Test Driven Development For Frontends

Slides up at scottsauber.com

scottsauber

Audience

- Frontend Developers
- Familiar with testing
- Interested in learning TDD

Agenda

- What is TDD?
- Why TDD?
- Tools you can use
- What do I test?
- Live Demos

Goals

- Learn “best practices*” for writing frontend tests
- Learn how to TDD with React

* Synonym for “Just My Opinions” and I’ll probably find a way I like better in the future

Who am I?

- Director of Engineering at Lean TECHniques
- Co-organizer of Iowa .NET User Group
- Microsoft MVP
- Friend of Redgate
- Blog at scottsauber.com
- Used React, Blazor, or Angular last 7 years

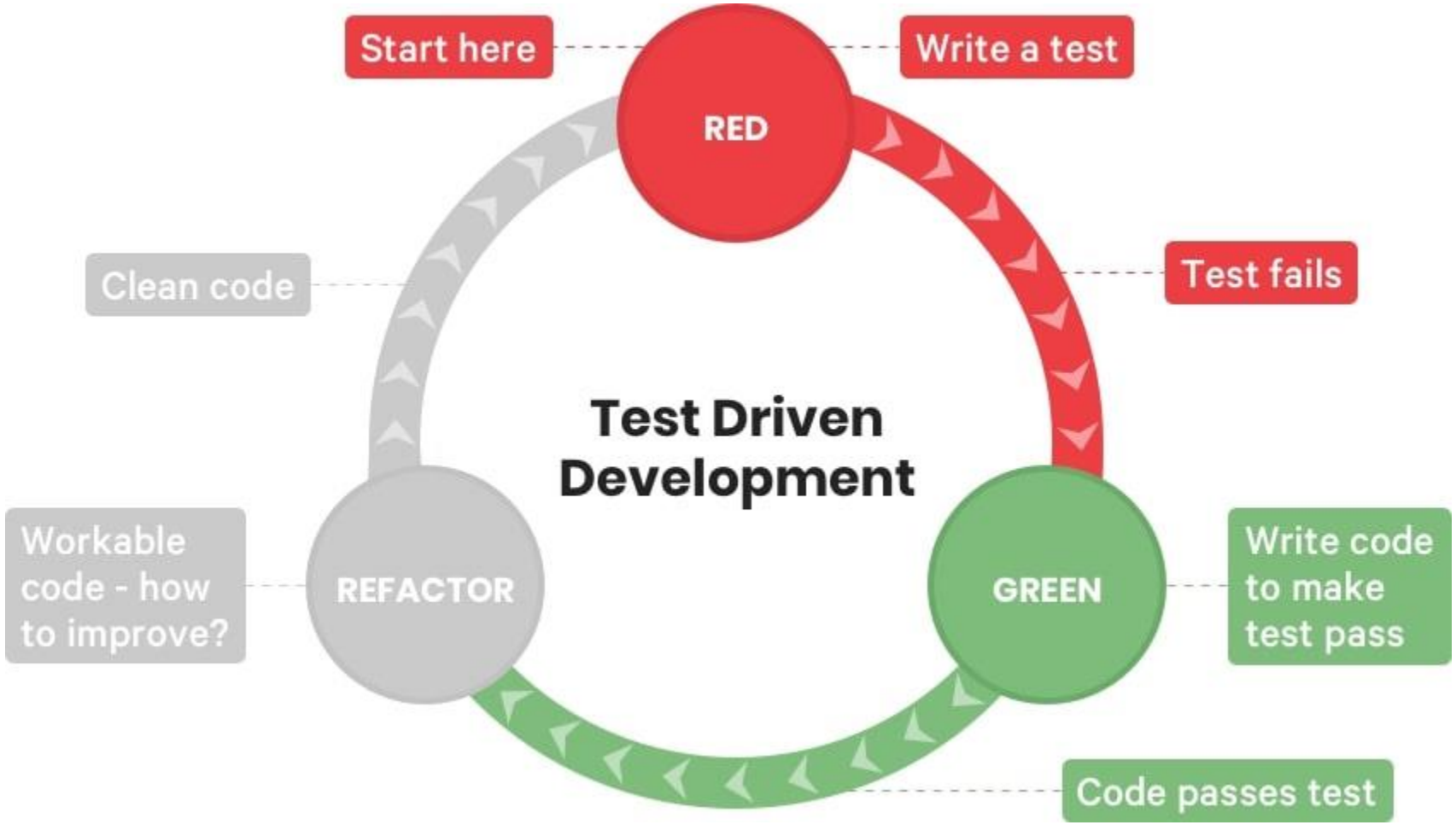
Why do we write tests?

