Webページ記述内のプログラム断片に対するDOM treeを用いた構文木の構成手法

立道 昂太[1]　吉田 敦[2]　蜂巣 吉成[2]　張 漢明[2]　野呂 昌満[2]

概要：オープンソースのリポジトリサイトやプログラムの解説サイトなど，プログラムを掲載したWebページは多く存在しており，ソフトウェア開発や学習において広く利用されている．Webページのレイアウトは作成者によって異なり，作成者のコーディングスタイルに従ってプログラム記述は書かれているので，必ずしも理解しやすいものではない．プログラム記述を構文解析することで得られる構文木をDOM木に構成できるならば，クライアントサイドからbookmarkletを用いてプログラム記述のスタイルやレイアウトを操作することや，別サイトへのリンクの追加など高度な理解支援も可能になる．本論文ではプログラム記述の操作支援を目的として，Webページ記述内のプログラム解析手法を提案する．Webページのプログラム記述の構成を分類，プログラム記述箇所の特定方法と，構文木をDOM木に変換する方法を提案することで，機能の実現をする．

A Conversion Method of Program Fragments into a DOM Tree as Syntax Tree

Tatemichi Kota[1]　Yoshida Atsushi[2]　Hachisu Yoshinari[2]　Han-Myung Chang[2]　Noro Masami[2]

Abstract: There are many web pages including program descriptions, such as repository sites and learning sites for programming, and they contribute software development and programming education. These pages, however, are sometimes difficult to read for readers, because they are designed from author's views. In this paper, we show a conversion method of program descriptions to sub-trees as parts of the DOM tree. After invoking the conversion method at client side using bookmarklet, we can change program descriptions to our favorite style and layout and add high-level capabilities by modifying the DOM tree. Our method consist of two techniques, detection of program fragments in web pages and construction of the syntax tree of a program fragment including HTML tags. We also show an experiment for evaluating accuracy of our method.

1. はじめに

オープンソースのリポジトリサイトや，プログラムの解説サイトなど，プログラムを掲載したWebページは多く存在しており，ソフトウェア開発や学習において広く利用されている．Webページ上のプログラム記述は，単純にテキストとして掲載される以外にも，閲覧者が理解しやすくなるように，HTMLタグやJavaScriptのライブラリを用いて見栄えが装飾されている．しかし，これはWebページ作成者の立場での読みやすさを基準としているので，必ずしも閲覧者の読みやすさと一致するとは限らない．掲載されているプログラム記述のコーディングスタイルが閲覧者の慣れたスタイルと異なっていることから，理解しづらいと感じる場合もある．しかし，閲覧者はサーバ側に存在するWebページの記述を書換えることはできない．

閲覧者がクライアント上でWebページの内容を加工する方法としてbookmarklet[1]の利用がある．Webページを見やすい形や読みやすい状態に変更したい場合には，bookmarkletを用いてJavaScriptを実行し，Webページを構成するDOM木を操作する方法が用いられる．実際に，背景色の変更，バナー広告を非表示にするbookmarkletなどが公開されている[2]．Webページ上のプログラム記述も見やすい形や読みやすい状態に変更したい場合，プログラム記述の構文木もDOM木の一部にできるならば，構文

[1] 南山大学大学院 数理情報研究科
Graduate School of Mathematical Sciences and Information Engineering, Nanzan University
[2] 南山大学 情報理工学部
Department of Software Engineering, Nanzan University

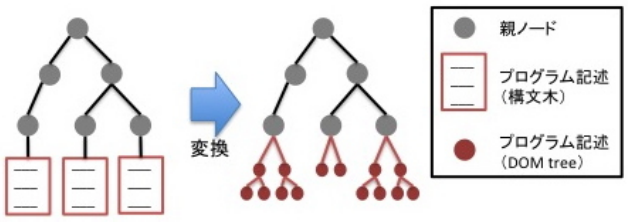

図 1 構文木から DOM tree への変換
Fig. 1 Conversion of program fragments into DOM trees as syntax tree.

