2016 年 02 月 10 日

平成 27 年度　情報・通信工学専攻　コンピュータサイエンスコース　修士論文発表会

Web Components 開発におけるドキュメント同時生成手法

寺田研究室 学籍番号:1431017 海老澤 雄太 指導教員 寺田 実，岩崎 英哉

1　背景と目的

Web Components は，Web ページ中の構成要素を再利用するための技術である．独自タグの利用やカプセル化など，開発者とユーザの両方にとって有用な機能を提供する．現在，Google Chrome や webcomponents.js などの Polyfill ライブラリを使用することで，Web Components の提供する機能を体験することができる．

コンポーネント作成において，使用方法や API を記したドキュメントの用意は重要である．ソースコード中の特殊なコメント（ドキュメントコメント）を使用するなど，ソースコードとドキュメントを同時生成する手法があるが，Web Components への適用は困難である．そのため，ドキュメントを個別に作成するか，ソースコード中のコメントをドキュメントとして扱うことが一般的である．

本研究では，ドキュメント作成にかかるコストを削減し開発効率を向上させることを目的に，従来手法に代わるソースコードとドキュメントの同時生成手法を提案する．そして，提案手法によってソースコードとドキュメントを生成するためのツールを作成し，提案手法の特徴や利点欠点を議論していく．

2　提案手法

2.1　設計

提案手法では，文芸的プログラミング [2] のように，ドキュメント記述中にコード片を埋め込むことで両方の記述を同時に行う．このとき，ドキュメント記述に一定の制限や形式化を施すことで，記述や出力ドキュメントのフォーマットの統一を達成する．

入力文書をシステムが解析して文書に埋め込まれたコード片を抽出し，ドキュメントの内容から生成可能なスケルトンコードにそれらを適用していくことで，完成したソースコードを生成する．また，埋め込まれたコード片などの

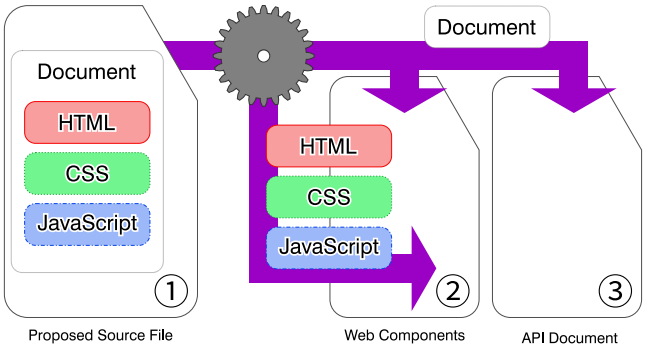

図 1　提案手法の概念図

```
1  # レベル1の見出し
2  ## レベル2の見出し
3
4  空行から空行までの間が1つの段落として表現される．
5  テキストについては *斜体*
6  もしくは **強調** の装飾が可能．
7
8  - リスト構造も
9      - 入れ子で
10         - 記述が可能
11
12 ```javascript
13 // プログラムコードの埋め込み例
14 console.log("Hello, world.");
15 ```
```

図 2　Markdown 記法の例

ドキュメントに不必要な情報を除去し，記述内容を整形することで統一フォーマットのドキュメントを生成する．

2.2　ドキュメント記法

TEX をはじめ多くのドキュメント記法が存在するが，提案手法で用いるドキュメント記法として，Gruber が提唱する Markdown 記法を採用した．Markdown 記法は，図 2 に示すようなテキストメールの装飾に似た記述で，文書構造を表現する．その記述の簡易さや見やすさ，およびコード片の埋め込みが可能なため，この記法を採用した．

3　記法

提案手法における Web Components の記述例を図 3 に示す．これは図 4 に示す簡易カウンターのソースコードとそのドキュメント（図 5）を定義するための記述例である．

記述は作成する Web Components の名前（独自タグ名として使用）のレベル 1 の見出し（16 行目）ではじめ，特定のタイトルをつけたレベル 2 の見出しから見出しまでの範囲（以下，セクションと呼称）に様々なコンポーネントの定義を記述していく．例えば，マークアップ構造であれば structure セクション（17 行目から 23 行目）に，メソッド定義ならば methods セクション（31 行目から 39 行目）に記述していく．その他，「プロパティ定義」「デモコード」「テストコード」などの記述が可能である．

