suora

Hans-Christian Otto

@muhdiekuh

Testing React Applications

suora

@muhdiekuh

Hans-Christian Otto

suora

Hans-Christian Otto

Testing React Applications

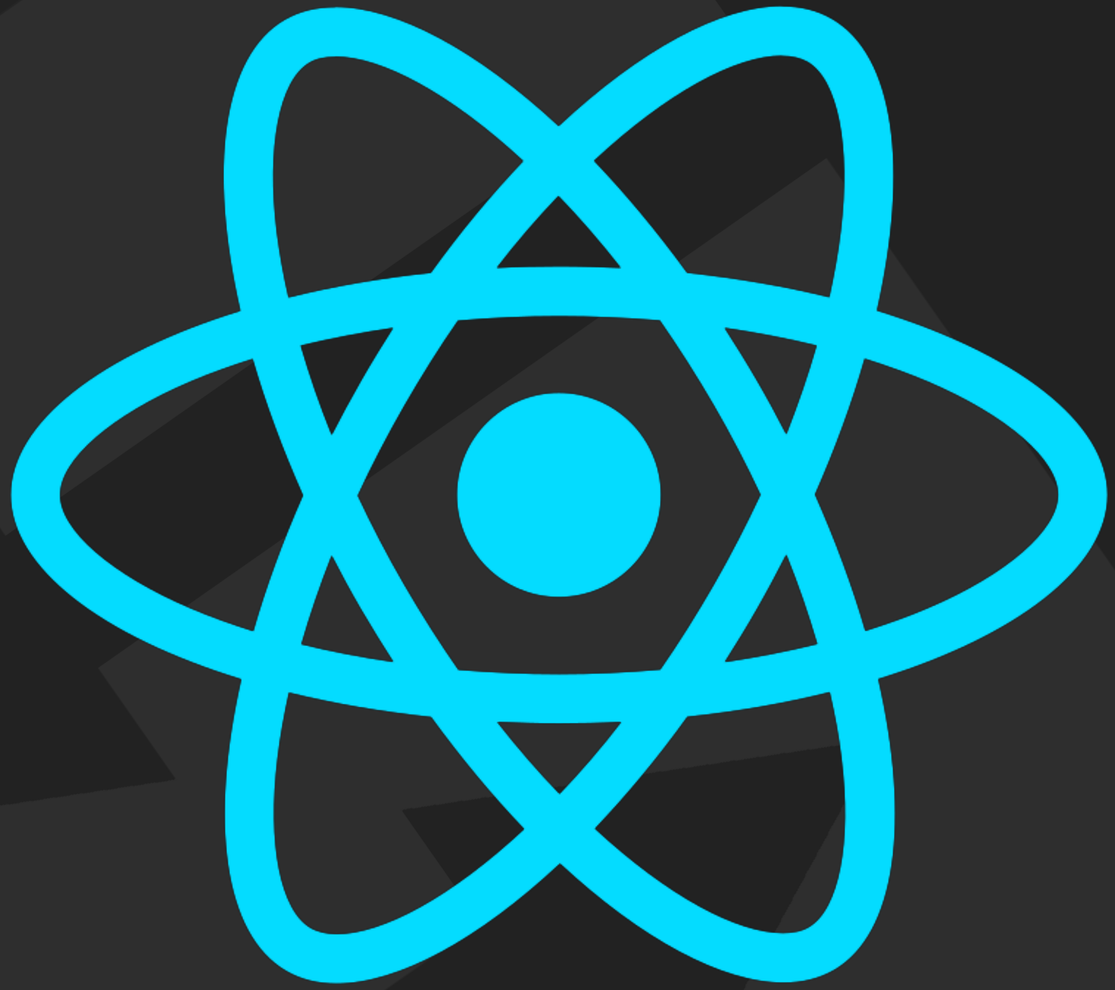
Photo by Will Francis on Unsplash

suora

Hans-Christian Otto

How to create nice react components?

