EPISODE 1292

[INTRODUCTION]

[00:00:00] JM: There are over 4 billion people using email. Many people use email for business communications or quick questions to colleagues. And they're sending repetitive template-based information to potential customers and freshly hired employees. They're repeating lots of the same phrases. We actually repeat phrases in a lot of written formats. How often do you copy and paste the same thing to multiple people?

The company Text Blaze is making the workday a little faster, more productive and more convenient with their short cut to snippet software product. With Text Blaze, you can save any snippet, or text, or template, including templates that need fill-in the blank sections to keyboard shortcuts. And then you type that keyboard shortcut into Gmail, or Google Docs, or LinkedIn, or Salesforce, or wherever else you need to use your saved snippet.

In this episode, we talked to Scott Fortmann-Roe, who is the CTO at Text Blaze. I hope you enjoy this very interesting episode about browser extensions, and text snippets, and how to be more productive. Scott is a very experienced engineer, and I think you're going to enjoy this episode.

Our first book is coming soon. *Move Fast* is a book about how Facebook builds software. It comes out July 6, and it's something we're pretty proud of. We've spent about two and a half years on this book. And it's been a great exploration of how one of the most successful companies in the world builds software. In the process of writing *Move Fast*, I was reinforced with regard to the idea that I want to build a software company. And I have a new idea that I'm starting to build. The difference between this company and the previous software companies that I've started is I need to let go of some of the responsibilities of Software Engineering Daily. We're going to be starting to transition to having more voices on Software Engineering Daily. And in the long run, I think this will be much better for the business, because we'll have a deeper, more diverse voice about what the world of software entails.

If you are interested in becoming a host, please email me, jeff@softwareengineeringdaily.com. This is a paid opportunity. And it's also a great opportunity for learning, and access, and growing your personal brand. Speaking of personal brand, we are starting a YouTube channel as well. We'll start to air choice interviews that we've done in-person at a studio. And these are high-quality videos that we're going to be uploading to YouTube. And you can subscribe to those videos at YouTube and find the Software Daily YouTube channel.

Thank you for listening. Thank you for reading. I hope you check out Move Fast. And very soon, thanks for watching Software Daily.

[INTERVIEW]

[00:03:00] JM: Scott, welcome to show.

[00:03:01] SFR: Thanks for having me, Jeff.

[00:03:03] JM: You have built a very interesting company. And I want to get into what the company is. But first, I want to talk about Chrome extensions, because Chrome extensions have always seemed like an amazing platform with lots of potential. But the problem is that nobody wants to install Chrome extensions, right? Isn't it really hard to build a business around a Chrome extension?

[00:03:27] SFR: I wouldn't say that. So Chrome, of course, is a huge platform. There is I think over a billion users on the Chrome, who use Chrome. So it's a huge platform with a lot of potential users and a massive market. The platform is very powerful. There's a lot you can do with it. There are some concerns about security and privacy. In various cases, there're been well-publicized cases where spammers or scammers have created Chrome extensions that are either adware, or redirect your search page, or things like that.

The Chrome team over, I'd say, the past two years has really been trying to crack down on that. So they've developed a number of new policies, technologies, review processes to really try to address that problem. So, for example, when I started Text Blaze two years ago, you submit your extensions to Chrome Web Store. 15 minutes later, it gets published. Now you submit to

the Chrome Web Store, and either an hour later, or up to three weeks later, it's published, because they've added much more review processes where they will actually have some time go manual review, make sure it's legitimate, and confirm all that before publishing it. So they're taking it very seriously, which has some impact on development as an engineer, because when your review process might take three weeks, that obviously impacts how you can develop and sort of your release cadence. But it's good to increase the trust of the platform.

[00:04:43] JM: So what is the process of getting a Chrome extension accepted by the store these days? How similar is it to the onerous process of getting your mobile app accepted?

[00:04:55] SFR: I've never done a mobile app, but my understanding is they are converging. So again, two years ago, you just write some code, submit it. They run some automated checks. And 15 minutes later it's good to go. But now, again, it takes a lot more work, because it could be a fast review, and it could be up in an hour. But it could be something that takes three weeks. And if know the review might take three weeks, that obviously puts certain constraints on development. Namely, you need to make sure when you push code, there are no bugs, right? It's good code. Because if you push out a bug and – Well, it could take three weeks to get that fixed, right? Because it could take three weeks for your next release to get through the review process. So you really got to do a lot of testing, quality assurance. Make sure it works great.

We do a lot of automated sort of CI/CD testing, where we use Docker images, where we have different Chrome builds. So we have the current Chrome version, but we also have up to six older versions, because some of our users are on the long tail and don't update that much. And we run through a large number of Selenium tests where we actually spin up the Chrome instance in Docker. We install the extension. We run it on a whole bunch of site.

