研究指導　中澤 真　准教授

WEB サイトのマルチデバイス化を図る効率的移行手法 ～大学の WEB サイトを事例に～

榎本舞　　佐藤李歩

1. はじめに

近年，iPhoneをはじめとしたスマートフォンやタブレットなどのスマートデバイスが急速に我々の身の回りで普及している[1]．それに伴い，利用者が所有する電子機器の操作環境は，パソコンでのマウスクリックに加え，スマートデバイスを指でタップするという操作が登場し，ユーザーインターフェースなど WEB サイトの閲覧環境が多様化した．

これに対し，多くの WEB サイトではスマートデバイスでアクセスした際，PC 用のサイズのまま表示され，拡大しないと文字が小さくて見えにくいなど，スマートデバイスへの対応が遅れている．これは現行の WEB サイトがスマートデバイスの急速な普及に追い付けず，WEB サイトを改修しようとしても膨大な時間や費用，人員が必要となってしまうことに二の足を踏んでしまうことが原因である．

そこで本研究では，HTML4 で構築された公立法人会津大学短期大学部の WEB サイトを事例に，マルチデバイス化する際の移行作業量を分析する．この結果に基づき，修正作業量を抑えつつ，スマートデバイスにも対応できる操作性を備えたサイトを構築するための効率的移行手法を提案する．

2. HTML4 で構築された WEB サイトの問題点

2.1 スマートデバイスの普及と WEB サイトの対応の遅れ

インターネット白書 2012 によると，2010 年時点でのスマートフォンの出荷台数は 684 万台，タブレットの出荷台数は 129 万台であった．また，2013 年ではスマートフォンの出荷台数は 2812 万台，タブレットの出荷台数は 761 万台である．さらに 2016 年ではスマートフォンは 3559 万台，タブレットは 1510 万台にまで上昇する見込みである [2]．これらのスマートデバイスの主要な用途は，情報検索や WEB サイトの閲覧であるにもかかわらず，現行の多くの WEB サイトはこの急激なスマートデバイスの普及に対応が追い付いていない．

それを顕著に表したものが，日経 BP コンサルティングが調査した「全国大学サイト・ユーザビリティ調査」である[3]．調査対象は全国の国立大学 66 校，公立大学 18 校，私立大学 127 校の計 211 校となっている．スマートフォン非対応の大学 WEB サイトは全体の 108 校，メニューのみのスマートフォン対応は 22 校，コンテンツもスマートフォン対応になっているのは 81 校のみであった．このようにスマートデバイスに完全に対応できている大学が半数にも満たないことからも WEB サイト運営側の対応が非常に遅れていることがわかる．

この結果，スマートフォンの利用者は WEB 閲覧時に不便を強いられることになる．図 1 はこれらの利用者の WEB 閲覧時の不満な点を示したものであるが，「文字が小さすぎて見にくい」が 69.2%，「指先での画面の拡大・縮小をするのが面倒」が 56.1%など，多くの不便な点が挙げられた．このことから，現行のWEBサイトはスマートフォンユーザーにとって多くの不満が残されていることがわかる．

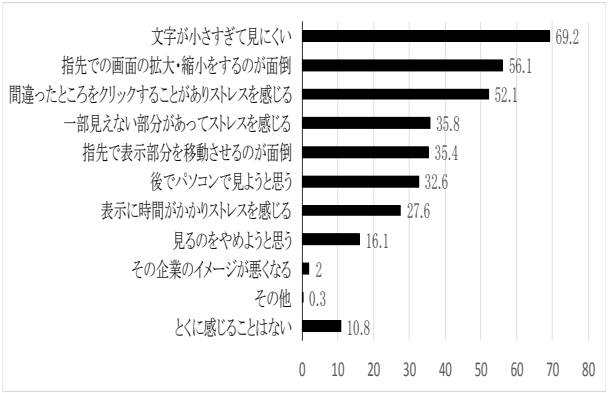


図 1:PC サイトをスマートフォンで閲覧した際の不満[4]

