M-096

# 実際的なページ操作の共有を実現したウェブブラウジング協調システム

# Collaborative Web Browsing System Realizing Sharing of Practical Page Operations

中村 大介† 疋田 輝雄†
Daisuke Nakamura Teruo Hikita

## 1. はじめに

インターネットが普及した社会でデジタルデバイドが問題となっている．インターネットを利用する人としない人での得られる情報の格差である．インターネット非利用者の利用しない理由は，よく分からない，使い方が難しいなどである．彼らは近くに教えてくれる人がいれば，そして習える場所があれば利用するという．このような問題の解決策として，協調システムによる遠隔補助が考えられる．利用者と教える人がウェブブラウザの画面を共有することで，インターネット利用時に距離の制限なく視覚的な補助を行なうことができる．

我々はウェブブラウザを用いてインターネットにアクセスする二人以上のユーザが，同一のウェブページを共有しながら閲覧できる協調システムを開発した．本システムでは一般的なウェブブラウザ上でウェブページを共有することができる．そのためセキュリティに関してはブラウザ本来のものを利用することができる．ユーザは特別のソフトウェアやプラグインをインストールする必要はない．機能としては，ウェブページ操作とユーザ操作がユーザ間で双方向に共有される．ウェブページ内のスクリプトや一部の動的な書き換えにも対応している．

本システムではウェブページを共有化し，共有サーバに一時的に保存することでウェブブラウジング共有システムを実現した．共有サーバは共有機能の制御も行なう．ウェブページを共有化するとは，ウェブページを取得し，共有が可能なように書き換えることである．ウェブブラウザと共有サーバの通信は DWR を利用したサーバプッシュ型の通信を行なう．

本論文では，2 節で関連研究とシステムについての考察を行なう．3 節ではシステムの構築形態について議論する．4 節では本システムの GUI と機能，仕様について説明する．5 節では同一生成元ポリシーについて考察する．6 節では本システムの通信部分で利用している DWR について説明する．7 節では本システムの実装について記述する．8 節で本論文のまとめを述べる．

## 2. 関連研究とシステム

ウェブブラウジングの共有を目的とした協調システムの実現では様々な方法が考案されており，技術の流れと共にその方法も変化している．

文献[10]は Java アプレットを利用してウェブベースの協調アプリケーション構築環境を開発している．ウェブブラウジングの共有機構を Java の API として実装し，Java アプレットを通じてブラウザ情報を共有する．文献[5]では専用クライアントを利用した協調アプリケーションを実現している．この文献の本来の目的はウェブブラウジングの共有ではないが，一方向共有機能をもつ専用クライアントをもつ．専用クライアントを構築すればウェブブラウジングの共有は比較的容易に実現できる．しかし専用クライアントのインストールが必要となるため，ユーザ側から見た初期導入の敷居は高くなる．

文献[9]では Thin クライアント方式の協調システム構築基盤を開発している．この方式では，Java アプレットや専用クライアントを利用する方法では難しかった，機能やデザインの再構築を簡単に行なうことができる．この文献の協調システム構築基盤における問題点を考察したのが文献[6]である．現実のシステムに即した問題点について丁寧に考察し，SSL に対応する構築形態を提案している．

Sync+[11]はプラグインを利用してウェブブラウザ同期を実現する．NetConcierge というサービスに技術が応用されており，実際に企業で利用された実績のあるサービスである．通信に P2P を利用している点が他のシステムとは大きく異なる．非常に多機能だが，IE 専用である．SSL, JavaScript, iframe には対応していない．

文献[3]はプロキシを利用してウェブページの書き換えを行なう方法で CWB と呼ばれる協調的なウェブブラウジングを実現している．通信にはポーリングを利用している．文献[1]はプロキシを利用した方法に加えて Java アプレットを利用することで，サーバプッシュの協調環境を実現している．また SSL にも対応しており，ページのレイアウトについて言及するなど，ウェブブラウジング共有に対する深い考察がされている．プロキシを利用した方法にサーバプッシュ技術を加えたものが文献[7]である．スムーズなウェブブラウジング共有が実現されているが，共有機能はウェブページ遷移だけである．

