# JavaScript を用いた Web アプリケーション開発支援の研究
## − 状態遷移機械を用いた開発方法の提案 −

2007MI090 上久保 湖美　　2007MI135 三浦 弘貴
指導教員　蜂巣 吉成

## 1　はじめに

JavaScript を用いて操作性を向上させた Web アプリケーションが普及している．Web アプリケーションでは文書構造を HTML, 見栄えを CSS, 振舞いを JavaScript と分離して記述することが推奨され [1]，文書構造と振舞いは jQuery[2] により分離できる．一般的に Web アプリケーションでは，操作性の向上や機能追加などのために，HTML 要素を変更することがよくあるが，jQuery を用いた記述では変更箇所が散在していることがあり，変更に手間がかかる．例えば，HTML 要素をブラウザに出力する方法として，テキストから画像へ変更することがある．このとき，変更する HTML 要素に対しての処理が複数の関数内に存在していることがあり，複数の箇所を変更する必要がある．変更の種類としては，振舞いの変更と見栄えの変更もあるが，これらは JavaScript を関数ごとに変更することで容易にできる．

本研究では，変更箇所を局所化した変更のしやすい Web アプリケーションの開発支援を目的とし，状態遷移機械を用いた Web アプリケーションの開発支援方法を提案する．Web アプリケーションの振舞いを HTML 要素ごとの処理に分離し，状態遷移機械を考える．複数の HTML 要素はイベント送信によって協調動作をおこなう．通信処理は通信オブジェクトと HTML 要素の状態遷移機械とのイベント送信によりおこなわれる．状態遷移機械を HTML 要素ごとで考えているので，散在していた処理は分離した上でまとめられる．HTML 要素の変更をおこなう場合，変更したい HTML 要素以外の HTML 要素の処理を変更することなく，変更箇所を局所化できる．

本研究では，JavaScript により記述され，Ajax 技術を用いて通信をおこなうページ遷移を伴わない Web アプリケーションのクライアント側を対象とする．JavaScript の記述には jQuery を用いる．

## 2　Web アプリケーションの問題点

### 2.1　対象となる Web アプリケーション

一般に対象とする Web アプリケーションは次のような処理や関数で構成される．

(a)　ユーザが HTML 要素を操作したときに実行される関数
(b)　サーバとの通信処理，サーバからの取得した結果をどのように扱うかの処理
(c)　サーバからの取得した結果により，HTM 要素を操作する処理
(d)　時間によって処理を実行するタイマー関数
(e)　ブラウザオブジェクトの処理

本研究では (c) のうち既存の HTML 要素に対する処理を対象とする．なお，HTML 要素を新規作成する処理は，パターン化してブラウザに反映する処理を簡潔に記述する方法が提案されている [3]．(e) のブラウザオブジェクトのうち history オブジェクトなどに関しては，ページ遷移について扱っていないので，対象外とする．

### 2.2　Web アプリケーションの変更

従来の Web アプリケーションでは, ユーザからクリックなどのイベントが起こると JavaScript 内の関数を呼び出し，複数の HTML 要素を参照し振舞いをおこなうことが多い．また，サーバと通信をおこなった結果から HTML 要素を操作する際，通信をおこなう関数内で HTML 要素を参照することもある．ユーザからのイベントで呼び出された関数と通信における関数など複数の関数にひとつの HTML 要素に対する処理が記述されている場合があり，HTML 要素の変更をおこなう際，変更箇所が散在し手間がかかる．

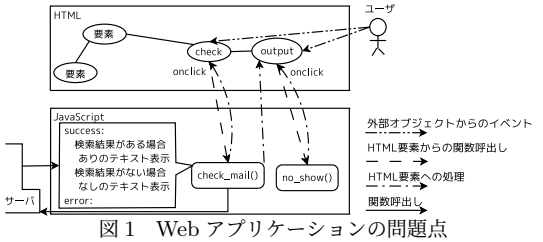


図 1　Web アプリケーションの問題点

**図 1** はメールアプリケーションにおける未読メール数を表示する処理を示した図である．アプリケーションはサーバと通信をおこない，未読メールがあるならば「未読あり」の表示し，なければ「未読なし」と表示する．サーバとの通信はユーザが check 要素をクリックしたときにおこなわれる．未読表示はユーザが「未読あり」を押すとメールを開き「未読なし」に変わる．output 要素の処理は複数の関数内に存在するだけでなく，通信をおこなう関数内でも結果がある場合とない場合の処理があり，テキスト表示を画像表示に変更したい場合，変更箇所の特定や JavaScript コードの変更に手間がかかる．

