EPISODE 956

[INTRODUCTION]

[00:00:00] JM: The vision of code-free programming has existed for decades. Software engineers have always dreamed of empowering non-technical users with the same creative tools that programmers have access to. For many years, the underlying technology of the web was not powerful enough to make this dream a reality. Platforms such as WordPress, Squarespace and Wix have allowed for some of the functionality of programming without writing code, but the scope of what those tools could accomplish was limited.

Today, the web had caught up. Improvements in browser technology and client devices mean that the end user has access to more powerful technology. The API economy encapsulates large amounts of functionality into cheap, well-defined functionality. No code platforms are finding success among developers and non-developers after persisting for several years as their technology matured.

Bubble is a code-free platform for building websites, startups and internal tools. Emmanuel Straschnov and Joshua Haas are the founders of Bubble and they join the show to tell the story of Bubble and give their perspective on engineering, business and the low code, or no code, or code-free, or code light, whatever you want to call it, the lack of code movement.

The story of Bubble is strikingly similar to that of no code tools, Airtable and Webflow, which we've covered in previous episodes. All of these products have taken years to get to maturity with no shortcuts. Only gritty, difficult engineering problems and performance improvements. Each of these no code platforms has an inspiring story behind them and persistent founders who eventually got their product to success. That's why I love these no code stories.

We're going to be at KubeCon San Diego 2019, Erica and I, the two main members of Software Engineering Daily. We're also going to be at AWS Reinvent Las Vegas and we're planning a meet up at Reinvent on Tuesday, December 3rd. It's not that formal. We don't even have a venue yet. If you have a venue that you can help us with, please send me an email, Jeff@softwareengineeringdaily.com. I don't know how many people will want to come to this

meet up, but there's a link in the show notes if you are going to be at Reinvent and you want to come to our meet up.

With that, let's get on with the episode.

[SPONSOR MESSAGE]

[00:02:52] JM: When you listen to Spotify, or read the New York Times, or order lunch on Grubhub, you get a pretty fantastic online experience. But that's not an easy thing to pull off, because behind-the-scenes, these businesses have to handle millions of visitors. They have to update their inventory or the latest news in an instant and ward off the many scary security threats of the internet.

How do they do it? They use Fastly. Fastly is an edge cloud platform that powers today's best brands so that their websites and apps are faster, safer and way more scalable. Whether you need to stream live events, handle Black Friday traffic, or simply provide a safe, reliable experience, Fastly can help.

Take it for a spin and try it for free for visiting fastly.com/sedaily. Everybody needs a cloud platform to help you scale your company. Everybody needs a CDN. Check it out by visiting fastly.com/sedaily.

[INTERVIEW]

[00:04:07] JM: Emmanuel and Josh, welcome to Software Engineering Daily.

[00:04:08] ES: Hey! Thanks.

[00:04:10] JM: You guys are working on Bubble. Bubble is described a code-free programming language. Explain what that means.

[00:04:16] ES: What that means is that we've basically created a virtual interface that lets you describe what you want to build without running code, and what you can build today is basically

web applications. Pretty much, any website that people would be familiar with, like Twitter, Airbnb, these kinds of websites can be built on our platform without writing code. It's still programming, but it's without code.

[00:04:37] JM: Describe that experience in more detail. How does somebody build an app on Bubble?

[00:04:42] ES: You would start usually with a design aspect of it. Basically, by dragging and dropping elements on the page, on like an empty canvas as a background of your application. It's a little bit like Visual Basic back in the days. The way you program it is by defining what we call workflows, which are going to be basically when the user clicks on this button, do this, do this, do that. This, this and that being creating something in a database, charging a credit card, sending an email.

If you look at most of these websites that people are very familiar with, once again, you can usually build these websites with those elementary actions. So what we've done is basically prebuilt those elementary blocks that people assemble together.

[00:05:22] JM: The field of low code or code-free, it's been around for a while at this point. People have been talking about it, working with these platform for about maybe 8 years, maybe 10 years. It's gradually found more and more transaction, but it's still pretty far under the radar. Why is that? Why is it taking so long for people to adapt to this technology?

[00:05:46] JH: I think there are a couple of factors. One is it really started a lot of it in the enterprise space first, like a lot of the early big players were in enterprise, and enterprise software doesn't like spread in the same kind of like viral mindshare kind of way.

The other thing is just the technology hasn't really been mature, because building a platform that can completely replace programming is a huge investment. We spend basically our first five years of existence as a company just like heads down bootstrapping building the product, and you kind of need that deep upfront tech investment to have a platform that's usable enough. I think a lot of it is like there're just players emerging today that are finally good enough to be interesting to the broader market.

[00:06:36] ES: One thing I'll add to that is the level of skepticism is fairly high, because it's been tried for many, many years. You don't want to go to market too soon with a half-baked solution, otherwise it kind of reinforces the skepticism of everyone. That's why you want to take your time to do it.

[00:06:52] JM: Was there an inflection point where the technology got good enough? I don't know. Maybe React or the V8 getting fast enough, smartphones getting good enough. Do you guys remember any specific inflection point?

[00:07:09] JH: Yeah. I mean, I think Amazon Web Services honestly has been huge, and they've been around for a while, but really that kind if enabling technology has made a pretty big difference. I think you're absolutely right about V8 getting faster, because most of the solutions are very JavaScript intensive. We certainly use a ton of JavaScript and having a fast engine is really important here.

But I would say it's more a market maturity thing than a technology maturity. A lot of the early internet boom was like, "Hey, if you get a team of smart engineers, you can build a winning product in the space." Then it takes few years for that to go from, "Okay, and now let's go horizontal with this. Let's go global. Let's take niches that like are smaller, but still need a good softer product," and that's really where no code gets powerful by like sort of taking this thing that right now is only available in tech hubs and kind of spreading it to the rest of the world. I think that's the real driver is my guess personally.

[00:08:19] JM: What I think is profound about this low code stuff is that there is this niche of developer, entrepreneur kind of person who is somewhere between programmer and designers and product managers and just those like plucky hacker dudes that you see at Startup Grind that like maybe they don't know how to do anything except send a mail merge, but they're so agro and they're so persistent that it's like you want that person on your team.

Low code is like some type of each of those people, somewhere in the Venn diagram. I want to understand who's the prototypical user that is picking this thing up and trying to build a product around it?

[00:09:12] ES: I mean, that's pretty much what you're describing. In terms of demographics, it's a little bit more diverse. The age range of our users is actually pretty interesting. It goes literally from like high school students to people that are retired many, many years. We have some users that reached out recently, they were like 72. We have a lot of people that used to play a lot with FileMaker Pro 20 years ago or Microsoft Taxes users today. But the core of our business here is like non-technical, but tech-savvy startup people. People much like you described all the girls that goes to Startup Grind.

One thing I would say that you say it's a niche. I'm not sure it fits that niche actually. It's a pretty large crowd. I would say for one person that can write good production code, you probably have like 10 or like 50 people that would go to those events that are like not technical savvy, and those are the people we talk to.

