Appendice

E

EXTEND SCRIPT ET GOLIVE SDK

Une des nouvelles fonctionnalités la plus complexe et la plus puissante introduite dans GoLive 5 est l'Extend Script (script d'extension). Il vous permet d'ajouter de nouvelles possibilités et de nouvelles fonctionnalités à GoLive en écrivant du code JavaScript et des balises spéciales fournies avec le SDK (*Software Development Toolkit* — Outils de Développement de Logiciel) de l'Extend Script. Vous pouvez ajouter de nouveaux menus, de nouvelles palettes, des cases personnalisées, des Inspecteurs et plus encore. Il est hors du sujet de ce livre d'examiner l'Extend Script en détails mais cet appendice montrera brièvement comment écrire une extension standard de GoLive. Pour cela, vous devez avoir une connaissance de base de HTML et JavaScript.

Il est aussi possible d'écrire des modules d'extension binaire en utilisant C/C++ au lieu de JavaScript. Cela permet d'ajouter des capacités très sophistiquées à goLive au-delà de ce qu'il est possible de faire avec l'Extend Script standard. Ecrire un tel code requiert la connaissance du compilateur C et est dédié aux développeurs avancés uniquement.

Débuter avec Extend Script

Pour utiliser le module d'extension créé avec le SK et tester vos propres modules d'extension, vous devez d'abord vous assurer que le module Extend Script est activé dans les Préférences. L'installeur de GoLive active ce module

par défaut et vous ne devriez donc pas faire de modifications à moins de l'avoir déjà désactivé.

Vous désirerez peut-être essayer l'un des exemples d'extension livré avec le SDK pour vous donner une idée de ce qu'il peut faire. Si vous possédez un Mac, vous trouverez ces exemples dans le dossier SDK de votre CD-ROM d'installation. Si vous utilisez Windows, les exemples sont installés sur votre disque dur, dans le même dossier que celui de l'application GoLive. Pour installer les exemples d'extensions, glissez-les simplement dans le sous-dossier Extend Scripts du dossier Module.

La documentation du SDK et les échantillon de code sont régulièrement mis à jour. Au moment où vous lirez ce livre, une nouvelle version sera probablement disponible sur le suite d'Adobe à :
http://partners.adobe.com/asn/developer/gapsdk/GoLiveSDK.html.
Assurez-vous de toujours utiliser la dernière version.

Examiner la structure d'un Extend Script

Chaque module Extend Script est défini par un fichier nommé `Main.html`. Chaque script possède son fichier `Main.html` qui est situé dans son dossier propre, à l'intérieur du dossier Extend Scripts. Si des images ou d'autres ressources sont requises par le module (si, par exemple, l'Extend Script définit une nouvelle palette ou une boîte personnalisée), elles doivent être placées dans le même dossier que le fichier `Main.html`. La Figure E.1 montre la structure du répertoire d'un Extend Script.

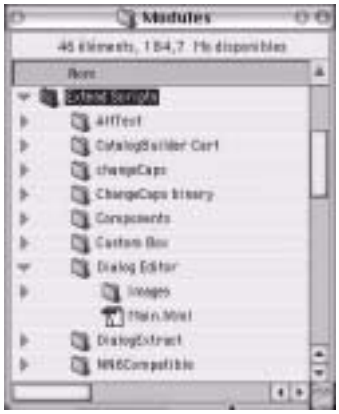

Figure E.1 : La structure du dossier Extend Scripts.

Bien que le fichier `Main.html` soit un fichier texte standard et contiennent les balises `<head>`, `<body>`, et d'autres code HTML, vous ne verrez en fait jamais cette page dans un

navigateur. Ce fichier contient des balises spécifiques SDK et du code JavaScript que GoLive interprète dès que vous lancez l'application et qui devient partie intégrante de celle-ci.

Comme GoLive charge les modules Extend Script lors de son lancement, le test de vos modules pourra prendre du temps car vous serez chaque fois obligé de quitte GoLive et de le relancer. Vous pouvez réduire ce temps de chargement de GoLive durant vos test en désactivant les modules que vous n'utilisez pas dans les Préférences.

Le code type d'une extension SDK se présente comme suit :

```
<html>
<head>
<title>GoLive 5 Extension Module</title>
</head>
<body>
  <!--Module name-->
<jsxmodule timeout=0 debug>

<script language="javascript"><!--
    //JavaScript that processes events goes here
    //-->
    </script>

    <!--various SDK tags to define palettes, menus etc. go here-->
  </body>
</html>
```

Comme vous pouvez le voir, son aspect n'est pas très différent de celui d'une page HTML standard.