So **[inaudible 00:06:04]** Text Blaze, what it is. But for us, the key thing is compatibility with different types of editors, because we influence editors in web browsers. So we have this very extensive test suite that tests all the popular editor platforms. So like CKEditor, TinyMCE, Quill. Just any sort of editor that **[inaudible 00:06:24]** any sort of editors out there. We have test to make sure extension works correctly with it. And then also a whole bunch of different websites. So like Facebook, Google Docs, Gmail, etc. So we just run through – Whenever we do a

release, we test all those different sites across up to six different Chrome versions, and make sure the extension works correctly across them.

[00:06:43] JM: The product is not super easy to explain. So I'd like to take a few cracks at what Text Blaze actually does.

[00:06:55] SFR: Please do. This is a constant issue in our marketing.

[00:06:58] JM: Right. Okay, a good example is I'm a support agent. Basically I'm doing like some set of tasks like over and over again, every single day with some variability. So if I'm, for example, working as an Amazon support agent and I'm managing returns, for example, that's a pretty straightforward example. You go into the Amazon chat system. You say, "Hey, I need to return order identification number 56329." And the support agent says something that is basically a canned response, like, "Okay, I understand you need to return this item. And let me get that started for you. Let me kick-off the returns process."

If I was a support agent, I would be typing that command 500 times a day. But with Text Blaze, I could just type /return, like the word return, the string return, and use Text Blaze to expand that into a response, into a commonly used response. Have I explained that correctly?

[00:08:06] SFR: Exactly. So the basic idea is you create templates. You give those templates shortcuts anywhere on the web, any text box on the web, where you type that shortcut, the template is inserted. So in your example, your template is here's how to return it, or your return is approved. And your shortcut is /return. And then in your intercom, or Zendesk or your internal company's chat system, you type /return, and that message gets inserted right away. So that's the basic idea of the product.

So the basic idea is kind of pretty simple, right? Like it's not rocket science. Templates have been around forever. There are other competitors to us out there that do things like this. Templates are built into every serious product already, right? And your Gmail has templates. Your Salesforce has templates, etc. But we take that sort of simple concept of, "Hey, a template and a shortcut to save that automated – Save that repetitive typing." And we take it a lot farther

than we think anyone has ever really done before, which is sort of why we think it's really exciting.

So before I get to like the band stuff, I sort of like to geek-out on the band stuff, like formulas, and programming, and all that cool stuff. But like, really, from a user standpoint, just that basic template shortcut is incredibly valuable. Like you have the people who are doing the same thing over and over again, and it just saves so much time and reduces mistakes too. So instead of typing that out and potentially make a mistake, or copying and paste it, it just does it and it does it fast, it does it seamlessly, again, wherever they are. So it's just hugely valuable for people who have the repetitive tasks.

I want to make it like clear, like this is not automating anything. This is not like replacing the agent. This is not like you're talking about chatbot somewhere who has sort of programming. Like we all sort of hate that, like when we're talking with a robot. This is still the humans in the loop, the humans evaluating the customer messaging and deciding, “Hey, how am I going to handle this?” And they can always jump out of their script if it's not working and the customer has unique issues.

Let's get to sort of the advanced – If that's okay, can we get to sort of the advanced stuff?

[00:10:03] JM: Yeah, please, please.

[00:10:04] SFR: Okay, awesome. So you start with just that basic text template, right? That customer return message. And then we have more advanced feature the user can use in the template. So one is forms. And forms basically allows the user to embed form fields, like text boxes, or drop down menus, or check boxes in the templates. So when they have a template about return, they might say, “Hi, Jeff. We've approved your return. It will arrive in 10 days.”

Now, in the template they might say like, “Hi XXX–” That's something we do. Sometimes we could tempted like XXX that we have to remember and go and replace the XXX with the person’s name. Maybe they forget. They send it. It's embarrassing. It’s no good. What forms do is you can do high form texts, or high textbox. And now when you use the snippet, a little window will pop up and there'll be a little text box right where that name was. And you can enter

Jeff, or you can enter Scott, or whoever you're talking to. So it creates a little form you can fill up when you use a snippet. So that's step one that's super useful when sort of making these snippets a little dynamic, sort of standardizing processes where you want users to enter information, and they use of templates, etc. So forms. So step one. Sorry. Any questions about that? Is that super clear?

[00:11:18] JM: No, keep going.

[00:11:19] SFR: Okay. So you take forms. And then another thing we've done, we've added formulas. So like in Excel, where you have your little grid and then you create a formula that references cells and do math or other logic, we get the same thing in Text Blaze. So we have like a called command. So there's the form text command for a little textbox. There's the for menu command for a drop down menu. And then there's the equals command, just a little equal sign where you can write arbitrary sort of mathematical formula. So you could do something simple, like one plus one, right? And then when you use the snippet, instead of inserting one plus one there, it would insert two, and it would do that math for you. Now, that's not super useful by itself. Where it gets powerful is it can connect the form fields to the formulas.

