デジタル・ドキュメント　35－1
（２００２．　９．　２０）

# JavaScriptを用いたXLinkの実装方法について

大坂　哲司
株式会社フジミック（ＸＭＬコンソーシアム基盤技術部会兼務）

**概要：** XLinkはリソースとリソースとの関係付けを定義するもので、HTMLのハイパーリンクを超える機能を有している。本稿では、第三者リンクに基づくXLink実装を、JavaScriptのみを用いて実現し、その実装手段を報告する。XLink実装の結果、1対Nのマルチリンク、双方向リンク、CD-ROMなど書き込みができないコンテンツへのリンクなど、リンク連鎖ができるようになり、XLinkの有用性についても言及する。

# XLink Implementation using JavaScript

Tetsushi Osaka
Fujimic Inc.

Abstract: XLink defines relating with a resource and a resource and has the function exceeding the hyperlink of HTML. In this paper, XLink implementation based on a third party link is realized only using JavaScript, and the implementation means is reported. As a result of XLink implementation, a links chain, such as a multi-link of 1:N, such as a bidirections link, such as a link to CD-ROM contents which cannot do writing, comes be made, and reference is made also about the usefulness of XLink.

## 1 はじめに

XLinkはXMLを支援するXML基盤技術の一つで、2001年6月W3CはW3C勧告としてXLink1.0（XML Linking Language）を公開した[XLink]。XLinkは大変シンプルな仕様ではあるが、大変強力な仕様である。XLink勧告後に策定されたSVG(Scalable Vector Graphics)などのXML規格は、XLinkの機能を取り入れているものが多くなっている。

XLinkはHTMLのハイパーリンク機能を装備している。XLinkを簡潔に表現すると、“XLinkとは、一つのリソース（開始リソース）ともう一つのリソース（終了リソース）を関連付ける機構”となる[奥井、山本]。XLinkでは具体的なリソース記述方法は規定していない。リソース自身の記述は、URLによるファイル全体、XPathやXPointerによるファイルの一部分を指定することができる。またこれら以外にブラウザ上の処理、XQueryやCGIなどのようなサーバー上の処理を行うアプリケーションをリソースと指定することもできる。

これまでにいくつかのXLinkの実装例が報告されている[XLink][DuCharme2002]。これらのXLink実装方法は、Javaベースのブラウザで XLinkを実現するもの[富士通]、サーバーでXLST変換によってダイナミックHTMLを生成するもの[DuCharme2001]、サーバーで XInclude[XInclude]のようにデータリンクを実現するもの[Williams]、Mozilla などのブラウザに組み込まれ XLink を実施するものなどがある。先ごろリリースされたMozillaは単純リンク機能の実装に留まっている。XLink の真髄である拡張リンクにあるが、まだXLinkの機能

をフルスペックで搭載した一般的なブラウザは出現していない。本稿では、一般的なブラウザ上で拡張リンクを含む JavaScript による XLink の実装方法について、また実装した XLink 機能の評価について報告する。

## 2 XLink の実装について

XLink はサーバーサイトで実施するのもよいが、XLink はブラウザ上で動かしてこそ、ユーザーの利便性に寄与し、新しいインターフェイスを我々に提供してくれる。本稿では JavaScript による HTML-DOM 制御技術を駆使して、一般的なブラウザ上で XLink 機能（特に拡張リンクと第三者リンク）の実現を試みた。XLink 実装の開発環境並びに作動環境は、ブラウザとしてインターネットエクスプローラ(IE5.0 以上)を用い、リソースとして、私が管理する Web サイトのリソース並びに私の PC にあるローカルディスクのリソースを対象とした。XLink の実装方法を①XLink のシミュレーション、②XLink の基本要件とシステムのモジュール構成、③リンクポイントの埋め込み、④スクリプトの埋め込み、⑤XLink 実装手順について、順を追って説明する。

### 2．1 XLink のシミュレーション

リスト1 リソースの記述

```
<resource xlink:type="locator" xlink:label="A" group="A" xlink:href="a.html" xlink:title="Aのページ"/>
<resource xlink:type="locator" xlink:label="B" group="A" xlink:href="a.html#abc" xlink:title="画像表示"/>
<resource xlink:type="locator" xlink:label="C" group="C" xlink:href="c.html" xlink:title="Cのページ"/>
<resource xlink:type="locator" xlink:label="D" group="D" xlink:href="d.jpg" xlink:title="Dの画像"/>
<resource xlink:type="resource" xlink:label="E" group="E" xlink:href="javascript:deleteEmbed()" xlink:title="削除"/>
<resource xlink:type="locator" xlink:label="F" group="F" xlink:href="f.html" xlink:title="Fのページ"/>
```

