# HTML5/RIA 技術の業務システムへの応用と今後の展望

## HTML5/RIA Technology for Business Systems and View for Future

川　田　清　忠

**要　約**　HTML5 は，多様な先進技術を統合した Web アプリケーション・プラットフォームである．デザイン，マルチメディア，オフライン，インタラクション，コミュニケーションの機能強化が図られ，セマンティクスの新たな恩恵を享受することが期待されているが，今後の業務システム開発においては，クロスブラウザ問題の対処等の十分な準備が必要である．

**Abstract**　HTML5 is the web applications platform that integrates a lot of advanced technologies. It has enhanced the design, multimedia, offline, interaction, and communication, and is expected to enjoy new benefits of the semantic. However, it must be prepared enough for the problem like the cross-browser in the business systems development of future.

### 1. は　じ　め　に

Rich Internet Application（以降，RIA と呼ぶ）というイディオムは，グラフィック・アニメーションの視覚効果や音声・映像を含むマルチメディア等の豊かな表現力と，デスクトップ・アプリケーションの GUI（Graphical User Interface）のような高い操作性と高度な機能を有する Web アプリケーションを意味する．RIA 技術は，例えば，インターネット電子商取引における広告やプロモーション，オンラインゲーム，動画・エンターテイメント等の顧客の興味と関心を惹き付ける為の目的と，株取引等のコンシューマ向けサービスの質を向上する目的で広く普及しており，昨今では，企業内の業務システムでも使われるようになっている．

業務システムにおいて「Rich」とは満足感を意味するバズワード[*1]であり，初期の Web，つまり，文書の閲覧を主体とした HTML（Hyper Text Markup Language）の表現力と操作性における「Poor（貧弱さ）」に対する反意語として扱われる．現在の RIA 機能はプラグインとしてブラウザに組み込まれているが，HTML 以外の標準化されていない多様な先進技術が使われている．この状況に対して Web 関連技術の標準化団体である W3C（World Wide Web Consortium）は，RIA と同等の Web アプリケーション機能の実現を目指した最新 HTML 仕様の検討と標準化を開始した．それは HTML の第 5 版となることから「HTML5（エイチティーエムエルファイブ）」と呼ばれる[1]．

本稿では次世代の Web 標準技術として羨望される HTML5 について，業務システムへの応用という視点で想定される効果とその技術等の特徴について解説し，将来に向けた展望について現時点での考察を述べる．

## 2. Web 関連技術の変遷

本章では，現在（2011 年 10 月時点，以降同様）までの Web 関連技術の変遷を振り返り，RIA 技術および HTML5 の出現に至るまでの背景を再確認する．

### 2.1 ハードウェア・アーキテクチャの変遷

かつて，金融業をはじめとする大手企業はメインフレームと呼ばれる大型汎用コンピュータによるセンター集中処理で，自社の基幹となる業務システムを運営していた．当時のメインフレームの機材は非常に高価であり，OS や仕組みの専門性が高いことから，この時代の業務システムの直接的な利用者は情報システム部門と運用担当部門に限られていた．

次の段階では，オープンな OS である UNIX や Windows を搭載した小中型サーバ，PC 等のハードウェア性能の向上とダウンサイジングに伴い，クライアント/サーバ型の分散処理システムが構築された．機材の省スペース化と低価格化により，大手以外の企業でも各部門・各担当者毎にサーバや PC が導入され，コンピュータ・システムは近代における業務の基盤となった．

そして，WWW（World Wide Web）の普及とネットワークの高速化に従い，インターネット・ビジネスが爆発的に増加していく．MVC（Model View Controller）モデルに基づいた 3 層アーキテクチャは，不特定多数の利用者による急激なアクセス数の増加に対し，ハードウェア構成の拡張性に優れていた．また，Web サービスはブラウザがあれば利用することができ，それまで業務システムが抱えていたクライアント・アプリケーションのプログラム配布やバージョン管理などの管理コスト問題を緩和できた為，多くの業務システムに適用され，現在に至るハードウェア・アーキテクチャの基礎となった．

### 2.2 HTML 周辺技術の変遷

そもそも，HTML は文書を記述するための言語であり，写真や画像ファイルがリンクできる程度の動きの無い静的な表現しかできなかった．しかし，動的なアクションに対する要求は高く，それ故に，HTML 標準以外の様々な技術や仕組みが考案されることになる（図 1）．