[00:10:03] JM: The other thing that I think is cool is I think we're starting to see pretty well-defined patterns for building your first version of a product on a low code or a code free platform. It's not like the platforms try to restrict you from moving off of it incrementally. There are patterns for moving off of it incrementally, for exporting bits of functionality. Have you seen any examples where people have built their MVP on Bubble? Gone to market and then broken out their code into like a Node.js app and a React app and kind of liberated themselves?

[00:10:47] JH: Yeah. We definitely see some customers go to market and then they'll do something like Bubble connects to APIs. They'll build an API server, move like some of their custom-like computational logic on to that and then use Bubble as the frontend and their custom server as the backend.

I mean, our goal is we want to keep them on Bubble just because – Not because we want to keep them, but because we want Bubble to be able to do the kinds of things that you need to build it for. But like realistically, there's always going to be use cases that are like one step ahead of whatever the current state of the art is. We definitely see people like using like APIs, using – We also let people extend Bubble with JavaScript. So using stuff like that to basically grow beyond like a prototype Bubble app.

[00:11:38] ES: Yeah, that final part is very important. Our goal is that people never migrate off Bubble, but extend the platform when needed. That's why I'm not necessarily a fan of this no code name. I mean, it's good because people are okay with it, but Bubble is mostly no code. But if you're a coder, you might have a lot of fun on Bubble writing JavaScript plugins that are like completely built with code.

But the reason why – I mean, the fact that people built an MVP on Bubble because it's faster and more efficient to get to market still holds when you're a much bigger company. So we just need the platform to be good enough to be able to get it to these people.

[00:12:11] JM: Now, as a developer, in some ways it's appealing to me, the idea of starting on a Bubble kind of platform, building on my MVP and then doing a gradual migration. But there's something about it that makes me resistant to taking that pattern. There's something about it that makes me – Even though it would probably be more code. I would rather just design all the React components myself. I'd rather just do it on myself, something in my gut. Maybe I've been overly-trained to build my own HTML.

[00:12:48] ES: Yeah, that's probably what it is.

[00:12:51] JM: You think so? Is there something rational –

[00:12:53] ES: I think it's a little bit of a cultural experience thing. If you're using React, you depend on React. React, it's open source. So you can go in the code if you need to. But in practice, you won't. We're just another framework out there. I think the more good success stories we have on top of us how comfortable you will. One of the reason why we've been doing pretty well is because we haven't been talking to software engineers. They don't have the same concerns as you have.

We want to convert more engineers to users, but our experience that it's actually like your good example, harder sale to an engineer, because an engineer – When you go with something like Bubble to someone, if it's a technical person, they're going to see limitations and worries first. If it's a non-technical person, they're going to see impairment first, incapabilities, because the alternative is not to have anything.

Currently, the big part of audience is non-technical and we don't have that challenge. With engineers, it still happens. I would say today is much better than like 5 years ago, but we still have a long way to go. But I'm hoping that at the end of this conversation, we'll have convinced you that your next web app should be built on Bubble.

[00:13:56] JM: Well, I don't even program anything anymore. I just do podcasts. But take me inside your empathy machine. How do you figure out how non-programmers think? If those are the people that you're going after – Are you guys both software engineers?

[00:14:11] JH: Yeah. So you're getting to the heart of the answer. Emmanuel codes, but –

[00:14:17] ES: I used to code a ton in high school, and for 10 years I actually was in management consulting that I went to business school. I arrived basically as a non-technical guy in the team. But then on the job I learned how to code again, because I realized pretty quickly that if I didn't code, there wouldn't have been much to do given like how much of a product company we are. But I try to keep – A lot of the product, like customer-facing product was actually built by me, and I try to keep that brain both like technical brain and non-technical person.

One thing I would say though is, again, there are a ton of people that are not technical, but they can use Microsoft Excel, which is a pretty complicated interface. Those people, they're non-technical, but you know how they think if you know how to use Excel. That's kind of what we modeled our interface to. I mean, it doesn't look like Excel, but the same logical steps and it's about knowing where to click and what to do. If you use yourself these products, even if you're technical, you can do a lot of smart things.

[00:15:09] JM: But what about when you're talking to the 72-year-old person or the high schooler who is not a programmer? Do you learn anything that's surprising to you about what is intuitive or what is not intuitive to them?

[00:15:23] ES: Yeah. I mean, we've very community-drive. So the feedback loop is very fast. So when something doesn't, we tend to hear about it within hours after we push them, to be honest.

One thing I'd say though is that there aren't that many difference between someone who's 72 or 16-years-old, because most people – I mean, 72, not everyone. Most people are fairly digitally native, which means they use to using the same website or the same product. So that created like a common bar for everyone when using a computer.

IPhoen is kind of the wrong example, because I think Apple products tend to simplify interfaces so much. I think Microsoft products are better examples of a complicated product that still empowers you to do a lot of things. Microsoft stuff is the most used software in the world, I think. So that bar is fairly consistent across ages.

[00:16:08] JM: You guys started Bubble in 2012. What was the original spec for the product?

[00:16:13] JH: As in like the original design vision or –

[00:16:16] JM: Yeah. What was the goal?

[00:16:18] JH: Pretty much the same as today actually. If you showed our 2012 selves what we've built, we'd be like, "Oh! Yeah. That's kind of what we're working on." Our first product true north was can you build Airbnb in Kickstarter? I actually remember testing elements of their design and seeing if we could like replicate their homepage just to make sure like our design engine could actually product a real like startups like webpage looks like basically?

[00:16:46] ES: The initial idea was to really empower non-technical people to start companies, because one thing is a technical vision that you have. The other thing is why do you do this. The reason why we're doing this is because we felt too many projects were just not happening because of the shortage either of engineers or a funding, which is essentially the same thing at the end of the day and we felt it was just very sad that a lot of people have good ideas and to just going to try them.

[00:17:11] JM: It's interesting. 2012, I'm trying to think back to what was in the water at that point. I guess 2008 was around the time when Airbnb got started. Four years in. It was really taking off. Startup fever was trying to brew.

[00:17:28] ES: Yeah. Facebook went public in 2012.

[00:17:31] JM: Right. Okay.

[00:17:32] ES: The social network was a hot movie, which I'm sure had a lot of impact on people.

[00:17:37] JM: I think it really did.

[00:17:38] ES: Yeah, it definitely did.

[00:17:39] JM: Then Stripe was – I guess Stripe –

[00:17:43] ES: Just started.

[00:17:43] JM: Yeah. Started to get interaction shortly after that. It's kind of been up into the right since then, right? People just wanted to be more and more startups. How has the pursuit of starting a company through starting with a low code application? How has that pattern advance or has it become more popular? Do you still feel like it's kind of like under the radar is a way to get started?

