PhoneGap & jQuery Mobile

Lesson 7

Thomas Mak

makzan@42games.net

Source Codes

https://github.com/makzan/PhoneGap-Course-Examples

Today

- Using Local Database
- Using Camera

Local Database

- WebSQL
- LocalStorage
- IndexedDB

Local Database

- WebSQL
  - It is a SQLLite implementation on browser.
  - Full SQLLite based SQL query.

    CREATE TABLE, SELECT, INSERT

Local Database

- LocalStorage
  - Simple key-value based storage.
  - simple access with localStorage object

    localStorage[“foo”] = “bar”;

    localStorage[“foo”]; // “bar”

Local Database

- IndexedDB
  - Database with key-value store
  - Store object

Local Database

```
openDatabase("diary", "1.0", "Diary DB", 1024*1024*10);
```

Code that opens a database.

Local Database

```
db.transaction(function(tx){
  tx.executeSql('SELECT * FROM something');
});
```

Code that execute SQL query.

Local Database

```
this.db.transaction(function(tx){
  tx.executeSql('INSERT INTO entries(name, content) VALUES(?, ?)',
          ['Thomas', 'Hello World']);
});
```

Code that insert new entry.

Local Database

```
// DB Module
(function(){
  var DB = (function(){
    function DB() {
      // init the database
    }

    DB.prototype.insert = function(entry)
    {
      // insert the entry into database
    }

    DB.prototype.query = function(callback)
    {
      // query the data from database
    }

    return DB;
  })();

  // export it in global scope.
  if (!this.diaryapp) this.diaryapp = {}
  this.diaryapp.DB = DB;

}).call(this);
```

Our DB class skeleton.

Local Database

```
    function DB() {
      this.db = openDatabase("diary", "1.0", "Diary DB", 1024*1024*10);
      // create the table
      this.db.transaction(function(tx){
        tx.executeSql('CREATE TABLE IF NOT EXISTS entries (id unique, lat, lng, note text,
video_url, audio_url, photo_url)');
      });

      // manage the entry ID ourselves
      if (localStorage["entries_id"] == undefined)
        localStorage["entries_id"] = 1;
    }
```

DB constructor. Create table and init the entry ID.

Local Database

```
function DB() {
  this.db = openDatabase("diary", "1.0", "Diary DB", 1024*1024*10);
  // create the table
  this.db.transaction(function(tx){
    tx.executeSql('CREATE TABLE IF NOT EXISTS entries (id unique, lat, lng, note text,
video_url, audio_url, photo_url)');
  });

  // manage the entry ID ourselves
  if (localStorage["entries_id"] == undefined)
    localStorage["entries_id"] = 1;
}
```

the table is named **entries** with fields: *id*, *lat*, *lng*, *note*, *video_url*, *audio_url* and *photo_url*.

Local Database

```
    function DB() {
      this.db = openDatabase("diary", "1.0", "Diary DB", 1024*1024*10);
      // create the table
      this.db.transaction(function(tx){
        tx.executeSql('CREATE TABLE IF NOT EXISTS entries (id unique, lat, lng, note text,
video_url, audio_url, photo_url)');
      });

      // manage the entry ID ourselves
      if (localStorage["entries_id"] == undefined)
        localStorage["entries_id"] = 1;
    }
```

if we manage the entry ID ourselves, it is good place to init it here with localStorage.

Local Database

```
    DB.prototype.insert = function(entry)
    {
      this.db.transaction(function(tx){
        tx.executeSql('INSERT INTO entries(id, lat,lng,note,video_url,audio_url,photo_url)
VALUES(?, ?, ?, ?, ?, ?, ?)',
          [localStorage["entries_id"], entry.lat, entry.lng, entry.note, entry.video_url,
entry.audio_url, entry.photo_url]);
          localStorage["entries_id"]++;
      });

    }
```

The insert method simply inserts data into database and increases the entries_id in localStorage.

Local Database

```
DB.prototype.query = function(callback)
{
  this.db.transaction(function(tx){
    tx.executeSql('SELECT * FROM entries', [], function(tx, results){
      if (callback) callback(results);
    });
  });
}
```

the query function is a wrapper of the SELECT SQL query. Note that we need a callback on this.

Diary Module

```
// Diary Module
(function(){
  var Diary = (function(){
    function Diary() {}
    Diary.prototype.listEntries = function(element)
    {
      window.diaryapp.db.query(function(results){
        $(element).empty();
        for (var i = results.rows.length - 1; i >= 0; i--) {
          var entry = results.rows.item(i);
          $(element).append("<li>" + entry.id + ' ' + entry.note + "</li>");
        };
        $(element).listview('refresh');
      });
    }

    return Diary;
  })();

  this.diaryapp.Diary = Diary; // export it in global scope.

}).call(this);
```

we use the DB objects when listing entries.

