02 **Component Tests: Rendering**

Testing React Apps

What is a component test?

React Testing Library

React Testing Library

`@testing-library/react`

testing-library.com/react

What to test?

- Call methods/functions?
- Check state values?

"We don’t encourage reading implementation details like state variables themselves…Instead, test observable behavior — i.e. what your component renders or does."

-- @dan_abramov, 2019-03-06 tweet

Testing the Contract

"A component contract is the agreement between a component and the rest of the application."

-- Edd Yerburgh, *Testing Vue.js Applications*

"Other components can assume the component will fulfill its contractual agreement and produce the agreed output if it’s provided the correct input."

-- Edd Yerburgh, ***Testing Vue.js Applications***

Question:

What are some of the kinds of inputs and outputs that components have?

Testing the Contract

Inputs:

- Props
- User interaction events

Outputs:

- Rendered UI
- Calls to external functions

Testing the Contract

Inputs:

- Props
- User interaction events

Outputs:

- **Rendered UI**
- Calls to external functions

To the code!

Getter Functions

- screen.getByText
- screen.getByAltText
- screen.getByTitle
- screen.getByLabelText

Text

```
return <Text>Hello, world!</Text>;
```

```
screen.getByText('Hello, world!')
```

Text Input: Label

```
return (
  <label>
    Enter your name
    <input type="text" ... />
  </label>
);

screen.getByLabelText('Enter your name') // gets the input
```

Image: Alt Text

```
return <img src={squirrel} alt="squirrel waving" />;
```

```
screen.getByAltText('squirrel waving')
```

SVG: Title

```
import {ReactComponent as WavingHand} from './waving-hand.svg';

return <WavingHand title="waving hand" />;

screen.getByTitle('waving hand')
```

Which Query Should I Use?

https://testing-library.com/docs/queries/about#priority

"your test should resemble how users interact with your code (component, page, etc.) as much as possible"

Which Query Should I Use?

- getByRole
- getByText
- getByDisplayValue
- getByPlaceholderText
- getByLabelText

Note on `getByRole()`

- It's the top recommendation
- Requires some learning about ARIA roles
- I recommend doing it

Confirming Presence

```
expect(screen.getBy…()).toBeVisible();
expect(screen.getBy…()).toBeInTheDocument();
```

Confirming Absence

Query Functions

Getter Functions

- screen.getByText
- screen.getByAltText
- screen.getByTitle
- screen.getByLabelText

Query Functions

- screen.**query**ByText
- screen.**query**ByAltText
- screen.**query**ByTitle
- screen.**query**ByLabelText

Confirming Absence

```
expect(screen.queryBy…()).not.toBeInTheDocument();
```

Testing the Contract

Inputs:

- Props
- User interaction events

Outputs:

- **Rendered UI**
- Calls to external functions

Testing the Contract

Inputs:

— Props

— User interaction events

Outputs:

✅ Rendered UI

- Calls to external functions

Testing the Contract

Inputs:

— **Props**

— User interaction events

Outputs:

✅ Rendered UI

- Calls to external functions

To the code!

Testing the Contract

Inputs:

— **Props**

— User interaction events

Outputs:

✅ Rendered UI

- Calls to external functions

Testing the Contract

Inputs:

✅ Props
- User interaction events

Outputs:

✅ Rendered UI
- Calls to external functions

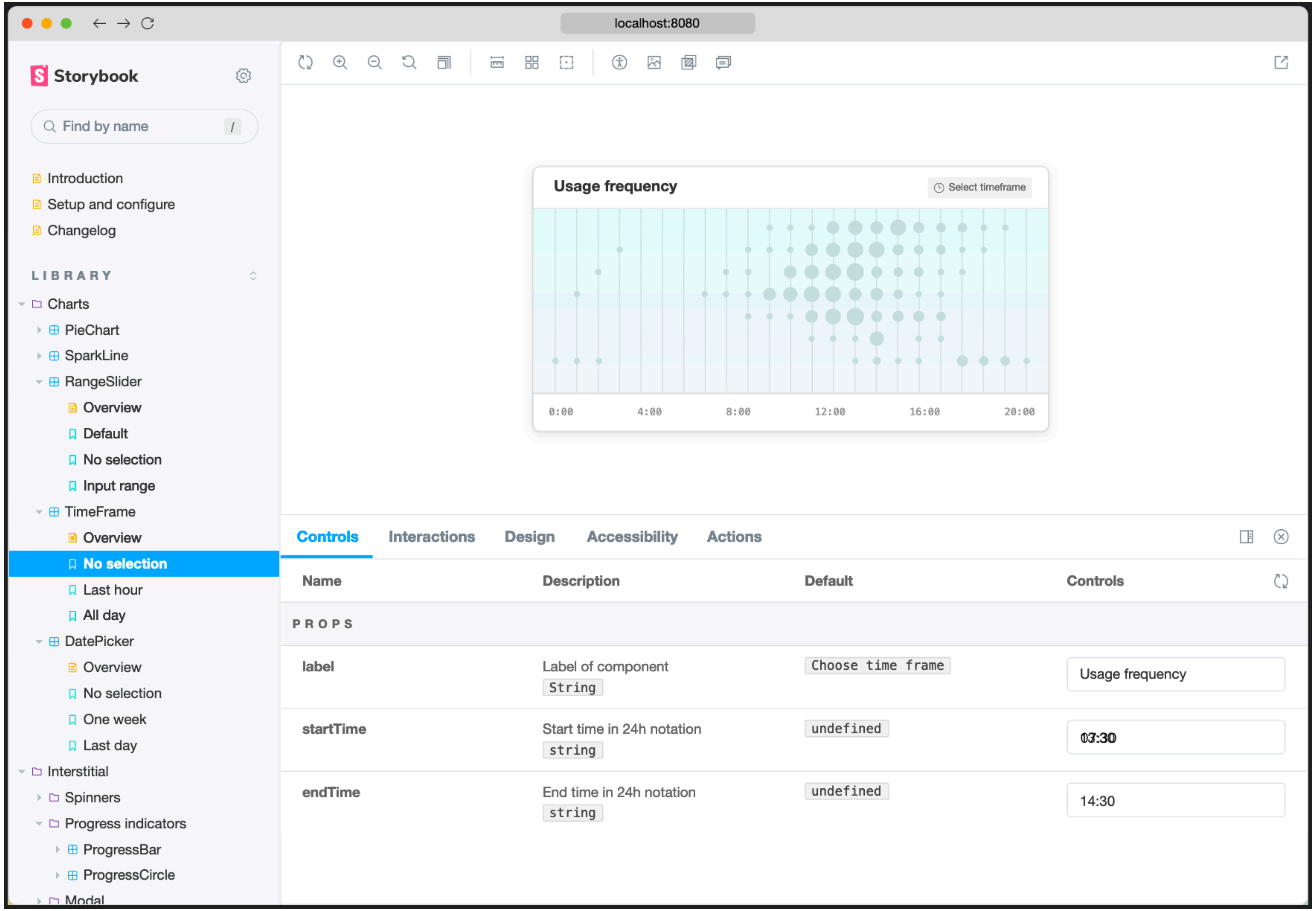
Exercise 1

- react-testing-exercises
- `exercises/Exercise-1.md`

For help:
testing-library.com/docs/queries/about

A few more notes about testing rendering and props

Testing Appearance



- Introduction
- Setup and configure
- Changelog

LIBRARY

- Charts
  - PieChart
  - SparkLine
  - RangeSlider
    - Overview
    - Default
    - No selection
    - Input range
  - TimeFrame
    - Overview
    - **No selection**
    - Last hour
    - All day
  - DatePicker
    - Overview
    - No selection
    - One week
    - Last day
- Interstitial
  - Spinners
  - Progress indicators
    - ProgressBar
    - ProgressCircle
  - Modal

Controls | Interactions | Design | Accessibility | Actions

Name	Description	Default	Controls
PROPS			
label	Label of component `String`	`Choose time frame`	Usage frequency
startTime	Start time in 24h notation `string`	`undefined`	03:30
endTime	End time in 24h notation `string`	`undefined`	14:30

What about snapshot tests?

```
import renderer from 'react-test-renderer';
import Hello from './Hello';

it('renders correctly', () => {
  const tree = renderer.create(<Hello name="Josh" />).toJSON();
  expect(tree).toMatchSnapshot();
});
```