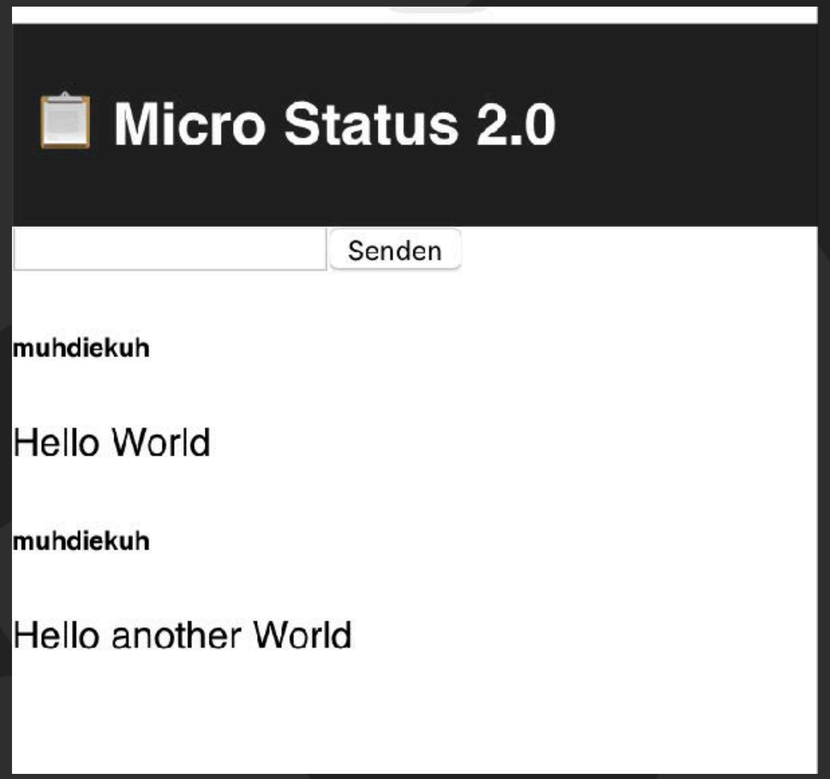
Photo by Will Francis on Unsplash



suora

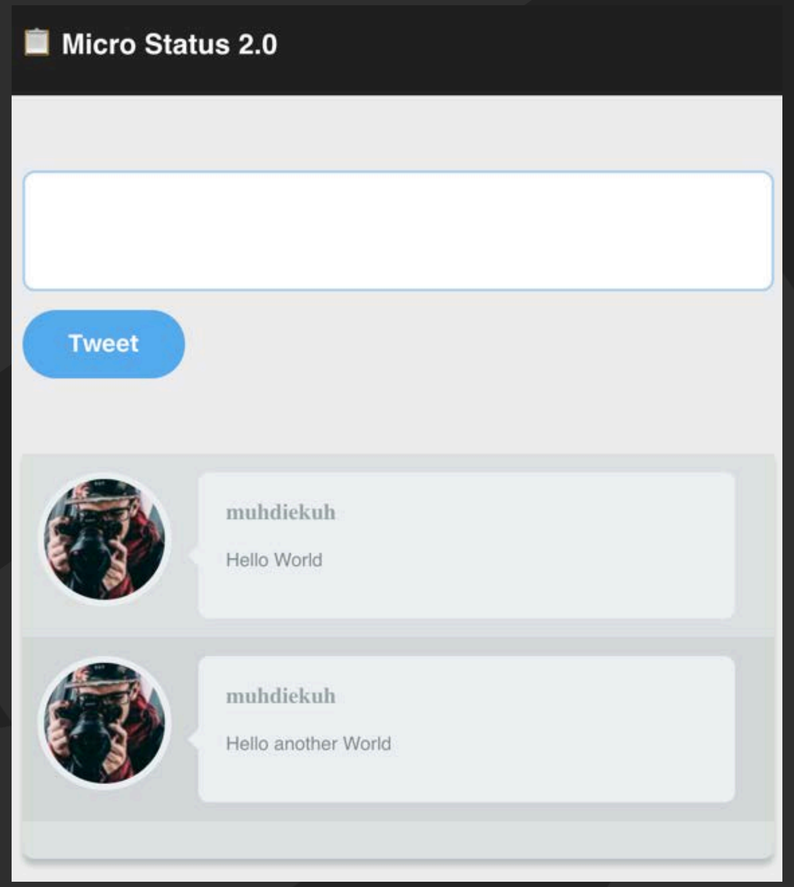
Let's create our very own Twitter

Photo by Marco Secchi on Unsplash


Micro Status 2.0
Senden
muhdiekuh
Hello World
muhdiekuh
Hello another World

Marcel made this look good.

Thanks, Marcel!



@MarcelFahle

suora

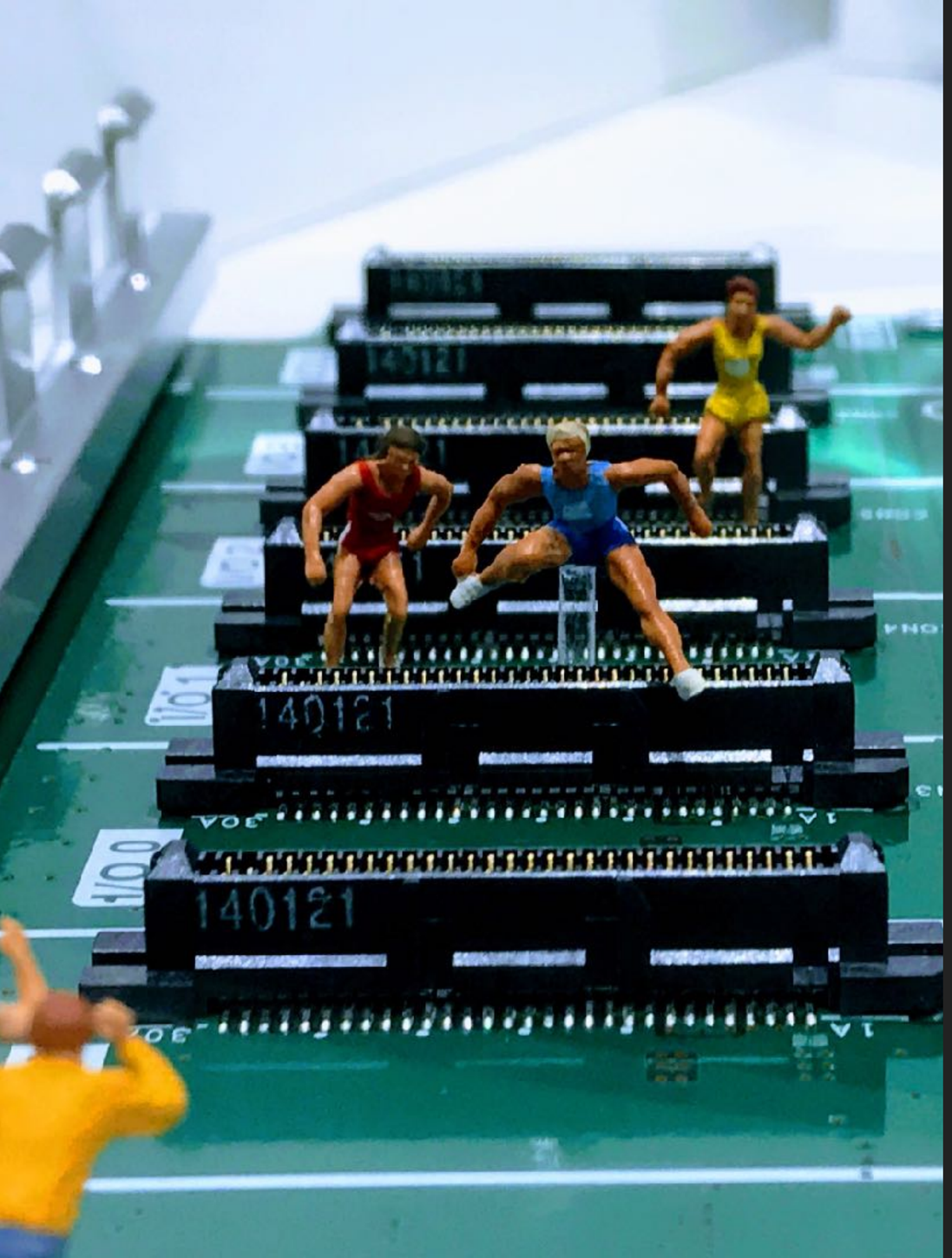
Let’s talk about testing.

suora

What is easy to test?

