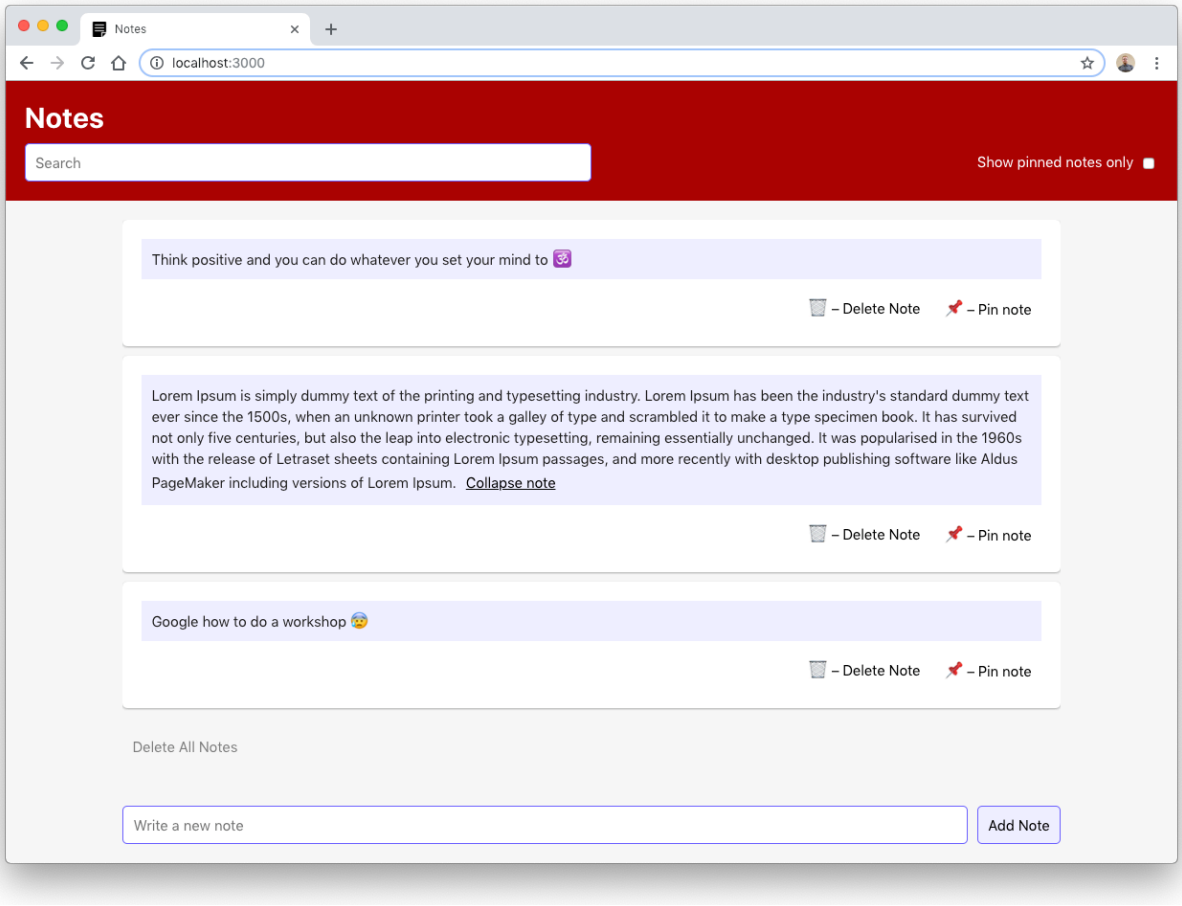
# Intro to Frontend Testing

## Workshop at Hamburg Coding School

Henning Muszynski - 23. January 2019



# Assignment 1: Unit Testing

Remember Unit Testing is used to verify that individual and isolated parts (i.e. units) of our application work as expected. Therefore we will start by testing the utils functions we use throughout the application:

