Extending
ADOBE® DREAMWEAVER® CS4

Adobe

Adobe® Dreamweaver® CS4 拡張ガイド (Windows® / Mac OS 版)

Adobe Systems Incorporated, 345 Park Avenue, San Jose, California 95110, USA.

目次

第 6 章：Dreamweaver ドキュメントオブジェクトモデル

第 7 章：挿入バーオブジェクト

第 8 章：ブラウザ互換性チェックの問題用 API

第 9 章：コマンド

第 10 章：メニューおよびメニューコマンド

第 11 章：ツールバー

第 12 章：レポート

第 13 章：タグライブラリとタグエディタ

第 14 章：プロパティインスペクタ

第 15 章：フローティングパネル

第 16 章：ビヘイビア

第 17 章：サーバビヘイビア

第 18 章：データソース

第 19 章：サーバフォーマット

第 20 章：コンポーネント

第 21 章：サーバモデル

第 22 章：データトランスレータ

第 23 章：C レベル拡張機能

第 24 章：Shared フォルダ

第 1 章：はじめに

『**Dreamweaver CS4** 拡張ガイド』では、Dreamweaver での拡張機能の構築に使用する、Adobe® Dreamweaver® CS4 のフレームワークとアプリケーションプログラミングインターフェイス（API）について説明します。『**Dreamweaver CS4** 拡張ガイド』で説明する内容は次のとおりです。

- 各タイプの拡張機能の動作
- Dreamweaver で様々なオブジェクトを実装するために呼び出される API 関数
- メニュー、フローティングパネル、サーバビヘイビアなどの Dreamweaver の機能
- 各タイプの拡張機能の簡単な例
- 様々な HTML ファイルや XML ファイルのタグを編集してコマンドやドキュメントタイプを追加することで Dreamweaver をカスタマイズする方法

Dreamweaver の拡張機能で各種のサポート操作を実行するために使用できるユーティリティや汎用 JavaScript API については、『**Dreamweaver API** リファレンス』を参照してください。データベースで動作する拡張機能を作成する場合は、『Dreamweaver ユーザガイド』で、データベースへの接続を確立する方法を説明しているトピックを参照してください。

拡張機能について

Dreameaver の大部分の拡張機能は、HTML と JavaScript で記述されます。本マニュアルは、Dreamweaver、HTML、XML および JavaScript プログラミングに精通している方を対象としています。さらに、C 拡張機能を実装する場合は、C ダイナミックリンクライブラリ（DLL）の作成方法および使用方法を理解していることを前提としています。Web アプリケーションを構築するための拡張機能を記述する場合は、Active Server Page（ASP）、ASP.NET、Hypertext Preprocessor（PHP）、Adobe® ColdFusion®、Java Server Pages（JSP）など、少なくとも 1 つのプラットフォームでのサーバサイドスクリプトの記述に精通している必要があります。

拡張機能のインストール

拡張機能の記述処理を理解するには、Adobe Exchange の Web サイト http://www.adobe.com/go/exchange_jp で提供されている拡張機能やリソースも役立ちます。既存の拡張機能をインストールすると、自分で拡張機能を記述する際に使用する必要があるツールがいくつか判明します。

1. Adobe ダウンロード Web サイト http://www.adobe.com/go/downloads_jp から、Adobe® Extension Manager をダウンロードし、インストールします。
2. http://www.adobe.com/go/exchange_jp から Adobe Exchange Web サイトにログオンします。
3. 使用可能な拡張機能から、いずれかを選択します。「ダウンロード」リンクをクリックして、拡張機能パッケージをダウンロードします。
4. 拡張機能パッケージを、Dreamweaver がインストールされているフォルダの Dreamweaver¥Downloaded Extensions フォルダに保存します。
5. Extension Manager で、ファイル／拡張機能をインストールを選択します。Dreamweaver で、コマンド／拡張機能の管理を選択し、Extension Manager を起動します。

Extension Manager により、拡張機能が Downloaded Extension フォルダから Dreamweaver に自動的にインストールされます。

拡張機能の中には、使用する前に Dreamweaver を再起動しなければならないものもあります。拡張機能をインストールする際に Dreamweaver を実行していた場合は、アプリケーションを終了してから再起動するようにメッセージが画面に表示されることがあります。

インストール後に拡張機能に関する基本的な情報を表示するには、Dreamweaver でコマンド／拡張機能の管理を選択し、Extension Manager を起動します。

拡張機能の作成

Dreamweaver 拡張機能を作成する前に、Adobe Exchange Web サイト http://www.adobe.com/go/exchange_jp で、作成する拡張機能が既に存在していないか確認します。ニーズを満たす拡張機能が見つからない場合は、次の手順を実行して拡張機能を作成します。

- 作成する拡張機能の種類を決定します。拡張機能の種類について詳しくは、74 ページの「Dreamweaver 拡張機能の種類」を参照してください。
- マニュアルを参照して、作成する拡張機能の種類を確認します。拡張機能の種類の作成について理解するには、適切なトピックで示された簡単な拡張機能の例を作成することをお勧めします。
- 変更または作成する必要があるファイルを判断します。
- 必要に応じて拡張機能のユーザインターフェイス（UI）を計画します。
- 必要なファイルを作成し、適切なフォルダに保存します。
- 新しい拡張機能を認識するように、Dreamweaver を再起動します。
- 拡張機能をテストします。
- 他のユーザと共有できるように拡張機能をパッケージ化します。詳しくは、80 ページの「Extension Manager の操作」を参照してください。

拡張機能を開発する際に使用できるその他のリソース

Dreamweaver のオンラインフォーラムに登録し、拡張機能を開発している他のデベロッパーと情報を交換することができます。このオンラインフォーラムには、
http://www.adobe.com/cfusion/webforums/forum/categories.cfm?forumid=12&catid=190&entercat=y からアクセスできます。

Dreamweaver CS4 の新機能

Dreamweaver CS4 の拡張可能な新しい機能とインターフェイスを次に示します。

- 関連ファイルを使用した Web ページの表示および編集
- ライブビュー機能を使用して Web ページを Web ブラウザと同じように表示
- 垂直分割ビュー機能を使用して HTML ページのコードとデザインを並べて表示
- ワークフローを中断せずにエラーメッセージを表示
- Adobe Photoshop を開かずに Adobe Photoshop スマートオブジェクトを使用してソースイメージをアップデート

これらの各機能に関連する新しい関数がユーティリティ API と JavaScript API に追加されています。これらの関数について詳しくは、『**Dreamweaver API** リファレンス』を参照してください。

ドキュメンテーションリソースセンター

Adobe から出版されているマニュアルを参照して、Dreamweaver のスキル向上に役立てることができます。専門家によって執筆された最新のコンテンツは、http://www.adobe.com/support/documentation/buy_books.html でチェックできます。

非推奨の関数

Dreamweaver では、いくつかの機能が削除されました。ユーティリティ API および JavaScript API から削除された関数については、『Dreamweaver API リファレンス』を参照してください。

本マニュアルの表記上の規則

本マニュアルでは、次のような表記規則を使用しています。

- コードの書式が設定されているフォントは、コードの一部および API リテラルを示します。このフォントで表記される項目には、クラス名、メソッド名、関数名、タイプ名、スクリプト、SQL ステートメント、HTML タグ、XML タグおよび属性名などがあります。
- *イタリックコード*の書式が設定されているフォントは、コード内の置き換え可能な項目を示します。
- 継続記号（¬）は、1 行の長いコードが 2 行以上にまたがっていることを示します。本マニュアルの書式に基づくマージン制限により、本来は 1 行として入力する必要のあるコード行を分割して表記しています。コード行をコピーするときは、継続記号を削除して 1 行として入力してください。
- 関数の引数が波カッコ（{ }）で囲まれている場合は、その引数が省略可能であることを示しています。
- 接頭辞 dreamweaver を持つ関数名（dreamweaver.***funcname*** など）は、コードでは省略して dw.***funcname*** と記述することができます。本マニュアルでは、関数定義と索引では完全な dreamweaver. 接頭辞を使用しています。ただし、例の多くでは省略形の接頭辞 dw. を使用します。

  本マニュアルでは、以下の語句を使用します。

- デベロッパー - 拡張機能を作成するデベロッパー
- ユーザ - Dreamweaver の使用者
- ビジター - ユーザが作成した Web ページの閲覧者

第 2 章：Dreamweaver のカスタマイズ

Adobe Dreamweaver 拡張機能を作成して使用するだけでなく、Dreamweaver を様々な方法でカスタマイズして快適で効率的な作業環境にすることができます。

Dreamweaver のカスタマイズ方法

Dreamweaver をカスタマイズするには、いくつかの一般的な方法があります。これらの方法の一部は、『Dreamweaver ユーザガイド』で説明されています。様々な領域で環境設定を行うことができます。例えば、環境設定パネル（編集／環境設定または Dreamweaver ／環境設定（Mac OS X））を使用して、アクセシビリティ、コードカラーリング、フォント、ハイライト表示、ブラウザでプレビューなどを設定できます。また、キーボードショートカットエディタ（編集／キーボードショートカット）を使用するか、設定ファイルを編集すると、キーボードショートカットを変更できます。

デフォルトのドキュメントのカスタマイズ

DocumentTypes¥NewDocuments フォルダには、Dreamweaver を使用して作成できる各種のデフォルト（空白）のドキュメントが格納されています。ファイル／新規を選択し、「基本ページ」カテゴリー、「動的ページ」カテゴリーまたは「その他」カテゴリーから項目を選択して新しい空白のドキュメントを作成すると、このフォルダ内の該当するデフォルトのドキュメントに基づいて新規ドキュメントが作成されます。既定のデフォルトのドキュメントに表示される内容を変更するには、このフォルダ内の該当するドキュメントを編集します。

注意：サイト内のすべてのページに共通のエレメント（著作権情報など）や共通のレイアウトを組み込む場合は、デフォルトのドキュメントを変更するのではなく、テンプレートとライブラリ項目を使用することをお勧めします。テンプレートとライブラリ項目について詳しくは、『Dreamweaver ユーザガイド』を参照してください。

ページデザインのカスタマイズ

Dreamweaver には、定義済みのカスケーディングスタイルシート、フレームセット、ページデザインが多数用意されています。ファイル／新規を選択すると、これらのデザインに基づいてページを作成できます。

利用できるデザインをカスタマイズするには、BuiltIn¥css、BuiltIn¥framesets、BuiltIn¥Templates、BuiltIn¥TemplatesAccessible の各フォルダのファイルを編集します。

注意：「ページデザイン」カテゴリーと「ページデザイン（アクセシブル）」カテゴリーに一覧表示されているデザインは、Dreamweaver テンプレートファイルです。テンプレートについて詳しくは、『Dreamweaver ユーザガイド』を参照してください。

また、BuiltIn フォルダのサブフォルダにファイルを追加して、カスタムページデザインを作成することもできます。新規ドキュメントダイアログボックスにファイルの説明を表示するには、ページデザインファイルに対応するデザインノートファイルを、該当する _notes フォルダ内に作成します。

ダイアログボックスの外観のカスタマイズ

オブジェクト、コマンド、ビヘイビア用のダイアログボックスのレイアウトは、HTML フォームで指定されます。これらのフォームは、Dreamweaver アプリケーションフォルダ内の Configuration フォルダの HTML ファイルにあります。Dreamweaver で他のフォームを編集する場合と同じように、これらのフォームを編集します。詳しくは、『Dreamweaver ユーザガイド』を参照してください。

注意：マルチユーザオペレーティングシステムでは、Dreamweaver 設定ファイルを編集するのではなく、ユーザの Configuration フォルダ内の設定ファイルのコピーを編集する必要があります。詳しくは、76 ページの「マルチユーザ設定フォルダ」を参照してください。

ダイアログボックスの外観を変更する

1. Dreamweaver で、編集／環境設定を選択し、「コードの書き換え」カテゴリーを選択します。
2. 「ペーストするときにフォームアイテムの名前を変更」オプションをオフにします。

   このオプションをオフにすると、フォームアイテムをコピー＆ペーストしたときに、そのフォームアイテムの元の名前が保持されます。
3. 「OK」をクリックして環境設定ダイアログボックスを閉じます。
4. ハードディスク上の Configuration¥Objects、Configuration¥Commands または Configuration¥Behaviors の各フォルダで該当する HTM ファイルを見つけます。
5. Configuration フォルダ以外の場所にファイルのコピーを作成します。
6. Dreamweaver で、その作成したコピーを開き、フォームを編集して保存します。
7. Dreamweaver を終了します。
8. 変更したファイルを Configuration フォルダにコピーして、元のファイルと置き換えます。後で必要な場合に復元できるように、元のファイルのバックアップを作成しておくことをお勧めします。
9. Dreamweaver を再起動して、変更を確認します。

ダイアログボックスは外観だけを変更し、機能は変更しないでください。Dreamweaver がダイアログボックスから取得する情報を変更前と同じように使用できるよう、変更後も同じタイプのフォームエレメントが同じ名前で含まれている必要があります。

例えば、コメントオブジェクトはダイアログボックスのテキスト領域から入力テキストを取得し、簡単な JavaScript 関数を使用してそのテキストを HTML コメントに変換し、コメントをドキュメントに挿入します。ダイアログボックスについて記述したフォームは、Configuration¥Objects¥Invisibles フォルダ内の Comment.htm ファイルにあります。このファイルを開いて、サイズやその他のテキスト領域の属性を変更することができますが、textarea タグを完全に削除したり、name 属性の値を変更したりすると、コメントオブジェクトが正しく機能しなくなります。

デフォルトのファイルタイプの変更

デフォルトでは、Dreamweaver が対応するすべてのファイルタイプがファイル／開くを選択して表示されるダイアログボックスに表示されます。このダイアログボックスのポップアップメニューを使用して、表示を特定のファイルタイプに制限することができます。ある特定のファイルタイプ（ASP ファイルなど）が作業のほとんどに含まれている場合は、表示のデフォルトを変更できます。Dreamweaver Extensions.txt ファイルの最初の行に表示されたファイルタイプが、デフォルトになります。

注意：Dreamweaver で開くことができないファイルも含め、すべてのファイルタイプをファイル／開くで表示されるダイアログボックスで確認するには、「すべてのファイル (*.*)」を選択する必要があります。*). これは、Dreamweaver で開くことができるファイルだけを表示する「すべてのドキュメント」とは異なります。

Dreamweaver のファイル／開くで表示されるデフォルトのファイルタイプを変更する

1. Configuration フォルダ内の Extensions.txt ファイルのバックアップコピーを作成します。
2. テキストエディタで Extensions.txt を開きます。
3. 新しいデフォルトに対応する行をカットします。次に、その行がファイルの最初の行になるように、ファイルの先頭にペーストします。
4. ファイルを保存します。

5 Dreamweaver を再起動します。

新しいデフォルトを確認するには、ファイル／開くを選択してファイルタイプのポップアップメニューを確認します。

ファイル／開くで表示されるダイアログボックスのメニューに新しいファイルタイプを追加する

1 Configuration フォルダ内の Extensions.txt ファイルのバックアップコピーを作成します。

2 テキストエディタで Extensions.txt を開きます。

3 新規ファイルタイプごとに新しい行を追加します。新しいファイルタイプに使用するファイル拡張子を大文字で入力して、それぞれをカンマで区切ります。次に、コロンを入力し、短い説明を追加します。この説明は、ファイル／開くで開くダイアログボックスに表示されるファイルタイプのポップアップメニューに表示されます。

例えば、JPEG ファイルの場合は、「JPG,JPEG,JFIF:JPEG Image Files」と入力します。

4 ファイルを保存します。

5 Dreamweaver を再起動します。

変更を確認するには、ファイル／開くを選択してファイルタイプのポップアップメニューをクリックします。

サードパーティ提供タグの解釈のカスタマイズ

ASP、Adobe ColdFusion、JSP および PHP などのサーバサイドテクノロジーでは、HTML ファイル内で HTML 以外の特殊なコードを使用します。サーバはそのコードに基づいて HTML コンテンツを作成し、提供します。Dreamweaver は、HTML 以外のタグを検出すると、Dreamweaver で HTML 以外のタグを読み取って表示する方法を定義したサードパーティ提供タグのファイル内の情報とそれを比較します。

例えば、ASP ファイルには通常の HTML の他に ASP コードが含まれており、サーバはこれを解釈する必要があります。ASP コードは HTML タグとほぼ同じように見えますが、ペアの区切り記号でマークされています。ASP コードは <% で始まり、%> で終了します。Dreamweaver の Configuration¥ThirdPartyTags フォルダには、Tags.xml というファイルが格納されています。このファイルでは、ASP コードなど様々なサードパーティ提供タグの形式が説明され、Dreamweaver でのコードの表示方法が定義されています。Tags.xml での ASP コードの定義内容に基づいて、Dreamweaver は、区切り記号に囲まれているものは何であれ解釈を試みません。代わりに、デザインビューで ASP コードを示すアイコンを表示するだけです。独自のタグデータベースファイルで、Dreamweaver によるタグの読み取りおよび表示方法を定義することができます。タグセットごとに、タグの表示方法を指定する新規タグデータベースファイルを作成します。

注意：この節では、Dreamweaver でのカスタムタグの表示方法を定義する方法について説明しますが、カスタムタグの内容またはプロパティの編集方法を指定する方法については説明しません。プロパティインスペクタを作成してカスタムタグのプロパティを検査および変更する方法については、209 ページの「プロパティインスペクタ」を参照してください。

各タグデータベースファイルによって、カスタムタグの名前、タイプ、内容モデル、レンダリングスキームおよびアイコンが定義されます。タグデータベースファイルはいくつでも作成できますが、Dreamweaver で読み取って処理するために、作成したすべてのファイルを Configuration¥ThirdPartyTags フォルダ内に置く必要があります。タグデータベースファイルには、.xml ファイル拡張子が付けられます。

例えば、フリーランスのデベロッパーとして一度に複数の関連のないサイトで作業している場合は、1 つのファイルに特定サイトのタグ仕様をすべて格納できます。次に、そのタグデータベースファイルに、サイトの管理者となる人に引き渡すカスタムアイコンおよびプロパティインスペクタを含めます。

タグ仕様は、tagspec という XML タグで定義します。例えば、次のコードは、happy というタグの仕様を記述したものです。

```
<tagspec tag_name="happy" tag_type="nonempty" render_contents="false" content_model=
"marker_model" icon="happy.gif" icon_width="18" icon_height="18"></tagspec>
```

tagspec を使用して、次の 2 種類のタグを定義できます。

- 通常の HTML スタイルのタグ

  happy タグの例は通常の HTML スタイルタグです。開始タグ <happy> タグで始まり、開始タグと終了タグでデータを囲み、終了タグ </happy> で終わります。

- ストリング区切りのタグ

  ストリング区切りのタグは、1 つのストリングで始まり、もう一方のストリングで終わります。これはコンテンツを囲まず、終了タグがないため、空の HTML タグ（img など）に似ています。happy タグがストリング区切りのタグであれば、タグ仕様に start_string 属性と end_string 属性が含まれます。ASP タグは、ストリング区切りのタグです。これは、ストリング <% で始まってストリング %> で終了し、終了タグはありません。

次に、tagspec タグの属性と有効な値について説明します。ストリング区切りのタグでは、アスタリスク（*）が付けられた属性は無視されます。オプションの属性は、属性リスト内に波カッコ（{}）で囲まれて示されます。波カッコで囲まれていない属性はすべて必須です。

<tagspec>

説明

サードパーティ提供のタグに関する情報を提供します。

属性

tag_name、{tag_type}、{render_contents}、{content_model}、{start_string}、{end_string}、{detect_in_attribute}、{parse_attributes}、icon、icon_width、icon_height、{equivalent_tag}、{is_visual}、{server_model}

- tag_name は、カスタムタグの名前です。ストリング区切りのタグの場合、tag_name は、指定したプロパティインスペクタをタグに使用できるかどうかを判別するためだけに使用されます。プロパティインスペクタの最初の行に、このタグ名がアスタリスクに囲まれて示されている場合、プロパティインスペクタはこのタイプのタグに使用できます。例えば、ASP コードのタグ名は ASP です。ASP コードを検査できるプロパティインスペクタには、最初の行に *ASP* が含まれています。プロパティインスペクタ API について詳しくは、209 ページの「プロパティインスペクタ」を参照してください。
- tag_type は、タグが空かどうか（img タグのように）や、開始タグと終了タグの間にデータがあるかどうか（code タグのように）を指定します。この属性は、ストリング区切りではない通常のタグに必須です。ストリング区切りのタグは常に空であるため、無視されます。有効な値は "empty" と "nonempty" です。
- render_contents は、タグの内容がデザインビューに表示されるかどうかや、内容ではなく指定されたアイコンが表示されるかどうかを指定します。この属性は、空でないタグでは必須ですが、空のタグでは無視されます （空のタグには内容がありません）。この属性は、属性の外側に表示されるタグにのみ適用されます。その他のタグの属性値の内部に表示されるタグの内容は、レンダリングされません。有効な値は、"true" または "false" です。
- content_model は、そのタグに格納できる内容の種類と、HTML ファイル内のどこにそのタグを表示できるかを示します。有効な値は、"block_model"、"head_model"、"marker_model" および "script_model" です。
  - block_model は、div や p などのブロックレベルのエレメントをタグに格納できることと、body セクションや、div、layer、td など、その他 body コンテンツのタグ内にのみタグを表示できることを指定します。
  - head_model は、タグにテキストの内容を格納できることと、head セクションにのみタグを表示できることを指定します。
  - marker_model は、タグに有効な HTML コードを格納できることと、HTML ファイル内の任意の場所にタグを表示できることを指定します。Dreamweaver の HTML バリデータは、marker_model として指定されたタグを無視します。ただし、バリデータはそのようなタグの内容は無視しません。したがって、タグ自体を任意の場所に表示できる場合でも、場所によっては、タグの内容が原因で HTML が無効になることがあります。例えば、ドキュメントの head セクションで、有効な head エレメントの外側にプレーンテキストを表示することはできないので、プレーンテ

キストを含む marker_model タグを head セクションに置くことはできません。プレーンテキストを含むカスタムタグを head セクションに置くには、タグの内容モデルを head_model（marker_model ではなく）として指定します。marker_model は、インライン（例えば段落内など、p や div などのブロックレベルエレメントの内部）で表示する必要があるタグに使用します。タグの前後に改行を付けてタグを独立した段落として表示する必要がある場合は、このモデルを使用しないでください。

- script_model は、ドキュメントの開始と終了の HTML タグ間の任意の場所にタグを表示できることを示します。Dreamweaver がこのモデルを有するタグを検出すると、そのタグの内容はすべて無視されます。これは、Dreamweaver で解析しないマークアップ（特定の ColdFusion タグなど）に使用します。

- start_string は、ストリング区切りのタグの開始を示す区切り記号を指定します。ストリング区切りのタグは、コメントを表示できるドキュメント内の任意の場所に表示できます。Dreamweaver は、start_string と end_string の間では、タグの解析やエンティティまたは URL のデコードを実行しません。end_string が指定されている場合、この属性は必須です。
- end_string は、ストリング区切りのタグの終わりを示す区切り記号を指定します。start_string が指定されている場合、この属性は必須です。
- detect_in_attribute は、属性名または値の内部にストリングが表示されている場合でも、start_string と end_string の間（これらのストリングが定義されていない場合は開始タグと終了タグの間）にあるデータをすべて無視するかどうかを指定します。通常、ストリング区切りのタグの場合は "true" に設定します。デフォルトは、"false" です。例えば、ASP タグは、属性値の内部に表示される場合や、引用符（"）を含んでいる場合があります。ASP タグ仕様では detect_in_attribute="true" と指定されているので、ASP タグが属性値の内部に表示される場合、Dreamweaver は内部の引用符も含め ASP タグを無視します。
- parse_attributes は、タグの属性を解析するかどうかを指定します。これが "true" に設定されている場合（デフォルト）、Dreamweaver はタグの属性を解析します。"false" に設定されている場合、Dreamweaver は、引用符の外側に表示される次の山形閉じカッコまでのすべてを無視します。例えば、この属性を "false" に設定する必要があるのは、cfif などのタグです（<cfif a is 1> は、属性名と値のペアとして解析できません）。
- icon は、タグに関連付けられたアイコンのパスとファイル名を指定します。この属性は、空のタグ、およびドキュメントウィンドウのデザインビューに内容が表示されない空ではないタグに必須です。
- icon_width は、アイコンの幅をピクセル単位で指定します。
- icon_height は、アイコンの高さをピクセル単位で指定します。
- equivalent_tag は、特定の ColdFusion フォーム関連タグに相当する簡単な HTML を指定します。これは、他のタグと共に使用するためのものではありません。
- is_visual は、タグがページの表示に、直接影響をおよぼすかどうかを示します。例えば、ColdFusion タグ cfgraph では、is_visual の値は指定されません。したがって、値はデフォルトの "true" になります。ColdFusion タグ cfset は、is_visual を "false" に設定するように指定されています。サーバマークアップタグの表示と非表示は、環境設定ダイアログボックスの「不可視エレメント」カテゴリーで制御します。ビジュアルサーバマークアップタグの表示と非表示は、非ビジュアルサーバマークアップタグの表示と非表示とは別に設定できます。
- server_model を指定すると、指定されたサーバモデルに属するページにのみ tagspec タグが適用されます。server_model を指定しないと、tagspec タグはすべてのページに適用されます。例えば、ASP タグと JSP タグの区切り記号は同じですが、JSP の tagspec タグでは server_model が "JSP" として指定されるので、Dreamweaver によって JSP ページで該当する区切り記号を持つコードが検出された場合は、JSP アイコンが表示されます。JSP 以外のページでそうしたコードが検出された場合は、ASP アイコンが表示されます。

コンテンツ
なし（空のタグ）。

コンテナ
なし。

例

```
<tagspec tag_name="happy" tag_type="nonempty" render_contents="false" content_model=
"marker_model" icon="happy.gif" icon_width="18" icon_height="18"></tagspec>
```

デザインビューにおけるカスタムタグの表示方法

ドキュメントウィンドウのデザインビューにおけるカスタムタグの表示方法は、tag_type 属性および render_contents 属性の値（tagspec タグの属性）によって異なります。tag_type の値が "empty" の場合は、icon 属性で指定されたアイコンが表示されます。tag_type の値が "nonempty" で、render_contents の値が "false" の場合は、空のタグの場合と同じようにアイコンが表示されます。次の例では、前述部分で定義した happy タグのインスタンスが HTML でどのように表示されるかを示します。

```
<p>This is a paragraph that includes an instance of the <code>happy</code>
tag (<happy>Joe</happy>).</p>
```

タグ仕様では render_contents が "false" に設定されているので、happy タグの内容（Joe という単語）はレンダリングされません。代わりに開始タグと終了タグおよびその内容が単一のアイコンとして表示されます。

空ではないタグで render_contents の値が "true" に設定されている場合、デザインビューにアイコンは表示されません。代わりに、開始タグと終了タグの間の内容（例えば、<mytag>This is the content between the opening and closing tags</mytag> でタグに囲まれているテキスト）が表示されます。表示／不可視エレメントが有効になっている場合は、「ハイライト」環境設定で指定されたサードパーティ提供タグのカラーを使用して内容がハイライトされます。ハイライトが適用されるのは、タグデータベースファイルで定義されたタグだけです。

サードパーティ提供タグのハイライトカラーを変更する

1. 編集／環境設定を選択し、「ハイライト」カテゴリーを選択します。
2. 「サードパーティタグカラー」ボックスをクリックして、カラーピッカーを表示します。
3. カラーを選択し、「OK」をクリックして環境設定ダイアログボックスを閉じます。カラー選択について詳しくは、『Dreamweaver ユーザガイド』を参照してください。

サードパーティ提供タグの上書きの回避

Dreamweaver では特定の種類の HTML コードエラーが修正されます。詳しくは、『Dreamweaver ユーザガイド』を参照してください。デフォルトでは、.asp（ASP）、.cfm（ColdFusion）、.jsp（JSP）および .php（PHP）など、特定のファイル拡張子を持つファイルの HTML は変更されません。このデフォルトは、このような HTML 以外のタグに含まれるコードが誤って修正されないようにするためのものです。Dreamweaver の書き換え動作のデフォルトを変更して、ファイルを開いたときに HTML を書き換えられるようにすることや、Dreamweaver によって書き換えられないタイプのリストに他のファイルタイプを追加することができます。

特定の特殊文字をプロパティインスペクタに入力すると、Dreamweaver がそれらの文字を数値に置き換えて、エンコードします。通常は、このエンコードを適用したほうが、特殊文字が各種プラットフォームとブラウザで正しく表示される可能性が高くなります。ただし、こうしたエンコードはサードパーティ提供のタグに干渉する可能性があるため、サードパーティ提供のタグを含むファイルで作業している場合は、Dreamweaver のエンコード動作を変更することもできます。

より多くの種類のファイルで HTML の書き換えを可能にする

1. 編集／環境設定を選択し、「コードの書き換え」カテゴリーを選択します。
2. 次のオプションのいずれかを選択します。
   - 無効なネストや開始タグを修正する
   - 余分な終了タグを削除する
3. 次のいずれかの操作を実行します。
   - 「コードの書き換え禁止 : 拡張子付きのファイル内」オプションで拡張子のリストから拡張子を削除します。

- 「コードの書き換え禁止 : 拡張子付きのファイル内」オプションをオフにします。このオプションをオフにすると、すべてのタイプのファイルで HTML が書き換えられます。

Dreamweaver で書き換えないファイルタイプを追加する

1 編集／環境設定を選択し、「コードの書き換え」カテゴリーを選択します。

2 次のオプションのいずれかを選択します。

- 無効なネストや開始タグを修正する
- 余分な終了タグを削除する

3 「コードの書き換え禁止 : 拡張子付きのファイル内」オプションがオンになっていることを確認し、テキストフィールド内のリストに新しいファイル拡張子を追加します。

ファイル／開くで開くダイアログボックスに表示されるファイルタイプのポップアップメニューに新しいファイルタイプが表示されない場合は、Configuration/Extensions.txt ファイルにそれを追加します。詳しくは、5 ページの「デフォルトのファイルタイプの変更」を参照してください。

Dreamweaver のエンコードオプションをオフにする

1 編集／環境設定を選択し、「コードの書き換え」カテゴリーを選択します。

2 「特殊文字」オプションのいずれかまたは両方をオフにします。

その他の「コードの書き換え」環境設定について詳しくは、『Dreamweaver ユーザガイド』を参照してください。

マルチユーザ環境での Dreamweaver のカスタマイズ

Dreamweaver は、Microsoft® Windows® XP、Windows Vista、Mac OS® X. などのマルチユーザオペレーティングシステムでカスタマイズできます。任意のユーザのカスタマイズ設定は、他のユーザの設定に影響しません。マルチユーザオペレーティングシステムで Dreamweaver を初めて実行したときに、設定ファイルがユーザの Configuration フォルダにコピーされます。ダイアログボックスとパネルを使用して Dreamweaver をカスタマイズすると、Dreamweaver の設定ファイルではなくユーザの設定ファイルが修正されます。マルチユーザ環境で Dreamweaver をカスタマイズするには、Dreamweaver の設定ファイルではなく、該当するユーザの設定ファイルを編集します。大半のユーザに影響する変更を行うには、Dreamweaver の設定ファイルを編集します。ただし、対応するユーザの設定ファイルを既に開いているユーザには、変更は表示されません。すべてのユーザに影響する変更を行うには、拡張機能を作成し、Extension Manager を使用してインストールします。

注意：古いマルチユーザオペレーティングシステム（Windows 98、Windows ME および Mac OS 9.x）では、すべてのユーザが 1 つの Dreamweaver 設定ファイルを共有します。

ユーザの Configuration フォルダの場所は、ユーザのプラットフォームによって異なります。

Windows XP プラットフォームでは次の場所を使用します。

ハードディスク :¥Documents and Settings¥ ユーザ名 ¥Application Data¥Adobe¥Dreamweaver CS4¥Configuration

注意：このフォルダは、隠しフォルダ内部にある可能性があります。

Windows Vista プラットフォームでは次の場所を使用します。

ハードディスク :¥Users¥ ユーザ名 ¥AppData¥Roaming¥Adobe¥Dreamweaver CS4¥Configuration

Mac OS X プラットフォームでは次の場所を使用します。

ハードディスク :\Users/ ユーザ名 /Library/Application Support/Adobe/Dreamweaver CS4/Configuration

注意：マルチユーザオペレーティングシステムで、すべてのユーザが使用できる状態で拡張機能をインストールするには、管理者（Windows）または root（Mac OS X）としてログインする必要があります。

Dreamweaver を初めて実行したときに、一部の設定ファイルのみが Configuration フォルダにコピーされます。コピーされるファイルは、Configuration フォルダの version.xml ファイルで指定されます。アプリケーションから Dreamweaver をカスタマイズすると、設定ファイルがユーザの Configuration フォルダにコピーされます。例えば、スニペットパネルで定義済みのコードスニペットの 1 つを変更すると、ファイルがコピーされます。ユーザの Configuration フォルダにあるファイルは常に、Dreamweaver の Configuration フォルダのファイルよりも優先されます。設定ファイルをカスタマイズするには、ファイルがユーザの Configuration フォルダに存在する必要があります。ファイルがまだコピーされていない場合は、ユーザの Configuration フォルダのファイルをコピーして編集します。

マルチユーザ環境での設定ファイルの削除

マルチユーザオペレーティングシステムを使用している場合、設定ファイルを削除する操作（スニペットパネルから定義済みのスニペットを削除するなど）を Dreamweaver 内で実行すると、ユーザの Configuration フォルダに mm_deleted_files.xml というファイルが作成されます。ファイルが mm_deleted_files.xml にある場合、Dreamweaver はそのファイルが存在しないかのように動作します。

設定ファイルを非アクティブにする

1. Dreamweaver を終了します。
2. テキストエディタを使用して、ユーザの Configuration フォルダ内の mm_deleted_files.xml を編集します。ファイルに項目タグを追加し、非アクティブにする設定ファイルのパス（Dreamweaver の Configuration フォルダを基準とする相対パス）を指定します。

   注意：Dreamweaver で mm_deleted_files.xml を編集しないでください。
3. mm_deleted_files.xml を保存して閉じます。
4. Dreamweaver を再起動します。

mm_deleted_files.xml タグのシンタックス

mm_deleted_files.xml ファイルには、Dreamweaver によって無視される設定ファイルを示す、構造化された項目リストが含まれています。これらの項目は XML タグで指定されており、テキストエディタで編集可能です。

後に続く mm_deleted_files.xml タグのシンタックスの説明では、オプションの属性は、属性リスト内に波カッコ（{}）で囲まれて示されます。波カッコで囲まれていない属性はすべて必須です。

<deleteditems>

説明

Dreamweaver で削除されたものとして扱われる項目のリストを保持するコンテナタグ。

属性

なし。

コンテンツ

このタグには、少なくとも 1 つの item タグを含める必要があります。

コンテナ

なし。

例

```
<deleteditems>
<!-- item tags here -->
</deleteditems>
```

<item>

説明

Dreamweaver によって無視される設定ファイルを示します。

属性

name

name 属性は設定ファイルへのパスを示します。Configuration フォルダを基準とする相対パスです。Windows では円記号（¥）、Macintosh® ではコロン（:）を使用して、パスを区切ります。

コンテンツ

なし（空のタグ）。

コンテナ

このタグは、deleteditems タグに含める必要があります。

例

```
<item name="snippets\headers\5columnwith4links.csn" />
```

マルチユーザ環境での Dreamweaver の再インストールとアンインストール

Dreamweaver のインストール後、再インストールする場合や新しいバージョンにアップグレードする場合は、Dreamweaver によって既存のユーザ設定ファイルのバックアップコピーが自動的に作成されるので、それらのファイルをカスタマイズしてある場合でも、カスタマイズ内容を確認することができます。Dreamweaver をマルチユーザシステムからアンインストールする場合（これは、管理者権限のあるユーザだけが実行できます）、各ユーザの Configuration フォルダを自動的に削除することができます。

FTP マッピングの変更

FTPExtensionMap.txt ファイル（Windows）および FTPExtensionMapMac.txt ファイル（Macintosh）は、ファイル拡張子を FTP 転送モード（ASCII または BINARY）にマップします。

2 つの各ファイルの各行には、ファイル拡張子（GIF など）と、その拡張子を持つファイルを転送する際に 2 つの FTP 転送モードのどちらを使用するかを示す単語として ASCII または BINARY が含まれています。Macintosh では、各行にクリエータコード（DmWr など）とファイルタイプ（TEXT など）が含まれます。指定したファイル拡張子を持つファイルを Macintosh でダウンロードしたときに、指定したクリエータとファイルタイプがファイルに割り当てられます。

転送中のファイルにファイル拡張子がない場合は、BINARY 転送モードが使用されます。

注意：Dreamweaver では、Macbinary モードでのファイル転送は行われません。Macbinary モードでファイルを転送する必要がある場合は、他の FTP クライアントを使用する必要があります。

次の例では、.html の拡張子を持つファイルを ASCII モードで転送するよう指示する Macintosh ファイルの行を示します。

```
HTML DmWr TEXT ASCII
```

FTPExtensionMap.txt ファイル（Windows）でも FTPExtensionMapMac.txt ファイル（Macintosh）でも、指定した行のエレメントはすべてタブで区切られます。拡張子と転送モードは、大文字で入力します。

デフォルトを変更するには、テキストエディタでファイルを編集します。

新しいファイル拡張子に関する情報を追加する

1 テキストエディタで拡張子マップファイルを編集します。

2 空白行で、ファイル拡張子を大文字で入力して Tab キーを押します。

3 Macintosh では、クリエータ、タブ、ファイルタイプ、さらにもう 1 つタブを追加します。

4 「**ASCII**」または「**BINARY**」を入力して FTP 転送モードを設定します。

5 ファイルを保存します。

Dreamweaver で拡張可能なドキュメントタイプ

XML は、複雑なドキュメントとデータ構造を定義するための高度なシステムです。Dreamweaver では、いくつかの XML スキーマを使用して、サーバビヘイビア、タグとタグライブラリ、コンポーネント、ドキュメントタイプおよび参照情報に関する情報を整理します。

Dreamweaver で拡張機能を作成し、実際に使用してみると、拡張機能で使用されるデータを管理するために XML ファイルの作成や既存の XML ファイルの修正を行う機会がたくさんあることに気付くはずです。通常は、Configuration フォルダ内の適切なサブフォルダから既存のファイルをコピーし、それをテンプレートとして使用できます。

ドキュメントタイプ定義ファイル

拡張可能なドキュメントタイプの中心となるコンポーネントは、ドキュメントタイプ定義ファイルです。定義ファイルは複数になる場合もありますが、すべて Configuration¥DocumentTypes フォルダに格納されます。各定義ファイルには、少なくとも 1 つのドキュメントタイプに関する情報が含まれています。ドキュメントタイプごとに、サーバモデル、カラーコーディングスタイル、説明などの基本情報が記述されています。

注意：Dreamweaver のドキュメントタイプ定義ファイルと、XML の DTD（document type definition：ドキュメントタイプ定義）を混同しないでください。Dreamweaver のドキュメントタイプ定義ファイルには、一連の documenttype エレメントが含まれています。各エレメントにより、ドキュメントタイプと関連付けられた定義済みのタグと属性の集まりが定義されます。Dreamweaver を起動すると、ドキュメントタイプ定義ファイルが解析され、すべての定義済みのドキュメントタイプに関する情報のデータベースがメモリ内に作成されます。

Dreamweaver には、初期ドキュメントタイプ定義ファイルがあります。これは、MMDocumentTypes.xml というファイルで、アドビ システムズ社が提供するすべてのドキュメントタイプ定義が含まれています。

ドキュメントタイプ	サーバモデル	内部タイプ	ファイル拡張子	前のサーバモデル
ASP.NET C#	ASP.NET-Csharp	Dynamic	aspx、ascx	
ASP.NET VB	ASP.NET-VB	Dynamic	aspx、ascx	
ASP JavaScript	ASP-JS	Dynamic	asp	
ASP VBScript	ASP-VB	Dynamic	asp	
ColdFusion	ColdFusion	Dynamic	cfm、cfml	UltraDeveloper 4 ColdFusion
ColdFusion Component		Dynamic	cfc	

ドキュメントタイプ	サーバモデル	内部タイプ	ファイル拡張子	前のサーバモデル
JSP	JSP	Dynamic	jsp	
PHP	PHP	Dynamic	php、php3	
ライブラリ項目		DWExtension	lbi	
ASP.NET C# テンプレート		DWTemplate	axcs.dwt	
ASP.NET VB テンプレート		DWTemplate	axvb.dwt	
ASP JavaScript テンプレート		DWTemplate	aspjs.dwt	
ASP VBScript テンプレート		DWTemplate	aspvb.dwt	
ColdFusion テンプレート		DWTemplate	cfm.dwt	
HTML テンプレート		DWTemplate	dwt	
JSP テンプレート		DWTemplate	jsp.dwt	
PHP テンプレート		DWTemplate	php.dwt	
HTML		HTML	htm、html	
ActionScript		Text	as	
CSharp		Text	cs	
CSS		Text	css	
Java		Text	java	
JavaScript		Text	js	
VB		Text	vb	
VBScript		Text	vbs	
Text		Text	txt	
EDML		XML	edml	
TLD		XML	tld	
VTML		XML	vtm、vtml	
WML		XML	wml	
XML		XML	xml	

新しいドキュメントタイプを作成する場合は、アドビ システムズ社が提供するドキュメント定義ファイル（MMDocumentTypes.xml）にエントリを追加するか、Configuration¥DocumentTypes フォルダにカスタム定義ファイルを追加します。

注意：NewDocuments サブフォルダは、Configuration¥DocumentTypes フォルダにあります。このサブフォルダには、各ドキュメントタイプのデフォルトページ（テンプレート）が含まれています。

ドキュメントタイプ定義ファイルの構造

次の例では、一般的なドキュメントタイプ定義ファイルがどのようなものかを示します。

```
<?xml version="1.0" encoding="utf-8"?>
<documenttypes     xmlns:MMString="http://www.adobe.com/schemes/data/string/">
    <documenttype
        id="dt-ASP-JS"
        servermodel="ASP-JS"
        internaltype="Dynamic"
        winfileextension="asp,htm, html"
        macfileextension=asp, html"
        previewfile="default_aspjs_preview.htm"
        file="default_aspjs.htm"
        priorversionservermodel="UD4-ASP-JS" >
        <title>
            <loadString id="mmdocumenttypes_0title" />
        </title>
        <description>
            <loadString id="mmdocumenttypes_0descr" />
        </description>
    </documenttype>
    ...
</documenttypes>
```

注意：ドキュメントタイプのカラーコーディングは、Configuration¥CodeColoring フォルダにある XML ファイルで指定します。

前の例では、loadstring エレメントによって、Dreamweaver が ASP-JS タイプのドキュメントのタイトルと記述に使用するローカライズされるストリングが識別されます。ローカライズされるストリングについて詳しくは、20 ページの「ローカライズされるストリングの提供」を参照してください。

次の表では、ドキュメントタイプ定義ファイル内で使用できるタグと属性について説明します。

タグ	属性	必須	説明
documenttype （ルート）		必須	親ノード。
	id	必須	すべてのドキュメントタイプ定義ファイルで固有の識別子。
	servermodel	オプション	関連付けられたサーバモデルを指定します。大文字と小文字が区別されます。デフォルトで有効な値は以下のとおりです。 ASP.NET C# ASP.NET VB ASP VBScript ASP JavaScript ColdFusion JSP PHP MySQL getServerModelDisplayName() 関数の呼び出しでは、これらの名前が返されます。サーバモデル実装ファイルは、Configuration¥ServerModels フォルダに格納されています。 拡張機能を開発する際は、このリストにはない新しいサーバモデルを作成できます。

タグ	属性	必須	説明
	internaltype	必須	Dreamweaver でのファイルの処理方法の広範な分類。internaltype では、デザインビューがこのドキュメントに対して有効であるかどうかが識別され、Dreamweaver テンプレートや拡張機能などの特殊な機能が処理されます。 有効な値は以下のとおりです。 Dynamic DWExtension（特殊な表示領域があります） DWTemplate（特殊な表示領域があります） HTML HTML4 Text（コードビューのみ） XHTML1 XML（コードビューのみ） すべてのサーバモデル関連のドキュメントタイプは、Dynamic にマップする必要があります。HTML は HTML にマップします。CSS、JS、VB、CS などのスクリプトファイルは、Text にマップします。 internaltype が DWTemplate の場合は、dynamicid を指定します。それ以外の場合、サーバビヘイビアパネルまたはバインディングパネルでは、新規ドキュメントダイアログボックスで作成される新しい空のテンプレートは認識されません。このテンプレートのインスタンスは、単なる HTML テンプレートのためです。
	dynamicid	オプション	動的ドキュメントタイプの固有の識別子への参照。この属性は、internaltype が DWTemplate の場合のみ有効です。この属性で、動的テンプレートを動的ドキュメントタイプに関連付けることができます。
	winfileextension	必須	Windows のドキュメントタイプと関連付けられたファイル拡張子。複数のファイル拡張子を指定するには、カンマで区切ったリストにします。リストの最初の拡張子は、ユーザが documenttype ドキュメントを保存したときに Dreamweaver によって使用される拡張子です。 サーバモデルに関連付けられていない 2 つのドキュメントタイプが同じファイル拡張子を持つ場合、最初のドキュメントタイプがその拡張子のドキュメントタイプとして認識されます。

タグ	属性	必須	説明
	macfileextension	必須	Macintosh のドキュメントタイプと関連付けられたファイル拡張子。複数のファイル拡張子を指定するには、カンマで区切ったリストにします。リストの最初の拡張子は、ユーザが documenttype ドキュメントを保存したときに Dreamweaver によって使用される拡張子です。 サーバモデルに関連付けられていない 2 つのドキュメントタイプが同じファイル拡張子を持つ場合、最初のドキュメントタイプがその拡張子のドキュメントタイプとして認識されます。
	previewfile	オプション	新規ドキュメントダイアログボックスのプレビュー領域にレンダリングされるファイル。
	file	必須	DocumentTypes¥NewDocuments フォルダにあるファイルで、新しい documenttype ドキュメントのテンプレートコンテンツを含みます。
	priorversionservermodel	オプション	このドキュメントのサーバモデルに Dreamweaver UltraDeveloper 4 と同等のものがある場合、その古いバージョンのサーバモデルの名前を指定します。 UltraDeveloper 4 ColdFusion は、有効な旧サーバモデルです。
title （サブタグ）		必須	新規ドキュメントダイアログボックスの「空白ドキュメント」のカテゴリー項目として表示されるストリング。このストリングは、定義ファイルに直接配置することも、ローカライズのために間接的に参照することもできます。このストリングのローカライズについて詳しくは、20 ページの「ローカライズされるストリングの提供」を参照してください。 フォーマットできません。したがって、HTML タグを指定することはできません。
description （サブタグ）		オプション	ドキュメントタイプを記述するストリング。このストリングは、定義ファイルに直接配置することも、ローカライズのために間接的に参照することもできます。このストリングのローカライズについて詳しくは、20 ページの「ローカライズされるストリングの提供」を参照してください。 フォーマットすることができ、HTML タグを指定できます。

注意：ユーザが新しいドキュメントを保存すると、Dreamweaver によって、そのドキュメントタイプと関連付けられた現在のプラットフォーム用の拡張子のリストが確認されます。例えば、winfileextension や macfileextension です。リストの最初のストリングが選択され、デフォルトのファイル拡張子として使用されます。このデフォルトのファイル拡張子を変更するには、カンマで区切られたリスト内の拡張子の順序を変更し、新しいデフォルトの拡張子がリストの先頭になるようにします。

Dreamweaver を起動すると、すべてのドキュメントタイプ定義ファイルが読み取られ、有効なドキュメントタイプのリストが作成されます。存在しないサーバモデルを含む定義ファイル内のすべてのエントリは、非サーバモデルドキュメントタイプとして処理されます。内容が不適切なエントリや固有ではない ID は無視されます。

ドキュメントタイプ定義ファイルが破損しているか、Configuration¥DocumentTypes フォルダで使用できない場合、エラーメッセージが表示され、Dreamweaver が終了します。

動的テンプレートの定義

動的ドキュメントタイプに基づいたテンプレートを作成できます。これらのテンプレートは、「動的テンプレート」と呼ばれます。動的テンプレートの定義には、以下の 2 つの要素が不可欠です。

- 新しいドキュメントタイプの internaltype 属性の値を DWTemplate にする必要があります。
- dynamicid 属性を設定し、値として既存の動的ドキュメントタイプの識別子への参照を指定する必要があります。

次の例では、動的ドキュメントタイプを定義しています。

```
<documenttype
    id="PHP_MySQL"
    servermodel="PHP MySQL"
    internaltype="Dynamic"
    winfileextension="php,php3"
    macfileextension="php,php3"
    file="Default.php">
    <title>PHP</title>
    <description><![CDATA[PHP document]]></description>
</documenttype>
```

次に、この PHP_MySQL 動的ドキュメントタイプに基づいた、以下の動的テンプレートを定義できます。

```
<documenttype
    id="DWTemplate_PHP"
    internaltype="DWTemplate"
    dynamicid="PHP_MySQL"
    winfileextension="php.dwt"
    macfileextension="php.dwt"
    file="Default.php.dwt">
    <title>PHP Template</title>
    <description><![CDATA[Dreamweaver PHP Template document]]></description>
</documenttype>
```

Dreamweaver を使用して、DWTemplate_PHP タイプの新しい空のテンプレートを作成すると、ユーザはファイルに PHP サーバビヘイビアを作成できます。また、新しいテンプレートのインスタンスを作成する際に、そのインスタンスに PHP サーバビヘイビアを作成できます。

前の例では、ユーザがテンプレートを保存すると、.php.dwt 拡張子がファイルに自動的に追加されます。ユーザがテンプレートのインスタンスを保存すると、Dreamweaver によって .php 拡張子がファイルに追加されます。

ドキュメント拡張子とファイルタイプの追加および変更

デフォルトでは、Dreamweaver が対応するすべてのファイルタイプがファイル／開くを選択して表示されるダイアログボックスに表示されます。ドキュメントタイプを作成したら、デベロッパーは、該当する Extensions.txt ファイルを更新する必要があります。ユーザがマルチユーザシステム（Windows XP、Windows Vista、Mac OS X など）を使用している場合もあります。その場合は、別の Extensions.txt ファイルがユーザの Configuration フォルダにあります。Extensions.txt ファイルは、Dreamweaver によって検索および解析されるインスタンスであるため、ユーザ側で更新する必要があります。

ユーザの Configuration フォルダの場所は、ユーザのプラットフォームによって異なります。

Windows XP プラットフォームでは次の場所を使用します。

ハードディスク **:**¥Documents and Settings¥ ユーザ名 ¥Application Data¥Adobe¥Dreamweaver CS4¥Configuration

注意：このフォルダは、隠しフォルダ内部にある可能性があります。

Windows Vista プラットフォームでは次の場所を使用します。

ハードディスク **:**¥Users¥ ユーザ名 ¥AppData¥Roaming¥Adobe¥Dreamweaver CS4¥Configuration

Mac OS X プラットフォームでは次の場所を使用します。

ハードディスク :Users/ ユーザ名 /Library/Application Support/Adobe/Dreamweaver CS4/Configuration

Dreamweaver がユーザの Configuration フォルダで Extensions.txt ファイルを見つけられない場合は、Dreamweaver の Configuration フォルダが検索されます。

注意 : マルチユーザプラットフォームでは、Dreamweaver は Dreamweaver の Configuration フォルダではなく、ユーザの Configuration フォルダにある Extensions.txt ファイルのコピーを解析します。したがって、Dreamweaver の Configuration フォルダにある Extensions.txt のコピーを編集しても、Dreamweaver では認識されません。

ドキュメント拡張子を作成するには、新しい拡張子を既存のドキュメントタイプに追加するか、ドキュメントタイプを作成します。

新しい拡張子を既存のドキュメントタイプに追加する

1. MMDocumentTypes.xml を編集します。
2. 既存のドキュメントタイプの winfileextension および macfileextension 属性に、新しい拡張子を追加します。

新しいドキュメントタイプを追加する

1. Configuration フォルダ内の Extensions.txt ファイルのバックアップコピーを作成します。
2. テキストエディタで Extensions.txt を開きます。
3. 新規ファイルタイプごとに新しい行を追加します。新しいファイルタイプに使用するファイル拡張子を大文字で入力して、それぞれカンマで区切ります。次に、コロンを入力し、短い説明を追加します。この説明は、ファイルタイプのポップアップメニューに表示されます。ファイル／開くを選択して表示されるダイアログボックスに、ポップアップメニューが表示されます。

   例えば、JPEG ファイルの場合は、「JPG,JPEG,JFIF:JPEG Image Files」と入力します。
4. Extensions.txt ファイルを保存します。
5. Dreamweaver を再起動します。

変更を確認するには、ファイル／開くを選択してファイルタイプのポップアップメニューをクリックします。

Dreamweaver のファイル／開くで表示されるデフォルトのファイルタイプを変更する

1. Configuration フォルダ内の Extensions.txt ファイルのバックアップコピーを作成します。
2. テキストエディタで Extensions.txt を開きます。
3. 新しいデフォルトに対応する行をカットします。次に、その行がファイルの最初の行になるように、ファイルの先頭にペーストします。
4. Extensions.txt ファイルを保存します。
5. Dreamweaver を再起動します。

変更を確認するには、ファイル／開くを選択してファイルタイプのポップアップメニューをクリックします。

関連項目

http://www.adobe.com/go/16410_jp

ローカライズされるストリングの提供

ドキュメントタイプ定義ファイル内では、<title> および <description> サブタグにより、表示されるタイトルとドキュメントタイプの説明が指定されます。サブタグで、2 つのサブタグのローカライズされるストリングを提供するためのプレースホルダとして、MMString:loadstring ディレクティブを使用できます。この処理は、プレースホルダとしてストリング識別子を使用し、ページで使用する特定のストリングを指定するサーバサイドスクリプトに似ています。プレースホルダには、特殊なタグを使用するか、値が置換されるタグ属性を指定できます。

ローカライズされるストリングを提供する

1 ドキュメントタイプ定義ファイルの先頭に、以下のステートメントを置きます。

```
<?xml version="1.0" encoding="utf-8"?>
```

2 MMString ネームスペースを <documenttypes> タグで宣言します。

```
<documenttypes
    xmlns:MMString="http://www.adobe.com/schemes/data/string/">
```

3 ドキュメントタイプ定義ファイル内のローカライズされるストリングを提供する位置で、MMString:loadstring ディレクティブを使用して、ローカライズされるストリングのプレースホルダを定義します。このプレースホルダは、次の方法のいずれかで指定できます。

```
<description>
    <loadstring>myJSPDocType/Description</loadstring>
</description>
```

または

```
<description>
    <loadstring id="myJSPDocType/Description" />
</description>
```

これらの例では、myJSPDocType/Description は、ローカライズされるストリングのプレースホルダとして機能する固有のストリング識別子です。ローカライズされるストリングは、次の手順で定義します。

4 Configuration¥Strings フォルダに、ローカライズされるストリングを定義する新しい XML ファイルを作成します。または既存のファイルを編集することもできます。例えば、以下のコードを Configuration/Strings/strings.xml ファイルに配置すると、myJSPDocType/Description ストリングが定義されます。

```
<strings>
...
    <string id="myJSPDocType/Description"
        value=
        "<![CDATA[JavaServer Page with <em>special</em> features]]>"
    />
...
</strings>
```

注意：上記の例の myJSPDocType/Description などのストリング識別子は、アプリケーション内で固有である必要があります。Dreamweaver を起動すると、Configuration¥Strings フォルダ内のすべての XML ファイルが解析され、これらの固有のストリングが読み込まれます。

ドキュメントタイプ定義ファイルにおけるルール

Dreamweaver では、1 つのサーバモデルに関連付けられている複数のドキュメントタイプで、ファイル拡張子を共有することができます。ASP-JS と ASP-VB で、.asp をそのファイル拡張子として要求することができます。どのサーバモデルが優先されるかについては、315 ページの「canRecognizeDocument()」を参照してください。

Dreamweaver では、サーバモデルに関連付けられていないドキュメントタイプが、ファイル拡張子を共有することはできません。

2 つのドキュメントタイプにより同一のファイル拡張子が要求され、そのうちの 1 つのタイプはサーバモデルに関連付けられていて、もう 1 つのタイプはサーバモデルに関連付けられていない場合は、関連付けられていないドキュメントタイプが優先されます。ファイル拡張子 .sam を使用している、サーバモデルに関連付けられていない SAM というドキュメントタイプがあり、このファイル拡張子を ASP-JS ドキュメントタイプに追加するとします。Dreamweaver ユーザが、拡張子が .sam のファイルを開くと、Dreamweaver によって、ASP-JS ではなく SAM ドキュメントタイプがそのファイルに割り当てられます。

Dreamweaver でドキュメントを開く

ユーザがドキュメントファイルを開くと、Dreamweaver では、一連の手順に従って、ファイル拡張子に基づいてドキュメントタイプが識別されます。

固有のドキュメントタイプが見つかった場合は、そのタイプが使用され、ユーザが開こうとしているドキュメントに関連付けられているサーバモデルがあれば、それが読み込まれます。ユーザが Dreamweaver UltraDeveloper4 サーバビヘイビアの使用を選択した場合は、Dreamweaver により、該当する UltraDeveloper4 サーバモデルが読み込まれます。

ファイル拡張子が複数のドキュメントタイプにマップされている場合は、以下のアクションが実行されます。

- 静的ドキュメントタイプがドキュメントタイプのリストにある場合、それが優先される。
- すべてのドキュメントタイプが動的な場合は、これらのドキュメントタイプと関連付けられたサーバモデルのアルファベット順のリストが作成され、各サーバモデルで canRecognizeDocument() 関数が呼び出される（詳しくは、315 ページの「canRecognizeDocument()」を参照してください）。Dreamweaver によって、戻り値が収集され、最も大きい正の整数を返したサーバモデルが判別されます。関連付けられたサーバモデルから最も大きい整数が返されたドキュメントタイプが、開こうとしているドキュメントに割り当てられるドキュメントタイプになります。ただし、複数のサーバモデルから同じ整数が返された場合、これらのサーバモデルのアルファベット順のリストが検索され、リストで最初に検出されたドキュメントタイプが使用されます。例えば、ASP-JS と ASP-VB の両方が ASP ドキュメントを要求し、それぞれの canRecognizeDocument() 関数により同じ値が返された場合、ASP-JS の方がアルファベット順で前になるため、そのドキュメントは ASP-JS に割り当てられます。

Dreamweaver がファイルの拡張子をドキュメントタイプにマップできない場合、ドキュメントはテキストファイルとして開かれます。

ワークスペースレイアウトのカスタマイズ

Dreamweaver では、ワークスペースのレイアウトをカスタマイズすることができます。例えば、指定したレイアウトに表示するパネル、パネルの位置とサイズ、折りたたみまたは展開の状態、アプリケーションウィンドウの位置とサイズ、ドキュメントウィンドウの位置とサイズなどをカスタマイズすることができます。

ワークスペースのレイアウトは、設定またはワークスペースのレイアウトフォルダにある XML ファイルで指定します。次の節では、XML タグのシンタックスについて説明します。オプションの属性は、属性リスト内に波カッコ（{}）で囲まれて示されます。波カッコで囲まれていない属性はすべて必須です。

<panelset>

説明

パネルセットの説明の始まりを示す一番外側のタグ。

属性

なし。

コンテンツ

このタグには少なくとも 1 つの application、document または panelset タグが含まれています。

コンテナ
なし。

例
```
<panelset>
    <!-- panelset tags here -->
</panelset>
```

<application>

説明
アプリケーションウィンドウの最初の位置とサイズを指定します。

属性
rect、maximize

- rect はアプリケーションウィンドウの位置とサイズを指定します。このストリングは、「左 上 右 下」の形式で整数で指定します。
- maximize はブール値です。起動時にアプリケーションウィンドウを最大化する場合は true、それ以外の場合は false です。デフォルト値は true です。

コンテンツ
なし。

コンテナ
このタグは、panelset タグに含める必要があります。

例
```
<panelset>
    <application rect="0 0 1000 1200" maximize="false">
    </application>
</panelset>
```

<document>

説明
ドキュメントウィンドウの最初の位置とサイズを指定します。

属性
rect、maximize

- rect はドキュメントウィンドウの位置とサイズを指定します。このストリングは、「左 上 右 下」の形式で整数で指定します。maximize 値が true の場合、rect 値は無視されます。
- maximize はブール値です。起動時にドキュメントウィンドウを最大化する場合は true、それ以外の場合は false です。デフォルト値は true です。

コンテンツ
なし。

コンテナ
このタグは、panelset タグに含める必要があります。

例
```
<panelset>
    <document rect="100 257 1043 1200" maximize="false">
    </document>
</panelset>
```

<panelframe>

説明
パネルグループ全体について記述します。

属性
x、y、{width, height}、dock、collapse

- x は、パネルグループの左揃えの位置を指定します。この値は、整数または画面に対して相対的な値にすることができます。整数値が画面上にない場合、パネルグループは、画面上に表示可能な最も整数に近い位置に表示されます。相対的な値として「left」または「right」を指定できます。これらの値は、仮想画面のどちらの端にパネルグループのどちらの端を揃えるかを示します。
- y は、パネルグループの上揃えの位置を指定します。この値は、整数または画面に対して相対的な値にすることができます。整数値が画面上にない場合、パネルグループは、画面上に表示可能な最も整数に近い位置に表示されます。相対的な値として「top」または「bottom」を指定できます。これらの値は、仮想画面のどちらの端にパネルグループのどちらの端を揃えるかを示します。
- width は、パネルグループの幅のピクセル値です。この属性はオプションです。width を指定しないと、パネルグループの組み込みのデフォルトが使用されます。
- height は、パネルグループの高さのピクセル値です。この属性はオプションです。height を指定しないと、パネルグループの組み込みのデフォルトが使用されます。
- dock は、アプリケーションフレームのどちらの端をパネルグループにドッキングするかを指定するストリング値です。Macintosh ではパネルグループをドッキングできないので、この属性は無視されます。
- collapse はブール値です。true は、パネルグループが折りたたみ状態であることを示し、false は、パネルグループが展開状態であることを示します。Macintosh ではパネルはフローティング状態になるので、この属性は無視されます。

コンテンツ
このタグには、少なくとも 1 つの panelcontainer タグを含める必要があります。

コンテナ
このタグは、panelset タグに含める必要があります。

例
```
<panelset>
    <panelframe rect="196 453 661 987" visible="true" dock="floating">
        <!-- panelcontainer tags here -->
    </panelframe>
</panelset>
```

<panelcontainer>

説明
パネルグループ全体について記述します。

属性
expanded、title、{ height,} activepanel、visible、maximize、maxRestorePanel、maxRestoreIndex、maxRect、tabsinheader

- expanded はブール値です。パネルが展開状態の場合は true、それ以外の場合は false です。
- title は、パネルのタイトルを指定するストリング値です。
- height はパネルの高さのピクセル値を示す整数値です。この属性はオプションです。height を指定しない場合、各パネルの組み込みデフォルトが使用されます。

  注意：Width は、親から継承されます。
- activepanel は、一番前面にあるパネルの ID を示す数値です。
- visible はブール値です。パネルが表示されている場合は true、それ以外の場合は false です。
- maximize はブール値です。最初に表示するときにパネルを最大化する場合は true、それ以外の場合は false です。
- maxRestorePanel は、復元するパネルの ID を示す数値です。
- maxRect は、パネルを最大化したときの位置およびサイズを示すストリング値です。このストリングは、「左 上 右 下」の形式で整数で指定します。
- tabsinheader はブール値です。タブをヘッダーバーの下ではなく、ヘッダー内に配置する場合は true、それ以外の場合は false です。

コンテンツ
このタグには、少なくとも 1 つの panel タグを含める必要があります。

コンテナ
このタグは、panelframe タグに含める必要があります。

例

```
<panelset>
    <panelframe rect="196 453 661 987" visible="true" dock="floating">
        <panelcontainer title="Color" height="250" visible="true" expanded="true"
        activepanel="20">
            <!-- panel tags here --->
        </panelcontainer>
    </panelframe>
</panelset>
```

<panel>

説明
パネルコンテナ内に表示されるパネルを指定します。

属性
id、visibleTab

- id は、パネルの ID を示す数値です。次の表に値のリストを示します。

本ソフトウェア	ID	パネル
Adobe® Flash®	1	プロパティ
	2	アクション
	3	行揃え
	4	ビヘイビア
	5	コンポーネント
	6	コンポーネントインスペクタ
	7	カラーミキサー
	8	色見本
	9	ヒストリ
	10	情報
	11	ライブラリ
	12	ムービーエクスプローラ
	13	出力
	14	プロパティ
	15	プロジェクト
	16	変形
	17	シーン
	18	ストリング
	19	デバッガ
	101–110	ライブラリ
Dreamweaver	1	プロパティ
Flex Builder	1	プロパティ

- visibleTab はブール値です。タブとパネルを表示する場合は true、それ以外の場合は false です。

コンテンツ
なし。

コンテナ
このタグは、panelcontainer タグに含める必要があります。

例

```
<panelset>
    <panelframe rect="196 453 661 987" visible="true" dock="floating">
        <panelcontainer title="Color" height="250" visible="true" expanded="true"
        activepanel="20">
            <panel id="20"></panel>
        </panelcontainer>
    </panelframe>
</panelset>
```

コーディングツールバーのカスタマイズ

コーディングツールバーは、初期状態で 15 のボタンを表示します。これは、利用可能なボタンのサブセットです。コーディングツールバーは、Configuration/Toolbars/Toolbars.xml ファイルを編集してツールバーに表示されるボタンやボタンの表示順序を変更することによりカスタマイズできます。Extension Manager を使用して、ユーザ固有のボタンをツールバーに挿入することもできます。

ボタンの順序を変更する

1 Configuration/Toolbars/toolbars.xml ファイルを開きます。

2 次のコメントを探して、コードビューのツールバーのセクションを見つけます。

```
<!-- Code view toolbar -->
```

3 ツールバー上に表示する順序になるようにボタンのタグをコピーしてペーストします。

4 ファイルを保存します。

ボタンを削除する

1 Configuration/Toolbars/toolbars.xml ファイルを開きます。

2 次のコメントを検索することにより、コーディングツールバーのセクションの位置を特定します。

```
<!-- Code view toolbar -->
```

3 削除するボタンをコメントで囲みます。

次の例では、ツールバーに表示されないようにコメントで囲まれたボタンを示します。

```
<!-- remove button from Coding toolbar
    <button id="DW_ExpandAll"
        image="Toolbars/images/MM/T_ExpandAll_Sm_N.png"
        disabledImage="Toolbars/images/MM/T_ExpandAll_Sm_D.png"
        tooltip="Expand All"
        domRequired="false"
        enabled="dw.getFocus(true) == 'textView' || dw.getFocus(true) == 'html'¬
            "command="if (dw.getFocus(true) == 'textView' || dw.getFocus(true) ¬
            == 'html') dw.getDocumentDOM().source.expandAllCodeFragments();"
        update="onViewChange" />
-->
```

4 ファイルを保存します。

ツールバーに表示されていないボタンを表示するには、XML ファイル内でボタンを囲んでいるコメントを削除します。

キーボードショートカットマッピングの変更

Dreamweaver には、Dreamweaver の機能へのキーボードショートカットが多数用意されています。デフォルトのキーボードショートカットは menus.xml ファイルに一覧表示され、US キーボード用に作成されています。Dreamweaver では多数のショートカットが提供されているので、非英数字のショートカット（a ～ z と 0 ～ 9 以外の文字）は他国語キーボード用にマッピングし直す必要があります。そのため、Dreamweaver は、他国語キーボード用のキーボードショートカットマッピングを定義する XML ファイルを多数備えています。これらのファイルは Configuration¥Menus¥Adaptive Sets フォルダにあります。Dreamweaver では、コンピュータに接続されている他国語キーボードを認識すると、キーボードショートカットをそのキーボード用のマッピングファイルに自動的にリセットします。キーボードレイアウトに使用できる適切なファイルが存在しない場合、そのキーボードレイアウトで機能しないショートカットは削除されます。

キーボードショートカットのマッピングファイルには、該当するキーボードレイアウトの 2 文字の言語コードを使用した名前が付けられています。例えば、ドイツ語のキーボードレイアウトは de.xml です。1 つの言語で国によって異なるキーボードレイアウトがある場合は、マッピングファイルの名前として 2 文字の言語コードの後にハイフン（-）と 2 文字の国コードが使用されています。例えば、fr-ca.xml はカナダフランス語キーボードレイアウト用のファイル名です。2 文字の言語コードは ISO 639（http://en.wikipedia.org/wiki/List_of_ISO_639_codes）に定義されています。国コードは ISO 3166（http://en.wikipedia.org/wiki/ISO_3166-1_alpha-2）に定義されています。

コンピュータでアクティブなキーボード言語設定が変更されると、Dreamweaver は、そのキーボードの国および言語に対して適切なキーボードショートカットのマッピングファイルが存在するかどうかをチェックします。最初に国用のマッピングファイルがチェックされ、そのファイルが存在しない場合は、その国の言語のファイルがチェックされます。例えば、コンピュータにカナダフランス語のキーボードを接続している場合、最初にカナダフランス語のキーボードレイアウト用の fr-ca.xml が検索されます。そのファイルが存在しない場合、fr.xml が検索されます。次の表は Dreamweaver で提供されるマッピングファイルのリストです。

ファイル名	Windows プラットフォーム	Macintosh プラットフォーム
ca.xml	カタロニア語	カタロニアスペイン語
de.xml	ドイツ語（ドイツ、オーストリア）	オーストリア語、ドイツ語
de-ch.xml	ドイツ語（スイス）	スイスドイツ語
es.xml	スペイン語（International Sort）、スペイン語（Traditional Sort）	スペイン語 - ISO
fr.xml	フランス語（フランス）	フランス語
fr-ca.xml	フランス語（カナダ）	カナダ語 - CSA
fr-ch.xml	フランス語（スイス）	スイスフランス語
it.xml	イタリア語（イタリア）、イタリア語（スイス）	イタリア語 - Pro
it-mac.xml	該当なし	イタリア語
ja.xml	日本語	日本語
nl-be.xml	オランダ語（ベルギー）、フランス語（ベルギー）	ベルギー語
zh-cn.xml	中国語（PRC）、中国語（シンガポール）	簡体字中国語

Dreamweaver が提供するキーボードレイアウト以外のキーボードレイアウトを使用している場合、そのキーボード用のマッピングファイルを作成して、Configuration¥Menus¥Adaptive Sets フォルダに配置できます。

キーボードマッピングファイルを作成する

1 Configuration¥Menus¥Adaptive Sets フォルダにあるキーボードマッピングファイルの 1 つをコピーし、そのファイルに、使用しているキーボードに該当する 2 文字の言語名を使用した、拡張子が .xml の名前を付けます。

既存のファイルをコピーする場合、使用しているキーボードレイアウトに近いキーボードマッピングファイルをコピーします。例えば、スウェーデン語キーボードのキーボードマッピングファイルを作成する場合、スウェーデン語キーボードはドイツ語キーボードレイアウトに似ているので、de.xml をコピーします。

2 作成した新しいキーボードマッピングファイルを、Configuration¥Menus¥Adaptive Sets フォルダ以外のフォルダに移動します。

3 Dreamweaver でキーボードマッピングファイルを開きます。

4 お使いのキーボードレイアウト用の shortcut タグを削除または追加します。

変更するキーボード用の shortcut タグを決める際に参考として、US キーボードのショートカットのセットと使用している言語のショートカットのセットを比較できます。（次の手順では、2 種類のキーボードレイアウトのショートカットを比較します）。

5 キーボードショートカットを変更した後、ファイルを保存して Configuration¥Menus¥Adaptive Sets に再度移動します。

変更するキーボード用の shortcut タグを決定する

1 まだ設定していない場合、コンピュータのアクティブなキーボード言語設定を、使用する言語に切り替えます。この設定は、コンピュータのオペレーティングシステムで行います。例えば、Windows ではコントロールパネルで言語を選択できます。

2 編集／キーボードショートカットを選択して Dreamweaver キーボードショートカットエディタを起動します。

3 ダイアログボックスの右上にある 3 番目のイメージボタンをクリックします。ボタンの上にポインタを置くと、「セットを HTML として書き出す」というツールヒントが表示されます。

HTML ファイルとして保存ダイアログが表示され、現在のキーボードレイアウトのキーボードショートカットサマリーファイルの名前を入力するよう要求されます。

4 サマリーファイルを保存して、キーボードショートカットエディタダイアログボックスを閉じます。

5 キーボードレイアウトを US キーボードに切り替えます（コンピュータのオペレーティングシステムを使用）。

6 編集／キーボードショートカットを選択して Dreamweaver キーボードショートカットエディタを起動します。

7 ダイアログボックスの右上にある 3 番目のイメージボタンをクリックして、設定を HTML ファイルとして書き出します。

8 サマリーファイルを保存して、キーボードショートカットエディタダイアログボックスを閉じます。

9 これで 2 つのキーボードショートカットサマリーファイルを印刷して比較し、お使いの言語キーボードレイアウト用に削除されたすべてのショートカットを確認できます。これらのショートカットには、使用するキーボードレイアウト上でのみ使用可能なキーを使用して新しいショートカットの再割り当てを行う必要があります。

2 つのファイルの比較により得られた情報にもとづいて、再割り当てを行う各ショートカットに対して shortcut タグを追加または削除することにより、キーボードレイアウトマッピングファイルを更新できます。

キーボードレイアウトマッピング XML ファイル

次の例では、フランス語キーボードマッピングファイル（fr.xml）のフォーマットを示します。

```
<shortcutset language="French">
<shortcut key="Cmd+[" newkey="Cmd+&ugrave;"/>
<shortcut key="Cmd+]" newkey="Cmd+)"/>
<shortcut platform="win" key="Cmd+Shift+>" newkey="Cmd+Opt+Shift+,"/>
<shortcut platform="mac" key="Cmd+Shift+>" newkey="Cmd+<"/>
<shortcut platform="win" key="Cmd+Shift+<" newkey="Cmd+Shift+,"/>
<shortcut key="Cmd+'" newkey="Cmd+Shift+=" />
...
</shortcutset>
```

キーボードレイアウト XML ファイルのシンタックスは、次のとおりです。

```
<shortcutset language="language_name">
<shortcut key="key_combination" newkey="key_combination"/>
<shortcut platform="op_system" key="key_combination" newkey="key_combination"/>
</shortcutset>
```

このコードでは：

- **language_name** は、フランス語、スペイン語、ドイツ語などのキーボードの言語を示します。
- **key_combination** は、Cmd+[、Cmd+Shift+>、Ctrl+$ などのキーボードショートカットを示します。
- key は、置き換えられるキーボードショートカットを示します。
- newkey は、key を置き換えるキーボードショートカットを示します。
- platform=**op_system** は、ショートカットを適用するシステムを示します。win または mac を指定します。プラットフォームを指定しない場合、ショートカットは両方のプラットフォームに適用されます。

第 3 章：コードビューのカスタマイズ

Adobe Dreamweaver のコードビューでは、読みやすく正確なコードをすばやく入力するために 2 つのツールを使用します。この 2 つのツールは、コードヒントとコードカラーリングです。また、Dreamweaver では、指定したターゲットブラウザにコードが適合しているかどうかが検証され、デフォルトの HTML フォーマットを変更することができます。

コードヒントとコードカラーリングをカスタマイズするには、これらを実装する XML ファイルを修正します。コードヒントのメニューに項目を追加するには、CodeHints.xml ファイルまたは SpryCodeHints.xml ファイルにエントリを追加します。カラースキームを修正するには、コードカラーリングのスタイルファイルである Colors.xml を修正します。コードカラーリングのスキームを変更または新しく追加するには、CodeColoring.xml などのコードカラーリングのシンタックスファイルを修正します。また、Cascading Style Sheet（CSS）プロファイルファイルをターゲットブラウザに合わせて修正して、Dreamweaver で CSS プロパティと値を検証する方法を変更できます。Dreamweaver のデフォルトの HTML フォーマットは、環境設定ダイアログボックスで変更することもできます。次の項では、これらの機能をカスタマイズする方法を説明します。

コードヒントについて

コードヒントは、コードビューで特定の文字パターンを入力したときに Dreamweaver によって表示されるメニューです。コードヒントによって、入力しようとしているストリングの候補のリストが提示されるため、入力の手間を省くことができます。入力しようとしているストリングがメニューに表示されている場合は、そのストリングまでスクロールして Enter キーまたは Return キーを押すと、入力が完了します。例えば、「<」を入力すると、ポップアップメニューにタグ名のリストが表示されます。タグ名の残り部分を入力する代わりに、メニューからタグを選択してテキストに追加することができます。また、Dreamweaver は、Spry フレームワークのコードヒントを搭載しています。

Dreamweaver のコードヒントメニューは、Configuration¥CodeHints フォルダ内にある CodeHints.xml ファイルおよびその他の XML ファイルから読み込まれます。Dreamweaver にコードヒントメニューを追加するには、この項で説明する XML スキーマフォーマットを使用して、ユーザ独自の XML ファイルでこれらを定義し、Configuration¥CodeHints フォルダ内に配置します。

Dreamweaver にコードヒントファイルのコンテンツが読み込まれた後に、JavaScript を使用して動的に新しいコードヒントメニューを追加することもできます。例えば、JavaScript のコードによって、バインディングパネルのセッション変数リストの項目を設定します。同じコードを使用してコードヒントメニューを追加すると、ユーザがコードビューで **"Session."** と入力したときに、セッション変数のメニューが表示されます。JavaScript を使用したコードヒントメニューの追加または修正について詳しくは、『**Dreamweaver API** リファレンス』の「コード関数」を参照してください。

コードヒントメニューの種類によっては、XML ファイルや JavaScript API を使用して表示できないものがあります。CodeHints.xml ファイル、SpryCodeHints.xml ファイル、および JavaScript API はコードヒントエンジンの便利なサブセットを備えていますが、Dreamweaver の機能の一部は利用できません。例えば、カラーピッカーを表示する JavaScript フックは存在しないため、JavaScript を使用して属性値メニューを表すことはできません。実行できるのは、テキスト項目のメニューを表示し、そのメニューからテキストを挿入することだけです。

注意：テキストを挿入すると、挿入ポイントは挿入されたストリングの後に移動します。

CodeHints.xml ファイル

CodeHints.xml ファイルには、次のエンティティが含まれます。

- すべてのメニューグループのリスト

  環境設定ダイアログボックスでコードヒントカテゴリーを選択すると、このメニューグループのリストが表示されます。環境設定ダイアログボックスを開くには、編集／環境設定を選択します。Dreamweaver には、タグ名、属性名、属性値、関数の引数、オブジェクトメソッドと変数、HTML エンティティというコードヒントメニューのメニューグループまたはタイプが用意されています。

- 各メニューグループの説明

  コードヒントカテゴリーの環境設定ダイアログボックスでリストからメニューグループを選択すると、説明が表示されます。選択したエントリの説明は、メニューグループリストの下に表示されます。

- コードヒントメニュー

  メニューは、コードヒントメニューを起動するパターンと、コマンドのリストで構成されています。例えば、& というパターンでは、&、>、< などのメニューが起動されます。

  次の例に、CodeHints.xml ファイルのフォーマットを示します（太字のタグについては、35 ページの「コードヒントのタグ」を参照してください）。

```
<codehints>
<menugroup name="HTML Entities" enabled="true" id="CodeHints_HTML_Entities">
    <description>
    <![CDATA[ When you type a '&', a pop-up menu shows
        a list of HTML entities. The list of HTML entities
        is stored in Configuration/CodeHints.xml. ]]>
    </description>
    <menu pattern="&amp;amp;">
            <menuitem value="&amp;amp;amp;" texticon="&amp;amp;"/>
            <menuitem value="&amp;amp;lt;" icon="lessThan.gif"/>
    </menu>
</menugroup>

<menugroup name="Tag Names" enabled="true" id="CodeHints_Tag_Names">
    <description>
            <![CDATA[ When you type '<', a pop-up menu shows
            all possible tag names.0You can edit the list of tag
            names using the
            <a href="javascript:dw.popupTagLibraryEditor()"> Tag Library
                 Editor </a>]]>
        </description>
</menugroup>

<menugroup name="Function Arguments" enabled="true"
        id="CodeHints_Function_Arguments">
    <description>
        ...
    </description>
    <function pattern="ArraySort(array, sort_type, sort_order)"
            doctypes="CFML"/>
    <function pattern="Response.addCookie(Cookie cookie)"
            doctypes="JSP"/>
</menugroup>
</codehints>
```

JavaScript コードヒント

Dreamweaver は、Spry フレームワークのコードヒントをサポートしています。Spry コードヒントファイル（SpryCodeHints.xml）の基本的なフォーマットは、CodeHints.xml ファイルと同じです。Spry コードヒントファイルでは、method などの新しいキーワードのほか、クラスメンバーリストをクラス（Spry.Data.XMLDataSet など）に関連付けるための新しい属性 classpattern を使用します。クラスのクラスメンバーリストは、メニュー（メソッド、プロパティ、およびイベント）内にネストされます。

<method> タグとその属性は、function タグとその属性に似ています。ただし、関連付けを行うためには、親の menu タグが classpattern 属性を持っている必要があります。また、プロパティ用の property タグと、イベント用の event タグがあります。これらのタグは、対応するアイコンでコードヒントポップアップメニューに表示されます。さらに、パラメータヒントをサポートするための parammenu タグと parammenuitem タグがあります。

次の例に、SpryCodeHints.xml ファイルのフォーマットを示します（太字のタグについては、35 ページの「コードヒントのタグ」を参照してください）。

```
<function pattern="XMLDataSet(xmlsource, xpath, {options})"
    caseSensitive="true" />
<menu classpattern="Spry.Data.XMLDataSet">
    <property pattern="url" icon="shared/mm/images/hintMisc.gif" />
    <property pattern="xpath" icon="shared/mm/images/hintMisc.gif" />
    ...
    ...
    <method pattern="getData()" icon="shared/mm/images/hintMisc.gif" />
    <method pattern="getData()" icon="shared/mm/images/hintMisc.gif" />
    <method pattern="loadData()" icon="shared/mm/images/hintMisc.gif" />
    <method pattern="getCurrentRow()" icon= ".../hintMisc.gif" />
    <method pattern="setCurrentRow(rowID)" icon= ".../hintMisc.gif" />
    <method pattern="setCurrentRowNumber(rowNumber)" icon= ".../hintMisc.gif" />
    <method pattern="getRowNumber(rowObj)" icon= ".../hintMisc.gif" />
    <method pattern="setColumnType(columnName, columnType)" icon= ".../hintMisc.gif" />
        <parammenu pattern='"' name="columnName" index="0" type="spryDataReferences">
        </parammenu>
        <parammenu pattern='"' name="columnType" index="1" type="enumerated">
            <parammenuitem label="integer" value="integer" icon=".../hintMisc.gif"/>
            <parammenuitem label="image" value="image" icon=".../hintMisc.gif"/>
            <parammenuitem label="date" value="date" icon=".../hintMisc.gif"/>
            <parammenuitem label="string" value="string" icon=".../hintMisc.gif"/>
        </parammenu>
    </method>
    <method pattern="getColumnType(columnName)" icon= ".../hintMisc.gif" />
    <method pattern="distinct()" icon= ".../hintMisc.gif" />
    <method pattern="getSortColumn()" icon= ".../hintMisc.gif" />
    <method pattern="sort()" icon= ".../hintMisc.gif" />
     ...
     ...
    <event pattern="onCurrentRowChanged" icon= ".../hintMisc.gif" />
    <event pattern="onDataChanged" icon= ".../hintMisc.gif" />
     ...
     ...
</menu>
<function pattern="Accordion(element,{options})" caseSensitive="true" />
<menu classpattern="Spry.Widget.Accordion">
    <method pattern="openNextPanel()" icon= ".../hintMisc.gif" />
    <method pattern="openPreviousPanel()" icon= ".../hintMisc.gif" />
```

```
    <method pattern="openFirstPanel()" icon= ".../hintMisc.gif" />
     ...
     ...
</menu>
</function>
<function pattern="XMLDataSet(xmlsource, xpath, {options})" caseSensitive="true">
    <parammenu pattern='{,' name="options" index="2" type="optionArray"
                allowwhitespaceprefix="true">
        <parammenuitem label="sortOnLoad" value="sortOnLoad:"
                icon="shared/mm/images/hintMisc.gif" datatype="string"/>
            <optionparammenu pattern="sortOrderOnLoad" label="sortOrderOnLoad"
                    value="sortOrderOnLoad:" icon="shared/mm/images/hintMisc.gif"
                    type="enumerated" datatype="string">
                <optionparammenuitem label="ascending" value="ascending"
                    icon="shared/mm/images/hintMisc.gif"/>
                <optionparammenuitem label="descending" value="descending"
                    icon="shared/mm/images/hintMisc.gif"/>
            </optionparammenu>
    </parammenu>
</function>
```

クラスの宣言

次のフォーマットは、クラスに変数を関連付けることによりクラスを宣言します。

```
    <variablename>[space][= operator][new keyword][space]<classname>
```

以下に例を挙げます。

```
    var dsFoo = new Spry.Data.XMLDataSet("products.xml", "products/product");
    var fooAccordion = new Spry.Widget.Accordion("accordionDivId");
```

Spry.XML.DataSet クラス名の場合、ColorCoding.xml ファイルでこれを再宣言する必要があります。こうすることにより、カラーリングステートエンジンは、このクラスがクラスのインスタンスであると認識し、宣言の左側で定義される変数名を取得して、そのページで対応するクラスタイプとともに変数のリストに保存します（上記の例では、変数 fooAccordion とクラスタイプ Spry.Widget.Accordion)。

CodeColoring.xml でクラス名を再宣言するシンタックスは、次のようになります。

```
<classlibrary name="Spry Keywords" id="CodeColor_JavascriptSpry">
    <class>Spry.Data.XMLDataSet</class>
    <class>Spry.Widget.Accordion</class>
</classlibrary>
```

このシンタックスでは :

- classlibrary は、クラスをカラー id "CodeColor_JavascriptSpry" にグループ化するための新しいタグです。

  class は、クラスライブラリで利用できる個々のクラスをリストするために使用されます。クラスのリストを拡張すると、別の Spry パッケージ（Debug、Data、Util、Widget、Effect など)、またはその他の Ajax（Asynchronous JavaScript and XML）ツールキットや JavaScript ライブラリから他の Spry クラスを含めることができます。

Crosstag 属性コードヒント

Dreamweaver では、Spry の属性名および属性値に対してコードヒントを利用できます。これらの属性は、複数のタグに共通であるため、Dreamweaver では、各 **tag**.vtm ファイルを開いて Spry 属性リストを追加することはしません。代わりに、新しい XML フォーマットを使用して属性グループ（spry:region、spry:repeat など）とタググループを定義し、Configuration¥TagLibraries ディレクトリにある Spry.vtm という単一の VTM ファイルに適用します。

Spry 属性グループ化フォーマット

次のコードは、.vtm ファイルのフォーマットを示します。このフォーマットを使用すると、特定のタグに適用する属性を指定することができます。

注意：Spry 属性グループ化用のフォーマットは、Spry フレームワーク外でも使用することができます。

```
<crosstag_attributes>
    <attributegroup id="group_id_1" name="group_name_1">
        <attrib name = "fooAttr1">
        <attrib name = "barAttr1">
        ...
        <taggroup>
            <tag name = "fooTag1">
            <tag name = "barTag1">
            ...
        </taggroup>
    </attribgroup>
    <attributegroup id="group_id_2" name="group_name_2">
        <attrib name = "fooAttr2">
        <attrib name = "barAttr2">
        ...
        <taggroup>
            <tag name = "fooTag2">
            <tag name = "barTag2">
            ...
        </taggroup>
    </attribgroup>
</crosstag_attributes>
```

このコードでは :

- attributegroup は、後に続くタググループに適用する属性をリストします。
- taggroup は、前にある属性が適用されるタグをリストします。

例

```
<crosstag_attributes>
    <attribgroup id="spryRegionAttrs" name="Spry1.2">
        <attrib name="spry:region"  type="spryDataSet" allowmultiplevalues="yes"/>
        <attrib name="spry:detailregion"  type="spryDataSet" allowmultiplevalues="yes"/>
        <attrib name="spry:content"/>
        <attrib name="spry:if"/>
        <attrib name="spry:choose">
        <attrib name="spry:when"/>
        <attrib name="spry:default"/>
        <attrib name="spry:state" type="Enumerated">
            <attriboption value="read"   caption="read"/>
            <attriboption value="loading" caption="loading"/>
            <attriboption value="failed"  caption="failed"/>
        </attrib>
        <taggroup>
            <tag name="div"/>
            <tag name="span"/>
            ...
        </taggroup>
    </attribgroup>
    <attribgroup id="spryBehaviorAttrs" name="Spry1.2">
        <attrib name="spry:hover"  type="cssStyle"/>
        <attrib name="spry:select" type="cssStyle"/>
        <attrib name="spry:odd"   type="cssStyle"/>
        <attrib name="spry:even"   type="cssStyle"/>
        <taggroup>
            <tag name="a"/>
            <tag name="abbr"/>
            <tag name="acronym"/>
            ...
        </taggroup>
    </attribgroup>
</crosstag_attributes>
```

コードヒントのタグ

コードヒント XML ファイルでは、コードヒントメニューを定義する以下のタグを使用します。これらのタグを使用して、別のコードヒントメニューを定義できます。

<codehints>

説明

codehints タグは、CodeHints.xml ファイルおよび SpryCodeHints.xml ファイルのルートです。

属性

なし。

コンテンツ

menugroup タグ。

コンテナ

なし。

例

```
<codehints>
```

<menugroup>

説明

menugroup タグは、それぞれがメニューのタイプに対応しています。Dreamweaver によって定義されているメニュータイプは、環境設定ダイアログボックスでコードヒントカテゴリーを選択すると表示されます。環境設定ダイアログボックスを表示するには、編集 / 環境設定を選択します。

新しいメニューグループを作成することも、既存のグループに追加することもできます。メニューグループは、メニューの論理的なコレクションで、使用可能かどうかを環境設定ダイアログボックスで設定できます。

属性

name、enabled、id

- name 属性は、ローカライズされた名前で、環境設定ダイアログボックスのコードヒントカテゴリーのメニューグループのリストに表示されます。
- enabled 属性は、現在そのメニューグループがチェックマーク付きであるか、つまり使用可能かどうかを表します。使用可能なメニューグループは、環境設定ダイアログボックスのコードヒントカテゴリーのリストに表示されるとき、その横にチェックマークが付きます。メニューグループを使用可能にするには、値に true を指定し、使用不可にするには false を指定します。
- id 属性は、メニューグループを示す識別子（ローカライズ対象外）です。

コンテンツ

description タグ、menu タグ、および function タグ。

コンテナ

codehints タグ。

例

```
<menugroup name="Session Variables" enabled="true" id="Session_Code_Hints">
```

<description>

説明

description タグのコンテンツは、環境設定ダイアログボックスでメニューグループを選択すると表示されるテキストです。この説明テキストは、メニューグループのリストの下に表示されます。必要に応じて、この説明テキストの中に 1 つの a タグを記述することができます。この場合、href 属性には、ユーザがリンクをクリックしたときに Dreamweaver によって実行される JavaScript URL を指定する必要があります。ストリング内に特殊文字や不正な文字がある場合は、XML CDATA コンストラクトで囲むと、Dreamweaver によってテキストとして処理されます。

属性

なし。

コンテンツ

説明テキスト。

コンテナ
menugroup タグ。

例
```
<description>
<![CDATA[ To add or remove tags and attributes, use the
    <a href="javascript:dw.tagLibrary.showTagLibraryEditor()">Tag Library Editor</a>.]]>
</description>
```

<menu>

説明
このタグは、1 つのポップアップメニューを表します。ユーザが pattern 属性でストリングの最後の文字を入力すると、メニューが表示されます。例えば、セッション変数のコンテンツを表示するメニューに、「Session.」という pattern 属性を指定することができます。

属性
pattern、doctypes、casesensitive、classpattern、displayrestriction

- pattern 属性には、入力文字のパターンを指定します。このパターンが入力されると、コードヒントメニューが表示されます。パターンの最初の文字が半角英数字またはアンダースコアの場合は、ドキュメント内のこのパターンの前にある文字が半角英数字とアンダースコアのどちらでもない場合にのみメニューが表示されます。例えば、パターンが「Session.」の場合、ユーザが「my_Session.」と入力するとメニューは表示されません。
- doctypes 属性では、指定したドキュメントタイプに対してのみ、メニューをアクティブにするように指定します。この属性を使用して、ASP-JavaScript（ASP-JS）、Java Server Pages（JSP）、Adobe ColdFusion などの関数名のリストを指定できます。doctypes 属性は、ドキュメントタイプ ID をカンマで区切ったリストとして指定します。Dreamweaver のドキュメントタイプのリストについては、Dreamweaver の Configuration¥Documenttypes¥ フォルダ内の MMDocumentTypes.xml ファイルを参照してください。
- casesensitive 属性では、パターンで大文字と小文字を区別するかどうかを指定します。casesensitive 属性に指定できる値は、true、false、または doctypes 属性に指定したカンマ区切りリストのサブセットです。ドキュメントタイプのリストを使用すると、一部のドキュメントタイプに対してはパターンの大文字と小文字を区別し、その他のドキュメントタイプに対しては区別しないように指定できます。この属性を省略した場合のデフォルト値は false です。casesensitive 属性の値が true の場合は、ユーザが入力したテキストが pattern 属性に指定されたパターンと完全に一致する場合にだけコードヒントメニューが表示されます。casesensitive 属性の値が false の場合は、パターンが小文字で入力されたテキストが大文字の場合でもメニューが表示されます。
- classpattern 属性は、クラスメンバーリストをクラスに関連付けます。
- displayrestriction 属性を使用すると、CodeColoring.xml で定義されたカラーコーディングスキームに基づいて、コードヒントメニューが特定のプログラム言語のシンタックスブロックに限定されます。例えば、displayrestriction= "JavaScript" と指定すると、コードヒントメニューは JavaScript シンタックスブロックに限定されます。

コンテンツ
menuitem タグ。

コンテナ
menugroup タグ。

例
```
<menu pattern="CGI." doctypes="ColdFusion">
```

<menuitem>

説明

このタグでは、コードヒントポップアップメニューの項目のテキストを指定します。menuitem タグでは、ユーザがこの項目を選択したときにテキストに挿入される値も指定されます。

属性

label、value、{icon}、{texticon}、object、source

- label 属性は、ポップアップメニューに表示されるストリングです。
- value 属性は、ユーザがこのコマンドを選択したときにドキュメントに挿入されるストリングです。ユーザがメニューから項目を選択して Enter キーまたは Return キーを押すと、メニューが表示された後にユーザが入力したすべてのテキストが置換されます。パターンに一致する文字をユーザが入力するのはメニューが表示される前なので、その文字が再度挿入されることはありません。例えば、アンパサンド（&）を表す HTML エンティティである &amp を挿入する場合、次の menu タグおよび menuitem タグを定義します。

  ```
  <menu pattern="&amp;amp;">
  <menuitem label="&amp;amp;amp;" value="amp;" texticon="&amp;amp;"/>
  ```

  アンパサンド（&）文字は、メニューが表示される前に入力されるため、value 属性には含まれていません。
- icon はオプションの属性で、メニューテキストの左側にアイコンとして表示されるイメージファイルへのパスを指定します。アイコンの場所は、Configuration フォルダを基準とする相対 URL として指定します。
- texticon はオプションの属性で、イメージファイルの代わりにアイコン領域に表示されるテキストストリングを指定します。この属性は、HTML エンティティメニューに対して使用します。
- object 属性では、メニュー項目の属するタイプを指定します。例えば、組み込みのデータタイプである「String」や、ユーザ定義のデータ型のカスタム JavaScript ファイルを指定できます。
- source 属性では、定義元の位置を指定します。例えば、「DOM/Javascript/ custom file.js」と指定します。

コンテンツ

なし。

コンテナ

menu タグ。

例

```
<menuitem label="CONTENT_TYPE" value="&amp;quot;CONTENT_TYPE&amp;quot;)
    " icon="shared/mm/images/hintMisc.gif" />
```

<function>

説明

CodeHints.xml ファイルで使用します。このタグは、menu タグに代わり、コードヒントポップアップメニューに表示する関数引数とオブジェクトメソッドを指定します。ユーザがコードビューで関数名またはメソッド名を入力すると、関数プロトタイプのメニューが開き、現在の引数が太字で表示されます。このメニューはカンマを入力するたびに更新され、次の引数が太字で表示されます。例えば、ColdFusion ドキュメントに ArrayAppend という関数名を入力すると、コードヒントのメニューには ArrayAppend(array, value) と表示されます。array の後ろにカンマを入力すると、メニューが更新され、ArrayAppend(array, value) と表示されます。

オブジェクトメソッドの場合は、オブジェクト名を入力すると、そのオブジェクトに対して定義されているメソッドのメニューが表示されます。

認識された関数は、Configuration¥CodeHints フォルダ内にある XML ファイルに格納されます。

属性

pattern、doctypes、casesensitive

- pattern 属性には、関数の名前とその引数のリストを指定します。メソッドの場合は、pattern 属性にはオブジェクトの名前、メソッドの名前、およびメソッドの引数を記述します。関数名の場合は、ユーザが「functionname()」と入力するとコードヒントメニューが表示され、このメニューに関数の引数のリストが表示されます。オブジェクトメソッドの場合は、ユーザが「objectname.」（ピリオドを含む）と入力すると、コードヒントメニューが表示されます。このメニューには、オブジェクトに対して指定されているメソッドが表示されます。その後に、関数の場合と同様にメソッドの引数のリストがコードヒントメニューに表示されます。
- doctypes 属性では、指定したドキュメントタイプに対してのみ、メニューをアクティブにするように指定します。この属性を使用して、ASP-JavaScript（ASP-JS）、Java Server Pages（JSP）、Macromedia ColdFusion などの関数名のリストを指定できます。doctypes 属性は、ドキュメントタイプ ID をカンマで区切ったリストとして指定します。Dreamweaver のドキュメントタイプのリストについては、Dreamweaver の Configuration¥Documenttypes¥ フォルダ内にある MMDocumentTypes.xml ファイルを参照してください。
- casesensitive 属性では、パターンで大文字と小文字を区別するかどうかを指定します。casesensitive 属性に指定できる値は、true、false、または doctypes 属性に指定したカンマ区切りリストのサブセットです。ドキュメントタイプのリストを使用すると、一部のドキュメントタイプに対してはパターンの大文字と小文字を区別し、その他のドキュメントタイプに対しては区別しないように指定できます。この属性を省略した場合のデフォルト値は false です。casesensitive 属性の値が true の場合は、ユーザが入力したテキストが pattern 属性に指定されたパターンと完全に一致する場合にだけコードヒントメニューが表示されます。casesensitive 属性の値が false の場合は、パターンが小文字で入力されたテキストが大文字の場合でもメニューが表示されます。

コンテンツ

なし。

コンテナ

menugroup タグ。

例

```
// function example
<function pattern="CreateDate(year, month, day)" DOCTYPES="ColdFusion" />
// object method example
<function pattern="application.getAttribute(String name)" DOCTYPES="JSP" />
```

<method>

説明

Spry フレームワークに使用されます。このタグは、menu タグに代わり、コードヒントポップアップメニューに表示するメソッドを指定します。コードビューでメソッド名を入力すると、メソッドプロトタイプのメニューが開き、そのメソッドのパラメータのリストが表示されます。一連のパラメータを入力すると、入力と同時にトラッキングされます。メソッドにパラメータがない場合、Dreamweaver は、カッコ " () " を付加してメソッド呼び出しを閉じます。

現在の引数が太字で表示されます。このメニューはカンマを入力するたびに更新され、次の引数が太字で表示されます。オブジェクトメソッドの場合は、オブジェクト名を入力すると、そのオブジェクトに対して定義されているメソッドのメニューが表示されます。

属性

pattern、icon、object、source、constructor、static、retType

- pattern 属性では、メソッドの名前とその引数のリストを指定します。オブジェクトとメソッドの名前、およびメソッドの引数も指定します。メソッドの引数のリストがメニューに表示されます。ユーザが「objectname.」（ピリオドを含みます）と入力すると、コードヒントメニューが表示されます。このメニューには、オブジェクトに対して指定されているメソッドが表示されます。コードヒントメニューが開き、関数の場合と同じようにメソッドの引数のリストが表示されます。
- icon 属性では、使用されるアイコンを指定します。
- object 属性では、メニュー項目の属するタイプを指定します。例えば、組み込みのデータタイプである「String」や、ユーザ定義のデータ型のカスタム JavaScript ファイルを指定できます。
- source 属性では、定義元の位置を指定します。例えば、「DOM/Javascript/ custom file.js」と指定します。
- constructor 引数はブール値です。constructor を true に設定すると、オブジェクトインスタンスを構成するメソッドで、他のオブジェクトメソッドと比較して別にマークされているメソッドが指定されます。
- static 引数はブール値です。static を true に設定すると、そのメソッドを特定のオブジェクトインスタンスに適用せず、オブジェクトタイプ自体に適用するように指定されます。例えば、

  ```
  Date.parse(dateString)
  ```

- retType 属性は、メソッドの戻り値のタイプを指定しますが、そのタイプは同時にコードヒントのカスケードをサポートするためのオブジェクトタイプにもなります。

