レスポンシブ web デザインのレイアウト設計を支援する領域特化言語の提案

2010SE254 鵜飼 菜生
指導教員: 横山 哲郎

1 はじめに

スマートフォンやタブレットなど携帯情報端末の急速な普及により, 私たちは小さな画面で web サイトを閲覧する機会が増えた. そこで, 大きな画面にも小さな画面にも対応する web デザインの重要性が高まっている.

画面が小さいということは, 表示できる情報の量だけでなく最適なレイアウトも変化するため, 画面の大きさに合わせたレイアウトを適宜用意する必要がある. その方法としてサイト閲覧に使用されている端末を検知し, CSS (Cascading Style Sheets)で HTML ファイルを装飾するレスポンシブ web デザインがある.

HTML ファイルに記述された要素の表示形式を定める CSS では変数や関数の利用ができないため, 他の一般的なプログラミング言語に比べて制約が多い. レスポンシブ web デザインにおける画面に合わせたレイアウトの変更はメディアクエリーというモジュールを用いて, 1 つの要素に対しての処理を複数の記述から条件によって選択する形で実現させている.

現在レスポンシブ web デザインをより手早く構築する方法は複数存在するが, 最終的な CSS ファイルの記述に関しては条件分岐を用いた煩雑なものになっていることが多い. また, 1つの HTML 要素に着目した場合, レイアウトパターンが変更されるたびに, 要素の大きさや余白について記述しなければならない. レイアウトの切り替えに関する部分を分離して宣言的に記述することができれば CSS ファイル全体の可読性, 保守性の向上に繋がると考えられる.

一方で我々がソフトウェアを開発するために, プログラミングを行う場合, 自動的にエディタがインデントを整え, 閉じ括弧の位置を調整する, 改行を追加するなどの処理を加えていることがある. これにはプリティプリンタという理論が用いられている. テキストを対象として, 最適なレイアウトを自動生成する技術である.

本論文では, 画面の大きさに合わせたレイアウト記述の部分を改善するためにプリティプリンタの理論を応用した web 開発用の新言語を提案する.

図1のように複数のレイアウトパターンを持つ web サイトの CSS ファイルを例として開発する. この例は, 実際の web サイトのレイアウトとして有用なものになるように, 画面の右側にメニューを配置した 2 カラムの web サイトを想定している.

レスポンシブ web デザインのサイト開発ではコンピュータ, タブレット型端末, スマートフォンの 3 種類の表示デバイスを想定して, レイアウトのパターンも 3 種類用意することが多い. 今回のレイアウト例でもそれにならってパターンを 3 つ用意した.

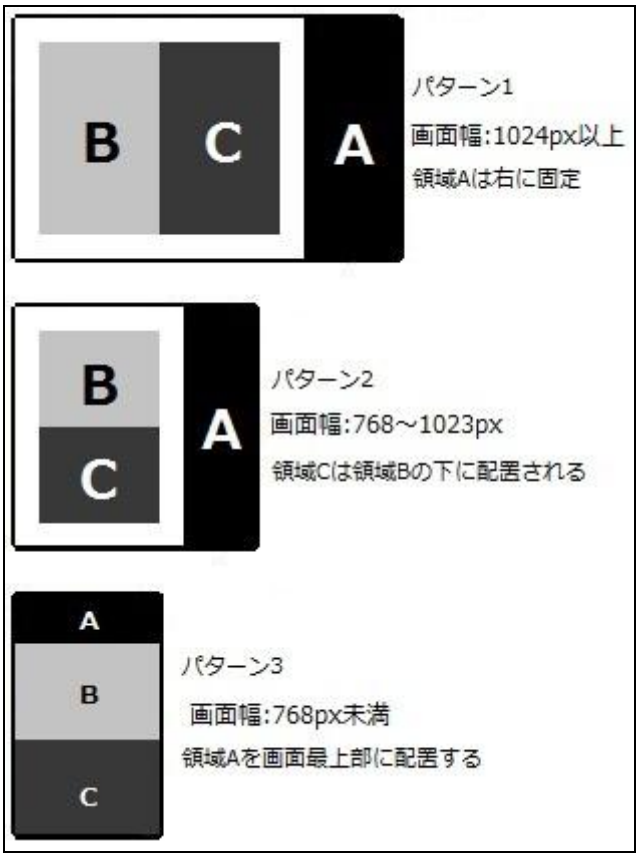

図 1 本研究にて取り扱うレイアウト例

2 関連研究

レスポンシブ web デザインにて多く用いられる技術のひとつにメディアクエリーがある. W3C によって提唱された CSS3 に導入されているモジュールである. メディアクエリーを利用することで, 1 つの CSS ファイルで複数の出力デバイスごとにスタイルの調整が可能になった. 例えば, 家庭用テレビから web サイトが閲覧される場合とタブレット端末から閲覧される場合を想定し, それぞれに最適なレイアウトを適用することができる[1].

現在レスポンシブ web サイトの開発にはデザインフレームワークが用いられることがある. これは, web サイト作成の際に必要な CSS や JavaScript があらかじめ作成された状態で配布されるテンプレートを指す. Twitter Bootstrap など, 様々な意匠のデザインフレームワークが開発されている[2].

デザインフレームワークとして配布される CSS ファイルの作成は, CSS 拡張メタ言語を用いて記述, コンパイルして行われることがある. Twitter Bootstrap では CSS 拡張メタ言語のひとつである LESS という言語が使用されている.

