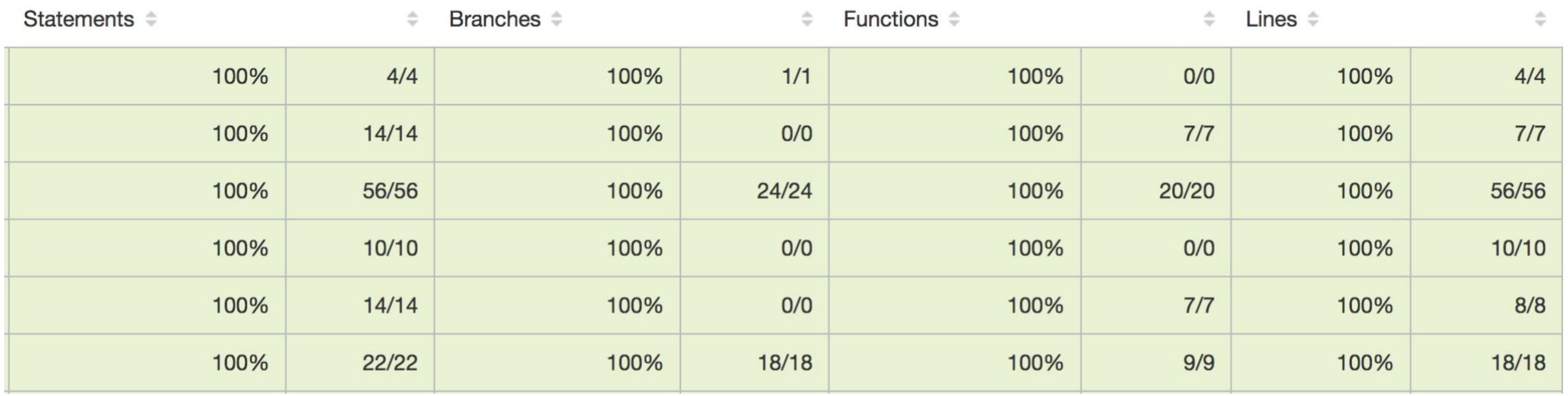
Beyond Code Coverage:

