クライアントサイドにおける Web ページ内のプログラム記述の閲覧機能拡張に関する研究
– プログラム記述への理解支援機能の挿入方法 –

2008MI147 森島 敦子　　2008MI223 椎名 優貴　　2008MI251 土屋 陽平
指導教員　吉田 敦

1　はじめに

近年，プログラミングの学習やソフトウェア開発の参考にするために，プログラム記述を含む Web ページを閲覧する機会が増加している．閲覧者は目的に合うプログラム記述を探す際，多くのリポジトリサイトや解説サイトを閲覧する．

Web ページ内のプログラム記述は，必ずしも理解しやすいものではない．具体的には，記述者のコーディングスタイルによって，閲覧者が見栄えが悪いと感じる場合がある．また，閲覧者が必要としない関数が多く記述されている場合は閲覧の妨げになる．理解しやすい見栄えに変更する簡単な方法は，サーバサイドで変更することであるが，サーバサイドですべての閲覧者の要求を満たす見栄えにはできない．また，閲覧者はサーバサイドの見栄えを変更できない．別の手段として．閲覧者が既存の理解支援可能なツールや環境 [1][2][5] を用いる方法がある．しかし，理解支援を行うまでに，既存のツールをインストールする手間や 1 つ 1 つ Web ページ内からプログラム記述をダウンロードする手間がかかる．また，プログラム記述をダウンロードすることで，HTML のタグなどで付加された情報が失われる．

以上のように閲覧機能の拡張を行う際にはサーバサイドではなく，クライアントサイドで直接 Web ページ上に閲覧機能を挿入する方法が必要である．また，閲覧者の要求は多彩であり，それらをすべて満足するような機能は提供できない．よって，汎用的な機能を提供すること，閲覧者自身が機能を定義することも必要である．

本研究では，クライアントサイドにおける閲覧機能拡張を目的に，Web ページ内への理解支援機能の挿入方法を提案する．これにより，Web ページ内のプログラム記述を閲覧する際に，閲覧者が必要に応じて閲覧機能を利用できる．閲覧機能を作成し，利用する場合は，プログラムについての基本知識を持つユーザを対象とする．

Web 上での閲覧機能の実現には 2 つの問題がある．1 つ目は，クライアントサイドで閲覧機能を挿入する際に，プログラム記述の解析結果の保持方法が明らかでないという問題である．2 つ目は，直接 Web ページ上に機能を付加する仕組みが明らかでないという問題である．

プログラム記述の解析結果の保持については，プログラム記述の解析結果を DOM tree 化することで，木構造を崩さず保持できる．直接，Web ページ上に機能を付加する仕組みについては，bookmarklet を用いて理解支援機能の挿入を行うことでクライアントサイドからの機能挿入を実現する．また，ユーザが自らの要求に応じて閲覧機能の拡張を容易に行えるよう，閲覧機能の仕様や記述方法をまとめる．

2　関連研究・技術

プログラム記述の理解支援を目的とした，Web 上で実現される閲覧機能として，Syntax Highlighter[3] が挙げられる．Syntax Highlighter は分類ごとに異なる色やフォントで表示することによって，テキストの可読性を向上させ，文脈をより明瞭にすることができる．

Syntax Highlighter は，Web ページ作成者が HTML にもともと埋め込むものであり，プログラム記述の見栄えはサーバーサイドに依存する．よって，クライアントサイドからの閲覧機能拡張を行う本研究とはアプローチが異なる．

3　閲覧機能

本章では，Web ページ内に含まれるプログラム記述に対し，クライアントサイドから閲覧機能を挿入する方法を提案する．なお，閲覧機能とは，**図 1** のようにプログラム記述の表現を変更し，プログラムの可読性を向上させる機能である．

プログラム記述の表現を変更するには，変更対象となるプログラム記述の構文情報が必要となる．なお，プログラム記述に対する構文解析については，本研究室において現在研究されている HTML 文書内のプログラム記述に対し構文解析を行う手法 [6] を用いて構文解析が行われているものとする．これら，プログラム記述の構文解析結果は何らかの形で保持する必要がある．