2.2 HTML4 とマルチデバイス化

スマートデバイスに対応できていない WEB サイトの多くは，HTML4 で構築されている．対応できていない原因の 1 つが，HTML4 にはサイズ指定をするタグが多数存在していることにある．このようなタグを用いて PC 用にサイズ指定したサイトを構築すると，画面サイズの異なるスマートデバイスでは適正なサイズで表示されなくなってしまう．これを回避してマルチデバイス化するためには，少なくとも同じ WEB サイトを PC 用，タブレット用，スマートフォン用と 3 つ用意し，それらを JavaScript などで自動的に切り替えるようにする必要がある．しかし，ページの更新作業は 3 倍かかることになり非常に効率が悪く，しかも同じ内容のページが複数存在することは SEO[1]に対しても問題が多い．そこで，これらの問題を解決するために登場したのが HTML5 である．

HTML5 は HTML4 に代わるマークアップ言語とし

[1] Search Engine Optimization：検索エンジン最適化．検索結果の上位に自らの Web サイトが表示されるように工夫すること．

て 2008 年に草案が発表され，2014 年に正式な仕様として勧告される予定の WEB ページを構成するための言語である．ユーザーの多様な閲覧環境に応じて最適化されたページが自動表示される仕組みを備えているため，サイト管理者がデバイス別，OS 別に複数のページを用意する必要もなくなるという大きな長所を持つ[5]．しかし，HTML5 ではマルチデバイス対応のために CSS で代用可能なタグがすべて廃止され，HTML4 で広く利用されていた画面分割のレイアウトを構成するための Frame タグも廃止されるなど，HTML4 と異なる部分が多い．このため，HTML4 から HTML5 への移行には，多くの手間と時間がかかってしまう．

そこで，本研究では，既存の HTML4 を可能な範囲で利用しながら，レスポンシブ WEB デザイン[2]対応が可能な CSS3 を用いてマルチデバイス化への移行を図る場合と，HTML5 と CSS3 を組み合わせた場合の作業工数を分析・比較し，効率的な移行のための手法を明らかにする．

2.3 本学 WEB サイトの問題点

マルチデバイス対応における現行 WEB サイトの問題点を明らかにするために，研究の題材として公立法人会津大学短期大学部の WEB サイトを用いることにする．

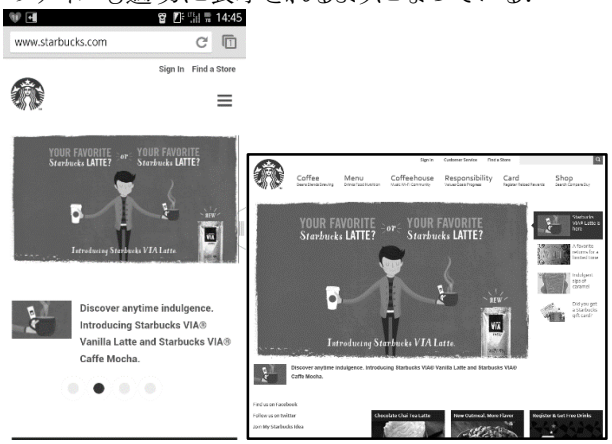
1 つ目の問題点は Frame タグを使用していることにある．これは HTML5 では廃止されたタグであり，このタグを残したままマルチデバイス対応させることはできない．以前は画面の分割レイアウトを構成するために Frame タグがよく用いられていたが，その画面サイズの設定が固定化されているため，仮に PC サイズで構築されていた場合にはスマートデバイスでは適切なサイズで表示されなくなってしまう．このため，閲覧者は頻繁に表示サイズの拡大・縮小を繰り返すことになり，著しくユーザビリティが低下してしまう．

2 つ目の問題点は HTML 内でのレイアウト指定である．画像や表などのサイズ指定を HTML 内で設定してしまうと，そのレイアウトはどのデバイスから閲覧しても変化しないため，マルチデバイス化を図ることができなくなってしまう．次章ではこれらの問題を解決するための基本的な考え方について述べる．

