Vol.2016-GN-97 No.34
Vol.2016-CDS-15 No.34
Vol.2016-DCC-12 No.34
2016/1/22

多様な閲覧環境に対応する Web 制作を支援するアプリケーション

遠藤崇[†1]　速水治夫[†1]

概要：多様な閲覧環境に対応する Web 制作の手法としてレスポンシブ Web デザインがある．この手法では新しく登場する閲覧環境にも対応できる，コンテンツの更新が容易，URL が統一されるため検索ヒットに有利といった利点があり，注目されている．しかしその実装には 1 つの HTML ファイルで複数の閲覧環境に対応するために，環境ごとに表示確認を交互に行う手間，デザイン仕様の変更が多いといった課題がある．本研究ではこれらを解決するため，ブラウザ上で複数の閲覧環境ごとに表示確認とデザインの変更を実現するアプリケーションを提案する．本アプリケーションは Brackets の拡張機能として実装し，HTML のインラインフレームと Brackets のライブプレビュー機能を用いて，表示確認とデザインの変更機能を実現する．

キーワード：レスポンシブ Web デザイン，Web 制作，マルチデバイス，Brackets

An Application that support Web production Supporting various browsing environments

TAKASHI ENDO[†1]　HARUO HAYAMI[†1]

Abstract: There is a Responsive Web design as a method of Web production supporting various browsing environments. This method has been attention there are advantages such as can support also a newly appeared browsing environment, updating of contents easily, a searching hit advantageously because an URL is unified. However, its implementation has problems that labor to perform alternately confirmation display for each environment because supporting multiple browsing environments in a single HTML file, changing of design specifications often. To resolve these in this research, we propose an application that enables confirmation display and changing design for each browsing environment on browser. This application is implemented as an extension of Brackets, by using inline frame of HTML and Live Preview function of Brackets, confirmation display function and changing design one realize.

Keywords: Responsive Web Design, Web production, Multi-device, Brackets

1. はじめに

近年，スマートフォンやタブレットが登場し，Web サイトの閲覧にもよく使われるようになっている．総務省の調査[1]によれば，平成 26 年末のインターネット利用端末の種類は PC が 75.2%，スマートフォンが 47.1%，タブレットが 14.8%となっている．この他にもゲーム機が 7.5%，テレビが 5.0%と続いている．Web サイトの閲覧環境は多様化しており，制作を行う Web デザイナには多様な閲覧環境への対応が求められている．

対応する手法として 2 つの手法がある．振り分け型とレシポンシブ Web デザインである．

振り分け型は閲覧環境ごとに HTML ファイルとそれに対応する CSS ファイルを用意する手法と，レスポンシブ Web デザインは，HTML ファイルを 1 つとし，閲覧環境ごとに CSS ファイルを用意する手法の 2 つである．

振り分け型は JavaScript やサーバサイドでユーザエージェント情報から端末の名称や OS の名称などを基準に，用意された HTML ファイルに移動させる．

レスポンシブ Web デザインは画面解像度などを基準に使用する CSS を切り替えてレイアウトを変化させる．具体的にはフルードグリッド，フルードメディア，メディアクエリの 3 つの構成要素を用いて実装する[2].

フルードグリッドはレイアウトのガイドラインであるグリッドの幅を可変にし，画面の幅に応じてレイアウトの幅が変化するようにする．フルードメディアは，画像や動画などのメディアの幅と高さを可変にし，画面の幅に応じてメディアの幅と高さが変化するようにする．メディアクエリは画面解像度などの情報により使用する CSS を分岐する．

レスポンシブ Web デザインは振り分け型と比較した場合に次の利点がある[3].

- 端末や OS に依存しないため，新しく登場する閲覧環境にも対応できる
- HTML ファイルは 1 つであるため，コンテンツの更新が容易
- CMS のテンプレートも 1 つで済む
- URL が統一されるため，検索ヒットなどに有利

一方で，1 つの HTML ファイルで複数の閲覧環境に対応

†1 神奈川工科大学
Kanagawa Institute of Technology

Vol.2016-GN-97 No.34
Vol.2016-CDS-15 No.34
Vol.2016-DCC-12 No.34
2016/1/22

するため，画面解像度の異なる環境ごとに表示確認を行う手間があることや，デザイン仕様の変更が多いことが課題として挙げられる．

表示確認については振り分け型の場合，それぞれの閲覧環境に対応した HTML ファイルと CSS ファイルの組み合わせを制作し，組み合わせごとに正常な表示ができるかを確認する．レスポンシブ Web デザインの場合は，ある閲覧環境を想定した HTML ファイルと CSS ファイルを制作し，すべての閲覧環境で正常な表示ができるかを確認しながら，対応する閲覧環境の CSS を追加していく．ある閲覧環境に対して CSS を追加した際に，別の閲覧環境で表示が崩れることがあり，修正が必要になる．

