HTML5 Web Storage, IndexedDB and File System

File System

Craig Shoemaker
craigshoemaker.net
@craigshoemaker

pluralsight
hardcore developer training

Contents

Introduction

Web Storage

IndexedDB

File System

Libraries

Introduction, Concepts & Initialization

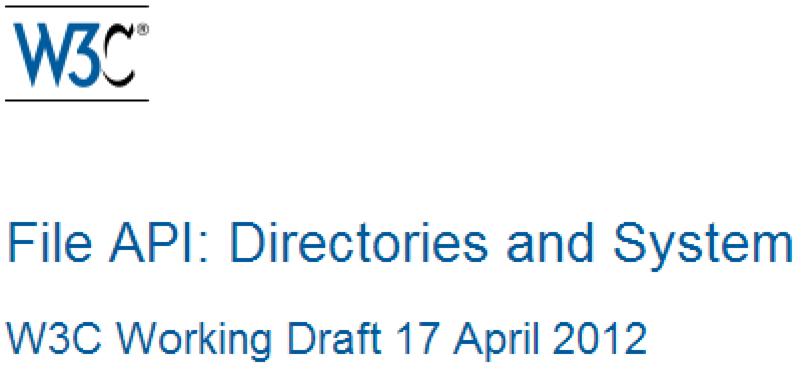
What is HTML5 File System?

File System Features

Asynchronous API

Sandboxed Data

Capacity

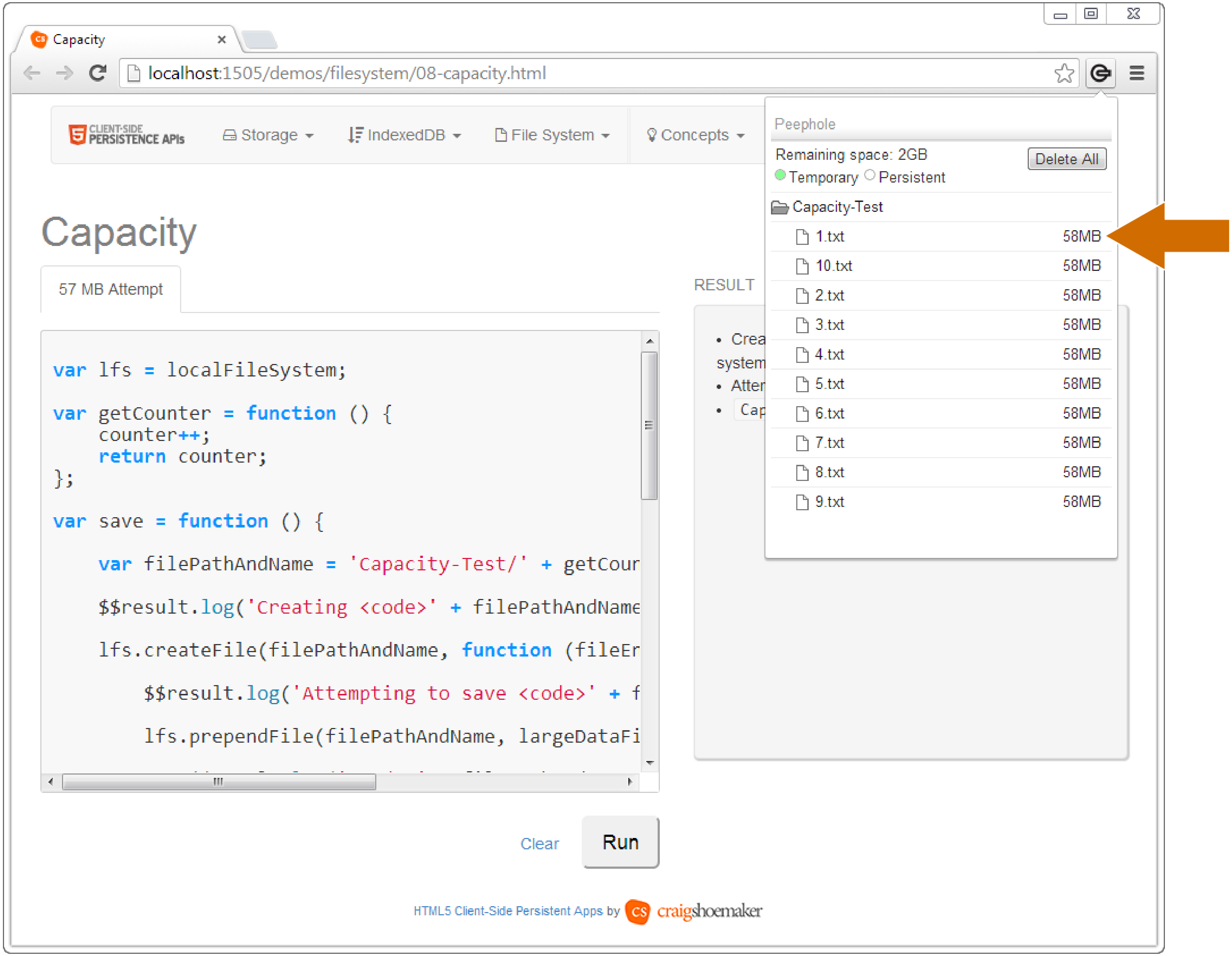


File API: Directories and System

W3C Working Draft 17 April 2012

File System Specification

http://bit.ly/1dhEL2b


Capacity
localhost:1505/demos/filesystem/08-capacity.html
CLIENT-SIDE PERSISTENCE APIs
Storage
IndexedDB
File System
Concepts
Peephole
Delete All
RESULT
Clear
Run
HTML5 Client-Side Persistent Apps by craigshoemaker

Capacity

57 MB Attempt

```
var lfs = localFileSystem;

var getCounter = function () {
    counter++;
    return counter;
};

var save = function () {

    var filePathAndName = 'Capacity-Test/' + getCour

    $$result.log('Creating <code>' + filePathAndName

    lfs.createFile(filePathAndName, function (fileEr

        $$result.log('Attempting to save <code>' + f

        lfs.prependFile(filePathAndName, largeDataFi
```