3 本学 WEB サイトにおける課題解決策

先に述べたように，WEB サイトのマルチデバイス化を図るにあたり，本研究ではレスポンシブ WEB デザインという手法を採用する．この手法のメリットとしては，単一のファイルでページレイアウトに関する設定をするため，情報の更新作業が容易になることである．また，すべてのデバイスで同一の URL が表示されるため SEO にも適している．

レスポンシブ WEB デザインの実例として，スターバックスコーヒーカンパニー株式会社[3]の WEB サイトを図 2 に示す．この図 2 からわかるように単一ファイルで作成されているにもかかわらず，デバイスごとに異なる画面が表示されていることが確認できる．また，スマートフォンで閲覧した際のタップ操作に対応できるよう，通常より大きいボタンが配置され，文字や画像のサイズも適切に表示されるようになっている．



スマートフォンサイト　　PC サイト

図 2:レスポンシブ WEB デザイン実装サイトの事例

図 2 のようなマルチデバイス対応サイトを構築する場合，一番小さいウィンドウサイズであるスマートフォンサイズの WEB サイトを基本として考える必要がある．スマートフォンは表示できる情報量が最も少ないため，これに適したシンプルな形で情報を配置することが重要である．また，メモリサイズの小さなスマートフォンは，ページの読み込む速度が遅くなるという問題もあるため，この点からも情報量を最小限に抑えたものを基本形とする必要がある．

スマートフォンサイズ用の WEB サイトの基本形を構築後，より画面サイズの大きいミディアムサイズ及びラージサイズの 2 種類のページレイアウトを CSS を用いて追加構築することが理想形である．この 2 種類のサイズを利用者のデバイスに応じて自動的に切り替える機能がメディアクエリである．メディアクエリとは，利用者が所持しているデバイスの画面サイズなどに応じて CSS の適応を変化させる条件式のことである．例えば，1024px 以下のウィンドウサイズに適応できる設定の条件式は以下の通りである[6]．

```
@media only screen and (max-width:1024px)
```

---

2 様々な種類のデバイスのインターフェースや画面サイズに応じて，WEB ページの表示が最適になるように単一のファイルで対応する手法．

3 http://www.starbucks.com/

図 3 と図 4 は画面サイズを 1024px 以下と 1025px 以上に場合分けし，各々の画面サイズに応じてメディア特性を設定するようにした事例である．近年は画面サイズが異なったスマートフォンが増加しているため，製作者と利用者の双方にとって，このメディアクエリは重要な機能となっている．そのため本研究でもメディアクエリを利用し，マルチデバイス化を図る．

```
@media only screen and(maxwidth:1024px){
nav#mainNav{
clear:both;
width:880px;
margin:0 auto 10px;
border:1px solid #e2d7c2;
}
```

図3:画面サイズが1024px以下のメディアクエリ

```
@media only screen and (min-width: 1025px){
nav#mainNav{
clear:both;
position:relative;
z-index:200;
width:100%;
}
```

図4:画面サイズが1025px以上のメディアクエリ

なお，本研究では WEB ページの表示結果を検証するウェブブラウザとして，Internet Explorer10 以上，Firefox，Google Chrome の 3 種類を用いる．理由としては 2013 年 11 月時点において，ブラウザシェアが 50%程度であるが[4]，いずれも HTML5 と CSS3 の多くの項目をサポートしており[7]，今後主流のブラウザとなることが確実であるからである．

3.1 トップページとメニューバーの移行作業

まず，トップページのマルチデバイス化のために，ページ全体のレイアウトについて考える．既存サイトでは画面を分割レイアウトするために Frame タグを使用していた．しかし，このタグは HTML5 で既に廃止されているため利用することは適切ではない．そこで Frame タグをすべて削除し，代替手段として HTML5 から新たに追加された<nav>タグや<section>タグを設置することで WEB サイトの構造を明確化し，CSS3 でこれをレイアウト設定できるようにする．<nav>タグはナビゲーション領域を表すために使用され，この後述べるメニューを構成する際にも利用する[8]．<section>タグは文章の意味や構成によって領域を区別するために使用され，文書構造が一目でわかるようにすると同時に，ページデザインを構造に応じて設定できるようにする役割を持っている．

