RIA（Rich Internet Application）の現状と考察

京都情報大学院大学 准教授 田渕 篤

◆ 1. はじめに

2011 年秋，惜しまれつつこの世を去った Apple 社 CEO の Steve Jobs 氏。彼は，1985 年に一度 Apple 社を追われた後，再起を期して NeXT 社を創業した（10 年後に Apple 社に買収される）が，そこでの彼の仕事が，いわゆる WWW（World Wide Web）の黎明に貢献していることは，あまり知られていない事実かもしれない。WWW の提唱者である Tim Berners-Lee は，1990 年末に世界初の Web サーバと Web ブラウザとを試作し稼働させたが，その開発環境となったのが NeXT 社製のワークステーション（NeXTcube）だったのである [1]。

Tim Berners-Lee が WWW を構想した背景には，彼が当時所属していた欧州原子核研究機構（CERN）において，研究者たちが膨大な数の技術文書を相互にリンクし，効率よく閲覧する『分散文書管理システム』への強い要求があったという。それから 20 余年を経て，人々が Web の世界に求めるものは大きく様変わりした。その最も大きな変化の一つが「文書から RIA（Rich Internet Application）へ」という流れである。

すなわち Web の利用者は，静的な文書データをリンクをたどって順次閲覧するだけに収まらず，より対話的で動的な『アプリケーション』として，Web 上のコンテンツを操作することに価値を見いだしているのである。その変化に伴い，Web ブラウザも，文書閲覧のツールから Web アプリケーションを実行する基盤ソフトウェアへと進化していった。また，汎用の Web ブラウザだけでなく，用途ごとに特化した単体 RIA の開発／実行基盤も広がりを見せている。

本稿では，まず RIA が持つべき特色について概観する。次いでその開発／実行環境として，本命の HTML5 + JavaScript，および対抗馬となる Adobe Flex，JavaFX のそれぞれの優位点／難点について考察していく。

◆ 2．RIA（Rich Internet Application）とは

Rich Internet Application という用語が公に用いられたのは，2002 年の Macromedia 社（後に Adobe 社が買収）が提出した技術レポート [2] が最初と言われている。RIA の登場以前，Web の利用者は，Web サーバが画面 1 ページ分の HTML データを生成・送信し，Web ブラウザがそれをレイアウトして表示する，という単純なサーバ／クライアントモデルのもとで，コンテンツを閲覧していた。しかし，エンドユーザ向け PC の性能が上がるにつれ，クライアント側アプリケーションに対して，より対話性や表現力の高いユーザインタフェース --- 例えば Web コンテンツの動的な書き換えや更新時の視覚効果など --- が求められ，また実装できるようになってきた。このことが，今日の RIA 技術の進展に繋がっている [3]。

先の Macromedia 社の技術レポートでは，RIA が持つべき特色として，具体的に以下のようなものを挙げている。

- ●プログラム実行・データ通信・コンテンツ表示を効率よく行う実行環境
- ●文字・音声・動画などの各種メディア，及びそれらの送受信の統合
- ●強力で拡張性の高いデータオブジェクトモデル
- ●再利用可能なコンポーネントによる容易な開発
- ● HTML 以外（XML 等）のデータサービスとの連携
- ●オンライン／オフラインを問わない動作
- ●マルチプラットフォーム／マルチデバイスへの対応

これらを実現するアプリケーション基盤は，以下の 3 つの方式に大別される [4]。

1. Web ブラウザそのものの機能を拡張して，HTML で表現された Web コンテンツに JavaScript や CSS による対話性や表現力を持ち込む方法。Ajax 技術によるコンテンツの部分更新等もその例である。Web ブラウザは，元々マルチプラットフォーム対応かつ無償で提供されるため，既に幅広いユーザに浸透している，という利点がある。
2. Web ブラウザに各 RIA 専用のプラグインを組込み，Web ページ内に埋め込む形（<object> タグ等）で RIA としてのユーザインタフェースを提供する方法。Adobe Flash や Java Applet 等がこれに当たる。Web ブラウザの表現力の制約を受けない柔軟なユーザインタフェースが提供でき，ユーザも通常の Web ページへのアクセスと同じ感覚で利用できる，という利点がある。
3. 用途別の単体デスクトップアプリケーションとして RIA を提供する方法。Adobe AIR，JavaFX 等がこれに当たる。RIA の実行プログラムやコンテンツは，アプリケーションサーバを介して個別にユーザに配布される。動作させるのに Web ブラウザが不要という利点がある

