10'00"00 インターネット10分講座 INTERNET

HTML5

HTMLはHyper Text Markup Languageの略で、いわゆるWebのコンテンツを記述するための言語です。その第5版にあたる「HTML5」は、文書の論理構造をより明確に記述できること、アプリケーションの作成を容易にすることを目的としています。その動きは2004年に始まり、2008年に最初のドラフトが公開されました。2012年末には仕様策定作業はほぼ終了し、各種のブラウザでもサポートされ始めています。今回の10分講座では、アプリケーションを作れるようになったHTML5を紹介します。

◆ はじめに

HTML5は、HTML 4.01では表現しきれなかった文章の論理構造と、ユーザーインタフェース(UI) を定義し、さらにアプリケーションを作成するためのAPI (Application Programing Interface)を策定するといった機能の充実を図っています。これによって、広く普及したHTML 4.01を置き換えていくことを目的としています。さらにHTML 4.01では曖昧な解釈が可能だった部分をはっきりと定義して、Webブラウザ間の互換性を高めることにも留意しています。後方互換性については、HTML5でコンテンツを作成する場合に必要な要件と、Webブラウザがサポートすべき要件とを定義することで対応しています。

なお、HTML5ではextension specificationsという仕組みが取り入れられ、段階的に新しい機能を追加していけるようになっています。そのため詳細を知らない人は「HTML5」と一括りにとらえてしまいますが、厳密に言えば、現時点で規格化されている部分は狭義のHTML5でしかありません。規格化をめざして各種Webブラウザ上でいろいろな機能が実現されていて、それを含めて広義のHTML5と言えるでしょう。本稿では、新しく規格化されたこの狭義のHTML5の部分と、それ以外の実装である広義のHTML5とに分けて、解説をしていきます。

◆ 狭義のHTML5

まずは規格となっている狭義のHTML5から見ていきましょう。HTML 4.01と構文上で大きく違っているのは、

- doctype宣言や文字コードの指定が簡略化されている(コード1、2)
- 本文中にMathML(Mathematical Markup Language)やSVG(Scalable Vector Graphics)という別の言語で記述できる

ことです。さらに代表的な違いを、追加された要素(いわゆるタグ。以下、エレメント)、変更されたエレメント、変更された属性(エレメントにさまざまな指定を行うオプション)、削除されたエレメントから確認してみましょう。

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html lang="ja">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=ISO-2022-JP">
  <meta http-equiv="Content-Script-Type" content="text/javascript">
  <meta http-equiv="Content-Style-Type" content="text/css">
  <title>html4 の骨組み </title>
 </head>
 <body>
  <h1>html4 の骨組み </h1>
 </body>
</html>
```

コード1
HTML4.01 のサンプル。1 行目やヘッダ内の記述が多くなっています