次にメニューバー部分をマルチデバイス化するための代替案として，アコーディオンメニューとプルダウンメニューを組み合わせた方式とワンキャンバスレイアウト方式の 2 つの手法を検討する．

3.1.1 アコーディオン・プルダウンメニューを用いたケース

現行の WEB サイトは，サイト訪問者のタイプに応じて選択するメインメニューと，実際の各項目を選択するサブメニューで構成されている．これらのメニューは PC で閲覧した場合でもスマートデバイスで表示した場合でも変わることはない．しかし，スマートデバイスのタップ操作や画面サイズの特徴を考えると，異なるデバイスで同じナビゲーションメニューを使用するのは適切ではない．そこで，PC サイズ用にはプルダウンメニュー，スマートフォンサイズにはアコーディオンメニューと，異なるナビゲーション機能を HTML5 で実現する手法を考える．アコーディオンメニューとはタップやクリックをすると，図 5 のようにサブメニューが表示される仕組みのことである．

図 5:スマートフォンサイトのアコーディオンメニュー

[4] http://news.mynavi.jp/news/2013/12/04/255/

```
<nav id="mainNav">
<div class="inner">
<aclass="menu" id="menu"><span>MENU</span></a>
<div class="panel">
<ul>
<li>
<a href="subpage.html"><strong>総合</strong></a>
<ul class="sub-menu">
<li><a href="subpage.html">短大ガイド</a></li>
<li><a href="subpage.html">学科・カリキュラム</a></li>
<li><a href="subpage.html">卒業後の進路</a></li>
<li><a href="subpage.html">施設案内</a></li>
<li><a href="subpage.html">入試案内</a></li>
</ul>
</li>
```

図 6:メニュー部分の HTML ソース

このメニューは限られた領域でメニュー項目を大きく表示できるため，画面サイズの小さいスマートフォンに適したインターフェースとなっている．一方，メニュー項目が多い場合には縦に長い構成となるため，PC で用いた場合には横方向の領域を有効に利用できず，スクロール操作を利用者に強いることになってしまい，適切な手段ではない．

もう一つのプルダウンメニューとはクリックした場所から下に垂れ下がるように表示されるメニューのことである(図 7).

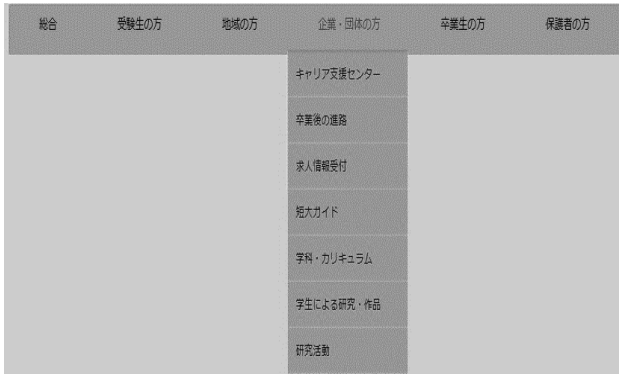


図 7:PC サイトのプルダウンメニュー

このメニューの場合，最上位の階層のメニュー項目を横に配置するため，画面サイズの大きな PC で利用するのに適したインターフェースとなっている．しかし，横幅のサイズが狭いスマートフォンには，メニューをすべて画面内に表示できない場合が生じるため不適切である．以上のことから，PC 用にはプルダウンメニューを，スマートデバイス用にはアコーディオンメニューを採用することにした．

PC 用のナビゲーションとしてプルダウンメニューを採用することのメリットは他にもある．これまでの Frame タグで構成されたサイトではサブメニューが常時表示されるため，トップページにおいて一定の領域を占領していた．これに対してプルダウンメニューでは，クリック又はオンマウスされたときのみ一時的に表示されるため，領域を有効に活用することができる．加えて，このプルダウンメニューはオンマウスでサブメニューが表示されるため，1回ずつメニューをクリックする手間も省くことが可能となる．