User-Centered UI Testing

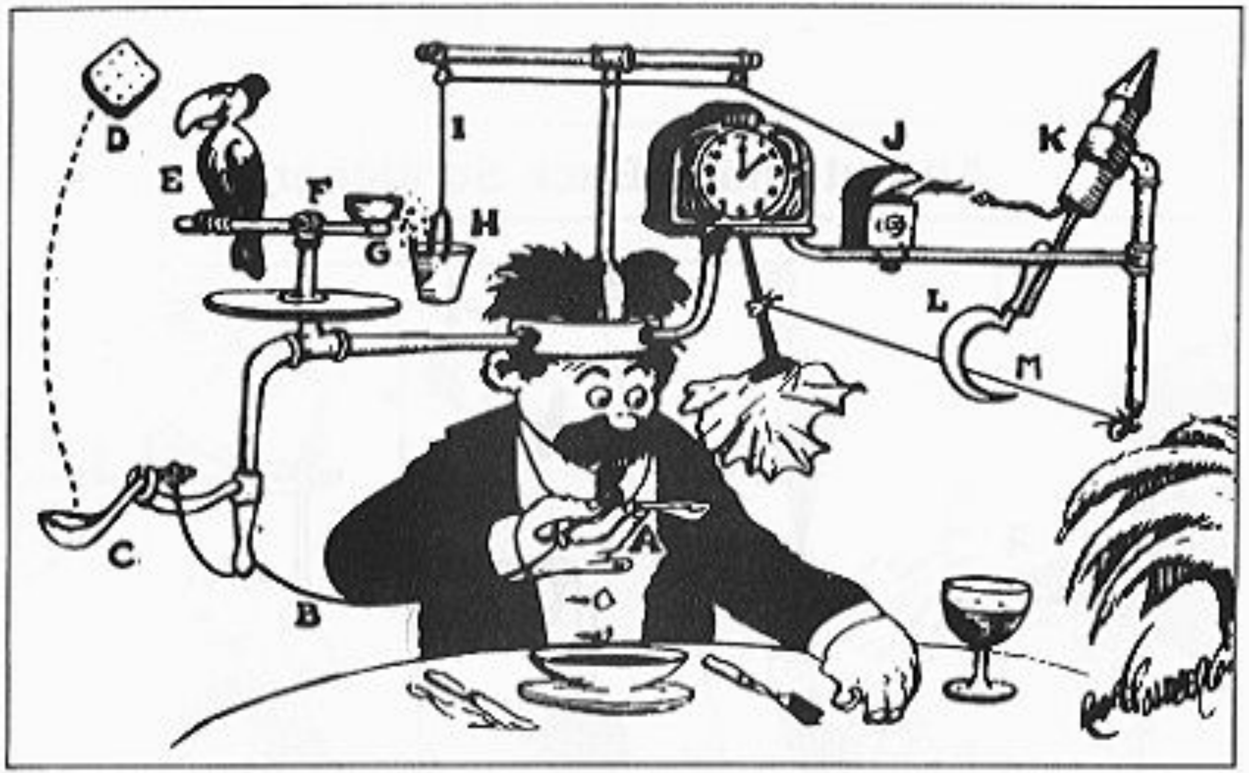
Caitlin Collins & Mike Eastes

NEW

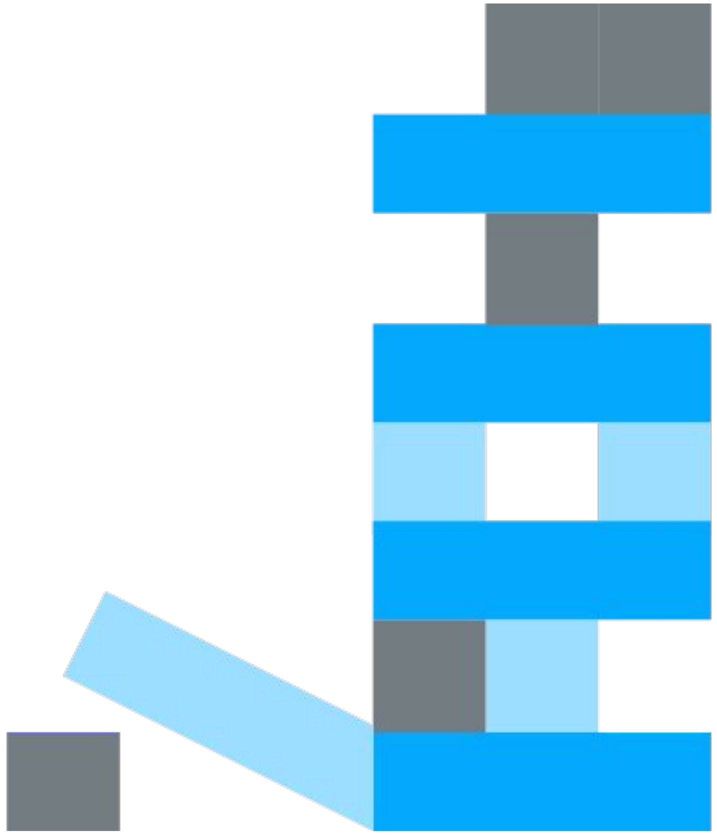


Statements		Branches		Functions		Lines	
100%	4/4	100%	1/1	100%	0/0	100%	4/4
100%	14/14	100%	0/0	100%	7/7	100%	7/7
100%	56/56	100%	24/24	100%	20/20	100%	56/56
100%	10/10	100%	0/0	100%	0/0	100%	10/10
100%	14/14	100%	0/0	100%	7/7	100%	8/8
100%	22/22	100%	18/18	100%	9/9	100%	18/18

Self-Operating Napkin



Cheerios



Developers

Product Owner

QA Engineers

[SPOILER]

This was our team.

Four Lessons

[Earned via ~~blood~~, sweat, and tears.]

