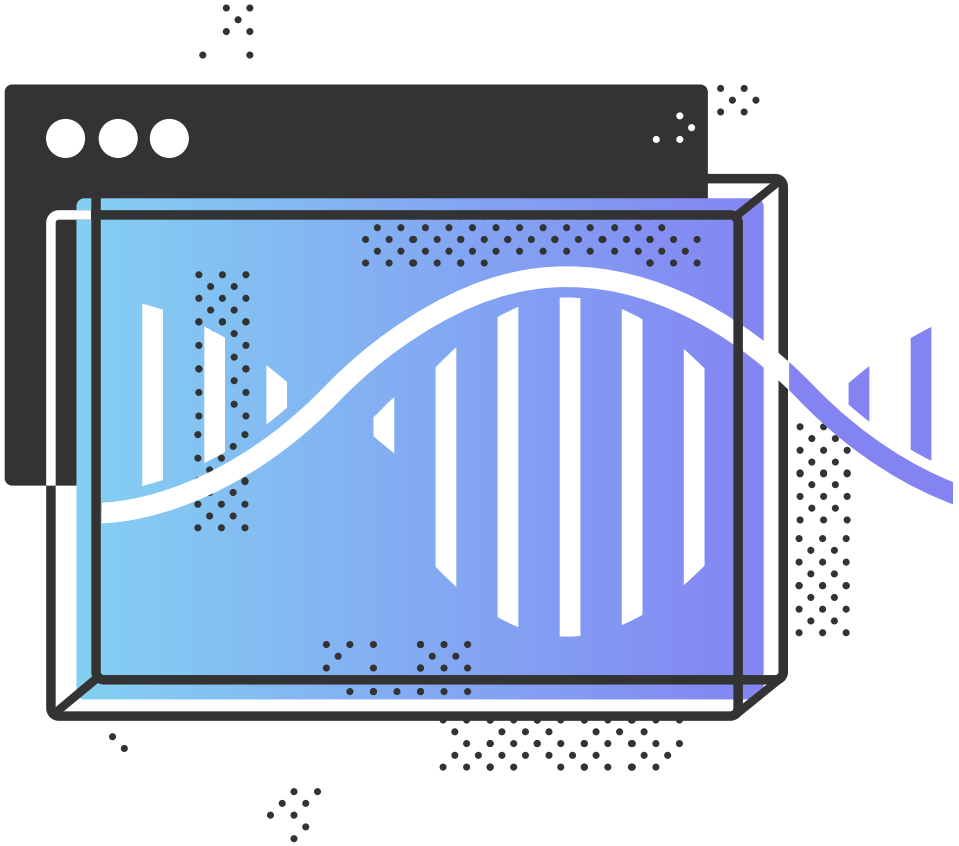
MDN Web Testing



MDN web docs

moz://a

Acknowledgements

The web testing report would not be possible without many contributions. The core team responsible for executing the study, analyzing the results, and publishing the report are:

- Michael Hablich, Philip Jägenstedt, and Robert Nyman at Google
- Kyle Pflug, Mateo Chavez, and Zoher Ghadyali at Microsoft
- Chris Mills, James Graham, and Nancy Hang at Mozilla
- Allison McKeever and John Nolan at Pinpoint

We would like to thank the MDN Product Advisory Board members for reviewing the study goals design and/or this report: Chris Mills (Mozilla), Daniel Appelquist (Samsung), Dominique Hazaël-Massieux (W3C), Joe Medley (Google), Sheila Moussavi (Bocoup), Kyle Pflug (Microsoft), and Reeza Ali (Microsoft).

Finally, a huge thank you to our anonymous interview participants, who were generous with their time and helped us get a clearer picture of web testing issues.

Table of Contents

Introduction

Why Web Testing?

In the MDN Web DNA study for 2019 and 2020, developers ranked the need "Having to support specific browsers, (e.g., IE11)" as the most frustrating aspect of developing for the web, among 28 needs. The second and third most frustrating needs also related to browser compatibility:

2) Avoiding or removing a feature that doesn't work across browsers
3) Making a design look/work the same across browsers

In 2020, we released our browser compatibility research results, which offer a deeper dive into attempting to identify specific issues about browser compatibility that causes frustration and what could be done to improve the situation.

Having already focused our attention on browser compatibility, this year, we focused on the fourth most frustrating aspect of developing for the web, "Testing across browsers."

Based on the 2019 ranking of "testing across browsers," we introduced a new question to the DNA survey in 2020, "What are the biggest pain points for you when it comes to web testing?" We wanted to understand more about this need and the underlying issues. Respondents could choose among the following answers:

- Time spent on manual testing (e.g. due to lack of automation)
- Slow-running tests
- Running tests across multiple browsers
- Test failures are hard to debug or reproduce
- Lack of debug tooling support (browser dev tools or IDE integration)
- Difficulty diagnosing performance issues
- Tests are difficult to write
- Difficult to set up an adequate test environment
- No pain points
- Other

In 2020, 7.5% of respondents (out of 6,645) 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.

To better understand the nuances behind these results, we ran a qualitative study on web testing. The study consisted of twenty one-hour interviews with web developers who took the 2020 DNA survey and agreed to participate in follow-up research.

The results will help us understand whether we accelerate work on WebDriver Bidirectional Protocol (BiDi) or if the unmet needs lie elsewhere. Our work on WebDriver BiDi is based on the assumption that the feature gap between single-browser test tooling and cross-browser test tooling is a source of pain. More detail on the struggles developers have will focus the priorities and technical design of that specification to address the pain points.

Study Respondents

Respondent Overview

We recruited 20 participants for this study from their responses to the MDN Wed DNA 2020 survey. We only reached out to respondents who fit within the Testing Technicians segment.

Of the twenty participants, most were in the United States, but we did have developers from:

- Australia
- Germany
- India
- Ukraine

Type of Developer & Years of Experience	Number of Participants
Front-end developer, >1 year of experience	1
Front-end developer, 1-2 years of experience	1
Front-end developer, 3-5 years of experience	2
Front-end developer, 6-9 years of experience	2
Front-end developer, 10+ years of experience	4
Front-end developer & Build Engineer/DevOps	1
Full-stack developer, 1-2 years of experience	2
Full-stack developer, 3-5 years of experience	4
Full-stack developer, 6-9 years of experience	1
Full-stack developer, 10+ years of experience	2

Critical Themes

Critical Themes

We knew going into this study that web developers experience frustration with testing. We wanted to learn why. What we found is that it depends, and what it depends on is nuanced.

Testing Technicians
In the MDN Web DNA report for 2020, we reported the results of our segmentation study. One of the seven segments that emerged was 'Testing Technicians.' The somewhat-whimsical names created based on which needs pop up as more important for each group don't capture all the frustration's nuances. The name implies that the segment does testing and therefore find frustration while doing tests. This is true, but what's also true is that developers see a high barrier to entry to testing, and that contributes to their frustration.

Testing Takes Time, and Time is a Precious Resource
Defining a testing workflow, choosing tools, writing tests, and running tests are all things that take time. Many developers face pressure to develop and launch under tight deadlines. Testing or not testing is a tradeoff between the perceived value that testing adds compared to the time it will take to implement.

> *"...this is the crummy reality of most web development shops. There's a schedule, unrealistic expectations, and deadlines. Most of the time, developers get pressed up against that. When you are in the corner, and you have to have a product by a certain time, testing even though it might pay off in the long run, is not what you think about in the short term. It's hard to get into this complicated process of setting up a test environment and making it reliable, and then writing all the tests."*
> *- Front-end developer, 3-5 years of experience*

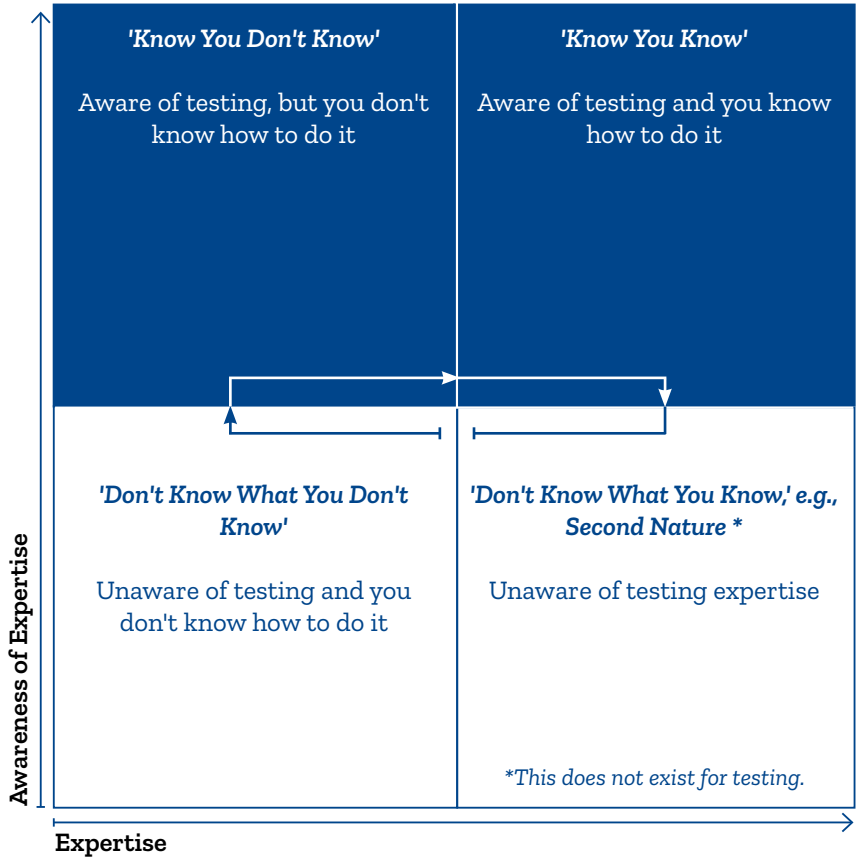
Perceived Competence
A helpful lens to view the results is through the four psychological states involved in progressing from incompetence to competence in a skill. Martin M. Broadwell, a management trainer, developed the model in 1969. The four stages are:

- **Unconsciously Incompetent** - does not understand or know how to do something, and does not necessarily recognize the deficit
- **Consciously Incompetent** - does not understand or know how to do something, but recognizes the deficit
- **Consciously Competent** - understands or knows how to do something. However, demonstrating the skill requires concentration.
- **Unconsciously Competent** - The individual has had so much practice with a skill that it has become "second nature" and can be performed easily.

This model is useful for framing how web developers talk about their experiences with testing, though replacing consciousness with awareness and competence with expertise. Developers best fit within two of the four categories:

- Consciously Incompetent, which we refer to as 'Know You Don't Know'
- Consciously Competent, which we refer to as 'Know You Know'

Critical Themes



Perceived Competence

'Know You Don't Know'
Developers who fit into this stage are aware of testing but are limited by their lack of knowledge with testing. This lack of knowledge is a barrier to successfully implementing a testing strategy.

Time is a compounding factor. Developers know there is more for them to learn when it comes to testing, what tests to run, how to set them up, what tools to use to aid in the process but to master those skills, they need the time to devote to learning. Time is hard to come by when there is pressure from their organization to move quickly.

'Know You Know'
Developers in this stage are aware of what testing is and how to do it, which doesn't mean they consider testing less frustrating than their peers who know less about testing. Instead, they face a lack of time and resources, and competing priorities that prevent them from testing as they would in their ideal world.

'Don't Know What You Don't Know'
We recruited participants from the 2020 MDN Web DNA who fell in the 'Testing Technicians' segment, meaning the audience for this study was aware of web testing. We presume there is a population of developers not aware of web testing. However, they were not the target of this study.

'Don't Know What You Know'
While an intriguing category, our findings suggest no one has the time or resources to devote to testing to the point it becomes second nature. Therefore, none of the participants fall within this category.

Insights

Testing Terminology Causes Confusion

At the beginning of this study, browser vendors had questions they wanted answers to about certain test types. In a 'Rapid-fire Round' of questions, we said the different test types and asked them to share what they knew about the test, if anything. We learned that for some developers, what constitutes a test type is unclear.

The tests stakeholders wanted to learn about were:

- Accessibility testing
- Component testing
- Cross-browser testing
- End-to-end testing a.k.a. integration testing
- Performance testing
- Unit testing
- Visual testing a.k.a. screenshot testing

"...I don't know a lot of the meaning of the terms integration testing and end-to-end testing, which is another piece I think that people like to throw around these terms, but they're very complicated..."
-Front-end developer, less than 1 year of experience

"...people think of testing as an advanced thing, partly because [people like to use] terms that [are] not easy to explain...When you're talking about testing, it's hard to define integration testing in a couple of words, so it makes it more difficult for people who don't know what it means to understand what it means."
-Front-end developer, less than 1 year of experience

"My vocabulary for testing is wishy-washy, I throw out [terms] like, 'Testing,' and 'The big testing.' I don't have enough of a vocabulary to formulate the questions and move forward on the path of learning how to test properly."
-Front-end developer, 1-2 years of experience

"I have worked only with end-to-end tests simply to emulate actions and some unit tests on [a] much lower level when you feel you need them. I haven't seen integration tests, but I think I understand there is something in the middle between end-to-end and unit tests..."
-Front-end developer, 3-5 years of experience

"I have an idea [of what component testing is], but I don't think that applies...it's me putting together different terms because we are programming components in Vue and in our JavaScript libraries, so if it's React, Vue, or Angular it doesn't really matter its components so I test these components, but I doubt that it's actually component testing."
-Front-end developer, 10+ years of experience

"I couldn't give you a clear definition of how integration testing differs from functional or unit tests. I'm repeating testing terminology I've heard in the past. [I think] integration tests are tests that test something more than just a single function, like going through a whole process, but my confidence in my answer is low."
-Front-end developer, 10+ years of experience

Beyond the terminology, one developer felt that testing and development are intertwined. It's hard to discern when you're doing what because they see testing as baked into the development process.

