TESTING REACT COMPONENTS (MMT-M2020)

FHS (TMAYRHOFER.LBA@FH-SALZBURG.AC.AT)

RECAP HOMEWORK

RECAP HOMEWORK
MULTIPLE ASSERTIONS IN ONE TEST

```
it('score of a roll is stored into the score', () => {
    const game = new Game();
    game.roll(5);
    expect(game.score).toBe(5);
    game.roll('/');
    expect(game.score).toBe(10);
    game.roll('X');
    expect(game.score).toBe(30);
})
```

RECAP HOMEWORK
MULTIPLE ASSERTIONS IN ONE TEST

» usually a sign that a test is doing to much

» split into multiple tests

```
it('when 5 pins are hit, score is 5', () => {
    const game = new Game();
    game.roll(5);
    expect(game.score).toBe(5);
})

it('when a strike was hit, score is 10', () => {
    const game = new Game();
    game.roll('x');
    expect(game.score).toBe(10);
})

it('when a strike was followed by 5 pins, score is 20', () => {
    const game = new Game();
    game.roll('x');
    game.roll(5);
    expect(game.score).toBe(20);
})
```

RECAP HOMEWORK
MAGIC STRINGS

```
// ...
    if (item === '-') {
  //            ^^^
  // extract magic strings to constants and give them names
        return accumulated;
    }

    if (item === '/') {
        let nextThrow = getNextThrows(frames, index, i, 1)[0];
        if (nextThrow === '-')
            nextThrow = 0;
        else if (nextThrow === 'X')
            nextThrow = 10;
        else
            nextThrow = Number(nextThrow);
        return 10 + nextThrow;
    }
```

RECAP HOMEWORK
MAGIC STRINGS

» extract magic strings into constants

```
const MISS = '-'
const SPARE = '/'
const STRIKE = 'X'

// ...
    if (item === MISS) {
//              ^^^^
// replace magic characters with constants
        return accumulated;
    }
//...
```

