Web Application

RICOH
imagine. change.

目次

RICOH
imagine. change.

序章 自己紹介

入門編 Webアプリと操作部ブラウザーNX

- 操作部ブラウザーNX基礎知識

基礎編 操作部ブラウザーNX用JavaScript Library

- どうやってMFPの機能を使うの？

実践編 サンプルコード紹介

- WebアプリからMFPの機能を使ってみよう！

RICOH
imagine. change.

Web Application
入門編

～Webアプリと操作部ブラウザーNX～

Webアプリケーションって？

- Webブラウザ上で動作するアプリケーション

- スマホからアクセスできる場合、デザイン次第でスマホアプリと同じような見た目・操作感を実現することが可能
  ex) facebook, cookpadのwebページ

操作部ブラウザーNXって？

- MultiLink-Panel上に搭載されているソリューション向けブラウザ
- Webアプリから操作部ブラウザーNX用JavaScriptを利用可能

操作部ブラウザーNX用JavaScriptって？

RICOH
imagine. change.

- 例えば、PCからWebサイトを見ていると下のように「印刷」ボタンが付いてることがありますよね。

対応している機器・
ブラウザでしか動作しない

【ソースコード】

```
<form>
<input type="button" value="印刷" onclick="window.print();" />
</form>
```

操作部ブラウザーNX用JavaScriptって？

RICOH imagine. change.

- 同様に、Multi-Link Panel上の操作部ブラウザーNXでしか動作しないJavaScriptがあります。

例

印刷開始	ricoh.dapi.app.printer.start()
スキャン開始	ricoh.dapi.app.scanner.start()
機器番号を取得する	ricoh.dapi.getSerialNumber()

詳細は実践編で紹介します

簡単なアプリケーション例

RICOH
imagine. change.