コンテンツ

なし。

コンテナ

menu タグ。

<parammenu>

説明

任意のオブジェクト（JavaScript）に使用され、メソッドまたは関数が取るパラメータのパラメータヒントを指定します。

属性

pattern、name、index、type

- pattern 属性では、コードヒントメニューを起動する文字を指定します。この引数は必須です。
- name 属性では、パラメータの名前を指定します。この引数は必須です。
- index 属性では、ヒントを表示するパラメータのインデックス番号（ゼロから開始）を指定します。この引数は必須です。
- type 属性では、データタイプを指定します。サポートされているデータタイプは以下のとおりです。
  - enumerated（デフォルト）は、表示するネストされた <optionparammenuitem> のリストを示します。
  - spryDataReferences は、Spry データセット列のリストを示します。
  - cssStyle は、ページで利用できる CSS クラスのリストを示します。
  - cssId は、ページで利用できる CSS セレクタ ID ルールのリストを示します。
  - optionArray は、表示するネストされた <optionparammenu> および <parammenuitem> を示します（オプション配列のパラメータタイプをサポートするために使用されます）。

コンテンツ
なし。

コンテナ
method タグまたは function タグ。

<parammenuitem>

説明
任意のオブジェクト（JavaScript）に対して、メソッドまたは関数が取るパラメータのパラメータヒントを指定します。

属性
label、value、icon、{datatype}、object、source

- label 属性では、必ず表示される名前を指定します。この引数は必須です。
- value 属性では、コードヒントメニューで項目を選択したときに必ずドロップする値を指定します。この引数は必須です。
- icon 属性では、コードヒントメニューで必ず使用するアイコンを指定します。この引数は必須です。
- datatype 属性では、コードヒントメニューで値を選択するとき、必ず閉じ引用符を付加することを示す string を指定することができます。この引数はオプションです。
- object 属性では、メニュー項目の属するタイプを指定します。例えば、組み込みのデータタイプである「String」や、ユーザ定義のデータ型のカスタム JavaScript ファイルを指定できます。
- source 属性では、定義元の位置を指定します。例えば、「DOM/Javascript/ custom file.js」と指定します。

コンテンツ
なし。

コンテナ
parammenu タグ。

<optionparammenu>

説明
任意のオブジェクト（JavaScript）に使用され、メソッドまたは関数が取る引数のオプション配列のヒントを指定します。オプション配列は、option:value のフォームでサブ引数を取ることのできる引数です。ほとんどの Spry オブジェクトでは、オプション配列引数を使用して、オブジェクト（Data Set、Widget、Effect など）のビヘイビアを設定できるようにしています。一般にオプション配列は、{option1: value1, option2: value2, option3: value3, ...} のように表示されます。

属性
pattern、label、value、icon、type

- pattern 属性では、コードヒントメニューを起動する文字を指定します。この引数は必須です。
- label 属性では、パラメータの名前を指定します。この引数は必須です。
- value 属性では、コードヒントを選択したときに挿入するパラメータの値を指定します。この引数は必須です。
- icon 属性では、使用するアイコンを指定します。この引数は必須です。

- type 属性では、データタイプを指定します。サポートされているデータタイプは以下のとおりです。
  - enumerated（デフォルト）は、表示するネストされた optionparammenuitem のリストを示します。
  - spryDataReferences は、Spry データセット列のリストを示します。
  - cssStyle は、ページで利用できる CSS クラスのリストを示します。
  - cssId は、ページで利用できる CSS セレクタ ID ルールのリストを示します。

コンテンツ
なし。

コンテナ
parammenu タグ（データタイプ optionArray）。

<optionparammenuitem>

説明
任意のオブジェクト（JavaScript）に使用され、メソッドまたは関数が取るパラメータのパラメータヒントを指定します。

属性
label、value、icon、{datatype}

- label 属性では、表示する名前を指定します。この引数は必須です。
- value 属性では、コードヒントメニューで項目を選択したときにドロップする値を指定します。この引数は必須です。
- icon 属性では、コードヒントメニューで使用するアイコンを指定します。この引数は必須です。
- datatype 属性では、コードヒントメニューで値を選択するとき、閉じ引用符を付加することを示す string を指定することができます。この引数はオプションです。

コンテンツ
なし。

コンテナ
<optionparammenu> タグ。

<property>

説明
このタグは、オブジェクトのプロパティまたはフィールドを表し、次の標準属性を備えています。

属性
label、value、icon、object、source、static、propType、item

- label 属性は、ポップアップメニューに表示されるストリングです。
- value 属性は、ユーザがこのコマンドを選択したときにドキュメントに挿入されるストリングです。ユーザがメニューから項目を選択して Enter キーまたは Return キーを押すと、メニューが表示された後にユーザが入力したすべてのテキストが置換されます。パターンに一致する文字をユーザが入力するのはメニューが表示される前なので、その文字が再度挿入されることはありません。

- icon はオプションの属性で、メニューテキストの左側にアイコンとして表示されるイメージファイルへのパスを指定します。アイコンの場所は、Configuration フォルダを基準とする相対 URL として指定します。
- object 属性では、メニュー項目の属するタイプを指定します。例えば、組み込みのデータタイプである「String」や、ユーザ定義のデータ型のカスタム JavaScript ファイルを指定できます。
- source 属性では、定義元の位置を指定します。例えば、「DOM/Javascript/ custom file.js」と指定します。
- static 引数はブール値です。static を true に設定すると、そのメソッドを特定のオブジェクトインスタンスに適用せず、オブジェクトタイプ自体に適用するように指定されます。例えば、

  ```
  Number.MAX_VALUE
  ```

- propType 属性はプロパティのタイプを指定しますが、そのタイプは同時にコードヒントのカスケードをサポートするためのオブジェクトタイプにもなります。例えば、

  ```
  domElement.innerHTML.<code hints for String type>
  ```

- propType 属性がコレクション コンテナ タイプである場合、item 属性によって要素のタイプを指定します。item によって、コンテナ内の各項目のタイプが指定されます（すべて同じタイプの要素から構成されている場合）。

コンテンツ
なし。

コンテナ
menu タグ。

<event>

説明
このタグは、オブジェクトのイベントを表し、次の標準属性を備えています。

属性
label、icon、source、object

- label 属性は、イベントの名前になります。
- icon 属性で、メニューテキストの横に表示されるアイコンのイメージファイルへのパスを指定します。
- source 属性では、定義元の位置を指定します。
- object 属性では、コマンドが属するタイプを指定します。

コンテナ
menu タグ。

例

```
<event label="onblur" source="DOM 1&amp;amp;2" icon="shared/mm/images/codeHintEvent.gif"/>
```

コードカラーリングについて

Dreamweaver では、コードビューに表示されるコードカラーリングスキームをカスタマイズまたは拡張できます。これにより、新しいキーワードをスキームに追加したり、新しいドキュメントタイプのコードカラーリングスキームを追加できます。例えば、クライアントサイドのスクリプトで使用する JavaScript 関数を開発する場合に、関数の名前をキーワードセク

ションに追加して、環境設定ダイアログボックスで指定されたカラーで表示されるようにできます。同様に、アプリケーションサーバの新しいプログラミング言語を開発し、Dreamweaver ユーザがその言語を使用してページを作成できるように新しいドキュメントタイプを配布する場合、そのドキュメントタイプ用のコードカラーリングスキームを追加できます。

Dreamweaver には JavaScript 関数 dreamweaver.reloadCodeColoring() があり、これにより、手動で編集されたコードカラーリング XML ファイルを再読み込みできます。この関数について詳しくは、『Dreamweaver API リファレンス』を参照してください。

コードカラーリングスキームを更新したり新しいスキームを追加するには、コードカラーリング定義ファイルを修正する必要があります。

コードカラーリングファイル

Dreamweaver では、コードカラーリングのスタイルとスキームを、Configuration¥CodeColoring フォルダ内の XML ファイルで指定します。コードカラーリングのスタイルファイルでは、シンタックス定義で定義されるフィールドのスタイルを定義します。スタイルファイルには、<codeColors> のルートノードがあります。コードカラーリングスキームファイルには、コードカラーリングのシンタックス定義と、<codeColoring> のルートノードがあります。

Dreamweaver が提供するコードカラーリングスタイルファイルは、Colors.xml です。Dreamweaver が提供するコードカラーリングシンタックスファイルは、CodeColoring.xml、ASP JavaScript.xml、ASP VBScript.xml、ASP.NET CSharp.xml および ASP.NET VB.xml です。

次の例は、Colors.xml ファイルからの抜粋で、コードカラーリングスタイルファイルのタグの階層を示したものです。

```
<codeColors>
    <colorGroup>
        <syntaxColor id="CodeColor_HTMLEntity" bold="true" />
        <syntaxColor id="CodeColor_JavascriptNative" text="#009999" />
        <syntaxColor id="CodeColor_JavascriptNumber" text="#FF0000" />
...
        <tagColor id="CodeColor_HTMLStyle" text="#990099" />
        <tagColor id="CodeColor_HTMLTable" text="#009999" />
        <syntaxColor id="CodeColor_SpryAttributes" text="#FF6208" />
...
    </colorGroup>
</codeColors>
```

色は、RGB（赤、緑、青）の 16 進数で指定されます。例えば、上に示した XML コードのステートメント text="009999" は、青緑色を ID "CodeColor_JavascriptNative" に割り当てます。

次の例は、CodeColoring.xml ファイルからの抜粋で、コードカラーリングスキームファイルのタグの階層、およびコードカラーリングスタイルファイルとコードカラーリングスキームファイルの関係を示しています。

```
<codeColoring>
    <scheme name="Text" id="Text" doctypes="Text" priority="1">
        <ignoreTags>Yes</ignoreTags>
        <defaultText name="Text" id="CodeColor_TextText" />
        <sampleText doctypes="Text">
<![CDATA[Default file syntax highlighting.
The quick brown fox
jumped over the lazy dog.
]]>
        </sampleText>
    </scheme>
    <scheme name="HTML" id="HTML" doctypes=
"ASP.NET_VB,ASP.NET_CSharp,ASP-JS,ASP-VB,ColdFusion,CFC,HTML,JSP,PHP_MySQL,LibraryItem,
    WML,XSLT" priority="50">
        <ignoreCase>Yes</ignoreCase>
        <ignoreTags>No</ignoreTags>
        <defaultText name="Text" id="CodeColor_HTMLText" />
        <defaultTag name="Other Tags" id="CodeColor_HTMLTag" />
        <defaultAttribute />
        <commentStart name="Comment" id="CodeColor_HTMLComment"><![CDATA[<!--]]>
            </commentStart>
    ...
    <tagGroup name="HTML Anchor Tags" id="CodeColor_HTMLAnchor" taglibrary="DWTagLibrary_html"
         tags="a" />
    <tagGroup name="HTML Form Tags" id="CodeColor_HTMLForm" taglibrary="DWTagLibrary_html" tags
="select,form,input,option,textarea" />
```

Colors.xml ファイルの syntaxColor タグと tagColor タグでは、色とスタイルの値を id ストリング値に割り当てています。id の値は、スタイルを scheme タグに割り当てるために CodeColoring.xml ファイルで使用されます。例えば、CodeColoring.xml ファイルの抜粋の中で示した defaultTag タグの id は "CodeColor_HTMLComment" です。Colors.xml ファイルでは、id の値が "CodeColor_HTMLComment" であるものに、text= の値 "#999999"（灰色）が割り当てられます。

Dreamweaver には、コードカラーリングスキームとして、Default、HTML、JavaScript、ASP_JavaScript、ASP_VBScript、JSP および ColdFusion があります。デフォルトスキームには "Text" と等しい id 値が含まれています。Dreamweaver では、コードカラーリングスキームが定義されていないドキュメントタイプに対してデフォルトスキームを使用します。

コードカラーリングファイルには、次のタグが含まれます。これらについては、下で説明します。

scheme、blockEnd、blockStart、brackets、charStart、charEnd、charEsc、commentStart、commentEnd、cssImport/、cssMedia/、cssProperty/、cssSelector/、cssValue/、defaultAttribute、defaultTag、defaultText/、endOfLineComment、entity/、functionKeyword、idChar1、idCharRest、ignoreCase、ignoreMMTParams、ignoreTags、isLocked、keyword、keywords、numbers/、operators、regexp、sampleText、searchPattern、stringStart、stringEnd、stringEsc、tagGroup

<scheme>

説明

scheme タグでは、コードテキストブロックのコードカラーリングを指定します。1 つのファイルに複数のスキームを指定して、様々なスクリプトやタグ言語に対して異なる色を指定できます。各スキームには優先度があるため、あるスキーマが割り当てられているテキストブロック内に別のスキーマが割り当てられているテキストブロックをネストすることができます。

Dreamweaver CS4 以降、コードカラーリングパーサーでは <scheme> タグが同じ ID とマージされます。競合しないすべてのタグが既存の <scheme> に追加されます。ファイルの日付が最新のスキームにより競合が解決されます。

属性
name、id、priority、{doctypes}

- ***name="scheme_name"***。スキームに名前を割り当てるストリング。カラーリングスキームの編集ダイアログボックスにスキーム名が表示されます。HTML Comment のように、スキーム名とフィールド名の組み合わせが表示されます。名前を指定しない場合、カラーリングスキームの編集ダイアログボックスにはスキームのフィールドが表示されません。カラーリングスキームの編集ダイアログボックスについて詳しくは、62 ページの「スキームの編集」を参照してください。
- ***id="id_string"***。必須。色とスタイルをこのシンタックスの項目にマップする識別子ストリングです。
- ***priority="string"***。値の範囲は "1" ～ "99" です。最も高い優先度は "1" です。これは、スキームの優先度を指定します。より高い優先度を持つブロックの内側では、ブロックは無視されます。同じか、より低い優先度を持つブロックの内側では、ブロックは優先されます。優先度を指定しない場合、デフォルト値の "50" に設定されます。
- ***doctypes="doc_list"***。オプション。このコードカラーリングスキームが適用されるドキュメントタイプを、カンマ区切りのリストで指定します。この値は、複数の開始ブロックと終了ブロックが同じ拡張子を使用する場合の競合を解消するために必要です。

コンテンツ
```
blockEnd, blockStart, brackets, charStart, charEnd, charEsc, commentStart, commentEnd,
cssProperty/, cssSelector/, cssValue/, defaultAttribute, defaultText/, endOfLineComment,
entity/, functionKeyword, idChar1, idCharRest, ignoreCase, ignoreMMTParam, ignoreTags,
keywords, numbers/, operators, regexp, sampleText, searchPattern, stringStart, stringEnd,
stringEsc, urlProtocol, urlProtocols
```

コンテナ
codeColoring タグ

例
```
<scheme name="Text" id="Text" doctypes="Text" priority="1">
```

<blockEnd>

説明
オプション。このスキームのテキストブロックの終わりを区切るテキスト値です。blockEnd タグと blockStart タグはペアで使用し、その組み合わせは一意でなければなりません。値の大文字と小文字は区別されません。blockEnd 値には、1 つの文字を指定することもできます。このタグは、複数のインスタンスを使用することもできます。blockEnd ストリングについて詳しくは、60 ページの「ワイルドカード文字」を参照してください。

属性
なし。

例
```
<blockEnd><![CDATA[--->]]></blockEnd>
```

<blockStart>

説明

オプション。カラーリングスキームを別のカラーリングスキーム内に埋め込むことができる場合にだけ指定します。blockStart タグと blockEnd タグはペアで使用し、その組み合わせは一意でなければなりません。値の大文字と小文字は区別されません。blockStart 値は、2 文字以上の長さが必要です。このタグは、複数のインスタンスを使用することもできます。blockStart ストリングについて詳しくは、60 ページの「ワイルドカード文字」を参照してください。blockStartscheme 属性について詳しくは、57 ページの「スキームブロック区切り記号のカラーリング」を参照してください。

属性

canNest、doctypes、id、name、scheme

- canNest。ネストできるスキームかどうかを指定します。値は、Yes または No です。デフォルト値は No です。
- doctypes="***doc_type1, doc_type2,...***"。必須。このコードカラーリングスキームをネストできるドキュメントタイプを、カンマ区切りのリストで指定します。ドキュメントタイプは、Dreamweaver の Configuration¥Document Types¥ フォルダ内にある MMDocumentTypes.xml ファイルで定義されます。
- id="***id_string***"。scheme="customText" の場合、この属性は必須です。色とスタイルをこのシンタックスの項目にマップする識別子ストリングです。
- name="***display_name"***。scheme="customText" の場合にカラーリングスキームの編集ダイアログボックスに表示されるストリング。
- scheme。必須。これは、blockStart ストリングと blockEnd ストリングのカラーリング方法を定義します。スキーム属性に指定できる値について詳しくは、57 ページの「スキームブロック区切り記号のカラーリング」を参照してください。

例

```
<blockStart doctypes="ColdFusion,CFC" scheme="innerText" canNest="Yes"><![CDATA[<!---]]>
</blockStart>
```

<brackets>

説明

カッコを表す文字のリストです。

属性

name、id

- name="***bracket_name***"。カッコのリストに名前を割り当てるストリング。
- id="***id_string***"。必須。色とスタイルをこのシンタックスの項目にマップする識別子ストリングです。

例

```
<brackets name="Bracket" id="CodeColor_JavaBracket"><![CDATA[{[()]}]]>
</brackets>
```

<charStart>

説明

文字の開始の区切り記号を表すテキストストリングを含みます。charStart タグと charEnd タグはペアで指定する必要があります。複数の charStart … charEnd ペアを使用できます。

属性
なし。

例
```
<charStart><![CDATA[']]></charStart>
```

<charEnd>

説明
文字の終わりの区切り記号を表すテキストストリングを含みます。charStart タグと charEnd タグはペアで指定する必要があります。複数の charStart … charEnd ペアを使用できます。

属性
なし。

例
```
<charEnd><![CDATA[']]></charEnd>
```

<charEsc>

説明
エスケープ文字を表すテキストストリングを含みます。複数の charEsc タグを使用できます。

属性
なし。

例
```
<charEsc><![CDATA[\]]></charEsc>
```

<commentStart>

説明
コメントブロックの開始を区切るテキストストリング。commentStart タグと commentEnd タグはペアで指定する必要があります。複数の commentStart… /commentEnd のペアを使用できます。

属性
なし。

例
```
<commentStart><![CDATA[<%--]]></commentStart>
```

<commentEnd>

説明
コメントブロックの終わりを区切るテキストストリング。commentStart タグと commentEnd タグはペアで指定する必要があります。複数の commentStart… /commentEnd のペアを使用できます。

属性
なし。

例
```
<commentEnd><![CDATA[--%>]]></commentEnd>
```

<cssImport/>

説明
CSS の style エレメントの @import 関数に対するコードカラーリングルールを示す空のタグ。

属性
name、id

- name="***cssImport_name***"。CSS の @import 関数に名前を割り当てるストリング。
- id="***id_string***"。必須。色とスタイルをこのシンタックスの項目にマップする識別子ストリングです。

例
```
<cssImport name="@import" id="CodeColor_CSSImport" />
```

<cssMedia/>

説明
CSS の style エレメントの @media 関数に対するコードカラーリングルールを示す空のタグ。

属性
name、id

- name="***cssMedia_name***"。CSS の @media 関数に名前を割り当てるストリング。
- id="***id_string***"。必須。色とスタイルをこのシンタックスの項目にマップする識別子ストリングです。

例
```
<cssMedia name="@media" id="CodeColor_CSSMedia" />
```

<cssProperty/>

説明
CSS ルールを示し、コードカラーリング属性を保持する空のタグです。

属性
name、id

- name="***cssProperty_name***"。CSS プロパティに名前を割り当てるストリング。
- id="***id_string***"。必須。色とスタイルをこのシンタックスの項目にマップする識別子ストリングです。

コードカラーの設定
CSS プロパティ

例

```
<cssProperty name="Property" id="CodeColor_CSSProperty" />
```

<cssSelector/>

説明

CSS ルールを示し、コードカラーリング属性を保持する空のタグです。

属性

name、id

- name="***cssSelector_name***"。CSS セレクタに名前を割り当てるストリング。
- id="***id_string***"。必須。色とスタイルをこのシンタックスの項目にマップする識別子ストリングです。

例

```
<cssSelector name="Selector" id="CodeColor_CSSSelector" />
```

<cssValue/>

説明

CSS ルールを示し、コードカラーリング属性を保持する空のタグです。

属性

name、id

- name="***cssValue_name***"。CSS 値に名前を割り当てるストリング。
- id="***id_string***"。必須。色とスタイルをこのシンタックスの項目にマップする識別子ストリングです。

例

```
<cssValue name="Value" id="CodeColor_CSSValue" />
```

<defaultAttribute>

説明

オプション。このタグは、タグベースのシンタックス（ignoreTags="No"）にのみ適用されます。このタグがある場合、すべてのタグ属性は、このタグに割り当てられたスタイルに従ってカラーリングされます。このタグを省略した場合、属性はタグと同じ色になります。

属性

name •デフォルトの属性に名前を割り当てるストリング。

例

```
<defaultAttribute name="Attribute"/>
```

<defaultTag>

説明

このタグは、スキームのタグのデフォルトの色とスタイルを指定するために使用します。

属性

name、id

- name="***display_name***"。Dreamweaver のコードカラーエディタに表示されるストリング。
- id="***id_string***"。必須。色とスタイルをこのシンタックスの項目にマップする識別子ストリングです。

例

```
<defaultTag name="Other Tags" id="CodeColor_HTMLTag" />
```

<defaultText/>

説明

オプション。このタグがある場合、他のタグで定義されていないすべてのテキストは、このタグに割り当てられたスタイルに従ってカラーリングされます。このタグを省略すると、テキストは黒で表示されます。

属性

name、id

- name="***cssSelector_name***"。CSS セレクタに名前を割り当てるストリング。
- id="***id_string***"。必須。色とスタイルをこのシンタックスの項目にマップする識別子ストリングです。

例

```
<defaultText name="Text" id="CodeColor_TextText" />
```

<endOfLineComment>

説明

現在の行の終わりまで続いているコメントの開始を区切るテキストストリング。複数の endOfLineComment… /endOfLineComment タグを使用できます。

属性

なし。

例

```
<endOfLineComment><![CDATA[//]]></endOfLineComment>
```

<entity/>

説明

HTML の特殊文字を認識すること、および、カラーリング属性を保持していることを示す空のタグ。

属性

name、id

- name="***entity_name***"。エンティティに名前を割り当てるストリング。
- id="***id_string***"。必須。色とスタイルをこのシンタックスの項目にマップする識別子ストリングです。

例

```
<entity name="Special Characters" id="CodeColor_HTMLEntity" />
```

<functionKeyword>

説明

関数を定義するキーワードを識別します。Dreamweaver では、これらのキーワードを使用してコードナビゲーションを実行します。複数の functionKeyword タグを使用できます。

属性

name、id

- name="***functionKeyword_name***"。functionKeyword ブロックに名前を割り当てるストリング。
- id="***id_string***"。必須。色とスタイルをこのシンタックスの項目にマップする識別子ストリングです。

例

```
<functionKeyword name="Function Keyword" id="CodeColor_JavascriptFunction">function</functionKeyword>
```

<idChar1>

説明

識別子の最初の文字として Dreamweaver が識別できる文字のリスト。

属性

name、id

- name="***idChar1_name***"。識別子の文字のリストに名前を割り当てるストリング。
- id="***id_string***"。必須。色とスタイルをこのシンタックスの項目にマップする識別子ストリングです。

例

```
<idChar1>_$abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ</idChar1>
```

<idCharRest>

説明

識別子の残りの文字として認識される文字のリスト。idChar1 を指定しないと、識別子のすべての文字がこのリストを使用して検証されます。

属性

name、id

- name="***idCharRest_name***"。stringStart ブロックに名前を割り当てるストリング。
- id="***id_string***"。必須。色とスタイルをこのシンタックスの項目にマップする識別子ストリングです。

例

```
<idCharRest name="Identifier" id="CodeColor_JavascriptIdentifier">
    _$abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789</idCharRest>
```

<ignoreCase>

説明
トークンをキーワードと比較するときに大文字と小文字の違いを無視するかどうかを指定します。値は、Yes または No です。デフォルトは、Yes です。

属性
なし。

例
```
<ignoreCase>Yes</ignoreCase>
```

<ignoreMMTParams>

説明
MMTInstance:Param タグ、<!-- InstanceParam タグ、または <!-- #InstanceParam タグに他とは違う色を付けるかどうかを指定します。値は、Yes または No です。デフォルトは、Yes です。これにより、テンプレートを使用するページで適切にカラーリングが処理されます。

属性
なし。

例
```
<ignoreMMTParams>No</ignoreMMTParams>
```

<ignoreTags>

説明
マークアップタグを無視するかどうかを指定します。値は、Yes または No です。デフォルトは、Yes です。No に設定するのは、< と > で区切られるタグマークアップ言語のシンタックスの場合です。プログラミング言語のシンタックスでは、Yes に設定します。

属性
なし。

例
```
<ignoreTags>No</ignoreTags>
```

<isLocked>

説明
このスキームに一致するテキストを、コードビューで編集できない設定にするかどうかを指定します。値は、Yes または No です。デフォルトは、No です。

属性
なし。

例

```
<isLocked>Yes</isLocked>
```

<keyword>

説明

キーワードを定義するテキストのストリング。複数の keyword タグを使用できます。キーワードの 1 文字目には任意の文字を使用できますが、2 文字目以降は、a-z、A-Z、0-9、_、$ または @ のいずれかである必要があります。

コードカラーは、含まれる keyword タグによって指定されます。

属性

なし。

例

```
<keyword>.getdate</keyword>
```

<keywords>

説明

category 属性で指定されたタイプのキーワードのリスト。複数の keywords タグを使用できます。

属性

name、id

- name="***keywords_name***"。キーワードのリストに名前を割り当てるストリング。
- id="***id_string***"。必須。色とスタイルをこのシンタックスの項目にマップする識別子ストリングです。

コンテンツ

<keyword></keyword>

例

```
<keywords name="Reserved Keywords" id="CodeColor_JavascriptReserved">
    <keyword>break</keyword>
    <keyword>case</keyword>
</keywords>
```

<numbers/>

説明

認識する必要がある数値を指定する空のタグ。このタグは、カラー属性も保持します。

属性

name、id

- name="***number_name***"。numbers タグに名前を割り当てるストリング。
- id="***id_string***"。必須。色とスタイルをこのシンタックスの項目にマップする識別子ストリングです。

例

```
<numbers name="Number" id="CodeColor_CFScriptNumber" />
```

<operators>

説明

演算子として認識される文字のリスト。

属性

name、id

- name="***operator_name***"。演算子文字のリストに名前を割り当てるストリング。
- id="***id_string***"。必須。色とスタイルをこのシンタックスの項目にマップする識別子ストリングです。

例

```
<operators name="Operator" id="CodeColor_JavaOperator"><![CDATA[+-*/%<>!?:=&|^~]]></operators>
```

<regexp>

説明

searchPattern タグのリストを指定します。

属性

name、id、delimiter、escape

- name="***stringStart_name***"。検索パターンのストリングのリストに名前を割り当てるストリング。
- id="***id_string***"。必須。色とスタイルをこのシンタックスの項目にマップする識別子ストリングです。
- delimiter。正規表現を開始および終了する文字またはストリング。
- escape。文字の特殊な処理を示す文字やストリング（エスケープ文字またはストリング）。

コンテンツ

<searchPattern></searchPattern>

例

```
<regexp name="RegExp" id="CodeColor_JavascriptRegexp" delimiter="/" escape="\\">
    <searchPattern><![CDATA[(\s*/\e*\\/]]></searchPattern>
    <searchPattern><![CDATA[=\s*/\e*\\/]]></searchPattern>
</regexp>
```

<sampleText>

説明

カラーリングスキームの編集ダイアログボックスのプレビューウィンドウに表示される表示テキスト。カラーリングスキームの編集ダイアログボックスについて詳しくは、62 ページの「スキームの編集」を参照してください。

属性

doctypes

- doctypes=***"doc_type1, doc_type2,..."***。このサンプルテキストが表示されるドキュメントタイプ。

例

```
<sampleText doctypes="JavaScript"><![CDATA[/* JavaScript */
function displayWords(arrayWords) {
    for (i=0; i < arrayWords.length(); i++) {
        // inline comment
        alert("Word " + i + " is " + arrayWords[i]);
    }
}

var tokens = new Array("Hello", "world");
displayWords(tokens);
]]></sampleText>
```

<searchPattern>

説明

サポートされているワイルドカード文字を使用して正規表現による検索パターンを定義する文字のストリング。複数の searchPattern タグを使用できます。

属性

なし。

コンテナ

regexp タグ。

例

```
<searchPattern><![CDATA[(\s*/\e*\\/]]></searchPattern>
```

<stringStart>

説明

これらのタグには、ストリング開始の区切り記号を表すテキストストリングが含まれます。stringStart タグと stringEnd タグは、ペアで指定する必要があります。複数の stringStart … stringEnd ペアを使用できます。

属性

name、id、wrap

- name="***stringStart_name***"。stringStart ブロックに名前を割り当てるストリング。
- id="***id_string***"。必須。色とスタイルをこのシンタックスの項目にマップする識別子ストリングです。
- wrap="true" または "false"。コードカラーリングで、次の行に送られるテキストストリングを認識するかどうかを定義します。デフォルトは、"true" です。

例

```
<stringStart name="Attribute Value" id="CodeColor_HTMLString"><![CDATA["]]></stringStart>
```

<stringEnd>

説明

コードストリングの終わりの区切り記号を表すテキストストリングを含みます。stringStart タグと stringEnd タグは、ペアで指定する必要があります。複数の stringStart … stringEnd ペアを使用できます。

属性
なし。

例
```
<stringEnd><![CDATA["]]></stringEnd>
```

<stringEsc>

説明
ストリングエスケープ文字の区切り記号を表すテキストストリングを含みます。複数の stringEsc タグを使用できます。

属性
なし。

例
```
<stringEsc><![CDATA[\]]></stringEsc>
```

<tagGroup>

説明
このタグでは、タグをグループ化して、固有の色とスタイルを割り当てることができます。

属性
id、name、taglibrary、tags

- id="***id_string***"。必須。色とスタイルをこのシンタックスの項目にマップする識別子ストリングです。
- name="***display_name***"。Dreamweaver のコードカラーエディタに表示されるストリング。
- taglibrary="***tag_library_id***"。対象のタググループが属するタグライブラリの識別子。
- tags="**tag_list**"。タググループを構成する 1 つのタグ、または複数のタグのカンマ区切りのリスト。

例
```
<tagGroup name="HTML Table Tags" id="CodeColor_HTMLTable" taglibrary="DWTagLibrary_html"
    tags="table,tbody,td,tfoot,th,thead,tr,vspec,colw,hspec" />
```

スキームブロック区切り記号のカラーリング

blockStart scheme 属性は、ブロックを開始および終了するストリング、またはブロックの区切り記号のカラーリングを制御します。次の値は、blockStart 属性の有効な値です。

注意：blockStart.scheme 属性と scheme タグを混同しないように注意してください。

innerText

この値を指定すると、ブロック区切り文字の色として、そのブロック区切り文字内のスキームのデフォルトテキストと同じ色が使用されます。

Template スキームは、このスキームの効果の例です。Template スキームは、編集できないことを示すために灰色で表示された読み取り専用コードのブロックに一致します。ブロック区切り記号は、<!#EndEditable -->-- および <!-- #BeginEditable "..." --> ストリングであり、これらも編集できないため、灰色で表示されます。

サンプルコード

```
<!-- #EndEditable -->
    <p><b><font size="+2">header</font></b></p>
    <!-- #BeginEditable "test" -->
    <p>Here's some editable text </p>
    <p> </p>
    <!-- #EndEditable -->
```

例

```
<blockStart doctypes="ASP-JS,ASP-VB, ASP.NET_CSharp, ASP.NET_VB, ColdFusion,CFC, HTML,
    JSP,LibraryItem,PHP_MySQL" scheme="innerText"><![CDATA[<!--\s*#BeginTemplate]]>
    </blockStart>
```

customText

この値を指定すると、ブロック区切り文字の色にカスタムカラーを使用するように Dreamweaver に指示できます。

サンプルコード

赤で表示されている PHP スクリプトのブロック区切り文字は、customText 値の効果の例を示しています。

```
<?php
    if ($loginMsg <> "")
    echo $loginMsg;
    ?>
```

例

```
<blockStart name="Block Delimiter" id="CodeColor_JavaBlock" doctypes="JSP"
    scheme="customText"><![CDATA[<%]]></blockStart>
```

outerTag

outerTag 値は、blockStart タグと blockEnd タグの両方が完全なタグであること、およびタグを囲むスキームに従ってタグがカラーリングされることを指定します。

<script> と </script> のストリングが blockStart タグと blockEnd タグである JavaScript スキームで、この値の例を示します。このスキームは、タグを認識しない JavaScript コードのブロックと一致しているため、区切り文字は、それらを囲むスキームで指定された色で表示される必要があります。

サンプルコード

```
<script language="JavaScript">
    // comment
    if (true)
    window.alert("Hello, World");
    </script>
```

例

```
<blockStart doctypes="PHP_MySQL" scheme="outerTag">
    <![CDATA[<script\s+language="php">]]></blockStart>
```

innerTag

この値は、タグのカラーリングが区切り文字内のスキームから取得される点を除き、outerTag の値と同じです。現在は html タグで使用されています。

nameTag

この値は、blockStart ストリングでタグが始まり、blockEnd ストリングでタグが終了すること、および、これらの区切り文字がスキームのタグ設定に基づいてカラーリングされることを指定します。

このタイプのスキームでは、cfoutput タグなど、他のタグ内に埋め込むことのできるタグが表示されます。

サンプルコード

```
<input type="text" name="zip"
    <cfif newRecord IS "no">
    <cfoutput query="employee"> Value="#zip#" </cfoutput>
    </cfif>
    >
```

例

```
<blockStart doctypes="ColdFusion,CFC" scheme="nameTag">
    <![CDATA[<cfoutput\n]]></blockStart>
```

nameTagScript

この値は、nameTag スキームと同じです。ただし、内容は name=value という属性のペアではなく、代入ステートメントや代入式などのスクリプトです。

このタイプのスキームでは、ColdFusion の cfset タグ、cfif タグ、および cfifelse タグなど、タグ自体にスクリプトを格納する特殊なタイプのタグが表示されます。このタグは、他のタグ内に埋め込むことができます。

サンプルコード

59 ページの「nameTag」のサンプルテキストを参照してください。

例

```
<blockStart doctypes="ColdFusion,CFC" scheme="nameTagScript"><![CDATA[<cfset\n]]></blockStart>
```

スキームの処理

Dreamweaver には、基本的なコードカラーリングモードが 3 つあります。CSS モード、スクリプトモード、およびタグモードです。

各モードでは、特定のフィールドにのみコードカラーリングが適用されます。次の表では、各モードでコードカラーリングが適用されるフィールドを示します。

フィールド	CSS	タグ	スクリプト
defaultText		X	X
defaultTag		X	
defaultAttribute		X	
comment	X	X	X
string	X	X	X
cssProperty	X		
cssSelector	X		
cssValue	X		
character		X	X

フィールド	CSS	タグ	スクリプト
function keyword			X
identifier			X
number		X	X
operator			X
brackets		X	X
keywords		X	X

スキームの定義をより柔軟に処理するために、ワイルドカード文字とエスケープ文字を指定することができます。

ワイルドカード文字

Dreamweaver でサポートされるワイルドカード文字のリストを以下に示します。ワイルドカード文字を表すストリングと使用法についても説明します。

ワイルドカード	エスケープストリング	説明
ワイルドカード	*	ワイルドカードの次の文字が見つかるまで、ルール内のすべての文字をスキップします。例えば、<MMTInstance:Editable name="*"> は、このタイプのタグのうち、name 属性が指定されているすべてのタグと一致します。
エスケープ文字を含むワイルドカード	\e*x	x は、エスケープ文字です。 これは、エスケープ文字を指定できる点を除いて、ワイルドカードと同じです。エスケープ文字の次の文字は無視されます。これにより、ストリング内でワイルドカードの後に文字を指定し、ワイルドカード処理を終了する条件には一致しないようにできます。 例えば、/\e*\\/ は、先頭と末尾がスラッシュ（/）で、バックスラッシュ（\）の後にはスラッシュを含めることができる JavaScript の正規表現と一致します。バックスラッシュはコードカラーリングのエスケープ文字であるため、コードカラーリング XML でバックスラッシュを指定する場合は、前にバックスラッシュを指定する必要があります。
オプションのホワイトスペース	\s*	これは、ホワイトスペースなし、任意の数のホワイトスペース、または改行文字と一致します。 例えば、<!--\s*#include は、#include トークンの前にホワイトスペースがあるかどうかに関係なく、ASP インクルードディレクティブに一致します。これは、どちらの場合も条件を満たすためです。 ホワイトスペースワイルドカードは、ホワイトスペースと改行文字の任意の組み合わせと一致します。
必須のホワイトスペース	\s+	これは、ホワイトスペース、または改行文字と一致します。 例えば、<!--#include\s+virtual は、#include と virtual の間にホワイトスペースの任意の組み合わせが含まれる ASP インクルードディレクティブと一致します。ホワイトスペースは、これらのトークンの間に指定する必要がありますが、有効なホワイトスペース文字を任意に組み合わせることができます。 ホワイトスペースワイルドカードは、ホワイトスペースと改行文字の任意の組み合わせと一致します。

エスケープ文字

Dreamweaver でサポートされるエスケープ文字のリストを以下に示します。エスケープ文字を指定するストリングと使用法についても説明します。

エスケープ文字	エスケープストリング	説明
バックスラッシュ	\\	バックスラッシュ（\）は、コードカラーリングのエスケープ文字です。このため、コードカラーリングルールで指定する場合は、エスケープする必要があります。
ホワイトスペース	\s	このエスケープ文字は、改行エスケープ文字とは一致しないスペースやタブなどの非表示文字と一致します。 オプションのホワイトスペースと必須のホワイトスペースのワイルドカードは、ホワイトスペース文字と改行文字の両方に一致します。
改行	\n	このエスケープ文字は、改行文字およびキャリッジリターン文字と一致します。

ストリングの最大長

1 つのデータストリングに許される長さの最大値は 100 文字です。例えば、次の blockEnd タグには、ワイルドカード文字が含まれています。

```
<blockEnd><![CDATA[<!--\s*#BeginEditable\s*"\*"\s*-->]]></blockEnd>
```

Dreamweaver が自動的に生成するオプションのホワイトスペースのワイルドカードストリング（\s*）がシングルスペース文字であるとすると、このデータストリングの長さは、26 文字に名前のワイルドカードストリング（*）の長さを加えたものになります。

```
<!-- #BeginEditable "\*" -->
```

したがって、編集可能な領域名には、最大で 100 文字から 26 文字を引いた 74 文字を使用できます。

スキームの優先度

Dreamweaver では、次のアルゴリズムを使用してコードビューのテキストシンタックスをカラーリングします。

1 最初のシンタックススキームは、現在のファイルのドキュメントタイプに基づいて決められます。ファイルドキュメントタイプは、scheme.documentType 属性と照合されます。一致するタイプが見つからない場合は、scheme.documentType = "Text" のスキームが使用されます。

2 blockStart… blockEnd ペアが指定されているスキームは、ネストできます。ネストできるスキームのうち、現在のファイル拡張子が blockStart.doctypes 属性のリストのいずれかに含まれているものは現在のファイルで有効であり、それ以外のスキームは無効です。

注意：すべての blockStart/blockEnd の組み合わせは一意である必要があります。

スキームは、scheme.priority の値が外側のスキームの値以上である場合に限り、他のスキーム内にネストできます。優先度が同じ場合、スキームは、外側のスキームの body ステート内にのみネストできます。例えば、<script>...</script> ブロックは、タグを記述できる <html>...</html> ブロック内にのみネストできます。タグ、属性、ストリング、コメントなどの内部にはネストできません。

外側のスキームよりも優先度が高いスキームは、外側のスキーム内であれば、ほとんどあらゆる場所にネストできます。例えば、<html>...</html> ブロックの body ステート内以外に、<%...%> ブロックをタグ、属性、ストリング、コメントなどの内部にもネストできます。

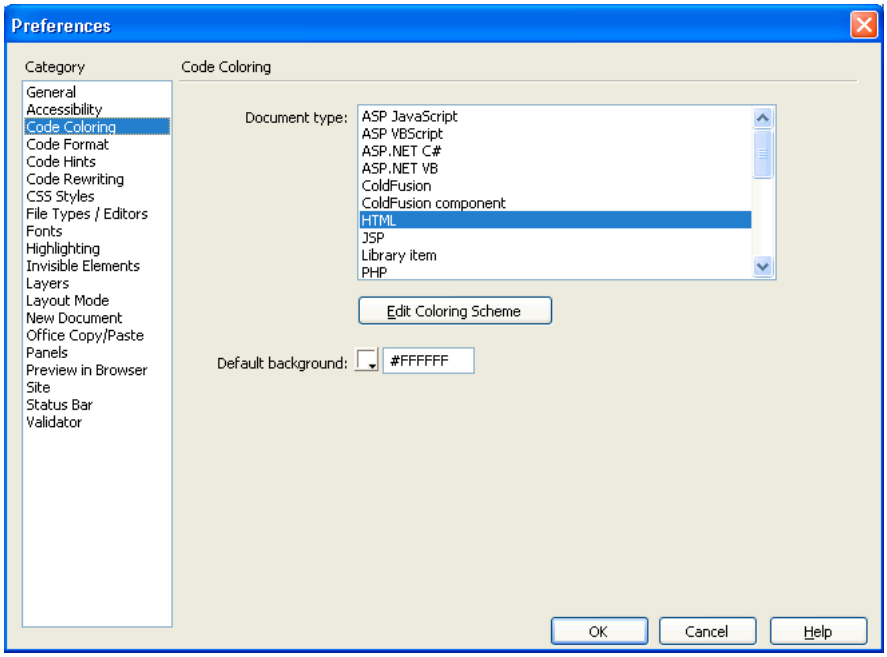
ネストレベルは最大 4 です。

3 blockStart ストリングを照合する場合は、一致するストリングの中で最も長いものが常に使用されます。

4 現在のスキームの blockEnd ストリングに達すると、シンタックスカラーリングは、blockStart ストリングが検出されたステートに戻ります。例えば、<%...%> ブロックが HTML ストリング内に見つかると、カラーリングはその HTML ストリングの色に戻ります。

スキームの編集

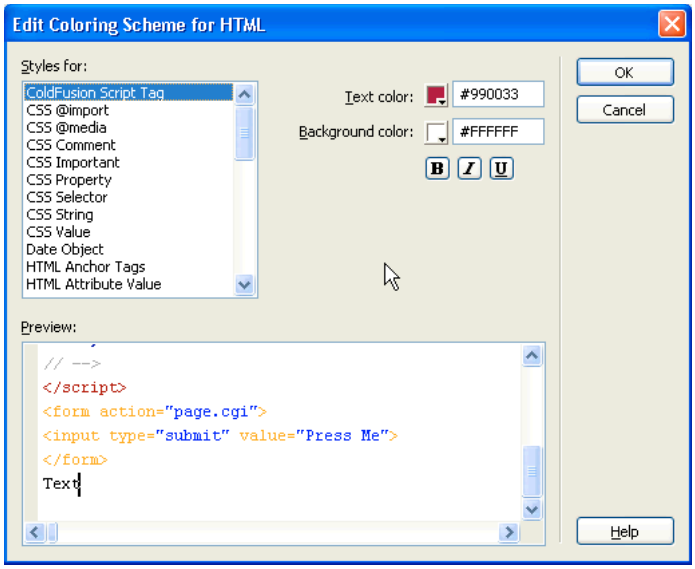
コードカラーリングスキームのスタイルを編集するには、コードカラーリングファイルを編集するか、または、次の図で示すように、Dreamweaver の環境設定ダイアログボックスのコードカラーリングカテゴリーを選択します。



stringStart などの複数回指定できるフィールドの場合は、色とスタイルの設定は最初のタグだけで指定します。色とスタイルの設定を複数のタグに分け、後から環境設定ダイアログボックスを使用して色とスタイルを編集すると、データが失われます。

注意：変更する前に、すべての XML ファイルのバックアップコピーを作成することをお勧めします。環境設定ダイアログボックスで色とスタイルの設定を編集する前に、手動で行ったすべての変更を検証してください。環境設定ダイアログボックスで無効な XML ファイルを編集すると、データが失われます。

環境設定ダイアログボックスのコードカラーリングカテゴリーを使用してスキームのスタイルを編集するには、ドキュメントタイプをダブルクリックするか「カラーリングスキームの編集」ボタンをクリックして、カラーリングスキームの編集ダイアログボックスを表示します。



特定のエレメントのスタイルを編集するには、スタイルリストで編集するエレメントを選択します。スタイルペインに一覧表示される項目には、編集中のスキームのフィールド、およびそのスキーム内のブロックとして表示されるスキームが含まれます。例えば、HTML スキームを編集する場合、CSS ブロックと JavaScript ブロックのフィールドも一覧表示されます。

スキームについて一覧表示されるフィールドは、XML ファイルで定義されているフィールドに対応しています。scheme.name 属性の値は、スタイルペインに一覧表示される各フィールドよりも優先されます。名前のないフィールドは一覧表示されません。

特定エレメントのスタイルまたはフォーマットには、ボールド、イタリック、アンダーライン、背景色、コードカラーリングがあります。スタイルペインでエレメントを選択してから、これらのスタイル特性を変更します。

プレビュー領域には、現在の設定でサンプルテキストがどのように表示されるかが示されます。サンプルテキストは、スキームの sampleText の設定が使用されます。

スタイルリストでの選択を変更するには、プレビュー領域でエレメントを選択します。

スキームのエレメントの設定を変更すると、コードカラーリングファイルに値が格納され、元の設定が上書きされます。「OK」をクリックすると、すべてのコードカラーリングの変更が自動的にリロードされます。

コードカラーリングの例

次に示すコードカラーリングの例は、CSS ドキュメントと JavaScript ドキュメントのコードカラーリングスキームを示します。JavaScript の例のキーワードのリストは、短くするために一部を省略しています。

CSS コードカラーリング

```
<scheme name="CSS" id="CSS" doctypes="CSS" priority="50">
    <ignoreCase>Yes</ignoreCase>
    <ignoreTags>Yes</ignoreTags>
    <blockStart doctypes="ASP-JS,ASP-VB,ASP.NET_CSharp,ASP.NET_VB,ColdFusion,
        CFC,HTML,JSP,LibraryItem,DWTemplate,PHP_MySQL" scheme="outerTag">
        <![CDATA[<style>]]></blockStart>
    <blockEnd><![CDATA[</style>]]></blockEnd>
        <blockStart doctypes="ASP-JS,ASP-VB,ASP.NET_CSharp,ASP.NET_VB,ColdFusion,
        CFC,HTML,JSP,LibraryItem,DWTemplate,PHP_MySQL" scheme="outerTag">
        <![CDATA[<style\s+\*>]]></blockStart>
    <blockEnd><![CDATA[</style>]]></blockEnd>
    <commentStart name="Comment" id="CodeColor_CSSComment"><![CDATA[ /*]]></commentStart>
    <commentEnd><![CDATA[*/]]></commentEnd>
    <endOfLineComment><![CDATA[<!--]]></endOfLineComment>
    <endOfLineComment><![CDATA[-->]]></endOfLineComment>
    <stringStart name="String" id="CodeColor_CSSString"><![CDATA["]]></stringStart>
    <stringEnd><![CDATA["]]></stringEnd>
    <stringStart><![CDATA[']]></stringStart>
    <stringEnd><![CDATA[']]></stringEnd>
    <stringEsc><![CDATA[\]]></stringEsc>
    <cssSelector name="Selector" id="CodeColor_CSSSelector" />
    <cssProperty name="Property" id="CodeColor_CSSProperty" />
    <cssValue name="Value" id="CodeColor_CSSValue" />
    <sampleText doctypes="CSS"><![CDATA[/* Comment */
H2, .head2     {
            font-family : 'Sans-Serif';
            font-weight : bold;
            color : #339999;
        }]]>
    </sampleText>
</scheme>
```

CSS サンプルテキスト

次に示す CSS スキームのサンプルテキストは、CSS コードカラーリングスキームの例です。

```
/* Comment */
H2, .head2     {
    font-family : 'Sans-Serif';
    font-weight : bold;
    color : #339999;
    }
```

次に示す Colors.xml ファイルの行では、サンプルテキストに使用される色とスタイルの値が示されています。これらの値は、コードカラーリングスキームにより割り当てられたものです。

```
<syntaxColor id="CodeColor_CSSSelector" text="#FF00FF" />
<syntaxColor id="CodeColor_CSSProperty" text="#000099" />
<syntaxColor id="CodeColor_CSSValue" text="#0000FF" />
```

JavaScript コードカラーリング

```
<scheme name="JavaScript" id="JavaScript" doctypes="JavaScript" priority="50">
        <ignoreCase>No</ignoreCase>
        <ignoreTags>Yes</ignoreTags>
        <blockStart doctypes="ASP-JS,ASP-VB,ASP.NET_CSharp,ASP.NET_VB,ColdFusion,
            CFC,HTML,JSP,LibraryItem,DWTemplate,PHP_MySQL" scheme="outerTag">
            <![CDATA[<script>]]></blockStart>
        <blockEnd><![CDATA[</script>]]></blockEnd>
        <blockStart doctypes="ASP-JS,ASP-VB,ASP.NET_CSharp,ASP.NET_VB,ColdFusion,
            CFC,HTML,JSP,LibraryItem,DWTemplate,PHP_MySQL" scheme="outerTag">
            <![CDATA[<script\s+\*>]]></blockStart>
        <blockEnd><![CDATA[</script>]]></blockEnd>
        <commentStart name="Comment" id="CodeColor_JavascriptComment">
            <![CDATA[/*]]></commentStart>
        <commentEnd><![CDATA[*/]]></commentEnd>
        <endOfLineComment><![CDATA[//]]></endOfLineComment>
        <endOfLineComment><![CDATA[<!--]]></endOfLineComment>
        <endOfLineComment><![CDATA[-->]]></endOfLineComment>
        <stringStart name="String" id="CodeColor_JavascriptString">
            <![CDATA["]]></stringStart>
        <stringEnd><![CDATA["]]></stringEnd>
        <stringStart><![CDATA[']]></stringStart>
        <stringEnd><![CDATA[']]></stringEnd>
        <stringEsc><![CDATA[\]]></stringEsc>
        <brackets name="Bracket" id="CodeColor_JavascriptBracket">
            <![CDATA[{[()]}]]></brackets>
        <operators name="Operator" id="CodeColor_JavascriptOperator">
            <![CDATA[+-*/%<>!?:=&|^]]></operators>
        <numbers name="Number" id="CodeColor_JavascriptNumber" />
        <regexp name="RegExp" id="CodeColor_JavascriptRegexp" delimiter="/" escape="\\">
            <searchPattern><![CDATA[(\s*/\e*\\/]]></searchPattern>
            <searchPattern><![CDATA[=\s*/\e*\\/]]></searchPattern>
        </regexp>
        <idChar1>_$abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ</idChar1>
        <idCharRest name="Identifier" id="CodeColor_JavascriptIdentifier">
            _$abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789</idCharRest>
        <functionKeyword name="Function Keyword" id="CodeColor_JavascriptFunction">
            function</functionKeyword>
        <keywords name="Reserved Keywords" id="CodeColor_JavascriptReserved">
            <keyword>break</keyword>
. . .
        </keywords>
        <keywords name="Native Keywords" id="CodeColor_JavascriptNative">
            <keyword>abs</keyword>
. . .
```

```
        </keywords>
        <keywords id="CodeColor_JavascriptNumber">
            <keyword>Infinity</keyword>
            <keyword>Nan</keyword>
        </keywords>
        <keywords name="Client Keywords" id="CodeColor_JavascriptClient">
            <keyword>alert</keyword>
. . .
        </keywords>
        <sampleText><![CDATA[/* JavaScript */
function displayWords(arrayWords) {
    for (i=0; i < arrayWords.length(); i++) {
        // inline comment
        alert("Word " + i + " is " + arrayWords[i]);
    }
}
var tokens = new Array("Hello", "world");
displayWords(tokens);
]]></sampleText>
    </scheme>
```

JavaScript サンプルテキスト

次に示す JavaScript スキームのサンプルテキストは、JavaScript コードカラーリングスキームの例です。

```
/* JavaScript */ function displayWords(arrayWords) {
    for (i=0; i < arrayWords.length(); i++) {
        // inline comment
        alert("Word " + i + " is " + arrayWords[i]);
    }
}
var tokens = new Array("Hello", "world");
displayWords(tokens);
```

次に示す Colors.xml ファイルの行では、サンプルテキストに使用される色とスタイルの値が示されています。これらの値は、コードカラーリングスキームにより割り当てられたものです。

```
<syntaxColor id="CodeColor_JavascriptComment" text="#999999" italic="true" />
<syntaxColor id="CodeColor_JavascriptFunction" text="#000000" bold="true" />
<syntaxColor id="CodeColor_JavascriptBracket" text="#000099" bold="true" />
<syntaxColor id="CodeColor_JavascriptNumber" text="#FF0000" />
<syntaxColor id="CodeColor_JavascriptClient" text="#990099" />
<syntaxColor id="CodeColor_JavascriptNative" text="#009999" />
```

コード検証について

コードビューでドキュメントを開くと、ユーザが選択したターゲットブラウザで使用できないタグ、属性、CSS プロパティ、または CSS の値が、開いたドキュメントで使用されていないことが自動的に検証されます。エラー個所には、赤い曲線のアンダーラインが付けられます。

また、Dreamweaver は、新しいブラウザ互換性チェック機能を備えています。これにより、ブラウザ間のレンダリング問題を引き起こす可能性のある HTML と CSS の組み合わせを検索します。

ブラウザプロファイルは、Dreamweaver の Configuration フォルダにある Browser Profile フォルダに保存されます。各ブラウザプロファイルは、ブラウザ名に基づく名前のテキストファイルとして定義されます。例えば、Internet Explorer バージョン 6.0 のブラウザプロファイルは Internet_Explorer_6.0.txt です。CSS を使用する準備としてターゲットブラウザを確認するサポートのために、ブラウザの CSS プロファイル情報がブラウザプロファイルに対応する名前と _CSS.xml の

サフィックスを持つ XML ファイルに保存されます。例えば、Internet Explorer 6.0 の CSS プロファイルは、Internet_Explorer_6.0_CSS.xml に保存されます。確認時にエラーがレポートされた場合は CSS プロファイルファイルを変更できます。

CSS プロファイルファイルは、3 つの XML タグ、css-support、property および value から構成されます。次の節では、これらのタグについて説明します。

<css-support>

説明
このタグは、特定のブラウザでサポートされている property タグと value タグのセットのルートノードです。

属性
なし。

コンテンツ
property タグと value タグ。

コンテナ
なし。

例

```
<css-support>
. . .
</css-support>
```

<property>

説明
対象のブラウザプロファイルに、サポートされる CSS プロパティを定義します。

属性
name、names、supportlevel、message

- name="***property_name***"。サポートを指定するプロパティの名前。
- names="***property_name*, property_name,..."**。サポートを指定するプロパティ名のカンマで区切られたリスト。

  names 属性は、ショートハンドの一種です。例えば、次の names 属性は、それに続く name 属性を定義するショートハンドメソッドです。

```
<property names="foo,bar">
    <value type="named" name="top"/>
    <value type="named" name="bottom"/>
</property>
<property name="foo">
    <value type="named" name="top"/>
    <value type="named" name="bottom"/>
</property>
<property name="bar">
    <value type="named" name="top"/>
    <value type="named" name="bottom"/>
</property>
```

- supportlevel="error"、"warning"、"info"、または "supported"。プロパティをサポートするレベルを指定します。指定しない場合は、"supported" であると見なされます。「supported」以外のサポートレベルを指定し、メッセージ属性を省略すると、デフォルトのメッセージ「CSS プロパティ **property_name** はサポートされていません。」が使用されます。
- message="***message_string***"。message 属性は、ドキュメントでプロパティが検出されたときに表示されるメッセージストリングを定義します。メッセージストリングでは、そのプロパティ値に関して考えられる制限事項または回避策が説明されます。

コンテンツ
value

コンテナ
css-support

例
```
<property name="background-color" supportLevel="supported">
```

<value>

説明
現在のプロパティでサポートされる値のリストを定義します。

属性
type、name、names、supportlevel、message

- type="any"、"named"、"units"、"color"、"string"、または "function"。値のタイプを指定します。"named"、"units"、または "color" を指定する場合は、name 属性か names 属性で、この項目に適合する値 ID を指定する必要があります。"units" 値は、数値の後に names 属性で指定した units 値のいずれかが続く値と一致します。
- name="***value_name***"。CSS 値識別子。ハイフン（-）以外のスペースや句読点は使用できません。CSS プロパティで有効な値のいずれかの名前は、プロパティの親ノードで指定されます。これによって特定の値または units 指定子を識別できます。
- names="***name1, name2, . .***"。値 ID のカンマ区切りのリストを指定します。
- supportlevel="error"、"warning"、"info"、または "supported"。この値に対するブラウザでのサポートレベルを指定します。指定しない場合は、"supported" であると見なされます。
- message="***message_string***"。message 属性は、ドキュメントでプロパティ値が検出されたときに表示されるメッセージストリングを定義します。message 属性が省略されると、「**value_name** はサポートされていません。」というメッセージストリングが表示されます。

コンテンツ
なし。

コンテナ
property

例

```
<property name="margin">
    <value type="units" name="ex" supportlevel="warning"
        message="The implementation of ex units is buggy in Safari 1.0."/>
    <value type="units" names="%,em,px,in,cm,mm,pt,pc"/>
    <value type="named" name="auto"/>
    <value type="named" name="inherit"/>
</property>
```

デフォルトの HTML フォーマットの変更

全般的なコードフォーマットの環境設定を変更するには、環境設定ダイアログボックスのコードフォーマットカテゴリーを使用します。特定のタグと属性のフォーマットを変更するには、タグライブラリエディタ（編集／タグライブラリ）を使用します。詳しくは、Dreamweaver のヘルプメニューの「**Dreamweaver** の使い方」を参照してください。

タグライブラリの設定フォルダのサブフォルダにある、タグに対応する VTM ファイルを編集することにより、タグのフォーマットを編集することもできます。ただし、Dreamweaver 内でフォーマットを変更するほうが簡単です。

VTM ファイルを追加または削除するには、TagLibraries.vtm ファイルを編集する必要があります。Dreamweaver では、TagLibraries.vtm ファイルに示されていない VTM ファイルは無視されます。

注意：このファイルは、Dreamweaver ではなくテキストエディタで編集してください。

左右分割ビューについて

左右分割ビュー機能により、コードビューとデザインビュー、またはコードビューとコードレイアウトモードを左右に並べて表示できます。2 台のモニタによるワークステーション構成を使用している場合、この機能を利用して、一方のモニタにコードを表示しながら、もう一方のモニタにデザインビューを表示して作業することができます。

左右分割ビュー機能を使って、次の操作を行えます。

- コードビューとデザインビューの方向（水平または垂直）を選択できます。
- コードビューとデザインビュー、およびスプリットコードビューの方向を水平または垂直に切り替えられます。

Dreamweaver を再起動してドキュメントを開くか作成すると、前回使用されたサイズと方向でコードビューとデザインビューが開き、ドキュメントが表示されます。dreamweaver.setSplitViewOrientation() 関数を使用して方向を設定し、dreamweaver.setPrimaryView() 関数を使用してプライマリビューを設定します。これらの関数の使用方法について詳しくは、『Dreamweaver API リファレンス』の「左右分割ビュー関数」を参照してください。

関連ファイルについて

関連ファイル機能を利用して、作業中のファイルに関連付けられているサポートファイルと関連ファイルにアクセスできます。関連ファイルには、CSS ファイル、スクリプトファイル、サーバサイドインクルード（SSI）ファイル、XML ファイルなどがあります。

例えば、CSS ファイルがメインファイルに関連付けられている場合、この CSS ファイルの表示と編集にこの機能を活用できます。さらに、関連ファイルを編集しながらメインファイルも表示できます。

関連ファイルの動作

関連ファイルを利用することで、次のような操作を実行でき、編集作業の効率が高まります。

- メインファイルを表示しながら関連ファイルを表示し、アクセスすることができます。関連ファイル（CSS ファイルなど）を含むページは、次のように表示できます。
  - 一方にページのデザイン
  - もう一方に関連ファイル
- 関連ファイルバーには、親 HTML の生成に影響するドキュメントが表示されます。ソース HTML、生成 HTML、第 1 レベルの子ドキュメントを表示できます。
- 関連ファイルバーに表示される関連ファイルを選択すると、次の操作を実行できます。
  - コードビューに関連ファイルを表示し、編集できます。
  - デザインビューに親ページを表示できます。
- デザインビューでコンテンツを選択して、関連ファイルに変更を加え、デザインビューを更新しても、選択は維持されます。
- 関連ファイルのコードを変更すると、その変更がデザインビューに反映されます。

ファイルが見つからない場合、空白のウィンドウ枠の上部のバーに、ファイルが見つからないという内容のメッセージが表示されます。

関連ファイルの用語

関連ファイルに関してよく使用される用語を以下で説明します。

用語	説明	例
最上位のドキュメント	ユーザが開く任意のドキュメント	
親ドキュメント	デザインビューに表示される任意の最上位ドキュメント	• HTML — .lbi、.dwt を含む • CFML • PHP

用語	説明	例
第 1 レベルの子ドキュメント	親ドキュメントから 1 レベル下にある任意のドキュメント。このドキュメントは HTML コードの生成に影響します（CSS は例外）。CSS ファイルには他の CSS ファイルもインクルードできますが、ページに適用される最終的なスタイルはすべての CSS ファイルによって決定されます。	• <SCRIPT src="file.js"> によって指定されるスクリプトファイル • サーバサイドインクルード • 外部 CSS • Spry XML および HTML データセット • ライブラリ項目 • <iframe> - リモートソース • Object タグ
下位レベルの子ドキュメント	親ドキュメントから 2 レベル以上離れた任意のドキュメント。このドキュメントは HTML コードの生成に影響します。	• PHP 内の PHP • DTD • テンプレート
非関連ファイル	HTML コードの生成に影響しない任意のドキュメント、またはユーザが現在編集していない任意のファイル。	• イメージファイル • メディアファイル • <a> タグによって外部リンクされているファイル

サポートされている関連ファイルは以下のとおりです。

タイプ	説明	ネストレベル
クライアントサイドスクリプト	全言語	1 （スクリプトのネストは不可）
サーバサイドインクルード	次の拡張可能条件がすべて満たされる場合 • サーバモデルが定義済み • SSI ステートメント （パターン）が定義済み • DW doctype が定義済み 例外：HTML ドキュメントでは、Apache スタイルのインクルードファイルステートメント（<!--#include ... -->）は認識されます。	1
Spry データセット		1 （スクリプトのネストは不可）
CSS	• すべてのメディアタイプに対応するすべての外部 CSS • DTSS	無制限

関連ファイル API

関連ファイルメニューをカスタマイズすることで、次の内容を表示できます。

- 関連ファイルのファイル名
- ソース HTML と生成ソースコード

dreamweaver.openRelatedFile() 関数を使用すると関連ファイルがコードビューに表示され、dreamweaver.getActiveRelatedFilePath() 関数を使用すると現在開いている関連ファイルのパスが表示されます。これらの API の使用方法について詳しくは、『Dreamweaver API リファレンス』の「関連ファイル関数」を参照してください。

ライブビューについて

ライブビュー機能を使用すると、Dreamweaver を終了することなく、ブラウザで表示されるときの Web ページをプレビューすることができます。ユーザはそのままコードに直接アクセスして、コードを編集できます。コードに加えられた変更はすぐに反映されます。そのため、変更した Web ページをすぐに表示することができます。CSS ファイルを編集すると、ファイルの現在の状態は維持されますが、CSS の変更は適用されます。ユーザは Dreamweaver から Web ブラウザに切り替えなくても、ページを編集しながらロールオーバーなどの JavaScript 効果を確認することができます。

ライブビュー機能ではシステムの Flash プラグイン（%SYSTEM%¥Macromed¥Flash、¥Library¥Internet Plug-Ins¥）が使用されます。システムバージョンのプラグインが使用できない場合、任意の Firefox バージョンにフォールバックすることがあります。

プラグインが見つからない場合、情報バーに通知が表示されます。CSS パネルには、ソースが別の位置で生成されている場合でも、ライブビューの表示内容に適用されている現在の CSS が常時表示されます。ユーザはスタイルシートの追加や削除を実行できます。ただし、インライン CSS または <head> タグ内の CSS に対するその他の編集はロックされています。CSS パネルでは、編集不可のルールは読み取り専用としてマークされています。

Dreamweaver API は次の目的に使用できます。

- デザインビューモードの取得と設定
- サーバを使用したライブビューモードの取得と設定
- ライブビューの初期設定の取得
- ライブビューの依存ファイルの取得と設定
- ライブビューのパラメータの表示

これらの API について詳しくは、『Dreamweaver API リファレンス』の「ライブビュー関数」を参照してください。

簡単なライブビューの例

この例では、Dreamweaver を使用して、ユーザがコマンドメニューのコマンドをクリックしたときに単純なブラウザを作成するコマンドを作成します。この例を試す前にコマンドの作成に関する詳細を確認するには、142 ページの 130 ページの「コマンド」を参照してください。Dreamweaver で基本の HTML ファイルを新規に開きます。これが、コマンド定義ファイルになります。このファイルに liveviewexample.htm という名前を付けて保存します。コマンド定義ファイルは、次の例に示すような内容になります。

```
<!DOCTYPE HTML SYSTEM "-//Adobe//DWExtension layout-engine 10.0// dialog"> <html
xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Browser Test</title>
<script type="text/javascript">
var browserLoad = function(e)
{
    var uri = document.getElementById("uri");
    uri.value = e.currentBrowserLocation;
}
var promptToNavigate = function (e)
{
    if( ! confirm(" Is it ok to go from \n" + e.currentBrowserLocation + " to page \n " +
e.requestedBrowserLocation ) )
    {
        e.preventDefault();
    }
}
function initUI()
{
    var browser = document.getElementById("browser");
    browser.addEventListener("BrowserControlBeforeNavigation",
promptToNavigate, true);
    browser.addEventListener("BrowserControlLoad", browserLoad, true);
    browser.openURL("http://www.adobe.com");
}
function loadUri()
{
    var uri = document.getElementById("uri");
    var browser = document.getElementById("browser");
    browser.openURL(uri.value);
}
function showSource(){
    var browser = document.getElementById("browser");
    alert(browser.getWindowObj().document.documentElement.outerHTML);
}
function commandButtons() {
                            return new Array( "Close", "window.close()",
                            "Load URI", "loadUri()",
                            "Show Source", "showSource()"
);
}
</script>
</head>
<body onLoad="initUI()">
<form>
<p>
<label>
<input id="uri" type="text" style="width:500px">
</label>
</p> <mm:browsercontrol id="browser" style="width:500px; height:300px;"/> </form> </body> </html>
```

第 4 章：Dreamweaver の拡張

Dreamweaver には機能の追加やカスタマイズのためのツールが数多く用意されています。

Dreamweaver 拡張機能を作成するときは、2 ページの「拡張機能の作成」で説明されている手順に従います。

Dreamweaver では次の機能を使用して拡張機能を作成できます。

- HTML パーサー（レンダラーとも呼ばれます）。拡張機能のユーザインターフェイス（UI）をデザインできます。パーサーは、フォームフィールド、絶対位置のエレメント、イメージおよびその他の HTML エレメントを使用します。Dreamweaver には、独自の HTML パーサーがあります。
- フォルダのツリー。Dreamweaver のエレメントと拡張機能を実装および設定するファイルを整理、保存します。
- 一連の API（アプリケーションプログラミングインターフェイス）。JavaScript を介して Dreamweaver の機能にアクセスできます。
- JavaScript インタープリタ。拡張機能ファイル内の JavaScript コードを実行します。Dreamweaver では、Netscape JavaScript バージョン 1.5 インタープリタが使用されます。このバージョンと以前のバージョンのインタープリタ間での変更点について、詳しくは 78 ページの「Dreamweaver の拡張機能での JavaScript の処理方法」を参照してください。

Dreamweaver 拡張機能の種類

次に、Dreamweaver の各種の拡張機能について説明します。

挿入バーオブジェクト 挿入バーに変更を加えます。オブジェクトは、通常、ドキュメントへのコードの挿入を自動化するために使用されます。また、ユーザ入力を収集するフォームや、入力を処理する JavaScript を含めることもできます。オブジェクトファイルは、Configuration¥Objects フォルダに格納されています。

コマンド ほぼすべてのタスクを実行できます。ユーザによる入力はある場合もない場合もあります。通常、コマンドファイルはコマンドメニューから呼び出しますが、他の拡張機能から呼び出すこともできます。コマンドファイルは、Configuration¥Commands フォルダに格納されています。

メニューコマンド コマンド API を拡張して、メニューからのコマンドの呼び出しに関連するタスクを実行できるようにします。メニューコマンド API では、動的サブメニューも作成できます。

ツールバー Dreamweaver ユーザインターフェイスの既存のツールバーへのエレメントの追加や、新しいツールバーの作成を実行できます。新しいツールバーは、デフォルトのツールバーの下に表示されます。ツールバーファイルは、Configuration¥Toolbars フォルダに格納されています。

レポート 独自のサイトレポートの追加や、Dreamweaver に付属しているレポートの修正を実行できます。結果ウィンドウ API を使用して、スタンドアローンレポートを作成することもできます。

タグライブラリおよびエディタ 関連付けられたライブラリファイルと共に動作します。タグライブラリおよびエディタ拡張機能を使用すると、既存のタグダイアログの属性の修正、新しいタグダイアログの作成、タグライブラリへのタグの追加を実行できます。タグライブラリおよびエディタ拡張機能ファイルは、Configuration¥TagLibraries フォルダに格納されています。

プロパティインスペクタ プロパティインスペクタパネルに表示されます。Dreamweaver の大部分のインスペクタは、中核的な製品コードの構成要素です。これらのインスペクタは変更できません。ただし、カスタムプロパティインスペクタファイルは、組み込まれている Dreamweaver プロパティインスペクタインターフェイスをオーバーライドしたり、カスタムタグを検査する新しいインスペクタを作成することができます。インスペクタは、Configuration¥Inspectors フォルダに格納されています。

フローティングパネル フローティングパネルを Dreamweaver ユーザインターフェイスに追加します。フローティングパネルでは、選択範囲、ドキュメント、タスクなどを操作できます。また、役立つ情報を表示できます。フローティングパネルファイルは、Configuration¥Floaters フォルダに格納されています。

ビヘイビア JavaScript コードをドキュメントに追加できます。JavaScript コードにより、ブラウザでドキュメントが表示されているときのイベントに応答する特定のタスクが実行されます。ビヘイビア拡張機能は、Dreamweaver のビヘイビアパネルの + メニューに表示されます。ビヘイビアファイルは、Configuration¥Behaviors¥Actions フォルダに格納されています。

サーバビヘイビア サーバサイドコード（ASP、または ColdFusion）のブロックをドキュメントに追加します。サーバサイドコードにより、ドキュメントをブラウザで表示している間にサーバ上でタスクが実行されます。サーバビヘイビアは、Dreamweaver の サーバビヘイビアパネルの + メニューに表示されます。サーバビヘイビアファイルは、Configuration¥ServerBehaviors フォルダに格納されています。

データソース データベースに保存されている動的データへの接続を構築できます。データソース拡張機能は、バインディングパネルの + メニューに表示されます。データソース拡張機能ファイルは、Configuration¥Data Sources フォルダに格納されています。

サーバフォーマット 動的データのフォーマットを定義できます。

コンポーネント コンポーネントパネルに新しい種類のコンポーネントを追加できます。コンポーネントは、ColdFusion コンポーネント（CFC）など、いくつかの一般的で最新のカプセル化方策に対して Dreamweaver で使用される用語です。

サーバモデル 新しいサーバモデルのサポートを追加できます。Dreamweaver では、最も一般的なサーバモデル（ASP、JSP、ColdFusion、PHP および ASP.NET）がサポートされています。サーバモデル拡張機能は、カスタムサーバソリューション、異なる言語またはカスタマイズされたサーバでのみ必要です。サーバモデルファイルは、Configuration¥ServerModels フォルダに格納されています。

データトランスレータ HTML 以外のコードを HTML コードに変換します。これは、ドキュメントウィンドウのデザインビューで表示されます。これらの拡張機能は、HTML 以外のコードをロックし、Dreamweaver 側で解析されないようにします。トランスレータファイルは、Configuration¥Translators フォルダに格納されています。

Dreamweaver を拡張する他の方法

Dreamweaver の次のエレメントの機能を拡張したり、用途に合わせて調整することができます。

ドキュメントタイプ Dreamweaver で様々なサーバモデルを扱うしくみを定義します。サーバモデルのドキュメントタイプに関する情報は、Configuration¥DocumentTypes フォルダに格納されています。詳しくは、13 ページの「Dreamweaver で拡張可能なドキュメントタイプ」を参照してください。

コードスニペット 再利用可能なコードブロックです。Dreamweaver の Configuration¥Snippets フォルダに CSN（コードスニペット）ファイルとして保存され、スニペットパネルからアクセスできます。追加のコードスニペットファイルを作成して Snippets フォルダにインストールし、利用可能にすることができます。

コードヒント このメニューを使用すると、入力しようとしているストリングの候補のリストが提示されるため、入力の手間を省くことができます。入力しようとしているストリングがメニューに表示されている場合は、そのストリングを選択すると、入力している場所に挿入されます。コードヒントメニューは、Configuration¥CodeHints フォルダの codehints.xml ファイルで定義されます。このファイルに、新しく定義したタグまたは機能の新しいコードヒントメニューを追加できます。

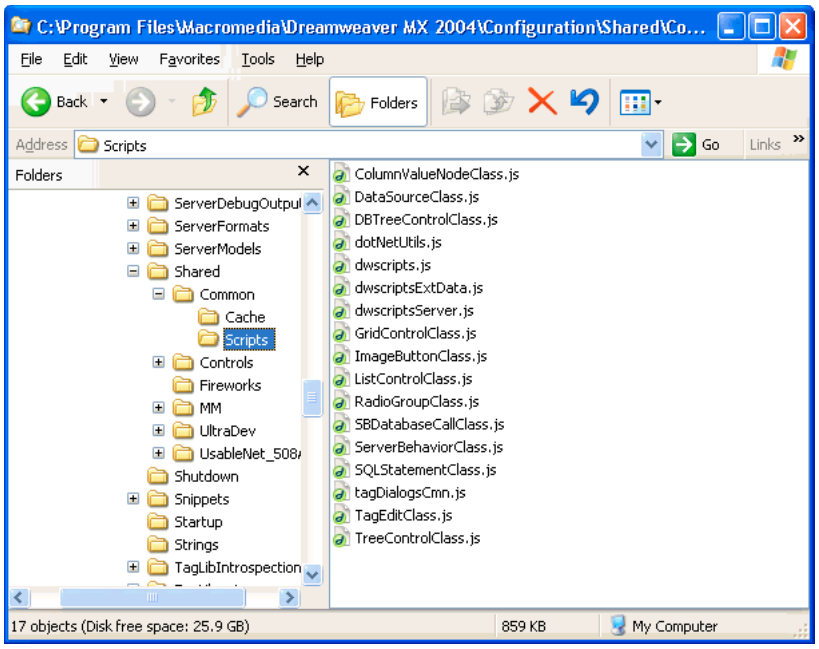
メニュー Configuration¥Menus フォルダの menus.xml ファイルで定義されます。拡張機能に使用する新しい Dreamweaver メニューを追加するには、それらのメニュータグを menus.xml ファイルに追加します。詳しくは、139 ページの「メニューおよびメニューコマンド」を参照してください。

設定フォルダと拡張機能

Dreamweaver の Configuration フォルダに格納されているフォルダおよびファイルには、Dreamweaver に付属の拡張機能が含まれています。拡張機能を作成したら、Dreamweaver 側で認識されるように、それらのファイルを適切なフォルダに保存する必要があります。例えば、プロパティインスペクタの拡張機能を作成する場合は、Configuration¥Inspectors フォルダにファイルを保存します。Adobe Exchange Web サイト（www.adobe.com/go/exchange）から拡張機能をダウンロードしてインストールした場合は、Extension Manager により拡張機能ファイルが適切なフォルダに自動的に保存されます。

Dreamweaver の Configuration フォルダ内のファイルを、サンプルとして使用できます。ただし、これらのファイルの内容は一般に、Adobe Exchange Web サイトから入手可能な通常の拡張機能より複雑です。Configuration フォルダの各サブフォルダの内容について詳しくは、Configuration_ReadMe.htm ファイルを参照してください。

Configuration¥Shared フォルダは、特定の拡張機能タイプ専用ではありません。このフォルダは、複数の拡張機能によって使用されるユーティリティ関数、クラスおよびイメージを集中的に格納するフォルダです。Configuration¥Shared¥Common フォルダ内のファイルは、様々な拡張機能で役立つように設計されています。これらのファイルは、JavaScript 手法の例やユーティリティとして使用できます。オブジェクトに対する有効なドキュメントオブジェクトモデル（DOM）参照の作成、現在の選択範囲が特定のタグ内であるかどうかのテスト、ストリング内の特殊文字のエスケープなど、特定のタスクを実行する関数については、まずこのフォルダを調べてください。共通ファイルを作成する場合は、次の図に示されているように、Configuration¥Shared¥Common フォルダ内に別のサブフォルダを作成し、そこにファイルを保存する必要があります。



Configuration¥Shared¥Common¥Scripts のフォルダ構造

フォルダについて詳しくは、360 ページの「Shared フォルダ」を参照してください。

マルチユーザ設定フォルダ

Windows XP、Windows 2000 および Macintosh OS X のマルチユーザオペレーティングシステムの場合、Dreamweaver によって Dreamweaver の Configuration フォルダの他に、ユーザごとに別の設定フォルダが作成されます。Dreamweaver または JavaScript 拡張機能により Configuration フォルダに対して書き込みが行われると、代わりにユーザ設定フォルダに自動的に書き込まれます。これにより、他のユーザのカスタマイズ設定を変更することなく、各

Dreamweaver ユーザが設定をカスタマイズできます。詳しくは、10 ページの「マルチユーザ環境での Dreamweaver のカスタマイズ」および『Dreamweaver API リファレンス』の「ファイルアクセスおよびマルチユーザ設定 API」を参照してください。

起動時または終了時におけるスクリプトの実行

コマンドファイルを Configuration¥Startup フォルダに配置すると、Dreamweaver の起動時にそのコマンドが実行されます。スタートアップコマンドは、menus.xml ファイル、ThirdPartyTags フォルダ内のファイル、およびその他すべてのコマンド、オブジェクト、ビヘイビア、インスペクタ、フローティングパネルまたはトランスレータの前に読み込まれます。これらのスタートアップコマンドを使用して、menus.xml ファイルやその他の拡張機能ファイルを修正できます。警告の表示、ユーザに対する情報入力の要求、または dreamweaver.runCommand() 関数の呼び出しを行うことができます。ただし、Startup フォルダ内から、有効なドキュメントオブジェクトモデル（DOM）を要求するコマンドを呼び出すことはできません。Dreamweaver DOM について詳しくは、94 ページの「Dreamweaver ドキュメントオブジェクトモデル」を参照してください。

これと同様に、コマンドファイルを Configuration¥Shutdown フォルダに配置すると、そのコマンドは Dreamweaver の終了時に実行されます。シャットダウンコマンドから、dreamweaver.runCommand() 関数の呼び出し、警告の表示、ユーザに対する情報入力の要求を行うことができます。ただし、シャットダウンプロセスを中止することはできません。

コマンドについて詳しくは、130 ページの「コマンド」を参照してください。dreamweaver.runCommand() 関数について詳しくは、『Dreamweaver API リファレンス』を参照してください。

拡張機能のリロード

Dreamweaver での作業中に拡張機能を編集する場合は、拡張機能をリロードして Dreamweaver に変更を認識させます。

拡張機能のリロード

1 挿入パネルのタイトルバーで、Ctrl キー（Windows）または Option キー（Macintosh）を押しながらカテゴリーメニューをクリックします。

注意：このオプションは、タブモードで操作しているときは表示されません。タブモードの場合は、パネルメニュー（左上）を右クリックします。「メニューとして表示」を選択し、Ctrl キーを押しながら「一般」をクリックして、「拡張機能のリロード」コマンドのあるメニューを表示します。

2 「拡張機能のリロード」を選択します。

注意：マルチユーザオペレーティングシステムでは、マスターの設定ファイルを編集するのではなく、ユーザの Configuration フォルダ内の設定ファイルのコピーを編集します。詳しくは、76 ページの「設定フォルダと拡張機能」を参照してください。

拡張機能 API

拡張機能 API は、各タイプの拡張機能を実装するために Dreamweaver で呼び出される関数を提供します。拡張機能タイプごとの記述に従って関数のボディを作成し、Dreamweaver が想定する戻り値を指定する必要があります。

C プログラミング言語で直接開発する場合は、DLL（ダイナミックリンクライブラリ）を作成するための C 拡張機能 API を使用できます。これらの API で提供される機能により、作成した C DLL が JavaScript の中にラップされるため、拡張機能は Dreamweaver 内でシームレスに機能します。

拡張機能 API のマニュアルには、各関数が呼び出されたときの動作と想定している戻り値について、概要が記載されています。

ユーティリティ API および JavaScript API は、拡張機能で特定のタスクを実行するために使用する関数を提供します。ユーティリティ API および JavaScript API について詳しくは、『Dreamweaver API リファレンス』を参照してください。

Dreamweaver の拡張機能での JavaScript の処理方法

Dreamweaver では、起動時に Configuration¥ 拡張機能名フォルダがチェックされます。そのフォルダ内に拡張機能ファイルがある場合、JavaScript は、以下の手順で処理されます。

- SCRIPT の開始タグと終了タグの間のすべてのコードをコンパイルします。
- SCRIPT タグ内のコードで、関数宣言の一部ではないものをすべて実行します。

  注意：グローバル変数の初期化が必要な拡張機能もあるため、この手順は起動時に必ず実行する必要があります。

SRC 属性で SCRIPT タグに指定されている外部 JavaScript ファイルについては、以下のアクションが実行されます。

- ファイルの読み込み
- コードのコンパイル
- プロシージャの実行

注意：拡張機能ファイルのいずれかの JavaScript コードにストリング " が含まれている場合、JavaScript インタープリタはそのストリングを終了スクリプトタグとして読み込み、閉じていないストリングリテラルエラーを報告します。この問題を解決するには、ストリングを小部分に分割し、それらを「<' + '/SCRIPT>」のように連結します。

コマンド拡張機能タイプおよびビヘイビアアクション拡張機能タイプについては、ユーザがメニューからコマンドまたはアクションを選択すると、onLoad イベントハンドラ内のコード（body タグ内に表示されている場合）が実行されます。

ドキュメントの本文にオブジェクト拡張機能のフォームが含まれている場合は、onLoad イベントハンドラ内のコード（body タグ内）が実行されます。

以下の拡張機能の onLoad ハンドラは body タグでは無視されます。

- データトランスレータ
- プロパティインスペクタ
- フローティングパネル

また、すべての拡張機能ファイルにおいて、ユーザがアタッチ先のフォームフィールドを操作すると、Dreamweaver によって他のイベントハンドラのコード（onBlur="alert('This is a required field.')" など）が実行されます。

Dreamweaver では、リンク内のイベントハンドラを使用できます。リンクのイベントハンドラでは、次の例に示すようなシンタックスを使用する必要があります。

```
<a href="#" onMouseDown=alert('hi')>link text</a>
```

プラグイン（常に play に設定）は、拡張機能の BODY でサポートされています。document.write() ステートメント、Java アプレット、Microsoft ActiveX コントロールは、拡張機能ではサポートされていません。

ヘルプの表示

displayHelp() 関数は、いくつかの拡張機能 API に含まれています。この関数を拡張機能に組み込むと Dreamweaver で次の 2 つの処理が行われます。

- インターフェイスに「ヘルプ」ボタンが追加されます。
- ユーザが「ヘルプ」ボタンをクリックすると displayHelp() が呼び出されます。

ヘルプを表示するには、displayHelp() 関数のボディを作成する必要があります。displayHelp() 関数のコードをどのように記述するかによって、拡張機能でのヘルプの表示方法が決まります。dreamweaver.browseDocument() 関数を呼び出すと、警告ボックスの別の絶対位置のエレメントにメッセージ表示するなど、ヘルプの表示方法のカスタマイズや、ブラウザでのファイルの表示を行うことができます。

次の例では、dreamweaver.browseDocument() を呼び出すことにより、displayHelp() 関数を使用してヘルプを表示します。

```
// The following instance of displayHelp() opens a browser to display a file
// that explains how to use the extension.
function displayHelp() {

    var myHelpFile = dw.getConfigurationPath() + "ExtensionsHelp/myExtHelp.htm";
    dw.browseDocument(myHelpFile);
}
```

拡張機能のローカライズ

次の方法を使用すると、拡張機能をローカル言語に簡単にトランスレートできます。

- 拡張機能を HTML ファイルと JavaScript ファイルに分けます。HTML ファイルは、複製してローカライズできます。JavaScript ファイルはローカライズされません。
- JavaScript ファイルでは表示ストリングを定義しないでください。警告と UI コードをチェックします。ローカライズ可能なストリングをすべて Dreamweaver の Configuration¥Strings フォルダ内の個別の XML ファイルに抽出します。
- 必須のイベントハンドラを除き、HTML ファイルでは JavaScript コードを記述しないようにしてください。こうすることで、HTML ファイルが複製され他の複数の言語にトランスレートされた後で、複数のトランスレート結果のバグを何度も修正する必要がなくなります。

XML ストリングファイル

Dreamweaver の Configuration¥Strings フォルダ内の XML ファイルにストリングをすべて保存します。これにより、関連する拡張機能ファイルを多数インストールする場合は、単一の XML ファイルですべてのストリングを共有できます。場合によっては、C++ 拡張機能と JavaScript 拡張機能の両方から同じストリングを参照することもできます。

例えば、myExtensionStrings.xml というファイルを作成できます。次の例に、ファイルの形式を示します。

```
<strings>
    <!-- errors for feature X -->
    <string id="featureX/subProblemY" value="There was a with X when you did Y. Try not to
        do Y!"/>
    <string id="featureX/subProblemZ" value="There was another problem with X, regarding Z.
        Don't ever do Z!"/>
</strings>
```

ここで、JavaScript ファイルは、次の例に示すように dw.loadString() 関数を呼び出すことによって、これらのトランスレート可能なストリングを参照できます。

```
function initializeUI()
{
    ...
    if (problemYhasOccured)
    {
        alert(dw.loadString("featureX/subProblemY");
    }
}
```

ストリング識別子ではスラッシュ（/）を使用できますが、スペースは使用できません。必要に応じて、スラッシュを使用して階層を作成します。また、単一の XML ファイルにすべてのストリングを組み込むことができます。

注意：Configuration¥Strings フォルダ内の cc で始まるファイルは、Contribute ファイルです。例えば、ファイル ccSiteStrings.xml は Contribute ファイルです。

埋め込まれた値を持つローカライズ可能なストリング

表示ストリングの中には、値が埋め込まれているものがあります。errMsg() 関数を使用して、それらのストリングを表示することができます。errMsg() 関数は C の printf() 関数に似ていて、Configuration¥Shared¥MM¥Scripts¥CMN フォルダの string.js ファイル内にあります。プレースホルダー文字としてパーセント記号（%）と s を使用して、ストリング内での値の表示位置を示し、ストリングと変数名を引数として errMsg() に渡します。以下に例を挙げます。

```
<string id="featureX/fileNotFoundInFolder" value="File %s could not be found in folder %s."/>
```

次の例に、ストリングと埋め込まれる変数が alert() 関数に渡されるところを示します。

```
if (fileMissing)
{
    alert( errMsg(dw.loadString("featureX/fileNotFoundInFolder"),fileName,
        folderName) );
}
```

Extension Manager の操作

他のユーザ用の拡張機能を作成している場合は、Adobe Exchange Web サイト（www.adobe.com/go/exchange）のヘルプで How to Create an Extension カテゴリーを選択し、表示されるガイドラインに従って、その拡張機能をパッケージする必要があります。拡張機能を作成し、Extension Manager でテストしたら、ファイル／拡張機能の作成を選択します。拡張機能をパッケージしたら、ファイル／拡張機能の送信を選択して、Extension Manager からパワーアップセンターに拡張機能を送信できます。

Adobe Extension Manager は Dreamweaver に付属しています。使用方法について詳しくは、ヘルプファイルおよび Adobe Exchange Web サイトを参照してください。

第 5 章：拡張機能のユーザインターフェイス

拡張機能の多くは、ユーザからユーザインターフェイス（UI）を介して情報を受け取るように構築されています。例えば、marquee タグに対するプロパティインスペクタ拡張機能を作成する場合は、ユーザが方向や高さなどの属性を指定するための方法を作成する必要があります。作成した拡張機能をアドビ システムズ社に提出して認証を受けるには、Adobe Exchange Web サイト（www.adobe.com/go/exchange）の Extension Manager ファイルに記載されているガイドラインに従う必要があります。このガイドラインは、デベロッパーの創造性を制限するものではありません。認証された拡張機能が Adobe Dreamweaver のユーザインターフェイス（UI）内で効果的に機能すること、および拡張機能の UI のデザインによってその機能が損なわれないようにすることを目的としています。

拡張機能ユーザインターフェイスのデザインに関するガイドライン

拡張機能は、一般に、ユーザが頻繁に行うタスクを実行するために作成します。タスクには繰り返し実行される部分があり、拡張機能を作成することで繰り返しのアクションを自動化できます。タスクの一部の手順が変更されたり、拡張機能が処理するコードの特定の属性が変更されたりすることがあります。このような変数値に対するユーザ入力を受け取るために、ユーザインターフェイスを構築します。

例えば、Web カタログを更新する拡張機能を作成することができます。ユーザは、イメージソース、商品の説明、価格の各値を定期的に変更する必要があります。値は変動しますが、これらの値を取得して Web サイトに表示できるようにフォーマットする処理は変わりません。単純な拡張機能では、フォーマット処理のみが自動化され、イメージソース、商品の説明、価格の新しい更新値はユーザが手動で入力します。より強力な拡張機能では、これらの値がデータベースから定期的に取得されます。

拡張機能ユーザインターフェイスの目的は、ユーザが入力する情報を受け取ることにあります。この情報を使用して、拡張機能で実行する反復タスクの中で変動する部分を処理します。Dreamweaver では、拡張機能ユーザインターフェイスのコントロールを作成するための基本的な構成要素として HTML および JavaScript フォームのエレメントがサポートされており、独自の HTML レンダラーを使用してユーザインターフェイスが表示されます。このため、拡張機能ユーザインターフェイスは、テキスト説明フィールドとフォーム入力フィールドの 2 列の表を含む HTML ファイルのように単純なものにできます。

拡張機能をデザインするときは、必要な変数とそれらを最も効果的に処理できるフォームエレメントを決定する必要があります。

拡張機能ユーザインターフェイスをデザインする際は、以下の基本的なガイドラインを考慮します。

- 拡張機能に名前を付ける場合は、HTML ファイルの title タグにその名前を入力する。この名前は拡張機能のタイトルバーに表示されます。
- テキストラベルをユーザインターフェイスの左側に右揃えで配置し、テキストボックスは右側に左揃えで配置する。このように配置すると、テキストボックスの開始位置をユーザが簡単に把握できます。テキストボックスの後ろには、説明や単位を示す短いテキストを配置できます。
- チェックボックスおよびラジオボタンのラベルは、ユーザインターフェイスの右側に左揃えで配置する。
- 読み取り可能なコードでは、論理名をテキストボックスに割り当てる。Dreamweaver で拡張機能ユーザインターフェイスを作成する場合は、プロパティインスペクタまたはクイックタグ編集を使用して、フィールドに名前を割り当てることができます。

一般的な手順では、ユーザインターフェイス作成後に、拡張機能コードをテストして以下のユーザインターフェイス関連タスクが正常に実行されるかどうかを確認します。

- テキストボックスから値を取得する。
- テキストボックスにデフォルト値を設定する、または選択内容から値を収集する。
- ユーザドキュメントに変更を適用する。

Dreamweaver HTML レンダリングコントロール

Dreamweaver の以前のバージョン（Dreamweaver 4 以前）では、Microsoft Internet Explorer や Netscape Navigator と比較すると、フォームコントロールの周囲により広いスペースがレンダリングされます。Dreamweaver では、独自の HTML レンダリングエンジンを使用して拡張機能ユーザインターフェイスが表示されるので、拡張機能ユーザインターフェイスのフォームコントロールは、その周囲に余分なスペースを空けてレンダリングされます。

これより後のバージョンの Dreamweaver では、フォームコントロールのレンダリングが改善されて、ブラウザでの表示に近づきました。改善されたレンダリングを活用するには、次の例で示すように、3 種類の新しい DOCTYPE ステートメントのいずれかを拡張機能ファイルで使用します。

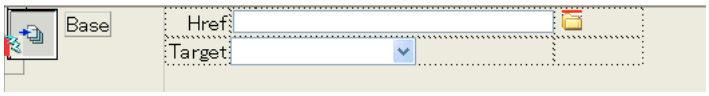
```
<!DOCTYPE HTML SYSTEM "-//Adobe//DWExtension layout-engine 10.0//dialog">
<!DOCTYPE HTML SYSTEM "-//Adobe//DWExtension layout-engine 10.0//floater">
<!DOCTYPE HTML SYSTEM "-//Adobe//DWExtension layout-engine 10.0//pi">
```

ほとんどの場合、DOCTYPE ステートメントはドキュメントの先頭行に配置する必要がありますが、拡張機能固有のディレクティブとの競合を避けるために、DOCTYPE ステートメントとディレクティブは任意の順序で配置できるようになりました。ただし、どちらも HTML 開始タグの前に配置する必要があります。以前のバージョンでは、拡張機能固有のディレクティブをファイルの 1 行目に記述する必要がありました。その一例が、プロパティインスペクタファイルの先頭にあるコメント、つまりコマンド内の MENU-LOCATION=NONE ディレクティブです。

新しい DOCTYPE ステートメントを使用すると、作成した拡張機能を Dreamweaver のデザインビューで表示できます。拡張機能は、実際にユーザに表示されるときと同じように表示されます。

ダイアログの 3 とおりの使用例を、Configuration¥Commands フォルダ内の CFLoginWizard.htm、TeamAdminDlgDates.html、TeamAdminDlgDays.html の各ファイルで確認できます。

次の例では、最初にフォームコントロールのレンダリングを改善する DOCTYPE ステートメントを使用しないベースプロパティインスペクタを示し、次に DOCTYPE ステートメントを使用するベースプロパティインスペクタを示します。



拡張機能でのカスタムユーザインターフェイスコントロール

Dreamweaver では、標準の HTML フォームエレメントに加えて、柔軟なプロレベルのインターフェイスを作成できるカスタムコントロールがサポートされています。

編集可能な選択リスト

編集可能な選択リスト（コンボボックスとも呼ばれます）を使用すると、選択リストとテキストボックスの機能を結合できます。

拡張機能ユーザインターフェイスでは、select タグで定義されるポップアップメニューが頻繁に使用されます。editable="true" を select タグに追加すると、拡張機能でポップアップメニューを編集できるようになります。デフォルト値を設定するには、editText 属性と、選択リストで表示する値を指定します。

次の例では、編集可能な選択リストの設定を示します。

```
<select name="travelOptions" style="width:250px" editable="true" editText="other
    (please specify)">
<option value="plane">plane</option>
<option value="car">car</option>
<option value="bus">bus</option>
</select>
```

拡張機能で選択リストを使用するときは、編集可能な属性の有無とその属性の値を確認します。値が存在しない場合は、選択リストが編集不可であることを示すデフォルト値の false が選択リストから返されます。

編集可能な選択リストには、標準の編集不可能な選択リストと同じく、selectedIndex プロパティ（95 ページの「Dreamweaver DOM のオブジェクト、プロパティ、およびメソッド」を参照してください）が設定されます。テキストボックスが選択されると、このプロパティから -1 が返されます。

アクティブな編集可能テキストボックスの値を拡張機能に読み込むには、editText プロパティの値を読み取ります。editText プロパティからは、ユーザが編集可能なテキストボックスに入力したストリング、または editText 属性の値が返されます。テキストが入力されていない場合や、editText に値が指定されていない場合は、空のストリングが返されます。

Dreamweaver では、編集可能なポップアップメニューを制御するために、select タグに以下のカスタム属性が追加されています。

属性名	説明	指定可能な値
editable	ポップアップメニューに編集可能なテキスト領域があることを宣言します。	ブール値（true または false）
editText	編集可能なテキスト領域のテキストを保持または設定します。	任意の値のストリング

注意：Dreamweaver では編集可能な選択リストを使用できます。

次の例では、一般的な JavaScript 関数を使用して、編集可能な選択リストを含むコマンド拡張機能を作成します。

作成例

1 テキストエディタで新しい空のファイルを作成します。

2 次のコードを入力します。

```
<html>
<head>
    <title>Editable Dropdown Test</title>
    <script language="javascript">
    function getAlert()
    {

        var i=document.myForm.mySelect.selectedIndex;
        if (i>=0)
        {
            alert("Selected index: " + i + "\n" + "Selected text " +
                    document.myForm.mySelect.options[i].text);
        }
        else
        {
            alert("Nothing is selected" + "\n" + "or you entered a value");
        }
    }
    function commandButtons()
    {
        return new Array("OK", "getAlert()", "Cancel", "window.close()");
    }
    </script>
</head>

<body>
<div name="test">
<form name="myForm">
<table>
    <tr>
        <td colspan="2">
        <h4>Select your favorite</h4>
        </td>
    </tr>
    <tr>
        <td>Sport:</td>
        <td>
        <select name="mySelect" editable="true" style="width:150px"
            editText="Editable Text">
            <option> Baseball</option>
            <option> Football </option>
            <option> Soccer </option>
        </select>
        </td>
    </tr>
</table>
</form>
</div>
</body>
</html>
```

3 ファイルに EditableSelectTest.htm という名前を付けて Dreamweaver の Configuration¥Commands フォルダに保存します。

作成例をテストする

1 Dreamweaver を再起動します。

2 コマンド／ EditableSelectTest を選択します。

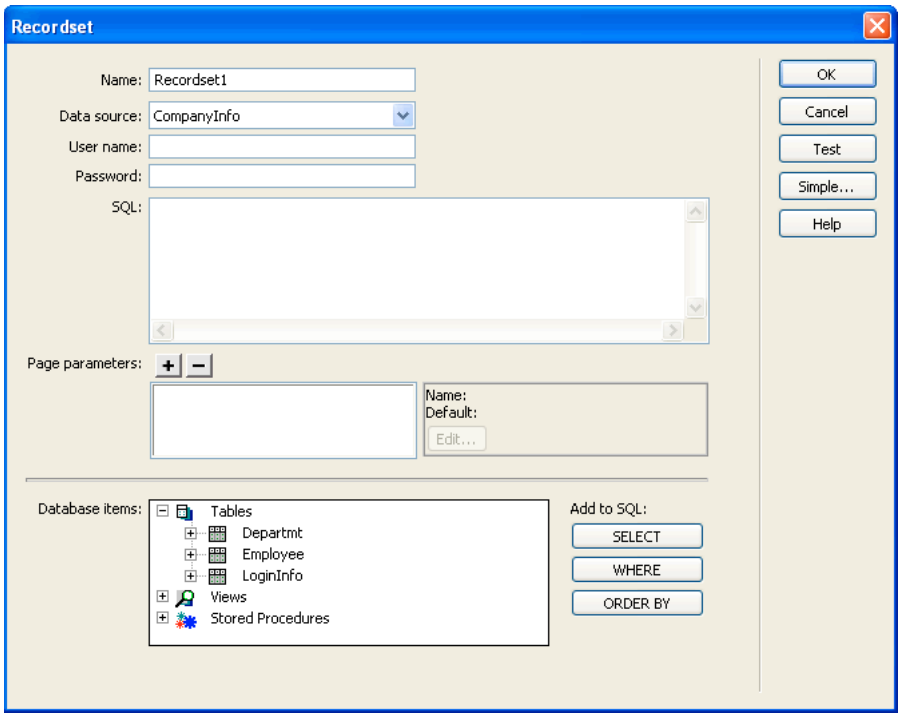
リストから値を選択すると、値のインデックスとテキストを示す警告メッセージが表示されます。値を入力すると、何も選択されていないことを示す警告メッセージが表示されます。

データベースコントロール

データベースコントロールは、データ階層とフィールドを簡単に表示できるようにします。

Dreamweaver では、HTML の select タグを拡張して、データベースツリーコントロールを作成できます。また、変数グリッドコントロールを追加することもできます。データベースツリーコントロールではデータベーススキーマが表示され、変数グリッドコントロールでは情報が表形式で表示されます。

次の図は、データベースツリーコントロールと変数グリッドコントロールを使用する高度なレコードセットダイアログボックスを示しています。



データベースツリーコントロールの追加

データベースツリーコントロールには、以下の属性があります。

属性名	説明
name	データベースツリーコントロールの名前。
control.style	ピクセル単位の幅と高さ。
type	コントロールのタイプ。
connection	接続ダイアログボックスで定義されたデータベース接続の名前。空白の場合、コントロールは表示されません。
noexpandbuttons	この属性を指定すると、ツリーコントロールに展開用のプラス（+）記号と折りたたみ用のマイナス（-）記号、または Macintosh でこの記号に相当する三角形の矢印が描画されません。この属性は、マルチコラムリストコントロールを描画する場合に便利です。
showheaders	この属性を指定すると、各列の名前を列挙するヘッダーがツリーコントロールの最上部に表示されます。

select タグの内部に挿入されたオプションのタグはすべて無視されます。

データベースツリーコントロールをダイアログボックスに追加するには、以下のサンプルコードを使用できます。引用符で囲まれた変数は適切な値に置き換えます。

```
<select name="DBTree" style="width:400px;height:110px" ¬
type="mmdatabasetree" connection="connectionName" noexpandbuttons showHeaders></select>
```

connection 属性を変更し、選択したデータを取得してツリーに表示することができます。DBTreeControl 属性は、新しいタグの JavaScript ラッパーオブジェクトとして使用できます。その他の例については、Configuration¥Shared¥Common¥Scripts フォルダにある DBTreeControlClass.js ファイルを参照してください。

