清水 孝将

株式会社ミガロ.

システム事業部 システム1課

JC/400でJavaScriptを活用

Webサーバーや IBM i とアクセスすることなく、動的な画面機能を実装しよう。
JC/400を用いて、Webアプリケーションへの JavaScriptの実装方法と活用例を紹介。

略歴
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4年目で Delphi/400 および 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。

1.はじめに

JC/400 の画面は、HTML だけで作成することができる。しかし、HTML 自体は静的なコンテンツなため、エラー項目の色の反転や項目のチェックなどの動的な Web 画面を作成しようとしても、HTML だけでは実現は不可能である。

JC/400 の機能を利用して、項目がブランクか否かをチェックすることも可能であるが、チェックのたびに、Web サーバーや IBM i とアクセスすることとなり、レスポンスが心配されるケースもあるだろう。

そこで JavaScript の出番である。JavaScript を活用することで、簡単な入力チェックや背景の反転など、ダイナミックな動作が実現できる。Web サーバー、IBM i とアクセスすることなく、クライアントサイドだけで項目をチェックすることも可能なのである。

今回は、JC/400 の Web アプリケーションへの JavaScript の実装方法と活用例を紹介したい。

2.JavaScriptの参照方法

JavaScript は、外部ファイルに記述する方法と、HTML 内に記述する方法の2つが存在する。

●外部ファイルに記述する方法

複数の HTML 文書から参照する関数は、外部ファイルに記述することで、各 HTML 内に記述せずに利用することが可能である。

アプリケーション内で共通化が可能な関数は、外部ファイルに記述しておくとよいだろう。【ソース 1】

● HTML 内に記述する方法

HTML 内に記述する場合は、同文書内のみで関数を使用することができる。内部的な変数を定義するのに適しているので、画面固有の処理は HTML 内に記述するとよいだろう。【ソース 2】

今回使用する JavaScript は、すべて HTML 内に記述する方法で進めていきたい。JavaScript は HEAD 要素、BODY 要素の任意の場所に記述可能であるが、通常は HEAD 要素内に記述することが主流である。【ソース 3】

3.入力項目のブランクチェックの実装

ここからは、実際に JavaScript のコーディングを解説していきたい。

今回は、入力項目のブランクチェックを例として紹介する。図1のように、項目 A と項目 B が必須入力の画面を設計した。【図 1】

要素と属性の概念

最初に、簡単ではあるが HTML や JavaScript における「要素」という用語の定義について説明しておきたい。要素とは、HTML 内に使われている部品の1つ1つのことを指している。入力項目やボタン、コンボボックス、画像のそれぞれが1つの要素なのである。

図1の画面の要素をそれぞれ記述したものが、ソース4である。【ソース4】

＜INPUT＞タグや＜A＞タグ、＜IMG＞タグが画面の部品、いわゆる要素となる。ここでの項目Aと項目Bは、＜INPUT＞要素となっている。

また、要素の1つ1つはそれぞれ「属性」を持っており、実際にWebブラウザに表現される長さや形、内容は属性によって決定する。後述のID属性はその1つである。＜INPUT＞要素は他に、type属性やvalue属性も持っている。

type属性は、＜INPUT＞の種類を定義する。例えば、今回の“text”という設定値であれば単1行入力項目となり、“button”と設定すればボタンの形となる。

value属性は、type属性がtextの場合、入力値を保持している。このvalueを利用して、値のチェックを行うのである。

要素の取得

まずは最初に、チェック対象とする項目の要素を取得する必要がある。

要素を取得する方法は何通りかあるが、今回は、JC/400とHTMLとの連携に利用されるID属性から要素を取得する、getElementById（要素のID）メソッドを使用する。

項目Aには“INPTA”、項目Bには“INPTB”とそれぞれにIDを割り振っている。

ソース5は、関数initpage（）という処理の中で、JavaScript内の変数“ObjINPTA”“ObjINPTB”に対して、それぞれ項目Aと項目Bの要素を格納する記述である。予め変数に代入しておくことで以後の処理に活用できる。【ソース5】

関数の名称は任意で定義できるが、initpage（）という名称には大きな意味がある。JC/400ではWebページを生成するとき、JavaScript内にinitpage（）という名称の関数を定義しておくと、自動的にBODY要素のOnloadイベント時に処理を行う仕組みになっている。（イベントの説明は後述する）

これで、画面が表示されたタイミングで、ObjINPTAとObjINPTBという変数内に項目Aと項目Bが格納されている状態になるのである。

エラーチェック関数の実装

エラーチェック関数の実装についての処理は、すべてinitpage（）関数で取得した変数ObjINPTAとObjINPTBの要素を利用する処理となり、それらを記述する。

