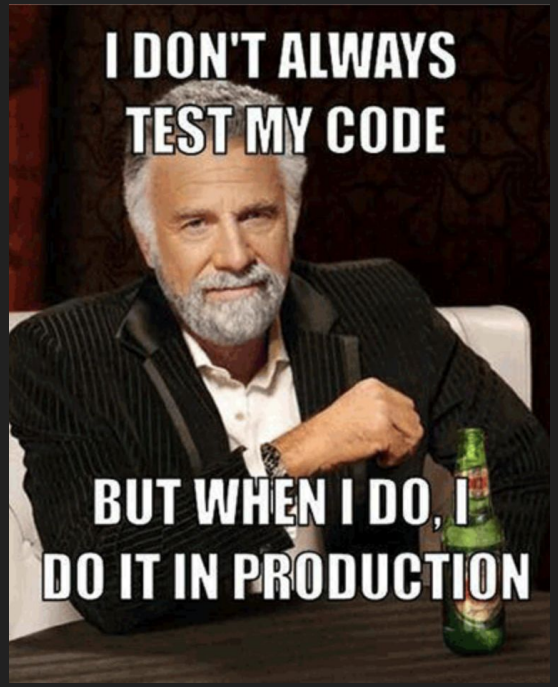
Testing in React

Goal - understand the basic concepts of software testing and show you the tools needed to write good tests in React

What we will cover

- General overview
- Different types of testing
- Writing actual tests
- Testing Environment
- Coding Demo



General overview

Different types of testing

Writing actual tests

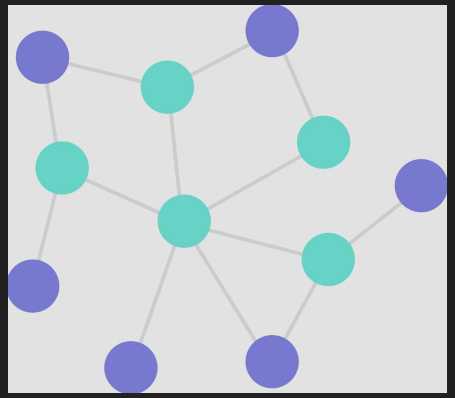
Testing Environment

Coding Demo

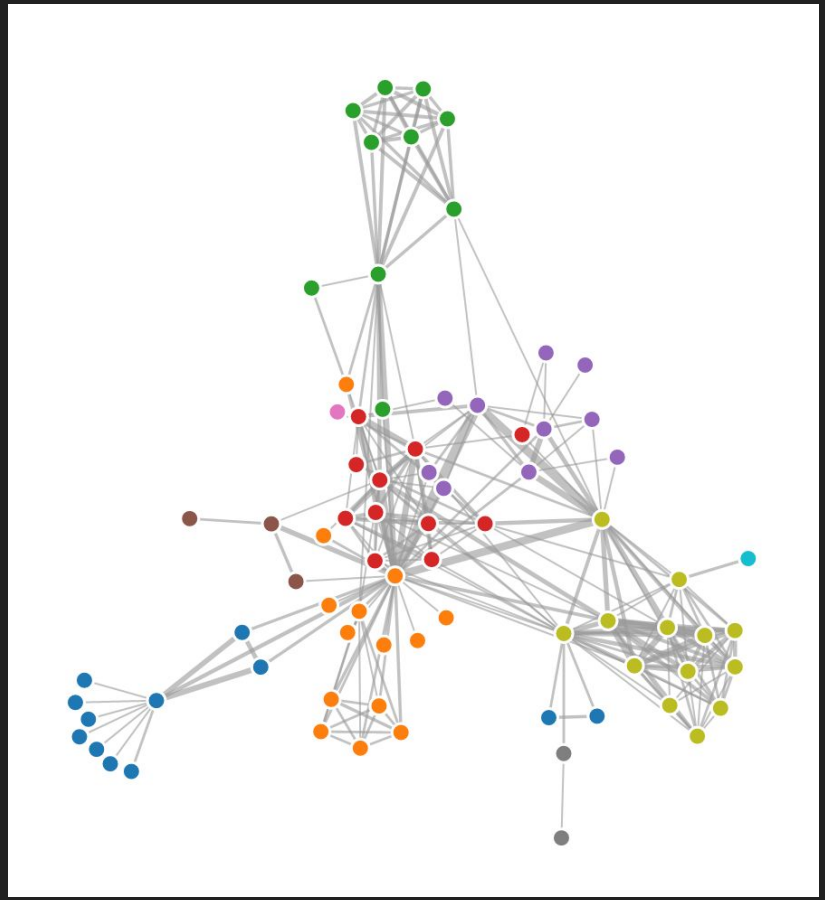
What is software testing?

