Webオーサリング手法の提案と実装

Web標準の進化に対応した

Proposal and Implementation of a Web Authoring Method Corresponding to Web Standards Evolution

朱　　槿，中里秀則，浦野義頼，金　　群
Jin ZHU, Hidenori NAKAZATO, Yoshiyori URANO, Qun JIN

概要

「標準準拠」Webデータが「標準」自体の進化によって「新標準非準拠」データになってしまう可能性がある．Webデータを「新標準」に準拠させるために，大きな労力を掛けてWebデータを作り直す必要があり，時間的制約や金銭的制約などの点から困難であることが多い．従来のWebオーサリングツールはある時点のWeb標準に準拠したWebデータの効率作成に重点を置いているが，「標準」の進化に伴う既存Webデータのアップグレードについては配慮していないか，配慮していても対応が十分でないなどの問題がある．そこで，本研究ではWebページのレイアウトデザインを実際のWebデータ作成から完全に切り離し，デザインデータを単独で管理する手法を提案し，試作システムを実装した．本システムを用いて評価実験を行い，提案手法によってWeb標準の進化に素早く対応可能で且つ簡易・効率的なWebオーサリングが出来ることを確認し，その有効性を考察した．

1　はじめに

Web標準とは，国際的な標準化団体によって策定された，Webで標準的に利用されている技術や仕様の総称であり，代表的な存在としてW3C（World Wide Web Consortium）が策定・勧告しているWeb作成に関わる（X）HTML，CSSなどがある．Web標準はアクセスした利用者の誰もが容易に情報を共有できる状態にあることを考慮して策定されているため，Webデータの制作はWeb標準に準拠すれば，そのデータに含まれる情報がアクセスされやすくなり，Webアクセシビリティの向上や検索エンジン最適化（SEO：Search Engine Optimization）にも繋がると考えられる［1］［12］［16］［23］．しかし，これらの「標準」自体も進化しており（図１），バージョンアップする際に仕様が大幅に拡張・修正されることがあるので，Webページを作成する時点での「標準」に基づいて作成した「標準準拠」データが「標準」自体の進化により「新標準非準拠」データになってしまい，ユーザは不利益を被る可能性を常に持っている．

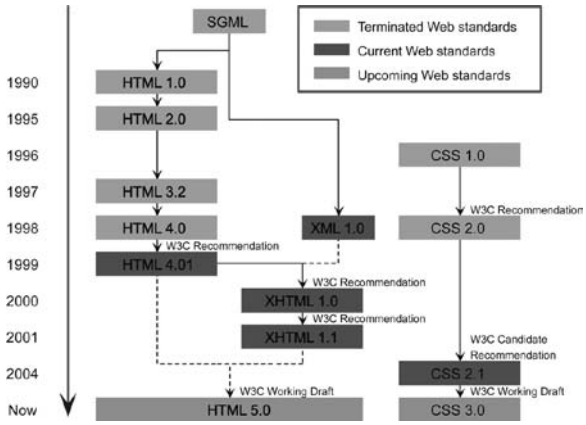


図１　Web標準の進化

例えば，CSSがW3Cに策定・勧告された以前では，段組（Multi-Column）などのような複雑なWebページレイアウトを実現するために，<table>要素を利用するのは「標準」であったが，1996年，文書の構造と体裁を分離させるという理念の影響を受け，CSSが「標準」としてW3Cにより策定・勧告された後，「表」として使用されるべき<table>要素をレイアウトに使用することが（X）HTML本来の目的から外れており，「Webに表示されるものの意味を実際に伝えるものではない．構造という面から見ると，でたらめに並べられた文字列と何ら変わらない」［9］．その結果として，以前テーブルタグでレイアウトを組んだWebデータは全て「新標準非準拠」データになり，CSSで作成した「標準準拠」データと比べるとコードが煩雑であるため，検索ロボットに読み取られにくく，検索に不利となった．

また，もう一つの例として，HTML5が策定される以前は，音声（オーディオ）を再生するために，JavaScriptでFlash，Real Player，Windows Media Playerなどの外部プラグインを埋め込んで，音声ファイルを制御する方法は一般的であったが，HTML5では新たに<audio>タグが追加され，このタグのsrc属性に音声ファイルのURLを記入するだけで，音声を再生することが可能である．JavaScriptで書いたデータと比べ

て，<audio>タグで書いた「HTML5準拠」データのソースコードは明らかに簡潔明瞭であり，また外部プラグインに依存しないメリットもある．

これらの問題を解決するために，Webページの内容とレイアウトデザインは変わらないという前提で，新しい「標準」に対してWebデータを適用させることが可能な構造をとり，且つアップグレードを適宜行える必要がある．

そこで，Webデータをアップグレードする際に，如何にユーザに掛かる負担を軽減できるかを検討する必要がある．

現在，Webデータの作成に於いて様々な手法・ツールが研究・開発されている．[2] [3] [4] [5] [6] [10] [11] [13] [14] [17] [21]．しかし，複雑なデザインを作成できる高機能の手法・ツールを利用するには(X)HTMLやCSSなどの専門知識の習得が必要であり，ユーザにとっては敷居が高いものとなってしまっている．ユーザが専門知識を持たなくてもWebデータを作成できるテンプレートベースの手法・ツールも存在するが，デザインがテンプレートに制限されるので，ユーザがイメージした通りに作成できない場合があり，機能性とユーザビリティの両立が実現されていない課題がある．また，これらの手法・ツールの共通の問題として，ある時点のWeb「標準」に準拠したWebデータの効率的な作成に重点を置いているが，「標準」の進化に伴う既存Webデータのアップグレードについては配慮していない問題点が挙げられる．

そこで，筆者らはWebページのレイアウトデザインを実際のWebデータ作成から完全に切り離してデザインデータを単独で管理し，そしてデザインデータとWeb標準のマッピングにより「標準」の進化に対応できるWebデータ作成手法を提案し，さらにこの手法に基づいた機能性とユーザビリティの両立を実現した実験システムを実装した．

以下本稿では，まず2章で関連研究と既存Webオーサリングツールを論じる．続く3章で提案手法の概要を述べ，4章で実験システムの実装と評価について述べる．5章で今後の研究課題について触れ，6章で本論文をまとめる．

2 関連研究と既存Webオーサリングツール

(X)HTML，CSSなどのWebデータの作成を支援するために，様々な手法・ツールソフトウェアが研究・開発されている．それらはテキストエディタ型，レイアウトエンジンベース型，Webベース型の3つにおよそ分類することが可能である．