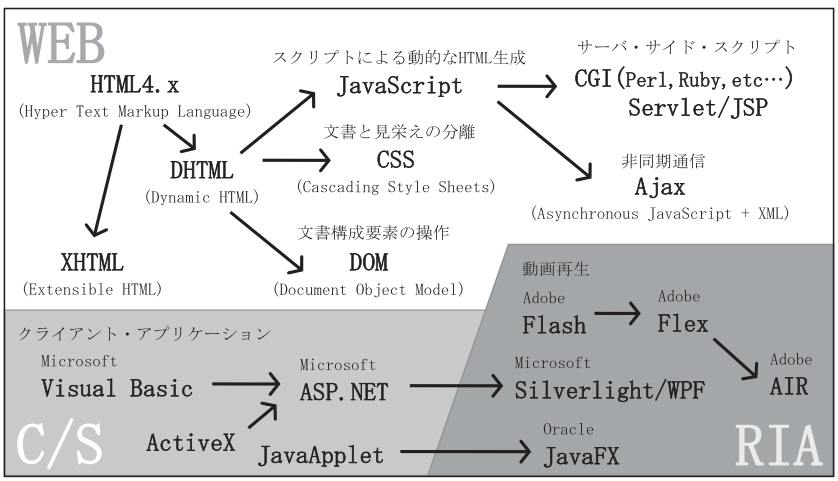



図 1　HTML 周辺技術の変遷

インタラクティブ性を高める為の技術として，クライアント/サーバ型のアプリケーションの流れを汲む ActiveX コントロールや，JavaApplet（アプレット）等のクライアント側で実行されるプログラムが HTML に埋め込まれて配信された．しかし，プログラムの読み込み時間の遅さや，セキュリティ上の脆弱性，バージョン管理等の運用面での負荷が課題とされた．

CGI（Common Gateway Interface）は，サーバ側でスクリプトを実行する為の仕組みであり，HTML 文書を書き換えてスクリプトの実行結果を動的に反映することができた．また，サーバ側で実行されるアプレットとして Servlet（サーブレット）/JSP が発展するが，これらの仕組みはページを更新する際に HTML 文書全体の再読み込みが必要であった為，サーバやネットワーク機器に負荷を与えることが課題とされた．

DHTML（Dynamic HTML）では，HTML の見栄え部分は CSS（Cascading Style Sheet）に分離して記述でき，JavaScript と DOM（Document Object Model）を使って文書構成要素の操作ができるようになった．また，JavaScript での非同期通信（XMLHttpRequest）を基幹技術とし，ページの一部だけを動的に更新することが可能となった Ajax（Asynchronous JavaScript+XML）は，Google マップの登場と共にセンセーショナルな技術として広く知られることになる．

Adobe 社の Flash[*2] は動画の再生技術として広く普及し，その基盤でアプリケーション開発を可能とする為の Flex が代表的な RIA 技術として発展する．また，Microsoft 社の Visual Basic や ActiveX などのデスクトップ技術は，.NET Framework をベースとした ASP.NET で Web サービスに対応し，さらに RIA 技術の Silverlight/WPF へと進化していく．これらは独自の開発言語が使われる為に専門的な技術者を必要とするが，OS やブラウザに依存しないクロスプラットフォームであることが特徴として謳われている[*3]．また，Silverlight や Adobe AIR では，ブラウザの外で個別のプロセスとしてアプリケーションを実行する機能も備わっている．

他にも JavaApplet の課題・問題点を改善した JavaFX のように，現在では様々な RIA 製品が利用可能である[2]．

### 2. 3　Web 技術の標準化

現在使用されている HTML の仕様は，バージョン 4.0 が 1997 年に W3C の勧告となり，続いて 1999 年にバージョン 4.01 がマイナーバージョンアップ版として勧告されているが，それ以降，新しいバージョンの勧告は出されていない．2000 年に勧告された XHTML1.0（Extensible Hyper Text Markup Language）は，HTML4.01 をベースとして策定されたマークアップ言語であり，従来の HTML 文書を，構造化したデータとして解釈できるようにすることを目的として XML（Extensible Markup Language）に準拠した厳密な文法を取り入れている．

2007 年，W3C は XHTML2.0 のワーキング・グループを発足し，また，これと並行して HTML の最新バージョンである HTML5 の仕様検討を開始した．HTML5 のベース仕様として採用された Web Applications 1.0 は，2004 年に Apple，Mozilla Foundation，Opera Software のメンバにより設立された WHATWG（Web Hypertext Application Technology Working Group）[*4] が策定した Web アプリケーションの為のプラットフォーム仕様であり，2011 年 5 月 25 日時点で最終草案（Last Call）が策定され，2014 年の正式勧告（Recommendation）を目標として仕様の検討が継続されている．尚，2009 年，W3C は HTML5 の仕様策定

を重要視し，XHTML2.0 のワーキング・グループを打ち切ることを発表した．今後，XHTML は HTML5 での XML 構文の仕様「XHTML5」として規定される．

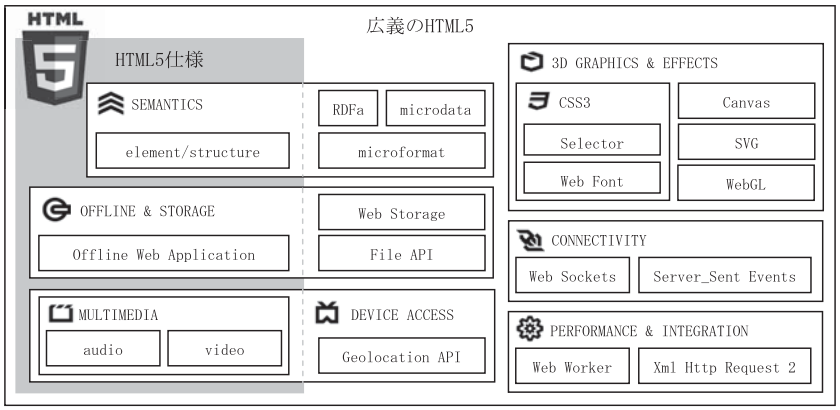
## 3. HTML5 の特徴

本章では HTML5 周辺技術の種類と範囲，および新しい概念や基本ルールについて記述する．

### 3. 1 HTML5 の範囲

HTML5 では，マークアップ要素に明確なセマンティクス（意味付け）を与え，プレゼンテーショナルな要素を廃止して CSS に任せるなどの強化が施される．また，Web アプリケーションに対応する為の canvas 要素や，audio 要素，video 要素，Web Sockets や Web Storage，Web Worker 等のエキサイティングな JavaScript API が新たに追加される．