3.1.2 ワンキャンバスレイアウトの実装

ワンキャンバスレイアウトとは，スマートフォンのような狭い画面サイズではコンテンツの一部を画面の外に配置し，ボタンをクリックすることで隠れた部分を表示する仕組みのことである[6]. 図 8 に示したように，初期状態ではメニューのないトップページを表示し，特定部分がタップされた場合にのみメニューを表示させる．結果として，狭い画面領域を無駄なく活用することができるようになる．PC のように画面が広いデバイスの場合には，メディアクエリを用いてメニューバーを常に表示させることも可能である(図 9)．

ナビゲーションインターフェースとして，ワンキャンバスレイアウトと先に述べたアコーディオンメニューはユーザビリティ的に一長一短がある．また，移行のための作業量も，どちらも新規にスタイルシートを構築しなければならないため大差ないことが確認できた．本学サイトの場合では，メニューの項目数がかなり多く，メニュー表示が縦に長くなり過ぎてしまう可能性があるため，アコーディオンメニューを使用するのが適切である．

図 8:スマートフォンサイトの
ワンキャンバスレイアウト

図 9:PC サイトのワンキャンバスレイアウト

3.2 新着情報リストの移行作業

本学の WEB サイトにはトピックス[新着情報]の部分が存在するが[9], この部分は, 古い情報を消し, 新しい情報を手動で更新する仕組みになっている. その作業は一から構築するため手間がかかる. よって, 効率的な作業を行うにあたり object タグを代替した手法とボックスを構築する手法を比較検討する.

3.2.1 object タグの使用をする手法

object タグを用いて本学の WEB サイトの新着情報の部分を構築した場合, 頻繁に更新する情報を独立した HTML ファイル上で編集でき, 運用上のメリットが得られる. 一方, object タグは埋め込む HTML ファイルの情報量にかかわらず, object タグの属性として設定したサイズでしか情報が表示されない. 現在のサイトでは新着及び重要なお知らせは, 常にトップページ上に表示される設計となっているが, これを実現するためには掲載すべき情報の件数に応じて, object のサイズを常に調整する作業が発生してしまう. これは非常に手間のかかる作業となってしまうため, 総合的に判断すると object タグを用いた手法は適切ではないと結論付けられる.

3.2.2 ボックスで領域を設定する手法

ボックスを構築する方法は, CSS を使用して[新着情報]の領域を確保するものである. 現行の WEB サイトと大きく異なる点は table タグを使用していないところにある. table タグをページのレイアウト調整や位置調整のために使うことは可能であるが, 強引に位置や大きさなどを設定してしまうため, マルチデバイス対応の妨げになってしまう.

一方, CSS で特定のボックス領域を設定した場合は, デバイスに応じてボックス領域の位置や大きさを動的に調整することができるようになる. このため, 本学サイトの新着情報の表示領域はこの手法を用いるのが適切である.

メニューなどのナビゲーションや新着履歴などのトップページのマルチデバイス化は, HTML のファイル数が数個であるため移行作業量に関しては大きな問題にはならない. そのため, マルチデバイス化した際のユーザビリティのみを評価対象として移行のための手段を選択すればよい.

4. 本学の WEB サイトの各ページのマルチデバイス化を図る効率的移行手法

トップページやメニューの移行作業と異なり, メニューのリンク先にある膨大なページについてのマルチデバイス化に関しては, その移行のための作業量をいかに効率化するかが重要な課題となる. 表 1 は 2014 年 1 月末時点における本学 WEB サイトのファイル数を示したものであるが, これを見ても大量の HTML ファイルに対して, マルチデバイス対応のための作業が必要となることがわかる. しかも, この表には PDF ファイルや, 一時的な情報として掲載されている新着情報などの HTML は除かれているため, 作業量はさらに増える可能性がある.

表 1:本学の WEB サイトのファイル数

1ファイル中に含まれる情報	ファイル数
文字のみ	97
画像のみ	11
表のみ	8
文字＋画像	229
文字＋表	32
画像＋表	13
文字＋画像＋表	10
文字＋画像＋オーディオ	2
文字＋画像＋インラインフレーム	2
文字＋テキスト入力欄	9
合計	413

これらのページの移行作業の手段としては 2 つ考えられる. 1 つ目は HTML5 にこだわらずに現行の HTML を極力維持し, レスポンシブ WEB デザインで設計された WEB サイトにするために最低限必要な作業だけをする方法である. これは図 10 の(ア)の方法である. HTML4 のサイズ指定などの情報を削除し, これらの情報を CSS3 へと移す作業が必要となる.

