

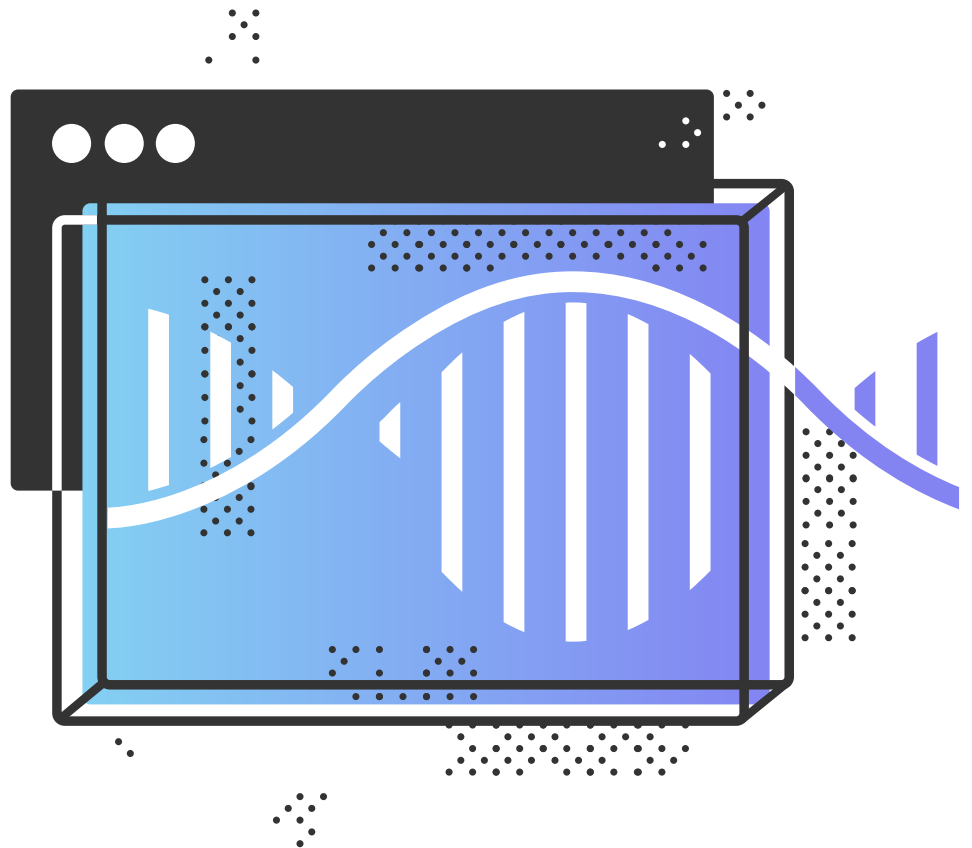
MDN

Web Developer

Needs Assessment

2019

 MDN web docs
moz://a



Acknowledgements

This report is the result of contributions from the MDN Product Advisory Board (PAB) which consists of the following companies:

- Bocoup
- Google
- Microsoft
- Mozilla
- Samsung
- W3C

We'd like to thank the following people for their invaluable contributions to the MDN Developer Needs Assessment:

All PAB members: Ali Spivak, Chris Mills, Daniel Appelquist, Dominique Hazael-Massieux, Joe Medley, Jory Burson, Kadir Topal, Kyle Pflug, Meggin Kearney, Patrick Kettner, Robert Nyman, and Travis Leithead

Outside of the PAB, individuals who contributed significantly include: Andreas Bovens, Andrew Overholt, Dietrich Ayala, Harald Kirschner, Harleen Batra, Marcos Caceres, Martin Balfanz, Nancy Hang, Panagiotis Astithas, Philip Jägenstedt, Rick Byers, Shi Li Li, Venetia Tay

Contributors from Pinpoint, which is the agency that conducted the study and authored this report: Allison McKeever, Dwayne King, Gabe Grayum

Table of Contents

Introduction.....	3
Study Participants.....	5
Target.....	6
Recruited.....	7
Actual.....	8-14
Needs Assessment.....	15
What is a Need?.....	16
Ranking Methodology.....	17
Themes.....	18
Overall Ranking.....	19
Segmented View of the Needs Ranking.....	20-25
How Developers Felt About the Needs List.....	26
Overall Satisfaction with the Web.....	27-28
What's Missing From the Web.....	29-30
Access to Hardware.....	31
Browser Compatibility.....	32
Access to Filesystem.....	33
Performance.....	34
PWA Support.....	35
Debugging.....	36
Access to Native APIs.....	37
Technologies.....	38
Programming Languages.....	39-43
Adoption of New Technologies.....	44-45
Browsers.....	46-47
Pilot Interview Findings.....	48
Enhanced View of the Top Ten Needs.....	49-59
Enhanced View of the Remaining Needs.....	60-68
Outliers.....	69-75
Overcoming Frustrations.....	76-77
Developer Needs are Situational.....	78-80
Broader Picture.....	81-88
Conclusion.....	89-90
Methodology.....	91-93

Introduction

Introduction

The report you are now reading is the first edition of a global, annual study of designer and developer needs on the web. It is intended to shape the future of the web platform.

On single-vendor platforms, only one organization has to research developer needs and decide how to address them in the future. It's not that straightforward on the web, where multiple organizations need to be involved in feature decisions, from browser vendors to standards bodies and industry. As a result, change can be slow to come, which means that pain points may take a long time to address.

In discussions with people involved in the standardization and implementation of web platform features, one need came up repeatedly, "We need to hear more from developers."

That is what the MDN Web Developer Needs Assessment aspires to be: the voice of developers and designers working on the web.

Like the community, this assessment is not owned by a single organization. It was not tailored to fit the priorities of participating browser vendors, or to mirror other existing assessments. These findings are published under the umbrella of the MDN Product Advisory Board, and the survey used for data collection was designed with input from more than 30 stakeholders representing board member organizations including browser vendors, the W3C, and industry.

This report would not exist without the input of more than 28,000 developers and designers from 173 countries who took the twenty minutes necessary to complete the survey. That's more than 10,000 hours contributed by the community to provide an understanding of the pain points, wants, and needs of people working to build the web.

The input provided by survey participants is already influencing

how browser vendors prioritize feature development to address the needs of designers and developers, both on and off the web. By producing this report annually, it will be possible to track changing needs and pain points over time, enabling all stakeholders to see the impact of their efforts on the future of the web.

Study Participants



Target

For the first iteration of the Developer Needs Assessment, our focus was people who spend at least some of their time writing code for the Web. This includes developers and designers. Inherent in this target audience is a selection bias of those who are working on the Web today. The voice of those who have abandoned the platform, whether because of dissatisfaction or other reasons, is left to future iterations of this study. Similarly, those who cannot or do not choose the Web platform are not a part of this study.



Recruited

When the survey launched, it was announced on MDN as well as through tweets and other social network posts of the MDN community. The initial responses are the most diverse as participants were drawn in through the various social network promotions. As time progressed, the banner on MDN remained and was the prominent recruiting vehicle. The active publicity on MDN created another selection bias towards those who use MDN. However, MDN serves a large percentage of the developer and designer community.

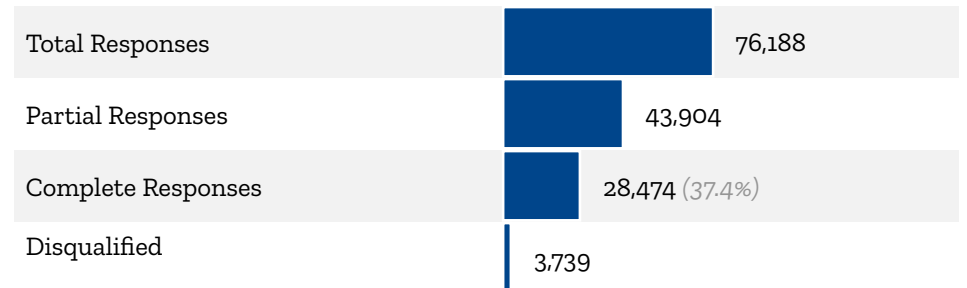


Actual

The survey had 76,118 responses. Of those, 28,474 counted as complete responses for a completion rate of 37.4%. Partial and disqualified responses were not included in the analysis.

Survey Responses

The completed responses are further broken down by gender, region, country, type of developer, and experience level on the following pages.

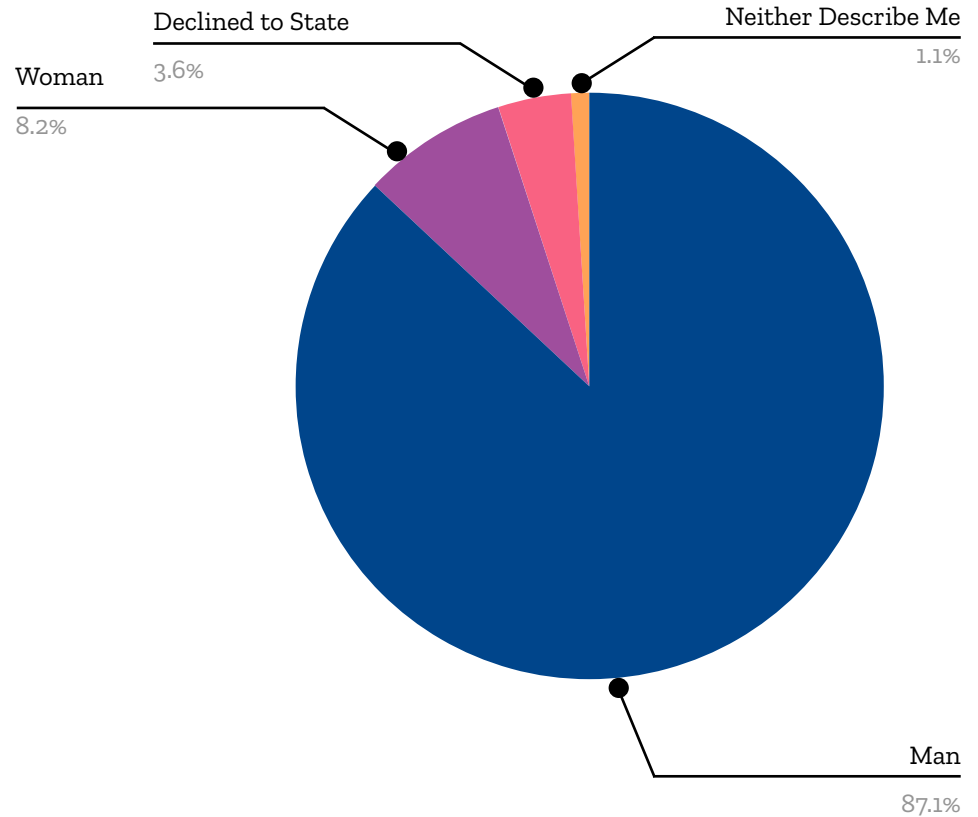


Survey Responses By Gender

A goal from the onset of this project was to have a broad, global representation of the developer community. Despite attempts to get the survey in front of representative audiences, 87.1% of the respondents were men compared to 8.2% women and 1.1%, where neither option was suitable. To put this into perspective, the US Bureau of Labor Statistics¹ estimates that women's participation in the software developer workforce is more like 20%, though it's not immediately obvious what constitutes their definition of the software developer workforce compared to the audience for this study. When filtering our results by respondents from the United States who selected woman, we have a representation of 10.9%

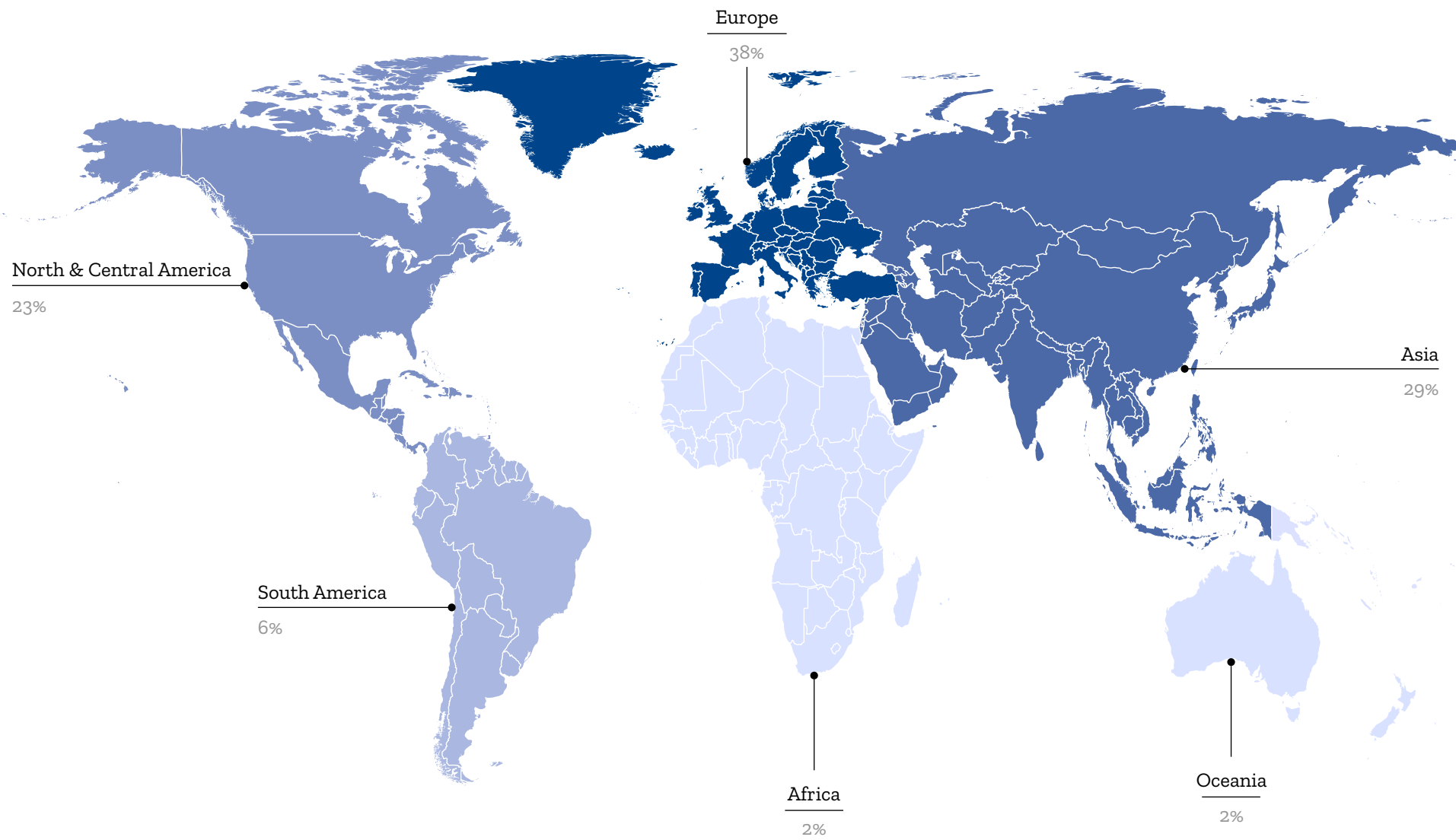
This discrepancy in genders is another bias in the first version of the MDN Web DNA, and unfortunately, is a common problem with many developer surveys. The difference in representation could be a result of how we fielded the survey. Our methods may have contributed to a less representative audience by utilizing outlets that unintentionally exclude or dissuade women and other minority groups from participation. We did attempt to gather more diversity by sending it to specific women-groups. In future iterations, we will continue to aim for fair representation and ways to mitigate or account for the bias.

The options provided as answers to choose from were carefully considered and vetted by Mozilla's legal team. The four choices offered were intentional. We launched the survey globally and had optional questions that asked for personally identifiable information. The degree of legal recognition provided to people who do not identify with a gender consistent with the gender assigned at birth varies widely throughout the world. We did not want to have data on gender that could put people in harm's way. Of completed responses, 55.2% answered the optional questions, which asked for personally identifiable information.



