

ADOBE® INDESIGN® CS5.5



DIDACTICIEL DE SCRIPTS ADOBE INDESIGN CS5.5

© 2011 Adobe Systems Incorporated. Tous droits réservés.

Didacticiel de scripts Adobe® InDesign® CS5.5

Si ce guide est distribué avec un logiciel comprenant un accord avec l'utilisateur final, ce guide, ainsi que le logiciel qu'il décrit, sont fournis sous licence et ne peuvent être utilisés ou copiés qu'en accord avec les termes de cette licence. Sauf autorisation spécifiée dans la licence, aucune partie de ce guide ne peut être reproduite, enregistrée ou transmise sous quelque forme que ce soit, par quelque moyen que ce soit, électronique, mécanique ou autre, sans l'autorisation écrite préalable d'Adobe Systems Incorporated. Veuillez noter que le contenu de ce guide est protégé par des droits d'auteur, même s'il n'est pas distribué avec un logiciel accompagné d'un contrat de licence pour l'utilisateur final.

Les informations contenues dans ce document sont données à titre purement indicatif. Elles peuvent être modifiées sans préavis et ne constituent pas un engagement de la part d'Adobe Systems Incorporated. Adobe Systems Incorporated ne peut être tenu pour responsable des erreurs ou des inexactitudes apparaissant dans les informations fournies par ce guide.

Nous attirons votre attention sur le fait que les illustrations ou images que vous pouvez être amené à incorporer à vos projets peuvent être protégées par des droits d'auteur, auquel cas leur exploitation sans l'autorisation de l'auteur constituerait une violation de ces droits. Veuillez à obtenir toutes les autorisations requises de la part des auteurs.

Toutes les références à des noms de société dans les modèles cités en exemple sont indiquées uniquement à des fins de démonstration et ne se réfèrent à aucune organisation existante.

Adobe, le logo Adobe, Creative Suite, InDesign, Illustrator et Photoshop sont des marques ou des marques déposées d'Adobe Systems Incorporated aux Etats-Unis et/ou dans d'autres pays. Microsoft et Windows sont des marques ou des marques déposées de Microsoft Corporation aux Etats-Unis et dans d'autres pays. Apple et Mac OS sont des marques d'Apple Computer, Incorporated, déposées aux Etats-Unis et dans d'autres pays. Toutes les autres marques citées sont la propriété de leurs détenteurs respectifs.

Adobe Systems Incorporated, 345 Park Avenue, San Jose, Californie 95110, Etats-Unis. Avertissement à l'attention des utilisateurs finaux du gouvernement des Etats-Unis. Le logiciel et la documentation sont des articles commerciaux, (« Commercial Items »), selon la définition de ce terme dans l'article 48 C.F.R. §2.101, composés d'un logiciel commercial (« Commercial Computer Software ») et d'une documentation commerciale relative au logiciel (« Commercial Computer Software Documentation »), selon la définition de ces termes dans l'article 48 C.F.R. §12.212 ou 48 C.F.R. §227.7202, selon le cas. Conformément aux articles 48 C.F.R. §12.212 ou 48 C.F.R. §§227.7202-1 à 227.7202-4, selon le cas, le logiciel commercial et la documentation commerciale relative au logiciel sont cédés sous licence aux utilisateurs du gouvernement des Etats-Unis (a) en tant qu'articles commerciaux uniquement et (b) avec les seuls droits conférés à tout autre utilisateur final tenu par les termes et conditions stipulés ici. Droits non publiés réservés en vertu de la législation américaine sur les droits d'auteurs. Adobe Systems Incorporated, 345 Park Avenue, San Jose, CA 95110-2704, Etats-Unis. Pour les utilisateurs finaux du gouvernement des Etats-Unis, Adobe s'engage à respecter toutes les lois sur l'égalité des chances, y compris, le cas échéant, les dispositions du décret-loi (Executive Order) 11246, tel qu'amendé, la section 402 de l'Acte d'assistance à la réhabilitation des vétérans du Vietnam (the Vietnam Era Veterans Readjustment Assistance Act) de 1974 (38 USC 4212) et la section 503 de l'Acte de réhabilitation (Rehabilitation Act) de 1973, tel qu'amendé, ainsi que les règlements de l'article 41 C.F.R., sections 60-1 à 60-60, 60-250 et 60-741. Les règlements et la clause d'action affirmative contenus dans la phrase précédente doivent être inclus comme référence.

Sommaire

Introduction	5
Prise en main	5
Installation des scripts	6
Exécution d'un script	6
Utilisation du panneau Scripts	6
A propos des langages de script	7
JavaScript	7
Windows	8
Mac OS	8
Langage de script à utiliser	8
Utilisation des scripts de ce document	9
Votre premier script InDesign	9
AppleScript	9
JavaScript	10
VBScript	10
Analyse du script	10
Scripts et modèle d'objet InDesign	11
Terminologie des scripts	11
Commentaires	11
Valeurs	12
Variables	14
Opérateurs	16
Instructions conditionnelles	16
Structures de contrôle	16
Fonctions et gestionnaires	17
Présentation du modèle d'objet InDesign	17
Consultation du modèle d'objet InDesign	20
Mesures et positionnement	25
Ajout de caractéristiques à « Hello World »	26
AppleScript	26
JavaScript	28
VBScript	29

Construction d'un document	30
Configuration des unités de mesure et des marges des planches types	32
AppleScript	32
JavaScript	32
VBScript	33
Ajout d'une grille de ligne de base	33
AppleScript	34
JavaScript	34
VBScript	34
Ajout d'éléments de gabarit	34
AppleScript	35
JavaScript	35
VBScript	36
Ajout de blocs de texte types	36
AppleScript	37
JavaScript	37
VBScript	37
Remplacement des éléments de gabarit et ajout de texte	38
AppleScript	38
JavaScript	38
VBScript	38
Ajout et application d'un style de paragraphe	39
AppleScript	39
JavaScript	40
VBScript	41
Importation d'un fichier texte	41
AppleScript	41
JavaScript	42
VBScript	42
Importation d'une image	42
AppleScript	43
JavaScript	44
VBScript	45
Perfectionnement	46

Didacticiel de scripts Adobe InDesign CS5.5

Introduction

Les scripts représentent la fonction la plus puissante d'Adobe® InDesign®. Aucune autre fonction n'est aussi rapide, facile et économique que le script.

Le présent document s'adresse à tous les utilisateurs d'InDesign. Si vous n'avez jamais créé un script, ce document vous aidera à commencer. Si vous avez déjà créé des scripts pour d'autres applications, vous apprendrez à appliquer vos connaissances aux scripts InDesign.

Ce document traite également de l'installation et de l'exécution d'un script InDesign, ainsi que des atouts et des limitations des scripts InDesign. Vous trouverez des indications sur les logiciels dont vous avez besoin pour commencer à écrire vos propres scripts.

Une fois que vous aurez acquis les connaissances de base sur les scripts d'InDesign à l'aide de ce didacticiel, vous pourrez consulter le *Guide des scripts d'Adobe InDesign*, qui explore les scripts de manière plus détaillée. Le *Guide des scripts d'Adobe InDesign* contient des centaines de scripts pédagogiques portant sur des sujets tels que la mise en forme du texte, la recherche ou la modification de texte, l'association de scripts et d'éléments de menu, la création d'objets sur une page et l'exportation de documents.

Prise en main

À l'aide des scripts, vous pouvez reproduire la plupart des opérations que vous exécutez par le biais de l'interface utilisateur. Vous pouvez dessiner des cadres, entrer du texte et le mettre en forme, importer des images et imprimer ou exporter les pages du document. Toute opération apportant une modification au document ou à son contenu peut être réalisée à l'aide d'un script. Les scripts vous permettent même d'exécuter certaines opérations qui ne sont pas réalisables avec l'interface utilisateur.

Les scripts permettent de créer des menus, d'ajouter des éléments de menu, de créer et d'afficher des boîtes de dialogue et des panneaux, mais également de répondre aux sélections de l'interface utilisateur. Ils peuvent lire et écrire des fichiers texte, analyser des données XML et communiquer avec d'autres applications.

Les scripts permettent d'effectuer toutes sortes de tâches, des plus simples (telles que la définition d'un taquet de tabulation à l'emplacement du pointeur de texte) aux plus complexes, comme la définition d'un ensemble complet de fonctions (on doit la fonction d'exportation XHTML d'InDesign à un script). Vous pouvez commencer avec des scripts simples qui n'exécutent qu'une action, puis passer aux scripts qui automatisent le processus de publication tout entier.

Les tâches que vous ne pouvez pas exécuter à l'aide de scripts, telles que la définition d'un espace de travail ou d'un ensemble de raccourcis clavier, sont des tâches associées à l'interface utilisateur. En outre, les scripts ne permettent pas d'ajouter de nouveaux types d'objets à un document InDesign ou d'ajouter une nouvelle fonctionnalité de base au programme, telle qu'un moteur de composition de texte. Pour ce type d'extension, vous devez utiliser le kit SDK (Software Developer's Kit) d'InDesign, qui vous indique comment écrire des modules externes compilés à l'aide du langage C++.

Les scripts sont souvent associés dans les esprits à des tâches longues et répétitives, telles que la mise en page d'un annuaire téléphonique. Notez, toutefois, qu'ils permettent également d'exécuter d'autres tâches, telles que les suivantes :

- ▶ Automatisation d'une myriade de petites tâches fastidieuses quotidiennes
- ▶ Personnalisation d'InDesign pour l'adaptation aux habitudes de travail et mises en page de l'utilisateur
- ▶ Réalisation d'effets artistiques, difficiles voire impossibles à obtenir par d'autres moyens

Installation des scripts

L'installation d'un script InDesign est simple à réaliser : enregistrez le fichier de script dans le dossier Scripts Panel du dossier Scripts, dans votre dossier InDesign. (Si le dossier Scripts n'existe pas, créez-le.)

Vous pouvez également placer le script dans le dossier Scripts Panel de votre dossier de préférences, à l'emplacement suivant :

Windows® XP : C:\Documents and Settings*<username>*\Application Data
 \Adobe\InDesign\Version 7.0*<locale>*\Scripts

Windows® Vista : C:\Users*<username>*\AppData\Roaming
 \Adobe\InDesign\Version 7.0*<locale>*\Scripts

Mac OS® : /*<username>*/Library/Preferences
 /Adobe InDesign/Version 7.0/*<locale>*/Scripts

où *<username>* correspond à votre nom d'utilisateur et *<locale>* désigne votre région et votre langue, par exemple, en_US.

Une fois dans le dossier, le script s'affiche dans le panneau Scripts de l'application InDesign. Pour afficher le panneau, choisissez la commande Fenêtre > Utilitaires > Scripts.

Vous pouvez également placer des alias ou des raccourcis de scripts (ou de dossiers contenant des scripts) dans le dossier Scripts Panel. Ils s'afficheront dans le panneau Scripts.

Pour exécuter un script particulier au démarrage d'InDesign, placez ce dernier dans un dossier nommé « Startup Scripts », sous le dossier Scripts (si le dossier Scripts n'existe pas, créez-le.)

Exécution d'un script

Pour exécuter un script, vous devez afficher le panneau Scripts (choisissez la commande Fenêtre > Utilitaires > Scripts), puis cliquer deux fois sur le nom du script dans le panneau Scripts. De nombreux scripts affichent des éléments d'interface utilisateur (boîtes de dialogue ou panneaux, par exemple), ainsi que des messages d'alerte, si nécessaire.

Utilisation du panneau Scripts

Le panneau Scripts d'InDesign représente le meilleur moyen d'exécuter la plupart des scripts InDesign. Si ce panneau n'est pas visible, vous pouvez l'afficher en choisissant la commande Fenêtre > Utilitaires > Scripts.

Les scripts activés à partir du panneau Scripts sont exécutés plus rapidement que ceux activés à partir du Finder (Mac OS®) ou de l'Explorateur (Windows®).

Pour accélérer l'exécution des scripts, vous pouvez également désactiver la réactualisation de l'écran lors de l'exécution des scripts. Pour cela, désactivez l'option Activer la réactualisation dans le menu du panneau Scripts. Pour afficher les opérations du script à mesure qu'elles s'exécutent, activez cette option.

Le panneau Scripts peut exécuter des AppleScripts (fichiers avec l'extension `.spt`, `.as` ou `.applescript`) compilés ou non, des JavaScripts (fichiers avec l'extension `.js` ou `.jsx`), des VBScripts (fichiers avec l'extension `.vbs`) ou des programmes exécutables à partir du panneau Scripts.

Pour modifier un script affiché dans le panneau Scripts, sélectionnez-le, puis choisissez la commande Modifier le script dans le menu du panneau Scripts. Vous pouvez également maintenir la touche Option (Mac OS) ou Alt (Windows) enfoncée et cliquer deux fois sur le nom du script. Le script s'ouvre dans l'éditeur défini pour son type de fichier.

Pour ouvrir le dossier contenant un script affiché dans le panneau Scripts, sélectionnez le script, puis choisissez la commande Faire apparaître dans le Finder (Mac OS) ou Faire apparaître dans l'Explorateur (Windows). Vous pouvez également maintenir la touche Commande (Mac OS) ou les touches Ctrl+Maj (Windows) enfoncées et cliquer deux fois sur le nom du script. Le dossier contenant le script s'ouvre dans le Finder (Mac OS) ou l'Explorateur (Windows).

En règle générale, les scripts s'exécutent sous la forme d'une série d'opérations (actions), ce qui signifie que vous pouvez annuler les modifications apportées au document par le script en choisissant la commande Annuler dans le menu Edition. Vous pouvez ainsi dépanner un script, en revenant aux étapes précédentes pour chaque modification. Les scripts peuvent modifier le mode annulation et enregistrer toutes les opérations de script en une seule étape d'annulation. Ceci permet d'accélérer considérablement l'exécution des scripts. Pour plus de détails sur cette opération, consultez le *Guide des scripts d'Adobe InDesign* correspondant au langage de script de votre choix.

Pour ajouter un raccourci clavier à un script, choisissez la commande Edition > Raccourcis clavier, sélectionnez un ensemble de raccourcis modifiable dans le menu Ensemble, puis sélectionnez l'option Scripts dans le menu Zone du produit. Une liste de scripts apparaît dans le panneau Scripts. Sélectionnez un script, puis affectez-lui un raccourci clavier comme vous le feriez pour n'importe quelle autre fonction d'InDesign.

A propos des langages de script

Le langage utilisé pour rédiger des scripts dépend du système de script de votre plate-forme de travail : AppleScript pour Macintosh, VBScript pour Windows ou JavaScript pour les deux plates-formes. Bien que les langages d'écriture de script diffèrent, leur fonctionnement avec InDesign est très similaire.

Les exemples de scripts contenus dans ce document sont donnés pour tous ces langages. La conversion d'un script d'un langage à un autre est un procédé assez simple.