W3C が HTML5 技術の普及推進を目的として作成した HTML5 LOGO[*5] では，八つのクラスで HTML5 技術を分かりやすく表現しており，それぞれ，SEMANTICS（文書構造，意味付け），OFFLINE & STORAGE（オフライン処理），DEVICE ACCESS（GPS，カメラ，マイク等の周辺機器へのアクセス），CONNECTIVITY（リアルタイム通信，高速通信），MULTIMEDIA（オーディオ，ビデオ），3D GRAPHICS & EFFECTS（二次元・三次元図形，視覚効果），PERFORMANCE & INTEGRATION（アプリケーション性能と統合），CSS3（スタイルシート）に分類している．これらのクラスをもとに，HTML5 仕様と代表的な関連技術を含めた広義の HTML5 の範囲を示したのが図 2 である．




W3C HTML5 LOGOのTechnology Classをもとに作成

**図 2 HTML5 の範囲**

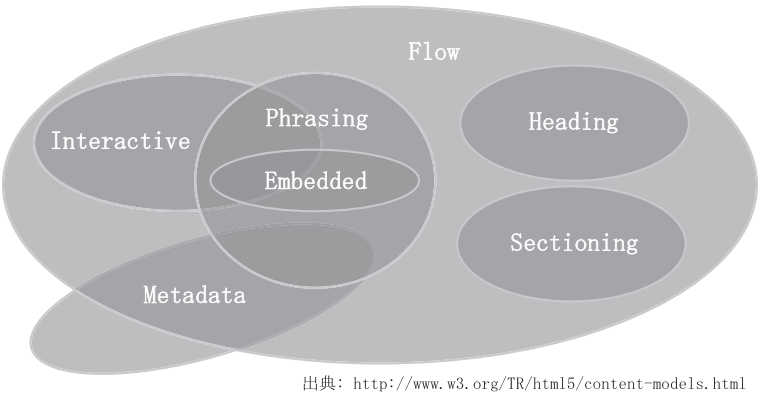
これらに含まれる CSS3（Cascading Style Sheet Level 3），SVG（Scalable Vector Graphics），およびいくつかの JavaScript API の仕様は，厳密には HTML5 仕様とは別枠の仕様として策定されているが，一般的には暗黙的に HTML5 に含める場合が多い．従って，本稿において取り扱う HTML5 の範囲は広義の HTML5 を対象とする．

### 3. 2 カテゴリ

従来の HTML の要素は，ブロック・レベルとテキスト・レベル（インライン）という大雑

把な概念での分別がなされていたが，HTML5 の各々の要素は，その要素が持つ特徴の種類と，その要素に含めることができる内容についての要件が詳細に規定され，従来と比較してマークアップの規則が明確になっている．

類似した特徴を持つ要素の集合を「カテゴリ」と呼ぶ．カテゴリは，メタデータ・コンテンツ，フロー・コンテンツ，インタラクティブ・コンテンツ，フレージング・コンテンツ，ヘディング・コンテンツ，セクショニング・コンテンツ，エンベデッド・コンテンツの七つに分類されている．特定のカテゴリに属さない独自の特徴を持った要素もあるが，一般的に各々の要素は，これらのカテゴリのいずれかひとつ，または，複数のカテゴリに属することになる．各カテゴリの関係性を図 3 に示す．




出典：http://www.w3.org/TR/html5/content-models.html

**図 3　カテゴリ関係図**

ある要素に含めることができる子要素やテキスト等の内容，および，それぞれの内容の要件を記述したものを「コンテンツ・モデル」と呼ぶ．コンテンツ・モデルには，カテゴリが記述されることもあるが，その場合は“そのカテゴリに属する要素”という意味になる．また，「transparent（透過）」と記述される場合は“親要素のコンテンツ・モデルを透過的に継承する”という意味として扱われ，例えば，br のような子要素を持たない場合のコンテンツ・モデルには「Empty（空）」と記述される．

### 3.3　アウトライン

一般的な文書では，章，節，項等のそれぞれの見出しによって階層的な区分け（セクション）を作ることで文脈を整理しており，このような階層構造を「アウトライン」と呼んでいる．

HTML5 では，セクショニング・コンテンツとヘディング・コンテンツが新たに規定されており，マークアップのアウトラインを形成することができる[*6]．セクショニング・コンテンツ（section，article，nav，aside）は，セクションの範囲を定義する為に新しく追加された要素である．ヘディング・コンテンツ（h1，h2，h3，h4，h5，h6，hgroup）は，各セクションの見出しを定義する要素であり，それ自身が暗黙的にセクションを表すことができる．また，「セクショニング・ルート」と呼ばれる要素（blockquote，body，details，fieldset，figure，td）は，その要素を最上位とする文書を記述することができ，他とは独立したアウトラインを形成することができる．