¹ <https://www.bls.gov/opub/reports/womens-databook/2017/home.htm>

Survey Responses By Region



Survey Responses By Country

The survey was localized from English into eight languages listed alphabetically:

- Arabic
- Chinese (simplified)
- French
- Japanese
- Korean
- Portuguese (Brazil)
- Russian
- Spanish

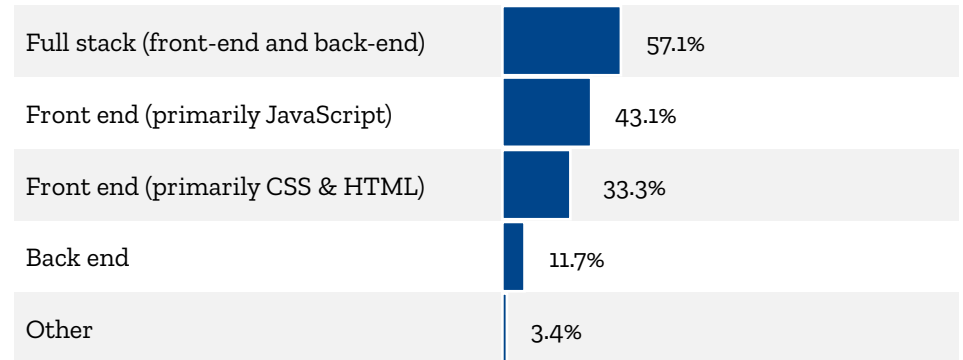
These languages are a combination of stakeholder input as well as what is most accessed on MDN. The translations offered likely influenced who participated in the study and might be an explanation for why there are not more respondents from India.

Though some countries' response rate seems low compared to expectations, the survey includes responses from 173 countries. Nine countries represent 61.7% of the answers. The other 164 countries had 3% or less of total responses. The nine countries with the most significant participation were:

- United States - 17.9%
- China - 8.2%
- Russia - 7.7%
- India - 6.2%
- Germany - 5.7%
- France - 4.9%
- United Kingdom - 4.4%
- Ukraine - 3.6%
- Canada - 3.1%

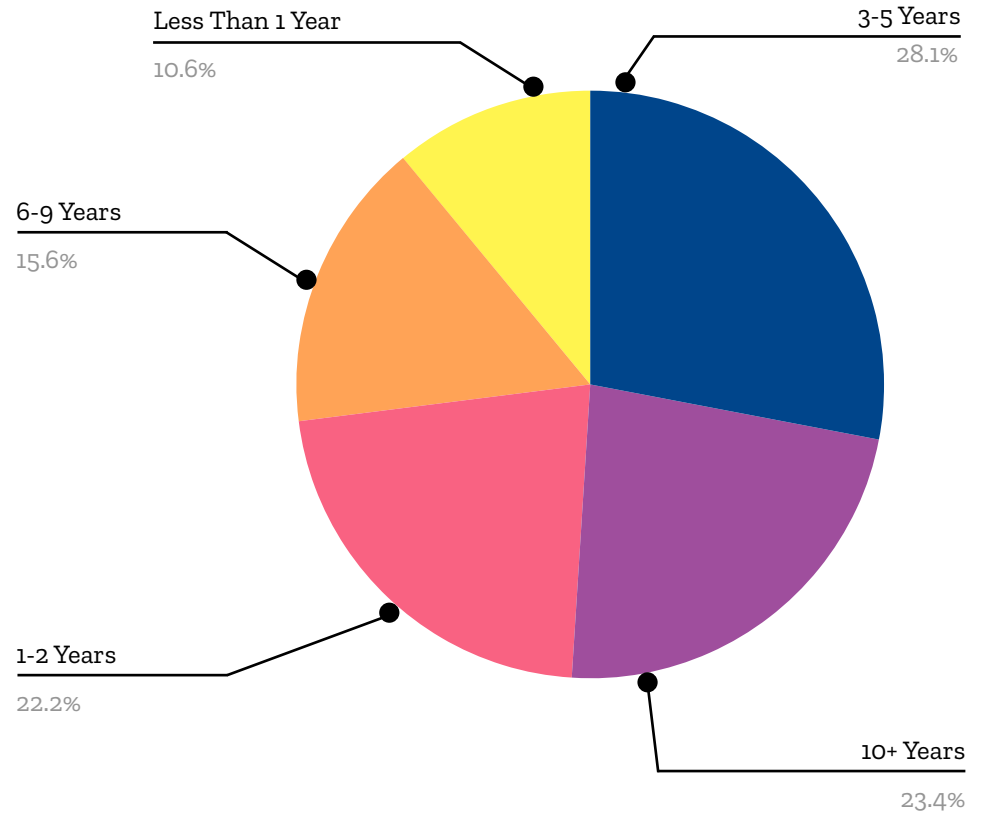
Survey Responses By Type of Developer

Participants were asked, "Which best describes the type of web developer you are?" and were allowed to select multiple options. Most respondents identified as Full Stack or Front-end. The latter had two variations to pick: primarily JavaScript or primarily CSS and HTML. Full stack had the most representation at 57.1%. Back end had the least representation at 11.7%.



Survey Responses By Experience Level

The experience level, measured by the number of years spent as a developer, was fairly even across recent (two years or less) to experienced developers (over ten years), 32.8%, and 23.4%, respectively. The lowest participation was from those with mid-level experience (6-9 years) at 15.6%. Additionally, more than half of all participants, 60.9%, had less than six years of experience.



Needs Assessment

What is a Need?

Before sharing the top ten needs, we're briefly describing what a need is to help set the context for the following Findings section.

The need statements were informed from the fourteen pilot interviews conducted at the beginning of this project. The statements are written from the point of view of a web developer. The outline we used to create the need statements was:

I am a _____ (persona) trying to _____ (verb) but _____ (barrier) because _____ (cause), which makes me feel _____ (emotional reaction).

Putting this into action, it could read as follows:

I am a tourist trying to travel to another country but am struggling to understand the Visa process because it's complex and poorly communicated, which makes me feel frustrated.

We drew upon common practices in design thinking as well as product-development processes for inspiration when deciding to use need statements in the survey. Because they are written from the point of view of developers, we felt it would be an intuitive way to read, interpret, and rank to get to the top ten.

The need statements for this project were centered around the emotional reaction of frustration. If a web developer experiences frustration in regards to web development, there may be an underlying opportunity for browser vendors to help solve that frustration.

Ranking Methodology

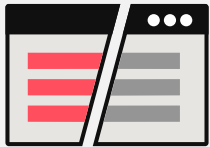
Using the MaxDiff methodology, we asked survey respondents to rank a total of 28 need statements. Respondents saw twelve sets comprised of four need statements. For each set they were instructed to pick the one need that causes them the least frustration and the one need that causes them the most frustration. A single need statement could appear more than once within the twelve sets.

It is important to note that just because a need may not rank as the least frustrating within a set, that does not mean it causes the least frustration. It could imply that the respondent does not have experience with the subject matter or does not prioritize that subject within their work.

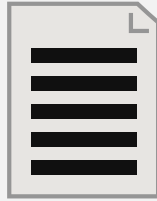
For example, “Making sites accessible,” ranked 24th out of 28, and that’s pretty low. That doesn’t mean making sites accessible is easy, without room for improvement. What we learned during the pilot interviews is that developers and designers are not always given latitude to spend the necessary time on accessibility. Therefore, because they cannot spend the time on it, accessibility does not create frustration. If in the future, developers and designers spend more time on accessibility, their perception of the frustration may change, and so would the ranking.

Themes

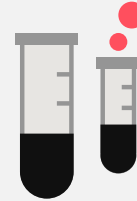
The 28 needs cover 14 different themes. As was mentioned earlier, four of the top ten needs relate to browser compatibility, making it the highest ranked theme. Rounding out the top five themes are Documentation, Testing, Debugging, and Frameworks.



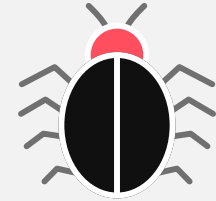
Browser Compatibility



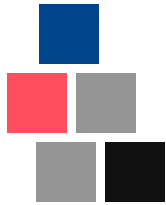
Documentation



Testing



Debugging



Frameworks



Privacy



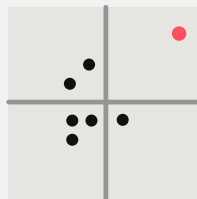
Security



Authentication



Performance



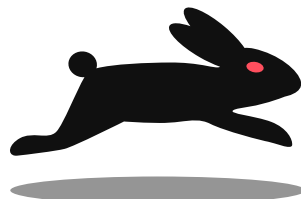
Outliers



Design



Localization



Pace of Change



Accessibility

Overall Needs Ranking

One is the most frustrating and 28 is the least frustrating.

1. Having to support specific browsers (e.g., IE11).
2. Outdated or inaccurate documentation for frameworks and libraries.
3. Avoiding or removing a feature that doesn't work across browsers.
4. Testing across browsers.
5. Making a design look/work the same across browsers.
6. Discovering bugs not caught during testing.
7. Supporting multiple frameworks in the same code base.
8. Keeping up with a large number of new and existing tools or frameworks.
9. Managing user data to comply with laws and regulations.
10. Understanding and implementing security measures.
11. Integrating with third parties for authentication.
12. Pinpointing existing performance issues.
13. Running end-to-end tests.
14. Lack of device APIs allowing for access to hardware.
15. Outdated documentation for HTML, CSS and JavaScript.
16. Determining the root cause of a bug.
17. Capability of the web to support a specified layout.
18. Knowing what browsers support a specific technology.
19. Achieving visual precision on stylized elements (e.g., buttons).
20. Running front-end tests.
21. Implementing localization.
22. Keeping up with changes to the web platform.
23. Implementing performance optimizations.
24. Making sites accessible.
25. Getting users to grant permissions to Web APIs (e.g., geo-location).
26. Deciding what to learn next to keep my skill set relevant.
27. Finding a community of peers.
28. Fixing a bug once it's been identified.

Segmented View of the Needs Ranking

The Overall Needs Ranking shows how the general population, the 28,474 completed survey responses, ranked the 28 needs.

The data is further analyzed by different segments to provide a more nuanced view. When seeing the differences, keep in mind that deviation from the general population is not good or bad; it's just a visual representation of the actual data.

The segments included in this report are:

Country - The seven shown are there because they are the countries that have the most traffic on MDN, the primary recruiting vehicle used for the survey.

Gender - When looking at the results, remember the percentage of respondents who selected "Man" when asked about the gender they identify with (87.1%) and the rate of respondents who selected "Woman" (8.2%). More respondents chose "Man," which helps explain why when filtering the needs list by gender, men's ranking of the needs doesn't differ from the general population and women do. Again, differing from the general population is not good or bad; it's just a visual representation of the data.

Satisfaction with the Web - In the survey, we asked respondents to rate their overall satisfaction with the Web as a platform and set of tools (more on this starting on page 27) using a Likert scale from very satisfied to very dissatisfied. This segmentation includes those who were very satisfied and satisfied compared to those who are dissatisfied and very dissatisfied.

Years of Experience - We provided data for those who have two years or less experience writing code for the Web and those with ten or more years of experience.

Type of Front End Developer - Survey respondents were able to pick which type of developer best describes them, and there were two options for Front End Developers. Those who primarily write code using JavaScript and those who primarily use HTML and CSS. We provided the difference in need ranking for these two.

Needs By Country

General Population

	Brazil	China	France	India	Japan	Russia	U.S.
Having to support specific browsers (e.g., IE11).	1	1	1	1	1	1	1
Outdated or inaccurate documentation for frameworks and libraries.	3	2	2	2	3	5	2
Avoiding or removing a feature that doesn't work across browsers.	2	4	5	4	4	3	3
Testing across browsers.	4	3	3	6	2	2	4
Making a design look/work the same across browsers.	5	6	4	3	5	4	7
Discovering bugs not caught during testing.	6	9	6	8	8	7	5
Supporting multiple frameworks in the same code base.	7	7	7	7	6	6	8
Keeping up with a large number of new and existing tools or frameworks.	8	10	12	5	16	9	6
Managing user data to comply with laws and regulations.	9	23	8	13	10	10	13
Understanding and implementing security measures.	14	16	17	12	12	14	9
Integrating with third parties for authentication.	17	13	11	15	11	12	11
Pinpointing existing performance issues.	15	11	14	14	18	13	10
Running end-to-end tests.	11	12	10	11	9	25	12
Lack of device APIs allowing for access to hardware.	10	5	13	9	17	11	23
Outdated documentation for HTML, CSS and JavaScript.	16	8	9	18	7	19	19
Determining the root cause of a bug.	13	21	18	19	13	18	14
Capability of the web to support a specified layout.	19	24	15	16	14	17	18
Knowing what browsers support a specific technology.	12	17	19	10	20	8	25
Achieving visual precision on stylized elements (e.g., buttons).	22	15	20	25	19	15	21
Running front-end tests.	20	20	16	24	15	23	16
Implementing localization.	23	22	23	21	22	26	15
Keeping up with changes to the web platform.	25	19	25	20	21	16	20
Implementing performance optimizations.	26	14	24	17	23	20	22
Making sites accessible.	24	28	21	26	25	22	17
Getting users to grant permissions to Web APIs (e.g., geo-location).	21	18	22	23	24	21	26
Deciding what to learn next to keep my skill set relevant.	18	25	26	22	26	24	24
Finding a community of peers.	27	27	27	27	27	27	27
Fixing a bug once it's been identified.	28	26	28	28	28	28	28

Needs By Gender

General Population

	Female	Male
Having to support specific browsers (e.g., IE11).	1	1
Outdated or inaccurate documentation for frameworks and libraries.	2	2
Avoiding or removing a feature that doesn't work across browsers.	3	3
Testing across browsers.	6	4
Making a design look/work the same across browsers.	8	5
Discovering bugs not caught during testing.	4	6
Supporting multiple frameworks in the same code base.	5	7
Keeping up with a large number of new and existing tools or frameworks.	7	8
Managing user data to comply with laws and regulations.	14	9
Understanding and implementing security measures.	9	10
Integrating with third parties for authentication.	11	11
Pinpointing existing performance issues.	10	12
Running end-to-end tests.	16	13
Lack of device APIs allowing for access to hardware.	12	14
Outdated documentation for HTML, CSS and JavaScript.	15	15
Determining the root cause of a bug.	17	16
Capability of the web to support a specified layout.	20	17
Knowing what browsers support a specific technology.	19	18
Achieving visual precision on stylized elements (e.g., buttons).	25	19
Running front-end tests.	23	20
Implementing localization.	22	21
Keeping up with changes to the web platform.	18	22
Implementing performance optimizations.	13	23
Making sites accessible.	26	24
Getting users to grant permissions to Web APIs (e.g., geo-location).	21	25
Deciding what to learn next to keep my skill set relevant.	24	26
Finding a community of peers.	27	27
Fixing a bug once it's been identified.	28	28

Needs By Satisfaction with the Web

General Population

	<i>Satisfied</i>	<i>Dissatisfied</i>
Having to support specific browsers (e.g., IE11).	1	1
Outdated or inaccurate documentation for frameworks and libraries.	2	2
Avoiding or removing a feature that doesn't work across browsers.	3	6
Testing across browsers.	4	4
Making a design look/work the same across browsers.	5	3
Discovering bugs not caught during testing.	6	8
Supporting multiple frameworks in the same code base.	7	7
Keeping up with a large number of new and existing tools or frameworks.	8	5
Managing user data to comply with laws and regulations.	9	16
Understanding and implementing security measures.	10	13
Integrating with third parties for authentication.	11	10
Pinpointing existing performance issues.	12	15
Running end-to-end tests.	13	14
Lack of device APIs allowing for access to hardware.	14	19
Outdated documentation for HTML, CSS and JavaScript.	15	17
Determining the root cause of a bug.	16	20
Capability of the web to support a specified layout.	17	9
Knowing what browsers support a specific technology.	18	18
Achieving visual precision on stylized elements (e.g., buttons).	19	11
Running front-end tests.	20	21
Implementing localization.	21	22
Keeping up with changes to the web platform.	22	12
Implementing performance optimizations.	23	25
Making sites accessible.	24	23
Getting users to grant permissions to Web APIs (e.g., geo-location).	25	26
Deciding what to learn next to keep my skill set relevant.	26	24
Finding a community of peers.	27	27
Fixing a bug once it's been identified.	28	28

Needs By Years of Experience

General Population

	2 or less	10+
Having to support specific browsers (e.g., IE11).	1	1
Outdated or inaccurate documentation for frameworks and libraries.	2	2
Avoiding or removing a feature that doesn't work across browsers.	3	4
Testing across browsers.	4	3
Making a design look/work the same across browsers.	5	8
Discovering bugs not caught during testing.	6	6
Supporting multiple frameworks in the same code base.	7	7
Keeping up with a large number of new and existing tools or frameworks.	8	5
Managing user data to comply with laws and regulations.	9	9
Understanding and implementing security measures.	10	13
Integrating with third parties for authentication.	11	10
Pinpointing existing performance issues.	12	12
Running end-to-end tests.	13	11
Lack of device APIs allowing for access to hardware.	14	18
Outdated documentation for HTML, CSS and JavaScript.	15	17
Determining the root cause of a bug.	16	15
Capability of the web to support a specified layout.	17	16
Knowing what browsers support a specific technology.	18	23
Achieving visual precision on stylized elements (e.g., buttons).	19	20
Running front-end tests.	20	14
Implementing localization.	21	22
Keeping up with changes to the web platform.	22	19
Implementing performance optimizations.	23	24
Making sites accessible.	24	21
Getting users to grant permissions to Web APIs (e.g., geo-location).	25	25
Deciding what to learn next to keep my skill set relevant.	26	26
Finding a community of peers.	27	27
Fixing a bug once it's been identified.	28	28