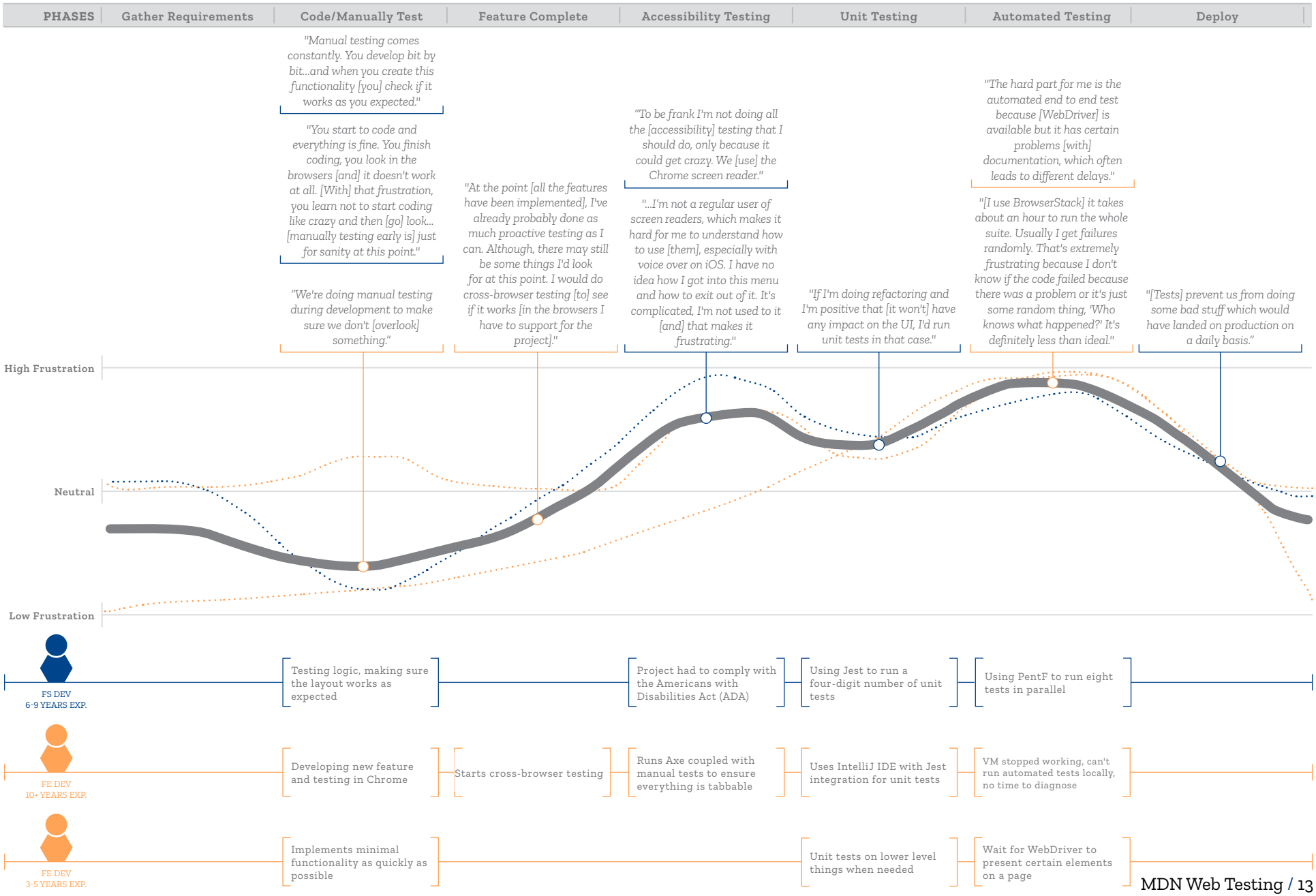
"It's hard to say [how much time I spend testing] because [testing] isn't a separate process. It's something that in an unclear way is baked into the entire process of development in general. Unclear in the sense that it's not clear to me what was testing and what was untesting? The two processes were not separated out in an understandable difference, clear in an epistemological sense like, 'It is clear to me that this is one thing, and this is another thing.' There was that lack of mental clarity when I would approach the question of testing...it's maladaptive to productivity. It does not help you do the thing you want to do."
-Full-stack developer, 1-2 years of experience

Testing Workflows

Testing Workflow

Key

- Finds accessibility testing the most frustrating
- Finds automated testing the most frustrating
- Trend Line



Testing Workflow

Key

- Finds gathering requirements frustrating, and a continuous aspect of the project
- Trend Line

PHASES	Gather Requirements	Code/Manually Test	Run into Problems	Share Screenshots	Iterate	Testing	Deploy

"Often, clients don't know exactly what they want, so that can be annoying."

"...most of [my] clients come to me and say, 'Hey, we want this, just start on this.' I'm like, 'Let's first make this thing clear.' And they're like, 'No, no, just start the project."

"Writing the code tends to be the shortest part. I've discovered all the ways that don't work. I've discovered all the quirks that I need to watch out for."

"I've been doing this long enough that I check MDN, Can I Use, I make sure that the stuff that I'm using is going to be cross-functional it's going to hit every single web browser that we support."

"I test when I'm implementing to make sure that I'm going in the right direction."

" Run into problems that seemed to happen every time, regardless [of how well we map out our specs]."

"...on screen share [my clients] have to [dictate] things to me, 'This is what we want, and that's what we want,' because most of [my] clients are not technical they can only explain when they look at your screen."

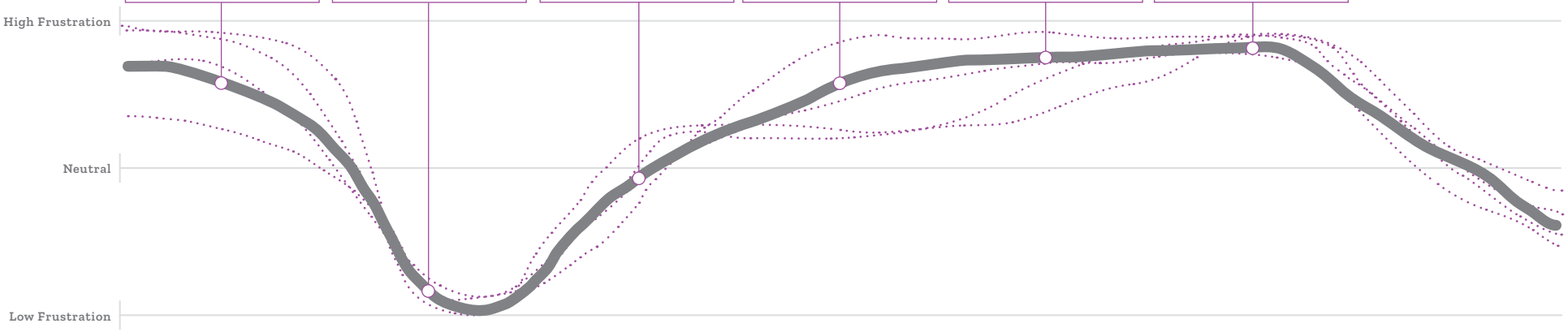
"I could take a screenshot of the Cypress code...I could show that to our QA person who can read code, but doesn't like programming and he would be able to read that and say, 'Yes, you have written what I wanted you to write."

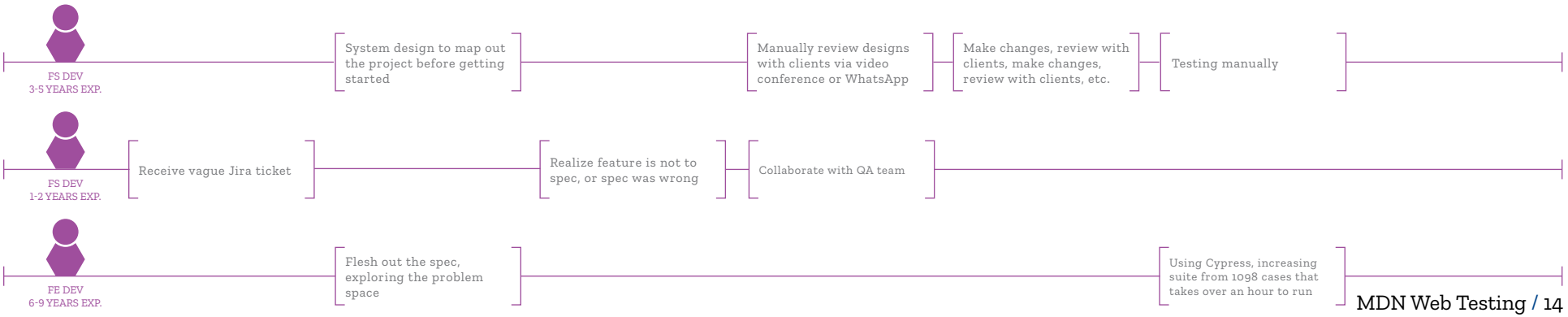
"There's a lot of points where there's loop back. At any point we could either A) realize that the feature as we're testing is not up to spec or B) realize that what we had originally defined as the specification is not what the user had genuinely wanted or meant... either of those two things can bring you back to any other point in the process."

"[I'm] testing by hand. I wish I could cut down the time on that, but all of the existing solutions end up taking more time than what it would take to do [it] by hand."

"Part of the project I'm working on is going through and writing in missing tests to capture existing functionality."

"The testing I usually follow with my clients is manual testing because there a lot of requirement changes, stuff keeps changing and changing and changing."





How the Testing Workflow Came to Be

Like so many other aspects of web testing, how developers' testing workflows came to be depends. The best way to describe how the workflow came to be is evolutionary. A subtlety of the evolutionary workflow is that what is, is not what's best, so there's continuous improvements.

"Part of our project that I'm working on right now is going through, writing in tests for missing functionality. Well, not missing functionality, missing tests to capture existing functionality...I can't go into details too much, but we had a small [greenfield] project that got spun up...because they were working so fast, and doing everything, they decided that they were not going to write tests. Now it's a moneymaker for the company, and we can't afford to break it. So, I'm going through and writing tests for those cases."
-Front-end developer, 6-9 years of experience

"A lot of [our testing workflow is] trial and error. A lot of it is from YouTube videos, articles, and resources that I see [and] read...hearing how other web developers describe their process, and lectures...assimilating, synthesizing those opinions into something that works for me. That's how I got to where my process is now."
-Front-end developer, 1-2 years of experience

"Our approach is quite organic. I think that's why I have a little bit [of] issues putting it into words."
-Front-end developer, 10+ years experience

"...a bit of a problem, like ontogeny and phylogeny, is [that] currently, we don't have a good workflow. We're in the process of sorting it out. On the one hand, there's the entire company timeline where all of our testing was completely ad hoc and now we're starting to develop something, not ad hoc and what that would look like."
-Full-stack developer, 1-2 years of experience

A few developers inherited their testing workflow. One developer didn't like what they inherited because of the language it was written in, so they took it upon themselves to create a team to change it. Another developer inherited their workflow due to dedicated testing personnel leaving an organization. Others have had to step in to help fill any gaps. In this situation, everyone is doing the best they can, but that doesn't leave much room for improvement. Another developer is trying to turn an inherited, manual testing process into an automatic one.

"I inherited [our test suite]...it was written in Java of all languages, which is terrible. Yes, I love Java [sarcasm]. With a team we rewrote it in JavaScript, and that was a big success because it was faster, [and] we got fewer failures. That was the initial push, just rewrite it. After that, it's just incremental as we add features, we add more tests."
-Front-end developer, 10+ years of experience

"It's a staffing issue, where we had someone thinking about [testing], dedicated to it. We lost that person on the team and we feel that we need that as a company. We're trying to find someone to replace [them] in [their] absence...one of the developers is the Technical Director at the company, he steers our requirements for what kind of tests we need to do. We had a dedicated QA person and testing person that helped us establish some of the tests we have. Some of our existing developers have learned what we could from that person, though they're not with the company anymore. We've just continued with what they helped us start but haven't been able to improve or add on to it."
-Front-end developer, 10+ years of experience

"[My employer has] a great big spreadsheet of everything that they [test], and they do it manually using Chrome. This is where I'm coming in. I'm trying to take that smoke test and do it using Cypress so that we can use multiple browsers and automate the whole thing. But

currently, all of our tests are done manually."
-Full-stack developer, 1-2 years of experience

One developer correlates good testing practices with delivering work without faults or issues. So, they devote time to testing so that they can keep up their reputation of delivering good work.

"I was the lead developer for most of my projects, it was my responsibility to test every bit of it to make sure that the product is delivered correctly without faults...I do make sure that I test thoroughly to make sure that the thing I deliver is perfect because I have this feeling that if a client comes to me and if I deliver something that has like even bits of faults, then it's a bad impression of myself, which is important to me. This is why I make sure that I test every bit of the project or the product before I deliver it."
-Full-stack developer, 3-5 years of experience

Types of Testing

Accessibility Testing

There are five main insights that came out of this study regarding accessibility.

