



Скрипты в InDesign

руководство для умных дизайнеров и ленивых верстальщиков

Основы объектной модели

Эффективная работа с текстом,
изображениями, таблицами,
фреймами и другими объектами

Проверка публикаций
перед цветоделением

Автоматизация спуска полос

Практические примеры
для оптимизации верстки



МАСТЕР

+CD



Михаил Борисов

Скрипты в InDesign руководство для умных дизайнеров и ленивых верстальщиков

Санкт-Петербург

«БХВ-Петербург»

2008

УДК 681.3.06
ББК 32.973.26-018.2
Б82

Борисов М. А.

Б82 Скрипты в InDesign: руководство для умных дизайнеров и ленивых верстальщиков. — СПб.: БХВ-Петербург, 2008. — 368 с.: ил. + CD-ROM — (Мастер)

ISBN 978-5-9775-0202-3

Рассмотрены вопросы создания скриптов в InDesign (CS2/CS3) для автоматизации работы дизайнера и верстальщика. В основу книги положено подробное описание реальных скриптов, разработанных автором в процессе многолетней практики. Приведена объектная модель InDesign. Показано создание окон диалога, управление документом и взаимодействие с мастер-страницами. Рассмотрены особенности использования скриптов при работе с текстовыми фреймами, форматировании текста и таблиц, использовании стилей, работе с изображениями. Уделено внимание практическим вопросам проверки корректности публикации перед выводом на цветodelение, автоматизации спуска полос. Показано взаимодействие с другими редакторами, входящими в состав пакета Creative Suite. В приложении приведены краткие справочники по JavaScript, VisualBasic и AppleScript. На компакт-диске располагаются примеры скриптов.

Для дизайнеров и верстальщиков

УДК 681.3.06
ББК 32.973.26-018.2

Группа подготовки издания:

Главный редактор	<i>Екатерина Кондукова</i>
Зам. главного редактора	<i>Игорь Шишигин</i>
Зав. редакцией	<i>Григорий Добин</i>
Редактор	<i>Анна Кузьмина</i>
Компьютерная верстка	<i>Ольги Сергиенко</i>
Корректор	<i>Зинаида Дмитриева</i>
Дизайн серии	<i>Инны Тачиной</i>
Оформление обложки	<i>Елены Беляевой</i>
Зав. производством	<i>Николай Тверских</i>

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 24.12.07.

Формат 70×100^{1/16}. Печать офсетная. Усл. печ. л. 29,67.

Тираж 2000 экз. Заказ №

"БХВ-Петербург", 194354, Санкт-Петербург, ул. Есенина, 5Б.

Отпечатано с готовых диапозитивов
в ГУП "Типография "Наука"

199034, Санкт-Петербург, 9 линия, 12

ISBN 978-5-9775-0202-3

© Борисов М. А., 2008

© Оформление, издательство "БХВ-Петербург", 2008

Оглавление

Глава 1. Общие сведения о скриптинге.....	9
1.1. Выбор языка	12
1.2. Инструментарий.....	16
1.2.1. AppleScript.....	17
1.2.2. JavaScript.....	17
1.2.3. VBA.....	19
1.3. Дом для скрипта.....	20
Глава 2. Общие сведения об объектной модели.....	25
2.1. Коллекции.....	26
2.2. Работаем с выделением	30
Глава 3. Диалоговые окна.....	33
3.1. Базовые методы.....	34
3.1.1. <i>alert()</i>	34
3.1.2. <i>confirm()</i>	34
3.1.3. <i>prompt ()</i>	35
3.2. Расширенные методы	35
3.3. Создание разных языковых версий	44
3.4. Новое в Creative Suite 3	45
Глава 4. Документы.....	49
4.1. Открытие документа.....	50
4.2. Сохранение документа	51
4.3. Закрытие документа.....	52
4.4. Работа с единицами измерения.....	54
4.5. Определение координат.....	56
4.6. Использование мастер-страниц	57
4.7. Печать документов.....	59
4.8. Экспорт публикации	62
4.8.1. Экспорт в PDF	62
4.8.2. Экспорт в EPS.....	65
4.8.3. Экспорт в HTML	67

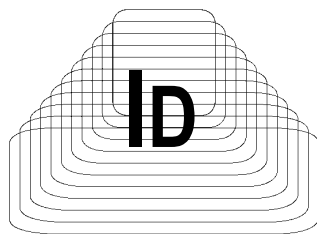
4.9. Выделение объектов заданного типа.....	72
4.10. Экспорт выделенных объектов	78
4.11. Группировка объектов	83
4.12. Трансформация объектов	83
4.13. Метла для монтажного стола	90
4.14. Спуск полос	92
Глава 5. Текстовые фреймы	97
5.1. Поиск текстового фрейма и страницы	101
5.2. Эти неуловимые абзацы	102
5.3. Добавление текста.....	104
5.4. Замена текста.....	106
5.5. Импорт текста	107
5.6. Вставка специальных символов.....	112
5.7. Удаление фреймов	114
5.8. Некоторые вопросы импорта документов из MS Office	114
5.9. Связывание текстовых фреймов	116
5.10. Поиск переполнения	118
5.11. Создание заякоренных фреймов	122
5.12. Перемещение объектов.....	124
5.13. Выполнение обтекания.....	126
5.14. Создание распашных заголовков.....	126
5.15. Расстановка колонтитулов	129
Глава 6. Форматирование текста.....	139
6.1. Установка свойств текста.....	140
6.2. Установка позиций табуляции	141
6.3. Работа с цветом	148
6.4. Использование стилей	149
6.4.1. Создание стиля символов	149
6.4.2. Создание стиля абзаца	150
6.4.3. Создание вложенного стиля	150
6.4.4. Удаление неиспользуемых стилей.....	153
6.5. Супермегаметла для публикации.....	157
Глава 7. Поиск и замена.....	161
7.1. Встроенные функции InDesign CS, CS2	162
7.1.1. Поиск текста без форматирования	162
7.1.2. Замена текста без форматирования	163
7.1.3. Замена текста и форматирования	163
7.2. Возможности InDesign CS3	164
7.2.1. Синтаксис регулярных выражений, используемых в InDesign CS3.....	166
7.3. Использование возможностей JavaScript.....	169
7.3.1. Проверка регулярных выражений JavaScript.....	171
7.3.2. Удаление пустых фреймов	173

7.3.3. Удаление пустых абзацев	173
7.3.4. Автоматизация форматирования	175
7.3.5. Автоматический корректор	178
7.3.6. Расстановка переносов	185
Глава 8. Таблицы	189
8.1. Создание таблицы	191
8.1.1. Объединение ячеек	193
8.1.2. Разбиение ячеек	194
8.1.3. Присвоение строкам чередующейся заливки	194
8.1.4. Задание свойств таблицы	195
8.1.5. Определение положения курсора в таблице	195
8.2. "Резиновые" таблицы	197
8.3. Быстрый перенос таблиц из Word	199
8.4. Форматирование таблицы	203
Глава 9. Работа с изображениями	217
9.1. Управление связями	219
9.2. Поиск изображений с разрешением менее заданного	221
9.3. Импорт графики	226
9.4. Создание каталога изображений	231
9.5. Автомат для создания фреймов	239
9.6. Автомат по раскладке рекламы на листе	246
Глава 10. Работа с контурами	257
Глава 11. Межпрограммное взаимодействие в Creative Suite 2	271
11.1. Автоматическая проверка публикации	271
11.2. Проверка самой верстки	273
11.3. Bridge и его роль в Creative Suite	274
11.4. Проверки в Illustrator	276
11.5. Скрипты	277
11.5.1. Этап 1. Проверка в InDesign	277
11.5.2. Этап 2. Проверка векторных иллюстраций в Illustrator	289
11.5.3. Этап 3. Конечные штрихи	293
ПРИЛОЖЕНИЯ	299
Приложение 1. Краткое сравнение синтаксиса AppleScript, JavaScript и VBScript	301
П1.1. Комментарии	301
П1.2. Переменные	302
П1.3. Преобразование из одного типа в другой	303
П1.4. Присвоение значений	304
П1.5. Сравнение значений	304

