JavaScript Testing Survey 2018 Results

Gojko Adzic, gojko@neuri.co.uk

25 February 2018

Contents

Introduction

This document summarises the results of the JavaScript Testing Survey, conducted between 15 January and 15 February 2018.

Software developers answered questions relating to their testing tools and practices, and rated the benefits and usability of their testing approach. The questions relating to *process benefits* asked respondents to rate how much confidence their tests provide to release frequently, how well they prevent bugs and how well they document design decisions. The questions relating to *tool usability* asked respondents to rate how easily they work with their tools, and how easily they can write, maintain and understand test cases. The combined scores for those groups of questions give a nice indication about how happy teams are with their tooling choices.

Of course, the process benefits depend more on the contents of the test cases rather than the tools themselves, but a big difference in the benefits rating could point out that a particular tool is not suitable for some key workflow. This means that the tool is better if it scores higher on the usability rating, as long as its process benefit ratings within reasonable range of similar top-voted tools.

Also, note that the benefits and usability questions were formulated so that people rated their entire setup, not an individual tool, which is important where they were using a combination of tools. I have separated out results for teams using individual tools and combinations of tools below, to make it easier to evaluate the differences.

Key findings

- Mocha is still the most popular tool individually, followed by Jasmine and Jest.
- All the other tools are significantly less popular.
- Although Cucumber JS on its own isn't popular enough to get to the top 3 choices on the list, combining Cucumber with one of the more popular tools seems to make a big difference for developer usability.
- The highest rated tool choice, regarding developer usability, is the combination of Cucumber and Jest, scoring 10.44 (70%). This is higher than any single tool rating, or any other combination. However, this combination is just above the margin of error for the survey, so more data is needed to fully confirm this finding.
- Roughly 12% of the respondents used no test automation tools for JavaScript code

Notes on the survey population and accuracy

The survey includes 638 responses. Assuming 22,000,000 software developers worldwide[1], the margin of error is 4% at the 95% confidence level. This means that a difference less than 26 respondents is within the margin of error. (At the 99% confidence level, the margin of error is 5%, or roughly 32 respondents).

[1]Evans Data Corporation, Global Developer Population and Demographic Study 2017, https://evansdata.com/reports/viewRelease.php?reportID=9

The maximum score for combined ratings is 15, so a score of 10 would correspond to a 66% rating.

Most questions are multiple-choice, meaning the total number of responses for all answers can be over 100%.

Here are some interesting statistics about the respondents (answers below margin of error omitted):

Table 1: What kind of applications does you mostly work on?

Answer	Responses	Response %
Both	440	64%
Front end (web UI, mobile apps. . .)	159	23%
Back end (node.js, APIs, web servers. . .)	84	12%

Table 2: Does your product use any front-end framework?

Answer	Responses	Response %
React	292	43%
Angular	231	34%
Vue	61	9%
Our own	52	8%

Table 3: What types of testing does your team practice on the current project?

Answer	Responses	Response %
Automated Unit testing	557	82%
Manual exploratory testing	396	58%
Automated integration testing	389	57%
Automated acceptance testing	264	39%
Manual scripted testing	198	29%
Usability testing	161	24%
Automated performance testing	120	18%

Survey Results

Answers below the margin of error are mostly omitted from the tables below. Percentages of responses are rounded to the nearest number.

Which JavaScript test automation frameworks does your team use?

Table 4: Popular test automation frameworks

Tool	Responses	Response %	Process rating	Usability rating
Mocha	307	45%	10.31	9.23
Jasmine	258	38%	9.91	8.84
Jest	201	29%	10.07	9.49
Cucumber JS	106	16%	10.55	9.83
qUnit	32	5%	10.06	8.78

Table 5: Aggregated numbers of unique entries for test frameworks by respondent

Count	Responses	Response %
0	81	12%
1	339	50%
2	163	24%
3	74	11%

Table 6: Only teams using a single test framework

Tool	Responses	Response %	Process rating	Usability rating
Mocha	106	16%	10.37	9.63
Jasmine	102	15%	9.62	8.87
Jest	67	10%	9.63	9.66
Cucumber JS	32	5%	10.09	10.03

