Desarrollo de aplicaciones móviles con PhoneGap

API Reference

- Accelerometer
- Camera
- Capture
- Compass
- Connection
- Contacts
- Device
- Events
- File
- Geolocation
- Globalization
- InAppBrowser
- Media
- Notification
- Splashscreen
- Storage

Para poder utilizar todo esto
únicamente tenemos que declarar:

```
<script src="phonegap.js"></script>
```

Hoy vamos a ver

- Contacts API
- Camera API
- File API
- Demo

Contacts

> The `contacts` object provides access to the device contacts database.

Nos permite consultar la base de datos de contactos del dispostivo, así como crear, y borrar contactos.

```
navigator.contacts.create()
navigator.contact.find()
contact.clone()
contact.remove()
```

Creación de un nuevo contacto

La creación de un contacto se hace de forma directa:

```
var contact = navigator.contacts.create();
```

En este punto tenemos un objeto Contact al que le podemos ir incorporando propiedades con la información del contacto: `id`, `displayName`, `phoneNumbers`, etc...

```
contact.name = “A name”;
contact.note = “A note”;
```

El último paso es guardar el contacto:

```
contact.save();
```

Búsqueda de contactos

Para iniciar una búsqueda hay que ejecutar el siguiente código:

```
navigator.contacts.find(contactFields,
onContactSearchSuccess,
onContactSearchError,
searchOptions)
```

El parámetro contactFields define los campos que serán incluidos en la búsqueda.

Podemos incluir todos los campos:

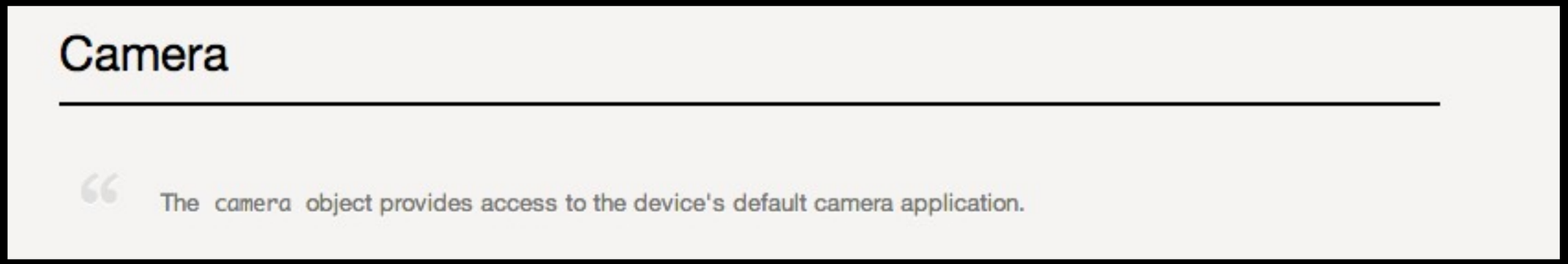
```
contactFields = [‘*’];
```

Podemos incluir solo ciertos campos:

```
contactFields = [‘displayName’, ‘name’, nickname];
```

El parámetro searchOptions, consta de dos parámetros,y define como es realizada la búsqueda:

```
var searchOptions = {filter: searchStr, multiple: true}
```



El Camera API proporciona el método `navigator.camera.getPicture` y un objeto `cameraOptions`.

navigator.camera.getPicture es utilizado para obtener la imagen, y cameraOptions es usado para especificar diversas opciones como calidad de la imagen, formato del fichero, etc.

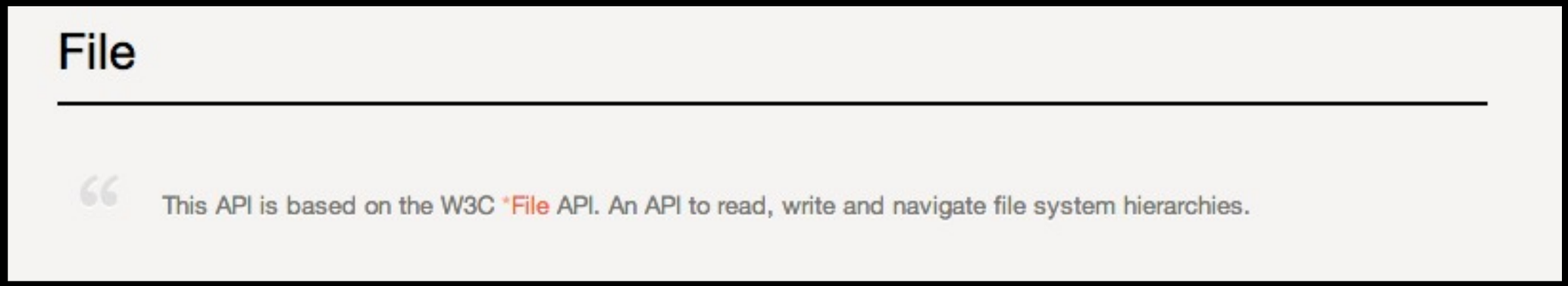
Acceso a la cámara de fotos

```
$("#camera-button").on(click, function() {
    navigator.camera.getPicture(onCameraSuccess, onCameraError);
});

function onCameraSuccess(imageURL) {
    // do whatever you want
}

function onCameraError(error) {
    // error
}
```