RECAP HOMEWORK
NO LINTING

» next time -2 points

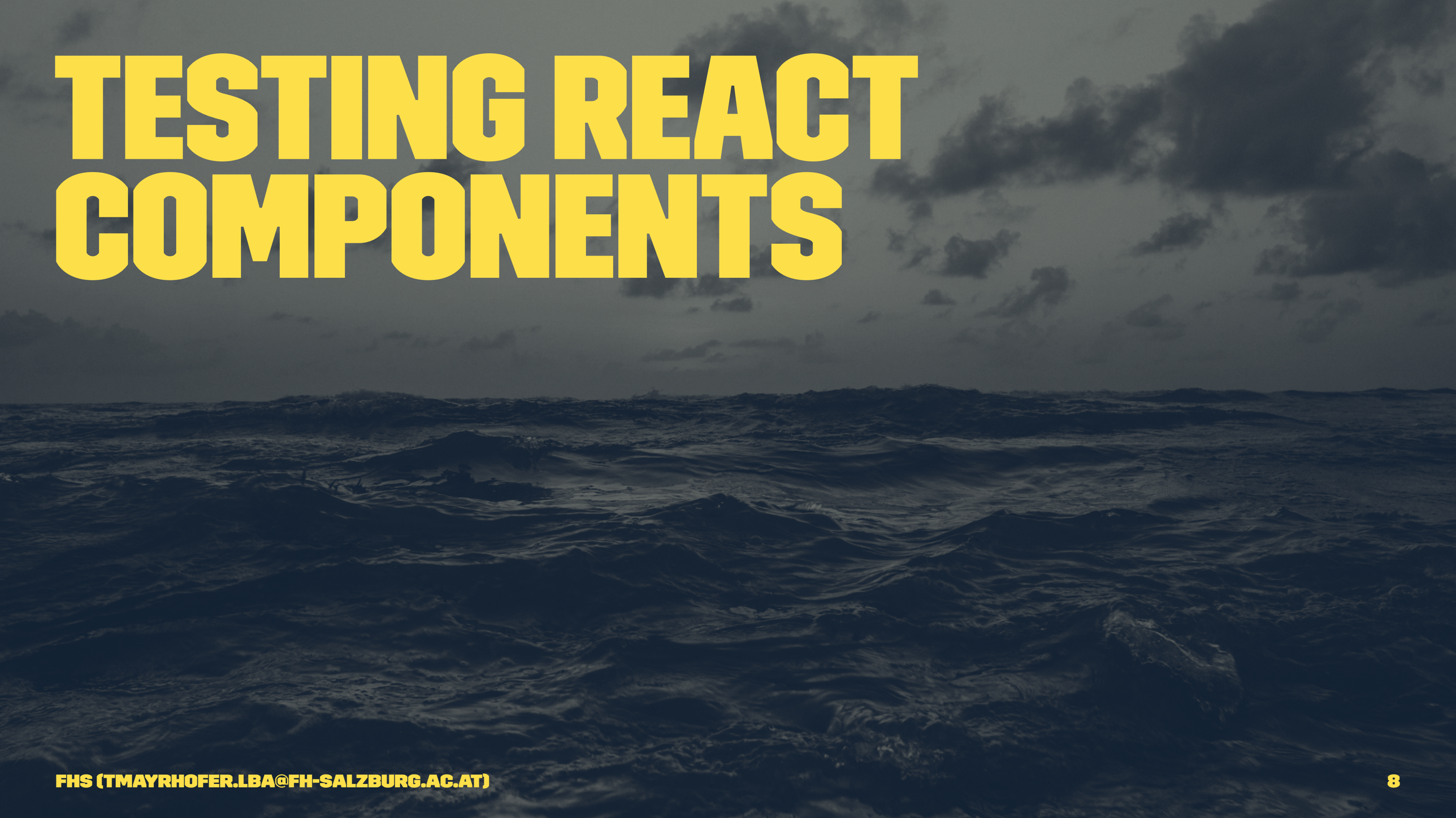
» please run

  » npx eslint --init

  » install linter

  » npx mrm lint-staged

  » automatically verify code before each commit


TESTING REACT
COMPONENTS

TESTING REACT COMPONENTS
TOOLS

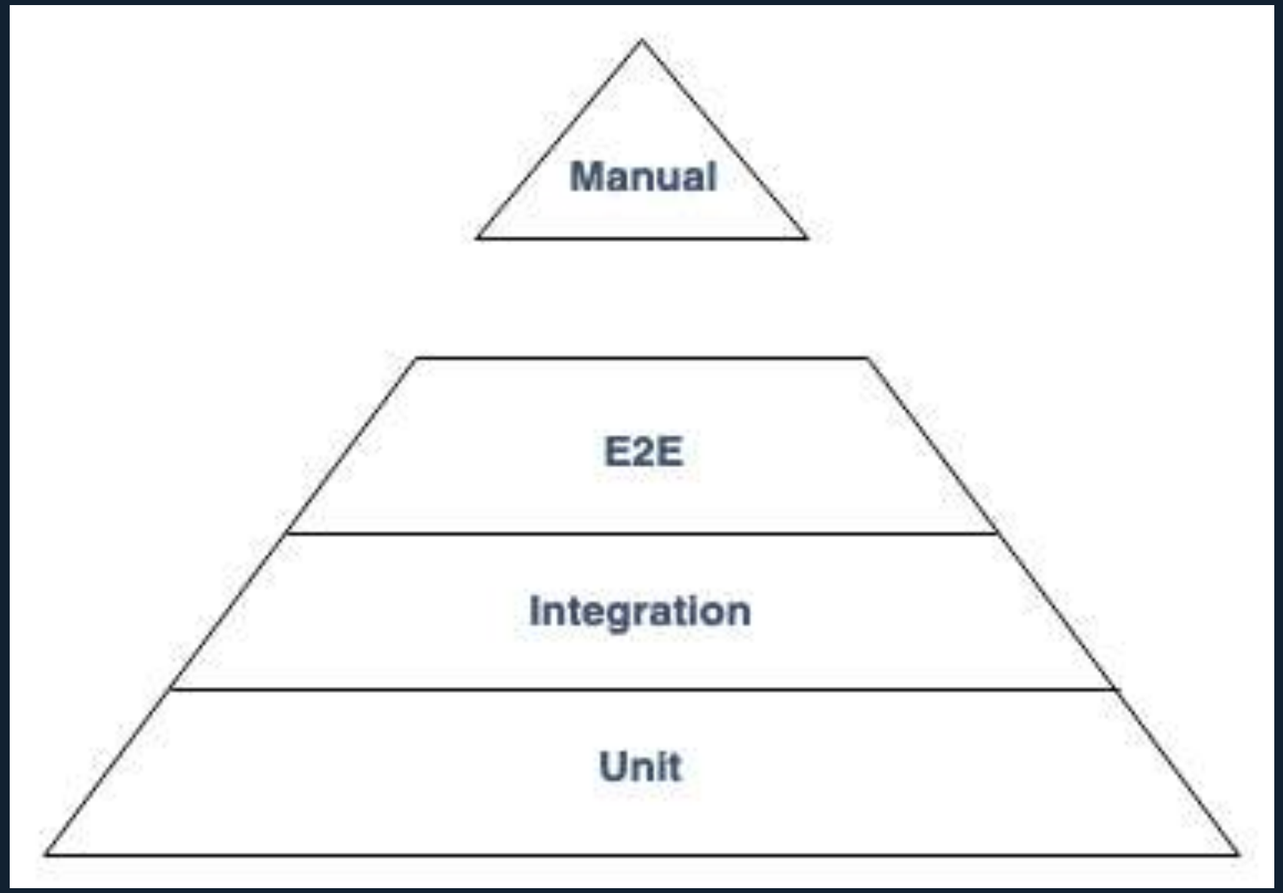
» Cypress (e2e testing "Software Quality Assurance")