First, when a developer has first hand experience needing assistive technology themselves or seeing others needing it, accessibility is more of a priority. On the flip side, if a developer doesn't have first-hand experience using assistive technologies or an influence in their life that uses assistive technologies, it can be hard for them to understand how best to develop for accessibility.

> *"I wish I had a license to JAWS or something along those lines where I could actively test the accessibility stuff myself. I am dyslexic, so I make extensive use of Firefox Reader mode, and it's very frustrating to me, like when I get to pages like the AWS documentation, where the reader mode and Firefox just doesn't work. I have to go through so much documentation all the time... I wish I had tools that I could use to better vet the accessibility aspect of things."*
> *-Front-end and Build Engineer/DevOps - 10+ years of experience*

> *"Accessibility is important to me because my father was colorblind and I have an interest in that sort of stuff. It's not important to anyone else. Once I do it the first time, usually I just leave it. If someone else breaks it, I'll probably never notice cause no one ever tests that sort of thing."*
> *-Full-stack developer, 3-5 years of experience*

> *"I try to make sure my website is accessible, but in the past I haven't put a whole lot of effort into it. I've used all of the standard tags and stuff to make sure everything could be picked up by a screen reader correctly. I haven't taken the effort to use the screen reader, myself, to see if my sites are accessible. I probably should do that though...it really hasn't been a thing that I thought about...when you're writing a site, or at least for me, I don't really think about people using screen readers and stuff because I've never used one and I've never seen anyone use one. I should try and remember it, but it's not something I actively think about. I try to follow best practices..."*
> *-Full-stack developer, 3-5 years of experience*

> *"I tried to use, I think it's called VoiceOver on the Mac, and it just did not work. I don't know what the reason for that would be, but it did not work for me. That's one thing that we probably should be testing...because I think that's supposed to be something people actually use, but somebody would have to probably come in and configure it for me and there's not enough interest...in management to do that. We haven't gotten an explicit [customer] request for supporting accessibility in our product...there's going to be the moment where somebody is going to say, 'We need it to be accessible,' and I'd rather have it at least a little bit accessible than completely inaccessible...that's why it's less than ideal, but it's just the economics, I think."*
> *-Front-end developer, 10+ years of experience*

Second, some developers take it upon themselves to do their best to follow accessibility best practices while coding or at least attempt to make things accessible with the idea that accessibility might be a future concern. This includes:

- Alt text on images and caption and summaries on tables
- Semantic elements
- Creating links as anchors
- Color contrasts

> *"There are things that I keep in mind...stuff I want to have...I always put alternative text on images and tables, if [Salesforce] allows for that...the previous project I was on didn't want [alternative text] on a [layout oriented] table...[the accessibility work I do is] not something I've ever actually told my bosses about..."*
> *-Full-stack developer, 3-5 years of experience*

Accessibility Testing

"I try to follow [accessibility] best practices [when coding]...the code should be readable to a human, you should be able to see if it's a link [and] if it's a link, it shouldn't be a div, it should be an anchor. If it's built the way it was meant to be built, then it's inherently going to be more accessible because it's more likely for there to be compatibility."
-Front-end developer, less than one year of experience

"We don't [do accessibility testing], it's something that I care about. Currently, we don't actually do any real accessibility testing and the app is by no means accessible whatsoever. It's mostly just, I'm trying to lay a bit of groundwork for accessibility in the future."
-Full-stack developer, 1-2 years of experience

"I would say [accessibility testing] has, unfortunately, not been discussed or prioritized. I do think it's something that we will end up being like, 'Oh, wait, we didn't do any of this,' and we'll have to go back and start to do it...We're a media company, so there are parts of accessibility that are a requirement that we have to do...from the perspective of the individual apps being accessible from the standards...going from a colorblind perspective from the branding and styles that are picked. There's no explicit direction and there's no explicit testing to ensure that any of that is occurring currently...there's some amount of [accessibility] that's just inherently done by us, but it's not due to any accountability that we should be imposing on ourselves."
-Front-end developer, 10+ years of experience

One developer tries to follow best practices, but feels some aspects of accessibility aren't things that can be checked programmatically very well.

"[Accessibility testing] is Incredibly important, but also something that can not be done programmatically very well. There are a lot of things you can check...make sure there is alt text on the images, make sure buttons have actual text or labels, make sure inputs are associated with a label and surrounded by a form if need be. But, color contrast, that's not something you can programmatically check...I just built a color contrast tool. One of the problems I ran into was that if the text is on something with a transparent background and behind it is an image, I have no clue what color that image is, what gradients that has...I used to use WAVE, and now the new thing I use is I think it's called WebAxe...it does as much as it can with the automated parts, checking for the alt tags, semantic structure, but it can't check it all...the fact that I can't check the entire semantic structure, you can't verify that there's no H2 nested underneath H3. Then, make sure that if there is a nav link, there's a skip nav link as well, so you can jump over that. It doesn't make sure that the alt tags are descriptive. It just checks, do they exist?"
-Front-end developer, 6-9 years of experience

Third, some developers feel there is no time to devote to accessibility or it's not an explicit requirement and therefore not something that they'll spend the time on.

"Accessibility testing would be in my realm, but sadly that's usually the thing we do afterward, if at all...I think Accessibility testing is helping people with disabilities, [and] everyone else as well. The better a page is written [for] accessibility, the easier it is to navigate with the keyboard for everyone, but it's quite a bit more design and dev time. Hence you're not doing it while you're on high pressure to get things out. Once [it's] out in the wild, you're trying to minimize your changes so that you're not breaking stuff...as a developer, you think of parts that are easy to implement while you're on the initial work, and you implement them or they often get put aside, except...in government where it's required to have them."
-Front-end developer, 10+ years of experience

"I have an extension that will tell me about anything in my code that

Accessibility Testing

I'm missing for accessibility...that is something I personally want to spend time on is adding either some automated thing that will help check for accessibility or [if] I could actually spend time, using screen readers or making sure I'm checking contrast and stuff. That's something that I'm actually just going to [try and] remember that I want to cover in my own testing and validation."
-Front-end developer, 1-2 years of experience

"Accessibility, if you mean for visually impaired or simply different devices and different screen resolutions, honestly speaking, no, I have never done it at all. I wasn't asked and do not need it for my personal needs."
-Front-end developer, 3-5 years of experience

One developer mentioned there is a growing awareness of the importance of accessibility and they are starting to incorporate it into their process.

"[Accessibility testing] is something that we were negligent on for a long time, and have come to understand in the last couple of years that it's more important. We've started using a manual checklist from a project called the accessibility project. It's like the a11Yproject.com [it] is a community project that has really great checklists of what you need to make sure is accessible on your website or web app. We've used that, we're actually in the process of boiling it down into a checklist that we can actually use when we start a project, before we launch a project, [or] when major changes happen in our project. We've also been finding that the development tools built into the browsers have been getting better at helping us test accessibility things, Firefox dev tools in particular. Now they make it easy to check tab order for keyboard accessibility, color contrast. We do test for color contrast [we have] a minimum color contrast for text and icons."
-Front-end developer, 10+ years of experience

Fourth, some developers do accessibility testing infrequently. When it comes up as something that needs to be done for a project, they feel they're continuing having to relearn. This fits within the consciously incompetent category discussed in the Critical Themes.

"[I'm frustrated with accessibility testing] because I'm not doing it often enough, and therefore if I can and want to spend the time to improve the codebase here, let's say forms, for example, and images and stuff, I'd have to go back to finding documents again, to read what needs to be done for a single A, double A, or triple A nodes or level of accessibility...it's a bit rare, so you'd have to read about it again and again, every now and then...I can't work away as easily as I do with other stuff that I do every day."
-Front-end developer, 10+ years of experience

"I think web developers could use help understanding accessibility testing, and helping people understand why it's important, how it's valuable and how it's necessary. I think that was something where we felt like we were experts for a long time, but didn't have a great understanding of [accessibility testing]. I've only started to improve that. We have found good resources from the Mozilla hacks blog and MDN. And though I don't think that [she is] at Mozilla right now, Jen Simmons was a developer who produced a bunch of materials on YouTube that were really helpful to our team. All of that stuff is great. Keep that coming."
-Front-end developer, 10+ years of experience

Lastly, there's a perception that accessibility doesn't rest on the shoulders of developers alone. It's also the responsibility of a brand or design team to consider accessibility, especially when a developer doesn't have the charter to make changes.

"Accessibility...that's more on the design side. I'm not a graphic de-

signer...I'm not able to correct anything on the design side. That's a designer thing...they do the design, that's a finished thing [and] I need to respect that. I need to do that site according to that design. Of course, I can propose things and I can talk to the designer and say, 'Well, can I change that or not?' It will be according to the designers [decision to make a change or not]...color contrast...that's a difficult thing because you need to go according to the branding and the thing you are doing, if it fits that specific product you're managing and the branding of the client [that] of course goes with the accessibility and usability. We need to check that."
-Full-stack developer, 3-5 years of experience

Component Testing

Developers shed some light on their approach to creating components and some of the simple tests they have for each component.

> *"Most of the components I write by hand. I try to avoid components written by other people, so I don't test them really, because I don't use them. Almost everything I use, I write myself in JavaScript at least."*
> *-Full-stack developer, 3-5 years of experience*

> *"I try to test each simple component. That's the keyword there, like here's the login screen, the login button, whatever the component that tells you if you're signed in or not, here it is. Here are its properties. Does it work typically? That's, as far as I've been able to comfortably get on component testing, because then you get into mocking rest requests and faking, you're trying to capture all this stuff. That's another absolutely infuriating thing is the total and complete made up syntax of things like Mocha and Chai where it's just 'test.that.this.is.=()5' And you're like, what does that even mean? That's not a thing. That's not how any of this works, but it's a scripting language, so you can do whatever you want."*
> *-Full-stack developer, 1-2 years of experience*

One developer had experience with snapshot testing, but learned there wasn't much value in it over time. Their code changed to frequently making snapshots outdated, so maintaining the tests was difficult.

> *"I forget what the software package is called, but we were doing what are called snapshot tests. This is mostly in [a] React based web development. We'd build a React component, you see what the output it creates, you would store that. Then, the test would make sure it matches the output that you expected. But, and we were new to testing, we went overboard with that and we built a zillion snapshot tests. We thought we were awesome. And then we realized that anytime we change anything, we had to change every single test. We got to be doing that so quickly and automatically that we were barely paying attention to updating the test. The tests were valuable on paper, but in practice, they weren't actually helping us or at least the trade-off of time investment. It wasn't helping us catch real bugs or real problems."*
> *-Front-end developer, 10+ years of experience*

One developer had heard of snapshot testing, but didn't see value in it and therefore doesn't do it.

> *"Snapshot testing. Seems to be all the rage with some web developers, especially people using React, and it has never made sense to me.... Maybe I've never seen it done correctly before. Maybe it's that I've never set up snapshots myself from scratch before, just because my experience working with them in other people's codebases has been absolutely awful. Every time you make a change in one of them [it] fails, you get this enormous unreadable diff between what was expected and what you see now. It may be machine-readable, but it's not readable to the human eye. You basically end up just thinking, 'Well, I look at it in the browser and it looks okay to me,' so guess let's change the tests that the current correct snapshot is this new one. Little, do you know, that you've broken something and you won't find that out until later. I find that kind of test is completely useless. I've never intentionally worked them into editing. Again, I hasten to add again that maybe I've just never seen it set up correctly before."*
> *-Full-stack developer, 10+ years of experience*

Cross-Browser Testing

A common theme in how developers approach cross-browser testing is that they will develop in their preferred or primary browser, and then, later in the process, test in other browsers. This can create frustration because what they built may not be compatible with other browsers. On the flip-side, experienced developers tend to have a better grasp of what works or doesn't work across browsers, so they will defensively code against those incompatibilities, making early testing across browsers less of a priority. While a developer's preferred browser is a personal choice, many cited Firefox or Chrome.

> *"Sometimes you build something wonderful, you're only testing in your primary browser. And then you spin up the secondary browser and you realize that what you built was not cross-functional."*
> *-Front-end developer, 6-9 years of experience*

> *"I don't tend to worry that much about browser compat until near the end of the process. I develop against Firefox because that has been my favorite since Firefox 1 back in the day, and the dev tools are becoming wonderful. Many thanks to the Mozilla developers who have been working on those recently. I test in Chrome as well, because my client is mainly using Chrome. Other than that though, I fool around on my phone because I need things to work on mobile too. I know that Firefox in responsive mode...isn't necessarily exactly the same as what I see in Firefox on my phone or Chrome on my phone. So I tested those two. Those are the things [I] actually test with in-person as a matter of course, during development, but I don't touch the other browsers until really late in the procedure, just because for the most part, these nice new tools and I'll toot my own horn a little bit because of my experience, there are certain things I know won't work [on other browsers] so I'll either avoid them or I'll defensively code around them from the get-go. I don't tend to even check until really late in the process."*
> *-Full-stack developer, 10+ years of experience*

Some developers rely on browsers' dev tools as a resource to check visuals across browsers. A common pain point with using dev tools is that they don't render things accurately as one might experience on an actual device. One way some developers have overcome this problem is to test on the actual device or have others to help them test across devices.

