HTML5Apps

Deliverable D1.2

Standardisation Report 2

Project	
Grant Agreement number	611327
Project acronym:	HTML5Apps
Project title:	HTML5 for Apps: Closing the Gaps
Funding Scheme:	Coordination & Support Action
Date of latest version of Annex I against which the assessment will be made:	May 24, 2013
Document	
Deliverable number:	D1.2
Deliverable title	Standardisation Report 2
Contractual Date of Delivery:	M24
Actual Date of Delivery:	09 November 2015
Author (s):	Dominique Hazael-Massieux Dr. Dave Raggett
Reviewer (s):	Dr. Philipp Hoschka
Work package no.:	1
Work package title:	WebOS APIs
Work package leader:	Dr. Dave Raggett
Distribution:	PU
Version/Revision:	1.0
Draft/Final:	Final
Total number of pages (including cover):	30
Keywords:	Web page, scripting, APIs, permissions, trust, privacy

Disclaimer

This document contains description of the HTML5Apps project work and findings.

The authors of this document have taken any available measure in order for its content to be accurate, consistent and lawful. However, neither the project consortium as a whole nor the individual partners that implicitly or explicitly participated in the creation and publication of this document hold any responsibility for actions that might occur as a result of using its content.

This publication has been produced with the assistance of the European Union. The content of this publication is the sole responsibility of the HTML5Apps consortium and can in no way be taken to reflect the views of the European Union.

The European Union is established in accordance with the Treaty on European Union (Maastricht). There are currently 27 Member States of the Union. It is based on the European Communities and the member states cooperation in the fields of Common Foreign and Security Policy and Justice and Home Affairs. The five main institutions of the European Union are the European Parliament, the Council of Ministers, the European Commission, the Court of Justice and the Court of Auditors. (http://europa.eu.int/)

HTML5Apps is a project funded in part by the European Union.

Document History

Version	Date	Comment
0.5	29/09/2015	First draft report
0.6	12/10/2015	Updated draft report after first internal review with mainly editorial changes
0.7	5/11/2015	Updated draft report after second internal review using new outline
1.0	06/11/2015	Final version

.

TABLE OF CONTENTS

1. INTRODUCTION

As per the HTML5Apps description of work, this deliverable describes “achievements in terms of standardizing WebOS APIs (including standardization documents)”. Standardisation documents are included via reference to a URL rather than copied into this document.

HTML5 has a number of attractive properties for app development over native app technologies, including its open ecosystem, a large size of Web developers, technology openness and its cross-device capabilities.

However, at the outset of the HTML5Apps project, HTML5 had a number of limitations compared to native apps, including in particular its limited access to some “operating system” functionalities needed for developing “apps”.

The HTML5Apps project was set up to close the gap between HTML5 apps and native apps in this area by helping to define and standardize a runtime environment, a security model, and associated APIs for building Web applications with comparable capabilities to native applications.

HTML5Apps helped to make major progress in this area: by supporting the standardization of

- an *app manifest specification* that enables a centralized place to put metadata associated with a web application
- a more clearly delineated security model with the concept of secure contexts and through a *permissions API specification*
- an execution model for web applications that make them more like Operating System processes by letting them live beyond the lifetime of a tab or of the browser itself, using the *service worker specification*
- a *Web Bluetooth specification* and a *Web NFC specification*, based on a better understood model on how to bring non-Web-compatible security models to Web browsers, namely that these "legacy" protocols have to opt-in to Web compatibility
- a more generic approach to how to build sensor APIs for the web with the work on a *generic Sensor API specification*

In addition, to further the adoption of HTML5 standards for OS-level functionality in applications developed using Web technology, HTML5Apps developed

- an open-source implementation of the W3C Battery API for Apache Cordova
- an open-source implementation of the W3C Notifications API for Apache Cordova

This report is structured as follows: In Section 2, we report on the approach taken for standardization of the execution and security model. In Section 3, we report on the

standardization of individual APIs. In Section 4, we describe the project's contributions to the Apache Cordova project. In Section 5, we give an overview of the changes to specifications during the HTML5Apps project and the value added by the project.

Please note that this document includes hypertext links to background materials including draft specifications. These can be followed when viewing the electronic version of this document.

