VUE.JS ENTERPRISE CRASH COURSE - PDF

LESSON 4

E2E TESTING

Lesson 4

E2E Testing

Welcome to the forth lessons of this crash course on enterprise web development with Vue.js.

As we've previous said, enterprise apps should be reliable and maintainable. A key way to achieve this is by adding tests to your app.

In the last lessons, we took a look at unit testing, so in this lesson, we'll look at the other important type of test which is end-to-end testing.

After an overview of end-to-end testing and how it works, we'll then finish with a disucssion about what the optimal blend of unit and end-to-end tests should be.

Unit tests

Testing small, atomic parts of the code in isolation e.g. functions, components.

E2E tests

Testing the entire application to check main use cases e.g. "find an item", "log in", etc.

So we've seen how unit tests allow us to test atomic parts of the application.

Differences

In contrast, end-to-end testing tests the complete, integrated application, including both front and backend app, running like they would in production.

Our aim in an end-to-end test is to ensure the main use cases run smoothly. For example, can the user login, or find an item, or add that item to their shopping cart?

To test a use case, we want to interact with the app like a user would.

For this reason, E2E tests get executed in a web browser. The test script will use a browser automation driver to programmatically click buttons, fill out forms, change the URL, and so on.

By default, Vue CLI will install Chrome drivers for E2E testing, but you can set up a variety of browsers based on your app's needs.

E2E test runners

```
$ vue add e2e-cypress

## or

$ vue add e2e-nightwatch
```

The two best known browser automation frameworks are Cypress.io and Nightwatch.js. Both of these are available as Vue CLI plugins that can easily be added to a project.

E2E test example

tests/e2e/specs/test.js

```
module.exports = {
  "should log in": (browser) {
    browser
      .url("https://localhost:8080/login")
      .waitForElementVisible("form#login")
      .setValue("input[name='email']", "test@test.com")
      .setValue("input[name='password']", "abcd1234")
      .click("button[type='submit']")
      .assert.urlEquals("https://localhost:8080")
  }
};
```

Here's a simple example of an E2E test using the Nightwatch framework

Test description

In this test, we're checking if the user can login.

URL

So first we instruct the browser to navigate to the login page.

Wait

Then we wait for the login form to be visible, remembering that it may take a few seconds for a Vue app to load and render.

Form

We then get the browser to fill out the form and click the submit button.

Assertion

And finally, we make an assertion that we've been re-directed to the homepage, which tells us the login was successful.

Unit vs E2E

Do we need both?

To finish this video, I want to address an important question: do we need both unit and E2E tests?

The best way to answer that is to firstly understand that both unit tests and E2E tests have unique strengths and weaknesses.

Unit test pros and cons

Pros

- Run fast
- Precise

Cons

- Time consuming to write
- Indirect correlation between tests passing and app working

Let's start with unit tests.

The pros are that they...

Fast

run fast, and

Precise

they're precise, allowing us to pin-point issues in the code right down to a specific function.

But the downsides of unit tests are that...

Time consuming

in large application particularly, it's time-consuming to write unit tests for every part of the code.

Narrow

And not only that, but even if the unit test suite passes, it doesn't tell us if the application actually works in the hands of users.

E2E test pros and cons

Pros

- Implicitly test many things
- Ensure whole system is working

Cons

- Slow to run
- Brittle
- Can’t pinpoint failure

What about E2E tests? What are their strengths and weaknesses?

Integrated

The first main strength of E2E testing is that they allows us to implicitly test a lot of functionality all at once,

Broad

and a passing E2E test suite usually means we have a working application, which is the main goal of web development.

E2E tests have weaknesses too, though.

Slow

Firstly, they're usually slow to run. Since you have to boot the whole app on a server and then in a browser to test it, it commonly takes 5 or 10 mins for an E2E suite to complete.

Brittle

E2E can also be brittle. An inconsequential change, like changing a class on an HTML tag, can bring down the entire suite.

Imprecise

And finally, E2E tests won't pinpoint the cause of a failure. You'll know, for example, that the login process doesn't work, but you probably won't know whether it's a problem in the frontend or the backend, let alone which specific part of the code.

Verdict...(use some of both.)

With the pros and cons of unit and E2E testing now understood, hopefully it's clear why the best plan is to...

use some of both.

There's an optimal combination where the weakness of one type of testing will mostly nullified by the strength of the other, giving you a fair amount of certainty that your app is working without overburdening your QA developer, and, at the same time, allowing you to quickly pin-point and eliminate any bugs that arise.

In the next video...(deployment)

That wraps up our discussion about testing, so in the final video we'll be talking about another crucial aspect of enterprise web app development which is ...

deployment. Thanks for watching and I'll see you in the next video.