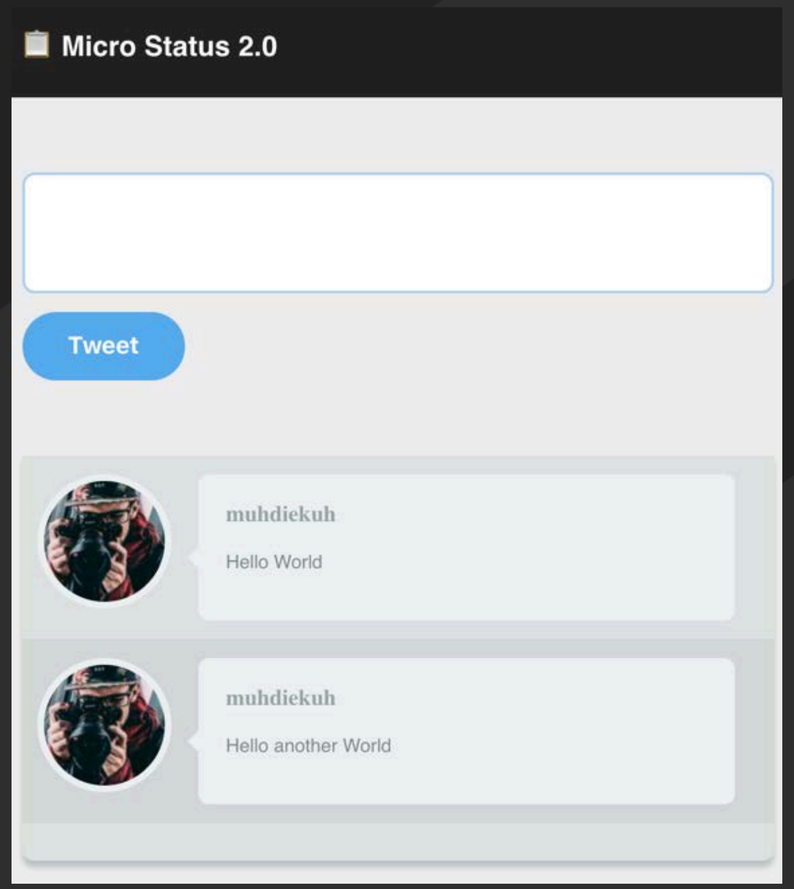
Photo by Simon Rae on Unsplash

Functions



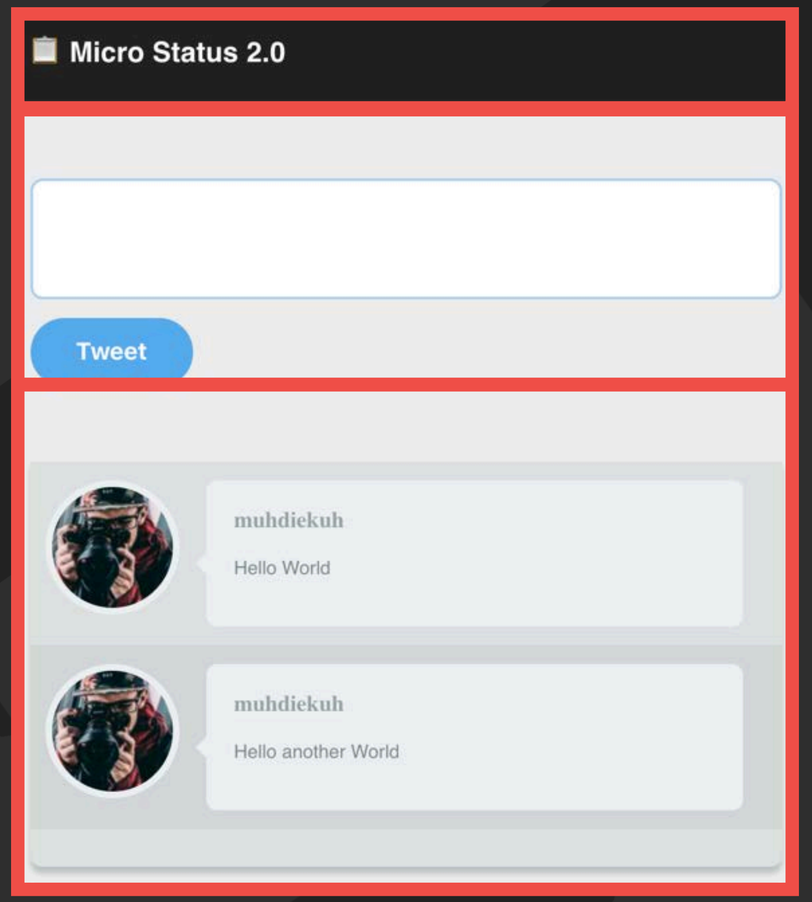
Functions that are:

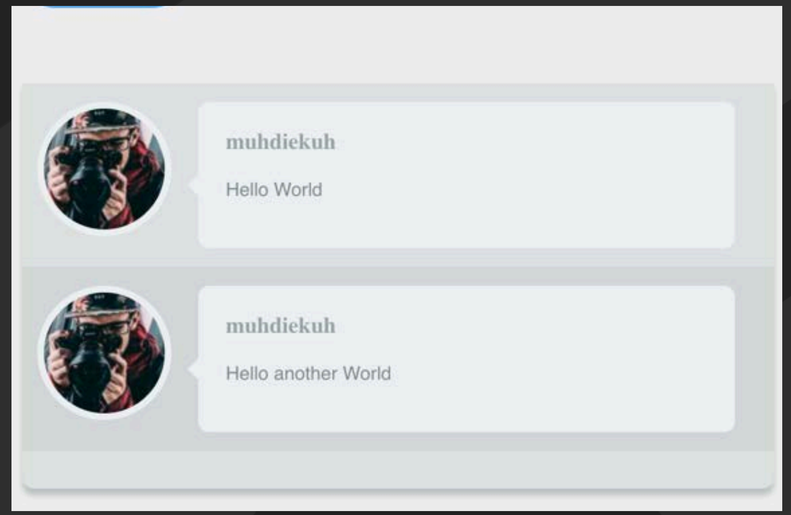
- ✓Small (Have a single responsibility)
- ✓Pure (Have no Side Effects)
- ✓Deterministic



- 😔Small (Have a single responsibility)
- 😔Pure (Have no Side Effects)
- 😔Deterministic

Let’s split it up!