2 つ目の方法は次世代を見据えて完全に HTML5 の仕様に適合する WEB サイトへと移行する方法である. これは図 10 の(イ)の方法であり, HTML5 と CSS3 を組み合わせてレスポンシブ WEB デザインを実現する.

(ア) HTML4+CSS3
(イ) HTML5+CSS3

図 10:マルチデバイス化のための 2 つの手法

本研究ではこれら 2 つの手法を作業工数で比較する. HTML ファイルのマルチデバイス対応作業は, ページレイアウトに関する作業, 表部分に関する作業, 画像部分に関する作業の 3 つに分類することができる. 最終的な総移行作業工数は次式によって求める

ことができる. この数が少なく, かつ構築された WEB サイトのユーザビリティが一定の品質を保つことができるような手法を選択することが本章での目的である.

総移行作業工数=

(ページレイアウトのための移行作業工数×該当 HTML ファイル数)

+(画像のための移行作業工数×該当 HTML ファイル数)

+(表のための移行作業工数×該当 HTML ファイル数)

4.1 ページレイアウトに関する移行作業

まず, ページレイアウトをマルチデバイス化するのに必要な移行作業内容とその作業工数について, 2 つの手法別にまとめた結果を表 2 と表 3 に示す. ここで, 作業工数は 1 つのタグあるいは属性を削除・追加・コピー&ペーストした場合を 1 とする. なお, 同じ内容を複数削除する場合には, エディタの置換機能を用いることで一括削除ができるものと考え, この場合も1とカウントすることとする.

表 2:ページレイアウトに関する移行作業工数 (HTML4+CSS3)

ページレイアウト (HTML4+CSS3)	作業目的	作業内容	作業工数
HTML	マルチデバイス対応	サイズ指定されたタグ・属性の削除	14
		画像による装飾の削除(境界線など)	1
		metaタグ内にviewport属性値を追加	1
		tableタグの撤去とその他のタグ撤去	6
		CSSへのリンク設定	1
	効率的な運用のための メニューファイルの外部化	予め用意したJavaScriptファイルに メニューバーのみを 保存したファイル名を追加する。それをメ ニューバーを表示したいページにコピーする。	1
	HTML4の要素追加	追加	1
	見出しと境界線のマルチデバイス化	<h1>追加	1
CSS	全体設定	背景画像の設定	1
		maginの設定(左右)	2
	見出しと境界線のマルチデバイス化	border-bottom:4px dotted red;を<h1>に追加	1
	マルチデバイス対応	メディアクエリの設定	1
		maginの設定(右左)	2

表 3:ページレイアウトに関する移行作業工数 (HTML5+CSS3)

ページレイアウト (HTML5+CSS3)	作業目的	作業内容	作業工数
HTML	マルチデバイス対応	サイズ指定されたタグ・属性の削除	15
		画像による装飾の削除(境界線など)	1
		metaタグ内にviewport属性値を追加	1
		tableタグの撤去とその他のタグ撤去	27
		CSSへのリンク設定	1
	効率的な運用のための メニューファイルの外部化	予め用意したJavaScriptファイルに メニューバーのみを 保存したファイル名を追加する。それをメ ニューバーを表示したいページにコピーする。	1
	HTML5の要素追加	<section>追加	1
	見出しと境界線のマルチデバイス化	<h1>追加	1
CSS	全体設定の追加	backgroundの設定	1
		maginの設定(左右)	2
	見出しと境界線のマルチデバイス化	border-bottom:4px dotted red;を<h1>に追加	1
	マルチデバイス対応	メディアクエリの設定	1
		マージン左右	2

表 2, 表 3 の結果を集計し, ページレイアウトのマルチデバイス化に必要な 1 ページあたりの作業工数を 2 つの手法で比較した結果が表 4 である. 重要なのはファイル数の多い HTML の小計値であり, この値を比較すると HTML4+CSS3 の作業工数が HTML5+CSS3 の作業工数の半分ほどであることが確認できる.

