Desarrollo de aplicaciones móviles con PhoneGap

Hoy: El API de PhoneGap

La versión actual es la 2.6.0, aunque prácticamente cada mes tenemos una nueva versión:

PhoneGap 2.5.0 Released 28 Feb 2013 ▸	PhoneGap 2.4.0 Released 07 Feb 2013 ▸	PhoneGap 2.3.0 Released 07 Jan 2013 ▸
PhoneGap 2.2.0 Released 01 Nov 2012 ▸	PhoneGap 2.1.0 Released 21 Sep 2012 ▸	PhoneGap 2.0.0 Released 20 Jul 2012 ▸

http://docs.phonegap.com

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>
```

APIS que vamos a ver hoy

- Events: registro de eventos que se producen en el contexto de una aplicación móvil.
- Device: información acerca del hardware y software del dispositivo
- Connection: Información sobre el estado de la conexión a red del dispositivo (wifi, 3G, sin conexión, etc...)
- Geolocation: Ubicación de la posición espacial del dispositivo.
- Accelerometer: Permite obtener la orientación del dispositivo en un espacio tridimensional.
- Storage: Persistencia de datos.

Events

> Cordova lifecycle events.

Con el API de Events podemos asignar manejadores a los diferentes eventos que ocurren en el contexto de un dispositivo móvil:

evento deviceready: inicialización de PhoneGap

application status events: paso de la aplicación a primer plano, segundo plano, etc.

network events: cambios en la conectividad de la aplicación (online, offline)

button events: pulsación de los botones del dispositivo

Para crear un manejador de eventos, lo hacemos de la forma habitual en JavaScript:

```
document.AddEventListener(“online”, updateNetworkStatus)
```

Por supuesto, también lo podemos hacer con jQuery:

```
$(document).on(“online”, updateNetworkStatus)
```

Evento deviceready

Es una parte fundamental de una aplicación PhoneGap.

Este evento es disparado por PhoneGap para indicar que ha terminado su inicialización y que sus APIs están disponibles para su uso.

Text

Una aplicación deberá usar el API de PhoneGap únicamente cuando el evento deviceready haya sido disparado.

Implementación típica de un manejador de eventos para ondeviceready

```
<body onload="onBodyLoad()">

<script>

function onBodyLoad() {
    document.AddEventListener("deviceready", onDeviceReady, false);
}

function onDeviceReady() {
    // phonegap is ready
}

</script>

</body>
```

Application Status Events

Los teléfonos modernos permite cambiar de aplicaciones. Una aplicación puede pasar de ejecutarse en primer plano a ejecutarse en segundo plano o viceversa.

Si la aplicación pasa a segundo plano, se dispara el evento pause. En el caso de pasar de segundo plano a primer plano, se dispara el evento resume.

El propósito de estos eventos es permitir a una apliación efectuar tareas de limpieza que pueden ser necesarias antes de efectuar la transición.

Device

> The `device` object describes the device's hardware and software.

El objeto device name permite acceder a información relativa al hardware y software del dispositivo. Las propiedades a las que podemos acceder son las siguientes:

device.name : devuelve el nombre asignado al dispositivo; dependiendo de la plataforma puede ser asignado por el fabricante del dispositivo o por el usuario.

device.phonegap : devuelve la versión de PhoneGap usada para desarrollar la aplicación.
device.platform : devuelve el nombre de la plataforma móvil en la que la aplicación se está ejecutando.

device.uuid: devuelve el universally unique identifier asociado al dispositivo.

device.version: devuelve la versión del sistema operativo del dispositivo.

Connection

> The connection object gives access to the device's cellular and wifi connection information.

Podemos acceder a este objeto mediante navigator.network.connection

Expone una única propiedad, connection.type, que indica el tipo de conexión a la red que tenemos disponible.

```
var networkStatUS = navigator.network.connection.type;
if (networkStatus == Connection.NONE) {
    // No network
}
```

Geolocation

> The `geolocation` object provides access to the device's GPS sensor.

Con este API podemos chequear manualmente la posición espacial de nuestro dispositivo, o utilizar un temporizador que cada vez que llegue a cero nos notificará una nueva posición.

Obtención manual de la posición del dispositivo

```
var geolocationOptions = {
    timeout: 3000,
    enableHighAccuray: true
};

navigator.geolocation.getCurrentPosition(
  onGeolocationSuccess,onGeolocationError,
  geolocationOptions);
);
```

```
function onGeolocationSuccess(position) {
  // we have a new position
}
```

Coordinates

A set of properties that describe the geographic coordinates of a position.

Properties :

- **latitude**: Latitude in decimal degrees. *(Number)*
- **longitude**: Longitude in decimal degrees. *(Number)*
- **altitude**: Height of the position in meters above the ellipsoid. *(Number)*
- **accuracy**: Accuracy level of the latitude and longitude coordinates in meters. *(Number)*
- **altitudeAccuracy**: Accuracy level of the altitude coordinate in meters. *(Number)*
- **heading**: Direction of travel, specified in degrees counting clockwise relative to the true north. *(Number)*
- **speed**: Current ground speed of the device, specified in meters per second. *(Number)*

Obteniendo la posición mediante un temporizador

```
watchID = navigator.geolocation.watchPosition(
onGeolocationSuccess, onGeolocationError,
geolocationOptions);
```

Una aplicación puede utilizar posterioremente la variable watchID para cancelar el temporizador, mediante la función clearWatch:

```
navigation.geolocation.ClearWatch(watchID)
```

Geolocations options

- *enableHighAccuracy*: Se habilita si queremos obtener datos más precisos.
- *frequency*: Esta opción solo es válida cuando estamos usando un temporizador, e indica cada cuanto tiempo se debe obtener la posición.
- maximun Age: Especifica el tiempo máximo que una posición cacheada será aceptada por la aplicación.
- *timeout*: máxima cantidad de tiempo que puede pasar desde que se hace una llamada a getCurrentPosition o watchPosition hasta que se obitene una posición.

Permite obtener la orientación de dispositivo en un espacio tridimensional.

Su “filosofía” es la misma que el geolocation API, que acabamos de ver.

```
navigator.accelerometer.getCurrentAcceleration(onAccelSuccces, onAccelFailure);

watchID = navigator.accelerometer.watchAcceleration(onAccelSuccces,
onAccelFailure);

navigator.accelerometer.clearWatch(watchID);
```

La mayoría de los navegadores compatibles con HTML5 proporcionan a las aplicaciones una API JavaScript que permite:

Lectura y escritura de pares clave/valor
acceso a lectura y escritura de una base de datos SQL local (Web SQL).

Una aplicación de PhoneGap puede aprovechar estas características del navegador, pero no es algo específico de PhoneGap.

Local Storage

Mediante esta opción podemos almacenar datos mediante pares clave/valor.

Un buen caso de uso sería almacenar opciones de configuración de nuestra aplicación.

Para almacenar un valor, ejecutamos el siguiente código:

```
window.localStorage.setItem("position", "value");
```

Para obtener un valor almacenado, ejecutamos el siguiente código:

```
var myData = window.localStorage.setItem("position");
```

Demo

- evento deviceready
- network events
- algunos elementos nuevos de jquery mobile (navbar, fieldsets)
- uso de watchPosition()
- localStorage
- API de google maps