Have you written tests for that ?

Mike Smith

 @mikerhyssmith

BuzzFeed News Quizzes TV & Movies Shopping Celebrity Sign In LOL win

This one crazy trick lets you deploy at 4pm every Friday

Posted on May 28, 2014, at 7:27 p.m.

Mike Smith
BuzzFeed Staff

This well known approach means you can probably have slightly more confidence deploying at any time but you should still probably not deploy on friday afternoons.

Posted on May 28, 2014, at 7:27 p.m.

Mike Smith
BuzzFeed Staff

oxbotica

CALC-001

Implement Subtract

Attach | Create subtask | Link issue | Link page

Description

Definition of Done

None

Activity Comments

Add a comment...

Pro tip: press **M** to comment

STATUS

To Do

REPORTER

Mike Smith

ASSIGNEE

Mike Smith

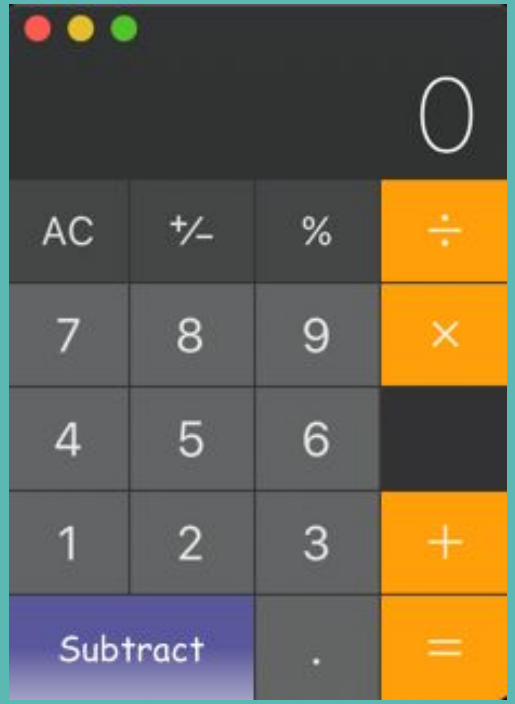
TIME TRACKING

No time logged 1d remaining

SPRINT

```
it("calculator test 3", () ⇒ {
    expect(subtract(6,4)).toEqual(2);
})
```

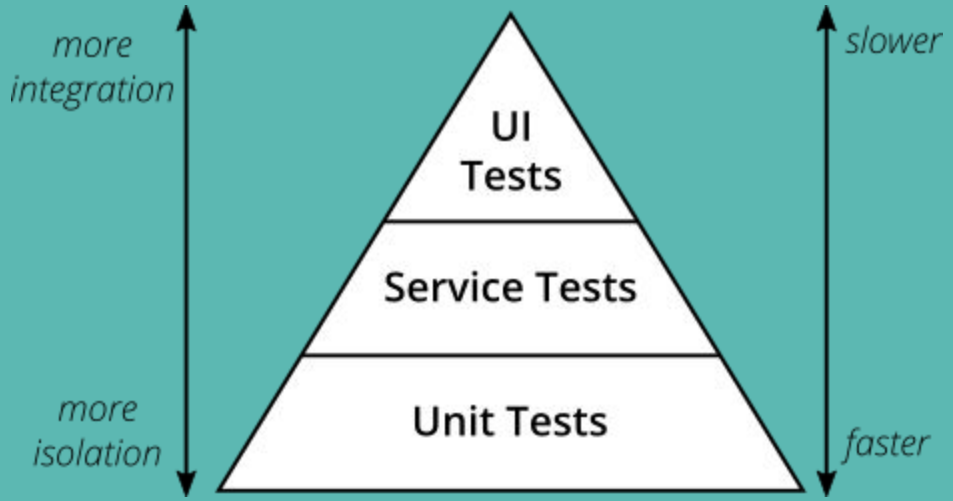
```
PASS test/calculator.test.js
  Calculator
    ✓ calculator test 3 (3ms)

---------------|----------|----------|----------|----------|-------------------|
File           |  % Stmts | % Branch |  % Funcs |  % Lines | Uncovered Line #s |
---------------|----------|----------|----------|----------|-------------------|
All files      |      100 |      100 |      100 |      100 |                   |
 calculator.js |      100 |      100 |      100 |      100 |                   |
---------------|----------|----------|----------|----------|-------------------|
Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
Time:        1.445s
```




```
function subtract(a, b) {
    return 2;
}
```

Have you written THE RIGHT tests for that ?