Lesson 1:
Redefine a Unit of Work

A piece of code that

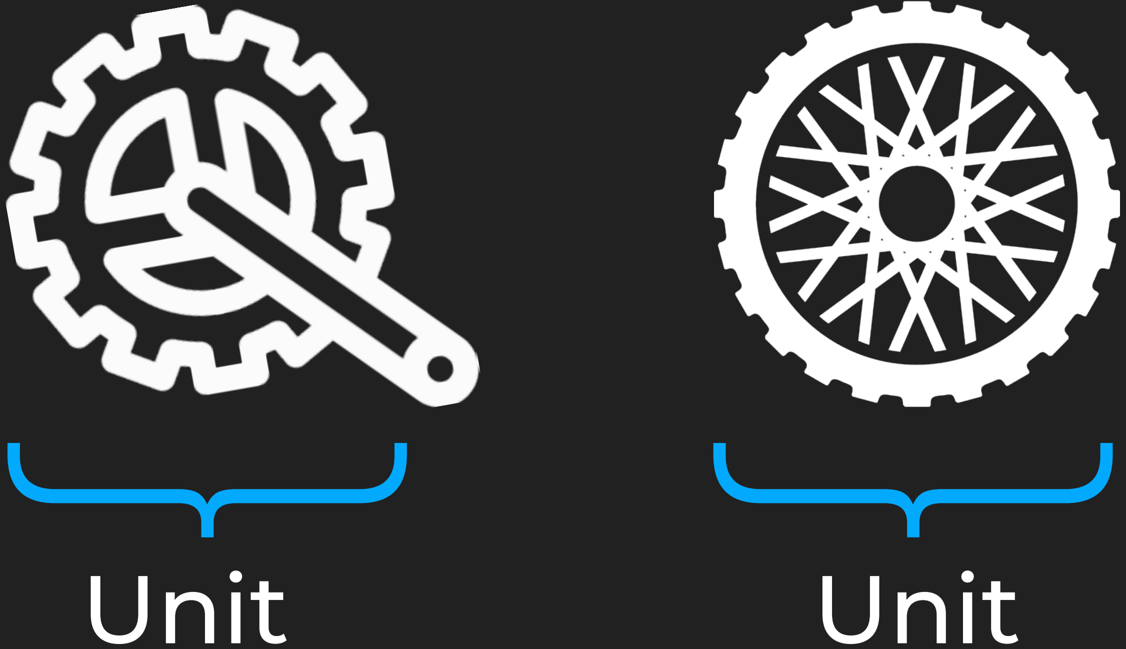
invokes a unit of work

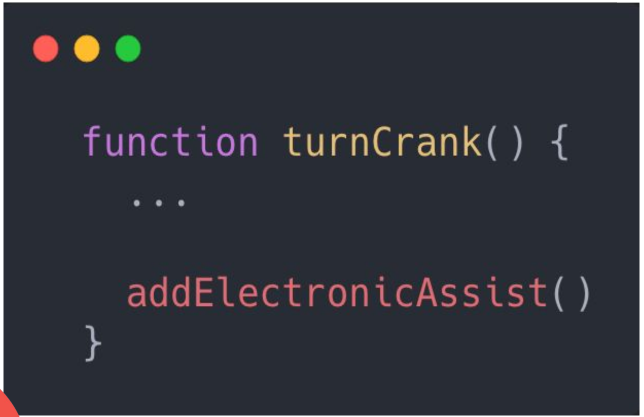
in the system and then

checks a single assumption

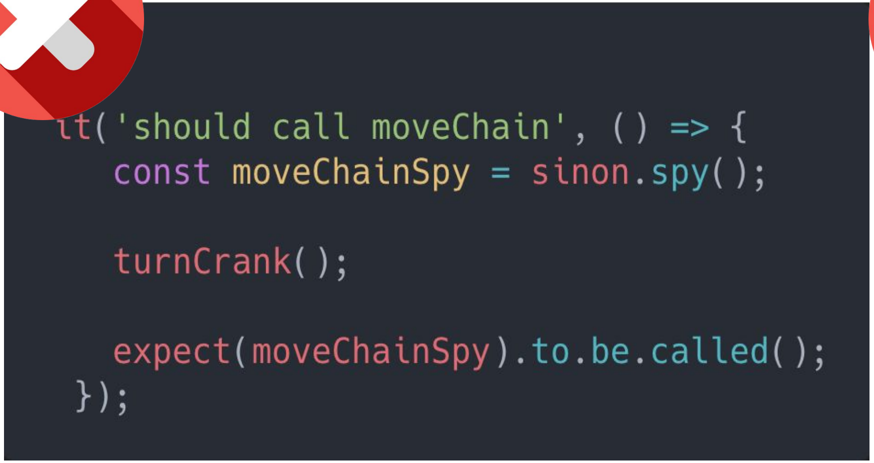
about the behavior of that unit of work.

What’s a **Unit**?