JavaScript

InDesign prend en charge le langage JavaScript pour la création de scripts multiplate-formes sous Mac OS et Windows. La prise en charge de JavaScript par InDesign repose sur l'implémentation de JavaScript pour Adobe appelée ExtendScript. L'interpréteur ExtendScript est conforme à la norme ECMA 262 en vigueur pour JavaScript. Toutes les fonctions de langage JavaScript 1.5 sont prises en charge. Adobe Illustrator®, Adobe Photoshop®, ainsi que d'autres produits Adobe Creative Suite® utilisent également l'interpréteur ExtendScript JavaScript.

Même si vous pouvez écrire des scripts dans d'autres versions de JavaScript, telles que Microsoft® JScript (Windows) ou JavaScript OSA de Late Night Software (Mac OS), les termes utilisés dans ces langages ne sont pas les mêmes que ceux utilisés dans ExtendScript. Les exemples ExtendScript ne fonctionnent pas dans d'autres versions JavaScript.

REMARQUE : les outils et fonctions ExtendScript sont utilisés dans plusieurs produits Adobe. Cela a mené à la consolidation de toute la documentation ExtendScript. Pour en savoir plus sur les outils JavaScript tels que le module d'interface utilisateur ScriptUI et ExtendScript Toolkit (environnement de développement JavaScript et inspecteur de modèles d'objet), reportez-vous au *Guide des outils JavaScript*.

Windows

Pour utiliser les scripts d'InDesign sous Windows, vous pouvez utiliser soit JavaScript, soit une version de Microsoft Visual Basic, telle que VBScript.

Les scripts d'apprentissage Visual Basic sont écrits en langage VBScript. Nous avons choisi le langage VBScript, car vous n'avez besoin d'aucun logiciel supplémentaire pour exécuter ou modifier des fichiers VBScript ; vous pouvez les modifier avec n'importe quel éditeur de texte et les exécuter à l'aide du panneau Scripts d'InDesign.

Les autres versions de Visual Basic incluent Visual Basic 5 Control Creation Edition (CCE), Visual Basic 6, Visual Basic .NET et Visual Basic 2005 Express Edition. Les versions de Visual Basic antérieures à Visual Basic .NET fonctionnent bien avec les scripts InDesign. La version Visual Basic .NET et les versions plus récentes ne fonctionnent pas aussi bien, car elles n'utilisent pas le type de données `Variant` qui est largement présent dans les scripts InDesign.

De nombreuses applications, par exemple Microsoft Word, Microsoft Excel, Microsoft Visio ou AutoCAD contiennent Visual Basic for Applications (VBA). Bien qu'il ne soit pas intégré au programme InDesign, vous pouvez utiliser VBA pour créer des scripts InDesign.

Pour utiliser VBScript ou Visual Basic dans les scripts InDesign sous Windows, vous devez installer InDesign depuis le compte d'un utilisateur bénéficiant des droits d'administrateur. Une fois l'installation terminée, n'importe quel utilisateur peut exécuter des scripts InDesign. Les utilisateurs bénéficiant des droits d'administrateur et d'utilisateur avec pouvoirs peuvent ajouter des scripts au panneau Scripts d'InDesign.

Mac OS

Pour utiliser les scripts InDesign sous Mac OS, vous pouvez utiliser soit JavaScript, soit AppleScript. Pour écrire des AppleScripts, vous devez posséder AppleScript version 1.6 ou supérieure, ainsi qu'un éditeur de scripts AppleScript. AppleScript, fourni avec tous les systèmes Apple®, peut être téléchargé gratuitement à partir du site Web d'Apple. L'éditeur de scripts d'Apple est fourni avec Mac OS ; il est accessible à partir des menus :

Mac OS X 10.5 Applications > AppleScript > Script Editor

Mac OS X 10.6 Applications > Utilities > AppleScript Editor

Des éditeurs de script tiers, tels que Script Debugger (de Late Night Software, <http://www.latenightsw.com>), sont également disponibles.

Langage de script à utiliser

Si vous avez déjà écrit des scripts, utilisez le langage que vous connaissez. Si vous n'avez jamais écrit de scripts ou si vous devez utiliser vos scripts sur les versions Mac OS et Windows d'InDesign, utilisez le langage JavaScript. Si vous devez communiquer avec des applications autres que des applications Adobe sur votre système, utilisez le langage approprié standard de la plate-forme (AppleScript pour Mac OS ou VBScript pour Windows).

Nous ne pouvons pas vous fournir une documentation détaillée sur AppleScript, JavaScript ou VBScript. Vous aurez donc peut-être besoin de vous procurer la documentation spécifique à un ou plusieurs de ces langages de script.

REMARQUE : vous pouvez également utiliser la plupart des langages de programmation (par exemple, Python ou C#) qui peuvent se connecter au système d'automatisation standard de la plate-forme ; cela ne fait toutefois pas partie du contenu du présent document.

Utilisation des scripts de ce document

Pour afficher ou modifier les scripts de ce document, ouvrez le fichier de script d'apprentissage correspondant (le nom de fichier figure devant chaque script) dans l'application d'édition de scripts de votre choix.

Pour exécuter un script, installez-le dans le dossier Scripts Panel (voir la section [« Installation des scripts » à la page 6](#)), puis procédez comme suit :

1. Choisissez la commande Fenêtre > Utilitaires > Scripts pour afficher le panneau Scripts.
2. Pour exécuter le script, cliquez deux fois sur son nom dans le panneau Scripts.

Pour enregistrer un script que vous avez modifié, enregistrez-le au format de fichier texte brut dans le dossier Scripts Panel (voir la section [« Installation des scripts » à la page 6](#)) avec l'extension appropriée :

AppleScript : .applescript

JavaScript : .jsx

VBScript : .vbs

REMARQUE : si vous entrez manuellement les exemples JavaScript présentés dans ce document, veillez à respecter la casse indiquée. JavaScript est sensible à la casse, et les scripts ne fonctionneront pas s'ils ne respectent pas la casse indiquée. Les exemples AppleScript et VBScript ne sont pas sensibles à la casse. Il est toutefois conseillé d'utiliser les fichiers de script fournis avec ce didacticiel.

REMARQUE : si vous copiez et collez des scripts de ce document, soyez conscient que les sauts de ligne provenant de la mise en page du document peuvent entraîner des erreurs dans vos scripts. Ces erreurs pouvant être difficiles à repérer, il est recommandé d'utiliser les scripts fournis avec ce didacticiel.

Votre premier script InDesign

Vous allez maintenant créer un script InDesign pour générer un document, ajouter un bloc de texte et entrer du texte dans ce bloc. Ce script vous permet d'effectuer les opérations suivantes :

- ▶ Etablissement de la communication avec InDesign
- ▶ Création d'un document
- ▶ Création d'un bloc de texte sur une page spécifique
- ▶ Ajout de texte à un bloc de texte

AppleScript

Lancez l'application de l'éditeur de scripts (situé dans le dossier Applications du dossier AppleScript). Entrez le script suivant (ou ouvrez le script d'apprentissage HelloWorld.applescript) :

```
tell application "Adobe InDesign CS5"
set myDocument to make document
tell page 1 of myDocument
set myTextFrame to make text frame
set geometric bounds of myTextFrame to {"6p", "6p", "24p", "24p"}
set contents of myTextFrame to "Hello World!"
end tell
end tell
```

Enregistrez le script au format de fichier texte avec l'extension `.applescript` dans le dossier Scripts Panel (voir la section « [Installation des scripts](#) » à la page 6). Pour exécuter le script, cliquez deux fois sur son nom dans le panneau Scripts ou cliquez sur Exécuter dans la fenêtre de l'éditeur de scripts.

JavaScript

Lancez l'utilitaire ExtendScript Toolkit (ou un éditeur de texte). Entrez le script suivant (ou ouvrez le script d'apprentissage `HelloWorld.jsx`):

```
var myDocument = app.documents.add();
var myTextFrame = myDocument.pages.item(0).textFrames.add();
myTextFrame.geometricBounds = ["6p", "6p", "24p", "24p"];
myTextFrame.contents = "Hello World!";
```

Enregistrez le script au format de fichier texte brut avec l'extension `.jsx` dans le dossier Scripts Panel (voir la section « [Installation des scripts](#) » à la page 6). Pour exécuter le script, cliquez deux fois sur son nom dans le panneau Scripts ou sélectionnez la commande InDesign dans le menu déroulant cible de l'application dans l'utilitaire ExtendScript Toolkit, puis cliquez sur le bouton Exécuter.

VBScript

Lancez un éditeur de texte (par exemple, Bloc-notes), puis entrez le script suivant (ou ouvrez le script d'apprentissage `HelloWorld.vbs`):

```
Set myInDesign = CreateObject("InDesign.Application")
Set myDocument = myInDesign.Documents.Add
Set myTextFrame = myDocument.Pages.Item(1).TextFrames.Add
myTextFrame.GeometricBounds = Array("6p", "6p", "24p", "24p")
myTextFrame.Contents = "Hello World!"
```

Enregistrez le script au format de fichier texte avec l'extension `.vbs` dans le dossier Scripts Panel (voir la section « [Installation des scripts](#) » à la page 6). Pour exécuter le script, cliquez deux fois sur le nom du script dans le panneau Scripts d'InDesign.

Analyse du script

Voici une analyse étape par étape des opérations réalisées par le script Hello World (dans chaque langage de script) :

1. Etablissement de la communication avec l'objet d'application InDesign

AppleScript: `tell application "Adobe InDesign CS5"`

JavaScript: L'application est désignée par le nom `app`.

VBScript: `Set myInDesign = CreateObject("InDesign.Application")`

2. Création d'un document et d'une référence au document

AppleScript: `Set myDocument to make document`

JavaScript: `Var myDocument = app.documents.add();`

VBScript: `Set myDocument = myInDesign.Documents.Add`

3. Création d'un bloc de texte sur la première page et d'une référence au bloc de texte

```
AppleScript: tell page 1 of myDocument
              set myTextFrame to make text frame
```

```
JavaScript:   var myTextFrame = myDocument.pages.item(0).textFrames.add();
```

```
VBScript:    Set myTextFrame = myDocument.Pages.Item(1).TextFrames.Add
```

4. Définition des délimitations géométriques (emplacement des bords supérieur, gauche, inférieur et droit) du bloc de texte. Dans cette étape, le script utilise des remplacements d'unités de mesures (p pour les picas) pour garantir la taille exacte du bloc de texte, sans tenir compte des unités de mesure par défaut. Les emplacements sont présentés dans une liste ou un tableau de valeurs ; chaque langage de script crée les tableaux de données d'une manière légèrement différente. Pour plus de détails sur les variables de tableau de données, voir la section [« Variables de tableau de données » à la page 15](#).

```
AppleScript: set geometric bounds of myTextFrame to {"6p", "6p", "24p", "24p"}
```

```
JavaScript:   myTextFrame.geometricBounds = ["6p", "6p", "24p", "24p"];
```

```
VBScript:    myTextFrame.GeometricBounds = Array("6p", "6p", "24p", "24p")
```

5. Ajout de texte au bloc de texte en définissant la propriété de contenu sur une chaîne

```
AppleScript: set contents of myTextFrame to "Hello World!"
```

```
JavaScript:   myTextFrame.contents = "Hello World!";
```

```
VBScript:    myTextFrame.Contents = "Hello World!"
```

Scripts et modèle d'objet InDesign

Cette section traite de la terminologie des langages de script en général, et plus particulièrement de la création de scripts InDesign.

Terminologie des scripts

Cette section traite des termes et concepts utilisés couramment dans la création de scripts.

Commentaires

Les commentaires permettent d'ajouter un descriptif à un script. Les commentaires sont ignorés pendant l'exécution d'un script (ils ne génèrent donc aucune erreur). Ils permettent de documenter les opérations d'un script (pour mémoire ou pour un autre développeur). Dans ce document, nous utilisons des commentaires pour les scripts d'apprentissage.

Pour inclure un commentaire :

- En AppleScript, insérez `--` à gauche du commentaire ou placez le commentaire entre `(*` et `*)`.
Par exemple :

```
--this is a comment
(* and so is this *)
```

- ▶ En JavaScript, insérez // à gauche du commentaire ou placez le commentaire entre /* et */ .
Par exemple :

```
// this is a comment
/* and so is this */
```

- ▶ En VBScript, tapez Rem (pour « remarque ») ou ' (guillemet droit simple) à gauche du commentaire. Saisissez le marqueur de commentaire au début d'une ligne pour mettre toute cette ligne en commentaire. Par exemple :

```
Rem this is a comment
' and so is this
```

Valeurs

Le corps d'un caractère, la position d'un bloc de texte sur une page et la couleur de contour d'un rectangle sont autant d'exemples de *valeurs* utilisées pour la création de scripts InDesign. Les valeurs sont des données utilisées par un script pendant son exécution.

Le *type* d'une valeur définit quelle sorte de donnée elle contient. Le type de valeur du contenu d'un mot, par exemple, est une chaîne de texte. De même, le type de valeur de l'interligne d'un paragraphe est un nombre. En général, les valeurs utilisées dans un script sont des nombres ou du texte. Le tableau ci-après répertorie et décrit les types de valeurs les plus couramment utilisées pour la création de scripts InDesign.

Type de valeur	Nature	Exemple
Boolean	True ou False logique	True
Integer	Nombres entiers (sans virgule). Les entiers peuvent être positifs ou négatifs. En VBScript, vous pouvez utiliser des données de type long pour les entiers. En AppleScript, vous pouvez également utiliser indifféremment des données de type long ou fixe pour les nombres entiers et les nombres réels.	14
Double (VBScript), fixed or real (AppleScript), number (JavaScript)	Nombre très précis pouvant être décimal.	13.9972

Type de valeur	Nature	Exemple
String	Série de caractères de texte. Les chaînes de texte sont présentées entre guillemets anglais.	"I am a string"
Array (VBScript, JavaScript) or list (AppleScript)	Liste de valeurs (ces valeurs pouvant être de tout type).	AppleScript: { "0p0", "0p0", "16p4", "20p6" } VBScript: Array("0p0", "0p0", "16p4", "20p6") JavaScript: ["0p0", "0p0", "16p4", "20p6"]

Conversion de valeurs d'un type à un autre

Tous les langages de script pris en charge par InDesign permettent de convertir des valeurs variables d'un type à un autre. Parmi les conversions les plus communes, figure la conversion de nombres en chaînes de texte (ce qui permet d'entrer des nombres en toutes lettres ou de les afficher dans des boîtes de dialogue) ou de chaînes de texte en nombres (ce qui permet de définir un corps ou une position de page). Reportez-vous aux exemples suivants.

