



JavaScript の使い方

Release 12

© 2002-2008 Unify Corporation All rights reserved. Sacramento California, USA

No part of this tutorial may be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any language or computer language, in any form or by any means, electronic, mechanical, magnetic, optical, chemical, manual or otherwise without the prior written consent of Unify Corporation.

Unify Corporation makes no representations or warranties with respect to the contents of this document and specifically disclaims any implied warranties of merchantability or fitness for any particular purpose. Further, Unify Corporation reserves the right to revise this document and to make changes from time to time in its content without being obligated to notify any person of such revisions or changes.

The Software described in this document is furnished under a Software License Agreement. The Software may be used or copied only in accordance with the terms of the license agreement. It is against the law to copy the Software on tape, disk, or any other medium for any purpose other than that described in the license agreement.

The Unify Corporation Documentation Group values and appreciates any comments you may have concerning our documents. Please address comments to:

doc@unify.com

1-800-24 UNIFY or 1-800-GO-UNIFY;(916) 928-6400
FAX (916) 928-6401

UNIFY and DataServer are registered trademarks of Unify Corporation. Unify NXJ is a trademark of Unify Corporation. Java and J2EE are registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries. JReport is a trademark of Jinfonet Corporation. IBM, Lotus, Lotus Notes, Cloudscape, and WebSphere are trademarks of International Business Machines Corporation in the United States, other countries, or both. CAS AHL Technology and ecKnowledge are registered trademarks of CAS AHL Technology, Inc. in the U.S. and other countries. All other products or services mentioned herein may be registered trademarks, trademarks, or service marks of their respective manufacturers, companies, or organizations.

Name: Working with JavaScript

Release: Unify NXJ 12

Last Revision: January 23, 2009 2:58 pm

JavaScript の使い方

このマニュアルでは、JavaScript ベースのコントロールを作成して NXJ フォームに追加する方法を説明します。JavaScript ベースのコントロールとは、実行時のユーザインタフェースが JavaScript で記述されているコントロールです。JavaScript ベースのコントロールには以下の 2 つのカテゴリがあります。

- フォームの部品として作成されるコントロール
- 再利用可能なスタンドアロンコントロール

このマニュアルでは、それぞれのタイプの JavaScript ベースのコントロールを作成する作業について説明します。

フォームのパーツとして作成する JavaScript コントロール

このセクションでは、JavaScript を使って NXJ フォームにデジタル時計を追加する例を示します。埋込み型 JavaScript コントロールを作成するには、以下のタスクを実行します。

- タスク 1: JavaScript ソースファイルの作成
- タスク 2: JavaScript をフォームに配置する
- タスク 3: JavaScript に関連するプロパティを設定
- タスク 4: アプリケーションの実行

タスク 1: JavaScript ソースファイルの作成

最初に、時計の関数宣言を含む JavaScript ソースファイルを作成します。アプリケーションデザイナーで、Static Content フォルダを右クリックして、**新規作成 > JavaScript** を選択します。これは自動的に新しい .js ファイルを作成し、開かれるのでそのファイルを使って編集します。

下のコードをコピーして、新しいファイルに貼り付けます。

```
// Writes the current time in the clock <div>
function setClock()
{
    var time = new Date();
    var hours = time.getHours();
    var minutes = time.getMinutes();
    minutes=((minutes < 10) ? "0" : "") + minutes;
    var seconds = time.getSeconds();
    seconds=((seconds < 10) ? "0" : "") + seconds;
    document.getElementById('clock').innerHTML =
hours + ":" + minutes + ":" + seconds;
    //set a timer to change the clock every second
    timer = setTimeout("setClock()",1000);
}

// Renders the clock <div> to the HTML document
document.write("<div id='clock'></div>");
setClock();
}
```

これを保存して 'clock' に名前を変更します。.js 拡張子は、アプリケーションデザイナーによって追加されます。拡張子は、表示されたオブジェクト名には現れません。

タスク 2: JavaScript をフォームに配置する

時計を追加したいフォームを開きます。最初に、コントロールパレットの JavaScript ボタンをクリックして、次にデザインパネルのフォームをクリックします。この操作で、フォームにコントロールを配置し、'X' とボックスを使い中央に JavaScript と書かれた様式で表示されます。このコントロールは、Java スクリプトの実行時の見え方を表現するのではなく、フォームにスペースを確保するサイズを設定します。



