

Плагины

Инструкция пользователя

Плагины Metabot — это JavaScript библиотеки, скрипты которых доступны для повторного использования в различных скриптах ботов.

Существует два вида плагинов:

- **Плагины бизнеса** — это собственные JavaScript библиотеки, созданные вами и доступные в любом боте вашего бизнеса. Другим бизнесам эти библиотеки не доступны;
- **Общие (предопределенные) плагины** — это JavaScript библиотеки доступные в любом бизнесе, любого бота. Разработкой таких библиотек занимается команда Metabot.

Другими словами, вы можете создать свои собственные JavaScript библиотеки или использовать предопределенные, готовые библиотеки.

Список общих плагинов

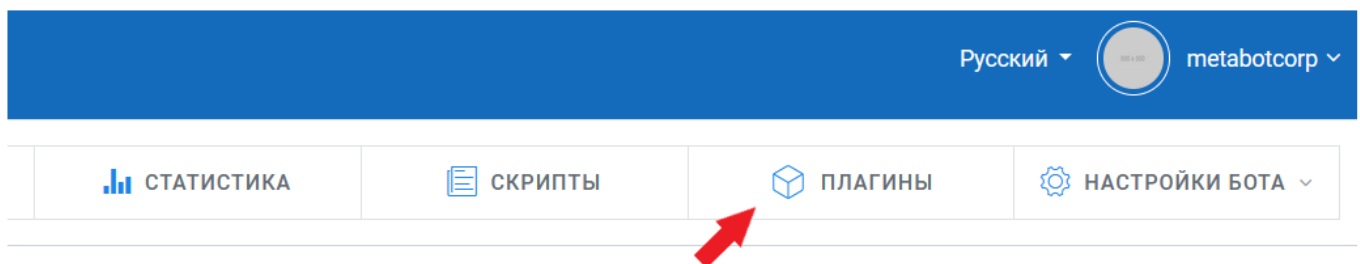
Наименование	Подключение	Пример	Дополнительно
moment	require()	let moment = require('moment')	
moment-with-locales	require()	let moment = require('moment-with- locales')	Это OpenSource библиотека для работы с датами, https://momentjs.com/ Версия библиотеки moment.js: 2.29.3
Common.Bot.Commands	require() snippet()	require('Common.Bot.Com mands') snippet('Common.Bot.Com mands')	

Если для вашей ситуации доступно подключение через **require** – то это **более предпочтительный** с точки зрения производительности вариант, используйте именно

его. Если подключение с помощью `require()` недоступно или не приемлемо, то используйте подключение через `snippet()`.

На данный момент подключение **собственных библиотек** бизнеса доступно **только** через `snippet()`.

Интерфейс и логика настройки плагинов



Интерфейс плагинов на платформе Metabot24 состоит из двух уровней:

- **Плагины** — в данном разделе настраивается общее описание библиотеки и пространство имен для скриптов вложенных в плагин. Название плагина похоже на название директории в которой хранится ваша библиотека;

ПЛАГИНЫ БИЗНЕСА

[Главная](#) / [Плагины бизнеса](#)

ПЛАГИНЫ БИЗНЕСА

+ Создать

ID	АКТИВЕН	НАИМЕНОВАНИЕ	ЗАГОЛОВОК В ИНТЕРФЕЙСЕ	СОЗДАН	ОБНОВЛЕН	ОПЕРАЦИИ
2	Да	Test	Тестовый плагин	2022-06-06 16:51:20	2022-06-06 17:00:25	<div><div><div><div></div></div><div><div></div></div><div><div></div></div></div><div>Скрипты</div></div>

Комментарий:

Описание плагина

+ Создать

- **Скрипты плагинов** — это сами скрипты которые вы можете использовать в своих ботах. Название соответствует названию файла к которому вы обращаетесь.

СКРИПТЫ ПЛАГИНА "TEST"

+ Создать

ID	АКТИВЕН	НАИМЕНОВАНИЕ	ЗАГОЛОВОК В ИНТЕРФЕЙСЕ	СОЗДАН	ОБНОВЛЕН	ОПЕРАЦИИ
3	Да	HelloWorld	Тестовый скрипт	2022-06-06 17:34:02	2022-06-06 17:34:02	<div><div></div><div></div><div></div></div>

Исходный код:

```
let text = "Hello World!";
```

+ Создать

Полное имя библиотеки «клеится» из трех составляющих:

- Уровень доступа (Common или Business);
- Название плагина;
- Название скрипта плагина.

