JavaScript: скрипты для InDesign. Беседы с ChatGPT

https://dotextok.ru • 2025

Как появилась эта книга

Я начал заниматься скриптами для индизайна в 2005 году. Работы было много, сроки всегда жёсткие, в таком режиме всегда актуален поиск решений, экономящих время. И интересно было не только использовать найденные решения, но и делать свои. Коллекция постепенно пополнялась, были публикации в Publish. И я стал думать, что может, книгу написать о скриптинге для индизайна. Поскольку небольших модулей накопилось много, и я из них собирал свои скрипты, рабочее название книги было «Создание скриптов в стиле лего». Но это не было таким сильным желанием, что надо всё бросить и только делать книгу. Просто периодически появлялась такая мысль, я записывал вопросы, которые хорошо бы осветить, стала вырисовываться очерёдность глав. Но дальше дело не шло.

Время на месте не стоит, и неожиданно в нашей жизни появился искусственный интеллект как помощник решения разных задач. И осенью 2024 года я стал пробовать подключать искусственный интеллект к решению задачи индексирования русских книг. Эта работа был интересной не столько в результатах решения этой задачи, сколько в том, что я понял, какие же мощные эти языковые модели. И тут снова проснулась та мысль про книгу.

Она уже требовала действий, поскольку появились такие новые возможности, и надо было определиться, на какую же языковую модель ориентироваться. Я попробовал несколько самых популярных моделей и для работы с книгой выбрал ChatGPT, вариант InDesign-Script-Assistant.

И написание книги стало видеться уже совершенно иначе. У меня к этому времени была написана примерная очерёдность глав. Но в процессе обсуждений с ChatGPT структуры возможной книги неожиданно оказалось, что искусственный интеллект предложил более полный вариант. И мне это понравилось. Это была уже серьёзная причина перейти, наконец, от периодических размышлений к конкретным действиям по подготовке книги.

Главная идея, которую я хотел реализовать, если когда-нибудь возьмусь написать книгу об ExtendScript для InDesign — сделать её не как учебник по языку, а как руководство по автоматизации реальных задач верстки. Собственно, смастерить тот конструктор скриптов. Но беседы с ChatGPT убе-

Беседы? С программой?!

Это были реальные разговоры. Мы в обычном разговоре, если повторят вопрос, всегда отвечаем другими словами. И мне поначалу было непривычно, что ИИ тоже не повторяется. Поэтому важное правило, если вы будете общаться с какой-нибудь языковой моделью: ***Записывать в файл вопрос, прежде чем его задавать.***

Это, кстати, поможет исключить грамматические ошибки. Всё общение идёт на русском языке. Полученный ответ надо тоже переносить сразу в файл. Хотя на экране и формируется лента вопросов-ответов, не раз бывало, что ChatGPT уставал отвечать, весь текст диалога стирался и снова был чистый экран. Если такое произойдёт, обратно текст беседы не вернёшь. А постоянная запись вопросов и ответов поможет всё сохранить.

дили, что надо делать другую книгу. Только компоновка модулей в скрипты — это мало, можно сделать намного более информативную книгу. И мы взялись за работу. Я задавал тему, вопросы, а искусственный интеллект очень быстро давал развёрнутые ответы. Нужно было сперва проверить работоспособность предлагаемых скриптов, что-то отправить на переделку. Иная формулировка вопроса часто давала другой ответ. И работа над точностью формулировок была очень интересной!

ChatGPT устойчиво называет текстовый фрейм текстовой рамкой. Сначала я сопротивлялся, в потом оставил. Вот такой перевод английского text frame в этих книгах. И ещё пролезло и осталось обращение с читателем на *ты*. Не везде, но во многих объяснениях. Это из тех диалогов, там обращение было всегда на ты.

У меня нет желания иметь именно бумажный вариант книги. Полиграфического исполнения. Мне представляется, что желающих освоить мастерство скриптописания много больше, чем современные тиражи книг. И по этой причине я решил сделать только электронный вариант в формате PDF. Поэтому я не ограничивал себя в выборе размера страницы, он A4, можно напечатать для себя. И ограничения на использование цветов тоже нет. Книга в первую очередь для чтения на экране, поэтому используется кегль 12. Немного больше 200 страниц.

Неожиданно для меня процесс подготовки материалов вдруг обозначил важность базовых знаний JavaScript в том плане, что эти знания надо читателям дать в этом проекте. Да, сейчас информации просто избыток, и важным становятся две вещи: во-первых, время, которое мы тратим в океане информации, пытаясь найти нужное, и второе — уверенность, что то, что нашёл, тебе подходит. Мы постарались, чтобы эта информация была добротной.

Но добавление сведений об особенностях языка программирования заметно выпадало из того курса, по которому я и ChatGPT двигались. По крайней мере, мне это так виделось. Поэтому я просто сделал ещё одну книгу. «JavaScript. Краткое описание». Вместе с ChatGPT, конечно. Переменные, разные операции, условия, циклы, массивы, объекты, регулярные выражения и пр. Всего 85 страниц формата A4.

А та самая первая идея — публиковать готовые модули, из которых монтируются новые скрипты, она реализована в Приложении, в главе «Часто используемые фрагменты кода». Там, конечно, не все модули. Но 116 штук, это тоже немалое число.

Возможно, что чего-то мы не учли. Не знаю, будет ли вариант «исправленное и дополненное», но вы всегда можете уточнить у ChatGPT или какой-то другой языковой модели любой вопрос по скриптингу.

Я понимаю, что не всем вообще нужен этот скриптинг. Всё, что нужно, давно придумано, и уже положено в папку Community. Но есть ведь и такие верстальщики, кто имеет мысли, что в конкретном процессе хорошо бы улучшить. Но не знает, как это сделать. Таким людям эти две книги точно помогут. А какой кайф, какое счастье видеть, что у тебя это получается! Скриптинг — это своего рода квинтэссенция творчества. Но если у конструктора от чертежа до натурного проекта проходят недели, то тут между идеей и реализацией очень небольшое время. А если привлечь к решению задач ChatGPT, то результаты будут поразительными! Но надо обязательно понимать, что хочется получить, и какие инструменты для этого потребуются. Искусственный интеллект может отвечать на хорошо сформулированные запросы, а чтобы хорошо формулировать, надо быть в теме.

Есть такая идея, что якобы чтобы стать мастером в каком-то деле, нужно 10 тысяч часов практики. Возможно, это так. Но хорошо подобранные средства обучения всегда ускоряют процесс. Если бы 20 лет назад у меня были такие книги, мне было бы много проще и быстрее продвигаться в деле написания скриптов для индизайна.

Знаете, на что похожи первые успехи в деле создания скриптов? Вы видели видео, как дальтоники впервые видят краски мира? В архиве к этим книгам есть ссылка на него. Похожесть в том, что как те люди, до этого видевшие мир в оттенках серого цвета, вдруг видят его в ярких цветах, и это шквал эмоций, так и ты ещё вчера не знал про себя, что ты вообще можешь сделать программу, и вдруг — вот, она есть и работает! Тут, конечно, у вас будут не слёзы, как в том видео, а гордость, буквально крылья будто раскрываются. А если у вас есть крылья, не мешайте им расти. Не все скрипты, что вы сделаете, будут идеальными. У меня немало скриптов для себя, не на показ. Но если продолжить метафору с крыльями, то чем больше в крыле перьев, тем оно шире, сильнее. И тут перья — это сделанные вами скрипты. Берегите их все. Со временем из них получатся фрагменты для сбора новых программ.