木の操作も DOM 木の操作として実現できる．構文木を DOM 木の一部にすることで，jQuery などの JavaScript の既存のライブラリをそのまま活用できる．また，構文木としての操作と同時に DOM 木としての操作も行うので，変更がブラウザ上の表示にすぐに反映されることから，理解支援となる機能を実現しやすい．これにより，プログラム記述についてもクライアントサイドから閲覧者の好みのスタイルに変更することが可能である．例えば，型と変数を見やすくするために，それぞれの色を変更する機能が考えられる．これは，プログラム記述を構文解析することで得られる，型や変数などの字句に対する構文上の種別を用いることで実現できる．

構文木を DOM 木の一部として構成できるのであれば，色やインデントなどのスタイルを変更するだけでなく，他のプログラム記述やライブラリ関数の説明ページへのリンクを追加する機能や，プログラム記述内の関数名一覧を表示し，選択した関数へジャンプする機能など，より高度な理解支援も可能になる．同一ページ内の関数へのジャンプ機能では，字句に対する構文上の種別を用いることで，プログラム記述に含まれるすべての関数名を取得する操作や，関数の記述に対してアンカーとなるタグを付ける操作などが実現できる．

本論文では，クライアントサイドで Web ページ内のプログラム記述を DOM 木の部分木として操作するために，Web ページ記述内のプログラム解析手法を提案する．解析手法としては，プログラム記述を構文解析し，**図 1** に示すように構文木を DOM 木に変換することで，プログラム記述の字句や構文ごとの操作を実現する．

プログラム記述を DOM 木に変換する際には，以下の点が問題となる．

- Web ページ内のプログラム記述の範囲特定方法
- HTML 化されたプログラム断片の解析方法

Web ページ内のプログラムの記述範囲の内側に HTML タグが含まれているとき，プログラム記述が複数に分断されている場合と，最も内側のタグを誤判定している場合の区別がつきにくいことから，範囲の特定が難しい．また，HTML 化されたプログラム記述は，一部の断片だけが記述されたプログラム断片であったり，C 言語のように前処理前の記述が存在する場合や，記述内に HTML タグが含まれる可能性もあるので，一般的な解析器では構文解析できない．本論文では，これらの解決方法として，Web ページ記述内に含まれるプログラム断片を保持するノードを特定し，そのノードの構文木を DOM 木に変換できることを示す．以降では DOM 木への変換を行う際の問題を第 2 節で述べる．Web ページ上のプログラム断片を解析する際の問題である，プログラム記述の範囲を特定する方法を第 3 節で，構文木の構成と DOM 木への変換方法を第 4 節で述べる．これらの提案方法についての評価と考察を第 5 節で示す．なお，本論文では，プログラミング言語として実際の開発にも用いられ，多くの解説サイトが存在する C 言語を対象とする．

2. DOM 木への変換における問題

Web ページ内のプログラム記述を解析するためには，まず，その記述範囲を特定する必要がある．Web ページ内のプログラム記述の範囲は，プログラム全体を囲う最も内側のタグ，すなわち，プログラム記述を包含する DOM 木のノードを特定することで達成できる．しかし，解説文などの自然言語を含む Web ページの場合，プログラム記述とそれ以外との区別をする必要がある．また，プログラム記述の範囲の内側に HTML タグが含まれている場合には，プログラム記述が複数に分断されている場合と，最も内側のタグを誤判定している場合の区別がつきにくいという問題があり，単純には基準を決定することはできない．

