**userEvent** — это утилита из библиотеки @testing-library/user-event, которая предоставляет высокоуровневые методы для имитации взаимодействия пользователя с DOM-элементами (клик, ввод текста, наведение мыши, нажатия клавиш и так далее). Это более мощная альтернатива методам вроде fireEvent, так как она лучше эмулирует реальное поведение пользователя.

**Установка**

Если userEvent не установлен, его нужно добавить в проект:

npm install --save-dev @testing-library/user-event

**Основные методы userEvent**

import userEvent from '@testing-library/user-event';

**1. userEvent.click(element)**

Осуществляет клик на элементе.

test('клик по кнопке вызывает действие', () => {

  render( console.log('Вы кликнули!')}>Кнопка);

  const button = screen.getByRole('button', { name: /кнопка/i });

  userEvent.click(button);

  // Проверить что-то, связанное с кликом

});

**2. userEvent.type(element, text, options?)**

Эмулирует ввод текста в поле ввода. Метод поддерживает ввод по символу, как это было бы сделано вручную.

element\*\* — DOM-элемент для ввода текста

text — текст, который нужно ввести.

options (опционально): объект настроек, например { delay: 100 } для эмуляции задержки между вводом символов.

test('ввод текста в поле ввода', () => {

 render(<Компонент/>);

  const input = screen.getByPlaceholderText(/ваше имя/i);

  userEvent.type(input, 'Иван');

  expect(input).toHaveValue('Иван');

});

С имитацией задержки между символами:

await userEvent.type(input, 'Иван', { delay: 100 });

**3. userEvent.clear(element)**

Удаляет весь текст из элемента

Пример:

test('очистка поля ввода', () => {

 render(<Компонент/>);

  const input = screen.getByDisplayValue(/изначальный текст/i);

  userEvent.clear(input);

  expect(input).toHaveValue('');

});

**4. userEvent.hover(element)**

Эмулирует наведение курсора мыши на элемент.

Пример:

test('наведение показывает подсказку', async () => {

  render(<Компонент/>);

  const button = screen.getByRole('button', { name: /наведите/i });

  userEvent.hover(button);

  expect(screen.getByRole('tooltip')).toBeVisible();

});

**5. userEvent.unhover(element)**

Убирает курсор мыши с элемента, как будто пользователь больше не наведен.

**6. userEvent.selectOptions(select, values)**

Выбирает опцию/опции в элементе select.

select\*\* — элемент `` для взаимодействия.

values\*\* — значение (или массив значений) value для выбора.

test('выбор значения из списка', () => {

  render();

  const select = screen.getByRole('listbox');

  userEvent.selectOptions(select, ['1', '3']);

  expect(screen.getByText('Раз').selected).toBe(true);

  expect(screen.getByText('Два').selected).toBe(false);

  expect(screen.getByText('Три').selected).toBe(true);

});

**7. userEvent.keyboard(text)**

Эмулирует ввод текста с клавиатуры или нажатия клавиш.

text\*\* — строка, включающая символы или комбинации клавиш (например, '{enter}').

Пример:

test('ввод текста с клавиатуры', () => {

  render();

  const input = screen.getByRole('textbox');

  userEvent.keyboard('Привет!');

  expect(input).toHaveValue('Привет!');

});

test('эмуляция нажатия клавиши', () => {

  const handleKeyDown = jest.fn();

  render();

  const input = screen.getByRole('textbox');

  userEvent.keyboard('{enter}');

  expect(handleKeyDown).toHaveBeenCalled();

});

Заключение

userEvent – это мощный инструмент для тестирования приложений React и любых других интерфейсов, которые требуют эмуляции взаимодействия пользователя с DOM. Его методы помогают создавать более реалистичные сценарии тестирования, что делает ваши тесты ближе к поведению реального 