```
<!doctype html>
<html lang="ja">
 <head>
  <meta charset="utf-8">
  <title>html5 の骨組み </title>
 </head>
 <body>
  <h1>html5 の骨組み </h1>
 </body>
</html>
```

コード2
HTML5 のサンプル。HTML4.01 に比べると、 ずいぶんシンプルになりました

なお、ここでは全体を俯瞰するにとどめ、細かい文法までは解説しません。また、HTML5のすべてを網羅的に取り上げているわけでもありません。正確な情報は、W3CのWebサイトでご確認ください。

HTML Current Status
http://www.w3.org/standards/techs/html#w3c_all

◆ アプリケーションAPI

HTML5では、いわゆるWebアプリケーションを作成するためのAPIを規定しています。代表的なものとして、

- 映像や音声を再生するAPI
- オフライン動作を可能とするApplication cache API
- draggable属性と組み合わせてドラッグ&ドロップを扱うAPI
- ユーザーとの対話を可能とするAPI(alert()、prompr()など)
- タイマーによって自動的に呼び出されるスケジュールAPI

などがあります。既に広く普及しているものもあれば、新しいAPIもあります。いずれにしても、HTML5として規格化することで相互運用性が高まり、さまざまなWebブラウザで同じように動作することが期待できます。

◆ 新しいエレメント

HTML5で新しく追加されたエレメントを、**表1**に示します。

表1
HTML5で新設されたエレメント。videoやrubyなどはブラウザによってだいぶ挙動が違います

グループ	エレメント名	機能
文書の論理構造記述用	article	section などを含んだ、一つの独立した記事を示す。
	aside	補足的なコンテンツを示す。
	figure	図、写真、グラフなどを示す。
	footer	フッタ用。
	header	ヘッダ用。
	main	主たるコンテンツを示す。
	nav	メニューなどのナビゲーションを示す。
	section	章などのまとまりを示す。
マルチメディア	audio	音声再生用。
	video	動画再生用。
グラフィックス	canvas	ビットマップグラフィックス用。
ユーザーインタフェース	progress	進行状況を示す。
	meter	範囲を持つ数値の表示用。
ルビ	ruby	ルビをふる範囲の指定用。
	rt	ルビそのものの指定用。
	rp	ルビを表す記号指定用。

論理構造記述用のうち、header、footer、nav、figureなどは使い方もわかりやすく、すぐに利用できます。対してsectionは節を表し、articleは独立した記事を表しますが、主観によっても変わってくるので使い方の難しいエレメントです。

HTML5ではaudioエレメントとvideoエレメントがあるので、Webブラウザだけで音声と動画を再生可能です。ですが、動画のコーデックまでは標準化されておらず、sourceエレメントで複数のファイルを指定してフォールバックする、というのが2013年9月時点での現状です。

canvasはビットマップを扱うエレメントですが、描画命令によってインタラクティブな表示を行えます。また、2Dだけではなく3Dも扱えるようになっていて、特に3DについてはWebGLと呼ばれています。またベクターグラフィックスを扱うために、SVGを規格として取り込んでいます。

progressは処理の進捗などを示すのに使います。典型的なものとしては、処理が進むにつれてバーが伸びていくといった表示などに使われます。対してmeterはある範囲内にある数値を表すのに使います。例えば、記憶領域の残り容量などです。prograssと違って、時間的に変化するものを表示するのには使いません。

rubyは、いわゆる「ふりがな」を表現するものです。理想的な場合は本文中の漢字の上に小さく表示されますが、Webブラウザによってかなり表示が違ってきます。

◆ 変更されたエレメント

使い方や意味の変わったエレメントもいくつかあります。**表2**に代表的なものをまとめました。

表2
HTML5で使い方の変わったエレメント。smallやstrongはより論理構造的な意味合いが強くなりました

エレメント	従来の使い方	新しい使い方
a	インライン要素のみ含める。	ブロック要素も含められる。
cite	参考文献を示す。	作品タイトルを示す。
dl	見出しと説明からなる定義リストを示す。	名前と値で構成される連想リストを示す。
small	文字を小さくする。	細則などの補足的説明を示す。
strong	文字列の最強調。	重要な文章を示す。

使い勝手の面では、aエレメントの中にインライン要素だけでなくブロック要素を入れられるようになったのが便利です。また、citeやdlなどは、正しいものも間違ったものも含めて現状の使われ方を考慮した上で、新しい使い方を定義しています。より論理構造を記述するように変化したのがsmallやstrongで、従来とはかなり意味合いが違っています。

