DIG 4104C: WEB DESIGN WORKSHOP

Lecture 08: Extending web apps offline: Client storage with the Web Storage API, Client side database management

Summer 2015

D. Novatnak

Outline

Objectives

- Expand student knowledge of the Web Storage API and the Indexed Database API
- Expose students to the features of the Web Storage API and the Indexed Database API in HTML5
- Expose students to the language and syntax necessary to leverage the Web Storage API and the Indexed Database API in their own applications
- Extend the student's knowledge of data storage within the browser

At the end of this lecture, you should be able to….

- Utilize the Web Storage API and the Indexed Database API in their own projects
- Discuss the capabilities of both API
- Discuss the syntax of both API

HTML5 WEB STORAGE API

TAKE IT OFFLINE

The old and the new

- Cookie - Are a built-in way of sending text values back and forth from server to client.
  - Servers can use the values they put into these cookies to track user information across web pages.
- Cookies have some well-known drawbacks:
  - Cookies are extremely limited in size.
    - Only about 4KB of data can be set
  - Cookies are transmitted back and forth from server to browser on every request scoped to that cookie.
    - Cookie data is visible on the network, making them a security risk when not encrypted
    - Any data persisted as cookies will be consuming network bandwidth every time a URL is loaded.

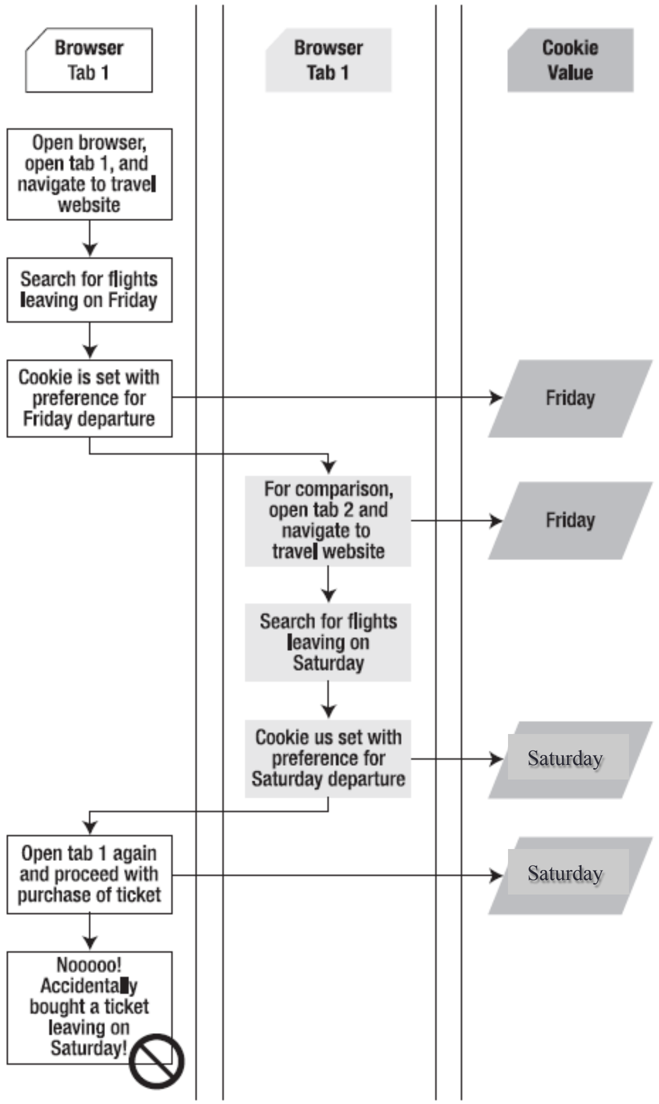
Cookies in a nutshell

- Potential drawbacks
  - Very limited space (4 kb per domain)
  - Cookie information passed unconditionally back and forth between client and server
  - Possibility of state inconsisten-cies between client and server

Plugging Data Leaks

- It is very common for users to open multiple windows as they shop for travel deals, comparing flights from different vendors for the same departure time.
- If a user switches back and forth between browser windows while comparing prices and availability, they are likely to set cookie values in one window that will be unexpectedly applied to another window served from the same URL on its next operation.

Plugging Data Leaks (CONT.)