Setting things up:

- [ ] Run `yarn install` in the project directory to install all dependencies
- [ ] Create new directory `__test__` in the `utils` directory
- [ ] Create your first test file called `Filter.test.js` in the newly created `__test__` directory

Test your first function:

- [ ] import the `filterPinnedNotes` utility function from the `Filter` module in the directory above
- [ ] Write your first test to ensure that the `filterPinnedNotes` function returns all pinned notes in an array of notes
  - [ ] Hint: You need to create the notes you pass to the function manually in your test. An exemplary note might look like this:
    `{ id: 1, text: “Test Note”, pinned: true }`
- [ ] Run jest with `yarn run` and make sure your first test passes
  - [ ] Take a moment to celebrate this achievement 🎉🙌
- [ ] Write another test to ensure filterPinnedNotes returns an empty array when being called without any notes

Go ahead test the rest of the module:

- [ ] import `removeNoteById` from the `Filter` module
- [ ] Write a test to ensure that the note with the given id is deleted from the array
- [ ] Write a test to ensure that an empty array is returned when no notes are given
- [ ] import `filterByText` from the `Filter` module
- [ ] Write a test to ensure that the note with the given id is deleted from the array
- [ ] Write a test to ensure that an empty array is returned when no notes are given

Now we will go over the second aspect of Unit Testing: testing components in isolation:

- [ ] In the `components` directory you will find `__test__` directory. Create a new file `Note.test.js` inside of it
- [ ] In your newly created component test file please import `React` (from the package `react`), `render`, `fireEvent` and `cleanup` from the package `react-testing-library` and the `Note` component which lies in the directory above
  - [ ] Hint: You can peek in the other component test files in this directory to get you started
- [ ] Write a test that renders a note and assert that it has an element with `note-content` testId that has the correct textContent
  - [ ] Also assert that the elements with the testIds `delete-note` and `pin-note` are defined
- [ ] Make sure you call `cleanup` after each test
- [ ] Write a test that renders a note, clicks the delete button and asserts that `onDeleteClick` handler has been called
- [ ] Write a test that renders a note clicks the pin button and asserts that the `onPinClick` handler has been called
- [ ] Write a test that renders a pinned note clicks the unpin button and asserts that the `onPinClick` handler has been called

# Assignment 2: Integration Testing

We will now go on verify that multiple units of our app work together as expected:

- [ ] In the top level `__test__` directory (inside the `src` folder) create a new integration test file called `AppCanCreateNotes.test.js`
- [ ] In the integration test file import `React` from the `react` package, `render` and `fireEvent` from the `react-testing-library` package and the `App` component from the directory above
- [ ] Write a test that ensures the app can be used to create a note:
  - [ ] Render the `App`
  - [ ] Assert that the `composer-input` and `submit-button` are displayed
  - [ ] Assert that no notes are displayed
  - [ ] Change the value of the `composer-input` to a note text and click the `submit-button`
  - [ ] Assert that a note with the entered text exists

# Assignment 3: End to End Testing

Now we're going into end to end testing. We'll start our app and write a test that will click through to ensure the critical paths and features are working. We'll also take some screenshots of the main views of our app to ensure the visual integrity is maintained when developing new features:

- [ ] Start the app by running yarn start
- [ ] Start cypress by running yarn cypress:open
- [ ] In the integration directory create a new test file named app.spec.js
- [ ] Visit the app which should be running on http://localhost:3000 and take a screenshot of the initial view
- [ ] Create two notes, one of them with a longer text (to ensure it gets collapsed)
- [ ] Pin the second note
- [ ] Reload the page and assert that both notes are displayed
- [ ] Take another screenshot of the notes list
- [ ] Type something in the search input to ensure only one note is matched and filtering by text works
- [ ] Reset the search input and toggle the filter for pinned notes to ensure only one note is matched filtering pinned notes work
- [ ] Click the delete all button and confirm the deletion