Một trong những tính năng mới không kém phần quan trọng của HTML5 đó là sử dụng bộ nhớ cache của trang web để làm việc ngoại tuyến. Nó có những lợi thế nhất định như sau:

* Duyệt web ngoại tuyến – người dùng có thể duyệt toàn bộ trang web hay sử dụng ứng dụng web ở chế độ ngoại tuyến.
* Các tài nguyên cache lưu trữ cục bộ nên giúp load nhanh hơn.
* Giảm tải server, duyệt chỉ tải tài nguyên về khi nó bị thay đổi.

Application Cache (hoặc AppCache) cho phép một lập trình viên xác định các tập tin mà trình duyệt sẽ đưa vào bộ nhớ cache vì vậy người dùng có thể sử dụng khi ngoại tuyến. Ứng dụng sẽ hoạt động chính xác, ngay cả khi người dùng nhấn nút refresh trong khi khi họ đang ngoại tuyến.

1. Tập tin Cache Mainfest

Tập tin cache manifest là một tập tin văn bản đơn giản liệt kê các nguồn tài nguyên trình duyệt sẽ lưu vào bộ nhớ đệm để truy xuất offline.

* Tham chiếu tới một tập tin mainfest

Để sử dụng được nó, trước hết ta cần include thuộc tính mainfest vào HTML:

<html manifest="example.appcache">

...

</html>

Thuộc tính manifest nên được đưa vào mỗi trang mà chúng ta muốn lưu trữ. Trình duyệt không lưu vào bộ nhớ đệm một trang nếu nó không chứa thuộc tính này (ngoại trừ nó được liệt kê rõ ràng trong tập tin manifest ). Điều này có nghĩa rằng bất kỳ trang nào có thuộc tính mainfest sẽ được mặc nhiên thêm vào bộ nhớ cache của ứng dụng.., không cần phải liệt kê tất cả các trang trong tập tin mainfest.

Thuộc tính manifest cũng có thể trỏ đến một URL tuyệt đối hoặc đường dẫn tương đối, nhưng một URL tuyệt đối phải dưới cùng gốc như trang hiện tại. Một tập tin manifest có thể có bất kỳ phần mở rộng nào, chỉ cần nội dung của nó đúng định dạng.

<html manifest="http://www.example.com/example.mf">

...

</html>

Một tập tin mainfest cần được cung cấp kèm theo mime-type text/cache-manifest, thêm nó và tập tin tùy chỉnh tới server hoặc tập tin cấu hình .htaccess:

AddType text/cache-manifest .appcache

Hoặc:

- url: /mystaticdir/(.\*\.appcache)

static\_files: mystaticdir/\1

mime\_type: text/cache-manifest

upload: mystaticdir/(.\*\.appcache)

* Cấu trúc của một tập tin mainfest

Một tập tin mainfest đơn giản thì có dạng như thế này:

CACHE MANIFEST

index.html

stylesheet.css

images/logo.png

scripts/main.js

Trình duyệt sẽ sử dụng tài nguyên cache tới khi nó được thay đổi, vì vậy, một khi các nguồn tài nguyên đó thay đổi, ta cần tạo một đánh dấu thay đổi trong tập tin cache để trình duyệt nhận biết và cập nhật lại tài nguyên đó. Việc cập nhật cache sẽ được nói đến trong phần sau.

Một tập tin mainfest có 3 phần riêng biệt: CACHE, NETWORK và FALLBACK.

CACHE: tất cả những tập tin được liệt kê trong phần này, sẽ được lưu vào bộ nhớ cache sau khi chúng được tải xuống lần đầu tiên.

NETWORK: ngược lại với CACHE, các tài nguyên liệt kê trong phần này phải được kết nối mạng, nó sẽ ko lưu vào bộ nhớ cache ngay cả khi người dùng ngoại tuyến.

FALLBACK: trang dự phòng trong trường hợp một tài nguyên không thể truy cập. URI đầu tiên là tài nguyên chính, URI thứ hai là cái dự phòng. Cả hai đều phải là địa chỉ tương đối và nằm cùng gốc với tập tin mainfest.