- スキャンした原稿をサーバーにアップロード

Webアプリの良いところ

- コンテンツの更新が簡単。
- アプリケーションをインストールする必要がないので手軽に始められる。

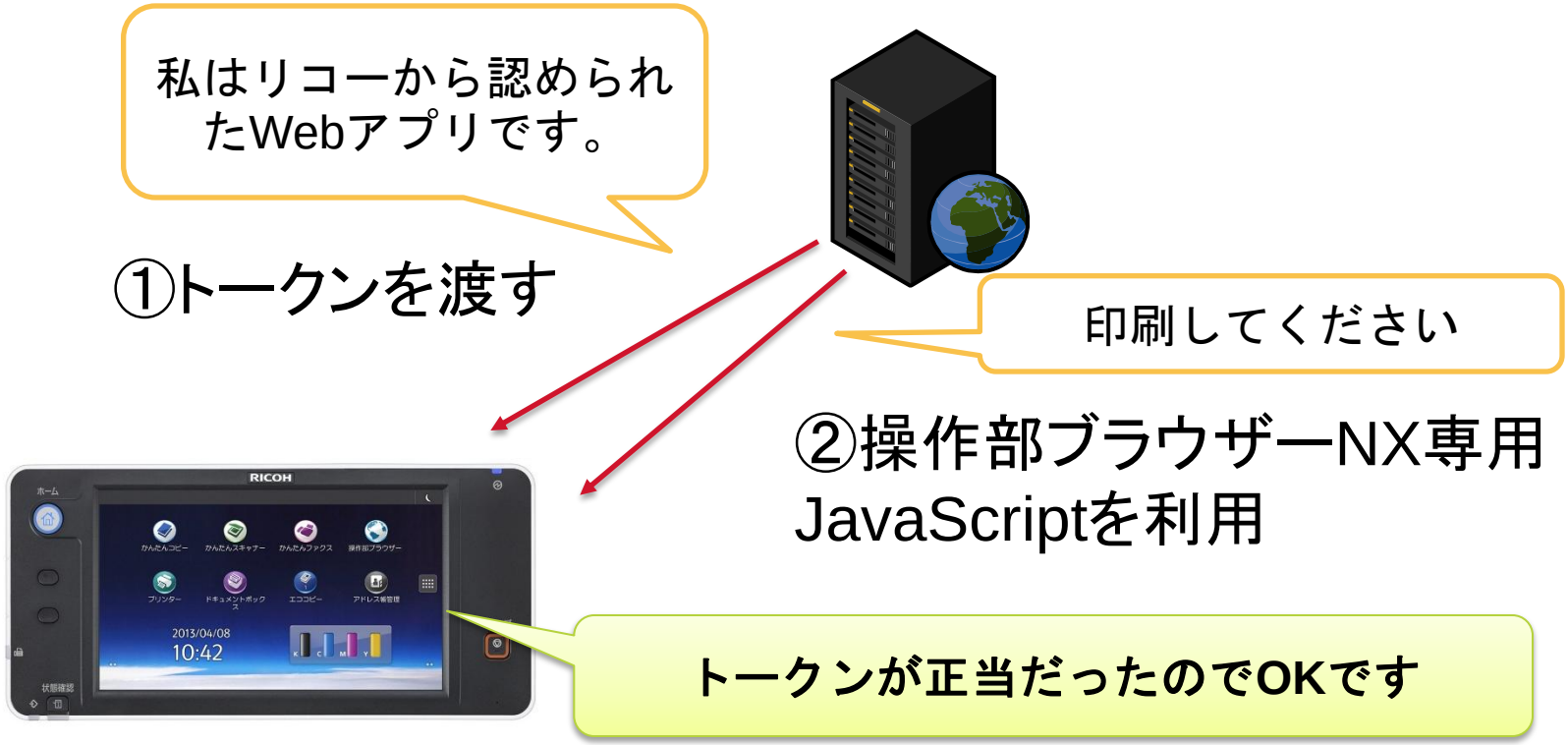
Webアプリの悪いところ

- アクセスするだけで利用できてしまうので･･･
  悪意のあるアプリケーションに攻撃される恐れが！

悪意のあるアプリから守る仕組み

RICOH
imagine. change.

- リコーから発行された「トークン」を使って自分が正当なアプリであることを機器に証明しないと、操作部ブラウザーNX専用JavaScriptを利用できない



悪意のあるアプリから守る仕組み

RICOH
imagine. change.

- リコーから発行された「トークン」を使って自分が正当なアプリであることを機器に証明しないと、操作部ブラウザーNX専用JavaScriptを利用できない

この仕組みを
「正当性検証」(validation)
といいます。

具体的な実装方法は実践編で紹介します

トークンが正当だったのでOKです

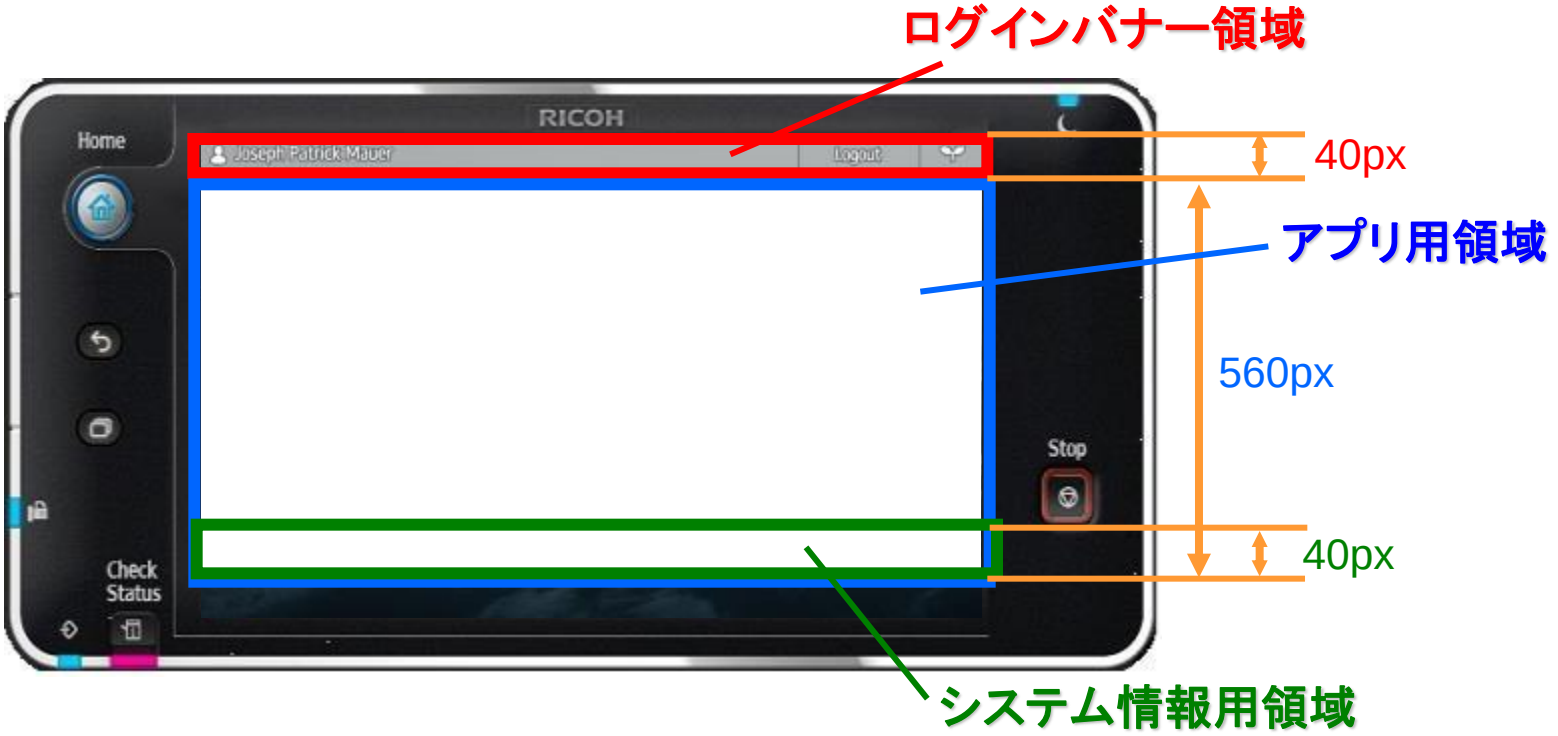
スペック

- ブラウザエンジン
  - Android 2.3.4 Webkit ベース
  - HTML5/CSS3
  - JavaScript/AJAX
  - Webkit セキュリティパッチ