JavaScript ボタン

タスク 3: JavaScript に関連するプロパティを設定

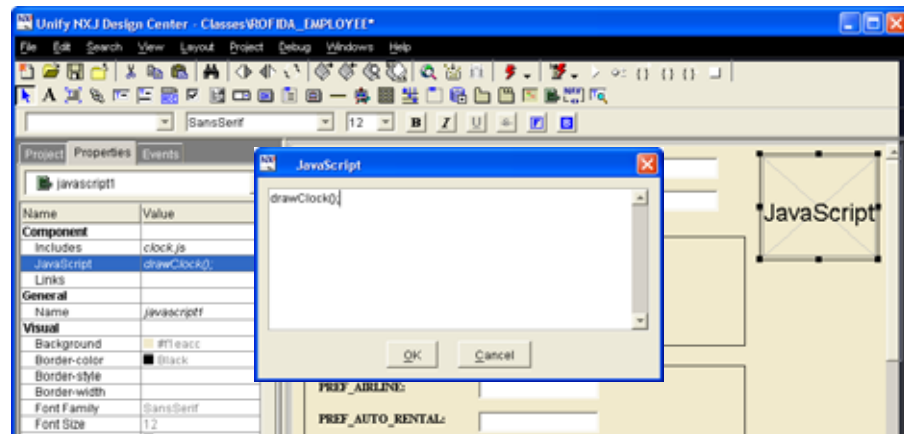
3つのコントロールプロパティを使って JavaScript コントロールを設定します。Includes、JavaScript、Links です。

- Includes プロパティ

JavaScript コントロールの **Includes** プロパティの値の列をクリックします。Select JavaScript Files ダイアログが表示されるので、フォームに呼び出したい JavaScript を含むファイルを指定します **clock.js** ファイルを選択して、矢印 ボタンを使って含むの列に移動します。OK ボタンをクリックして、ダイアログを閉じます。

- JavaScript プロパティ

JavaScript コントロールの **JavaScript** プロパティの値の列をクリックします。この操作は、実行したい JavaScript を入力することのできる JavaScript エディタを開きます。多くのサードパーティ JavaScript コンポーネントは、オブジェクトの初期化と適切な HTML を表現するためのメソッドを持っています。サンプルでは、上記の drawClock() メソッドによって行われます。これを呼び出すには、ポップアップダイアログに drawClick() を入力します。



次に **OK** ボタンをクリックして、プロパティの設定を保存します。

- Links プロパティ

JavaScript コントロールの Links プロパティの値の列をクリックします。この操作は、Style Sheet ファイルのドロップダウンリストを開きます。JavaScript コンポーネントが専用のスタイルセットを使っている場合、フォームに含めるためにここにスタイルセットを追加することができます。このサンプルは追加のファイルを必要としないため何も入力しません（定義済みスタイルの1つを使用するか、**default.css** ファイルで新しいスタイルを定義することができます。）

タスク 4: フォームの実行

最後に、フォームを実行すると、時計が現在の時刻を示してフォーム上に現れます。

RERS Employee Details DEPT: 101 9:51:31

EMP_ID: 1101 MANAGER:

Personal Information

FIRST_NAME: Hugh

スタンドアローン JavaScript コントロール

スタンドアローン JavaScript コントロールは、NXJ アプリケーションのコンポーネントとして使用できます。スタンドアローン JavaScript コントロールには、JavaScript フィールドコントロールと JavaScript トリムコントロールの 2 種類があります。

- JavaScript フィールドコントロール：値を持つ JavaScript フィールドコントロールです。NXJ プログラミング言語の観点からは、テキストフィールドなどの他のフィールドコントロールに似ています。JavaScript フィールドコントロールの例としては日付選択コントロールやスピナコントロールがあります。
- JavaScript トリムコントロール：値を持たない JavaScript トリムコントロールです。NXJ プログラミング言語の観点からは、行などの他のトリムコントロールに似ています。