For example, if you had a – We'll move away from the return example. If you had some sort of invoice snippet, or some sort of sending out a quote snippet, or a receipt. You could have a price form field that you could enter the price of the product to $100. Then you could have a formula that referenced that price field. So you could have for price, say $100. And they kept price of tax where the formula is the price times 1.1, or 1.2, for a 10% or a 20% tax. So now you've created this dynamic where when you open up, a little box shows, and you can enter the price at $100. And the price of tax will spit out 110 in real time as you add it. So all this is updating live.

So this lets you start to embed really dynamic logic in the snippet where you have the form fields, and then you have this dynamic logic that's based on what you enter. So you can start to embed your business logic in the snippet. You can take that and you can spin further. We have sort of like if commands. So that's sort of like a formula, but it will conditionally hide and show text. So you could say, "Hey, if the price is above $100, offer free shipping. If it's less than $100,

don't offer free shipping." And you could encode that directly in the snippet, and it will be fully dynamic as the user edits it.

We start with plaintext. And we really want to do plain text as well. But then we have these ways to extend Text Blaze beyond it to create these really interactive documents. Or you can embed business –They're like almost apps, but like documents that are dynamic and respond to the user input. Anyways, so that sort of, again, start simple text, but offer the progression path.

[00:13:40] JM: Convince me that this is not a narrow tool for super users of niche web automation tools.

[00:13:53] SFR: Oh, yeah. So I would say two things on that. So, well, first, we have about 80,000 actives on the Chrome Web Store. So we're growing steadily. So we're getting great engagement there. But in terms of sort of like how I think about the product and how we think about – I'd say two things. One, I look at two sort of analog that I think are super interesting. One is Excel. Excel, or Lotus, or Google Sheets, right? It's like one of the most successful sort of programming computing paradigm ever, right? Like this is incredible. This sort of simple table, right? It's an incredibly powerful. And I think it's that way because of two things. One, it's super simple to start using it, right? Like your grandmother can use Google Sheets to make her Christmas shopping list, right? And it works great for that. It's awesome. Just like a static list. Maybe she does some coloring. Maybe she adds something – Maybe – But like just a static list is amazing. But then it offers formulas where you can now make things dynamic. And then beyond that it offers VB script or App Script, and then like you're running your business on it and you're doing these incredibly powerful things.

So, first, we want to sort of do what Excel is doing, but for documents, right? Where Text Blaze starts super simple. It's just plain text. Anyone can adopt it. Anyone can use it. We have users across every vertical. So customer success, customer support, what you mentioned. And that's our main one. But we have users, and property management, and healthcare, and teaching, doctors. Like so many different people have value in this type of software. I think the three most interesting vertical users in my opinion, we had one who had reached out to me who was a sushi chef. We had one who was a priest. And then one who was a tarot card reader. And she

use Text Blaze to speed up her creation of her daily fortunes every day. So the use cases are sort of very, very wide.

Anyways, back to sort of the Excel analogy. You start really simple with text, because you don't need to have this dynamic stuff. Like it just works for plain text, and it's super simple. And it solves that email. Like you probably reach out to many different people of your podcast scheduling. I could make a Text Blaze template for that, send it off super easily. But then it offers the formulas. It offer dynamic behavior. It offers the ability to integrate with other apps easily. So it allows you to over time do more and more of your sort of business logic that's fed to a company in Text Blaze. And we're seeing that in Text Blaze. We're seeing people start out very simple. And then we're seeing some companies evolve and use more and more complex functionality. Not all companies. Some just keep it very simple and sort of figuring out how to educate people over time as a key challenge for a company. But we're seeing that growth patterns happen. So that's sort of one point in my mind. Any questions about that? Otherwise, I'll move on to the second one.

[00:16:37] JM: You know, I'd like to zoom-in on one particular use case to help make sure that people are not losing the lead here. We already kind of ran through the support agent example in some detail. But another one of your testimonials on your website is from a therapist. So I can imagine a therapist in a remote therapy session taking notes on a patient. Why would Text Blaze be useful for that use case?

[00:17:03] SFR: Sure. Absolutely. So we actually see a fair amount of adoption in the medical space, or a fair amount of interest in it. And it's just healthcare professionals. They have a lot of sort of structured input where they have various questions after a consultation or a meeting, and either freeform questions or with various potential answers. And Text Blaze has like text boxes, it has a drop down menus, that makes it easy to implement and structure those sort of questionnaires.

Additionally, with the dynamic logic of Text Blaze, you can implement sort of smart follow-up questions. So if you asked, "Hey, are you a smoker? Yes or no?" And they say, "Yes," you can offer a follow-up question about their smoking habits. And if they say no, then you can hide that question. And that question doesn't show up. So we actually have one doctor in my mind up in

Australia with these super complex sort of diagnostic templates where he goes through. He asked a whole bunch of questions. And reveals, hides additional questions as the top ones are filled out. And finally, it spits out an answer or a recommendation, “Hey, you need to come in for a checkup,” or whatever the case may be.