デザイン仕様の変更については振り分け型の場合，閲覧環境ごとに HTML 構造が別々にあるため，他の閲覧環境を考慮することなくデザインができる．レスポンシブ Web デザインの場合は HTML 構造が共通であり，変更は他の閲覧環境に影響を与える．そのため，実装の困難なデザインが存在する．デザインしたものの実装が困難な際には，仕様を変更する必要がある．

こうした課題に対応した先行事例として，Dreamweaver CC のデバイスプレビュー機能, Responsive Design Testing, Responsinator がある．

Dreamweaver CC のデバイスプレビュー機能[4]はエディタである Dreamweaver CC と複数の閲覧環境を LAN によって接続し，エディタで開いている HTML ファイル，CSS ファイルをプレビューするものである．編集内容はリアルタイムに反映される．しかしこの方法は実機の用意が必要であり，コストが高い．

Responsive Design Testing[5]，Responsinator[6]はいずれも指定した URL の HTML ファイルを複数のインラインフレームによって複数の画面解像度で表示確認することができるものである．しかし，この方法では表示確認のみで，デザインを変更した際にはサーバにアップロードし，再度表示確認が必要になる．デザインの変更には対応できていない．

本研究では，ブラウザ上で複数の閲覧環境ごとに表示確認とデザインの変更を実現するアプリケーションを提案する．

本アプリケーションは, 普段からレスポンシブ Web デザインによる Web 制作を行う人を対象とし，レスポンシブ Web デザインにより制作された HTML ファイル, CSS ファイルを複数の画面解像度で表示する機能，CSS を編集する機能を提供する．

複数の画面解像度で表示する機能については，HTML のインラインフレームを用いる．同じ HTML ファイルを参照するインラインフレームを異なるサイズで複数表示することで実現する．

CSS を編集する機能は Web 制作に特化したエディタである Brackets[7]のライブプレビュー機能を用いる．ライブプレビュー機能は HTML ファイルをブラウザで開き，HTML や CSS の編集内容をリアルタイムにブラウザの表示に反映させる．

第 2 章では，アプリケーションの構成，ユースケース，ユーザインターフェースについて述べる．第 3 章では，評価実験の方法，結果，考察について述べる．

2. 提案アプリケーション

第 2 章では，アプリケーションの構成，ユースケース，ユーザインターフェースについて述べる．

2.1 アプリケーション構成

図 1 にアプリケーションの構成とフローを示す．本アプリケーションは Brackets の拡張機能と HTML で実装されたページで構成される．

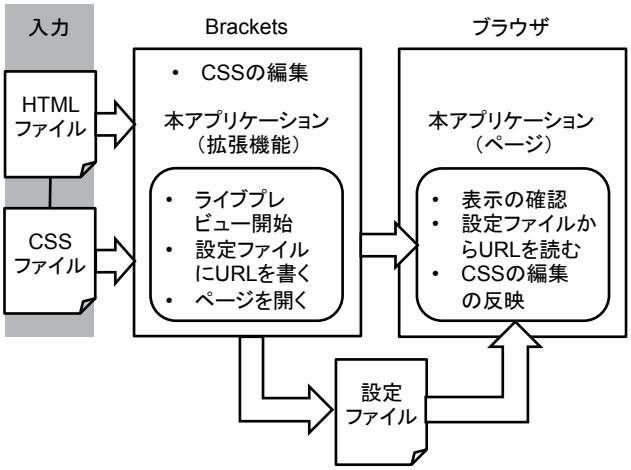

図 1　アプリケーションの構成とフロー

Figure 1　Flow and Composition of the Application.