AppleScript

```
--To convert from a number to a string:
set myNumber to 2
set myString to (myNumber as string)
--To convert from a string to a number:
set myString to "2"
set myNumber to (myString as integer)
--if your string contains a decimal value, use "as real" rather than "as integer"
```

JavaScript

```
//To convert from a number to a string:
myNumber = 2;
myString = myNumber + "";
//To convert from a string to an integer:
myString = "2";
myNumber = parseInt(myString);
//If your string contains a decimal value, use "parseFloat" rather than "parseInt":
myNumber = parseFloat(myString);
//You can also convert strings to numbers using the following:
myNumber = +myString;
```

VBScript

```
Rem To convert from a number to a string:
myNumber = 2
myString = cstr(myNumber)
Rem To convert from a string to an integer:
myString = "2"
myNumber = cInt(myString)
Rem If your string contains a decimal value, use "cDbl" rather than "cInt":
myNumber = cDbl(myString)
```

Variables

Une *variable* est le conteneur d'une valeur. Les variables portent ce nom car les valeurs qu'elles contiennent peuvent changer. Une variable peut contenir un nombre, une chaîne de texte ou une référence à un objet InDesign. Les variables portent un nom. Vous ferez donc référence à une variable par son nom. Pour placer une valeur dans une variable, vous devez l'*assigner* à cette variable.

Dans le premier exemple de script, plus haut, les variables `myDocument` et `myTextFrame` sont utilisées pour éviter d'indiquer la spécification entière de l'objet (par exemple, `text frame 1 of page 1 of document 1` ou `app.documents.item(0).pages.item(0).textFrames.item(0)`), à chaque référence à l'objet.

Dans tous les exemples de scripts et les scripts d'apprentissage accompagnant InDesign, les variables commencent par `my`. Vous pouvez ainsi différencier facilement les variables créées dans un script des termes du langage de script.

Affectation d'une valeur à une variable

L'affectation de valeurs ou de chaînes de texte à des variables est très simple, comme l'indique le tableau suivant :

Langage	Exemples d'affectation d'une valeur à une variable
AppleScript	<pre>set myNumber to 10 set myString to "Hello, World!" set myTextFrame to make text frame at page 1 of myDocument</pre>
JavaScript	<pre>var myNumber = 10; var myString = "Hello, World!"; var myTextFrame = myDocument.pages.item(0).textFrames.add();</pre>
VBScript	<pre>myNumber = 10 myString = "Hello, World!" Set myTextFrame = myDocument.Pages.Item(1).TextFrames.Add</pre>

REMARQUE : en JavaScript, toute variable non précédée de `var` est considérée par défaut comme globale, non liée à une fonction spécifique. `var` n'est pas obligatoire, mais il est recommandé de l'utiliser dans les scripts comprenant plusieurs fonctions. En AppleScript et VBScript, les variables sont locales sauf si elles sont définies comme globales. Autrement dit, elles ne sont plus valides en dehors de la fonction dans laquelle elles ont été créées.

Dans la mesure du possible, utilisez des noms évocateurs pour les variables, tels que `firstPage` ou `corporateLogo` plutôt que `x` ou `c`, par exemple. Votre script sera beaucoup plus lisible. L'utilisation de noms longs n'affecte en rien la vitesse d'exécution du script.

Les noms de variables doivent être composés d'un seul mot, mais vous pouvez utiliser les majuscules (pour écrire `myFirstPage`, par exemple) ou les traits de soulignement (`my_first_page`) afin de les rendre plus lisibles. Les noms de variables ne peuvent pas commencer par un chiffre, ni contenir des signes de ponctuation ou des guillemets de citation anglais.

Variables de tableau de données

AppleScript, JavaScript et VBScript prennent en charge les tableaux de données (*arrays*). Un tableau de données est un type de variable représenté par une liste de valeurs. En AppleScript, un tableau de données est appelé *liste*. Plusieurs exemples de définition de tableaux de données sont présentés ci-dessous.

Langage	Exemples de définition de tableaux de données
AppleScript	<code>set myArray to {1, 2, 3, 4}</code>
JavaScript	<code>myArray = [1, 2, 3, 4];</code>
VBScript	<code>myArray = Array(1, 2, 3, 4)</code>
Visual Basic.NET	<code>myArray = New Double (1, 2, 3, 4)</code>

Pour faire référence à un élément d'un tableau de données, vous devez utiliser son numéro dans le tableau. En VBScript et en JavaScript, le premier élément d'un tableau de données est le numéro 0 ; en AppleScript, il s'agit du numéro 1. Des exemples de référence aux éléments d'un tableau de données sont fournis dans le tableau suivant :

Langage	Exemples de référence aux éléments d'un tableau de données
AppleScript	<code>set myFirstArrayItem to item 1 of myArray</code>
JavaScript	<code>var myFirstArrayItem = myArray[0];</code>
VBScript	<code>myFirstArrayItem = myArray(0)</code>

REMARQUE : vous pouvez utiliser l'instruction `OptionBase` de Visual Basic pour attribuer le numéro 1 au premier élément d'un tableau de données. Dans les exemples de ce manuel, le premier élément d'un tableau est le numéro 0 et non pas le numéro 1. Il s'agit de l'option par défaut. Si vous avez choisi 1 pour `OptionBase`, vous devrez ajuster toutes les références des tableaux de données en conséquence dans les exemples de scripts.

Un tableau de données peut en contenir un autre, comme dans les exemples ci-dessous.

Langage	Exemples
AppleScript	<code>set myArray to {{0, 0}, {72, 72}}</code>
JavaScript	<code>var myArray = [[0,0], [72,72]];</code>
VBScript	<code>myArray = Array(Array(0,0), Array(72, 72))</code>
Visual Basic.NET	<code>myArray = New Array(New Double(0,0), NewDouble (0,0))</code>

Détermination du type de valeur d'une variable

Parfois, un script doit prendre des décisions en fonction du type de valeur d'un objet. Si vous travaillez sur un script opérant sur une sélection de texte, par exemple, vous devrez peut-être faire en sorte que ce script s'arrête si le type de sélection est un élément de page. Tous les langages de script permettent de déterminer le type d'une variable.

Apple-Script

```
-- Given a variable of unknown type, "myMysteryVariable"...
set myType to class of myMysteryVariable
--myType will be an AppleScript type (e.g., rectangle)
```

JavaScript `//Given a variable of unknown type, "myMysteryVariable"...
myType = myMysteryVariable.constructor.name;
//myType will be a string corresponding to the JavaScript type (e.g., "Rectangle")`

VBScript `Rem Given a variable of unknown type, "myMysteryVariable"...
myType = TypeName(myMysteryVariable)
Rem myType will be a string corresponding to the variable type (e.g., "Rectangle")`

Opérateurs

Les opérateurs utilisent des variables ou des valeurs pour exécuter des calculs (addition, soustraction, multiplication et division) et renvoyer une valeur. Par exemple :

```
MyWidth/2
```

Renvoie une valeur égale à la moitié du contenu de la variable `myWidth`.

Vous pouvez également utiliser des opérateurs pour effectuer des comparaisons (est égal à (=), est différent de (<>), est supérieur à (>) ou est inférieur à (<)). Par exemple :

```
MyWidth > myHeight
```

Renvoie la valeur `true` (ou 1) si `myWidth` est supérieur à `myHeight` ou `false` (0) si ce n'est pas le cas.

Tous les langages de script fournissent des opérateurs « utilitaires » supplémentaires. En AppleScript et en VBScript, l'opérateur `&` permet de concaténer (joindre) deux chaînes de texte :

```
"Pride " & "and Prejudice"
```

Renvoie la chaîne de texte suivante :

```
"Pride and Prejudice"
```

Pour joindre deux chaînes de texte en JavaScript, utilisez le signe plus (+), comme suit :

```
"Pride " + "and Prejudice"  
//returns the string: "Pride and Prejudice"
```

Instructions conditionnelles

« Si l'objet sélectionné est un rectangle, définissez son contour sur une épaisseur de 12 points. » Ceci est un exemple d'*instruction conditionnelle*. Les instructions conditionnelles permettent au script d'évaluer des conditions (telles que la couleur de l'objet sélectionné, le nombre de pages de la composition ou la date) et d'agir en fonction du résultat. Une instruction conditionnelle commence presque toujours par `if`.

REMARQUE : les instructions conditionnelles effectuent souvent des comparaisons logiques. En AppleScript et en VBScript, le signe égal (=) permet de comparer des objets. En JavaScript, le signe égal affecte une valeur à une variable ; utilisez un signe égal double (==) pour la comparaison des objets.

Structures de contrôle

Si vous pouviez parler à InDesign, vous pourriez lui dire notamment « Répète vingt fois la procédure suivante. ». En terminologie des scripts, cette instruction est appelée *structure de contrôle*. Les structures de contrôle gèrent les processus répétitifs ou *boucles*. Une boucle répète une action continuellement, avec ou sans changement entre les instances (ou *itérations*) de la boucle, jusqu'à ce qu'une condition spécifique soit remplie. Les structures de contrôle commencent en général par `repeat` (en AppleScript) ou par `for` (en JavaScript et VBScript).

Fonctions et gestionnaires

Les *fonctions* (en VBScript ou JavaScript) ou les *gestionnaires* (en AppleScript) sont des modules auxquels vous pouvez faire référence dans un script. Généralement, vous envoyez une valeur ou une série de valeurs à une fonction (ou à un gestionnaire) et obtenez une ou plusieurs autres valeurs. Le code des fonctions et des gestionnaires permet tout simplement d'éviter de saisir les mêmes lignes de code de façon répétée dans un script.

En AppleScript, les gestionnaires commencent par `on`. En JavaScript et en VBScript, les fonctions commencent par `function`.

Présentation du modèle d'objet InDesign

Lorsque vous pensez à InDesign et aux documents InDesign, vous organisez certainement le programme et ses composants dans votre esprit. Vous savez que les paragraphes sont contenus dans des blocs de texte qui apparaissent sur une page, qu'une page fait partie d'une planche, et que plusieurs planches composent un document. Les documents contiennent des couleurs, des styles, des calques et des planches types. A mesure que vous créez les mises en page, vous pensez intuitivement à leur ordre.

InDesign « appréhende » le contenu d'un document de la même manière. Un document contient des pages, qui à leur tour contiennent des éléments de page (blocs de texte, rectangles, ellipses, et ainsi de suite). Les blocs de texte contiennent des caractères, des mots, des paragraphes et des blocs ancrés ; les blocs graphiques contiennent des images, des fichiers EPS ou des fichiers PDF ; les groupes contiennent d'autres éléments de page. Tous ces éléments sont les *objets* qui composent une publication InDesign et qui sont utilisés pour rédiger les scripts InDesign.

Les objets de votre publication sont organisés dans un ordre spécifique : les blocs figurent sur les pages, lesquelles figurent dans le document, qui à son tour figure dans l'objet d'application InDesign. Les termes *modèle d'objet* et *hiérarchie* font référence à cette structure. Il est important de comprendre la définition du modèle d'objet pour bien identifier l'objet que vous voulez utiliser, et votre meilleur guide pour créer des scripts dans InDesign est votre connaissance de l'application elle-même.

Les objets possèdent des *propriétés* (attributs). Par exemple, les propriétés d'un objet texte incluent la police utilisée pour la mise en forme du texte, le corps et l'interligne appliqués au texte.

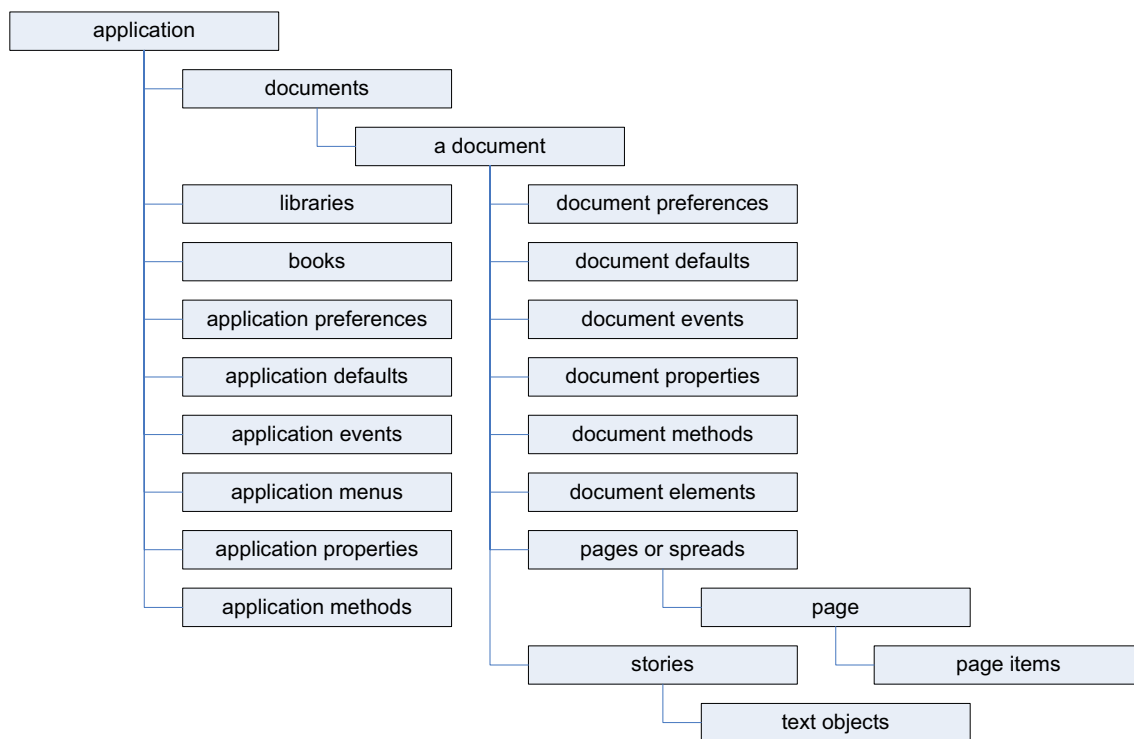
Les propriétés possèdent des valeurs. Ainsi, le corps des caractères du texte peut être exprimé par un nombre (en points) ou par la chaîne « Auto » pour l'interligne automatique. La propriété de couleur de fond du texte peut être définie sur une couleur, un dégradé, une encre mélangée ou une nuance.

Les propriétés peuvent être en *lecture/écriture* ou en *lecture seule*. Vous pouvez définir les propriétés en lecture/écriture sur d'autres valeurs, mais pas les propriétés en lecture seule.

Les objets possèdent également des *méthodes* (les verbes de création de scripts ou les opérations qu'un objet peut exécuter). Par exemple, l'objet de document contient des méthodes d'impression, d'exportation et d'enregistrement.

Les méthodes possèdent des *paramètres* ou des valeurs définissant l'effet de la méthode. Par exemple, la méthode d'importation (ou de placement) d'un document contient un paramètre définissant le fichier à importer. Les paramètres sont *obligatoires* ou *facultatifs*, selon la méthode.

