Keep Calm and Stay Rational - FAANG is not doctrine - Just because Facebook uses it doesn't mean you have to too

Written by Seth Corker on Benevolent Bytes

Facebook, Apple, Amazon, Netflix and Google (FAANG) are the highest performing US technology companies right now. With that title comes significant sway over what technology choices we make as software engineers. They sponsor conferences, open-source tools and they export their beliefs around the tech world. For better or worse, they influence many decisions we make when designing software systems. How much weight *should* their opinions hold?

Why developers worship idols

The thought is, if you do everything like Facebook then some of their success will rub off on your product. You'll be able to serve millions of users with unparalleled performance. Build your product to scale and the users will flock to you. This thinking is unrealistic. It doesn't work like that. Cargo cult programming fails to understand the whole picture.

You don't have the same problems The problems that these companies have solved is particular to companies who've grown to massive scale. The systems they use as large behemoths of tech are very different from those needed by smaller businesses. They're made to handle challenges you just don't have yet and might never have. These companies didn't start large and although their aspirations were big, they likely didn't make the same decisions as their large competitors when first starting out. Your use cases are very different and you should focus on adopting technology that's best for you and your users at the time.

When do you have the same problem? Having just said that you don't share the same problems as a big company like Google or Facebook is likely true but there are common challenges that everyone faces. In these instances, leveraging the expertise of a big company can make sense. Let's use a popular library as an example.

React is a solution to a specific problem. It's a library I use daily and many companies from small startups to large enterprises use it some capacity. It was created by Facebook to solve their *own* problems but it's a problem many companies face.

- **Will it suit everyone?**
- No.
- **Does it suit people who want to create more interactive and dynamic UIs with JavaScript?**
- Yes.
- **Should I use it?**
- Depends, did you ask yourself if it's a problem your facing and why it's the best option?

React is just one of many libraries and frameworks that's used to enhance interactive experiences on the web. It's not the only one, maybe you want to try a more opinionated framework then Google has Angular. Maybe neither of these solutions works for you, your peers and your particular use cases. That's okay too, you don't have to follow the framework zeitgeist.

Everyone's doing it If you're reading blog posts and technical articles, watching videos and attending conferences, you might get bombarded with the sense that there is something new and better that's just been released. You might get the feeling you're lagging behind. You'll feel obligated to use the latest and greatest in fear of being ostracised for your archaic tech decisions. Conversely, you could code like nobody's watching and not care what anyone thinks. Just because other's are using something doesn't mean you need to drop everything and do a full rewrite.

It's good to keep up to date with what's out there but you want to keep one eye on the outside world and one eye on the product your building now. You don't necessarily want to be an early adopter, with that title comes a lot of headaches and heartache. If you encounter a problem that can be solved by something else, evaluate it and see if it's worthwhile. The cost is much greater than just adopting a new library or framework; it's added training, maintenance and support.

It will solve all our problems! You've determined that a framework used by Google is the perfect fit for your product. Everything looks good but there's a problem, you've already got a working system that uses a different framework. This new one might be faster and easier to use so let's just... Stop! Before you proceed with ripping up hours of work, take a deep breath and think. Replacing a framework is never as trivial it may

seem on the surface. There's always trade-offs, wasted time and more training to get everyone up to speed. Consider swapping a framework, a major pivot.

It's easy to caught up in the hype and jump onto the next new thing. It's new. It's exciting. It promises to solve all our problems. It's unlikely that this new framework will solve all of your problems, if it does, it may just create new problems instead. In software, everything has a trade-off, you just have to choose which features are the most important and if the trade-off is worthwhile. You could tear everything up and rewrite your codebase using this new framework but what's the real benefit, is there an actual problem being solved or does this new framework have the prestige of being new and shiny?

Photo by Ryan Bahm on Unsplash

What should I use?

Identify an audience and a specific problem. The challenges in crafting a solution should become apparent sooner or later. You need to take into consideration the environment in which you're making a decision too. If your entire team is skilled in particular technologies, it should be counted when evaluating a switch to a new technology. Don't just jump on something because a FAANG company has had success using it. These companies have a wealth of knowledge and expertise and should be looked to, just not as the be-all and end-all.

When looking at technology choices, see why they were created. Ask what problem they are trying to solve and why if it's a good solution. The technology should solve a problem you have now or in the very near future. Ask what does it do well and what it prioritizes. You need to identify alignment with the priorities of the solution and your own product's priorities. Finally, you should also note when the solution would not be a good fit so you can identify the limitations and work around them.

Title card photo by Edwin Andrade on Unsplash