複数のドメイン上の既存Webサイトに新たな機能を追加する手法

坂入 隆　大湖 卓也　牟田 英正

A Method to Add New Functions to Existing Websites on Multiple Domains

Takashi Sakairi　Takuya Ohko　Hidemasa Muta

既存の複数のドメイン上のWebサイトにアクセシビリティを高めるなどの新たな機能を追加するという要求がある. これらの要求に対して, 現在, 提案されている解決策は, ActiveXのような特定のクライアント環境でのみ動作する技術を用いている. 本論文では, ダイナミックHTMLを用いて, 複数のクライアント環境で動作する手法について述べる.

There are demands to add new functions such as improving accessibility to existing websites on multiple domains. Currently proposed solution to this demand is to use technologies such as Active X that could only run on a specific client environment. This paper discusses the solution that works on multiple client environments by use of dynamic HTML.

Key Words & Phrases：アクセシビリティ, Webサイト, ダイナミックHTML, JavaScript, クッキー
accessibility, Website, dynamic HTML, JavaScript, cookie

1. はじめに

現在ではWebは, 業務を遂行したり日常生活を送ったりする上で, なくてはならないものとなっている. 情報の伝達および電子商取引などの様々な処理のために, 多くのWebサイトが構築されている. これら既存のWebサイトから発信されるWebコンテンツに様々な機能を追加し, 利用者固有の要求を満たすサービスへの期待が高まっている.

既存のWebサイトに追加したい機能として代表的なのが, アクセシビリティ［1, 2］を高めるための機能である. 画面の拡大, 色の組み合わせの変更, 音声での読み上げといった機能を追加することによって, 視覚障害者や高齢者のアクセシビリティを高めることができる.

その外にも, 既存のWebサイトに追加すると便利な機能として, 例えば次のようなものが考えられる.
(a) 子供向けに漢字に振り仮名を表示する.
(b) 頭文字だけをとった略語を展開する.
(c) 外国語を翻訳する.

既存Webサイトに新たな機能を追加するのが利用者の場合［3］と, Webサイト管理者の場合［4, 5］とでは, 対処のための要件が以下のように異なる. 利用者が追加する場合は, 任意のWebサイトに新たな機能を追加することができるが, 利用者が自分でアプリケーションを購入し, 導入するという作業を行う必要がある. Webサイト管理者が追加する場合は, 特定のWebサイトでしか使えないが, 情報提供者であるWebサイト管理者がサービスの費用を負担することになり, 利用者にとって費用や導入の手間などの負担が少ない. 公共性の高いWebサイトでは, 利用者に大きな手間をかけることなく, アクセシビリティを高めることが望まれている. また, 企業などのWebサイトでも, 利用者の利便性を高めることは重要である. 本論文では, Webサイト管理者が追加する場合を扱う.

Webサイト管理者が既存Webサイトに新たな機能を追加しようとする場合に, 単独のWebサイトに対してだけ機能を追加するとは限らない. 複数のWebサイトに対して同時に機能の追加を行いたい場合がある. また, これらの複数のWebサイトが同一のドメインに存在するとは限らない. 例えば, 地方自治体が既存Webサイトにアクセシビリティの機能を追加しようとすると, その地方自治体が運用しているWebサイトだけでなく, その地方自治体が出資している第三セクターが運用しているWebサイトなど多くの関連Webサイトが対象になるが, これらのWebサイトは別のドメインとなる. また, 企業グループの場合でも, それぞれの企業が運用しているWebサイトは別のドメインとなる場合がある.

Webサイト管理者が既存Webサイトに新たな機能を追加する手法には, 大きく分けて次の2通りある.
① 既存のHTMLをサーバー側で変換する手法［6, 7］

提出日：2005年8月31日　再提出日：2005年12月7日

② 既存のHTMLをサーバー側で変換しない手法[4, 5]

①の手法では, 画面の拡大といった単純なことだけではなく, 携帯電話向けにCHTMLやWMLに変換したり, イメージファイルを縮小したりといったことも可能である. しかし, 変換のためのサーバーが必要になり大掛かりなものになる. ②の手法では, 既存のHTMLを読み込んだWebブラウザのDOM [8] をActiveXやJavaScriptで操作することになる. そのため, HTMLを変換する手法よりもできることが限られるが, サーバーの負荷がほとんどなく, 多くの一般的なWebサーバーに対応できるという利点がある. 本論文では, ②の手法を扱う.

