A Brand New Offline Web
Identifying and Mitigating Hurdles for Developers

Magnus Olsson, Ericsson AB, magnus.olsson@ericsson.com
Niklas Widell, Ericsson AB, niklas.widell@ericsson.com

Position Paper for W3C Workshop on The Future of Off-line Web Applications
November 5th, 2011, Redwood City, CA, USA

Abstract
This is position paper intended to present visions and ideas from Ericsson in the area of "Off-line" Web Applications. It describes a number of realistic scenarios where current solutions are insufficient to provide robust service to users. Ericsson sees a great value in web applications in general, and considers the capacity to robustly address offline scenarios of high value.

Introduction
End user experience can be significantly impaired if connectivity is not sufficiently responsive in terms of bandwidth and latency.

This used to be the mantra that we all shared in the early days of mobile web. "Always Best Connected" helped us to drive the expectations towards faster and more responsive radio layer technology. Eventually hardware started to become more "mobile friendly" as well and web browsers capable of rendering web content with more tasteful fidelity became available.

Yet hardware and connectivity are only two aspects of the life cycle of a web application that by nature should not be thought of as constrained to reside on a single device. The continuos evolution of web technology has brought us to the point where web applications are (almost) functionally interchangeable with native applications. As the web is pervasive across many domains, from PC to mobile to Set-top-box, it thus offers great opportunities for writing portable applications, with very natural integration into server side.

The benefits of web apps as highly portable highly functional entities motivates effort to improve and develop both technologies and tools to solve offline issues. In order to reach a common understanding we have elected to outline a set of important aspects that needs to be taken into account and - either mitigated or "designed for" but not to be ignored.

Volatility of User data
User data is important, no matter how you think of it, anything that is created by an end user, be it a note on a "remember the milk list" or a dissertation text entered on a web form, it carries a value to an end-user and any consequences of loosing that data must not be under estimated. We have outlined a few scenarios that is important to consider;

- Privacy and Integrity
- Device Malfunction
- Storage Exhaustion

Privacy and Integrity
User data should not be disseminated to applications unless it can be given away securely and that the act is somehow previously understood by the end-user or is otherwise made obvious at the instance of an API call.
It can be argued for the sake of user experience that handing over user data should also be related to something that will give a real and noticeable value in return (albeit the reward may be at some later point into the future).

For web applications that remain on the device this leads to the question when and how security permissions are to be requested, at the point of installation or when the actual feature is used, or some mixture in between. This is further complicated by the fact that the web application might cache the result from a secure api call for later use without the knowledge of the user. Studies (such as the recent thesis study "The Brand of Security" available on Ericsson Labs) show that is complicated to communicate security decisions to users, allowing the information to be cached locally under uncertain terms increases the complexity of those decisions.

Paying careful attention to state of the art application stores and how they obtain user consent give us a clue that applications tend to request full access to sensitive user data without providing any good motivation or obvious value. Therefore the de-coupling of installation of a web application from the actual granting of access rights, need to be re-visited.

Device malfunction and data robustness

User data maintained "off the web" will be stored in many places, hopefully that also imply some measure of redundancy by chance at least and by design at best. The possibility of device data storage becoming corrupt due to malfunction, interrupted communication or developer error needs to be considered. There is likely a number of methods that can be used by a developer to mitigate the risk of having data becoming corrupt. For instance, since offline web applications by their very nature are intended by design to be reliant on caching mechanisms whose behavior is not always fully transparent, good tools are needed to test applications

However as more services and household applications move into the cloud and data is distributed on multiple screens (and islands of storage) developers also face more challenges trying to keep this data in synch. Providing developers with good tools to maintain consistency of data across devices and sessions may prove to be vital in order to help developers provide a sense of trust to end-users.

Engaging with a healthcare application /service on a mobile device must not depend on having a nuclear battery to avoid loosing any vital sensor data

Storage exhaustion

So what if all is fine and we already solved the case of keeping our data secure and consistent - *surely we have covered all our bases then?*
Well that might have been true if we would keep our fixed "umbilical" wires connecting us to an NFS storage server with a logical drive size only limited to the speed of adding more virtual disk space! In reality there is a fixed size on most of the known semi-connected devices.

This matters even more as end-users tend to download more and more data for offline use not simply accepting it to become a collateral damage of a cache policy. Developers may need our support here in order to discover and implement better mechanisms to provide applications with predictable storage area or simply being able to detect if the available space is adequate or at least possible to re-size.

This also ties into life-cycle issues for applications. For instance, it is possible to have a way to allow applications with locally saved data push that data to the site the application came from, so that it is not lost when the local storage is emptied. This is to ensure that, as local storage use becomes more used, and the contained data more valuable, valued data belonging to one application can be controlled when the user cleans out caches.

Storage Exhaustion is a pain in existing mobile devices supporting download of native applications hogging memory faster than a user can swipe the application back to the trash can. Lets try to avoid turning web applications into "memory sponges".

Cost of User Data

Assuming a device is always connected goes from being a blessing to being a menace at the instance that the end-user leave their home base (network) e.g. roaming or visiting non-free WLAN access. Typically the roaming fee (e.g. as required by visiting network and pushed forward by home operator on top of a fixed rate plan) are very costly and need to be carefully managed. Allowing a web application to detect roaming (which is supported in cache manifest) provide for a change of working mode such as avoiding automatic refresh of data store.

However cutting the cord might not be what the end-user expect. If the additional cost could be absorbed as part of the application itself then it might also be possible to allow the application (or developer) to determine and manage (using QoS) how the (now costly) bits of data could be used. Finding out an appropriate mechanism to make a web application aware of data plans and not only connection on or off might be useful to consider.

Another example being an ability for web applications (and cloud services) to leverage alternative data connections to lower the cost for the user, and to be able to have some flexibility when it comes to handling what is to be cached.

State of the Art

- HTML5 generation of Web Browsers with support for a multitude of different types of local storage (key-value and regular databases)

- Cloud based Application Stores - Perfectly well adapted to serve content to many different screens (or devices) and even persist application state between devices and sessions e.g. web based eBook readers.

- Web and Native Hybrids - combining web content with a native front end that is (by nature) capable to leverage more advanced (and lower layer) features (such as radio layer management *APIs*) of mobile devices.

Conclusions

Success of offline web applications is not only how to mimic (or emulate) any existing native application (like an agitated specie of chicken using Java script), they must also address issues that are taken for granted in native app development and use. The Web is by nature agnostic to classes of devices as long as a device can still faithfully handle markup, logic and style along with ability to render and encode various media resources.

We should leverage that inherent capability and make sure that going offline with the web is not permanently cementing web applications in a second rate solution destined to be clumsily dressed up as a native application. Web applications need to be free to roam and any additional cost should also be integrated with the ticket to go online on the broadband highway.

Some aspects to remember when going offline;

- Secure and Integrity protected user data
- Device Malfunction and ability to synch data
- Storage Exhaustion - clever management of storage space
- Cost of User Data - QoS shades of grey and new connections

Offline and Cloud are two sides of the same coin, a good strategy to leverage capabilities in servers and devices are needed in order to sustain a vital pipe to the end-user.

A good an efficient offline aware "web stack" will be key to accelerate more mobile content and enabling - a Brand New Offline Web.

References

W3C Workshop on The Future of Off-line Web Applications
http://www.w3.org/2011/web-apps-ws/
Thesis study "The Brand of Security" on Ericsson Labs.
https://labs.ericsson.com/developer-community/blog/omg-camera-still