HolyJS Interview Questions

So the point is: We'd like to ask some questions about your experience with startups: were there any fails you've seen or inspiring successes

Although I've been a freelance developer for 11 years, I've haven't been involved with many traditional startup situations – at least not as a traditional employee. I'd been brought on as an outside developer or consultant for specific tasks.

I wouldn't mention any failures by name, but I'll say that there are many aspects of "startup culture" that seem like a failure, to me. That there is a lot of money being tossed around there does not automatically make the work important or world-changing, and I think a lot of the industry is not honest with itself about that.

As far as inspiring successes, I had what I considered a *personal* success when I built a "social network" service entirely from scratch, writing my own authentication libraries and other things that I absolutely should not have done! It was very, very early in my career and a very ambitious undertaking, but it shipped, never broke (amazingly!) and I learned a tremendous amount in the process. This was in a pre-iPad, barely post-iPhone, still-designing-around-Internet Explorer 6.0 world. I would not do that again and I'd be a little embarrassed to share that code, but I'm proud of the result.

5:50

how the life of a startup differs from life of an enterprise from the view of independent developer

There's more money and certainty in enterprise work, but startups almost always sound sexier – particularly when you're younger. With enterprise work I don't worry about getting paid, I don't get equity offers, if I do a good job I can expect repeat business.

Working with startups is a little bit more of a gamble, but I think you can get good at choosing the *right* kind of startup projects so that that's not always the case.

After a few stressful projects early on I came up with an expression: “Be careful about getting in on the ground-floor with other people’s dreams.” Staying up all night and debugging things for your own projects is one thing. Doing it for another person, no matter how much they’re paying me sometimes, is another. After doing that a couple times I'm more mindful about which startup projects I entertain and have gotten better at recognizing the "headaches" ahead of time.

I did work for a professional sports team for a year and a half, and there was a lot in common with startup. We had a small developer team, lots of different projects that were being defined as we went along and a surprisingly constrained budget situation. Frequent pivoting and internal organization seemed like some of the biggest challenges, which is something I had grown to expect more from startup culture. Things aren't always what you think i guess!

5:52

what are the differences among countries you've been working in in terms of technical and social conditions?

I spent a year working and traveling around the globe visiting and living in 18 different countries. It completely changed me and I think about it all the time.

While I was traveling then I would reach out to local developers most of the time, partly to get a feel for what it was like to be a developer in that area and also as an excuse to network and socialize in a new place. I can’t speak to what it's like to work in any of these countries in an authoritative way, obviously, but I definitely had perceptions and found it interesting to see how things were different in other countries.

Some things are consistent. The eagerness to connect and network seems to be everywhere I go, and particularly enthusiastic and optimistic among developers. The second I tell people what I do business cards come out of the woodwork and people want to immediately start talking about projects. Spain, Turkey, Kenya, Serbia, Thailand, Japan. It happened in all these places. In one memorable moment I knocked on the wrong door, looking for a web development office. Through my broken Spanish we were able to clear up the confusion, but I left with businesses cards, and email address and an earnest suggestion to "get in touch."

Where the work come from seems to vary and can be more limited in some places. In Turkey for example I mets many wonderful, friendly developers of all kinds in Turkey. Their skillsets and tech-stacks of choice seemed to be varied, but a huge number of them seemed to do work for Türk Telekom, which surprised me a little. Perhaps that's normal or coincidental with the crew I people I found.

I met with a dev and design shop in Nairobi! That was one of the highlights of this endeavor to meet other developers, for a lot of reasons. They focused on small business websites, but also work for American agencies and firms – sending the kind of email sI'd honestly get in the middle of the night back in the U.S. and probably delete in the morning without much thought. But putting a face to that kind of email was a good experience. I mean… I still trash them most the time, but I try to do it more nicely? They worked in shifts to overlap with typical U.S. hours. They also had a substantial portion of their business dealing with text messaging and M-PESA, which is a huge deal in East Africa. Fascinating to learn about stuff like that and see it in-action firsthand.