既存のHTMLを読み込んだWebブラウザのDOMをActiveXを用いて操作することが可能である. しかし, ActiveXを用いると, クライアント環境としてWindowsとInternet Explorerの組み合わせに限定されてしまう. JavaScriptを用いてDOMを操作すると, 多くのクライアント環境で利用することが可能である. しかし, Webブラウザの別のフレームやウィンドウを制御することは, Webブラウザのセキュリティの制限により, そのフレームやウィンドウに表示されているWebページが同じドメインに存在する場合しか許されていない. そのため, 従来の技術では, 複数のドメイン上の既存Webサイトに統一的に新たな機能を追加するためには, クライアント環境に固有な技術を使わなければならないという問題がある.

また, このような新たな機能をセッションを越えて利用するためには, 表示倍率や配色などのパーソナライズのための情報を保存する必要がある. このような情報は個人情報であるので, 可能ならばサーバー側に保存しないことが望ましい. Webブラウザがこのような情報を保管する場合には, クッキー [9] を用いることになる. しかし, クッキーを用いるとドメインの異なるWebサイト間で情報を共有することができないという問題がある.

本論文では, これらの問題を解決する手法を提案する. 以下では, まず, ダイナミックHTMLを用いて既存Webサイトに新たな機能を追加する基本的な手法について説明する. 次に, 我々が提案する複数ドメインの問題を考慮した拡張について述べる. 最後に, 本論文をまとめる. 本論文の手法では, クライアント環境としてWindowsとInternet Explorerの組み合わせに加えて, LinuxやWindowsとMozillaの組み合わせなどを使うことが可能である.

2. 基本となる手法

2.1 ダイナミックHTML

Webブラウザ上に読み込まれたHTMLは, DOM [8] と呼ばれる木構造で表現される. JavaScriptを用いると, このDOMに対して構造や属性を変えるなどの様々な操作を行うことができる. HTMLのDOMを操作してWebブラウザ上の表示を変化させることは, ダイナミックHTMLと呼ばれている. WebブラウザによってJavaScriptからDOMを操作するAPIに違いがあるが, 注意深くプログラミングすることでAPIの違いを吸収し, 複数の種類のWebブラウザで動作するJavaScriptを作成することが可能である.

既存のWebサイトのWebページをWebブラウザのフレームに読み, そのフレーム内のHTMLのDOMをJavaScriptで操作すれば, 様々な新たな機能を追加することができる. 例えば, マウスポインタの指す位置にある要素を反転し, さらに別のフレームに拡大して表示するといったことができる(**図1**).

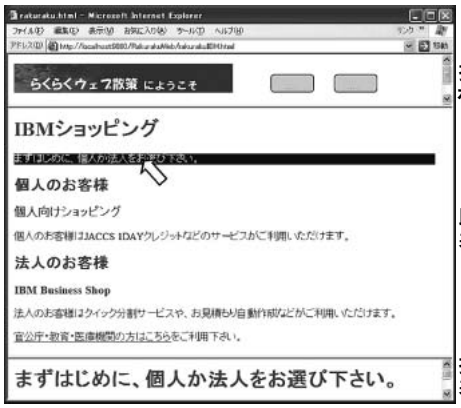

図1. 文字列を拡大する例

既存Webサイトを表示するフレームのページが遷移するたびに, そのページの読み込みが終了した時点で初期処理を行う必要がある. この初期処理は, FRAME要素のonloadイベントハンドラで設定する. HTML 4.01の仕様 [10] では, onloadイベントハンドラは, BODY要素とFRAMESET要素に対してしか設定できないことになっているが, Internet ExplorerでもMozillaでもFRAME要素に設定できるように拡張されている. これらのWebブラウザがバージョンアップされるときには, 互換性を考慮されるはずなので, これらのWebブラウザでは今後ともFRAME要素のonloadイベントハンドラを利用できるはずである.

