M-024

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
An Application that support Web production Supporting various browsing environments

遠藤 崇† 速水治夫†
Takashi Endo Haruo Hayami

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．このことから Web サイトの閲覧環境は多様化しており，Web 制作をする 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．

多様な閲覧環境に対応する Web 制作の手法として 2 つの手法がある．振り分け型とレシポンシブ Web デザインの 2 つである．

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である．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である．

振り分け型は端末の名称や OS の名称などを基準に，対象の HTML ファイルに移動させる手法である．ブラウザが持つユーザエージェント情報には端末の名称や OS の名称などが含まれており，JavaScript やサーバサイド技術を用いてこれらを取得する．

レスポンシブ Web デザインは画面解像度などを基準に対象とする CSS を切り替えてレイアウトを変化させる手法である．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．

CSS の記述はベースとして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].

レスポンシブ Web デザインは振り分け型と比較した利点を以下に示す[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

† 神奈川工科大学大学院情報工学専攻 Graduate School of Information Technology, Kanagawa Institute of Technology

しかし，レスポンシブ Web デザインには画面解像度の異なる閲覧環境ごとに表示確認をする手間があること，デザイン仕様の変更が多いといった課題があり，本論文ではその解決策を提案する．

2. 課題と解決策

2.1 課題

レスポンシブ Web デザインが持つ課題として，1 つの HTML ファイルで複数の閲覧環境に対応するため，画面解像度の異なる閲覧環境ごとに表示確認をする手間があることや，デザイン仕様の変更が多いことが挙げられる．

表示確認については振り分け型の場合，各閲覧環境に対応した HTML ファイル・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，ベースとなる閲覧環境向けの CSS を表示確認しながら，差分となる閲覧環境向けの CSS を追記する．ある閲覧環境向けの CSS を編集した際に，別の閲覧環境で表示が崩れることがあるため，他の閲覧環境でも表示確認をする．

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．そのため，仕様の変更は起こらない．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，デザイン仕様を変更して実装することがある．

2.2 先行事例

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[4]・Responsive Design Testing[5]・Responsinator[6]がある．

Dreamweaver CC のデバイスプレビュー機能はテキストエディタである Dreamweaver CC と複数の閲覧環境を LAN（Local Area Network，構内ネットワーク）を用いて接続し，テキスト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，コストが高いという課題がある．

Responsive Design Testing，Responsinator 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を用い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．そのため，デザイン仕様の変更には対応できていない．

2.3 解決策

本研究では，Web 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 制作をする人を想定ユーザとし，プロトタイプの Web サイトについて表示確認やデザインの変更をすることを目的とする．

機能として，レスポンシブ Web デザインにより制作された HTML ファイルと CSS ファイルを複数の画面解像度で表示する機能・CSS を編集する機能の 2 つを提供する．

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．インラインフレームは 1 つのページ上に別のページを埋め込んで表示する．同じ HTML ファイルを表示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 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．

3. 提案アプリケーション

3.1 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．

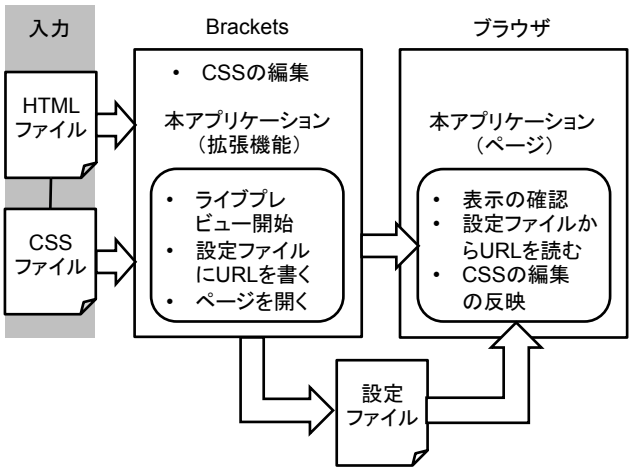

図 1　アプリケーションの構成とフロー

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．デフォルトの動作は単一プレビューのため，複数プレビューが可能な新しい動作に変更する．変更には Brackets が拡張機能向けに提供している Live Preview API[8]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る．このボタンが押されると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．新しい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ライブプレビューの URL を書き込み，ページをブラウザで開く動作にする．