## 3　状態遷移機械による開発支援

本研究では，HTML 要素に着目し，HTML 要素の状態，ユーザ操作などのイベントによる遷移，2.1 節であげた処理・関数を HTML 要素ごとの処理に分離し遷移に伴うアクションとしてまとめ，それらを組み合わせて HTML 要素ごとの状態遷移機械として考える．

これらの状態遷移機械はイベントを送受信し，状態の遷移によってアクションを起こすことで，Web アプリケーションが動作する．**図 2** に本研究による Web アプリケーション開発の概要を示す．点線で囲んだように HTML 要素ごとの状態遷移機械として分離し，開発

者は従来の JavaScript の記述ではなく，HTML 要素ごとの状態遷移機械を記述する．状態遷移機械による記述を変換器で JavaScript コードに変換し，Web アプリケーションに適用することで変更のしやすい Web アプリケーションが開発できる．

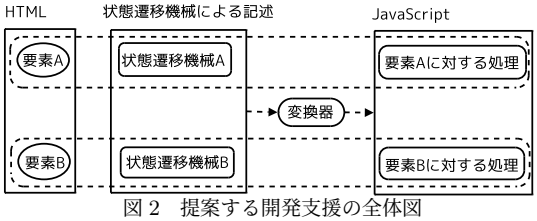


図 2　提案する開発支援の全体図

### 3.1　振舞いの分離

変更箇所を局所化するために，振舞いの分離をおこない，HTML 要素ごとに処理をまとめる．振舞いの分離をおこない HTML 要素ごとの処理として分離し，イベントの送受信により HTML 要素を協調動作させる．

通信をおこなう関数では，通信の結果によって HTML 要素の見栄えや振舞いを操作していることが多い．**図 3** の場合，通信処理と output 要素の見栄えの変更処理を分離して状態遷移機械におけるアクションとする．通信処理を通信オブジェクトとして HTML 要素の状態遷移機械にイベントを送信することで通信処理と HTML 要素を協調動作させる．

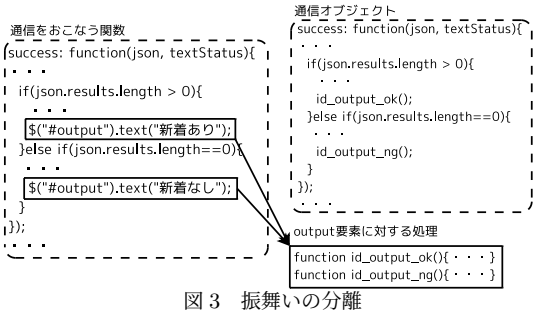


図 3　振舞いの分離

Web アプリケーションは HTML 要素に対する処理と各オブジェクトによって構成されるものとし，イベント送信で関連のある HTML 要素を協調動作させることで，振舞いを分離する．

振舞いを分離し，状態遷移機械を用いることで，HTML 要素ごとに処理をまとめることができ，変更箇所の識別が容易になる．Web アプリケーション全体は状態遷移機械の集合として捉えることができ，状態遷移機械とオブジェクト間のイベントの送受信を考えれば状態遷移機械ごとの再利用が可能となる．

### 3.2　Web アプリケーションの構成

従来の Web アプリケーションでは，**図 1** のように，ひとつの HTML 要素の振舞いが複数の関数にまたがって存在し，output 要素を変更すると，自身の振舞い no_show() 関数と check 要素をクリックしたときに実行される check_mail() 関数を変更する必要があり変更箇所が散在していた．状態遷移機械を用いることで，**図 4** のようなモデル図になる．check_mail() 関数から output 要素に対する処理 ok()・ng() を分離した上で，**図 4** の HTML 要素と JavaScript 内の振舞いを四角で囲んだように HTML 要素ごとで処理を分けて考えれば，それぞれの状態遷移機械ごとで変更をおこなうことができ，HTML 要素の変更に対して処理の変更箇所を限定できる．

**図 4** は，HTML 要素ごとで振舞いを分離した上で，check 要素と output 要素の状態遷移機械をモデル化した図となっている．**図 4** に示したように Web アプリケーションは次の要素から構成される．

- HTML 要素の状態遷移機械
- 通信オブジェクト
- タイマーオブジェクト
- ブラウザオブジェクト

通信オブジェクトは 2.1 節の (b) をおこなう．タイマーオブジェクトは (d) をおこなう．イベントの受信によりタイマーを開始し，タイムアウトすると，状態遷移機械や通信オブジェクトへイベント送信をする．ブラウザオブジェクトは (e) をおこなう．