2.1 テキストエディタ型ツール（Text Editor）

インターネットの黎明期においては，テキストエディタ型（Text Editor）のWeb作成ツールが主流であった．テキストエディタ型ツールとは本来，(X)HTMLのタグやCSSの属性等を一文字一文字手打ちする作業を省いたり，整形したりするためのものである．具体的には，Webデータの種類による予約語の色分け（シンタックスハイライト），入力の補完，コンパイラや外部アプリケーションとの連携，自動インデントなどの機能が挙げられる．代表的なものはWindows用のTeraPad [17]，EmEditor [4]，Mac OS X用のmi [10]，skEdit [14]，Unix系OS用のKate [7]などが挙げられる．

テキストエディタ型ツールを利用することで，直接Webデータのソースコードを編集できるため，自由度のある高度なWebデータを作成できる一方，ユーザはソースコードを直接編集するための(X)HTML，CSSなどの専門知識が要求される．また，構造が複雑なWebページを作成するには大量のコーディング作業が必要になるため，ユーザにとって大きな負担になる．さらに，テキストエディタ型ツールには基本的にプレビュー画面は用意されておらず，ユーザはブラウザで表示を確認しながら編集作業を進めなければならないため，非効率的なWebデータ作成となる問題もある．

2.2 レイアウトエンジンベースWYSIWYG型Webオーサリングツール

テキストエディタ型ツールの問題点を改善し，WYSIWYG（What You See Is What You Get）のWebデータ作成を実現するために，一部のWebオーサリングツールにはレイアウトエンジン（Layout Engine or Rendering Engine）[1]をベースにしたプレビュー（or デザインビュー）機能を導入し，(X)HTMLとCSSへの編集結果がWebページレイアウトに対する影響をその場で確認することができるようなWeb作成環境を提供している．代表的なものには，Adobe Dreamweaver [3]，IBM Homepage Builder [6]，Microsoft Expression Web [5]などがある．

しかし，文献[15]で述べられているように，これらのツールにより段組などのような複雑なレイアウトデザインを持つWebページを作成する場合は，プレビュー画面でレイアウトを確認しながら，(X)HTMLタグ及びCSSの細かい設定・入力を行わなければならないため，専門知識を持たないユーザにとって簡単ではない．また，これらのツールは高機能を備える反面，利用方法自体が複雑であるため，ユーザにとって習得作業が負担となる．

1 レイアウトエンジン（Layout Engine or Rendering Engine）は，Webページ記述用マークアップ言語（HTML，XMLなど）とそれを修飾するフォーマット情報（CSS，XSLなど）で書かれたデータを実際に画面に表示するための計算を行うプログラムである

２.３　WebベースWebオーサリングツール

２.１と２.２の共通の問題として，Webデータを作成・更新するたびに，FTPなどのツールを利用して，サーバからデータファイルをダウンロード，アップロードといった煩雑な作業を行わなければならない．これは日々の更新作業に於いて，ユーザに大きな負担を掛ける．この問題を解決するために，近年，オンライン編集可能なWebベースWebオーサリングツールが研究・開発されている．

CrespoらはHTML知識の待たないユーザのために，WebブラウザベースのWebページ・Webアプリケーション制作ツールWebWriterを提案・開発した［2］．ユーザはHTMLの専門知識がなくても，WebWriterにある「コンテンツ追加メニュー」をクリックしてコンテンツの種類を決定し，そしてその属性を指定すればテキストや画像などのコンテンツを表示するためのHTMLコードがCGIプログラムによりWebページに自動的に追加される．非常に理に適ったツールであるが，CSSに対応しないため，構造的に複雑なWebページが作れない問題が挙げられる．

西らが提案している手法では，DOMとJavaScriptの併用によりWebページのHTMLとCSSを操作し，ユーザが行った一連の編集がHTMLとCSSのファイルに直接反映させずに，別途CGIによりHTMLとCSSを編集するためのアクションを外部JavaScriptファイルとして生成し，さらにこのJavaScriptファイルをHTMLが読み込むことによってWebブラウザにおけるWYSIWYGのWebデータ編集を実現している［11］．しかし，この手法で編集作成したWebページを見る毎に，ブラウザにおけるJavaScriptの実行による編集環境が必要であるため，特に編集内容が多い場合にはそのWebページの表示が遅くなることが考えられる．また，編集結果は直接HTMLのソースコードに反映されないため，そのWebページが表現したい内容を適切に検索エンジンへ伝えられない問題もある．

岡田らは，構造化された妥当なXHTMLコンテンツを容易に生成するために，WebブラウザのDOM APIを利用したWebブラウザベースの構造化エディタを提案している［13］．このエディタでは，ユーザがWebブラウザの表示上の最上部にあるドロップダウンメニューと最下部にあるノードパスバーを操作することにより，コーディングなしで整形式（Well-Formed）のXHTMLを作成・編集できる．適用できる範囲はXHTMLデータに限られているので，ドキュメントの構造的な表現とその内容以外編集・作成できない問題がある（例えば，段組みのレイアウト表現）．また，DOM APIの仕様はW3Cに定義されているが，実装は各メーカーに委ねられており，各ブラウザが採用しているDOM APIの間に多くの実装の違いが存在している．そのため，DOM APIベースのWebデータ作成はユーザが利用するブラウザによって異なる可能性がある（例えば，ウィンドウ内周（コンテンツ表示領域）の高さを取得するためのプロパティであるinnerHeightは，Netscape Navigator4.X以降のNetscape，Firefoxを始めとしたMozilla系ブラウザ，Opera，Safariでは対応しているが，Internet Explorerでは対応していない）．

また，関連研究としては，2005年より一般的に普及したコンテンツマネージメントシステム（Content Management System，CMS）がある．例としてCunninghamらが提案しているWiki［21］が挙げられる．既存のWebページを，マークアップ言語を用いてWebブラウザから編集できる特徴があるが，文書の書き換えに重点を置いているので，画像を多用し，レイアウトが複雑であるデザイン重視のWebページ作成が困難である．また，特有の文書マークアップはHTML などと比べて簡潔であるが，マークアップそのものの習得はユーザにとって負担になる．その他CMS ではテンプレートベースのBlog などが挙げられる．熟練Webデザイナーによって事前に作ったWebページのテンプレートをベースにしたWeb作成環境を提供しているので，HTML，CSS などの専門知識を持たないユーザでも，テキストや画像等のコンテンツを用意すれば，Webデータを作成できる．しかし，それはWebデータ作成時にレイアウトなどの編集がテンプレートに制限されるという意味合いも持つ．

加えて，以上の既存研究・ツールは何れも共通として大きな問題を抱えている．ある時点のWeb標準に準拠したWebデータの簡易・効率作成に重点を置いているが，１章にて述べたWeb標準の進化に伴う既存Webデータのアップグレード問題について，配慮していないか，配慮しても十分でないなどの問題がある．