- Process of evaluating and verifying that software is doing what we expect
- Can be performed by a human (e.g. QA Engineer) or machine (e.g. coded by developer/SDET)
- Many different kinds of testing
- Many different tools available to us

Our Code



Time



Writing tests - tradeoffs

Pros

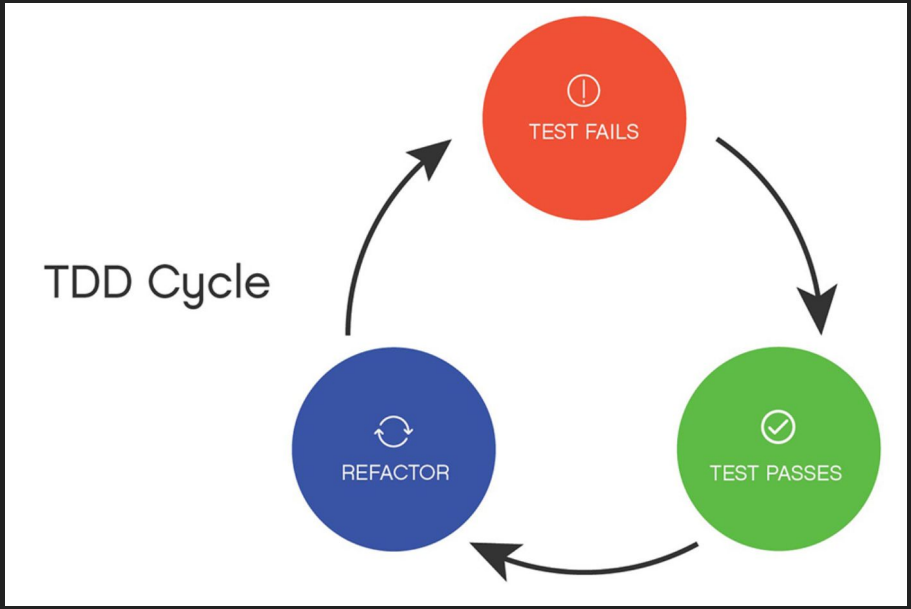
- Doing major refactors is easier
- Catches bugs earlier
- More confident in code quality

Cons

- Additional things to maintain
- Slows down development
- CI pipeline takes longer
- As application gets more complex writing tests can become more difficult
- Expensive

Test-Driven Development

- Software development process where test cases are written before the code
- Works when business logic and rules are well defined



Brenan Keller
@brenankeller

A QA engineer walks into a bar. Orders a beer. Orders 0 beers. Orders 99999999999 beers. Orders a lizard. Orders -1 beers. Orders a ueicbksjdhd.

First real customer walks in and asks where the bathroom is. The bar bursts into flames, killing everyone.

General overview

Different types of testing

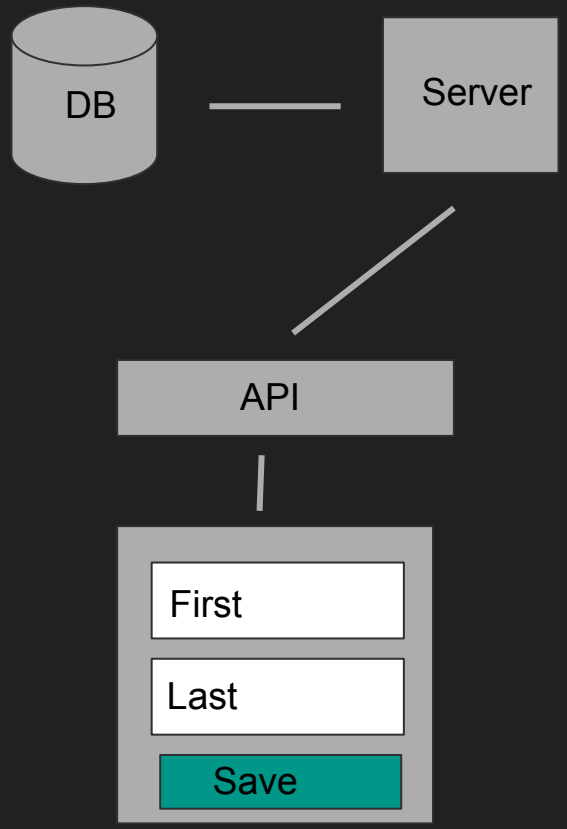
Writing actual tests

Testing Environment

Coding Demo

Why different types/levels?