Путь к настоящему мастерству не быстрый. Идти придётся долго. Но зачем идти, если есть у вас есть крылья? Желаю вам удачи в освоении и применении скриптов!

Михаил Иванюшин
https://dotextok.ru | dotextok@gmail.com | 2025

P.S. Этот «двухтомник» стоит столько, сколько один мой скрипт. В июне 2024 года случился инсульт. 24 дня меня возвращали к нормальной жизни. Удалось. И после выписки я повысил стоимость программ. Потому что хоть и выписали, но почти 10 тысяч ежемесячно уходит на лекарства. И я даже не сопротивляюсь. Такие дела.

Оглавление

Введение **1**
Что такое ExtendScript 1
В чём разница между ExtendScript и JavaScript. 2
Где хранить и запускать скрипты 3
Как искать объекты в DOM InDesign 4

Глава 1. Основы DOM InDesign **7**
Структура документа InDesign: документ → страницы → текстовые фреймы → абзацы → символы 7
Объекты и свойства: как читать и изменять 9
Навигация по объектам документа 11
Введение в .parent, .appliedObjectStyle, .texts, .allPageItems 13
Получение всех текстовых рамок на странице 15
Поиск объектов с нужными стилями 16

Глава 2. Работа с текстом **18**
Абзацы и символы: как применять стили 18
Поиск и замена текста 19
Работа с grep-выражениями. 20
Управление стилями через скрипт 22
Автоматическая замена кавычек и дефисов 24
Назначение стилей заголовкам по шаблону 25
Выделение абзацев без стиля 26

Глава 3. Работа с таблицами **28**
Создание и редактирование таблиц 28
Настройка строк, ячеек, отступов 29
Работа с таблицами внутри текста 31
Преобразование CSV в таблицу 33
Выравнивание отступов во всех ячейках 34
Поиск пустых строк и удаление 36

Глава 4. Работа с изображениями и графикой. **37**
Вставка изображений 37
Управление рамками: размеры, обтекание 38
Проверка ссылок 41
Автоматическая замена всех заглушек на реальные изображения 42
Проверка разрешения всех картинок 44
Массовое выравнивание и кадрирование изображений. 45

Глава 5. Управление страницами и мастерами 49

Добавление, удаление, переименование страниц 49
Применение мастер-страниц 50
Создание шаблонов для обложек, глав и т.д. 52
Автоматическое создание колонтитулов 55
Переименование страниц с содержанием 58
Генерация оглавления из стилей 60

Глава 6. Работа со стилями 62

Поиск стилей по имени 62
Массовое изменение свойств стилей в InDesign 64
Удаление неиспользуемых стилей 65
Объединение дублирующихся стилей 65
Создание стиля таблицы программно 67
Применение объектного стиля к найденным элементам 68

Глава 7. Верстка данных (Data Publishing) 70

Импорт данных из CSV 70
Импорт данных из JSON 71
Импорт данных из XML 73
Вставка контента по шаблону 74
Использование переменных, маркеров и тегов 76
Верстка прайс-листа по шаблону 78
Автоматизация каталога товаров 79
Генерация персонализированных сертификатов 82

Глава 8. Проверка и очистка документа 85

Поиск ошибок: переполнение, пустые рамки 85
Удаление пустых страниц 87
Проверка стилей и объектов 88
Скрипт «Preflight»: 10 проверок за секунду 89
Удаление невидимых объектов 91
Подсветка нестандартизированных стилей 92

Глава 9. Диалоги и интерфейсы 95

Вводные диалоги Window, Dialog, Palette 95
Выпадающие списки, чекбоксы, текстовые поля, радиокнопки 97
Многостраничное окно 98
Сохранение пользовательских настроек 101
Простая форма для вставки текста с параметрами 102
Интерфейс выбора пресета экспорта 105
Диалог для настройки таблицы 107

Глава 10. Экспорт и автоматизация публикации 110
Экспорт в PDF, JPG, EPUB 110
Экспорт в EPUB вариант Fixed Layout 113
Управление настройками экспортных пресетов 115
Массовый экспорт с переименованием 117
Экспорт всех страниц по одной в JPG 119
Пакетный экспорт глав в отдельные PDF 121
Скрипт публикации для печати и веба 123

Приложение 125
Часто используемые фрагменты кода 125
Список DOM-объектов InDesign с описанием 173
Вопросы, на которые ответил ChatGPT 175
Что делать, если ChatGPT отвечает неправильно 203

В архиве есть большинство скриптов из глав книги. Их названия совпадают с номерами страниц, где они размещены.

Введение

Что такое ExtendScript
В чём разница между ExtendScript и JavaScript
Где хранить и запускать скрипты
Как искать объекты в DOM InDesign

Что такое ExtendScript

ExtendScript — это диалект JavaScript, разработанный Adobe, предназначенный для автоматизации и расширения возможностей приложений Adobe Creative Cloud, таких как InDesign, Illustrator, Photoshop и другие.

Ключевые особенности ExtendScript

Характеристика	Описание
Основан на JavaScript	Базируется на спецификации ECMAScript 3, но со своими расширениями
Поддерживает объекты Adobe	Использует объектные модели (DOM) конкретных приложений, например app.activeDocument в InDesign
Встроен в приложения Adobe	Можно запускать напрямую из панели Scripts в InDesign, Illustrator и др.
Использует ExtendScript Toolkit (устаревший) или Visual Studio Code	ExtendScript Toolkit — старое окружение, сейчас рекомендуют использовать VS Code + UXP/CEP
Доступ к файловой системе	Позволяет читать, записывать, создавать и удалять файлы и папки
Имеет систему вывода и отладки	Команда $.writeln() или alert() помогает отлаживать скрипты

Что можно делать с ExtendScript в InDesign?

Категория	Примеры автоматизации
Текст	Поиск/замена, применение стилей, проверка ошибок
Таблицы	Автозаполнение, выравнивание, форматирование
Графика	Массовая вставка изображений, обрезка
Страницы	Добавление, удаление, применение мастеров
Экспорт	PDF, JPEG, EPUB с заданными параметрами
Интерфейс	Создание диалогов и меню с выбором опций

Чем ExtendScript отличается от обычного JavaScript?

