6K-05

JavaScript 関数機能の拡張 -オウン変数と関数クローニング-

柏倉　歩†　大谷　真‡
湘南工科大学†　湘南工科大学‡

1. はじめに

Web プログラミングではユーザがクリックした回数を記憶し、それによって表示を変更するなどオウン変数を必要とすることが多い。JavaScript ではクロージャを使えばオウン変数を実現できる。しかし、"関数を生成する関数"を使う必要があるため一般の Web プログラマには難しすぎる。 本研究では JavaScript 言語の関数定義機能を拡張し、プログラマがクロージャを意識したコードを書くことなく簡単にオウン変数を使えるようにした。更に関数クローニングも導入し機能強化を図った。またプリプロセッサも開発した。

2. JavaScript

JavaScript[1]は Netscape Communications 社が 1995 年に開発し、その後標準化もなされ今や多くのプラットフォームでサポートされている。クラス機能を持たないオブジェクト指向言語であり、第 1 級の関数型言語でもある。主として Web プログラミングで広く使われている。クラス機能がないことが欠点だという主張もあり、クラス機能を追加した新言語(TypeScript、JSX、Dart など)も登場してきているが、一方クラスがないことが言語に柔軟性を与えているとの主張もある[2]。本研究は後者の立場に立っている。

3. 課題と解決アプローチ

3.1 課題

Web プログラミングではオウン変数を必要とすることが多く、JavaScript ではクロージャを使えば実現できる。しかしクロージャを用いたプログラミングは一般の Web プログラマには難しいとされる。例えば、Web プログラマ向け国内書籍を調査したところ、JavaScript 書籍のクロージャ掲載率は 15%(6/40 冊)、JavaScript を活用した HTML 作成書籍にクロージャの掲載はなかった(0/14 冊)。このためクロージャ機能を直接使わずにオウン変数が使える機能が必要とされている。

3.2 解決アプローチ

本研究では既に開発されつつあるクラスベースではなく JavaScript の柔軟さを活かすことのできるクラスレスベース(プロトタイプベース)プログラミングで解決する。その前提のもと、現在の関数定義に新たな文法を追加し、"関数を生成する関数"という複雑な概念を使用せず、オウン変数が定義できるようにする。

(1) オウン変数機能の追加

プログラマはオウンブロックを使用し、その中でオウン変数(関数・オブジェクトも可)を定義する。使い易さを考え、関数呼び出しは通常の JavaScript と同じとする。これによりプログラマは見かけ上クロージャを使用せず、オウン変数を使用できるようになる。

(2) 関数クローニング機能の追加

2 つのボタンのクリック回数を別々に記憶しておきたい場合などに、関数の処理は同じだが別の関数を作りたい場合がある。クラスベースの言語の場合は同じクラスのインスタンスを作ることで対処できるが、クラスベースでないオブジェクト指向言語である JavaScript では一般的に Web プログラマに分かり易い手段がない。そこで関数のクローニングという新パラダイムを導入する。

(3) プリプロセッサの開発

追加した関数機能を標準の JavaScript 処理系で処理させるため、標準的な JavaScript に変換するプリプロセッサを開発する。また、変換後の可読性保持のため、できるだけ元のコードを生かした変換とする。

4. オウン変数機能

4.1 仕様

関数定義の中にオウン変数宣言用にオウンブロックを追加した。図 1 の通り文法を拡張した。

```
function 関数名([仮引数リスト]){
  @own
    var オウン変数宣言 1;
    var オウン変数宣言 2;
    ・・・
  @end
  文
}
```

オウンブロックと呼ぶ

図 1 オウンブロックの追加

オウン変数宣言が 1 つなら省略構文を使用できる。

```
@own オウン変数宣言;
```

オウンブロック内で宣言された変数はローカルスコープにあるものの、関数がリターンした後も保持され、次の関数コール時に値が引き継がれる。なお、オウンブロックには変数以外に関数、オブジェクトを宣言することもできる。使用例を図 2 に示す。

```
function fn(){
  @own
    var cnt = 0;
  @end
  return (cnt += 1);
}
fn(); //1 が返る
fn(); //2 が返る
```

オウン変数が 1 つなので以下でも可
@own cnt = 0;

