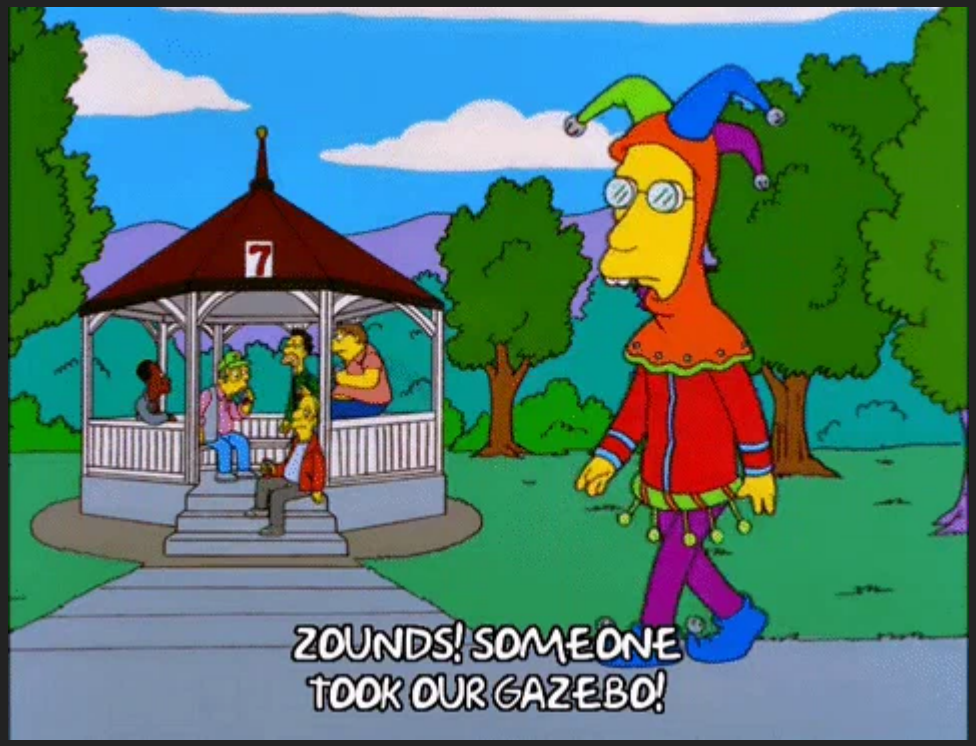
INTRODUCTION TO JEST

A front-end testing library built by Facebook

ABOUT ME

Jennifer Kaplan
Full Stack Developer Intern at Crane.ai
General Assembly alumna


7
ZOUNDS! SOMEONE
TOOK OUR GAZEBO!

AGENDA

- Why you should use Jest to test React
- Overview of snapshot testing
- Frequently encountered bugs
  - Or rather, the things that I encounterd that prevented my tests from passing

WHY YOU SHOULD USE JEST TO TEST REACT?

- Jest is built by Facebook who also builds React
- Top Google results for Mocha testing React are from 2015 and early 2016
- Jest was neglected for a long time, but for the past year they completely rewrote it and now it is great
- Front-end only testing is easy, testing components with API calls from the backend are more complex

WHAT IS SNAPSHOT TESTING?

- Allows you to quickly write tests for static data
- The Jest team wants this to be a quick way to write tests so you will write more tests
- If the data changes, you don't need to rewrite your test, Jest has a simple command to update the test

HOW SNAPSHOT TESTING WORKS

- You write a simple test with snapshot, then it saves that into a folder of snapshots.
- Then when you test it, it compares the page to the snapshot.
- If they match, the test passes. If it doesn't match, it shows you the changes and you can run the test again with `npm test -u` to update the snapshot.
- Therefore if the data changes, you don't need to write a new test.

  For example, a few hours ago my coworker asked me to update his tests because he changed the database from Algolia to Mongo

SAMPLE SIMPLE SNAPSHOT TEST

This is the base test we will use tonight and we will be modifying to get it to pass. This does work if the API call is on the React side, but if it is on Node it fails.