CSS 拡張メタ言語とは CSS の文法はほぼそのままに，繰り返しや条件分岐による制御を可能にした言語のことである．拡張メタ言語のファイルをコンパイルして生成された CSS ファイルが実際のブラウザ表示に用いられる[2].

プリティプリンタとは，テキストを対象に，レイアウトを自動生成する技術である．プログラムのソースコードを編集する場合，プリティプリンタライブラリを導入していればコードの文脈に応じて自動でインデントを下げる・空白行を挿入するなど，読みやすいレイアウトに変換して出力する．プログラミング言語の領域にてプリティプリンタは広く用いられ，ライブラリとしても浸透しており，その理論は成熟していると言える[3].

3 アプローチ

プリティプリンタによるソースコードの整形と web デザインの間に「画面上でのレイアウト」という点での類似性から，整形される対象をテキストから HTML 要素に置き換えることで，web デザインにプリティプリンタの理論を応用する．本研究ではこれを実現するように CSS を拡張することを目標とする．

具体的な web サイト開発としての目標は図 1 のようなレイアウトパターンを持ち，それを画面の大きさに合わせて切り替えるレスポンシブ web デザインの web サイトを開発する．

CSS にプリティプリンタの理論を導入する方法としては埋込み型領域特化言語を用いる．拡張メタ言語ではあらかじめ用意された言語の機構しか用いることができないため，言語開発者の意図した機能しか利用することができない場合が多い．しかし，埋込み型領域特化言語ではホスト言語のコンビネータとして実装されるため，ホスト言語の機能を利用できるという利点がある．

また，開発例のレイアウトの動きとほぼ同じ挙動をする web サイトを既存の手法で実装することで，開発者が自ら要素の配置や余白を均一にするために CSS の規則を踏まえて各数値を設定する必要があることを明らかにした．

4 領域特化言語の仕様

Web サイトを構成する HTML タグは抽象データ型として実装されているものとすると，プリティプリンタの導入に必要な物は，「HTML タグでは処理できないレイアウトの情報を格納するデータ型と関数」，「要素の配置を決定する演算子」，「最適なレイアウトを適用する関数」の 3 つである．

抽象データ型 `Layout` を用意し、これに隣接する要素との相関関係などの HTML 及び CSS では処理できないレイアウトの情報を格納する．

要素の配置を決定する演算子として 2 つの演算子を用意する．

```
</> :: Layout → Layout → Layout
```

この演算子は要素を順に横に並べたレイアウトと，右側の要素を下にして縦に並べたレイアウトを値として返す．

```
<¥> :: Layout → Layout → Layout
```

この演算子は右側の要素を左側に入れ替えて横に並べたレイアウトと，右側の要素を下にして縦に並べたレイアウトを値として返す．

ここでは，領域 A, B, C はそれぞれ `AAA`，`BBB`，`CCC` という名前の `Layout` 型データとして取り扱うことにする．今回の開発例について上記 2 つの演算子を用いて，以下のように記述する．

```
AAA <¥> (BBB </> CCC)
```

レイアウトの集合から最も整ったものを選ぶ関数として `pretty` を用いる．

```
pretty :: Int → Layout → String
```

この関数 `pretty` は整数型の値を 1 つ取得し，`Layout` 型データに表示領域の横幅として反映させる．

レスポンシブ web デザインの実現のためにはこの関数で表示領域の幅を取得し，取得した値に応じたレイアウトパターンを適用する必要がある．表示領域に関する情報の取得はメディアクエリーを用いた場合でも行われているため，十分に可能であると言える．

関数 `pretty` の処理中に照合する最適なレイアウトを判断する要素として，属性 `min_width` を導入する．これは `Layout` 型の値に「最小の幅」の情報を追加する．

以上の要素を新しい領域特化言語として導入することで，各要素の配置を最適化し，結果として図 1 のようにレイアウトが決定する．

今回提案する新しい領域特化言語で開発した場合の大きな利点は1つの領域に対してレイアウトに関する記述をする回数が減少するという点である．メディアクエリーを用いた場合，表示領域の幅が 1024px 以上のとき，768px ～1023px のとき，768px 未満のときと 3 回領域 A, B, C についてそれぞれ余白，幅，表示位置を指定しなければならない．しかし，新しい領域特化言語では要素の配置に関する記述を分離し宣言的に記述することができる．また，文章の総量も削減することができる．

5 おわりに

既存の方法ではレイアウトパターンの数だけ，HTML 要素の配置や大きさを指定するスタイルシートを用意しなければならないが，プリティプリンタを応用することでレイアウトパターンを分離して記述することができる．要素の配置については自動生成させることでレイアウトに関する記述の可読性・保守性を高めることができる．

今回開発の例として作成したレイアウト以外にも，この領域特化言語を用いることで同様の効果が期待できる．

今後の課題としては，ここまでの仕様や理論を基に既存の HTML タグや CSS の属性を新言語向けに再定義し，新言語を実用に向けて開発を進めることが挙げられる．

参考文献

[1] Rivoal, F: Media Queries, W3C Recommendation, CSS Current Status (online), available from 〈http://www.w3.org/TR/2012/REC-css3-mediaqueries-20120619/〉 (accessed 2013-07-07).

[2] Lerner, R.M.: At the forge: Design Frameworks, Linux J., Vol.2012, No.217, pp.26 - 32 (2012).

[3] John, H.: The Design of a Pretty-printing Library, *Proc. Advanced Functional Programming*, Lecture Notes in Computer Science, Vol.925, pp.53 - 96 (1995).