ExtendScript	JavaScript (в браузере)
Доступ к Adobe API	Нет доступа к DOM браузера
Файловая система доступна	Ограниченный доступ в браузерах
`alert()` используется	`console.log()` в браузере
`$.writeln()` вывод в консоль ExtendScript Toolkit	Не работает в обычных браузерах
Adobe DOM: `app.documents`	DOM браузера: `document.body`

В чём разница между ExtendScript и JavaScript

Разница между **ExtendScript** и **JavaScript** — в первую очередь в **среде исполнения** и **возможностях API**. Вот полное сравнение по ключевым параметрам:

1. Среда исполнения

Параметр	ExtendScript	JavaScript (в браузере)
Где работает	Внутри приложений Adobe (InDesign, etc.)	В браузерах (Chrome, Firefox...)
Движок исполнения	ExtendScript Engine (на базе ES3)	V8, SpiderMonkey, etc. (ES6+)
Разработка	ExtendScript Toolkit / VS Code + CEP/UXP	Chrome DevTools, VS Code
Интерфейс пользователя	`Dialog`, `Window`, `alert()` (Adobe UI)	HTML + CSS

2. Доступ к API

Возможности	ExtendScript	JavaScript в браузере
Adobe DOM (например, `app.activeDocument`)	Полный доступ	Нет доступа
Системные файлы (`File`, `Folder`)	Есть доступ	Ограничен (из соображений безопасности)
HTML/DOM браузера	Нет доступа	Полный доступ
Поддержка сетевых запросов (`fetch`)	Нет нативно	Да

3. Язык и синтаксис

Параметр	ExtendScript	JavaScript (современный)
Версия ECMAScript	ECMAScript 3	ECMAScript 6 и выше
`let`, `const`, стрелочные функции	Не поддерживаются	Поддерживаются
Модули (import/export)	Нет поддержки	Да
Поддержка классов	Ограничена (через функции)	Полноценная

4. Примеры использования

ExtendScript:

```
var doc = app.activeDocument;
doc.pages[0].textFrames[0].contents = "Привет из ExtendScript!";
```

JavaScript в браузере:

```
document.querySelector("h1").textContent = "Привет из JavaScript!";
```

5. Типичные задачи

Задача	ExtendScript	JavaScript (в браузере)
Автоматизация верстки в InDesign	Да	Нет
Управление HTML-страницами	Нет	Да
Создание пользовательских диалогов	Через `Window`, `Dialog`	Через HTML/CSS
Модификация системных файлов	Можно читать/писать `File`, `Folder`	Ограничено (sandbox)

Вывод:

ExtendScript	JavaScript
Инструмент для автоматизации Adobe	Инструмент для работы в вебе
Ориентирован на InDesign, Illustrator	Ориентирован на HTML-страницы
Старый, но мощный API для профи	Современный, гибкий, повсюду

Если ты хочешь ускорить работу в InDesign, тебе нужен ExtendScript. Если хочешь делать интерактивные сайты, тебе нужен JavaScript (современный).

Где хранить и запускать скрипты

ExtendScript-файлы (скрипты .jsx) для Adobe InDesign могут храниться в нескольких стандартных папках в зависимости от операционной системы и цели использования:

Стандартные папки для хранения скриптов:

macOS

Пользовательские скрипты:
~/Library/Preferences/Adobe InDesign/[версия]/[язык]/Scripts/Scripts Panel/
Системные скрипты (доступны всем пользователям):
/Applications/Adobe InDesign [версия]/Scripts/Scripts Panel/

Windows

Пользовательские скрипты:
C:\Users\[имя_пользователя]\AppData\Roaming\Adobe\InDesign\[версия]\[язык]\Scripts\Scripts Panel\
Системные скрипты:
C:\Program Files\Adobe\Adobe InDesign [версия]\Scripts\Scripts Panel\

66. Пример изменения названия кнопки при нажатии

```
var win = new Window ("dialog", "Пример изменения названия кнопки");
var button = win.add("button", [0,0,250, 28], "Число оставшихся нажатий: 3");
var count = 3;
button.onClick = function () { // button.onClick
count = count-1;
var nameButton = "Число оставшихся нажатий: " + count;
button.text = nameButton;
if (count == 0) {
    alert("Число нажатий исчерпано.");
    win.close();
    }
} // button.onClick
///
win.show();
```

67. Варианты программ добавления нулей в начале числа

```
// pad2 - добавление нуля, если число меньше 10
function pad2(number) {
    number = (number < 10 ? '0' : '') + number;
    number = number.substring(0,2);
    return number;
}
// другой вариант pad2
function pad2 (n) {
 return ('0'+n).slice(-2);
}

// Функция pad2 спользуется, например, для оформления времени. Число часов и минут всегда один или два разряда.
timeHM = pad2(hours) + ":" + pad2(minutes);

//Аналогичное решение для чисел разрядностью один, два, три знака:
function pad3 (n) {
 return ('00'+n).slice(-3);
}

// Функция добавления нулей в начале числа, работающая с числами от 1 до 100 млн.
function addZero (n) {
var for1digt = "00000000";
var for2digt = "0000000";
var for3digt = "000000";
var for4digt = "00000";
```

```
var for5digt = "0000";
var for6digt = "000";
var for7digt = "00";
var for8digt = "0";

if (n <10) return (for1digt + n);
if (n >9 && n <100) return (for2digt + n);
if (n >99 && n <1000) return (for3digt + n);
if (n >999 && n <10000) return (for4digt + n);
if (n >9999 && n <100000) return (for5digt + n);
if (n >99999 && n <1000000) return (for6digt + n);
if (n >999999 && n <10000000) return (for7digt + n);
if (n >9999999 && n <100000000) return (for8digt + n);
return n; // очень большое число :))
}
```

68. Вершины многоугольника

```
/// число вершин полигона
var selP = app.selection[0];
var numCorners = selP.paths[0].pathPoints.length;
////
var obj = app.selection[0];
var objPath = obj.paths[0];
$.writeln("-----------------");
var entirePathLength = objPath.entirePath.length;
var pathPointsLength = objPath.pathPoints.length;
for (i = 0; i < entirePathLength; i++) { // i++
    $.writeln(i + " | objPath.entirePath[i][0] = " +objPath.entirePath[i][0].toFixed(1)
+ " | objPath.entirePath[i][1] = " + objPath.entirePath[i][1].toFixed(1));
    } // i++
$.writeln("=======");
for (i = 0; i < pathPointsLength; i++) { // i++
    $.writeln(i + " | objPath.pathPoints[i].anchor[0] = " +objPath.pathPoints[i].
anchor[0].toFixed(1) + " | objPath.pathPoints[i].anchor[1] = " + objPath.
pathPoints[i].anchor[1].toFixed(1));
    } // i++
```

результат одинаковый — и entirePath, и pathPoints адресуются к одним и тем же точкам на контуре

69. Выделение строк таблицы одна за другой

Задача: выделить всю таблицу, в которой курсор.

Достаточно выделить первую строку таблицы, а потом последнюю с вариантом SelectionOptions.ADD_TO.

И вся таблица сразу выделится. Это очень быстро работает.