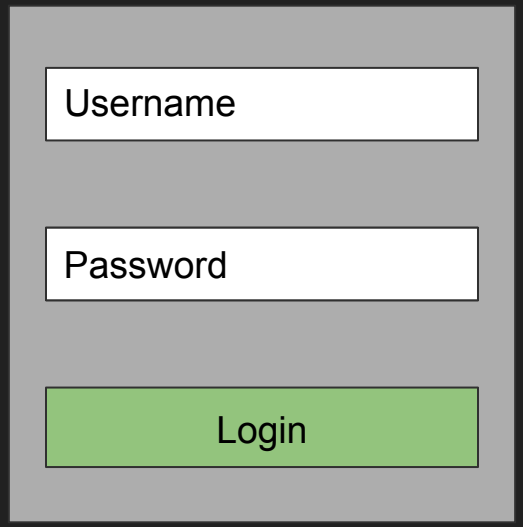
- It makes it easier to pinpoint the point of failure
- Easier to test more things
- Example - writing one test for a user profile



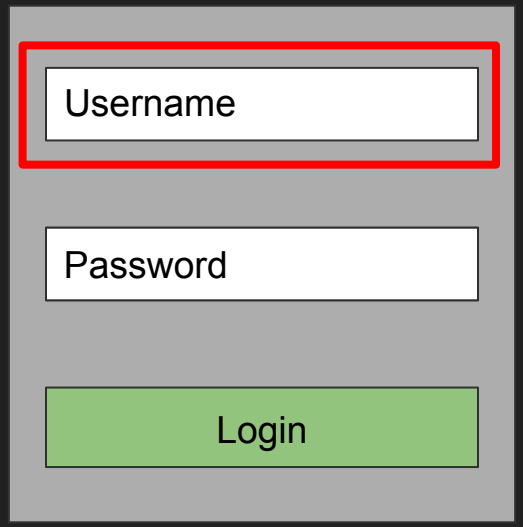
Main types of tests we care about

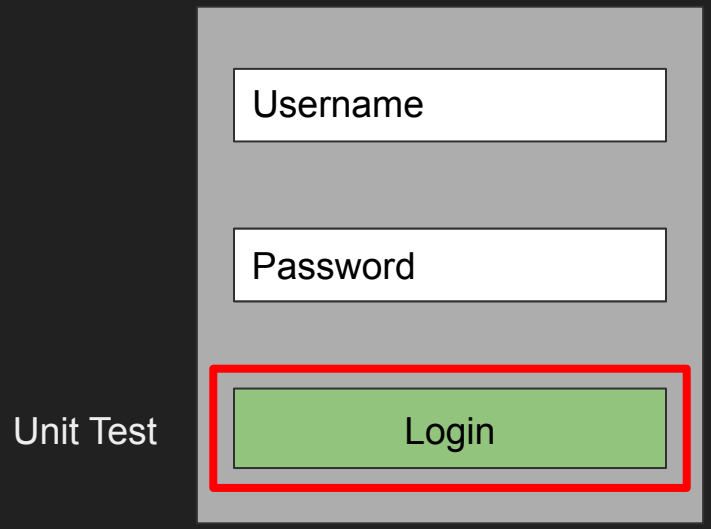
- Unit - validating that each software “unit” performs as expected where a unit is the smallest testable component
  - individual functions and classes
- Integration - ensures components operate as a group
  - Involving network activity
- Acceptance - ensures whole system works as intended