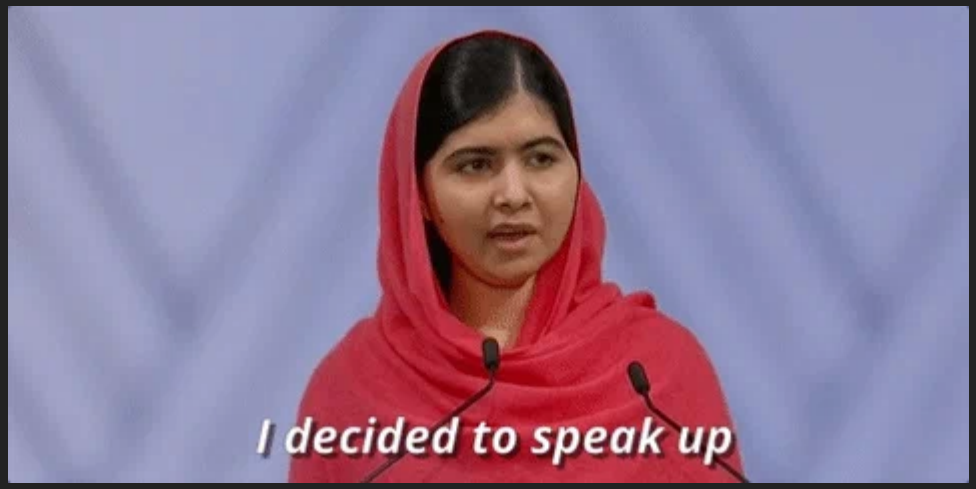
```
it('loads in pokemon list', () => {
  const tree = renderer.create(<App />).toJSON();
  expect(tree).toMatchSnapshot();
});
```

THIS IS THE DATA REACT IS GETTING FROM THE API

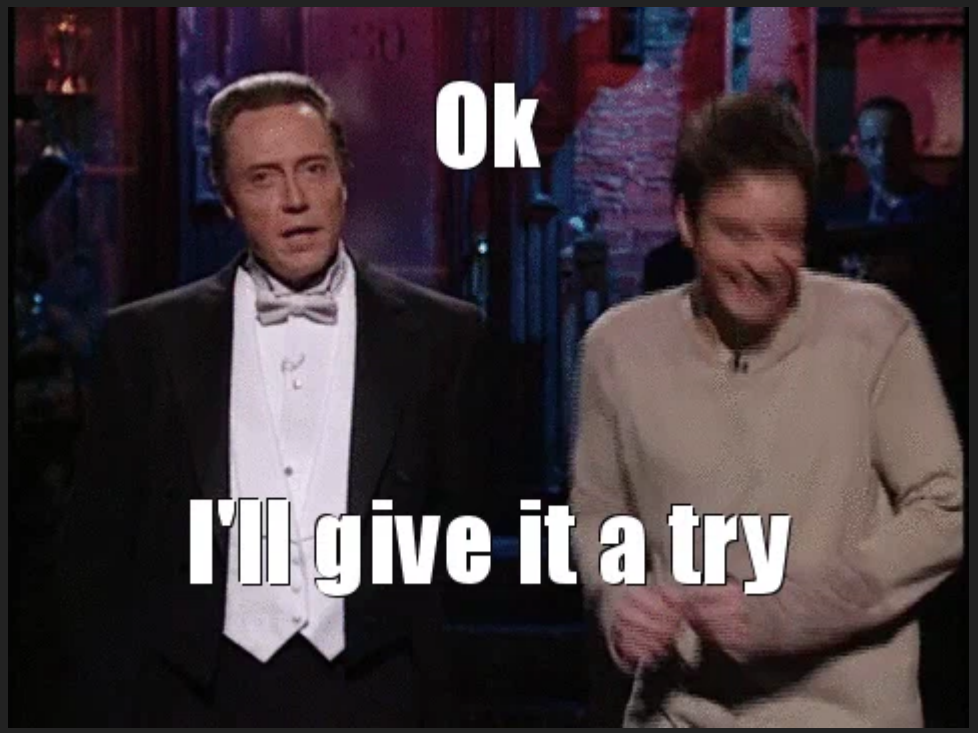
```
const data = {
  names: [
    {
        "url": "http://pokeapi.co/api/v2/pokemon/1/",
        "name": "bulbasaur"
    },
    // shortened for length
  ]
}
```

DOES EVERYONE UNDERSTAND THE BASICS OF SNAPSHOT TESTING?