Webデータのアップグレード問題について配慮していないツールを利用する場合は，新しい「標準」に合わせて既存データをアップグレードするには，ユーザ自ら一件ずつ編集する必要があり，ユーザにとって大きな負担になる．それは一から作り直すと表現しても過言では無い程の重負荷である．

一方，Dreamweaverなどの一部のツールの新しいバージョンでは既存（X)HTMLデータのソースコードを他標準への「変更（Convert)」機能が追加されている．しかし，この機能による変更はソースコードに含まれるタグやDOCTYPE宣言などの記述の置換に過ぎず，標準進化に従って完全に対応したとは言えない．例えば，‹table›要素で作った段組構造のHtmlデータをHtml4.0以後の標準へアップグレードする場合は，本来その段組構造を構成する‹table›タグを削除し，CSSで作り直さなければならないが，これらのツールを用いて「変更」する場合は，ツールがソースコードだけでその中に含まれる‹table›要素は「表」その物を表すために使っているか，或いは段組構造を実現するために使っているかと判断できないので，

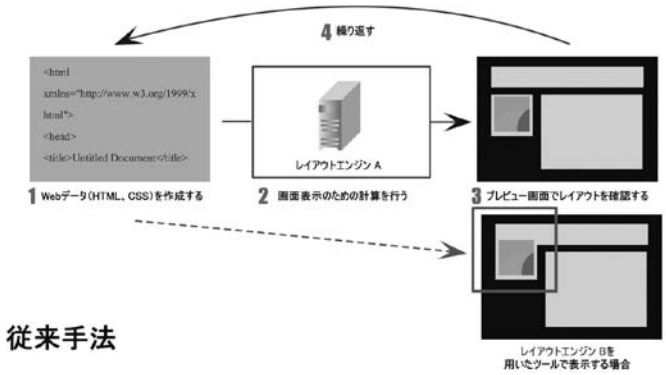


図2　コード編集をベースにした従来手法

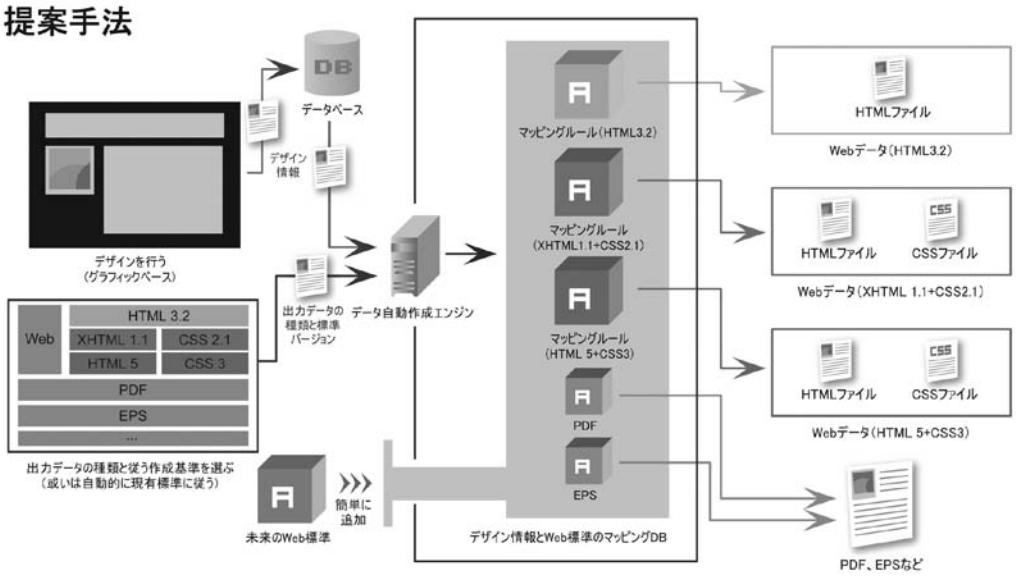


図3　提案手法

<table>要素は変更されずに，そのまま残ってしまう．また，この「変更」作業はWeb標準の進化に従って自動的に実行されるのではないため，ユーザ自らデータファイルを一つずつ開いて行わなければならない．

3　提案手法

2章で述べた問題点を解決し，Web標準の進化を対応可能で且つ簡易・効率的なWebデータ作成を実現するために，筆者らは従来の研究（図2）とは異なる視点に立って，Webページのレイアウトデザインを実際のWebデータ作成から完全に切り離す手法を提案している（図3）．

3.1　提案手法の概要

Webページには「見える部分」と「見えない部分」がある．「見える部分」とはWebページがブラウザなどのツールで開くときに画面に出力されるコンテンツ（テキスト，画像など）とレイアウトなどの視覚的表現である．「見えない部分」とはWebページに記述された，(X)HTMLタグやCSSなどである．検索エンジンや視覚障害者にも対応するために，Web標準に従ったWebデータ作成は不可欠であるが，一般のエンドユーザにとってWebデータの「見えない部分」より「見える部分」が重要であることは述べる迄もない．つまり，Web標準が進化した場合は「見えない部分」を新しい標準に合わせて作り直す必要があるが，「見える部分」を変える必要がない．

従来の手法ではWebデータの「見えない部分」のソースコードを直接弄ることで「見える部分」における視覚的表現の編集を実現する．しかし，この故にソースコードの変動に伴い，「見える部分」も変わってしまう可能性があるので，「見える部分」が変わらない前提で，「見えない部分」だけを作り直すWebデータのバージョンアップ作業を従来の手法で行うのは簡単ではない．特に，Webページの視覚的表現は(X)HTMLとCSSなどの複数のWebデータによって実現される場合，なおさら煩雑となる．

そこで，提案手法では，まずエンドユーザ（制作者）によりWebデータの「見える部分」を作成する．

作成した「見える部分」をデザイン情報としてデータベースに保存し，単独で保管する．そして，デザイン情報と出力作成時点でのWeb標準バージョン情報をWebデータ自動作成エンジンに送信し，デザイン情報とWeb標準のマッピングデータベースに基づいて自動的に実際のWebデータを作成する．Web標準がバージョンアップするたびに，デザイン情報と新しいWeb標準のマッピングルールをマッピングデータベースに追加するだけで，エンドユーザが以前作ったWebデータの「見えない部分」が新しい標準に従って自動作成エンジンにより自動的に一括で再生成でき，エンドユーザには負担を一切掛けない．

また，本手法ではWebデータの「見えない部分」が全て自動作成エンジンにより自動的に生成されるので，エンドユーザが（X)HTML，CSSなどの専門知識を持たなくても，デザインだけを行えば，Webデータが作成可能である．