П1.6. Массивы	305
П1.6.1. Вложенные массивы	305
П1.7. Определение типа переменной	305
П1.8. Объединение строк	306
П1.9. Проверка условий.....	306
П1.10. Циклы	307
П1.11. Функции	307
П1.12. Пример	307
П1.12.1. AppleScript	308
П1.12.2. JavaScript.....	308
П1.12.3. VBScript	309
Приложение 2. JavaScript	310
П2.1. Переменные	310
П2.1.1. Задание переменных	310
П2.1.2. Типы переменных	311
Целочисленные значения.....	311
Значения с плавающей точкой.....	312
Логические значения	312
Строковые значения	312
Объекты	312
Специальное значение.....	312
П2.1.3. Преобразование типов	313
П2.2. Операции.....	313
П2.2.1. Операции сравнения	313
П2.2.2. Арифметические операции.....	314
П2.2.3. Логические операции.....	315
П2.2.4. Операции со строками	316
П2.2.5. Операции присваивания	316
П2.3. Ветвления (операторы условий)	317
П2.3.1. Оператор <i>if...else</i>	317
П2.3.2. Оператор <i>switch</i>	317
П2.3.3. Оператор <i>try...catch</i>	318
П2.4. Циклы	319
П2.4.1. Оператор <i>while</i>	320
П2.4.2. Оператор <i>do...while</i>	320
П2.4.3. Оператор <i>for</i>	321
П2.4.4. Оператор <i>break</i>	322
П2.4.5. Оператор <i>continue</i>	322
П2.4.6. Оператор <i>for...in</i>	323
П2.4.7. Оператор <i>with</i>	324
П2.5. Объекты	324
П2.5.1. Строковые объекты (<i>String</i>).....	325
Методы	325
Свойства	328

П2.5.2. Массивы	328
Свойства	329
Методы	329
П2.5.3. Функции	335
Вызов функции	335
Рекурсивные функции	335
Оператор <i>return</i>	336
П2.5.4. Объект <i>Math</i>	337
П2.6. Регулярные выражения	338
П2.6.1. Специальные символы в регулярных выражениях.....	339
П2.6.2. Опции поиска.....	341
П2.6.3. Свойства.....	342
Свойства <i>\$1</i> , ..., <i>\$9</i>	342
Свойство <i>input</i>	342
Свойство <i>lastIndex</i>	343
Свойство <i>lastMatch</i>	343
Свойство <i>lastParen</i>	343
Свойство <i>leftContext</i>	344
Свойство <i>rightContext</i>	344
П2.6.4. Методы.....	344
Метод <i>match</i>	344
Метод <i>search</i>	345
Метод <i>replace</i>	345
Метод <i>exec</i>	347
Метод <i>test</i>	348
П2.7. Комментарии	348
Приложение 3. Зарезервированные слова.....	349
Приложение 4. ExtendScript Editor	350
П4.1. Установка режима отладки	350
П4.2. Интерфейс.....	351
Приложение 5. Работа с файловой системой	353
П5.1. Объект <i>Path</i>	353
П5.2. Объект <i>File</i>	354
П5.3. Объект <i>Folder</i>	356
П5.4. Получение ссылки на скрипт	359
П5.5. Запуск связанного скрипта	360
П5.6. Получение результата работы скрипта.....	361
Приложение 6. Описание компакт-диска.....	362
Предметный указатель	364

ГЛАВА 1



Общие сведения о скриптинге

Любой человек в своей работе старается стать профессионалом. Повышение уровня мастерства может идти разными путями, но обязательно через овладение новыми знаниями и умениями, которые помогают решать весь спектр стоящих перед ним задач: как творческих, так и более тривиальных производственных. Все это в полной мере относится к работе за компьютером: современное программное обеспечение (ПО) предоставляет различные способы решения задач проще, быстрее, качественнее, надежнее, с каждой новой версией обрастая все новыми возможностями. Однако, несмотря на свою мощь, охватить необъятное невозможно — фактически разработчики ПО создают инструменты для решения лишь общих, наиболее востребованных операций. В результате любые специфические функции, необходимые в вашем производственном процессе, придется решать через набор типовых инструментов, что, как правило, выливается в непроизводительные затраты сил и времени.

Типичный пример из области предпечатной подготовки: автоматическая расстановка колонтитулов в препресс-пакетах не предусмотрена, ручная же расстановка — дело хлопотное и малотворческое. Или же верстка по всем типографским правилам, которая тоже требует существенных трат времени. В то же время переключившись на плечи машины подобных трудоемких операций, требующих повышенного внимания, позволит не только повысить эффективность работы, оставляя время для творческих задач, но и уберечь от пресловутого "человеческого фактора", неизбежного при большом объеме рутинной работы.

Для автоматизации рутинных операций в любом ПО, претендующем на роль профессионального, разработчиками, как правило, предусматриваются несколько способов.

Макросы (Actions) в зависимости от контекста использования обладают большей или меньшей функциональностью. В Photoshop под ними подразумеваются наборы команд, имитирующих нажатие клавиш и считывание зна-

чений из диалоговых окон, что является простейшим вариантом автоматизации. Достоинство всего одно — простота реализации: для создания макроса не требуется никакая специальная подготовка.

В других приложениях на макросы возлагается гораздо большая функциональность — типичным примером могут быть макросы, поддерживаемые офисным ПО, в которых реализованы возможности языка Visual Basic for Applications (в том числе использование модулей других программ через механизм ActiveX).

На другом полюсе — *плагин-модули* (plug-ins), которые создаются независимыми разработчиками. Они имеют наиболее полный доступ ко всем ресурсам приложений, используют функции оптимизации кода, распределения памяти, а также позволяют ограничивать свое незаконное распространение. По функциональности могут сравниться с самой хост-программой. Их написание требует специальной подготовки и хороших навыков программирования на языках высокого уровня (C++ и др.).

Наибольшую популярность в среде препресс-подготовки получили *скрипты*. Несмотря на их естественные ограничения в сравнении с плагинами, они прекрасно подходят для решения большинства задач, встающих перед пользователями. С одной стороны, по своей функциональности они находятся посередине между двумя полюсами, с другой — их написание не требует серьезных познаний в программировании. При желании скриптинг можно освоить за несколько месяцев без отрыва от основной работы (при условии, что раньше вы совершенно ничего не писали). Если же вы уже занимались программированием (например, в рамках Web-проектов), то задача значительно упрощается, поскольку фактически половину требуемых навыков вы уже имеете: по сложности освоения оба механизма — логика выполнения операций и доступ к элементам публикации — сравнимы.