Для примера добавлен медленный вариант -- функция selectRows, там добавление строка за строкой.
Он закомментирован.
На больших таблицах задержка уже заметна.

```
var sel = app.selection[0];
if (sel == null || sel.constructor.name != "InsertionPoint" || sel.parent.constructor.
name != "Cell") {
    alert("Курсор должен быть в ячейке.");
    exit();
    }
var table = sel.parent.parent;

/*
// медленный вариант
var endRowIndex = table.rows.length-1;
selectRows (table, 0, endRowIndex);
if (app.selection[0].constructor.name != "InsertionPoint") exit(); // чтобы не выде-
лять уже выделенное
*/
// быстрый вариант
app.selection = NothingEnum.NOTHING;
app.select(table.rows[0]);
app.select(table.rows[ table.rows.length-1], SelectionOptions.ADD_TO);
exit();

////
function selectRows (table, startRowIndex, endRowIndex) {  // selectRows
app.selection = NothingEnum.NOTHING;
var _row = table.rows;
app.select(_row[startRowIndex-1]);
for (var i=startRowIndex; i<endRowIndex;i++) { app.select(_row[i],
SelectionOptions.ADD_TO); }
} // selectRows
```

70. Варианты проверки выделения

```
sel = app.selection[0];
if (sel == null || (sel.constructor.name != "Character" && sel.constructor.name
!= "Word" && sel.constructor.name != "Paragraph" && sel.constructor.name !=
"Text" && sel.constructor.name != "TextColumn" && sel.constructor.name !=
"TextStyleRange")) {
    alert("Перед запуском скрипта должен быть выделен текст.",programTitul);
    exit();
    }
```

```
sel = app.selection[0];
if ((sel == null) || (sel.hasOwnProperty("texts") != true) || ((sel.
hasOwnProperty("texts") == true) && (sel.constructor.name == "TextFrame"))) { //
== null
    alert("Выделите текст для обработки. Поставьте курсор в текст для обработки
всего материала.",programTitul);
    exit();
    }  // == null

sel = app.selection[0];
if (sel.constructor.name == "TextFrame" || sel.hasOwnProperty("pointSize") != true
|| sel.constructor.name == "InsertionPoint") {
    alert("Должна быть выделена область текста.",programTitul);
    exit();
    }

var mySel = app.selection[0];
if (mySel == null || (mySel.constructor.name != "InsertionPoint" || mySel.parent.
constructor.name != "Story")) { // == null
    alert("Перед запуском скрипта курсор должен стоять в тексте статьи. Ничего
выделять не надо.",programTitul);
    exit();
    } // == null

var myStory = mySel.parentStory;
var myPara = mySel.paragraphs[0];
if ( myStory.length == mySel.index) {
    alert("Курсор не должен быть в конце статьи.",programTitul);
    exit();
    }

if ( myPara.characters[-1].parentTextFrames[0].parentPage == null || mySel.
parentTextFrames[0].parentPage == null ) { /// null null
    alert("Выбранный абзац должен быть полностью на странице.",programTitul);
    exit();
    } /// null null

// Если курсор в тексте, то выделение всего текста статьи
if (sel.constructor.name == "InsertionPoint") { story.characters.itemByRange(0,-1).
select(); sel = app.selection[0]; }
```

71. Горизонтальная линия в меню

Горизонтальная линия в меню на всю ширину окна

```
separator1 = win.add ("panel"); // Помещаем на экран горизонтальную линию
```

Вопросы, на которые ответил ChatGPT

Практика программирования скриптов для индизайна предполагает наличие знаний, относящихся к созданию скриптов на JavaScript это в первую очередь использование ExtendScript — диалекта JavaScript для индизайна. Но в практике программирования используются и такие особенности, которые есть в самом JavaScript. И обычно эти моменты в руководствах по программированию скриптов отсутствуют. Да и то, что есть в ExtendScript, часто требует дополнительной информации, а её нет.
Я сделал небольшую подборку ответов на те вопросы, которые до появления систем ИИ для меня оставались не до конца понятными. Обычно одного ответа ChatGPT хватало для получения полного ответа.
То, что информации иногда просто нет, подтверждает и ChatGPT, в разговоре про EventListener была такая мысль: "*Не все события документированы, но они работают*".

Об EventListener

Расскажи про использование EventListener в скриптах для InDesign. Какие обработчики событий поддерживаются, как их программировать?

В ExtendScript для Adobe InDesign **EventListener** используется *для прослушивания событий* и *реакции на действия пользователя или приложения*, например:

- открытие документа,
- изменение объекта,
- бездействие (idle),
- клик мыши,
- закрытие окна и т. д.

Основной синтаксис

```
someObject.addEventListener("eventName", handlerFunction);
```

или

```
someObject.addEventListener("eventName", function(event) {
    // обработка события
});
```

Поддерживаемые события (наиболее популярные)

Событие	Описание
afterOpen	Срабатывает после открытия документа
beforeClose	До закрытия документа
afterClose	После закрытия документа
afterSave	После сохранения
afterSelectionChanged	При изменении выделения

onIdle	Срабатывает при бездействии (через **IdleTask**)|
onInvoke	При вызове скрипта (в диалогах и меню)

Где можно вешать обработчики?

- **app** — глобальное приложение
- **app.documents[0]** — конкретный документ
- **textFrame**, **page**, **button** и т. д. — конкретные объекты

Примеры

Пример 1: Событие после открытия документа

```
app.addEventListener("afterOpen", function(ev) {
    alert("Документ открыт: " + ev.target.name);
});
```

Пример 2: Отслеживание изменения выделения

```
app.addEventListener("afterSelectionChanged", function(ev) {
    $.writeln("Изменилось выделение.");
});
```

Пример 3: Использование `IdleTask` с событием `onIdle`

```
var idle = app.idleTasks.add({ name: "IdleLogger", sleep: 1000 });
idle.addEventListener("onIdle", function() {
    $.writeln("Скрипт запущен при простое");
});
```

Пример 4: Удаление обработчика

```
var handler = function(ev) {
    $.writeln("Выделение изменилось");
};
app.addEventListener("afterSelectionChanged", handler);

// Позже:
app.removeEventListener("afterSelectionChanged", handler);
```

Важно знать

- Не все события документированы, но они работают.
- Некоторые события требуют перезапуска InDesign для корректного срабатывания (особенно связанные с интерфейсом).
- **Сохраняй ссылки на обработчики**, чтобы потом их можно было удалить.
- Обработчики, добавленные во время выполнения скрипта, не сохраняются между сеансами InDesign.

Примеры использования EventListener в InDesign

```
// 1. Обработчик после открытия документа
app.addEventListener("afterOpen", function(event) {
    alert("Документ открыт: " + event.target.name);
});
```

```
// 2. Обработчик изменения выделения
var onSelectionChange = function(event) {
    $.writeln("Выделение изменено: " + new Date());
};
app.addEventListener("afterSelectionChanged", onSelectionChange);

// 3. IdleTask с событием onIdle
var idleTask = app.idleTasks.add({ name: "IdleLogger", sleep: 2000 });
idleTask.addEventListener("onIdle", function(event) {
    $.writeln("InDesign простаивает: " + new Date());
});

// 4. Удаление обработчика (если нужно позже)
// app.removeEventListener("afterSelectionChanged", onSelectionChange);
```

Варианты EventListener

Есть два варианта указания обработчика события: someObject.addEventListener("eventName", handlerFunction) и someObject.addEventListener("eventName", function(event). В чём отличие handlerFunction от function(event) ?

Отличие между двумя способами указания обработчика:

Вариант 1: Именованная функция

```
function handlerFunction(event) {
    // обработка события
}

someObject.addEventListener("eventName", handlerFunction);
```

Вариант 2: Анонимная функция

```
dEventListener("eventName", function(event) {
    // обработка события
});
```

Разница между ними

Критерий	Именованная функция (handlerFunction)	Анонимная функция (function(event))
Повторное использование	Да, можно вызывать повторно в коде	Нет, используется только один раз
Удаление обработчика	Да, можно удалить обработчик	Нет, невозможно удалить напрямую
Читаемость	Более понятна при отладке	Может быть сложнее для понимания
Область видимости	Глобальная/локальная — зависит от объявления	Локальная, часто внутри блока

Почему это важно?