[00:18:06] JH: It's starting to break into the mainstream. I think we are on the threshold. Even like the last six months, I think the dialogue around it has changed. It's starting – We're talking to like various accelerators. Like everyone at Y Combinator knows Bubble is a thing, for instance. It's starting to like be in the consciousness that this is a tool in the toolkit.

There're still a lot of the use no code tools for prototyping and then switch to code. That's still a big thing people think. But we're starting to see people build prototype or an MVP and then they launch and it's like, "Well, are you going to rebuilt it? Are you going to get customers and iterate?" You have to make that tradeoff, and people are increasingly making the tradeoff to just, well, go a little bit further before moving off. We just want to like keep on pushing that limit point out and out and out basically.

[00:19:02] JM: The old thing was you'd have a landing page, which is like the email address or like –

[00:19:09] ES: LaunchTrack.

[00:19:09] JM: Or like [inaudible 00:19:11]. What did you say?

[00:19:12] ES: LaunchTrack.

[00:19:13] JM: LaunchTrack. I don't know what that is.

[00:19:14] ES: That was a company that was just doing that.

[00:19:16] JM: Oh! Just landing page.

[00:19:17] JH: It was like 2012. LaunchTrack was just like huge, I think.

[00:19:20] JM: Okay, or there was like the flow where it's like you get started and you click and it's like, "Oh! It's a Typeform."

[00:19:26] JH: Yeah.

[00:19:27] JM: Oh! Your startup is a Typeform. That worked for some people. Not anymore, I don't think. I don't think people go for that as much – But kind of like, "I'd drop off."

[00:19:36] JH: Yeah. That's kind of like part of the pattern. That was like step one. The very basic idea of validation before starting coding, then it got to like, "All right, build something a little bit more sophisticated." The WordPress blog built a Squarespace page, and then I think the new generation of no code tools are just taking it a step further, because at the end of the day, where the whole philosophy of that landing page came from was like the lean startup iterate idea. Don't invest engineering effort building something until you know it works. Anything that like reduces the cost of testing things out and iterating is just going to get adapted basically.

[00:20:21] JM: From 2012 to 2019 is a lot of time in the world of technology. I can imagine there has been a lot of refactoring in the lifetime of Bubble. I can imagine there have been some very, very painful sprints where you realized –

[00:20:40] JH: There were.

[00:20:40] JM: All right. We're moving off a backbone. We're going to X. Then maybe we're going from X to React. How many times have you had to refactor the frontend?

[00:20:51] JH: We did something, which some mornings I wake up and think this was brilliant. Some mornings I wake up and think this was terrible. I'm just still on the fence. We basically built around JavaScript framework because right around the time we were starting was like JavaScript frameworks were a thing, but they weren't mature. It was not clear who the winner was going to be. It was kind of like chaos. It wasn't like today where React is like the dominant player. So we rolled our own.

It's super tightly integrated with our overall programming language. So like we have this like really tight coupling between it and our core technology. Every time we bring on a new engineer, we have a learning curve problem, because they're like, "What is this? It doesn't look like what they've been doing at their previous job. But on the other hand, because it's ours, we've evolved it with the need of the company and we haven't had to like rip it out and go from like the latest framework to the next one to the next one.

[00:21:49] JM: When you say you have – Bubble is a programming language, is it actually a programming language or is it just like you're talking about the visual interface?

[00:21:58] JH: I consider it a real programing language. I mean, it's Turing complete. You could theoretically build any algorithm. I wouldn't necessarily recommend it. It's designed for a specific use case and expressing like an arbitrary piece of code in it could get a little hair, but you could. Even though there's no code, it's definitely programming. It flexes the same logical muscles. It flexes the same conceptual thinking.

[00:22:29] ES: To give another definition of what programming is, you can say it's a Turing complete system or you can say can you have bugs? A lot of no code tools out there, you can't have bugs. If you build a website on those, I don't know, Webflow, or tools like that, you can't go wrong. I mean, you can do as many mistake, but the website is still going to be beautiful into what it's expected to do.

On Bubble, it's very easy to have bugs. When you build your workflows, if you mess the order of the actions or something like this, you don't going to have the expected behaviors. Those bugs are not going to be on our code side. It's going to be on your application side, which is why one of the most important features we have is actually a debugger, because our users need to be able to debug the application. Because you can have bugs, it is programming. Whether you type on a keyboard or you use your mouse and a visual interface, it's almost a detail.

[SPONSOR MESSAGE]

[00:23:23] JM: Feature flagging makes it easy for your team to quickly change the way that your product works, and CloudBees Rollout lets you manage feature flags easily. When you have a solution to manage feature flags at scale, you're empowered to continuously and intelligently rollout changes as soon as they are code complete on any platform, even mobile. You can decouple development from code release for a real-time change control. You can rollback only the changes that you don't want, or keep them around.

You can toggle features. You can use multi-varied flags for A-B testing and you can remove misbehaving features with a kill switch. All of these is part of the feature flag platform that is CloudBeess Rollout.

Visit softwareengineeringdaily.com/cloudbees and try a free 14-day trial and experience how CloudBees Rollout can help you with every release. Visit softwareengineeringdaily.com/cloudbees to get a free trial. Cloudbees Rollout is trusted by large users, such as Zekdesk and Jet.com. Try it out today at softwareengineeringdaily.com/cloudbees.

[INTERVIEW CONTINUED]

[00:24:47] JM: Since you've mentioned WebFlow, I think generational one of the side builders as Squarespace, Wix. Oh! I guess generation one is like WordPress. Generation 1.5 is Wix, Squarespace, Weebly, whatever. A bunch of these things. I guess Shopify maybe is like 1.75, generation 1.75. Then generation 2.0 is like WebFlow, Bubble, Airtable maybe, Retool maybe. These kinds of things.

The thing about both of these generations is I think they get earmarked as competitive, but it's more like – This is like everybody in the world should have one of these at last. Your blog should be one of these, or your personal website, or your personal business. Literally, everybody in the world should have a website, and probably they should not be editing HTML. So they should be using WordPress, or Weebly, or Wix, or whatever. Depending on their level of expertise.

But since you mentioned WebFlow specifically, I think this is the product that Bubble gets compared to the most. How do you see the market break down between WebFlow and Bubble?

[00:26:06] ES: I mean, currently, I think we'll see how it plays out in the future. But we're talking two fairly different people. Actually, even though people compare, it's like my understand of WebFlow is that they actually kind of much more to designers, originally. I think they've been very strong with freelancers and agencies.

We go more for like business people, product managers, marketers, that want to create functionality over design. On WebFlow, it's probably design over functionality, which is why by the way when you look at the limitations of both products – I mean, WebFlow is a fantastic tool to create beautiful interfaces. On the backend, it's fairly limited.

We are great on the backend. You can do pretty much anything you want. At the frontend, you can do a lot, but to be honest, you have to know what you're doing to make something pretty. It's pretty easy to do something that is not pretty if you don't pay attention to what you do because we have different focuses.