- 操作部ブラウザーNX専用JavaScript API

画面サイズ

- パネルサイズ: 1024 x 600
- うち、アプリで使える領域は1024 x 560



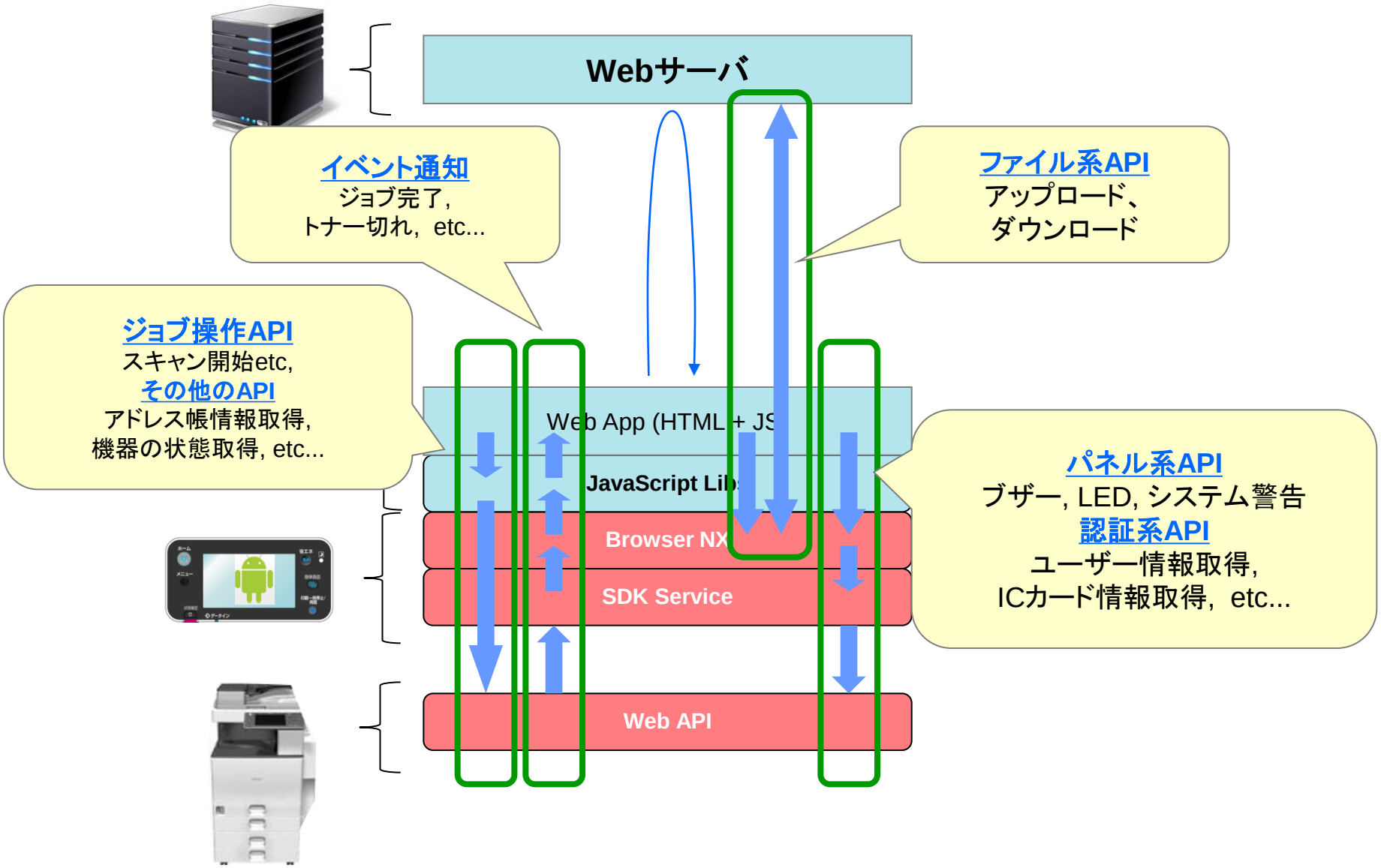
RICOH
imagine. change.