L'illustration suivante est une vue d'ensemble du modèle d'objet InDesign. Ce diagramme ne représente pas la liste complète des objets disponibles dans les scripts InDesign. Il s'agit simplement d'une structure conceptuelle permettant de comprendre la relation entre les types d'objets.



Les objets figurant dans le diagramme sont expliqués dans le tableau suivant :

Terme	Signification
Application	InDesign.
Application defaults	Les paramètres par défaut de l'application, tels que les couleurs, les styles de paragraphe et les styles d'objet. Les valeurs par défaut de l'application concernent tous les nouveaux documents.
Application events	Les événements qui se produisent à mesure qu'un utilisateur travaille dans l'application ou qu'un script s'exécute. Les événements sont générés par l'ouverture, la fermeture ou l'enregistrement d'un document ou par la sélection d'une commande de menu. Les événements peuvent déclencher des scripts.
Application menus	Les menus, sous-menus et menus contextuels qui s'affichent dans l'interface utilisateur d'InDesign. Vous pouvez attacher des scripts aux menus pour exécuter des commandes.
Application methods	Les opérations que l'application peut effectuer, notamment rechercher et remplacer du texte, copier la sélection, créer des documents ou ouvrir des bibliothèques.
Application preferences	Par exemple, des préférences de texte, des préférences d'exportation PDF ou des préférences de document. De nombreux objets de préférences sont également disponibles au niveau du document. Tout comme dans l'interface utilisateur, les préférences d'application s'appliquent aux nouveaux documents. Les préférences de document modifient les paramètres d'un document spécifique.

Terme	Signification
Application properties	Les propriétés de l'application, notamment le chemin complet de l'application, ses paramètres régionaux et le nom d'utilisateur.
Books	Une collection de livres ouverts.
Document	Un document InDesign.
Document defaults	Les paramètres par défaut du document, tels que les couleurs, les styles de paragraphe et les paramètres par défaut de mise en forme de texte.
Document elements	Par exemple, les articles, les graphiques importés et les pages d'un document. L'illustration précédant ce tableau indique des pages et des articles, car ces objets sont d'importants conteneurs d'autres objets. Toutefois, les éléments de document incluent également des rectangles, des ovales (ellipses), des groupes, des éléments XML, ainsi que tout autre type d'objet que vous pouvez importer ou créer.
Document events	Les événements qui se produisent au niveau du document, par exemple, l'importation de texte (voir « Application events » plus haut dans le tableau).
Document methods	Les opérations que le document peut effectuer, notamment fermer, imprimer ou exporter un document.
Document preferences	Les préférences d'un document, telles que les préférences de repères, d'affichage ou de document.
Document properties	Par exemple, le nom de fichier du document, le nombre de pages ou l'emplacement de l'origine (point zéro).
Documents	Une collection de documents ouverts.
Libraries	Une collection de bibliothèques ouvertes.
Page	Une seule page dans un document InDesign.
Page items	Tout objet que vous pouvez créer ou placer sur une page. Il existe de nombreux types d'éléments de page, tels que des blocs de texte, des rectangles, des lignes graphiques ou des groupes.
Pages or spreads	Les pages ou les planches dans un document InDesign.
Stories	Le texte dans un document InDesign.
Text objects	Les caractères, mots, lignes, paragraphes et colonnes de texte sont des exemples d'objets texte dans un article InDesign.

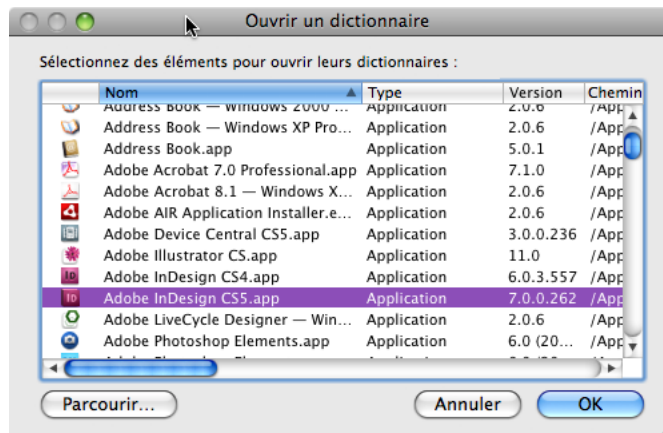
Consultation du modèle d'objet InDesign

Vous pouvez consulter le modèle d'objet InDesign depuis votre application d'édition de scripts. Toutes les informations de référence sur les objets et leurs propriétés et méthodes sont stockées dans le modèle et peuvent être affichées.

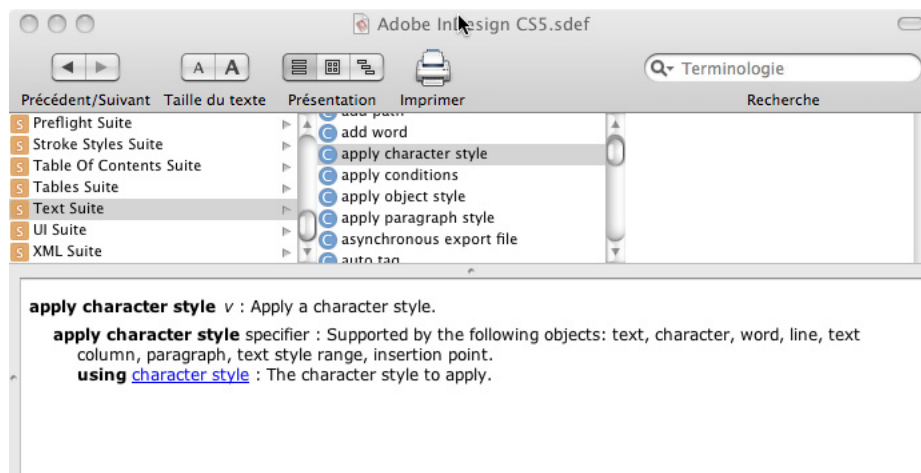
AppleScript

Pour afficher le dictionnaire AppleScript d'InDesign :

1. Lancez InDesign.
2. Lancez l'éditeur de scripts Apple.
3. Dans l'éditeur de scripts, choisissez la commande **Fichier > Ouvrir un dictionnaire**. L'éditeur de scripts affiche une liste des applications prenant en charge les scripts.



4. Sélectionnez votre copie d'InDesign, puis cliquez sur **Sélectionner**. L'éditeur de scripts affiche une liste des suites (collections d'objets associés) d'InDesign.

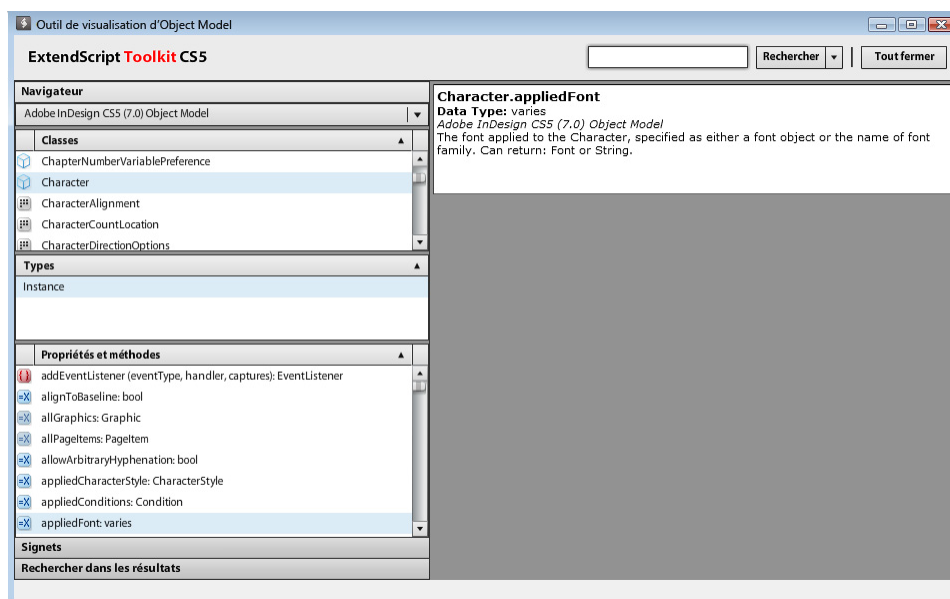


5. Sélectionnez une suite pour afficher les objets et méthodes (commandes) qu'elle contient. Sélectionnez un objet pour afficher ses propriétés.

JavaScript

Pour afficher le modèle d'objet InDesign dans l'utilitaire ExtendScript Toolkit :

1. Lancez l'utilitaire ExtendScript Toolkit.
2. Choisissez la commande Aide > Outil de visualisation d'Object Model.
3. Sélectionnez Adobe InDesign CS5 Object Model dans le panneau Navigateur. L'utilitaire ExtendScript Toolkit charge le fichier d'aide associé au modèle d'objet et affiche une liste d'objets de script InDesign dans le panneau Classes.
4. Dans la liste d'objets du panneau Classes, sélectionnez l'objet à afficher, puis cliquez sur la propriété ou la méthode à détailler dans la liste Propriétés et méthodes. L'utilitaire ExtendScript Toolkit affiche d'autres informations sur la propriété ou la méthode sélectionnée.



Pour plus de détails sur l'outil de visualisation de modèle d'objet de l'utilitaire ExtendScript Toolkit, voir le *Guide des outils JavaScript*.

REMARQUE : le nom de classe (objet) indiqué dans l'outil de visualisation de modèle d'objet utilise une casse différente de celle appliquée aux instances de ce même objet dans un script. JavaScript étant sensible à la casse, vous devez saisir le terme avec la casse appropriée dans un script. Par exemple, le terme « Documents » apparaît dans la liste des classes, mais vous devez utiliser « app.documents » pour faire référence à cette classe dans un script. La casse adaptée apparaît toujours dans le panneau Propriétés et méthodes de l'outil de visualisation de modèle d'objet lorsque le parent de l'objet est sélectionné dans le panneau Classes.

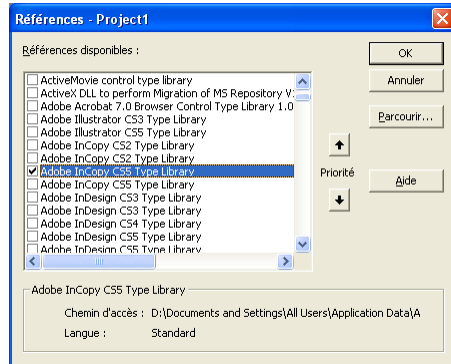
VBScript

Pour afficher le modèle d'objet InDesign, vous devez disposer d'un éditeur/débugueur VBScript ou d'une version de Visual Basic, ou encore d'une application incorporant Visual Basic for Applications.

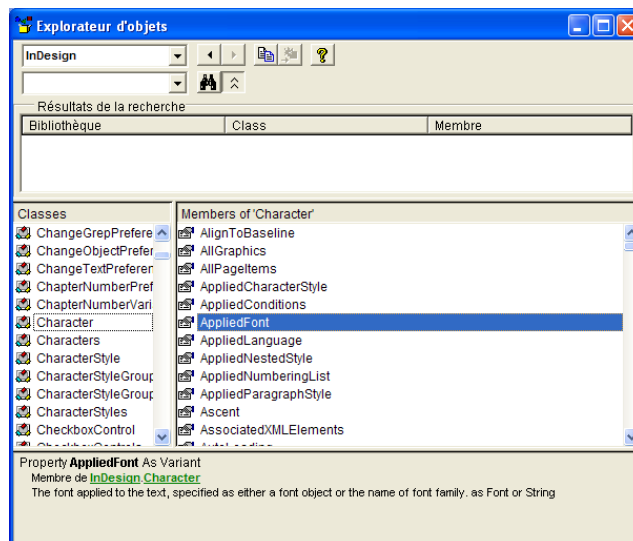
Visual Basic 6

Pour afficher le modèle d'objet à l'aide de Visual Basic 6 :

1. Créez un projet Visual Basic, puis choisissez la commande **Projet > Références**. Visual Basic affiche la boîte de dialogue **Références**.



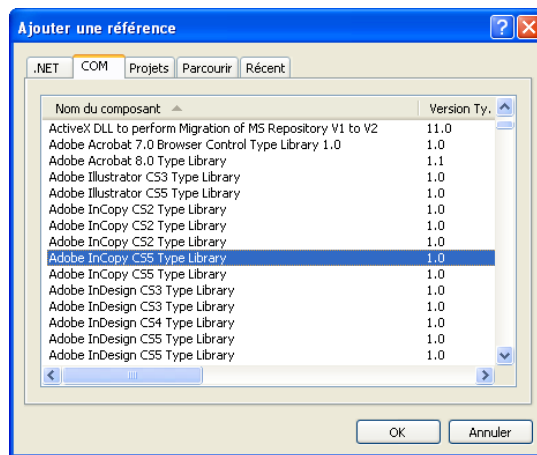
2. Dans la liste de références disponibles, sélectionnez l'option **Adobe InDesign CS5 Type Library**, puis cliquez sur **OK**. Si la bibliothèque n'apparaît pas dans la liste des références disponibles, cliquez sur **Parcourir**, puis recherchez et sélectionnez le fichier **Resources for Visual Basic.tlb**, habituellement situé dans le répertoire `C:\Documents and Settings\nom d'utilisateur\Application Data\Adobe\InDesign\Version 7.0\langue\Scripting Support\`. Si nécessaire, recherchez le fichier. Après avoir localisé le fichier, cliquez sur le bouton **Ouvrir** pour ajouter la référence à votre projet.
3. Choisissez la commande **Affichage > Explorateur d'objets**. Visual Basic affiche la boîte de dialogue **Explorateur d'objets**.
4. Choisissez « **InDesign** » dans la liste de bibliothèques ouvertes du menu **Projet/bibliothèque**. Visual Basic affiche les objets composant le modèle d'objet InDesign.
5. Cliquez sur une classe d'objet. Visual Basic affiche les propriétés et les méthodes de l'objet. Pour plus de détails sur une propriété ou une méthode, sélectionnez-la. Visual Basic affiche sa définition au bas de la fenêtre **Explorateur d'objets**.