Other difference: Sometimes clean water is a harder problem than internet access. Working in a timezone about 10 hours ahead of most of my clients back in the U.S. seemed ideal to me and conducive to a great work/life balance: I could work in the morning and evenings where the hours overlapped and spent the daylight hours out and about exploring.

5:52

any specifics like internet connection, government rules, time zones etc

In some countries I was blocked from accessing certain websites. Some I was prepared for and had workarounds ready, such as in China, but others surprised me. In South America I was blocked by a couple hosting companies which surprised me. But it usually not too hard to circumnavigate – VPNs and proxies are a developer's friend! I will say I learned some interesting things about the Internet in North Korea during a visit there... but I never tried to work from there. Fairly certain it would be impossible.

The wifi situation in airports differs all over the world.

I was also surprised to find the Internet is metered in some countries. It was in Iceland. Was in a house with 4 or 5 other people and it ran out and slowed to a trickle one day. At the end of the month it renewed, which happened to be the next day, but that was something I wasn't used to. In the US the non-mobile plans are almost never data capped and I never think about it.

5:53

was there any weird experience? Expamples are extremely welcome

Haha. There are always weird experiences!

Some conversations that were broken. Getting paid in Bitcoin wasn't weird but kind of funny and new.

5:56

could you tell then a couple of words about your commercial projects you earn money from. Here we expect you to tell some details about your technology stack and reasons you've picked some technologies

I don't really have any commercial projects that bring me in money. It's all development, consulting and – lately – teaching.

I think JavaScript has given me the most bang for my buck throughout my career, but that's not very particularly specific or helpful! I have a few internal tools I use to manage my time, projects and billing in Python and Ruby. Most of my newer, more fun stuff seems to be built with Express. I still have a good amount of work through agencies with WordPress and have inherited a couple PHP projects over the past couple years. I'm trying to find a good project for React.js but it hasn't happened yet.

You have to pick your stacks based on a blend of what is useful to you, what you find innately appealing/easy to understand and where the community traction is. The latter probably has more influence in what any of us use than we'd like to think.

I truly contend that for 90% of what we're all doing the language does not matter as much as we think it does. Different modes of thinking, different efficiencies that *do* matter at scale for really large companies with sophisticated needs (Google, Facebook, Netflix, others). But for the vast majority of us, how often does that really come into play? I think about this a lot.

5:59

opensource is a modern topic and even a buzzword. How do yhou feel about writing OSS, some say that OSS will rule the world in 10 years. Corporates invest tons of money into this. Does that mean we all will freelance and write open source code soon

It already does rule the world! Maybe it just doesn't look the way people think :)

The web literally couldn't run without open-source, so I have a hard-time seeing it as a buzzword or even as being particularly modern. I think our attitudes and conceptions of open-source have certainly changed though. There is a pre and post GitHub perception of what open-source means to some people that I can't quite distinguish but I can see it. There's also a big difference in my mind between GNU and the Free Software Foundation, who embrace OSS as a sort of philosophy, and developers contributing to corporate-initiated OSS projects as a path to some kind of career.

Some OSS is released as a means to empower or provide a public good. I think of SecureDrop as an example of this. Other times I feel like its corporations basically trying to get free labor for an internally developed tool. That may work fine so long as the it continues to align with the corporations needs, but once they shift focus and leave the project for greener pastures the project's fate might be sealed. Truthfully, I don't trust Facebook much in this department, and they're the corporation I think of first when I talk about this distrust. Maybe this has been a large contributor to my slow adoption of React.js!

I mean, strictly speaking, most of the open-source projects we're talking about are tools and frameworks for building other things. Almost all of us use these things even if we're not actively contributing to their development. I've made tiny contributions only when I stumble across edge cases and bugs or need to develop an adapter or plugin to make it talk to another service that it doesn't currently support.