Browser Tab 1
Browser Tab 1
Cookie Value
Open browser, open tab 1, and navigate to travel website
Search for flights leaving on Friday
Cookie is set with preference for Friday departure
Friday
For comparison, open tab 2 and navigate to travel website
Friday
Search for flights leaving on Saturday
Cookie us set with preference for Saturday departure
Saturday
Open tab 1 again and proceed with purchase of ticket
Saturday
Nooooo! Accidentally bought a ticket leaving on Saturday!

Persistent Local Storage

- Native client applications use operating system to store data such as preferences or runtime state
- Stored in registry, INI files, XML or other places using key/value pairs
- Web applications can’t do this

HTML5 Web Storage

- The HTML5 Web Storage concept is supposed to replace cookies with a more up-to-date way of storing data on the client side
- Used through the **HTML5 Web Storage API**
- Often referred to as ”local storage”…

History

IE: DHTML Behaviors

- userData behavior allowed 64K per domain
- Hierarchical XML-based structure

Adobe Flash (2002)

- "Flash cookies" or Local Shared Objects
- Allows Flash objects to store 100K data per domain temporarily

History

AMASS (AJAX Massive Storage System)

- Brad Neuberg
- Flash-to-JavaScript bridge
- Limited by Flash design quirks

Flash 8: ExternalInterface (2006)

- Easier to access Local Shared Objects

AMASS rewritten

- Integrated into Dojo Toolkit: dojox.storage
- 100KB storage
- Prompts user for exponentially increased storage

History

- **Google: Gears (2007)**
  - Open source browser plug-in
  - Provides additional capability in browsers (geolocation API in IE)
  - API to embedded SQL database
  - Unlimited data per domain in SQL database tables
- By 2009 **dojox.storage** could auto-detect and provide unified interface for Flash, Gears, Adobe AIR and early prototype of HTML5 storage (in older version of Firefox)

