Р7-ОФИС

Руководство по работе с плагинами

Введение

Десктопные редакторы документов **Р7-Офис** поддерживают плагины: как визуальные, так и невизуальные. Разработчики могут использовать плагины, чтобы добавить в редакторы определенную функциональность, не относящуюся напрямую к форматам OOXML. Разработчикам предоставляется API для взаимодействия с редакторами документов.

Плагины могут быть визуальными и невизуальными. При использовании визуальных плагинов открывается окно для выполнения какого-либо действия или рисуется определенный интерфейс в панели редактора. Невизуальные плагины представляют собой кнопки, при нажатии которых производятся манипуляции с документом.

Помимо манипуляций с документом или обработки его содержимого плагин может реализовывать OLE-объект. Это единственный способ, дающий возможность стороннему разработчику получать доступ в форматную часть документа, т.е. записывать свою информацию не только во время работы редактора, но и непосредственно в файл.

Плагины можно добавить в любой редактор (редактор текстовых документов, таблиц или презентаций). В последующих разделах можно найти информацию об устройстве плагинов и о добавлении их в редакторы.

Структура плагинов

Каждый плагин для десктопной версии редакторов - это **zip-архив** директории с файлами. (подробную информацию о добавлении плагинов в редакторы можно найти в разделе "Установка плагинов в редакторах документов").

Папка должна содержать следующие файлы:

- **config.json** - файл конфигурации плагина. Содержит основные данные, необходимые для регистрации плагина в редакторах.
- **index.html** - точка входа плагина. Подключает файлы *config.json* и *pluginBase.js* (базовый файл для работы с плагинами).
- **pluginCode.js** - файл с кодом плагина. Содержит код JavaScript того плагина, который требуется подключить к редакторам.

Создание файла конфигурации плагина: config.json

Файл *config.json* - это файл конфигурации плагина, содержащий основные данные, необходимые для регистрации плагина в редакторах.

Ниже приводится пример кода для файла *config.json* (этот код используется для плагина **Шахматы**, но любой другой файл *config.json* создается аналогично):

```
{
"baseUrl": "",
"guid": "asc.{FFE1F462-1EA2-4391-990D-4CC84940B754}",
"name": "chess(fen)",
"variations": [
            {
                        "buttons": [
                                    { "text": "OK", "primary": true },
                                    { "text": "Cancel", "primary": false }
                        ],
                        "description": "chess",
                        "EditorsSupport": ["word", "cell", "slide"],
                        "icons": ["chess/icon.png", "chess/icon@2x.png"],
                        "initData": "",
                        "initDataType": "ole",
                        "initOnSelectionChanged": true,
                        "isInsideMode": false,
                        "isModal": true,
                        "isUpdateOleOnResize": true,
                        "isViewer": true,
                        "isVisual": true,
                        "url": "chess/index.html"
                        }
            ]
};
```

Имя	Описание	Тип
baseUrl	Путь к плагину. Все остальные пути вычисляются относительно этого пути. Если плагин установлен на сервер, там прописывается дополнительный параметр - путь к плагинам. Если *baseUrl == ""*, то используется путь ко всем плагинам.	строка

guid	Идентификатор плагина. **Должен** быть типа *asc.{uuid}*.	строка
name	Имя плагина, которое будет отображаться на панели плагинов.	строка
variations	Вариации плагина или "подплагины" - См. раздел "Вариации плагинов" ниже.	массив
variations.buttons	Перечень кнопок в окне плагина с возможностью скиновать. Используются только для визуальных плагинов в их собственном окне, например, *isVisual == true && isInsideMode == false*. Кнопки могут быть главными или не главными. Флаг primary влияет только на скин кнопки.	массив
variations.description	Описание плагина.	строка
variations.EditorsSupport	Для каких редакторов предназначен плагин (*"word"* - редактор текстовых документов, *"cell"* - редактор электронных таблиц, *"slide"* - редактор презентаций).	массив
variations.icons	Графические файлы иконок плагина, используемых в редакторах: для обычных экранов и иконки со вдвое большим разрешением для дисплеев Retina.	массив
variations.initData	Всегда равно *""*. Это данные, которые передаются плагину из редактора при запуске плагина (например, если *initDataType == "text"*, плагин при запуске будет получать выделенный текст).	строка
variations.initDataType	Тип данных, которые выделяются в редакторе и передаются плагину: *"text"* - текстовые данные, *"html"* - HTML-код, *"ole"* - данные OLE-объекта, *"none"* - никакие данные не передаются плагину.	строка

variations.initOnSelectionChanged	Определяет, отслеживает ли плагин изменение выделения текста в окне редактора.	логи-ческое значение
variations.isInsideMode	Определяет, должен ли плагин отображаться в панели редактора, а не в собственном окне (используется только для визуальных немодальных плагинов). Обязательно должно выполняться следующее правило: *isModal != isInsideMode*.	логи-ческое значение
variations.isModal	Определяет, надо ли открывать плагин в отдельном модальном окне (используется только для визуальных плагинов). Обязательно должно выполняться следующее правило: *isModal != isInsideMode*.	логи-ческое значение
variations.isUpdateOleOnResize	Определяет, нужно ли перерисовывать OLE-объект при изменении размера объекта в редакторе (векторная отрисовка). Используется только для OLE-объектов, то есть *initDataType == "ole"*.	логи-ческое значение
variations.isViewer	Определяет, доступен ли плагин, если документ доступен только в режиме просмотра.	логи-ческое значение
variations.isVisual	Определяет, является ли плагин визуальным (при их использовании открывается окно для выполнения какого-либо действия или рисуется определенный интерфейс в панели редактора) или невизуальным (представляют собой кнопки, при нажатии которых производятся манипуляции с документом).	логи-ческое значение
variations.url	Точка входа в плагин, то есть HTML-файл, подключающий файл *pluginBase.js* (базовый файл для работы с плагинами) и запускающий код плагина. См. раздел index.html для	строка

	получения подробной информации.	

