pando.com

HTML, Javascript and the app-ification of the Web

by Steven Willmott , Written On December 6, 2012 • Dec. 6, 2012 • 3 min read • original

At 3scale we're often asked, "Which Web architecture will win? HTML5 or App+API?" And our answer is generally, "Both, they are converging." A great recent post by Zendesk's Alexander Aghassipour and Shajith Chacko on Single page apps illustrates a big piece of how this is happening.

The post described in a nutshell what might be one of the most powerful trends in Web app design -- the move from multipage Web applications to single page applications driven by javascript and access to a powerful API. New frontend frameworks like Backbone.js, Ember.js, Angular.js, and Meteor are driving this trend, and Zendesk is not alone in adopting it. Gmail and Google Docs have long used single page models. New apps like Trello are using it out of the gate, and elements of single page apps (such as infinite scrolling/loading, in-line content) are found on popular consumer sites such as Twitter, Facebook, and Pintrest.

Single Page Applications work by loading a single HTML page to the user's browser and subsequently never navigating away from this page. Instead, content, functional buttons, and actions are implemented as Javascript actions that call a backend API and update both the rendered content and backup server state as the

user engages with the application. The inherent advantages of this are that once loaded and executing, a single page app typically feels much more responsive to user actions, needs less roundtrips to the backend Web server and gives the user an experience closer to that of a native mobile or desktop application.

While there are still issues with how history is handled (much improved in HTML5) and SEO that mean the approach might not be a good fit for content heavy sites, expect a big shift towards this type of architecture especially for SAAS Web Apps where it provides a genuine jump forward in usability.

As well as being an evolution in Web design however, this shift also signals a much wider trend in in the Web as a whole -- a separation of the functional layer as an API and the presentation layer in the form of the App (HTML5 or native):

- The Single Page + API Model is much more flexible than a HTML heavy page based application since the underlying API can be used to power a multitude of different front ends for different contexts, form factors and device types. Once the API is in-place, it can often be used to serve multiple audience with different needs (e.g. channel partners v's end users).

- The model also means that native web apps get much closer in feel to mobile and desktop applications - helping users get a similar user experience whatever platform they are on. Mobile Backend as a Service platforms such as Stackmob, Parse, Appcelerator, Kinvey, and Kii are also helping this convergence by standardising some elements of the backend API and providing cross-platform user experiences that resemble single page models.

- Protocols like openAuth (oAuth) are becoming widely adopted

> as the gold standard for user authorization and provide a common pattern which separates login/authorization issues out from the application and content. For the first time this means that user identity can be cleanly separated from content, functionality and user-experience.

Single Page Apps arguably represent the final step in the unbundling of Web technology -- which started by separating out formatting from content (CSS), progressed to flexibility in structure (XML and XSLT), moved to disconnecting the page model from server calls (AJAX) and now unbundles the page structure from application navigation. This is a profound change in the way the Web works.

The potential for Single Page Apps arguably is that they can provide an experience very much analogous to native mobile and desktop apps anywhere where a web-browser with Javascript can be executed.

While there are many positives to the architecture, there are also some negatives (see here for a great contrarian view) -- at least in the short term. SPA Apps are still hard to develop: State needs to replace page location, and this requires extensive thought at development time. Javascript performance is also poor in older browsers and some features such as HMTL5 History API are not available (eg in Microsoft Internet Explorer not before v10), meaning workarounds are needed and performance may be slow in some browsers. Conventional Web Error codes such as 404s, 503s, etc. are also not automatically handled and need to be explicitly handled in the App code. The boost in interface flexibility while positive, also makes it much easier to break standard web navigation conventions and potentially confusing users -- so Web UI design may need to evolve a new set of conventions.

It is early days for this Web trend, but look out for Single Page Apps, APIs, and Javascript as a combination that will begin to become the norm for many modern applications.

So again, when we're asked, "HTML5 of App+API?", we say "Both" -- the two worlds are converging faster than ever before and Single Page Apps are a big piece of the puzzle. It also seems likely that as issues with navigation, history, and SEO impacts are resolved Single Page Apps might also start appearing for content heavy sites -- leading to an "App-ification" of even more of the Web.

[Zendesk's original post gives an intro as to why Single Page Apps might be worth considering for your App, and there's a great comparison of some of the leading frameworks on Steve Andersons's Blog.]

[Illustration by Hallie Bateman]

Original URL:
https://pando.com/2012/12/06/html-javascript-and-the-app-ification-of-the-web/