清水 孝将 / 伊地知 聖貴

株式会社ミガロ.

システム事業部 システム1課

JC/400でポップアップウィンドウの制御&活用ノウハウ!

JC/400 の標準機能と JavaScript の 2 つのアプローチにより、
ポップアップの開発手法、活用例を紹介する。

略歴　清水 孝将
1983 年 10 月 04 日生
2008 年甲南大学文学部卒
2008 年株式会社ミガロ . 入社
2008 年 04 月システム事業部配属

現在の仕事内容
入社 5 年目で Delphi/400 や JC/400 の開発業務を担当。Web に関する知識や技術を身につけ、Web アプリケーションのスペシャリストを目指している。

略歴　伊地知 聖貴
1988 年 10 月 13 日生
2011 年立命館大学映像学部卒
2011 年株式会社ミガロ . 入社
2011 年 04 月システム事業部配属

現在の仕事内容
Delphi/400 や JC/400、RPG の開発を担当。幅広いプログラム言語を身につけ、さまざまなニーズに対応できる SE を目指している。

1.Web開発におけるポップアップウィンドウの活用

「ポップアップウィンドウ（以下､ポップアップ)」とは､画面上のボタンクリックなどによって新たに立ち上がるウィンドウのことである。

例えば、一覧でレコードを表示する機能と、一覧から選択したレコードの詳細内容を表示する機能を、Web で実現する場合を想定してほしい。この場合、一覧画面でボタンをクリックすることで、すべてが詳細画面に切り替わってしまうと、一覧画面の他のレコード情報と比較して詳細画面を見ることができなくなってしまう｡このような場合に､ポップアップを有効に活用できる。

Web でのポップアップの表示には、2 種類の方法がある。

●新規ウィンドウ

1 つ目は、新規ウィンドウとして表示する方法である。新規ウィンドウの生成は､JavaScript の window.open() メソッドを用いることで簡単に実装できるため、誰でもすぐに実現が可能である。

ただし、この方法はブラウザやセキュリティソフトの機能によって、ポップアップがブロックされることが多い。その理由は、この機能が昨今、フィッシング広告としても使われており、対策としてブロック機能が用意されているからである｡その結果､最近ではこの形式のポップアップは使用されることが少なくなっている。

● HTML にタグ要素追加

2 つ目は、HTML 上へ新たなタグ要素を追加することで、同じウィンドウ上にポップアップとして表示する方法である。この方法は、JavaScript のコーディングを必要とするが、新規ウィンドウが立ち上がらないため、ポップアップをブロックされることがないという特徴を持っている。

JC/400 標準のポップアップ機能には、2 つ目の方法を採用している。

2.JC/400標準機能による実現

JC/400 では、HTML へ新たなタグ要素を追加することで、同じウィンドウ上にポップアップが表示可能である。

実装方法も非常に簡単である。HTML を JC/400 Designer で配布する際に、画面右上にある「新規ウィンドウを開きますか？」のチェックボックスをオンにし、その画面の表示位置（top,left)、幅（width)、高さ（height)、クラス（適用する CSS）を設定して配布するだけである。【図 1】【図 2】

また、ポップアップとして配布することによる RPG のコーディングへの影響はなく、ポップアップであることを意識せず、通常画面と同じ手法で開発することが可能である。

3.JavaScriptによる実現

JC/400 では、IBM i にリクエストを

図1

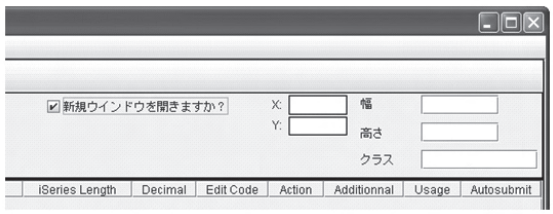

新規ウインドウを開きますか？
X:
Y:
幅
高さ
クラス
iSeries Length
Decimal
Edit Code
Action
Additionnal
Usage
Autosubmit