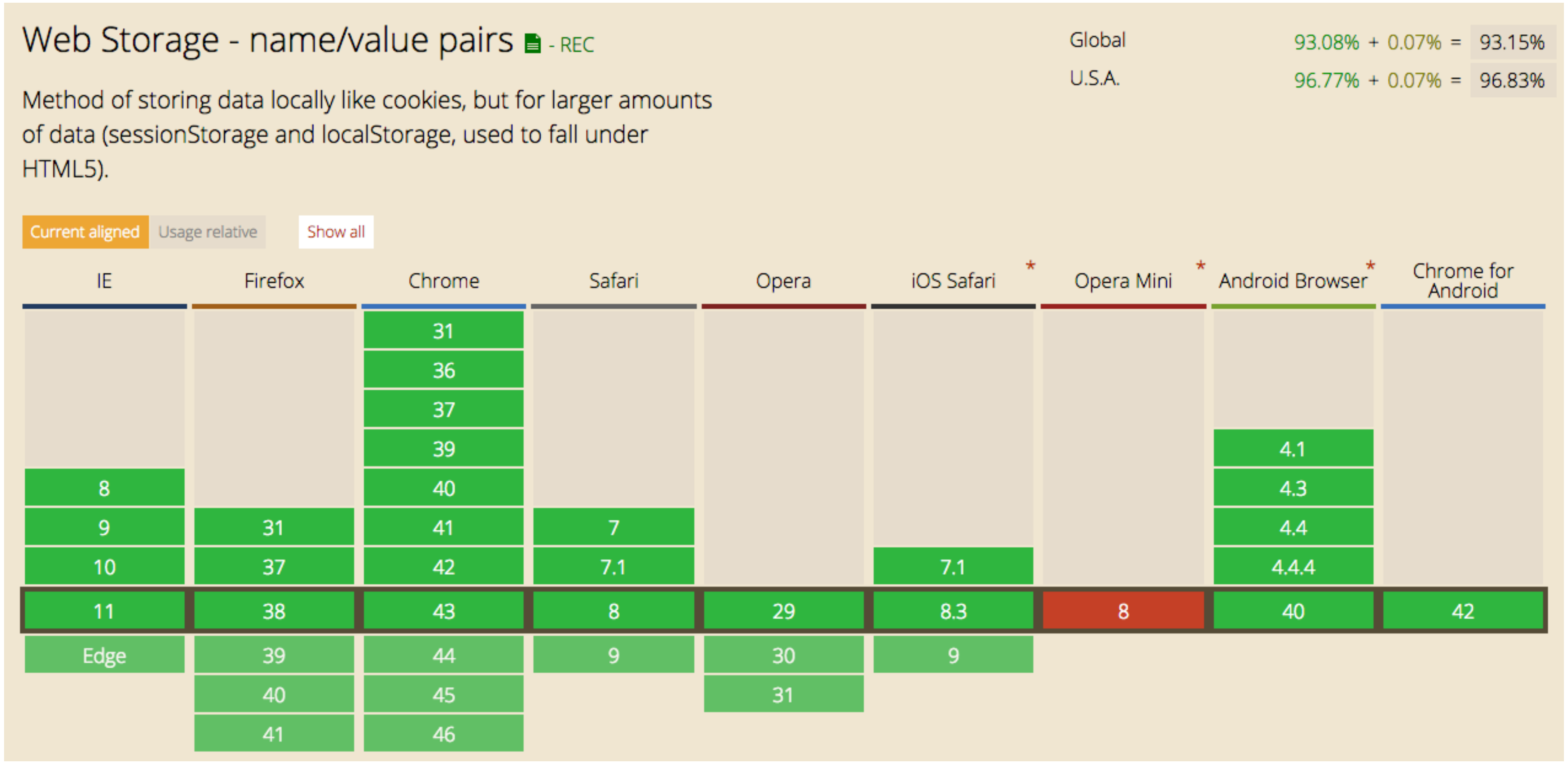
Web Storage

- By using either **sessionStorage** or **localStorage**
  - Developers can choose to let those values survive either across page loads in a single window or tab or across browser restarts, respectively.
  - Stored data is not transmitted across the network, and is easily accessed on return visits to a page
  - Larger values can be persisted using the Web Storage API values as high as a few megabytes.

HTML5 Web Storage

- Main advantages of HTML5 Web storage
  - Several mb (5-10) of storage available per domain
  - Fully controlled by the client, potentially redu-cing network traffic

Browser compatibility



Web Storage - name/value pairs - REC

Global	93.08%	+ 0.07%	=	93.15%
U.S.A.	96.77%	+ 0.07%	=	96.83%

Method of storing data locally like cookies, but for larger amounts of data (sessionStorage and localStorage, used to fall under HTML5).

Current aligned | Usage relative | Show all

IE	Firefox	Chrome	Safari	Opera	iOS Safari *	Opera Mini *	Android Browser *	Chrome for Android
		31						
		36						
		37						
		39					4.1	
8		40					4.3	
9	31	41	7				4.4	
10	37	42	7.1		7.1		4.4.4	
11	38	43	8	29	8.3	8	40	42
Edge	39	44	9	30	9			
	40	45		31				
	41	46						

Using HTML5 Storage - StorageEvent Object

PROPERTY	TYPE	DESCRIPTION
`key`	string	the named key that was added, removed, or modified
`oldValue`	any	the previous value (now overwritten), or null if a new item was added
`newValue`	any	the new value, or null if an item was removed
`url`*	string	the page which called a method that triggered this change

Checking for Browser Support

```
function checkStorageSupport() {
    //sessionStorage
    if (window.sessionStorage) {
        alert('This browser supports sessionStorage');
    } else {
        alert('This browser does NOT support
    sessionStorage');
    }
    //localStorage
    if (window.localStorage) {
        alert('This browser supports localStorage');
    } else {
        alert('This browser does NOT support localStorage');
    }
}
```

Setting and Retrieving Values

- The object implementing the Web Storage API is attached to the window, so window.sessionStorage contains the functions you need to call.
- Setting

```
window.sessionStorage.setItem(‘myFirstKey’, ‘myFirstValue’);
```

- Retrieving

```
alert(window.sessionStorage.getItem(‘myFirstKey’));
```

HTML5 Web Storage

- The Web Storage API is accessed through the global object **localStorage** (supported by ”all” browsers, even IE8 ☺)
- Main methods

```
localStorage.setItem(key, value);
var value = localStorage.getItem(key);
```

