Eric Elliott
Dec 9, 2015

Says science: There is significant empirical evidence that TDD works*.

I didn't forget. Medium lost it when they changed how inline notes work. Grumpy now. Anyway, the Microsoft/IBM/Springer materials are pretty famous you can probably find them quickly with a Google search. Otherwise you'll have to wait for me to get around to it.);

Read more…
2
Conversation with Eric Elliott.

Georgi Georgieff
Nov 7, 2015

Since I write tests with jasmine, I will just leave this here:

Having only actual/expected is not descriptive enough.
If I have 10 tests all I will see at the bottom
//expect(actual).toBe(expected);

Read more…
3
1 response

Eric Elliott
Nov 7, 2015

This is one of the problems with Jasmine.

Sometimes using literal expectations does not clearly describe the requirement that you're testing. In other words, with Jasmine, it is not always possible to answer all 5 questions that every unit test must answer.

See "Why I Use Tape Instead of Mocha and So Should You."

4
Recommended by Eric Elliott (author)

Warren Wise
Mar 28

I found the link to the Springer Science research paper. It's just one study, not multiple. I didn't find it impressive, but it's more objective than what's typical:

http://research.microsoft.com/en-us/groups/ese/nagappan_tdd.pdf

I like TDD as a design approach. It allows the right design and architecture to emerge, if used as intended. Unless you're building something for the umpteenth time, you may get the design wrong at first. And over time, a design easily becomes unsuited to new demands on the system.

TDD (as explained by Kent Beck, and other early XP practitioners) makes sense to me on its own. The little bit of exposure to, and experience with, it convinces me of its power. For those who need something more objective, the study may be helpful in persuading them to try it for themselves and to form their own opinion.

1
Conversation with Eric Elliott.

Roan Bester
Jun 3

TDD is not always fully possible, especially when working with proprietary tech. When you set out, you need to first learn the proprietary kit/framework, then write some minimal code to test that your code hooks up to everything as advertised. This is essentially pioneering code where the design is more informed by the proprietary tech than by you.

Read more…
1 response

Eric Elliott
Jun 4

This is what integration tests are for. See "Unit vs Functional vs Integration Tests"

3
Conversation with Eric Elliott.

Artem Stepanenko
Sep 2

I like your article very much. But not everything is clear to me.

Since you use only `assert.equal(...)`, how do you verify that the system under test calls a specific function from the dependent object? Or you don't test it at all?

There are pretty many applications of this kind of tests.

1 response

Eric Elliott
Sep 3

Create a spy with something like sinon.js.

Assign the results of `spy.calledOnce` to the actual value. Assign `true` to your expected value.

If you want to assert for call arguments, most spy/stub libraries also have something like `.calledWith()` that takes your expected arguments and returns true...

Ahmed Şeref Güneysu
Jul 7

TDD can encourage more modular designs

Writing tests requires testable code which obeys *Single Responsibility rule*(The “S” of SOLID principles)

1

Marco Romero
Aug 30, 2015

Hi Eric, great article. I agree on the importance and benefits of implementing TDD approach on every project. I am also promoting this practice with my teams and company. Jus a few days ago I wrote an article about the value of TDD and how to implement it on a project. I would really appreciate your feedback about it.

Btw, I admire the work you are doing with #jshomes

Ian Thomas
Sep 24, 2015

I also like JB Rainsberger’s five things each unit test should do: 0, 1, some, lots, oops! https://vimeo.com/80533536

Shirley Perkins
Sep 27

2017 Unit testing trends & best practice frameworks: http://blog.testproject.io/2016/09/01/front-end-development-unit-test-automation-trends2/