CACHE MANIFEST

*# 2010-06-18:v2*

*# Explicitly cached 'master entries'.*

CACHE:

/favicon.ico

index.html

stylesheet.css

images/logo.png

scripts/main.js

*# Resources that require the user to be online.*

NETWORK:

login.php

/myapi

http:*//api.twitter.com*

*# static.html will be served if main.py is inaccessible*

*# offline.jpg will be served in place of all images in images/large/*

*# offline.html will be served in place of all other .html files*

FALLBACK:

/main.py /static.html

images/large/ images/offline.jpg

\*.html /offline.html

1. Cập nhật Cache

Các ứng dụng web sẽ sử dụng bộ nhớ cache tới khi có một trong các điều sau đây xảy ra:

* Người dùng xóa dữ liệu trình duyệt của trang web.
* Tập tin mainfest bị thay đổi.
* Bộ nhớ ứng dụng cache được cập nhật
* Trang thái Cache

Đối tượng window.applicationCache truy cập tới bộ nhớ cache của trình duyệt, vì vậy ta sử dụng trạng thái trả về của nó để kiểm tra tình trạng của cache:

var appCache = window.applicationCache;

switch (appCache.status) {

case appCache.UNCACHED: *// UNCACHED == 0*

return 'UNCACHED';

break;

case appCache.IDLE: *// IDLE == 1*

return 'IDLE';

break;

case appCache.CHECKING: *// CHECKING == 2*

return 'CHECKING';

break;

case appCache.DOWNLOADING: *// DOWNLOADING == 3*

return 'DOWNLOADING';

break;

case appCache.UPDATEREADY: *// UPDATEREADY == 4*

return 'UPDATEREADY';

break;

case appCache.OBSOLETE: *// OBSOLETE == 5*

return 'OBSOLETE';

break;

default:

return 'UKNOWN CACHE STATUS';

break;

};

Để tiến hành cập nhật bộ nhớ cache, đầu tiên gọi applicationCache.update(). Nó sẽ cố gắng cập nhật bộ nhớ cache của người dùng (đòi hỏi các tập tin manifest đã thay đổi). Cuối cùng, khi applicationCache.status ở trạng thái UPDATEREADY, gọi applicationCache.swapCache() để đổi thay bộ nhớ cache cũ.

var appCache = window.applicationCache;

appCache.update(); *// Attempt to update the user's cache.*

...

if (appCache.status == window.applicationCache.UPDATEREADY) {

appCache.swapCache(); *// The fetch was successful, swap in the new cache.*

}

Như vậy, chúng ta chỉ tải lại các tài nguyên cache khi nó thật sự thay đổi, vì vậy ở sự kiện tải trang ta cài đặt lắng nghe sự kiện “updateready”. Nhờ vậy, nếu tài nguyên không có thay đổi gì thì trình duyệt không cần tải về ngay cả khi người dùng tải lại trang:

*// Check if a new cache is available on page load.*

window.addEventListener('load', function(e) {

window.applicationCache.addEventListener('updateready', function(e) {

if (window.applicationCache.status == window.applicationCache.UPDATEREADY) {

*// Browser downloaded a new app cache.*

*// Swap it in and reload the page to get the new hotness.*

window.applicationCache.swapCache();

if (confirm('A new version of this site is available. Load it?')) {

window.location.reload();

}

} else {

*// Manifest didn't changed. Nothing new to server.*

}

}, false);

}, false);

* Sự kiện AppCache

Chúng ta có thể thêm vào xử lý các sự kiện trong quá trình làm việc với bộ nhớ cache, tùy vào từng sự kiện cụ thể mà ta có những xử lí phù hợp. Sau đây là ví dụ thêm lắng nghe các sự kiện cache:

function handleCacheEvent(e) {

*//...*

}

function handleCacheError(e) {

alert('Error: Cache failed to update!');};

*// Fired after the first cache of the manifest.*

appCache.addEventListener('cached', handleCacheEvent, false);

*// Checking for an update. Always the first event fired in the sequence.*

appCache.addEventListener('checking', handleCacheEvent, false);

*// An update was found. The browser is fetching resources.*

appCache.addEventListener('downloading', handleCacheEvent, false);

*// The manifest returns 404 or 410, the download failed,*

*// or the manifest changed while the download was in progress.*

appCache.addEventListener('error', handleCacheError, false);

*// Fired after the first download of the manifest.*

appCache.addEventListener('noupdate', handleCacheEvent, false);

*// Fired if the manifest file returns a 404 or 410.*

*// This results in the application cache being deleted.*

appCache.addEventListener('obsolete', handleCacheEvent, false);

*// Fired for each resource listed in the manifest as it is being fetched.*

appCache.addEventListener('progress', handleCacheEvent, false);

*// Fired when the manifest resources have been newly redownloaded.*

appCache.addEventListener('updateready', handleCacheEvent, false);