Effective Use of HTML5 for Developing Offline Web Applications

K. Ram Sandilya[1], Ramidi Sudheer[2], Velagapudi Sreenivas[3], K V D Kiran[4]

IV/IV B.Tech, Department of CSE, K L University, Andhra Pradesh, India
[1]sandy.kambhampati@gmail.com
IV/IV B.Tech, Department of CSE, K L University, Andhra Pradesh, India
[2]sudheerfriend01@gmail.com
Asst Professor, Department of CSE, K L University, Andhra Pradesh, India
[3]velagapudi@kluniversity.in
Asst Professor, Department of CSE, K L University, Andhra Pradesh, India
[4] kiran_cse@kluniversity.in

Abstract[1][2]- One of the major constraints of web applications has always been connectivity. We imagined leveraging the browser to bring fully competent web applications to the desktop, but failed due to the lack of decent browser support. Although there were some caching techniques available before, they were never really designed with the intention of making web applications run completely offline, making them fragile and complex to set up. HTML5 tries to make up for this missing browser capability by introducing the offline application cache; a more reliable way to make web applications truly available even offline. With the offline capabilities and local persisted storage features, you can deliver the same rich user experiences online and offline that were previously available only in proprietary desktop application development frameworks

Key words – HTML5, Offline Browsing, Web application, Web DB.

I. INTRODUCTION

HTML 5 is the most recent version of Hyper Text Markup Language (HTML), a language proposed by Opera Software for Web (World Wide Web) content presentation.
HTML5 covers the features of HTML4, XHTML 1 and DOM2HTML. It is a formatting language that programmers and developers use to create documents on the Web. With a number of new elements, attributes, 2D and 3D graphics, video, audio elements, local Storage facility, local SQL database support and a lot more exciting features, HTML 5 has brought a monumental change in World Wide Web. HTML 5 is nothing but using shorthand for continuous innovation in a client-centered application with new tags on a general development framework with CSS3 and JavaScript. It supports both desktop deployment and mobile deployment. Smart phones like Apple iPhone, Google Android, and phones running Palm WebOS have gained huge popularity with HTML 5 based rich Web applications. HTML 5 contains a number of new and easily understood elements in various areas. Some of these are as follows:
HTML controls used in UI Multimedia file supports like video and audio Database support like local SQL database
A number of new Application Programming Interfaces (APIs)
These elements can be used in interactive websites with little customization to add an attractive effect and enhance the performance.
Apart from the above features, the beauty of HTML5 comes with the fact that it supports the development of Web applications that can be used in offline mode when there is not a prolonged and constant access to the internet due to usage policies like social networking APIs (such as Facebook, Twitter and so on).To store the data locally and to allow the applications to run offline, HTML5 provides three different APIs and these are:
Web storage: Can be used for basic local storage with key-value pairs
Offline storage: Can be used to save files for offline use Index DB: Supports relational database storage.

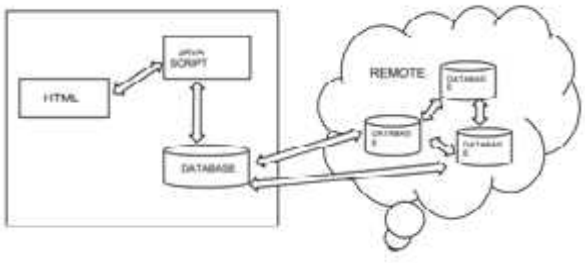

Fig 1. Storage architecture of a web application

There are primarily two offline capabilities in HTML5: application caching and offline storage (or "client-side storage"). The distinction is core application logic versus data. Application caching involves saving the application's core logic and user-interface. Offline storage is about

capturing specific data generated by the user, or resources the user has expressed interest in.

Web application capabilities are always been dependent on connectivity. At its simplest, an offline web application is a list of URLs — HTML, CSS, JavaScript, images, or any other kind of resource. The home page of the offline web application points to this list, called a manifest file, which is just a text file located elsewhere on the web server.

II. PRELUDE TO OFFLINE WEB APPLICATION API AND WEB STORAGE API

To give the user the experience of the seamless Web browsing, most of the browsers provide the facility of caching. But caching suffers from the following limitations:

- Developers do not have any control over the pages to be cached. Hence they cannot cache all the files of any specific application.
- It does not provide a reliable way for you to access pages while you are in offline mode (for example, Airplanes).

To overcome the above limitations, HTML5 comes up with the concept of Offline Web application API (App Cache) which provides the following features:[3]

- The user can use cloud functions on a mobile device, work offline with a locally deployed application on a local database, and share data with the rest of the cloud when going online again.
- App Cache caches only the specified pages that are included in the manifest file and hence it is reliable.
- With a higher degree of certainty, it gives the user the control of the way the Web applications should look when offline.

