國元 祐二

株式会社ミガロ.

RAD事業部 技術支援課



[SmartPad4i]

# SmartPad4iのインターフェース機能拡張

**略歴**
1979年3月27日生まれ
2002年3月 追手門学院大学 文学部アジア文化学科卒業
2010年10月 株式会社ミガロ.入社
2010年10月 RAD事業部配属

**現在の仕事内容**
SmartPad4i（JC/400）、Business4Mobile、Valenceの製品試験やサポート業務、導入支援などを行っている。

1. はじめに
2. JavaScriptについて
2-1. JavaScriptとは
2-2. jQueryとは
2-3. jQueryの入手
2-3. jQueryの設定
3. インターフェースの機能拡張
3-1. 入力制限機能付きの入力欄
3-1-1. 入力欄の機能拡張
3-1-2. 実装方法
3-2. 入力可能なコンボボックス
3-2-1. コンボボックスの機能拡張
3-2-2. 実装方法
3-3. 選択リストが絞り込み可能なコンボボックス
3-3-1. 絞り込み可能なコンボボックス
3-3-2. 実装方法
4. まとめ

## 1. はじめに

SmartPad4i はIBM iのモダナイゼーションツールである。既存のRPG／COBOLの資産や知識を活用して、Webやモバイルアプリケーションが作成できる。

IBM iのグリーン画面で作成されたアプリケーションでは、DDS表示装置ファイルで画面を定義するが、SmartPad4iではHTMLで画面を定義する。画面への入出力処理を変更することで、既存の業務ロジックを活用しながら、簡単にSmartPad4iアプリケーションへ置き換えられる。

画面定義のHTMLは、最新のHTML、JavaScript、CSSを活用することで自由なインターフェースが構築できるのも魅力である。80桁×24行の制限がある、CUIのグリーン画面では実現できないインターフェースが作成可能である。

SmartPad4iのアプリケーションは、簡単なHTMLとRPG／COBOLの知識があれば作成できる。SmartPad4iの標準機能や、簡易なHTMLで作成できない機能が必要な場合には、JavaScriptを使用して機能拡張に対応する。

本稿では、サポートに問い合わせのあった内容の中から、SmartPad4iをご利用いただいているお客様にぜひ活用していただきたいと感じた、SmartPad4iの標準機能だけでは作成できないインターフェース機能拡張方法の手順を説明する。

## 2. JavaScriptについて

### 2-1.JavaScriptとは

JavaScriptは1995年、アメリカのプログラマー、Brendan Eich（ブレンダン・アイク）氏によって設計されたプログラミング言語だ。当初はHTMLへ動きを与えるために開発された言語である。現在では、Webブラウザ上で動作するだけでなく、サーバーサイドで動作する実行環境もあり、Web開発の領域で広く活用されている。

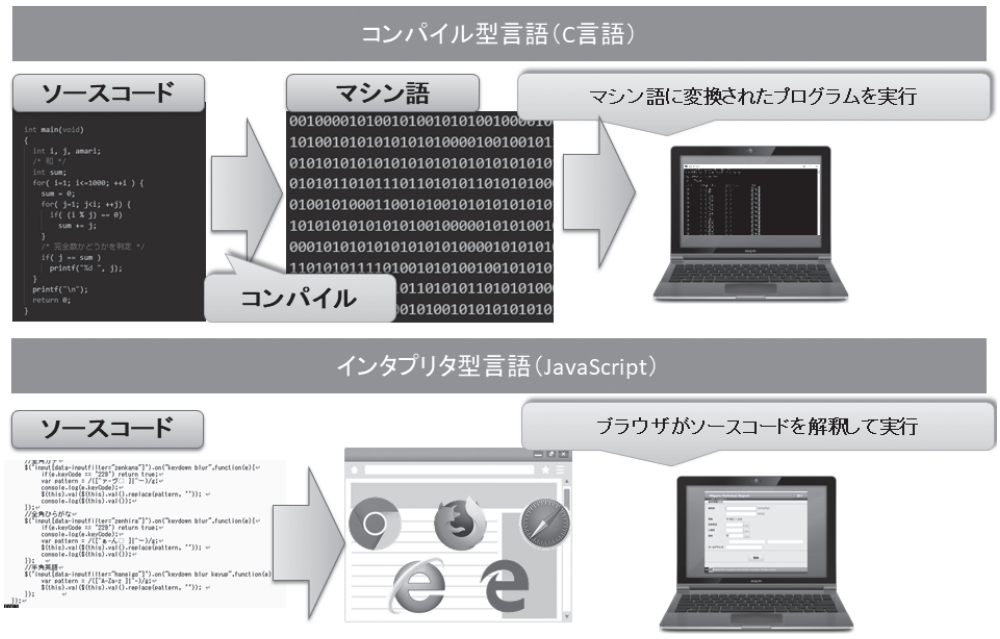
SmartPad4iでも、機能を追加する場合にはJavaScriptを使用する。JavaScriptはプログラミング言語であるが、難しく考える必要はない。JavaScriptは比較的簡単な言語で、初めてプログラミングをする初心者であっても、さまざまなことが実現可能となる。

JavaScriptが比較的簡単な言語とされている理由は2つある。1つは、変数にあらゆる型の値を代入できる点である。JavaScriptには型はあるが、型の制約は弱く、変数にどのような値も設定可能なので、プログラミング時に型を強く意識する必要がない。