HTML5 Web Storage

- Keys and values are always strings…
- …but we know how to convert any object to a string, and vice versa!

```
var asString = JSON.stringify(asObject);
var asObject = JSON.parse(asString);
```

HTML5 Web Storage

- You can also retrieve the number of items in **localStorage**:

```
var count = localStorage.length;
```

- …and retrieve a specific key by its index:

```
var aKey = localStorage.key(7);
```

Local Versus Session Storage

sessionStorage	**localStorage**
Values persist only as long as the window or tab in which they were stored.	Values persist beyond window and browser lifetimes.
Values are only visible within the window or tab that created them.	Values are shared across every window or tab running at the same origin.

Web Storage API Attributes and Functions

```
interface Storage {
   readonly attribute unsigned long length;
   getter DOMString key(in unsigned long index);
   getter any getItem(in DOMString key);
   setter creator void setItem(in DOMString key, in any
   data);
   deleter void removeItem(in DOMString key);
   void clear();
};
```

HTML5 Web Storage

- Since keys are strings, they can be ”anything”
- However, keys must (obviously) be unique!
- If we insert a key/value pair into **localStorage** using an existing key, the key/value pair with the existing key is overwritten
- We are responsible for applying a key naming scheme (e.g. ”xxx_0”, ”xxx_1”,…)

HTML5 Web Storage

- You can also access **localStorage** in an associative map-style:

```
// These two are equivalent

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

HTML5 Web Storage

- It is possible to delete a specific entry in **localStorage** (for the domain in question), given the key for the entry:

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

HTML5 Web Storage

- It is also possible to clear all entries in **localStorage** (for the domain in question)

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

Communicating Web Storage Updates

- To register to receive the storage events of a window's origin, simply register an event listener

```
window.addEventListener("storage", displayStorageEvent, true);

// display the contents of a storage event
function displayStorageEvent(e) {
    var logged = "key:" + e.key + ", newValue:" + e.newValue +
    ", oldValue:" +
    e.oldValue +", url:" + e.url + ", storageArea:" +
    e.storageArea;
    alert(logged);
}
```

JSON Object Storage

- Why not just have everything stored in the application-specific array, i.e. both keys and values?

- You can do that, but…
  - …the ***JSON.stringify/parse*** operations will then have to process all of your data, whenever something changes – this is not very efficient!

JSON Object Storage

- Although the specification for HTML5 Web Storage allows for objects of any type to be stored as keyvalue pairs, in current implementations, some browsers limit values to be text string data types.
- JSON is a standard for data-interchange that can represent objects as strings and vice-versa.
- We can use it to serialize complex objects in and out of Web Storage in order to persist complex data types.

JSON Object Storage

