

UNIT TESTING

Who am i?

Aleksey Karpenko

~7 years in JavaScript

What I expect of you.

- Know some TypeScript

What I expect of you.

- Know some TypeScript
- export / import a function

What I expect of you.

- Know some TypeScript
- export / import a function
- Ask questions

Plan for today

- What are unit tests
- Why we need them
- How to setup your testing env
- How to write + organise your tests
- What are test runners
- What are test matchers
- What are stubs, mocks, spies
- Difference between env's
- Setup's / tare-down's

Who uses tests

- Evolution
- Open Source
- `_.stubString()` ?
- Your pet projects?

How can you describe your code?

How can you describe your code?

- Folder/file names, file structure

How can you describe your code?

- Folder/file names, file structure
- Function/variable names

How can you describe your code?

- Folder/file names, file structure
- Function/variable names
- Types (typescript)

How can you describe your code?

- Folder/file names, file structure
- Function/variable names
- Types (typescript)
- Comments (js-doc)

How can you describe your code?

- Folder/file names, file structure
- Function/variable names
- Types (typescript)
- Comments (js-doc)
- Tests

How can you describe your code?

- Folder/file names, file structure
- Function/variable names
- Types (typescript)
- Comments (js-doc)
- Tests
- Documentation


In the perfect world

- Names + Types + Tests + jsdoc > Documentation

Example 1 Shopping cart

MY BAG

Items are reserved for 60 minutes




£22.95

adidas Originals t-shirt with shattered logo in black

Bk1 - black 1 | M | Qty 1

Save for later



£18.35 ~~£22.95~~

adidas Originals t-shirt with shattered logo in yellow

Ye1 - yellow 1 | M | Qty 1

Save for later

SUB-TOTAL £41.30

TOTAL








Sub-total £41.30

Delivery ⓘ

Standard Delivery (Free) ▼

CHECKOUT

WE ACCEPT:



Got a discount code? Add it in the next step.

Unit Test

- Is testing a smallest possible “Unit” of code

Unit Test

- Is testing a smallest possible “Unit” of code
- Verifies correct behaviour of your code

Unit Test

- Is testing a smallest possible “Unit” of code
- Verifies correct behaviour of your code
- Provides FAST FEEDBACK

Unit Test

- Is testing a smallest possible “Unit” of code
- Verifies correct behaviour of your code
- Provides FAST FEEDBACK
- Reduces development time when more features introduced