Table 7: Only teams developing React applications

Tool	Responses	Response %	Process rating	Usability rating
Jest	162	24%	10.11	9.72
Mocha	162	24%	10.49	9.51
Jasmine	91	13%	9.98	9.03
Cucumber JS	53	8%	10.70	10.26

Table 8: Only teams developing Angular applications

Tool	Responses	Response %	Process rating	Usability rating
Jasmine	142	21%	9.96	8.92
Mocha	94	14%	9.97	8.76
Cucumber JS	42	6%	10.05	9.36
Jest	38	6%	10.92	9.68

Table 9: Only teams developing mostly back-end applications

Tool	Responses	Response %	Process rating	Usability rating
Mocha[2]	46	6%	10.87	9.48
Jasmine	20	2%	9.60	8.80
Jest	14	2%	9.79	8.14
Cucumber JS	13	1%	10.85	10.0
Intern	3	0%	12.00	8.33
qUnit	3	0%	10.67	9.00

Table 10: Combined ratings for teams using more than one tool

Tool	Responses	%	Process rating	Usability rating
Jasmine+Mocha	121	18%	10.12	8.79
Jest+Mocha	98	14%	10.39	9.22
Jasmine+Jest	65	10%	10.17	8.92
Cucumber JS+Mocha	45	7%	**11.09**	9.69
Cucumber JS+Jasmine	38	6%	10.58	9.66
Cucumber JS+Jest	27	4%	10.78	**10.44**

Which JS mocking and expectation libraries do you use?

Table 11: Utility testing libraries by popularity

Library	Responses	Response %
Chai	252	37%
built-in libraries with the test automation tool	243	36%
Sinon	183	27%
Testdouble	25	4%

[2]Only the first answer in this table is above the margin of error, the others are there more for illustrative purposes.

Do you use any code coverage tools for JavaScript?

Table 12: Test code coverage tools

Answer	Responses	Response %
We don't track test coverage	378	55%
Istanbul	132	19%
included with our test tool	120	18%

Do you use any continuous integration tools or services?

Table 13: Continuous integration tools – note the ratings are for the whole tool setup by the team, not just the entry in the table

Tool	Responses	%	Process rating	Usability rating
Jenkins	320	47.0%	9.87	9.02
TeamCity	85	12.0%	9.86	9.15
Travis	72	11.0%	**10.68**	9.18
CircleCI	69	10.0%	10.30	**9.29**
Home-grown	27	4.0%	8.70	8.44

Do you use browser stacks/automation tools?

Table 14: Browser automation tools – note the ratings are for the whole tool setup by the team, not just the entry in the table

Tool	Responses	%	Process rating	Usability rating
Selenium	295	43%	10.15	9.21
Protractor	96	14%	10.15	9.24
Browserstack	81	12%	9.98	8.79
Nightwatch	47	7%	10.49	**9.42**
Sauce labs	41	6%	**10.98**	9.37

Do you use any of these browser test runners?

Table 15: Browser test runners

Answer	Responses	Response %
Karma	259	38%
included in our test tool	102	15%

Do you use any exception tracking tools?

Table 16: Exception and crash tracking tools

Answer	Responses	Response %
Not tracking errors	362	53%
Web analytics (eg Google Analytics)	105	15%
Sentry	104	15%
TrackJS	26	4%
Rollbar	23	3%

More information

Check out https://gojko.net/2018/02/25/javascript-testing-tools.html for a more in-depth analysis of the tool popularity findings.

Follow-up research

If you would like to participate in a similar research next year, or get notified when I publish the next results, please sign up here http://eepurl.com/dl0dT1. Alternatively, if you're just interested in the results, follow gojkoadzic on Twitter.

About the author

Gojko Adzic is a partner at Neuri Consulting LLP. He is the winner of the 2016 European Software Testing Outstanding Achievement Award, and the 2011 Most Influential Agile Testing Professional Award. Gojko's book Specification by Example won the Jolt Award for the best book of 2012, and his blog won the UK Agile Award for the best online publication in 2010.

Gojko is a frequent speaker at software development conferences and one of the authors of MindMup and Claudia.js.

As a consultant, Gojko has helped companies around the world improve their software delivery, from some of the largest financial institutions to small innovative startups. Gojko specialises in are agile and lean quality improvement, in particular impact mapping, agile testing, specification by example and behaviour driven development.

Copyright note

[3]https://creativecommons.org/licenses/by/4.0/