**Mobile Assignment – IndexedDB Code Explanation**

|  |  |  |  |
| --- | --- | --- | --- |
| PURPOSE | CODE | EXPLANATION | CHANGES (in red) |
| Set Variables | //Mobile Application to hold Persontal Contacts  //Used for Bradfield Senior College Assignment  //Created by: Peter Cox - PCSquared  //Created on: Saturday 3rg August 2013    //SET VARIABLES  var localDatabase = {}; //NAMESPACE use for our database  var databaseName = "ContactDB"; //Database Name  var version = **2**; // database version  var db = null; // This will hold the instance of our databse when it is generated  //OBJECT STORE  var ObjectStoreName = "MyContacts";  var ObjectStoreKeyPathName = "id"; | var localDatabase: We set a namespace we can use to hold the indexedDB Object. | var databaseName = "ContactDB";    var version = **2**; |
| Create a set of record that will be added when you first open your mobile app | //initial data to populate  const contactsData = [  { Fname: "Santa", Lname: "Clause", nickname: "santa", tel\_H: "1234567", tel\_M: "040123456",  dob: "1/01/1950",  email: "santa@northpole.com" , group: "VIP"},  { Fname: "Easter", Lname: "Bunny", nickname: "bunny", tel\_H: "654321", tel\_M: "040654321",  dob: "2/02/1979",  email: "bunny@easterisland.org" , group: "VIP"}  ];    //http://www.onlywebpro.com/2012/12/23/html5-storage-indexeddb/ | Create an Array and put some records in it.  We will use this later to fill up your database table when you first create it. | Change both records   |  |  | | --- | --- | | **Record** | **Value** | | 'Fname', | Santa | | 'Lname', | Clause | | 'nickname' | , santa | | 'tel\_H', | 1234567 | | 'tel\_M', | 1234567 | | 'email', | santa@northpole.com | | 'dob', | 1/01/1950 | | 'group', | VIP | |
| Testing if your browser has IndexedDB | //---------------- TESTING FOR indexedDB----------------------------------------------------------------//  //The root object for IndexedDB API is called indexedDB.  //BUT current implementations of IndexedDB hide under different browser prefixes- so we test them here  localDatabase.indexedDB = window.indexedDB || window.mozIndexedDB || window.webkitIndexedDB || window.msIndexedDB;  //Also references to some window.IDB\* objects:IDBKeyRange and IDBTransaction should be tested as well  localDatabase.IDBKeyRange = window.IDBKeyRange || window.webkitIDBKeyRange;  localDatabase.IDBTransaction = window.IDBTransaction || window.webkitIDBTransaction;    console.log('opening local database');    //Here CHECK for the presence of this IndexedDB OBJECT to see whether the current browser supports IndexedDB or not.  if (!localDatabase.indexedDB) {  window.alert("Your browser doesn't support a stable version of IndexedDB.")  } | This tests to see if your Browser has a stable version of indexedDB. |  |
| We request to OPEN a Database | //---------------- OPEN DATABASE-------------------------------------------------------------------------//  // Here we make an asynchronous API (REQUEST) to Open our Database Using the OPEN method  // Paremeter: 1. Database Name 2. Database Version number  var **request** = localDatabase.**indexedDB**.**open**(databaseName,version);    //Our CALL will RETURN a reference to a "REQUEST" object  //which exposes two HANDLER events: ON-SUCCESS and ONERROR.    // ONERROR  **request.onerror** = function(event) {  console.log("Request Error: newBD: " + event.target.errorCode);  };    // ONSUCCESS  **request.onsuccess** = function(event) {  //REQUEST here was generated with a call to indexedDB.open(),  // so request.result is an INSTANCE of IDBDatabase,    **db = request.result;** *// set db as instance of IDBDatabase*    console.log("Opens DB - success: "+ " ---- " + db + "---" + db.version);  };    // Check if opened in other TAB  request.onblocked = function(event) {  // If some other tab is loaded with the database, then it needs to be closed  // before we can proceed.  alert("Please close all other tabs with this site open!");  }; | Here we ASK to open a Database using the indexedDB Object METHOD open.  This open method takes two Parameters ( database name and its version number)  NOTE:  When you ask (OR REQUEST) to do something what is RETURNED is if you were Successfully Or an Error occurred.  **request.onerror**   * *message*   **request.onsuccess**   * *Create an INSTANCE of the DATABASE. You will use this instance later on when you code the different functions.* |  |
| If the Database you requested to open is NOT present then the function “onupgradeneeded” is automatically triggered.  This is where you create the TABLES for your database | //When the database did not previously exist, an onupgradeneeded event is triggered.  //The onupgradeneeded callback is the only place in our code that we can alter the structure of the database.  //In the handler for this event, you should create the object stores (tables) and indexes needed for this version of the //database:    //To update the database, or to create, delete or modify the database,  //then you have to implement the onupgradeneeded handler or which will be called as  //part of a versionchange transaction that allows you to make any changes on the database.    request.onupgradeneeded = function(event) {  //event.request.result is an INSTANCE of IDBDatabase  //set db as instnce of IDBDatabase  var db = event.target.result;  //checks if there is an older version of the datbase  if(event.oldVersion < 10){  console.log("OLD VERSION " + event.oldVersion);  }    console.log(" onupgradeneeded " + db);  //---------CRETAE AN OBJECT STORE -----------------------------//  //To create an OBJECT STORE we use the createObjectStore() METHOD.  //This method accepts 2 parameters:  // - name of the store : I have named this Object Store "Contacts"  // - parameter object : a key path (optionally have a key generator)  // \* keyPath that is the property that makes an individual object in the store unique  //Explanation  //KEY: A data value by which stored values are organized and retrieved in the object store.  //KEY: Each record in the object store is uniquely identified by a “key.”  //Here, I have use the “id” as keyPath, which is unique value in the object store,  //We must make sure that the “id” property must be present in every objects in the object store.    var objectStore = db.createObjectStore(ObjectStoreName, {keyPath: ObjectStoreKeyPathName, autoIncrement: true });  //What is an Object Store?  //The mechanism by which data is stored in the database.  //The object store persistently holds records, which are key-value pairs.  //Records within an object store are sorted according to the keys in an ascending order.  objectStore.createIndex('Fname', 'Fname', { unique: false });  objectStore.createIndex('Lname', 'Lname', { unique: false });  objectStore.createIndex('nickname', 'nickname', { unique: false });  objectStore.createIndex('tel\_H', 'tel\_H', { unique: false });  objectStore.createIndex('tel\_M', 'tel\_M', { unique: false });  objectStore.createIndex('email', 'email', { unique: false });  objectStore.createIndex('dob', 'dob', { unique: false });  objectStore.createIndex('group', 'group', { unique: false });    //populate db  for (var i in contactsData) {  console.log("adding");  objectStore.add(contactsData[i]);  }  } |  | 'Fname', 'Fname',  'Lname', 'Lname',  'nickname', 'nickname',  'tel\_H', 'tel\_H',  'tel\_M', 'tel\_M',  'email', 'email',  'dob', 'dob',  'group', 'group', |