So anyway, Text Blaze have that ability where they can implement sort of whatever they want, because it's just a text document at the end of day that can lay out things, dial things how they want. But then they can embed the form fields. And if they want, they can also embed it a bit of dynamic logic for follow-up questions or additional sort of behavior.

[00:18:30] JM: Okay, I get it. At this point, I think the listeners probably get it. This is a dynamic tool for making templated and advanced workflows around typing plain text on the Internet.

[00:18:47] SFR: Sorry, it's also styled text. We also of course have styled – I just want to mention that.

[00:18:51] JM: Styled texts as well. Very big domain. Tell me about the engineering. Like we’ll start with just the level one engineering.

[00:19:01] SFR: Sure, absolutely. Not to return to level one.

[00:19:05] JM: Okay, just give me the high-level engineering. And then we can dive down in a little more detail.

[00:19:10] SFR: Right. So Text Blaze, and I think many products are like this. Text Blaze is one of the product if you look at that. And you're like, “Hey, I could create that in a weekend, or two weekends max, right?” And as engineers, we all have a bad habit of both doing this and critiquing it, right? Like look at Facebook. “Oh, yeah, Facebook. No problem. It's easy to make.” And we all know how it exists. But like it’s still easy to do, right? And Text Blaze, right? You're looking at it and you’re, “Great.” You have some text templates. Okay. You insert them into page. Okay, yeah, maybe there's some sharing and collaboration, but that's not too hard to do.

But there's actually a lot that goes under the hood of Text Blaze and there's some things about sort of like talk about on the engineering side that might be interesting and might be sort of unexpected. So the first thing that is pretty surprising is how hard it is to insert text into a webpage. You would never think this is one of the hardest part of Text Blaze. But just the act of, "Hey, I have a template. The user typed this shortcut." Now let's get rid of that shortcut and put the template text into the page. Then actually, it's quite, quite difficult. And basically why it is is the web is full of so many different types of text editors, so many different approaches to text editors.

On the simple level, the easy level, you have form elements like the text area tag, or the input tag and experience. Those are relatively simple to do, because generally, you don't have too much dynamic behavior or event listeners operating on them. Like maybe you have to fire off on change event, when you enter text and maybe do some other things. But it's pretty straightforward. But then once you get to style text, so style text, right, and the way to do it – And I'm not sure how technical, but like, let me know if this is too basic or too advanced.

[00:20:48] JM: No. It's a show for engineers. So go for it.

[00:20:51] SFR: Right. So the way to do it, and I'm sure many of your listeners know this, is you have this content editable tech, where you just stick the content editable attribute on any HTML element and then the text within it will be editable by the browser. So that's how basically every or almost every – We can talk about some exceptions, but almost every rich text editor works on the web. You have a div, kind of editable, and then the text inside it is editable HTML. So that's the basis for, again, almost every edit attribute. But there're so many variants to how they behave.

So 10 years ago, 15 years ago, this was simple. You had sort of old school editors, like TinyMC3, or older version of CKEditor, and basically that they will just mark an area of content editable. They would add some buttons to make things bold, or add links or etc. And the user would just edit the text and to be editing HTML, they do it. And then when the parent app want to get the page, they'd grabbed the HTML. They would then sanitize it and save it somewhere in a database.

So for those types of editors, inserting text is straightforward. You can just edit the DOM. You can just stick in your new text, add a new span, put in the new text, and is updated. And that works. However, again, that's 10 years ago. Maybe five years ago or somewhere in that, a lot of that started to change. You had sort of newer editors, fancier editors, and they wouldn't just trust the DOM. They wouldn't say, “Whatever is in the DOM, that's my text.” They would keep their own internal state representation. So you have editors like Quill, where Quill has its own internal format that’s called the Delta format. And that is the representation of the document for Quill.

Now, they have that the mirror that out to the page, but the master is this internal representation. So if you just go edit the DOM and add text or remove text, it's going to the DOM and the internal representation will not match. And the editor may crash. It may behave badly. It may copy stuff, delete stuff. It won't work well. So with these newer editors, you can't just go in and edit the doc. You can't make that easy approach. You have to be more sophisticated.

So there's that branch. And then even a little later, there's a whole set of new editors that before input event. The input event, these newer technologies that are coming out. And they work similar, they have their own stake, but they use a whole set of different events on how they track things. So how you work those new editors, you actually have to emulate exactly what would happen if that user inserted text. So a lot of the Text Blaze insertion is carefully emulating what would happen. Again, if a real browser user came in and stuck some text in the page, firing off the exact sequence of events the browser expects, and again, just doing that as closely as possible. So whatever the editor, they see the text inserted, just as if the user had typed that text themselves. So a ton of work has gone into that. So that's one thing. I do want to mention that I said there were some editors that don't follow this. So there's a few. And I just maybe highlight one, if you think it might be interesting.