リスト2 アークの記述

```
<arc xlink:type="arc" xlink:from="A" xlink:to="C" xlink:show="replace" xlink:actuate="onRequest"/>
<arc xlink:type="arc" xlink:from="A" xlink:to="F" xlink:show="replace" xlink:actuate="onRequest"/>
<arc xlink:type="arc" xlink:from="C" xlink:to="A" xlink:show="replace" xlink:actuate="onRequest"/>
<arc xlink:type="arc" xlink:from="C" xlink:to="F" xlink:show="replace" xlink:actuate="onRequest"/>
<arc xlink:type="arc" xlink:from="F" xlink:to="A" xlink:show="replace" xlink:actuate="onRequest"/>
<arc xlink:type="arc" xlink:from="F" xlink:to="C" xlink:show="replace" xlink:actuate="onRequest"/>
<arc xlink:type="arc" xlink:from="B" xlink:to="D" xlink:show="embed" xlink:actuate="onRequest"/>
<arc xlink:type="arc" xlink:from="D" xlink:to="E" xlink:show="delete" xlink:actuate="onRequest"/>
```

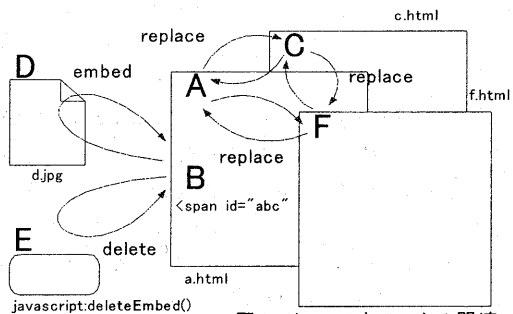

図1 リソースとアークの関連

一般の XLink 解説書では、図 1 に示したリソースとアークの関連を XLink によって説明するだけに留まり、XLink の具体的な動きや実装方法の解説は大変少ない。本稿では XLink の動きを図 2 に示すものと定め、この動きに従った XLink を実装する。その前に、この動きをダイナミック HTML でシミュレーションすると次のようになる。

リスト 1 においてリソース B（ラベル B を持つリソースをリソース B と呼ぶ）は"a.html#abc"と言うように XPointer のベアネームで記述したリソースで、a.html 中で id="abc"を持つ要素がリンクポイントとなる。このリンクポイントにマウスカーソルが入ると“画像表示”と言う吹き出しが表示される。この表示はリソース B に記述されている属性 xlink:title の値を HTML の title 属性に代入して吹き出し表示をする。このような演出以外にリンクポイントに対して、マウスフォーカスに合わせてリンクポイントの文字の色を替えたり、イメージボタンであればマウスフォ

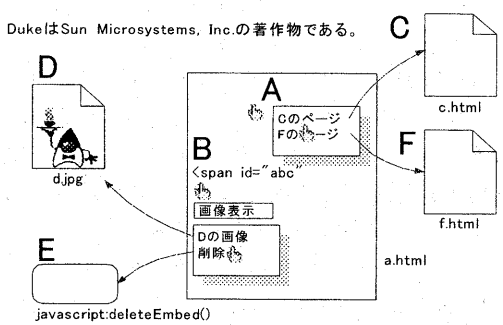

図2 ダイナミックHTMLによるXLinkシミュレーション

ーカスに合わせイメージを変えたりすることで、リンクポイントの存在を明示する。この明示のために、リンクポイントにonmouseover、onmouseoutイベントを生じさせる処理を加える。更にこのリンクポイントに対してonclickイベントが生じたとき、複数のリンク先を示すポップアップメニューを表示する処理を加える。メニューの項目が選ばれた時、項目が示すリンク先にジャンプする（A要素による記述）。このように1つのリンクポイントから複数のリンク先を表示することによって、1対多のマルチリンクが表現できる。一方、リソースAは明確なリンクポイントが指定されていない。このケースでは、リソースBで示されるリンクポイントやA要素などHTML標準のリンクポイント以外の全画面において、マウスがクリックされたとき上述のポップアップメニューを表示し、複数のリンク先へリンクさせることができる。上述の動きはダイナミックHTMLを用いてシミュレーションすることができる。