既存Webサイト自体がフレームを利用している場合には, そのままではフレーム内でページを遷移したときに, 初期処理を呼び出すことができない. 事前にフレームセットを定義するWebページを修正し, FRAME要素のonloadイベントハンドラを設定しておく必要がある. ただし, ほとんどのWebサイトでは, フレームセットを定義するWebページの数は多くないので, 大きな作業にはならない.

初期処理では, onmouseoverやonmouseoutなどのイベントハンドラを設定するなどの処理を行う. このことにより, 既存Webサイトを表示するフレーム上で利用者がマウスポインタを移動したときにDOMを操作することができるようになる.

また, 追加する機能によっては, 初期処理でDOMの構造を変更する必要がある. 文単位で文字を拡大するという機能とTRLのような頭文字をとった略語をTokyo Research Laboratoryのように展開するという機能の2つの機能を考える. onmouseoverやonmouseoutなどのイベントハンドラを用いて, 文や略語の上にマウスポインタが移動したり文や略語の上からマウスポインタが離れたりしたときに, そのことを検知するためには, 文や略語が独立したHTMLの要素となっている必要がある.

そのためには, 文や略語を値として持つSPAN要素を追加するようにDOMの構造を変更すれば良い. SPAN要素には, どのような目的で追加したものなのかをclass属性で示しておく. 例えば, 文のSPAN要素にはsentenceというclass属性を, 略語のSPAN要素にはacronymというclass属性を設定する.

2.2 ドメインの問題

既存Webサイトを表示するフレームで, 別のドメイン上のページに遷移すると, 初期処理を行うイベントハンドラを実行することができない. これは, セキュリティ上の問題があるので, 別ドメインのページのDOMを操作できないように, Webブラウザが制限しているためである.

ドメインをJavaScriptで設定することにより, ドメインの範囲を広げることができる. 例えば, w3.trl.ibm.comとw3.watson.ibm.comのすべてのWebページに,

```
document.domain = "ibm.com";
```

というJavaScriptを埋め込めば, それらのすべてのWebページは同じドメインとして扱われ, お互いにJavaScriptでDOMを操作することができる. HTMLのDOMの仕様[9]では, document.domainは変更できないことになっているが, Internet ExplorerでもMozillaでもこの方法により, 変更することが可能である. ただし, この方法には, 次に述べる2つの欠点があるため, 本論文では採用していない.

1つ目は, Webサーバーに負荷がかかってしまうという欠点である. なぜなら, この方法では, 既存のすべてのWebページをWebサーバー側で変換しなければならないからである.

2つ目は, 地方自治体と第三セクターのWebサーバーや企業グループ内のそれぞれの企業のWebサーバーを同じドメインとして扱うことができないという欠点である. なぜなら, この方法では, ibm.comのようにドメインの第2レベル(ibm.co.jpなどの場合は第3レベル)が共通でなければならないからである.

3. 複数ドメイン上のWebサイトへの拡張

3.1 別のドメイン上のWebページへの遷移

2.2節で述べたように, 基本となる手法を単純に適用しただけでは, 新たな機能を利用しているときに, 別のドメイン上のWebページへ遷移するとその機能が利用できなくなってしまう. 本節では, このようなときにも利用者に負担を掛けることなく同じ機能を利用できるように, 前章の手法を拡張する.

新たな機能を統一的に追加しようとする複数のWebサーバーの中から1つを選び, このWebサーバーに特別な役割を持たせる. 以下では, このWebサーバーを主Webサーバーと呼び, それ以外のWebサーバーを従Webサーバーと呼ぶ(**図2**).