Le SDK fournit plusieurs balises spéciales que vous pourrez utiliser pour définir de nouveaux éléments d'interface dans GoLive comme des palettes, des menus, des Inspecteurs, des balises personnalisées et des boîtes de dialogue. Toutes les balises SDK commence par les lettres *jsx* afin de les distinguer des autres balises standard. Le tableau suivant montre la liste de toutes les balises SDK et leur fonction :

Tableau E- 1
Les balises SDK de GoLive et leur fonction

Balise	Fonction
<jsxmodule>	Définit le nom du module, le délai d'erreur et l'état de débogage.
<jsxlocale>	Vous permet de fournir les versions internationalement localisées du texte présenté par votre module.
<jsxdialog>	Définit une boîte de dialogue.
<jsxpalette>	Définit une palette flottante similaire à la fenêtre de l'Inspecteur.
<jsxcontrol>	Définit des contrôles comme les cases à cocher, les champs de modification et les champs URL pour les palettes Inspecteur.

Tableau E- 1
Les balises SDK de GoLive et leur fonction

Balise	Fonction
<jsxpalettegroup>	Définit un "onglet" gérant des éléments de palette dans la palette Objets.
<jsxpaletteentry>	Définit un élément individuel de palette et la balise personnalisée qui sera écrite dans la page lorsque cet élément sera utilisé.
<jsxelement>	Définit une boîte personnalisée.
<jsxinspector>	Définit un Inspecteur.
<jsxmenubar>	Enveloppe un bloc de définition de menu.
<jsxmenu>	Définit un menu individuel ou un sous-menu.
<jsxitem>	Définit un article de menu.
<img>	Utilisé pour référencer toutes images utilisées par votre extension. Typiquement une palette d'icônes.

Vous pouvez mélanger des balises HTML normales avec des balises SDK dans le fichier `Main.html`. La plupart d'entre elles seront ignorées par GoLive à moins qu'elles soient englobée dans un conteneur SDK, mais elle peuvent néanmoins être utilisée pour afficher l'usage et les informations de copyright de l'extension au cas où un utilisateur ouvrirait ce fichier.

Tout comme les balises HTML standard, ces balises spécifiques possèdent leur propre jeu d'attributs qui modifient leur comportement. Vous pouvez, par exemple, définir la largeur et la hauteur d'une boîte personnalisée de taille fixe en paramétrant les attributs `fixedWidth` et `fixedHeight` de la balise.

Vous n'avez pas besoin d'écrire les balises dans un ordre spécifique dans le fichier `Main.html`, mais certaines balises requièrent la présence de certaines autres. C'est le cas de la balise `<jsxinspector>`, par exemple, car elle présente un Inspecteur dans une boîte personnalisée qui d'abord être définie.

Explorer JavaScript et le modèle Objet de GoLive

Dès que vous avez défini les éléments visuels de l'extension à l'aide des balises, vous définir sa fonctionnalité. Vous utiliserez JavaScript pour la rendre vivante.

GoLive développe l'arborescence d'un document comme un modèle d'objet JavaScript standard. Cela signifie que tous les éléments de marquage d'une page sont accessibles en tant qu'objets JavaScript et que leurs attributs le sont en tant que propriétés des ces objets. Vous

pouvez également référencer plusieurs propriétés et objets globaux, comme le nom du module courant, par exemple.

La coquille JavaScript

GoLive fournit une palette de coquilles JavaScript extrêmement utiles lors du développement d'un module car elles vous permettent de voir les propriétés courante de tout objet en tapant simplement le nom de la propriété de cet objet dans le champ de commande. Si la coquille JavaScript n'est pas visible sur votre écran, vous pouvez l'appeler en sélectionnant JavaScript Shell dans le menu Fenêtre. Essayez cette coquille en tapant **`document.title`** dans le champ Command et en appuyant sur la touche Entée ou retour chariot. Le nom du document courant devrait apparaître dans le panneau Result, comme le montre la Figure E.2 Si aucun document n'est ouvert, ce panneau affichera une erreur La coquille vous permet également d'utiliser directement une commande JavaScript. Essayez de taper **`alert("Salut tout le monde");`** dans le champ Command et en validant avec la touche Entrée ou retour chariot.