3.2 「見える部分の作成」（ユーザによるレイアウトデザイン）

ユーザはGUI（図4）を利用して下記のように自らイメージしたWebページのレイアウトデザインを行う．

3.2.1 ページのデザイン

ユーザはGUIの最下部に位置する「Property Bar」（図4－1）を利用して，ページの幅（width），高さ（height），寄せ（align），バックグランド（background），ボーダー（border），マージン（margin）などを設定し，デザインを行う．ページの幅と高さは数字（絶対的な長さ）で設定しても良いし，パーセンテージ（画面の大きさとの比率）で設定しても良いし，設定しなくても良い．設定しない場合はコンテンツの内容に従って自動的に決まる．

3.2.2 コンテンツの追加とレイアウトデザイン

GUIの左部分に用意されているツールバー（Toolbar）（図4－2）を使って，テキストコンテンツと非テキストコンテンツ（画像，映像，Flash）などをページに追加し，さらに直感的操作でコンテンツのサイズ（size），位置（location）を設置する．また，「Property Bar」の併用でテキストのフォント（font），サイズ（size），色（color），ハイパーリンク（Hyperlink），バックグランド（background），ボーダー（border）などのレイアウトデザインを合わせて行うことが可能である．

図4　Webデザイン用GUI

3.2.3 デザイン情報の管理

Webページのレイアウトデザインが完了した後にSave Button（図4－3）をクリックしてデータベースにデザイン情報を送信し，単独で保存を行う．保存したページを編集・更新する際には，デザイン情報をデータベースよりGUIに読み込んでから行う．

デザイン情報はContents List，Structure Tree，及びProperties List の3部分によって構成される．Contents List はWebページに含まれているコンテンツの一覧である．コンテンツがユーザにWebデザインGUI に追加される際，そのコンテンツのタイプ，ファイルアドレスなどの基本情報がインデックス付きでContents List に追加される．Contents List はWebページに含まれるコンテンツの絶対状況を表すものであるため，Webページに同じコンテンツが複数追加されても，Contents List に一つしか登録しない．また，Structure Tree はWebページを構成するコンテンツ間の論理的な構造を記録する．Properties List はStructure Tree のノードにあるコンテンツがそのノードにおいての属性情報を記録する．

3.3 「見えない部分の作成」（デザイン情報とWeb標準情報に基づいたWebデータ自動生成）

デザイン情報がデータベースに保存された後にシステムはWebデータ自動作成エンジンにデータ作成の命令を指示する．Webデータ自動作成エンジンによる「見えない部分」の作成は以下の3段階で実現する．

3.3.1 デザイン情報とWeb標準情報の取得

デザイン情報はWebデータ自動作成エンジンによりデータベースから取得する．その際，Webデータの「見えない部分」の生成基準であるWeb標準情報はWebデータ作成時点の最新基準をデフォルトと設定しているが，ユーザが別標準でデータを作成することも「Property Bar」（図4－1）から可能である．

3.3.2 マッピングルールの選定

Webデータ自動作成エンジンは取得したWeb標準情報に基づいてデザイン情報とWeb標準のマッピングDBから適切なマッピングルールを選定する．デザイン情報とのマッピングルールはWeb標準ごとに作成する．図5はデザイン情報とそれに対応するWeb標準のマッピングルールの一例を示す．ユーザが画像コンテンツflower_1.jpgをWebページに追加

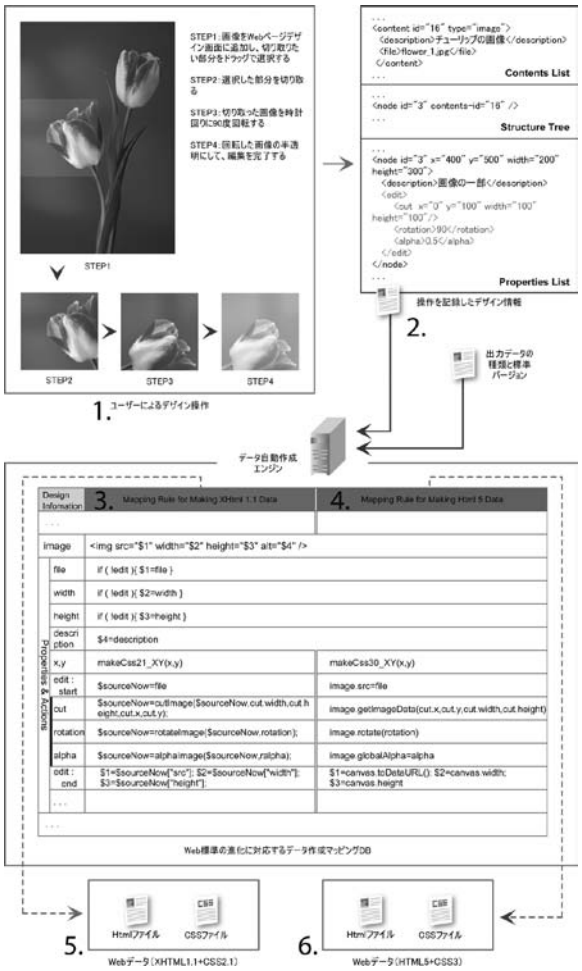


図５　デザイン情報とWeb標準のマッピング

し，そしてその画像コンテンツに対して行う「切り取り」，「回転」，「透明化」の一連デザイン操作（図５－１）は，図５－２のようなデザイン情報としてデータベースに保存される．図５－２のデザイン情報に対応するXhtml1.1標準（以下，標準A）とHtml５標準（以下，標準B）のマッピングルールはそれぞれ図５－３ の列（以下，ルールA），図５－４の列（以下，ルールB）に示す[2]．画像要素のマークアップは標準Aと標準Bに於いて，共に<img>となっていることから画像要素のマークアップを生成する変換式はルールAとルールB共通のとなる．$1，$2，$3，$4はそれぞれ画像のソースファイルアドレス，幅，高さ，説明文を保存する変数で，画像に対する編集がなかった場合（相応デザイン情報に<edit>がない場合）は，デザイン情報のcontents list→content→file，property list→node→width，property list→node→height，property list→node→descriptionにより値を取る．また，画像に対する編集があった場合（相応デザイン情報に<edit>がある場合），標準Aに於いては画像編集のための要素がないため，本ツールで別途用意してい