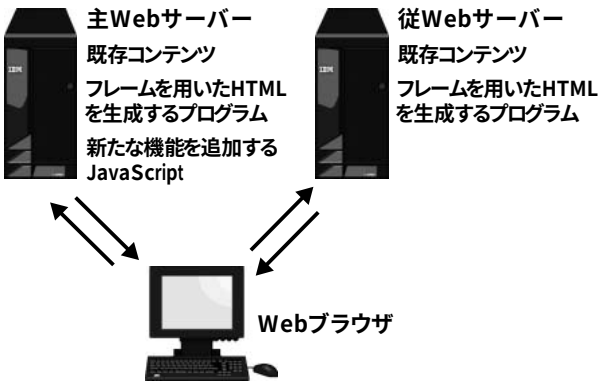

図2. システムの構成

すべてのWebサーバーには既にコンテンツが存在している. これらのコンテンツを変更することなく, Webサーバーで実行されるプログラムとWebブラウザで実行されるJavaScriptを追加する. Webサーバーで実行されるプログラムは, フレームを用いたHTMLを生成するものであり, 同じ機能を持つものをそれぞれのWebサーバーに導入する. Webブラウザで実行されるJavaScriptは, アクセシビリティなどの新たな機能を追加するものであり, 主Webサーバーにのみ配置する.

以下では, 主Webサーバーをw3.trl.ibm.com, 従Webサーバーをw3.watson.ibm.comとw3.almaden.ibm.comとして説明する.

図3にフレームを用いたHTMLを生成するプログラムのJSPによる実装の例を示す.

7行目のJavaScriptは主Webサーバーに置かれているものを参照している. 11行目で既存コンテンツを表示するFRAME要素のonloadイベントハンドラが設定

```
1 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
2 <HTML>
3 <HEAD>
4 <%@ page language="java" contentType="text/html; charset=SHIFT_JIS" pageEncoding="SHIFT_JIS" %>
5 <META http-equiv="Content-Type" content="text/html; charset=SHIFT_JIS">
6 <TITLE>rakuraku.jsp</TITLE>
7 <SCRIPT type="text/javascript" src="http://w3.trl.ibm.com/RakurakuWeb/rakuraku.js"></SCRIPT>
8 </HEAD>
9 <FRAMESET rows="20%, *, 30%, 0%">
10   <FRAME src="controller.html" name="controllerFrame">
11   <FRAME src="<%=request.getParameter("href") %>" name="contentsFrame" onload="init(contentsFrame)">
12   <FRAME src="magnifier.html" name="magnifierFrame">
13   <FRAME name="personalizedDataFrame">
14   <NOFRAMES>
15   <BODY>
16   <P>This page uses frames. The current browser you are using does not support frames.</P>
17   </BODY>
18   </NOFRAMES>
19 </FRAMESET>
20 </HTML>
```

図3.　フレームを用いたHTMLを作成するJSPの例

```
1 var RAKURAKU_JSPS = new Object();
2 RAKURAKU_JSPS["w3.trl.ibm.com"] = "http://w3.trl.ibm.com/RakurakuWeb/rakuraku.jsp";
3 RAKURAKU_JSPS["w3.watson.ibm.com"] = "http://w3.watson.ibm.com/RakurakuWeb/rakuraku.jsp";
4 RAKURAKU_JSPS["w3.almaden.ibm.com"] = "http://w3.almaden.ibm.com/RakurakuWeb/rakuraku.jsp";
```

図4.　ドメインごとに新しい機能を追加するURLを登録するJavaScriptの例

されている．コンテンツの指定には，hrefというパラメータでURLを渡すことがsrc属性に示されている．

図4にドメインごとに同じ機能を追加するためにフレームを用いたHTMLを作成するURLを登録するJavaScriptの例を示す．これは，図3の7行目で指定されているJavaScriptの一部である．

例えば，

http://w3.trl.ibm.com/news.html

に新たな機能を伴ってアクセスするには，

http://w3.trl.ibm.com/RakurakuWeb/rakuraku.jsp?
href=http://w3.trl.ibm.com/news.html

と指定すれば良い．

前章で述べた既存Webページを読み込んだときの初期処理で次の処理を追加する．まず，そのWebページ内のすべてのA要素のhref属性とFORM要素のaction属性を調べる．次に，これらの属性が現在のドメインではないが同じ機能を追加するWebサーバーのドメインであれば，属性を新たな機能を伴ってアクセスするものに変え，target属性も_topに変える．このことにより，事前に定義しておいた別のドメイン上のWebページに遷移しても，同じ機能を伴ってアクセスすることができる．