By Front End Developer

General Population

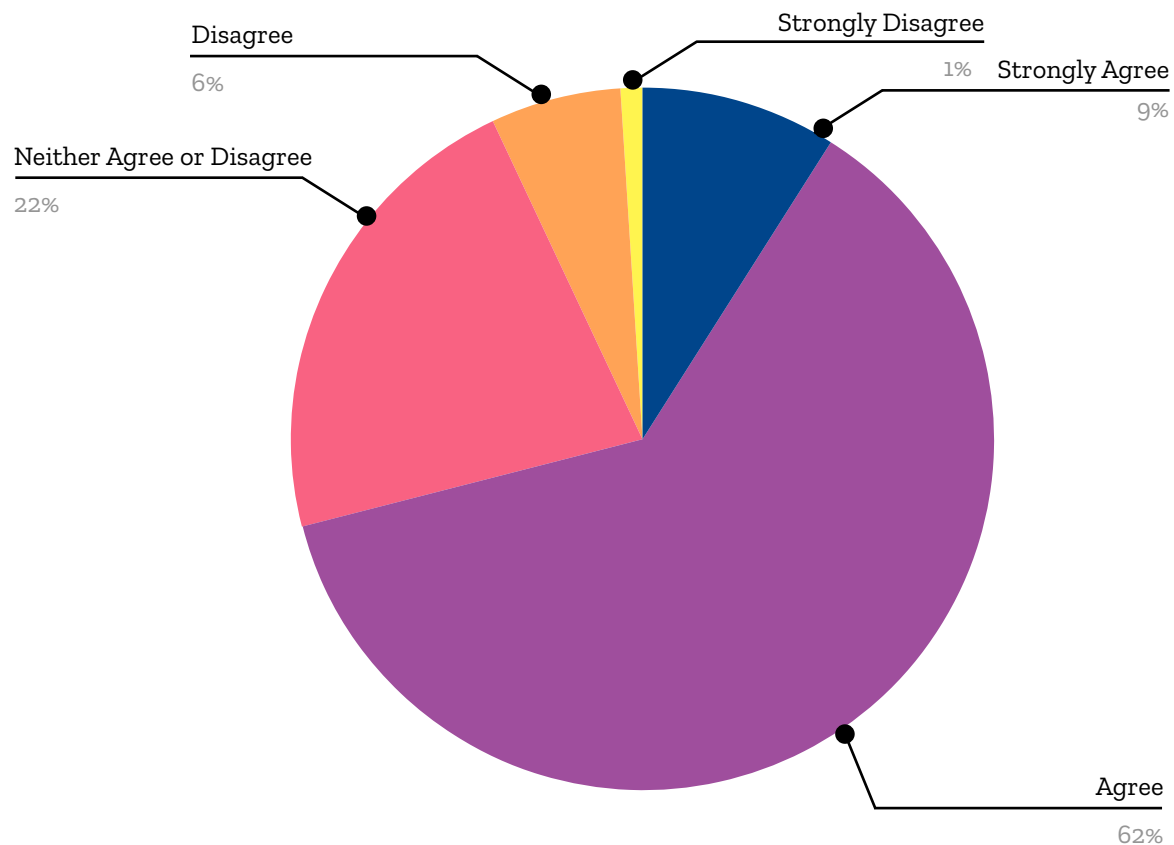
- Having to support specific browsers (e.g., IE11).
- Outdated or inaccurate documentation for frameworks and libraries.
- Avoiding or removing a feature that doesn't work across browsers.
- Testing across browsers.
- Making a design look/work the same across browsers.
- Discovering bugs not caught during testing.
- Supporting multiple frameworks in the same code base.
- Keeping up with a large number of new and existing tools or frameworks.
- Managing user data to comply with laws and regulations.
- Understanding and implementing security measures.
- Integrating with third parties for authentication.
- Pinpointing existing performance issues.
- Running end-to-end tests.
- Lack of device APIs allowing for access to hardware.
- Outdated documentation for HTML, CSS and JavaScript.
- Determining the root cause of a bug.
- Capability of the web to support a specified layout.
- Knowing what browsers support a specific technology.
- Achieving visual precision on stylized elements (e.g., buttons).
- Running front-end tests.
- Implementing localization.
- Keeping up with changes to the web platform.
- Implementing performance optimizations.
- Making sites accessible.
- Getting users to grant permissions to Web APIs (e.g., geo-location).
- Deciding what to learn next to keep my skill set relevant.
- Finding a community of peers.
- Fixing a bug once it's been identified.

JavaScript CSS/HTML

1	1	1
2	2	5
3	4	3
4	3	10
5	5	13
6	7	4
7	6	6
8	8	2
9	11	11
10	9	7
11	12	9
12	10	12
13	13	15
14	14	14
15	15	19
16	17	8
17	18	23
18	16	22
19	23	26
20	22	24
21	20	21
22	21	18
23	19	20
24	25	25
25	24	17
26	26	16
27	27	27
28	28	28

How Developers Felt About the Needs List

Because the Developer Needs Assessment is intended to be reproduced annually, we asked survey respondents whether the list of 28 needs was a fair representation of the needs they experience as a web developer. While most respondents agreed the list was representative, 21.6% neither agreed nor disagreed which means there is room for improvement in the needs list.



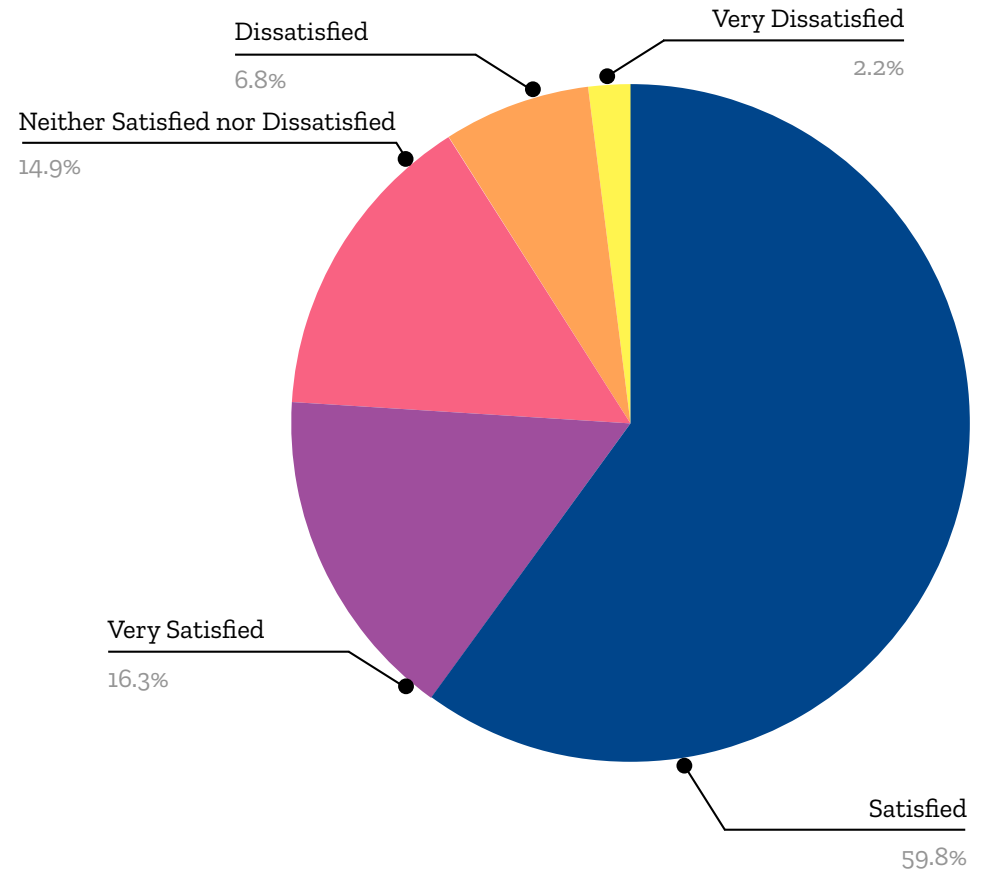
Overall Satisfaction With the Web

Overall Satisfaction With the Web

Through this project, we wanted a way to measure how satisfied web developers and designers are with the Web as a platform. We wanted this to be a repeatable question to measure changes in satisfaction over time.

We asked survey respondents, "How would you rate your overall satisfaction with the Web, as a platform and set of tools, to enable you to build what you need or want?"

We learned that a majority, 76.1%, of respondents are either very satisfied or satisfied with the Web, whereas 9% are either very dissatisfied or dissatisfied.



What's Missing From the Web

What's Missing From the Web

The survey had one question where the response was open-ended. The question was, "What are things that you would like to be able to do on the Web but lack web platform features to do?" This was an optional question, not requiring a response.

We asked this open-ended question because we didn't know what answers to suggest as possibilities, but intend to use the results gathered to inform future iterations of this question.

Of the 28,474 completed survey responses, we had 12,359 respondents answer this question, which is just shy of half. Open-ended questions are difficult to analyze because you cannot be sure how a respondent interpreted the question, and therefore what context to apply to their answer. With that in mind, we went through the responses and eliminated answers that did not answer the question at hand. For example, many responses had some form of, 'non-applicable,' or, 'nothing is missing.' After deleting non-pertinent answers, the remaining responses totaled 9,570.

From there, we selected a random 1,000 answers to categorize into themes.

Aside from a surprising number of responses alluding to the desire to have the Web make a pot of coffee and achieve world domination, developers' wants fell into 109 categories. However, of those categories, only seven had 3% or more of the answers:

- Access to Hardware (12.4%)
- Browser Compatibility (8.6%)
- Access to Filesystem (4.7%)
- Performance (3.4%)
- PWA support (3.4%)
- Debugging (3.3%)
- Access to Native APIs (3%)

Falling just shy of the 3%, at 2.9%, were answers about needs related to CSS. Rounding out the 2% level were answers about avoiding JavaScript on the web platform (2.3%) and needs about gaming (2%).

The remaining 99 categories, each accounted for less than 2% of the 1000 responses analyzed.

Another note on the methodology for analyzing these answers is that since the question was open-ended, we received answers that had a list of things the respondent wanted to achieve on the web, but lacked the platform features to do so. To avoid giving one respondent's answer more weight than those who only provided one item, we took the first response out of the list. In some cases, we could not make sense of the first response or it did not answer the question. In those instances, we moved on to the next item in the list, until we either deleted their response (for not answering the question) or categorized the first answer in the list that made sense considering the question.

On the following pages are short descriptions, which further explain the category, beyond just the title, as well as verbatims from survey responses. We chose the verbatims that best capture the deeper meaning of the category.



Access to Hardware

Many of the answers categorized as access to hardware said simply that. Respondents wrote in answers such as, “access to hardware.” Some responses included more context. Access to hardware APIs was a common response. Other common requests were access to contacts, calendars, and the camera. Verbatims that best capture the deeper meaning of the category:

- Access Hardware APIs on devices, deploy applications with native-like performance and functionality using only web technologies.
- Access low-level hardware on clients’ browsers.
- Better access to hardware. We do a lot of scanning, it would be super helpful if there were an easier way to go from a scan to a canvas, to reading what is marked on a page.



Browser Compatibility

Answers in this category asked for consistency in cross-browser compatibility, and across various devices and platforms. Another common desire was for better testing across browsers. CSS and JavaScript are the two most referenced languages in context to Browser Compatibility. Verbatims that best capture the deeper meaning of the category:

- The same cross-browser rendering, so clients don't complain fonts look good in one browser but not in another.
- Use new technologies as standard without worrying about browser support
- Supporting IE is a hell which we do because we have to, not because we want to.
- Create the same experience across browsers without worrying about CSS caveats. It's beyond infuriating coding in Chrome, turning to IE11 where everything is messed up.



Access to Filesystem

Access to Filesystem could have been a subcategory of Access to Hardware, however the sheer volume of responses asking for access to the filesystem felt large enough to make this its own category. Many of the answers categorized with Access to Filesystem said simply that. Some answers acknowledge that users might need to grant permission before gaining access to the filesystem. Verbatims that best capture the deeper meaning of the category:

- Secure and usable solution for persistent local storage.
- Save files to the user's computer directly (after being granted permission)
- Granting limited access to file system directly from browser. It's highly infeasible but would allow direct interactions if done right.
- Handle the system files (to allow a user to work locally while having an online web application without Node.js)



Performance

Respondents talked about performance in different ways. Some referenced knowing a users' bandwidth capacity. Other referred to specific programming languages, most notably JavaScript. Performance was also measured compared to the speed of native apps. Verbatims that best capture the deeper meaning of the category:

- I would love to see better performance. A lot of stuff is only possible with a strong back-end, because the browsers are just too slow. Performance is the most important factor to improve.
- See how much environmental resources I am using, as a part of performance.
- Native mobile app speed in web apps.
- JS performance is poor. It would be fun to write a Python or Java browser, but unfortunately, this has not changed.
- Accurately determine a user's current bandwidth capacity.



PWA Support

Most responses categorized as PWA Support were about the lack of support and the desire to see full support. Not all vendors offer the same support for PWAs, which creates frustration for developers. Another common answer is having PWA as an option in app stores. Verbatims that best capture the deeper meaning of the category:

- Full PWA support from all vendors. I would like to develop web-view-based apps publicly available on the web and supported on all platforms.
- PWAs are getting better — we're still waiting to have a similar experience on iOS as on Android. Also Firefox does not offer to "install" a PWA.
- Web can do most of the things which native does but web app is not commonly used like app, PWA should be there in major app stores as a category.
- Have installable apps appear in app stores.



Debugging

Respondents who wrote answers about debugging often wanted better tools to make it easier and more convenient to debug from capturing bugs to resolving bugs. Verbatims that best capture the deeper meaning of the category:

- Better capture and report a scenario that leads to a bug in the user's browser.
- More convenient way to debug.
- Modify the source code while debugging.
- Mobile debugging.



Access to Native APIs

Answers categorized as Access to Native APIs were similar to Access to Hardware or Access to Filesystem in that developers wanted integration with a device OS. Also, having web apps behave more like native apps. Verbatims that best capture the deeper meaning of the category:

- Interacting with other desktop applications.
- System OS integrations.
- Use an expanded set of platform provided components.
- I would like Web Apps to be more native.