Diary Module

```
PhoneGapManager.prototype.prepareDatabase = function() {
  if (!window.diaryapp) window.diaryapp = {}
  window.diaryapp.db = new window.diaryapp.DB();
}
```

remember we have PhoneGapManager class?
we will init the DB object and expose it to global scope after the PhoneGap is ready.

Diary Module

```
$(document).bind("deviceready", (function(){
  this.isReady = true;
  console.log ('phonegap is ready');
  this.locateMyself();
  this.prepareDatabase();
  // list the entries once the database ready.
  (new Diary()).listEntries($('#entries-list'));
}).bind(this));
```

previously we have deviceready listener inside PhoneGapManager class. That's a good place to invoke the prepareDatabase method.

Diary Module

```
$(document).bind("deviceready", (function(){
  this.isReady = true;
  console.log ('phonegap is ready');
  this.locateMyself();
  this.prepareDatabase();
  // list the entries once the database ready.
  (new Diary()).listEntries($('#entries-list'));
}).bind(this));
```

then after we init the database, we refresh the diary entries list.

Diary Module

```
$("#create-button").click(function(){
  alert("creating dummy diary entry.");
  window.diaryapp.db.insert({
    'lat': '23.123123',
    'lng': '44.556677',
    'note': 'Long Text Here',
  })
  (new Diary()).listEntries($('#entries-list'));
});
```

We will leave the UI things for you. But we need some dummy data to test our database. So we create a button to insert dummy data into the database with the API we just defined.

Diary Module

```
<h4>Actions</h4>
<ul data-role='listview'>
  <li>
    <a href="#" id='create-button'>Create Entry</a>
  </li>
</ul>


<h4>Diary Entries</h4>
<ul data-role='listview' id='entries-list'></ul>
```

let's back to our HTML with an <ul> list and an a link to create dummy data.

Capture Photo

Capture Photo

```
navigator.camera.getPicture(onPhotoSuccess, onPhotoFail, {
  quality: 20,
  destinationType: navigator.camera.DestinationType.DATA_URL,
  saveToPhotoAlbum: true,
  targetWidth: 1024,
  targetHeight: 1024
});
```

The official way to use camera in PhoneGap

Capture Photo

```
navigator.camera.getPicture(onPhotoSuccess, onPhotoFail, {
  quality: 20,
  destinationType: navigator.camera.DestinationType.DATA_URL,
  saveToPhotoAlbum: true,
  targetWidth: 1024,
  targetHeight: 1024
});
```

photo quality, from 1- 100.

Capture Photo

```
navigator.camera.getPicture(onPhotoSuccess, onPhotoFail, {
  quality: 20,
  destinationType: navigator.camera.DestinationType.DATA_URL,
  saveToPhotoAlbum: true,
  targetWidth: 1024,
  targetHeight: 1024
});
```

either
DATA_URL (base64 encoded)
or FILE_URI (file path)

Capture Photo

```
navigator.camera.getPicture(onPhotoSuccess, onPhotoFail, {
  quality: 20,
  destinationType: navigator.camera.DestinationType.DATA_URL,
  saveToPhotoAlbum: true,
  targetWidth: 1024,
  targetHeight: 1024
});
```

indicate if a copy of the photo is saved to album.

Capture Photo

```
navigator.camera.getPicture(onPhotoSuccess, onPhotoFail, {
  quality: 20,
  destinationType: navigator.camera.DestinationType.DATA_URL,
  saveToPhotoAlbum: true,
  targetWidth: 1024,
  targetHeight: 1024
});
```

the target dimension with aspect ratio unchanged.

Capture Photo

```
// Other options
sourceType : Camera.PictureSourceType.CAMERA,
allowEdit : true,
encodingType: Camera.EncodingType.JPEG,
```

either from *CAMERA*, *PHOTOLIBRARY* or *SAVEDPHOTOALBUM*

Capture Photo

```
// Other options
sourceType : Camera.PictureSourceType.CAMERA,
allowEdit : true,
encodingType: Camera.EncodingType.JPEG,
```

in iPhone, this allow users to crop or rotate the photo before passing it back to the app.

Capture Photo

```
// Other options
sourceType : Camera.PictureSourceType.CAMERA,
allowEdit : true,
encodingType: Camera.EncodingType.JPEG,
```

either JPEG or PNG.

Capture Photo

```
// Camera Module
(function(){
  var Camera = (function(){
    function Camera() {

    }

    Camera.prototype.getPicture = function(callback)
    {
      // logic to get Picture
    }

    return Camera;
  })();

  // export it in global scope.
  if (!this.diaryapp) this.diaryapp = {}
  this.diaryapp.Camera = Camera;

}).call(this);
```

so here is the Camera class skeleton.