» React Test Utils (Facebook, low level)

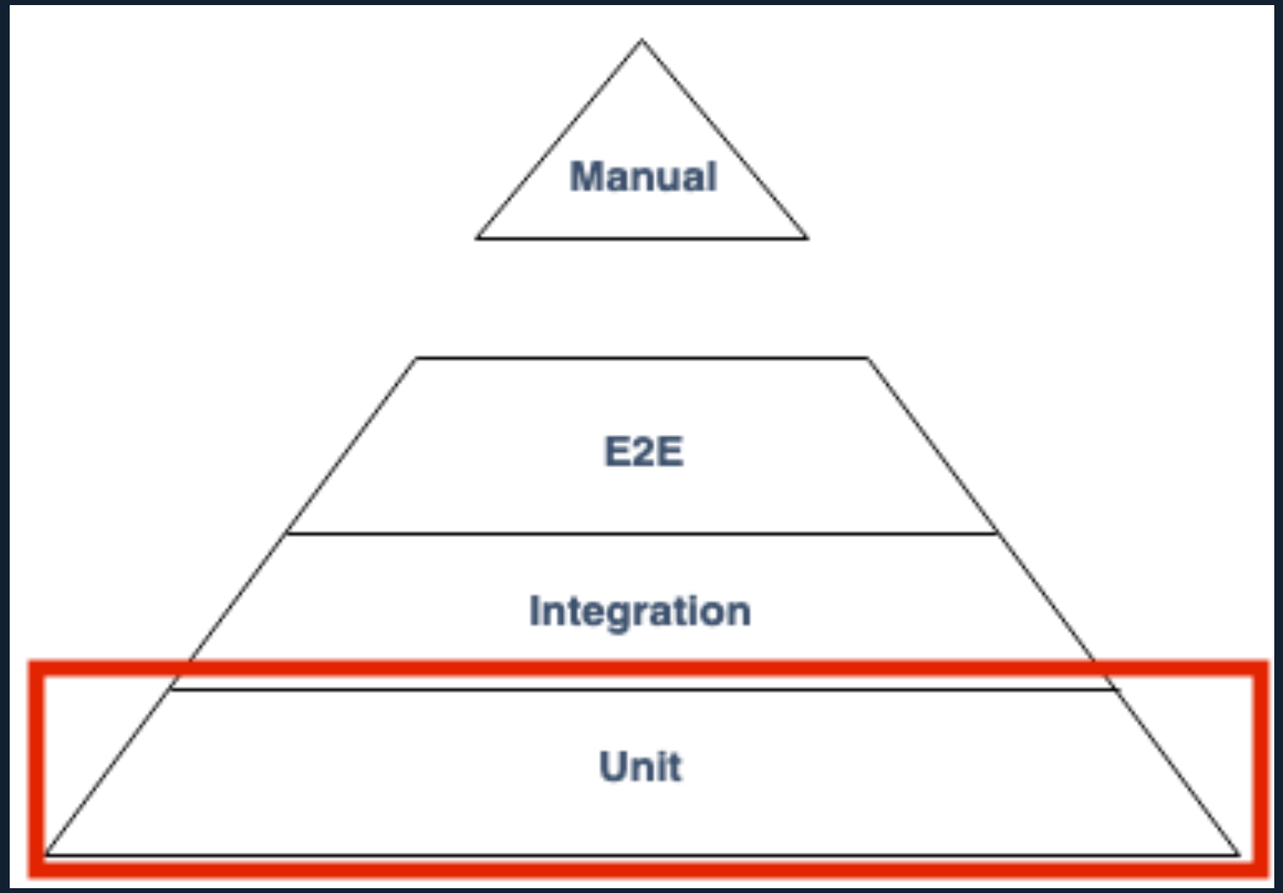
» Enzyme (AirBnB, more high level)

» React Testing Library (best of both worlds)