JavaScript ベースのコントロールは、以下の 3 つのファイルで定義します。

- JavaScript ソースファイル。
- オプションのグラフィカル JavaBean。JavaBean は、アプリケーションデザイナーでコントロールを描画し、JavaScript を生成してクライアント上でコントロールをインスタンス化するために使用します。
- オプションの BeanInfo クラス。BeanInfo は、アプリケーションデザイナーを拡張してコントロールのカスタムプロパティをサポートするために使用します。

このマニュアルでは、JavaScript ベースのフィールドコントロールを例として使用します。使用するコントロールは、ユーザが値を設定するために使用できるスピナオブジェクトです。



JavaScript ベースのコントロールを作成するには以下の作業を実行します。

- タスク 1: JavaScript ソースを記述する
- タスク 2: (オプション) グラフィカル JavaBean を作成する
- タスク 3: (オプション) BeanInfo を作成する
- タスク 4: コンポーネントをフォームや他のコンテナに配置する
- タスク 5: アプリケーションの実行

作業内容は以下のセクションで説明します。

タスク 1: JavaScript ソースを記述する

JavaScript ファイルには、コントロールを表すオブジェクトのコンストラクタがあることが必要です。通常、コンストラクタには多くの引数があります。これらの引数には前後を囲む **div** の名称もあります。コンストラクタには以下の機能があります。

- 視覚的に表現したコントロールを含めるようにドキュメントを変更する。これは、`document.write` を呼び出すか **div** に内部タグを設定すると実行できます。
- コントロールに一連のメソッドを定義する。これらのメソッドには `getValue` と `setValue` があります。また、すべてのコントロール専用プロパティで使用できるゲッターとセッターもあります。

ゲッターには引数がなく、プロパティの現在の値を返します。セッターには引数が 1 つあります。プロパティの値です。値が有効であれば `true` を返し、値が無効であれば `false` を返します。

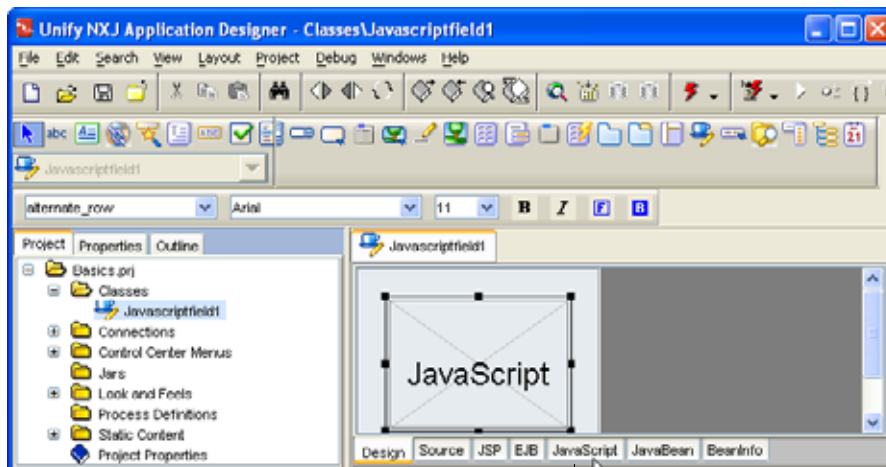
コントロールがフィールドの場合、`getValue/setValue` メソッドのペアを定義する必要もあります。これらのメソッドにはコントロール専用プロパティのゲッターおよびセッターと同じシングニチャがありますが、取得して設定するのはコントロールの値です。

JavaScript フィールドの作成

1. アプリケーションデザイナーで **ファイル > プロジェクトを開く** を選択し、JavaScript コントロールを含めるプロジェクトを開きます。
2. **Classes** フォルダを右クリックして **新規作成 > JavaScript Field** を選択し、新しい空の JavaScript フィールドコントロールを作成します。

コントロールが作成され、デフォルトの名前が割り当てられます。コントロール名を変更するには、プロジェクトタブで名前を右クリックして**名前の変更**を選択します。

コンテンツパネルのデザインタブに、図のような JavaScript プレイスホルダコントロールが表示されます。



JavaScript タブ(ステップ 3)