Unit Tests

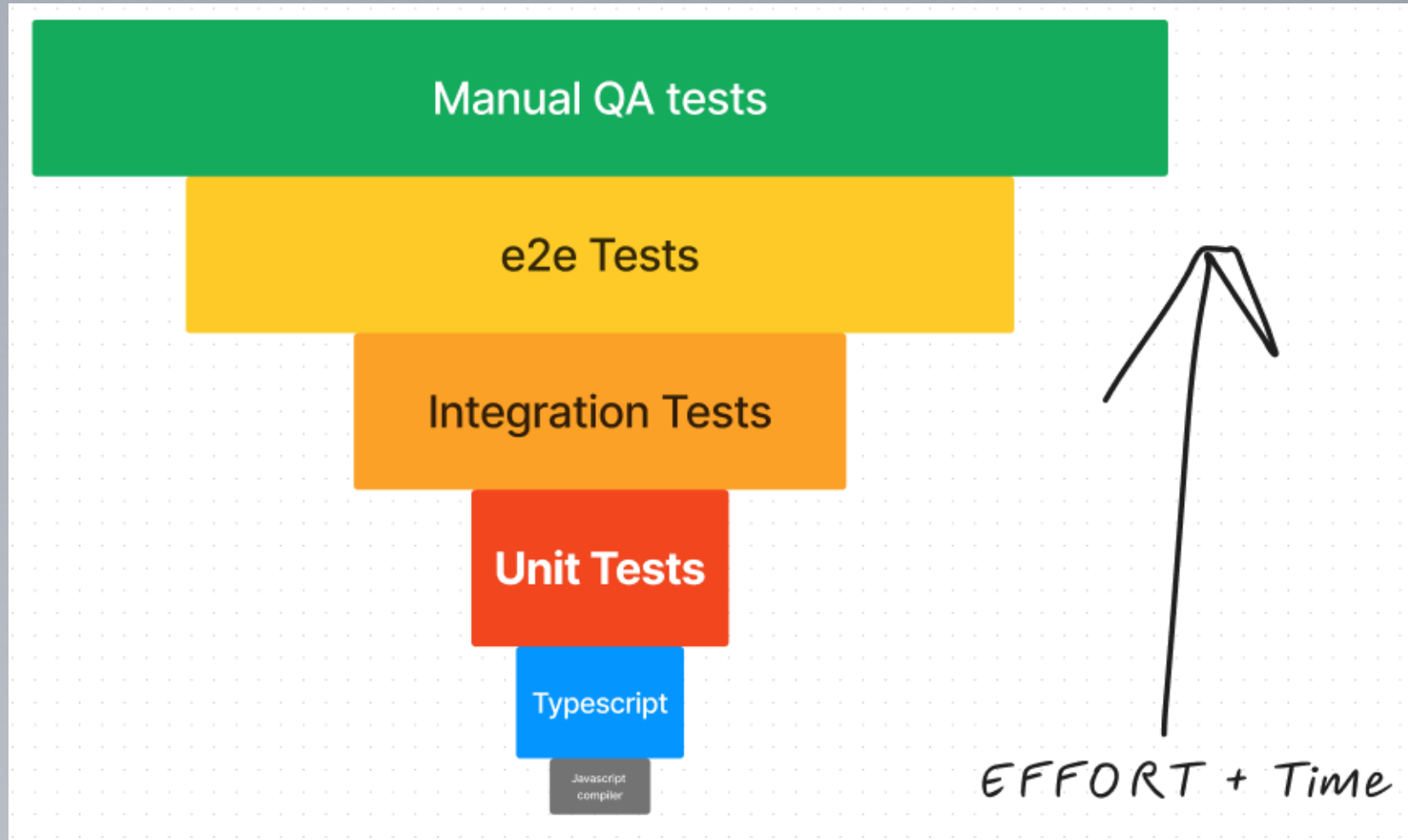
- Should fail with legit problem

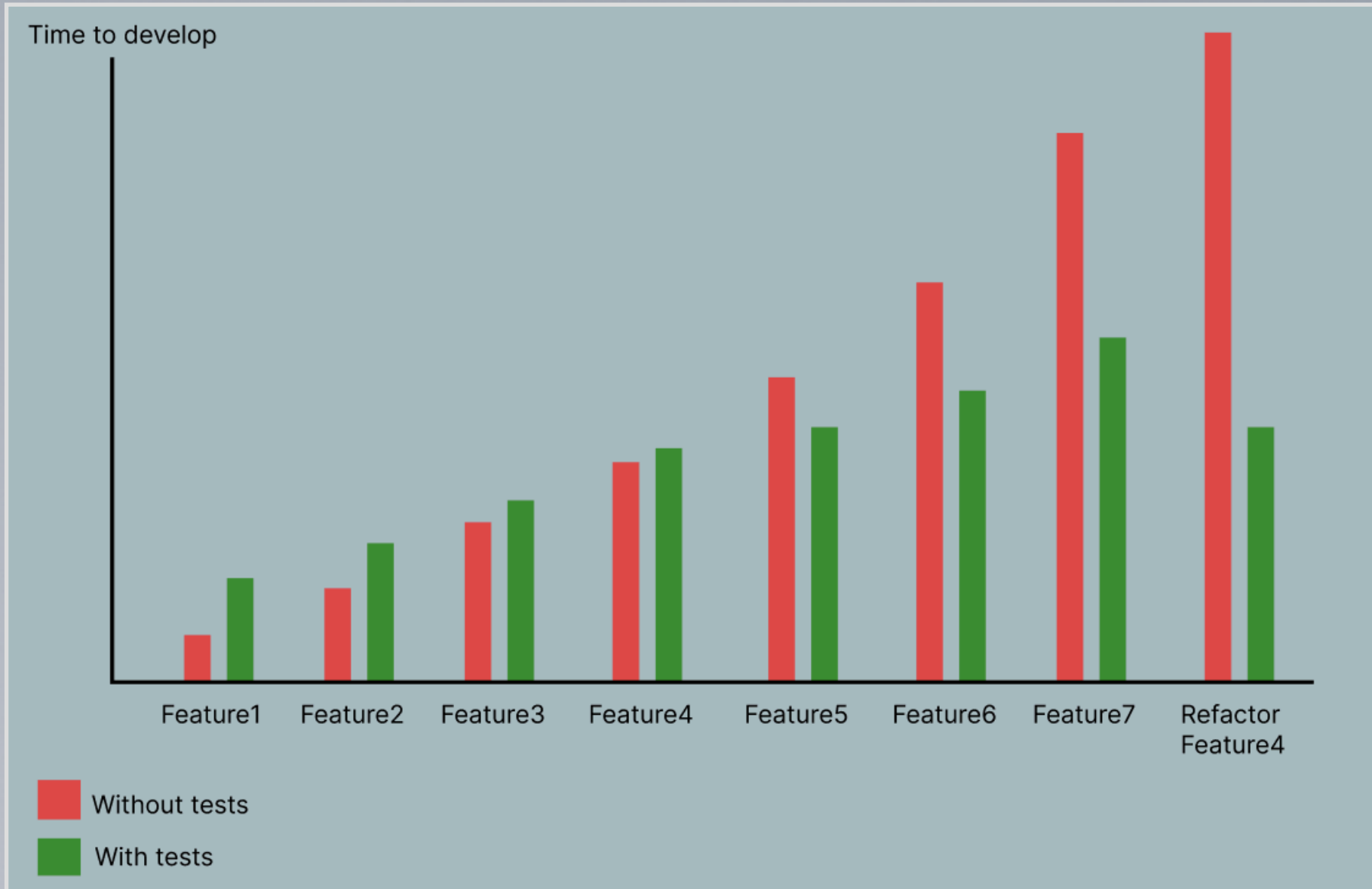
Unit Tests

- Should fail with legit problem
- Are not created equal

Unit Tests

- Should fail with legit problem
- Are not created equal
- Do not equal better design





Example 2 (basic setup)

- `npm i jest @types/jest ts-jest`
- Setup `jest.config.js`
- Setup `tsconfig.json`
- Basic runs + fn's

Testing Libs

- Jest
- Vitest
- Mocha
- AVA
- Karma

Testing Libs

- Cypress
- Playwright
- Selenium
- Webdriver.io
- Puppeteer

Testing Libs

- Enzyme
- @testing-library/react
- jsdom

Unit Tests

- Storybook (components)
- Sinon (Mocks, spy's)
- Chai (assertion)
- “node:assert/strict”
- supertest (nodejs server)
- Polly.JS (HTTP)

TDD / BDD

- Define a test set for the unit first
- Make the tests fail
- Implement the unit
- Verify that the implementation of the unit makes the tests succeed



```
3 const hw = 'Hello World'
4
5 assert(hw === 'Hello World')
6 assert.equal(hw, 'Hello World')
```

Assert



```
1 const hw = 'Hello, world!'
2
3 hw.should.be.a('string')
4 hw.should.equal('Hello, world!')
5 hw.should.have.lengthOf(13)
```

Should



```
1 const hw = 'Hello, world!'
2
3 expect(hw).toBe('Hello, world!')
4 expect(hw).toHaveLength(13)
```

Expect



```
1 const hw = "Hello World";  
2  
3 expect(hw).toBe.a("string");  
4 expect(hw).toEqual("Hello World");  
5 expect(hw).toHaveLength(11);
```



```
1 const hw = "Hello World";  