WEB STORAGE API

[4]To overcome the limitations of cookies, Web storage APIs are being introduced in HTML5 that can be used to store the data locally on the user's machine. This API allows the developers to store the data in the form of KEY-VALUE pairs that can be accessed by the Web applications through scripting. Web storage works exclusively with client side scripting where data can be transmitted to the server on requirement basis and a Web server cannot write the data to Web storage as well.

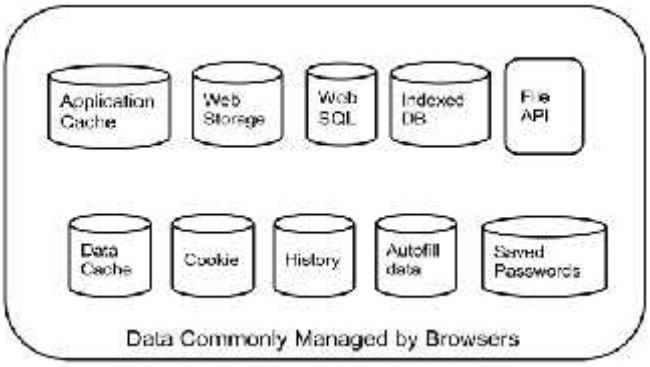

Fig 2. Data stored by a web browser

HTTP Cookies versus Webstorage API

- HTTP Cookies suffer from limited data storage (e.g. 4KB), whereas the data storage capacity of Webstorage API for most of the browsers is 5MB as per w3c specification with an exception to IE8 (and upwards) that supports up to 10MB.
- Cookies limit the accessibility of data to a certain domain name or a URL path where as Webstorage can limit the access to a certain domain name, or to domain TLD (like .org) or for the given user session.
- It is possible to iterate through all the key/value pairs in HTTP cookies, whereas in Webstorage, it is not possible to iterate through keys unless the key is known.
- Data stored in cookies are passed on by every request to the server, making it very slow and ineffective.
- However, the Webstorage API data is not passed on by every server request and used only when asked.
- Webstorage is more public then cookies. This is the reason why we need to take special precautions to ensure the security.

III. THE CACHE MANIFEST

[5]An offline web application revolves around a cache manifest file. The cache manifest in HTML5 is a software storage feature which provides the ability to access a web application even without a network connection. In order to bootstrap the process of downloading and caching these resources, you need to point to the manifest file, using a manifest attribute on your <html>element.

Web applications consist of web pages that need to be downloaded from a network. For this to happen there must be a network connection. However there are many instances when users cannot connect to a network due to reasons beyond their control. HTML5 provides the ability to access the web application even without a network connection using

the cache manifest.

File headers:

The cache manifest file consists of three section headers.

1. Explicit section with the header CACHE.
2. Online whitelist section with the header NETWORK.
3. Fallback section with the header FALLBACK.

Cache manifest file can be located anywhere on web server, but it must be served with the content type text/cache-manifest.

Example for Cache Manifest:

/test.css

/test.js

/test.png

Example for Cache Manifest Network:

/checking.cgi

CACHE: /test.css

/test.js

/test.png

Example for Cache Manifest Fallback:

/ /offline.html

NETWORK:

…

IV. THE FLOW OF EVENTS

[6]In offline web applications, the cache manifest, and the offline application cache ("appcache") are semi-magical terms. First, we have to consider flow of events. Specifically, DOM events. When browser visits a page that points to a cache manifest, it fires off a series of events on the window.applicationCache object.

1. As soon as the browser notices a manifest attribute on the <html> element, it fires a checking event. (All the events listed here are fired on the window.applicationCache object.)

   The checking event is always fired, regardless of whether we have previously visited this page or any other page that points to the same cache manifest.

2. If browser has never seen this cache manifest before...

   - It will fire a downloading event, then start to download the resources listed in the cache manifest.
   - While it's downloading, your browser will periodically fire progress events, which contain information on how many files have been downloaded already and how many files are still queued to be downloaded.

3. After all resources listed in the cache manifest have been downloaded successfully, the browser fires one final event, cached.

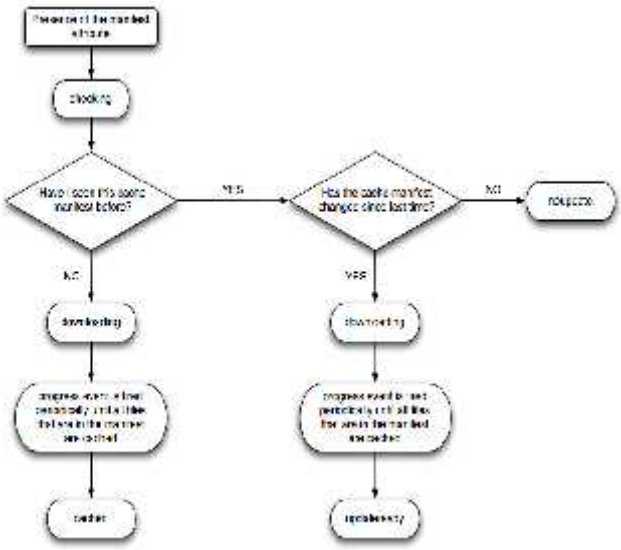

Fig 3. Flow Diagram of cache manifest

4. On the other hand, if a previously visited this page or any other page that points to the same cache manifest, then your browser already knows about this cache manifest. It may already have some resources in the appcache. It may have the entire working offline web application in the appcache. So now the question is, has the cache manifest changed since the last time browser checked it?