## 3. システムの構築形態

### 3.1 既存の構築形態

2 節で述べたように，ウェブブラウジング共有システムに関する研究は多数あるが，その多くがウェブベースの協調アプリケーション構築に関している．このような研究は遠隔教育や電子会議，コンタクトセンターなどの協調システムの構築や既存のウェブアプリケーションに共有機能をもたせるための再構築を目的としている．そのため，任意のドメインのウェブページを共有する必要がなく，共有は同一ドメイン内に限定される．共有を同一ドメイン内に限定したシステムの構築では，クライアントを中心とした構築形態とサーバを中心とした構築形態がある．

クライアントを中心とした構築形態では Java アプレッや専用クライアントを利用する方法がある．ウェブブラウジング共有システムのクライアントサイドに Java アプレットや専用クライアントを利用するのは，HTTP がクライアントからの要求に対して応答する形式であるため，

†明治大学 理工学研究科 Dept. of Computer Science, School of Science and Technology, Meiji University

一般的なウェブブラウザのままではサーバプッシュ型ウェブアプリケーションの構築が不可能なためである．しかし現在では一般的なウェブブラウザを利用できるサーバプッシュ型の技術が存在しているため，Java アプレットや専用クライアントを利用する必要性は低下している．

サーバを中心とした構築形態では Thin クライアント方式がある．サーバサイドにプッシュサーバを用意し，クライアントサイドにプッシュ通信モジュールを組み込むことで HTTP の問題を解決できる．

しかし，これらの構築形態にはともに二つの問題がある．先に示した共有が同一ドメイン内に限定されるという問題と，SSL に対応できないという問題である．共有が同一ドメイン内に限定されるという問題は協調アプリケーション構築では問題にならないが，SSL に対応できないということはコンタクトセンターなどのログイン後のユーザの入力補助を行なうような協調アプリケーション構築において大きな問題である．

この二つの問題を解決する構築形態がプロキシを利用した構築形態である．プロキシ内に共有するウェブページを取り込み，書き換え，一時的に保存することで共有を同一ドメイン内に限定しないシステムを構築することができる．また，プロキシに HTTP リクエストの転送機能をもたせることで SSL に対応することができる．しかし，共有を同一ドメイン内に限定するかしないかという選択はシステムの目的による．ウェブベースの協調アプリケーション構築であれば異なるドメインのウェブページを共有する必要はない．一方，インターネットに存在する任意のウェブサイトを共有したい場合は，異なるドメインのウェブページも共有できることが必須となる．

プロキシを利用した構築形態では，構築上の長所も多くある．上記の二つの構築形態ではウェブアプリケーションサーバ内にウェブブラウジング共有機構が構築されていた．しかしプロキシを利用する構築形態ではプロキシにウェブブラウジング共有機構をもたせることができるので，ウェブアプリケーションとウェブブラウジング共有機構を分離することができる．そのため，既存のウェブアプリケーションを変更することなく協調アプリケーションに再構築することが可能となる．また，クライアントサイドに一般的なウェブブラウザを利用することができるため，ユーザ側の負担を最小限に抑えることができる．クライアントサイドのセキュリティに関してもウェブブラウザに依存することができる．

これまで紹介したシステム構築形態はすべてクライアント・サーバ方式であったが，P2P 方式のシステム構築形態もある．P2P 方式ではサーバがないため，クライアントが共有機能を制御する機構をすべてもつ必要がある．クライアント単体でウェブブラウジングの共有が可能な点は大きな利点であるが，P2P 方式は通信が一対一に限られてしまうため，本システムの構築形態には適用できない．

### 3.2 本システムの構築形態

我々が構築するウェブブラウジング共有システムの目的は，インターネットの利用補助である．つまりユーザの中に最低一人はインターネットの利用に不慣れな人が含まれる．そのようなユーザが利用することを想定して，可能な限り利用しやすいシステム構築形態を取る必要がある．クライアントサイドの要件としては，一般的に利用されているウェブブラウザをそのままの状態で利用できることが望ましい．加えて，複数の種類のウェブブラウザで動作することも求められる．サーバサイドの要件としては，インターネット上に存在する任意のウェブページを共有できることが必須である．