**MOBILE FUNCTIONS**

SEARCH (BY ID)

|  |  |  |  |
| --- | --- | --- | --- |
| PURPOSE | CODE | EXPLANATION | CHANGES (in red) |
| Get record id  REQUEST a Transaction to the database ObjectStore( table) | function searchC() {  //GET SEARCH ID  // get the record ID from the user search page form  var read\_id = Number(document.getElementById("txtReadID").value);  //get transaction to Contact Store  var transaction = db.transaction(ObjectStoreName,"readonly");  var objectStore = transaction.objectStore(ObjectStoreName);  var request = objectStore.get(read\_id); | This is a search done by looking up the primary key called id. Note the variable “db” is an instance of your database.   1. get the id entered 2. Need to open a transaction to the Database: This takes two parameters ( table name AND action on the table eg readonly) 3. Then you open a transaction to the Object Store ( table) 4. Then use the returned object method “GET” to | This is the” form ID” from the user search page form  txtReadID |
| Requested transaction –returned error | //ONERROR  request.onerror = function(event) {  alert("Unable to retrieve daa from database!");  }; |  |  |
| Requested transaction –returned success | //ONSUCCESS  request.onsuccess = function(event) {  document.getElementById("output").innerHTML = "request.onsuccess: " + request.result;  // Do something with the request.result!  if(request.result) {  document.getElementById("output").innerHTML ="ID: " + request.result.id +                                        " Name: " + request.result.Fname + " DOB: " + request.result.dob +                                       " Email: " + request.result.email;  } else {  document.getElementById("output").innerHTML ="ID couldn't be  found   in   your    database!";  }  }; |  |  |
|  | } //end search fn |  |  |