現時点では，方式 1 に当たる HTML5（+ JavaScript）による RIA が，普及度や開発コスト等の点で本命と言える。しかし，Adobe 社（Flash / AIR）や Oracle 社（Java）といった有力企業も，方式 2 または 3 による RIA の開発／実行環境を自社製品として今後も提供していくことを表明しており，これらを対抗馬とする覇権争いは当面続くと予想される [5]。

◆ 3．本命：HTML5（+ JavaScript）

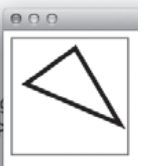
HTML5 は，Web ページ記述用言語である HTML の第 5 版 [6] であり，現在 Web 技術の標準化団体である W3C において，2014 年の正式勧告を目指して仕様の策定が進められている。

HTML5 では，マークアップ用のタグ仕様の改訂に加え，Web コンテンツを各種プログラムから操作するための新たな API 仕様を多く含んでいる。即ち，RIA の構築に必要なオブジェクトやメソッド等を Web ページ内で直接記述できるようにして，RIA を Web ブラウザのネイティブアプリケーションとして動作させようという方向性が見て取れる。

近年の Web ブラウザでは，表示中の HTML 要素を動的に操作するために，DOM（Document Object Model）と呼ばれるオブジェクト指向の API（この仕様も W3C が標準化している）が実装されている [7]。HTML5 の新しい API 仕様では，この DOM API を拡張する形で提供されるものと，DOM 要素ではなくクライアント側 OS の機能（ファイルシステムやネットワーク等）との橋渡しを担うものとが新たに加わっている。

例えば，HTML5 で導入される canvas タグ（図形の描画領域）は，それ自体で何らかの図形を表すものではなく，canvas の DOM 要素から取得した描画コンテキストに対して，描線の指定や塗りつぶし等を指示するメソッドを動的に呼び出して描画を行なう（図表 1）。同様に video（動画）や audio（音声）のタグで挿入されたメディアの再生／停止等も，DOM 要素へのメソッド呼び出しを介して行なう。一方，ブラウザ側で連想配列形式の任意データを保存／取得する API である localStorage オブジェクトは，対応する HTML 要素は無く，ローカルファイルシステムとのインタフェースを提供する存在である。

```
<!DOCTYPE html>
<html lang="ja">
<head>
  <meta charset="utf-8"/>
  <script type="text/javascript">
  onload = function(){ // 三角形を描く
    var ctx = document.getElementById("c")
        .getContext("2d");
    ctx.lineWidth = 5;
    ctx.beginPath();    ctx.moveTo(70, 10);
    ctx.lineTo(120,95); ctx.lineTo(15,50);
    ctx.closePath();    ctx.stroke();
  };
  </script>
</head>
<body>
  <canvas id="c" width="128" hei
  style="border:2px solid gray"/>
</body>
</html>
```



図表 1 canvas API を用いた図形描画の例（田渕が作成）

これらの API は，特定のプログラム言語に依存する仕様ではないが，Web ブラウザ上では JavaScript のプログラムから実行するのが現実的である。JavaScript は，2000 年頃まではブラウザ間の互換性に問題が多かったが，その後標準化が進み，特に Ajax（非同期通信による DOM 要素の動的更新）の技術が注目されたことで，RIA の記述言語として定着していった。

また近年は，DOM API の上位に当たる様々な JavaScript ライブラリがオープンソースで提供されており，RIA の開発効率の向上に貢献している。中でも jQuery[8] は，

- ●基本的な DOM 操作からイベント処理，Ajax，視覚効果まで，RIA に必要な機能を幅広くカバーする。更にプラグイン方式での機能拡張も容易である。
- ●操作対象とする DOM 要素の集合を CSS のセレクタ式で選別できるため，従来の Web 技術との親和性が高い。
- ●ライブラリのサイズ自体が小さく（約 30KB），ブラウザへの負荷が少ない。