以上の諸条件から，我々はプロキシを利用した構築形態を本システムに適用する．ただし一般のユーザ同士が利用するウェブサービスを想定しているため，SSL には対応しない．この構築形態ではサーバプッシュ型の通信を実装する必要があるが，我々の研究ではウェブブラウジングの共有が目的であるため，サーバプッシュ型の通信をサポートするフレームワークを利用するのみに留める．

ここで，プロキシを利用した構築形態の問題点について議論する．その第一は，ウェブページを共有化するために追加したスクリプトがウェブページ本来のスクリプトの動作を阻害してしまう可能性がある点である．この問題はウェブページの共有化の際に，ウェブページ本来のスクリプトの動作を阻害しないような共有化を行なうことで解決していく．第二は，ウェブページの信頼性の問題である．これは共有化されたウェブページが悪意のないウェブページであるとユーザに信頼してもらう以外に解決方法はない．第三は法的な問題である．プロキシを利用した構築形態では，著作権の存在するウェブページをキャッシュしてしまっている．しかし，日本では 2009 年 6 月 12 日に改正著作権法が成立しており，事実上ウェブページのキャッシュが認められた．ウェブページを共有化しているという問題もあるが，改正著作権法にある「必要と認められる限度」であるかどうかはここでは議論しない．

## 4. システムの機能

我々は誰でも利用できるような簡易な操作機能をもち，かつ高い共有機能をもつシステムを目指す．ページ遷移だけでなく，ボタンやフォーム値の操作も共有する．ボタンの共有によって JavaScript に対応する．加えて，スクロールやポインタ，ウィンドウサイズの共有も行なう．共有は二人以上のユーザ間で可能である．

### 4.1 GUI 仕様

本システムは図 4.1 のような GUI をもつ．

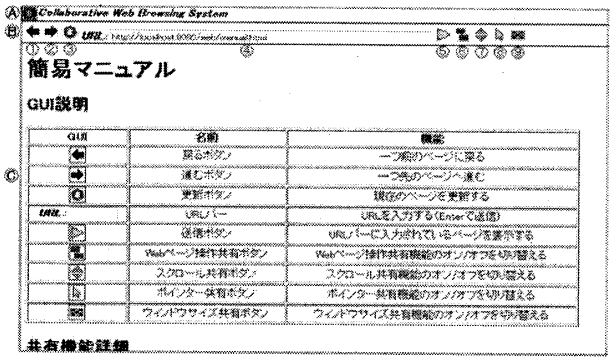


図 4.1　GUI

本システムは HTML を利用した三つのフレームによって構成される．上段Ⓐはロゴとシステム名を表示するインフォメーションフレーム，中段 Ⓑはシステム操作のためのコントロールフレーム，下段Ⓒは任意のウェブページを表示するブラウジングフレームである．コントロールフレームには各ブラウザ固有のブラウザ操作ボタンとは別に，本システム専用のブラウザ操作ボタンを設ける．それが，①戻るボタン，②進むボタン，③更新ボタン，④URL バー，⑤送信ボタンである．また，本システム独自の共有機能の操作に必要な，⑥ウェブページ共有ボタン，⑦スクロール共有ボタン，⑧ポインタ共有ボタン，⑨ウィンドウサイズ共有ボタンがある．

共有機能の一例として，ウェブページ遷移共有機能の動作を図 4.2 に示す．三つの HTML ファイルを例として使用する．page A は page B へのリンクをもち，page B は page C へのリンクをもつ．page C はリンクをもたない．A と B の二つのクライアントが本システムを利用する．状態 1 では A，B 共に page A が表示されている．A はリンクをクリックして Page B へ遷移する．状態 2 ではウェブページ共有ボタンは押されていないので，B は page A を表示したままである．A はウェブページ共有ボタンをオンにし，リンクをクリックして page C へ遷移する．状態 3 ではウェブページ共有機能がオンになっているので，B には自動的に A と同じ page C が表示される．