Another reason why I'm not a huge fan of the no code thing is when you say no code, people think it's all these products are competing with other. But actually I don't think they are. Airtable is for one specific need that is completely different from WebFlow, for instance. Bubble in the

middle. Retool is for big companies doing retool, doing like [inaudible 00:27:12] queries visually and stuff like this. Each of those products actually have very different need – Feel a different need to a different market, and also this market is actually huge. We're not really competing with each other, I think.

[00:27:25] JM: That's a good point, because I think it's describing a category by what that category is lacking. It doesn't really make any sense.

[00:27:32] ES: Yeah. If no code means there is node. I mean –

[00:27:35] JM: Yes. It's my water bottle.

[00:27:37] ES: Facebook is no code.

[00:27:39] JM: Facebook is no code.

[00:27:40] ES: Well, that's a media thing.

[00:27:43] JM: Yeah. This is a no code podcast.

[00:27:45] ES: I prefer talking about visual programming, because if you look at that stuff form that angle, then it's – First of all, not many players exist in that space. It's a little bit narrow, but it also explains exactly what it is.

[00:27:57] JM: Right. Yeah. You have something like PowerPoint. PowerPiont you could argue is visual programming, but not really. Maybe? Who cares?

[00:28:05] ES: Some people now put Codeout.

[00:28:08] JM: What is that?

[00:28:08] ES: Codeout.io.

[00:28:09] JM: I've heard about that. I haven't listed that.

[00:28:12] ES: It's a little bit – It's closer to Airtable. It's like a more interactive Google Slide Google document, and people put them into no code space as well. Again, the no code definition is a little bit too broad.

[00:28:24] JM: Man! I bet the kids who are doing Minecraft right now, these are going to be the kids who like totally dominate these platforms, these low code or no code or code-free things.

Getting back to the engineering side of the thing, you said the backend is more programmable in contrast to WebFlow perhaps. What does that mean? I guess I could imagine Airbnb, for example. When Airbnb got started, it's basically like a CRUD app and all their hard work is basically how do we build this market. All their hard work is kind of offline. But overtime it's become like a data platform. They have tons of data work to do. They have built just gigantic search index and they have to do information retrieval and recommendations and all kinds of things.

You can imagine, there was this period of time where they were going from this go-to-market struggle to building out a data platform. If I was doing that with my Bubble-based version of Airbnb, is the backend going to be flexible enough to offer me all the data infrastructure needs that I have?

[00:29:27] JH: Yes. So probably not to the Airbnb scale today. What you can do is you can push data to third-party machine learning APIs. Google's released some stuff. Amazon's released some stuff. Sort of outsource the like really difficult number crunching to like the specialist services then pull the data back into Bubble and sort of process it.

We definitely are more geared towards that early day Airbnb more of a CRUD platform, but we try and make ourselves as extensible as possible to let you sort of evolve in that direction. We hope to like continue – We want to keep on adding to our functionality. We'd love to reach a point where it's one button click and we tell you how to like process all the data in your database. Build an in-house recommendation system that you can install easily. Today, you do

probably have to work with external services once you get to Airbnb scale. But I see this evolving over the long term.

[00:30:33] JM: There's a distinctive demo experience on Bubble. If I go on Bubble and I scroll down the page, I can do a one click try out the editor and it takes me into actually a version of your homepage. Your homepage is actually built in Bubble, which is pretty cool. Just the idea that the user who is on your landing page is really experiencing the kind of thing that you can build with Bubble. It's sort of a meta-experience. You can actually kind of imply that, "Look. You too can build your own startup on Bubble."

When I click into that experience and the editor is loading. The editor is going to load the editable bubble.com, or what is it? Bubble.io experience. There's some startup, which is totally understand. This is a very complex application. I would expect nothing faster. But I want to know what's going on. When I click on instantiate my own editable version of this website, what's going on in that startup time when that editor is loading?

[00:31:41] JH: Yeah. So we have the homepage as this like file, this basically giant, several megabyte JSON document, and when you navigate to it, we create a copy of it. Put it into a database and then fetch it out of the database, send it to your web browser and then draw it on the screen in real-time, basically.

[00:32:05] ES: We do that for each person, because we don't want more than one person to modify the same version of what we call the private version of the homepage. By the way, it's not just a homepage. Our entire application is built at Bubble, like account management, subscriptions, pricing, marketplace. Everything is built on Bubble actually. It's only the editor that's native.

[00:32:28] JM: Right. But when that gigantic blob of JavaScript is being loaded, what's going on there? What's the JSON blob de-compilation thing?

[00:32:40] JH: Yeah. I mean, it's mostly just moving the data around, honestly. Moving it to the database. We have this in-house built database system for managing these applications. We need to copy it into there then sending over the wire to your laptop.

We have the whole thing loaded on your computer. Bubble is kind of the interpreter. The editor basically runs the application, renders it in front of you. Sort of walks through this whole document. Says like, “Hey, there’s a button here. So let’s draw a button here.” “Hey, there’s like a workflow here. So create the workflow on the workflow tab.” It’s basically just like walking through the entire thing.

[00:33:24] JM: What cloud providers do you use?

[00:33:26] JH: Amazon Web Services.

[00:33:27] JM: Completely?

[00:33:29] JH: We use some third-party. We use like LIbrato for monitoring. We use SendGrid for sending emails. Cloudflare, yup.

[00:33:38] ES: But it’s mostly AWS.

[00:33:41] JM: What have been the AWS services that have surprised you in their usefulness?

[00:33:45] JH: We stick to the basics. We use EC2. We use RDS. We use their Redis ElasticCache Hosting. We use S3.

[00:33:55] ES: Lambda a little bit.

[00:33:56] JH: Lambda. Yeah, Lambda is pretty cool. Those are the main ones, honestly. We’re kind of like plain close the bare metal in terms of technology, because we’re building a system. So we end up having to do a lot of low-level stuff sometimes. We keep things kind of simple on the tech stuff.

[00:34:13] ES: To be fair, it’s not been true all the time. We kind of gotten more and more bare-boned. On the database side, we started with what? Cloudant and Firebase and Elasticsearch and then PostgreS, which I think was a good evolution.

[00:34:27] JM: What was the first one?

[00:34:28] ES: Cloudant, I think.

[00:34:29] JM: Cloudant.

[00:34:30] ES: Yeah,

[00:34:30] JH: Yeah IBM.

[00:34:33] ES: Then Firebase a year after that.

[00:34:36] JM: Right, because Firebase –

[00:34:39] ES: Before they got acquired.

[00:34:41] JM: Okay. So what? 2013 or something, 2014? I use Firebase a lot today, but I guess Firebase is kind of too expensive if you're scaling, right?

[00:34:50] ES: I mean, to be honest. Back then, it was not a price issue. It was more reliability issue.

[00:34:54] JH: And more our fault than Firebase's fault. We were doing things to Firebase that like the developers clearly have never anticipated. We hit like the maximum JSON nesting depth. We were writing like millions of records to a single node. We're doing all kinds of crazy shit with it.