Design Infomation	Mapping Rule for Making PDF Data (Using FPDF Class)
・・・	
image	$pdf->Image($1,$2,$3,$4,$5)
Properties & Actions: file	if (!edit){ $1=file }
Properties & Actions: width	if (!edit){ $4=width }
Properties & Actions: height	if (!edit){ $5=height }
Properties & Actions: description	PDF_create_annotation($pdf,x-10,y-10,x,y ,'Text',description)
Properties & Actions: x,y	$2=x,$3=y
Properties & Actions: edit : start	$sourceNow=file
Properties & Actions: cut	$sourceNow=cutImage($sourceNow,cut.width,cut.height,cut.x,cut.y);
Properties & Actions: rotation	$sourceNow=rotateImage($sourceNow,rotation);
Properties & Actions: alpha	$sourceNow=alphaImage($sourceNow,ralpha);
Properties & Actions: edit : end	$1=$sourceNow["src"]; $2=$sourceNow["width"]; $3=$sourceNow["height"];
Properties & Actions: ・・・	
・・・	

図６　PDF出力用のマッピングルール

る画像作成エンジンによりデザイン情報に基づいた新しい画像ファイルの自動作成を行い，新しい画像のファイルアドレス，幅，高さを変数$1，$2，$3に返す．一方，標準Bには画像編集用のcanvas要素が新たに追加されているため，別途新しい画像を作成する必要はなく，代わりに標準Bの仕様に従って「画像編集用」と「変数$1，$2，$3への値設定用」canvas操作スクリプトをWebデータの中に生成する．

マッピングルールを追加すれば，デザイン情報に基づいてWebページ以外の仕様のデータも作成できる．例えば，図５のデザイン情報に基づいてPDFを作成するためのマッピングルールは図６に示す．

３．３．３　Webデータの自動生成

自動作成エンジンはデザイン情報とWeb標準のマッピングルールに基づいてWeb標準に準拠したWebデータを自動的に生成する．Webデータを生成する際に，自動作成エンジンはまずデザイン情報を一行ずつ順番に読み出す．次に，読み出した情報を持って３．３．２で選定したマッピングルールの先頭から順に走査して最初にマッチしたものから実際のデータを作成するための変換式を獲得する．そして，変換式を使って実際のWebデータを生成する．

また，インターネットにあるWebページをブラウザで閲覧する際には，まずWebデータをダウンロードすることから始まるため，Webデータのファイルサイズは小さければ小さいほどダウンロード時間を短縮

2　ルールAとルールBの比較を見やすくするために，ルールAとルールBにおいて同じである「デザイン情報とのマッチパターン」部分を一列にした

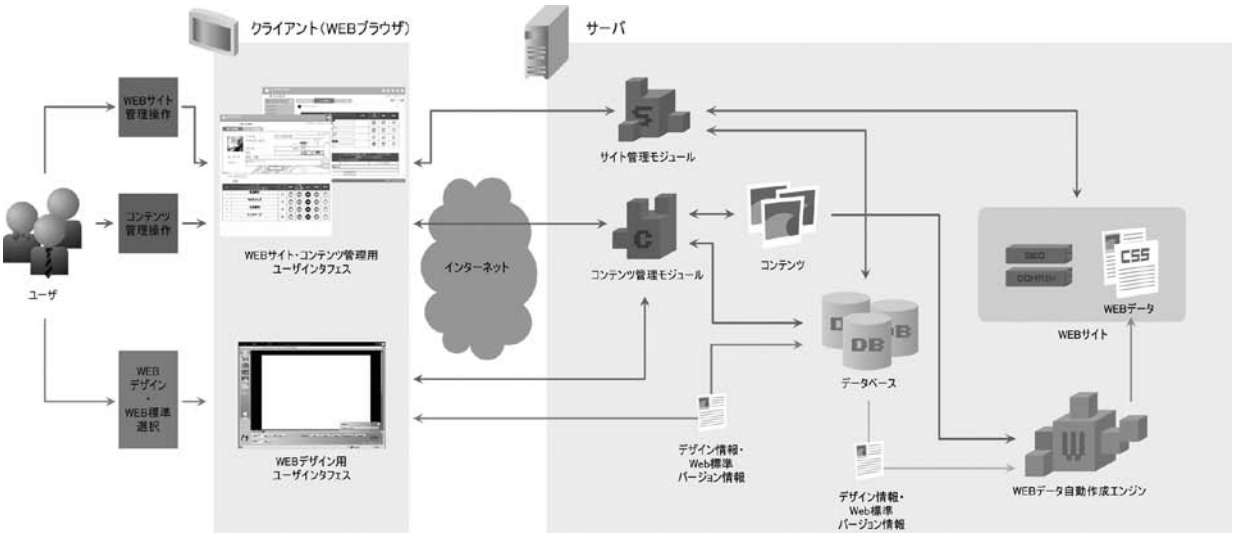


図7　実験システムの構成

することができ，読み手の満足度向上につながる．本ツールでは，自動作成エンジンがデータを生成する際に「HTMLタグの簡易化」や「CSSのグループ化」などの手法を使ってファイルサイズの軽量化も合わせて行う．

4 実装と評価

4.1 実験システムの実装

筆者らは3章の提案手法に基づいて実験システムを実装した．システムのアクセシビリティとユーザビリティを向上させ，システム自体のインストールと作成したWebデータをインターネットにアップロード・ダンロードにかかる手間省略を考慮し，実験システムはRIA（Rich Internet Application）として開発を行った．RIAはクライアント/サーバシステムのユーザインターフェースの豊かな表現力・操作性と，Webシステムのアプリケーションの容易な展開・運用性を持ち合わせており，Adobe社により提案されている技術である．

図7に示すように，実験システムはサーバ側のサイト管理モジュール，コンテンツ管理モジュール，Webデータ自動製作エンジンとクライアント側のWebサイト・コンテンツ管理用インターフェス，Webデザイン用ユーザインターフェースによって構成される．サーバ側の各モジュールとクライアント側のWebサイト・コンテンツ管理用インターフェスはPHPにより開発し，Webデザイン用ユーザインターフェースは，直感的な操作環境を実現するために，オープンソースフレームワークAdobe Flexを用いて実装している．Adobe Flexで作成されたアプリケーションはFlash プレイヤー上で動作するため，Flash プラグインがブラウザにインストールされていればOS に依存せず動作する．

また，デザイン情報とWeb標準のマッピングルールはHtml3.2（過去），Html4.01＋CSS2.0（現在），Xhtml1.1＋CSS2.1（現在），Html5＋CSS3（未来）の主となる4種類を用意し，デフォルトのWebデータ作成標準はXhtml1.1＋CSS2.1と設定している．