HTML 文書の表現をクライアントサイドから変更するには，DOM tree を構築し操作することで実現できる．プログラム記述を構文解析し，その解析結果の構造を DOM tree で表すことで，木構造を崩すことなくクラ

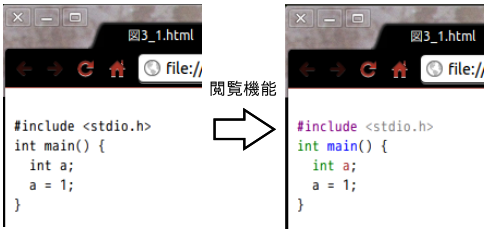

図 1　例：emacs のデフォルトの文字色に変更する

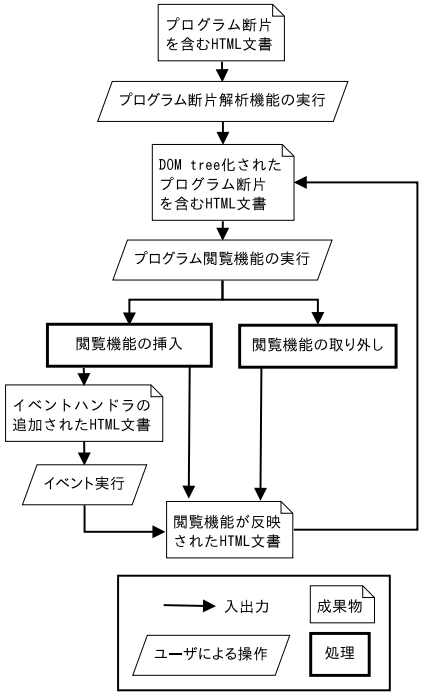

図 2　クライアントサイドから閲覧機能を挿入する流れ

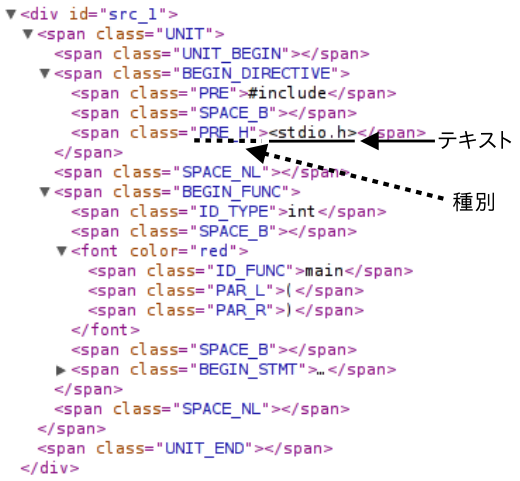

図 3　DOM tree 化したプログラム記述の例

イアントサイドで保持できる．DOM tree の構造で保持することで，JavaScript を用いた操作が容易となる．また，DOM tree を用いて構築した構文木を操作することで，同時に Web ページの表現を変更でき，クライアントにかかる手間が少量で済むといった利点がある．

DOM tree 化したプログラム記述に対して，bookmarklet を用いて閲覧機能を挿入することによって，構文解析の結果を利用した機能をクライアンドサイドから挿入できる．その際，複数の機能の共通点をフレームワーク化し，閲覧機能の記述方法を定義することによって，ユーザによる機能拡張が容易となる．

3.1　閲覧機能の挿入の流れ

クライアントサイドから閲覧機能を挿入する際の処理の流れを**図 2** に示す．プログラム記述を含む Web ページを閲覧しているユーザがプログラム記述解析機能を実行すると，プログラム記述の解析結果は，DOM tree を構築し保持される．このとき，ブラウザの出力画面に変化はない．プログラム閲覧機能を実行すると，DOM tree で表現された解析結果に対して閲覧機能が挿入される．イベントハンドラを用いた場合はイベントを実行することにより機能が反映され，イベントハンドラを用いない場合は直接機能が反映される．このとき複数の機能を実現する方法としてコンテキストメニューを用い，また挿入した閲覧機能の取り外しを行うこともできる．

