HTTP mocking with MSW

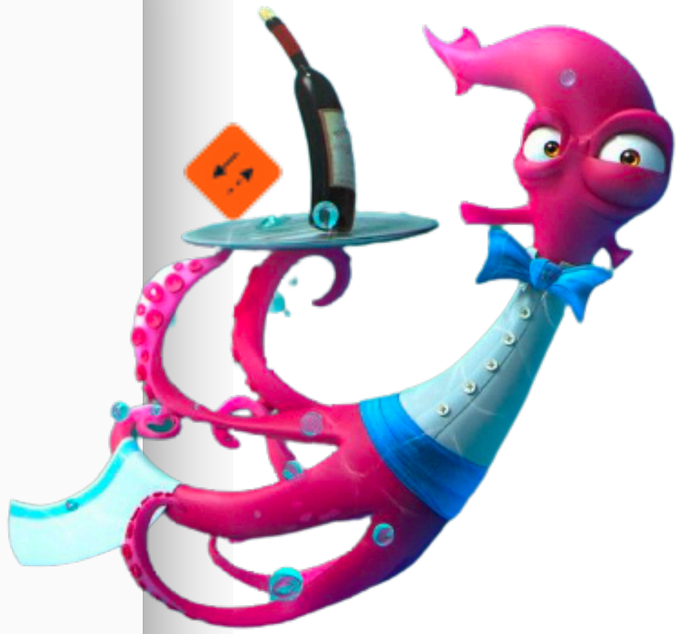
(Happy Path)

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

// here we go with the serverSetup
const server = setupServer();
const rickAndMortyApi = `https://rickandmortyapi.com/api/character/1`;

beforeAll(() => {
  server.listen();
});
afterEach(() => {
  server.resetHandlers();
});
afterAll(() => {
  server.close();
});

  test("Mock successful request 🚀", async () => {
    const mockResponse = {
      name: "Facundo",
      species: "human",
      status: "low-battery",
    };
    server.use(
      rest.get(rickAndMortyApi, (req, res, ctx) => {
        return res(ctx.json(mockResponse));
      })
    );
    render(<App />);
    await 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
    );
  });
```



HTTP mocking with MSW

(Unhappy Path)

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

// here we go with the serverSetup
const server = setupServer();
const rickAndMortyApi = `https://rickandmortyapi.com/api/character/1`;

beforeAll(() => {
  server.listen();
});
afterEach(() => {
  server.resetHandlers();
});
afterAll(() => {
  server.close();
});

test(`Mock request failure 🧨💥`, async () => {
    const message = `R.I.P 🪦`;
    server.use(
      rest.get(rickAndMortyApi, (req, res, ctx) => {
        // It's pretty same as before but setting an error status code
        return res(ctx.delay(1000), ctx.status(500), ctx.json({ message }));
      })
    );
    render(<App />);
    await userEvent.click(screen.getByRole("button", { name: /get rick/i }));
    await waitForElementToBeRemoved(() => screen.getByText(/loading/i), {
      timeout: 2000,
    });
    expect(screen.getByRole("alert")).toHaveTextContent(message);
  });
```