24. Unit & Integration Testing

Github repo

To Do

- [ ] Tools for testing React
- [ ] Add Features/Tests to our App
- [ ] `debug()` and `prettyDOM()`
- [ ] Mocking AJAX Requests and Functions
- [ ] Coverage Reports

Setup & Teardown

- Tests should represent how a user (or other code) would interact with our application
- It's important to properly setup the test conditions to isolate the piece of functionality under test
- Once the test has been executed, tear down all setup to leave no traces for the next test
- It's important to scope variables appropriately to make sure that there won't be leaks or interference with other tests

Tools for testing React

- Jest
  - Jest is the framework we use to run our tests
  - Comes with `create-react-app`, so no need to configure
  - `npm run test` will start Jest in watch mode and run the tests
- DOM Testing Library
  - A set of tools to help target DOM elements and trigger DOM events
- React Testing Library
  - Built on top of the DOM Testing Library, gives us more possibilities to target and render React elements to make them possible to test
- JestDOM
  - JestDOM is a set of matchers (like `.toHaveClass()` or `.toBeVisible()`) to help target elements in the DOM

Passing Flags to Scripts

- We can define our own scripts in `package.json`

```
"scripts": {
  "start": "react-scripts start",
  "build": "react-scripts build",
  "test": "react-scripts test",
  "eject": "react-scripts eject",
  "list": "ls"
}
```

- We can run these scripts with `npm run script-name` or `yarn script-name`
- We can also pass flags to our scripts
- Using `npm`, we have to add `--` before passing flags

```
npm run script-name -- --flag-name

yarn script-name --flag-name
```

- Eg. to pass "-la" to our `list` script, we'd use `npm run list -- -la` or `yarn list -la` (try it yourself!)

Coverage Reports

- A coverage report shows us how much of our code is covered by the tests we've written
- The code coverage of our tests is important, but it's more important to have solid tests with a little less coverage than easy tests with a lot of coverage
- It's okay to not have 100% coverage, it's almost impossible!
- `npm run test -- --coverage` will start Jest in watch mode and show the coverage status after each test
- If you notice that your coverage report is empty, add the `watchAll=false` flag

```
npm run test -- --coverage --watchAll=false
```

Add Features/Tests to App

- TDD: unit test
  - choose a valid response for the computer player (currently hard-coded)

- TDD: integration test
  - clicking on the robot head will toggle the cheating boolean
- mocking
  - test fetching high scores (mock Axios)

`getBy` & `queryBy`

- One small thing about `getBy` and `queryBy` to be aware of is that `getBy` will throw an error if the element is not found
- `queryBy` will return only null, so it's up to the context to guide you which you should use

Skipping Tests

- For various reasons, you might want to skip a particular test
- To skip a test, use either `xit` or `test.skip`

```
// using test
test('this test will run', () => {});
test.skip('this test will be skipped', () => {});

// using it
it('this test will run', () => {});
xit('this test will be skipped', () => {});
```

Notes and example app based on lectures by Andy and Francis!

Useful Links

- DOM Testing Library
- React Testing Library
- Which query should I use?
- Jest-DOM
- Testing Library Async Functions
- Jest --coverage issue

View on Compass | Leave Feedback