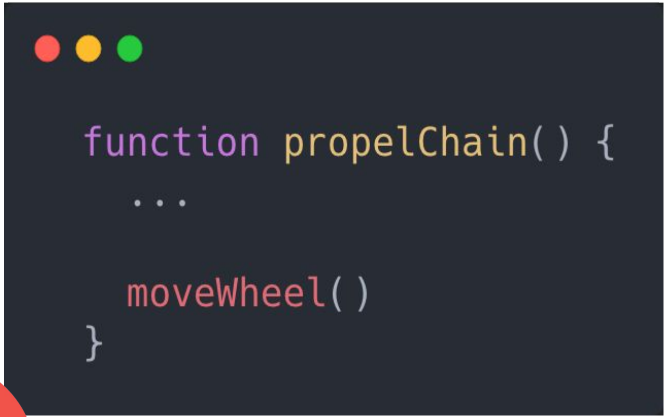

Unit
Unit



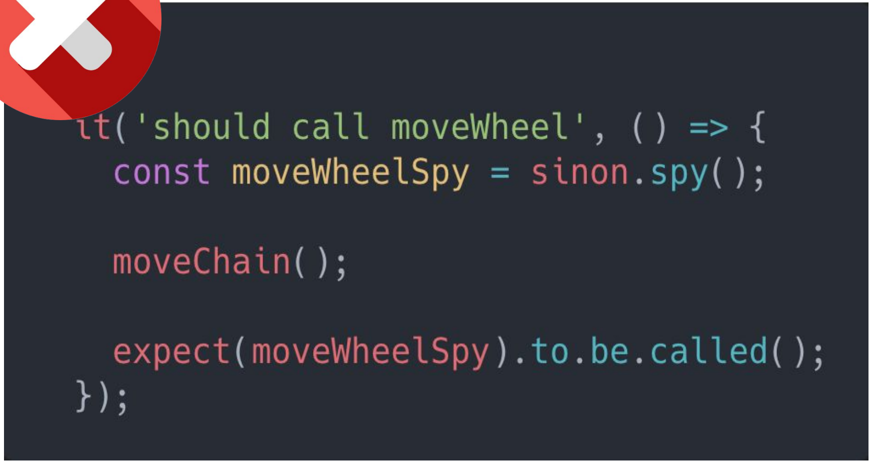
```
function turnCrank() {
  ...

  addElectronicAssist()
}
```



```
it('should call moveChain', () => {
  const moveChainSpy = sinon.spy();

  turnCrank();

  expect(moveChainSpy).to.be.called();
});
```



```
function propelChain() {
  ...

  moveWheel()
}
```



```
it('should call moveWheel', () => {
  const moveWheelSpy = sinon.spy();

  moveChain();

  expect(moveWheelSpy).to.be.called();
});
```

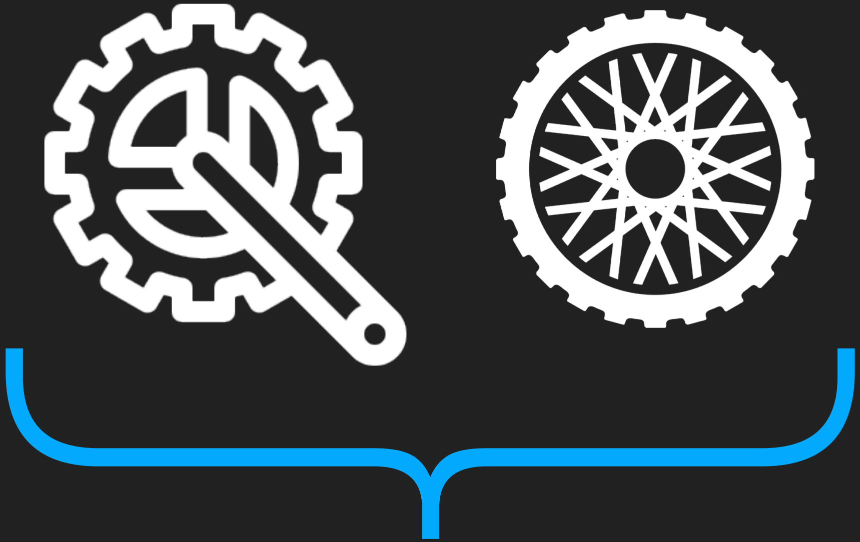
A unit of work can span

a single **method**, a whole **class** or **multiple classes**

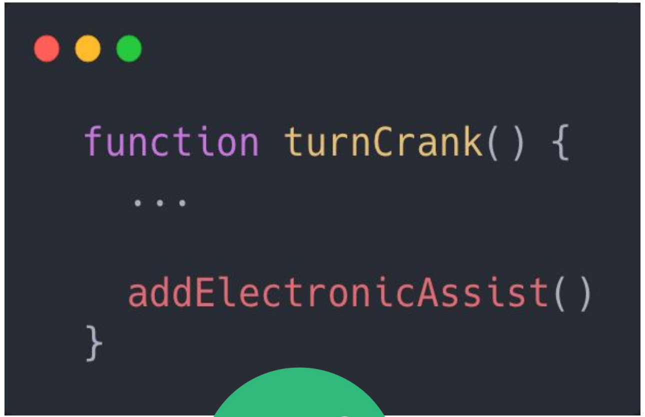
working together to achieve

one single logical purpose

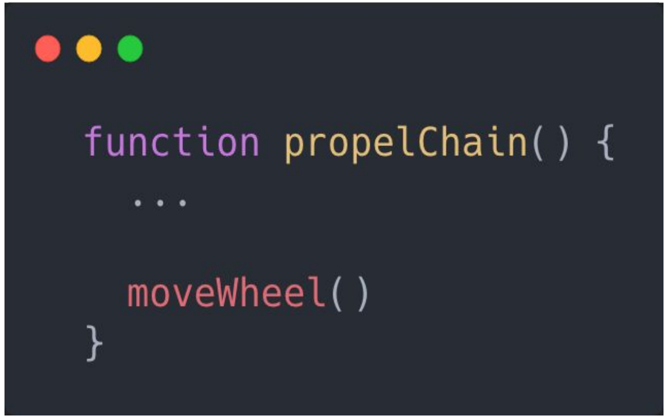
that can be verified.