3. **JavaScript** タブをクリックすると JavaScript テンプレートファイルが表示されます。

アプリケーションデザイナーで、コントロールの JavaScript ファイルのデフォルトのテンプレートが作成されます。このファイルにはコントロール名を名前に持つコンストラクタがあり、アンダースコアの代わりにスラッシュとバックスラッシュが使用されています。コンストラクタには引数が 1 つあります。div の名称です。

コントロールがフィールドの場合、ファイルにも空の `getValue` メソッドと `setValue` メソッドが含まれます。コンストラクタはメソッドを設定し、`registerComponent` を呼び出します。コントロールがトリムの場合、コンストラクタは `registerComponent` だけを呼び出します。


```

function Javascriptfield2(divName)
{
var div = window.document.getElementById(divName);
if (div)
{
div.getvalue = this.getvalue;
div.setvalue = this.setvalue;
div.entervalue = this.entervalue;
// add other code to create the element
}
}
Javascriptfield2.prototype.getvalue = function()
{
return this.value;
}
Javascriptfield2.prototype.setvalue = function(value)
{
this.value = value;
// add code to display value
}
Javascriptfield2.prototype.entervalue = function()
{
// add code to get input value from the user
this.setvalue(value);
}

```

4. 独自のコードをテンプレートに追加します。

例えば、スピナ JavaScript 定義は以下のようになります。

```

function Spinner(divName)
{
this.div = window.document.getElementById(divName);
if (this.div)
{
this.div.getvalue = Spinner.prototype.getvalue;
this.div.setvalue = Spinner.prototype.setvalue;
this.div.entervalue = Spinner.prototype.entervalue;
this.div.value = 0;
// add other code to create the element
var parentDiv;
if (this.div.parentNode)
parentDiv = this.div.parentNode;
else if (this.div.parentElement)
parentDiv = this.div.parentElement;
var divWidth = parentDiv.style.width.slice(0,-2);
var index = document.images.length;
document.write('<table cellpadding="0" cellspacing="0" border="0"><tr><td
rowspan="2">');
document.write('<input style="float: left; width: ` + (divWidth - 11) + `px"
type="text" name="' + divName + `input" id="' + divName + `input">');
document.write('</td><td>');
document.write('<a href="about:blank" ` +
`onclick="document.images[` + index + `]._sp.onClickUp(); return false;" `
+
`onmousedown="document.images[` + index + `].src = ` + `../Static_Content/
SpinnerUpPressed.png` + `"; ` +

```

```

        `onmouseup="document.images[ ` + index + ` ].src = ¥'../Static_Content/
SpinnerUpUnpressed.png¥';" ` +
        `onmouseout="document.images[ ` + index + ` ].src = ¥'../Static_Content/
SpinnerUpUnpressed.png¥';" ` +
        `id="" + divName + `up">');
        document.write(`');
        document.write(`</a></td></tr><tr><td>');
        document.write(`<a href="about:blank" ` +
        `onclick="document.images[ ` + (index + 1) + ` ]._sp.onClickDown(); return
false;" ` +
        `onmousedown="document.images[ ` + (index + 1) + ` ].src = ¥'../
Static_Content/SpinnerDownPressed.png¥';" ` +
        `onmouseup="document.images[ ` + (index + 1) + ` ].src = ¥'../
Static_Content/SpinnerDownUnpressed.png¥';" ` +
        `onmouseout="document.images[ ` + (index + 1) + ` ].src = ¥'../
Static_Content/SpinnerDownUnpressed.png¥';" ` +
        `id="" + divName + `down">');
        document.write(`');
        document.write(`</a></td></tr></table>');
        document.images[index]._sp = this;
        document.images[index + 1]._sp = this;
        this.div.textField = document.all[divName + `input'];
    }
}
Spinner.prototype.getValue = function()
{
    return this.textField.value;
}
Spinner.prototype.setValue = function(value) {
    this.textField.value = value;
}
Spinner.prototype.enterValue = function() {
    // add code to get input value from the user
    // We are using a standard text field as the main object in our
    // Javascript component, so this is not needed.
}
Spinner.prototype.onClickUp = function() {
    this.div.textField.value++;
}
Spinner.prototype.onClickDown = function() {
    this.div.textField.value--;
}
}