拡張機能は Brackets のライブプレビュー機能の動作を変更する. Brackets では拡張機能向けに Live Preview API を提供しており[8]，これを用いる．拡張機能をインストールするとツールバーにボタンが追加され，ボタンが押されると API を用いてデフォルトの動作を変更し，ライブプレビューを開始する．変更後の動作はライブプレビューの URL を取得し，ページが読み込む設定ファイルに書き込み，ページをブラウザで開く処理にする．

ページは表示を確認するために使用され，ブラウザ上で使用する．ページには複数のインラインフレームがあり，フレームで表示する対象にライブプレビューの URL を設定することで，複数のライブプレビューが可能になる．ライブプレビューの URL は設定ファイルから読み取られ, 書き込みは拡張機能が行う．このページによって複数の画面解像度で表示する機能を実現する．ブラウザ上で表示確認をするのはブラウザの持つ HTML レンダリングエンジンを使用するためである．CSS を編集する機能は Brackets の

Vol.2016-GN-97 No.34
Vol.2016-CDS-15 No.34
Vol.2016-DCC-12 No.34
2016/1/22

エディタ機能を用い，編集内容の反映はページ上で行うことにより実現する．

ページには複数のインラインフレームを配置した表示確認画面と，表示確認画面で使用する画面解像度などを設定できる設定画面があり，最初は設定画面が表示される．画面遷移には CSS を用いている．

2.2 ユースケース

使用にあたってはあらかじめ制作した HTML ファイルと CSS ファイルが必要である．

ユーザは Brackets 上で HTML ファイルや CSS ファイルを編集する．表示確認をする際には，拡張機能により追加されたツールバーのボタンを押す．アプリケーションはページをブラウザで開き，設定画面が表示される．また，表示確認の対象とする HTML ファイルを自動で指定する．Brackets では HTML ファイルと CSS ファイルが自動的に関連付けられるため，CSS ファイルのみを開いている場合でも，表示対象とする HTML ファイルを指定できる．また，CSS ファイルは HTML ファイルから読み込まれるため，指定しない．

次にユーザは表示を確認したい画面解像度を指定する．デフォルトではスマートフォン，タブレット，PC を想定した画面解像度が指定されており，これに対して新しい画面解像度の追加や，削除が行える．指定を完了するとページは表示確認画面に遷移し，指定した複数の画面解像度で表示確認を行うことができるようになる．

表示確認を行い，表示が正常でない場合には CSS の編集を行う．Brackets で HTML ファイルのタグや，CSS ファイルのセレクタ，プロパティにカーソルを置くと，ページ内の各インラインフレームで編集箇所がハイライトされる．

ハイライトなどにより編集箇所を探し，Brackets で CSS を編集するとページ内の各インラインフレームにリアルタイムに編集内容が反映される．

2.3 ユーザインターフェース

図 2 に表示確認画面を示す．インラインフレームは 2 つあり，各フレームのサイズはフレーム上部のタブによって切り替える．タブは設定画面でユーザが指定した複数の画面解像度がタブになる．

レスポンシブ Web デザインでは，画面解像度の小さい閲覧環境向けの CSS を完成させてから差分として徐々に画面解像度の大きい閲覧環境向けの CSS を追記する．処理性能や回線速度の比較的低いスマートフォンなどは画面解像度の大きい閲覧環境向けの CSS を読み込まなくて済み，高速な表示ができるためである[3].

従って，差分となる閲覧環境を比較できるよう，2 つのインラインフレームを並べて表示するユーザインターフェースとした．

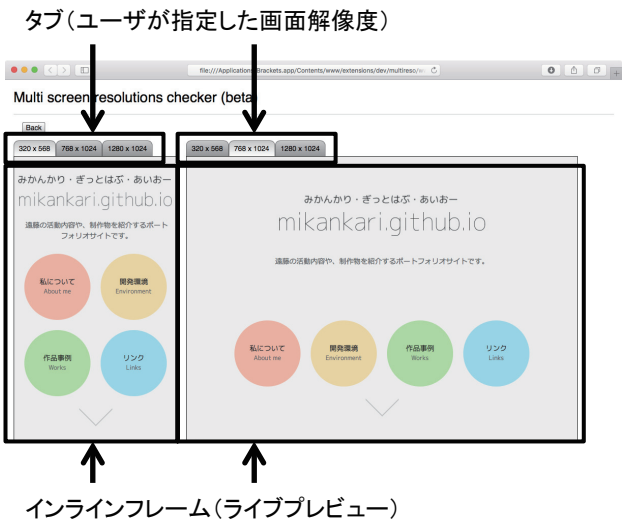

図 2　表示確認画面

Figure 2　Display confirmation page.