Una vez que la imagen ha sido obtenida de la cámara, se ejecuta la función onCameraSuccess, que toma como parámetro la URI apuntando al fichero que contiene la imagen que ha sido creada.

Camera options

- quality: valores de hasta 100, que devuelve la imagen con la mayor calidad posible.
- destinationType: una URI apuntando al fichero del dispositivo que contiene la fotografia
- sourceType: podemos obtener la foto tanto de la cámara como de la libreria de fotos del teléfono.
- allowEdit: Permite editar la imagen (mover y escalarla) seleccionada antes de devolver el control a la aplicación.
- encodingType: JPEG o PNG
- targetWidth y targetHeight: se puede seleccionar uno u otro y la imagen será escalada para mantener la relación de aspecto
- mediaType



Con esta API tenemos, la posbilidad de acceder al sistema de archivos del dispositivo, de forma que podemos crear, leer, escribir, mover y borrar ficheros del mismo.

En este API hay una gran cantidad de tipos de objetos...

FileSystem
DirectoryEntry
DirectoryReader
File
FileEntry
FileError
FileTransfer
FileUploadOptions
FileUploadResult
FileWriter
Flags
LocalFileSystem
Metadata

Acceso al sistema de ficheros (objeto FileSystem)

El sistema de ficheros se representa por un objeto de tipo FileSystem.

Lo primero que tenemos que hacer es obtener una referencia al objeto de tipo FileSystem que representa el sistema de ficheros del dispositivo:

```
function onDeviceReady() {
   window.requestFileSystem(LocalFileSystem.PERSISTENT,
   0,
   onFileSystemSuccess,
   null);
}

function onFileSystemSuccess(fileSystem) {
   // get directories and files
   // filesystem.name
   // filesystem.root
}
```

```
function onFileSystemSuccess(fileSystem) {
    // get directories and files
    // filesystem.name
    // filesystem.root
}
```

El objeto filesystem contiene las siguiente propiedades:

- name: Un string que contiene el nombre del sistema de ficheros del dispositivo.
- root: Un objeto de tipo DirectoryEntry, que representa el directorio raiz del sistema de ficheros.

Un objeto de tipo DirectoryEntry contiene una gran cantidad de métodos para interaccionar y manipular el sistema de ficheros: getDirectory, getFile, moveTo.. (documentación de PhoneGap).

Entonces, la idea es que a partir de filesystem.root puedo ir moviéndome por los distintos ficheros y directorios del dispositivo.

Ejemplo

```
filesystem.root.getFile("sample.txt", {create: false}, processEntry, onFileError);

function processEntry(theFile) {
    //
}
```

Estoy ejecutando el método getFile de la propierdad root del objeto que representa el sistema de ficheros.

El parámetro theEntry es un objeto de tipo FileEntry

Escritura de ficheros (Objeto FileWriter)

Para escribir en un fichero se usa un objeto de tipo FileWriter.

Una vez tengo una referencia a un objeto de tipo FileEntry (mediante el método getFile) tengo que usar un objeto de tipo FileWriter para poder escribir

```
function processEntry(theFile) {
    theFile.createWriter(onCreateWriterSuccess,
onFileError);
}


function onCreateWriterSuccess(writer) {
    writer.write("This is an example");
}
```

Secuencia completa

1. referencia al sistema de ficheros

```
function onDeviceReady() {
    window.requestFileSystem(LocalFileSystem.PERSISTENT,
    0, onFileSystemSuccess, null);
}
```

2. Obtengo la referencia al fichero

```
function onFileSystemSuccess{
    filesystem.root.getFile("sample.txt",
        {create: false}, processEntry, onFileError);
}
```

3. Creo el FileWriter

```
function processEntry(theFile) {
    theFile.createWriter(onCreateWriterSuccess,
onFileError);
}
```

4. Escribo el fichero (por fin!)

```
function onCreateWriterSuccess(writer) {
    writer.write("This is an example");
}
```

Lectura de ficheros

El procedimiento para leer ficheros es el mismo que para escribir, pero sustituyendo los FileWriter por los FileReader

Transferencia de ficheros a un servidor (objetos FileTransfer y FileUploadOptions)

El API de PhoneGap incluye un objeto de tipo FileTransfer que permite a las aplicaciones subir ficheros a un servidor remoto.