При подключении библиотеки «составляющие» разделяются точкой, а при обращении к методам библиотеки из JS «составляющие клеятся» вместе, т.е. пишутся слитно, без точки.

Сниппеты

Перед тем как разобрать пример использования собственного плагина, необходимо понять принцип действий сниппетов. Работа сниппетов аналогична макросам, т.е. это просто текст, который подставляется вместо указанного макроса.

Доступны два варианта подключения сниппета:

```
snippet("your_snippet_name")
или
[[: your_snippet_name: ]]
```

Оба варианта работают как макрос, т.е.указанный вариант объявления сниппета в JavaScript коде будет заменен на код, который указан в скрипте плагина, к которому мы обращаемся.

В содержимом скрипта плагина можно указывать любой JavaScript код, главное чтобы ваш код, в котором выполняется макроподстановка сниппета был валиден после замены

Рассмотрим сначала простой пример указания значения переменной, вместо JavaScript. Вместо JavaScript кода сниппета можно указать любую комбинацию, например:

```
let text = snippet("mysnippet");
```

А в самом сниппете указать любое значение (число, текст, булево и т.п.), например, если в сниппете мы указали код:

```
"Hello World! "
```

То, запускаемый JS интерпретатором код примет следующий вид:

```
let text = "Hello World! "
```

Обратите внимание, что символ точка с запятой является частью определения макроса, и может указываться после `snippet`, а может не указываться. Если вам необходимо чтобы «конечный исходный код», после замены содержал точку с запятой, то укажите ее в самом JavaScript коде сниппета (скрипта плагина), т.е.:

```
"Hello World! ";
```

Тогда, запускаемый JS интерпретатором код примет следующий вид:

```
let text = "Hello World! ";
```

Сниппет идентичен макроподстановке. Макроподстановка выполняется до запуска скрипта и не является командой интерпретатора JavaScript, поэтому если вы прокомментируете объявление сниппета, то он все равно будет подставлен в исходный код ! Чтобы прокомментировать сниппет нужно прокомментировать его и нарушить синтаксис его объявления, чтобы система не нашла макрос со сниппетом и не выполнила макроподстановку, например написать: `s!snippet("your_snippet_name")`.

Пример №1: Общий плагин (скрипт общего плагина)

Полное имя библиотеки «клеится» из трех составляющих:

- Общий плагин (Common);
- Название плагина «Bot»;
- Скрипт «Commands».

Подключается такая библиотеки с помощью кода:

```
require('Common.Bot.Commands');
```

После подключения, в JavaScript автоматически будет доступна переменная **CommonBotCommands**.

К методам CommonBotCommands можно обращаться, например для отправки сообщения в мессенджер из JavaScript:

```
CommonBotCommands.SendText('Текст отправленный из V8');
```

Полный список команд каждой библиотеки выходит за рамки данной документации и может постоянно дополняться.

В данном примере речь идет именно об общей библиотеке (Common), внутри которой уже описан специальный объект, который можно использовать с помощью методов, чтобы добиться такого же варианта использования вы должны самостоятельно описать свой объект на JavaScript. Оформление в виде JavaScript объекта при создании вашего собственного плагина бизнеса не является обязательным, за счет того что ваши скрипты будут подключаться в виде сниппетов. Вы можете просто описать любые переменные и функции в вашей библиотеке и обращаться к ним по имени.

Для библиотек уровня бизнеса автоматически создаваемые переменные-объекты не доступны (или доступны только в тех библиотеках которые для вас создаст команда Metabot).

Отметим, что существуют другие библиотеки, например, такие как moment.js. Подключение и использование таких библиотек отличается от использования библиотек созданных на платформе. Например, для moment.js, не нужно указывать три «составляющие» названия скрипта (уровень доступа, плагин и скрипт). Для moment.js мы явно объявляем переменную и «экспортируем в нее библиотеку»:

```
let moment = require('moment');
```

Подключение различных библиотек может отличаться и зависит от реализации самой библиотеки. Но, в основном как подключать и будет ли доступна автоматически переменная-объект понятно из названия библиотеки (если при подключении нет указания уровня доступа, то скорее всего это OpenSource библиотека).