3. 評価実験

第 3 章では，評価実験の方法，結果，考察について述べる．

3.1 方法

提案アプリケーションにより，閲覧環境ごとの表示確認を行う手間が削減されたか，デザイン仕様の変更に対応できるようになったかを確認するため，普段からレスポンシブ Web デザインによる Web 制作を行っている大学生 3 名を対象に評価実験を行った．

実験は表示確認の方法について，普段使用している方法を用いる「普段」と，提案アプリケーションを用いる「提案」の 2 回，同じ Web サイトの実装を行った．実装はまず各方法について，あらかじめ用意した仕様に沿って実装する「制作」を行い，次に各方法について仕様変更を提示して実装する「修正」を行った．いずれもエディタは Brackets を用いる．4 通りの作業時間を計測し，最後にアンケート評価を行った．

それぞれ同じ Web サイトを実装すると 2 回目に行う方法は 1 度実装したことのある慣れた仕様となり，有利になる．そのため被験者によって方法の順序を変えている．被験者 3 人のうち，2 人は「提案」を先に行い，次は「普段」を行う．1 人は「普段」を先に行い，次に「提案」を行う．

仕様はファイル構造，HTML 構造，CSS のセレクタとプロパティ一覧，表示例を用意し，制作する直前に印刷したものを提示した．

ファイル構造はディレクトリを 2 つ用意し，「普段」の際に実装した Web サイトと，「提案」の際に実装した Web サイトでディレクトリを分けて保存するよう指示している．

HTML 構造を図 3 に示す．HTML はヘッダー，コンテンツ，フッターから構成され，コンテンツには見出しと段落

Vol.2016-GN-97 No.34
Vol.2016-CDS-15 No.34
Vol.2016-DCC-12 No.34
2016/1/22

を持ったセクションが複数ある．

```
html
└ head
  └ meta[charset=”UTF-8”]
  └ title
  └ link[rel=”stylesheet”, href=”css/main.css”]
└ body
  └ header
    └ h1{タイトル}
  └ div#contents.clearfix
    └ div.main
      └ section
        └ h2{見出し}
        └ div.folder
          └ p{段落}
      └ section
        └ h2{見出し}
        └ div.folder
          └ p{段落}
    └ div.sub1
      └ aside
        └ h2{見出し}
        └ div.folder
          └ p{段落}
    └ div.sub2
      └ aside
        └ h2{見出し}
        └ div.folder
          └ p{段落}
  └ footer
    └ div.copyright{(c) 2015 anyone}
```

図 3　HTML 構造

Figure 3　HTML structure.

CSS のセレクタとプロパティ一覧を図 5 に示す．また表示例を図 4 に示す．CSS は画面解像度の小さい閲覧環境向けに 1 カラムのレイアウトになるような CSS を指示している．また，CSS の後方にメディアクエリを指定し，その中で画面解像度が中程度以上の閲覧環境向けに 2 カラムのレイアウトになるような CSS を指示している．HTML 中の div.main は左側，div.sub1 と div.sub2 は右側に配置される．

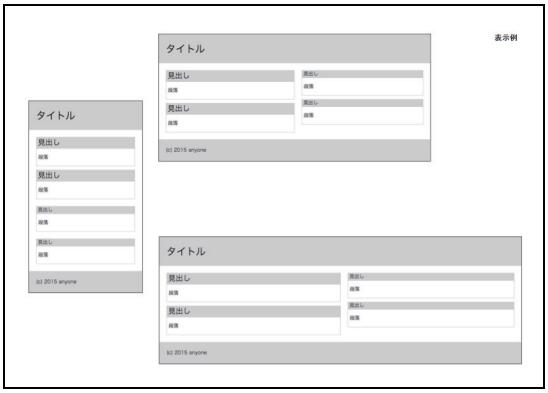

図 4　表示例

Figure 4　Display examples.

```
body, h1, h2
    - margin: 0px
    - padding: 0px
    - font-weight: normal
body
    - font-family: sans-serif
    - font-size: 87.5%
header, #contents, footer
    - padding: 20px
header, footer
    - background-color: #cccccc
section:not(:last-child), aside:not(:last-child)
    - margin-bottom: 10px
section h2, section .folder, aside h2, aside .folder
    - padding: 0px 5px
section h2, aside h2
    - background-color: #cccccc
section .folder, aside .folder
    - border: 1px solid #cccccc
aside
    - margin-top: 20px
aside h2
    - font-size: 100%

@media screen and (min-width: 600px)
.clearfix:after
    - content: ""
    - clear: both
    - display: block
.clearfix>*
    - float: left
.main
    - width: 50%
.sub1, .sub2
    - width: 50%
aside
    - margin-left: 20px
    - margin-top: 0px
.sub2 aside
    - margin-top: 10px
```

図 5　CSS セレクタとプロパティ一覧

Figure 5　Selectors and Properties List.

