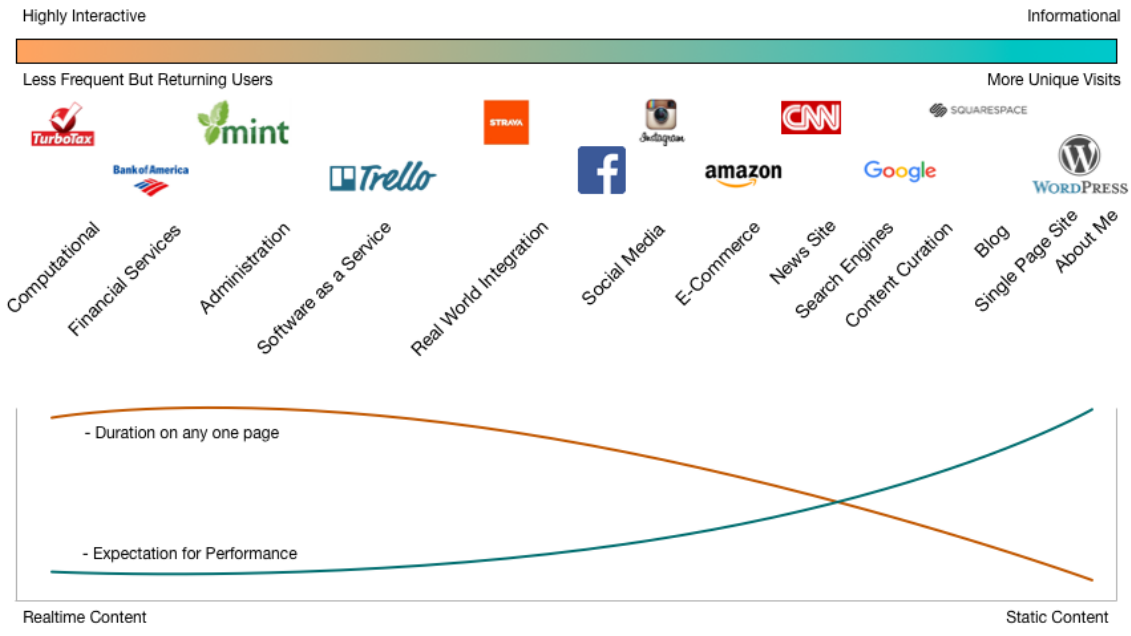
High Peak Solutions
Software Consultancy

Website vs Webapp - A Spectrum

There is an outstanding discussion about the difference between a website and a web-app. Having recently worked for a SaaS company as well as an online retailer, I have come to my own conclusions about this discussion through the lens of both content and performance.

http://stackoverflow.com/questions/8694922/whats-the-difference-between-a-web-site-and-a-web-application

Online services fall along a spectrum from low volume interactive sites to high volume informational sites. Depending on the purpose of your site, you may make design and performance considerations based on your content and usage. For example, SaaS (Software as a Service) may allow you the ability to be highly interactive with your users in terms of feedback, and they may have tolerance for ongoing development, or pre-login performance. Public sites are expected to be informational; there is less tolerance for quirks, poor performance, or mistakes that block the narrow but primary usage. These generalities should direct the discussion of both front end and backend technologies due to various limitations and specialities. It is helpful to first understand the nature of the site or service one is building in terms of expected usage patterns and amount of content change.



One main consideration is whether to use a front end framework (or selecting one). To be brief, a front end framework (such as AngularJS or EmberJS) can be used to build a single page app (SPA) in which the majority of the resources are downloaded on the initial load, may be heavily cached and will intelligently download other resources or data as necessary. A drawback of this architecture is (without server side rendering) the initial cost of the first time load of 'any' page, after which subsequent navigation can be a very fast smooth experience. To put this into the spectrum above, front end frameworks may not be the best choice for a static, low-content changing informational site (far right) where any page is likely to be an entry point. Instead they are a better candidate for webapps where a user comes with intention to use a product and will likely tolerate a loading screen (behind which the app can load), making the experience more transparent.

High performance high volume sites (on the right end of the spectrum) have other design concerns. As the lower tolerance for slow loading increases the bounce rate and frustration of customers, measures like server side rendering and caching can be implemented but at the cost of added complexity for updating content or adding new content. There are also SEO concerns, bookmarking, and high navigational concerns as users fleetingly bounce from page to page, and more often on mobile devices and networks where they have even higher expectations and caching isn't as powerful.

Placing your product on this spectrum can help as a guideline when making architecture decisions early on. You can avoid unintended consequences and difficulties by looking at sites and services in similar industries and using technologies that pair well with the usage and performance paradigm that best reflects the need of your users or customers.

In reality most online services land somewhere in the middle; an admin component that is low usage, highly interactive - manages a storefront, blog, or homepage that has a high percentage of unique visits and low content turnover. Other examples of this middle ground are social media and e-commerce sites where users are both creators and consumers and expectations for performance (and availability) are high.

Taking these usage patterns, the amount of dynamic content and expectations into consideration will help develop better user experiences and in turn will increase the value and satisfaction of your product.

Post By: Scott Traver scott@highpeaksolutions.com