Первый механизм предполагает знание базовых инструкций — операций сравнения, циклов, ветвлений; второй — четкое знание объектной модели приложения. По сравнению со скриптингом в Photoshop и Illustrator механизм, заложенный в InDesign, развит в наибольшей степени, давая возможность, например, создавать сложные пользовательские диалоговые окна, что, в свою очередь, позволяет наращивать функциональность скрипта.

Такое внимание разработчиков к программированию в InDesign вполне объяснимо: среди всех пакетов Adobe наибольший эффект от применения скриптов достигается именно в нем — ведь диапазон задач, потенциально поддающихся автоматизации при верстке публикаций, просто огромен. Особенно ощутима польза при обработке однородной, заранее подготовленной каким-либо образом информации. Например, верстка справочников, телепрограмм, прайс-листов, разнообразной финансовой, технической документации вообще немыслима без скриптов.

Скорость выполнения скриптов сравнительно высока, что для большинства препресс-задач вполне достаточно.

С выпуском программного комплекса Creative Suite (в частности, с появлением Adobe Bridge) сфера применения скриптов значительно расширилась, что наиболее ярко проявилось в среде межпрограммного взаимодействия (в рамках Creative Suite). Так, например, если в векторные макеты, помещенные в публикацию, были встроены растровые изображения, то через скриптинг можно подключить к обработке Photoshop. В этом случае происходит соединение InDesign с Illustrator и, в случае необходимости, дальнейшее переключение на Photoshop с последующим возвратом по цепочке назад. Фактически механизмы, заложенные в Bridge, на свой манер повторяют существующие в Visual Basic for Applications — мощном механизме межпрограммного взаимодействия под Windows.

Необходимо отметить значительные изменения, произошедшие в недавно вышедшем Creative Suite 3, которые коснулись в том числе и скриптинга, благодаря чему можно с уверенностью говорить, что, начиная с этой версии пакета, автоматизация обрела гибкость, необходимую для решения задач практически любой сложности.

Что касается платформы Macintosh, то скрипты на AppleScript способны реализовать широчайший набор команд, эквивалентный существующим в Visual Basic for Applications, и являются аналогичным инструментом автоматизации производственных процессов на системном уровне.

Отчетливо понимая, что подавляющая часть пользователей InDesign ранее не сталкивались с программированием, поспешу развеять возможные сомнения по поводу того, что творческий склад ума и программирование — вещи несовместимые. Дело в том, что для программирования на уровне скриптов (подчеркиваю — на уровне скриптов) совершенно не обязательно иметь особый склад мышления — вполне достаточно хотеть этому научиться и, конечно же, иметь определенный запас времени. Сам автор книги в прошлом был в подобной ситуации, пребывая в твердой уверенности, что программирование — не его призвание. Тем не менее, скажу по своему опыту: даже не имея никакого представления о программировании, освоить скриптинг под InDesign — вполне посильная задача. Написание скриптов значительно проще настоящих программ, для которых используются языки высокого уровня, где требуется специальная подготовка и опыт.

Цель книги — помочь в изучении скриптинга, поскольку справочное руководство от Adobe представляет собой 700-страничный фолиант (и это только для версии CS2!), в котором очень кратко перечислены базовые сведения о методах и свойствах объектов. Фактически содержание справочного руководства — одна большая таблица, в которой поиск даже штатными средствами

занимает осязаемое время. Поэтому в предлагаемой книге, во-первых, дается общая информация по написанию скриптов, во-вторых, объясняются взаимосвязи основных элементов публикации между собой, и, наконец, даны примеры использования наиболее часто употребляемых в публикации объектов и др.

Еще одно ценное качество книги, которое, безусловно, по достоинству оценят пользователи, — ее можно использовать как библиотеку уже готовых решений. В ней приведено множество полезных скриптов, большую часть которых можно без каких-либо (минимальных) переделок использовать в своей повседневной работе. И хотя скрипты редко бывают универсальными, тем не менее большинство приведенных в книге примеров пригодится многим, поскольку являются плодом работы автора, который профессионально занимается версткой и для облегчения своей ежедневной работы создал их для себя. Если же по какой-то причине создание скрипта затруднительно — можно связаться с автором для решения конкретной задачи, хотя все же надеюсь, что изложенный в данной книге материал будет достаточным для самостоятельного решения большинства стоящих перед вами задач.

Поскольку книга писалась в то время, когда Creative Suite 3 только-только появился, у автора не было возможности детально ознакомиться со всеми новшествами скриптинга в InDesign CS3. Основные изменения в новой версии, касающиеся скриптинга, коснулись, во-первых, интерфейсных возможностей, а также более тесного взаимодействия различных скриптов между собой в рамках выполнения глобальных задач на уровне организации замкнутых производственных процессов (например, упростилось взаимодействие с другими редакторами пакета Creative Suite). Это позволяет создавать на основе скриптов законченные коммерческие решения, что для подавляющего большинства пользователей InDesign с учетом времени, требуемого на их изучение, и вдобавок к тому достаточно узкой специализации, не принципиально. В то же время те возможности InDesign Creative Suite 3, которые представляют непосредственный интерес с точки зрения верстки, в данной книге отражение нашли.

1.1. Выбор языка

Скриптинг представляет собой процесс написания управляющих команд под определенное приложение. Их можно разделить на две группы: управляющие и исполняющие.

Исполняющие — это команды, ограниченные исключительно рабочей средой приложения (в нашем случае — InDesign): перейти на страницу, передвинуть объект, отформатировать абзац, вставить текст и т. п.

Кроме них нужен механизм, который бы позволял в зависимости от сложившейся ситуации направлять работу в требуемое русло (если..., то...), выполнять математические операции, проводить всякого рода проверки и т. п. Таким образом, нужна некая база, которая будет выполнять исключительно управляющие функции (при этом сами команды типа перехода на конкретную страницу или перемещения объекта будут играть лишь исполняющую роль). На роль такого управляющего отлично подходят три кандидата: Visual Basic, AppleScript и JavaScript. Каждый из них имеет обширный и достаточно удобный набор функций для того, чтобы выполнить любую специфическую задачу.

Выбор того или другого языка диктуется несколькими соображениями:

- ❑ на платформе Macintosh существует только AppleScript;
- ❑ для Windows выбор несколько шире: предлагаются Visual Basic и JavaScript.

Каждый из них имеет свои преимущества и недостатки.

Visual Basic — творение Microsoft, а потому имеет широчайший набор методов. Недостаток — совершенно не поддерживается на Macintosh. Этого недостатка лишен язык JavaScript. Он кроссплатформенный, т. е. будет работать в любой установленной системе. Его синтаксис отличается от Visual Basic, однако в силу того, что многие пользователи InDesign в той или иной степени сталкивались с Web-проектами, а потому уже хоть немного знакомы с языком.

В отличие от JavaScript, Visual Basic предоставляет гораздо более широкие возможности по автоматизации рабочих процессов, позволяя через ActiveX-компоненты обращаться к любым приложениям, зарегистрированным в системе — например, подключиться к Word, Excel, Access и т. п.

Visual Basic и AppleScript являются "полноценными" языками программирования, позволяя решать задачи системного уровня. Этого никак нельзя сказать о JavaScript, поскольку он ориентирован исключительно на использование возможностей той среды, в которой сценарии исполняются (в нашем случае — InDesign, который предоставляет JavaScript доступ к своим объектам, позволяя управлять их поведением).