各方法で制作をした後，図 6 に示す仕様変更後の表示例を印刷したものを提示し修正を行う．仕様変更は CSS を変更し，画面解像度の大きい閲覧環境向けに 3 カラムのレイアウトになるような CSS を追加することを口頭で指示する．具体的には HTML 中の div.sub1 と div.sub2 を別々のカラムにする．他の閲覧環境ではレイアウトは変化しない．

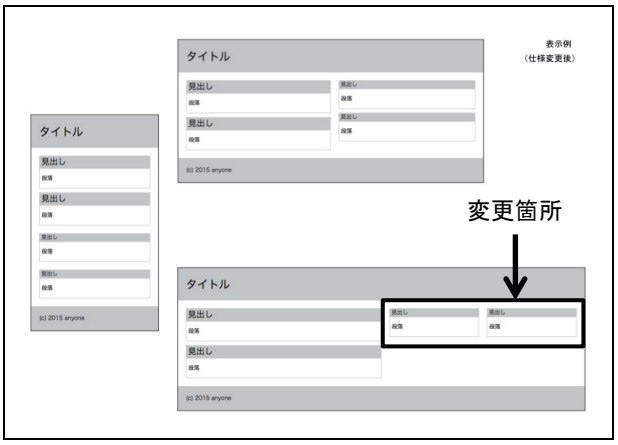

図 6　仕様変更後の表示例

Figure 6　Display examples after specification changed.

作業時間はファイルの作成から各閲覧環境で表示例に沿った表示ができていることを確認し終わるところまでを計

Vol.2016-GN-97 No.34
Vol.2016-CDS-15 No.34
Vol.2016-DCC-12 No.34
2016/1/22

測した．計測は各自のスマートフォンのストップフォッチアプリケーションを使用した．

すべての実装が終了した後，アンケート評価を行った．質問項目は 5 段階評価が次の 2 項目である．

- 以前の手法と比べて，
  表示の確認は使いやすくなりましたか
- 以前の手法と比べて，
  CSS の柔軟な変更は使いやすくなりましたか

自由記述の項目は次の 2 項目である．

- 表示の確認について，使いやすかった理由，
  使いにくかった理由は何ですか
- CSS の柔軟な変更について，使いやすかった理由，
  使いにくかった理由は何ですか

3.2 結果

以降，各被験者を A，B，C で表す．被験者 A，B は「提案」,「普段」の順で実装を行い，被験者 C は「普段」,「提案」の順で行っている．また,「普段」の表示確認の方法は，被験者 A はブラウザの開発ツールを用い，被験者 B，C はブラウザウィンドウのリサイズを用いた．普段から Brackets を使用しているのは被験者 A の 1 名である．

被験者 C はブラウザの開発ツールを使おうとして失敗したため,「提案（制作)」の時間が増えている．そのため後日，全員が再度実装を行った．

表 1 に 1 回目の被験者ごとの作業時間，表 2 に 2 回目の作業時間を示す．表中の数字は分，秒を表している．

表 1　1 回目の被験者の作業時間
Table 1　Working time of subjects for first time.

被験者	A	B	C
普段（制作）	10’ 43”	13’ 25”	15’ 46”
提案（制作）	08’ 37”	16’ 09”	08’ 31”
普段（修正）	02’ 11”	02’ 54”	07’ 45”
提案（修正）	01’ 46”	02’ 55”	01’ 15”

表 2　2 回目の被験者の作業時間
Table 2　Working time of subjects for second time.

被験者	A	B	C
普段（制作）	08’ 12”	11’ 50”	06’ 49”
提案（制作）	07’ 17”	08’ 15”	05’ 45”
普段（修正）	01’ 43”	01’ 05”	01’ 04”
提案（修正）	01’ 40”	01’ 02”	01’ 12”

表 3 に作業時間の平均と差を示す．差は「制作」と「修正」それぞれについて「普段」から「提案」を引いたものである．すなわち提案アプリケーションの使用により削減された時間である．

表 3　被験者の作業時間平均と差
Table 3　Working time Average and Difference of subjects.

	1 回目（C を除く）		2 回目	
	平均	差	平均	差
普段（制作）	12’ 04”	-00’ 19”	08’ 57”	01’ 51”
提案（制作）	12’ 23”		07’ 06”	
普段（修正）	02’ 23”	-00’ 12”	01’ 17”	-00’ 01”
提案（修正）	02’ 20”		01’ 18”	

表 4 はアンケート評価のうち，5 段階で尋ねた項目の回答である．表中の数字は評価値を表している．

表 4　アンケート評価の回答
Table 4　Answer of evaluation questionnaire.

