Platform Cookies

The new web architecture of applications cuts out some technologies built into the browser, upon which many web stacks rely. Perhaps most importantly, browser cookies used to store information about a user's interaction with the application stack are no longer available, since the consumer of the application's endpoint is not a browser but the Facebook Platform.

At first glance, sending cookies from the browser along with the request to the application stack might appear to be a good solution. However, the domain of these cookies is then "*http://facebook.com*", when, in fact, the cookie information pertains to the experience provided by the application domain.

The solution? Endow Facebook with the role of the browser, by duplicating this cookie functionality within Facebook's own stores. If an application's FBML service sends back headers attempting to set a browser cookie, Facebook simply stores this cookie information keyed on the (`user`, `application_id`) pair. Facebook then "recreates" these cookies as a browser would when sending subsequent requests to this application stack by this user.

This solution is simple and requires the developer to change very little of his assumptions when moving his HTML stack over to the FBML service role. Note that this information cannot be used when a user decides to navigate to an HTML stack that this application may provide. On the other hand, it can be useful to separate a user's application experience on Facebook from her experience on the application's HTML site.

FBJS

When the application stack is consumed as an FBML service rather than directly by the user's browser, Facebook has no opportunity to execute the browser-side script. Directly returning this developer content untouched (an insufficient solution, as presented at the beginning of the FBML section) could solve this, yet it violates the Facebook-imposed constraints on the display experience. For instance, Facebook does not want `onload` events shooting out pop-up windows when a user's profile page loads. However, restricting all JavaScript precludes much useful functionality, such as Ajax or dynamically manipulating page content without reloading.

Instead, FBML interprets the contents of developer-provided `<script>` trees and other page elements with these constraints in mind. On top of that, Facebook provides JavaScript libraries to make common scenarios easy yet controlled. Together, these modifications constitute Facebook's Platform JavaScript emulation suite, called FBJS, which makes applications dynamic yet safe by:

- Rewriting FBML attributes to enforce virtual document scope
- Deferring active script content until a user initiates action on the page or element
- Providing Facebook libraries to implement common script scenarios in a controlled way