Testing React

Testing with Enzyme

- **shallow()**
  - renders component, but not the contents of any children
- **mount()**
  - renders the entire hierarchy
- **render()**
  - does a full render, but a static one

Challenge: fetch

```javascript
useEffect(() => {
  const fetchData = () => {
    fetch('/api/films/').
      .then(response => {
          if (!response.ok) {
            throw new Error(response.status_text);
          }
          return response.json();
        }).
      .then(data => { 
          setFilms(List(data));
        }).
    .catch(err => console.log(err));
  };
  fetchData();
}, []);
```

Solution: mockFetch

```javascript
const mockResponse = (data) => {
  return Promise.resolve({ 
    ok: true, 
    json: () => Promise.resolve(data) 
  });
};

const mockFetch = (url, options) => {
  if (options) {
    if (options.method === 'PUT') {
      const data = JSON.parse(options.body);
      return mockResponse(data);
    }
  } else {
    return mockResponse(films);
  }
};
```
Testing with hooks

• `useState()` hides the state from Enzyme
  • testing state is dubious value anyway

• hooks cause re-renders that are hard to capture
  • the `act()` function should be used around code that will lead to a state update (mounting, interacting, etc…)

```javascript
act(() => {
  // render components
  comp.update();
  // make assertions
});
```

Enzyme and `act()`

• Enzyme uses `act()` internally so we don’t need to
• But, our mocked fetch() is asynchronous…

```javascript
const comp = mount(<FilmExplorer />);
await act(async() => await flushPromises());
comp.update();
```

Initial render

update flushes the pending re-render from the state change

wait for Promises to resolve — wrapped in act() because a state change will happen