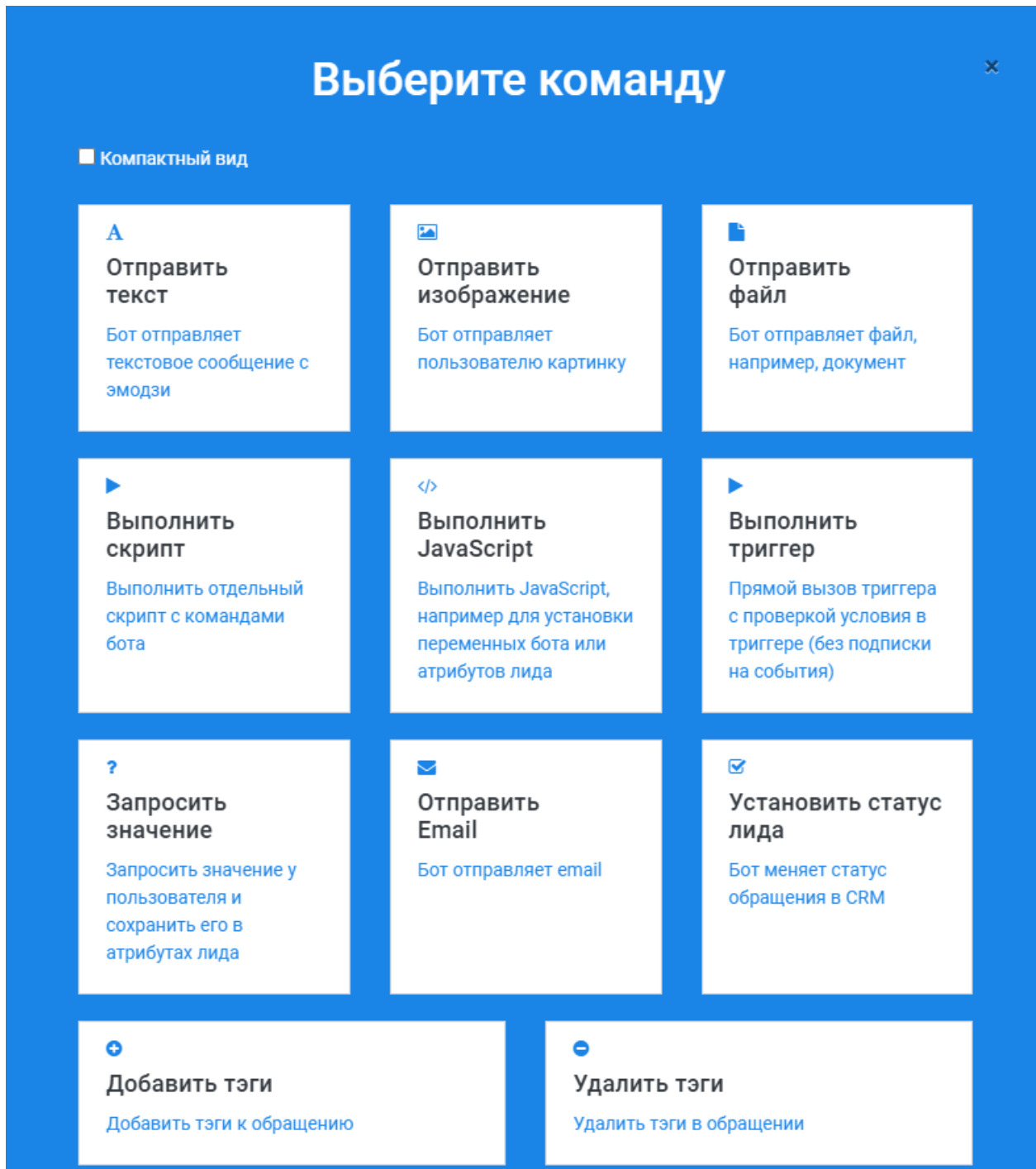


Работа с командами скриптов

- [Список команд](#)
- [Отправить текст](#)
- [Отправить изображение и Отправить файл](#)
- [Отправить Email](#)
- [Установить статус лида](#)
- [Выполнить скрипт](#)
- [Запросить значение](#)
- [Добавить тэги и удалить тэги](#)
- [Добавить контекст и удалить контекст](#)
- [Повторить вопрос](#)
- [Выполнить триггер](#)
- [Выполнить JavaScript](#)
- [Стоп](#)
- [Распознать NLP намерение](#)
- [Перевести на оператора и Вернуть боту](#)
- [Выполнить JavaScript Callback](#)
- [Доступные макропеременные для команды Отправить текст и Email](#)
- [Регулярные выражения](#)
- [Условие выполнения команды JS](#)

Список команд

Ввод любой команды в скрипт начинается с выбора команды из списка **Команд:**





Распознать NLP намерение

По распознанному намерению будет найден и запущен соответствующий скрипт. Если интент не будет распознан или скрипт по интенту не будет найден, то будет выполнена следующая команда скрипта.



Выполнить JavaScript Callback

Запустить JavaScript Callback с замыканием состояния на текущей команде



Выполнить асинхронный API-запрос

Выполнить асинхронный API-запрос с замыканием состояния на текущей команде и ожиданием результата выполнения запроса



Добавить контекст

Добавить контекст обращения



Удалить контекст

Удалить контекст обращения



Перевести на оператора

Перевести диалог на оператора CRM



Вернуть боту

Вернуть диалог от оператора обратно на бота



Повторить вопрос

Повторить вопрос



Стоп

Остановить выполнение, не выводить меню

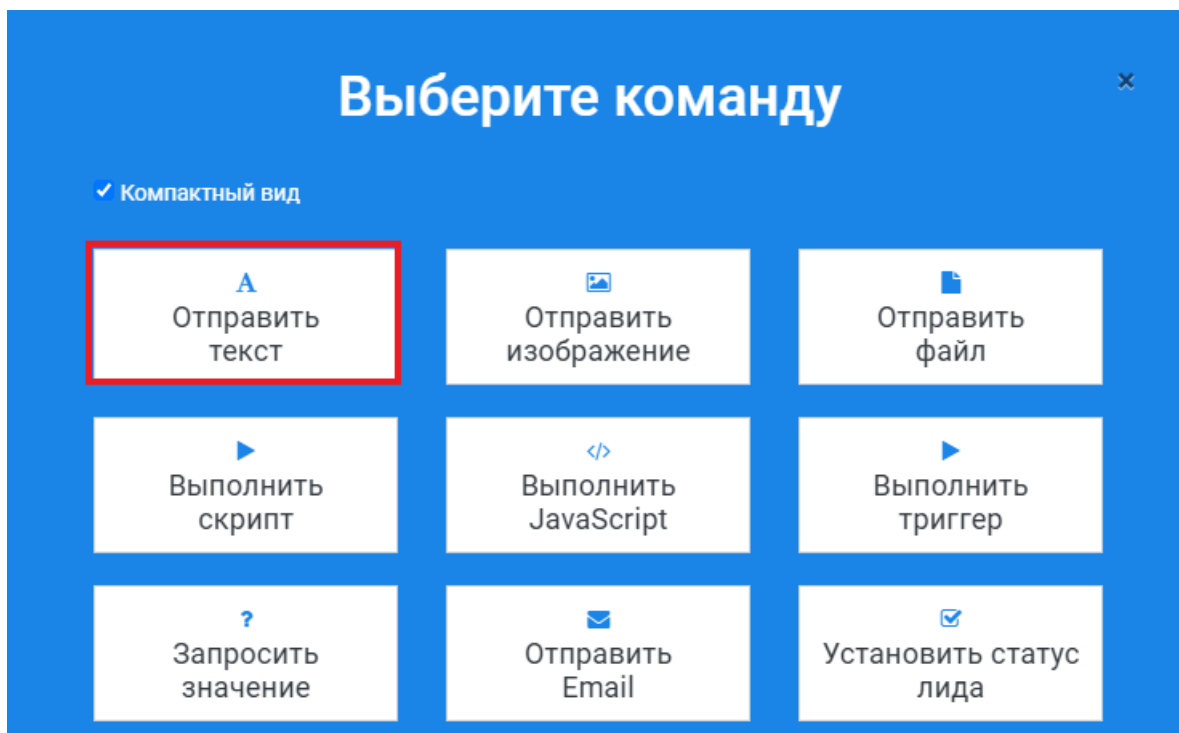
Для более компактного вида можно активировать пункт **Компактный вид**.