Несмотря на определенные отличия между языками, способ их взаимодействия с InDesign совершенно идентичен.

Объем функциональности JavaScript (текущая версия 1.5) определен в стандарте ECMA 262. Не утомляя читателей глубокими сведениями о нем, замечу лишь, что в нем продумано все, что требуется для полноценной и относительно комфортной работы. В данном случае речь идет лишь о возможности реализации тех или иных действий, без оценки эффективности инструмента-

рия. Интересующимся могу порекомендовать ознакомиться с более продвинутыми спецификациями JavaScript 1.6 и 1.7, поддерживаемыми известным браузером FireFox.

Ядро JavaScript 1.5 состоит из небольшой группы фундаментальных объектов, среди которых — строки (Strings), массивы (Array), пользовательские функции (function) и математические функции (Math), управляющие структуры и операторы и др. Каждый объект имеет свои свойства и методы, которые и реализуют всю функциональность языка.

В целях безопасности в стандарт не включены некоторые механизмы — например, работа с файловой системой (создание, открытие, перемещение, удаление файлов и папок), запуск других программ и т. п., что хоть в какой-то мере служит сдерживанию распространения вирусов и всякого рода malware через интернет-браузеры. Соответственно, каждый разработчик ПО самостоятельно реализовывает недостающие компоненты в нужном объеме, исходя из принципа необходимой достаточности — естественно, вопросы обеспечения безопасности в таком случае также полностью возлагаются на него. Исходя из потребностей специалистов предпечатной подготовки, Adobe расширила определенные в стандарте средства JavaScript инструментами для доступа к файловой системе (редакция известна как ExtendScript).

Учитывая значительную распространенность JavaScript и в то же время стремясь расширить сферу применения скриптов, Adobe поступила достаточно мудро: она позволила скриптам, работающим в своих приложениях, вызывать другие скрипты, причем они могут быть написаны на разных (поддерживаемых) языках. Это позволяет, с одной стороны, обойти ограничения языка, а с другой — использовать уже имеющуюся библиотеку скриптов, написанных на привычном языке. Например, JavaScript может вызывать блок, написанный на Visual Basic и пользоваться всеми преимуществами такого распределения ролей.

Главным критерием при выборе языка программирования является его конечная нацеленность: если предполагается использование скрипта в сочетании с другими приложениями (разработанными не Adobe), то единственным вариантом будет либо Visual Basic (Windows), либо AppleScript (Macintosh). Подключение программ не из пакета Creative Suite при предпечатной подготовке — явление крайне редкое, поэтому данное ограничение JavaScript для рассматриваемых в данной книге задач значения не имеет. Более того, при необходимости можно делать вставки на Visual Basic либо AppleScript, что вообще нивелирует отличия.

Если сравнить функциональность скриптинга в InDesign с QuarkXPress, то необходимо отметить, что, во-первых, в QuarkXPress реализована поддержка исключительно AppleScript, поэтому программирование для него возможно

лишь на платформе Macintosh. Причину такого состояния дел, по всей видимости, следует искать в традиционной ориентации препресс-процессов на данную платформу. Во-вторых, разработка Adobe гораздо более завершенная и зрелая — это касается не только функциональности, но и качества реализации (больше ошибок, недочетов разработчиков).

Что скрипты могут? С их помощью можно выполнять любые операции, доступные через меню и палитры программы. Вы можете создавать новые документы, страницы, текстовые фреймы, форматировать текст, вставлять графику, отправлять на печать и экспортировать содержимое.

Новое в InDesign Creative Suite 3

В Creative Suite 3 сфера применения скриптов значительно расширилась и теперь с их помощью можно реализовывать те функции, которые через программный интерфейс не доступны.

Среди возможностей:

- создание компилированных скриптов (с расширением `jsxbin`), что дает возможность защитить свой скрипт от несанкционированного копирования, а также повысить его быстродействие;
- создание собственных меню, в том числе контекстно-зависимых;
- поддержка событий, позволяющая выполнять те или иные действия при наступлении определенных условий (например, сразу после открытия документа);
- назначение объектам скрипта, что дает возможность возложить на них функции автоматического отслеживания изменений в документе и выполнения соответствующих операций (например, изменение своих размеров и т. п.);
- возможность сохранения значений переменных после выполнения скрипта с предоставлением их для использования другим скриптам;
- отображение процента выполнения задания, что полезно при выполнении объемных публикаций или задействовании других программ из пакета Creative Suite.

В Creative Suite 3 Adobe пошла дальше и расширила возможности Bridge (позволяя, например, подключение к FTP-серверам или передачу данных по HTTP-протоколу и т. п.), усовершенствовала взаимодействие с другими приложениями Adobe, а также реализовала возможность подключения внешних модулей, написанных на C++, что открывает поистине безграничные возможности для полной автоматизации ряда производственных процессов.

В InDesign 3 значительно расширилась база для применения скриптов — в первую очередь за счет возможности запуска скрипта из другого скрипта, причем на любом из поддерживаемых языков: на платформе Windows — JavaScript и Visual Basic, на Macintosh — AppleScript и JavaScript.

Подобное взаимодействие используется в серьезных проектах по автоматизации рабочего процесса, например, для обеспечения связи с другими компонентами издательского комплекса, например с MS Word. Другой пример — получение выборок из Access, что необходимо для взаимодействия с БД (JavaScript не позволяет этого напрямую).

Еще одна причина — возможность привязки скрипта к любому объекту. Как известно, каждый объект в публикации имеет свойство `label` (ярлык), в котором может храниться любой текст. В Creative Suite 3 развили данный вопрос, и теперь, если в `label` записан текст скрипта, программа может его исполнять автоматически. Такой механизм позволяет достичь еще более глубокой степени автоматизации верстального процесса, поскольку объекты сами могут отслеживать изменение ситуации и выполнять заложенные в них действия.

Чего скрипты не могут? Им закрыт доступ к трем типам операций:

- ☐ изменение цветовой модели документа;
- ☐ доступ к содержимому системного буфера (это ограничение в некоторых случаях можно обойти);
- ☐ установка параметров рабочего окружения.

Также скриптинг не поддерживает создание пользовательских типов объектов (просто новые объекты — без проблем), а также реализацию глубинных механизмов — например, собственного модуля, выполняющего композицию текста. Для таких случаев предусмотрен более серьезный инструментарий (Software Development Kit), который позволяет создавать плагин-модули с использованием C++.

1.2. Инструментарий

В зависимости от используемой среды вам понадобится различный инструментарий. Для создания скриптов для Macintosh потребуются интерпретаторы JavaScript или AppleScript версий 1.6 и выше, а также собственно редактор AppleScript (оба идут в стандартной поставке с Mac OS). Те, кому мало функциональности стандартного редактора, могут попробовать более продвинутый Script Debugger разработки Late Night Software (<http://www.latenightsw.com>).

Как уже говорилось, для написания пользовательских сценариев для продуктов Adobe в среде Windows можно использовать JavaScript (не путайте с JScript — Microsoft-версией языка, она не поддерживается), либо продукты семейства Microsoft Visual Basic — например, VBScript, Visual Basic 5, Visual Basic 6, Visual Basic .NET, Visual Basic 2005 Express Edition. При этом следует учитывать, что, начиная с Visual Basic .NET, функциональность скриптов ниже, поскольку в .NET не поддерживается тип данных `Variant`, широко используемый в InDesign.

