Ivan MADJAROV

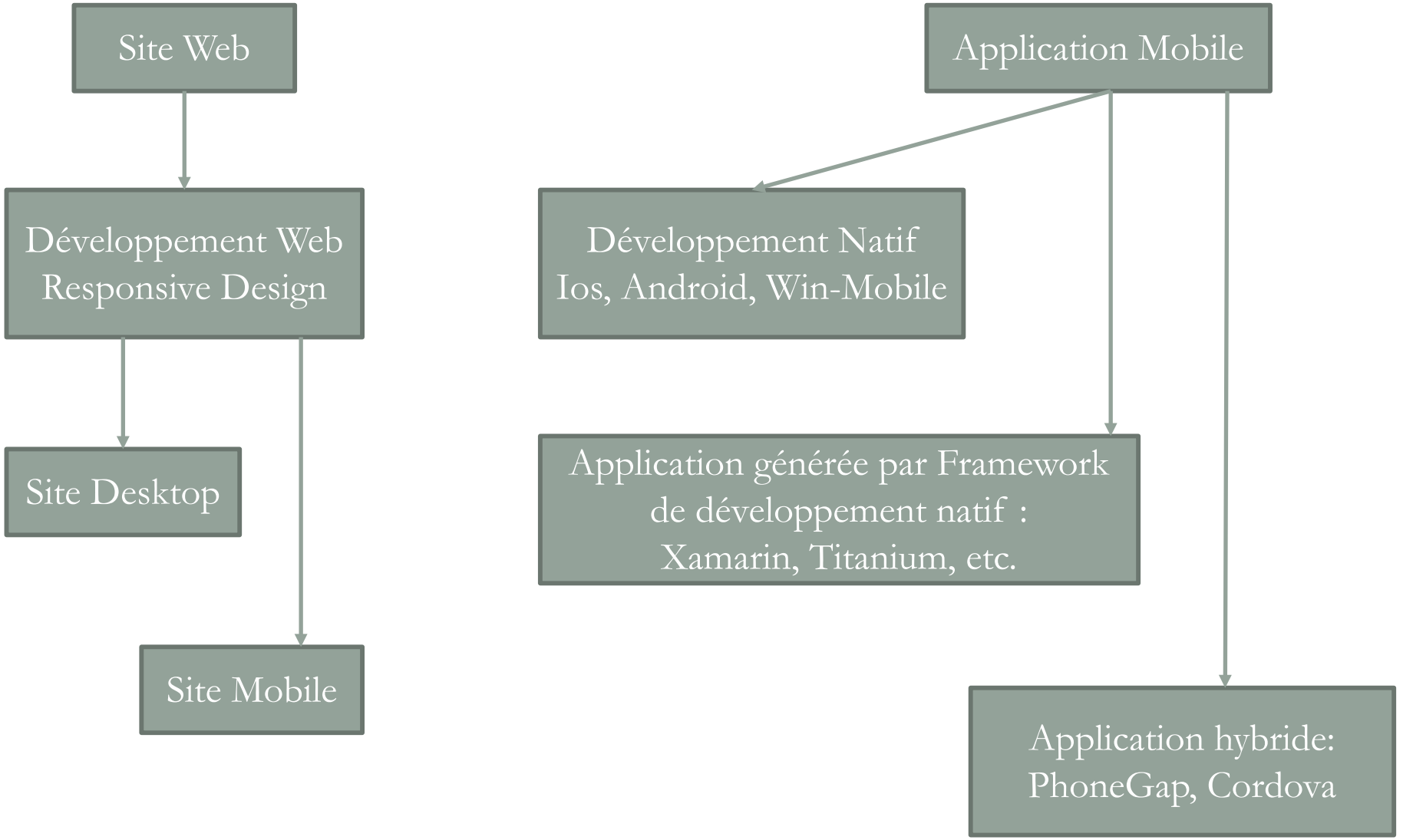
# APPLICATION MOBILE CORDOVA

## HTML5
## CSS3
## JavaScript
## Cordova

# Introduction

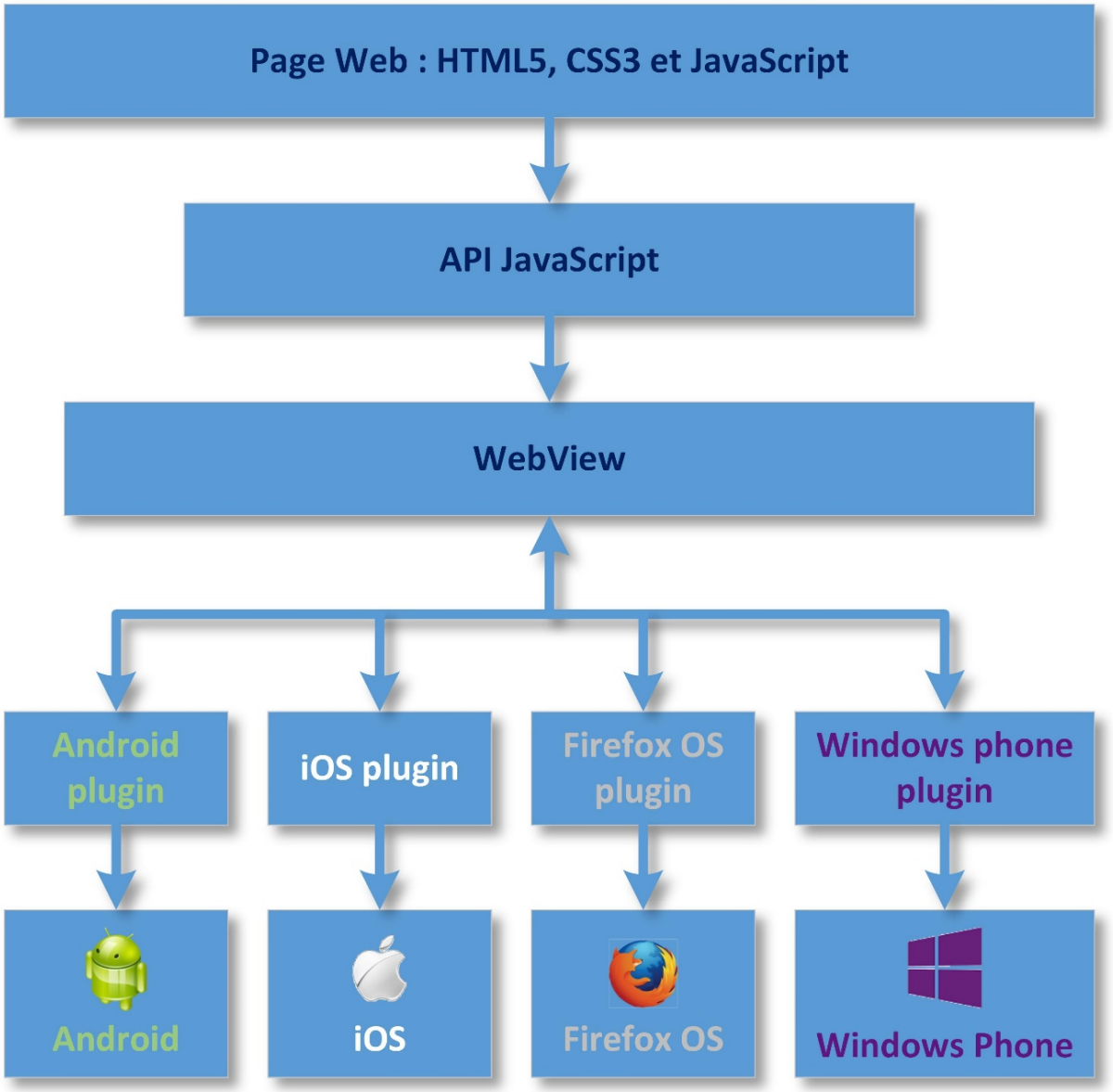
- Développement d'applications multiplateformes pour appareils mobiles:
  - Une tâche complexe
  - Les différents systèmes sont incompatibles en portage d'applications.
- Alternative de développement multiplateforme mobile:
  - Cordova : HTML5, CSS3 et JavaScript.
  - Cordova est un conteneur pour interfacer l'application Web avec les fonctionnalités natives de l'appareil mobile.
    - La caméra, le GPS, le système de fichiers, la géolocalisation, etc.
  - Pour les étudiants DUT R&T et M2P SIR cette plateforme met en valeur les connaissances acquises dans les modules de développement Web et la programmation pour appareils mobiles.

