ARQUIVOS, DATABASE E CONNECTION

“Não há razão para qualquer indivíduo ter um computador em casa.” (Popular Mechanics, 1949)

GERENCIAMENTO DE ARQUIVOS

- **Oferece uma API para ler, escrever e navegar no sistema de arquivos, baseado na *File API da W3C(http://www.w3.org/TR/FileAPI/)*;**
- **Para se ter acesso a funcionalidade:**
  - **Necessário instalar o *plugin* org.apache.cordova.file;**

GERENCIAMENTO DE ARQUIVOS

- **Objetos disponíveis:**
  - ***DirectoryEntry***
  - ***DirectoryReader***
  - ***FileEntry***
  - ***FileError***
  - ***FileReader***
  - ***FileSystem***
  - ***FileWriter***
  - ***LocalFileSystem***

GERENCIAMENTO DE ARQUIVOS

- **DirectoryEntry**
- **Representa um diretório no sistema de arquivos.**

Propriedade	Descrição
isFile	Sempre *false*.
isDirectory	Sempre *true*.
name	Nome do diretório.
fullPath	Caminho completo da raiz até o diretório.
filesystem	Sistema de arquivos onde reside o diretório.

GERENCIAMENTO DE ARQUIVOS

– DirectoryEntry

Método	Descrição
remove	Remove o diretório.
copyTo	Copia o diretório para outro local.
moveTo	Move o diretório para outro local.
removeRecursively	Remove recursivamente o diretório e seus subdiretórios.
createReader	Criar um objeto DirectoryReader para ler as entradas do diretório.

GERENCIAMENTO DE ARQUIVOS

– DirectoryEntry

```
function win(entry) {
    console.log("New Path: " + entry.fullPath);
}

function fail(error) {
    alert(error.code);
}

function copyDir(entry) {
    var parent = document.getElementById('parent').value,
        parentName = parent.substring(parent.lastIndexOf('/')+1),
        newName = document.getElementById('newName').value,
        parentEntry = new DirectoryEntry(parentName, parent);

    // copy the directory to a new directory and rename it
    entry.copyTo(parentEntry, newName, success, fail);
}
```

GERENCIAMENTO DE ARQUIVOS

- **DirectoryReader**
- **Lista os diretórios e arquivos localizados em um diretório;**
- **Possui o método readEntries para ler as entradas no diretório;**

```
function success(entries) {
    var i;
    for (i=0; i<entries.length; i++) {
        console.log(entries[i].name);
    }
}

function fail(error) {
    alert("Failed to list directory contents: " + error.code);
}

// Get a directory reader
var directoryReader = dirEntry.createReader();

// Get a list of all the entries in the directory
directoryReader.readEntries(success,fail);
```

GERENCIAMENTO DE ARQUIVOS

- **FileEntry**
- **Representa um arquivo no sistema de arquivos.**

Propriedade	Descrição
isFile	Sempre *true*.
isDirectory	Sempre *false*.
name	Nome do arquivo.
fullPath	Caminho completo da raiz até o arquivo.
filesystem	Sistema de arquivos onde reside o arquivo.

GERENCIAMENTO DE ARQUIVOS

– **FileEntry**

Método	Descrição
remove	Remove o arquivo.
copyTo	Copia o arquivo para outro local.
moveTo	Move o arquivo para outro local.
createWriter	Criar um objeto FileWriter para escrever dados no arquivo.
file	Retorna o arquivo e suas propriedades.

GERENCIAMENTO DE ARQUIVOS

– FileEntry: exemplos

```
function success(entry) {
    console.log("Removal succeeded");
}

function fail(error) {
    alert('Error removing file: ' + error.code);
}

// remove the file
entry.remove(success, fail);
```

```
function success(writer) {
    writer.write("Some text to the file");
}

function fail(error) {
    alert(error.code);
}

// create a FileWriter to write to the file
entry.createWriter(success, fail);
```

GERENCIAMENTO DE ARQUIVOS

- **FileWriter**
- **Permite criar um arquivo e escrever dados nele;**
- **A escrita de dados pode ser feita pelo método *write*;**

```
function win(writer) {
    // fast forwards file pointer to end of file
    writer.seek(writer.length);
};

var fail = function(evt) {
    console.log(error.code);
};

entry.createWriter(win, fail);
```

GERENCIAMENTO DE ARQUIVOS

- **FileReader**
- **Permite acesso de leitura a um arquivo;**
- **Método *readAsText* permite ler como arquivo texto;**

```
function win(file) {
    var reader = new FileReader();
    reader.onloadend = function (evt) {
        console.log("read success");
        console.log(evt.target.result);
    };
    reader.readAsText(file);
};

var fail = function (error) {
    console.log(error.code);
};

entry.file(win, fail);
```

GERENCIAMENTO DE ARQUIVOS

- **FileError**
- **Objeto que representa um erro ao manipular-se arquivos/diretórios;**
- **Os erros podem ser:**

```
FileError.NOT_FOUND_ERR
FileError.SECURITY_ERR
FileError.ABORT_ERR
FileError.NOT_READABLE_ERR
FileError.ENCODING_ERR
FileError.NO_MODIFICATION_ALLOWED_ERR
FileError.INVALID_STATE_ERR
FileError.SYNTAX_ERR
FileError.INVALID_MODIFICATION_ERR
FileError.QUOTA_EXCEEDED_ERR
FileError.TYPE_MISMATCH_ERR
FileError.PATH_EXISTS_ERR
```

