JavaScript unit test tools for TDD

Asked 16 years, 1 month ago Modified 2 years, 5 months ago

Viewed 540k times

715

This question's answers are a community effort. Edit existing answers to improve this post. It is not currently accepting new answers or interactions.

I've looked into and considered many JavaScript unit tests and testing tools, but have been unable to find a suitable option to remain fully TDD compliant. So, is there a JavaScript unit test tool that is fully TDD compliant?

javascript unit-testing tdd

Share Follow

edited Jan 18, 2014 at 1:56

community wiki
16 revs, 12 users 32%
Mark Levison

Comments disabled on deleted / locked posts / reviews

13 Answers

Sorted by: Highest score (default)

1518

Karma or Protractor

Karma is a JavaScript test-runner built with Node.js and meant for unit testing.

The Protractor is for end-to-end testing and uses Selenium Web Driver to drive tests.

Both have been made by the Angular team. You can use any assertion-library you want with either.

Screencast: Karma Getting started

related:

- Should I be using Protractor or Karma for my end-to-end testing?
- Can Protractor and Karma be used together?

pros:

- Uses Node.js, so compatible with Win/OS X/Linux
- Run tests from a browser or headless with PhantomJS
- Run on multiple clients at once
- Option to launch, capture, and automatically shut down browsers
- Option to run server/clients on development computer or separately

- Run tests from a command line (can be integrated into ant/maven)
- Write tests xUnit or BDD style
- Supports multiple JavaScript test frameworks
- Auto-run tests on save
- Proxies requests cross-domain
- Possible to customize:
    - Extend it to wrap other test-frameworks (Jasmine, Mocha, QUnit built-in)
    - Your own assertions/refutes
    - Reporters
    - Browser Launchers
- Plugin for WebStorm
- Supported by NetBeans IDE

Cons:

- Does not support Node.js (i.e. backend) testing
- No plugin for Eclipse (yet)
- No history of previous test results

mocha.js

I'm totally unqualified to comment on mocha.js's features, strengths, and weaknesses, but it was just recommended

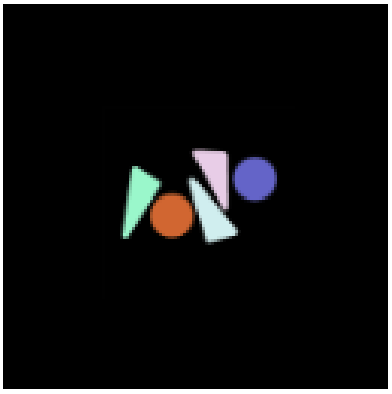
to me by someone I trust in the JS community.

List of features, as reported by its website:

- browser support
- simple async support, including promises
- test coverage reporting
- string diff support
- JavaScript # API for running tests
- proper exit status for CI support etc
- auto-detects and disables coloring for non-ttys
- maps uncaught exceptions to the correct test case
- async test timeout support
- test-specific timeouts
- growl notification support
- reports test durations
- highlights slow tests
- file watcher support
- global variable leak detection
- optionally run tests that match a regexp
- auto-exit to prevent "hanging" with an active loop
- easily meta-generate suites & test-cases
- mocha.opts file support
- clickable suite titles to filter test execution

- node debugger support
- detects multiple calls to done()
- use any assertion library you want
- extensible reporting, bundled with 9+ reporters
- extensible test DSLs or "interfaces"
- before, after, before each, after each hook
- arbitrary transpiler support (coffee-script etc)
- TextMate bundle

yolpo



> This no longer exists, redirects to sequential.js instead

Yolpo is a tool to visualize the execution of JavaScript. JavaScript API developers are encouraged to write their use cases to show and tell their API. Such use cases forms the basis of regression tests.

AVA

Futuristic test runner with built-in support for ES2015. Even though JavaScript is single-threaded, I/O in Node.js can happen in parallel due to its async nature. AVA takes advantage of this and runs your tests concurrently, which is especially beneficial for I/O heavy tests. In addition, test files are run in parallel as separate processes, giving you even better performance and an isolated environment for each test file.

- Minimal and fast
- Simple test syntax
- Runs tests concurrently
- Enforces writing atomic tests
- No implicit globals
- Isolated environment for each test file
- Write your tests in ES2015
- Promise support

- Generator function support
- Async function support
- Observable support
- Enhanced asserts
- Optional TAP o utput
- Clean stack traces

Buster.js

A JavaScript test-runner built with Node.js. Very modular and flexible. It comes with its own assertion library, but you can add your own if you like. The assertions library is decoupled, so you can also use it with other test-runners. Instead of using `assert(!...)` or `expect(...).not...`, it uses `refute(...)` which is a nice twist imho.