3.2　DOM tree 化による解析結果の保持

対象とするプログラム記述の構文解析結果は，DOM tree を構築し，保持する．プログラム記述は TEBA[7] によって構文解析され，また，TEBA の解析結果は各字句に対して種別，識別番号，テキストという情報を持っている．プログラム記述の各字句に対して，種別情報をタグとして持たせることで，抽象構文木を DOM tree を用いて構築する．タグを用いたプログラム記述の DOM tree 化の例を**図 3** に示す．

4　閲覧機能の挿入方法

4.1　閲覧機能

閲覧機能には以下の 3 つの機能が必要である．

- 挿入
- 動作
- 取り外し

閲覧機能の実行が動作部分である．取り外し機能は，ユーザが必要なくなった機能だけを取り外す場合に必要となり，これをあらかじめ提供することによって，ユーザビリティが向上する．また，取り外しのときに必要な閲覧機能付加前の DOM tree の複製は，挿入時に事前準備として行うものとする．

4.2　仕組み

4.2.1　挿入

図 4 のような bookmarklet を用いることによって，html の head 内に外部の JavaScript ファイルを呼び出す内容が加わる．このように外部呼び出しを用いることによって bookmarklet の文字数制限を回避できる．ユーザが機能拡張を行うためには，この外部の JavaScript ファイルに関数として機能内容を記述することによって，機能拡張が可能となる．

4.2.2　動作

機能の実現方法として，要素を指定し，スタイル操作やイベント操作を記述することで DOM tree に対して操作を加えることができる．その際，jQuery[4] という

```
javascript:(function(){var%20url='js ファイルのパス';
var%20d=document;var%20e=d.createElement('script');e.src=url;
d.getElementsByTagName('head')[0].appendChild(e);})();
```

図 4　外部の JS ファイルを読み込む bookmarklet 記述

JS ライブラリを用いることによって記述が簡単になる．

要素の指定は構文情報を指定する．DOM tree 化した際に保持している構文情報は class として保持している．例えば，字句情報を表す単体字句に対して機能を挿入するときは ID_TYPE，PRE_H など，構文情報を表す部分木に対して挿入するときは BEGIN_FUNC，BEGIN_STMT などである．

4.2.3　取り外し

bookmarklet で挿入した機能を取り外す方法として，Web ブラウザの更新があるが，複数の機能がすべて外れる．ユーザが特定の機能のみを外したい場合に取り外し機能が必要となる．

取り外しを実現する方法として，DOM tree を複製する方法を用いる．機能付加前のプログラムを保持したまま，その DOM tree を複製し，その複製したものに対して機能付加を行う．機能付加前の DOM tree は HTML 内で記述したままにしておくが，非表示にすることでユーザの目には映らない状態にしておく．この複製を繰り返すと DOM tree 上では初期の状態のプログラム記述の上に保持する．

この記述はすべての閲覧機能を作成するとき，ユーザの記述箇所の前に記述する．これにより，閲覧機能が挿入される度に，自動で HTML 内にプログラム記述の箇所の DOM tree を保存する．

DOMtree を保存した後，表示されている DOM tree を 1 つ削除し，1 番上にあるプログラム記述の DOM tree を表示する．これにより，機能が 1 つ挿入される前の DOM tree が表示され，機能の効果が元に戻され、取り外しが完了する．

4.3　機能の呼び出し方法

機能の呼び出し方法としてコンテキストメニューを用いる方法がある．コンテキストメニューには複数の機能を整理しやすいというメリットがある．

コンテキストメニューを用いて機能挿入および取り外しを行う場合，挿入・動作・取り外しが記述された 1 つの JS ファイルが必要となる．また，その JS ファイルにはコンテキストメニューの記述もされている．コンテキストメニューでは，メニューの項目として，閲覧機能に加え，複製と取り外し機能はあらかじめ提供する．