Pen-testing
Smoke
Integration
Unit
Visual Regression
Mutation
Accessibility
Static
End 2 End
Contract
Performance
Fuzzy



2012 (Martin Fowler) - https://martinfowler.com/bliki/TestPyramid.html

THE FOUR TYPES OF TESTS

End to End

A helper robot that behaves like a user to click around the app and verify that it functions correctly.

Sometimes called "functional testing" or e2e.

Integration

Verify that several units work together in harmony.

Unit

Verify that individual, isolated parts work as expected.

Static

Catch typos and type errors as you write the code.

2018 (Kent Dodds) - https://testingjavascript.com/

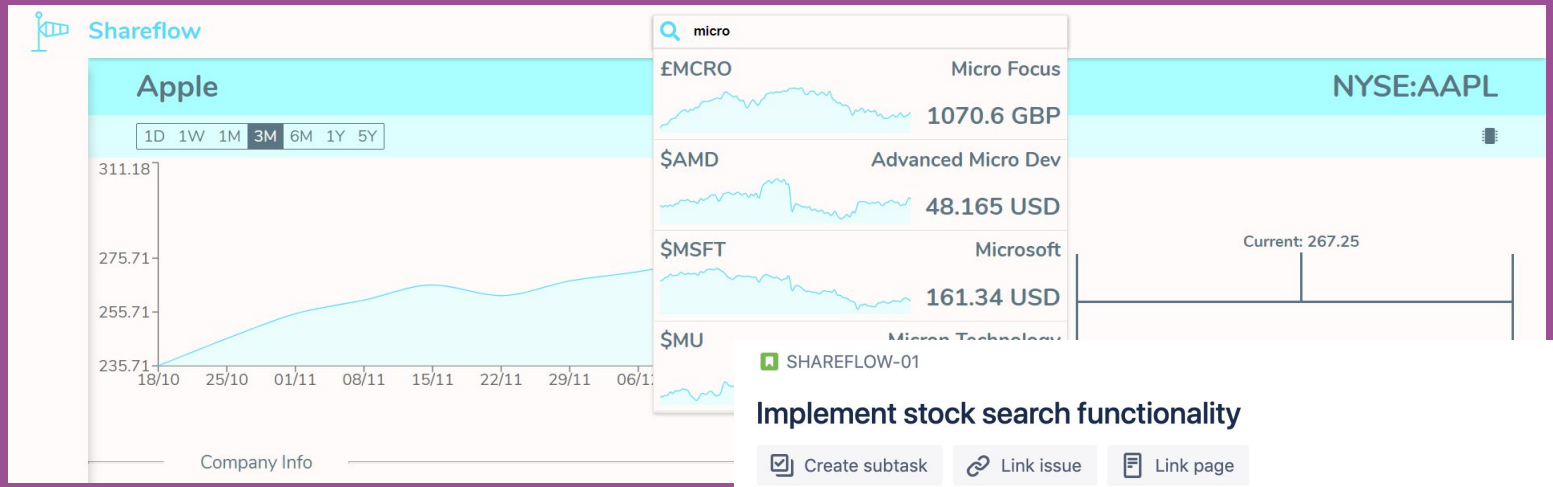
What are the right tests ?

The cheapest test which gives you

1. Confidence
2. Stability
3. Documentation

Cheapest ?

1. Quickest to write
2. Quickest to run
3. Simplest to write
4. Runs on the cheapest resources



SHAREFLOW-01

Implement stock search functionality

Create subtask | Link issue | Link page

Description

As a user I would like to be able to search for stocks

As a user I would like to be able to view price and daily data for a stock

As a user I would like to be able to navigate to view details about these stocks

Definition of Done

None

Activity Comments

Add a comment...