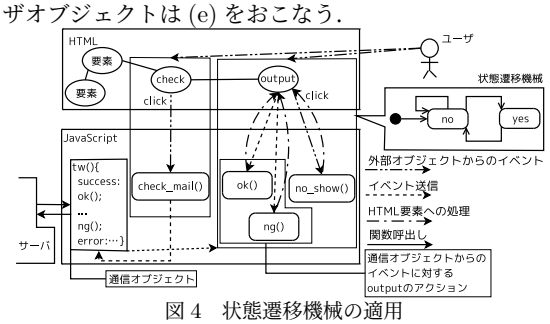


図 4　状態遷移機械の適用

**図 4** は，ユーザが check 要素をクリックしたときに，check 要素の状態遷移機械から通信オブジェクトに対してイベント送信をし，サーバとの通信をおこなった結果を output 要素の状態遷移機械に対してイベント送信する．また，ユーザが output 要素をクリックしたときは，出力されている「新着あり」のテキスト表示を，「新着なし」というテキスト表示に切り替える処理をする no_show() 関数が実行される．

上述のように，状態遷移機械とオブジェクト間のイベント送信を考えて，Web アプリケーションを構成し，状態遷移機械は各要素ごとで考えるものとする．

### 3.3　状態遷移機械

状態遷移機械の状態，イベント，アクションについては，次のように考える．状態遷移機械は HTML 要素とそれに対する処理をおこなう関数によって構成され，オブジェクト間や状態遷移機械間でイベント送信をおこなうことで，Web アプリケーションを構成する．

#### 3.3.1　状態

状態については，次の 2 つに分類する．

(a)　ブラウザに表示された HTML 要素の見栄えの違いを区別するための状態

(b)　JavaScript のプログラムにおいて HTML 要素に対する処理の違いを区別するための状態

ボタンをクリックすると画像が拡大するアプリケーションを考える．画像は 3 回まで拡大可能とする．このとき拡大する画像は見栄えに変化があり (a) として

分類できる．また，ボタンはブラウザに表示されているHTML 要素の見栄えは変わらないが，状態はクリックしたときに画像が変化する状態から，クリックしても画像を変更しなくなる状態へ変化している．このように，ブラウザの表示の変化はなく，JavaScript の処理が変化しているだけである状態は (b) に分類される．

#### 3.3.2 イベント

イベントはユーザの操作による外部イベントと状態遷移機械やオブジェクト間で送受信される内部イベントに分類される．ひとつの関数内に存在する複数の HTML 要素に対する処理をひとつひとつの HTML 要素に対する処理に分離し，関連のある HTML 要素を協調動作させるためにイベント送信をおこなう．

通信オブジェクトの呼び出しや通信オブジェクトからの結果をイベント送信と考えることで，通信オブジェクトは他の HTML 要素と関係し，イベントをきっかけにアクションを起こさせる．タイマー関数による他の関数の呼び出しは，イベントによってタイマーオブジェクトが起動し，他の HTML 要素へイベント送信しているものとする．

#### 3.3.3 イベントに対するアクション

アクションとしては，HTML 要素自身に対する振舞い, HTML 要素自身に対する見栄えの変更, 他の HTML 要素へのイベント送信を記述する．他の HTML 要素の見栄えの変更は，イベント送信をすることで他の HTML 要素自身でおこない，各 HTML 要素に対しての処理として記述を分離する．

### 3.4 グループ化

ある HTML 要素が表示されると別の HTML 要素が非表示に変わるような，ふたつ以上の HTML 要素が連動して動作する場合は HTML 要素をグループ化する．連動している HTML 要素を子要素，子要素を囲む HTML 要素を親要素とし，親要素の状態遷移機械に連動して複数の子要素の状態遷移機械を管理する．グループ化をすることで，イベント送信先を複数箇所で指定していたのに対し，親要素へイベント送信を考えるのみで子要素の切替えがおこなえる．また，他の HTML 要素へのイベント送信も親要素のみで扱える．イベントの受信が多く，他の状態遷移機械との関係を複雑にすると「切り替え機能」として変更するときに手間となってしまう．切り替え機能としては，親要素に対してイベントが起こったとき，親要素が子要素に対してのイベント送信をする．それぞれの子要素内では，子要素自身の状態を判断して状態を切り替える．