Несмотря на поддержку Visual Basic, в установочный пакет InDesign его интерпретатор не включен, поскольку он идет с офисными приложениями пакета MS Office (в виде Microsoft Visual Basic for Applications (VBA)), достаточное лишь при инсталляции включить соответствующую опцию.

Для корректной работы с Visual Basic необходимо, чтобы InDesign устанавливался пользователем с правами администратора. С запуском скриптов проблем не возникнет, но вот добавление новых доступно лишь членам групп Administrator (Администратор) либо Power Users (Опытные пользователи).

В отличие от VBA, поддержка JavaScript заложена в дистрибутив. Она включает в себя все возможности JavaScript 1.5 и соответствует нынешнему стандарту ECMA 262. Определенные в этом стандарте функции расширены операциями с файлами и папками.

1.2.1. AppleScript

Для просмотра свойств и методов, доступных в AppleScript:

1. Загрузите InDesign.
2. Загрузите Apple Script Editor.
3. В Apple Script Editor выберите **File | Open Dictionary**.
4. Из списка приложений выберите InDesign. Apple Script Editor отобразит список всех коллекций объектов.
5. Выберите конкретную коллекцию для просмотра доступных в ней методов и свойств.

1.2.2. JavaScript

Для просмотра свойств и методов, доступных в JavaScript:

1. Загрузите ExtendScript Toolkit.
2. В ExtendScript Toolkit выберите **Help | InDesign CS2 Main Dictionary**.
3. Из списка классов выберите интересующий вас объект.
4. Для детальной информации по любому методу или свойству выберите их, после чего в окнах **Properties** и **Methods** появится их описание (рис. 1.1).

Писать скрипт можно в любом текстовом редакторе, но для того чтобы пользоваться отладчиком (ошибки будут всегда, особенно на стадии изучения, а отладчик позволяет запустить скрипт пошагово и, таким образом, достаточно быстро локализовать проблему), файлу нужно дать расширение `jsx` (с ним в Creative Suite 2 ассоциирован редактор ExtendScript Editor, устанавливающийся с любым ПО от Adobe). Если используется Creative Suite первой версии, то расширение `js` на `jsx` менять не нужно. Добавление в скрипт строки `$.level=1` даст возможность отладчику остановиться в указанном вами месте (его помечают через `$.bp()`). К слову сказать, отладчик в новой версии более "продвинутый", пользоваться им гораздо удобнее, чем в предыдущей версии, поэтому настоятельно рекомендую использовать именно Creative Suite 2.

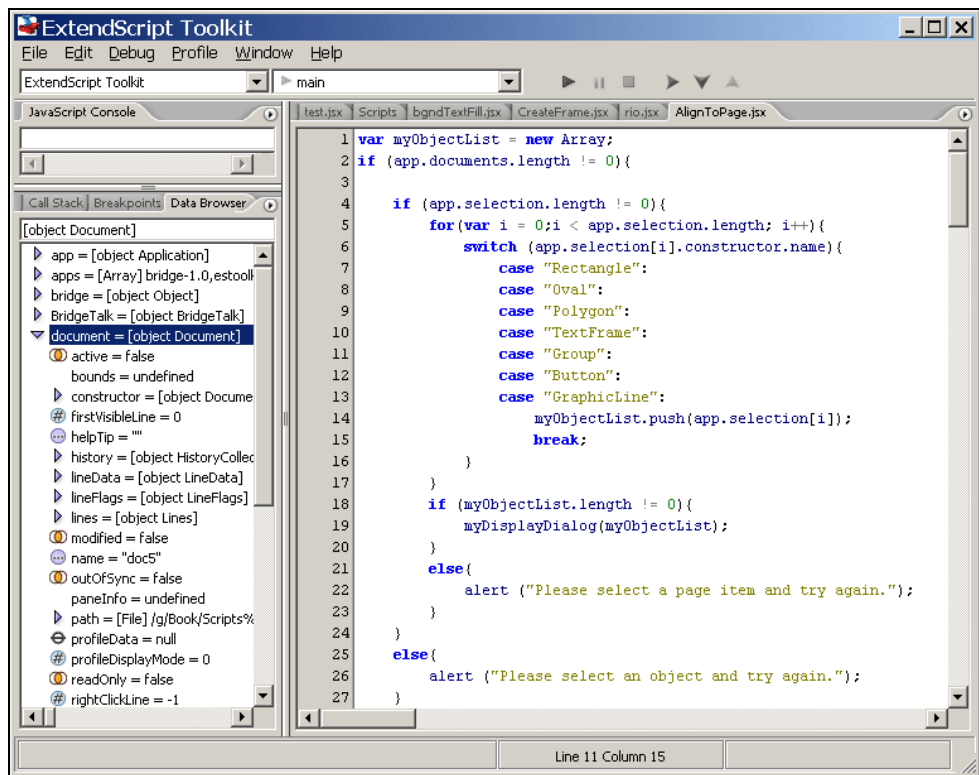


Рис. 1.1. Интерфейс ExtendScript

При активном тестировании может возникнуть ситуация, когда скрипт начинает "глючить", т. е. выдавать ошибку там, где ее на самом деле нет. Как правило, это возникает в том случае, когда скрипт повторно применяется к одному и тому же документу. При этом память "засоряется" историей ваших ошибок, поэтому желательно тестовый файл перед повторным запуском скрипта закрыть (в этом случае память очищается) и вновь открыть. Конечно, при интенсивном тестировании это раздражает, а определить, что скрипт работает уже некорректно, невозможно, поэтому возьмите за правило заново открывать файл через несколько запусков скрипта. Если же скрипт вообще начинает "спотыкаться на ровном месте", перезагрузите отладчик и, желательно, хост-приложение.

Для удобства в левом окне отладчика (панель **Data Browser**) отображаются текущие значения всех используемых переменных, что позволяет следить за ними без каких-либо дополнительных действий. Если вам потребуется узнать какое-то конкретное значение, можно вписать необходимую строку в верхнем окне панели **JavaScript Console**.

И напоследок. Перед запуском скрипта в отладчике проверьте, какое хост-приложение выбрано в качестве рабочего (панель **Target Application**). По умолчанию отладчик настроен на ExtendScript Toolkit, поэтому будьте бдительны.

1.2.3. VBA

Для просмотра свойств и методов, доступных в Visual Basic for Applications (для запуска редактора откройте Word, перейдите на вкладку Макросы (в Office 2007), введите имя макроса (должно начинаться с буквы) и потом нажмите кнопку **Создать**. Загрузится Microsoft Visual Basic.