表 4:ページレイアウトの総移行作業工数の比較

ページレイアウト		HTML4+CSS3	HTML5+CSS3
HTML	コピー&ペースト	5	5
	削除	21	43
小計		26	48
CSS	コピー&ペースト	7	7
	削除	0	0
計		33	55

4.2 表に関する移行作業

次に, 表をマルチデバイス化するのに必要な移行作業内容とその作業工数について 2 つの手法別にまとめた結果を表 5 と表 6 に示す.

表 5:表に関する移行作業工数(HTML4+CSS3)

表 (HTML4+CSS3)	作業目的	作業内容	作業工数
HTML	マルチデバイス対応	サイズ指定されたタグ・属性の削除	1
		font size="2"(一括置換)	1
CSS	CSS	tableタグ	1
		trタグ	1

表 6:表に関する移行作業工数(HTML5+CSS3)

表 (HTML5+CSS3)	作業目的	作業内容	作業工数
HTML	マルチデバイス対応	サイズ指定されたタグ・属性の削除	2
		tableタグの装飾削除	6
	1セルごとの設定削除	bgcolor="cccccc"一括置換	1
		<font size="2" face="MS Pゴシック, Osaka">一括置換	1
CSS	マルチデバイス対応	table	1
		thタグ	1
		border:none	1
		trタグ	1
	tableタグの設定	border:1; cellpadding:2; を記述	2
	表設定	align:left	1
		valign:top	1
		background-color:#cccccc;	1
		background-color:#666666;	1
		align:center	1

表 5, 表 6 の結果を集計し, 表のマルチデバイス化に必要な 1 ページあたりの作業工数を 2 つの手法で比較した結果が表 7 である. この値を比較すると HTML4+CSS3 の作業工数が HTML5+CSS3 の作業工数より非常に少ないことがわかる.

表 7:表における総移行作業工数の比較

表レイアウト		HTML4+CSS3	HTML5+CSS3
HTML	コピー&ペースト	0	0
	削除	1	10
小計		1	10
CSS	コピー&ペースト	2	11
	削除	0	0
計		3	21

4.3 画像に関する移行作業

最後に，画像をマルチデバイス化するのに必要な移行作業内容とその作業工数について 2 つの手法別にまとめた結果を表 8 と表 9 に示す．

表 8: 画像に関する移行作業工数(HTML4+CSS3)

画像 (HTML4+CSS3)	作業目的	作業内容	作業工数
HTML	table代替要素追加	タグ追加	1
	1セルごとの設定削除	width="220"一括置換	1
		height="165"一括置換	1
		tr一括置換	1
		td一括置換	1
CSS	画像サイズ設定	max-width	1
		height	1

表 9:画像に関する移行作業工数(HTML5+CSS3)

画像 (HTML5+CSS3)	作業目的	作業内容	作業工数
HTML	マルチデバイス対応	画像装飾要素削除	6
	HTML5の要素追加	<fihgure>タグ追加	1
	1セルごとの設定削除	tr一括置換	1
		td一括置換	1
CSS	画像配置設定要素追加	HTMLソース内にある<br clear="all">をCSSに記述する。内容はclear:bothと記入。	1
	画像サイズ設定	max-width	1
		height	1

表 8, 表 9 の結果を集計し，画像のマルチデバイス化に必要な 1 ページあたりの作業工数を 2 つの手法で比較した結果が表 10 である．この値を比較するとHTML4+CSS3 の作業工数が HTML5+CSS3 の作業工数よりも少ないことが確認できる．

表 10:画像の総移行作業工数の比較

画像レイアウト		HTML4+CSS3	HTML5+CSS3
HTML	コピー＆ペースト	1	1
	削除	4	8
小計		5	9
CSS	コピー＆ペースト	2	4
	削除	0	0
計		7	13