アウトラインは，HTML 文書の論理的な構造の中核となる概念である．セマンティクス（5.1 節参照）の恩恵を享受する上で，アウトラインや各要素の持つ意味合いを強く意識してマークアップをすることが，今後の重要なキーポイントとなる．

## 4. HTML5 の技術解説

HTML5 にはどのような技術があるか．本章では業務システムへの適用を想定し，いくつかの例を挙げて解説する．

### 4.1 デザイン

業務システムにとって「見栄え」は，必ずしも機能やコストを差し置いて優先される要件ではない．しかし，文字の大きさ，色，形，ページのレイアウト等のデザインは，そのシステムを利用する上での「理解しやすさ」「使いやすさ」といったユーザビリティ[3]の向上に欠かせない要因のひとつである．

従来の HTML では，非推奨とされながらもプレゼンテーショナルな要素が使われていた．例えば，font 要素は，文字の大きさ（size 属性）や文字の色（color 属性）を指定する為の，いわゆる物理要素であるが，HTML5 では，見た目を定義する為に使われていたこれらの要素や属性を廃止し，スタイルシート（CSS3）に本来の役割を委譲することで，HTML 文書における「構造と体裁の分離」を強化している．

CSS3 では，スタイル適用対象の選択子（セレクタ）で扱える新しい擬似クラスが追加されており，HTML の要素にレイアウト目的で記述された余分な id や class の定義が削減できることから，メンテナンス性の向上が期待されている．さらに，従来の Web 画面では，見出しやボタンの装飾の為に，わざわざ画像ファイルを作成して貼り付ける場合があったが，CSS3 では Web フォント（@font-face）を使用して，クライアント環境にインストールされていない任意のフォントを指定できるようになり，ブロック要素のプロパティを設定するだけで角丸の矩形（border-radius）や影（box-shadow），グラデーション（gradient）等のグラフィカルな視覚効果が表現できるようになった（図 4）．


※Google Chrome 13.0でレンダリング

**図 4　CSS3 サンプル**

その他に，移動・回転・拡大・縮小といった文字や図形の変形（transforms）や，シンプルなアニメーション（transitions）の表現まで簡単に実装が可能になるが，これらは表現力の向上のみならず，HTML ページ全体のデータ・サイズの削減と，それに伴う実行時のメモリ使用量およびネットワーク/サーバ負荷等のシステム的な観点でのリソース消費量の削減効果も見込まれている．また，ページのレイアウト構成についても，段組（Multi-column Layout Module）や柔軟なレイアウト設定が可能な BOX プロパティ（display:box;）が用意され，多様化するメディア（画面サイズ）毎に別々のスタイルシートを適用することも可能となり

(Media Query)，ソース一元化による開発生産性向上等のエンジニアリング面での効果も期待されている．

ただし，現段階では各ブラウザベンダーの独自拡張機能という位置付けのプロパティや，先行的に実装されているプロパティが混在している．それらは将来の仕様変更リスクに備え，プロパティ名の先頭にベンダープレフィックスと呼ばれる識別子が付けられている．その場合は，実装の際にベンダープレフィックスの無いプロパティと，ベンダープレフィックスを冠したプロパティを，それぞれ記述しておく等の暫定的な処置が必要となる．

### 4.2　グラフィックス

大量データからの情報抽出と画像描画の高速処理を実現した Silverlight の PivotViewer は，データの分類とプレゼンテーションを同時に具現化した新しい表現方法を示してくれている[*7]．このようなグラフィカル表現は，チャートの描画や BI（Business Intelligence）のダッシュボード等，業務システムにおいても有効なデータ可視化ツールとして利用されている．

HTML5 では，ブラウザ上で図形や線分等のグラフィック描画をする為の Canvas2D[*8] と SVG の二つの仕様が策定された．Canvas2D はビットマップ画像を描画する為に策定された仕様であり，canvas 要素で HTML 文書内に描画領域を指定し，2D コンテキストという JavaScript の API を使用して図形を描くことができる．SVG はベクター形式の図形を記述する為のマークアップ言語であり，基本的な図形要素として矩形（rect），円形（circle），楕円形（ellipse），線分（line），折線（polyline），多角形（polygon）が用意されており，複雑なベジェ曲線（path）を描くこともできる．SVG は XML をベースとした独自の言語であるが，HTML 文書中に svg 要素を直接記述することが可能になり，この仕様を「インライン SVG」と呼んでいる．

どちらのグラフィック描画方式を採用するかの選択には，両者の特徴を十分に押さえておく必要がある．例えば，ベクター形式の画像は，ビットマップ画像と比較して拡大や回転等の変形に強いという特性を持っている．Canvas 画像は拡大すると輪郭がぼやけてしまうが，SVG 画像はエッジがはっきりとしている（図 5）．また，Canvas 画像では，画像の一部のみを書き換えることができない為，アニメーションを実装する際は，画像全体を塗りつぶすか一旦消去して再描画しなければならない．