などの特徴があり，開発者の人気を集めている。

◆ 4．対抗馬 (1)：Adobe Flex（Flash / AIR）

Adobe Flex[9] は，Adobe（前 Macromedia）Flash から派生した RIA 開発基盤である。Flash は，対話的なアニメーションやゲームを提供するマルチメディアツールとして広く認知されている。Flex は，その対話的なユーザインタフェースの構成要素やバックエンドの通信部分等を ActionScript 言語のクラスライブラリとして再構築したものと位置づけられる。

Flex からは，従来の Web ページ埋め込み型の Flash アプリケーションだけでなく，Adobe AIR[10] と呼ばれる単体のデスクトップアプリケーションも作成することができる。即ち，共通の開発基盤から，ターゲットに応じて異なるタイプの RIA を作り分けられるという特徴がある。

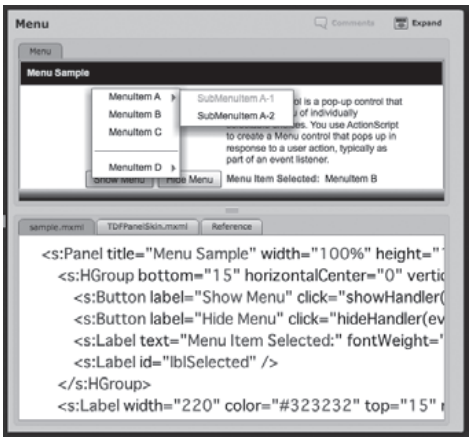
●実行環境：Flex ベースのアプリケーションは，Flash / AIR とも SWF 形式のバイナリファイルにコンパイルされ，クライアント PC で実行される。

Flash アプリケーションは，<object> タグ等で Web ページ内に埋め込まれ，Web ブラウザのプラグイン（Flash Player）を介して動作する。一方，AIR アプリケーションは，インストーラ形式で利用者に配布され，単体で動作する（即ち実行時に Web ブラウザは不要）。ただし，AIR 専用の実行時ライブラリ（AIR Runtime）を事前にインストールしておく必要がある。

いずれも実行時に PC のファイルシステムやデバイスを利用したり，任意のサーバと通信したりすることが可能である。

●開発環境：Flex の ActionScript クラスライブラリやコンパイラ等（SDK）は，オープンソース化されており無償で入手できる。SDK の最新版は ver.4.6 である。

Flash / AIR とも，ActionScript 言語のみでアプリケーションを作成できるが，画面設計（部品配置，視覚効果，画面遷移など）は専用の XML タグ（MXML）や CSS で宣言的に定義し，イベント処理の手続き部分のみを ActionScript で記述，という作成方法も可能で，こちらの方が開発効率が良い（図表 2）。



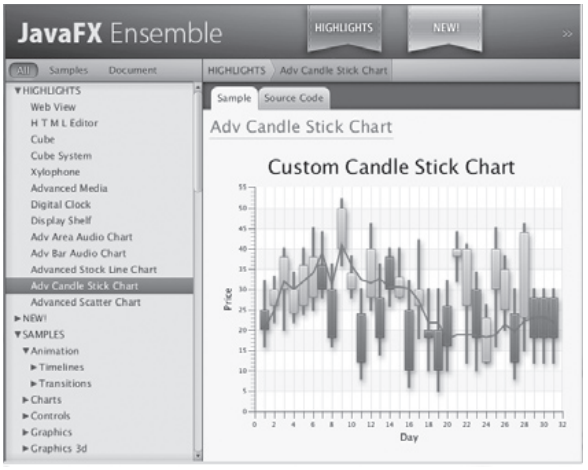
図表 2 Adobe Flex の MXML と ActionScript による GUI 定義の例（Adobe 提供）

また，Eclipse ベースの統合開発環境である Flash Builder[11] も提供されており，Flash または AIR どちらのタイプのアプリケーション開発にも使える。対話的な画面エディタとデバッガ，他のグラフィックツール（Adobe Photoshop 等）との連携，プロジェクト管理などの機能が提供されている。有償ツールであるが，学生・教職員は無償で使用できる。

◆ 5．対抗馬 (2)：JavaFX

JavaFX[12] は，Sun Microsystems 社（ 現 Oracle 社 ） が 2007 年頃から推進している Java の RIA 構築用クラスライブラリである。Java は，Web ブラウザ内に埋め込む GUI アプリケーションである Applet を古くから提供しており，その意味では RIA プラットフォームの先駆と言える。