変数グリッドコントロールの追加

変数グリッドコントロールには、以下の属性があります。

属性名	説明
name	変数グリッドコントロールの名前。
style	ピクセル単位のコントロールの幅。
type	コントロールのタイプ。
columns	各列には名前が必要で、それぞれをカンマで区切ります。
columnWidth	各列の幅。それぞれをカンマで区切ります。幅を指定しないと、各列の幅が均等になります。

次の例では、ダイアログボックスに単純な変数グリッドコントロールを追加します。

```
<select name="ParamList" style="width:515px;" ¬
type="mmparameterlist columns"="Name,SQL Data ¬
Type,Direction,Default Value,Run-time Value" size=6></select>
```

次の例では、幅の異なる 5 つの列から成る幅 500 ピクセルの変数グリッドコントロールを作成します。

```
<select
    name="ParamList"
    style="width:500px;"
    type="mmparameterlist"
    columns="Name,SQL Data Type,Direction, Default Value,Run-time Value"
    columnWidth="100,25,11,"
    size=6>
```

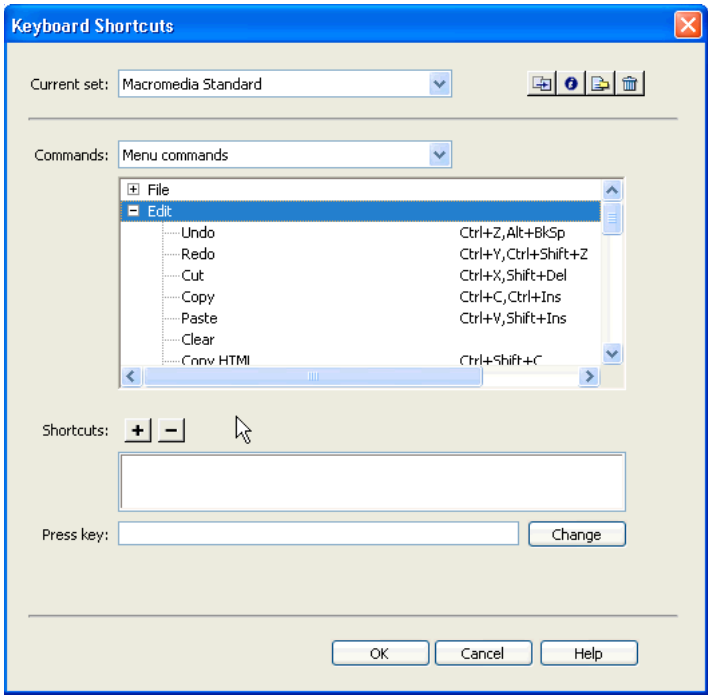
この例では、幅 182 ピクセルの空白の列が 2 列作成されます。つまり、指定した列の幅の合計が 136 ピクセル、変数グリッドコントロール全体の幅が 500 ピクセルなので、残りのスペースが 364 ピクセルとなり、これを 2 で割った 182 ピクセルが残りの 2 つの列の幅となります。

この変数グリッドコントロールには、変数グリッドコントロールのデータにアクセスして操作する JavaScript ラッパーオブジェクトもあります。その実装は、Configuration¥Shared¥MM¥Scripts¥Class フォルダ内の GridControlClass.js ファイルにあります。

ツリーコントロール

ツリーコントロールでは、展開と折りたたみが可能なノードで情報を整理します。

ツリーコントロールでは、データが階層形式で表示され、ユーザはツリーのノードを展開したり折りたたんだりできます。MM:TREECONTROL タグを使用すると、任意の種類の情報に対してツリーコントロールを作成できます。85 ページの「データベースツリーコントロールの追加」で説明したデータベースツリーコントロールとは異なり、データベースとの関連付けは不要です。Dreamweaver のキーボードショートカットエディタでは、次の図のようにツリーコントロールが使用されます。



ツリーコントロールの作成

MM:TREECONTROL タグでは、ツリーコントロールを作成するほかに、次のように 1 つ以上のタグを使用して構造を追加することもできます。

- MM:TREECOLUMN は、ツリーコントロールの列を定義するオプションの空のタグです。
- MM:TREENODE は、ツリー内のノードを定義するオプションのタグです。空のタグではなく、他の MM:TREENODE タグのみを含めることができます。

MM:TREECONTROL タグには、以下の属性があります。

属性名	説明
name	ツリーコントロールの名前。
size	オプション。コントロールに表示する行数。デフォルトは 5 行です。
theControl	オプション。theControl 属性のノード数が size 属性の値を超えると、スクロールバーが表示されます。
multiple	オプション。複数選択が可能になります。デフォルトは単一選択です。
style	オプション。ツリーコントロールの高さと幅のスタイル定義。指定すると、size 属性より優先されます。
noheaders	オプション。列ヘッダーを非表示に指定します。

MM:TREECOLUMN タグには、以下の属性があります。

属性名	説明
name	列の名前。
value	列のヘッダーに表示されるストリング。
width	ピクセル単位の列幅（比率はサポートされていません）。デフォルトは 100 です。
align	オプション。列内でテキストを左揃え、右揃え、中央揃えのどの方法で配置するかを指定します。デフォルトは左揃えです。
state	列を表示するかどうかを指定します。

読みやすくするために、TREECOLUMN タグは次の例のように MM:TREECONTROL タグの直後に配置してください。

```
<MM:TREECONTROL name="tree1">
<MM:TREECOLUMN name="Column1" width="100" state="visible">
<MM:TREECOLUMN name="Column2" width="80" state="visible">
...
</MM:TREECONTROL>
```

次の表では、MM:TREENODE の属性について説明します。

属性名	説明
name	ノードの名前。
value	指定したノードのコンテンツを格納します。複数の列を指定する場合は、棒線で区切って入力します。空の列を指定するには、棒線（|）の前にスペースを 1 つ挿入します。
state	ストリング "expanded" でノードの展開を指定し、ストリング "collapsed" でノードの折りたたみを指定します。
selected	ツリーに MULTIPLE 属性が指定されている場合は、この属性を複数のツリーノードで設定することによって複数のノードを選択できます。
icon	オプション。使用する組み込みアイコンのインデックス。0 から始まり、次のように設定します。 0 = アイコンなし、1 = Dreamweaver ドキュメントアイコン、2 = マルチドキュメントアイコン。

例えば、次のツリーコントロールでは、すべてのノードが展開されます。

```
<mm:treecontrol name="test" style="height:300px;width:300px">

<mm:treenode value="rootnode1" state="expanded">
<mm:treenode value="node1" state="expanded"></mm:treenode>
<mm:treenode value="node3" state="expanded"></mm:treenode>
</mm:treenode>

<mm:treenode value="rootnode2"0state="expanded">
<mm:treenode value="node2" state="expanded"></mm:treenode>
<mm:treenode value="node4" state="expanded"></mm:treenode>
</mm:treenode>

</mm:treecontrol>
```

ツリーコントロールの内容の操作

ツリーコントロールとその内部のノードは、HTML タグとして実装されます。実装されたツリーコントロールとノードは、Dreamweaver によって解析され、ドキュメントツリーに保存されます。この HTML タグは、その他のドキュメントノードと同じ方法で操作できます。DOM 関数について詳しくは、94 ページの「Dreamweaver ドキュメントオブジェクトモデル」を参照してください。

ノードを追加する 既存のツリーコントロールにプログラムでノードを追加するには、innerHTML プロパティ（MM:TREECONTROL タグまたは既存の MM:TREENODE タグ）を設定します。ツリーノードの innerHTML プロパティを設定すると、ネストされたノードが作成されます。

次の例では、ツリーの最上位レベルにノードを追加します。

```
var tree = document.myTreeControl;
//add a top-level node to the bottom of the tree
tree.innerHTML = tree.innerHTML + '<mm:treenode name="node3" value="node3">';
```

子ノードを追加する 現在選択されているノードに子ノードを追加するには、選択しているノードの innerHTML プロパティを設定します。

次の例では、現在選択されているノードに子ノードを追加します。

```
var tree = document.myTreeControl;
var selNode = tree.selectedNodes[0];
//deselect the node, so we can select the new one
selnode.removeAttribute("selected");
//add the new node to the top of the selected node's children
selNode.innerHTML = '<mm:treenode name="item10" value="New item11" expanded selected>' + ¬
selNode.innerHTML;
```

ノードを削除する 現在選択されているノードをドキュメント構造から削除するには、innerHTML プロパティまたは outerHTML プロパティを使用します。

次の例では、選択されたノードと子ノードをすべて削除します。

```
var tree = document.myTreeControl;
var selNode = tree.selectedNodes[0];
selNode.outerHTML = "";
```

カラーボタンコントロールの追加

カラーボタンコントロールを使用すると、拡張機能にカラーピッカーインターフェイスを追加できます。

Dreamweaver では、テキスト、チェックボックス、ボタンなどの標準的な入力タイプに加えて、拡張機能の追加入力タイプである mmcolorbutton もサポートされています。

コードで <input type="mmcolorbutton"> を指定すると、ユーザインターフェイスにカラーピッカーが表示されます。input タグで value 属性を設定することで、カラーピッカーのデフォルトカラーを設定できます。値を設定しないと、カラーピッカーはデフォルトのグレーで表示され、入力オブジェクトの value プロパティから空白のストリングが返されます。

次の例は、有効な mmcolorbutton タグを示しています。

```
<input type="mmcolorbutton" name="colorbutton" value="#FF0000">
<input type="mmcolorbutton" name="colorbutton" value="teal">
```

カラーボタンには、カラーの変更時に実行される onChange というイベントがあります。

テキストボックスとカラーピッカーは常に同期しておきます。次の例では、テキストボックスのカラーとカラーピッカーのカラーを同期させるテキストボックスを作成します。

```
<input type = "mmcolorbutton" name="fgcolorPicker" onChange="document.fgcolorText.value=this.value">
<input type = "test" name="fgcolorText" onBlur="document.fgColorPicker.value=this.value">
```

この例では、ユーザがテキストボックスの値を変更した後に、Tab キーを押すか別の場所をクリックします。これによりカラーピッカーが更新されて、テキストボックスで指定されたカラーが表示されます。ユーザがカラーピッカーで新しいカラーを選択すると、テキストボックスが更新されて、そのカラーに対応した 16 進数の値が表示されます。

Dreamweaver への Flash コンテンツの追加

Flash コンテンツ（SWF ファイル）は、オブジェクトの一部として、またはコマンドとして Dreamweaver インターフェイスに表示できます。この Flash のサポートは、Flash フォーム、アニメーション、ActionScript などの Flash コンテンツを使用する拡張機能を構築する際に特に便利です。

基本的に、Dreamweaver のオブジェクトおよびコマンドのダイアログ表示機能を利用するには（オブジェクトの構築については 103 ページの「挿入バーオブジェクト」を、コマンドについては 130 ページの「コマンド」を参照）、form タグを object タグと共に使用して Dreamweaver のダイアログボックスに Flash コンテンツを埋め込みます。

簡単な Flash ダイアログボックスの例

この例では、Dreamweaver を使用してコマンドを作成します。ユーザがコマンドメニューでそのコマンドをクリックすると、myFlash.swf という名前の SWF ファイルが表示されます。この例を試す前にコマンドの作成に関する詳細を確認するには、130 ページの「コマンド」を参照してください。

注意：この例では、Dreamweaver アプリケーションインストールフォルダの Configuration¥Commands フォルダに myFlash.swf という SWF ファイルが存在することが前提となります。自分の SWF ファイルでこの例をテストするには、アプリケーションの Commands フォルダにその SWF ファイルを保存し、myFlash.swf の全インスタンスをそのファイル名で置き換えます。

Dreamweaver で、基本の HTML ファイルを新規に開きます。これが、コマンド定義ファイルになります。次のように、ページの冒頭で title の開始タグと終了タグの間に「**My Flash Movie**」と入力します。

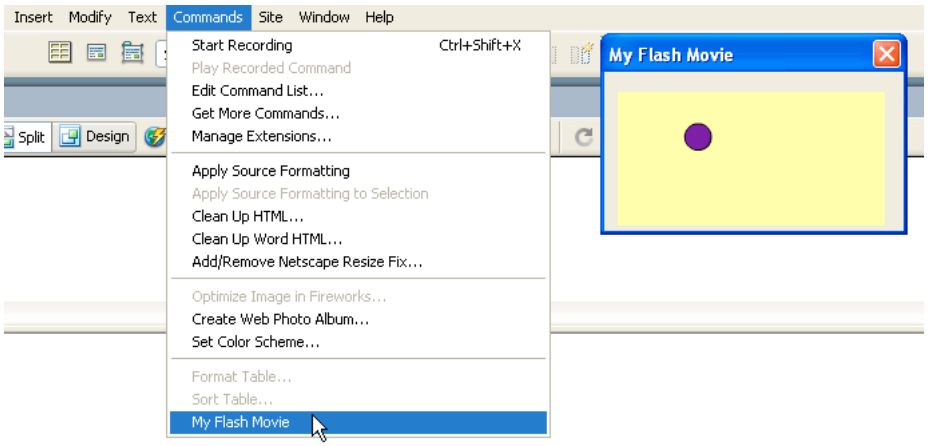
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>My Flash Movie</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>
```

次に、このファイルをアプリケーションの Configuration¥Commands フォルダに My Flash Movie.htm として保存します。ファイルはまだ閉じないでください。ここでファイルを保存すると、SWF ファイルに相対パスを埋め込むことができます。そうしなければ、絶対パスが使用されます。

HTML ドキュメントに戻り、body の開始タグと終了タグの間に form の開始タグと終了タグを追加します。次に、form タグ内で、挿入／メディア／ Flash オプションを使用して、コマンド定義ファイルに SWF ファイルを追加します。プロンプトが表示されたら、Commands フォルダで SWF ファイルを選択し、「OK」をクリックします。コマンド定義ファイルは、次の例のようになります。ただし、width 属性と height 属性は、SWF ファイルのプロパティによって異なります。

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
    <title>My Flash Movie</title>
    <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>
<body>
<body>
<form>
    <object id="FlashID" classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000" width="700" height="150">
    <param name="movie" value="myFlash.swf">
    <!--[if !IE]>-->
    <object type="application/x-shockwave-flash" data="myFlash.swf" width="700"  height="150">
    <!--<![endif]-->
    <param name="quality" value="high"/>
    <param name="wmode" value="opaque" />
    <param name="swfversion" value="8.0.35.0" />
    <!-- This param tag prompts users with Flash Player 6.0 r65 and higher to download the latest version
of Flash Player. Delete it if you don't want users to see the prompt. -->
    <param name="expressinstall" value="../../../../../ColdFusion8/wwwroot/lr/Scripts/expressInstall.swf"
/>
    <!-- The browser displays the following alternative content for users with Flash Player 6.0 and older.
-->
    <div>
    <h4>Content on this page requires a newer version of Adobe Flash Player.</h4>
    <p><a href="http://www.adobe.com/go/getflashplayer"><img
src="http://www.adobe.com/images/shared/download_buttons/get_flash_player.gif" alt="Get Adobe Flash
player" width="112" height="33" /></a></p>
    </div>
    <!--[if !IE]>-->
</object>
<!--<![endif]-->
</object>
</form>
</body>
</html>
```

ファイルを再び保存します。次に、Dreamweaver を終了して再起動します。コマンド／My Flash Movie オプションを選択すると、次の図のように Dreamweaver のダイアログボックスに SWF ファイルが表示されます。



この例は、Dreamweaver の SWF ファイルサポートの単純な実装方法を示しています。オブジェクトとコマンドの構築やより複雑なフォームの構築を習得すると、SWF ファイルを Dreamweaver 拡張機能に統合することができます。これによって、より動的なユーザ操作を実現できるようになります。詳しくは、130 ページの「コマンド」で commandButtons() 関数の記述に関する項目を参照してください。この関数を使用すると、SWF ファイルを表示するダイアログボックスにボタンを追加できます。

Photoshop の統合とスマートオブジェクト

Adobe® Dreamweaver CS4® では、Photoshop ファイルをスマートオブジェクトとして読み込み、処理します。Photoshop を使用して元のイメージに加えた変更は、すぐに Dreamweaver に反映されます。Photoshop を Dreamweaver に統合する API について詳しくは、『Dreamweaver API リファレンス』の「Photoshop との統合」を参照してください。

スマートオブジェクトの例

この例では、Dreamweaver を使用して、ユーザがコマンドメニューのコマンドをクリックしたときに Photoshop ファイル（PSD）を更新するコマンドを作成します。このコマンドを機能させるには、HTML ページ上にスマートオブジェクト Web イメージが配置されていることが必要です。この例を試す前にコマンドの作成に関する詳細を確認するには、130 ページの「コマンド」を参照してください。

Dreamweaver で、基本の HTML ファイルを新規に開きます。これが、コマンド定義ファイルになります。コマンド定義ファイルは、次の例に示すような内容になります。

```
<html xmlns:MMStr ing="http://www.adobe.com /schemes/data/   string/">
<head>
    <title> Smart Objects API</ title >
    <SC RIPT SRC="../Shared/Common/Scripts /dwscripts.js"></SC RIPT>
    <SCRIPT LANGUAGE="Javascript">
function invokeSmartObjectJavaScriptAPICall() {
    var selection = dw.getSelection();
    if (!selection) {
    alert("Err: No selection!");
    return;
    }
    var node = dw.offsetsToNode(selection[0], selection[1]);
    if (!node) {
                alert("Err: No Node!");
                return;
                }
    var imageSrc = node.getAttribute("src");
    if (!imageSrc) {
                    alert("Err: No src attribute!");
                    return;
                    }
    var fullPath = getFullPath(imageSrc);
    if (!fullPath) {
                    alert("Err: No path!");
                    return;
                    }
    //alert(fullPath);
    alert("updateSmartObjectFromOriginal");
    dw.updateSmartObjectFromOriginal(fullPath);
}
    <script>
</head>
<body onload="invokeSmartObjectJavaScriptAPICall()">
</body>
</html>
```

次に、ファイルに smartobjects.htm という名前を付けて、アプリケーションフォルダの Configuration¥Commands フォルダに保存します。プロンプトが表示されたら、Commands フォルダで SWF ファイルを選択し、「OK」をクリックします。

第 6 章：Dreamweaver ドキュメントオブジェクトモデル

Adobe Dreamweaver のドキュメントオブジェクトモデル（DOM）は、拡張機能の開発のために非常に重要な構造です。

DOM は、マークアップ言語を使用して作成されたドキュメントの構成を定義します。DOM では、タグと属性をオブジェクトとして表すことによって、プログラミング言語を使用して、ドキュメントとそのコンポーネントにアクセスしたり、操作することができます。

Dreamweaver DOM について

HTML ドキュメントの構造は、ドキュメントツリーとして扱うことができます。ルートは html タグであり、最も大きい 2 つの幹は head タグと body タグです。head の支脈は title、style、script、isindex、base、meta および link です。body タグの支脈には、次のものがあります。

- 見出し（h1、h2 など）
- ブロックレベル要素（p、div、form、その他）
- インライン要素（br、img、その他）
- その他の要素タイプ

これらの支脈の葉（リーフ）には width、height、alt などの属性があります。

DOM では、ツリー構造が親ノードと子ノードから成る階層として保存および表示されます。ルートノードには親ノードがなく、リーフノードには子ノードがありません。HTML 構造の各レベルで、HTML エレメントは、JavaScript に至るまでをノードとして操作できます。この構造により、ドキュメントやドキュメント内のすべてのエレメントにアクセスできます。

JavaScript では、ドキュメント内のオブジェクトを名前またはインデックスで指定できます。例えば、名前または ID を持つ「myButton」という送信ボタンが、ドキュメントの最初のフォームで 2 番目のエレメントだとします。このとき、次に示すこのボタンへの両方の参照が有効です。

- 名前の場合は、document.myForm.myButton のように指定します。
- インデックスの場合は、document.forms[0].elements[1] のように指定します。

オプションのグループのように同じ名前を持つオブジェクトは、1 つの配列にまとめられます。0 から始まるインデックスを 1 ずつ増分することで、配列内の特定のオブジェクトにアクセスできます。例えば、「myForm」という名前のフォームの中にある「myRadioGroup」という名前の 1 番目のオプションは、document.myForm.myRadioGroup[0] で参照できます。

ユーザドキュメントの DOM と拡張機能の DOM の区別

ユーザのドキュメントの DOM と、拡張機能の DOM を区別することが重要です。このトピックの内容は、両方の Dreamweaver ドキュメントに適用されますが、各 DOM の参照方法は異なります。

ブラウザでの JavaScript の扱いに習熟している方は、document と記述することにより、アクティブなドキュメント内のオブジェクトを参照できます （例：document.forms[0]）。Dreamweaver では、document は拡張機能ファイルを参照し、document.forms[0] は拡張機能ユーザインターフェイスの最初のフォームを参照します。ユーザのドキュメント内のオブジェクトを参照するには、dw.getDocumentDOM()、dw.createDocument()、またはユーザドキュメントのオブジェクトを返す別の関数を呼び出す必要があります。

例えば、アクティブなドキュメントで最初のイメージを参照するには、dw.getDocumentDOM().images[0] と記述します。また、次の例で示すように、ドキュメントオブジェクトを変数に格納し、その後でこの変数を参照することもできます。

```
var dom = dw.getDocumentDOM();  //get the dom of the current document
var firstImg = dom.images[0];
firstImg.src = "myImages.gif";
```

このような記述法は、Configuration フォルダ内のファイル、特にコマンドファイルで一般的です。
dw.getDocumentDOM() メソッドについて詳しくは、dreamweaver.getDocumentDOM() 関数（『Dreamweaver API リファレンス』）を参照してください。

Dreamweaver DOM

Dreamweaver DOM は、World Wide Web Consortium（W3C）DOM レベル 1 の仕様のオブジェクト、プロパティ、およびメソッドのサブセットに、Microsoft Internet Explorer 4.0 DOM の一部のプロパティを追加したものです。

Dreamweaver DOM のオブジェクト、プロパティ、およびメソッド

次の表では、Dreamweaver DOM でサポートされているオブジェクト、プロパティ、メソッド、およびイベントを示します。中には、特定のオブジェクトのプロパティとしてアクセスすると読み取り専用となるプロパティもあります。黒丸（•）は、このように使用されると読み取り専用になるプロパティを示します。

オブジェクト	プロパティ	メソッド	イベント
window	navigator • document • innerWidth • innerHeight • screenX • screenY •	alert() confirm() escape() unescape() close() setTimeout() clearTimeout() setInterval() clearInterval() resizeTo()	onResize
navigator	platform •	なし	なし
document	forms •（form オブジェクトの配列） images •（image オブジェクトの配列） layers（LAYER、ILAYER および絶対位置のエレメントの配列） child オブジェクト（名前指定）• nodeType • parentNode • childNodes • previousSibling • nextSibling • documentElement • body • URL • parentWindow •	getElementsByTagName() getElementsByAttributeName() getElementById() hasChildNodes()	onLoad
すべてのタグまたはエレメント	nodeType • parentNode • childNodes • tagName • previousSibling • nextSibling • attributes（名前指定） innerHTML outerHTML	getAttribute() setAttribute() removeAttribute() getElementsByTagName() getElementsByAttributeName() hasChildNodes()	

オブジェクト	プロパティ	メソッド	イベント
form	すべてのタグに使用可能なプロパティと、以下のプロパティ tags:elements （button、checkbox、password、radio、reset、select、submit、text、file、hidden、image および textarea の各オブジェクトの配列） mmcolorbutton child オブジェクト（名前指定）•	すべてのタグに使用可能なメソッドのみ	なし
layer	すべてのタグに使用可能なプロパティと、以下のプロパティ visibility left top width height zIndex	すべてのタグに使用可能なメソッドのみ	なし
image	すべてのタグに使用可能なプロパティと、以下のプロパティ src	すべてのタグに使用可能なメソッドのみ	onMouseOver onMouseOut onMouseDown onMouseUp
button reset submit	すべてのタグに使用可能なプロパティと、以下のプロパティ form •	すべてのタグに使用可能なメソッドと、以下のメソッド blur() focus()	onClick
checkbox radio	すべてのタグに使用可能なプロパティと、以下のプロパティ checked• form •	すべてのタグに使用可能なメソッドと、以下のメソッド blur() focus()	onClick
password text file hidden image（フィールド）textarea	すべてのタグに使用可能なプロパティと、以下のプロパティ form • value	すべてのタグに使用可能なメソッドと、以下のメソッド blur() focus() select()	onBlur onFocus
select	すべてのタグに使用可能なプロパティと、以下のプロパティ form • options•（option オブジェクトの配列） selectedIndex	すべてのタグに使用可能なメソッドと、以下のメソッド blur()（Windows のみ） focus()（Windows のみ）	onBlur（Windows のみ） onChange onFocus（Windows のみ）

オブジェクト	プロパティ	メソッド	イベント
option	すべてのタグに使用可能なプロパティと、以下のプロパティ text	すべてのタグに使用可能なメソッドのみ	なし
mmcolorbutton	すべてのタグに使用可能なプロパティと、以下のプロパティ name value	なし	onChange
array boolean date function number object string regexp	Netscape Navigator 4.0 に適合	Netscape Navigator 4.0 に適合	なし
text	nodeType • parentNode • childNodes • previousSibling • nextSibling • data	hasChildNodes()	なし
comment	nodeType • parentNode • childNodes • previousSibling • nextSibling • data	hasChildNodes()	なし
NodeList	length •	item()	なし
NamedNodeMap	length •	item()	なし

document オブジェクトのプロパティとメソッド

次の表では、Dreamweaver によりサポートされる document オブジェクトのプロパティとメソッドについて詳しく説明します。黒丸（•）は、読み取り専用のプロパティを示します。

プロパティまたはメソッド	戻り値
nodeType •	Node.DOCUMENT_NODE
parentNode •	null
parentWindow •	ドキュメントの親ウィンドウに相当する JavaScript オブジェクト（このプロパティは Microsoft Internet Explorer 4.0 DOM で定義されていますが、DOM レベル 1 または 2 の一部ではありません）。
childNodes •	NodeList（document オブジェクトのすぐ下に存在するすべての子ノードを含む）。通常、ドキュメントには、単一の子オブジェクトである HTML オブジェクトが含まれます。
previousSibling •	null
nextSibling •	null
documentElement •	html タグに相当する JavaScript オブジェクト。これは document.childNodes の値の取得と、HTML タグの NodeList からの抽出を簡単に行うためのプロパティです。
body •	body タグに相当する JavaScript オブジェクト。これは document.documentElement.childNodes を簡単に呼び出し、body タグを NodeList から抽出するためのプロパティです。フレームセットドキュメントでは、このプロパティによって、一番外側のフレームセットのノードが返されます。
URL •	ドキュメントの *file:*//URL。ファイルが保存されていない場合は、空白のストリングになります。
getElementsByTagName(tagName)	NodeList（*tagName* という種類のタグ（img、div など）をたどっていくときに使用）。 *tagName* 引数が「LAYER」である場合、すべての LAYER タグ、ILAYER タグおよびすべての絶対位置のエレメントが返されます。 *tagName* 引数が「INPUT」である場合、すべてのフォームエレメントが返されます。name 属性を 1 つ以上の *tagName* オブジェクトに対して指定する場合は、HTML 4.01 の仕様に従った文字で始めてください。そうでないと、この関数によって返される配列が不正な長さになります。
getElementsById(Id)	指定された *id* でエレメントノードを取得します。この場合 *id* は取得するエレメントの ID を含むストリングです。 var dom = dw.getDocumentDOM(); var contObj = dom.getElementById('content'); alert("The element with the id 'content' is a " + contObj.tagName);
getElementsByAttributeName(attrName)	NodeList（例えば、「for」属性を持つすべてのエレメントなど、*attrName* 属性を持つエレメント全体に使用）。DOM レベル 1 または 2 の一部ではありません。
getElementById(id)	指定された ID を持つ HTML エレメント。
hasChildNodes()	true

HTML のプロパティとメソッド

次の表では、Dreamweaver の HTML エレメントのプロパティとメソッド、およびその戻り値または説明を示します。黒丸（•）は、読み取り専用のプロパティを示します。

プロパティまたはメソッド	戻り値
nodeType •	Node.ELEMENT_NODE
parentNode •	親タグ。これが HTML タグである場合、document オブジェクトが返されます。
childNodes •	タグのすぐ下に存在するすべての子タグを含む NodeList。
previousSibling •	このノードの直前の兄弟ノード。例えば、HTML ドキュメントでの previousSibling は、body エレメントの場合は head エレメントです。
nextSibling •	このノードの直後の兄弟ノード。例えば、HTML ドキュメントでの nextSibling は、head エレメントの場合は body エレメントです。head 内の script、style または meta タグは、head エレメントの子ノードになります。
tagName •	IMG、A、DIV などの、エレメントの HTML tagName。返される値は常に大文字です。
attrName	指定したタグ属性の値を含むストリング。タグ。*attrName* 属性が JavaScript 言語で予約された語（*class* など）になっている場合、attrName 属性は使用できません。この場合は、getAttribute() および setAttribute() を使用してください。
innerHTML	開始タグと終了タグの間にあるソースコード。例えば、 Hello, World! というコードでは、p.innerHTML により Hello, World! が返されます。このプロパティに書き込むと、DOM ツリーがドキュメントの新しい構造を反映して直ちに更新されます （このプロパティは Microsoft Internet Explorer 4.0 DO で定義されていますが、DOM レベル 1 または 2 の一部ではありません）。
outerHTML	このタグのソースコード。タグ自体も含みます。前述のコード例の場合、p.outerHTML は Hello, World! を返します。このプロパティに書き込むと、DOM ツリーがドキュメントの新しい構造を反映して直ちに更新されます（このプロパティは Microsoft Internet Explorer 4.0 DO で定義されていますが、DOM レベル 1 または 2 の一部ではありません）。
getAttribute(*attrName*)	属性を明示的に指定した場合はその属性値。指定しない場合は null を返します。
getTranslatedAttribute(*attrName*)	指定した属性のトランスレート値。または、属性値がトランスレートされていない場合は、getAttribute() が返す値。このプロパティは DOM レベル 1 には含まれていませんが、属性のトランスレートをサポートするように Dreamweaver 3 に追加されています。
setAttribute(*attrName*, *attrValue*)	戻り値なし。指定された値の例 img.setAttribute("src", "image/roses.gif") に対して指定された属性を設定します。
removeAttribute(*attrName*)	戻り値なし。このタグに対して指定した属性とその値を HTML から削除します。
getElementsByTagName(*tagName*)	NodeList（*tagName* という種類の子タグ（IMG、DIV など）をたどっていくときに使用）。 *tagName* 引数が「layer」である場合、すべての LAYER タグ、ILAYER タグおよびすべての絶対位置のエレメントが返されます。 *tagName* 引数が「input」である場合、すべてのフォームエレメントが返されます。name 属性を 1 つ以上の tagName オブジェクトに対して指定する場合は、HTML 4.01 の仕様に従った文字で始めてください。そうでないと、この関数によって返される配列が不正な長さになります。
getElementsByAttributeName(*attrName*)	NodeList（例えば、「for」属性を持つすべてのエレメントなど、*attrName* 属性を持つエレメント全体に使用）。DOM レベル 1 または 2 の一部ではありません。
hasChildNodes()	タグに子があるかどうかを示すブール値。
hasTranslatedAttributes()	タグにトランスレートされた属性があるかどうかを示すブール値。このプロパティは DOM レベル 1 には含まれていませんが、属性のトランスレートをサポートするように Dreamweaver 3 に追加されています。

text オブジェクトのプロパティとメソッド

HTML ドキュメント内の連続している各テキストブロック（例えば p タグ内のテキスト）は、JavaScript オブジェクトによって表されます。text オブジェクトには子オブジェクトがありません。次の表では、DOM レベル 1 に準拠し、Dreamweaver で使用される text オブジェクトのプロパティとメソッドについて説明します。黒丸（•）は、読み取り専用のプロパティを示します。

プロパティまたはメソッド	戻り値
nodeType •	Node.TEXT_NODE
parentNode •	親タグ
child Nodes •	空白
previousSibling •	このノードの直前の兄弟ノード。例えば、 blah blah というコードでは、 タグに 3 つの子ノード（テキストノード、エレメントノード、テキストノード）があります。3 番目の子ノードの previousSibling は タグ、最初の子ノードの previousSibling は null です。
nextSibling •	このノードの直後の兄弟ノード。例えば、 blah blah というコードでの nextSibling は、p タグの最初の子ノード場合は タグ、3 番目の子ノードの nextSibling は null です。
data	実際のテキストストリング。テキスト内のエンティティは、単一文字で表されます（例えば、Joseph & I というテキストは Joseph & I として返されます）。
hasChildNodes()	false

comment オブジェクトのプロパティとメソッド

JavaScript オブジェクトは各 HTML コメントを表します。次の表では、DOM レベル 1 に準拠し、Dreamweaver で使用される comment オブジェクトのプロパティとメソッドについて詳しく説明します。黒丸（•）は、読み取り専用のプロパティを示します。

プロパティまたはメソッド	戻り値
nodeType •	Node.COMMENT_NODE
parentNode •	親タグ
childNodes •	空白の NodeList 配列
previousSibling •	このノードの直前の兄弟ノード。
nextSibling •	このノードの直後の兄弟ノード。
data	コメントマーカーの間にあるテキストストリング（<!-- と -->）
hasChildNodes()	false

dreamweaver オブジェクトと site オブジェクト

Dreamweaver には、DOM によってアクセス可能な標準オブジェクトに加え、2 つのカスタムオブジェクト、dreamweaver および site が実装されています。これらのカスタムオブジェクトは、API や拡張機能のコードで頻繁に使用されます。dreamweaver オブジェクトおよび site オブジェクトのメソッドについて詳しくは、『Dreamweaver API リファレンス』を参照してください。

dreamweaver オブジェクトのプロパティ

dreamweaver オブジェクトには、次に示す 2 つの読み取り専用のプロパティがあります。

- appName プロパティには、"Dreamweaver" 値があります。
- appVersion プロパティには、「***versionNumber.releaseNumber.buildNumber***[***languageCode***] (***platform***)」フォームの値があります。

  例えば、appVersion プロパティの値は、スウェーデン語版の Dreamweaver Windows バージョンでは「8.0.XXXX [se] (Win32)」となり、英語版の Macintosh バージョンでは「8.0.XXXX [en] (MacPPC)」となります。

  注意：ヘルプ／バージョン情報を選択すると、バージョン番号とビルド番号を確認できます。

appName プロパティと appVersion プロパティは、Dreamweaver 3 から実装されたもので、それ以前のバージョンの Dreamweaver では使用できません。

Dreamweaver の実際のバージョンを調べるには、まず appVersion の存在を確認してから、次の例に示すように、バージョン番号を調べます。

```
if (dreamweaver.appVersion && dreamweaver.appVersion.indexOf('3.01') != -1){¬
    // execute code
}
```

dreamweaver オブジェクトには、ユーザのオペレーティングシステムの言語を確認できる systemScript というプロパティがあります。拡張機能のコードに各国語版のオペレーティングシステムに対応した特別な処理を記述する必要がある場合は、次の例で示すように、このプロパティを使用します。

```
if (dreamweaver,systemScript && (dreamweaver.systemScript.indexOf('ja')!=-1){¬
    SpecialCase }
```

systemScript プロパティにより、各国語版のオペレーティングシステムに応じて、以下の値が返されます。

言語	値
日本語	ja
韓国語	ko
繁体字中国語	zh_tw
簡体字中国語	zh_cn

すべてのヨーロッパ言語版のオペレーティングシステムでは、'en' が返されます。

site オブジェクト

site オブジェクトにはプロパティはありません。site オブジェクトのメソッドについて詳しくは、『Dreamweaver API リファレンス』を参照してください。

第 7 章：挿入バーオブジェクト

挿入バーに項目を追加することで、ユーザの反復タスクの自動化、およびユーザが特定の属性の設定に使用するダイアログボックスの作成ができます。

オブジェクトは、Dreamweaver アプリケーションフォルダ内の Configuration¥Objects フォルダに保存されています。Objects サブフォルダは、挿入バー上のオブジェクトの位置に応じてグループ化され、これらのファイルを開いて現在のオブジェクトの構造を確認できます。例えば、Configuration¥Objects¥Common¥Hyperlink.htm ファイルを開くと、挿入バーのハイパーテキストリンクオブジェクトのボタンに対応するコードを確認できます。

次の表にオブジェクトを作成するために使用するファイルの一覧を示します。

パス	ファイル	説明
Configuration¥Objects¥**objecttype**¥	**objectname**.htm	ドキュメントに挿入するものを指定します。
Configuration¥Objects¥**objecttype**¥	**objectname**.js	実行する関数が含まれています。
Configuration¥Objects**¥objecttype**¥	**objectname**.gif	挿入バーに表示されるイメージが格納されています。
Configuration¥Objects	insertbar.xml	挿入バーに表示されるオブジェクトとそれらの順番を指定します。

オブジェクトファイルの動作

オブジェクトは、特定のコードストリングをユーザドキュメントに挿入します。オブジェクトを使用すると、ユーザはメニュー上のアイコンやオプションをクリックすることで、イメージ、絶対位置（AP）のエレメント、テーブルなどのコンテンツを追加できます。

オブジェクトには、次のコンポーネントがあります。

- ドキュメントに挿入する対象を定義する HTML ファイル。

  オブジェクトファイルの head セクションは、body セクションのフォーム入力を処理し、ユーザのドキュメントに追加する内容をコントロールする JavaScript 関数を含むか、または外部の JavaScript ファイルを参照します。オブジェクトファイルの body には、オブジェクトのパラメータ（テーブルに挿入する行と列の数など）を受け取り、ユーザが属性を入力するためのダイアログボックスをアクティブにする HTML フォームが含まれる場合があります。

  注意：ただし、最も単純なオブジェクトには、挿入される HTML のみが含まれ、body タグおよび head タグは含まれません。詳しくは、Adobe サポートセンターの「Dreamweaver のカスタマイズ」を参照してください。

- 挿入バーに表示される 18 x 18 ピクセルのイメージ。
- insertbar.xml ファイルへの追加部分。insertbar.xml ファイルは、挿入バー上でのオブジェクトの表示位置を定義します。

挿入バーのアイコンをクリックするか、挿入メニューの項目を選択することで、オブジェクトを選択できます。オブジェクトを選択すると、次のイベントが発生します。

1 Adobe Dreamweaver によって、ダイアログボックスを表示するかどうかを決定する canInsertObject() 関数が呼び出されます。

オブジェクトファイルにある form タグが検索されます。フォームが存在しており、「一般」環境設定ダイアログボックスの「オブジェクト挿入中にダイアログを表示」オプションが選択されている場合、windowDimensions() 関数が定義されていれば、この関数が呼び出されます。Dreamwaver では、この関数を呼び出すことで、フォームを表示するダイアログボックスのサイズを判別します。オブジェクトファイルにフォームが存在しない場合、ダイアログボックスは表示されず、手順 2 がスキップされます。

2 手順 1 でダイアログボックスが表示された場合、ユーザはオブジェクト用のパラメータ（テーブルの行数と列数など）をそのダイアログボックスのテキストフィールドに入力し、「OK」をクリックします。

3 Dreamweaver によって objectTag() 関数が呼び出され、その戻り値がドキュメントの現在の選択範囲の後に挿入されます。現在の選択範囲は置き換えません。

4 objectTag() 関数が見つからない場合は、代わりに insertObject() 関数が検索され、呼び出されます。

挿入バー定義ファイル

Configuration¥Objects¥insertbar.xml ファイルでは、挿入バーのプロパティを定義します。この XML ファイルには、各オブジェクトの定義がオブジェクトが表示される順序で格納されています。

Dreamweaver を初めて起動すると、挿入バーはドキュメントの上に水平に表示されます。その後、その表示状態と位置がレジストリに保存されます。

Insertbar.xml のタグの階層

次の例では、insertbar.xml ファイルのネストされたタグの形式と階層を示します。

```
<?xml version="1.0" ?>
<!DOCTYPE insertbarset SYSTEM "-//Adobe//DWExtension insertbar 10.0">
<insertbar xmlns:MMString="http://www.adobe.com/schemes/data/string/">
<category id="DW_Insertbar_Common" MMString:name="insertbar/categorycommon" folder="Common">
        <button id="DW_Hyperlink" image="Common\Hyperlink.png"
        MMString:name="insertbar/hyperlink" file="Common\Hyperlink.htm" />
        <button id="DW_Email" image="Common\E-Mail Link.png"
        MMString:name="insertbar/email" file="Common\E-Mail Link.htm" />
        <separator />
    <menubutton id="DW_Images" MMString:name="insertbar/images"
    image="Common\Image.png">
        <button id="DW_Image" image="Common\Image.png"
        MMString:name="insertbar/image" file="Common\Image.htm" />
    ...
        </menubutton>
        <separator />
            <button id="DW_TagChooser" MMString:name="insertbar/tagChooser"
            image="Common\Tag Chooser.gif" command="dw.showTagChooser()"
            codeOnly="TRUE"/>
    </category>
        ...
    </insertbar>
```

注意：insertbar タグと category タグでは、終了タグである </insertbar> および </category> を使用して、タグのコンテンツの終了を示します。button タグ、checkbutton タグ、separator タグでは、終了カッコの前にスラッシュ（/）を使用して、属性とコンテンツの終了を示します。

挿入バーを定義するタグ

insertbar.xml ファイルには、次のタグと属性が含まれます。

<insertbar>

説明

このタグは、挿入バーの定義ファイルのコンテンツを示します。</insertbar> 終了タグによりコンテンツの終わりが指定されます。

属性

なし。

例

```
<insertbar>
    <category id="DW_Insertbar_Common" folder="Common">
        <button id="DW_Hyperlink" image="Common\Hyperlink.gif"
        file="Common\Hyperlink.htm"/>0
...
</insertbar>
```

<category>

説明

このタグは、挿入バーのカテゴリー（「一般」、「フォーム」、「HTML」など）を定義します。</category> 終了タグによりカテゴリーコンテンツの終わりが指定されます。

注意：デフォルトでは、挿入バーは用途別のカテゴリー（「一般」、「フォーム」、「HTML」など）に編成されます。以前のバージョンの Dreamweaver では、挿入バーはタブによって同様に編成されていました。ユーザは挿入バーのオブジェクトの編成方法（カテゴリー別かタブ別か）を指定できます。ユーザがタブによる編成を選択すると、カテゴリーのタグでは各タブが定義されます。

属性

id、{folder}、{showIf}

例

```
<category id="DW_Insertbar_Common" folder="Common">
    <button id="DW_Hyperlink" image="Common\Hyperlink.gif"
    file="Common\Hyperlink.htm"/>
</category>
```

<menubutton>

説明

このタグでは、挿入バーのポップアップメニューを定義します。

属性

id、image、{showIf}、{name}、{folder}

例

```
<menubutton
    id="DW_ImageMenu"
    name="Images"
    image="Common\imagemenu.gif"
    folder="Images">
    <button id="DW_Image"
        image="Common\Image.gif"
        enabled=""
        showIf=""
        file="Common\Image.htm" />
</menubutton>
```

<button />

説明

このタグは、ユーザがクリックする挿入バー上のボタンを定義します。ユーザがこのボタンをクリックすると、command 属性または file 属性で指定したコードが実行されます。

属性

id、image、name、{canDrag}、{showIf}、{enabled}、{command}、{file}、{tag}、{codeOnly}

例

```
<button id="DW_Object"
image="Common\Object.gif"
name="Object"
enabled="true"
showIf=""
file="Common\Obect.htm"
    />
```

<checkbutton />

説明

チェックボタンは、状態をオンまたはオフに切り替えられるボタンです。オンになっている場合、チェックボタンは押された状態でハイライト表示されます。オフの場合は、平らな状態で表示されます。Dreamweaver には、マウスポインタを上に置いた状態、押した状態、押したままマウスポインタを上に置いた状態、および押したまま無効な状態があります。クリックするとチェックボタンの状態が確実に切り替わるように、コマンドで設定する必要があります。

属性

id、image、checked、{showIf}、{enabled}、{command}、{file}、{tag}、{name}、{codeOnly}

例

```
<checkbutton id="DW_StandardView"
name = "Standard View"
image="Tools\Standard View.gif"
checked="_View_Standard"
command="dw.getDocumentDOM().setShowLayoutView(false)"/>
```

<separator/>

説明
このタグは、挿入バーに縦線を表示します。

属性
{showIf}

例
```
<separator showIf="_VIEW_CODE"/>
```

挿入バーを定義するタグの属性

挿入バーを定義するタグの属性の意味は次のとおりです。

id="unique id"

説明
id 属性は、挿入バーに表示されるボタンの識別子です。id 属性は、ファイル内のエレメントについて一意である必要があります。

例
```
id="DW_Anchor"
```

image="image_path”

説明
この属性では、挿入バーに表示されるアイコンファイルへのパスを指定します。これは、Dreamweaver の Configuration フォルダへの相対パスです。アイコンは、Dreamweaver でレンダリングできる任意の形式を使用できますが、通常は GIF または JPEG ファイル形式で、サイズが 18 x 18 ピクセルです。

例
```
image="Common/table.gif"
```

canDrag="Boolean”

説明
この属性では、ユーザがアイコンをコードやワークスペースにドラッグして、ドキュメントにオブジェクトを挿入できるかどうかを指定します。指定しない場合は、デフォルト値の true が使用されます。

例
```
canDrag="false"
```

showIf="enabler"

説明

この属性では、指定した Dreamweaver イネーブラの値が true の場合にのみ、挿入バーにこのボタンを表示するように指定します。showIf を指定しない場合、ボタンは常に表示されます。指定できるイネーブラは _SERVERMODEL_ASP、_SERVERMODEL_ASPNET、_SERVERMODEL_JSP、_SERVERMODEL_CFML（すべてのバージョンの Adobe ColdFusion の場合）、_SERVERMODEL_CFML_UD4（UltraDeveloper バージョン 4 の ColdFusion の場合のみ）、_SERVERMODEL_PHP、_FILE_TEMPLATE、_VIEW_CODE、_VIEW_DESIGN、_VIEW_LAYOUT、_VIEW_EXPANDED_TABLES および _VIEW_STANDARD です。

複数のイネーブラを指定するには、AND を意味するカンマをイネーブラ間に挿入します。NOT を指定するには、感嘆符（!）を使用します。

例

ボタンを ASP ページのコードビューでのみ表示する場合は、次のようにイネーブラを指定します。

```
showIf="_VIEW_CODE, _SERVERMODEL_ASP"
```

enabled="enabler"

説明

この属性では、**DW_enabler** の値が true の場合にユーザがこの項目を使用できることを指定します。enabled 関数を指定しないと、項目はデフォルトによって常に有効になります。指定できるイネーブラは、SERVERMODEL_ASP、_SERVERMODEL_ASPNET、_SERVERMODEL_JSP、_SERVERMODEL_CFML（ColdFusion のすべてのバージョンの場合）、_SERVERMODEL_CFML_UD4（ColdFusion の UltraDeveloper バージョン 4 の場合のみ）、_SERVERMODEL_PHP、_FILE_TEMPLATE、_VIEW_CODE、_VIEW_DESIGN、_VIEW_LAYOUT、_VIEW_EXPANDED_TABLES、および _VIEW_STANDARD です。

複数のイネーブラを指定するには、AND を意味するカンマをイネーブラ間に挿入します。NOT を指定するには、感嘆符（!）を使用します。

例

ボタンをコードビューでのみ使用可能にするには、次のように指定します。

```
enabled="_VIEW_CODE"
```

これにより、他のビューではボタンが淡色表示になります。

checked="enabler"

説明

checked 属性は、checkbutton タグを使用するときに必要です。

項目は、**DW_enabler** の値が true の場合にオンになります。指定できるイネーブラは _SERVERMODEL_ASP、_SERVERMODEL_ASPNET、_SERVERMODEL_JSP、_SERVERMODEL_CFML（ColdFusion のすべてのバージョンの場合）、_SERVERMODEL_CFML_UD4（ColdFusion の UltraDeveloper バージョン 4 の場合のみ）、_SERVERMODEL_PHP、_FILE_TEMPLATE、_VIEW_CODE、_VIEW_DESIGN、_VIEW_LAYOUT、_VIEW_EXPANDED_TABLES および _VIEW_STANDARD です。

複数のイネーブラを指定するには、AND を意味するカンマをイネーブラ間に挿入します。NOT を指定するには、感嘆符（!）を使用します。

例

```
checked="_View_Layout"
```

command="API_function"

説明

挿入するコードを含む HTML ファイルを Dreamweaver から参照する代わりに、このタグを使用することにより、このタグで設定したボタンがクリックされたときに Dreamweaver が実行するコマンドを指定します。

例

```
command="dw.showTagChooser()"
```

file="file_path"

説明

file 属性は、オブジェクトファイルのパスを指定します。このパスは、Dreamweaver の Configuration フォルダへの相対パスです。name 属性を指定しないと、オブジェクトファイルのタイトルがオブジェクトアイコンのツールヒントに使用されます。

例

```
file="Templates/Editable.htm"
```

tag="editor"

説明

この属性では、Dreamweaver でタグエディタを起動するように指定します。コードビューで、tag 属性が定義されている場合に、ユーザがオブジェクトをクリックすると、タグダイアログボックスが表示されます。コードビューで tag 属性と command 属性を指定すると、タグエディタが起動します。デザインビューで、codeOnly="TRUE" であり、file 属性を指定していない場合は、コードビューとデザインビューが起動し、コード内にフォーカスが配置されて、タグエディタが起動します。

例

```
tag = "form"
```

name="tooltip_text"

説明

name 属性では、マウスポインタをオブジェクト上に置いたときに表示されるツールヒントを指定します。オブジェクトファイルを指定して、name 属性を指定しないと、オブジェクトファイルの名前がツールヒントに使用されます。

注意： name 属性が指定されていないオブジェクトは、挿入バーの UI のお気に入りカテゴリーにグループ化することはできません。

挿入バーの一部のオブジェクトでは、接頭辞が MMString である様々な name 属性を使用します。MMString は、ローカライズされたストリングを表します。値については、「79 ページの「拡張機能のローカライズ」」で説明します。

例

```
name = "cfoutput"
```

挿入バーの修正

オブジェクトのカテゴリー間での移動、カテゴリーの名前変更、およびウィンドウからのオブジェクトの削除を行うことができます。挿入バーに変更を表示するには、Dreamweaver を再起動するか、拡張機能をリロードする必要があります。拡張機能のリロードについては、77 ページの「拡張機能のリロード」を参照してください。

挿入バーの異なるカテゴリー間で、または同一カテゴリー内の別の場所へのオブジェクトの移動またはコピー

1. insertbar.xml ファイルのバックアップコピーを保存します。バックアップには、insertbar.backup.xml などの名前を付けます。
2. 元の insertbar.xml ファイルを開きます。
3. 移動またはコピーするオブジェクトを表す button タグを見つけます。例えば、イメージオブジェクトを一般カテゴリーから別の場所に移動するには、button タグ（id 属性が "DW_Image"）を見つけます。
4. button タグ全体をカットまたはコピーします。
5. オブジェクトの移動先またはコピー先のカテゴリーを表す category タグを見つけます。
6. オブジェクトを表示するカテゴリー内の場所を指定します。
7. コピーした button タグをペーストします。
8. insertbar.xml ファイルを保存します。
9. 拡張機能をリロードします。

挿入バーからのオブジェクトの削除

1. insertbar.xml ファイルのバックアップコピーを保存します。バックアップには、insertbar.backup.xml などの名前を付けます。
2. 元の insertbar.xml ファイルを開きます。
3. 削除するオブジェクトを表す button タグを見つけます。
4. button タグ全体を削除します。
5. insertbar.xml ファイルを保存します。
6. オブジェクトの HTML、GIF および JavaScript ファイルをディスク上の現在のフォルダから移動し、insertbar.xml ファイルで一覧表示されていないフォルダに置きます。例えば、Configuration¥Objects フォルダに Unused という名前の新しいフォルダを作成し、オブジェクトのファイルをそこに移動します。オブジェクトを削除しても問題がないことが確実である場合は、ファイルを完全に削除してもかまいません。ただし、削除したオブジェクトを後で復元する必要が生じた場合に備えて、ファイルのバックアップを取っておくことをお勧めします。
7. 拡張機能をリロードします。

挿入バーのカテゴリーの順序の変更

1. insertbar.xml ファイルのバックアップコピーを保存します。バックアップには、insertbar.backup.xml などの名前を付けます。
2. 元の insertbar.xml ファイルを開きます。
3. 移動するカテゴリーに対応する category タグを見つけ、そのタグおよびそれに含まれているすべてのタグを選択します。
4. タグをカットします。
5. 新しい位置にタグをペーストします。タグを他の category タグ内にペーストしないように注意します。
6. insertbar.xml ファイルを保存します。
7. 拡張機能をリロードします。

新規カテゴリーの作成

1 insertbar.xml ファイルのバックアップコピーを保存します。バックアップには、insertbar.backup.xml などの名前を付けます。

2 元の insertbar.xml ファイルを開きます。

3 カテゴリーのデフォルトのフォルダと、そのカテゴリーに表示する一連のオブジェクトを指定して、新しい category タグを作成します。

4 insertbar.xml のタグのシンタックスについて詳しくは、105 ページの「挿入バーを定義するタグ」を参照してください。

5 insertbar.xml ファイルを保存します。

6 拡張機能をリロードします。

挿入バーへの新しいオブジェクトの追加

オブジェクトを挿入バーに追加できます。挿入バーに変更を表示するには、Dreamweaver を再起動するか、拡張機能をリロードする必要があります。拡張機能のリロードについては、77 ページの「拡張機能のリロード」を参照してください。

1 HTML を使用してユーザのドキュメントに特定のコードストリングを定義します。必要に応じて JavaScript も使用できます。

2 Dreamweaver インターフェイスでボタンを示すグラフィック（18 x 18 ピクセル）を指定するか作成します。

これより大きいイメージを作成すると、Dreamweaver では 18 x 18 ピクセルにサイズ調整されます。オブジェクトのイメージを作成しない場合、挿入バーには、デフォルトのオブジェクトアイコンと疑問符（?）が表示されます。

3 新しいファイルを Configuration¥Objects フォルダに追加します。

4 これらの新しいファイルの場所を識別し、ボタンの外観の属性を設定するには、insertbar.xml ファイルを編集します（詳しくは、104 ページの「挿入バー定義ファイル」を参照してください）。

5 Dreamweaver を再起動するか、拡張機能をリロードします。

新しいオブジェクトが挿入バーの指定した場所に表示されます。

注意：オブジェクトファイルはいくつかのフォルダに分けて保存することができますが、必ず各ファイルに固有の名前を付けてください。例えば、dom.insertObject() 関数は、サブフォルダに関係なく Objects フォルダ内でファイルを検索します（dom.insertObject() 関数について詳しくは、『Dreamweaver API リファレンス』を参照してください）。Button.htm というファイルが Forms フォルダにあり、Button.htm という別のオブジェクトファイルが MyObjects フォルダにあると、Dreamweaver はこの 2 つを区別できません。Button.htm の 2 つのインスタンスが別個に存在すると、dom.insertObject() によって「ボタン」というオブジェクトが 2 つ表示され、ユーザはこの 2 つを区別できない場合があります。

挿入メニューへのオブジェクトの追加

挿入メニュー（または別のメニュー）上のオブジェクトの位置を追加または制御するには、menus.xml ファイルを修正します。このファイルでは、Dreamweaver のメニュー構造全体が制御されます。menus.xml ファイルの修正について詳しくは、139 ページの「メニューおよびメニューコマンド」を参照してください。

拡張機能を他の Dreamweaver ユーザに配布する場合は、80 ページの「Extension Manager の操作」を参照して、拡張機能のパッケージについて確認してください。

簡単な挿入オブジェクトの例

この例では、ユーザがボタンをクリックすることによって選択したテキストに線（取り消し線）を追加できるように、オブジェクトを挿入バーに追加します。このオブジェクトは、ユーザがドキュメントに編集上のコメントを付ける場合に便利です。

この例ではテキスト処理を実行するので、挿入バーの HTML カテゴリーのテキストポップアップメニューからオブジェクトを見本として参照できます。例えば、Bold、Emphasis、Heading の各オブジェクトファイルでは、選択したテキストをタグで囲むという類似の機能を確認できます。

取り消し線挿入オブジェクトを作成するには、次の手順を実行します。

HTML ファイルの作成

オブジェクトのタイトルは、title の開始タグと終了タグの間で指定します。また、スクリプト言語に JavaScript を指定します。

1 新しい空白のファイルを作成します。

2 次のコードを追加します。

```
<html>
<head>
<title>Strikethrough</title>
<script language="javascript">
</script>
</head>
<body>
</body>
</html>
```

3 ファイルを Strikethrough.htm として Configuration¥Objects¥Text フォルダに保存します。

JavaScript 関数の追加

この例では、ビヘイビアを定義して取り消し線オブジェクトのコードを挿入する JavaScript 関数を追加します。すべての API 関数をファイルの head セクションに配置する必要があります。Configuration¥Objects¥Text¥Em.htm などの既存のオブジェクトファイルは、同様の形式の関数およびコメントを使用します。

オブジェクト定義ファイルが使用する最初の関数は、isDOMRequired() です。この関数は、次に進む前に、デザインビューを既存のコードビューと同期する必要があるかどうかを指定します。しかし、例えば、上付き文字オブジェクトは、コードビュー内で他の多くのオブジェクトと共に使用される可能性があるので、強制的に同期させる必要はありません。

isDOMRequired() 関数の追加

1 Strikethrough.htm ファイルの head セクションで script 開始タグと終了タグの間に次の関数を追加します。

```
<script language="javascript">
    function isDOMRequired() {
        // Return false, indicating that this object is available in Code view.
        return false;
}
</script>
```

2 ファイルを保存します。

次に、objectTag() または insertObject() を次の関数で使用するかどうかを決めます。取り消し線オブジェクトは、選択したテキストを s タグで囲むだけなので、insertObject() 関数を使用する基準に当てはまりません（詳しくは、120 ページの「insertObject()」を参照してください）。

objectTag() 関数内で dw.getFocus() を使用して、コードビューが現在のビューであるかどうかを判別します。コードビューに入力フォーカスがある場合、関数は選択したテキストを適切なタグ（大文字または小文字）で囲みます。デザインビューに入力フォーカスがある場合、この関数は dom.applyCharacterMarkup() を使用して、選択したテキストに書式を割り当てます。この関数は、サポートされているタグにのみ機能します。詳しくは、「dom.applyCharacterMarkup()」（『Dreamweaver API リファレンス』）を参照してください。それ以外のタグまたは演算子では、別の API 関数の使用が必要となる場合があります。書式が適用されたら、メッセージや要求を表示せずに、挿入ポイント（カーソル）がドキュメントに戻ります。次の手順は、この objectTag() 関数を示します。

objectTag() 関数の追加

1 Strikethrough.htm ファイルの head セクションで isDOMRequired() 関数の後に次の関数を追加します。

```
function objectTag() {
    // Determine if the user is in Code view.
    var dom = dw.getDocumentDOM();
    if (dw.getFocus() == 'textView' || dw.getFocus(true) == 'html'){
        var upCaseTag = (dw.getPreferenceString("Source Format", "Tags Upper Case", "") ==
        'TRUE');
        // Manually wrap tags around selection.
        if (upCaseTag){
            dom.source.wrapSelection('<S>','</S>');
        }else{
            dom.source.wrapSelection('<s>','</s>');
        }
    // If the user is not in Code view, apply the formatting in Design view.
    }else if (dw.getFocus() == 'document'){
        dom.applyCharacterMarkup("s");
    }
    // Just return--don't do anything else.
    return;
}
```

2 ファイルを Strikethrough.htm として Configuration¥Objects¥Text フォルダに保存します。

HTML ファイルの head セクションに JavaScript 関数を追加する代わりに、別の JavaScript ファイルを作成できます。複数の関数を含むオブジェクトの場合、または別のオブジェクトから共有される可能性のある関数の場合は、JavaScript 関数を分離しておくと便利です。

サポートしている JavaScript 関数の HTML オブジェクト定義ファイルからの分離

1 新しい空白のファイルを作成します。

2 すべての JavaScript 関数をそのファイルにペーストします。

3 次の例で示すように、関数を Strikethrough.htm から削除し、JavaScript ファイル名を script タグの src 属性に追加します。

```
<html>
<head>
<title>Strikethrough</title>
<script language="javascript" src="Strikethrough.js">
</script>
</head>
<body>
</body>
</html>
```

4 Strikethrough.htm ファイルを保存します。

5 JavaScript 関数を追加したファイルを Strikethrough.js として Configuration¥Objects¥Text フォルダに保存します。

挿入バーのイメージの作成

1 次の図を参照して、挿入バーで使用する GIF イメージ（18 x 18 ピクセル）を作成します。

2 このファイルを Strikethrough.gif として Configuration¥Objects¥Text フォルダに保存します。

insertbar.xml ファイルの編集

次に、Dreamweaver で挿入バーのインターフェイスとこれらの 2 つの項目を関連付けるために、insertbar.xml ファイルを編集する必要があります。

注意：insertbar.xml ファイルを編集する前に、元のファイルを insertbar.xml.bak としてコピーしてバックアップを保存できます。

insertbar.xml ファイル内のコードでは、挿入バーのすべての既存オブジェクトを特定します。

- XML ファイルの各 category タグでは、インターフェイスのカテゴリーを作成します。
- 各 menubutton タグでは、挿入バーのポップアップメニューを作成します。
- XML ファイルの各 button タグでは、挿入バーにアイコンを配置し、適切な HTML ファイルまたは関数と関連付けます。

挿入バーへの新しいオブジェクトの追加

1 insertbar.xml ファイルの先頭付近で次の行を検索します。

```
<category id="DW_Insertbar_Common" MMString:name="insertbar/category/common" folder="Common">
```

この行は、挿入バーの一般カテゴリーの始まりを示しています。

2 category タグの後で改行し、button タグを挿入し、このタグに取り消し線オブジェクトの id、image および file 属性を割り当てます。

ID はボタンの一意の名前である必要があります（標準の命名規則に従い、このオブジェクトでは DW_Text_Strikethrough を使用します）。image 属性と file 属性は、サポートしているファイルの場所を Dreamweaver に指示します。以下に例を示します。

```
<button id="DW_Text_Strikethrough"
image="Text\Strikethrough.gif"
file="Text\Strikethrough.htm"/>
```

3 insertbar.xml ファイルを保存します。

4 拡張機能をリロードします（詳しくは、77 ページの「拡張機能のリロード」を参照してください）。

挿入バーの一般カテゴリーの先頭に、新しいオブジェクトが表示されます。

ダイアログボックスの追加

オブジェクトにフォームを追加し、指定したコードが挿入される前にユーザがパラメータを入力できるように設定できます。例えば、ハイパーリンクオブジェクトで、ユーザがテキスト、リンク、ターゲット、カテゴリーのインデックス、タイトル、アクセスキーの値を入力するように要求できます。この例では、前の例で使用した取り消し線オブジェクトにフォームを追加します。このフォームは、テキストカラーを赤に変更して strike-through タグを追加するオプションをユーザに提供するダイアログボックスを開きます。

この例では、Strikethrough.js という別の JavaScript ファイルを既に作成してあることを前提としています。

まず、Strikethrough.js で、ユーザがテキストカラーを変更したときにフォームから呼び出される関数を追加します。この関数は、取り消し線オブジェクトの objectTag() 関数に似ていますが、この関数はオプションです。

関数の作成

1 Strikethrough.js の objectTag() 関数の後に次のコードを入力して fontColorRed() という関数を作成します。

```
function fontColorRed(){
    var dom = dw.getDocumentDOM();
        if (dw.getFocus() == 'textView' || dw.getFocus(true) == 'html'){
            var upCaseTag = (dw.getPreferenceString("Source Format", "Tags Upper Case", "")
            == 'TRUE');
            // Manually wrap tags around selection.
            if (upCaseTag){
                 dom.source.wrapSelection('<FONT COLOR="#FF0000">','</FONT>');
            }else{
                dom.source.wrapSelection('<font color="#FF0000">','</font>');
                 }
         }else if (dw.getFocus() == 'document'){
                dom.applyFontMarkup("color", "#FF0000");
                }
    // Just return -- don't do anything else.
    return;
}
```

注意：dom.applyCharacterMarkup() はフォントカラーの変更をサポートしていないので、フォントカラーを変更するための適切な API 関数を見つける必要があります（詳しくは、『Dreamweaver API リファレンス』の「dom.applyFontMarkup()」を参照してください）。

2 ファイルを Strikethrough.js として保存します。

次に、Strikethrough.htm ファイルでフォームを追加します。この例で示すフォームは、ユーザがクリックしてオンにすると fontColorRed() 関数を呼び出す簡単なチェックボックスです。form タグを使用してフォームを定義し、table タグを使用してレイアウトを制御します。table タグを使用しないと、ダイアログボックスが単語の途中で折り返されたり、サイズが不自然になることがあります。

フォームの追加

1 body タグの後に次のコードを追加します。

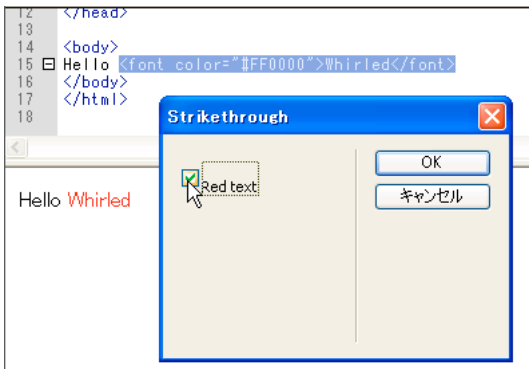
```
<form>
<table border="0" height="100" width="100">
    <tr valign="baseline">
    <td align="left" nowrap>
    <input type="checkbox" name="red" onClick=fontColorRed()>Red text</input>
    </td>
    </tr>
</table>
</form>
```

2 ファイルを Strikethrough.htm として保存します。

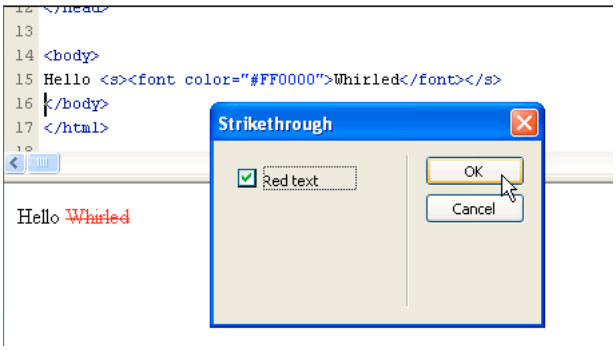
3 拡張機能をリロードします（詳しくは、77 ページの「拡張機能のリロード」を参照してください）。

ダイアログボックスのテスト

1 「Red text」チェックボックスをオンにします。

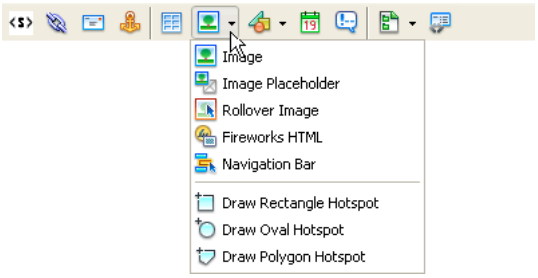


2 「OK」をクリックして objectTag() 関数を実行し、取り消し線を追加します。



挿入バーのポップアップメニューの作成

Dreamweaver の挿入バーに、オブジェクトの新しい整理方法が導入されました。挿入バーでポップアップメニューがサポートされ、次の図のように、オブジェクトを小さなグループに整理できるようになりました。



次の例では、挿入バーに「Editorial」という新しいカテゴリーを作成し、そのカテゴリーにポップアップメニューを追加します。既に作成した取り消し線オブジェクトをポップアップメニューに含め、このオブジェクトを、作成する Blue Text オブジェクトと同じグループに分類します。Editorial カテゴリーのオブジェクトを使用して、次の操作を実行できます。

- ファイルに編集上のコメントを残します。
- 削除するコンテンツに取り消し線を引き、新しいコンテンツを青色にします。

ファイルの整理

1 Dreamweaver のインストールフォルダに新しい Configuration¥Objects¥Editorial フォルダを作成します。

2 取り消し線オブジェクトの例で使用したファイル（Strikethrough.htm、Strikethrough.js、Strikethrough.gif）を Editorial フォルダにコピーします。

Blue Text オブジェクトの作成

1 HTML フォームを作成します。

2 次のコードを追加します。

```
<html>
<head>
<title>Blue Text</title>
<script language="javascript">
//--------------- API FUNCTIONS---------------
function isDOMRequired() {
    // Return false, indicating that this object is available in Code view.
    return false;
}
function objectTag() {
    // Manually wrap tags around selection.
    var dom = dw.getDocumentDOM();
    if (dw.getFocus() == 'textView' || dw.getFocus(true) == 'html'){
        var upCaseTag = (dw.getPreferenceString("Source Format", "Tags Upper Case", "") ==
        'TRUE');
         // Manually wrap tags around selection.
         if (upCaseTag){
              dom.source.wrapSelection('<FONT COLOR="#0000FF">','</FONT>');
        }else{
              dom.source.wrapSelection('<font color="#0000FF">','</font>');
              }
     }else if (dw.getFocus() == 'document'){
            dom.applyFontMarkup("color", "#0000FF");
            }
    // Just return -- don't do anything else.
    return;
}
</script>
</head>
<body>
</body>
</html>
```

3 ファイルを AddBlue.htm として Editorial フォルダに保存します。

次に、Blue Text オブジェクトのイメージを作成します。

イメージの作成

1 18 x 18 ピクセルの GIF ファイルを作成します。イメージの外観は次のようになります。

2 イメージを AddBlue.gif という名前で Editorial フォルダに保存します。

次に、insertbar.xml ファイルを編集します。このファイルでは、挿入バーのオブジェクトとその場所を定義します。様々な menubutton タグと その属性が category タグには含まれています。これらの menubutton タグでは、HTML カテゴリーの各ポップアップメニューを定義します。menubutton タグ内では、各 button タグによってポップアップメニューの項目が定義されます。

insertbar.xml の編集

1 ファイルの先頭付近でコードの次の行を検索します。

```
<insertbar xmlns:MMString="http://www.adobe.com/schemes/data/string/">
```

insertbar タグが挿入バーの内容全体の先頭です。

2 その行の後に、作成する Editorial カテゴリーを指定する新しい category タグを追加します。次に示す例のように、固有の ID、名前、フォルダ属性を指定し、category の終了タグを追加します。

```
<category id="DW_Insertbar_Editorial" name="Editorial" folder="Editorial">
</category>
```

3 拡張機能をリロードします。拡張機能のリロードについては、77 ページの「拡張機能のリロード」を参照してください。

挿入バーに Editorial カテゴリーが表示されます。

4 category の開始タグと終了タグの間に、menubutton タグと次の属性を使用してポップアップメニューを追加します。このとき、固有の ID も含めます。

```
<menubutton id="DW_Insertbar_Markup" name="markup" image="Editorial\Strikethrough.gif"
folder="Editorial">
```

属性について詳しくは、107 ページの「挿入バーを定義するタグの属性」を参照してください。

5 新しいポップアップメニューのオブジェクトを button タグを使用して追加します。

```
<button id="DW_Editorial_Strikethrough" image="Editorial\Strikethrough.gif"
file="Editorial\Strikethrough.htm"/>
```

6 取り消し線オブジェクトボタンのタグの後に、ハイパーテキストオブジェクトを次のように追加します。

```
<button id="DW_Blue_Text" image="Editorial\AddBlue.gif name="Blue Text" file="Editorial\AddBlue.htm"/>
```

注意：button タグには、独立した終了タグはありません。"/>" で終了します。

7 </menubutton> 終了タグを使用して、ポップアップメニューを終了します。

次のコードでは、ポップアップメニューと 2 つのオブジェクトを追加したカテゴリー全体を示します。

```
<category id="DW_Insertbar_Editorial" name="Editorial" folder="Editorial">
    <menubutton id="DW_Insertbar_Markup" name="markup"
    image="Editorial\Strikethrough.gif" folder="Editorial">
        <button id="DW_Editorial_Strikethrough"
        image="Editorial\Strikethrough.gif" file="Editorial\Strikethrough.htm"/>
        <button id="DW_Blue_Text" image="Editorial\AddBlue.gif" name="Blue Text"
        file="Editorial\AddBlue.htm"/>
    </menubutton>
</category>
```

新しいポップアップメニューのテスト

1 拡張機能をリロードします。拡張機能のリロードについては、77 ページの「拡張機能のリロード」を参照してください。

2 Editorial メニューをクリックします。

次のポップアップメニューが表示されます。

オブジェクト API 関数

この節では、オブジェクト API の関数について説明します。insertObject() 関数または objectTag() 関数のいずれかを定義する必要があります。これらの関数について詳しくは、120 ページの「insertObject()」を参照してください。その他の関数はオプションです。

canInsertObject()

対応バージョン
Dreamweaver MX

説明
この関数では、オブジェクトダイアログボックスを表示するかどうかを決定します。

引数
なし。

戻り値
ブール値。

例
次のコードでは、ドキュメントに特定のストリングが含まれていることが確認されてから、選択したオブジェクトをユーザが挿入できるようになります。

```
function canInsertObject(){
    var docStr = dw.getDocumentDOM().documentElement.outerHTML;
    var patt = /hava/;
    var found = ( docStr.search(patt) != -1 );
    var insertionIsValid = true;
    if (!found){
        insertionIsValid = false;
        alert("the document must contain a 'hava' string to use this object.");}
    return insertionIsValid;}
```

displayHelp()

説明
この関数が定義されている場合は、パラメータダイアログボックスの「OK」と「キャンセル」の下に「ヘルプ」ボタンが表示されます。この関数は、ユーザが「ヘルプ」ボタンをクリックすると呼び出されます。

引数
なし。

戻り値
なし。

例
次の例では、ブラウザで myObjectHelp.htm ファイルを表示します。このファイルは、拡張機能の使用方法を説明します。

```
function displayHelp(){
    var myHelpFile = dw.getConfigurationPath() +
        '/ExtensionsHelp/myObjectHelp.htm';
    dw.browseDocument(myHelpFile);
}
```

isDOMRequired()

説明
この関数では、コマンドを実行するのに有効な DOM がオブジェクトに必要かどうかを判断します。この関数から true の値が返された場合、またはこの関数が定義されていない場合、Dreamweaver はコマンドに有効な DOM が必要であるとみなし、ドキュメントのコードビューとデザインビューを同期させてからコマンドを実行します。同期を行うことにより、コードビューのすべての編集内容がデザインビューに更新されます。

引数
なし。

戻り値
コマンドを実行するために有効な DOM が必要であれば true、不要であれば false。

insertObject()

対応バージョン
Dreamweaver MX

説明

この関数は、objectTag() 関数が定義されていない場合は必須です。この関数は、ユーザが「OK」をクリックすると呼び出されます。この関数は、ユーザのドキュメントにコードを挿入してダイアログボックスを閉じるか、エラーメッセージを表示してダイアログボックスを開いたままにします。これは、objectTag() 関数の代わりにオブジェクトで使用される代替関数として機能します。ユーザが現在の挿入ポイントにテキストを挿入しているとはみなさず、ユーザが「OK」をクリックしたときにデータを検証できます。次の条件のいずれかに該当する場合は、insertObject() 関数を使用する必要があります。

- 複数の位置にコードを挿入する必要がある場合。
- 挿入ポイント以外の位置にコードを挿入する必要がある場合。
- コードを挿入する前に入力を検証する必要がある場合。

以上の条件に該当しない場合は、objectTag() 関数を使用します。

引数

なし。

戻り値

エラーメッセージを含むストリングまたは空白のストリング。空白のストリングが返された場合は、ユーザが「OK」をクリックすると、オブジェクトダイアログボックスが閉じます。空白でない場合は、エラーメッセージが表示され、ダイアログボックスは表示されたままになります。

イネーブラ

canInsertObject()

例

次の例では、コードを挿入する前に入力を検証する必要があるため、insertObject() 関数を使用します。

```
function insertObject() {
     var theForm = document.forms[0];
     var nameVal = theForm.firstField.value;
     var passwordVal = theForm.secondField.value;
     var errMsg = "",
     var isValid = true;
     // ensure that field values are complete and valid
     if (nameVal == "" || passwordVal == "") {
         errMsg = "Complete all values or click Cancel."
     } else if (nameVal.length < 4 || passwordVal.length < 6) {
            errMsg = "Your name must be at least four characters, and your password at
            least six";
    }
    if (!errMsg) {
         // do some document manipulation here. Exercise left to the reader
        }
    return errMsg;
}
```

objectTag()

説明

objectTag() 関数と insertObject() 関数は互いに排他的です。1 つのドキュメント内で両方の関数を定義すると、objectTag() 関数が使用されます。詳しくは、120 ページの「insertObject()」を参照してください。

この関数では、ユーザのドキュメントにコードストリングを挿入します。空白のストリングまたは null（"return"）が返されるのは、Dreamweaver に対して何も実行しないように指示することを意味します。

注意：return ステートメントより前に編集が手動で実行されていることを前提としているため、この場合、何も実行しないことは「キャンセル」をクリックすることと同じではありません。

Dreamweaver では、コードビューにフォーカスがあり、挿入ポイントが置かれているのではなく範囲が選択されている場合、その選択範囲が objectTag() 関数によって返されるストリングで置き換えられます。これは true となります（objectTag() 関数で空白のストリングが返される場合や、何も返されない場合も同様です）。objectTag() 関数から空白のストリングまたは null が返されるのは、ドキュメントが既に手動で編集されているためです。そうでない場合は、通常、選択範囲が二重引用符（""）で置き換えられ、編集内容が削除されます。

引数
なし。

戻り値
ユーザのドキュメントに挿入されるストリング。

例
次の objectTag() 関数の例は、ある特定の ActiveX コントロールとプラグインに対する一組の OBJECT/EMBED を挿入します。

```
function objectTag() {
return '\n' +
'<OBJECT CLASSID="clsid:166F1OOB-3A9R-11FB-8075444553540000" \n'¬
+ 'CODEBASE="http://www.mysite.com/product/cabs/¬
myproduct.cab#version=1,0,0,0" \n' + 'NAME="MyProductName"> \n' ¬
+ '<PARAM NAME="SRC" VALUE=""> \n' + '<EMBED SRC="" HEIGHT="" ¬
WIDTH="" NAME="MyProductName"> \n' + '</OBJECT>'
}
```

windowDimensions()

説明
この関数では、オプションダイアログボックスに特定のサイズを設定します。この関数が定義されていない場合、ウィンドウのサイズは自動的に計算されます。

注意：640 x 480 ピクセル以上のオプションダイアログボックスを使用する場合に限り、この関数を定義するようにしてください。

引数
platform

- **platform** 引数の値は、ユーザのプラットフォームに応じて、"macintosh" または "windows" になります。

戻り値
"widthInPixels,heightInPixels" という形式のストリング。

返されるサイズはダイアログボックス全体のサイズよりも小さくなります。これは「OK」と「キャンセル」に使用する領域が含まれていないためです。返されたサイズにすべてのオプションが収まらない場合は、スクロールバーが表示されます。

例

次の windowDimensions() 関数の例では、パラメータダイアログボックスのサイズを、Windows の場合は 648 x 520 ピクセルに、Macintosh の場合は 660 x 580 ピクセルにそれぞれ設定します。

```
function windowDimensions(platform){
    var retval = ""
    if (platform = = "windows"){
    retval = "648, 520";
    }else{
    retval = "660, 580";
    }
    return retval;
}
```

第 8 章：ブラウザ互換性チェックの問題用 API

Adobe Dreamweaver では、BCC（browser compatibility check：ブラウザ互換性チェック）機能を利用して、ブラウザレンダリングバグを引き起こすおそれのある HTML および CSS の組み合わせを見つけることによって、どのブラウザでも正常に動作する（つまり、同じ外観と機能を備えた）ページレイアウトを作成することができます。この機能では、ユーザのドキュメントを検索して問題のある HTML と CSS の組み合わせを見つけるのに JavaScript を使用します。JavaScript コードは、問題検出ファイルと呼ばれる HTML ファイルに保存されます。これらのファイルが正常に機能するためには、ファイルを Configuration¥BrowserProfiles¥Issues¥ フォルダに保存する必要があります。

検出機能について

ユーザが、最初にブラウザ互換性チェックの実行を選択した場合（およびターゲットブラウザダイアログボックスで「OK」をクリックした場合）、次のような一連のイベントが発生します。

1. 選択したブラウザ用のプロファイルが Configuration¥BrowserProfiles¥ フォルダから読み込まれます。
2. 各問題の一意の ID を取得するために Configuration¥BrowserProfiles¥Issues¥ フォルダ内の各問題用ファイルにある getIssueID() 関数が呼び出されます。
3. 各問題に対して getAffectedBrowserDisplayNames() 関数（定義されている場合）が呼び出されます。
4. その問題が、選択したブラウザに影響を与えるかどうかを判別するために getAffectedBrowserProfiles() 関数が呼び出されます。
5. 問題の検出時に結果パネルに表示される名前を決定するために、各問題に対して getIssueName() 関数が呼び出されます。
6. 問題の検出時に結果パネルの右側およびツールヒント（コードビューでユーザが問題上にマウスを置いたとき）に表示されるテキストを決定するために getIssueDescription 関数が呼び出されます。

前述の手順 6 の後に BCC 設定ダイアログボックスでブラウザを選択するたび、およびこの後にブラウザ互換性チェックを実行するたびに次の一連のイベントが発生します。

一連のイベント

1. 現在のドキュメントに適用するスタイルは影響を受けるブラウザによって読み込まれるため、インライン、ヘッダー、外部スタイルシート内でこのスタイルが定義されているかどうかが解析されます。
2. 影響を受けるブラウザに適用される各問題用ファイルにある findIssue() 関数が呼び出されます。

問題例

Configuration¥BrowserProfiles/Issues¥ フォルダにあるファイル ColAndColgroupCapturedByCaption.htm および ColAndColgroupCapturedByCaption.js の例を以下に示します。

ColAndColgroupCapturedByCaption.htm

```
<!DOCTYPE HTML SYSTEM "-//
//DWExtension layout-engine 5.0//dialog">
<html>
<head>
<title>Col and Colgroup Captured by Caption</title>

<script src="../../Shared/Common/Scripts/dwscripts.js"></script>
<script src="issue_utils.js"></script>
<script src="ColAndColgroupCapturedByCaption.js"></script>
<script>
//--------------- LOCALIZEABLE GLOBALS------------------
var ISSUE_NAME = "Col and Colgroup/Caption Conflict";
var ISSUE_DESC = "If the caption tag is placed directly after the opening table tag as required by the HTML
4.01 specification, any styles applied to col and colgroup tags in the same table are ignored.";

//--------------- END LOCALIZEABLE---------------
</script>
</head>

<body>
</body>
</html>
```

ColAndColgroupCapturedByCaption.js

```
function findIssue(){
  var DOM = dw.getDocumentDOM();
  var issueNodes = new Array();

  if (DOM){
    // first see if there are any caption tags in the doc.
    var captions = DOM.getElementsByTagName('caption');

    // declare a mess of variables that we'll need in the
    // for loop below.
    var currCap = null, props = null, parentTable = null;
    var colgroups = null, cols = null, allcol = null;
    var property = "", definedStyles = new Array();

    // ok, now loop through all the captions, if any.
    for (var i=0; i < captions.length; i++){
      currCap = captions[i];
      parentTable = currCap.parentNode;

      // the caption is only a problem if it's in the valid
      // spot (i.e., the first child of the table)
      if (currCap == parentTable.childNodes[0]){

        // find all colgroup and col tags that are in the
        // same table as the caption.
        colgroups = parentTable.getElementsByTagName('colgroup');
        cols = parentTable.getElementsByTagName('col');
        allcol = colgroups.concat(cols);

        for (var x=0; x < allcol.length; x++){
          // if styles are declared for any colgroup or col
          // tag in this table, we have a problem node. don't
          // bother looking further.
          props = window.getDeclaredStyle(allcol[x]);
          property = "";
          definedStyles.length = 0;
```

```
          for (property in props) {
            definedStyles.push(property);
          }
          if (definedStyles.length > 0){
            issueNodes.push(currCap);
            break;
          }
        }
      }
    }
  }
  return issueNodes;
}
function getAffectedBrowserDisplayNames(){
  return new Array("Safari 2.0");
}
function getAffectedBrowserProfiles(){
  return new Array("Safari 2.0");
}
function getIssueID(){
  return "COL_AND_COLGROUP_CAPTURED_BY_CAPTION";
}
function getIssueName(){
  return ISSUE_NAME;
}
function getIssueDescription(){
  return ISSUE_DESC;
}
function getConfidenceLevel(){
  //DETCON 4
  return issueUtils.CONFIDENCE_HIGH;
}
```

問題用 API 関数

問題用 API の関数は、getAffectedBrowserDisplayNames() を除き、すべてが必須の関数です。すべての拡張 API と同様に、各関数の body 部を作成し、適切な値を Dreamweaver に戻す必要があります。ブラウザ互換性チェック用の関数について詳しくは、『Dreamweaver API リファレンス』のトピック「ページのコンテンツ」を参照してください。

findIssue()

対応バージョン
Dreamweaver CS3

説明
特定のブラウザのレンダリングにおける問題を引き起こす CSS および HTML の組み合わせに該当するドキュメントを検索します。

引数
なし。

戻り値
問題を表すエレメントノードの配列。ユーザが、いずれかのブラウザ互換性の問題から次の問題へ移動すると、これらのノードが選択されます。

例
次の findIssue() 関数は <button> タグ（float: left または float: right が適用されたタグ）の配列を返します。

```
function findIssue(){
    var DOM = dw.getDocumentDOM();
    var issueNodes = new Array();
    var buttons = DOM.getElementsByTagName('button');
    var props = null;
    for (var i=0; i < buttons.length; i++){
        props = window.getDeclaredStyle(buttons[i]);
        if (props.cssFloat == "left" || props.cssFloat == "right"){
            issueNodes.push(buttons[i]);
        }
    }
    return issueNodes;
}
```

getAffectedBrowserProfiles()

対応バージョン
Dreamweaver CS3

説明
この問題に関連するブラウザのリストを提供します。

引数
なし。

戻り値
ブラウザ名の配列。それぞれのブラウザ名は、有効なブラウザプロファイル内の最初の行に完全に一致する必要があります（Configuration¥BrowserProfiles フォルダにあるテキスト（TXT）ファイルを参照）。

例

```
function getAffectedBrowsers(){
    return new Array("Microsoft Internet Explorer 5.0",
        "Microsoft Internet Explorer 5.5",
        "Microsoft Internet Explorer 6.0");
}
```

getAffectedBrowserDisplayNames()

対応バージョン
Dreamweaver CS3

説明
この問題に関連する、ユーザに表示されるブラウザ名のリストを提供します。この関数はオプションです。省略した場合は、getAffectedBrowserProfiles() によって提供されるプロファイル名が代わりに使用されます。

引数
なし。

戻り値
ブラウザ名の配列。この配列は、getAffectedBrowserProfiles() によって返される配列に対応している必要があります。

例
```
function getAffectedBrowsers(){
    return new Array("IE/Win 5.0",
        "IE/Win 5.5",
        "IE/Win 6.0");

}
```

getIssueID()

対応バージョン
Dreamweaver CS3

説明
問題に対する一意の ID を提供します。

引数
なし。

戻り値
一意の問題 ID を表すストリング。

例
```
function getIssueID() {
    return "EXPANDING_BOX_PROBLEM";
}
```

getIssueName()

対応バージョン
Dreamweaver CS3

説明
問題の名前または簡単な説明を提供します。

引数
なし。

戻り値
問題の名前または簡単な説明を含むストリング。

例

```
function getIssueName(){
    return "The Expanding Box Problem";
}
```

getIssueDescription()

対応バージョン

Dreamweaver CS3

説明

問題の詳細な説明を提供します。

引数

なし。

戻り値

問題の名前または簡単な説明を含むストリング。

例

```
function getIssueDescription(){
    return "Fixed-dimension boxes will incorrectly expand to fit their
            content instead of clipping content at the specified width
            or height.";
}
```

第 9 章：コマンド

Adobe Dreamweaver のコマンドを使用して、現在使用中のドキュメント、その他の開いているドキュメント、ローカルドライブにあるすべての HTML ドキュメントに対して、ほぼすべての種類の編集を行うことができます。コマンドを使用すると、HTML タグとその属性、コメントおよびテキストを挿入、削除または再配置することができます。

コマンドは HTML ファイルです。コマンドファイルの body セクションには、コマンドのオプション（テーブルのソート方法とソートの基準とする列など）を入力するための HTML フォームを含めることができます。また、コマンドファイルの head セクションには、body セクションのフォーム入力を処理し、ユーザのドキュメントへの編集内容をコントロールする JavaScript 関数が含まれます。

次の表に、コマンドを作成するために使用するファイルの一覧を示します。

パス	**ファイル**	**説明**
Configuration¥Commands¥	**commandname**.htm	ユーザインターフェイスを指定します。
Configuration¥Commands¥	**commandname**.js	実行する関数が含まれています。

コマンドのしくみ

コマンドを含むメニューをユーザがクリックすると、次のような一連のイベントが発生します。

1. Dreamweaver により canAcceptCommand() 関数が呼び出され、メニュー項目を無効化するかどうかが決定されます。canAcceptCommand() 関数から値 false が返された場合、そのコマンドはメニュー内で淡色表示になり、プロシージャは停止します。canAcceptCommand() 関数から値 true が返された場合、そのプロシージャは続行できます。
2. ユーザがメニューからコマンドを選択します。
3. 選択されたコマンドファイルに receiveArguments() 関数が定義されている場合、Dreamweaver によりこの関数が呼び出され、メニュー項目または dreamweaver.runCommand() 関数から渡された引数が処理されます。dreamweaver.runCommand() 関数について詳しくは、『Dreamweaver API リファレンス』を参照してください。
4. commandButtons() 関数が定義されている場合、この関数が呼び出されて、オプションダイアログボックスの右側に表示されるボタンと、ユーザがそのボタンをクリックしたときに実行されるコードが決定されます。
5. Dreamweaver によりコマンドファイルの form タグが検索されます。フォームが存在する場合、windowDimensions() 関数が呼び出され、ファイルの body エレメントを表示するオプションダイアログボックスのサイズが決定されます。windowDimensions() 関数が定義されていない場合、ダイアログボックスのサイズは自動的に決定されます。
6. コマンドファイルの body タグに onLoad ハンドラがある場合は、ダイアログボックスが表示されるかどうかにかかわらず、Dreamweaver によりハンドラが実行されます。ダイアログボックスが表示されない場合、この後のイベントは発生しません。
7. ユーザがコマンドのオプションを選択します。ユーザがフィールドに進むたびに、フィールドに関連付けられたイベントハンドラが実行されます。
8. ユーザが commandButtons() 関数によって定義されているボタンのいずれかをクリックします。
9. 関連付けられているコードが実行されます。コマンドのスクリプトのいずれかが window.close() 関数を呼び出すと、ダイアログボックスが閉じます。

コマンドメニューへのコマンドの追加

Dreamweaver により、Configuration¥Commands フォルダ内のすべてのファイルが、コマンドメニューの最後に自動的に追加されます。コマンドメニューにコマンドが表示されないようにするには、ファイルの最初の行に次のコメントを挿入します。

```
<!-- MENU-LOCATION=NONE -->
```

この行が存在する場合、そのファイルに対してはメニュー項目が作成されないため、コマンドを実行するには dw.runCommand() を呼び出す必要があります。

簡単なコマンドの例

この簡単な拡張機能は、コマンドメニューに項目を追加し、ドキュメントで選択したテキストを大文字または小文字に変換できるようにします。メニュー項目をクリックすると、選択内容を送信するための 3 つのボタンを備えたインターフェイスがアクティブになります。

この拡張機能を作成するには、UI を作成し、JavaScript コードを記述し、拡張機能をテストします。

この例では、UI を格納する Change Case.htm と JavaScript コードを格納する Change Case.js という 2 つのファイルを Commands フォルダに作成します。必要に応じて Change Case.htm ファイルのみを作成して、JavaScript コードを head セクションに格納することもできます。

ユーザインターフェイスの作成

このユーザインターフェイスは、ユーザが大文字または小文字を選択できる 2 つのオプションを含むフォームです。

1 空白のファイルを作成します。

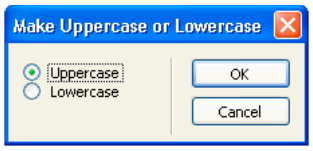
2 次のコードをファイルに追加してフォームを作成します。

```
<!DOCTYPE HTML SYSTEM "-//Adobe//DWExtension layout-engine 10.0//dialog">
<HTML>
<HEAD>
<Title>Make Uppercase or Lowercase</Title>
<SCRIPT SRC="Change Selection Case.js"></SCRIPT>
</HEAD>
<BODY>
<form name="uorl">
    <table border="0">
        <!--DWLayoutTable-->
        <tr>
            <td valign="top" nowrap> <p>
                <label>
                <input type="radio" name="RadioGroup1" value="uppercase" checked>
                Uppercase</label>
                <br>
                <label>
                <input type="radio" name="RadioGroup1" value="lowercase">
                Lowercase</label>
            </p></td>
        </tr>
    </table>
</form>
</BODY>
</HTML>
```

3 ファイルを Change Case.htm として Configuration¥Commands フォルダに保存します。

Title タグの内容である Make Uppercase or Lowercase が、ダイアログボックス上部のバーに表示されます。フォーム内では、2 つのセルを持つテーブルによってエレメントのレイアウトが制御されます。テーブルのセルには「大文字」と「小文字」の 2 つのオプションがあります。「大文字」オプションには checked 属性があるため、このボタンがデフォルトの選択になります。また、ユーザの選択肢は 2 つのオプションのいずれかを選択するか、コマンドをキャンセルするかです。

フォームの外観は次の図のとおりです。



commandButtons() 関数は、ユーザが選択内容の送信または操作のキャンセルを行うための「OK」と「キャンセル」を提供します。詳しくは、136 ページの「commandButtons()」を参照してください。

JavaScript コードの記述

次の例は、Dreamweaver によって呼び出される canAcceptCommand() と commandButtons() という 2 つの拡張機能 API 関数および changeCase() という 1 つのユーザ定義の関数から構成されています。ユーザ定義の関数は commandButtons() 関数によって呼び出されます。

この例では、次のタスクを実行するための JavaScript を作成します。

コマンドを有効または無効（淡色表示）にするかを決定する

コマンド作成の最初のタスクは、メニュー項目をアクティブにする場合と淡色表示にする場合を決定することです。ユーザがコマンドメニューをクリックすると、Dreamweaver により各メニュー項目の canAcceptCommand() 関数が呼び出され、各メニュー項目を有効にする必要があるかどうかが判別されます。canAcceptCommand() が値 true を返す場合、そのメニュー項目のテキストはアクティブ（有効）であると扱われ、表示されます。canAcceptCommand() が値 false を返す場合、そのメニュー項目は淡色表示になります。この例では、ユーザがドキュメント内のテキストを選択しているときは、メニュー項目がアクティブになります。

1 新しい空白のファイルを作成します。

2 次のコードを追加します。

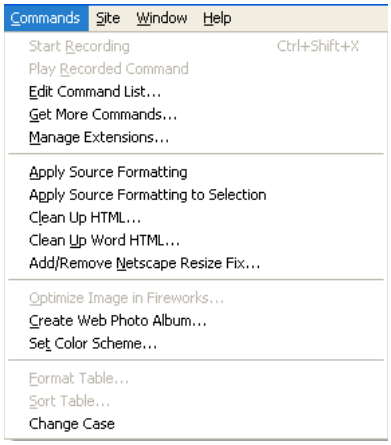
```
function canAcceptCommand(){
    var theDOM = dw.getDocumentDOM(); // Get the DOM of the current document
    var theSel = theDOM.getSelection(); // Get start and end of selection
    var theSelNode = theDOM.getSelectedNode(); // Get the selected node
    var theChildren = theSelNode.childNodes; // Get children of selected node
    return (theSel[0] != theSel[1] && (theSelNode.nodeType == Node.TEXT_NODE¬
        || theSelNode.hasChildNodes() && (theChildren[0].nodeType == ¬
        Node.TEXT_NODE)));
}
```

3 このファイルを Change Case.js として Configuration¥Commands フォルダに保存します。

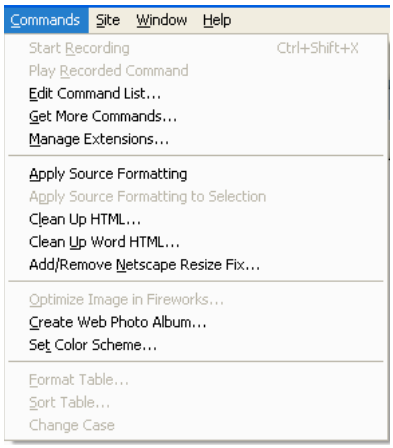
canAcceptCommand() 関数の最初の行では、ユーザのドキュメントの DOM を取得し、ドキュメントオブジェクトに関して getSelection() 関数を呼び出すことにより、選択されたテキストを取得します。次に、以下のコードで示すように、関数は選択されたテキストを含むノードと、それに続く子ノードを取得します。次に、最後の行で、選択範囲またはその最初の子がテキストであるかどうかがチェックされ、その結果が true または false の値として返されます。

return ステートメントの最初の部分（theSel[0] != theSel[1]）で、ユーザがドキュメント内で何かを選択しているかどうかがチェックされます。変数 theSel は、2 つのスロットからなる配列で、ドキュメント内の選択範囲の開始および終了オフセットを保持します。2 つの値が等しくない場合は、内容が選択されています。2 つのスロットの値が等しい場合は、挿入ポイントだけがあり、何も選択されていません。

return ステートメントの次の部分（&& (theSelNode.nodeType == Node.TEXT_NODE）では、選択したノードタイプがテキストであるかどうかがチェックされます。テキストである場合は、canAcceptCommand() 関数が値 true を返します。ノードタイプがテキストでない場合は、テストが続行され、ノードに子があるかどうか（|| theSelNode.hasChildNodes()）、および最初の子ノードがテキストかどうか（&&(theChildren[0].nodeType == Node.TEXT_NODE))）がチェックされます。両方の条件が true である場合、canAcceptCommand() は値 true を返し、次の図で示すようにコマンドメニューの下部にあるメニュー項目が有効化されます。



それ以外の場合、canAcceptCommand() は値 false を返し、次の図で示すようにメニュー項目が淡色表示されます。



「OK」および「キャンセル」に関数をリンクする

ユーザが「OK」または「キャンセル」をクリックした場合に、拡張機能で適切なアクションを実行する必要があります。適切なアクションを決定するには、どちらかのボタンがクリックされたときに実行する JavaScript 関数を指定します。

1 Configuration¥Commands フォルダの Change Case.js ファイルを開きます。

2 次のコードをファイルの末尾に追加します。

```
function commandButtons() {
    return new Array("OK", "changeCase()", "Cancel", "window.close()");
}
```

3 ファイルを保存します。

commandButtons() 関数を使用すると、Dreamweaver により「OK」と「キャンセル」が提供され、ユーザがこれらのボタンをクリックしたときの動作を Dreamweaver に指示することができます。commandButtons() 関数が Dreamweaver に対して、ユーザが「OK」をクリックしたときには changeCase() を呼び出し、「キャンセル」をクリックしたときには window.close() を呼び出すように指示します。

大文字または小文字をユーザが選択できるようにする

ユーザがメニュー項目をクリックしたとき、拡張機能ではユーザが大文字か小文字を選択できるメカニズムが必要です。この UI では、2 つのラジオボタンを定義することによってこのメカニズムを提供し、ユーザが大文字か小文字を選択できるようにしています。

1 Change Case.js ファイルを開きます。

2 次のコードをファイルの末尾に追加します。

```
function changeCase() {
    var uorl;
    // Check whether user requested uppercase or lowercase.
    if (document.forms[0].elements[0].checked)
        uorl = 'u';
    else
        uorl = 'l';
    // Get the DOM.
    var theDOM = dw.getDocumentDOM();
    // Get the outerHTML of the HTML tag (the
    // entire contents of the document).
    var theDocEl = theDOM.documentElement;
    var theWholeDoc = theDocEl.outerHTML;
    // Get the node that contains the selection.
    var theSelNode = theDOM.getSelectedNode();
    // Get the children of the selected node.
    var theChildren = theSelNode.childNodes;
    var i = 0;
    if (theSelNode.hasChildNodes()){
        while (i < theChildren.length){
            if (theChildren[i].nodeType == Node.TEXT_NODE){
                var selText = theChildren[i].data;
                var theSel = theDOM.nodeToOffsets(theChildren[0]);
                break;
            }
            ++i;
        }
    }
    else {
    // Get the offsets of the selection
    var theSel = theDOM.getSelection();
    // Extract the selection
    var selText = theWholeDoc.substring(theSel[0],theSel[1]);
}
if (uorl == 'u'){
        theDocEl.outerHTML = theWholeDoc.substring(0,theSel[0]) +
        selText.toUpperCase() + theWholeDoc.substring(theSel[1]);
    }
    else {
        theDocEl.outerHTML = theWholeDoc.substring(0,theSel[0]) +
        selText.toLowerCase() + theWholeDoc.substring(theSel[1]);
    }
// Set the selection back to where it was when you started
    theDOM.setSelection(theSel[0],theSel[1]);
    window.close(); // close extension UI
}
```