もう1つは、プログラム環境を構築するための敷居が低い点である。JavaScriptのプログラミングには特別な環境は必要なく、PCにインストールされているブラウザで動作を確認できる。そのため、JavaScriptは初めてのプログラム作成に適している。

またC言語やRPG／COBOLなどは、コンパイル（プログラム言語で記述されたソースコードをコンピュータ上で実行

図1　JavaScriptはインタプリタ型言語




ソース1

**HTMLに<script>タグを追加して処理する方法**

```
<script>
    //JavaScriptの処理を記述する
</script>
```

**外部ファイルとしてJavaScriptを読込み処理する方法**

```
<script src = "/smartpad4i/html/js/sample.js"></script>
```

src属性にファイルパスを記述
/smartpad4i/html/js/sample.js
に配置した場合

図2　JavaScriptは</body>タグ前に記述

**headタグ内に記述**

```
<html>
<head>
<script>
    //複雑なJavaScriptの処理
</script>
</head>
<body>
   <!--  画面の要素表示 -->
</body>
</html>
```


**bodyタグの閉じタグ前に記述**

```
<html>
<head>
</head>
<body>
   <!--  画面の要素表示 -->
<script>
    //複雑なJavaScriptの処理
</script>
</body>
</html>
```

可能な形式に変換する処理）が必要である。しかしJavaScriptは、HTMLに記述したソースコードを即座にブラウザ上で実行できる。記述した言語をそのまま読み込み、動作するインタプリタ型言語であるからだ。たとえ誤ってソースコードを記述したとしても、書き直すだけで即座に動作へ反映される。【図1】

次は、JavaScriptの使い方について説明する。JavaScriptはHTMLにscriptタグを追加して処理する方法と、外部ファイルとしてJavaScriptを読み込み処理する方法の2種類ある。【ソース1】

scriptタグ自体はHTMLのどこに記述してもよいが、通常「headタグ内」/「bodyタグ内」/「bodyタグの閉じタグ前」に記述する。以前は、headタグ内に記述されることが多かった。しかし現在では、「bodyタグの閉じタグ前」に記述することが多い。これはブラウザがJavaScriptを処理する際、HTMLの表示処理を停止してしまうためだ。

つまり、「headタグ内」に記述したJavaScriptの処理で時間がかかる場合、JavaScriptの処理が完了するまで、画面は表示されない。そのため、ページを読み込ませる前にJavaScriptを読み込む必要がある場合には、「headタグ内」にscriptタグを記述し、それ以外の場合は画面の表示速度を考慮し、「bodyタグの閉じタグ前」にscriptタグを記述する。【図2】

JavaScriptは、scriptタグを記述するだけで使用できるので非常に簡単である。記述したscriptタグのプログラムはブラウザ上で動作するが、ブラウザの種類により、JavaScriptは方言のように少し違った記述が必要となる。

ブラウザごとに対応するJavaScriptを記述するのは骨が折れるので、記述の違いを吸収してくれるライブラリを使用すると便利である。ライブラリとして最も使用されているのが、次に解説するjQueryである。

## 2-2. jQueryとは

jQueryとは、米国のプログラマー、John Resig（ジョン・レッシグ）氏によって開発・公開されたJavaScript用のライブラリである。jQueryは著作権表示を消さなければ、商用・非商用を問わず、誰でも自由に利用できる。jQueryを使用すると、HTMLの要素取得やイベント処理、アニメーション、Ajax処理などを簡単に実装可能となる。

jQueryについては、「2015年 No.8 テクニカルレポート」にある『スマートデバイス開発で役立つ　画面拡張テクニック』でも取り上げたので、ご一読いただけると幸いだ。

## 2-3. jQueryの入手

jQueryは、SmartPad4iをインストールするとWebサーバー上に展開される。そのため、インターネット上からjQueryのライブラリをダウンロードしなくても使用できる。もちろん、SmartPad4iのインストーラーに含まれていない、最新のjQueryをインターネットからダウンロードして使用することも可能である。最新のjQueryは公式サイトからダウンロードできる。【図3】

jQuery 公式サイト
https://jquery.com

リンク先では、圧縮版（compressed, production_jQuery）をダウンロードする。jQueryのデバッグを行いたい場合には、非圧縮版（uncompressed, development_jQuery）をダウンロードすればよい。

## 2-4. jQueryの設定

jQueryを使用するための設定方法は簡単だ。HTML内にスクリプトタグを追加するだけで使用できる。スクリプトタグでは、src属性にjQueryファイルへのパスを設定するだけである。

