マルチディスプレイ・レスポンシブウェブシステムにおける複数ブラウザ間の状態共有

坂井成道[†1] 南圭祐[†1] 峰松美佳[†1] 川添博史[†1] 会津宏幸[†1]

スマートフォンやスマートテレビをはじめ，ネットワークに接続可能なディスプレイ搭載機器が普及し，ユーザはそれらを複数種類所持するようになった．今後，ユーザがそれぞれの機器の特色を活かしながら同時に組み合わせて利用することが想定される．一般に，複数の機器を連携させるアプリケーションは単体の機器上で動作するアプリケーションよりも設計や実装が難しい．我々はこれまで，複数の機器のブラウザが連携して動作する新しいウェブ体験「マルチディスプレイ・レスポンシブウェブ」を提案してきた．本稿では，マルチディスプレイ・レスポンシブウェブシステムを動的なウェブページに適用するにあたり必要なページ状態の共有について検討した方式について述べる．

State Sharing Between Web Browsers in Multi Display Responsive Web System

NARUMICHI SAKAI[†1] KEISUKE MINAMI[†1] MIKA MINEMATSU[†1]
HIROSHI KAWAZOE[†1] HIROYUKI AIZU[†1]

1. はじめに

スマートフォンやスマートテレビ [1] のようなネットワーク対応機器を，ユーザが利用する機会が増加している．さらに，ディスプレイを搭載した機器が多数普及したことと，それらの低価格化により一人のユーザが複数の機器を所持するようになったことで，ユーザが従来よりも多くのディスプレイを通して情報収集や情報発信，コンテンツの享受をすることが多くなっている．これにともなって，機器を同時に複数組み合わせて利用するマルチ機器（マルチスクリーン）のユースケースが想定されている．たとえば，テレビで動画を見ながら動画の感想をスマートフォンでSNSに投稿するなど，機器によって役割は違うものの，ユーザの一連の行動がコンテンツを楽しむというひとつの目的の元にある，といった例は現在もよく見られる．

一般に，複数の機器を連携させるアプリケーションの開発は，単体の機器上で動作するアプリケーションの開発よりも困難になる．例えば，複数のスクリーンを自由にレイアウトできる場合や使用する機器の数が多い場合，それらが自由に参加離脱することや，機器の種類や数に応じてレイアウトを適切に変更することを想定してアプリケーションを設計しなければならない．

一方，スマートフォンに代表されるように，機器のディスプレイ解像度は多岐にわたっている．従来のシングルスクリーンアプリケーション開発においても，開発者はこれらのさまざまなディスプレイ解像度に合わせてアプリケーションを開発し，テストを行う必要があった．ディスプレイ解像度のバリエーションへ対応する技術として，ウェブの世界では「レスポンシブウェブ」という概念を用いることがある．これは HTML5[2] の柔軟性を利用して，図 1 のように機器のディスプレイ特性に合わせて表示を行う．例えば，PC のブラウザで表示した場合には 3 ペイン構成になっているウェブページを，ディスプレイの幅の小さなモバイル機器のブラウザでは 1 ペインに構成する．

我々はこの概念をマルチディスプレイに拡大した，「マルチディスプレイ・レスポンシブウェブ」を過去に提案した [3]．一方，近年，ブラウザ上のスクリプトによるページ要素の書き換えによるアプリケーションとしての利用が増えている．参考文献 [3] の方式では，この動的な要素書き換えに対応していなかった．ブラウザが保持するページの状態に一貫性を持たせ，ページの状態変更を共有させることが課題であることがわかった．

本稿では，マルチディスプレイ・レスポンシブウェブを動的な要素書き換えが生じるウェブページへ対応させる方式，すなわち，ページの状態変更の共有について網羅的に検討した結果を述べる．

†1（株）東芝 研究開発センター ネットワークシステムラボラトリー
Network System Laboratory, Corporate Research & Development Center