3 このファイルを Change Case.js として Configuration¥Commands フォルダに保存します。

changeCase() 関数はユーザ定義の関数であり、ユーザが「OK」をクリックすると、commandButtons() 関数により呼び出されます。この関数は、選択されたテキストを大文字または小文字に変更します。この UI ではラジオボタンが使用されるため、2 つの選択肢のどちらかがオンになっていることを前提にしてコードを記述できます。ユーザがどちらも選択しない場合を想定してテストを行う必要はありません。

changeCase() 関数は、最初にプロパティ document.forms[0].elements[0].checked をテストします。document.forms[0].elements[0] プロパティは、現在のドキュメントオブジェクトで最初のフォームにある最初のエレメント（拡張機能の UI）を表します。checked プロパティの値は、エレメントがオン（有効）であるときは true、オフ（無効）であるときは false です。インターフェイスでは、elements[0] が最初のラジオボタン（「大文字」ボタン）を示します。ユーザが「OK」をクリックするときは、必ずどちらかのラジオボタンがオンになっているため、コードでは、大文字が選択されていない場合は小文字が選択されているものと見なされます。関数は、変数 uorl を u または l に設定し、ユーザの応答を保存します。

関数の残りのコードでは、選択されたテキストを取得し、大文字または小文字の指定された方に変換し、ドキュメント内の元の場所にコピーします。

ユーザのドキュメントで選択されたテキストを取得するために、関数は DOM を取得します。次に、関数は、ドキュメントのルートエレメント（html タグ）を取得します。最後に、ドキュメント全体を theWholeDoc 変数に抽出します。

次に、changeCase() が getSelectedNode() を呼び出して、選択されたテキストを含むノードを取得します。また、選択範囲がテキストを含むタグである場合は、子ノードも取得します（theSelNode.childNodes）。このタグの例には text などがあります。

子ノードがある場合（hasChildNodes() が値 true を返す場合）は、コマンドが子ノードをループしてテキストノードを検索します。テキストノードが検出された場合は、テキスト（theChildren[i].data）が selText に保存され、テキストノードのオフセットが theSel に保存されます。

子ノードがない場合は、コマンドが getSelection() を呼び出し、選択範囲の開始および終了オフセットを theSel に保存します。次に、これら 2 つのオフセット間のストリングを抽出し、selText に保存します。

次に、関数は、uorl 変数を調べて、ユーザが大文字を選択したかどうかを判別します。ユーザが大文字を選択した場合、関数によって、HTML コードがドキュメントの 3 つの部分別に記述されます。先頭部分はドキュメントの先頭から選択範囲の先頭まで、次は選択したテキストを大文字に変換したもの（selText.toUppercase()）、最後は選択したテキストの末尾からドキュメントの末尾までです。

ユーザが小文字を選択した場合、関数は同じ操作を行いますが、selText.toLowerCase() を呼び出すことにより、選択したテキストを小文字に変換します。

最後に、changeCase() によって選択範囲をリセットし、window.close() を呼び出して UI を閉じます。

拡張機能のテスト

Commands フォルダ内にファイルを配置した後に、拡張機能をテストすることができます。

1 Dreamweaver を再起動するか、拡張機能をリロードします。拡張機能のリロードについては、77 ページの「拡張機能のリロード」を参照してください。

「大文字 / 小文字の変更」エントリがコマンドメニューに表示されます。

2 ドキュメント内にいくつかテキストを入力します。

3 テキストを選択します。

注意：「大文字 / 小文字の変更」は、ユーザがドキュメント内のテキストを選択するまで淡色表示になっています。

4 コマンドメニューから「大文字 / 小文字の変更」を選択します。

テキストの大文字と小文字が変更されます。

コマンド API 関数

コマンド API のカスタム関数は必須ではありません。

canAcceptCommand()

説明
この関数は、コマンドが現在の選択範囲に適しているかどうかを判断します。

注意：canAcceptCommand() は、値 false を返すケースが少なくとも 1 つある場合に限り定義するようにしてください。この関数が定義されていない場合、コマンドは選択範囲に適していると見なされます。これにより、時間を節約し、パフォーマンスを向上させることができます。

引数
なし。

戻り値
コマンドが許可される場合は true。値が false の場合、メニュー内のコマンドが淡色表示になります。

例
次の canAcceptCommand() 関数の例は、選択範囲がテーブルである場合にのみコマンドを使用可能にします。

```
function canAcceptCommand(){
    var retval=false;
    var selObj=dw.getDocumentDOM.getSelectedNode();
    return (selObj.nodeType == Node.ELEMENT_NODE && ¬
    selObj.tagName=="TABLE");{
        retval=true;
}
return retval;
}
```

commandButtons()

説明
この関数は、オプションダイアログボックスに表示されるボタンと、それらのボタンをクリックした際のビヘイビアを定義します。この関数が定義されていない場合、ボタンは表示されず、コマンドファイルの body セクションがダイアログボックス全体に拡大表示されます。

デフォルトでは、これらのボタンはダイアログボックスの上部に表示されます。commandButtons() 関数で 2 つの追加の値を指定することにより、これらのボタンをダイアログボックスの下部に表示できます。

デフォルトでは、ボタンは右揃えで表示されます。値 PutButtonsOnLeft を指定すると、その後のボタンは同じ行に左揃えで表示されます。

引数
なし。

戻り値
偶数のエレメントを含む配列。最初のエレメントは、一番上のボタンのラベルを含むストリングです。2 番目のエレメントは、一番上のボタンがクリックされた際のビヘイビアを定義する JavaScript コードのストリングです。残りのエレメントにより、その他のボタンが同様に定義されます。

例
次の commandButtons() の例は、「OK」、「キャンセル」、「ヘルプ」の 3 つのボタンを定義し、ダイアログボックスの右上隅（デフォルトの位置）にそれらのボタンを表示します。

```
function commandButtons(){
return new Array("OK" , "doCommand()" ,
                 "Cancel" , "window.close()" ,
                 "Help" , "showHelp()");
}
```

ボタンの配置と整列はカスタマイズできます。

例
次の commandButtons() の例は、ダイアログボックスの下部にボタンを表示します。返される配列の最初の値が PutButtonsOnBottom の場合、ボタン名の 1 つと共に 2 番目の値に defaultButton を指定できます。このボタンはデフォルトで選択され、Enter キーを押すとこのボタンが使用されます。この例では、OKbutton がデフォルトとして定義されます。

```
function commandButtons(){
return new Array("PutButtonsOnBottom" , "OkButton defaultButton" ,
                 "OK" , "doCommand()" ,
                 "Cancel" , "window.close()" ,
                 "Help" , "showHelp()");
}
```

例
次の例は、PutButtonsOnLeft を使用して「ヘルプ」ボタンを左揃えで表示します。

```
function commandButtons(){
return new Array("PutButtonsOnBottom", "OkButton defaultButton",
                  "OK", "doCommand()",
                  "Cancel", "window.close()",
                  "PutButtonsOnLeft",
                  "Help" , "showHelp()");}
```

isDOMRequired()

説明
この関数では、コマンドを実行するのに有効な DOM が必要かどうかを判断します。この関数から値 true が返された場合、またはこの関数が定義されていない場合、Dreamweaver はコマンドに有効な DOM が必要であるとみなし、ドキュメントのデザインビューとコードビューを同期させてからコマンドを実行します。同期を行うことにより、コードビューのすべての編集内容がデザインビューに反映されます。

引数
なし。

戻り値
コマンドを実行するために有効な DOM が必要であれば true、不要であれば false。

receiveArguments()

説明
この関数は、メニュー項目または dw.runCommand() 関数から渡された引数を処理します。

引数
{arg1}、{arg2}、...{argN}

- arguments 属性が menuitem タグに定義されている場合、この属性の値が引数として receiveArguments() 関数に渡されます。また、dw.runCommand() 関数で引数をコマンドに渡すこともできます。

戻り値
なし。

windowDimensions()

説明
この関数は、パラメータダイアログボックスに特定のサイズを設定します。この関数が定義されていない場合、ウィンドウのサイズは自動的に計算されます。

注意：640 x 480 ピクセル以上のパラメータダイアログボックスを使用する場合に限り、この関数を定義するようにしてください。

引数
platform

- **platform** 引数の値は、ユーザのプラットフォームに応じて、"macintosh" または "windows" になります。

戻り値
"widthInPixels,heightInPixels" という形式のストリング。

返されるサイズはダイアログボックス全体のサイズよりも小さくなります。これは「OK」と「キャンセル」に使用する領域が含まれていないためです。返されたサイズにすべてのオプションが収まらない場合は、スクロールバーが表示されます。

例
次の windowDimensions() 関数の例では、パラメータダイアログボックスのサイズを 648 x 520 ピクセルに設定します。

```
function windowDimensions(){
    return "648,520";
}
```

第 10 章：メニューおよびメニューコマンド

Adobe Dreamweaver では、Dreamweaver の Configuration¥Menus フォルダ内にある menus.xml ファイルで定義された構造からすべてのメニューが作成されます。menus.xml ファイルを編集すれば、メニューコマンドの再配置、名前の変更、および削除ができます。メニューコマンドのキーボードショートカットの追加、変更、削除も実行できます。ただし、ほとんどの場合はキーボードショートカットエディタを使用してキーボードショートカットを編集する方が簡単です（Dreamweaver ヘルプを参照してください）。Dreamweaver メニューの変更内容は、次回に Dreamweaver を起動したとき、または拡張機能をリロードしたときに有効になります。

menus.xml ファイル

menus.xml ファイルには、メニューバー、メニュー、メニューコマンド、セパレータ、ショートカットリストおよびキーボードショートカットの構造化されたリストが格納されています。XML タグはこれらの項目を記述しており、テキストエディタで編集できます。

注意：メニューを変更するときには注意が必要です。Dreamweaver では、XML シンタックスエラーが含まれるメニューまたはメニューコマンドは無視されます。

マルチユーザオペレーティングシステムの場合、Dreamweaver で加えた変更は menus.xml に反映されます。例えば、キーボードショートカットエディタを使用してキーボードショートカットを変更するとします。この場合、ユーザの Configuration フォルダに menus.xml ファイルが作成されます。マルチユーザオペレーティングシステムで menus.xml をカスタマイズするには、ユーザの Configuration フォルダにあるファイルのコピーを編集します。それ以外の場合、Dreamweaver によってコピーがまだ作成されていなければ、ユーザの Configuration フォルダにマスターの menus.xml ファイルをコピーします。詳しくは、76 ページの「マルチユーザ設定フォルダ」を参照してください。

XML エディタで menus.xml を開く場合は、menus.xml ファイル内のアンパサンド（&）に関するエラーメッセージが表示される場合があります。menus.xml の編集は、Dreamweaver 内ではなく、テキストエディタで行います。XML に関する基本情報については、Dreamweaver ヘルプを参照してください。

注意：現在の menus.xml ファイルやその他の Dreamweaver 設定ファイルを修正する場合は、修正する前に必ずそのファイルのバックアップコピーを作成します。バックアップの作成を忘れていた場合でも、menus.xml を Configuration フォルダの menus.bak のコピーに置き換えることができます。

メニューバー（開始と終了の menu bar タグでタグ付けされています）は、個別のメニューまたはメニューセットです。例えば、メインメニューバー、別のサイトパネルメニューバー（Windows でのみ表示され、Macintosh では表示されません）および各コンテキストメニューのメニューバーがあります。それぞれのメニューバーに、メニューが含まれます。1 つのメニューは、1 つのメニュータグに含まれます。各メニューには、メニューコマンドがあり、それぞれが menuitem タグとその属性により記述されています。メニューには、セパレータ（セパレータタグにより記述）やサブメニューを含めることもできます。

メニューコマンドに関連付けられている主要キーボードショートカットだけでなく、Dreamweaver では代替ショートカットやコンテキスト別ショートカットも使用できます。例えば、Ctrl + Y キー（Windows）または Command + Y キー（Macintosh）は、「やり直し」のショートカットです。一方、Ctrl + Shift + Z キーまたは Command + Shift+ Z キーは「やり直し」の代替ショートカットです。メニューコマンドのタグで表すことができない、このような代替ショートカットやコンテキスト別ショートカットは、menus.xml ファイル内のショートカットリストで定義されます。shortcutlist タグで記述された各ショートカットリストにはショートカットが含まれ、それぞれがショートカットタグで記述されています。

次の節では、menus.xml ファイルのタグのシンタックスについて説明します。オプションの属性は、属性リスト内に波カッコ（{}）で囲まれて示されます。波カッコで囲まれていない属性はすべて必須です。

<menubar>

説明
Dreamweaver のメニュー構造のメニューバーに関する情報を指定します。

属性
name、{app}、id、{platform}

- name メニューバーの名前。name は必須属性ですが、値として "" を指定できます。
- app メニューバーを使用できるようにするアプリケーションの名前。現在は使用されていません。
- id メニューバーのメニュー ID。menus.xml ファイル内のメニュー ID は、それぞれ固有である必要があります。
- platform メニューバーを指定したプラットフォーム上でのみ表示することを示します。有効値は、"win" および "mac" です。

コンテンツ
このタグには、menu タグを含める必要があります。

コンテナ
なし。

例
メイン（ドキュメントウィンドウ）メニューバーでは、次の menubar タグを使用します。

```
<menubar name="Main Window" id="DWMainWindow">
<!-- menu tags here -->
</menubar>
```

<menu>

説明
Dreamweaver のメニュー構造に表示されるメニューまたはサブメニューに関する情報を指定します。

属性
name, {app}, id, {platform}, {showIf}

- name メニューバーに表示されるときのメニューの名前。Windows でメニューのアクセスキー（ニーモニック）を設定するには、アクセス文字の前にアンダースコア（_）を使用します。Macintosh では、アンダースコアが自動的に削除されます。
- app メニューを使用できるようにするアプリケーションの名前。現在は使用されていません。
- id メニューのメニュー ID。ファイル内の ID は、それぞれ固有である必要があります。
- platform メニューを所定のプラットフォーム上でのみ表示することを示します。有効値は、"win" および "mac" です。
- showIf Dreamweaver の、指定したイネーブラの値が true の場合にのみ、メニューが表示されるように指定します。指定可能なイネーブラは、_SERVERMODEL_ASP、_SERVERMODEL_ASPNET、_SERVERMODEL_JSP、_SERVERMODEL_CFML（すべてのバージョンの Adobe ColdFusion の場合）、_SERVERMODEL_CFML_UD4（UltraDeveloper バージョン 4 の ColdFusion の場合）、_SERVERMODEL_PHP、_FILE_TEMPLATE、_VIEW_CODE、_VIEW_DESIGN、_VIEW_LAYOUT、_VIEW_EXPANDED_TABLES および _VIEW_STANDARD です。複数のイネーブラを指定できます。この場合、AND を意味するカンマをイネーブラ間に挿入します。"!" で NOT を指定することもできま

す。例えば、メニューを ASP ページのコードビューでのみ表示する場合は、属性を showIf="_VIEW_CODE, _SERVERMODEL_ASP" と指定します。

コンテンツ

このタグには、menuitem タグおよび 1 つまたは複数の separator タグを含めることができます。また、その他の menu タグ（サブメニュー作成に使用）および標準の HTML コメントタグも含めることができます。

コンテナ

このタグは、menubar タグに含める必要があります。

例

```
<menu name="_File" id="DWMenu_File">
    <!-- menuitem, separator, menu, and comment tags here -->
</menu>
```

<menuitem>

説明

Dreamweaver メニューのメニューコマンドを定義します。

属性

name、id、{app}、{key}、{platform}、{enabled}、{arguments}、{command}、{file}、{checked}、{dynamic}、{isdomrequired}、{showIf}

- name メニューに表示されるメニューコマンド名。アンダースコアは、次の文字がそのコマンドのアクセスキー（ニーモニック）であることを示します（Windows のみ）。
- id Dreamweaver で項目を識別するために使用されます。この ID は、メニュー構造全体で固有である必要があります。menus.xml に新しいメニューコマンドを追加する場合は、自社名やその他固有のストリングを各メニューコマンドの ID の接頭辞として使用し、一意性が損なわれないようにします。
- app メニューコマンドを使用できるようにするアプリケーションの名前。現在は使用されていません。
- key コマンドのキーボードショートカット（存在する場合）。次のストリングを使用して、修飾キーを指定します。
- Cmd は、Ctrl キー（Windows）または Command キー（Macintosh）を指定します。
- Alt および Opt は両方とも、Alt キー（Windows）または Option キー（Macintosh）を指定します。
- Shift は、両方のプラットフォームで Shift キーを指定します。
- Ctrl は、両方のプラットフォームで Ctrl キーまたは Control キーを指定します。
- 指定するショートカットに複数の修飾キーを使用する場合は、プラス（+）記号で修飾キーを分割します。例えば、Cmd+Opt+5（key 属性）は、ユーザが Ctrl + Alt + 5 キー（Windows）または Command + Option + 5 キー（Macintosh）を押したときに、メニューコマンドが実行されることを意味します。
- 特殊キーは、F1 から F12、PgDn、PgUp、Home、End、Ins、Del、Tab、Esc、BkSp および Space のように、名前で指定されます。修飾キーは、特殊キーにも適用できます。
- platform 項目が表示されるプラットフォームを示します。有効値は、Windows を意味する "win" または Macintosh を意味する "mac" です。platform 属性を指定しないと、メニューコマンドが両方のプラットフォームで表示されます。メニューコマンドの動作をプラットフォームごとに変える場合は、同じ名前を持つ（ただし、ID は異なる）2 つのメニューコマンドを指定します。つまり、1 つで platform="win" と設定し、もう 1 つで platform="mac" と設定します。

- enabled メニューコマンドが現在有効かどうかを決定する JavaScript コード（通常は JavaScript の関数呼び出し）を指定します。関数によって、値 false が返されると、メニューコマンドが淡色表示になります。デフォルト値は "true" ですが、値が "true" の場合でも、明確にするために常に値を指定することをお勧めします。
- argumentsfile 属性で指定した JavaScript ファイル内のコードに Dreamweaver から渡される引数を指定します。属性の値を区切るために使用する二重引用符（"）内では、引数は一重引用符（'）で囲みます。
- command ユーザがメニューからこの項目を選択すると実行される JavaScript 式を指定します。複雑な JavaScript コードの場合は、代わりに JavaScript ファイル（file 属性で指定）を使用します。メニューコマンドごとに、file または command を指定する必要があります。
- file メニューコマンドをコントロールする JavaScript を含む HTML ファイルの名前。Configuration フォルダを基準にしたファイルの相対パスを指定します。例えば、ヘルプ／ようこそメニューコマンドでは、file="Commands/Welcome.htm" と指定します。file 属性は、command、enabled および checked 属性より優先されます。メニューコマンドごとに、file または command を指定する必要があります。ヒストリパネルを使用したコマンドファイル作成について詳しくは、Dreamweaver ヘルプを参照してください。独自の JavaScript コマンド記述について詳しくは、130 ページの「コマンド」を参照してください。
- checked メニューでメニューコマンドの横にチェックマークが付いているかどうかを示す JavaScript 式。この式で true と評価された場合は、項目がチェックマーク付きで表示されます。
- dynamic この属性がある場合は、メニューコマンドが HTML ファイルによって動的に決定されることを示します。このファイルには、メニューコマンドのテキストと状態を設定する JavaScript コードが含まれます。タグを dynamic として指定した場合は、file 属性も指定する必要があります。
- isdomrequired このメニューコマンドのコードを実行する前にデザインビューとコードビューを同期させるかどうかを示します。有効値は、"true"（デフォルト）と "false" です。この属性を "false" に設定すると、このメニューコマンドによって行われるファイルの変更には、Dreamweaver DOM が使用されません。DOM について詳しくは、94 ページの「Dreamweaver ドキュメントオブジェクトモデル」を参照してください。
- showIf Dreamweaver の、指定したイネーブラの値が true の場合にのみ、メニュー項目が表示されるように指定します。指定可能なイネーブラは、_SERVERMODEL_ASP、_SERVERMODEL_ASPNET、_SERVERMODEL_JSP、_SERVERMODEL_CFML（すべてのバージョンの Adobe ColdFusion の場合）、_SERVERMODEL_CFML_UD4（UltraDeveloper バージョン 4 の ColdFusion の場合）、_SERVERMODEL_PHP、_FILE_TEMPLATE、_VIEW_CODE、_VIEW_DESIGN、_VIEW_LAYOUT、_VIEW_EXPANDED_TABLES および _VIEW_STANDARD です。複数のイネーブラを指定できます。この場合、AND を意味するカンマをイネーブラ間に挿入します。"!" で NOT を指定することもできます。例えば、メニューコマンドを ColdFusion ページではなく、コードビューで表示する場合は、属性を showIf="_VIEW_CODE, !_SERVERMODEL_CFML" と指定します。

コンテンツ
なし（空のタグ）。

コンテナ
このタグは、menu タグに含める必要があります。

例

```
<menuitem name="_New" key="Cmd+N" enabled="true" command="dw.createDocument()" id="DWMenu_File_New" />
```

<separator>

説明
セパレータをメニュー内の対応する場所に表示する必要があることを示します。

属性
{app}

app セパレータを表示するアプリケーションの名前。現在は使用されていません。

コンテンツ
なし（空のタグ）。

コンテナ
このタグは、menu タグに含める必要があります。

例
```
<separator />
```

<shortcutlist>

説明
menus.xml ファイル内のショートカットリストを指定します。

属性
{app}、id、{platform}

- app ショートカットリストを使用できるようにするアプリケーションの名前。現在は使用されていません。
- id ショートカットリストの ID。ショートカットが関連付けられている Dreamweaver 内のメニューバー（またはコンテキストメニュー）のメニュー ID と同じである必要があります。有効値は、"DWMainWindow"、"DWMainSite"、"DWTimelineContext" および "DWHTMLContext" です。
- platform ショートカットリストを所定のプラットフォーム上でのみ表示することを示します。有効値は、"win" および "mac" です。

コンテンツ
このタグには、shortcut タグを含めることができます。また、HTML コメントタグと同じシンタックスを使用するコメントタグを含めることもできます。

コンテナ
なし。

例
```
<shortcutlist id="DWMainWindow">
<!-- shortcut and comment tags here -->
</shortcutlist>
```

<shortcut>

説明
menus.xml ファイル内のキーボードショートカットを指定します。

属性

key、{app}、{platform}、{file}、{arguments}、{command}、id、{name}

- key キーボードショートカットを有効にするキーの組み合わせ。シンタックスについて詳しくは、141 ページの「<menuitem>」を参照してください。
- app ショートカットを使用できるようにするアプリケーションの名前。現在は使用されていません。
- platform ショートカットを、ここで指定したプラットフォーム上でのみ機能させます。有効値は、"win" および "mac" です。この属性を指定しない場合は、両方のプラットフォームでショートカットが機能します。
- file キーボードショートカットを使用したときに Dreamweaver で実行される JavaScript コードを含むファイルへのパス。file 属性は command 属性よりも優先されます。ショートカットごとに、file または command を指定する必要があります。
- argumentsfile 属性で指定した JavaScript ファイル内のコードに Dreamweaver から渡される引数を指定します。属性の値を区切るために使用する二重引用符（"）内では、引数は一重引用符（'）で囲みます。
- command キーボードショートカットを使用したときに Dreamweaver で実行される JavaScript コード。ショートカットごとに、file または command を指定します。
- id ショートカットの固有の識別子。
- name キーボードショートカットによって実行されるコマンドの名前。メニューコマンド名のスタイルで示します。例えば、F12 ショートカットの name 属性は、「プライマリブラウザでプレビュー」です。

コンテンツ

なし（空のタグ）。

コンテナ

このタグは、shortcutlist タグに含める必要があります。

例

```
<shortcut key="Cmd+Shift+Z" file="Menus/MM/Edit_Clipboard.htm"
arguments="'redo'" id="DWShortcuts_Edit_Redo" />
```

<tool>

説明

1 つのツールを表します。menus.xml ファイル内にサブタグとしてツールのすべてのショートカットを含みます。

属性

{name}、id

- name ツール名のローカライズ版。
- id ショートカットを適用するツールを特定する内部ツール識別子。

コンテンツ

このタグには、activate、override または action タグを含めることができます。

コンテナ

このタグは、menu タグに含める必要があります。

例

```
<tool name="Hand tool" id="com.Macromedia.dreamweaver.tools.hand">
    <!-- tool tags here -->
</tool>
```

<action>

説明

キーの組み合わせと、ツールがアクティブな場合にそのキーの組み合わせが押されたときに実行する JavaScript を含んでいます。

属性

{name}、key、command、id

- name アクションのローカライズ版。
- key アクションの実行に使用するキーの組み合わせ。シンタックスについて詳しくは、141 ページの「<menuitem>」を参照してください。
- command 実行する JavaScript ステートメント。この属性は command 属性（143 ページの「<shortcut>」の）と同じフォーマットです。
- id アクションの参照に使用する固有の ID。

コンテンツ

なし（空のタグ）。

コンテナ

このタグは、tool タグに含める必要があります。

例

```
<action name="Set magnification to 50%" key="5" command="dw.activeViewScale = 0.50" id ="DWTools_Zoom_50" />
```

<activate>

説明

ツールをアクティブにするキーの組み合わせを含んでいます。

属性

{name}、key、id

- name アクションのローカライズ版。
- key ツールをアクティブにするキーの組み合わせ。シンタックスについて詳しくは、141 ページの「<menuitem>」を参照してください。
- id アクションの参照に使用する固有の ID。

コンテンツ

なし（空のタグ）。

コンテナ
このタグは、tool タグに含める必要があります。

例
```
<activate name="Switch to Hand tool" key="H" id="DWTools_Hand_Active1" />
```

<override>

説明
ツールを一時的にアクティブにするキーの組み合わせを含んでいます。別のモーダルツールを使用しているときに、ユーザは、このキーを押したままにしてこのツールに切り替えることができます。

属性
{name}、key、id

- name アクションのローカライズ版。
- key ツールをすぐにアクティブにするキーの組み合わせ。シンタックスについて詳しくは、141 ページの「<menuitem>」を参照してください。
- id アクションの参照に使用する固有の ID。

コンテンツ
なし（空のタグ）。

コンテナ
このタグは、tool タグに含める必要があります。

例
```
<override name="Quick switch to Hand tool" key="Space" id="DWTools_Hand_Override" />
```

メニューおよびメニューコマンドの変更

menus.xml ファイルを編集することによって、メニュー内またはメニュー間でのメニューコマンドの移動、セパレータのメニューへの追加とメニューからの削除、およびメニューバー内またはメニューバー間でのメニューの移動が行えます。

他のメニューと同じ手順を使用して、コンテキストメニューへの項目の移動や、コンテキストメニューからの項目の移動が行えます。

詳しくは、139 ページの「menus.xml ファイル」を参照してください。

メニューコマンドを移動する

1. Dreamweaver を終了します。
2. menus.xml ファイルのバックアップコピーを作成します。
3. BBEdit、Macromedia® HomeSite、Wordpad などのテキストエディタで menus.xml を開きます（Dreamweaver では開かないでください）。
4. menuitem タグ全体（最初の <menuitem から最後の /> まで）をカットします。

5 メニューコマンドの新しい位置に挿入ポイントを配置します（挿入ポイントは必ず、menu タグと、対応する /menu タグの間に配置してください）。

6 新しい位置にメニューコマンドをペーストします。

メニューコマンドの移動時にサブメニューを作成する

1 メニュー内（menu タグと、対応する /menu タグの間の任意の位置）に挿入ポイントを配置します。

2 メニュー内に新しい menu タグと /menu タグのペアを挿入します。

3 新しいサブメニューに新しいメニューコマンドを追加します。

2 つのメニューコマンドの間にセパレータを挿入する

- separator/ タグを 2 つの menuitem タグの間に配置します。

既存のセパレータを削除する

- 対応する separator/ 行を削除します。

メニューを移動する

1 Dreamweaver を終了します。

2 menus.xml ファイルのバックアップコピーを作成します。

3 BBEdit、HomeSite、Wordpad などのテキストエディタで menus.xml を開きます（Dreamweaver では開かないでください）。

4 開始タグの menu から終了タグの /menu まで、メニュー全体とその内容をカットします。

5 メニューの新しい位置に挿入ポイントを配置します （挿入ポイントは必ず、menubar タグと、対応する /menubar タグの間に配置してください）。

6 新しい位置にメニューをペーストします。

メニューコマンドまたはメニューの名前の変更

menus.xml ファイルを編集すれば、メニューコマンドやメニューの名前を簡単に変更できます。

1 Dreamweaver を終了します。

2 menus.xml ファイルのバックアップコピーを作成します。

3 HomeSite、BBEdit、Wordpad などのテキストエディタで menus.xml を開きます（Dreamweaver では開かないでください）。

4 メニューコマンドの名前を変更する場合は、該当する menuitem タグを見つけて、その name 属性の値を変更します。メニューの名前を変更する場合は、該当する menu タグを見つけて、その name 属性の値を変更します。いずれの場合も、id 属性は変更しないでください。

5 menus.xml を保存して、閉じます。次に、再度 Dreamweaver を起動して変更部分を確認します。

キーボードショートカットの変更

デフォルトのキーボードショートカットでは不便な場合、既存のショートカットの変更や削除、または新しいショートカットの追加が可能です。キーボードショートカットエディタを使用するのが、一番簡単な方法です（詳しくは、Dreamweaver ヘルプを参照してください）。必要に応じて、menus.xml で直接キーボードショートカットを修正することもできます。ただし、キーボードショートカットエディタを使用する場合と比べて、menus.xml ではショートカットを入力する際に間違いが起こりやすくなります。

1. Dreamweaver を終了します。
2. menus.xml ファイルのバックアップコピーを作成します。
3. BBEdit、HomeSite、Wordpad などのテキストエディタで menus.xml を開きます（Dreamweaver では開かないでください）。
4. Dreamweaver サポートセンター（http://www.adobe.com/go/dreamweaver_support から入手できるキーボードショートカット表を参照して、使用されていないショートカットまたは再割り当てを行うショートカットを見つけます。

   キーボードショートカットを再割り当てする場合は、後から参照できるように、印刷されたコピーの表でショートカットを変更します。
5. キーボードショートカットを再割り当てする場合は、該当するショートカットが割り当てられているメニューコマンドを見つけ、そのメニューコマンドから key="shortcut" 属性を削除します。
6. キーボードショートカットに割り当てまたは再割り当てを行うメニューコマンドを見つけます。
7. そのメニューコマンドに既にキーボードショートカットが割り当てられている場合は、その行の key 属性を見つけます。まだ、ショートカットが割り当てられていない場合は、key="" を menuitem タグ内で属性と属性の間の任意の場所に追加します。
8. key 属性の二重引用符（"）の間に、新しいキーボードショートカットを入力します。

   キーを組み合わせる場合は、キー同士の間にプラス（+）記号を使用します。モディファイヤーについて詳しくは、menuitem タグの説明（「141 ページの「<menuitem>」」）を参照してください。

   割り当てたキーボードショートカットを他のメニューコマンドでも使用していて、それを削除しない場合、割り当てたショートカットは menus.xml で一番先に出現するメニューコマンドにのみ適用されます。

   注意：Windows 専用メニューコマンドと Macintosh 専用メニューコマンドに対しては、同じキーボードショートカットを使用できます。
9. 新しいショートカットを、キーボードショートカット表の該当する位置に記入します。

ポップアップメニューおよびコンテキストメニュー

Dreamweaver では、多くのパネルやダイアログボックスで、ポップアップメニューおよびコンテキストメニューを利用しています。menus.xml ファイルで定義されるコンテキストメニューと、その他の XML ファイルで定義されるコンテキストメニューがあります。ポップアップメニューおよびコンテキストメニューで項目の追加、削除または修正を行えますが、通常は拡張機能を作成して、そういった変更を行うことをお勧めします。

Dreamweaver では、次のポップアップメニューおよびコンテキストメニューは、menus.xml と同じタグを使用して XML ファイルで指定されます。

- バインディングパネルの + ポップアップメニューに一覧表示されるデータソースは、DataSources フォルダのサブフォルダ内にある DataSources.xml ファイルで指定されます。
- サーバビヘイビアパネルの + ポップアップメニューに一覧表示されるサーバビヘイビアは、ServerBehaviors フォルダのサブフォルダ内にある ServerBehaviors.xml ファイルで指定されます。
- フォーマットリストの編集ダイアログボックスの + ポップアップメニューに一覧表示されるサーバフォーマットは、ServerFormats フォルダのサブフォルダにある ServerFormats.xml ファイルで指定されます。

- バインディングパネルでバインディングに使用するフォーマットポップアップメニューの項目は、ServerFormats フォルダのサブフォルダにある Formats.xml ファイルで指定されます。フォーマットの追加ダイアログボックスを使用して、Dreamweaver 内からこのメニューにエントリを追加できます。
- タグライブラリエディタダイアログボックスのメニューコマンドは、TagLibraries¥TagImporters¥TagImporters.xml ファイルで指定されます。
- サーバビヘイビアビルダーの一部である、新規サーバビヘイビアダイアログボックスでパラメータに使用するメニューコマンドは、Shared¥Controls¥String Menu¥Controls.xml で指定されます。
- ColdFusion コンポーネントに関連付けられたコンテキストメニューの項目は、Components¥ColdFusion¥CFCs¥CFCsMenus.xml で指定されます。
- ColdFusion データソースに関連付けられたコンテキストメニューの項目は、Components¥ColdFusion¥DataSources¥DataSourcesMenus.xml で指定されます。
- 各種サーバコンポーネントに関連付けられたコンテキストメニューの項目は、Components フォルダのサブフォルダにある XML ファイルで指定されます。

メニューコマンド

メニューコマンドを使用して、より柔軟で動的なメニューを作成することができます。メニューコマンドは、現在のドキュメント、開いている他のドキュメント、またはローカルハードドライブ上の任意の HTML ドキュメントに対して、ほぼすべての編集を実行できます。

メニューコマンドは、menus.xml ファイル内の file 属性（menuitem タグの）で参照される HTML ファイルです。メニューコマンドファイルの body セクションには、コマンドのオプションを入力するための HTML フォームを含めることができます。例えば、テーブルのソート方法やソートの基準とする列などです。メニューコマンドファイルの head セクションには、JavaScript 関数が含まれます。この関数は、body セクションのフォーム入力を処理し、ユーザのドキュメントへの編集を制御します。

メニューコマンドは Dreamweaver アプリケーションフォルダ内の Configuration¥Menus フォルダに保存されています。

注意：Mac OS X 10.3 のコマンドは、youUser/Library/Application Support/Adobe/Dreamweaver 9/Commands/yourcommandname.html に保存されています。

次の表にメニューコマンドを作成するために使用するファイルの一覧を示します。

パス	ファイル	説明
Configuration¥Menus¥	menus.xml	メニューバー、メニュー、メニューコマンド、セパレータ、ショートカットリストおよびキーボードショートカットの構造化されたリストが格納されています。新しいメニューおよびメニューコマンドを追加するには、このファイルを変更します。
Configuration¥Menus¥	**commandname.**htm	メニューコマンドに必要な関数が含まれています。

注意：カスタムメニューコマンドを Dreamweaver に追加する場合は、それらを Menus フォルダの最上位レベルに追加するか、サブフォルダを作成します。MM フォルダは、Dreamweaver の既定のメニューコマンド用に予約されています。

コマンドメニューの修正

menus.xml ファイルを編集せずに、コマンドメニューに特定の種類のコマンドを追加したり、追加したコマンドの名前を変更したりできます。menus.xml について詳しくは、146 ページの「メニューおよびメニューコマンドの変更」を参照してください。

注意：Dreamweaver では、「コマンド」という用語に 2 つの意味があります。厳密に言うと、コマンドは特定の種類の拡張機能です。ただし、状況によっては、「コマンド」は「メニュー項目」と同じ意味で使用されます。この場合、機能や実装方法に関係なく、Dreamweaver のメニューに表示されるあらゆる項目を意味します。

コマンドメニューに自動的に配置される新規コマンドを作成するには、ヒストリパネルを使用します。または、Extension Manager を使用して、コマンドなど、新しい拡張機能をインストールすることもできます。詳しくは、Dreamweaver ヘルプを参照してください。

コマンドメニューの項目を並べ替えるには、またはメニュー間で項目を移動するには、menus.xml ファイルを編集する必要があります。

作成したコマンドの名前を変更する

1 コマンド／コマンドリストの編集を選択します。

ダイアログボックスが表示され、名前を変更できるすべてのコマンドが一覧表示されます。デフォルトのコマンドメニューに含まれるコマンドは、この一覧には表示されず、ここで説明する方法で編集することはできません。

2 名前を変更するコマンドを選択します。

3 コマンドの新しい名前を入力します。

4 「閉じる」をクリックします。

コマンドメニューでコマンドの名前が変更されます。

作成したコマンドを削除する

1 コマンド／コマンドリストの編集を選択します。

ダイアログボックスが表示され、削除できるすべてのコマンドが一覧表示されます。デフォルトのコマンドメニューに含まれるコマンドは、この一覧には表示されず、ここで説明する方法で削除することはできません。

2 削除するコマンドを選択します。

3 「削除」をクリックし、コマンドの削除を確認します。

コマンドが削除されます。コマンドを削除すると、単にメニューからメニューコマンドが削除されるだけでなく、コマンドのコードを含むファイルも削除されます。この方法を使用する前に、本当にそのコマンドを削除してもよいか、必ず確認してください。ファイルを削除せずにコマンドメニューからコマンドを削除するには、Configuration¥Commands 内のそのファイルを別のフォルダに移動します。

4 「閉じる」をクリックします。

メニューコマンドの動作

ユーザがメニューコマンドを含むメニューをクリックすると、次のようなイベントが発生します。

1 メニューの menuitem タグのいずれかに dynamic 属性がある場合、Dreamweaver は関連するメニューコマンドファイルから getDynamicContent() 関数を呼び出して、メニューに含める項目を取り込みます。

2 Dreamweaver により、メニューで参照される各メニューコマンドファイルの canAcceptCommand() 関数が呼び出され、そのコマンドが選択項目に対して適切であるかどうかがチェックされます。

- canAcceptCommand() 関数から値 false が返されると、そのメニュー項目は淡色表示になります。
- canAcceptCommand() 関数から値 true が返された場合、またはこの関数が定義されていない場合は、Dreamweaver により isCommandChecked() 関数が呼び出され、メニュー項目の横にチェックマークを表示するかどうかが決まります。isCommandChecked() 関数が定義されていない場合、チェックマークは表示されません。

3 Dreamweaver により setMenuText() 関数が呼び出され、メニューに表示するテキストが決定されます。

setMenuText() 関数が定義されていない場合、Dreamweaver により menuitem タグで指定されているテキストが使用されます。

4 ユーザがメニューから項目を選択します。

5 選択されたメニューコマンドファイルに receiveArguments() 関数が定義されている場合、Dreamweaver によりこの関数が呼び出され、メニュー項目から渡された引数がコマンドにより処理されます。

注意：これが動的メニュー項目の場合、メニュー項目の ID が唯一の引数として渡されます。

6 commandButtons() 関数が定義されている場合、この関数が呼び出されて、オプションダイアログボックスの右側に表示されるボタンと、ユーザがそのボタンをクリックしたときに実行されるコードが決定されます。

7 Dreamweaver はメニューコマンドファイルの form タグを検索します。

- フォームが存在する場合、windowDimensions() 関数が呼び出されて、ファイルの BODY エレメントを表示するオプションダイアログボックスのサイズが決定されます。
- windowDimensions() 関数が定義されていない場合、ダイアログボックスのサイズは自動的に決定されます。

8 メニューコマンドファイルの body タグに onLoad ハンドラがある場合は、ダイアログボックスが表示されるかどうかにかかわらず、Dreamweaver によって関連するコードが実行されます。ダイアログボックスが表示されない場合、この後のイベントは発生しません。

9 ユーザがダイアログボックスのオプションを選択します。ユーザがフィールドに進むたびに、フィールドに関連付けられたイベントハンドラが実行されます。

10 ユーザが commandButtons() 関数によって定義されているボタンのいずれかをクリックします。

11 Dreamweaver により、クリックされたボタンに関連付けられたコードが実行されます。

12 メニューコマンドファイルのスクリプトのいずれかにより window.close() 関数が呼び出されると、ダイアログボックスが閉じます。

簡単なメニューコマンドの例

この簡単なメニューコマンドの例では、「取り消し」と「やり直し」のメニューコマンドがどのように機能するかを示します。「取り消し」メニューコマンドは、ユーザの編集操作の結果を取り消します。また、「やり直し」メニューコマンドは、取り消し操作を取り消し、ユーザが最後に行った編集操作の結果を復元します。

この例を実装するには、メニューコマンドを作成し、JavaScript コードを記述して、コマンドファイルを Menu フォルダに配置します。

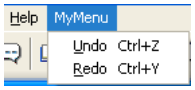
メニューコマンドの作成

「取り消し」および「やり直し」メニューコマンドを含む MyMenu というメニューを作成するには、次の HTML メニュータグを追加します。menus.xml の最後の </menu> 終了タグの後にタグを追加します。

```
<menu name="MyMenu" id="MyMenu_Edit">
<menuitem name="MyUndo" key="Cmd+Z" file="Menus/MyMenu.htm" arguments="'undo'" id="MyMenu_Edit_Undo" />
<menuitem name="MyRedo" key="Cmd+Y" file="Menus/MyMenu.htm" arguments="'redo'" id="MyMenu_Edit_Redo" />
</menu>
```

key 属性は、ユーザがメニューコマンドを読み込む際に入力するキーボードショートカットを定義します。file 属性は、Dreamweaver によりコマンドが読み込まれたときに実行されるメニューコマンドファイルの名前を指定します。arguments 属性の値は、Dreamweaver によって receiveArguments() 関数が呼び出されたときに渡される引数を定義します。

次の図に、これらのメニューコマンドを示します。



メニューの JavaScript コード

ユーザが MyMenu メニューの「取り消し」または「やり直し」のいずれかを選択すると、file 属性（menuitem タグの）で指定した MyMenu.htm コマンドファイルが Dreamweaver によって呼び出されます。Dreamweaver の Configuration¥Menus フォルダに MyMenu.htm コマンドファイルを作成し、canAcceptCommand()、receiveArguments() および setMenuText() という 3 つのメニューコマンド API 関数を追加して、「取り消し」および「やり直し」メニュー項目に関連付けられたロジックを実装します。次の節では、これらの関数について説明します。

canAcceptCommand()

Dreamweaver により、MyMenu メニューのメニュー項目ごとに canAcceptCommand() 関数が呼び出され、そのメニュー項目を有効または無効のいずれにするかが決定されます。MyMenu.htm ファイルでは、canAcceptCommand() 関数が arguments[0] の値をチェックし、Dreamweaver により「やり直し」メニュー項目または「取り消し」メニュー項目が処理中であるかどうか判別されます。引数が "undo" である場合は、canAcceptCommand() 関数によりイネーブラ関数の dw.canUndo() が呼び出され、true または false の戻り値を返します。同様に、引数が "redo" である場合は、canAcceptCommand() 関数によりイネーブラ関数の dw.canRedo() が呼び出され、その値を Dreamweaver に返します。canAcceptCommand() 関数から false が返されると、関数を呼び出す対象となったメニュー項目が淡色表示になります。次の例では、canAcceptCommand() 関数のコードを示しています。

```
function canAcceptCommand()
{
    var selarray;
    if (arguments.length != 1) return false;
    var bResult = false;

    var whatToDo = arguments[0];
    if (whatToDo == "undo")
    {
        bResult = dw.canUndo();
    }
    else if (whatToDo == "redo")
    {
        bResult = dw.canRedo();
    }
    return bResult;
}
```

receiveArguments()

receiveArguments() 関数が呼び出され、menuitem タグに対して定義した引数が処理されます。「取り消し」および「やり直し」メニュー項目について、receiveArguments() 関数は、dw.undo() または dw.redo() を呼び出します。これは、引数 arguments[0] の値が "undo" または "redo" のどちらであるかによって決まります。dw.undo() 関数は、フォーカスが置かれているドキュメントウィンドウ、ダイアログボックスまたはパネルでユーザが実行した前の手順を取り消します。dw.redo() 関数は、最後に取り消した操作をやり直します。

receiveArguments() 関数のコード例を次に示します。

```
function receiveArguments()
{
    if (arguments.length != 1) return;

    var whatToDo = arguments[0];
    if (whatToDo == "undo")
    {
        dw.undo();
    }
    else if (whatToDo == "redo")
    {
        dw.redo();
    }
}
```

このコマンドでは、receiveArguments() 関数によって引数が処理され、コマンドが実行されます。複雑なメニューコマンドでは、別の関数を呼び出してコマンドを実行することもあります。例えば、以下のコードは最初の引数が "foo" かどうかをチェックし、そうである場合は、doOperationX() 関数を呼び出して 2 番目の引数を渡します。最初の引数が "bar" の場合は、doOperationY() 関数を呼び出して 2 番目の引数を渡します。doOperationX() 関数または doOperationY() 関数によって、コマンドが実行されます。

```
function receiveArguments(){
    if (arguments.length != 2) return;

    var whatToDo = arguments[0];

    if (whatToDo == "foo"){
        doOperationX(arguments[1]);
    }else if (whatToDo == "bar"){
        doOperationX(arguments[1]);
    }
}
```

setMenuText()

setMenuText() 関数の呼び出しにより、メニュー項目として表示されるテキストが決定されます。setMenuText() 関数を定義しない場合は、name 属性（menuitem タグの）で指定したテキストが使用されます。

setMenuText() 関数は、Dreamweaver から渡される引数の値 arguments[0] をチェックします。引数の値が "undo" の場合は、dw.getUndoText() 関数が呼び出されます。"redo" の場合は、dw.getRedoText() 関数が呼び出されます。dw.getUndoText() 関数は、取り消す操作を指定するテキストを返します。例えば、ユーザが複数のやり直し操作を実行する場合、dw.getUndoText() は「Undo Edit Source」というメニューテキストを返します。同様に、dw.getRedoText() 関数は、やり直す操作を指定するテキストを返します。ユーザが複数の取り消し操作を実行する場合は、dw.RedoText() 関数が「Redo Edit Source」というメニューテキストを返します。

setMenuText() 関数のコード例を次に示します。

```
function setMenuText()
{
    if (arguments.length != 1) return "";

    var whatToDo = arguments[0];
    if (whatToDo == "undo")
        return dw.getUndoText();
    else if (whatToDo == "redo")
        return dw.getRedoText();
    else return "";
}
```

Menu フォルダへのコマンドファイルの配置

メニューに「取り消し」および「やり直し」メニュー項目を実装するには、MyMenu.htm コマンドファイルを Dreamweaver の Configuration¥Menus フォルダまたは作成したサブフォルダに保存する必要があります。ファイルの場所は、menuitem タグで指定した場所と一致する必要があります。このコマンドファイルを使用できる状態にするには、Dreamweaver を再起動するか、拡張機能をリロードします。拡張機能をリロードする方法について詳しくは、77 ページの「拡張機能のリロード」を参照してください。

メニューコマンドを実行するには、メニュー項目が有効な状態のときにこれを選択します。Dreamweaver では、「150 ページの「メニューコマンドの動作」」で説明されているように、コマンドファイルの関数が呼び出されます。

動的メニューの例

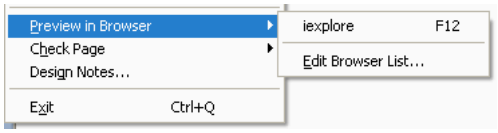
この例では、使用可能なブラウザのリストを表示する、Dreamweaver のブラウザでプレビューサブメニューを実装します。また、この実装例では、現在のファイルまたはサイトパネルで選択したファイルをユーザが指定したブラウザで開きます。この動的メニューを実装するには、動的メニュー項目を作成し、JavaScript コードを記述します。

動的メニュー項目の作成

menus.xml ファイル内の次のメニュータグで、ファイルメニューのブラウザでプレビューサブメニューを定義します。

```
<menu name="_Preview in Browser" id="DWMenu_File_PIB">
    <menuitem dynamic name="No Browsers Selected"
        file="Menus/MM/PIB_Dynamic.htm" arguments="'No Browsers'"
        id="DWMenu_File_PIB_Default" />
    <separator />
    <menuitem name="_Edit Browser List..." enabled="true"
        command="dw.editBrowserList()" id="DWMenu_File_PIB_EditBrowserList" />
    </menu>
```

最初の menuitem タグで、「指定ブラウザなし」というデフォルトのメニュー項目が定義されます。このメニュー項目は、環境設定ダイアログボックスの「ブラウザでプレビュー」でブラウザを指定しなかった場合に表示されます。ただし、Microsoft Internet Explorer ブラウザを指定した場合は、次の図のようにサブメニューが表示されます。



最初のメニュー項目の name 属性により、コマンドファイルの PIB_Dynamic.htm が指定されます。このファイルには、次の行が含まれます。

```
<SCRIPT LANGUAGE="javascript" SRC="PIB_Dynamic.js"></SCRIPT>
```

script タグには、PIB_Dynamic.js ファイルの JavaScript コードが含まれ、これによりブラウザでプレビューサブメニューに関連する JavaScript コードが指定されます。このコードは直接 PIB_Dynamic.htm ファイルに保存することもできますが、このコードを別のファイルに保存することにより、複数のコマンドで同じコードを使用できるようになります。

動的メニュー項目の JavaScript コード

最初の menuitem タグに dynamic 属性があるため、次の例に示すように、PIB_Dynamic.js ファイルの getDynamicContent() 関数が呼び出されます。

```
function getDynamicContent(itemID)
{
    var browsers = null;
    var PIB = null;
    var i;
    var j=0;
    browsers = new Array();
    PIB = dw.getBrowserList();

    for (i=0; i<PIB.length; i=i+2)
    {
        browsers[j] = new String(PIB[i]);

        if (dw.getPrimaryBrowser() == PIB[i+1])
            browsers[j] += "\tF12";
        else if (dw.getSecondaryBrowser() == PIB[i+1])
            browsers[j] += "\tCmd+F12";

        browsers[j] += ";id='"+escQuotes(PIB[i])+"'";

        if (itemID == "DWPopup_PIB_Default")
            browsers[j] = MENU_strPreviewIn + browsers[j];

        j = j+1;
    }
    return browsers;
}
```

getDynamicContent() 関数は、dw.getBrowserList() 関数を呼び出して、環境設定の「ブラウザでプレビュー」セクションで指定されたブラウザ名の配列を取得します。この配列には、各ブラウザの名前と実行ファイルへのパスが含まれます。次に、配列 (i=0; i<PIB.length; i=i+2) 内の項目ごとに、getDynamicContents() 関数が、ブラウザの名前 (PIB[i]) を browsers (browsers[j] = new String(PIB[i]);) という 2 番目の配列に移動します。ブラウザがプライマリブラウザまたはセカンダリブラウザとして指定されている場合は、この関数が、ブラウザを呼び出すキーボードショートカットキーの名前を追加します。次にストリング ";id=" を追加し、その後に一重引用符で囲んだブラウザの名前を追加します（;id='iexplore' など）。itemID 引数が "DWPopup_PIB_Default" の場合、関数が接頭辞として配列項目に Preview in というストリングを付けます。getDynamicContent() 関数は、環境設定ダイアログボックスに一覧表示される各ブラウザのエントリの作成後、Dreamweaver に配列 browsers を返します。ブラウザが選択されていない場合は、関数が null という値を返し、メニューには「指定ブラウザなし」と表示されます。

canAcceptCommand()

次に、canAcceptCommand() 関数が、menuitem タグ（これらのタグは file 属性があるコマンドファイルを参照）ごとに呼び出されます。canAcceptCommand() 関数が戻り値として false を返すと、そのメニュー項目は淡色表示されます。canAcceptCommand() 関数が true を返すと、メニュー上でその項目が有効になります。この関数が true を返した場合、またはこの関数が定義されていない場合は、isCommandChecked() 関数が呼び出され、メニュー項目の横にチェックマークを表示するかどうかが決まります。isCommandChecked() 関数が定義されていない場合、チェックマークは表示されません。

```
function canAcceptCommand()
{
    var PIB = dw.getBrowserList();

    if (arguments[0] == 'primary' || arguments[0] == 'secondary')
        return havePreviewTarget();

    return havePreviewTarget() && (PIB.length > 0);
}
```

PIB_Dynamic.js ファイル内の canAcceptCommand() 関数が、環境設定ダイアログボックスで作成されたブラウザリストを再度取得します。次に、この関数によって最初の引数（arguments[0]）がプライマリまたはセカンダリであるかどうかがチェックされます。プライマリまたはセカンダリである場合は、havePreviewTarget() 関数が返した値が返されます。プライマリまたはセカンダリではない場合は、havePreviewTarget() 関数の呼び出しをテストし、さらに、指定されているブラウザがあるかどうかをテストします（PIB.length > 0）。「両方の」テストの結果が true である場合は、関数が値として true を返します。テストのうちいずれか、または両方のテストの結果が false である場合は、関数が値として false を返します。

havePreviewTarget()

havePreviewTarget() 関数は、ユーザ定義の関数であり、ブラウザに表示する有効なターゲットが Dreamweaver にある場合は、値として true を返します。有効なターゲットとは、サイトパネル内のドキュメントまたは選択されたファイルグループです。havePreviewTarget() 関数のコード例を次に示します。

```
function havePreviewTarget()
{
    var bHavePreviewTarget = false;

    if (dw.getFocus(true) == 'site')
    {
        if (site.getFocus() == 'remote')
        {
            bHavePreviewTarget = site.getRemoteSelection().length > 0 &&
                                        site.canBrowseDocument();
        }
        else if (site.getFocus() != 'none')
        {
            var selFiles = site.getSelection();

            if (selFiles.length > 0)
            {
                var i;

                bHavePreviewTarget = true;

                for (i = 0; i < selFiles.length; i++)
                {
                    var selFile = selFiles[i];

                    // For server connections, the files will
                    // already be remote URLs.

                    if (selFile.indexOf("://") == (-1))
                    {
                        var urlPrefix = "file:///";
                        var strTemp = selFile.substr(urlPrefix.length);

                        if (selFile.indexOf(urlPrefix) == -1)
                            bHavePreviewTarget = false;
                        else if (strTemp.indexOf("/") == -1)
                            bHavePreviewTarget = false;
                        else if (!DWfile.exists(selFile))
                            bHavePreviewTarget = false;
                        else if (DWfile.getAttributes(selFile).indexOf("D") != -1)
                            bHavePreviewTarget = false;
                    }
                    else
                    {
```

```
                    bHavePreviewTarget = true;
                }
            }
        }
    }
    else if (dw.getFocus() == 'document' ||
        dw.getFocus() == 'textView' || dw.getFocus("true") == 'html' )
    {
        var dom = dw.getDocumentDOM('document');
        if (dom != null)
        {
            var parseMode = dom.getParseMode();
            if (parseMode == 'html' || parseMode == 'xml')
                bHavePreviewTarget = true;
        }
    }

    return bHavePreviewTarget;
}
```

havePreviewTarget() 関数は、bHavePreviewTarget という値を、デフォルトの戻り値として false に設定します。この関数は、dw.getFocus() 関数を呼び出して 2 つの基本的なテストを行い、現在アプリケーションのどの部分に入力フォーカスがあるかを判別します。最初のテストでは、サイトパネルにフォーカスがあるかどうかをテストします（if (dw.getFocus(true) == 'site')）。サイトパネルにフォーカスがない場合は、2 番目のテストで、ドキュメント（dw.getFocus() == 'document'）、テキストビュー（dw.getFocus() == 'textView'）またはコードインスペクタ（dw.getFocus("true") == 'html'）にフォーカスがあるかどうかをチェックします。いずれのテスト結果も true ではない場合、関数は値として false を返します。

サイトパネルにフォーカスがある場合、関数はビュー設定がリモートビューかどうかをチェックします。ビュー設定がリモートビューの場合、リモートファイルがあれば、関数が bHavePreviewTarget を true に設定し（site.getRemoteSelection().length > 0）、これらのファイルをブラウザで開くことができます（site.canBrowseDocument()）。ビュー設定がリモートビューではなく、ビューが「なし」でもない場合、関数は選択したファイルのリストを file:/// URL の形式で取得します（var selFiles = site.getSelection();）。

選択したリスト内の項目ごとに、関数は「://」という文字列が存在するかどうかをテストします。この文字列が検出されない場合は、コードによりリスト項目に対して一連のテストが実行されます。項目が file:/// URL の形式でない場合は（if (selFile.indexOf(urlPrefix) == -1)）、戻り値が false に設定されます。接頭辞 file:/// に続くストリングの残りの部分にスラッシュ（/）が含まれていない場合は（if (strTemp.indexOf("/") == -1)）、戻り値が false に設定されます。ファイルが存在しない場合は（else if (!DWfile.exists(selFile))）、戻り値が false に設定されます。最後に、指定したファイルがフォルダかどうかがチェックされます（else if (DWfile.getAttributes(selFile).indexOf("D") != -1)）。selfile がフォルダの場合は、関数が値として false を返します。ターゲットがファイルの場合は、関数が bHavePreviewTarget の値を true に設定します。

ドキュメント、テキストビューまたはコードインスペクタに入力フォーカスがある場合は（else if (dw.getFocus() == 'document' || dw.getFocus() == 'textView' || dw.getFocus("true") == 'html')）、関数が DOM を取得し、ドキュメントが HTML または XML ドキュメントかどうかをチェックします。HTML または XML ドキュメントの場合は、関数が bHavePreviewTarget を true に設定します。最終的に、関数は bHavePreviewTarget に保存していた値を返します。

receiveArguments()

receiveArguments() 関数が呼び出され、メニュー項目から渡されたあらゆる引数をコマンドによって処理します。ブラウザでプレビューメニューについては、receiveArguments() 関数がユーザの選択したブラウザを呼び出します。receiveArguments() 関数のコード例を次に示します。

```
function receiveArguments()
{
    var whichBrowser = arguments[0];
    var theBrowser = null;
    var i=0;
    var browserList = null;
    var result = false;

    if (havePreviewTarget())
    {
        // Code to check if we were called from a shortcut key
        if (whichBrowser == 'primary' || whichBrowser == 'secondary')
        {
            // get the path of the selected browser
            if (whichBrowser == 'primary')
            {
                theBrowser = dw.getPrimaryBrowser();
            }
            else if (whichBrowser == 'secondary')
            {
                theBrowser = dw.getSecondaryBrowser();
            }

            // Match the path with the name of the corresponding browser
            // that appears in the menu.
            browserList = dw.getBrowserList();
            while(i < browserList.length)
            {
                if (browserList[i+1] == theBrowser)
                    theBrowser = browserList[i];
                i+=2;
            }
        }
        else
            theBrowser = whichBrowser;
        // Only launch the browser if we have a valid browser selected.
        if (theBrowser != "file:///" && typeof(theBrowser) != "undefined" &&
        theBrowser.length > 0)
        {
            if (dw.getFocus(true) == 'site')
            {
                // Only get the first item of the selection because
                // browseDocument() can't take an array.
                //dw.browseDocument(site.getSelection()[0],theBrowser);
                site.browseDocument(theBrowser);
            }
            else
                dw.browseDocument(dw.getDocumentPath('document'),theBrowser);
```

```
        }
        else
        {
            // Otherwise, F12 or Ctrl+F12 was pressed, ask if the user wants
            // to specify a primary or secondary browser now.
            if (whichBrowser == 'primary')
            {
                result = window.confirm(MSG_NoPrimaryBrowserDefined);
            }
            else if (whichBrowser == 'secondary')
            {
                result = window.confirm(MSG_NoSecondaryBrowserDefined);
            }

            // If the user clicked OK, show the prefs dialog with the browser panel.
            if (result)
                dw.showPreferencesDialog('browsers');
        }
    }
}
```

この関数は、最初に変数 whichBrowser を Dreamweaver から渡される値である arguments[0] に設定します。その他のデフォルト値を設定すると共に、関数は result をデフォルト値の false に設定します。

変数が初期化された後で、receiveArguments() 関数はユーザ定義の関数 havePreviewTarget() を呼び出し、結果をテストします。テストの結果が true なら、関数はユーザがプライマリブラウザまたはセカンダリブラウザを選択しているかどうかをチェックします。選択されている場合、関数は変数 theBrowser をブラウザを起動する実行ファイルのパスに設定します（dw.getPrimaryBrowser() または dw.getSecondaryBrowser()）。次に、関数は dw.getBrowsersList() から返されたブラウザのリストを調べるループを実行します。リスト内のブラウザへのパスがプライマリブラウザまたはセカンダリブラウザへのパスと一致する場合、関数は変数 theBrowser を browserList の一致する値に設定します。この値には、ブラウザの名前とブラウザを起動する実行ファイルへのパスが含まれます。havePreviewTarget() が false の値を返してきたら、関数は変数 theBrowser を、変数 whichBrowser の値に設定します。

次に receiveArguments() 関数は theBrowser 変数をテストして、この変数がパスで開始していないこと、"undefined" でないこと、0 より大きいことを確認します。これらの条件がすべて true で、サイトパネルにフォーカスがある場合、receiveArguments() 関数は site.browseDocument() 関数を呼び出し、サイトパネルで選択されたファイルで、選択されたブラウザを呼び出します。サイトパネルにフォーカスがない場合は、receiveArguments() 関数が dw.browseDocument() 関数を呼び出して、現在のドキュメントのパスと、このドキュメントを開くブラウザの名前を指定する変数 theBrowser の値を渡します。

ユーザがショートカットキー（F12 キーまたは Ctrl + F12 キー）を押したにもかかわらず、プライマリブラウザおよびセカンダリブラウザが指定されていない場合は、ダイアログボックスが表示され、それをユーザに知らせます。ユーザが「OK」をクリックすると、関数は dw.showPreferencesDialog() 関数と browsers 引数を呼び出して、その時点でユーザがブラウザを指定できるようにします。

メニューコマンド API 関数

メニューコマンド API のカスタム関数は必須ではありません。

canAcceptCommand()

説明

メニュー項目をアクティブにするか、または淡色表示にするかを決定します。

引数
{arg1}、{arg2}、...{argN}}

これが動的メニュー項目の場合、getDynamicContents() 関数によって指定された固有の ID が唯一の引数になります。動的メニューではない場合、arguments 属性が menuitem タグに対して定義されていれば、その属性値が canAcceptCommand() 関数（および 161 ページの「isCommandChecked()」関数、162 ページの「receiveArguments()」関数、162 ページの「setMenuText()」関数）に 引数として渡されます。arguments 属性は、同じメニューコマンドを呼び出す 2 つのメニュー項目を区別するときに便利です。

注意：動的メニュー項目では、arguments 属性は無視されます。

戻り値
ブール値。項目を有効にする必要がある場合は true、そうでない場合は false が返されます。

commandButtons()

説明
オプションダイアログボックスの右側に表示されるボタンと、それらのボタンをクリックした際のビヘイビアを定義します。この関数が定義されていない場合、ボタンは表示されず、メニューコマンドファイルの body セクションがダイアログボックス全体に拡大表示されます。

引数
なし。

戻り値
偶数のエレメントを含む配列。最初のエレメントは、一番上のボタンのラベルを含むストリングです。2 番目のエレメントは、一番上のボタンがクリックされた際のビヘイビアを定義する JavaScript コードのストリングです。残りのエレメントにより、その他のボタンが同様に定義されます。

例
次の commandButtons() 関数の例では、「OK」、「キャンセル」および「ヘルプ」の 3 つのボタンを定義します。

```
function commandButtons(){
    return new Array("OK" , "doCommand()" , "Cancel" , ¬
    "window.close()" , "Help" , "showHelp()");
}
```

getDynamicContent()

説明
メニューの動的な部分の内容を取得します。

引数
menuID

menuID 引数には、メニュー項目に関連付けられている id 属性値（メニュー項目に関連付けられている menuitem タグの）を指定します。

戻り値

ストリングの配列。各ストリングには、メニュー項目の名前および固有の ID が含まれており、これらはセミコロンで区切られます。この関数から null 値が返されると、メニューは変更されません。

例

次の getDynamicContent() 関数の例では、4 つのメニュー項目（My Menu Item 1、My Menu Item 2、My Menu Item 3 および My Menu Item 4）の配列が返されます。

```
function getDynamicContent(){
    var stringArray= new Array();
    var i=0;
    var numItems = 4;

    for (i=0; i<numItems;i++)
        stringArray[i] = new String("My Menu Item " + i + ";¬
        id='My-MenuItem" + i + "'");

    return stringArray;
}
```

isCommandChecked()

説明

メニュー項目の横にチェックマークを表示するかどうかを決定します。

引数

{arg1}、{arg2}、...{argN}

これが動的メニュー項目の場合、getDynamicContents() 関数によって指定された固有の ID が唯一の引数になります。動的メニューではない場合、arguments 属性が menuitem タグに対して定義されていれば、その属性値が isCommandChecked() 関数（および 159 ページの「canAcceptCommand()」関数、162 ページの「receiveArguments()」関数、162 ページの「setMenuText()」関数）に引数として渡されます。arguments 属性は、同じメニューコマンドを呼び出す 2 つのメニュー項目を区別するときに便利です。

注意：動的メニュー項目では、arguments 属性は無視されます。

戻り値

ブール値。メニュー項目の横にチェックマークを付ける必要がある場合は true が、そうでない場合は false が返されます。

例

```
function isCommandChecked()
{
    var bChecked = false;
    var cssStyle = arguments[0];

    if (dw.getDocumentDOM() == null)
        return false;

    if (cssStyle == "(None)")
    {
        return dw.cssStylePalette.getSelectedStyle() == '';
    }
    else
    {
        return dw.cssStylePalette.getSelectedStyle() == cssStyle;
    }
        return bChecked;
}
```

receiveArguments()

説明

メニュー項目または dw.runCommand() 関数から渡された引数を処理します。動的メニュー項目の場合、動的メニュー項目 ID を処理します。

引数

{arg1}、{arg2}、...{argN}

これが動的メニュー項目の場合、getDynamicContents() 関数によって指定された固有の ID が唯一の引数になります。動的メニューではない場合、arguments 属性が menuitem タグに対して定義されていれば、その属性値が receiveArguments() 関数（および 159 ページの「canAcceptCommand()」関数、161 ページの「isCommandChecked()」関数、162 ページの「setMenuText()」関数）に引数として渡されます。arguments 属性は、同じメニューコマンドを呼び出す 2 つのメニュー項目を区別するときに便利です。

注意：動的メニュー項目では、arguments 属性は無視されます。

戻り値

なし。

例

```
function receiveArguments()
{
    var styleName = arguments[0];
    if (styleName == "(None)")
        dw.getDocumentDOM('document').applyCSSStyle('','');
    else
        dw.getDocumentDOM('document').applyCSSStyle('',styleName);
}
```

setMenuText()

説明

メニューに表示するテキストを指定します。

注意：160 ページの「getDynamicContent()」を使用している場合は、この関数を使用しないでください。

引数
{arg1}、{arg2}、...{argN}

arguments 属性が menuitem タグに対して定義されていれば、その属性値が setMenuText() 関数（および 159 ページの「canAcceptCommand()」関数、161 ページの「isCommandChecked()」関数、162 ページの「receiveArguments()」関数）に引数として渡されます。arguments 属性は、同じメニューコマンドを呼び出す 2 つのメニュー項目を区別するときに便利です。

戻り値
メニューに表示するストリング。

例

```
function setMenuText()
{
    if (arguments.length != 1) return "";

    var whatToDo = arguments[0];
    if (whatToDo == "undo")
        return dw.getUndoText();
    else if (whatToDo == "redo")
        return dw.getRedoText();
    else return "";
}
```

windowDimensions()

説明
パラメータダイアログボックスに特定のサイズを設定します。この関数が定義されていない場合、ウィンドウのサイズは自動的に計算されます。

注意：640 x 480 ピクセルより大きいダイアログボックスを使用する場合に限り、この関数を定義します。

引数
platform

platform 引数の値は、ユーザのプラットフォームに応じて、"macintosh" または "windows" になります。

戻り値
"widthInPixels,heightInPixels" という形式のストリング。

返されるサイズはダイアログボックス全体のサイズよりも小さくなります。これは「OK」と「キャンセル」に使用する領域が含まれていないためです。返されたサイズにすべてのオプションが収まらない場合は、スクロールバーが表示されます。

例
次の windowDimensions() の例では、パラメータダイアログボックスのサイズを 648 x 520 ピクセルに設定します。

```
function windowDimensions(){
    return "648,520";
}
```

第 11 章：ツールバー

Adobe Dreamweaver のツールバーは、簡単に作成することができます。ツールバーを作成するには、ツールバーを定義するファイルを作成し、そのファイルを Configuration¥Toolbars フォルダ内に配置します。

次の表にツールバーを作成するために使用するファイルの一覧を示します。

パス	ファイル	説明
Configuration¥Toolbars¥	toolbars.xml	このファイルを編集して、ツールバーの内容を変更します。
Configuration¥Toolbars¥	**newtoolbar**.xml	このファイルを作成して、ツールバーを作成します。
Configuration¥Toolbars¥	**imagefile**.gif	ツールバーのコントロールのアイコンイメージです。
Configuration¥Commands¥	MyCommand.htm	ツールバーの項目に関連付けられているコマンドファイルです。

ツールバーのしくみ

ツールバーは、Dreamweaver の主要フォルダである Configuration フォルダの Toolbars フォルダに格納されている XML ファイルとイメージファイルによって定義されます。Dreamweaver のデフォルトのツールバーは、Configuration¥Toolbars フォルダ内の toolbars.xml ファイルに格納されています。Dreamweaver が起動されると、Toolbars フォルダ内のすべてのツールバーファイルが読み込まれます。元の toolbars.xml ファイルを修正しなくても、Toolbars フォルダにファイルをコピーするだけで、新しいツールバーを追加できます。

ツールバーの XML ファイルでは、複数のツールバーとツールバー項目が定義されます。ツールバーとは、ボタン、テキストボックス、ポップアップメニューなどの項目の一覧です。ツールバー項目とは、ユーザがツールバー内でアクセスできる個々のコントロールのことです。

プッシュボタンやポップアップメニューなどの一部の種類のツールバーコントロールには、アイコンイメージが関連付けられています。アイコンイメージは、Toolbars フォルダ内のイメージフォルダに格納されます。イメージのファイル形式は、Dreamweaver で描画できる任意の形式を使用できますが、通常は GIF または JPEG です。アドビ システムズ社によって作成されたツールバーのイメージは、Toolbars¥images¥MM フォルダに格納されています。

メニューの場合と同様に、個々のツールバー項目の機能は、項目の属性またはコマンドファイルを使用して指定できます。アドビ システムズ社によって作成されたツールバーのコマンドファイルは、Toolbars¥MM フォルダに格納されています。


ツールバーの API はメニューコマンドの API と互換性があるため、ツールバーコントロールでメニューコマンドファイルを再利用できます。

メニューとは異なり、ツールバー項目は、それらを使用するツールバーとは切り離して定義することができます。この柔軟性により、itemref タグを使用して複数のツールバーでツールバー項目を使用することができます。

ツールバーの初回読み込み時に、ツールバー定義によってツールバーの表示状態と位置が設定されます。その後、表示状態と位置は、レジストリ（Windows）または Dreamweaver の環境設定ファイル（Macintosh）に保存され、そこから復元されます。

ツールバーの動作

Windows では、Dreamweaver のツールバーは Windows の標準のツールバーと同様に動作します。Dreamweaver のツールバーの特性は以下のとおりです。

- ツールバーをドラッグ& ドロップして、ドッキング、ドッキング解除および他のツールバーを基準にした再配置を行うことができます。
- ツールバーは、フレームウィンドウの上部または下部に水平方向にドッキングすることができます。
- ツールバーのサイズは固定されています。コンテナが縮小した場合や他のツールバーが隣に配置された場合にも縮小することはありません。
- 表示／ツールバーメニューで、ツールバーの表示 / 非表示を切り替えることができます。
- ツールバーをオーバーラップすることはできません。
- ツールバーは、ドラッグしている間はアウトラインだけが表示されます。

Macintosh では、ツールバーは常にドキュメントウィンドウに結合されます。メニューでツールバーの表示 / 非表示を切り替えることはできますが、ツールバーのドラッグ& ドロップ、再配置、ドッキング解除を行うことはできません。

Dreamweaver ワークスペースでは、すべての Dreamweaver ドキュメントウィンドウが 1 つの親フレームに組み込まれ、ツールバーをワークスペースフレームとドキュメントウィンドウのどちらにドッキングするかを指定できます。

Dreamweaver ワークスペースフレームにドッキングしたツールバーの場合、各ツールバーのインスタンスは 1 つだけです。この場合、ツールバーは常に前面にあるドキュメントに対して動作します。Dreamweaver ワークスペースでは、ツールバーを挿入ツールバーの上、下、左、右のいずれかの位置にドッキングできます。Dreamweaver ワークスペースフレームに結合されているツールバーは、ドキュメントウィンドウが開いていなくても自動的に無効になることはありません。ドキュメントが開いていない場合にツールバー項目が無効になるかどうかは、ツールバー項目によって決まります。

ドキュメントウィンドウにドッキングしたツールバーの場合、ウィンドウごとに 1 つのインスタンスがあります。ドキュメントウィンドウに結合されているツールバーは、結合先のドキュメントウィンドウが前面のドキュメントでない場合は完全に無効になり、結合先のドキュメントウィンドウが前面に表示されると、すべての更新ハンドラを再実行します。

ドキュメントウィンドウと Dreamweaver ワークスペースフレームの間でツールバーをドラッグ& ドロップすることはできません。

ツールバーコマンドの動作

ツールバーが描画されるときに、次のイベントが発生します。

1. ツールバーのコントロール項目ごとに、file 属性があるかどうかが判別されます。
2. file 属性がある場合は、canAcceptCommand() 関数が呼び出されて、ドキュメントの現在のコンテキストでコントロールを有効にするかどうかが決定されます。

   例えば、Dreamweaver ツールバーにある「タイトル」テキストボックスの場合、canAcceptCommand() 関数は現在の DOM があるかどうか、および現在のドキュメントが HTML ファイルであるかどうかをチェックします。この両方の条件が真の場合は、この関数から true が返され、ツールバー上でこのテキストボックスが有効になります。
3. file 属性がある場合は、checked、command、DOMRequired、enabled、script、showif、update および value が指定されていても無視されます。
4. file 属性がない場合は、checked、command、DomRequired などのツールバーコントロール項目に対して設定されている属性が処理されます。

   個々の項目タグ属性について詳しくは、「176 ページの「項目タグ属性」」を参照してください。
5. 更新サイクルごとに getCurrentValue() 関数が呼び出されて、コントロールに対して表示する値が決定されます。この更新サイクルは、update 属性で指定されています。

6 ユーザがツールバー上の項目を選択します。

7 receiveArguments() 関数が呼び出されて、ツールバー項目の arguments 属性で指定されている引数が処理されます。

ツールバーコマンド API の個々の関数について詳しくは、181 ページの「ツールバーコマンド API 関数」を参照してください。

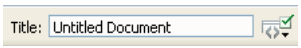
単純なツールバーコマンドファイル

この簡単な例では、Dreamweaver ドキュメントツールバーに表示される「タイトル」テキストボックスを実装します。このテキストボックス項目では、現在の Dreamweaver ドキュメントの名前を入力することができます。このツールバーの例を実装するには、次の手順を実行します。

テキストボックスの作成

Dreamweaver にツールバーを追加するには、ツールバー定義を含む XML ファイルを Dreamweaver の Configuration フォルダ内にある Toolbars フォルダに配置します。

次の図は、「タイトル」テキストボックスを示しています。



以下のツールバーの editcontrol 項目では、「Title:」というラベルの付いたテキストボックスを定義します。

```
<EDITCONTROL ID="DW_SetTitle"
    label="Title: "
    tooltip="Document Title"
    width="150"
    file="Toolbars/MM/EditTitle.htm"/>
```

tooltip 属性により、ユーザがマウスポインタをテキストボックスの上に置くとツールヒントボックスにタイトルが表示されます。width 属性では、フィールドのサイズをピクセル単位で指定します。file 属性では、テキストボックスに対して動作する JavaScript 関数を EditTitle.htm ファイルに含めるように指定します。Dreamweaver ドキュメントツールバーの完全な定義を確認するには、toolbars.xml ファイル内のメインツールバー（id="DW_Toolbar_Main"）を参照してください。

テキストボックスの JavaScript コード

ユーザがテキストボックスを操作すると、Toolbars¥MM フォルダ内の EditTitle.htm コマンドファイルが呼び出されます。このファイルには、「タイトル」テキストボックスに対して動作する 3 つの JavaScript 関数が格納されています。具体的には、canAcceptCommand()、receiveArguments() および getCurrentValue() です。

canAcceptCommand() による ツールバー項目の有効化

canAcceptCommand() 関数は、現在のドキュメントオブジェクトモデル（DOM）があるかどうか、およびドキュメントが HTML として解析されているかどうかをチェックする 1 行のコードから構成されています。この関数は、これらのテストの結果を返します。条件が true である場合は、ツールバー上でこのテキストボックス項目が有効化されます。関数が値 false を返した場合は、項目が無効化されます。

この関数は次のようになります。

```
function canAcceptCommand()
{
    return (dw.getDocumentDOM() != null && dw.getDocumentDOM().getParseMode() == 'html');
}
```

receiveArguments() による タイトルの設定

次の例で示すように、ユーザが「タイトル」テキストボックスに値を入力して Enter キーを押すか、このコントロールからフォーカスを移動すると、receiveArguments() 関数が呼び出されます。

この関数は次のようになります。

```
function receiveArguments(newTitle)
{
    var dom = dw.getDocumentDOM();
    if (dom)
        dom.setTitle(newTitle);
}
```

ユーザが入力した値 newTitle が receiveArguments() 関数に渡されます。receiveArguments() 関数は、まず現在の DOM があるかどうかをチェックします。現在の DOM がある場合は、receiveArguments() 関数が newTitle を dom.setTitle() 関数に渡して新しいドキュメントタイトルを設定します。

getCurrentValue() による タイトルの取得

onEdit イベントハンドラのデフォルトの更新頻度で決まる更新サイクルごとに getCurrentValue() 関数が呼び出されて、コントロールに対して表示する値が決定されます。「タイトル」テキスト編集コントロールには update 属性がないため、onEdit ハンドラのデフォルトの更新頻度によって更新頻度が決定されます。

「タイトル」テキストボックスの場合、以下の getCurrentValue() 関数が、JavaScript アプリケーションプログラミングインターフェイス（API）関数 dom.getTitle() を呼び出して、現在のタイトルを取得して返します。

この関数は次のようになります。

```
function getCurrentValue()
{
    var title = "";
    var dom = dw.getDocumentDOM();
    if (dom)
        title = dom.getTitle();
    return title;
}
```

ユーザがドキュメントのタイトルを入力するまで、getTitle() 関数は「無題ドキュメント」を返し、これがテキストボックスに表示されます。ユーザがタイトルを入力すると、getTitle() 関数はその値を返し、これが新しいドキュメントタイトルとして表示されます。

「タイトル」テキストボックスの JavaScript 関数を含む完全な HTML ファイルを確認するには、Toolbars¥MM フォルダ内の EditTitle.htm ファイルを参照してください。

MM フォルダは、Adobe ファイル用に予約されています。Toolbars フォルダ内に別のフォルダを作成して、JavaScript コードをその中に配置します。

ツールバー定義ファイル

ツールバーは、ラジオボタン、チェックボタン、編集ボックスなどのツールバー項目のリストであり、必要に応じて separator タグで分割されます。各ツールバー項目は、itemref タグを使用した項目への参照、separator タグを使用したセパレータ、またはチェックボックスや編集ボックスなどについての完全なツールバー項目定義のいずれかです。詳しくは 171 ページの「ツールバー項目タグ」を参照してください。

各ツールバー定義ファイルは、次の宣言で始まります。

```
<?xml version="1.0" encoding="optional_encoding"?>
<!DOCTYPE toolbarset SYSTEM "-//Macromedia//DWExtension toolbar 5.0">
```

エンコードを省略すると、オペレーティングシステムのデフォルトのエンコードが使用されます。

ファイルは、この宣言の後ろに続く 1 つの toolbarset タグとそのタグに含まれる任意の数の toolbar タグ、itemref タグ、separator タグ、include タグおよび **itemtype** タグで構成されます。ここで、**itemtype** タグは、button、checkbutton、radiobutton、menubutton、dropdown、combobox、editcontrol または colorpicker です。次の例は、toolbars.xml ファイルからの抜粋で、ツールバーファイル内のタグの階層を示しています。この例では、ツールバー項目の属性の説明をエリプシス（. . .）にしています。

```
<?xml version="1.0"?>
<!DOCTYPE toolbarset SYSTEM "-//Adobe//DWExtension toolbar 10.0">
<toolbarset>

<!-- main toolbar -->
    <toolbar id="DW_Toolbar_Main" label="Document">
        <radiobutton id="DW_CodeView" . . ./>
        <radiobutton id="DW_SplitView" . . ./>
        <radiobutton id="DW_DesignView" . . ./>
        <separator/>
        <checkbutton id="DW_LiveDebug" . . ./>
        <checkbutton id="DW_LiveDataView" . . ./>
        <separator/>
        <editcontrol id="DW_SetTitle" . . ./>
        <menubutton id="DW_FileTransfer" . . ./>
        <menubutton id="DW_Preview" . . ./>
        <separator/>
        <button id="DW_DocRefresh" . . ./>
        <button id="DW_Reference" . . ./>
        <menubutton id="DW_CodeNav" . . ./>
        <menubutton id="DW_ViewOptions" . . ./>
    </toolbar>
</toolbarset>
```

次に、ツールバーの各タグについて説明します。

<toolbar>

説明

ツールバーを定義します。項目とセパレータは、指定された順序で左から右に表示されます。項目のレイアウトは自動的に行われます。ツールバーファイルでは項目間の間隔を指定しませんが、特定のタイプの項目の幅を指定することができます。

属性

id、label、{container}、{initiallyVisible}、{initialPosition}、{relativeTo}

- id="***unique_id***"。必須。識別子ストリングは、ファイル内、およびそのファイルに含まれるすべてのファイル内で、一意である必要があります。ツールバーを操作する JavaScript API 関数は、ツールバーをこの ID で参照します。これらの関数について詳しくは、『Dreamweaver API リファレンス』を参照してください。同じファイルに含まれている 2 つのツールバーの ID が同一である場合は、エラーが表示されます。
- label="string"。必須。label 属性はラベルを指定します。ラベルは、Dreamweaver でユーザに表示される文字列です。ラベルは、表示／ツールバーメニューに表示されます。ツールバーがフローティング状態の場合は、ツールバーのタイトルバーにも表示されます。
- container="mainframe" または "document"。デフォルト値は "mainframe"。Windows 上の Dreamweaver ワークスペース内のツールバーのドッキング先を指定します。コンテナを "mainframe" に設定すると、ツールバーは外側のワークスペースフレームに表示され、前面のドキュメントに対して機能します。コンテナを "document" に設定すると、ツールバーは各ドキュメントウィンドウに表示されます。Macintosh では、各ドキュメントウィンドウにすべてのツールバーが表示されます。

- initiallyVisible="true" または "false"。このタグでは、ツールバーが Toolbars フォルダから最初に読み込まれるときにツールバーを表示するかどうかを指定します。2 回目以降は、ユーザが表示 / 非表示を制御します。ユーザが Dreamweaver を終了すると、現在の状態がシステムレジストリ（Windows）または Dreamweaver の環境設定ファイル（Macintosh）に保存されます。Dreamweaver が再起動すると、レジストリまたは環境設定ファイルから設定が復元されます。ツールバーの表示 / 非表示は、dom.getToolbarVisibility() 関数と dom.setToolbarVisibility() 関数を使用して操作できます。詳しくは、『Dreamweaver API リファレンス』を参照してください。initiallyVisible 属性を設定しないと、デフォルトの true が使用されます。
- initialPosition="top"、"below" または "floating"。ツールバーが初めて読み込まれるときにツールバーを最初に配置する位置（他のツールバーからの相対位置）を指定します。指定できる intialPosition の値については、以下で説明します。
- top。これがデフォルトの位置です。つまり、ツールバーはドキュメントウィンドウの最上部に表示されます。指定したウィンドウタイプの複数のツールバーに top を指定すると、ツールバーは読み込み時に検出された順序で表示されます。ツールバーが別々のファイルに格納されている場合は、この検出順序を予測できません。
- below。ツールバーは relativeTo 属性で指定されているツールバー直下の行の先頭に表示されます。relativeTo で指定されたツールバーが見つからない場合は、エラーが表示されます。複数のツールバーに同じツールバーを基準として below を指定すると、ツールバーは読み込み時に検出された順序で表示されます。ツールバーが別々のファイルに格納されている場合は、この検出順序を予測できません。
- floating。ツールバーは、初期状態ではウィンドウにドッキングされず、ドキュメントの上でフローティング状態になります。ツールバーは自動的に他のフローティングツールバーからずらして配置されます。Macintosh の場合、floating は、top と同様に扱われます。

  initiallyVisible 属性の場合と同様に、initialPosition 属性はツールバーが初めて読み込まれるときにのみ適用されます。その後、ツールバーの位置は、レジストリ（Windows）または Dreamweaver の環境設定ファイル（Macintosh）に保存されます。dom.setToolbarPosition() 関数を使用して、ツールバーの位置をリセットすることができます。dom.setToolbarPosition() 関数について詳しくは、『Dreamweaver API リファレンス』を参照してください。

  initialPosition 属性を指定しないと、ツールバーは読み込み時に検出された順序で配置されます。
- relativeTo="***toolbar_id***"。initialPosition 属性で below が指定されている場合、この属性は必須です。それ以外の場合、この引数は無視されます。指定した ID のツールバーの下にこのツールバーが配置されます。

コンテンツ

toolbar タグには、include、itemref、separator の各タグと、button、combobox、dropdown などの個別の項目定義が含まれています。指定できる項目定義について詳しくは、171 ページの「ツールバー項目タグ」を参照してください。

コンテナ

toolbarset タグ。

例

```
<toolbar id="MyDWedit_toolbar" label="Edit">
```

<include/>

説明

現在のファイルの読み込みを続行する前に、指定されたファイルからツールバー項目を読み込みます。組み込まれたファイルに定義されているツールバー項目は、現在のファイルで参照できます。ファイルが別のファイルを再帰的に組み込もうとすると、エラーメッセージが表示され、その再帰的な組み込みは無視されます。組み込まれたファイル内の toolbar タグはスキップされますが、それらのツールバー内のツールバー項目は現在のファイルでの参照に使用できます。

属性

- 組み込むツールバー XML ファイルの file パス。このパスは、Toolbars フォルダからの相対パスです。

コンテンツ

なし。

コンテナ

toolbar タグまたは toolbarset タグ。

例

```
<include file="mine/editbar.xml"/>
```

<itemtype/>

説明

個々のツールバー項目を定義します。ツールバー項目には、ボタン、ラジオボタン、チェックボタン、コンボボックス、ポップアップメニューなどがあります。定義可能なツールバー項目のタイプのリストについては、171 ページの「ツールバー項目タグ」を参照してください。

属性

属性は定義する項目によって異なります。ツールバー項目に指定できる属性のリストについては、176 ページの「項目タグ属性」を参照してください。

コンテンツ

なし。

コンテナ

toolbar タグまたは toolbarset タグ。

例

```
<button id="strikeout_button" .../>
```

<itemref/>

説明

前のツールバーの内側またはすべてのツールバーの外側に定義されたツールバー項目を参照し、現在のツールバーに組み込みます。

属性

id、{showIf}

- id="***id_reference***"。必須。事前に定義されているか、ファイルに組み込まれている項目の ID を指定する必要があります。Dreamweaver では、前方参照は許可されていません。ツールバー項目タグが参照している ID が定義されていない場合は、エラーが報告され、その参照は無視されます。
- showIf="***script***"。指定したスクリプトが値 true を返した場合にのみ、該当項目がツールバー上に表示されるように指定します。例えば、showIf を使用して、指定したアプリケーションだけで特定のボタンを表示したり、Adobe ColdFusion、ASP、JSP などのサーバサイドの言語でページが記述されている場合にだけ特定のボタンを表示したりできます。showIf

を指定しない場合、項目は常に表示されます。このプロパティは、項目のイネーブラが動作するたびにチェックされます。つまり、update 属性の値に従ってチェックされます。この属性は慎重に使用してください。この属性は、項目定義またはツールバーからの項目の参照で使用できます。定義と参照の両方で showIf 属性を指定すると、両方の条件が真である場合にのみ項目が表示されます。showIf 属性は、コマンドファイル内の showIf() 関数と同じです。

コンテンツ
なし。

コンテナ
toolbar タグまたは toolbarset タグ。

例
```
<itemref id="strikeout_button">
```

<separator/>

説明
ツールバー内の現在の場所にセパレータを挿入します。

属性
{showIf}

- showif 属性は、指定されたスクリプトが true を返した場合にのみ、ツールバーにセパレータが表示されるように指定します。例えば、showIf 属性を使用して、指定したアプリケーションだけでセパレータを表示したり、ページに特定のドキュメントタイプがある場合にだけセパレータを表示したりできます。showIf 属性を指定しない場合、セパレータは常に表示されます。

コンテンツ
なし。

コンテナ
toolbar タグ。

例
```
<separator/>
```

ツールバー項目タグ

各タイプのツールバー項目には、固有のタグ、および必須属性とオプション属性のセットがあります。toolbar 項目は、ツールバーの内側または外側で定義できます。一般には、ツールバーの外側でツールバー項目を定義し、itemref タグを使用してツールバー内でこれらの項目を参照することをお勧めします。

ツールバーには以下のタイプの項目を定義できます。

<button>

説明
このプッシュボタンをクリックすると、特定のコマンドが実行されます。外観と動作は、Dreamweaver ツールバーの「リファレンス」ボタンと同じです。

属性
id、image、tooltip、command、{showIf}、{disabledImage}、{overImage}、{label}、{file}、{domRequired}、{enabled}、{update}、{arguments}

各属性について詳しくは、176 ページの「項目タグ属性」を参照してください。

コンテンツ
なし。

コンテナ
toolbar タグまたは toolbarset タグ。

例
```
<BUTTON ID="DW_DocRefresh"
    image="Toolbars/images/MM/refresh.gif"
    disabledImage="Toolbars/images/MM/refresh_dis.gif"
    tooltip="Refresh Design View (F5)"
    enabled="((dw.getDocumentDOM() != null) && (dw.getDocumentDOM().getView() != 'browse')
        && (!dw.getDocumentDOM().isDesignViewUpdated()))"
    command="dw.getDocumentDOM().synchronizeDocument()"
    update="onViewChange,onCodeViewSyncChange"/>
```

<checkbutton>

説明
チェックボタンは、オンの状態とオフの状態があり、クリックされると特定のコマンドを実行するボタンです。オンの場合は、押された状態でハイライト表示されます。オフの場合は、平らな状態で表示されます。Dreamweaver で実装されているチェックボタンの状態は、マウスポインタを上に置いた状態、押した状態、押したままマウスポインタを上に置いた状態、および押したまま無効な状態です。checked 属性または isCommandChecked() 関数で指定されているハンドラでは、チェックボタンがクリックされたときにチェックボタンの状態が切り替わるようにする必要があります。

属性
id、{showIf}、image、{disabledImage}、{overImage}、tooltip、{label}、{file}、{domRequired}、{enabled}、checked、{update}、command、{arguments}

各属性について詳しくは、176 ページの「項目タグ属性」を参照してください。

コンテンツ
なし。

コンテナ
toolbar タグまたは toolbarset タグ。

例

```
<CHECKBUTTON ID="DW_LiveDebug"
    image="Toolbars/images/MM/debugview.gif"
    disabledImage="Toolbars/images/MM/globe_dis.gif"
    tooltip="Live Debug"
    enabled="dw.canLiveDebug()"
    checked="dw.getDocumentDOM() != null && dw.getDocumentDOM().getView() == 'browse'"
    command="dw.toggleLiveDebug()"
    showIf="dw.canLiveDebug()"
    update="onViewChange"/>
```

<radiobutton>

説明

ラジオボタンは、オフになっているときに飛び出したボタンのように見える点を除くと、チェックボタンとまったく同じです。Dreamweaver で実装されているラジオボタンの状態は、マウスポインタを上に置いた状態、押した状態、押したままマウスポインタを上に置いた状態、および押したまま無効な状態です。Dreamweaver では、ラジオボタンの相互排他が強制されません。checked 属性または isCommandChecked() 関数で指定されているハンドラでは、ラジオボタンのオン状態とオフ状態が互いに矛盾しないようにする必要があります。

ラジオボタンは、Dreamweaver ドキュメントツールバーの「コードビュー」ボタン、「デザインビュー」ボタンおよび「コードビューおよびデザインビュー」ボタンと同様に動作します。

属性

id、image、tooltip、checked、command、{showIf}、{disabledImage}、{overImage}、{label}、{file}、{domRequired}、{enabled}、{update}、{arguments}

各属性について詳しくは、176 ページの「項目タグ属性」を参照してください。

コンテンツ

なし。

コンテナ

toolbar タグまたは toolbarset タグ。

例

```
<RADIOBUTTON ID="DW_CodeView"
    image="Toolbars/images/MM/codeView.gif"
    disabledImage="Toolbars/images/MM/codeView_dis.gif"
    tooltip="Show Code View"
    domRequired="false"
    enabled="dw.getDocumentDOM() != null"
    checked="dw.getDocumentDOM() != null && dw.getDocumentDOM().getView() == 'code'"
    command="dw.getDocumentDOM().setView('code')"
    update="onViewChange"/>
```

<menubutton>

説明

メニューボタンは、menuid 属性で指定されているコンテキストメニューを呼び出すボタンです。Dreamweaver で実装されているメニューボタンの状態は、マウスポインタを上に置いた状態と押した状態です。Dreamweaver ではメニュー矢印（ボタンにメニュー項目が関連付けられていることを示す下向きの矢印）が描画されません。開発者が、この矢印をアイコンに組み込む必要があります。メニューボタンの例としては、Dreamweaver ドキュメントツールバーの「ファイル管理」ボタンと「コードナビゲーション」ボタンが挙げられます。

属性

id、image、tooltip、menuID、domRequired、enabled、{showIf}、{disabledImage}、{overImage}、{label}、{file}、{update}

各属性について詳しくは、176 ページの「項目タグ属性」を参照してください。

コンテンツ

なし。

コンテナ

toolbar タグまたは toolbarset タグ。

例

```
<MENUBUTTON ID="DW_CodeNav"
    image="Toolbars/images/MM/codenav.gif"
    disabledImage="Toolbars/images/MM/codenav_dis.gif"
    tooltip="Code Navigation"
    enabled="dw.getFocus() == 'textView' || dw.getFocus() == 'html'"
    menuID="DWCodeNavPopup"
    update="onViewChange"/>
```

<dropdown>

説明

ドロップダウンメニュー（またはポップアップメニュー）は、エントリを選択すると特定のコマンドを実行し、関連付けられている JavaScript 関数に基づいて自身を更新する編集不可のメニューです。ドロップダウンメニューは、小さなプロパティインスペクタサイズではなく標準サイズである点を除くと、外観と動作はテキストプロパティインスペクタの「フォーマット」コントロールと同じです。

属性

```
id, tooltip, file, enabled, checked, value, command, {showIf}, {label}, {width}, {domRequired}, {update},
{arguments}
```

各属性について詳しくは、176 ページの「項目タグ属性」を参照してください。

コンテンツ

なし。

コンテナ

toolbar タグまたは toolbarset タグ。

例

```
<dropdown id="Font_Example"
    width="115"
    tooltip="Font"
    domRequired="false"
    file="Toolbars/mine/fontExample.htm"
    update="onSelChange"/>
```

<combobox>

説明

コンボボックスは、ユーザがエントリを選択したときやテキストボックスで編集を行ってフォーカスを切り替えたときにコマンドを実行する、編集可能なポップアップメニューです。このメニューは、小さなプロパティインスペクタサイズではなく標準サイズである点を除くと、外観と動作はテキストプロパティインスペクタの「フォント」コントロールと同じです。

属性

id、file、tooltip、enabled、value、command、{showiI}、{label}、{width}、{domRequired}、{update}、{arguments}

各属性について詳しくは、176 ページの「項目タグ属性」を参照してください。

コンテンツ

なし。

コンテナ

toolbar タグまたは toolbarset タグ。

例

```
<COMBOBOX ID="Address_URL"
    width="300"
    tooltip="Address"
    label="Address: "
    file="Toolbars/MM/AddressURL.htm"
    update="onBrowserPageBusyChange"/>
```

<editcontrol>

説明

編集コントロールボックスは、ユーザがボックス内のテキストを変更してフォーカスを切り替えたときにコマンドを実行するテキスト編集ボックスです。

属性

id、tooltip、file、value、command、{showIf}、{label}、{width}、{domRequired}、{enabled}、{update}、{arguments}

各属性について詳しくは、176 ページの「項目タグ属性」を参照してください。

コンテンツ

なし。

コンテナ

toolbar タグまたは toolbarset タグ。

例

```
<EDITCONTROL ID="DW_SetTitle"
    label="Title: "
    tooltip="Document Title"
    width="150"
    file="Toolbars/MM/EditTitle.htm"/>
```

<colorpicker>

説明

カラーピッカーはカラーのウィンドウです。これには、ユーザが新しいカラーを選択したときにコマンドを実行するテキストボックスが関連付けられていません。外観と動作は、Dreamweaver のプロパティインスペクタのカラーピッカーと同じです。別のアイコンを指定してデフォルトのアイコンと置き換えることができます。

属性

id、tooltip、value、command、{showIf}、{image}、{disabledImage}、{overImage}、{label}、{colorRect}、{file}、{domRequired}、{enabled}、{update}、{arguments}

各属性について詳しくは、176 ページの「項目タグ属性」を参照してください。

コンテンツ

なし。

コンテナ

toolbar タグまたは toolbarset タグ。

例

```
<colorpicker id="Color_Example"
    image="Toolbars/images/colorpickerIcon.gif"
    disabledImage="Toolbars/images/colorpickerIconD.gif"
    colorRect="0 12 16 16"
    tooltip="Text Color"
    domRequired="false"
    file="Toolbars/mine/colorExample.htm"
    update="onSelChange"/>
```

項目タグ属性

ツールバー項目に属性とコマンドを割り当てることで、その項目の外観と動作を指定することができます。また、他のツールバーファイルを組み込んで、他のツールバーで定義されているツールバー項目を参照することもできます。ツールバー項目タグの属性の意味は次のとおりです。

id="unique_id"

必須。id 属性は、ツールバー項目の識別子です。id 属性は、現在のファイル内、および現在のファイルに組み込まれるすべてのファイル内で一意である必要があります。itemref タグは、項目の id を使用して項目を参照し、ツールバーに組み込みます。

例

```
<button id="DW_DocRerefresh" . . . >
```

showIf="script"

オプション。この属性では、スクリプトが値 true を返した場合にのみツールバーに項目が表示されるように指定します。例えば、showIf 属性を使用して、Adobe ColdFusion、ASP、JSP などのサーバサイドの言語でページが記述されている場合にのみ特定のボタンを表示することができます。showIf を指定しない場合、項目は常に表示されます。

showIf 属性は、項目のイネーブラが動作するたびにチェックされます。つまり、update 属性の値に従ってチェックされます。showIf 属性の使用は控えめにしてください。

showIf 属性は、項目定義または itemref タグでの項目の参照で指定できます。定義と参照で showIf 属性を指定すると、両方の条件が真である場合にのみ項目が表示されます。showIf 属性は、ツールバーコマンドファイル内の showIf() 関数と同じです。showIf 属性と showif() 関数の両方を指定した場合は、関数が属性よりも優先されます。

例

```
showIf="dw.canLiveDebug()"
```

image="image_path"

この属性は、ボタン、チェックボタン、ラジオボタン、メニューボタンおよびコンボボタンの場合は必須です。また、image 属性は、カラーピッカーの場合はオプションです。これ以外の項目タイプの場合は無視されます。image 属性では、ボタンに表示するアイコンファイルのパスを指定します。このパスは Configuration フォルダからの相対パスです。アイコンのファイル形式は、Dreamweaver で描画できる任意の形式を使用できますが、通常は GIF または JPEG です。

カラーピッカーにアイコンを指定すると、カラーピッカー全体がそのアイコンで置き換えられます。colorRect 属性も同時に指定すると、指定した四角形の中のアイコンの最上部に現在のカラーが表示されます。

例

```
image="Toolbars/images/MM/codenav.gif"
```

disabledImage="image_path"

オプション。disabledImage 属性は、ボタン、チェックボタン、ラジオボタン、メニューボタン、カラーピッカーおよびコンボボタンの場合以外は無視されます。この属性では、ボタンが無効になっている場合に表示するアイコンファイルのパスを指定します。このパスは Configuration フォルダからの相対パスです。disabledImage 属性を指定しないと、ボタンが無効になっている場合に、image 属性で指定されているイメージが表示されます。

例

```
disabledImage="Toolbars/images/MM/codenav_dis.gif"
```

overImage="image_path"

オプション。overImage 属性は、ボタン、チェックボタン、ラジオボタン、メニューボタン、カラーピッカーおよびコンボボタンの場合以外は無視されます。この属性では、ユーザがマウスポインタをボタンの上に移動したときに表示するアイコンファイルのパスを指定します。このパスは Configuration フォルダからの相対パスです。overImage 属性を指定しないと、ユーザがマウスポインタをボタンの上に移動してもボタンは変化しません。ただし、ボタンの周囲に輪が描画されます。

