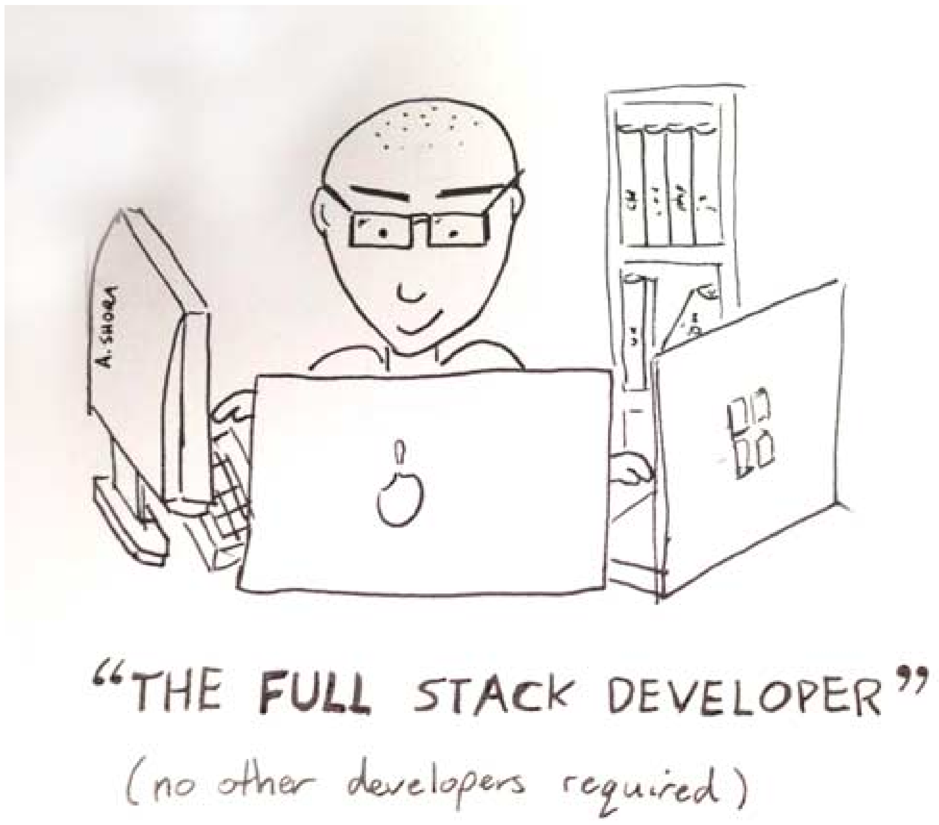
The Myth of the Full-stack Developer

A well-balanced article/controversial rant about the term 'full-stack developer'. By Andy Shora

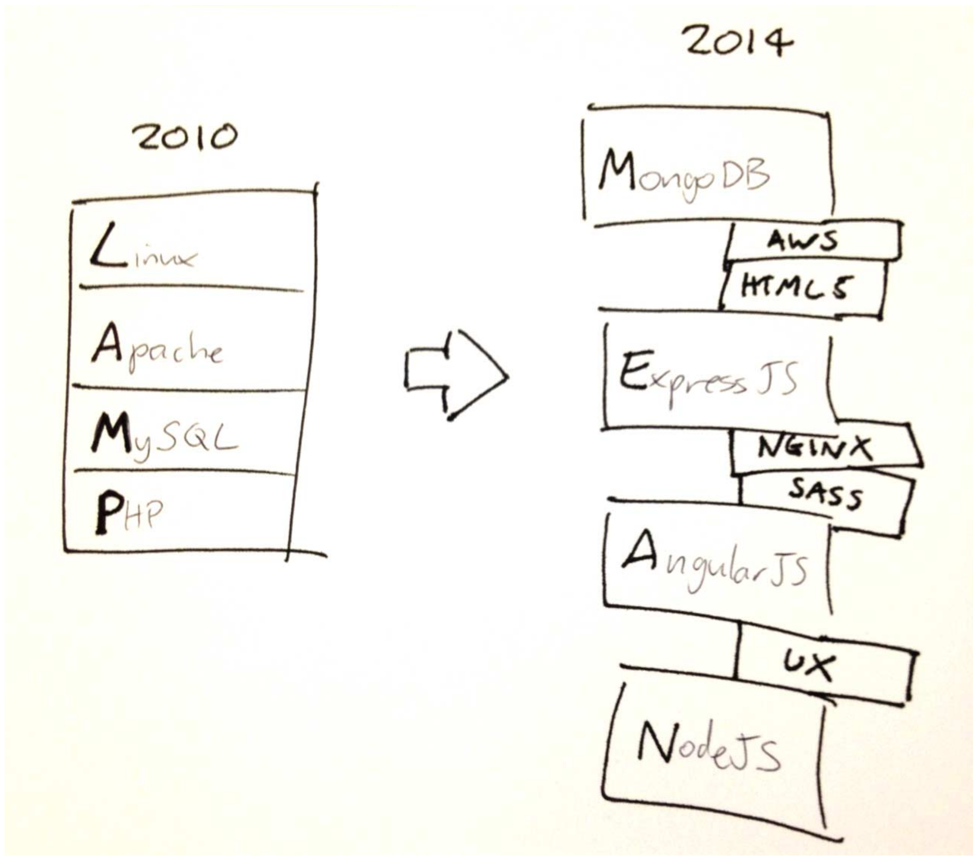
← back to andyshora.com | Tweet this article