Вариации плагина

Для чего плагину нужны вариации? Очень просто: плагин может не только выполнять какое-то действие, но и содержать настройки или окно "О программе". Например, плагин-переводчик: самому плагину не требуется визуальное окно для перевода, так как это выполняется нажатием одной кнопки, но настройки плагина (направление перевода) и окно "О программе" должны быть визуальными. Поэтому потребуются как минимум две вариации плагина (сам перевод и настройки), или три, если надо добавить окно "О программе" с информацией о плагине и его разработчиках или о программном обеспечении, использованном для создания плагина.

> Для правильной работы плагина файлы *.html* для **всех** вариаций должны быть помещены в корневую папку плагина вместе с файлом конфигурации *config.json*.

Создание основного HTML-файла плагина: index.html

Каждый плагин работает в своем iframe. Редактор подключает файл *index.html*, указанный в файле конфигурации плагина *config.json*. Файл *index.html* - это точка входа в плагин. Файл *index.html* подключает файл *pluginBase.js* - базовый файл, необходимый для работы с плагинами.

Ниже приводится пример кода для файла *index.html* (этот код используется для плагина **Шахматы**, но любой другой файл *index.html* создается аналогично):

```
<!DOCTYPE html>
<html>
	<head>
		<meta charset="UTF-8" />
		<title>Chess</title>
		<script type="text/javascript" src="../pluginBase.js"></script>
		<script type="text/javascript" src="chess.js"></script>
	</head>
	<body style="width: 100%; height: 100%; margin: 0; padding: 0; overflow: hidden;">
		<div id="chess" style="margin: 0; padding: 0;"></div>
	</body>
</html>
```

Раздел *<head>...</head>* содержит ссылки на все скрипты и таблицы стилей, необходимые для правильной работы плагина (как локальные, так и удаленные, если плагин их использует). Этот раздел также содержит ссылку на базовый файл *pluginBase.js,* необходимый для правильной работы плагинов с редакторами и содержащий базовый метод работы плагинов.

Раздел Раздел *<body>...</body>* может содержать теги * ... * с заполнителями, куда будут вставлены компоненты плагина. Поведение этих компонентов плагина описывается в самом файле *pluginCode.js*.

Написание кода плагина

Основной код плагина находится в файле *.js,* описывающем, какие действия и каким образом должен выполнять плагин. Этот файл размещается в корневой папке плагина вместе с файлами *config.json* и *index.html*.

Как это работает?

В коде каждого плагина содержится объект *window.Asc.plugin*, который, в свою очередь, имеет несколько методов для взаимодействия с редакторами документов, таблиц и презентаций.

Для работы плагина разработчик должен определить два обязательных метода для объекта *window.Asc.plugin: window.Asc.plugin.init* и *window.Asc.plugin.button*. После этого используется метод *window.Asc.plugin.callCommand* для передачи данных в редакторы с помощью встроенных функций API Генератора документов.

Если плагин работает с OLE-объектом, для управления им используется метод *window.Asc.plugin.executeCommand*.

Рассмотрим, как это сделано в плагине *helloworld.js*:

```
(function (window, undefined) {
	window.Asc.plugin.init = function () {
		this.callCommand(function() {
			var oDocument = Api.GetDocument();
			var oParagraph = Api.CreateParagraph();
			oParagraph.AddText("Hello world!");
			oDocument.InsertContent([oParagraph]);
		}, true);
	};
	window.Asc.plugin.button = function (id) {
	};
})(window, undefined);
```

При инициализации плагина (*window.Asc.plugin.init = function () {...}*) редактор формирует абзац с фразой 'Hello World', а затем использует API Генератора документов для создания документа с этим текстом (с помощью метода *window.Asc.plugin.callCommand - this.callCommand(function() {...})*).

> Пожалуйста, обратите внимание, что функции Генератора документов уже включены в редакторы: это то, как плагины взаимодействуют с редакторами. Отдельная версия Генератора документов потребуется, только если вы хотите создать документ, не используя редакторы.

Единственная кнопка 'OK' (*window.Asc.plugin.button = function (id) {...}*) используется для создания текста и завершения работы с плагином.

Таким же образом можно создать любой другой плагин. В настоящее время плагины поддерживают не только передачу текста или форматированного текста в редакторы, но и встраивание OLE-объектов (например, плагин YouTube, встраивающий видео с YouTube на страницу редактора).

объект window.Asc.plugin

Описание

Объект, создаваемый при запуске плагина. Имеет несколько методов, некоторые из них необязательные и используются только в случае необходимости.