```

5. ファイル > 保存 を選択して JavaScript を保存します。

フォームスクリプトおよび Java ソースとは異なり、JavaScript 構文エラーは NXJ デザインセンタに報告されません。

タスク 2: (オプション) グラフィカル JavaBean を作成する

JavaScript コントロール上でコントロール専用プロパティをサポートしたり、デザインパネルでコントロールを描画する方法をカスタマイズする場合は、JavaBean を定義する必要があります。

JavaBean は Swing コンポーネントを継承する必要があります。また、getInstantiationCode メソッドも必要です。このメソッドには引数がなく、コントロー

ルをインスタンス化するための JavaScript スニペットを返します。ほとんどの場合、`getInstantiationCode` メソッドはこのスニペットを bean のプロパティ値の根拠にします。

タスク 1 でコントロールが作成された時点で、アプリケーションデザイナーはコントロールのテンプレート JavaBean ファイルを作成しました。



コントロールが別の JavaBean コントロールを継承する場合、テンプレートは以下のコードを構成します。

```
package <package name of component>;
import <name of base class's bean>;
class <component name>Bean
extends < base class' bean>
{
}
}
```

JavaBean ファイルを編集するには、アプリケーションデザイナーのコンテンツパネルで JavaBean タブをクリックします。コントロールの JavaBean が正しくコンパイルされると、JavaBean が表示され、コントロールが描画されます。

プロパティタブには、標準のコントロールプロパティである Background や Font があります。

JavaScript コンポーネントはコントロール専用プロパティを定義できます。これらのプロパティを使って、コンポーネントの実行時の外観とクライアントサイドの動作を設定します。コンポーネント専用プロパティは、JavaBean にゲッターとセッターを追加してオプションで BeanInfo にコードを追加して定義します。

コンポーネント専用プロパティは設計時と実行時に設定できます。サーバとクライアントの間の通信プロトコルには、名前と値のペアのリストをやりとりすることが必要です。クライアントサイドのコードは、コンポーネントのゲッターとセッターを呼び出してプロパティの取得と設定を行います。

コンパイラでは、コンポーネント専用プロパティを取得して設定するフィールドスタイル構文をサポートするために、JavaBean の内部調査、プロパティ名の指定、対応する jBiz クラスのコード生成を行うことができます。

コンポーネントをインスタンス化するコードの構成は以下のとおりです。

```
<div> <script language="javascript">
snippet returned by getInstantiationCode
</script> </div>
```

div の属性にはコンポーネントの標準コントロールプロパティ（Background、Foreground、Width、Height など）が設定されています。アプリケーションデザイナーがインスタンス化コードを生成する場合、getInstantiationCode で戻されるスニペットを実行して、\$DIV の各インスタンスを div の名称に置き換えます。

タスク 3: (オプション) BeanInfo を作成する

BeanInfo は JavaBean のメタデータを提供する Java クラスで、Beans プロパティの名前と各プロパティのカスタムエディタです。BeanInfo の使用は JavaBeans の仕様で定義されています。タスク 1 で JavaScript コントロールを作成したとき、アプリケーションデザイナーは以下のようなデフォルトの BeanInfo を作成しました。

- JavaScript コントロールが別の JavaScript コントロールを継承し、ベースコントロールに BeanInfo がある場合、アプリケーションデザイナーはそのベースコントロールの BeanInfo を継承する BeanInfo を作成します。
- それ以外の場合、BeanInfo には何も含まれません。これは、アプリケーションデザイナーはコントロールの BeanInfo クラスをコンパイルせず、BeanInfo クラスがない bean の Java 規則に基づいて bean のプロパティと配置済みエディタを指定する必要があることを示しています。

パレットに JavaScript ベースのコントロールを配置すると、アプリケーションデザイナーは BeanInfo を使ってコントロールのカスタムアイコンを検索します。BeanInfo がアイコンの相対パスを返せば、Static Content に対して相対的であると見なされます。