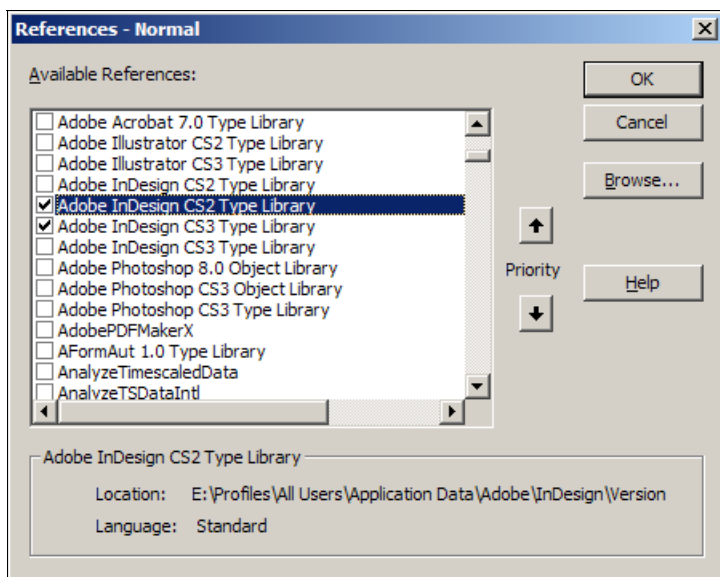


Рис. 1.2. Подключение необходимых библиотек в Visual Basic

1. Выберите **Tools | References**. В списке найдите **Adobe InDesign xxx Type Library** (здесь xxx — либо CS2, либо CS3), поставьте флажок напротив и нажмите кнопку **OK** (рис. 1.2). Если по каким-либо причинам библиотека (ResourcesforVisualBasic.myTable) в списке не появилась, нажмите кнопку **Browse** и вручную укажите ее месторасположение. Обычно она находится в папке ~:\Documents and settings\user_name\Application Data\Adobe\InDesign\Version ...\Scripting Support\.
2. Выберите **View | Object Browser**.
3. В списке **Libraries** выберите InDesign. Visual Basic отобразит список всех объектов InDesign (рис. 1.3).

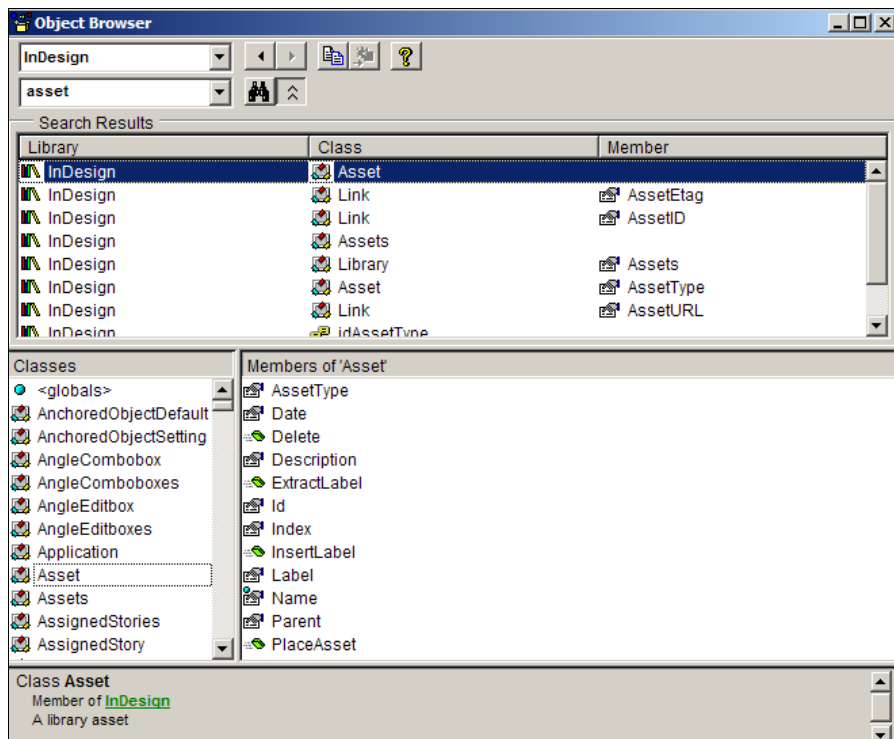


Рис. 1.3. Очень удобный интерфейс Visual Basic, с описанием поддерживаемых методов и свойств

4. Выберите имя любого объекта. В соответствующих окнах будут отображены свойства и методы данного объекта.

1.3. Дом для скрипта

Для того чтобы скрипт был "виден", его размещают в папке Install\Adobe\InDesign\Presets\Scripts, где Install — папка, в которой установлены приложения (обычно ~\Program Files\, здесь ~ — системный диск), либо ~\Documents and Settings\user_name\Application Data\Adobe\InDesign\Scripts, а для Macintosh: ~/Library/Preferences/Adobe InDesign/Scripts.

При необходимости в эти папки можно помещать не сами скрипты, а только ярлыки на них.

Новое в InDesign Creative Suite 3

В InDesign CS3 изменился путь для расположения пользовательских скриптов. Теперь они должны были размещены в папке пользователя drive_name:\Profiles\user_name\Application Data\Adobe\InDesign\Version 5.0\Scripts\Scripts Panel.

Если необходимо, чтобы некоторые скрипты исполнялись в момент запуска редактора, их помещают в папку `Install\Adobe\Adobe InDesign CS3\Scripts`.

В отличие от других продуктов Adobe, InDesign (любой версии) в запущенном состоянии автоматически сканирует папки, и как только появится новый скрипт, он сразу же, без перезагрузки редактора, становится доступным для использования.

В InDesign предусмотрена специальная палитра **Scripts (Window | Automation | Scripts)**, откуда можно запускать пользовательские скрипты (рис. 1.4). Это очень удобно, особенно при интенсивном использовании скриптов, поскольку отпадает необходимость каждый раз путешествовать по меню в поисках нужной операции.

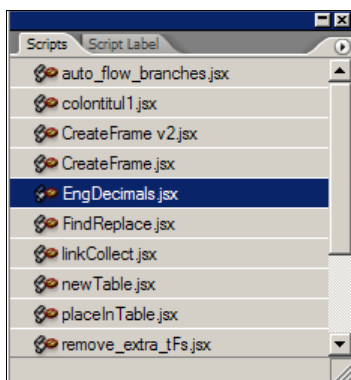


Рис. 1.4. Палитра **Scripts**

Скрипты, запущенные через палитру **Scripts**, работают быстрее, чем запущенные через Проводник (Finder).

В панели могут быть запущены скрипты с расширениями `spt`, `as`, `applescript` (для AppleScript), `js` и `jsx` (JavaScript), `vbs` (VBScript). Кроме того, скрипты на JavaScript могут быть компилированными (`jsxbin`).

Открыть скрипт можно несколькими способами: в любом редакторе, либо, что проще, нажав клавишу `<Alt>` (`<Option>` — в MacOS) и дважды щелкнув на его названии в палитре **Scripts**. При этом он откроется в ExtendScript Editor (если имеет расширение `jsx`), если расширение — `js`, то в редакторе, ассоциированном с данным типом файлов.

Если нужно открыть папку, в которой находится скрипт, достаточно нажать комбинацию клавиш `<Ctrl>+<Shift>` (`<Command>` — в MacOS) и также дважды щелкнуть на названии скрипта в палитре. Альтернативный вариант — использовать команду **Reveal in Explorer (Reveal in Finder** — в MacOS) из меню палитры **Scripts**.

По умолчанию в Creative Suite 2 скрипты выполняются как набор последовательных действий, что очень помогает на стадии отладки, поскольку позволяет отслеживать изменения после каждого шага. Обратная сторона медали — для отмены действия скрипта придется нажимать комбинацию клавиш `<Ctrl>+<Z>` множество раз, что в случае значительных изменений становится практически нереальной задачей.