```
<script>
var data;
function loadData() {
  data = JSON.parse(sessionStorage["myStorageKey"])
}
function saveData() {
  sessionStorage["myStorageKey"] = JSON.stringify(data);
}
window.addEventListener("load", loadData, true);
window.addEventListener("unload", saveData, true);
</script>
```

Summary

- Web storage replaces cookie technology
- API accessed through the **localStorage** property, supported by ”all” browsers

- API rather small and straightforward
  - Can store/retrieve key/value pairs
  - All data is stored as strings
  - Use JSON operations for conversion

Summary

- Web storage is domain-specific, not application-specific
- A strategy for managing keys within a domain must be defined
- Typical strategy
  - Define application-specific key prefixes
  - Keep all keys in an array, which itself is also stored in local storage, using an application-specific key

- What if the storage should be session-oriented, i.e. everything should disappear when the session is over?
  - Use **sessionStorage** instead of **localStorage**
  - The API is exactly the same…

LOCAL DATABASE

BEYOND KEY / VALUE PAIRS

Beyond Key/Value Pairs: Competing Visions

- There are many choices for database storage:
  - SQL-92
  - Oracle SQL
  - Microsoft SQL
  - MySQL
  - PostgreSQL
  - SQLite SQL

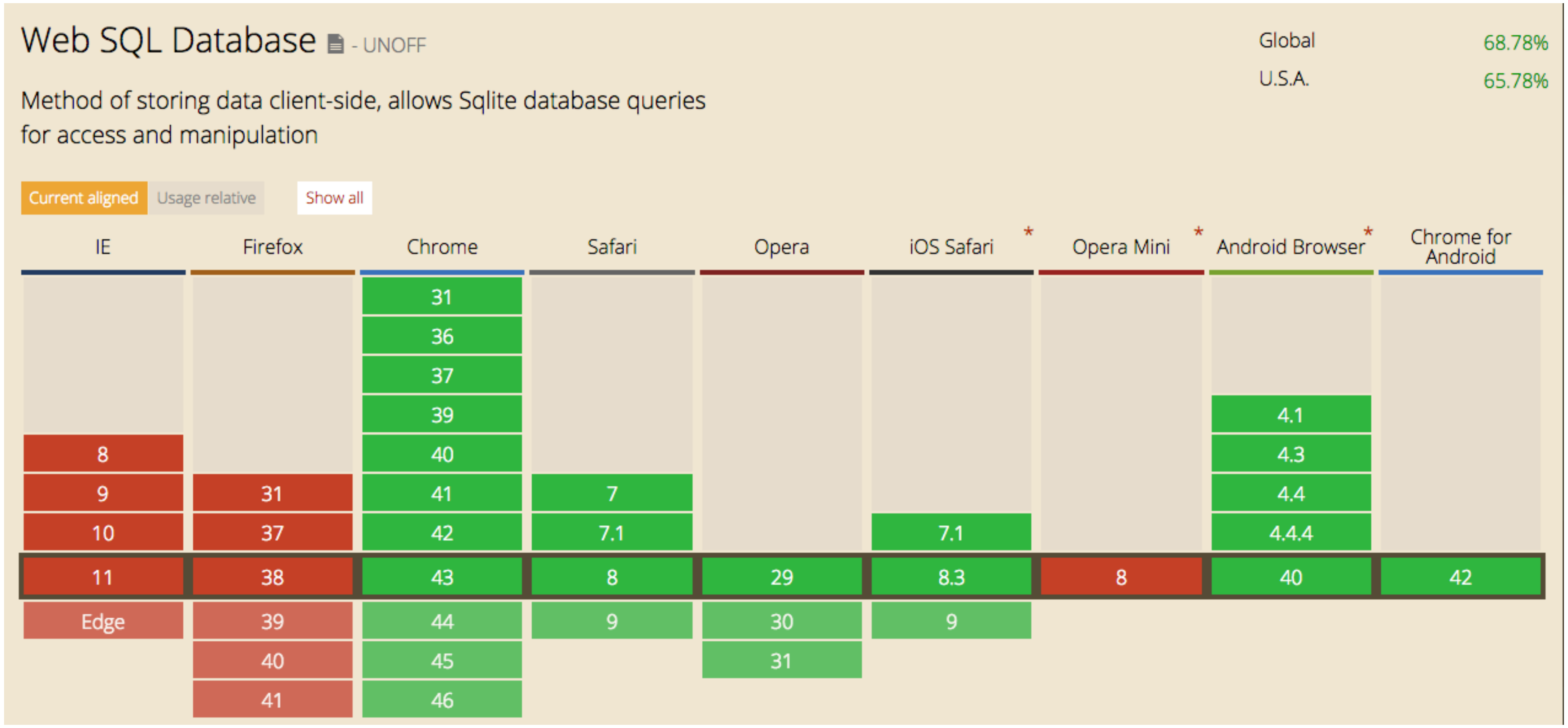
- But these are all server side! What can we do locally?
  - Web SQL Database --depreciated
  - Indexed Database API

Beyond Key/Value Pairs: Competing Visions

- WebSQL Database
  - 2007 – Google Gears (based on SQLite) -> Web SQL Database