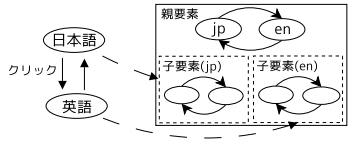


図 5　グループ化した HTML 要素のモデル図

### 3.5 class 属性による状態遷移機械

本研究では振舞いを HTML 要素の処理ごとに分離し，状態遷移機械でまとめる．しかし，HTML 要素ごとで状態遷移機械を適用すると状態遷移機械の数が増え，変更の際に複数の状態遷移機械を変更する必要がある．現在の Web アプリケーションでは CSS で class セレクタを使用し，文書内で複数の HTML 要素に対してスタイルを適用している．class によって指定された見栄えが同じときに同じ振舞いをしていることが多い．同じ振舞いをする HTML 要素がある場合，class ごとの状態遷移機械として処理を共通化する．

### 3.6 状態遷移機械の記述

従来の開発方法ではユーザからのイベントを受け，イベントごとに関数で処理を記述していた．状態遷移機械の記述では，各 HTML 要素の状態の遷移に伴うアクションを記述する．従来も JavaScript により状態遷移機械をコードとして記述することは可能であるが，状態やイベントが判別しにくくなることや状態を遷移させる記述が必要となる．本研究では，状態遷移機械を明示的に記述するための記法を提案し，変換器により JavaScript コードを生成することで，HTML 要素ごとに処理を分離して記述でき，状態に伴うアクションがわかりやすくなる．**図 6** に提案する記述の具体例を示す．

```
#button({
  (can, click[num<4], can)([
    id_color_change();
    num++;
  ])
  (can, click[num==4], cannot)([
    id_color_change();
    num++;
  ])
  (cannot, click, cannot)([
    alert("選択できません");
  ])
});
```

```
要素名({
  (遷移前の状態，イベント名，遷移後の状態)([
    アクション
  ])
});

イベント送信：(属性名)_(要素名)_(イベント名)
```

図 6　状態遷移機械の記述

HTML 要素ごとに ({,}) 内に状態遷移と遷移に伴うアクションを記述していく．**図 6** の#button のように id 属性による要素指定は「#」を，class 属性による要素指定は「.」を要素名の前に記述する．

状態遷移は遷移前の状態，イベント名，遷移後の状態をカンマで区切り括弧で囲い記述する．**図 6** の (can, click[num==4], cannot) のように記述することで，状態の遷移を表す．click などのユーザからのイベント（外部イベント）についてはそのまま記述し，内部イベントは，イベント名の後ろに小括弧を記述する．ガード条件は**図 6** の click[num==4] のようにイベント名の後ろに [, ] を記述し中に遷移の条件を記述する．

HTML 要素のアクションとして他の HTML 要素へイベントを送信する場合，id 属性による要素指定は id を，class 属性による要素指定は cl を属性名とし，関数を呼び出すことで表現する．イベント送信の際にアクションで算出された計算結果や，自身の現在の状態を他の HTML 要素に送信する場合は，イベント名の後ろの小括弧の中に引数を記述する．状態遷移機械の初期状態は遷移の最初に記述された状態とし，state 属性に状態を保持させる．**図 6** の状態遷移機械では can が初期状態となる．

## 4 実装と評価

### 4.1 実装

本研究で提案する状態遷移機械を用いる方法でアプリケーションが開発できることを確認するために変換器の実装をおこなった．

状態遷移機械による記述から，出力される JavaScript ファイルの仕様を決めた．ユーザからのイベントは jQuery を使用した関数として出力する．ページが読み込まれた際に ready メソッドにより HTML 要素へ状態を表す state 属性を追加する．追加した state 属性に初期状態を保持させ，イベントの受信により属性値を変化させることで状態を遷移させる．今回，変換器は Perl で実装し，70 行になった．

### 4.2 評価

Web アプリケーションに提案方法を適用して，実際に機能変更をおこない，変更の容易性の評価をおこなった．評価には TwitterAPI を使用した Web アプリケーションなど 4 つのアプリケーションで HTML 要素の変更をおこなった．

#### 4.2.1 変更の容易性

評価にあたり，メールアプリケーションにおける未読メールの有無を表示する処理を例にあげる．このアプリケーションで未読あり／なしの表示を画像による表示へと変更をおこなった．