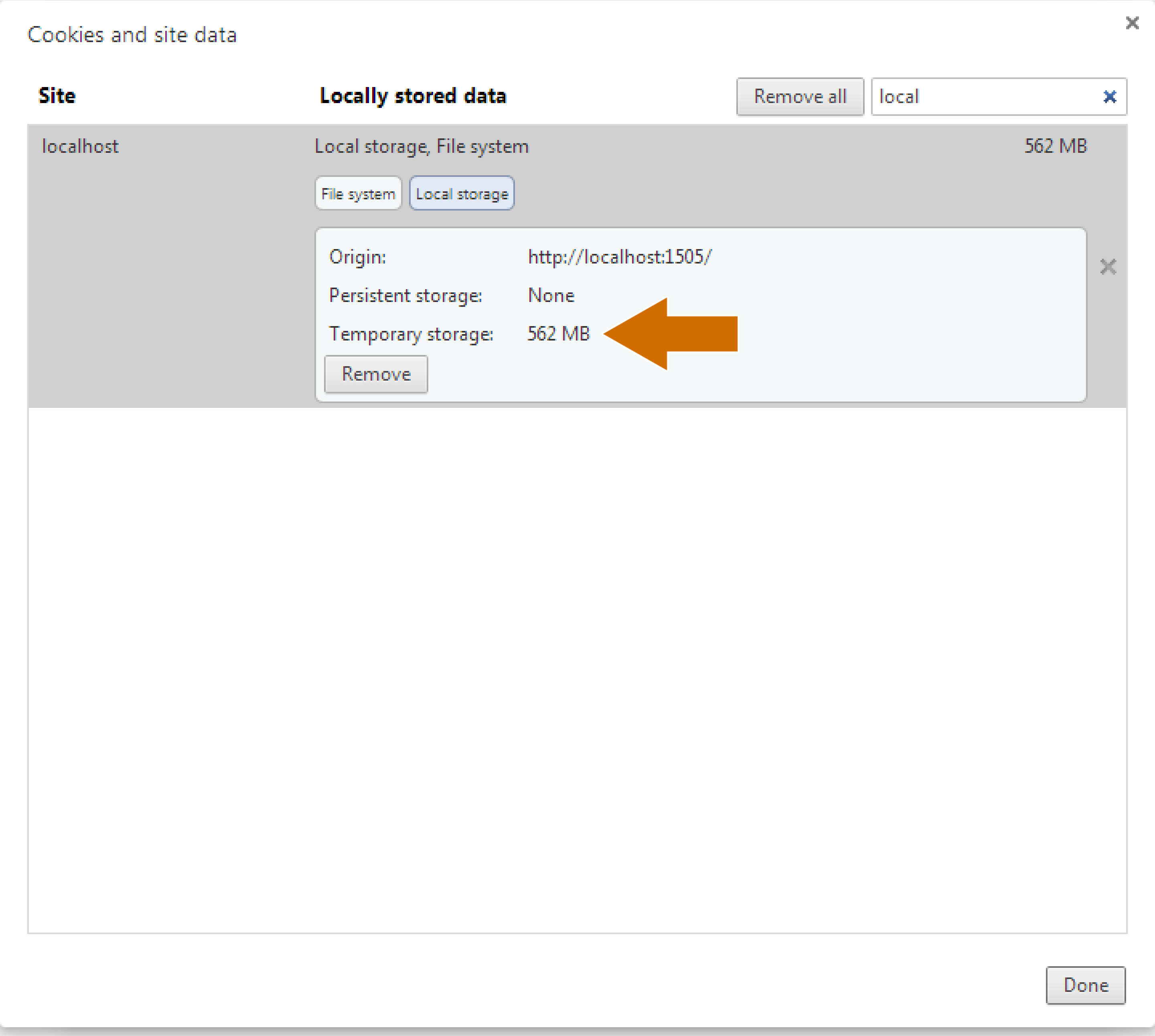
- Crea
system
- Atter
- Cap

Remaining space: 2GB

● Temporary ○ Persistent

Capacity-Test

1.txt	58MB
10.txt	58MB
2.txt	58MB
3.txt	58MB
4.txt	58MB
5.txt	58MB
6.txt	58MB
7.txt	58MB
8.txt	58MB
9.txt	58MB



Cookies and site data

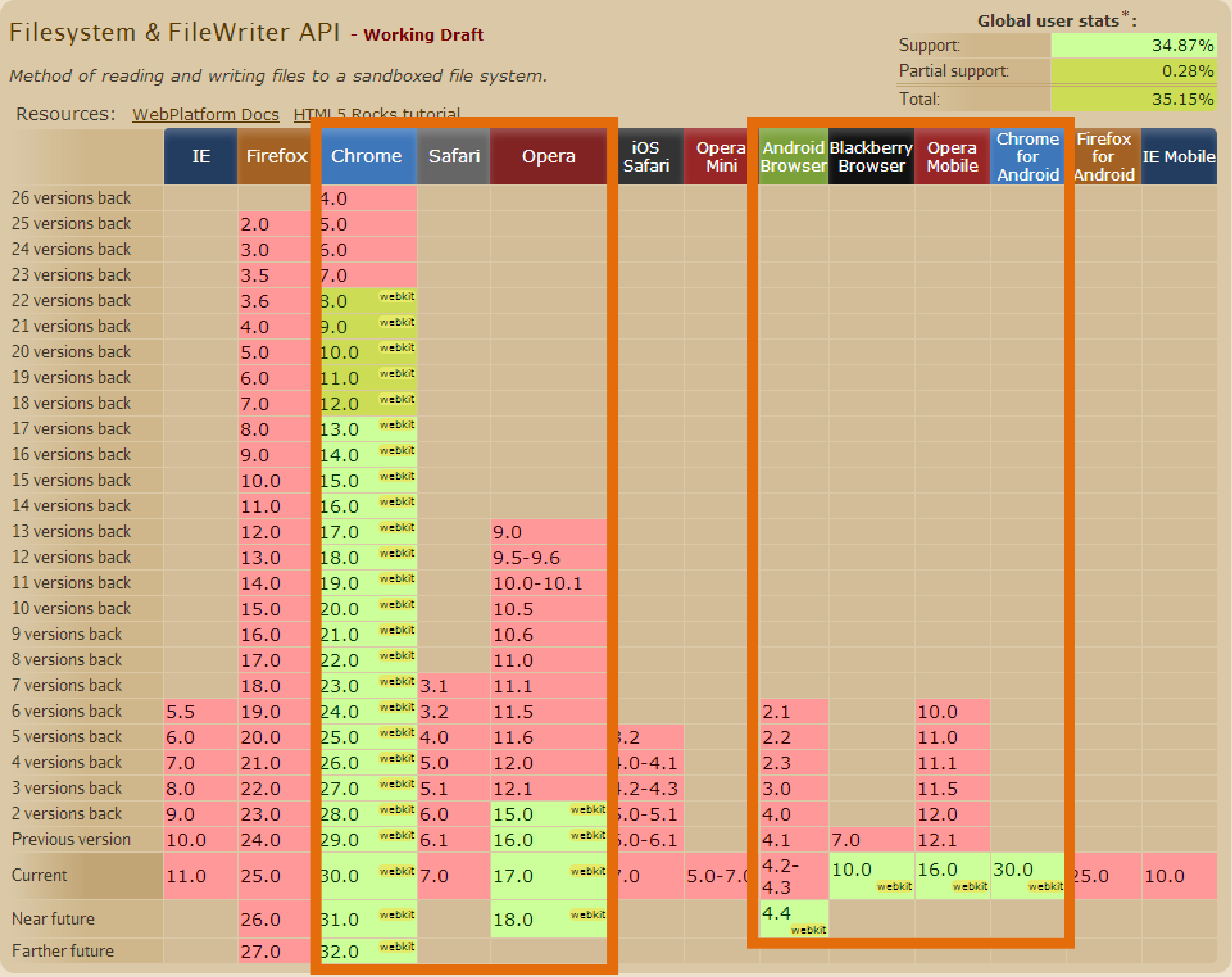
Site	Locally stored data	
localhost	Local storage, File system	562 MB