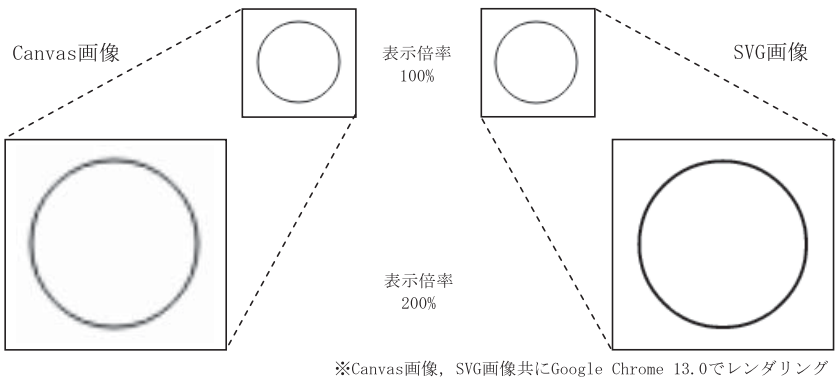



**図 5　Canvas 画像と SVG 画像の拡大比較**

インライン SVG は，モバイル標準ブラウザでは対応されていない．Canvas2D は，Internet Explorer（以下，IE と呼ぶ）の古いバージョン[*9]では対応されていないが，擬似的に canvas 要素が利用できるような JavaScript ライブラリが Google から提供されており[*10]，現時点ではモバイルを含めた殆どの主要ブラウザでの利用が可能である．

### 4. 3　マルチメディア

動画は，テキストや静止画と比較して単位時間当たりの情報伝達量が多く，具体性，確実性に優れており，提案やプレゼンテーション・広告等の訴求力および説得力を高めることのできるツールとして既に多くの場面で活用されている．

Flash Player は，YouTube やニコニコ動画等の著名な動画共有サイトで採用されていることでも知られており，Web ブラウザ上で動画を再生するソフトとして，全世界での普及率が 95％[*11]に達する程のデファクトスタンダード製品である．しかし，プロプライエタリ（独自の）なプラグインソフトをインストールする必要があり，数多度のセキュリティ脆弱性対応アップデートを要することもある．また，スマートフォンやタブレット等の新興デバイスの標準ブラウザでは，Flash Player や Silverlight のプラグイン対応が不十分であり[*12]，動画や Flex 等で作成されたリッチ・コンテンツを閲覧することができないという課題が生じている．

HTML5 では，audio 要素や video 要素を HTML 文書に記述することで，特別なソフトやプラグインがなくても音楽や動画の再生が可能となり，尚且つ，JavaScript を使って変速再生や逆再生のコントロールも可能となる[*13]．また，スマートフォンやタブレットの標準ブラウザでも対応されている為，Flash に代わるモバイル向けコンテンツのプラットフォームとしても期待感が高まっている．ただし，HTML5 の動画規格については，ブラウザによって対応方針が異っており[4][5]，当面は，それぞれのブラウザで取り扱うことができるビデオ・ファイルの種類に関心が向けられることになろう．尚，現時点では，高品質だが特許料の支払いを要する動画コーデック H.264 を推進する IE，Safari と，Google が開発したオープンな動画規格 WebM（動画コーデックは VP8）を推進する Firefox，Opera，Google Chrome の二派に大別されている．

### 4. 4　オフライン

一般的に，ユーザはインターネットやイントラネットに接続したオンラインの状況で Web アプリケーションを利用しており，ネットワークに接続していないオフラインの状況では，勤務状況報告や営業日報を入力することすらできない．HTML5 では，オフラインでも Web アプリケーションを利用することができる新たな機構を提供する．

Offline Web[*14]は，アプリケーションの操作を継続する為に必要な HTML や CSS，JavaScript ファイル等をローカル環境に蓄えておく「キャッシュ・マニフェスト」というメカニズムであり，キャッシュすべき対象を記述したマニフェスト・ファイルを html 要素の manifest 属性に指定するだけで利用することができる．

Web Storage[*15]は，キーと値が対になったデータ・アイテムをローカル環境に保存することができる．データの永続性については，セッションの有効期間のみ一時的に利用可能な「セッション・ストレージ」と，保存期限が無く永続的に利用可能な「ローカル・ストレージ」の二種類が選択できる．これらの性質は HTTP Cookie に似ているが，相違点としてクライアント

/サーバ間の HTTP 通信に付随したデータ送受信がされないということと，同一 Origin（ドメイン）の場合，複数のアプリケーションでストレージを共有するということが挙げられる．また，HTTP Cookie では，データ容量の上限は 4K バイトであったが，Web Storage では制限容量の規定はされておらず，より大きなサイズのデータを扱えるようになる．ただし，一般的なデータベースのような複雑な検索ができない点や，HTTP Cookie のように古いデータに有効期限が設定できない点には留意が必要である．

### 4.5 インタラクション

Web アプリケーションの操作性や利便性を向上させる為の技術として，HTML5 では，デスクトップ環境との間でデータ連携する機構や，JavaScript を並列処理する機構が提供される．

File API[*16] は，クライアント環境のデスクトップ・ファイルの読み取りを可能にする．ただし，現時点ではセキュリティを考慮して，ファイルの選択にはユーザ自身の操作を介することが前提とされており，同様に，ファイルの更新や削除をすることも制限されている．

Drag and Drop[*17] は，HTML の任意の要素や選択されたテキストのドラッグ＆ドロップを実現する．前述の File API と組み合わせて，デスクトップ上のファイルをドロップすることもできる．