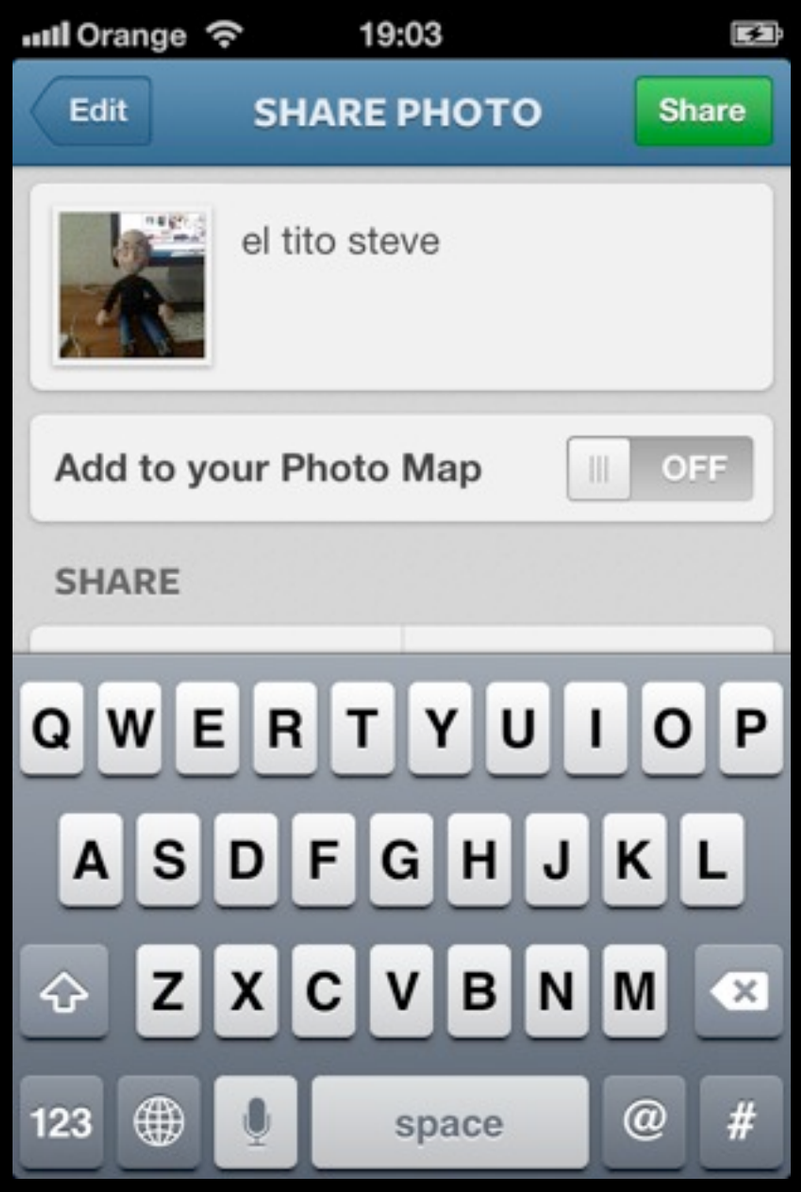
```
var ft = new FileTransfer();
ft.upload(fileURI, serverURL, onUploadSuccess, onUploadError, fileUploadOptions);

function onUploadSuccess(ur) {
    // do whatever you want
}
```

Canvas

```
<canvas id="my-first-canvas" width="360" height="240"></canvas>

<script type="text/javascript">

var canvas = document.getElementById('my-first-canvas');
var context = canvas.getContext('2d');
context.strokeRect(20, 30, 100, 50);

</script>
```

Con el elemento canvas es posible crear herramientas y juegos para las que previamente eran necesarias tecnologías como flash.

Demo: Aplicación tipo Instagram



Demo

- método getPicture y objeto cameraOptions del camera API
- Uso del elemento <canvas> para aplicar los efectos.
- Uso de google app engine como servidor
- uso de objetos FileWriter para escribir ficheros
- uso del objeto FileTransfer para tranferir ficheros.

Demo

instaphonegap.appspot.com

GET: muestro las fotos

instaphonegap.appspot.com/upload
POST: obtengo fichero y texto que se envía y lo guardo en el datastore. Haremos el POST desde nuestra aplicación,