# Application Web hybride et native



# Principe de fonctionnement

- Cordova est une API JavaScript.
- Cordova sert de *wrapper* pour rendre cohérente l'application mobile avec les dispositifs de l'appareil.
- Systèmes concernés:
  - iOS, Android, Windows Phone, Firefox OS, Ubuntu Phone.



# Installer et configurer Cordova

1. Installer *NodeJS* ( `https://nodejs.org/` ).
   - *NodeJS* est une plateforme événementielle en JavaScript orientée vers les applications réseaux avec une bibliothèque de serveur HTTP intégrée.
2. Installer *Cordova* à partir de l'invité de commandes ( cmd - admin):

   ```
   >npm install -g cordova@latest
   ```
3. Installer la dernière version du *Java Development Kit* à partir du site

   ```
   >http://www.oracle.com/technetwork/java/javase/downloads/
   ```
   - Déclarer la variable d'environnement `JAVA_HOME` pointant vers la racine du dossier Java, Ex: `C:\Program Files\Java\jdk1.8.0_65`.
4. Installer le *SDK Android* ( `http://developer.android.com/sdk/` )
   - Ajouter les dossiers `tools` et `platform-tools` dans le `PATH` des variables d'environnement.
5. Si absent, installer *Ant* ( `http://ant.apache.org/bindownload.cgi` ) et l'ajouter au `PATH` du système. *Ant* gère la compilation du projet.

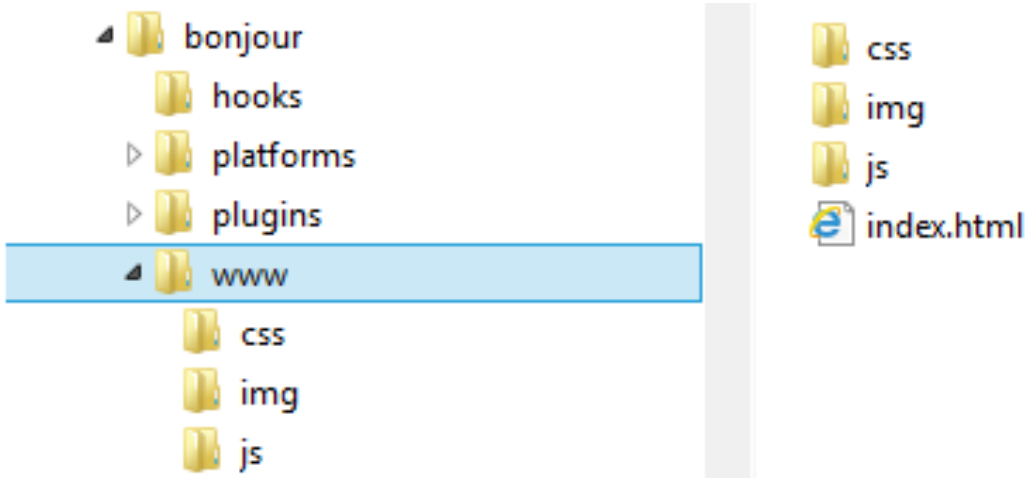
# Créer une application Cordova

- On peut créer le projet "*bonjour*" avec la commande :

```
>cordova create bonjour
```

- On accède au répertoire du projet :

```
>cd bonjour
```




- On ajoute la plateforme cible :

```
>cordova platform add android --save
```

- La configuration du projet est ainsi sauvegardée dans le fichier *bonjour/config.xml*. Le fichier contient le point d'entrée de l'application *<content src="index.html" />*. L'élément *<name>* spécifie le nom de l'application.
- Le fichier *bonjour/www/index.html* contient le squelette d'une application qu'on peut modifier ou remplacer.

# Code d'une application Cordova

```
<html>
<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="initial-scale=1, width=device-width"/>
  <title>Cordova Android</title>
  <script type="text/javascript" src="cordova.js"> </script>
  <script type="text/javascript">
    var showMessageBox = function() {
       document.write("<h2 style='text-align: center;'>
            Bonjour<br/>tout le monde<br/>de<br/>Cordova</h2>
       <img style='margin:0px auto;display:block'
            src='img/logo.png' height='auto' width='auto'/>");
    }
    function init(){
      document.addEventListener("deviceready", showMessageBox, true);
    }
  </script>
</head>
<body onload="init();">
</body>
</html>
```

# Mise en route d'une application Cordova

- Pour compiler le projet :

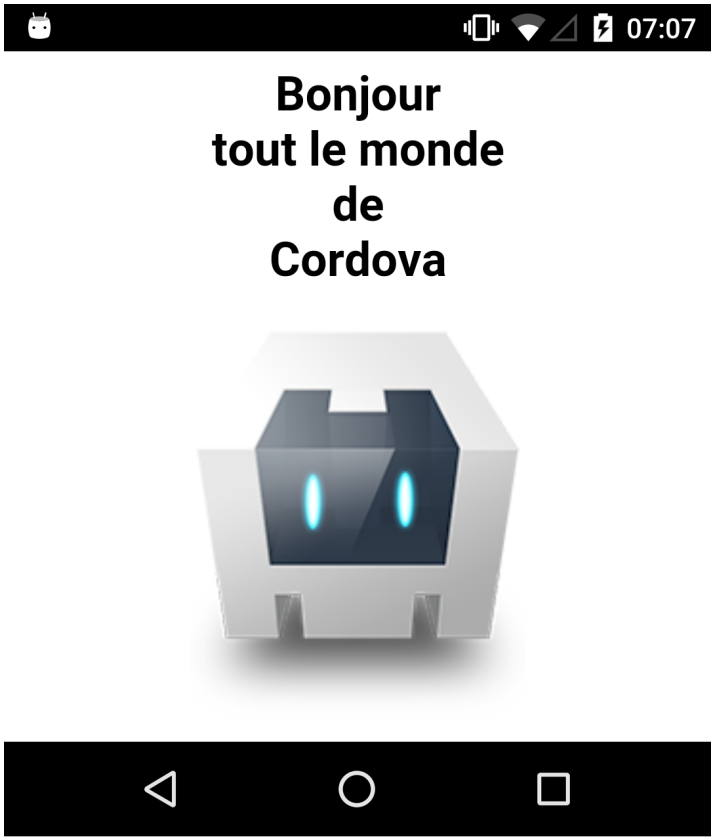
```
>cordova build
```

- Pour tester l'application on peut lancer le simulateur intégré dans le SDK Android :

```
>cordova emulate android
```

- ou sur un appareil mobile par connexion USB :

```
>cordova run android
```




Remarque: Les pilotes USB pour smartphone et tablette sont accessibles à installer à partir du répertoire *android-sdk*

# Comment ça marche Cordova

- La balise `<meta>` de la section `<head>` dans le fichier `index.html` spécifie le codage du document avec l'attribut `charset` et son adaptabilité vis-à-vis les différentes tailles et résolutions d'écrans mobiles avec l'attribut `viewport`.
- La bibliothèque *JavaScript* nécessaire au fonctionnement de *Cordova* est introduite dans la section `<head>` avec la balise `<script>` :

```
<script type="text/javascript" src="cordova.js"> </script>
```

- Les fonctionnalités de la bibliothèque ne sont pas disponibles immédiatement. La réception d'une notification est nécessaire. Un appel à la fonction `init()` est effectué au chargement de la page *HTML* au travers l'évènement `onload` référencé par la balise `<body>`. Après chargement de l'API, la fonction `showMessageBox` est appelée :

```
document.addEventListener("deviceready", showMessageBox, true);
```

# Une application Cordova responsive

- On obtient plus facilement une présentation élaborée et adaptative (*responsive*) sur un appareil mobile avec la technique des feuilles de style (CSS3).
- La norme CSS3 déclare la mise en forme des pages HTML et exploite ainsi les avantages de la séparation du contenu et de la forme.
  - La spécification CSS3 *Media Queries* définit les techniques pour adapter la présentation en fonction du périphérique utilisé.
- Le site de W3Schools (`http://www.w3schools.com/w3css/`) propose un projet CSS3 adapté aux appareils mobiles.
  - W3.CSS est un *framework* compact, rapide et plus petit par rapport à d'autres du même type. Facile à apprendre, il est entièrement basé sur la norme CSS, (*aucune bibliothèque jQuery ou JavaScript n'est nécessaire*).

