Sam Janvey

Harvard Extension School

DGMD E27: Modern and Mobile Front End Web Design II

Front End Framework vs. No Framework (DIY)

|  |  |  |  |
| --- | --- | --- | --- |
| Problem | UIKit | DIY using CSS3 | Verdict |
| *Readability of responsive widths in layout design* | UIKit offers a very simple syntax for declaring widths similar to “reduced fractions” from our youth! Far simpler than competitive frameworks and can be applied to a multitude of elements, not just grid layout. | Completely modular and customizable, however may be difficult for an outsider to comb through the media queries to understand exactly what is changing at each breakpoint. | UIKit’s implementation of width (which also ties into grid) is far superior for most projects for the main reason that it is easier to prototype a website and get a responsive layout established quickly. The readability in UIKit is the obvious choice considering the widths are clearly indicated in the class names. |
| *Scalability of a site after prototyped using a CSS Framework* | Assuming a prototype has already been set up using UIKit, how easy is it to scale the site to include more features and more developers? The framework is well documented and the class names are very readable, making it easy to build upon the prototype. Some classes do include css and js which could cause issues with overriding certain features over time since there isn’t a great separation of concerns. | Presuming the prototype was developed using CSS variables to control layout and responsiveness, the project becomes easier to scale, not harder. All additional features of the site are implemented a la carte with any concern of class name collision or unintended consequences from overriding. | Over time, the best solution to scaling a site is to develop the features you need yourself and use CSS variables to adjust for responsiveness (Grid column widths, etc) and overall theming. |
| *Solving design issues which require JavaScript* | As mentioned, many classes in UIKit contain a combination of CSS and JS which help implement common design features such as a picture slideshow. With the inclusion of a few class names, you can have a fully responsive slideshow at your fingertips without writing one line of JS. | Writing a custom slideshow by hand at this point is akin to trying to build your own hammer by forging metal and whittling wood. It’s a feature that’s so common, it makes sense to use a framework even if it’s just for this feature. | UIKit, and frankly any framework, would be preferable because it is simply not worth the time and effort to implement common features requiring CSS/JS combination. Perhaps for less common features, it would make sense to bypass the framework’s built-in functionality, but that would be project specific. |
| *Debugging layout issues* | Before any customizations are made, combing through the code of a UIKit project only involves viewing the HTML markup and class names. Yes, you need to be familiar with the classes and their parent/child implications, but the process of debugging is as simple as stripping away classes and ‘seeing what happens’. | In my opinion, debugging custom css (especially layouts) can be very cumbersome when looking at both the markup in the HTML and the underlying CSS. One way to make this process easier is to incorporate CSS variables as much as possible (and SASS to a lesser extent) so that a developer could simply change the variable values and ‘see what happens’. | I’m leaning towards frameworks in this category for the less obvious reason of a lack of separation of concerns actually speeds up the debugging process. Since framework classes often combine CSS and JS, adding/removing them generates larger changes helping to speed up the ‘see what happens’ process! |
| *Making sure your site behaves as expected across multiple browsers* | A framework like UIKit (could also apply to Bootstrap, Tailwind, Foundation, etc) is more apt to common cross-browser issues and has likely encountered and solved them. Therefore, when implementing a feature using UIKit, you are likely to not have cross browser compatibility issues. | Since not all CSS features are widely adopted by all browsers, developers must go through the painstaking process of checking browser compatibility and implementing appropriate fallbacks for older browsers. It can be done, it just takes more time. | Frameworks are the clear winner in this category for the sole reason that you can take most of the features for granted that they will work across most browsers. |

In my analysis of using a CSS framework versus not, of course the answer is: **it depends!** But I won’t cop out that easily.

The clear winner is the framework for the obvious reason that it reduces the amount of time and effort for a developer to prototype a site. Yes, there is a learning curve with every framework, and every framework likes to distinguish themselves in some way (resulting in quite the tribal warfare on social media!) but it does not make sense to re-invent the wheel each time you begin a project. We know that we are going to have some sort of grid system to lay out the content on our page, why not simply indicate how many columns and at what breakpoint we want the grid to react? Of course, you could roll your own grid system using CSS variables, which is a fantastic way to learn grids, but haven’t you just accomplished your own framework at that point? Ultimately, we are trying to articulate our (or our clients) vision in a way that communicates to our users, why spend valuable brain cells trying to line up boxes on a page or simply just centering stuff!

The one thing that does irk me about frameworks is the lack of a separation of concerns that’s been drilled into my head since I started to learn development. By writing class names that combine multiple CSS features and JavaScript (like UIKit’s navigation Overlay), we make it more difficult to customize and override the feature. *This is going to depend heavily on the framework’s documentation and how clearly it is written for developers to follow*. UIKit has very easy to follow documentation as well as a loyal community who can help solve problems, another core consideration for choosing a framework. I would concede there must be some happy medium between using a framework to pound out a prototype, and rolling up your sleeves and getting dirty under the hood. UIKit seems to have found that middle ground that allows the developer enough freedom to modify the overall theme of the page without having to spend time on mundane details (like element widths). At the end of the day, we are both developers AND designers and one half of that equation should not overtake the other half. The time that **isn’t** spent moving boxes around the page is more time spent enhancing the brand of your client, or writing custom code for a cool feature.