> *"We're mainly using the developer tools in the browser. If I was using Chrome, I'd be using the developer tools they have. If it was an old version of Firefox, I'd obviously be using that. I was testing an old version of Firefox and I saw that on one of the features I was using, an error popped up [and] said it wasn't supported. I wouldn't have known that unless I tested it. First, I'd actually look in the documentation and make sure I was correct about the error message. [If it was right], then I just had to refactor my code, test and make sure it wasn't using that feature and then test it again. Really, the only tools that [we use] throughout that process are the web developer tools and VirtualBox [which is used to create a virtual machine] with the target operating system."*
> *-Front-end developer, 6-9 years of experience*

> *"When I'm doing cross-browser [testing] I use Edge and Firefox. I'll open up the website, check it out with the dev tools and resize it [to] see if things look visually correct. Also, I'll open up the different dev tools to check mobile responsitivity on the dev tools. Nowadays, since Edge and Firefox are on Android phones, when I get to mobile testing, besides just Chrome, I'll also check Edge and Firefox on my phone."*
> *-Front-end developer, 1-2 years of experience*

> *"I use Chrome dev tools to test my CSS in the responsive mode. Everything would look okay to me on my end, but then [my client] would send screenshots [saying], 'This is not done, this task is not completed, and this and that,' I had no idea what's going on because I would create new profiles for devices and none of them seemed to fix the issue. I*

think testing the CSS [across] browsers and cross devices is the worst frustration for me."
-Full-stack developer, 3-5 years of experience

"[Testing on hardware] is really helpful. The dev tools, since it's a simulated environment, is not a one-to-one for how it looks on an actual device at that actual resolution. Making sure that I test on my physical phone helps me visually make sure that things look right [and] that things are spaced out correctly....also the resolution in pixels is a bit different on a physical phone versus my different laptops and dev tools. Being able to actually see [the application] on a physical device is really helpful and will get me to tweak certain things to make sure that it looks good [across devices]."
-Front-end developer, 1-2 years of experience

"Most of [mobile testing] I can do in the dev tools on the desktop by resizing screen sizes...the sizes might not be 100% accurate...[the window on the devices] might be smaller [and] touch in Firefox doesn't quite work...and in Chrome they mimic it. In the end, you have to retest it on the device. What's pretty cool on Firefox is the debugging. I can launch my Android phone with Firefox on it with a USB [cable]. Then, I have my desktop dev tools with the open website on the phone and can debug there much better on the phone...that's one of the best tools if you have an issue that you can't solve visually on desktop....it's something I rarely use sadly for how cool it is. Luckily, I guess I don't have to."
-Front-end developer, 10+ years of experience

"I say browser testing, but that I would expand it to device testing, making sure whatever we've built works as intended on all the different clients that it might run on. That might be a slew of web browsers across different operating systems or mobile devices. Sometimes there's some tools we use to help us with that though. It tends to be not automated for us where we open it up on the device. If we don't have the device we'll use BrowserStack as a tool to get virtual access to devices or browsers we don't have, but we're opening the thing up and trying it with those with different browsers, making sure it works."
-Front-end developer, 10+ years of experience

"We have our own lab...[with] real devices and we test on those...testing involves going through a checklist and checking the list on each one of those devices, mainly for functionality...you start check on those functionality items, and then you start checking the how the page looks, if there's any wrong element, or something not fitting in the layout...this testing is local. Then...I needed to test up on the real server and control the time on which this page, this site, for example, will be open for the client to provide their feedback. That's it mainly"
-Full-stack developer, 3-5 years of experience

"We had a device library, so we would just literally go and pick up devices that had certain things installed on [them]. We had a windows computer, we had all kinds of phones...there was a whole testing group at [my] last [job] they had different networks you jumped on, and they would have different latency speeds, different settings...it was pretty slick, but it was a lot of going and checking out the device and bringing it back to your desk and hooking it up to computer. It's a lot of work still."
-Front-end developer, 6-9 years of experience

One developer offered input for what they would like to see in a tool to help overcome this issue.

"I was ready to ship something and I asked my coworker to check on their [smaller resolution] laptop than mine...she noticed that this carousel I made was broken for that resolution, but it worked on mobile and larger desktops, and [in my] dev tools...getting other people to validate [on their devices is] helpful...that's exactly where automated

Cross-Browser Testing

> *testing tools would be helpful...if I had some automated tool, [what] I would want is either I push to my Git repo or push to Amazon and it's at an actual URL and this automated tool will...check my website at different physical device screen resolutions...make some screenshots ... what would be the most helpful for me to see out of the output of this testing...I want to see if things look broken, if things are slow...I would also want this tool to be able to mimic physical inputs like scrolling or swiping on devices and being able to tell me if things work well or if things are slow."*
> *-Front-end developer, 1-2 years of experience*

One developer referenced Normalize.css as a resource for improving their experience conducting cross-browser tests.

> *"...Normalize.css...I'll install something that will try to make the differences in HTML elements between browsers...consistent. That actually helps a significant amount because a lot of slight CSS things, like margins are maybe a little bit different between browsers...since that's normalized it takes out a lot of headaches between checking different browsers. That's why that's not as frustrating as it might be for some people."*
> *-Front-end developer, 1-2 years of experience*

End-to-end Testing a.k.a. Integration Testing

Nearly half of the developers in this study either did end-to-end testing or expressed a desire to start doing it. A main reason developers adopt end-to-end testing as a part of their workflow is that it can provide confidence in their code. Developers see this type of testing as difficult, but worth the time.

> *"[When I can implement end to end testing it will give me] a little more confidence in making sweeping changes in one feature without then having to go and make sure that everything is still okay."*
> *-Full-stack developer, 10+ years of experience*

> *"...as you go up into integration tests and end-to-end tests, the tests get harder to write but are much more useful...if you want to write good tests, and you want to make sure that your product is safe and secure, you're going to spend a lot of time writing those tests and you're going to have to make them resilient, make sure that when they fail, they fail because the actual product fail, not that they're flaky."*
> *-Front-end developer, 6-9 years of experience*

Stakeholders had specific questions on whether or not selecting elements with end-to-end tests is a major challenge. Of developers who had experience with this, they were split in terms of the challenges they faced. One said it was a challenge, and another said their experience was positive, generally.

> *"[Selecting elements is a major challenge] and making sure that elements exist because the webpage has been so dynamic. You select an element, you click the button, say it is, and then you wait for a modal to appear. With Cypress, so much of the time you can make it wait an arbitrary amount of time or you can make it wait until this thing shows up, but no longer than two seconds."*
> *-Front-end developer, 6-9 years of experience*

> *"[Selecting elements during end-to-end testing] is positive, generally. Selecting elements is one of those parts that is easy and reliable. Generally, most of our elements have classes. Those are easy to select...the more important elements usually have IDs, so those are easy to select. WebDriver's method of selection is very straightforward and recognizable. It's basically jQuery. Very rarely am I trying to select something that I can't or I don't often run into an issue where I'm like, 'Oh, why isn't this selection working?' Only if someone has changed a class name and I didn't know about it. That's when the real danger is there. [This happens] occasionally, I find selectors that are more reliable, but that's more just on how we've set things up, more of a design conflict than software conflict."*
> *-Front-end developer, 3-5 years of experience*

Performance Testing

Developer perceptions of performance testing ranged from I don't do it because no one has asked for it or it's not necessary for our site, to manual performance testing using the network tab on developer tools to frustrations with it being unpredictable and susceptible to outside influences.

> *"I wasn't asked to do [performance testing]. I have not worked on such projects with that high [of a] load, so it was not necessary."*
> *-Front-end developer, 3-5 years of experience*

> *"I try to make sure performance is quite good. I don't do any automated testing. I'll open up the network tab on the developer tools and make sure everything loads in a reasonable amount of time. I don't have any fixed amount for that, but if it feels like it's taking too long I'll try and cut down. But I also don't use any big, heavy frameworks. I try to do everything in Vanilla JS. [Performance] is not an issue too often...I've only really tested [performance] in Firefox because a main issue is downloading everything from the server. In the past, I've had issues with fonts being too big and that's been slowing down the site. I've never had an issue with my code taking too long to run in the browser really, so I only test on one browser because it's not worth testing in others. Generally, the performance is pretty similar between browsers for the stuff I'm doing at least anyway. Actually, one exception to that would be the rendering engine I wrote in the browser that ran pretty slow on mobile devices, but it was made as a proof of concept so I wasn't too concerned about that."*
> *-Full-stack developer, 3-5 years of experience*

> *"I've done [performance testing]. It's hard and it's very mercurial. It really depends on your device. Usually, I'm on a top-end laptop. I wrote some tests and I think I used Puppeteer to write the tests and [I'm] running the same suite of tests on my device multiple times [and it] gives you multiple answers. You couldn't really say, 'Okay, well, the performance is obviously degrading because [of x,y, or z], it could be hundreds of milliseconds off because my other browser decided to check for email. It does definitely depend on the browser, but it depends on so many things. There could be a windstorm when you're trying to use WiFi and [that's] going to affect how your device connects to the servers."*
> *-Front-end developer, 6-9 years of experience*

For those developers who do performance testing, they mentioned using GT Metrics, Lighthouse, or Google speed insights.

> *"I'm a back-end person too...you need to check how fast a server is responding to a query from the customer...I do perform performance testing and I test the speed of the website using tools like Google-based speed insights or GT metrics. I mostly use GT metrics, but I also take help from web speed insights."*
> *-Full-stack developer, 3-5 years of experience*

> *"We started using Lighthouse for [performance testing]...a tool that was on many other testing sites apart from the browser tool. I wasn't using that, but I discovered the tool and I told my superiors, 'Hey, there's this tool. Can we use that tool? And they told me, 'Yeah, that was what we were looking for, for performance testing.' We were using a lot of other things. I'm not sure what they were using, but those tools were also providing the performance scores on many items SEO, loading, and that kind of thing [so] we started using Lighthouse for that. I'm pretty happy about that because Lighthouse right away...it says what you need to correct and to check to enhance performance on your site or the things you're doing."*
> *-Full-stack developer, 3-5 years of experience*

For those developers who do performance testing, they mentioned using GT Metrics, Lighthouse, or Google speed insights.

Performance Testing

"I'm a back-end person too...you need to check how fast a server is responding to a query from the customer...I do perform performance testing and I test the speed of the website using tools like Google-based speed insights or GT metrics. I mostly use GT metrics, but I also take help from web speed insights."
-Full-stack developer, 3-5 years of experience

"We started using Lighthouse for [performance testing]...a tool that was on many other testing sites apart from the browser tool. I wasn't using that, but I discovered the tool and I told my superiors, 'Hey, there's this tool. Can we use that tool? And they told me, yeah, that was what we were looking for, for performance testing. We were using a lot of other things. I'm not sure what they were using, but those tools were also providing the performance scores on many items SEO, loading, and that kind of thing [so] we started using Lighthouse for that. I'm pretty happy about that because Lighthouse right away...it says what you need to correct and to check to enhance performance on your site or the things you're doing."
-Full-stack developer, 3-5 years of experience

One developer expressed a desire for their team to do even more performance testing than what they are currently doing. That said, they work in an organization that has their web team siloed. Some of their desire to see more things does not necessarily mean something isn't being done, rather they may not be aware of it.

"We do run Lighthouse occasionally, and we have numbers that we report on, but we haven't been reporting on [page load times] lately. I would like to see us go even beyond that and have more metrics, alarming, and things of that nature driven towards our latencies when communicating with our APIs and not just focus on page speed. Our team is the render portion of our application and less the backend services that drive all the data. I don't know what level of performance testing is done or available from the backend side of things to test our different APIs and whatnot. I would like to see us at least drive metrics on those so that we can know. Outside of that, prior to releasing to production, I would like to see, even if it's as limited as Google Lighthouse or Chrome Lighthouse, I'm not sure how it's branded, but running that against the application to give us insights...again, there's no feedback, pipeline or loop regarding performance testing. It's more an as we feel needed basis currently."
-Front-end Developer, 10+ years of experience

One developer expressed a desire for performance tools to help them understand their bundle size between releases and alert them to when there is a significant change. They feel tooling in this area is lacking.

"There was [a tool] called GT metrics that would do regular performance testing on a site. We haven't used that in quite a while. These days we rely on some of the development tools built into the browsers like the Firefox Dev Tools, the network panel telling us how long things are taking, or the profiling panel. [We] also use some of the Google tools. Whatever they call it these days, Google Page Speed or Google Lighthouse. I think I'm using different terms that all describe the same toolset. It's hard to keep up with the terminology. I guess performance testing, [is] definitely something we keep in mind [but it] tends to be something we do manually during testing, not something we've automated. [We haven't automated performance testing because] I'm not sure if we've found tools that make it easy for us to do, but I know that we have talked about it in the past. For example, bundle size, having something tell us when we make a pull request on GitHub that before this pull request, people were going to download 278 kilobytes of JavaScript and now they're going to download twice that or something like that to help us keep a closer eye on it, but we don't have anything like that in place right now. We'd like to, I'd like to."
-Front-end developer, 10+ years of experience

Unit Testing

Stakeholders didn't have specific questions regarding unit tests. They see these tests as a well-developed space without a lot of pain points. What we learned with this study, like many other aspects of web testing is that developer's perceptions of unit tests vary. Some developers see unit tests as simpler and less mystical than end-to-end tests. Some developers talked about end-to-end tests as rewarding, whereas others talk about unit tests as rewarding. Some see a lot of value in unit tests and others don't.