Web Workers[*18] は，JavaScript を並列処理する為の「ワーカ」と呼ばれる仕組みを提供する．あまりにも重い JavaScript の処理が実行されるような状況や，ネットワークが混雑して通信の応答が返ってこないような状況では，ブラウザがフリーズして画面の操作ができなくなることがある．このような場合に，ワーカを利用して JavaScript の処理をバックグラウンドで実行し，ブラウザの待ち状態を回避することで，画面の操作性を向上し，処理効率を改善することができる．ただし，ワーカで使用できる API は次節に述べる XMLHttpRequest や Web Sockets 等の一部に限られている．また，セキュリティ上の対処として，ワーカからの DOM 操作は制限されている．

### 4.6 コミュニケーション

ある Web サイト（ドメイン）で読み込まれたページから，別の Web サイト（クロスドメイン）のページやデータにアクセスしようとすることは，ブラウザによって厳重に制限されている．これは「クロスドメイン制約」と呼ばれており，XSS（Cross Site Scripting）対策に代表されるセキュリティ面での考慮であるが，Web API を利用したマッシュアップ・サービスを構築しようとする場合には不便な制約であった．

XMLHttpRequest Level 2[*19] は，これまで制限されていた Ajax でのクロスドメイン通信を可能にする仕組みである．クロスドメイン通信のベースとなる仕様として Cross-Origin Resource Sharing[*20] に準拠している．この仕様は，リクエスト元とリクエスト先を「オリジン」という概念で捉え，異なるオリジン同士での資源共有の方法を規定している．

Web Messaging[*21] は，例えば，iframe 要素で作成されているような，それぞれ独立した複数のガジェット間で，クロスドメインのデータ受け渡しを可能にする「クロスドキュメント・メッセージング」という機構を提供する．さらに，上りと下りの二つの通信ポートを持ったチャネルを利用することもできる．

HTTP 通信プロトコルは，リクエスト＆レスポンス型のステートレスな通信方式であり，

クライアントから要求を送信し，サーバからの応答を受信すると一連の通信手続きが終了してしまう為，セッションを管理するにはHTTP Cookieを用いる等の代替手段が必要であった．

Web Sockets[*22]は，ステートフルなソケット通信を実現する仕組みである．クライアントとサーバの間でコネクションを確立した後は，任意のタイミングでの双方向通信が可能となる．HTTP通信と比較してオーバーヘッドが少ない為，頻繁にデータ送受信を必要とする場合の通信方式に適しており，リアルタイム性を要求される場合には，サーバからPUSH型データを送信することも可能である．尚，API部分の仕様はW3Cが策定しており，通信プロトコルの仕様はIETF（Internet Engineering Task Force）が策定している．

## 5. 今後の展望

業務システムのHTML5対応をする上で検討が必要な課題と，現時点で考えられる施策について意見を述べる．

### 5.1 セマンティクス

table要素でレイアウトが構成されたレガシーなHTML文書や，div要素が溢れるHTML文書のコードは，メンテナンスする人間にとってもブラウザや検索エンジンにとっても理解が難しいものである．セマンティックWeb[*23]では，HTML文書やそこに含まれるデータの意味をコンピュータに分かり易く記述することが，現在においては未だ存在しない将来の新しいエクスペリエンスの基礎になると考えられている．

HTML5では，HTML文書における文書構造の改善と意味付けの明確化を標榜し，新たな要素の追加と仕様の改定が検討されている．3.3節で述べたアウトラインも，そのうちのひとつであり，セクショニング・コンテンツやヘディング・コンテンツは，そこに記述されている内容が，主要な記事を表しているのか，補足的な記事を表しているのかという，文書全体の論理構造を明示的に表現する．また，HTML文書データが持つ意味合いを適切なマークアップで記述する為に，装飾的な要素の内，ある要素（basefont，big，center，font，strike，tt）は排除され，ある要素（i，b，em，strong，hr，small）は，文書の意味を表す為の役割を再び割り当てられた．

また，検索エンジン等のマシンが読み取ることのできる付加情報（メタデータ）を記述する為の仕組みとして，以前から提唱されているRDFaやMicroformatに加えて，MicrodataやCustom Data Attributeという仕様が検討されている．

これらはWebアプリケーションが発展していく上で大いに期待されている技術であるが，現時点で具体的な活用方法は確立されていない．その為，セマンティクスについては，今後，更なる議論と時間を要すると考えられる．また，古いブラウザでは，新しく追加された要素や改定された仕様は正しく解釈されないので，次節で述べるクロスブラウザの問題にも留意しなければならない．しかし，マークアップ要素の本来の意味を理解し，文法に則った文書を記述すること自体は，メンテナンス性と品質の向上という重要なメリットに繋がる．社内データの登録や検索・分析を目的とした業務システムのHTML文書は，情報の公開を目的としたブログ等と違って文書の意味付けが難しい場合があるが，これらの標準化動向を踏まえた開発基準の見直しと検討を推奨する．

### 5. 2　クロスブラウザ

ブラウザはインターネットにアクセスする為の HTTP ユーザエージェントであり，取得した HTML 文書の構文を解析し，それを人間が理解できるような文字や画像にして描画する[*24] クライアント・ソフトウェアである．現在利用されているブラウザは，StatCounter[*25] で確認できるように，様々なベンダーとバージョンが混在している状況である．