図2

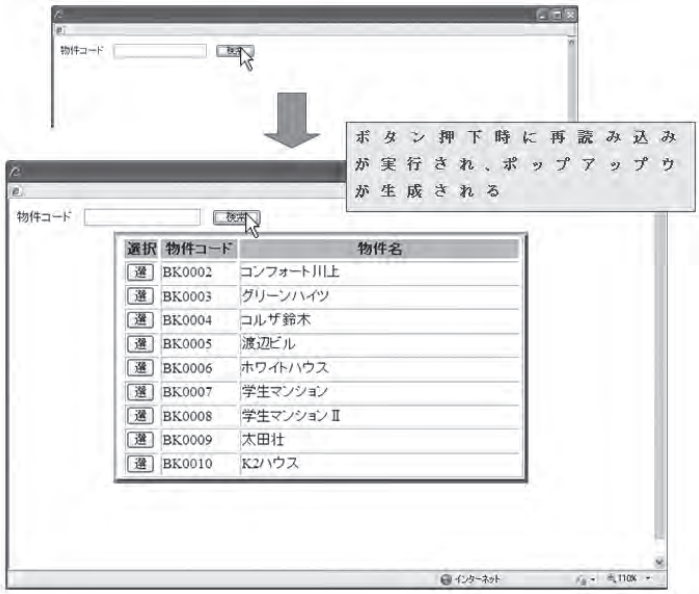
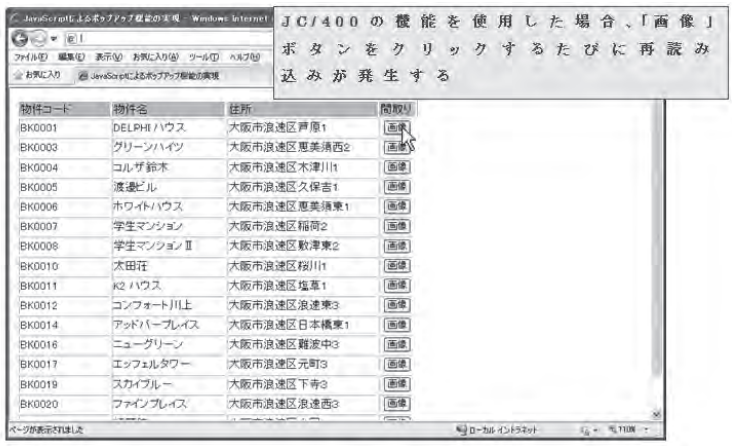

物件コード
検索
ボタン押下時に再読み込みが実行され、ポップアップウが生成される
選択
物件コード
物件名
選 BK0002 コンフォート川上
選 BK0003 グリーンハイツ
選 BK0004 コルザ鈴木
選 BK0005 渡辺ビル
選 BK0006 ホワイトハウス
選 BK0007 学生マンション
選 BK0008 学生マンションⅡ
選 BK0009 太田荘
選 BK0010 K2ハウス
インターネット

図3


JC/400の機能を使用した場合、「画像」ボタンをクリックするたびに再読み込みが発生する
物件コード
物件名
住所
間取り
BK0001 DELPHIハウス 大阪市浪速区戎原1 画像
BK0003 グリーンハイツ 大阪市浪速区恵美須西2 画像
BK0004 コルザ鈴木 大阪市浪速区木津川1 画像
BK0005 渡辺ビル 大阪市浪速区久保吉1 画像
BK0006 ホワイトハウス 大阪市浪速区恵美須東1 画像
BK0007 学生マンション 大阪市浪速区稲荷2 画像
BK0008 学生マンションⅡ 大阪市浪速区敷津東2 画像
BK0010 太田荘 大阪市浪速区桜川1 画像
BK0011 K2ハウス 大阪市浪速区塩草1 画像
BK0012 コンフォート川上 大阪市浪速区浪速東3 画像
BK0014 アッドパープレイス 大阪市浪速区日本橋東1 画像
BK0016 ニューグリーン 大阪市浪速区難波中3 画像
BK0017 エッフェルタワー 大阪市浪速区元町3 画像
BK0019 スカイブルー 大阪市浪速区下寺3 画像
BK0020 ファインプレイス 大阪市浪速区浪速西3 画像

送り、そこでHTMLを再作成することにより、ポップアップを実現している。そのため、ポップアップを表示する際には、画面の再読み込みが必要である。

IBM i でHTMLを再作成する理由は、ポップアップにIBM i のDB情報を反映するからである。したがって、DB情報を取得しなくてもよい簡易な画面をポップアップさせたい場合は、再読み込みの必要がなくなる。