If not, please speak up. What is coming will get confusing if you are already hazy on the idea.


I decided to speak up

SO NOW WE RUN THAT SIMPLE TEST...



FIRST ERROR!

```
FAIL  tests/app.test.js
● loads in pokemon list

  TypeError: Cannot read property 'names' of undefined
```

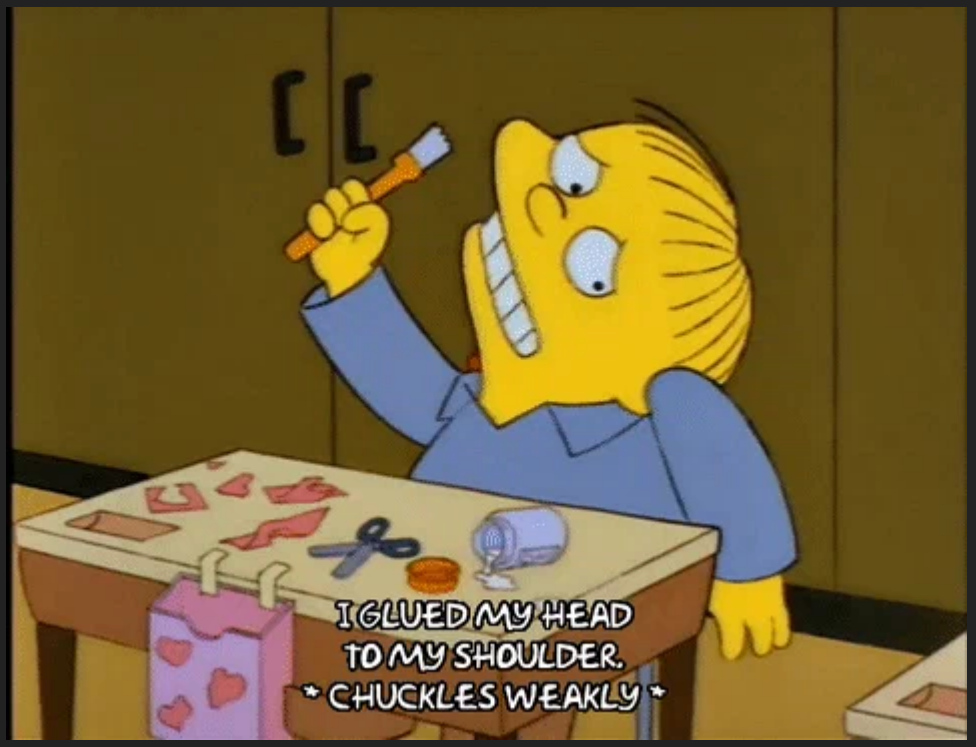
THAT WAS CAUSED BY MY DATA BEING STRUCTURED LIKE THIS:

```
const data = {
  names: [
    {
        "url": "http://pokeapi.co/api/v2/pokemon/1/",
        "name": "bulbasaur"
    },
    // shortened for length
  ]
}
```

THE SOLUTION IS TO PASS IN DATA TO THE TEST

Meaning literally copy and paste the API response from the previous slide into your test

SO THEN WE COPY AND PASTE



AND IN YOUR TEST ADD THIS:

```
const data = {
  names: [
    {
        "url": "http://pokeapi.co/api/v2/pokemon/1/",
        "name": "bulbasaur"
    },
    // shortened for length
  ]
}

it('loads in pokemon list', () => {
  const tree = renderer.create(<App data={[data.names]} />).to
```

NOW LETS TRY THIS AGAIN...

NEW ERROR: NOT A FUNCTION

```
FAIL  tests/app.test.js
● loads in pokemon list

  TypeError: getPokemon is not a function

    at App.componentDidMount (src/App.js:13:4)
```

SO THEN WE GOT TO APP.JS LINE 13 AND WE SEE THIS:

```
componentDidMount() {
  this.getPokemon();
}
```

That is clearly a function...