インターネットの普及期における主要ブラウザ Internet Explorer 6.0（以降，IE6 と呼ぶ）は，2001 年のリリース以降，ブラウザ市場を殆ど独占するまでに利用率を拡大し，多くの企業において標準ブラウザと位置付けられるようになった．その後，IE6 の開発は一時期停滞していたが，安定性と互換性を維持したまま，長い間，企業内の業務システムの開発基準とされていた．そして，2005 年頃からタブ・ブラウジング等の新しい機能を有したモダンブラウザ[*26] が台頭し，次いで 2006 年には IE6 の次期バージョン IE7 がリリースされた．しかし，それまで IE6 の機能を基準として開発していた業務システムには，新しいブラウザでは画面の閲覧や操作ができなくなるような DOM および CSS の非互換があり，自社標準ブラウザのバージョンアップを見送る企業が散見された．DOM や CSS は動的な HTML を実現する為の IE6 の独自機能であったが，それらの標準化と仕様の策定が後追いとなったことで，他のモダンブラウザとの間に実装の差異が生じる結果となった．つまり，非互換の多くは IE6 が Web 標準に準拠しなくなったことに起因しているといえる．Microsoft 社は，自身が運営する特設サイト「The Internet Explorer 6 Countdown」[*27] を公開し，IE6 の利用終了と新しいバージョンへのアップグレードを推奨しているが，未だ IE6 を標準ブラウザとしている企業は少なくない．

IE6 のように，10 年以上も前にリリースされた古いブラウザでは，当然，HTML5 の対応はされていないが，モダンブラウザにおいても，HTML5 のどの仕様が実装されているかには，ブラウザ毎，またはそれらのバージョン毎に差異がある[*28]．例えば，画面の入力項目をマークアップする form 機能には，input 要素に新しい type 属性がいくつか追加されたが，日付入力（〈input type=″date″〉）項目にカレンダを表示するような便利な機能は，Opera 以外では実装されていない（図 6）．また form 機能には，output 要素が新しく追加されているが，現時点ではいずれのブラウザもこれを実装していない．そして今後，どのブラウザも HTML5 の仕様を完全に実装するとは限らないのである．


**図 6　Opera の日付入力タイプ**

クロスブラウザとは，HTML5 の仕様が策定される以前にリリースされた古いブラウザと，各モダンブラウザの HTML5 対応状況の差異に対する検討議題である．これまでは，どのブラウザでも同じように閲覧できることをクロスブラウザ対応のゴールとして捉えていたが，現状では，その前に非常に険しく困難な道程が続いている．

これに対し，HTML 文書の本来の目的は情報の伝達にあるということに主眼を置き，“必要な情報を欠落させない”という観点で見直された開発アプローチがある．これは，機能的に上位のブラウザと下位のブラウザに対し，それぞれに見合ったデザインと機能を提供するという考え方で「Progressive Enhancement」と呼ばれている[6]．

例えば，Modernizr[*29] のサイトでは，古いブラウザで閲覧した場合でも，画面の表示が崩れて読み難くなったり，情報が欠落しないような考慮がされており，FireFox6 には角丸の矩形やグラデーション，Web フォント等の CSS3 の新しい機能を使ったデザインを提供し，IE6 には CSS3 を使わないシンプルなデザインを提供するという実装がされている（図 7）．


IE6でブラウジング　　FireFox6でブラウジング

**図 7　Progressive Enhancement**

HTML5 に対応していないブラウザでも実装が可能な UI（User Interface）機能を提供する JavaScript ライブラリがある．jQuery UI[*30] には，アコーディオン・パネルやボタン等の部品が用意されており，古いブラウザ上でも簡単にインタラクティブな UI を利用することができる．また，jQuery Mobile[*31] にはモバイル向けの画面フレームが用意されており，画面遷移（ページ切り替え）時のアニメーション効果（Transition）も容易に実装することができる．その他にも，古い環境で新しい機能をエミュレートする「Polyfill」と呼ばれる JavaScript ライブラリを併用する実装方法もある．

モダンブラウザのバージョンアップ・サイクルは，年々短くなる傾向にあり，業務システムの製作者にとっては，設計やテストはおろか，動作保証についても悩ましい状況である．クロスブラウザの対応方針を固める上では，ブラウザ毎の HTML5 対応状況と差異を把握することが重要であり，これらの開発アプローチや実装方法等の予備知識を整理し，事前に評価しておくことが肝要である．

## 6. お わ り に

これまでの Web 技術には様々な制約が設けられていたが，HTML5 は通信，インタラクション，ストレージ，マルチスレッド等のアプリケーション機能を強化している．これらの Web アプリケーションは，スマートフォンやタブレット等のモバイル端末，無線ブロードバンド，

仮想化技術，クラウド等の新技術と並び，今後のワークスタイルやライフスタイルに大きな変化をもたらす重要な役割を担っている．また，タッチパネルやカメラによるジェスチャ操作，音声入力，GPS や各種センサ等の新しいデバイスやインタフェースとの融合により，これまでにないユーザビリティと，高度なユーザエクスペリエンスを実現する可能性がある．

