Storage

- Persistent local storage is where native client applications have held an advantage over web applications.
- The operating system typically provides a layer for storing and retrieving application-specific data
- These values may be stored in the registry, INI files, XML files, or some other place
- If your native client application needs local storage beyond key/value pairs, you can embed your own database, invent your own file format, or any number of other solutions.

History

- Microsoft invented DHTML Behaviors, and one of these behaviors was called userData.
- userData allows web pages to store up to 64 KB of data per domain, in a hierarchical XML-based structure
- Trusted domains, such as intranet sites, can store 10 times that amount
- IE does not present any form of permissions dialog, and there is no allowance for increasing the amount of storage available.

History

In 2002, Adobe introduced a feature in Flash 6 that gained the unfortunate and misleading name of “Flash cookies.”

- The feature is properly known as Local Shared Objects
- Briefly, it allows Flash objects to store up to 100 KB of data per domain
- Flash gives each domain 100 KB of storage for free.
- Beyond that, it prompts the user for each order of magnitude increase in data storage (1 Mb, 10 Mb, and so on).

History

In 2007, Google launched Gears, an open source browser plugin aimed at providing additional capabilities in browsers.

- Gears provided an API to an embedded SQL database based on SQLite.
- By 2010, Google had shifted efforts toward bringing all of the Gears capabilities into web standards like HTML5

Cookies

Cookies were invented early in the web’s history, and indeed they can be used for persistent local storage of small amounts of data. But they have three potentially dealbreaking downsides:

- Included with every HTTP request, slowing down web applications by transmitting the same data over and over
- Sending data unencrypted over the internet (unless your entire web application is served over SSL)
- limited to about 4 KB of data — enough to slow down your application, but not enough to be useful

Session Storage

Session storage is intended for short-lived data

- Data stored in the sessionStorage object is shared only with pages from the same domain
- Access to the session storage is available through the sessionStorage global object
- Does not persist after the window/tab is closed
- Every time a user opens a page in a new window/tab, the browser creates a new session storage

Local Storage

- Allows to store data for more than a single session
- Access to the local storage is available through the localStorage global object
- Its functionally is identical to the sessionStorage, except that data stored in the local storage area is persistent
- Shared across multiple windows/tabs

Limitations

- 5 megabytes storage space each origin by default
- QUOTA_EXCEEDED_ERR is the exception if you exceed your storage quota of 5 megabytes
- no browser supports any mechanism for web developers to request more storage space

Compatibility

- Chrome 5.0+
- Firefox 3.5+
- Safari 4.0+
- Internet Explorer 8+

Storage

- is based on named key/value pairs
- You store data based on a named key, then you can retrieve that data with the same key
- The named key is a string.
- The data can be any type supported by JavaScript, including strings, Booleans, integers, or floats

Set Value

```
localStorage.setItem("key", value);
localStorage["key"] = value;
```

- the data is stored as string.

Get Value

```javascript
var value = localStorage.getItem("key");
var value = localStorage["key"];
```

- If you are storing and retrieving anything other than strings, use functions like parseInt() or parseFloat() to coerce your retrieved data into the expected JavaScript datatype.

Delete Value

```
localStorage.removeItem("key");
```

- Calling removeItem() with a non-existent key will do nothing.

Clear All Values

```
localStorage.clear();
```

Local Storage and Objects

```javascript
var car = {};
car.wheels = 4;
car.doors = 2;
car.sound = 'vroom';
car.name = 'Lightning McQueen';
localStorage.setItem( 'car', car );
console.log( localStorage.getItem( 'car' ) );
```

Trying this out in the console shows that the data is stored as its string representation "[object Object]" and not the real object information

Storing objects as JSON

```
var car = {};
car.wheels = 4;
car.doors = 2;
car.sound = 'vroom';
car.name = 'Lightning McQueen';
localStorage.setItem( 'car', JSON.stringify(car) );
console.log( JSON.parse( localStorage.getItem( 'car' ) ) )
```

You can work around this by using the native JSON.stringify() and JSON.parse() methods

Beyond Named Key-Value Pairs

Indexed Database API, formerly known as WebSimpleDB, now affectionately known as IndexedDB.

- The Indexed Database API exposes what’s called an object store
- An object store shares many concepts with a SQL database
- There are “databases” with “records,” and each record has a set number of fields.
- Each field has a specific datatype, which is defined when the database is created
- You can select a subset of records, then enumerate them with a cursor.
- Changes to the object store are handled within transactions.

IndexDB

- Both IndexDB and LocalStorage allow saving objects in browsers
- Both support offline use
- Although more powerful, IndexDB is much harder to learn
- For many usecases, the functionality of Local Storage is enough

https://developer.mozilla.org/en-US/docs/Web/API/IndexedDB_API

Summary

- LocalStorage is a relatively new browser API
- LocalSotrage API is widely adapted and easy to use
- It support offline / local storage up to 5 MB of data
- String and JSON are the preferred formats