Пример: невозможно удалить анонимную функцию

```
app.addEventListener("afterOpen", function(event) {
    $.writeln("Документ открыт");
});

app.removeEventListener("afterOpen", function(event) {
    $.writeln("Документ открыт"); // ← это другая функция!
}); //  не сработает
```

Пример: удаление именованного обработчика

```
function onOpen(event) {
    $.writeln("Документ открыт");
}
app.addEventListener("afterOpen", onOpen);
app.removeEventListener("afterOpen", onOpen); //  работает
```

Вывод

- Используйте *именованные функции*, если:
  - планируете удалять обработчик позже;
  - обработчик повторяется в коде;
  - нужен читаемый стек вызовов при ошибках.

- Используйте *анонимные функции*, если:
  - обработка однократна и простая;
  - не требуется удаление обработчика;
  - используете addEventListener внутри ограниченного контекста (модуль, IIFE и т. д.).

Пример: Разница между именованной и анонимной функцией как обработчиком событий

```
// === ИМЕНОВАННЫЙ ОБРАБОТЧИК ===
function namedHandler(event) {
    $.writeln("Сработал именованный обработчик: " + event.eventType);
}

// Добавление и удаление работают
app.addEventListener("afterSelectionChanged", namedHandler);

// Через 10 секунд удалим обработчик
$.sleep(10000); // Задержка 10 сек (для теста в InDesign)
app.removeEventListener("afterSelectionChanged", namedHandler);
$.writeln("Именованный обработчик удалён");

// === АНОНИМНЫЙ ОБРАБОТЧИК ===
app.addEventListener("afterSelectionChanged", function(event) {
```

```
    $.writeln("Сработал анонимный обработчик: " + event.eventType);
});

// Попытка удалить анонимный обработчик — не сработает!
app.removeEventListener("afterSelectionChanged", function(event) {
    $.writeln("Сработал анонимный обработчик: " + event.eventType);
}); // Это другая функция — не удалится
```

Использование анонимных функций

Расскажи про анонимную функцию в создании скриптов для InDesign

Анонимная функция — это функция (без имени), которую можно:
- присвоить переменной,
- передать как аргумент в другую функцию,
- сразу же выполнить (IIFE — немедленно вызываемая функция).

В InDesign-скриптах (ExtendScript) анонимные функции особенно полезны для инкапсуляции кода, обработки событий, и безопасного запуска логики без засорения глобального пространства имён.

Синтаксис

```
var myFunc = function() {
    // код
};
myFunc(); // вызов
```

Или немедленное выполнение:

```
(function() {
    $.writeln("Это анонимная самовызываемая функция");
})();
```

Примеры использования

1. Инкапсуляция кода (защита глобального пространства)

```
(function() {
    var localVar = "Пример";
    $.writeln("Локальная переменная: " + localVar);
})();

// $.writeln(localVar); // Ошибка: localVar не определена
```

2. Обработка в forEach

```
var items = ["Red", "Green", "Blue"];
items.forEach(function(color) {
```

```
    $.writeln("Цвет: " + color);
});
```

3. Временные функции в логике InDesign

```
var doc = app.activeDocument;
(function(doc) {
    if (doc.pages.length > 5) {
        $.writeln("В документе много страниц: " + doc.pages.length);
    }
})(doc);
```

Зачем использовать?

Локальность: переменные и функции не выходят за пределы блока.
Безопасность: меньше шансов переопределить существующие переменные или функции.
Чистота кода: удобна в модулях и автозапускаемых скриптах.

Учитывай:

- ExtendScript не поддерживает стрелочные функции (`()=>{}`).
- Работает на синтаксисе JavaScript до ES5 (без `let`, `const`, `class` и т. д.).

Другие примеры использования анонимных функций в InDesign

```
//Пример 1: Самовызываемая анонимная функция (IIFE)
(function() {
    $.writeln("Пример анонимной функции: самозапуск");
})();

// Пример 2: Анонимная функция в переменной
var logDocName = function() {
    if (app.documents.length > 0) {
        $.writeln("Открыт документ: " + app.activeDocument.name);
    } else {
        $.writeln("Документы не открыты.");
    }
};
logDocName();

// Пример 3: Анонимная функция в forEach
var colors = ["Cyan", "Magenta", "Yellow", "Black"];
colors.forEach(function(color) {
    $.writeln("Цвет: " + color);
});

// Пример 4: Локальный скрипт внутри InDesign (инкапсуляция)
(function(doc) {
    if (doc.pages.length > 1) {
        $.writeln("В документе несколько страниц: " + doc.pages.length);
    }
})(app.activeDocument);
```

Что делать, если ChatGPT отвечает неправильно

Искусственный интеллект всегда должен понимать из запроса, что это о работе индизайна а не основной версии JavaScript. Поэтому в своих запросах всегда было уточнение ExtendScript InDesign. Можно так сформулировать уточнение: target InDesign, главное, чтобы оно было. Потому что даже с ним бывали случаи, когда ИИ выдавал информацию, не относящуюся к индизайну.
Вот, например, был такой вопрос:
ExtendScript InDesign. Расскажи о JSON, как его применять в InDesign

Он вывел справку об этом инструменте, предложил пример скрипта для работы.
Привёл уточнение:
JSON.stringify() и JSON.parse() появились только в более новых версиях ExtendScript. Убедись, что ты работаешь в InDesign CC 2015+ (лучше 2020+).

Но это всё не работает, отладчик пишет: JSON is undefined
На форуме Adobe есть какие-то невнятные советы использовать библиотеки. Но это уже выходит за рамки данной книги. В главе 7 в «Импорт данных из JSON» приведён работающий вариант обработки формата JSON.

Все коды, которые предлагает Chat GPT, надо сразу проверять.
Как-то он придумал, что у абзаца есть границы:

```
var paragraphWidth = paragraphs[i].bounds[3] - paragraphs[i].bounds[1];
```

Достаточно было сказать, что границы есть у фрейма, а не абзаца, и код был переделан.

В другом примере ChatGPT вдруг решил, что можно сдвинуть абзац таким образом:

```
 // Сдвигаем абзац, сохраняя вертикальное положение
  currentParagraph.insertionPoints[0].horizontalOffset = newLeft;
```

Когда я отфутболил этот код:
Не работает.
Значение .horizontalOffset можно только читать, но нельзя изменять.
он предложил правильный вариант.

Каков итог? Искусственный интеллект не напишет скрипты за нас. Но он может быть очень хорошим помощником, если нам надо это сделать. Для этого понимать объектную модель индизайна, иметь представление о возможностях языка JavaScript и иметь навыки программирования.

JavaScript: краткое описание

https://dotextok.ru • 2025

При подготовке книги «JavaScript: скрипты для InDesign. Беседы с ChatGPT» в какой-то момент стало ясно, что размещение в этой книге всей справочной информации о командах, методах и других особенностях JavaScript сделает её несколько тяжёлой для чтения. Поэтому я принял решение сделать отдельную книгу, краткое описание JavaScript. Она тоже готовилась с помощью ChatGPT, все ответы искусственного разума проверялись, редактировались. Иногда требовались уточнения прежде чем поместить материал в книгу.