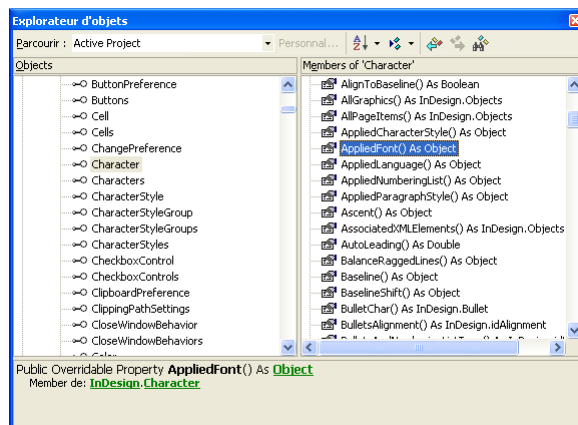
Visual Basic.NET

Pour afficher le modèle d'objet à l'aide de Visual Basic.NET :

1. Créez un projet Visual Basic, puis choisissez la commande **Projet > Ajouter la référence**. Visual Basic affiche la boîte de dialogue d'ajout de références.
2. Sélectionnez l'onglet **COM**.
3. Dans la liste de références disponibles, sélectionnez l'option **Adobe InDesign CS5 Type Library**, puis cliquez sur le bouton **Sélectionner**. Visual Basic.NET ajoute la référence à la liste de composants sélectionnés. Si la bibliothèque n'apparaît pas dans la liste des références disponibles, cliquez sur **Parcourir**, puis recherchez et sélectionnez le fichier `Resources for Visual Basic.tlb`, habituellement situé dans le répertoire `C:\Documents and Settings\nom d'utilisateur\Application Data\Adobe\InDesign\Version 7.0\langue\Scripting Support\`. Après avoir localisé le fichier, cliquez sur le bouton **Ouvrir** pour ajouter la référence à votre projet.



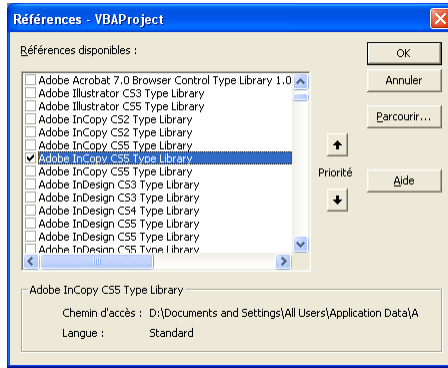
4. Cliquez sur le bouton **OK**.
5. Choisissez la commande **Affichage > Explorateur d'objets**. Visual Basic affiche l'onglet **Explorateur d'objets**.
6. Choisissez l'option `interop.indesign` dans la liste de bibliothèques ouvertes de la fenêtre **Objets**. Visual Basic.NET affiche les objets composant le modèle d'objet InDesign.
7. Cliquez sur une classe d'objet. Visual Basic.NET affiche les propriétés et les méthodes de l'objet. Pour plus de détails sur une propriété ou une méthode, sélectionnez-la. Visual Basic.NET affiche sa définition au bas de la fenêtre **Explorateur d'objets**.



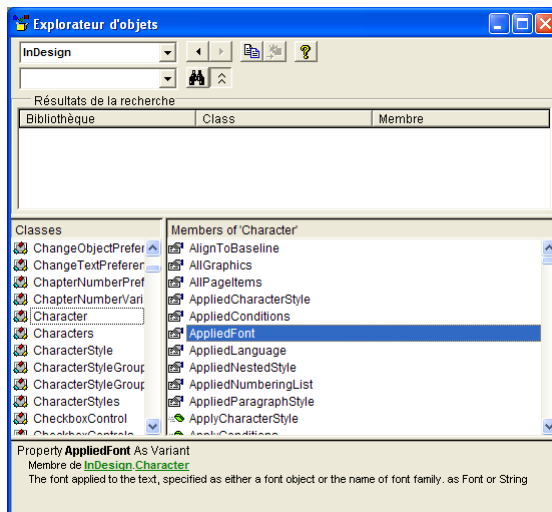
Visual Basic for Applications

Pour afficher le modèle d'objet à l'aide de Visual Basic for Applications depuis Microsoft Excel :

1. Lancez Excel.
2. Choisissez la commande Outils > Macro > Visual Basic Editor. Excel affiche la fenêtre de l'éditeur Visual Basic.
3. Choisissez la commande Outils > Références. L'éditeur Visual Basic affiche la boîte de dialogue Références.



4. Dans la liste de références disponibles, sélectionnez Adobe InDesign CS5 Type Library, puis cliquez sur OK. Visual Basic ajoute la référence à la liste de composants sélectionnés. Si la bibliothèque n'apparaît pas dans la liste des références disponibles, cliquez sur Parcourir, puis recherchez et sélectionnez le fichier Resources for Visual Basic.tlb, habituellement situé dans le répertoire ~:\Documents and Settings\\Application Data\Adobe\InDesign\Version 7.0\\Scripting Support\. Après avoir localisé le fichier, cliquez sur le bouton Ouvrir pour ajouter la référence à votre projet.
5. Choisissez la commande Affichage > Explorateur d'objets. L'éditeur Visual Basic affiche la fenêtre Explorateur d'objets.
6. Choisissez l'option InDesign dans le menu déroulant Bibliothèques. L'éditeur Visual Basic affiche une liste des objets de la bibliothèque d'objets InDesign.
7. Cliquez sur un nom d'objet. L'éditeur Visual Basic affiche les propriétés et les méthodes de l'objet. Pour plus de détails sur une propriété ou une méthode, sélectionnez-la. L'éditeur Visual Basic affiche sa définition au bas de la fenêtre Explorateur d'objets.



Mesures et positionnement

Dans InDesign, tous les éléments et objets sont placés sur une page en fonction des mesures que vous spécifiez. Il est donc très utile de connaître le fonctionnement du système de coordonnées d'InDesign et les unités de mesure qu'il utilise.

Coordonnées

Comme tout autre programme de mise en page et de dessin, InDesign utilise une géométrie bidimensionnelle simple pour définir la position des objets sur une page ou une planche. La composante horizontale d'une paire de coordonnées est appelée *x* et sa composante verticale, *y*. Vous pouvez voir ces coordonnées dans le panneau Transformation ou Contrôle lorsqu'un objet est sélectionné avec l'outil Sélection. Comme dans l'interface utilisateur d'InDesign, les coordonnées sont mesurées par rapport à l'emplacement actuel de l'origine (point zéro) de la règle.

Il existe une différence entre les coordonnées utilisées dans InDesign et le système de coordonnées géométrique traditionnel. Sur l'axe vertical (ou *y*) d'InDesign, les coordonnées *au-dessous* de l'origine sont des nombres positifs, et celles situées *au-dessus*, des nombres négatifs.

REMARQUE : les coordonnées de l'emplacement d'un point de tracé sont renvoyées par InDesign dans l'ordre *x, y*. Lorsque vous définissez l'emplacement d'un point de tracé, InDesign attend les coordonnées dans le même ordre. InDesign renvoie certaines coordonnées dans un ordre différent, et celles-ci doivent lui être fournies dans le même ordre. Les délimitations géométriques et les délimitations visibles sont des tableaux de données contenant quatre coordonnées. Ces coordonnées définissent dans l'ordre les bords supérieur, gauche, inférieur et droit du cadre de sélection de l'objet (soit *y1, x1, y2, x2*).

Utilisation des unités de mesure

Pour envoyer des valeurs de mesure à InDesign, vous pouvez utiliser soit des nombres (14.65, par exemple), soit des chaînes de mesure (« 1p7.1 », par exemple). Si vous envoyez des nombres, InDesign utilise les unités de mesure actives de la composition. Si vous envoyez des chaînes de mesure (voir le tableau ci-dessous), InDesign utilise les unités de mesure spécifiées dans chaque chaîne.

InDesign renvoie les coordonnées et autres valeurs de mesure en utilisant les unités de mesure actives de la composition. Dans certains cas, ces unités ne sont pas représentées sous la même forme que les valeurs de mesure figurant dans le panneau Transformation d'InDesign. Si le système de mesure actif utilise les picas, par exemple, InDesign renvoie des valeurs fractionnaires sous forme de nombres décimaux, au lieu d'utiliser les conventions « picas et points » du panneau Transformation. Ainsi, « 1p6 » est renvoyé sous la forme « 1.5 ». InDesign procède de cette manière, car le système de scripts ne peut pas exécuter d'opération arithmétique avec des chaînes de mesure ; si vous additionnez « 0p3.5 » et « 13p4 », par exemple, vous allez obtenir une erreur de script ; en revanche, si vous additionnez 0,2916 et 13,333 (les mêmes mesures après leur conversion), le résultat sera correct.

Si votre script dépend de l'addition, la soustraction, la multiplication ou la division de valeurs de mesure spécifiques, il est préférable de définir les unités de mesure correspondantes au début du script. A la fin du script, vous pourrez rétablir les unités de mesure initialement utilisées avant l'exécution du script. Vous pouvez également utiliser des remplacements d'unités de mesures, comme dans beaucoup de nos exemples de scripts. Un remplacement d'unité de mesure est une chaîne contenant un caractère spécial, comme dans les exemples ci-dessous.

Remplacement	Signification	Exemple
c	Cicéros (le c peut être suivi de didots, si nécessaire)	1.4c
cm	Centimètres	.635cm
i (ou in)	Pouces	.25i

Remplacement	Signification	Exemple
mm	Millimètres	6.35mm
p	Picas (le p peut être suivi de points, si nécessaire)	1p6
pt	Points	18pt

Ajout de caractéristiques à « Hello World »

Nous allons maintenant créer un script qui modifie la composition « Hello World » créée avec notre premier script. Ce second script illustre les opérations suivantes :

- ▶ Obtention du document actif
- ▶ Utilisation d'une fonction (ou d'un *gestionnaire* dans AppleScript)
- ▶ Obtention des dimensions et des marges des pages du document actif
- ▶ Redimensionnement d'un bloc de texte
- ▶ Changement de la mise en forme du texte dans son bloc

AppleScript

Vérifiez que le document « Hello World » est bien ouvert ; le script suivant fait en effet référence à des objets que nous avons créés dans ce script. Si vous avez fermé le document sans l'enregistrer, il suffit de réexécuter le script `HelloWorld.applescript` pour créer un nouveau document.

Ouvrez le script d'apprentissage `ImprovedHelloWorld.applescript` ou procédez comme suit pour créer le script :

1. Dans l'éditeur de script, choisissez la commande Fichier > Nouveau pour créer un script.
2. Entrez le code ci-dessous.

```
--Improved "Hello World"
tell application "Adobe InDesign CS5"
  --Get a reference to a font.
  try
    --Enter the name of a font on your system, if necessary.
    set myFont to font "Helvetica"
  end try
  --Get the active document and assign
  --the result to the variable "myDocument."
  set myDocument to document 1
  tell myDocument
    --Use the handler "myGetBounds" to get the bounds of the
    --"live area" inside the margins of page 1.
    set myBounds to my myGetBounds(myDocument, page 1)
```

```

    tell text frame 1 of page 1
      --Resize the text frame to match the page margins.
      set geometric bounds to myBounds
      tell paragraph 1
        --Change the font, size, and paragraph alignment.
        try
          set applied font to myFont
        end try
        set point size to 72
        set justification to center align
      end tell
    end tell
  end tell
end tell
--myGetBounds is a handler that returns the bounds
--of the "live area" of a page.
on myGetBounds(myDocument, myPage)
tell application "Adobe InDesign CS5"
  tell document preferences of myDocument
    myPageWidth to page width
    set myPageHeight to page height
  end tell
  tell margin preferences of myPage
    if side of myPage is left hand then
      set myX2 to left
      set myX1 to right
    else
      set myX1 to left
      set myX2 to right
    end if
    set myY1 to top
    set myY2 to bottom
  end tell
  set myX2 to myPageWidth - myX2
  set myY2 to myPageHeight - myY2
  return {myY1, myX1, myY2, myX2}
end tell
end myGetBounds

```

3. Enregistrez le script dans un fichier de texte brut avec l'extension `.applescript` dans le dossier Scripts Panel (voir la section [« Installation des scripts » à la page 6](#)).

Après avoir ouvert ou créé le fichier de script, vous pouvez exécuter le script depuis l'éditeur de scripts ou à partir du panneau Scripts d'InDesign.

JavaScript

Vérifiez que le document « Hello World » est bien ouvert ; le script suivant fait en effet référence à des objets que nous avons créés dans ce script. Si vous avez fermé le document sans l'enregistrer, il suffit de réexécuter le script `HelloWorld.jsx` pour créer un nouveau document.

Ouvrez le script d'apprentissage `ImprovedHelloWorld.jsx` ou procédez comme suit pour créer le script :

1. Entrez le code JavaScript ci-dessous dans un nouveau fichier texte.

```
//Improved Hello World!
//Enter the name of a font on your system, if necessary.
try{
    myFont = app.fonts.item("Arial");
}
catch (myError){};
var myDocument = app.documents.item(0);
with(myDocument){
    var myPage = pages.item(0);
    var myBounds = myGetBounds(myPage,myDocument);
    with(myDocument.pages.item(0)){
        //Get a reference to the text frame.
        var myTextFrame = textFrames.item(0);
        //Change the size of the text frame.
        myTextFrame.geometricBounds = myBounds;
        var myParagraph = myTextFrame.paragraphs.item(0);
        myParagraph.appliedFont = myFont;
        myParagraph.justification = Justification.centerAlign;
        myParagraph.pointSize = 48;
    }
}
//myGetBounds is a function that returns the bounds
//of the "live area" of a page.
function myGetBounds(myDocument, myPage){
    var myPageWidth = myDocument.documentPreferences.pageWidth;
    var myPageHeight = myDocument.documentPreferences.pageHeight
    if(myPage.side == PageSideOptions.leftHand){
        var myX2 = myPage.marginPreferences.left;
        var myX1 = myPage.marginPreferences.right;
    }
    else{
        var myX1 = myPage.marginPreferences.left;
        var myX2 = myPage.marginPreferences.right;
    }
    var myY1 = myPage.marginPreferences.top;
    var myX2 = myPageWidth - myX2;
    var myY2 = myPageHeight - myPage.marginPreferences.bottom;
    return [myY1, myX1, myY2, myX2];
}
```

2. Enregistrez le script dans un fichier de texte brut avec l'extension `.jsx` dans le dossier `Scripts Panel` (voir la section [« Installation des scripts » à la page 6](#)).

Après avoir ouvert ou créé le fichier de script, vous pouvez exécuter le script depuis l'utilitaire `ExtendScript Toolkit` ou à partir du panneau `Scripts d'InDesign`.

VBScript

Vérifiez que le document « Hello World » est bien ouvert ; le script suivant fait en effet référence à des objets que nous avons créés dans ce script. Si vous avez fermé le document sans l'enregistrer, il suffit de réexécuter le script `HelloWorld.vbs` pour créer un nouveau document.

