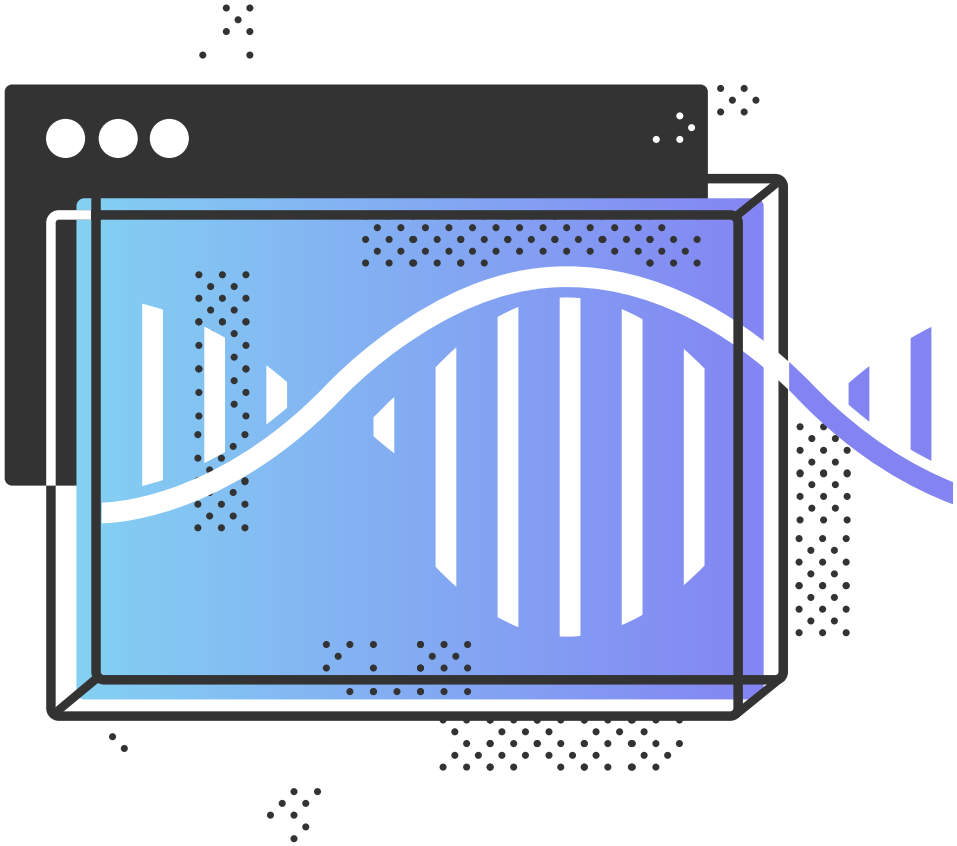
MDN Web Developer Needs Assessment 2020



MDN web docs

moz://a

Acknowledgements

The second iteration of the report would not be possible without many contributions. The core team responsible for executing the survey, analyzing the results, and publishing the report are:

- Chris Mills and Nancy Hang at Mozilla
- Alex Klapheke, Allison McKeever, and Jon Godin at Pinpoint

The MDN Product Advisory Board (PAB) was a major contributor to the project and consists of the following companies:

Bocoup
Special acknowledgments for:

- Boaz Sender
- Jory Burson
- Sheila Moussavi
- Simon Pieters

Google
A special thank you to Google for their financial sponsorship of this study. Special acknowledgments for:

- Helen Harris
- Joe Medley
- Philip Jägenstedt
- Robert Nyman

Microsoft
A special thank you to Microsoft for their financial sponsorship of this study. Special acknowledgments for:

- Kyle Pflug
- Reeza Ali

Mozilla
Special acknowledgments for:

- James Graham
- Maja Frydrychowicz

Samsung
Special acknowledgment for:

- Daniel Appelquist

W3C
Special acknowledgment for:

- Dominique Hazael-Massieux

A special thanks to Kadir Topal and Dietrich Ayala who originally conceived of this project in 2018.

Table of Contents

Introduction

Introduction

Welcome to the second edition of the MDN Web Developers Needs Assessment (DNA) — a global, annual study of developer needs on the web. MDN Web DNA aspires to be the voice of developers and designers working on the web.

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.

Like the community, this assessment is not owned by a single organization. It is 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 our respondents from 2019 and 2020 who, this year, took an average of twenty-one minutes to complete the survey. Between last year and this year, the community has contributed more than 10,000 hours 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 developers on 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

2019 vs 2020

Between our first iteration of the MDN Web DNA and the second iteration in 2020, we changed our approach to data analysis. In the first iteration, the team relied on reporting features available in our survey platform. This year, the team hired an experienced data scientist to conduct analysis and employ data science best practices. You can read more about our approach to analysis in the methodology section.

Another difference between the first iteration and second iteration is the response rate to the survey. This year, there were fewer participants and all we can do is speculate as to why. We suspect that one reason we saw a drop in participation is because we added more new questions this year than we removed, which increased the mean time it took to complete the survey by six minutes, up to 21 minutes from fifteen minutes. Another reason might be that 2020 is an unprecedented year with the global, coronavirus pandemic. How much that affected our response rates, we'll never know.

In 2020, we increased our efforts to recruit more diverse participants by reaching out to different organizations and nonprofits whose missions are to amplify the voices of and provide resources to marginalized communities of web developers. Even with these recruiting efforts, the percentage of respondents who identify as women or those who chose not to identify their genders was down between this year and last.

Because the nature of this study is an open call for participants, year over year comparisons are not apples to apples. However, there were no significant changes in the overall makeup of participants.

Throughout the report we specify where there were changes in the survey questions.

Survey Responses

Respondent Overview

Target
Our target audience for the second iteration of this study was the same as the first, people who spend at least some of their time writing code for the Web. 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 community.

Actual
After fielding the survey for three weeks, we have 6,645 cleaned, completed responses — cleaned meaning the response met the criteria of our data model. How that compares to 2019 is below. Because the nature of this study is an open call for participants, year over year comparisons are not apples to apples. However, there were no significant changes in the overall makeup of participants.

2019	2020	Delta
Total: 76,188	Total: 30,844	55,344 Fewer in 2020
Eligible, Complete: 26,854	Eligible, Complete: 6,645	20,209 Fewer in 2020
Completion Rate: 37.4%	Completion Rate: 21.2%	Completion Rate Dropped by 15.2%
Partial: 49,334	Partial: 22,840	Partial Increased by 16.4% *(as % of Total Responses)*
Disqualified: 5,430	Disqualified: 1,359	Disqualified Decreased by 3% *(as % of Total Responses)*

Responses By Gender

A goal from the onset of this project was to have a broad, global representation of the developer community. Despite increased attempts to get the survey in front of representative audiences, 87% of the respondents identify as men which is similar to last year (87.1%). Our representation of respondents who identify as female is down from 8.2% in 2019 to 6.8% in 2020. More respondents saw neither option as suitable up to 1.8% from 1.1% in 2019. More people declined to state their gender, 4.3% compared to 3.6% last year.

To put the breakdown by gender into perspective, the US Bureau of Labor Statistics[1] 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 10.8% which is a small decline from 2019 where our representation was 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. For 2020, we added a new, optional question which asked respondents whether they identify as a minority within their country and 16.1% do identify as a minority. Only 1% of respondents chose not to respond.

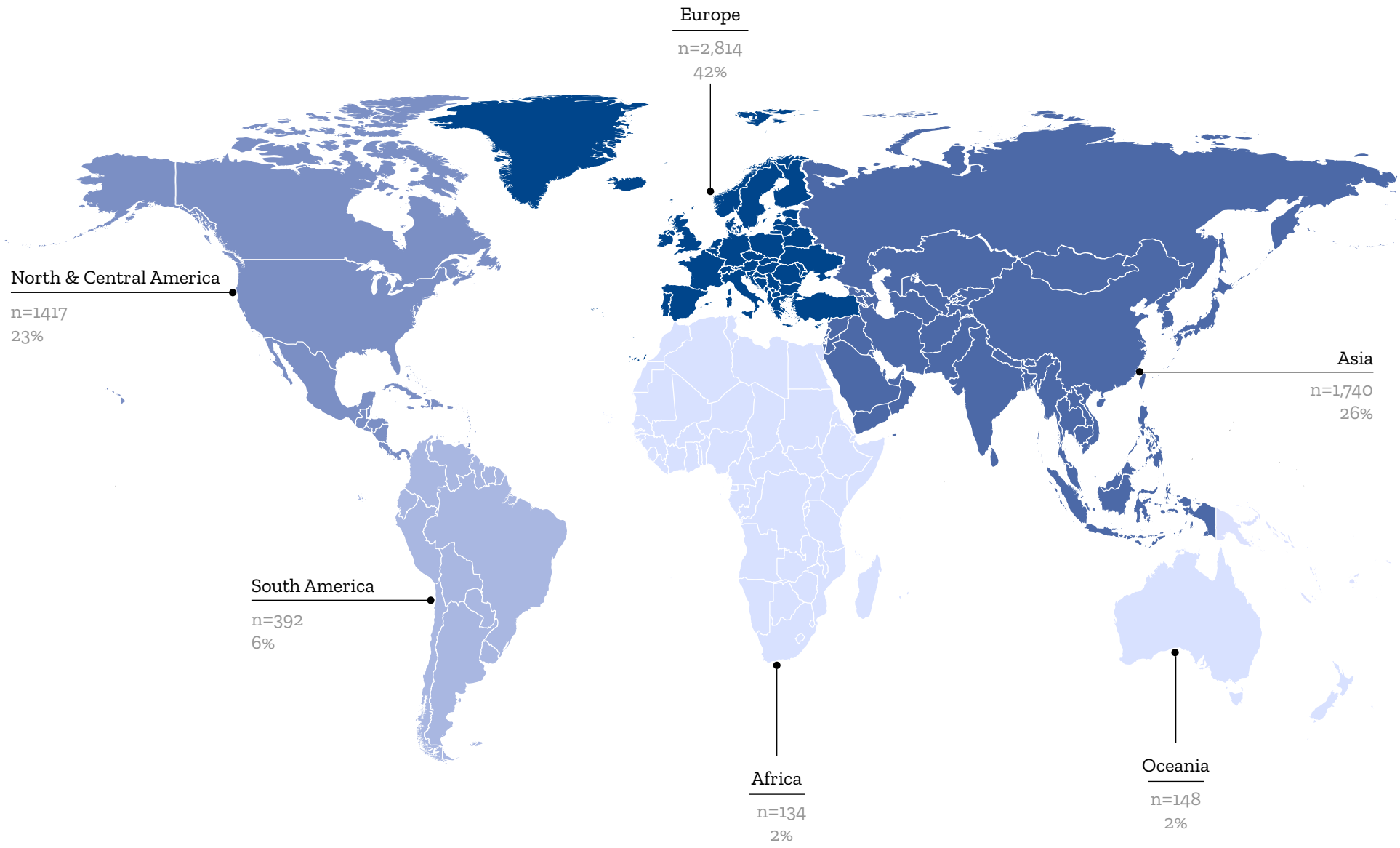
We did attempt to gather more diversity by sending it to specific women-groups. In future iterations, we will continue to aim for fair representation and ways to mitigate or account for the bias.