## 2．2 XLinkの基本要件とシステムのモジュール構成

以上述べた動きは通常ダイナミックHTMLで行うポップアップメニューによる動きそのものであるが、擬似的にXLinkを実装することができる。ダイナミックHTMLによるXLinkシミュレーションから、XLink実装の基本要件として次の項目が挙げられ、これらの項目をクリアすることにより、ブラウザ上で作動する本来のXLinkが実装できる。①XLinkで記述されたXMLに従って、リンクが実行されなければならない。②XLinkのリンクは指定した範囲で、リンク連鎖が続かなければならない。③XLinkはHTML上で作動しなければならない。④プレーンなHTMLに対して、XLinkの動き（リンクポイントの生成、XLinkを動かすJavaScriptの埋め込み）が付加できなければならない。

これらの要件を満たすXLink制御プログアムXLinkAPIは、次のモジュールから構成される。①XLink全体を制御するプログラム、②HTML-DOM並びにXML-DOMを制御するプログラム、③HTMLの埋め込み表示を制御するプログラム、④通常のページへのリンク制御APIを埋め込むプログラム、⑤リソースのXPointer/XPathを解釈し、リンクポイントを生成するプログラム、⑥画面上でマウスの動きを制御するプログラムから成る。本稿ではリンク連鎖に重要な⑤のリンクポイントの埋め込み、④のリンク制御APIの埋め込み、その詳細を以下に述べる。

## 2．3 リンクポイントの埋め込み

リンクポイントの形態は図2に示すように2つある。一つはリソースBが示すようにXPointerによってリンクポイントが明示されているケース、もう一つはリソースAのようにリンクポイントが明示されず、しかも全画面をリンクポイントするケースである。前者を“明示されたリンクポイント”と称し、後者を“明示されていないリンクポイント”と称する。“明示されたリンクポイント”はXPointer/XPathで定義されるリソースとなる。即ち、テキスト要素、画像、ボタンなどBODY要素配下にある指定された要素がリンクポイントとなる。これらの要素の中でテキスト要素は明示的にP要素から始まるテキストもあれば、BODY要素のテキストノードとして始まるテキストもある。このテキスト要素の全部または一部分をリンクポイントする指定方法がXPointerである。本稿ではXPointerプロセッサについて言及しないが、XPointerはきめ細かで強力なリソース記述手段である。しかしテキストの一部へのリンク修飾は通常SPAN要素によって行うが、BODY要素直下のテキストノードに対してリ

ンク修飾（要素の追加）はなかなか難しいものである。“明示されていないリンクポイント”では、明示されたリンクポイントやA要素などのリンクポイントを除いた全画面においてマウスクリックを受け付ける。

インターネットエクスプローラではinnerHTMLやouterHTML[MSDN1]によってHTML文中のテキストに対して図3のようにリンクポイント（ここではテキスト"cd"を有するSPAN要素をリンクポイントとする）の修飾が簡単に行える。この方法の問題点は、修飾を行った親要素、この例ではP要素全体が青色のリバース文字表示になり、マウスクリックによって修飾表示が表示される。BODY要素直下のテキストに対して修飾処理を行う場合、画面全体がリバース文字表示となり、大変見にくくなる。また、HTML文への文字列置換によるテキスト処理では、置換文字列がHTML文に複数含まれる場合、どの文字列が対象となっているか、判定が困難になることも考えられる。HTMLはDOMを有し、HTML文はHTML-DOMに展開される。HTML文が持っている文書構造に基づき解釈並びに処理によって、このリバース表示が解消できる。

```
<p id="A">abcdef</p>
      var str=A.innerHTML;
  ↓   A.innerHTML="ab<span>cd</span>ef";
<p id="A">ab<span>cd</span>ef</p>
```

図3 innerHTMLによるテキストノード修飾

本稿ではXPointerで定義されたリソースを開始リソースと明示するため、HTML-DOMをappendChild並びにinsertBefore関数[MSDN2]によって操作することで、直接テキストノードの全部または一部をリンクポイントの修飾を行った。この修飾では、図4に示すように対象となるテキストを含むテキストノードが単独で親要素中にあるケースと、このテキストノードが弟要素を持つケースで異なる処理を設けなければならない。