また、jQueryはCDN（コンテンツデリバリネットワーク）からも読み込める。CDNはWebコンテンツをインターネット経由で配信するために最適化されたネットワークである。CDNを利用する場合のメリットは、サーバーにjQueryファイルを配置する必要がなくなる点と、CDN経由で直接ダウンロードすることにより、サーバーの負荷を減らせる点だ。デメリットは、クライアント端末がインターネット接続可能でなければならない点である。

jQueryのCDNは、「jQuery CDN」「Google Hosted Libraries」「Microsoft Ajax Content Delivery Network」などで公開されている。この中から、jQuery公式のCDNの使用方法について説明する。jQuery CDNの使用方法は、ブラウザで「jQuery CDN」のサイトにアクセス後、利用したいjQueryバージョンのリンクをクリックする。表示されたscriptタグをコピーしてHTMLに追加すればよい。【図4】

jQuery CDN
https://code.jquery.com/

なお本稿で紹介する機能は、SmartPad4iデフォルトで配置されるjQueryで実装している。【ソース2】

次に、jQueryを使用したSmartPad4iインターフェース機能拡張の実装方法を紹介する。

# 3. インターフェースの機能拡張

## 3-1.入力制限機能付きの入力欄

### 3-1-1. 入力欄の機能拡張

SmartPad4iではHTMLの作成後、SmartPad4i Designerというツールで入力欄の型や長さを設定すると、自動的にエラーチェックする入力欄が作成できる。

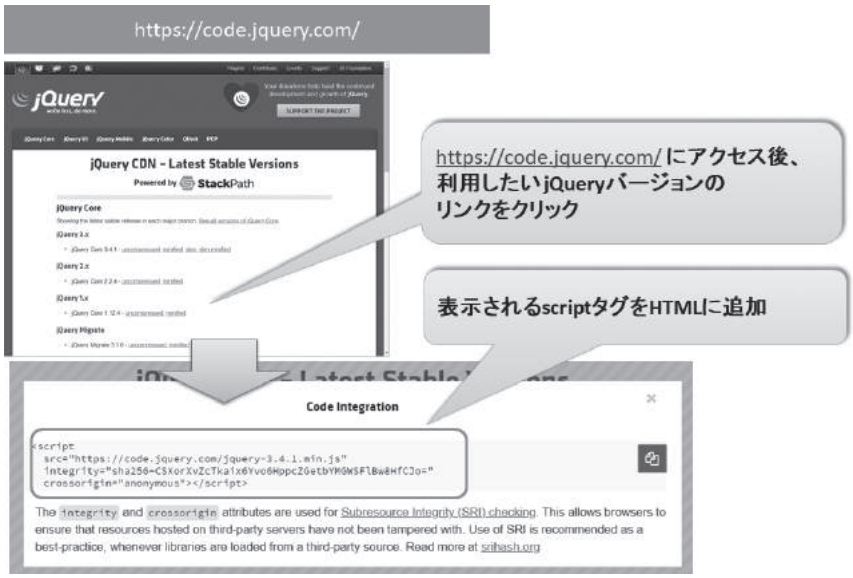
たとえば、IBM iのAタイプフィールドのように、シングルバイト文字のみ入力可能な入力欄や、Oタイプフィールドのようにシングルバイト文字とマルチバイト文字の両方が入力可能で、シフト文字を考慮した文字数チェックを行う入力欄が簡単に作成可能である。

それだけでも充分便利なチェック機能であるが、全角カナのみの入力を許可したい場合や、ひらがなだけを許可したい場合などは、SmartPad4iの通常機能では実現できない。機能を追加するには、JavaScript（jQuery）を使用する。

ここでは、入力欄の入力制限機能の作成方法を紹介する。入力欄の入力制限機能として、全角カナ以外の入力値はクリアする、IBM i TypeがOtypeの入力欄を例に挙げる。【図5】

図3　jQueryのダウンロード


図4　jQuery CDN




ソース2

**SmartPad4i デフォルトで配置されるjQueryの読込み**

```
<script src="/smartpad4i/APP_SP4i/AR/theme/jquery-1.10.2.min.js"></script>
```

**ダウンロードしたjQueryの読込み**

```
<script src="/smartpad4i/lib/jquery-3.4.1.min.js"></script>
```

/smartpad4i/lib/jquery-3.4.1.min.js に配置した場合

**CDNjQueryの読込み**

```
<script
  src="https://code.jquery.com/jquery-3.4.1.min.js"
  integrity="sha256-CSXorXvZcTkaix6Yvo6HppcZGetbYMGWSFlBw8HfCJo="
  crossorigin="anonymous">
</script>
```

### 3-1-2　実装方法

HTMLの入力欄定義には、HTML5のマークアップ仕様で追加されたカスタムデータ属性を使用する。カスタムデータ属性は、HTMLタグに独自の属性を追加できる機能で、すべてのHTMLタグに設定できる。

設定方法は、タグの属性に「data-」から始まり、ハイフン以後は、アルファベットで任意の名前を開発者が自由に定義できる。入力制限機能のため、カスタムデータ属性の名称を「data-inputfilter」とした。カスタムデータ属性の値には、「zenkana」を設定している。

また同種のカスタムデータ属性を、複数のタグに設定できるので、カスタムデータ属性data-inputfilter = "zenkana"が設定されているすべてのinputタグに、全角カナのみが入力できるフィルタを適用できる。

JavaScriptは外部ファイルにして読み込むため、外部ファイルfilter.jsの読み込み設定を＜/body＞前に追加した。＜/body＞前に追加した理由は、ページのレンダリング完了後にJavaScriptファイルを読み込むことで、画面の表示速度を向上させるためである。なおfilter.jsは、［DocumentRoot］/SmartPad4i/html/js/に配置している。【ソース3】

ここで使用しているjQuery/JavaScriptの関数・メソッドは、【図6】になる。入力制限の機能は、4つのメソッドの使用方法をおさえるだけで作成が可能である。【ソース4】