なお，Webサーバソフトウェアは Apache HTTP Server，データベースは Sun Mysqlを利用している．

4.2 評価

本手法の特徴を以下に述べる．本手法によって作成したWebデータはWeb標準の進化に従い自動的にアップグレードできるため，ユーザには一切負担を掛けない．また，ユーザは直感的な操作で「見える」部分のデザインだけを行えば，Webデータの「見えない」部分も自動的にWeb上で生成されるため，ユーザはFTP，HTML，CSSなどの専門知識を学ぶ必要がなく，Webデータ作成・更新時のコーディングやデータアップロード・ダウンロードなどにかかる手間と時間も節約できる．本手法で採用しているデータ作成手法は拡張性に優れる．例えば，現在のWeb標準ではWeb作成時の操作履歴を記録するタグが定義されていないため，Webデータの中に一連の編集動作が反映できずに，Webデータをある動作時点以前の状態へ戻すのは困難である．従来の一部の手法・ツールにおいてはWebデータを作成・編集するときの履歴を一時的にメモリに保存することでこの問題を解決しているが，編集したデータファイルを一旦保存して閉じると，一連の操作履歴もその時点でメモリから消えてしまい，編集途中の状態に戻れなくなる．本手法においてはデザイン時の情報に基づいて自動的にWebデータを生成する手法を採用しているため，現有のデザイン情報の中に操作履歴を記録する部分を追加すれば，Webデータが以前のどの操作時点の状態にも戻れると考えられる．もう一つの例を挙げると，現時点では本ツールによってWebデータしか作成できないが，デザイン情報とのマッピングルールを追加すれば，PDFやEPSなどのデータの作成も実現できる．

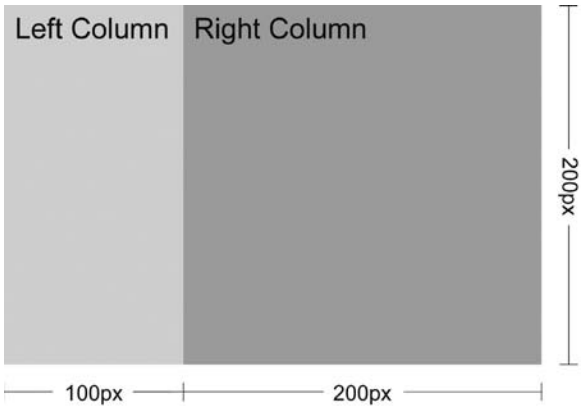


図８　２段組構造のWebデザイン例

本手法の有用性・有効性を検証し，本手法に基づいて開発した実験システムによって，妥当性がありかつ進化するWeb標準に準拠したWebデータを効率的に作成できることを確認するために，筆者らは評価実験を行った．

４.２.１　実験１

段組構造Webページの作成は最も有名な課題の一つとしてWeb制作の世界でよく知られている．その原因は作成方法自体が煩雑であるだけではなく，作成方法がWeb標準の進化に従って大きく変わっていることにも関係している．例えば，図８のような２段組のレイアウトを実現するには，Html3.2準拠のHtmlコードが

```
<table width="300" border="0" cellspacing="0"
cellpadding="0">
 <tr>
    <td width="100" height="200" valign="top"
bgcolor="#CCCCCC">Left Column</td>
    <td width="200" height="200" valign="top"
bgcolor="#999999">Right Column</td>
 </tr>
</table>
```

Xhtml1.1＋CSS2.1準拠のXhtmlコードが

```
<div id="left-col">
        Left Column
</div>
<div id="right-col">
        Right Column
</div>
```

CSSコードが

```
#left-col {
        float: left;
        height: 200px;
        width: 100px;
        background-color: #CCCCCC;
}
#right-col {
        float: left;
        height: 200px;
        width: 200px;
        background-color: #999999;
}
```

Html5＋CSS3の場合のHtmlコードが

```
<div id="container">
  <div id="left-col">
    Left Column
  </div>
  <div id="right-col">
    Right Column
  </div>
</div>
```

となり，CSSコードが

```
#container {
        display: -moz-box;
        display: -webkit-box;
}
#left-col {
        height: 200px;
        width: 100px;
        background-color: #CCCCCC;
}
#right-col {
        width: 200px;
        background-color: #999999;
}
```

となる．

この例で分かるように段組構造を実現するにはHtmlタグやCSSの細かい設定・調整が必要であるため，従来の手法・ツールで作る場合はユーザが一定以上のWeb作成専門知識と経験を持たなければならない．また，２章で述べたようにWeb標準進化による段組構造データのアップグレード問題は従来の手法・ツールでは未だ上手く解決されていない．

本手法で作成したWebデータが実証１を満たしているか否かを検証するために，まず実験システムのWebレイアウトデザインGUIを利用し，テキスト，画像，FLASHなどのコンテンツを含んだ段組構造のWebページのデザインを行い（図９），そしてWeb

図９　段組構造のWebデザイン例

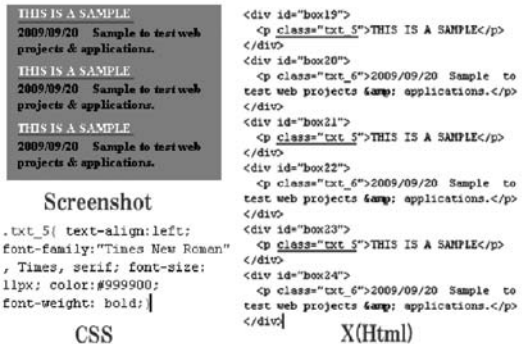


図10　CSSのグループ化

データ自動作成エンジンにより，Html3.2，Html4.01＋CSS2.0，Xhtml1.1＋CSS2.1，Html5＋CSS3のWeb標準でそれぞれ実際のWebデータの自動生成を行った．

そして生成されたWebデータをWindows，MacOS，Linux，FreeBSD，Solarisにおいて主流ブラウザのInternet Explorer，Mozilla Firefox，Opera，Safari，Google Chromeで表示を確認した．結果としては，Html3.2，Html4.01＋CSS2.0，Xhtml1.1＋CSS2.1の標準で生成されたWebデータが各種OS・ブラウザにおいて正しく表示できた．一方，現時点ではCSS3のdisplay: box 属性を対応しているブラウザはFirefox，Safari，Google Chrome の三つしかないため，Html5＋CSS3で生成されたデータがInternet Explorer，Operaで表示されるとき，レイアウトが崩れたことを確認した．この問題の解決策としてはHtml5＋CSS3がW3C RecommendationになるまでXhtml1.1＋CSS2.1をデフォルト出力標準にすることが考えられる．また，Web標準の策定機関W3Cが提供している公式妥当性検証ツールのMarkup Validation Service(X(HTML) 用) [19] とCSS Validation Service（CSS 用）[18] を利用して，自動生成されたWebデータの妥当性検証を行い，実験システムで自動生成された４種類のデータがそれぞれ出力する前に決まった標準に準拠していることが確認できた．さらに，生成したCSSとX(HTML) を確認すると，Webページにある同じスタイルを持つ要素はグループ化されていることが分かる（図10）[3]．このことから，ファイルサイズ縮減に成功していることが分かる．以上の評価から本手法で作成したWebデータは妥当・効率的かつWeb標準に準拠していることが証明されている．また，一つのデザイン情報から３種類のWeb標準に準拠したWebデータを自動的に作成出来ていることから，本手法がWeb標準の進化に対応可能であることが証明されている．