> A browser JavaScript testing toolkit. It does browser testing with browser automation (think JsTestDriver), QUnit style static HTML page testing, testing in headless browsers (PhantomJS, jsdom, ...), and more. Take a look at the overview!
>
> A Node.js testing toolkit. You get the same test case library, assertion library, etc. This is also great for hybrid browser and Node.js code. Write your test case with `Buster.JS` and run it both in Node.js and in a real browser.

Screencast: Buster.js Getting started (2:45)

pros:

- Uses Node.js, so compatible with Win/OS X/Linux
- Run tests from a browser or headless with PhantomJS (soon)
- Run on multiple clients at once
- Supports Node.js testing
- Don't need to run server/clients on development computer (no need for IE)
- Run tests from a command line (can be integrated into ant/maven)
- Write tests xUnit or BDD style
- Supports multiple JavaScript test frameworks
- Defer tests instead of commenting them out
- SinonJS built-in
- Auto-run tests on save
- Proxies requests cross-domain
- Possible to customize:
    - Extend it to wrap other test-frameworks (JsTestDriver built in)
    - Your own assertions/refutes
    - Reporters (xUnit XML, traditional dots, specification, tap, TeamCity and more built-in)

    - Customize/replace the HTML that is used to run the browser-tests
- TextMate and Emacs integration

Cons:

- Stil in beta so can be buggy
- No plugin for Eclipse/IntelliJ (yet)
- Doesn't group results by os/browser/version like TestSwarm *. It does, however, print out the browser name and version in the test results.
- No history of previous test results like TestSwarm *
- Doesn't fully work on windows as of May 2014

* TestSwarm is also a Continuous Integration server, while you need a separate CI server for `Buster.js`. It does, however, output xUnit XML reports, so it should be easy to integrate with Hudson, Bamboo or other CI servers.

TestSwarm

https://github.com/jquery/testswarm

TestSwarm is officially no longer under active development as stated on their GitHub webpage. They recommend Karma, browserstack-runner, or Intern.

Jasmine

This is a behavior-driven framework (as stated in quote below) that might interest developers familiar with Ruby or Ruby on Rails. The syntax is based on RSpec that are used for testing in Rails projects.

Jasmine specs can be run from an HTML page (in qUnit fashion) or from a test runner (as Karma).

> Jasmine is a behavior-driven development framework for testing your JavaScript code. It does not depend on any other JavaScript frameworks. It does not require a DOM.

If you have experience with this testing framework, please contribute with more info :)

Project home: http://jasmine.github.io/

QUnit

QUnit focuses on testing JavaScript in the browser while providing as much convenience to the developer as possible. Blurb from the site:

> QUnit is a powerful, easy-to-use JavaScript unit test suite. It's used by the jQuery, jQuery UI, and

> jQuery Mobile projects and is capable of testing any generic JavaScript code

QUnit shares some history with TestSwarm (above):

> QUnit was originally developed by John Resig as part of jQuery. In 2008 it got its own home, name and API documentation, allowing others to use it for their unit testing as well. At the time it still depended on jQuery. A rewrite in 2009 fixed that, now QUnit runs completely standalone. QUnit's assertion methods follow the CommonJS Unit Testing specification, which was to some degree influenced by QUnit.

Project home: http://qunitjs.com/

Sinon

Another great tool is sinon.js by Christian Johansen, the author of Test-Driven JavaScript Development. Best described by himself:

> Standalone test spies, stubs and mocks for JavaScript. No dependencies works with any unit testing framework.

Intern

The Intern Web site provides a direct feature comparison to the other testing frameworks on this list. It offers more features out of the box than any other JavaScript-based testing system.

JEST

A new but yet very powerful testing framework. It allows snapshot based testing as well this increases the testing speed and creates a new dynamic in terms of testing

Check out one of their talks:
https://www.youtube.com/watch?v=cAKYQpTC7MA

Better yet: Getting Started

Share Improve this answer Follow

edited Jul 12, 2022 at 0:39

community wiki
55 revs, 32 users 34%
gregers

2 Jasmine can work headless using V8, but you can also use it interactively. While the DOM is not necessary with respect to Jasmine, *your codebase* might access DOM. With discipline it is possible to eliminate, guard with conditions, or provide mocks for parts of the code that access the DOM and run tests completely apart from HTML fixtures. You can also get command-line support and fixtures using add-ons. – jerseyboy Jan 15, 2012 at 6:19

2 @rehevkor5: Selenium is for integration testing, while the tools here are for unit testing. typemock.com/unit-tests-integration-tests – gregers Jun 14, 2012 at 11:58

31 Almost every single test runner relies on a browser. Wtf, doesn't anyone ever run unit tests *only* on the server-side???? – user9903 Jan 19, 2014 at 19:41

2 Wouldn't it be better to split / divide each alternative in different answers? It might invalidate the current votes on this one, but I think it'd make the most sense. – cregox Mar 31, 2015 at 23:30

2 @Raisen You can plug ES 2015 into most of them with Babel, but AVA by Sindre Sorhus has it built in. – gregers Dec 8, 2015 at 8:13

64

Take a look at the Dojo Object Harness (DOH) unit test framework which is pretty much framework independent harness for JavaScript unit testing and doesn't have any Dojo dependencies. There is a very good description of it at Unit testing Web 2.0 applications using the Dojo Objective Harness.

