Goals for Convergence of Installable Web App Technologies

W3C Workshop on The Future of Off-line Web Applications

5 November 2011, Santa Clara, CA, USA

Submitted by: AT&T

Contributors: Bryan Sullivan
Dan Druta

1. Introduction

While the title of the workshop is "The Future of Off-line Web Applications", we believe W3C can more usefully focus on this as "The Future of Installable Web Applications". Thus our focus in this paper, while aligned with the discussion scope of the Call for Participation, takes a broader view of the subject, as offline use cases are one aspect of the controlled persistence ("installation") of Web app content on user devices, whether initiated through user action (e.g. download of a widget package), or browsing to a website which references an AppCache manifest. Also, just as important as offline use cases are use cases in which the Web app is actually connected most of the time, but some (even all) app content is locally stored, with all the related advantages (and challenges) of local content storage.

We have a connected world in which most knowledge and information is moving into the cloud. Even so, many devices are still occasionally disconnected from networked sources of content and apps. The current approaches to supporting Web app use when disconnected are substantially different, with resulting interoperability limitations that are inhibiting growth of offline-capable Web apps. Since most of the same solutions are used for installed Web apps (i.e. apps that are specifically intended to be persistent on the user's device), the same limitations are also inhibiting Web apps from achieving their potential of equality with native apps, as user-installable apps. A key goal of ours is thus to consider and harmonize the objectives, approaches, and capabilities of supporting installable Web applications via existing solutions, e.g.

- W3C Widgets
- HTML5's AppCache API
- Vendor-specific packaging/manifests, e.g. as used by Google Chrome Web Apps and Firefox Open Web Apps.

With good progress toward HTML5 support in most browsers, the increased capabilities available to Web applications are helping the open Web platform to expand developer options beyond native app ecosystems.   Once a Web app is installed, we believe the user experience can achieve complete transparency as to the nature of the application being used (e.g. native or Web), while also being enhanced by the expanded options that the Web provides for application discovery and distribution. Addressing installable Web apps simply from a packaging and Web user agent processing perspective however is likely to leave significant issues unresolved, and also inhibit adoption as service providers struggle to consistently integrate these technologies into existing app development and distribution ecosystems. We thus need to consider the role and use of these technologies in the end-to-end app lifecycle, not just from a Web user agent perspective.

Lastly, current supporting technologies for installable Web apps and offline use cases need further focus, to resolve known issues and limitations being uncovered through implementation experience.

2. Achieving Convergence in Packaging

This is an important topic for any developer, initiative, or service provider that is trying to build a business based upon installable Web app technologies, independent of the Web user agent in use. Eliminating the existing fragmentation is necessary before installable Web apps can achieve the potential for broad deployment and scalability that is their chief differentiator, compared with native apps. W3C should thus undertake a survey of the objectives, approaches, and capabilities of existing installable Web app packaging and manifest schemas, and the related Web user agent processing requirements, as a preface to harmonized requirements and recommendations.

Also, where extensions to the W3C Widgets manifest schema have been defined by implementations, the implementers should help W3C consider the rationale for these extensions, as they may need to be included in a harmonized solution.

3. Key Capabilities across the Web App Lifecycle

While caching applies for a Web application discovered while browsing, we need to be able to address scenarios where the user has received a packaged Web application via a media card, email or any other form of binary storage medium. The expectation in this case should be that the Web application could work right away, even in a disconnected environment.

Key aspects we need to consider when defining a framework for these Web applications are related to:

- Development: the developer should be able to specify if the app supports disconnected mode or not

- Distribution: different forms of distribution should be supported, with related questions, e.g. what gets distributed, e.g. a simple URL, manifest file, full app package…
- First time execution: depending on the distribution mechanism, first time execution may require some form of additional download
- If caching is involved, what are the triggers for caching refresh and who are the actors involved, including scenarios where user requested update and developer update push are supported
- Existing Web applications have little need for versioning, but a disconnected application has more complex dependencies that may require proper versioning, e.g. to enable sync with its source, or use/migration of previously stored content upon app update
- With the proliferation of malware into the Web it is important to properly identify and trust the origin and integrity of a Web application and have mechanisms in place to disable the app if proven to be malicious or compromised
- Once the application is obsolete or the user decides to stop using it, cleanup processes have to be defined, and allow developer and user choice as to what happens with any local data, etc.

4. Addressing Limitations of Supporting Technologies

Implementation and app development experience has uncovered some limitations of key Web app supporting technologies. These limitations need solutions, or at least developer guidelines for how to address the related challenges when developing applications. Some known limitations include (not an exhaustive list):

- Widget support for origin-dependent features: Web apps such as widgets, that have no Web server based origin, cannot participate in Web-based authentication schemes such as OAuth without extensions to the Web runtime environment. Such capabilities are needed for access to many Web-based services. Similarly, other features that were designed around a typical Web origin (e.g. the window.opener() interface) are likely to be unusable with widget-based Web apps. W3C should address these limitations, so that widgets can use the same Web service design techniques as normal Web server based Web apps.
- Web Storage: the simplicity and functionality of the localstorage API for Web app controlled persistent storage of string data makes this API a key supporting feature of installable Web apps. However, the potential for storage access collision between apps of the same origin, and the

related lack of a storage mutex feature, has caused some discussion of dropping the localstorage feature. This would be unfortunate, since as noted above this is a very simple and useful feature, compared to the complexity of defining and accessing structured databases such as defined by Web SQL and Indexed DB. W3C should either address the limitations in the localstorage API, or produce developer guidelines that clarify how to avoid storage collisions (e.g. by uniquely prefixing storage keys).

- Server-Sent Events: as noted in the current draft of Server-Sent Events, some valuable use cases for server-sent events cannot be addressed by the current API, e.g. supporting delivery of events when offline, or when maintaining a data connection is not justified due to a low rate of event delivery, but it is desirable to retain the ability for servers to send adhoc events. Following an HTML working group discussion led by AT&T in the 2009 TPAC, use of connectionless bearers such as SMS and OMA Push as potential approaches to these use cases was added as an informative section in Server-Sent Events. Led by AT&T, these bindings for the EventSource API are currently being discussed in the WAC and a prototype is being developed. We would welcome the opportunity to discuss the binding approach, e.g. as an extension or new version of the Server-Sent Events specification, or a new API specific to connectionless Server-Sent Events delivery.

5. Conclusion

The Web as we know it is changing. To fulfill expectations for the Web as a first-class application platform supporting a wide diversity of devices and use contexts, users must be provided with a Web app experience rivaling the simplicity, transparency and reliability provided by native applications. In order to achieve that goal in diverse devices such as TVs, cars, household appliances, and mobile devices, we must consider the end-to-end lifecycle of Web apps, and address use cases involving Web app installation, offline use, and intermittent connectivity.