4.4 検証結果

これらの結果に基づき，総移行作業工数を計算した結果が表 11 及び図 11 である．これらの検証結果を見ると，HTML5+CSS3 の約半分であることが明らかになった．特にすべてのページに対して必要となる，ページ全体のレイアウト設定をマルチデバイス化する作業が，もっとも差を大きくしている要因である．どちらの方法でレスポンシブ WEB デザインの WEB サイトへと移行しても，表示結果や操作性に関する部分ではほとんど差は無いため，効率化のみを重視するならば HTML4+CSS3 による移行が適している．ただし，長期的に見た場合，HTML5+CSS3 で構成されたサイトは，デバイスやブラウザなどプラットフォームがさらに多様化した場合にも柔軟に対応することができる．このため，まずは作業効率を優先して，HTML4+CSS3 による移行作業を済ませ，時間やコストをかけながら徐々に完全な HTML5+CSS3 へと移行するのが理想的である．

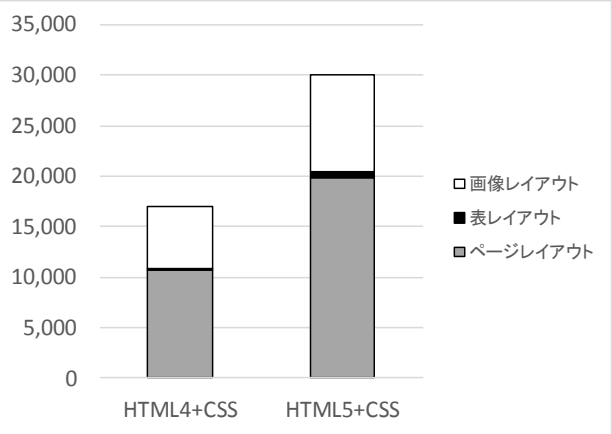


図 11:各項目の総移行作業工数の検証結果

表 11:各項目の総移行作業工数の比較

作業内容	該当ページ	HTML4+CSS		HTML5+CSS	
		作業工数/ページ	小計	作業工数/ページ	小計
ページレイアウト	413	26	10,738	48	19,824
表レイアウト	63	1	63	10	630
画像レイアウト	267	23	6,141	36	9,612
		合計	16,942	合計	30,066

5. むすび

本研究では，WEB サイトのマルチデバイス化を図る効率的移行手法の検証をした．効率的な移行作業を行うことで製作者側はマルチデバイス対応を可能とした WEB サイトを構築することができる．しかし，作業工数については比較検証ができたが，サイトのユーザビリティの比較については十分でない部分があった．また，運用時の情報更新作業の効率化についても，現行の WEB サイトと同程度のレベルでしか実現できなかった．

今後，更新作業の効率化のために MovableType などの CMS の実装も検討し，この場合の移行作業時がどの程度のものになるのかを明らかにする必要がある．

謝辞

本研究の調査にあたり，多大なご協力をいただいた公立法人会津大学短期大学部の実習助手の酒井円氏および産業情報学科デザイン情報コース 1 年生渡邉貴美子氏に厚くお礼申し上げます．

参考文献

[1] 総務省,平成 24 年版情報白書,ICT が導く震災復興・日本再生の道筋,2012.
[2] 総務省,インターネット白書 2012,財団法人インターネット協会,2012.
[3] 日経 BP コンサルティング,全国大学サイト・ユーザビリティ調査,http://consult.nikkeibp.co.jp/consult/report/uni/2013/
[4] 株式会社 D2C,モバイル利用動向調査(2012 年 8 月調査),http://www.d2c.co.jp/news/2012/20121018-1526.html
[5] TIME&SPACE,次世代 Web プラットフォーム「HTML5」が変える未来 PART1. HTML5 が秘める 3 つの「可能性」, http://time-space.kddi.com/special/specialreport/20130203/index.html
[6] 小川裕之,レスポンシブ web デザイン入門 マルチデバイス時代の web デザイン手法,マイナビ,2013.
[7] findmedyIP.com,HTML5&CSS3Support,http://fmbip.com/litmus/
[8] 片渕彼富,山田祥寛,HTML5 基礎,マイナビ,2011.
[9] 公立大学法人会津大学短期大学部,トピックス[新着情報],http://www.jc.u-aizu.ac.jp/