[00:23:48] JM: Yeah, go ahead.

[00:23:50] SFR: So Google Docs, right? I'm sure everyone uses Google Docs at some point in time. When you go to edit a Google document, you have this big canvas. You can edit things. And when I say canvas, I don't mean HTML canvas. I mean an area where you can edit things. And you might think that's one big content editable. That would be the standard way to implement. However, Google Docs actually does something very different. In Google Docs,

what you're editing or what you think you're editing, you're not editing that all. That is just a picture basically. It's a DOM structure, but a static. What you're actually editing Google Docs is directing your keystrokes and everything you type towards an off-screen iframe, which is just a completely empty iframe. There's nothing in it, except it's instrumented to listen to your keystrokes and carefully exactly as you take them, listen to them, and then use those keystrokes to update their document model of how they structure your document.

So in Google Docs, when you're editing this, they're like – It's kind of editable, but always empty. There's nothing in it. So like there's just nothing there, right? The actual content is just a memory on then mirrored back to this representation. So something like that we have to, again, it's all about the key events. All about find the right key events. So at Google Doc, think the user are actually inserted the texts themselves.

[00:25:02] JM: You know, I'm kind of curious about how you just manage code in this thing. So this is a lot of JavaScript, it sounds like. Like are you using any frameworks? Are you writing your own frameworks? How are you managing all this code?

[00:25:17] SFR: So yeah, our code base is almost on JavaScript. We have some Python for Selenium tests. But everything else is JavaScript. So yeah, let's talk about how we manage it. So first, our code is split into sort of a few separate apps. So the main ones are the Chrome extension, the actual extension itself. And then we have a separate dashboard project, which if you go to dashboard.blaze today, that's where you create your snippets. That where you edit your snippets, modify user settings, etc.

So let's talk about the dashboard first. So that's the single page app. It's a React app. And we use create web app. It works quite well. Technologies we use to, again, React. We use the MaterialUI for our component library, which we're very, very happy with. We make extensive use of type annotations. So we don't use TypeScript. But we do use JS Doc type annotations. We all use VS code just picks up those annotations and checks the code on the fly, which we find works very, very well to catch a whole host of errors while still keeping our code as regular JavaScript.

Backend, we use Firebase and Google Cloud. So the Google Cloud Storage to store image. We use Firestore as our database. And that works out really great to allow real time collaboration between the users. Firestore, the ability to listen to events in real time on the data –

[00:26:31] JM: Can I ask you a quick question?

[00:26:33] SFR: Sure.

[00:26:33] SFR: Is your Firebase getting too expensive yet?

[00:26:36] SFR: Not yet. We are just coming out of the startup credits they offered. So we're going to be paying our first bill within a few months. It's not too high right now. But definitely that there is some component of lock-in. And what are your thoughts on that?

[00:26:48] JM: So Firebase – I love Firebase. I think they basically built, I don't want to pick favorites here, but the coolest Platform as a Service right now. I mean, that's maybe an overgeneralization. What they've built is like Heroku if it was owned by Google. That's what they evolved into. And so they've really managed it intelligently and moved into all these really interesting adjacencies. Like Firebase Hosting is really good. All the kind of different sugary little high-level services they have are nice. It's just a great high-level cloud and a place to build. The only thing that concerns me about it is their pricing controls effectively. And from what I've seen some Firebase bills, I've seen some very high Firebase bills from things that look like if they were on less sugary infrastructure would be like a third of the cost. I could be wrong. I'm not 100% sure of this. So I don't want to criticize Firebase. Yeah. So I mean, a very big fan of Firebase is what I'll say.

[00:27:51] SFR: You definitely have to understand that the pricing model, and to some extent, you have to build your app around it, or not build your app on Firebase if your app doesn't meet the pricing model, which, of course, is restricting. So as an example that, we were exploring adding a commenting feature to Text Blaze allowing users to comment on their snippets, again, collaboration, allow them to add comment, discuss to snippets, add suggestions, all of that amongst the team. And that was a feature where we had to spend a lot of time and to thinking of like how do we stick within, in this case, Firestore? How do we really sort of stick within the

Firestore limits? Because Firestore is an awesome database. It has that real-time thinking. It's really cool. I can do a fair amount of stuff, but can't do any joints. Like no joins at all. Completely hopeless if you want to do joins. And there're some other limits to it. And also the pricing model you're building document reads or writes. So reads are very expensive, even if you have a very small document. So we spent a lot of time sort of thinking how we're going to build that feature and sort of in some ways building almost around the capabilities of Firestore. So yeah, to your point, like it's fantastic, but it is limiting. And you really have to be conscious of sort of the tech and the pricing model if you sort of use this managed platform.

[00:29:06] JM: And ignoring costs for a moment, what are the advantages of building on Firebase?

[00:29:11] SFR: I don't want to be a shill.

[00:29:13] JM: Please, please be shill, even though you worked at Google Cloud, because you're very non shilly of guy. So you're the best shill.

