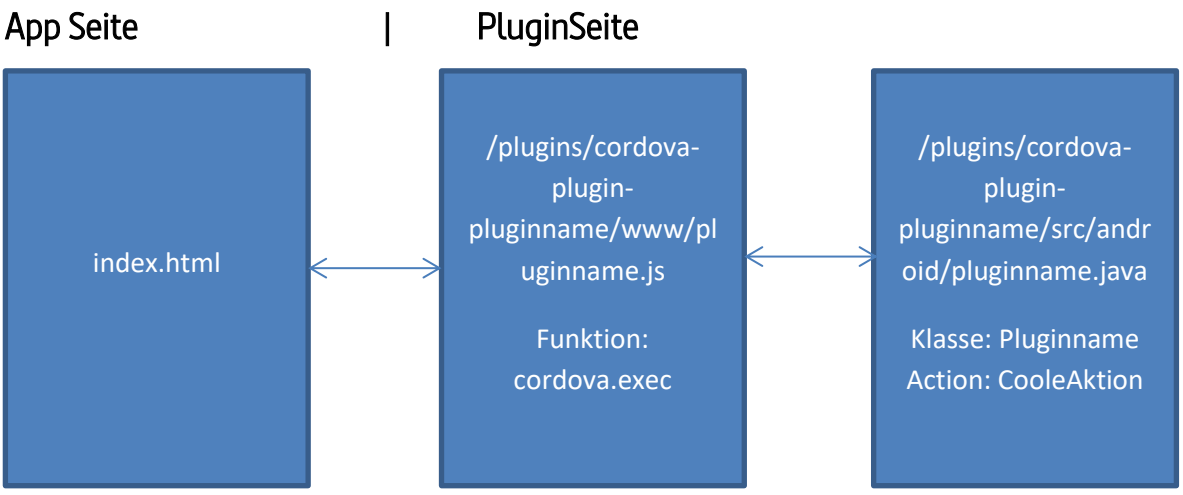
Ein eigenes Plugin für Cordova erstellen.

Diese Anleitung ist ein Versuch einer einfachen und praktischen Dokumentation, zur Erklärung, wie ein Plugin für cordova erstellt wird. Die Motivation für diese Anleitung stammt anhand mangelnder Dokumentationen im Netz. Weiterhin waren die englischsprachigen Anleitungen, welche ich gefunden habe, meist fehlerhaft.
Skills: Es sollten grundlegende Kenntnisse mit dem Umgang von Cordova vorhanden sein.

Die Struktur (Von Javascript zu Java)

Cordova bietet über die Schnittstellenfunktion cordova.exec den Weg von Javascript zu nativen Java. Mit dieser Funktion kann das Smartphone direkt angesprochen werden.

Die Ordnerstruktur:
- index.html
- plugins/cordova-plugin-pluginname/www/pluginname.js
- plugins/cordova-plugin-pluginname/src/android/pluginname.java



Ein neues Cordova Projekt erstellen

```
$ cordova create TestApp com.bitworker.testapp
```

Eine Plattform zum Projekt zufügen

In das Projektverzeichnis wechseln

```
$ cordova platform add android
```

Plugman installieren

```
$ npm install –g plugman
```

Ein Plugin erstellen

Zuerst in den Ordner wechseln indem das Plugin entwickelt wird

- $ plugman create --name <pluginName> --plugin_id <pluginID> --plugin_version <version> [--path <directory>] [--variable NAME=VALUE]

Beispiel:

```
$ plugman create --name TestPlugin --plugin_id cordova-plugin-testplugin --plugin_version 0.0.1
```

Damit das Plugin später bei npm veröffentlicht werden kann ist die die Namenstruktur cordova-plugin-pluginname essentiell.

Eine Plattform zum Plugin zufügen

- In das Plugin Verzeichnis wechseln

```
$ plugman platform add --platform_name android
```

Package.json Datei erstellen

- $ plugman createpackagejson <directory>
- Schreibe ein Punkt (.) am Ende, um das aktuelle Verzeichnis zu verwenden.

Beispiel:

Der Befehl startet ein kurzes Abfragescript aus. In Klammern stehen die Vorschläge, welche automatisch übernommen werden, wen keine Eingabe erfolgt und mit Enter direkt abgeschlossen wird.