The 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, 50.4% answered the optional question, which asked for personally identifiable information.

n = 6645

1 https://www.bls.gov/opub/reports/womens-databook/2017/home.html

Responses By Region



Responses By Country

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

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

Last year, we had eight languages. We opted not to translate the survey into Arabic for this year because it accounted for less than 1% of the survey responses last year.

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.

The survey includes responses from 137 countries, down from 2019's 173 countries. 37 countries have 30 or more respondents each.

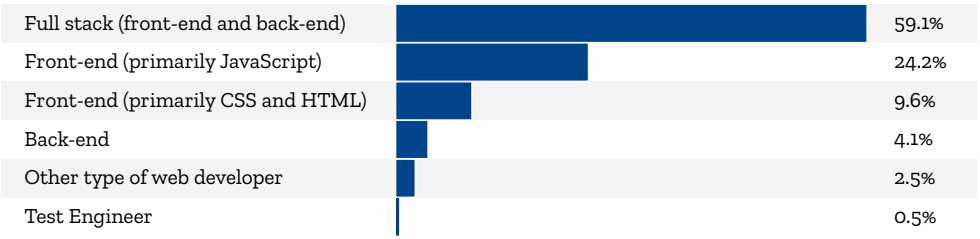
The countries with the most significant participation, measured by 300 responses or more are:

- United States - 16.6%
- Germany 7.4%
- Russia - 7%
- China - 6.4%
- France - 5.9%
- United Kingdom - 5%
- India - 4.5%
- Canada - 3.1%

Responses By Type of Developer

Like 2019, participants were asked, "Which best describes the type of web developer you are?" However this year, respondents were only allowed to select one option whereas last year they could select all that apply. We also included a new type of developer in this year's survey, Test Engineer, but it was not a popular choice, with only .5% of respondents selecting it as their option.

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 59.1%, up from 57.1% last year. Back end had the least representation at 4.1, down from 11.7% last year%.

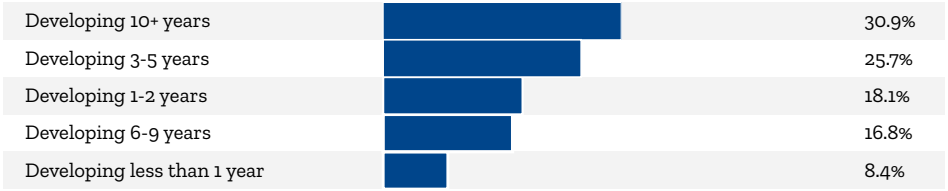


n = 6645

Responses By Experience Level

This year, the respondents were fairly even across developers who have less than or equal to five years of experience developing for the web and those that have six years or more, at 52.2%, and 47.8% respectively. In 2019, the breakdown wasn't as close. We had more developers with five years of experience or less than we did for those with six years or more, 60.2%, and 39.8% respectively.

The largest group in this year's study were developers with ten or more years of experience, at 30.9% of the respondents. Whereas in 2019, the largest group were developers with 3-5 years of experience, at 28.4% of respondents.



n = 6645

Overall Satisfaction With the Web

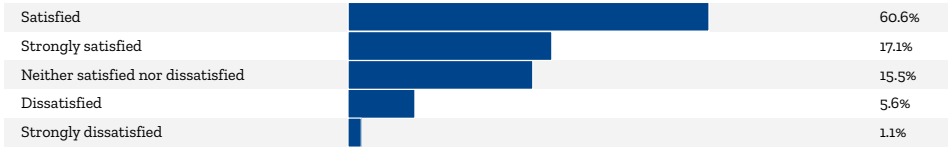
Overall Satisfaction With the Web

In 2019, we established a question about web developers' overall satisfaction with the web. We intended for the question to be repeated in future studies, creating a baseline measurement to see how satisfaction ratings change over time. The question was repeated in this year's survey, with a slight word change from very to strongly. We also changed the position of the question having it appear before the Needs Assessment. We thought asking a question about satisfaction after having respondents sort through things that cause frustration when developing for the web might lend a negativity bias to the results.

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, 77.7%, of respondents are either strongly satisfied or satisfied with the Web, whereas 6.7% are either very dissatisfied or dissatisfied. This represents a slight but not appreciably meaningful increase in satisfaction compared to last year, with a concomitant slight decline in dissatisfaction.

In 2019, 77% of respondents were very satisfied or satisfied with the Web, whereas 8.4% were very dissatisfied or dissatisfied.



n = 6,645

Satisfaction by Subcategory

For a more nuanced view of overall web satisfaction, we added new questions for 2020. We asked respondents to rate their satisfaction with different subcategories of the web. We chose the categories based on the need themes from 2019. The subcategories were:

- Browser compatibility (differences between implementations)
- Documentation for the Web platform (HTML, CSS, JavaScript, etc.)
- Documentation for frameworks and libraries
- Browser developer tools
- IDEs
- Automated browser testing tools
- Tools for understanding and improving performance
- Tools for understanding and improving accessibility
- Tools for understanding and improving privacy and security

Each of the Satisfaction subcategories are shown below, as compared to the mean overall satisfaction score:

n = 6,645

Satisfaction with the following rates higher than the overall satisfaction, whereas the remaining sub categories score lower than overall satisfaction:

- Documentation for the Web platform (HTML, CSS, JavaScript, etc.)
- Browser developer tools
- IDEs

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.

Changes Between 2019 and 2020

We made changes to the needs list between the first and second iteration of the study, but kept the overall list at 28. There were minor edits to the phrasing of some of the need statements, but more importantly we removed some statements and added new ones.

We removed the following need statements in the 2020 survey:

- Deciding what to learn next to keep my skill set relevant. *(Ranked 19 in 2019)*
- Finding a community of peers. *(Ranked 27 in 2019)*
- Fixing a bug once it's been identified. *(Ranked 28 in 2019)*

Those were replaced with these new need statements:

- Working with different tracking protection and data storage policies in browsers.
- Using web technologies in a native or hybrid context (e.g. using WebViews, Electron, CEF, or mini-apps).
- Lack of support for progressive web apps (PWAs)

Ranking Methodology

Using the Maximum Difference Scaling (MaxDiff) methodology, we asked survey respondents to evaluate a total of 28 need statements. Respondents saw sixteen sets comprising five need statements, ensuring that each of the 28 needs was seen ~3x by each respondent over the length of the exercise. 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 sixteen sets.

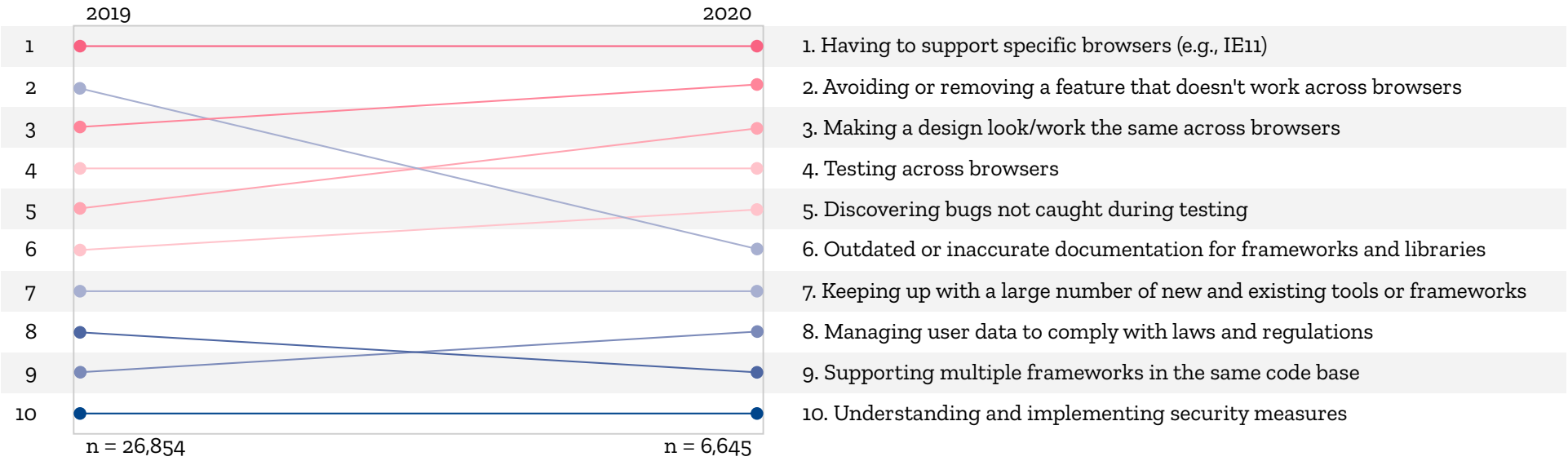
It is important to note that just because a need may not rank as the least frustrating within a set, that does not mean it causes the least frustration. It could imply that the respondent does not have experience with the subject matter or does not prioritize that subject within their work.

For 2020, we employed more sophisticated analysis of the MaxDiff data. We expand on that in the methodology section. In short, this year we:

- Used Python to code, clean, and visualize the data
- Employed the Choice-Based Conjoint/Hierarchical Bayes (CBC/HB) standalone estimation module from Sawtooth Software to estimate MaxDiff utilities
- Eliminated inconsistent responders
- Eliminated speeders who were below a response consistency threshold
- Ratio scaled the needs, converting them to importance (frustration) scores that sum to 100 across the set of items