- We want confidence our application works
- Minimize manual verification
- Document behavior through tests

How to TDD?

1. Think
2. Write a test that describes the behavior you want to see
3. Run the test and watch it fail *for the right reason*
4. Write code to make it pass
5. Refactor
6. Repeat

How to TDD?


Start here
Write a test
RED
Test fails
Clean code
Test Driven Development
Workable code - how to improve?
REFACTOR
GREEN
Write code to make test pass
Code passes test

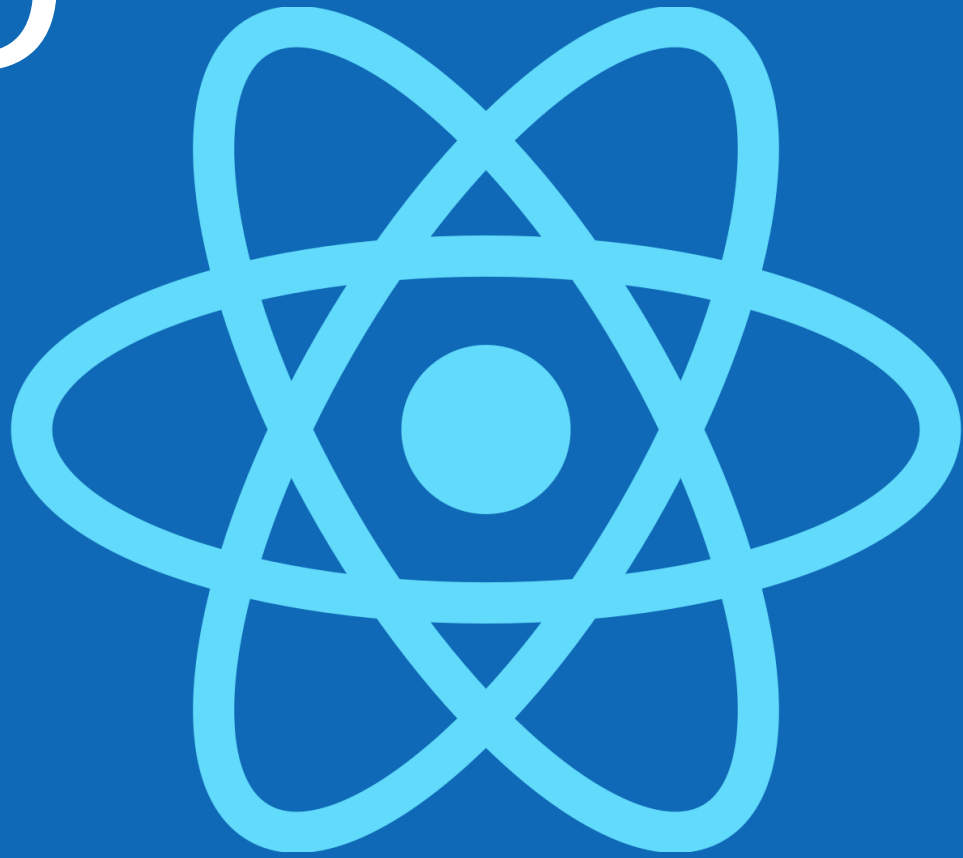
Why Test Driven Development?

- It’s a disciplined way of working
- A great way to focus
- A great way to get feedback on if your code and design sucks
- A great way to facilitate pair programming
- Often leads to very little time in the debugger
- Oh yeah… and the regression tests are nice too

What is *NOT* TDD?