テキストノードしかないケース

```
<p>abcdef</p>
          ↓   appendChild()
<p>abcdef<span>cd</span></p>
          ↓   appendChild()
<p>abcdef<span>cd</span> ef</p>
          ↓
<p>ab<span>cd</span>ef</p>
```

テキストノードの後ろに弟要素があるケース

```
<p>abcdef<br/>・・・</p>
          ↓   insertBefore()
<p>abcdef<span>cd</span><br/>・・・</p>
          ↓   insertBefore()
<p>abcdef<span>cd</span> ef<br/>・・・</p>
          ↓
<p>ab<span>cd</span>ef<br/>・・・</p>
```

図4 HTML-DOM操作によるテキストノード修飾の過程

まず前者のケースでは次の手順でテキストノードへのリンクポイント修飾を行う。①テキスト"cd"を持つSPAN要素を生成し、この要素をP要素の子供要素として追加する。②テキスト"ef"を持つテキストノードを生成し、このノードをP要素の子供ノードとして追加する。③テキスト"abcdef"を持つテキストノードの値にテキスト"ab"を代入する。以上の処理によってテキスト"cd"をSPAN要素のテキストとすることができる。

次に後者のケースについて、前者と同様の手順を試みたところ、この例ではP要素の最後尾にSPAN要素とテキストノードが追加され、元の文書構造を大きく損なうものとなった。そこで、弟要素を持つテキストノードへのリンクポイント修飾は、次の手順で行った。①テキスト"cd"を持つSPAN要素を生成し、この要素をBR要素の前に挿入する。②テキスト"ef"を持つテキストノードを生成し、このノードをBR要素の前に挿入する。③テキスト"abcdef"を持つテキストノードの値にテキスト"ab"を代入する。以上の処理によって後者のケースにおいてもリンクポイントの修飾ができた。これ以外のケースとして、テキストノード全体をリンクポイントとする場合、appendChildまたはinsertBeforeによってSPAN要素を生成し、元のテキストノードを削除することで、リンクポイントを生成できる。このような手順によって生成したリンクポイントは、同様の方法によって元のテキストノードに戻すことも可能である。SPAN要素にはリソースを効率よく探すため、id属性にリソースのラベル情報を代入した。

出来上がったSPAN要素にマウスアクションを加えるため、onmouseover、onmouseout、onclick（ここではonmouseupを採用）に対応するJavaScriptの関数を埋め込んでリンクポイントに機能を与えた。関数の埋め込み

は、SPAN 要素のノードを node とし、onmouseover に対する JavaScript 関数を MouseOver（例えば、このテキストにマウスがフォーカスした時、文字を赤色にする関数）とすると、node.onmouseover =MouseOver;によって当ノードに対して JavaScript 関数の付加処理ができる。また利用者にリンクポイントを明示するため付加情報として、その箇所の背景色を変えたり(node.style. backgroundColor="#cfcfcf";)、A 要素をまねてアンダーラインを引いたり(node.style. textDecoration="underline";)することも必要で、この制御は CSS のスタイル属性を操作して行った。

### 2．4 スクリプトの埋め込み

リンクポイントを実装しても、それを実施するスクリプト（リンク制御 API）がなければ処理が実施できない。通常のページをリソースとして XLink による連鎖を実施するためには、そのページに対して動的にリンク制御 API を埋め込むことが必須となる。通常のページへのリンク制御 API の埋め込みは次のように行った。

図 6 にあるようにリンク制御 API はフレーム menu 上の xlink.html 中の属性 id の値に"menu"を持つ SCRIPT 要素の中にある。即ち、menu.document.scripts["menu"].text によってリンク制御 API が取り出せる。このデータを stext とする。次にフレーム doc に表示されたページの document.documentElement によって得たルートから HEAD 要素を取得する。HEAD 要素がなければ、HEAD 要素をルートノードに付加する。この HEAD 要素に新たな SCRIPT 要素を追加し、このノードを snode とする。snode.text=stext によって、リンク制御 API はフレーム doc に表示されているページに追加される。