# Les plugins Cordova

- Un plugin Cordova est un code d'extension (*add-on*) qui fournit l'interface JavaScript pour communiquer avec les composants natifs de l'appareil mobile.
- Le plugin permet à l'application d'utiliser les capacités natives du périphérique au-delà de ce qui est disponible pour une application Web classique.
- Pour ajouter la dernière version du plugin pour la caméra :

```
>cordova plugin add cordova-plugin-camera@latest --save
```

- Pour enlever le plugin de la caméra :

```
>cordova plugin rm cordova-plugin-camera --save
```

# L'API Cordova : Device

- L'objet Device fournit des d'informations sur l'unité mobile et sur l'application en cours.
- L'API s'installe par le plugin *cordova-plugin-device*.

```
>cordova plugin add cordova-plugin-device@latest --save
```

- Dans le code on fait appel à la fonction *onBodyLoad()* lorsque l'événement *onLoad* est déclenché au chargement de la page HTML.
- Après notification du chargement de l'API une liste non-ordonnée compacte les informations dans la fonction *onDeviceReady()*.
- L'attribut *innerHTML* remplace le contenu identifié par *devInfo* sur la page HTML.

# L'API Cordova : Device

```
function onBodyLoad() {
document.addEventListener("deviceready",onDeviceReady,true);
}
function onDeviceReady() {
var tmpStr='';
tmpStr+='<ul class="w3-ul">';
tmpStr+='<li>Cordova Version: '+device.cordova+'</li>';
tmpStr+='<li>Operating System: '+device.platform+'</li>';
tmpStr+='<li>OS Version: '+device.version+'</li>';
tmpStr+='<li>Device Model: '+device.model+'</li>';
tmpStr+='<li>Universally Unique Identifier: '+device.uuid+
</li>';
tmpStr+='</ul>';
document.getElementById('devInfo').innerHTML=tmpStr;
}
```

# L'API Cordova : Device

# L'API Cordova : Device

```
<!DOCTYPE html>
<html>
<head>
<meta content="text/html; charset=utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0" />
<link rel="stylesheet" href="css/w3.css">
<script src="cordova.js"></script>
<script>
function onBodyLoad() {
document.addEventListener("deviceready", onDeviceReady, true);
}
function onDeviceReady() {
var tmpStr = '';
tmpStr += '<ul class="w3-ul">';
tmpStr += '<li> Cordova Version: ' + device.cordova + '</li>';
tmpStr += '<li> Operating System: ' + device.platform + '</li>';
tmpStr += '<li> OS Version: ' + device.version + '</li>';
tmpStr += '<li> Device Model: ' + device.model + '</li>';
tmpStr += '<li> Universally Unique Identifier: ' + device.uuid + '</li>';
tmpStr += '</ul>';
document.getElementById('devInfo').innerHTML = tmpStr;
}
</script>
</head>
<body onload="onBodyLoad()">
<h2 class="w3-container w3-center w3-teal"> Information sur <br/>l'API Cordova</h2>
<p class="w3-center">L'application fait appel à cordova-plugin-device</p>
<p class="w3-center" id="devInfo">En attente d'initialisation de l'API Cordova</p>
</body>
</html>
```

# L'API Cordova : Connect

- L'objet *Connect* fournit des informations sur la connexion réseau de l'appareil et la nature de cette connexion :
  - WiFi, 4G, 3G, ou autre.
- En général, l'application se sert de cette information pour déterminer le moment de transfert de données volumineux depuis ou vers un serveur, ainsi que pour la prise en compte de la précision d'une géolocalisation.
- Pour utiliser l'objet Cordova Connect il faut ajouter le plugin :

```
>cordova plugin add
    cordova-plugin-network-information@latest --save
```

# L'API Cordova : Connect

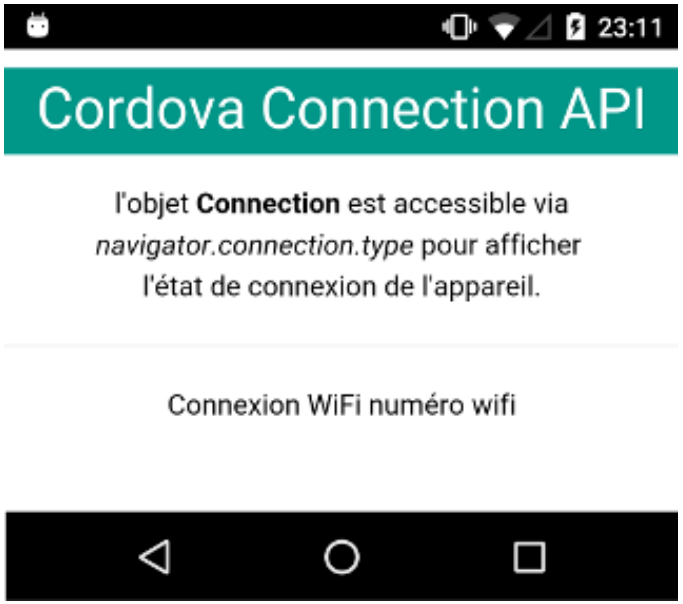
- La propriété `navigator.connection.type` de l'objet `Connection` renvoie l'information sur la connexion disponible pour l'application.
- Un ensemble de constantes définissent le type de connexion :
  - `Connection.CELL_4G`,
  - `Connection.WIFI, ...`
- Dans la fonction `onDeviceReady()` on définit le tableau associatif `states` initialisé avec les différents éléments d'une connexion à identifier par le retour de l'instruction suivante :

```
var networkState = navigator.connection.type;
```

- Le code de l'exemple suivant montre l'identification du type de connexion en cours orientée normalement au transfert de données.

# L'API Cordova : Connect

```
function onBodyLoad() {
document.addEventListener("deviceready", onDeviceReady, false);
}
function onDeviceReady() {
var networkState = navigator.connection.type;
var states = {};
states[Connection.UNKNOWN]='Inconnue';
states[Connection.ETHERNET]='Ethernet';
states[Connection.WIFI]   ='WiFi';
states[Connection.CELL_2G]='cellulaire 2G';
states[Connection.CELL_3G]='cellulaire 3G';
states[Connection.CELL_4G]='cellulaire 4G';
states[Connection.CELL]   ='générique';
states[Connection.NONE]   ='aucune';
document.getElementById('netInfo').innerHTML
= "Connexion "+states[networkState];
}
```



# L'API Cordova : Geolocation

- L'API *Geolocation* est basée sur la spécification de W3C.
- L'objet `navigator.geolocation` permet l'accès aux données de localisation du capteur GPS (*Système de Positionnement Global*) de l'appareil ou déduites des signaux du réseau mobile
  - les IDs cellulaires GSM/CDMA
  - Wi-Fi (l'adresse IP, RFID, les adresses MAC)
  - Bluetooth.
- Les données de géolocalisation sont considérées comme sensibles. Ces données révèlent l'endroit d'utilisation du GPS.
- Une application devrait afficher une notice avant d'accéder aux données par respect de confidentialité et ainsi permettre de recueillir l'autorisation de l'utilisateur de poursuivre.