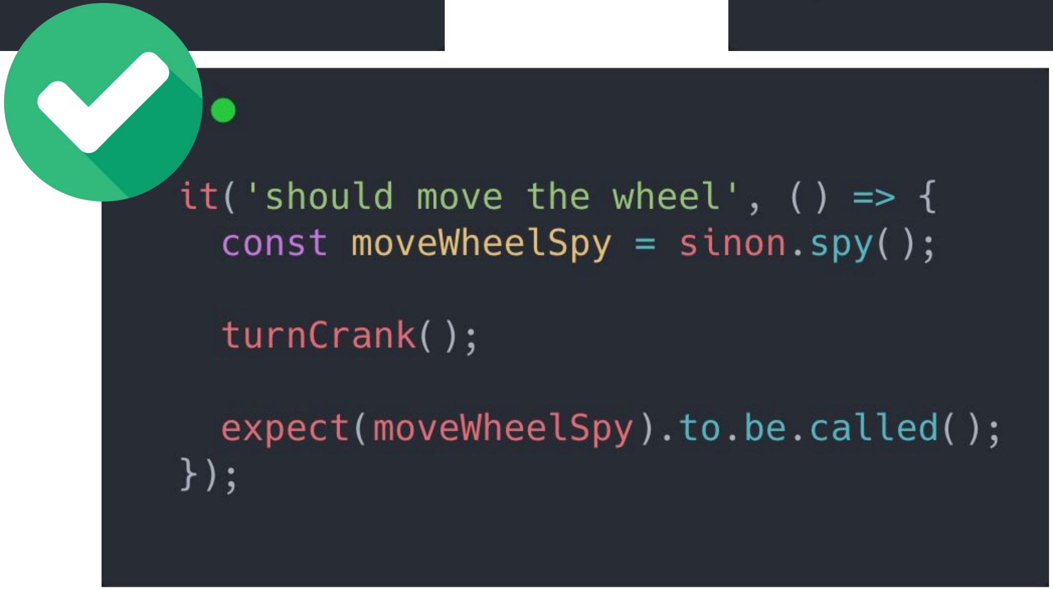
Unit



```
function turnCrank() {
  ...

  addElectronicAssist()
}
```



```
function propelChain() {
  ...

  moveWheel()
}
```



```
it('should move the wheel', () => {
  const moveWheelSpy = sinon.spy();

  turnCrank();

  expect(moveWheelSpy).to.be.called();
});
```

Lesson 2:

Write User-Centered Tests

Why do we write tests?

For our users!!

What’s the problem with non-user centered tests?

Tests are **passing**

Code coverage is **100%**

Nothing works...

Example

How do we improve the tests?

[redacted]

Test for each **user action**

[redacted]

Mock only **global state**

[redacted]

Validate output as **perceived by the user**

W [illegible] de

Write **short, easy-to-read** tests

Back to the Example

Use the right tool for the job: Testing Library

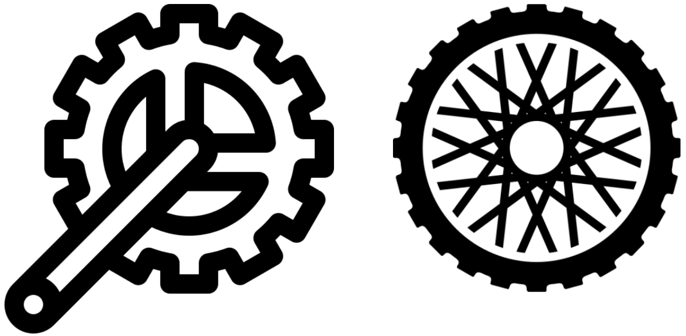
- We used “React Testing Library”
- Encourages good testing practices
- Available for most major frameworks
- Works with any test runner (Jest, Mocha, Karma, etc.)

https://testing-library.com/

Lesson 3:

Use every testing layer.

