3K-3

Webアプリケーション開発のための分散JavaScript言語*

加藤真人†
東京工業大学大学院 情報理工学研究科
数理・計算科学専攻

脇田建‡
東京工業大学大学院 情報理工学研究科
数理・計算科学専攻

1 はじめに

Web **アプリケーション**とは，Webブラウザ上で実行され，Webサーバーと相互通信を行いながら動作するアプリケーションのことである．今日のインターネットにおいてWebアプリケーションは重要な役割を担っている．例えば，現在コミュニケーションツールとして盛んに用いられているTwitterの新しいバージョンでは，本格的にWebアプリケーションとしての色を強め，JavaSccriptを用いたリアルタイムな発言の送受信を行ってる．Twitterの例にもれず，Webアプリケーションの発展はインターネットでのコミュニケーションをよりインタラクティブでリアルタイムなものとした．

その一方で，Webアプリケーションの開発は通常のアプリケーション開発と異なり不便な点も多い．例えば，多くのWebアプリケーションではクライアント側とサーバー側での開発言語が異なる．先述のTwiterではクライアント側はJavaScript，サーバー側はScalaといった具合である．このことはプログラマに必要以上の学習コストを強いる．またWebアプリケーション開発の問題としてもう一つ，サーバークライアント間の通信のプログラムをしなければならないというものがある．これもプログラマの責任となり，本来のアプリケーションロジックとは違ったコーディングを必要とする．

本研究ではこれらの問題を解決する手段として，**分散** *JavaScript* **言語**を提案する．分散JavaScriptでは単一の言語，さらには単一のソースコードでサーバークライアント双方の処理を記述することができる．また通信処理も言語の機能としてサポートし，複雑な記述をせずに両者をシームレスに繋ぐことができる．本稿では分散JavaScriptの概略を示し，その利点について論じる．

* A distributed JavaScript language for developing web applications.

† Manato Kato, Dept. of Mathematical and Computing Sciences, Tokyo Institute of Technology

‡ Ken Wakita, Dept. of Mathematical and Computing Sciences, Tokyo Institute of Technology

2 関連研究

既存の解決方法として，まず *Google Web Toolkit* (*GWT*) [1] が挙げられる．GWTは，サーバー用とクライアント用，それぞれのプログラムをJavaで記述することができ，サーバー用のプログラムはそのままサーバーサイドJavaとして，クライアント用のプログラムはGWTのJava-to-JavaScriptコンパイラによってブラウザではJavaScriptのプログラムとして実行される．GWTを使用することでプログラマは両者間を同一の言語で開発することができ，複数言語の学習からは開放される．しかし，サーバー用のプログラムとクライアント用のプログラムは分けて記述する必要があり，通信はRPCによって行う．データのやり取りもシリアライズ可能なオブジェクトに限られる．

また，本研究と似た研究として，稲津らの研究 [2] が挙げられる．この研究ではサーバー側とクライアント側，両者のプログラムをJavaScriptの同一ソースに記述し，適切に分割し実行する．変数にサーバー用かクライアント用かを示す識別子をつけることにより，分割の際の指標とする．単一のJavaScriptにより両者のプログラムを記述するという点で本研究と似ているが，本研究ではさらに多対多の通信も扱うことができる．

3 提案手法

以上を踏まえ，本研究で提案する分散JavaScriptを説明する．分散JavaScriptは，JavaScriptに分散言語の機構を取り入れ，一台のサーバーと複数のクライアントの処理を一つのソースコードによって記述できるようにしたものである．

分散JavaScriptでは単一のソースコードにサーバークライアント両方の処理を記述することができ，従来のWebアプリケーション開発のように複数の言語を使い分けて開発する必要がない．単一のソースコードに記述された分散JavaScriptのコードはサーバー上とク

Copyright ©2011 Information Processing Society of Japan.
All Rights Reserved.