Это не был абсолютно новый материал для меня. С 2005 года накопилось много своих записей по разным вопросам. Но когда задался целью сделать краткий справочник, то основной вопрос был — что в нём должно быть и в какой очередности, что помещать, что уже неинтересно. И из вопросов к Chat GPT, и моих записей длиной 20 лет сложилась структура этого документа.

При создании скриптов я начинал работать и продолжаю в ExtendScript Toolkit (ESTK) — это среда разработки от Adobe, предназначенная для написания, отладки и запуска скриптов на ExtendScript (расширенная версия JavaScript) для приложений Adobe, включая InDesign, Illustrator и Photoshop. Сейчас Adobe якобы прекратила поддержку ESTK и рекомендует использовать Visual Studio Code. Утверждение «прекратила поддержку» весьма условное, ESTK продолжает работать точно так же, как и раньше. Информации об использовании Visual Studio Code в этой книге нет.

Очень важный для программирования скриптов вопрос — создание полноценных оконных интерфейсов — помещён в основную книгу. И коль скоро это электронный документ, в нём нет особого смысла делать ещё и индексный указатель, все термины ищутся обычным поиском.

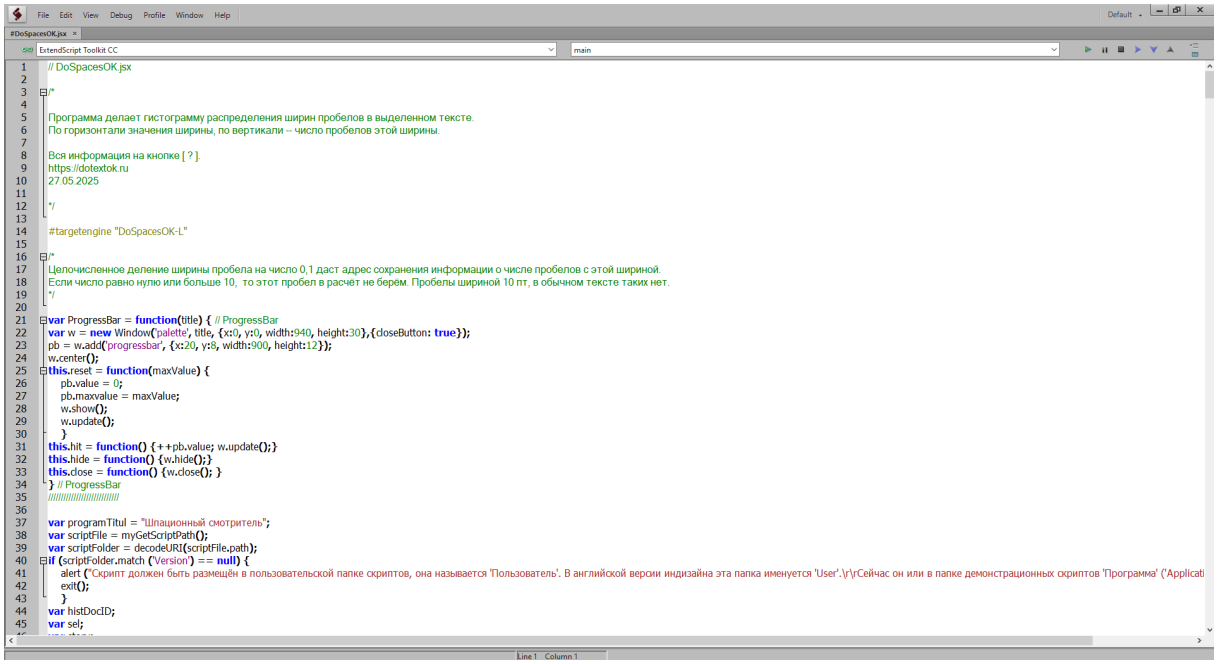
Оглавление

Отладчик программ ExtendScript Editor **1**
Переменные **6**
Основы работы с переменными 6
Преобразование одного типа в другой 7
Сравнение переменных 9
Определение типа переменной 12
Арифметические операции **15**
Логические операции **18**
Операции со строками **22**
Операции присваивания. **26**
Оператор with. **27**
Операторы условий (ветвления) **29**
if / else if / else. 29
Тернарный оператор **? :** 29
switch 30
Оператор try catch. 31
Циклы **34**
for — классический цикл 34
while — цикл с предусловием 34
do...while — цикл с постусловием 34
for...in — перебор свойств объекта. 35
Оператор continue. 36
Оператор break 37
Объекты **40**
Массивы **43**
Вложенные массивы **46**
Методы работы с массивами **48**
Функции **62**
Окна сообщений и диалогов **67**
Оператор alert() 67
Оператор confirm() 67
Оператор prompt() 68
Создание диалоговых окон интерфейса **69**
Регулярные выражения, общее описание **77**
Регулярные выражения, опции поиска **80**
Регулярные выражения, свойства **81**
Работа скриптов с файловой системой **82**
Основные различия между файловыми системами Windows и Mac: 83

Отладчик программ ExtendScript Editor

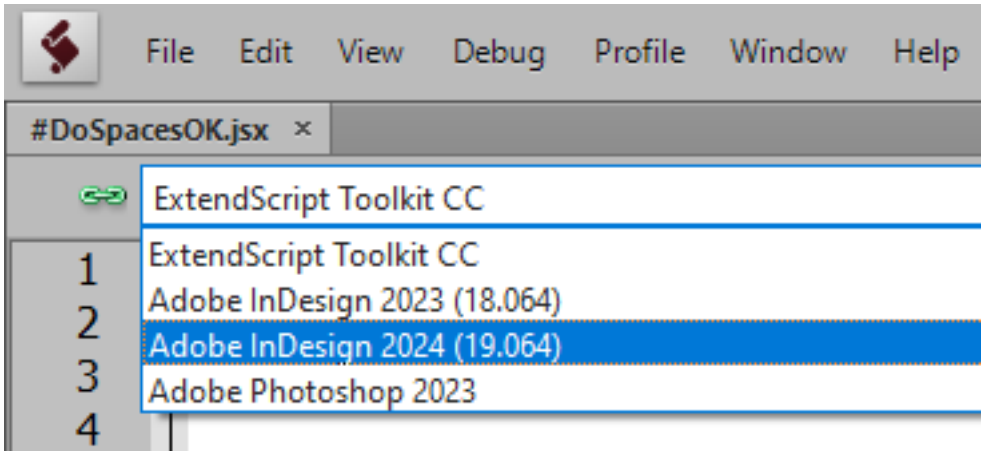
Открытие рабочего окна

Скрипты отлаживаются в специальной программе. Откройте панель скриптов, выберите нужный файл с расширением jsx и в контекстном меню, оно появится при нажатии на правую кнопку мыши выберите Показать в проводнике. В английской версии — Reveal in Explorer. Откроется папка, где этот скрипт. Двойной щелчок по нему откроет окно отладчика. Вот пример, как это может выглядеть:



Выбор активной программы индизайна

В левом верхнем углу список программ, которые могут работать с ESTK.