Методы и свойства

Имя	Описание	Тип	Наличие
button	Кнопки, используемые с плагином и поведение плагина при нажатии на кнопку. Этот метод вызывается при нажатии на любую из кнопок плагина.	функция	обяза-тельный
callCommand/executeCommand	Метод *window.Asc.plugin.callCommand* используется для передачи данных в редактор. Заменил метод *window.Asc.plugin.executeCommand* при работе с текстами, чтобы упростить	функция	обяза-тельный

	синтаксис сценария, который требуется передать редакторам. Метод *window.Asc.plugin.executeCommand* преимущественно используется для работы с OLE-объектами.		
executeMethod	Используется для выполнения с помощью плагина определенных методов редактора.	функция	необязательный
info	Содержит все сведения о редакторе, в котором используется плагин.	объект	необязательный
init	Данные, передаваемые в плагин, и информация о том, что и каким образом нужно с ними сделать. Этот метод вызывается при старте плагина.	функция	обязательный
onExternalMouseUp	Определяет действие, которое должно выполняться, когда кнопка мыши отпущена вне iframe плагина.	функция	необязательный
onMethodReturn	Возвращает результат ранее выполненного метода.	функция	необязательный

Пример

```
(function(window, undefined){
        var text = "Hello world!";
        window.Asc.plugin.init = function() {
                Asc.scope.text = text;
                this.callCommand(function() {
                        var oDocument = Api.GetDocument();
                        var oParagraph = Api.CreateParagraph();
                        oParagraph.AddText(Asc.scope.text);
                        oDocument.InsertContent([oParagraph]);
```

```
            }, true);
        };
        window.Asc.plugin.button = function(id)
        {
        };
})(window, undefined);
```

window.Asc.plugin.button(id)

Описание

Кнопки, используемые с плагином и поведение плагина при нажатии на кнопку. Этот метод вызывается при нажатии на любую из кнопок плагина.

Параметры

Имя	Описание	Тип
id	Индекс кнопки в массиве *"buttons"* файла конфигурации плагина *config.json*. Если *id == -1*, считается, что нажат крестик 'Закрыть окно' или работа плагина прервана.	число

Пример

```
window.Asc.plugin.button = function (id) {
        this.executeCommand("close", '');
};
```

window.Asc.plugin.callCommand(fCommand, isClose)

Описание

Новый метод, используемый для передачи данных в редактор. Заменил метод *executeCommand* при работе с текстами, чтобы упростить синтаксис сценария, который требуется передать редакторам с помощью API Генератора документов. Позволяет плагину передавать структурированные данные, которые можно вставить в итоговый файл (форматированные абзацы, таблицы, части текста, отдельные слова и т.д.).

> Команды Генератора документов можно использовать только для создания содержимого и его вставки в редактор документов (используя метод *Api.GetDocument().InsertContent(...)*). Это ограничение введено из-за возможности совместного редактирования в онлайн-редакторах. Данное ограничение не применяется, если вы создаете плагин для десктопных редакторов, работающих с локальными файлами.

Параметры

Имя	Описание	Тип
fCommand	Команда кода JavaScript, формирующая структурированные данные, которые можно вставить в итоговый файл (форматированные абзацы, таблицы, части текста, отдельные слова и т.д.) и которые передаются в редакторы. Команда должна быть совместима с синтаксисом Генератора документов.	строка
isClose	Определяет, должно ли окно плагина закрываться после исполнения кода или оставаться открытым и ожидать других команд или действий. Значение *true* используется, чтобы закрыть окно плагина после выполнения функции, заданной в параметре *fCommand*. Значение *false* используется, чтобы выполнить команду и оставить окно открытым, в ожидании следующей команды.	логи-ческое значение

Метод *window.Asc.plugin.callCommand* выполняется в собственном контексте, изолированно от других данных кода JavaScript. Если требуется передать ему какие-то параметры или другие данные, для этого надо использовать объект *Asc.scope*.

Пример

```
window.Asc.plugin.init = function () {
        this.callCommand(function() {
                var oDocument = Api.GetDocument();
                var oParagraph = Api.CreateParagraph();
                oParagraph.AddText("Hello world!");
                oDocument.InsertContent([oParagraph]);
        }, true);
};
```

window.Asc.plugin.executeCommand(type, command)

Описание

Используется для передачи данных в редактор. Этот метод преимущественно используется для работы с OLE-объектами и по-прежнему поддерживается для использования с текстом в целях совместимости с более ранними версиями плагинов.

Второй параметр - это код JavaScript для работы с API Генератора документов, позволяющий плагину передавать структурированные данные, которые можно вставить в итоговый файл (форматированные абзацы, таблицы , части текста и отдельные слова и т.д.).

> Команды Генератора документов можно использовать только для создания содержимого и его вставки в редактор документов (используя метод *Api.GetDocument().InsertContent(...)*). Это ограничение введено из-за возможности совместного редактирования в онлайн-редакторах. Данное ограничение не применяется, если вы создаете плагин для десктопных редакторов, работающих с локальными файлами.

Параметры

Имя	Описание	Тип
type	Тип команды. Значение *"close"* используется, чтобы закрыть окно плагина после выполнения функции, заданной в параметре *command*. Значение *"command"* используется, чтобы выполнить команду и оставить окно открытым, в ожидании следующей команды.	строка
command	Команда кода JavaScript, формирующая структурированные данные, которые можно вставить в итоговый файл (форматированные абзацы, таблицы , части текста и отдельные слова и т.д.) и которые передаются в редакторы. Команда должна быть совместима с синтаксисом Генератора документов.	строка

При создании/редактировании OLE-объектов для работы с ними используются два расширения:

- *Api.asc_addOleObject(window.Asc.plugin.info)* - используется для создания OLE-объекта в документе;
- *Api.asc_editOleObject(window.Asc.plugin.info)* - используется для редактирования созданного OLE-объекта.
- При создании/редактировании объектов их свойства можно передавать объекту *window.Asc.plugin.info,* отвечающему за то, как выглядит объект.