# L'API Cordova : Geolocation

- La fonctionnalité demande plugin *cordova-plugin-geolocation*.
- Les autorisations nécessaires à la géolocalisation sont apportées au fichier : *platforms/android/res/xml/config.xml*

```
<feature name="Geolocation">
<param name="android-package"
       value="org.apache.cordova.GeoBroker" />
</feature>
```

et au fichier : *platforms/android/AndroidManifest.xml*

```
<uses-permission android:name =
       "android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name=
       "android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name=
       "android.permission.ACCESS_LOCATION_EXTRA_COMMANDS"
/>
```

# L'API Cordova : Geolocation



- Pour détecter la position de l'appareil on fait appel à la fonction asynchrone `getCurrentPosition(par1,[par2],[par3])` dans la fonction `onDeviceReady()`.
- La fonction renvoie la position de l'appareil sous la forme d'objet :

```
navigator.geolocation.getCurrentPosition(onSuccess,
                                         onError);
```

- La fonction `onSucces` est appelée si les mesures de géoposition sont transmises. Elle prend en paramètre l'objet de la localisation pour exposer les coordonnées à travers la propriété `coords`.

```
function onSuccess(pos) {
    var latitude = pos.coords.latitude;
    var longitude = pos.coords.longitude;
}
```

- La fonction `onError` est appelée si aucune donnée n'est transmise.

# L'API Cordova : Geolocation

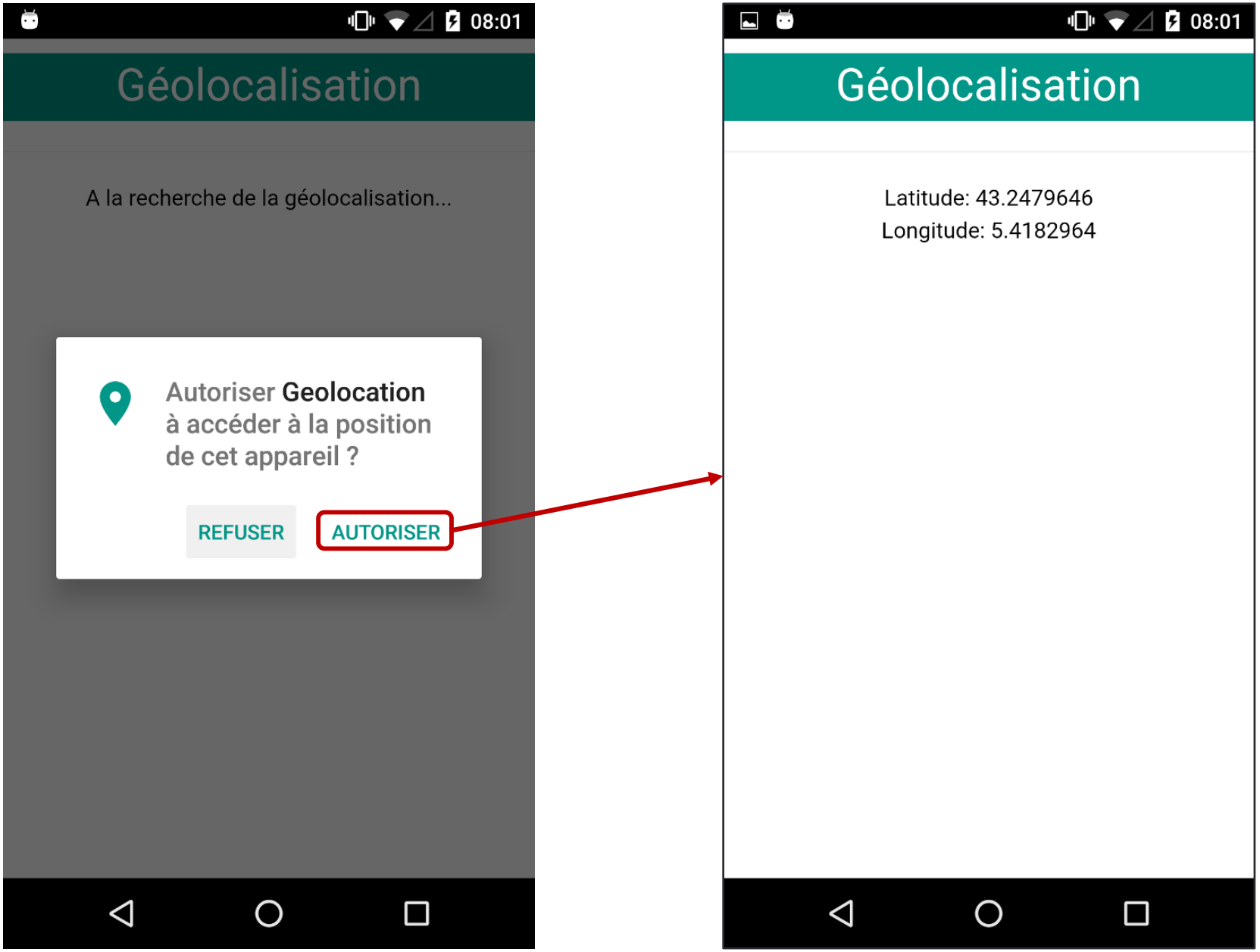
- L'objet géolocalisation fournit des informations sur la mesure d'un ensemble de propriétés incluant :
  - Latitude,
  - Longitude,
  - Altitude,
  - Accuracy,
  - Altitude,
  - Accuracy,
  - Heading,
  - Speed,
  - Timestamp.

| Property | Returns |
|---|---|
| coords.latitude | The latitude as a decimal number (always returned) |
| coords.longitude | The longitude as a decimal number (always returned) |
| coords.accuracy | The accuracy of position (always returned) |
| coords.altitude | The altitude in meters above the mean sea level (returned if available) |
| coords.altitudeAccuracy | The altitude accuracy of position (returned if available) |
| coords.heading | The heading as degrees clockwise from North (returned if available) |
| coords.speed | The speed in meters per second (returned if available) |
| timestamp | The date/time of the response (returned if available) |

# L'API Cordova : Geolocation

```
function onBodyLoad() {
document.addEventListener("deviceready",onDeviceReady,true);
}
function onDeviceReady() {
navigator.geolocation.getCurrentPosition
                                        (onSuccess, onError);
}
function onSuccess(pos) {
document.getElementById('geoloc').innerHTML = 'Latitude: ' +
      pos.coords.latitude + '<br />' +
                    'Longitude: ' + pos.coords.longitude;
}
function onError(error) {
document.getElementById('geoloc').innerHTML ='code: ' +
      error.code + '<br />message: ' + error.message;
}
```

# L'API Cordova : Geolocation



# L'API Cordova : Camera

- L'API *Camera* peut prendre des photos avec la caméra de l'appareil mobile, ou peut choisir des images à partir d'une bibliothèque.
- Pour implémenter ses fonctionnalités il faut ajouter le plugin *cordova-plugin-camera* dans le projet.
- L'objet *navigator.camera* donne l'accès aux fonctions de la caméra.
- L'application devrait afficher une notice avant d'accéder aux fonctionnalités de l'API Camera par respect de la confidentialité et ainsi permettre de recueillir l'autorisation de l'utilisateur de poursuivre.

# L'API Cordova : Camera

- On fait appel à la fonction `getPictures` pour ouvrir l'application par défaut de l'appareil mobile permettant à l'utilisateur de prendre des photos ou de consulter la galerie photos.