Web Application
基礎編

～MFPの機能を使うには？～

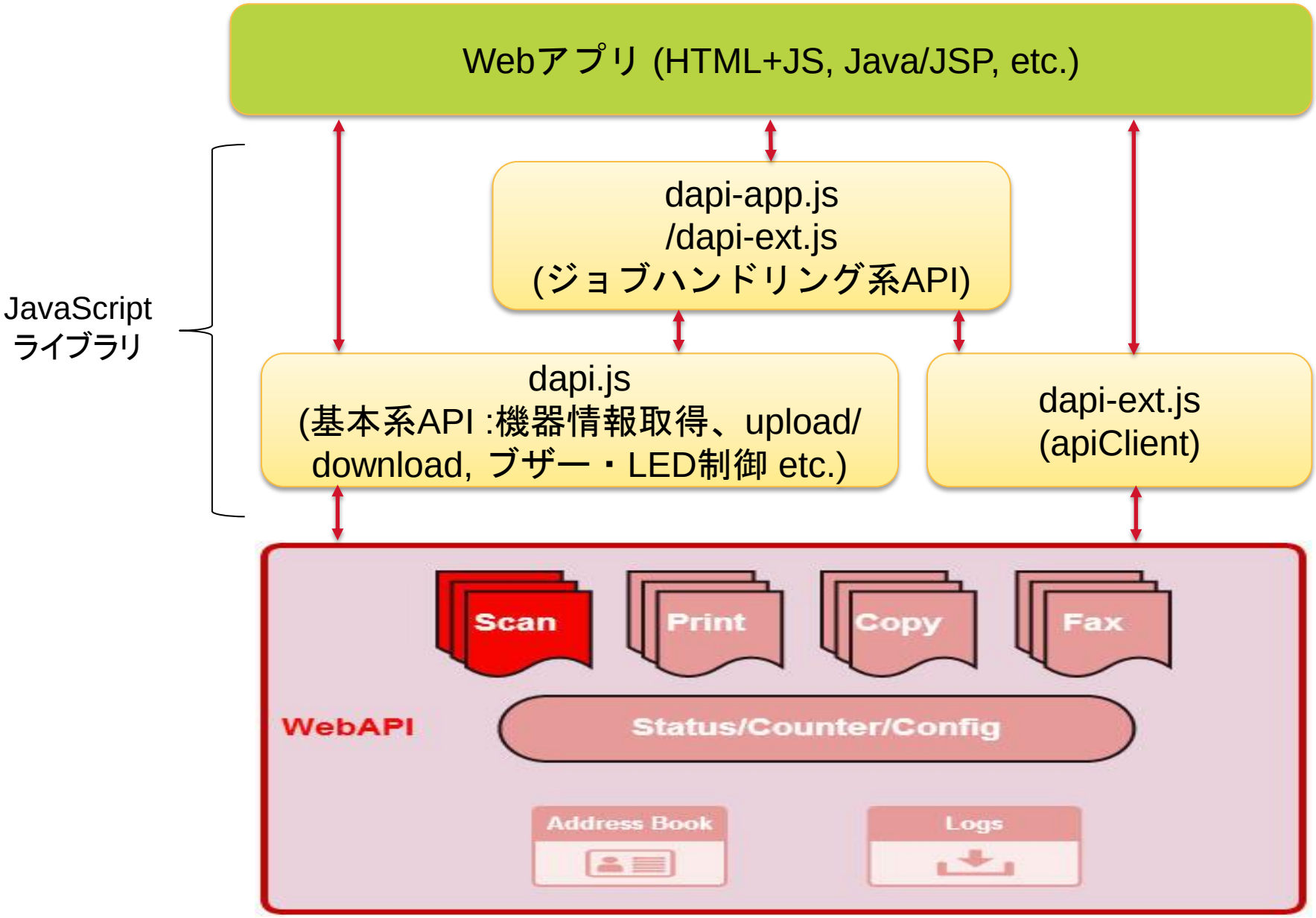
アーキテクチャ

RICOH
imagine. change.



操作部ブラウザーNX用JavaScript Library

RICOH
imagine. change.



JavaScriptライブラリの使い方

RICOH
imagine. change.

```
<html>
  <head>
    <title>サンプルページ</title>
  </head>
  <body>
    ...
   <script src="./js/ricoh.dapi.ext.js"></script>
    ...
  </body>
</html>
```

jsファイルのパス

HTMLファイル内にインポート文を書くだけ

操作部のブザーを鳴らしてみよう

RICOH
imagine. change.

```
<html>

<header><title>Buzzer sample</title></header>
<body onload="createBuzzerSound()">

<script src="./js/ricoh.dapi.js"></script>

<script type="text/javascript">
  function createBuzzerSound(){
    ricoh.dapi.buzzer(2);
  }
</script>

</body>
</html>
```

Javascriptライブラリ
インポート

JavaScript APIを
コール

操作部ブラウザーNX用JavaScriptの種類

RICOH
imagine. change.