2. Web Application Execution and Security Model Standards

During the first year of the project, the work on the Runtime and Security Model for Web applications led to the realization that three main technical pieces were needed to improve the way Web applications get developed and adopted by end-users:

- A model to manage the lifecycle of Web applications, especially in the offline context and in the case where the application is not running in the foreground,
- A mechanism for Web applications to be exposed to end-users on par with what end users get with native applications,
- A more cohesive approach to managing permissions, that is easier to understand for end-users and to control for developers (as the work described in the deliverable Future Standards Report 1 D2.2[1] demonstrated)

During the second year of the project, significant progress on these three components has been achieved, with the development and early adoptions of the Service Worker specification, the App manifest specification and the Permissions API. Figure 2.1 illustrates how these specifications have enabled the execution and security model of HTML5 applications to become competitive with the functionality available in native platforms.

[1] https://html5appsproject.files.wordpress.com/2015/02/d2-2-html5apps-futurestandardsreport.pdf

Execution Model Feature	iOS	Android	Web before HTML5Apps	Web after HTML5Apps	Specifications
Installation	From app store	From app store	Inconsistent and hard to do for the user	Consistent medata for UI integration Simplified user interaction	Web App Manifest
Offline operations	Default behavior	Default behavior	Not possible in most cases	Simplified model for interacting with and without network	Service Workers
Background operations	Available via SDK	Available via SDK	Impossible	Support for reacting to Push notifications, framework in place for additional operations (geofencing, data synchronization)	Service Workers, Push API
Get additional access to privileged features	Can determine if user prompt is needed	Until Android M, only available at install or update. Since Android M, can determine if user prompt if needed	Can only prompt the user (impossible to determine if permission is available)	Can determine if user prompt is needed	Permissions API

Figure 2.1: *Progress on HTML5 Execution and Security Model during HTML5Apps*

2.1. Service Workers Specification

Mobile users often have to put up with a loss of access to the Internet, e.g. no local WiFi access and a very weak cellular signal, especially in rural areas and ferry crossings when some distance from land. On aeroplanes, everyone is used to having to switch their phone or tablet into flight safe mode. Some airlines are now introducing in flight WiFi access, but this is still quite expensive. As a result, it is common to want to use applications off line.

The Service Workers specification describes a method that enables applications to take advantage of persistent background processing, including hooks to enable bootstrapping of web applications while offline. The core of this system is an event-driven Web Worker, which responds to events dispatched from documents and other sources. A system for managing installation, versions, and upgrades is provided. The service worker is a generic entry point for event-driven background processing in the Web Platform that is extensible by other specifications.

This is critical to enabling web applications to compete with native applications..

In particular, two of the features that were most frequently requested by European SMEs developing HTML5-based applications in the interviews that HTML5Apps conducted (see deliverable D2.3):

- Offline usage
- The ability to use remote "Push" notifications to send timely alerts to the user, independently of whether the app is currently running in an open browser or not

At this time, the Service Workers specification is expected to have as great an impact on how Web applications are built,as the AJAX approach had 10 years ago. Feedback from early adopters in the developers community seem to confirm that, Service Workers represent a major architectural improvement to the Web,.

Moreover, Service Workers can be easily deployed, since they can be added to existing applications as a "progressive enhancement" feature, i.e. one that will improve the experience of users of browsers which support it, without breaking the experience of less advanced browsers.

The most recent public draft is from 25 June 2015. This API is now in a last call for comments in preparation for release as a W3C Candidate Recommendation.

Service Worker is implemented on Google Chrome since 2015, and an implementation in Firefox is expected by the end of 2015. Microsoft is considering an implementation as well.

Link to Standardisation document

http://www.w3.org/TR/service-workers/

2.2. App Manifest Specification

The app manifest specification provides the ability to launch Web applications from app icons in the same way as native apps. When you visit a web sit,e you can be offered the means to install an app launcher on your home screen. This is based on the Web page providing a machine interpretable link to the app manifest.

The specification defines a JSON-based manifest that provides developers with a centralized place to put metadata associated with a web application. This includes, but is not limited to, the web application's name, links to icons, as well as the preferred URL to open when a user launches the web application.

The manifest also allows developers to declare a default orientation for their web application, as well as providing the ability to set the display mode for the application (e.g., in fullscreen).