Example

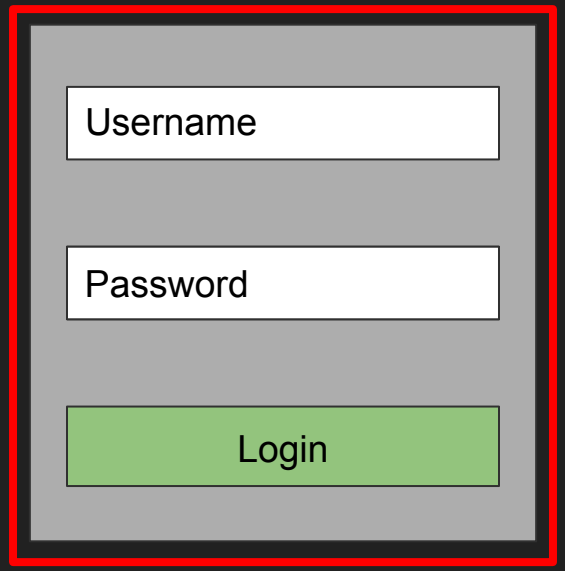


Unit Test

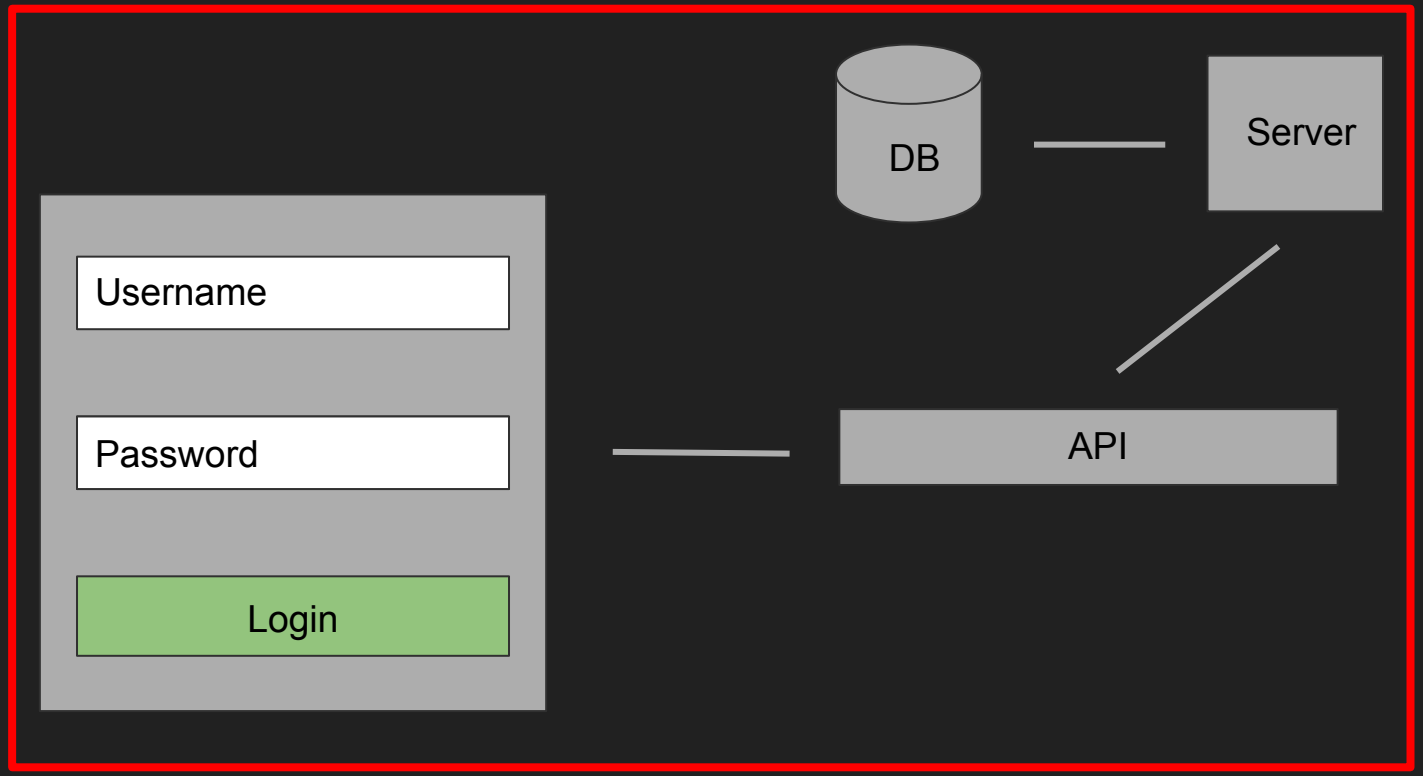



Username
Password
Unit Test
Login

Integration test

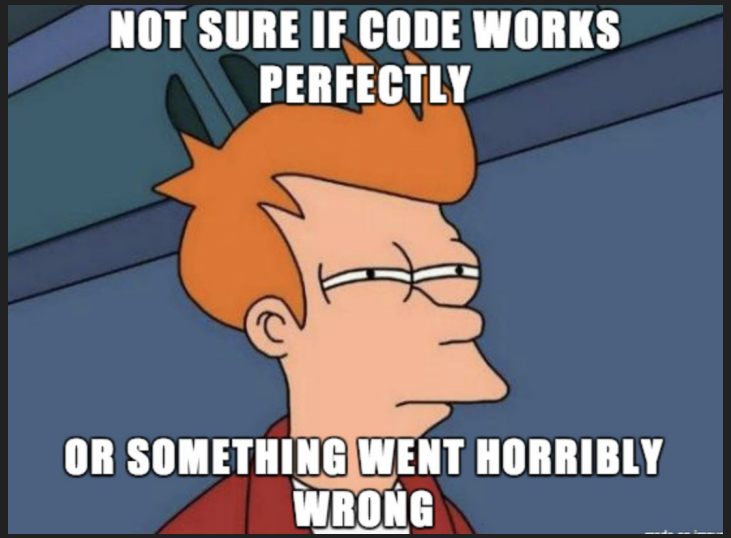


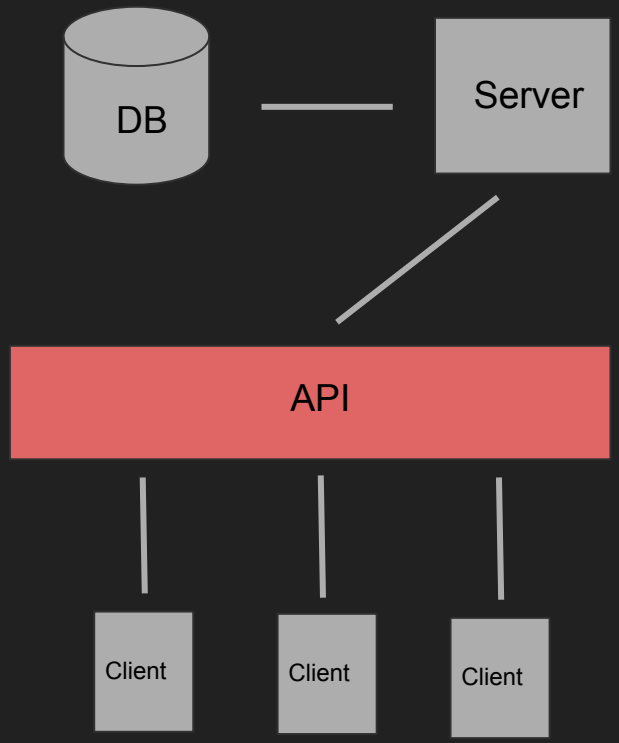
Acceptance test



Other types of testing

- Performance testing - how software performs under different workloads
  - In FE code case, how it works in other browsers