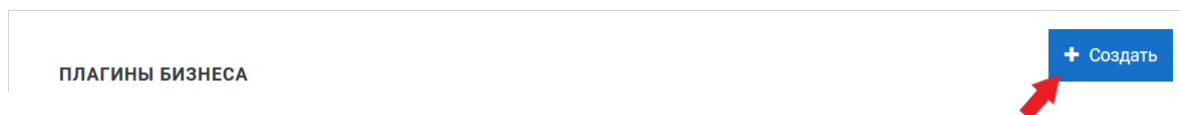
Если вы подключаете к одному скрипту несколько сниппетов, то переменные объявленные внутри каждого из сниппетов не должны пересекаться между собой по имени, а также переменные объявленные в сниппете и вашем скрипте, к которому вы подключаете сниппет не должны пересекаться.

Пример №2: Плагин бизнеса (скрипт вашего плагина)

Создадим новый плагин для формирования текста, который мы будем отправлять пользователю с помощью атрибутов бота.

ПЛАГИНЫ БИЗНЕСА

[Главная](#) / [Плагины бизнеса](#)



Заполним все необходимые поля.

ПЛАГИНЫ БИЗНЕСА

[Главная](#) / [Плагины бизнеса](#) / [Создание плагина](#)

Создание плагина

☒ Активен

Наименование: ⓘ

Notifications

Название должно начинаться с заглавной латинской буквы. Последующие символы - латинские буквы в любом регистре или цифры.

Заголовок в интерфейсе:

Уведомления

Комментарий:

Тестовый плагин для уведомлений в боте

Создать

Далее, перейдем в скрипты плагина.

ПЛАГИНЫ БИЗНЕСА

[Главная](#) / [Плагины бизнеса](#)

ПЛАГИНЫ БИЗНЕСА

+ Создать

ID	АКТИВЕН	НАИМЕНОВАНИЕ	ЗАГОЛОВОК В ИНТЕРФЕЙСЕ	СОЗДАН	ОБНОВЛЕН	ОПЕРАЦИИ
2	Да	Notifications	Уведомления	2022-06-06 16:51:20	2022-06-06 17:50:24	<div><div><div><div></div></div><div><div></div></div><div><div></div></div></div><div>↓ Скрипты</div></div>

Комментарий:

Тестовый плагин для уведомлений в боте

+ Создать

И создадим новый скрипт.

ПЛАГИНЫ БИЗНЕСА

[Главная](#) / [Плагины бизнеса](#) / [Скрипты плагина "Notifications"](#)

СКРИПТЫ ПЛАГИНА "NOTIFICATIONS"

+ Создать

Заполним в открывшемся окне необходимые поля.

ПЛАГИНЫ БИЗНЕСА

[Главная](#) / [Плагины бизнеса](#) / [Скрипты плагина "Notifications"](#) / Создание скрипта

Создание скрипта

☒ Активен

Наименование:

Заголовок в интерфейсе:

Исходный код:

```
1 const greetMsg = 'Привет, ' + lead.getData('name') + '!';
2 memory.setAttr("greet", greetMsg);
```

Комментарий:

Укажем в исходном коде скрипта:

```
const greetMsg = 'Привет, ' + lead.getData('name') + '!';
memory.setAttr("greet", greetMsg);
```

Здесь мы сохраняем текст приветствия в memory (временном атрибуте бота).

Полное имя библиотеки «клеится» из трех составляющих:

- Плагин бизнеса (Business);
- Название плагина «Notifications»;
- Скрипт «HelloLead».