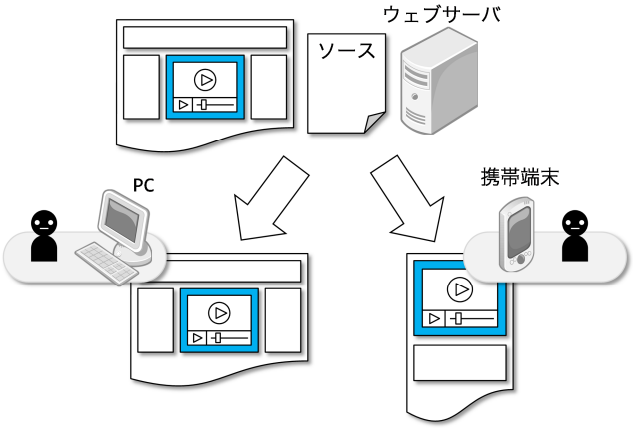

図 1：レスポンシブウェブのイメージ

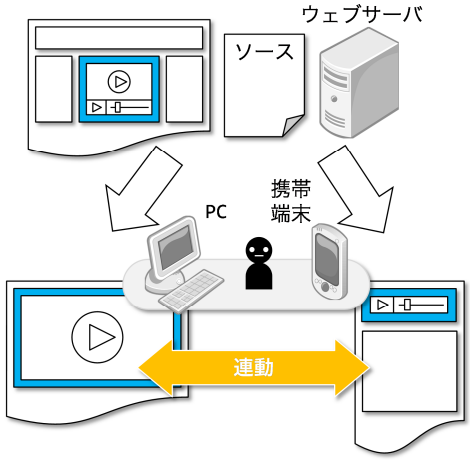

図 2：マルチディスプレイ・レスポンシブウェブのイメージ

2. 動的な要素書き換えが発生するウェブページへの適用

2.1 動的なウェブページ

動的な要素書き換えが発生する一般のウェブページでは，ページ閲覧中にページ内のスクリプト処理が DOM (Document Object Model) 変更をすることがある．DOM 変更とは，DOM 状態の変更のことで，DOM 状態とは，ページを構成する部品である DOM 要素とそれらの親子関係である DOM ツリーの状態を指す．

従来のウェブページ閲覧時，すなわち，ひとつの動的なウェブページをひとつのブラウザで表示する場合，DOM 状態はひとつのブラウザが保持している．Ajax (Asynchronous JavaScript™ XML) 等を用いてサーバと通信を行うようなページにおいても，DOM 状態の保持自体はひとつのブラウザが行う．複数のブラウザが同じページを閲覧する場合にも，それぞれのブラウザがそれぞれの内部で DOM 状態を保持しており，それらは互いに独立している．

2.2 マルチディスプレイ・レスポンシブウェブへの適用

従来のウェブとは異なり，マルチディスプレイ・レスポンシブウェブでは，ひとつのページを複数のブラウザで連動表示させる．このため，特に動的な要素書き換えが発生するウェブページにおいては，あるブラウザ上で発生した DOM 変更を他のブラウザが知る必要がある．

DOM 変更を共有するためには，ブラウザ同士が直接 DOM 状態を共有する方法と，仲介者を介在させて共有する方法がある．後者は DOM 状態の共有を必ず仲介者を通して行い，仲介者が常に最新の DOM 状態を保持する．最新の DOM 状態の保証しやすさの観点から，後者の方法を採用した．

3. DOM 変更共有方式検討

我々は，複数のブラウザ間の DOM 変更共有のために，図 3 のようにユーザ側のブラウザであるスレーブの他にマスタを用いた．マスタを含めた各ブラウザはそれぞれページの DOM 状態を保持する．以降，マスタブラウザをマスタ，スレーブブラウザをスレーブと略して表記する．

マスタは，スレーブが保持する DOM 状態を統括するブラウザである．スレーブは，基本的にマスタの指示に従って自身が保持する DOM 状態を変更する．マスタは任意の機器上で動作させることができる．本稿では，プロキシサーバ上でこれを動作させた．

スレーブは機能組み込みモジュールを通して，DOM 変更共有のための機能を得る．本稿ではこの機能組み込みモジュールの詳細は省略する．詳しくは参考文献 [3]を参照のこと．

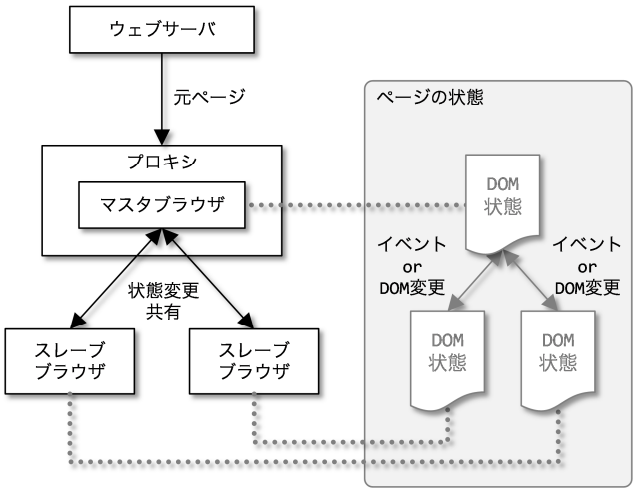

図 3：マスタブラウザを用いたイベントおよび DOM 変更の共有

* JavaScript は，米国 Oralce Corporation およびその子会社，関連会社の米国およびその他の国における登録商標または商標です．

一般に，共有対象となる DOM 変更は，ページ作者があらかじめ定義したスクリプト処理が発端となる．スクリプト処理には「イベント」により発動した「イベントハンドラ」を含む．イベントとは，ページ閲覧ユーザからブラウザへの入力や，ページ読み込み状態の変更等の契機のことで，イベントハンドラとは，イベントを受けて発動するスクリプト処理である．DOM 変更は，発動したイベントハンドラの内部でなされることがある．