４.２.２　実験2

本手法の効率性を検証するために，ユーザ実験を併せて行った．関連研究［11］において西らは受験対象をWebページを作成した経験があり，HTML，CSS及びFTPに関して十分な知識を持つWeb作成経験者とWeb作成専門知識・経験を持たない小学生の２グループ（A）（B）に分けて，それぞれに「タイトル」，「顔写真」，「氏名」，「年齢」，「血液型」，「生年月日」及び「50字程度の自己紹介文」によって構成する「自己紹介ページ」を作成させる．作成されたWebページの例を図11に示す．なお，実験結果の誤差を抑えるために，グループ（B）の被験者を選出する際には基本的なキーボード，マウス操作ができることを前提条件として設けている．また，「自己紹介ページ」の構成要素である「顔写真」と「自己紹介文」は事前に用意した．実験が始まる前に，グループ（A）と（B）の被験者に対してそれぞれ30分の講義を行い，被験者全員に本実験の目的と本ツールの使用方法の説明を行う．実験中に不明な点及び質問がある場合には，指導者に質問可能とする．

関連研究の手法と比較するために，本研究で十分な専門知識を持つWeb制作経験者とWeb制作専門知識・経験を持たない小学生の２グループの被験者を集め，関連研究［11］と同じ内容の実験を行った．

実験結果：グループ（A）の６人の被験者が作成にかかった合計時間コストは19分24秒，平均時間コストは３分14秒．グループ（B）の23人の被験者が作成にかかった合計時間コストは４時間３分48秒，平均時間

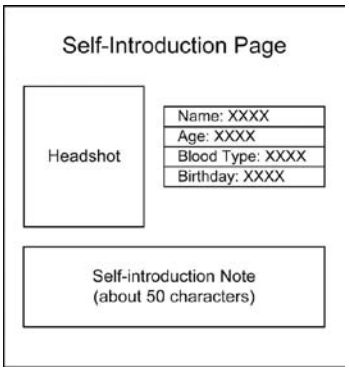
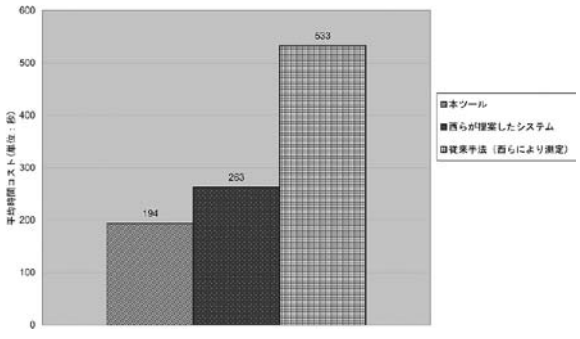
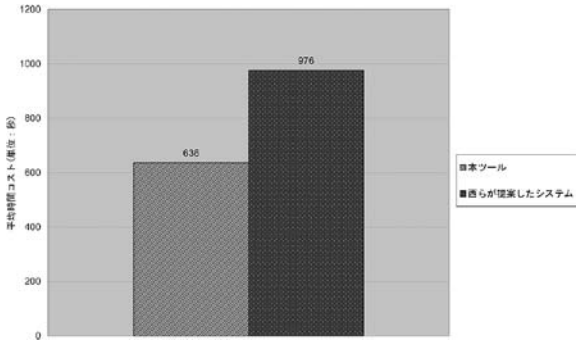


図11　ユーザ実証実験で作成されるWebページ例

3　ソースコードを見やすくするために，改行とスペースを挿入している



（a）Web制作経験者



（b）Web制作未経験者

図12　Web作成にかかる時間コスト

コストは10分36秒であった．この測定結果を①西らが提案した手法，②西らにより測定された従来手法[4]による作成時間と比較する（図12）[5]．

図12に示すように，Web制作経験の有無に関わらず，本手法を利用する場合は，西らが提案した手法より大幅にコストが削減できることが確認できた．また，Web制作経験者を対象とした実験では，本手法の利用によって，西らが定義した「従来手法」より明らかにコストを低減したことを判明した．その理由は従来の手法・ツールはソースコードの編集を通じて，Webデータの視覚的表現を実現・調整しているため，段組のような複雑な構造を持つWebページを作成するにはユーザから多少直接コードの部分を触らなければならない．特に近年Webページが（X)HTMLとCSSの複数のデータに構成されているので，それぞれのデータを編集するために，複数編集画面間の切り替えは頻繁にやらなければならなく，手間も時間もかかる．また，殆どの従来ツールはWebデータを作成・編集した後に，FTPなどのツールを使ってサーバへアップロードする必要があり，非常に煩雑な作業にもなる．西らなどが提案した手法はDOM操作を通じてWeb上で直接Webデータのソースコードを編集することができ，ファイルアップロードなどの手間を減らしたが，可能な編集動作はソースコード及びDOMの仕様に制限されているため，効率的な編集は実現しにくい．本手法の場合はFlashの特徴を生かして開発したGUIを利用して制限なしのデザインが行える．また，デザイン作業が終わった後，Webデータの作成はWeb上で自動生成されるので，ユーザにとっては一切負担が掛からなく，Web編集及び作成にかかる時間が削除できた．

また，本手法を用いることによる，Webデータの準拠仕様を変更する際の効率具合を評価するために，本研究でHTML，CSSなどのWeb技術を精通し，３年以上Webデータ制作の経験を持つWebデザイナーを４人集め，実験を実行した．実験の事前準備として，まず試作システムで図９のデザインを行い，HTML3.2，XHTML1.1＋CSS2.1の標準でそれぞれのWebデータ（A）と（B）を作成した．これらのデータは自動作成エンジンによって自動的に生成され，生成に係る時間はそれぞれ0.0221秒，0.0234秒であった．本手法による準拠仕様変更はソースコードを修正することではなく，自動作成エンジンにより変更後の標準に従って新たにソースコードの生成を行うことから，生成にかかる時間は仕様変更に係る時間と置き換えることが可能である．評価実験においては，まず被験者の４人を２人ずつランダムにグループ（A）と（B）に分け，グループ（A）へWebデータ（A）を，グループ（B）へWebデータ（B）を渡す．次に，グループ（A）の被験者はWebデータ（A）をXHTML.1＋CSS2.1の仕様に変更させ，グループ（B）の被験者はWebデータ（B）をHTML3.2の仕様に変更させる．被験者がWebデータを作成する際に使うオーサリング方法については指定せず，被験者にとって一番早くWebデータを作成できる方法を利用することを要求した．グループ（A）と（B）の被験者が制作にかかった平均時間コストはそれぞれ27分47秒と23分05秒であった．上記の結果から，Webデータの準拠仕様を変更する際においても本手法が効率的であることが証明された．