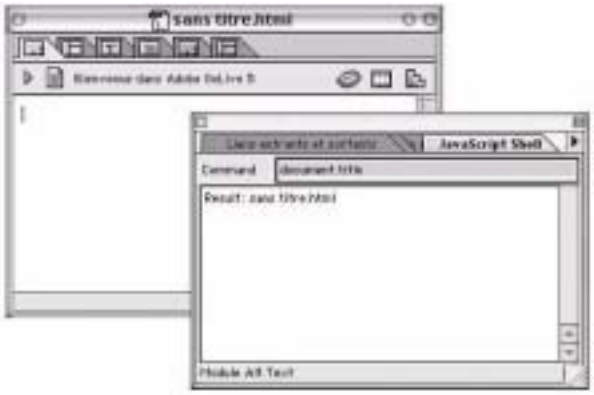

Figure E.2 : La palette Shell JavaScript (Coquille JavaScript).

Comme il a été dit précédemment, si vous modifiez le code de votre module d'extension, vous devrez quitter et relancer GoLive pour que celles-ci prennent effet. Cependant, si vous n'opérez des changements que dans le code JavaScript et dans dans le balises, vous n'avez pas besoin de quitter GoLive — vous pouvez recharger le code JavaScript à la volée pour toutes vos extensions. Utilisez pour cela le menu déroulant de la palette JavaScript Shell (accessible par la flèche noir située en haut et à gauche) et choisissez Reload all scripts (recharger tous les scripts). Cela vous fera économiser beaucoup de temps.

Les fonctions

Toutes les interactions avec les éléments de votre extension sont gérées par des fonctions JavaScript qui répondent à des évènements JavaScript. Différentes fonctions standard sont appelées par GoLive lorsque certains évènements surviennent — quand un utilisateur sélectionne un article de menu. Par exemple, GoLive appelle la fonction `menuSignal()` chaque fois qu'un article de menu est sélectionné. Si vous définissez cette fonction dans votre module

d'extension, alors GoLive exécutera le code à chaque sélection d'article de menu. Il existe trop de fonctions standard pour les répertorier toutes ici, mais elle sont couvertes en détails dans la documentation PDF qui accompagne le SDK de GoLive.

Tout comme vous pouvez écrire du code pour gérer les fonctions standard, vous pouvez définir vos propres fonctions, comme vous le faites en utilisant le JavaScript standard. En fait, si vous êtes familier avec le développement en JavaScript, vous pouvez déjà posséder certaines fonctions qui seront réutilisables dans une extension GoLive.

Ecrire un Extend Script

Pour vous donner une idée claire de la manière dont tout cela s'organise, je vais vous montrer le processus d'écriture d'un simple Extend Script. Cet exemple d'extension n'est par particulièrement utile — il inverse le texte sélectionné — mais il démontre comment fonctionne le SDK.

En premier lieu, créez un dossier à l'intérieur du dossier Modules/Extend Scripts et donnez-lui un nom unique. Puis utilisez GoLive pour créer une page HTML vierge que vous enregistrez dans le dossier que vous venez de créer sous le nom `Main.html`.

Le résultat de cet exercice se trouve sur le CD-ROM. Après avoir terminé cette première exploration des Extend Scripts, vous pourrez vérifier votre travail en ouvrant le script situé dans le dossier Exercices qui est organisé par chapitre.

Définir le module

Passez en mode source. La première balise que vous allez ajouter est la balise `<jsxmodule>`. Tapez ensuite les attribut suivant comme cela :

```
<jsxmodule name="revers-o-matic" timeout="0" debug>
```

Cette balise fait trois choses :

- Elle spécifie le nom du module. Si vous n'en spécifiez aucun ici, GoLive utilisera celui du dossier qui contient ce fichier `Main.html`.
- Elle active le débogage. cela vous permet d'utiliser la fonction intégrée de débogage de GoLive pour vérifier les problèmes de code.
- Elle définit un délai d'erreur à zéro. Si vous ne paramétrez pas cet attribut, dans certaines circonstances, si votre code contient des erreurs, GoLive peut entrer dans une boucle sans fin. Cet attribut force le délai de la boucle , évitant un gel de l'application.

Créer le menu

Maintenant, définissons l'article du menu. Comme cet Extend Script ne réalise qu'une seule fonction, vous n'avez besoin que d'un seul article de menu et il n'est donc pas nécessaire de

créer un nouveau menu complet. GoLive permettant d'ajouter des article de menu dans le menu spécial, c'est ce que nous allons faire.

