Developing Offline HTML5 Applications (Caching, LocalStorage, IndexedDB)

Overview

As long as you have all the necessary files downloaded to your computer, your browser can run an HTML5 application without the need for an internet connection. Of course, you would not be able to access any external databases while in this state, but, if your application is simple enough not to be constantly making use of an API, this is something you can work around. One solution would be to simply provide the user with the database file itself, but this file might be very large, and not every user would need the same information available to them offline - as they may only use a portion of the data.

Fortunately, HTML5 provides us with a number of technologies that allow us to store data within the user's browser itself, without having to transfer over a direct copy of the database. I shall be going over a number of these technologies in the section below.

Technologies

Caching

Caching is the act of storing frequently used data in a more easily accessible location, so that it may be retrieved more quickly. When it comes to offline applications, it has the added bonus of providing the client with the data they use the most without needing an internet connection, meaning that they will be able to use the application as they would normally the majority of the time. Before HTML5, websites would make use of cookies to store such data on the client's browser[1], but we now have the option to use actual datastores instead. HTML5 provides two long-term datastores for client-side storage, each with its own advantages and disadvantages.

(Note: Do not store any sensitive information in these datastores. They are not private and can be viewed by anyone who has access to your user's browser.)

LocalStorage[2]

LocalStorage is the most basic of the two long-term datastores. It stores key-value pairs, and supports operations for setting, getting, and removing data based on their keys. It can hold around five megabytes of data for a single web application, but this amount varies from browser to browser[3]. It has the advantage of being very simple to use, but the disadvantage of not having the proper structure of a database, and thus not being able to be queried.

IndexedDB[4]

IndexedDB is the long-term datastore that you want to use when dealing with a more complex set of data. It has its own equivalent of tables in the form of "object stores", which, as well as having keys for each of their records, can be given custom indices that consist of one or more fields to make locating and retrieving data even quicker. It also has a much greater storage limit than LocalStorage, with sizes varying from 50MB to 2GB depending on the user's browser of choice and available disk space[5].

Whether you should use IndexedDB or LocalStorage depends on the scope of your application. If your application uses a very simple dataset, then there is no need to use IndexedDB. If your application requires a very complex dataset, then LocalStorage will not suffice. One is not inherently superior to the other; they just serve different purposes.

Bibliography

1. https://www.w3schools.com/html/html5_webstorage.asp
2. https://developer.mozilla.org/en-US/docs/Web/API/Window/localStorage
3. https://stackoverflow.com/questions/2989284/what-is-the-max-size-of-localstorage-values
4. https://developer.mozilla.org/en-US/docs/Web/API/IndexedDB_API
5. https://stackoverflow.com/questions/5692820/maximum-item-size-in-indexeddb