Web ページ内のプログラム記述の解析では，HTML タグが含まれている可能性があることや，構文を満たしていない可能性もあることを考慮する必要がある．解析時に HTML タグを削除する方法もあるが，そのページのスタイルに合うように CSS を用いて色付けなどがされていると，見栄えを損い可読性が低下する．ゆえに，タグを削除するかどうかは閲覧者に委ねられるように，タグを残したまま解析することが望ましい．また，Web ページに掲載されるプログラム記述は，一部の断片であったり，C 言語のように前処理前の記述が存在するので，一般的な解析器では構文解析できない．さらに，タグを残しつつ構文木を構築する必要がある．

3. Web ページ内のプログラム記述の範囲特定方法

3.1 Web ページのプログラム記述構成の分類

プログラム記述の記述範囲を特定するにあたり，タグの挿入される位置について，以下のように分類した．

(1) プログラム記述内にタグを含まない
(2) プログラム記述内にタグを含む
　(a) 字句単位でタグが挿入されている
　(b) 字句や構文要素単位でタグが挿入されている

図 2 プログラム記述の HTML ソース例
Fig. 2 A program fragment of HTML source.

(c) その他の位置にタグが挿入されている

HTML ソースでのプログラムの記述例を**図 2** で示す．分類 (1) のプログラム記述内にタグを含まない例は，図 2 のようにプログラム記述全体がタグで囲まれているが，プログラム記述中にタグは存在しない場合である．分類 (2) のプログラム記述内にタグを含む記述方法は，タグの挿入位置によって 3 通りに分類できる．分類 (a) はプログラムの字句単位で，分類 (b) は字句や構文要素単位でタグが挿入されている場合である．これらは，プログラムを HTML 化するツールを用いた場合に構築されうる構造である．分類 (c) は先に挙げた 2 種以外の位置にタグが挿入されている場合である．図 2 のように行末に br タグが存在しているプログラム記述などはこれに分類される．

プログラム記述内にタグが含まれている場合，そのタグを本来特定したいプログラム記述全体を囲むタグと誤って判定し，求めるプログラム記述を特定できない可能性がある．次の節で具体的にどのようにプログラム記述範囲を特定するのかを示す．

3.2 範囲特定方法

特定すべきプログラム記述は，プログラム記述全体を囲む最も内側のタグの組で括られた範囲に存在する．DOM 木においては，プログラム記述を構成するすべてのテキストノードを子孫に持ち，かつ，最も葉に近いノードの子要素に存在する．次の 2 つの方法で特定することを考える．

- タグの種別や構造に着目した範囲特定方法
- プログラム記述の特徴に着目した範囲特定方法

一般的に，タグの使い方には共通性があると考えられる．例えばプログラム記述を Web ページ上に表示させる場合には，pre タグが用いられることが多い．プログラム記述を囲むタグの種別や構造に共通する特徴に着目することで，プログラム記述を含むテキストノードを特定できる．しかし，Web ページのタグの種別や構造は作成者によって異なるので，すべての Web ページの DOM 木がその特徴に当てはまるとは限らない．

プログラム記述の特徴に着目した範囲特定方法は，親ノードのタグの種別や構造に依存せずにどのような構造の DOM 木でもプログラム記述を特定するために，テキストノードのプログラム記述の特徴に着目する．プログラム記述は自然言語に比べて，予約語などの特定の文字列や記号が頻繁に記述されている．ゆえに，テキストノード内に存在する字句の中で予約語や記号の占める割合が多い場合，そのテキストノードはプログラム記述である可能性が高い．この割合が高いテキストノードだけを抽出することで，どのような構造の DOM 木でもプログラム記述範囲を特定できる．しかし，テキストノードの中身がプログラム記述である割合の基準は明確に定まっておらず，プログラム記述だけを確実に特定できるとは限らない．以降の節でそれぞれの方法を説明する．

表 1 プログラム記述を含むテキストノードの特徴
Table 1 Positions of text nodes including a program fragment.