```
openDatabase('documents', '1.0', 'Local document
storage', 5*1024*1024, function (db) {
  db.changeVersion('', '1.0', function (t) {
    t.executeSql('CREATE TABLE docids (id, name)');
  }, error);
});
```

Browser Compatibility



Web SQL Database - UNOFF

Global 68.78%
U.S.A. 65.78%

Method of storing data client-side, allows Sqlite database queries for access and manipulation

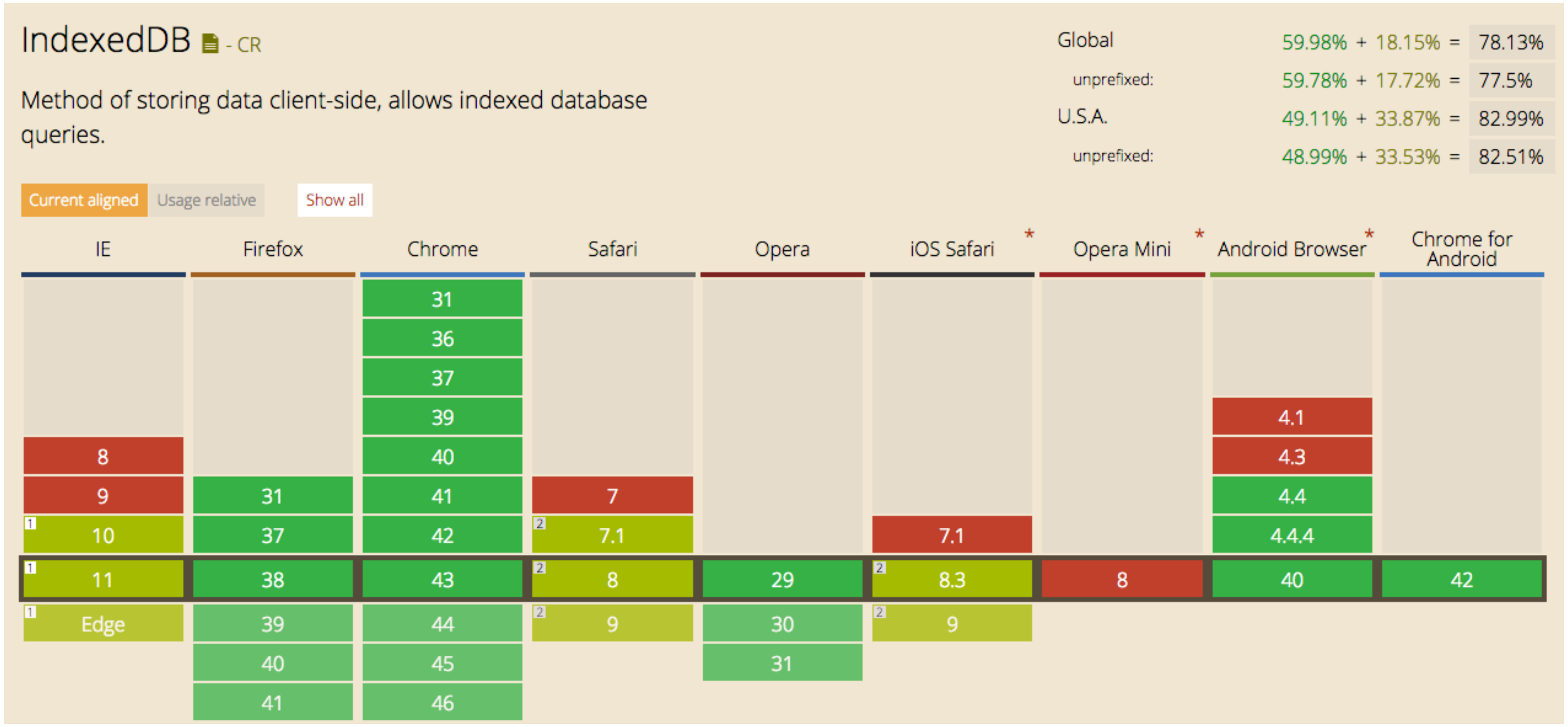
Current aligned | Usage relative | Show all

IE	Firefox	Chrome	Safari	Opera	iOS Safari *	Opera Mini *	Android Browser *	Chrome for Android
		31						
		36						
		37						
		39					4.1	
8		40					4.3	
9	31	41	7				4.4	
10	37	42	7.1		7.1		4.4.4	
11	38	43	8	29	8.3	8	40	42
Edge	39	44	9	30	9			
	40	45		31				
	41	46						

Beyond Key/Value Pairs: Competing Visions

- Indexed Database API
  - Replaces the depreciated WebSimpleDB
  - Now colloquially referred to as "indexedDB"
- **object store**
  - Shares many concepts with a SQL database:
    - Records (keys or attributes)
    - Fields (values)
    - Datatypes
- But has no query language!

Browser Compatibility



IndexedDB - CR

Method of storing data client-side, allows indexed database queries.

Global		59.98%	+ 18.15%	=	78.13%
	unprefixed:	59.78%	+ 17.72%	=	77.5%
U.S.A.		49.11%	+ 33.87%	=	82.99%
	unprefixed:	48.99%	+ 33.53%	=	82.51%

Current aligned Usage relative Show all

IE	Firefox	Chrome	Safari	Opera	iOS Safari *	Opera Mini *	Android Browser *	Chrome for Android
		31						
		36						
		37						
		39					4.1	
8		40					4.3	
9	31	41	7				4.4	
[1] 10	37	42	[2] 7.1		7.1		4.4.4	
[1] 11	38	43	[2] 8	29	[2] 8.3	8	40	42
[1] Edge	39	44	[2] 9	30	[2] 9			
	40	45		31				
	41	46						

Key concepts and usage

- IndexedDB is a transactional database system, like a SQL-based RDBMS; however whereas the latter uses tables with fixed columns, IndexedDB is a JavaScript-based object-oriented database.
- IndexedDB lets you store and retrieve objects that are indexed with a key; any objects supported by the structured clone algorithm can be stored.
- You need to specify the database schema, open a connection to your database, and then retrieve and update data within a series of transactions.
- *Note: 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.*

Synchronous and asynchronous

- Operations performed using IndexedDB are done asynchronously, so as not to block the rest of an application's running.
- IndexedDB originally included both an asynchronous API and a synchronous API
- The synchronous API being intended for use only with **Web Workers**.
- The synchronous version was removed from the spec because its need was questionable, however it may be reintroduced in the future if there is enough demand from web developers.