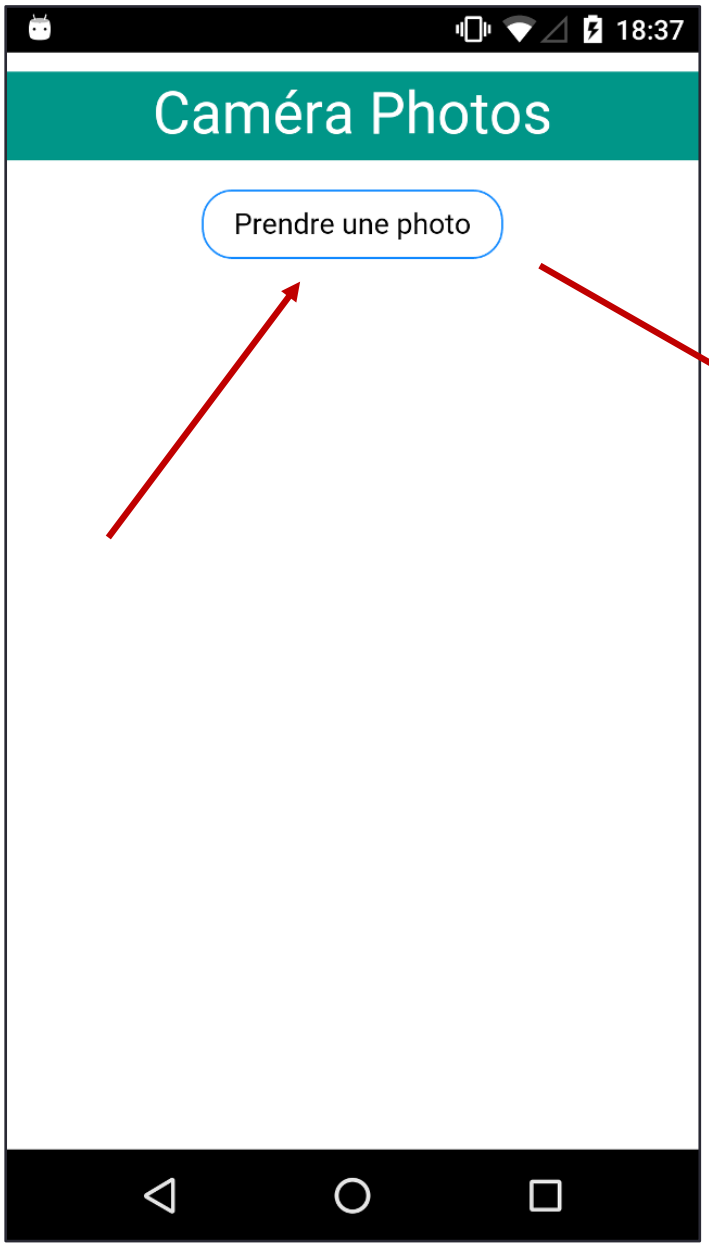
```
navigator.camera.getPicture(onSuccess,onError,[Options]);
```

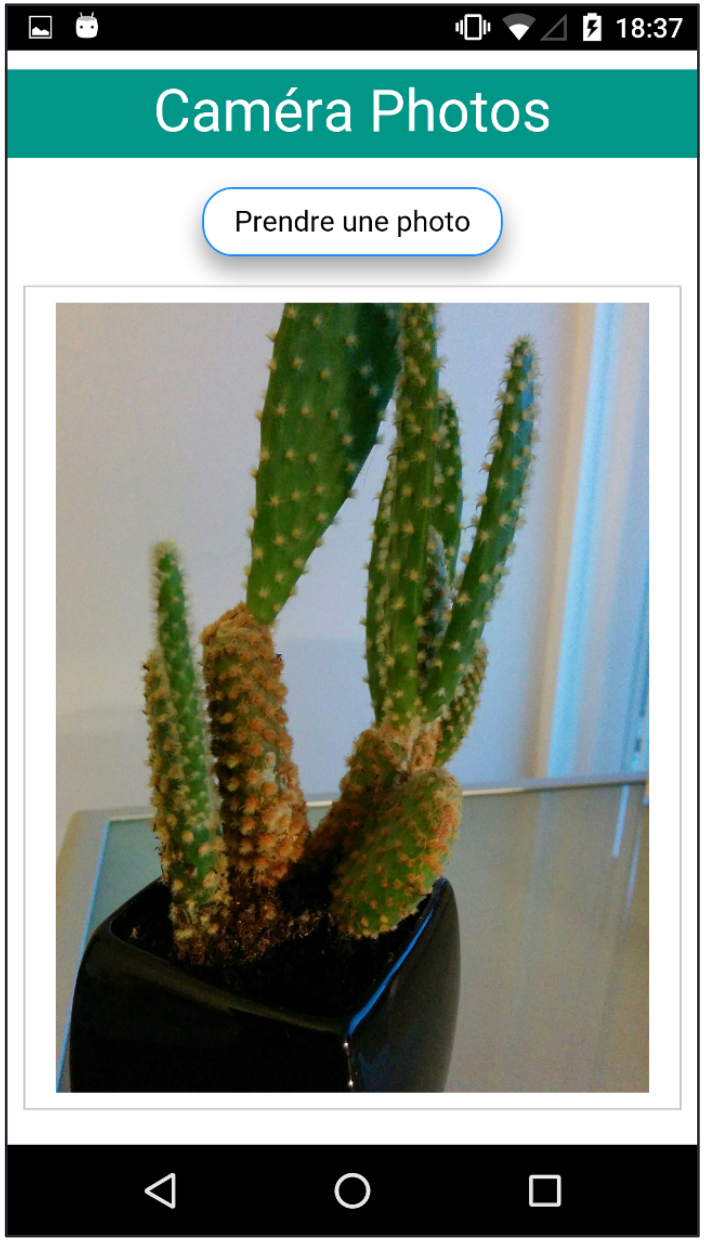
- Trois options sont disponibles :
  1. `Camera.PictureSourceType.CAMERA` active l'application caméra pour la prise de photos;
  2. `Camera.PictureSourceType.PHOTOLIBRARY`, ou
  3. `Camera.PictureSourceType.SAVEDPHOTOALBUM`, ouvre une boîte de dialogue pour accéder à une image existante.
- La fonction passe au "callback" (`onSuccess`) l'image comme une chaine de caractères encodée en base64 ou l'URI du fichier image.

# L'API Cordova : Camera

```
<!DOCTYPE html>
<html>
  <head>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Take Picture</title>
    <link rel="stylesheet" href="css/w3.css">
    <style>
    P { text-align: center }
    </style>
    <script type="text/javascript" charset="utf-8" src="cordova.js"></script>
    <script type="text/javascript" charset="utf-8">
    function onBodyLoad() {
        document.addEventListener("deviceready", onDeviceReady, true);
    }
    function onDeviceReady() {
        document.getElementById("cameraTakePicture").addEventListener("click", cameraTakePicture);
    }
    function cameraTakePicture() {
        navigator.camera.getPicture(onSuccess, onFail, {
            quality: 50,
            destinationType: Camera.DestinationType.DATA_URL
        });
        function onSuccess(imageData) {
            var image = document.getElementById('myImage');
            image.style.display = 'block';
            image.src = "" + imageData;
        }
        function onFail(message) {
            alert('Failed because: ' + message);
        }
    }
    </script>
  </head>
  <body onload="onBodyLoad()">
    <h2 class="w3-container w3-center w3-teal">Cam&eacute;ra Photos</h2>
    <p>
    <button class="w3-btn w3-white w3-border w3-border-blue w3-round-xlarge" id="cameraTakePicture">Prendre une photo</button>
    </p>
    <img style="display:none; margin: 0 auto; width:95%; height:auto;" class="w3-border w3-padding" id="myImage" src="" />
  </body>
</html>
```

# L'API Cordova : Camera



# L'API Cordova : Accelometer

- Un accéléromètre est un capteur qui permet de mesurer l'accélération linéaire d'un mobile. Ainsi, l'API *Accelometer* suit le mouvement de l'appareil en trois dimensions.
- Pour implémenter la fonctionnalité il faut ajouter le plugin *cordova-plugin-device-motion* dans le projet.

```
>cordova plugin add cordova-plugin-device-motion
```

- L'objet *navigator.accelerometer* donne l'accès aux fonctions de l'accéléromètre.
- Pour recueillir les informations du mouvement on fait appel à la fonction : `getCurrentAcceleration(onSuccess, onError);`
- La fonction passe au "callback" (*onSuccess*) les données de accélération.