### 2．5 XLink 実装の手順

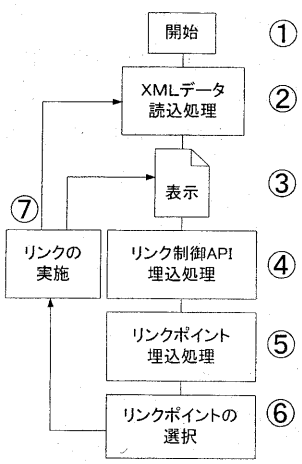


図5 XLink実装のフロー

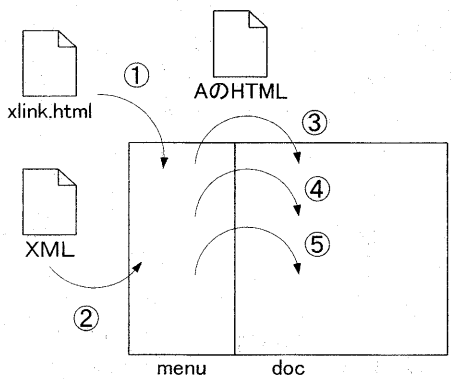


図6 ブラウザ上でのXLink実装

XLink の実装手順は図 5 に示し、処理説明のため図 6、7 を用意した。これらの図中の丸番号は以下の丸番号と対応する。①ブラウザ表示に開始するに当たり、XLink 全体を制御する XLinkAPI を有する xlink.html をフレーム menu に読み込む。XLink によって表示されるドキュメントはフレーム doc とし、この 2 つのフレームの表示比率は 0%と 100%とする。この指定によりフレーム menu に表示される xlink.html は全く見えない状態となる。XLinkAPI の構成は、DOM を制御する dom.js、XLink を制御する xlink.js、リンク制御 API と称し画面上でマウスの動きを制御する xcontrol.js、リソースに記述されている XPointer/XPath を解釈しリンクポイントを生成する xpointer.js から成り[添付資料 1]、XLink に関する制御のすべてをこのフレーム menu で行う。xlink.html に

指定されているXLink情報が格納されているXMLファイルをxsrcとする。②xsrcを読み込み、リスト1に示したリソース情報を配列href、配列label、配列titleに格納する。先頭リソースをsrcとする。③srcが示すリソースをフレームdocに表示する。この表示が完了しなければ次の処理が行えない。そこでHTMLの読込完了を次のように行った。読込完了確認は、タイマーを起動しdoc.document.readyStateが"complete"になるまで100msec間隔でチェックを行った。読み込み完了後、リスト2に従い当該リソースに含まれるトラバースリストを求める。④フレームdocに表示されているページのHTML-DOMに対してリンク制御APIを新しいSCRIPT要素として追加する。⑤トラバースリストに従い、当該HTML上にリンクポイントを埋め込む。図7に示すように、リソースBのリソース実体"a.html#abc"に対してid="abc"を持つ要素の内側に、このリソースのラベル情報をid属性に持つSPAN要素を設ける。またリソースBのxlink:titleをtitle属性とすると、"画像表示"と言う吹き出しが表示される。⑥ブラウザ上でこのリンクポイントをクリックすると、マウスイベントが生じたノード（ここではid="B"を持つSPAN要素）のid属性の値Lを求める。この値は⑤で述べたようにリソースのラベル情報である。このLに従いリスト2から該当するトラバースリストを求め、メニューを作成する。"明示されたリンクポイント"はこのような操作になるが、"明示されていないリンクポイント"では次のような処理となる。マウスイベントが生じると、現在表示のURLを取得し、配列hrefと対応するリソースのラベル情報を求める。このラベル情報に基づきリスト2から該当するトラバースリストを求め、メニューを作成する。メニューの各項目には図7の⑥に示すようにSPAN要素で記述されている。この要素のテキストはリソースDのxlink:titleの値を、showやhref属性も対応するリソースDの属性を、title属性は任意でxlink:titleの値でもxlink:hrefの値でもよい。またこのSPAN要素のid属性は"xlink"とする。またこれらのメニュー項目をDIV要素でくくり、これをメニューオブジェクトとする。このオブジェクトに対して、マウスイベント関数を定義し、メニュー項目の選択時のマウスアクションに対応させる。このような処理を加えたメニューが表示される。⑦メニュー中の項目を選択するとマウスイベントが生じる。もしイベントを生じた要素のid属性が"xlink"ならば、その要素のshowとhref属性の値を取得し、showが示す表示モードに従いhrefの値をsrcとし、③の処理へ戻る。もしshowが示す値が"delete"ならば、hrefが示す関数を実行し、メニュー選択が継続する。またhrefが示す値が"xml"であった場合、hrefの値をxsrcにセットし、②の処理へ戻る。②へ戻る処理はlinkbaseの実装に他ならない。

