Data

Marquise Slaughter

Rasmussen College

Author Note

This paper is being submitted on March 03, 2019, for Darice Corey’s CIS4655C Section 01 Advanced Mobile Web Application Development - Online Plus - 2019 Winter Quarter Term 2

Data Models

Web SQL databases have three core methods including openDatabase, transaction, and executeSql (Tutorials Point 2019). The open database can essentially create, open, and manage a database within a single line of code. However, the database schema must be defined in creation. Database creation and the opening is done by creating a declaration for the database variable and setting parameters such as the database name, description, version, size of the database, and creation callback in the event creation is needed (Tutorials Point 2019). The execution of queries to this database is demonstrated by using the “database.transaction ()” function (Tutorials Point 2019). Within the database transaction query, it can be used similarly to a MySQL database in the sense that an additional declaration is needed to complete the execution of the query or creation of a database.

The creation of queries is done relatively simple as well using SQL insert queries to create tables and add data to the table within the database (Mahemoff 2010). The addition of values can be executed with or without an additional statement, meaning data can be declared in the database creation statement (Tutorials Point 2019). When reading functions, callback functions are used (Tutorials Point 2019). Web SQL databases are more general in context and data could be roughly anything (Tutorials Point 2019). An example of creating a database would be something like this “var db = openDatabase (‘database name’, ‘version number’, ‘description of database’, ‘size of database’);” the creation of a table within the database would likely follow to organize data and would be presented as “db.function)function (data type for database) { database type.executeSql(‘CREATE TABLE IF NOT EXISTS “table name” (unique table keys such as id, or name, etc.)’);” (Tutorials Point 2019). A real-world application could be made for a DMV application that utilizes a database and table. The table name would be drivers with a unique key of ID numbers, and data would be added from a user interface supported by console applications as well as an insert operation taking place for each new citizen without an ID to insert into the table an ID number and citizen name, etc.

Indexed databases allow for dynamic database creation and data insertion using object stores (Bell & Alabbas 2018). A statement is used to request the database creation with a name declaration (Bell & Alabbas 2018). After creation, both queries and data creation can be utilized within functions (Bell & Alabbas 2018). Data additions take place through a “store.put” statement and all is executed through the use of a “db = request.result” statement stored within a function (Bell & Alabbas 2018). Unique keys can be stored by simply including the “{unique:true}” as part of the record creation statement (Bell & Alabbas 2018).

Logic can be included within the code to query the database based on the parameters, and error handling takes place using a “report (tx.error)” statement printing any errors or simply notifying the user the query was unsuccessful in finding the intended file (Bell & Alabbas 2018). Database termination takes place through the use of a simple “db.close()” statement, thereby terminating the database connection after the query is complete (Bell & Alabbas 2018). In comparison to the Web SQL database, indexed databases can be utilized by multiple clients (pages and workers) at the same time, making indexed databases useful for transactions as multiple instances of reading and writing may be taking place at any given moment (Bell & Alabbas 2018). Database upgrade must take place using an “upgraded needed” event handler, “upgrade needed” event handlers can only be utilized when all other connections to the database are closed (Bell & Alabbas 2018). An example of a data model for an indexed database is a library. The library is indexed based on the ISBN number, title, and can be indexed further for creating search parameters such as the year or author name as well (Bell & Alabbas 2018). The query would be written as “var request = index.get (“Title Name”)” while storing variables are written as “store.put ({title: “insert title”, author: “insert author”, ISBN: “ISBN number”}).

File system API is primarily for storage of large files and binary content such as images, audio, video, PDFs, etc. with a hierarchical system of files sitting on the user’s hard drive (Bell 2018). The hierarchal system is similar to the windows file directory making nearly group of files indexable within the directory such as a user’s files on a windows system (Bell 2018). The user files on a windows system may include other files in an attempt to organize such as documents, and within this directory, a list of documents and each assigned a web kit relative path. Other examples include a videos file and a list of videos within such directory with each assigned a web kit relative path, music with a list of music files within such directory with each assigned a web kit relative path, and photos with a list of photos within such directory with each assigned a web kit relative path.

The file system also has drag and drop functionality for files using a “webkitGetAsEntry” method and event listeners (Bell 2018). The File system entry interface has multiple attributes such as isFile, isDirectory, name, fullpath, filesystem, and in most cases a get parent method that queues a task and evaluates the data held within the attributes (Bell 2018). Creating an entry begins with a function and multiple console.log entries to define the name of the file, path, and rather or not the file is a file (Bell 2018). While utilizing the FileSystemDirectoryEntry interface that allows you to use getFile and getDirectory utilizes more complex data for directory entries where there are many optional fields to provide such as file path, errorcallback, successcallback for confirmation, etc. (Bell 2018).

# References

Bell, J. (2018, November 9). *File and Directory Entries API*. Retrieved from WICG: https://wicg.github.io/entries-api/

Bell, J., & Alabbas, A. (2018, January 30). *Indexed Database API 2.0*. Retrieved from w3.org: https://www.w3.org/TR/IndexedDB-2/

Mahemoff, M. (2010, October 1). *ClientSide Storage*. Retrieved from html5rocks: https://www.html5rocks.com/en/tutorials/offline/storage/

Tutorials Point. (2019). *HTML5 - Web SQL Database*. Retrieved from tutorialspoint: https://www.tutorialspoint.com/html5/html5\_web\_sql.htm