Storage limits and eviction criteria

- There are a number of web technologies that store data of one kind or another on the client-side (i.e. on your local disk), IndexedDB being the most commonly talked about one.
- The process by which the browser works out how much space to allocate to web data storage and what to delete when that limit is reached is not simple, and differs between browsers.

IndexedDB Interfaces

- To get access to a database, call **open()** on the indexedDB attribute of a window object.
- This method returns an IDBRequest object; asynchronous operations communicate to the calling application by firing events on IDBRequest objects.

IndexedDB Interfaces

- `IDBEnvironment`
  - Provides access to IndexedDB functionality. It is implemented by the window and worker objects.
- `IDBFactory`
  - Provides access to a database. This is the interface implemented by the global object indexedDB and is therefore the entry point for the API.
- `IDBOpenDBRequest`
  - Represents a request to open a database.
- `IDBDatabase`
  - Represents a connection to a database. It's the only way to get a transaction on the database.
- `IDBRequest`
  - Generic interface that handles database requests and provides access to results.

Retrieving and modifying data

- `IDBTransaction`
  - Represents a transaction. You create a transaction on a database, specify the scope (such as which object stores you want to access), and determine the kind of access (read only or readwrite) that you want.
- `IDBObjectStore`
  - Represents an object store that allows access to a set of data in an IndexedDB database, looked up via primary key.
- `IDBIndex`
  - Also allows access to a subset of data in an IndexedDB database, but uses an index to retrieve the record(s) rather than the primary key. This is sometimes faster than using IDBObjectStore.
- `IDBCursor`
  - Iterates over object stores and indexes.
- `IDBCursorWithValue`
  - Iterates over object stores and indexes and returns the cursor's current value.
- `IDBKeyRange`
  - Defines a range of keys that can be used to retrieve data from a database in a certain range.

Custom event interfaces

- This specification fires events with:

- `IDBVersionChangeEvent`
  - The IDBVersionChangeEvent interface indicates that the version of the database has changed, as the result of an IDBOpenDBRequest.onupgradeneeded event handler function.

Examples

- To-do Notifications (view example live)
- Storing images and files in IndexedDB

Questions?

References

- Crowther, R., & Lennon, J. (n.d.). HTML5 in action.
- Indexed Database API (n.d.). Retrieved June 12, 2015, from http://www.w3.org/TR/IndexedDB/
- Web Storage (Second Edition) (n.d.). Retrieved June 12, 2015, from http://www.w3.org/TR/webstorage/