- TDD is not a synonym for writing tests
- TDD is not writing ALL the tests up front
- TDD does not mean no bugs ever (just less)
- TDD zealots are harmful

Applying TDD to React



Introduction to Tools

- Jest
- @testing-library/react

Jest

- Test framework
- Zero config
- Assertions
- Mocking
- Watch

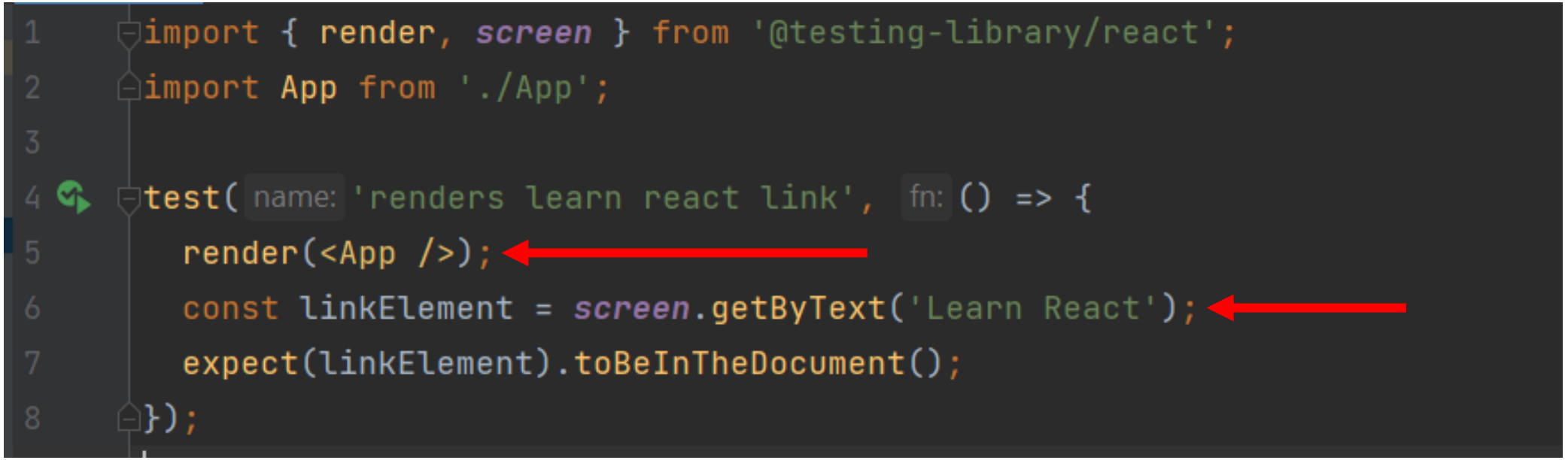
Jest


import { render, screen } from '@testing-library/react';
import App from './App';
test(name: 'renders learn react link', fn: () => {
render(<App />);
const linkElement = screen.getByText('Learn React');
expect(linkElement).toBeInTheDocument();
});

