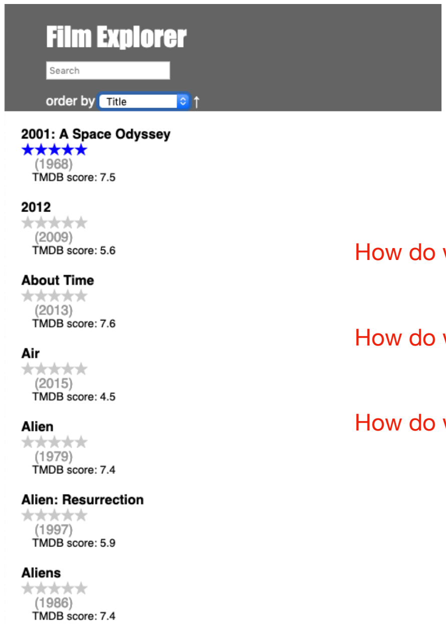
CS 312 Software Development

Testing React

Testing React



Film Explorer

Search

order by Title ↑

2001: A Space Odyssey
★★★★★
(1968)
TMDB score: 7.5

2012
★★★★★
(2009)
TMDB score: 5.6

About Time
★★★★★
(2013)
TMDB score: 7.6

Air
★★★★★
(2015)
TMDB score: 4.5

Alien
★★★★★
(1979)
TMDB score: 7.4

Alien: Resurrection
★★★★★
(1997)
TMDB score: 5.9

Aliens
★★★★★
(1986)
TMDB score: 7.4

How do we test that the ordering is correct?

How do we test opening and closing the detail view?

How do we test clicking on the stars?

Testing Library

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

- Kent C. Dodds

- `Test DOM nodes, not components`
- `Tests should work the way the application is to be used`

https://testing-library.com/docs/guiding-principles

Testing Library

React Testing Library

- `render(component)`
  - Takes in a React component and performs a virtual render
  - can accept JSX: render(<FilmExplorer />)
  - returns an object containing the rendered component, a rerender function, and all of the queries
- `rerender(component)`
  - returned by **render**, used to change props on a mounted component
- `cleanup()`
  - unmount React trees (we don’t need to do this since we use jest)
- `act()`
  - wrapper around React act() function — make sure React tasks are complete

Testing Library
Query variants

- `getBy* / getAllBy*`
  - queries the DOM for the first matching node/array of matches
  - throws error if none are found
  - single version throws error if more than one is found
- `queryBy* / queryAllBy*`
  - queries the DOM for the first matching node/array of matches
  - returns null or empty [] if none are found
  - single version throws error if more than one is found
- `findBy* / findAllBy*`
  - returns a Promise which resolves when a matching node(s) is found
  - throws error after 1000ms if none are found
  - single version throws error if more than one is found

Testing Library
Query types (partial list)

- `ByText`
  - looks for an element based on the text contents of the node
- `ByRole`
  - search based on the role of the component (e.g., listitem, button, textbox, etc…)
  - can narrow the search with options like the aria-label
- `ByTestId`
  - looks for specific components based on `data-testid` value
  - this is basically the cheat code and not really in the spirit of the library

A query is a variant + a type
e.g., queryByText() or findAllByRole

Testing Library
Tools

- `screen`
  - allows us to query the entire DOM (`screen.findByText('example')`)
  - screen.debug() prints out the virtual DOM
- `fireEvent.`*`type`*`(component, event properties)`
  - allows us to simulate user interaction
  - *type* is any kind of HTML event: click, change, drag, drop, keyDown, etc…

Matchers: jest-dom

- `toBeDisabled`
- `toBeEnabled`
- `toBeEmpty`
- `toBeEmptyDOMElement`
- `toBeInTheDocument`
- `toBeInvalid`
- `toBeRequired`
- `toBeValid`
- `toBeVisible`
- `toContainElement`
- `toContainHTML`
- `toHaveAttribute`
- `toHaveClass`
- `toHaveFocus`
- `toHaveFormValues`
- `toHaveStyle`
- `toHaveTextContent`
- `toHaveValue`
- `toHaveDisplayValue`
- `toBeChecked`
- `toBePartiallyChecked`
- `toHaveDescription`

https://github.com/testing-library/jest-dom

Example

uses mock function for callback

finds the section

renders the component

```
test('Clicking on a section displays titles', async () => {
    const selectFunction = jest.fn();
    render(<IndexBar collection={articles} select={selectFunction} />);
    const section = await screen.findByText(sampleSections[0]);

    fireEvent.click(section);

    const titles = await screen.findAllByTestId('title');

    const expectedArticles = articles.filter(
      (article) => article.title.charAt(0).toUpperCase() === sampleSections[0]
    );

    expect(titles).toHaveLength(expectedArticles.length);

    expectedArticles.forEach((article) => {
      expect(screen.getByText(article.title)).toBeVisible();
    });
  });
```

clicks on the section

finds all titles

tests that each title is visible

Mocking fetch

fetch-mock-jest

```
        beforeAll(() => {
            localFilms = films.map((film) => ({ ...film }));

            fetchMock.reset();
            fetchMock.get(
              `${server}/api/films`,
              () => localFilms
            );
            fetchMock.put(
              `${server}/api/films/102`,
              (url, options) => {
                const id = 102;

                const modifiedFilm = JSON.parse(options.body);
                localFilms = localFilms.map((film) => {
                  if (id === film.id) {
                    return modifiedFilm;
                  } else {
                    return film;
                  }
                });
                return modifiedFilm;
              }
            );
```

http://www.wheresrhys.co.uk/fetch-mock/