HTML5 Offline Storage

Sang Shin
Founder & Chief Instructor
JPassion.com
"Learn with Passion!"

Topics

- Offline web applications
- HTML5 Offline storage types
- Application cache
- Local and session storage
- IndexedDB
- File system
- Quota API
- Online/Offline events

Acknowledgments

- Some of the contents of this presentation is borrowed from HTML5 tutorial of Mozilla Development Network according to the licensing terms of “Creative Commons Attribution-ShareAlike 2.5”
  - http://creativecommons.org/licenses/by-sa/2.5/

Offline Web Applications

Offline Web Applications

- Your Web app (browser-based app) needs to function without connectivity
  - One big reason why people still use desktop apps
- Your Web app needs to present a consistent UI to your users so that even if they are offline, they can still see and use some of your application
  - Build your apps with offline in mind first

HTML5 Offline Storage Types & Technologies

HTML5 Offline Storage Types

- Temporary (Transient)
  - Automatically given to each app
  - Shared among all web apps running in the browser. The shared pool can be up to half of the of available disk space
- Persistent
  - Stays in the browser unless the user expunges it
  - An application can have a larger quota for persistent storage than temporary storage, but it must request storage using the Quota Management API and the user must grant it
- Unlimited
  - Unlimited storage is similar to persistent storage, but it is available only to Chrome apps and extensions (.crx files)

HTML5 Offline Storage Technologies

- Application cache
- Local and session storage
- IndexedDB
- Local file system

Application Cache

Why Application Cache (AppCache)?

- Why Application cache?
  - Old browsers have caching mechanisms, but they're unreliable and don't always work as you might expect
  - Old browsers do not allow app-specific control for developers
- Application cache enables
  - Offline browsing - users can navigate your full site even when they're offline
  - Higher speed - cached resources are local, and therefore they get loaded faster
  - Reduced server load - the browser will only download resources from the server that have changed
- Where Application cache information specified?
  - In the “Application cache manifest file”

Application Cache Manifest File

- What is application cache manifest file for?
  - You specify which application files the browser should cache and make available to offline users in the App. Cache Manifest file
- Where do you specify the location of application cache manifest file?
  - In the <html ..> tag with “manifest” attribute

```
<!DOCTYPE html>
<html manifest="my_cache1.appcache">
<head>
  ...
</head>
<body>
  ...
</body>
</html>
```

Application Cache Manifest File

- A manifest can have three distinct sections – each section is marked with a header
  - CACHE - default section for entries. Files listed under this header (or immediately after the CACHE MANIFEST) will be explicitly cached after they're downloaded for the first time.
  - NETWORK - resources that require a connection to the server regardless. All requests to these resources bypass the cache, even if the user is offline. Wildcards may be used.
  - FALLBACK - An optional section specifying fallback pages if a resource is inaccessible. The first URI is the resource, the second is the fallback. Both URIs must be relative and from the same origin as the manifest file. Wildcards may be used.

Updating Cache

- Once an application is offline, it remains cached until one of the following happens:
  - The user manually clears their browser's data storage for your site
  - The manifest file is modified. Note: updating a data file listed in the manifest doesn't mean the browser will re-cache that resource. The manifest file itself must be altered.
  - The app cache is programmatically updated

Programmatic Control

- Browsers provide *window.applicationCache* object for programmatic access to the browser's app cache
- You can get the current status of the cache
  - *var appCache = window.applicationCache;*
  - *console.log(appCache.status)*
- Status of the cache
  - *Uncached, Idle, Checking, Downloading, UpdateReady, Obsolete*

