Testing Component Events

Liam McLennan
PRODUCT DELIVERY CONSULTANT

@liammclennan https://withouttheloop.com

Overview

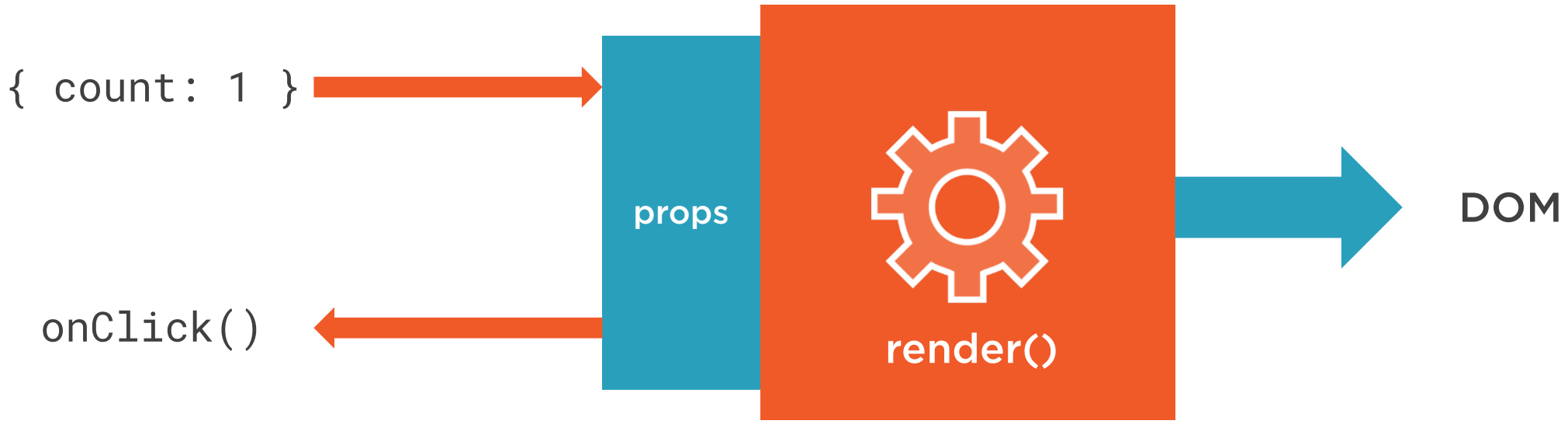
Testable components and test strategy

Testing events with React Testing Library

Testing events with mocking

Testing events with Test Renderer

Designing Components to Be Testable



Choosing a Testing Strategy

Testing Component Rendering

1. Set props and render
2. Make assertions against the output

Testing Component Events

1. Set props and render
2. Find the elements you need
3. Trigger events on elements
4. Make assertions against the events produced

Testing Component Events with React Testing Library

```
const r = render(<Counter count={0} />);

const button = r.getByText('OK');

fireEvent.click(button);

fireEvent(button, new MouseEvent('click', {
  bubbles: true,
  cancelable: true
});
```

Firing Events

Demo

Testing component events

```
it('should do something async', () => {

    return Promise.resolve(1 + 1)
        .then((v) => expect(2).toBe(v));

});
```

Testing Asynchronous Code with Jest

Test methods must return a promise

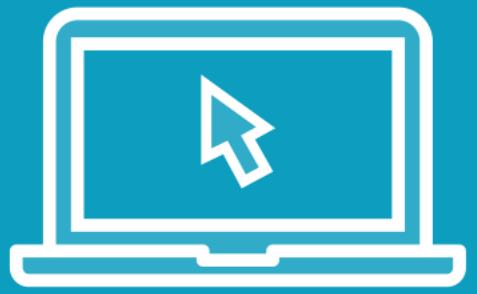
Testing Asynchronous Code with Jest

```
it('should …', () => {
  return Promise.resolve(1 + 1)
    .then((v) => expect(2).toBe(v));
});


it('should …', async () => {
  await Promise.resolve(1 + 1)
    .then((v) => expect(2).toBe(v));
});
```

Testing Asynchronous Code with Jest

```
it('should …', async () => {
  await Promise.resolve(1 + 1)
    .then((v) => expect(2).toBe(v));
});

it('should work for async functions', async () => {
  await expect(Promise.resolve(1 + 1)).resolves.toBe(3);
});
```

Demo



Testing asynchronous components

Testing Component Events with Mocking

Mocking Functions

```
const f = jest.fn();

f(1,2,3);
f('a');

expect(f.mock.calls).toEqual(
  [
    [1,2,3],
    ['a']
  ]
);
```

Mocking Functions with Implementation

```
const f = jest.fn((a,b) => a + b);

expect(f(4,8)).toBe(12);

expect(f.mock.calls).toEqual([
  [4,8]
]);

expect(f.mock.results).toEqual([{
  type: "return",
  value: 12
}]);
```

Mocking Functions with Fixed Results

```
const f = jest.fn();
f.mockReturnValueOnce(12).mockReturnValue(0);
expect(f()).toBe(12);
expect(f()).toBe(0);
expect(f()).toBe(0);
```

Demo

Testing the DateSlider component events

Testing Events with Test Renderer

```
const root = TestRenderer.create(<Counter />).root;
const p = root.findByType(“p”);
TestRenderer.act(() => { p.props.onClick(); });
expect(p.props.children).toEqual([1, ‘ ah ah ah’]);
```

API

Demo

Testing the DateSlider component events with Test Renderer

Test Renderer

```
const tr = TestRenderer.create(
  <Counter />);

const p = tr.root.findByType('p’);

TestRenderer.act(() =>
  { p.props.onClick(); });

expect(p.props.children).toEqual(
  [1, ' ah ah ah’]);
```

React Testing Library

```
const { queryByText } = render(
  <Counter />);

const p = queryByText(/ah ah ah/);

fireEvent.click(p);


expect(p.textContent).toBe(
  '1 ah ah ah’);
```

Summary

- Testing events often requires mocks
- Same testing strategy for different tools
- Testing library and test renderer are both viable