外部ファイルfilter.jsのスクリプトは、$（function（）で書き始めている。$（function（）｛｝）;の記述方法は、HTMLが読み込まれた後、function関数を実行する。つまりfunction（）の後に記述された、｛｝の処理が実行される。

全角カナのみ許可したい入力欄には、カスタムデータ属性にdata-inputfilter = "zenkana"を設定した。

まず、カスタムデータ属性が設定された要素を取得する必要がある。jQueryでは、要素の取得に$()関数を利用する。$（）関数ですべてのinputタグを取得したい場合には、$（'input'）と定義する。

inputタグの中からカスタムデータ属性が設定された要素を取得するには、[]を使用して、プロパティで絞り込む。つまり、$（'input［data-inputfilter = "zenkana"］'）の記述は、すべてのinputタグからdata-inputfilterカスタムデータ属性が設定されていて、かつ値がzenkanaの要素だけ取得するという意味になる。

取得した要素にイベントを設定するには、onメソッドを使用する。onメソッドでは、第1引数に対象となるイベントの文字列、第2引数に実行する関数を設定できる。第1引数のイベント文字列は、半角スペース区切りで複数設定できる。

注意が必要なのは、keyイベントのkeyCodeが229（IME入力中）の場合に、処理を抜けるよう記述している点である。キーボードが押下されると、keyイベントが発生し、keyCodeにはどのキーを入力したかの値が設定される。

しかしIME入力時には、どのキーが入力されてもkeyCodeに229が設定されるため、入力の確定が行われるまで、処理しないようにしている。

以降の処理は単純で、キー入力（keydownイベント）、またはフォーカスが外れた（blurイベント）場合に、入力された値を正規表現でチェックして文字置換するだけである。

正規表現とは、複数ある文字列の集合を1つの形式で表現する手法だ。正規表現を使用すれば、特定の文字列パターンが出現するかどうか、またどこに出現するかを特定できる。設定した/（[^ァ-ヴ　］|^ー）/gの正規表現は、全角カナと半角スペース、全角スペース以外の文字列を特定できる。

全角カナの場合は、小文字の“ァ”から“ヴ”以外の文字を特定するようにパターンを作成している。【図7】は文字コード順に、カタカナ、ひらがなを並べた表である。ひらがなの場合は、“ぁ”から“ん”ですべてを網羅できるが、カタカナの場合は、“ン”ではなく、“ヴ”までになる点に注意する。小文字の“ヵ”や“ヶ”に関しては、1ヵ所や1ヶ月などの使われる文字で、カタカナではないので含めていない。

パターンに一致した値、つまり全角カナと半角スペース、全角スペース以外の文字列を空文字（''）に置き換えることで、入力制限を実現している。jQueryを使用すると、少しの記述で入力制限できる入力欄が作成できる。

また正規表現のパターンを変えることで、さまざまな入力制限が作成できる。たとえば、全角ひらがなであれば/（[^ぁ-ん　］|^ー）/gの正規表現パターンで入力制限が可能である。【ソース5】

全角カナのみ許可したい入力欄には、inputタグのカスタムデータ属性にdata-inputfilter = "zenkana"、全角ひらがなのみ許可したい入力欄には、inputタグのカスタムデータ属性にdata-inputfilter = "zenhira"を設定する。

外部JavaScriptのfilter.jsを読み込み、HTMLのカスタムデータ属性を設定するだけで、機能が追加できるので非常に便利だ。

さらに半角文字の入力制限であれば、keyupイベントを追加することで、半角英字以外入力できない状態にもできる。さまざまな入力制限が作成できるので、ぜひ活用してほしい。【図8】

## 3-2.入力可能なコンボボックス

### 3-2-1 コンボボックスの機能拡張

HTMLのコンボボックス（selectタグ）は、リスト内から選択するだけで、任意の文字は入力できない。ここではリストから選択でき、入力欄としても利用可能なコンボボックスの機能拡張方法を紹介する。

具体例として、フォーカス移動するとIBM iファイルに登録されているJANコード情報をリスト表示しつつ、入力欄としても利用できる入力可能なコンボボックスを作成する。【図9】【図10】

### 3-2-2 実装方法

ここで追加している、jQuery/JavaScriptの関数・メソッドは、【図11】のとおりである。

入力可能なコンボボックスの機能は、3-1で使用したjQuery／JavaScriptのメソッドと、追加9つのメソッドの使用方法により作成が可能である。

リストの一覧からの選択＋文字入力の機能を実現するには、入力欄の要素とコンボボックスの要素を組み合わせて作成する必要がある。HTMLの定義では、入力欄（inputタグ）とコンボボックス（selectタグ）の2つを使用する。

またカスタムデータ属性を利用して、関連付けを行っている。入力欄（input

図5　入力画面例


ソース3

入力欄のHTML定義

```
<input data-inputfilter="zenkana" type="text" name="INNM2" id="INNM2"> (カナ)
```

HTML5のマークアップ仕様で追加された
カスタムデータ属性を使用

JavaScript 外部ファイルの定義

```
～　省略　～
</form>
</div>
<script src="/smartpad4i/html/js/filter.js"></script>
</body>
</html>
```