Top Ten Needs

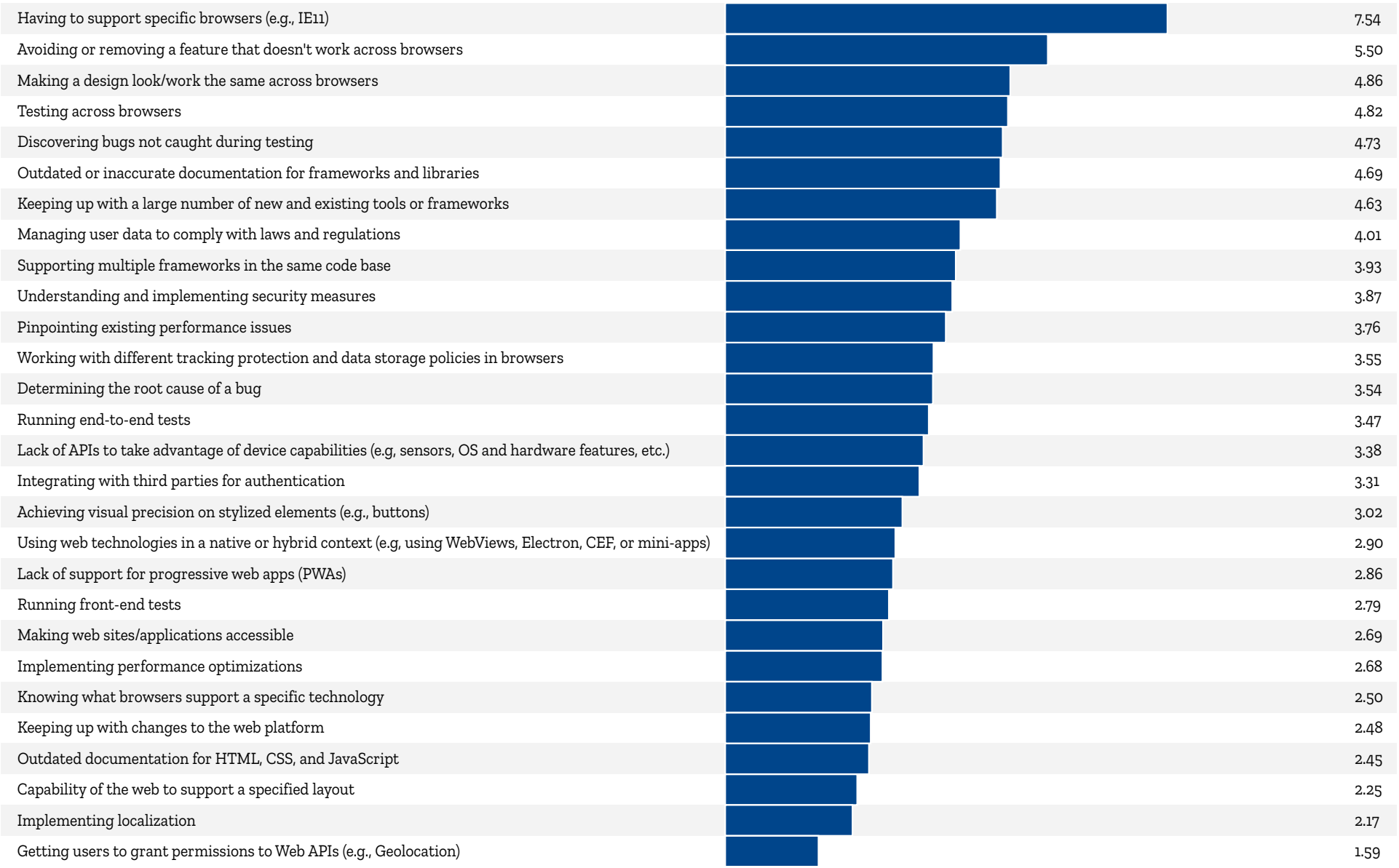
The chart below shows the changes in the top ten needs between 2019 and 2020. The top ten needs stayed the same, however their order changed.



Ranking of All Needs

The chart on the following page display bars of the mean frustration scores for all 28 needs. These sum to 100 across the needs and are ratio scaled. Any mathematical relationship can be evaluated across the items. For example, an item with a score of 6 is 2x more frustrating than an item with a score of 3; an item with a score of 5 is 5x more frustrating than an item with a score of 1. Simply divide the larger score by the smaller score to learn any ratio difference between items.

Ranking of All Needs



n = 6,645

Needs Segmentation

Underneath the surface, there is a large degree of heterogeneity when it comes to developer needs and frustrations. The mean Max-Diff scores can hide this heterogeneity. New for this year are results of a segmentation analysis to better understand the richness of the data. Seven segments emerged. We created somewhat-whimsical names based on which needs pop up as more important for each group.

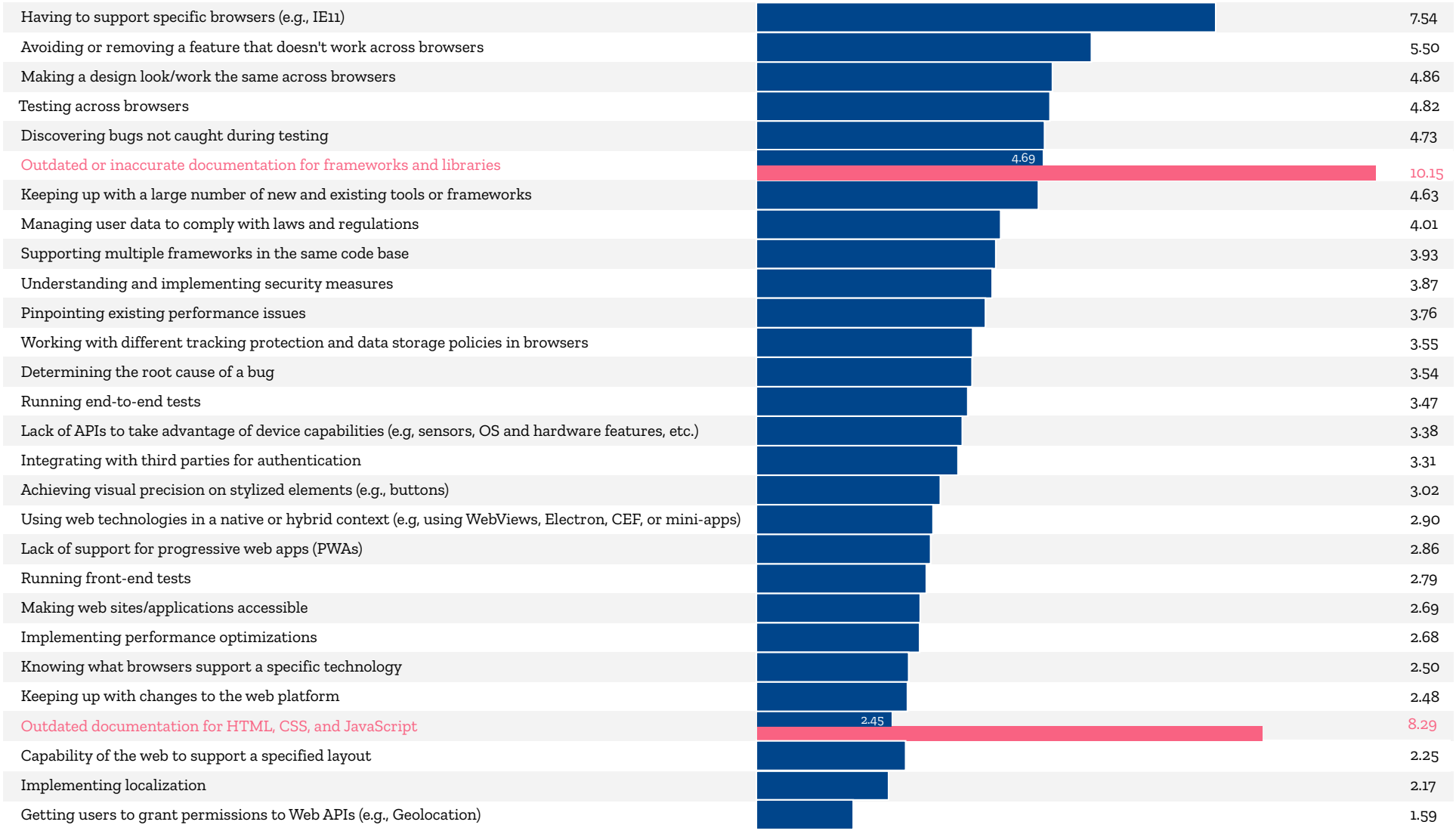
1. Documentation Disciples
2. Browser Beaters
3. Progressive Programmers
4. Testing Technicians
5. Keeping Currents
6. Performance Pushers
7. Regulatory Wranglers

We arrived at these segments from a k-prototypes model with seven clusters or segments. More on how we arrived at these segments, and why we chose the k-prototypes model is in the methodology section.

Each segment has widely divergent needs that surface as the most frustrating when compared to the overall mean scores.

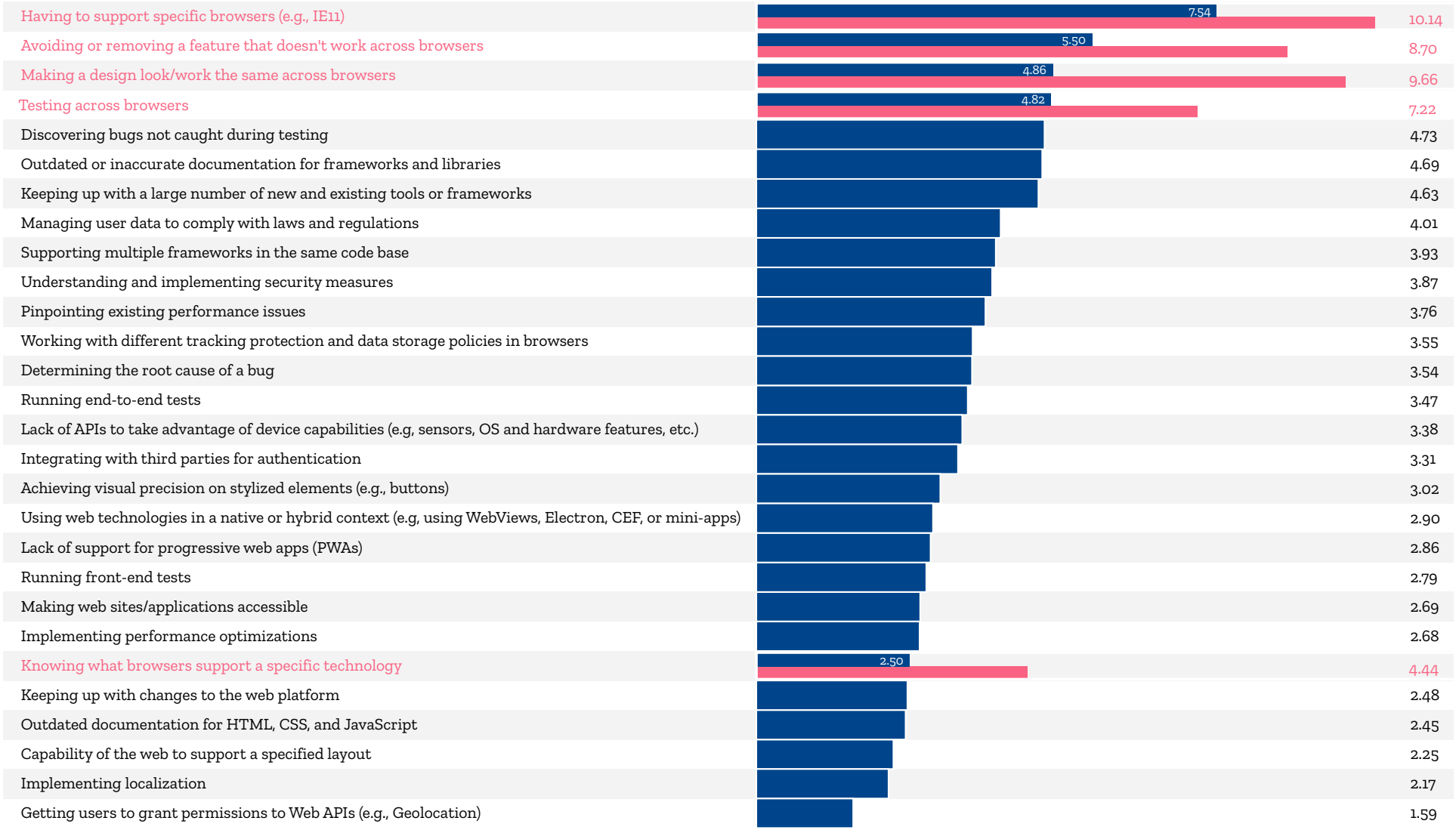
Documentation Disciples

This segment makes up 13% of our respondents. As you may have noticed with the name, their top frustrations are outdated documentation for frameworks and libraries and outdated documentation for HTML, CSS, and JavaScript. Their mean importance scores vary on each need statement, but the chart only highlights the top frustrations for this segment.



