Session 8 - Application cache (The cache manifest)

We will cover:
1) Setting up your site to be available offline
2) Working with fallback pages for content that is not available
3) Using chrome to check the status of your cache
4) The various pitfalls that can cause problems when using application cache.

Background: The application cache allows you to save your application (web pages, css, javascript, images etc.) locally on the device. This is an extremely useful feature in the mobile world, where internet connections may be patchy and slow.

Using the application cache makes your app. load faster (no network latency) and can make your application available even when the user is offline. The way the cache works also means that you, as a developer, can role out updates in the background.

The application cache is awesome. If you plan to use it in your projects you need to carefully test and make sure you understand the inner workings of what, on first glance, appears to be something relatively simple. The fact that the first article to really help me get my head around how to get the app cache working properly was “kill me, kill me now” should indicate that this is an area where dragons lie.

It is currently supported in the following mobile OS’s and browsers:

Android 2.0 +
iPhone 2.1 +
Opera Mobile 10.1 +
WebOS 2.0 +

Once you’ve got your head around the problems you are likely to experience it’s a very, very powerful tool.

How it works:

We control our cache with a manifest file. The manifest file is a simple text file. The name of the text file can be anything with any extension.

IMPORTANT: It must however be served with a MIME type of text/cache-manifest.

The content and structure of the cache manifest:

1) It must always starts with CACHE MAINIFEST
2) Comment lines always start with the # character
3) There are three main sections that you can specifiy:

```
CACHE:
# this is the default section under this text all paths or URL to items that you want to
# be cached should be listed. One item per line

FALLBACK:
# List of urls or paths with alternatives for display when the resources are unavailable.

NETWORK:
# A whitelist of all resources that should never be cached. The wildcard * maybe used in
# this section if you wish to force all resources not listed to be collected from the
# server.
```

CACHE MANIFEST

Paths in the manifest:

```
#relative path to the manifest
path/relative/to/manifest

#absolute path for site
/absolute/path

#full url
http://example.com/a/full/url

#full url with the same protocol as the host web page.
//example.com/
```

We tell the web page that it should be cached by adding a manifest attribute in our html tag, the value of the attribute should be the location of a valid manifest file.

```
<!DOCTYPE HTML>
<html manifest=”myManifest.manifest”>
    ...
</html>
```

A cache update is only attempted if the manifest file on the server is different from the current browsers manifest file.

When an update is attempted the browser looks for all the files that should be cached that are specified in the new cache manifest. IMPORTANT: Only if all the files exist, will the cache be updated i.e. listing one file that doesn’t exist, or is a redirect in the cache manifest file will prevent all files from being updated.

NOTE: an application cache has no expiration date.

NOTE: no support for resource redirects.

NOTE: for SSL resources, all resources must be on the same domain.

Reference
One of the best articles on application cache.
http://diveintohtml5.org/offline.html

A great GoogleTechTalk discussing the ins and out of application cache.
http://www.youtube.com/watch?v=CoUSIBep1G8&feature=youtube_gdata_player

Working with standard caching techniques (not application cache):
http://www.mnot.net/cache_docs/

Step 8a - Adding the cache manifest

We will add a manifest file to our mobile ui library file so that they are available to you offline.

1) Create a manifest file called step8a.manifest

```
CACHE MANIFEST
# v0.1
NETWORK:
*
```

2) Wire step8a.html to the cache manifest by adding the manifest attribute to the html tag.

```
<html lang="en" manifest="step8a.manifest">
```

3) Surf to the page in chrome and look at the output in console.

4) Surf to the page on your mobile.

5) Enable flight mode and refresh the page

6) Change some of the text on the page. and refresh your chrome browser. Look at the output in the chrome console.

7) Change the version number of the cache manifest.

8) Update the page again in chrome. Look at the console. The update should appear the next time you refresh the browser.

9) Refresh the browser. Hurray it's updated.

10) Rename the cache manifest file. This will make our cache obsolete and effectively erase the cache.

11) Refresh the browser. Look at the console it should indicate the cache is now obsolete.

Step 8b - A multipage cache with fallbacks

In this scenario we have a site with multiple pages, to many to cache them all (think wikipedia articles). What we want to do is cache the pages our user, visits so they are available offline. When the user is off line any pages that they have not visited and are therefore not cached should be replaced with a notcached.html website to inform the user they must go on line to retrieve the data.

1) Open folder session8/step8b and look at the content.

2) Wire each page to the cache manifest by adding the manifest attribute to the html tag.

```
<html lang="en" manifest="multipage.manifest">
```

3) Surf in chrome to one of the pages and look at the output in console. Make sure the cached event is successful. `Application Cache Cached event` **should be the last line.**

4) Now create a new html page called notcached.html in the step8b folder. Give it some content so you will recognise it.

5) Adjust the multipage.manifest file so our notcached.html page will be triggered if content can't be found:

```
CACHE MANIFEST
# v0.1
FALLBACK:
/ notcached.html
NETWORK:
*
```

6) Now on your mobile device surf to page1.html page3.html and page5.html.

7) Switch your mobile into flight mode and surf to pages1 to 5. If all has worked well page2.html and page4.html should be replaced by notcached.html.

Step 8c - Taking our wiki app offline.

In this exercise you will wire up your wiki app, step8c.html, and take it offline. Use the Network feature in chrome to see what files it accesses and decide which files should be cached and which ones should be loaded from the network.

1) Create a cache manifest and connect it to step8c.html, test that it works in chrome.

2) Add the relevant files to the different sections of the cache-manifest.

3) When you think that you have everything the way it should be make sure your cache is valid by loading your page in chrome and looking at the console. If it is valid test it on your mobile.

4) Open step8c.html on your mobile.

5) Disconnect your mobile from the wifi network and refresh your web page, make sure that you can still search with the app.

6) If your app does not work recheck your cache and remember it can take two browser refreshes before changes are visible.

Extra exercise if time permits:

Look at the js/aircon.js file. I have created this file to provide chrome like app cache logging on iOS. The browser support a number of appCache commands, although what we can do with javascript and the application cache is slightly limited. There is enough functionality for basic logging.

Experiment with adding the aircon.js file to your page. (don't forget to update you manifest file, and that it will take two reloads before it is visible).

Enable safari's debug console on your iOS device. Settings --> Safari --> Developer --> Debug Console = On.

Surf to your webpage on you iOS device and open the console. You should have chrome like logging of the cache status.