Before



Unit Tests

End-to-End Tests

Unit Tests

One unit of work

+

Virtual DOM

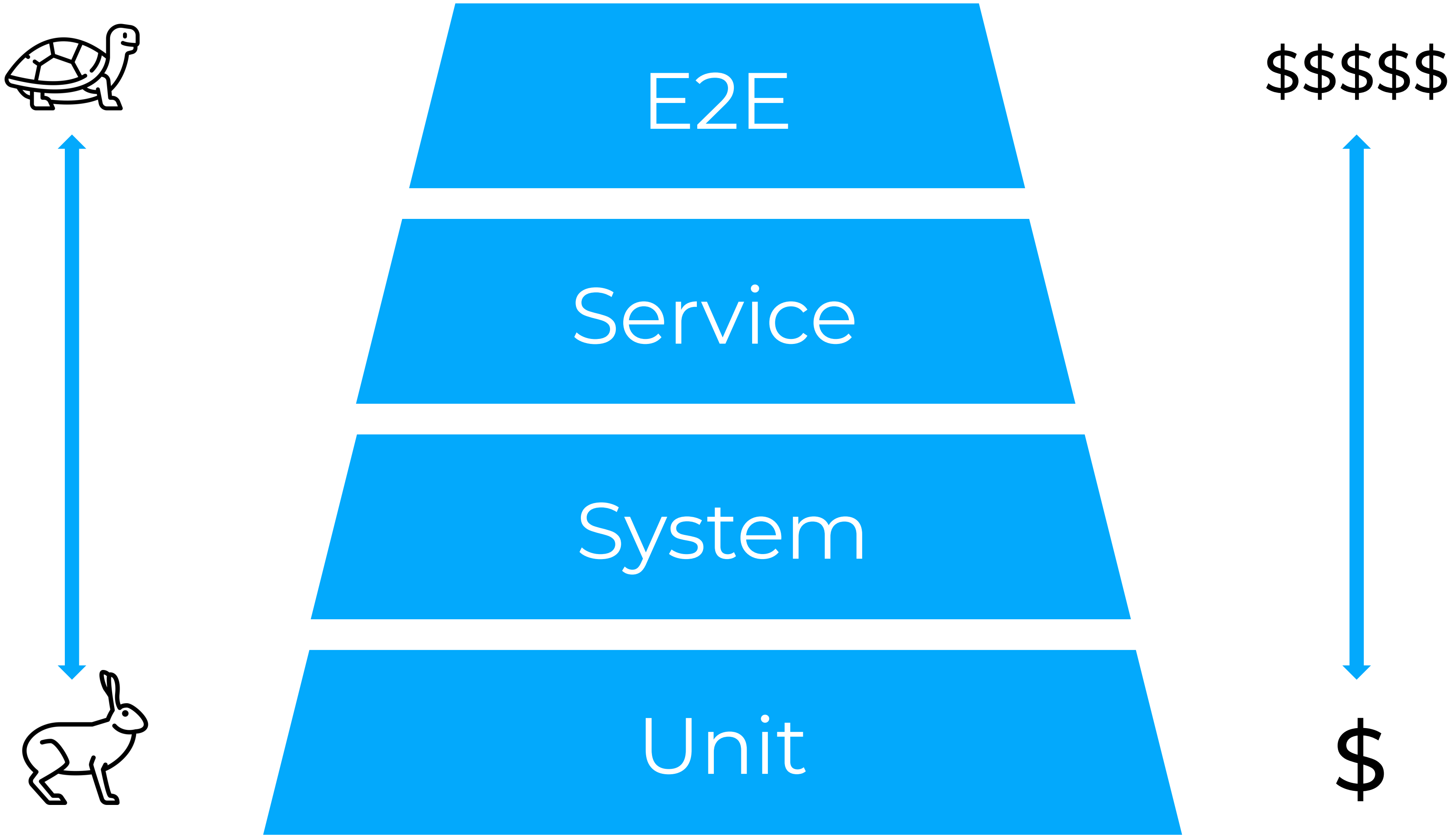
E2E Tests

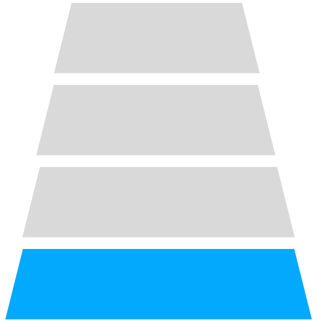
Entire application

+

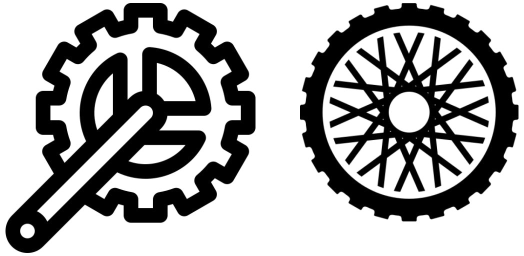
Real browser

Overuse of E2E tests is like making a deal with the devil.


E2E
Service
System
Unit
$$$$$
$



Unit



One Unit of Work

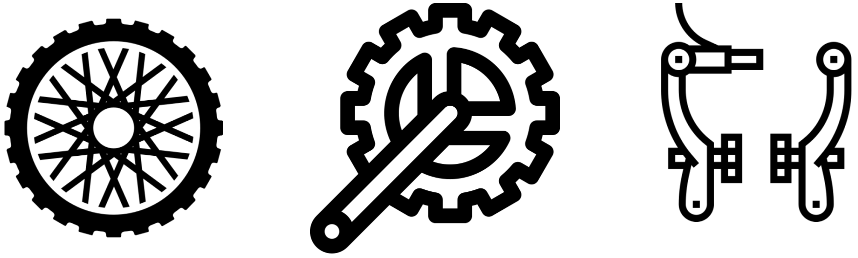
Fast & Cheap

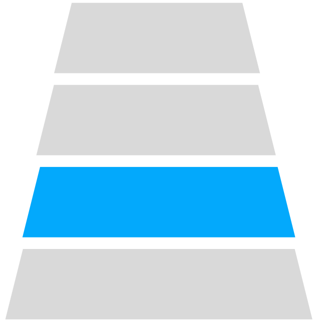
Virtual DOM (jsdom)

