The best mobile Ajax application ...

is the one that's never written

A position paper for the W3C/OpenAjax Workshop on Mobile Ajax
by Mark Baker of Coactus Consulting

Hypertext inventor Ted Nelson talked about "the two Gods of literature"[1], publisher and reader, and the battle between them in determining what the reader ultimately consumes. Traditional Web programming with HTML and CSS (no script) provides effectively equal power to each God, permitting the publisher to provide the content together with declarations about how that content could be presented, but enabling the reader to instruct its agent to ignore and/or extend those declarations as it sees fit. Scripting, on the other hand, puts all the power in the hands of the publisher, providing the reader two options; either execute it to be able to consume the content it contains, or don't and don't.

The "reader power" afforded by declarative programming is especially important for mobile use for at least two reasons. First, it allows the user agent to repurpose the content so it can be presented to the user in the context of the particular nuances of the input and output methods of the device. Second, it allows the user agent to be shipped with code which is optimized for the processing of the predefined markup language, where "optimized" can be along any dimension: speed, size, battery consumption, etc..

Fortunately, there exists an approach which permits us to have the best of both worlds. If we factored out the commonly used script components, and extended HTML so that those components could be instantiated declaritively, then we could use declarative programming for these common components but continue to use script for the less common cases. As an example, "drag and drop" is a reasonably common feature implemented by scripts, but could easily be accomplished via a declarative approach [2].

One downside of this hybrid approach is that standardization of the extensions - and deployment of software which uses them - can take a very long time. However it is not as big a problem as it first seems. Somewhat ironically, we can use script to help us incrementally deploy declarative content, by associating that content with script which interprets and processes the extension tags when browsers don't support them (see [2] again for an example of this).

Some Javascript toolkits already provide for a somewhat similar approach, at least regarding the binding of HTML extensions to script. Dojo includes "Dijits" (Dojo widgets) which permits, for example, HTML forms to be extended with attributes whose value explicitly references a Dojo-specific scripted TextBox widget. Once the Dojo libraries are linked in, the extended processing occurs. All Dijits seem to be missing is to acknowledge the value of standardizing their HTML extensions.

HTML 5 (ne a bunch of WHAT WG specs) can also be seen in a new light when considering its role in this proposed approach. It defines (amoungst other things) a number of extensions that aim to do in

a declarative manner what is currently done with script. For example, in-browser form validation. The approach outlined here also suggests that there would be value in developing a script which could process the HTML 5 extensions for HTML 4 browsers.

Note: see also the TAG's view on declarative vs. imperative, in their Rule of Least Power finding[3].

[1] Theodore Nelson. The Future of Information: Ideas, Connections and the Gods of Electronic Literature. ASCII Corp, 1997. Unpublished.

[2] http://www.markbaker.ca/blog/2006/07/21/declarative-drag-and-drop/

[3] http://www.w3.org/2001/tag/doc/leastPower.html