Additionally, the manifest allows a developer to "scope" a web application to a URL. This restricts the URLs to which the manifest is applied and provides a means to "deep link" into a web application from other applications.

Using this metadata, user agents can provide developers with means to create user experiences that are more comparable to that of a native application.

The most recent public draft is from 21 September 2015.

App manifest is implemented both by Chrome and Firefox, and under consideration by Microsoft.

Link to Standardisation document

http://www.w3.org/TR/appmanifest/

2.3. Permissions API Specification

Privacy and security are key considerations. Native apps are subject to vetting before being released on app stores, e.g. Apple’s “App Store” and Google’s “Play”. Web browsers by contrast have to deal with un-vetted applications from anywhere on the Internet.

Browsers ask users for permission to grant access to applications for capabilities that are deemed risky in respect to access to personal data. To date, W3C has handled this separately for the API for each capability. The permissions API offers a more generic approach as a basis for developers to customise the user experience.

The permissions API specification enables apps to test the permission status for a given capability, e.g. has permission been granted, declined or will the user be prompted for a decision when the capability is next invoked. This is something that emerged from a desire by app developers to provide a better user experience, e.g. not to invoke capabilities that the user has previously declined, or to provide a justification for the user to grant a permission prior to the browser prompting him/her for a decision

Permissions is available in Google Chrome, and should be implemented in Firefox in 2016.

Link to Standardisation document

http://www.w3.org/TR/permissions/

3. APPLICATION PROGRAMMING INTERFACES (APIS) STANDARDS

During the first year of the project, the HTML5Apps project staff supported the System Applications Working Group in its work dedicated to APIs running outside the browser to facilitate the development of APIs whose security model was not compatible with the browser model.

Given the significant progress on the security and execution model in browsers described in Section 2, the HTML5Apps project refocused its efforts in the second year of the project on APIs with broader appeal in the developers community and that were compatible with the browser security model. That change of focus in particular led the project to close the System Applications Working Group in August 2015.

Based on the work on priorities for future API standards in the "future standards" deliverable D2.2, in this reporting period, in this reporting period,the HTML5Apps project focused its efforts on five important APIs for HTML5 application developers: wake lock, audio output, a generic approach to manage data from sensors, as well as specific APIs to cater for the increase usage of both Bluetooth and NFC technologies.

3.1. Wake Lock API

Deliverable 2.2 of the previous reporting period identified the need to control if and when a mobile device should turn off its screen automatically – while that behavior is in general useful to save battery, it is also problematic in cases where the user is using an app or a service that doesn't require touch screen interactions (e.g. watching a video).

Based on the use cases and requirements the project helped gather in its first year, standardization work started on an API to bring control on this "wake lock" to browsers.

The most recent published Working Draft within the reporting period is from 4 September 2015.

Link to Standardization document

http://www.w3.org/TR/wake-lock/

3.2. Audio Output API

Also identified in deliverable 2.2, the ability to control on which audio output devices a given sound is played is important to a number of applications with more advanced sound usage; for instance, a teleconferencing system (as enabled by the WebRTC technology) will likely want to distinguish where the "ring" sound plays from where the voice of the participants should be heard.

Standardization on an Audio Output Devices API was thus started to fulfill that need, while preserving the users from rogue apps who might abuse of this e.g. to play ads on the loudspeaker.

The first published Working Draft of that API was published on 10 February 2015.

Link to Standardization document

http://www.w3.org/TR/2015/WD-audio-output-20150210/

3.3. Generic Sensor API

This work was started as a way to provide a unified approach to developing APIs for individual sensors.

This specification defines a framework for exposing sensor data to the Open Web Platform in a consistent way. It does so by defining a blueprint for writing specifications of concrete sensors along with an abstract Sensor interface that can be extended to accommodate different sensor types.

The most recent editor's draft within the reporting period is from 28 August 2015.

Link to Standardisation document

https://w3c.github.io/sensors/

3.4. Bluetooth API

Bluetooth Low Energy (BLE), a profile of the Bluetooth technology optimized for its low power consumption, is a key enabler to connect new devices (including wearables and health sensors) to mobile devices.

To bring access to these BLE devices from within Web applications, a Web Bluetooth Community Group was set up, and published first drafts of use cases and API specification during the operation of the project (27 July 2014), with several new drafts published since.