# L'API Cordova : Accelometer

- Dans le fichier *index.html* on ajoute un bouton pour récupérer les données de l'accéléromètre

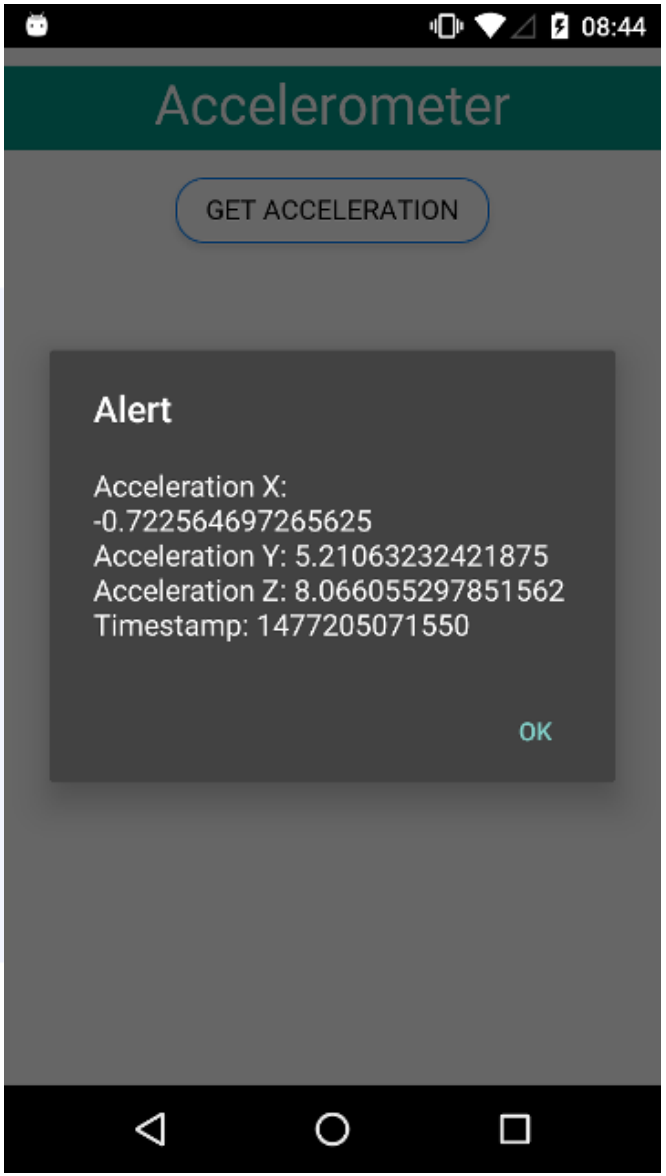
```
<button id="getAcceleration">ACCELERATION</button>
```

- On installe dans la fonction `onDeviceReady` l'écouteur du bouton (*event listener*).

```
document.getElementById("getAcceleration")
        .addEventListener("click", getAcceleration);
```

# L'API Cordova : Accelometer

```
<!DOCTYPE html>
<html>
  <head>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Take Picture</title>
    <link rel="stylesheet" href="css/w3.css">
    <style>
    P { text-align: center }
    </style>
    <script type="text/javascript" charset="utf-8" src="cordova.js"></script>
    <script type="text/javascript" charset="utf-8">
    function onBodyLoad() {
        document.addEventListener("deviceready", onDeviceReady, true);
    }
    function onDeviceReady() {
        document.getElementById("getAcceleration").addEventListener("click",
        getAcceleration);
    }
    function getAcceleration(){
        navigator.accelerometer.getCurrentAcceleration(onSuccess, onError);

        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('Erreur!');
        };
    }
    </script>
  </head>
  <body onload="onBodyLoad()">
    <h2 class="w3-container w3-center w3-teal">Accelerometer</h2>
    <p><button class="w3-btn w3-white w3-border w3-border-blue w3-round-xlarge" id =
    "getAcceleration">GET ACCELERATION</button></p>
  </body>
</html>
```



# L'API Cordova : Device Orientation

- Le compas est utilisé pour montrer l'orientation de l'appareil par rapport au nord géographique.
- Pour implémenter la fonctionnalité il faut ajouter le plugin *cordova-plugin-device-orientation* dans le projet.

```
>cordova plugin add cordova-plugin-device-orientation
```

- L'objet *navigator.compass* donne l'accès aux fonctions de l'orientation géographique.
- Pour recueillir les informations de l'orientation de l'appareil on fait appel à la fonction :

```
navigator.compass.getCurrentHeading(onSuccess, onError);
```

- La fonction passe au "callback" (*onSuccess*) les données de l'orientation.

# L'API Cordova : Device Orientation

- Dans le fichier *index.html* on ajoute un bouton pour récupérer les données de l'accéléromètre

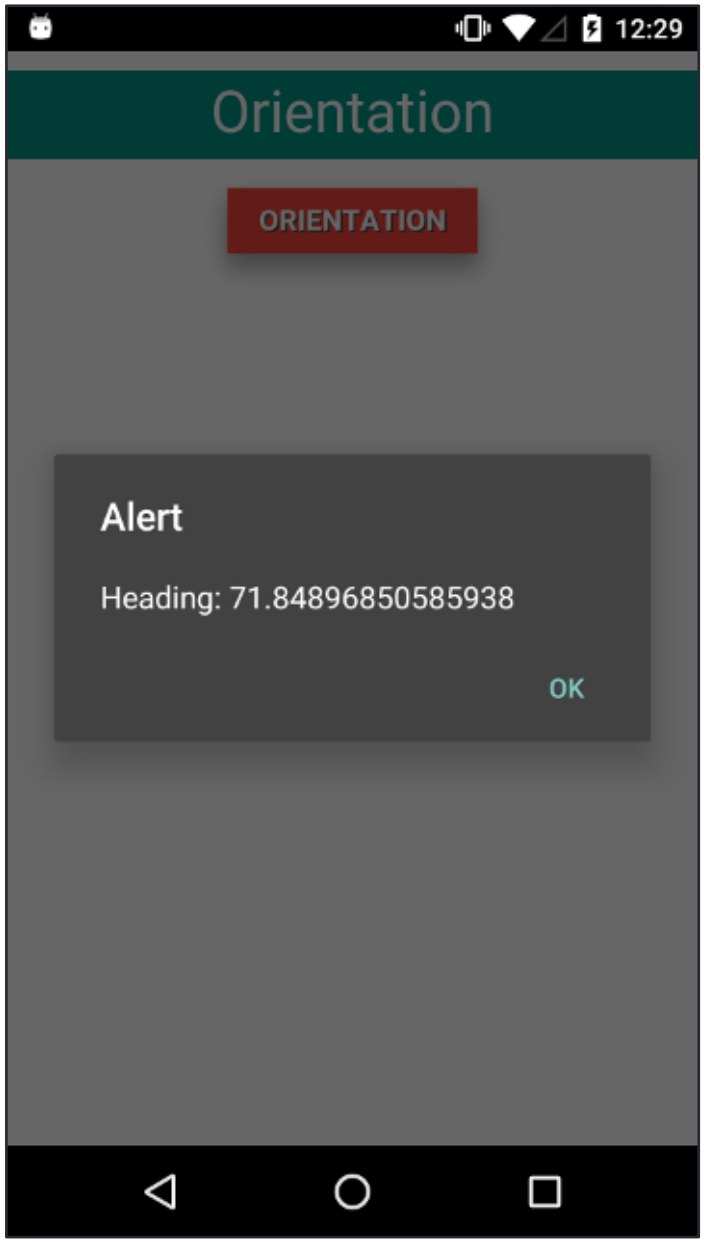
```
<button id = "getOrientation">ORIENTATION</button>
```