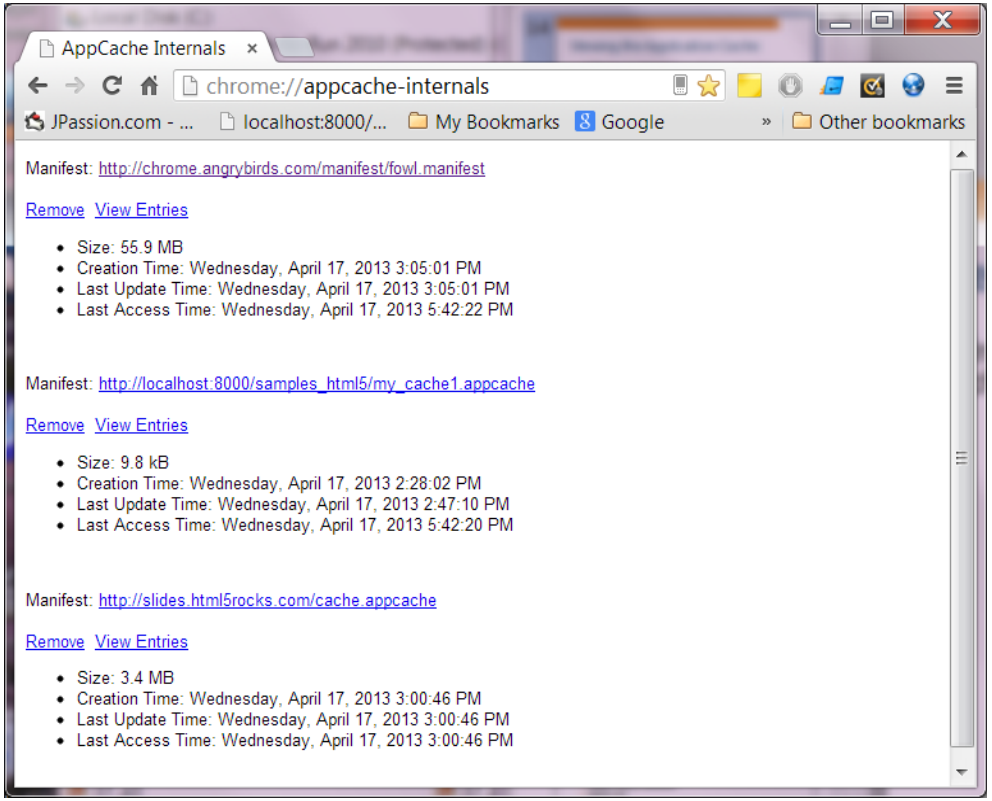
AppCache Event

- AppCache events are fired by the browser by default
- You can also register custom event handlers to catch these events

```
var appCache = window.applicationCache;

// 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);

// More events
```