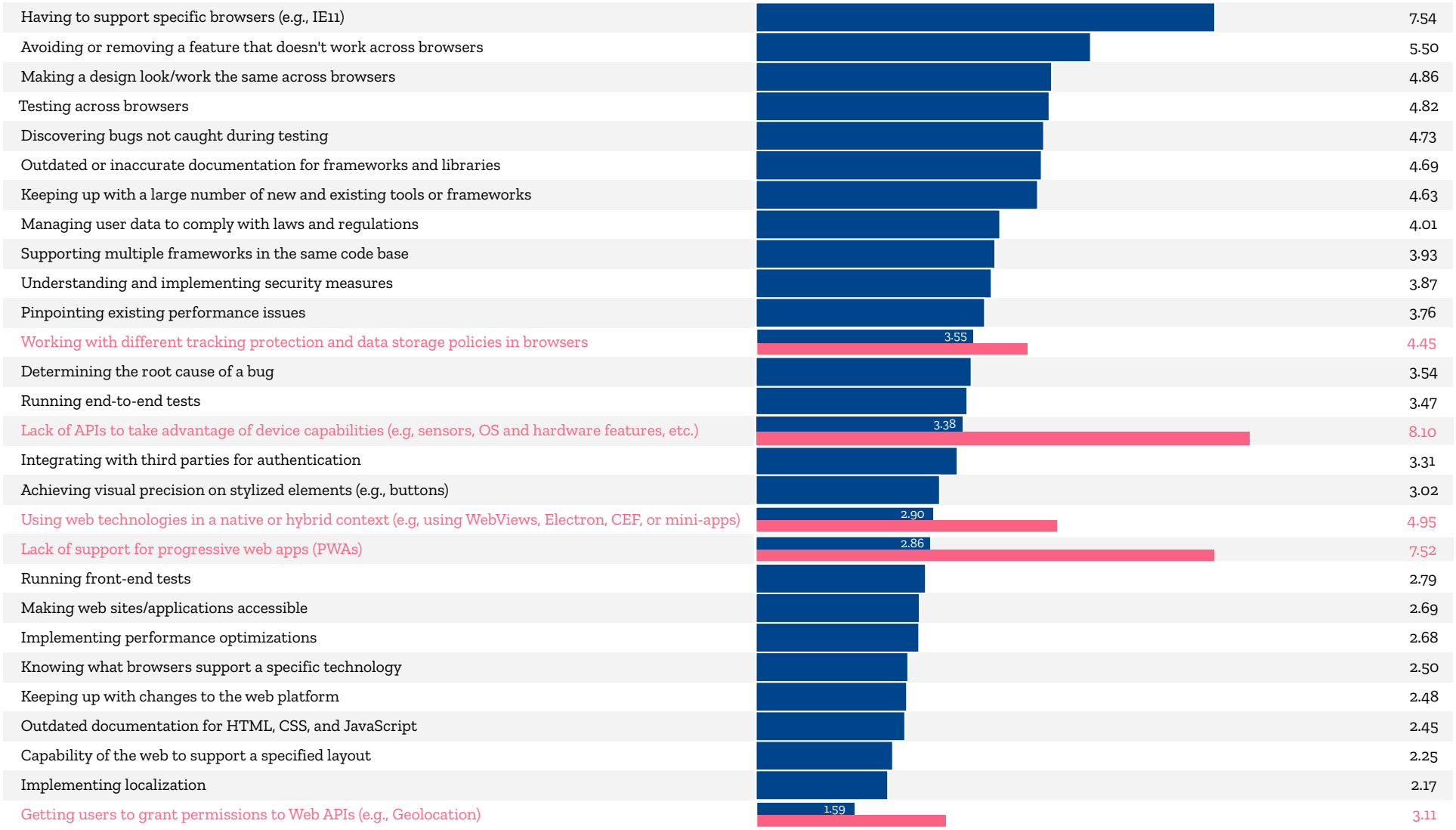
Browser Beaters

This segment makes up 21% of our respondents, and is the largest segment. Their top frustrations are clustered around issues with browser compatibility, design and layout. Their mean importance scores vary on each need statement, but the chart only highlights the top frustrations for this segment.



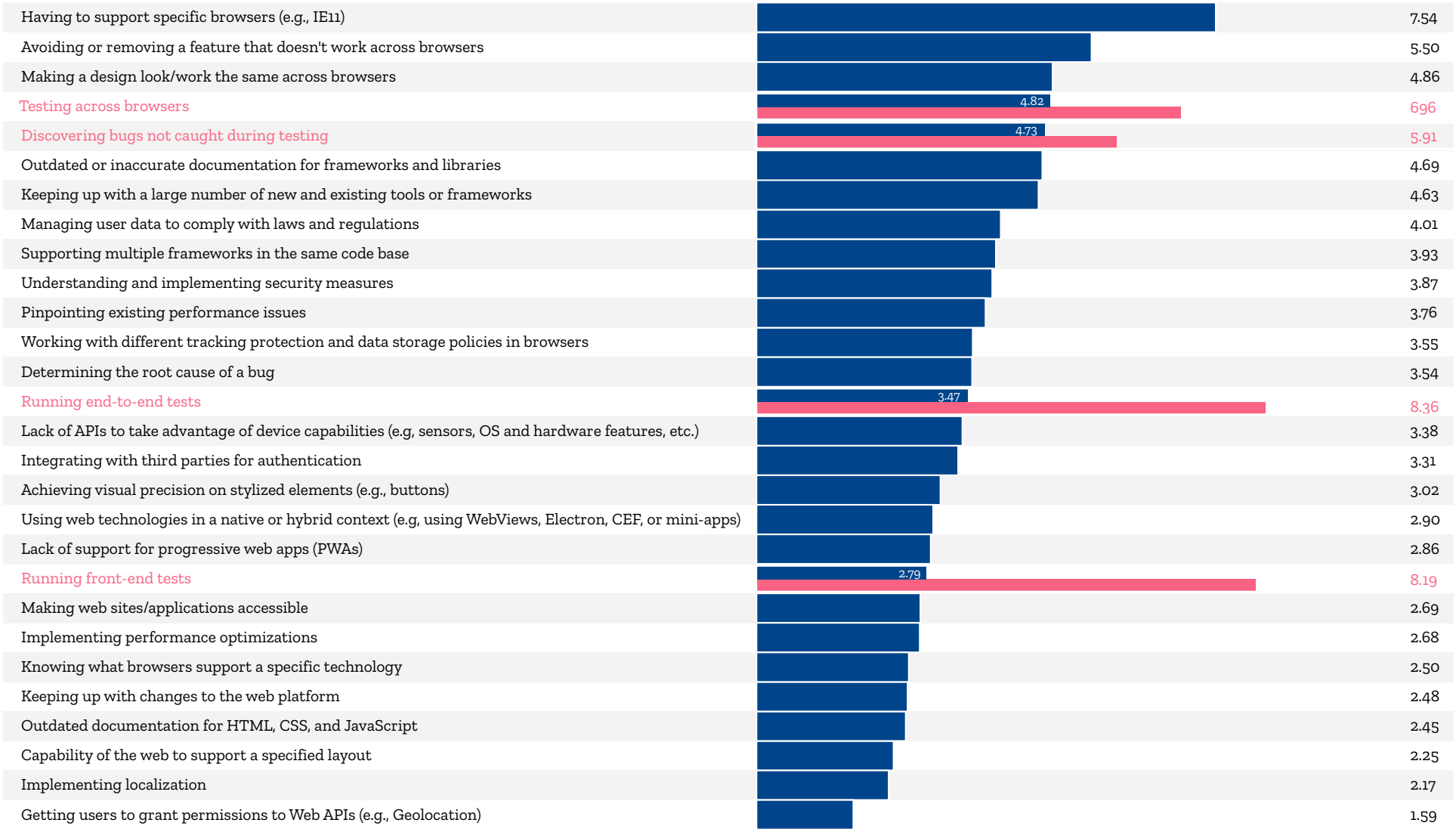
Progressive Programmers

This segment makes up 11% of our respondents. Their top frustrations are clustered around lack of APIs, lack of support for Progressive Web Apps (PWAs), and using web technologies. For them, browser related needs were typically less frustrating than the overall mean. Their mean importance scores vary on each need statement, but the chart only highlights the top frustrations for this segment.