An alpha version of the Bluetooth API is in available in Chrome, and Firefox devolopers expressed interest in an implementation

Link to Standardisation document

https://webbluetoothcg.github.io/web-bluetooth/

3.5. NFC API

Another mechanism used to link mobile devices to their physical environment relies on Near-Field-Communications (NFC).

A Web NFC Community Group was set up to develop an API that grants access to data stored on NFC devices, based on an innovative model of security to ensure the compatibility of the NFC model with the specifics of the Web origin-based security model.

The NFC API has an experimental build for Chrome, and Firefox develpers showed interest in an implementation.

Link to Standardisation document

http://w3c.github.io/web-nfc/

4. OPEN SOURCE IMPLEMENTATION OF STANDARD APIS

While the progress enabled by the evolution of the security and execution model described in Section 2 paves the way for Web applications to take an increasing share of the initial user experience of users with a given mobile service, many developers will need to keep developing native applications.

Indeed, as illustrated in the European SMEs interviews conducted by the project (see deliverable 4.4), native apps are still required in the following cases:

- to supplement the features not yet available to Web browsers, including those described in section 2 that haven't been deployed
- to ensure visibility in the native application stores
- to satisfy specific marketing requirements of their customers.

Interestingly enough, HTML5 is widespread use today even for the development of native apps Hybrid application frameworks allow web developers to use HTML, JavaScript and CSS, etc. as a basis for creating native applications for platforms like iOS and Android.

To make sure HTML5 remains a good platform to bridge the Web and native world, , the HTML5Apps project invested resources in bringing APIs alignment with the most popular "hybrid " applications framework, Apache Cordova.

4.1. Apache Cordova

Following a recommendation by the European Commission (HTML5Apps should be "engaging industry and open source communities (e.g. building on initial approach to Apache Cordova)"), in the second year of the project, the HTML5Apps increased its collaboration and accelerated its impact for hybrid application developers.

During the first year of the project, the HTML5Apps staff established a cooperation with the maintainers of the popular hybrid application development framework, Cordova (also knows under its former name, Phonegap).

Indeed, having determined that the approach of mixing native and Web technologies to develop hybrid apps and that Cordova is the most widely used framework to do that, the HTML5Apps project worked with them to understand if and how much their Software Development Kit (SDK) aligned with the APIs that W3C standardises for the Web runtime engines on which it is built.

As explained in deliverable 2.2, the results of that investigation showed that there was clear interest in this alignment and early work in that direction was started during the first year of the project with the adaptation of Cordova2's vibration feature to the W3C standardised API.

The HTML5Apps project thus took a more pro-active role to push progress in this space, and contributed directly to the coding efforts required to bring this alignment.

4.2. Battery API

More specifically, the HTML5Apps project first developed, tested and documented a rewrite of the Cordova Battery plug-in to make it compatible with the W3C Battery API, a target that had been identified during the first year of the project.

Given Cordova's cross-platform nature, the HTML5Apps project made sure in particular to test that rewrite on several platforms (Android, FirefoxOS, Windows), taking into account that one of the values the API is expected to provide (remaining charging and discharging time) is not systematically available on some of the platforms.

Also, since the said API makes use of a new JavaScript construct (called Promises) that Cordova had not adopted yet in its SDK, the project also worked with the Cordova maintainers to determine and document how this construct should be imported as a module and provided via Cordova's dependency management system.

That rewrite was submitted to the Cordova project as a GitHub pull request3, which the project then advocated for adoption to the Cordova maintainers.

4.3. Notifications API

The current Cordova SDK provides a dialog plug-in that allows developers to display alert messages and confirmation pop-up. Since mobile operating systems provide more refined ways for attracting their users' attention, W3C is standardising an API that enables developers to make use of the much more friendly notification of these systems.

The HTML5Apps project thus developed, tested and documented a possible replacement of the existing dialog plug-in by one that would make use of the W3C notification API.

---

2 https://html5appsproject.files.wordpress.com/2015/02/d2-2-html5apps-futurestandardsreport.pdf

3 https://github.com/apache/cordova-plugin-battery-status/pull/24

After discussing this proposed rewrite with the Cordova maintainers, it appeared that the feature the Cordova community is most interested in is the ability of receiving so-called “push” notifications (i.e. notifications sent by a server component even if the client application is not running). W3C has also ongoing standardisation in this space, and third-party developers have started providing Cordova plug-ins which provide such a feature based on the W3C API.

