Test Driven Development

In this lesson, we'll learn what Test Driven Development (TDD) is, the three categories of tests involved, what Unit Tests are and how to build and run them.

WE'LL COVER THE FOLLOWING

Types of tests

There are three broad categories of tests that can be automated in web apps.

1. **Unit tests**

- These test individual functions. For example, if you have a function that's supposed to return true given a certain input and false otherwise, that's easily testable with unit tests.
- It only covers JavaScript code

2. **Integration tests**

- These test functions that are working together. For example, when one function calls another and uses that result in a different function call.
- It only covers JavaScript code. A lot of business logic can be tested here.

3. **Acceptance Tests**

- These test functionality in the web app as a whole through the client interface. For example, when the user clicks on a button, a popup appears.
- A lot of the user experience guarantees are tested here. JavaScript should not be called directly; these tests are concerned with the outcome of user-taken actions.

People have different interpretations of how tests should be split, and there's

People have different interpretations of how tests should be split, and there's no standard approach here. It doesn't matter, though. Labels on tests aren't as important as simply having tests that cover the critical parts of your app. You also shouldn't write tests for the sake of writing tests. Having 100% code coverage (meaning you have tests that hit every line of code) is a mostly meaningless goal. I'd much rather have a concentrated set of tests on the parts that are prone to bugs than an equal distribution of tests that cover 100% of my code.

So let's try some test-driven development! This just means we write the tests before we have the code to make them pass.

In a real application, you'll want to use a testing framework. These make it much more pleasant to write and run your tests. There are many to choose from for each category of tests, but the differences are mostly syntactical. Writing the actual tests is easy in any framework or library. **The hard part is choosing what to test and how to test it.**

Q Which of the following is not a useful test?

COMPLETED 0%

1 of 1

Show Hint

How to run tests?

One thing you might be wondering is how JavaScript tests are run. The code we're testing is meant to run on a browser, and JavaScript is interpreted by browsers too.

The answer is "with Node." Node has become the standard JavaScript runtime for tasks that occur outside of a browser, like testing. With Node, you can load just the JavaScript code you wrote, and call functions directly.

For acceptance tests that require the interaction of HTML and CSS, these are typically run in an automated fashion with **web drivers**. Web drivers allow remote control of browsers. For visual feedback, web drivers can hook into Firefox or Chrome. They can also use what's called a "headless browser," which removes the graphical portion of a browser.