Отличие версий

В Creative Suite 3 ситуация изменилась: теперь скрипт рассматривается как единая команда, и для отмены ее действия достаточно однократного нажатия комбинации клавиш `<Ctrl>+<Z>`. Чтобы иметь возможность пошагового отслеживания, нужно активизировать опцию **Undo Affects Entire Script** из меню панели **Scripts**. Выполнение скрипта как единой команды ускоряет процесс его исполнения, поскольку не тратятся дополнительные ресурсы на создание точек восстановления.

Кроме запуска скриптов непосредственно из палитры **Scripts**, в Creative Suite 3 предусмотрено использование клавиатурных эквивалентов. Для назначения скрипту эквивалента используйте **Edit | Keyboard Shortcuts**, выберите набор ускорителей из меню **Set**, после чего перейдите на **Product Area | Scripts**. В новом окне выберите необходимый скрипт и присвойте еще неиспользованное клавиатурное сокращение.

Чтобы при отладке в ExtendScript автоматически запускалась объектная модель именно для InDesign, первая строка в скрипте должна содержать указание на хост-приложение:

```
#target = "InDesign-4.0"
```

Цифры в названии указывают номер версии InDesign (табл. 1.1).

Таблица 1.1. Соответствие идентификаторов номерам версий

Номер версии	Идентификатор
InDesign Creative Suite	3
InDesign Creative Suite 2	4
InDesign Creative Suite 3	5

Отсутствие явного указания на номер версии, например,

```
#target = "InDesign"
```

приводит к использованию самой последней установленной на компьютере версии продукта.

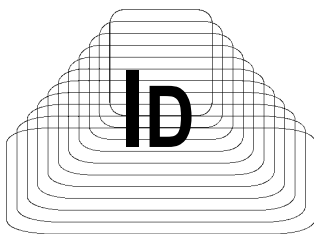
При переходе на InDesign CS3 помните, что из-за изменений в его объектной модели по сравнению с предыдущей версией некоторые ранее работающие скрипты перестают нормально функционировать. Выхода два:

- ☐ переписать код под CS3;
- ☐ вставить в начале скрипта строку

```
app.scriptingVersion = 4.0
```

Она принудительно активизирует исполнительный механизм от предыдущей версии вместо самой последней установленной на компьютере.

ГЛАВА 2



Общие сведения об объектной модели

Все современные приложения (и InDesign в том числе) используют объектно-ориентированную модель для доступа к воздействуемым объектам. Что это значит?

Представьте себе дом, в котором есть двери, окна, комнаты — это все как раз и есть разные объекты, принадлежащие данному дому. Каждый сложный объект состоит из более мелких, которые являются зависимыми по отношению к своему родителю. Так, например, светильник имеет несколько лампочек, которые, в свою очередь, состоят из нити накаливания, цоколя, колбы и т. д.

Одни объекты (например, двери) могут быть открыты или закрыты, иметь цвет, размеры, отличаться фурнитурой, другие (например, светильник) — открытыми или закрытыми быть не могут, зато могут быть включенными; это свойства объектов. Наконец, одни объекты можно только открывать и закрывать, другие — только включать, а третьи — открывать, закрывать и включать. Это методы, которыми воздействуют на объекты.

Если провести аналогию с языком, то *объекты* можно рассматривать как существительные, *методы* — как глаголы, а *свойства* — как прилагательные. Складывая из них предложения, мы получаем текст, который понятен собеседнику, точно также набор команд в скрипте будет понятен InDesign.

Воздействовать на объекты можно несколькими способами. Например, дать команду "Открыть двери" — при этом все двери дома будут открыты. В то же время команда "Открыть дверь" останется без исполнения — ведь совершенно не ясно, о какой именно двери идет речь. Второй вариант — дать команду "Открыть дверь в гостиной", она четко указывает, какую дверь выбрать (при условии, что в гостиной только одна дверь). Точно так же команда "Открыть двери в гостиной" не будет исполнена, если в ней только одна дверь. Таким

образом, давая ту или иную команду нужно четко представлять объект воздействия и понимать логику предстоящих действий.

Кроме того, нельзя сбрасывать со счетов фактор скорости. Например, неэффективной будет команда "Открыть двери" для того, чтобы открыть лишь конкретную дверь в конкретной комнате. Это особенно проявляется при больших объемах обработки, например работе с таблицами: число ячеек растет в геометрической прогрессии от размеров таблицы и форматирование каждой в отдельности даже на мощной машине займет значительное время.

В то же время, подходить к скриптингу нужно творчески. Например, операция "Открыть двери" займет меньше времени, чем поочередное выполнение команд "Открыть дверь в спальне", "Открыть дверь в детской" и т. д. для всех комнат дома.

2.1. Коллекции

Для удобства работы объекты с одинаковыми свойствами объединены в *коллекции* (или *классы*). Так, двери — это одна коллекция, обладающая своими уникальными свойствами, окна — другая и т. д. В InDesign каждый объект, будь то текстовый фрейм, символ текста, абзац или цвет, принадлежит своей коллекции (текстовым фреймам, символам, абзацам, образцам цвета соответственно), причем сама коллекция также является объектом, с собственными свойствами и методами. Использование коллекций очень удобно с практической точки зрения: они позволяют, например, вместо команды "Открыть в объекте № 123 объект номер 12345" указать "Открыть четвертую дверь на втором этаже", что более понятно.

Любая коллекция объектов InDesign является по своему внутреннему строению массивом JavaScript. К каждому элементу массива можно обращаться несколькими способами. Самый распространенный — по порядковому номеру (или *индексу*), задавая его в квадратных скобках. Счет ведется с 0: `array[0]` — самый первый элемент, `array[array.length-1]` — последний.

Кроме использования порядковой (*абсолютной*) нумерации, предусмотрена *относительная*, при которой можно полностью отказаться от указания конкретного индекса объекта, достаточно сказать: "Открыть предыдущую дверь" или "Закрыть следующую за предыдущей", что во многих случаях оказывается даже более эффективно. Естественно, при этом предполагается, что базовый объект должен быть в любом случае определен. Используется, как правило, только в тех случаях, когда получение индекса по каким-либо причинам затруднено. Наиболее яркий пример — работа на уровне абзацев (они — понятие абстрактное, вычисляются исключительно по признаку конца строки), далее об этом пойдет речь подробнее.

Еще один вариант обращения к объекту — по имени (`array["My First"]`), которое по понятным причинам должно быть уникальным. Второй способ используется достаточно редко, т. к. предварительно придется задать имя объекта, что не всегда оправдано, хотя в некоторых случаях является единственным выходом.

Имя коллекции всегда указывается во множественном числе (в конце — "s") с той целью, чтобы лишний раз подчеркнуть, что речь идет о множестве (массиве) объектов, а не о каком-то конкретном объекте, например, `paragraphs`. Указание порядкового номера объекта в коллекции дает ссылку на конкретный объект, который, в полном согласии с логикой, используется уже в единственном числе (`paragraph`) — именно так информацию, касающуюся абзацев, нужно искать в Руководстве по скриптингу.

Знание объектной модели InDesign — фундаментальное требование для написания скриптов, без нее можно уподобиться мартышке из известной басни, у которой было множество очков, но она не знала, что с ними делать. Точно так же, не зная причинно-следственных связей, вы не сможете сделать и шага, поскольку InDesign будет вежливо, но настойчиво информировать, что у него не получается включить дверь или выключить окно.