- Regression testing - check whether new features break or degrade functionality
- Stress testing - testing how much strain the system can take before it fails
- Penetration testing - is my system secure




DB
Server
API
Client
Client
Client

General overview

Different types of testing

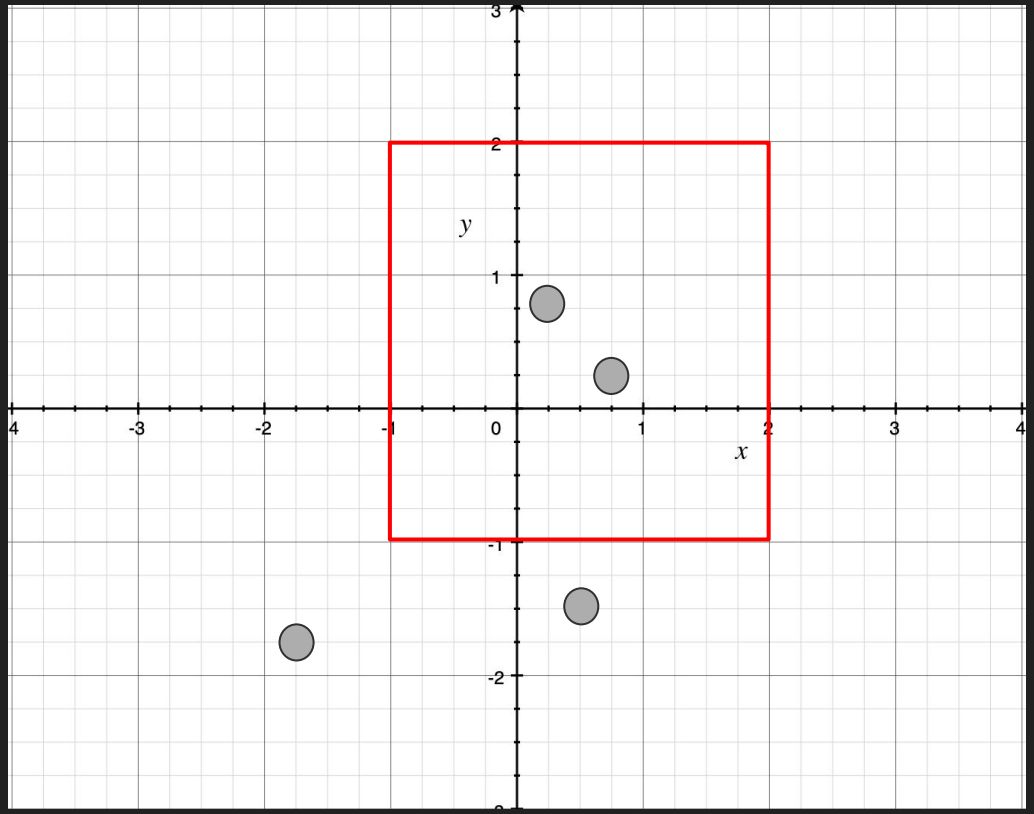
Writing actual tests

Testing Environment

Coding Demo

Structure - “Arrange-Act-Assert ”