ソース6は、関数CheckBlank（）を定義している。この中でObjINPTAとObjINPTBのValue属性の値をチェックし、ブランクの場合は警告を表示後、処理を中断するように記述している。【ソース6】

あとは、これを実行ボタンのクリック時に呼び出す設定を行うだけである。

イベントハンドラへの設定

イベントハンドラとは、ユーザーが行った動作や操作に対して、特定の処理を与えるためのトリガーとなる命令である。

要素にはそれぞれ、JavaScriptのイベントハンドラが準備されている。例えばonclickは文字通り、ボタンや画像がクリックされたときに実行される。

使用可能なイベントハンドラは要素によってさまざまであり、今回は画像リンクを設定している＜A＞要素のイベントハンドラを使用する。なお、＜A＞要素にもonclickのイベントハンドラは存在するが、onclickイベントはJC/400が先にハンドリングしてしまう。よって、onclickの前に動作するイベントとして、onmousedown使用することにする。

ソース7は、onmousedownイベント時に、前述したCheckBlank（）が実行されるように設定している状態である。これにより、実行ボタンがマウスでクリックされたタイミングで、エラーチェックの処理が走る仕組みが完成した。【ソース7】

図2は、項目Aに値が入力されていない場合に、実行ボタンをクリックしたときのイメージである。【図2】

項目Aに値を入力した場合は、図3のようになる。【図3】

4.エンターキーの制御

JC/400アプリケーションではエンター押下時に、画面がSubmitされる。エンター押下では、onmousedownのイベントは実行されないためエラーチェックも実行されない。

RPG側の制御で、何も処理をせずに画面をリフレッシュするだけの制御にすることも可能だが、ユーザーによっては更新してしまったと思われる方もいるであろう。

以降に、エンターキーを無効にする制御を紹介したい。

イベントキーコードの変換

ブラウザでは、キーボードの押下やマウス押下のタイミングで、イベントキーコードを取得する仕組みとなっている。（今回は、JC/400がサポート対象としているInternet Explorerのバージョン8を前提として話を進める）

Internet Explorer 8では、エンターキー押下時に“13”のイベントキーコードを取得する。このイベントキーコードを内部的に置き換えることで、エンターキーを無効化する。

ソース8は、取得したイベントキーコードが“13”の場合、イベントキーコードを“99999”に置き換える処理である。ちなみに“99999”というイベントキーコードのキーは実際に存在しないため、結果何も起こらないという仕組みとなる。【ソース8】

ソース9のように、＜BODY＞要素のイベントハンドラに設定する。これにより、フォーム上で発生するonkeydownイベントに対して、この関数が実行される。【ソース9】

なお余談ではあるが、タブキーのイベントキーコードは“9”なので、この応用で簡単にエンター押下時の項目移動を実現することも可能である。

5.エラー項目の色の反転

ここまでの処理で、必須項目のブランクチェックが実現できた。

次は、エラー項目をより視覚的に分かりやすくするために、項目の背景色を反転させたいと思う。

前述のCheckBlank（）処理に、項目の色を反転させる処理を追記した。ObjINPTA.style.backgroundColorに、色の値をセットすることで実現してい

ソース1

```
<script type="text/javascript" src="/jaci400/js/JCTest.js">↓
</script>↓
```

ソース2

```
<script language="JavaScript">↓
<!--↓
//ここにJavaScriptを記述↓
↓
-->↓
</script>↓
↓
```

ソース3

```
<HEAD>↓
<script type="text/javascript" src="/jaci400/js/JCTest.js">↓
</script>↓
<script language="JavaScript">↓
<!--↓
//ここにJavaScriptを記述↓
-->↓
</script>↓
</HEAD>↓
```

図1

ソース4

```
<BODY onkeydown="return PreventEnterPost(event);">↓
<FORM method ="post">↓
項目A <INPUT type ="text" value = "" id  = 'INPTA' >※必須入力<BR>↓
<BR>↓
項目B <INPUT type ="text" value = "" id  = 'INPTB' >※必須入力<BR>↓
<BR>↓
項目C <INPUT type ="text" value = "" id  = 'INPTC' ><BR>↓
<BR>↓
項目D <INPUT type ="text" value = "" id  = 'INPTD' ><BR>↓
<BR>↓
<A href="#" onmousedown ="CheckBlank();" ><img src="/jaci400/images/button2.gif" id="BTUNA" border="0"></A>
<BR>↓
</FORM>↓
</BODY>↓
```