また，制作者情報や依存関係などのメタ情報は，YAML Front Matter という Markdown 文書で広く使われる手法で記述する．このメタ情報には，ソースコード生成後に外部ツールを用いてテストコードを実行するなど，処理系への動作指定も含まれる．

4　実装

提案手法による記述を行った文書から，Web Components のソースコードとドキュメントを生成するシステムを，Node.js で動作するツールとして実装した．ソースコードを GitHub[*1] にて公開している．

[*1] https://github.com/e8l/wcWEB

```
# mini-counter
## structure
マークアップ構造に関する説明
```html
<div id="container">
  <span id="num">0</span>
</div>
```
properties
count
:   type Number
:   `0`

カウンターの値

methods
reset
:   no arguments

カウンタの値を0に戻す
```javascript
this.count = 0;
this.$.num.textContent = "0";
```
lifecycle
attached
```javascript
this.$.num.addEventListener("click",
  (function(e){
    ++this.count;
    this.$.num.textContent = this.count;
  }).bind(this));
```
```

図 3　提案手法による Web Components の記述

```
<link rel="import" href="../polymer/polymer.html">
<dom-module name="mini-counter">
  <template>
    <div id="container">
      <span id="num">0</span>
    </div>
  </template>
  <script>
    (function() {
      Polymer({
        is: "mini-counter",
        properties: {
          count: {type: Number, value: 0}
        },
        attached: function() {
          this.$.num.addEventListener("click",
            (function(e){
              ++this.count;
              this.$.num.textContent = this.count;
            }).bind(this));
        },
        reset: function() {
          this.count = 0;
          this.$.num.textContent = "0";
        }
      });
    })();
  </script>
</dom-module>
```

図 4　図 3 の記述より生成される Web Components

システムでは，Markdown 文書の抽象構文木を作成し，ソースコード生成のための情報抽出とドキュメント生成のための整形を行う．抽象構文木の作成とドキュメント生成には，Pandoc を用いた．ソースコード生成は，Mustache を用いて，雛形にデータを適用することで生成する．

また，システムは入力文書のメタ情報に応じて，ソースコードの生成後に外部ツールの実行を行う．それにより，ソースコードの最小化やテストコードの実行などを行う．

5　議論

提案手法により，Web Components においても，ソースコードとドキュメントの同時生成を可能にしている．また，ドキュメントコメントを用いる手法と比較するとドキュメントの内容からソースコードの一部を生成するため，記述の重複を削減できている．

しかし，現時点では生成したソースコードに誤りがあった場合，その箇所と提案手法で記述した文書中の位置との

図 5　図 3 の記述より生成されるドキュメント

対応付けが困難である．また，GitHub にて公開されている Web Components7 種を提案手法で記述したところ，ドキュメント化の必要のない API の定義が提案手法では煩雑となる場合があった．

6　関連研究

Web Components とは異なる Web ページの構成要素のコンポーネント化として，Benson らの Cascading Tree Sheet[1] や，山本らの Cop[4] などがある．

Markdown 記法を扱ったものとしては，数式やプログラムコードを含む複雑な文書を作成するための Leijen の Madoko[3] や，WebAPI 定義のための API Blueprint[*2]が挙げられる．

7　まとめと今後の課題

本研究では，Web Components を対象に，ソースコードとドキュメントの同時生成手法を提案した．

提案手法では，記述の重複が削減できるが，内部処理の記述やデバッグについて改善点が見られた．

今後は，システムを含めたこれらの課題の解決とともに，実験を通じて定性的・定量的な評価を行っていきたい．

参考文献

[1] Edward O. Benson and David R. Karger. Cascading Tree Sheets and Recombinant HTML: Better Encapsulation and Retargeting of Web Content. WWW '13, pp. 107–118.

[2] Donald E. Knuth. Literate Programming. *Comput. J.*, Vol. 27, No. 2, pp. 97–111.

[3] Daan Leijen. Madoko: Scholarly Documents for the Web. DocEng '15, pp. 129–132. ACM, 2015.

[4] 山本竜太郎, 岩崎英哉. Cop: 部品化を指向した Web アプリケーション開発の枠組. 第 57 回プログラミングシンポジウム予稿集, pp. 119–129.

対外発表

本研究は，第 57 回プログラミング・シンポジウムにて，登壇発表およびポスター・デモ発表を行った．また，夏のプログラミング・シンポジウム 2014 では，卒業研究に関する内容で登壇発表を行った．

[*2] https://apiblueprint.org/