TDD - React

9:30 - Setup React App
10:00 - Write first tests
11:00 - Implement React Component
12:00 - Lunch
13:00 - Refactor & Mocking
14:00 - Second React Component

Plan for the day

Objectives

- Understand ATDD & TDD
- Understand testing within React App
- Understand when to use Jest & when to use Cypress
- Understand Cypress best practices e.g. selectors

Setup

npm init react-app my-app

cd my-app

yarn start

React

http://localhost:3000

Testing

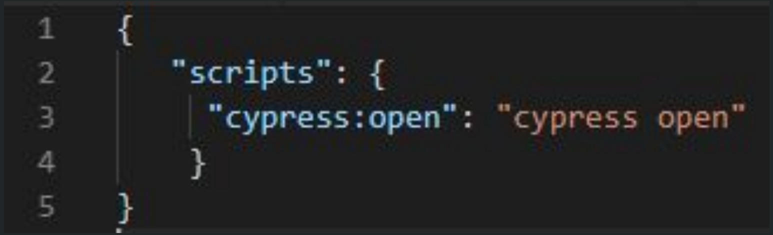
```
yarn add --dev
@testing-library/react
```

React

Components

Cypress

yarn add --dev cypress



```
{
    "scripts": {
      "cypress:open": "cypress open"
     }
}
```

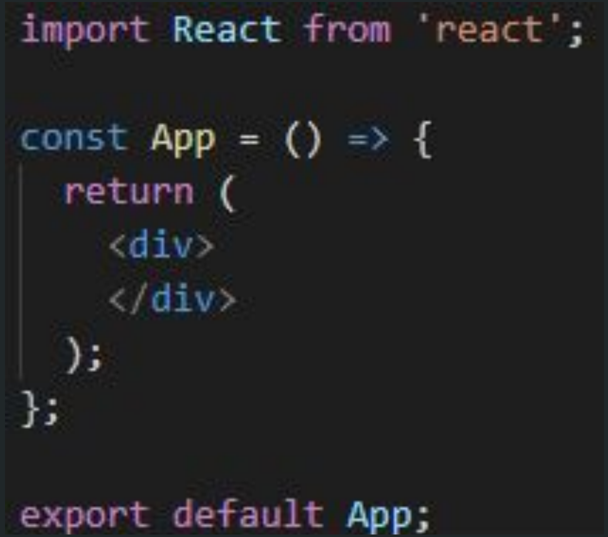
npm run cypress:open

Delete

All the following files and folders:

- cypress/integration/examples/
- src/App.css
- src/App.test.js
- src/logo.svg

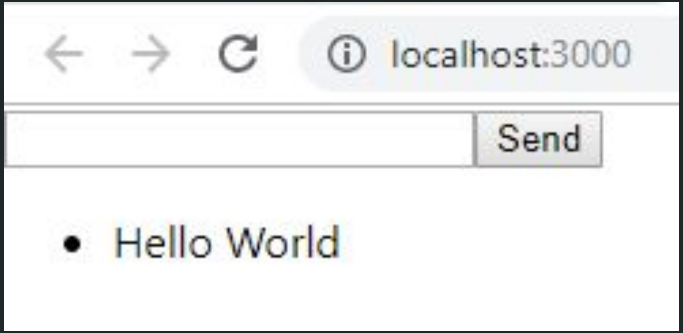
Replace the contents of src/App.js with the following:



```
import React from 'react';

const App = () => {
  return (
    <div>
    </div>
  );
};

export default App;
```