例えば、図3のような物件情報を照会する画面で、各物件の間取り画像を表示するようなポップアップのケースでは、必要な情報は画像ファイルの保管パスだけである。それだけの情報ならば、呼出元画面の読み込み時に取得しておけば、再度DB情報を取得する必要がない。そして再読み込みがなくなれば、ポップアップの表示速度が上がり、明細ごとの画像の確認がスムーズになる効果も期待できる。【図3】

このような、簡易なポップアップの表示は、JavaScriptを利用することで実現可能である。【図4】

次項から、JavaScriptを用いて、再読み込みの発生しないポップアップの表示に必要なコーディングを解説する。【ソース1】

3-1. タグ要素の作成

ポップアップのもととなるタグ要素を作成する。このタグ要素をHTML上に追加することで、ポップアップが表示できる。

HTMLの記述は、ソース2のように、<BODY>タグの中にブロック要素の<DIV>やインライン要素の<SPAN>タグを記述し、さらにその中に<TABLE>や<INPUT>タグを記述するといった階層構造になっている。【ソース2】

その階層構造における上下の階層の関係のことを「親子関係」と呼んでいる。例えば、③と④の関係は<SPAN>タグが親となり、<TABLE>タグが子となっている。④と⑤で見た場合は<TABLE>タグが親となり、<THEAD>タグが子という関係となる。

また、親要素への処理は、そのまま子要素にも適用されるという特性を持っている。親であるタグ要素をHTMLに追加すると、その子であるタグ要素も追加される。

この親要素は、ソース1の①にあるdocument.createElement（タグ要素名）メソッドによって、「ParentTag」という変数名で作成している。

3-2. タグ要素へ子要素を追加

作成したParentTagの子要素となるタグを追加する。追加する子要素をソース1の②で「ChildTag」という変数名で作成している。

ParentTagとChildTagは、どちらもタグ要素だが、ソース1の①と②では、作成方法が異なる。それは、①がオブジェクト型として定義されており、②が文字型として定義されている点である。

理由は、ParentTagがHTMLに記述された状態で考えると分かりやすい。

HTML上にParentTagを追加すると、HTMLは<DIV></DIV>と記述される。この状態では、タグの中の記述がないので、画面上には何も表示されない。HTMLを記述する際は、その<DIV></DIV>の中に、子となるタグ要素を記述して画面を作成していく。

これは、JavaScriptでinnerHTMLというプロパティを用いることでも可能である。innerHTMLは、タグ要素の中のHTML記述という意味のため、文字型を対象としている。つまり、<DIV></DIV>の意味を持つParentTagの中のHTML記述として、ChildTagをinnerHTMLで追加するため、ChildTagは文字型である必要があるというわけである。この処理は、ソース1の③で行っている。

また、innerHTMLは文字型として親要素の中に記述するため、追加されるタグ要素の数に制限はなく、タグ要素のスタイルシートなども反映できるという特徴も持っている。

3-3. HTMLへタグ要素を追加

続いて、親要素として定義されたタグ要素を、HTMLに追加する。

タグ要素をHTMLへ追加するには、ソース1の④にあるdocument.body.appendChild（オブジェクト型）というメソッドを用いる。これは、タグ要素を、HTML記述内で最上位の階層にあたる<BODY>タグの子として追加するという意味である。引数がオブジェクト型となっているのは、JavaScript上でタグ要素を作成すると、オブジェクト型として定義されるからである。

つまり、ChildTagのような文字型変数として作成した変数は、引数として渡すことができない。この引数に、オブジェクト型として作成されたタグ要素のParentTagを指定することで、HTMLにタグ要素を追加できるようにしているのである。

以上の手順で、HTML上へのタグ要素の追加が完了し、画面上にポップアップを表示可能になった。

なお、ソース1では、HTMLにタグを追加する処理の他に、ParentTag.style.positionなどの記述がある。これは、HTMLに追加するタグ要素のスタイルシートの設定をするための処理である。