[00:35:12] JM: Because they were the "real-time database" and you're like, "Okay. We're building real-time. We'll use that."

[00:35:21] ES: That is exactly what happened.

[00:35:21] JH: That is 100% what happened.

[00:35:24] JM: Wow! Then Elasticsearch?

[00:35:28] JH: Yeah. Basically because we wanted a lot – We're doing a lot of complicated like search and like filtering and like querying functionality for our users, and Elasticsearch gave us a bunch of that out of the box because it just ships with it. But Elasticsearch isn't like great as a primary data store for like data reliability issues and we ran into like some scaling issues, because we do some weird stuff with like – Every single one of our users has their own unique schema, which causes like all kinds of havoc.

What we eventually decided was like, yeah, Elasticsearch buys us a ton of stuff, but we need something a little bit more flexible and low-level. So we eventually switched to PostgreS. Went for the sort of conservative solution, but it's been a pretty good tool for us.

[00:36:17] JM: I'm just trying to imagine, because for those who haven't seen Bubble. It's literally like – It's kind of like a Wiziwig editor. Drag and drop your different website components. Obviously, the model is represented in your browser and it's getting kind of replicated to the Bubble backend so that you have this synchronized model of the website application that you're building. I'm trying to understand how Elasticsearch would work as a backend for that system.

[00:36:51] ES: That was not for to store the application object. That was to store the user's application – The applications data. If you create Airbnb, that's where you would store the apartments.

[00:37:03] JM: Oh! Okay.

[00:37:04] ES: Which is actually one of the more challenging issues, because that's where each application has a different schema. We know what an application looks like, because we define that ourselves. What we don't know is the different types a user is going to define, like the Bubble builder, is going to define all the different types to describe the data of this application and we need to be able to – We saw a lot of crazy shit and we need to be able to handle a lot of that.

[00:37:26] JM: Interesting. Because I would imagine that Bubble is mostly used for like fairly simple CRUD applications and it wouldn't –

[00:37:33] ES: No.

[00:37:33] JM: No? Well, give me something complex. What's the most complex app somebody has built on Bubble?

[00:37:39] ES: The most complex app is dividend here. It's a fintech company. It's actually fairly large now and they're building – It's a fintech business. Basically, financial company selling loans to people that want to install a solar panel on their roof. They're a bit fairly complicated web platform where homeowners can apply to get approved for a loan. Installers can see all the different people they have in their portfolio and report the – Who has been approved for the loan, first of all, then report the success, the progress of the – Actually, installing the solar panel on the roof, and installers can put money and get returns.

This platform is fairly complicated where you can have a lot of different situations. It is CRUD. I mean, at the end of the day, all applications are just about finding data and modifying it. But the different flows, the different flows where the data can go through and the different conditions that apply to each different object is safe, like in the hundreds or thousands. They build that entire in Bubble. [inaudible 00:38:30] it's pretty large. I think they process more than a billion dollars of loans actually through a Bubble build software.

Another example I would give, which is not necessarily the most complicated, but the craziest thing, is when you create a tool that's extremely open-ended like Bubble, you basically empower them to do crazy things, because we can add complexity just by adding a few clicks. We've seen some people with thousands of the fields on a data type like doing all kind of like the most crazy way to describe the data in a way that was kind of working for them. We have to enter that. In many ways, I would say we had more trouble making sure it works well for them than for this business that process a billion dollars of loans.

[00:39:07] JM: Right. I mean, I've seen some crazy spreadsheets. I've seen crazy spreadsheets.

[00:39:12] ES: Excatly. Right. Yeah, Excel is a perfect example of you create an open-ended tool, look at the result.

[00:39:17] JM: Right.

[00:39:18] ES: That's what cool about Bubble. That's why I love my job.

[00:39:22] JM: I'm with you. I mean, that's one of the things that's very motivational about working in software, is it is this creative pursuit where in many cases you're building something that allows other people to be creative on top of it. I mean, that's exactly what Bubble is.

Dividend? That's the name of the company that's been really successful and it's building on Bubble completely still, and they haven't migrated off of Bubble.

[00:39:44] ES: Actually, they are. They told us a year and a half ago, because they got acquired and when they got acquired, it wasn't an acqui-hire. It was an acquisition, the acquiring entity was like, "Okay. What is this? We need to move something more traditional." A year a half ago, they told us they wouldn't move off. Honestly, they still haven't.

Because what's happening is it takes so much time. That's where I think no code is going to win eventually, because it's so much faster and cheaper to build on top of a no code platform, but it's very hard for their engineering team to catch up, because what's happening is – They're still operating. They keep modifying the product and improving the product. They're building on top of Bubble. So the pace to improve the product is faster than the engineers trying to catch up. Are they moving off? Yes. When? Honestly, I don't know, and they don't know either.

[00:40:35] JM: If they're doing data science, whatever, everybody does ETL anyway. The complex backend stuff is all like data science. Run an ETL job. I mean, I imagine you can just export all the data from PostgreS and then just put it into Spark or whatever and you got your platform. What do you even need to migrate for?

[00:40:56] JH: Yeah. That's kind of how we see the universe.

[00:40:59] ES: That's our thesis.

[00:41:00] JH: Yup.

[00:41:02] JM: It almost kind of seems like M&A would be easier also. It's all like, "Hey! Here's our app. It's in Bubble." That's all there is to it. Your diligence process if literally like we're on this platform. We don't know how it works. It's proprietary, but at least it's not like a nest of AWS services and external APIs. It's like fairly easy to understand.

What happens when an application scales on Bubble?

[00:41:30] JH: Yeah. Our model is basically we give users more server capacity as they get bigger and as they like upgrade. Today, what we do for our biggest users is we eventually move them to their own like private server cluster. We would actually like to eventually not have to do that and have like our core system be able to like scale up even to like our biggest customers. But what we do today is basically a private Bubble instance that just like grows with the customer's data.

[00:42:05] JM: Have there been any particularly hard scalability problems to solve either in terms of your own scalability problems or customer scalability problems?

[00:42:15] JH: I mean, all the time. This is basically what like 70% of our engineering work is because people are writing queries, and they don't have a mental model nor should they need to have a mental model of how the query is being executed. So it's kind of like the problem that like a SQL database designer, like someone building a SQL database has to solve. How do you execute? How do you build a query planner and execute the expression? But at least for like building a SQL database, you expect the end user to like sort of be a programmer. You tell them like create an index if you need to accelerate that.

We have to guess. We actually like behind-the-scenes will build indexes on-the-fly based on like usage patterns and sort of try and guess what queries need to be optimized. We also have like a query rewrite engine where we take a query. Sort of analyze its structure. See if it's like mathematically equivalent to a simpler version of the query and try and convert it. We do like a ton of stuff on basically scaling a user application. That's like our basically our biggest technical problem.

[00:43:26] JM: Wow!