ページは表示確認のために用いられ，ブラウザで開いて使用する．ページには複数のインラインフレームがあり，インラインフレームで表示する対象に設定ファイルから読み込んだ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．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用いるためである．CSS を編集する機能は Brackets のテキストエディタ機能を用い，編集内容はページ上に反映する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表示する表示確認画面と，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．

3.2 ユーザインタフェース

図 2 に表示確認画面を示す．表示確認画面にはインラインフレームが 2 つあり，各インラインフレームのサイズはフレーム上部のタブを用いて切り替える．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．

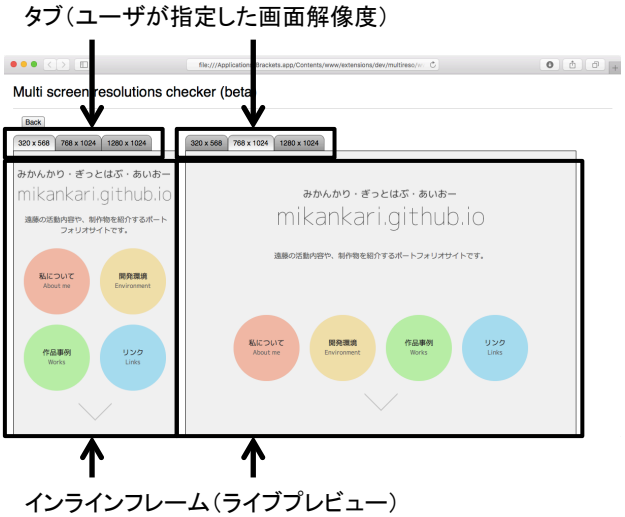

図 2　表示確認画面

レスポンシブ Web デザインでは，ベースとなる閲覧環境向けの CSS を表示確認しながら，差分となる閲覧環境向けの CSS を追記する．このため，ベースとなる閲覧環境と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フェースとした．

3.3 使い方

使用するにあたって，あらかじめ制作した HTML ファイル・CSS ファイルが必要である．

まずユーザは Brackets 上で HTML ファイル・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，図 3 に示す設定画面が表示される．また，表示確認の対象とする HTML ファイルに Brackets で開いている HTML ファイルを自動で指定する．

CSS ファイルは HTML ファイルに CSS ファイルへリンクする記述があるため指定しない．

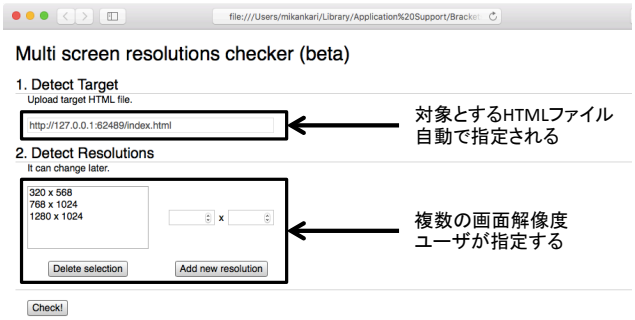

図 3　設定画面

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，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対象の HTML ファイル・CSS ファイルの表示確認をすることができるようになる．

表示確認をして，表示が正常でない場合には CSS の編集をする．Brackets で HTML ファイルのタグ・CSS ファイルのセレクタや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．

4. 評価実験

4.1 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する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した．

評価実験は Web サイト実装の作業時間比較とアンケート評価をした．

4.1.1 Web サイト実装の作業時間比較

Web サイト実装の作業時間比較はあらかじめ用意した仕様に沿って実装する「制作」をし，次に各方法について仕様変更を提示して実装する「修正」をする．またそれぞれの実装において，通常使用している表示確認の方法を用いる「通常」と，提案アプリケーションを用いる「提案」の 2 回の方法で実装した．いずれの方法も同じ Web サイトを実装し，実装には Brackets を用いた．4 通りの作業時間を計測し，最後にアンケート評価をした．

