cordova-plugin-lpapi插件使用方法

创建项目并引入插件

1. 安装cordova
   ```
   npm install -g cordova
   ```
2. 通过cordova创建项目
   ```
   cordova create MyApp
   cordova create hello com.exemple.hell HelloWorld
   ```
3. 添加平台
   ```
   cd hello
   cordova platform add android
   cordova platform add ios
   ```
4. 编译
   ```
   cordova build
   cordova build android
   ```
5. 运行
   ```
   cordova run android
   ```
6. 添加lpapi插件
   ```
   cordova plugin add cordova-plugin-lpapi
   ```
7. 引入测试代码
   在根目录下的 plugins文件夹中找到cordova-plugin-lpapi文件夹，将demo中的文件复制

参考代码

```
async printTest() {
        // 1. 连接打印机
        const printerName = "DP23S-8888888888";
        //
        const labelWidth = 45;
        const labelHeight = 45;
        const orientation = 0;
        //
        const margin = 5;
        const qrcodeWidth = 35;
        const qrcodeText = "德佟电子科技（上海）有限公司";
        LPAPI.openPrinter(printerName, (info) => {
                if (info) {
                        // 开始打印任务
                        await LPAPI.startJob(labelWidth, labelHeight, orientation);
                        // 绘制二维码相关打印信息;
                        await LPAPI.drawQRCode(qrcodeText, margin, margin, labelWidth - margin ,
                        // 提交打印任务，开始打印;
                        await LPAPI.commitJob();
                }
        });
}
```

插件接口介绍

```
/**
 * 以字符串的形式返回已经安装过的所有打印机名称，不同打印机名称间以英文","分隔。
 * @param names 打印机型号，
 *      空字符串表示返回所有型号的打印机
 *      要获取多个型号的打印机，多个型号之间可以用";"进行分割;
 * @param success 成功回调函数;
 * @param fail 失败回调函数;
 * 平台支持：Android+IOS;
 */
    getPrinters(names): Promise<{name}[]>;

/**
 * 打开指定名称的打印机(异步调用)。
 * @param printerName
 *            打印机名称。打印机名称类型：
 *            1、空字符串：打开当前客户端系统上的第一个支持的打印机;
 *            2、打印机型号：例如：DT20S;
 *            3、打印机名称：例如：DT20S-60901687;
 *            4、MAC地址：打开指定地址的打印机，例如：00:18:E4:0C:68:CA。
 * @param success 成功回调函数;
 * @param fail 失败回调函数;
 * 平台支持：Android+IOS;
 */
    openPrinter(printerName, callback): Promise<{} | null>;

/**
 * 断开当前打印机的连接。
 * 平台支持：Android+IOS;
 */
    closePrinter(): void;

/**
 * 得到当前使用的打印机名称。
 *
 * @return 如果已连接打印机，则返回当前使用的打印机名称，否则返回空字符串。
 * 平台支持：Android+IOS;
 */
    getPrinterName(): Promise<string>;

/**
```

```
 * 获取当前打印机的链接信息;
 *
 * @param success 结果回调函数;
 * 平台支持: Android;
 */
    getPrinterState(): Promise<{name, value}>;

/**
 * 获取当前已连接打印机的详细信息;
 *
 * @param success 结果回调函数;
 * 平台支持: Android + IOS;
 */
    getPrinterInfo(): Promise<{}>;

/**
 * 判断当前打印机是否打开（连接成功）?
 * 平台支持: Android;
 */
    isPrinterOpened(): Promise<boolean>;

/**************************************************************************
 * 打印任务的开始，分页，结束等操作。
 **************************************************************************/

/**
 * 以指定的参数，开始一个打印任务。
 *
 * @param width
 *            标签宽度（单位mm）。
 * @param height
 *            标签高度（单位mm）。
 * @param orientation
 *            标签打印方向。0：不旋转；90：顺时针旋转90度；180：旋转180度；270：逆时针旋转90度。
 * @return 成功与否?