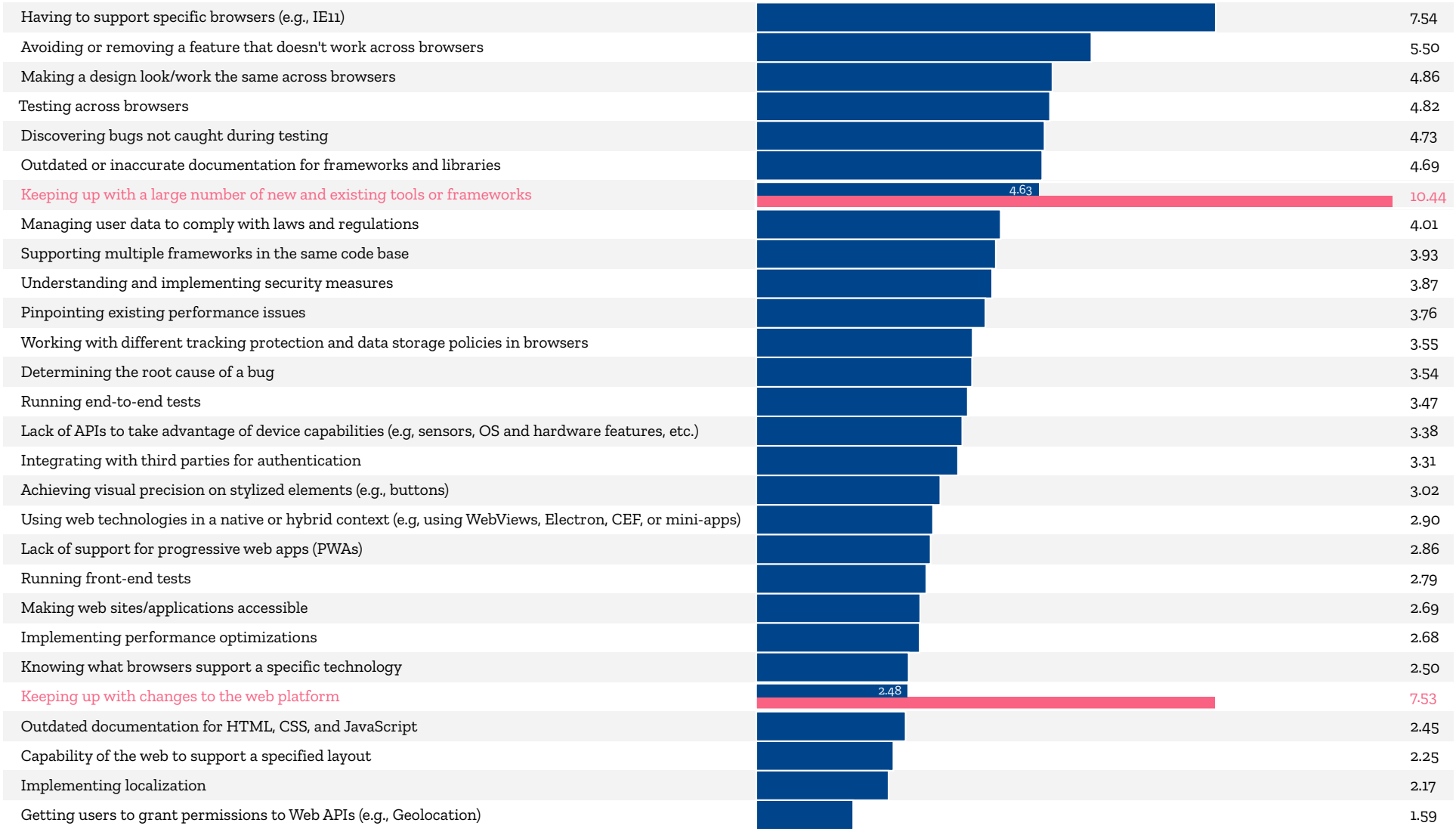
Testing Technicians

This segment makes up 13% of our respondents. Needs statements relating to testing, whether end-to-end, front-end, or testing across browsers, caused the most frustration for this segment. Their mean importance scores vary on each need statement, but the chart only highlights the top frustrations for this segment.



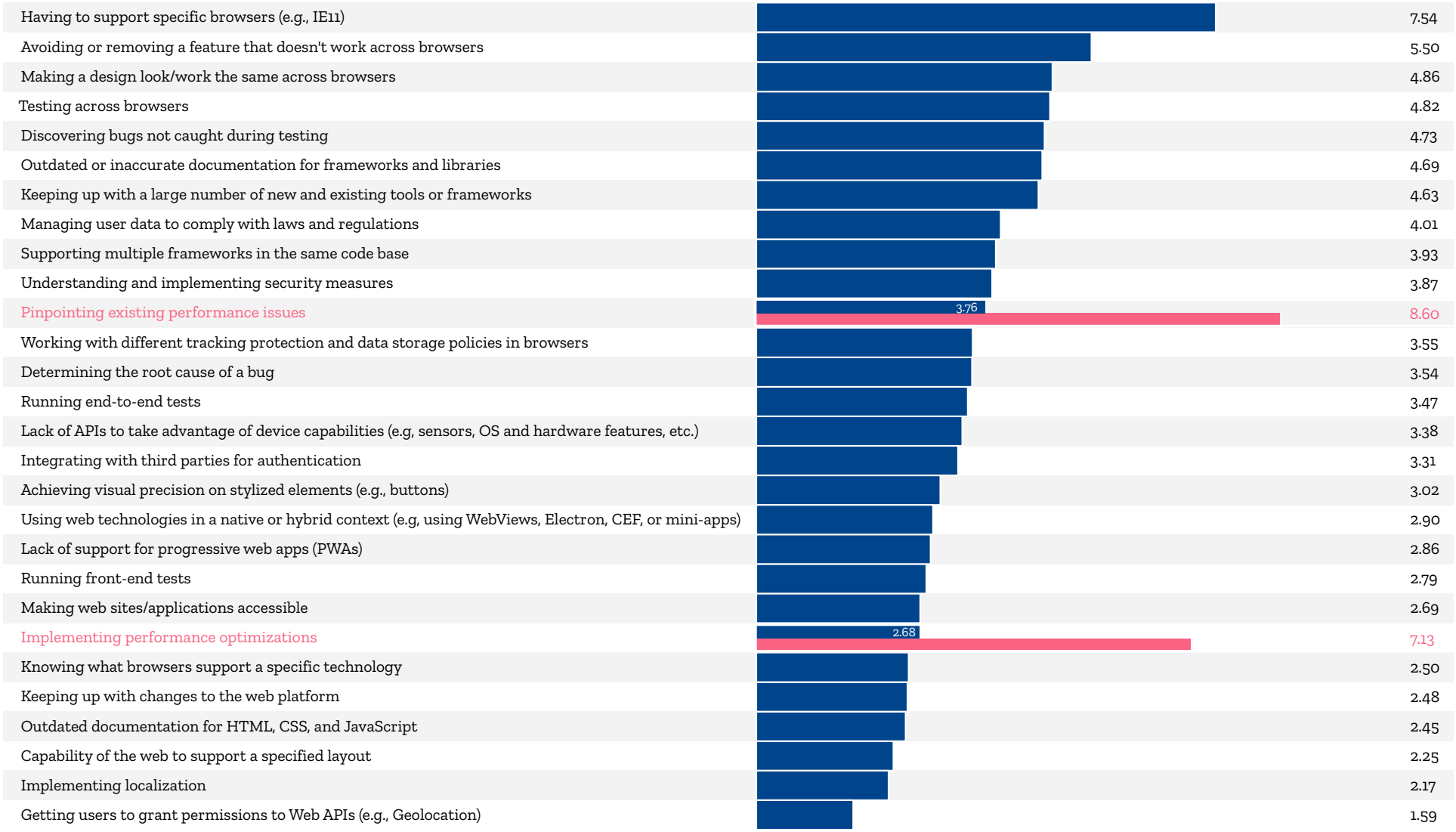
Keeping Currents

This segment makes up 13% of our respondents. The need statements that this segment found most frustrating were keeping up with a large number of new and existing tools and frameworks and keeping up with changes to the web platform. Their mean importance scores vary on each need statement, but the chart only highlights the top frustrations for this segment.



Performance Pushers

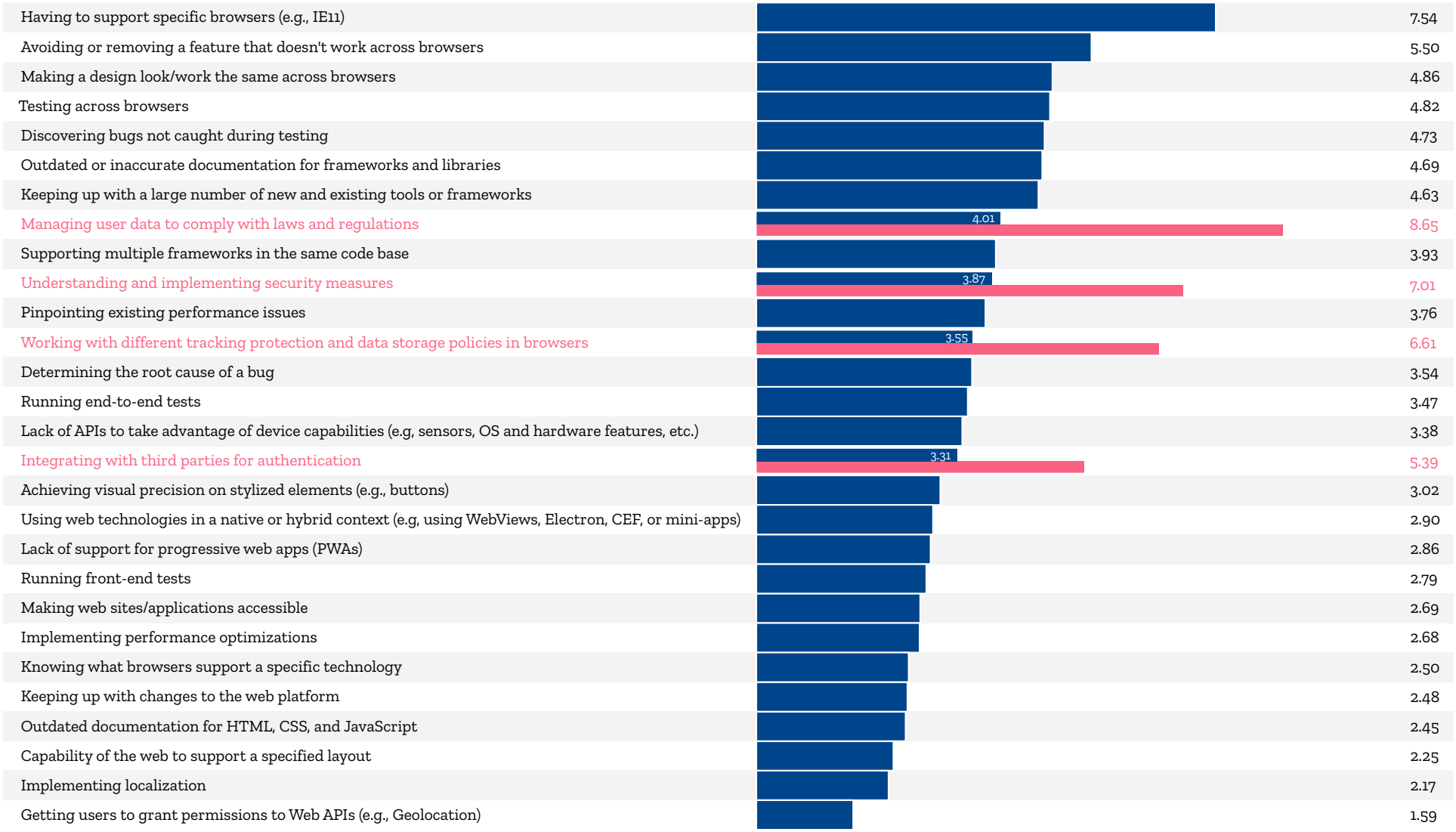
This segment makes up 15% of our respondents. Needs statements relating to performance and bugs are the top frustrations for this segment. Needs related to testing were rated as less frustrating than the overall mean, but discovering bugs not caught during testing is higher. Their mean importance scores vary on each need statement, but the chart only highlights the top frustrations for this segment.



n = 6,645
n = 979

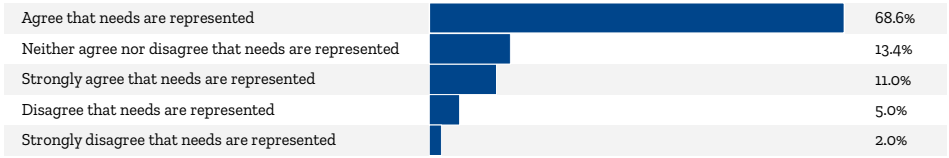
Regulatory Wranglers

This segment makes up 14% of our respondents. This is the more eclectic segment, with a bigger assortment of needs rating higher than the overall mean. However, compliance with laws and regulations for managing user data is the most frustrating need. Their mean importance scores vary on each need statement, but the chart only highlights the top frustrations for this segment.