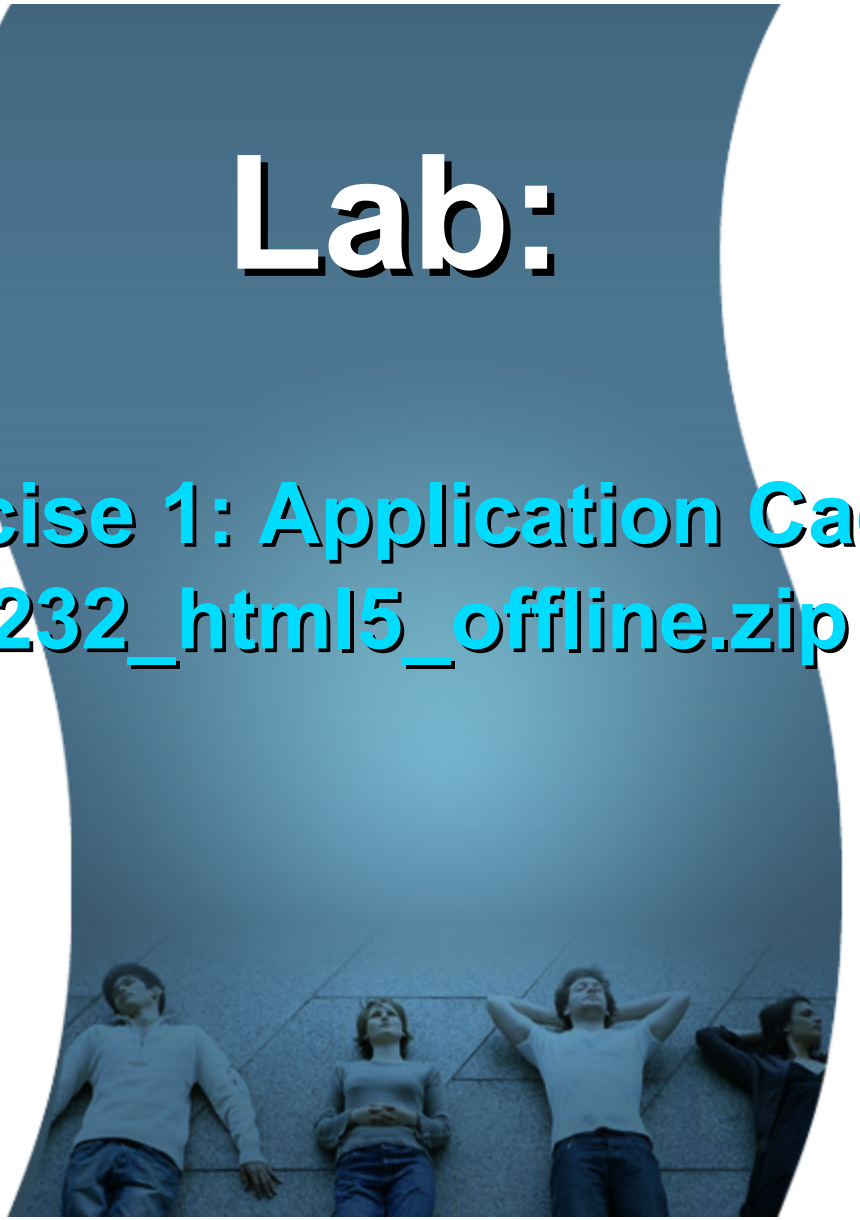
Viewing the AppCache Contents

- In Chrome, you can see the AppCache contents from *chrome://appcache-internals*



Lab:

Exercise 1: Application Cache
1232_html5_offline.zip



Local and Session Storage

Why HTML5 Storage API?

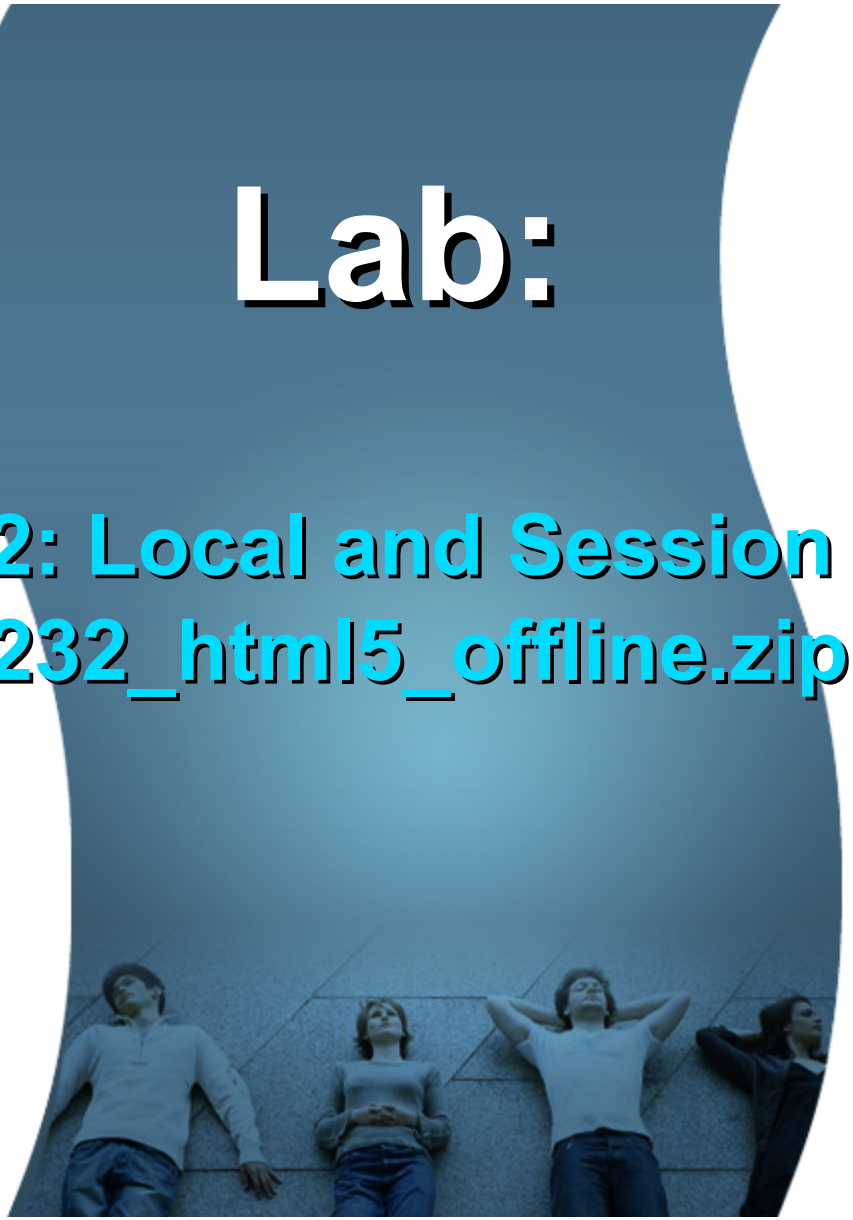
- Cookie was only storage option (available to developers) before HTML5
- Cookies limit how much you can save (typically 4K limit)
  - You cannot save large document
- Cookies are transmitted back and forth for every request
  - Waste of network bandwidth
  - Security risk