Ouvrez le script d'apprentissage `ImprovedHelloWorld.vbs` ou procédez comme suit pour créer le script :

1. Lancez un éditeur de texte (Bloc-notes, par exemple).
2. Entrez le code ci-dessous.

```
Set myInDesign = CreateObject("InDesign.Application")
Rem Enter the name of a font on your system, if necessary.
Set myFont = myInDesign.Fonts.Item("Arial")
Set myDocument = myInDesign.ActiveDocument
Set myPage = myDocument.Pages.Item(1)
Rem Get page width and page height using the function "myGetBounds".
myBounds = myGetBounds(myDocument, myPage)
Set myTextFrame = myPage.TextFrames.Item(1)
Rem Resize the text frame to match the publication margins.
myTextFrame.GeometricBounds = myBounds
Set myParagraph = myTextFrame.Paragraphs.Item(1)
Rem Change the font, size, and alignment.
If TypeName(myFont) <> "Nothing" Then
    myParagraph.AppliedFont = myFont
End If
myParagraph.PointSize = 48
myParagraph.Justification = idJustification.idCenterAlign
Rem myGetBounds is a function that returns the bounds
Rem of the "live area" of a page.
Function myGetBounds(myDocument, myPage)
    myPageWidth = myDocument.DocumentPreferences.PageWidth
    myPageHeight = myDocument.DocumentPreferences.PageHeight
    If myPage.Side = idPageSideOptions.idLeftHand Then
        myX2 = myPage.MarginPreferences.Left
        myX1 = myPage.MarginPreferences.Right
    Else
        myX1 = myPage.MarginPreferences.Left
        myX2 = myPage.MarginPreferences.Right
    End If
    myY1 = myPage.marginPreferences.Top
    myX2 = myPageWidth - myX2
    myY2 = myPageHeight - myPage.MarginPreferences.Bottom
    myGetBounds = Array(myY1, myX1, myY2, myX2)
End Function
```

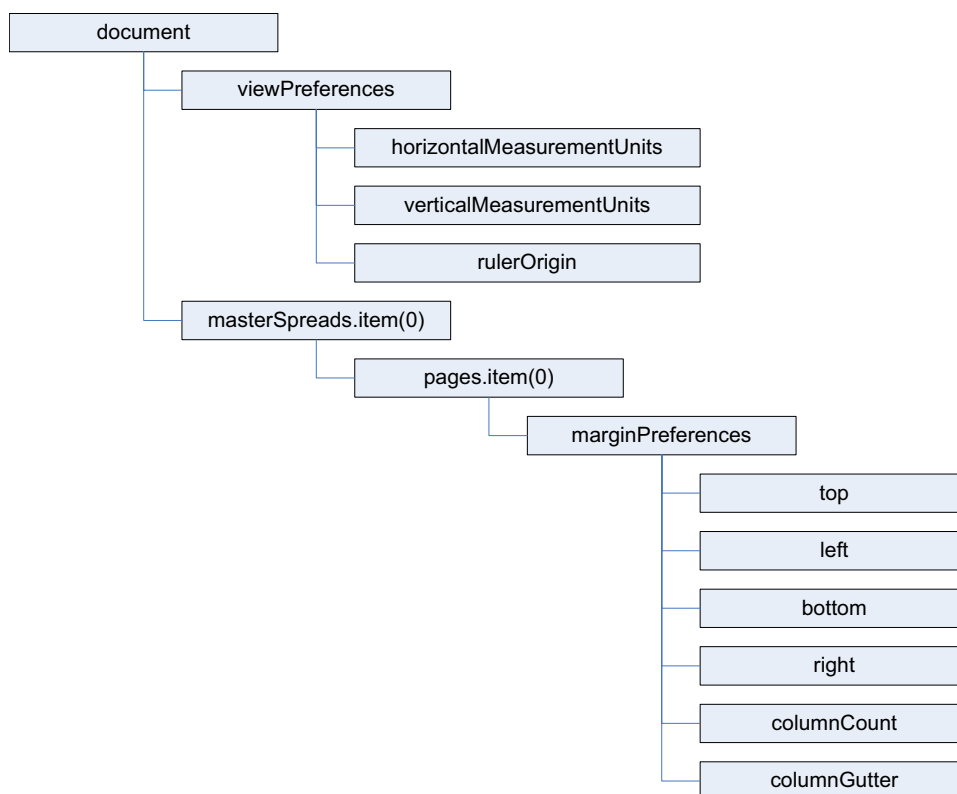
3. Enregistrez le texte dans un fichier de texte brut avec l'extension `.vbs` dans le dossier Scripts (voir la section [« Installation des scripts » à la page 6](#)).

Après avoir ouvert ou créé le fichier de script, vous pouvez exécuter le script à partir du panneau Scripts d'InDesign.

Construction d'un document

Bien entendu, le script « Hello World! » ne sera pas d'une grande utilité pour votre travail quotidien, mais il vous a permis de vous familiariser avec les notions de base de la création de scripts InDesign. Dans la section suivante, nous allons vous présenter un script plus élaboré qui implique des techniques de création de scripts que vous êtes susceptibles d'utiliser dans vos propres scripts.

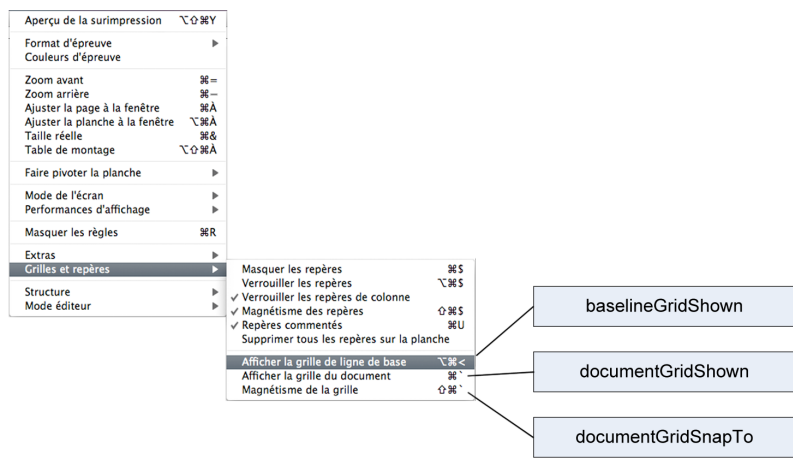
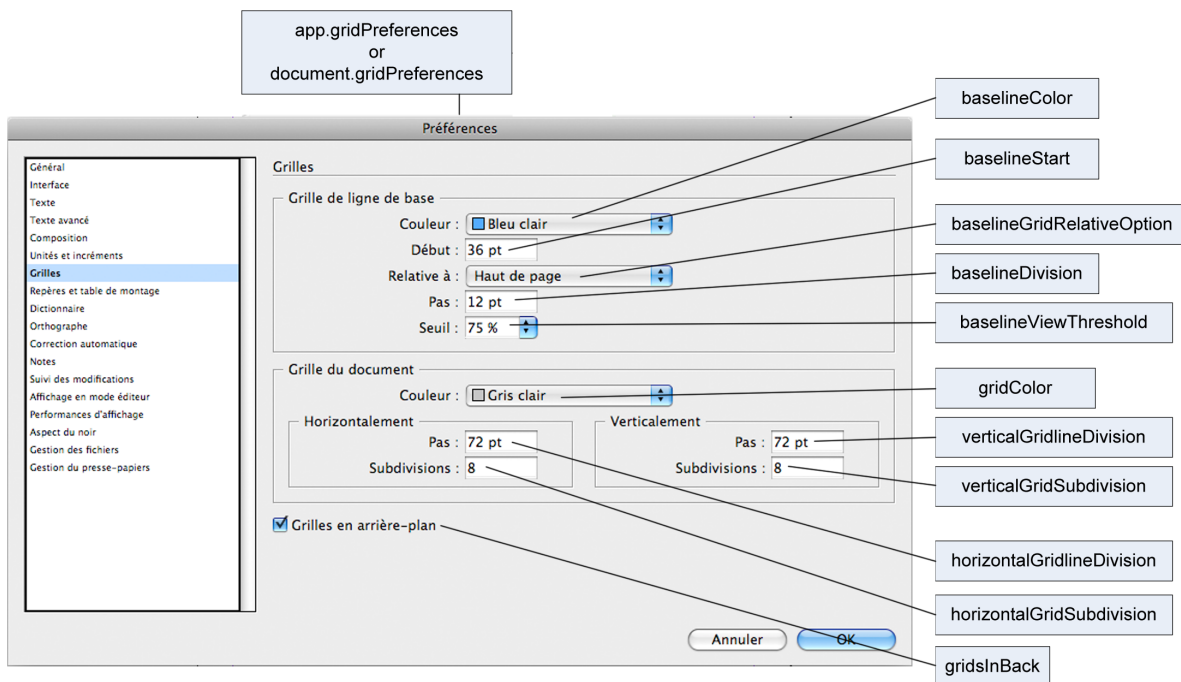
Vous pouvez utiliser un script InDesign à tout moment dans votre processus de production. Toutefois, nous commencerons par créer un script débutant au même point que vous : nous allons créer un nouveau document, définir des marges de page, puis définir et appliquer des gabarits. Le schéma ci-après représente les objets que nous allons utiliser.



Dans cette section, nous allons étudier le script d'apprentissage DocumentTemplate. Nous diviserons le script en une série de blocs, chaque bloc représentant une zone ou une opération spécifique de création de scripts InDesign.

REMARQUE : l'illustration ci-dessus utilise la version JavaScript des termes de création de scripts. Pour le langage AppleScript, vous ajoutez des espaces entre les mots (`view preferences`, au lieu de `viewPreferences`) ; pour le langage VBScript, vous utilisez un numéro d'élément commençant à 1, au lieu de 0 (`masterSpreads.item(1)`, au lieu de `masterSpreads.item(0)`). En dehors de ces légères différences d'espacement et de casse, et des quelques mots propres à chaque langage, les termes utilisés dans les différents langages sont identiques.

Les objets dans le modèle d'objet correspondent en général aux noms des commandes dans l'interface utilisateur, comme indiqué dans le schéma ci-après (lequel utilise aussi la forme JavaScript des termes de création de scripts).



Dans les sections suivantes, nous allons aborder les différentes zones fonctionnelles du script DocumentConstruction. Ouvrez ce script dans l'éditeur de scripts du langage de votre choix pour pouvoir suivre la procédure.

Configuration des unités de mesure et des marges des planches types

L'extrait de script suivant indique comment créer un document et définir les marges de la première planche type.

AppleScript

```
tell application "Adobe InDesign CS5"
  --Create a new document.
  set myDocument to make document
  --Set the measurement units and ruler origin.
  set horizontal measurement units of view preferences to points
  set vertical measurement units of view preferences to points
  set ruler origin of view preferences to page origin
  --Get a reference to the first master spread.
  set myMasterSpread to master spread 1 of myDocument
  --Get a reference to the margin preferences of the first page in the master spread.
  tell margin preferences of page 1 of myMasterSpread
    --Now set up the page margins and columns.
    set left to 84
    set top to 70
    set right to 70
    set bottom to 78
    set column count to 3
    set column gutter to 14
  end tell
  --Page margins and columns for the right-hand page.
  tell margin preferences of page 2 of myMasterSpread
    set left to 84
    set top to 70
    set right to 70
    set bottom to 78
    set column count to 3
    set column gutter to 14
  end tell
end tell
```

JavaScript

```
//Create a new document.
var myDocument = app.documents.add();
//Set the measurement units and ruler origin.
myDocument.viewPreferences.horizontalMeasurementUnits = MeasurementUnits.points;
myDocument.viewPreferences.verticalMeasurementUnits = MeasurementUnits.points;
myDocument.viewPreferences.rulerOrigin = RulerOrigin.pageOrigin;
//Get a reference to the first master spread.
var myMasterSpread = myDocument.masterSpreads.item(0);
//Get a reference to the margin preferences of the first page in the master spread.
var myMarginPreferences = myMasterSpread.pages.item(0).marginPreferences;
//Now set up the page margins and columns.
myMarginPreferences.left = 84;
myMarginPreferences.top = 70;
myMarginPreferences.right = 70;
myMarginPreferences.bottom = 78;
myMarginPreferences.columnCount = 3;
myMarginPreferences.columnGutter = 14;
```

```
//Page margins and columns for the right-hand page.
var myMarginPreferences = myMasterSpread.pages.item(1).marginPreferences;
myMarginPreferences.left = 84;
myMarginPreferences.top = 70;
myMarginPreferences.right = 70;
myMarginPreferences.bottom = 78;
myMarginPreferences.columnCount = 3;
myMarginPreferences.columnGutter = 14;
```

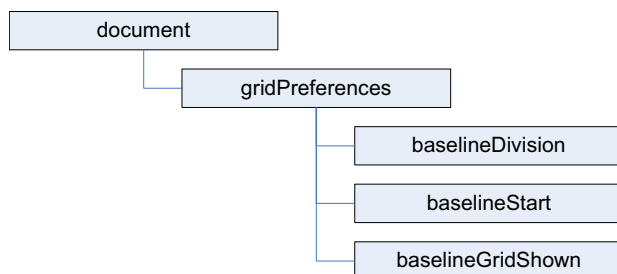
VBScript

Entrez le code suivant dans votre éditeur de script ou de texte, ou ouvrez le script d'apprentissage DocumentConstruction.vbs.

```
Set myInDesign = CreateObject("InDesign.Application")
Rem Create a new document.
Set myDocument = myInDesign.Documents.Add()
Rem Set the measurement units and ruler origin.
myDocument.ViewPreferences.HorizontalMeasurementUnits = idMeasurementUnits.idPoints
myDocument.ViewPreferences.VerticalMeasurementUnits = idMeasurementUnits.idPoints
myDocument.ViewPreferences.RulerOrigin = idRulerOrigin.idPageOrigin
Rem Get a reference to the first master spread.
Set myMasterSpread = myDocument.MasterSpreads.Item(1)
Rem Get a reference to the margin preferences of the first page in the master spread.
Set myMarginPreferences = myMasterSpread.Pages.Item(1).MarginPreferences
Rem Now set up the page margins and columns.
myMarginPreferences.Left = 84
myMarginPreferences.Top = 70
myMarginPreferences.Right = 70
myMarginPreferences.Bottom = 78
myMarginPreferences.ColumnCount = 3
myMarginPreferences.ColumnGutter = 14
Rem Page margins and columns for the right-hand page.
Set myMarginPreferences = myMasterSpread.Pages.Item(2).MarginPreferences
myMarginPreferences.Left = 84
myMarginPreferences.Top = 70
myMarginPreferences.Right = 70
myMarginPreferences.Bottom = 78
myMarginPreferences.ColumnCount = 3
myMarginPreferences.ColumnGutter = 14
```

Ajout d'une grille de ligne de base

Maintenant que la planche type est configurée, nous pouvons ajouter une grille de ligne de base. Le schéma ci-après présente la relation entre les objets avec lesquels nous allons travailler (il utilise la forme JavaScript des termes de création de scripts).