Technologies



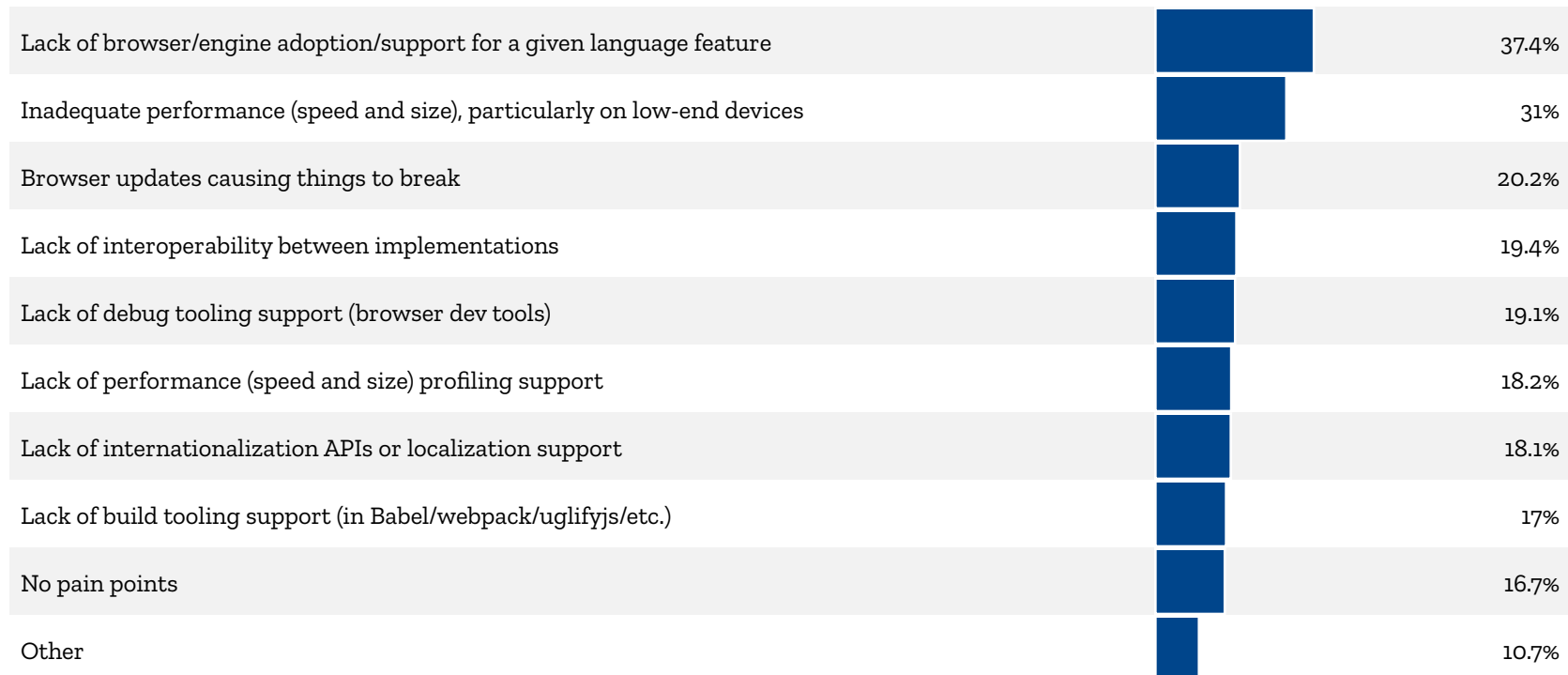
Programming Languages

As core technologies for the web, we wanted to understand developers' pain points using JavaScript, HTML, CSS, and WebAssembly.

JavaScript

The biggest pain point for JavaScript was, "Lack of browser/engine adoption/support for a given language feature." Though, 16% of respondents have no pain points with JavaScript.

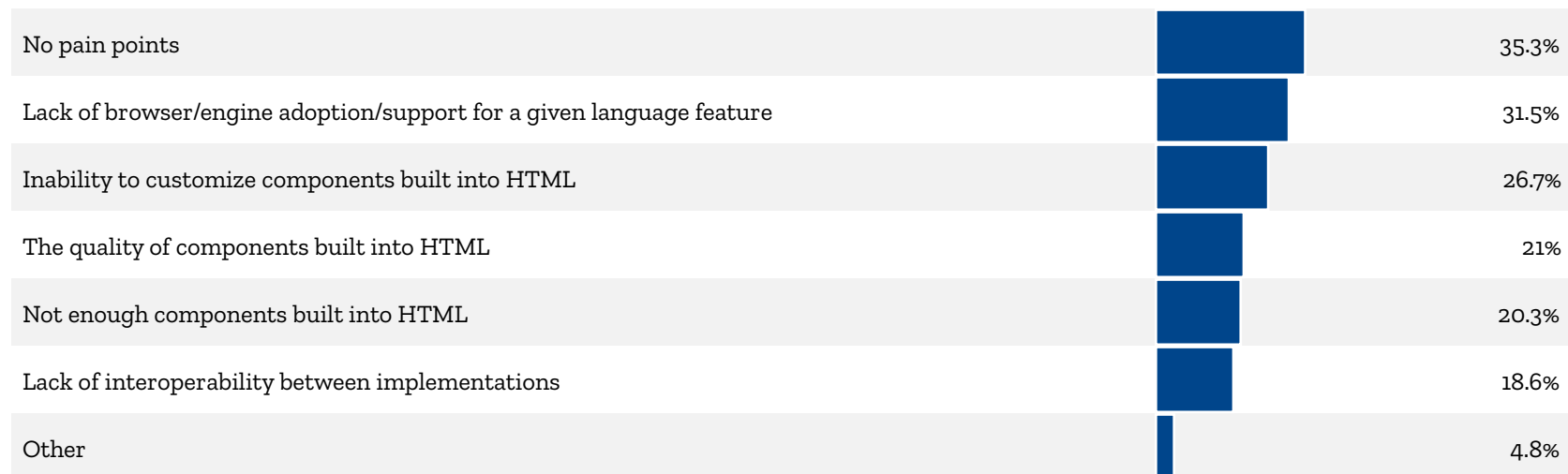
JavaScript - Biggest Pain Points



HTML

HTML seems to be the shining star of these languages as roughly a third of respondents (35.3%) have no issues with the language. That said, the biggest pain point with HTML is a lack of adoption and support for a given feature at 31.5%.

HTML - Biggest Pain Points



Nearly half (44%) of respondents said their biggest pain point with CSS is challenges creating the layout specified. Since CSS is the language to style webpages, the results suggest that achieving a desired layout is difficult to do using CSS. The results can be interpreted in different ways so further investigation is needed. One way to interpret this response is that CSS is causing developers a lot of grief when using the language to create layouts. Another way to interpret this is that developers are trying to do their best using CSS to achieve a web-friendly layout from a design that wasn't intended for the web.

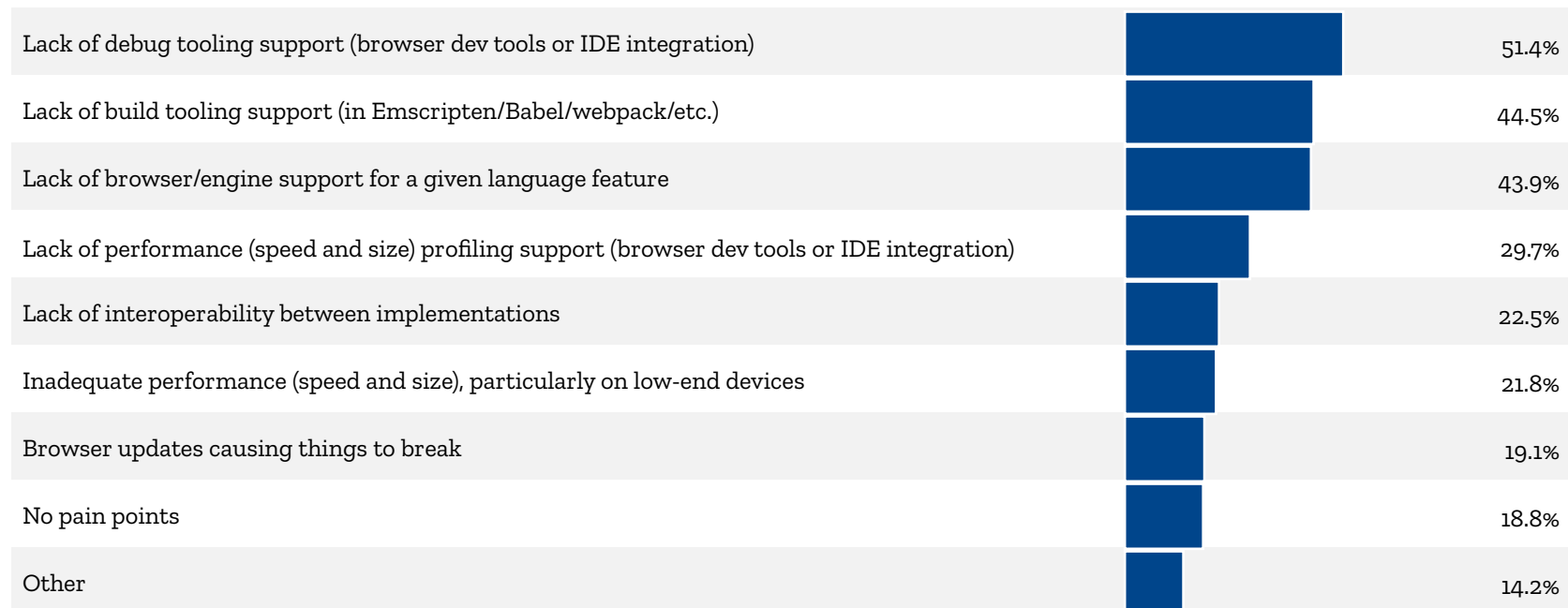
CSS - Biggest Pain Points



WebAssembly

WebAssembly is a new technology, so only a handful of respondents, 851, which is just shy of 3% of the completed responses, had experience enough to respond. The results below are from a much smaller sample than those for JavaScript, HTML, and CSS. The largest pain, with 51.4% is a lack of debug tooling support.

WebAssembly - Biggest Pain Points



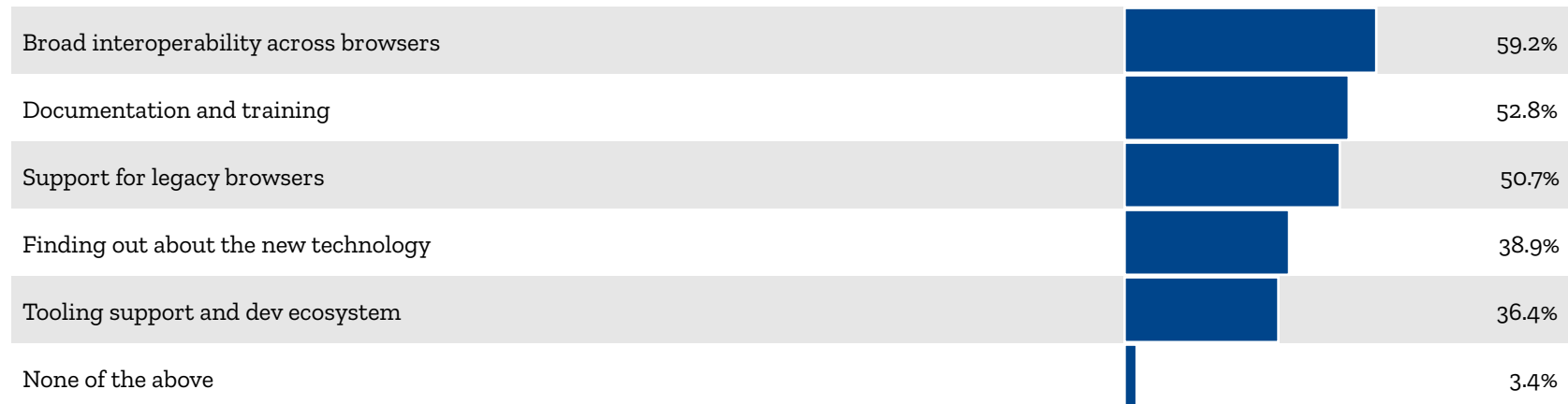


Adoption of New Technologies

Adoption of New Technologies

The biggest barrier developers face when adopting new technology is broad interoperability across browsers. This is in-line with the top ten needs where four had to do with browser compatibility.

Adoption of New Technologies



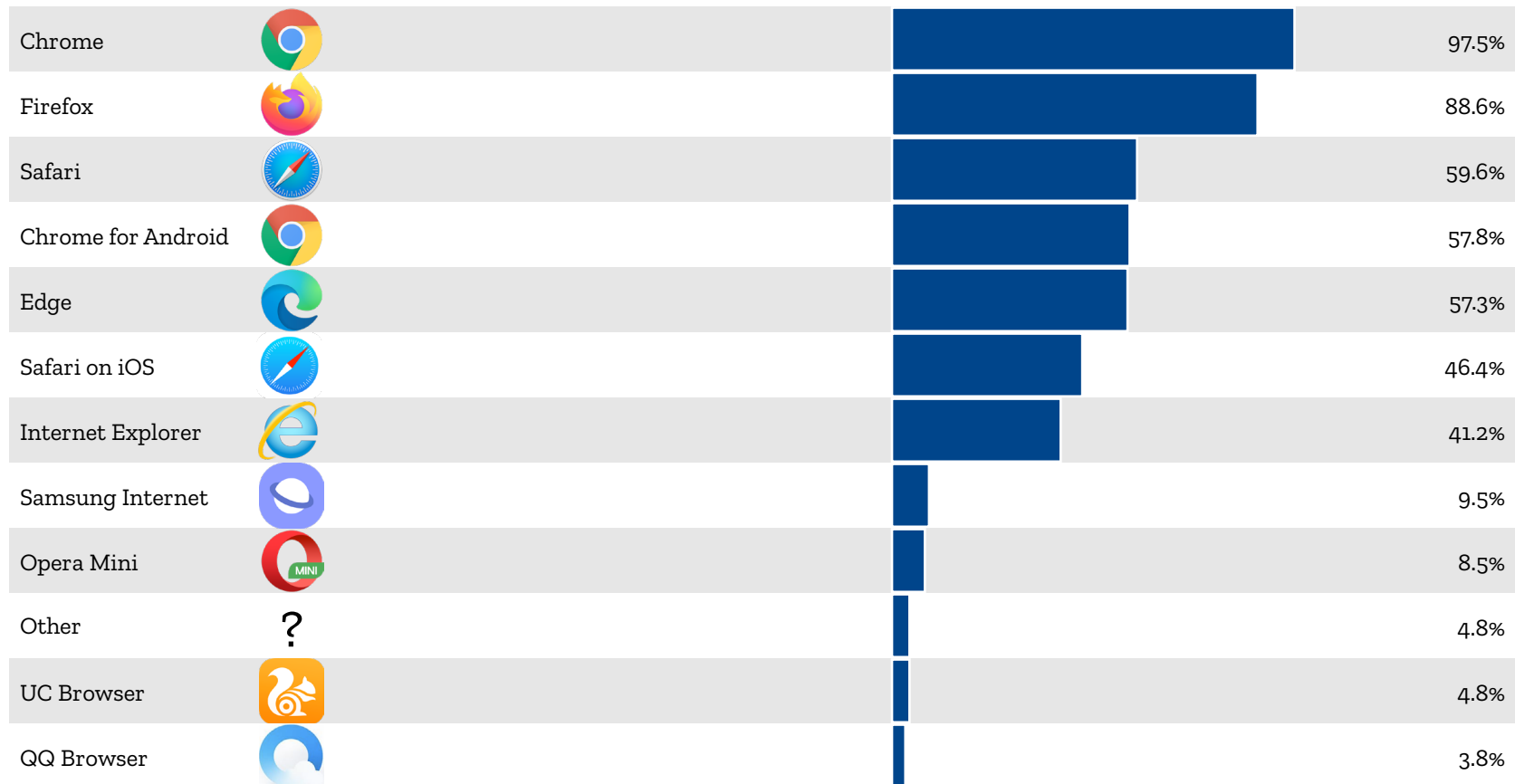


Browsers

Browsers

Chrome and Firefox lead the pack in terms of browsers developers support, 97.5% and 88.6% respectively. Third is Safari at 59.6%.

Browsers Developers Support

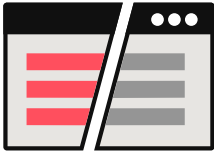


Pilot Interview Findings



Enhanced View of the Top Ten Needs

Browser Compatibility



In the full list of needs browser compatibility was the dominant theme. Four need statements made the top ten:

#1 Having to support specific browsers (e.g., IE11).
#3 Avoiding or removing a feature that doesn't work across browsers.

#4 Testing across browsers.

#5 Making a design look/work the same across browsers.

Of all the pilot interview findings, browser compatibility was the topic discussed using strong language especially if IE was at the heart of an issue.

While IE11 is decreasing in share as Microsoft shifts users to Edge (and the forthcoming Chromium-based version), it continues to be a large source of frustration to developers due to being over-represented in enterprises, legacy line of business apps, and certain industries. The prevalence of users accessing the web through IE is beyond both web developers' control and Microsoft's. In some cases, it's the management of a company that dictates what browsers should be supported, ignoring data that suggests users are accessing a site through browsers other than those mandated.

"I don't know why people still use IE. I find [browser compatibility] very frustrating, especially as a front end designer. IE is such a curse my god, why do people use it?"

"Browser compatibility, by and large, shouldn't be an issue and for us specifically is a nightmare...we have so many people accessing our application from their office, which means they are running IE11. Which is something that I don't care about, except I needed to code for it to keep my job. It's just a hair-pulling frustration...it is tragic."

"[Browser compatibility] is important, but I think people put importance on the wrong parts of it. Everywhere I've worked they say, 'It has to work on IE11,' and I'm like, 'Really because I think it needs to work on mobile Safari but not IE11.' It's frustrating trying to get stuff working on IE...the manufacturers have said stop using that."

"...the most annoying thing for me is...browser compatibility, to feel like something is simple enough to where, it shouldn't render differently in Edge, but it does. That's the big one, especially because [in my industry of] oil and gas, most people aren't very technically inclined..."