Origin: http://localhost:1505/

Persistent storage: None

Temporary storage: 562 MB

Browser Support



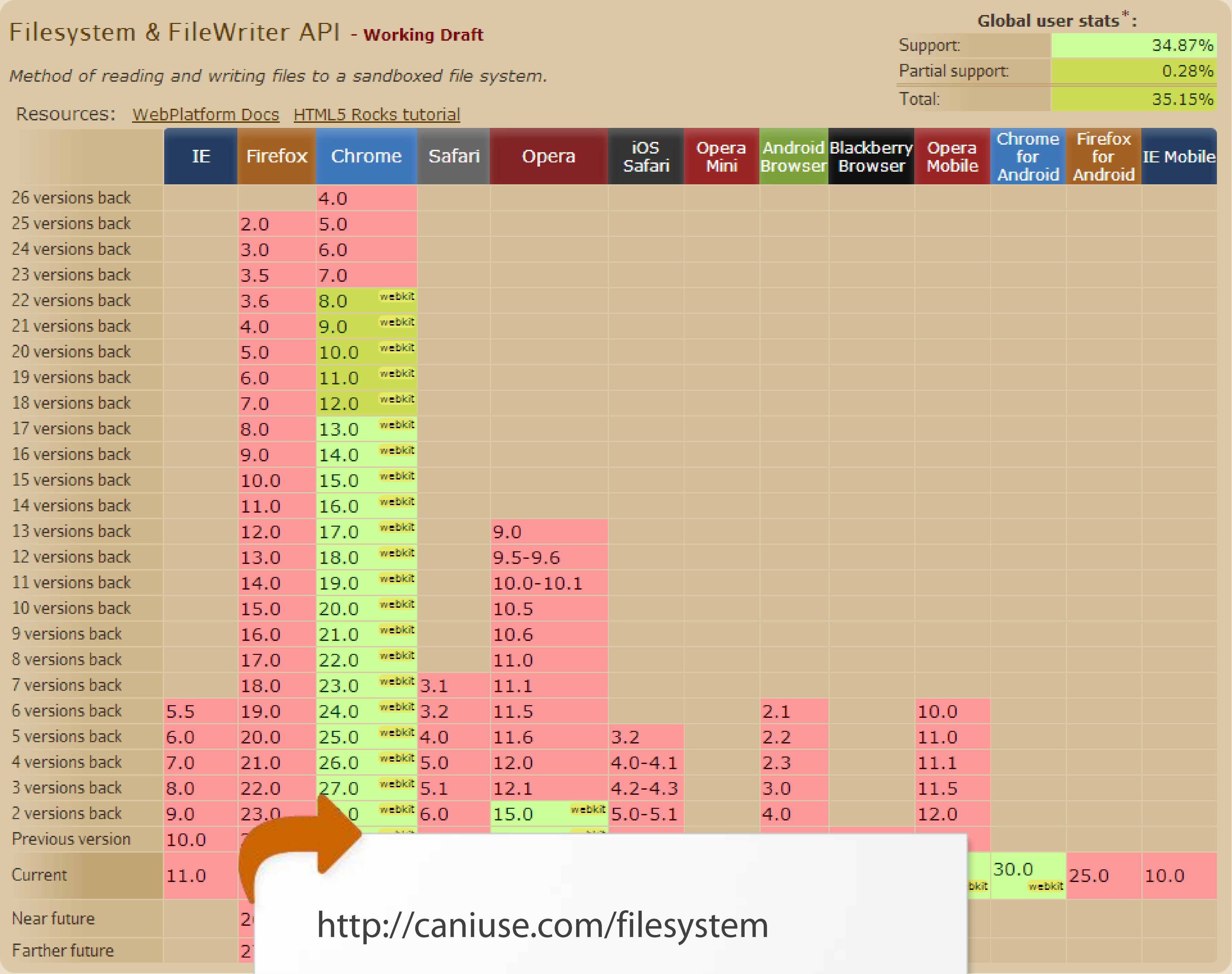
Filesystem & FileWriter API - Working Draft

Method of reading and writing files to a sandboxed file system.

Resources: WebPlatform Docs HTML5 Rocks tutorial

Global user stats*:	
Support:	34.87%
Partial support:	0.28%
Total:	35.15%

	IE	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini	Android Browser	Blackberry Browser	Opera Mobile	Chrome for Android	Firefox for Android	IE Mobile
26 versions back			4.0										
25 versions back		2.0	5.0										
24 versions back		3.0	6.0										
23 versions back		3.5	7.0										
22 versions back		3.6	8.0 webkit										
21 versions back		4.0	9.0 webkit										
20 versions back		5.0	10.0 webkit										
19 versions back		6.0	11.0 webkit										
18 versions back		7.0	12.0 webkit										
17 versions back		8.0	13.0 webkit										
16 versions back		9.0	14.0 webkit										
15 versions back		10.0	15.0 webkit										
14 versions back		11.0	16.0 webkit										
13 versions back		12.0	17.0 webkit		9.0								
12 versions back		13.0	18.0 webkit		9.5-9.6								
11 versions back		14.0	19.0 webkit		10.0-10.1								
10 versions back		15.0	20.0 webkit		10.5								
9 versions back		16.0	21.0 webkit		10.6								
8 versions back		17.0	22.0 webkit		11.0								
7 versions back		18.0	23.0 webkit	3.1	11.1								
6 versions back	5.5	19.0	24.0 webkit	3.2	11.5			2.1		10.0			
5 versions back	6.0	20.0	25.0 webkit	4.0	11.6	.2		2.2		11.0			
4 versions back	7.0	21.0	26.0 webkit	5.0	12.0	.0-4.1		2.3		11.1			
3 versions back	8.0	22.0	27.0 webkit	5.1	12.1	.2-4.3		3.0		11.5			
2 versions back	9.0	23.0	28.0 webkit	6.0	15.0 webkit	.0-5.1		4.0		12.0			
Previous version	10.0	24.0	29.0 webkit	6.1	16.0 webkit	.0-6.1		4.1	7.0	12.1			
Current	11.0	25.0	30.0 webkit	7.0	17.0 webkit	.0	5.0-7.0	4.2-4.3	10.0 webkit	16.0 webkit	30.0 webkit	25.0	10.0
Near future		26.0	31.0 webkit		18.0 webkit			4.4 webkit					
Farther future		27.0	32.0 webkit										



Filesystem & FileWriter API - Working Draft

Method of reading and writing files to a sandboxed file system.