NOW WE NEED TO TURN THIS FUNCTION INTO A FUNCTION

In the test file, you want to add this outside of your test

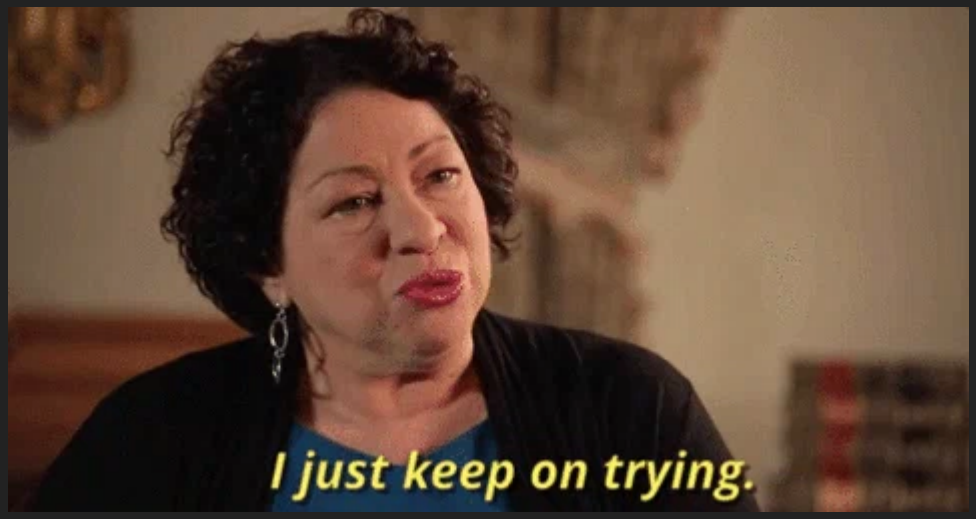
```
const getPokemon = () => {}
```

AND WE CHANGE THE TEST TO THIS:

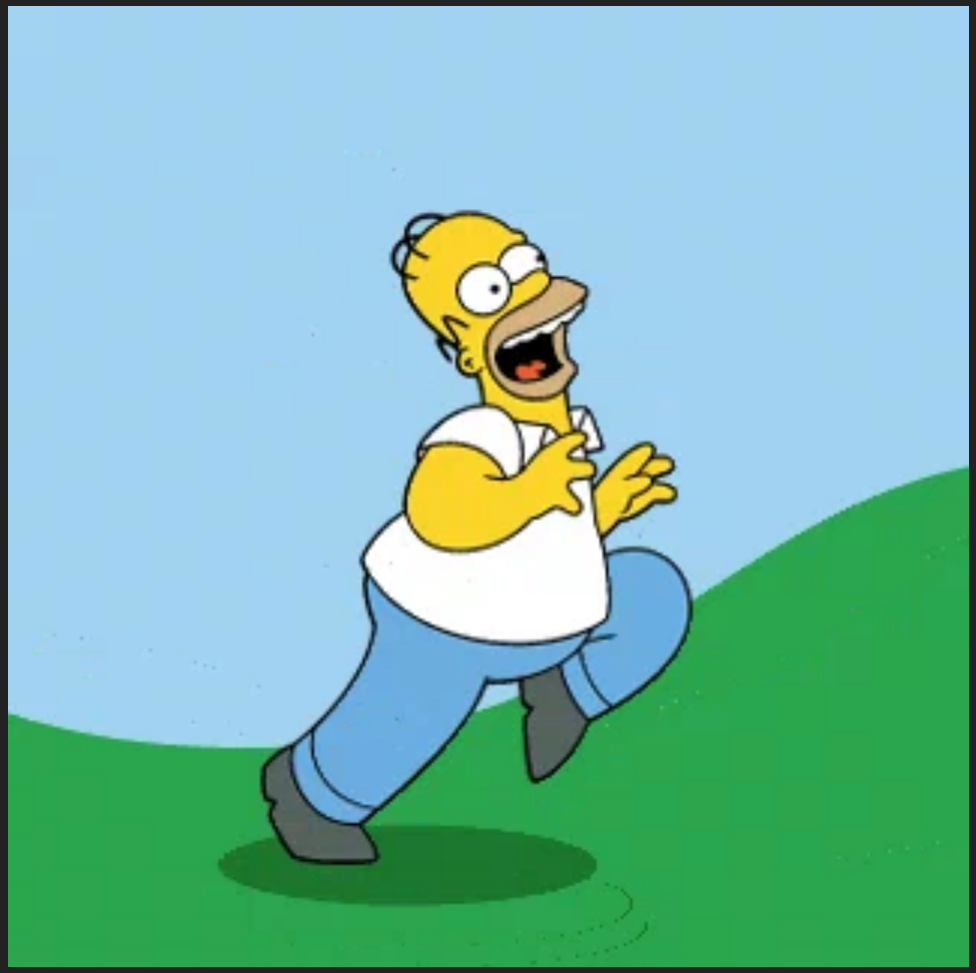
```
const getPokemon = () => {}

it('loads in pokemon list', () => {
  const tree = renderer
  .create(<App getPokemon={getPokemon} data={[data.names]} />)
  .toJSON();
  expect(tree).toMatchSnapshot();
});
```