- **Arrange -** all necessary preconditions and inputs
- Act - on the object or method under test
- Assert - that the expected results have occurred


3
2
y
1
-4
-3
-2
-1
0
1
2
3
4
x
-1
-2

Structure - “Arrange-Act-Assert ”

- **Arrange -** all necessary preconditions and inputs
- Act - on the object or method under test
- Assert - that the expected results have occurred

```
import { findPointsInBox } from "src/calc/findPointsInBox.js";

test(() => {
  const points = [
    { x: 3, y: 1 },
    { x: 1, y: 1 },
    { x: 1, y: 3 }
  ];
  const box = {
    min: { x: 0, y: 0 },
    max: { x: 2, y: 2 }
  };

  const pointsInBox = findPointsInBox(box, points);

  asset(pointsInBox.length, 1);
  asset(pointsInBox[0], { x: 1, y: 1 });
});
```

Testing Recipes - common patterns for React components

- Rendering - how the component looks
- Data fetching - mock API requests with dummy data
- Mock modules - mock entire node modules (e.g. Google Maps)
- Events
- Timers

Snapshots

- Testing that compares the stored “good output” with the current output of a component
- “Good output” can be text (e.g. rendered DOM tree) or even screenshots
- Only concerned with detecting unexpected UI changes

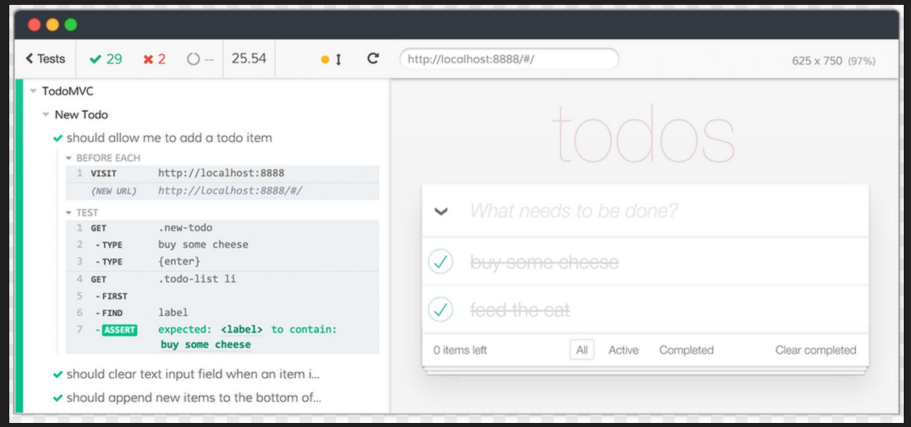
Testing Tools - Test runners

- Test runners - run tests in a reliable and predictable way
- Libraries are written in the same language as the projects codebase
- Generally written by developer or a SDET
- Examples of test runners for React
  - Jest (included in create-react-app)
  - Mocha

```
(base) ➜  typescript-jest-example git:(master) ✗ npm run tests

> typescript-jest-example@1.0.0 tests /Users/dmar/Repos/typescript-jest-example
> jest --config jest-config.json

 PASS  tests/utils.spec.ts
  Utils test case
    ✓ Add 1 + 2, should return 3 (6 ms)
    ✓ Add -1 and 2 should return an Error (5 ms)

----------|---------|----------|---------|---------|-------------------
File      | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s
----------|---------|----------|---------|---------|-------------------
All files |     100 |      100 |     100 |     100 |
 utils.ts |     100 |      100 |     100 |     100 |
----------|---------|----------|---------|---------|-------------------
Test Suites: 1 passed, 1 total
Tests:       2 passed, 2 total
Snapshots:   0 total
Time:        1.597 s, estimated 2 s
Ran all test suites.
```

Testing Tools - Integration/Acceptance

- Setting up a realistic browser environment (Firefox, Safari, etc …) where you simulate an actual user
- Testing on a real API or Mock API
- Headless mode
- Testing frameworks like **Cypress**, Puppeteer, and CodeceptJS