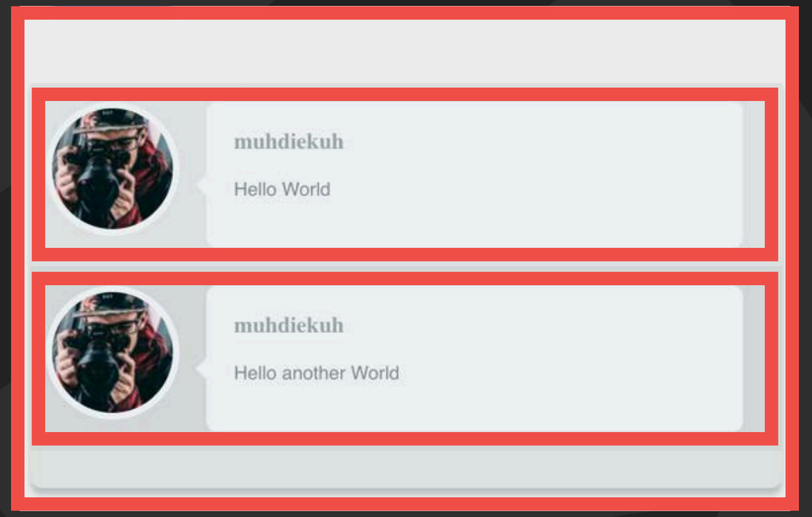

Micro Status 2.0
Tweet
muhdiekuh
Hello World
muhdiekuh
Hello another World

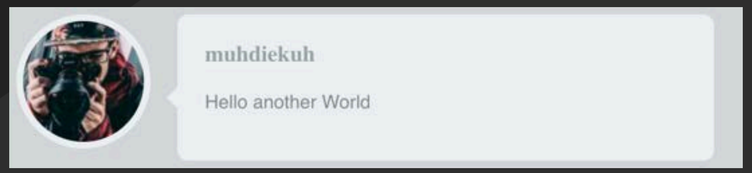


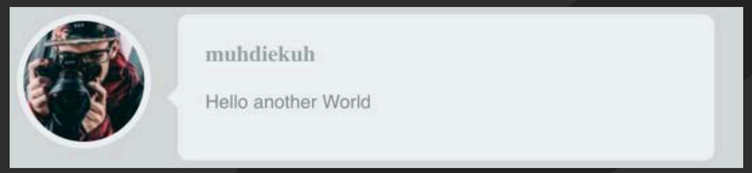
😐Small (Have a single responsibility)

😺Pure (Have no Side Effects)

😐Deterministic


muhdiekuh
Hello World
muhdiekuh
Hello another World


muhdiekuh
Hello another World



- 😺Small (Have a single responsibility)
- 😺Pure (Have no Side Effects)
- 😺Deterministic

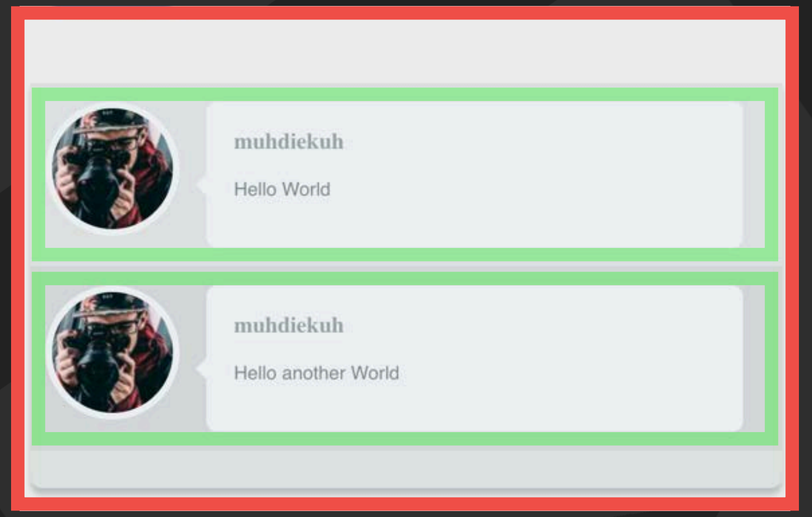

suora
Photo by Markus Spiske on Unsplash


```
<TweetListItem
  tweet={{
    message: "Hello World!",
    userId: "tobmaster",
    id: 3,
    date: "2018-09-24T09:16:52+0200"
  }}
/>;
```

```
<li>
    <div class="avatar">
        <img src="http://i.pravatar.cc/100?u=tobmaster" />
    </div>
    <div class="tweet-wrapper">
        <div class="tweet">
            <h6>tobmaster</h6>
            <p>Hello World</p>
        </div>
        <div class="arrow" />
    </div>
</li>
```

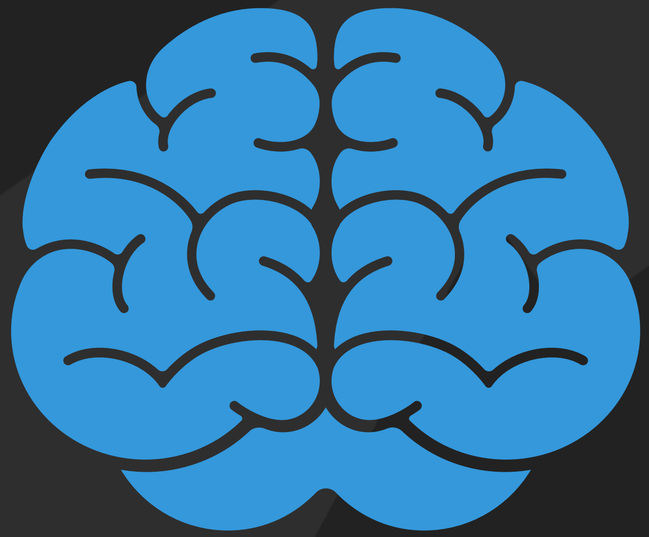
But...