Resources: WebPlatform Docs HTML5 Rocks tutorial

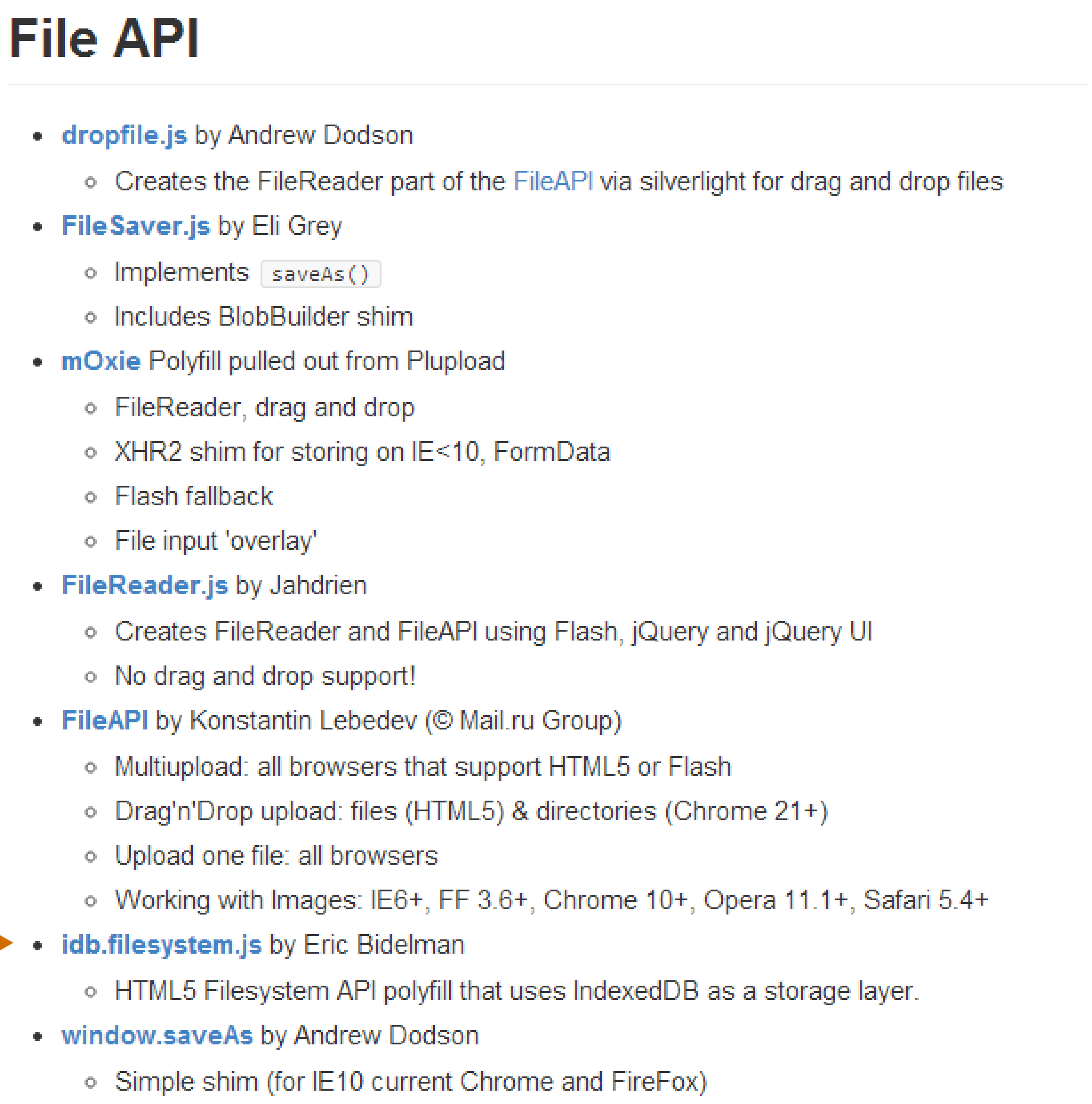
Global user stats*:

Support:	34.87%
Partial support:	0.28%
Total:	35.15%

	IE	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini	Android Browser	Blackberry Browser	Opera Mobile	Chrome for Android	Firefox for Android	IE Mobile
26 versions back			4.0										
25 versions back		2.0	5.0										
24 versions back		3.0	6.0										
23 versions back		3.5	7.0										
22 versions back		3.6	8.0 webkit										
21 versions back		4.0	9.0 webkit										
20 versions back		5.0	10.0 webkit										
19 versions back		6.0	11.0 webkit										
18 versions back		7.0	12.0 webkit										
17 versions back		8.0	13.0 webkit										
16 versions back		9.0	14.0 webkit										
15 versions back		10.0	15.0 webkit										
14 versions back		11.0	16.0 webkit										
13 versions back		12.0	17.0 webkit		9.0								
12 versions back		13.0	18.0 webkit		9.5-9.6								
11 versions back		14.0	19.0 webkit		10.0-10.1								
10 versions back		15.0	20.0 webkit		10.5								
9 versions back		16.0	21.0 webkit		10.6								
8 versions back		17.0	22.0 webkit		11.0								
7 versions back		18.0	23.0 webkit	3.1	11.1								
6 versions back	5.5	19.0	24.0 webkit	3.2	11.5			2.1		10.0			
5 versions back	6.0	20.0	25.0 webkit	4.0	11.6	3.2		2.2		11.0			
4 versions back	7.0	21.0	26.0 webkit	5.0	12.0	4.0-4.1		2.3		11.1			
3 versions back	8.0	22.0	27.0 webkit	5.1	12.1	4.2-4.3		3.0		11.5			
2 versions back	9.0	23.0	0 webkit	6.0	15.0 webkit	5.0-5.1		4.0		12.0			
Previous version	10.0												
Current	11.0									bkit	30.0 webkit	25.0	10.0
Near future		2											
Farther future		2											

