# Cache size limitations

INTRODUCTION

For traditional Web sites of normal size, there is no problem. Just create a manifest file as we've shown you (in the "make your Web site run offline" unit), and your Web site should work offline!

On today's (2015) desktop browsers, the available cache size is generally sufficient per domain, and there is no need to worry.

If a user visits your Web site and if this Web site asks to be cached, and if there is no more free space available, the browser will find some (in general by asking the user permission to increase the cache size limit for the current domain - see screenshots-, or by clearing least recently used data). On mobile devices, the cache size limits are lower but reasonable.
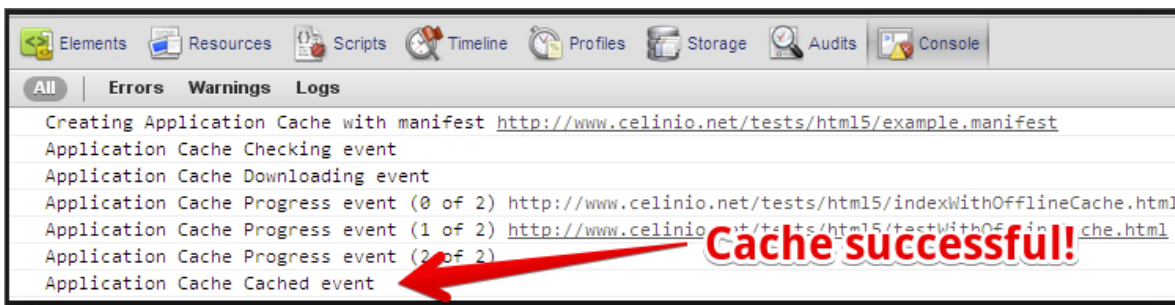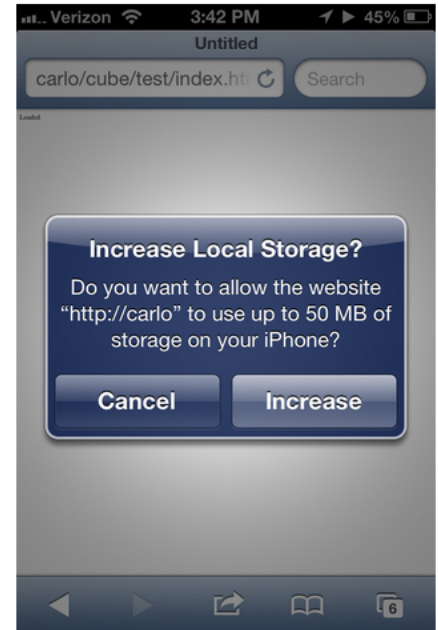
With some atypical applications that need to cache large amounts of data (images, videos), there may indeed be inadequate space in the cache. In this case, the browser prints an error message in the console. It's also possible to use a JavaScript API to detect this situation and show the user an error or a warning message in the page "Sorry, this Web site will not be available offline as we could not cache it due to size limitations...". This advanced part will not be covered in this course.

```
    ◄   ►   Current Log
    preloading of 6e0319df-0f3b-4a0b-9d4b-52a7b099a330 ends with status: cached      main.js:21
    preloading of 6e8919a1-2b4d-43d4-8d80-555985376966 ends with status: cached      main.js:21
    preloading of 7096513e-3c9b-4cef-8b27-c1f55aa31e  ends with status: cached       main.js:21
    preloading of 71c7ec2f-46d7-4b77-bfcc-1bcb10347   ends with status: cached       main.js:21
  ⊗ Application Cache update failed, because size quota was exceeded.
```

Anyway, if you wish to find out more about size limits and what the various HTML5 persistence mean, please read the rest of this page. As such, **there will be no exercises related to the optional content material below!**

FOR TRADITIONAL WEB PAGES

There is a size limit, but it varies depending on many parameters!

There are size limitations, but they vary from one browser to another. And they also vary depending on whether your browser is running on a mobile or desktop device. Finally they may vary depending on the size of the hard disk or the size of the memory of the device the browser is running on!
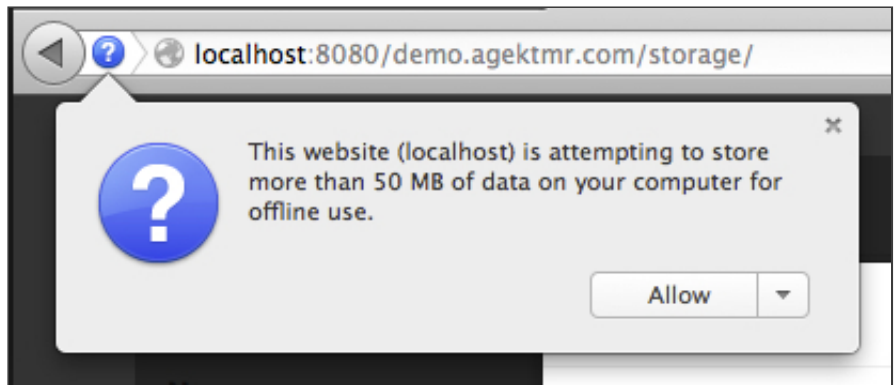
For example, with Chrome there is no maximum size limit for cached data (it is computed depending on the configuration and on the current amount of data already persisted). But if you use the LocalStorage persistence for storing data, (see next section of this course), then, for this special kind of data, Chrome applies a 10Mbytes per domain limit. And this is completely different on FireFox, on Safari, etc. The W3C specification recommends at least 5Mbytes per domain with support for at least 20 domains.