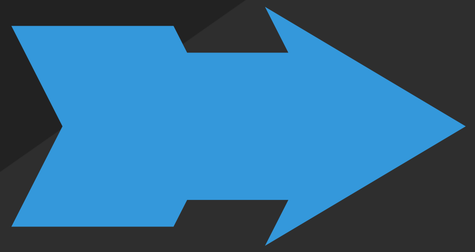
But…
What about the list?

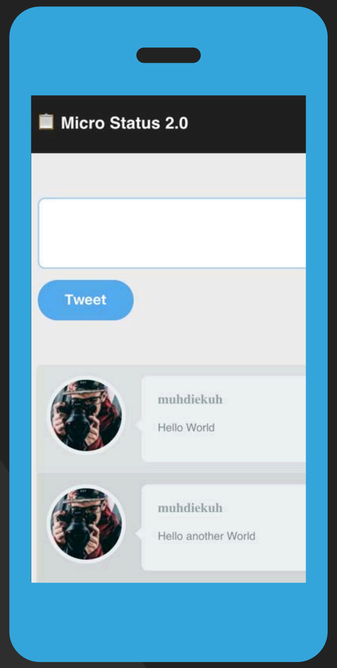

muhdiekuh
Hello World
muhdiekuh
Hello another World

Introducing:

components as a function of state







State

Component

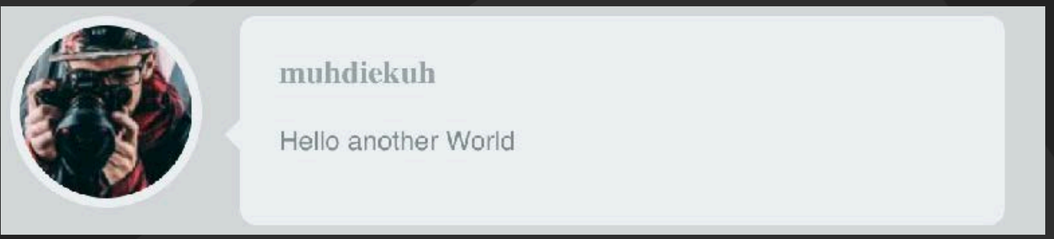
Application / UI

Presentational Components

Container Components


suora
Photo by Markus Spiske on Unsplash

suora



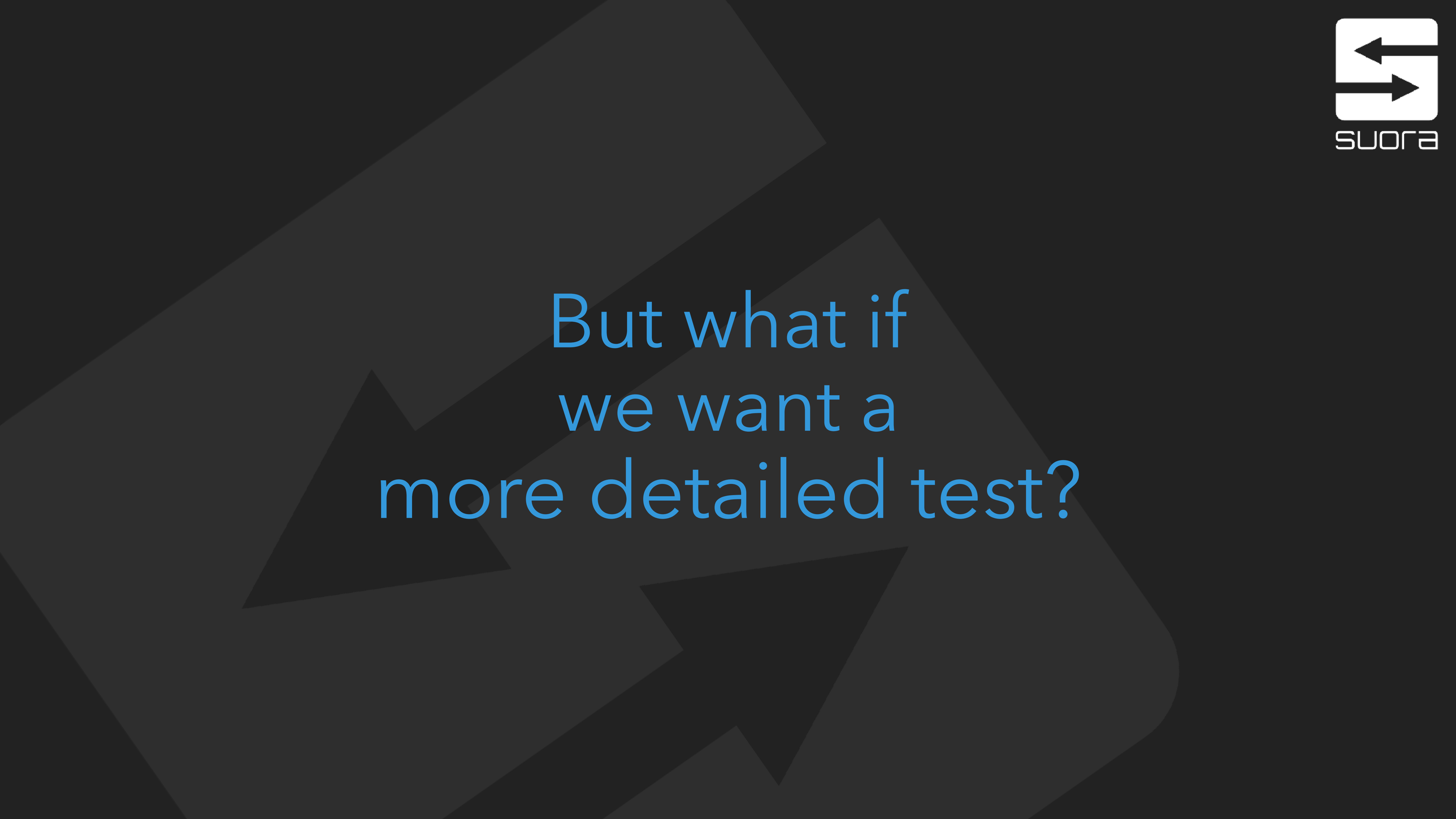
```
export class TweetListItem extends React.Component {
  public render() {
    const { tweet } = this.props;
    return (
      <li>
        <div className="avatar">
          <img src={`http://i.pravatar.cc/100?u=${tweet.userId}`} />
        </div>
        <div className="tweet-wrapper">
          <div className="tweet">
            <h6>{tweet.userId}</h6>
            <p>{tweet.message}</p>
          </div>
          <div className="arrow" />
        </div>
      </li>
    );
  }
}
```


```
 9 const tweet: Tweet = {
10   id: 1,
11   date: '2018-09-23T12:25:41.801Z',
12   message: 'Hello World!',
13   userId: 'muhdiekuh',
14 };
```


```
  2 import * as ReactDOM from 'react-dom';
[…]
 16 it('renders without crashing', () => {
 17   const div = document.createElement('div');
 18   ReactDOM.render(<TweetListItem tweet={tweet} />, div);
 19 });
```


```
21 it('renders and has a child', () => {
22   const div = document.createElement('div');
23   ReactDOM.render(<TweetListItem tweet={tweet} />, div);
24
25   const header =
div.childNodes[0].childNodes[1].childNodes[0].childNodes[0];
26
27   expect(header.nodeName).toEqual('H6');
28   expect(header.textContent).toEqual('muhdiekuh');
29 });
```


```
31 it('renders and somewhere has the authors name', () => {
32   const div = document.createElement('div');
33   ReactDOM.render(<TweetListItem tweet={tweet} />, div);
34   expect(div.innerHTML).toContain('muhdiekuh');
35 });
```



