El API de Cordova. Parte I

1) El acelerómetro
2) La brújula
3) Terminal
4) Geolocalización
5) Almacenamiento
6) Contactos

Germán Caballero Rodríguez

El acelerómetro

Nos devuelve el movimiento del dispositivo sobre los ejes XYZ.

Cómo muchas de las funcionalidades de las que dispone el dispositivo y podemos usar con PhoneGap, se nos permite activar/desactivar por necesidades de nuestro desarrollo así como su parametrización.

El acelerómetro

cordova-plugin-device-motion

```
document.addEventListener("deviceready", onDeviceReady, false);
function onDeviceReady() {
    navigator.accelerometer.getCurrentAcceleration(onSuccess, onError);
}
navigator.accelerometer.getCurrentAcceleration (accelerometerSuccess, accelerometerError);
function onSuccess(acceleration) {
    alert('Acceleration X: ' + acceleration.x + '\n' +
          'Acceleration Y: ' + acceleration.y + '\n' +
          'Acceleration Z: ' + acceleration.z + '\n' +
          'Timestamp: '      + acceleration.timestamp + '\n');
}
function onError() {
    alert('onError!');
}
```

La brújula (Compás/COMPASS)

Nos da la dirección a la que está apuntando el dispositivo.

Cómo muchas de las funcionalidades de las que dispone el dispositivo y podemos usar con PhoneGap, se nos permite activar/desactivar por necesidades de nuestro desarrollo así como su parametrización.

La brújula (Compás/COMPASS)

cordova-plugin-device-orientation

La brújula es un sensor que detecta la dirección a la que el dispositivo apunta, por lo general de la parte superior del dispositivo. Mide en grados de 0 a la 359.99, donde 0 es el norte.

Para obtener el rumbo del compás actual. La dirección de la brújula se devuelve a través de un objeto **CompassHeading** usando la función de devolución de llamada **compassSuccess**.

La brújula (Compás/COMPASS)

cordova-plugin-device-orientation

```
document.addEventListener("deviceready", onDeviceReady, false);
function onDeviceReady() {
    navigator.compass.getCurrentHeading (compassSuccess,     compassError);
}
function onSuccess(heading) {
    alert('Heading: ' + heading.magneticHeading);
};

function onError(error) {
    alert('CompassError: ' + error.code);
};
```

La brújula (Compás/COMPASS)

navigator.compass.watchHeading

Obtiene el rumbo actual del dispositivo en un intervalo regular. Cada vez que se recupera, se ejecuta la función de devolución de llamada headingSuccess.

El ID de reloj regresado hace referencia al intervalo del reloj de la brújula.

La brújula (Compás/COMPASS)

```
function onSuccess(heading) {
    var element = document.getElementById('heading');
    element.innerHTML = 'Heading: ' + heading.magneticHeading;
};
function onError(compassError) {
    alert('Compass error: ' + compassError.code);
};
var options = {
    frequency: 3000
}; // Update every 3 seconds
var watchID = navigator.compass.watchHeading(onSuccess, onError,
options);
```

Terminal

cordova-plugin-console

Este plugin está destinado a garantizar que console.log () es tan útil como lo puede ser. Se añade función adicional para iOS, Ubuntu, Windows Phone 8 y Windows.

Si está satisfecho con la forma en que **console.log ()** funciona para usted, entonces es probable que no necesite este plugin.

Este plugin define un objeto global **console**. Sin embargo el objeto está en el ámbito global, las características proporcionadas por este plugin no están disponibles hasta después del evento deviceready.

Terminal

Métodos soportados

console.log
console.error
console.exception
console.warn
console.info
console.debug
console.assert
console.dir
console.dirxml
console.time
console.timeEnd
console.table

Geolocalización

cordova-plugin-geolocation

Este plugin proporciona información sobre la ubicación del dispositivo, tales como latitud y longitud. El objeto nos da acceso al sensor GPS.

Cómo muchas de las funcionalidades de las que dispone el dispositivo y podemos usar con PhoneGap, se nos permite activar/desactivar por necesidades de nuestro desarrollo así como su parametrización.

Este plugin define un objeto global

navigator.geolocation

Geolocalización

Métodos

navigator.geolocation.getCurrentPosition

navigator.geolocation.watchPosition

navigator.geolocation.clearWatch

Geolocalización

navigator.geolocation.getCurrentPosition

navigator.geolocation.getCurrentPosition (

geolocationSuccess,

[geolocationError],
[geolocationOptions]);