例

```
overImage="Toolbars/images/MM/codenav_ovr.gif"
```

tooltip="tooltip string"

必須。この属性では、マウスポインタをツールバー項目の上に重ねたときに表示する識別テキスト（ツールヒント）を指定します。

例

```
tooltip="Code Navigation"
```

label="label string"

オプション。この属性では、項目の横に表示されるラベルを指定します。ラベルにはコロンが自動的に追加されません。ボタン以外の項目のラベルは、常に項目の左に配置されます。ボタン、チェックボタン、ラジオボタン、メニューボタンおよびコンボボタンのラベルは、ボタン内部の、アイコンの右側に配置されます。

例

```
label="Title: "
```

width="number"

オプション。この属性では、項目の幅をピクセル単位で指定します。この属性はテキストボックス、ポップアップメニューおよびコンボボックスの項目のみに適用されます。width 属性を指定しない場合は、適切なデフォルトの幅が使用されます。

例

```
width="150"
```

menuID="menu_id"

メニューボタンとコンボボタンの場合は、関連するコマンドファイル内で getMenuID() 関数を指定しない限り、この属性が必須となります。その他のタイプの項目の場合、menuID 属性は無視されます。この属性では、ユーザがボタン、メニューボタン、コンボボタンをクリックしたときに表示されるコンテキストメニューが格納されているメニューバーの ID を指定します。この ID は、menus.xml ファイル内の menubar タグの ID 属性から取得されます。

例

```
menuID="DWCodeNavPopup"
```

colorRect="left top right bottom"

image 属性が指定されているカラーピッカーの場合、この属性はオプションです。その他のタイプの項目と image 属性が指定されていないカラーピッカーの場合、colorRect 属性は無視されます。colorRect 属性を指定すると、カラーピッカーで現在選択されているカラーがアイコンの左または上部を基準とした位置の四角形の中に表示されます。colorRect 属性を指定しない場合は、現在のカラーがイメージ上に表示されません。

例

```
colorRect="0 12 16 16"
```

file="command_file_path"

ポップアップメニューとコンボボックスの場合、この属性は必須です。その他のタイプの項目の場合、file 属性はオプションです。この属性で、項目の挿入、更新、実行を行う JavaScript 関数が格納されているコマンドファイルのパスを指定します。このパスは Configuration フォルダからの相対パスです。file 属性は、enabled 属性、checked 属性、value 属性、update

属性、domRequired 属性、menuID 属性、showIf 属性および command 属性よりも優先されます。一般に、file 属性でコマンドファイルを指定すると、タグで指定されている同等の属性はすべて無視されます。コマンドファイルについて詳しくは、181 ページの「ツールバーコマンド API 関数」を参照してください。

例

```
file="Toolbars/MM/EditTitle.htm"
```

domRequired="true" または "false"

オプション。メニューの場合と同様に、domRequired 属性で、関連するコマンドが実行される前にデザインビューをコードビューと同期させるかどうかを指定します。この属性を指定しない場合は、デフォルトの true が使用されます。この属性は、ツールバーコマンドファイル内の isDOMRequired() 関数と同じです。

例

```
domRequired="false"
```

enabled="script"

オプション。メニューの場合と同様に、スクリプトは項目が有効になっているかどうかを示す値を返します。この属性を設定しない場合は、デフォルトの「有効」が使用されます。enabled 属性は、ツールバーコマンドファイル内の canAcceptCommand() 関数と同じです。

例

```
enabled="dw.getFocus() =='textView' || dw.getFocus() == 'html'"
```

checked="script"

チェックボタンおよびラジオボタンの場合、この属性は必須です。その他のタイプの項目の場合、checked 属性は無視されます。メニューの場合と同様に、スクリプトは項目がオンまたはオフのどちらになっているかを示す値を返します。checked 属性は、ツールバーコマンドファイル内の isCommandChecked() と同じです。この属性を設定しない場合は、デフォルトの「オフ」が使用されます。

例

```
checked="dw.getDocumentDOM() != null && dw.getDocumentDOM().getView() == 'code'"
```

value="script"

ポップアップメニュー、コンボボックス、テキストボックスおよびカラーピッカーの場合、この属性は必須です。その他のタイプの項目の場合、value 属性は無視されます。

ポップアップメニューやコンボボックスに表示する値を決定するために、まずメニュー内の各項目に対して isCommandchecked() が呼び出されます。いずれかの項目に関して isCommandchecked() 関数が値 true を返すと、最初の項目の値が表示されます。どの項目についても値 true が返されない場合や、isCommandChecked() 関数が定義されていない場合は、getCurrentValue() 関数が呼び出されるか、value 属性で指定されているスクリプトが実行されます。コントロールがコンボボックスの場合は、返された値が表示されます。コントロールがポップアップメニューの場合は、返された値が一時的にリストに追加され、それが表示されます。

それ以外の場合は、表示する現在の値がスクリプトから返されます。ポップアップメニューまたはコンボボックスの場合、この値はメニューリスト内の項目のうちの 1 つです。コンボボックスおよびテキストボックスの場合、値はスクリプトが返すストリングです。カラーピッカーの場合、値は有効なカラーであることが必要ですが、これは強制ではありません。

value 属性は、ツールバーコマンドファイル内の getCurrentValue() 関数と同じです。

update="update_frequency_list"

オプション。この属性では、enabled、checked、showif および value の各ハンドラが項目の表示状態を更新するために実行される頻度を指定します。update 属性は、ツールバーコマンドファイル内の getUpdateFrequency() 関数と同じです。

メニュー項目とは異なり、ツールバー項目は常に表示されているので、ツールバー項目の更新頻度を指定する必要があります。そのため、常に可能な限り低い頻度を選択し、enabled、checked および value の各ハンドラを可能な限り単純にしてください。

次のリストに、**update_frequency_list** に指定できる最小頻度から最大頻度までのハンドラを示します。update 属性を指定しない場合、更新頻度にはデフォルトの onEdit 頻度が使用されます。複数の更新頻度をカンマで区切って指定することができます。ハンドラは、以下の指定されたイベントのいずれかが発生したときに実行されます。

- onServerModelChange は、現在のページのサーバモデルが変更されたときに実行されます。
- onCodeViewSyncChange は、コードビューとデザインビューが同期したとき、または同期しなくなったときに実行されます。
- onViewChange は、ユーザがコードビューとデザインビューの間でフォーカスを切り替えたときか、ユーザがコードビュー、デザインビューまたはコードとデザインビューの間で切り替えを行ったときに実行されます。
- onEdit は、デザインビューでドキュメントを編集するたびに実行されます。コードビューで行った変更内容を、このイベントのトリガにすることはできません。
- onSelChange は、デザインビューで選択範囲が変更されるたびに実行されます。コードビューで行った変更内容を、このイベントのトリガにすることはできません。
- onEveryIdle は、アプリケーションがアイドル状態のときに定期的に実行されます。enabler/checked/showif/value ハンドラは頻繁に実行されるため、この処理には時間がかかることがあります。この頻度は、特別な場合に有効状態を変更する必要のあるボタンにだけ使用してください。また、ハンドラの処理は短時間で終了する必要があります。

  注意：いずれの場合も、実際は、指定のイベントが発生した後のアプリケーションが静止状態のときにハンドラが実行されます。編集または選択範囲の変更を行うたびにハンドラが実行される保証はありません。つまり、ハンドラは編集または選択範囲の変更がある程度まとまってから実行されます。ユーザがツールバー項目をクリックすると、ハンドラは必ず実行されます。

例

```
update="onViewChange"
```

command="script"

この属性は、メニューボタン以外のすべての項目で必須です。メニューボタンの場合、command 属性は無視されます。この属性では、ユーザが以下のいずれかの操作を行ったときに実行される JavaScript 関数を指定します。

- ボタンをクリックする。
- ポップアップメニューまたはコンボボックスで項目を選択する。
- Tab キーを押してテキストボックスまたはコンボボックスから出るか、テキストボックスまたはコンボボックス内で Return キーを押すか、テキストボックスまたはコンボボックスの外部でクリックする。
- カラーピッカーからカラーを選択する。

command 属性は、ツールバーコマンドファイル内の receiveArguments() 関数と同じです。

例

```
command="dw.toggleLiveDebug()"
```

arguments="argument_list"

オプション。この属性では、ツールバーコマンドファイル内の receiveArguments() 関数に渡すカンマで区切られた引数のリストを指定します。arguments 属性を指定しない場合は、ツールバー項目の ID が渡されます。さらに、ポップアップメニュー、コンボボックス、テキストボックスおよびカラーピッカーの場合は、arguments 属性で指定されている引数や項目 ID（引数が指定されていない場合）の前に、それぞれの現在の値が最初の引数として渡されます。

例

「取り消し」ボタンと「やり直し」ボタンがあるツールバーでは、次の例で示すように、各ボタンでメニューコマンドファイル Edit_Clipboard.htm が呼び出され、アクションを指定する引数が渡されます。

```
<button id="DW_Undo"
 image="Toolbars/images/MM/undo.gif"
 disabledImage="Toolbars/images/MM/undo_dis.gif"
 tooltip="Undo"
 file="Menus/MM/Edit_Clipboard.htm"
 arguments="'undo'"
 update="onEveryIdle"/>
<button id="DW_Redo"
 image="Toolbars/images/MM/redo.gif"
 disabledImage="Toolbars/images/MM/redo_dis.gif"
 tooltip="Redo"
 file="Menus/MM/Edit_Clipboard.htm"
 arguments="'redo'"
 update="onEveryIdle"/>
```

ツールバーコマンド API 関数

属性のスクリプトを指定した場合、通常はコマンドファイル内の JavaScript 関数を使用して属性を実装することもできます。このアクションは、テキストボックスのコマンドハンドラのように、関数で引数を指定する必要がある場合に必要です。ポップアップメニューとコンボボックスの場合は必須です。

ツールバー項目のコマンドファイル API はメニューコマンドファイル API の拡張なので、メニューコマンドファイルをツールバーコマンドファイルとして直接再利用できます。その際にツールバーに固有の関数を追加することもあります。

canAcceptCommand()

対応バージョン

Dreamweaver MX

説明

ツールバー項目が有効になっているかどうかを判別します。有効状態が項目のデフォルトの状態であるため、値 false を返すケースが少なくとも 1 つある場合以外は、この関数を定義する必要はありません。

引数

ポップアップメニュー、コンボボックス、テキストボックスおよびカラーピッカーの場合、最初の引数はコントロール内の現在の値です。getDynamicContent() 関数は、必要に応じて個別の ID をポップアップメニュー内の項目に割り当てることができます。ポップアップメニュー内の選択した項目に ID が割り当てられている場合は、値の代わりにその ID が canAcceptCommand() に渡されます。コンボボックスの場合、テキストボックスの現在の内容がポップアップメニュー内のエントリと一致しないときは、テキストボックスの内容が渡されます。テキストボックスの内容がリスト内のエントリと一致するかどうかを判別するために、大文字と小文字の区別をせずにポップアップメニューとの比較が行われます。

toolbars.xml ファイルでこのツールバー項目の arguments 属性を指定している場合は、次にそれらの引数が渡されます。arguments 属性を指定していない場合は、項目の ID が渡されます。

戻り値
ブール値。項目が有効な場合は true、そうでない場合は false が返されます。

例
```
function canAcceptCommand()
{
    return (dw.getDocumentDOM() != null);
}
```

getCurrentValue()

対応バージョン
Dreamweaver MX

説明
項目内に表示される現在の値を返します。ポップアップメニュー、コンボボックス、テキストボックスおよびカラーピッカーの場合は、getCurrentValue() 関数が呼び出されます。ポップアップメニューの場合、現在の値はメニュー内の項目のうちの 1 つである必要があります。その値がポップアップメニュー内にない場合は、最初の項目が選択されます。コンボボックスおよびテキストボックスの場合、この値は関数が返すストリングです。カラーピッカーの場合、値は有効なカラーであることが必要ですが、これは強制ではありません。この関数は、value 属性と同じです。

引数
なし。

戻り値
表示する現在の値を含むストリング。カラーピッカーの場合は、選択したカラーの RGB 形式を含むストリング（白を表す #FFFFFF など）が返されます。

例
```
function getCurrentValue()
{
    var title = "";
    var dom = dw.getDocumentDOM();
    if (dom)
        title = dom.getTitle();
    return title;
}
```

getDynamicContent()

対応バージョン
Dreamweaver MX

説明
ポップアップメニューとコンボボックスの場合、この関数は必須です。メニューの場合と同様に、この関数はポップアップメニューに表示するストリングの配列を返します。各ストリングは、必要に応じて末尾を ";id=id" にすることができます。ID が指定されている場合は、メニューに表示される実際のストリングの代わりにその ID が receiveArguments() 関数に渡されます。

この関数は、メニュー内のエントリのリストが固定されている場合でも使用されるので、getDynamicContent() という名称は実態を表していません。例えば、Configuration¥Menus¥MM フォルダの Text_Size.htm ファイルは動的メニューではありません。このファイルは、一連の静的メニュー項目の各項目から呼び出されることを目的として設計されています。ただし、単に使用可能なフォントサイズのリストを返す getDynamicContent() 関数を追加することで、同じコマンドファイルをツールバーのポップアップメニューに使用することもできます。ツールバー項目では、返された配列内のストリングに含まれているアンダースコアが無視されるので、メニューコマンドファイルをそのまま利用できます。メニュー項目は動的とマークされていないので、メニューコマンドファイルでは getDynamicContent() 関数は無視されます。

引数
なし。

戻り値
メニューに表示されるストリングの配列。

例
```
function getDynamicContent()
{
    var items = new Array;
    var filename = dw.getConfigurationPath() + "/Toolbars/MM/AddressList.xml";
    var location = MMNotes.localURLToFilePath(filename);
    if (DWfile.exists(location))
    {
        var addressData = DWfile.read(location);
        var addressDOM = dw.getDocumentDOM(dw.getConfigurationPath() +
            '/Shared/MM/Cache/empty.htm');
        addressDOM.documentElement.outerHTML = addressData;
        var addressNodes = addressDOM.getElementsByTagName("url");
        if (addressNodes.length)
        {
            for (var i=0; i < addressNodes.length ; i++ )
            {
                items[i] = addressNodes[i].address + ";id='" +
                    addressNodes[i].address + "'";
            }
        }
    }
    return items;
```

getMenuID()

対応バージョン
Dreamweaver MX

説明
メニューボタンの場合にのみ有効です。getMenuID() 関数は、ユーザがボタンをクリックしたときに表示されるメニューの ID を取得するために呼び出されます。

引数
なし。

戻り値
menus.xml ファイルで定義されているメニュー ID を含むストリング。

例

```
function getMenuID()
{
    var dom = dw.getDocumentDOM();
    var menuID = '';
    if (dom)
    {
        var view = dom.getView();
        var focus = dw.getFocus();
        if (view == 'design')
        {
            menuID = 'DWDesignOnlyOptionsPopup';
        }
        else if (view == 'split')
        {
            if (focus == 'textView')
            {
                menuID = 'DWSplitCodeOptionsPopup';
            }
            else
            {
                menuID = 'DWSplitDesignOptionsPopup';
            }
        }
        else if (view == 'code')
        {
            menuID = 'DWCodeOnlyOptionsPopup';
        }
        else
        {
            menuID = 'DWBrowseOptionsPopup';
        }
    }
    return menuID;
}
```

getUpdateFrequency()

対応バージョン
Dreamweaver MX

説明
enabled 属性、checked 属性、showif 属性および value 属性のハンドラが項目の表示状態を更新するために実行される頻度を指定します。

メニューとは異なり、ツールバー項目は常に表示されているため、ツールバー項目の更新頻度を指定する必要があります。そのため、常に可能な限り低い頻度を選択し、enabled、checked および value の各ハンドラを可能な限り単純にしてください。

この関数は、ツールバー項目の update 属性と同じです。

引数
なし。

戻り値
カンマで区切られた更新ハンドラのリストを含むストリング。指定可能な更新ハンドラのリストについては、180 ページの「update="update_frequency_list"」を参照してください。

例
```
function getUpdateFrequency()
{
    return onSelChange";
}
```

isCommandChecked()

対応バージョン
Dreamweaver MX

説明
この関数は、項目が選択されているかどうかを示す値を返します。ボタンの場合、「Checked」はボタンがオンの状態、つまり押された状態で表示されていることを意味します。isCommandChecked() 関数は、ツールバー項目タグの checked 属性と同じです。

引数
ポップアップメニュー、コンボボックス、テキストボックスおよびカラーピッカーの場合、最初の引数はコントロール内の現在の値です。getDynamicContent() 関数は、必要に応じて個別の ID をポップアップメニュー内の項目に割り当てることができます。ポップアップメニュー内の選択した項目に ID が割り当てられている場合は、値の代わりにその ID が isCommandChecked() 関数に渡されます。コンボボックスの場合、テキストボックスの現在の内容がポップアップメニュー内のエントリと一致しないときは、テキストボックスの内容が渡されます。テキストボックスが一致するかどうかを判別するために、大文字と小文字の区別をせずにメニューとの比較が行われます。

arguments 属性を指定している場合は、次にそれらの引数が渡されます。arguments 属性を指定していない場合は、項目の ID が渡されます。

戻り値
ブール値。項目が有効な場合は true、そうでない場合は false が返されます。

例
次の例では、段落フォーマットと CSS スタイルのポップアップメニュー内でオンにする項目（存在する場合）を指定します。

```
function isCommandChecked()
{
    var bChecked = false;
    var style = arguments[0];
    var textFormat = dw.getDocumentDOM().getTextFormat();

    if (dw.getDocumentDOM() == null)
        bChecked = false;

    if (style == "(None)")
        bChecked = (dw.cssStylePalette.getSelectedStyle() == '' || textFormat ==
"" || textFormat == "P" || textFormat == "PRE");
    else if (style == "Heading 1")
        bChecked =0(textFormat == "h1");
    else if (style == "Heading 2")
        bChecked =0(textFormat == "h2");
    else if (style == "Heading 3")
        bChecked =0(textFormat == "h3");
    else if (style == "Heading 4")
        bChecked =0(textFormat == "h4");
    else if (style == "Heading 5")
        bChecked =0(textFormat == "h5");
    else if (style == "Heading 6")
        bChecked =0(textFormat == "h6");
    else
        bChecked = (dw.cssStylePalette.getSelectedStyle() == style);
    return bChecked;
}
```

isDOMRequired()

対応バージョン
Dreamweaver MX

説明
ツールバーコマンドの実行に有効な DOM が必要かどうかを指定します。この関数が値 true を返した場合、またはこの関数が定義されていない場合は、コマンドに有効な DOM が必要であると見なされ、ドキュメントのコードビューとデザインビューを同期させてから関連するコマンドが実行されます。この関数は、ツールバー項目タグの domRequired 属性と同じです。

引数
なし。

戻り値
ブール値。DOM が必要な場合は true、そうでない場合は false が返されます。

例

```
function isDOMRequired()
{
    return false;
}
```

receiveArguments()

対応バージョン
Dreamweaver MX

説明
ツールバー項目から渡された引数を処理します。receiveArguments() 関数は、ツールバー項目タグの command 属性と同じです。

引数
ポップアップメニュー、コンボボックス、テキストボックスおよびカラーピッカーの場合、最初の引数はコントロール内の現在の値です。getDynamicContent() 関数は、必要に応じて個別の ID をポップアップメニュー内の項目に割り当てることができます。ポップアップメニュー内の選択した項目に ID が割り当てられている場合は、値の代わりにその ID が receiveArguments() 関数に渡されます。コンボボックスの場合、テキストボックスの現在の内容がポップアップメニュー内のエントリと一致しないときは、テキストボックスの内容が渡されます。テキストボックスが一致するかどうかを判別するために、大文字と小文字の区別をせずにメニューとの比較が行われます。

arguments 属性を指定している場合は、次にそれらの引数が渡されます。arguments 属性を指定していない場合は、項目の ID が渡されます。

戻り値
なし。

例
```
function receiveArguments(newTitle)
{
    var dom = dw.getDocumentDOM();
    if (dom)
        dom.setTitle(newTitle);
}
```

showIf()

対応バージョン
Dreamweaver MX

説明
関数が値 true を返した場合にのみツールバーに項目が表示されるように指定します。例えば、showif() 関数を使用して、ページに特定のサーバモデルがある場合にのみ特定のボタンを表示することができます。showif() 関数が定義されていない場合、項目は常に表示されます。showIf() 関数は、ツールバー項目タグの showIf 属性と同じです。

showIf() 関数は、項目のイネーブラが動作するたびに呼び出されます。つまり、getUpdateFrequency() 関数が返す値に従って呼び出されます。

引数
なし。

戻り値
ブール値。項目が表示される場合は true、そうでない場合は false が返されます。

例

```
function showif()
{
        var retval = false;
        var dom = dw.getDocumentDOM();

        if(dom)
        {
            var view = dom.getView();
            if(view == 'design')
            {
                retval = true;
            }
        }
        return retval;
}
```

第 12 章：レポート

Adobe Dreamweaver では、サイトレポートおよびスタンドアローンレポートの 2 種類のレポートをサポートしています。

サイトレポート

レポート API を使用して、独自のサイトレポートの作成や、Dreamweaver に付属しているレポートの修正を実行できます。サイトレポートには、レポートダイアログボックスからのみアクセスできます。

サイトレポートは、Dreamweaver の Configuration¥Reports フォルダに格納されています。Reports フォルダには、レポートをカテゴリー別にまとめたサブフォルダがあります。レポートが分類されるカテゴリーは 1 つだけです。カテゴリー名の最大文字数は 31 文字です。各サブフォルダには、_foldername.txt というファイルが含まれている場合があります。このファイルがある場合、Dreamweaver では、このファイルのコンテンツがカテゴリー名として使用されます。_foldername.txt がない場合は、フォルダ名がカテゴリー名として使用されます。

レポートダイアログボックスで複数のサイトレポートを選択すると、すべての結果が結果パネルの「サイトレポート」タブの同じ結果ウィンドウに置かれます。次にサイトレポートを実行すると、これらの結果が置き換えられます。

次の表にサイトレポートを作成するために使用するファイルの一覧を示します。

パス	ファイル	説明
Configuration/Reports/{type/}	**reportname**.js	レポートの内容を生成する関数を含んでいます。
Configuration/Reports/{type/}	**reportname**.htm	適切な JavaScript ファイルを呼び出します。必要な場合は、レポートの、設定ダイアログボックスのユーザインターフェイス（UI）を定義します。
Configuration/Reports/	Reports.js	レポートの生成に使用される一般的な関数を提供します。

サイトレポートの動作

1. レポートには、サイト／レポートのコマンドでアクセスします。このコマンドを選択すると、指定のターゲットで実行するレポートを選択するためのダイアログボックスが表示されます。
2. ユーザは、レポート内容ポップアップメニューから、選択したレポートを実行するファイルを選択します。このメニューには、「現在のドキュメント」、「現在のローカルサイト全体」、「サイト内の選択したファイル」および「フォルダ」コマンドがあります。「フォルダ」コマンドを選択すると、フォルダを選択するための「参照」ボタンとテキストフィールドが表示されます。
3. パラメータ指定のあるレポートは、「設定」ボタンをクリックしてからパラメータに値を入力することによってカスタマイズできます。ユーザがレポートパラメータを設定できるようにするには、レポートに設定ダイアログボックスを組み込む必要があります。ただし、このダイアログボックスはオプションです。すべてのレポートに、ユーザによるパラメータ指定が要求されるわけではありません。レポートに設定ダイアログボックスがない場合は、ユーザがリストからレポートを選択したときに「設定」ボタンが淡色表示されます。
4. ユーザは、レポートを選択して設定を指定した後で、「実行」ボタンをクリックします。

   注意：レポートに preventFileActivity ハンドラがある場合、ユーザはこのレポートの実行中は他のファイル操作を実行することはできません。

この時点で、結果パネルの「サイトレポート」タブにあるすべてのアイテムがクリアされます。レポート処理が開始される前に、各レポートで beginReporting() 関数が呼び出されます。レポートがこの関数を使用して false という値を返した場合、そのレポートはレポート処理から除外されます。

5 各ファイルは、レポートダイアログボックスで processFile() 関数を使用して選択された個々のレポートに渡されます。結果リストにこのファイルの情報を出力する必要のあるレポートは、dw.resultsPalette.siteReports.addResultItem() 関数を呼び出します。この処理は、ユーザが選択したファイルがすべて処理されるか、ウィンドウの下にある「停止」ボタンがクリックされるまで続行されます。Dreamweaver には、処理中の各ファイルの名前と、これから処理される残りのファイルの数が表示されます。

6 すべてのファイルを処理してレポート処理が完了すると、各レポートで endReporting() 関数が呼び出されます。

簡単なサイトレポートの例

簡単な拡張機能の例では、特定のファイル、サイト全体、選択されたファイルまたはフォルダで参照されたすべてのイメージが一覧表示され、「Site Results」タブの結果ウィンドウのレポートに表示されます。

この拡張機能を作成するには、レポート定義を作成し、JavaScript コードを記述します。

この例では、レポートの定義を格納する List images.htm と、このレポートに固有の JavaScript コードを格納する List Images.js という 2 つのファイルを HTML Reports フォルダに作成します。さらに、Dreamweaver に付属の Reports.js ファイルを参照します。

レポート定義の作成

レポート定義では、レポートダイアログボックスに表示されるレポートの名前を指定し、必要な JavaScript ファイルをすべて呼び出し、必要に応じて設定ダイアログボックスのユーザインターフェイスを定義します。

1 Configuration/Reports/HTML Reports/List images.htm ファイルを作成します。

2 次のコードを追加して、HTML ページのタイトルのレポートダイアログボックスに表示するレポートの名前を指定します。

```
<html>
<head>
<title>List Images</title>
```

3 ファイルの末尾に script タグを追加し、src 属性で Reports.js ファイルを指定します。

```
<script src="../Reports.js"></script>
```

4 ファイルの末尾に、別の script タグを追加し、次に src 属性で作成する List Images.js ファイルを指定します。

```
<html>
<head>
<title>List Images</title>
<script src="../Reports.js"></script>
<script src="List Images.js"></script>
```

5 head タグを閉じ、body の開始タグと終了タグを含めて、html タグを閉じます。

```
</head>
<body>
</body>
</html>
```

6 ファイルを List Images.js として Configuration/Reports/HTML Reports フォルダに保存します。

JavaScript コードの記述

Dreamweaver には、Reports.js ファイルが用意されています。Reports.js の任意の関数を呼び出すことができます。しかし、独自のサイトレポートに固有のすべての関数を含む JavaScript ファイルも作成する必要があります。

1 次の内容の Configuration¥Reports¥HTML Reports¥List Images.js ファイルを作成します。

```
// Function: configureSettings
// Description: Standard report API, used to initialize and load
//the default values. Does not initialize the UI.
//
function configureSettings() {
    return false;
}
// Function: processFile
// Description: Report command API called during file processing.
//
function processFile (fileURL) {
    if (!isHTMLType(fileURL)) //If the file isn't an HTML file
        return;                     //skip it.
    var curDOM = dw.getDocumentDOM(fileURL);     // Variable for DOM
    var tagList = curDOM.getElementsByTagName('img'); // Variable for img tags
    var imgfilename; // Variable for file name specified in img tag
    for (var i=0; i < tagList.length; i++) { // For img tag list
        imgfilename = tagList[i].getAttribute('src'); // Get image filename
        if (imgfilename != null) { // If a filename is specified
        // Print the appropriate icon, HTML filename,
        // image filename, and line number
        reportItem(REP_ITEM_CUSTOM, fileURL, imgfilename,
                curDOM.nodeToSourceViewOffsets(tagList[i])); }
    }
}
```

2 ファイルに List Images.js という名前を付けて Configuration¥Reports¥HTML Reports フォルダに保存します。

スタンドアローンレポート

結果ウィンドウ API を使用すると、スタンドアローンレポートを作成できます。スタンドアローンレポートは、レポート API ではなく、結果ウィンドウ API を直接使用する標準コマンドです。スタンドアローンレポートには、他のコマンドと同様に、メニューまたは別のコマンドからアクセスします。

スタンドアローンレポートは、Dreamweaver の Configuration¥Commands フォルダに格納されています。スタンドアローンレポートのカスタムコマンドは、コマンドメニューに表示されます。

新しいスタンドアローンレポートを実行すると、新しい結果ウィンドウが毎回作成されます。

パス	ファイル	説明
Configuration/Commands	**commandname**.htm	ユーザがコマンドを選択したときに表示される UI を定義し、レポートを生成するために必要なアクションを含む JavaScript コードまたは JavaScript ファイルへの参照を含みます。
Configuration/Commands	**commandname.**js	結果ウィンドウを生成し、レポートをウィンドウに格納します。

スタンドアローンレポートの動作

1 レポートを生成するために作成したカスタムコマンドで dw.createResultsWindow() 関数を呼び出し、返された結果オブジェクトをウィンドウ変数に保存して、新しい結果ウィンドウを開きます。この処理の残りの関数は、このオブジェクトのメソッドとして呼び出されます。

2 カスタムコマンドで、結果ウィンドウオブジェクトのメソッドとして setTitle() 関数と SetColumnWidths() 関数を呼び出して、結果ウィンドウのタイトルとフォーマットを初期化します。

3 addItem() 関数を呼び出して結果ウィンドウへのアイテムの追加を直ちに開始するか、結果ウィンドウオブジェクトのメソッドとして setFileList() 関数と startProcessing() 関数を呼び出してファイルリスト内のファイルの繰り返し処理を開始します。

4 コマンドにより resWin.startProcessing() が呼び出されると、Dreamweaver がリスト内の各ファイル URL に対して processFile() 関数を呼び出します。processFile() 関数は、スタンドアローンのコマンドで定義します。この関数の引数は、ファイル URL のみです。結果ウィンドウオブジェクトの setCallbackCommands() 関数を使用すると、他のコマンドで processFile() 関数を呼び出すようにできます。

5 addItem() 関数を呼び出すには、スタンドアローンのコマンドによって作成された結果ウィンドウに processFile() 関数がアクセスできる必要があります。processFile() 関数では、結果ウィンドウオブジェクトの stopProcessing() 関数を呼び出して、ファイルリストの処理を停止することもできます。

簡単なスタンドアローンレポートの例

簡単なスタンドアローンレポートの拡張機能では、特定のファイルで参照されたすべてのイメージが一覧表示され、結果ウィンドウのレポートに表示されます。

この拡張機能を作成するには、ダイアログボックスの UI を作成し、JavaScript コードを記述します。

この例では、ユーザがカスタムコマンドを選択した際に表示されるダイアログボックスの UI を定義する List images.htm と、このレポートに固有の JavaScript コードを格納する Listimages.js という 2 つのファイルを Configuration¥Commands フォルダに作成します。

ダイアログボックスの UI の作成

HTML ファイルの body セクションは、ユーザがカスタムコマンドを選択して必要な JavaScript ファイルを呼び出した場合に表示されるダイアログボックスの内容を指定します。

1 Configuration/Commands/Listimages.htm ファイルを作成します。

2 Listimages.htm ファイルに次のコードを入力します。

```
<html>
<head>
<title>Standalone report example</title>
<script src="Listimages.js">
</script>
</head>
<body>
<div name="test">
<form name="myForm">
<table>
<tr>
<td>Click OK to display the standalone report.</td>
</tr>
</table>
</form>
</div>
</body>
```

3 ファイルに Listimages.htm という名前を付けて Configuration¥Commands フォルダに保存します。

JavaScript コードの記述

スタンドアローンレポートに固有のすべての関数を含む JavaScript ファイルを作成します。

1 次のコードを含む Listimages.js ファイルを Configuration¥Commands フォルダに作成します。

```
function stdaloneresultwin()
{
    var curDOM = dw.getDocumentDOM("document");
    var tagList = curDOM.getElementsByTagName('img');
    var imgfilename;
    var iOffset = 0;
    var iLineNumber = 0;

    var resWin = dw.createResultsWindow("Images in File", ["Line", "Image"]);

    for (var i=0; i < tagList.length; i++)
    {
        // Get the name of the source file.
            imgfilename = tagList[i].getAttribute('src');
        // Get the character offset from the start of the file
        // to the start of the img tag.
        iOffset = curDOM.nodeToOffsets(curDOM.images[i]);
        // Based on the offset, figure out what line in the file
        // the img tag is on.
        iLineNumber = curDOM.getLineFromOffset(iOffset[0]);
        // As long as the src attribute specifies a file name,
            if (imgfilename != null)
        { // display the line number, and image path.
            resWin.addItem(resWin, "0", "Images in Current File", null, ¬
                null, null, [iLineNumber, imgfilename]);
            }
    }
    return;
}

// add buttons to dialog
function commandButtons()
{
    return new Array("OK", "stdaloneresultwin()", "Cancel", "window.close()");
}
```

2 ファイルに Listimages.js という名前を付けて Configuration¥Commands フォルダに保存します。

レポート API 関数

レポート API では、必須の関数は、processFile() だけです。その他の関数はすべてオプションです。

processFile()

対応バージョン
Dreamweaver 4

説明
この関数は、処理するファイルがある場合に呼び出されます。「レポート」コマンドはファイルを修正せずに処理し、dw.ResultsPalette.SiteReports()、addResultItem() または resWin.addItem() 関数を使用して、そのファイルの情報を返します。この処理が終了すると、各ファイルの DOM が自動的に解放されます。

引数
strFilePath

strFilePath 引数は、処理するファイルのフルパスとファイル名です。

戻り値
なし。

beginReporting()

対応バージョン
Dreamweaver 4

説明
この関数は、レポート処理の開始時点で、レポートが実行される前に呼び出されます。「レポート」コマンドがこの関数を使用して false の値を返した場合は、「レポート」コマンドがレポートの実行から除外されます。

引数
target

target 引数は、レポートセッションのターゲットを指定するストリングです。"CurrentDoc"、"CurrentSite"、"CurrentSiteSelection"（サイトで選択したファイル用）または "Folder:+ ユーザが選択したフォルダのパス "（"Folder:c:temp" など）のいずれかの値を指定します。

戻り値
ブール値。true（レポートが正常に実行された場合）または false（レポートの実行から **target** が除外された場合）。

endReporting()

対応バージョン
Dreamweaver 4

説明
この関数は、レポート処理の最後に呼び出されます。

引数
なし。

戻り値
なし。

commandButtons()

対応バージョン
Dreamweaver 4

説明
オプションダイアログボックスの右側に表示されるボタンと、それらのボタンをクリックした際のビヘイビアを定義します。この関数が定義されていない場合、ボタンは表示されず、レポートファイルの body セクションがダイアログボックス全体に拡大表示されます。

引数
なし。

戻り値
偶数のエレメントを含む配列。最初のエレメントは、一番上のボタンのラベルを含むストリングです。2 番目のエレメントは、一番上のボタンがクリックされた際のビヘイビアを定義する JavaScript コードのストリングです。残りのエレメントにより、その他のボタンが同様に定義されます。

例
次の commandButtons() 関数の例では、「OK」、「キャンセル」および「ヘルプ」の 3 つのボタンを定義します。

```
function commandButtons(){
    return new Array("OK" , "doCommand()" , "Cancel" , ¬
    "window.close()" , "Help" , "showHelp()");
}
```

configureSettings()

対応バージョン
Dreamweaver 4

説明
このレポートが選択されたときに、レポートダイアログボックスで「レポート設定」ボタンを有効にするかどうかを決定します。

引数
なし。

戻り値
ブール値。「レポート設定」ボタンを有効にする必要がある場合は true、そうでない場合は false が返されます。

windowDimensions()

対応バージョン
Dreamweaver 4

説明

パラメータダイアログボックスに特定のサイズを設定します。この関数が定義されていない場合、ウィンドウのサイズは自動的に計算されます。

注意：640 x 480 ピクセル以上のオプションダイアログボックスを使用する場合に限り、この関数を定義するようにしてください。

引数

platform

platform 引数の値は、ユーザのプラットフォームに応じて、"macintosh" または "windows" になります。

戻り値

"widthInPixels,heightInPixels" という形式のストリング。

返されるサイズはダイアログボックス全体のサイズよりも小さくなります。これは「OK」と「キャンセル」に使用する領域が含まれていないためです。返されたサイズにすべてのオプションが収まらない場合は、スクロールバーが表示されます。

例

次の windowDimensions() 関数の例では、パラメータダイアログボックスのサイズを 648 x 520 ピクセルに設定します。

```
function windowDimensions(){
    return "648,520";
}
```

第 13 章：タグライブラリとタグエディタ

Dreamweaver では、各タグに関する情報（タグの属性を含む）が Configuration¥TagLibraries フォルダ内のサブフォルダに保存されます。タグエディタとタグ選択関数は、このフォルダに保存されている情報を使用して、タグの操作と編集を実行します。

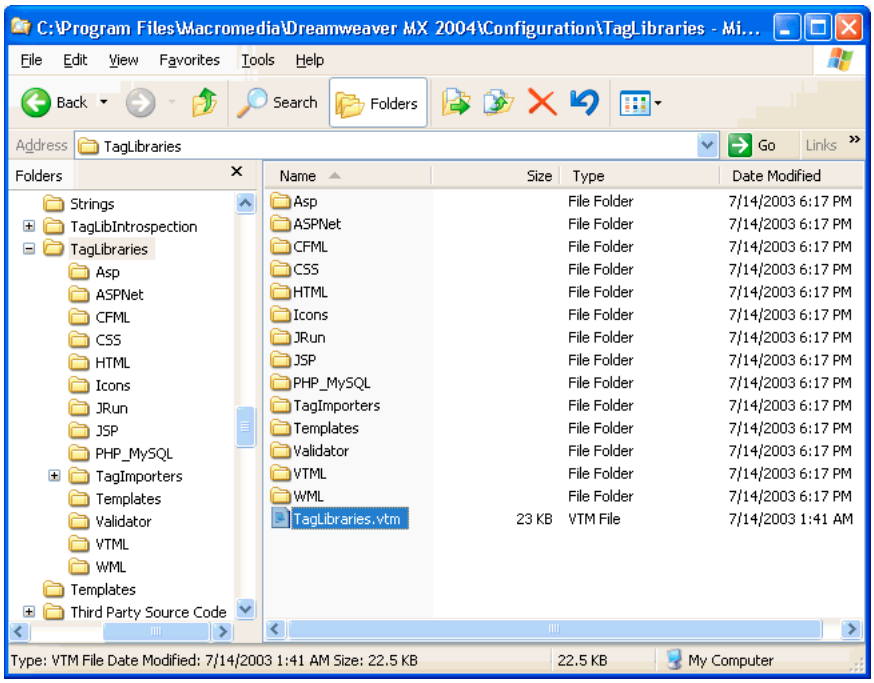
Dreamweaver には、HTML、ASP.NET、CFML、JRun および JSP の各言語に対応したエディタが用意されています。Dreamweaver 付属のタグエディタをカスタマイズすることも、新しいタグエディタを作成することもできます。また、タグライブラリに新しいタグを追加することもできます。

カスタムタグエディタを作成する前に、タグライブラリの構造について理解しておきます。次の表に、タグライブラリを作成するために使用するファイルの一覧を示します。

パス	**ファイル**	**説明**
Configuration¥TagLibraries¥	TagLibraries**.**vtm	インストールされているすべてのタグの一覧を表示します。
Configuration¥TagLibraries¥**language**¥	**tag.**vtm	タグに関する情報が記載されています（タグ属性、終了タグの有無、フォーマット規則など）。
Configuration¥TagLibraries¥**language**¥	**Tagimagefile**.gif	プロパティインスペクタに表示するオプションのファイルです。

タグライブラリのファイル形式

タグライブラリは、シングルルートファイル、インストールされているすべてのタグが列記された TagLibraries.vtm ファイルおよびタグライブラリの各タグの VTML ファイルで構成されています。TagLibraries.vtm ファイルは目次として機能し、個々のタグの VTML ファイルへのポインタを含んでいます。次の図は、マークアップ言語によって VTML ファイルが構成されている状態を示しています。



アドビ システムズ社の Macromedia HomeSite ユーザであれば、VTML ファイル構造であることがわかりますが、Dreamweaver では VTML ファイルの使用方法が HomeSite とは異なります。最大の相違点は、Dreamweaver には拡張ユーザインターフェイス（UI）を表示する独自の HTML レンダラーがあるため、ユーザインターフェイス（UI）レンダリングプロセスでは Dreamweaver の VTML ファイルが使用されないことです。

次の例は、TagLibraries.vtm ファイルの構造を示しています。

```
<taglibraries>
<taglibrary name="Name of tag library" doctypes="HTML,ASP-JS,ASP-VB" tagchooser="relative
    path to TagChooser.xml file" id="DWTagLibrary_html">
    <tagref name="tag name" file="relative path to tag .vtm file"/>
</taglibrary>
<taglibrary name="CFML Tags" doctypes="ColdFusion" servermodel="Cold Fusion"
    tagchooser="cfml/TagChooser.xml" id="DWTagLibrary_cfml">
    <tagref name="cfabort" file="cfml/cfabort.vtm"/>
</taglibrary>
<taglibrary name="ASP.NET Tags" doctypes="ASP.NET_CSharp,ASP.NET_VB" servermodel="ASPNet"
    prefix="<asp:" tagchooser="ASPNet/TagChooser.xml" id="DWTagLibrary_aspnet">
    <tagref name="dataset" file="aspnet/dataset.vtm" prefix="<mm:dataset"/>
</taglibrary>
</taglibraries>
```

taglibrary タグにより、タグが 1 つのタグライブラリにまとめられます。タグを読み込んだり、新しいタグを作成する際に、タグをタグライブラリにまとめることができます。通常、taglibrary のグループは、JavaServer Pages（JSP）TLD ファイル、XML document type definition（DTD）ファイル、ASP.NET ネームスペースまたは他の論理グループに定義されたタグのセットと一致します。

次の表は、taglibrary の属性のリストです。

属性	説明	必須 / オプション
taglibrary.name	UI でタグライブラリを参照するときに使用されます。	必須
taglibrary.doctypes	このライブラリがアクティブなドキュメントタイプを示します。ライブラリがアクティブなときは、ライブラリタグがコードヒントメニューに表示されます。名前の競合が発生する可能性があるため、すべてのタグライブラリを同時にアクティブにすることはできません。例えば、HTML ファイルと WML ファイルには互換性がありません。	必須
taglibrary.prefix	指定されると、タグライブラリ内のタグは taglibrary.prefix+ tagref.name. という形式になります。例えば、taglibrary.prefix が "<jrun:" で tagref.name が "if" の場合、タグの形式は "<jrun:if" となります。これは、特定のタグ用にオーバーライドすることができます。	オプション
taglibrary.servermodel	タグライブラリ内のタグがアプリケーションサーバで実行されると、servermodel 属性により、タグのサーバモデルが識別されます。タグがサーバ側のタグではなくクライアント側のタグの場合、servermodel 属性は省略されます。servermodel 属性は、ターゲットブラウザチェック機能でも使用されます。	オプション
taglibrary.id	ファイル内の他のタグライブラリの taglibrary.ID 属性とは異なる、任意のストリングを指定します。ID 属性は Extension Manager で使用されるため、MXP ファイルでは、新しい taglibrary ファイルと tags ファイルを TagLibraries.vtm ファイルに挿入できます。	オプション
taglibrary.tagchooser	このタグライブラリに関連付けられている TagChooser.xml ファイルへの相対パス。	オプション

次の表は、tagref 属性のリストです。

属性	説明	必須 / オプション
tagref.name	UI でのタグの参照に使用されます。	必須
tagref.prefix	コードビューでのタグの表示方法を指定します。tagref.prefix 属性を使用すると、現在のタグの接頭辞が指定されます。この属性を定義すると、この属性によって taglibrary.prefix 属性に指定された値がオーバーライドされます。	オプション
tagref.file	タグの VTML ファイルを参照します。	オプション

tagref.prefix 属性は taglibrary.prefix 属性をオーバーライドできるため、この 2 つの属性の関係で混乱が生じることがあります。次の表に、taglibrary.prefix 属性と tagref.prefix 属性の関係を示します。

taglibrary.prefix は定義されているか	tagref.prefix は定義されているか	タグの接頭辞
いいえ	いいえ	'<' + tagref.name
はい	いいえ	taglibrary.prefix + tagref.name
いいえ	はい	tagref.prefix
はい	はい	tagref.prefix

Dreamweaver では、VTML ファイル形式の修正バージョンを使用してタグが定義されます。次の例では、個別のタグの定義に必要なすべてのエレメントを示します。

```
<tag name="input" bind="value" casesensitive="no" endtag="no">
    <tagformat indentcontents="yes" formatcontents="yes" nlbeforetag nlbeforecontents=0
    nlaftercontents=0 nlaftertag=1 />
    <tagdialog file = "input.HTM"/>
    <attributes>
        <attrib name="name"/>
        <attrib name="wrap" type="Enumerated">
            <attriboption value="off"/>
            <attriboption value="soft"/>
            <attriboption value="hard"/>
        /attrib>
        <attrib name="onFocus" casesensitive="yes"/>
        <event name="onFocus"/>
    </attributes>
</tag>
```

次の表は、タグを定義する属性のリストです。

属性	説明	必須 / オプション
tag.bind	データバインディングパネルで使用されます。この種類のタグを選択すると、bind 属性により、データバインディングのデフォルトの属性が示されます。	オプション
tag.casesensitive	タグ名で大文字と小文字が区別されるかどうかを指定します。区別される場合は、タグライブラリで指定されたとおりの文字種（大文字または小文字）で、タグがユーザのドキュメントに挿入されます。区別されない場合は、環境設定ダイアログボックスの「コードフォーマット」タブで指定されているデフォルト（大文字または小文字）に従って、タグがドキュメントに挿入されます。casesensitive が省略された場合、タグでは大文字と小文字が区別されません。	オプション
tag.endtag	タグに開始タグと終了タグの両方があるかどうかを指定します。例えば、input タグには終了タグがなく、/input タグに相当するものはありません。終了タグがオプションの場合は、ENDTAG 属性を「Yes.tag」に設定する必要があります。空のタグに対して XML シンタックスを適用するには、xml を指定します。例えば、<tag name="foo" endtag="xml" tagtype="empty"> を指定すると、<foo/> が挿入されます。	オプション
tagformat	タグのフォーマット規則を指定します。Dreamweaver MX より以前の Dreamweaver では、この規則が SourceFormat.txt ファイルに保存されていました。	オプション
tagformat.indentcontents	このタグのコンテンツをインデントするかどうかを指定します。	オプション
tagformat.formatcontents	このタグのコンテンツを解析するかどうかを指定します。コンテンツが HTML ではないタグに対しては、この属性が No に設定されます（SCRIPT や STYLE などのタグ）。	オプション
tagformat.nlbeforetag	このタグの前に改行文字を挿入するかどうかを指定します。挿入しない場合は 0、挿入する場合は 1 を指定します。	オプション
tagformat.nlbeforecontents	このタグのコンテンツの前に挿入する改行文字の数です。	オプション
tagformat.nlaftercontents	このタグのコンテンツの後に挿入する改行文字の数です。	オプション

属性	説明	必須 / オプション
tagformat.nlaftertag	このタグの後に改行文字を挿入するかどうかを指定します。挿入しない場合は 0、挿入する場合は 1 を指定します。	オプション
attrib.name	ソースコードに表示される属性の名前です。	必須
attrib.type	省略された場合は、attrib.type に "text" が指定されます。指定可能な値は次のとおりです。TEXT（フリーテキストコンテンツ）、ENUMERATED（列挙値のリスト）、COLOR（名前または 16 進によるカラー値）、FONT（フォント名またはフォントファミリ）、STYLE（CSS スタイル属性）、CSSSTYLE（CSS クラス名）、CSSID（CSS クラス ID）、FILEPATH（完全なファイルパス）、DIRECTORY（フォルダパス）、FILENAME（ファイル名のみ）、RELATIVEPATH（相対的に表されたパス）、FLAG（値を含んでいない ON/OFF 属性）。	オプション
attrib.casesensitive	属性名で大文字と小文字が区別されるかどうかを指定します。CASESENSITIVE 属性が省略されている場合、属性名では大文字と小文字が区別されません。	オプション

注意：Dreamweaver MX よりも前のバージョンでは、タグ情報が Configuration¥TagAttributeList.txt ファイルに保存されます。

タグ選択

タグ選択を使用すると、機能グループ内のタグを表示して、使用頻度の高いタグに簡単にアクセスすることができます。タグ選択にタグまたはタグのセットを追加するには、追加するタグをタグライブラリに追加する必要があります。そのためには、タグライブラリエディタダイアログボックスを使用するか、または MXP ファイルにパッケージされている Dreamweaver 拡張機能をインストールします。

TagChooser.xml ファイル

TagChooser.xml ファイルには、タグ選択に表示されるタググループを編成するためのメタデータが含まれています。Dreamweaver 付属の各タグは機能グループに保存され、タグ選択で使用できるようになっています。TagChooser.xml ファイルを編集することにより、既存のタグを再グループ化したり、新しいタグをグループにまとめたりすることができます。サブカテゴリーを作成してタグの編成方法をカスタマイズすると、ユーザにとって最も重要なタグに簡単にアクセスできるようになります。

TagLibraries.vtm ファイルでは、TagChooser.xml ファイルをポイントする taglibrary.tagchooser 属性を使用できます。既存の TagChooser.xml ファイルを変更するか、新規のファイルを作成した場合は、タグ選択が正常に機能するように、taglibrary.tagchooser 属性で正しい位置をポイントする必要があります。

taglibrary.tagchooser 属性がない場合は、タグ選択により TagLibraries.vtm ファイル内のツリー構造が表示されます。

TagChooser.xml ファイルは Configuration¥TagLibraries¥**TagLibraryName** フォルダに格納されています。次の例では、TagChooser.xml ファイルの構造を示します。

```
<?xml version="1.0" encoding="iso-8859-1" standalone="yes" ?>
<tclibrary name="Friendly name for library node" desc='Description for incorporated
    reference' reference="Language[,Topic[,Subtopic]]">
    <category name="Friendly name for category node" desc='Description for incorporated
    reference' reference="Language[,Topic[,Subtopic]]" id="Unique id">
        <category name="Friendly name for subcategory node" ICON="Relative path"
        desc='Description for incorporated reference' reference="Language,Topic[,Subtopic]"
        id="Unique id">
            <element name="Friendly name for list item" value='Value to pass to visual dialog
            editors' desc='Description for incorporated reference'
            reference="Language[,Topic[,Subtopic]]" id="Unique id"/>
            ... more elements to display in the list view ...
        </category>
        ... more subcategories ...
    </category>
    ... more categories ...
</tclibrary>
```

次の表は、TagChooser.xml ファイルで使用可能なタグのリストです。

タグ	説明	必須 / オプション
tclibrary	最も外側のタグで、このタグライブラリのタグ選択の構造をカプセル化します。	必須
tclibrary.name	ツリービューノードに表示される値です。	必須
tclibrary.desc	タグ選択ダイアログボックスの「タグ情報」セクションに表示される HTML ストリングです。desc 属性がない場合、「タグ情報」セクションにはリファレンスパネルの情報が表示されます。tclibrary.reference と交換可能です。	オプション（desc と reference は互いに排他的）
tclibrary.reference	タグ選択ダイアログボックスの「タグ情報」セクションに表示される、言語、トピックおよびサブトピックを記述する値です。tclibrary.desc と交換可能です。	オプション（desc と reference は互いに排他的）

次の表に示すように、category タグは、tclibrary のノードの下にあるツリービュー内の他のすべてのノードを表します。

タグ	説明	必須 / オプション
category.name	ツリービューノードに表示される値です。	必須
category.desc	タグ選択ダイアログボックスの「タグ情報」セクションに表示される HTML ストリングです。desc と reference attr のどちらも指定されていない場合、「タグ情報」セクションには何も表示されません。	オプション（desc と reference は互いに排他的）
category.reference	「タグ情報」セクションに表示される、言語、トピックおよびサブトピックを記述する値です。	オプション（desc と reference は互いに排他的）
category.icon	GIF アイコンへの相対パスです。	オプション
category.id	このファイル内の他のカテゴリーの category.id 属性と異なる任意のストリングです。	必須

次の表は、挿入するタグを表す element タグの属性のリストです。

属性	説明	必須 / オプション
element.name	リストビュー項目として表示される値です。	必須
element.value	コードに直接配置される値、またはビジュアルダイアログボックスに渡されるパラメータです。	必須
element.desc	組み込まれたリファレンスパネルに表示される HTML ストリングです。この値が指定されていない場合、組み込まれたリファレンスパネルには、reference 属性によってリファレンスコンテンツが表示されます。	オプション（desc と reference は互いに排他的）
element.reference	言語、トピック、サブトピックをそれぞれ記述する、カンマで区切られた 3 つのストリングです。この情報はリファレンスパネルに表示されます。最初のストリングは必須です。2 番目のストリングは element タグでのみ必須で、category タグおよび tclibrary タグではオプションです。3 番目のストリングはオプションです。	オプション（desc と reference は互いに排他的）
element.id	このファイル内の他のエレメントの element.id 属性と異なる任意のストリングです。	オプション

新しいタグエディタを作成する簡単な例

この節の例では、天候データベースから現在の温度を抽出する仮想の Adobe ColdFusion タグ cfweather を使用して、新しいタグエディタの作成に必要な手順を示します。

次の表では、cfweather タグの属性について説明します。

属性	説明
zip	5 桁の郵便番号
tempaturescale	温度の単位（摂氏または華氏）

この新しいタグエディタを作成するには、タグを登録し、タグ定義を作成し、タグエディタ UI を作成し、タグをタグ選択に追加します。

タグのタグライブラリへの登録

Dreamweaver で新しいタグが認識されるには、そのタグが Configuration¥TagLibraries フォルダにある TagLibraries.vtm ファイルで識別される必要があります。ただし、Windows XP、Windows 2000、Windows NT、Mac OS X などのマルチユーザプラットフォームで作業をしている場合は、ユーザの Configuration フォルダに別の TagLibraries.vtm ファイルがあります。このファイルは Dreamweaver が検索および解析するインスタンスであるため、更新する必要があります。

ユーザの Configuration フォルダの場所は、使用するプラットフォームによって異なります。

Windows 2000 および Windows XP プラットフォームの場合

```
<drive>:\Documents and Settings\<username>\Application Data\Adobe\\xc2
Dreamweaver CS4\Configuration
```

注意：Windows XP では、このフォルダが隠しフォルダ内に存在する場合があります。

Mac OS X プラットフォームの場合

```
<drive>:Users:<username>:Library:Application Support:Adobe:¬
Dreamweaver CS4:Configuration
```

ユーザの Configuration フォルダで TagLibraries.vtm ファイルが見つからない場合は、Dreamweaver の Configuration フォルダ内でファイルが検索されます。

注意：マルチユーザプラットフォームでは、Dreamweaver の Configuration フォルダにある TagLibraries.vtm のコピーを編集しても、その変更が Dreamweaver によって認識されません。Dreamweaver では、Dreamweaver の Configuration フォルダではなく、ユーザの Configuration フォルダにある TagLibraries.vtm ファイルのコピーが解析されます。

cfweather タグは ColdFusion タグなので、cfweather タグを登録する適切なタグライブラリグループが既に存在します。

1 テキストエディタで TagLibraries.vtm ファイルを開きます。

2 既存のタグライブラリをスクロールして、CFML タグを検索します。

3 次の例に示すように、新しいタグリファレンスエレメントを追加します。

```
<tagref name="cfweather" file="cfml/cfweather.vtm"/>
```

4 ファイルを保存します。

これで、タグがタグライブラリに登録されました。登録されたタグには、cfweather.vtm タグ定義ファイルへのファイルポインタがあります。

タグ定義（VTML）ファイルの作成

タグ選択またはタグエディタで登録済みのタグを選択すると、タグ定義用の対応する VTML ファイルが検索されます。

1 テキストエディタで、次のコンテンツのファイルを作成します。

```
<TAG NAME="cfweather" endtag="no">
            <TAGFORMAT NLBEFORETAG="1" NLAFTERTAG="1"/>
            <TAGDIALOG FILE="cfweather.htm"/>

            <ATTRIBUTES>
                <ATTRIB NAME="zip" TYPE="TEXT"/>
                <ATTRIB NAME="tempaturescale" TYPE="ENUMERATED">
                    <ATTRIBOPTION VALUE="Celsius"/>
                    <ATTRIBOPTION VALUE="Fahrenheit"/>
                </ATTRIB>
            </ATTRIBUTES>
</TAG>
```

2 cfweather.vtm ファイルを Configuration¥TagLibraries¥CFML フォルダに保存します。

Dreamweaver では、タグ定義ファイルを使用することにより、cfweather タグに対してコードヒント、コードの完成、タグフォーマットの各機能を実行できます。

タグエディタのユーザインターフェイスを作成する

1 cfweather.htm ファイルを Configuration¥TagLibraries¥CFML フォルダに保存します。

```
<!DOCTYPE HTML SYSTEM "-//Adobe//DWExtension layout-engine 10.0//dialog">
<html>
<head>
<title>CFWEATHER</title>
<script src="../../Shared/Common/Scripts/dwscripts.js"></script>
<script src="../../Shared/Common/Scripts/ListControlClass.js"></script>
<script src="../../Shared/Common/Scripts/tagDialogsCmn.js"></script>
<script>

/************************ GLOBAL VARS *************************/
var TEMPATURESCALELIST;    // tempaurelist control (initialized in initializeUI())
var theUIObjects;          // array of UI objects used by common API functions
/**************************************************************/

// inspectTag() API function defined (required by all tag editors)
function inspectTag(tagNodeObj)
{
    // call into a common library version of inspectTagCommon defined
    // in tagDialogCmns.js (note that it's been included)
    // For more information about this function, look at the comments
    // for inspectTagCommon in tagDialogCmn.js
    tagDialog.inspectTagCommon(tagNodeObj, theUIObjects);
}
function applyTag(tagNodeObj)
{
    // call into a common library version of applyTagCommon defined
    // in tagDialogCmns.js (note that it's been included)
    // For more information about this function, look at the comments
    // for applyTagCommon in tagDialogCmn.js
    tagDialog.applyTagCommon(tagNodeObj, theUIObjects);
}
function initializeUI()
{
    // define two arrays for the values and display captions for the list
    control
    var theTempatureScaleCap = new Array("celsius","fahrenheit");
    var theTempatureScaleVal = new Array("celsius","fahrenheit");

    // instantiate a new list control
    TEMPATURESCALELIST = new ListControl("thetempaturescale");

    // add the tempaturescalelist dropdown list control to the uiobjects
    theUIObjects0= new Array(TEMPATURESCALELIST);

    // call common populateDropDownList function defined in tagDialogCmn.js to
    // populate the tempaturescale list control
    tagDialog.populateDropDownList(TEMPATURESCALELIST, theTempatureScaleCap,
    theTempatureScaleVal, 1);
}
</script>

</head>
<body onLoad="initializeUI()">
<div name="General">
```

```
    <table border="0" cellspacing="4">
        <tr>
            <td valign="baseline" align="right" nowrap="nowrap">Zip Code: </td>
            <td nowrap="nowrap">
                <input type="text" id="attr:cfargument:zip" name="thezip" attname="zip"
                style="width:100px"0/> 
            </td>
        </tr>
        <tr>
            <td valign="baseline" align="right" nowrap="nowrap">Type: </td>
            <td nowrap="nowrap">
                <select name="thetempaturescale" id="attr:cfargument:tempaturescale"
                attname="tempaturescale" editable="false" style="width:200px">
                </select>
            </td>
        </tr>
    </table>
</div>
</body>
</html>
```

次に、タグエディタが動作することを確認します。

2 Dreamweaver を起動します。

3 コードビューで「**cfweather**」と入力します。

4 タグを右クリックします。

5 コンテキストメニューから、cfweather 編集タグを選択します。

タグエディタが起動すれば、タグエディタは正常に作成されています。

タグ選択へタグを追加する

1 次の例で示すように、cfweather タグを含む Third Party Tags という新しいカテゴリーを追加することで、Configuration¥TagLibraries¥CFML フォルダの TagChooser.xml ファイルを修正します。

```
<category name="Third Party Tags" icon="icons/Elements.gif" reference='CFML'>
    <element name="cfweather" value='cfweather zip="" temperaturescale="fahrenheit">' />
</category>
```

注意：マルチユーザプラットフォームでは、ユーザの Configuration フォルダにも TagChooser.xml ファイルがあります。マルチユーザプラットフォームについて詳しくは、203 ページの「タグのタグライブラリへの登録」を参照してください。

次に、cfweather タグがタグ選択に表示されることを確認します。

2 挿入／タグを選択します。

3 「CFML タグ」グループを展開します。

4 タグ選択の一番下に表示される「Third Party Tags」グループを選択します。右側のリストボックスに cfweather タグが表示されます。

5 cfweather を選択し、「挿入」ボタンをクリックします。

タグエディタが表示されます。

タグエディタ API 関数

新しいタグエディタを作成するには、inspectTag()、validateTag()、applyTag() の各関数を実装する必要があります。実装例については、204 ページの「タグエディタのユーザインターフェイスを作成する」を参照してください。

inspectTag()

対応バージョン
Dreamweaver MX

説明
この関数は、タグエディタが最初に表示されるときに呼び出されます。呼び出された関数は、引数として、ユーザが編集しているタグを dom オブジェクトで受け取ります。この関数は編集中のタグから属性値を抽出し、その属性値を使用してタグエディタでフォームエレメントを初期化します。

引数
tag

- tag 引数は、編集されたタグの DOM ノードです。

戻り値
なし。

例
ユーザが次のタグを編集するとします。

```
<crfweather zip = "94065"/>
```

エディタに zip 属性を編集するためのテキストフィールドがある場合は、そのフィールドに空白ではなく実際の郵便番号が表示されるように、フォームエレメントを初期化する必要があります。

この初期化を実行するコードは次のとおりです。

```
function inspectTag(tag)
{
    document.forms[0].zip.value = tag.zip
}
```

validateTag()

対応バージョン
Dreamweaver MX

説明
ユーザがツリーコントロール内のノードをクリックするか、「OK」をクリックすると、この関数によって、現在表示されている HTML フォームエレメントで入力の検証が実行されます。

引数
なし。

戻り値
ブール値。HTML フォームエレメントへの入力値が有効な場合は true、入力値が無効な場合は false。

例
ユーザが、テーブルを作成するときにテーブルの行数として負の整数を入力したとします。validateTag() 関数によって、無効な入力が検出され、警告メッセージが表示され、false 値が返されます。

applyTag()

対応バージョン
Dreamweaver MX

説明
ユーザが「OK」をクリックすると、validateTag() 関数が呼び出されます。validateTag() 関数から値 true が返されると、この関数が呼び出され、現在編集中のタグを表す dom オブジェクトが渡されます。この関数により、フォームエレメントから値が読み取られ、その値が dom オブジェクトに書き込まれます。

引数
tag

- tag 引数は、編集中のタグの DOM ノードです。

戻り値
なし。

例
cfweather の例の続きとして、次のコードでは、ユーザが郵便番号を 94065 から 53402 に変更した場合に、ユーザのドキュメントを更新して新しい郵便番号が使用されるようにするために、dom オブジェクトを更新します。

```
function applyTag(tag)
{
    tag.zip = document.forms[0].zip.value
}
```

第 14 章：プロパティインスペクタ

プロパティインスペクタは、Dreamweaver のインターフェイスで最も頻繁に使用されるフローティングパネルです。プロパティインスペクタは、選択対象の名前、サイズ、外観などの属性を定義、参照、変更するために欠かせない機能です。さらに、選択した要素を編集するために内部エディタや外部エディタを起動することもできます。

次の表に、プロパティインスペクタを作成するために使用するファイルの一覧を示します。

パス	ファイル	説明
Configuration¥Inspectors¥	**Propertyinspectornam**e.htm	プロパティインスペクタのユーザインターフェイス（UI）を定義します。
Configuration¥Inspectors¥	**Propertyinspectorname**.js	プロパティインスペクタに必要な関数が含まれています。
Configuration¥Inspectors¥	**Tagimagefile**.gif	プロパティインスペクタに表示するオプションのファイルです。

プロパティインスペクタファイル

Dreamweaver には、既にプロパティインスペクタ用のインターフェイスがいくつか組み込まれており、これにより多くの標準 HTML タグにプロパティを設定することができます。これらの組み込みインスペクタは、コア Dreamweaver コードの一部であるため、Configuration フォルダ内には、対応するプロパティインスペクタファイルは見つかりません。しかし、カスタムプロパティインスペクタファイルを使用して、組み込まれているインターフェイスをオーバーライドしたり、カスタムタグ用の新しいインスペクタを作成することができます。カスタムプロパティインスペクタファイルは、Dreamweaver アプリケーションフォルダ内の Configuration¥Inspectors フォルダに保存されているファイルです。

プロパティインスペクタ HTML ファイルでは、HTML の開始タグの直前にコメント（doctype コメントも含む）が必要です。次に例を示します。

```
<!-- tag:serverModel:tagNameOrKeyword,priority:1to10,selection:¬ exactOrWithin,hline,vline, serverModel-->
<!DOCTYPE HTML SYSTEM "-//Adobe//DWExtension layout-engine 10.0//pi">
```

このコメントには、以下のエレメントが含まれます。

- ***serverModel*** エレメントは、指定されたサーバモデルがアクティブになっている場合にのみ、このプロパティインスペクタを読み込む必要があることを示します。
- tagNameOrKeyword エレメントには、検査対象のタグか、*COMMENT*（コメント用）、*LOCKED*（ロックされた領域用）または *ASP*（ASP タグ用）のいずれかのキーワードを指定します。
- 1to10 エレメントは、プロパティインスペクタファイルの優先度を示します。1 を指定すると、選択範囲を検査できるインスペクタが他にない場合にのみ、このインスペクタが使用されます。10 を指定すると、選択範囲を検査できる他のすべてのインスペクタよりも、このインスペクタが優先されます。
- ***exactOrWithin*** エレメントは、選択範囲がタグの内側でもよいか（within）、または確実にタグを含む必要があるか（exact）を示します。
- ***hline*** エレメント（オプション）は、拡張モードのときにインスペクタの上部と下部を分けるグレーの横線を表示するかどうかを示します。
- vline エレメント（オプション）は、インスペクタのタグ名フィールドと他のプロパティを分けるグレーの縦線を表示するかどうかを示します。

- ***serverModel*** エレメント（オプション）は、プロパティインスペクタのサーバモデルを示します。プロパティインスペクタのサーバモデルは、ドキュメントのプロパティインスペクタと同じである必要があります。異なる場合、プロパティインスペクタは現在の選択範囲のプロパティを表示するために使用されません。例えば、ドキュメントのサーバモデルが Adobe ColdFusion であるとします。プロパティインスペクタのサーバモデルは ASP です。この場合、ドキュメント内の選択範囲に対してプロパティインスペクタは使用されません。

例えば happy タグを検査するインスペクタの場合、次のようなコメントを使用します。

```
<!-- tag:happy, priority:8,selection:exact,hline,vline,serverModel:ASP -->
```

場合によっては、Dreamweaver 拡張機能のレンダリングだけを使用し、以前のレンダリングエンジンを使用しないように指定する必要があります。そのためには、次の例に示すように、タグコメントの直前に以下の行を挿入します。

```
<!--DOCTYPE HTML SYSTEM "-//Adobe//DWExtension layout-engine 10.0//pi"-->
```

プロパティインスペクタファイルの body セクションには、HTML フォームが含まれています。ただし、Dreamweaver はダイアログボックスにフォームの内容を表示するのではなく、フォームを使用してプロパティインスペクタの入力領域とレイアウトを定義します。

プロパティインスペクタファイルの head セクションには、JavaScript 関数または JavaScript ファイルへの参照が含まれています。

プロパティインスペクタファイルの機能

Dreamweaver を起動すると Configuration¥Inspectors フォルダにある各 HTM ファイルと HTML ファイルの最初の行が読み取られ、プロパティインスペクタのタイプ、優先度および選択タイプを定義するコメントストリングが検索されます。最初の行がこのコメントでないファイルは無視されます。

ユーザが Dreamweaver 内で範囲を選択したり、別の場所に挿入ポイントを移動すると、次のようなイベントが発生します。

1. 選択タイプが within であるインスペクタが検索されます。
2. 選択タイプが within のインスペクタが見つかると、現在選択されているタグから上方向にドキュメントツリーが検索され、この選択範囲を囲むタグにインスペクタがあるかどうかがチェックされます。within インスペクタがない場合、選択タイプが exact のインスペクタが検索されます。
3. インスペクタを含む最初のタグごとに、各インスペクタの canInspectSelection() 関数が呼び出されます。この関数が値 false を返した場合、そのインスペクタは現在の選択範囲に対して適用不可と見なされます。
4. canInspectSelection() 関数を呼び出した後に、適用可能なインスペクタが複数残った場合、これらのインスペクタが優先度でソートされます。
5. 同じ優先度を持つインスペクタが複数ある場合、インスペクタは名前のアルファベット順で選択されます。
6. 選択されたインスペクタは、プロパティインスペクタフローティングパネルに表示されます。プロパティインスペクタファイルに displayHelp() 関数が定義されている場合は、小さい疑問符（?）アイコンがインスペクタの右上隅に表示されます。
7. inspectSelection() 関数が呼び出され、現在の選択範囲に関する情報が収集され、インスペクタのフィールドに値が取り込まれます。
8. プロパティインスペクタのインターフェイスの各フィールドに関連付けられたイベントハンドラは、各フィールドで入力が行われると実行されます。例えば、onBlur イベントは、setAttribute() 関数を呼び出して、ユーザが入力した値に属性を設定します。

簡単なプロパティインスペクタの例

次のプロパティインスペクタは、Microsoft Internet Explorer でのみ使用可能な marquee タグを検査します。この例では、プロパティインスペクタで direction 属性の値を設定します。marquee タグの他の属性値を設定するには、この例をモデルとして使用します。

この拡張機能を作成するには、ユーザインターフェイスを作成し、JavaScript コードを記述し、イメージを作成し、テストします。

ユーザインターフェイスの作成

プロパティインスペクタに表示されるフォームを含む HTML ファイルを作成します。

1 新しい空白のファイルを作成します。

2 ファイルの最初の行に、プロパティインスペクタを特定する次のコメントを追加します。

```
<!-- tag:MARQUEE,priority:9,selection:exact,vline,hline -->
```

3 ドキュメントのタイトルと作成する JavaScript ファイルを指定するために、次のコードをコメントの後に追加します。

```
<HTML>
<HEAD>
<TITLE>Marquee Inspector</TITLE>
<SCRIPT src="marquee.js"></SCRIPT>
</HEAD>
<BODY>

</BODY>
</HTML>
```

4 プロパティインスペクタに表示される内容を指定するには、body の開始タグと終了タグの間に以下を追加します。

```
<!-- Specify the image that will appear in the Property inspector -->
<SPAN ID="image" STYLE="position:absolute; width:23px; height:17px; ¬
    z-index:16; left: 3px; top: 2px">
    <IMG SRC="marquee.png" WIDTH="36" HEIGHT="36" NAME="marqueeImage">
</SPAN>
<SPAN ID="label" STYLE="position:absolute; width:23px; height:17px; ¬
    z-index:16; left: 44px; top: 5px">Marquee</SPAN>

<!-- If your form fields are in different AP elements, you must ¬
    create a separate form inside each AP element and reference it as ¬
    shown in the inspectSelection() and setInterjectionTag() functions. -->

<SPAN ID="topLayer" STYLE="position:absolute; z-index:1; left: 125px; ¬
    top: 3px; width: 431px; height: 32px">
<FORM NAME="topLayerForm">
    <TABLE BORDER="0" CELLPADDING="0" CELLSPACING="0">
        <TR>
            <TD VALIGN="baseline" ALIGN="right">Direction:</TD>
            <TD VALIGN="baseline" ALIGN="right">
            <SELECT NAME="marqDirection" STYLE="width:86"
                onChange="setMarqueeTag()">
                <OPTION VALUE="left">Left</OPTION>
                <OPTION VALUE="right">Right</OPTION>
            </SELECT>
        </TR>
    </TABLE>
</FORM>
</SPAN>
```

5 ファイルに marquee.htm という名前を付けて Configuration¥Inspectors フォルダに保存します。

JavaScript コードの記述

JavaScript 関数を追加して、選択範囲を調べることができること、およびプロパティインスペクタに適切な値が入力されることを確認する必要があります。

1 新しい空白のファイルを作成します。

2 選択範囲に marquee タグが含まれる場合に必ずプロパティインスペクタが表示されるように指定するには、次の関数を追加します。

```
function canInspectSelection(){
    return true;
}
```

3 テキストフィールドに表示される direction 属性を更新するには、ファイルの末尾に次の関数を追加します。

```
function inspectSelection(){
    // Get the DOM of the current document.
    var theDOM = dw.getDocumentDOM();
    // Get the selected node.
    var theObj = theDOM.getSelectedNode();

    // Get the value of the DIRECTION attribute on the MARQUEE tag.
    var theDirection = theObj.getAttribute('direction');

    // Initialize a variable for the DIRECTION attribute to -1.
    // This is used to store the menu index that corresponds to
    // the value of the attribute.
    // var typeIndex = -1;
    var directionIndex = -1;

    // If there was a DIRECTION attribute...
    if (theDirection){
        // If the value of DIRECTION is "left", set typeIndex to 0.
        if (theDirection.toLowerCase() == "left"){
            directionIndex = 0;
        // If the value of DIRECTION is "right", set typeIndex to 1.
            }else if (theDirection.toLowerCase() == "right"){
                directionIndex = 1;
                }
            }
    // If the value of the DIRECTION attribute was "left"
    // or "right", choose the corresponding
    // option from the pop-up menu in the interface.
    if (directionIndex != -1){
        document.topLayer.document.topLayerForm.marqDirection.selectedIndex =¬
        directionIndex;
    }
}
```

4 現在の選択範囲を取得して、プロパティインスペクタのテキストボックスに direction 属性の値を表示するには、ファイルの末尾に次の関数を追加します。

```
function setMarqueeTag(){
    // Get the DOM of the current document.
    var theDOM = dw.getDocumentDOM();
    // Get the selected node.
    var theObj = theDOM.getSelectedNode();

    // Get the index of the selected option in the pop-up menu
    // in the interface.
    var directionIndex = ¬
        document.topLayer.document.topLayerForm.marqDirection.selectedIndex;
    // Get the value of the selected option in the pop-up menu
    // in the interface.
    var theDirection = ¬
        document.topLayer.document.topLayerForm.marqDirection.¬
        options[directionIndex].value;

    // Set the value of the direction attribute to theDirection.
    theObj.setAttribute('direction',theDirection);
}
```

5 ファイルに marquee.js という名前を付けて Configuration¥Inspectors フォルダに保存します。

イメージの作成

必要に応じて、プロパティインスペクタに表示されるイメージを作成することができます。

1 幅と高さがそれぞれ 36 ピクセルのイメージを作成します。

2 イメージに marquee.gif という名前を付けて Configuration¥Inspectors フォルダに保存します。

一般的に、Dreamweaver によってサポートされている形式であれば、任意の形式でプロパティインスペクタのイメージを保存することができます。

プロパティインスペクタのテスト

最後に、プロパティインスペクタをテストします。

1 Dreamweaver を再起動します。

2 新しい HTML ページを作成するか、既存の HTML ページを開きます。

3 以下をページの body セクションに追加します。

```
<MARQUEE></MARQUEE>
```

4 追加したテキストをハイライト表示します。

marquee タグに対して作成したプロパティインスペクタが表示されます。

5 プロパティインスペクタで、direction 属性の値を入力します。

ページのタグが変更され、direction 属性およびプロパティインスペクタで入力した値が含まれます。

プロパティインスペクタ API 関数

必須のプロパティインスペクタ API 関数は、canInspectSelection() と inspectSelection() の 2 つです。

canInspectSelection()

説明
プロパティインスペクタが現在の選択範囲に適しているかどうかを判断します。

引数
なし。

注意：現在の選択範囲を JavaScript オブジェクトとして取得するには、dom.getSelectedNode() を使用します。dom.getSelectedNode() の詳細については、『Dreamweaver API リファレンス』を参照してください。

戻り値
ブール値。インスペクタが現在の選択範囲を検査できる場合は true、検査できない場合は false が返されます。

例
次の canInspectSelection() 関数の例で、値 true が返されるのは、選択範囲内に CLASSID 属性があり、その属性の値が "clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"（Adobe Flash Player 用のクラス ID）である場合です。

```
function canInspectSelection(){3
    var theDOM = dw.getDocumentDOM();
    var theObj = theDOM.getSelectedNode();
    return (theObj.nodeType == Node.ELEMENT_NODE && ¬
        theObj.hasAttribute("classid") && ¬
        theObj.getAttribute("classid").toLowerCase()== ¬
        "clsid:D27CDB6E-AE6D-11cf-96B8-444553540000");
}
```

displayHelp()

説明
この関数が定義されている場合、疑問符（?）アイコンがプロパティインスペクタの右上隅に表示されます。この関数は、ユーザが疑問符アイコンをクリックすると呼び出されます。

引数
なし。

戻り値
なし。

例
次の displayHelp() 関数の例では、ファイルをブラウザウィンドウに開きます。このファイルは、プロパティインスペクタのフィールドについて記述されたものです。

```
function displayHelp(){
    dw.browseDocument('http://www.hooha.com/dw/inspectors/inspHelp.html');
}
```

inspectSelection()

説明
現在の選択範囲の属性に基づき、テキストフィールドの内容を更新します。

引数
maxOrMin

- **maxOrMin** 引数は、プロパティインスペクタが拡張モードと縮小モードのどちらになっているかによって、max または min になります。

戻り値
なし。

例
次の inspectSelection() 関数の例では、content 属性の値が取得され、keywords というフォームフィールドにその値が取り込まれます。

```
function inspectSelection(){
    var dom = dreamweaver.getDocumentDOM();
    var theObj = dom.getSelectedNode();
    document.forms[0].keywords.value = theObj.getAttribute("content");
}
```

第 15 章：フローティングパネル

プロパティインスペクタによるサイズとレイアウトに制限されずに、フローティングパネルやインスペクタを作成できます。

現在の選択範囲のプロパティを設定する場合、通常はまずカスタムプロパティインスペクタを使用します。しかし、カスタムフローティングパネルを使用すると、ドキュメント全体または複数の選択範囲に関する情報を、より広い領域に柔軟に表示できます。

カスタムパネルを作成し、ウィンドウメニューに追加できます。メニューシステムへの項目の追加について詳しくは、139 ページの「メニューおよびメニューコマンド」を参照してください。

次の表に、フローティングパネルを作成するために使用するファイルの一覧を示します。

パス	ファイル	説明
Configuration¥Floaters¥	**panelname**.htm	フローティングパネルのタイトルバーに表示されるテキストを指定し、フローティングパネルを定義し、必要な JavaScript 関数を含んでいます。
Configuration¥Menus¥	**menus**.xml	メニューにコマンドを追加します。

フローティングパネルファイルの動作

カスタムフローティングパネルファイルは、アプリケーションフォルダ内の Configuration¥Floaters フォルダに保存されている HTML ファイルです。フローティングパネルファイルの body セクションには、HTML フォームが含まれています。フォームエレメントに関連付けられているイベントハンドラによって、現在のドキュメントに編集を加える JavaScript コードが呼び出されることがあります。

Dreamweaver には、ウィンドウメニューからアクセスできるフローティングパネルがいくつか組み込まれています （これらの組み込みパネルは、コア Dreamweaver コードの一部であるため、Configuration¥Floaters フォルダ内にはこれらのパネルに対応するフローティングパネルファイルは含まれていません）。

カスタムフローティングパネルは、Dreamweaver に組み込まれているフローティングパネルと同様に、移動したり、サイズを変更できるほか、複数のパネルをタブ化して組み合わせることができます。しかし、次の点で、組み込みフローティングパネルと異なります。

- カスタムフローティングパネルはデフォルトのグレーで表示されます。bgcolor 属性を body タグで設定しても、何も効果はありません。
- すべてのカスタムフローティングパネルは、ドキュメントウィンドウの手前に表示されるか、または非アクティブ時にドキュメントウィンドウの背面に表示されます。フローティングパネルの表示場所は、「パネル」環境設定の「All Other Floaters」の設定によって決まります。

フローティングパネルファイルは、その他の拡張機能ファイルとも若干異なります。他の拡張機能ファイルと異なり、前回の終了時にフローティングパネルが表示されていない限り、Dreamweaver では、次の起動時にフローティングパネルファイルがメモリに読み込まれることはありません。Dreamweaver の終了時にフローティングパネルが表示されていない場合、次のいずれかの関数から参照して、パネルを定義するファイルを読み込みます。

- dreamweaver.getFloaterVisibility()
- dreamweaver.setFloaterVisibility()
- dreamweaver.toggleFloater()

これらの関数について詳しくは、『Dreamweaver API リファレンス』を参照してください。

Configuration フォルダのいずれかのファイルで dw.getFloaterVisibility(***floaterName***) 関数、dw.setFloaterVisibility(***floaterName***) 関数または dw.toggleFloater(***floaterName***) 関数を呼び出すと、次のような一連のイベントが発生します。

1 ***floaterName*** が予約されたフローティングパネル名ではない場合、Configuration¥Floaters フォルダ内で floaterName.htm というファイルが検索されます。予約されたフローティングパネル名の全一覧については、『Dreamweaver API リファレンス』で dreamweaver.getFloaterVisibility() 関数の説明を参照してください。floaterName.htm が見つからない場合、floaterName.html が検索されます。それでもファイルが見つからない場合は、この後のイベントは発生しません。

2 フローティングパネルが初めて読み込まれる際、initialPosition() 関数が定義されていれば、この関数が呼び出され、画面上でフローティングパネルのデフォルトの位置が決まります。また、initialTabs() 関数が定義されていれば、この関数が呼び出され、フローティングパネルのデフォルトのタブグループが決まります。

3 selectionChanged() 関数と documentEdited() 関数は、フローティングパネルが表示されていないときに変更があったと想定される場合に呼び出されます。

4 フローティングパネルが表示されている場合は、次のアクションが発生します。

- 選択範囲が変更されると、selectionChanged() 関数が定義されていれば、この関数が呼び出されます。
- ユーザがドキュメントを変更すると、documentEdited() 関数が定義されていれば、この関数が呼び出されます。
- フローティングパネルインターフェイスの各フィールドに関連付けられているイベントハンドラは、ユーザがそれらのフィールドに進むにつれて実行されます。例えば、onClick イベントハンドラ（dw.getDocumentDOM().body.innerHTML=" を実行するイベントハンドラ）が設定されているボタンをクリックすると、body の開始タグと終了タグの間のすべての項目が削除されます。

  フローティングパネルは、body タグで onShow() と onHide() の 2 つの特殊なイベントをサポートします。

5 ユーザが Dreamweaver を終了すると、フローティングパネルの現在の表示状態、位置およびタブグループが保存されます。次回 Dreamweaver を起動すると、前回の終了時に表示されていたフローティングパネルのフローティングパネルファイルが読み込まれます。次に、前回と同じ位置とタブグループにフローティングパネルが表示されます。

簡単なフローティングパネルの例

この例のスクリプトエディタ拡張機能では、選択されているスクリプトマーカーの基となる JavaScript コードをデザインビューに表示するフローティングパネルが作成されます。スクリプトエディタは、定義されている HTML フォームの textarea エレメント内の JavaScript コードを scriptlayer というフローティングパネルに表示します。選択したコードをフローティングパネル内で変更すると、updateScript() 関数が呼び出され、変更が保存されます。スクリプトマーカーを選択せずにスクリプトエディタを呼び出すと、(no script selected) が blanklayer というフローティングパネルに表示されます。

この拡張機能を作成するには、フローティングパネルを作成し、JavaScript コードを記述し、メニュー項目を作成します。

フローティングパネルの作成

この拡張機能用の HTML ファイルの先頭には、標準のドキュメントヘッダー情報と、フローティングパネルのタイトルバーに「Script Editor」というテキストを表示するための title タグが含まれています。

HTML ファイルのヘッダーの作成

1 新しい空白のドキュメントを作成します。

2 次のコードを入力します。

```
<!doctype html public "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>Script Editor</title>
<script language="JavaScript">
```

この拡張機能では、ユーザがスクリプトマーカーを選択していない場合に (no script selected) を表示するフローティングパネルと、選択されているスクリプトマーカーの基となる JavaScript コードを表示するフローティングパネルの 2 つを定義します。次のコードは、これら 2 つのフローティングパネル、または blanklayer および scriptlayer という絶対位置の（AP）エレメントを定義します。

2 つのフローティングパネルの作成

1 HTML ファイルのヘッダーの後に次のコードを追加します。

```
<body>
<div id="blanklayer" style="position:absolute; width:422px; ¬
height:181px; z-index:1; left: 8px; top: 11px; ¬
visibility: hidden">
<center>
<br>
<br>
<br>
<br>
<br>
(no script selected)
</center>
</div>

<div id="scriptlayer" style="position:absolute; width:422px; ¬
height:181px; z-index:1; left: 8px; top: 11px; ¬
visibility: visible">
<form name="theForm">
<textarea name="scriptCode" cols="80" rows="20" wrap="VIRTUAL" ¬
onBlur="updateScript()"></textarea>
</form>
</div>

</body>
</html>
```

2 ファイルを scriptEditor.htm という名前で Configuration¥Floaters フォルダに保存します。

どちらの div タグも、style 属性を使用して、フローティングパネルの位置（absolute）、サイズ（width:422px と height:181px）およびデフォルトの visibility 設定（visible）を指定します。blanklayer パネルは center 属性および連続した改行（br）タグを使用して、テキストをパネルの中央に配置します。scriptlayer パネルは textarea 1 つを含むフォームを作成し、選択された JavaScript コードを表示します。また、textarea タグは、選択されているコードが変更されたことを示す onBlur イベントが発生したときに、変更されたテキストをドキュメントに書き戻すための updateScript() 関数を呼び出すことも指定します。

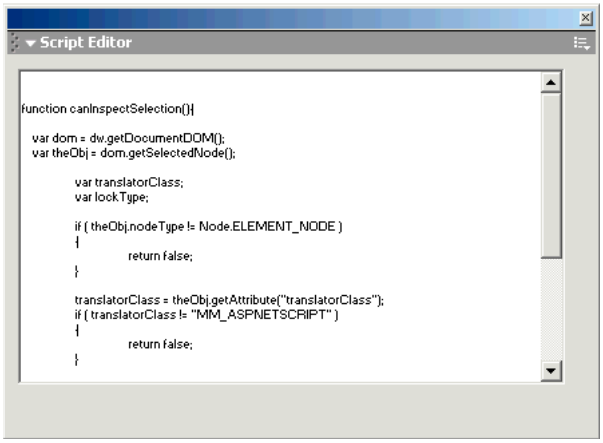
JavaScript コードの記述

スクリプトエディタ用の JavaScript コードは、Dreamweaver によって呼び出される selectionchanged() というフローティングパネル関数と、updateScript() というユーザ定義関数で構成されています。

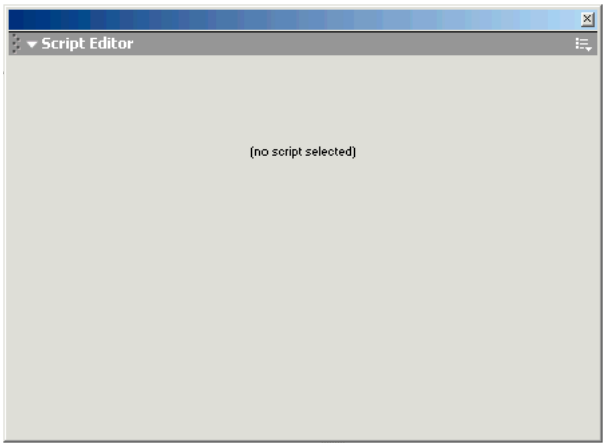
selectionChanged()：スクリプトマーカーが選択されているかどうかの判断

selectionChanged() 関数は、スクリプトマーカーがデザインビューで選択されているか判断します。スクリプトマーカーは、ドキュメントの body セクションに JavaScript ルーチンが存在し、環境設定ダイアログボックスの「不可視エレメント」セクションで「スクリプト」がオンになっている場合にデザインビューに表示されます。次の図は、スクリプトマーカーのアイコンを示しています。

selectionChanged() 関数は、まず dw.getDocumentDOM() 関数を呼び出し、ユーザのドキュメントに対するドキュメントオブジェクトモデル（DOM）を取得します。次に、getSelectedNode() 関数を呼び出して、そのユーザのドキュメントについて選択されているノードがエレメントであるかをまず調べ、次にそれが SCRIPT タグであるかを調べます。これらの条件がどちらも成り立つ場合は、selectionChanged() 関数によって scripteditor フローティングパネルが表示され、基になる JavaScript コードがこのレイヤーに読み込まれます。また、visibility プロパティ（blanklayer フローティングパネルのプロパティ）の値が hidden に設定されます。次の図は、選択された JavaScript コードが表示された scriptlayer フローティングパネルです。



選択されているノードがエレメントではないか、script タグでない場合は、selectionChanged() 関数によって blanklayer フローティングパネルが表示され、scriptlayer パネルは非表示になります。blanklayer フローティングパネルには、次の図に示すようなテキスト (no script selected) が表示されます。



selectionChanged() 関数を追加する

1 Configuration¥Floaters フォルダにある scriptEditor.htm というファイルを開きます。

2 ファイルのヘッダーセクションに次のコードを入力します。

```
function selectionChanged(){
    /* get the selected node */
    var theDOM = dw.getDocumentDOM();
    var theNode = theDOM.getSelectedNode();

    /* check to see if the node is a script marker */
    if (theNode.nodeType == Node.ELEMENT_NODE && ¬
    theNode.tagName == "SCRIPT"){
        document.layers['scriptlayer'].visibility = 'visible';
        document.layers['scriptlayer'].document.theForm.¬
        scriptCode.value = theNode.innerHTML;
        document.layers['blanklayer'].visibility = 'hidden';
    }else{
        document.layers['scriptlayer'].visibility = 'hidden';
        document.layers['blanklayer'].visibility = 'visible';
    }
}
```

3 ファイルを保存します。

updateScript()：変更の書き込み

updateScript() 関数は、onBlur イベントが textarea（scriptlayer パネル内）で発生したときに、選択されているスクリプトを書き戻します。textarea フォームエレメントには JavaScript コードが含まれていて、onBlur イベントが、textarea から入力フォーカスが失われたときに発生します。

1 Configuration¥Floaters フォルダにある scriptEditor.htm というファイルを開きます。

2 ファイルのヘッダーセクションに次のコードを入力します。

```
/* update the document with any changes made by
    the user in the textarea */

    function updateScript(){
    var theDOM = dw.getDocumentDOM();
    var theNode = theDOM.getSelectedNode();
    theNode.innerHTML = document.layers['scriptlayer'].document.¬
    theForm.scriptCode.value;
}

</script>
</head>
```

3 ファイルを保存します。

メニュー項目の作成

スクリプトエディタのコードを Configuration¥Floaters フォルダに保存するだけでは十分ではありません。dw.setFloaterVisibility('scriptEditor',true) 関数または dw.toggleFloater('scriptEditor') 関数を呼び出して、フローティングパネルを読み込み、表示できるようにする必要があります。スクリプトエディタの呼び出し元として最もわかりやすいのは、menus.xml ファイルで定義されるウィンドウメニューです。menuitem タグを作成して、次の図に示すように、ウィンドウメニューにスクリプトエディタ拡張機能のエントリを作成する必要があります。

デザインビューで現在のドキュメントのスクリプトマーカーを選択し、続いてスクリプトエディタメニュー項目を選択すると、Script Editor フローティングパネルが呼び出され、スクリプトマーカーの基となる JavaScript コードが表示されます。スクリプトマーカーを選択していないときにメニュー項目を選択すると、blanklayer パネルに (no script selected) のテキストが表示されます。

1 Configuration¥Menus フォルダにある menus.xml ファイルを開きます。

2 <menuitem name="Tile _Vertically" で開始するタグを検索し、終了タグ /> の後に挿入ポイントを配置します。

3 新しい行に、次のコードを挿入します。

```
<menuitem name="Script Editor" enabled="true" ¬
command="dw.toggleFloater('scriptEditor')"¬
checked="dw.getFloaterVisibility('scriptEditor')" />
```

4 ファイルを保存します。

フローティングパネル API 関数

フローティングパネル API のカスタム関数は、すべてオプションです。

この節で説明する機能の中には、Windows オペレーティングシステムでのみ機能するものがあります。関数についての説明では、これに該当するかどうかを示してあります。

displayHelp()

説明

この関数が定義されている場合は、ダイアログボックスの「OK」と「キャンセル」の下に「ヘルプ」ボタンが表示されます。この関数は、ユーザが「ヘルプ」ボタンをクリックすると呼び出されます。

引数

なし。

戻り値

なし。

例

```
// the following instance of displayHelp() opens
// in a browser a file that explains how to use
// the extension.
function displayHelp(){
    var myHelpFile = dw.getConfigurationPath() +
        '/ExtensionsHelp/superDuperHelp.htm';
    dw.browseDocument(myHelpFile);
}
```

documentEdited()

説明

この関数は、フローティングパネルが非表示から表示に切り替わったとき、および現在の編集操作全体が完了したときに呼び出されます。つまり、複数の編集操作が行われた後で、この関数が呼び出されることがあります。この関数は、フローティングパネルがドキュメントの編集内容をトラッキングする必要がある場合にのみ定義します。

注意：documentEdited() 関数を呼び出すとパフォーマンスに影響を及ぼすので、この関数は必要な場合のみ定義してください。

引数

なし。

戻り値

なし。

例

次の documentEdited() 関数の例では、ドキュメント内の AP エレメントが検索され、AP エレメント数を表示するテキストフィールドが更新されます。

```
function documentEdited(){
    /* create a list of all the AP elements in the document */
    var theDOM = dw.getDocumentDOM();
    var layersInDoc = theDOM.getElementsByTagName("layer");
    var layerCount = layersInDoc.length;
    /* update the numOfLayers field with the new layercount */
    document.theForm.numOfLayers.value = layerCount;
}
```

getDockingSide()

対応バージョン

Dreamweaver MX

説明

フローティングパネルをドッキングできる場所を指定します。この関数により、"left"、"right"、"top" および "bottom" のいくつかを組み合わせたストリングが返されます。ストリングにラベルが含まれている場合は、そのラベルに対応する場所にフローティングパネルをドッキングできます。この関数が定義されていない場合は、フローティングパネルのドッキングはできません。

この関数を使用して、特定のパネルが Dreamweaver ワークスペースの特定の場所にドッキングしたり、パネルが互いにドッキングしたりすることを防ぐことができます。

引数
なし。

戻り値
"left"、"right"、"top" および "bottom" のいずれかを含む、またはこれらを組み合わせたストリング。このストリングにより、フローティングパネルをドッキングできる場所が指定されます。

例

```
getDockingSide()
{
    return dock_side = "left top";
}
```

initialPosition()

説明
フローティングウィンドウが最初に呼び出されたときの初期位置を決定します。この関数が定義されていない場合は、画面の中央がデフォルトの表示位置となります。

引数
platform

- **platform** 引数の値は、ユーザのプラットフォームに応じて "Mac" または "Win" のいずれかになります。

戻り値
"leftPosInPixels,topPosInPixels" という形式のストリング。

例
次の initialPosition() 関数の例では、フローティングパネルを最初に表示するときに、Windows では画面の左から 420 ピクセル、上から 20 ピクセルの位置に、また Macintosh では画面の左から 390 ピクセル、上から 20 ピクセルの位置に表示するように指定します。

```
function initialPosition(platform){
    var initPos = "420,20";
    if (platform == "macintosh"){
        initPos = "390,20";
    }
    return initPos;
}
```

initialTabs()

説明
フローティングパネルを最初に表示するときに、このパネルと一緒にタブで組み合わせて表示するフローティングパネルを決定します。指定したフローティングパネルの中に、前回表示されたものがある場合、そのパネルはタブグループには含まれません。2 つの特定のカスタムフローティングパネルをタブで組み合わせて表示するには、それぞれのフローティングパネルが initialTabs() 関数でもう一方のフローティングパネルを参照するように設定します。

引数
なし。

戻り値
"floaterName1,floaterName2,...floaterNameN" という形式のストリング。

例
次の initialTabs() 関数の例では、フローティングパネルを最初に表示するときに、scriptEditor フローティングパネルとタブグループ化して表示するように指定します。

```
function initialTabs(){
    return "scriptEditor";
}
```

isATarget()

対応バージョン
Dreamweaver MX（Windows のみ）

説明
このフローティングパネルに他のパネルをドッキングできるかどうかを指定します。isATarget() 関数を指定していない場合、このパネルに他のパネルをドッキングすることはできません。この関数は、ユーザがこのパネルを他のパネルと組み合わせようとしたときに呼び出されます。

引数
なし。

戻り値
ブール値。他のフローティングパネルがこのパネルにドッキングできる場合は true、そうでない場合は false が返されます。

例

```
isATarget()
{
    return true;
}
```

isAvailableInCodeView()

説明
コードビューの選択時にフローティングパネルを有効にするかどうかを決定します。この関数が定義されていない場合、フローティングパネルはコードビューで使用できません。

引数
なし。

戻り値
ブール値。コードビューでフローティングパネルを有効にする場合は true、そうでない場合は false が返されます。

isResizable()

対応バージョン
Dreamweaver 4

説明
ユーザがフローティングパネルのサイズを変更できるかどうかを決定します。この関数を定義しない場合、または戻り値が true の場合は、ユーザ側でフローティングパネルのサイズを変更できます。戻り値が false の場合、フローティングパネルのサイズをユーザが変更することはできません。

引数
なし。

戻り値
ブール値。ユーザがフローティングパネルのサイズを変更できる場合は true、変更できない場合は false が返されます。

例
次の例では、ユーザがフローティングパネルのサイズを変更できないように指定します。

```
function isResizable()
{
    return false;
}
```

selectionChanged()

説明
フローティングパネルが非表示から表示に切り替わったとき、および選択範囲が変更されたとき（フォーカスを新規のドキュメントに切り替えたり、挿入ポイントが現在のドキュメントの新しい位置に移動した場合）に呼び出されます。この関数は、フローティングパネルが選択範囲をトラッキングする必要がある場合のみ定義します。

注意：selectionChanged() を呼び出すとパフォーマンスに影響を及ぼすので、この関数は明らかに必要な場合のみ定義するようにしてください。

引数
なし。

戻り値
なし。

例
次の selectionChanged() の例では、選択範囲がスクリプトマーカーであるかどうかによって、フローティングパネルにそれぞれ異なる AP エレメントが表示されます。選択範囲がスクリプトマーカーである場合、Dreamweaver により、scriptlayer AP エレメントが表示されます。選択範囲がスクリプトマーカーではない場合、blanklayer AP エレメントが表示されます。

```
function selectionChanged(){
    /* get the selected node */
        var theDOM = dw.getDocumentDOM();
        var theNode = theDOM.getSelectedNode();
    /* check to see if the node is a script marker */
        if (theNode.nodeType == Node.ELEMENT_NODE && ¬
        theNode.tagName == "SCRIPT"){
            document.layers['blanklayer'].visibility = 'hidden';
            document.layers['scriptlayer'].visibility = 'visible';}
        else{
            document.layers['scriptlayer'].visibility = 'hidden';
            document.layers['blanklayer'].visibility = 'visible';
        }
}
```

パフォーマンスについて

カスタムフローティングパネルで selectionChanged() 関数または documentEdited() 関数を宣言すると、Dreamweaver のパフォーマンスが低下することがあります。これは、Dreamweaver のアイドル状態継続時間が 1/10 秒を超えた場合は、キーを押したりマウスをクリックしたりするたびに documentEdited() 関数と selectionChanged() 関数が呼び出されるからです。パフォーマンスへの影響をテストする場合は、大きいドキュメント（100 KB 以上の HTML）をできる限り使用し、様々なシナリオでフローティングパネルをテストする必要があります。

パフォーマンスの低下を防ぐには、setTimeout() 関数を使用します。setTimeout() 関数には、ブラウザの場合と同様に、呼び出される JavaScript とその JavaScript が呼び出されるまでの待ち時間（ミリ秒）の 2 つの引数を指定します。

setTimeout() メソッドを使用して、処理を一時停止することができます。この一時停止の間に、ユーザはアプリケーションの操作を続行できます。スクリプトの処理中は画面が停止し、ユーザが編集できなくなるので、これらの一時停止を明示的に組み込む必要があります。一時停止の間は、インターフェイスやフローティングパネルの更新もできなくなります。

次の例は、ドキュメントのすべての AP エレメントに関する情報を表示するフローティングパネルからの抜粋です。このコードは setTimeout() メソッドを使用して、各 AP エレメントの処理後に 1/2 秒間だけ一時停止します。

```
/* create a flag that specifies whether an edit is being processed, and set it to false.*/
document.running = false;
/* this function called when document is edited */
function documentEdited(){
    /* create a list of all the AP elements to be processed */
    var dom = dw.getDocumentDOM();
    document.layers = dom.getElementsByTagName("layer");
    document.numLayers = document.layers.length;
    document.numProcessed = 0;
    /* set a timer to call processLayer(); if we didn't get
     * to finish processing the previous edit, then the timer
     * is already set. */
    if (document.running = false){
        setTimeout("processLayer()", 500);
    }
    /* set the processing flag to true */
    document.running = true;
}
/* process one AP element*/
function processLayer(){
    /* display information for the next unprocessed AP element.
        displayLayer() is a function you would write to
        perform the "magic".0*/
     displayLayer(document.layers[document.numProcessed]);
    /* if there's more work to do, set a timeout to process
     * the next layer.0.If we're finished, set the document.running
     * flag to false. */
    document.numProcessed = document.numProcessed + 1;
    if (document.numProcessed < document.numLayers){
        setTimeout("processLayer()", 500);
    }else{
        document.running = false;
    }
}
```