Capture Photo

```
Camera.prototype.getPicture = function(callback)
{
  var onPhotoSuccess = function(imageData)
  {
    var imageSrc = "" + imageData
    if (callback) callback(imageSrc);
  }

  var onPhotoFail = function(message)
  {
    alert('Camera Failed: ' + message);
  }

  navigator.camera.getPicture(onPhotoSuccess, onPhotoFail, {
    quality: 20,
    destinationType: navigator.camera.DestinationType.DATA_URL,
    saveToPhotoAlbum: true,
    targetWidth: 1024,
    targetHeight: 1024
  });
}
```

the getPicture implementation with callback methods put together.

Capture Photo

```
<a data-role='button' id='capture-photo'>Capture photo</a>

<img id='photo'>
```

image and the button defines in HTML.

Capture Photo Orientation

- One issue is that the orientation information may be lost when taking picture. That means we do not know if we are taking the photo in landscape or portrait mode.

Capture Photo Orientation

- In Android, it assumes the photo is taken in landscape mode if no orientation provided.
- So we need to handle it ourselves.
- We can use accelerometer to know the orientation.
- We can use <canvas> tag to do that.

Capture Photo Orientation

```
PhoneGapManager.prototype.watchAcceleration = function() {
  this.watchID = navigator.accelerometer.watchAcceleration(
  //success callback:
  function(acceleration){
    this.acceleration = acceleration;
  },
  // error callback:
  function(){
    // error on getting acceleration.
  },
  // options:
  { frequency: 3000 });
}
```

it's easy to watch the accelerometer.
let's add this method in PhoneGapManager class that wraps the PhoneGap accelerometer method.

Capture Photo Orientation

```
PhoneGapManager.prototype.watchAcceleration = function() {
  this.watchID = navigator.accelerometer.watchAcceleration(
  //success callback:
  function(acceleration){
    this.acceleration = acceleration;
  },
  // error callback:
  function(){
    // error on getting acceleration.
  },
  // options:
  { frequency: 3000 });
}
```

an important thing is to tune the frequency.

Capture Photo Orientation

```
$(document).bind("deviceready", (function(){
  this.isReady = true;
  console.log ('phonegap is ready');
  this.locateMyself();
  this.watchAcceleration();
  this.prepareDatabase();
  // list the entries once the database ready.
  (new Diary()).listEntries($('#entries-list'));
}).bind(this));
```

and we need to invoke the method when deviceready.

Capture Photo Orientation

- warning: a long block of coding coming.

```
var fixPhotoOrientation = function(imageSrc, successCallback)
{

  var accelerationX = window.diaryapp.phonegapManager.acceleration.x;
  var accelerationY = window.diaryapp.phonegapManager.acceleration.y;
  var orientation = (acceleration.x > acceleration.y) ? "landscape" : "portrait";

  var imgElm = $('<img/>');

  // hide this temp img element.
  imgElm.css({position: 'absolute', left: '0px', top: '-999999em', maxWidth: 'none',
width: 'auto', height: 'auto'});

  // bind the error in case errors occur.
  imgElm.bind('error', function() {
    alert('Failed on loading image.');
  });

  // the image is not ready at once, need to handle it after 'load' event
  imgElm.bind('load', function() {
    var canvas = document.createElement("canvas");
    var context = canvas.getContext('2d');
    var imageWidth = imgElm.width();
    var imageHeight = imgElm.height();

    var canvasWidth = imageWidth;
    var canvasHeight = imageHeight;
    var canvasX = 0, canvasY = 0, degree = 0;

    // Continue to next page.

    if (orientation=='portrait') {
```

```
      var canvasWidth = imageWidth;
      var canvasHeight = imageHeight;
      var canvasX = 0, canvasY = 0, degree = 0;

      // Continue from last page.

      if (orientation=='portrait') {
        // swap the canvas width and height;
        canvasWidth = imageHeight;
        canvasHeight = imageWidth;
        canvasY = imageHeight * (-1);
        // set to rotate 90 degrees.
        degree = 90;
      }
      $(canvas).attr('width', canvasWidth);
      $(canvas).attr('height', canvasHeight);

      // rotate after setting canvas width and height
      context.rotate(degree * Math.PI / 180);

      // draw after set dimension and rotation.
      context.drawImage(imgElm[0], canvasX, canvasY);

      // call back the result
      if (successCallback) successCallback(canvas.toDataURL());
    }); // end of 'load' binding

    // src and append must appear later than 'load' listener.
    imgElm.attr('src', imageSrc);
    $('body').append(imgElm);
  }
```

What this logic do is place the image into a canvas and then rotate it if it is taken in portrait.
Then it returns the image data back.