 * 平台支持: Android+IOS;
 */
    startJob(width, height, orientation): Promise<boolean>;

/**
 * 开始一个打印页面。
 * @param success 成功回调函数;
 * @param fail 失败回调函数;
```

```
 * 平台支持: Android;
 */
    startPage(): Promise<boolean>;

/**
 * 结束一个打印页面。
 * 平台支持: Android;
 */
    endPage(): void;

/**
 * 结束绘图任务。
 * 调用endJob()后，并未将打印任务提交给打印机，但是可以通过调用getJobPages(),然后以图片形式获取打印
 * 平台支持: Android;
 */
    endJob(): void;

/**
 * 提交打印数据，进行真正的打印。
 * @param success 成功回调函数;
 * @param fail 失败回调函数;
 * 平台支持: Android+IOS;
 */
    commitJob(): Promise<boolean>;

/**
 * 取消当前的打印操作，用于在提交打印任务后执行取消操作。
 * 平台支持: Android;
 */
    cancel(): void;

/**
 * 调用函数endJob() 之后可以调用该函数，获取base64图片格式的打印任务;
 * @param success 数据获取成功回调函数(返回值);
 * 平台支持: Android;
 */
    getJobPages(success): Promise<[]>;

/**
 * 打印base64格式的图片对象
 * @param image base64格式图片或者URL路径;
 * @param args 配置其他打印参数;
 * 平台支持: Android;
```

```
 */
    printImage(image, args): Promise<any>;

/*************************************************************************
 * 打印参数设置。
 *************************************************************************/

/**
 * 得到当前打印动作的顺时针旋转角度。
 *
 * @param success 成功回调函数（返回值）；
 * 当前打印动作的顺时针旋转角度（0，90，180，270）。
 * 平台支持：Android;
 */
    getItemOrientation(): Promise<number>;

/**
 * 设置打印动作的旋转角度。
 *
 * @param orientation
 *            orientation：旋转角度。参数描述如下：
 *            0：不旋转；
 *            90：顺时针旋转90度；
 *            180：旋转180度；
 *            270：逆时针旋转90度。
 * 平台支持：Android + IOS;
 */
    setItemOrientation(orientation): void;

/**
 * 设置打印动作的水平对齐方式。
 *
 * @param alignment
 *            水平对齐方式。参数描述如下：
 *            0：水平居左（默认方式）；
 *            1：水平居中；
 *            2：水平居右。
 * 平台支持：Android + IOS;
 */
    setItemHorizontalAlignment(alignment): Promise<number>;

/**
 * 设置打印动作的垂直对齐方式。
```

```
     *
     * @param alignment
     *            垂直对齐方式，参数描述如下：
     *            0：垂直居上（默认方式）；
     *            1：垂直居中；
     *            2：垂直居下。
     * 平台支持：Android + IOS;
     */
        setItemVerticalAlignment(alignment): void;

    /*************************************************************************
     * 打印对象的绘制操作。
     *************************************************************************/

    /**
     * 打印文本。
     *
     * @param text
     *            文本内容。
     * @param x
     *            打印对象的位置(单位mm)。
     * @param y
     *            打印对象的位置(单位mm)。
     * @param width
     *            打印对象的宽度(单位mm)。
     * @param height
     *            打印对象的高度(单位mm)。
     *            height 为 0 时，真正的打印文本高度会根据内容来扩展；否则当指定的高度不足以打印指定的
     * @param fontHeight
     *            文本的字体高度(单位mm)。
     * @param fontStyle
     *            文本的字体风格（可按位组合），可以不指定，默认为0（正常）。0：正常；1：粗体；2：斜体；
     * @return 打印成功与否?
     */
        drawText(text, x, y, width, height, fontHeight, fontStyle): void;

    /**
     * 以指定的线宽，打印矩形框。
     *
     * @param x
     *            绘制的矩形框的左上角水平位置（单位mm）。
     * @param y
     *            绘制的矩形框的左上角垂直位置（单位mm）。
```

```
     * @param width
     *            绘制的矩形框的水平宽度（单位mm）。
     * @param height
     *            绘制的矩形框的垂直高度（单位mm）。
     * @param lineWidth
     *            矩形框的线宽（单位mm）。矩形框的线宽是向矩形框内部延伸的。
     * @return 打印成功与否?
     */
        drawRectangle(x, y, width, height, lineWidth): void;

    /**
     * 打印填充的矩形框。
     *
     * @param x
     *            绘制的填充矩形框的左上角水平位置（单位mm）。
     * @param y
     *            绘制的填充矩形框的左上角垂直位置（单位mm）。
     * @param width
     *            绘制的填充矩形框的水平宽度（单位mm）。
     * @param height
     *            绘制的填充矩形框的垂直高度（单位mm）。
     * @return 打印成功与否?
     */
        fillRectangle(x, y, width, height): void;

    /**
     * 以指定的线宽，打印圆角矩形框。
     *
     * @param x
     *            绘制的圆角矩形框的左上角水平位置（单位mm）。
     * @param y
     *            绘制的圆角矩形框的左上角垂直位置（单位mm）。
     * @param width
     *            绘制的圆角矩形框的水平宽度（单位mm）。
     * @param height
     *            绘制的圆角矩形框的垂直高度（单位mm）。
     * @param cornerWidth
     *            圆角宽度（单位mm）。
     * @param cornerHeight
     *            圆角高度（单位mm）。
     * @param lineWidth
     *            圆角矩形框的线宽（单位mm）。圆角矩形框的线宽是向圆角矩形框内部延伸的。
     * @return 打印成功与否?
```

```
 */
    drawRoundRectangle(x, y, width, height, cornerWidth, cornerHeight, lineWidth): void;

/**
 * 打印填充的圆角矩形框。
 *
 * @param x
 *            绘制的填充圆角矩形框的左上角水平位置（单位mm）。
 * @param y
 *            绘制的填充圆角矩形框的左上角垂直位置（单位mm）。
 * @param width
 *            绘制的填充圆角矩形框的水平宽度（单位mm）。
 * @param height
 *            绘制的填充圆角矩形框的垂直高度（单位mm）。
 * @param cornerWidth
 *            圆角宽度（单位mm）。
 * @param cornerHeight
 *            圆角高度（单位mm）。
 * @return 打印成功与否?