既存コンテンツに，JavaScriptを用いてページの遷移が記述されている場合がある．そのようなときには，上述のA要素とFORM要素の変換のように簡単には変換することはできない．JavaScriptを用いたページの遷移を止めるか，JavaScriptを修正する必要がある．

3.2 ドメイン間のクッキーの共有

1章で述べたように，ドメイン間でクッキーを共有するためには工夫が必要である．Webブラウザのセキュリティの制限により，Webページから別のドメインが管理しているクッキーを読んだり書いたりすることはできない．

我々は，クッキーを管理するための大きさ0のフレームを用いてこの問題を解決した．図3の13行目のフレームがそれである．以下では，このフレームをクッキー管理フレームと呼ぶ．また，この図の10行目のフレームを追加機能フレームと呼ぶ．

(a) クッキーの読み込み

図5に従Webサーバー上のWebページから主Webサーバーが管理するクッキーを読み込む手順を記す．

手順1. 追加機能フレームは，主Webサーバーに対して，クッキーを読むJavaScriptを含むHTMLを要求する．このとき，ターゲットとしてクッキー管理フレームを指定する．また，パラメータと

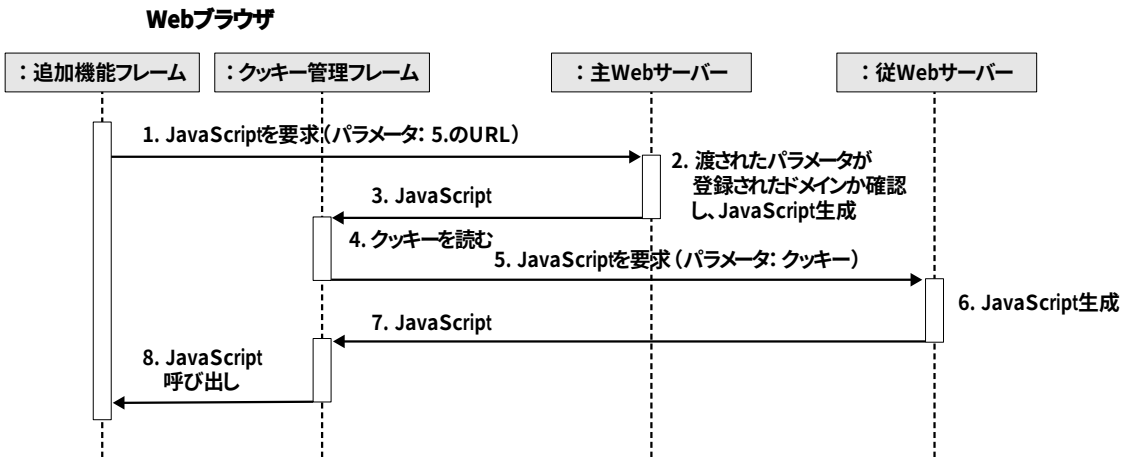

図5.　クッキーを読み込む手順

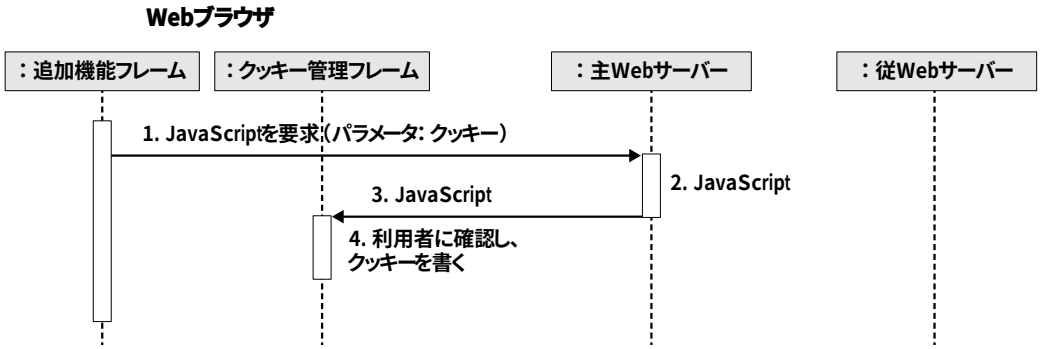

図6.　クッキーを書き出す手順

して, 手順5のあて先となるURLを渡す.

