Which JS library should you use?

There are a huge number of libraries out there, and for any given bit of functionality, there are likely *multiple* libraries that accomplish that bit of functionality. For example, there are so many date-picker libraries out there, that there are articles like "Top 15 jQuery DatePickers" to try to help developers pick from them.

But too many choices can turn into decision paralysis for us web developers. How do we know which one is best? What if we make the wrong choice?

There's often not a single "best choice" in web development. But there are often better choices than others, and thinking through the considerations below can help you make the better choice.

Since a JS library is often used when developing a user-facing product, these considerations should satisfy two audiences: the developers that must code and maintain the code that uses the library (like you!), and the users that will interact with it.

Will it be a good developer experience?

- **Well documented**: It should be easy to find a reference of function signatures, demos of actual usage, and a more narrative how-to-use guide. If a library has no documentation, it's usually a sign that they are not the most developer-friendly.
- **Flexible**: The demos in the documentation might look great - but might want to use a library in a slightly or completely different way than what the demos show. Look for signs of flexibility - Is it easy to send in configuration options? Is there a documented plugin architecture? Does it trigger many events that you could hook your code into?
- **Responsive community**: You *will* have questions. You *will* encounter bugs. Ideally, you'll be able to figure them out with developers, whether that's the maintainers or users. If the library is hosted on a version control site like Github, you can look at:
    - -- **Number of forks**: Lots of forks (or stars) means there are at least a lot of developers that cared enough to fork the library. That doesn't mean they'll help you, but it's a start! Large libraries often have thousands of forks, more niche libraries have 100s or 10s of forks.
    - -- **Number of issues**: Are there many open issues? That might be a sign that there's not a community effort around responding and closing issues. It can also mean it's just a very popular project with a lot of ideas for improvement, so continue on to the next point.
    - -- **Vibe on issues**: Read through a few issues and pull requests. Are the maintainers receptive to feedback? Do they answer usage questions? Do you get a positive or negative vibe from the conversations on them?
    - -- **External community**: Are questions about the library answered on StackOverflow? Are there libraries that build on top of the library? Many smaller libraries won't be big enough to have a visible external community, but bigger ones like Modernizr or Backbone have significant ones, and that's a big motivation for using them. You can do a search on the internet for the library name to see what kind of results you find.

- **Actively maintained**: Browsers change frequently. Libraries that once worked can suddenly stop working, because they relied on some quirk of the browser that changed. This is specially true of HTML5 shims and polyfills, because browsers are frequently releasing new versions with evolving implementations of the HTML5 elements. Check the date of the most recent commit.
- **Future thinking**: If you're looking for an HTML5 "shim", prefer a "polyfill" - a shim that mimics the API. That way, theoretically, when all your users were using browsers that supported the technology, you'd be able to stop using the library entirely, with no change to your code at all. For example, if you're using a library to use video in your webpage, use a polyfill that will let you use the HTML5 `video` tag, and it will replace it with a fallback technology like Flash in older browsers.
- **Tested**: All good libraries should include tests that make sure their functionality works as expected. When a library is tested, then we can have confidence there will be some degree of backwards compatibility in new versions of the library.
- **Clean code**: We could treat open-source libraries as black boxes, and refuse to look inside of them, but sometimes, you may need to dig inside of the library code to debug an issue or add a new bit of functionality. Take a quick look at the code and see how easy it is to read, and if it has any red flags, like big chunks of commented-out lines of code.

Will it be a good user experience?

If the JS library does not create a UI component, then only the first few of these matter.

- **File size**: How much will it contribute to how much JS your users have to download? For context, jQuery gzipped and minified is 18k and Select2 is 7K.
- **Performance**: Besides size, other aspects of a JS library can affect its performance, like if it does heavy DOM manipulation, graphics rendering, computation, synchronous storage calls, etc. Look for promises of great performance on the documentation, and of course, try it out yourself.
- **Browser support**: Check that it supports all your desired browsers. Many libraries these days purposefully don't support older browsers (which your webpage may need to support), because they're designed to be lightweight and only for mobile browsers.
- **Accessibility**: Many libraries for UI components look great, but they are not accessible (they do not work well for users with visual disabilities). For a quick check, you can run WAVE on the library's demos page.
- **Responsive**: If your users will ever use the UI component from a library on a mobile browser, then it should work well for them there. Are the buttons big enough? Does it use touch events? Does it scale to small screen sizes?

If you've considered all that criteria, and still can't decide between a handful of libraries, then you might try the call-a-friend approach: ask colleagues or developer friends what library they use. You might just find a crowd favorite.

Remember: there isn't one right answer, there isn't one best choice. Also, you don't have to comprehensively review every JS library you are thinking about using, especially if you're working on projects for your own. You can just pick a library and see what you like about it while you use it. You'll start to build a list in your head of your favorite libraries to use, and your own criteria for libraries, and that will help you in your future decisions.