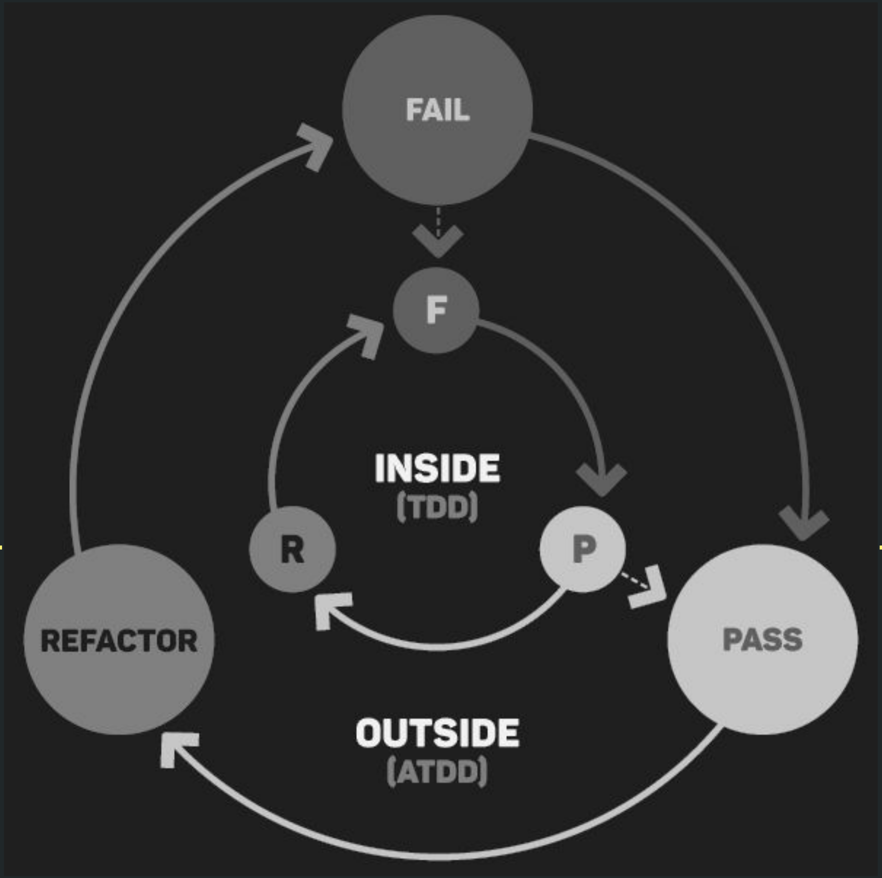
List App

List of items

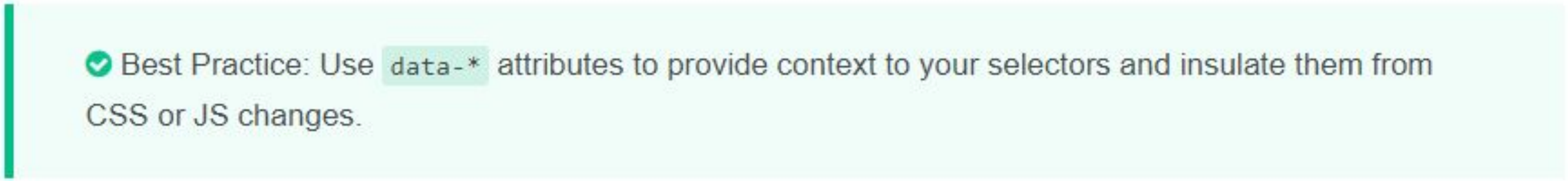


- Hello World

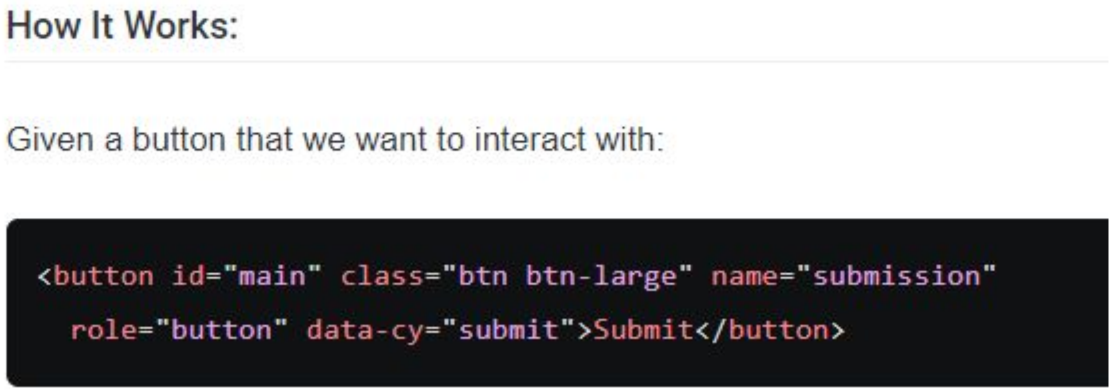
ATDD



Cypress - Best Practices



✓ Best Practice: Use `data-*` attributes to provide context to your selectors and insulate them from CSS or JS changes.