AppleScript

```
set myGridPreferences to grid preferences
set baseline division of myGridPreferences to 14
set baseline start of myGridPreferences to 70
set baseline grid shown of myGridPreferences to true
```

JavaScript

```
var myGridPreferences = myDocument.gridPreferences;
myGridPreferences.baselineDivision = 14;
myGridPreferences.baselineStart = 70;
myGridPreferences.baselineGridShown = true;
```

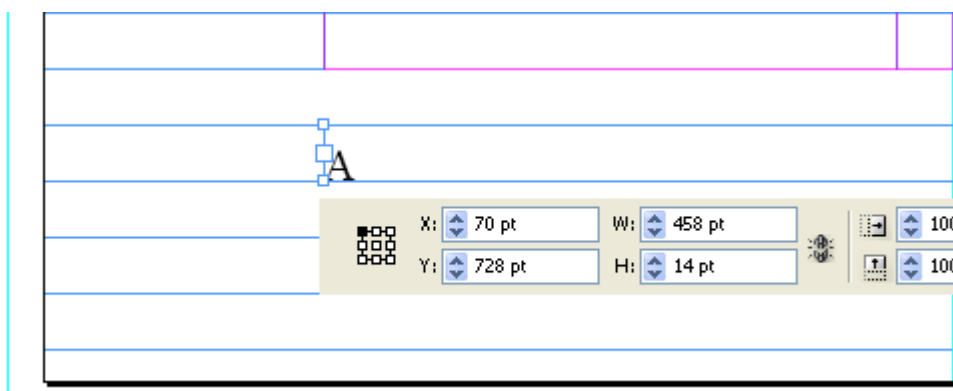
VBScript

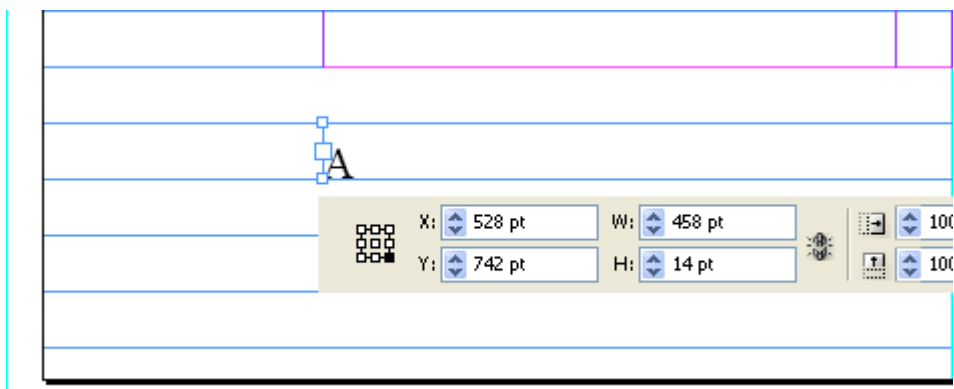
```
Set myGridPreferences = myDocument.GridPreferences
myGridPreferences.BaselineDivision = 14
myGridPreferences.BaselineStart = 70
myGridPreferences.BaselineGridShown = True
```

Ajout d'éléments de gabarit

Nous allons maintenant ajouter deux blocs de texte aux pages de gabarit. Ces blocs sont destinés à contenir le caractère spécial de numérotation de page automatique et seront positionnés au bas de la page.

Dans l'exemple « Hello World », nous avons créé un bloc de texte et spécifié sa position et sa taille à l'aide de la propriété de délimitation géométrique (un tableau de données contenant les coordonnées des bords supérieur, gauche, inférieur et droit du bloc). Ces coordonnées correspondent aux coins du bloc, comme ils apparaîtraient dans le panneau Contrôle. Les délimitations géométriques sont les suivantes : haut = 728, gauche = 70, bas = 742 et droite = 528, comme indiqué dans les deux illustrations ci-après.





AppleScript

```

set myLeftPage to page 1 of myMasterSpread
set myRightPage to page 2 of myMasterSpread
tell myLeftPage
    set myLeftFooter to make text frame
    set geometric bounds of myLeftFooter to {728, 70, 742, 528}
    set first baseline offset of text frame preferences of myLeftFooter to leading
offset
    set contents of myLeftFooter to auto page number
    set point size of character 1 of parent story of myLeftFooter to 11
    set leading of character 1 of myLeftFooter to 14
end tell
tell myRightPage
    set myRightFooter to make text frame
    set geometric bounds of myRightFooter to {728, 84, 742, 542}
    set first baseline offset of text frame preferences of myRightFooter to leading
offset
    set contents of myRightFooter to auto page number
    set point size of character 1 of parent story of myRightFooter to 11
    set leading of character 1 of myRightFooter to 14
    set justification of character 1 of myRightFooter to right align
end tell
  
```

JavaScript

```

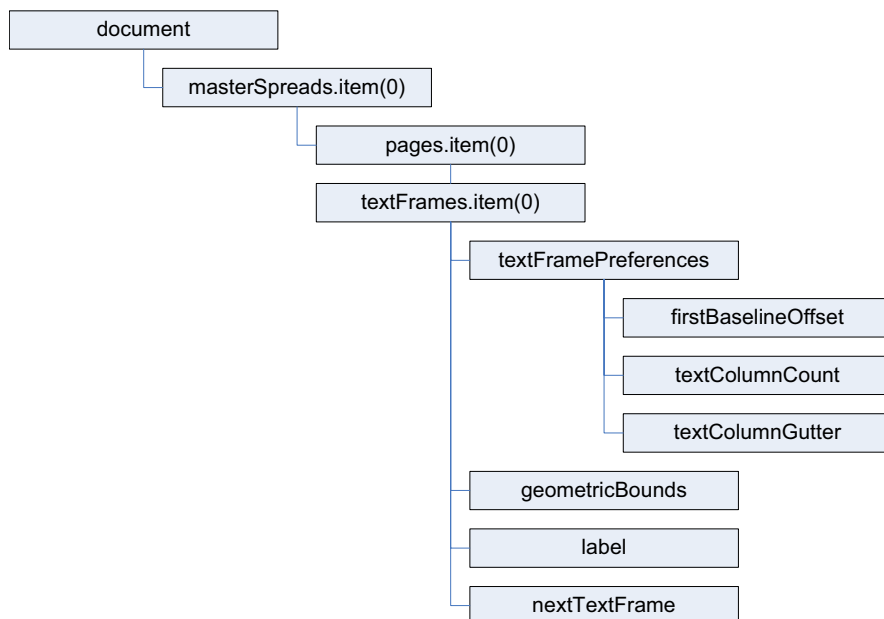
var myMasterSpread = myDocument.masterSpreads.item(0);
var myLeftPage = myMasterSpread.pages.item(0);
var myRightPage = myMasterSpread.pages.item(1);
var myLeftFooter = myLeftPage.textFrames.add();
myLeftFooter.geometricBounds = [728, 70, 742, 528];
myLeftFooter.textFramePreferences.firstBaselineOffset = FirstBaseline.leadingOffset;
myLeftFooter.contents = SpecialCharacters.autoPageNumber;
myLeftFooter.parentStory.characters.item(0).pointSize = 11;
myLeftFooter.parentStory.characters.item(0).leading = 14;
var myRightFooter = myRightPage.textFrames.add();
myRightFooter.geometricBounds = [728, 84, 742, 542];
myRightFooter.textFramePreferences.firstBaselineOffset = FirstBaseline.leadingOffset;
myRightFooter.contents = SpecialCharacters.autoPageNumber;
myRightFooter.parentStory.characters.item(0).pointSize = 11;
myRightFooter.parentStory.characters.item(0).leading = 14;
myRightFooter.parentStory.characters.item(0).justification =
Justification.rightAlign;
  
```

VBScript

```
Set myMasterSpread = myDocument.MasterSpreads.Item(1)
Set myLeftPage = myMasterSpread.Pages.Item(1)
Set myRightPage = myMasterSpread.Pages.Item(2)
Set myLeftFooter = myLeftPage.TextFrames.Add
myLeftFooter.GeometricBounds = Array(728, 70, 742, 528)
myLeftFooter.TextFramePreferences.FirstBaselineOffset =
idFirstBaseline.idLeadingOffset
myLeftFooter.Contents = idSpecialCharacters.idAutoPageNumber
myLeftFooter.ParentStory.Characters.Item(1).PointSize = 11
myLeftFooter.ParentStory.Characters.Item(1).Leading = 14
Set myRightFooter = myRightPage.TextFrames.Add()
myRightFooter.GeometricBounds = Array(728, 84, 742, 542)
myRightFooter.TextFramePreferences.FirstBaselineOffset =
idFirstBaseline.idLeadingOffset
myRightFooter.Contents = idSpecialCharacters.idAutoPageNumber
myRightFooter.ParentStory.Characters.Item(1).PointSize = 11
myRightFooter.ParentStory.Characters.Item(1).Leading = 14
myRightFooter.ParentStory.Characters.Item(1).Justification =
idJustification.idRightAlign
```

Ajout de blocs de texte types

Nous allons à présent ajouter des blocs de texte types. Le schéma ci-après présente les objets et les propriétés que nous utiliserons (sous la forme JavaScript des termes de création de scripts).



AppleScript

```
tell myLeftPage
    set myLeftTextFrame to make text frame
    set geometric bounds of myLeftTextFrame to {70, 70, 714, 528}
    set first baseline offset of text frame preferences of myLeftTextFrame to leading
offset
    set text column count of text frame preferences of myLeftTextFrame to 3
    set text column gutter of text frame preferences of myLeftTextFrame to 14
    --Add a label to make the frame easier to find later on.
    set label of myLeftTextFrame to "BodyTextFrame"
end tell
tell myRightPage
    set myRightTextFrame to make text frame
    set geometric bounds of myRightTextFrame to {70, 84, 714, 542}
    set first baseline offset of text frame preferences of myRightTextFrame to leading
offset
    set text column count of text frame preferences of myRightTextFrame to 3
    set text column gutter of text frame preferences of myRightTextFrame to 14
    --Add a label to make the frame easier to find later on.
    set label of myRightTextFrame to "BodyTextFrame"
end tell
--Link the two frames using the next text frame property.
set next text frame of myLeftTextFrame to myRightTextFrame
```

JavaScript

```
var myLeftPage = myMasterSpread.pages.item(0);
var myRightPage = myMasterSpread.pages.item(1);
var myLeftTextFrame = myLeftPage.textFrames.add();
myLeftTextFrame.geometricBounds = [70, 70, 714, 528];
myLeftTextFrame.textFramePreferences.firstBaselineOffset =
FirstBaseline.leadingOffset;
myLeftTextFrame.textFramePreferences.textColumnCount = 3;
myLeftTextFrame.textFramePreferences.textColumnGutter = 14;
//Add a label to make the frame easier to find later on.
myLeftTextFrame.label = "BodyTextFrame";
var myRightTextFrame = myRightPage.textFrames.add();
myRightTextFrame.geometricBounds = [70, 84, 714, 542];
myRightTextFrame.textFramePreferences.firstBaselineOffset =
FirstBaseline.leadingOffset;
myRightTextFrame.textFramePreferences.textColumnCount = 3;
myRightTextFrame.textFramePreferences.textColumnGutter = 14;
//Add a label to make the frame easier to find later on.
myRightTextFrame.label = "BodyTextFrame";
//Link the two frames using the nextTextFrame property.
myLeftTextFrame.nextTextFrame = myRightTextFrame;
```

VBScript

```
Set myLeftTextFrame = myLeftPage.TextFrames.Add
myLeftTextFrame.GeometricBounds = Array(70, 70, 714, 528)
myLeftTextFrame.TextFramePreferences.FirstBaselineOffset =
idFirstBaseline.idLeadingOffset
myLeftTextFrame.TextFramePreferences.TextColumnCount = 3
myLeftTextFrame.TextFramePreferences.TextColumnGutter = 14
```

```

Rem Add a label to make the frame easier to find later on.
myLeftTextFrame.Label = "BodyTextFrame"
Set myRightTextFrame = myRightPage.TextFrames.Add
myRightTextFrame.GeometricBounds = Array(70, 84, 714, 542)
myRightTextFrame.TextFramePreferences.FirstBaselineOffset =
idFirstBaseline.idLeadingOffset
myRightTextFrame.TextFramePreferences.TextColumnCount = 3
myRightTextFrame.TextFramePreferences.TextColumnGutter = 14
Rem Add a label to make the frame easier to find later on.
myRightTextFrame.Label = "BodyTextFrame"
Rem Link the two frames using the nextTextFrame property.
myLeftTextFrame.NextTextFrame = myRightTextFrame

```

Remplacement des éléments de gabarit et ajout de texte

Nous allons maintenant remplacer l'un des blocs de texte types créé précédemment et ajouter du texte.

AppleScript

```

tell text frame 1 of page 2 of master spread 1 of myDocument
  set myTextFrame to override destination page page 1 of myDocument
end tell
--Add text by setting the contents of an insertion point to a string.
--In AppleScript, "return" is a return character.
set contents of insertion point 1 of myTextFrame to "Headline!" & return

```

JavaScript

```

var myTextFrame =
myDocument.masterSpreads.item(0).pages.item(1).textFrames.item(0).override(myDocument
.pages.item(0));
//Add text by setting the contents of an insertion point to a string.
//In JavaScript, "\r" is a return character.
myTextFrame.insertionPoints.item(0).contents = "Headline!\r";

```

VBScript

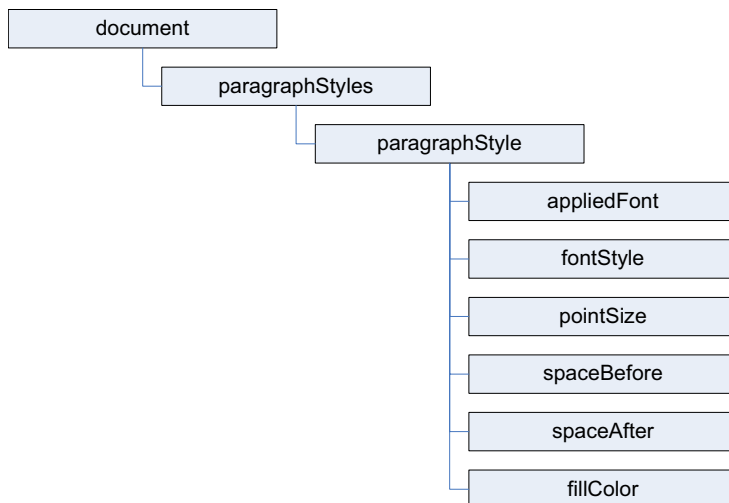
```

Set myTextFrame =
myDocument.MasterSpreads.Item(1).Pages.Item(2).TextFrames.Item(1).Override(myDocument
.Pages.Item(1))
Rem Add text by setting the contents of an insertion point to a string.
Rem In VBScript, vbCrLf is a return character.
myTextFrame.InsertionPoints.Item(1).Contents = "Headline!" & vbCrLf

```

Ajout et application d'un style de paragraphe

Nous proposons d'ajouter un style de paragraphe au titre pour le mettre en forme. Pour cela, nous devons créer le style de paragraphe. Le schéma ci-après indique les objets et les propriétés qui seront utilisés (encore une fois, la forme JavaScript des termes de création de scripts est utilisée dans le texte).