Отправить текст

Результатом работы данной команды является текстовое сообщение отправленное пользователю бота.

Функциональную команду отправки текстового сообщения можно использовать в любой части вашего бота. Она наиболее часто используется при разработке скриптов. На основании этой команды строится большинство разговорных диалогов.

Добавить в скрипт ее можно, выбрав пункт **Отправить текст** из списка **Команд**:



После выбора данной команды в списке **Команд** перед вами откроется диалоговое окно настройки свойств:

Добавить команду

Настройте свойства команды

☐ Использовать условие

Тип команды:

Отправить текст

Browse

Текст:

Команда Метабот24 рада приветствовать вас на нашей платформе! 🤖 😊
👤 Чтобы вам было легче освоиться здесь, рекомендуем ознакомиться с нашей документацией. Это можно сделать по ссылке: <https://docs.metabot24.ru/books/metabot-platform>

Создать

С пунктом **Использовать условие** вы можете познакомиться на странице [Условие выполнения команды JS](#).

В поле **Текст** можно ввести текстовое сообщение, эмодзи для эмоциональной разгрузки текста и ссылки на сторонние ресурсы, например, Youtube, Telegram. Помимо этого вы можете использовать макропеременные в вашем сообщении.

Так же текст можно форматировать в некоторых мессенджерах, что так же позволит сделать его более приятным глазу и читабельным. Вот несколько примеров форматирования:

Мессенджер	Форматирование	Спецсимволы
Telegram	Жирный текст	**Жирный текст**
	Текст курсивом	<i>__Текст курсивом__</i>
	Зачеркнутый текст	~~Зачеркнутый текст~~
	Моноширинный текст	<code>`Моноширинный текст`</code>

```Моноширинный текст 2```	
Ссылка в тексте	[Текст](Ссылка)
Скрытый текст или spoiler	Скрытый текст

--	--	--

WhatsApp	Жирный текст	<b>*Жирный текст*</b>
	Текст курсивом	<i>_Текст курсивом_</i>
	Зачеркнутый текст	~Зачеркнутый текст~
	Моноширинный текст	```Моноширинный текст```
	Цитирование	> Цитирование
	Встроенный код	`Код`

Макропеременные доступные для команды **Отправить текст** можно посмотреть на странице [Доступные макропеременные для команды Отправить текст и Email](#)

# Отправить изображение и Отправить файл

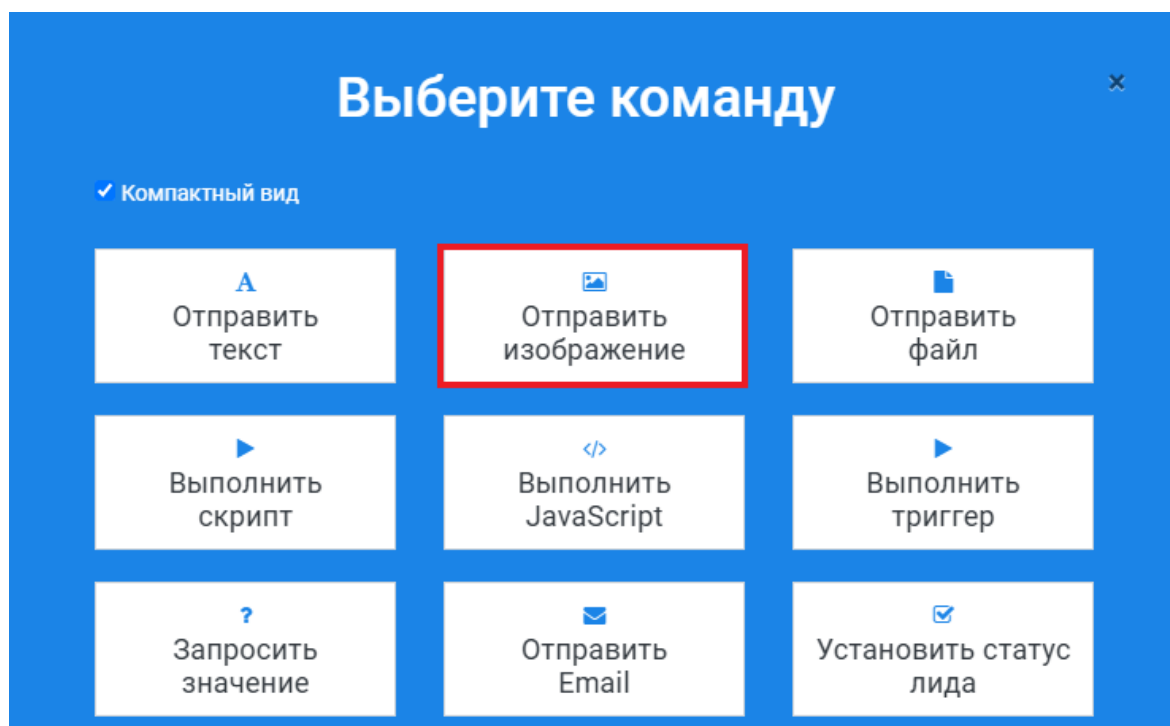
Результатом работы данных команд является сообщение с изображением или файлом отправленное пользователю бота.

Функциональную команду **Отправки изображения** можно использовать для передачи информации пользователю в графическом виде, что часто бывает проще сделать, чем написать текст.

На основании команды **Отправки файла** можно настроить отправку текстового файла, презентации, прайс-листа и файла другого формата.

## Отправить изображение

Данную команду можно добавить в скрипт, выбрав пункт **Отправить изображение** из списка **Команд**:



Настройка полей

После выбора данной команды в списке **Команд** перед вами откроется диалоговое окно настройки свойств:

## Добавить команду

Настройте свойства команды

☐ Использовать условие

Тип команды:

Отправить изображение

Browse

Файл с изображением:

Выберите файл metabot-ogi.png

Создать

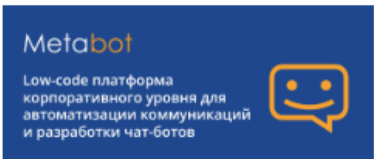
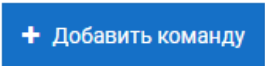
С пунктом **Использовать условие** вы можете познакомиться на странице [Условие выполнения команды JS](#).

В поле **Файл с изображением** необходимо с помощью кнопки **Выберите файл**, загрузить файл с изображением из проводника. Данная команда поддерживает любые графические файлы.

По кнопке **Создать** как итог команда появится в редакторе скрипта:

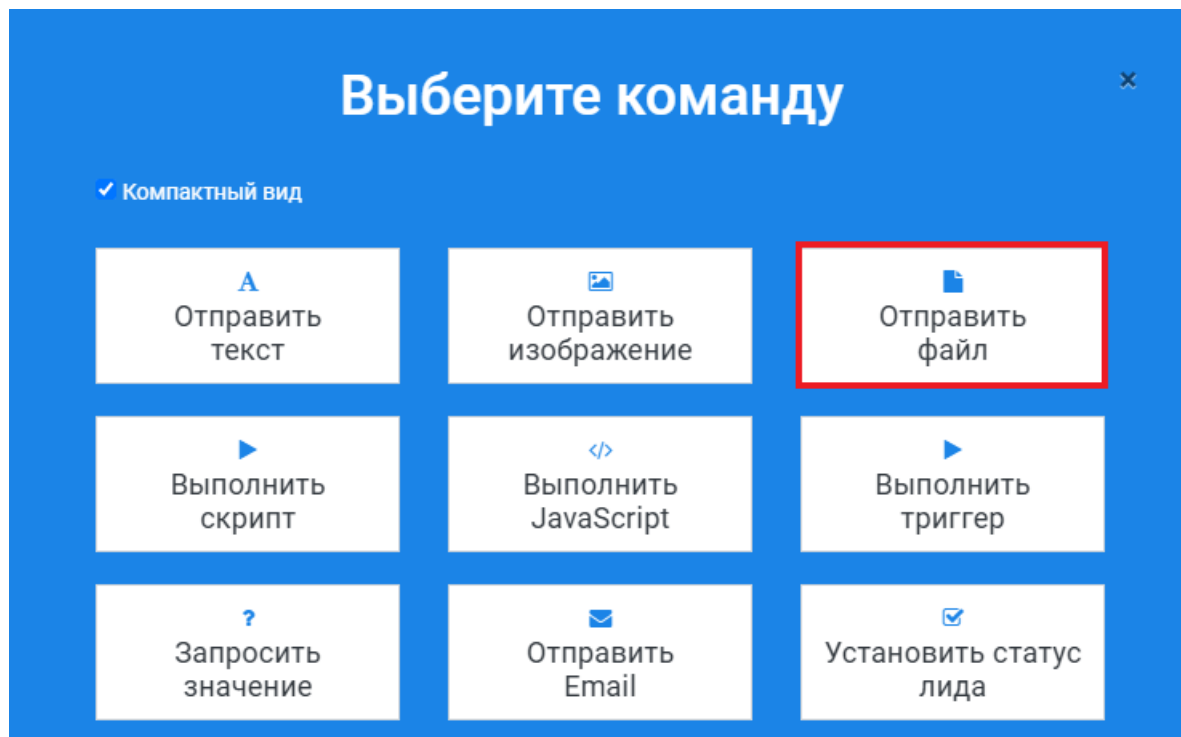
### КОМАНДЫ

Эти команды бот выполняет до того как покажет меню.

КОМАНДА	СОДЕРЖИМОЕ	ОПЕРАЦИИ
Отправить изображение		 
		

# Отправить файл

Данную команду можно добавить в скрипт, выбрав пункт **Отправить файл** из списка **Команд**:



## Настройка полей

После выбора данной команды в списке **Команд** перед вами откроется диалоговое окно настройки свойств:

# Добавить команду

Настройте свойства команды

☐ Использовать условие

Тип команды:

Отправить файл

Browse

Файл:

Выберите файл

Metabot.xlsx

Создать

В поле **Файл** необходимо с помощью кнопки **Выберите файл**, загрузить файл из проводника. Данная команда поддерживает любые форматы файлов.

По кнопке **Создать** как итог команда появится в редакторе скрипта:

КОМАНДЫ

Эти команды бот выполняет до того как покажет меню.

КОМАНДА	СОДЕРЖИМОЕ	ОПЕРАЦИИ
Отправить файл	Metabot.xlsx	<div><div></div><div></div></div>

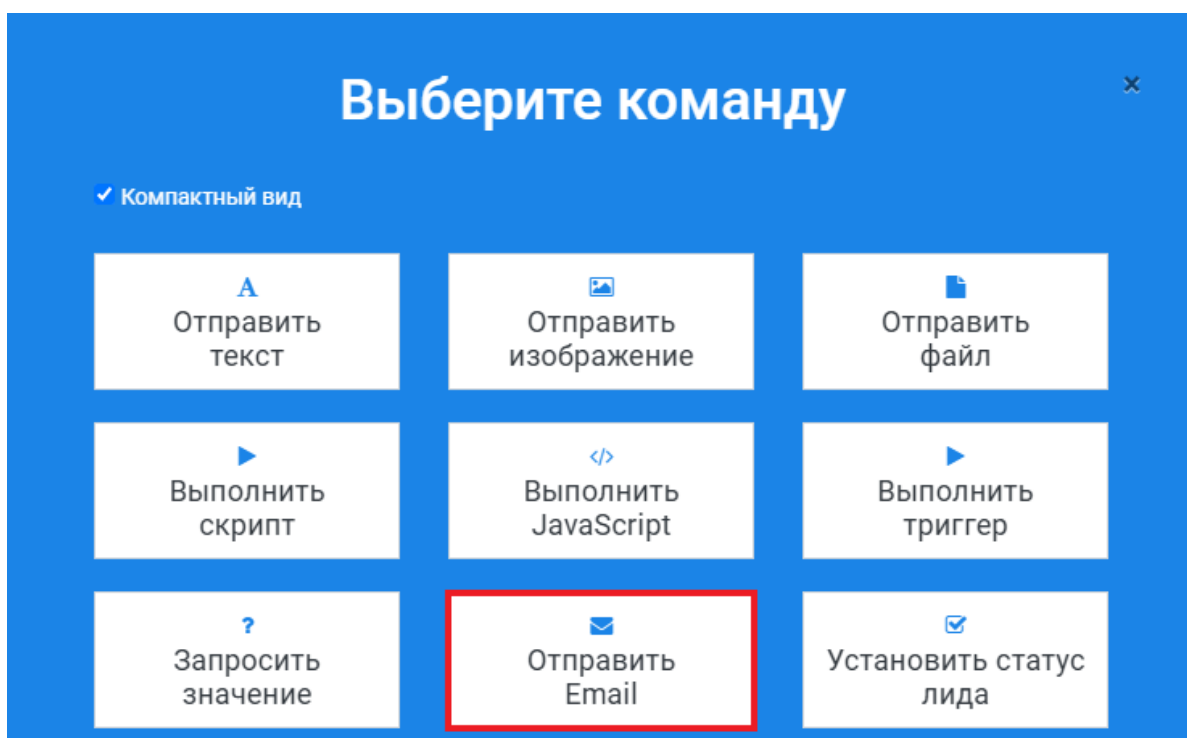
+ Добавить команду

# Отправить Email

Результатом работы данной команды является письмо отправленное пользователю на электронную почту.

Чаще всего Email уведомление используется в случае, если пользователь оставил заявку на консультацию по услугам, покупку товара или консультацию специалиста. Таким образом, обеспечивается уведомление владельца бизнеса о новой заявке, оставленной пользователем через бота.