"Full-stack."

My defensive tendencies are normally put on high alert when I hear that magic phrase. Stacks are a lot bigger than what they used to be, and being able to claim one has acquired refined skills at every layer of web development is certainly not a small claim. Does this mean you have a broad range of skills or you specialise in everything?

For a time (allegedly) Facebook only hired full-stack engineers. This was of course when they were building the first few versions of Facebook, which, lets face it had a relatively simple php backend and wasn't anything special design-wise.



Stacks are a lot bigger than they used to be...

Full-stack used to mean less layers.

Coding php or Python, jQuery, HTML, CSS then transferring a few files via FTP to your shared hosting account or dedicated server? You were full-stack. HTML was trivial, and even thinking of implementing any proper application code in JavaScript wasn't possible.

My beef is not with people who can do all of the above, it's with the label 'full-stack developer'. **What does it mean in 2014?**

At the level of a senior full-stack engineer we're perhaps talking about architecting a modular Backbone/AngularJS front-end whilst optimising content delivery and tweaking hardware accelerated layers in CSS, followed by implementing an async non-blocking backend (which also pre-renders templates on the server), and pushing to an AWS cluster which has been built with security and scaleability in mind. Not to mention design. Designing UI responsively and mobile-first is essential, utilising a CSS pre-processor to save time. And remember to setup Nagios for monitoring. Oh, and ideally when the Continuous Integration server detects a bad build because your end-to-end tests have failed, get it to send you an SMS with the build error messages. Well, the last two are probably for bonus points.

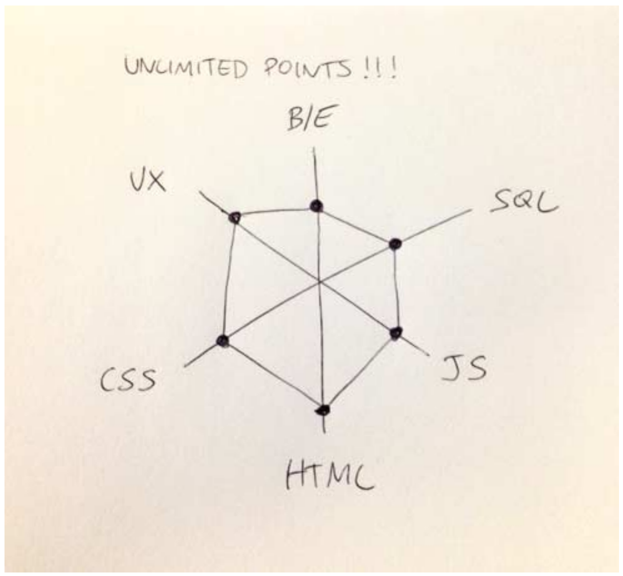
Ask someone who labels themselves 'full-stack'. Do they believe they have familiarity with different layers, or true mastery?