- On installe dans la fonction `onDeviceReady` l'écouteur du bouton (*event listener*).

```
document.getElementById("getOrientation")
        .addEventListener("click", getOrientation);
```

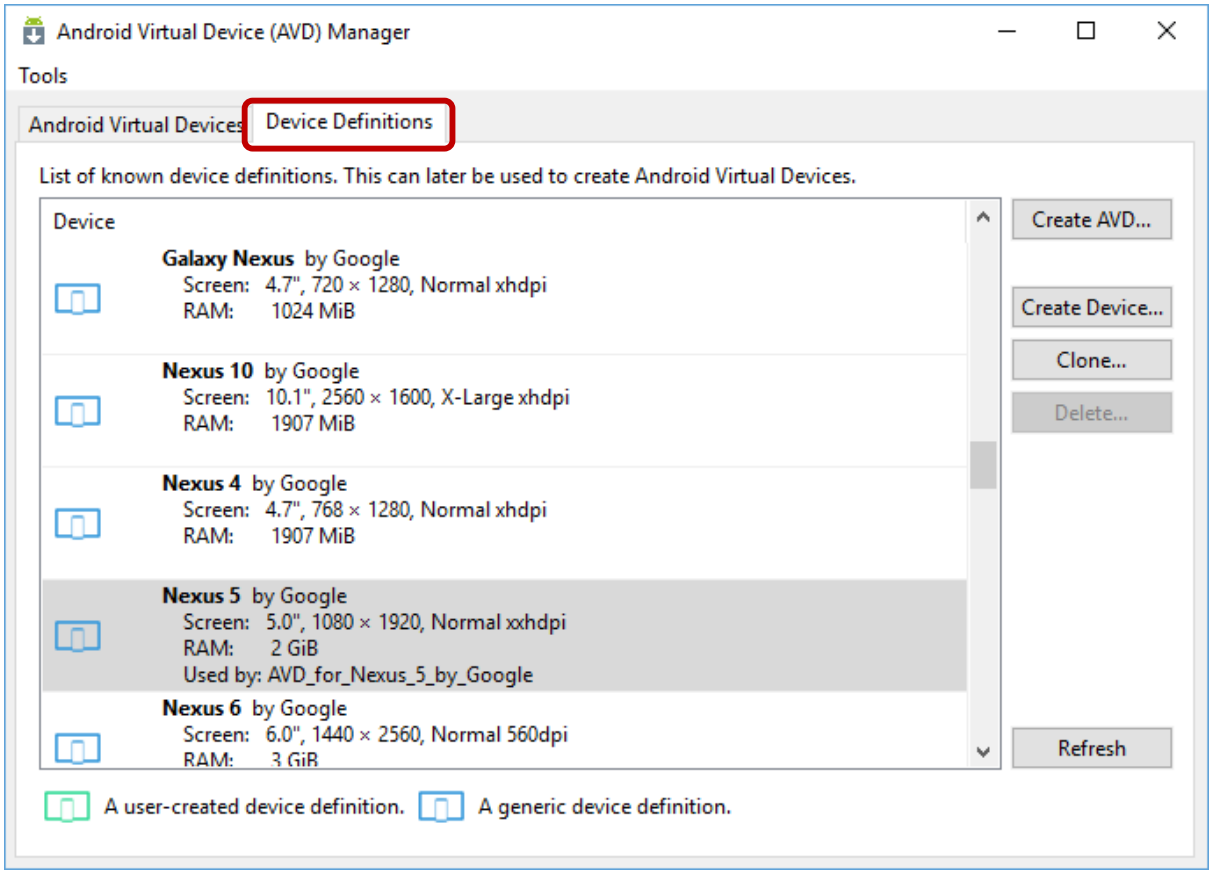
# L'API Cordova : Device Orientation

```
<!DOCTYPE html>
<html>
  <head>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Take Picture</title>
    <link rel="stylesheet" href="css/w3.css">
    <style>
    P { text-align: center }
    </style>
    <script type="text/javascript" charset="utf-8" src="cordova.js"></script>
    <script type="text/javascript" charset="utf-8">
    function onBodyLoad() {
        document.addEventListener("deviceready", onDeviceReady, true);
    }
    function onDeviceReady() {
        document.getElementById("getOrientation").addEventListener("click",
        getOrientation);
    }
    function getOrientation(){
        navigator.compass.getCurrentHeading(compassSuccess, compassError);

        function compassSuccess(heading) {
            alert('Heading: ' + heading.magneticHeading);
        };

        function compassError(error) {
            alert('CompassError: ' + error.code);
        };
    }
  </script>
  </head>
  <body onload="onBodyLoad()">
    <h2 class="w3-container w3-center w3-teal">Orientation</h2>
    <p><button class="w3-btn w3-red w3-text-shadow" id="getOrientation"><b>
    ORIENTATION</b></button></p>
  </body>
</html>
```

