CÂMERA, ACELERÔMETRO, COMPASSO E OUTROS EVENTOS

“Nunca confie em um computador que você não pode jogar pela janela.” Steve Wozniak

OBJETO NAVIGATOR.CAMERA

- **Objeto que permite acesso a câmera do dispositivo;**
- **Acessando a funcionalidade:**
  - **Necessário instalar o *plugin* org.apache.cordova.camera;**

OBJETO NAVIGATOR.CAMERA(1)

- **Métodos:**
  - **getPicture: Tira uma foto utilizando a câmera. A foto pode ser acessada através de uma *String* codificada com *base64* ou uma URI;**
  - **cleanup: Remove fotos do armazenamento temporário.**

OBJETO NAVIGATOR.CAMERA(2)

- **getPicture**:

```
navigator.camera.getPicture( cameraSuccess, cameraError, [ cameraOptions ] );
```

- ***cameraOptions***:

```
{ quality : 75,
  destinationType : Camera.DestinationType.DATA_URL,
  sourceType : Camera.PictureSourceType.CAMERA,
  allowEdit : true,
  encodingType: Camera.EncodingType.JPEG,
  targetWidth: 100,
  targetHeight: 100,
  popoverOptions: CameraPopoverOptions,
  saveToPhotoAlbum: false };
```

```
Camera.DestinationType = {
    DATA_URL : 0,       // Return image as base64-encoded string
    FILE_URI : 1,       // Return image file URI
    NATIVE_URI : 2      // Return image native URI (e.g., assets-library:// on iOS or
content:// on Android)
};
```

OBJETO NAVIGATOR.CAMERA(3)

- **Exemplo 01:**

```
navigator.camera.getPicture(onSuccess, onFail, { quality: 50,
    destinationType: Camera.DestinationType.DATA_URL
});

function onSuccess(imageData) {
    var image = document.getElementById('myImage');
    image.src = "" + imageData;
}

function onFail(message) {
    alert('Failed because: ' + message);
}
```

OBJETO NAVIGATOR.CAMERA(4)

- **Exemplo 02:**

```
navigator.camera.getPicture(onSuccess, onFail, { quality: 50,
    destinationType: Camera.DestinationType.FILE_URI });

function onSuccess(imageURI) {
    var image = document.getElementById('myImage');
    image.src = imageURI;
}

function onFail(message) {
    alert('Failed because: ' + message);
}
```

OBJETO NAVIGATOR.ACCELEROMETER

- **Objeto capaz de capturar os movimentos do dispositivos nas coordenadas x, y e z;**
- **Necessário instalar o *plugin* org.apache.cordova.device-motion;**

OBJETO NAVIGATOR.ACCELEROMETER(1)

- **Métodos:**
- **getCurrentAcceleration:** Retorna a aceleração em x, y e z;

```
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!');
};

navigator.accelerometer.getCurrentAcceleration(onSuccess, onError);
```

OBJETO NAVIGATOR.ACCELEROMETER(2)

- **Métodos:**
- **watchAcceleration: Retorna a aceleração em x, y e z em um intervalo de tempo;**

```
var watchID = navigator.accelerometer.watchAcceleration(accelerometerSuccess,
                                                         accelerometerError,
                                                         [accelerometerOptions]);
```

```
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!');
};

var options = { frequency: 3000 };  // Update every 3 seconds

var watchID = navigator.accelerometer.watchAcceleration(onSuccess, onError, options);
```

OBJETO NAVIGATOR.ACCELEROMETER(3)

- **Métodos:**
- **clearWatch:** **Para de observar a aceleração;**

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

OBJETO NAVIGATOR.ACCELEROMETER(4)

- **Objetos:**
- **Acceleration: Objeto que contém dados da aceleração.**

- **x:** Amount of motion on the x-axis. Range [0, 1] (`Number`)
- **y:** Amount of motion on the y-axis. Range [0, 1] (`Number`)
- **z:** Amount of motion on the z-axis. Range [0, 1] (`Number`)
- **timestamp:** Creation timestamp in milliseconds. (`DOMTimeStamp`)

OBJETO NAVIGATOR.COMPASS

- **Objeto que permite perceber a direção do dispositivo;**
- **Acessando a funcionalidade:**
  - **Necessário instalar o *plugin* org.apache.cordova.device-orientation;**

OBJETO NAVIGATOR.COMPASS(1)

- **Métodos:**
  - **getCurrentHeading**: Retorna a direção corrente em graus. Valores entre 0 e 359.99;
  - **watchHeading**: Em um determinado intervalo de tempo observa a direção. Retorna o ID do observador.
  - **clearWatch**: Para de observar a bússola. Utiliza o ID do observador.