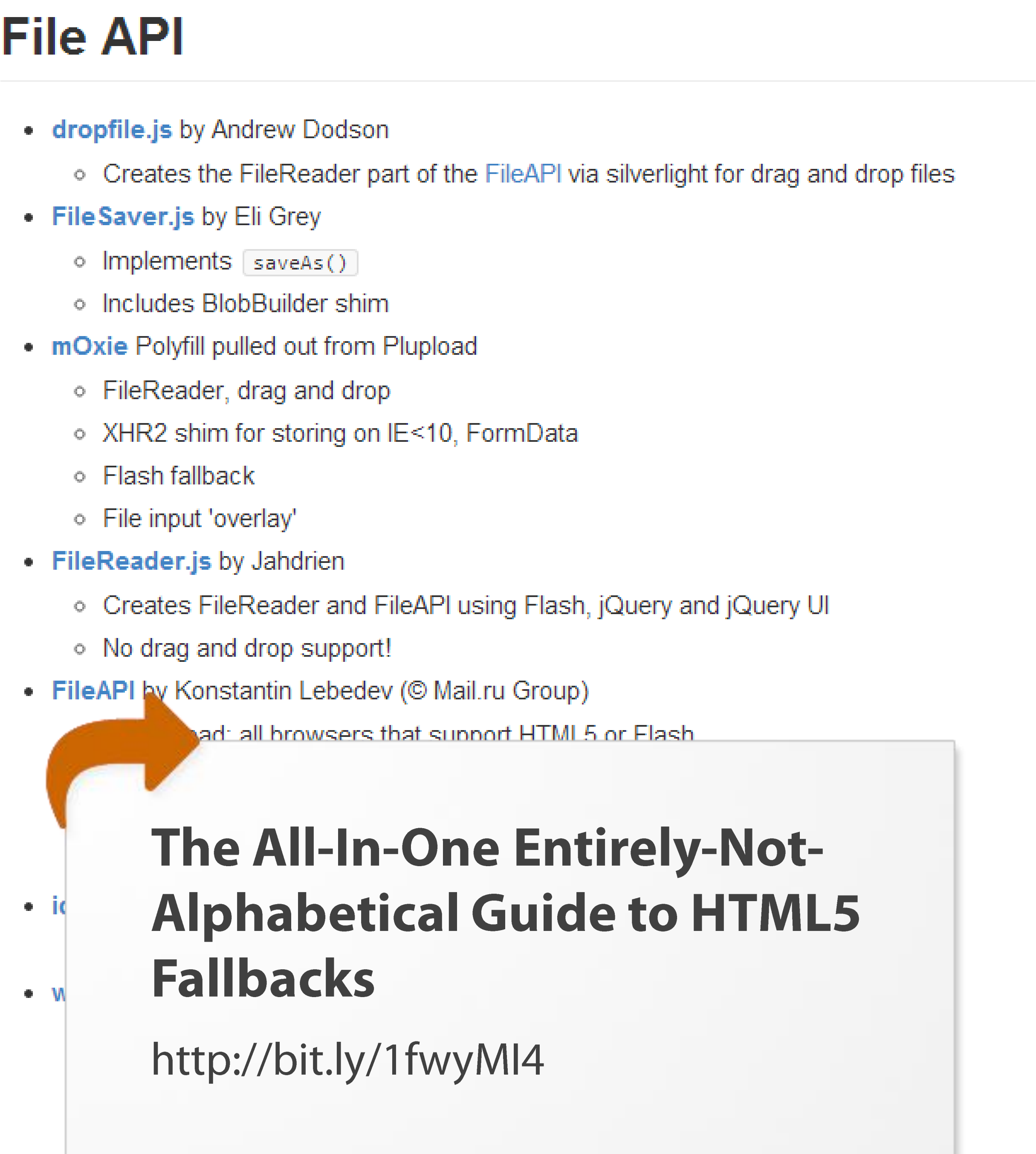
http://caniuse.com/filesystem

Fallbacks & Polyfills



File API

- **dropfile.js** by Andrew Dodson
  - Creates the FileReader part of the FileAPI via silverlight for drag and drop files
- **FileSaver.js** by Eli Grey
  - Implements `saveAs()`
  - Includes BlobBuilder shim
- **mOxie** Polyfill pulled out from Plupload
  - FileReader, drag and drop
  - XHR2 shim for storing on IE<10, FormData
  - Flash fallback
  - File input 'overlay'
- **FileReader.js** by Jahdrien
  - Creates FileReader and FileAPI using Flash, jQuery and jQuery UI
  - No drag and drop support!
- **FileAPI** by Konstantin Lebedev (© Mail.ru Group)
  - Multiupload: all browsers that support HTML5 or Flash
  - Drag'n'Drop upload: files (HTML5) & directories (Chrome 21+)
  - Upload one file: all browsers
  - Working with Images: IE6+, FF 3.6+, Chrome 10+, Opera 11.1+, Safari 5.4+
- **idb.filesystem.js** by Eric Bidelman
  - HTML5 Filesystem API polyfill that uses IndexedDB as a storage layer.
- **window.saveAs** by Andrew Dodson
  - Simple shim (for IE10 current Chrome and FireFox)



File API

- dropfile.js by Andrew Dodson
  - Creates the FileReader part of the FileAPI via silverlight for drag and drop files
- FileSaver.js by Eli Grey
  - Implements `saveAs()`
  - Includes BlobBuilder shim
- mOxie Polyfill pulled out from Plupload
  - FileReader, drag and drop
  - XHR2 shim for storing on IE<10, FormData
  - Flash fallback
  - File input 'overlay'
- FileReader.js by Jahdrien
  - Creates FileReader and FileAPI using Flash, jQuery and jQuery UI
  - No drag and drop support!
- FileAPI by Konstantin Lebedev (© Mail.ru Group)
  - ad: all browsers that support HTML5 or Flash
- ic
- w