GERENCIAMENTO DE ARQUIVOS

- **FileSystem**
- **Representa um sistema de arquivos;**
- **Possui como propriedades:**
  - **name: nome do sistema de arquivos;**
  - **root: diretório raiz do sistema de arquivos;**

```
// Wait for device API libraries to load
//
document.addEventListener("deviceready", onDeviceReady, false);

// device APIs are available
//
function onDeviceReady() {
    window.requestFileSystem(LocalFileSystem.PERSISTENT, 0, onFileSystemSuccess, fail);
}

function onFileSystemSuccess(fileSystem) {
    console.log(fileSystem.name);
    console.log(fileSystem.root.name);
}

function fail(error) {
    console.log(error.code);
}
```

GERENCIAMENTO DE ARQUIVOS

- **LocalFileSystem**
- **Permite acesso a raiz do sistema de arquivos local;**
- **Possui como métodos:**
  - **requestFileSystem: Requisita o sistema de arquivos;**
  - **resolveLocalFileSystemURI: Retorna uma *FileEntry* ou *DirectoryEntry* usando a URI local;**

GERENCIAMENTO DE ARQUIVOS

- **LocalFileSystem**

```
// Wait for device API libraries to load
//
document.addEventListener("deviceready", onDeviceReady, false);

// device APIs are available
//
function onDeviceReady() {
    window.requestFileSystem(LocalFileSystem.PERSISTENT, 0, onFileSystemSuccess, fail);
    window.resolveLocalFileSystemURI("file:///example.txt", onResolveSuccess, fail);
}

function onFileSystemSuccess(fileSystem) {
    console.log(fileSystem.name);
}

function onResolveSuccess(fileEntry) {
    console.log(fileEntry.name);
}

function fail(error) {
    console.log(error.code);
}
```

STORAGE

- **LocalStorage:** também conhecido como web storage, simple storage ou session storage;
- O armazenamento acontece no formato de pares chave/valor;
- Disponível no componente WebView;
- Mais informações: http://www.w3.org/TR/webstorage/

STORAGE

- **IndexedDB:** disponível via WebView.
- São mantidos índices para localizar os valores;
- Também utiliza os pares chave/valor;
- Mais informações em: http://www.w3.org/TR/IndexedDB/.

STORAGE

- **WebSQL:** Oferece uma API para acesso a uma base de dados com tabelas e acesso via SQL;
- Disponível também via WebView;
- Mais informações em: http://dev.w3.org/html5/webdatabase/

STORAGE

```
document.addEventListener("deviceready", onDeviceReady, false);

// PhoneGap is ready
//
function onDeviceReady() {
    var db = window.openDatabase(“DEMODB", "1.0", "PhoneGap Demo", 200000);
    db.transaction(populateDB, errorCB, successCB);
}

// Populate the database
//
function populateDB(tx) {
     tx.executeSql('DROP TABLE IF EXISTS DEMO');
     tx.executeSql('CREATE TABLE IF NOT EXISTS DEMO (id unique, data)');
     tx.executeSql('INSERT INTO DEMO (id, data) VALUES (1, "First row")');
     tx.executeSql('INSERT INTO DEMO (id, data) VALUES (2, "Second row")');
}

// Transaction error callback
//
function errorCB(err) {
    alert("Error processing SQL: "+err);
}

// Transaction success callback
//
function successCB() {
    alert("success!");
}
```

STORAGE

```
function queryDB(tx) {
    tx.executeSql('SELECT * FROM DEMO', [], querySuccess,
errorCB);
}

function querySuccess(tx, results) {
   console.log("Rows Affected = " + results.rowsAffected);
   console.log("Lenght = " + results.rows.length);
}

function errorCB(err) {
    alert("Error processing SQL: "+err.code);
}

// adicionar ao método onDeviceReady
db.transaction(queryDB, errorCB);
```

STORAGE: navegadores

Browser Support

								 Mini	 Mobile
Web Storage - name/value pairs	4+	3.5+	4+	10.5+	8+	3.2+	2.1+	—	11.5+
IndexedDB	23+	10+	—	15+	—	—	4.4	—	0
Web SQL Database	4+	—	3.1+	10.5+	—	3.2+	2.1+	—	11.5+

OBJETO NAVIGATOR.CONNECTION

- Oferece acesso a informações sobre a conexões do dispositivo(wifi ou rede de dados);
- É necessário instalar o plugin:
  - org.apache.cordova.network-information

OBJETO NAVIGATOR.CONNECTION(1)

- **Propriedade: *connection.type***
  - **Permite determinar o tipo e o estado da conexão do dispositivo;**
- **Constantes:**
  - Connection.UNKNOWN
  - Connection.ETHERNET
  - Connection.WIFI
  - Connection.CELL_2G
  - Connection.CELL_3G
  - Connection.CELL_4G
  - Connection.CELL
  - Connection.NONE

OBJETO NAVIGATOR.CONNECTION(2)

```
<script type="text/javascript" charset="utf-8" src="cordova.js"></script>
<script type="text/javascript" charset="utf-8">

// Wait for device API libraries to load
//
document.addEventListener("deviceready", onDeviceReady, false);

// device APIs are available
//
function onDeviceReady() {
    checkConnection();
}

    function checkConnection() {
        var networkState = navigator.connection.type;

        var states = {};
        states[Connection.UNKNOWN]  = 'Unknown connection';
        states[Connection.ETHERNET] = 'Ethernet connection';
        states[Connection.WIFI]     = 'WiFi connection';
        states[Connection.CELL_2G]  = 'Cell 2G connection';
        states[Connection.CELL_3G]  = 'Cell 3G connection';
        states[Connection.CELL_4G]  = 'Cell 4G connection';
        states[Connection.CELL]     = 'Cell generic connection';
        states[Connection.NONE]     = 'No network connection';

        alert('Connection type: ' + states[networkState]);
    }

</script>
```