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, 13.4% neither agreed nor disagreed which means there is room for improvement in the needs list. This is an improvement from 2019, where 21.6% neither agreed nor disagreed.



n = 6,645

What's Missing From the Web

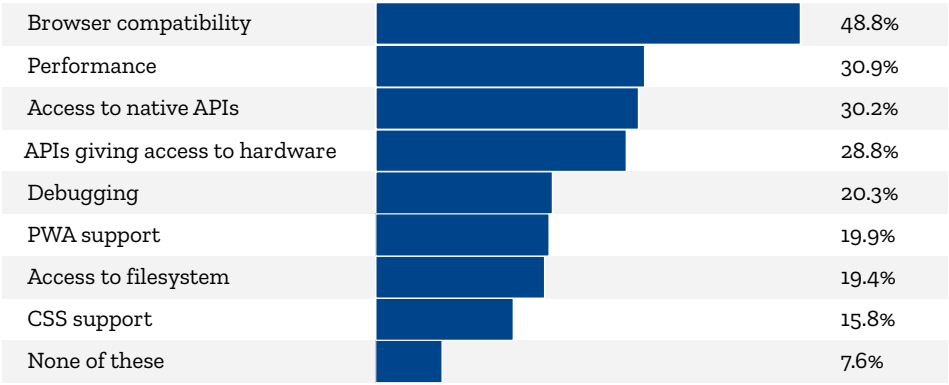
What's Missing From the Web

In 2019, we included the open-ended, "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.

Last year, we hand categorized the answers. We took a random 1,000 answers and manually categorized them into 109 categories, up to three categories per answer. We used the first or most prominent issue as the first category. Of the 109 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%)

We added a new question this year that built upon what we learned from the open-ended answers from last year. We increased the list above by adding CSS support, which accounted for 2.9% of the categorized answers from last year. The question was, "Which do you feel are most lacking from the web platform?" Respondents were allowed to select up to three. Browser Compatibility was the most selected option at 48.8% of the responses.



n = 6645

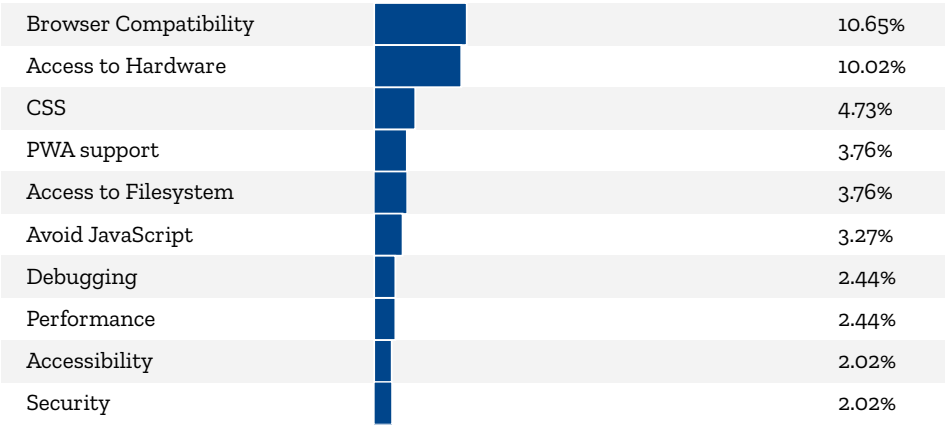
What's Missing From the Web

Following the above question, we added an open-ended question, "What other things would you like to be able to do on the Web but lack web platform features to do?" This is a slight word change between 2019 and 2020 with the phrase, 'what other things,' being key. However, that nuance may have been lost on respondents as evidenced by the analysis of their responses or they reiterated how important an option was to them.

To analyze the open-ended responses from this year, we employed natural language processing techniques. Specifically, we trained a decision-tree-based model on the 2019 answers and categories. The caveats are, it only works on a limited number of categories, and it can only assign one category per response. On the 2019 data, it predicted one of the three categories 92% of the time.

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 1,437.

The model used 60 categories. However, of those categories and removing 'Other,' only ten had 2% or more of the answers.



n = 648

The results of the open-ended responses match up pretty well with the options provided in the question before the open-ended question, with accessibility and security bubbling up as strong contenders for what's missing from the web. Select verbatims that help convey the deeper meaning of the category are on the following pages.

What's Missing From the Web

Browser Compatibility

"Browser cross-compatibility and extensibility in secure configurations under which plugins are created for platforms."

"Fully cross-browser compatibility. No 'implementation-specific' points in standards or drafts."

"I'd love to write some HTML/CSS and not have it behave differently in six months when the specs/browsers change yet again."

Access to Hardware

"Hardware/Native API. We do a lot of automation to support doctors, think automatically position windows across multiple screens among other things. We currently have to install a desktop app that the website can talk to make this work well. That combined with dictation software creates a barrier between us and the doctors."

It's worth noting that the following quote is a bit of an outlier in that they are asking for something many web developers are asking for, however, there's a lack of trust that browser vendors will implement it fairly.

"As a developer, I'd like native and hardware API access in order to build a wider range of software on the web platform. But the certainty that such APIs would instantly be abused, and that browser vendors (looking at you, Google) would fail to provide sufficiently strong and user-comprehensible security and privacy controls, leads me to hope those APIs never ship."

"User-allowed access to hardware."

CSS

"Tons of CSS things.. better attr(), better calc()..."

"Support more features only available via CSS preprocessors (e.g. SCSS), e.g. loops, conditions, macros, more functions."

"Support for CSS across browsers is super important to me -- I am one of the only developers in my office who works on CSS but it has the biggest impact on what our clients care about (branding, etc)."

Access to Filesystem

"I clicked Access to filesystem, but to emphasize it, I put it here, too."

"Full access to a folder for media management. Persistent state storage on the filesystem, not subject to the whims of browser storage cleanup."

"Having access to a virtual filesystem without all of the extra steps and external libraries normally associated with it."

PWA Support

"I would love to make a webapp, that Chrome/Chromium doesn't break with each third release, because they hate users so hard. Other than that: PWA on desktop, not mobile."

"The problem is that PWA doesn't work in Safari, and there's no way to run chrome on iOS, so we are blocked on iOS."

"Bundle a PWA in a single file to distribution, such as a zip file that can be run like a native app from the OS."

What's Missing From the Web

Avoid JavaScript

"Completely Delete JavaScript from the whole Universe."

"The lack of precise decimal numbers in JavaScript has been a pain point for a long time particularly considering that the apps I work on have to deal primarily with money. If there is one thing I would love to have in JavaScript is a decimal type like the one in .Net."

"Basically I want all browsers to be rewritten to use WebAssembly on top of which DOM/DOM manipulation is implemented on top of which, next level up, JavaScript, etc. are implemented in the VM/IL virtual assembly op code language of the WebAssembly VM implementation."

Performance

"Better bind back and frontend. Node/TS helps a lot, but the solutions are immature and pretty much non-performant, and that severely limits the amount of stuff that can be done on the web. Nowadays browsers have incredible capabilities, and are as complex as an OS, so I would love to see support for more complex, performant, and united apps."

"Build high-performance solutions (C++ like performant) with strong graphical features. An app that would resemble an AAA video game but in the browser. I'm looking forward to WASM going into mainstream."

"Rendering performance and options are disappointing compared to native. The primary issue is the 1-2 frames of additional delay introduced by the web browser when using canvas/WebGL to display mouse/keyboard inputs. Configuring vsync/gsync/high refresh rate can also be frustrating."

Debugging

"Easier debugging of browser apps from command line for automated testing. Yes, there are hacky ways to do it, but first class ways of debugging (tests passed/tests failed) from the command line would be such a time saver."

"Debugging should be more traceable. For example using Vue or other web frameworks sometimes an error is marked as if the root cause were the framework (does not pinpoint the correct js file)."

"Containers have made the use of a lot of debugging/dev tools harder or not able to be used."

Security

"While I still may be new to the coding journey, security...on the internet is still the number one concern. Personally, I feel the complex jargon used to describe how web security functions in the modern era makes learning and implementing best security practices one of the most difficult things as a new programmer, even if you go with the third-party options."

"Proper Firefox extensions, I would love to go back to the state before crippling them with webextensions and questionable "security" decisions (companies should allow users to install extensions from anyone, not just subjects approved by browser companies. You cannot in Firefox nor Chrome fully install an extension which was not signed by Mozilla/Google)."

"Interact with OS specific APIs to store and receive secrets like e.g. PGP keys in order to allow E2E and "crowd" encryption. Currently it is

only possible to store such secrets in the local storage which is a security risk and can end in data loss when the user clears the data."

Accessibility

"Accessibility tools and features are grossly lacking."

"Accessibility support beyond most trivial of basics requires heavy research, and knowing if you're doing it well is [shrug].

"Accessibility, screen reader supports are chaos right now. As every application is trying to be WCAG compliant it make sense [that] ARIA comes by default or have better APIs to handle."

Technologies

Programming Languages

Programming Languages

Being core programming languages for the web, we wanted to know what pain points developers have when using JavaScript, HTML, CSS, and WebAssembly.

In 2019, for each of the languages we asked, "What are the biggest pain points for you when it comes to [programming language] development? Select all that apply."

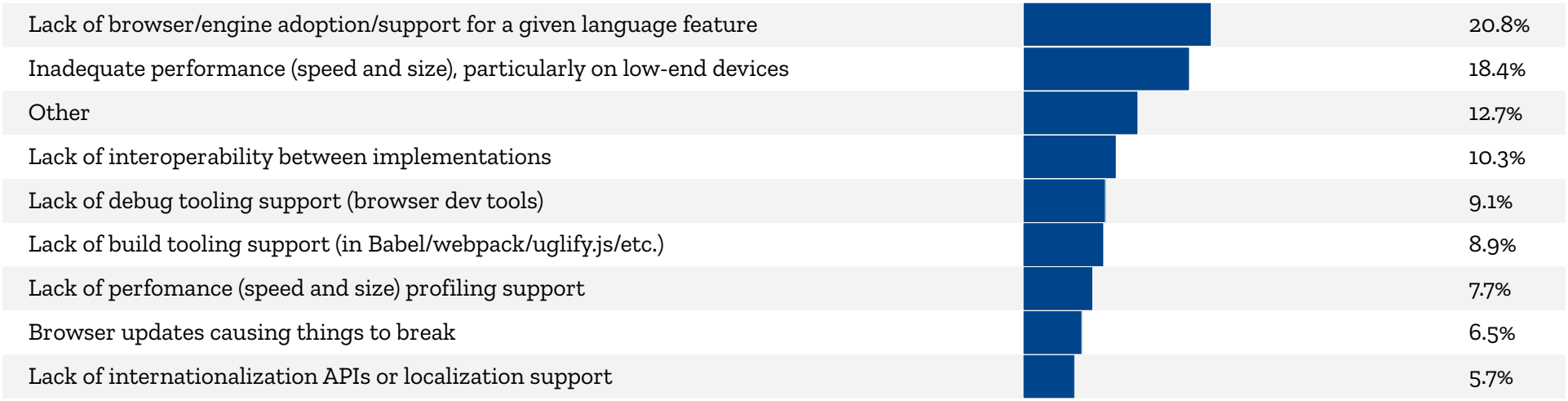
We asked the same question, but new for this year, we then followed up with a second question, taking all the answers selected from the previous question, we asked them to pick the biggest. This allowed us to report more accurate numbers in terms of which is truly the biggest pain point.