These conversations, and the recurrent difficulty in identifying which W3C API were available via which Cordova plug-in (core or third-party) led the HTML5Apps project to document and advertise to hybrid app developers what solutions are available to them to develop hybrid apps using the W3C notification and push API, and more broadly to provide an authoritative source of Cordova plug-ins (core or third-party) that guide developers wanting to stay as close as possible to W3C standard APIs. This is described further in the following section.

4.4. List of W3C-compliant Cordova plug-ins

Based on its experience with researching and discussing Cordova plug-ins that align with W3C APIs, the HTML5Apps project continued its research on which other plug-ins might already exist that provide a satisfactory equivalent of W3C APIs to hybrid apps developers.

The project then started and populated a GitHub repository that lists for a number of W3C APIs the available Cordova plug-ins that provide an equivalent API to Cordova developers. The list of equivalence is reproduced below in its state as of the end of the project (September 20154):

W3C Specification	Cordova Plugin	Note
Background Synchronization	cordova-plugin-service-worker-background-sync	iOS only
Battery	Cordova core plugin	Patch submitted to align with W3C API
File	Cordova core plugin	This plugin is based on an early version of Filen API, the api does not match the current spec.
Geolocation	Cordova core plugin	

4 https://github.com/w3c/cordova-plugin-W3C-aligned

W3C Specification	Cordova Plugin	Note
Local Notification	cordova-plugin-service-worker-notifications	Basic Notification API for iOS and Android but Service Worker Functionality only in iOS
Service Worker	cordova-plugin-service-worker	iOS only
Screen Orientation	cordova-plugin-screen-orientation	This plugin is based on an early version of Screen Orientation API, the api does not match the current spec.
Vibration	Cordova core plugin	
Web RTC	cordova-plugin-iosrtc	iOS only

Overall, the progress achieved on improving two existing Cordova plug-ins as well as documenting which other third-party plug-ins align with W3C APIs is well on track to produce a lasting impact in the life of hybrid apps developers, reducing their needs to maintain multiple version of their HTML5 apps for in-browser and in-store distribution.

5. VALUE ADDED BY PROJECT TO STANDARDISATION PROCESS

Following its DoW (Description of Work) of for Task 1.2 of WP1 ("*In order to develop ... standards, members of the HTML5Apps project team will take the role of so-called "W3C team contacts*"), the project added value in the development of the relevant specifications by providing W3C team contacts for the standardization groups developing the specifications listed in Section 2.1.

A W3C team contact acts as the interface between the Group Chair ("Chair"), Group Members, and the W3C Team. Many of the team contact's tasks involve helping the Chair complete his or her roles, while others involve direct action from the Contact. The team contact role is largely one of communication. This involves becoming as aware as possible of the technical requirements and issues in the group, and simultaneously being aware of the general architecture of the Web as evolving in the other work of W3C.

In their function as W3C team contact, HTML5Apps staff played a critical role in progressing standardization work through both administrative and technical contributions.

Administrative contributions included

- The organization of face-to-face meetings (e.g. in San Jose, California in late October 2014).
- Organizing transition of documents through the W3C standardization process (e.g. relicensing process required for transitioning an initial NFC specification focused on Web runtimes into a community group and published the specification as a W3C Note)
- The HTML5Apps staff was also responsible for maintaining the infrastructure on which groups have conducted its work, including public home pages, a public mailing lists and a set of Wiki pages.

Technical contributions included

- For the *Web App Manifest specification*, the HTML5Apps project researched existing problems and solutions around "deep linking", i.e. the ability to integrate native and Web apps by creating links across the two ecosystems, which resulted in changes to the Web App Manifest specification
- For the *Permissions API*, the HTML5Apps project conducted during its first year research on the complexity and difficulty of managing permissions across APIs (as documented in deliverable 2.2). Its further efforts in promoting these results, accompanied by the discussions held at the permissions meeting organized by the project (as documented in deliverable 1.1) were key in scoping the problems solved by the Permissions API and served as trigger to the start of that work

