Chapter 3 – Advanced Events (React Testing Library)

Sunday On-Demand – Summary

In this chapter, we focus on **advanced user interactions** in React Testing Library, especially when dealing with forms, checkboxes, and popups. We also review important testing concepts and explain why **userEvent** is preferred over **fireEvent**.

Quick Refresher

When testing UI behavior, we should always think like a **real user**, not like the browser. That's exactly the philosophy behind React Testing Library.

```
test("checkbox is unchecked by default", async () => {
  const user = userEvent.setup();
  render(<SummaryForm />);

  const checkbox = screen.getByRole("checkbox", { name: /terms/i });

  expect(checkbox).not.toBeChecked();

  await user.click(checkbox);
  expect(checkbox).toBeChecked();

  await user.click(checkbox);
  expect(checkbox).not.toBeChecked();
});
```

Code Explanation

1. **userEvent.setup()**
    - Starts a user interaction session.

- It simulates **real user behavior** (mouse clicks, typing, focus, etc.).
- It returns a promise-based API, so interactions must be awaited.

2. **render(<SummaryForm />)**
   - Renders the component we want to test.
3. **screen.getByRole("checkbox", { name: /terms/i })**
   - Finds the checkbox the same way a user would: by its role and accessible name.
   - This improves accessibility and test reliability.
4. **Initial State Assertion**

   expect(checkbox).not.toBeChecked();
5. User Interaction

   await user.click(checkbox);
6. **State Change Assertions**

   After first click → checkbox should be checked.

   After second click → checkbox should return to unchecked.

Why We Replaced fireEvent with userEvent

fireEvent

- Simulates **browser events**, not real users.
- Triggers events instantly.
- Can lead to unrealistic tests.

userEvent

- Simulates **real user interactions**.
- Handles focus, keyboard, mouse, and delays properly.
- Encourages better testing practices.
- Matches how users actually use the app.

✅ **Conclusion:**
userEvent is closer to real-world usage and should be your default choice.

Screen Queries – Quick Summary

Type of Query	0 Matches	1 Match	>1 Matches	Retry (Async/Await)
Single Element				
`getBy...`	Throw error	Return element	Throw error	No
`queryBy...`	Return `null`	Return element	Throw error	No
`findBy...`	Throw error	Return element	Throw error	Yes
Multiple Elements				
`getAllBy...`	Throw error	Return array	Return array	No
`queryAllBy...`	Return `[]`	Return array	Return array	No
`findAllBy...`	Throw error	Return array	Return array	Yes

Advanced User Interaction Testing with React Testing Library

Example Test Code

```
test("checkbox is unchecked by default", async () -> {
  1 const user = userEvent.setup();
    render(<SummaryForm ());
  2 const checkbox = screen.getByRole("checkbox", { name: /terms/i })
  3 expect(checkbox),not toBeChecked();
  4 await user.click(checkbox);
  5 await user.click(checkbox); expect(checkbox).not toBeChed();
```

1. **Async Test**: user interaction takes time, so await/async is required.
2. **Create User**: Starts a "user session". Must begin with **userEvent.setup()**.
3. **Render the Form:** Render the form with **render**(<SummaryForm />).
4. **Find by Role**: Use an accessibility-first query to find the checkbox.
5. **Await user**.click(checkbox);

Example Test

Notes

- While **fireEvent** simulates browser-generated events, **userEvent** mimics real user interaction: clicks, typing, and more.
- **Important**: Always start by creating a user session with **userEvent.setup()**:
- All user interactions (e.g. user click(_)) are asynchr onous. Always use

UserActon Simulates real users:

fireEvent	userEvent
Simulates browser-generated events	Simulates realistic user actions
Immediate, simpler Less realistic	Clicks a box the way a user would
Not recommended	More accurate, async

Screen Query Refresher

- getByQueryType
- getAllByQueryType
- findByQueryType
- findAllByQueryType

Notes

- While **fireEvent** simulates browser-generated events,
- Prefer accessibility queries (e.g. getByRole)

React Testing Library Philosophy

- Don't test internals – test like a user
- Prefer accessibility queries (e.g. **getByRole**)