Pro tip: press **M** to comment

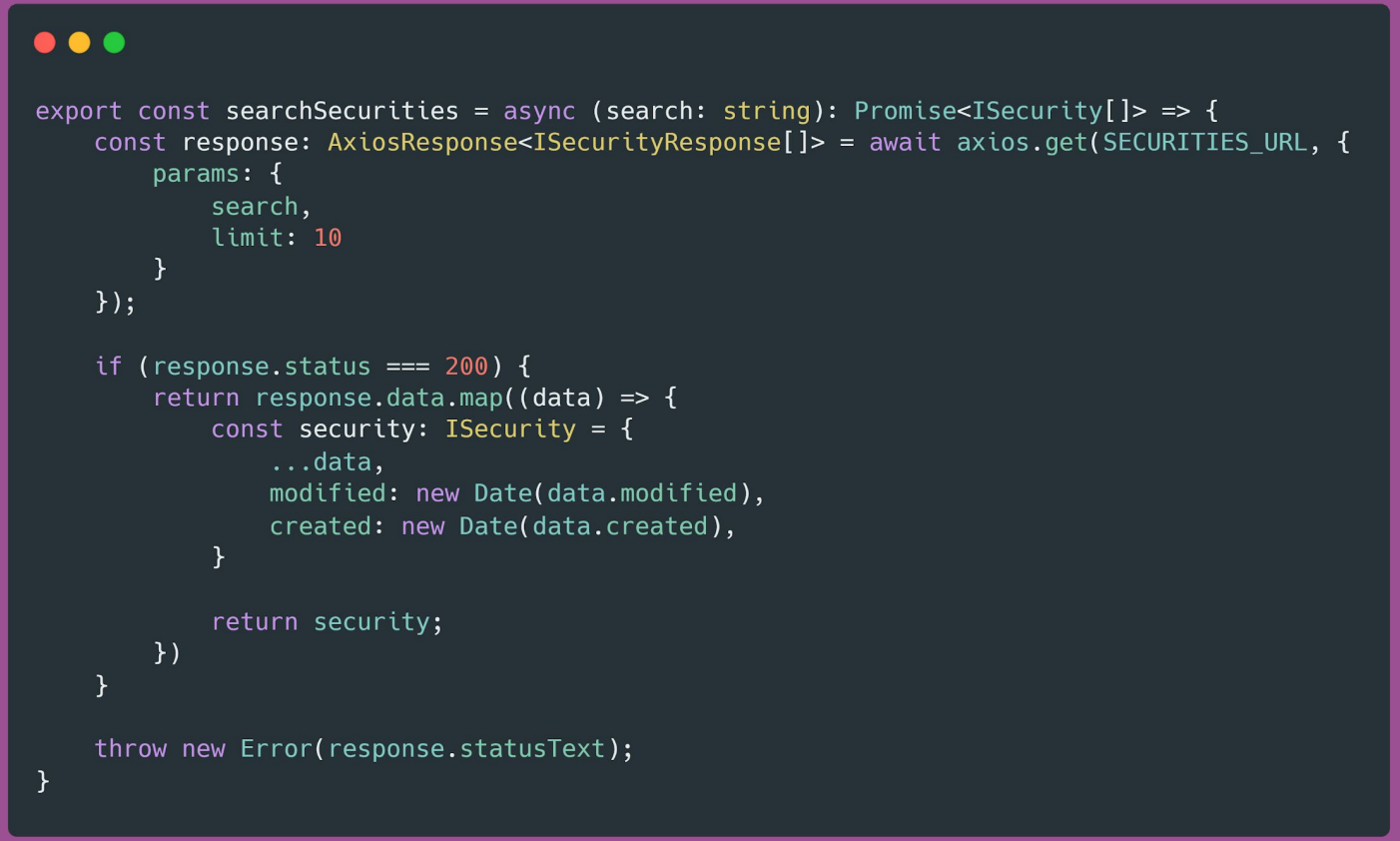
Unit

Writing Unit tests

Given

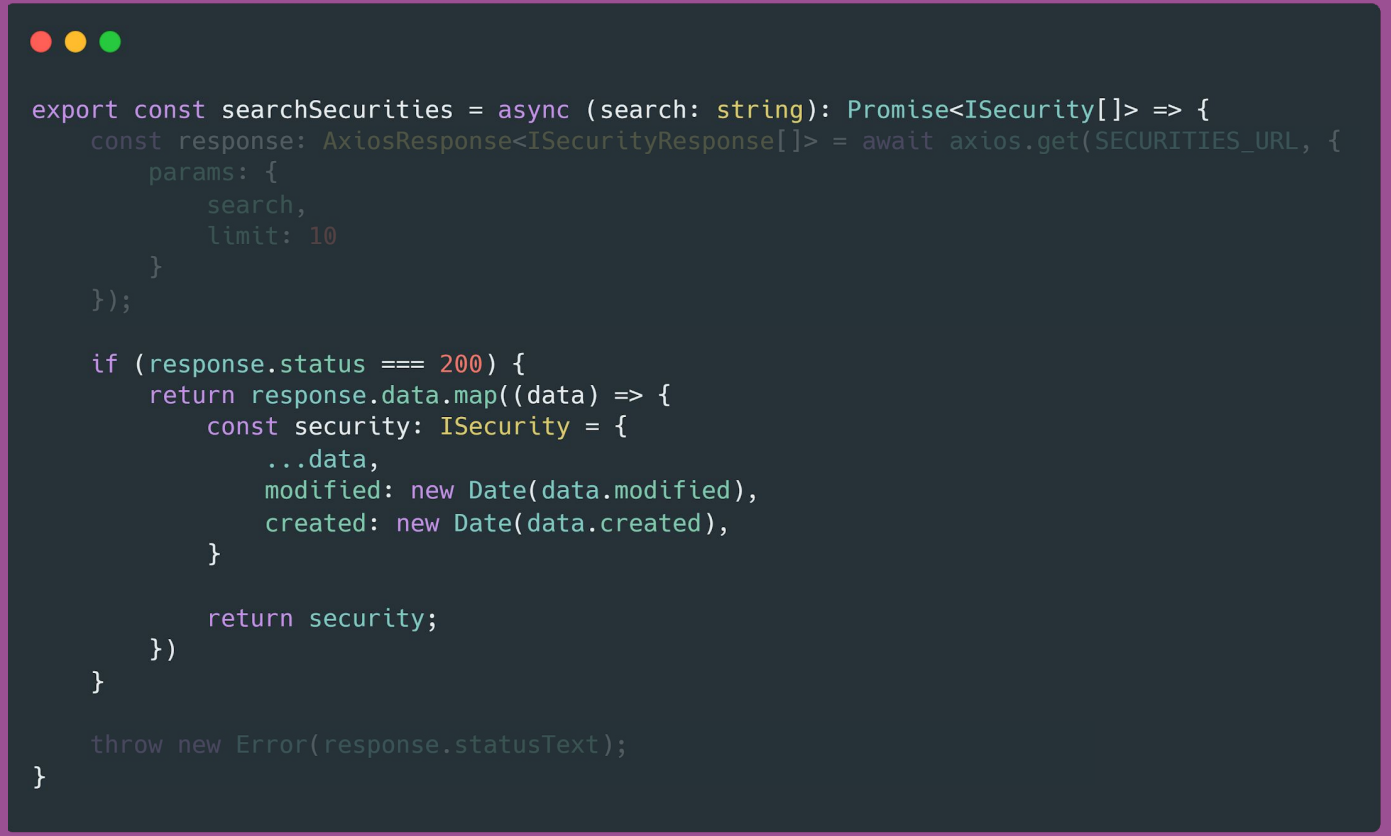
When

Then



```
export const searchSecurities = async (search: string): Promise<ISecurity[]> => {
    const response: AxiosResponse<ISecurityResponse[]> = await axios.get(SECURITIES_URL, {
        params: {
            search,
            limit: 10
        }
    });

    if (response.status === 200) {
        return response.data.map((data) => {
            const security: ISecurity = {
                ...data,
                modified: new Date(data.modified),
                created: new Date(data.created),
            }

            return security;
        })
    }

    throw new Error(response.statusText);
}
```

```
export const searchSecurities = async (search: string): Promise<ISecurity[]> => {
    const response: AxiosResponse<ISecurityResponse[]> = await axios.get(SECURITIES_URL, {
        params: {
            search,
            limit: 10
        }
    });

    if (response.status === 200) {
        return response.data.map((data) => {
            const security: ISecurity = {
                ...data,
                modified: new Date(data.modified),
                created: new Date(data.created),
            }

            return security;
        })
    }

    throw new Error(response.statusText);
}
```



```
export const searchSecurities = async (search: string): Promise<ISecurity[]> => {
    const response: AxiosResponse<ISecurityResponse[]> = await axios.get(SECURITIES_URL, {
        params: {
            search,
            limit: 10
        }
    });

    if (response.status === 200) {
        return response.data.map((data) => {
            const security: ISecurity = {
                ...data,
                modified: new Date(data.modified),
                created: new Date(data.created),
            }

            return security;
        })
    }

    throw new Error(response.statusText);
}
```

```ts
export const searchSecurities = async (search: string): Promise<ISecurity[]> => {
    const response: AxiosResponse<ISecurityResponse[]> = await axios.get(SECURITIES_URL, {
        params: {
            search,
            limit: 10
        }
    });

    if (response.status === 200) {
        return response.data.map((data) => {
            const security: ISecurity = {
                ...data,
                modified: new Date(data.modified),
                created: new Date(data.created),
            }

            return security;
        })
    }

    throw new Error(response.statusText);
}
```

