Outline

- Web storage
- Web SQL database
- Indexed DB

Web Storage

- Key-value pairs set by JavaScript code
  - setItem (key, value), getItem (key)
  - Value is of type text
    - JSON.stringify(), JSON.parse() may be used to store other types
- Similar to cookies
- But are not sent to the server
- Two types
  - sessionStorage
  - localStorage

Web Storage

- sessionStorage
  – Data stored with the window object
  – Data discarded when window is closed
  – Useful for window-specific state data
- localStorage
  – Data available to all windows/tabs from same source
  – Data available even after the window is closed
  – Useful for
    - Application settings
    - Data that has not yet been sent to server or stored in database

Web Storage

- More on the web storage API
  – length: hold the number of key-value pairs
  – key(i): returns the i´th key
  – removeItem(key): removes the key-value pair
  – clear: removes all the key-value pairs
- Debugging tools:
  – Check the textbook or the documentation for your favourite browser

Web SQL database

- HTML5 defines a Web SQL database API
- Data stored in an SQLite database
- Standard SQL statements to
  - Create tables
  - Select rows
  - Insert rows
  - Update rows
  - Delete rows

Web SQL database operations

- Opening a database

```
db = openDatabase(name, ver, dispName, maxSize);
```

- Doing a transaction

```
db.transaction (function (t) {
   // SQL statements inside the transaction
});
```

- Execute SQL statements

```
t.executeSQL ('CREATE TABLE IF NOT EXIST tablename'
   // The table definition
);
```

Web SQL database operations

- Inserting values

```
t.executeSQL('INSERT INTO tablename(field1,field2)'
            +' VALUES (?, ?)', [var1, var2],
            successFunction, errorHandler);
```

- Querying

```
t.executeSQL ('SELECT *  FROM tablename '
            +'WHERE field1=?', [var1],
      function (t, res) {
         for (var i=0; i<res.rows.length; i++) {
            // Do something clever with
            // res.rows.item(i);
         }
      }, errorHandler);
```

Web SQL database issues

- Standardisation/implementation issues
  - Specification deprecated by W3C in 2010
  - IE and Firefox does not support Web SQL
- Technical issues
  - Developer aesthetics
  - Which SQL?
  - SQLite is the only implementation

Indexed DB

- An alternative browser database
  - Proposed by Mozilla
  - The API is currently a W3C candidate recommendation
  - Easy access to persistent object using JavaScript
  - No advanced query language
  - Uses indexes for content-based access

Indexed DB process

- Typical sequence of operations
  1. Open a database and start a transaction
  2. Create an object store
  3. Make a request to do some database operation, like adding, deleting 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

Indexed DB transaction

- Currently IndexedDBs are asynchronous
  - onerror() and onsuccess() callbacks called when requested operation is completed
- Transactions are short lived
  - Entering browser event loop will implicitly end the transaction
    - So no alert() calls inside an transaction

Objects and keys

- Objects in IndexedDB
  - An object store may hold any type of object
  - All objects are identified by the key value
  - Keys should be immutable
- Key paths in IndexedDB
  - When a JavaScript object property holds the key value
- Key generators (autoincrement) in IndexedDB
  - Creates keys automatically (in no key value is provided)

Creating an object store

```
request = indexedDB.open(databaseName);
request.onerror = function(event) {
  alert("Something failed: " + event.target.message);
};
request.onsuccess = function(event) {
  db = event.target.result;
  if (model.version != db.version) {
    verRequest = db.setVersion(version);
    verRequest.onsuccess = function(event) {
      store = db.createObjectStore(objectStoreName, {
        keyPath: "msgId",
        autoIncrement: true
      });
      store.createIndex("sender", "sender", {unique: false});
      store.createIndex("timeStamp", "timeStamp", {unique: false});
    };
  }
};
```

Adding/modifying objects

- Two alternative functions
  - add(): for adding new objects
  - put(): for modifying objects or adding new if the objects do not exist already

Adding objects to IndexedDB

```
const messages = [
   { sender: "+47112233", timeStamp: "2013-01-23T22:20+01:00",
     text: "How are you?" },
   { sender: "+47112233", timeStamp: "2013-01-23T22:21+01:00",
     text: "I'm quite fine ;-)" }
];

if ('webkitIndexedDB' in window) {
   window.IDBTransaction = window.webkitIDBTransaction;
}
var READ_WRITE = window.IDBTransaction.READ_WRITE;

transaction = db.transaction([objectStoreName], READ_WRITE);
store = transaction.objectStore(objectStoreName);
for (i in messages) {
     request = store.add(messages[i]);
}
```

Iterating a cursor

```
transaction = db.transaction([objectStoreName], READ_ONLY);
store = transaction.objectStore(objectStoreName);
request = store.openCursor();

request.onsuccess = function(event) {
   cursor = event.target.result;
   if (cursor) {
      renderMessage(cursor.value);
      cursor.continue();
   }
}
```

Object retrieval

- Two alternative functions
  - get(): for retrieving individual objects
  - openCursor(): for iterating over a set of objects
- Complex queries:
  - Indexes can be used to sort the result set
  - Cursor ranges may be defined for iterating over a subset of the result set
  - Filter functions may be called when iterating the cursor for selecting given objects

Deleting IndexedDB data

- Deleting objects in an object store
  - delete(key)
  - clear()
- Deleting an object store
  - deleteObjectStore(name), may only be called from within setVersion() success handler

IndexedDB issues

- Implementation
  - Currently only in IE 10+, Firefox, Chrome, Opera
  - Implementations differ in different browsers

  - http://caniuse.com/sql-storage
  - http://caniuse.com/indexeddb

Why bother?

- Why bother having databases inside the browser
  - Local caches in synchronised web pattern
  - Persistent stores for offline web apps – e.g., applications logs
- Why not use XML?
  - Data and version management
  - Impedance mismatch
  - Sandbox limitations

Conclusion

- Three ways to store local data in web applications
  - Web storage
  - Web SQL database
  - IndexedDB

  - External libraries can provide a common interface (simulates IndexedDB) no matter the underlying implementation (https://bitbucket.org/ytkyaw/ydn-db/wiki/Home)