プログラム記述が存在するテキストノードの特徴	件数
pre タグの子ノード	28
body タグの孫ノード (pre タグの子でない)	9
分類不能	3

3.2.1 タグに着目した範囲特定方法

プログラム記述を含むテキストノードを特定する手段として，プログラム記述のテキストノードの親ノードに対応するタグの種類と位置に着目する．親ノードの位置や対応する HTML タグの種類に特徴があるならば，その特徴を基にして記述範囲が特定できる．タグの特徴を探すために，プログラム記述を含む Web ページを検索し，無作為に選んだ 40 件について調査した結果を**表 1** に示す．

表 1 よりプログラム記述のテキストノードは，大きく分けて pre タグの子である場合と body タグの孫であり，かつ，pre タグの子でない場合の 2 種類に分類できる．なお，pre タグの子であり，かつ，body タグの孫の場合は pre タグの子であるとする．分類不能の 3 件に関しては全体の件数から見て数が少なく，共通性がないので例外とする．この 2 種類の構造に着目することで，プログラム記述範囲を特定する．ただし，非プログラム記述がこの条件を満たすことがあるので，プログラム記述と非プログラム記述の区別をつけるために，各ノードの内のテキストに含まれる記述に着目して絞込みが可能か調べた．

プログラム記述のみに頻出し，かつ，非プログラム記述には表われにくい特徴的な記述が分かれば，プログラム記述と非プログラム記述を区別できる．プログラム記述のみに頻出する記述として，演算子や括弧を表す記号や，予約語や main などの文字列が考えられる．ここで，Web ページのテキストノードを基に，記号や予約語などの字句のプログラム記述中での出現頻度を調査する．P_{frac} は対象の字句が Web ページに出現する回数の内，何割がプログラム記述に出現するかをパーセンテージで表し，式 1 で定義される．対象の字句ごとに，複数の Web ページでの出現回数の合計から P_{frac} の値を求め，その値が大きい場合，対象

の字句はプログラム記述に頻出する記述であると言える．

$$P_{frac} = \frac{\text{プログラム記述中での出現回数合計}}{\text{Web ページ中での出現回数合計}} * 100 (1)$$

C 言語の解説サイトの中からプログラム記述と自然言語を含む Web ページをランダムに 10 件選択し，調査した代表的な字句と Web ページ全体の内のプログラム記述中での出現頻度を**表 2** に示す．

調査から，void などの予約語と関数名である main，中括弧やセミコロンなどの記号が，C 言語のプログラム記述に頻出し，かつ，非プログラム記述には頻出しないことがわかった．これらがテキストノード内に占める割合が高い場合，そのテキストノードがプログラム記述であると判定する．本論文では，これらの字句を「プログラム記述に頻出する字句」と呼ぶ．記号の括弧やコロン，予約語の for や if などの字句もプログラム記述に頻出するが，これらは非プログラム記述にも頻出しているので，判定基準には用いない．また，「プログラム記述に頻出する字句」は，C 言語の解説を行なっている Web ページのプログラム記述に頻出する字句を調査した結果から設定したものであり，必ずしも一般性があるとは言えない．実際に特定する際には，解析対象の Web ページに合わせてプログラム記述に頻出する字句の定義を調整する必要がある．

3.2.2 記述の特徴に着目した範囲特定方法

前節で示したタグに着目した範囲特定方法は，タグの種別や位置に特徴がない Web ソースではプログラム記述が特定できない．そこで，タグの特徴や位置に依存しない，プログラム記述の特徴に着目する．

プログラム記述の特徴として，第 3.2.1 節で定義した「プログラム記述に頻出する字句」を利用する．テキストノード内に存在する字句の中で「プログラム記述に頻出する字句」の占める割合が多い場合，そのテキストノードはプログラム記述である可能性が高い．「プログラム記述に頻出する字句」の占める割合を求めながらすべてのテキストノードを出現順に辿り，その割合が著しく増減した時，そのテキストノードの前後がプログラム記述と非プログラム記述の境目であると判定する．例えば，割合が直前のテキストノードの値よりも高くなった場合，このテキストノードがプログラム記述の開始であると特定できる．