第 16 章：ビヘイビア

ビヘイビアという用語は、イベントとアクションの組み合わせを表します。イベントの例には、onClick、onLoad、onSubmit などがあります。アクションの例には、Check Plug-in、Go to URL、Swap Image などがあります。どの HTML エレメントがどのイベントを受け付けるかはブラウザによって決まります。各ブラウザでサポートされているイベントを含むファイルは、Adobe Dreamweaver アプリケーションフォルダ内の Configuration¥Behaviors¥Events フォルダに保存されています。

通常、アクションファイルの body セクションには、HTML フォームが含まれています。HTML フォームでは、アクション用のパラメータを使用できます（例えば、どの絶対位置のエレメントを表示または非表示にするかを指定するパラメータ）。アクションファイルの head セクションには、body コンテンツからのフォーム入力を処理する JavaScript 関数が含まれています。また、これらは、ユーザのドキュメントに挿入される関数、引数およびイベントハンドラを制御します。

ユーザと関数を共有したり、同じ JavaScript 関数を繰り返し挿入したりするときは、ビヘイビアアクションを記述します。アクション用の各パラメータは毎回変更してください。

注意：ビヘイビアを使用して直接 VBScript 関数を挿入することはできません。ただし、applyBehavior() 関数の DOM（Document Object Model）を編集することにより、間接的に VBScript 関数を追加できます。

次の表に、ビヘイビアアクションを作成するために使用するファイルの一覧を示します。

パス	ファイル	説明
Configuration¥Behaviors¥Actions¥	**behavior action**.htm	ファイルのボディには、アクションのパラメータ用の HTML フォームが含まれます。ファイルのヘッドには、JavaScript 関数が含まれます。

注意：Web アプリケーション機能を提供するサーバビヘイビアについて詳しくは、「240 ページの「サーバビヘイビア」」を参照してください。

ビヘイビアの動作

ユーザが Dreamweaver ドキュメント中の HTML エレメントを選択してビヘイビアパネルの + ボタンをクリックすると、次のような一連のイベントが発生します。

1. Dreamweaver により各アクションファイルの canAcceptBehavior() 関数が呼び出され、このアクションがドキュメントまたは選択されたエレメントに対して適切かどうかが判断されます。

   この関数から false が返されると、そのアクションはアクションポップアップメニューで淡色表示になります。例えば、ユーザのドキュメントに SWF ファイルがない場合は、「Shockwave のコントロール」アクションが淡色表示になります。イベントのリストが返された場合は、各イベントが、現在選択されている HTML エレメントやターゲットブラウザの有効なイベントと比較され、一致するイベントが見つかるまで検索されます。canAcceptBehavior() 関数により検索された一致するイベントは、イベントポップアップメニューのリストの上部に挿入されます。一致するイベントがない場合は、その HTML エレメントのデフォルトのイベントが一番上のメニュー項目になります（デフォルトのイベントには、イベントファイル内でアスタリスク（*）が付きます）。メニューの残りのイベントは、イベントファイルから集められます。

2. ユーザがアクションポップアップメニューからアクションを選択します。

3. Dreamweaver は windowDimensions() 関数を呼び出して、パラメータダイアログボックスのサイズを決定します。windowDimensions() 関数が定義されていない場合は、サイズが自動的に計算されます。

body エレメントのコンテンツに関わらず、ダイアログボックスの右端には常に、「OK」ボタンと「キャンセル」ボタンが表示されます。

4 Dreamweaver により、アクションファイルの BODY エレメントを含むダイアログボックスが表示されます。アクションファイルの body タグに onLoad ハンドラが含まれている場合は、Dreamweaver によって実行されます。

5 ユーザがアクションのパラメータを入力します。ユーザがフォームフィールドに進むたびに、フォームフィールドに関連付けられたイベントハンドラが実行されます。

6 ユーザが「OK」をクリックします。

7 Dreamweaver により、選択されたアクションファイルの behaviorFunction() 関数と applyBehavior() 関数が呼び出されます。これらの関数により、ユーザのドキュメントに挿入されるストリングが返されます。

8 ユーザが後から「アクション」列でそのアクションをダブルクリックすると、Dreamweaver によってパラメータダイアログボックスが再び表示され、onLoad ハンドラが実行されます。次に、選択されたアクションファイルの inspectBehavior() 関数が呼び出されます。この関数によって、ユーザが以前に入力したデータが各フィールドに取り込まれます。

ユーザのファイルへの複数の関数の挿入

アクションを使用して、複数の関数（メインのビヘイビア関数と不特定数の補助関数）を head セクションに挿入することができます。各アクションファイルにおける関数定義が完全に同じであれば、複数のビヘイビアで補助関数を共有することもできます。共有する関数の定義を完全に同じにするための 1 つの方法としては、各補助関数を 1 つの外部 JavaScript ファイルに保存しておき、<SCRIPT SRC="***externalFile***.js"> を使用して、このファイルを適切なアクションファイルに挿入します。

ユーザがあるビヘイビアを削除すると、Dreamweaver によって、そのビヘイビアに関連付けられた補助関数のうち、使用されていないものがすべて削除されます。ただし、他のビヘイビアで使用されている補助関数は削除されません。Dreamweaver で補助関数が削除される際、必要な関数が削除されないように、安全性を重視したアルゴリズムが採用されているため、不要な関数がユーザのドキュメントに残る場合もあります。

戻り値が必要なアクションの処理

例えば onMouseOver="window.status='This is a link'; return true" のように、イベントハンドラが値を返さなければならない場合がありますが、Dreamweaver によって "return behaviorName(args)" アクションがイベントハンドラに挿入されると、リストにある残りのビヘイビアはスキップされます。

この問題を回避するには、document.MM_returnValue 変数を behaviorFunction() 関数から返されるストリング内にある適切な戻り値に設定します。この変数を設定すると、Dreamweaver はイベントハンドラのアクションリストの最後に return document.MM_returnValue を挿入します。MM_returnValue 変数の使用例については、Dreamweaver アプリケーションフォルダ内の Configuration¥Behaviors¥Actions フォルダにある Validate Form.js ファイルを参照してください。

簡単なビヘイビアの例

ビヘイビアの動作と作成方法について理解するには、実際のコード例を見ながら学習すると効果的です。Dreamweaver アプリケーションフォルダ内の Configuration¥Behaviors¥Actions フォルダに例があります。ただし、これらの例の多くは非常に複雑です。この例は単純なのでビヘイビア作成の学習に適しています。最初は、簡単な Call JavaScript.htm というアクションファイルと、これに対応するすべての JavaScript 関数を含む Call JavaScript.js を使用して学習することをお勧めします。

ビヘイビアを作成するには、拡張機能を作成し、参照する HTML ファイルを作成して、ビヘイビアをテストします。

ビヘイビア拡張機能の作成

以下のコードは、比較的簡単な例を示しています。このコードでは、ブラウザのブランド名をチェックします。ブランド名が Netscape Navigator の場合と、Microsoft Internet Explorer の場合とで、別々のページが開きます。このコードは、Opera や WebTV などの他のブラウザをチェックするように拡張することや、URL へのアクセス以外のアクションを実行するように修正することが簡単にできます。

1 新しい空白のファイルを作成します。

2 次のコードをファイルに追加します。

```
<!DOCTYPE HTML SYSTEM "-//Adobe//DWExtension layout-engine 10.0//dialog">
<html>
<head>
<title>behavior "Check Browser Brand"</title>
<meta http-equiv="Content-Type" content="text/html">
<script language="JavaScript">

// The function that will be inserted into the
// HEAD of the user's document
function checkBrowserBrand(netscapeURL,explorerURL) {
    if (navigator.appName == "Netscape") {
        if (netscapeURL) location.href = netscapeURL;
    }else if (navigator.appName == "Microsoft Internet Explorer") {
        if (explorerURL) location.href = explorerURL;
    }
}

//****************** API **********************

function canAcceptBehavior(){
    return true;
}

// Return the name of the function to be inserted into
// the HEAD of the user's document
function behaviorFunction(){
        return "checkBrowserBrand";
}

// Create the function call that will be inserted
// with the event handler
function applyBehavior() {
    var nsURL = escape(document.theForm.nsURL.value);
    var ieURL = escape(document.theForm.ieURL.value);
    if (nsURL && ieURL) {
        return "checkBrowserBrand(\'" + nsURL + "\',\'" + ieURL + "\')";
    }else{
        return "Please enter URLs in both fields."
    }
}

// Extract the arguments from the function call
// in the event handler and repopulate the
// parameters form
function inspectBehavior(fnCall){
    var argArray = getTokens(fnCall, "()',");
    var nsURL = unescape(argArray[1]);
    var ieURL = unescape(argArray[2]);
    document.theForm.nsURL.value = nsURL;
    document.theForm.ieURL.value = ieURL;
}
```

```
//***************** LOCAL FUNCTIONS ******************

// Put the pointer in the first text field
// and select the contents, if any
function initializeUI(){
    document.theForm.nsURL.focus();
    document.theForm.nsURL.select();
}

// Let the user browse to the Navigator and
// IE URLs
function browseForURLs(whichButton){
    var theURL = dreamweaver.browseForFileURL();
    if (whichButton == "nsURL"){
        document.theForm.nsURL.value = theURL;
    }else{
        document.theForm.ieURL.value = theURL;
    }
}

//*************** END OF JAVASCRIPT *****************
</script>
</head>
<body>
<form method="post" action="" name="theForm">
<table border="0" cellpadding="8">
<tr>
<td nowrap="nowrap">  Go to this URL if the browser is ¬
    Netscape Navigator:<br>
<input type="text" name="nsURL" size="50" value="">  
<input type="button" name="nsBrowse" value="Browse..." ¬
    onClick="browseForURLs('nsURL')"></td>
</tr>
<tr>
<td nowrap="nowrap">  Go to this URL is the browser is ¬
    Microsoft Internet Explorer:<br>
<input type="text" name="ieURL" size="50" value="">  
<input type="button" name="ieBrowse" value="Browse..." ¬
    onClick="browseForURLs('ieURL')"></td>
</tr>
</table>
</form>
</body>
</html>
```

3 ファイルを Configuration/Behaviors/Actions/BrowserDependentURL.htm として保存します。

参照する HTML ファイルの作成（非推奨）

参照する HTML ファイルを作成します。これらのファイルは、ブラウザが Internet Explorer の場合に参照するファイルと、ブラウザが Netscape Navigator の場合に参照するファイルです。

1 以下の内容のファイルを作成します。

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title>Internet Explorer Only</title>
</head>

<body>
This is the page to go to if you are using Internet Explorer.
</body>
</html>
```

2 ファイルをコンピュータのサイト内に iecontent.htm として保存します。

3 以下の内容の別のファイルを作成します。

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title>Netscape Navigator content</title>
</head>

<body>
This is the page to go to if you are using Netscape Navigator.
</body>
</html>
```

4 このファイルを netscapecontent.htm として iecontent.htm ファイルと同じフォルダに保存します。

5 Dreamweaver を再起動します。

6 以下の内容の HTML ファイルを作成します。

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title>Which browser</title>
</head>

<body>
</body>
</html>
```

7 このファイルを whichbrowser.htm として iecontent.htm ファイルと同じフォルダに保存します。

8 ビヘイビアパネルの + ボタンをクリックし、「チェックブラウザ」ビヘイビアを選択します。

9 URL に移動するオプションの横にある参照ボタンをクリックし、netscapecontent.htm ファイル（ブラウザが Netscape Navigator の場合）を選択します。または iecontent.htm ファイル（ブラウザが Internet Explorer の場合）を選択します。

10「OK」をクリックします。

指定した JavaScript が whichbrowser.htm ファイルに追加され、ファイルが次のように表示されます。

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title>Which browser</title>
<script language="JavaScript" type="text/JavaScript">
<!--
function checkBrowserBrand(netscapeURL,explorerURL) {
    if (navigator.appName == "Netscape") {
        if (netscapeURL) location.href = netscapeURL;
    }else if (navigator.appName == "Microsoft Internet Explorer") {
        if (explorerURL) location.href = explorerURL;
    }
}
//-->
</script>
</head>

<body onLoad="checkBrowserBrand('netscaptecontent.htm','iecontent.htm')">
</body>
</html>
```

ビヘイビアのテスト

1 whichbrowser.htm ファイルをブラウザで参照します。

2 使用するブラウザに応じて、iecontent.htm または netscapecontent.htm が表示されます。

ビヘイビア API 関数

必須のビヘイビア API の関数は、applyBehavior() と behaviorFunction() です。その他の関数はオプションです。

applyBehavior()

説明

この関数は、behaviorFunction() によって挿入される関数を呼び出すイベントハンドラをユーザのドキュメントに挿入します。applyBehavior() 関数を使用すると、ユーザのドキュメントで他の編集も行うことができますが、ビヘイビアが適用されるオブジェクトや、アクションを受け取るオブジェクトを削除しないようにしてください。

applyBehavior() 関数を記述するとき、ユーザのドキュメントを編集する方法を決める必要があります。例えば、ドキュメントのボディにある script タグ中にコードを挿入することがあります。これを行うには、標準の DOM 編集 API を使用します。

引数

uniqueName

この引数には、ユーザのドキュメント内の全ビヘイビアの全インスタンスで一意になる識別子を指定します。この引数の形式は **functionNameInteger** です。**functionName** は、behaviorFunction() によって挿入される関数の名前です。この引数は、ユーザのドキュメントにタグを挿入し、その NAME 属性に一意の値を割り当てる場合に役立ちます。

戻り値

ユーザのドキュメントに挿入する関数呼び出しを含むストリング。通常、この関数呼び出しはユーザからのパラメータを受け取った後に挿入されます。applyBehavior() 関数がユーザの入力を無効と判断した場合は、関数呼び出しの代わりにエラーストリングを返すことができます。返されたストリングが空白（return "";）の場合、Dreamweaver からエラーは通知されません。ストリングが空白ではなく、関数呼び出しでもない場合、ダイアログボックスには **Invalid input supplied for this behavior:** and the string returned from applyBehavior() というテキストが表示されます。戻り値が null（return;）の場合、Dreamweaver からエラーの発生が通知されますが、詳細情報は提供されません。

注意：JavaScript インタープリタからエラーが通知されないようにするには、返されるストリング内の引用符（""）の前にバックスラッシュ（\）を指定する必要があります。

例

次の applyBehavior() 関数の例では、MM_openBrWindow() という関数呼び出しが返され、ユーザが指定したパラメータ（ウィンドウの高さと幅の指定、ウィンドウにスクロールバー、ツールバー、ロケーションバーなどの機能を設定するかどうかの指定、およびウィンドウで開く URL の指定）が渡されます。

```
function applyBehavior() {
    var i,theURL,theName,arrayIndex = 0;
    var argArray = new Array(); //use array to produce correct ¬
        number of commas w/o spaces
    var checkBoxNames = new Array("toolbar","location",¬
        "status","menubar","scrollbars","resizable");

    for (i=0; i<checkBoxNames.length; i++) {
        theCheckBox = eval("document.theForm." + checkBoxNames[i]);
        if (theCheckBox.checked) argArray[arrayIndex++] = (checkBoxNames[i] + "=yes");
    }
    if (document.theForm.width.value)
        argArray[arrayIndex++] = ("width=" + document.theForm.width.value);
    if (document.theForm.height.value)
        argArray[arrayIndex++] = ("height=" + document.theForm.height.value);
    theURL = escape(document.theForm.URL.value);
    theName = document.theForm.winName.value;
    return "MM_openBrWindow('"+theURL+"','"+theName+"','"+argArray.join()+"')";
}
```

behaviorFunction()

説明

この関数は、次のタグで囲んだ関数をユーザのドキュメントの head セクションに挿入します。

```
<SCRIPT LANGUAGE="JavaScript"></SCRIPT>
```

引数

なし。

戻り値

ユーザのドキュメントに挿入する、JavaScript 関数を含むストリング、または関数の名前を含むストリング。この値は毎回完全に同じである必要があります。ユーザの入力によって変更することはできません。ドキュメント内のエレメントにアクションを適用した回数に関係なく、関数が挿入されるのは 1 回だけです。

注意：JavaScript インタープリタからエラーが通知されないようにするには、返されるストリング内の引用符（""）の前にバックスラッシュ（\）エスケープ文字を置く必要があります。

例

次の behaviorFunction() 関数の例は、MM_popupMsg() という関数を返します。

```
function behaviorFunction(){
    return ""+
    "function MM_popupMsg(theMsg) { //v1.0\n"+
    "alert(theMsg);\n"+
    "}";
}
```

次の例は、前述の behaviorFunction() 宣言と同じ動作をします。このメソッドは、Dreamweaver に付属しているすべてのビヘイビアで behaviorFunction() 関数を宣言するために使用します。

```
function MM_popupMsg(theMsg){ //v1.0
    alert(theMsg);
}

function behaviorFunction(){
    return "MM_popupMsg";
}
```

canAcceptBehavior()

説明

この関数は、選択された HTML エレメントに対してアクションが許可されているかどうかを判断し、そのトリガとなるデフォルトのイベントを指定します。また、SWF ファイルなどの特定のオブジェクトがユーザのドキュメント内に存在するかどうかをチェックすることができ、これらのオブジェクトがない場合はそのアクションを許可しません。

引数

HTMLElement

この引数には、選択された HTML エレメントを指定します。

戻り値

次のいずれかの値が返されます。

- アクションが許可されても、優先イベントを持たない場合は true。
- このアクションの優先イベントのリスト（先度の高い方から順に並んでいます）。優先イベントを指定すると、選択されたオブジェクトのデフォルトのイベントがオーバーライドされます（デフォルトのイベントには、イベントファイル内でアスタリスク（*）が付いています）。「228 ページの「ビヘイビアの動作」」の手順 1 を参照してください。
- アクションが許可されない場合は false。

  canAcceptBehavior() 関数から false が返されたアクションは、ビヘイビアパネルのアクションポップアップメニューで淡色表示されます。

例

次の canAcceptBehavior() 関数の例は、ドキュメントに名前の付いたイメージがある場合、ビヘイビアの優先イベントのリストを返します。

```
function canAcceptBehavior(){
    var theDOM = dreamweaver.getDocumentDOM();
    // Get an array of all images in the document
    var allImages = theDOM.getElementsByTagName('IMG');
    if (allImages.length > 0){
        return "onMouseOver, onClick, onMouseDown";
    }else{
        return false;
    }
}
```

displayHelp()

説明

この関数が定義されている場合は、パラメータダイアログボックスの「OK」ボタンと「キャンセル」ボタンの下に「ヘルプ」ボタンが表示されます。この関数は、ユーザが「ヘルプ」ボタンをクリックすると呼び出されます。

引数

なし。

戻り値

なし。

例

```
// the following instance of displayHelp() opens
// in a browser a file that explains how to use
// the extension.
function displayHelp(){
    var myHelpFile = dw.getConfigurationPath() +
        '/ExtensionsHelp/superDuperHelp.htm';
    dw.browseDocument(myHelpFile);
}
```

deleteBehavior()

説明

この関数は、applyBehavior() 関数で実行された編集をすべて取り消します。

注意：ユーザがビヘイビアパネルのビヘイビアを削除すると、Dreamweaver によって、そのビヘイビアに関連付けられている関数宣言およびイベントハンドラが自動的に削除されます。deleteBehavior() 関数は、applyBehavior() 関数によってユーザのドキュメントに別の編集が実行された場合（タグが挿入された場合など）に限り定義する必要があります。

引数

applyBehaviorString

この引数には、applyBehavior() 関数により返されたストリングを指定します。

戻り値

なし。

identifyBehaviorArguments()

説明

この関数は、ビヘイビア関数呼び出しからの引数を、ナビゲーションリンク、依存ファイル、URL、Netscape Navigator 4.0 スタイルリファレンス、またはオブジェクト名として識別します。これにより、ユーザがドキュメントを別の場所に保存した場合に、ビヘイビアの URL を更新できるようにしたり、参照されるファイルをサイトマップに表示したり、サーバへのアップロードやサーバからのダウンロードの際に、これらのファイルを依存ファイルとして扱うように指定したりします。

引数

theFunctionCall

この引数には、applyBehavior() 関数により返されたストリングを指定します。

戻り値

カンマで区切られた、関数呼び出しの引数のタイプのリストを含むストリング。このリストの長さは、関数呼び出しの引数の数と一致する必要があります。引数には、次のいずれかのタイプを指定します。

- nav の引数タイプは、引数がナビゲート可能な URL であるため、サイトマップに表示する必要があることを示します。
- dep の引数タイプは、引数が依存ファイルの URL であるため、このビヘイビアを含むドキュメントをサーバに対してアップロードまたはダウンロードする際には、他のすべての依存ファイルと共に含める必要があることを示します。
- URL の引数タイプは、引数がナビゲート可能な URL かつ依存 URL であるか、タイプが不明の URL であるため、サイトマップに表示し、サーバとの間でアップロードまたはダウンロードを実行する際には依存ファイルとして扱う必要があることを示します。
- NS4.0ref の引数タイプは、引数が Navigator 4.0 スタイルのオブジェクトリファレンスであることを示します。
- IE4.0ref の引数タイプは、引数が Internet Explorer DOM 4.0 スタイルのオブジェクトリファレンスであることを示します。
- objName の引数タイプは、引数がオブジェクトの NAME 属性で指定されている、単なるオブジェクト名であることを示します。このタイプは Dreamweaver 3 から追加されました。
- other の引数タイプは、引数が上記のどのタイプでもないことを示します。

例

次の identifyBehaviorArguments() 関数の簡単な例は、「ブラウザウィンドウを開く」ビヘイビアアクションに使用できます。このアクションでは、常に 3 つの引数（開く URL、新しいウィンドウの名前、およびウィンドウプロパティのリスト）を使用する関数が返されます。

```
function identifyBehaviorArguments(fnCallStr) {
    return "URL,other,other";
}
```

「レイヤーの表示 / 非表示」のように、引数の数が可変であるビヘイビア関数には、より複雑な identifyBehaviorArguments() 関数が必要になります。identifyBehaviorArguments() 関数のこの例では、引数は最小限になっています。また、追加する引数の数は、常に最小数の倍数になります。つまり、引数の最小数が 4 である関数の場合、引数の数が 4 個、8 個、12 個になることはあっても、10 個になることはありません。

```
function identifyBehaviorArguments(fnCallStr) {
    var listOfArgTypes;
    var itemArray = dreamweaver.getTokens(fnCallStr, '(),');

    // The array of items returned by getTokens() includes the
    // function name, so the number of *arguments* in the array
    // is the length of the array minus one. Divide by 4 to get the
    // number of groups of arguments.
    var numArgGroups = ((itemArray.length - 1)/4);
    // For each group of arguments
    for (i=0; i < numArgGroups; i++){

    // Add a comma and "NS4.0ref,IE4.0ref,other,dep" (because this
    // hypothetical behavior function has a minimum of four
    // arguments the Netscape object reference, the IE object
    // reference, a dependent URL, and perhaps a property value
    // such as "show" or "hide") to the existing list of argument
    // types, or if no list yet exists, add only
    // "NS4.0ref,IE4.0ref,other,dep"
    var listOfArgTypes += ((listOfArgTypes)?",":"") + ¬
    "NS4.0ref,IE4.0ref,other,dep";
    }
}
```

inspectBehavior()

説明

この関数は、ユーザのドキュメント内で以前に適用されたビヘイビアの関数呼び出しを調べ、その結果に従って、パラメータダイアログボックスの各オプションの値を設定します。inspectBehavior() 関数が定義されていない場合は、各オプションのデフォルトの値が表示されます。

注意：inspectBehavior() 関数は、applyBehaviorString 引数から渡された情報だけを使用する必要があります。この関数内では、例えば dreamweaver.getDocumentDOM() を使用して、ユーザのドキュメントに関する他の情報を取得しないでください。

引数

applyBehaviorString

この引数には、applyBehavior() 関数により返されたストリングを指定します。

注意：'onClick="someBehavior(); return document.MM_returnValue;"' と似たコードが HTML エレメントに含まれており、ビヘイビアのメニューから新しいビヘイビアを追加する場合、Dreamweaver は、新しいビヘイビア UI がポップアップ表示されると同時に inspectBehavior() を呼び出し、空白のストリングをパラメータとして渡します。したがって、次の例に示すように、applyBehaviorString パラメータを必ず確認してください。

```
function inspectBehavior(enteredStr){
    if(enteredStr){
    //do your work here
    }
}
```

戻り値

なし。

例

次の inspectBehavior() 関数の例は、Display Status Message.htm ファイルからの抜粋です。このビヘイビアが最初に適用されたときにユーザが選択したメッセージを、パラメータダイアログボックスの「メッセージ」フィールドに設定します。

```
function inspectBehavior(msgStr){
    var startStr = msgStr.indexOf("'") + 1;
    var endStr = msgStr.lastIndexOf("'");
    if (startStr > 0 && endStr > startStr) {
        document.theForm.message.value = ¬
        unescQuotes(msgStr.substring(startStr,endStr));
    }
}
```

注意：unescQuotes() 関数について詳しくは、Configuration¥Shared¥Common¥Scripts¥CMN フォルダにある dwstring.js ファイルを参照してください。

windowDimensions()

説明

この関数は、パラメータダイアログボックスに特定のサイズを設定します。この関数が定義されていない場合、ウィンドウのサイズは自動的に計算されます。

注意：640 x 480 ピクセル以上のパラメータダイアログボックスを使用する場合に限り、この関数を定義するようにしてください。

引数

platform

platform 引数の値は、ユーザのプラットフォームに応じて、macintosh または windows になります。

戻り値

widthInPixels,heightInPixels という形式のストリング。

返されるサイズはダイアログボックス全体のサイズよりも小さくなります。これは「OK」ボタンと「キャンセル」ボタンに使用する領域が含まれていないためです。返されたサイズにすべてのオプションが収まらない場合は、スクロールバーが表示されます。

例

次の windowDimensions() の例は、パラメータダイアログボックスのサイズを 648 x 520 ピクセルに設定します。

```
function windowDimensions(){
    return "648,520";
}
```

第 17 章：サーバビヘイビア

Adobe® Dreamweaver® には、ユーザが自分のドキュメントにサーバビヘイビアを追加して次のようなサーバ側タスクを実行するためのインターフェイスが用意されています。

- ユーザの条件に基づいたレコードのフィルタリング
- レコード間のページング
- 詳細ページへの結果リストのリンク
- 結果セットへのレコードの挿入

Dreamweaver の使用中に、同じランタイムコードを何度もドキュメントに挿入することがあります。このような場合は、頻繁に使用されるコードブロックを含んだドキュメントを更新する手順を自動化する拡張機能を作成します。カスタムサーバビヘイビアを実装するためのサーバビヘイビアビルダーのインターフェイスでの作業について詳しくは、『Dreamweaver ファーストステップガイド』の「カスタムサーバビヘイビアの追加」を参照してください。また、サーバビヘイビアのサポートファイルでの作業と、作成されたサーバビヘイビアと対話する関数について詳しくは、この章を参照してください。個別の関数について詳しくは、『Dreamweaver API リファレンス』の「サーバビヘイビア関数」と「Extension Data Manager 関数」を参照してください。Dreamweaver では現在、ASP/JavaScript、ASP/VBScript、ColdFusion、PHP/MySQL の各サーバモデルにランタイムコードを追加するサーバビヘイビア拡張機能がサポートされています。

サーバビヘイビアの用語

サーバビヘイビアに関連してよく使用される用語を以下で説明します。

サーバビヘイビア拡張機能 サーバビヘイビア拡張機能は、サーバ側コードと Dreamweaver を結ぶインターフェイスです。サーバビヘイビア拡張機能は、JavaScript、HTML および Extension Data Markup Language（EDML）で構成されます。EDML は、拡張データ専用に作成された XML です。Configuration¥ServerBehaviors フォルダ内のインストールフォルダには、これらのファイルのサンプルがサーバモデルごとに格納されています。拡張機能のスクリプトを記述する際は、dwscripts.applySB() 関数を使用して、EDML ファイルの読み取り、拡張機能のコンポーネントの取得およびユーザドキュメントへの適切なコードブロックの追加が Dreamweaver によって行われるようにします。

サーバビヘイビアインスタンス Dreamweaver でコードブロックがユーザのドキュメントに挿入されると、挿入されたコードでサーバビヘイビアのインスタンスが構成されます。ほとんどのサーバビヘイビアは何度も適用することができるので、複数のサーバビヘイビアインスタンスが作成されることになります。各サーバビヘイビアインスタンスは、Dreamweaver インターフェイスのサーバビヘイビアパネルに表示されます。

ランタイムコード ランタイムコードは、サーバビヘイビアが適用されるとドキュメントに挿入される一連のコードブロックです。通常、これらのコードブロックには、<% ... %> タグで囲まれた ASP スクリプトなどのサーバ側コードが含まれています。

構成要素 サーバビヘイビア拡張機能によって、ユーザのドキュメントにコードブロックが挿入されます。コードブロックは、1 つの連続したスクリプトのブロックです。例えば、サーバ側タグ、HTML タグ、あるいはサーバ側機能を Web ページに追加する属性などが挙げられます。EDML ファイルでは、各コードブロックが構成要素として定義されます。1 つのサーバビヘイビアの全構成要素で、1 つの構成要素グループが形成されます。

注意：構成要素、構成要素グループおよび Dreamweaver EDML ファイルの構造については、241 ページの「EDML（Extension Data Markup Language）」を参照してください。

Dreamweaver のアーキテクチャ

サーバビヘイビアビルダーを使用して Dreamweaver 独自の拡張機能を作成すると、Dreamweaver ドキュメントへのサーバビヘイビアコードの挿入をサポートする複数のファイル（EDML ファイルと HTML スクリプトファイル）が作成されます（一部のビヘイビアでは、追加機能のために JavaScript ファイルも参照されます）。このアーキテクチャにより、API を簡単に実装できます。また、ランタイムコードと Dreamweaver でのその配置方法を切り離すことができます。このトピックでは、これらのファイルの修正方法について説明します。

サーバビヘイビアのフォルダとファイル

各サーバビヘイビアのユーザインターフェイス（UI）は、Configuration¥ServerBehaviors¥ServerModelName フォルダに格納されています。ServerModelName は、ASP_Js（JavaScript）、ASP_Vbs（VBScript）、ColdFusion、PHP_MySQL または Shared（クロスサーバモデル実装）のいずれかのサーバタイプです。

EDML（Extension Data Markup Language）

サーバビヘイビアビルダーを使用すると、2 つの EDML ファイルが生成されます。つまり、ユーザがサーバビヘイビアビルダーで指定した名前と一致する EDML グループファイルと EDML 構成要素ファイルです。グループファイルでは、関連する構成要素が定義されて、コードブロックが表示されます。グループでは、個別のサーバビヘイビアを作成するために組み合わされる構成要素が定義されます。

グループファイル

グループファイルには、構成要素のリストが含まれます。構成要素ファイルには、サーバモデル固有のすべてのコードデータが含まれます。構成要素ファイルは、複数の拡張機能で使用できるため、複数のグループファイルで同じ構成要素ファイルを参照することができます。

次の例では、サーバビヘイビアの EDML グループファイルの上位レベルを示します。すべてのエレメントと属性のリストについては、253 ページの「EDML グループファイルタグ」を参照してください。

```
<group serverBehavior="Go To Detail Page.htm" dataSource="Recordset.htm">
    <groupParticipants selectParticipant="goToDetailPage_attr">
        <groupParticipant name="moveTo_declareParam"0partType="member"/>
        <groupParticipant name="moveTo_keepParams"0partType="member"/>
        <groupParticipant name="goToDetailPage_attr" partType="identifier" />
    </groupParticipants>
</group>
```

groupParticipants ブロックタグに囲まれた各 groupParticipant タグは、使用するコードブロックを含む EDML 構成要素ファイルを示します。name 属性の値は、構成要素ファイル名から .edml 拡張子を除いたものです（moveTo_declareParam 属性など）。

構成要素ファイル

構成要素は、ページ上の 1 つのコードブロックを表します。これには、サーバタグ、HTML タグ、属性などがあります。構成要素ファイルは、Dreamweaver ドキュメントの作成者が使用できるように、グループファイルに含まれている必要があります。1 つの構成要素ファイルを複数のグループファイルで使用できます。

例えば、moveTo_declareParam.edml ファイルには次のコードが含まれます。

```
<participant>
    <quickSearch><![CDATA[MM_paramName]]></quickSearch>
    <insertText location="aboveHTML+80">
<![CDATA[
<% var MM_paramName = ""; %>
]]>
    </insertText>
    <searchPatterns whereToSearch="directive">
        <searchPattern><![CDATA[/var\s*MM_paramName/]]></searchPattern>
    </searchPatterns>
</participant>
```

ドキュメントにサーバビヘイビアを追加する際は、コードの挿入先、コードの内容、Dreamweaver の作成者またはデータによってランタイムに置き換えられるパラメータなどの詳細情報が必要です。各 EDML 構成要素ファイルには、このような詳細が各コードブロックごとに記述されています。構成要素ファイルに記述される具体的なデータは次のとおりです。

- コードおよび固有のインスタンスの挿入先は、次の例に示すように、insertText タグパラメータで定義されます。

  ```
  <insertText location="aboveHTML+80">
  ```

- 既にページ上にあるインスタンスを認識する方法は、次の例に示すように、searchPatterns タグで定義されます。

  ```
  <searchPatterns whereToSearch="directive">
      <searchPattern><![CDATA[/var\s*MM_paramName/]]></searchPattern>
  </searchPatterns>
  ```

  searchPatterns ブロックタグ内の各 searchPattern タグでは、ランタイムコードのインスタンスの検索および特定パラメータの抽出に使用されるパターンが指定されます。詳しくは、276 ページの「サーバビヘイビアの手法」を参照してください。

スクリプトファイル

各サーバビヘイビアには、サーバビヘイビアコードと Dreamweaver インターフェイスの統合を制御するスクリプトへのリンクおよび関数を含んだ HTML ファイルもあります。このファイルの編集に使用できる関数については、249 ページの「サーバビヘイビアの実装関数」を参照してください。

簡単なサーバビヘイビアの例

この例では、新しいサーバビヘイビアを作成するプロセスを示します。この例から、Dreamweaver で生成されるファイルとその処理方法がわかります。サーバビヘイビアビルダーインターフェイスの操作について詳しくは、『Dreamweaver ファーストステップガイド』の「カスタムサーバビヘイビアの追加」を参照してください。この例では、ASP サーバから「Hello World」と表示されます。Hello World ビヘイビアは、構成要素を 1 つだけ（1 つの ASP タグ）備え、ページでの修正や追加は行いません。

ビヘイビアを作成するには、動的ページドキュメントを作成し、新しいサーバビヘイビアを定義し、挿入するコードを定義します。

動的ページドキュメントの作成

1. Dreamweaver で、ファイル／新規メニューオプションを選択します。
2. 新規ドキュメントダイアログボックスで、カテゴリーに「ダイナミックページ」、動的ページに「ASP JavaScript」を選択します。
3. 「作成」をクリックします。

新しいサーバビヘイビアの定義

注意：サーバビヘイビアパネルが表示されていない場合は、ウィンドウ／サーバビヘイビアのメニューオプションを選択します。

1 サーバビヘイビアパネルで、「+」ボタンをクリックし、「新規サーバビヘイビア」メニューオプションを選択します。

2 新規サーバビヘイビアダイアログボックスで、ドキュメントタイプとして「ASP JavaScript」、名前として「Hello World」を選択します（「既存のサーバビヘイビアをコピー」チェックボックスはオフのままにしておきます）。

3 「OK」をクリックします。

挿入するコードの定義

1 挿入するコードブロックの「+」ボタンをクリックします。

2 新規コードブロックの作成ダイアログボックスで、「**Hello_World_block1**」と入力します（この情報は自動的に入力される場合があります）。

3 「OK」をクリックします。

4 「コードブロック」テキストフィールドに、「**<% Response.Write("Hello World") %>**」と入力します。

5 コードの挿入ポップアップメニューで、「選択範囲に対する相対位置」を選択して、ユーザがドキュメントでのコードの挿入位置を制御できるようにします。

6 相対位置ポップアップメニューで、「選択範囲の後」を選択します。

7 「OK」をクリックします。

サーバビヘイビアパネルで、+ メニューのポップアップリストに新しいサーバビヘイビアが表示されます。また、Dreamweaver ファイルのインストールフォルダ内の Configuration¥ServerBehaviors¥ASP_Js フォルダに、次の 3 つのファイルが追加されています。

- グループファイル：Hello World.edml
- 構成要素ファイル：Hello World_block1.edml
- スクリプトファイル：Hello World.htm

注意：マルチユーザ設定で作業している場合、これらのファイルは Application Data フォルダに配置されます。

サーバビヘイビア API 関数が呼び出される状況

サーバビヘイビア API 関数は、次のような場合に呼び出されます。

- findServerBehaviors() 関数は、ドキュメントを開くときと、構成要素が編集されるときに呼び出されます。この関数はユーザドキュメント内でサーバビヘイビアのインスタンスを検索します。findServerBehaviors() 関数は、検出した各インスタンスに JavaScript オブジェクトを作成し、JavaScript プロパティを使用して状態情報をオブジェクトに追加します。
- analyzeServerBehavior() 関数が実装されている場合は、findServerBehaviors() 関数がすべて呼び出された後で、ユーザのドキュメントで検出されたビヘイビアインスタンスごとにこの関数が呼び出されます。

  findServerBehaviors() 関数は、通常、ビヘイビアオブジェクトを作成する際に、incomplete、participants、selectedNode、title の 4 つのプロパティを設定します。ただし、プロパティの一部は、他のすべてのサーバビヘイビアが自身のインスタンスを検出するまで設定しない方が簡単な場合もあります。例えば、「次のレコードへ移動」ビヘイビアには、リンクオブジェクトとレコードセットオブジェクトの 2 つの構成要素があります。このビヘイビアの findServerBehaviors() 関数でレコードセットオブジェクトを検索するよりも、レコードセットビヘイビアの findServerBehaviors() 関数が実行されるまで待つ方が簡単です。これは、レコードセットによって、それ自体のインスタンスがすべて検索されるためです。

「次のレコードへ移動」ビヘイビアの analyzeServerBehavior() 関数が呼び出されると、ドキュメント内のすべてのサーバビヘイビアオブジェクトを含む配列が取得されます。次に、この配列内でレコードセットオブジェクトが検索されます。

解析中に、複数のビヘイビアによって、ユーザのドキュメントの単一タグがそのビヘイビアのインスタンスとして識別される場合があります。例えば、「動的属性」ビヘイビアの findServerBehaviors() 関数によって、ユーザのドキュメントの input タグに関連付けられている「動的属性」ビヘイビアのインスタンスが検出され、同時に、「ダイナミックテキストフィールド」ビヘイビアの findServerBehaviors() 関数によって、同じ input タグが検査され、「ダイナミックテキストフィールド」ビヘイビアのインスタンスが検出されたとします。この場合、サーバビヘイビアパネルには、「動的属性」ブロックと「ダイナミックテキストフィールド」の両方が表示されます。この問題を修正するには、analyzeServerBehavior() 関数で、重複するサーバビヘイビアを 1 つ残してすべて削除する必要があります。

サーバビヘイビアを削除するには、analyzeServerBehavior() 関数を使用して、任意のサーバビヘイビアの deleted プロパティを true に設定します。deleted プロパティが true に設定された状態で analyzeServerBehavior() 関数が呼び出されると、そのビヘイビアはリストから削除されます。

- ユーザがサーバビヘイビアパネルの「+」ボタンをクリックすると、ポップアップメニューが表示されます。

  このメニューの内容を決定するために、まずビヘイビアと同じフォルダにある ServerBehaviors.xml ファイルが検索されます。ServerBehaviors.xml ファイルでは、メニューに表示する HTML ファイルが参照されています。

  参照されている HTML ファイルに title タグが含まれている場合は、その title タグのコンテンツがメニューに表示されます。例えば、ServerBehaviors/ASP_Js/ GetRecords.htm ファイルに <title>Get More Records</title> というタグが含まれている場合は、「**Get More Records**」というテキストがメニューに表示されます。

  ファイルに title タグが含まれていない場合は、ファイル名がメニューに表示されます。例えば、GetRecords.htm ファイルに title タグがない場合は、GetRecords というテキストがメニューに表示されます。

  ServerBehaviors.xml ファイルがない場合、またはフォルダに ServerBehaviors.xml ファイルに定義されていない HTML ファイルが含まれる場合は、各ファイルの title タグがチェックされ、title タグまたはファイル名がメニューに表示されます。

  ServerBehaviors フォルダ内のファイルをメニューに表示しない場合は、HTML ファイルの最初の行に次のステートメントを挿入します。

  ```
  <!-- MENU-LOCATION=NONE -->
  ```

- ユーザがメニューから項目を選択すると、canApplyServerBehavior() 関数が呼び出されます。この関数から true が返されると、ダイアログボックスが表示されます。ユーザが「OK」をクリックすると、applyServerBehavior() 関数が呼び出されます。
- ユーザが既存のサーバビヘイビアをダブルクリックして編集すると、ダイアログボックスが表示され、onLoad ハンドラが body タグにある場合はそれが実行され、さらに inspectServerBehavior() 関数が呼び出されます。inspectServerBehavior() 関数により、現在の引数値でフォームエレメントが構成されます。ユーザが「OK」をクリックすると、applyServerBehavior() 関数が再度呼び出されます。
- ユーザが「-」ボタンをクリックすると、deleteServerBehavior() 関数が呼び出されます。deleteServerBehavior() 関数によって、ドキュメントからビヘイビアが削除されます。
- ユーザがサーバビヘイビアを選択して「カット」または「コピー」コマンドを使用すると、サーバビヘイビアを表すオブジェクトがそのサーバビヘイビアの copyServerBehavior() 関数に渡されます。copyServerBehavior() 関数によって、後でペーストするために必要なその他のプロパティがサーバビヘイビアオブジェクトに追加されます。

  copyServerBehavior() 関数の動作が終了すると、サーバビヘイビアオブジェクトがクリップボードに配置できるフォームに変換されます。オブジェクトが変換されると、オブジェクトを参照しているすべてのプロパティが削除されます。数値、ブール値またはストリング以外のオブジェクトのプロパティはすべて失われます。

  ユーザが「ペースト」コマンドを使用すると、クリップボードの内容がアンパックされ、新しいサーバビヘイビアオブジェクトが生成されます。新しいオブジェクトは、オブジェクトを参照するプロパティがない点を除き、元のオブジェクトと同じです。新しいサーバビヘイビアオブジェクトが pasteServerBehavior() 関数に渡されます。渡されたビヘイビア

は、pasteServerBehavior() 関数によってユーザのドキュメントに挿入されます。pasteServerBehavior() 関数の動作が終了すると、findServerBehaviors() 関数が呼び出され、ユーザのドキュメント内にあるすべてのサーバビヘイビアの新しいリストが取得されます。

ビヘイビアはドキュメント間でコピー＆ペーストできます。copyServerBehavior() 関数と pasteServerBehavior() 関数は、ビヘイビアオブジェクトのプロパティのみに依存して情報を交換します。

サーバビヘイビア API

サーバビヘイビアを制御するには、次の API 関数を使用します。

analyzeServerBehavior()

対応バージョン
Dreamweaver UltraDev 1（英語版）

説明
サーバビヘイビアでその incomplete プロパティと deleted プロパティが設定されるようにします。

ページに含まれるすべてのサーバビヘイビアに対して findServerBehaviors() 関数が呼び出されると、ユーザのドキュメントにあるすべてのビヘイビアの配列が表示されます。この配列に含まれる JavaScript オブジェクトごとに、analyzeServerBehavior() 関数が呼び出されます。例えば、「ダイナミックテキスト」ビヘイビアに対しては、DynamicText.htm ファイルまたは DynamicText.js ファイルの analyzeServerBehavior() 関数が呼び出されます。

analyzeServerBehavior() 関数を使用する目的の 1 つは、ビヘイビアオブジェクトのすべてのプロパティ（incomplete、participants、selectedNode および title）の設定を完了することです。場合によっては、findServerBehaviors() 関数でユーザのドキュメントにあるサーバビヘイビアの完全なリストが生成された後の方が、このタスクを簡単に実行できます。

analyzeServerBehavior() 関数を使用するもう 1 つの目的は、複数のビヘイビアがユーザのドキュメント内の同じタグを参照している場合にそれを通知することです。この場合は、1 つを除くすべてのビヘイビアが deleted プロパティによって配列から削除されます。

Recordset1、DynamicText1、DynamicText2 の各サーバビヘイビアがページに存在するとします。DynamicText1 と DynamicText2 のサーバビヘイビアは、どちらもページ上に Recordset1 を必要とします。findServerBehaviors() 関数でこれらのサーバビヘイビアが検出されると、3 つのサーバビヘイビアに対して analyzeServerBehavior() 関数が呼び出されます。analyzeServerBehavior() 関数が DynamicText1 に対して呼び出されると、ページ上のすべてのサーバビヘイビアオブジェクトの配列で、Recordset1 に属するサーバビヘイビアオブジェクトが検索されます。Recordset1 に属するサーバビヘイビアオブジェクトが見つからない場合は、incomplete プロパティが true に設定され、問題があることをユーザに警告する感嘆符がサーバビヘイビアパネルに表示されます。同様に、analyzeServerBehavior() 関数が DynamicText2 に対して呼び出された場合も、Recordset1 に属するオブジェクトが検索されます。Recordset1 は他のサーバビヘイビアに依存しないので、この例では analyzeServerBehavior() 関数を定義する必要はありません。

引数
serverBehavior、{**serverBehaviorArray**}

- **serverBehavior** 引数は、解析されるビヘイビアを表す JavaScript オブジェクトです。
- {***serverBehaviorArray***} 引数は、ページで検出されたすべてのサーバビヘイビアを表す JavaScript オブジェクトの配列です。

戻り値
なし。

applyServerBehavior()

対応バージョン
Dreamweaver UltraDev 1（英語版）

説明
ダイアログボックスのフォームエレメントから値を読み取り、ユーザのドキュメントにビヘイビアを追加します。ユーザがサーバビヘイビアダイアログボックスで「OK」をクリックすると、この関数が呼び出されます。この関数が正常に終了すると、サーバビヘイビアダイアログボックスは閉じます。関数が失敗した場合は、サーバビヘイビアダイアログボックスは閉じずに、エラーメッセージが表示されます。この関数では、ユーザのドキュメントを編集できます。

詳しくは、250 ページの「dwscripts.applySB()」を参照してください。

引数
serverBehavior

serverBehavior JavaScript オブジェクトはサーバビヘイビアを表し、既存のビヘイビアを修正するために必要です。新しいビヘイビアの場合、この引数は null となります。

戻り値
この関数の実行に成功した場合は空の文字列、失敗した場合はエラーメッセージ。

canApplyServerBehavior()

対応バージョン
Dreamweaver UltraDev 1（英語版）

説明
ビヘイビアが適用可能かどうかを判断します。この関数は、サーバビヘイビアダイアログボックスが表示される前に呼び出されます。この関数から true が返されると、サーバビヘイビアダイアログボックスが表示されます。false が返されると、サーバビヘイビアダイアログボックスは表示されず、サーバビヘイビアの挿入も中断されます。

引数
serverBehavior

serverBehavior JavaScript オブジェクトはビヘイビアを表し、既存のビヘイビアを修正するために必要です。新しいビヘイビアの場合、この引数は null になります。

戻り値
ブール値。ビヘイビアが適用可能である場合は true、そうでない場合は false。

copyServerBehavior()

対応バージョン
Dreamweaver UltraDev 1（英語版）

説明
copyServerBehavior() 関数を実装するかどうかは、オプションです。ユーザは、指定したサーバビヘイビアのインスタンスをコピーできます。次の例では、この関数がレコードセットに対して実装されています。ユーザがサーバビヘイビアパネルまたはデータバインディングパネルでレコードセットを選択し、「コピー」コマンドを使用すると、ビヘイビアがクリップボードにコピーされます。また、「カット」コマンドを使用すると、ビヘイビアが切り取られてクリップボードに置かれます。この関数を実装しないサーバビヘイビアでは、「コピー」コマンドおよび「カット」コマンドは機能しません。詳しくは、243 ページの「サーバビヘイビア API 関数が呼び出される状況」を参照してください。

copyServerBehavior() 関数は、ストリングに変換できるビヘイビアオブジェクトのプロパティだけに依存して、pasteServerBehavior() 関数と情報を交換します。クリップボードには未処理のテキストのみが格納されるので、ドキュメント内の participant ノードを解決し、未処理のテキストを二次プロパティに保存する必要があります。

注意：ユーザが Dreamweaver の任意のドキュメントにビヘイビアをペーストできるようにするには、pasteServerBehavior() 関数も実装する必要があります。

引数
serverBehavior

- **serverBehavior** JavaScript オブジェクトはビヘイビアを表します。

戻り値
ブール値。ビヘイビアがクリップボードに正常にコピーされた場合は true、そうでない場合は false。

deleteServerBehavior()

対応バージョン
Dreamweaver UltraDev 1（英語版）

説明
ユーザのドキュメントからビヘイビアを削除します。この関数は、ユーザがサーバビヘイビアパネルの「-」ボタンをクリックすると呼び出されます。この関数ではユーザのドキュメントを編集できます。

詳しくは、251 ページの「dwscripts.deleteSB()」を参照してください。

引数
serverBehavior

- **serverBehavior** JavaScript オブジェクトはビヘイビアを表します。

戻り値
なし。

displayHelp()

説明
この関数が定義されている場合は、ダイアログボックスの「OK」と「キャンセル」の下に「ヘルプ」ボタンが表示されます。この関数は、ユーザが「ヘルプ」ボタンをクリックすると呼び出されます。

引数
なし。

戻り値
なし。

例
```
// the following instance of displayHelp() opens
// in a browser a file that explains how to use
// the extension.
function displayHelp(){
    var myHelpFile = dw.getConfigurationPath() +
        '/ExtensionsHelp/superDuperHelp.htm';
    dw.browseDocument(myHelpFile);
}
```

findServerBehaviors()

対応バージョン
Dreamweaver UltraDev 1（英語版）

説明
ユーザのドキュメントで、この関数自体のインスタンスを検索します。findServerBehaviors() 関数は、検出した各インスタンスに対して JavaScript オブジェクトを作成し、オブジェクトの JavaScript プロパティとして状態情報を追加します。

必須のプロパティは、incomplete、participants、title、selectedNode の 4 つです。必要に応じてその他のプロパティも設定できます。

詳しくは、250 ページの「dwscripts.findSBs()」および「dreamweaver.getParticipants()」（『Dreamweaver API リファレンス』）を参照してください。

引数
なし。

戻り値
JavaScript オブジェクトの配列。配列の長さは、ページで検出されたビヘイビアインスタンスの数に一致します。

inspectServerBehavior()

対応バージョン
Dreamweaver UltraDev 1（英語版）

説明
指定されたビヘイビアオブジェクトに基づいて、サーバビヘイビアダイアログボックスの設定を決定します。ユーザがサーバビヘイビアダイアログボックスを開くと、inspectServerBehavior() 関数が呼び出されます。ただし、この関数が呼び出されるのは、ユーザが既存のビヘイビアを編集するときに限られます。

引数
serverBehavior

serverBehavior 引数は、ビヘイビアを表す JavaScript オブジェクトです。これは、findServerBehaviors() 関数から返されるオブジェクトと同じです。

戻り値
なし。

pasteServerBehavior()

対応バージョン
Dreamweaver UltraDev 1（英語版）

説明
pasteServerBehavior() 関数が実装されている場合は、この関数を使用して、指定したサーバビヘイビアのインスタンスをペーストできます。ユーザがサーバビヘイビアをペーストすると、クリップボードの内容が整理され、新しいビヘイビアオブジェクトが生成されます。新しいオブジェクトは、ポインタのプロパティがない点を除き、元のオブジェクトとまったく同じです。新しいビヘイビアオブジェクトは pasteServerBehavior() 関数に渡されます。pasteServerBehavior() 関数は、ビヘイビアオブジェクトのプロパティに基づいてユーザのドキュメントに挿入する内容を決定します。次に、渡されたビヘイビアは、pasteServerBehavior() 関数によってユーザのドキュメントに挿入されます。pasteServerBehavior() が終了すると、findServerBehaviors() 関数が呼び出されて、ユーザのドキュメントにあるすべてのサーバビヘイビアの新しいリストが取得されます。

pasteServerBehavior() 関数を実装するかどうかは、オプションです。詳しくは、243 ページの「サーバビヘイビア API 関数が呼び出される状況」を参照してください。

注意：この関数を実装する場合は、copyServerBehavior() 関数も実装する必要があります。

引数
behavior

behavior JavaScript オブジェクトは、ビヘイビアを表します。

戻り値
ブール値。ビヘイビアがクリップボードから正常にペーストされた場合は true、そうでない場合は false。

サーバビヘイビアの実装関数

ここに挙げる関数では、HTML スクリプトファイル内または HTML スクリプトファイルに示された指定の JavaScript ファイル内で挿入や編集を行うことができます。

dwscripts.findSBs()

対応バージョン
Dreamweaver MX（この関数によって、Dreamweaver の前バージョンの findSBs() 関数が置き換えられています）

説明
現在のページにあるサーバビヘイビアのすべてのインスタンスとすべての構成要素を検索します。タイトル、タイプ、participants 配列、weights 配列、types 配列、selectedNode 値および incomplete フラグを設定します。また、レコードセット、名前、列名などのユーザ定義可能なプロパティの配列を保持するパラメータオブジェクトも作成します。この配列は、findServerBehaviors() 関数を使用して取得できます。

引数
serverBehaviorTitle

serverBehaviorTitle 引数は、EDML タイトルにタイトルが指定されていないときに使用されるオプションのタイトルストリングです。これは、ローカリゼーションに便利です。

戻り値
必要なプロパティが定義されている JavaScript オブジェクトの配列。ページにサーバビヘイビアのインスタンスがない場合は、空白の配列が返されます。

例
以下の例では、現在のユーザのドキュメントに含まれる特定のサーバビヘイビアのすべてのインスタンスを検索します。

```
function findServerBehaviors() {
    allMySBs = dwscripts.findSBs();
    return allMySBs;
}
```

dwscripts.applySB()

対応バージョン
Dreamweaver MX（この関数によって、Dreamweaver の前バージョンの applySB() 関数が置き換えられています）

説明
サーバビヘイビアのランタイムコードを挿入または更新します。**sbObj** 引数の値が null の場合は、新しいランタイムコードが挿入されます。それ以外の場合は、**sbObj** オブジェクトで示される既存のランタイムコードが更新されます。ユーザ設定は JavaScript オブジェクトのプロパティとして設定し、**paramObj** として渡す必要があります。これらの設定は、EDML の挿入テキストに @@paramName@@ として宣言されているすべての引数と一致している必要があります。

引数
paramObj、**sbObj**

- **paramObj** 引数は、ユーザ設定が含まれているオブジェクトです。
- **sbObj** 引数は、既存のサーバビヘイビアを更新する場合は既存のサーバビヘイビアオブジェクト、それ以外の場合は null です。

戻り値
ブール値。サーバビヘイビアが正常にユーザのドキュメントに挿入された場合は true、そうでない場合は false。

例
次の例では、paramObj オブジェクトにユーザの入力値を挿入してから、dwscripts.applySB 関数を呼び出して、これらの入力値と sbObj サーバビヘイビアを渡します。

```
function applyServerBehaviors(sbObj) {
    // get all UI values here...
    paramObj = new Object();
    paramObj.rs= rsName.value;
    paramObj.col = colName.value;
    paramObj.url = urlPath.value;
    paramObj.form__tag = formObj;
     dwscripts.applySB(paramObj, sbObj);
}
```

dwscripts.deleteSB()

対応バージョン
Dreamweaver MX（この関数によって、Dreamweaver の前バージョンの deleteSB() 関数が置き換えられています）

説明
sbObj サーバビヘイビアインスタンスの構成要素をすべて削除します。EDML ファイルに delete タグを使って特別な削除命令が記述されていない限り、構成要素全体が削除されます。複数のサーバビヘイビアインスタンスに属している構成要素（参照カウントが 2 以上）は削除されません。

引数
sbObj

- **sbObj** 引数は、ユーザのドキュメントから削除するサーバビヘイビアのオブジェクトインスタンスです。

戻り値
なし。

例
次の例では、EDML ファイルの sbObj サーバビヘイビアのすべての構成要素を削除します。ただし、delete タグで保護されているものを除きます。

```
function deleteServerBehavior(sbObj) {
    dwscripts.deleteSB(sbObj);
}
```

EDML ファイル

ファイルを編集するときは、Dreamweaver のコーディング規則に従う必要があります。要素間の依存関係には注意してください。例えば、挿入されるタグを更新する場合は、検索パターンも更新する必要があります。

注意：EDML ファイルは Dreamweaver MX の新機能です。旧バージョンのサーバビヘイビアの操作については、以前のバージョンの『Dreamweaver 拡張ガイド』を参照してください。

正規表現

JavaScript 1.5 で実装されている正規表現について理解する必要があります。また、サーバビヘイビアの EDML ファイルでの適切な正規表現の使用についても把握しておく必要があります。例えば、quickSearch の値では正規表現を使用できませんが、searchPattern タグでデータを検索して抽出する際には、正規表現を使用します。

正規表現でテキストストリングを記述するときは、特別な意味が割り当てられた文字（メタキャラクタ）を使用し、事前に定義されたルールに従って、テキストの表現、分割、処理を行います。正規表現は、一般化された方法でパターンを表現できる強力な解析ツールであり処理ツールです。

JavaScript 1.5 に関する参考書には、必ず正規表現に関する節または章があります。この節では、Dreamweaver サーバビヘイビアの EDML ファイルで、正規表現を使用してランタイムコード内の引数を検索し、その値を抽出する方法について説明します。ユーザがサーバビヘイビアを編集するたびに、ランタイムコードのインスタンスから事前引数値を抽出する必要があります。抽出プロセスには正規表現を使用します。

サーバビヘイビアの EDML ファイルで役立つ数種類のメタキャラクタおよびメタシーケンス（特殊文字のグループ）について理解しておく必要があります。これらを以下の表に示します。

正規表現	説明
\	特殊文字をエスケープします。例えば「\.」と記述すると、メタキャラクタである「.」をリテラル文字として使用できます。同様に、「/」や「)」をリテラル文字として使用するには、「\/」および「\)」のように記述します。
/ ... /i	メタシーケンスを検索する際に、大文字と小文字を区別しません。
(...)	メタシーケンス内に、括弧で囲んだサブ表現を作成します。
\s*	ホワイトスペースを検索します。

<searchPatterns whereToSearch="directive"> という EDML タグは、ランタイムコードの検索が必要であることを宣言しています。各 <searchPattern>...</searchPattern> サブタグでは、ランタイムコード内で識別が必要な 1 つのパターンが定義されます。「Redirect If Empty」の例では、2 つのパターンが存在します。

次の例では、<% if (@@rs@@.EOF) Response.Redirect("@@new__url@@"); %> から引数値を抽出するために、rs および new__url というストリングを識別する正規表現を記述します。

```
<searchPattern paramNames="rs,new__url">
    /if d ((\w+)\.EOF\) Response\.Redirect\("([^\r\n]*)"\)/i
</searchPattern>
```

この処理では、ユーザのドキュメントが検索され、一致するストリングがあれば引数値が抽出されます。括弧で囲まれた最初のサブ表現（\w+）では、rs の値が抽出されます。括弧で囲まれた 2 番目のサブ表現（[^\r\n]*）では、new_url の値が抽出されます。

注意：文字シーケンス「[^\r\n]*」は、Macintosh および Windows 環境で、改行以外の任意の文字と一致します。

EDML 構造

サーバビヘイビアグループを識別するには、固有のファイル名を使用する必要があります。関連する構成要素ファイルを使用するグループファイルが 1 つしかない場合は、その構成要素ファイル名とグループ名を同じにします。この方法を使用すると、サーバビヘイビアグループファイル updateRecord.edml が構成要素ファイル updateRecord_init.edml と連動するようになります。複数のサーバビヘイビアグループで構成要素ファイルを共有する場合は、一意の記述名を割り当てます。

注意：EDML ネームスペースは、フォルダ構造とは無関係に共有されるので、一意のファイル名を使用してください。Macintosh での制限のため、ファイル名は .edml 拡張子を含めて 31 文字以内で指定する必要があります。

サーバビヘイビアのランタイムコードは、EDML ファイルに含まれています。EDML パーサーでランタイムコードと EDML マークアップが混同されないように、CDATA タグでランタイムコードを囲む必要があります。CDATA タグは、文字データ、つまり EDML マークアップ以外のすべてのテキストを表します。CDATA タグを使用すると、EDML パーサーはこれをマークアップとして解釈せず、プレーンテキストのブロックと見なします。CDATA でマークするブロックは、<![CDATA[で開始し、]]> で終了します。

次の例で示すように、**Hello, World** というテキストを挿入する場合、EDML は簡単に指定できます。

```
<insertText>Hello, World</insertText>
```

ただし、内部に <img src='foo.gif'> などのタグを含むコンテンツを挿入すると、EDML パーサーで混乱が生じる可能性があります。この場合は、次の例で示すように、CDATA 構文にそのタグを埋め込みます。

```
<insertText><![CDATA[<img src='foo.gif'>]]></insertText>
```

次の例で示すように、ASP ランタイムコードは CDATA タグ内に含まれます。

```
<![CDATA[
    <% if (@@rs@@.EOF) Response.Redirect("@@new__url@@"); %>
]]
```

CDATA タグがあるために、<%= %> という ASP タグとそのタグ内にある他のコンテンツは処理されません。代わりに、次の例で示すように、EDM（Extension Data Manager）が未解釈のテキストを受け取ります。

```
<% if (Recordset1.EOF) Response.Redirect("http://www.Adobe.com"); %>
```

以下の EDML 定義は、CDATA タグの適切な使用例を示しています。

EDML グループファイルタグ

ここに挙げるタグと属性は、EDML グループファイル内で有効です。

<group>

説明
このタグは、構成要素のグループに関するすべての指定を含んでいます。

親
なし

タイプ
ブロックタグ

必須
はい

<group> の属性

次の項目は、グループタグの有効な属性です。

version

説明
この属性は、現在のサーバビヘイビアターゲットを処理する Dreamweaver サーバビヘイビアのバージョンを定義します。Dreamweaver CS3 の場合、バージョン番号は 9 です。バージョンが指定されていない場合は、バージョン 7 と想定されます。このリリースの Dreamweaver では、サーバビヘイビアビルダーで作成されるすべてのグループと構成要素の version 属性が 9.0 に設定されます。この属性のグループバージョンは、現時点では効力がありません。

親
group

タイプ
属性

必須
いいえ

serverBehavior

説明
serverBehavior 属性は、グループを使用できるサーバビヘイビアを示します。ドキュメント内でグループの構成要素の quickSearch ストリングが検出されると、serverBehavior 属性で示されるサーバビヘイビアにより、findServerBehaviors() 関数が呼び出されます。

1 つのサーバビヘイビアに複数のグループが関連付けられているときは、サーバビヘイビアで使用するグループを特定しなければならない場合があります。

親
group

タイプ
属性

必須
いいえ

値
値は、次の例で示すように、Configuration¥ServerBehaviors フォルダに含まれる任意のサーバビヘイビア HTML ファイルの正確な名前（パスを除く）です。

```
<group serverBehavior="redirectIfEmpty.htm">
```

dataSource

説明
この高度な機能では、Dreamweaver に追加される可能性のある新しいデータソースがサポートされます。

使用されるデータソースに応じて、サーバビヘイビアのバージョンを変えることができます。例えば、「リピート領域」サーバビヘイビアは、標準の Recordset.htm データソース向けに設計されています。Dreamweaver が拡張されて新しいタイプのデータソース（COM オブジェクトなど）がサポートされるようになった場合は、新しいリピート領域を実装するグループファイルで dataSource="COM.htm" を設定できます。新しいデータソースを選択すると、「リピート領域」サーバビヘイビアによって、新しいリピート領域の実装が適用されます。

親
group

タイプ
属性

必須
いいえ

値
次の例で示すような、Configuration¥DataSources フォルダに含まれるデータソースファイルの正確な名前。

```
<group serverBehavior="Repeat Region.htm" dataSource="myCOMdataSource.htm">
```

このグループでは、COM データソースを使用する場合に、使用される「リピート領域」サーバビヘイビアの新しい実装を定義します。applyServerBehaviors() 関数では、次の例で示すように、パラメータオブジェクトの MM_dataSource プロパティを設定することにより、このグループを適用するように指定できます。

```
function applyServerBehavior(ssRec) {
    var paramObj = new Object();
    paramObj.rs = getComObjectName();
    paramObj.MM_dataSource = "myCOMdataSource.htm";

    dwscripts.applySB(paramObj, sbObj);
}
```

subType

説明
この高度な機能では、サーバビヘイビアの複数の実装がサポートされます。

ユーザの選択に応じて、サーバビヘイビアのバージョンを変えることができます。複数のグループファイルが関連するサーバビヘイビアを適用する場合は、subType 値を渡すことで適切なグループファイルを選択できます。指定された subType 値を持つグループが適用されます。

親
group

タイプ
属性

必須
いいえ

値
次の例で示すような、適用されるグループを決定する一意のストリング。

```
<group serverBehavior="myServerBehavior.htm" subType="longVersion">
```

このグループ属性は、myServerBehavior サブタイプの長いバージョンを定義します。subType="shortVersion" 属性で特定されるバージョンもあります。applyServerBehaviors() 関数では、次の例で示すように、パラメータオブジェクトの MM_subType プロパティを設定することにより、このグループを適用するように指定できます。

```
function applyServerBehavior(ssRec) {
    var paramObj = new Object();
    if (longVersionChecked) {
        paramObj.MM_subType = "longVersion";
    } else {
        paramObj.MM_subType = "shortVersion";
    }
    dwscripts.applySB(paramObj, sbObj);
}
```

<title>

説明
このストリングは、現在のドキュメントで検出されたサーバビヘイビアインスタンスごとに、サーバビヘイビアパネルに表示されます。

親
group

タイプ
ブロックタグ

必須
いいえ

値
値はプレーンテキストのストリングです。次の例で示すように、各インスタンスを一意にするために引数名を含むことができます。

```
<title>Redirect If Empty (@@recordsetName@@)</title>
```

<groupParticipants>

説明
このタグは、groupParticipant 宣言の配列を含んでいます。

親
group

タイプ
ブロックタグ

必須
はい

<groupParticipants> の属性

次の項目は、groupParticipants タグの有効な属性です。

selectParticipant

説明
サーバビヘイビアパネルでインスタンスが選択されたときに、ドキュメント内で選択されてハイライト表示される構成要素を指定します。このパネルに表示されるサーバビヘイビアインスタンスは、選択された構成要素の順に並べられるので、構成要素が表示されない場合でも selectParticipant 属性を設定する必要があります。

親
groupParticipants

タイプ
属性

必須
いいえ

値
次の例で示すように、**participantName** の値は、グループの構成要素として表示される 1 つの構成要素ファイルの正確な名前（.edml 拡張子を除く）です。詳しくは、258 ページの「name」を参照してください。

```
<groupParticipants selectParticipant="redirectIfEmpty_link">
```

<groupParticipant>

説明
このタグは、グループに構成要素が 1 つ含まれていることを表します。

親
groupParticipants

タイプ
タグ

必須
はい（少なくとも 1 つが必要）

<groupParticipant> の属性

次の項目は、groupParticipant タグの有効な属性です。

name

説明
この属性は、グループに取り込まれる特定の構成要素を指定します。このタグの name 属性は、構成要素のファイル名から .edml ファイル拡張子を除いたものと同じにする必要があります。

親
groupParticipant

タイプ
属性

必須
はい

値
次の例で示すように、値は任意の構成要素ファイルの正確な名前（.edml 拡張子を除く）です。

```
<groupParticipant name="redirectIfEmpty_init">
```

この例では、redirectIfEmpty_init.edml ファイルを参照しています。

partType

説明
この属性は、構成要素のタイプを指定します。

親
groupParticipant

タイプ
属性

必須
いいえ

値
identifier、**member**、**option**、**multiple**、**data**

- **identifier** は、グループ全体を識別する構成要素です。この構成要素がドキュメント内で検出されると、そのグループは存在すると見なされます。partType 属性が指定されていない場合は、これがデフォルト値になります。
- **member** は、グループの通常メンバーです。単独で検出された場合は、グループが識別されません。グループ内で検出されないと、そのグループは不完全であると見なされます。
- **option** は、構成要素がオプションであることを示します。この構成要素が検出されなくても、グループは完全であると見なされ、サーバビヘイビアパネルに incomplete フラグは設定されません。
- **multiple** は、構成要素がオプションであり、その構成要素の複数のコピーがサーバビヘイビアに関連付けられる可能性があることを示します。構成要素をグループ化するときは、値がそれぞれに異なる可能性があるため、この構成要素に一意の引数は使用されません。

- **data** は、プログラマが他のグループデータのリポジトリとして使用する非標準の構成要素です。それ以外の場合は無視されます。

EDML 構成要素ファイル

ここに挙げるタグと属性は、EDML 構成要素ファイル内で有効です。

<participant>

説明
このタグは、1 つの構成要素に関するすべての指定を含んでいます。

親
なし

タイプ
ブロックタグ

必須
はい

<participant> の属性

次の項目は、participant タグの有効な属性です。

version

説明
この属性は、現在のサーバビヘイビアターゲットを処理する Dreamweaver サーバビヘイビアのバージョンを定義します。Dreamweaver CS3 の場合、バージョン番号は 9 です。バージョンが指定されていない場合は、バージョン 7 と想定されます。このリリースの Dreamweaver では、サーバビヘイビアビルダーで作成されるすべてのグループと構成要素の version 属性が 0.0 に設定されます。

注意：構成要素の version 属性がグループの version 属性と異なる場合は、構成要素の version 属性が優先されます。ただし、構成要素で version 属性が指定されていない場合は、構成要素ファイルでグループの version 属性が使用されます。

構成要素ファイルでは、この属性によってコードブロックをマージするかどうかが決まります。この属性がないか、バージョン 4 以前に設定されている構成要素では、挿入されたコードブロックはページ上の他のコードブロックとマージされません。バージョン 5 以降に設定されている構成要素は、可能であればページ上の他のコードブロックとマージされます。HTML タグの上下の構成要素でのみ、コードブロックがマージされます。

親
participant

タイプ
属性

必須
いいえ

<quickSearch>

説明
このタグは、パフォーマンス上の理由から使用される単純な検索ストリングです。正規表現で指定することはできません。このストリングが現在のドキュメントで検出された場合は、残りの検索パターンが呼び出されて特定のインスタンスが検索されます。このストリングを空白にして、常に検索パターンを使用することができます。

親
participant

タイプ
ブロックタグ

必須
いいえ

値
searchString は、構成要素が存在する場合にページ上に表示されるリテラルストリングです。高いパフォーマンスを得るためには、このストリングをできるだけ一意にする必要がありますが、完全に一意でなくてもかまいません。このストリングでは大文字と小文字は区別されません。ただし、次の例で示すように、ユーザが変更できる不要なスペースには注意してください。

```
<quickSearch>Response.Redirect</quickSearch>
```

quickSearch タグが空白の場合は一致していると見なされ、searchPattern タグに定義されている正規表現を使用して、さらに正確な検索が行われます。これは、単純なストリングでは確実な検索パターンを表現できないために正規表現が必要となる場合に便利です。

<insertText>

説明
このタグは、ドキュメントに挿入する内容および挿入する位置に関する情報を提供します。これには、挿入するテキストが含まれます。テキストの中でカスタマイズする部分は、@@parameterName@@ という形式で指定します。

トランスレータのみの構成要素など、このタグを必要としない場合もあります。

親
implementation

タイプ
ブロックタグ

必須
いいえ

値

値は、ドキュメントに挿入されるテキストです。テキストにカスタマイズの必要な部分が含まれている場合は、その部分を後からパラメータとして渡すことができます。引数は、2 つの @ 記号（@@）で囲む必要があります。このテキストは EDML 構造と混同される可能性があるため、次の例で示すように CDATA 構造体を使用してください。

```
<insertText location="aboveHTML">
    <![CDATA[<%= @@recordset@@).cursorType %>]]>
</insertText>
```

テキストが挿入される際に、@@recordset@@ 引数がユーザの指定したレコードセット名で置き換えられます。条件付きコードブロックおよびコードブロックの繰り返しについて詳しくは、『Dreamweaver ファーストステップガイド』の「カスタムサーバビヘイビアの追加」を参照してください。

<insertText> の属性

次の項目は、insertText タグの有効な属性です。

location

説明

この属性は、構成要素テキストを挿入する位置を指定します。挿入位置は、whereToSearch 属性（searchPatterns タグ）に関係するため、注意して両方の属性を設定してください（詳しくは、263 ページの「whereToSearch」を参照してください）。

親

insertText

タイプ

属性

必須

はい

値

aboveHTML[+***weight***]、**belowHTML**[+***weight***]、**beforeSelection**、**replaceSelection**、**wrapSelection**、**afterSelection**、**beforeNode**、**replaceNode**、**afterNode**、**firstChildOfNode**、**lastChildOfNode**、**nodeAttribute[+attribute]**

- **aboveHTML**[+***weight***] を指定すると、テキストが HTML タグの上に挿入されます。これはサーバコードにのみ適しています。ウエイト（weight）には 1 ～ 99 の整数を指定できます。ウエイトは、異なる構成要素間の相対順序を保持するために使用されます。レコードセットのウエイトは 50 であるため、次の例で示すように、レコードセット変数を参照する構成要素のウエイトをそれより重く（60 など）指定して、コードがレコードセットより下に挿入されるようにします。

  ```
  <insert location="aboveHTML+60">
  ```

  ウエイトを指定しないと、内部的に 100 のウエイトが割り当てられ、次の例で示すように、ウエイトが指定されたすべての構成要素の下にその構成要素が追加されます。

  ```
  <insert location="aboveHTML">
  ```

- **belowHTML**[+***weight***] は aboveHTML に似ていますが、構成要素は終了 /HTML タグの下に追加されます。
- **beforeSelection** を指定すると、現在の選択範囲または挿入ポイントの前にテキストが挿入されます。選択範囲がない場合は、body タグの終わりにテキストが挿入されます。

- **replaceSelection** を指定すると、現在の選択範囲がテキストで置換されます。選択範囲がない場合は、body タグの終わりにテキストが挿入されます。
- **wrapSelection** を指定すると、現在の選択範囲が均衡化され、その選択範囲の前にテキストが挿入され、さらに選択範囲の後に適切な終了タグが追加されます。
- **afterSelection** を指定すると、現在の選択範囲または挿入ポイントの後ろにテキストが挿入されます。選択範囲がない場合は、body タグの終わりにテキストが挿入されます。
- **beforeNode** を指定すると、ノードの前（DOM 内の特定の位置）にテキストが挿入されます。dwscripts.applySB() などの関数を呼び出して挿入する場合は、**paramObj** パラメータとしてノードポインタを渡す必要があります。このパラメータのユーザ定義名を、nodeParamName 属性で指定する必要があります（262 ページの「nodeParamName」を参照してください）。

  つまり、指定する位置に node という単語が含まれている場合は、必ず nodeParamName タグを宣言してください。
- **replaceNode** を指定すると、ノードがテキストで置換されます。
- **afterNode** を指定すると、ノードの後ろにテキストが挿入されます。
- **FORM** タグの先頭にテキストを挿入する場合などに firstChildOfNode を指定すると、ブロックタグの最初の子としてテキストが挿入されます。
- **FORM** タグの最後にコードを挿入する場合（非表示のフォームフィールドを追加する場合に便利です）などに lastChildOfNode を指定すると、テキストはブロックタグの最後の子として挿入されます。
- **nodeAttribute[+attribute]** では、タグノードの属性を設定します。属性が存在しない場合は、この値によって属性が作成されます。

  例えば、<insert location="nodeAttribute+ACTION" nodeParamName="form"> を使用して、フォームの ACTION 属性を設定します。これにより、ユーザの FORM タグが <form> から <formaction="myText"> に変わります。

  属性を指定しないと、**nodeAttribute** の場所によってテキストが直接開始タグに追加されます。例えば、タグにオプションの属性を追加するには、insert location="nodeAttribute" を使用します。これは、ユーザの INPUT タグを <input type="checkbox"> から <input type="checkbox" %if(foo)Reponse.Write("CHECKED")%> に変更する際に使用できます。

  注意：location="nodeAttribute" 属性値を指定すると、最後の検索パターンによって属性の開始位置と終了位置が決まります。最後のパターンでステートメント全体が検出されるようにしてください。

nodeParamName

説明
この属性は、ノードを基準とする挿入位置にのみ使用します。挿入時にノードを渡すために使用するパラメータの名前を示します。

親
insertText

タイプ
属性

必須
この属性は、挿入場所に node という単語が含まれている場合にのみ必要です。

値

tagtype__Tag は、パラメータオブジェクトを使用して dwscripts.applySB() 関数に渡されるノードパラメータにユーザが指定した名前です。例えば、フォームにテキストを挿入する場合は、**form_tag** パラメータを使用できます。次の例で示すように、サーバビヘイビアの applyServerBehavior() 関数で、**form_tag** パラメータを使用して、更新するフォームを指定することができます。

```
function applyServerBehavior(ssRec) {
    var paramObj = new Object();
    paramObj.rs = getRecordsetName();
    paramObj.form_tag = getFormNode();
    dwscripts.applySB(paramObj, sbObj);
}
```

次の例で示すように、EDML ファイルに form_tag ノードパラメータを指定できます。

```
<insertText location="lastChildOfNode" nodeParamName="form_tag">
    <![CDATA[<input type="hidden" name="MY_DATA">]]>
</insertText>
```

テキストが lastChildOfNode の値として挿入され、パラメータオブジェクトの form_tag プロパティを使用して特定のノードが渡されます。

<searchPatterns>

説明

このタグは、ドキュメント内の構成要素テキストの検索に関する情報を提供します。また、構成要素の検索に使用するパターンのリストが含まれています。複数の検索パターンが定義されている場合は、そのすべてが検索対象のテキストで検出される必要があります（検索パターンには論理 AND 関係が適用されるため）。ただし、isOptional フラグでパターンがオプションとしてマークされている場合を除きます。

親

implementation

タイプ

ブロックタグ

必須

いいえ

<searchPatterns> の属性

次の項目は、searchPatterns タグの有効な属性です。

whereToSearch

説明

この属性は、構成要素テキストを検索する場所を指定します。この属性は挿入位置に関係するので、各属性を注意して設定してください（詳しくは、261 ページの「location」を参照してください）。

親

searchPatterns

タイプ
属性

必須
はい

値
directive、***tag+tagName***、**tag+***、**comment**、**text**

- **directive** を指定すると、すべてのサーバディレクティブ（サーバ固有のタグ）が検索されます。ASP および JSP の場合は、すべての <% ... %> スクリプトブロックが検索されます。

  注意：タグ属性は、ディレクティブを含んでいても検索されません。

- 次の例で示すように、**tag+tagName** を指定すると、指定されたタグのコンテンツが検索されます。

  ```
  <searchPatterns whereToSearch="tag+FORM">
  ```

  この例では、form タグのみが検索されます。デフォルトでは、outerHTML ノード全体が検索されます。INPUT タグの場合は、スラッシュ（/）の後にタイプを指定します。次の例では、すべての送信ボタンが検索されます。

  ```
  <searchPatterns whereToSearch="tag+INPUT/SUBMIT">.
  ```

- 次の例で示すように、**tag+*** を指定すると、任意のタグのコンテンツが検索されます。

  ```
  <searchPatterns whereToSearch="tag+*">
  ```

  この例では、すべてのタグが検索されます。

- 次の例で示すように、**comment** を指定すると、HTML コメントである <! ... > の内部のみが検索されます。

  ```
  <searchPatterns whereToSearch="comment">
  ```

  この例では、<!-- my comment here --> などのタグが検索されます。

- 次の例で示すように、**text** を指定すると、未処理のテキストセクション内のみが検索されます。

  ```
  <searchPatterns whereToSearch="text">
      <searchPattern>XYZ</searchPattern>
  </searchPatterns>
  ```

  この例では、XYZ というテキストを含むテキストノードが検索されます。

<searchPattern>

説明
このタグは、構成要素テキストを識別し、そこからパラメータ値を抽出するためのパターンです。それぞれのパラメータサブ表現は、括弧 () で囲む必要があります。

パラメータを指定しないパターン（構成要素テキストの識別に使用）、パラメータを 1 つだけ指定したパターンまたは多数のパラメータを指定したパターンを使用できます。オプションと指定されていないパターンはすべて検出される必要があります。また、各パラメータを指定および検索できるのは一度だけです。

searchPattern タグの使用について詳しくは、276 ページの「サーバビヘイビアの検索」を参照してください。

親
searchPatterns

タイプ
ブロックタグ

必須
はい

値
searchString、**/regularExpression/**、**<empty>**

- **searchString** の値は、大文字と小文字を区別する単純な検索ストリングです。パラメータの抽出には使用できません。
- **/regularExpression/** は、正規表現を使用した検索パターンです。
- **<empty>** は、パターンが指定されていない場合に使用します。これは常に一致すると見なされ、値全体が最初のパラメータに割り当てられます。

次の例では、<%= RS1.Field.Items("author_id") %> という構成要素テキストを識別するために、単純なパターンを定義し、その後に 2 つのパラメータ値の抽出も行う正確なパターンを指定します。

```
<searchPattern>Field.Items</searchPattern>
<searchPattern paramNames="rs,col">
    <![CDATA[
    /<%=\s*(\w+)\.Field\.Items\("(\w+)"\)/
    ]]>
</searchPattern>
```

この例では、正確なパターン検索が行われ、最初のサブ表現（\w+）の値が rs パラメータに、2 番目のサブ表現（\w+）の値が col パラメータに割り当てられます。

注意：正規表現は、必ずスラッシュ（/）で開始し、スラッシュで終了します。このように指定しないと、リテラルストリングとして検索されます。正規表現の後に正規表現変換修飾子 i を付ける（/pattern/i のように）と、大文字と小文字が区別されなくなります。例えば、VBScript では大文字と小文字が区別されないため、/pattern/i を使用する必要があります。JavaScript では大文字と小文字が区別されるため、/pattern/ を使用する必要があります。

場合によっては、限られた検索範囲の内容全体をパラメータに割り当てることが必要となります。この場合は、次の例で示すように、パターンを指定しません。

```
<searchPatterns whereToSearch="tag+OPTION">
    <searchPattern>MY_OPTION_NAME</searchPattern>
    <searchPattern paramNames="optionLabel" limitSearch="innerOnly">
    </searchPattern>
</searchPatterns>
```

この例では、optionLabel パラメータを innerHTML の内容全体（OPTION タグ内）に設定しています。

<searchPattern> の属性

次の項目は、searchPattern タグの有効な属性です。

paramNames

説明
この属性は、値が抽出されるパラメータ名をカンマで区切ったリストです。これらのパラメータは、サブ表現の順番に割り当てられます。単一のパラメータを割り当てるか、またはカンマで区切ったリストを使用して複数のパラメータを割り当てることができます。パラメータを指定しない挿入句表現が使用されている場合は、パラメータ名リストにプレースホルダーとしてカンマを追加することができます。

パラメータ名は、挿入テキストおよび更新パラメータに指定された名前と一致している必要があります。

親
searchPattern

タイプ
属性

必須
はい

値
paramName1、**paramName2**、...

各パラメータ名は、挿入テキストで使用されているパラメータの名前と完全に一致している必要があります。例えば、挿入テキストに @@p1@@ が含まれている場合は、この名前のパラメータを 1 つだけ定義する必要があります。

```
<searchPattern paramNames="p1">patterns</searchPattern>
```

1 つのパターンで複数のパラメータを抽出するには、パターンに出現するサブ表現の順にカンマで区切って並べられたパラメータ名のリストを使用します。次の例が検索パターンを示すとします。

```
<searchPattern paramName="p1,,p2">/(\w+)_(BIG|SMALL)_(\w+)/</searchPattern>
```

2 つのパラメータが、その間にあるテキストと共に抽出されます。<%= a_BIG_b %> というテキストでは、検索パターンの最初のサブ表現が a と一致するため、p1="a" です。2 番目のサブ表現は無視されます（,, が paramName 値の中にあるため）。3 番目のサブ表現は、b と一致するので、p2="b" です。

limitSearch

説明
この属性は、検索範囲を whereToSearch タグの一部分に限定します。

親
searchPattern

タイプ
属性

必須
いいえ

値
all、**attribute+attribName**、**tagOnly**、**innerOnly**

- **all**（デフォルト値）を指定すると、whereToSearch 属性に指定されたタグ全体が検索されます。
- 次の例で示すように、**attribute+attribName** を指定すると、指定された属性の値内のみが検索されます。

```
<searchPatterns whereToSearch="tag+FORM">
    <searchPattern limitSearch="attribute+ACTION">
        /MY_PATTERN/
    </searchPattern>
</searchPatterns>
```

この例では、ACTION 属性（FORM タグ）の値のみが検索されます。この属性が定義されていない場合は、タグが無視されます。

- **tagOnly** を指定すると、外側のタグのみが検索され、innerHTML タグは無視されます。この値は、whereToSearch がタグである場合にのみ有効です。
- **innerOnly** を指定すると、innerHTML タグのみが検索され、外側のタグは無視されます。この値は、whereToSearch がタグである場合にのみ有効です。

isOptional

説明
この属性は、構成要素の検出に検索パターンが不要であることを示すフラグです。これは、構成要素が複雑で、抽出するパラメータが重要ではない場合に便利です。構成要素を明確に識別するいくつかのパターンを作成し、重要でないパラメータを抽出するオプションのパターンを指定することができます。

親
searchPattern

タイプ
属性

必須
いいえ

値
true、**false**

- 値 **true** は、構成要素を識別するための searchPattern が不要な場合に指定します。
- 値 **false**（デフォルト値）は、searchPattern タグが必要な場合に指定します。

例えば、次のような単純なレコードセットストリングがあるとします。

```
<%
var Recordset1 = Server.CreateObject("ADODB.Recordset");
Recordset1.ActiveConnection = "dsn=andescoffee;";
Recordset1.Source = "SELECT * FROM PressReleases";
Recordset1.CursorType = 3;
Recordset1.Open();
%>
```

検索パターンによって、構成要素を識別し、いくつかのパラメータを抽出する必要があります。ただし、**cursorType** などのパラメータが見つからない場合でも、このパターンをレコードセットとして認識する必要があります。**cursor** パラメータはオプションです。EDML では、この検索パターンが次の例のように記述されます。

```
<searchPattern paramNames="rs">/var (\w+) = Server.CreateObject/
</searchPattern>
<searchPattern paramNames="src">/ActiveConnection = "([^\r\n]*)"/</searchPattern>
<searchPattern paramNames="conn">/Source = "([^\r\n]*)"/</searchPattern>
<searchPattern paramNames="cursor" isOptional="true">/CursorType = (\d+)/
</searchPattern>
```

最初の 3 つのパターンは、レコードセットを識別するために必要です。最後のパラメータが検出されなくても、レコードセットは識別されます。

<updatePatterns>

説明
この高度なオプションの機能によって、構成要素を正確に更新できます。このタグを使用しない場合は、構成要素テキスト全体を毎回置き換えることで、構成要素が自動的に更新されます。updatePatterns タグを指定する場合は、構成要素内の各パラメータを検出して置換する特定のパターンをこのタグに記述する必要があります。

ユーザが構成要素テキストを編集する場合は、このタグが役立ちます。このタグによって、テキストの変更が必要な部分のみが的確に更新されます。

親
implementation

タイプ
ブロックタグ

必須
いいえ

<updatePattern>

説明
このタグは、構成要素テキストの正確な更新を可能にするための特殊なタイプの正規表現です。挿入テキスト（@@paramName@@ の形式）で宣言されているすべての固有なパラメータに対して、最低 1 つの更新パターン定義が必要です。

親
updatePatterns

タイプ
ブロックタグ

必須
はい（updatePatterns タグを宣言している場合は少なくとも 1 つ必要）

値
値は、**/(pre-pattern)parameter-pattern(post-pattern)/** の形式で、括弧で囲まれた 2 つのサブ表現の間にあるパラメータを検出する正規表現です。挿入テキスト内の一意の @@paramName@@ ごとに、少なくとも 1 つの更新パターンを定義する必要があります。以下は、挿入テキストの例です。

```
<insertText location="afterSelection">
    <![CDATA[<%= @@rs@@.Field.Items("@@col@@") %>]]>
</insertText>
```

ページ上にあるこの挿入テキストの特定のインスタンスは、次の例のように記述できます。

```
<%= RS1.Field.Items("author_id") %>
```

rs および col の 2 つのパラメータがあります。このテキストを、ページに挿入した後で更新するには、2 つの更新パターンを定義する必要があります。

```
<updatePattern paramName="rs" >
    /(\b)\w+(\.Field\.Items)/
</updatePattern>
<updatePattern paramName="col">
    /(\bItems\(")\w+("\))/
</updatePattern>
```

リテラル文字の括弧およびその他の特殊な正規表現文字は、前にバックスラッシュ（\）を付けることでエスケープされています。\w+ と定義されている中央の表現は、rs パラメータおよび col パラメータに渡された最新の値で更新されます。値 RS1 および author_id は、新しい値で更新されます。

何度も出現する同じパターンを同時に更新することができます。そのためには、終了スラッシュの後に正規表現のグローバルフラグ g を指定します（**/pattern/g** のように）。

構成要素テキストが長く複雑である場合は、次の例で示すように、1 つのパラメータを更新するために複数のパターンが必要となる場合があります。

```
<% ...
    Recordset1.CursorType = 0;
    Recordset1.CursorLocation = 2;
    Recordset1.LockType = 3;
%>
```

3 つの位置すべてでレコードセット名を更新するには、次の例で示すように、1 つのパラメータに対して 3 つの更新パターンが必要です。

```
<updatePattern paramName="rs">
    /(\b)\w+(\.CursorType)/
</updatePattern>
<updatePattern paramName="rs">
    /(\b)\w+(\.CursorLocation)/
</updatePattern>
<updatePattern paramName="rs">
    /(\b)\w+(\.LockType)/
</updatePattern>
```

これでレコードセットに新しい値を渡すことが可能となり、3 つの位置でレコードセットが的確に更新されます。

<updatePattern> の属性

次の項目は、updatePattern タグの有効な属性です。

paramName

説明

この属性は、構成要素の更新に使用する値を持つパラメータの名前を示します。このパラメータは、挿入テキストおよび検索パラメータに指定されたものと一致している必要があります。

親

updatePattern

タイプ

属性

必須

はい

値
値は、挿入テキストに使用されているパラメータの正確な名前です。次の例では、挿入テキストに値 @@rs@@ が含まれている場合、同じ名前のパラメータが必要になります。

```
<updatePattern paramName="rs">pattern</updatePattern>
```

<delete>

説明
このタグは、構成要素の削除方法を制御するための高度なオプション機能です。このタグを使用しないと、構成要素は完全に取り除く方法で削除されます。ただし、その構成要素を参照するサーバビヘイビアが存在しない場合に限ります。delete タグを指定すると、構成要素を削除しないように、または部分的に削除するように指定できます。

親
implementation

タイプ
タグ

必須
いいえ

<delete> の属性
次の項目は、delete タグの有効な属性です。

deleteType

説明
この属性は、実行する削除のタイプを指定するために使用します。構成要素がディレクティブ、タグ、属性のどれであるかによって、この属性の意味が異なります。デフォルトでは、構成要素全体が削除されます。

親
delete

タイプ
属性

必須
いいえ

値
all、**none**、**tagOnly**、**innerOnly**、**attribute+attribName**、**attribute+***

- **all**（デフォルト値）を指定すると、ディレクティブまたはタグ全体が削除されます。属性の場合は、定義全体が削除されます。
- **none** を指定すると、この構成要素は自動的に削除されなくなります。

- **tagOnly** を指定すると、外側のタグのみが削除され、innerHTML タグのコンテンツはそのまま残されます。属性の場合は、外側のタグがブロックタグであると、そのタグも削除されます。ディレクティブの場合は何も削除されません。
- **innerOnly** をタグに適用すると、コンテンツ（innerHTML タグ）のみが削除されます。属性の場合は、値のみが削除されます。ディレクティブの場合は何も削除されません。
- **attribute+attribName** をタグに適用すると、指定した属性のみが削除されます。ディレクティブと属性の場合は、何も削除されません。
- **attribute+*** をタグに指定すると、すべての属性が削除されます。ディレクティブと属性の場合は、何も削除されません。

サーバビヘイビアによって選択したテキストがリンクに変換される場合は、次の例に示すように、外側のタグを削除するだけでリンクを削除できます。

```
<delete deleteType="tagOnly"/>
```

この例では、リンク構成要素が <A HREF="...">HELLO</A> から HELLO に変更されています。

<translator>

説明
このタグは、構成要素をトランスレートして、そのレンダリング方法を変更し、カスタムプロパティインスペクタが表示されるようにするための情報を提供します。

親
implementation

タイプ
ブロックタグ

必須
いいえ

<searchPatterns>

説明
このタグは、ドキュメント内の指定された各インスタンスを Dreamweaver で検索できるようにします。複数の検索パターンが定義されている場合は、そのすべてが検索対象のテキストに含まれている必要があります（検索パターンには論理 AND 関係が適用されるため）。ただし、isOptional フラグでパターンがオプションとマークされている場合を除きます。

親
translator

タイプ
ブロックタグ

必須
はい

<translations>

説明
このタグには、構成要素を検索する場所およびその構成要素の処理を示す、トランスレートの指示のリストが含まれています。

親
translator

タイプ
ブロックタグ

必須
いいえ

<translation>

説明
このタグには、トランスレートの指示が 1 つあります。この指示には、構成要素の位置、実行するトランスレートのタイプ、および構成要素テキストを置換するコンテンツが含まれています。

親
translations

タイプ
ブロックタグ

必須
いいえ

<translation> の属性

次の項目は、translation タグの有効な属性です。

whereToSearch

説明
この属性は、テキストを検索する場所を指定します。これは挿入位置に関係するため、それぞれの場所を注意して設定してください（詳しくは、261 ページの「location」を参照してください）。

親
translation

タイプ
属性

必須
はい

limitSearch

説明
この属性は、検索範囲を whereToSearch タグの一部分に限定します。

親
translation

タイプ
属性

必須
いいえ

translationType

説明
この属性は、実行するトランスレートのタイプを指定します。これらのタイプは事前に設定され、トランスレート特有の機能を提供します。例えば、dynamic data を指定すると、トランスレートされるデータが Dreamweaver の動的データと同じように動作するようになります。つまり、デザインビューでは動的データプレースホルダーのように波括弧（{}）で囲まれて動的な背景色で表示され、サーバビヘイビアパネルにも表示されます。

親
translation

タイプ
属性

必須
はい

値
dynamic data、**dynamic image**、**dynamic source**、**tabbed region start**、**tabbed region end**、**custom**

- 次の例で示すように、**dynamic data** を指定すると、トランスレートされたディレクティブが Dreamweaver 動的データと同じように表示され、動作します。

```
<translation whereToSearch="tag+IMAGE"
    limitSearch="attribute+SRC"
    translationType="dynamic data">
```

- 次の例で示すように、**dynamic image** を指定すると、トランスレートされた属性が Dreamweaver 動的データと同じように表示され、動作します。

```
<translation whereToSearch="IMAGE+SRC"
    translationType="dynamic image">
```

- 次の例で示すように、**dynamic source** を指定すると、トランスレートされたディレクティブが Dreamweaver 動的ソースと同じように表示され、動作します。

```
<translation whereToSearch="directive"
    translationType="dynamic source">
```

- 次の例で示すように、***tabbed region start*** を指定すると、トランスレートされた <CFLOOP> タグによって、タブグループ化されたアウトラインの開始点が定義されます。

```
<translation whereToSearch="CFLOOP"
    translationType="tabbed region start">
```

- 次の例で示すように、***tabbed region end*** を指定すると、トランスレートされた </CFLOOP> タグによって、タブグループ化されたアウトラインの終了点が定義されます。

```
<translation whereToSearch="CFLOOP"
    translationType="tabbed region end">
```

- **custom** はデフォルト値で、Dreamweaver の内部機能はトランスレートに追加されません。次の例で示すように、この値はカスタムプロパティインスペクタに挿入するタグを指定する際によく使用されます。

```
<translation whereToSearch="directive"
    translationType="custom">
```

<openTag>

説明
これは、translation セクションの始めに挿入されるオプションのタグです。このタグを使用すると、他の一部の拡張機能（カスタムプロパティインスペクタなど）でトランスレートを検出できるようになります。

親
translation

タイプ
ブロックタグ

必須
いいえ

値
tagName は有効なタグ名です。既知のタグタイプと競合しないように、一意の名前にする必要があります。例えば、<openTag>MM_DYNAMIC_CONTENT</openTag> を指定すると、動的データが MM_DYNAMIC_CONTENT タグにトランスレートされます。

<attributes>

説明
このタグは、openTag タグで指定されたトランスレート済みのタグに追加される属性のリストを含んでいます。一方、openTag タグが定義されておらず、searchPattern タグに tag が指定されている場合、このタグには検出されたタグに追加されるトランスレートされた属性のリストが含まれます。

親
translation

タイプ
ブロックタグ

必須
いいえ

<attribute>

説明
このタグは、トランスレートされたタグに追加される属性（またはトランスレートされた属性）を 1 つ指定します。

親
attributes

タイプ
ブロックタグ

必須
はい（少なくとも 1 つが必要）

値
attributeName="**attributeValue**" という指定により、属性に値を設定します。通常、属性名は固定されており、次の例で示すように、値にはパラメータパターンによって抽出されるパラメータ参照が含まれています。

```
<attribute>SOURCE="@@rs@@"</attribute>
<attribute>BINDING="@@col@@"</attribute>
```

または

```
<attribute>
    mmTranslatedValueDynValue="VALUE={@@rs@@.@@col@@}"
</attribute>
```

<display>

説明
このタグは、トランスレートに挿入されるオプションの表示ストリングです。

親
translation

タイプ
ブロックタグ

必須
いいえ

値
displayString は、テキストと HTML で構成される任意のストリングです。パラメータパターンによって抽出されたパラメータ参照を含めることもできます。例えば、<display>{@@rs@@.@@col@@}</display> と指定すると、トランスレートが {myRecordset.myCol} とレンダリングされます。

<closeTag>

説明
このオプションのタグは、トランスレートされたセクションの最後に挿入します。このタグを使用すると、他の一部の拡張機能（カスタムプロパティインスペクタなど）でトランスレートを検出できるようになります。

親
translation

タイプ
ブロックタグ

必須
いいえ

値
tagName は有効なタグ名です。この名前は、トランスレートの **openTag** タグと一致している必要があります。

例
<closeTag>MM_DYNAMIC_CONTENT</closeTag> を指定すると、動的データは </MM_DYNAMIC_CONTENT> タグで終了するようにトランスレートされます。

サーバビヘイビアの手法

この節では、サーバビヘイビアを作成および編集するための一般的な手法と高度な手法について説明します。ここで示す方法は、主に EDML ファイルでの特定の設定に関するものです。

サーバビヘイビアの検索

サーバビヘイビアは、次の方法で検索できます。

- 検索パターンの記述
- オプションの検索パターンの使用

検索パターンの記述
サーバビヘイビアを更新または削除するには、個々のインスタンスをドキュメント内で検索する方法を Dreamweaver に指定します。これには、quickSearch タグと、searchPatterns タグに含まれる searchPattern タグが少なくとも 1 つ必要です。

quickSearch タグは、正規表現ではなく、サーバビヘイビアがページに存在することを示すストリングとして表します。大文字と小文字は区別されません。簡潔かつ一意にする必要があり、スペースやユーザが変更できるセクションは使用しないようにします。次の例では、単純な ASP JavaScript タグで構成される構成要素を示します。

```
<% if (Recordset1.EOF) Response.Redirect("some_url_here") %>
```

次の例では、quickSearch ストリングでこのタグをチェックします。

```
<quickSearch>Response.Redirect</quickSearch>
```

パフォーマンス上の理由から、サーバビヘイビアのインスタンスの検索では、quickSearch パターンが最初に処理されます。このストリングがドキュメント内で検出され、構成要素によってサーバビヘイビアが識別されると（グループファイルでこの構成要素に partType="identifier" を指定）関連するサーバビヘイビアのファイルが読み込まれ、findServerBehaviors() 関数が呼び出されます。構成要素に検索またはデバッグに使用できる信頼性のあるストリングがない場合は、次の例で示すように、quickSearch ストリングを空白にしておくこともできます。

```
<quickSearch></quickSearch>
```

この例では、サーバビヘイビアが必ず読み込まれ、ドキュメントを検索することができます。

次に、searchPattern タグでドキュメントが quickSearch タグよりも厳密に検索され、構成要素コードからパラメータ値が抽出されます。この検索パターンは、検索する場所（whereToSearch 属性）を指定するために、特定のパターンを含む一連の searchPattern タグを使用します。これらのパターンでは、簡単なストリングも正規表現も使用できます。前述のサンプルコードは ASP ディレクティブであるため、次の例で示すように、whereToSearch="directive" の指定と正規表現によって、ディレクティブが識別され、パラメータが抽出されます。

```
<quickSearch>Response.Write</quickSearch>
<searchPatterns whereToSearch="directive">
    <searchPattern paramNames="rs,new__url">
        /if\s*\((\w+)\.EOF\)\s*Response\.Redirect\("([^\r\n]*)"\)/i
    </searchPattern>
</searchPatterns>
```