Пример с OLE-объектом

```
window.Asc.plugin.button = function (id) {
        var _info = window.Asc.plugin.info;
        var _method = (_info.objectId === undefined) ? "asc_addOleObject" : "asc_editOleObject";
         _info.width = _info.width ? _info.width : 70;
        _info.height = _info.height ? _info.height : 70;
        _info.widthPix = (_info.mmToPx * _info.width) >> 0;
        _info.heightPix = (_info.mmToPx * _info.height) >> 0;
        _info.imgSrc = window.g_board.getResult(_info.widthPix, _info.heightPix).image;
        _info.data = window.g_board.getData();
        var _code = "Api." + _method + "(" + JSON.stringify(_info) + ");";
        this.executeCommand("close", _code);
};
```

Пример с текстом (не используется, но поддерживается в целях совместимости)

```
window.Asc.plugin.init = function () {
        var sScript = 'var oDocument = Api.GetDocument();';
        sScript += 'oDocument.CreateNewHistoryPoint();';
        sScript += 'oParagraph = Api.CreateParagraph();';
        sScript += 'oParagraph.AddText(\'Hello word!\');';
        sScript += 'oDocument.InsertContent([oParagraph]);';
        window.Asc.plugin.info.recalculate = true;
        this.executeCommand("close", sScript);
};
```

window.Asc.plugin.executeMethod (Name, [args], callback)

Описание

Метод объекта *window.Asc.plugin*, который позволяет с помощью плагина выполнять определенные методы редакторов. *Name* - это имя определенного метода, который должен выполняться, *[args]* - это аргументы используемого метода (если они есть), а *callback* - это результат, возвращаемый методом. Последний параметр необязательный,

если он опущен, то для возвращения результата выполнения метода будет использоваться метод *window.Asc.plugin.onMethodReturn*.

Ниже приводятся подробные сведения о доступных методах объекта *window.Asc.plugin.executeMethod*.

Методы и свойства

На данный момент доступны следующие 6 методов, которые могут быть таким образом выполнены:

Имя	Описание
InsertAndReplaceContentControls	Этот метод добавляет элемент управления содержимым с данными.
RemoveContentControls	Этот метод позволяет удалить несколько элементов управления содержимым.
GetAllContentControls	Этот метод позволяет получить информацию о всех элементах управления содержимым, которые добавлены на страницу.
AddContentControl	Этот метод позволяет добавить в документ пустой элемент управления содержимым.
RemoveContentControl	Этот метод позволяет удалить элемент управления содержимым, но при этом оставить данные, которые в нем были.
GetCurrentContentControl	Этот метод позволяет получить идентификатор выделенного элемента управления содержимым.

> Для корректной работы плагина необходимо дожидаться выполнения очередного метода прежде чем выполнять новый метод.

window.Asc.plugin.executeMethod ("InsertAndReplaceContentControls", [args], callback)

Описание

Этот метод добавляет элемент управления содержимым с данными. Эти данные задаются с помощью js-кода для Генератора документов, или просто ссылкой на документ, к которому предоставлены соответствующие права доступа.

Использование

Использовать метод надо так:

```
window.Asc.plugin.executeMethod("InsertAndReplaceContentControls", [obj]);
```

Где *obj* - это массив из JSON:

```
[
	{
		"Props": {
			"Id": Number,
			"Tag": "{String}",
			"Lock": Number
		},
		"Script": "var oDocument = Api.GetDocument();var oParagraph = Api.CreateParagraph();oParagraph.AddText('Hello world!');oDocument.InsertContent([oParagraph]);"
	}
]
```

Каждый новый элемент в массиве будет создавать новый элемент управления содержимым.

```
[ {"Props": {...}, "Url": "http://"}, {"Props": {...}, "Url": "http://."}, {...}, {...}]
```

Ключ *Props* может иметь следующие значения:

- "Id" (например, *{"Id": 2}*) - уникальный идентификатор для элемента управления содержимым. Его можно использовать для того, чтобы найти конкретный элемент управления содержимым и ссылаться на него в своем коде.

- "Tag" (например, *{"Tag": "String"}*) - тег для элемента управления содержимым. Один тег может быть присвоен нескольким элементам управления содержимым для того, чтобы можно было ссылаться на них в своем коде.
- "Lock" (например, *{"Lock": 0}*) - значение, которое определяет возможность удаления и редактирования для элемента управления содержимым. Значения могут быть следующими:

Числовое значение	**Редактирование**	**Удаление**
0	Нет	Да
1	Нет	Нет
2	Да	Нет
3	Да	Да

Кроме ключа *Props* должна быть пара вида "Ключ/Значение" с ключом *Url* или *Script*, определяющая данные, которые будут находиться в элементе управления содержимым. Если используется ключ *Url*, его значением должна быть ссылка на файл, к которому предоставлены соответствующие права доступа. Если используется ключ *Script*, его значением должен быть скрипт, который будет выполняться для того, чтобы сгенерировать данные в элементе управления содержимым.