実行例

図 2 オウンブロックの使用例

4.2 変換方式

図 1 をプリプロセッサで図 3 の通り変換する。

```
var 関数名 = (function(){
  var オウン変数宣言 1;
```

Extension of the JavaScript function feature - own variables and function cloning -

†Ayumu Kashiwakura, Shonan Institute of Technology
‡Makoto Oya, Shonan Institute of Technology

Copyright ©2015 Information Processing Society of Japan.
All Rights Reserved.

```
   var オウン変数宣言 2;
   ・・・
   return function 関数名([仮引数リスト]) {
      文
   };
}());
```

図 3 オウンブロックの変換規則

例えば、図 2 は図 4 のように変換する。

```
var fn = (function() {
   var cnt = 0;
   return function fn() {
      return (cnt += 1);
   };
}());
fn(); //1 が返る
fn(); //2 が返る
```

図 4 オウンブロックの変換例

5. 関数クローニング

5.1 仕様

関数定義を行うと関数クローニングを行う clone メソッドが自動的に生成されるようにした。図 5 では関数名の関数をクローン関数名にクローニングしている。

```
function 関数名([仮引数リスト]) {
   @own
     var オウン変数宣言 1;
     var オウン変数宣言 2;
     ・・・
   @end
   文
}
var クローン関数名 = 関数名.clone();
```

図 5 関数クローニング

関数クローニングの使用例を図 6 に示す。

```
function f(v) {
   @own max = undefined;
   if (max < v) {
      max = v;
   }
   return max;
}
var g = f.clone();
f(20); f(10); f(40);   //順に 20、20、40 が返る
g(10); g(40); g(20); //順に 10、40、40 が返る
```

図 6 関数クローニングの例

更にクローニングする際にクローン元関数内にあるオウン変数の値を引数で指定できるようにした。

5.2 変換方式

図 5 は図 7 の通り変換する。

```
var 関数名_kgen = (function() {
   var オウン変数宣言 1;
   var オウン変数宣言 2;
   ・・・
   return function 関数名([仮引数リスト]) {
      文
   };
});
var 関数名 = 関数名_kgen();
関数名.clone = 関数名_kgen;
```

図 7 関数クローニングの変換規則

6. プリプロセッサの開発

6.1 様々な関数宣言方法に対応

JavaScript の関数宣言には 4 つの方法(関数宣言文、有名関数リテラル、無名関数リテラル、Function コンストラクタ)が存在する。これら全てをサポートした。

6.2 変換方法

4.2、5.2 で述べた形で変換する。ただし、Function コンストラクタで宣言された関数に関しては、一旦変換メソッドを通し、関数宣言文で宣言された関数にしてから変換を行うようにしている。関数内にオウンブロックがない場合、無名関数はそのまま、有名関数は関数クローニングを可能にし、出力する。 また、オブジェクト内に存在する関数(図 8)、return 文で返される関数、関数引数に与えられた関数もサポートした。

```
var obj = {x : 1, y : 2,
           f : function() {
                 @own cnt = 0;
                 return cnt++;
            }};
obj.f();
↓変換後
var obj = {x : 1, y : 2, f_kgen : (function() {
   var cnt = 0;
   return function() {
      return cnt++;
   };
})};
obj.f = obj.f_kgen();
obj.f.clone = obj.f_kgen;
obj.f();
```

図 8 オブジェクト内関数の例

6.3 開発結果

プリプロセッサは JavaScript を使用し、開発した。このプリプロセッサによってオウン変数、関数クローニングを実現できた。 実際にはプリプロセッサを組み込んだ変換用 Web ページを用意した。それを通すことで拡張関数を用いたプログラムの変換を行う。

7. 考察と今後の課題

(1) 考察

関数拡張のスペックを考案し、それを JavaScript に変換するプリプロセッサを開発することはできた。しかし、現状、関数の誤検出や検出漏れが存在する。これは、しっかりとした構文解析を行っていないことが原因だと考えられる。

(2) 今後の課題

関数検出の精度を高めるため構文解析ツール(KMyacc など)を使用し、構文解析を行い、それを基に関数検出を行うようにする必要がある。

参考文献

[1]井上他、パーフェクト JavaScript、技術評論社、2012 年
[2]Douglas Crockford、JavaScript:The Good Parts-「良いパーツ」によるベストプラクティス、オライリージャパン、2008 年

Copyright ©2015 Information Processing Society of Japan.
All Rights Reserved.