3-4. HTMLからJavaScriptを呼び出す

ここから、ソース1の関数をHTML上で呼び出す方法を解説する。

●イベントハンドラ

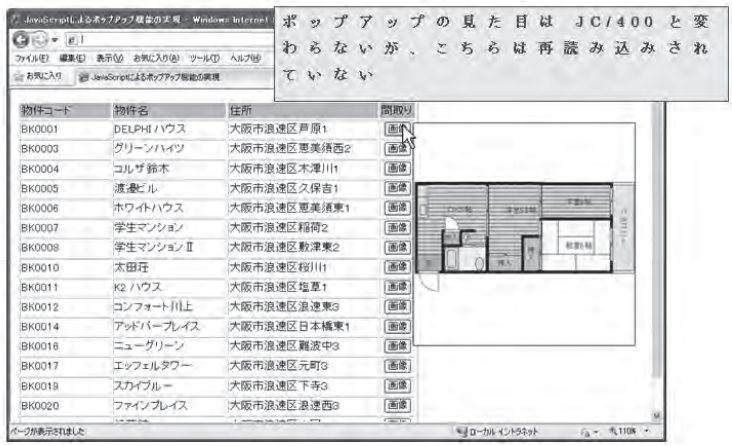
HTML上でJavaScriptを使用するには、「イベントハンドラ」を使用する。

イベントハンドラは、画面上で発生するさまざまなアクションに対して、JavaScriptの処理を行うためのトリガーとなる命令である。これには、マウスをクリックした時に発生するonClickや、マウスカーソルが上に載った時に発生するonMouseOverなど、さまざまな種類が用意されている。

イベントハンドラは、ソース3のようにHTMLのタグ内に記述する。ソース3では、ボタンをクリックすると、「ボタンがクリックされました。」というメッセージダイアログが表示される。【ソース3】

また、ソース3のonClickの””で囲まれた中に、JavaScriptの処理を記述する。ここの処理には、簡単な処理以外は、関数を作成して呼び出すことが多い。理由は、イベントハンドラ内の処理はそこでしか使用できないため、別の場所で同じ処理をしたい場合に再び記述する必

図4

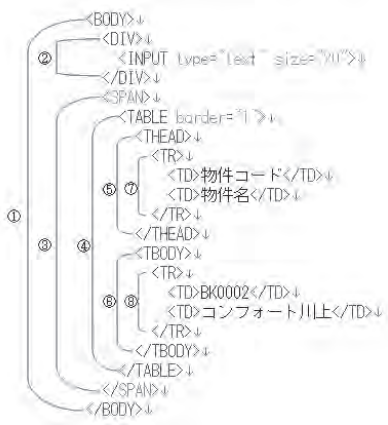


ソース1

```
// ポップアップ画面の親となるタグ要素↓
var ParentTag = document.createElement("DIV");↓ ①
↓
// ポップアップ表示するHTMLを記述↓
var ChildTag = '';↓
ChildTag += '<TABLE cellpadding="0" cellspacing="0">';↓
ChildTag += '<TBODY>';↓
ChildTag += '<TR>';↓
ChildTag += '<TD><IMG Style="Width: 160px; Height: 120px;"></TD>';↓ ②
ChildTag += '</TR>';↓
ChildTag += '</TBODY>';↓
ChildTag += '</TABLE>';↓
↓
// 画像ファイルのディレクトパス↓
var Path = 'http://Treport/img/';↓
↓
function PopCreate(iLeft, iTop, sImgname) {↓
  // 初期表示時処理↓
  if (ParentTag.innerHTML == '') {↓
    // ポップアップ画面のHTMLをセット↓
    ParentTag.innerHTML = ChildTag;↓ ③
    // ポップアップ画面共通のスタイルシートを設定↓
    ParentTag.style.position = 'absolute';          // 位置を絶対座標指定とする↓
    ParentTag.style.zIndex = 0x7FFFFFFF;            // 常に最前面に表示↓
    ParentTag.style.background = 'white';           // 背景色↓
    ParentTag.style.border = 'solid 1px black';     // 枠↓
    // BODYの子要素としてタグ要素を追加↓
    document.body.appendChild(ParentTag);↓ ④
  }↓
  ↓
  // ポップアップが非表示の場合の処理↓
  if (ParentTag.style.display != 'block') { ↓
    // ポップアップ画面個別のスタイルシートを設定↓
    ParentTag.style.left   = iLeft;    // 左位置↓
    ParentTag.style.top    = iTop;     // 上位置↓
    // ポップアップ画面を表示する↓
    ParentTag.style.display = 'block';↓
  }↓
  // ポップアップが表示されている場合の処理↓
  else {↓
    // ポップアップ画面を非表示にする↓
    ParentTag.style.display = 'none';↓
  }↓
  // IMGタグにサムネイル画像を適用する↓
  ParentTag.getElementsByTagName('IMG')[0].src = Path + sImgname;↓
}↓
```