コンテキストメニューでは，メニューから機能を選択することによって機能が反映される．つまり，メニューから使いたい機能を選択する操作が必要となるので，機能挿入の時点で機能を反映できない．よって機能挿入の時点で反映させる機能を作成する場合は，その機能 1 つを 1 つの JS ファイルに記述して呼び出して使用する．このとき，イベントハンドラを用いることで，コンテキストメニューを用いる場合に比べ，余分な手間を省けるので，ユーザビリティが向上する．この 2 つの方法は，ユーザの要求に応じて使い分けるものとする．

イベントハンドラを用いて機能挿入および取り外しを行う場合，挿入・動作機能が記述された JS ファイルと，取り外しが記述された JS ファイルの、計 2 つの JS ファイルが必要となる．

5　評価と考察

5.1　閲覧機能挿入の評価

3 章，4 章では DOM tree 化されたプログラム記述の生成方法と閲覧機能を実装した．DOM tree 化されたプログラム記述を正しく生成できているのかを検証する．

閲覧機能はコンテキストメニューを用いる場合とイベントハンドラを用いる場合の 2 つがある．閲覧機能を作成する上で，実際にユーザが記述しなければならない箇所を列挙する．列挙した項目が自動化できるかどうかを評価/考察する．

5.1.1　プログラム記述の DOM tree 化と閲覧機能の挿入

プログラム記述の DOM tree 化が正しく生成できるのかを確認する．DOM tree の要素を指定し，その要素を呼び出すことできれば正しく DOM tree 化できたことを判定できる．要素の指定方法として，クラスの指定と id の指定がある．クラスの指定には，解析結果を用いた 24 パターンや html タグ，body タグがある．id タグの src_1 がある．これらの 27 パターンの入力を検証した．

プログラム記述の DOM tree 化は解析結果が字句情報と構文情報が正しい箇所に保持されていることを確認した．また，閲覧機能の挿入のときに，保持した解析結果を用いて DOM tree の要素に対して操作を加えることができた．以上のことから指定した要素に対して，閲覧機能を挿入することができる．

5.1.2　コンテキストメニューとイベントハンドラ

コンテキストメニューは，右クリックで閲覧機能の一覧が表示され，選択した閲覧機能が実行されることを確認した．また，イベントハンドラは，イベント実行により作成した閲覧機能の挿入を行うことを確認した．

閲覧機能を複数実行し，その中の閲覧機能が正しく取り外せたかどうかを確認する．閲覧機能の挿入するときに機能付加前のプログラム記述の複製し，複製したものに対して機能付加を行うことで確認した．また，もともとあるプログラム記述は非表示にする複製されたプログラム記述を削除し，機能付加前の表示に戻ることを確認した．

コンテキストメニューを用いた場合の記述の変更について評価する．bookmarklet のパスを 2 つの JS ファイルに変更するので 2 箇所変更する．挿入の記述は，複製の記述を関数にまとめて，**図 5** のように記述を行う．変更箇所は 3 箇所である．動作の記述は，ユーザが作成する閲覧機能の記述を 1 つの関数にまとめて記述を行う．

```
$(function() {
  $('要素1').contextMenu('myMenu1',
  {
   bindings: {
    'clone':function(t) {
      $("要素2:first").clone().insertBefore("要素3");
      $("要素4:eg(1)").hide();
    },
    'remove': function(t) {
      $("要素5:visible").remove();
      $("要素6:hidden:first").show();
    },
    '関数名':function(t) {
       …閲覧機能の記述…
    }
 });
});
```

図 5　コンテキストメニューを用いた機能拡張の例

変更箇所は 2 箇所である．取り外しの記述は，削除の記述を関数にまとめて記述を行う．変更箇所は 2 箇所である．すべて合わせて 12 箇所の変更を行う．11 箇所の変更において 8 箇所の同様の記述を行った．