外部ファイルの読込み

図6　3-1で使用するjQuery、JavaScriptのメソッド

| 関数/メソッド | 機能 |
|---|---|
| $() | 要素の取得 |
| .on | 要素にイベント処理を追加 |
| .val | 要素のValue値を取得、設定 |
| .replace | 文字列の置換処理 |

タグ）には、カスタムデータ属性 data-customlist を、コンボボックス（select タグ）にはカスタムデータ属性 data-customlistc を定義している。

カスタムデータ属性の値で、入力欄とコンボボックスを関連付けるため、それぞれ値に“1”を設定した。さらに、コンボボックスは入力欄にフォーカスが移動した時に表示するため、css の属性 display:none を使用して、画面表示時は非表示に設定している。【ソース 6】【ソース 7】

入力欄にフォーカスが移動した場合の処理を追加するため、jQuery の on メソッドを使用した。最初に、カスタムデータ属性 data-customlistc が設定されている要素（コンボボックス）を hide メソッドで非表示にしている。

次に、data-customlistc が設定されている要素（コンボボックス）位置を css メソッドで横 0、縦 0 の位置に移動させた。“this”の記述は、イベント発生元の要素を指すため、フォーカスが移動した入力欄となる。

入力欄に設定されているカスタムデータ属性を取得するには、data メソッドを使用する。data メソッドで、入力欄のカスタムデータ属性、data-customlist の値を取得し、その値をもとに、filter メソッドで入力欄の下に表示させたいコンボボックスを絞り込んでいる。

またコンボボックスでリストから選択した値を、どの入力欄へ戻すかを設定するため、コンボボックスに data-returnkey というカスタムデータ属性を追加した。値には入力欄の id 属性を attr メソッドで取得し、設定している。リストとなるコンボボックスの位置が入力欄下になるよう調整後、コンボボックスを show メソッドで表示している。

【ソース 7】のプログラムは 10 行程度になったが、JavaScript のメソッドチェーンを使用すると 1 行で記述可能である。jQuery はとくにメソッドチェーンが効果的に使えるよう設計されているため、効率的にプログラムを作成できる。メソッドチェーンで記述したのが【ソース 8】である。【ソース 7】【ソース 8】は同じ処理になる。

【ソース 9】はコンボボックスが選択された場合や、別の入力欄にフォーカスが移動した場合の処理である。

カスタムデータ属性 data-customlist が、設定されていない入力欄にフォーカス移動した場合、コンボボックスを非表示にしている。また入力欄以外の要素がクリックされた場合にも、コンボボックスを非表示にしている。

リストから選択された場合には、選択値を入力欄に設定する。その入力欄を特定するため、入力欄にフォーカスが移動した際に、カスタムデータ属性 data-returnkey という属性をコンボボックスへ動的に設定している。

例では、data-returnkey ＝“INP1”が設定されるため、値を設定する入力欄（id 属性＝“INP1”）が特定できる。

これで入力欄にフォーカスが移動後、入力欄の下に関連するコンボボックスが表示され、リストから選択でき、入力欄としても利用可能なコンボボックスが完成した。

### 3-3. 選択リストが絞り込み可能なコンボボックス

#### 3-3-1　絞り込み可能なコンボボックス

自動入力補完機能のように、コンボボックスのリストを絞り込めば操作性がよくなる。3-2 で実装したコンボボックスをさらに便利に利用できるよう、入力欄に値を入力すると、リストが絞り込まれるように実装する方法を紹介する。【図 12】

#### 3-3-2　実装方法

ここで解説する jQuery ／ JavaScript の関数・メソッドは、【図 13】のとおりである。この機能は、これまでに使用した jQuery ／ JavaScript のメソッドと、追加 5 つのメソッドの使用方法により作成が可能である。

実装方法は、3-2 で実装した JavaScript に入力欄の keyup イベントを追加するだけである。【ソース 10】

$(function() {}); の中に、コンボボックスのリストをバックアップするための変数、clonecmb を宣言している。コンボボックスのリストは、入力欄へ入力された値をもとに絞り込む。絞り込みの際、一致しないリストを削除するため、初回に全リストを clone メソッドでバックアップしている。

そして、keyup と focus のイベント処理を記述した。イベントの中の処理は、【ソース 11】のとおりである。

jQuery では、$('［属性 ^ = " 値 "］') とセレクタを記述すると、属性値を前方一致で検索して対象の要素が取得できる。コンボボックスの全リストを入力された値の前方一致で検索して、not メソッドで一致しないリストを取得後、remove メソッドで削除している。

Chrome、FireFox、Safari、Microsoft Edge では、コンボボックスのリスト（option タグ）にスタイルシートを適用することで、リストを表示・非表示にできるが、Internet Explorer では残念ながら動作しない。そのため、コンボボックスのリストを直接削除する方法で実装した。

## 4. まとめ

本稿では、サポートに問い合わせのあった内容をレポートの題材とした。SmartPad4i は自由なインターフェースを作成できるモダナイゼーションツールであるが、jQuery を活用することでさらなる機能拡張も可能であることが実感できた。

実装が難しいと予想した問い合わせでも、JavaScript を使用してカスタマイズすることで、多くの機能は実現ができた。工夫次第で SmartPad4i アプリケーションは素晴らしいインターフェースを作成できるので、ぜひ本稿の内容を活用いただきたい。