2  
3 hw.should.be.a("string");  
4 hw.should.equal("Hello World");  
5 hw.should.have.lengthOf(11);
```



```
7 const hw = "Hello World";  
8  
9 expect(hw).toBe(expect.any(String));  
10 expect(hw).toBe("Hello World");  
11 expect(hw).toHaveLength(11);
```



```
1 cy.get(".hello-world")  
2   .should("be.visible")  
3   .and("have.text", "Hello World")  
4   .and("have.class", "hello-world")
```

BDD - Implementation vs Behaviour

- Behaviour - “We don’t care how you come up with the result, we are making sure that its correct”
- Implementation - We don’t care what the result is, we make sure you take the right steps

BD Testing in Jest

- I `describe` my function
 - It `should` behave like this
 - I `expect` my result `to be` that
-
- You chain together natural language assertions

Inattention blindness + tunnel visioning



Avoid Bad testing practices

- Avoid mocking everything
- Avoid testing implementation
- Avoid chasing coverage

Organisation

- describe, it, test
- .skip
- .only
- .todo
- .each
- .failing

Matchers

- `.toBe()` vs `.toEqual()`
- `.toHaveBeenCalled()`
- `.toMatchSnapshot()`
- `expect.any`
- `.not`
- `.resolves` `.rejects`

Matchers `.toBe()` vs `.toEqual()`

`.toBe()` compares object reference
`.toEqual()` compares objects deeply

Use `.toBe()` with primitive values
Use `.toEqual()` with Object types

Mocks / Stubs

- `jest.fn()`
- `jest.spyOn(object, "function")`
- `.mockReturnValue()`
- `jest.requireActual()`