[00:43:27] ES: The good news is that you solved the problem for one application, you solve it for most people. That's where we started to be an interesting place. I think, to-date, we had more than 300,000 apps created on Bubble, or 250 or something. All these applications, you can see different patterns looking at large scale. Then when we solve it for someone, we prove it for everybody else. It's not an endless race, otherwise that wouldn't be possible.

[00:43:52] JM: 300,000 paying customers?

[00:43:54] ES: No. I wish. Our free plan is very generous. We have a ton of people that just use our free plan.

[00:44:02] JM: What I've heard about Heroku, Heroku has their famous free plan. I've heard that it gets abused basically where – I mean, you can set up your – You can deploy your node application to Heroku and it will fall asleep, which means that basically if nobody accesses it for a while, it's taken offline essentially. Next time you need to summon it. You have a cold start problem, because the application actually has to get loaded. Do you do something like that for the free applications?

[00:44:36] JH: No. We actually invested a lot in building technology that lets us share infrastructure at a more granular level so that the cold starts are like measured in like single digit milliseconds rather than spinning up a whole container.

We do things like our node servers have this co-routine implementation that lets us basically jump between different user apps. We like run an app for like a millisecond or two then switch to

another app, then switch to another app within the same node process. It's a little hairy, kind of cool. Fairly proud of it and definitely –

[00:45:20] JM: What's the buzz factor on that part of the application?

[00:45:23] JH: It's a little worse than I would like it to be. Let's put it like that.

[00:45:28] JM: You're the only person that knows how to use that part.

[00:45:31] JH: Not quite that, but it definitely like I'm the only person who really understands it end-to-end. Then I think there're other people in the team who could probably wait and then debug it if they have to.

[00:45:44] JM: I think that's the only way to build this kind of application, to build something like Bubble. I actually interviewed Vlad from WebFlow a few weeks ago and I'm sure you guys have gone through a period like this, but there's some period where it's like 8 months when he was building WebFlow with, I guess, his brother and their other cofounder.

Where, they were just like totally in the weeds, solving these really, really tough technical problems, because the whole low code thing, as you said, this really has to be something where you're like, "We're going to do this. It's going to take us forever. We're going to do it. We decided. We're going to ride the technological wave until our efforts converge with the state-of-the-art." Until that time, you're just solving really, really hairy technical problems.

[00:46:31] JH: Yeah. I mean, I definitely feel their pain. We have done a lot of that. When we bring on new engineers, the first four months on the job, honestly, are pretty rough. They basically have to absorb years and years and years' worth of deep technical investment in a codebase written by a couple of people moving fast, and it's tough. But we're really starting to see like the team taking on a life on its own and like making the systems their own, and that's like so inspiring.

I mean, both of Emmanuel and I do dive in and write code like we are not out of the day-to-day coding yet, but we're getting to the point where it's a lot more management and working with engineers than us writing code ourselves, and that's like a really cool transition.

[00:47:23] JM: That must be gratifying. How many people on your team at this point?

[00:47:27] ES: Two people started this week. So I recall 17, I think, including the two of us.

[00:47:30] JM: 17. Okay. What makes Bubble kind of an interesting story is the fact that you were basically bootstrapped – People keep telling me that bootstrap is a bad word now, like self-funded is the like way –

[00:47:44] ES: Client-funded.

[00:47:45] JM: What is it?

[00:47:47] ES: Client-funded.

[00:47:47] JM: Client-funded. Right, or self-funded. But like you raised a seed round 7 years later. Why?

[00:47:56] JH: Yeah. It's more of a question of like why not originally, because we always wanted to be big. I think what we're trying to do to be successful, we're building a massive platform and eventually you have to get to a really large scale. It is the kind of business that VC money makes sense with.

The reason we waited so long is once you get on that VC treadmill of like producing numbers to justify the raise, to invest and produce new numbers just for the raise, you have to be ready to take off. Our initial strategy was like the opposite of that, like our initial strategy was work with like 5 customers, then like 20 customers, like really investing in making those customers successful to like sort of prove out that like this tool would work for people. We just weren't ready for the VC treadmill. We actually saw this happen to a competitor that started around the same time as us. They took funding.

[00:48:56] ES: Did YC with the traditional Silicon Valley thing and that did not did well.

[00:49:02] JH: Very smart team, very competent engineers. Shipped a product that was like good, but not really primetime ready, and you just can't grow on that kind of product fast enough.

[00:49:14] ES: Especially, again, for startups like us, like in the no code space. In the first years of you building your product, users always ask you still today, the most common email we get from new users is, "Hey, I can build this? Can I build that?" Until you're in a position where you tell them yes most of the time, you don't want to many users, because it's a waste of time to tell them no and it's a waste of opportunity, but the likelihoods they'll come back is fairly low.

If you take money too soon, external funding too soon from VCs, you're going to try to get more users to go to your website, because at the end of the day, that's what investors are going to care about. Then you're going to tell them no, and it's going to be not beneficial for the business at all.

For other businesses, I would say it's a little bit different. If you're building your marketplace where you kind of need to fake it on one side so that the other side catches up, then it probably makes sense for many. For something like Bubble, it's not the case. By the way, WebFlow bootstrapped our many – Didn't bootstrap, but self-finance after the seed around for a very long time before the last round they did.

[00:50:08] JM: And it was brutal for them. Another reason I like being in the engineering world is that people such as yourselves are willing to be fairly misunderstood for a long period of time. You basically have no choice but to be misunderstood. You used to work at Bridgewater. You worked at Bridgewater for three years. My guess if you were making a much better living at Bridgewater than in the first three years of Bubble.

[00:50:35] JH: I mean, my salary has still not caught up to what I was making at Bridgewater, honestly. But at the end of the day, working on something that you believe in pays for a lot of quality of life. At some point, it's tough, and we went through something like – Honestly, some

pretty dark periods where we were small enough that it wasn't obvious this was ever going to become a real thing and the money wasn't like coming in yet and we had been doing it – I think the worst part was like 3 or 4 years in I think –

[00:51:10] ES: Three years in, yeah.

[00:51:12] JH: Three years in. That was not fun.

[00:51:13] ES: We still hadn't really taken salary out. Personal finances were starting to become like a four-digit number on your back account, which is low. But we really think what we're trying to do here needs to exist. It's a very important thing. It depends how much you believe in what you're doing. The advice I would give to anyone that wants to start a company, because it is really hard, is would you work on that if you don't have funding and you don't have salary for two years? If it's no, you probably should enter this. But it's hard.

[00:51:42] JM: Did you guys do consulting or something?

[00:51:45] ES: No, actually. Both fulltime. Yeah.

[00:51:49] JM: You did loans? Are you serious that three years and you were down to four digits in your bank account?

[00:51:54] ES: I was. I think Joshua is doing a little bit better. I had an MBA to payback. So I had to payback my MBA. That's how I came to the U.S. But we both had like fairly nice savings from our previous jobs.

[00:52:03] JH: Yeah. We did – Like the big client we mentioned, Dividend, they were paying us to do sort of consulting on top of our platforms. So they were basically paying us to do feature development work. Even when our customer base was small, we were still making some real money from our customers. It wasn't like completely just watching our savings drain away. It was just like mostly watching our savings just drain away.