Ajoutez le code suivant à votre document, juste en-dessous de la balise `<jsxmodule>`.

```
<jsxmenubar>
  <jsxmenu name="special">
    <jsxitem name="reversomatic" title="Texte inverse">
  </jsxmenu>
</jsxmenubar>
```

Comme vous pouvez le voir, le marquer créant le menu est suffisamment explicite. La balise `<jsxmenubar>` englobe le bloc de définition de menu. Toute balise de définition de menu dans `Main.html` doit être placée dans le conteneur `<jsxmenubar> </jsxmenubar>`. Ce conteneur définit un menu réel — en général, il est utilisé pour créer un menu autonome, lais en spécifiant "special" dans l'attribut `name`, vous ajoutez l'article de menu dans le menu Spécial existant. Vous ne pouvez pas ajouter d'articles de menu dans d'autres menus standard.

La balise `<jsxitem>` définit l'article de menu lui-même. Le nom de l'attribut sera utilisé plus tard par votre JavaScript — il nomme l'objet article de menu dans le modèle d'objet de Golive. L'attribut `title` est le texte réel qui s'affichera dans le menu.

Si vous le désirez, vous pouvez tester ce menu dès maintenant — enregistrez le fichier `Main.html`, quittez puis relancez GoLive. Si vous allez ensuite dans le menu Spécial, vous verrez un menu appelé "Texte inverse", comme le montre la Figure E.3. Le menu ne fera pas autre chose que s'afficher à ce point l'exercice. Mais nous y sommes presque.

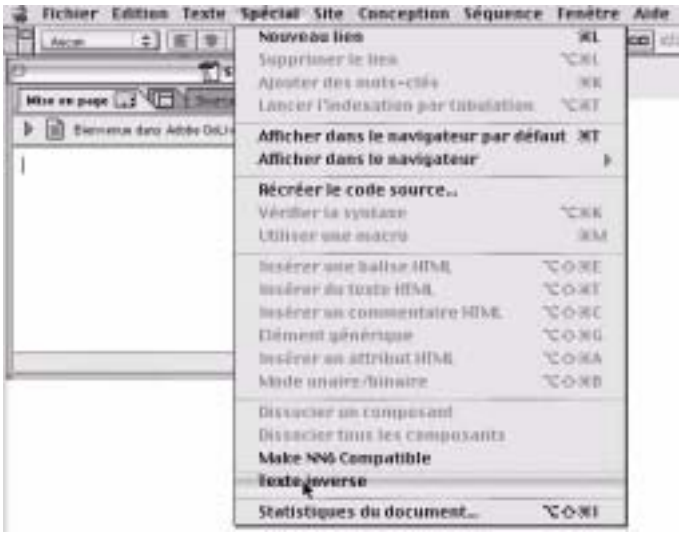

Figure E.3 : Le nouvel article de menu s'affiche dans le menu Spécial.

Gérer l'évènement du menu

Maintenant il vous faut indiquer quoi faire à GoLive lorsque cet article de menu sera sélectionné. Ré-ouvrez le fichier Main.html et passez en mode Mise en page. Vous remarquerez que les balises de marquage SDK sont montrées comme balise non-HTML. Comme vous avez besoin de créer un peu de code JavaScript, faites glisser l'icône JavaScript depuis l'onglet Standard de la palette Objets dans la page, en vous assurant de la placer entre les blocs `<jsxmodule>` et `<jsxmenubar>`. Double cliquez sur l'icône JavaScript pour ouvrir la fenêtre de l'Editeur.

Vous n'êtes pas obligé d'utiliser l'Editeur JavaScript pour modifier vos scripts — il est possible de le faire directement en mode Source — mais l'Editeur fournit des fonctionnalités pratiques d'édition du code, comme la coloration de la syntaxe et la vérification d'erreur qui permettent un débogage plus facile.