- For the *Wake Lock API*, the HTML5Apps project contributed a first technical proposal to serve as a basis for the API; which provided useful insights as to how to integrate that API with the overall eventing model of the Web platform
- For the *Audio Output Devices API*, the HTML5Apps project contributed to the formal design of the API by providing "WebIDL" expertise to the discussions, worked on ensuring the design was compatible with the overall evolution of the HTML language (with which it integrates) and set up coordination meetings with other W3C groups impacted by that work (HTML Working Group, Audio Working Group and Second Screen Working Group).
- For the *Generic Sensor API*: the HTML5Apps project contributed on the gathering of experience of developing previous sensors APIs for the Web, as well as to the overall architecture on how such a generic sensor API should be built to enable concrete sensors APIs in the future.
- For the *Bluetooth LE API specification*: the HTML5Apps project started a review of how Bluetooth LE could apply to the Web and had supported the creation of a pre-standardization Community Group on the topic[5] in July 2014. The group has since made good progress in defining what a Bluetooth LE API would look like; the HTML5Apps project has been monitor that work closely and has brought logistical assistance to the group to ensure its progress and to facilitate its likely transition to formal standardization.

[5] http://www.w3.org/community/web-bluetooth/

6. CHANGES TO SPECIFICATIONS DURING HTML5APPS OPERATION

The table below provides update of the table presented in D1.1 stating the changes that have been made to relevant specifications during the operations of HTML5Apps , following up on a recommendation by the EU commission...

Relevant Specification	Changes made during HTML5Apps project
Run-time & Security	A major change during the project operation was that this specification (which already existed when the project started) was split into several parts, namely: • JSON-based manifest format - published as Working Draft on 17 December 2013 and ownership switched to another W3C Working Group. • AppURI for access to resources packaged as part of the app - published as Last Call Working Draft on 29 May 2014. • App Lifecycle for managing lifecycle of app and associated events with dependency on work on Service Workers, last draft 11 June 2014. The Run-time & Security specification was republished as a Working Group note on 6 August 2015.

Relevant Specification	Changes made during HTML5Apps project
Task Scheduler API	During the project operation, the name of this specification was changed from Alarm API to Task Scheduler to better reflect its scope, e.g. handling complications in managing time zones. Several new drafts were produced during the project operation. Major changes in this draft included addressing issues such as: • specifying behaviour when a scheduled task was missed because the clock jumped parsed the scheduled time, • what happens when alarms fire when device is off, • privacy and security issues, • potential need for recurring alarms, • updating the specification to use Service Workers. The Task Scheduler API specification was republished as a WG Note on 23 July 2015.
Contacts API	Several new drafts were published during the project operation These drafts included changes to details of use of promises (using Promise.then() instead of Promise.do()). The Contacts API specification was republished as a WG Note on 02 June 2015, under the permissive document license with a view to enabling it to form the basis for further work by others, e.g. in a W3C Community Group. The decision to adopt the permissive license followed a request by the editors and a call for comments from the Working Group.

Relevant Specification	Changes made during HTML5Apps project
Messaging API	Changes based upon implementation experience during project operation included: • better support for conversation objects, • clarifying how message id's are generated, ways to deal with SMS middleware that don't use the same segment size for all segments, • decoupling the storage interface for future-proofing. The Messaging API specification was republished as a WG Note on 02 June 2015, under the permissive document license with a view to enabling it to form the basis for further work by others, e.g. in a W3C Community Group. The decision to adopt the permissive license followed a request by the editors and a call for comments from the Working Group.
Telephony API	Several new drafts were published during the operation of the project Changes included modifications based upon feedback by telephony experts. The Telephony API specification was republished as a WG Note on 02 June 2015, under the permissive document license with a view to enabling it to form the basis for further work by others, e.g. in a W3C Community Group. The decision to adopt the permissive license followed a request by the editors and a call for comments from the Working Group.

Relevant Specification	Changes made during HTML5Apps project
TCP UDP Sockets API	Several new drafts were published during the operation of the project) involving substantial changes based upon feedback from browser vendors. Changes included: • separating socket creation from connection, • support for extended socket flags for enhanced control. The TCP UDP Sockets API specification was republished as a WG Note on 23 July 2015, under the permissive document license with a view to enabling it to form the basis for further work by others, e.g. in a W3C Community Group. The decision to adopt the permissive license followed a request by the editors and a call for comments from the Working Group.
WebOS Bluetooth API	First drafts of use cases and API specification published during the operation of the project (27 July 2014), several new drafts published since. Changes in updated drafts included: • new use cases (e.g. NFC handover), • prioritisation of use cases.