TESTING REACT COMPONENTS
TESTING PYRAMID



TESTING REACT COMPONENTS
TESTING PYRAMID



TESTING COMPONENTS
VERIFY THAT SOME TEXT IS RENDERED

```
// button.spec.js
import React from 'react'
import { render, cleanup, queryByText } from '@testing-library/react'
import { Button } from './Button'

afterEach(cleanup)
describe('Button', () => {
  it('renders given text', () => {
    // Arrange
    const givenText = 'A Button'

    // Act
    const { container } = render(<Button>{givenText}</Button>)

    // Assert
    expect(queryByText(container, givenText)).toBeTruthy()
  })
})
```

TESTING COMPONENTS
VERIFY THAT SOME TEXT IS RENDERED

» Run the test and watch it fail

» The test will guide you towards the implementation

TESTING COMPONENTS
VERIFY THAT SOME TEXT IS RENDERED

```
// button.js
import React from 'react'

const Button = ({ children }) => {
  return <>{children}</>
}
```

TESTING COMPONENTS
VERIFY THAT AN HTML TAG IS RENDERED

```
// button.spec.js
import React from 'react'
import { render, cleanup, queryByText } from '@testing-library/react'
import { Button } from './Button'

afterEach(cleanup)
describe('Button', () => {
  // ... other tests

  it('renders a button html tag', () => {
    const givenText = 'A Button'
    const { container } = render(<Button>{givenText}</Button>)

    expect(container.querySelectorAll('button'))
// 1)                 ^^^^^^^^^^^^^^^^^^
      .toHaveProperty("length", 1)
// 2)  ^^^^^^^^^^^^^^

// 1) get all button HTML tags from the rendered container
// 2) verify that there is exactly one button HTML tag
  })
})
```

TESTING COMPONENTS
VERIFY THAT AN HTML TAG IS RENDERED

```
import React from 'react'

export const Button = ({ children }) => (
  <button> { /* render a Button instead of a fragment */ }
    {children}
  </button>
)
```

TESTING USER INTERACTIONS

TESTING USER INTERACTIONS

» How can we test user interactions?

  » eg. when the button was clicked?

```
import React from 'react'

const Button = ({ onClick, children }) => {
  //                ^^^^^^^
  // How can we verify that the onClick handler is
  // called when the button is clicked?
  return <button>{children}</button>
}
```

TESTING USER INTERACTIONS
SPY OBJECTS

- Spy objects are stubs that also record the way they were called
- Useful when:
  - A hard to verify side effect is triggered (eg. E-Mail sending)

```
it('sends an email on sign up', () => {
  const sendEmail = jest.fn()
  const signUp = signUp({ sendEmail }, username, password)
  expect(sendEmail).toHaveBeenLastCalledWith({ username, password })
})
```

TESTING USER INTERACTIONS
SPY OBJECTS

» Spy objects can be used to verify user interactions in react components

```
// button.spec.js

  it('when button was clicked, calls given onClick handler', () => {
    const onClick = jest.fn() // create function spy
    const { container } = render(<Button onClick={onClick}>My Button</Button>)
    fireEvent.click(container.querySelector('button'))
// 1)                 ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
// 2)          ^^^^^
// 1) select the DOM element to be clicked
// 2) fire the click event

    expect(onClick).toHaveBeenCalled()
//  verify that the onClick spy has been called at least once
  })
```

TESTING USER INTERACTIONS
SPY OBJECTS

```
import React from 'react'

const Button = ({ onClick, children }) => (
  <button onClick={onClick}>
{/* 1)               ^^^^^^^   */}
    {children}
  </button>
)
// 1) set click handler on the button DOM element
```

EXERCISE 1

- You're building an issue tracking system
- Build a component which displays the names of assignees
  - eg. `<Assignees assignees={['Mike', 'Sepp', 'David']} />`
  - build a simple ul

EXERCISE 2

- WITH more than 3 assigness,
  - only display 3 assignees
  - display a show more button

EXERCISE 3

- WHEN the show more button was clicked
  - display all assignees
  - a show less button is displayed instead of a show more button
  - AND the show less button was clicked, only displays 3 assignees