"... this [doesn't] strictly [relate] to accessibility it more relates to browser compatibility. But Edge is not particularly accessible. It's optimization for JAWS, which is the only popular software for professional blind users is abysmal...it's recommended that Edge is not used at all, which makes users divert or revert to older versions of IE, which is a mess."

Browser compatibility is getting better though, at least some see a light at the end of the tunnel.

"There are not many problems we have with browser compatibility... every once in a while, we're supporting IE11, and that's the only old browser we're supporting. Even that we're noticing that less than half a percent of our users are using IE 11..."

"If you think about the amount of weird tricks [you had] to pull off ten years ago to make sure that your website looks good in every web browser, and the amount of things you have to do now is kindergarten level. You just have to do a few things every browser now supports a decent enough level of features."

While it might seem easy to point fingers at IE, all browsers have

their quirks, and developers have had to learn those quirks and adapt to make their site work well enough after tediously testing across browsers and, in some cases, having to remove a feature.

"...it's hard to achieve cross browser compatibility so everything works...it's a lot of testing...to do with all these browsers, which [isn't] a lot of fun..."

"Modern browsers are supposed to support same-site cookie attribute. But it turns out the latest version [of Safari] has a bug...it not being compatible with this specific version of Safari means that we cannot use them in general...all this is really frustrating."

"...you have to make sure that whatever you develop works as expected. There are fringe cases, say, if I were to make a web form [and it] works fine and dandy in Chrome, but for whatever reason, maybe it's not behaving in another browser. Then you go down the rabbit hole of trying to debug or trying to research what the root cause is. Sometimes it's due to an unsupported API or feature that's being used, and that being the hallmark feature of the app..."

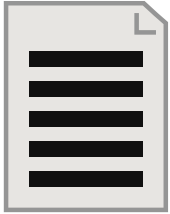
"...I understand the need to cater to browser compatibility, for example, you can pull up the metrics of any given project or site, and then you can look at the technology or the browser that's being used to visit your site...you can make a case to say, 'Well, a lot of our users are using a specific browser, we need to ensure that our project looks [in] the best light. That's what can withhold certain features that you implement on your project.'"

Though not captured as a need statement by itself, at the crux of browser compatibility is the users' experience. If it doesn't work across the browsers your users use, then they aren't going to be accessing your site, which has business implications.

"I think it's very important that your audience can use your product no matter what device or browser they're using, they should be able to use your service. I think this is very important to be able to deploy your product across a range of browsers."

"[Browser compatibility is] an implicit point where it is so important because everyone should have the same experience."

Documentation



Survey participants rated aspects of documentation as the second most frustrating need:

#2 Outdated or inaccurate documentation for frameworks and libraries.

Documentation is so important, and hence leads to such high frustration because it is one of the things that gives developers an accurate picture of how things should work. It is also one of their go-to resources for overcoming frustration, which is why developers find inaccurate or outdated documentation so frustrating.

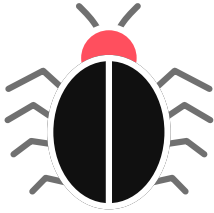
"...[documentation] really gives you the right picture of how things should be working...if you aren't sure how things are supposed to work it's so hard to get them right or it's so hard to identify possible issues with what you're doing."

"Documentation is [the central thing for helping me overcome frustration] partly because...you think you knew something but then after reading up on said feature or said documentation, it's like, 'Ah, I should be implementing it this way,' rather than a learned know-how, so to speak."

Documentation was one of the reasons why developers dislike frameworks. They find there is not enough information about how to use a framework optimally.

"...nobody tells us what is the optimal way to use [a framework]. It took me a lot of time to understand how to format my bootstrap file...this is the tree I have to make to make sure that it is optimal and it can be used by multiple different devices. That is something nobody tells us, exactly...That's something I really like about Vue.JS they tell how to use it optimally."

Debugging



In the full list of needs, debugging debuted at #6:

#6 Discovering bugs not caught during testing.

The frustrations developers experience with debugging are more about the time it takes and the fact that if you're debugging, then something is wrong. Trying to find the bug and how it manifests in a browser is also a source of frustration.

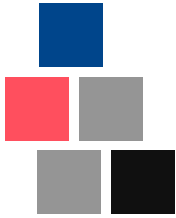
"Debugging is always irritating...[It's] a never-ending process. I can have a hundred test cases, I can debug [them all]. I'll come back, one user will call, 'This particular thing is not working,' so I have to go back again, and realize, 'Okay, this is again a problem because [of] something so debugging is frustrating."

"[The] level of frustration [for debugging] would be high up there... debugging a layout it isn't so much of a mystery if it's HTML and CSS...in the context of JavaScript I feel like you're always debugging throughout the process, making sure that it hits certain breakpoints. If I hit all the marks...for me, then I'm satisfied. Debugging should definitely be more along the lines of ops [and] code versioning because I feel like that's where a lot of that happens in tandem."

"...the only reason that [debugging is] frustrating is because of the multiple places to go to do it when you're actually having to confirm how it appears in a browser."

"Debugging is just frustrating because if you are already debugging, you've done something wrong. You shouldn't write to the point where you debug, you should just test so that you don't waste time debugging...it's important because if you are debugging that means there's something that's not working right that's bothering you. Hopefully, you catch it before it's live."

Frameworks



One of the unknowns stakeholders had about web developers is why they like frameworks — what about frameworks do they like? It turns out, the relationship with frameworks is more of a love/hate meaning developers don't necessarily like frameworks. In the survey, frameworks made the cut in two of the top ten needs:

#7 Supporting multiple frameworks in the same code base.

#8 Keeping up with a large number of new and existing tools or frameworks.

Some developers envy the server-side world where there's more stability and only a few leading frameworks, whereas, on the front end side, developers feel they are getting whiplash just trying to keep up with the many frameworks that exist.

"I think it's frustrating that there's no good way to manage your dependencies of applications on the web, compared to Maven in the Java world, which has been a long, steady, common tool which has been available for many years."

"It could very well [be]...frameworks are entering a maturity where you'll have a few, React and Angular, in perpetuity, the same way that the server-side frameworks have settled down into three or four that everyone uses. But it's not there yet, and it's a lot of whiplashing and eye-rolling...but it also keeps me employed."

Some developers try to avoid frameworks altogether mainly because it's too hard to keep up with new and existing frameworks.

"...the pace of change in the framework arena has a high level of frustration, so I try to avoid them."

"I'm not a big fan of frameworks... I don't feel like they get you that much...[frameworks] always look more complicated than they ought to be [but] they're presented [as] the greatest thing since sliced bread. I'm not convinced."

On the flipside, frameworks offer efficiencies allowing developers to reuse something that's already been solved for.

"I use the term framework to mean a new set of modules that have been prebuilt to trivialize the tasks...if there's a complex series of blocks of code that you need...in order to get around a certain issue of rendering something or processing something, then somebody is going to create a library to make that more efficient."

"React is a joy to use, Angular is a very solid one. You can pick and choose whatever you want now. Each one of the big ones, it's really well made...allows you to do whatever you need. [Frameworks are] important because it does allow many companies to create something good with not much effort...it's to the point where it's nice, but it's somewhat in the background because everybody got so good at it."

Participants mentioned browser-based APIs, web components and object-oriented JavaScript as potential solutions or methods for reducing the need for frameworks.

"...in the last few years, the browser's did enormous development...all these browser APIs are good enough in order to avoid specific platforms. What I've learned from the last 10-15 years is that every, let's say, half a year, some kind of framework is hyped. People tell you, 'It's the best that they've ever seen.' Half a year later, something else comes around, and you have to change your code base. The most reliable project I've seen that works over many years, just purely use browser based API...I resorted to using pure browser, cross browser supported

APIs...they work for many years"

"...a lot of the problems that frameworks are trying to solve are artifacts of the non-object-oriented JavaScript, the older JavaScript way of doing things... it's my hypothesis that a lot of that goes away if you just take a more modern design approach to developing for the web,... I've been playing with web sockets...it's just a shared object. That's one way of doing it. [Another way is] with remote method indication."

"I believe in web components. It will be nice if [frameworks] are not going to be as important as before."

Privacy



In the full list of needs tested in the survey, there were two need statements that related to privacy. One made the top ten list, coming in ninth:

#9 Managing user data to comply with laws and regulations.

Depending on the part of the world a developer resides, privacy is seen as getting better (GDPR) or worse (Australian Assistance and Access Bill). These laws and regulations have implications on both protecting a developer's privacy as a user of the web, how they approach their work, as well as their career trajectory. When laws and regulations were mentioned in conjunction with privacy, there was a belief that maintaining privacy, whether their own or users, is beyond the scope of technology, meaning Governments need to step in to hold technology companies accountable for their use of private data. However, in some countries, it can be the Government who is exploiting data.

"I saw [privacy] more in terms of [the] end customer instead of me [the] developer... me as a user I'm feeling much safer now, in terms of privacy than a year and a half ago because...if you're in Europe, GDPR is going to keep your data in a much better place."

"It feels like many services and apps...are designed to mislead people that don't know about the technicalities. I think [privacy] is super important and it's frustrating that it's hard to protect users, but I think this should improve with regulations...with laws like GDPR from Europe. I mean, it's the only way really. Tech solutions like the Do Not Track setting in your browser...I'm sure people wanting to track you [will] exploit your data and everything. They will always find ways unless it's really regulated. I don't think that technology alone can protect people."

"...the impact [of GDPR]...we have to do the work to ensure that when people ask for us to delete the data, we need to ensure that we are deleting all the data that we have from them...that means deleting anything for logs, making sure that everything is deleted from all databases...before this we were still doing that kind of work but we didn't have the same pressure I suppose...and it wasn't clear for users that they had the right to exercise that..."

"privacy...that depends on the company where you work honestly. In Europe, [privacy is] much easier...because you have to specify everything you do and they have to be able to delete everything. GDPR helps so much."

"[Privacy] that's a big one...we just pass the Assistance and Access Bill, a huge blow to the Australian security and IT industry...you can be instructed to make changes to your app so that the government can get access to it, and you're not allowed to tell anyone. It's at the forefront of everyone's minds here...if I were to go look for another job, there's plenty of companies who said they don't want to hire devs in Australia anymore because we're suddenly a security risk."

Beyond the laws and regulations surrounding privacy, the pilot interviews revealed that privacy is personal. One thing top of mind when discussing privacy was protecting their privacy as a user of the Web, followed by the role of privacy in their work. When it came to protecting their own privacy and the privacy of their users, developers felt it is hard to do. There is a belief that maintaining their privacy is their right, and it is something they value, so they try to keep their users' privacy safe too. Maintaining privacy comes with tradeoffs in terms of the features you're able to access.

"You're asking me why privacy matters? It's my right."

"When I use the web, I feel uncomfortable. I ask myself, 'Is this secure enough or will I have to use incognito version?' I'm not sure how much I can protect my information and security on the web...what is important for me is that I keep these values in doing my development."

"...it's quite frustrating to protect my privacy on the web, but it's really important for [users] to keep their privacy safe. I think most developers should consider this when they build a website. As a consumer, I think it's really important to keep my privacy safe. I don't want to give my specifics or I don't want to receive any ads based on my activity on the web."

"...being a German [privacy is] very important to us...I think this is the hardest part to achieve typically because you want to have all these cool features, want to use all these platforms, so it's hard to achieve privacy, but still, at the same time it's very important to us...this is one of the most difficult issues and that's why I'm using Firefox and not Chrome for instance today."

"...I'm thinking that mostly as a user...you install an app and it wants permissions. I'm always like, 'Why does it need that, it doesn't need that,' you don't want to read [the privacy policy] you just click okay...it's a problem...you can't trust these apps...they can access all your data and all with any webpage you visited...I find that very frustrating as a user it ought to be more precise about exactly what it's going to do and what it won't do...it's just creepy."

Security



Rounding out the top ten needs is security, coming in at tenth:

#10 Understanding and implementing security measures.

What did we learn about security from the pilot interviews that lead to this need? First, let's look at how people defined security. Developers used terms such as authentication and privacy in their definitions.

"Security means that no one can get access to any part of the system that they are not authorized to access, and no one person can see any data from the system that they're not allowed to access."

"[When I think about security] I think about authentication...how do you successfully guard against malicious login or malicious attacks on things that are supposed to be secure."

"...[a site] is considered secure [if] it's working in the way you intended, respecting all the access rules...when your users are aware of what's happening with the data, [and] when the data users submit is only used for what they intended [it] to [be used for]."

"Security to me is like the authenticity of data, authentication of users, and being sure that the other guy I'm exchanging data with is the guy he pretends to be...it's very important to have trust so that your users know that they receive data from you and send data to me. I think authentication here is key."

"Privacy is about keeping people's data safe whereas security can have further implications like your app being brought down or somebody hacking it [so it] no longer does what you're expecting it to do. Like proper password protection and stuff to stop people's accounts

from being able to get hacked, and then that information being used for other stuff...so privacy is a part of security."

Beyond the definitions of security, the main thing we heard is that web developers don't know enough about security which creates a feeling of concern and frustration that someone with more knowledge can find and exploit security flaws. This feeling causes stress, especially because security is viewed as very important.

"Security is one of those things that I'm sure we should be doing better on, we haven't had any issues yet, but all you need is one issue..."

"I always have concerns that there are security issues that I haven't figured out yet. I've thought very hard about whether or not it's possible for malicious third parties to authenticate as someone else...I haven't figured [it] out yet but I feel like I'm not quite smart enough in this area, to know whether or not that's possible."

"...[with security] I always have this fear...I use JWT to authenticate my request after initiating. I always have that doubt that it's not unbreakable...a good hacker might be doing that...I think there are a lot of things to be learned and steps to be followed. I have not done that. Maybe it's my ignorance on that side."

"Security is very important for any of our customers at any time for anyone to be honest...you have to make sure that [user] data is contained with us...applying security it's very frustrating because I myself am not very good at security and I personally feel that there are not enough online resources that actually explain why your security might be flawed or what are you doing wrong that might lead to flaw in security. Mostly, I don't really understand security."

We also heard that the ways to implement security feel like a hack

in and of themselves, which is a result of the evolution of the web. As security became important, new ways of ensuring security evolved out of necessity, but those were reactionary developments.

“Security is so important...having bad security can be catastrophic for your business and for individuals. It’s one of the most important things right now...every other day there is some breach and a lot of user data is exposed. It’s so bad...it’s so frustrating because it’s so hard to get it right. There are so many things on the web that are broken from the beginning because the way that the web has evolved...it’s really hard to get it right. Many of the things that we’ve used in the past for security to protect things like tokens...they feel like a patch or a hack on top of the whole system...feeling that it’s impossible to get your app to be truly secure [because] that doesn’t exist. There will always be vulnerabilities in everything in protocols and fundamental libraries...it’s so hard and so important at the same time.”

The third thing we heard about security is a two-way relationship with users. If you get security wrong it adversely affects users. However, you may implement the best security features you know how to, but if users access a site through passwords, as a web developer you have very little control over whether or not they choose a secure password.

“Security is essential...it requires so much attention, and it can be frustrating...because it’s something that you don’t want to get wrong. It’s going to stress you out because getting it wrong might be a really bad thing for your users.”

“...I’m trying to make sure that people [use] better passwords for our system...we give them logins for the production server and also a staging server...‘123’ is their password. I have to make sure that they change that but sometimes I don’t know what password they’re using because it’s their password, I’m not supposed to know it. What we find

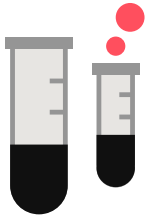
is that we’re only as secure as the passwords that our users use.”

As you can see with the perception that security feels like a hack and being tied to customer logins, security is closely related to needs or frustrations that might arise due to the pace of change of the web and authentication.



Enhanced View of the Remaining Needs

Testing



During the kickoff meeting for this project, stakeholders shared their thoughts on what was already known about testing. What was captured is that automated testing is hard to get started and there are many different types of testing like unit, integration, and functional. What was unknown was the types of tests developers are running and how or if that causes frustration.

The pilot interviews revealed that some developers feel that it's frustrating to keep track of all the available tools to conduct a proper test and automated tests are not always an appropriate method for the web.

"The frustrating part [about testing is] not being aware of the tools available to conduct a proper test."

"The web is not an automated test driven place because there's too much interdependency between JavaScript, HTML and CSS. I didn't really see the benefit of doing automated testing here...typically you react to user events and then change something on the UI...I think that is better tested by a human...while back end code that interacts with databases, servers, and stuff like this, where the logic is pretty clear that's easily testable using unit testing or something..."