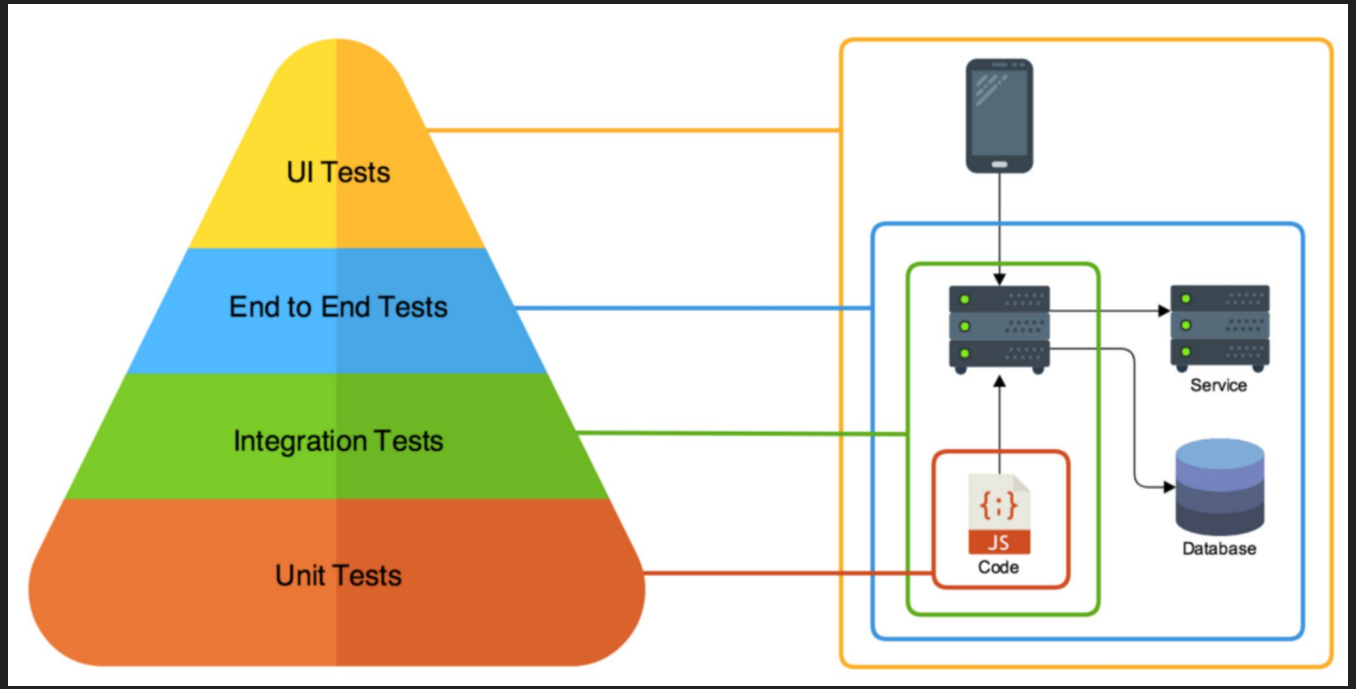
General overview

Different types of testing

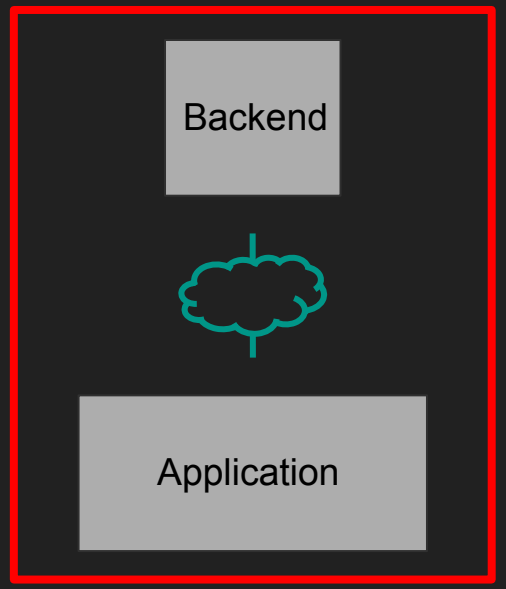
Writing actual tests

Testing Environment

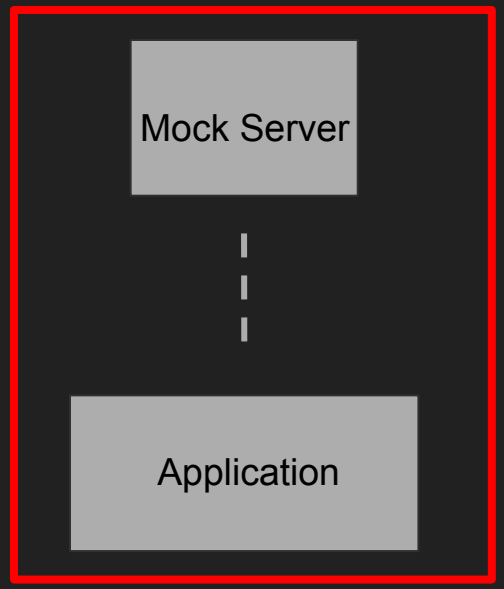
Coding Demo


UI Tests
End to End Tests
Integration Tests
Unit Tests
Service
Database
JS
Code

Real API



Mock API



Mock API

Pros

- Minimal setup required
- Lots of existing FE tools for generating fake data
- Testing environment becomes FE only
- Speed of running tests is much faster
- FE isn’t blocked by development work on BE