- dapi.js
  - 機器情報
    - 機器番号, アドレス帳, etc.
  - 操作部操作
    - ブザー, ハードキー, 省エネモードコントロール,etc.
  - ファイル操作
    - ダウンロード、アップロード、消去
  - 認証系
    - ICカード
  - Job系
    - Scan/Print/Fax/Copy
- dapi-ext.js
  - Ajaxコールのラッパー
    - 任意のWebAPI呼び出し
  - Jobフレームワーク
    - Scan/Print/Copy/Fax
- dapi-app.js
  - 簡易Job操作フレームワーク
    - Scan/Print

操作部ブラウザーNX用JavaScriptの種類

- dapi.js
  - 機器情報
    - 機器番号, アドレス帳, etc.
  - 操作部操作
    - ブザー, ハードキー, 省エネモードコントロール,etc.
  - ファイル操作
    - ダウンロード、アップロード、消去
  - 認証系
    - ICカード
  - Job系
    - Scan/Print/Fax/Copy
- dapi-ext.js
  - Ajaxコールのラッパー
    - 任意のWebAPI呼び出し
  - Jobフレームワーク
    - Scan/Print/Copy/Fax
- dapi-app.js
  - 簡易Job操作フレームワーク
    - Scan/Print

RICOH
imagine. change.

Web Application
実践編

～サンプルコードの紹介～

サンプルコード紹介

- ricoh.dapi.js
- ricoh.dapi.ext.js
- ricoh.app.js

サンプルコード紹介

- ricoh.dapi.js
- ricoh.dapi.ext.js
- ricoh.app.js

ricoh.dapi.js

RICOH
imagine. change.

- 基本的な様々な機能を提供しています
  - ricoh.dapi
    - WebAPIを利用するのに必要な基本的な関数を提供
  - ricoh.dapi.net
    - ファイルのダウンロード、アップロードの機能を提供する
  - ricoh.dapi.auth
    - 認証に関わる機能を提供
  - ricoh.dapi.navigator
    - navigator属性の代替機能を提供
  - ricoh.dapi.setting
    - 設定に関わる機能を提供（Cookieなど）
  - ricoh.dapi.event
    - WebAPIからのイベント受信に必要な機能を提供する

API仕様書の見方

doc/ja/jsapi/100-03-010.htm

ricoh.dapi.js

RICOH
imagine. change.

- 基本的な様々な機能を提供しています
  - ricoh.dapi
    - WebAPIを利用するのに必要な基本的な関数を提供
  - ricoh.dapi.net
    - ファイルのダウンロード、アップロードの機能を提供する
  - ricoh.dapi.auth
    - 認証に関わる機能を提供
  - ricoh.dapi.navigator
    - navigator属性の代替機能を提供
  - ricoh.dapi.setting
    - 設定に関わる機能を提供（Cookieなど）
  - ricoh.dapi.event
    - WebAPIからのイベント受信に必要な機能を提供する

ricoh.dapi.net

RICOH
imagine. change.

関数

- upload
  指定したURLにスキャンした原稿をアップロードする
- download
  指定したURLから印刷用のファイルをダウンロードする
- abort
  アップロード/ダウンロードをキャンセルする
- removeFile
  ダウンロードしたファイルを削除

イベント通知

- onUploadStateChange / onDownloadStateChange :
  アップロード・ダウンロードの状態変化を通知する
  例）接続中、アップロード中、完了、進捗, etc...

ricoh.dapi.net

RICOH
imagine. change.

関数

- upload
  指定したURLにスキャンした原稿をアップロードする
- download
  指定したURLから印刷用のファイルをダウンロードする
- abort
  アップロード/ダウンロードをキャンセルする
- removeFile
  ダウンロードしたファイルを削除

イベント通知

- onUploadStateChange / onDownloadStateChange :
  アップロード・ダウンロードの状態変化を通知する
  例）接続中、アップロード中、完了、進捗, etc...

ricoh.dapi.net – サンプル

スキャンしたファイルをアップロードする

リクエストIDを設定。イベント通知時の識別用

```
reqId = ( new Date() ).getTime();
reqUrl = "http://example.com/upload";
uploadOptions = {
    “params": {
        "fileName": “file.pdf"
    }
}
ricoh.dapi.net.upload( reqId, reqUrl, filepath, uploadOptions );
```

アップロード先URLを設定

ファイル名などを設定

アップロード対象のファイルパス

ricoh.dapi.net – サンプル

アップロードの状況を知る

```
ricoh.dapi.net.onUploadStateChange = function( requestId, state,
            result, status, error, progress, responseBody) {
  if ( requested === reqId ) {
    if ( state === “success” ) {
      // アップロード成功時の処理
    } else if ( state === “failure”) {
      // アップロード失敗時の処理
    } else {
      //  その他の状態
    }
  }
};
```

リクエストIDで識別

サンプルコード紹介

- ricoh.dapi.js
- ricoh.dapi.ext.js
- ricoh.app.js

ricoh.dapi.ext.js

- apiClient機能と、スキャナ・プリンタ・ファクス・コピーを操作する機能を提供します。

  - ricoh.dapi.apiClient
  - ricoh.dapi.scanner
  - ricoh.dapi.printer
  - ricoh.dapi.fax
  - ricoh.dapi.copy