4. HTML 化されたプログラム断片の解析方法

4.1 構文木の構成

プログラム記述を構文解析し，DOM 木に変換するためには，次の 2 点を実現する必要がある．

- 構文として不完全なプログラムの解析
- HTML タグを含んだプログラムの解析

以下の各節でそれぞれの解析方法を説明する．

4.1.1 構文として不完全なプログラムの解析

Web ページ上に掲載されているプログラム記述はプログラムの断片であり，構文を満たしていない可能性がある．さらに，C 言語の場合，前処理前の記述を含むと一般的な解析器では構文解析できない．本論文では，解析器として TEBA[3] を用いることで，プログラム断片など構文として不完全なプログラム記述を解析する．TEBA は前処理を行う前のソースコードをそのまま構文解析できる．第 3.1 節で分類したプログラム記述内にタグを含まないものは，TEBA を用いることで解析ができる．ただし，不等号やダブルクォーテーションなどの記号は，HTML 文書では特殊文字としてエスケープされているので，これらの文字参照は予め字句として定義しておく必要がある．また，タグを含んでいる場合は現状の TEBA では解析できない．この解決策を次の節で述べる．

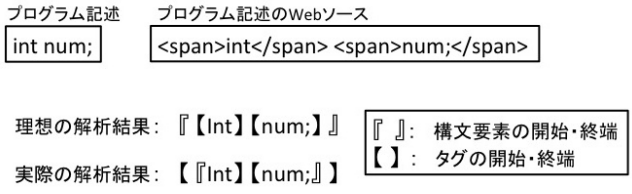

図 3 入れ子構造の問題
Fig. 3 An example of the nested structure problem.

4.1.2 HTML タグを含んだプログラムの解析

第 3.1 節で述べたプログラム記述内のタグに関する分類のうち，分類 (2) に該当するプログラム記述については，タグを含んだまま構文解析を行う必要がある．タグは構文解析をするうえで必要がない字句なので，本研究ではタグを空白類の字句として定義することで解析を可能にする．

分類 (a)，(b) を対象とした解析は，タグを空白類の字句として扱うことで実現できるが，仮想字句の挿入によって入れ子構造の問題が生じる場合がある．TEBA は構文要素を構成する字句系列を囲うように，その構文要素の範囲を表す仮想字句を挿入する．この仮想字句は空文字を持ち，プログラムのテキストとしては存在しない．仮想字句は，構文要素の開始位置の字句の直前と，終了位置の字句の直後に挿入される．このとき，開始タグもしくは終了タグのどちらかが，構文要素を構成する字句系列の開始もしくは終了の位置に存在し，それぞれの対応するタグがその字句系列内に存在する場合，入れ子構造が生じる．

実際に起こりうる入れ子構造の例を**図 3** を用いて示す．理想の解析結果のような構造ならば，正しく構文木を構築できる．しかし，仮想字句は int の字句の直前と ; の字句の直後に挿入されるので，span タグの入れ子よりも内側に付与される．また，int を囲む span タグの終了タグは，「int num;」という構文要素を構成する字句系列の中に存在しているので，入れ子構造となる．

図 3 の例では，構文要素の開始を表す仮想字句と隣接する開始タグ，もしくは，構文要素の終了を表す仮想字句と

表 2 プログラム記述に頻出する字句の調査
Table 2 Investigation of frequencies of lexical tokens in program fragments.