Добавить в скрипт ее можно, выбрав пункт **Email** из списка **Команд**:



После выбора данной команды в списке **Команд** перед вами откроется диалоговое окно настройки свойств:

# Добавить команду

Настройте свойства команды



☐ Использовать условие

Тип команды:

Email	Browse
-------	--------

Получатель(-ли):

hi@metabot.org
----------------

Тема:

Показательное письмо для урока
--------------------------------

Содержимое:

<p>&lt;b&gt;Если у вас возникнут вопросы, вы можете обратиться за помощью к команде Metabot24.&lt;/b&gt;&lt;br&gt; &lt;br&gt; Звоните 📞 8 499 938-58-98&lt;br&gt; &lt;br&gt; Пишите на почту hi@metabot.org&lt;br&gt; &lt;br&gt; &lt;b&gt;Мы всегда на связи и рады помочь!&lt;/b&gt;</p>
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Создать

С пунктом **Использовать условие** вы можете познакомиться на странице [Условие выполнения команды JS](#).

Следующие поля являются обязательными:

- **Получатель(-ли)** — email пользователя-получателя уведомления. Пользователей может быть несколько, тогда email указываются последовательно через запятую (например: email1@mail.ru, email2@mail.ru и т.д.);

Вы можете взять email пользователей-получателей из атрибутов бота. Для этого в поле **Получатель(-ли)** введите `{{<имя атрибута>}}`, например: `{{email}}`

- **Тема** — тема email уведомления;
- **Содержимое** — текстовое содержания email уведомления (может содержать макропеременные).

Помните, что **текстовое содержимое** email письма - это **HTML разметка**! Это значит, что все текстовые строки должны заключаться в символы `<b></b>`, а перенос строки обозначаться символом `<br>`. Макропеременные описываются без заключения в символы.

Макропеременные доступные для команды **Email** можно посмотреть на странице [Доступные макропеременные для команды Отправить текст и Email](#).

По кнопке **Создать** как итог команда появится в редакторе скрипта:

#### КОМАНДЫ

Эти команды бот выполняет до того как покажет меню.

КОМАНДА	СОДЕРЖИМОЕ	ОПЕРАЦИИ
Email	<div>Получатель(-ли): hi@metabot.org</div> <div>Тема: Показательное письмо для урока</div> <div>&lt;b&gt;Если у вас возникнут вопросы, вы можете обратиться за помощью к команде Metabot24.&lt;/b&gt;&lt;br&gt;&lt;br&gt;Звоните ☎ 8 499 938-58-98&lt;br&gt;&lt;br&gt;Пишите на почту hi@metabot.org&lt;br&gt;&lt;br&gt;&lt;b&gt;Мы всегда на связи и рады помочь!&lt;/b&gt;</div>	<div> </div>

+ Добавить команду

# Установить статус лида

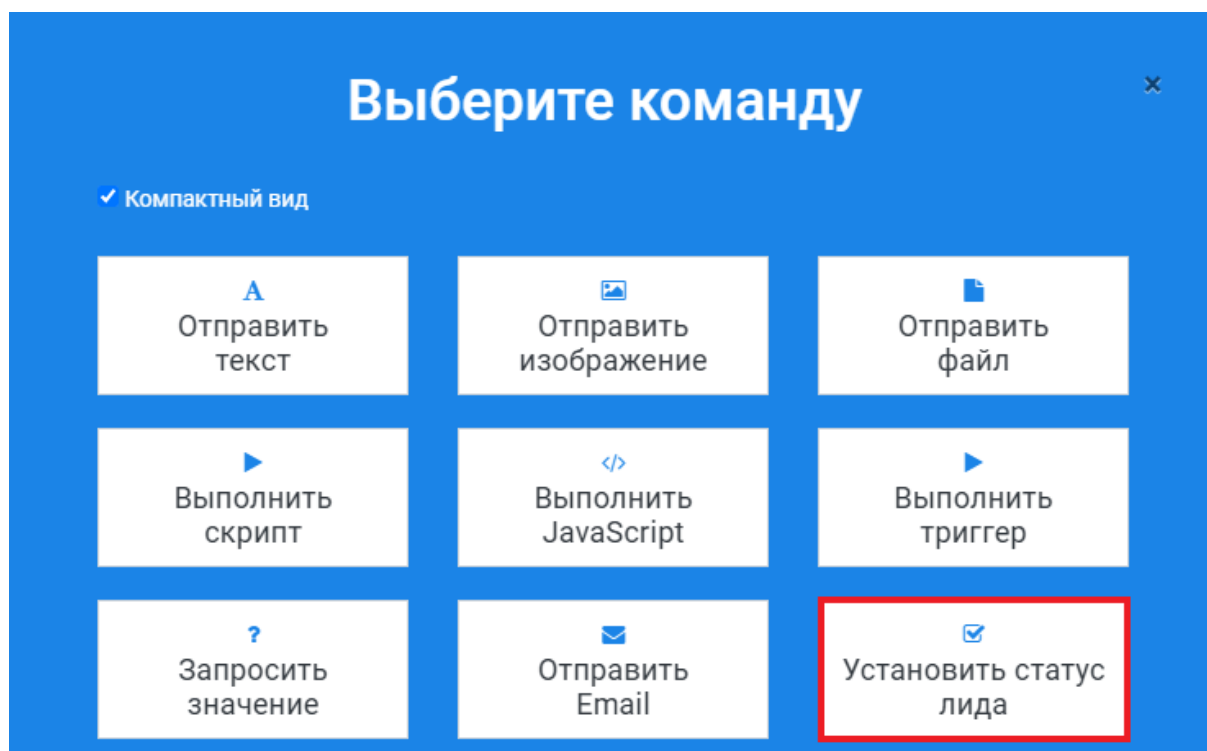
Результатом работы данной команды является присвоение статуса лиду в боте.

Статусы необходимы для классификации клиентов по этапам их обработки. С помощью статусов бот различает логику взаимодействия с клиентом.

Общий принцип данной команды таков — обязательно должен присутствовать начальный статус и два финальных статуса — для успешного и не успешного исхода. Между ними может быть любое количество статусов, это зависит от специфики вашего бизнеса и настройки бота.

Подробнее со статусами вы можете познакомиться на странице [Статусы](#).

Данную команду можно добавить в скрипт, выбрав пункт **Установить статус лида** из списка **Команд**:



После выбора данной команды в списке **Команд** перед вами откроется диалоговое окно настройки свойств:

# Добавить команду

Настройте свойства команды

☐ Использовать условие

Тип команды:

Установить статус лида

Browse

Новый статус обращения:

4960 - Начальный статус

▼

Создать

С пунктом **Использовать условие** вы можете познакомиться на странице [Условие выполнения команды JS](#).

В поле **Новый статус обращения** необходимо выбрать статус из выпадающего списка или добавить новый статус.

Новый статус добавляется кнопкой **Создать новый** в выпадающем списке:

Новый статус обращения:

4960 - Начальный статус

▲

[ Не указан ]

[ Создать новый ]

4960 - Начальный статус

4961 - Неуспешный исход

В отрывшемся диалоговом окне следует заполнить поле **Название** и нажать кнопку **Создать**:

# Создание статуса



Название:

Успешный исход

Создать

По кнопке **Создать** в диалоговом окне настроек свойств как итог команда появится в редакторе скрипта:

## КОМАНДЫ

Эти команды бот выполняет до того как покажет меню.

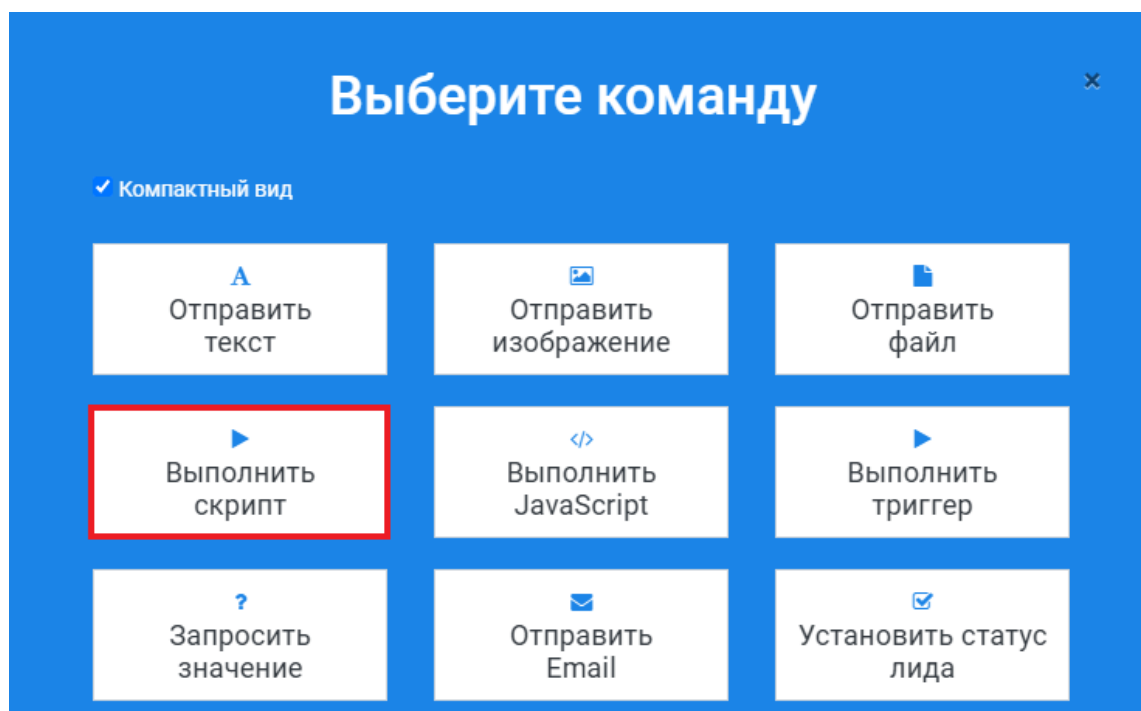
КОМАНДА	СОДЕРЖИМОЕ	ОПЕРАЦИИ
Установить статус лида	Успешный исход	 
 Добавить команду		

# Выполнить скрипт

Результатом работы данной команды является мгновенный переход бота на указанный скрипт.

Данная команда помогает настроить маршруты в боте и связать скрипты между собой. Самым ярким примером ее использования является создание в боте главного меню, к которому в конце будут приходить все маршруты.

Добавить в скрипт ее можно, выбрав пункт **Выполнить скрипт** из списка **Команд**:



После выбора данной команды в списке **Команд** перед вами откроется диалоговое окно настройки свойств:

# Добавить команду



Настройте свойства команды

☐ Использовать условие

Тип команды:

Выполнить скрипт	Browse
------------------	--------

Скрипт:

Меню	▼
------	---

Создать

С пунктом **Использовать условие** вы можете познакомиться на странице [Условие выполнения команды JS](#).

В параметре **Скрипт** необходимо выбрать из выпадающего списка скрипт, в который будет совершен переход.

По кнопке **Создать** как итог команда появится в редакторе скрипта:

## КОМАНДЫ

Эти команды бот выполняет до того как покажет меню.

КОМАНДА	СОДЕРЖИМОЕ	ОПЕРАЦИИ
Выполнить скрипт	Меню	<div><div> Перейти к скрипту</div><div></div></div>

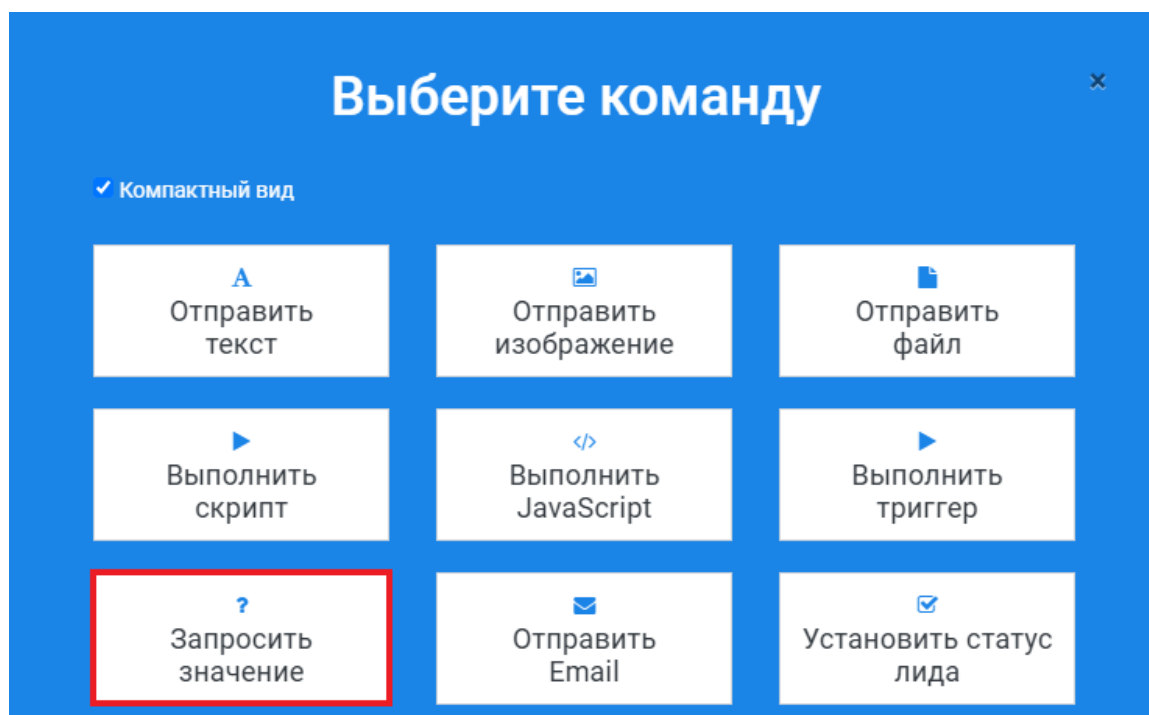
+ Добавить команду

# Запросить значение

**Результатом работы данной команды является сохранение значения данного пользователем в атрибуты лида.**

Общий принцип данной команды таков - бот задает вопрос пользователю, пользователь вводит ответ, значение этого ответа запоминается в боте. В большинстве своем эта команда используется для сбора контактных данных клиента.

Добавить в скрипт ее можно, выбрав пункт **Запросить значение** из списка **Команд**:



После выбора данной команды в списке **Команд** перед вами откроется диалоговое окно настройки свойств:

# Добавить команду



Настройте свойства команды

☐ Использовать условие

Тип команды:

Запросить значение	Browse
--------------------	--------

Имя переменной:

email
-------

Шаблон валидации:

Email	▼
-------	---

Регулярное выражение:

[^@\\s]+@[^@\\s]+\\.([^@\\s]+)
--------------------------------

Сообщение об ошибке:

Некорректный формат email
---------------------------

Подсказка-приглашение:

Введите свой адрес электронной почты.
---------------------------------------

Создать

С пунктом **Использовать условие** вы можете познакомиться на странице [Условие выполнения команды JS](#).

Следующие поля являются обязательными:

- **Имя переменной** — имя, с которым переменная будет записана в базу данных;