HTMLから呼び出す関数

ソース2



要があり、保守性が悪くなるからである。

● PopCreate の呼び出し

ソース 4 では onClick を用いて、ソース 1 に記述された関数 PopCreate を呼び出している。PopCreate には引数として、以下の 3 つが用意されている。【ソース 4】

・iLeft（タグの X 座標）
・iTop（タグの Y 座標）
・sImgname（サムネイル画像の名称）

iLeft と iTop は、ポップアップの表示位置を指定する。ここで設定した値が、ソース 1 内のスタイルシートの設定に反映される。なお、表示位置は固定値で入れても問題はないが、明細の各行にボタンを設定する場合、固定値では常に同じ場所にポップアップが表示されることとなる。

今回は位置の決め方の例として、ソース 5 を用意した。この処理では、引数として、対象のタグ要素を渡し（this は、イベントが発生したタグ要素をオブジェクトとして指定している）、タグ要素の座標位置と画面のスクロール幅を取得することで、タグの座標位置を計算している。【ソース 5】

sImgname は、表示するサムネイル画像のパスを指定する。sImgname もまた、固定値として記入してもよいが、そうすると、明細で表示した場合に、各データに対応する画像を表示することができなくなる。

対処方法として、隠しフィールドを用意し、そこへ画面読み込み時に RPG で画像名をセットすることで、各データに対応した画像表示が実現可能になる。

以上が、JavaScript でポップアップを表示する方法となる。

4.JavaScriptでの活用例

JavaScript で作成するポップアップは、DB 情報を取得しないため、画面の再読み込みが発生せずシームレスに表示できることが特徴である。したがって、ポップアップを HTML 上の入力補助機能として活用することで、システムの運用効率を上げることも可能である。

その 1 例として、日付入力の補助として、カレンダーをポップアップ表示する方法を解説する。【図 5】

●カレンダーのポップアップ

ソース 6 では、ボタンの onClick で、Calendar という関数を呼び出している。【ソース 6】

この関数が、カレンダーのポップアップを呼び出す処理を行っている。Calendar の引数は、「タグの X 座標 , タグの Y 座標 , 日付を返すタグ要素」となっている。最初の 2 つの引数は、ポップアップを表示する位置を指定しており、ソース 4 と同じである。

3 つ目の引数が、ポップアップしたカレンダーで選んだ日付を表示するタグ要素を指定している。document.getElementById とは、HTML から指定した ID を持つタグ要素を探し、それをオブジェクト型として取得することができるメソッドである。

「Calendar」関数の中では、カレンダーの表示を行っている。ソース 1 で変更が必要な記述は②の ChildTag だけであり、ここにカレンダーを表示する HTML を記述すればよい。

なお、カレンダーを表示する HTML の作成については、月の移動など複雑なコーディングが必要なこともあり、ここでは説明を割愛させていただく。カレンダー作成のノウハウは、インターネットや書籍に豊富に存在しており、それらを参考にするのもよいと思われる。

JavaScript によるポップアップ表示の手法は、さまざまな場面で活用できる。例として挙げたカレンダー以外にも、システム開発で役立ててほしい。

補足:Ajaxを活用してHTMLを取得する

ソース 1 では、追加するタグ要素を JavaScript ソース内にそのまま記述した。今回は簡易な画面だったが、ポップアップ表示する HTML ソースの量が多い場合、ソースの量が膨大になり、保守性が悪くなる。

そこで、この補足の項では Ajax を活用し、外部記述された HTML をポップアップ表示する方法を解説する。

Ajax とは「Asynchronous JavaScript + XML」の略であり、一般的に、サーバと XML 形式のデータのやり取りを行うものである。ただし、必ずしもデータが XML 形式である必要はなく、TXT、CSV、HTML 形式等のデータも取得可能となっている。また、Ajax も JavaScript なので、新たに別の言語を使う必要がなく扱いやすい。