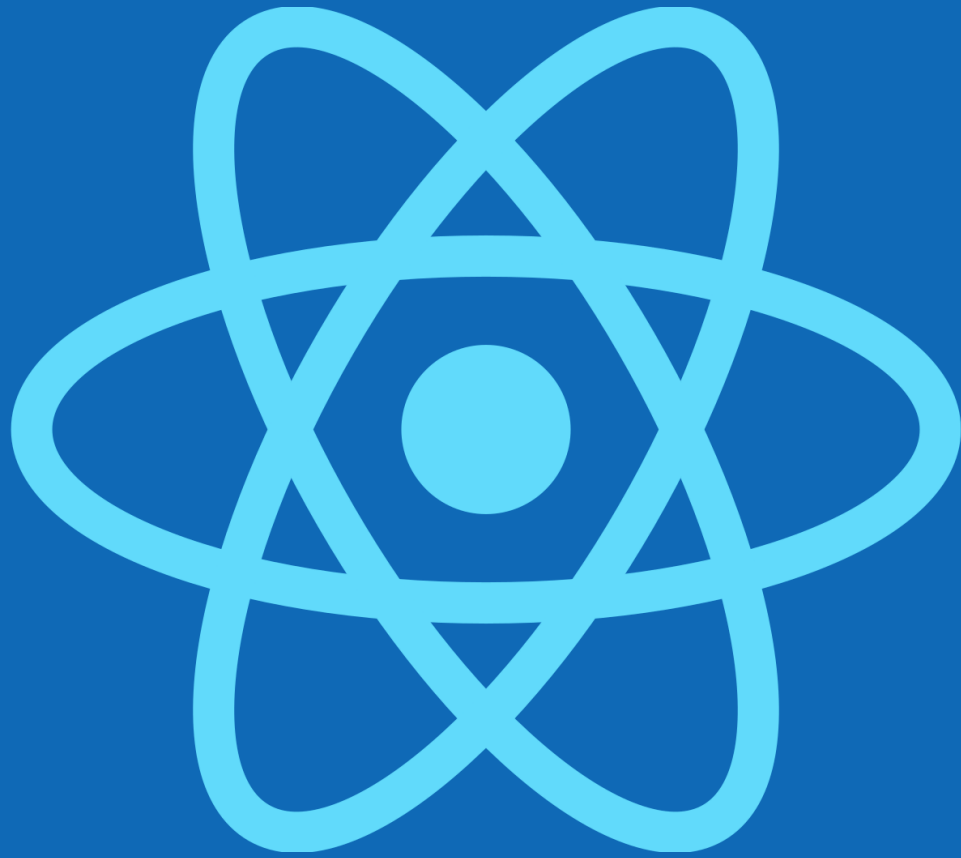
React Testing Library

- @testing-library/react is the package
- Utilities for testing React
- Encourages behavior-style tests
- Encourages avoiding testing implementation details
- DOM queries that promote accessibility
- Promotes deep rendering

React Testing Library



Demo



What should I test?

- Behavior
- Not that CSS classes exist or any other attributes directly exist
- Behavior
- If I can delete code that breaks your app, but your tests don’t – that’s a problem
- If my tests break but my application isn’t - that’s a problem
- Don’t use snapshots… (mostly)
- Snapshots don’t capture desired behavior
- Too many implementation details (i.e. classes, DOM nodes, etc.)
- Only use snapshots when doing a total refactor but output should be the same
- Then delete the test

“The more your tests resemble the way your software is used the more confidence they can give you.”