Applet と比較すると，JavaFX は，RIA で特に重要な視覚表現の実装が強化され，GPU アクセラレーションを活用するグラフィックエンジン（Prism）や，アニメーション・エフェクト等を制御するクラスが追加されている（図表 3）。また画面設計やイベント処理も手軽に記述できるように進化している。



図表 3：JavaFX の XYChart クラスによるグラフ表示の例（Oracle 提供）

●実行環境：JavaFX は，通常の Java アプリケーションと同じバイトコードにコンパイルされ，専用の実行時ライブラリ上で動作する。ライブラリの最新版は Ver.2.0 で，現在 Windows 版のみが提供されている（SDK も同じ）。2013 年リリース予定の Java SE 8 では，他の OS の Java 実行環境（JRE）にも同包される予定である。

アプリケーションの配布は，Applet と同じく Web ブラウザに埋め込む場合と，ダウンロードして単体のデスクトップアプリケーションとしてインストールする場合とに対応している。後者は，Java Web Start（JWS）と呼ばれる独自の配布方式 [13] によって，バージョンや付随ライブラリなどの管理も自動的に行なうのが一般的である。

いずれの場合も，Adobe Flex と同じく，任意のサーバとの通信，ローカルファイルシステムや組込みデータベースの利用などが可能である。

●開発環境：JavaFX を直接サポートする統合開発環境は，現在 Oracle が提供する NetBeans IDE 7.1 のみであるが，純粋に Java プログラムとして作成するなら，Eclipse など他の Java 向け開発ツールも用いることができる。

JavaFX の当初の大きな特徴として，JavaFX Script と呼ばれる独自のスクリプト言語で画面設計の大半を宣言的に記述できる，という点があった。しかし，最新版では事実上撤回され，Adobe Flex / AIR と同様の XML タグ（FXML）と CSS を併用する画面設計のアプローチが採用されている [14]。

イベント処理については，通常の Java 言語でのメソッド記述／実行だけでなく，JavaScript や Ruby など Java 仮想マシン上で解釈できるスクリプト言語でも記述できるよう API が拡張されている。

◆ 6．考察

以上，３つの RIA 構築基盤の現状を概観してきたが，それらが開発側（プログラマ／デザイナ）と利用者側とでどのような優位点／難点を持っているかを考察する。

◆ 6.1　プログラマの立場から：

まず，開発ツールを揃える金銭的コストについては，有償の Flash Builder を事実上使わざるを得ない Adobe Flex が不利であろう。また，学習コストの点でも，Adobe Flex の ActionScript は，開発経験や既存コードの蓄積が豊富な Java や JavaScript に比べて若干不利であろう。ただ，ActionScript が Ver.3.0 で完全なオブジェクト指向プログラミング（OOP）に対応したことで，今後 Java などで OOP の経験を持つプログラマを取り込める可能性はある。

開発効率，特にデバッグにおいては，コンパイラで厳密なデータ型チェックのできる JavaFX や Adobe Flex が優位で，インタプリタ型言語の JavaScript は必然的に不利となる。

クロスプラットフォーム対応という点では，現時点では，スマートフォンを含む主要な OS・デバイスに対応している Adobe Flex が最も優位である（ただし Adobe は，スマートフォン向け Flash Player の開発中止を表明しており，今後はスマートフォン向けには AIR のみが選択肢となる [15]）。HTML5 は，W3C が正式な仕様勧告を出す 2014 年までは，各 Web ブラウザベンダが個別に実装を進めており，ブラウザ間で互換性の

あるコードかどうか悩む状況が当面続くであろう。JavaFX も，開発／実行ライブラリ自体が Windows 以外の OS やモバイル環境への対応で出遅れている。

◆ 6.2 デザイナの立場から：

RIA では GUI の視覚デザインが重要であり，グラフィックデザイナとプログラマとの共同制作が必要となるケースが多い。その点では，デザイナ向けの使い勝手の良いツールを提供あるいは連携できていない JavaFX はかなり不利である。

Adobe Flex を Flash Builder 上で利用する場合，同じ Adobe 社のグラフィックツールの成果物をデザイン要素としてスムーズに取り込めるため，デザイナの支持を得やすい。HTML5 は，もともと Web デザイナと JavaScript プログラマの協業が進んでおり，またデザイナにとっての学習コストが比較的低い jQuery などのライブラリの普及も優位に働いている。