状態遷移機械の適用前

```
function tw(){
 . . .
 if(json.results.length > 0){
  . . .
  $(<div>).attr("id","n_mail")
          .attr("value", "未読あり");
 }else{
  . . .
  $(<div>).attr("id","n_mail")
          .attr("value","未読なし");
}
. . .
$(function(){
 . . .
 $("#n_mail").click(function(){
  $("#n_mail").attr("value", "未読なし");
  mail_open();
 });
 . . .
```

状態遷移機械の適用後

```
#n_mail({
 (no, ok(num), yes)([
  $("#n_mail").attr("value", "未読あり");
 ])
 (no, ng(), no)([
  $("#n_mail").attr("value", "未読あり");
 ])
 (yes,click,no)([
  $("#n_mail").attr("value", "未読なし");
  id_mail_open();
 ])
})

#search({
 . . .
```

図 7　状態遷移機械の適用

id 名が n_mail 要素でメールの有無をテキスト表示させる HTML 要素に対する処理について考える．状態遷移機械を適用する前は**図 7** のように，n_mail 要素の処理に対する記述が通信関数内や click メソッドによる関数内に複数箇所あった．表示方法をテキスト表示から画像へ変更する際，複数箇所ある n_mail 要素に対する JavaScript コードの記述を変更する必要がある．

状態遷移機械の記述にすることで HTML 要素ごとで処理がまとまって記述される．HTML 要素を変更したことにより書換えなければならない箇所は n_mail 要素に対する処理だけと識別することができ，従来は必要であった複数の呼び出し箇所の識別や変更が不要になった．テキスト表示をおこなう処理と画像表示をおこなう処理の状態遷移機械は同じであるので，遷移に伴うアクションを変更するだけで差し替えがおこなえた．

#### 4.2.2 class 属性による状態遷移機械

本研究では同じ振舞いをする HTML 要素を class 属性で指定し，共通の状態遷移機械を適用した．class による処理としては class 全体に対する処理と個々の HTML 要素に対する処理がある．

まず，画像が複数あり，ある画像をクリックするとその説明文をアラート表示するアプリケーションを考える．それぞれの画像はひとつの状態を持ち，遷移に伴いアラートを表示する．クリックされた画像は自分自身を指す「$(this)」でイベントが起こった HTML 要素自身を処理対象として遷移させることができた．

また，「フレームを追加／削除」ボタンをクリックすると複数の画像の周囲に枠線が描画されるアプリケーションでは，class に対してイベント送信をすることで，すべての画像の状態を遷移させることを確認した．

複数の画像がありボタンを押すと連動する画像が拡大するアプリケーションを考える．複数の画像は同じ状態遷移機械となり，class による状態遷移機械として管理する．ボタンの HTML 要素からイベントを送信する際，class のうちの特定の HTML 要素を指定することができず，状態を遷移させることができなかった．

class 属性の特定の HTML 要素に対してイベント送信をする際に，HTML 要素を指定して遷移させる方法も考えられるが，class 属性全体にイベント送信をおこなう処理に影響がでてしまう．Web アプリケーションでは class 属性が同時に動作する場合と特定の HTML 要素が動作する場合のどちらの処理も考えられることから共通のクラスを持つ HTML 要素全体にイベントを送ることと個々の HTML 要素にイベントを送ることを区別する必要がある．

## 5 おわりに

本研究では，変更のしやすい Web アプリケーションの開発支援を目的とし，状態遷移機械を用いた開発支援方法の提案をおこなった．複数の HTML 要素に対する振舞いを HTML 要素ごとの処理に分離し，状態遷移機械を用いることで，変更箇所の識別を容易にした．また，提案した方法を適用し，変更の容易さについて評価した．今後の課題としては class 属性の状態遷移機械についての問題や状態遷移機械を部品として再利用することの検討などがあげられる．

## 参考文献


[1] S. Willison, “*Unobtrusive JavaScript with jQuery*, ” `http://simonwillison.net/static/2008/xtexh/`, 2008.

[2] J. Resig and the jQuery Team, “*jQuery*, ” `http://jquery.com/`, 2009.

[3] 近藤友紀，村松雅大，“*JavaScript* を用いた *Web* アプリケーション開発支援の研究–*HTML* 要素と通信データの対応関係を用いた記述方法の提案–”，南山大学　数理情報学部　情報通信学科，2010 年度　卒業論文要旨集