Setup / Taredown

- beforeAll
- beforeEach
- afterEach
- afterAll

Timers

- `.useFakeTimers()`
- `.runOnlyPendingTimers()`
- `.useRealTimers()`
- `.clearAllTimers()`

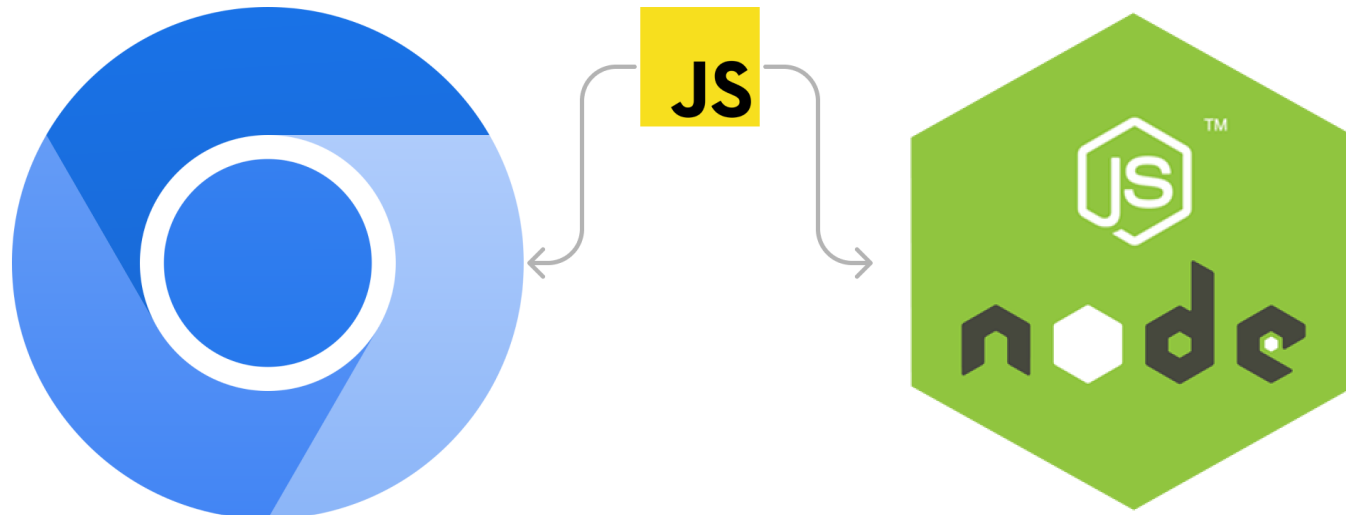
REACT TESTING

Example 4 (react setup)

```
npm i vite @testing-library/jest-dom @testing-library/react @types/  
react @types/react-dom react react-dom jest-environment-jsdom redux  
react-redux @reduxjs/toolkit
```

Example 4 (react setup)

- js-dom vs node env
- Setup jest.config.js
- Setup tsconfig.json





```
1  const jsdom = require('jsdom')
2  const { JSDOM } = jsdom
3
4  const dom = new JSDOM(`<!DOCTYPE html><p>Hello world</p>`)
5
6  const hw = dom.window.document.querySelector('p').textContent
```


Example 4 (react setup)

- render
- query an element
- fireEvent
- act
- waitFor
- match

Example 4 (react setup)

- getBy, queryBy, findBy
- role, label, placeholder, text,
- displayValue, alt, title, testId

Example 5 (Testing Redux)

1. Initialise store
2. Dispatch some actions
3. Verify the state

Example 5 (Testing React with Redux)

1. Initialise store
2. Wrap your component with Provider
3. Dispatch some actions
4. Verify the state is correct
5. Verify the component shows correct things

- Don't Write Comments by CodeAesthetic
- Unit Testing Principles, Practices, and Patterns
- bestofjs.org
- Jest documentation
- Vitest documentation
- Testing Library docs
- Funfunfunction - Unit testing in JavaScript

- <https://github.com/threepointone/react-act-examples/blob/master/sync.md>
- <https://bestofjs.org/>
- <https://jestjs.io/>
- <https://vitest.dev/>
- <https://testing-library.com/>
- <https://www.youtube.com/watch?v=Bf7vDBBOBUA>
- <https://www.youtube.com/watch?v=5iJWOPaNZDA&t=1731s>
- <https://www.youtube.com/watch?v=Eu35xM76kKY>
- https://en.wikipedia.org/wiki/Inattentional_blindness
- https://en.wikipedia.org/wiki/Test-driven_development
- <https://dev.to/thejaredwilcurt/why-you-should-never-use-tobe-in-jest-48ca>