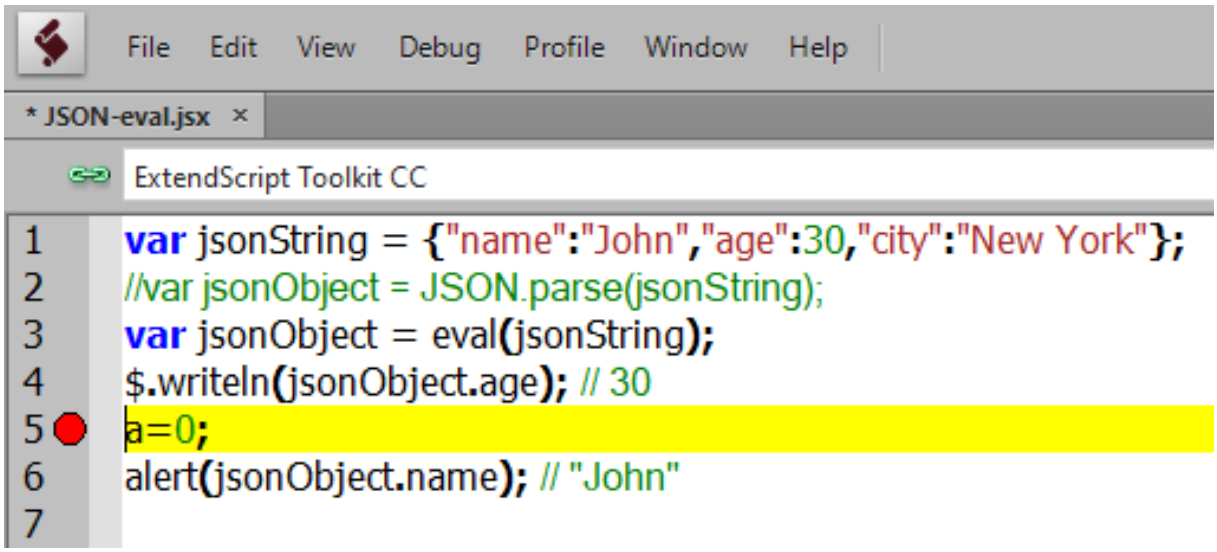
При первом запуске обычно активен вариант Extendcript Toolkit CC. Надо выбрать используемую сейчас программу индизайна. В данном случае это версия 2024.

Создание нового скрипта

Выберите File > New JavaScript (Ctrl+N), и откроется новое окно. В этом окне можно поместить подготовленный текст, или создать новый.

Простой пример отладки

Вот окно с небольшой программой и инструментами для её отладки:



Что тут есть:

- строка 2: эта строка отключена. Две прямых наклонных черты делают текст комментарием. Строка скрипта в теле программы осталась, но в исполнении не участвует.
- строка 4: оператор $.writeln выводит на консоль данные, указанные в круглых скобках. Доступ к окну консоли тут: Window > JavaScript Console.
- строка 5: щелчок справа от номера ставит в этом месте красный кружок. Это точка промежуточного останова программы. Запуск программы Debug > Run F5. В момент достижения точки останова строка будет отмечена жёлтым цветом, как показано на рисунке. В программе можно иметь несколько точек останова. Нажатие клавиши F5 запустит продолжение выполнения программы после достижения очередной точки останова. Выключение точки останова: Debug > Toggle Breakpoint F9.
- строка 6: вывод информации на экран. Каждый вывод на экран требует от пользователя нажатия ОК, и это может быть утомительно. Иногда целесообразнее собирать информацию на консоли, и затем анализировать её.

Другие возможности отладки

Останов при достижении переменной конкретного значения.

Например, скрипт в процессе работы изменяет счётчик count, и надо посмотреть результат после того, как в файле значение этого счётчика станет равным 17. Остановку на достижении этого условия можно сделать так:

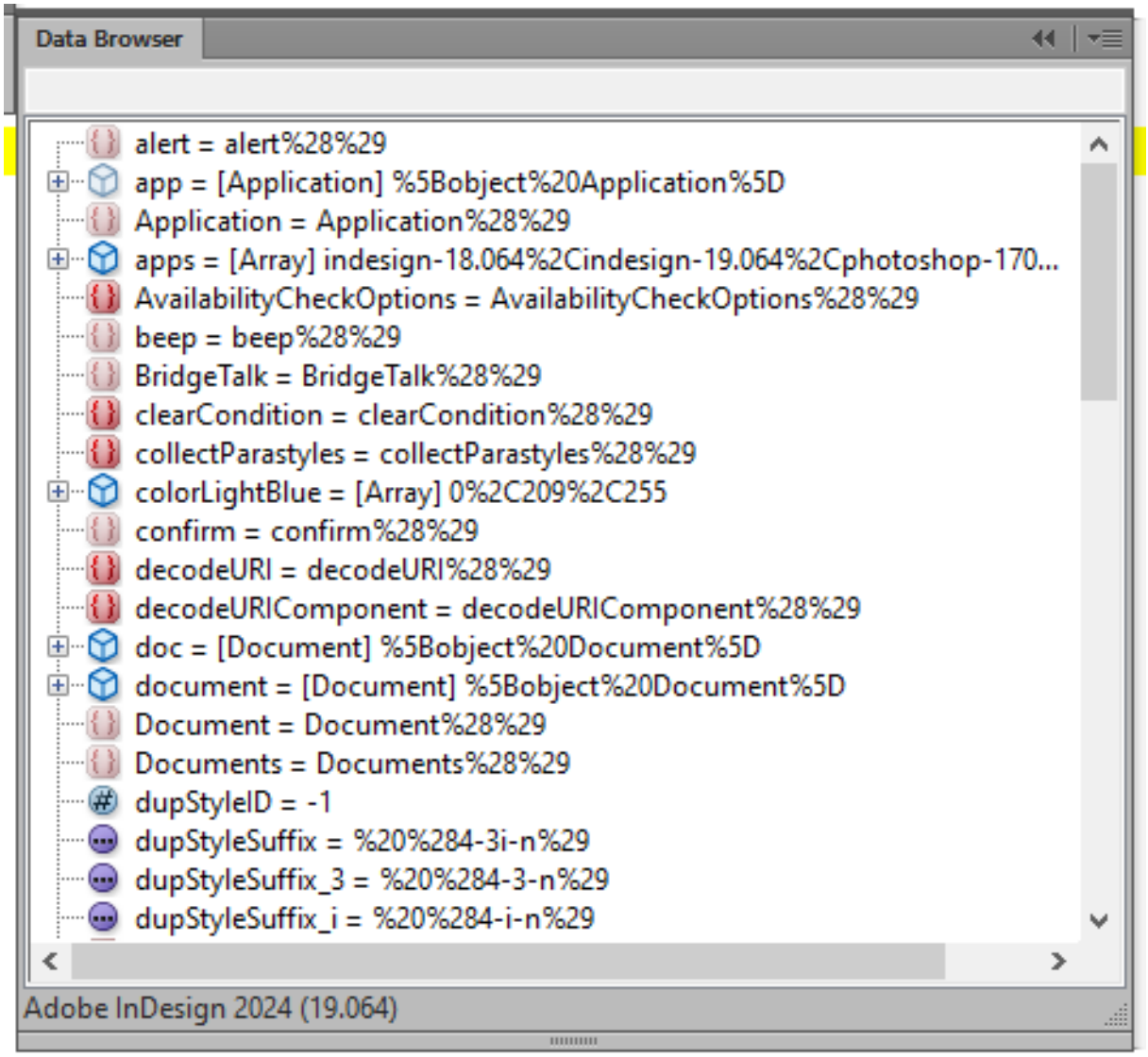
```
$.bp(count == 17)
```

Построчное прохождение программы

После достижения точки останова можно останавливаться после выполнения каждой строки программы. Тут есть два варианта: Debug > Step Into F11 и Debug > Step Over F10. Первый вариант предполагает выполнение абсолютно всех строк программы по мере их достижения. А второй запускает выполнение функции или подпрограммы, но в их строки не заходит. Тут важно, чтобы подпрограмма выполнилась, и после этого перейти к следующей строке основной программы. Нужна небольшая практика, чтобы понять эти различия.