Подключается с помощью кода:

```
snippet('Business.Notifications.HelloLead');
```


СКРИПТЫ ПЛАГИНА "NOTIFICATIONS"

[+ Создать](#)

ID	АКТИВЕН	НАИМЕНОВАНИЕ	ЗАГОЛОВОК В ИНТЕРФЕЙСЕ	СОЗДАН	ОБНОВЛЕН	ОПЕРАЦИИ
3	Да	HelloLead	Вывод текста	2022-06-06 17:34:02	2022-06-06 18:38:49	✎ 🗑 ↑

Исходный код:

```
const greetgMsg = 'Привет, ' + lead.getName() + '!';
memory.setAttr("greet", greetMsg);
```

Для проверки работы плагина, в скрипт бота необходимо добавить две команды:

- **Выполнить JavaScript** — здесь мы подключаем сниппет, для заполнения атрибута(макропеременной) во временной памяти бота:

```
snippet('Business.Notifications.HelloLead');
```

- **Отправить текст** — в содержимом использована макропеременная, текст который будет отправлен в мессенджер:

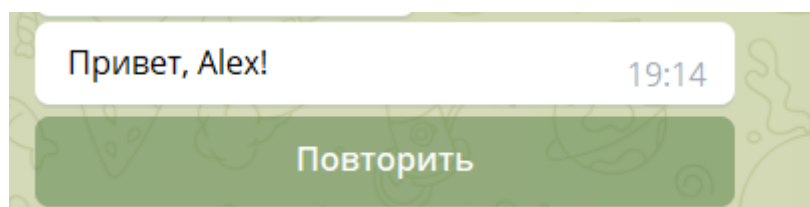
```
{{ &$greet }}
```

КОМАНДЫ

Эти команды бот выполняет до того как покажет меню.

КОМАНДА	СОДЕРЖИМОЕ	ОПЕРАЦИИ
Выполнить JavaScript	<code>snippet('Business.Notifications.HelloLead');</code>	✎ 🗑 ↑
Отправить текст	<code>{{ &\$greet }}</code>	✎ 🗑 ↑

Результат работы скрипта в телеграм:



Пример №3: Использование общего плагина в плагине бизнеса

Модифицируем пример 2 так, чтобы текст отправлялся не с помощью команды **Отправить текст**, а прям из JS кода плагина.

ПЛАГИНЫ БИЗНЕСА

[Главная](#) / [Плагины бизнеса](#) / Скрипты плагина "Notifications"

СКРИПТЫ ПЛАГИНА "NOTIFICATIONS"						+ Создать
ID	АКТИВЕН	НАИМЕНОВАНИЕ	ЗАГОЛОВОК В ИНТЕРФЕЙСЕ	СОЗДАН	ОБНОВЛЕН	ОПЕРАЦИИ
3	Да	HelloLead	Вывод текста	2022-06-06 17:34:02	2022-06-06 19:45:51	✎ 🗑 ↑

В исходном коде скрипта укажем:

```
require('Common.Bot.Commands');
const greetMsg = 'Привет, ' + lead.getData('name') + '!';
CommonBotCommands.sendText(greetMsg);
```

ПЛАГИНЫ БИЗНЕСА

[Главная](#) / [Плагины бизнеса](#) / [Скрипты плагина "Notifications"](#) / Редактирование скрипта "HelloLead"

Редактирование скрипта "HelloLead"

☒ Активен

Наименование:

Заголовок в интерфейсе:

Исходный код:

```
1 require('Common.Bot.Commands');
2 const greetMsg = 'Привет, ' + lead.getData('name') + '!';
3 CommonBotCommands.sendText(greetMsg);
```

Комментарий:

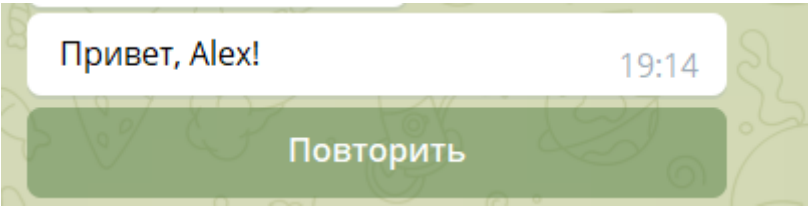
Удалим команду **Отправить текст**, команду **Выполнить JavaScript** оставляем без изменений.

КОМАНДЫ

Эти команды бот выполняет до того как покажет меню.

КОМАНДА	СОДЕРЖИМОЕ	ОПЕРАЦИИ
Выполнить JavaScript	<pre>snippet('Business.Notifications.HelloLead');</pre>	<div><div></div><div></div><div></div></div>

Результат работы данного скрипта будет идентичным примеру 2:



Версия #2
Юрий Гарашко создал 12 June 2023 10:59:05
Ирина Петрова обновил 5 February 2024 12:16:46