Writing Unit tests

Given

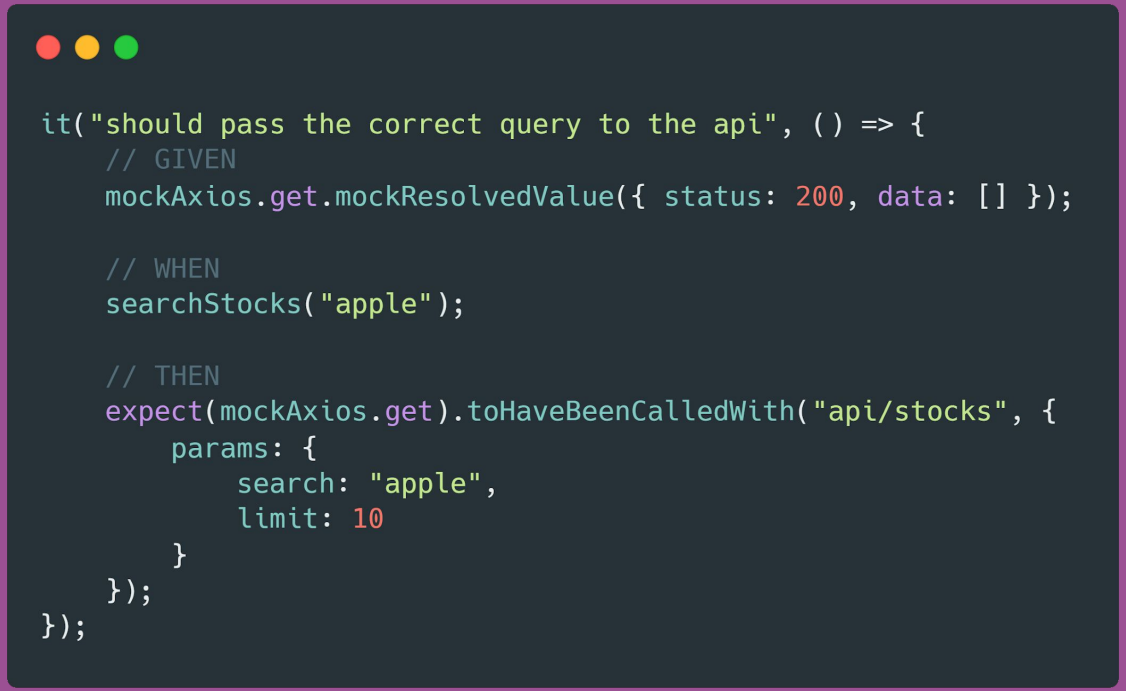
I want to search for a stock

When

I search for apple stock

Then

It should request apple stock from the search endpoint

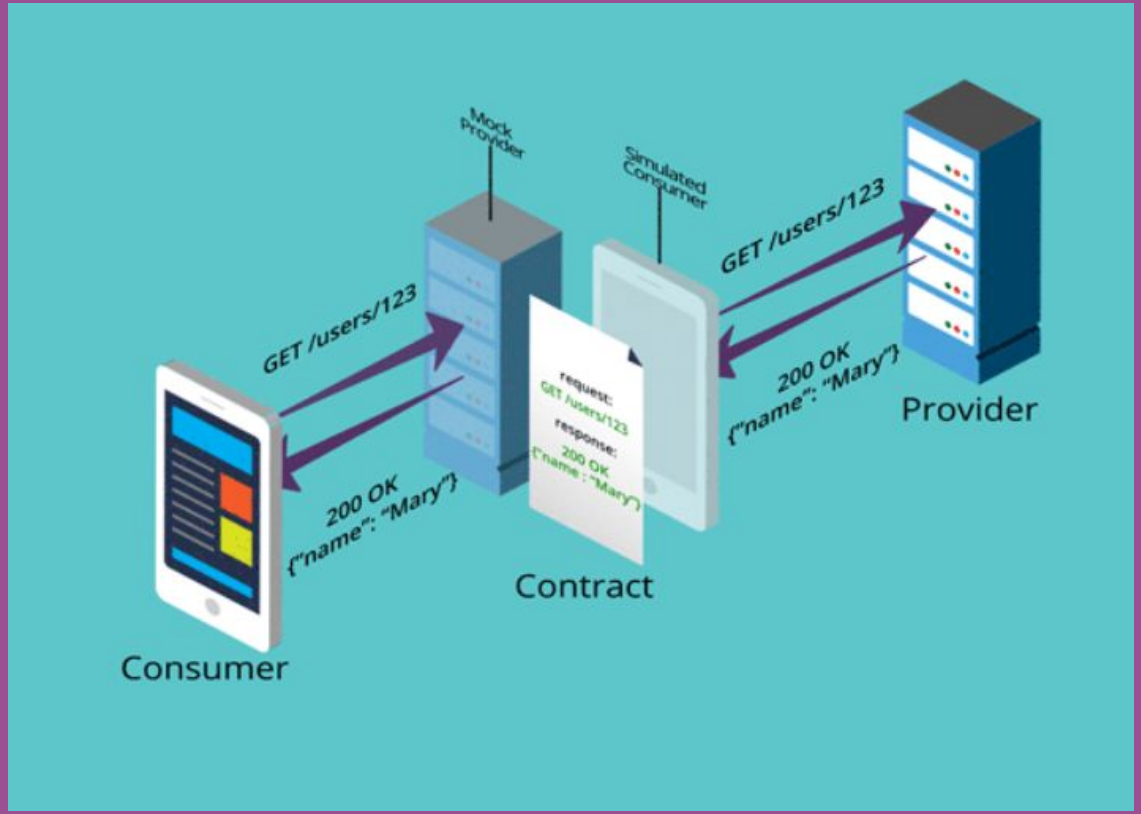


```
it("should pass the correct query to the api", () => {
    // GIVEN
    mockAxios.get.mockResolvedValue({ status: 200, data: [] });

    // WHEN
    searchStocks("apple");

    // THEN
    expect(mockAxios.get).toHaveBeenCalledWith("api/stocks", {
        params: {
            search: "apple",
            limit: 10
        }
    });
});
```

```
it("should return a formatted stock if succesful response", () => {
    // GIVEN
    mockAxios.get
        .mockResolvedValue({ status: 200, data: [MOCK_STOCK] });

    // WHEN
    const result = searchStocks("apple");

    // THEN
    expect(result).toEqual(EXPECTED_STOCK);
});

it("should throw an error if the response is not 200", () => {
    // GIVEN
    mockAxios.get
        .mockResolvedValue({ status: 500, statusText: "API Error" });

    // WHEN
    expect(searchSecurities("apple"))
        // THEN
        .toThrowError("API Error");
});
```