EXERCISE 4

- WITH less than 4 assignees,
  - don't display a "show more" button

HOMEWORK

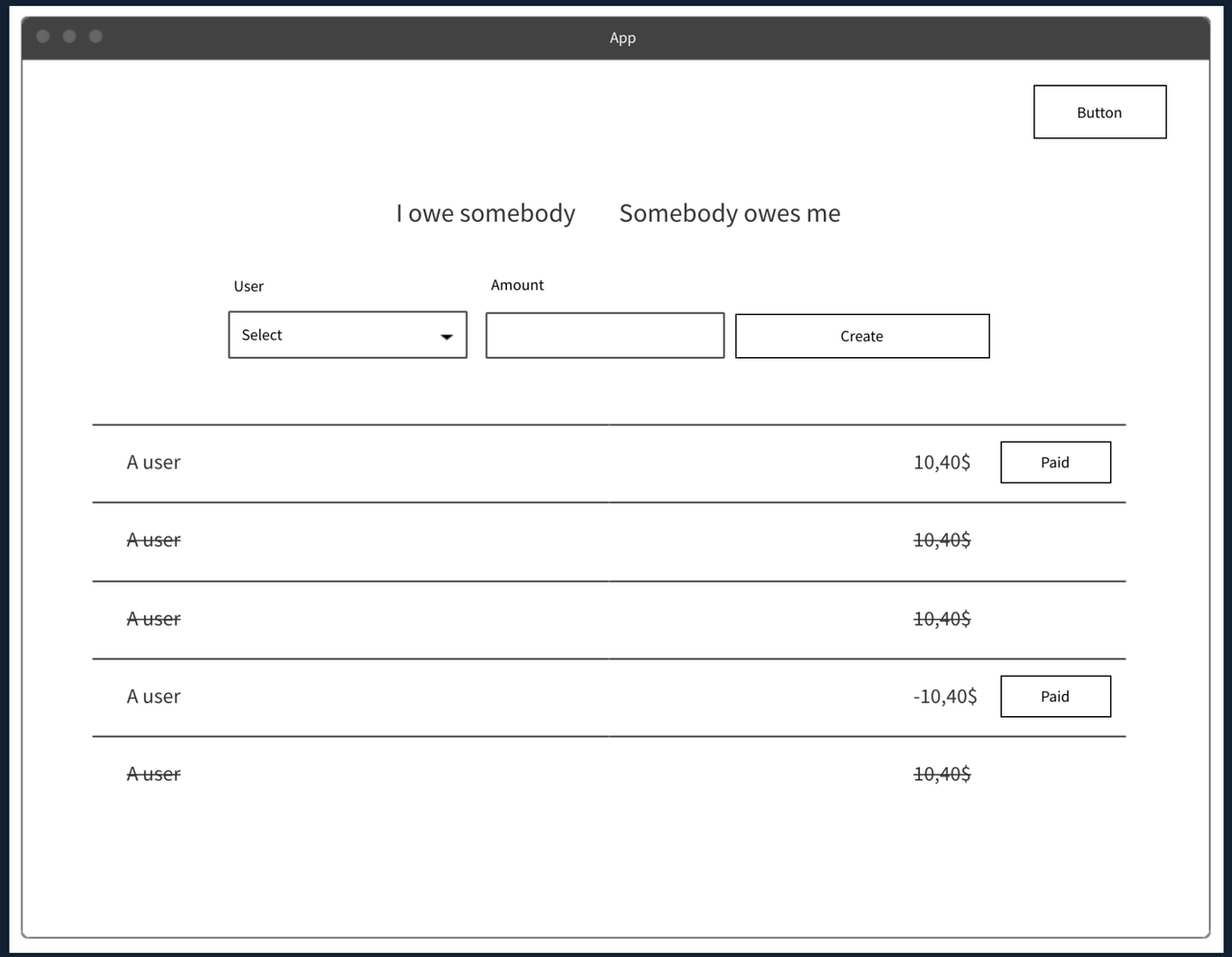
HOMEWORK 1

- Build 2 React Components using TDD
- Try to not open the browser during implementation
- Work in a strict TDD loop
  - commit before every new test
  - write descriptive commit messages
- You can use as a starting point https://github.com/webpapaya/fhs-react-redux-starter-kit

HOMEWORK 2

- MoneyTransactionCreate
  - users => { id, name }
  - onSubmit => { debitorId, creditorId, amount }
- MoneyTransactionList (Lists all Money Transactions)
  - moneyTransactions
  - onMoneyTransactionPaid => { id, paidAt: (new Date()).toISOSTring() }

HOMEWORK 3


App
Button
I owe somebody
Somebody owes me
User
Amount
Select
Create
A user
10,40$
Paid
~~A user~~
~~10,40$~~
~~A user~~
~~10,40$~~
A user
-10,40$
Paid
~~A user~~
~~10,40$~~

FEEDBACK

» Questions: tmayrhofer.lba@fh-salzburg.ac.at

» https://de.surveymonkey.com/r/8TW92LL