Pros of Unit Tests:

"[Unit tests are] extremely rewarding. I worked at a company that did multi-million dollar enterprise software with exactly zero tests. When you screw up in production it goes to the most senior guy typically to fix it. For a number of years, I was getting called at all hours dealing with all kinds of stupid things. Then it was like, 'Oh, we should maybe have tests, maybe we should use a language that has tests'. Since learning the hard way, there is a joy in saving your code, having the editor run some unit tests, gives you the thumbs up and knowing that a change you made today isn't going to completely screw something up that you forgot about from three years ago."
-Full-stack developer, 1-2 years of experience

"I think that [unit tests] still have a very important purpose. End-to-end catches those macro bugs, but we don't have a strict review or way to catch software bugs like unit tests would. So it's preferable in that way, or at least desired in that way."
-Front-end developer, 3-5 years of experience

"...where I find unit tests to be a lot more prevalent and a lot more useful in terms of confidence is typically on that API layer or that service layer. I find them to be a lot more important at that point...I think there's a lot more benefit in that backed world for having unit test coverage and being confident in those services and APIs that you're delivering over the front end side of your application..."
-Front-end developer, 10+ years of experience

Cons of Unit Tests:

We heard stories of how unit tests are hard in an Angular or React environment, where there are issues with DOM APIs, and how they aren't helpful as they tend to age as utilities come and go.

"The way our React is written, it's not the most testable. It's not the most unit test friendly. I think there have been attempts in the past to set up unit testing, but to varying degrees of success. We didn't know what we should have done...and we don't really have time to rewrite or investigate how to make the testing work while we're billing clients for time that we need to be developing. It's unfortunate since [we're] testing an already workable product, generally, it's lower on the totem pole with the time that we can spend on other things."
-Front-end developer, 3-5 years of experience

"We're doing our unit testing, which unfortunately we do not write enough unit tests because...I feel that in the Angular ecosystem, we have no idea how to do it in a lot of ways. They've got this whole reactive functional programming paradigm and they've got this Marbles Test tool, but it's very outside of how you would normally write or, I feel it is..."
-Front-end and Build Engineer/DevOps, 10+ years of experience

"[I unit test] usually just for utility functions, they're fairly small, self-contained pure functions...if you put in the same value every time you get the same value out every time they're not stateful...one [pain point] is so many utilities have [an] inherent state, anything that uses the DOM API you end up having to mock quite a bit like, 'Oh, my little function only does one thing. I asked it to do this and it gives me this

back every time.' Unbeknown to you, in the middle of that function, it's calling to the DOM saying, 'Give me a list of class names on this object.'"
-Front-end developer, 6-9 years of experience

"The quickest and easiest test to write are the unit tests, but they are also probably the least helpful...units and utilities come and go. You write them to use them, but it's not the core of your program."
-Front-end developer, 6-9 years of experience

Visual Testing a.k.a. Screenshot Testing

A lot of the developers in this study have an informal approach to visual testing. This could be comparing the site to a Figma file and eyeballing for differences, checking different breakpoints, and using dev tools.

> *"[I've done] a little bit [with visual testing]... I'm a front-end developer, so I do a lot of eyeballing stuff and I'm always given Figma files or PDFs or whatever. I bring it up on the screen next to the actual website and make sure it looks the same."*
> *-Front-end developer, 6-9 years of experience*

> *"A lot of [how I visual test] is using dev tools and checking different breakpoints because I have my laptop screen resolution, but also I have a really large monitor... I'll see what things look like at a large desktop and scaled down to about 320 pixels wide...like an iPhone 5 width and see if that looks good on dev tools."*
> *-Front-end developer, 1-2 years of experience*

> *"I have breakpoints in my base CSS and I [have] responsive rules set up that I copy over for all my projects so that changes infrequently. I'll add extra things if something looks wrong [in] a tablet view versus a really small mobile versus like an extra-large desktop...for the products I work on mobile is king. I optimize first for mobile and then make sure it's okay on desktop."*
> *-Front-end developer, 1-2 years of experience*

Some of the participants had experience with screenshot testing tools but opt out of using them because they don't see the value in them. Some even see these tools as a source of frustration because they haven't seen them work well or the results are hard to parse in terms of what's actually an issue to pay attention to.

> *"...testing for CSS issues with some automated tool is really hard. Most of the time, it just doesn't work. Theoretically, you could use visual regression or visual snapshot tests but in my experience, this is even more frustrating than testing other things. What we typically do is test the logic. Does it work as expected, if the layout works as expected, but we have no regression tests for these."*
> *-Full-stack developer, 6-9 years of experience*

> *"I don't know what the name of this tool was called, but there's a tool where you can overlay two pictures on top of each other to see where the differences are...it's usually more annoying...the frontend changes a lot. You change some content, so you make a button [and now] the text and the button are slightly wider. The column gets pushed out a little bit. It's all supposed to flow like that, it's supposed to be responsive, but this tool will start highlighting everything...you're not quite sure what's signal and what's noise."*
> *-Front-end developer, 6-9 years of experience*

One of the developers who didn't see the value in screenshot testing tools suggested a few ideas for improvements. He wanted a tool to check on a variety of screen sizes and detect motion too, not just static pictures.

> *"I would want it to check on a variety of screen sizes, and I want it to be more than just static pictures. So many of the bugs that I've got in there are like when I click this, it doesn't fade in like it should, it snaps in too fast...it's the movement between transition states."*
> *-Front-end developer, 6-9 years of experience*

One developer wasn't familiar with the term screenshot testing, but intuited what it meant and felt like it was a bad idea.

> *"I don't know what [screenshot testing] means, but it sounds like a bad idea. It sounds like it would either be you take a screenshot of a page and you compare it to another screenshot of a page. [Maybe] you*

have some AI that combs over it and tries to compare the two of them, which sounds like a bad idea. I don't trust robots. Or you just literally compare them to see if they are the same image, it's like you move the button two pixels. But of course, I don't actually know what the word screenshot means. It just sounds like it would be something like that. Isn't there a means of testing where you take a person and they do some stuff and you record all the stuff that they do? All the mouse movements, the typing, and things. And then, you move the zoom with the button two pixels and it breaks. It doesn't have the name screenshot testing, but I feel like it has the name something similar to that."
-Full-stack developer, 1-2 years of experience

One developer who doesn't do visual testing right now, would like to do it in the future. What's stopping them from implementing it is a lack of time and a lack of knowledge about how to get started.

"We don't [do visual testing], I'm trying to sell people on it right now as part of our quality issues because we don't have the capacity to do a proper regression testing of our system anymore...I might get the language a little bit messed up here, but the visual regression testing piece is something that feels like good low-hanging fruit for us. We already have Selenium. There's a little bit of questioning in my mind about tooling for it. That's always been the question. I know that there are a few tools that can do it. I feel like maybe even Puppeteer could do it back in the day or had a plugin...I don't exactly know how to set it up off the top of my head. I'm the one that's writing parts of our build pipelines. If I can't get the time to research it, it just doesn't get set up. Although I think that it's a really good opportunity and it goes back to I wish I saw how other people were doing it in their Jenkins pipelines and could like leverage it for our .NET stuff or for our component stuff, our design system that we're building. I'm trying to get us moved over to Jest and I forget the name of it, but Jest also has a visual regression piece to it that you can leverage.[1] My hope is that for our front end SPAs, [we can] get our shared components into a library with Storybook...every time that you update stuff that goes through a CI/CD for the frontend pieces and leverages just visual regression tools."
-Front-end and Build Engineer/DevOps - 10+ years of experience

1) The participant was likely referring to https://jestjs.io/docs/snapshot-testing.

Testing Specifics

Time Spent Testing

The amount of time spent testing differs for developers on a number of factors ranging from internal (company structure, tools, etc) and external (clients, production timeline, etc) factors. Several developers also noted that the type of test can impact the amount of time spent testing and what functionality is being tested. The amount of time developers spend on testing varied from 20-50% of their production time. An outlier in the study involved a full-stack developer reporting that they spend 100% of their time building up their smoke tests with Cypress and Gherkin. Part of the differences in the time spent testing is because of the different testing workflows. Developers who tend to do manual testing during development spend more time testing on a daily basis than those who wait until a project is feature complete to test.

> *"I would say [testing is] about 50% [of my] time. Let's say you invest ten hours into building features and about ten hours into testing this feature. Unless it's some text change or something, but if it's a large feature, which would include some styling logic and stuff like that, it's as much testing time [as] implementation time."*
> -Full-stack developer, 6-9 years of experience

> *"[Over the past month] or so I've been spending all of my time testing because I'm building up our smoke tests, getting all of the Cypress together, the Gherkin, and such. So 100% of [my] time [I spend testing]. Before then, I honestly probably spent like 10% of the time testing, and doing a really poor job of it, being really terrible at it, pushing code with bugs and having people not get angry with me, but maybe they should have."*
> -Full-stack developer, 1-2 years of experience

> *"[I test] every day...I usually type two or three lines of code and go to a browser and make sure that I'm not going off on a tangent, except like, if I'm doing refactoring and I'm positive that the refactoring doesn't have any impact on the UI...I'd run unit tests in that case. I don't think there's a day that I don't test...it's either manual testing, unit testing, or automated testing."*
> -Front-end developer, 10+ years of experience

Factors that changed the amount of time spent testing included technologies like Salesforce, where the program has its own built-in programming language. Creating tests that support IE11 was a pain point for some developers, as the tests that work for other browsers do not work in IE11 and can lead to bugs and problems. Testing is perceived to be quicker for projects with a simple API.

> *"Maybe a quarter of [my] time [I spend testing], which is honestly pretty pathetic. The reason there's so little testing is because on Salesforce, testing is not required for JavaScript. Salesforce has a built-in programming language called APEX. It's very similar to Java, it's backend so it's on the server. That actually does have required unit testing. So for example, you can't move your server from the testing servers to the production servers without getting at least 75% code coverage on every single APEX class, which is like Java. There is no such restriction on JavaScript."*
> *-Full-stack developer, 3-5 years of experience*

> *"[My satisfaction with testing] depends on the job and the tools available. If ... we are doing JavaScript and we need to support IE, I'm not satisfied with the time required for doing this process...[when] Firefox is the main thing then you don't need to worry because everything works there and everything looks beautiful. Sometimes, Google Chrome will struggle with [testing] some things...IE and Edge are the main browsers that take time because those tests don't work...there's a lot of frustration because, 'Why does this test not work if it's supposed to work?'"*
> *-Full-stack developer, 3-5 years of experience.*

Time Spent Testing

"[The time I spend testing] depends on the size of the project and depends on how annoying it is. If it's a simple REST API where...it's simple credentialing stuff, [the number of tests is] pretty small because there's not much going on. We had a project where it was a whole bunch of microservices, they all talked to each other, they all did a bunch of things. [We] needed to spin up, 'Here's two fake customers. Here is 50 fake pieces of equipment that the system tracks here is the catalog that the customer's interested [in]... building [all] this junk out, and it got big, [we] end up with easily one-to-one test to code size..."
-Full-stack developer, 1-2 years of experience

When time spent testing relates to the specific type of test being performed, developers had different emotions on the types of testing. Some note wanting to spend less time with cross-browser testing around the specific pain points of IE and non-evergreen browsers. Others feel they don't spend enough time on unit tests or end-to-end tests because just getting the suite of tests setup is time consuming.

"No [I am not happy with the time and effort I spend on testing]. I have to accept that we still need to support IE, but then if you have all the frustration that we get with having to run the automated tests that's where my cup overflows like, 'Ugh.' Ideally it would be definitely less time. It should take less time to do all this."
-Front-end developer, 10+ years of experience

"Broadly, we don't do unit tests as much as we should. I don't think we run unit tests because we don't have a suite setup. It's installed we could run them, but we don't just because we haven't developed them. We don't have a testing layer that interfaces directly with the API either. I don't think that's a choice we made that we wanted to make. It's just we haven't had time to develop that. I feel like there might be other things that I don't know about that I'm missing, but in my mind end-to-end and unit testing are [ones we don't spend a lot of time on]."
-Front-end developer, 3-5 years of experience

Some developers noted their firms are slow to discover the value of testing, and that causes more challenges in adopting good testing practices. Developers also noted that when firms bring new developers onboard, the hiring cost is related to the capabilities of the developer, and you would expect a capable developer to know how to test and be able to perform testing in a timely manner.

"We don't have the muscle memory to include [testing] in all of our estimates when we're assuming how long something's going to take or how much it's going to cost. We get better at that over time. To succinctly answer your question, no we feel like we need to do more [testing]."
-Front-end developer, 10+ years of experience.

"Hiring developers, the cost is very closely related to their capabilities. If they are very capable developers, you could expect them to know how to do testing, why to do testing and to do it in a decent amount of time. Those are expensive and the clients don't really want to pay money...it becomes more of an issue of 'Okay, who can do it? How well can you do it? And how much does it cost to do it?'"
-Full-stack developer, 3-5 years of experience

Some developers noted their firms are slow to discover the value of testing, and that causes more challenges in adopting good testing practices. Developers also noted that when firms bring new developers onboard, the hiring cost is related to the capabilities of the developer, and you would expect a capable developer to know how to test and be able to perform testing in a timely manner.

Time Spent Testing

"We don't have the muscle memory to include [testing] in all of our estimates when we're assuming how long something's going to take or how much it's going to cost. We get better at that over time. To succinctly answer your question, no we feel like we need to do more [testing]."
-Front-end developer, 10+ years of experience.