If you want to automate the UI testing (a sore point of many developers) — check out doh.robot *(temporary down. update: other link http://dojotoolkit.org/reference-guide/util/dohrobot.html)* and dijit.robotx *(temporary down)*. The latter is designed for an acceptance testing. Update:

Referenced articles explain how to use them, how to emulate a user interacting with your UI using mouse

and/or keyboard, and how to record a testing session, so you can "play" it later automatically.

Share Improve this answer Follow

edited Aug 25, 2011 at 18:30

community wiki
4 revs, 3 users 63%
Eugene Lazutkin

Thanks for the suggestion of Dojo Object Harness, I never would have found it. I appreciate the other suggestions - but one step at a time. – Mark Levison Nov 19, 2008 at 15:06

I've actually used this in a previous project, and found it invaluable. But then again, I can't compare - haven't used any other TDD framework. – Rakesh Pai Nov 19, 2008 at 17:20

Thanks for reporting dead links. I updated one of them, and will replace links to robots docs as soon as I they are up on a new web site. – Eugene Lazutkin Mar 25, 2011 at 18:01

One thing I don't like about DOH is that line numbers aren't reported when assertions fail. Commenting them out manually and re-running the test works. – Aram Kocharyan Jan 1, 2014 at 8:08

Dojo is replacing DOH with TheIntern testing framework. TheIntern is very powerful and has substantial improvements. sitepen.com/blog/2014/02/18/… – user64141 Nov 20, 2014 at 22:15

Chutzpah - A JavaScript Test Runner

35

I created an open source project called Chutzpah which is a test runner for JavaScript unit tests. Chutzpah enables you to run JavaScript unit tests from the command line and from inside of Visual Studio. It also supports running in the TeamCity continuous integration server.

Share Improve this answer Follow

edited Jun 4, 2015 at 20:22

community wiki
3 revs, 2 users 93%
Matthew Manela

7 I just started using Chutzpah to run Jasmine tests inside visual studio - it's nicely integrated: right click in the test file and chose 'run js tests' or 'run JS tests in browser'. I run the same jasmine tests using JSTestDriver. I prefer Chutzpah because I specify which files I depend upon being loaded at the top of the test file. For JSTestDriver I need a separate config file. – GarethOwen Mar 7, 2012 at 16:00

28

The JavaScript section of the Wikipedia entry, List of Unit Testing Frameworks, provides a list of available choices. It indicates whether they work client-side, server-side, or both.

Share Improve this answer Follow

answered Jun 1, 2009 at 18:07

community wiki
Pete TerMaat

BusterJS

15

There is also BusterJS from Christian Johansen, the author of Test Driven Javascript Development and the Sinon framework. From the site:

> Buster.JS is a new JavaScript testing framework. It does browser testing by automating test runs in actual browsers (think JsTestDriver), as well as Node.js testing.

Share Improve this answer Follow

edited Nov 8, 2015 at 13:32

community wiki
2 revs, 2 users 92%
Tauren

google-js-test:

11

JavaScript testing framework released by Google: https://github.com/google/gjstest

> - Extremely fast test startup and execution time, without having to run a browser.

- Clean, readable output in the case of both passing and failing tests.
- A browser-based test runner that can simply be refreshed whenever JS is changed.
- Style and semantics that resemble Google Test for C++.
- A built-in mocking framework that requires minimal boilerplate code (e.g. no `$tearDown` or `$verifyAll`) with style and semantics based on the Google C++ Mocking Framework.

There are currently no binaries for Windows

Share Improve this answer Follow

edited Jun 20, 2020 at 9:12

community wiki
3 revs, 2 users 54%
user

3 It seems to have almost zero interest on Github, also it requires unix-bases OS, and I'm a huge windows fan, I don't leave my house without kissing my windows machine goodbye. – vsync Nov 7, 2015 at 22:58

We are now using Qunit with Pavlov and JSTestDriver all together. This approach works well for us.