イベントハンドラを用いた場合の記述の変更について評価する．bookmarklet のパスを 2 つの JS ファイルに変更するので 2 箇所変更する．挿入の記述は，複製の記述を関数にまとめて記述を行う．変更箇所は 3 箇所である．動作の記述は，ユーザが作成する閲覧機能の記述を 1 つの関数にまとめて記述を行う．変更箇所は 2 箇所である．取り外しの記述は，削除の記述を関数にまとめて記述を行う．変更箇所は 2 箇所である．すべて合わせて 12 箇所の変更を行う．12 箇所の変更において 8 箇所の同様の記述を行った．

5.2　処理の自動化

重複した記述を減らすことでユーザーは，閲覧機能の拡張をより容易に行うことが可能となる．次に示す方法によって改善可能だと考えられる．

html 文書内にコンテキストメニューに表示される項目名は，ユーザが書くことなく，自動生成できると考えられる．その理由として，項目名や呼び出す関数は JSON 形式でテキストに記述させておき，そのテキストに対して，eval 関数を適用すれば JavaScript 記述として読み込むことができる．ユーザが 1 つ 1 つの項目名を記述するよりも，項目名や閲覧機能を挿入する要素，呼び出す関数をテキストのテンプレートにまとめる．テキストは eval で評価することで実装が自動化できる．

複製を行うときや複製したものを削除する記述で対象とする要素の指定は，現在ユーザが記述する必要があり，最も重複する記述であった．この要素の指定の記述を自動化することができればユーザに重複した記述や記述量を減らすことができる．方法として，複製する要素の指定は，プログラム記述の数や閲覧機能の挿入箇所の状況によって要素を変える必要がある．閲覧機能が挿入された要素から親ノードをたどっていき，プログラム記述のマーカーを得る．これにより，どのプログラム記述を複製するか判定する．

6　おわりに

本研究では，Web ページ内のプログラム記述の可読性を高めることを目的とし，Web ページ内のプログラム記述に対してクライアントサイドから挿入可能な DOM tree 変換方法を提案した．プログラム記述の解析結果を保持するためにプログラム記述を DOM tree 化し，タグとして字句情報や構文情報を付加した．これらの DOM tree を用い，ユーザが各々の要求に応じた閲覧機能を拡張できるように，閲覧機能の仕様や記述方法をまとめ，その拡張性について検証，考察を行った．

今後の課題としては，ユーザがより容易に記述できるようにフレームワークを充実させることが必要である．また，ユーザが記述する環境としてのライブラリ化やコンポーネント化，アーキテクチャ等についても考察する必要がある．

参考文献

[1] Eclipse, “Eclipse - The Eclipse Foundation open source community website.,”http://www.eclipse.org/, 2011.

[2] GNU GLOBAL, “GNU GLOBAL source code tag system,”http://www.gnu.org/software/global/, 2011.

[3] Google Project Hosting, “*syntaxhighlighter - Free syntax highlighter written in Java Script - Google Project Hosting*,” http://code.google.com/p/syntaxhighlighter/, 2011.

[4] The jQuery Project, “*jQuery: The Write Less, Do More, JavaScript Library*,” http://jquery.com/, 2010.

[5] 大橋洋貴，山本晋一郎，阿草清滋，“ハイパーテキストに基づいたソースプログラム・レビュー支援ツール，” 電子情報通信学会技術研究報告．SS，ソフトウェアサイエンス，pp.15-22，1998.

[6] 松野秀泰，森瑞穂，“クライアントサイドにおける Web ページ内のプログラム記述の閲覧機能拡張に関する研究 -HTML 文書内のプログラム記述の抽出と解析，” 南山大学 数理情報学部 情報通信学科 2011 年度卒業論文要旨集.

[7] 吉田敦，蜂巣吉成，沢田篤史，張漢明，野呂昌満，“属性付き字句系列に基づくプログラム書換え支援環境の試作，” ソフトウェアエンジニアリング最前線 (ソフトウェア・エンジニアリング・シンポジウム 2010 予稿集)，pp.119-126，Aug. 2010.