[00:29:21] SFR: Well, so you get huge sort of rapid development, rapid prototyping, like it is like if you're going to prototype something, like build it on Firebase. Like there are very few better technologies to do it. Like you can move incredibly fast as sort of you're experimenting and you're growing up. They have an auth product that will handle user authentication, user account signup, password resets all of that for you. Firestore, we've talked about like, is this great sort of NoSQL document type database that lets you easily store data, query data, etc. Firebase storage is sort of a frontend on Google Cloud Storage, but with this nice friendly API. And a lot of these services, like Firestore, Google Cloud Storage, they have this what they call rules, Firestore Rules, that basically allow you to do declarative access control. So you basically define this file – It looks kind of like JSON, but it's not all JSON, where you define sort of the access rules for your various resources. And then they manage all the access for you. So you're just building your frontend app. You're defining the backend rules. And then you don't need to have a server at all for most of the stuff you're doing, because the Firestore database is connecting directly from your client app. The rules are validating whether the – With auth, validating whether the user should have access to that resource. And it's just so clean. It's so simple. It's so easy to do. And that's fantastic.

You mentioned hosting. We use that too. We have Cloud Function, Firebase functions, which are layer on top of Google Cloud Functions that provide some additional sort of ease of use sort of more declarative ways of defining functions. So it's just an incredible platform to start building your app. Another big benefit of it as part of Google. So as we're going through this, I mentioned Firebase functions is actually Cloud Functions. Firebase Storage is actually Google Cloud Storage with some additional services. So you have all of Google Cloud services also very cleanly integrated with your apps, with your data, etc. So for instance, we make heavy use of BigQuery. Like BigQuery is, in my opinion, like one of the best Google Cloud technologies. It's really amazing. So we make heavy use of BigQuery to analyze our data and run various services. And there are other parts of Google Cloud that we can pull in once as we move sort of beyond Firebase in places.

[00:31:26] JM: I actually didn't know about that. So you got good integration between Firebase and BigQuery?

[00:31:32] SFR: So no, not Firestore. Unfortunately, it's not that great the direct integration. It's okay. That's not great basically. What you can do is you can do – easily export from Firestore and then import to BigQuery. But there's no sort of federated or real time query of Firestore from within BigQuery. Hopefully, they'll add that in the future. But that's not what we have right now.

[00:31:52] JM: So you did work at Google Cloud for a couple years?

[00:31:56] SFR: Yes.

[00:31:57] JM: I'd like to hear about that. So Google Cloud is probably underrated, kind of has a second mover advantage after – or maybe third mover advantage, you could even say, because they really kind of the game, I guess, kind of behind Azure. What are the selling points of Google Cloud when you pit it against Azure and AWS as an ecosystem?

[00:32:20] SFR: So, full disclosure, I've used AWS a little bit, but I haven't used it too much. So like I've never used Azure. So I don't really want to be like, “Well, Google can’t do this, and AWS can’t,” because I frankly don't know if that's true. I really like Google Cloud because I think the

services are relatively simple, but they're well-developed. It let's do one thing. I just think it's when I look at AWS, I'm like so many services. And some of them are so narrowly targeted at little things. Like Google Cloud that they have like pub/sub, and they have Google Cloud Storage, and they have BigQUery, and these services just work. They work together. It's clean. In my mind is simple and easy to use. So I'm a big fan of using it. But again, like I can't really give you a comparison with AWS or Azure, because I don't have the expertise. I don't want to misspeak about them.

[00:33:05] JM: And so have you gotten to a point where – I guess you've already mentioned BigQuery. Are there any other components of Google Cloud that you've needed to expand beyond Firebase to?

[00:33:18] SFR: Absolutely. Again, BigQuery fantastic. We use it extensively. We use Google Cloud Tasks for task queues for parts of our operations. We make extensive use of the monitoring, auditing, logging services, part of Google Cloud. We're exploring using something like data loss protection for a client to scan their snippet automatically to make sure that they're not including any sensitive information. There're a lot of services we're using at Google Cloud. Just think of top off my head to actually pull up. No, not going to do that right now. But yeah, there's this whole depth one to – And again, BigQuery is the star in my mind. But there is whole depth of additional things that integrate with your services. So for example, data loss prevention, like that can scan a Google Cloud Storage bucket. And since your Firebase Storage bucket has just a Google Cloud Storage bucket in the backend, like it integrates cleanly instantly. I'm happy, again, slightly biased. But I think it's a very great platform. So we're talking about the tech of the app. We haven't talked about Google a little much, as much as you want.

[00:34:15] JM: Yeah. Yeah. I'm curious about the infrastructure. Okay, we can pop-out a little bit and come back to – Okay, so we talked about kind of the code management.

[00:34:24] SFR: Yeah. So there's a couple I wanted to talk about that beyond that. If that's okay, on the code app. We talked a bit about the dashboard. But I just want to touch on the extension, because the needs of the extension are very different.