Окно с объектной моделью индизайна

Если перейдёте по адресу Window > DataBrowser, то откроется такое окно:



Из него можно получить практически любую информацию об отлаживаемой программе. Но с ним надо научиться работать, и это процесс не быстрый.

Установка уровня детализации сообщений $.level

В ExtendedScript предусмотрены четыре уровня детализации сообщений об ошибках и предупреждениях, которые отображаются при выполнении скриптов:

0 – Только ошибки (выводятся только сообщения об ошибках, которые критически влияют на выполнение скрипта).

1 – Предупреждения (включаются предупреждения о потенциальных проблемах, но не блокируют выполнение).

2 – Информационные сообщения (выводятся сообщения для отслеживания нормального выполнения скрипта, не являющиеся ошибками).

3 – Полная информация (включает более подробные сообщения для отладки, показывающие каждое выполнение и изменение значений переменных).

Нужное значение указывается так:

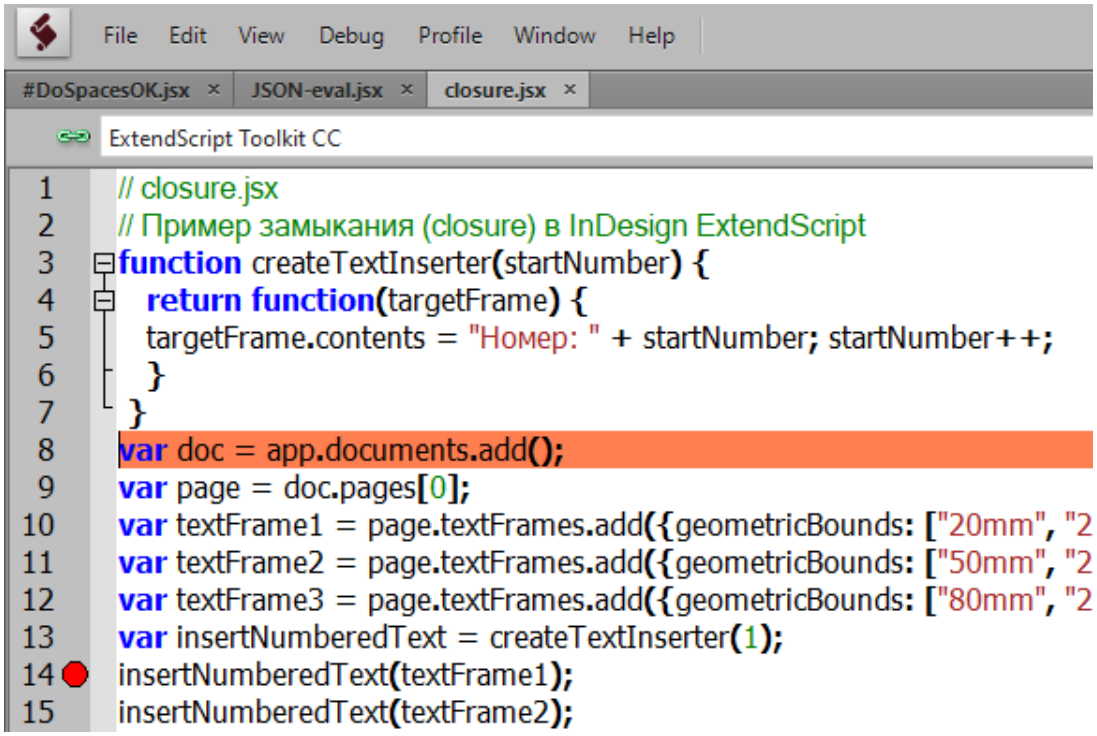
$.level = <одно из этих чисел>
Значение по умолчанию: 3.
На форумах программистов я видел рекомендации указывать значение этого параметра равным 1 или 2. Но в моей практике не было изменять этот параметр.

Со всеми возможностями этого окна вы познакомитесь поближе сами, когда дело дойдёт до реальной работы. И ChatGPT в этом поможет.

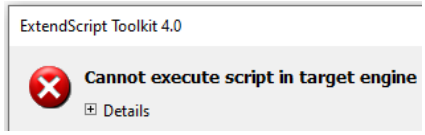
Что делать, если скрипт в отладчике не работает

Такое иногда бывает, и вот несколько советов, которые, надеюсь, помогут.

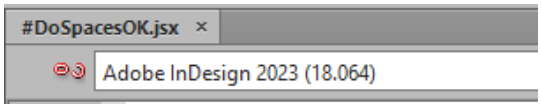
Одна из строк кода отмечена красным цветом



Причина: остался выбранным вариант ExtendedScript Toolkit. Выберите нужный вариант индизайна.
По этой же причине — остался отладчик вместо индизайна — будет появляться окно Cannot execute script.

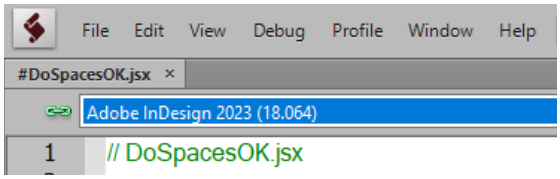


Значок разорванной цепочки напротив строки с вариантом программы



По каким-то причинам оборвалась связь скрипта и исполняемой программы. Например, приложение закрылось. После его повторной загрузки надо в этом окне выбрать сперва

ExtendedScript Toolkit, а затем нужную программу.



Значок зелёной цепочки означает, что скрипт готов работать с этим вариантом индизайна.

При запуске скрипта его окно диалога на экране отсутствует

Эта неожиданность бывает, когда в окне установки варианта окна стоит "palette". Для таких случаев надо всегда определять в скрипте #targetengine.

В сети можно найти такую рекомендацию по оформлению этой директивы:

```
#targetengine "session"
```

Не делайте так. Всегда в качестве параметра этой директивы указывайте название скрипта, например,

```
#targetengine "DoSpacesOK"
```

и тогда никогда не будет даже ничтожной вероятности получить неуловимую ошибку, когда разные программы используют одно и то же пространство для хранения своих параметров.

О работе с этой директивой есть ещё такой материал:

https://adobeindesign.ru/2011/03/10/chto-za-dzhinn-etot-targetengine/

Вёрстка должна быть в кайф.
Даже очень сложная.
Но без скриптов это невозможно.

https://dotextok.ru
Заходите! Тут очень много полезного для ускорения и облегчения кропотливой работы с текстом.

Если есть вопросы, пишите: dotextok@gmail.com