"Hiring developers, the cost is very closely related to their capabilities. If they are very capable developers, you could expect them to know how to do testing, why to do testing and to do it in a decent amount of time. Those are expensive and the clients don't really want to pay money...it becomes more of an issue of 'Okay, who can do it? How well can you do it? And how much does it cost to do it?"
-Full-stack developer, 3-5 years of experience

Developers note that tight production schedules cause testing to become less of a priority in order to deliver on time. This causes frustration when a client is not clear on their requirements or changes the requirements partway through the project, because developers need to write new tests and update old ones. Some developers believe that clients are not interested in testing because they do not understand the benefits of testing or what types of testing can benefit the project.

"We have a pretty tight production schedule. Part of what takes up most of my time is fixing bugs, developing features, and managing deployment... I don't think that there's a reasonable way that any of those three things could actually be reduced in size." -Front-end developer, 3-5 years of experience.

"From an engineering leadership perspective, [testing has] always been a priority, but from the realistic expectation of delivering the product, it then becomes less of a priority in terms of ensuring that we're adopting these testing practices. The longer we go without it, the harder it is for people to jump on board."
-Front-end developer, 10+ years of experience.

"...most of the time, I try to convince [clients] that putting effort into testing is worth the time. But sometimes they are like, 'Yeah, Whatever. I don't care ...this might sound a little rude, but they have to feel the pain somehow. If they have had some serious issues on production, no one was there to fix them, it was the middle of the night, and some clients complained about it... typically after that, they tend to agree that testing is important and can help..."
-Full-stack developer, 6-9 years of experience

Developers may have a bare minimum of testing that they will complete on projects in order to feel more comfortable. Developers would like to cut down on the amount of time spent manually testing, but feel the existing options to do this would end up taking more time. Testing should be able to catch real-world bugs in the development environment, but this is currently not the normal case.

"[I'm happy with the amount of time I spend testing]....every developer [has] their bare minimum [of testing] that they feel they [need to] cover to feel comfortable and things are shippable."
-Front-end developer, 1-2 years of experience.

"[spending time on testing is worth it if it] is catching real-world issues...we had a security issue that didn't show up in our development environment, but did happen in production. So testing that would catch real-world bugs...if it's catching things, notifying us of them, and we didn't know about them, otherwise that would be a real value."
-Front-end developer, 10+ years of experience

Time Spent Testing

Test Maintenance

The time developers spend maintaining tests also varied. Maintaining tests is important because it boosts developers' confidence in those tests. If they aren't maintained and tests fail, there is less value in testing. For some developers, maintaining tests is correlated with significant changes to a particular feature.

> *"[Our tests find something broken] maybe once a week, which is good. Generally people aren't trying to push things that break stuff [laughter], but once a week someone will either make a change in the database, change the API, or change a component one of those three places and then somewhere else something pops up. We're not perfect. There are lots of failing tests that are failing because of unmaintained tests, which is not good. If you don't have confidence in those tests, then they're not really useful. Maintaining them is actually a higher priority thing than it seems."*
> *-Front-end developer, 3-5 years of experience*

> *"I think [time spent testing is] worth it because for the most part, if I'm not making changes to a particular file, all the tests for that file are still fine and they're not going to change. Nothing's going to suddenly fail from there. Usually, the only time that I am maintaining tests is when I'm making a significant change to that particular feature anyway, or perhaps in integration tests where something linked has changed."*
> *-Full-stack developer, 10+ years of experience*

Test Suite Size

When asked about the size of the test suite, developers indicated that the size of the suite varies throughout the product lifecycle, between each project, and even by the employer. Some developers could estimate the size of their test suite, some even with surprising accuracy, while other developers struggled to guess at the size of their test suite. The size of the test suite grows naturally as the project moves further along in its lifecycle, and occasionally with updates. As the test suite grows in size, earlier tests may become redundant to have, yet, they are not easy to get rid of. While keeping tests up to date is important, this is not always done after the product goes live.

The test suite typically starts as manual, and some developers quickly try to start automating the processes. Respondents indicated that the project can drive what specific testing needs to be done (i.e. an e-commerce site needs to have a working cart and checkout process, so tests are geared to that need). This can lead to smaller functions, like a zoom in/out feature, not being tested before going live.

> *"[The number of test cases we run] really depends. There are so many nuances. You definitely want to test the happy paths. You want to make sure that if it's an e-commerce site that 'add to cart' works, checkout works. We want to make sure that searching for a product works. You know, all these things that are high traffic, meaningful user flows. You want to make sure that those keep working, but there are little nuances making sure that the carousel works or the image zoom works. You want to capture those, but nobody's going to leave your site if those don't work."*
> *-Front-end developer, 6-9 years of experience.*

> *"What we have when it comes to automated tests are lots of unit tests, I mean, it's a four-digit number of unit tests on the project."*
> *-Full-stack developer, 6-9 years of experience.*

> *"[There are] 1098 test cases...this is an old project, several years old and it's been continuously developed so it's larger. I've worked on several greenfield projects, new projects, and [the test suite is smaller] in the hundreds."*
> *-Front-end developer, 6-9 years of experience*

> *"Sometimes you have too many tests because there are some tests which tests for, I don't know, trivial stuff...I guess if you could test it but doesn't actually prevent issues."*
> *-Full-stack developer, 6-9 years of experience.*

> *"I think currently we have 20 to 24 tests roughly, plus maybe seven unit tests...35 automated tests. I feel like we have over a hundred manual tests."*
> *-Full-stack developer, 1-2 years of experience.*

> *"Per browser, it's about 200 test cases. I don't even know if that's big or small. It's growing every time there's a new feature, there's one or two more tests."*
> *-Front-end developer and JavaScript, 10+ years of experience.*

Errors in the Tests vs. Errors in the Code

Developers find there can be errors in both the code and the tests that they run. Errors can occur in tests for several reasons. Learning to write tests is a process, so when testing and coding go right in development they should work when deployed, but if that doesn't happen developers need to look at their own mistakes. In some cases how the test is programmed can be the cause of the problems.

> *"I went to a bootcamp and we were taught test driven development... you write the tests first and make sure they all fail. Then as you write the code, they all pass. It turns out that's only really useful if you know upfront exactly what your code should be doing. That's usually not the case...writing tests after the code has been a pragmatic way to do things...I used to write unit tests for all kinds of things, and then found out...they would fail for reasons other than the actual code failing. If I wrote it for things that weren't self-contained units, if they had DOM integrations or [if] it was trying to capture a user flow...I was mocking way too much. My unit tests have cut down and my end-to-end tests have built up, and I almost do no integration testing."*
> *-Front-end developer, 6-9 years of experience*

In some cases the test can fail because of problems with the code, or the code running in the browser. In cases where tests fail in Selenium, it is often an error in timing and not in the test. Developers noted being surprised by how often tests fail from an authentication happening in the browser that isn't timed correctly.

> *"...if I have a failure in a test it's because there's a problem with either my code or with the code running in the browser...Selenium is frustrating because it could be that Selenium didn't run it on time, so [my test] fails. Sometimes there's some authentication happening [in] the browser that runs the tests for some reason and it doesn't have the time correctly...the test fails because of that. It's surprising how much my test fails because of that reason."*
> *-Front-end developer, JavaScript 10+ years of experience*

Some developers felt that the failures occurred equally in both tests and the code. When refactoring, one developer experienced an even failure rate with their tests and their code. Part of that can be contributed to badly named fixtures. As they change the fixture name, they see less errors in their tests. Linting is also used to catch the issues in the code before testing. Improving the fidelity of the tests can improve validity of the tests and minimize false positives. If the test is a false negative it is likely that the tests are not resilient to changes in other parts of the code.

> *"I would say [whether a test catches an error in the application or an error in the code] depends on what I'm doing. If I'm refactoring, then it can be 50/50, because a lot of the time when refactoring, I might be changing the badly named fixtures for example, and that's going to break the entire test suite because they're calling out to those by name."*
> *-Full-stack developer, 10+ years of experience*

> *"There are classes of problems that [the automated] tests seem to catch very reliably and quite well. It tends to be the things that are simpler to quantify. Like I mentioned, linting,...but that part works well. If somebody doesn't follow the code style ... that doesn't follow the rule set we have that does get caught. When it comes to more real-world bugs, I don't have a great sense of how effective it is. So in a way that's a no, it's not that I know it's ineffective, but I don't know how effective it is."*
> *-Front-end developer, 10+ years of experience*

> *"By increasing the fidelity of your testing, you're actually making it useful in a meaningful way. If you have low fidelity testing, that's actually worse than no testing because it encourages you to ignore errors. So high fidelity testing is in fact the most important thing, or it's critical to testing at all, is making sure that all your tests are expected to work."*

Errors in the Tests vs. Errors in the Code

-Front-end developer, 3-5 years of experience

A common pain point with discovering errors comes when testing in the development environment does not replicate the real world environment. Developers would like their testing to discover errors before deploying. A benefit of developers having time to improve the fidelity of testing can encourage developers to find and address errors before they become a problem.

"The cases tend to be places where different systems interact together. An example I can think of recently is we had an issue with the configuration on one of our web servers...it was something that didn't work in production, but worked in the test environment and worked on our development environment. Our tests didn't sufficiently replicate the real world environment. Things looked like [they] were great until we deployed it...That's a case where in any individual pull request or code change we might've made, there was no explicit bug. Once all the pieces got put together, only then did a problem reveal itself."
-Front-end developer, 10+ years of experience

Testing Tools

There's an Air of Indifference About Testing Tools

There are myriad tools developers use to help them with testing. Generally, developers didn't have glowing reviews of the different tools. That said, it wasn't all negative either. Conversations about specific testing tools were often along the lines of, 'I think there's a tool,' or 'If I remember correctly,' which can imply that the tools they've used or heard of haven't made a strong impression.

> *"Cypress sounds familiar. To be honest, I don't think I've heard of any of those though.*
> *-Front-end developer, 6-9 years of experience*

> *"If [Playwright is] what I think it is, I think it might be somehow related to an open-source tool from Microsoft. I might be conflating it with something else that lets you write JavaScript tests...that's as much as I know about it..."*
> *-Front-end developer, 10+ years of experience*

> *"I have heard of Puppeteer, that doesn't still exist, right? I feel like they deprecated Puppeteer[1] because it used to work with Google, or maybe I'm thinking of the headless browser extension for Chrome that they were doing because Chromium eventually adopted the ability to run in a headless mode. People are now largely using Chrome headless as opposed to the dedicated, headless Puppeteer solution."*
> *-Front-end and Build Engineer/DevOps, 10+ years of experience*

The testing tools stakeholders wanted to learn about were:

- Accessibility testing: Lighthouse or webhint.io *(the latter of which no one brought up)*
- Component testing: Jest
- Cross-browser testing: Cypress, Playwright, and Puppeteer
- End-to-end testing a.k.a. integration testing: Cypress, Puppeteer's ARIA selectors, and Selenium
- Performance testing: optimizing for specific metrics like Core Web Vitals, or Lighthouse CI *(the former of which no one brought up)*
- Visual testing a.k.a. screenshot testing: Percy *(which no one brought up)*

1) Puppeteer is still maintained. We believe this developer was confusing PhantomJS with Puppeteer.

Cypress

Of the tools stakeholders were interested in during the 'Rapid Fire' round of questioning, Cypress was one that was frequently brought up by developers. Generally, developers were split on whether or not they liked the tool.

The Pros

- It's a popular tool
- Integrations
- Easy to use from a syntax point of view
- Documentation
- Real-time views when running locally

"[With] Cypress I feel it's possible for me to be a one-person QA automation department because it's a very robust tool. It doesn't take as much fiddling as using the React testing library. Our smoke tests are already very strongly end-to-end tests, and for that in particular, it makes sense to use something that's aligned that builds itself and does end-to-end really well. Because it's popular, it has a lot of integrations that are pretty useful. In particular, the Gherkin integration that allows us to run those specification files as separate tests that allows the two of them to talk to one another that was very useful. The testing library integration that the two of them can talk to one another."
-Front-end developer, 1-2 years of experience

"Cypress has been a good tool that I've used in the past with other screenshot testing tools...[Cypress] makes it relatively easy as an engineer from a syntax point of view of how you build the tests...because it's running in a browser in the background...the framework they provide enables you to easily interact with your application and get pieces of information out of it. There are additional tools if you dive into Cypress full-fledged...you can see the test operating and performing the actions that you're wanting it to perform in real-time, if you want to run them locally...in terms of some of the more traditional frameworks to achieve this can be more heavy-handed versus the helpers and functions that Cypress has. It's also well-documented, it seems to have a lot of community-driven support around it too, in terms of just recommendations in general."
-Front-end developer, 10+ years of experience

The Cons

- Limited in the browsers it supports
- It requires a lot of code

"[Cypress is clunky]...it has its own folder structure, which isn't very portable. It's not like Mocha or Chai. Those can sometimes be framework agnostic. When you write your tests you're writing them in Cypress. You download and you have to run the electron app. The physical runtime for testing is very heavy. It's not just a simple command-line tool."
-Front-end developer, 1-2 years of experience

"I actually am writing tests for Cypress right now. It's very powerful. It's a lot of code though. I think it's the best E2E tool I've used yet, but that doesn't mean that I love it. The biggest con is that it's only Chrome."
-Front-end developer, 6-9 years of experience

Playwright

Of those developers who had heard of, and even used or at least tested Playwright, the biggest selling point is that it addresses one of the more pressing cons of Cypress; cross browser support.

> *"I'm quite intrigued by Playwright...it's similar to Puppeteer, but they directly created it with more cross-browser options."*
> *-Front-end developer, 10+ years of experience*

One developer felt that a tool like Playwright would be helpful in convincing clients to allow developers to invest time toward testing.