Cons

- API can become out of sync with mock server (requires maintenance)
- Not always obvious when mock server and actual API are out of date
- Client will only be as good as your mock server

Real API

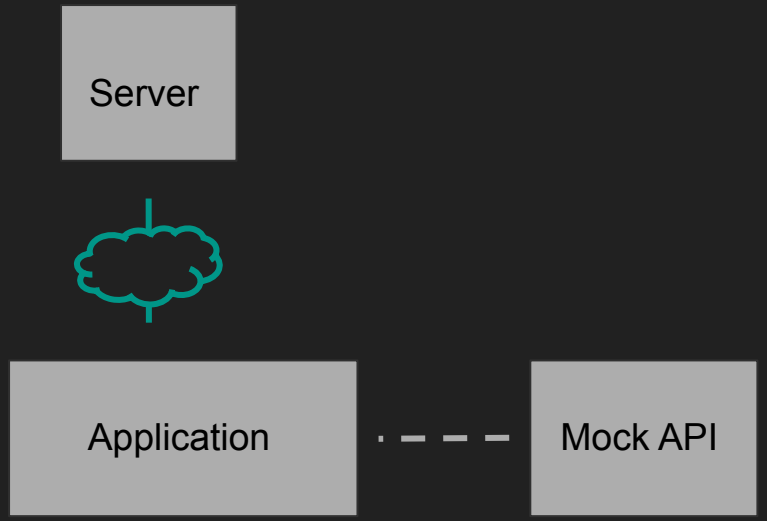
Pros

- Tests are more accurate
- Only one API to maintain

Cons

- Slower
- Testing environment becomes more difficult to set up
- Client development is now blocked by backend development
- Tests are less stable

Hybrid Approach



In practice testing on FE is added once codebase matures

- Writing and maintaining tests take a lot of time and energy
- We can deliver features quicker without them
- On day zero when a feature is written, they serve no value to our end user
- Testing FE code becomes complicated due to rendering UI
- Setting up CI environment is tricky

Sidenote on Continuous Integration (CI)

- Continuous integration (CI): software development practice where developers regularly merge their code changes into a central repository, after which automated builds and tests are run.
- Github actions, Teamcity, etc

Some checks were not successful

10 successful and 2 failing checks

Hide all checks

✓	**Athenian Pull Request Labeler / labeler (pull_request_target)** Successful in 4s		Details
✓	**Checks / ESLint (pull_request)** Successful in 3m		Details
✓	**PR Jira Title Checker / Jira Issue Checker (pull_request)** Successful in 4s		Details
✓	**Checks / Type Check (pull_request)** Successful in 3m		Details
✓	**Frontend Dev deployment / Get Frontend Stack Name / Get Frontend Stack Name (pull_reques...**		Details
✓	**Checks / Circular Dependencies, and Unused Code Check (pull_request)** Successful in 2m		Details
✗	**Checks / Jest Unit Tests (pull_request)** Failing after 2m		Details
✓	**Frontend Dev deployment / Build And Push Frontend Docker Image To ECR / Build And Push Fr...**		Details
✗	**Frontend Dev deployment / Cypress Tests / Cypress Tests (pull_request)** Failing after 47m		Details
✓	**Frontend Dev deployment / Deploy Frontend Ephemeral Environment To Kubernetes / Deploy F...**		Details
✓	**Backend Build (Testing / E2E)** — TeamCity build finished	Required	Details
✓	**E2E (Testing / E2E)** — TeamCity build finished	Required	Details

This branch has no conflicts with the base branch

Merging can be performed automatically.

Questions???

General overview

Different types of testing

Writing actual tests

Testing Environment

Coding Demo

Show me the code

- Jest (integration/unit)
  - Neat features: async test suites
- Cypress (end-to-end/UI)
  - Neat features: retry logic
- Helpful links
  - https://reactjs.org/docs/testing-recipes.html
  - https://docs.cypress.io/api/table-of-contents
  - https://jestjs.io/docs/api