The All-In-One Entirely-Not-Alphabetical Guide to HTML5 Fallbacks

http://bit.ly/1fwyMI4

Caveats

Browser Support

Callback Heavy

Not Transactional

No Search Support

Case sensitivity

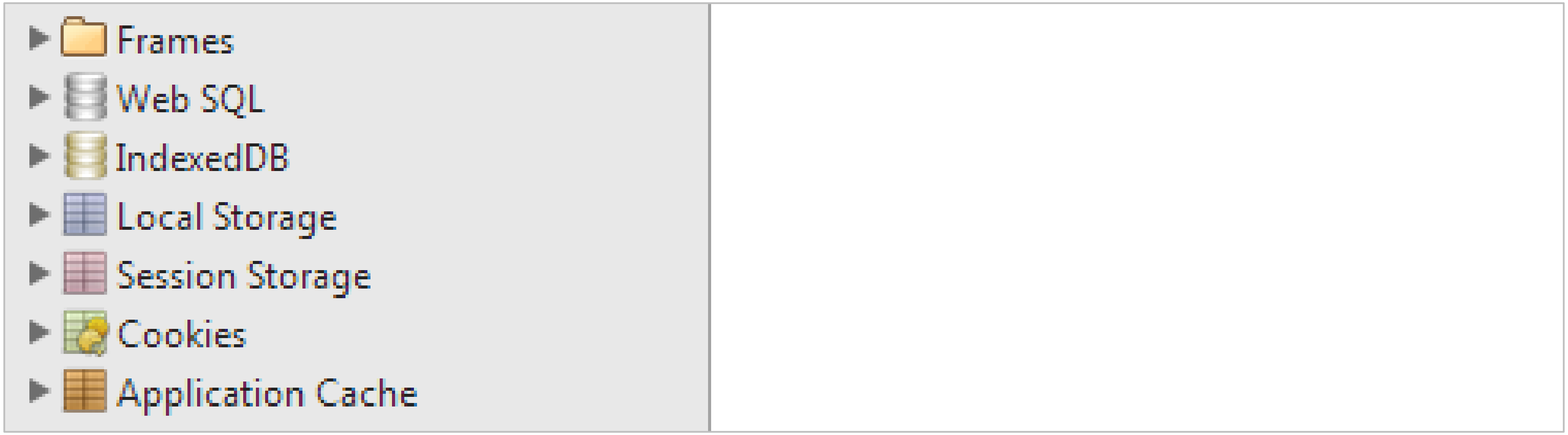
Browser-Specific

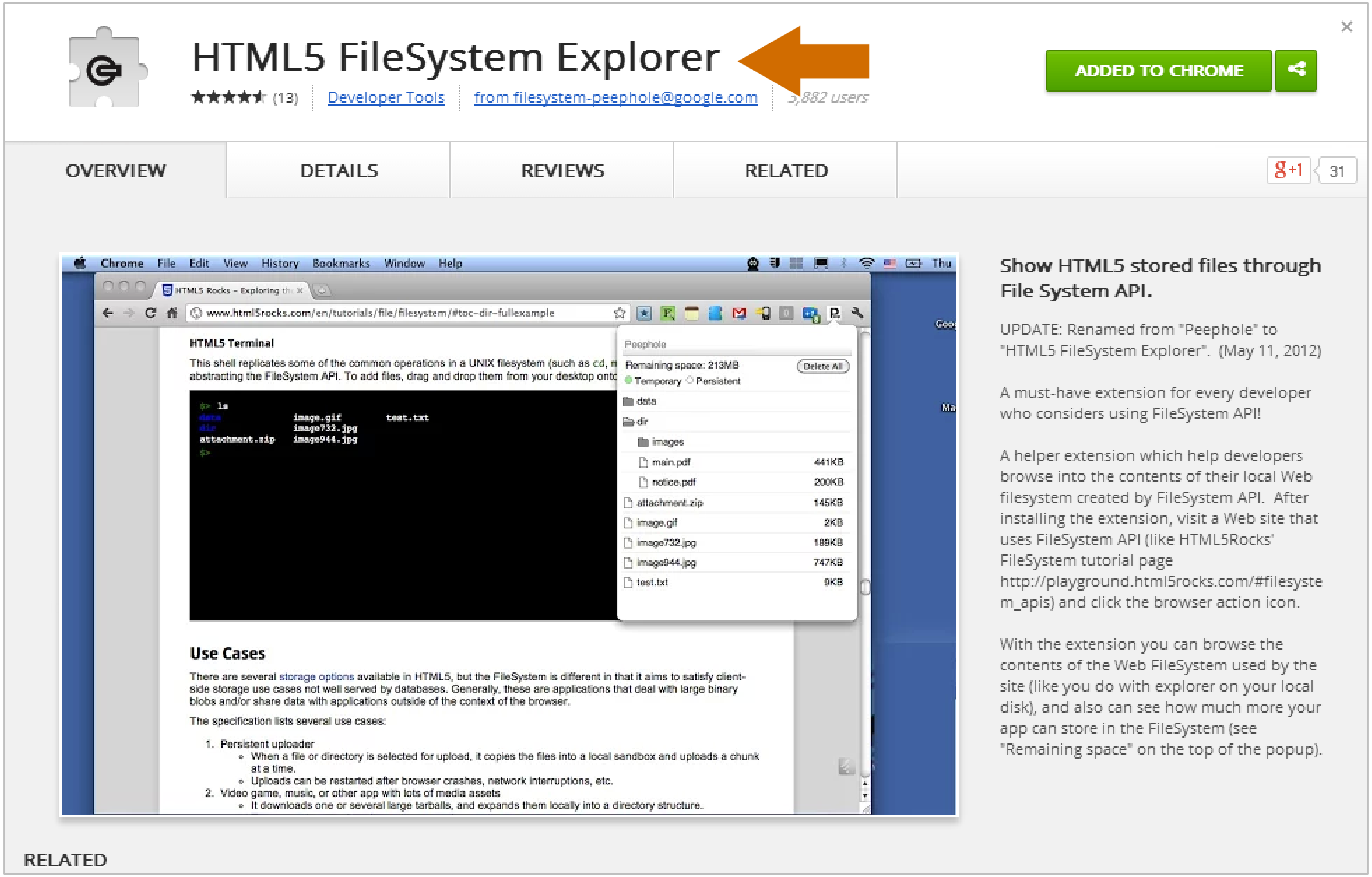
Storage Types

Temporary

Persistent

Inspecting the File System


Frames
Web SQL
IndexedDB
Local Storage
Session Storage
Cookies
Application Cache



HTML5 FileSystem Explorer

Chrome
File
Edit
View
History
Bookmarks
Window
Help
Thu
HTML5 Rocks - Exploring th ×
www.html5rocks.com/en/tutorials/file/filesystem/#toc-dir-fullexample

HTML5 Terminal

This shell replicates some of the common operations in a UNIX filesystem (such as cd, m abstracting the FileSystem API. To add files, drag and drop them from your desktop onto

```
$> ls
data                image.gif       test.txt
dir                 image732.jpg
attachment.zip      image944.jpg
$>
```

Peephole

Remaining space: 213MB Delete All

● Temporary ○ Persistent

data	
dir	
images	
main.pdf	441KB
notice.pdf	200KB
attachment.zip	145KB
image.gif	2KB
image732.jpg	189KB
image944.jpg	747KB
test.txt	9KB

Use Cases

There are several storage options available in HTML5, but the FileSystem is different in that it aims to satisfy client-side storage use cases not well served by databases. Generally, these are applications that deal with large binary blobs and/or share data with applications outside of the context of the browser.

The specification lists several use cases:

1. Persistent uploader
   - When a file or directory is selected for upload, it copies the files into a local sandbox and uploads a chunk at a time.
   - Uploads can be restarted after browser crashes, network interruptions, etc.
2. Video game, music, or other app with lots of media assets
   - It downloads one or several large tarballs, and expands them locally into a directory structure.

Show HTML5 stored files through File System API.