  各アプリ状態の管理、
  ジョブを生成するためのAPI

  - ricoh.dapi.scannerJob
  - ricoh.dapi.printerJob
  - ricoh.dapi.faxJob
  - ricoh.dapi.copyJob

  ジョブに関するAPI

ricoh.dapi.ext.js

- apiClient機能と、スキャナ・プリンタ・ファクス・コピーを操作する機能を提供します。
  - ricoh.dapi.apiClient
  - ricoh.dapi.scanner
  - ricoh.dapi.printer
  - ricoh.dapi.fax
  - ricoh.dapi.copy

  各アプリ状態の管理、ジョブを生成するためのAPI
  - ricoh.dapi.scannerJob
  - ricoh.dapi.printerJob
  - ricoh.dapi.faxJob
  - ricoh.dapi.copyJob

  ジョブに関するAPI

apiClientを使ってWebAPIをコール

- apiClientを使って任意のWebAPIをコールできます

例：アドレス帳のデーター覧を取得

```
var contactOptions = {
    "path": "/addressbook/entries",
    "method": "get",
    "ssl": true,
    "params": params,
    "async": false
};

var contactList = ricoh.dapi.apiClient.request( contactOptions );
```

WebAPIをコールするためのオプションをセット

apiClientを使ってWebAPIをコール

ricoh.dapi.ext.js

- apiClient機能と、スキャナ・プリンタ・ファクス・コピーを操作するための機能を提供します。
  - ricoh.dapi.apiClient
  - ricoh.dapi.scanner
  - ricoh.dapi.printer
  - ricoh.dapi.fax
  - ricoh.dapi.copy

  各サービス状態の管理、ジョブを生成するためのAPI

  - ricoh.dapi.scannerJob
  - ricoh.dapi.printerJob
  - ricoh.dapi.faxJob
  - ricoh.dapi.copyJob

  ジョブに関するAPI

サービスとジョブって？

RICOH
imagine. change.

