Dealing with third party libraries

In Do you know where your third parties are going? we discussed part of the problem: How to measure the additional impact these scripts have on our site's performance when they add more scripts that may block rendering?

This post will cover some additional concerns I have with third party scripts.

We need to evaluate third party libraries for use in our applications. This doesn't mean to only look at the code and decide which one you like best but it also means to test the code in your application and see which library or script works best in your specific situation.

The first example that came to mind is date/time manipulation and moment.js versus date.fns. There are many comparisons between the two and with many other date/time manipulation libraries.

I first start by researching what others have said about the libraries and how they've compared them. There's been plenty written about the libraries and their differences:

- The 7 best JavaScript date libraries
- momentjs vs date-fns
- Why you should use Date.fns for manipulating dates with Javascript

That's just a starting point, I will then run tests to validate the opinion I formed from reading other's research. I may have constraints that are not present for the people who wrote the articles I read and viceversa.

Some of the questions I would ask, based on research and needs.

1. Does the library affect performance due to size? The larger a library is the longer it'll take to download and parse before it can be used
    1. Is there a way to reduce the size of the library without using bundlers? Lo Dash and Date.fns also provide modules for individual properties for use with common.js require statements
2. How easy is it for a bundler like WebPack or Rollup to tree shake the library to remove unused code from this package?

3. Am I required to use modules and/or transpiler to make the library work?
    1. Am I already using modules or bundlers?
    2. Do I have to support browsers that don't support modules or imports?
4. Is there an existing API in the ECMA 262 specification that does what I need?
    1. Is the proposal part of the ES262 spec? Meaning it has already been published or is at stage 4 waiting for the next version of the spec?
    2. If it's not at stage 4 then what stage? What browsers have implemented the proposal before moving to stage 4?

The last point is important. As ECMAScript continues to evolve, new proposals will appear that will provide native support for the feature only in browsers that support the feature when it reaches stage 4 in the TC39 development process and becomes part of the annual specification.

Taking date/time manipulation as an example, again. Once the Temporal proposal makes its way through the TC39 system, assuming that it does, we have another standard way to manage dates. At that point I hope we'll re-evaluate using libraries as our primary way to handle dates and times in Javascript.

Everywhere you find Javascript you will find similar examples of complexity and performant code and we'll have to continue to priotitize between libraries based on the version of Javascript you work with, any potential need to support older browsers, and your performance budget.