"I don't automate my test normally like you're supposed to...I like it to run on both the server side and the client side, and I can test it interactively. It's just not always possible to do things to automate the test...it's not always clear how to do that, and how to check the result...if I can play around with it, that's the way I like to test things. And it can be frustrating, it's always frustrating when things don't work."

Participants felt testing can be frustrating because it can take a significant amount of time to write certain tests and the ease of writing

those tests is related to a developers' experience with testing.

"A couple of reasons come to my mind immediately [about why testing is frustrating]. The front end testing it's not always easy to do unit testing. That's what I have felt. I have not written extensive test cases before...I know the importance of test driven development and we should write code for testing, and then only write the application so that it won't fail, and those kind of things. Also, I think [it's frustrating] because I haven't done it that much. I always used to manually test my code..."

"Some projects you might not have time to do the unit tests, there is a time factor involved where you don't have the time to write first the test cases, then you write the code that is failing, and then you have to make it pass. We don't have that kind of time for the project, because it's a quickly moving product."

"We also do system tests that are harder to write [than unit or integration tests] because there are more things to consider [and more] time consuming to write when considering all the interactions with the browser. That's why testing is not zero frustration."

Opinions did differ, slightly. On the flipside of not having the time to test, is the belief that taking the time to run tests can reduce stress.

"Wow, if you're a developer, [testing is] your bread and butter. It shouldn't be stressful to test, it should be something that you want to do. Because you know this is going to actually reduce stress."

The types of tests developers run varies as do the tools they use to test.

"[I do] unit testing, but it's component level testing. A single compo-

nent testing TypeScript and HTML together...we used to test JavaScript, whereas now we try to test the whole component together...the back end devs do some kind of back end testing. We'll run automated end to end tests, you login and go through a flow on a page. We were using Protractor and Selenium. Now we're using Test Cafe and we do a heap of manual testing. We have as many testers as we have devs, and they do all of the cross browser testing, cross device testing, all that kind of stuff."

"[I run] as many tests as possible. I'm a true believer of the green-red testing, then integration tests to make sure things work, and then use Cypress or Puppeteer...maybe your QA is going to use Selenium...you'll have to cover the critical path of your website, at least with end to end testing...there's so much choice...you can test it by checking the values, you can abstract other things in a snapshot testing. As long as you are able to check your volume on your logic, and then check that your components are well structured, you should be in a good position to iterate quickly without worrying too much about what could break."

"...on the testing side, all of the angular testing tools they just flatten out all of your asynchronous code, and magically make it synchronous so that you can understand it and follow the flow more easily."

Authentication



Authentication was often described as difficult and frustrating for a variety of reasons, but mainly due to the complicated nature of it and the threat that exists if you get it wrong. However, authentication can be easier if a developer is willing to divest control to a third party.

"It's hard to write a secure API for mobile apps to authenticate against. There are flows but they are very convoluted. [It's] not nice for users and not nice for developers to implement. The whole authentication mechanism with passwords and everything that feels like it should be much easier, especially with mobile where you have other ways to authenticate...like using your fingerprint or using your face. I hope it improves with standards, but it still feels very frustrating in the sense that is so hard to get it right."

"OAuth2...it's a complicated sort of thing...the thing that bothered me about it more than anything was that [logging in with Google or Facebook] even though they use the same app, the code that was used [was] messy. Even though they were using the same protocol, they weren't really the same, they were just really different...I started downloading...a piece of suggested code to run on the server...even though they're using the same protocol they were so different as far as actually setting them up on my server and how to handle it...it was just confusing, it all [should] be simpler than that."

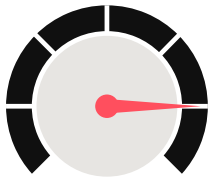
"I think the whole the whole notion of authentication and authorization it's just fraught with difficulty...you've got people out there all the time, they're trying to screw with it, trying to break it. It's difficult and frustrating and important."

"Authentication is generally a nightmare...I'm not that comfortable when there's opaque layers of abstraction, and authentication very rapidly becomes like, 'Oh, there's this encrypt hashing algorithm,' I

don't know what that is and frankly, neither do any of my buddies who actually got bachelor's degrees in computer science. It rapidly becomes impenetrable save for certain people. And that's even setting aside the issue of, 'Are you storing this as a cookie, or are you storing this as a JSON Web Token?' Well, that's a new technology that has a ton of pointy edges. And now the tide is moving against JSON Web tokens..."

"Setting up authentication is either a bit of a nightmare if you're doing it yourself, or it's incredibly simple, if you're willing to just divest yourself of control over everything, just a leitmotif, a theme in web development."

Performance



Performance was not often mentioned as one of the top items that causes frustration, however, it is something that developers find important. It's important because it directly impacts the users' experience on a site. Performance also ties back to a broad view of accessibility where a site should perform across a range of devices.

"Performance can affect a lot of users' experiences [and whether] they stay on the page...they need to see something...with information as soon as they land. It's one of the things to keep the user on the site or the application."

"Performance that should be your first things together with [privacy, testing, tools, and accessibility] because you want the user to be able to reach your website anywhere, no matter the connection no matter the device."

"[Performance] is closely tied to accessibility in a lot of ways, which makes it quite important. If your app is huge and slow then it's annoying for someone on a brand new MacBook, but it's unusable for someone on a low end phone, and wasting people's bandwidth and download and stuff. So it's important in that respect..."

"Performance in terms of user experience, I think it's very important to ensure that your app or your project is performing the way [your users] expect it to..."

"...the page should load quickly and shouldn't keep the user waiting."

Developers referenced performance as a source of frustration because it's hard to find the time to make a site perform well on low powered devices or difficult to pinpoint what is causing the performance issue. This fits with the survey results where, "Pinpointing

existing performance issues," ranked as the twelfth need, out of 28. Another reason performance was mentioned as a source of frustration is the nature of the business itself. Businesses that experience rapid growth in a short amount of time strain systems and businesses must adapt.

"...it's hard to find the thing that makes your application slow. And it's hard to debug, sometimes how or what affects performance. We have to isolate everything and try to iteratively find out what's degraded performance."

"I don't feel that frustrated with performance in general, I think that are a lot of tools both server side and client side that help...in many cases it's just hard to find the time to do the work and to make sure that your app works on not as powerful devices."

"We're doing a lot better. Six months ago, we were doing a lot worse with performance. The problem was that, [and] this is a good problem to have, we were constantly getting new [users]. We're probably about 100,000 unique [users] every month [whereas] four months ago, we're probably about 30,000. We've tripled in the last three months. That has put a lot of strain on our systems. We've had to work hard to... parallelize our computation."

Developers referenced many server side and client side tools that help them resolve performance issues. Most often referenced were network debugging tools, developer consoles, Lighthouse, and New Relic tools.

"...with my current app that I'm working on, the performance is so bad that I just run it through Lighthouse, and that's enough to say, 'Hey, it's taking eight seconds to load or 20 seconds to load sometimes... there's some pretty obvious things that we can do.' In my previous

place, we got beyond that, where we needed to look more into the details of what we could work on, again Lighthouses was mostly what we used because it does give you a breakdown of stuff like that. There's other Google tools that we've been using that I do not recall the name of...we've got monitoring and stuff that I can look at. Everyday, I get warnings, 'Oh, the app is taking more than two seconds to respond...yep, that's pretty standard.' Those are driven from New Relic."

"I'll go into the developer console and look at the network tab, and see how fast things are loading, how many round trips it's making just to load the first page...you can fix by bundling stuff together into one file that loads instead of having to make twenty different trips to the server to get the pieces to put the page together...there's a lot of different little tricks that you can do to make things load more quickly."

During the kickoff, stakeholders had questions about what it means to be performant. The pilot interviews did not reveal any one measure, instead it was unique to the developers and the types of projects they're working on. Developers define their own performance metrics as needed in a way that made sense for their work.

"I measure [performance] in terms of payload, time to interaction, and amount of requests. Of course, this needs to be set to comply on rules that change from project to project."

"In my case, I [measure performance] on what is an acceptable response time...since I work mostly in the server. I just look at graphs and try to see if most requests are within acceptable response time."

"Well, of course, you will always [have] some kind of [performance] metric...sometimes it's very easy. If you have a video conversation...you need to have some very hard limits on round trip or on frames per second. I typically try to define [a] performance measure, which gives me

an indication of what good actually means...I try to always measure it in some way."

"[When] we're talking about the entire system as a whole, they are different aspects of performance...from a web developers point of view, we need to make sure that [our users]...believes that, even if it takes longer than it should, [it] doesn't feel like it takes a very long time."

Localization



The frustrations developers experience with localization expand beyond the traditional notion of accounting for the visual impacts of different languages, although that can be a source of frustration. Localization can mean understanding cultures and preferences in those cultures in terms of how people formally write their names so that you

can account for that when designing input fields or understanding date and time format preferences. Getting these things wrong can lead to a bad user experience. Localization is also seen as something that is not just in the realm of responsibilities for a developer, it's something developers and designers have to work on together.

"Most of my work has been aimed at Australian customers, [but a] related thing we did deal with at my last job was people's names and things like the idea that everyone has a first name and a family name and that they should be in that order, which is not even a little bit true...It's a localization [issue] because the people were in Australia but they were from somewhere else. It's a problem because we're doing identity matching so we need to be able to match people's names across different identities. Basically, the answer was, 'If your name isn't in a standard format, you have to go into a shop and deal with it,' which was not ideal...the people who originally designed the system were all white, Australian men...because it was built that specific way in the first place, it was hard to change it later. Whereas if they had people from more diverse backgrounds to start with, then it would have been built in from the beginning instead of tacked on later."

"[I was] using an app today that wouldn't let me change the date out of American format. So I'm really frustrated about localization. That was as a user...It's so annoying, especially living in Australia. You're using stuff from overseas almost always, and if you can't localize it, it's really noticeable all the time...it's super frustrating, because it's just like, 'My God, everyone's not where you are.'"

"Localization is just hard to get right. It's not just a developer issue it's something that developers and designers have to work together to manage. It's just stressful because, in many organizations, there is not a real care for it. If you do localization as a second thought your interface is going to look horrible and then you have to fix it. But you didn't start with a user interface that was enabling the ability to do it, so then you're just going to waste time."

"Localization can be difficult because sometimes you have to render the page from right side to left side, it's not left to right. There are several steps involved. Typically that can [create] a little bit of frustration if you want to do localization in a language that you don't know well. For example, a person who knows Spanish can do the localization well if he knows English and Spanish. If they don't know Spanish, and if they had to do it, I think it there might be a little bit of frustration."

Pace of Change



Developers acknowledge the web is an environment that changes, often quickly. It's a reality that they've come to accept — almost with an air of resignation.

"...we're pretty good about keeping on top of things...whatever happens, happens."

"As a developer, I think I should just accept sometimes the reality is the way it is so... if the environment changes quite fast...no matter how fast or how slow, I need to adapt."

"Trends it's important to follow them because it's nice to see where the industry is going...they're not as frustrating as I initially thought because, in the end, it's just something that is always going to change. They're going to be frustrating, or less frustrating, every few years because they're going to be different...maybe you're not going to like all of them, but they're not even something that you have to follow strictly. So if it's something you don't really like, you can still rely on something else."

"Every time I try to learn something new, I always think it is more complicated than it needs to be, and I think everybody has that feeling."

Not all change is bad, as change can lead to advancements in technology and more efficient ways of doing things but the frequency is difficult to keep up with and has implications for their work. Mainly, there's a fear that as new technologies and tools become available, it will require work to migrate over and may break things in the process.

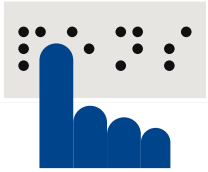
"...the pace of change in the web arena is quite high. That's frustrating somehow because you have to migrate all your projects and from one

to the other tool."

"Pace of change is frustrating and very important especially when it comes to updates, and how fast things change and things break. Sometimes you don't know how, or why. And it's so hard to keep up, especially when you are trying to maintain a different set of apps and some of them are legacy apps...because in many cases, it means your apps break."

"...frameworks that are used for web development, in large part have to do with the efficiency of doing certain tasks that are currently popular and have not been efficient before... like encryption, were the MD5 checksum was discovered to be imperfect, at some point, cryptologically speaking...anyone who used it for extremely secure purposes needed to stop, which is stressful...that had to do with more of an advancement in math, which wasn't anybody's fault...I worry that an existing technology would be obsolete artificially quickly. If it was used in a form, or if your projects were dependent on it."

Accessibility



Similar to authentication, accessibility, developers described it as difficult and frustrating. Developers understand the steps to follow to make a site accessible, but sometimes those steps are hard to get right, and the standards leave a lot to be desired. Often, a developer might be in a situation where their organization does not prioritize

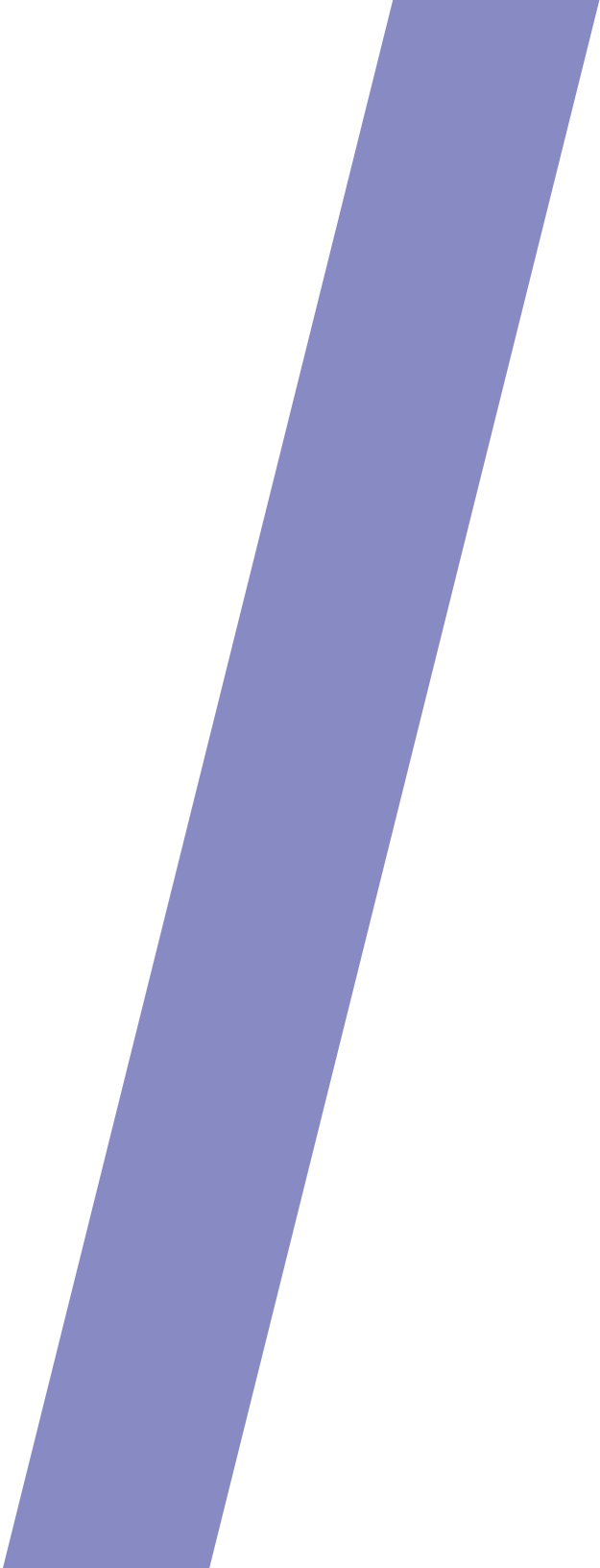
accessibility because, perhaps, it hasn't come up as an issue, yet. The awareness of the need isn't there. In these cases, accessibility is seen as less important than the time and effort required to implement it correctly.

"Accessibility this is so important...it's frustrating because it's hard to get it right. I don't think most web developers, myself included, put enough effort to make our apps accessible because it's hard. You need to use these tools that simulate how it will be to use our app [on] a screen reader. The tools are there, we just don't use them. I don't think it's fair to say I'm frustrated with accessibility because it's just a matter of putting in the work to get it right. But it's definitely super important because there's so many things when you are developing a web app you just assume people are going to use it in the way you use it...when you don't see the problem yourself, you don't think of fixing it. So maybe this awareness what is missing in general?"