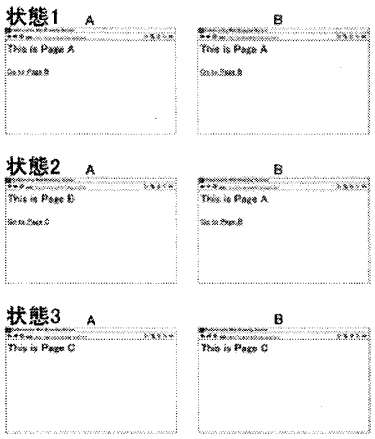


図 4.2　ウェブページ遷移共有機能の動作

## 4.2 ブラウザ機能と共有機能

A) ブラウザ機能は，

- 戻る機能
- 進む機能
- 更新機能
- URL バーの機能
- 送信機能

である．各ウェブブラウザの機能とは別の，本システム固有のブラウザ機能である．機能の内容と操作方法はほぼ同様であり，各機能は対応するボタンの押下によって実行される．B)で後述するウェブページ共有ボタンがオンの場合，これらのブラウザ機能の操作は共有される．これらの機能で利用する履歴はユーザ毎に記録される本システム固有の履歴であり，各ウェブブラウザの履歴とは異なる．

戻るボタンを押すと戻る機能が実行される．一つ前に表示されていたウェブページを表示する．進むボタンを押すと進む機能が実行される．一つ後に表示されていたウェブページを表示する．更新ボタンを押すと更新機能が実行される．URL バーには現在表示されているウェブページの URL が表示される．URL バーにはユーザが自由に URL を入力できる．URL バーにフォーカスがある状態で Enter キーを押す，もしくは送信ボタンを押すと送信機能が実行される．現在 URL バーに入力されている URL のウェブページを表示する．

B) 共有機能はウェブページ操作共有機能とユーザ操作共有機能の二つに大別できる．ウェブページ操作共有機能は，

- ウェブページ遷移共有機能
- ボタン共有機能
- フォーム値共有機能

である．ユーザ操作共有機能は，

- スクロール共有機能
- ポインタ共有機能
- ウィンドウサイズ共有機能

である．各ボタンはトグルボタンになっており，共有機能のオンオフが可能である．初期設定はすべての共有機能がオフである．オンの場合には各操作が共有相手に反映されるようになる．ウェブページ遷移共有機能は A)のブラウザ機能を共有するかどうかのオンオフも操作する．

ウェブページ共有ボタンがオンになると，ウェブページ操作共有機能の三つすべてがオンになる．ウェブページ遷移共有機能がオンになると，自身のウェブページ遷移が共有相手に反映される．ウェブページ遷移とは，ハイパーリンクのクリック，戻るボタン，進むボタン，更新ボタンの押下，URL バーへの URL 入力後の Enter キーおよび送信ボタンの押下によるウェブページ遷移のことである．ボタン共有機能がオンになると，自身のボタン操作が共有相手に反映される．フォーム値共有機能がオンになると，自身のフォーム値の操作が共有相手に反映される．フォーム値とはテキストボックスへの入力文字のことである．

スクロール共有ボタン，ポインタ共有ボタン，ウィンドウサイズ共有ボタンがオンになると，自身の各操作が共有相手に反映されるようになる．共有用のポインタである共有ポインタはドラッグが可能である．あるユーザが共有ポインタをドラッグした場合，ポインタ共有機能がオフの共有相手にも自動的に共有ポインタが表示される．ポインタ共有機能をオフにした場合，自身の共有ポインタは消え，位置も初期化される．

## 4.3 共有の双方向性

一般に，共有機能には二つの選択肢を存在させることができる．自身の操作を共有相手に反映させるかどうかという選択肢と，共有相手の操作を自身に反映させるかどうかという選択肢である．後者の選択肢が存在する場合，自身の操作が共有相手に反映されたかどうかを知るすべが必要となる．文献[3]のシステムでは二つの選択肢をそれぞれマスターとスレーブと呼び，任意に設定可能

としている．このシステムでは Session Status window という共有相手の状態を確認できる機能を付けることで上記の問題を解決している．

本システムでは自身の操作を共有相手に反映させるかどうかという選択肢のみ存在する．つまり，各共有機能をオンにした共有相手の操作は常に自身に反映され，拒否することができない．これは本システムがウェブブラウジングの共有を前提としているためである．また，本システムを利用するユーザの中で最低 1 人はインターネットに不慣れなユーザであると想定するため，共有を行なう上で可能な限りそのようなユーザが行なう操作を少なくするためでもある．本システムの利用と並行して独自のウェブブラウジングを行ないたい場合は，新しいウィンドウやタブで行なってもらうしかない．しかし，現在の一般的なウェブブラウザの多くはタブブラウザであるため，この仕様は大きな問題ではない．