Пример 1

```
{ "Url": "https://example.com/script.docbuilder"}
```

Пример 2

```
"Script": "var oDocument = Api.GetDocument();
var oParagraph=Api.CreateParagraph();
oParagraph.AddText('Helloworld!');
oDocument.InsertContent([oParagraph]);"
```

Возвращает

Метод возвращает данные, которые содержатся в созданном элементе управления содержимым (в формате JSON)

```
[ { "Tag": "Document", "Id": 0, "Lock": 0, "InternalId": "1_713" } ]
```

window.Asc.plugin.executeMethod ("RemoveContentControls", [args], callback)

Описание

Этот метод позволяет удалить несколько элементов управления содержимым.

Использование

Использовать метод надо так:

```
window.Asc.plugin.executeMethod("RemoveContentControls", [obj]);
```

Где *obj* - это массив из JSON следующего вида:

```
[ { "InternalId": "id" }, { "InternalId": "id2" }, ... ]
```

Возвращает

Метод возвращает значение *undefined*.

window.Asc.plugin.executeMethod ("GetAllContentControls", callback)

Описание

Этот метод позволяет получить информацию о всех элементах управления содержимым, которые добавлены на страницу.

Использование

Использовать метод надо так:

```
window.Asc.plugin.executeMethod("GetAllContentControls");
```

Возвращает

Метод возвращает данные следующего вида (JSON):

```
[ { "Tag": "Document", "Id": 0, "Lock": 0, "InternalId": "1_713" } ]
```

window.Asc.plugin.executeMethod ("AddContentControl", [args], callback)

Описание

Этот метод позволяет добавлять в документ пустой элемент управления содержимым.

Использование

Использовать метод надо так:

```
window.Asc.plugin.executeMethod("AddContentControl", [wrap, obj]);
```

Где:

- *wrap* - это числовое значение, определяющее тип элемента управления содержимым. Может иметь одно из следующих значений: **1** (block - блочный) или **2** (inline - строчный),
- *obj* - это объект JSON следующего вида:

```
{ "Id" : 0, "Lock" : 0, "Tag" : "{tag}"}
```

Объект *obj* может иметь следующие значения:

- "Id" (например, *{"Id": 2}*) - уникальный идентификатор для элемента управления содержимым. Его можно использовать для того, чтобы найти конкретный элемент управления содержимым и ссылаться на него в своем коде.
- "Tag" (например, *{"Tag": "String"}*) - тег для элемента управления содержимым. Один тег может быть присвоен нескольким элементам управления содержимым для того, чтобы можно было ссылаться на них в своем коде.
- "Lock" (например, *{"Lock": 0}*) - значение, которое определяет возможность удаления и редактирования для элемента управления содержимым. Значения могут быть следующими:

Числовое значение	Редактирование	Удаление
0	Нет	Да
1	Нет	Нет
2	Да	Нет
3	Да	Да

Возвращает

Метод возвращает объект JSON с данными о созданном элементе управления содержимым следующего вида (JSON):

```
{ "Tag": "{tag}", "Id": 0, "Lock": 0, "InternalId": "1_713"}
```

window.Asc.plugin.executeMethod ("RemoveContentControl", callback)

Описание

Этот метод позволяет удалить выделенный элемент управления содержимым, но при этом оставить данные, которые в нем были.

Использование

Использовать метод надо так:

```
window.Asc.plugin.executeMethod("RemoveContentControl");
```

Элемент управления содержимым, в котором находится курсор, будет удален.

Возвращает

Метод возвращает значение *undefined*.

window.Asc.plugin.executeMethod ("GetCurrentContentControl", callback)

Описание

Этот метод позволяет получить идентификатор выделенного элемента управления содержимым (то есть того, в котором находится курсор).

Использование

Использовать метод надо так:

```
window.Asc.plugin.executeMethod("GetCurrentContentControl");
```

Возвращает

Метод возвращает ID элемента.

window.Asc.plugin.info

Описание

Во время работы плагина доступен вспомогательный объект *window.Asc.plugin.info*. В нем хранится вся информацию о редакторе, в котором используется плагин: редактор документов, таблиц или презентаций (для этого используется метод *editorType*) и дополнительные параметры для OLE-объектов (их ширина, высота, коэффициент перевода миллиметров в пиксели для векторной отрисовки OLE-объекта, а также некоторые другие параметры).

Этот объект используется для изменения данных объекта и передачи дополнительных параметров при выполнении метода *window.Asc.plugin.executeCommand*. Например, если меняется содержимое документа и требуется пересчет, параметр *window.Asc.plugin.info.recalculate* должен иметь значение *true*. Это необходимо, так как пересчет асинхронен, к тому же может потребоваться подгрузить данные (например, шрифт).

Ниже приводятся подробные сведения о доступных методах и свойствах объекта *window.Asc.plugin.info*.

Методы и свойства

Имя	Описание	Тип	Доступ
data	Данные OLE-объекта, которые требуется передать объекту window.Asc.plugin.info и которые используются плагином.	данные OLE-объекта	запись
editorType	Возвращает тип редактора, в котором в данный момент запущен плагин.	строка	только чтение
guid	GUID OLE-объекта в текущем документе.	строка	только чтение
height	Высота OLE-объекта в миллиметрах.	число	только чтение
imgSrc	Ссылка на изображение (визуальное представление), хранящееся в OLE-объекте и используемое плагином.	строка	запись
mmToPx	Коэффициент перевода миллиметров в пиксели для векторной отрисовки OLE-объекта.	число	только чтение
objectId	Идентификатор OLE-объекта в текущем документе.	строка	только чтение
recalculate	Принудительный пересчет данных документа.	логическое значение	запись
resize	Проверяет, был ли изменен размер OLE-объекта.	логическое значение	только чтение

