# HTTP mocking with MSW

(Unhappy 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 message = `R.I.P 🪦`;

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

test(`Mock request failure 🧨💥`, async () => {
  server.use(
    rest.get(rickAndMortyApi, (req, res, ctx) => {
      return res(ctx.delay(1000), ctx.status(500), ctx.json({ message }));
    })
  );
  render(<App />);
  userEvent.click(screen.getByRole("button", { name: /get rick/i }));
  await waitForElementToBeRemoved(() => screen.getByText(/loading/i), {
    timeout: 2000,
  });
  expect(screen.getByRole("alert")).toHaveTextContent(message);
});
```
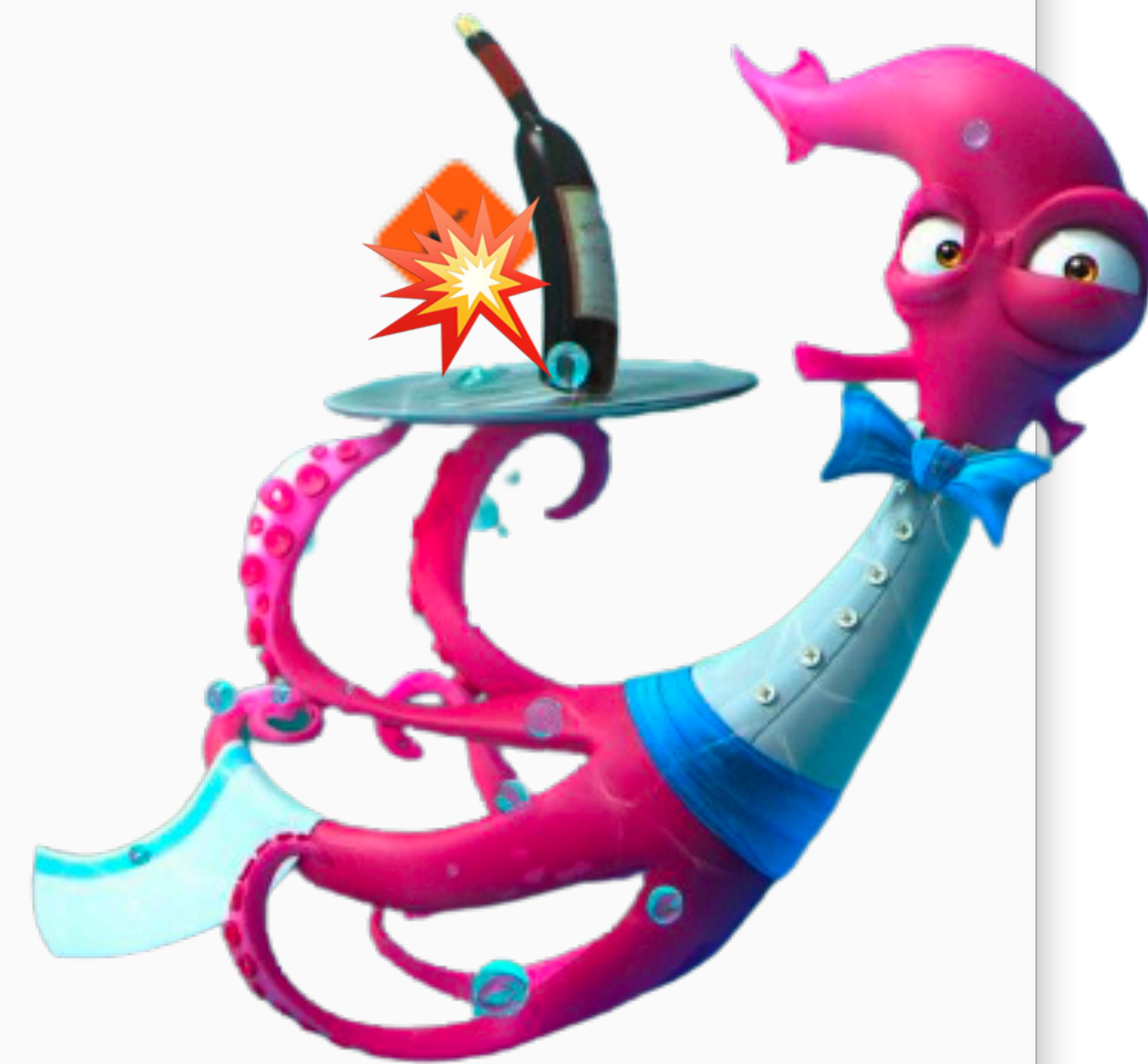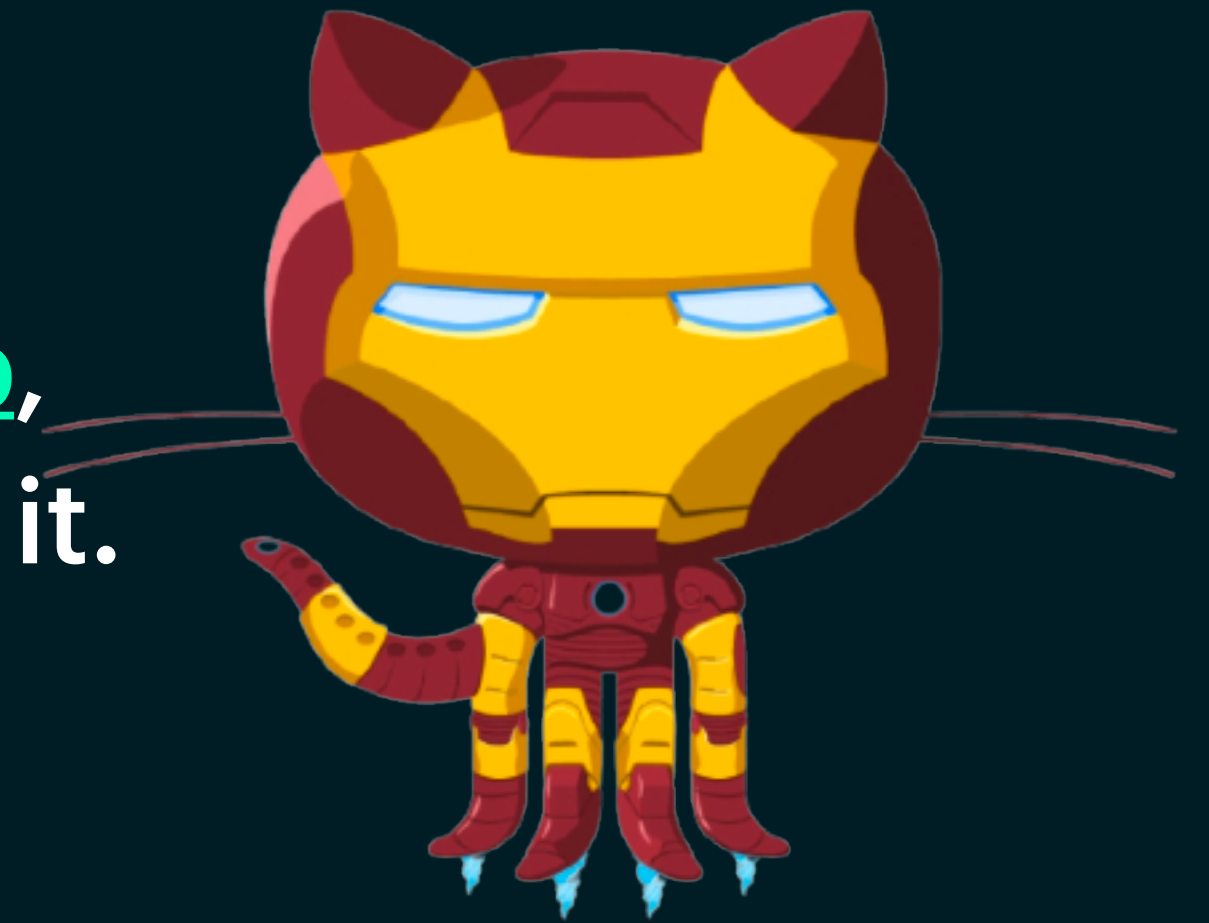
Ok, but show me the code

I created a sample repo on <u>my GitHub</u>,
it has way more deeper explanation on it.

Also, keep tuned, we'll be writing a **VAIRIX** blog on React-Testing

(Yup, we're hiring 😉)

fpetre@vairix.com