手順2. 主Webサーバーは, 渡されたURLが事前に登録されたドメインのものであるかを確認し, 登録されたものであれば, クッキーを読むJavaScriptを含むHTMLを生成する.

手順3. 主Webサーバーは, クッキー管理フレームに対して, 生成したJavaScriptを含むHTMLを送る.

手順4. クッキー管理フレームは, 受け取ったJavaScriptを実行することにより, 主Webサーバーが管理するクッキーを読む.

手順5. クッキー管理フレームは, 従Webサーバーに対して, 追加機能フレームへ読み込んだクッキーを知らせるためのJavaScriptを含むHTMLを要求する. このとき, パラメータとして, 読み込んだクッキーを渡す.

手順6. 従Webサーバーは, 追加機能フレームへ読み込んだクッキーを知らせるためのJavaScriptを含むHTMLを生成する.

手順7. 従Webサーバーは, クッキー管理フレームに対して, 生成したJavaScriptを含むHTMLを送る.

手順8. クッキー管理フレームは, 追加機能フレームのJavaScriptを呼び出すことによって, 読み込んだクッキーを知らせる.

(b) クッキーの書き出し

図6に従Webサーバー上のWebページから主Webサーバーが管理するクッキーに書き出す手順を記す.

手順1. 追加機能フレームは, クッキー管理フレームに対して, クッキーを書くJavaScriptを含むHTMLを要求する. このとき, ターゲットとしてクッキー管理フレームを指定する. また, パラメータとして, 書き出すクッキーの値を渡す.

手順2. 主Webサーバーは, ダイアログボックスを出して利用者に確認後にクッキーを書くJavaScriptを含むHTMLを生成する.

手順3. 主Webサーバーは, クッキー管理フレームに対して, 生成したJavaScriptを含むHTMLを送る.

手順4. クッキー管理フレームは, 受け取ったJavaScriptを実行することにより, ダイアログボックスを出して利用者にクッキーの書き出しを確認する. 利用者から確認が得られたら, 主Webサーバーが管理するクッキーを書く.

手順2の中で, どのWebサイトからの要求であるのかを正確に知ることができれば, 手順4で利用者に確認を取る必要がない. そのために, HTTPリクエストのrefererフィールド［11］(注: このフィールドは, Webブラウザが HTTPリクエストを出すときに表示しているURLを示す. 英語としてはreferrerが正しいが, HTTP

リクエストのフィールド名としてはrefererを用いる.）を使うということが考えられる. しかし, refererフィールドは, Webブラウザの設定やファイアウォールの設定によっては, 正しくサーバーに送られない. そのため, 手順2の中でこのフィールドを用いてJavaScriptを生成すべきかどうか判定することはできない.

4. おわりに

本論文では, 既存Webサイトをほとんど変更することなく, アクセシビリティを高めるなどの新たな機能を追加する手法を提案した. この手法では, JavaScriptを用いているために, クライアントの環境が特定のOSやWebブラウザに依存しないようにできる. また, 複数のドメイン上に存在する関連するWebサイト間でページを遷移したときにも, 利用者に負担をかけることなく, 新たに追加した機能を利用できるようにすることができる. さらに, 複数のドメイン上の関連するWebサイト間でパーソナライズのための情報をサーバーに保管することなく共有することができる.

本論文で提案している手法の制限として, 既存コンテンツにフレームを使っている場合には, フレームセットを定義するWebページを修正する必要がある. ただし, ほとんどのWebサイトでは, フレームセットを定義するWebページの数は多くないので, 大きな作業にはならない. また, JavaScriptを用いてページの遷移を記述している場合にも, 修正する必要がある. ただし, JavaScriptを使わないようにWebブラウザを設定している場合にも正しくページ遷移するように考慮されたWebサイトでは, JavaScriptを用いたページ遷移は多く用いられていないと考えられる.

