2019/1/20

Fitbit_OSアプリ作成のヒント集(その１)

0.概要
Fitbit_OSアプリ作成において参考になるような情報をまとめた。
公開されているソースから参考になるものを抜粋したり、実際に作成してみた気がついたことをまとめたものである。

1.古いSDKのバージョンでエラー無しでビルドできたものがSDKのバージョンを上げたらエラーになった。
(1)解決方法
　ディレクトリのデフォルト設定が変更になったらしいので以下のような変更をするとエラーが解消する。

修正例：

```
import * as hrm from "hrm";
→
import * as hrm from "./hrm";
```

2.画面タッチとBACKボタンでの画面切り替え
#BACKボタンはscreen1画面以外のときに有効となる
(1)resources/index.gui

```
<svg>
  <svg id="screen1">

    <svg>
            #scrren1用の画面をSVGで記述する。
    </svg>
    <rect id="button1" pointer-events="all" width="100%" height="100%" fill="red" fill-opacity="0.0
" />

  </svg>

  <svg id="screen2" display="none">

    <svg>
            #screen2用の画面をSVGで記述する
    </svg>
    <rect id="button2" pointer-events="all" width="100%" height="100%" fill="blue" fill-opacity="0.0
" />

  </svg>

  <svg id="screen3" display="none">

    <svg>
     #screen3用の画面をSVGで記述する
    </svg>
    <rect id="button3" pointer-events="all" width="100%" height="100%" fill="green" fill-opacity="0.
0" />

  </svg>

</svg>
```

(2)app/index.js

```
import document from "document";

let screen1 = document.getElementById("screen1");
let screen2 = document.getElementById("screen2");
let screen3 = document.getElementById("screen3");

let button1 = document.getElementById("button1");
let button2 = document.getElementById("button2");
let button3 = document.getElementById("button3");

function showScreen1() {
  console.log("Show screen 1");
  screen1.style.display = "inline";
  screen2.style.display = "none";
  screen3.style.display = "none";
}

function showScreen2() {
  console.log("Show screen 2");
  screen1.style.display = "none";
```

```
  screen2.style.display = "inline";
  screen3.style.display = "none";
}

function showScreen3() {
  console.log("Show screen 3");
  screen1.style.display = "none";
  screen2.style.display = "none";
  screen3.style.display = "inline";
}


button1.onclick = function() {
  showScreen2();
}

button2.onclick = function () {
  showScreen3();
}

button3.onclick = function() {
  showScreen1();
}


document.onkeypress = function(evt) {
    if (evt.key === "back") {
      if (screen3.style.display === "inline") {
        // Go to screen 2
        showScreen2();
        evt.preventDefault();
      } else if (screen2.style.display === "inline") {
        // Go to screen 1
        showScreen1();
        evt.preventDefault();
      } else if (screen1.style.display === "inline") {
        // Default behaviour to exit the app
      }
    }
  }
```

(3)参考ＵＲＬ
https://community.fitbit.com/t5/SDK-Development/Wanted-example-of-multiple-UI-views-and-back-navigation/m-p/2300847
Wanted: example of multiple UI views and back navigation

3.マスク用png画像はグレースケールであること
アニメーションの画像用のpng画像はグレースケールである必要がある。
画像編集ソフトなどで修正するとデフォルトがRGBモードが場合、自動的にＲＧＢモードになり
見かけ上、区別がつかないので、そのまま保存して、組み込んだ場合、期待したアニメショーンが動かないことになる。
　したがって修正した画像を保存する場合、グレースケールに変換して保存すること。
ちなみにＧＩＭＰ画像エディタの場合、以下の操作で行なう：
(1)画像ウィンドウのメニューより、画像 → モード → グレースケールでグレースケールに変換する
(2)ファイル→名前を付けてエクスポートで画像名を付けてpngで保存する

4.exportした変数には外部モジュールからは代入はできない(read-onlyになる)
最近のjavascriptの言語仕様でexportした変数に外部モジュールから参照するとread-onlyになり代入はできない。
代入したい場合、関数として定義して、その定義の中で代入を行なう。

例
```
//-----------------------------
//app/foo.js

export let abc=123;

export function assigenABD(val) {
   abc = val;
}

//-----------------------------
//app/othesrs.js

import * as foo from './foo'

consoole.log(foo.abc); // 123が出力される
foo.abc = 456; // エラーになる
```

```
foo.assignABC(456); // エラーにならず代入が可能
```

5.installで失敗してinternal-errorが出るようになった場合、
いったんスマフォ側でuninstallしたほうが良いようだ。そうでないと新たにインストールできないようだ。

以上