But what if
we want a
more detailed test?

Snapshot Tests


```
 3 import * as renderer from ,react-test-renderer';
[…]
37 it('renders according to snapshot', () => {
38   const tree =
renderer.create(<TweetListItem tweet={tweet} />).toJSON();
39   expect(tree).toMatchSnapshot();
40 });
```

src/component/__snapshots__/
TweetListItem.test.tsx.snap

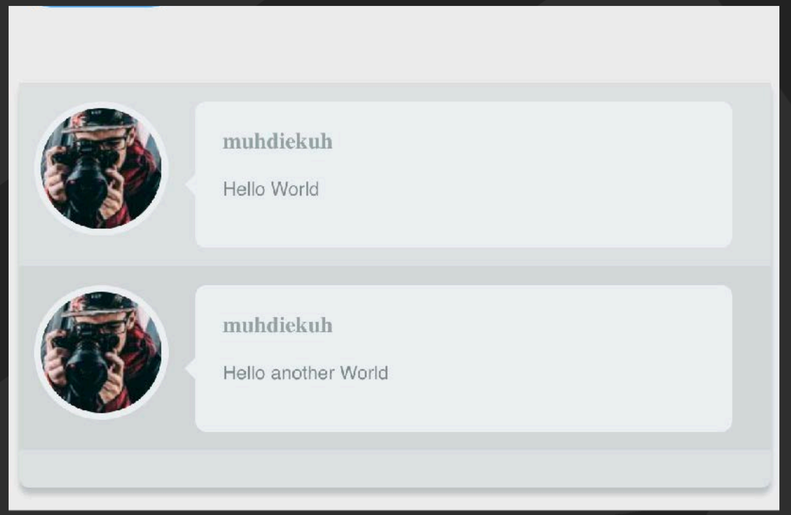

// Jest Snapshot v1, https://goo.gl/fbAQLP
exports[`renders according to snapshot 1`] = `

<div
className="avatar"
>
<img
src="http://i.pravatar.cc/100?u=muhdiekuh"
/>

<div
className="tweet-wrapper"
>
<div
className="tweet"
>
<h6>
muhdiekuh
</h6>

Hello World!