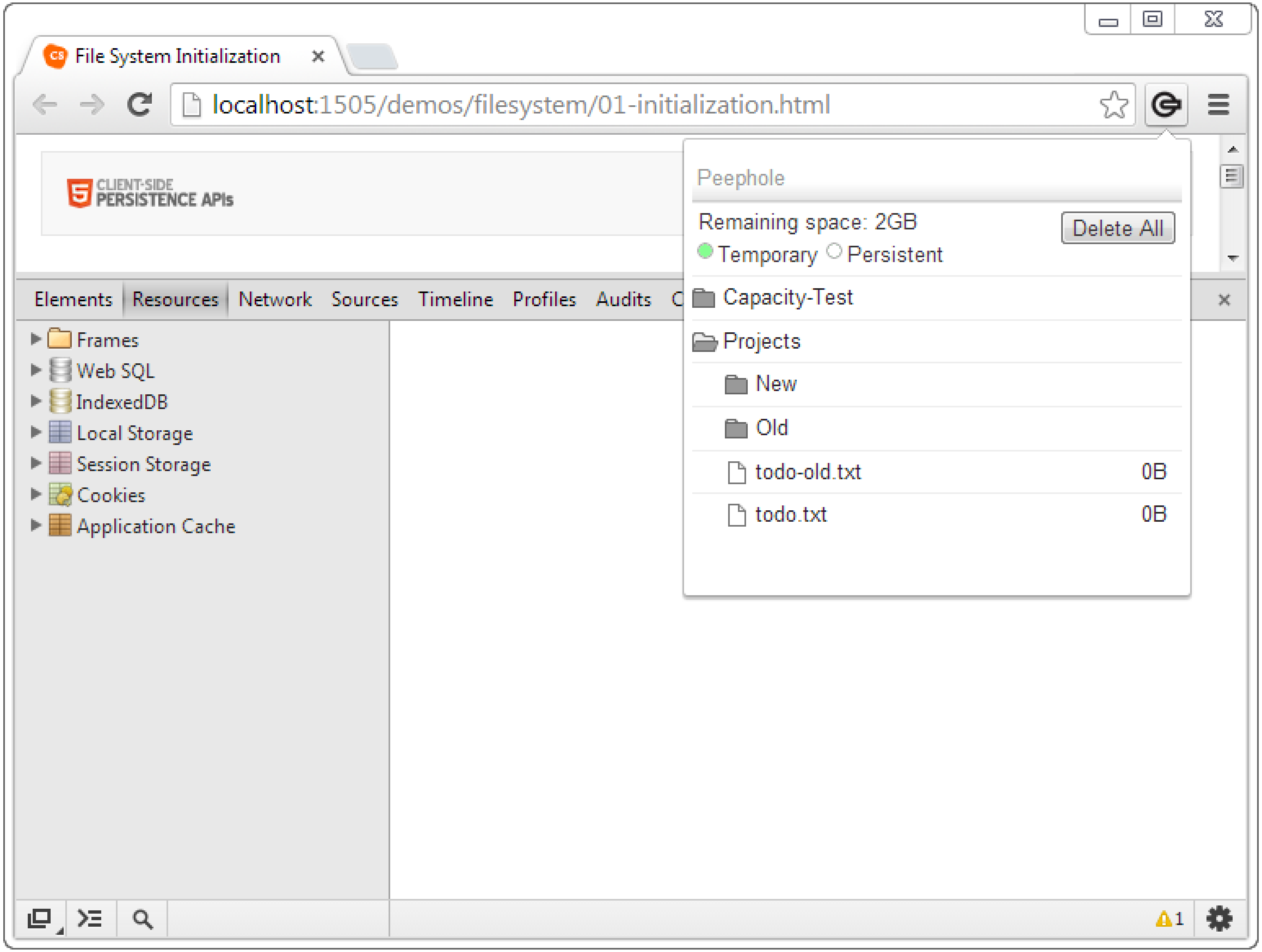
UPDATE: Renamed from "Peephole" to "HTML5 FileSystem Explorer". (May 11, 2012)

A must-have extension for every developer who considers using FileSystem API!

A helper extension which help developers browse into the contents of their local Web filesystem created by FileSystem API. After installing the extension, visit a Web site that uses FileSystem API (like HTML5Rocks' FileSystem tutorial page http://playground.html5rocks.com/#filesystem_apis) and click the browser action icon.

With the extension you can browse the contents of the Web FileSystem used by the site (like you do with explorer on your local disk), and also can see how much more your app can store in the FileSystem (see "Remaining space" on the top of the popup).