[00:34:35] JM: It's the Chrome extension.

[00:34:36] SFR: Yes, the Chrome extension.

[00:34:37] JM: Which for people who don't know the runtime is like a sandbox JavaScript environment, right?

[00:34:43] SFR: Yes. So basically, the extension, you can think of it two parts. One is called the background script and one is called the content script. So the background script is sort of a persistent, off screen page. So it's a page that's persistent for sort of the app of the extension, where you do anything a web page can, but the user never sees it. So it's where you can store your data or maintain persistent event. Whatever you want, timers and anything like that. And then there's a content script. And a content script is what's injected into each and every page. And it allows you to respond to user actions. It's what actually does the insertion of the snippet text into the page. So you have these two components. So I think a key part of extensions development is you want to make that content script as small and as light as possible, because it's going to be injected into every single page. So like we make extensive use of npm. We actually use yarn. But we've installed node packages. That are works great. Sometimes not as sensitive enough to sort of bundle size as we should be, right? Because it's so easy to install a package and then get a great feature for user, but it has an impact on the bundle size. And that's not great, but it's okay. Like it's not the end of the world. People are on their powerful laptops loading the dashboard and the snippets. But in the content script, because you're injecting every page, like a large bundle size would have a huge impact on sort of browser performance. So in the contract script, like that is a single file completely hand written. Not a single third party dependency pulled in by NPM. I think we have one or two-third party dependency, but we just version directly in the code rather than using a package manager to make a small package size as we want.

So that's sort of a key thing into developing performance extensions is really like optimizing the content script size-wise, but also if you're like adding an event listener, or anything like that having as little as possible there. So you don't impact the underlying pages, because it's going to be copied across every single tab the user has.

[00:36:37] JM: And can you talk about the kind of dynamics of the usage? So, for example, if I'm a power user of Text Blaze and I define some templates, I've got some customization, take me through what is going on as I am in my day-to-day workflow. Like is Text Blaze sort of like scraping the screen? Does it to have some event listener thing? It's like figuring out what I'm doing? Just give me a sense of what the runtime is doing.

[00:37:07] SFR: Yeah. So, obviously, define these shortcuts. And then you take them into. We have other ways of entering snippets, but if you forget them, but the shortcut is the main one in the preferred way, because it's the most efficient. So to do that, we have a couple of event listeners that are listening for keystrokes. And we keep a short buffer of your most recent keystrokes. You can customize how long that is, but it's on the order of a few seconds. If you type a character that's invalid in a snippet shortcut like a space, we clear that buffer, if you change the focus, or you click somewhere, we clear the buffer. But it's a very short temporary buffer of your recent keystrokes. We then match that locally. So everything's happening locally on your computer. None of this is happening in the cloud. We then match that against the list of your snippets, which we've download via Firestore, and see if any of those snippets should trigger. If one of the matches, we then go through. And we talked about how we do insertion, but we basically bleed-out that shortcut, and we insert the content of the snippet into the pages itself. Again, all this locally. We're not sending what you type off your computer. If you're using our forms product, again, that's happening locally on your computer. What you're typing in there is all purely local. So obviously, privacy security, super important. This tool like this does have a high-level of access. It needs a high-level of access to work correctly. But we want to make sure we operate as secure and private as we can for the user.

[00:38:27] JM: What's the hardest engineering problem that you've had to solve thus far?

[00:38:31] SFR: I talked about the editor stuff. Again, that sounds simple. It's actually super, super hard. Things keep popping up with it. I would say that's the hardest. Other than that, again, just sort of compatibility overall across sites and making sure we have no impact is very significant. Yes, this is hard. This is a hard problem. But it's a hard problem we have full control over. And that's the problem of insert – The dynamic logic in the snippets, like the formulas, the if statements, etc. So the editor problem is a very hard problem that we don't have full control of,

right? Because any site might decide to do something absolutely crazy if their editor. But sort of making the snippets dynamic is something we do have control over.

So let's talk a little bit about that. So again, I mentioned you can write formulas, you can create these if statements, you can create this dynamic logic. So your snippet is highly dynamic. How that works is basically on a formula level, we have a custom parser rewrote using a software called Nearly that looks very good. So we have this formula evaluator that supports mathematic operations. It supports functions. It supports strings. It has a whole variety of capabilities. So when you run your snippet, we go through. We find any dynamic content in it like formulas. We evaluate those. And you can also define temporary variable so you can assign a formula to a variable. So we just go through that with everything. Part of that too, since you have these variables is actually your snippet is less like a program. I'm going to back to more like Excel, right?

In the Excel, it's more of a graph, where you can have this cell depends on that cell, which depends on that cell, which depends on that cell. And you can have this sort of complex set of references. It's the same thing in this snippet, where you don't evaluate the snippet from top to bottom. It could happen in any order. You may have the definition of a form field at the very bottom. That's used higher up in the snippet. So as we evaluate these formulas, we actually have to map out the graph between the different dependencies in the snippet, then evaluate that whole graph ethical.