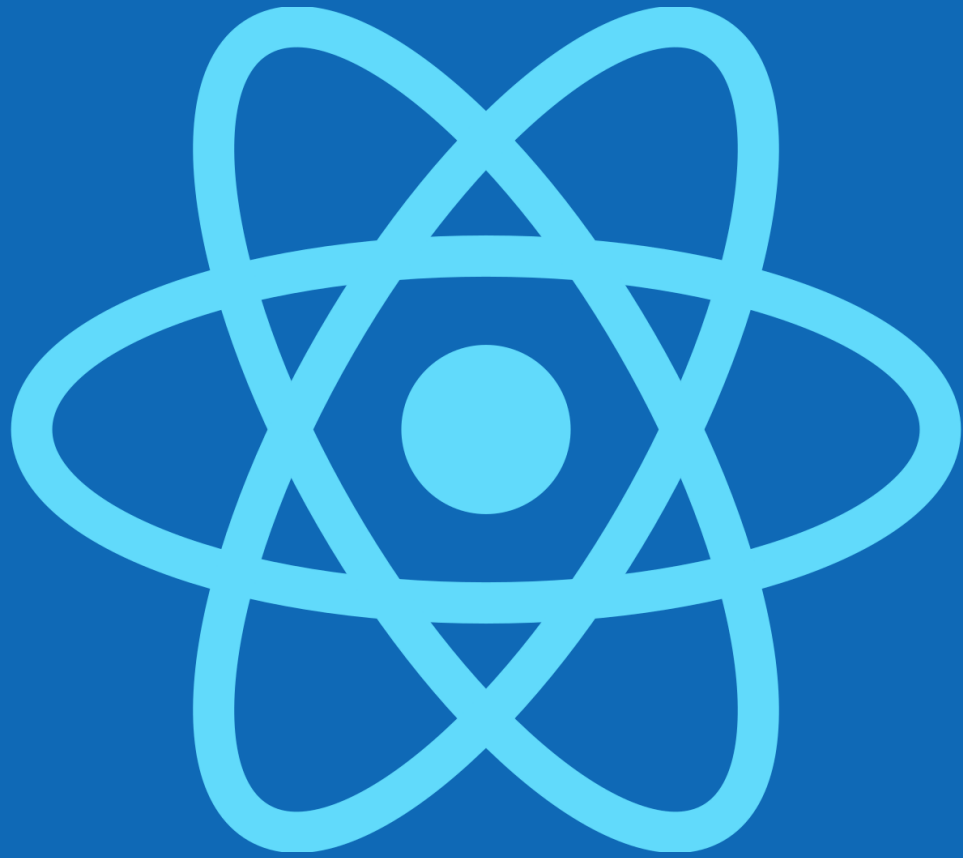
Kent C Dodds

react-testing-library creator

How do I structure tests?

- Avoid lots of describes
- Avoid lots of beforeEach nested in describes
- Avoid a top-level describe for the component you're testing
- You already know ^ by the file you're in
- Put tests next to the file they're testing
- High cohesion

Live Coding!

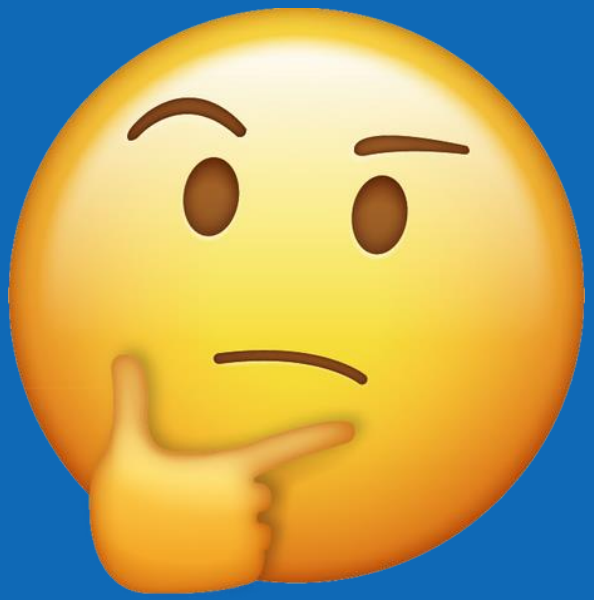


How can I get started with TDD?

- When you get a bug report coming in
- Write a failing test that proves the bug exists
- Make it pass

But I don’t have time!

Why?



My boss won’t let me!

What about this person?

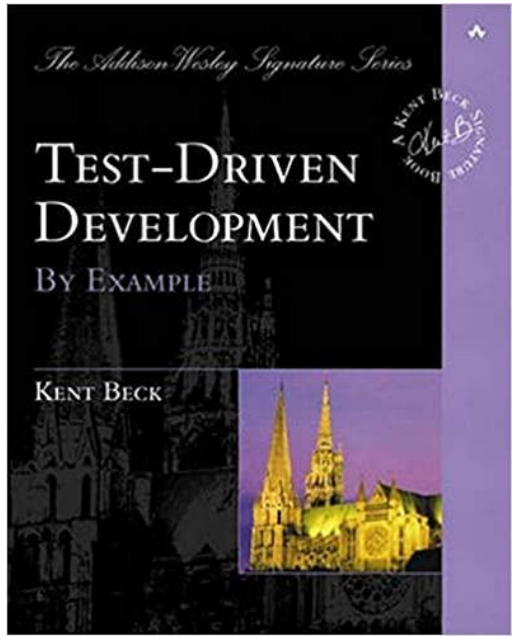
You don’t get better at TDD by NOT doing TDD

Takeaways

- Why you should TDD
- How to test React
- What to test in React
- How to get started TDDing React

Resources

- TDD By Example by Kent Beck
- Write Tests blog post by Kent C Dodds
- https://github.com/scottsauber/talks
- This slide deck



Questions?

Slides up at scottsauber.com

scottsauber

Thanks!

Slides up at scottsauber.com

scottsauber