M

ソース4

**Filter.jsの記述内容**

```
$(function(){
      //全角カナ
      $('input[data-inputfilter="zenkana"]').on("keydown blur",function(e){
            if(e.keyCode == '229') return true;
            var pattern = /([^ァ-ヴ　]|^ー)/g;
            $(this).val($(this).val().replace(pattern, ''));
      });
});
```

正規表現で入力制限

**正規表現の記述**

```
var pattern = /([^ァ-ヴ␣␣]|^ー)/g;
```

全角スペースと半角スペースを設定

図7　文字コード順について

ひらがな文字コード順

| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ぁ | あ | ぃ | い | ぅ | う | ぇ | え | ぉ | お | か | が | き | ぎ | く | ぐ | け | げ | こ | ご |
| さ | ざ | し | じ | す | ず | せ | ぜ | そ | ぞ | た | だ | ち | ぢ | っ | つ | づ | て | で | と |
| ど | な | に | ぬ | ね | の | は | ば | ぱ | ひ | び | ぴ | ふ | ぶ | ぷ | へ | べ | ぺ | ほ | ぼ |
| ぽ | ま | み | む | め | も | ゃ | や | ゅ | ゆ | ょ | よ | ら | り | る | れ | ろ | ゎ | わ | ゐ |
| ゑ | を | ん | | | | | | | | | | | | | | | | | |

カナ文字コード順

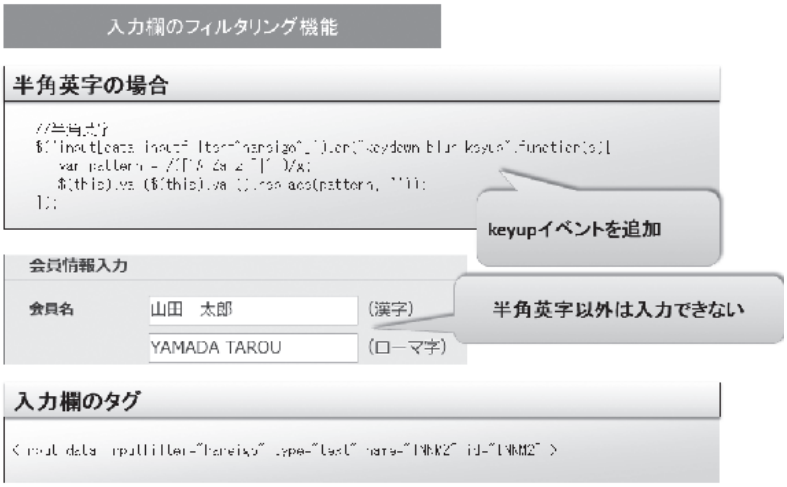
| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ァ | ア | ィ | イ | ゥ | ウ | ェ | エ | ォ | オ | カ | ガ | キ | ギ | ク | グ | ケ | ゲ | コ | ゴ |
| サ | ザ | シ | ジ | ス | ズ | セ | ゼ | ソ | ゾ | タ | ダ | チ | ヂ | ッ | ツ | ヅ | テ | デ | ト |
| ド | ナ | ニ | ヌ | ネ | ノ | ハ | バ | パ | ヒ | ビ | ピ | フ | ブ | プ | ヘ | ベ | ペ | ホ | ボ |
| ポ | マ | ミ | ム | メ | モ | ャ | ヤ | ュ | ユ | ョ | ヨ | ラ | リ | ル | レ | ロ | ヮ | ワ | ヰ |
| ヱ | ヲ | ン | ヴ | ヵ | ヶ | | | | | | | | | | | | | | |

ソース5

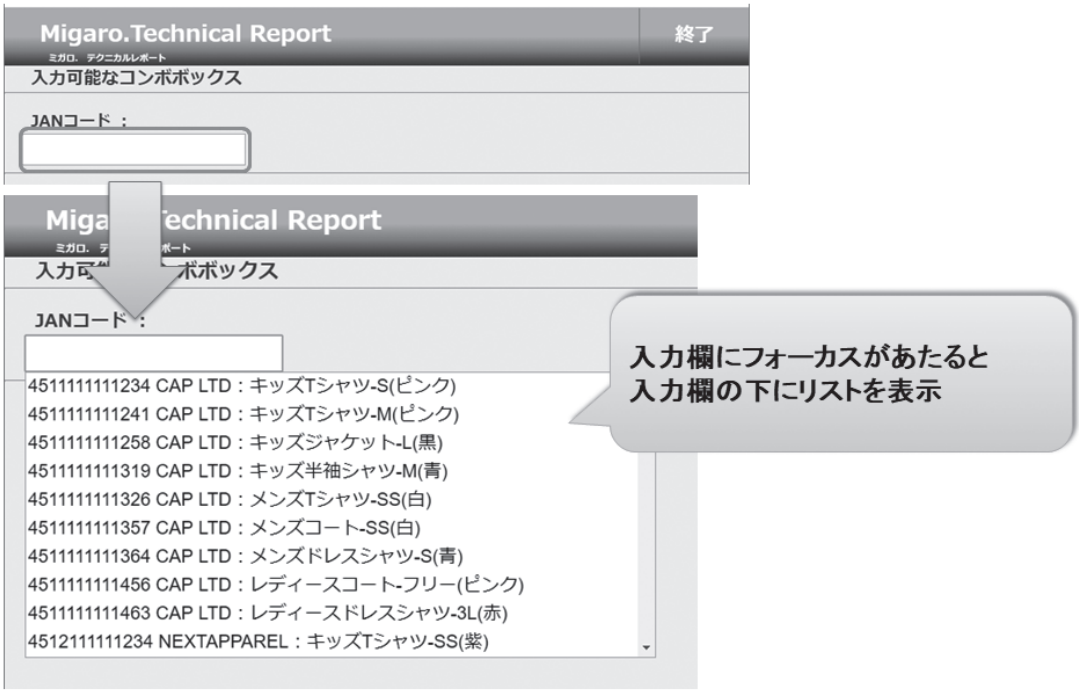
**Filter.jsの記述内容**

```
$(function(){
      ～省略～
      //全角ひらがな
      $('input[data-inputfilter="zenhira"]').on("keydown blur",function(e){
            if(e.keyCode == '229') return true;
            var pattern = /([^ぁ-ん　]|^ー)/g;
            $(this).val($(this).val().replace(pattern, ''));
      });
});
```