So if the question is will we all be working exclusively on tools? I mean, no. Working on tools is great, but I think the real work is when you use them to build things non-developers will use. That's where you run into the real-world edge cases and bugs and problems you need to solve.

Side note: my biggest OS contribution is completely frivolous: konami-js. I'm proud of that :)

[6:01]

Now common developer practice is to find a warm place in a large company to get $100k/year salary. How do you think everything will change with OSS widely spreading all over the world

Is that a common practice? :)

Like I said, I think it's already spread. To a very large degree OSS has the leveled the playing field in a lot of areas. To touch on a previous project I mentioned, I think it's fabulous that a publication like the New Yorker can run SecureDrop and provide a safe place for anonymous tips. If that was proprietary and secretive we'd almost certainly see fewer organizations able to offer that service.

So I think the playing-field will continue to be leveled. Technologies that seem unattainable through expense or other means for smaller organizations 10 or 15 years ago will become cheaper and more available. Publishing platforms, security, voice and image recognition, AI and deep-learning – these are all things I can drag-and-drop into my project these days, which is pretty remarkable!

On the flip side of that optimism: the tools are open-source, but the end-product often isn't. And more and more the organizations that dictate the direction and popularity of these tools are larger corporations with very different interests and needs.

But, from an employment perspective, which is what I think this question is asking, I'm not sure open-vs-closed software makes as huge a difference. I this is a very U.S.-centric perspective on that though.

[6:04]

And the last one: lifecycle of a modern JS framework is quite short now (a couple of years often). Now React, Redux, Angular, Vue etc rule the JS world, how long will they live and what's next? Can you give a piece of advice on where to invest my time and efforts to be up-to-date in a couple of years?

If the lifecycle for a tool is short, I'd argue that's not a lifecycle — that's a trend. Man, I am realizing how old and crotchety I'm sounding with some of these answers! :)

People like to reinvent wheels every so often. I actually think that's a good thing, for learning, but a difficult thing when these new wheels are being touted as the direction everything needs to go. What's funny is we've been down the road before. I remember Google Web Toolkit and SproutCore and Cappucino and other frameworks designed to help us build better web apps. The tools are reinvented from the ground-up but the concepts are typically only tweaked. I think that's important to remember.

The best advice I think I can give: Learn how to learn things, ideally quickly. Know enough to get the "gist" of something and know where you'll need to look later to nail down some of the specifics. If you can, identify with being a developer that knows the concepts integral to programming in any language or framework, identify with the underlying language the things are built-in, but I'd say be wary of identifying *too* strongly with the tools.

In my mind it's a little like being a contractor that specializes in only using a certain brand of hammer. Would you hire that contractor to work on your kitchen? Would you find it a little odd that they market themselves that way? Maybe, but ultimately what you want is someone that builds things and understands how they are built.

I guess, in short: don't confuse your skills with your tools. I see a lot of people that do and when the technology shifts they feel like they've been left behind.

A while ago I took a Lyft somewhere and ended up talking about work with the driver. The driver told me they used to do web development work too. I asked what kind of work and they said "A lot of PHP and MySQL development for financial institutions… But all of the PHP & MySQL jobs dried up! Too bad."

I mean, that stack isn't super sexy or cutting edge now, but the underlying concepts and being able to write object-oriented code that interacts with a SQL database should be a *very* transferrable skillset! I think sometimes people identify too strongly with the language/framework/tool.

So learn the underlying concepts, learn the core language behind the frameworks. Keep your general programming chops and concepts sharp, but I'd argue be wary of identifying too strongly with being a "React Developer" or "Angular Developer." There are a lot of jobs out there now for that, and there may be for a long time, but if you continue down that path… Someday it's all going to be legacy work. I mean, hey, there are still COBOL programmers and Cold Fusion shops out there, so who am I to judge? :)

6:05

That's it i guess. Are these questions ok? We can skip or change some of them