- 例えば、スキャナでいうと・・・

読み取り開始した

2枚目を読み取った

読み取りがキャンセルされた

紙詰まりが発生

メンテナンス中

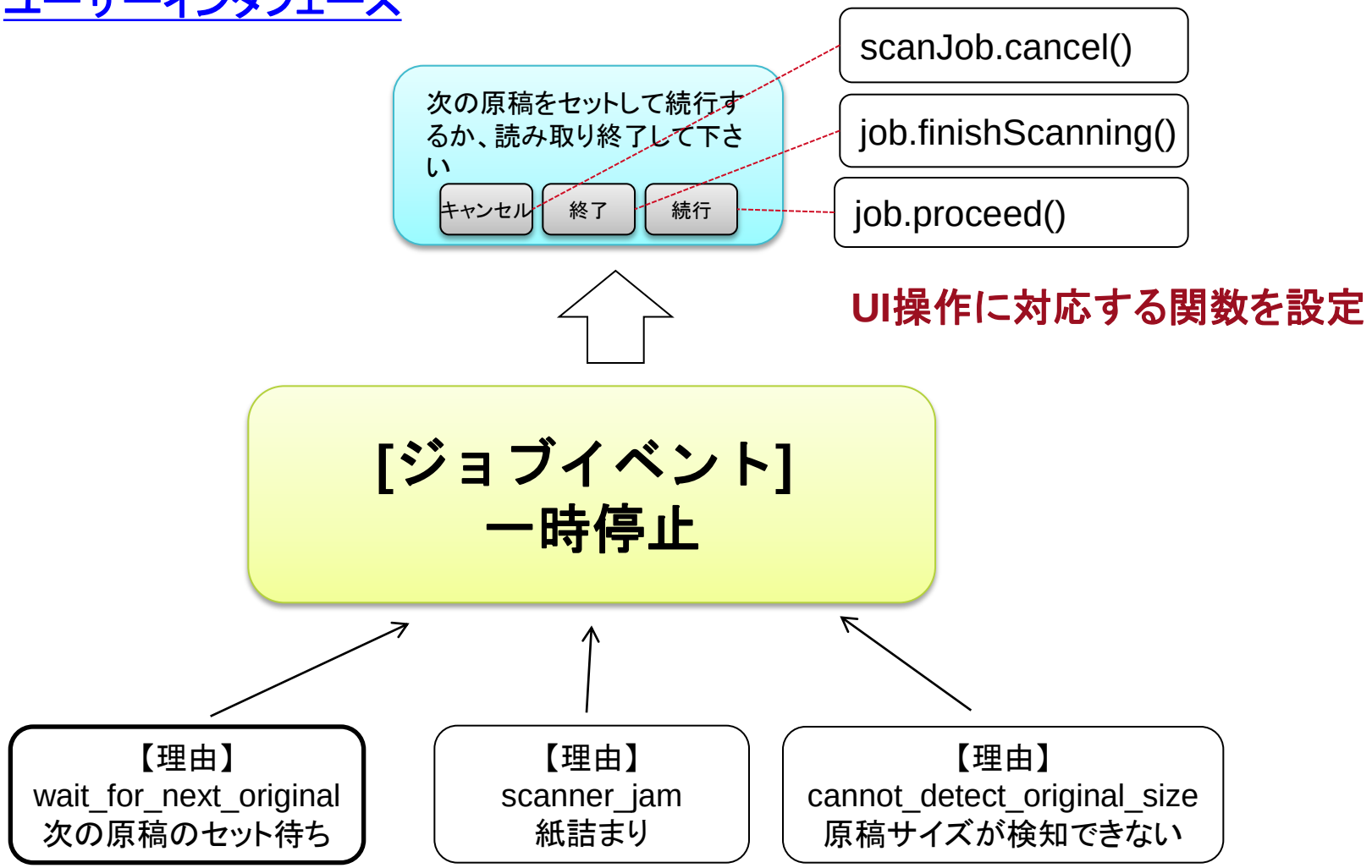
ビジー状態

スキャンジョブイベント・スキャンジョブ状態：
ある一連の読み取り作業に対するイベント・状態
⇒他のジョブには関係しない

スキャナ(サービス)イベント・スキャナ(サービス)状態：
機器（のスキャナ機能）に対するイベントや状態
⇒全てのジョブに関連する

ジョブイベントのハンドリング

RICOH
imagine. change.



スキャン開始までの手順

RICOH
imagine. change.

1. スキャンジョブのオブジェクトを生成

```
var scanJob = ricoh.dapi.scanner.createJob();
```

2. ジョブイベントハンドラをセット

```
scanJob.onCompleted = function() {
        alert( “ジョブが正常に完了しました." );
}
```

ジョブイベントの種類は
・Pending(開始前)
・Processing(ジョブ中)
・Stopped(一時停止)
・Completed(終了)
・Canceled(中止)
・Aborted(システムにより中止)
があります

3. スキャン設定をセットする

```
var scanOptions = {
        "jobSetting" : {
          "jobMode" : "scan_and_send",
          "fileSetting" : { "fileFormat" : "pdf" },
          "destinationSetting" : { ... }
}
```

4. ジョブ開始

```
scanJob.start( scanOptions, function( error ) {
      if ( error ) { alert( "failed to create a job. error: " + error.errors[0].message_id );  }
    });
```

サンプルコード紹介

- ricoh.dapi.js
- ricoh.dapi.ext.js
- ricoh.app.js

ricoh.dapi.app.js

- ジョブ実行後のイベントハンドリングやUIの構築を簡単に使えるようにしたライブラリです。
  - ricoh.dapi.app.scanner
  - ricoh.dapi.app.printer

例えば･･･
- 原稿ガラスでのスキャン実行時の次原稿操作
- サーバーへの画像ファイルのアップロード
- ユーザーがジョブをキャンセルしたときの後処理　etc.

MFPの知識が少なくても
簡単に機能を利用できるようになっています

アプリ構築例

- 読み取り時の動作仕様を考える

スキャン中にイベント発生

一時停止

ダイアログ表示

理由を
表示する

それぞれの操作が可能な
ときだけボタンを表示する

紙詰まりです。

キャンセル　終了　再開

実装例

RICOH
imagine. change.

一時停止ジョブイベント時に呼ばれる

理由を表すローカライズ済み
メッセージが入る

```
ricoh.dapi.app.scanner.onStopped = function ( msg, details ) {
    var buttons = [];
    if ( details.cancel ) {
        buttons.push( { text: “Cancel”, click: details.callbacks.cancel } );
    }
    if ( details.finish ) {
        buttons.push( { text: “Finish”, click: details.callbacks.finish } );
    }
    if ( details.proceed ) {
        buttons.push( { text: “Continue”, click: details.callbacks.proceed } );
    }

    $( “#dialog” ).text( msg ).dialog( { modal: true, buttons: buttons } );
}
```

キャンセル可能？

キャンセル処理や後処理も
提供しています

終了可能？

再開可能？

ダイアログ表示

ハンドラ登録

RICOH
imagine. change.

- 一時停止以外のイベント時にもハンドラを指定できます

(例）

名称	概要
onReady	サービスが利用可能状態になったときのイベントハンドラ
onUnready	サービスが利用不可状態になったときのイベントハンドラ
onProcessing	ジョブが実行状態になったときのイベントハンドラ
onStopped	ジョブが停止状態になったときのイベントハンドラ
onCompleted	ジョブが正常終了したときのイベントハンドラ
onAborted	エラーやユーザ操作によりジョブが中断したときのイベントハンドラ
onAlert	エラー時のイベントハンドラ

アプリ構築例

RICOH
imagine. change.