   - If the answer is no, the cache manifest has not changed, browser will immediately fire a noupdate event and work is done.

- If the answer is yes, the cache manifest *has* changed; browser will fire a downloading event and start re-downloading every single resource listed in the cache manifest.
- While it's downloading, browser will periodically fire progress events, which contain information on how many files have been downloaded already and how many files are still queued to be downloaded.
- After all resources listed in the cache manifest have been re-downloaded successfully, the browser fires one final event, updateready. This is a signal, that the new version of offline web application is fully cached and ready to be used offline. The new version is not yet in use. To "hot-swap" to the new version without forcing the user to reload the page, you can manually call the window.applicationCache.swapCache() function.

V. STEPS TO FOLLOW FOR CREATING OFFLINE WEB APPLICATION

Checking Browser Compatibility

Due to the variation in browser support, first we should check the compatibility of the browser before availing an application offline.

It can be done in two ways:

- Without using Modernizer Object
- Using the Modernizer Object

Creating a manifest file

This file resides on the server and decides which files should be stored client-side in the browser's AppCache, that can be used when the user goes offline .This file contains a list of URLs to various resources such as Javascript files, HTML CSS, images and flash files. The manifest file has a basic structure that should start with CACHE MANIFEST. File names must be listed exactly as they appear on disk as it is case sensitive.

Using the Application Cache API

To use the files from application cache, we can use the application Cache API. The "window.applicationCache" object provides events and properties that can be used to deal with data retrieval. Current status of the application cache can be checked using window.applicationCache.status.

Events

Certain events also get fired, depending on what is going on with the AppCache at the moment.

Add the manifest file to the local server

Once the .manifest file is created we need to add that file to the root Web directory of our application. The directive makes sure that every manifest file is served.

Linking manifest file to html page

The manifest file once created can be added to the html pages with the help of "manifest" attribute of the <html> element.

VI. RISKS INVOLVED IN OFFLINE WEB APPLICATION

[7]Although the offline web application is a serious advantage. The menace in building this kind of applications is if even a single resource listed in your cache manifest file fails to download properly, the entire process of caching your offline web application will fail. Your browser will fire the error event, but there is no indication of what the actual problem was. The second important point is something that is not an error, but it will look like a serious browser bug until we realize what's going on. It has to do with exactly how browser checks whether a cache manifest file has changed. These issues are making the offline web applications unstable.

VII.BROWSER'S SUPPORT FOR OFFLINE APPLICATIONS

As HTML is a client-oriented technology, the features that are supported always depend on the client browser itself.[8] As with many of the features of HTML5, not all browsers support the offline web application caching feature. Below is the list of prominent browsers that do support the offline feature.

1. Chrome 5.0 and more
2. Safari 4.0 and more

3. Firefox 3.5 and more

Offline Web Application API is fully supported by most modern browsers and mobile phones including:

- Android 2.0+
- I Phone 2.1+
- Blackberry 5.0+
- Firefox 3.5+
- Chrome 5.0+
- Opera 10.6+
- Safari 4.0+

VIII.ADVANTAGES OF OFFLINE WEBAPPLICATIONS:

- Used to view and write e-mail message
- Documents and presentations can be edited offline, in places that does not have Wi-Fi.
- Game state, navigation location, or storing some specific information that can be used across the entire Web application
- Intimating the user about the changes done (for example, account no.) if the same page is opened in multiple tabs.

IX.CONCLUSION:

At this point of time, support for the development of offline Web applications under the light of HTMl5 has limited scope and is not supported by most of the browsers available in the market. Different browser providers are following different approaches to achieve the same. This approach is also suffering from the size constraint as per browser implementation. However, going forward, it is expected that these APIs will be supported by almost all the leading browsers so that there will be a unified approach for the implementation and users can get the best out of these.

X. REFERENCES:

[1] "Mac Developer Library: System-Declared Uniform Type Identifiers". Apple. 2009-11-17.

[2] "HTML5 Differences from HTML4". WorkingDraft. World Wide Web Consortium.

[3] http://viralpatel.net/blogs/2010/10/introduction-html5-domstorage-api-example.html

[4] http://dev.opera.com/articles/view/offline-applications-html5-appcache/

[5] Dion Almaer, "How to take your Web Application Offline with Google Gears",

[6] Google System Blog, "YouTube Uploader Powered by Gears",

[7] Wordpress Security, "How to enable Google Gears in WordPress2.7",

[8] Google, "Google Reader – Offline reading",