OBJETO NAVIGATOR.COMPASS(2)

- **Atributos do objeto CompassOptions**

Atributo	Descrição
frequency	Frequência em milisegundos para retornar a direção.
filter	Mudança em graus antes de iniciar a observar a direção.

OBJETO NAVIGATOR.COMPASS(4)

- **Atributos do objeto CompassHeading**

Atributo	Descrição
magneticHeading	A direção em um determinado momento no tempo.
trueHeading	Direção relativa ao polo norte geográfico em graus. Valores negativos indicam que não foi possível determinar.
headingAccuracy	Diferença, em graus, entre a direção real e a reportada.
timestamp	O tempo no qual a direção foi determinada em milisegundos.

OBJETO NAVIGATOR.COMPASS(5)

- **Tipos de erro CompassError**

  - `CompassError.COMPASS_INTERNAL_ERR`
  - `CompassError.COMPASS_NOT_SUPPORTED`

OBJETO NAVIGATOR.COMPASS(6)

- **getCurrentHeading**

```
// PhoneGap is ready
//
function onDeviceReady() {
    navigator.compass.getCurrentHeading(onSuccess, onError);
}

// onSuccess: Get the current heading
//
function onSuccess(heading) {
    alert('Heading: ' + heading.magneticHeading);
}

// onError: Failed to get the heading
//
function onError(compassError) {
    alert('Compass Error: ' + compassError.code);
}
```

OBJETO NAVIGATOR.COMPASS(7)

- **clearWatch**

```
var watchID = navigator.compass.watchHeading(onSuccess, onError, options);

// ... later on ...

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

EVENTOS DO CORDOVA

- **backbutton**
  - **Disparado quando o usuário pressiona o botão voltar;**

```
document.addEventListener("backbutton", yourCallbackFunction, false);
```

```
document.addEventListener("backbutton", onBackKeyDown, false);

function onBackKeyDown() {
    // Handle the back button
}
```

EVENTOS DO CORDOVA

- **menubutton**
  - **Disparado quando o usuário pressiona o botão menu;**

```
document.addEventListener("menubutton", yourCallbackFunction, false);
```

```
document.addEventListener("menubutton", onMenuKeyDown, false);

function onMenuKeyDown() {
    // Handle the back button
}
```

EVENTOS DO CORDOVA

- **batterycritical**
  - **Nível da bateria está abaixo de um limite crítico;**

```
window.addEventListener("batterycritical", yourCallbackFunction, false);
```

```
window.addEventListener("batterycritical", onBatteryCritical, false);

function onBatteryCritical(info) {
    // Handle the battery critical event
    alert("Battery Level Critical " + info.level + "%\nRecharge Soon!");
}
```

EVENTOS DO CORDOVA

- **batterylow**
  - Nível da bateria está abaixo de um limite baixo;

```
window.addEventListener("batterylow", yourCallbackFunction, false);
```

```
window.addEventListener("batterylow", onBatteryLow, false);

function onBatteryLow(info) {
    // Handle the battery low event
    alert("Battery Level Low " + info.level + "%");
}
```

EVENTOS DO CORDOVA

- **batterystatus**
  - **Retorna se houve alteração no status/nível da bateria;**

```
window.addEventListener("batterystatus", yourCallbackFunction, false);
```

```
window.addEventListener("batterystatus", onBatteryStatus, false);

function onBatteryStatus(info) {
    // Handle the online event
    console.log("Level: " + info.level + " isPlugged: " + info.isPlugged);
}
```