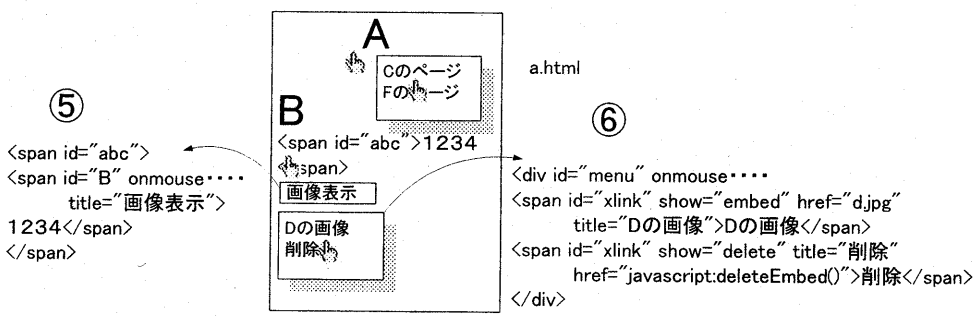


**図7 リンク機構の埋め込み**

上記に示した処理はすべてブラウザ上のJavaScriptに記述した。これらの処理は、先に述べたXLink実装の基本要件をすべて満たすもので、XLinkで記述されたXMLに従って、その中に記述されている範囲内でリンクの連鎖が確認できた。またブラウザ上に表示されているHTMLに対して、リンクポイントの生成、XLinkを動かすJavaScriptの埋め込みなどの付加処理ができるようになり、CD-ROM中にあるHTMLのような書き込み不可のリソースもXLink対象のリソースとすることができた。

## 3 実装したXLinkのテスト

実装したプログラムを評価するため、図8に示すように7個のリソースと17個のアークを持つXLinkを用意し、この範囲でXLinkの実装テストを行った[大坂2]。図8において、リソースGではリソースFへのリンクしかない単一リンクが、リソースEではリソースA、B、D及びGの4方向へのマルチリンクが、また至るところで双方向リンクが見られ、厳しい実装テストができるものとなっている。またこのテストでは、“明示されていないリンクポイント”によるリンクの連鎖が中心となる。

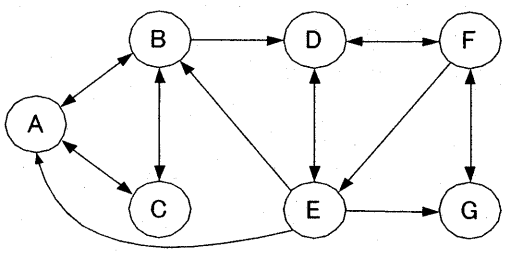


図8 XLinkのテスト

テストした結果、マウスクリックによって当該リソースからトラバースされるリソースがメニュー表示される。メニュー中の項目を選択すると、選択したリソースが表示され、XLinkファイルに記述されている範囲で、XLinkが正しく連鎖・作動することを確認した。このテストで準備したリソースには、元々HTML文中にA要素によるリンクが含まれている。これら要素からリンクされたリソースへのXLink対応は未実装で、ここでXLinkの連鎖が途絶えてしまう。XLinkで定義されたリンクの島からA要素によって別のリンクの島へリンクを連鎖させる処理は、今後の課題である。

## 4 結果と今後の展望

今回ブラウザ上においてJavaScriptで記述したXLinkAPIによってXLinkを実装し、良好で満足する結果を得た。実装に要したJavaScriptの総コード量は、添付資料1に示すようにわずか1350ステップであった。ここで作成したXLinkAPIによって、XLinkの拡張リンク、第三者リンク、プレーンなHTMLへの動的なリンクの埋め込み、XLinkによるリンクの連鎖などのXLinkのフルスペックが実装できた。しかし、XLinkのテストを行う中で、大きな制約にぶつかった。この制約はJavaScriptのセキュリティモデルによるもので、XLinkAPIを備えているドメインにあるHTML、XMLなどのファイルしかリソースすることができず、しかもこのドメイン以外にあるファイルをリソースとして扱うことができない。具体的に表現すると、ここで作成したXLinkAPIは、例えばhttp://www.w3.org/にあるファイルの中身をアクセスできない。現状では、いずれにせよ、自分が管理している範囲内でしか、XLinkを作動できない状況にある。