JavaScript

2.5% of our overall respondents said they do not use JavaScript. Of those who do use JavaScript, 14.9% said they have no pain points. Of those who do have issues with JavaScript, the biggest pain point is the same as last year, "Lack of browser/engine adoption/support for a given language feature." Other ranks third, which suggests there are pain points not captured in our list.

We had respondents who selected JavaScript as one of the languages they use define where they are using JavaScript:

- 45.1% are using JavaScript on a browser
- .8% are using JavaScript on a server
- 51.6% are using JavaScript on both a browser and a server



n = 5,472

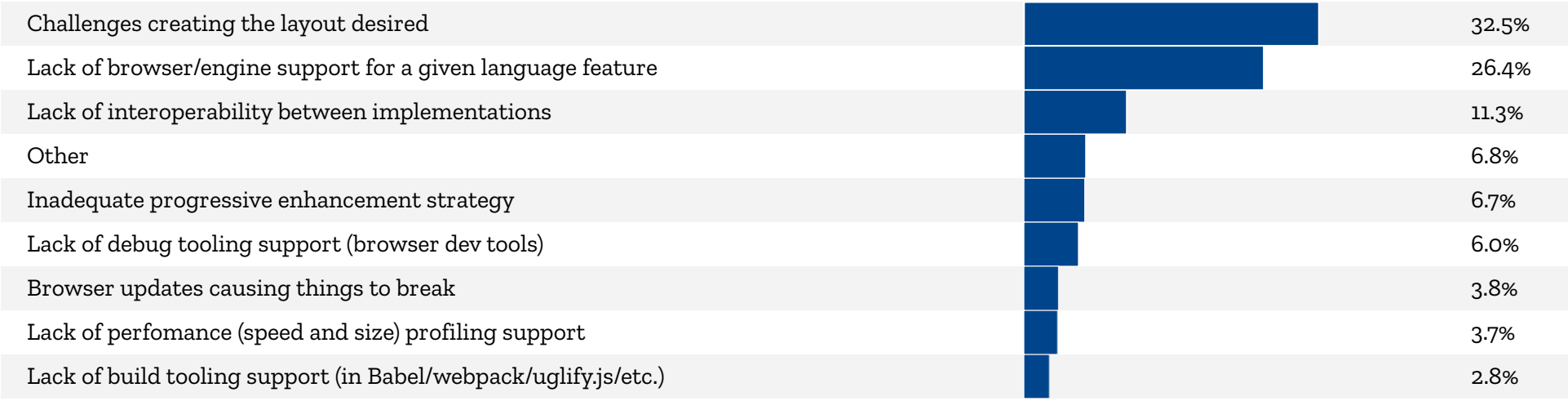
HTML

For those who use HTML, 32.6% said they have no pain points. Of those who do have issues with HTML, the biggest pain point is the same as JavaScript, "Lack of browser/engine adoption/support for a given language feature." A close second is, "Inability to customize components built into HTML."

n = 4,063

CSS

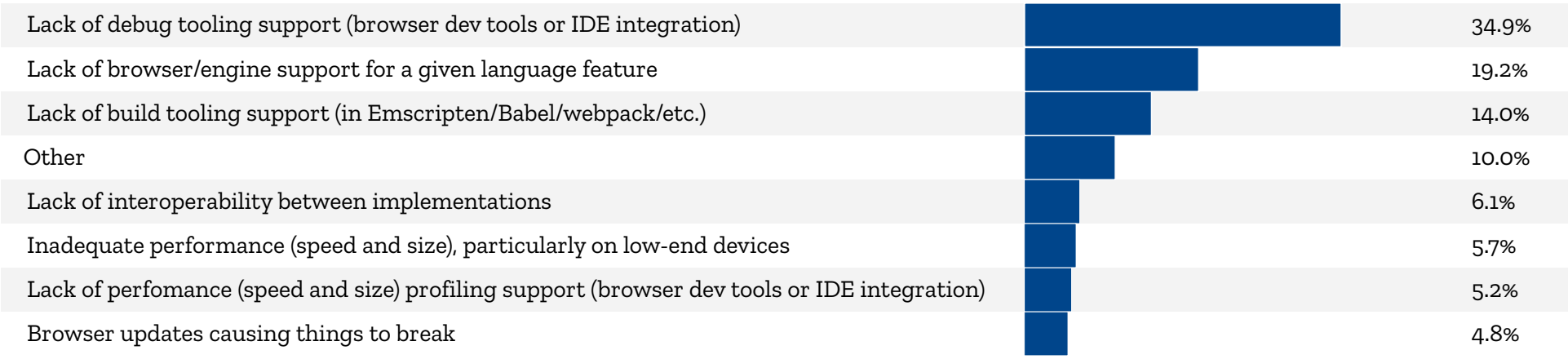
For those who use CSS, 12.5% said they have no pain points. Of those who do use CSS, 32.5% said their biggest pain point is challenges creating the layout specified. This was the same pain point as last year. Similar to JavaScript, Other ranks high on the lists, which suggests there are pain points not captured in our list.



n = 5,017

WebAssembly

For those who use WebAssembly, 20.8% said they have no pain points. Since Web Assembly is still considered a relatively new language, the respondents who use the language were able to provide information about their pain points. The largest pain, with 34.9%, is a lack of debug tooling support. This was the same pain point as last year. Like CSS, Other ranks fourth, which suggests there are pain points not captured in our list.



Lack of debug tooling support (browser dev tools or IDE integration)	34.9%
Lack of browser/engine support for a given language feature	19.2%
Lack of build tooling support (in Emscripten/Babel/webpack/etc.)	14.0%
Other	10.0%
Lack of interoperability between implementations	6.1%
Inadequate performance (speed and size), particularly on low-end devices	5.7%
Lack of perfomance (speed and size) profiling support (browser dev tools or IDE integration)	5.2%
Browser updates causing things to break	4.8%

n = 229

Adoption of New Technologies

Adoption of New Technologies

The biggest barrier developers face when adopting a new technology is broad interoperability across browsers, which is the same as last year. A close second is support for legacy browsers. Considering that having to support specific browsers is the overall number one frustration developers have when developing for the web, it's not surprising that barriers to adopting new technologies are related to browser compatibility.

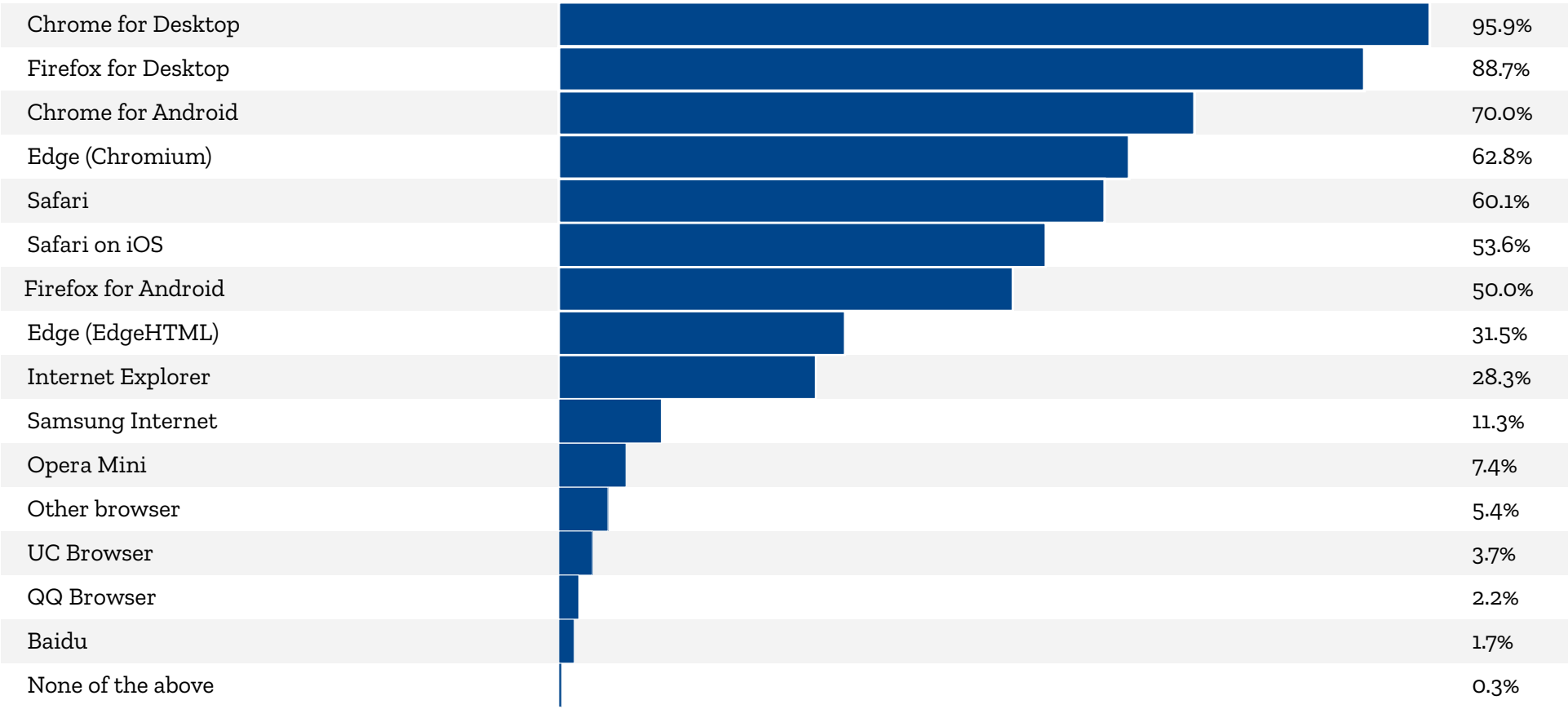
New for this year was the answer, "Organizational approval."