アプリケーションデザイナーには以下の 3 つの Swing クラスがあり、カスタムエディタとして使用できます。

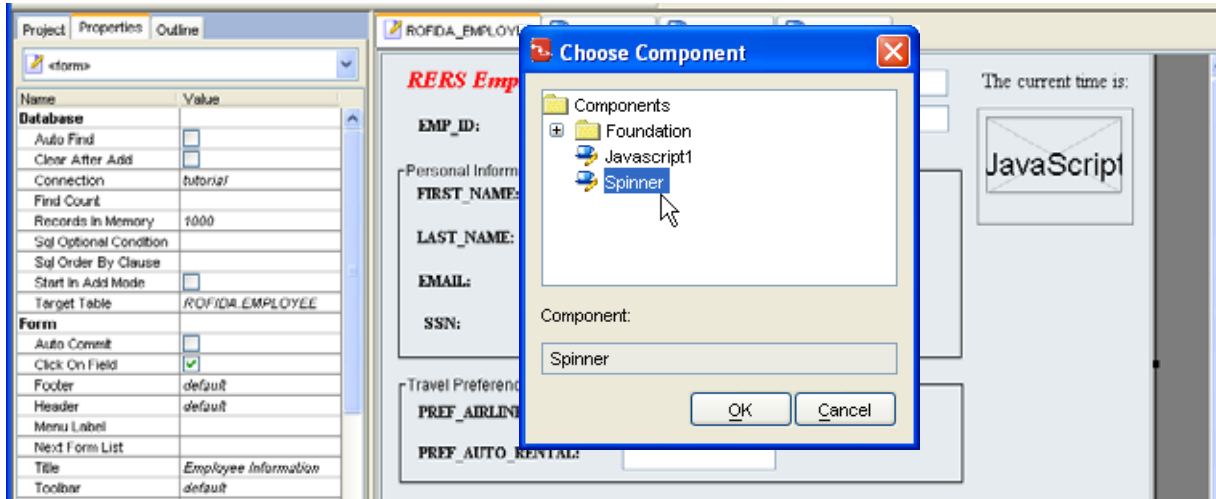
- テキストフィールド用、... ボタンあり
- アプリケーションデザイナーの色選択用
- アプリケーションデザイナーのタグエディタ用

タスク 4: コンポーネントをフォームや他のコンテナに配置する

フォームや他のコンテナに JavaScript コンポーネントを配置するステップは、以下のとおりです。

1. アプリケーションデザイナーで、JavaScript コンポーネントを配置するコンテナを開きます。
2. コンポーネントの挿入場所にあるコンテナを右クリックして、コンポーネントの挿入を選択します。

- コンポーネントの選択ダイアログで、挿入する JavaScript ベースのコンテナをクリックします。



- コンポーネントを設定します。

JavaBean コンポーネントのインスタンスを編集する場合は、以下のガイドラインに従ってください。

コンポーネントの JavaBean が正しくコンパイルされると、JavaBean が表示され、コントロールが描画されます。それ以外の場合、コンポーネントはプレースホルダボックスとして描画されます。

選択した JavaScript ベースのコントロールのプロパティタブには、標準のコントロールプロパティ（Background や Font など）があり、さらに Component Properties グループに以下のコンポーネント専用プロパティがあります。

- コンポーネントに BeanInfo 定義があり、BeanInfo のコンパイルが成功している場合、アプリケーションデザイナーは BeanInfo を使ってプロパティの一覧を指定します。
- それ以外の場合は、JavaBean のコンパイル完了後アプリケーションデザイナーは bean のゲッターとセッターを使ってプロパティの一覧を指定します。
- それ以外の場合、bean は bean 専用プロパティがないものとみなします。
- コンポーネントが JavaScript ベースのフィールドコントロールであれば、アプリケーションデザイナーは Target Field など標準フィールドプロパティも表示します。

コンポーネント専用プロパティに値をセットするには、プロパティの値の列をクリックします。BeanInfo がプロパティのカスタムエディタを定義していれば、アプリケーションデザイナーはそのカスタムエディタを配置済みエディタとして使用します。それ以外の場合、bean にカスタマイザがあれば、アプリケーションデザイナーはそのカスタマイザを表示します。

タスク 5: アプリケーションの実行

通常の方法でアプリケーションを実行または配備します。