る。これは、スタイルの値を動的に置き換えている処理である。【ソース 10】

図 4 のように、エラーとなった項目が赤色に反転された状態となる。【図 4】

6.最後に

今回紹介した項目のブランクチェックは、おそらくほぼすべてのシステムで活用できるのではないかと思い選んだ。

JC/400 を導入いただいている皆様の中には、新たに HTML を学習された方も多いのではないだろうか。そのような中、JavaScript まではなかなか手が出せないという言葉もよく聞く。

しかし、JavaScript を活用することでより、よいシステムを実現することが可能なこともまた事実である。また JavaScript は、HTML と同様に書籍やインターネット上にノウハウやリファレンス情報が豊富に存在する。何か実装したい処理をキーワードでインターネット検索をしてみると、簡単に情報が見つかったりする。一見馴染みなくとっつきにくいと思っていても、他のプログラム言語に比べても学習しやすいものではないかと思う。

本稿が JavaScript の活用のきっかけになれば幸いである。

M

ソース5

```
var ObjINPTA = null;↓
var ObjINPTB = null;↓
↓
function initpage(){ ↓
 ObjINPTA = document.getElementById("INPTA"); ↓
 ObjINPTB = document.getElementById("INPTB"); ↓
}↓
```

ソース6

```
function CheckBlank(){↓
   if (ObjINPTA.value == "") {↓
   alert('項目Aは必須入力です。');↓
   return false;↓
   }  ↓
   if (ObjINPTB.value == '') {↓
   alert('項目Bは必須入力です。');↓
   return false;↓
   }↓
}↓
```

ソース7

```
<A href="#" onmousedown ="CheckBlank();" ><img src="/jaci400/images/button2.gif"
```

図2

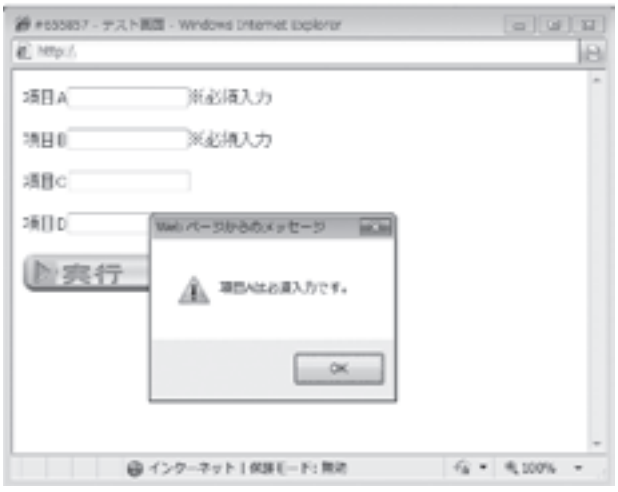
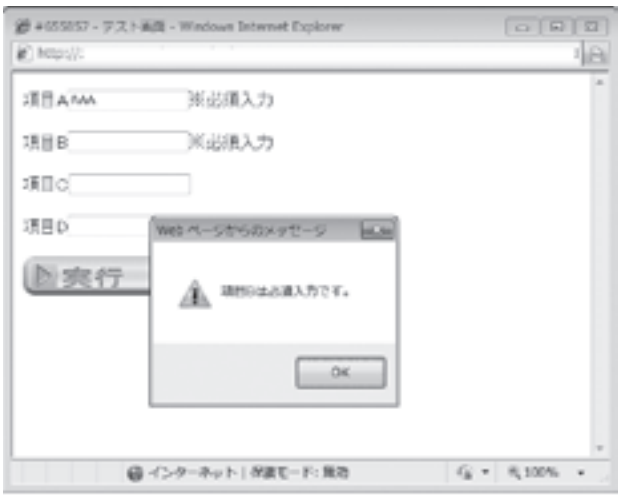


図3



ソース8

```
function PreventEnterPost(e){ ↓
     var e = window.event; ↓
     if(e.keyCode == 13){↓
    window.event.keyCode = 99999;↓
  } ↓
}↓
```

ソース9

```
<BODY onkeydown="return PreventEnterPost(event);">↓
```

ソース10

```
function CheckBlank(){↓
   ObjINPTA.style.backgroundColor =  "white" ;↓
   ObjINPTB.style.backgroundColor =  "white" ;↓
   if (ObjINPTA.value == "") {↓
   alert('項目Aは必須入力です。');↓
   ObjINPTA.style.backgroundColor = "red" ;↓
   return false;↓
   }  ↓
   if (ObjINPTB.value == '') {↓
   alert('項目Bは必須入力です。');↓
   ObjINPTB.style.backgroundColor = "red" ;↓
    return false;↓
   }↓
}↓
```

図4