We recommend reading this article from HTML5 rocks (January 2014). The author wrote a Web application that tries to fill all available space and "guess how the different persistent storage spaces" are managed by different browsers. He tried with all major browsers on mobile devices and on desktop.

On the right: a screenshot of some of the resulting measures (extract from the above

article).

Also, for greater detail, see this paper about "Managing HTML5 offline storage".

A quota management JavaScript API exists!

In order to make Web applications aware of the free space available for persisting their data, a Quota Management API is being developed. Note that this API specification is under active discussion and is subject to change in the future. It's still experimental.

Desktop

	Chrome 40	Firefox 34	Safari 6, 7	Safari 8	IE 9	IE 10, 11
Application Cache	up to quota	500MB, Unlimited	Unlimited?	Unlimited?		100MB?
FileSystem	up to quota					
IndexedDB	up to quota	50MB, Unlimited		up to quota?		10MB, 250MB (~999MB)
WebSQL	up to quota		5MB, 10MB, 50MB, 100MB, 500MB, 600MB, 700MB...	5MB, 10MB, 50MB, 100MB, 500MB, 600MB, 700MB...		
LocalStorage	10MB	10MB	5MB	5MB	10MB	10MB
SessionStorage	10MB		Unlimited	Unlimited	10MB	10MB

This demo shows the used quota and the remaining free space I can get with Chrome, for persisting local data:

Show me my usage and free space remaining

Used space = 19256, remaining space = 3102400001

Try this example on JS Bin that shows the available disk space for all HTML5 persistence means (cache, localStorage, IndexedDB, etc.). Try it with Google Chrome: as of June 2015, it is the only browser that implements the Quota API.

Source code extract:

```javascript
function showFreeSpace() {
   //the type can be either TEMPORARY or PERSISTENT

webkitStorageInfo.queryUsageAndQuota(webkitStorageInfo.TEMPORARY,
          onSuccess, onError);
}

function onSuccess(usedSpace,remainingSpace) {
    var displaySpace =document.querySelector("#space");
```

```
        displaySpace.innerHTML = "Used space = " + usedSpace + ",
    remaining space = "
                                        +remainingSpace;

    }

    function onError(error) {
        console.log('Error', error);
    }
```

APPLICATIONS FROM STORES OR APPLICATIONS THAT RUN ON A WEB-BASED OPERATING SYSTEM

A note about "non traditional Web applications"

Finally, the amount of free storage space is managed differently if your Web application is not a "traditional HTML page" loaded in a browser, i.e. using a `http://` or `https://`URL. It may be a browser extension, a browser application (coming from a store such as Chrome store, Windows store, FireFox OS store, etc), it may run on a Playstation 4 or on a Xbox One.
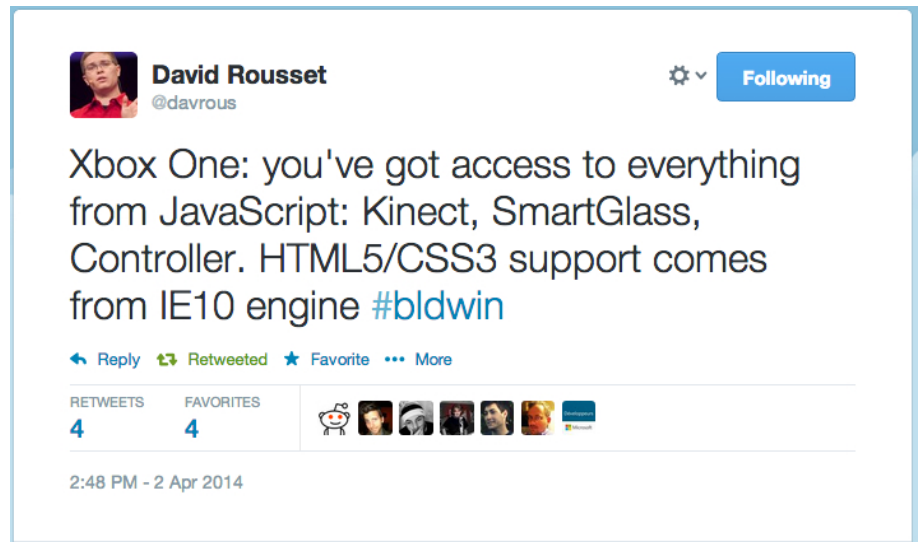
In these cases, HTML5 applications can get more privileges such as reading/writing to your hard disk and benefit from a large disk space. Imagine that they are similar to applications you install on your smartphone: you accept that they use XXX megabytes, use your GPS without asking for permission, etc. This course will not cover this type of application, but only focuses on traditional Web pages and applications.

Don Olmstead ← **PS4 Dashboard developer**   #PS4
Il y a 4 mois   Public

Activités liées aux posts   Ignorer

Now that the #PS4 has been released I can finally start talking about my piece of the puzzle.

When you login to your PS4 you are running #WebGL code. The PlayStation Store, the Music and Video Applications, as well as a good chunk of UX are all rendered within the browser.

I spent a good amount of time tuning our WebGL rendering engine, and I will be speaking at +SFHTML5 about how to optimize WebGL usage within the context of that work. There will be plenty of great tips on how you can speed up your own WebGL applications so get your slot now. And for those of you can't make it in person it will be live streamed on Google Developers Live.