- **Шаблон валидации** — список шаблонов для вводимого пользователем значения (при выборе шаблона появятся два дополнительных поля):
  - **Регулярное выражение** — выражение задающее возможные значения, которые пользователь может ввести;

- **Сообщение об ошибке** — сообщение, выскакивающее при введении пользователем значения несоответствующего регулярному выражению;
- **Подсказка-приглашение** — текстовое сообщение помогающее пользователю понять какое значение ему требуется ввести.

## Шаблоны валидации

Шаблон валидации задает ограничение на вводимые пользователем значения. Если пользователь ввел значение несоответствующее шаблону, то ему придет сообщение об ошибке. Нужно это для исключения разных форматов одной переменной. Ведь есть человеческий фактор и каждый может написать ту же дату по-разному.

Ниже перечислены все существующие на платформе шаблоны и их значение.

Шаблон валидации	Описание
Дата ДД.ММ.ГГГГ	Дата в соответствующем формате
Дата ММ.ДД.ГГГГ	Дата в соответствующем формате
Дата ГГГГ.ММ.ДД	Дата в соответствующем формате
Российский номер телефона	Номер телефона в формате +7 (xxx) xxx-xxxx
Российский номер телефона, компактно	Номер телефона в формате 7xxxxxxxxxx
Российский мобильный	Мобильный в формате +7 (9xx) xxx-xxxx
Российский мобильный, компактно	Мобильный в формате 79xxxxxxxxxx
Число	Только числовые символы
Email	Формат email (*@*.*)
Произвольное рег. выражение	Регулярное выражение которое вы можете создать сами

Как составить свое произвольное регулярное выражение можно узнать на странице [Регулярные выражения](#).

По кнопке **Создать** как итог команда появится в редакторе скрипта:

КОМАНДЫ

Эти команды бот выполняет до того как покажет меню.

КОМАНДА	СОДЕРЖИМОЕ	ОПЕРАЦИИ
Запросить значение	Введите свой адрес электронной почты: [ Email ]   Регулярное выражение:Email ([^\s]+@[^\s]+\.[^\s]+, «Некорректный формат email»)	 
<div>+ Добавить команду</div>		

Введенные значения будут записаны в раздел **Атрибуты** — тут их можно редактировать и удалять. Так же они отобразятся в поле **Атрибуты** в разделе **Лиды**.

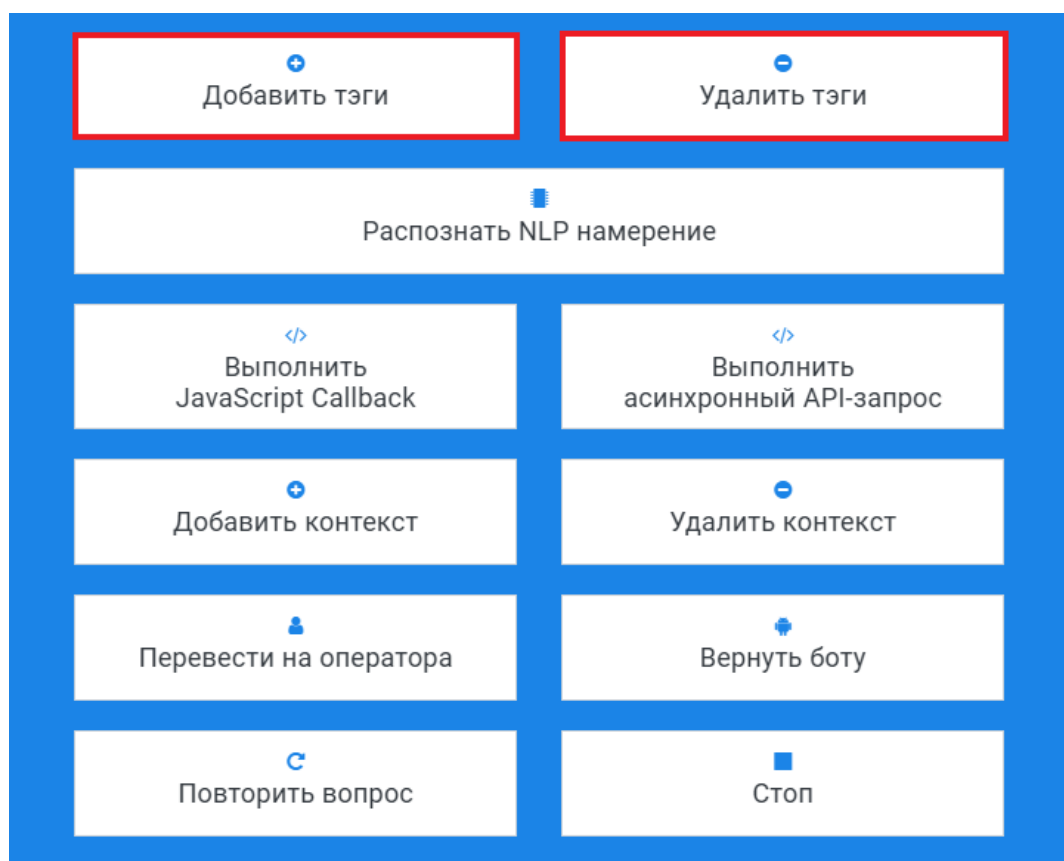
# Добавить тэги и удалить тэги

Результатом работы данных команд является добавление тэга в атрибуты пользователя или удаление из атрибутов.

На основании этих команд можно проанализировать в каких разделах бота был пользователь. Тэги записываются в хранилище на платформе и могут быть использованы для дальнейшего взаимодействия с пользователем. Также с помощью тэга можно настроить рассылку сообщений для цифровой автоворонки продаж.

Подробнее с тэгами вы можете ознакомиться в статье [Структура платформы](#) в разделе **11. Тэги и атрибуты**

Добавить в скрипт их можно, выбрав пункты **Добавить тэги** и **Удалить тэги** из списка **Команд**:



После выбора команды в списке **Команд** перед вами откроется диалоговое окно настройки свойств:

## Добавить команду



Настройте свойства команды

☐ Использовать условие

Тип команды:

Добавить тэги

Browse

Тэги:

тэг × тэг_2 ×

Введите имя тэга и нажмите **"** или **'**

Создать

С пунктом **Использовать условие** вы можете познакомиться на странице [Условие выполнения команды JS](#).

В поле **Тэги** необходимо ввести имя тэга и нажать **Enter**. Вы можете вводить в это поле больше одного тэга.

Имена тегов пишутся одним словом, без пробелов. Если тэг состоит из нескольких слов, то слова разделяются между собой знаком **"_"**