HTML5 Storage API

- Session storage API
  - Persists only as long as the window or a tab is alive
  - Values are visible only within the window or tab in which it is created
  - Survives page reloading but will be deleted when the user closes the window or the tab
- Local storage API
  - Persists beyond page restarts
  - Values are shared across every window or tab from the same origin
  - Good for storing preferences
- Only String data type can be stored

Lab:

Exercise 2: Local and Session Storage
1232_html5_offline.zip



IndexedDB

What is and Why IndexedDB?

- IndexedDB allows storage of significant amounts of structured data and for high performance searches on this data using indexes
  - Example: a catalog of DVDs in a lending library
- One of the main design goals of IndexedDB is to allow large amounts of data to be stored for offline use
- IndexedDB provides powerful query abilities

IndexedDB Comparison

- Local storage vs. IndexedDB
  - Local storage - lets you store/retrieve data using a simple key-value pair, limited in size
  - IndexedDB - a more powerful option, lets you locally store large numbers of ad-hoc data (objects) and retrieve data using robust data access mechanisms
- Replaces Web SQL
  - WebSQL Database is deprecated by W3C in 2010
  - WebSQL Database is a relational database access system, whereas IndexedDB is an indexed table system.
  - IndexedDB provides indexing, transactions, querying, cursor support

Inner workings of IndexedDB

- IndexedDB lets you store and retrieve objects which are indexed with a key
- All changes that you make to the database happen within transactions
- Like most web storage solutions, IndexedDB follows a same-origin policy
  - So while you can access stored data within a domain, you cannot access data across different domains.

IndexedDB:
Key Concepts

IndexedDB Key Concept #1

- IndexedDB databases store key-value pairs
  - The values can be complex structured objects, and keys can be properties of those objects
  - You can create indexes that use any property of the objects for quick searching, as well as sorted enumeration

IndexedDB Key Concept #2

- IndexedDB is built on a transactional database model.
  - Everything you do in IndexedDB always happens in the context of a transaction - you cannot execute commands or open cursors outside of a transaction
  - This transaction model is really useful when you consider what might happen if a user opened two instances of your web app in two different tabs simultaneously. Without transactional operations, the two instances could stomp all over each other's modifications.

IndexedDB Key Concept #3

- The IndexedDB API is mostly asynchronous
  - The API doesn't give you data by returning values synchronously; instead, you have to pass a callback function.

IndexedDB:
How to use it

Basic Usage Pattern

1. Open a database and start a transaction.
2. Create an object store.
3. Make a request to do some database operation, like adding or retrieving data.
4. Wait for the operation to complete by listening to the right kind of DOM event.
5. Do something with the results (which can be found on the request object).