SEARCH (BY NICKNAME)

|  |  |  |  |
| --- | --- | --- | --- |
| PURPOSE | CODE | EXPLANATION | CHANGES (in red) |
| Get record id  REQUEST a Transaction to the database ObjectStore( table) | //get search value  var read\_id = document.getElementById("txtReadID").value;  // Create transaction  var transaction = db.transaction(ObjectStoreName,"readonly");  var objStore = transaction.objectStore(ObjectStoreName);    // get index for field  var index = objStore.index("nickname");  //ONSUCCESS  //Use index gets method to read field VLAUE  index.get(read\_id).onsuccess = function(event) {  document.getElementById("output").innerHTML = ("FOUND: Nickname " + event.target.result.nickname + " Name " + event.target.result.Fname + " " + event.target.result.Lname );  };  //ONERROR  index.get(read\_id).onerror = function(event) {  document.getElementById("output").innerHTML = ("NOT FOUND in Database" );  };  //NOTE: If there is mre than one nickname the same you always get the one with the lowest key value.  //If you want to get ALL nicknames the same USE either:  //- index.openCursor()  //- index.openKeyCursor | This is a search done by looking up the primary key called id. Note the variable “db” is an instance of your database. **Create transaction**   1. get the id entered 2. Need to open a transaction to the Database: This takes two parameters ( table name AND action on the table eg readonly) 3. Then you open a transaction to the Object Store ( table) 4. Then use the returned object method “GET” to | This is the” form ID” from the user search page form  txtReadID |
| Requested transaction –returned error |  |  |  |
| Requested transaction –returned success |  |  |  |
|  |  |  |  |

DISPLAY

|  |  |  |  |
| --- | --- | --- | --- |
| PURPOSE | CODE | EXPLANATION | CHANGES (in red) |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

EDIT

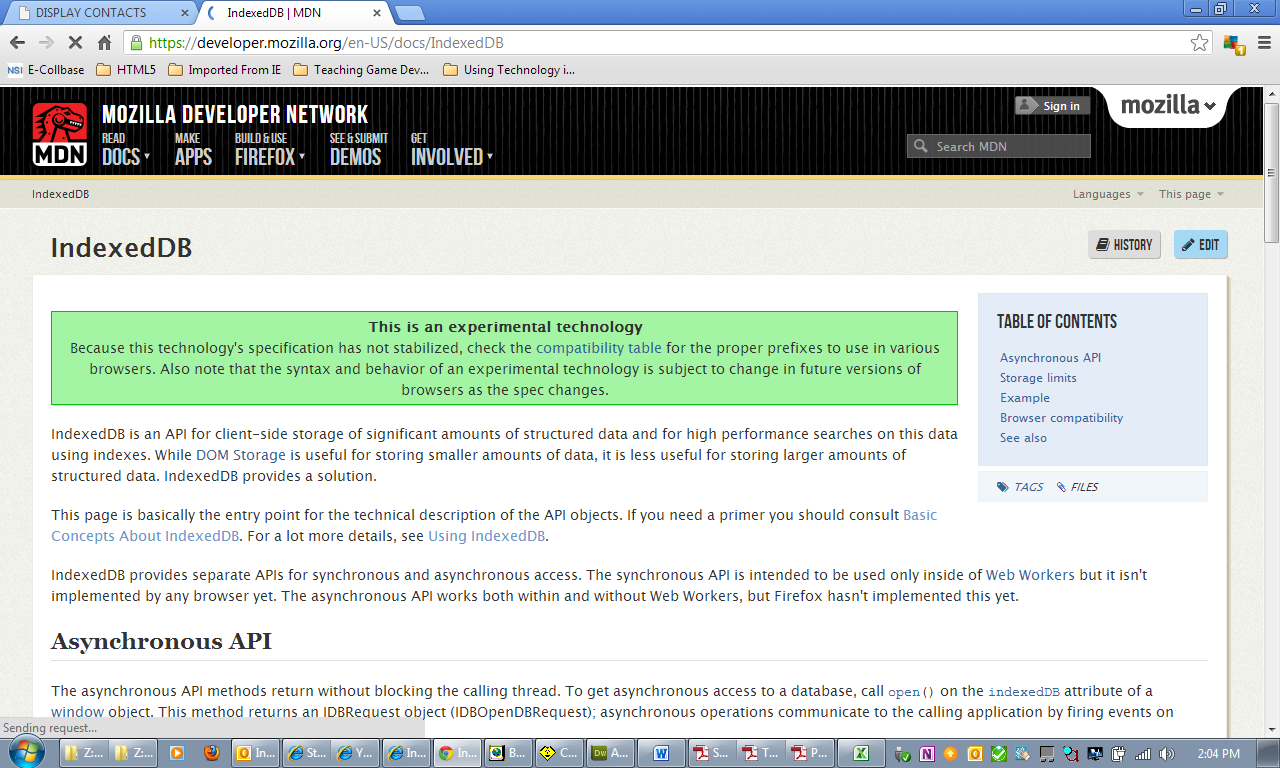
|  |  |  |  |
| --- | --- | --- | --- |
| PURPOSE | CODE | EXPLANATION | CHANGES (in red) |
|  |  |  |  |
|  |  |  |  |