検索ストリングは、先頭と末尾にスラッシュ（/）を付加することで、正規表現として定義されます。スラッシュに続く i は、大文字と小文字を区別しないことを示します。正規表現の中では、カッコ () やピリオド（.）などの特殊文字が、前にバックスラッシュ（\）を付けることでエスケープされます。rs と new_url の 2 つのパラメータは、カッコで囲まれたサブ表現によってストリングから抽出されます。パラメータはカッコで囲む必要があります。この例の (\w+) と ([^\r\n]*) はパラメータを示しています。これらの値は、$1 および $2 が通常返す正規表現値に対応します。

オプションの検索パターン

いくつかのパラメータが見つからない場合でも、構成要素を識別することができます。構成要素には、電話番号など、オプションの情報が格納されています。このような場合には、次の ASP コードを使用することができます。

```
<% //address block
    LNAME = "joe";
    FNAME = "smith";
    PHONE = "123-4567";
%>
```

次のような検索パターンを使用できます。

```
<quickSearch>address</quickSearch>
<searchPatterns whereToSearch="directive">
    <searchPattern paramNames="lname">/LNAME\s*=\s*"([^\r\n]*)"/i</searchPattern>
    <searchPattern paramNames="fname">/FNAME\s*=\s*"([^\r\n]*)"/i</searchPattern>
    <searchPattern paramNames="phone">/PHONE\s*=\s*"([^\r\n]*)"/i</searchPattern>
</searchPatterns>
```

前述の例では、電話番号を指定する必要があります。ただし、次の例で示すように、isOptional 属性を追加することによって、電話番号をオプションにすることができます。

```
<quickSearch>address</quickSearch>
<searchPatterns whereToSearch="directive">
    <searchPattern paramNames="lname">/LNAME\s*=\s*"([^\r\n]*)"/i</searchPattern>
    <searchPattern paramNames="fname">/FNAME\s*=\s*"([^\r\n]*)"/i</searchPattern>
    <searchPattern paramNames="phone" isOptional="true">/PHONE\s*=\s*"([^\r\n]*)"/i¬
    </searchPattern>
</searchPatterns>
```

これで、電話番号が検出されなくても、構成要素が認識されます。

構成要素の照合方法

サーバビヘイビアに複数の構成要素がある場合は、それらをユーザドキュメント内で識別し、照合する必要があります。ユーザがサーバビヘイビアの複数のインスタンスをドキュメントに適用する場合は、それに応じて構成要素の各グループを照合する必要があります。構成要素が正しく照合されるためには、パラメータを変更または追加したり、構成要素を一意に識別されるように構築したりします。

照合には、いくつかのルールが必要です。構成要素は、同じ名前のパラメータの値がすべて同じである場合に一致します。html タグの前後には、指定したパラメータ値のセットがある構成要素のインスタンスが 1 つしかない場合もあります。html.../html タグの中では、選択範囲や挿入に使用する共通ノードを基準とした位置によって構成要素が照合されます。

パラメータがない構成要素は、次のグループファイルがあるサーバビヘイビアの例のように、自動的に照合されます。

```
<group serverBehavior="test.htm">
    <title>Test</title>
    <groupParticipants>
        <groupParticipant name="test_p1" partType="identifier" />
        <groupParticipant name="test_p2" partType="identifier" />
    </groupParticipants>
</group>
```

以下の例では、html タグの前に、2 つの単純な構成要素を挿入します。

```
<% //test_p1 %>
<% //test_p2 %>
<html>
```

これらの構成要素は検出および照合され、**Test** というタイトルが一度だけサーバビヘイビアパネルに表示されます。サーバビヘイビアを再び追加しても、構成要素が存在しているので、何も追加されません。

これらの構成要素に固有のパラメータが指定されている場合は、html タグの前に複数のインスタンスを挿入できます。例えば、構成要素に name パラメータを追加すると、ユーザは Test というサーバビヘイビアダイアログに固有名を入力できるようになりますが、ユーザが **aaa** という名前を入力すると、次の構成要素が挿入されます。

bbb などの別の名前でサーバビヘイビアを再び追加すると、ドキュメントは次の例のようになります。

```
<% //test_p1 name="aaa" %>
<% //test_p2 name="aaa" %>
<html>
```

サーバビヘイビアパネルには、2 つの Test のインスタンスが表示されます。ユーザが **aaa** という名前の 3 つ目のインスタンスをページに追加しようとすると、この名前のインスタンスは存在するため、何も追加されません。

html タグ内の照合には、位置情報も使用されます。次の例では、2 つの構成要素があり、1 つを選択範囲の前に、もう 1 つを選択範囲の後にそれぞれ追加します。

```
<% if (expression) { //mySBName %>
```

ここはランダムな HTML 選択範囲です。

```
<% } //end mySBName %>
```

この 2 つの構成要素にはパラメータが指定されていないので、1 つのグループとしてまとめられます。ただし、次の例で示すように、このサーバビヘイビアの別のインスタンスを HTML 内の任意の位置に追加できます。

```
<% if (expression) { //mySBName %>
```

ここはランダムな HTML 選択範囲です。

```
<% } //end mySBName %>
```

ここでさらに HTML が追加されます。

```
<% if (expression) { //mySBName %>
```

もう 1 つ HTML の選択範囲があります。

```
<% } //end mySBName %>
```

これで、それぞれの構成要素にまったく同じインスタンスが 2 個ずつ存在することになり（これは同じ HTML 内に限り可能です）、ドキュメントで検出される順に照合されます。

次の例では、照合上の問題とその回避方法を示します。ある動的データの税金を計算し、その結果を選択範囲に表示する構成要素を作成できます。

```
<% total = Recordset1.Fields.Item("itemPrice").Value * 1.0825 %>
<html>
<body>
    The total (with taxes) is $<%=total%>
</body>
</html>
```

2 つの構成要素は、共通のパラメータがないので照合されます。しかし、このサーバビヘイビアに 2 つ目のインスタンスを追加すると、コードは次のようになります。

```
<% total = Recordset1.Fields.Item("itemPrice").Value * 1.0825 %>
<% total = Recordset1.Fields.Item("salePrice").Value * 1.0825 %>
<html>
<body>
    The total0(with taxes) is $<%=total%>
    Sale price (with taxes) is $<%=total%>
</body>
</html>
```

このサーバビヘイビアには total というパラメータが 1 つしかないので、正しく動作しません。この問題を解決するには、パラメータに一意の値を指定し、構成要素の照合に使用できるようにします。次の例では、列の名前を使用して total 変数名が一意になるようにします。

```
<% itemPrice_total = Recordset1.Fields.Item("itemPrice").Value * 1.0825 %>
<% salePrice_total = Recordset1.Fields.Item("salePrice").Value * 1.0825 %>
<html>
<body>
    The total0(with taxes) is $<%=itemPrice_total%>
    Sale price (with taxes) is $<%=salePrice_total%>
</body>
</html>
```

これで、検索パターンで構成要素が一意に識別され、照合されます。

検索パターンの解決

Dreamweaver では、構成要素の searchPatterns 機能を使用して、次のアクションを実行できます。

- ファイル転送の依存関係
- 任意のファイルリファレンスのファイルパス（インクルードファイルのファイルパスなど）の更新

Dreamweaver でサーバモデルが作成される際に、すべての構成要素が特別な paramNames 属性についてスキャンされて、パターンのリストが作成されます。ファイルの依存性をチェックし、パス名を決める URL を検索するために、各 searchPattern タグ（内部の paramNames 属性の 1 つが _url で終わっている）が使用されます。複数の URL を 1 つの searchPattern タグで指定できます。

トランスレータの searchPattern タグで、paramNames 属性値が _includeUrl で終わるものそれぞれについて、その searchPattern タグを使用してページ上のインクルードファイルステートメントがトランスレートされます。すべての URL リファレンスがトランスレートされるわけではないので、別のサフィックスストリングを使用してインクルードファイルの URL が識別されます。また、インクルードファイルとしてトランスレートできるのは、1 つの URL のみです。

searchPatterns タグを解決する際には、以下のアルゴリズムが使用されます。

1 whereToSearch 属性が searchPatterns タグ内で検索されます。

2 属性値が tag+ で始まる場合、残りのストリングはタグ名であると見なされます（タグ名にはスペースを使用できません）。

3 limitSearch 属性が searchPattern タグ内で検索されます。

4 属性値が attribute+ で始まる場合、残りのストリングは属性名であると見なされます（属性名にはスペースを使用できません）。

この 4 つの手順が正常に実行された場合は、タグと属性の組み合わせが存在すると見なされます。そうでない場合には、searchPattern タグ（paramName 属性に _url サフィックスと定義済みの正規表現が含まれる）が検索されます （正規表現については、252 ページの「正規表現」を参照してください）。

次の searchPatterns タグの例には、検索パターンが含まれません。これは、タグ（cfinclude）と属性（template）を組み合わせて、依存ファイルのチェックやパスの固定などを行うために URL が分離されるためです。

```
<searchPatterns whereToSearch="tag+cfinclude">
    <searchPattern paramNames="include_url" limitSearch="attribute+template" />
</searchPatterns>
```

タグと属性の組み合わせ（前の例を参照）は、トランスレートには適用されません。これは、純粋なテキストへのトランスレートが常に JavaScript レイヤーで行われるからです。ファイルの依存性のチェック、パスの固定などは C レイヤーで行われます。C レイヤーでは、ドキュメントが内部的にディレクティブ（純粋なテキスト）とタグ（有効なツリー構造に解析済み）に分割されます。

サーバビヘイビアの更新

サーバビヘイビアは、次の方法で更新できます。

- 置換による更新
- 厳密な更新

置換による更新

デフォルトでは、EDML 構成要素ファイルに <updatePatterns> タグが含まれないため、構成要素のインスタンスは完全な置換によってドキュメント内で更新されます。ユーザが既存のサーバビヘイビアを編集して「OK」をクリックすると、値の変更されたパラメータを含む構成要素は削除されます。次に、新しい値を含む構成要素が同じ場所に再挿入されます。

ユーザがドキュメント内の構成要素コードをカスタマイズすると、検索パターンで古いコードを検索した場合にその構成要素が認識されないことがあります。短い検索パターンを使用すると、ユーザがドキュメントで構成要素コードをカスタマイズできるようになります。ただし、サーバビヘイビアのインスタンスを更新すると、構成要素が置換されて、カスタム編集が失われる可能性があります。

厳密な更新

場合によっては、構成要素コードをドキュメントに挿入した後にカスタマイズできるようにする方が望ましいことがあります。これを実現するには、検索パターンを制限し、EDML ファイルで更新パターンを指定します。構成要素をページに追加すると、サーバビヘイビアはその特定の部分のみを更新するようになります。次の例は、2 つのパラメータがある簡単な構成要素を示しています。

```
<% if (Recordset1.EOF) Response.Redirect("some_url_here") %>
```

この例では、次のような検索パターンを使用できます。

```
<quickSearch>Response.Write</quickSearch>
<searchPatterns whereToSearch="directive">
    <searchPattern paramNames="rs,new__url">
        /if\s*\((\w+)\.EOF\)\s*Response\.Redirect\("([^\r\n]*)"\)/i
    </searchPattern>
</searchPatterns>
```

次の例で示すように、このコード内の特定のインスタンスに別のテストを追加したとします。

```
<% if (Recordset1.EOF || x > 2) Response.Redirect("some_url_here") %>
```

EOF パラメータの後のカッコが検索されるため、検索パターンは失敗します。検索パターンに一致する対象を広げるには、次の例に示すように、検索パターンを分割して短くします。

```
<quickSearch>Response.Write</quickSearch>
<searchPatterns whereToSearch="directive">
    <searchPattern paramNames="rs">/(\w+)\.EOF/</searchPattern>
    <searchPattern paramNames="new__url">
        /if\s*\([^\r\n]*\)\s*Response\.Redirect\("([^\r\n]*)"/i
    </searchPattern>
</searchPatterns>
```

検索パターンを短縮すると、適応できる範囲が広がり、ユーザはコードに追加できるようになります。ただし、サーバビヘイビアによって URL が変更される場合は、ユーザが「OK」をクリックすると、構成要素が置換され、カスタマイズした内容が失われます。より厳密に更新するには、各パラメータを更新するパターンを含む updatePatterns タグを追加します。

```
<updatePatterns>
    <updatePattern paramNames="rs">/(\b)\w+(\.EOF)/\</updatePattern>
    <updatePattern paramNames="new__url">
        /(Response\.Redirect\(")[^\r\n]*(")/i
    </updatePattern>
</updatePatterns>
```

更新パターンでは、カッコの配置を逆にして、パラメータの前後にあるテキストを囲みます。検索パターンでは textBeforeParam(param)textAfterParam パラメータを使用しますが、更新パターンでは (textBeforeParam)param(textAfterParam) パラメータを使用します。2 つのカッコで囲まれたサブ表現の間にあるテキストは、すべて新しいパラメータ値に置き換えられます。

サーバビヘイビアの削除

サーバビヘイビアは、次の方法で削除できます。

- デフォルトの削除方法と従属カウント
- delete フラグを使用した構成要素の削除の制限

デフォルトの削除方法と従属カウント

「-」（マイナス）ボタンをクリックするか、Delete キーを押すと、サーバビヘイビアパネルで選択したインスタンスを削除できます。他のサーバビヘイビアと共有されているものを除き、すべての構成要素が削除されます。具体的には、複数のサーバビヘイビアに同じノードを指す構成要素ポインタが定義されている場合、そのノードは削除されません。

デフォルトでは、タグ全体を除去すると構成要素は削除されます。挿入位置が wrapSelection の場合は、外側のタグだけが削除されます。属性の場合は、属性宣言全体が削除されます。次の例では ACTION 属性（form タグ）にある属性構成要素を示します。

```
<form action="<% my_participant %>">
```

属性を削除すると、form タグのみが残ります。

delete フラグを使用した構成要素の削除の制限

構成要素の削除方法を制限するには、EDML ファイルに delete タグを追加します。次の例は、リンクの href 属性である構成要素を示しています。

```
<a href="<%=MY_URL%>">Link Text</a>
```

この属性構成要素を削除すると、残されるタグは <a>Link Text となり、Dreamweaver のリンクとして表示されなくなります。属性値だけを削除することをお勧めします。削除は、構成要素の EDML ファイルに次のタグを追加することにより実行されます。

```
<delete deleteType="innerOnly"/>
```

もう 1 つの方法は、タグ全体を削除することです。<delete deleteType="tagOnly"/> を入力すると、属性が削除されます。この結果のテキストは **Link Text** となります。

共有メモリ JavaScript ファイル

複数の HTML ファイルが特定の JavaScript ファイルを参照している場合は、その JavaScript が共通の場所に読み込まれ、そこで HTML ファイルが同じ JavaScript ソースを共有できます。これらのファイルには、次の行が含まれます。

```
//SHARE-IN-MEMORY=true
```

JavaScript ファイルに SHARE-IN-MEMORY ディレクティブがあり、HTML ファイルが SCRIPT タグを使用して（SRC 属性を指定）そのディレクティブを参照している場合は、その JavaScript が 1 つの共通の場所に読み込まれ、すべての HTML ファイルにコードが暗黙的に含まれます。

注意：この共通の場所に読み込まれる JavaScript ファイルはメモリを共有するので、JavaScript ファイル間で同じ宣言を記述することはできません。共有メモリファイルに変数または関数が定義され、別の JavaScript ファイルにも同じ変数または関数が定義されると、名前の競合が発生します。新しい JavaScript ファイルを記述する際は、これらのファイルと命名規則に注意します。

第 18 章：データソース

データソースファイルは、Configuration¥DataSources¥**ServerModelName** フォルダに格納されています。Dreamweaver では現在 ASP.Net/C#、ASP.Net/VisualBasic、ASP/JavaScript、ASP/VBScript、Adobe ColdFusion、JSP および PHP/MySQL サーバモデルがサポートされています。各サーバモデルのサブフォルダには、そのサーバモデルのデータソースに関連付けられた HTML ファイルおよび EDML ファイルがあります。

次の表にデータソースを作成するために使用するファイルの一覧を示します。

パス	ファイル	説明
Configuration¥DataSources¥**ServerModelName**	**datasourcename**.htm	データソースの名前と JavaScript サポートファイルの場所を指定します。
Configuration¥DataSources¥**ServerModelName**	**datasourcename**.edml	データソース値を表すために Dreamweaver によってドキュメントに挿入されるコードを定義します。
Configuration¥DataSources¥**ServerModelName**	**datasourcename**.js	必要なコードをドキュメントに追加、挿入、およびドキュメントから削除する JavaScript 関数が含まれます。

データソースの動作

Dreamweaver では、バインディングパネルを使用して動的データを追加できます。+ メニューに表示される動的データオブジェクトは、ページに指定されたサーバモデルに基づいています。例えば、レコードセット、コマンド、リクエスト変数、セッション変数および ASP アプリケーションのアプリケーション変数を挿入することができます。詳しくは、関数 dreamweaver.dbi.getDataSources() について、『Dreamweaver API リファレンス』を参照してください。

以下の手順は、動的データの追加に関連する処理について説明します。

1 ユーザがバインディングパネルの + メニューをクリックすると、ポップアップメニューが表示されます。

メニューの内容を決定するために、まず Dreamweaver によってデータソースと同じフォルダ内で DataSources.xml ファイルが検索されます（Configuration¥DataSources¥ASP_Js¥DataSources.xml など）。DataSources.xml ファイルには、ポップアップメニューの内容が記述されています。これには、ポップアップメニューに配置される HTML ファイルへの参照が記述されています。

Dreamweaver によって、参照される各 HTML ファイルに title タグがあるかどうかがチェックされます。ファイルに title タグが含まれている場合は、title タグのコンテンツがメニューに表示されます。ファイルに title タグがない場合は、メニューにはファイル名が使用されます。

DataSources.xml ファイルの読み取りが完了すると、メニューに表示する必要がある他の項目について、フォルダの残りが検索されます。DataSources.xml ファイルが存在しない場合も同じ処理が行われます。まだメニューに含まれていないファイルがメインフォルダにあれば、メニューに追加されます。まだメニューにないファイルがサブフォルダにある場合は、Dreamweaver によってサブメニューが作成され、そこにファイルが追加されます。

2 + メニューから項目を選択すると、addDynamicSource() 関数が呼び出されます。この関数は、データソースのコードをユーザのドキュメントに追加するために呼び出されます。

3 該当するサーバモデルフォルダ内の各ファイルで、それぞれのファイルの findDynamicSources() 関数が呼び出されます。返された配列の値ごとに、同じファイルで generateDynamicSourceBindings() 関数が呼び出されます。この関数は、ユー

ザのドキュメントの各データソースに含まれるすべてのフィールドの新しいリストを取得するために呼び出されます。これらのフィールドは、動的データダイアログボックス、動的テキストダイアログボックス、またはバインディングパネルで、ツリーコントロールとして表示されます。ASP ドキュメントのデータソースツリーは、次の例のように表示されます。

```
Recordset (Recordset1)
    ColumnOneInRecordset
    ColumnTwoInRecordset
Recordset (Recordset2)
    ColumnOfRecordset
Request
    NameOfRequestVariable
    NameOfAnotherRequestVariable
Session
    NameOfSessionVariable
```

4 ユーザがバインディングパネルのデータソース名をダブルクリックすると、ツリー内の編集処理のために editDynamicSource() 関数が呼び出されます。

5 ユーザが「-」ボタンをクリックすると、ツリーから現在のノード選択内容が取得され、deleteDynamicSource() 関数に渡されます。addDynamicSource() 関数により以前追加されたコードが削除されます。現在の選択内容を削除できない場合は、関数からエラーメッセージが返ります。deleteDynamicSource() 関数から返された後、findDynamicSources() 関数および generateDynamicSourceBindings() 関数が呼び出されることにより、データソースツリーが更新されます。

6 ユーザがデータソースを選択し、動的データダイアログボックスまたは動的テキストダイアログボックスで「OK」をクリックすると、generateDynamicDataRef() 関数が呼び出されます。ユーザがバインディングパネルで挿入またはバインドをクリックしたときにも、同じ処理が行われます。ドキュメントの現在の挿入ポイントに戻り値が挿入されます。

7 ユーザが動的データダイアログボックスを使用して動的データオブジェクトを編集した場合、データソースツリー内の選択内容をオブジェクトに初期化する必要があります。ユーザが動的テキストダイアログボックスを使用して動的データオブジェクトを編集した場合も同様です。ツリーコントロールを初期化する場合、該当するサーバモデルフォルダの各ファイルが検索されます。例えば、Configuration¥DataSources¥ASP_Js フォルダなどです。inspectDynamicDataRef() 関数の実装を呼び出す各ファイルが検索されます。

Dreamweaver によって inspectDynamicDataRef() 関数が呼び出され、動的データオブジェクトがユーザドキュメントのコードからツリーのアイテムに変換されます。これは generateDynamicDataRef() 関数が呼び出されたときの処理と逆の処理になります。inspectDynamicDataRef() 関数が 2 つのエレメントを含む配列を返す場合、現在の選択内容にバウンドされたツリー項目の視覚的なヒントが表示されます。

8 ユーザが選択内容を変更するたびに、inspectDynamicDataRef() 関数が呼び出されます。この関数は、新しい選択内容が動的テキストか、動的属性を持つタグかを判断するために呼び出されます。選択内容が動的テキストである場合、バインディングパネルに現在の選択内容のバインディングが表示されます。

9 動的テキストオブジェクト、またはユーザがページに既に追加した動的属性のデータフォーマットを変更できます。そのためには、動的データダイアログボックス、動的テキストダイアログボックス、またはバインディングパネルを使用します。フォーマットを変更すると、ユーザドキュメントに挿入するストリングを取得するために generateDynamicDataRef() 関数が呼び出されます。その後、そのストリングが formatDynamicDataRef() 関数に渡されます（詳しくは、300 ページの「formatDynamicDataRef()」を参照してください）。formatDynamicDataRef() 関数が返すストリングが、ユーザドキュメントに挿入されます。

簡単なデータソースの例

この拡張機能では、ColdFusion ドキュメントのバインディングパネルにカスタムデータソースが追加されます。ユーザは、必要な変数を新しいデータソースから指定できます。

この例では、MyDatasource というデータソースが作成されます。このデータソースには、MyDatasource.js JavaScript ファイル、MyDatasource_DataRef.edml ファイル、およびユーザが特定の変数名を入力するためのダイアログボックスを実装する MyDatasource 変数コマンドファイルが含まれています。MyDatasource の例は、Cookie 変数データソースと URL 変数データソースの実装に基づいています。これらのデータソースのファイルは、Configuration¥DataSources¥ColdFusion フォルダにあります。

このデータソースを作成するには、定義ファイル、EDML ファイル、JavaScript ファイルおよびサポートコマンドファイルを作成した後、新しいデータソースをテストします。

データソース定義ファイルの作成

データソース定義ファイルは、バインディングパネルの + メニューに表示されるデータソース名と、データソースの実装に使用する JavaScript サポートファイルの場所を Dreamweaver に通知します。

ユーザがバインディングパネルの + メニューをクリックすると、現在のサーバモデルの DataSources フォルダが検索され、このフォルダの HTML（HTM）ファイルで定義されている使用可能なデータソースがすべて収集されます。このため、ユーザが新しいデータソースを利用できるようにするには、データソース名とすべての JavaScript サポートファイルの場所を指定するデータソース定義ファイルを作成する必要があります。そこでは、title タグでデータソース名を指定し、script タグで JavaScript サポートファイルの場所を指定します。

さらに、このデータソースを実装するには、サポートファイルがいくつか必要です。通常、これらのサポートファイルで関数を使用する必要はありませんが、関数を使用すると便利な場合がよくあります。この例では関数の使用が必須です。例えば dwscriptsServer.js ファイルには、このデータソースの実装に使用する dwscripts.stripCFOutputTags() 関数が含まれています。また、DataSourceClass.js ファイルを使用して DataSource クラスのインスタンスを作成し、findDynamicSources() 関数の戻り値として使用します。

1 新しい空白のファイルを作成します。

2 次のコードを入力します。

```
<HTML>
<HEAD>
<TITLE>MyDatasource</TITLE>
<SCRIPT SRC="../../Shared/Common/Scripts/dwscripts.js"></SCRIPT>
<SCRIPT SRC="../../Shared/Common/Scripts/dwscriptsServer.js"></SCRIPT>
<SCRIPT SRC="../../Shared/Common/Scripts/DataSourceClass.js"></SCRIPT>
<SCRIPT SRC="MyDatasource.js"></SCRIPT>
</HEAD>
<body></body>
</HTML>
```

3 ファイルを MyDatasource.htm として Configuration¥DataSources¥ColdFusion フォルダに保存します。

EDML ファイルの作成

EDML ファイルでは、データソース値を表すために Dreamweaver によってドキュメントに挿入されるコードを定義します。EDML ファイルについて詳しくは、240 ページの「サーバビヘイビア」を参照してください。ユーザが特定の値をデータソースからドキュメントに追加すると、実行時に実際の値に変換されるコードが挿入されます。構成要素の EDML ファイルでは、ドキュメントのコードを定義します（詳しくは、259 ページの「EDML 構成要素ファイル」を参照してください）。

MyDatasource 変数に対しては、ColdFusion コード、<cfoutput>#MyXML.***variable***#</cfoutput> が挿入されるようにします。ここで、**variable** は、データソースからユーザが必要とする値です。

1 新しい空白のファイルを作成します。

2 次のコードを入力します。

```
<participant>
    <quickSearch><![CDATA[#]]></quickSearch>
    <insertText location="replaceSelection"><![CDATA[<cfoutput>#MyDatasource.
        @@bindingName@@#</cfoutput>]]></insertText>
    <searchPatterns whereToSearch="tag+cfoutput">
        <searchPattern paramNames="sourceName,bindingName"><![CDATA[/#(?:\s*\w+\s*\()?
        (MyDatasource)\.(\w+)\b[^#]*#/i]]></searchPattern>
    </searchPatterns>
</participant>
```

3 ファイルを MyDatasource_DataRef.edml として Configuration¥DataSources¥ColdFusion フォルダに保存します。

データソースの API 関数を実装する JavaScript ファイルの作成

データソース名、サポートスクリプトファイル名および Dreamweaver の作業ドキュメント用コードを定義した後、Dreamweaver 用の JavaScript 関数を指定して、必要なコードをドキュメントに追加する機能、挿入する機能、およびドキュメントから削除する機能を実現します。

Cookie 変数データソースの構造に基づいて、次の例に示すように MyXML データソースを実装できます。「MyDatasource_Variable」コマンド（addDynamicSource() 関数で使用される）は、「288 ページの「ユーザ入力用のサポートコマンドファイルの作成」」で定義されます。

1 新しい空白のファイルを作成します。

2 次のコードを入力します。

```
//************** GLOBALS VARS *****************
var MyDatasource_FILENAME = "REQ_D.gif";
var DATASOURCELEAF_FILENAME = "DSL_D.gif";

//****************** API **********************
function addDynamicSource()
{
    MM.retVal = "";
    MM.MyDatasourceContents = "";
    dw.popupCommand("MyDatasource_Variable");
    if (MM.retVal == "OK")
    {
        var theResponse = MM.MyDatasourceContents;
        if (theResponse.length)
        {
            var siteURL = dw.getSiteRoot();
            if (siteURL.length)
            {
                dwscripts.addListValueToNote(siteURL, "MyDatasource", theResponse);
            }
            else
            {
                alert(MM.MSG_DefineSite);
            }
        }
        else
        {
        alert(MM.MSG_DefineMyDatasource);
        }
    }
}

function findDynamicSources()
{
    var retList = new Array();
```

```
    var siteURL = dw.getSiteRoot()

    if (siteURL.length)
    {
        var bindingsArray = dwscripts.getListValuesFromNote(siteURL, "MyDatasource");
        if (bindingsArray.length > 0)
        {

    // Here you create an instance of the DataSource class as defined in the
    // DataSourceClass.js file to store the return values.

        retList.push(new DataSource("MyDatasource",
                                    MyDatasource_FILENAME,
                                    false,
                                    "MyDatasource.htm"))
        }
    }

    return retList;
}

function generateDynamicSourceBindings(sourceName)
{
    var retVal = new Array();

    var siteURL = dw.getSiteRoot();

    // For localized object name...
    if (sourceName != "MyDatasource")
    {
        sourceName = "MyDatasource";
    }

    if (siteURL.length)
    {
        var bindingsArray = dwscripts.getListValuesFromNote(siteURL, sourceName);
        retVal = getDataSourceBindingList(bindingsArray,
                                          DATASOURCELEAF_FILENAME,
                                          true,
                                          "MyDatasource.htm");
    }

    return retVal;
}

function generateDynamicDataRef(sourceName, bindingName, dropObject)
{
    var paramObj = new Object();
    paramObj.bindingName = bindingName;
    var retStr = extPart.getInsertString("", "MyDatasource_DataRef", paramObj);

    // We need to strip the cfoutput tags if we are inserting into a CFOUTPUT tag
    // or binding to the attributes of a ColdFusion tag. So, we use the
    // dwscripts.canStripCfOutputTags() function from dwscriptsServer.js

    if (dwscripts.canStripCfOutputTags(dropObject, true))
    {
        retStr = dwscripts.stripCFOutputTags(retStr, true);
    }
    return retStr;
}
```

```
function inspectDynamicDataRef(expression)
{
    var retArray = new Array();

    if(expression.length)
    {
        var params = extPart.findInString("MyDatasource_DataRef", expression);
        if (params)
        {
            retArray[0] = params.sourceName;
            retArray[1] = params.bindingName;
        }
    }

    return retArray;
}
function deleteDynamicSource(sourceName, bindingName)
{
    var siteURL = dw.getSiteRoot();

    if (siteURL.length)
    {
        //For localized object name
        if (sourceName != "MyDatasource")
        {
            sourceName = "MyDatasource";
        }

        dwscripts.deleteListValueFromNote(siteURL, sourceName, bindingName);
    }
}
```

3 ファイルを MyDatasource.js として Configuration¥DataSources¥ColdFusion フォルダに保存します。

ユーザ入力用のサポートコマンドファイルの作成

addDynamicSource() 関数には、「dw.popupCommand("MyDatasource_Variable")」コマンドが含まれています。このコマンドは、ユーザが特定の変数名を入力するためのダイアログボックスを開きます。ただし、MyDatasource 変数については、ダイアログボックスを作成する必要があります。

ユーザにダイアログボックスを提供するには、HTML のコマンド定義ファイルと JavaScript のコマンド実装ファイルのセットを新規作成します。コマンドファイルについて詳しくは、130 ページの「コマンドのしくみ」を参照してください。

コマンド定義ファイルを使用して、サポート実装 JavaScript ファイルの場所を Dreamweaver に指定します。定義ファイルでは、ユーザに表示されるダイアログボックスの形式も指定されます。JavaScript サポートファイルによって、ダイアログボックスのボタンと、ダイアログボックスからのユーザ入力を割り当てる方法が決定されます。

コマンド定義ファイルの作成

1 新しい空白のファイルを作成します。

2 次のコードを入力します。

```
<!DOCTYPE HTML SYSTEM "-//Adobe//DWExtension layout-engine 10.0//dialog">
<html>
<head>
<title>MyDatasource Variable</title>
<script src="MyDatasource_Variable.js"></script>
<SCRIPT SRC="../Shared/MM/Scripts/CMN/displayHelp.js"></SCRIPT>
<SCRIPT SRC="../Shared/MM/Scripts/CMN/string.js"></SCRIPT>
<link href="../fields.css" rel="stylesheet" type="text/css">
</head>
<body>
<form>
    <div ALIGN="center">
        <table border="0" cellpadding="2" cellspacing="4">
            <tr>
                <td align="right" valign="baseline" nowrap>Name:</td>
                <td valign="baseline" nowrap>
                    <input name="theName" type="text" class="medTField">
                </td>
            </tr>
        </table>
    </div>
</form>
</body>
</html>
```

3 ファイルを MyDatasource_Variable.htm として Configuration¥Commands フォルダに保存します。

注意：MyDatasource_Variable.js ファイルは、次の手順で作成する実装ファイルです。

JavaScript サポートファイルの作成

1 新しい空白のファイルを作成します。

2 次のコードを入力します。

```
//****************** API **********************

function commandButtons(){
    return new Array(MM.BTN_OK,"okClicked()",MM.BTN_Cancel,"window.close()");
}

//***************** LOCAL FUNCTIONS*****************

function okClicked(){
    var nameObj = document.forms[0].theName;

    if (nameObj.value) {
        if (IsValidVarName(nameObj.value)) {
            MM.MyDatasourceContents = nameObj.value;
            MM.retVal = "OK";
            window.close();
        } else {
            alert(nameObj.value + " " + MM.MSG_InvalidParamName);
        }
    } else {
        alert(MM.MSG_NoName);
    }
}
```

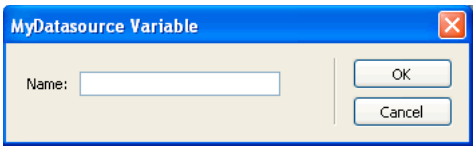
3 ファイルを MyDatasource_Variable.js として Configuration¥Commands フォルダに保存します。

新しいデータソースのテスト

これで、Dreamweaver を開き（既に開いている場合には再起動して）、ColdFusion ファイルを開くか新しいファイルを作成できるようになります。

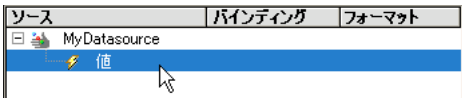
1 ドキュメント内にポインタを配置した状態で、バインディングパネルの + メニューをクリックし、使用可能なデータソースをすべて表示します。MyDatasource がリストの一番下に表示されます。

2 データソースのオプションから MyDatasource をクリックすると、作成した MyDatasource 変数ダイアログボックスが表示されます。

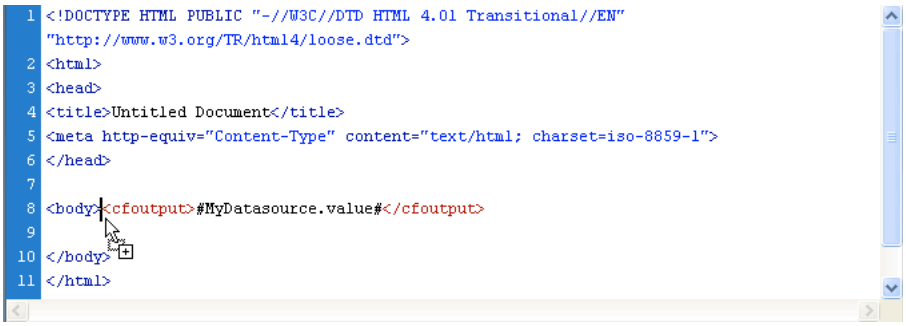


3 ダイアログボックスに値を入力し、「OK」をクリックします。

バインディングパネルにデータソースがツリー形式で表示され、データソース名の下にダイアログボックスで入力した変数が表示されます。



4 変数をドキュメントにドラッグします。これによって、適切なコードが EDML ファイルから挿入されます。



データソース API 関数

データソース API の関数を使用すると、データソースの検索、追加、編集および削除ができます。また、動的なデータオブジェクトを生成し、検査することもできます。

addDynamicSource()

対応バージョン
Dreamweaver UltraDeveloper 1

説明
この関数は、動的データソースを追加します。各データソースファイルにこの関数が実装されているため、+ メニューからデータソースを選択すると、Dreamweaver によって適切な addDynamicSource() 関数の実装が呼び出されます。

例えば、レコードセットまたはコマンドの場合は、ドキュメントに新しいサーバビヘイビアを挿入する dw.serverBehaviorInspector.popupServerBehavior() 関数が呼び出されます。リクエスト変数、セッション変数、またはアプリケーション変数の場合は、変数の名前を収集するための HTML/JavaScript ダイアログボックスが表示されます。収集された変数名は、ビヘイビアによって後で使用するために保存されます。

addDynamicSource() 関数から返された後、Dreamweaver によってデータソースツリーの内容が消去され、findDynamicSources() 関数と generateDynamicSourceBindings() 関数が呼び出されて、データソースツリーに新しい内容が取り込まれます。

戻り値
なし。

deleteDynamicSource()

対応バージョン
Dreamweaver UltraDeveloper 1

説明
ツリー内でデータソースを選択して「-」ボタンをクリックすると、この関数が呼び出されます。

例えば、Dreamweaver でレコードセットまたはコマンドが選択されると、deleteDynamicSource() 関数から dw.serverBehaviorInspector.deleteServerBehavior() 関数が呼び出されます。リクエスト変数、セッション変数、またはアプリケーション変数が選択された場合、この関数では変数が削除されたことが記録され、以後その変数は表示されません。deleteDynamicSource() 関数が返ってきた後、Dreamweaver によってデータソースツリーの内容が消去され、findDynamicSources() 関数および generateDynamicSourceBindings() 関数が呼び出されて、ユーザドキュメントで使用されるすべてのデータソースの新しいリストが取得されます。

引数
sourceName、**bindingName**

- **sourceName** 引数は、子ノードが関連付けられている最上位レベルノードの名前です。
- **bindingName** 引数は、子ノードの名前です。

戻り値
なし。

displayHelp()

説明
この関数が定義されている場合は、ダイアログボックスの「OK」と「キャンセル」の下に「ヘルプ」ボタンが表示されます。この関数は、ユーザが「ヘルプ」ボタンをクリックすると呼び出されます。

引数
なし。

戻り値
なし。

例
```
// The following instance of displayHelp() opens
// a file (in a browser) that explains how to use
// the extension.
function displayHelp(){
    var myHelpFile = dw.getConfigurationPath() +
        '/ExtensionsHelp/superDuperHelp.htm';
    dw.browseDocument(myHelpFile);
}
```

editDynamicSource()

対応バージョン
Dreamweaver MX

説明
ユーザがバインディングパネルで編集するデータソースの名前をダブルクリックすると、この関数が呼び出されます。この関数を実装してツリー内のユーザの編集を処理できます。それ以外の場合は、データソースに一致するサーバビヘイビアが自動的に起動されます。拡張機能のデベロッパーは、この関数を使用してサーバビヘイビアのデフォルトの実装をオーバーライドし、カスタムハンドラを提供できます。

引数
sourceName、**bindingName**

- **sourceName** 引数は、子ノードが関連付けられている最上位レベルノードの名前です。
- **bindingName** 引数は、子ノードの名前です。

戻り値
ブール値。関数によって編集が処理された場合は true、そうでない場合は false が返されます。

findDynamicSources()

対応バージョン
Dreamweaver UltraDeveloper 1

説明
この関数は、動的データダイアログボックス、動的テキストダイアログボックス、またはバインディングパネルに表示される、データソースツリー内の最上位レベルノードを返します。各データソースファイルには、findDynamicSources() 関数の実装があります。Dreamweaver では、ツリーを更新する際に、DataSources フォルダ内のすべてのファイルが読み取られ、各ファイルの findDynamicSources() 関数が呼び出されます。

戻り値
オブジェクトごとにプロパティを 5 つまで指定できる JavaScript オブジェクトの配列。これらのプロパティについては以下で説明します。

- title プロパティには、各親ノードのアイコンの右に表示されるラベルストリングを指定します。title プロパティの指定は、常に必須です。
- imageFile プロパティは、動的データダイアログボックス、動的テキストダイアログボックス、またはバインディングパネルに表示されるツリーコントロールで親ノードを表すアイコン（GIF イメージ）を含むファイルのパスです。このプロパティは必須です。
- allowDelete プロパティはオプションです。このプロパティが false に設定されている場合、ユーザがバインディングパネルでこのノードをクリックすると、「-」ボタンが使用不可になります。このプロパティが true に設定されている場合、マイナス「-」ボタンが使用可能になります。このプロパティを定義しない場合、デフォルトは true になります。
- dataSource プロパティには、findDynamicSources() 関数が定義されているファイルの簡単な名前を指定します。例えば、Configuration¥DataSources/ASP_Js フォルダの Session.htm ファイルに定義されている findDynamicSources() 関数は、dataSource プロパティを session.htm に設定します。このプロパティは必須です。
- name プロパティは、データソースに関連付けられているサーバビヘイビア（存在する場合）の名前です。レコードセットなどの一部のデータソースには、サーバビヘイビアが関連付けられています。レコードセットを作成して、それに rsAuthors と名付けた場合、name 属性を rsAuthors にする必要があります。name プロパティの定義は必須ですが、サーバビヘイビアがセッション変数などのデータソースに関連付けられていない場合は、空白のストリング（""）を指定することもできます。

注意：これらのプロパティを定義する JavaScript クラスは、Configuration¥Shared¥Common¥Scripts フォルダの DataSourceClass.js ファイルにあります。

generateDynamicDataRef()

対応バージョン
Dreamweaver UltraDeveloper 1

説明
この関数は、子ノードの動的データオブジェクトを生成します。

引数
sourceName、**bindingName**

- **sourceName** 引数は、子ノードに関連付けられている最上位レベルノードの名前です。
- **bindingName** 引数は、動的データオブジェクトの生成元となる子ノードの名前です。

戻り値
ユーザのドキュメントに挿入するストリング。挿入の前に、このストリングを formatDynamicDataRef() 関数に渡してフォーマットすることができます。

generateDynamicSourceBindings()

対応バージョン
Dreamweaver UltraDeveloper 1

説明
この関数は、最上位レベルノードの子ノードを返します。

引数
sourceName

- **sourceName** 引数は、返される子ノードの最上位レベルノードの名前です。

戻り値
オブジェクトごとにプロパティを 4 つまで指定できる JavaScript オブジェクトの配列。これらのプロパティについては以下で説明します。

- title プロパティには、各親ノードのアイコンの右に表示されるラベルストリングを指定します。このプロパティは必須です。
- allowDelete プロパティはオプションです。このプロパティが false に設定されている場合、ユーザがバインディングパネルでこのノードをクリックすると、「-」ボタンが使用不可になります。このプロパティが true に設定されている場合、マイナス「-」ボタンが使用可能になります。このプロパティを定義しない場合、デフォルトは true になります。
- dataSource プロパティには、findDynamicSources() 関数が定義されているファイルの簡単な名前を指定します。例えば、Configuration¥DataSources¥ASP_Js フォルダの Session.htm ファイルに定義されている findDynamicSources() 関数は、dataSource プロパティを session.htm に設定します。このプロパティは必須です。
- name プロパティは、データソースに関連付けられているサーバビヘイビア（存在する場合）の名前です。このプロパティの指定は必須です。レコードセットなどの一部のデータソースには、サーバビヘイビアが関連付けられています。レコードセットを作成して、それに rsAuthors と名付けた場合、name プロパティを rsAuthors にする必要があります。セッション変数などその他のデータソースには、対応するサーバビヘイビアがありません。それらの name プロパティは、空白のストリング（""）にする必要があります。

注意：これらのプロパティを定義する JavaScript クラスは、Configuration¥Shared¥Common¥Scripts フォルダの DataSourceClass.js ファイルにあります。

inspectDynamicDataRef()

対応バージョン
Dreamweaver UltraDeveloper 1

説明
この関数は、動的データオブジェクトから、データソースツリー内の対応するノードを特定します。
inspectDynamicDataRef() 関数は、渡されたストリングを、generateDynamicDataRef() からツリーのノードごとに返されるストリングと比較します。ツリー内に一致するノードが見つかった場合は、inspectDynamicDataRef() 関数からそのノードが返されます。この関数では、2 つのエレメントを含む配列を使用して、ノードが識別されます。最初のエレメントは親ノードの名前、2 つ目のエレメントは子ノードの名前です。一致するノードが見つからなかった場合は、inspectDynamicDataRef() 関数から空白の配列が返されます。

inspectDynamicDataRef() 関数の各実装では、それぞれのオブジェクトタイプで一致するものだけがチェックされます。例えば、inspectDynamicDataRef() 関数のレコードセットの実装では、渡されたストリングがツリー内のレコードセットノードと一致する場合にのみ、一致として検出されます。

引数
string

- **string** 引数は動的データオブジェクトです。

戻り値
一致したノードを特定する 2 つのエレメント（親と子の名前）の配列。一致するノードが見つからなかった場合は null 値が返ってきます。

第 19 章：サーバフォーマット

283 ページの「データソース」では、Adobe Dreamweaver でサーバ式を適切な位置に追加して、ユーザのドキュメントに動的データを挿入する方法を説明しています。ビジターが Web サーバからドキュメントを要求すると、このサーバ式がデータベースの値、リクエスト変数のコンテンツまたはその他の動的な値に変換されます。Dreamweaver のサーバフォーマットにより、この動的な値をフォーマットし、ビジターに対してどのように表示するかを指定できます。

データフォーマットのしくみ

Dreamweaver を使用すると、組み込み済みのフォーマットによるデータのフォーマットができます。組み込み済みのフォーマットタイプを基にした新しいフォーマットを作成したり、カスタムフォーマットタイプを基にした新しいフォーマットを作成することもできます。

ユーザはいくつかの方法で動的データをフォーマットできます。データを HTML ドキュメントに挿入する前にフォーマットするには、フォーマットメニューを使用します。フォーマットメニューは、動的データダイアログボックス、動的テキストダイアログボックスまたはバインディングパネルにあります。フォーマットを作成するには、フォーマットメニューから「フォーマットリストを編集」コマンドを選択し、+ メニューからフォーマットタイプを選択します。+ メニューには、フォーマットタイプが一覧表示されます。フォーマットタイプとは、「通貨」、「日付 / 時刻」、「大文字 / 小文字」などの基本的なフォーマットカテゴリーです。フォーマットタイプには、フォーマットカテゴリーの一般的なパラメータがすべて用意されているので、フォーマットを効率よく作成できます。

例えば、通貨フォーマットを作成できます。通貨をフォーマットする際の処理は、数値からストリングへの変換、カンマと小数点の挿入、およびドル（$）などの通貨記号の挿入です。通貨フォーマットデータタイプには一般的なパラメータがすべて用意されており、ユーザは必要なパラメータの値を指定するよう要求されます。

サーバフォーマット API は、283 ページの「データソース」で説明した関数が返す動的データをフォーマットする API を表しています。両方のトピックで説明している関数が連携して、動的データがフォーマットされます。

すべてのフォーマットファイルは Configuration¥ServerFormats¥**currentServerModel** フォルダに格納されています。各サブフォルダには、1 つの XML ファイルと複数の HTML ファイルがあります。

Formats.xml ファイルには、フォーマットメニューのすべての選択肢が記述されています。「フォーマットリストの編集」オプションおよび「なし」オプションは自動的に追加されます。

このフォルダには、現在インストールされているフォーマットタイプごとに HTML ファイルが 1 つ格納されています。このフォーマットタイプには、AlphaCase、Currency、DateTime、Math、Number、Percent、Simple、Trim などがあります。

注意：通貨フォーマットは PHP サーバモデルには使用できません。

Formats.xml ファイル

Formats.xml ファイルには、フォーマットメニューの項目ごとに 1 つの format タグが記述されています。各 format タグは、次の必須属性を含んでいます。

- file=***fileName*** 属性は、"Currency" などのこのフォーマットタイプ用の HTML ファイルです。
- title=***string*** 属性は、フォーマットメニューに表示されるストリングです。例えば、"Currency - default" というストリングです。
- expression=***regexp*** 属性は、このフォーマットを使用する動的データオブジェクトと一致する正規表現です。この正規表現は、動的データオブジェクトに現在適用されているフォーマットを判断します。例えば、"Currency - default" フォーマットの正規表現は "<%\s*=\s*FormatCurrency\(.*, -1, -2, -2,-2\)\s*%>|<%\s*=\s*DoCurrency\(.*, -1, -2, -2, -2\)\s*%>" となり

ます。正規表現属性の値は、ファイル内のすべての format タグ間で一意にする必要があります。つまり、このフォーマットのインスタンスのみが正規表現と一致することを保証するために固有にする必要があります。

- visibility=***[hidden | visible]*** 属性は、フォーマットメニューに値を表示するかどうかを示します。visibility 属性の値が hidden である場合は、フォーマットがフォーマットメニューに表示されません。

format タグには、任意に命名した属性を追加することができます。

一部のデータフォーマット関数は、引数 **format** を必要とします。これは JavaScript オブジェクトです。このオブジェクトは、Formats.xml ファイル内の format タグに対応するノードです。このオブジェクトには、対応する format タグの各属性に対して JavaScript プロパティが設定されています。

次の例では、"Currency - default" ストリング用の format タグを示します。

```
<format file="Currency" title="Currency - default" ¬
expression="<%\s*=\s*FormatCurrency\(.*, -1, -2, -2, -2\)\s*%>|¬
    <%\s*=\s*DoCurrency\(.*, -1, -2, -2, -2\)\s*%>"
NumDigitsAfterDecimal=-1 IncludeLeadingDigit=-2 ¬
    UseParensForNegativeNumbers=-2 GroupDigits=-2/>
```

この場合のフォーマットタイプは、「通貨」です。フォーマットメニューには、"Currency - default" ストリングが表示されます。正規表現 <%\s*=\s*FormatCurrency\(.*, -1, -2, -2,-2\)\s*%>|<%\s*=\s*DoCurrency\(.*, -1, -2, -2, -2\)\s*%> は、ユーザのドキュメントに出現するこのフォーマットを検索します。

NumDigitsAfterDecimal、IncludeLeadingDigit、UseParensForNegativeNumbers および GroupDigits は、「通貨」フォーマットタイプのパラメータですが、必須ではありません。これらのパラメータは、「通貨」フォーマットタイプのパラメータダイアログボックスに表示されます。パラメータダイアログボックスは、ユーザがフォーマットリストを編集ダイアログボックスの + メニューから「通貨」フォーマットタイプを選択すると表示されます。これらのパラメータに指定された値によって、新しいフォーマットが定義されます。

フォーマットリストを編集のプラス（+）メニュー

ServerFormats フォルダ内のファイルをフォーマットリストを編集の + メニューに表示させない場合は、HTML ファイルの 1 行目に次のステートメントを追加します。

```
<!-- MENU-LOCATION=NONE -->
```

Dreamweaver では、まずデータフォーマットと同じフォルダ（Configuration¥ServerFormats¥ASP¥ など）にある ServerFormats.xml ファイルが検索され、メニューの内容が決定されます。ServerFormats.xml ファイルには、フォーマットリストを編集の + メニューの内容が記述されています。これには、このメニューに配置する HTML ファイルへの参照が含まれます。

Dreamweaver によって、参照される各 HTML ファイルに title タグがあるかどうかがチェックされます。ファイルに title タグが含まれている場合は、title タグのコンテンツがメニューに表示されます。ファイルに title タグがない場合は、メニューにはファイル名が使用されます。

ファイルが検索された後、またはこのファイルが存在しない場合は、フォルダの残りの部分がスキャンされ、メニューに表示する他の項目が検索されます。まだメニューに含まれていないファイルがメインフォルダにある場合は、それが追加されます。まだメニューにないファイルがサブフォルダにある場合は、Dreamweaver によってサブメニューが作成され、そこにファイルが追加されます。

データフォーマット関数が呼び出される状況

データフォーマット関数は、以下の状況で呼び出されます。

- 動的データダイアログボックスまたは動的テキストダイアログボックスで、ユーザがデータソースツリーからノードを選択し、フォーマットメニューからフォーマットを選択します。ユーザがフォーマットを選択すると、Dreamweaver によって generateDynamicDataRef() 関数が呼び出され、generateDynamicDataRef() 関数からの戻り値が formatDynamicDataRef() 関数に渡されます。formatDynamicDataRef() 関数からの戻り値は、ダイアログボックスの「コード」設定に表示されます。ユーザが「OK」をクリックすると、コードのストリングがユーザのドキュメントに挿入されます。次に、Dreamweaver によって applyFormat() 関数が呼び出され、関数宣言が挿入されます。詳しくは、293 ページの「generateDynamicDataRef()」を参照してください。ユーザがバインディングパネルを使用する場合も、同様の処理が行われます。
- ユーザがフォーマットを変更するか、または動的データアイテムを削除すると、deleteFormat() 関数が呼び出されます。deleteFormat() 関数は、ドキュメントからサポートスクリプトを削除します。
- ユーザがフォーマットリストを編集ダイアログボックスの + ボタンをクリックすると、特定のサーバモデルのすべてのフォーマットタイプを含むメニューが表示されます。各フォーマットタイプは、Configuration¥ServerFormats¥**currentServerModel** フォルダ内のファイルに対応します。

  ユーザが + メニューから選択したフォーマットがパラメータの指定を必要とする場合、Dreamweaver では onload ハンドラ（body タグに含まれるハンドラ）が実行され、そのフォーマットタイプのパラメータを示すパラメータダイアログボックスが表示されます。ユーザがこのダイアログボックスでフォーマットのパラメータを選択して「OK」をクリックすると、Dreamweaver によって applyFormatDefinition() 関数が呼び出されます。

  パラメータダイアログボックスの表示を必要としないフォーマットを選択した場合、ユーザが + メニューからフォーマットタイプを選択すると、Dreamweaver によって applyFormatDefinition() 関数が呼び出されます。
- その後、ユーザがフォーマットリストを編集ダイアログボックスでフォーマットを選択して「編集」ボタンをクリックすることによって、フォーマットを編集する場合は、パラメータダイアログボックスが表示される前に inspectFormatDefinition() 関数が呼び出されるので、フォームコントロールを適切な値に初期化できます。

サーバフォーマット API 関数

サーバフォーマット API は、以下のデータフォーマット関数で構成されています。

applyFormat()

対応バージョン
Dreamweaver UltraDeveloper 1

説明
この関数では、ユーザのドキュメントにフォーマット関数宣言を追加することにより、そのドキュメントを編集できます。ユーザが動的データダイアログボックス、動的テキストダイアログボックスまたはバインディングパネルの「フォーマット」フィールドからフォーマットを選択すると、Dreamweaver によって、ユーザのドキュメントに 2 つの変更が加えられます。HTML タグの前に適切なフォーマット関数が追加され（まだ追加されていない場合）、適切なフォーマット関数が呼び出されるように、動的データオブジェクトが変更されます。

Dreamweaver では、データフォーマットファイル内で applyFormat() JavaScript 関数が呼び出され、関数宣言が追加されます。また、動的データオブジェクトを変更するために、formatDynamicDataRef() 関数を呼び出します。

applyFormat() 関数では、DOM が使用されて、関数宣言がユーザのドキュメントの最上部に追加されます。例えば、ユーザが通貨／初期設定を選択した場合は、この関数によって Currency 関数宣言が追加されます。

引数
format

- **format** 引数は、適用するフォーマットを記述する JavaScript オブジェクトです。JavaScript オブジェクトは、Formats.xml ファイル内の format タグに対応するノードです。このオブジェクトには、対応する format タグの各属性に対して JavaScript プロパティが設定されています。

戻り値
なし。

applyFormatDefinition()

対応バージョン
Dreamweaver UltraDeveloper 1

説明
フォーマットリストを編集ダイアログボックスを使用して作成されたフォーマットに対する変更をコミットします。

フォーマットリストを編集ダイアログボックスでは、フォーマットを作成、編集および削除することができます。この関数は、ダイアログボックスで行われたフォーマットへのあらゆる変更をコミットするために呼び出されます。この関数では、任意に命名したその他のオブジェクトプロパティを設定することもできます。各プロパティは、Formats.xml ファイル内の format タグの属性として格納されます。

引数
format

- **format** 引数は、JavaScript の format オブジェクトに対応します。この関数では、JavaScript オブジェクトの expression プロパティをフォーマットの正規表現として設定する必要があります。この関数では、任意に命名したその他のオブジェクトプロパティを設定することもできます。各プロパティは、format タグの属性として格納されます。

戻り値
関数が正常終了した場合はフォーマットオブジェクト。エラーが発生した場合は、エラーストリングが返されます。空白のストリングが返された場合、フォームは閉じますが、新しいフォーマットは作成されません。これはキャンセル操作と同じです。

deleteFormat()

対応バージョン
Dreamweaver UltraDeveloper 1

説明
フォーマット関数の宣言をユーザのドキュメントの最上部から削除します。

ユーザが動的データダイアログボックス、動的テキストダイアログボックスまたはバインディングパネルで動的データオブジェクトのフォーマットを変更するか、フォーマットされた動的データオブジェクトを削除すると、Dreamweaver によって deleteFormat() 関数が呼び出されて、ドキュメントの先頭から関数宣言が削除され、動的データオブジェクトから関数呼び出しも削除されます。

deleteFormat() 関数では DOM を使用して現在のドキュメントの先頭から関数宣言を削除します。

引数
format

format 引数は、削除するフォーマットを記述する JavaScript オブジェクトです。JavaScript オブジェクトは、Formats.xml ファイル内の format タグに対応するノードです。

戻り値
なし。

formatDynamicDataRef()

対応バージョン
Dreamweaver UltraDeveloper 1

説明
フォーマット関数の呼び出しを動的データオブジェクトに追加します。ユーザが動的データダイアログボックス、動的テキストダイアログボックスまたはバインディングパネルの「フォーマット」テキストボックスからフォーマットを選択すると、Dreamweaver によって、ユーザのドキュメントに 2 つの変更が加えられます。HTML タグの前に適切なフォーマット関数が追加され（まだ追加されていない場合）、適切なフォーマット関数が呼び出されるように、動的データオブジェクトが変更されます。

Dreamweaver では、データフォーマットファイル内で applyFormat() JavaScript 関数が呼び出され、関数宣言が追加されます。また、動的データオブジェクトを変更するために、formatDynamicDataRef() 関数を呼び出します。

formatDynamicDataRef() 関数は、ユーザが動的データダイアログボックス、動的テキストダイアログボックスまたはバインディングパネルの「フォーマット」テキストボックスからフォーマットを選択すると呼び出されます。ユーザのドキュメントは編集されません。

引数
dynamicDataObject、**format**

- **dynamicDataObject** 引数は、動的データオブジェクトを含むストリングです。
- **format** 引数は、適用するフォーマットを記述する JavaScript オブジェクトです。JavaScript オブジェクトは、Formats.xml ファイル内の format タグに対応するノードです。このオブジェクトには、対応する format タグの各属性に対して JavaScript プロパティが設定されています。

戻り値
動的データオブジェクトの新しい値。

エラーが発生した場合は、一定の条件下で警告メッセージが表示されます。この関数から空白のストリングが返された場合は、「フォーマット」テキストボックスが「なし」に設定されます。

inspectFormatDefinition()

対応バージョン
Dreamweaver UltraDeveloper 1

説明
ユーザがフォーマットリストを編集ダイアログボックスでフォーマットを編集する際に、フォームコントロールを初期化します。

引数
format

format 引数は、適用するフォーマットを記述する JavaScript オブジェクトです。JavaScript オブジェクトは、Formats.xml ファイル内の format タグに対応するノードです。このオブジェクトには、対応する format タグの各属性に対して JavaScript プロパティが設定されています。

戻り値
なし。

第 20 章：コンポーネント

Adobe Dreamweaver は、最も一般的なタイプのコンポーネントの作成を数多くサポートしています。また Dreamweaver では、コンポーネントパネルに表示されるコンポーネントタイプを拡張することができます。

コンポーネントの基本について

プログラマは様々な方策を用いてその作業をカプセル化します。カプセル化については、仮想ブラックボックスに存在するエンティティの作成と考えることができます。カプセル化を使用するには、その機能を知っている必要はありません。ただし、カプセル化の実行に必要な情報とカプセル化の完了後に取得できる情報については、知っておく必要があります。例えば、あるプログラマが、従業員データベースから情報を取得するプログラムを作成するとします。すべてのユーザと他のプログラムは、このプログラムを使用してデータベースを照会することができます。つまり、プログラムは再利用可能です。

経験上、カプセル化を使用する適切に編成されたプログラムは管理、改良、再利用が簡単であることがわかっているからです。各種テクノロジーを使って、プログラマは様々な方法でこのカプセル化を実現することができます。その方策として、関数やモジュールなどが挙げられます。Dreamweaver では、Adobe ColdFusion コンポーネント（CFC）など、いくつかの一般的で最新のカプセル化方策に対してコンポーネントという用語を使用しています。したがって、ユーザが Dreamweaver で Web アプリケーションを構築するとき、CFC を使用すると、コンポーネントパネルが便利です。

Web サービス、JavaBeans または CFC といった最近のテクノロジーから開発されたコンポーネントでは、自身を記述できます。通常、コンポーネントを構成するファイルには、コンポーネントに関する情報が埋め込まれています。この情報を公開または共有するコンポーネント機能をイントロスペクトと呼びます。そのため、Dreamweaver などのプログラムは、コンポーネントが渡す関数のリストをそのコンポーネントに要求できます。（渡されるのは、他のプログラムから読み込むことのできる関数です。）使用しているテクノロジーによっては、そのコンポーネントに関する他の情報を得ることもできます。

コンポーネントパネルの拡張

現在のコンポーネントパネルに表示されないコンポーネントを使用するには、コンポーネントパネルのロジックを拡張します。その結果、新しい種類のコンポーネントをパネルから使用できるようになります。

Dreamweaver のコンポーネントパネルに新しいコンポーネントを追加するには、ユーザの環境で使用できるコンポーネントを見つけます。次に、各コンポーネントに説明を要求します（またはその説明が ASCII ファイルを使用して記述されているかどうかを解析します）。

コンポーネントの位置とコンポーネントの詳細を取得する方法は、テクノロジーによって異なります。また、ASP.NET、JSP/J2EE、ColdFusion などのサーバモデルによっても異なることがあります。したがって、コンポーネントパネルを拡張するために記述する JavaScript は、追加するテクノロジーによって異なります。ここでは、コンポーネントパネルに表示される情報を取得するときに役立つ関数について説明します。ただし、コンポーネントを見つけてイントロスペクトするロジックの多くは、ユーザ自身で記述する必要があります。これには、コンポーネントの内部構造を照会したり、フィールド、メソッド、プロパティを Dreamweaver で使用できるようにする記述が含まれます。

また、ASP.NET、JSP/J2EE、ColdFusion などのサーバモデルで、すべてのコンポーネントタイプをサポートしているわけではありません。例えば、ASP.NET の場合、Web サービスはサポートされますが JavaBeans はサポートされません。Adobe ColdFusion の場合は Web サービスと CFC がサポートされています。コンポーネントパネルに新しいコンポーネントタイプを追加する場合、そのタイプはサーバモデル固有であることが必要です。例えば、Dreamweaver ユーザが ColdFusion サイトをデザインしている場合は、コンポーネントパネルのポップアップメニューに CF コンポーネントが表示されます。

場合によっては、ファイルを変更するために、特定のコンポーネントに関連する関数を呼び出す JavaScript コードの記述が必要になります。

コンポーネントパネルのカスタマイズ

ユーザは、Dreamweaver のコンポーネントパネルで、コンポーネントの読み込みおよび操作ができます。これには、有効になっている各サーバモデルと互換性がある使用可能なコンポーネントタイプがすべて表示されます。例えば、CFC は Adobe ColdFusion ページでのみ機能し、コンポーネントパネル内の Adobe ColdFusion サーバモデルにのみ表示されます。

拡張性があるので、新しいコンポーネントタイプをパネルに追加できます。コンポーネントパネルに新しいコンポーネントタイプを追加するときは、次の一般的な手順に従います。

1 該当するサーバモデルに対して使用可能なコンポーネントタイプのリストにコンポーネントを追加します。

2 コンポーネントパネルまたはダイアログボックス（実装する拡張機能に応じて異なります）にコンポーネントを設定する命令を追加します。この命令は設定手順とも呼ばれ、インタラクティブに番号付けされた手順として表示されます。ユーザが既に完了した手順の横にチェックマークが表示されることを確認します。

3 ユーザのコンピュータまたは現在のサイトのみに存在するコンポーネントタイプのコンポーネントを一覧表示します。

4 ユーザがコンポーネントパネルの「+」ボタンをクリックしたときにコンポーネントを作成します。

また、ユーザがコンポーネントを編集したり、削除したりできるようにすることも可能です。

コンポーネントパネルのファイルのカスタマイズ

Configuration¥Components フォルダには、実装されたサーバモデルごとにサブフォルダがあります。コンポーネントファイルは、Configuration¥Components¥**server-model**¥**ComponentType** フォルダに格納されます。その他のサーバモデルや、サポートされているサーバの拡張機能を追加することもできます（詳しくは、315 ページの「サーバモデル」および 240 ページの「サーバビヘイビア」を参照してください）。

コンポーネントパネルで使用できるカスタムコンポーネントの作成

1 サポートしている JavaScript およびイメージファイルの位置を識別する HTML ファイルを作成します。

2 コンポーネントを有効にする JavaScript を記述します。

3 コンポーネントパネルでのコンポーネントの表示に使用するファイルとして GIF イメージファイルを新規作成するか、既存の GIF イメージファイルを指定します。

コンポーネントタイプがツリーコントロールビューに表示されるようにするには、関連付けられたオプションファイルを作成して、ツリーコントロールにデータを取り込みます。

コンポーネントタイプは、個別の Web ページ、一連の Web ページまたはサイト全体の 3 つのレベルで機能するように設定できます。JavaScript コードには、コンポーネントを持続するためのロジックを含める必要があります。例えば、セッション間でコンポーネント自体を保存し、新しいセッションの開始時に再度読み込む処理がこれに該当します。

新しいライトウェイトディレクトリアクセスプロトコル（LDAP）サービスコンポーネントの追加

1 Configuration¥Components¥Common¥WebServices アプリケーションフォルダ内のファイルなど、既存のコンポーネントタイプのファイルをモデルとして使用して、Dreamweaver のコンポーネントパネルに新しいコンポーネントタイ

プを表示するために必要なすべてのファイルに加え、オプションのファイルを任意に作成します。これらのファイルを以下の表に示します。

ファイル名	説明	必須 / オプション
*.htm	その他のサポートされている JavaScript ファイルおよび GIF ファイルを識別する拡張機能ファイル。	必須
*.js	コンポーネント API コールバックを実装する拡張機能ファイル。	必須
*.gif	コンポーネントポップアップメニューに表示されるイメージ。	必須
*Menus.xml	コンポーネントパネル構造を編成するメタデータのリポジトリ。一般的な WebServices コンポーネントではこのファイルは使用されません。Components¥ColdFusion¥WebServices アプリケーションフォルダにある WebServicesMenus.xml ファイルは、このファイルの一例です。	オプション
*.gif	次の例に示すような、有効にも無効にもできるツールバーのイメージ：ToolBarImageUp.gif ToolBarImageDown.gif ToolBarImageDisabled.gif またはツリーノードイメージ。	オプション

注意：1 つのコンポーネントに対応するすべてのファイルで同じ接頭辞を使用します。これにより、各ファイルとそれに対応するコンポーネントを識別しやすくなります。

2 JavaScript コードを記述し、新しいサーバコンポーネントを実装します。

拡張機能ファイル（HTM）では、SCRIPT タグで JavaScript コードの位置を定義します。これらの JavaScript ファイルは、Shared フォルダ、拡張機能ファイルと同じフォルダ、または複数のサーバモデルに適用するコードを格納する Common フォルダに格納できます。

例えば、Configuration¥Components¥Common¥WebServices¥ フォルダの WebServices.htm ファイルには次の行が含まれます。

```
<SCRIPT SRC="../../Common/WebServices/WebServicesCommon.js"></SCRIPT>
```

使用できるコンポーネント API 関数について詳しくは、305 ページの「コンポーネントパネル API 関数」を参照してください。

新しいサービスを追加する際は、コンポーネントパネルを使用してメタ情報を参照し、拡張機能を作成しながら簡単に情報を見ることができます。Dreamweaver では、追加されたコンポーネントを参照し、ノードをコンポーネントツリーに表示できます。コンポーネントパネルでは、コードビューでドラッグ＆ドロップとキーボードがサポートされています。

ツリーコントロールのプロパティ

ComponentRec プロパティを使用して、コンポーネントパネルのツリーコントロールにデータを取り込み、コンポーネントパネルの適切な位置に表示されるようにします。ツリーコントロールのすべてのノードでは、以下のプロパティが必要です。

プロパティ名	説明	必須 / オプション
name	ツリーノード項目の名前	必須
image	ツリーノード項目のアイコン。指定されていない場合、デフォルトのアイコンが使用されます。	オプション
hasChildren	ツリーコントロールの「+」ボタンおよび「-」ボタンのクリックに応答して子を読み込みます。値が取り込まれていないツリーを操作できます。	必須

プロパティ名	説明	必須 / オプション
toolTipText	ツリーノード項目のツールヒントテキスト	オプション
isCodeViewDraggable	項目をコードビューにドラッグ＆ドロップできるかどうかを決定します。	オプション
isDesignViewDraggable	項目をデザインビューにドラッグ＆ドロップできるかどうかを決定します。	オプション

例えば、次の WebServicesClass ノードには、その子として Web メソッドがあります。

```
this.name = "TrafficLocatorWebService";
this.image = "Components/Common/WebServices/WebServices.gif";
this.hasChildren = true;
this.toolTipText = "TrafficLocatorWebService";
this.isCodeViewDraggable = true;
// the following allows of enabling/disabling of the button that appears
// above the Component Tree
this.allowDelete = true;
this.isDesignViewDraggable = false;
```

コンポーネントパネル API 関数

この節では、コンポーネントパネルにデータを取り込むための API 関数について説明します。

getComponentChildren()

対応バージョン
Dreamweaver MX

説明
この関数は、アクティブな親 ComponentRec オブジェクトの子 ComponentRec オブジェクトのリストを返します。ルートレベルのツリー項目を読み込むには、この関数によって、関数の永続的な記憶域にあるメタデータを読み取る必要があります。

引数
{parentComponentRec}

parentComponentRec 引数は、親の componentRec オブジェクトです。これを省略すると、ルートノードの ComponentRec オブジェクトのリストが表示されます。

戻り値
ComponentRec オブジェクトの配列。

例
Configuration¥Components¥Common¥WebServices フォルダにある WebServices.js ファイルの function getComponentChildren(componentRec) を参照してください。

getContextMenuId()

対応バージョン
Dreamweaver MX

説明
コンポーネントタイプのコンテキストメニュー ID を返します。各コンポーネントタイプに、コンテキストメニューを関連付けることができます。コンテキストメニューのポップアップメニューは、**ComponentName**Menus.xml ファイルで定義されていて、menu.xml ファイルと同様に機能します。メニューストリングは静的または動的です。キーボードショートカット（アクセラレータキー）がサポートされます。

引数
なし。

戻り値
コンテキストメニュー ID を定義するストリング。

例
以下の例では、Adobe ColdFusion サーバモデルに関連付けられた CFC のコンポーネントパネルのオプションメニューを設定します。また、このメニューのキーボードショートカットも定義します。

```
function getContextMenuId()
{
    return "DWCFCsContext";
}
```

メニューの DWWebServicesContext は、Configuration¥Components¥ColdFusion¥CFCs¥CFCsMenus.xml ファイル内で次のように定義されています。

```
<menubar xmlns:MMString="http://www.macromedia.com/schemes/dat/string/" name="" id="DWCFCsContext">
    <menu MMString:name="Components/ColdFusion/CFCs/CFCsMenus_xml/DWContext_CFCs/menu/name"
id="DWContext_CFCs">
    <menuitem
MMString:name="Components/ColdFusion/CFCs/CFCsMenus_xml/DWContext_CFCs_createNewCFC/menuitem/name"
domRequired="false" enabled="true" command="createCFC()" id="DWContext_CFCs_createNewCFC" />
    <menuitem
MMString:name="Components/ColdFusion/CFCs/CFCsMenus_xml/DWContext_CFCs_editCode/menuitem/name"
domRequired="false" enabled="canGetSelectedCFC()" command="editCFC();" id="DWContext_CFCs_editCode" />
<separator/>
    <menuitem
MMString:name="Components/ColdFusion/CFCs/CFCsMenus_xml/DWContext_CFCs_getDetails/menuitem/name"
domRequired="false" enabled="canGetDetails()" command="getDetails()" id="DWContext_CFCs_getDetails" />
    <menuitem
MMString:name="Components/ColdFusion/CFCs/CFCsMenus_xml/DWContext_CFCs_getDescription/menuitem/name"
domRequired="false" enabled="canGetSelectedCFC()" command="getDescription()"
id="DWContext_CFCs_getDescription" />
    <separator/>
    <menuitem
MMString:name="Components/ASP_NET_Csharp/Connections/ConnectionsMenus_xml/DWShortcuts_ServerComponent_Ins
ert/menuitem/name" domRequired="false" enabled="insertCFCEnabled();" command="clickedInsertCFC();"
id="DWShortcuts_ServerComponent_Insert" />
    </menu>
</menubar>
```

getCodeViewDropCode()

対応バージョン
Dreamweaver MX

説明
この関数は、コンポーネントパネルからコードビューにドラッグ、カットまたはコピーするコードを取得します。

引数
componentRec

- ***componentRec*** 引数はオブジェクトです。

戻り値
コンポーネントのコードを格納したストリング。

例
以下の例では、Adobe ColdFusion コンポーネント（CFC）のコードを識別します。

```
  function getCodeVi e wDropCode(compone nt  Rec )
{
    var codeToDrop="";
    if (componentRec)
     {
        if  (componentRec.objectType == "Connection")
        {
            v ar  connPart = new Participant("d at asource_tag");
            var paramObj = new Obje ct( );
            paramObj.datasource = componentRec.name;
            codeToDrop =  con nPart.getInsertString(paramObj, "aboveHTML");}
        el se if ((componentRec.objectType == "Column")| |
                 (componentRec.objectType == "Paramete r "))
        {
            c  odeToDrop = componentRec.dropcode;
        }
        e ls e{
            codeToDrop =  co mponentRec.name;
            }
        }
        return codeToDrop;
}
```

getSetupSteps()

対応バージョン
Dreamweaver MX

説明
setupStepsCompleted() 関数によりゼロまたは正の整数が返される場合に、この関数が呼び出されます。この関数は、サーバサイド設定手順を制御します。これは、モーダルダイアログボックスを使用する拡張機能と、サーバコンポーネントを使用する拡張機能を使用して実装できます。

この関数では、拡張機能タイプに応じて、セットアップのステップダイアログボックスまたはコンポーネントパネルのいずれかに表示されるストリングの配列が返されます。

引数
なし。

戻り値
n + 1 個のストリングの配列。**n** は、以下に示す手順の数です。

- 設定手順のリスト上に表示されるタイトル。
- 手順ごとのテキストの説明。ここには、li タグ内に有効な HTML マークアップを入れることができます。

  リンク（a タグ）を以下の形式で手順のリストに含めることができます。

  ```
  <a href="#" onMouseDown="handler">Blue Underlined Text</a>
  ```

  "handler" 値 は、以下のストリングのいずれか、または "dw.browseDocument('http://www.adobe.com')" などの任意の JavaScript 式で置き換えることができます。
- "Event:SetCurSite" ハンドラは、現在のサイトを設定するためのダイアログボックスを表示します。
- "Event:CreateSite" ハンドラは、新しいサイトを作成するためのダイアログボックスを表示します。
- "Event:SetDocType" ハンドラは、ユーザのドキュメントのドキュメントタイプを変更するためのダイアログボックスを表示します。
- "Event:CreateConnection" ハンドラは、新しいデータベース接続を作成するためのダイアログボックスを表示します。
- "Event:SetRDSPassword" ハンドラは、Remote Development Service（RDS）ユーザ名とパスワードを設定するためのダイアログボックスを表示します（Adobe ColdFusion のみ）。
- "Event:CreateCFDataSource" ハンドラは、ブラウザに Adobe ColdFusion Administrator を表示します。

例
以下の例では、Adobe ColdFusion コンポーネントに 4 つの手順を設定し、4 番目の手順にハイパーテキストリンクを設定してユーザが RDS ユーザ名とパスワードを入力できるようにします。

```
function getSetupSteps()
{
    var doSDK = false;
    dom = dw.getDocumentDOM();
    if (dom && dom.serverModel)
    {
        var aServerModelName = dom.serverModel.getDisplayName();
    }
    else
    {
        var aServerModelName = site.getServerDisplayNameForSite();
    }
    if (aServerModelName.length)
    {
        if(aServerModelName != "ColdFusion")
        {
            if(needsSDKInstalled != null)
            {
                doSDK = needsSDKInstalled();
            }
        }
    }

    var someSteps = new Array();
    someSteps.push(MM.MSG_WebService_InstructionsTitle);
    someSteps.push(MM.MSG_Dynamic_InstructionsStep1);
    someSteps.push(MM.MSG_Dynamic_InstructionsStep2);
    if(doSDK == true)
    {
        someSteps.push(MM.MSG_WebService_InstructionsStep3);
    }
    someSteps.push(MM.MSG_WebService_InstructionsStep4);

    return someSteps;
}
```

setupStepsCompleted()

対応バージョン
Dreamweaver MX

説明
この関数は、「コンポーネント」タブが表示される前に呼び出されます。次に、getSetupSteps() 関数が呼び出されます（setupStepsCompleted() 関数からゼロまたは正の整数が返された場合）。

引数
なし。

戻り値
ユーザが既に完了した設定手順の数を表す整数。

- ゼロまたは正の整数の値は、完了した手順の数を示します。
- 値が -1 の場合は、必要な手順がすべて完了したため、手順リストが表示されないことを示します。

handleDesignViewDrop()

対応バージョン
Dreamweaver MX

説明
データベースパネルからデザインビューにテーブルやビューをドラッグ& ドロップする操作、またはコンポーネントパネルからデザインビューにコンポーネントをドラッグ& ドロップする操作を処理します。

引数
componentRec

- **componentRec** 引数は、以下のプロパティを含むオブジェクトです。
- name プロパティはツリーノード項目の名前です。
- image プロパティはツリーノード項目のオプションのアイコンです。省略すると、デフォルトのアイコンが使用されます。
- hasChildren プロパティは、ツリーノード項目が展開可能であるかどうかを示すブール値です。true の場合、ツリーノード項目に「+」および「-」ボタンが表示されます。false の場合、ツリーノード項目は展開できません。
- toolTipText プロパティはツリーノード項目のオプションのツールヒントテキストです。
- isCodeViewDraggable プロパティはツリーノード項目をコードビューにドラッグ& ドロップできるかどうかを示すブール値です。
- isDesignViewDraggable プロパティはツリーノード項目をデザインビューにドラッグ& ドロップできるかどうかを示すブール値です。

戻り値
ドロップ操作が成功したかどうかを示すブール値。成功した場合は true、そうでない場合は false。

例
以下の例では、コンポーネントがテーブルなのかビューなのかを判別して、適切な bHandled 値を返します。

```
function handleDesignViewDrop(componentRec)
{
    var bHandled = false;
    if (componentRec)
    {
        if ((componentRec.objectType == "Table")||
            (componentRec.objectType == "View"))
        {
            alert("popup Recordset Server Behavior");
            bHandled = true;
        }
    }
    return bHandled;
}
```

handleDoubleClick()

対応バージョン
Dreamweaver MX

説明

ユーザがツリー内のノードをダブルクリックすると、イベントハンドラが呼び出され、編集が可能になります。この関数はオプションです。この関数は、イベントハンドラが定義されないことを示す false 値を返す場合もあります。その場合、ダブルクリックすると、デフォルトのビヘイビアであるツリーノードの展開または縮小が実行されます。

引数

componentRec

- **componentRec** 引数は、以下のプロパティを含むオブジェクトです。
- name プロパティはツリーノード項目の名前です。
- image プロパティはツリーノード項目のオプションのアイコンです。このアイコンを省略すると、デフォルトのアイコンが使用されます。
- hasChildren プロパティは、ツリーノード項目が展開可能であるかどうかを示すブール値です。true の場合、ツリーノード項目に「+」および「-」ボタンが表示されます。false の場合、ツリーノード項目は展開できません。
- toolTipText プロパティはツリーノード項目のオプションのツールヒントテキストです。
- isCodeViewDraggable プロパティはツリーノード項目をコードビューにドラッグ& ドロップできるかどうかを示すブール値です。
- isDesignViewDraggable プロパティはツリーノード項目をデザインビューにドラッグ& ドロップできるかどうかを示すブール値です。

戻り値

なし。

例

以下の例では、拡張機能でツリーノード項目のダブルクリック処理が可能です。false 値が返された場合、デフォルトのビヘイビアはノードの拡張と縮小です。

```
function handleDoubleClick(componentRec)
{
    var selectedObj = dw.serverComponentsPalette.getSelectedNode();
    if(dwscripts.IS_WIN)
    {
        if (selectedObj && selectedObj.wsRec && selectedObj.wsRec[ProxyGeneratorNamePropName])
        {
            if (selectedObj.objectType == "Root")
            {
                editWebService();
                return true;
            }
            else if (selectedObj.objectType == "MissingProxyGen")
            {
                displayMissingProxyGenMessage(componentRec);
                editWebService();
                return true;
            }
        }
    }
    return false;
}
```

toolbarControls()

対応バージョン
Dreamweaver MX

説明
各コンポーネントタイプは、toolBarButtonRec オブジェクトのリストを返します。これは左から右の順でツールバーアイコンを表します。各 toolBarButtonRec オブジェクトには、以下のプロパティがあります。

プロパティ名	説明
image	イメージファイルへのパス。
disabledImage	オプション。「ツールバー」ボタンが無効な状態のイメージの検索パス。
pressedImage	オプション。「ツールバー」ボタンを押した状態のイメージの検索パス。
toolTipText	「ツールバー」ボタンのツールヒント。
toolStyle	左 / 右。
enabled	ブール値（true または false）を返す JavaScript コード。以下の条件に該当する場合に、イネーブラが呼び出されます。 • dreamweaver.serverComponents.refresh() が呼び出されたとき。 • ツリー内の選択範囲が変更されたとき。 • サーバモデルが変更されたとき。
command	実行される JavaScript コード。コマンドハンドラは、dreamweaver. serverComponents.refresh() 関数を使用して更新を強制できます。
menuId	ポップアップメニューボタンをクリックしたときの、ボタン固有のメニュー ID。この ID が存在する場合、これによってコマンドハンドラがオーバーライドされます。つまり、ボタンは、コマンドに関連付けられたボタン、またはコマンドに関連付けられたポップアップメニューを表示するボタンとなり、同時に両方にはなりえません。

引数
なし。

戻り値
左から右に並べたツールバーボタンの配列。

例
以下の例では、プロパティをツールバーのボタンに割り当てます。

```
function toolbarControls()
{
    var toolBarBtnArray = new Array();
    dom = dw.getDocumentDOM();
    var plusButton = new ToolbarControlRec();
    var aServerModelName = null;
    if (dom && dom.serverModel)
    {
        aServerModelName = dom.serverModel.getDisplayName();
    }
    else
    {
        //look in the site for potential server model
        aServerModelName = site.getServerDisplayNameForSite();
    }
    if (aServerModelName.length)
    {
        if(aServerModelName == "ColdFusion")
        {
            plusButton.image           = PLUS_BUTTON_UP;
            plusButton.pressedImage    = PLUS_BUTTON_DOWN;
            plusButton.disabledImage   = PLUS_BUTTON_UP;
            plusButton.toolStyle       = "left";
            plusButton.toolTipText    = MM.MSG_WebServicesAddToolTipText;
            plusButton.enabled        = "dwscripts.IS_WIN";
            plusButton.command        = "invokeWebService()";
        }
        else
        {
            plusButton.image          = PLUSDROPBUTTONUP;
            plusButton.pressedImage   = PLUSDROPBUTTONDOWN;
            plusButton.disabledImage  = PLUSDROPBUTTONUP;
            plusButton.toolStyle      = "left";
            plusButton.toolTipText    = MM.MSG_WebServicesAddToolTipText;
            plusButton.enabled        = "dwscripts.IS_WIN";
            plusButton.menuId         = "DWWebServicesChoosersContext";
        }
        toolBarBtnArray.push(plusButton);

        var minusButton            = new ToolbarControlRec();
        minusButton.image          = MINUSBUTTONUP;
        minusButton.pressedImage   = MINUSBUTTONDOWN;
        minusButton.disabledImage  = MINUSBUTTONDISABLED;
        minusButton.toolStyle      = "left";
        minusButton.toolTipText    = MM.MSG_WebServicesDeleteToolTipText;
        minusButton.command        = "clickedDelete()";
        minusButton.enabled  = "(dw.serverComponentsPalette.getSelectedNode() != null &&
            dw.serverComponentsPalette.getSelectedNode() &&
            ((dw.serverComponentsPalette.getSelectedNode().objectType=='Root') ||
            (dw.serverComponentsPalette.getSelectedNode().objectType == 'Error') ||
            (dw.serverComponentsPalette.getSelectedNode().objectType ==
            'MissingProxyGen')))";
        toolBarBtnArray.push(minusButton);

        if(aServerModelName != null && aServerModelName.indexOf(".NET") >= 0)
        {
            var deployWServiceButton           = new ToolbarControlRec();
            deployWServiceButton.image         = DEPLOYSUPPORTBUTTONUP;
            deployWServiceButton.pressedImage  = DEPLOYSUPPORTBUTTONDOWN;
            deployWServiceButton.disabledImage = DEPLOYSUPPORTBUTTONUP;
            deployWServiceButton.toolStyle     = "right";
            deployWServiceButton.toolTipText   = MM.MSG_WebServicesDeployToolTipText;
```

```
            deployWServiceButton.command  = "site.showTestingServerBinDeployDialog()";
            deployWServiceButton.enabled  = true;
            toolBarBtnArray.push(deployWServiceButton);
        }
        //add the rebuild proxy button for windows only.
        //bug 45552:
        if(navigator.platform.charAt(0) !="M")
        {
            var proxyButton            = new ToolbarControlRec();
            proxyButton.image          = PROXYBUTTONUP;
            proxyButton.pressedImage   = PROXYBUTTONDOWN;
            proxyButton.disabledImage  = PROXYBUTTONDISABLED;
            proxyButton.toolStyle      = "right";
            proxyButton.toolTipText    = MM.MSG_WebServicesRegenToolTipText;
            proxyButton.command        = "reGenerateProxy()";
            proxyButton.enabled        = "enableRegenerateProxyButton()";
            toolBarBtnArray.push(proxyButton);
        }
    }
    return toolBarBtnArray;
}
```

第 21 章：サーバモデル

サーバモデルとは、サーバ上でスクリプトを実行するテクノロジーのことです。ユーザは、新しいサイトを定義する際に、サイトレベルおよび個別のドキュメントレベルで使用するサーバモデルを識別できます。このサーバモデルによって、ユーザがドキュメントに追加するすべての動的エレメントが処理されます。

サーバモデル設定ファイルは、Configuration¥ServerModels フォルダに格納されています。このフォルダ内には、各サーバモデル独自の HTML ファイルが格納されています。このファイルで、サーバモデルが要求する関数のセットが実装されます。

サーバモデルのカスタマイズ

サーバモデル API で使用できる関数で、サーバモデルのいくつかの機能をカスタマイズできます。

Adobe Dreamweaver では、新規ユーザが初めて Dreamweaver を起動すると、サーバモデルを識別するように要求するメッセージが表示されます。ユーザがサーバモデルを識別しない場合に備えて、ユーザに必要な手順を完了するように要求する動的ダイアログボックスを作成できます。このダイアログボックスは、ユーザがサーバオブジェクトを挿入しようとすると表示されます。このダイアログボックスの作成について詳しくは、307 ページの「getSetupSteps()」関数および 309 ページの「setupStepsCompleted()」関数を参照してください。

必要に応じて、特化されたサーバモデルを作成することもできます。Dreamweaver 付属のサーバモデルを編集するのではなく、サーバモデルを新規に作成することをお勧めします。サーバモデルでサポートされるドキュメントタイプの新規作成について詳しくは、13 ページの「Dreamweaver で拡張可能なドキュメントタイプ」を参照してください。

新しいサーバモデルを作成する際は、サーバモデルファイルに canRecognizeDocument() 関数の実装を追加する必要があります。複数のサーバモデルが 1 つのファイル拡張子を要求した場合に、そのファイル拡張子の処理についてユーザのサーバモデルに与えられる優先レベルがこの関数で指定されます。

サーバモデル API の関数

この節では、Dreamweaver にサーバモデルを設定する関数について説明します。

canRecognizeDocument()

対応バージョン
Dreamweaver MX

説明
複数のサーバモデルが 1 つのファイル拡張子を要求している場合、ドキュメントを開くと、その拡張子と関連付けられたサーバモデルごとにこの関数が呼び出されます。これにより、そのドキュメントが特定の関数のファイルであると認識されるかどうかが調べられます。複数のサーバモデルが同じファイル拡張子を要求している場合、最も大きい整数を返したサーバモデルが優先されます。

注意：Dreamweaver で定義されているすべてのサーバモデルでは 1 の値が返されるため、サードパーティのサーバモデルによるファイル拡張子の関連付けを優先するように設定できます。

引数
dom

dom 引数は、Adobe ドキュメントオブジェクトです。これは、dreamweaver.getDocumentDOM() 関数によって返されます。

戻り値
ファイル拡張子について、サーバモデルに指定した優先度を示す整数。サーバモデルがファイル拡張子を要求しない場合は -1 の値が返され、それ以外の場合はゼロより大きい値が返されます。

例
次の例では、ユーザが現在のサーバモデルの JavaScript ドキュメントを開くと、サンプルコードにより値 2 が返されます。この値によって、現在のサーバモデルは Dreamweaver のデフォルトのサーバモデルよりも優先されます。

```
var retVal = -1;
var langRE = /@\s*language\s*=\s*(\"|\')?javascript(\"|\')?/i;
// Search for the string language="javascript"
var oHTML = dom.documentElement.outerHTML;
if (oHTML.search(langRE) > -1)
    retVal = 2;
return retVal;
```

getFileExtensions()

対応バージョン
Dreamweaver UltraDeveloper 1、Dreamweaver MX では非推奨

説明
サーバモデルが対応しているドキュメントファイルの拡張子を返します。例えば、ASP サーバモデルでは .asp および .htm ファイル拡張子がサポートされています。この関数からは、ストリングの配列が返されます。Dreamweaver では、これらのストリングがサイト定義ダイアログボックスの「アプリケーションサーバ」カテゴリーに表示される「ページの拡張」リストに取り込まれます。

注意：「ページの拡張」リストは、Dreamweaver 4 以前のバージョンでのみ表示されます。DreamweaverMX 以降では、サイト定義ダイアログボックスにファイル拡張子設定のリストは表示されません。代わりに、Extensions.txt ファイルが読み取られ、mmDocumentTypes.xml ファイルのエレメントが解析されます。この 2 つのファイルおよび documenttype エレメントについて詳しくは、13 ページの「Dreamweaver で拡張可能なドキュメントタイプ」を参照してください。

引数
なし。

戻り値
対応可能なファイル拡張子を表すストリングの配列。

getLanguageSignatures()

対応バージョン
Dreamweaver MX

説明
この関数は、スクリプト言語が使用するメソッドおよび配列の署名を記述するオブジェクトを返します。getLanguageSignatures() 関数は、以下のエレメントについて一般的な署名マッピングを言語固有のマッピングに対応付ける際に使用されます。

- 関数
- コンストラクタ
- ドロップコード（戻り値）
- 配列
- 例外
- プリミティブデータタイプに対するデータタイプマッピング

getLanguageSignatures() では、これらの署名宣言のマップが返されます。拡張機能を開発する際にこのマップを使用すると、ユーザが Web サービスメソッドなどをドラッグ& ドロップしたときに、ページ上にドロップされる言語固有のコードブロックを、そのページの適切なサーバモデルに基づいて生成できます。

この関数の記述方法の例については、JSP および ASP.NET サーバモデルの HTML 実装ファイルを参照してください。サーバモデル実装ファイルは、Configuration¥ServerModels フォルダに格納されています。

引数
なし。

戻り値
スクリプト言語の署名を定義するオブジェクト。このオブジェクトによって、一般的な署名が言語固有の署名にマップされます。

getServerExtension()

対応バージョン
Dreamweaver UltraDev 4（英語版）、Dreamweaver MX では非推奨

説明
この関数は、現在のサーバモデルを使用するファイルのデフォルトのファイル拡張子を返します。ユーザドキュメントが選択されていない場合は、現在選択されているサイトのサーバモデルが serverModel オブジェクトとして設定されます。

引数
なし。

戻り値
サポートされているファイル拡張子を表すストリング。

getServerInfo()

対応バージョン
Dreamweaver MX

説明
この関数は、JavaScript コード内からアクセスできる JavaScript オブジェクトを返します。このオブジェクトは、dom.serverModel.getServerInfo() JavaScript 関数を呼び出して取得できます。また、serverName、serverLanguage および serverVersion は特殊なプロパティで、以下の JavaScript 関数を使用してアクセスできます。

```
dom.serverModel.getServerName()
dom.serverModel.getServerLanguage()
dom.serverModel.getServerVersion()
```

引数
なし。

戻り値
サーバモデルのプロパティを含むオブジェクト。

例

```
var obj = new Object();
obj.serverName = "ASP";
obj.serverLanguage = "JavaScript";
obj.serverVersion = "2.0";
...
return obj;
```

getServerLanguages()

対応バージョン
Dreamweaver UltraDeveloper 1、Dreamweaver MX では非推奨

説明
この関数は、サーバモデルでサポートされているスクリプト言語をストリングの配列で返します。Dreamweaver では、これらのストリングがサイト定義ダイアログボックスの「アプリケーションサーバ」カテゴリーに表示される「スクリプト言語」リストに取り込まれます。

注意：「スクリプト言語」リストは Dreamweaver 4 以前のバージョンでのみ表示されます。DreamweaverMX 以降では、サポートされるスクリプト言語のリストはサイト定義ダイアログボックスに表示されず、getServerLanguages() 関数も使用されません。Dreamweaver では、サーバモデルごとにサーバ言語は 1 つのみなので、この関数は使用されません。

Dreamweaver の以前のバージョンでは、1 つのサーバモデルで複数のスクリプト言語をサポートできました。例えば、ASP サーバモデルでは JavaScript と VBScript がサポートされていました。

ServerFormats フォルダ内のファイルを特定のスクリプト言語のみに適用する場合は、HTML ファイルの 1 行目に次のステートメントを追加します。

```
<!-- SCRIPTING-LANGUAGE=XXX -->
```

この例では、XXX はスクリプト言語を表します。このステートメントを使用すると、現在選択されているスクリプト言語が XXX の場合のみ、サーバビヘイビアパネルの + メニューにサーバビヘイビアが表示されます。

引数
なし。

戻り値
サポートされているスクリプト言語を表すストリングの配列。

getServerModelExtDataNameUD4()

対応バージョン
Dreamweaver MX

説明
この関数は、Configurations¥ExtensionData フォルダにある UltraDeveloper 4 拡張機能データファイルにアクセスするために、Dreamweaver で使用するサーバモデル実装名を返します。

引数
なし。

戻り値
"ASP/JavaScript" などのストリング。

getServerModelDelimiters()

対応バージョン
Dreamweaver MX

説明
この関数は、アプリケーションサーバで使用されるスクリプト区切りを返し、コードブロックのマージにその区切りを使用できるかどうかを示します。この戻り値に JavaScript からアクセスするには、dom.serverModel.getDelimiters() 関数を呼び出します。

引数
なし。

戻り値
オブジェクトの配列。各オブジェクトには、以下の 3 つのプロパティを指定します。

- **startPattern** プロパティは、開始のスクリプト区切りに一致する正規表現（<% など）です。
- **endPattern** プロパティは、終了のスクリプト区切りに一致する正規表現（%> など）です。
- **participateInMerge** プロパティは、区切りで囲まれたコンテンツがブロックのマージに含まれるか（true）、含まれないか（false）を示すブール値です。

getServerModelDisplayName()

対応バージョン
Dreamweaver MX

説明
この関数は、このサーバモデルのユーザインターフェイスに表示される名前を返します。dom.serverModel.getDisplayName() 関数を呼び出して、JavaScript からこの値にアクセスできます。

引数
なし。

戻り値
"ASP JavaScript" などのストリング。

getServerModelFolderName()

対応バージョン
Dreamweaver MX

説明
この関数は、このサーバモデルに使用する、Configuration フォルダ内のフォルダ名を返します。dom.serverModel.getFolderName() 関数を呼び出して、JavaScript からこの値にアクセスできます。

引数
なし。

戻り値
"ASP_JS" などのストリング。

getServerSupportsCharset()

対応バージョン
Dreamweaver MX

説明
この関数は、指定された文字セットが現在のサーバでサポートされている場合は true を返します。JavaScript から、dom.serverModel.getServerSupportsCharset() 関数を呼び出して、サーバモデルが特定の文字セットをサポートしているかどうかを判別できます。

引数
metaCharSetString

metaCharSetString 引数は、ドキュメントの "charset=" 属性値を保持するストリングです。

戻り値
ブール値。

getVersionArray()

対応バージョン
Dreamweaver UltraDeveloper 1、Dreamweaver MX では非推奨

説明
この関数は、サーバテクノロジーからバージョン番号へのマッピングを取得します。この関数は、dom.serverModel.getServerVersion() 関数によって呼び出されます。

引数
なし。

戻り値
次の例で示すように、それぞれがバージョン名とバージョン値で構成されるバージョンオブジェクトの配列。

- ASP version 2.0
- ADODB version 2.1

第 22 章：データトランスレータ

データトランスレータは、特殊なマークアップを Adobe Dreamweaver で読み取りおよび表示可能なコードにトランスレートします。サーバサイドインクルード、条件付きの JavaScript ステートメント、またはその他のコード（PHP3、JSP、CFML、ASP など）は、特殊なマークアップの例です。Dreamweaver では、タグ内の属性、タグ全体、またはコードブロック全体をトランスレートできます。すべてのデータトランスレータ（ブロックやタグまたは属性）は、HTML ファイルです。

データトランスレーションには、JavaScript では実行できない複雑な操作、または C を使用した方が効率的に実行できる操作が含まれることがあります。これは特に、タグ全体またはコードブロックに当てはまります。C または C++ に精通している場合は、340 ページの「C レベル拡張機能」も参照してください。

次の表にデータトランスレータを作成するために使用するファイルの一覧を示します。

パス	ファイル	説明
Configuration¥ThirdPartyTags¥	**language**.xml	マークアップ言語のタグに関する情報が含まれています。
Configuration¥ThirdPartyTags¥	**language**.gif	言語のタグのアイコンです。
Configuration¥Translators¥	**language**.htm	データトランスレータの JavaScript 関数が含まれています。

トランスレータの動作

Dreamweaver は、タグ全体をトランスレートする場合でも属性だけをトランスレートする場合でも、すべてのトランスレータファイルを同様に処理します。Dreamweaver は起動時に Configuration¥Translators フォルダにあるファイルをすべて読み取り、getTranslatorInfo() 関数を呼び出してトランスレータについての情報を取得します。この際、getTranslatorInfo() 関数を含まないファイルや、エラーがあるためにこの関数が未定義と見なされるファイルは無視されます。

注意：JavaScript エラーが原因で起動時に障害が起こるのを防ぐため、トランスレータファイル内のエラーは、すべてのトランスレータが読み込まれた後で報告されます。トランスレータのデバッグについて詳しくは、328 ページの「トランスレータのバグの検索」を参照してください。

また、ユーザがトランスレートする必要がある新しいコンテンツを追加したり、既存のコンテンツを変更したりするたびに、「トランスレート」環境設定の指定に従って、Dreamweaver によって適用可能なすべてのトランスレータファイルの translateMarkup() 関数が呼び出されます。ユーザが次のアクションのいずれかを実行すると、Dreamweaver によって translateMarkup() 関数が呼び出されます。

- Dreamweaver でファイルを開いた場合。
- HTML パネルまたはコードビューで変更を行った後、デザインビューに切り替えた場合。
- 現在のドキュメントでオブジェクトのプロパティを変更した場合。
- 挿入バーまたは挿入メニューを使用して、オブジェクトを挿入した場合。
- 現在のドキュメントに別のアプリケーションで変更を加えた後、このドキュメントを更新した場合。
- ドキュメントにテンプレートを適用した場合。
- 他のウィンドウからドキュメントウィンドウに、またはドキュメントウィンドウ内で、コンテンツをペーストまたはドラッグした場合。

- 依存ファイルに変更を保存した場合。
- タグオブジェクトの innerHTML プロパティや outerHTML プロパティ、またはコメントオブジェクトの data プロパティを設定するコマンド、ビヘイビア、サーバビヘイビア、プロパティインスペクタ、または他の拡張機能ファイルを呼び出した場合。
- ファイル／変換／ 3.0 ブラウザ対応を選択した場合。
- 修正／変換／テーブルを絶対位置の DIV に変換を選択した場合。
- 修正／変換／絶対位置の DIV をテーブルに変換を選択します。
- クイックタグ編集でタグまたは属性を変更し、Tab キーまたは Enter キーを押した場合。

使用するトランスレータの種類の決定

すべてのトランスレータには、getTranslatorInfo() 関数と translateMarkup() 関数が必要です。またトランスレータは、Configuration¥Translators フォルダに格納されている必要があります。ただし、個々のトランスレータでは、以下に説明するように、ユーザのドキュメントに挿入するコードの種類およびコードの検査方法が異なります。

- 属性値を決定したり、特定条件に従って標準 HTML タグに属性を追加するサーバマークアップの一部をトランスレートするには、属性トランスレータを作成します。トランスレートされた属性を含む標準 HTML タグは、Dreamweaver に組み込まれているプロパティインスペクタで検査できます。カスタムプロパティインスペクタを作成する必要はありません（詳しくは、323 ページの「トランスレートされた属性のタグへの追加」を参照してください）。
- タグ全体（例えばサーバサイドインクルード（SSI）など）またはコードのブロック（例えば JavaScript、ColdFusion、PHP やその他のスクリプト）をトランスレートするには、ブロック / タグトランスレータを作成します。ブロック / タグトランスレータによって生成されたコードは、Dreamweaver に組み込まれているプロパティインスペクタでは検査できません。したがって、ユーザが元のコードのプロパティを変更できるようにするには、トランスレートされたコンテンツ用のカスタムプロパティインスペクタを作成する必要があります（詳しくは、325 ページの「トランスレートされたタグまたはコードブロックのロック」を参照してください）。