◆ 変更された属性

属性でもいくつか変更されていますので、**表3**にまとめました。

表3
HTML5で変更された属性。一番大きいのは、<a name="one">が使えないことです

エレメント	属性	変更点
a	name	利用不可。代わりに id 属性を使う。
a, area	target	非推奨ではなくなった。
img	border	"0" のみ指定可能。それ以外は CSS を使う。
	width, height	パーセントでの指定は不可。
input	type	tel、search、url、email、datetime、date、month、week、time、datetime-local、number、range、color が追加。
li	value	非推奨ではなくなった。
ol	start	非推奨ではなくなった。
script	type	JavaScript の場合は省略可能。
style	type	CSS の場合は省略可能。
table	border	"1" か "" のみ指定可能。CSS の利用が推奨される。
	summary	利用不可。

大きな変化としては、

- aエレメントのname属性が廃止。
- imgエレメントのwidth/height属性でパーセンテージ指定ができない。
- inputエレメントのtype属性がtel、search、url、email、datetime、date、month、week、time、datetime-local、number、range、colorと強化され、それぞれのデータに応じた入力手段や、入力データのチェックを、JavaScriptなどでプログラムせずに利用できる。

が挙げられます。これ以外にも、CSSで指定すべきとして削除された属性が相当数あります(**表4**)。同じように、削除されたエレメントは**表5**にまとめました。

総じて、HTMLではより文章構造を論理的に記述し、体裁はCSSで指定し、Webアプリケーションの作成を楽にする、といった方向性が見てとれます。

表4
CSSで表現すべきとして、削除された属性。左側はその属性を持っていたエレメント、右側が削除された属性

エレメント	属性
caption, iframe, img, input, object, legend, table, hr, div, h1, h2, h3, h4, h5, h6, p, col, colgroup, tbody, td, tfoot, th, thead, tr	align
body	alink, background, link, text, vlink
table, tr, td, th, body	bgcolor
object	border
table	cellpadding, cellspacing, frame, rules
col, colgroup, tbody, td, tfoot, th, thead, tr	char,charoff
br	clear
dl, menu, ol, ul	compact
iframe	frameborder,marginheight,marginwidth, scrolling
td, th	height, nowrap
img, object	hspace,vspace
s	noshade, size
li, ol, ul	type
col, colgroup, tbody, td, tfoot, th, thead, tr	valign
hr, table, td, th, col, colgroup, pre	width

表5
HTML5で廃止されたエレメント。basfontやisindexなどは、HTML 4.0の時点で将来的に廃止の予定となっていました

エレメント	意味	削除理由
acronym	頭文字を示す。	abbr とほぼ同じため。
applet	java アプレットを示す。	object の利用を推奨。
basefont	基準フォントサイズ	CSS で指定すべき。
big	文字を大きくする。	CSS で指定すべき。
center	文字の中央そろえ。	CSS で指定すべき。
dir	ディレクトリリスト。	実際は ul と変わらないため。
frame	ウィンドウの分割。	ユーザビリティを阻害するため。
frameset	ウィンドウの分割。	ユーザビリティを阻害するため。
isindex	入力フォームにおける 1 行入力。	<input type="text"> で代替すること。
noframes	ウィンドウの分割。	ユーザビリティを阻害するため。
strike	打ち消し線。	CSS で指定すべき。
tt	等幅フォント。	CSS で指定すべき。

◆ 広義のHTML5

規格としてのHTML5を俯瞰したところで、今度はより広い意味でのHTML5について紹介します。以下、HTML5とした場合は、広義のHTML5のことだと考えてください。