東京基礎研究所では, らくらくウェブ散策[4, 5]というインターネット閲覧支援ソフトウェアを2002年に開発した. らくらくウェブ散策は, 既存Webサイトに修正を加えることなく表示倍率や配色を変えたり, テキストを読み上げたりといった機能を持ち, 官公庁や民間企業のWebサイトで採用されている. ただし, 現在リリースされているものは, ActiveXを用いて実装されているため, クライアント環境としてWindowsおよびInternet Explorerが対象となっている. 本論分の手法により, らくらくウェブ散策を様々なクライアント環境で稼動させることが可能となる. 我々は, この手法をらくらくウェブ散策に応用し, 実環境においてこの手法の有効性を評価していく予定である.

参考文献

[1] World Wide Web Consortium, Web Accessibility Initiative（WAI）, http://www.w3.org/WAI/, 2005.8.30.

[2] M. Rowan et al., Evaluating Web Resources for Disability Access, *Proceedings of 4th ACM Conference on Assistive Technologies*, pp. 80-84, 2000.

[3] C. Asakawa et al., User Interface of a Home Page Reader, *Proceedings of 3rd ACM Conference on Assistive Technologies*, pp. 149-156, 1998.

[4] H. Muta et al., An Active-X-based Accessibility Solution For Senior Citizens, *Proceedings of CSUS's 20th Annual International Conference*, 2005.

[5] 日本IBM, らくらくウェブ散策, http://www-6.ibm.com/jp/accessibility/soft/rakuraku.html, 2005.8.30.

[6] R. Barrett et al., Intermediaries: An approach to manipulating information streams, *IBM System Journal*, Vol. 38, No. 4, pp. 629-641, 1999.

[7] A. W. Huang et al., A Semantic Transcoding System to Adapt Web Services for Users with Disabilities, *Proceedings of 4th ACM Conference on Assistive Technologies*, pp. 156-163, 2000.

[8] World Wide Web Consortium, Document Object Model (DOM) Level 2 HTML Specification Version 1.0, *W3C Recommendation 09 January 2003*, http://www.w3.org/TR/2003/REC-DOM-Level-2-HTML-20030109/, 2005.8.30.

[9] D. M. Kristol, HTTP Cookies: Standards, Privacy, and Politics, *ACM Transactions on Internet Technology*, Vol. 1, No. 2, pp. 151-198, 2001.

[10] World Wide Web Consortium, HTML 4.01 Specification, *W3C Recommendation 24 December 1999*, http://www.w3.org/TR/1999/REC-html401-19991224/, 2005.8.30.

[11] Internet Society, Hypertext Transfer Protocol -HTTP/1.1, *Request for Comments 2616*, June 1999, http://www.ietf.org/rfc/rfc2616.txt, 2005.8.30.

日本アイ・ビー・エム株式会社
東京基礎研究所
主任研究部員

坂入 隆 Takashi Sakairi

［プロフィール］
1987年，日本アイ・ビー・エム株式会社入社．東京基礎研究所で，文書処理系，グループウェア，情報可視化，ソフトウェア開発支援系などの研究に従事．ACM，情報処理学会各会員．
sakairi@jp.ibm.com

日本アイ・ビー・エム株式会社
アクセシビリティセンター
副主任開発技術係

大湖 卓也 Takuya Ohko

［プロフィール］
1998年に日本IBM入社．Web関連製品，ストレージ管理関連製品などの開発を経験．2003年より東京基礎研究所にてインターネット閲覧支援ソフトウェア「らくらくウェブ散策」の開発に従事．
ohkot@jp.ibm.com

日本アイ・ビー・エム株式会社
アクセシビリティセンター
主任開発技術担当部員

牟田 英正 Hidemasa Muta

［プロフィール］
1991年，日本IBMに入社．OS/2，システムソフトウェア（Desktop On-Call），組み込みソフトウェア（Java AWTほか）などの開発を経て，2000年から障害支援ソフトウェア開発に従事．「らくらくウェブ散策」の設計・開発を担当．現在，UPnP/DNLA対応デジタル家電のユーザーインターフェースI研究に従事している．
hmuta@jp.ibm.com