ADD

|  |  |  |  |
| --- | --- | --- | --- |
| PURPOSE | CODE | EXPLANATION | CHANGES (in red) |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

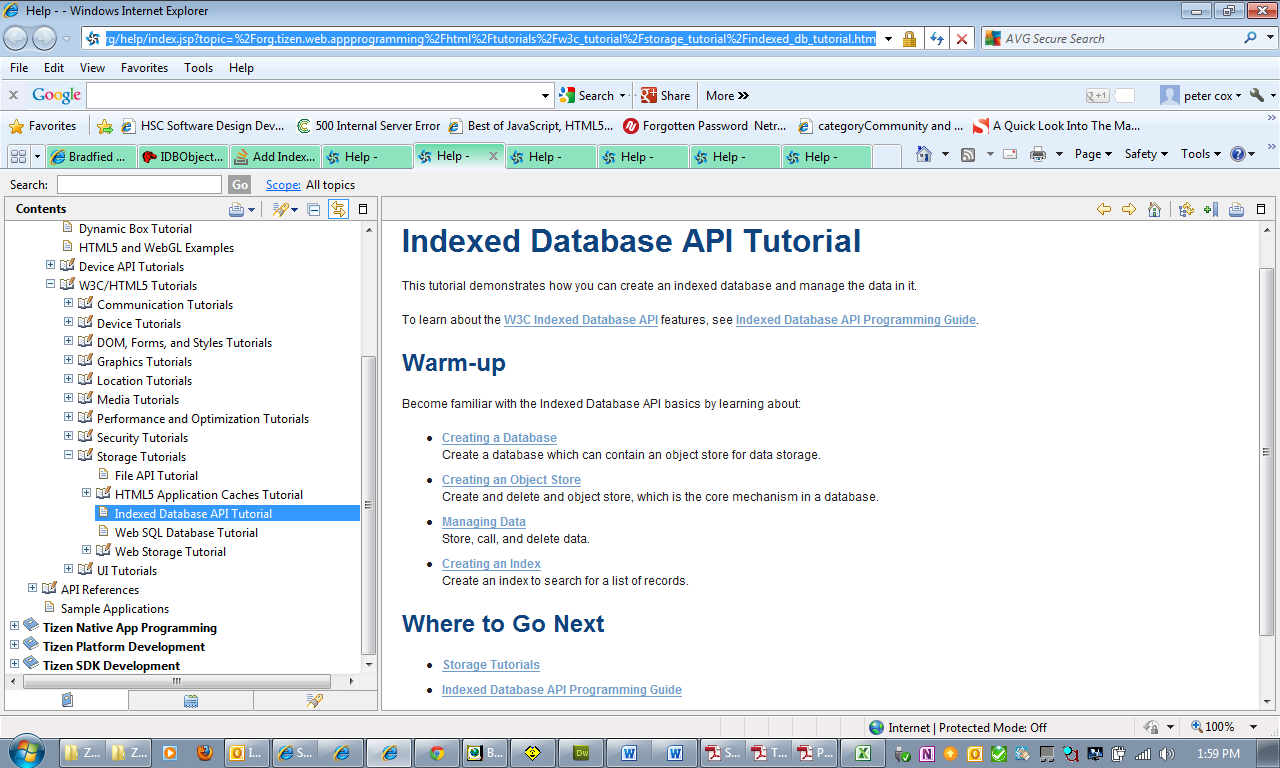
**Tutorials**

**Mozilla Developer Network**



[**https://developer.mozilla.org/en-US/docs/IndexedDB/Using\_IndexedDB**](https://developer.mozilla.org/en-US/docs/IndexedDB/Using_IndexedDB)

**Tizen Developers**



<https://developer.tizen.org/help/index.jsp?topic=%2Forg.tizen.web.appprogramming%2Fhtml%2Ftutorials%2Fw3c_tutorial%2Fstorage_tutorial%2Findexed_db_tutorial.htm>

**Examples**

* ToDo List  
  <http://blog.teamtreehouse.com/create-your-own-to-do-app-with-html5-and-indexeddb>
* <https://developer.mozilla.org/en-US/demos/tag/tech:indexeddb>