- name (cordova-plugin-testplugin)
- version (0.0.1)
- description
- git repository
- author
- license (ISC)

.

.

.

- Is this OK? (yes)

Leider macht Plugman beim Erstellen von Plugins Fehler, wenn der Name des Plugin die Strukur: cordova-plugin-pluginname hat (Also mit Bindestrichen). Diese Struktur ist aber wichtig, um das Plugin evtl. später bei npm (https://www.npmjs.com) veröffentlichen zu können. Wer nicht veröffentlichen möchte, kann auf die Struktur auch verzichten und einen Namen ohne Bindestriche eingeben. Am Ende dieses Artikels, erkläre ich welche Fallstricke behoben werden müssen.

Das Plugin im Cordova Projekt installieren

Bei mir ließ sich das Plugin mit dem Tool Plugman nicht installieren, weswegen ich das Plugin über Cordova installiert habe. Das geht genauso gut und hat keinerlei Nachteile.

Zuerst in das Cordova Projekt Verzeichnis wechseln

- $ cordova plugin add <Pfad zum PluginVerzeichnis>

Beispiel:

```
$ cordova plugin add ..\plugins\TestPlugin
```

Installiere und linke (symbolisch) das Plugin

Besser als es hart zu installieren, ist es, das Plugin symbolisch zu linken. So kann an dem Plugin entwickelt werden, ohne es jedes Mal in die APP kopieren zu müssen. **Achtung:** Auf Windows muss das Terminal (bzw. die Powershell) als Administrator ausgeführt werden, da es ansonsten zu Rechte Problemen kommt.

Zuerst wieder in das Projektverzeichnis wechseln:

```
$ cordova plugin add --link ~/pfad/zum/plugin
```

Das linken des Plugins ist unglaublich praktisch. Damit können Änderungen im nativen Code sofort mit einem Rebuild erzeugt werden. Es ist nicht notwendig das Plugin zuerst zu deinstallieren und dann wieder zu installieren. Bei Änderungen an der Javascript Seite des Plugins ist aber ein deinstallieren weiterhin notwendig. Praktisch gesehen ist damit die Datei: www/pluginname.js Datei im Plugin gemeint!

Der Testing/Development Prozess

Änderungen am Plugin erfordern eine komplette Neuinstallation des Plugin in der Cordova App. (Sofern nicht die oben beschriebene –link Methode verwendet wird). Dazu im Cordova APP Verzeichnis folgende Befehle ausführen. Das Löschen des Plugins ist essentiell, ansonsten werden die Änderungen nicht übernommen!

```
$ cordova plugin rm cordova-plugin-meinname
$ cordova plugin add ~/pfad/zum/plugin
```

Wird das Plugin gelinkt ändern sich die Reinstall Befehle wie folgt ab:

```
$ cordova plugin rm cordova-plugin-meinname
$ cordova plugin add –link ~/pfad/zum/plugin
```

Probleme mit Plugman

TestPlugin.java Fehler

Plugman produziert eine falsche erste Zeile. Java kennt im package keine Bindestriche.

```
package cordova-plugin-testplugin;
```

Einfach die Zeile wie folgt (Punkte statt Bindestriche) anpassen:

```
package cordova.plugin.testplugin;
```

plugin.xml Fehler bei Windows

Bei Windows wird die Datei plugin.xml ohne Zeilenumbrüche erstellt. Hier sollte also nochmal manuell nachgearbeitet werden. Ansonsten kann es schnell unübersichtlich werden.

Windows

Lässt Windows es nicht zu, via Plugman Powershellskripte (.ps1 Dateien) auszuführen, ist es nötig die Windows ExecutionPolicy anzupassen. **Achtung:** Dies kann die Sicherheit des System's massiv negativ beeinflussen. Die ExecutionPolicy sollte nach der Arbeit mit Plugman **unbedingt** wieder rückgängig gemacht werden! **Ich übernehme keine Haftung, wenn Schäden am System auftreten.**
Base: https://www.drwindows.de/windows-10-desktop/146722-poswershellscripte-ausfuehrbar.html