Structuring the database

- IndexedDB does not use Structured Query Language (SQL)
  - It uses queries on an index that produces a cursor, which you use to iterate across the result set
- IndexedDB uses object stores rather than tables, and a single database can contain any number of object stores.
- Whenever a value is stored in an object store, it is associated with a key

Lab:

Exercise 3: IndexedDB
1232_html5_offline.zip

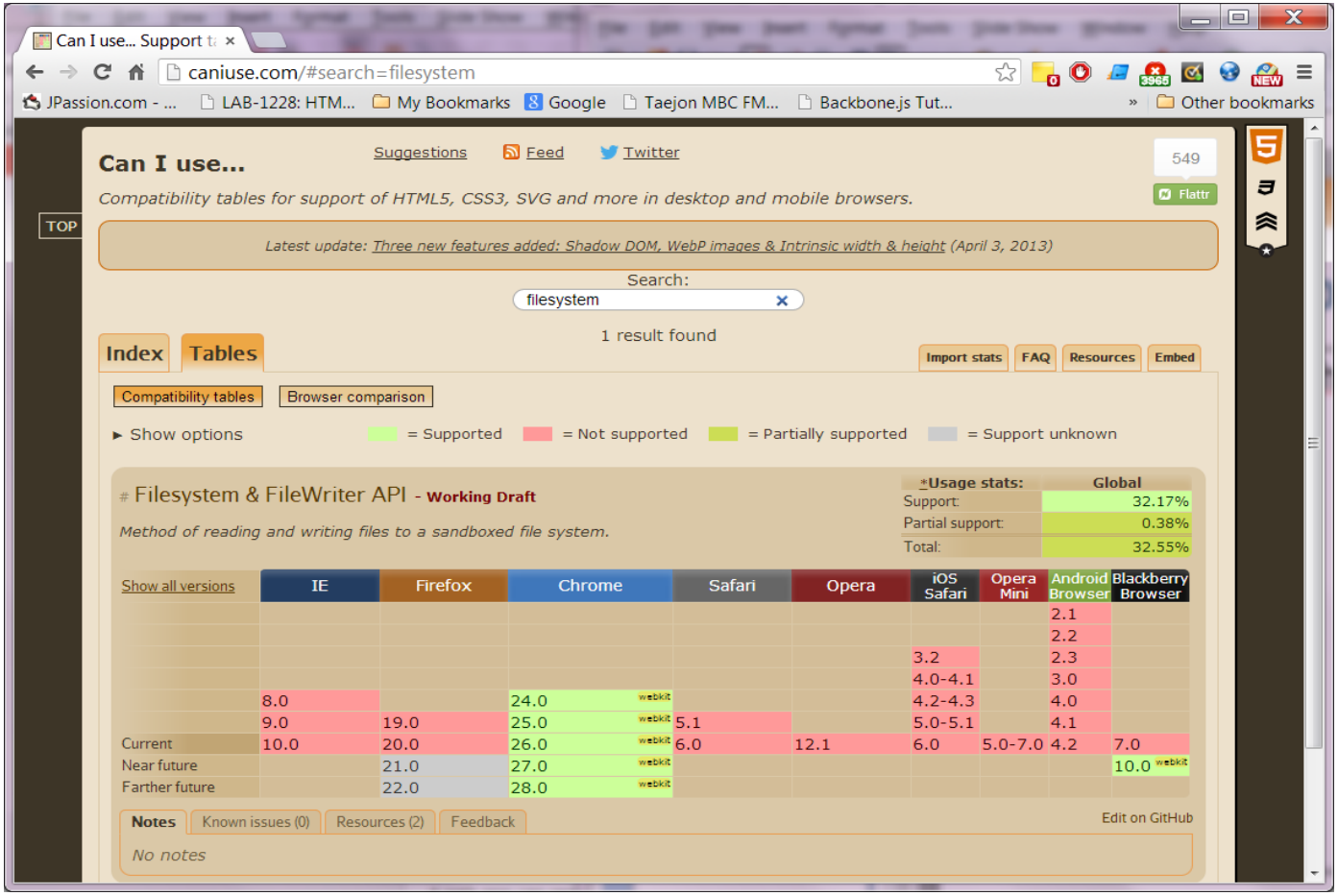
Local File System

FileSystem APIs

- What is FileSystem APIs?
  - With the FileSystem API, a web app can create, read, navigate, and write to a sandboxed section of the user's local file system
- 3 Types of FileSystem APIs
  - Reading and manipulating files: File/Blob, FileList, FileReader
  - Creating and writing: Blob(), FileWriter
  - Directories and file system access: DirectoryReader, FileEntry/DirectoryEntry, LocalFileSystem