ブラウザ間でこの DOM 変更を共有するには, DOM 変更の原因となったイベントを共有する方法と，DOM 変更そのものを共有する方法がある．

また，ブラウザにはスレーブとマスタが存在するため，共有方向は「スレーブ→マスタ」と「マスタ→スレーブ」の 2 種類となる．前者のスレーブはイベントの発生元であるため，以降,「発生元スレーブ」，後者のスレーブは変更を反映させる先であるため，以降,「反映先スレーブ」と呼ぶ．

2 種類の共有方向にて，イベントあるいは DOM 変更を共有する．表 1 に組み合わせをまとめる．

表 1：共有方向と共有対象の組み合わせ

		マスタ→スレーブ方向	
		イベント共有	DOM 変更共有
スレーブ→マスタ方向	イベント共有	イベント共有方式	ハイブリッド方式
	DOM 変更共有	×	DOM 変更共有方式

表 1 のように，共有方向と共有対象の組み合わせは 4 つ存在する．ただし,「×」と表記した組み合わせについては，実現できないため除外する．実現できない理由は，マスタ上で DOM 変更をその原因となったイベントへ変換できないためである．すなわち，DOM 変更がイベントに起因しない場合や複数のイベントに起因する場合があり，DOM 変更とイベントが必ずしも一対一に対応付けできないためである．

以降に述べるイベントは，ページ上のスクリプトにて DOM 変更を発生させるものとする．それぞれの方式について詳しく述べる．

3.1 イベント共有方式

発生元スレーブは，ページ上で発生したイベントをマスタへ送信し，マスタは反映先スレーブに対してそのイベントを送信する．変更反映先のスレーブ上のイベントハンドラが受信したイベントを処理し，処理内容に応じて DOM 変更を発生させる．発生元スレーブでもイベントの処理が行われ，DOM 変更が発生する．図 4 にフローを示す．

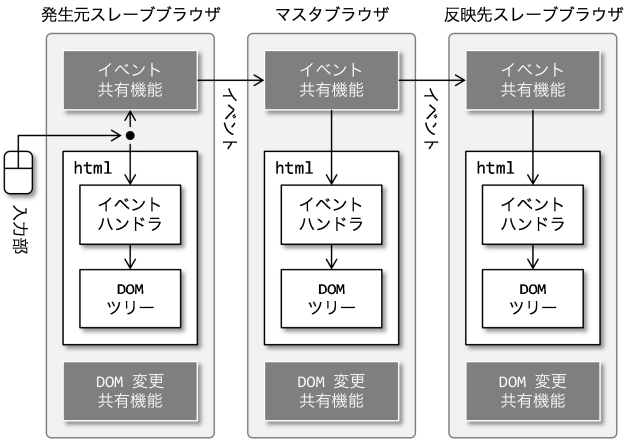

図 4：イベント共有方式によるイベント共有フロー

3.2 ハイブリッド方式

イベント共有方式と同様に，発生したイベントを発生元スレーブがそのままマスタへ送信する．マスタ上のイベントハンドラが受信したイベントを処理し，処理内容に応じて DOM 変更を発生させる．次に，発生した DOM 変更をマスタがスレーブへ送信する. スレーブは, 受信した DOM 変更を自身の DOM ツリーへ反映する．図 5 にフローを示す．

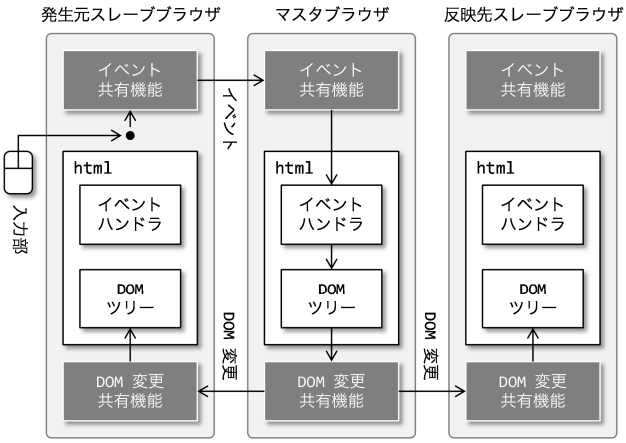

図 5：ハイブリッド方式によるイベントおよび DOM 変更共有フロー

3.3 DOM 変更共有方式

発生元スレーブ上のイベントハンドラがページ上で発生したイベントを処理し，結果生じた DOM 変更をマスタへ送信する．マスタは受信した DOM 変更を自身の DOM ツリーへ反映すると同時に，反映先スレーブへ同じ変更を送信する．反映先スレーブは，受信した DOM 変更を自身の DOM ツリーへ反映する．図 6 にフローを示す．