それぞれ同じ Web サイトを実装すると，後の方法は 1 度実装したことのある慣れた仕様となるため有利になる．そのため被験者によって方法の順序を変えている．図 4 は順序を表している．被験者 3 人のうち，2 人は「提案」を先に実装し，次は「通常」を実装する．1 人は「通常」を先に実装し，次に「提案」を実装する．

図 4　各グループの順序

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，「通常」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．

実装に用いた HTML 構造を図 5 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 5　HTML 構造

CSS のセレクタとプロパティ一覧を図 6 に示す．また表示例を図 7 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．HTML 中の各セクションである div.main，div.sub1，div.sub2 はすべて 1 列に配置される．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．各セクションのうち，div.main は左側，div.sub1 と div.sub2 は右側に配置される．

```
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 6　CSS セレクタとプロパティ一覧

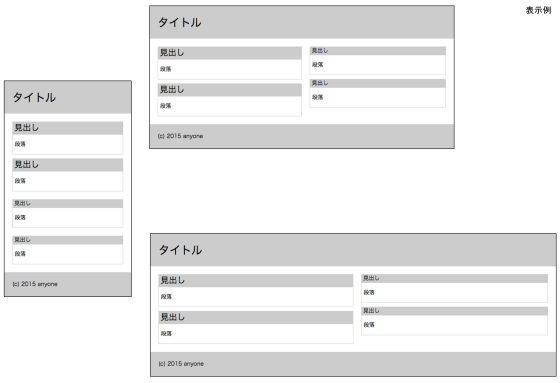

図 7　表示例

各方法で「制作」をした後，図 8 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」を実装する．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 を別々の列に配置する．他の閲覧環境ではレイアウトは変化しない．

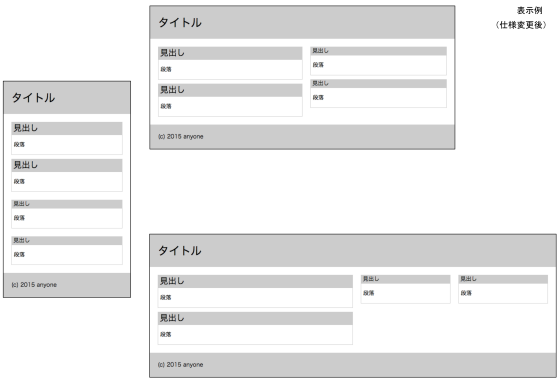

図 8　仕様変更後の表示例

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各自が計測した．

4.1.2 アンケート評価

すべての実装が終了した後，アンケート評価をした．質問項目は以下の 4 項目である．設問 1 と設問 3 は 5 段階評価であり，設問 2 と設問 4 は自由記述である．

設問1. 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
設問2. 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
設問3. 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
設問4. 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

4.2 結果

4.2.1 Web サイト実装の作業時間比較

被験者 A，B は「提案」，「通常」の順で実装し，被験者 C は「通常」，「提案」の順で行っている．また，「通常」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 を用いているのは被験者 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，「制作」を「提案」の方法で実装した時間が増えている．そのため後日，全員が再度実装した．以後，失敗に要した時間を含む実装を「当初」，再度した実装を「再度」として表す．

表 1 に「当初」の被験者ごとの作業時間，表 2 に「再度」の作業時間を示す．

表 1　「当初」の各被験者の作業時間（単位：分’秒”）

グループ	「通常」有利				「提案」有利	
被験者	A		B		C	
方法	通常	提案	通常	提案	通常	提案
制作	10’ 43”	08’ 37”	13’ 25”	16’ 09”	15’ 46”	08’ 31”
修正	02’ 11”	01’ 46”	02’ 54”	02’ 55”	07’ 45”	01’ 15”

表 2　「再度」の各被験者の作業時間（単位：分’秒”）

グループ	グループ 1（「通常」有利）				グループ 2（「提案」有利）	
被験者	A		B		C	
方法	通常	提案	通常	提案	通常	提案
制作	08' 12"	07' 17"	11' 50"	08' 15"	06' 49"	05' 45"
修正	01' 43"	01' 40"	01' 05"	01' 02"	01' 04"	01' 12"

表 3 に「当初」の，表 4 に「再度」の作業時間の平均と短縮時間を示す．短縮時間は「制作」と「修正」それぞれについて「通常」から「提案」を引いた差である．短縮割合は「通常」に占める短縮時間の割合である．どちらも提案アプリケーションを用いることで削減された時間と割合を表している．

表 3　「当初」の作業時間の平均と短縮時間，短縮割合（単位：分’秒”）

	平均		短縮時間	短縮割合
	通常	提案		
制作	12' 04"	12' 23"	-00' 19"	-2.62%