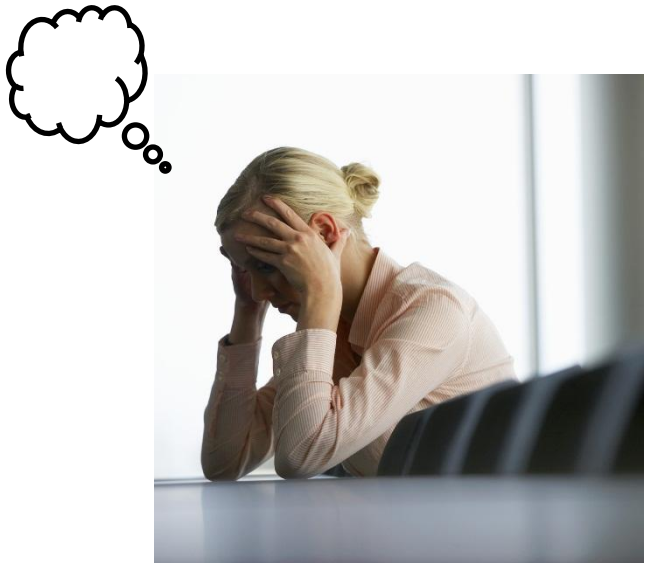
■ 読み取りの仕様を考える

- 読み取ったあとどうする？
  ⇒サーバーにアップロードする

読み取り設定

- ADF？原稿ガラス？
  ⇒どっちでもいいので自動で
- ファイル形式？
  ⇒PDF
- 片面？両面？
  ⇒両面左右開き
- その他の読み取り設定は？
  ⇒デフォルトで

MUSTで決めるのはこれだけ。
・読み取りして送信
・読み取りしてドキュメントボックス蓄積
・読み取りして一時蓄積(転送)
・蓄積文書の送信



実装例

RICOH
imagine. change.

```
$( function() {
  $( "#scan" ).button( { disabled: true } ).click( function() {

    var scanOptions = {
      "jobSetting": {
        "scanDevice":"auto",
        "fileSetting": { "fileFormat":"pdf" },
        “originalSide”:”top_to_top",
        "jobMode": "scan_and_store_temporary",
      }};

    var uploadOptions = {
      "url": “http://www.test.ricoh.co.jp/upload”
    };

    ricoh.dapi.app.scanner.start( scanOptions, uploadOptions );
  });
```

「#scan」のボタンが押された時の処理

読み取り設定を指定。
指定しなかったものはデフォルトになる。
jobModeの指定はMUST

このように指定すればサーバーへの
アップロードもやってくれる

読み取り開始！

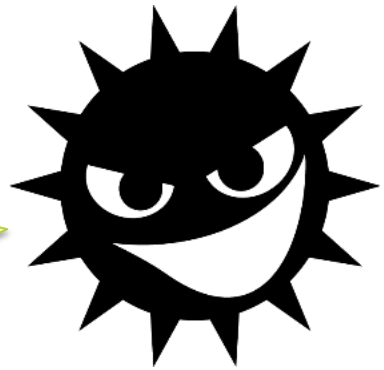
コードを動かす前に・・・

RICOH
imagine. change.

- 以上、今まで紹介しましたコードを実装・・・しただけでは実機では**動きません**。（エラーになります）

- 操作部ブラウザーNX用JavaScriptを使用する前に、必ず**正当性検証**の処理が必要です。

ただし、エミュレータでは
正当性検証しなくても動くので、
うっかり忘れないように注意！



正当性検証　方法①

RICOH
imagine. change.

正当性検証の処理は、操作部ブラウザーNX専用JavaScritを使う前、かつページ表示後1分以内に実施してください。

```
<html>
<head>
  <title>(指定したタイトル)</title>
</head>
<body onload="validate()">
  <script src="./js/ricoh.dapi.js"></script>
  <script>
    function validate() {
      var token = "(指定したトークン)";
      ricoh.dapi.validateAccessToken(token, init());
    }
  </script>
</body>
</html>
```

トークンはタイトルと結びついているので、指定したタイトルを使ってください

本当はトークンはこの例のようにべた書きしてはいけません

正当性検証が成功したら実行される処理

正当性検証　方法②

RICOH
imagine. change.

Scan/Print/Copy/Faxの各サービス系イベントの監視を開始する時にもトークンが必要。成功すれば正当性検証も完了します

```
<body onload="initScanner()">
  <script src="./js/ricoh.dapi.ext.js"></script>
  <script>
    function initScanner() {
      var token = "(指定したトークン)";
      ricoh.dapi.scanner.onStatusChange = function() {
        // サービス状態変化時の処理
      };
      ricoh.dapi.scanner.onInitResult = function(result) {
        // 初期化失敗の場合はリトライ
      };
      ricoh.dapi.scanner.init(token);
    }
  </script>
</body>
```

初期化の結果をチェック

スキャナサービスイベントの監視を開始

注意点

- 今回Webアプリケーションを選択する方は、実機から接続するためのサーバーの構築は各自でお願いいたします。

RICOH
imagine. change.

ご静聴ありがとうございました。

RICOH

imagine. change.