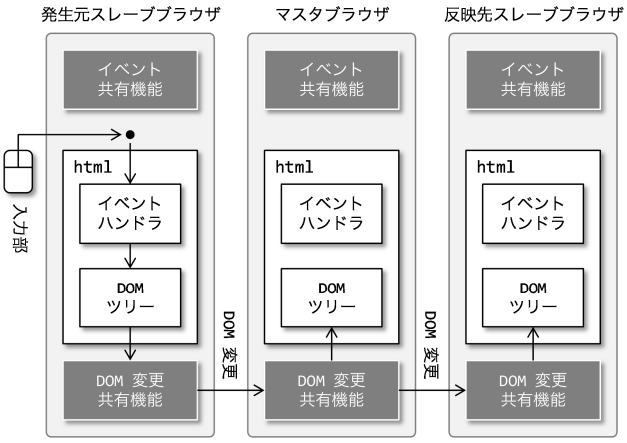

図 6：DOM 変更方式による DOM 変更共有フロー

4. 考察

それぞれの方式は，イベントあるいは DOM 変更が伝搬した結果の DOM ツリーの最終状態が異なる．最終状態はイベントを処理するイベントハンドラに依存し，イベントハンドラの処理結果はブラウザの環境によって異なる．ブラウザの環境は，発生元スレーブ，マスタ，反映先スレーブの 3 種類である．

「イベント共有方式」では，反映先スレーブがイベントハンドラにてイベントを処理する．反映先スレーブの最終状態は反映先スレーブの環境に依存する．

「ハイブリッド方式」では，マスタがイベントハンドラにてイベントを処理する．反映先スレーブの最終状態はマスタの環境に依存する．

「DOM 変更共有方式」では，発生元スレーブがイベントハンドラにてイベントを処理する．反映先スレーブの最終状態は発生元スレーブの環境に依存する．

採用すべき方式は，オリジナルページの意図によって異なる．例えば「イベント共有方式」は，スレーブそれぞれの横幅に応じた処理を行うなど，スレーブそれぞれのローカル環境に応じて処理を行いたいページに対して採用する．「ハイブリッド方式」は処理の内容を全てマスタに合わせたいとき，「DOM 変更共有方式」は発生元スレーブに合わせたいときに用いる．

例えば，カメラの画像を取り込んで表示するページの場合，コンテンツの意図が「シャッターを押したタイミングを共有する」ものであれば「イベント共有方式」を，意図が「撮影元が撮影した画像を共有する」ものであれば「DOM 変更方式」を採用する．「ハイブリッド方式」は，マスタの処理内容を強制的にスレーブへ反映させることができるため，バージョンや環境が多岐にわたる複数のスレーブそれぞれに処理内容を依存させたくない場合に採用する．

5. おわりに

多くの機器のディスプレイを連携させて利用するウェブのスタイルであるマルチディスプレイ・レスポンシブウェブにおいて，ページの状態変更の共有について検討した方式を網羅して述べた．動的な要素書き換えが発生するウェブページを複数の機器へ表示し，すべてのブラウザを表示の変化に追従させるためには，イベントおよび DOM ツリー変更の共有が必要になる．

ページの状態変更の共有方式は 3 通りある．それらは DOM ツリーの変更処理を行うイベントハンドラがそれぞれ異なるため，最終的に反映先スレーブに出力される DOM ツリーが異なる．オリジナルページの意図によって採用すべき方式が異なることを示し，その意図によってどの方式を採用すべきかの指針を示した．

参考文献

1) 坂本典哉：W3C での Web and TV の規格化動向，東芝レビュー，Vol.68, No.2, pp60-61 (2013)
2) HTML5, W3C (online), available from <http://www.w3.org/TR/html5/>
3) 坂井成道，峰松美佳，川添博史，会津宏幸：マルチディスプレイ・レスポンシブウェブ，DICOMO 2013, 2G-4
4) 小林透，瀬古俊一，川添雄彦：HTML5 によるマルチスクリーン型次世代 Web サービス開発，翔泳社 (2013)