HTML5の規格化で推し進められているのは、アプリケーションを記述することです。いわゆるWeb 2.0という単語が現れた頃から、WebはJavaScriptによるローカルな処理を増やしつつ、よりインタラクティブになり、OSの種類を問わずWebブラウザさえ動けばどんなマシンでも同じアプリケーションが使えるのではないか、と言われてきました。Google社のChrome OSなどもこの延長上にあると考えられます。ただ実際には、JavaScriptではローレベルなハードウェアの制御が難しく、いわゆるネイティブアプリケーションとは区別されていました。

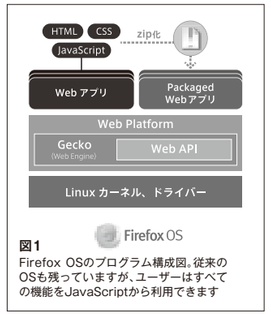

図1
Firefox OSのプログラム構成図。従来のOSも残っていますが、ユーザーはすべての機能をJavaScriptから利用できます

これを打破するのがHTML5だと言われています。一言で言ってしまえば、CやC＋＋用に提供されていたローレベルハードウェア制御用のAPIを、JavaScriptに対して提供しようというものです。実例としてはMozillaプロジェクトが開発するFirefox OSと呼ばれるソフトウェア群で動作する、スマートフォンがあります。ソフトウェア的な構成は**図1**のようになっており、いわゆるOSが無くなったわけではありません。しかし、タッチスクリーン、GPS、光センサー、カメラといったハードウェアをJavaScriptから直接制御できます。HTML＋CSS＋JavaScriptだけで、電話帳やカメラアプリケーションが作れてしまうのです。もちろん、電話帳は単に連絡先をリストするだけでなく、直接電話をかけられます。

また、Webベースではなく、独立したソフトウェアパッケージを作成することもできます。PCのWebブラウザからアクセスしたFirefox Marketplaceですが、App StoreやGoogle Playと同じようにアプリケーションを配布しています。ただ、後者と違ってスマートフォン上だけでなく、Windows上で動作するアプリケーションも含まれています。Windows用のアプリケーションはライブラリとしてfirefoxを使っていますが、Firefoxそのものは動いていなくても、きちんと独立して動作します。これらも、中身はHTML＋CSS＋JavaScriptで記述されています。

ハードウェアだけでなく、ファイルやデータベース、デバイスストレージ、コンタクトリストなども規格化が進んでいます。今まではできなかった、SDカードからMP3のデータを読み出して再生する、などということも可能になります。また、コミュニケーション関係では、Bluetooth、TCPソケット、ネットワークへの接続状況管理、SMSなどのAPIも整備されてきました。

とはいえ、CやC＋＋で書かれた既存のアプリケーションをJavaScriptに移植するには手間がかかります。そこで、C＋＋コンパイラによって生成された中間コード（Low Level Virtual Machine限定ですが）を、JavaScriptに変換するツールも開発されています。

これによって、C＋＋で記述されたゲーム実行環境が1週間ほどでHTML＋CSS＋JavaScriptの環境で動作するようになり、既存のゲームがほぼそのまま実行できるようになりました（**画面1**）。もちろん完全なものではありませんが、移植の手間がかなり減ることは事実です。

画面1
http://www.unrealengine.com/html5/
C++で書かれたプログラムをJavaScriptに移植したサンプルページ

◆ 通信スタイルの変化

HTMLで記述されたコンテンツを転送するプロトコルであるHTTPは、もともとクライアントからのリクエストに対してページを丸ごと転送して終わり、というシンプルなものです。実装が簡単になるという利点と引き替えに、サーバからクライアントに情報を送りつけたり、ページの一部を書き換えたりができません。これを既存の技術（JavaScriptによる通信と、同じくJavaScriptによるページの部分書き換え）を組み合わせることで解決してみせたのが、AJAX（Asynchronous JavaScript ＋ XML）です。Google Mapで地図のドラッグを実現して、一躍有名になりました。

とはいえ、ベースになっている通信技術はJavaScriptのXMLHttpRequestで、クライアントからサーバにリクエストを出すという点ではなんら変わりがありません。つまり、リアルタイムチャットのように、サーバ側の変化をクライアントに伝えて書き換えるアプリケーションを書くのは難しいということです。もちろん不可能ではなく、これを実現するためのCometと呼ばれるプログラミング手法も確立しています。しかし効率が悪く、よりスマートな解決方法が望まれていました。

こうした問題を解決するのがWebSocketです。HTTPやXMLHttpRequestとは違って接続したコネクションを維持し、専用のプロトコル（RFC 6455）を使ってクライアントからサーバへ、サーバからクライアントへとデータを双方向にやり取りします。いわゆるModern Browserすべてでサポートされており、リアルタイムチャットの実現例などもあります。まだすべての組み合わせで動作するほど枯れた技術ではありませんが、洗練されたPush技術として注目すべきでしょう。