AppleScript

```
--First, check to see if the paragraph style already exists.
try
    set myParagraphStyle to paragraph style "Heading 1" of myDocument
on error
    --The paragraph style did not exist, so create it.
    tell myDocument
        set myParagraphStyle to make paragraph style with properties {name:"Heading 1"}
    end tell
end try
--We'll need to create a color. Check to see if the color already exists.
try
    set myColor to color "Red" of myDocument
on error
    --The color did not exist, so create it.
    tell myDocument
        set myColor to make color with properties {name:"Red", model:process,
            color value:{0, 100, 100, 0}}
    end tell
```

```

end try
--Now set the formatting of the paragraph style.
try
    set applied font of myParagraphStyle to "Arial"
    set font style of myParagraphStyle to "Bold"
end try
set point size of myParagraphStyle to 24
set space after of myParagraphStyle to 24
set space before of myParagraphStyle to 24
set fill color of myParagraphStyle to color "Red" of myDocument
--Apply the style to the paragraph.
tell paragraph 1 of myTextFrame to apply paragraph style using myParagraphStyle with
clearing overrides
--You could also use:
--set applied paragraph style of paragraph 1 of myTextFrame to myParagraphStyle

```

JavaScript

```

var myParagraphStyle = myDocument.paragraphStyles.item("Heading 1");
try {
    var myName = myParagraphStyle.name;
}
catch (myError){
    //The paragraph style did not exist, so create it.
    myParagraphStyle = myDocument.paragraphStyles.add({name:"Heading 1"});
}
//We'll need to create a color. Check to see if the color already exists.
var myColor = myDocument.colors.item("Red");
try {
    myName = myColor.name;
}
catch (myError){
    //The color did not exist, so create it.
    myColor = myDocument.colors.add({name:"Red", model:ColorModel.process,
    colorValue:[0,100,100,0]});
}

//Now set the formatting of the paragraph style.
myParagraphStyle.appliedFont = "Arial";
myParagraphStyle.fontStyle = "Bold";
myParagraphStyle.pointSize = 24;
myParagraphStyle.spaceAfter = 24;
myParagraphStyle.spaceBefore = 24;
myParagraphStyle.fillColor = myDocument.colors.item("Red");
//Apply the style to the paragraph.
myDocument.pages.item(0).textFrames.item(0).paragraphs.item(0).applyParagraphStyle(
myParagraphStyle, true);
//You could also use:
//myDocument.pages.item(0).textFrames.item(0).paragraphs.item(0).appliedParagraphStyl
e = myParagraphStyle;

```

VBScript

```

Rem First, check to see if the paragraph style already exists.
Rem to do this, we disable error checking:
On Error Resume Next
Set myParagraphStyle = myDocument.ParagraphStyles.Item("Heading 1")
Rem if an error occurred on the previous line, then the paragraph
Rem style did not exist.
If Error.Number <> 0 Then
    Set myParagraphStyle = myDocument.ParagraphStyles.Add
    myParagraphStyle.Name = "Heading 1"
    Error.Clear
End If
Rem We'll need to create a color. Check to see if the color already exists.
Set myColor = myDocument.Colors.Item("Red")

If Error.Number <> 0 Then
    Set myColor = myDocument.Colors.Add
    myColor.Name = "Red"
    myColor.Model = idColorModel.idProcess
    myColor.colorValue = Array(0, 100, 100, 0)
    Error.Clear
End If
Rem Resume normal error handling.
On Error GoTo 0
Rem Now set the formatting of the paragraph style.
myParagraphStyle.AppliedFont = "Arial"
myParagraphStyle.FontStyle = "Bold"
myParagraphStyle.PointSize = 24
myParagraphStyle.SpaceAfter = 24
myParagraphStyle.SpaceBefore = 24
myParagraphStyle.FillColor = myDocument.Colors.Item("Red")
Rem Apply the style to the paragraph.
myDocument.Pages.Item(1).TextFrames.Item(1).Paragraphs.Item(1).ApplyParagraphStyle
myParagraphStyle, True
Rem You could also use:
Rem
myDocument.pages.item(1).textFrames.item(1).paragraphs.item(1).appliedParagraphStyle
= myParagraphStyle

```

Importation d'un fichier texte

Nous sommes prêts à importer un fichier texte. Nous allons ajouter le texte à la suite du titre, dans le premier bloc de texte de la première page. Le script ouvre une boîte de dialogue dans laquelle vous pouvez sélectionner le fichier texte à importer.

AppleScript

```

--Display a standard open file dialog box to select a text file.
set myTextFile to choose file ("Choose a text file")
--If a text file was selected, and if you didn't press Cancel,
--place the text file at the first insertion point after the headline.
if myTextFile is not "" then
    tell insertion point -1 of myTextFrame to place myTextFile
end if

```

JavaScript

```
//Display a standard open file dialog box to select a text file.
var myTextFile = File.openDialog("Choose a text file");
//If a text file was selected, and if you didn't press Cancel,
//place the text file at the first insertion point after the headline.
if((myTextFile != "") && (myTextFile != null)) {
    myTextFrame.insertionPoints.item(-1).place(myTextFile);
}
```

VBScript

```
Rem Display a standard open file dialog box to select a text file.
Rem VBScript does not have the ability to do this, so we'll use
Rem a JavaScript to get a file name. We'll run the JavaScript using
Rem InDesign's DoScript feature.
Rem Disable normal error handling.
On Error Resume Next
Rem Create a JavaScript as a string.
myJavaScriptString = "var myTextFile = File.openDialog("Choose a text
file");myTextFile.fsName;"
Rem Run the JavaScript using DoScript.
myFileName = myInDesign.DoScript(myJavaScriptString, idScriptLanguage.idJavascript)
If Error.Number = 0 Then
    Rem Place the text file at the end of the text frame.
    myTextFrame.InsertionPoints.Item(-1).Place myFileName
    Error.Clear
End If
Rem Restore normal error handling.
On Error GoTo 0
```

Importation d'une image

Vous importez une image tout comme s'il s'agissait d'un fichier texte. Le script ouvre une boîte de dialogue dans laquelle vous pouvez sélectionner l'image à importer. Lors de l'importation de l'image, InDesign renvoie une référence à l'image elle-même, plutôt qu'au bloc qui la contient. Pour obtenir une référence au bloc, utilisez la propriété `parent` de l'image. Une fois cette référence obtenue, vous pouvez appliquer un style d'objet au bloc.

AppleScript

```

--Display a standard open file dialog box to select a graphic file.
set myGraphicFile to choose file "Choose graphic file."
--If a graphic file was selected, and if you didn't press Cancel,
--place the graphic file on the page.
if myGraphicFile is not "" then
    set myGraphic to place myGraphicFile on page 1 of myDocument
    --Since you can place multiple graphics at once, the place method
    --returns an array. To get the graphic you placed, get the first
    --item in the array.
    set myGraphic to item 1 of myGraphic
    --Create an object style to apply to the graphic frame.
    try
        set myObjectStyle to object style "GraphicFrame" of myDocument on error
        --The object style did not exist, so create it.
        tell myDocument
            set myObjectStyle to make object style with properties{name:"GraphicFrame"}
        end tell
    end try
    set enable stroke of myObjectStyle to true
    set stroke weight of myObjectStyle to 3
    set stroke type of myObjectStyle to stroke style "Solid" of myDocument
    set stroke color of myObjectStyle to color "Red" of myDocument
    --The frame containing the graphic is the parent of the graphic.
    set myFrame to parent of myGraphic
    tell myFrame to apply object style using myObjectStyle

    --Resize the frame to a specific size.
    set geometric bounds of myFrame to {0, 0, 144, 144}
    --Fit the graphic to the frame proportionally.
    fit myFrame given proportionally
    --Next, fit frame to the resized graphic.
    fit myFrame given frame to content
    set myBounds to geometric bounds of myFrame
    set myGraphicWidth to (item 4 of myBounds) - (item 2 of myBounds)
    --Move the graphic frame.
    set myPageWidth to page width of document preferences of myDocument
    set myMarginPreferences to margin preferences of page 1 of myDocument
    set myTopMargin to top of myMarginPreferences
    move myFrame to {myPageWidth - myGraphicWidth, myTopMargin}
    --Apply a text wrap to the graphic frame.
    set text wrap mode of text wrap preferences of myFrame to bounding box text wrap
    set text wrap offset of text wrap preferences of myFrame to {24, 12, 24, 12}
end if
end tell

```

JavaScript

```
//Display a standard open file dialog box to select a graphic file.
var myGraphicFile = File.openDialog("Choose a graphic file");
//If a graphic file was selected, and if you didn't press Cancel,
//place the graphic file on the page.
if((myGraphicFile != "") && (myGraphicFile != null)){
    var myGraphic = myDocument.pages.item(0).place(myGraphicFile);
    //Since you can place multiple graphics at once, the place method
    //returns an array. To get the graphic you placed, get the first
    //item in the array (JavaScript arrays start with item 0).
    myGraphic = myGraphic[0];
    //Create an object style to apply to the graphic frame.
    var myObjectStyle = myDocument.objectStyles.item("GraphicFrame");
    try {
        var myName = myObjectStyle.name;
    }
    catch (myError){
        //The object style did not exist, so create it.
        myObjectStyle = myDocument.objectStyles.add({name:"GraphicFrame"});
    }
    myObjectStyle.enableStroke = true;
    myObjectStyle.strokeWeight = 3;
    myObjectStyle.strokeType = myDocument.strokeStyles.item("Solid");
    myObjectStyle.strokeColor = myDocument.colors.item("Red");
    //The frame containing the graphic is the parent of the graphic.
    var myFrame = myGraphic.parent;
    myFrame.applyObjectStyle(myObjectStyle, true);
    //Resize the frame to a specific size.
    myFrame.geometricBounds = [0,0,144,144];
    //Fit the graphic to the frame proportionally.
    myFrame.fit(FitOptions.proportionally);
    //Next, fit frame to the resized graphic.
    myFrame.fit(FitOptions.frameToContent);
    var myBounds = myFrame.geometricBounds;
    var myGraphicWidth = myBounds[3]-myBounds[1];

    //Move the graphic frame.
    var myPageWidth = myDocument.documentPreferences.pageWidth;
    var myTopMargin = myDocument.pages.item(0).marginPreferences.top;
    myFrame.move([myPageWidth-myGraphicWidth, myTopMargin]);
    //Apply a text wrap to the graphic frame.
    myFrame.textWrapPreferences.textWrapMode = TextWrapModes.BOUNDING_BOX_TEXT_WRAP;
    myFrame.textWrapPreferences.textWrapOffset = [24, 12, 24, 12];
}
```

VBScript

```

Rem create an object style
On Error Resume Next
Set myObjectStyle = myDocument.ObjectStyles.Item("GraphicFrame")
If Error.Number <> 0 Then
    Set myObjectStyle = myDocument.ObjectStyles.Add
    myObjectStyle.Name = "GraphicFrame"
    Error.Clear
End If
On Error GoTo 0
myObjectStyle.EnableStroke = True
myObjectStyle.StrokeWeight = 3
myObjectStyle.StrokeType = myDocument.StrokeStyles.Item("Solid")
myObjectStyle.StrokeColor = myDocument.Colors.Item("Red")
Rem Again, we'll use a JavaScript to get a file name.
Rem Disable normal error handling.
On Error Resume Next
Rem Create a JavaScript as a string.
myJavaScriptString = "var myTextFile = File.openDialog("Choose a graphic
file");myTextFile.fsName;"
Rem Run the JavaScript using DoScript.
myGraphicFileName = myInDesign.DoScript(myJavaScriptString,
idScriptLanguage.idJavascript)
If Error.Number = 0 Then
    On Error GoTo 0
    Set myGraphic = myDocument.Pages.Item(1).Place(myGraphicFileName)
    Rem Since you can place multiple graphics at once, the place method
    Rem returns an object collection. To get the graphic you placed, get the first
    Rem item in the collection.
    Set myGraphic = myGraphic.Item(1)
    Rem Create an object style to apply to the graphic frame.
    Rem The frame containing the graphic is the parent of the graphic.
    Set myFrame = myGraphic.Parent
    myFrame.ApplyObjectStyle myObjectStyle, True
    Rem Resize the frame to a specific size.
    myFrame.GeometricBounds = Array(0, 0, 144, 144)
    Rem Fit the graphic to the frame proportionally.
    myFrame.Fit idFitOptions.idProportionally
    Rem Next, fit frame to the resized graphic.
    myFrame.Fit idFitOptions.idFrameToContent
    myBounds = myFrame.GeometricBounds
    myGraphicWidth = myBounds(3) - myBounds(1)
    Rem Move the graphic frame.
    myPageWidth = myDocument.DocumentPreferences.PageWidth
    myTopMargin = myDocument.Pages.Item(1).MarginPreferences.Top
    myFrame.Move Array(myPageWidth - myGraphicWidth, myTopMargin)
    Rem Apply a text wrap to the graphic frame.
    myFrame.TextWrapPreferences.TextWrapMode =
    idTextWrapModes.idBoundingBoxTextWrap
    myFrame.TextWrapPreferences.TextWrapOffset = Array(24, 12, 24, 12)
End If

```

Perfectionnement

Désormais, vous savez comment créer un document, configurer des éléments de gabarit, entrer du texte, importer du texte, créer et appliquer des styles de paragraphe, créer et appliquer des styles d'objet, importer des images, adapter des images aux blocs et appliquer un habillage de texte. Le document que vous avez créé est un document simple, mais qui vous a permis de maîtriser les notions de base de la création de scripts InDesign. Pour chaque exemple, nous avons créé des objets, défini des propriétés d'objet et utilisé des méthodes d'objet.

Consultez le *Guide des scripts d'Adobe InDesign* pour passer à l'étape suivante d'apprentissage de création de scripts InDesign. Ce guide vous propose des didacticiels plus avancés traitant de la construction de documents, la mise en forme du texte, la recherche et le remplacement de texte, la création d'interfaces utilisateur, l'ajout de commandes de menu, ainsi que l'utilisation du langage XML et des règles XML.

Pour en savoir plus sur la création de scripts InDesign, vous pouvez également consulter le forum des utilisateurs consacré aux scripts InDesign (en anglais), à l'adresse suivante : <http://www.adobeforums.com>. Dans ce forum, les créateurs de scripts peuvent poser des questions, envoyer des réponses et partager leurs dernières compositions. Le forum contient plusieurs centaines d'exemples de scripts.

Consultez également la page d'accueil des scripts InDesign, à l'adresse <http://www.adobe.com/products/indesign/scripting/index.html>, pour obtenir d'autres informations sur les scripts InDesign.