被験者	A	B	C	平均
表示確認	5	4	4	4.33
CSS の編集	5	4	3	4.00

表示確認が使いやすい理由を尋ねた項目には次の回答があった．

A)　解像度を何度も指定せずとも 2 つの確認を瞬時にできるため
B)　タブで気軽に設定したサイズに切り替えられるから
C)　手動でサイズを変えるよりは楽な気がする

CSS の編集が使いやすい理由を尋ねた項目には次の回答があった．

A)　確認が瞬時にできるため、すぐに CSS が正しいかわかる
B)　適用すべきスタイルとそうでないスタイルを複数の画面で見比べながら作るのが楽だったから
C)　今回の実験で、いろいろなウィンドウサイズを見ながら CSS を調整する場面が少なく，あまり活用できなかったように感じた

3.3 考察

評価実験の結果から，提案アプリケーションの有効性を確認することができた．

作業時間において 2 回目の差は,「制作」については約 2 分短縮している．「修正」については「普段」が有利な被験者が 3 名中 2 名いる中で，1 秒の増加にとどまっている．また，被験者 C を除いた 1 回目の差は「普段」が有利な被験者が 2 名中 2 名である中で,「制作」,「実装」とも 20 秒以内の増加にとどまっている．提案アプリケーションを用いた実装方法により，表示確認と CSS の編集にかかる作業時間を短縮することができたと言える．

1 回目と 2 回目の平均を比較すると 4 通りとも,2 回目は 1 回目よりも短縮されている．特に「普段（制作)」は 3 分

程度の短縮であるのに対し「提案（制作）」は 5 分以上短縮されている．1 回目と 2 回目では各方法に対する習熟度が変化しており，表示確認について提案アプリケーションは習熟がしやすいと言える．このため作業時間の短縮につながったと考えられる．

アンケート評価においては，表示確認については全員が使いやすい理由として手間を削減できたことを回答している．CSS の編集については 2 名が使いやすい理由として仕様変更に対応しやすいことを回答している．被験者 C は複数の表示確認をしながら CSS の編集をする場面が少ないと回答している．

「制作」の際には 2 通りのレイアウトを同時に実装しており，それぞれ交互に表示確認する必要があるため，複数の表示確認は有効であった．しかし「修正」の際には 1 通りのレイアウトを実装しており，表示確認は修正対象の閲覧環境のみで，他の閲覧環境の表示確認は修正によって影響を受けていないか確認するにとどまる．そのため，複数の表示確認をしながら CSS の編集をする場面が少なくなっており，2 通り以上レイアウトの修正を行う際には有効であると考えられる．

以上のことから本アプリケーションを用いることで表示確認の時間の削減と，仕様変更に柔軟に対応できる CSS の編集が実現できたと考える．

4. おわりに

レスポンシブ Web デザインの持つ，閲覧環境ごとに表示確認を行う手間があることや，デザイン仕様の変更が多いといった課題を解決するために，本研究ではブラウザ上で閲覧環境ごとに表示確認とデザインの変更を実現するアプリケーションを提案した．評価実験では普段からレスポンシブ Web デザインによる Web 制作を行っている人に使ってもらい，課題が解決されていることを確認できた．

今後の課題として Brackets 以外でも使用できるようにするため，他のエディタ向けの拡張機能作成を考えている．

参考文献

[1] 総務省：情報通信白書平成 27 年度版 (2015)
[2] 菊池崇：レスポンシブ Web デザイン マルチデバイス時代のコンセプトとテクニック, アスキー・メディアワークス (2013)
[3] 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)
[4] Adobe：デバイスプレビューによるレスポンシブなデザイン | Adobe Dreamweaver CC tutorials，入手先〈https://helpx.adobe.com/jp/dreamweaver/how-to/device-preview-responsive-design.html〉（参照 2015-12-08）
[5] Matt, K.: Responsive Design Testing, available from 〈http://mattkersley.com/responsive/〉（accessed 2015-12-08）
[6] Tama, P. and Andy, H.: Responsinator, available from 〈http://www.responsinator.com/〉（accessed 2015-12-08）
[7] Adobe: Brackets，〈http://brackets.io/〉（accessed 2015-12-17）
[8] Adobe: LiveDevMultiBrowser - Brackets API, available from 〈http://brackets.io/docs/current/modules/LiveDevelopment/LiveDevMultiBrowser.html〉（accessed 2015-12-07）