Jest, Mocha

React Testing Library, Enzyme, Chai, Sinon

System





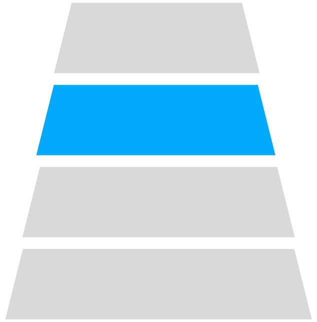
Real Web App, Stubbed APIs

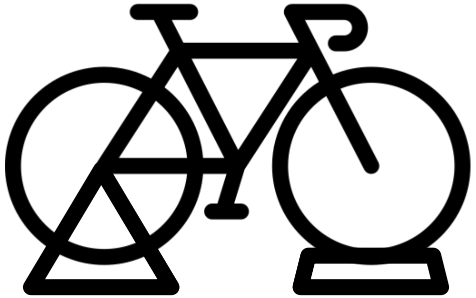
Moderately Fast, Moderately Cheap

Real DOM

Cypress, Puppeteer

Service

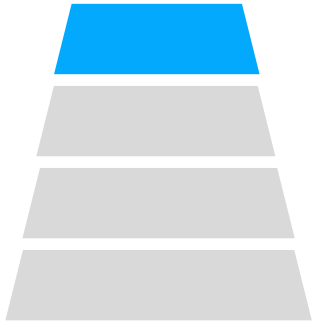




Real Web App, Real Service APIs
Stubbed Third-Party APIs
Slower, More Expensive

Cypress, Puppeteer, Selenium

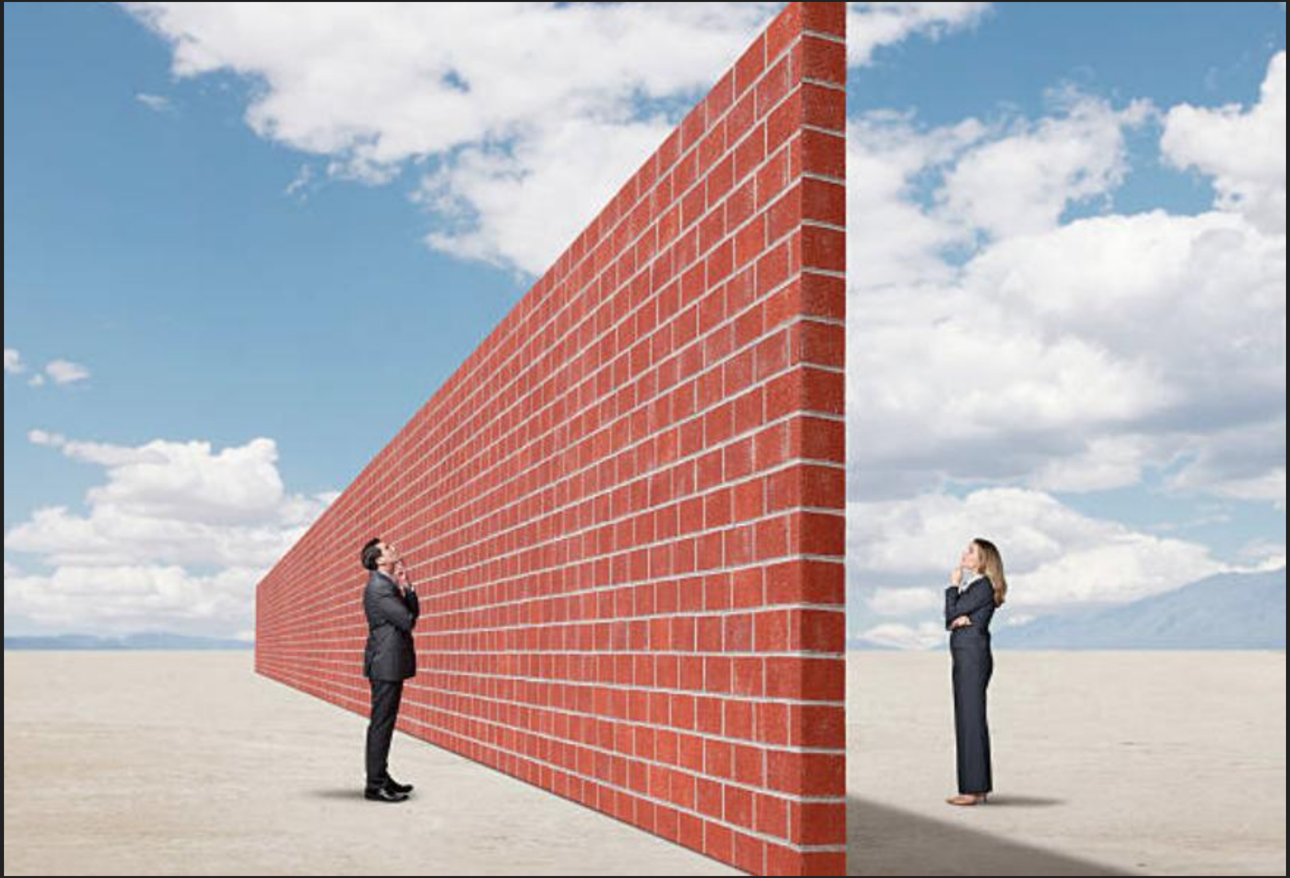
E2E



Real Web App, Real Service APIs
Real Third-Party APIs
Slow & Expensive

Cypress, Puppeteer, Selenium

Lesson 4:
Test Somewhere, Not Everywhere



Duplicating test effort increases **test bloat**, leading to a maintenance nightmare!