### 4.4 共有機能の仕様

共有可能なウェブページは HTML および XHTML で作成されているウェブページである．Flash で作成されているページは共有できない．また，ウェブページに含まれる動画や音楽の再生などの動作も共有することができない．上記とは別に，共有したいウェブページの存在するサーバがプログラムからの接続を拒否している場合には，ウェブページを共有化することができないためウェブブラウジングを共有することができない．通信プロトコルは http に対応し，ftp や https には対応しない．

対応ブラウザは，Internet Explorer，Firefox，Opera，Safari，Google Chrome である．各ユーザの利用するウェブブラウザおよびブラウザ設定が異なる場合，スクロール，ポインタ，ウィンドウサイズの共有が正確にならない場合がある．プラグインなどによってウェブブラウザのツールバーがデフォルトより多い場合も同様である．

## 5. 同一生成元ポリシー

同一生成元ポリシー（Same Origin Policy）[8]，[12]はウェブブラウザがもつセキュリティの一つである．あるドメインの生成元から読み込まれたドキュメントやスクリプトが，異なるドメインの生成元から読み込まれたドキュメントのプロパティを取得したり設定したりするのを防ぐ．各ウェブブラウザのセキュリティはそれぞれの開発元が別々に JavaScript のセキュリティモデルに従っているが，セキュリティとしての機能はほぼ同一である．クロスドメイン制限とも呼ばれ，クロスドメイン制限を超えたアクセスのことをクロスドメインアクセスという．

このセキュリティのため，ウェブブラウザで実行される JavaScript は自身が含まれるウェブページの生成元であるサーバと同一のドメインの要素にしかアクセスすることができない．本システムはコントロールフレーム内の JavaScript がブラウジングフレームのドキュメントを参照することで共有機能の制御を行なっている．つまり，ブラウジングフレームに異なるドメインのドキュメントが表示されている場合，共有機能が動作しない．本システムでは任意のドメインのウェブページをブラウジングフレームに表示する必要があるため，同一生成元ポリシーを回避する必要がある．我々は共有化したウェブページを共有サーバ内に一時的に保存することで同一生成元ポリシーの制限を回避した．この方法によって，ブラウジングフレームに表示される共有化したウェブページを本システムと同一のドメイン上に配置することができる．

## 6. DWR

本システムではクライアントサイドの JavaScript とサーバサイドの Java の通信に DWR（Direct Web Remoting）[2]，[13]を利用している．DWR は Java プログラマのための Ajax アプリケーション開発用フレームワークである．DWR を利用することで，クライアントサイドの JavaScript からサーバサイドの Java オブジェクトを呼び出すことができるようになる．クライアントとサーバの間の非同期通信，データ変換方式などは DWR が処理してくれる．

本システムでは上記の機能に加え，ReverseAjax という機能を利用している．この機能は，サーバサイドの Java からクライアントサイドの JavaScript を非同期に呼び出すことを可能にする．一般的には Comet と呼ばれるサーバプッシュ型の通信方法である．

## 7. 実装

本システムはウェブアプリケーションであり，実装方法は一般的な方法と大きくは変わらない．また，ウェブページを共有可能なものに書き換える方法も従来の研究と大きな差はないと思われるが，具体的に記述されたものがないように思われるため，本論文では具体的な実装についてある程度まで記述する．

### 7.1 システム構成

クライアントサイドの JavaScript とサーバサイドの Java がシステムを構成する．図 7.1 にシステム構成を示す．GetHTMLServlet.class はウェブページの共有化を行なう．共有化では WWW に接続する．また各共有機能は以下で示されるファイルによって実装される．

- ウェブページ遷移共有：SharingURL.js, URLServlet.class
- ボタン共有：SharingButton.js, ButtonServlet.class
- フォーム値共有：SharingFormValue.js, FormValueServlet.class
- スクロール共有：SharingScrollAmount.js, ScrollAmountServlet.class
- ポインタ共有：SharingPointer.js, PointerServlet.class
- ウィンドウサイズ共有：SharingWindowSize.js, WindowSizeServlet.class