width	Ширина OLE-объекта в миллиметрах.	число	только чтение

Пример

```
window.Asc.plugin.button = function (id) {
        var _info = window.Asc.plugin.info;
        var _method = (_info.objectId === undefined) ? "asc_addOleObject" : "asc_editOleObject";
        _info.width = _info.width ? _info.width : 70;
        _info.height = _info.height ? _info.height : 70;
        _info.widthPix = (_info.mmToPx * _info.width) >> 0;
        _info.heightPix = (_info.mmToPx * _info.height) >> 0;
        _info.imgSrc = window.g_board.getResult(_info.widthPix, _info.heightPix).image;
        _info.data = window.g_board.getData();
        var _code = "Api." + _method + "(" + JSON.stringify(_info) + ");";
        this.executeCommand("close", _code);
};
```

window.Asc.plugin.info.data

запись

Описание

Данные OLE-объекта, которые требуется передать объекту *window.Asc.plugin.info* и которые используются плагином.

Возвращает

Тип возвращаемого значения - строка

Пример

```
window.Asc.plugin.button = function (id) {
        var _info = window.Asc.plugin.info;
        var _method = (_info.objectId === undefined) ? "asc_addOleObject" : "asc_editOleObject";
        _info.width = _info.width ? _info.width : 70;
        _info.height = _info.height ? _info.height : 70;
        _info.widthPix = (_info.mmToPx * _info.width) >> 0;
        _info.heightPix = (_info.mmToPx * _info.height) >> 0;
        _info.imgSrc = window.g_board.getResult(_info.widthPix, _info.heightPix).image;
        _info.data = window.g_board.getData();
        var _code = "Api." + _method + "(" + JSON.stringify(_info) + ");";
        this.executeCommand("close", _code);
};
```

window.Asc.plugin.info.editorType

только чтение

Описание

Возвращает тип редактора, в котором в данный момент запущен плагин.

Значения могут быть следующими:

- **"word"** - редактор текстовых документов;
- **"cell"** - редактор электронных таблиц;
- **"slide"** - редактор презентаций.

Возвращает

Тип возвращаемого значения - строка

Пример

```
function createScriptFromArray (aSelected) {
	var sScript = '';
	if (aSelected.length > 0) {
		switch (window.Asc.plugin.info.editorType) {
			case 'word': {
				sScript += 'var oDocument = Api.GetDocument();';
				sScript += '\noDocument.CreateNewHistoryPoint();';
				sScript += '\nvar oParagraph, oRun, arrInsertResult = [], oImage;';
				sScript += '\noDocument.InsertContent(arrInsertResult);';
				break;
			}
		}
	}
	return sScript;
}
```

window.Asc.plugin.info.guid

только чтение

Описание

GUID OLE-объекта в текущем документе.

Возвращает

Тип возвращаемого значения - строка

Пример

```
window.Asc.plugin.init = function () {
        var plugin_uuid = window.Asc.plugin.info.guid;
};
```

window.Asc.plugin.info.height

только чтение

Описание

Высота OLE-объекта в миллиметрах.

> Если вам нужна высота OLE-объекта в пикселях для растрового представления, используйте метод *window.Asc.plugin.info.mmToPx* для преобразования значений.

Возвращает

Тип возвращаемого значения - число

Пример

```
window.Asc.plugin.button = function (id) {
        var _info = window.Asc.plugin.info;
        var _method = (_info.objectId === undefined) ? "asc_addOleObject" : "asc_editOleObject";
        _info.width = _info.width ? _info.width : 70;
        _info.height = _info.height ? _info.height : 70;
        _info.widthPix = (_info.mmToPx * _info.width) >> 0;
        _info.heightPix = (_info.mmToPx * _info.height) >> 0;
        _info.imgSrc = window.g_board.getResult(_info.widthPix, _info.heightPix).image;
        _info.data = window.g_board.getData();
        var _code = "Api." + _method + "(" + JSON.stringify(_info) + ");";
        this.executeCommand("close", _code);
};
```

window.Asc.plugin.info.imgSrc

запись

Описание

Ссылка на изображение (визуальное представление), хранящееся в OLE-объекте и используемое плагином.

Возвращает

Тип возвращаемого значения - строка

Пример

```
window.Asc.plugin.button = function (id) {
        var _info = window.Asc.plugin.info;
        var _method = (_info.objectId === undefined) ? "asc_addOleObject" : "asc_editOleObject";
        _info.width = _info.width ? _info.width : 70;
        _info.height = _info.height ? _info.height : 70;
        _info.widthPix = (_info.mmToPx * _info.width) >> 0;
        _info.heightPix = (_info.mmToPx * _info.height) >> 0;
        _info.imgSrc = window.g_board.getResult(_info.widthPix, _info.heightPix).image;
        _info.data = window.g_board.getData();
        var _code = "Api." + _method + "(" + JSON.stringify(_info) + ");";
        this.executeCommand("close", _code);
};
```

window.Asc.plugin.info.mmToPx

только чтение

Описание

Коэффициент перевода миллиметров в пиксели для векторной отрисовки OLE-объекта.

Значения высоты и ширины OLE-объекта возвращаются в миллиметрах, поэтому их потребуется перевести в пиксели для растрового представления.