以上のことから本手法の効率性が証明された．

5　今後の課題

現時点では本手法により作成できるものは静的なWebデータに限られている．静的なWebデータはSEO（検索エンジン最適化）上に有利であるが，EコマースやSNSなどのWebサービスにおいては，情報の検索や情報の頻繁的な更新に対応するためにデータベースと連動する動的なWebデータを扱わなければならない（PHP，JSP，ASPなど）．そのため，プロ

4　西らが定義した「従来手法」とは，受験者が普段おこなっているWebオーサリング方法である．「普段行っている方法」とは，受験者が一番早くWebデータを作成できると考える方法を指す（Webデータのソースコードを記述する方法や，Webオーサリングツールを用いる方法など）

5　厳密解ではないが比較のため参考値として図に示す

グラミングとデータベースの専門知識が不要なデータベース連動型Webアプリケーション作成を実現出来れば，社会の情報化推進に大きく貢献できると考えられ，今後の課題として研究を深める必要がある．

6 まとめ

Webデータの作成はWeb標準に準拠することが重要である．しかし，Web標準自体が進化しているため，「標準準拠」データが「標準」の進化により「新標準非準拠データ」になってしまう可能性が常にある．従来のWebデータ作成に関する研究・ツールは，ある時点のWeb標準に準拠したWebデータの効率作成に重点を置いて検討されているが，Web標準の進化に伴う既存Webデータのアップグレードについては配慮していないか，配慮していても対応が十分でないなどの問題がある．

この問題の解決策として，本論文ではWebページのレイアウトデザインを実際のWebデータ作成から完全に切り離し，デザインデータとして単独で保存し，そしてデザインデータとWeb標準のマッピングにより自動的にWebデータを生成する手法を提案し，Web標準の進化に対応可能で且つ簡易・効率的なWebオーサリングができる実験システムをRIAとして開発及び評価を行った．本手法に於いては，Web標準が進化した場合でもデザイン情報と新しいWeb標準のマッピングルールをマッピングデータベースに追加するのみで，エンドユーザが既に用意したWebデータの「見えない部分」を新しい「標準」に従って自動作成エンジンにより自動的に一括で再生成可能であり，エンドユーザには負担を一切掛けることはない．また，本手法ではWebデータの「見えない部分」が全て自動作成エンジンにより自動的に生成されるので，エンドユーザが（X)HTML，CSS，FTPなどの専門知識を持たなくても，Flexで開発した高機能GUIを用いて直感的デザイン操作のみを行うことによってWeb側で高度なWebデータが作成され，従来のWebオーサリング手法・ツールで解決できなかった機能性とユーザビリティの両立を実現している．それらは4章の評価実験により，有効性と妥当性を確認している．今後の課題として，検証精度向上を目標とした評価項目の詳細検討にユーザの労力（習得時間，作業時間等）及び構想実現満足度を含めたアンケート調査の検討を行うこと，及びデータベース連動型のWebアプリケーションの自動作成等が挙げられる．

参考文献

[1] Cederholm, D.: Web Standards Solutions: The Markup and Style Handbook（Special ed.), ISBN-13: 978-1430219200, friends of ED, 2009

[2] Crespo, A. and Bier, E. A.: WebWriter: A browser-based editor for constructing Web applications, in Proc.the 5th W3 Conference, Paris, France, 1996, pp. 1291-1306.

[3] Adobe Dreamweaver, http://www.adobe.com/products/dreamweaver/

[4] EmEditor, http://www.emeditor.com/

[5] Microsoft ExpressionWeb, http://www.microsoft.com/japan/products/expression/products/web overview.aspx

[6] IBM Homepage Builder, http://www-4.ibm.com/software/webservers/hpbuilder/

[7] Kate, http://kate-editor.org

[8] Musciano, C. and Kennedy, B.: HTML & XHTML: The Definitive Guide（6th ed.), ISBN-13:978-0596527327, O' Reilly Media, 2006

[9] Meyer, Eric A.: CSS: The Definitive Guide（3rd ed.), ISBN-13: 978-0596527334, O' Reilly Media,2006

[10] mi, http://mimikaki.net/

[11] 西健太郎, 新谷虎松, 松尾徳朗, 田代慎治, 伊藤孝行：既存Webブラウザを利用したオンライン編集可能なWebページの実現, 電気学会論文誌（部門誌）C, vol125, no 4, Apr 2005, pp. 660-665.

[12] 日本Webアクセシビリティ協会, Web標準に準拠するメリット, http://www.jawaa.or.jp/WebStandards/Merit.html

[13] 岡田尚希, 伊藤一成, DURST, M. J.: Webにおける構造化エディタを用いたコラボレーション環境の実現, 情処研報, 2007-6（DBS-141, GN-62), 2007, pp. 103-108

[14] skEdit, http://www.skti.org

[15] Sullivan, S.: Creating a simple threecolumn design with CSS and Dreamweaver 8, http://www.adobe.com/devnet/dreamweaver/articles/three column layout.html

[16] Thatcher, J. et al.: Web accessibility: web standards and regulatory compliance, ISBN-13: 978-1590596388, friends of ED, 2006

[17] TeraPad, http://www5f.biglobe.ne.jp/ t-susumu/library/tpad.HTML http://www.microsoft.com/expression/

[18] W3C, CSS Validation Service, http://jigsaw.w3.org/css-validator/

[19] W3C, Markup Validation Service, http://validator.w3.org/

[20] W3C, HTML 4.01 Specification, http://www.w3.org/TR/html4/present/graphics.html#h-15.2

[21] wiki, http://c2.com/cgi/wiki

[22] Zhu, J., Liu, X., Urano, Y. and Jin, Q. : A Novel WYSIWYG Approach for Generating Cross-Browser Web Data, in Proc. 2010 International Conference on Computational Science and Its Applications（ICCSA 2010, Fukuoka, Japan, 2010, pp. 155-164

[23] Zeldman, J. and Marcotte, E.: Designing with web standards（3rd ed.), ISBN-13: 978-0321616951, New Riders Press, 2009