Avoid JavaScript Flavor of the Day

If you've done JavaScript development for any length of time, you are familiar with the huge amount of churn that occurs within the JavaScript framework landscape. This has been covered and panned extensively and is often a source of pain amongst the JavaScript development community. [How it feels to learn JavaScript in 2016](https://hackernoon.com/how-it-feels-to-learn-javascript-in-2016-d3a717dd577f#.mswivflei) is a perfect example of what JavaScript developers can and often do go through today.

How do we effectively manage this churn, its associated pain and get real work done? I think an important first step is to avoid the JavaScript flavor of the day. Let me explain. It is all too tempting for us as developers to pick up and start using the latest and so-called greatest framework to make our jobs easier. We buy into the promises that the latest flavor of the day offers. Take AngularJS for example. It promised an easy and quick way to develop JavaScript MVC apps and it came from Google, a very well-known and successful web company. The JavaScript development community bought in big time. There are countless apps in production today that use AngularJS 1.x. It very much became the flavor of the day, so much so that some recruiters and job descriptions don't ask if you have JavaScript experience; they ask if you have AngularJS experience. So, what happened to that flavor of the day? We now have AngularJS 2.0. In effect, we are being told that AngularJS 1.x is crap and that, we should use Angular JS 2.0 instead, even if it means a rewrite of our project.

Our faith in AngularJS is broken so we start looking for a new flavor of the day to solve our problems and make our development lives easier. Enter React, the current flavor of the day. Like AngularJS, it promises to make our development lives easier. In React's case, it promises to make it painless to create interactive UIs and efficiently update and render just the right components when your data changes. Like AngularJS, React also comes from a very well-known and successful web company, namely Facebook. Perfect! I don't have to do this AngularJS thing anymore. I can use React and my development life will be grand. And I can trust React because it has the backing of a big, successful web company. So, we plunge head first into React development.

So, what happens when Facebook decides to radically change React or worse, remove it all together or change its license agreement making its use within our client or company prohibited? We are again stuck with a lot of code that is either obsolete, won't work or must be thrown away. It's the AngularJS story all over again. We are victims once again of buying into the flavor of the day.

What can we do to mitigate this, get real work done and prevent dependencies upon stale/obsolete flavors of the day? When choosing frameworks:

\* Avoid version 1.0 and the hype behind them. At a minimum, wait for version 2.0. Let others be the guinea pigs to flush out the flaws in the initial version. Limit 1.0 usage to proof of concept work.

\* Look for real, production level case studies from trusted companies using the frameworks you are considering. If other trusted companies are not using a framework, you should not either. Links to popular tech websites that simply mention a framework are not case studies.

\* Avoid frameworks that involve "transpiling" (i.e. converts the written source into JavaScript that browsers can actually execute). Examples: CoffeeScript, TypeScript and ES6. Frameworks like these can be extremely difficult to debug because what is actually executing in the browser does not match what is developed. And browser support for frameworks like these can be incomplete or missing altogether.

\* Don't use a framework and build it yourself instead. I am currently working at a client whose shopping site serves millions of customers where framework usage is significantly constrained. Our MVC and nearly all of the site's JavaScript is internally developed.

When recruiting and hiring JavaScript developers:

\* Ensure that recruits know JavaScript. Frameworks can be learned. Use [JavaScript: The Good Parts](http://shop.oreilly.com/product/9780596517748.do) as a guide for interview questions.

\* Ask recruits about which frameworks they used and why, focusing more on why and what alternatives were considered. Be wary of answers that imply "because it is the flavor of the day".

Most importantly, remember this. Code we develop and the frameworks upon which they depend will most likely last years, well beyond the typical flavor of the day timeframe. Make choices that will ensure that the code and the frameworks are maintainable and supportable.

What do you think? Feel free to reach out to me using the contact links below. I would like to hear from you.