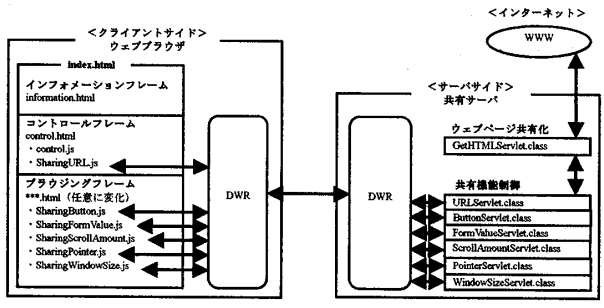


図 7.1　システム構成

本システムの共有サーバ上での実際のディレクトリ構成を図 7.2 に記述する．通信部分に利用している DWR 本体は dwr.jar である．共有化したウェブページは temp ディレクトリ以下に一時的に保存される．

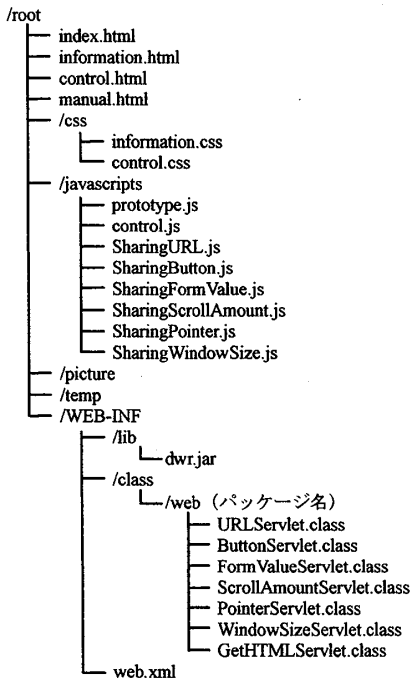


図 7.2　ディレクトリ構成

また，各共有機能で共通となるデータの流れを図 7.3 に示す．共有機能はオンの状態とする．

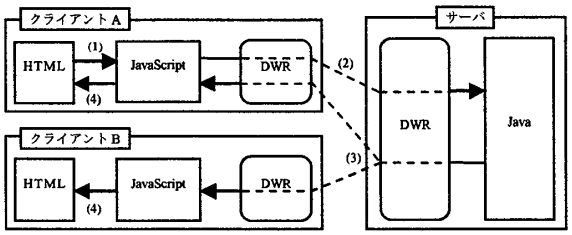


図 7.3　共有機能のデータの流れ

(1)ではページ内の変化を検出するイベントハンドラが JavaScript 関数を呼び出し，共有に必要な情報を HTML から取得する．(2)では(1)で呼び出された JavaScript 関数が DWR を通じてサーバサイドの Java を呼び出し，共有情報を渡す．(3)では(2)で呼び出された Java が DWR を通じてすべてのクライアントサイドの JavaScript 関数を呼び出し，共有情報を渡す．(4)では(3)で呼び出された JavaScript 関数が実行され，共有機能に応じた処理がなされる．

## 7.2 ウェブページの共有化

本システムで表示されるウェブページはウェブ上にあるものそのままではなく，共有サーバ内にある共有化されたウェブページである．共有サーバ内に再配置することで，5 節で述べた同一生成元ポリシーを回避する．また，共有化によってウェブページに共有機能を付加する．オリジナルのウェブページにはユーザによるページ内の各操作を検出する機能がないため，ソースコードを書き換えることで機能を追加している．ソースコードの書き換えは必要最低限であり，ウェブページ作成者の意図やデザインを変更してしまうような書き換えは行なわない．

ウェブページの共有化はウェブページ遷移共有機能での URL の受け渡しの中で行なわれる．共有化する対象は HTML ファイルの他に，その HTML ファイルが外部参照する CSS ファイル，JS ファイル，画像ファイル，フレーム内の HTML ファイルである．ソースコードの書き換えでは，本システム固有の JS ファイルへの外部参照の追加，共有機能を検知するためのタグの書き換えが行なわれる．