## 図8　入力画面例




## 図9　入力可能なコンボボックス①




## 図10　入力可能なコンボボックス②

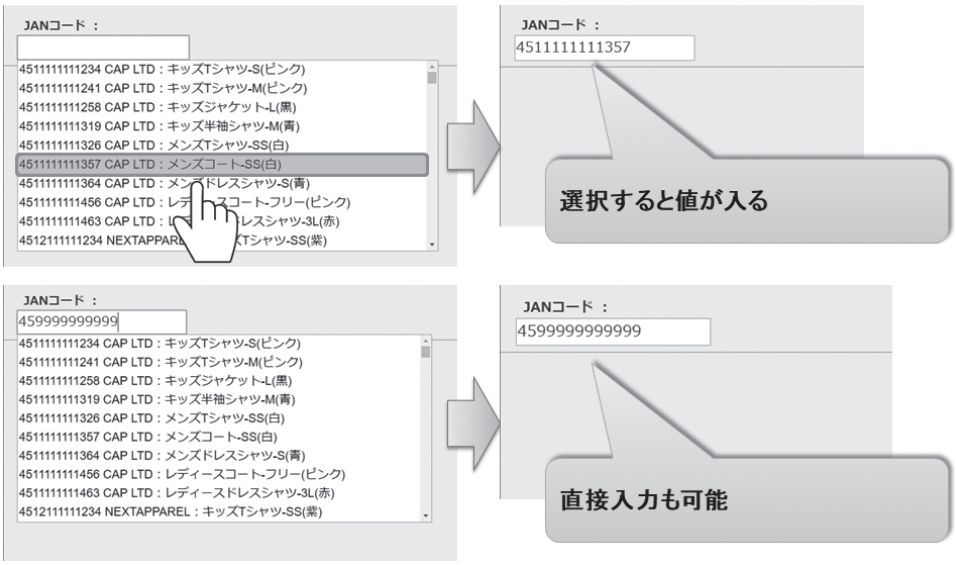


図11　3-2で使用するjQuery、JavaScriptのメソッド

| 関数/メソッド | 機能 |
|---|---|
| .hide | 要素の非表示 |
| .show | 要素の表示 |
| .css | css属性の取得、設定 |
| .filter | 要素の絞り込み |
| .data | 要素のカスタムデータ属性を取得、設定 |
| .attr | 要素の属性を取得、設定 |
| .offset | 要素の位置を取得、設定 |
| .outerHeight | 要素の高さを取得 |
| .find | 要素の子要素を取得 |

ソース6　入力欄とコンボボックスのHTML定義

```
<!-- 入力欄の定義 -->
<input type="text" name="INP1" id="INP1" style="margin-left:20px;position:relative"
    data-customlist="1">

<!--コンボボックスの定義 -->
<select id="CMB01" style="display:none;position:absolute;font-size:18px"
    data-customlistc="1" multiple size="10"></select>
```



ソース7　入力欄のイベント定義

```
<script>
  $(function(){
    $('[data-customList]').on({
      //入力欄にフォーカスが設定されると、関連するコンボボックスを表示
      'focus' : function(){
        //コンボボックスを非表示
        $('[data-customlistc]').hide();
        //コンボボックスの位置を初期化
        $('[data-customlistc]').css('left','0');
        $('[data-customlistc]').css('top','0');
        //入力欄のカスタムデータ属性に設定された値のコンボボックスを選択
        var combo =$('[data-customlistc]')
           .filter('[data-customlistc='+ $(this).data('customlist')+']');
        //コンボボックスで選択した値をどの入力欄に戻すのかを設定
        combo.data('returnkey',$(this).attr('id'));
        //コンボボックスの位置を入力欄と同じ座標に設定
        combo.offset($(this).offset());
        //コンボボックスの位置を入力欄の下に設定
        combo.css('top',$(this).outerHeight() + $(this).offset().top);
        //入力欄の値がコンボボックスの要素にある場合は選択状態に設定
        combo.val($(this).val());
        //コンボボックスの表示
        combo.show();
    /* ～省略 ソース9 の内容～ */
</script>
```