Two Changes to Tear Down the Wall

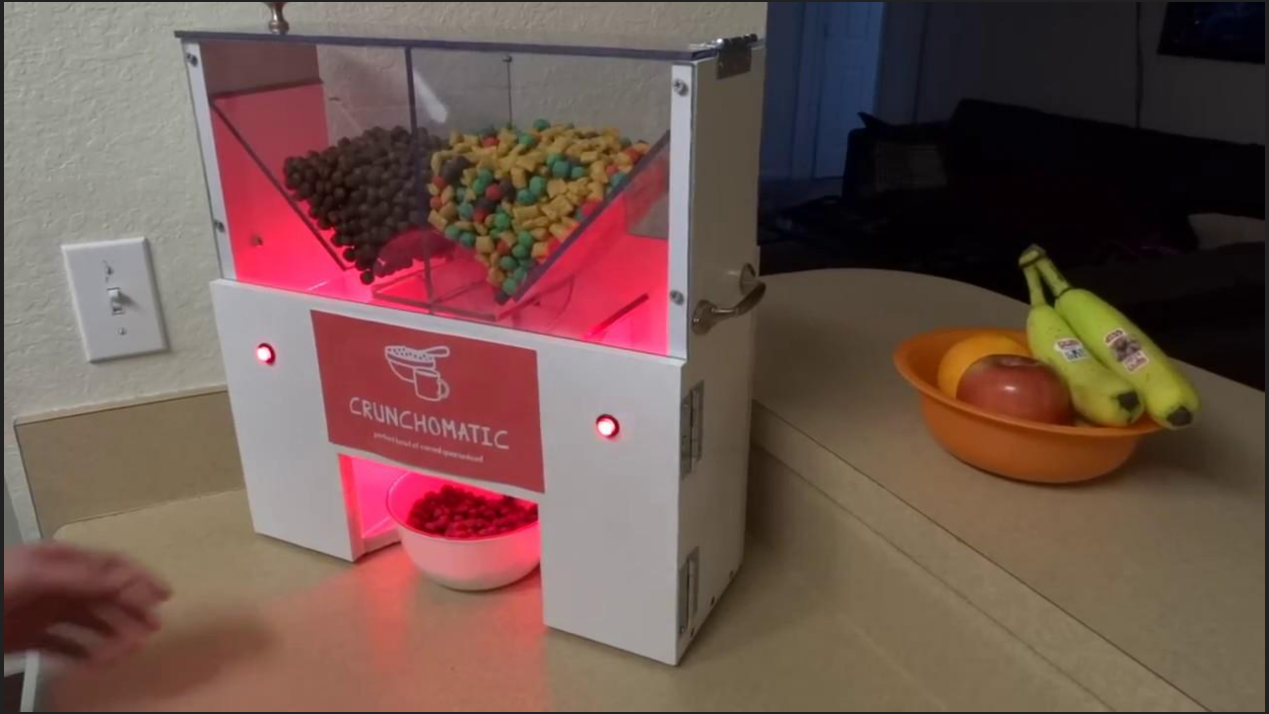
Everyone is responsible for quality.

EMFASIS

We actually talked to each other.

When writing new tests, consider your **comprehensive** test strategy

Cheerios


CRUNCHOMATIC
perfect bowl of cereal guaranteed

Future plans

- Automated testing of accessibility
- Balance between accurate user behavior and test performance
- Cleaning up old tests

Summary

1. Redefine a **unit of work**
2. Write **user-centered** tests
3. Utilize every layer of the **test pyramid**
4. Test **somewhere**, but not **everywhere**

Questions?

Caitlin Collins
caitlin.collins@code42.com

Mike Eastes
mike.eastes@code42.com