同時に，共有のための情報を保持する本システム固有の JS ファイルの生成が行なわれる．この JS ファイルは図 7.4 に示される button タグの情報を格納するボタン情報配列と図 7.5 に示される form タグおよび form タグ内の input タグの情報を格納するフォーム情報配列をもつ．ボタン情報配列は二次元配列となっており，一次元は 0 から始まるページ内の button タグのナンバー，二次元は連想配列としてタグ名と属性名が使われている．フォーム情報配列は三次元配列となっており，一次元は 0 から始まるページ内の form タグのナンバー，二次元は form タグ毎に 0 から始まる input タグのナンバー，三次元は連想配列としてタグ名と属性名が使われている．二つの情報配列にはそれぞれ図 7.4 と図 7.5 に示される値が格納される．情報配列に値を格納する際，タグに id 属性，name 属性がない場合は自動的に生成される．これらの配列を用いることで，操作されたボタンの生成元となるタグを判別することができる．ウェブページ共有機能がオンの場合，すべてのユーザのブラウジングフレームに表示されているウェブページは同じである．そのため，すべてのユーザにおいてこれら二つの情報配列の同一性が保証される．クライアントはサーバから配列名を受信するだけでページ内のどのタグの操作を実行すればよいのかを判断できる．

```
button[ ]['type'] : button タグの type 属性の値
button[ ]['buttonID'] : button タグの id 属性の値
button[ ]['buttonName'] : button タグの name 属性の値
button[ ]['JavaScript'] : button タグの機能を受けもつ JavaScript
```

図 7.4　ボタン情報配列の内容

```
form[ ][ ]['formID'] : form タグの id 属性の値
form[ ][ ]['formName'] : form タグの name 属性の値
form[ ][ ]['type'] : input タグの type 属性の値
form[ ][ ]['inputID'] : input タグの id 属性の値
form[ ][ ]['inputName'] : input タグの name 属性の値
form[ ][ ]['inputValue'] : input タグの value 属性の値
form[ ][ ]['JavaScript'] : input タグの機能を受けもつ JavaScript
```

図 7.5　フォーム情報配列の内容

## 7.3 共有化の内容

head タグ内に各共有機能を制御する JS ファイルへの外部参照を追加することでスクロール共有機能，ポインタ共有機能，ウィンドウサイズ共有機能を実装する．

a タグの書き換えによってハイパーリンクのクリックによるウェブページ遷移共有機能を実装する．a タグ内に

onclick イベントハンドラを追加し，引数として受け取った URL を共有サーバに送信する sendURL 関数をイベントハンドラに設定する．sendURL 関数は false を返すため，ハイパーリンク本来の機能は実行されない．図 7.6 に具体的な例を示す．

```
<a href="http://www.meiji.ac.jp">明治大学へのリンク</a>
                    ↓
<a href="http://www.meiji.ac.jp" onclick="top.control.sendURL('http:
//www.meiji.ac.jp')">明治大学へのリンク</a>
```

図 7.6　a タグの書き換え

button タグおよび form タグ内の input タグの書き換えによってボタン共有機能を実装する．button タグおよび input タグの type 属性が button の場合は汎用ボタンを共有するための書き換えを行なう．タグ内のイベントハンドラにボタン情報配列またはフォーム情報配列を共有サーバに送信する sendButton 関数を設定する．元々イベントハンドラに設定されていた JavaScript はボタン情報配列またはフォーム情報配列に格納される．input タグの type 属性が submit の場合はサブミットボタンを共有するための書き換えを行なう．onclick イベントハンドラを追加し，フォーム情報配列を共有サーバに送信する sendButton 関数を設定する．さらに，フォームを送信する JavaScript を新たに生成してフォーム情報配列に格納する．sendButton 関数は false を返すため，サブミットボタン本来の機能は実行されない．ボタン共有機能実行時の動作は汎用ボタン，サブミットボタンで共通となる．ボタン情報配列またはフォーム情報配列が参照され，実行する JavaScript 関数を取り出し，eval 関数を用いて二重に評価することで各ボタンの機能を実行する．図 7.7 に type 属性が button の button タグの書き換えの例を示す．また，図 7.8 にその時のボタン情報配列の値を示す．id 属性と name 属性の値となっている button0 は自動的に生成された値である．

```
<button type="button" onclick="hoge()">hoge</button>
                    ↓
<button id="button0" name="button0" type="button"
onclick="sendButton('button[0]')">hoge</button>
```

