Chapter 4: Using the Application Cache

Using the Application Cache

HTML5 has several options for storing data on devices. First among these—and perhaps the most important for our purposes—is the *application cache*, or *AppCache*.

You probably already know that a cache (pronounced *cash*) is a place to store or hide things. The AppCache provides a way for Web browsers to store files on a device and use the stored versions. Otherwise, the device would have to retrieve new files from the Web server each time you access a Web app.

By caching files locally, the AppCache dramatically improves the performance of Web apps. It also has the potential to make Web apps usable without an Internet connection.

To use the AppCache, you need to create a *cache manifest*. That's the file that tells the browser what files it should store on the device. It takes the form of a simple text file that's linked from every HTML page in your app.

Here's a sample cache manifest file:

```
CACHE MANIFEST

# version 1

CACHE:

index.html

docs.html

watzthis.css

buttons.png

http://code.jquery.com/mobile/1.0/jquery.mobile-1.0.min.css

http://code.jquery.com/jquery-1.6.4.min.js

http://code.jquery.com/mobile/1.0/jquery.mobile-1.0.min.js

NETWORK:

/register.html

FALLBACK:

*.html /offline.html
```

The manifest has three parts: the explicit section, the online whitelist section, and the fallback section.

The *explicit section* begins after the word *CACHE:*. It lists the network path or URL to each file that you want the browser to cache. This might include HTML files, CSS files, JavaScript files, images, audio files, video, and so forth.

When the browser first opens the cache manifest file, it will download all the files in the explicit section. If you list every file that your app needs in order to run in this section, you can turn off your device's Internet connection, and the app will continue to function normally once the files have downloaded.

As long as the cache manifest file isn't modified, the browser will continue to use the version of the files that it has cached every time it loads this site going forward. Sometimes, however, you may want the cached files to update to the latest versions. This is where the line of text above the list of files comes in:

```
# version 1
```

The #, or hash symbol, indicates that what follows is a comment for humans to read rather than for the browser. Because it only takes a change of one character in the cache manifest file to force the browser to reload all the files, this particular comment often appears in the cache manifest file to allow developers to control when end users will see the latest versions of files. For example, you might modify a bunch of files in your app and then deploy them to your server. After everything's in place, you could update the version comment to:

```
# version 2
```

Users would then begin to see the new files the next time they access the app while connected to the Internet. In effect, this is a way to install updates to mobile Web apps.

The part of the cache manifest beginning with the word NETWORK is the *online whitelist*. This is where you list resources that you don't want cached. For example, if you have a directory of images that you've specified should be cached in the Explicit section, but you want one of those images to never be cached, you'd list that image here.

Why would you want certain files to stay uncached? Well, a common use for the online whitelist section is to list resources, such as APIs, that need an Internet connection to function.

The *fallback section* is where you list files that the browser should display if a requested file is inaccessible.

Each line of the fallback section contains a file (or a pattern) followed by a space (or a tab) and then the file that should replace it.

```
*.html /offline.html
```

In my example above, the asterisk (*) tells the browser to replace any missing HTML file with a file called *offline.html*. This file would presumably contain a message to the effect of "You must be online in order to view this content."

Configuring Your Server for Cache Manifests

To use the AppCache, your Web server has to deliver the cache manifest file to your browser in the right format. This is a new technology, so Web servers generally don't have built-in support for cache manifests. Fortunately, it's easy to remedy this in most cases.

If you're using an Apache Web server (which is most likely the case if you host your app on the Linux operating system), you'll need to find or create the file called .htaccess in the root of your Web directory. Then add the following command inside that file:

```
Addtype text/cache-manifest .appcache
```

After you add this, Web browsers that support AppCache will understand that any file with the .appcache extension is a cache manifest.

It's easy to link to a cache manifest from within an HTML5 document. Simply add the manifest attribute to the <html> element on every HTML page of the site, like this:

```
<html manifest="http://www.watzthis.com/watzthis.appcache">
```

If your manifest contains a large number of files, or large files, the first load of a page that links to the manifest will be slower than normal. After that initial load, however, the app will be much faster. Why? Its resources are coming directly from device storage rather than being downloaded from the Internet.

We're coming to the end of another lesson! Please go to Chapter 5 for a quick recap.