調査対象字句	;	=	{	/*	,	void	main	for
プログラム記述中での出現割合	96%	93%	99%	99%	46%	90%	82%	35%

隣接する終了タグを入れ替えることで，入れ子構造を避けることができる．仮想字句とタグは Web ページ上のプログラム記述には表われない空字句なので，タグと仮想字句が隣接しているならば前後の並びが変わっても Web ブラウザから見たプログラム記述は変わらない．

タグに関する分類 (c) で入れ子構造の問題が生じた場合は，字句でも構文要素でもない単位でタグの挿入されているので，(a)，(b) と同様に字句の入れ替えによって解決できるとは限らない．対処方法として，字句単位で同じタグの組を挿入することで (a) の構造に近似させた後に，字句の入れ替えを行う方法が考えられる．ただし，CSS や JavaScript の指定によっては表示が崩れる可能性もある．

4.2 DOM 木への変換

プログラム記述を解析し，構文木が構成できたら DOM 木に変換する．解析されたプログラム記述は，仮想字句の開始と終了をそれぞれ span タグの開始と終了に置き換えることで DOM 木への変換される．TEBA の解析結果として得られた種別や属性値は span タグの class として定義することで，解析結果の種別ごとに指定できるようにする．また，不等号や空白文字などの記号はエスケープする．DOM 木構築の際に問題になる入れ子構造は，第 4.1 節で定義した方法で解消されているので，DOM 木が構成できる．

5. 評価・考察

プログラム記述の範囲特定方法と，HTML 化されたプログラム断片の解析方法について実装を行い，それぞれ実際の Web ページを用いて評価を行った．

5.1 実装

第 4 節で挙げた構文木の構成と DOM 木への変換をそれぞれ JavaScript で実装した．解析器として用いた TEBA は Perl で実装されている．本論文の提案手法をクライアントサイドで実現するためには，TEBA を JavaScript で記述するか，TEBA をサービス化してウェブで利用できる仕組みを用意する必要がある．TEBA による解析はサーバで変換して JSON 形式で受け取る仕様とし，変換は手動で行った．実装規模は**表 3** の通りである．

5.2 プログラム記述の範囲の特定方法の評価・考察

プログラム記述が存在するノードを特定できるのか，タグに着目した範囲特定方法と，記述の違いに着目した範囲特定方法の 2 つについて，第 3.2.1 節で調査した 40 件に新

表 3 実装規模
Table 3 The numbers of lines of code.

処理	行数
タグに着目した範囲特定方法	291
記述の特徴に着目した範囲指定方法	203
DOM 木への変換	74
JSON 形式への変換 (Perl)	46

たに 60 件加えた合計 100 件の Web サイトを対象として評価する．サイト内では，プログラム記述は統一的な方法で記述されているので，ページではなくサイト全体を対象として調査した．検証対象の Web サイトは第 3 節で考慮した，プログラム記述内にタグが含まれている場合と含まれていない場合，および，pre タグの子，body タグの孫ノードにプログラム記述が存在する場合のすべての場合を網羅している．なお，プログラム記述の位置が分類不能であり，かつ，プログラム記述内にタグが存在しない Web ページが存在する可能性があるが，新たに調査した 60 件にも存在しなかったので，評価対象としない．本論文ではページの都合上，評価結果を一部抜粋し，pre タグの子，body タグの孫ノードにプログラム記述が存在する場合と分類不能の場合，それぞれについてプログラム記述内にタグが含まれている場合と含まれていない場合，それぞれ 2 件づつ，計 10 件の Web サイト内のプログラム記述について**表 4**，**表 5** で示す．

評価としては，プログラム記述だけの範囲特定が「可」か，「不可」，もしくは，一部自然言語を含むが Web ページ内の全プログラム記述の範囲特定はできた「おおよそ可」かで判定する．プログラム記述内にタグが存在する場合，特定範囲の中に一部の非プログラム記述も存在している場合があった．これは，プログラム記述だけの範囲を特定できていないが，範囲内にすべてのプログラム記述が含まれているので，「おおよそ可」と分類する．分断されたプログラム記述が一部でも含まれなかった場合は「不可」と分類した．

評価結果から，タグに着目した範囲特定方法ならば，ほぼプログラム記述の範囲を特定できることがわかった．body タグの孫の場合，「おおよそ可」の断片が多いが，全断片の半数以上は特定可であるので，十分特定できていると言える．しかし，この特定方法はタグの位置にだけ着目しているので，分類不能の Web ページからはプログラム記述の範囲が特定できない．一方，記述の違いに着目した範囲指定方法ならば，すべての対象のプログラム記述をある程度特

表 4　タグに着目した範囲特定方法　評価結果
Table 4 The evaluation result of the detection method based on HTML tag position.

