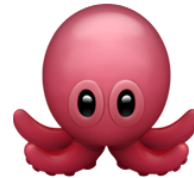Let's type something 🐙

```javascript
import { render, screen } from "@testing-library/react";
import userEvent from "@testing-library/user-event";
import { Form } from "../App";

test("Simulate keyboard typing with `userEvent`🎹", () => {
  const randomUser = {
    user: "fpetre@vairix.com",
    password: ":party-parrot:",
  };
  const handleSubmit = jest.fn();
  render(<Form handleSubmit={handleSubmit} />);
  const userInput = screen.getByLabelText(/user/i);
  const passwordInput = screen.getByLabelText(/password/i);
  const sendBtn = screen.getByRole("button", { name: /send/i });
  // // Now we'll use `userEvent.type` to fill the above inputs
  userEvent.type(userInput, randomUser.user);
  userEvent.type(passwordInput, randomUser.password);
  userEvent.click(sendBtn);
  expect(handleSubmit).toHaveBeenCalledWith(randomUser);
  expect(handleSubmit).toHaveBeenCalledTimes(1);
});
```
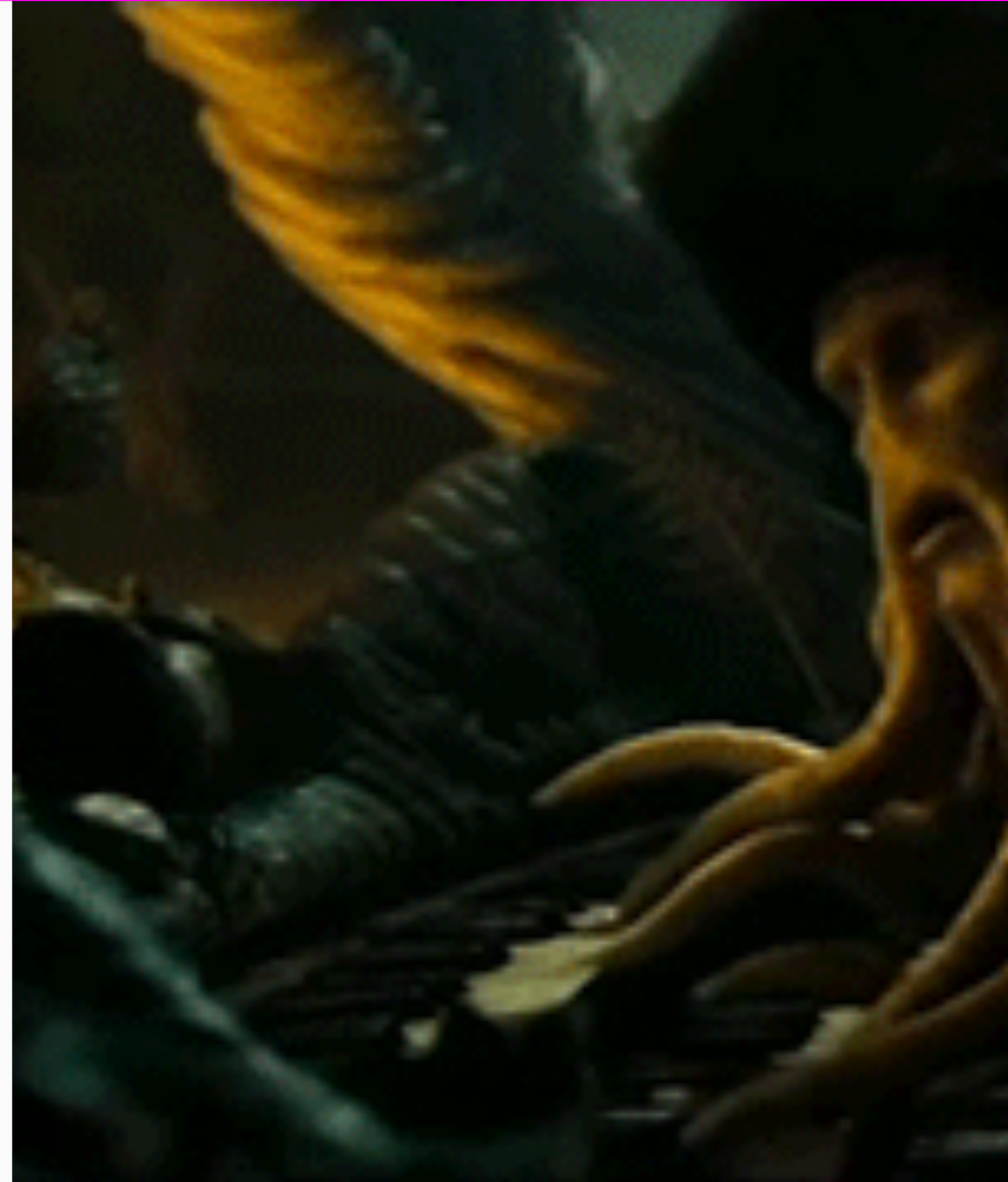
```javascript
import { render, screen } from "@testing-library/react";
import userEvent from "@testing-library/user-event";
import { Form } from "../App";

test("Simulate keyboard typing with `userEvent`🎹", () => {
  const randomUser = {
    user: "fpetre@vairix.com",
    password: ":party-parrot:",
  };
  const handleSubmit = jest.fn();
  render(<Form handleSubmit={handleSubmit} />);
  const userInput = screen.getByLabelText(/user/i);
  const passwordInput = screen.getByLabelText(/password/i);
  const sendBtn = screen.getByRole("button", { name: /send/i });
  // // Now we'll use `userEvent.type` to fill the above inputs
  userEvent.type(userInput, randomUser.user);
  userEvent.type(passwordInput, randomUser.password);
  userEvent.click(sendBtn);
  expect(handleSubmit).toHaveBeenCalledWith(randomUser);
  expect(handleSubmit).toHaveBeenCalledTimes(1);
});
```

HTTP mocking with MSW

(Happy Path)

```javascript
import { render, screen, waitForElementToBeRemoved } from "@testing-library/react";
import userEvent from "@testing-library/user-event";
import { rest } from "msw";
import { setupServer } from "msw/node";

const server = setupServer();
const rickAndMortyApi = `https://rickandmortyapi.com/api/character/1`;
const mockResponse = {
  name: "Facundo",
  species: "human",
  status: "low-battery",
};

beforeAll(() => {
  server.listen();
});
afterEach(() => {
  server.resetHandlers(); /* this prevents handlers conflicts */
});
afterAll(() => {
  server.close();
});

test("Mock successful request 🚀", async () => {
  server.use(
    rest.get(rickAndMortyApi, (req, res, ctx) => {
        return res(ctx.json(mockResponse));
      })
  );
  render(<App />);
  userEvent.click(screen.getByRole("button", { name: /get rick/i })); // 🖱
  await waitForElementToBeRemoved(() => screen.getByText(/loading/i)); // ⏳
  expect(screen.getByText(/name/i)).toHaveTextContent(mockResponse.name);
  expect(screen.getByText(/status/i)).toHaveTextContent(mockResponse.status);
  expect(screen.getByText(/species/i)).toHaveTextContent(
    mockResponse.species
  );
});
```