Demo

Contract



https://docs.pact.io/

Demo

Integration

Writing Integration Tests

 - Happy Path

 - Error State

 - Loading state

Writing Integration Tests

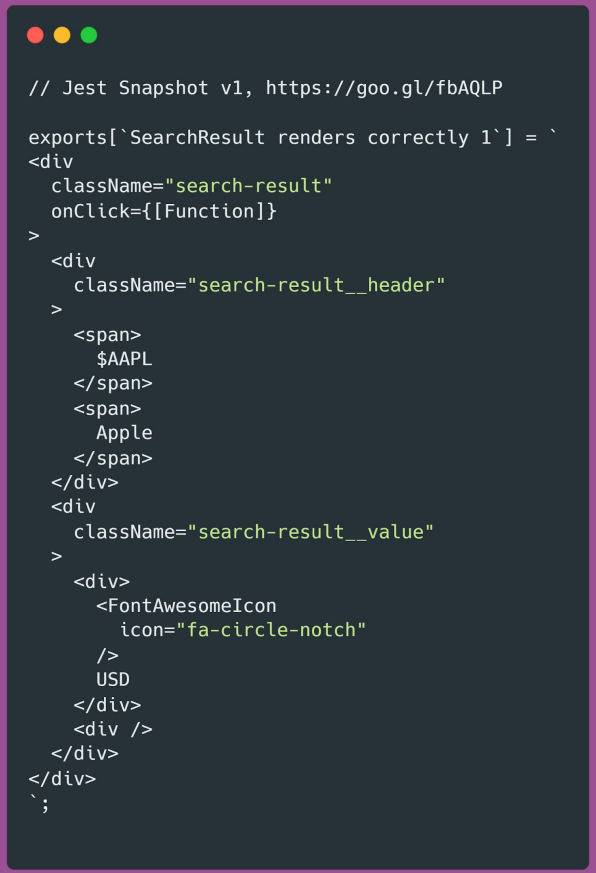
👍 Mocking 👍

Integration Test

Given
    We have price data for a given stock
When
    The search result is rendered
Then
    The price and daily data is rendered

```
it("renders stock data", () => {
    // GIVEN
    firebaseDataMock.mockImplementationOnce(() => ({ values: MOCK_INTRADAY_VALUES }));
    firebaseDataMock.mockImplementationOnce(() => ({ price: MOCK_PRICE }));

    // WHEN
    const wrapper = mount(<SearchResult security={INPUT_SECURITY} onSelect={ON_SELECT_MOCK} />);

    // THEN
    expect(wrapper.find(".search-result__value").text())
        .toContain("200");

    expect(wrapper.find(AreaChart).prop("data")).toEqual(
        [
            { value: 10, name: new Date(1549238400000) },
            { value: 20, name: new Date(1549324800000) }
        ]
    );
});
```

```
it("renders correctly", () => {
    const wrapper = shallow(
        <SearchResult
            security={INPUT_SECURITY}
            onSelect={ON_SELECT_MOCK}
        />)

    expect(wrapper)
        .toMatchSnapshot();
});
```



```
// Jest Snapshot v1, https://goo.gl/fbAQLP

exports[`SearchResult renders correctly 1`] = `
<div
  className="search-result"
  onClick={[Function]}
>
  <div
    className="search-result__header"
  >
    <span>
      $AAPL
    </span>
    <span>
      Apple
    </span>
  </div>
  <div
    className="search-result__value"
  >
    <div>
      <FontAwesomeIcon
        icon="fa-circle-notch"
      />
      USD
    </div>
    <div />
  </div>
</div>
`;
```