## ソース8　入力欄のイベント定義 メソッドチェーンで記述

```
<script>
  $(function(){
    $('[data-customList]').on({
      //入力欄にフォーカスが設定されると、関連するコンボボックスを表示
      'focus' : function(){
        $('[data-customlistc]').hide().css('left','0').css('top','0')
         .filter('[data-customlistc='+ $(this).data('customlist')+']')
         .data('returnkey',$(this).attr('id')).offset($(this).offset())
         .css('top',$(this).outerHeight() + $(this).offset().top)
          .val($(this).val()).show();
      },'click' : function(){
        //入力候補を表示しないための設定
        return false;
      }
    });

    /* ～省略 ソース9 の内容～ */
</script>
```

見やすいように改行を含めているが
1行で記述できる

## ソース9　コンボボックスのイベント定義

```
<script>
  $(function(){
    /*  ～省略 ソース7 の内容～ */

    //他の要素にフォーカスが当たるとコンボボックスを非表示
    $('input:not([data-customlist])').on('focus',function(){
      $('[data-customlistc]').hide();
    });

    //入力欄以外の要素をクリックしたらコンボボックスを非表示
    $(document).on('click',function(){
      $('[data-customlistc]').hide();
    });

    //コンボボックスが選択されると入力欄に値を設定
    $('[data-customlistc]').on('change',function(){
      $('#' + $(this).data('returnkey')).val($(this).val());
    });
  });
</script>
```

## 図12　絞り込み可能なコンボボックス

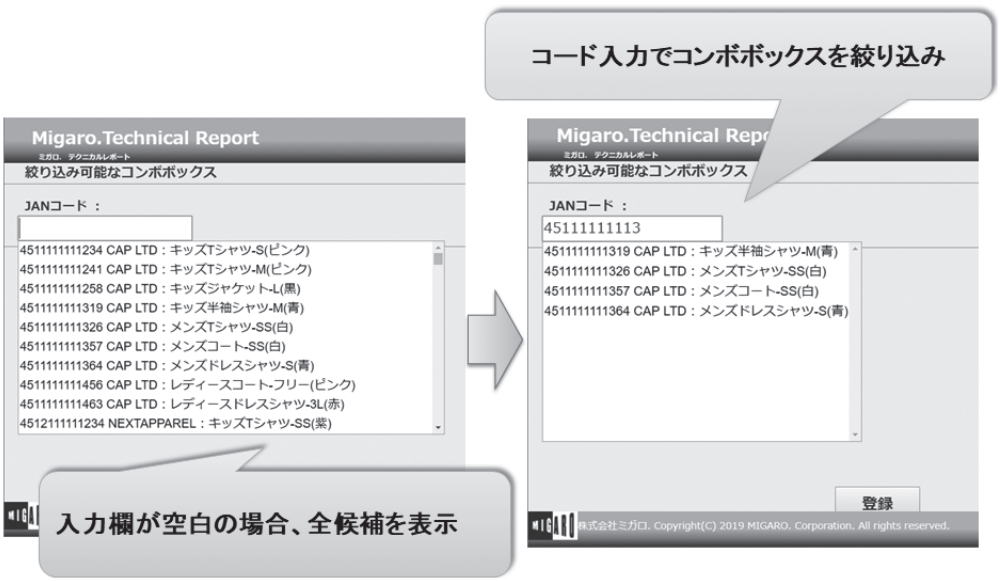


図13　3-3で使用するjQuery、JavaScriptのメソッド

| 関数/メソッド | 機能 |
|---|---|
| .remove | 要素の削除 |
| .append | 要素の追加 |
| .clone | 要素の複製 |
| .not | 指定した要素の除外 |
| .length | 要素の数を取得 |

ソース10　コンボボックスの絞り込み処理イベント定義

```
<script>
  $(function(){
    //コンボボックス項目のバックアップ
    var clonecmb = null;
    $('[data-customList]').on({
      //入力欄にフォーカスが設定されると、関連するコンボボックスを表示
      'focus' : function(){
        $('[data-customlistc]').hide().css('left','0').css('top','0')
         .filter('[data-customlistc='+ $(this).data('customlist')+']')
         .data('returnkey',$(this).attr('id')).offset($(this).offset())
         .css('top',$(this).outerHeight() + $(this).offset().top)
          .val($(this).val()).show();
      },'click' : function(){
        //入力候補を表示しないための設定
        return false;
      },//イベント追加
      'keyup focus' : function(){
          /* ～省略 ソース11 の内容～ */
      }
    });
    /* ～省略 ソース9 の内容～ */
</script>
```

コンボボックス項目のバックアップ

keyup と focusイベント

ソース11　コンボボックスの絞り込み処理

```
        var inputValue = $(this).val().toUpperCase();
        var findval = '[value^="' + inputValue + '"]';
        var cmb = $('[data-customlistc]')
          .filter('[data-customlistc='+ $(this).data('customlist')+']');
        if(!clonecmb){
          clonecmb = cmb.find('option').clone(false);
        }else{
          //全項目を再作成
          cmb.find('option').remove();
          cmb.append(clonecmb.clone(false));
        }
        if(inputValue==''){
          return;
        }else{
          //入力値がコンボボックスの項目に含まれない場合、項目を削除
          cmb.find('option').not(findval).remove();
          if(cmb.find('option').length == 0){
            //項目が存在しない場合
            cmb.hide();
          }else{
            //項目が存在する場合
            cmb.show();
          }
        }
```