How It Works:

Given a button that we want to interact with:

```
<button id="main" class="btn btn-large" name="submission"
  role="button" data-cy="submit">Submit</button>
```

Task 1

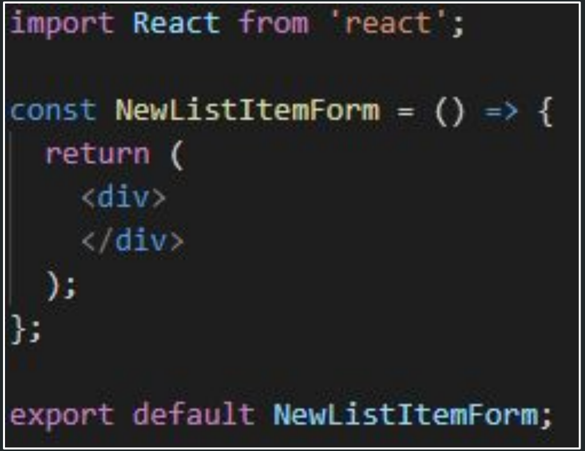
Create your cypress test:

new_item_form.spec.js

1. Visiting the web site
2. Entering the text “New message” into a message text field
3. Clicking a send button
4. Confirming that the message text field is cleared out

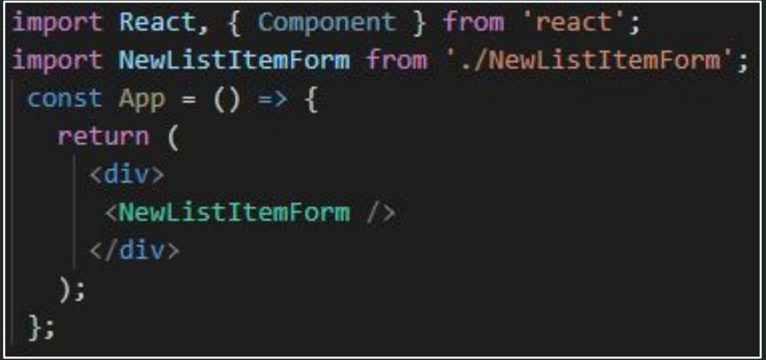
Run cypress test

NewListItemForm.js



```
import React from 'react';

const NewListItemForm = () => {
  return (
    <div>
    </div>
  );
};

export default NewListItemForm;
```

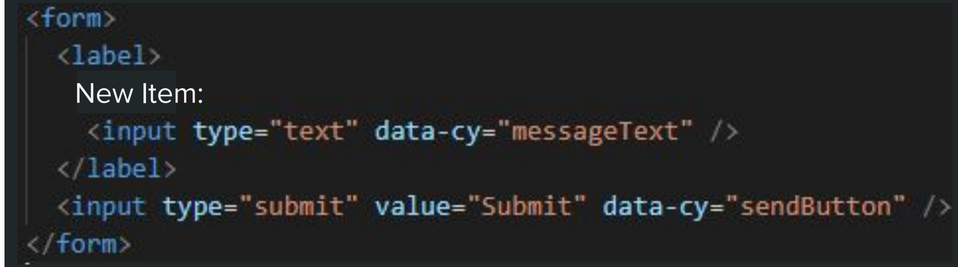
App.js



```
import React, { Component } from 'react';
import NewListItemForm from './NewListItemForm';
 const App = () => {
   return (
     <div>
      <NewListItemForm />
     </div>
   );
 };
```