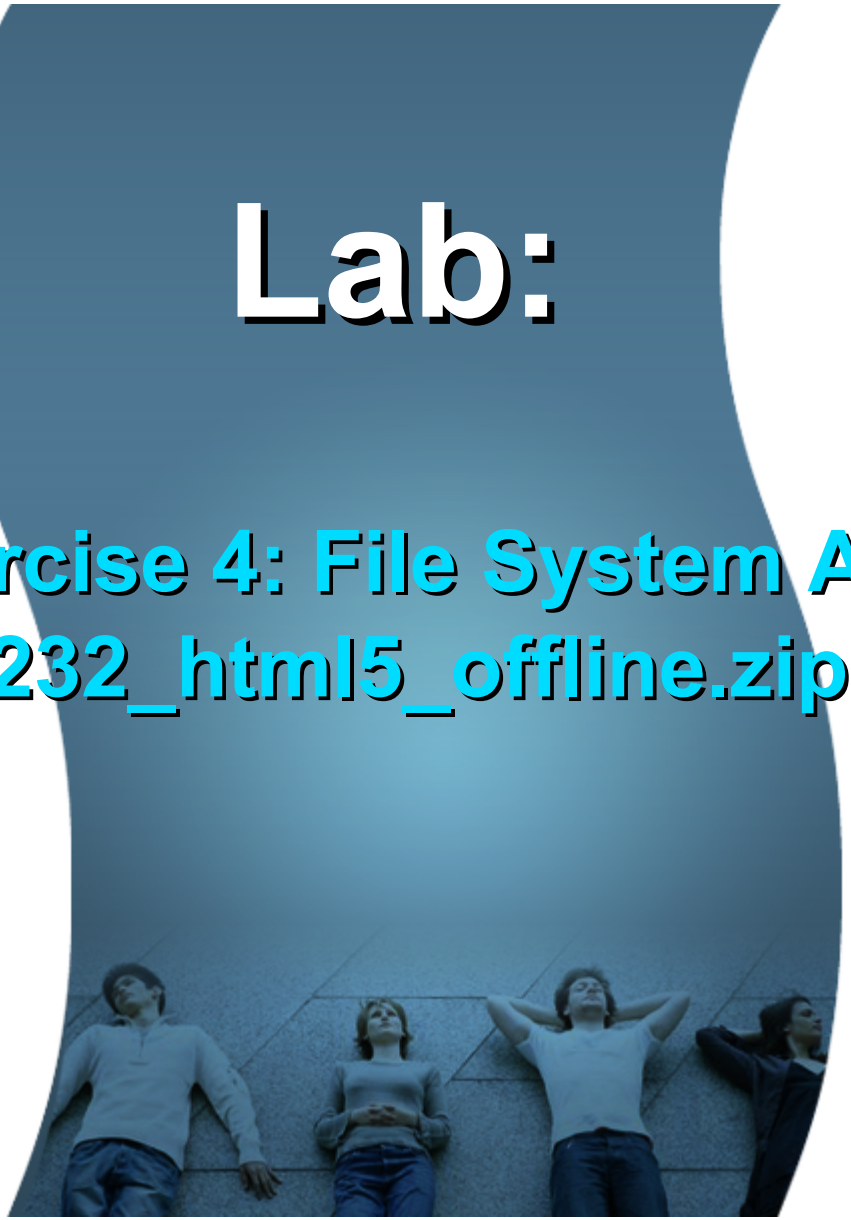
Browser Support (limited to Chrome)

- As of April, 2013 - http://caniuse.com/#search=filesystem



Lab:

Exercise 4: File System API
1232_html5_offline.zip



Quota API

Querying Storage Usage

- To query the storage size that is being used and the available space left for the host, call queryUsageAndQuota()

```
// Request storage usage and capacity left
window.webkitStorageInfo.queryUsageAndQuota(
      webkitStorageInfo.TEMPORARY,
     function(used, remaining) {
         console.log("Used quota: " + used + ", remaining quota: " + remaining);
     },
    function(error) {
        console.log('Error', error);
    }
);
```

