Test like you mean it with React Testing Library

@chrishutchinson

Testing React today

- Enzyme
- React Test Utils
- Cypress
- React Testing Library

Testing React today

- ~~Enzyme~~
- ~~React Test Utils~~
- ~~Cypress~~
- **React Testing Library**

Why?

React Testing Library provides light utility functions, that encourage better testing practices.

Its primary guiding principle is:

The more your tests resemble the way your software is used, the more confidence they can give you.

Demo #1 - Text

```tsx
import { render } from "@testing-library/react";

const Component = () => <div>Hello, world!</div>;

it("should render the text 'Hello, world!'", () => {
  const { getByText } = render(<Component />);

  getByText("Hello, world!");
});
```

Let's start with a simple React component 👋

RUNS should render the text 'Hello, world!'

Demo #2 - Placeholder text

```
const Component = () => (
  <form>
    <input type="text" placeholder="Location" />
  </form>
);

it("should render an input with a 'Location' placeholder", () => {
  const { getByPlaceholderText } = render(<Component />);

  getByPlaceholderText("Location");
});
```

Now, time for an input box

RUNS should render an input with a 'Location' placeholder

Demo #3 - Roles

```
const Component = () => (
  <form>
    <input type="text" placeholder="Location" />
    <button>Use your current location</button>
  </form>
);

it("should render an element with the role 'button'", () => {
  const { getByRole } = render(<Component />);

  getByRole("button");
});
```

How about a button?

RUNS should render an element with the role 'button'

Demo #4.1 - Firing events

Here are some more features

RUNS should change the button text to 'Loading...' when clicked

Demo #4.2 - Firing events

Let's refactor to support additional loading status strings

RUNS should change the button text to 'Loading...' when clicked

Demo #5 - Changing inputs

```
const Component = () => {
  const [loadingState, setLoadingState] = React.useState("none");
  const [location, setLocation] = React.useState(null);

  const handleClick = () => {
    setLoadingState("pending");
  };

const handleLocationChange = e => {
  setLocation(e.currentTarget.value);
};

return (
  <main>
    <fieldset>
      {!isValidLocation(location) && <p>Error: The location is not valid</p>}

      <input
        type="text"
```

Now let's validate the text input

RUNS should render an error message if an invalid location is entered

Demo #6.1 - Async requests

```
const Component = () => {
  const [loadingState, setLoadingState] = React.useState("none");
  const [location, setLocation] = React.useState(null);

  const handleClick = React.useCallback(async () => {
    setLoadingState("pending");
    const currentLocation = await getCurrentLocation();

    setLocation(currentLocation);
  }, []);

  const handleLocationChange = React.useCallback(e => {
    setLocation(e.currentTarget.value);
  }, []);

  return (
    <main>
      <fieldset>
        {!isValidLocation(location) && <p>Error: ...</p>}
```

RUNS should render the async value into the input box

Demo #6.2 - Async requests

```
return (
  <main>
    <fieldset>
      {!isValidLocation(location) && <p>Error: ...</p>}

      <input
        type="text"
        placeholder="Location"
        value={location}
        onChange={handleLocationChange}
      />

      <button onClick={handleClick}>
        {loadingState === "pending"
          ? "Loading..."
          : "Use your current location"}
      </button>
```

We need to turn it into an async test

RUNS should render the async value into the input box

Demo #7 - Mocking

```
const Component = ({ onSubmit }) => {
  const [loadingState, setLoadingState] = React.useState("none");
  const [location, setLocation] = React.useState(null);

  const handleClick = React.useCallback(async () => {
    setLoadingState("pending");

    const currentLocation = await getCurrentLocation();

    setLocation(currentLocation);
  }, []);

  const handleLocationChange = React.useCallback(e => {
    setLocation(e.currentTarget.value);
  }, []);

  const handleSubmit = React.useCallback(
    e => {
      e.preventDefault();
      onSubmit(location);
    },
    [location]
  );
```

RUNS should call the onSubmit prop with...

Demo #8 - Routing

Here we're rendering some React Router routes

RUNS should render the form when the `/form` route is visited

There are more features?!

✅ Server-side render + hydration
✅ Unmounting behaviour
✅ Re-render on prop changes
✅ Test custom hooks via `@testing-library/react-hooks`
✅ Human-like events via `@testing-library/user-events`

And not forgetting...

✅ React Native via `@testing-library/react-native`

`RUNS` should get to the end of the talk

You'll find these slides along with working tests and documentation links at:

💻 github.com/chrishutchinson/talk-react-testing-library