React - Forms

https://reactjs.org/docs/forms.html

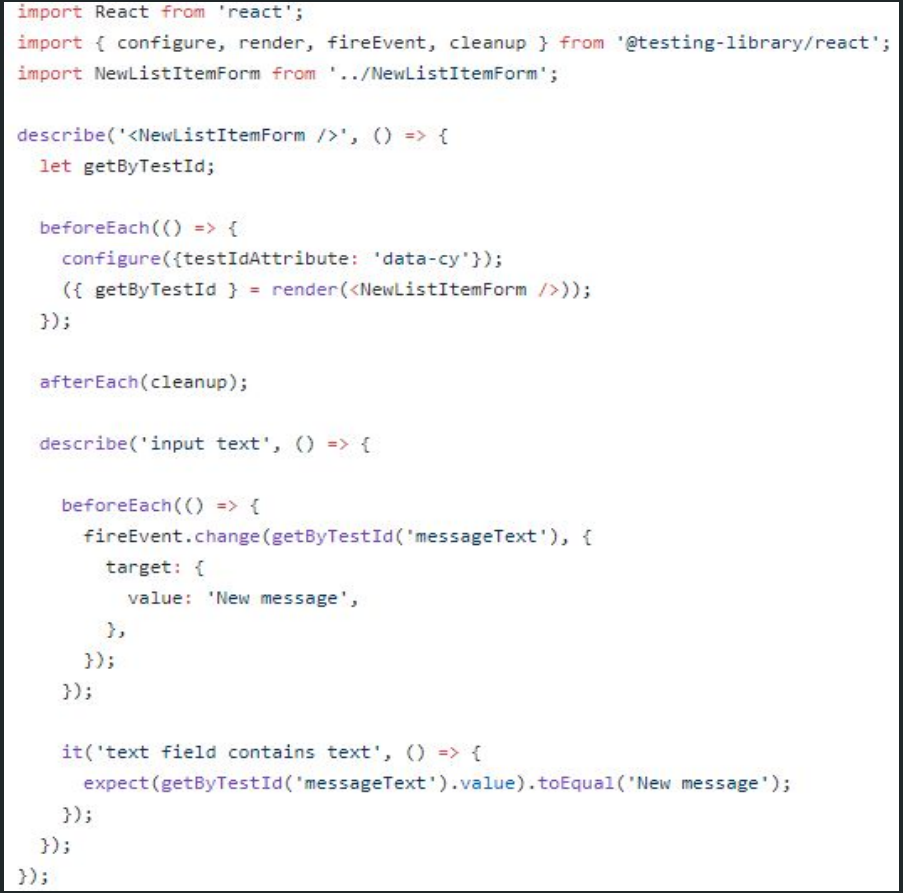


```
<form>
  <label>
    New Item:
    <input type="text" data-cy="messageText" />
  </label>
  <input type="submit" value="Submit" data-cy="sendButton" />
</form>
```

Task 2

Fix failing test

Unable to find an element by: [data-cy="messageText"]



```
import React from 'react';
import { configure, render, fireEvent, cleanup } from '@testing-library/react';
import NewListItemForm from '../NewListItemForm';

describe('<NewListItemForm />', () => {
  let getByTestId;

  beforeEach(() => {
    configure({testIdAttribute: 'data-cy'});
    ({ getByTestId } = render(<NewListItemForm />));
  });

  afterEach(cleanup);

  describe('input text', () => {

    beforeEach(() => {
      fireEvent.change(getByTestId('messageText'), {
        target: {
          value: 'New message',
        },
      });
    });

    it('text field contains text', () => {
      expect(getByTestId('messageText').value).toEqual('New message');
    });
  });
});
```

https://bit.ly/35uYEDO

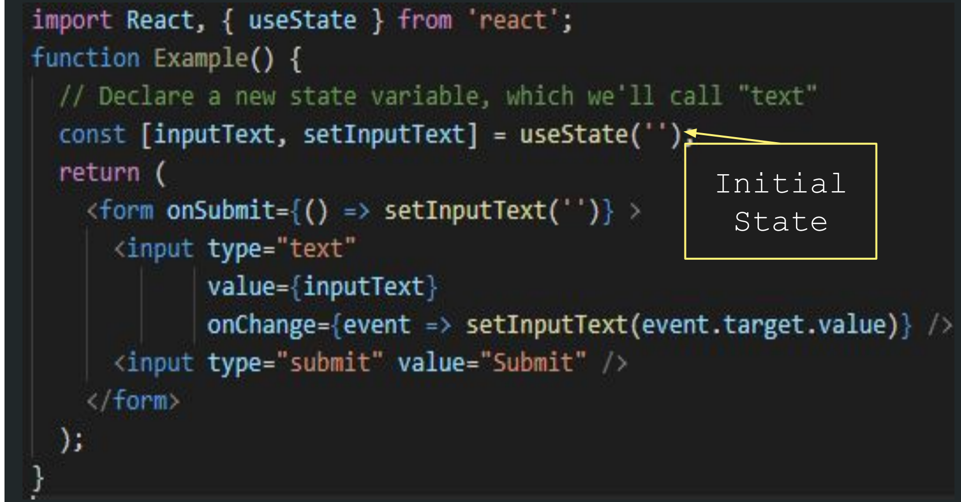
Can you stub html requests in cypress for the test?
https://bit.ly/34tBZqp