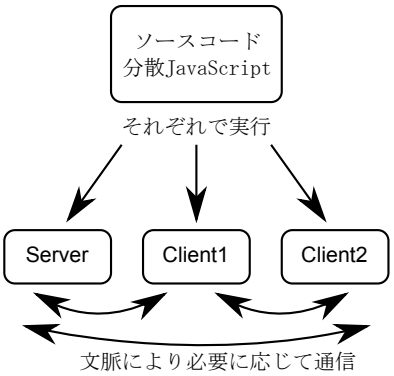


図 1　分散 JavaScript の動作イメージ

ライアント上でそれぞれ実行される．それぞれ通常の JavaScript と同様に動作するが，特別なオブジェクトである *RemoteObject* と呼ばれるオブジェクトのプロパティへアクセスをする時に通常と大きく異なる動作をする．RemoteObject は他のコンピュータ上にあるオブジェクトを指し，そのプロパティへのアクセスは通信を伴ったアクセスとなる．例えば，

```
// 変数 server はサーバーのトップレベルを指す
// RemoteObject
var serverValue = server.databaseValue;
server.permanentValue = userInputValue;
```

とすると，サーバーに定義された変数を他のクライアントから参照したり，代入したりすることができる．また，

```
// 変数 client はあるクライアントのトップレベルを指す
// RemoteObject
client.document.write(serverValue);
```

のようにし，サーバー上の値をクライアントに表示するといったようなことも可能である．これらのコードは実行時には通信を伴うが，それは言語の処理系が行うことであり，プログラマは通信のために特別なコードを書く必要はない．このように，サーバーとクライアントにまたがった通信を伴う処理をシームレスに記述することができるのが本言語の特徴である．

上記のコードは，

```
var clientDocument = client.document;
clientDocument.write(serverValue);
```

とすることもできる．この時，クライアントの document を指す RemoteObject が clientDocument 変数に代入される．このように，数や文字列などの基本型以外のオブジェクトや配列などの値は，RemoteObject としてリモートのコンピュータ上で参照することができる．

以下に例として簡単なチャットアプリケーションのコードを示す．コード中に登場するオブジェクト Octopus は本言語の実装名であり，ランタイム関数を提供するオブジェクトでもある．

```
var chatLog, talk;
if (Octopus.isServer()) {
  chatLog = "";
} else {
  talk = function(message) {
    var server = Octopus.getServer();
    server.chatLog += message + "<br>";
    document.write(server.chatLog);
  }
  document.getElementById("button").click =
  function(){
    talk(document.getElementById("form").value);
        document.getElementById("log").value =
            Octopus.getServer().chatLog;
  }
}
```

このようなコードを定義することで，form に発言内容を入力し，button を押すことでサーバーのログに発言内容が反映され，log にその内容が表示されるようになる．フォームの作成や発言内容の更新など細かいコードは省いているが，立派なチャットアプリケーションと言えるだろう．このように，分散 JavaScript 言語を用いれば，インタラクティブな Web アプリケーションを簡単に記述することができるようになる．

4　まとめ

Web アプリケーション開発を容易にする手段として，分散 JavaScript の提案を行った．分散 JavaScript を用いることで，サーバークライアントそれぞれで異なる言語を使用すること無く Web アプリケーションの開発をできるようになった．また，サーバー・クライアントにまたがる処理をシームレスに記述することもできるようになった．分散 JavaScript 言語を使用することで，プログラマはより本質的なコードに専念してプログラミングできるようになったと言える．

参考文献

[1] Bruce Johnson. Reveling in constraints. *Queue*, Vol. 7, pp. 30:30–30:37, July 2009.

[2] 稲津和磨, 岩崎英哉. サーバ/クライアント自動分割を備えた Web フレームワークの設計と実装. 情報処理学会論文誌 プログラミング, Vol. 3, No. 4, pp. 1–15, 2010.

Copyright ©2011 Information Processing Society of Japan.
All Rights Reserved.