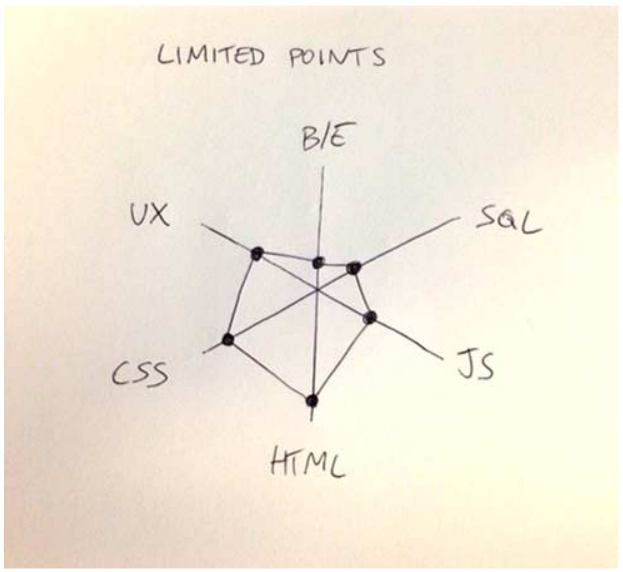
Identifying mastery

I used to work with a very smart guy who when interviewing junior web developers, used to draw a spider diagram with each of the spikes being different layers of web development; UX, HTML, CSS, JavaScript, backend scripting language, and SQL. When the young developers were asked to grade themselves out of 10 for each of the skill areas, most went for a good spread, but a rather high spread. It seemed **nobody wanted to exceed an 8, or drop below a 5 in any of the layers**. We can ignore the scale at which they were grading themselves, because we were simply interpreting the grades relatively against each other so we could get a good idea of what kind of developer the individual saw themselves as. I didn't hear the term 'full-stack' being used even once, but we got a feel that everyone was an all-round web developer. What happened next was more interesting.

On the second iteration the interviewees were given a maximum of 30 points to distribute as they pleased. After some frowning, some internal arguing, and finally the self-grading we saw a lot more skewed graphs. **The 7-5 well-spread ratings turned into more skewed 8-2's**. It seemed when faced with the harsh reality of admitting what they were really good at, they delivered, giving themselves 8's in their favoured areas and 2's in others. If the candidates really believed in their initial relative spread, we would see the same weighting in the second round, but instead **we saw front-end and back-end developers emerge from the crowd**.

Of course 99% of web developers have probably never sat through this test, and have perhaps have never been forced to reveal what their real spread of skills is.





Asking web developers to allocate points to their skillset.

How skilled are you in each discipline?

The basics of the languages/frameworks we learn today can often be picked up in a matter of hours. We no longer add skills to our CVs by taking a course, we simply download some code and start hacking through tutorials and demo code. The problem is, I feel the difference between knowing something in web development and truly mastering it is now becoming an increasingly blurred line.

It's very easy to become disillusioned with your own skills when you've deployed some code which has been consumed by lots of users. Let's say you've written a web application which has scaled well under load, and you've received great feedback. Are you a master of everything you used in this stack? Or are you simply good at implementing the layers you needed to make things work together? Because that's an entirely different skill, and in fact a very valuable one.

The skill of acquiring new skills

In my eyes the most valuable skill to have is the ability to learn, closely followed by the ability to know when you don't know something. I'm sure we've all met people who decide to put brakes on their own learning because they believe they've become a master at something. I don't know about you, but the smartest people I know (and the ones I have most respect for) are the ones who are still eager to learn new things every day, from everyone they meet. These people just don't stop growing, in terms of both ability and character.

Why I'm not a full-stack developer

At one point (probably when I was about twenty-two) I heard this phrase 'full-stack' and I thought "yeah, I'm one of them". Five years later after working in roles with ASP, php, .Net, Node and finally JavaScript and CSS, I only label myself as a front-end developer. Maybe I just wasn't that good at previous disciplines, or maybe since spending the last 2-3 years in pure JavaScript roles I just actually realised how high the limits are when you specialise. This is simply opportunity I did not have when I was busy in cross-discipline roles at smaller companies.

Sure, I've got some broad skills like a lot of developers. However, I'd like to think being a front-end developer now assumes some knowledge of UX, design, and how data is modelled and served on the backend, and this is why I think the term full-stack is old fashioned (if five years is old) and perhaps a bit smug.

Maybe, I just don't like the term.

Some people are willing to have a go at everything

Perhaps most respectable about all people who claim to be full-stack, is that is conveys ambition and persistance. The ideal developer should be willing to have a go at learning anything, but also acknowledge when they need help from a specialist in the team.

Perhaps I was too defensive at first, I've met too many people who pronounce the term in a smug way, instantly causing me to discredit them. However, say you're full-stack in a normal way, and there will be no scrutiny from my part. It won't be at all like meeting an investment banker or recruiter.

The employability of a true full-stack developer: HIGH.

The chances of finding a good full-stack developer: LOW.

Smaller companies and startups NEED full-stack developers. Developers are often forced to acquire new skills when the resources simply aren't available. I feel the problem for companies desperate to hire these guys and girls, is that the real multi-skilled developers are often lost in a sea of douchebags, claiming they know it all.

Please harass me in the comments

As I wait for the inevitable backlash of a thousand-strong army of full-stack developers on Hacker News, I sit here tinkering with a piece of JavaScript for my next 'big app', which will no doubt be undone by a security flaw that I've overlooked due to my lack of sysadmin skills.