Возвращает

Тип возвращаемого значения - число

Пример

```
window.Asc.plugin.button = function (id) {
        var _info = window.Asc.plugin.info;
        var _method = (_info.objectId === undefined) ? "asc_addOleObject" : "asc_editOleObject";
        _info.width = _info.width ? _info.width : 70;
        _info.height = _info.height ? _info.height : 70;
        _info.widthPix = (_info.mmToPx * _info.width) >> 0;
        _info.heightPix = (_info.mmToPx * _info.height) >> 0;
        _info.imgSrc = window.g_board.getResult(_info.widthPix, _info.heightPix).image;
```

```
        _info.data = window.g_board.getData();
        var _code = "Api." + _method + "(" + JSON.stringify(_info) + ");";
        this.executeCommand("close", _code);
};
```

window.Asc.plugin.info.objectId

только чтение

Описание

Идентификатор OLE-объекта в текущем документе.

Возвращает

Тип возвращаемого значения - строка

Пример

```
window.Asc.plugin.button = function (id) {
        var _info = window.Asc.plugin.info;
        var _method = (_info.objectId === undefined) ? "asc_addOleObject" : "asc_editOleObject";
        _info.width = _info.width ? _info.width : 70;
        _info.height = _info.height ? _info.height : 70;
        _info.widthPix = (_info.mmToPx * _info.width) >> 0;
        _info.heightPix = (_info.mmToPx * _info.height) >> 0;
        _info.imgSrc = window.g_board.getResult(_info.widthPix,
        _info.heightPix).image; _info.data = window.g_board.getData();
        var _code = "Api." + _method + "(" + JSON.stringify(_info) + ");";
        this.executeCommand("close", _code);
};
```

window.Asc.plugin.info.recalculate

запись

Описание

Позволяет принудительно пересчитать данные о содержимом документа.

Если меняется содержимое документа и требуется пересчет, параметр *window.Asc.plugin.info.recalculate* должен иметь значение *true*. Это необходимо, так как пересчет асинхронен, к тому же может потребоваться подгрузить данные (например, шрифт).

При использовании плагинов с OLE-объектами метод *isRecalculate* не требуется, так как документ пересчитывает данные сам.

Возвращает

Тип возвращаемого значения - логическое значение

Пример

```
window.Asc.plugin.init = function () {
        var sScript = 'var oDocument = Api.GetDocument();';
        sScript += 'oDocument.CreateNewHistoryPoint();';
        sScript += 'oParagraph = Api.CreateParagraph();';
        sScript += 'oParagraph.AddText(\'Hello word!\');';
        sScript += 'oDocument.InsertContent([oParagraph]);';
        window.Asc.plugin.info.recalculate = true;
        this.executeCommand("close", sScript);
};
```

window.Asc.plugin.info.resize

только чтение

Описание

Проверяет, был ли изменен размер OLE-объекта.

Если *window.Asc.plugin.info.resize === true,* объект будет перерисован.

Возвращает

Тип возвращаемого значения - логическое значение

Пример

```
if (window.Asc.plugin.info.resize === true) {
        return window.Asc.plugin.button(0);
}
```

window.Asc.plugin.info.width

только чтение

Описание

Ширина OLE-объекта в миллиметрах.

> Если вам нужна ширина OLE-объекта в пикселях для растрового представления, используйте метод *window.Asc.plugin.info.mmToPx* для преобразования значений.

Возвращает

Тип возвращаемого значения - число

Пример

```
window.Asc.plugin.button = function (id) {
        var _info = window.Asc.plugin.info;
        var _method = (_info.objectId === undefined) ? "asc_addOleObject" : "asc_editOleObject";
        _info.width = _info.width ? _info.width : 70;
        _info.height = _info.height ? _info.height : 70;
        _info.widthPix = (_info.mmToPx * _info.width) >> 0;
        _info.heightPix = (_info.mmToPx * _info.height) >> 0;
        _info.imgSrc = window.g_board.getResult(_info.widthPix, _info.heightPix).image;
        _info.data = window.g_board.getData();
        var _code = "Api." + _method + "(" + JSON.stringify(_info) + ");";
        this.executeCommand("close", _code);
};
```

window.Asc.plugin.init(data)

Описание

Данные, передаваемые плагину, и информация о том, что и как с ними надо сделать. Этот метод вызывается при старте плагина.

Параметры

Имя	Описание	Тип
data	Параметр *data* зависит от параметра *"initDataType"*, прописанного в файле конфигурации плагина *config.json*. Его значения могут быть следующими: *"none"* - пустая строка, *"text"* - выделенный в документе текст, *"html"* - выделенный фрагмент документа, *"ole"* - данные OLE-объекта.	строка

Пример

```
window.Asc.plugin.init = function () {
	this.callCommand(function() {
		var oDocument = Api.GetDocument();
		var oParagraph = Api.CreateParagraph();
		oParagraph.AddText("Hello world!");
		oDocument.InsertContent([oParagraph]);
	}, true);
};
```

window.Asc.plugin.init.onExternalMouseUp(fMouseUp)

Описание

Определяет действие, которое должно выполняться, когда кнопка мыши отпущена вне iframe плагина.

Параметры

Имя	Описание	Тип
fMouseUp	Функция, которая должна выполняться, когда кнопка мыши отпущена вне iframe плагина.	функция

Пример

```
window.Asc.plugin.onExternalMouseUp = function () {
	var evt = document.createEvent("MouseEvents");
	evt.initMouseEvent("mouseup", true, true, window, 1, 0, 0, 0, 0, false, false, false, false, 0, null);
	document.dispatchEvent(evt);
};
```

window.Asc.plugin.onMethodReturn

Описание

Метод объекта *window.Asc.plugin,* который позволяет возвращать результат ранее выполненного метода. Может использоваться для возвращения данных после выполнения метода *window.Asc.plugin.executeMethod(...)*.