今回は、この Ajax の機能を用いて、外部記述された HTML ファイルを取得する方法を解説する。

補足1. Ajaxオブジェクトの作成

Ajax でサーバと通信を行うためには、まず専用のオブジェクトを作成する必要がある。【ソース 7】

ソース 7 の①では、専用のオブジェクトである ActiveXObject を作成している。サーバとの通信は、このオブジェクトを介して行う。ただし、このオブジェクトは、Microsoft が提供しているブラウザ Internet Explorer でのみサポートされているオブジェクトである。

それ以外のブラウザで使用する場合には、XMLHttpRequest というオブジェクトが用意されている。ソース 7 の②では、IE 以外のブラウザの場合に、そのオブジェクトを使用して作成するようになっている。

補足2. サーバとの通信

オブジェクトの作成が完了したら、次は実際にサーバと通信を行う。

ソース 8 の①では、データの取得に open メソッドを使用している。この引数に、「メソッド ,URL, 同期方式 , ユーザー名 , パスワード」を指定し（ユーザー名・パスワードは必要な場合のみ）、次の行で send メソッドを実行することでサーバに対してリクエストが実行される。【ソース 8】

●同期・非同期

上記の引数の中に、「同期方式」というものが表示されている。これは Ajax の特徴ともいえるものであり、以下に説明する。

Ajax の通信方式には、「同期」「非同期」の 2 種類が用意されている。同期型の通信の場合、次の処理はサーバからの応答

ソース3

```
<INPUT type="button" onClick="alert('ボタンがクリックされました。')">
```

ソース4

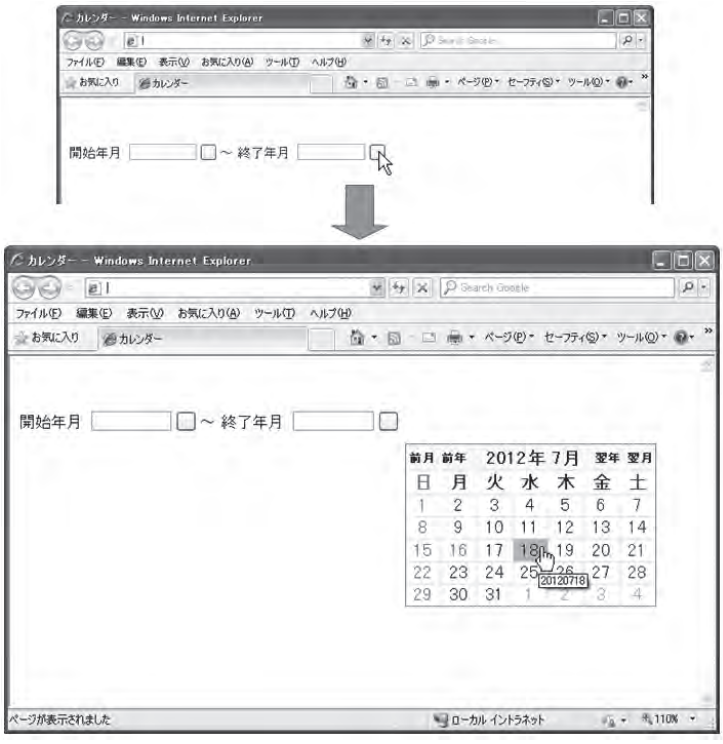
```
<TD><INPUT type="button" onClick="PopCreate(getElementPosition(this).right, getElementPosition(this).top,
this.parentNode.parentNode.childNodes[4].innerHTML );" value="画像"></TD>↓
<TD style="display: none;">IMAG01.jpg</TD>←
```

ソース5

ソース 5

```
function getElementPosition (obj){↓
  var html = document.documentElement;↓
  ↓
  //画面内座標を取得↓
  var rect = obj.getBoundingClientRect();↓
  var rectleft = rect.left - html.clientLeft;↓
  var recttop = rect.top - html.clientTop;↓
  ↓
  //スクロール幅を取得↓
  var scrollLeft = document.body.scrollLeft;↓
  var scrollTop  = document.body.scrollTop;↓
↓
  //ページ内での絶対座標を算出↓
  var left = rectleft + scrollLeft;↓
  var right = rectleft + scrollLeft+ obj.offsetWidth↓
  var top  = recttop + scrollTop;↓
  var bottom = recttop + scrollTop + obj.offsetHeight↓
  return {left:left, right:right, top:top, bottom:bottom};↓
}↓
```