トランスレートされた属性のタグへの追加

属性のトランスレートでは、Dreamweaver パーサーに依存してサーバマークアップが無視されます。デフォルトの Dreamweaver では、大部分の一般的なサーバマークアップ（ASP、CFML、PHP など）は無視されます。開始マーカーと終了マーカーにこれらと異なるサーバマークアップを使用する場合は、トランスレータが正しく動作するようにサードパーティのタグデータベースを修正する必要があります。サードパーティのタグデータベースの修正について詳しくは、『Dreamweaver ユーザガイド』の「Dreamweaver のカスタマイズ」を参照してください。

Dreamweaver で元のサーバマークアップの保存が処理される際に、ドキュメントウィンドウで表示できる有効な属性値がトランスレータによって生成されます。したがって、ユーザに対する表示内容に影響のない属性のみにサーバマークアップを使用している場合、トランスレータは不要です。

トランスレータによって、mmTranslatedValue という特殊属性がサーバマークアップを含むタグに追加されて、ドキュメントウィンドウの表示に影響する属性値が生成されます。ただし、mmTranslatedValue 属性とその値は、HTML パネルやコードビューでは表示されず、ドキュメントにも保存されません。

mmTranslatedValue 属性はそのタグ内において固有である必要があります。トランスレータで 1 つのタグ内で複数の属性をトランスレートする可能性がある場合は、例えば mmTranslatedValue 属性に番号を追加するルーチンをトランスレータに追加する必要があります（例えば、mmTranslatedValue1 のように、mmTranslatedValue2 のようにします）。

mmTranslatedValue 属性の値は、有効な属性 / 値ペアを少なくとも 1 つ含む URL エンコードされたストリングである必要があります。つまり、mmTranslatedValue="src=%22open.jpg%22" は、src="<? if (dayType == weekday) then open.jpg else closed.jpg" ?> と <? if (dayType == weekday) then src="open.jpg" else src="closed.jpg" ?> の両方に有効なトランスレーションです。mmTranslatedValue="%22open.jpg%22" は、属性を含まず値だけを含むので、いずれの例でも有効ではありません。

複数の属性の同時トランスレート

mmTranslatedValue 属性には、複数の有効な属性 / 値ペアを含むことができます。例えば、次のコードをトランスレートすると仮定します。

```
<img <? if (dayType==weekday) then src="open.jpg" width="320" height="100" else
src="closed.jpg" width="100" height="320" ?> alt="We're open 24 hours a day from
12:01am Monday until 11:59pm Friday">
```

次の例では、トランスレートされたマークアップの表示方法を示します。

```
<img <? if (dayType==weekday) then src="open.jpg" width="320" height="100" else
src="closed.jpg" width="100" height="320" ?>
mmTranslatedValue="src=%22open.jpg%22 width=%22320%22 height=%22100%22"
alt="We're open 24 hours a day from 12:01am Monday until 11:59pm Friday">
```

ここでは mmTranslatedValue にある属性 / 値ペアの間のスペースがエンコードされていません。Dreamweaver では、トランスレートされた値をレンダリングする際にこれらのスペースが検索されるため、mmTranslatedValue 属性の各属性 / 値ペアを個別にエンコードし、再び組み合わせて完全な mmTranslatedValue 属性を生成する必要があります。このプロセスの例については、329 ページの「簡単な属性トランスレータの例」を参照してください。

トランスレートされた属性の検査

これに対してトランスレートされた属性はロックを必要としないため、これらの属性を含むタグは簡単に検査することができます。

サーバマークアップがプロパティインスペクタに表示される 1 つの属性を指定すると、その属性がプロパティインスペクタに表示されます。

マークアップは、これにトランスレータが関連付けられているかどうかにかかわらず表示されます。ユーザがプロパティインスペクタに表示されているサーバマークアップを編集するたびに、トランスレータが実行されます。

サーバマークアップで 1 つのタグにある複数の属性が制御される場合、サーバマークアップはプロパティインスペクタに表示されません。ただし、選択されたエレメントに対してトランスレートされたマークアップが存在することを示す稲妻アイコンが表示されます。

注意：稲妻アイコンは、テキストセル、表セル、行、列が選択されている場合は表示されません。ユーザがパネルでサーバマークアップを編集し、このタイプのマークアップを処理するトランスレータが存在する場合、トランスレートが続行されます。

プロパティインスペクタのテキストボックスは編集可能です。ユーザは、サーバマークアップが制御する属性の値を入力できます。これは、複製された属性になります。特定の属性にトランスレートされた値と正規の値の両方が設定されている場合、Dreamweaver はトランスレートされた値をドキュメントウィンドウに表示し、重複する属性をトランスレータで検索して削除するかどうかを決定します。

トランスレートされたタグまたはコードブロックのロック

Dreamweaver で表示できるようにトランスレータを使用してマークアップを変更しても、ほとんどの場合、保存するのは変更したものではなく、元のマークアップです。こうした場合のために、Dreamweaver には、トランスレートされたコンテンツを囲んだり、元のコードを参照したりするための特殊な XML タグが用意されています。

これらの XML タグを使用すると、元の属性のコンテンツがコードビューで複製されます。ファイルが保存されると、元のトランスレートされていないマークアップがファイルに書き込まれます。トランスレートされていないコンテンツは、コードビューに表示される内容です。

次の例では、XML タグのシンタックスを示します。

```
<MM:BeginLock translatorClass="translatorClass" ¬
type="tagNameOrType" depFiles="dependentFilesList" ¬
orig="encodedOriginalMarkup">
Translated content
<MM:EndLock>
```

この例の値の意味は以下のとおりです。

- **translatorClass** 値はトランスレータの固有識別子で、getTranslatorInfo() 関数によって返される配列の最初のストリングです。
- **tagNameOrType** 値は、ロックに含まれるマークアップのタイプ（またはマークアップに関連付けられているタグ名）を識別するストリングです。このストリングに使用できるのは英数字とハイフン（-）およびアンダースコア（_）だけです。コンテンツに対して適切なプロパティインスペクタであるかどうかを判断するには、カスタムプロパティインスペクタの canInspectSelection() 関数でこの値を確認します。詳しくは、326 ページの「ロックされたコンテンツ用プロパティインスペクタの作成」を参照してください。ロックされたコンテンツは、Dreamweaver に組み込まれたどのプロパティインスペクタでも検査できません。例えば、type="IMG" を指定してもイメージパネルは表示されません。
- **dependentFilesList** 値は、ロックされたマークアップが依存するファイルのリスト（カンマ区切り）を含むストリングです。ファイルは、ユーザのドキュメントを基準とする URL として参照されます。ユーザが **dependentFilesList** ストリングで指定されているファイルのいずれかを更新すると、Dreamweaver によって、そのリストを含むドキュメントのコンテンツが自動的にトランスレートし直されます。
- **encodedOriginalMarkup** 値は、URL エンコードの小型のサブセットを使用してエンコードされた、トランスレート前の元のマークアップを含むストリングです。例えば、" には %22、< には %3C、> には %3E、% には %25 がそれぞれ使用されます。ストリングを URL エンコードする最も簡単な方法は、escape() メソッドの使用です。例えば myString の値が '<img src="foo.gif">' である場合、escape(myString) は %3Cimg%20src=%22foo.gif%22%3E を返します。

次の例は、例えば <!--#include virtual="/footer.html" --> というサーバサイドインクルードのトランスレートによって生成されたコード中のロックされた部分を示します。

```
<MM:BeginLock translatorClass="MM_SSI" type="ssi" ¬
depFiles="C:\sites\webdev\footer.html" orig="%3C!--#include ¬
virtual=%22/footer.html%22%20--%3E">
<!-- begin footer -->
<CENTER>
<HR SIZE=1 NOSHADE WIDTH=100%>

<BR>

[<A TARGET="_top" HREF="/">home</A>]
[<A TARGET="_top" HREF="/products/">products</A>]
[<A TARGET="_top" HREF="/services/">services</A>]
[<A TARGET="_top" HREF="/support/">support</A>]
[<A TARGET="_top" HREF="/company/">about us</A>]
[<A TARGET="_top" HREF="/help/">help</A>]
</CENTER>
<!-- end footer -->
<MM:EndLock>
```

script タグ内のコードをロックするトランスレータを作成すると、トランスレータが失敗します。例えば、次のコードがあるとします。

```
<script language="javascript">
<!--
function foo() {
alert('<bean:message key="show.message"/>');
}
// -->
</script>
```

次に bean:messagestruts タグのトランスレータを作成すると、MM:BeginLock セクション内に MM:BeginLock セクションを作成しているため、トランスレータが失敗します。この問題を解決するには、bean:message タグ（<%= My_lookup.lookup("show.message") %> のような正規 JSP タグを使用する）の周囲に JSP ラッパーを作成します。これにより、トランスレータはこのコードを省略し、トランスレータが成功します。

ロックされたコンテンツ用プロパティインスペクタの作成

トランスレータを作成した後、ユーザがそのプロパティ（挿入するファイルや条件ステートメントに含まれる条件の 1 つなど）を変更できるように、このコンテンツ用のプロパティインスペクタを作成する必要があります。トランスレートされた内容の検査をする場合、次のような場合に独特の問題が発生します。

- ユーザがトランスレートされたコンテンツのプロパティを変更し、その変更内容をトランスレートされていないコンテンツに反映させる必要がある場合。
- ドキュメントオブジェクトモデル（DOM）には、トランスレートされたコンテンツが含まれている（つまり lock タグとこれに囲まれているタグが DOM のノードである）が、outerHTML プロパティ（documentElement オブジェクトの）と、dreamweaver.getSelection() 関数および dreamweaver.nodeToOffsets() 関数が、トランスレートされていないソースに対して動作する場合。
- 検査するタグがトランスレートの前と後で異なる場合。

  例えば、HAPPY タグ用のプロパティインスペクタには、次の例に似たコメントが含まれます。

  ```
  <!-- tag:HAPPY,priority:5,selection:exact,hline,vline, attrName:xxx,¬ attrValue:yyy -->
  ```

  この場合、トランスレートされた HAPPY タグ用のプロパティインスペクタには、次の例に似たコメントが含まれます。

  ```
  <!-- tag:*LOCKED*,priority:5,selection:within,hline,vline -->
  ```

また、トランスレートされていない HAPPY プロパティインスペクタの canInspectSelection() 関数は簡単です。selection タイプが exact なので、それ以上分析することなく true を返すことができます。トランスレートされた HAPPY プロパティインスペクタの場合、この関数は複雑です。*LOCKED* というキーワードは、選択範囲がロックされた領域内にある場合はこのプロパティインスペクタが適切であることを示しますが、ドキュメントにはロックされた領域が複数あることが可能であるため、検査を続行してプロパティインスペクタがこのロックされた領域に適しているかどうかを判断する必要があります。

トランスレートされたコンテンツを検査する場合、次の問題も発生します。dom.getSelection() 関数を呼び出すと、デフォルトではトランスレートされていないソースにおけるオフセットが返されます。ロックされた領域だけが正確に選択されるように選択範囲を拡張するには、次の手法を使用します。

```
var currentDOM = dw.getDocumentDOM();
var offsets = currentDOM.getSelection();
var theSelection = currentDOM.offsetsToNode(offsets[0],offsets[0]+1);
```

2 番目の引数に offsets[0]+1 を指定することにより、オフセットをノードに変換した場合に選択範囲を開始の lock タグの中にとどめることができます。2 番目の引数に offsets[1] を指定すると、ロックの上のノードを選択してしまう危険があります。

範囲を選択（選択範囲の nodeType が node.ELEMENT_NODE であることを確認）したら、次の例で示すように、type 属性を検査してロックされた領域がこのプロパティインスペクタに適しているかどうかを判断することができます。

```
if (theSelection.nodeType == node.ELEMENT_NODE && ¬
theSelection.getAttribute('type') == 'happy'){
    return true;
}else{
    return false
}
```

トランスレートされたタグ用のプロパティインスペクタのテキストボックスの値を設定するには、orig 属性の値を解析することが必要です。例えば、トランスレートされていないコードが <HAPPY TIME="22"> で、プロパティインスペクタに Time テキストボックスがある場合、TIME 属性の値を orig ストリングから抽出する必要があります。

```
function inspectSelection() {
    var currentDOM = dw.getDocumentDOM();
    var currSelection = currentDOM.getSelection();
    var theObj = currentDOM.offsetsToNode(curSelection[0],curSelection[0]+1);

    if (theObj.nodeType != Node.ELEMENT_NODE) {
        return;
    }

    // To convert the encoded characters back to their
    // original values, use the unescape() method.
    var origAtt = unescape(theObj.getAttribute("ORIG"));

    // Convert the string to lowercase for processing
    var origAttLC = origAtt.toLowerCase();

    var timeStart = origAttLC.indexOf('time="');
    var timeEnd = origAttLC.indexOf('"',timeStart+6);
    var timeValue = origAtt.substring(timeStart+6,timeEnd);

    document.layers['timelayer'].document.timeForm.timefield.value = timeValue;
}
```

トランスレートされたタグのプロパティインスペクタにあるボックスに値を取り込むために orig 属性を解析した後、ユーザがテキストボックスの値を変更する場合は orig 属性の値を設定します。ロックされた領域では、変更が制限される場合があります。変更が制限されないようにするには、元のマークアップを変更してから再度トランスレート処理を実行します。

この手法は、トランスレートされたサーバサイドインクルード用のプロパティインスペクタ（Configuration¥Inspectors フォルダ内の ssi_translated.js ファイル）の setComment() 関数で使用されています。このプロパティインスペクタは、orig 属性を書き直す代わりに新規のサーバサイドインクルードのコメントを作成します。その後、ドキュメントのコンテンツが書き直されて、古いコメントの代わりに新しいコメントがドキュメントに挿入され、これで新しい orig 属性が生成されます。次のコードはこの手法の概要を示すものです。

```
// Assemble the new include comment. radioStr and URL are
// variables defined earlier in the code.
newInc = "<!--#include " + radioStr + "=" + '"' + URL + '"' + " -->";
// Get the contents of the document.
var entireDocObj = dreamweaver.getDocumentDOM();
var docSrc = entireDocObj.documentElement.outerHTML;
// Store everything up to the SSI comment and everything after
// the SSI comment in the beforeSelStr and afterSelStr variables.
var beforeSelStr = docSrc.substring(0, curSelection[0] );
var afterSelStr= docSrc.substring(curSelection[1]);
// Assemble the new contents of the document.
docSrc = beforeSelStr + newInc + afterSelStr;
// Set the outerHTML of the HTML tag (represented by
// the documentElement object) to the new contents,
// and then set the selection back to the locked region
// surrounding the SSI comment.
entireDocObj.documentElement.outerHTML = docSrc;
entireDocObj.setSelection(curSelection[0], curSelection[0]+1);
```

トランスレータのバグの検索

translateMarkup() 関数に特定のタイプのエラーがある場合、トランスレータは正常に読み込まれますが、呼び出すとエラーメッセージを表示せずに失敗します。これにより Dreamweaver が不安定になることを防ぐことはできますが、特に複数行のコードで小さいシンタックスエラーを探す場合などには、開発作業の妨げとなります。

トランスレータが失敗した場合の効果的なデバッグ方法の 1 つとして、次の手順で説明するように、トランスレータのコマンドへの変換があります。

1. トランスレータファイルのコンテンツ全体を新しいドキュメントにコピーし、それを Dreamweaver アプリケーションフォルダ内の Configuration¥Commands フォルダに保存します。
2. ドキュメントの一番上にある SCRIPT タグの間に、次の関数を追加します。

```
function commandButtons(){
    return new Array( "OK","translateMarkup(dreamweaver.¬
    getDocumentPath('document'), dreamweaver.getSiteRoot(), ¬
    dreamweaver.getDocumentDOM().documentElement.outerHTML); ¬
    window.close()", "Cancel", "window.close()");
}
```

3. translateMarkup() 関数の最後で、return**whateverTheReturnValueIs** という行をコメントアウトし、それを dreamweaver.getDocumentDOM().documentElement.outerHTML = ***whateverTheReturnValueIs*** という行に置き換えます。以下に例を示します。

```
    // return theCode;
    dreamweaver.getDocumentDOM().documentElement.outerHTML = theCode;
}
/* end of translateMarkup() */
```

4. ドキュメントの BODY に、テキストボックスがない以下のフォームを追加します。

```
<body>
<form>
Hello.
</form>
</body>
```

5 Dreamweaver を再起動し、コマンドメニューからトランスレータコマンドを選択します。「OK」をクリックすると translateMarkup() 関数が呼び出され、トランスレート処理をシミュレートします。

エラーメッセージが表示されずにトランスレートが失敗した場合は、コードに論理エラーがあると考えられます。

6 この場合は、alert() ステートメントを translateMarkup() 関数全体の必要な場所に追加して、正しいブランチを取得していることを確認し、各要所で変数とプロパティの値をチェックできるようにします。

```
for (var i=0; i< foo.length; i++){
    alert("we're at the top of foo.length array, and the value of i is " + i);
    /* rest of loop */
}
```

7 alert() ステートメントを追加した後、コマンドメニューからコマンドを選択して「キャンセル」をクリックし、もう一度同じコマンドを選択します。この操作によってコマンドファイルがリロードされ、加えた変更が反映されます。

簡単な属性トランスレータの例

属性のトランスレートについての理解を深めるには、コード例が参考になります。次のトランスレータでは、ASP または PHP に類似したシンタックスである「Pound Conditional」（Poco）というマークアップが処理されます。

属性トランスレータを作成するには、tagspec タグ、アイコンおよび属性トランスレータを作成します。

tagspec タグを作成する

このトランスレータを適切に動作させるには、まず Poco マークアップの tagspec タグを作成します。これにより、まだトランスレートされていない Poco ステートメントが Dreamweaver によって解析されるのを防ぎます。

1 新しい空白のファイルを作成します。

2 次のコードを入力します。

```
<tagspec tag_name="poco" start_string="<#" end_string="#>" ¬
detect_in_attribute="true" icon="poco.gif" icon_width="17" ¬
icon_height="15"></tagspec>
```

3 ファイルを poco.xml として Configuration¥ThirdPartyTags フォルダに保存します。

tagspec タグの作成例については、Configuration¥ThirdPartyTags フォルダの Tags.xml ファイルを参照してください。

アイコンを作成する

次に、Poco タグのアイコンを作成します。

1 Poco タグのアイコンとして使用する 18 x 18 ピクセルのイメージファイルを作成します。

2 ファイルに poco.gif という名前を付けて Configuration¥ThirdPartyTags フォルダに保存します。

属性トランスレータを作成する

属性トランスレータに必要な関数が含まれる HTML ファイルを作成します。

1 新しい空白のファイルを作成します。

2 次のコードを入力します。

```
<html>
<head>
<title>Conditional Translator</title>
<meta http-equiv="Content-Type" content="text/html; charset=">
<script language="JavaScript">

/************************************************************
 * This translator handles the following statement syntaxes: *
 * <# if (condition) then foo else bar #>                    *
 * <# if (condition) then att="foo" else att="bar" #>        *
 * <# if (condition) then att1="foo" att2="jinkies"          *
 * att3="jeepers" else att1="bar" att2="zoinks" #>           *
 *                                                          *
 * It does not handle statements with no else clause.       *
 ************************************************************/

var count = 1;

function translateMarkup(docNameStr, siteRootStr, inStr){
var count = 1;
// Counter to ensure unique mmTranslatedValues
var outStr = inStr;
// String that will be manipulated
var spacer = "";
// String to manage space between encoded attributes
var start = inStr.indexOf('<# if');   // 1st instance of Pound Conditional code
// Declared but not initalized //
var attAndValue;
// Boolean indicating whether the attribute is part of
// the conditional statement
var trueStart;
// The beginning of the true case
var falseStart;
// The beginning of the false case
var trueValue;
// The HTML that would render in the true case
var attName;
// The name of the attribute that is being
// set conditionally.
var equalSign;
// The position of the equal sign just to the
// left of the <#, if there is one
var transAtt;
// The entire translated attribute
var transValue;
// The value that must be URL-encoded
var back3FromStart;
// Three characters back from the start position
// (used to find equal sign to the left of <#
var tokens;
// An array of all the attributes set in the true case
var end;
// The end of the current conditional statement.
// As long as there's still a <# conditional that hasn't been
// translated.
while (start != -1){
    back3FromStart = start-3;
    end = outStr.indexOf(' #>',start);
    equalSign = outStr.indexOf('="<# if',back3FromStart);
    attAndValue = (equalSign != -1)?false:true;
    trueStart = outStr.indexOf('then', start);
```

```
    falseStart = outStr.indexOf(' else', start);
    trueValue = outStr.substring(trueStart+5, falseStart);
    tokens = dreamweaver.getTokens(trueValue,' ');

    // If attAndValue is false, find out what attribute you're
    // translating by backing up from the equal sign to the
    // first space. The substring between the space and the
    // equal sign is the attribute.
    if (!attAndValue){
        for (var i=equalSign; i > 0; i--){
            if (outStr.charAt(i) == " "){
                attName = outStr.substring(i+1,equalSign);
                break;
            }
        }
        transValue = attName + '="' + trueValue + '"';
        transAtt = ' mmTranslatedValue' + count + '="' + ¬
        escape(transValue) + '"';
        outStr = outStr.substring(0,end+4) + transAtt + ¬
        outStr.substring(end+4);
    // If attAndValue is true, and tokens is greater than
    // 1, then trueValue is a series of attribute/value
    // pairs, not just one. In that case, each attribute/value
    // pair must be encoded separately and then added back
    // together to make the translated value.
    }else if (tokens.length > 1){
        transAtt = ' mmTranslatedValue' + count + '="'
        for (var j=0; j < tokens.length; j++){
            tokens[j] = escape(tokens[j]);
            if (j>0){
                spacer=" ";
            }
            transAtt += spacer + tokens[j];
        }
        transAtt += '"';
        outStr = outStr.substring(0,end+3) + transAtt + ¬
        outStr.substring(end+3)

    // If attAndValue is true and tokens is not greater
    // than 1, then trueValue is a single attribute/value pair.
    // This is the simplest case, where all that is necessary is
    // to encode trueValue.
    }else{
        transValue = trueValue;
        transAtt = ' mmTranslatedValue' + count + '="' + ¬
        escape(transValue) + '"';
        outStr = outStr.substring(0,end+3) + transAtt + ¬
        outStr.substring(end+3);
    }
    // Increment the counter so that the next instance
    // of mmTranslatedValue will have a unique name, and
    // then find the next <# conditional in the code.
    count++;
    start = outStr.indexOf('<# if',end);
}
// Return the translated string.
return outStr
}
```

```
function getTranslatorInfo(){
    returnArray = new Array(7);

    returnArray[0] = "Pound_Conditional";               // The translatorClass
    returnArray[1] = "Pound Conditional Translator";    // The title
    returnArray[2] = "2";                               // The number of extensions
    returnArray[3] = "html";                            // The first extension
    returnArray[4] = "htm";                             // The second extension
    returnArray[5] = "1";                               // The number of expressions
    returnArray[6] = "<#";                              // The first expression
    returnArray[7] = "byString";                        //
    returnArray[8] = "50";                                  //
    return returnArray
}
</script>
</head>

<body>
</body>
</html>
```

3 ファイルを Poco.htm として Configuration¥Translators フォルダに保存します。

ブロック / タグトランスレータの簡単な例

トランスレートに関する理解を深めるには、JavaScript だけで記述されたトランスレータのコード例を参照してください。このコードは、C ライブラリの機能に依存しません。次のトランスレータの例は、C で記述した方がより効率的ですが、JavaScript バージョンの方が簡潔なので、トランスレータの動作を示す例としては最適です。

大部分のトランスレータと同様、このトランスレータもサーバビヘイビアを模倣するように記述されています。曜日、時刻、およびユーザのプラットフォームに従って、KENT タグがそれぞれ異なるエンジニアの画像と置き換えられるように、使用する Web サーバが設定されていることを前提とします。すると、トランスレータはローカルマシン上で同じ操作を実行します。

ブロック / タグトランスレータを作成する

1 新しい空白のファイルを作成します。

2 次のコードを入力します。

```
<html>
<head>
<title>Kent Tag Translator</title>
<meta http-equiv="Content-Type" content="text/html; charset=">
<script language="JavaScript">
/*********************************************************
 * The getTranslatorInfo() function provides information  *
 * about the translator, including its class and name,    *
 * the types of documents that are likely to contain the  *
 * markup to be translated, the regular expressions that  *
 * a document containing the markup to be translated      *
 * would match (whether the translator should run on all  *
 * files, no files, in files with the specified           *
 * extensions, or in files matching the specified         *
 * expressions).                                          *
 *********************************************************/
function getTranslatorInfo(){
    //Create a new array with 6 slots in it
    returnArray = new Array(6);

    returnArray[0] = "DREAMWEAVER_TEAM";           // The translatorClass
    returnArray[1] = "Kent Tags";                  // The title
    returnArray[2] = "0";                          // The number of extensions
    returnArray[3] = "1";                          // The number of expressions
    returnArray[4] = "<kent";                      // Expression
    returnArray[5] = "byExpression";               // run if the file contains "<kent"
    return returnArray;
}

/*************************************************************************
 * The translateMarkup() function performs the actual translation.        *
 * In this translator, the translateMarkup() function is written          *
 * entirely in JavaScript (that is, it does not rely on a C library) --   *
 * and it's also extremely inefficient. It's a simple example, however,   *
 * which is good for learning.                                            *
*************************************************************************/
function translateMarkup(docNameStr, siteRootStr, inStr){
    var outStr = "";                      // The string to be returned after translation
    var start = inStr.indexOf('<kent>');  // The first position of the KENT tag
                                          // in the document.
    var replCode = replaceKentTag();      // Calls the replaceKentTag() function
                                          // to get the code that will replace KENT.
        var outStr = "";                  // The string to be returned after translation
     //If the document does not contain any content, terminate the translation.
    if ( inStr.length <= 0 ){
         return "";
    }

    // As long as start, which is equal to the location in inStr of the
    // KENT tag, is not equal to -1 (that is, as long as there is another
    // KENT tag in the document)
    while (start != -1){
         // Copy everything up to the start of the KENT tag.
         // This is very important, as translators should never change
         // anything other than the markup that is to be translated.
         outStr = inStr.substring(0, start);
         // Replace the KENT tag with the translated HTML, wrapped in special
         // locking tags. For more information on the repacement operation, see
         // the comments in the replaceKentTag() function.
         outStr = outStr + replCode;

         // Copy everything after the KENT tag.
```

```
        outStr = outStr + inStr.substring(start+6);

        // Use the string you just created for the next trip through
        // the document. This is the most inefficient part of all.
        inStr = outStr;
        start = inStr.indexOf('<kent>');

    }
    // When there are no more KENT tags in the document, return outStr.
    return outStr;
}

/*************************************************************
* The replaceKentTag() function assembles the HTML that will  *
* replace the KENT tag and the special locking tags that will *
* surround the HTML. It calls the getImage() function to      *
* determine the SRC of the IMG tag.                           *
*************************************************************/
function replaceKentTag(){
    // The image to display.
    var image = getImage();
    // The location of the image on the local disk.
    var depFiles = dreamweaver.getSiteRoot() + image;
    // The IMG tag that will be inserted between the lock tags.
    var imgTag = '<IMG SRC="/' + image + '" WIDTH="320" HEIGHT="240" ALT="Kent">\n';
    // 1st part of the opening lock tag. The remainder of the tag is assembled
    below.
    var start = '<MM:BeginLock translatorClass="DREAMWEAVER_TEAM" type="kent"';
    // The closing lock tag.
    var end = '<MM:EndLock>';

    //Assemble the lock tags and the replacement HTML.
    var replCode = start + ' depFiles="' + depFiles + '"';
    replCode = replCode + ' orig="%3Ckent%3E">\n';
    replCode = replCode + imgTag;
    replCode = replCode + end;

    return replCode;
}

/*****************************************************************
 * The getImage() function determines which image to display      *
 * based on the day of the week, the time of day and the          *
 * user's platform. The day and time are figured based on UTC     *
 * time (Greenwich Mean Time) minus 8 hours, which gives          *
 * Pacific Standard Time (PST). No allowance is made for Daylight *
 * Savings Time in this routine.                                  *
*****************************************************************/
function getImage(){
    var today = new Date();                  // Today's date & time.
    var day = today.getUTCDay();             // Day of the week in the GMT time zone.
                                             // 0=Sunday, 1=Monday, and so on.
        var hour = today.getUTCHours();      // The current hour in GMT, based on the
                                             // 24-hour clock.
    var SFhour = hour - 8;                   // The time in San Francisco, based on the
                                             // 24-hour clock.
    var platform = navigator.platform;       // User's platform. All Windows computers
                                             // are identified by Dreamweaver as "Win32",
                                             // all Macs as "MacPPC".
        var imageRef;                        // The image reference to be returned.
            // If SFhour is negative, you have two adjustments to make.
            // First, subtract one from the day count because it is already the wee
```

```
            // hours of the next day in GMT. Second, add SFhour to 24 to
            // give a valid hour in the 24-hour clock.
        if (SFhour < 0){
            day = day - 1;
                // The day count back one would make it negative, and it's Saturday,
                // so set the count to 6.
                if (day < 0){
                        day = 6;
                }
            SFhour = SFhour + 24;
        }

        // Now determine which photo to show based on whether it's a work day or a
        // weekend; what time it is; and, if it's a time and day when Kent is
        // working, what platform the user is on.

        //If it's not Sunday
        if (day != 0){
             //And it's between 10am and noon, inclusive
             if (SFhour >= 10 && SFhour <= 12){
                    imageRef = "images/kent_tiredAndIrritated.jpg";
                //Or else it's between 1pm and 3pm, inclusive
             }else if (SFhour >= 13 && SFhour <= 15){
                    imageRef ="images/kent_hungry.jpg";
             //Or else it's between 4pm and 5pm, inclusive
             }else if (SFhour >= 16 && SFhour <= 17){
                    //If user is on Mac, show Kent working on Mac
                    if (platform == "MacPPC"){
                         imageRef = "images/kent_gettingStartedOnMac.jpg";
                    //If user is on Win, show Kent working on Win
                    }else{
                         imageRef = "images/kent_gettingStartedOnWin.jpg";
                    }
             //Or else it's after 6pm but before the stroke of midnight
             }else if (SFhour >= 18){
                     //If it's Saturday
                     if (day == 6){
                         imageRef = "images/kent_dancing.jpg";
                    //If it's not Saturday, check the user's platform
                    }else if (platform == "MacPPC"){
                         imageRef = "images/kent_hardAtWorkOnMac.jpg";
                    }else{
                         imageRef = "images/kent_hardAtWorkOnWin.jpg";
                    }
             }else{
                    imageRef = "images/kent_sleeping.jpg";
             }
        //If it's after midnight and before 10am, or anytime on Sunday
        }else{
             imageRef = "images/kent_sleeping.jpg";
        }

        return imageRef;
    }

    </script>
    </head>

    <body>
    </body>
    </html>
```

3 ファイルを kent.htm として Configuration¥Translators フォルダに保存します。

データトランスレータ API 関数

この節では、Dreamweaver でトランスレータを定義する際に使用する関数について説明します。

getTranslatorInfo()

説明
この関数は、トランスレータとそれが操作できるファイルに関する情報を提供します。

引数
なし。

戻り値
ストリングの配列。配列のエレメントは次の順序になっている必要があります。

1 **translatorClass** ストリングはトランスレータを一意に識別するものです。このストリングは文字で始まる必要があり、使用できるのは英数字とハイフン（-）およびアンダースコア（_）だけです。

2 **title** ストリングは、トランスレータについての 40 文字以内の説明です。

3 **nExtensions** ストリングは、これに続くファイル拡張子の数を指定します。**nExtensions** がゼロの場合、すべてのファイルでトランスレータを実行できます。**nExtensions** がゼロの場合、配列内の次のエレメントは **nRegExps** になります。

4 **extension** ストリングでは、このトランスレータで操作できるファイルの拡張子（"htm" や "SHTML" など）を指定します。このストリングでは、大文字と小文字が区別されません。また、先頭にピリオドを付けることはできません。配列には、**extension** エレメントが、**nExtensions** で指定された個数含まれている必要があります。

5 **nRegExps** ストリングでは、これに続く正規表現の数が指定されます。**nRegExps** がゼロの場合、配列内の次のエレメントは **runDefault** になります。

6 **regExps** ストリングでは、チェックできる正規表現を指定します。配列には、**regExps** エレメントが、**nRegExps** で指定されているエレメントの数と同数含まれている必要があります。また、トランスレータがファイルに対して操作を開始する前に、少なくとも 1 つの **regExps** エレメントがドキュメントのソースコードの一部と一致する必要があります。

7 **runDefault** ストリングでは、このトランスレータを実行するタイミングを指定します。以下に、使用可能なストリング値を示します。

ストリング	定義
"allFiles"	トランスレータが常に実行されるように設定します。
"noFiles"	トランスレータが実行されないように設定します。
"byExtension"	拡張機能ファイルで指定されているいずれかのファイル拡張子のファイルについて、トランスレータが実行されるように設定します。
"byExpression"	指定された正規表現の 1 つに一致する表現がドキュメントに含まれる場合に、トランスレータが実行されるように設定します。
"bystring"	指定されたストリングの 1 つに一致するものがドキュメントに含まれる場合に、トランスレータが実行されるように設定します。

注意：runDefault を "byExtension" に設定して、拡張子を指定しないと（詳しくは、手順 4 を参照してください）、"allFiles" に設定した場合と同じ結果になります。runDefault を "byExpression" に設定して、式を指定しないと（詳しくは、手順 6 を参照してください）、"noFiles" に設定した場合と同じ結果になります。

8 **priority** ストリングは、このトランスレータを実行するためのデフォルトの優先度を指定します。優先度には、0 ～ 100 の数値を指定します。優先度を指定しないと、デフォルトの優先度は 100 になります。優先度は 0 が最も高く、100 が最も低くなります。複数のトランスレータをドキュメントに適用する場合は、この設定によってトランスレータを適用する順序を制御します。最も高い優先度から適用されます。同じ優先度に指定されているトランスレータが複数ある場合は、translatorClass のアルファベット順に適用されます。

例

次の getTranslatorInfo() 関数の例では、サーバサイドインクルード用のトランスレータに関する情報を提供します。

```
function getTranslatorInfo(){
    var transArray = new Array(11);
    transArray[0] = "SSI";
    transArray[1] = "Server-Side Includes";
    transArray[2] = "4";
    transArray[3] = "htm";
    transArray[4] = "stm";
    transArray[5] = "html";
    transArray[6] = "shtml";
    transArray[7] = "2";
    transArray[8] = "<!--#include file";
    transArray[9] = "<!--#include virtual";
    transArray[10] = "byExtension";
    transArray[11] = "50";
    return transArray;
}
```

translateDOM()

対応バージョン

Dreamweaver CS3

説明

Dreamweaver は 2 つのトランスレート処理を実行します。最初の処理によりすべてのトランスレータで translateMarkup() 関数が呼び出されます。これらの関数が呼び出されると、2 番目の処理により translateDOM() 関数が呼び出されます。渡された dom が変換する dom です。2 番目の処理中は、トランスレートされた属性を処理する編集のみが許可されます。

引数

dom、**sourceStr**

- **dom** 引数。
- **sourceStr** 引数には、translateMarkup に渡されるストリングと同じストリングを指定します。これはリファレンス用に提供されていますが、すべてのトランスレートは **dom** 引数（**sourceStr** 引数ではなく）で行う必要があります。

戻り値

なし。

例

```
translateDOM( dom, sourceStr );        // 戻り値なし
```

次の translateDOM() 関数のインスタンスにより、ドキュメント内で ID div1 を伴うタグが隠されます。

```
function translateDOM(dom, sourceStr){
    var div1 = dom.getAttributeById("div1");
    if (div1){
    div1.style.display = "none";
    }
}
```

translateMarkup()

説明
この関数は、トランスレートを実行します。

引数
docName、**siteRoot**、**docContent**

- **docName** 引数には、トランスレートするドキュメントの file:// URL を含むストリングを指定します。
- **siteRoot** 引数には、トランスレートするドキュメントがあるサイトのルートに対する、file:// URL を含むストリングを指定します。ドキュメントがサイト外にある場合、このストリングは空白になります。
- **docContent** 引数には、ドキュメントのコンテンツを含むストリングを指定します。

戻り値
トランスレートされたドキュメントを含むストリング。トランスレートされたものがない場合は、空白のストリングが返されます。

例
次の translateMarkup() 関数の例では、translateASP() という C 関数を呼び出します。この関数は、ASPTrans と呼ばれるダイナミックリンクライブラリ（DLL）（Windows）またはコードライブラリ（Macintosh）に含まれています。

```
function translateMarkup(docName, siteRoot, docContent){
    var translatedString = "";
    if (docContent.length > 0){
    translatedString = ASPTrans.translateASP(docName, siteRoot, ¬
    docContent);
    }
    return translatedString;
}
```

JavaScript のすべての例については、329 ページの「簡単な属性トランスレータの例」または 332 ページの「ブロック / タグトランスレータの簡単な例」を参照してください。

liveDataTranslateMarkup()

対応バージョン
Dreamweaver UltraDeveloper 1

説明
この関数は、ユーザがライブデータウィンドウを使用しているときにドキュメントをトランスレートします。ユーザが表示／ライブデータのコマンドを選択するか、「更新」ボタンをクリックすると、liveDataTranslateMarkup() 関数が、translateMarkup() 関数の代わりに呼び出されます。

引数

docName、**siteRoot**、**docContent**

- **docName** 引数には、トランスレートするドキュメントの file:// URL を含むストリングを指定します。
- **siteRoot** 引数には、トランスレートするドキュメントがあるサイトのルートに対する、file:// URL を含むストリングを指定します。ドキュメントがサイト外にある場合、このストリングは空白になります。
- **docContent** 引数には、ドキュメントのコンテンツを含むストリングを指定します。

戻り値

トランスレートされたドキュメントを含むストリング。トランスレートされたものがない場合は、空白のストリングが返されます。

例

次の liveDataTranslateMarkup() 関数の例では、translateASP() という C 関数を呼び出します。この関数は、ASPTrans と呼ばれる DLL（Windows）またはコードライブラリ（Macintosh）に含まれています。

```
function liveDataTranslateMarkup(docName, siteRoot, docContent){
    var translatedString = "";
    if (docContent.length > 0){
    translatedString = ASPTrans.translateASP(docName, siteRoot, docContent);
    }
    return translatedString;
}
```

第 23 章：C レベル拡張機能

C レベル拡張機能は、JavaScript とカスタム C コードを組み合わせて Adobe Dreamweaver の拡張機能ファイルを実装可能にする機構です。これを行うには、まず、C を使用して関数を定義し、ダイナミックリンクライブラリ（DLL）または共有ライブラリにバンドルして、そのライブラリを Dreamweaver アプリケーションフォルダ内の Configuration¥JSExtensions フォルダに保存します。次に、Dreamweaver JavaScript インタープリタを使用して、JavaScript からこれらの関数を呼び出します。

例えば、ユーザが指定したファイルの内容を現在のドキュメントに挿入する Dreamweaver オブジェクトを定義するとします。クライアント側の JavaScript はファイルの入出力（I/O）をサポートしないため、この機能を提供する C 関数の記述が必要です。

統合 C 関数 のしくみ

以下の HTML と JavaScript を使用して、簡単な Insert Text from File（ファイルからのテキスト挿入）オブジェクトを作成することができます。この例では、objectTag() 関数が readContentsOfFile() C 関数（myLibrary と呼ばれるライブラリに保存）を呼び出します。

```
<HTML>
<HEAD>
<SCRIPT>
function objectTag() {
    fileName = document.forms[0].myFile.value;
    return myLibrary.readContentsOfFile(fileName);
}
</SCRIPT>
</HEAD>

<BODY>
<FORM>
Enter the name of the file to be inserted:
<INPUT TYPE="file" NAME="myFile">
</FORM>
</BODY>
</HTML>
```

readContentsOfFile() 関数は、ユーザから引数のリストを受け取り、ファイル名の引数を取得します。次に、ファイルの内容を読み取り、それを返します。JavaScript のデータ構造および readContentsOfFile() 関数内に組み込まれている関数について詳しくは、342 ページの「C レベル拡張機能と JavaScript インタープリタ」を参照してください。

```
JSBool
readContentsOfFile(JSContext *cx, JSObject *obj, unsigned int ¬
argc, jsval *argv, jsval *rval)
{
    char *fileName, *fileContents;
    JSBool success;
    unsigned int length;

    /* Make sure caller passed in exactly one argument. If not,
     * then tell the interpreter to abort script execution.*/
    if (argc != 1){
        JS_ReportError(cx, "Wrong number of arguments", 0);
    return JS_FALSE;
    }

    /* Convert the argument to a string */
    fileName = JS_ValueToString(cx, argv[0], &length);
    if (fileName == NULL){
        JS_ReportError(cx, "The argument must be a string", 0);
        return JS_FALSE;
    }

    /* Use the string (the file name) to open and read a file */
    fileContents = exerciseLeftToTheReader(fileName);

    /* Store file contents in rval, which is the return value passed
    * back to the caller */
    success = JS_StringToValue(cx, fileContents, 0, *rval);
    free(fileContents);

    /* Return true to continue or false to abort the script */
    return success;
}
```

JavaScript エラーを発生させずに readContentsOfFile() 関数を正常に実行するには、使用するライブラリに MM_Init() 関数をインクルードして、JavaScript インタープリタに readContentsOfFile() 関数を登録する必要があります。起動時に Dreamweaver によってライブラリが読み込まれる際に、MM_Init() 関数が呼び出され、以下の 3 種類の情報が取得されます。

- 関数の JavaScript 名
- 関数へのポインタ
- 関数が受け入れる引数の数

MM_Init() 関数（ライブラリ myLibrary）の例を以下に示します。

```
void
MM_Init()
{
    JS_DefineFunction("readContentsOfFile", readContentsOfFile, 1);
}
```

使用するライブラリには、以下のマクロのうちいずれか 1 つのインスタンスが含まれている必要があります。

```
/* MM_STATE is a macro that expands to some definitions that are
 * needed to interact with Dreamweaver. This macro must
 * be defined exactly once in your library. */
MM_STATE
```

注意：ライブラリは、C または C++ で実装できますが、MM_Init() 関数と MM_STATE マクロを含むファイルは C で実装する必要があります。C++ コンパイラを使用すると、関数名が文字化けするので、Dreamweaver で MM_Init() 関数を検索できなくなります。

C レベル拡張機能と JavaScript インタープリタ

使用するライブラリの C コードは、以下のように数回にわたって Dreamweaver の JavaScript インタープリタと対話する必要があります。

- 起動時（ライブラリの関数を登録します）
- 関数の呼び出し時（JavaScript が C に渡す引数を解析します）
- 関数によって戻り値が返される前（戻り値をパッケージ化します）

これらのタスクを実行するため、インタープリタが数種類のデータタイプを定義して API を表示します。ここで説明するデータタイプと関数の定義は、mm_jsapi.h ファイルに保存されています。ライブラリが正常に機能するには、ライブラリの各ファイルの先頭で以下の行を指定して、mm_jsapi.h ファイルをインクルードする必要があります。

```
#include "mm_jsapi.h"
```

mm_jsapi.h ファイルをインクルードすると、MM_Environment 構造を定義する mm_jsapi_environment.h がインクルードされます。

データタイプ

JavaScript インタープリタでは次のデータタイプを定義できます。

typedef struct JSContext JSContext
この不透明データタイプへのポインタが C レベル関数に渡されます。API に含まれる一部の関数では、引数の 1 つとしてこのポインタが受け入れられます。

typedef struct JSObject JSObject
この不透明データタイプへのポインタが C レベル関数に渡されます。このデータタイプはオブジェクト（配列オブジェクトやその他のオブジェクトタイプ）を表します。

typedef struct JSVal JSVal
整数または、浮動小数点値、ストリング、オブジェクトへのポインタを含むことのできる不透明データ構造。API の一部の関数は、JSVal 構造体の内容を読み取ることで関数の引数の値を読み取ることができます。また、一部の関数を使用し、JSVal 構造体を記述することで、関数の戻り値を記述することができます。

typedef enum { JS_FALSE = 0, JS_TRUE = 1 } JSBool
ブール値を保存する簡単なデータタイプです。

C レベル API

C レベル拡張機能 API は次の関数で構成されています。

typedef JSBool (*JSNative)(JSContext *cx, JSObject *obj, unsigned int argc, jsval *argv, jsval *rval)

説明

この関数シグネチャで、次の状況での JavaScript 関数の C レベル実装を記述します。

- **cx** ポインタは、不透明 JSContext 構造へのポインタです。JavaScript API の一部の関数に渡す必要があります。この変数はインタプリタの実行コンテキストを保持します。
- **obj** ポインタは、スクリプトが実行されるコンテキストを持つオブジェクトへのポインタです。スクリプトの実行中は、this キーワードがこのオブジェクトと同意になります。
- **argc** 整数は、関数に渡される引数の個数です。
- **argv** ポインタは、JSVal 構造の配列へのポインタです。配列の長さは、エレメント argc 個です。
- **rval** ポインタは、単一の JSVal 構造へのポインタです。関数の戻り値を *rval に書き込む必要があります。

この関数が成功すると JS_TRUE が返され、失敗すると JS_FALSE が返されます。JS_FALSE が返された場合、現在のスクリプトの実行が中止され、エラーメッセージが表示されます。

JSBool JS_DefineFunction()

説明

この関数は、Dreamweaver で JavaScript インタープリタに C レベル関数を登録します。JS_DefineFunction() 関数で、**call** 引数に指定した C レベル関数を登録すると、それを **name** 引数に指定した名前で参照して、JavaScript スクリプトから呼び出すことができます。**name** 引数では、大文字と小文字が区別されます。

一般的に、この関数は、Dreamweaver が起動中に呼び出す MM_Init() 関数から呼び出されます。

引数

char ***name***、JSNative**call**、unsigned int nargs

- **name** 引数は、JavaScript に渡される関数名です。
- **call** 引数は C レベル関数へのポインタです。この関数では、readContentsOfFile と同じ引数を受け入れるだけでなく、成功または失敗を示す JSBool を返す必要があります。
- **nargs** 引数はこの関数が受け入れる引数の個数です。

戻り値

ブール値。成功した場合は JS_TRUE、失敗した場合は JS_FALSE が返されます。

char *JS_ValueToString()

説明

この関数は、JSVal 構造から関数の引数を抽出し、可能であればストリングに変換して、それを呼び出し元に渡します。

注意：返されたバッファポインタを修正しないでください。修正すると、JavaScript インタープリタのデータ構造が破損することがあります。ストリングを変更するには、文字を別のバッファにコピーし、JavaScript ストリングを作成します。

引数
JSContext *cx、JSVal v、符号なし整数 *pLength

- ***cx*** 引数は、JavaScript 関数に渡される不透明 JSContext ポインタです。
- ***v*** 引数は、Dreamweaver がストリングを抽出する JSVal 構造です。
- ***pLength*** 引数は符号なし整数へのポインタです。この関数は、*plength をストリングの長さと同じ長さにバイト単位で設定します。

戻り値
成功した場合はヌル終了されたストリングを参照するポインタが返され、失敗した場合は null 値を参照するポインタが返されます。終了時に呼び出し元ルーチンがこのストリングを解放しないようにしてください。

JSBool JS_ValueToInteger()

説明
この関数は、JSVal 構造から関数の引数を抽出し、可能であれば整数に変換して、それを呼び出し元に渡します。

引数
JSContext ***cx***、JSVal ***v***、long ***lp***

- **cx** 引数は、JavaScript 関数に渡される不透明 JSContext ポインタです。
- **v** 引数は、整数の抽出元の JSVal 構造です。
- **lp** 引数は、4 バイトの整数へのポインタです。この関数では、変換値が ***lp** に保存されます。

戻り値
ブール値。成功した場合は JS_TRUE、失敗した場合は JS_FALSE が返されます。

JSBool JS_ValueToDouble()

説明
この関数は、JSVal 構造から関数の引数を抽出し、可能であれば double に変換して、それを呼び出し元に渡します。

引数
JSContext ***cx***、JSVal ***v***、double *dp

- **cx** 引数は、JavaScript 関数に渡される不透明 JSContext ポインタです。
- **v** 引数は、double の抽出元である JSVal 構造です。
- **dp** 引数は 8 バイト長の double へのポインタです。この関数では、変換値が ***dp** に保存されます。

戻り値
ブール値。成功した場合は JS_TRUE、失敗した場合は JS_FALSE が返されます。

JSBool JS_ValueToBoolean()

説明
この関数は、JSVal 構造から関数の引数を抽出し、可能であればブール値に変換して、それを呼び出し元に渡します。

引数
JSContext ***cx***、JSVal ***v***、JSBool *bp

- **cx** 引数は、JavaScript 関数に渡される不透明 JSContext ポインタです。
- **v** 引数は、ブール値の抽出元の JSVal 構造です。
- **bp** 引数は、JSBool ブール値へのポインタです。この関数は、変換値を ***bp** に保存します。

戻り値
ブール値。成功した場合は JS_TRUE、失敗した場合は JS_FALSE が返されます。

JSBool JS_ValueToObject()

説明
この関数は、JSVal 構造から関数の引数を抽出し、可能であればオブジェクトに変換して、それを呼び出し元に渡します。オブジェクトが配列である場合、JS_GetArrayLength() と JS_GetElement() を使用してその内容を読み取ります。

引数
JSContext ***cx***、JSVal ***v***、JSObject **op

- **cx** 引数は、JavaScript 関数に渡される不透明 JSContext ポインタです。
- **v** 引数は、オブジェクトの抽出元である JSVal 構造です。
- **op** 引数は、JSObject ポインタへのポインタです。この関数では、変換値が ***op** に保存されます。

戻り値
ブール値。成功した場合は JS_TRUE、失敗した場合は JS_FALSE が返されます。

JS_ValueToUCString()

説明
この関数は、JSVal 構造から関数の引数を抽出し、可能であればストリングに変換して、それを呼び出し元に渡します。

注意：返されたバッファポインタを修正しないでください。修正すると、JavaScript インタープリタのデータ構造が破損することがあります。ストリングを変更するには、文字を別のバッファにコピーし、JavaScript ストリングを作成します。

引数
JSContext *cx、JSVal v、符号なし整数 *pLength

- ***cx*** 引数は、JavaScript 関数に渡される不透明 JSContext ポインタです。
- ***v*** 引数は、Dreamweaver がストリングを抽出する JSVal 構造です。
- ***pLength*** 引数は符号なし整数へのポインタです。この関数は、*plength をストリングの長さと同じ長さにバイト単位で設定します。

戻り値
成功した場合はヌル終了された UTF-8 ストリングを参照するポインタが返され、失敗した場合は null 値を参照するポインタが返されます。終了時に呼び出し元ルーチンがこのストリングを解放しないようにしてください。

JSBool JS_StringToValue()

説明
この関数は、ストリングの戻り値を JSVal 構造に保存します。これにより、新しい JavaScript ストリングオブジェクトが割り当てられます。

引数
JSContext *cx、JSVal *bytes、**size_t sz**、JSVal*vp

- ***cx*** 引数は、JavaScript 関数に渡される不透明 JSContext ポインタです。
- ***bytes*** 引数は、JSVal 構造に格納されるストリングです。ストリングはコピーされるので、呼び出し元はストリングが不要になった時点で、これを解放する必要があります。ストリングサイズが指定されていない場合（***sz*** 引数を参照）、ストリングをヌル終了する必要があります。
- ***sz*** 引数はストリングサイズ（バイト数）です。***sz*** が 0 の場合、ヌル終了されたストリングの長さが自動的に計算されます。
- *vp 引数は、ストリングの内容のコピー先の JSVal 構造へのポインタです。

戻り値
ブール値。成功した場合は JS_TRUE、失敗した場合は JS_FALSE が返されます。

JSBool JS_UCStringToValue()

説明
この関数は、ストリングの戻り値を JSVal 構造に保存します。これにより、新しい JavaScript ストリングオブジェクトが割り当てられます。

引数
JSContext ***cx***、JSVal *bytes、size_t***sz**、JSVal ****vp***

- ***cx*** 引数は、JavaScript 関数に渡される不透明 JSContext ポインタです。
- ***bytes*** 引数は、JSVal 構造に格納されるストリングです。ストリングはコピーされるので、呼び出し元はストリングが不要になった時点で、これを解放する必要があります。ストリングサイズが指定されていない場合（***sz*** 引数を参照）、ストリングをヌル終了する必要があります。
- ***sz*** 引数はストリングサイズ（バイト数）です。***sz*** が 0 の場合、ヌル終了されたストリングの長さが自動的に計算されます。
- *vp 引数は、ストリングの内容のコピー先の JSVal 構造へのポインタです。

戻り値
ブール値。成功した場合は JS_TRUE、失敗した場合は JS_FALSE が返されます。

注意：JS_UCStringToValue() メソッドは、UTF-8 ストリングを返すことを除き、すべての点で JSBool JS_StringToValue() と似ています。

JSBool JS_DoubleToValue()

説明
この関数は、浮動小数点値の戻り値を JSVal 構造に保存します。

引数
JSContext ***cx***、double ***dv***、JSVal ***vp***

- **cx** 引数は、JavaScript 関数に渡される不透明 JSContext ポインタです。
- **dv** 引数は 8 バイト長の浮動小数点値です。
- **vp** 引数は、倍精度数の内容のコピー先の JSVal 構造へのポインタです。

戻り値
ブール値。成功した場合は JS_TRUE、失敗した場合は JS_FALSE が返されます。

JSVal JS_BooleanToValue()

説明
この関数は、ブール値の戻り値を JSVal 構造に保存します。

引数
JSBool bv

- **bv** 引数はブール値です。成功した場合は JS_TRUE、失敗した場合は JS_FALSE が返されます。

戻り値
引数として関数に渡すブール値を含む JSVal 構造。

JSVal JS_IntegerToValue()

説明
この関数は、長整数値を JSVal 構造に変換します。

引数
lv

lv 引数は、JSVal 構造に変換する長整数値です。

戻り値
引数として関数に渡した整数を含む JSVal 構造。

JSVal JS_ObjectToValue()

説明
この関数は、オブジェクトの戻り値を JSVal に保存します。配列オブジェクトを作成するには JS_ NewArrayObject() を使用し、内容を定義するには JS_SetElement() を使用します。

引数
JSObject ***obj***

obj 引数は JSObject オブジェクトへのポインタで、オブジェクトを JSVal 構造に変換します。

戻り値
引数として関数に渡したオブジェクトを含む JSVal 構造。

char *JS_ObjectType()

説明
オブジェクトリファレンスを指定すると、JS_ObjectType() 関数によってそのオブジェクトのクラス名が返されます。例えば、オブジェクトが DOM オブジェクトの場合、この関数は「Document」を返します。オブジェクトがドキュメントのノードの場合は、「Element」を返します。配列オブジェクトの場合、関数は「Array」を返します。

注意：返されたバッファポインタを修正しないでください。修正すると、JavaScript インタープリタのデータ構造が破損することがあります。

引数
JSObject *obj

通常、この引数の受け渡しと変換には、JS_ValueToObject() 関数が使用されます。

戻り値
ヌル終了されたストリングへのポインタ。終了時に呼び出し元がこのストリングを解放しないようにしてください。

JSObject *JS_NewArrayObject()

説明
この関数は、JSVals の配列を含む新しいオブジェクトを作成します。

引数
JSContext ***cx***、unsigned int ***length***、JSVal *v

- **cx** 引数は、JavaScript 関数に渡される不透明 JSContext ポインタです。
- **length** 引数は、配列に含めることができる要素の数です。
- **v** 引数は、配列に格納される JSVals への省略可能なポインタです。戻り値が null でないときは、**v** が **length** で指定された数の要素を含む配列となります。戻り値が null の場合、配列オブジェクトの初期の内容は定義されていません。これは、JS_SetElement() 関数を使用して設定することができます。

戻り値
新規配列オブジェクトへのポインタ。または、失敗した場合は null 値。

long JS_GetArrayLength()

説明
この関数は、配列オブジェクトへのポインタを受け取り、この配列の要素数を取得します。

引数

JSContext ***cx***、JSObject***obj***

- **cx** 引数は、JavaScript 関数に渡される不透明 JSContext ポインタです。
- **obj** 引数は配列オブジェクトへのポインタです。

戻り値

配列内のエレメントの数を返します。失敗した場合は -1 を返します。

JSBool JS_GetElement()

説明

この関数は、配列オブジェクトの単一要素を読み取ります。

引数

JSContext ***cx***、JSObject ***obj***、unsigned int ***index***、JSVal *v

- **cx** 引数は、JavaScript 関数に渡される不透明 JSContext ポインタです。
- **obj** 引数は配列オブジェクトへのポインタです。
- **index** 引数は配列の整数インデックスです。最初の要素の index は 0 で、最後の要素の index は（length - 1）です。
- **v** 引数は jsval へのポインタで、配列に含まれる JSVal 構造の内容のコピー先を表します。

戻り値

ブール値。成功した場合は JS_TRUE、失敗した場合は JS_FALSE が返されます。

JSBool JS_SetElement()

説明

この関数は、配列オブジェクトの単一要素を書き込みます。

引数

JSContext *cx、JSObject *obj、unsigned int index、JSVal *v

- **cx** 引数は、JavaScript 関数に渡される不透明 JSContext ポインタです。
- **obj** 引数は配列オブジェクトへのポインタです。
- **index** 引数は配列の整数インデックスです。最初の要素の index は 0 で、最後の要素の index は（length - 1）です。
- **v** 引数は、配列に含まれる JSVal にコピーする内容が保存されている JSVal 構造へのポインタです。

戻り値

ブール値。成功した場合は JS_TRUE、失敗した場合は JS_FALSE が返されます。

JSBool JS_ExecuteScript()

説明
この関数は、JavaScript ストリングをコンパイルして実行します。スクリプトによって戻り値が生成されると、その値が ***rval** に返されます。

引数
JSContext ***cx***、JSObject ****obj***、**char *script**、**unsigned int*sz***、JSVal *rval

- **cx** 引数は、JavaScript 関数に渡される不透明 JSContext ポインタです。
- **obj** 引数は、スクリプトが実行されるコンテキストを持つオブジェクトへのポインタです。スクリプトの実行中は、this キーワードがこのオブジェクトと同意になります。これは通常、JavaScript 関数に渡される JSObject ポインタです。
- **script** 引数は、JavaScript コードを含むストリングです。ストリングのサイズが指定されていない場合（詳しくは、**sz** 引数を参照してください）、ストリングはヌル終了されている必要があります。
- **sz** 引数はストリングサイズ（バイト数）です。**sz** が 0 の場合、ヌル終了されたストリングの長さは、自動的に計算されます。
- **rval** 引数は、単一の JSVal 構造へのポインタです。関数の戻り値を ***rval** に格納します。

戻り値
ブール値。成功した場合は JS_TRUE、失敗した場合は JS_FALSE が返されます。

JSBool JS_ReportError()

説明
この関数は、スクリプトエラーとなった理由を説明します。スクリプトエラーについて値 JS_FALSE を返す前にこの関数を呼び出し、スクリプトが失敗した理由についてユーザに情報を提供します（例：「引数の数が正しくありません」）。

引数
JSContext ***cx***、char ****error***、size_t sz

- **cx** 引数は、JavaScript 関数に渡される不透明 JSContext ポインタです。
- **error** 引数は、エラーメッセージを含むストリングです。ストリングはコピーされるので、呼び出し元はストリングが不要になった時点で、これを解放します。ストリングのサイズが指定されていない場合（詳しくは、**sz** 引数を参照してください）、ストリングはヌル終了されている必要があります。
- **sz** 引数はストリングサイズ（バイト数）です。**sz** が 0 の場合、ヌル終了されたストリングの長さが自動的に計算されます。

戻り値
ブール値。成功した場合は JS_TRUE、失敗した場合は JS_FALSE が返されます。

ファイルアクセスおよびマルチユーザ設定 API

Adobe では、C レベル拡張機能を使用してファイルシステムにアクセスする際は、常にファイルアクセスおよびマルチユーザ設定 API を使用することをお勧めします。Configuration ファイル以外のファイルについては、関数で指定したファイルまたはフォルダにアクセスします。

Dreamweaver は、Windows XP、Windows 2000 および Mac OS X の各オペレーティングシステムでマルチユーザ設定をサポートしています。

通常、Windows の C:¥Program Folders フォルダのようなアクセス制限されたフォルダに Dreamweaver をインストールします。これにより、管理者権限を持つユーザのみが Dreamweaver の Configuration フォルダの内容を変更することができます。また、マルチユーザオペレーティングシステム上のユーザが、個別の設定を作成および管理できるように、Dreamweaver ではユーザごとに独立した Configuration フォルダが作成されます。Dreamweaver または JavaScript 拡張機能により、Dreamweaver の Configuration フォルダに対する書き込み操作が実行されると、そのフォルダではなく、自動的にユーザの Configuration フォルダに書き込まれます。この操作により、他のユーザのカスタマイズ設定を変更することなく、ユーザごとに Dreamweaver の設定をカスタマイズできます。

ユーザの Configuration フォルダが作成される場所は、そのユーザが完全な読み取りおよび書き込みの権限を持つ場所です。Configuration フォルダの位置は、使用するプラットフォームによって異なります。

Windows 2000 および Windows XP プラットフォームの場合：

```
<drive>:\Documents and Settings\<username>\Application Data\Adobe\\
Dreamweaver CS4\Configuration
```

注意：Windows XP では、このフォルダは隠しフォルダ内部にある可能性があります。

Mac OS X プラットフォームの場合：

```
<drive>:Users:<username>:Library:Application Support:Adobe:Dreamweaver CS4:Configuration
```

通常は JavaScript 拡張機能でファイルを開き、Configuration フォルダに書き込みます。JavaScript 拡張機能は、DWFile、または MMNotes を使用するか、URL を dreamweaver.getDocumentDOM() 関数に渡すことによって、ファイルシステムにアクセスできます。拡張機能が Configuration フォルダのファイルシステムにアクセスするときは、通常 dw.getConfigurationPath() 関数を使用してファイル名を追加するか、開いているドキュメントの dom.URL プロパティにアクセスしてファイル名を追加することによってパスを取得します。拡張機能は、dom.URL にアクセスし、ファイル名を取り出すことによって、パスを取得することもできます。dw.getConfigurationPath() 関数と dom.URL プロパティは、ドキュメントがユーザの Configuration フォルダにある場合でも、常に Dreamweaver の Configuration フォルダの URL を返します。

JavaScript 拡張機能によって Dreamweaver の Configuration フォルダのファイルが開かれるたびに、そのアクセスに対して Dreamweaver による割り込みが入り、先にユーザの Configuration フォルダがチェックされます。JavaScript 拡張機能で、DWFile または MMNotes を使用して Dreamweaver の Configuration フォルダにデータを保存する場合は、その呼び出しに対して Dreamweaver による割り込みが入ります。次に、その呼び出しがユーザの Configuration フォルダにリダイレクトされます。

例えば、Windows 2000 または Windows XP で、ユーザが file:///C|/Program Files/Adobe/Adobe Dreamweaver CS4/Configuration/Objects/Common/Table.htm を要求すると、Dreamweaver によって先に C:¥Documents and Settings¥ ユーザ名 ¥adobe¥Dreamweaver CS4¥Configuration¥Objects¥Common フォルダで Table.htm というファイルが検索され、見つかれば代わりにそれが使用されます。

C レベル拡張機能または共有ライブラリでは、Dreamweaver の Configuration フォルダに対する読み取りおよび書き込みに、ファイルアクセスおよびマルチユーザ設定 API を使用する必要があります。ファイルアクセスとマルチユーザ設定 API を使用すると、Dreamweaver はユーザの Configuration フォルダに対する読み取りと書き込みができるようになります。また、アクセス権限が不足してファイル操作に失敗することはなくなります。C レベル拡張機能がアクセスする、Dreamweaver の Configuration フォルダ内にあるファイルが DWFile 操作、MMNotes 操作、または DOM 操作を使用した JavaScript によって作成される場合は、ファイルアクセスおよびマルチユーザ設定 API を使用します。このようなファイルは、ユーザの Configuration フォルダに置かれることもあります。

注意：大部分の JavaScript 拡張機能は、ユーザの Configuration フォルダに書き込むように変更する必要はありません。ファイルアクセスおよびマルチユーザ設定 API 関数を使用するために更新が必要なのは、Configuration フォルダに書き込む C 共有ライブラリのみです。

Dreamweaver の Configuration フォルダからファイルを削除すると、エントリがマスクファイルに追加されます。このエントリは、ユーザインターフェイスに表示しない Configuration フォルダ内のファイルを示すために追加されます。マスクされたファイルまたはフォルダは Dreamweaver には存在していないように見えますが、そのフォルダに存在しています。

例えば、スニペットパネルのごみ箱アイコンを使用して、javascript というスニペットフォルダと onepixelborder.csn というファイルを削除した場合、Dreamweaver によって、ユーザの Configuration フォルダに mm_deleted_files.xml というファイルが書き込まれます。このファイルは次のようになります。

```
<?xml version = "1.0" encoding="utf-8" ?>
    <deleteditems>
    <item name="snippets/javascript/" />
    <item name="snippets/html/onepixelborder.csn" />
    </deleteditems>
```

Dreamweaver によってスニペットパネルが表示されるときに、ユーザの Configuration¥Snippets フォルダ内のすべてのファイルが読み取られます。また、Configuration¥Snippets¥javascript フォルダと Configuration¥Snippets¥html¥onepixelborder.csn ファイルを除き、Dreamweaver の Configuration¥Snippets フォルダ内にあるすべてのファイルも読み取られます。ファイルの読み取り結果のリストがスニペットパネルのリストに追加されます。

C レベル拡張機能で MM_ConfigFileExists() 関数を file:///c|Program Files/Adobe /Adobe Dreamweaver CS4/Configuration/Snippets/javascript/onepixelborder.csn という URL に対して呼び出すと、値 false が返されます。同様に、JavaScript 拡張機能で dw.getDocumentDom(file:///c|Program Files/Adobe/Adobe Dreamweaver CS4/Configuration/Snippets/javascript/onepixelborder.csn) を呼び出そうとすると、null が返されます。

mm_deleted_files.xml ファイルを編集して、オブジェクトや新しいダイアログボックスでごみ箱に入れた内容などのファイルが、Dreamweaver によってユーザインターフェイスに表示されないようにします。MM_DeleteConfigfile() 関数を呼び出して、ファイルパスを mm_deleted_files.xml ファイルに追加できます。

JS_Object MM_GetConfigFolderList()

対応バージョン
Dreamweaver MX

説明
この関数は、指定したフォルダのファイル、フォルダまたはその両方のリストを取得します。Configuration フォルダを指定すると、この関数はユーザの Configuration フォルダと Dreamweaver の Configuration フォルダの両方に存在するフォルダのリストを取得します。このリストは mm_deleted_files.xml ファイルでフィルタリングされたものになります。

引数
char *fileURL、**char *constraints**

- **char *fileUrl** 引数は、内容のリストが必要なフォルダを示すストリングへのポインタです。このストリングは、file:// URL 形式で指定する必要があります。この関数では、file:// URL ストリングに、有効なワイルドカード文字であるアスタリスク（*）と疑問符（?）を使用できます。任意の文字を表すにはアスタリスク（*）を、任意の 1 文字を表すには疑問符（?）をそれぞれ使用します。
- **char *contstraints** 引数は、files か directories または null 値です。null を指定すると、MM_GetConfigFolderList() 関数からはファイルとフォルダが返されます。

戻り値
ユーザの Configuration フォルダまたは Dreamweaver の Configuration フォルダにあるファイルまたはフォルダのリストを含む配列である JSObject。このリストは mm_deleted_files.xml ファイルでフィルタリングされたものになります。

例

```
JSObject *jsobj_array;
jsobj_array = MM_GetConfigFolderList("file:///¬
    c|/Program Files/Adobe/Adobe Dreamweaver CS3/Configuration", "directories" );
```

JSBool MM_ConfigFileExists()

対応バージョン
Dreamweaver MX

説明
この関数は、指定したファイルが存在するかどうかをチェックします。それが Configuration フォルダ内のファイルの場合、この関数は、ユーザの Configuration フォルダまたは Dreamweaver の Configuration フォルダでそのファイルを検索します。また、そのファイル名が mm_deleted_files.xml ファイルにあるかどうかもチェックします。目的の名前がこのファイルにある場合は、この関数は false を返します。

引数
char *fileUrl

char *fileUrl 引数は、目的のファイルを示すストリングへのポインタです。このストリングは file:// URL の形式で指定します。

戻り値
ブール値。成功した場合は JS_TRUE、失敗した場合は JS_FALSE が返されます。

例

```
char *dwConfig = "file:///c|/Program Files/Adobe/Adobe Dreamweaver CS3/
Configuration/Extensions.txt";
int fileno = 0;
if(MM_ConfigFileExists(dwConfig))
{
    fileno = MM_OpenConfigFile(dwConfig, "read");
}
```

int MM_OpenConfigFile()

対応バージョン
Dreamweaver MX

説明
この関数は、ファイルを開き、オペレーティングシステムのファイルハンドルを返します。オペレーティングシステムのファイルハンドルは、システムファイルの関数への呼び出しで使用できます。このファイルハンドルは、システムの _close 関数を呼び出して閉じる必要があります。

ファイルが設定ファイルの場合は、ユーザの Configuration フォルダまたは Dreamweaver の Configuration フォルダでファイルを検索します。書き込みのために設定ファイルを開くと、そのファイルが Dreamweaver の Configuration フォルダに存在していても、この関数ではユーザの Configuration フォルダにそのファイルを作成します。

注意：ファイルに書き込む前にそのファイルを読み取る場合は、ファイルを "read" モードで開きます。ファイルに書き込む場合は、読み取りハンドルを閉じ、ファイルを再度 "write" または "append" モードで開きます。

引数
char *fileURL、char *mode

- **char *fileURL** 引数は、開くファイルを示すストリングへのポインタです。このストリングは file:// URL の形式で指定します。Dreamweaver の Configuration フォルダのパスを指定した場合、ファイルを開く前に、MM_OpenConfigFile() 関数によってパスが特定されます。
- **char *mode** 引数は、ファイルを開く方法を示すストリングを参照します。null、「read」、「write」、「append」のいずれかのモードを指定できます。「write」モードを指定しても、目的のファイルが存在しない場合は、MM_OpenconfigFile() 関数によってそのファイルが作成されます。「write」モードを指定した場合、MM_OpenConfigFile() 関数によって排他的共有でファイルが開かれます。"read" モードを指定した場合は、MM_OpenConfigFile() 関数によって非排他的共有でファイルが開かれます。

  「write」モードでファイルを開いた場合、新しいデータが書き込まれる前にファイルの既存のデータが切り捨てられます。「append」モードでファイルを開いた場合、書き込むデータはすべてファイルの終わりに追加されます。

戻り値
このファイルのオペレーティングシステムのファイルハンドルである整数。ファイルが見つからないか、存在しない場合は、-1 が返されます。

例

```
char *dwConfig = "file:///c|/Program Files/Adobe/Adobe Dreamweaver CS3/
    Configuration/Extensions.txt";
int = fileno;
if(MM_ConfigFileExists(dwConfig))
{
    fileno = MM_OpenConfigFile(dwConfig, "read");
}
```

JSBool MM_GetConfigFileAttributes()

対応バージョン
Dreamweaver MX

説明
この関数は、ファイルを検索し、そのファイルの属性を返します。**fileURL** 以外の引数には、値が必要なければ null を設定できます。

引数
char *fileURL、unsigned long *attrs、unsigned long *filesize、unsigned long *modtime、unsigned long *createtime

- **char *fileURL** 引数は、属性を取得するファイルを示すストリングへのポインタです。このストリングは、file:// URL の形式で指定する必要があります。**fileURL** に Dreamweaver の Configuration フォルダのパスを指定すると、ファイルを開く前に MM_GetConfigFileAttributes() 関数によってパスが特定されます。
- **unsigned long *attrs** 引数には、返される属性のビット列を格納する整数のアドレスを指定します（得られる属性について詳しくは、「355 ページの「JSBool MM_SetConfigFileAttributes()」」を参照してください）。
- **unsigned long *filesize** 引数には、この関数からバイト単位で返されるファイルサイズを格納する整数のアドレスを指定します。
- **unsigned long *modtime** 引数には、関数から返される、このファイルの最終更新時刻が格納されている整数のアドレスを指定します。この時刻は、オペレーティングシステムのシステム時刻で設定されます。オペレーティングシステムの

システム時刻について詳しくは、「DWfile.getModificationDate()」（『Dreamweaver API リファレンス』）を参照してください。

- **unsigned long *createtime** 引数には、関数から返される、ファイルの作成時刻が格納されている整数のアドレスを指定します。この時刻は、オペレーティングシステムのシステム時刻で設定されます。オペレーティングシステムのシステム時刻について詳しくは、「DWfile.getCreationDate()」（『Dreamweaver API リファレンス』）を参照してください。

戻り値

ブール値。成功した場合は JS_TRUE、失敗した場合は JS_FALSE が返されます。ファイルが存在しないか、属性の取得時にエラーが発生した場合に JS_FALSE が返されます。

例

```
char dwConfig = "file:///c|/Program Files/Adobe/Adobe Dreamweaver CS3/
    Configuration/Extensions.txt";
unsigned long attrs;
unsigned long filesize;
unsigned long modtime;
unsigned long createtime;
MM_GetConfigAttributes(dwConfig, &attrs, &filesize, &modtime, &createtime);
```

JSBool MM_SetConfigFileAttributes()

対応バージョン

Dreamweaver MX

説明

この関数は、ファイルに対してユーザから指定された属性が現在の属性と異なる場合に、指定された属性を設定します。

指定されたファイル URL が Dreamweaver の Configuration フォルダにある場合、この関数は最初にファイルをユーザの Configuration フォルダにコピーしてから、属性を設定します。属性が現在のファイル属性と同じ場合、ファイルはコピーされません。

引数

char *fileURL、unsigned long attrs

- **char *fileURL** 引数は、属性を設定するファイルを示すストリングへのポインタです。このストリングは file:// URL の形式で指定します。
- **unsigned long attrs** 引数には、ファイルに設定する属性のビット列を指定します。以下の定数に論理 OR を使用して、属性を設定できます。

```
MM_FILEATTR_NORMAL
MM_FILEATTR_RDONLY
MM_FILEATTR_HIDDEN
MM_FILEATTR_SYSTEM
MM_FILEATTR_SUBDIR
```

戻り値

ブール値。成功した場合は JS_TRUE、失敗した場合は JS_FALSE が返されます。ファイルが存在しないか、削除するためにマークされている場合に、JS_FALSE が返されます。

例

```
char *dwConfig = "file:///c|/Program Files/Adobe/Adobe Dreamweaver CS3/
    Configuration/Extensions.txt";
unsigned long attrs;
attrs = (MM_FILEATTR_NORMAL | MM_FILEATTR_RDONLY);
int fileno = 0;
if(MM_SetConfigFileAttrs(dwConfig, attrs))
{
    fileno = MM_OpenConfigFile(dwConfig);
}
```

JSBool MM_CreateConfigFolder()

対応バージョン
Dreamweaver MX

説明
この関数は、指定された場所にフォルダを作成します。

fileURL 引数に Dreamweaver の Configuration フォルダに含まれるフォルダが指定されると、この関数はユーザの Configuration フォルダにそのフォルダを作成します。**fileURL** に指定されたフォルダが Dreamweaver の Configuration フォルダに含まれるものではない場合、この関数は指定されたとおりのフォルダを作成します。パスの上位フォルダが存在しない場合は、それらも作成します。

引数
char *fileURL

- **char *fileURL** 引数は、作成する Configuration フォルダを示す file:// URL ストリングへのポインタです。

戻り値
ブール値。成功した場合は JS_TRUE、失敗した場合は JS_FALSE が返されます。

例

```
char *dwConfig = "file:///c|/Program Files/Adobe/Adobe Dreamweaver CS3 -
    /Configuration/Extensions.txt";
MM_CreateConfigFolder(dwConfig);
```

JSBool MM_RemoveConfigFolder()

対応バージョン
Dreamweaver MX

説明
この関数は、フォルダとそれに含まれるファイルおよびサブフォルダを削除します。フォルダが Dreamweaver の Configuration フォルダにある場合は、mm_deleted_files.xml ファイル内で削除対象としてマークします。

引数
char *fileURL

- **char *fileURL** 引数は、削除するフォルダを示すストリングへのポインタです。このストリングは file:// URL の形式で指定します。

戻り値
ブール値。成功した場合は JS_TRUE、失敗した場合は JS_FALSE が返されます。

例

```
char *dwConfig = "file:///c|/Program Files/Adobe/Adobe Dreamweaver CS3 -
    /Configuration/Objects";
MM_RemoveConfigFolder(dwConfig);
```

JSBool MM_DeleteConfigFile()

対応バージョン
Dreamweaver MX

説明
この関数は、指定されたファイルが存在する場合は、それを削除します。ファイルが Dreamweaver の Configuration フォルダにある場合は、mm_deleted_files.xml ファイル内で削除対象としてマークします。

fileURL 引数で指定されたフォルダが Dreamweaver の Configuration フォルダに含まれるものではない場合、この関数は指定されたフォルダを削除します。

引数
char *fileURL

- **char *fileURL** 引数は、削除する Configuration フォルダを示すストリングへのポインタです。このストリングは file:// URL の形式で指定します。

戻り値
ブール値。成功した場合は JS_TRUE、失敗した場合は JS_FALSE が返されます。

例

```
char dwConfig = "file:///c:|Program Files/Adobe/Adobe Dreamweaver CS3
    /Configuration/Objects/insertbar.xml";
MM_DeleteConfigFile(dwConfig);
```

JavaScript からの C 関数の呼び出し

Dreamweaver における C レベル拡張機能の動作と、特定のデータタイプと関数に対する依存性について理解したら、ライブラリを構築して関数を呼び出す方法について説明します。

以下の例では、Dreamweaver アプリケーションフォルダの Tutorial_assets/Extending フォルダに次の 5 つのファイルが必要です。これらのファイルは、Macintosh と Windows の両方のプラットフォームでアーカイブとして使用されます。

- mm_jsapi.h ヘッダーファイルは、「342 ページの「C レベル拡張機能と JavaScript インタープリタ」」で説明されているデータタイプと関数を定義します。
- mm_jsapi_environment.h ファイルは、MM_Environment.h 構造を定義します。
- MMInfo.h ファイルは、デザインノート API へのアクセスを指定します。

- Sample.c サンプルファイルは、computeSum() 関数を定義します。
- Sample.mak make file を Microsoft Visual C++ で使用すると、Sample.c ソースファイルを DLL に構築できます。Sample.mcp は、これと同じ機能を果たすファイルで、Metrowerks CodeWarrior を使用して Mach-O Bundle を構築できます。また、Sample.xcode は、Apple Xcode で同じ機能を果たすファイルです。これ以外のツールを使用する場合は、makefile を作成できます。

Windows で VS.Net 2003 を使用して DLL を構築する

1 「ファイルの種類」を「すべてのファイル (*.*)」に設定し、ファイル／開く／ Sample.mak を選択します *). （VS.Net 2003 では、MAK ファイルを直接開くことはできません）。プロジェクトを新しい形式に変換するかどうかを確認するメッセージが表示されます。

2 ビルド／ソリューションのリビルドを選択します。

構築操作が完了すると、Sample.dll ファイルが Sample.mak を含むフォルダ（またはそのサブフォルダ）に作成されます。

Windows で Microsoft Visual C++ を使用して DLL を構築する

1 Microsoft Visual C++ でファイル／ワークスペースを開く／ Sample.mak を選択します。

2 ビルド／すべてリビルドを選択します。

構築操作が完了すると、Sample.dll ファイルが Sample.mak を含むフォルダ（またはそのサブフォルダ）に作成されます。

Macintosh で Metrowerks CodeWarrior 9 以降を使用して共有ライブラリを構築する

1 Sample.mcp を開きます。

2 プロジェクトを構築（プロジェクト／メイク）して、Mach-O Bundle を生成します。

構築操作が完了すると、Sample.bundle ファイルが Sample.mcp を含むフォルダに作成されます。

注意：生成された Mach-O Bundle は、Dreamweaver 8 以降でのみ使用できます。以前のバージョンの Dreamweaver では認識されません。

Macintosh で Apple Xcode 1.5 以降を使用して共有ライブラリを構築する

1 Sample.xcode を開きます。

2 プロジェクトを構築（ビルド／ビルド）して、Mach-O Bundle を生成します。

構築操作が完了すると、Sample.bundle ファイルが Sample.xcode ファイルを含むフォルダに作成されます。

注意：生成された Mach-O Bundle は、Dreamweaver 8 以降でのみ使用できます。以前のバージョンの Dreamweaver では認識されません。

Insert Horizontal Rule オブジェクトから computeSum() 関数を呼び出す

1 Dreamweaver アプリケーションフォルダ内の Configuration フォルダに、JSExtensions というフォルダを作成します。

2 JSExtensions フォルダに Sample.dll（Windows）または Sample.bundle（Macintosh）をコピーします。

3 Configuration¥Objects¥Common フォルダにある HR.htm ファイルを、テキストエディタで開きます。

4 alert(Sample.computeSum(2,2)); という行を、次のように objectTag() 関数に追加します。

```
function objectTag() {
    // Return the html tag that should be inserted
    alert(Sample.computeSum(2,2));
    return "<HR>";
}
```

5 ファイルを保存して Dreamweaver を再起動します。

computeSum() 関数を実行するには、挿入／ HTML ／区切り線を選択します。

2 + 2 の計算結果である 4 という数値を含むダイアログボックスが表示されます。

第 24 章：Shared フォルダ

Shared フォルダは、すべての拡張機能で頻繁に使用するユーティリティ関数、クラス、およびイメージを集中的に格納するフォルダです。すべての拡張機能から Shared フォルダのサブフォルダにあるファイルを参照できます。また、Adobe Dreamweaver に用意されているユーティリティにカスタム共通ユーティリティを追加することもできます。Windows XP、Windows 2000、および Mac OS X プラットフォームを使用している場合でもマルチユーザの Configuration フォルダに Shared フォルダがインストールされるので、ユーザごとのカスタム化が可能です。例えば、Adobe Exchange から拡張機能をインストールすると、その新しい拡張機能の内容は、Dreamweaver アプリケーションの Configuration¥Shared フォルダではなく、ユーザの Configuration¥Shared フォルダに追加されます。マルチユーザコンピュータでの Dreamweaver の Configuration フォルダについて詳しくは、76 ページの「マルチユーザ設定フォルダ」を参照してください。

Shared フォルダの内容

Shared フォルダにはサブフォルダがあり、複数の拡張機能で共有するファイルが格納されています。これらには、ユーザのフォルダシステムを参照する関数、ツリーコントロールを挿入する関数、編集可能グリッドを作成する関数などがあります。

注意：Shared フォルダにある JavaScript ファイルのコード内には、ファイルに記述された関数の詳細を説明するコメントが記載されています。

Shared フォルダの JavaScript ファイルを参照する以外にも、JavaScript ファイルが使用されている HTML ファイルを Configuration フォルダで検索して、JavaScript ファイルがどのように使用されているかを調べることもできます。

一般的に、ユーザが使用する関数やリソースは Common および MM フォルダに保存されています。また、新しい拡張機能で使用するリソースは Common フォルダに追加します。ユーティリティおよび関数を検索する際には、必ず最初に Shared/Common/Scripts フォルダを参照してください。ここにある関数とユーティリティは最新のものであり、Shared フォルダに対する正式なインターフェイスです。その他のフォルダにあるファイルは最新でない場合があるので、自己の責任において使用してください。

具体的には、Shared フォルダには以下の有用なフォルダがあります。

Common フォルダ

Common フォルダには、サードパーティの拡張機能で使用する共有スクリプトおよび共有クラスが格納されています。

ファイル	説明
CodeBehindMgr.js	分離コードドキュメントを作成する関数が格納されています。分離コードドキュメントを使用すると、UI（ユーザインターフェイス）ロジック用コードと UI デザイン用コードを分離した個別のページを作成できます。このファイルで定義された JSCodeBehindMgr のメソッドを使用すれば、分離コードドキュメントを新規作成したり、デザインドキュメントへのリンクを管理することができます。
ColumnValueNodeClass.js	データベース列を値にマップするための関数が格納されています。このファイルで定義された ColumnValueNode のメソッドを使用すると、データベース列の様々な値およびプロパティを取得および設定できます。Dreamweaver では、編集操作オブジェクトの適用と検査（レコードオブジェクトの挿入と更新）および SQLStatement クラスでの作業にこのストレージクラスを使用します。
CompilerClass.js	CompilerASPNetCSharp および CompilerASPNVBNet で使用するベースクラスの関数が格納されていますが、拡張して他のコンパイラをサポートすることも可能です。
DataSourceClass.js	292 ページの「findDynamicSources()」の戻り値の構造を定義する関数が格納されています。

ファイル	説明
DBTreeControlClass.js	データベースのツリーコントロールを構築する関数が格納されています。このクラスはデータベースツリーコントロールの作成とそれとの対話操作に使用します。高度なレコードセットサーバビヘイビアに使用するコントロールのようなデータベースツリーコントロールを作成するには、専用の <select> リストを、type="mmdatabasetree" を使用して HTML ファイル内に作成します。CBTreeControl クラスのコンストラクタに <select> リスト名を渡して、そのクラスを HTML コントロールに関連付けます。次に、DBTreeControl 関数を使用して、このコントロールを操作します。
dotNetUtils.js	オブジェクトのプロパティインスペクタや ASP.NET フォームコントロールの操作を容易にする関数が格納されています。これらはトランスレートされています。
dwscripts.js	すべての Dreamweaver 拡張機能に便利な関数を見つけるには、このメインファイルを参照してください。このファイルにはストリング、ファイル、デザインノートなどを操作する関数が格納されています。
dwscriptsExtData.js	このファイルは dwscripts.js ファイルの拡張です。このファイルを使用すると、サーバビヘイビア、特にサーバビヘイビア EDML ファイルの操作が容易になります。Dreamweaver のサーバビヘイビアの実装で広く使用します。
dwscriptsServer.js	このファイルは dwscripts.js ファイルの拡張です。サーバモデル特有の関数が格納されています。これらの関数の多くはサーバビヘイビアの操作で使用します。
GridControlClass.js	編集可能グリッドの作成と操作にはこのクラスを使用します。HTML に特別な選択リストを追加し、JavaScript でリストにこのクラスを添付すると、グリッドを操作できます。
ImageButtonClass.js	このクラスを使用すると、押した状態のボタン、押したままマウスポインタを上に置いた状態のボタン、マウスポインタを上に置いた状態のボタン、押したままで無効になっている状態のボタンの各外観を容易に制御できます。
ListControlClass.js	<select> タグ（リストコントロール）を管理する関数が格納されています。このファイルの ListControl オブジェクトのメソッドを使用すると、SELECT コントロールの値の取得、設定、変更ができます。
PageSettingsASPNet.js	ASP.NET ドキュメントのプロパティを設定する関数が格納されています。
RadioGroupClass.js	ラジオボタングループを定義し、管理する関数が格納されています。このファイルの RadioGroup オブジェクトのメソッドを使用すると、ラジオボタングループの値とビヘイビアの設定と取得ができます。このクラスを HTML のラジオボタンに関連付けてビヘイビアを制御します。
SBDatabaseCallClass.js	ServerBehavior クラスのサブクラスです。このクラスには、ストアードプロシージャの呼び出しや SQL の使用によるレコードセットの取得など、データベースの呼び出しに固有な機能が含まれています。これは、抽象的なベースクラスなので、それ自体では作成や使用ができません。使用するには、SBDatabaseCall() をサブクラス化し、プレースホルダ関数を実装する必要があります。Dreamweaver では、サーバビヘイビアに対するレコードセットおよびストアードプロシージャの実装にこのクラスを使用します。
ServerBehaviorClass.js	サーバビヘイビアについての情報を Dreamweaver に通知する関数が格納されています。このクラスは、ユーザ独自のサーバビヘイビアの一部としてサブクラス化できます。
ServerSettingsASPNet.js	ASP.NET サーバのプロパティを格納する関数が格納されています。
SQLStatementClass.js	SELECT、INSERT、UPDATE、DELETE、などの SQL ステートメントおよびストアードプロシージャステートメントの作成と編集に使用する関数が格納されています。
tagDialogsCmn.js	カスタムタグダイアログボックスの開発で使用する関数が格納されています。このファイルで定義されている tagDialog オブジェクトのメソッドを使用すると、特定のタグの属性と値を修正できます。
TagEditClass.js	現在のページの DOM を変更せずにタグを編集する関数が格納されています。このファイルで定義されている TagEdit オブジェクトのメソッドを使用すると、タグの値、属性、子の取得と設定が可能です。
TreeControlClass.js	Dreamweaver 内のツリーコントロールを管理する関数が格納されています。このファイルで定義されている TreeControl オブジェクトのメソッドを使用すると、ツリー内の値の取得、設定、整列ができます。このクラスを HTML の MM:TREECONTROL タグに添付して、ツリーコントロール機能を管理します。
XMLPropSheetClass.js	XML プロパティシートの位置と値を管理する関数が格納されています。

MM フォルダ

MM フォルダには、Dreamweaver に付属の拡張機能で使用する共有スクリプト、イメージ、クラスが格納されています。例えば、ナビゲーションバーの構築、プリロード呼び出しの指定、ショートカットキーの定義に使用するスクリプトが格納されています。

Scripts サブフォルダ

Scripts サブフォルダには、以下のユーティリティ関数が格納されています。

ファイル	説明
CFCutilities.js	Adobe ColdFusion コンポーネントに関連するユーティリティ関数が格納されています。これらの関数は、指定されたノードの開始タグからの属性の解析、CFC ツリーの解析、現在の URL DOM の取得、CFC DOM の取得などを実行します。
event.js	イベントの登録、関係オブジェクトへの menus.xml ファイルからのイベントの通知、および menus.xml ファイルに対するイベント通知機能の追加を実行する関数が格納されています。
FlashObjects.js	カラーピッカーの更新、16 進数カラーのチェック、絶対リンクのチェック、ファイル名への拡張子の追加、エラーメッセージの生成、Flash 属性の設定、Flash オブジェクトのリンクのチェックなどを実行する関数が格納されています。
insertFireworksHTML.js	Adobe Fireworks CS3 HTML コードを Dreamweaver ドキュメントに挿入する関数が格納されています。これらの関数は、現在のドキュメントが Fireworks ドキュメントかどうかのチェック、挿入ポイントへの Fireworks HTML の挿入、Fireworks スタイルブロックの Dreamweaver への更新などを実行します。また、このファイルには関連するユーティリティ関数も格納されています。
jumpMenuUI.js	Jump Menu オブジェクトおよび Jump Menu ビヘイビアで使用する関数が格納されています。これらの関数は、メニューオプションへのデータの取り込み、オプションラベルの作成、オプションの追加、オプションの削除などを実行します。
keyCodes.js	キーボードのキーコード配列が格納されています。
navBar.js	ナビゲーションバーおよびナビゲーションバーエレメントの操作に使用するクラスおよび関数が格納されています。ナビゲーションバーエレメントの追加、削除、および操作を実行する関数があります。
NBInit.js	ナビゲーションバーイメージビヘイビアに関連する関数が格納されています。
pageEncodings.js	各種の言語コードを定義します。
preload.js	BODY/onLoad MM_preloadImages ハンドラに対するプリロードイメージ呼び出しの追加と削除に使用する関数が格納されています。
RecordsetDialogClass.js	レコードセットサーバビヘイビアの UI を表示する静的クラスおよび関数が格納されています。これらの関数は、基本インターフェイスと詳細インターフェイスのどちらを表示するかを決定します。また、UI 実装間で共有される機能も持ち、UI 間の切り替えの調整も行います。
sbUtils.js	アドビ システムズ社のサーバビヘイビア内で使用する共有関数が格納されています。Configuration¥Shared¥Common/Scripts フォルダの dwscripts クラスには、より一般的な用途のユーティリティが収められています。
setText.js	式のストリングのエスケープ、エスケープ解除、および抽出を実行する関数が格納されています。
sortTable.js	テーブルの初期化や並べ替えを実行する関数や配列を並べ替える関数、マウスポインタを手の形のアイコンやポインタに設定する関数、ブラウザのタイプとバージョンをチェックする関数が格納されています。

Scripts フォルダにも Class および CMN という 2 つのサブフォルダがあります。

Class フォルダ

Class フォルダには、以下のユーティリティ関数が格納されています。

ファイル	説明
classCheckbox.js	HTML 拡張機能でチェックボックスコントロールの操作に使用します。
FileClass.js	ファイルシステムにあるファイルを表すクラスが格納されています。プラットフォーム間の互換性を確保するために、パスは URL で表されます。メソッドには、toString()、getName()、getSimpleName()、getExtension()、getPath()、setPath()、isAbsolute()、getAbsolutePath()、getParent()、getAbsoluteParent()、exists()、getAttributes()、canRead()、canWrite()、isFile()、isFolder()、listFolder()、createFolder()、getContents()、setContents()、copyTo()、remove() などがあります。
GridClass.js	MM:TREECONTROL を管理するクラスが格納されています。
GridControlClass.js	Common フォルダにある GridControlClass の旧バージョンです。詳しくは、Shared/Common/Scripts フォルダの GridControlClass.js ファイルを参照してください。
ImageButtonClass.js	Common フォルダにある ImageButtonClass の旧バージョンです。詳しくは、Shared/Common/Scripts フォルダの ImageButtonClass.js ファイルを参照してください。
ListControlClass.js	Common フォルダにある ListControlClass の旧バージョンです。詳しくは、SShared/Common/Scripts フォルダの ListControlClass.js ファイルを参照してください。
NameValuePairClass.js	名前と値のペアのリストを作成し管理します。名前には任意の文字を使用できます。値は空白でも構いませんが、null に設定すると値が削除されるので、設定しないでください。
PageControlClass.js	TabControl クラスとともに使用するページクラスの例です。TabControlClass.js の説明を参照してください。
PreferencesClass.js	コマンドの環境設定情報すべてを含むオブジェクトとメソッドが格納されています。
RadioGroupClass.js	Common フォルダにある RadioGroupClass の旧バージョンです。詳しくは、Shared/Common/Scripts フォルダの RadioGroupClass.js ファイルを参照してください。
TabControlClass.js	複数のタブビューを持つ拡張機能 page.lastUnload() の構築に使用します。

CMN フォルダ

CMN フォルダには、以下のユーティリティ関数が格納されています。

ファイル	説明
dateID.js	createDateID() および decipherDateID() という 2 つの関数が格納されています。"dayFormat"、"dateFormat"、および "timeFormat" という 3 つのストリングが指定されると、createDateID() はそれらの ID を作成します。日付の配列が指定されると、decipherDateID() は、dayFormat、dateFormat および timeFormat の 3 つの項目を持つ配列を返します。
displayHelp.js	指定されたヘルプドキュメントを表示する関数が格納されています。
docInfo.js	ユーザのドキュメントについての情報を提供する関数が格納されています。これらの関数は、指定したブラウザタイプおよびタグのオブジェクト参照の配列を返す、指定したタグ名のすべてのインスタンスを返す、現在の選択範囲を含むタグを検索する、などの操作を実行します。
DOM.js	Dreamweaver の DOM の操作で使用する一般的な補助関数が格納されています。アクティブドキュメントのルートノードの取得、指定された名前のタグの検索、指定された開始ノード以降にあるノードのリストの作成、指定されたタグが別のタグに含まれているかどうかのチェック、ビヘイビア関数での様々な操作などを実行する関数があります。
enableControl.js	受け取った引数に基づいてコントロールを有効または無効にする関数 SetEnabled() が格納されています。既に有効になっているコントロールを再度有効にしたり、既に無効になっているコントロールを再度無効にしたりすることもできます。
errmsg.js	ダイアログボックスに表示されるログページの配列にトレース出力を蓄積するログ関数が格納されています。
file.js	ファイル操作に関連する関数が格納されています。これらの関数を使用すると、ローカルファイル名の参照、相対パスからファイル URL パスへの変換、現在のドキュメントのファイル名の取得、指定したドキュメントが現在のサイトに保存されているかどうかの判定とそのドキュメントの相対パスの取得、指定したファイルが現在開いているかどうかの判定などを実行できます。

ファイル	説明
form.js	現在のドキュメントまたは AP エレメントにフォームが存在しない場合に、特定のテキストストリングの周囲にフォームを追加する関数が格納されています。オブジェクトが AP エレメントであるかどうかの判別、および挿入ポイントがフォーム内にあるかどうかの判別を実行する関数があります。
handler.js	イベントハンドラの関数の取得、イベントハンドラへの関数の追加、イベントハンドラからの関数の削除を実行する関数が格納されています。
helper.js	エンコードの置換、二重引用符（"）のエスケープ解除、ノードが選択範囲内にあるかどうかのチェック、重複するオブジェクト名のチェックの実行に便利な関数がいくつか格納されています。
insertion.js	テキストストリングをドキュメントの挿入ポイントに挿入する insertIntoDocument() 関数が格納されています。また、サポート関数 getHigherBlockTag() および arrContains() も格納されています。getHigherBlockTag() 関数は、次に高い順位の blockTag を取得します。この順位は blockTags 配列に定義されています。arrContains() 関数は、配列内で指定された項目を検索します。
localText.js	予約変数です。一般的な用途向けではありません。Startup¥mminit.htm、または Dreamweaver の Configuration¥Strings¥*.xml ファイルにあるストリングを使用してください。
menuItem.js	リストされたメニュー項目に対して値を追加または削除する関数が格納されています。
niceName.js	オブジェクト参照の配列を単純な名前の配列に変換する関数が格納されています。
quickString.js	小規模のストリングを、それぞれにメモリを割り当てずに集約する関数が格納されています。
string.js	テキストストリングの操作と解析に使用する一般的な関数セットが格納されています。これらの関数には、extractArgs()、escQuotes()、unescQuotes()、quoteMeta()、errMsg()、badChars()、getParam()、quote()、stripSpaces()、StripChars()、AllInRange()、reformat()、trim()、createDisplayString()、entityNameEncode()、entityNameDecode()、stripAccelerator()、SprintF() があります。
TemplateUtils.js	Dreamweaver テンプレートのユーティリティ関数が格納されています。これらの関数は、編集可能領域のドキュメントへの挿入、リピート領域のドキュメントへの挿入、ドキュメント内の指定された編集可能領域のスキャンなどを実行します。
UI.js	UI を制御する一般的な関数が格納されています。これらの関数は、現在のドキュメントでの指定されたオブジェクトの検索、ローカライズされたストリングを持つ選択リストオプションの読み込み、選択されたオプションの属性値を返す処理、および警告テキストメッセージの折り返しを実行します。

その他のフォルダ

以下のリストでは、Shared フォルダにあるその他の関連フォルダについて説明します。

Controls　Controls フォルダには、サーバビヘイビアの構築に使用するエレメントが格納されています。これらのコントロールには、テキストやレコードセットのメニューのインターフェイスが含まれています。

注意：これらのコントロールは、Dreamweaver サーバビヘイビアビルダーや Dreamweaver の多くのサーバビヘイビアで使用しますが、拡張機能でコントロールを管理する場合に便利なものもあります。

Fireworks　Fireworks フォルダには Fireworks の統合をサポートするファイルが格納されています。

UltraDev　このフォルダは主に下位互換性を維持するためのものなので、新しい拡張機能では使用しません。この機能のほとんどは Dreamweaver の Configuration¥Shared¥Common フォルダにもあるので、そちらを使用します。360 ページの「Common フォルダ」を参照してください。

Shared フォルダの使用

Dreamweaver の Configuration¥Shared¥Common フォルダには頻繁に使用される最新の機能が格納されているので、便利な拡張機能コードを探す際には、まずこのフォルダを参照します。

拡張機能は、それ自身の機能のために Shared フォルダ内のリソースを再利用できます。オブジェクト、コマンドなどの拡張機能では、Shared フォルダにある JavaScript ファイルのいずれかをソースファイルとして script タグで指定しておけば、その関数をファイル本文または別の JavaScript インクルードファイルで使用できます。オブジェクトおよびコマンドからいくつかの JavaScript ファイルにまとめてリンクすると、それらの JavaScript ファイルで Shared フォルダのリソースを再利用することもできます。

例えば、Configuration¥Objects¥Common アプリケーションフォルダにあるハイパーテキストオブジェクトファイル（Hyperlink.htm）を開きます。そのファイルの head タグには以下の行が記述されています。

```
<script language="javascript" src="../../Shared/Common/Scripts/ListControlClass.js"></script>
<script language="javascript" src="Hyperlink.js"></script>
```

また、関連づけられた Hyperlink.js ファイルを開くと、以下の行が記述されています。

```
LIST_LINKS = new ListControl('linkPath');
```

と

```
LIST_TARGETS = new ListControl('linkTarget');
```

Hyperlink.js では、new listControl 宣言を使用して新しい ListControl オブジェクトが 2 つ定義されています。Hyperlink.htm ファイルのコードによって、フォーム内でそれらのオブジェクトは SELECT コントロールに次のように割り当てられます。

```
<td  align="left"> <input name="linkText" type="text" class="basicTextField" value="">
```

と

```
<td align="left" nowrap><select name="linkPath" class="basicTextField" editable="true">
```

これにより、Hyperlink.js スクリプトは、LIST_LINKS オブジェクトまたは LIST_TARGETS オブジェクトのメソッドの呼び出し、またはプロパティの取得を実行できるので、フォーム内の SELECT コントロールとの対話操作が可能になります。

索引

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

あ

い

う

え

お

か

き

く

け

こ

さ

し

す

せ

そ

た

つ

て

と

の

は

ひ

も

ゆ

ら

り

れ

ろ

わ