n = 5,526

Browsers

Browsers Developers' Support

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

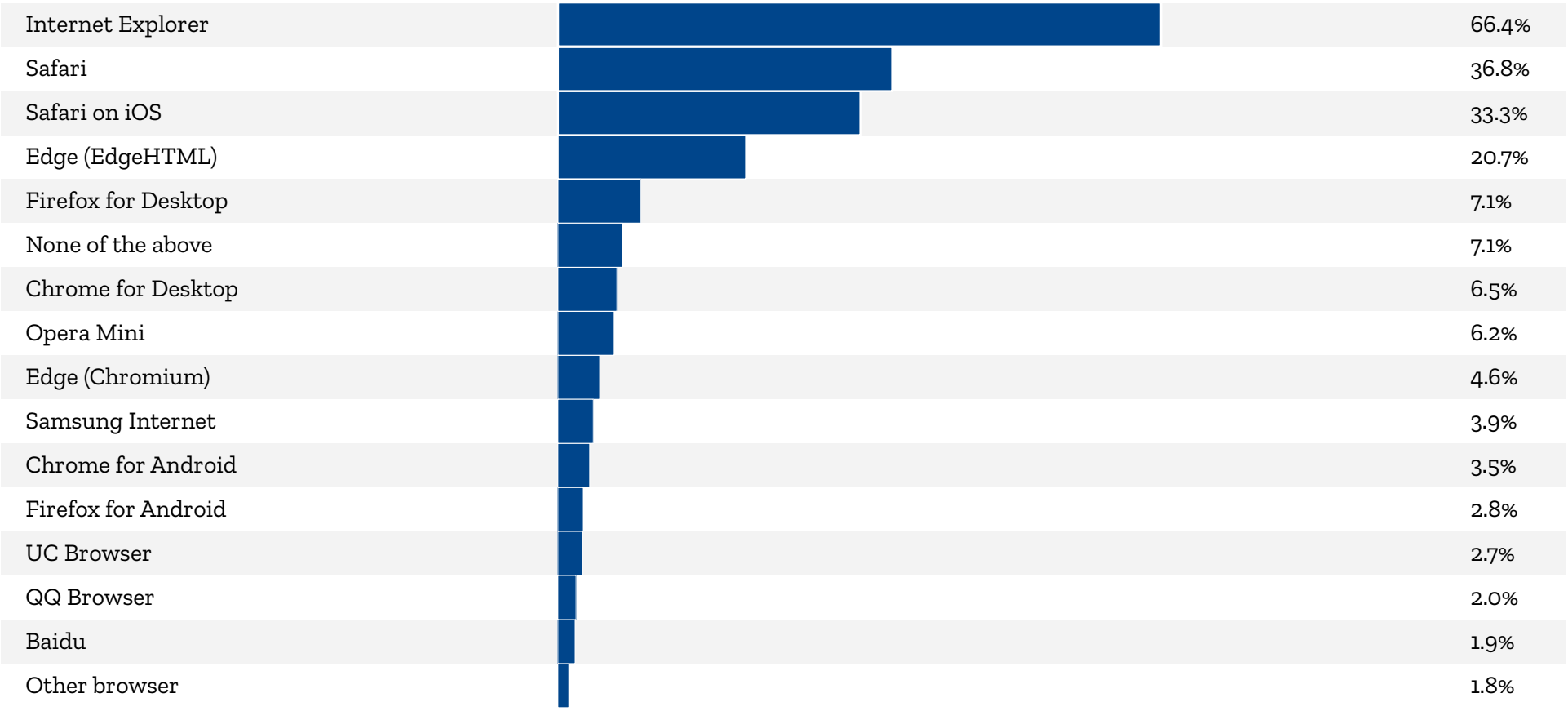


n = 6,645

Browsers That Cause Issues

We asked developers to rank which browsers cause the most issues and they were allowed to select up to three.

Though Internet Explorer is only supported by 28.3% of respondents, it causes the most issues for developers. Safari and Safari on iOS were distant second and third contenders.



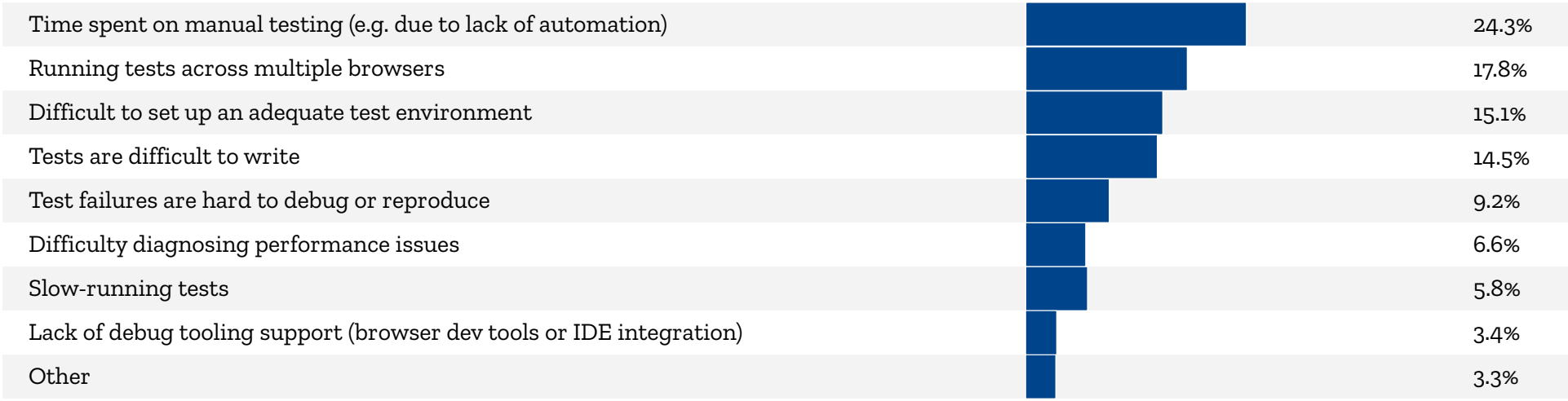
n = 6,645

Web Testing

Web Testing

We added a new question this year, "What are the biggest pain points for you when it comes to web testing?" What motivated this addition was the need "Testing across browsers" which ranked #4 last year as well as this year. We wanted to understand more about this need and what some of the underlying issues might be. 7.5% of respondents said they don't have pain points with web testing. For those who did, the biggest pain point is the time spent on manual testing.

One way to interpret these results is that the need is not merely an echo of browser compatibility issues and having to test multiple browsers. That's part of it, but where tests are automated, the next difficulties are about cross-browser testing, and setting up the test environment. This wouldn't automatically improve if the browser compatibility problem got better.



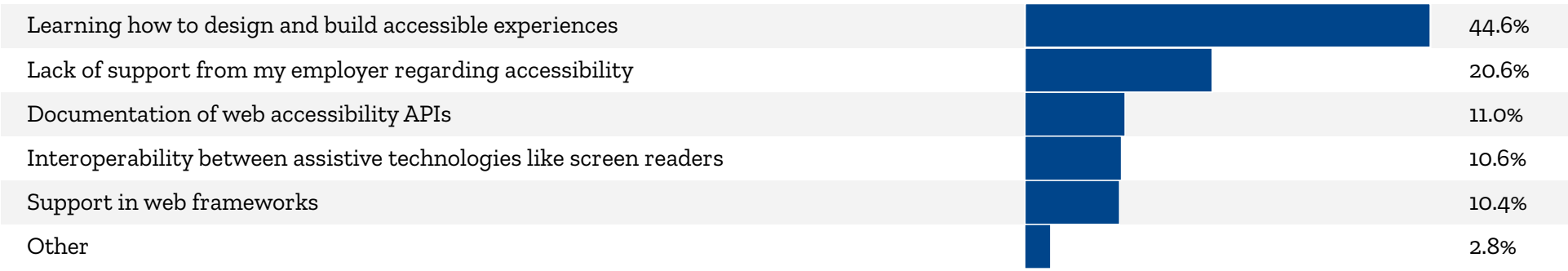
n =6,144

Accessibility

Accessibility

In the need rankings from last year, "Making sites accessible," ranked 24 out of 28. This year, it's 21 out of 28. In the needs section we noted 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. We used accessibility as an example in last year's report based on the pilot interview findings. We learned that developers 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 can spend more time on accessibility, then their perception of the frustration may change, and so would the ranking.

To get a better understanding of accessibility, we added a question to this year's survey. We asked, "What are the biggest pain points for you when it comes to web accessibility?" 16% of respondents said they don't have pain points with accessibility. For those who answered, the biggest pain point is learning how to design and build accessible experiences. The second pain point is a lack of support from my employer regarding accessibility.



n = 5,526

Conclusion

Conclusion

Respondents took this survey with the promise that their answers will influence how browser vendors prioritize feature development. That work is underway.

Google

"The MDN DNA report bridges a critical gap in understanding developer needs: replacing guesswork with actionable feedback. We will use it in 2021 to focus our work on improving the areas of the web platform that cause the most pain. Since the report shows that it's not just bugs or gaps in key areas, but also interoperability of those areas across browsers, we will work closely with other browser vendors to drive improvements across the board."

- Chris Harrelson, Senior Staff Software Engineer and Blink Rendering Lead

Microsoft

The Edge team used the Web DNA survey results from 2019 to inform their product roadmaps, understand the current needs of web developers, and to discover unmet needs. This has been a valuable data point for their platform, apps, and tools teams to direct planning and inform customer research. The results underscored the importance of improving compatibility solutions and understanding new tooling opportunities, while reinforcing the general principles behind their choice to move Microsoft Edge to Chromium, collaborate in open source, and ship across platforms. They're eagerly digging into the 2020 results and look forward to continuing to improve their plans based on this year's data.

Mozilla

"The MDN DNA report provides the Web Platform team at Mozilla with critical insight into how we can improve the platform for authors. The results of the survey highlight the importance of cross-browser compatibility, an area which we regard as critical for the health of the web. We will use the survey data to inform the 2021 planning for Gecko, Firefox's rendering engine, and help us focus our efforts on those areas which will have the biggest impact on web developers."

-Andrew Overholt, Senior Director of Engineering, Web Platform

Methodology

Methodology

MaxDiff

This year, we took a more thorough data analysis approach by employing data science best practices. This includes:

- Using Python to code, clean, and visualize the data
- Employing the Choice-Based Conjoint/Hierarchical Bayes (CBC/HB) standalone estimation module from Sawtooth Software to estimate MaxDiff utilities
- Eliminating inconsistencies, e.g., in different sets, a respondent may see 'Determining the root cause of a bug' and 'Discovering bugs not caught during testing.' In one set, they may indicate that 'Determining the root cause of a bug' is more frustrating; in a different set, they switched the order. After HB estimation, we utilize a statistic called Root Likelihood (RLH) to determine cutoffs for inconsistent/random responders, using a set of simulated random data to help determine the proper cutoff. Respondents failing to exceed the minimum RLH are removed from further analysis.
- Eliminating speeders, respondents who move through the survey in an impossibly short amount of time. We calculate the time of completion and drop anybody who is < ½ the median completion time, AND who also have an RLH < 0.5 (RLH ranges from 0 to 1, 1 being better). So in addition to the inconsistent responders removed above, we remove speeders who fail to meet a higher consistency threshold as well. We don't remove speeders who are consistent in their answers, in other words.

Needs Segmentation

To uncover the segments, we utilized four different unsupervised machine learning techniques (kmeans, hierarchical clustering, Archetype analysis, and k-prototypes), and evaluated multiple possible solutions with differing numbers of segments (over 40 solutions evaluated overall).

For each technique, we evaluated the candidate solutions using various statistical criteria, choosing a best candidate for each of the four approaches. These solutions were then profiled and evaluated more artistically on which told a better story. The solution we settled on and included in this report comes from a k-prototypes model with 7 clusters or segments.


MDN web docs
moz://a
pinpoint