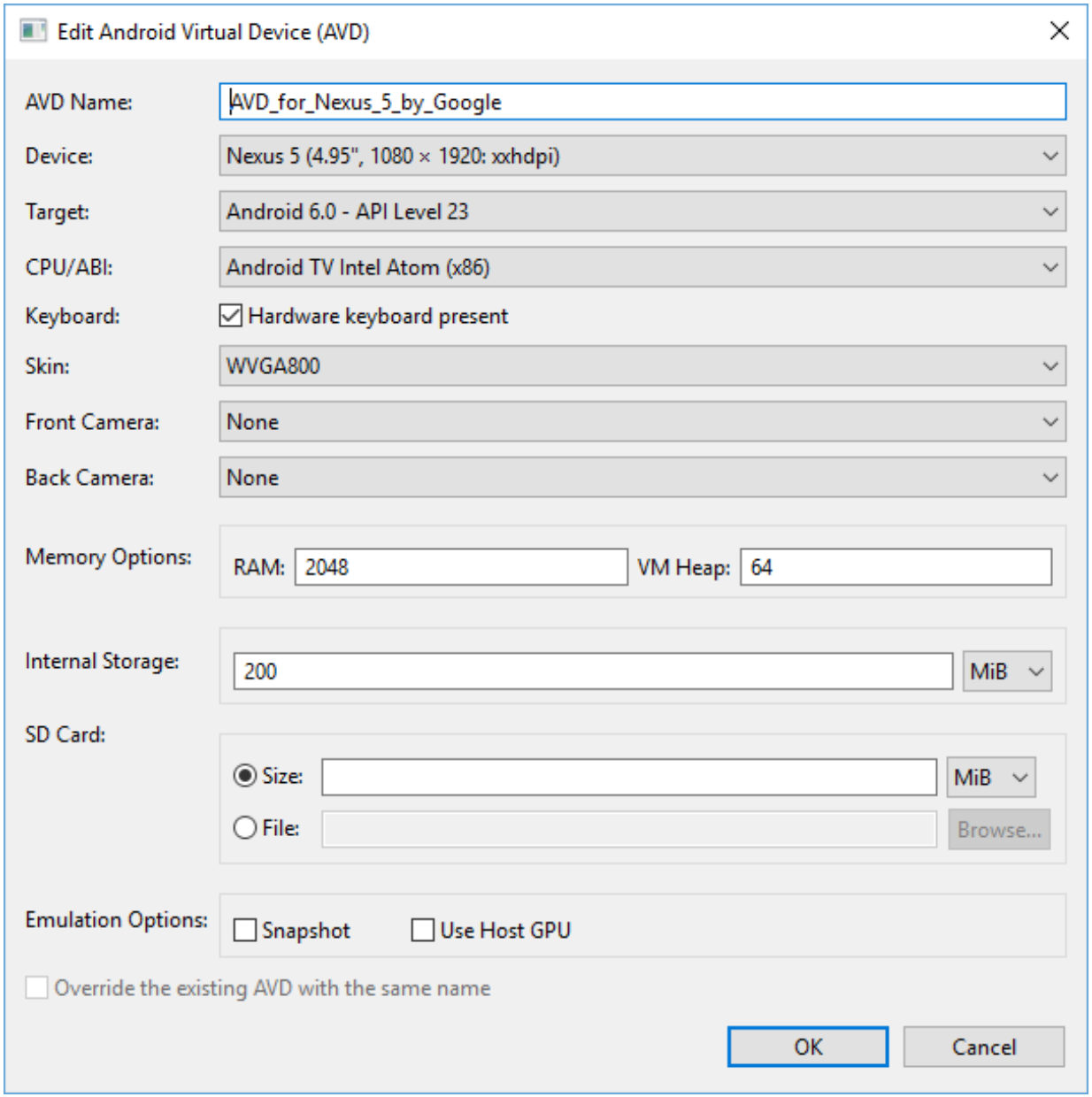


# Configurer un émulateur Android

- Le SDK d'Android ne fournit aucune instance d'émulateur par défaut.
- L'outil AVD Manager (*Android Virtual Device*) fournit l'interface pour la création d'unités virtuelles d'émulation.
- On peut choisir un élément du *Dispositif de définitions* dans la boîte de dialogue

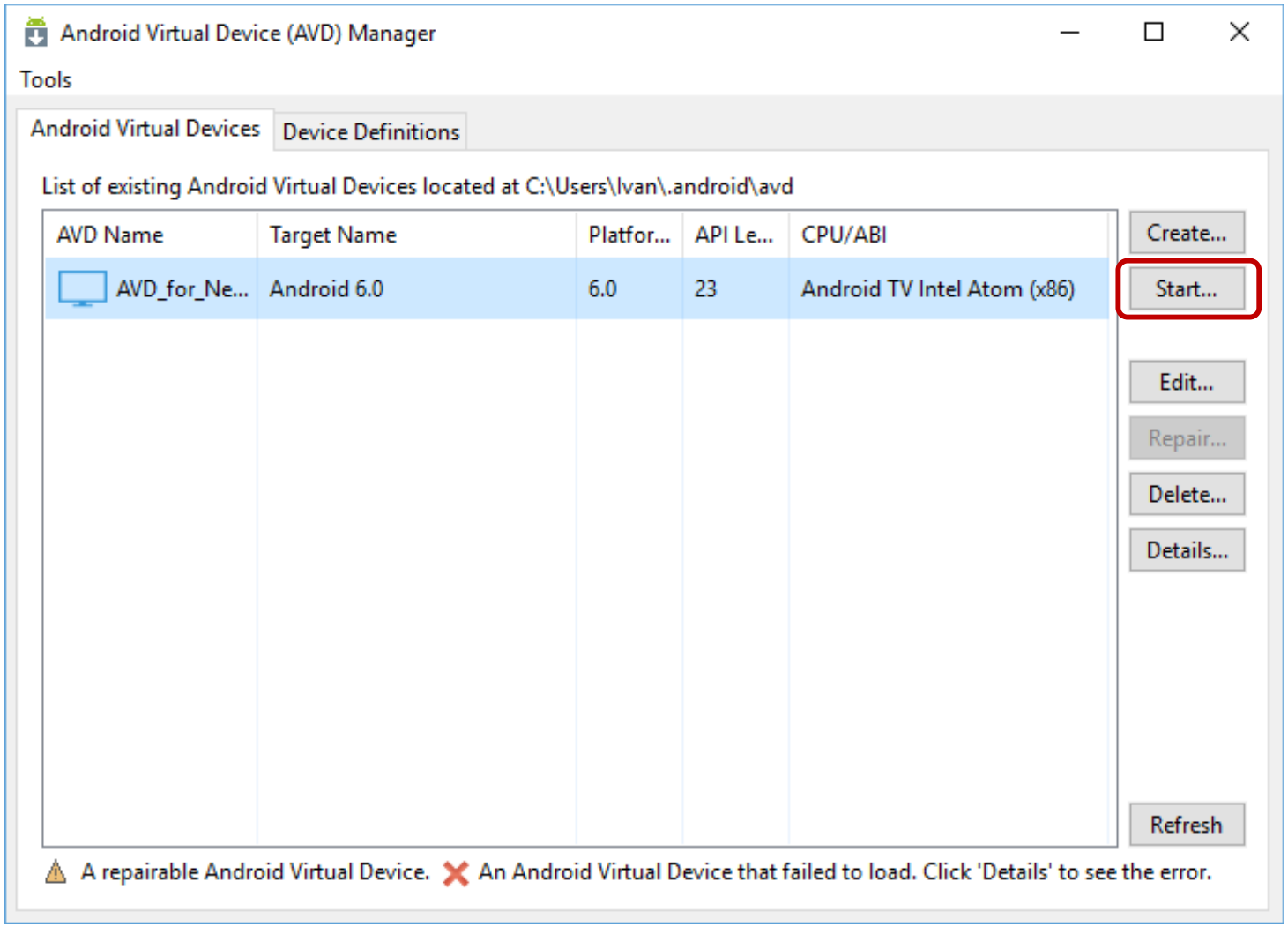


# Configurer un émulateur Android




La configuration peut être sauvegardée et modifiée à tout moment

# Configurer un émulateur Android




L'AVD apparaît dans la liste des périphériques virtuels Android Il faut le lancer par le bouton "Start" pour le rendre disponible pour l'application Cordova

# Configurer un émulateur Android


L'émulateur lancé, l'application Cordova prend l'unité active pour visualiser le résultat de l'application à l'écran

# Version embarquée d'un site Web

- Un site Web adaptatif, répondant aux règles du responsive design pour smartphones et tablettes peut être rendu accessible au travers une application Cordova.
- Ainsi, un site responsive qui s'adapte à tout type de support peut se faire embarquer par une application Cordova. On se sert de l'objet *window* de JavaScript pour ouvrir une fenêtre avec sa méthode *open* dans le navigateur.
- 3 paramètres sont à renseigner :

```
window.open(URL [,nom] [,options]);
```

  - *URL* contient l'adresse de la page à afficher.
  - *nom* de la fenêtre.
  - *options* à paramétrer pour la fenêtre

# Configurer un émulateur Android


L'émulateur lancé, l'application Cordova prend l'unité active pour visualiser le résultat de l'application à l'écran

# Version embarquée d'un site Web

- Un site Web adaptatif, répondant aux règles du responsive design pour smartphones et tablettes peut être rendu accessible au travers une application Cordova.
- Ainsi, un site responsive qui s'adapte à tout type de support peut se faire embarquer par une application Cordova. On se sert de l'objet *window* de JavaScript pour ouvrir une fenêtre avec sa méthode *open* dans le navigateur.
- 3 paramètres sont à renseigner :

```
window.open(URL [,nom] [,options]);
```

  - *URL* contient l'adresse de la page à afficher.
  - *nom* de la fenêtre.
  - *options* à paramétrer pour la fenêtre

# Version embarquée d'un site Web

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8" />
<meta name="format-detection" content="telephone=no" />
<!-- empêche l'affichage du numéro téléphone de l'unité portable -->
<!-- meta name="msapplication-tap-highlight" content="no" / à
utliser avec IE8 -->
<meta name="viewport" content="user-scalable=no, initial-scale=1,
maximum-scale=1, minimum-scale=1, width=device-width, height=device-
height, target-densitydpi=device-dpi" />
<title>Neticiens</title>
<script type="text/javascript">
window.open('http://ivmad.free.fr/neticien', '_self',
'location=no');
</script>
</head>
<body>
</body>
</html>
```

# Version embarquée d'un site Web