図5



が返ってきてからとなる。一方、非同期型の通信は、サーバからの応答を待たず、そのまま次の処理が実行される。非同期型処理のメリットは、容量の重いデータを取り扱う場合にも、サーバからの応答が完了する前に次の処理を実行できるという点が挙げられる。

補足3. 非同期型通信について

同期型と違い、非同期型通信では、サーバからの応答状況を手動で管理する必要がある。

サーバからの応答状況を知るには、onreadystatechange というプロパティを使用する。このプロパティで設定した処理は、サーバからの応答状況が変化するたびに実行される。また、サーバからの応答状況を知るには、readyState と status というプロパティが用意されており、このプロパティ値を onreadystatechange によって実行される処理で調べることで、データの取得が完了したタイミングで次の処理を実行できる。これらの処理を行っているのが、ソース 8 の②である。【ソース 8】

サーバより返ったデータは、responseText で文字型として取得できる。これをソース 1 の②で記述しているタグ要素の代わりに、ChildTag へ代入すれば、取得した外部 HTML ファイルを、ポップアップとして表示することができる。

5. まとめ

ポップアップは、さまざまな場面において活用することが可能である。JC/400 の標準機能を用いて、マスタ検索などを実現する方法と、JavaScript を用いて、DB との連携を必要としない簡易な画面を実現する方法の 2 種類を紹介させていただいた。

この 2 つを場面によって上手に使い分けることで、Web システムで実現できる機能の幅は大きく広がっていくだろう。Web 画面と聞くとブラウザによる制約が大きいというイメージを抱かれる方が多いかもしれないが、JavaScript をうまく活用することができれば、思った以上に柔軟にさまざまなことを Web で実現可能にできる。

昨今は、HTML5 の登場により、Web での可能性がさらに広がっている。本稿を第一歩として、成長を続ける Web の世界へ飛び込んでいただければ嬉しい。

M

ソース6

```
年月<INPUT type="text" id="BKDTF"><INPUT type="button" onClick="Calendar(getElementPosition(this).right,
getElementPosition(this).top, document.getElementById('BKDTF') );">
```

ソース7

```
/*********************************************************************************
 目的 : 非同期通信用オブジェクトの作成
 引数 : なし
 戻値 : なし
*********************************************************************************/
function createXMLHttpRequest() {
  // IEの場合、ActiveXObjectを使用
  if (window.ActiveXObject) {
    try {
      // MSXML2.XMLHTTPが使える場合は、使用する(Microsoft.XMLHTTPよりも高速)
      return XMLHttpObject = new ActiveXObject("Msxml2.XMLHTTP");
    }
    catch(e) {
      try {
        // MSXML2.XMLHTTPが使えない場合は、Microsoft.XMLHTTPを使用する
        return XMLHttpObject = new ActiveXObject("Microsoft.XMLHTTP");
      }
      catch(e) {
        return null;
      }
    }
  // IE以外の場合、XMLHttpRequestを使用
  }
  else if(window.XMLHttpRequest) {
    return XMLHttpObject = new XMLHttpRequest();
  }
  else {
    return null;
  }
}
```

①

②

ソース8

```
var Result = '';          // 取得データ受け渡し用変数

/*********************************************************************************
 目的 : HTMLより実行する処理
 引数 : path : HTMLファイルのパス
 戻値 : なし
*********************************************************************************/
function loadData(path) {
  // XMLオブジェクト作成
  var httpObj = createXMLHttpRequest();

  // 非同期通信のレスポンス処理 readyState=4:完了 status=200or201:正常終了
  var handleResult = function() {
    // 取得が完了したら、データを返す
    if (httpObj.readyState == 4 && (httpObj.status == 200 || httpObj.status == 201) ) {
      // 文字化け防止
      Result = httpObj.responseText;
    }
  };

  // readyStateプロパティの値が変化する度にレスポンス処理を実行
  if (httpObj) httpObj.onreadystatechange = handleResult;
  // データの取得処理
  if (httpObj) {
    // サーバーへ送信するメソッドを指定
    httpObj.open("GET", path, true);
    // サーバーへリクエスト送信
    httpObj.send(null);
  }
}
```

【ソース7】を呼び出している

②

①