◆ 6.3 ユーザの立場から：

最もユーザの目を引く GUI の美しさや対話性については，どの方式も一定の満足を得られる仕上がりになっており，その点での差別化は難しい。

実行環境の導入コストの点では，Web ブラウザと Flash Player の圧倒的な普及率を考えると，HTML5 と Adobe Flash が有利である。Adobe AIR および JavaFX も実行時ライブラリは無償だが，そのインストーラのダウンロードと実行の手間が避けられない（初回限りではあるが）。

アプリケーションやコンテンツへのアクセスの容易さの点では，Web ブラウザ内で動作する HTML5，Flash，Applet 方式の JavaFX が Web サーバ上の URL へのアクセスだけで利用し始められるのに対し，AIR と JWS 方式の JavaFX は，一旦アプリケーションをダウンロードしてインストールする手間がかかる（AIR の場合はマシンの管理者権限も必要）。

反面，AIR と JWS 方式の JavaFX は，単体アプリケーションだけに，手軽な起動（アイコンのダブルクリック）やメモリの消費量の少なさに利点がある。これは特にハードウェア資源に限りのあるスマートフォンでの動作では有利であろう。

◆ 7．終わりに

今回は，RIA のクライアントアプリケーションとしての開発／実行環境に焦点を当てたが，RIA は本来，データベースや業務ロジックなどを操作するサーバアプリケーションと連携した形で真価を発揮するものである。特に今後，クラウド・コンピューティングが普及するにつれ，より快適なユーザ体験をクライアント側にもたらす RIA の役割は重要になるであろう。開発者・利用者とも，サーバ側とクライアント側との機能分担や，開発効率と利便性とのバランスが最適化されるよう，RIA 構築基盤を選択していく必要がある。今後もその動向を見守っていきたい。

【参考文献（引用順）】

[1]. Tim Berners-Lee, "The WorldWideWeb browser", http://www.w3.org/People/Berners-Lee/WorldWideWeb.html

[2]. Jeremy Allaire,"Macromedia flash MX-A next-generation rich client", http://download.macromedia.com/pub/flash/whitepapers/richclient.pdf, 2002

[3]. 三井英樹, "Rich Internet Application (RIA) の時代", http://itpro.nikkeibp.co.jp/article/COLUMN/20060309/232091/,2006

[4]. 日経NETWORK, "RIAとは", http://itpro.nikkeibp.co.jp/article/Keyword/20071113/287037/, 2007

[5]. 吉田育代,"アドビ, MS,サンの「RIA開発における優位点」とは?", http://techtarget.itmedia.co.jp/tt/news/0904/03/news02.html, 2009

[6]. W3C,"HTML5--A vocabulary and associated APIs for HTML and XHTML", http://www.w3.org/TR/html5/, 2011

[7]. W3C, "Document Object Model (DOM) Technical Reports", http://www.w3.org/DOM/DOMTR, 2004

[8]. The jQuery Project, "jQuery : The Write Less, Do More, JavaScript Library", http://jquery.com/

[9]. Adobe Systems, "無償のオープンソースフレームワーク| Adobe Flex", http://www.adobe.com/jp/products/flex.html

[10]. Adobe Systems, "Adobe AIR 3", http://www.adobe.com/jp/products/air.html

[11]. Adobe Systems, "Adobe Flash Builder 4.6 Premium", http://www.adobe.com/jp/products/flash-builder.html

[12]. Oracle, "JavaFX | Rich Internet Applications Development", http://javafx.com

[13]. Sun Microsystems, "Java Web Start 1.5.0 開発者ガイド", http://www.adobe.com/jp/products/flash-builder.html

[14]. 櫻庭祐一, "JavaFX 2.0 -- Javaによるリッチクライアント基盤", http://www.infoq.com/jp/articles/javafx20_01（前編）, 同/ javafx20_01（後編）, 2011

[15]. Mike Chambers, "モバイルブラウザー向けFlash Player, Flash Platform, Flashの今後について", http://www.adobe.com/jp/aboutadobe/pressroom/pressreleases/20111117_adobe_flash.html, 2011

田渕　篤
Tabuchi　Atsushi
京都大学・工学修士，元 NEC 研究員