対象			範囲の特定		
分類	タグ	断片数	可	おおよそ可	不可
pre タグの子	無し	18	18	0	0
		10	9	0	1
	あり	20	20	0	0
		17	17	0	0
body タグの孫	無し	60	56	3	1
		45	42	3	0
	あり	48	30	18	0
		55	30	25	0
分類不能	あり	32	0	0	32
		22	0	0	22

表 5　記述の違いに着目した範囲特定方法　評価結果
Table 5 The evaluation result of the detection method based on frequency of lexical tokens.

対象			範囲の特定		
分類	タグ	断片数	可	おおよそ可	不可
pre タグの子	無し	18	7	11	0
		10	10	0	0
	あり	20	3	17	0
		17	14	3	0
body タグの孫	無し	60	18	40	2
		45	18	23	4
	あり	48	22	24	0
		55	33	22	0
分類不能	あり	32	22	8	2
		22	12	10	2

定できた．しかし，タグに着目したときは特定できた記述でも特定できない場合があった．範囲を特定する精度を求めるならばタグに着目し，どのような Web ページでも特定したいならば記述の違いに着目することが適している．

5.3 HTML 化されたプログラム断片の解析の評価・考察

構文木を構成する DOM 木を構成するために，入れ子構造の問題が解決できるのか，解決方法を用いて実際の Web ページを対象にして評価を行った．タグを含むプログラム断片の解析を評価するために，第 3.1 節で分類した，プログラム記述内にタグを含む場合の 3 パターンをすべて網羅するプログラム記述 20 件を評価対象とした．

この 20 件のプログラム断片を DOM 木に変換し，DOM 木が構文木を構成しているか目視で確認したところ，20 件中 1 件のプログラム断片において，DOM 木で構文木が構成できていなかった．このプログラム断片は，第 3.1 節で述べたプログラム記述内のタグに関する分類の (c) に当てはまるものであったので，隣接する仮想字句とタグを入れ替えるだけでは解決ができなかった．この例以外の 20 件中 19 件に関しては，すべての断片で構文木を構成する DOM 木を構成できた．ゆえに，今回の調査では問題は見つからず，入れ子構造の解決方法は妥当であると評価する．

6. おわりに

本論文では閲覧者がクライアントサイドから理解支援となる機能を作成・実行できる環境の実現を目的に，Web ページのプログラム記述特定と，構文木から DOM 木への変換について提案した．プログラム記述特定のために，Web ページのプログラム記述の構成を分類し，2 つの方法を提案した．また，構文木から DOM 木への変換する際に問題となる，入れ子構造の解決策を提案した．これらを用いることで，クライアントサイドから Web ページ内のプログラム断片を DOM 木の部分木として操作でき，プログラム記述の理解を支援する機能を実装可能になる．

今後の課題は，プログラム記述の特定が困難な HTML 記述に対する特定の基準を定めること，理解を支援する機能をユーザが容易に作成できるようなフレームワークを提案することが挙げられる．

謝辞　本研究の一部は，文部科学省研究費補助金基盤 (C)(課題番号:21500042, 22500036, 22500037, 24500049) の助成を受けた．

参考文献

[1] bookmarklets Home Page - free tools for power surfing. http://www.bookmarklets.com/.
[2] JAVASCRIPT::bookmarklet . http://bookmarklet.daa.jp/.
[3] 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. 情報処理学会論文誌, Vol. 53, No. 7, pp. 1882–1849, jul 2012.