Eine Powershell als Administrator ausführen.

```
$ Set-ExecutionPolicy -Scope 'LocalMachine' -ExecutionPolicy 'RemoteSigned'
```

Befehl zurücksetzen (Nach der Arbeit mit der Plugman - **WICHTIG!**)

```
$ Set-ExecutionPolicy -Scope 'LocalMachine' -ExecutionPolicy 'Undefined'
```

Fallstricke in plugins.xml

Von plugman produzierter Fehler in plugins.xml:

```
<param name="android-package" value="cordova-plugin-isscreenoff.IsScreenOff" />
```

Muss aber

```
<param name="android-package" value="cordova.plugin.isscreenoff.IsScreenOff" />
```

sein!!!

Die Programmierung

Ist das Plugin nun korrekt installiert können wir an die Funktionen und Methoden gegen.

Als Start erstellen wir eine Funktion in der index.html der TestApp

```
    document.addEventListener ('deviceready', function () {

                function testPlugin() {

                   cordova.plugins.IsScreenOff.coolMethod("coolMethod",
function(response){

                       console.log("SUC Message: " + response);

                   }, function(error){

                       console.log("ERR Message: " + error);

                   });

               }

               testPlugin();

    },

    false

  );
```

Diese Funktion ruft Funktion coolMethod in der Plugindatei www/isscreenoff.js auf

In der Datei www/isscreenoff.js ist der folgende Code enthalten:

```
var exec = require('cordova/exec');

exports.coolMethod = function (arg0, success, error) {

    cordova.exec(success, error, 'IsScreenOff', 'coolMethod', [arg0]);

};
```

Datei: www/isscreenoff.js

Der Befehl cordova.exec ruft nun den nativen Code in der Datei: src/android/IsScreenOff.java

```
package cordova.plugin.isscreenoff;

import org.apache.cordova.CordovaPlugin;

import org.apache.cordova.CallbackContext;

import org.json.JSONArray;

import org.json.JSONException;

import org.json.JSONObject;

/**

 * This class echoes a string called from JavaScript.

 */

public class IsScreenOff extends CordovaPlugin {

    @Override

    public boolean execute(String action, JSONArray arg, CallbackContext
callbackContext) throws JSONException {

        if (action.equals("coolMethod")) {

            String message = arg.getString(0);

            this.coolMethod(message, callbackContext);

            return true;

        }

        return false;

    }

    private void coolMethod(String message, CallbackContext callbackContext) {

        if (message != null && message.length() > 0) {

            callbackContext.success(message);

        } else {

            callbackContext.error("Expected one non-empty string argument.");

        }

    }

}
```

Plugin bei npmjs.com veröffentlichen

Wenn die Entwicklung des Plugin beendet ist, kann es bei npmjs.com veröffentlicht werden, um es der Community zur Verfügung zu stellen. Die Veröffentlichung ist sehr einfach.
Im ersten Schritt ein Konto bei npmjs.com anlegen. Anschließend kann das Plugin mit Plugman wie folgt veröffentlicht werden:

npm adduser # Nur, wenn noch kein Konto bei nmpjs.com vorhanden ist.
npm publish /pfad/zu/dem/Plugin

Wenn bereits ein npm Konto existiert lauten die Befehle wie folgt:

npm login <username>
npm publish /pfad/zu/dem/Plugin

Inegration mit der Cordova Plugin Search Console

Damit das Plugin bei https://cordova.apache.org/plugins gefunden werden kann muss die Datei package.json wie folgt angepasst werden. (ecosystem:cordova)

```
"keywords": [
  "ecosystem:cordova",
  "cordova-android",
  "cordova-ios",
  "cordova-windows"
]
```

Dies wird Support für Android, iOS und Windows erstellen. Wenn das Plugin mit Plugman erstellt wurde, sollten Zeilen bereits vorhanden sein.

Engines zufügen

```
"engines": {
  "cordovaDependencies": {
    "1.0.0": { "cordova-android": "<3.0.0"},
    "2.1.0": { "cordova-android": ">4.0.0"}
  }
}
```

Während die Veröffentlichung bei nmpjs.com relativ zügig von statten geht, dauert es bis zu 12 Stunden bis das Plugin auf der Cordova Seite verfügbar ist (https://cordova.apache.org/plugins)