Relevant Specification	Changes made during HTML5Apps project
Application Manifest	This started life as part of the Run-time & Security specification and was split off and subsequently transferred at the request of the editors to the Web Applications WG where it continues to be actively worked on. This specification defines a JSON-based manifest that provides developers with a centralised place to put metadata associated with a web application. This includes, but is not limited to, the web application's name, links to icons, as well as the preferred URL to open when a user launches the web application. The manifest also allows developers to declare a default orientation for their web application, as well as providing the ability to set the display mode for the application (e.g., in fullscreen). Additionally, the manifest allows a developer to "scope" a web application to a URL. This restricts the URLs to which the manifest is applied and provides a means to "deep link" into a web application from other applications. Using this metadata, user agents can provide developers with means to create user experiences that are more comparable to that of a native application. The most recent public draft is from 15 August 2015.
Permissions API	This spun off from discussions around the need to adapt the application user experience based upon the status of permissions for particular APIs. The Permissions API allows a web application to be aware of the status of a given permission, to know whether it is granted, denied or if the user will be asked whether the permission should be granted. The most recent public draft is from 07 April 2015.

Relevant Specification	Changes made during HTML5Apps project
Service Workers	This work grew of of the realisation of the limitations of previous W3C work to support offline applications, and the need for developers to be able to customise how network requests are handled when the device is offline. This is critical to enabling web applications to compete with native applications. This specification describes a method that enables applications to take advantage of persistent background processing, including hooks to enable bootstrapping of web applications while offline. The core of this system is an event-driven Web Worker, which responds to events dispatched from documents and other sources. A system for managing installation, versions, and upgrades is provided. The service worker is a generic entry point for event-driven background processing in the Web Platform that is extensible by other specifications. The most recent public draft is from 25 June 2015. This API is now in a last call for comments in preparation for release as a W3C Candidate Recommendation.
Wake Lock API	This work was started in the second year of the project based on the needs identified during the first year of the project. It defines a mechanism for Web applications to request that the screen does not lock automatically, enabling use cases such as video watching. This specification was published as a First Public Working Draft in February 2015, and then published as an updated Working Draft on September 4, 2015.

Relevant Specification	Changes made during HTML5Apps project
Audio Output API	This work was started in the second year of the project based on the needs identified during the first year of the project. It enables Web applications developers to route a particular audio stream to a specific audio output device (e.g. speaker phone or headset). This specification was published as a First Public Working Draft in February 2015.
Generic Sensor API	This work was started as a way to provide a unified approach to developing APIs for individual sensors. This specification defines a framework for exposing sensor data to the Open Web Platform in a consistent way. It does so by defining a blueprint for writing specifications of concrete sensors along with an abstract Sensor interface that can be extended to accommodate different sensor types. The most recent editor's draft is from 28 August 2015.
Bluetooth API	This work was started at the end of the first year of the project, following the need identified earlier that year to incorporate external sensors and wearables in Web applications. The pre-standardization specification is updated regularly based on developers and implementation feedback gathered via its pre-release in the Chrome browser on Android.

Relevant Specification	Changes made during HTML5Apps project
WebNFC API	This work was started at the end of the first year of the project, based on the progress on the execution and security model for Web APIs accomplished during this first year. The pre-standardization specification is up-dated regularly.

7. SUMMARY

HTML5Apps helped to make HTML5 a more powerful platform for app development, diminishing the need to use native app technologies for development. This was achieved by supporting standardization of a number of key system-level functions, including

- an execution model for HTML5 apps (service workers)
- a mechanism to install HTML5 apps on devices (app manifest)
- a security model for HTML5 apps (permissions API)

HTML5Apps also supported standardization of five important APIs for HTML5 application developers: wake lock, audio output, a generic approach to manage data from sensors, as well as specific APIs to cater for the increase usage of both Bluetooth and NFC technologies.

The project moderated, guided and in some cases directly instigated the development of these standards.

In addition, HTML5Apps contributed two open source implementations of W3C standards to Apache Cordova:

- an open-source implementation of the W3C Battery API
- an open-source implementation of the W3C Notifications