Quand un article de menu est sélectionné, GoLive appelle la fonction `<menuSignal>`. Pour définir la fonction, entrez le code suivant dans l'Editeur JavaScript :

```
function menuSignal(menuItem)
{
  switch (menuItem.name)
  {
  case "reversomatic":  checkselection(); break;
  default: alert ("Sorry, an error occurred.");
  }
}
```

Lorsque GoLive appelle la fonction, il envoie également l'argument `menuItem`. C'est l'objet d'article de menu standard de GoLive qui possède donc une propriété `name`, que vous tester en utilisant l'instruction `switch`. Cette instruction vérifie si `menuItem.name` est "reversomatic". C'est le nom que nous avons donné à l'article de menu dans la balise `<jsxitem>`. Si le nom du menu est "reversomatic" (ce qui est évidemment le cas) alors la fonction `checkselection()` est appelée; sinon un message d'erreur s'affiche.

Vérifier la validité de la sélection

La fonction `checkselection()` est requise car le script n'est valide que si l'utilisateur à auparavant sélectionné du texte. Vous ne voulez évidemment pas essayer et exécuter une fonction de manipulation de texte sur un tableau ou une image, et vous avez donc besoin de vérifier que la sélection active est du texte valide. C'est exactement ce que va faire le code suivant que vous allez ajouter à votre module :

```
function checkselection()
{
  //ensure that the user has selected text and not an image etc.
  if (document.selection.element.elementType != "text")
  {
    alert("Selectionner du texte contigu.");
    return false;
```

```
  }
  switch (document.selection.type)
  {
  // check the selection type

    case "full":  var el = document.selection.element;
                  var strt=document.selection.start;
                  var len=document.selection.length;
                  reverseText(el,strt,len);
                  break;
    case "none":  alert("Aucune selection de faite.");
                  break;
    case "point":  alert("Aucune selection de faite.");
                   break;
    case "complex":alert("Selectionner du texte contigu.");
                   break;
    case "part":  var el = document.selection.element;
                  var strt=document.selection.start;
                  var len=document.selection.length;
                  reverseText(el,strt,len);
                  break;
    case "outside":alert("Vous avez fait ce qui est  impossible!");
                   break;
    default:  alert("Une erreur est survenue.");
  }
}
```

Rappelez-vous que l'apostrophe est un code JavaScript de début ou de fin de chaîne de caractères. Si vous devez employer une apostrophe dans vos messages, faites la précéder du caractère d'échappement (anti-slash) en écrivant, par exemple "S\'il vous plaît". D'autre part, souvenez-vous également que seuls les navigateurs récent gèrent les caractères accentués. Soyez subtils dans vos messages si vous voulez les éviter — Ndt.

La première partie de cette fonction vérifie si l'élément sélectionné est bien du texte et non pas un autre type de balise. Elle le fait en vérifiant la valeur de la propriété `document.selection.element.elementType`. C'est une propriété standard de l'objet `element` qui peut contenir la valeur "text", "tag", "comment" ou "bad". Comme seule la sélection de texte vous intéresse, vous renvoyer un message d'erreur si la sélection est autre.

La section suivante de la fonction vérifie le type de sélection que l'utilisateur à fait. Lorsque vous sélectionnez un objet dans GoLive, vous pouvez le sélectionner en partie , dans sa totalité ou en sélectionner plusieurs éléments. La fonction de manipulation de texte ne peut pas s'exécuter sur plusieurs objets ou si aucun objet n'est sélectionné. Vous pouvez tester la propriété `type` de `document.selection` pour déterminer quel est l'état de la sélection courante.

Si la sélection est du type "full" ou "part" — en d'autre termes, si l'utilisateur à sélectionné tout ou partie d'un texte — alors la fonction `reverseText()` est appelée et lui sont envoyés comme arguments l'élément sélectionné, son début et sa longueur. Si l'utilisateur a fait une

sélection "complex" (e) (une sélection multiple, y compris une sélection de plusieurs paragraphes) ou s'il n'a fait aucune sélection, un message d'erreur s'affiche en utilisant la fonction JavaScript standard "`alert()`".

Effectuer le travail

Maintenant que vous savez avoir une sélection de texte valide, vous pouvez vous préoccuper de la manipulation de texte pour laquelle ce module est conçu. Tout le vrai travail est effectué par la fonction suivante que vous devriez ajouter maintenant à votre code :

```
function reverseText(el,strt,len)
{

var newtxt="";

//break selection into component parts

  var txt=el.getInnerHTML();
  var reverse=txt.substring(strt,(strt+len));
  var lefttxt=txt.substring(0,strt);
  var righttxt=txt.substring((strt+len),txt.length);

  for (i=0; i<=reverse.length; i++)
  {
//loop through the text in reverse and store the result
 var newtxt = newtxt + reverse.charAt(reverse.length-i);
  }

  //replace the current text with our new text
  el.setInnerHTML(lefttxt+newtxt+righttxt);

  //force GoLive to reparse the document so our changes are displayed
  document.reparse();
}
```