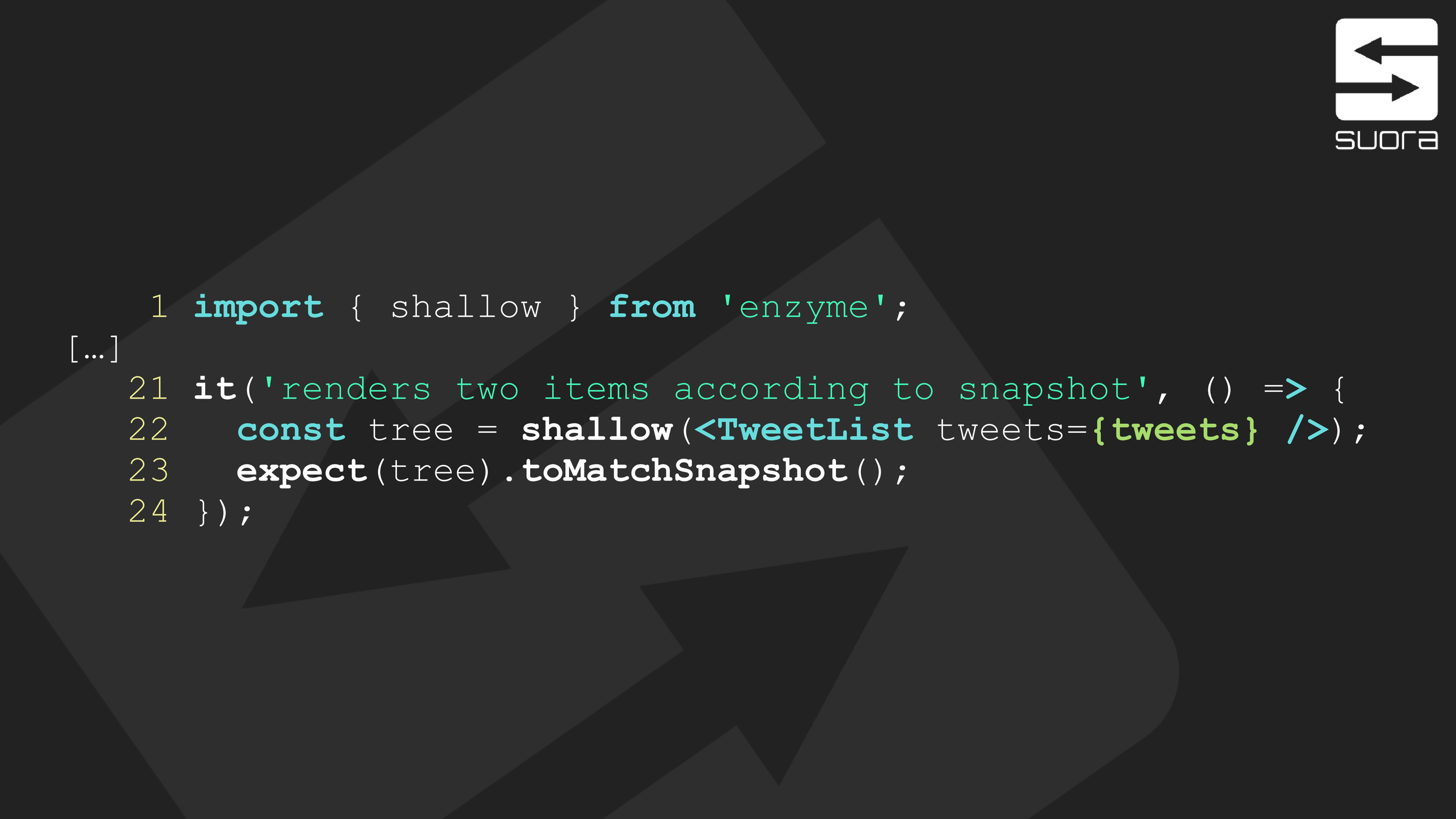
<div
suora


muhdiekuh
Hello World
muhdiekuh
Hello another World


```
 9 export class TweetList extends React.Component {
10   public render() {
11     const { tweets } = this.props;
12     return (
13       <ul className="timeline">
14         {tweets.map(tweet => (
15           <TweetListItem key={tweet.id} tweet={tweet} />
16         ))}
17       </ul>
18     );
19   }
20 }
```

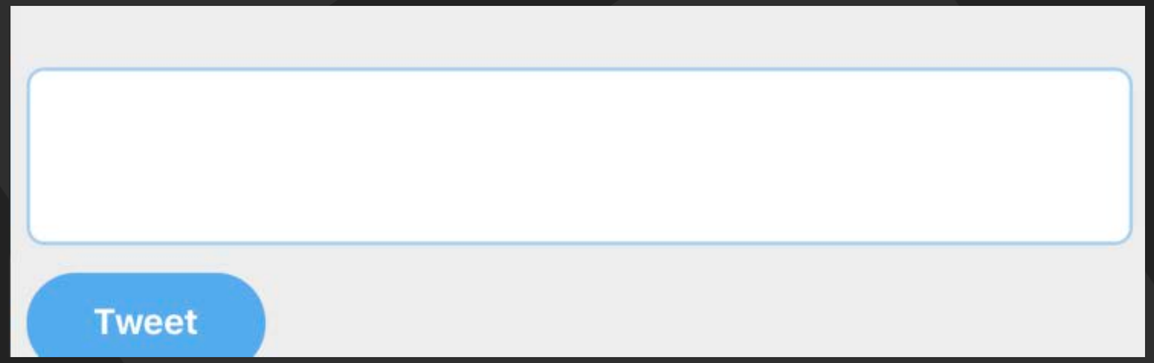
```
// Jest Snapshot v1, https://goo.gl/fbAQLP

exports[`renders multiple items according to snapshot 1`] = `
<ul
  className="timeline"
>
  <li>
    <div
      className="avatar"
    >
      <img
        src="http://i.pravatar.cc/100?u=muhdiekuh"
      />
    </div>
    <div
      className="tweet-wrapper"
    >
      <div
        className="tweet"
      >
        <h6>
          muhdiekuh
        </h6>
        <p>
          Hello World!
        </p>
      </div>
      <div
        className="arrow"
      />
    </div>
  </li>
  <li>
    <div
      className="avatar"
    >
      <img
```




```
  1 import { shallow } from 'enzyme';
[…]
 21 it('renders two items according to snapshot', () => {
 22   const tree = shallow(<TweetList tweets={tweets} />);
 23   expect(tree).toMatchSnapshot();
 24 });
```

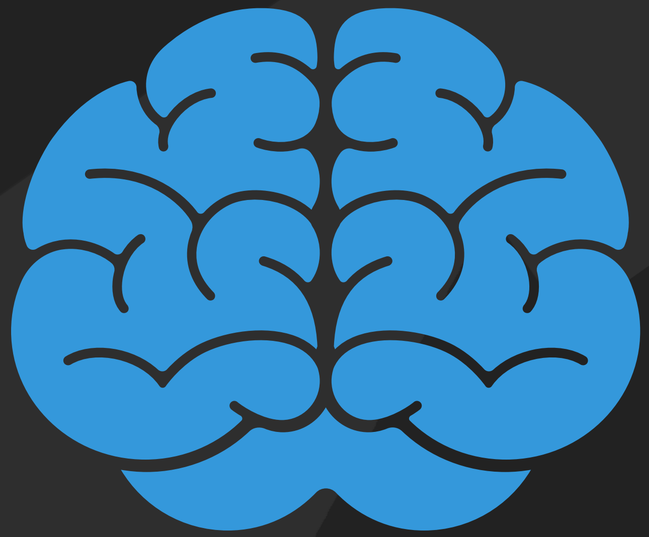
```
exports[`renders two items according to snapshot 1`] = `
<ul
  className="timeline"
>
  <TweetListItem
    key="1"
    tweet={
      Object {
        "date": "2018-09-22T12:25:41.801Z",
        "id": 1,
        "message": "Hello World!",
        "userId": "muhdiekuh",
      }
    }
  />
  <TweetListItem
    key="2"
    tweet={
      Object {
        "date": "2018-09-23T12:25:41.801Z",
        "id": 2,
        "message": "Hello from the other side!",
        "userId": "tobmaster",
      }
    }
  />
</ul>
`;
```

