“JavaScript® (often shortened to JS) is a lightweight, interpreted, object-oriented language with first-class functions, most known as the scripting language for Web pages, but used in many non-browser environments as well such as node.js or Apache CouchDB.”

JavaScript

If you have other helpful links to share, or find any of the links above no longer work, please [let us know](#).

todos



Double-click to edit a todo
Created by Oscar Godson
Refactored by Christoph Burgmer
Part of TodoMVC

```
Record [copy] [play] [pause] [refresh] todo.test.js
1 describe('Testing TODO list', () => {
2   beforeAll(async () => {
3     await page.goto('https://todomvc.com/examples/vanillajs/')
4   })
5
6   async function createTask(title) {
7     const field = await page.$('.new-todo')
8
9     await field.fill(title)
10    await field.press('Enter')
11  }
12
13  it('Can add tasks', async () => {
14    await createTask('task 1')
15    await createTask('task 2')
16    await createTask('task 3')
17
18    expect(await page.$$('.todo-list > li')).toHaveLength(3)
19  })
20
21  it('Can complete task', async () => {
22    const tasks = await page.$$('.todo-list > li')
23    const checkbox = await tasks[1].$('.toggle')
24    await checkbox.click()
25
26    expect(await tasks[1].getAttribute('class')).toEqual('completed')
27  })
28
29  it('Can remove task', async () => {
30    const tasks = await page.$$('.todo-list > li')
31    await tasks[0].hover()
32    const deleteButton = await tasks[0].$('.destroy')
33    await deleteButton.click()
34
35    expect(await page.$$('.todo-list > li')).toHaveLength(2)
36  })
37
38  it('Has two tasks, completed and not completed', async () => {
39    expect(await page.$$('.todo-list > .completed')).toHaveLength(1)
40    expect(await page.$$('.todo-list > :not(.completed)')).toHaveLength(2)
41  })
42 })
43
```

Explore .toggle

```
> page.$(.new-todo) ✓ — 29ms
> elementHandle.fill ✓ — 11ms
> elementHandle.press ✓ — 11ms
> page.$(.new-todo) ✓ — 16ms
> elementHandle.fill ✓ — 8ms
> elementHandle.press ✓ — 7ms
> page.$(.new-todo) ✓ — 23ms
```

69 сценариев

10 минут 30 секунд

≈32 секунды

69 сценариев

10 минут 30 секунд

≈32 секунды ≈8 секунд

Глава шесть

И ЦЕЛОГО МИРА МАЛО

Делаем выводы

Ручное 🖐️

Нет разработки

Нет поддержки

Люди страдают,
и ошибаются

Cypress 🗡️

Нужно разработать

Нужно поддерживать

Машины страдают,
и не ошибаются

Playwright 🪓

Нужно разработать

Нужно поддерживать

Машины страдают,
и не ошибаются

Cypress

Умеет всё про тесты

Проще для новичка

Однопоточный*

Сложный внутри

Playwright

Умеет меньше, но лучше

Сложнее, нужно въезжать

Многопоточный*

Простой как топор

	HEADLESS				
	PLAYWRIGHT	PUPPETEER	WDIO SELENIUM	WDIO DEVTOOLS	CYPRESS
Mean (s)	13.13	13.22	16.42	16.60	23.64
SD (s)	0.84	1.31	2.23	4.13	1.74
C_v	0.064	0.099	0.136	0.249	0.074
P95 (s)	14.55	14.87	19.94	26.22	26.72
Slower	-	0.68%	25.11%	26.45%	80.08%

<https://blog.checklyhq.com/cypress-vs-selenium-vs-playwright-vs-puppeteer-speed-comparison/>



Follow

Andrey Lushnikov

@aslushnikov

I ❤️ WEB. playwright @ microsoft. Former TL @ Chrome Puppeteer, former eng @ Chrome DevTools English / Russian

📍 San Francisco, CA [🔗 aslushnikov.com](https://aslushnikov.com) 🎂 Born December 1, 1989

📅 Joined October 2009

1.0.0 »→ 6 мая 2020

1.10.0 »→ 25 марта 2021

Один большой релиз в месяц

СКВОЗНЫЕ ТЕСТЫ — НЕ СЛОЖНО

СВОЙ ИНСТРУМЕНТ ДЛЯ ВСЕГО

НЕ БОЙТЕСЬ МЕНЯТЬ ИНСТРУМЕНТ

Эпиклог

Вопросы?