"Our application is not as accessible as it should be. I'm going to put [accessibility as] a low level of importance, not because it really should be that low but because we haven't prioritized it...we sell our software to companies that provide our software in an iframe to their customers, and they've never once mentioned accessibility to us. In my previous company accessibility was very important, we made sure that everything accessible. But my current company, it's it hasn't once been mentioned, and I'm keeping in the back of my head that at some point, we're going to need to work on this and make sure that our software is accessible, but it has not been mentioned by one of our customers."

"[Accessibility] not that I want it to be an afterthought, but it's the byproduct of the teams that I've been in. I do recognize how important it is, but it's just not something that's heavily invested upon, I think."

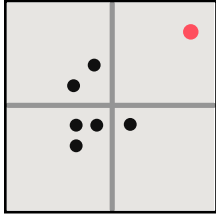
"I would say the level of frustration that [accessibility] generates is greater than its level of importance, and it is important. Without minimizing its importance, it is important, you should be able to access web technologies, regardless of whether you have a visual, physical, or cognitive impairment. Currently, ARIA and web accessibility specs are a mess. We get accessibility audits that will be contradictory between each other. It is important and the way it is currently implemented... and the way the specs are interpreted (or not interpreted) and are clear (or not clear) is just a nightmare."



Outliers

Some of the pilot interview findings shed light on needs that are outliers, which were not tested in the survey. We include them here as they are things that some web developers have to wrestle with, and that can cause frustration.

Ops



When referring to Ops, we mean anything related to creating the build and deploying it.

The audience who participated in the pilot interviews found Ops to be a source of frustration due to a lack of familiarity and practice. Most of the participants were not directly responsible for Ops

or their projects don't warrant that level of complexity.

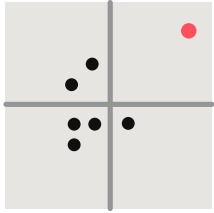
"I recognize that, for example, Ops, if I were more familiar or if I had enough repetition to get familiar then I could see that going down [in frustration levels] I think part of it is investing the time to get familiar with whatever the stack [is] or [the] core technologies that someone might be working with..."

"I started deploying on Heroku nowadays, I'm not very familiar with the continuous integration and deployment. I guess, maybe that's why this is a problem...I don't really understand the systems oftentimes."

"...deployment, I wish I knew more...I haven't had much opportunity to work with them and thus find them unclear. I don't know what a coop cluster is. I have the most basic familiarity with docker images. We run a couple things on docker containers...I would say they're incredibly important, at least to my job's codebase..I put [my] frustration [with it as] lower...because I haven't worked with them much. They just are opaque to me or unknowns."

...if your Ops are fine [then] handling many, many deployments is not going to be too stressful because if your operations are fine, deploying a lot...it's just not stressful, it's just something you have to get comfortable doing."

Tools



Tools is a broad subject with ties to other areas of web development such as debugging, testing, and browser compatibility. Developers are frustrated by the pace of change in the tools arena. It is difficult to keep up with what exists. Even if they feel up to date on their options, deciding which tool to use can be a chore.

"The situation that frustrated me about tools is more just me not knowing what's out there..."

"An amazing number of [tools] are available, like tools to minimize images, to clean up the code, or to modify the code...there's so many of them and they're incredibly useful, but sometimes it can be difficult to decide which one to use."

The reasons for choosing tools are whether or not something is supported, commonly used, or easy to use. That latter reason is a way in which developers measure how good a tool is because they don't want to spend all their time learning how to use a tool. Commonly used tools offer developers a level of assurance that it will likely continue to be supported into the future.

"...before we start using any open source project, we carefully evaluate whether or not we think that this is a viable and supported project, or else this will just become more technical debt for us to handle in the future."

"We try not to add lots of little libraries because we try not to have too many dependencies in our system. That tends to be where we have problems going forward. Like, we've integrated certain libraries into our system or use certain tools that are no longer supported. So we either have to start supporting them ourselves or abandon them."

"...in terms of tools there seems to be new tools that come out. I err on the side of whichever is common...that's the tool that I should be well-versed [in]... if I were to look up how to do something it has to go beyond what the documentation is willing to provide because, you'd have, for lack of a better term, superstar web devs, 'Oh, they use this particular editor,' if they use that then you know you have resources that you can go to."

"... the frustration [with tools] is the ramp up time to be brought up to speed and the level of importance is because the tools that you interface with you want to spend most of your time getting to the end point rather than learning the tool."

"I would choose a framework if it has a lot of libraries and tools that I can instantly use...and if it's already tested, if it's been there for some time. I would go with an old framework instead of new framework that is still developing...If there is a framework that's been in the market for more than five years, I think that will have more tools, more testing tools, they are time tested, but a project that is still developing may not have [that]."

Developers do their best to avoid tools that have quirks, bugs, or perhaps don't run on Linux (if that's their preferred operating system).

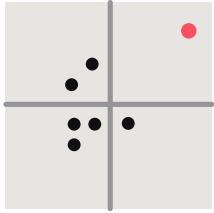
"I've been using Linux for a long time on the desktop and on servers. I'm used to things not working out of the box and [even] so the things that [still] frustrate me are [the tools] that don't work on Linux."

Obviously, [tools are] important because I'm using them all day, every day. They can be a bit frustrating because often the tools can't keep up with the rate of change. Also, I find that they can be a bit flaky, stuff that just doesn't quite work properly, which I think is partly just because getting things to work properly in JavaScript is

hard. Little issues, like DevTools in Chrome, for ages you couldn't resize the windows properly and it just seems to always be, I guess because they're not selling the tools...they seem to break a lot.

"The rest of the DevTools in the browsers are the most important thing I need, and they seem to be flaky a bit...little user interface issues that just make it frustrating...They're definitely getting better and it feels bitchy to complain about them when I'm getting them for free. The browser manufacturers are companies that are trying to make money just like everyone else, and the browser is in a market. I guess the DevTools tools are there to encourage developers to develop for their platform a bit more, but it still feels like it's a nice thing they're doing for us, rather than a core thing...I probably should expect them to do it if they want me to their platform."

Working With Others



Some of the pilot interview participants work on a team with others, which can cause frustrations. This is not unique to web developers, but how it impacts them is that it can create a more complex environment for managing code, and it can also dictate project requirements. There are also perceptions of different types of developers which color team dynamics. Most of these came about during deployment.

"Frameworks are the byproduct of what management or senior developers...dictate...I work with developers that are not willing to make a change or try different frameworks, because why [not] go with something that's tested?...that's when you start to get into red tape...I was trying to pitch possibly migrating to React...back in 2016...long story short, it wasn't something legal was okay with due to licensing. That in itself is frustrating, but there isn't really anything I could do about that"

"...in the company it is understood that [code architecture] requires time and that by just going fast all the time, sometimes you have to, because, 'hey, market, you know, if you have to do something for your product you have to, but as long as the company knows that if you go faster than you have to pay the price and do it again. [With good] code architecture the level of stress level can be reduced so much. There's nothing you can do about the importance because in the end, it's what is going to help you so much along the way...[it can] reduce debugging a lot because if you can separate your logic in nice pieces, you're not going to mix concerns. And so when something's not right, you know exactly where to look."

"I'm committing more often [than I would if I were working alone] following the guidelines of what the team set. Typically the way we do it is there's the master branch and then we have our own topic branches that we develop...in order to have a successful coding review, you have

to commit and push to your topic branch. Once the team is met with consensus, like, 'Okay, I think we're ready to go to the next version,' we then would merge all of our code to the new would be master."

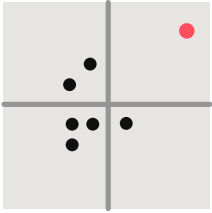
"I'm managing all these tools, along with one of the other developers, it can get a bit frustrating. For instance, yesterday, the other developer upgraded our PHP on one of our systems and Jenkins broke. We couldn't deploy any of our software...it took us three hours to figure out how to get our deployments working again."

"...code quality, that's something that becomes more and more important the more people working on [a] project. Same thing with architecture. The problem with all of those is they could be anywhere from less important, low level frustration to more important, high level frustration the more people there are in an organization...Code quality shows the respect you have for other developers, I think, because you're not the only person who's going to work on something. You have to make sure that you leave the code in a place that is going to make your life easier and everybody else's."

"[Ops is] getting the code out there and making sure it doesn't fall down. As a front end dev, it's mostly someone else's problem, not because I don't want to be involved, but I find that in a lot of places I've worked there's this attitude that front end devs don't know anything about Ops. It's weird, because it's like, 'Why would you expect the Java Devs to know how their stuff is deployed, but not expect me to know how my stuff is deployed?' Which I find frustrating...there was a bit of a lack of respect for the front end, which is one of the reasons I left."

"Debugging, I don't have a ton of tools based frustration or product based frustration with it. Mostly just reading other people's, and I count [me] from six months ago as other people, their poorly commented spaghetti code."

Hiring Web Developers



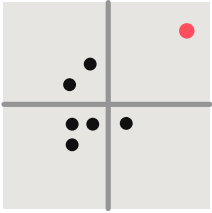
What we heard about hiring can be summarized with two additional need statements:

- Finding developers with the right skills
- Vetting the skills of potential hires

We learned during the pilot interviews that different regions of the world have different skill sets to offer. Hiring managers evaluate regional skill-sets when making decisions about where to post and advertise a job. Vetting the skills of developers, prior to offering them a job proves difficult, and if not properly vetted can lead to a poor hiring decision.

"...we've hired three developers that didn't work out...some of them... lied on the resume. They said they were good at certain aspects of technology where they weren't. Because of that...we had different expectations of them and they weren't very motivated to learn."

Infrastructure Variance



This refers to intermittent power outages that affect business decisions, e.g., where to locate data centers. We experienced the impact of this first hand while trying to conduct the interviews. One of the participants was located in an area where they were experiencing intermittent power outages and was unable to complete the interview.

“Early on, there was a decision not to deploy into the cloud because our parent company has his own data center, and the data center is located in [a country where] we have issues that sometimes the power goes out, or sometimes there is a network outage. It is a developing country [it] doesn’t have the same stability as you find in North America or Europe. We are looking into alternatives and whether or not we should be moving more of our services to the cloud, and how we can make things more stable.”



Overcoming Frustrations

Overcoming Frustrations

After exploring the topics that cause web developers frustration, but are also important to their work, the conversation shifted, focusing on the resources they use to overcome these frustrations. Two dominant resources help developers overcome frustrations. The first is a community of peers, typically personal acquaintances or acquaintances with few degrees of separation. The second is tapping into the resources available online.

When developers seek input from their community of peers, they usually have a sense of the skills and talents of others they are close with. They call upon people to help with a tactical problem such as troubleshooting an issue or help them with choices that will help them grow as a developer such as advice on what tools or frameworks they should invest their time to learn.

"I think the best way that you can learn how to do anything in this space seems to be to have a group of people who automatically or implicitly or without much effort on your part keep you aware of news that you would otherwise not see about the state of things."

"In the case of Ops, my friend is good at it. He just does it for me... that's literally how I solved by DevOps problems. I just tell him, 'Hey, this is my problem, can you help?'...he just tells you what to do next. And that's very useful."

"There are local developer communities. So it's very easy...[the] people are really helpful. I just say, 'Okay, I've done this, and what should I do next? What would you advise I do?'"

"...we have a few people who know [security] or have trained in sales and security, and we continuously go back to them...they're just friends."

When developers tap into online resources, they are using the usual suspects (listed in alphabetical order): freeCodeCamp, GitHub,

Google, MDN, Reddit, Stack Overflow, W3Schools, and YouTube. Part of what makes these resources so valuable in troubleshooting and overcoming frustrations is that they provide developers access to a community beyond that they have cultivated in their networks. This broader community has very likely experienced a similar situation and can help resolve the issue, whether that is shared through forums or videos. These sites also offer official documentation, which is a key resource in troubleshooting.

"The web is just an incredible resource because it's almost always the case that somebody has the same question you have..."

"The primary point of contact [for troubleshooting] is, how do I form a Google search to get to the information?"

"[To overcome frustrations with something I don't know] I will just Google and find information or YouTube. [Some common sites that I find to be helpful] are Stack Overflow, and of course, Mozilla, W3Schools...[the sites are] easy to access, easy to find information on what I need, well organized, visually attractive, and easy to understand."

"Documentation is [the central thing for helping me overcome frustration] partly because...you think you knew something but then after reading up on said feature or said documentation, it's like, 'Ah, I should be implementing it this way,' rather than a learned know-how, so to speak."



Needs Are Situational

Needs Are Situational

When it comes to web development, developers experience myriad frustrations, which can represent opportunities for browser vendors to address that frustration and thereby fulfilling a need.

While this report focuses on the top ten needs of web developers, there's more nuance to the frustrations than a top ten list can convey because needs are not static. At any given time, a web developer may be in a situation that affects what they find frustrating, why it's frustrating, and how it causes frustration. Four main situations emerged from the pilot interviews.

- Being a part of the broad web development community.
- Working on a project for oneself without external stakeholders.
- Working on a project with requirements.
- Adapting to work cohesively within an organization or as part of a team.

Let's consider some examples. First, we know one of the top 10 needs was, "Keeping up with a large number of new and existing tools or frameworks." Why and how this is frustrating is experienced differently in different situations.

Working on a project for oneself without external stakeholders.

"When I write code for myself I'm often doing something that's simple so I don't need frameworks, but more than that, I like to understand how stuff actually works and frameworks hide the implementation details."

Adapting to work cohesively within an organization or as part of a team.

"When I go to work, we use Angular and it's fine. If I was working somewhere else that used a different framework, I would have to adapt and learn that new framework. It's a byproduct of what management

or senior developers dictate. I work with developers that are not willing to make a change or try different frameworks. Their preference is to stick with something that's tested or something that's already been approved by legal. For instance, the team talked about using jQuery to build a web app. We could do that, but for efficiencies sake Angular or React would be better. Legal got involved and wouldn't let us use React due to concerns about licensing. We stuck with jQuery."

Being a part of the broad web development community:

"I have a community of web developers. We all have different perspectives about what's popular at the moment, what's worth learning or not. One friend encouraged me not to learn Node JS because he felt it was too specific. But, I've also read many times that it's useful and a good thing to learn. Angular is another example of something that I didn't even know existed six months ago and I'm already getting competing opinions about whether it's worth the time to learn."

The type of project a developer works on can create or eliminate frustrations because of the project's requirements.

"Our software only uses anonymized data that we received from a third party, so [privacy] is not very frustrating for us. [We] don't actually have any identifying data."

"I'm happy if my stuff runs well in Chrome. I'm not...trying to target old versions of Internet Explorer or something like that. So I don't really care [that much about browser compatibility]. But it is important, and I would expect it to be important to somebody who hired me to build something."

"I haven't produced real marketable stuff [where I] had to worry about [accessibility]...I do a lot of this stuff for my own amusement. I'm playing with it so it just hasn't been an issue for me. I recognize it's import-

ant, but I haven't had just haven't had to deal with it."

"Well, I think of available tools that you could I'm thinking of like dev ops, which I don't really need. So I don't know if that really applies to me there. Because I'm not doing anything complex enough to require, like any deployment, to, you know, more than one place or any real integration, outside resources. So I think I'm going to leave that alone."

"I'm basically trying to run a web server host some HTML, JavaScript, CSS, and deliver this to my clients. So just keeping a single web server up and running. It's easier than having a complex framework of micro-services powering my application."



Broader Picture

Externalities

The evolution of the web has led to the state of affairs where developers can pay for services, such as hosting, which reduces the need to own and maintain all of the infrastructure that was previously required to put something online. However, each piece you don't own creates a dependency on someone or something external to your work. Dependencies present risks, and developers have to decide whether the tradeoff is worth it or not. While the upside is that making things for the web is much faster and easier, there's a fear that their work might become obsolete should something happen to those who they are externally dependent on.

"...owning the entire spectrum is excessive, it's just ensuring that those trade-offs are either things that you [can] control the copy of that you're using, or that you're paying for service that you can take your money elsewhere."

"Unless you are starting with HTML, CSS, JavaScript and writing your own Ajax requests, using the breadth of the browser object model, XML HTTP request API to shoot code back and forth...you're always giving up some control over your code...In terms of importing code dependencies, that's the cost of doing business. It just makes writing human usable applications so much faster. But the act of hosting is following that same path...the system of hosting is also being abstracted away to third parties. There are horror stories every week of somebody whose business was just stuck...years of work [gone] because they had built everything into Google Cloud Platform and one of the developers that they had hired on to make some tweaks to their business application was using a blacklisted Gmail account. And because of that, the app certifications were all revoked....that's a huge externality, you have to think about...We're hosted on Heroku, we have stuff on AWS if by some chance X service provider disappeared, would your company also disappear? Like, if Amazon just poof was taken away by aliens tomorrow, our company would not exist...we have backups, we could spin up something else but there would be a week to two-week-long period

where we would just be building everything back up for availability."