 */
    fillRoundRectangle(x, y, width, height, cornerWidth, cornerHeight): void;

/**
 * 以指定的线宽，打印椭圆/圆。
 *
 * @param x
 *            绘制的椭圆的左上角水平位置（单位mm）。
 * @param y
 *            绘制的椭圆的左上角垂直位置（单位mm）。
 * @param width
 *            绘制的椭圆的水平宽度（单位mm）。
 * @param height
 *            绘制的椭圆的垂直高度（单位mm）。
 * @param lineWidth
 *            椭圆的线宽（单位mm）。椭圆的线宽是向椭圆内部延伸的。
 * @return 打印成功与否?
 */
    drawEllipse(x, y, width, height, lineWidth): void;

/**
 * 打印填充的椭圆/圆。
 *
 * @param x
```

```
     *            绘制的填充椭圆的左上角水平位置（单位mm）。
     * @param y
     *            绘制的填充椭圆的左上角垂直位置（单位mm）。
     * @param width
     *            绘制的填充椭圆的水平宽度（单位mm）。
     * @param height
     *            绘制的填充椭圆的垂直高度（单位mm）。
     * @return 打印成功与否?
     */
        fillEllipse(x, y, width, height): void;

    /**
     * 以指定的线宽，打印圆。
     *
     * @param x
     *            绘制的圆形的X轴圆心位置（单位mm）。
     * @param y
     *            绘制的圆形的Y轴圆心位置（单位mm）。
     * @param radius
     *            绘制的圆的半径（单位mm）。
     * @param lineWidth
     *            圆的线宽（单位mm）。圆的线宽是向圆内部延伸的。
     * @return 打印成功与否?
     */
        drawCircle(x, y, radius, lineWidth): void;

    /**
     * 打印填充的圆。
     *
     * @param x
     *            绘制的圆形的X轴圆心位置（单位mm）。
     * @param y
     *            绘制的圆形的Y轴圆心位置（单位mm）。
     * @param radius
     *            绘制的填充圆的半径（单位mm）。
     * @return 打印成功与否?
     */
        fillCircle(x, y, radius): void;
    /**
     * 打印线（直线/斜线）。
     *
     * @param x1
     *            线的起点的水平位置（单位mm）。
```

```
 * @param y1
 *            线的起点的垂直位置（单位mm）。
 * @param x2
 *            线的终点的水平位置（单位mm）。
 * @param y2
 *            线的终点的垂直位置（单位mm）。
 * @param lineWidth
 *            线宽（单位mm）。线宽是向线的下方延伸的。
 * @return 打印成功与否?
 */
    drawLine(x1, y1, x2, y2, lineWidth): void;

/**
 * 打印点划线。
 *
 * @param x1
 *            线的起点的水平位置（单位mm）。
 * @param y1
 *            线的起点的垂直位置（单位mm）。
 * @param x2
 *            线的终点的水平位置（单位mm）。
 * @param y2
 *            线的终点的垂直位置（单位mm）。
 * @param dashLen
 *            点划线线段长度的数组（单位mm）。
 * @param lineWidth
 *            线宽（单位mm）。线宽是向线的下方延伸的。
 * @return 打印成功与否。
 */
    drawDashLine(x1, y1, x2, y2, dashLen, lineWidth): void;
/**
 * 打印一维条码。
 *
 * @param text
 *            需要绘制的一维条码的内容。
 * @param x
 *            绘制的一维条码的左上角水平位置（单位mm）。
 * @param y
 *            绘制的一维条码的左上角垂直位置（单位mm）。
 * @param width
 *            一维条码的整体显示宽度。
 * @param height
 *            一维条码的显示高度（包括供人识读文本）。
```

```
     * @param textHeight
     *            供人识读文本的高度（单位mm），建议为3毫米。
     * @param type
     *            一维条码的编码类型参考文档。
     * @return 打印成功与否?
     */
        drawBarcode(text, x, y, width, height, textHeight, type): void;

    /**
     * 打印 QRCode 二维码。
     *
     * @param text
     *            需要绘制的QRCode二维码的内容。
     * @param x
     *            绘制的QRCode二维码的左上角水平位置（单位mm）。
     * @param y
     *            绘制的QRCode二维码的左上角垂直位置（单位mm）。
     * @param width
     *            绘制的QRCode二维码的水平宽度（单位mm）。
     * @return 打印成功与否?
     */
        drawQRCode(text, x, y, width): void;
    /**
     * 打印图片。
     *
     * @param image
     *            图片路径。
     * @param x
     *            打印对象在水平方向上的位置(单位mm)。
     * @param y
     *            打印对象在垂直方向上的位置(单位mm)。
     * @param width
     *            打印对象的宽度(单位mm)。
     * @param height
     *            打印对象的高度(单位mm)。
     * @param threshold
     *            绘制位图的灰度阀值。
     *            256 表示绘制灰度图片;
     *            257 表示绘制原色图片;
     *            0～255表示绘制黑白图片，原图颜色>=灰度阀值的点会被认为是白色，而原图颜色<灰度阀值的点
     * @return 打印成功与否?
     */
        drawImage(image, x, y, width, height, threshold): void;
```