[00:52:29] JM: Oh, okay. But when you get there – I mean, you get 7 years in and you got traction. I mean, it's cool that you have the nuanced perspective on venture capital, and I think the – It's like you think about like personal financing, personal loans, or like mortgages. We have a vernacular for like when and how you should take mortgages. It's a very sophisticated industry. You can get this kind of mortgage, or that kind of mortgage and it's like, "Yeah, a buffet to choose from."

Whereas venture capital, we talk about it in generally a little bit more rigid terms. There are people such as yourselves who are able to take more savvy approach and kind of look at it from a more what do I actually need from this vehicle and how can we satisfy our needs while also satisfying investors' needs? Tell me about structuring a deal in this kind of situation when you're 7 years in and you haven't raised any money and you've got a fairly profitable business.

[00:53:32] JH: Yeah. First of all, to your general point, I think the culture where raising VC funding is like a triumph in and of itself is like nuts.

[00:53:43] JM: It's like the same for a mortgage. It's like if you celebrated getting a mortgage or student loans.

[00:53:48] JH: Right, like, "Congratulations! We did it. We finally got our mortgage." Yeah. Actually, we ended up with lead investors who so far we really like, but it was like an interesting process, because the series A firms, we didn't look quite like a typical series A company. The seed firms were like a little too far long for most of them. So we're like kind of in this weird niche.

We eventually found a number of firms that like we're kind of flexible with their investment thesis and we're willing to just like say like, "Hey! You look cool." We're into that. So we're able to get some bids and like had a round that we're pretty happy with and great investors, but we did struggle a bit, because – Especially in the first few weeks of fundraising to like figure out who we should even be talking to.

[00:54:40] JM: Have you guys built any side businesses with Bubble yourselves?

[00:54:43] ES: Not really actually. I guess we should have, but we're a little bit too busy. I mean, I built a website for my wedding, but it's not a business. It's a personal website just to gather informations about the [inaudible 00:54:55] restrictions, but I did.

[00:54:58] JH: Yeah. I've built like a personal to-do tracker for myself, which I actually no longer use. I moved off into a different system. We definitely like use it for our own pet projects, but you can only like –

[00:55:12] ES: I mean, but most seriously though, a lot of our internal tools are built on Bubble. We don't use Jira. We use a Bubble built tool. We do a lot of things internally. In fact, Joshua's is talking about the onboarding process and part of our onboarding process – That's true for everyone, not just for engineers, is very heavily focused on becoming a Bubble expert quickly. We have actually a Bubble test to validate your knowledge of Bubble to make sure. Because we need everyone on the team to be extremely good at Bubble, especially engineers, but everyone.

Otherwise, engineers are naturally are not the typical users, and you don't want engineers to work on a product and not using themselves. You have to force that a little bit.

[SPONSOR MESSAGE]

[00:55:58] JM: Looking for a job is painful, and if you are in software and you have the skillset needed to get a job in technology, it can sometimes seem very strange that it takes so long to find a job that's a good fit for you.

Vettery is an online hiring marketplace to connect highly-qualified workers with top companies. Vettery keeps the quality of workers and companies on the platform high, because Vettery vets both workers and companies access is exclusive and you can apply to find a job through Vetter by going to vetter.com/sedaily. That's V-E-T-T-E-R-Y.com/sedaily.

Once you're accepted to Vettery, you have access to a modern hiring process. You can set preferences for location, experience level, salary requirements and other parameters so that you only get job opportunities that appeal to you.

No more of those recruiters sending you blind messages that say they are looking for a Java rockstar with 35 years of experience who's willing to relocate to Antarctica. We all know that there is a better way to find a job. So check out vettery.com/sedaily and get a $300 sign-up bonus if you accept a job through Vettery.

Vettery is changing the way people get hired and the way that people hire. So check outvettery.com/sedaily and get a $300 at bonus if you accept a job through Vettery. That's V-E-T-T-E-R-Y.com/sedaily.

Thank you to Vettery for being a sponsor of Software Engineering Daily.

[INTERVIEW CONTINUED]

[00:57:48] JM: Do you guys think there will be an open source tool like this, an open source visual programming tool?

[00:57:53] JH: I mean, there might be. We have considered someday like open sourcing our runtime engine or something like that. It's like sort of an aspirational idea down the line. I think with something like this, it's just like a huge time investment to build. So eventually it becomes a job and needs some kind of business model.

Like it's not the kind of thing that we could have built as a hobby or something like that. Who knows? I can certainly imagine either us doing something eventually or like someday once visual programming is just like the dominant paradigm, something like rebel group of hackers, does like the Linux style revolution. Who knows? But I don't see it happening tomorrow.

[00:58:37] JM: I think people are less resistant to paying 25 different SaaS providers every month than paying a gigantic build to Microsoft like it was in the 90s. I think people are content paying a little bit of tax to a collection of proprietary providers rather than one single system.

Josh, you worked at Bridgewater. So what cultural practices have you imported from Bridgewater into Bubble?

[00:59:11] JH: Yeah, great question. For people listening who don't know Bridgewater has like a sort of infamous culture, like the CEO, Ray Dalio published this book called *Principles,* which sort of lays out like this whole doctrine. I actually was at Bridgewater, super interest in time. I joined right when Dalio was institutionalizing the management principles. Actually, I started as a technologist, but I joined the team that he was like having like rollout the principles across the company. I sort of got to see firsthand the rollout of this.

My takeaway is like there is like things that were amazing about it and things that I disagree with and don't think worked at all. It's like a mixture of like wonderful and terrible basically. I've tried to take the wonderful.

I think the things I most appreciate about Bridgewater is how they think about hiring and talent. They really believe in hiring for values and deep capabilities rather than valuing resumes or skills or like short-term things. They have like a set of interview techniques, like asking people about their life stories, asking them like why over and over and over again to sort of get to deep understanding. I've definitely pulled that from –

I have taken the sort of intense candor piece of it to like my own personality flavoring of it. I'll often knock on Bridgewater is that it's just like a brutal culture, because people are so blunt with each other and super direct. I try and take the good part, which is like building like a relationship of trust where you can tell someone what you really think. But I think the thing that needs to be caveated with is that like you have to earn the right with someone to tell them, "Hey, I think what you did sucked because of X, Y and Z." If you just say that to someone in a context where you haven't demonstrated that you're trying to help them because you're working on the same mission, it causes a lot of pain. I sort of keep that lesson in the back of my head.

[01:01:27] JM: It's a long book. Principles is a long book.

[01:01:31] JH: Yeah. I had to read the whole thing, or I got to see the early drafts, the evolution overtime. Yeah.