"...my brother [a developer] feels he is at the whim of the creators of browsers...the concern is at some point, something that his work relies on will be obsolete by a decision that Google makes...or a decision that Microsoft makes...it's a little scary that things advance quickly...the thing that concerns me about changes [is that] updates would have to be made on account of third parties, such as the creators of the browsers... you don't have as much control over, for example, like the version of a compiler that you use, which is static...I'm referring to the obsolescence of technologies that people at one point relied on heavily...it's hard to predict which existing technology...will end up being the one in which a flaw is uncovered and a great deal of work left to be done to move to something else."

Browser Reborn

Developers have noticed that browsers are getting better, with Internet Explorer (IE) as the exception. Developers realize this wasn't just a kind move on the part of browser vendors to make developers' lives easier; they understand browsers have grown in monetary importance for their creators.

"[Browsers have gotten better] because the industry started to care... ten years ago...the market wasn't as competitive, the browsers were not actually...trying to do a good job in implementing things apart from Firefox..."

"[Browsers had] been left as a second or third class citizen for ages. Now...Google is pushing on it because they live on ads so they need the browser to do whatever. Microsoft went back and started doing their own browser and it was actually good, but probably took them too much time to get up to speed and now they have to change because Edge was not as good as it should have been. Mozilla, well, Firefox since they changed to Quantum it's... it was not getting much love before...even Apple now is doing good things on the web, even if they're doing it at their own pace with their own reasons, but that's Apple you cannot expect too much..."

Better browsers is a good thing, but the consolidation of browsers concerns to some. The first concern is about a lack of competition between browser vendors leading to a lack of technological advancements. Consolidation also has implications for compatibility, which manifests in different ways.

One view is that, as each browser vendor tries to outshine the other, those advances will create differences that may result in less compatibility. However, it's for the sake of advancements, so it's something developers with this view can live with.

On the flip side, some developers see competition creating more

compatible browsers because each will be trying to match at least what the other can do. But, as competition decreases, so might compatibility. Microsoft's decision to adopt Chromium is perceived as reducing competition, leaving Chromium with a larger market share. There is a fear that it will become harder for other browser vendors to compete with Chromium. The browsers with less market share may not invest as much in making their browser better, leading to it becoming outdated and causing compatibility issues.

"Browser compatibility...might lose importance because browsers are getting so much better...the further we go the less time I have to think about it because I know that everybody's competing..now that Microsoft decided to stop developing their engine I'm worried that Chromium is going to get too much market share and that is not going to help with browser compatibility."

"I think [people moving to Chromium is] a good thing as a designer... but as someone interested in the further development of technology, it's not the best thing because considering everyone will start using Chromium, there won't be a competition left. If there is no competition, the pace of development...decreases...there should always be some competition. I think Microsoft declared that they're shifting to Chromium as well, so if they do that competition will decrease, it will be hard for Mozilla to sustain with their engine. It's a good thing in terms of users or developers, but in terms of technological development of that particular set of a browser or how a browser evolves, it's something bad."

"...I would love to see Mozilla and Chrome competing with each other to improve the web, I see that the two are basically trying to improve the standard, I like it very much that there is no fixed standard anymore."

Knowing How Things Work

Web developers understand the Web, and the surrounding ecosystem is continually changing. With that comes pros and cons. Considering, in most cases, it is their livelihood; there is a strong desire to maintain some semblance of control and understanding of their work. One way developers do this is to invest time in understanding how things work — beyond the abstractions that modern conveniences such as frameworks present. Understanding the basics boosts developers' confidence that they can handle something when issues arise.

"I think [frameworks] are not that important because the fundamentals about how things work are more important than the specific frameworks, [but] for the web in general, and other developers...they allow so many more things to be done for the web."

"A basic knowledge of programming is really important because you will use basics even [as a] senior...it's really easy to make a basic mistake...it takes a lot of time to build the logic of basic programming knowledge."

"If I am writing code for myself, I generally don't use frameworks. I just go hardcore...[so] I know how it works, and then when I go and use it in a framework I get what the framework is doing. Whereas I think a lot of people coming in to front end, they learn the framework and they don't know the JavaScript that's going on underneath, which probably gets you by 80% of the time but then when something scary happens, it's like, 'Why did that happen? This is magic.' I did the same where I jumped into Angular and I'm like, 'No, this is way too much magic. I need to know what's actually going on.'"

"...in terms of obscuring what's actually happening...frequently, you don't need to know it's enough to say, 'Oh, in React land I'm passing this state property as a prop down to this child component,' and that's good enough, but what's actually happening underneath the hood..."

when it is time to do that digging, it can be really unclear, messy if you spent all your time just living in frameworks land..."

"If I learn something like basic JavaScript, then I think I can switch to different frameworks."

Experienced Developers Recognize Improvements

Much of this report's focus is on the needs of developers and what causes frustrations. These frustrations are representative across many years of experience from a beginner (less than a year) to seasoned (ten or more years). Developers who fall into, or near, the seasoned camp recognize that things on the Web are better, compared to when they first started. Improvements run the gamut from tooling, deployment, debugging, layout, privacy, ES6, etc.

"I think we have really great tools now to do web development compared to many years ago. The tools now, I mean, they can always be better, but it seems like we have really good tools [they] weren't so common 8-10 years ago."

"Testing is hugely important for making sure that things work as expected. There's a bunch of really great tools available to help with that now, which I find super helpful...It used to be super frustrating, but the tools are getting better and better."

"Deployment is really important, obviously, because you need to be able to put your app out there but it has become so much easier as well. Now we have services like Heroku where in just two clicks, you can deploy your app, like if it was magic. I remember in my first year we have our deploy and build scripts, they were written in Bash..."

"The new Grid and Flexbox [are] so good. And Firefox in particular, has really amazing support for being able to debug those layouts and see how they work...the grid Inspector, it's so good, so helpful because you can actually see what's going on. It's like, 'This is where the grid is, this is where the gaps are, this is why that grid is over there.' They've got the Flexbox inspector as well. It's like, 'Why are my things all over on the left? Oh, because this is what's being set, this space is empty gap, or this space is actually an element that turns out to be way bigger than I thought it was.' Getting people at work to move to that way of doing things instead of floats left, it's been fun."

"When functions first came out, they were impossible to debug in the dev tools because, I don't know what was going on it was just really hard. Whereas now the debugging tools have gotten way better and it's way easier to do that kind of thing. As JavaScript got more asynchronous debugging, got a lot harder. And now it's getting easier again, so that's been great...those tools that are in the dev tools have been amazing to help with debugging."

"I think there are great tools for debugging...I use the debugger in Chrome, mostly for JavaScript. For server side development I use the Ruby debugger. I also use logs. And just in general, the web console in Chrome as well, and the Network tab. All the developer tools they help with their debugging a lot."

"I think layout has become much less of a pain point. Layout has been more difficult in the past, designing all these huge tables to have to achieve a certain layout...I think performance has improved dramatically, I think the tools have improved dramatically, too. I think the pace of change is also higher than ever before, I think debugging has been harder in the past. I think all these tools have improved tremendously."

"Firefox or Safari implemented a really good Do Not Track engine. Safari doesn't let me know exactly which MacBook you are using, it's just going to look like a generic one. They do have special ways of changing the tracking cookie so it's not going to always be the same, and that's something that they just improved. And now Safari and Firefox are using the restrict mode, so if the user wants you can prevent people from tracking what you do. I think we're going in a better place in terms of privacy, I'm just optimistic about it."

"I used to be more frustrated and stressed out about frameworks because there are so many and they change so much, but I'm not

frustrated anymore about that. Since JavaScript ES6 everything has become standard and I feel like using vanilla JavaScript is so much nicer than before. And I don't think using a specific framework, knowing a specific framework is so key [as] it was in the past."

Keep in mind that while there are recognition and appreciation for how things have gotten better over time, it does not mean things are perfect. Some improvements are what lead to the current environment where developers are facing trade-offs when deciding how much to divest their control into external, third party products and services. That and the pace of change can be overwhelming.

What's Missing From the Web

The same group of seasoned developers had thoughts on what is currently missing from the web. These are similar things to what was captured in survey findings and are included here to provide verbatims with more context. In addition to what's missing, this group had thoughts on opportunities for improvement.

Web components and privacy could be better executed.

"Web components have a great opportunity, but they wasted so much time with things that...in terms of the choices that were made, didn't pay off like HTML imports. They we're not the best way to approach it because all of the things that they didn't allow in terms of good optimization."

"I think [how to overcome privacy frustrations] is the \$1 million question, right? Because I don't think there is this something where there is no good solution currently to it..."

Access to hardware is something that the web is missing, as well as push notifications and serverless communication.

"Hardware access...more device API's would be helpful. And I think, of course, also push notifications are a very important aspect. So if these two would work more reliably, or let's say would be better supported, I think I would be very helpful"

"This is a very personal opinion from a computer science guy...a more event-based browser to browser model, like in the web RTC environment. If I could place anything on the list, I would say, browser to browser communication, serverless communication applications. This is something I would like to see in the future so that basically you don't need a web server, just bring two people together like web RTC."

The Future of the Web

Not only do these seasoned developers recognize areas of the Web that have improved, could still use improvement or are missing entirely, they also had some thoughts to share about the future of the Web.

When it comes to the future of the web, these seasoned developers are excited for:

- WebAssembly
- Progressive Web Apps
- WebAuthn
- Speech to Text

“WebAssembly is going to change the web. This guy is an insane level of importance. It’s going to change the web from being a mostly consumption light editing environment to a fully-featured editing environment. It provides an increase in speed that is unheard of for the web.”

“I think if you allow web applications to be more like native apps, then you can basically replace many of these native apps with web pages, which gives them a simple deployment model and development model for web developers.”

“ I just saw, now, they finally decided on the spec of WebAuthn. So once browsers [are] actually going to implement it is going to be much easier and safer for users to authenticate...that’s probably me hoping for a better future.”

“I would like to see more is text to speech and speech to text...I think this is a very important topic for the future of the web...I’m not sure if, in a few years, we will still use graphical user interfaces to interact... And I think that web applications should also be able to run headless, without having or without being forced to have a user interface, graphical user interface as their main interface towards the user. I think speech is something that should get more attention...”

Though there are things to look forward to, there is a concern, and it’s not clear how widely this concern is shared. The concern is that the next generation of web developers might not see the importance of learning the fundamentals of computer science as it’s so easy to just run with examples.

“My hardest [job as a teacher], what I’m trying to do...is to convince [my students] not to go to W3schools.com and just do an example-driven web development. My students typically think they can Google every solution, while I try to explain to them that they should first use their brain before they do something. I think they would tell me that the web is some copy and paste place and they would throw all these computer science rules overboard and just try to copy from existing projects.”

Conclusion

Conclusion

All of the organizations behind this first iteration of the Developer Needs Assessment could not be more grateful for the 28,474 people who completed the survey, plus those who partially completed the survey. And, last but not least, the fourteen pilot interview participants, without whom, the survey would not have been possible.

This study was an opportunity for participants to influence how organizations like Mozilla, Google, Microsoft, Samsung, and the W3C prioritize feature development and standards.

The first version of the Developer Needs Assessment ended with valuable information that organizations can use to create a better experience for developers and designers who work on the Web. Since this was the first time the study was conducted, the findings are new to the organizations behind the work. We're still evaluating how best to use the results to influence feature development and standards. That said, we're happy to share the initial actions organizations are taking to improve developer and designers' experience working on the Web.

Google

"The Google web platform team is now using Developer Satisfaction (DSAT) as one of our top-level success metrics. We are excited to be using the MDN DNA as one of the main sources of data to help us to understand and prioritize the key areas of developer frustration."

- Rick Byers, Director of Engineering for Blink, Web Platform Team

"Google has been working with other Chromium contributors, such as Intel, to bring access to hardware capabilities to the web, from cameras and sensors to gamepad, USB and Bluetooth APIs. We believe that any task users want to accomplish with their devices should be possible with the web, so are continuing to work on web standards to

expose additional sensors, Serial, HID, NFC, and more."

-Joshua Bell, Technical Lead/Manager, Web Platform Team

W3C

"Early reports from the survey provided valuable input to several standardization and pre-standardization discussions at W3C's big annual meeting (TPAC). We anticipate the published report will continue to support standards progress."

-Wendy Seltzer, Strategy Lead

Mozilla

"In the Firefox team we are always listening to our community's needs in order to make product decisions. The comprehensive overview of the developer community's needs provided by the MDN DNA report is therefore absolutely essential to us and we are already incorporating its findings into our plans."

- Panagiotis Astithas, Sr. Engineering Manager, Firefox Developer Experience

Beyond individual organization's actions, and recognizing that interoperability is a major pain point for web developers, major browser vendors are working together on cross-browser test suites like web-platform-tests and Test262. The group working on web-platform-tests have taken, [as a 2020 priority](#), to identify and fix the most important test failures and improve the quality of the test suite.

Methodology

Methodology

Considering the pilot interviews provided an outside-in perspective that shaped the first iteration DNA survey, we wanted to offer a thorough write-up of the methodology for how we conducted the interviews.

Pilot Interviews

To ensure as broad an audience as possible, project stakeholders created recruiting criteria to help gather a diverse perspective from different countries and gender affiliations. We broadcasted a survey on MDN, and selected pilot interview participants from completed responses, ensuring we met the recruiting criteria.

Those who met the criteria were invited to participate in an hour-long (at most), remote interview. Narrative inquiry was chosen as the methodology as people make sense of their world by telling stories. Narrative analysis encourages participants to explain meaning and truth from their perspective.

A guided but flexible discussion was chosen to encourage participants to remain comfortable and conversational throughout the interview process. Language, tone, and topic were guided by the participant and used to inform areas of inquiry as well as contextual interpretation of the participant's words and actions.

In order to accurately map participants' perceptions of a topic related to web development and design, researchers provided an X/Y grid. The X-axis represented participants' perceived level of importance, while the Y-axis represented their perceived level of frustration of the topic of interest.

Participants were then given a set of cards that represent touchpoints of web development or design and were asked to place the

cards onto the grid according to their own opinion. The list of touchpoints were defined during the kickoff with stakeholders:

- Accessibility
- Authentication on Mobile
- Browser Compatibility
- Debugging
- Deployment
- Frameworks
- Layout
- Localization
- Mobile Payments
- Ops
- Pace of change
- Performance
- Privacy
- Security
- Testing
- Tools
- Trends

Researchers watched participants place the cards on the grid, and asked why cards were placed in a specific location or about the relationship between cards. They examined verbal cues in the context of the conversation, to guide the discussion toward experiences that participants found especially relevant, meaningful, or memorable.

If a participant didn't understand the meaning of a card, they were instructed to define what it meant to them, and if no material description followed, they were asked to leave it off the grid. At the end of the activity, participants could create cards representing a touchpoint not found on the list provided. They were instructed to place the newly defined touchpoint on the grid in the same way that the

pre-provided touchpoints were placed.

When permitted, interviews were recorded, and photos were taken of any artifacts created.

Analysis

Following the interviews, Pinpoint began analysis, which started with detailed coding of the transcribed interviews. They read through each interview transcript to review the narrative — what did the participant choose to talk about? What was interesting? How do they frame things? What’s unique about their language? How do they want to be perceived?

After coding, Pinpoint’s process transformed from information sifting to creating a narrative that highlights evidence (what they saw and heard from the interviews) to identify insights (what the evidence means), which helps unearth critical themes (why the evidence matters).

Survey Creation

Initial analysis of the pilot interview findings informed the survey questions, most importantly, the needs list. The original needs list included 59 need statements within eighteen categories. This list was pared down to the 28 needs tested in the survey. MDN Product Advisory Board (PAB) members, as well as others employees within the PAB organization helped weigh in on how to reduce the needs list. A shorter needs list was important; otherwise the time required to complete the survey would have been too great, and our participation rates would have been less.

Once a draft survey was ready, it was piloted through UserTesting as well as individual interviews with participants recruited from

MDN. These pilot efforts were an opportunity to gather early feedback from the perspective of our target audience to help ensure the questions were being interpreted as we intended.

Once the survey questions were finalized it was localized from English into eight languages:

- Arabic
- Chinese (simplified)
- French
- Japanese
- Korean
- Portuguese (Brazil)
- Russian
- Spanish

The survey launched mid-July of 2019 and closed in August of 2019.



pinpoint