La fonction commence par créer une variable de chaîne de caractères vide qui contiendra la nouvelle chaîne inversée. Comme la fonction `setInnerHTML()` est utilisée pour écrire le texte inversé sur le document et remplacer l'ancien, nous devons nous assurer de n'inverser que le texte sélectionné. La fonction découpe l'élément de texte en texte avant la sélection, texte après la sélection et le texte sélectionné lui-même. Puis une boucle teste l'ordre des caractère, l'inverse et le place dans la nouvelle variable grâce à la fonction JavaScript standard `charAt()`.

Puis la fonction appelle `serInnerHTML()` qui est une fonction GoLive chargée de remplacer le contenu d'un élément marqué par une chaîne spécifiée. Dans cet exemple, nous l'appliquons à l'élément texte qui contient la sélection courante. Cela écrit la nouvelle chaîne de caractères inversés dans le document HTML.

Enfin, la fonction appelle la fonction `document.reparse()` qui met à jour la vue en mode Mise en page.

C'est tout !. Il vous suffit maintenant de fermer l'Editeur JavaScript, d'enregistrer la page, de quitter et relancer GoLive pour tester votre nouvelle extension.

Tester et débuguer

Si tout va bien, vous devriez pouvoir sélectionner du texte sur une page, d'aller dans le menu Spécial et choisir Texte inverse. Si tout va vraiment très bien, le texte devrait s'inverser.

Cependant, il possible que surgisse une fenêtre ressemblant à celle montrée par la Figure E.4 c'est le debugueur JavaScript, que vous avez rendu actif dans la balise `<jsxmodule>`. La partie inférieure de cette fenêtre affiche votre code. La partie supérieure droite affiche la sortie de débogage, ici, l'erreur qui est la cause de l'affichage de cette fenêtre. La partie supérieure gauche affiche la partie du code qui était exécuté lorsque la fenêtre a été appelée.

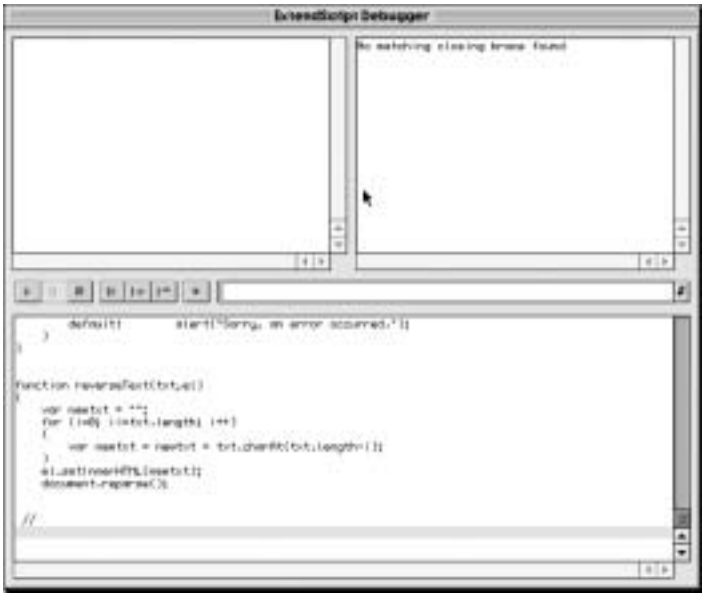

Figure E.4 : La fenêtre de débogage de Javascript.

Si cette fenêtre s'affiche chez vous, c'est que vous avez fait une erreur de typographie lorsque vous avez entré le code de l'exemple. La syntaxe de JavaScript est particulièrement spécifique et aussi sensible à la casse. Assurez-vous que vous avez entré le code correctement. Vous devriez pouvoir utiliser la fenêtre de débogage pour situer l'endroit où s'est produite l'erreur et la réparer.

En savoir plus sur l'Extend Script et le SDK

Cela n'était qu'un tout petit exemple de la puissance de L'Extend Script et du SDK. Beaucoup d'autres choses pourraient être expliquées ici. Votre prochaine étapes devrait donc être de lire la documentation d'Adobe et d'étudier les exemples pré-écrits qui sont fournis. Vous pourrez ainsi découvrir certaines des possibilités qu'offre le SDK