WebSocketはサーバ／クライアント間の通信用ですが、Webブラウザ同士でのリアルタイム通信を可能にするのがWebRTCです。わかりやすいところでは、ボイスチャット、ビデオチャット、ファイルの共有が挙げられます。これを特別なプラグインを使わず、サーバも必要とせずに実現できる予定です。また、前述のゲームエンジンなどと組み合わせれば、複数プレーヤーが戦うシューティングゲームや、いわゆるMORPG（Multiplayer Online Role Playing Game）などの実現も可能になります（**画面2**）。


画面2
http://hacks.mozilla.org/2013/03/webrtc-data-channels-for-great-multiplayer/
WebRTCを使ったマルチプレーのテスト。PCごとにゲームプログラムが動き、相互にデータをやり取り

◆ おわりに～Webのプラットフォーム化にむけて～

従来のネイティブアプリケーションを動かすには、対応するOSを用意する必要がありました。しかし、HTML5によるアプリケーションであれば、OSに関係なくHTML5をサポートしたブラウザがあれば動作します。もちろんOSそのものが無くなってしまうわけではありませんが、ユーザーからするとどんどん意識しなくて済むようになります。OSがハードウェアの違いを吸収したように、HTML5がOSの違いを吸収するのです。

ただ、HTML5が普及すると、インターネットに対する要求も変化するかもしれません。従来のWebは基本的にサーバ／クライアントモデルで、情報の流れはほぼ1方向だったと言えます（**図2**）。そのため、ユーザー数が増えたときにはCDN（Contents Delivery Network）を構築して対処してきました。しかしWebRTCで要求されるのは、サーバからクライアントへの流れではなく、Webブラウザ同士がPeer to Peerで高速にデータを交換できることです（**図3**）。LANでつながっていればさして問題はありませんが、ISPを経由してある程度離れたユーザー同士が情報を交換するとなると、従来のCDNでは対応できないかもしれません。

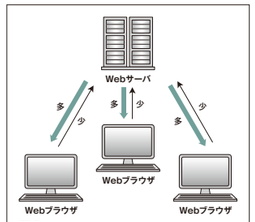

図2
現在のWebにおけるデータの流れ。サーバーからクライアントに流れる量が圧倒的に多くなっています

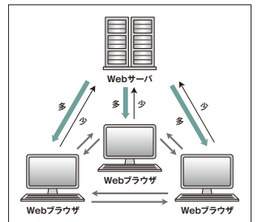

図3
WebRTCによるデータの流れ。量は多くなくても、クライアント同士の通信が行われます

こうした動きがどこまで拡大し、どこまで普及していくのはまだわかりません。けれども、HTML5によって、Webは確実にプラットフォームへの道を歩んでいます。

（一般社団法人Mozilla Japan テクニカルマーケティング 清水智公/JPNIC インターネット推進部 秋山智朗）

■CSS3

正確に言えばHTML5とは別の規格ですが、CSS3（Cascading Style Sheets 3）も着々と仕様の策定が進んでいます。HTMLが文書の論理構造を記述するのに対して、CSSは文字の大きさや色、間隔といった見た目を記述します。

最近注目されているレスポンシブデザインは、CSSのメディアクエリを使って実現することが多いようです。以前からscreen、printといった場合分けができましたが、加えてウィンドウの横幅に応じてCSSの指定を切り替えられるようになっています。HTML本文はなるべくシンプルでクリーンにしておいて、横幅に応じた表現をCSSで切り替えるわけです。デバイスに応じて破綻しないデザインを作るのは難しいですが、いったんデザインの決まった定型ページなら保守が楽になります。

文字関係では、Webフォントが注目されています。これは文字フォントをローカルに持つのではなく、インターネット上のサーバから持ってくるというものです。今までは指定したフォントを正確に表示するには、ユーザーのPCにそのフォントがインストールされている必要がありました。しかしWebフォントであれば、ユーザーPCの状態にかかわらずほぼ意図した通りに見せることができます。さらに拡大縮小も、コピー＆ペーストも自在ということになります。

さらに、文字の流し込み機能も実現されています。2段組のデザインでも、拡大縮小にあわせて自動的に文字が配分されます。さらに矩形だけではなく、任意の図形に対して文字を流し込む機能も検討されています。まだ標準的ではありませんが、Adobe社のWebサイト（http://html.adobe.com/webplatform/layout/shapes/）にいくつかサンプルが掲載されています。

文字以外でも、アニメーションや変形、トランジッションなどのサポートが進みつつあります。こちらは比較的Webブラウザによるサポートが進んでいるようで、これもAdobe社が作成している http://beta.theexpressiveweb.com/ でサンプルを見られます。