As we do that, like you can actually hide and show sections. Again, I mentioned you have these if statements. So when you evaluate one variable, that may show an additional part of the snippet, which may define some additional variables or may hide a part of the snippet that will remove variables. So snippet evaluation is actually fairly complex. Again, it's something we have full control over. There're not crazy sites with crazy compatibility issues doing very interesting things. But it is a very difficult problem to execute. At the end, I find it super interesting. It's super rewarding. You can create these super powerful little snippets where you can do a lot of logic in them. Did that make sense? I felt like sort of describing that graph was very unclear.

[00:41:24] JM: I understood it. I hear what you're saying.

[00:41:25] SFR: Okay, great.

[00:41:27] JM: Are there any – Do you run to a lot of performance bottlenecks when you're trying to make this Chrome extension work performantly and make it do like what you need to do on the user's browser?

[00:41:39] SFR: Absolutely. So performance is a huge issue. When you're typing and you type a shortcut, the snippet should insert instantly, like it should be instant. If the snippet takes a 10th of a second, like that's too long. If it takes two times a second, like you're for sure going to notice that and be like, “Maybe you didn’t type beyond it in that time.” Really, we want the snippet to sort of insert in like maybe 50 milliseconds or less. So it feels like I typed slash return, and bam, I have my new text there. So there's a lot going on there like to make that work that that can – And we really want to make it feel fast.

So the content script is the one that is, hey, checking, maintaining the shortcut buffer. But then it has to communicate to the background script, which is maintaining the list of actual snippets. The background snippet has to run through snippets. We enter often via copying and pasting because that's a very compatible way to insert text because most editors will handle copy and pasting very smoothly. So we have to load the snippet content in the clipboard, paste it in. If you're doing dynamic stuff, like we just talked about going on, if all that dynamic stuff, so there's a lot going on.

We've done a fair amount work profiling different parts of the app, making sure they're fast, moving on to another part of the app, profiling that so that there's a lot that goes into it. Yeah, it's never ending, right? Because as you add more features, of course, that can have a negative impact on performance. So it's always sort of, “Hey, does not optimize, but what about this? Can we optimize this more?” The answer is yes. There’s always more to optimize and improve.

[00:43:09] JM: Okay. And zooming out, what does the business look like? What's the kind of revenue profile trajectory, sales strategy? Give me the business perspective.

[00:43:23] SFR: Sure. So we took our first funding a few months ago. That's super exciting, so we're ramping up the team right now. Revenue is – So I don't know if I can give you a number

you ever, but it's solid. It's growing steadily. The sort of business is really about sort of improving the product and making it better, expanding the product to additional sort of screens and use cases. And then also really like seeing how we can build on top of this product.

So right now, we're Chrome extension. We'd like to be everywhere the user is, right? Right now, if you use Chrome, great. We're there. But on your Android or your iPhone, like we're not there, right? Those can be problems. So like one of our top use requests is, hey, make an iPhone or Android or a native Mac or native Windows version. So that's the thing we're exploring doing, one. Obviously, there's going to be a huge cost of that, not in terms of initial development but in terms of maintenance. And like now we have to sync up releases between the Chrome and the Android and iOS. So they're going to be huge for long terms of tax or friction on development, once we make that switch. And at some point, we think we need to branch out to additional platforms.

In terms of sort of expanding to additional use cases, we think they're sort of huge opportunity to use sort of what we developed for sort of additional types of workflows that our users have. For example, the last user survey we sent out to our users, we actually implement it as a text-based snippet. We took our text-based technology, we put it as a hosted website, and then we just send it out to user. So they came to this website. It was a interactive forum where they could fill in things, and they selected things. We revealed or showed different questions, and that was implemented as a snippet, right?

Our snippet is sort of dynamic. It’s so powerful that you can create these really interactive experiences for like, “Hey, maybe this could be useful as sort of like either a form solution or a low code, no code type app builder solution.” So we have a bunch of real estate use, like maybe they could build like a mortgage calculator. Then host it internally or internally via this snippet technology. So the stuff like that are things we're interested in. We have this great technology. We have the viewers who really love the product. Can we sort of build off that and not do things for the sake of doing but like expand to things that really are either technologically or sort of workflow are very adjacent to what we're doing now? So that's something we're very interested in. We're exploring, testing out, etc.

[00:45:40] JM: All right. Well, it's been a pleasure talking to you. Anything else you want to add about the business or the technology?

[00:45:46] SFR: No. It’s been exciting talking to you, Jeff. I like to think we've covered a lot. If I may say, we are hiring. So we're looking for fantastic talent. We're fully remote. If anyone wants to join, please reach out to me, Scott@blaze.today.

[00:45:57] JM: Awesome. Well, thank you very much, Scott. Thank you.