RELATED

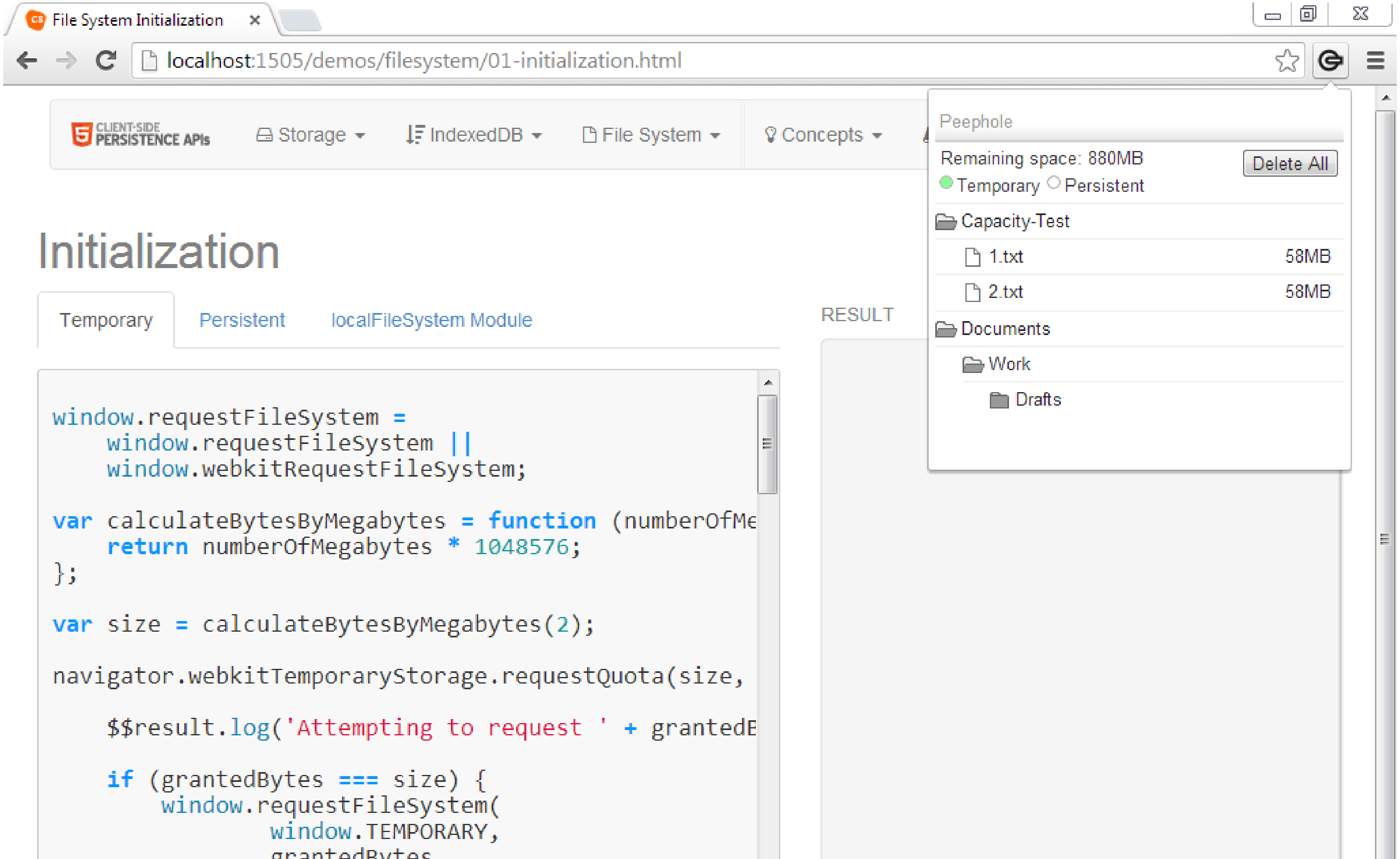


Remaining space: 2GB

● Temporary ○ Persistent

- Capacity-Test
- Projects
  - New
  - Old
  - todo-old.txt 0B
  - todo.txt 0B

Summary



File System Initialization
localhost:1505/demos/filesystem/01-initialization.html
CLIENT-SIDE PERSISTENCE APIs
Storage
IndexedDB
File System
Concepts
Temporary
Persistent
localFileSystem Module
RESULT
Peephole
Delete All

Initialization

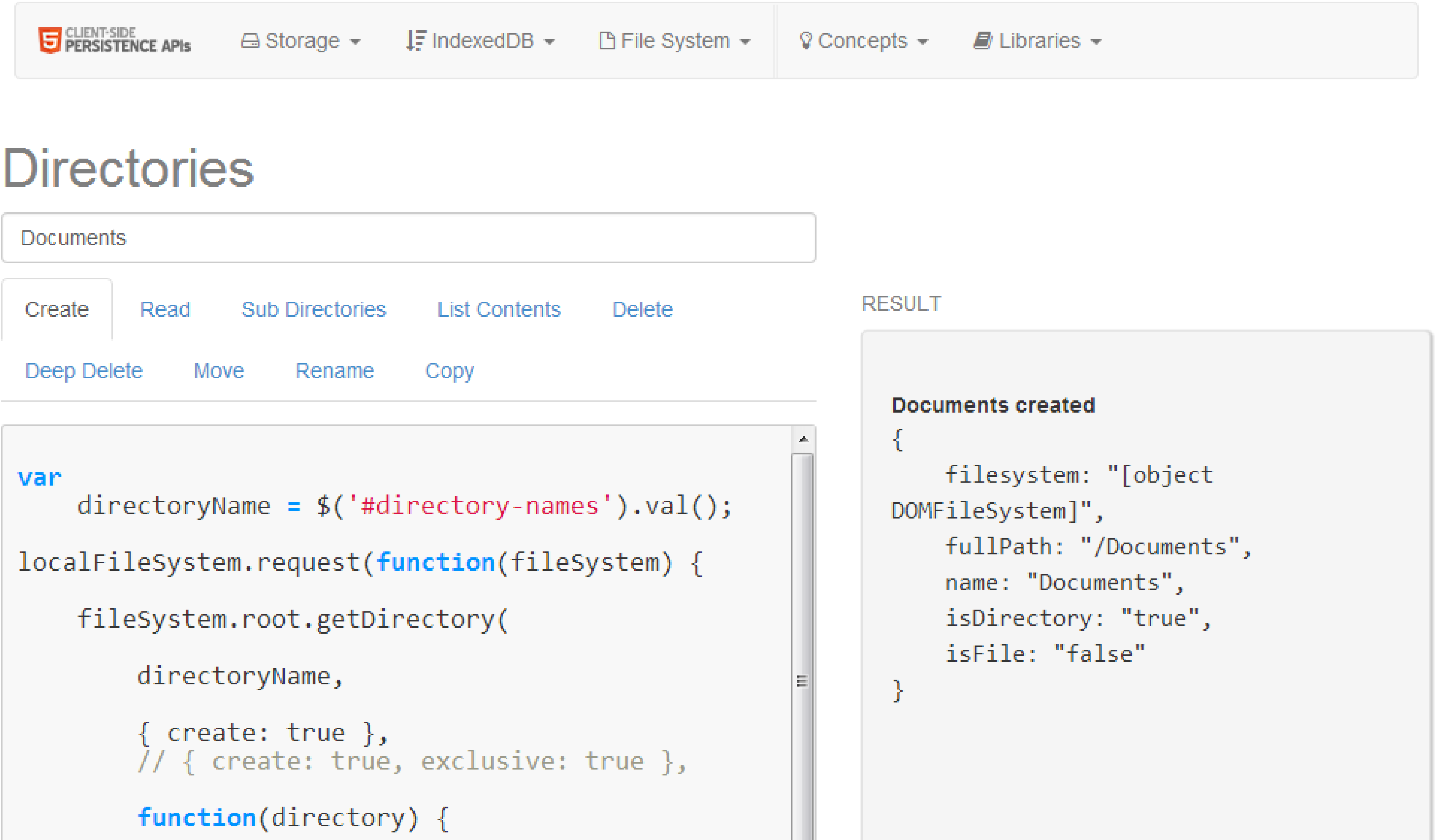
Remaining space: 880MB

● Temporary ○ Persistent

- Capacity-Test
  - 1.txt 58MB
  - 2.txt 58MB
- Documents
  - Work
    - Drafts

```
window.requestFileSystem =
    window.requestFileSystem ||
    window.webkitRequestFileSystem;

var calculateBytesByMegabytes = function (numberOfMe
    return numberOfMegabytes * 1048576;
};

var size = calculateBytesByMegabytes(2);

navigator.webkitTemporaryStorage.requestQuota(size,

    $$result.log('Attempting to request ' + grantedB

    if (grantedBytes === size) {
        window.requestFileSystem(
                window.TEMPORARY,
                grantedBytes
```

Up Next...



Directories

Documents

```
var
    directoryName = $('#directory-names').val();

localFileSystem.request(function(fileSystem) {

    fileSystem.root.getDirectory(

        directoryName,

        { create: true },
        // { create: true, exclusive: true },

        function(directory) {
```

```
Documents created
{
    filesystem: "[object
DOMFileSystem]",
    fullPath: "/Documents",
    name: "Documents",
    isDirectory: "true",
    isFile: "false"
}
```