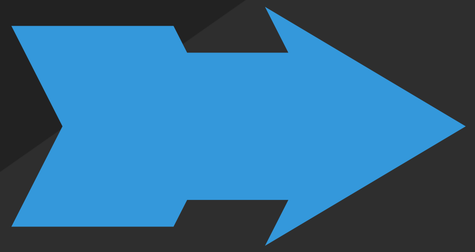

Tweet

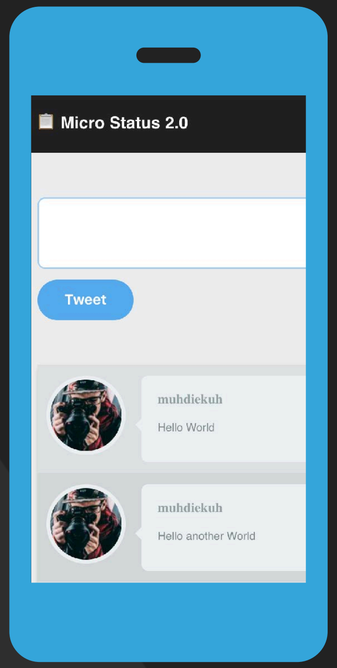

```
 1 import { mount } from 'enzyme';

14 it('contains a text after the user typed', () => {
15   const wrapper = mount(
16     <TweetWritingForm userId="muhdiekuh" addTweet={() => {}} />,
17   );
18   wrapper
19     .find('input[name="text"]')
20     .simulate('change', { target: { value: 'Hello World!' } });
21
22   // Ensure value is in the field
23   expect(wrapper).toMatchSnapshot();
24 });
25
```

```
26 it('submitting the form will result in a new tweet', () => {
27   const addTweetCallback = jest.fn();
28
29   const wrapper = mount(
30     <TweetWritingForm userId="muhdiekuh" addTweet={addTweetCallback} />,
31   );
32   wrapper
33     .find('input[name="text"]')
34     .simulate('change', { target: { value: 'Hello World!' } });
35
36   wrapper.find('button[type="submit"]').simulate('submit');
37
38   expect(addTweetCallback).toHaveBeenCalledTimes(1);
39   expect(addTweetCallback).toHaveBeenCalledWith(
40     expect.objectContaining({
41       userId: 'muhdiekuh',
42       message: 'Hello World!',
44     }),
45   );
50 });
```



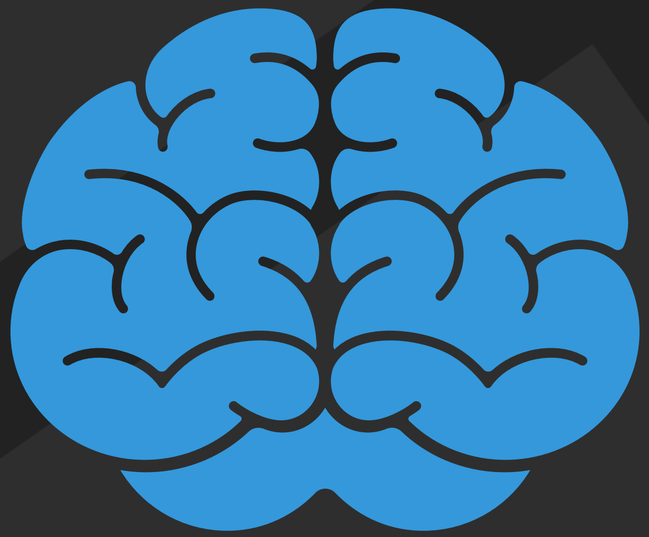




State

Component

Application / UI



Redux


```
12 export const fetchTweets = () => ({
13   type: FETCH_TWEETS,
14   payload: fetch('http://localhost:4712/tweets')
.then(response =>
15     response.json(),
16   ),
17 });
```


```
10 const mockStore = configureStore(middlewares);
11 describe('tweets action creators', () => {
14   fetch.resetMocks();
16
17   it('should execute fetch data', () => {
18     const store = mockStore({});
19     const tweet = { /* */ };
27     fetch.mockResponseOnce(JSON.stringify([tweet]));
28
30     return store.dispatch(fetchTweets()).then(() => {
31       const actions = store.getActions();
32       expect(actions[0]).toEqual({ type:
'FETCH_TWEETS_PENDING' });
33       expect(actions[1]).toEqual({
34         type: 'FETCH_TWEETS_FULFILLED',
35         payload: [tweet],
36       });
37     });
```

Esses poetas

Esse tem um plano, e já se aplaude.
Esse arde em memória, lembra e sopra.
Esse quer ter saudade, abre e morde.
Esse tem mistério, solta e fecha.
Esse faz humor, diz que desdiz.

Hooks


suora
And the real world?
Photo by Kyle Glenn on Unsplash


suora
Puppeteer-jest

storybook-storyshots

storyshots-puppeteer

react-testing-library

What's the take away?

Photo by Sandra Ahn Mode on Unsplash

Clean components are easy to test.


ASK MORE
QUESTIONS

https://github.com/SuoraGmbH/micro-status

suora

Thank you & have fun!

Hans-Christian Otto
Suora GmbH
c.otto@suora.com

@muhdiekuh
@SuoraGmbH
suora.com