AND NOW WE TRY THIS TEST AGAIN...



NOW WE HAVE A PASSING TEST!



SUMMARY OF THE PROBLEM

- Jest had a hard time accessing an array inside the `json`
- Jest didn't understand that function called inside of `componentDidMount()` was a function, so we had to make this explicit

OTHER HELPFUL TIPS!

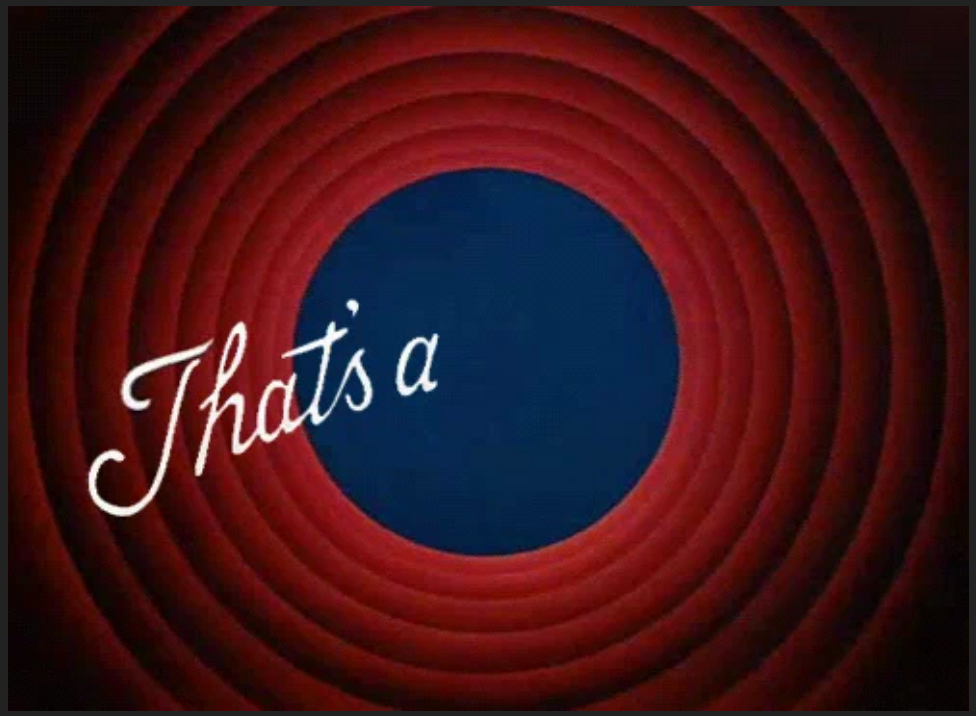
HOW TO MAKE MOCHA WORK WITH JEST

Add this to your package.json

```
"scripts": {
  "test-server": "mocha",
  "test-react": "jest",
  "test": "npm run test-server && npm run test-react"
},
```

Add this to your package.json to get it to work nicely with images, songs, gifs, etc

```
"jest": {
    "moduleNameMapper": {
      "\\.(jpg|jpeg|png|gif|eot|otf|webp|svg|ttf|woff|woff2|mp
      "\\.(css|less)$": "<rootDir>/client/src/stylesheets/"
    },
    "testPathIgnorePatterns": [
      "<rootDir>/node_modules/"
    ]
  },
```


That's a

ANY QUESTIONS?