Geolocalización

```
// onSuccess Callback
// This method accepts a Position object, which contains the
// current GPS coordinates
var onSuccess = function(position) {
    alert('Latitude: '          + position.coords.latitude          + '\n' +
          'Longitude: '         + position.coords.longitude         + '\n' +
          'Altitude: '          + position.coords.altitude          + '\n' +
          'Exactitud: '         + position.coords.accuracy          + '\n' +
          'Altitude Accuracy: ' + position.coords.altitudeAccuracy  + '\n' +
          'Título: '            + position.coords.heading           + '\n' +
          'Speed: '             + position.coords.speed             + '\n' +
          'Timestamp: '         + position.timestamp                + '\n');
};
// onError Callback receives a PositionError object
function onError(error) {
    alert('code: '    + error.code    + '\n' +
          'message: ' + error.message + '\n');
}
navigator.geolocation.getCurrentPosition(onSuccess, onError);
```

Almacenamiento (Web Storage)

La interfaz Storage:

```
interface Storage {
  readonly attribute unsigned long length;
  DOMString? key(unsigned long index);
  getter DOMString? getItem(DOMString key);
  setter void setItem(DOMString key, DOMString value);
  deleter void removeItem(DOMString key);
  void clear();
};
```

Almacenamiento (Web Storage)

Tenemos que tener en cuenta que LocalStorage es el almacenamiento que no expira, y SessionStorage es el que vive sólo en una sesión.

Se pueden guardar objetos JSON, pero se recomienda serializados, en String

Almacenamiento (Web Storage)

Objeto sessionStorage :

```
[NoInterfaceObject]
interface WindowSessionStorage {
  readonly attribute Storage sessionStorage;
};
Window implements WindowSessionStorage;
```

Almacenamiento (Web Storage)

Objeto sessionStorage :

```
sessionStorage.setItem("locura", true);
  var locura = sessionStorage.getItem("locura");
  console.log(locura); // true
  console.log(typeof locura); // 'string'
```

Almacenamiento (Local Storage)

Con JSON:

```
var personaAGuardar = JSON.stringify(persona);
localStorage.setItem("persona", personaAGuardar);
var personaGuardada = localStorage.getItem("persona");

console.log(typeof persona); //object

console.log(typeof personaGuardada); //string

var personaGuardada = JSON.parse(personaGuardada);
console.log(personaGuardada.locura); //true
```

Almacenamiento (WebSQL)

Base de datos embebida

WebSQL de la ayuda de las siguientes plataformas:

- Android
- BlackBerry 10
- IOS
- Tizen

Almacenamiento (WebSQL)

```
function prepareDatabase(ready, error) {
  return openDatabase('documents', '1.0', 'Offline document storage', 5*1024*1024, function (db) {
    db.changeVersion('', '1.0', function (t) {
      t.executeSql('CREATE TABLE docids (id, name)');
    }, error);
  });
}
function showDocCount(db, span) {
  db.readTransaction(function (t) {
    t.executeSql('SELECT COUNT(*) AS c FROM docids', [], function (t, r) {
      span.textContent = r.rows[0].c;
    }, function (t, e) {
      // couldn't read database
      span.textContent = '(unknown: ' + e.message + ')';
    });
  });
}
```

Almacenamiento (WebSQL)

PRACTICA AGENDA DE TELÉFONOS

Contactos

Cordova-plugin-contacts

Nos permite tratar la lista de contactos del dispositivo.

Podemos visualizar información o insertar contactos.

El método navigator.contacts.create es sincrono, y devuelve un nuevo objeto Contact.

Este método no guarda el objeto Contact en la base de datos de contactos del dispositivo, para lo cual es necesario invocar el método Contact.save.

Contactos

cordova-plugin-contacts

Nos permite tratar la lista de contactos del dispositivo.

Podemos visualizar información o insertar contactos.

El método **navigator.contacts.create** es sincrono, y devuelve un nuevo objeto Contact.

Este método no guarda el objeto Contact en la base de datos de contactos del dispositivo, para lo cual es necesario invocar el método **Contact.save**.

Contactos

El método **navigator.contacts.find** se ejecuta de forma asíncrona, buscando en la base de datos de contactos del dispositivo y devuelve una matriz de objetos de contacto.

Los objetos resultantes se pasan a la función de devolución de llamada (callback) **contactSuccess** especificado por el parámetro contactSuccess.

Contactos

```
function onSuccess(contacts) {
    alert('Found ' + contacts.length + ' contacts.');
};
function onError(contactError) {
    alert('onError!');
};
// find all contacts with 'Bob' in any name field
var options      = new ContactFindOptions();
options.filter   = "Bob";
options.multiple = true;
options.desiredFields = [navigator.contacts.fieldType.id];
options.hasPhoneNumber = true;
var fields       = [navigator.contacts.fieldType.displayName,
navigator.contacts.fieldType.name];
navigator.contacts.find(fields, onSuccess, onError, options);
```

Contactos

El método **navigator.contacts.find** se ejecuta de forma asíncrona, buscando en la base de datos de contactos del dispositivo y devuelve una matriz de objetos de contacto.

Los objetos resultantes se pasan a la función de devolución de llamada (callback) **contactSuccess** especificado por el parámetro contactSuccess.