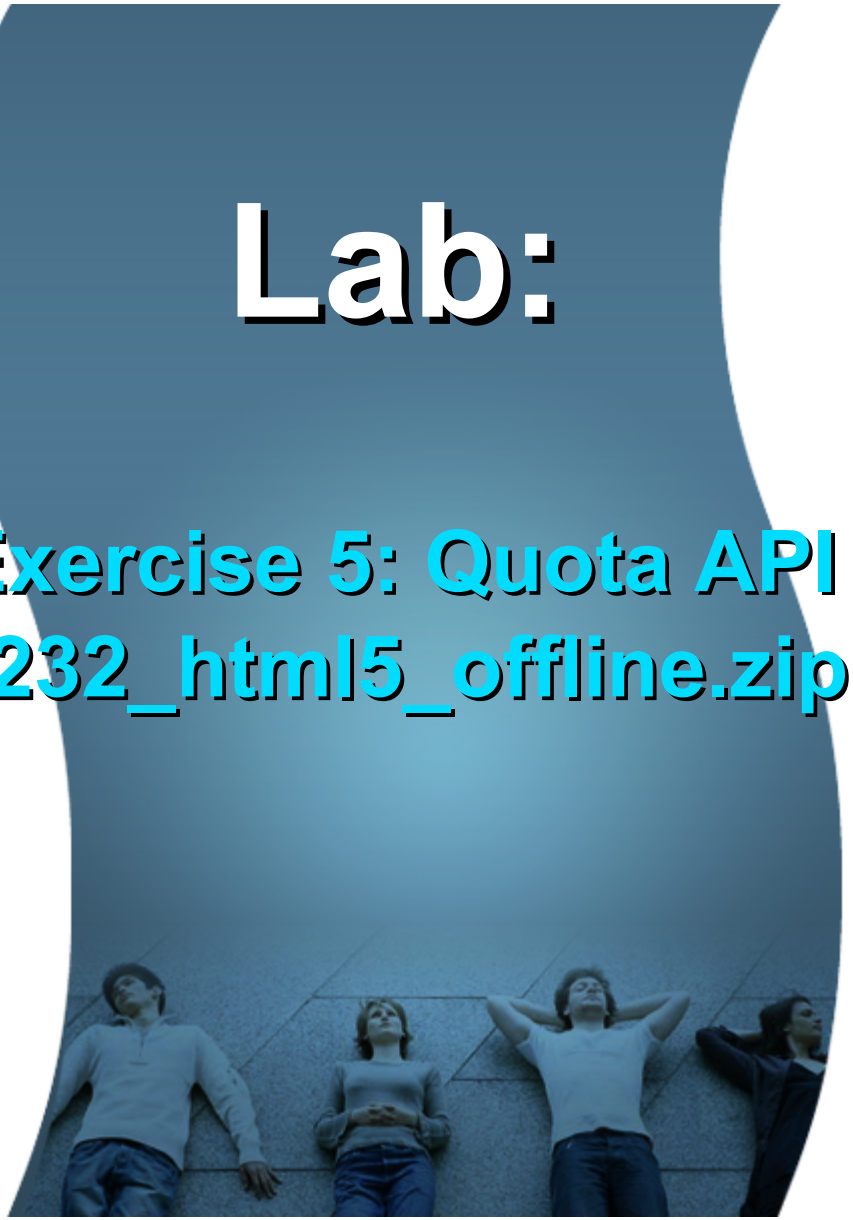
Asking for more storage

- For persistent storage for Files System API, the default quota is 0, so you need to explicitly request storage for your application. Call requestQuota() with the following
  - Type of storage
  - Size
  - Success callback

```
// Request Quota (only for File System API)
window.webkitStorageInfo.requestQuota(PERSISTENT, 1024*1024,
            function(used) {
               console.log('Used', used);
            },
           function(error) {
               console.log('Error', error);
           }
);
```

Lab:

Exercise 5: Quota API
1232_html5_offline.zip



Online/Offline Events

Online/Offline Detection

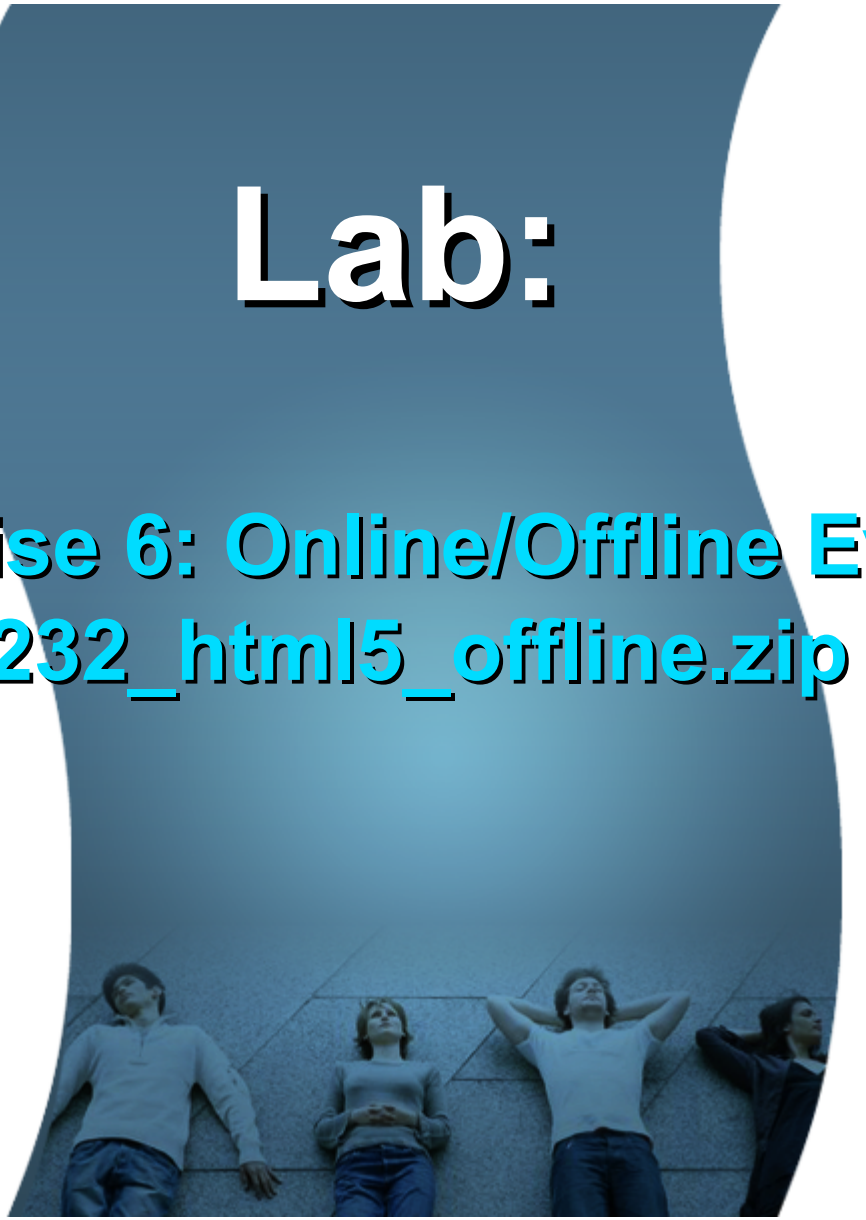
- *navigator.onLine* property
  - The events online and offline are fired when the value of this attribute changes
  - This attribute is inherently unreliable. A computer can be connected to a network without having Internet access
  - Only Chrome sets *navigator.onLine* property to “online”/“offline” when connectivity changes. Both Safari and Firefox never set the flag to false even if you remove the internet connection

Online/Offline Detection (Chrome)

```
function displayStatus(event) {
  statusElem.className = navigator.onLine ? 'online' : 'offline';
  statusElem.innerHTML = navigator.onLine ? 'online' : 'offline';
  state.innerHTML += '<li>New event: ' + event.type + '</li>';
}

window.addEventListener('online', displayStatus);
window.addEventListener('offline', displayStatus);
```

Lab:

Exercise 6: Online/Offline Event 1232_html5_offline.zip



Learn with Passion!
JPassion.com