図 7.7　button タグの書き換え

```
button[0]['type'] : button
button[0]['buttonID'] : button0
button[0]['buttonName'] : button0
button[0]['JavaScript'] : hoge()
```

図 7.8　ボタン情報配列の値

form タグ内の type 属性が text の input タグの書き換えによってフォーム値共有機能を実装する．onchange イベントハンドラと onkeyup イベントハンドラを追加し，フォーム情報配列と現在の input タグの値を共有サーバに送信する JavaScript 関数を設定する．さらに，input タグに値を代入する JavaScript 関数を生成し，フォーム情報配列に格納する．共有時にはボタン共有と同じく eval 関数を利用する．

## 8. おわりに

ウェブページ遷移の共有に加えて，ボタン共有，フォーム値共有，ユーザ操作共有という実際的なウェブページ共有機能をもつウェブブラウジング協調システムを実現した．今後追加されるべき機能としては，テキストエリア，チェックボックス，ラジオボタンなども共有できるフォーム値共有機能，チャット機能，グループ別共有機能が考えられる．チャット機能とグループ別共有機能は協調システムでは重要である．また動的ページが普及した現在のウェブの動向を考えると，DOM による動的書き換えや Ajax に対応していくことが今後のウェブブラウジング共有においての課題の一つであろう．

我々はインターネット利用の遠隔補助を目的とした協調システムを実現した．しかし，このシステムはもう一つの側面も併せもつ．我々が実装した同一ドメインに縛られない自由なウェブブラウジング共有は，ユーザに新しいインターネットの姿を提案していると考える．それは，従来一人で閲覧するものであったインターネットを，複数人で共有しながら閲覧するという概念である．本システムを応用すれば，多人数向けのウェブサイトや多人数で閲覧することによるページランクなどのウェブサービスも考えられる．近年注目されている協調的なウェブアプリケーション[4]を通じて，今後もインターネットが新しい姿を見せてくれることを期待したい．

### 参考文献


[1] Y. Aoki, “Building a Collaborative Web Environment for Supporting End Users”, IPSJ Journal,. Vol.43, No.2, pp.530-542 (2002).

[2] The DWR project, DWR - Easy Ajax for JAVA, http://directweb remoting.org/

[3] Alan W. Esenther, “Instant Co-Browsing: Lightweight Real-Time Collaborative Web Browsing”, WWW2002 Conference Proceedings Posrers, May, 2002.

[4] Google, Google Wave, http://wave.google.com/

[5] 伊豆陸, 中島伸介, 小山聡, 角谷和俊, 田中克己, “グループ型 Web 閲覧による探索アクティビティ情報の共有と利用”, 第 14 回データ工学ワークショップ (DEWS2003), March, 2003.

[6] 城島貴弘, 子林秀明, “Web サイトを用いた対面業務を可能とする Web ページ共有方式”, 情報処理学会 第 137 回マルチメディア通信と分散処理研究発表会, November, 2008.

[7] 柿元宏晃, 児玉政幸, 浅見昌平, 大囿忠親, 新谷虎松, “サーバ主導型 Push 配信を利用した同期分散型 Web ブラウジングシステムの試作”, 第 70 回情報処理学会全国大会講演論文集 (2008).

[8] Mozilla Foundation, Same origin policy for JavaScript – MDC, https://developer.mozilla.org/en/Same_origin_policy_for_JavaScript

[9] 中島一彰, 川本亜紀子, 大芝崇, 吉坂主旬, 田淵仁浩, “リアルタイム Web 共有方式による双方向コミュニケーション基盤”, 情報処理学会 第 50 回グループウェアとネットワークサービス研究会, pp.57-64, January, 2004.

[10] 篠崎雅英, 小林真, 坂入隆, “ブラウザ同期を用いた WWW 共有型アプリケーションの構築環境”, 情報処理学会 第 57 回全国大会, 3M-05, October, 1998.

[11] 株式会社シンクプラス, Sync+, http://www.syncplus.net/

[12] WHATWG, HTML 5, http://www.whatwg.org/specs/web-apps/ current-work/

[13] F. W. Zammetti, Practical DWR 2 Projects, APRESS, 2008.