9

[QUnit]

[Pavlov], [source]

[jsTestDriver], [source]

Share Improve this answer Follow

edited Mar 7, 2014 at 16:59

community wiki
4 revs, 4 users 59%
Tom Stickel

Would you care to explain what is the role in each of these in the whole testing process and how they connect with each other? – vsync Nov 7, 2015 at 23:01

Sorry it has been a long time and many contract jobs ago to recall the details on this. – Tom Stickel Nov 8, 2015 at 7:15

7

You have "runs on actual browser" as a pro, but in my experience that is a con because it is slow. But what makes it valuable is the lack of sufficient JS emulation from the non-browser alternatives. It could be that if your JS is complex enough that only an in browser test will suffice, but there are a couple more options to consider:

HtmlUnit: "It has fairly good JavaScript support (which is constantly improving) and is able to work even with quite complex AJAX libraries, simulating either Firefox or Internet Explorer depending on the configuration you

want to use." If its emulation is good enough for your use then it will be much faster than driving a browser.

But maybe HtmlUnit has good enough JS support but you don't like Java? Then maybe:

Celerity: Watir API running on JRuby backed by HtmlUnit.

or similarly

Schnell: another JRuby wrapper of HtmlUnit.

Of course if HtmlUnit isn't good enough and you have to drive a browser then you might consider Watir to drive your JS.

Share Improve this answer Follow

answered Nov 19, 2008 at 6:11

community wiki
Jeffrey Fredrick

7

YUI has a testing framework as well. This video from Yahoo! Theater is a nice introduction, although there are a lot of basics about TDD up front.

This framework is generic and can be run against any JavaScript or JS library.

Share Improve this answer Follow

edited Feb 26, 2013 at 11:03

community wiki
2 revs, 2 users 86%
Matthew Taylor

5 Be aware that Yahoo has stopped all new development of YUI – Peter V. Mørch Oct 24, 2014 at 7:04

4

You might also be interested in the unit testing framework that is part of qooxdoo, an open source RIA framework similar to Dojo, ExtJS, etc. but with quite a comprehensive tool chain.

Try the online version of the testrunner. Hint: hit the gray arrow at the top left (should be made more obvious). It's a "play" button that runs the selected tests.

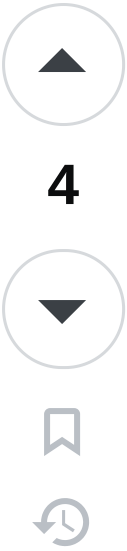
To find out more about the JS classes that let you define your unit tests, see the online API viewer.

For automated UI testing (based on Selenium RC), check out the Simulator project.

Share Improve this answer Follow

answered Nov 28, 2008 at 19:54

community wiki
Andreas Ecker



We added JUnit integration to our Java to Javascript code generator ST-JS (http://st-js.org). The framework generates to corresponding Javascript for both the tested code and the unit tests and sends the code to different browsers.

There is no need for a separate server as the unit test runner opens the needed http port (and closes it once the tests finished). The framework manipulates the Java stacktrace so that the failed asserts are correctly displayed by the JUnit Eclipse plugin. Here is a simple example with jQuery and Mockjax:

```
@RunWith(STJSTestDriverRunner.class)
@HTMLFixture("<div id='fortune'></div>")

@Scripts({ "classpath://jquery.js",
       "classpath://jquery.mockjax.js", "classpath://j
public class MockjaxExampleTest {
  @Test
  public void myTest() {
    $.ajaxSetup($map("async", false));
    $.mockjax(new MockjaxOptions() {
      {
        url = "/restful/fortune";
        responseText = new Fortune() {
          {
            status = "success";
            fortune = "Are you a turtle?";
          }
        };
      }
    });

    $.getJSON("/restful/fortune", null, new Callback3<
JQueryXHR>() {
      @Override
```

```
        public void $invoke(Fortune response, String p2,
          if (response.status.equals("success")) {
            $("#fortune").html("Your fortune is: " + res
          } else {
            $("#fortune").html("Things do not look good,
          }

        }
      });
      assertEquals("Your fortune is: Are you a turtle?",
    }

    private static class Fortune {
      public String status;
      public String fortune;
    }
}
```

Share Improve this answer Follow

edited Nov 12, 2013 at 17:20

community wiki
2 revs, 2 users 99%
alex.c

3

You should have a look at env.js. See my blog for an example how to write unit tests with env.js.

Share Improve this answer Follow

edited Feb 25, 2009 at 10:18

community wiki
2 revs
Aaron Digulla

2

[MochiKit](#) has a testing framework called SimpleTest that seems to have caught on. Here's a [blog post from the original author](#).

Share Improve this answer Follow

edited Jul 22, 2009 at 16:12

community wiki
2 revs, 2 users 75%
p.campbell

SimpleTest - another great option. I'm glad I'm not doing the investigative work here ;-) – Mark Levison Nov 19, 2008 at 15:05