Использование

Использовать метод надо так:

```
window.Asc.plugin.onMethodReturn = function(returnValue) {...}
```

Пример

```
window.Asc.plugin.executeMethod("InsertAndReplaceContentControls", [_obj]);
window.Asc.plugin.onMethodReturn = function(returnValue) {
        if (window.Asc.plugin.info.methodName == "InsertAndReplaceContentControls") {
                window.Asc.plugin.executeMethod("GetAllContentControls");
        } else if ("GetAllContentControls") {
                window.Asc.plugin.executeCommand("close",
console.log(JSON.stringify(returnValue)));
         }
};
```

Объект Asc.scope

Описание

Объект, используемый для передачи дополнительных данных (объекты, параметры, переменные и т.д.) методу *window.Asc.plugin.callCommand*, который исполняется в собственном изолированном контексте.

> Функции нельзя передавать методу *window.Asc.plugin.callCommand* с помощью объекта *Asc.scope*.

Пример

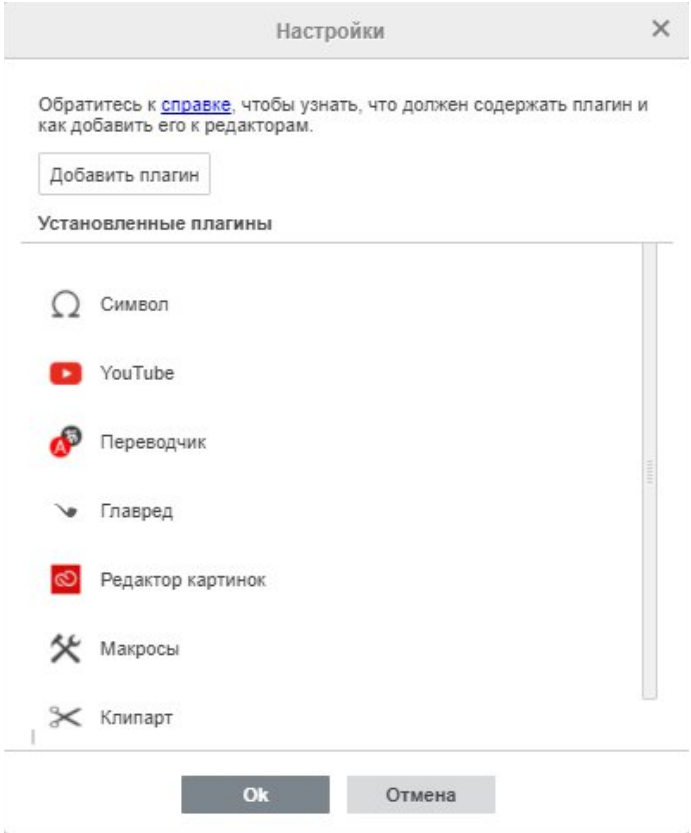
```
(function(window, undefined){
        var scopeText = ["Hello World!", "This is me!", "I'm glad to see you!!!"];
        window.Asc.plugin.init = function() {
                Asc.scope.st = scopeText;
                this.callCommand(function() {
                        var oDocument = Api.GetDocument();
                        var oParagraph = Api.CreateParagraph();
                        for (var i = 0; i < Asc.scope.st.length; i++)
```

```
                        {
                                oParagraph.AddText(Asc.scope.st[i] + "<br />");
                        }
                        oDocument.InsertContent([oParagraph]);
                }, true);
        };
        window.Asc.plugin.button = function(id)
        {
         };
})(window, undefined);
```

Установка плагинов в редакторах документов

Добавление плагина в десктопные редакторы

Необходимо взять папку плагина (она **должна** содержать файлы *config.json*, *index.html* и *pluginCode.js*), заархивировать ее содержимое и изменить расширение полученного файла на *.plugin*. После этого откройте вкладку **Плагины** и нажмите на кнопку **Настройки**. Откроется окно диспетчера плагинов:



Для загрузки созданного вами плагина используйте кнопку **Добавить плагин**. Он сразу же будет добавлен в открытые редакторы и во все редакторы, открытые позже. Для удаления выбранных плагинов используется кнопка **Удалить плагин** .

Удаление плагинов из редакторов документов

Для удаления установленных плагинов необходимо выполнить следующие действия:

- Для десктопных редакторов - для удаления выбранного плагина используйте кнопку **Удалить плагин** в окне **Настройки**.

Пример плагина

Чтобы понять, как работают плагины, как их можно написать и как добавить в редакторы документов, рассмотрите следующий пример плагина *helloworld.js*:

```
(function(window, undefined){
	var text = "Hello world!";
	window.Asc.plugin.init = function() {
		Asc.scope.text = text; // export variable to plugin scope
		this.callCommand(function() {
			var oDocument = Api.GetDocument();
			var oParagraph = Api.CreateParagraph();
			oParagraph.AddText(Asc.scope.text); // or oParagraph.AddText(scope.text);
			oDocument.InsertContent([oParagraph]);
		}, true);
	};
	window.Asc.plugin.button = function(id)
	{
	};
})(window, undefined);
```

Это самый простой плагин, который при нажатии на кнопку плагина вставляет в документ фразу 'Hello world!'.