React - State

https://reactjs.org/docs/hooks-overview.html#state-hook

Requirement:

Clear input field text when form is submitted



```
import React, { useState } from 'react';
function Example() {
  // Declare a new state variable, which we'll call "text"
  const [inputText, setInputText] = useState('');
  return (
    <form onSubmit={() => setInputText('')} >
      <input type="text"
             value={inputText}
             onChange={event => setInputText(event.target.value)} />
      <input type="submit" value="Submit" />
    </form>
  );
}
```

Task 3

Add “clear text from input field” test

Before

```
fireEvent
    .submit(getByTestId('submitButton'));
```

Add css styling
https://bit.ly/2S7Sje1

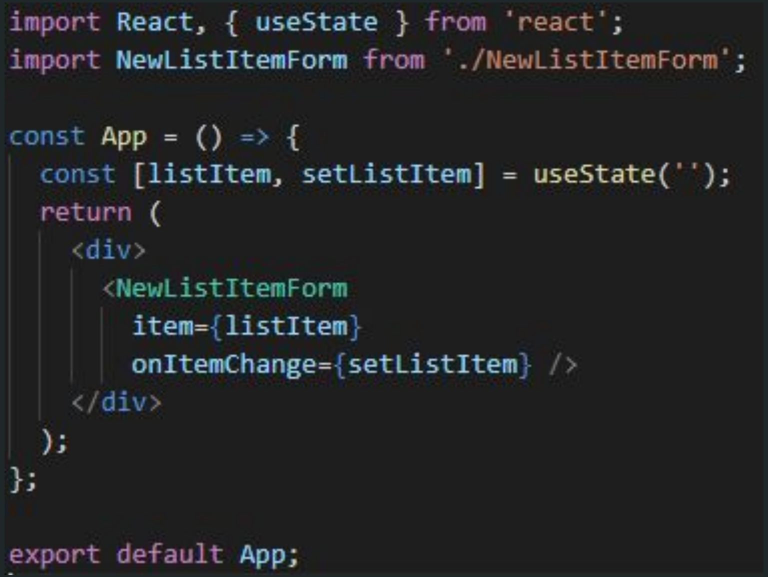
Re-run cypress test

Download axe - Check Accessibility
https://bit.ly/2rM2p9G

Or add cypress axe plugin
https://bit.ly/2rXV7zx

React - Lifting state up

https://reactjs.org/docs/lifting-state-up.html


```
import React, { useState } from 'react';
import NewListItemForm from './NewListItemForm';

const App = () => {
  const [listItem, setListItem] = useState('');
  return (
    <div>
      <NewListItemForm
        item={listItem}
        onItemChange={setListItem} />
    </div>
  );
};

export default App;
```