我々はこれまで複合文書をテーマについて研究を行ってきた[大坂、野村]。特にHTMLと異なるDOMを有するSVG(Scalable Vector Graphics)をHTMLの中に埋め込んだ複合文書の生成で、HTML-DOM、XML-DOM、SVG-DOMを積極的に制御してインタラクティブな複合文書を実現したもの[大坂1]で、本稿を導く基礎となっている。これらの研究は、複合文書の生成方法に注目したもので、埋め込まれるリソースは静的にバインディングされていた。

XLinkには振る舞い属性xlink:showがあり、この属性の値に”embed”を指定すると、指定されたリソースの埋め込み表示が指示できる。この埋め込み表示はそのまま複合文書の生成を意味するものである。XLinkの効果は、埋め込まれる側、埋め込む側のリソースを第三者リンクによって記述できること、XPointer/XPathのメカニズムよって対象コンテンツの一部分をリソースと定義できることが挙げられる。XLinkを積極的に利用することにより、ダイナミックな複合文書の生成が可能となる。XLinkの可能性を追求すると、何もないHTMLをリソースとし、このHTMLに対して複数のリソースを埋め込みするリンクを定義することで、これまでにない複合文書が作成できるかも知れない。またブラウザサイトにおけるXLinkのリンク機構も有効であるが、サーバーサイトにおけるXLinkの利用は新たな可能性も秘めていると思われる。

## 参考文献


[DuCharme2001] "Building XLink Applications with XSLT"、XML Conference & Exposition 2001、
http://www.idealliance.org/papers/xml2001/papers/html/06-02-01.html、2001

[DuCharme2002] "XLink: Who Cares?"、http://www.xml.com/pub/a/2002/03/13/xlink.html、2002

[MSDN1] "About Dynamic Content"、http://msdn.microsoft.com/workshop/author/dyncontent/content.asp

[MSDN2] "About the W3C Document Object Model"、http://msdn.microsoft.com/workshop/author/dom/domoverview.asp

[Williams] "データのための XML:XLink とデータ"、
http://www-6.ibm.com/jp/developerworks/xml/011214/j_x-xdxlnk-index.html、2001

[XInclude] "XML Inclusions (XInclude) Version 1.0 W3C Candidate Recommendation 21 February 2002"、
http://www.w3.org/TR/xinclude/、2002

[XLink] "XML Pointer, XML Base and XML Linking"、http://www.w3.org/XML/Linking、2001

[奥井、山本] "XLink、そして Xpointer/Xpath の威力"、JavaWorld, April 2002

[大坂、野村] "SVG-DOM によるアニメーションと XHTML 中心複合文書の可能性"、情報処理学会研究会報告、DD-32、3/2002

[大坂 1] " SVG、XHTML 中心の複合文書 異なる DOM 空間の操作"、XML コンソーシアム Week、6/2002

[大坂 2] " XLink について 付箋紙を XLink で動かしてみる"、XML コンソーシアム Week、6/2002

[中山、奥井] "XML 完全解説下" 、技術評論社、ISBN4-7741-1302-6、2001

[富士通] "XLink Tutorial"、http://www.labs.fujitsu.com/free/HyBrick/tutorial/index.html、1999


## 添付資料 1 プログラム構成

JavaScript による XLinkAPI は下記のプログラムから構成され、総ステップ数は 1350 ステップである。

| | | |
|---|---|---|
| dom.js | 170 ステップ | DOM を制御するプログラム。 |
| xlink.js | 660 ステップ | XLink(470steps)並びに埋め込み表示(190steps)を制御するプログラム。 |
| xcontrol.js | 280 ステップ | リンク制御 API と称し、画面上でマウスの動きを制御するプログラム。このプログラムは HTML 中に直に記述され、単独では存在しない。 |
| xpointer.js | 240 ステップ | リソースの XPointer/XPath を解釈し、リンクポイントを生成するプログラム。但し、XPointer はベアネームと string-range 関数への対応のみの、簡易的な実装となっている。 |

## 開発並びに作動確認環境

OS : Windows2000、WindowsNT、WindowsXP

Browser : IE5.0、IE5.5、IE6.0