[01:01:40] JM: Was there any moment where you're like, "This guy is completely mad. This is holacracy. I mean, it could have been holacracy. Holacracy was this big thing that was

institutionalized at Zappos and it, by all accounts, was really detrimental to the company. Great experiment for the overall ecosystem to see. I love it. Much like – I think as big of an experiment as what GitLab is doing with the all remote thing. Kind of beautiful from a macro perspective. Kind of scary if you're in the middle of that company and somebody is institutionalizing something this rebellious.

[01:02:23] JH: Yeah. I think the core of Bridgewater's experiment is scaling via culture versus scaling via budgets or processes or tools or whatever. I mean, Bridgewater has all that, but like they were really trying to scale via convincing everyone in the company to like follow a search and set of shared cultural norms as a way of operating successful at scale.

I personally don't – I mean, Bridgewater is a hugely successfully company. At some degree, I don't have the perspective to judge the whole system. I kind of don't feel like that part of it is a success. I think the culture, like a lot of their culture is like good. It's like healthy. It's like a good thing. But if you use it as a management tool and you create a situation where people are expected to conform to it as a way of being graded on their jobs, it kind of creates these like perverse incentive structures where people are trying to look like they're following the culture rather than trying to follow the spirit of the law.

My takeaway is like I certainly do not plan to like try and do the same thing at Bubble. I want to take some of the good feel of it of having like a really transparent and genuine and honest way of interacting, but I don't see institutionalizing culture as the primary management tool. I see it as like one tool in a portfolio of other things, like putting great processes in place and putting their organizational safeguards and checks and balances in place and designing an organization.

[01:04:21] JM: All right, last question. What would each of you be working on if you're not working on Bubble?

[01:04:27] ES: I mean, it's been 7 years now. So it's hard to imagine my life different, to be honest. I'm not sure we'd be in tech, first of all. I'm very French in some ways, and I still believe in government doing good things if well-managed. I might be trying to do some public work type of thing in France. I guess that's what I would be doing.

[01:04:46] JH: That's a less ambitious answer than I think if you had asked him a couple of years ago. I've heard like president of France.

[01:04:57] ES: It's the same answer, it's just a timeframe issue. That would come after. Yeah, I believe in the public space actually. From an American perspective, it's less the case, because government in America has much less power than it has in Europe, particularly in France. But in France, it does have a lot of power. Yeah, I would start with public work first before probably trying to be in office at some point.

[01:05:19] JM: I can't wait until our governments get more tech-enabled. It's going to be so awesome. The next generation of politicians is going to be so much more tech-savvy and intelligent with how they deploy stuff.

[01:05:29] JH: Yeah. Although it'll be interesting to see –

[01:05:32] ES: You might turn wrong. The Chinese government is very tech-savvy and they do some scary stuff.

[01:05:39] JM: I don't know if it's the government themselves that are tech-savvy. It's more like – They're like we want to see everything, and you must implement it. They point to the tech-savvy person.

[01:05:47] ES: Yeah, but they have the knowledge that it's even possible. I'm not sure people would know that in western governments. At the end of the day, what's happening is that they're leveraging technology to serve the government interest in a way that I think no other country does and it leads to some interesting ethical questions.

[01:06:02] JM: I think that generation Minecraft, whether you're talking about whatever the Minecraft equivalent in China is or the American Minecraft users, I don't think you're going to get any authoritarians out of that generations.

[01:06:15] JH: Yeah. I mean, it'll be interesting to see how it plays out. I think generation has change da lot where it like I think if you look what happened to the baby boomers, like the hippies to like what they became after that war done, so. I'm very interested to see what happens. I am excited.

I think you're absolutely right that like people who grow up tinkering and having a sense of like agency and creating the environment around them, whether that environment is made of blocks with weird troll creatures trying to kill you. I think it only leads to good stuff. I think we live in an interesting time. So it's kind of like a privilege to see how it's going to play out.

[01:06:56] JM: All right. So, now your answer.

[01:06:58] JH: Right. I would probably end up being a political philosophy blogger is probably what I'd end up doing. I'm just really interested in the questions of like how does societies work? I mentioned agency a second ago. Agency, like how do individuals have power? How does that flow into collections of people? What does like progress look like? What does it mean for society to go in a good direction? How do you reconcile some of the big challenges of, A, want to preserve individual liberty and freedom on the one hand, but B, dealing with like a world where we're all interconnected. If I like pollute it gets in someone else's face and takes the whole earth down with me. How do you reconcile that kind of stuff?

I could see myself just like deep-diving on that and like trying to come up with some kind of answer. I have no idea what it would be though.

[01:07:59] ES: I guess you can see that we both have interest in like the social matter, like public, like social issues. I think that's actually what got us together and what got us through all these years, particularly the hard years, because I really think what we – The problem we're trying to solve with Bubble is very important and not just empowering people to start startups without engineers. That's kind of the tip of the iceberg.

Fundamentally, what we're trying to work on is access to technology, computer literacy, ability to being independent and not depend on big tech companies. I think that has – What I'd say that if

we don't solve that, I hope someone else does, because this is a very important problem that is much bigger than creating a new piece of software.

[01:08:40] JM: I agree with you. It's very profound. I was thinking about the cans of worms of the interactions with people I've had in New York recently politically-wise. New York is kind of a hotbed of some interesting political beliefs these days, but we could save that for another podcast.

Guys, thanks for coming on. It's been really fun talking to you, and I'm with you on Bubble. I think it's an inspiring platform. I'm glad you guys are doing it.

[01:09:05] ES: Thank you very much.

[01:09:06] JH: Yeah, thanks so much for having us on. It was a lot of fun.

[SPONSOR MESSAGE]

[01:09:17] JM: You probably do not enjoy searching for a job. Engineers don't like sacrificing their time to do phone screens, and we don't like doing whiteboard problems and working on tedious take home projects. Everyone knows the software hiring process is not perfect. But what's the alternative? Triplebyte is the alternative.

Triplebyte is a platform for finding a great software job faster. Triplebyte works with 400+ tech companies, including Dropbox, Adobe, Coursera and Cruise Automation. Triplebyte improves the hiring process by saving you time and fast-tracking you to final interviews. At triplebyte.com/sedaily, you can start your process by taking a quiz, and after the quiz you get interviewed by Triplebyte if you pass that quiz. If you pass that interview, you make it straight to multiple onsite interviews. If you take a job, you get an additional $1,000 signing bonus from Triplebyte because you use the link triplebyte.com/sedaily.

That $1,000 is nice, but you might be making much more since those multiple onsite interviews would put you in a great position to potentially get multiple offers, and then you could figure out what your salary actually should be. Triplebyte does not look at candidate's backgrounds, like

resumes and where they've worked and where they went to school. Triplebyte only cares about whether someone can code. So I'm a huge fan of that aspect of their model. This means that they work with lots of people from nontraditional and unusual backgrounds.

To get started, just go to triplebyte.com/sedaily and take a quiz to get started. There's very little risk and you might find yourself in a great position getting multiple onsite interviews from just one quiz and a Triplebyte interview. Go to triplebyte.com/sedaily to try it out.

Thank you to Triplebyte.

[END]