Demo

Visual Regression

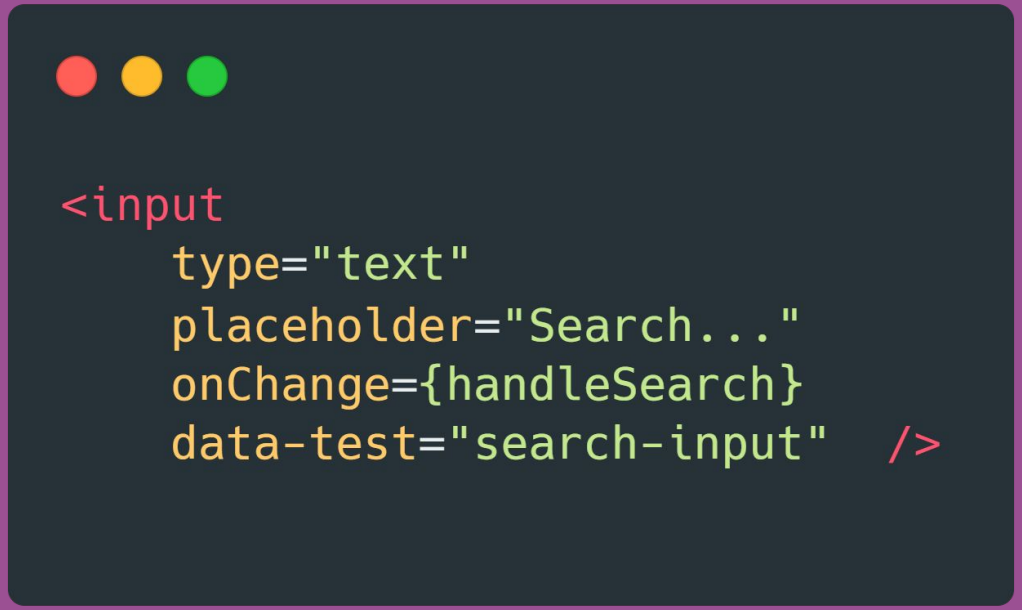
Demo

End 2 End Tests

E2E Test - Page Objects

```
import { Selector, t } from "testcafe";

export class Search {

    get searchInput() {
        return Selector('[data-test="search-input"]')
    }

    get searchResult() {
        return Selector('[data-test="search-result"]')
    }

    public enterSearchQuery(query: string) {
        return t.typeText(this.searchInput, query);
    }

}
```

E2E Test - data-test



```
<input
    type="text"
    placeholder="Search..."
    onChange={handleSearch}
    data-test="search-input"  />
```

E2E Test

As a user I want to be able
to search for a given stock
view its price data and
navigate to a page to find all of its
data

E2E Test

```
fixture`Search`
    .page("localhost:3000")


test("search for stock", async (t: TestController) => {
    await searchPage.enterSearchQuery("apple");

    await t.expect(searchPage.searchResult.count).eql(1);

    await t.expect(searchPage.priceChart.exists).eql(true);

    await t.click(searchPage.searchResult);

    await t.expect(stockPage.isRendered).eql(true);
    await t.expect(stockPage.stockName).eql("Apple");
})
```

E2E Test

Accessibility

(With Paypal's Automated Accessibility Testing Tool (AATT))

Error Level	Principle	Message	Code snippet	Techniques
error	Perceivable	This element has insufficient contrast at this conformance level. Expected a contrast ratio of at least 3:1, but text in this element has a contrast ratio of 1.54:1. Recommendation: change text colour to #001519.	Shareflow</span gt;	G145
error	Robust	This textinput element does not have a name available to an accessibility API. Valid names are: label element, title undefined, aria-label undefined, aria-labelledby undefined.	<input data-test="search-input" type="text" placeholder="Search...">	H91
error	Perceivable	This form field should be labelled in some way. Use the label element (either with a "for" attribute or wrapped around the form field), or "title", "aria-label" or "aria-labelledby" attributes as appropriate.	<input data-test="search-input" type="text" placeholder="Search...">	F68
notice	Operable	Check that the title element describes the document.	<title>Home - PayPal Accessibility Tool</title>	H25
notice	Understandable	Check that a change of context does not occur when this input field receives focus.	<input data-test="search-input" type="text" placeholder="Search...">	G107
notice	Perceivable	Check that the content is ordered in a meaningful sequence when linearised, such as when style sheets are disabled.		
warning	Perceivable	This element is absolutely positioned and the background color can not be determined. Ensure the contrast ratio between the text and all covered parts of the background are at least 4.5:1.	297.43	G18
warning	Perceivable	This element is absolutely positioned and the background color can not be determined. Ensure the contrast ratio between the text and all covered parts of the background are at least 4.5:1.	311.27	G18

Demo

Mutation

```
export const stockPrefixer = (stock: string, exchange: string) => {
  if (exchange === "LSE") {
      return `LON:${stock}`
  } else if (exchange = "NYSE") {
      return `NYSE:${stock}`;
  }
}
```

```
describe("stockPrefixer", () => {
  it("should add prefix for lon", () => {
      expect(stockPrefixer("PRTC", "LSE")).toEqual("LON:PRTC");
  })

  it("should add prefix for nyse", () => {
      expect(stockPrefixer("AAPL", "NYSE")).toEqual("NYSE:AAPL");
  })
})
```

https://stryker-mutator.io/

Summary

1. Divide and conquer with tests the same way you do with code

2. Try and find the cheapests tests possible to give you the confidence to ship your work

3. If you have very specialist requirements there is always something out there to help gain that confidence

oxbotica

Thank you !

Mike Smith

 @mikerhyssmith