Capture Photo Orientation

```
var onPhotoSuccess = function(imageData)
{
  var imageSrc = "" + imageData
  fixPhotoOrientation(imageSrc, function(newImageSrc) {
    if (callback) callback(newImageSrc);
  });

}
```

now, on the success camera callback, we put the image data into the fixOrientation function and use the rotated image as result.

Capture Photo Orientation

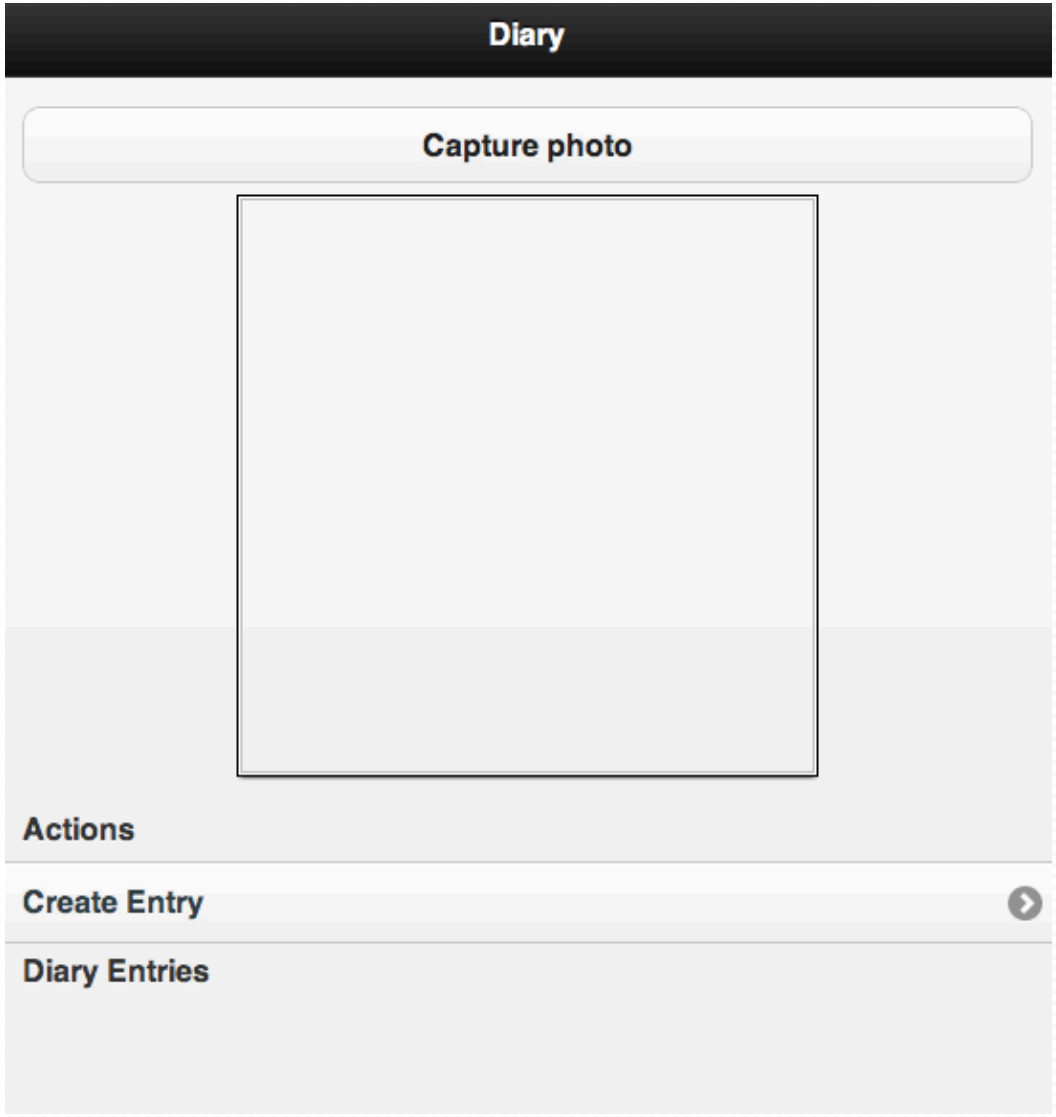
```
#photo {
  display: block;
  margin: 0 auto;
  width: 300px;
  height: 300px;

  background-image: url(http://placehold.it/700x1000.png&text=waiting...);
  background-position: center center;
  background-size: 150%;

  box-shadow: 0 0 0 1px white, 0 0 0 2px black, 0 2px 3px black;
}
```

and some style tweaking on the photo.

What we get now.



Exercise

- How you use the things learnt to create a diary app?