Поскольку объектное дерево InDesign очень развесистое, на рис. 2.1 дано лишь самое общее представление о нем. Толстая рамка означает коллекцию объектов, тонкая — конкретный представитель коллекции (для краткости показан лишь наиболее часто используемый объект — `textFrame`). Далее, в соответствующих главах будут подробнее рассматриваться коллекции, наиболее востребованные в скриптинге. Разобравшись с ними, вам не составит никакого труда использовать остальные объекты.

Рисунок не отображает взаимосвязей на более низких уровнях — например, таблица может содержать вложенную таблицу, группа иметь подгруппы, текстовый фрейм может включать закоренный фрейм, в котором, в свою очередь, может быть текст, таблица, изображение — глубина вложенности не ограничена.

На вершине иерархической лестницы, как и следовало ожидать, расположен объект `Application` — это сам InDesign. Да, не удивляйтесь — коль скоро мы используем объектную модель, само приложение также становится объектом. `Application` (будем использовать более краткий вариант — `app`) является родителем для всех остальных объектов: открытых в нем документов, стилей, цветов, диалогов, книг, библиотек, а также массы различных предустановок (`Preferences`, `Options`, `Settings`, `Presets`, `Defaults`). Большинство объектов-предустановок не могут быть изменены, а другие, такие как `changePreferences` (объект, в котором собраны все установки по поиску/замене), открыты для пользователя.

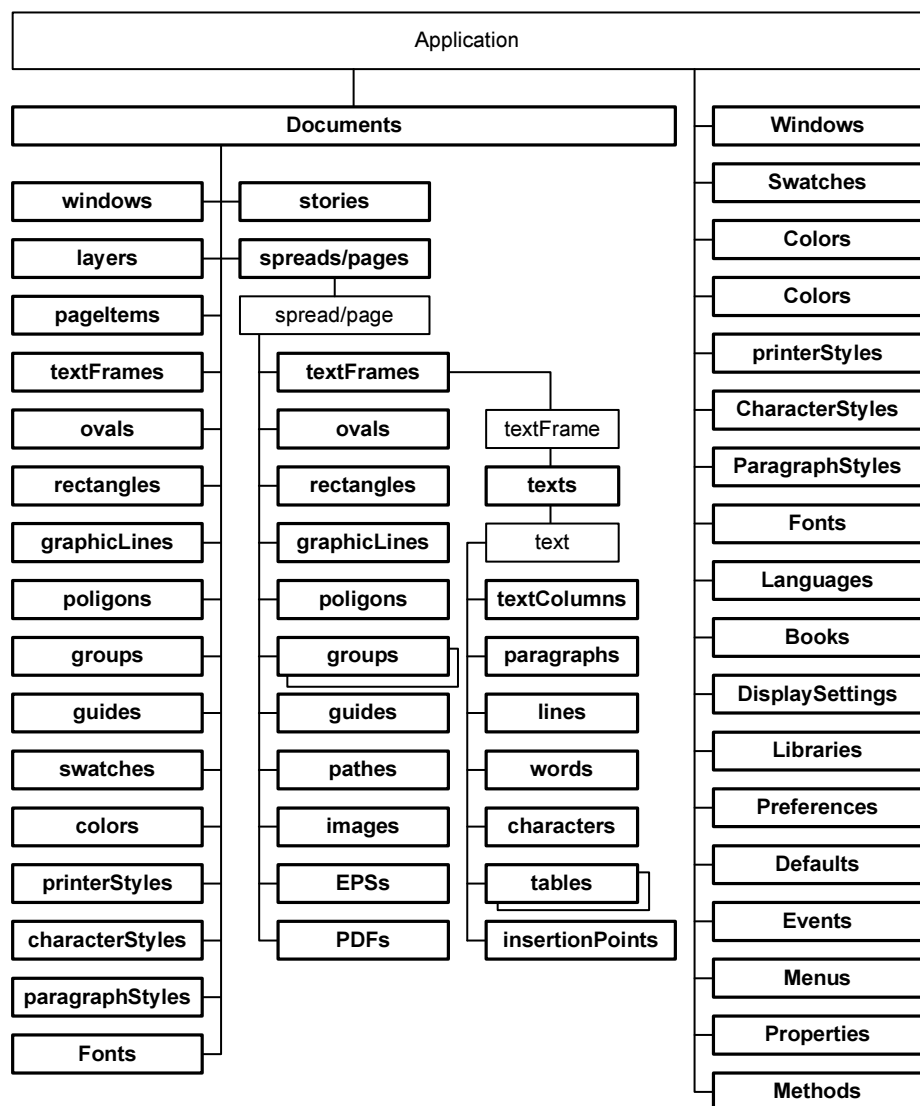


Рис. 2.1. Объектная модель InDesign

Объекты, которые чаще всего используются, — публикация (Document), материал (Story, цепочка связанных между собой текстовых блоков), поскольку он является непосредственным родителем для текстового содержимого (words, characters), а также текстовый блок (textFrame).

Как уже ранее отмечалось, объект, над которым будет производиться действие, нужно указывать однозначно. Однозначность достигается перечислени-

ем всей вышестоящей иерархии воздействуемого объекта: например, название документа, номер слоя, номер текстовой цепочки, номер страницы и т. д. вплоть до самого объекта.

Для того чтобы спуститься по иерархии до конкретного объекта, перечисляют всю вышестоящую цепочку объектов, разделяя каждый уровень точкой (.). Например, чтобы получить содержимое четвертого фрейма на второй странице открытого документа, нужно использовать такую конструкцию:

```
application.documents[0].pages[1].textFrames[3].contents
```

Скриптинг дает широкие возможности, однако не забывайте, что вам не все дозволено — например, порядок перечисления элементов не менее важен, и все попытки изменения своей объектной модели InDesign будут жестко пресекать.

Коллекция документов — набор всех открытых на данный момент публикаций. Соответственно к текущей публикации можно обратиться через `app.documents[0]`, предусмотрено и более удобное название — `app.activeDocument`. Материалов, как правило, в публикации несколько, поэтому доступ к ним возможен через `app.activeDocument.stories[index]`, и т. д.

Новое в InDesign Creative Suite 3

В связи со значительными изменениями, произошедшими в Creative Suite, писать скрипты стало гораздо проще. Одно из полезных нововведений — встроенный Object Browser, который помогает ориентироваться во всем многообразии иерархических связей InDesign. В то же время, во всяком случае, на данный момент, его использование малоэффективно, а вот включение интерактивного InDesign Object Library — настоящая находка.

Объектная модель InDesign отображается в любом из приложений, которое выбрано для редактирования скриптов.

Для того чтобы постоянно не повторять всю вышестоящую иерархию и тем самым повысить читаемость кода, рекомендуется использовать сокращенный доступ к объектам:

```
myDocument = app.documents[0]  
myPage = myDocument.pages[2]
```

Данной записью мы переменной `myDocument` присваиваем ссылку на объект `app.documents[0]`, а переменной `myPage` — ссылку на третью страницу.

Соответственно предыдущую запись

```
application.documents[0].pages[1].textFrames[3].contents
```

теперь можно переписать как

```
myPage.textFrames[3].contents
```