> *"The direction Playwright is heading feels really promising to me. The idea to have a modern tool to test in all browsers, in all important browsers at least, especially even now that it included support for mobile browsers. This might be a real game changer...I wish that all browsers contribute to that and implementing protocols for that because, once that's done, we would have way fewer bugs and fewer excuses to not write tests...I could convince more clients to invest time into writing tests if there were a tool that worked reliably in all major browsers. That's not true as of today, as far as I know."*
> *-Front-end developer, 6-9 years of experience*

Puppeteer

Developers had similar sentiment for Puppeteer as they did Playwright; the tool supports more than one browser, which is nice but that support is perceived as experimental and developers would prefer to test things manually than adopt Puppeteer.

> *"[Puppeteer] has experimental support for Firefox, but it's not official and it doesn't support testing for Safari or Internet Explorer. That's an issue. For Internet Explorer, I'll just open up VM and test everything by hand. For Safari I'll borrow one of my friend's Macs and test on that. In the end, it's easier just to test everything by hand rather than using Puppeteer..."*
> *Full-stack developer, 3-5 years of experience*

> *"...I would prefer to spend more time on testing in general. Multi-browser testing is still really hard. Puppeteer [produced] a version that you can use for Chrome and Firefox, but the setup for Firefox is really cumbersome, really hard to do...everybody's waiting for when it will be easier."*
> *Full-stack developer, 10+ years of experience*

One developer mentioned that they worry about third party support, too. They use BrowserStack and know that BrowserStack uses Selenium. They worry that if they go with Puppeteer locally it won't work with BrowserStack.

> *"One thing that I have seen examples [of] because I can't spend the time learning [about new tools], there's this new tool, Puppeteer or there's so many, but there's one that it's supposed to be the Selenium killer. It does seem a lot nicer...it runs on the browser itself. All those weird errors that Selenium gets because, 'Oh, the timing wasn't right'... From what I've read, all those things go away, which I would love. And on top of that, I think it runs several times faster because it only runs in the browser, but I have to support IE and there is no handler...there's a handler for Firefox they created recently, which was great. I haven't heard about Safari or IE, and I can't use it. And another thing, let's say, we get IE and Safari, does it work with third-party providers?... BrowserStack and all those guys use Selenium. If I managed to get it to work locally, but I try to use it with a third-party and the party only supports Selenium [I'm] back to square one. It's probably one of those chicken and eggs, if you don't get enough users, there is no interest, but if you don't [build] it, there's no way that I'm going to use it. Hopefully someday soon we will get rid of IE...[it will] just disappear but I'm not holding my breath for that, unfortunately."*
> *Front-end developer, 10+ years of experience*

Selenium

Developers had similar sentiment for Selenium as they did for Puppeteer and Playwright. Cross-browser support, or lack thereof, is an issue with Selenium, too.

> *"Selenium is finicky to be polite, but it is better than nothing. We use Selenium in combination at times with a tool called BrowserStack so that we can run [regression] tests, do things, different machines, but certain browsers like Safari can be way, way more finicky in Selenium than other browsers like Firefox and then Cypress, we have not adopted. I'm still very much on the fence about it."*
> *-Front-end and Build Engineer/DevOps, 10+ years of experience*

> *"[Selenium]...has some of the same issues as a Puppeteer. I think it supported Firefox. I don't think it supported Safari though, which was annoying...this was a couple years ago. And again, it was more effort than it's worth, I could just see the exact same thing by hand, rather than wasting time trying to automate the testing."*
> *Full-stack developer, 3-5 years of experience*

Some perceive Selenium as a tool that inconsistently returns errors in the test, but to be fair to Selenium, the developer wasn't sure if the issue was with how the tests were written (they inherit the tests) or the tool itself.

> *"I didn't actually write most of those tests in [Selenium]. We had a person on our team who wrote them, and I'm not sure if it was just that person in particular or how they wrote the tests, but it seemed very flaky to me. They're put up on hooks for continuous integration and you push up your PR and it'd fail, so you just re-push your PR [with] no changes and it would pass. That was frustrating. You never were sure if you were actually breaking something or if it was just being stubborn."*
> *-Front-end developer, 6-9 years of experience*

Lighthouse

Lighthouse received some positive feedback from developers. One wishes they can use it, but finds it difficult to find repositories that connect it with Jenkins, another tool in their ecosystem. Another developer feels it offers a good developer experience for any project.

> *"I really wish I could get lighthouse into our CI/CD project, but again, it's just a time and capacity piece. We're using Jenkins... a lot of this stuff is stuff everybody's doing; it's ridiculous to me because Jenkins is also open source [that] I can't find a great repository of pipelines that everybody is trying to shoot for the same thing. Why can't I find a repository of common deployment pattern pipelines that deal with a lot of this stuff and have like, 'Oh yep, Lighthouse is baked in, and it runs here, and this runs here.' That feels like it should be [but] nobody has the time or capacity."*
> *-Front-end and Build Engineer/DevOps, 10+ years of experience*

> *"I think that while [developers] like to believe we like complicated stuff, the developer experience is also very important. What I like about Lighthouse is that it's straightforward and it's dependable. You know exactly what to do, and you do the same thing every time. If you're using Cypress you have to set up the whole project [whereas] with Lighthouse you know it'll work on any project because it's any website and you just click the button."*
> *-Front-end developer, less than 1 year of experience*

Referencing the insight, 'There's an Air of Indifference About Testing Tools, one developer knew of Lighthouse as the 'Google performance testing tool.' For them, the branding of the tool hasn't made a lasting impression. Another developer likes the tool, but has some reservations about the value it provides after trying it out.

> *"When you say Lighthouse I think of the Google performance testing tool. If that's what it is, then I use it, I just don't remember if that's the exact name for it. They clearly have a branding problem there."*
> *-Front-end developer, 10+ years of experience*

> *"[I have] not [used Lighthouse] in a professional setting. I've poked around with it...it's nice. It gives you a lot of information [but] I'm not sure how useful that information is [such as] the time to first contentful paint, time to first interact and all that. Again, it's like the performance testing, those numbers change. They could change for any number of reasons. I know some places use Lighthouse automatically on [their] PR pushes as part of [their] continuous integration and that will stop a push that degrades it past a certain point. I would be worried to do that because I don't think it's stable enough personally."*
> *-Front-end developer, 6-9 years of experience*

Unique amongst the other tools mentioned, some developers spoke about the accessibility features of Lighthouse. While one mentioned it off the cuff as a tool they use in addition to Gatsby, another recognized Lighthouse's accessibility features in the browser.

> *"...there are tools out there that help with [accessibility] that we just haven't had the capacity to be able to start leveraging, but it would be great if it was more immediate in the browser, which I guess Lighthouse helps with in Chrome a little bit."*
> *-Front-end and Build Engineer/DevOps - 10+ years of experience*

Adopting Tools - The Goldilocks Principle

Borrowing from the story, Goldilocks and The Three Bears, web developers who have the autonomy to choose their tools, have a strategy of adopting tools that are just right for _____.
What completes that sentence depends on:

- Developers personal experiences
- The project
- The size of a company
- The size of the development team
- Costs to use a tool
- The size of a tools' user base
- Available documentation for a tool
- The features of a tool

"To be honest how I would pick [a new testing tool] is which one has the most users, where am I going to get the most support? What has the most downloads on NPM? Where clearly is everyone at? I would choose based on that."
-Front-end developer, 3-5 years of experience

"...the major criteria would be that [a tool] should test real browsers. If it's a headless browser, it has its place, but it won't satisfy me...you need to test real browsers which are used by users and [how] convenient or [easy that is in the tool]. If it's easy to use, if it has documentation..."
-Front-end developer, 3-5 years of experience

"[Tools should be] easy enough to use that I actually use them [so] there's not [a] technical hurdle to understanding the tool. Affordable, I guess this would vary wildly. We're a mid-sized company dealing with small to mid-sized clients. If we're paying for services, some of them are $10 a month some of them are $1,000 dollars a month, but I'm thinking of services that tend to be in the range of $100 dollars or less per month. I realize it's a big range, but I'm not thinking of enterprise software that's going to cost us $500,000 per license or something like that."
-Front-end developer, 10+ years of experience

"...in the next project, I will definitely try to use one of these tools be it Cypress, PentF, Playwright, I'm not sure which one [it] obviously depends on the project, maybe they are already using something, but [I want to use] some end-to-end test tooling."
-Full-stack developer, 6-9 years of experience

Doing the research to even understand whether a tool is 'just right' takes time and effort, and that is a deterrent to some developers.

"...part of the initial research [is] to see how familiar does the structure of the code the tool requires look to me...the more familiar it is to me, I hope it will be easier to implement. The size of the community or how frequently the tools are updated, and it's just been resolved so that if stuff breaks or doesn't work quite right in the environment I'm using that it might be fixed sooner rather than later...then I have to check my own project to see what would be good parts or minimum requirements of stuff that I need to have tested, is it 10 files or 10 functions, or is it a 100 functions, then I have an idea how much stuff I need to be writing. I hope I get an idea sooner rather than later to see how much time it might cost me [to implement]."
-Front-end developer, 10+ years of experience

"I did some research already on different automation tools like Puppeteer and whatnot, and it [takes] time to test any of them to actually get into [the] requirements...doing that all from scratch, how they interact in your different projects...because the projects are quite different, it would always be a different approach."
-Front-end developer 10+ years of experience

Adopting Tools - The Goldilocks Principle

"...these days I'm pretty busy so I don't have time to look for [testing] tools and then learn how to use them."
-Full-stack developer, 3-5 years of experience

"I think the reason why I haven't dug so deep into [testing tools] is, even though at my current job, I'm at the scale now where we're hitting like 100K users a month, but...I'm the only web developer. In terms of my weekly capacity I haven't seen enough value add for me to learn how to use those tools versus my speed of shipping something that looks good."
-Front-end developer, 1-2 years of experience

Browsers

Supporting Browsers - It's an 'If Then Else' Approach

Before diving into the specifics behind which browsers developers support during their testing process and why, it's important to share developers' perceptions of the ecosystem that browsers live within:

- Browsers have different engines, and different versions
- Some browsers are considered evergreen, but many developers still support older versions. Some developers even support unsupported browsers e.g. IE
- Browsers run on different devices, which all run on different operating systems
- Devices run the gamut in terms of their pixel density
- Lastly, all of this complexity operates on top of a network. Networks vary wildly in regards to latency.

"Testing is frustrating because there are so many variations of browsers and operating systems. I mentioned a few times that we use BrowserStack, which we love because it gives us access to a bunch of devices, but we also hate it because they're virtual. They're not quite as fast and responsive as a real device might be. So that can be pretty frustrating. Sometimes we blame the tool, but also I think if we build a really complex JavaScript app and we're testing it on IE11 on a virtualized system, IE11 is slow. It's frustrating to test that because we've probably written something that isn't well optimized for that. So that whole process can be frustrating."
-Front-end developer, 10+ years of experience

"The volume of different variations of browsers, browser engines, and operating systems is a lot to test for [but] that has gotten much better. We tend to think of the evergreen browsers, which are the browsers that are kept up to date pretty well by the operating system so people don't have to upgrade manually. I would include Chrome, Firefox, and modern versions of Internet Explorer, i.e, Edge. Older versions of IE and Edge [are] super frustrating and holds things back and just knowing the right tools to build good tests in terms of automated tests, even what's possible there that I find frustrating. It's something I just don't know enough about, and it's a bit overwhelming and I feel like there's more we could do, but I just don't know what it is."
-Front-end developer, 10+ years of experience

"It's a headache to go through all the browsers...with different screen sizes and different DPIs and all that stuff, it just makes the project more complicated because you spend most of your time [on] testing then implementation."
-Full-stack developer, 3-5 years of experience

Knowing that browsers are a part of a bigger ecosystem is necessary context for understanding the underlying complexities contributing to the decisions for which browsers to support in a testing process and why.

Stakeholders had assumptions that developers want to test as many browsers as they can and they're just limited by the browsers their tools support. What we learned is that developers' decisions for which browsers they support does not depend on the tools they use. If anything, what browsers they support drives the decisions for which tools they use or don't use.

The reasons why developers choose to support or not support different browsers are:

- Market Share
- User Base
- Project Requirements
- Time Tradeoff
- Access to a Browser

Supporting Browsers - It's an 'If Then Else' Approach

Market Share

Some developers aim to support the browsers that have the most market share. This seemed to be especially true for developers that have ten or more years of experience. Additionally, within this group, developers tended to support the browsers they most love, even if it doesn't have a high market share or is prominent in their user base.