Refactor

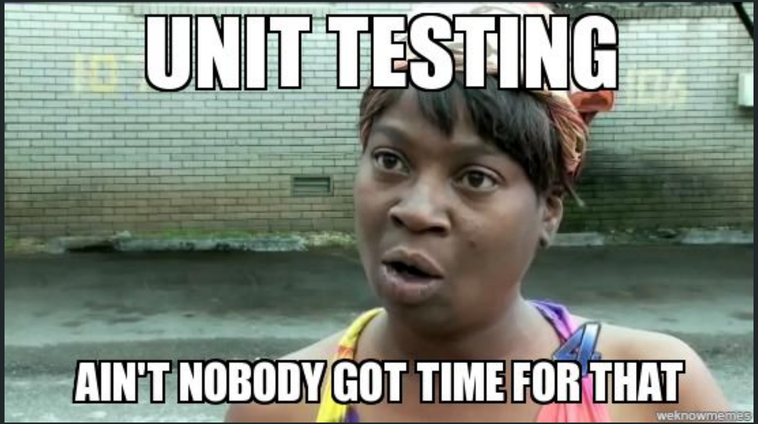
state > props

Props

```
props.listItem
```

```
props.onItemChange (event.target.value)
```

Refactor > Green



Mock

We need to mock the state now in our test

Now we have mocked the state change, we will now assert on our mock

```
const onItemChange = jest.fn()

render(<NewListItemForm item={''}
onItemChange={onItemChange} />)
```

https://jestjs.io/docs/en/expect#tohavebeencalled

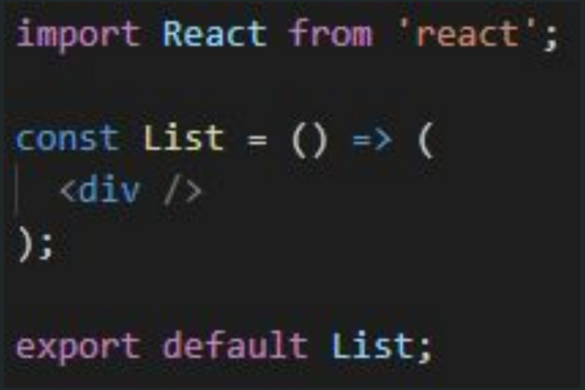
Don’t forget cypress tests

Task 4

Add cypress test for displaying the list

1. Visiting the web site
2. Entering the text “New message” into a message text field
3. Clicking a send button
4. Confirming that the “New message” we entered appears somewhere on screen

List.js

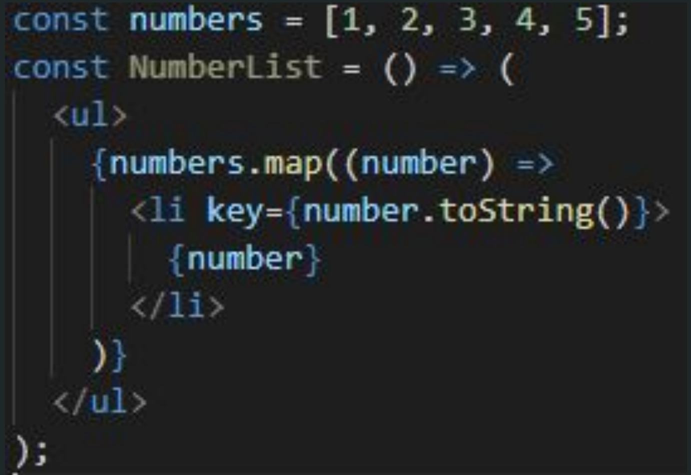


```
import React from 'react';

const List = () => (
  <div />
);

export default List;
```

Don’t forget to import in App.js

React - Unordered List

https://reactjs.org/docs/lists-and-keys.html#basic-list-component



```
const numbers = [1, 2, 3, 4, 5];
const NumberList = () => (
  <ul>
    {numbers.map((number) =>
      <li key={number.toString()}>
        {number}
      </li>
    )}
  </ul>
);
```

React - Lifting state up - pt 2

https://reactjs.org/docs/lifting-state-up.html

App.js

```
const [listItems, setListItems] =
useState([]);


<NewListItemForm
      item={listItem}
      onItemChange={setListItem}
      items={listItems}
      onItemsChange={setListItems}
/>
```

NewListItemForm.js

```
<form onSubmit={event => {
  props.onItemsChange([props.item,
...props.items])
  event.preventDefault()
  … }}
```

Task 5

Create jest test

list.spec.js

Add Visual Tests

https://bit.ly/35yxmwl

Re-run cypress test

Summary

- Understand ATDD & TDD ✓
- Understand testing within React App ✓
- Understand when to use Jest & when to use Cypress ✓
- Understand Cypress best practices e.g. selectors ✓