修正	02' 32"	02' 20"	-00' 12"	7.87%

表 4　「再度」の作業時間の平均と短縮時間，短縮割合（単位：分’秒”）

	平均		短縮時間	短縮割合
	通常	提案		
制作	08' 57"	07' 06"	01' 51"	20.73%
修正	01' 17"	01' 18"	-00' 01"	-0.86%

表 5 に「当初」と「再度」の習熟による時間差を示す．習熟による時間差は「制作」と「修正」，「通常」と「提案」の 4 通りの平均について「当初」から「再度」を引いた差である．「再度」は過去に実装したことのある慣れた仕様を実装することから，「当初」と「再度」では習熟度が変化している．習熟による時間差はこの習熟度を示している．

表 5　「当初」と「再度」の習熟による時間差（単位：分’秒”）

	習熟による時間差	
	通常	提案
制作	03' 07"	05' 17"
修正	01' 15"	01' 02"

4.2.2 アンケート評価

表 6 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 6　アンケート評価の回答

被験者	A	B	C	平均
設問 1	5	4	4	4.33
設問 3	5	4	3	4.00

設問 2 には以下の回答があった．

被験者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
被験者B. タブで気軽に設定したサイズに切り替えられるから
被験者C. 手動でサイズを変えるよりは楽な気がする

設問 4 には以下の回答があった．

被験者A. 確認が瞬時にできるため，すぐに CSS が正しいかわかる
被験者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
被験者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

4.3 考察

以下のことから，提案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対応しやすい CSS の編集が実現できたと考える．

4.3.1 Web サイト実装の作業時間比較

作業時間において「再度」の短縮時間は，「制作」については 1 分 51 秒短縮している．「修正」については「通常」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「当初」の短縮時間は「通常」が有利な被験者が 2 名中 2 名である中で，「制作」，「修正」とも 20 秒以内の増加にとどまっている．「通常」が有利である人数の方が多いことを考慮すると，「提案」が有利である人数と等しい場合には短縮時間はさらに延びることが考えられる．提案アプリケーションを用いた実装方法により，「制作」に要する表示確認と

「修正」に要する CSS の編集の作業時間を短縮することができたと言える．

「当初」と「再度」の習熟による時間差を比較すると 4 通りとも，「再度」は「当初」よりも短縮されている．特に「制作」・「通常」は 3 分 7 秒の短縮であるのに対して「制作」・「提案」は 5 分 17 秒短縮されている．「当初」と「再度」では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．

4.3.2 アンケート評価

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いる．2 通り以上のレイアウトの修正をする際には有効であると考えられる．

5. おわりに

レスポンシブ Web デザインでは，閲覧環境ごとに表示確認をする手間があることや，デザイン仕様の変更が多いといった課題がある．その問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，他のテキスト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] 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” (2013).
[3] 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” (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”, https://helpx.adobe.com/jp/dreamweaver/how-to/device-preview-responsive-design.html
[5] Matt K, “Responsive Design Testing”, http://mattkersley.com/responsive/
[6] Tama P and Andy H, “Responsinator”, http://www.responsinator.com/
[7] Adobe, “Brackets”, http://brackets.io/
[8] Adobe, “LiveDevMultiBrowser - Brackets API”, http://brackets.io/docs/current/modules/LiveDevelopment/LiveDevMultiBrowser.html