"For me, [which browsers we support] is a political decision. For the browser to have some say they need some market share, to have some market share they need users. I always stay on Chrome, which has the biggest market share [so it needs to be tested]. Safari has the second biggest market share, at least in Australia. This is true for mobile and on desktop these days as well. The next one is Edge, creeping up at 5-7%."
-Front-end Developer, 10+ years of experience

"Pretty much my main browser is Firefox...pre-Firefox times was Netscape and I've been supporting it ever since. If I am happy in Firefox I double check in Chrome because it has such a big market share and it always needs to work. After that in Safari because it's the next biggest market share, especially on mobile..."
-Front-end Developer, 10+ years of experience

"The most important thing for us on phones is whatever the default browser on the phone is that people are buying. That tends to be mobile Safari on iOS devices, and either Chrome on Android phones or whatever variation Samsung would have. A lot of the Samsung phones seem to have a variation of their own browser...It used to be the case where there was also the Android browser before Chrome became the default Android browser. We're not testing for that much anymore. Several of us are Firefox for Android users, so it gets tested and we love it, but I don't think a lot of our customers are using it, so we're not testing it heavily for that perspective. Then, it would be whatever browser ships on Samsung phones for the last couple of years, Internet Browser or whatever it's called these days."
-Front-end Developer, 10+ years of experience

"Here in Japan, there are still many Internet Explorer users. Many companies are not [changing] to more current or modern browsers because...the updating process will involve much time and effort from them. Unfortunately, for us, we need to support [Samsung] Internet Explorer."
-Full-stack developer, 3-5 years of experience

User Base

Because supporting browsers comes at a cost in terms of the time and resources required to make something work across browsers, some developers adopt the strategy of meeting their users where they are. They look to their user base, instead of market share, to see what browsers they should support. In some cases, if there aren't enough users on a browser they will stop supporting it to save time and money.

"...usually [developers] just work on one browser, but because we are talking about user interface websites, we need to test a lot of other browsers, including mobile browsers. According to our statistics, half of our users [are on] mobile browsers. This is true going into developing countries. You go to India, outside of major cities, nobody has a desktop, everybody accesses the internet exclusively, [on] some really cheap, extremely low CPU throw-it-away phone. That's our target, the same goes for China."-Full-stack Developer, 10+ years of experience

"We run monitoring on our users, so we know that Chrome is by far the most popular browser for our product. We need to make sure that Chrome is perfect."

-Front-end developer, 10+years of experience
"We have less than 1% of users on Internet Explorer. So, we decided to cut it because it was expensive to support. It was unproportionally high. It was a money decision not a technical issue."
- Full-stack Developer, 10+ years of experience

"[What browsers I support] depends. I normally do Firefox, Safari, and Chrome. One person, there's an old guy and he still uses Internet Explorer on his XP machine. So I had to make sure that worked."
-Full-stack developer, 3-5 years of experience

However, one developer was astute to point out that this approach may exclude potential users. If you only support the browsers your user base is already using, then you may be missing potential, new users who are on different browsers.

"When it comes to browsers on the desktop, we're primarily thinking about Chrome for Windows and Mac as it's very popular, a lot of people use it. Firefox for Windows and Mac. We'd also be keeping Linux in mind, although if it works in Chrome or Firefox it's probably going to work across all platforms...whatever Microsoft is calling their browser currently, including IE11. Anything before that we've long since disregarded primarily because it doesn't even support the basic security requirements for things like e-commerce, but we often do still support IE11 and any newer version of IE or Edge, [also] Safari on the Mac. We don't bother with testing significantly in any other browser, unless there is a special consideration. We're not testing in Opera, actually... there are several browsers that are based on the same rendering engine as Chrome, like Brave and Vivaldi. We don't bother testing on all those individually, partially because not many people are using them when they visit our sites. Though, that can be a catch-22."
-Front-end Developer, 10+ years of experience

One developer mentioned they would like to make decisions about what browsers to support based on their user data, but they don't have that information currently so they go by market share.

"We have no criteria, very classic. I'm just going by market share, essentially. I would really love to get [user] data and I would love to have it so that when we do make those decisions, we're doing it based on what our customers are actually using. At this point it's literally just me assuming."
-Full-stack developer, 1-2 years of experience

Project Requirements

Some developers work as freelancers, and let the project requirements and clients' requests determine what browsers they support, and the requests vary. Some clients request not only the latest version of browsers, but the last several versions.

"This job, we're only doing Safari and Chrome, not even mobile devices. My last job was IE 10 and 11, Edge, Chrome, the last two major versions of Firefox, the last two major versions of Safari. The last two major versions of Chrome for Android...there were two browsers that were very popular in Asia and Africa. And I want to say one of them is UC browser."
-Front-end developer, 6-9 years of experience

"[What browsers I test in] depends on the clients. Most clients [decide] which browsers they want to support and in our current client [chose] all [the] current major version and the two major versions before...Firefox's current version is 85. So [we officially support] 85, 84 and 83."
-Full-stack developer, 6-9 years of experience

"...iOS Safari, Chrome and Firefox on Android, that seems to be all that I've ever been asked to support. I hear that the Samsung browser

isn't the same as Chrome and you actually need to pay attention to it, but I've never had a client ask me to, so I usually have a quick look in BrowserStack and if it looks okay, then I just pretend it's okay."
-Full-stack developer, 10+ years of experience

"[I support] as few browsers as I can get away with. If a client tells me I'm allowed to drop IE11, I will absolutely drop [it]. I absolutely [support] all the evergreens...always the latest versions of Firefox, Chrome, and unfortunately Safari and Edge. Edge used to be the oddball, and now Safari is the oddball, and IE11 of course...this particular client has a lot of federal users from government agencies...IE11 is a must because of that."
-Full-stack developer, 10+ years of experience

One developer mentioned they will test important features in more browsers than they would normally support, e.g., Edge. And, the reason they don't normally support Edge is because they are on a Mac and struggle with virtual machines.

"One browser, I didn't include is Edge or Internet Explorer, but Internet Explorer is one of those browsers that isn't supported by all applications, so that doesn't make sense to test it. Edge officially is supported and well, it's pretty much a struggle to start a virtual machine with Windows, which is way slow and it doesn't work [well] with a Mac keyboard and it isn't a great experience, so most of the time I don't do that, except if it's a really important feature."
-Full-stack developer, 6-9 years of experience

Time Tradeoff

Remember, a critical theme of this research study is that testing takes time, and time is a precious resource. For some developers, time and complexity are factors when deciding what browsers to support. If supporting a browser takes too much time or is too complex, developers will opt out of supporting it.

"I also check IE, but part of me feels like, 'Screw it.' Some people on IE, or a browser popular outside the US, I don't even know what it's going to look like, but it's not worth my time to optimize for that."
-Front-end developer, 1-2 years of experience

"I think that WebDriver and Selenium support other browsers, but in terms of what we needed and setting things up, we just went with [the Chromium] default option. [We] didn't take the time to investigate other browsers. I couldn't say if it's just laziness or if we just are averse to dealing with setting up more things. In an experience that has often been frustrating, it's hard to make yourself want to go back and add more pieces to that puzzle."
-Front-end developer, 3-5 years of experience

Access to a Browser

Sometimes, a developer doesn't have access to a browser and therefore they don't support it even if that means they may be overlooking a portion of their user base.

"Perhaps I'm overlooking a large portion of the user base and leading to a bad experience just from my lack of experience and lack of exposure to Samsung Internet, because I don't have it, So [it's] not easy for me to test."
-Full-stack developer, 10+ years of experience

Supporting Browsers - An 'Anything Goes' Approach

While some developers had specific reasons for supporting or not supporting certain browsers, other developers took an 'anything goes' approach. For these developers, there is no standardization. Part of what contributes to a lack of standardization goes back to the timing and lack thereof.

"I don't know what [browsers] my teammates are using. [It's a] classic issue, we have pretty much no standardization whatsoever. I pretty much always test Chromium using the React developer tools, because I'm assuming that most of our customers are going to use Chromium, especially because we're a B2B project. I'm assuming that they're all going to use Chrome. If there are cross browser distinctions at this point at the company, we don't really have much time to devote to them. Given I have a very limited amount of time I'm trying to focus on what's probably going to be the most common browser."
-Full-stack developer, 1-2 years of experience

"[We test] Chrome in all cases, Firefox if somebody complains, and Safari if somebody complains. Typically someone on the team will have a Mac laptop of some kind, so Safari will get thrown into the mix. For my personal project I target mostly Chrome. Apple is absolutely infuriating with their lack of support for web standards."
-Full-stack developer, 1-2 years of experience

"[We] usually [test in] whatever browser you're using as an engineer. Chrome is usually heavily covered. Safari can sometimes be covered. In terms of Firefox, I don't know that we do any explicit testing on that browser, but those would be the three that we're supporting."
-Front-end developer, 10+ years of experience

"...There's a group effort among other members of our team who are not developers to use and audit the software. We have a pretty large manual test bank that we go through. When people are doing that, they're using their own browsers. Chrome and Firefox on our side are the most common. We don't use Edge when we're testing, which even as I'm saying it now, that's probably not very good. We probably should be using Edge to test. I know a lot of our clients get set up with Microsoft computers and just use Edge because it's there when it starts. So, definitely not perfect."
-Front-end developer, 3-5 years of experience

Resources for Learning About Testing

Resources for Learning About Testing

Not all web developers reported resources for continued learning. Some cited a lack of time, or a lack of knowledge around resources to learn about testing. A few reported learning from their coworkers. Of the sources reported, they ranged from different podcasts, social media sites, to a variety of websites and web tools, and others, including:

Social Media

- Medium
- Reddit
- Twitter
- YouTube

Websites

- ARS Technica
- Browserhacks
- CSS Tricks
- Google
- Hacker News
- MDN
- NPM Compare
- Turtorialspoint
- W3schools

Other Sources

- Linux Community
- Smashing Magazine
- Daily.dev
- Shop Talk Show podcast

"...MDN is usually my go-to tool for most of these questions or Stack Overflow, depending...Stack Overflow if it's a specific question, MDN usually, if I already know what I'm looking for, and just need the whole picture again."
-Front-end developer, 10+ years of experience

"[To keep up with testing] I used to have a lot of channels I subscribed to on YouTube...Medium, and Reddit. But now I have this cool plugin [daily.dev] installed. When I open a new tab, they have all the latest tech news...If you create an account, you can even subscribe to different channels, tags, or stuff you want. For the past two or three months, this is how I keep myself updated with tech news."
-Full-stack developer, 3-5 years of experience

"People from the team follow industry news and press, there would be blogs and social media of other developers that we would follow. Examples include the hacker news site or ARS Technica. ...There's a podcast called the Shop Talk Show that's geared toward front-end developers. They would get into talking about some of these testing workflows... MDN is a common resource for us."
-Front-end developer, 10+ years of experience

"Can I Use...has been immensely useful to a lot of different people. Then, MDN's wonderful documentation has helped a lot of people...before MDN everything was scattered around, and it was all a mess. You couldn't find anything authoritative, unless you wanted to read the W3C specs, which are dense. It's hard to get any meaning from them if you didn't actually write the things as far as I can tell."
-Full-stack developer, 10+ years of experience

Conclusion

Conclusion

The sponsors behind this work have familiarized themselves with the results of the MDN Web DNA Testing Follow-up research, and have offered their perspectives on what organizations will do to address developer pain points.

Mozilla

"Automated tests are central in allowing developers to create high quality experiences on the web. The MDN Web Testing research provides us with crucial insights into the state of web application testing, highlighting the ongoing demand for reliable and easy-to-author cross-browser tests. We look forward to working with the other browser vendors, and with providers of testing tools, to develop and standardize the next generation of browser automation APIs and help developers create the best cross-browser web applications."

- Andrew Overholt, Senior Director of Web Platform

Google

"The MDN Web Testing research has greatly improved our understanding of the challenges web developers face with testing, especially with end-to-end tests running across browsers. To support the next generation of testing tools, we're working with all browser vendors on WebDriver BiDi as a new foundation for cross-browser testing support. It's great that the MDN research confirms that our investment will help address significant pain points that developers are struggling with."

-Daniel Clifford, Principal Software Engineer and Manager of Chrome Developer Tooling

Microsoft

"The MDN Web DNA is a crucial tool for our team to listen to developers and align our product work to their needs. The new Web Testing research provides deeper insights into the frustrations and challenges developers face with cross-browser testing and tools. This will directly help our team plan improvements to our tools, automated testing solutions, and documentation."

-Kyle Pflug, Principal PM Lead Edge Dev Ecosystem

W3C

"As most developers know, testing is both critically useful and often hard to get right. The insights gathered by the MDN Web Testing research will help guide the W3C community ongoing work in this space, in particular the WebDriver work happening in the Browser Testing and Tools Working Group. The report makes it clear that any friction in the testing process makes it likely to end with less testing or less good testing. The W3C Developer Relations team is committed to bring a more systematic approach in ensuring the testing experience be considered as integral to the design of W3C technologies."

-Dominique Hazael-Massieux, Head of Developer Relations

Bocoup

"The MDN Web Testing research has confirmed some of our thesis regarding accessibility, in that manual accessibility testing is a pain point and developers want to work more on accessibility but often aren't set up to succeed due to company priorities or lack of accessibility policy throughout a project lifecycle. We will use these insights to inform our work to develop tooling to automate accessibility testing with screen readers as well as improve accessibility best practice documentation at the W3C, which we hope will have a positive impact towards addressing these pain points."

-Simon Pieters, Principal Open Web Engineer

Conclusion

"As an industry, we need to make this [testing] process more straightforward, less error prone. It's so much work, and so many parties made the process complicated."
-Front-end developer,
10+ years of experience

"My biggest gripe is the [testing] ecosystem has grown so broad, which is good and bad. There's almost too much choice. Some of [the tools will] eventually become the dominant ones in terms of their community [of users], but there always seems to be some new testing framework or solution...we have to decide, 'Do we migrate to this new platform to try [and] increase our confidence in what we're doing, or do we stick with what we have?' I've written tests in at least six different frameworks over the last decade. The web is very fractured to a degree."
-Front-end developer,
10+ years of experience

"It should take less time to do all this."
-Front-end developer,
10+ years of experience


MDN web docs
moz://a
pinpoint