業務システムにおける RIA の表現力や操作性は「高価な機能」といった解釈をされる場合があるが，HTML5 では，Web 標準の技術として RIA 機能を手軽に利用することができる．正式勧告となるまでには，まだ少し時間がかかるかも知れないが，多くの仕様が主要ブラウザに実装されて実用段階になった時に，新しい技術に迅速に対応し，顧客の投資効果を十分に引き出せるサービスを提供できるよう，新しい標準仕様に適した開発基準の整備を強化したい．

最後に，本稿の執筆にあたり，多くの有識者の方々からご助言とご指摘を頂いた．ここに記して感謝する．

---

＊ 1　意味をなさなくなった決まり文句［buzzword］.
＊ 2　旧 macromedia flash．2006 年に macromedia 社が adobe 社に買収された.
＊ 3　ただし，ランタイムのインストールが必要.
＊ 4　WHATWG FAQ〈http://wiki.whatwg.org/wiki/FAQ〉
＊ 5　W3C HTML5 LOGO〈http://www.w3.org/html/logo〉
＊ 6　これらはアウトライン形成の目的にのみ使用し，レイアウトやスタイリング目的には div 要素でマークアップすることが推奨されている.
＊ 7　「input and interactive devices collection」（Microsoft Research 社 Bill Buxton 氏）〈http://research.microsoft.com/en-us/um/people/bibuxton/buxtoncollection/pivot.htm〉
＊ 8　二次元画像を取り扱う．Canvas3D（WebGL）は現時点で未定義.
＊ 9　IE6/7/8 以下．Microsoft は IE9 から HTML5 の積極的な採用を表明している．ただし，IE9 は Windows XP をサポートしていない.
＊10　「ExplorerCanvas」ダウンロード・ページ〈http://excanvas.sourceforge.net/〉
＊11　「Rich Internet Application Statistics」より．〈http://www.riastats.com〉Silverlight も 70%を越えている.
＊12　Android2.2 以降で徐々に Flash Player に対応している．iOS は執筆時点で Flash Player の対応をしない方針を表明している.
＊13　YouTube では HTML5 トライアル・サイトを公開している．〈http://www.youtube.com/html5〉
＊14　Offline Web〈http://www.w3.org/TR/html5/〉
＊15　Web Storage〈http://www.w3.org/TR/webstorage/〉
＊16　File API〈http://www.w3.org/TR/FileAPI/〉
＊17　Drag and Drop〈http://www.w3.org/TR/html5/〉
＊18　Web Workers〈http://www.w3.org/TR/workers/〉
＊19　XMLHttpRequest Level 2〈http://www.w3.org/TR/XMLHttpRequest2/〉
＊20　Cross-Origin Resource Sharing〈http://www.w3.org/TR/cors/〉
＊21　Web Messaging〈http://www.w3.org/TR/webmessaging/〉
＊22　Web Sockets〈http://www.w3.org/TR/websockets/〉
＊23　W3C Semantic Web〈http://www.w3.org/standards/semanticweb/〉
＊24　「レンダリング（rendering）」という.
＊25　「StatCounter GlobalStats」〈http://gs.statcounter.com〉
＊26　Web 標準に一定の水準以上準拠しているブラウザ（もしくは，そのバージョン）．FireFox（Mozilla），Chrome（Google），Safari（Apple），Opera（Opera Software）等が該当する.
＊27　「The Internet Explorer 6 Countdown」〈http://www.ie6countdown.com〉
＊28　「When can I use...」各ブラウザの HTML5，CSS3，SVG 等のサポート状況が確認できる．〈http://caniuse.com〉
＊29　「Modernizr」HTML5 や CSS3 の適用状況を調べられる．〈http://www.modernizr.com〉
＊30　「jQuery user interface」〈http://jqueryui.com〉
＊31　「jQuery mobile」〈http://jquerymobile.com〉

**参考文献**
［1］ W3C, “HTML5 differences from HTML4”, May 2011, http://www.w3.org/TR/html5-diff/
［2］ RIA コンソーシアム,「主要 RIA 技術構造比較」, 2010 年 6 月, http://www.ria-jp.org/download/tech_comp20100601.pdf
［3］ ヤコブ・ニールセン, 篠原 稔和（訳）, 三好 かおる（訳）,「ユーザビリティエンジニアリング原論」, 東京電機大学出版局, 2002 年 7 月
［4］ 羽田野 太巳,「徹底解説 HTML5 API ガイドブックビジュアル系 API 編」,（株）秀和システム, 2011 年 1 月
［5］ wikipedia,「H.264 ウェブブラウザ」, http://ja.wikipedia.org/wiki/H.264
［6］ 菊池 崇,「クロスブラウザから Progressive Enhancement という制作コンセプトへ」, the edge newsletter, アドビシステムズ(株), 2009 年 2 月号, http://www.adobe.com/jp/newsletters/edge/february2009/articles/article4/index.html

※上記参考文献の URL は 2011 年 10 月時点での存在を確認.

**執筆者紹介** 川 田 清 忠（Kiyotada Kawata）
1992 年日本ユニシス(株)入社. 金融事業部の保険/証券担当 SE として Web システム等のシステム開発に従事. 2010 年より総合技術研究所に移籍し, スマートフォン等の新デバイスを含むクライアント領域の次世代技術の評価と研究を主体とした活動をする.