В случае команды **Удалить тэги**, если оставить поле **Тэги** пустым, то удалятся все тэги лида.

По кнопке **Создать** как итог команда появится в редакторе скрипта:

КОМАНДЫ

Эти команды бот выполняет до того как покажет меню.

КОМАНДА	СОДЕРЖИМОЕ	ОПЕРАЦИИ
Добавить тэги	<ul style="list-style-type: none"><li>тэг</li><li>тэг_2</li></ul>	<div><div></div><div></div><div></div></div>
Удалить тэги	тэг	<div><div></div><div></div><div></div></div>

+ Добавить команду

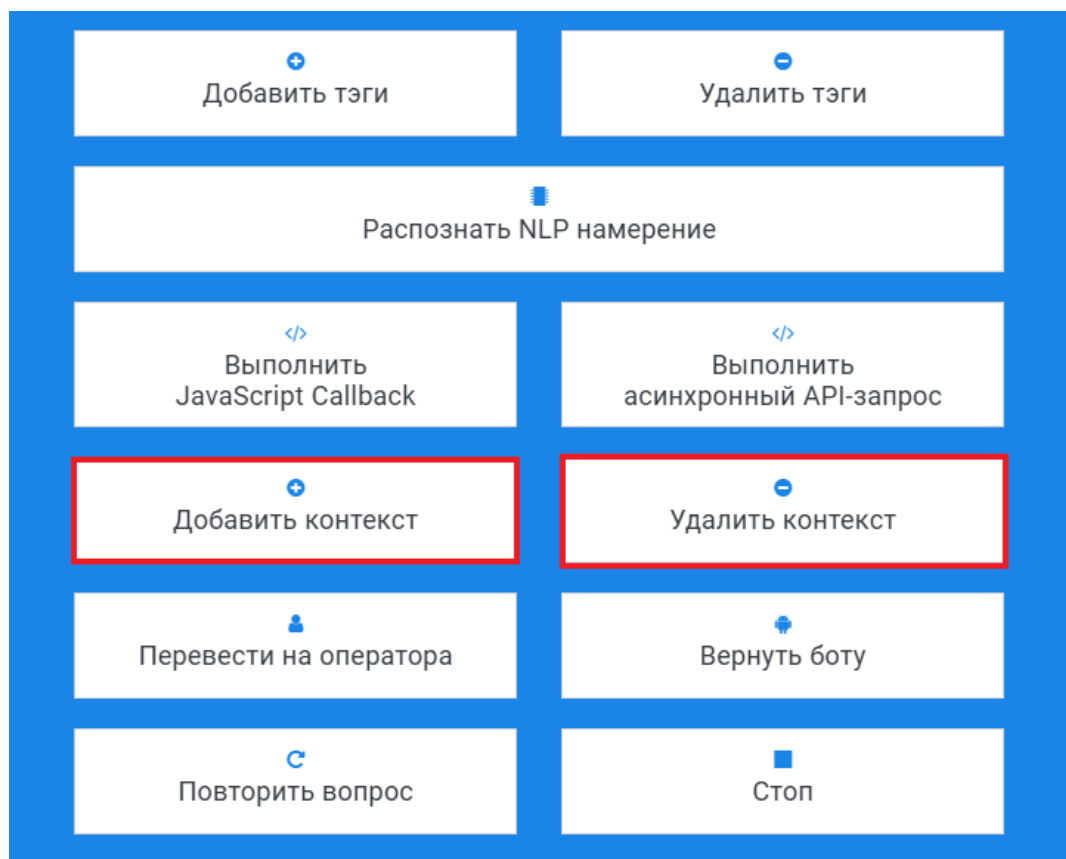
С тэгами так же можно работать с помощью JS методов. Все методы и их описание вы можете посмотреть в [Справочнике по функциям JS](#)

# Добавить контекст и удалить контекст

Результатом работы данных команд является добавление контекста скрипту или его удаление.

Контекст необходимо использовать как дополнительный параметр выполнения маршрутов при настройке бота. На его основании можно гибко настроить ветку выполнения маршрута по ранее выполненным действиям пользователя. Если при выполнении скрипта включается выполнение того или иного контекста, то бот будет выполнять только маршруты с данным контекстом.

Добавить в скрипт их можно, выбрав пункты **Добавить контекст** и **Удалить контекст** из списка **Команд**:



После выбора команды в списке **Команд** перед вами откроется диалоговое окно настройки свойств:

# Добавить команду

Настройте свойства команды



☐ Использовать условие

Тип команды:

Добавить контекст

Browse

Контекст:

контекст ×

Введите имя контекста и нажмите ";" или ":"

Создать

С пунктом **Использовать условие** вы можете познакомиться на странице [Условие выполнения команды JS](#).

В поле **Контекст** необходимо ввести имя контекста и нажать **Enter**. Вы можете вводить в это поле больше одного контекста.

Имена контекстов пишутся одним словом, без пробелов. Если контекст состоит из нескольких слов, то слова разделяются между собой знаком "_"

В случае команды **Удалить контекст**, если оставить поле **Контекст** пустым, то удалятся все контексты бота.

По кнопке **Создать** как итог команда появится в редакторе скрипта:

## команды

Эти команды бот выполняет до того как покажет меню.

КОМАНДА	СОДЕРЖИМОЕ	ОПЕРАЦИИ
Добавить контекст	контекст	  
Удалить контекст	контекст	  

+ Добавить команду

С контекстами так же можно работать с помощью JS методов. Все методы и их описание вы можете посмотреть в [**Справочнике по функциям JS**](#)

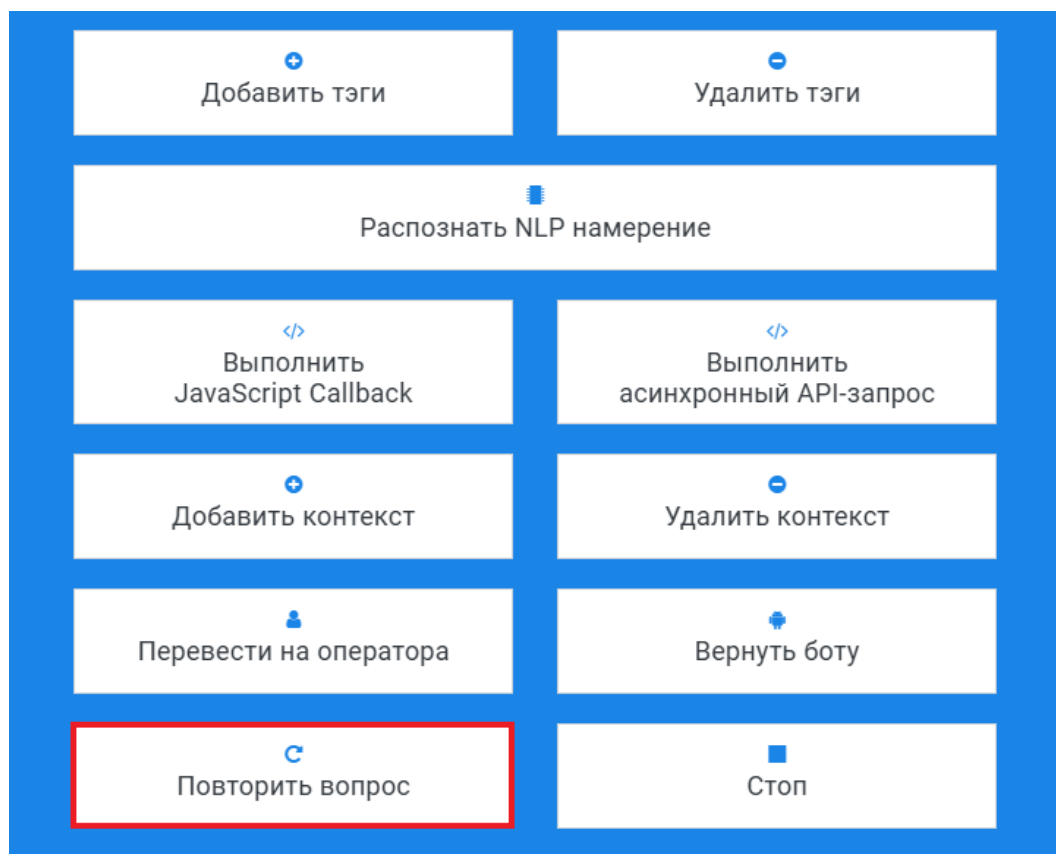
# Повторить вопрос

Результатом работы данной команды является выполнение последней команды после которой произошла ошибка.

Функциональную команду вывода повторного вопроса используют, если пользователь указал неправильный код в меню или система не распознала ответ на поставленный вопрос.

Команда **Повторить вопрос** применяется для дополнительного скрипта с типом **Fallback** и позволяет гибко настроить взаимодействие бота с пользователем. Таким образом, предотвращается остановка работы бота из-за нераспознанного или ошибочного ответа пользователя.

Добавить в скрипт ее можно, выбрав пункт **Повторить вопрос** из списка **Команд**:



После выбора данной команды в списке **Команд** перед вами откроется диалоговое окно настройки свойств:



# Добавить команду

Настройте свойства команды

☐ Использовать условие

Тип команды:

Повторить вопрос

Browse

Создать

С пунктом **Использовать условие** вы можете познакомиться на странице [Условие выполнения команды JS](#).

По кнопке **Создать** как итог команда появится в редакторе скрипта:

## КОМАНДЫ

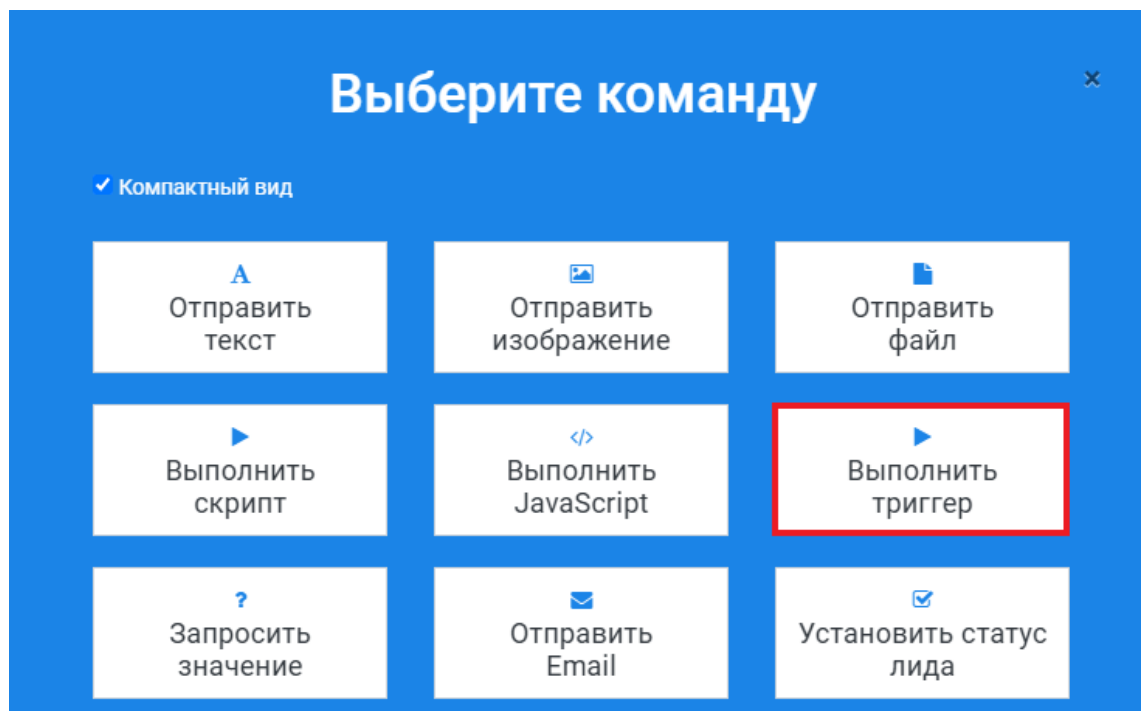
Эти команды бот выполняет до того как покажет меню.

КОМАНДА	СОДЕРЖИМОЕ	ОПЕРАЦИИ
Отправить текст	Извините, не понял ваш ответ. Попробуйте ввести по-другому.	  
Повторить вопрос		  
<a href="#">+ Добавить команду</a>		

Помните, что лучше всего использовать данную команду в скриптах типа **Fallback**. Добавьте в него команду **Отправить текст** с информационным сообщением пользователю о дальнейших действиях, а затем саму команду **Повторить вопрос**.

# Выполнить триггер

Результатом работы данной команды является прямой вызов триггера с проверкой условий в нем.



В большинстве случаев проще использовать данную команду для вызова триггера в нужном месте, чем использовать подписание на событие и настраивать триггер в точке скрипта с помощью JS условия.

С пунктом **Использовать условие** вы можете познакомиться на странице [Условие выполнения команды JS](#).

В команде есть следующие поля:

- **Триггер** — выпадающий список ранее созданных триггеров. Выбранный триггер будет вызываться командой;
- **Запустить указанный Триггер в** — дата и время с погрешностью в одну минуту, когда будет запущен триггер;
- **Запустить указанный Триггер через (минут)** — количество минут спустя которое будет запущен триггер.

Параметр **Запустить указанный Триггер через (минут)** игнорируется, если указано точное время запуска в поле **Запустить указанный Триггер в**.

По кнопке **Создать** как итог команда появится в редакторе скрипта:

#### КОМАНДЫ

Эти команды бот выполняет до того как покажет меню.

КОМАНДА	СОДЕРЖИМОЕ	ОПЕРАЦИИ
Выполнить триггер	Триггер В: 2023-02-25 14:36	<a href="#">✍ Перейти к триггеру</a> <a href="#">🗑</a>
<a href="#">+ Добавить команду</a>		

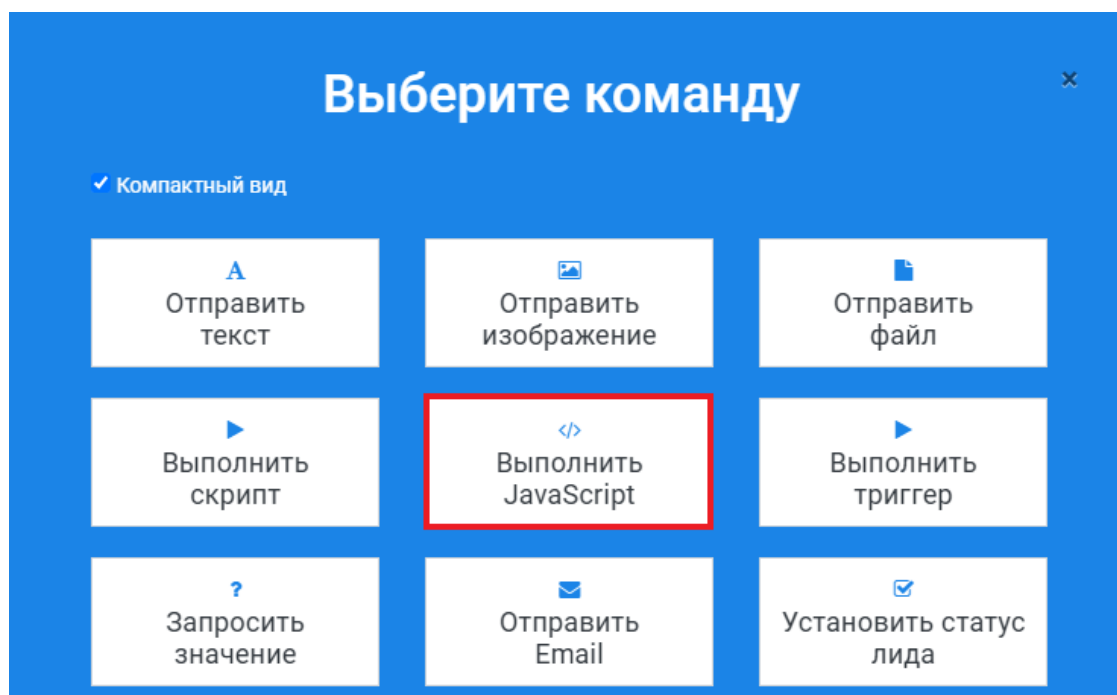
Здесь команда имеет операцию **Перейти к триггеру**. Переход по кнопке этой операции открывает форму редактирования данного триггера в разделе **Триггеры**.

# Выполнить JavaScript

Результатом работы данной команды является выполнение написанного в параметрах команды JavaScript кода.

Функциональная команда позволяет выполнить JavaScript, например для установки значений глобальных переменных (переменных бота) или атрибутов лида.

Добавить в скрипт ее можно, выбрав пункт **Выполнить JavaScript** из списка **Команд**:



После выбора данной команды в списке **Команд** перед вами откроется диалоговое окно настройки свойств:

# Добавить команду

Настройте свойства команды



☐ Использовать условие

Тип команды:

Выполнить JavaScript

Browse

JavaScript:

Создать

С пунктом **Использовать условие** вы можете познакомиться на странице [Условие выполнения команды JS](#).

В диалоговом окне, в поле **JavaScript** необходимо создать синтаксически верно написанный JavaScript.

Все методы и их описание вы можете посмотреть в [Справочнике по функциям JS](#).

По кнопке **Создать** как итог команда появится в редакторе скрипта:

## команды

Эти команды бот выполняет до того как покажет меню.

КОМАНДА	СОДЕРЖИМОЕ	ОПЕРАЦИИ
Выполнить JavaScript	<pre>var currentVar1Value = lead.getAttr('LeadVar1'); var glueStr = 'lead variable value '; var newValue = glueStr;  if (lead.isAttrExist('LeadVar1')) {     newValue = currentVar1Value + glueStr; }  if (currentVar1Value !== null) {     newValue = currentVar1Value + glueStr; }  lead.setAttr('LeadVar1', newValue);</pre>	<div></div>
<div> Добавить команду</div>		

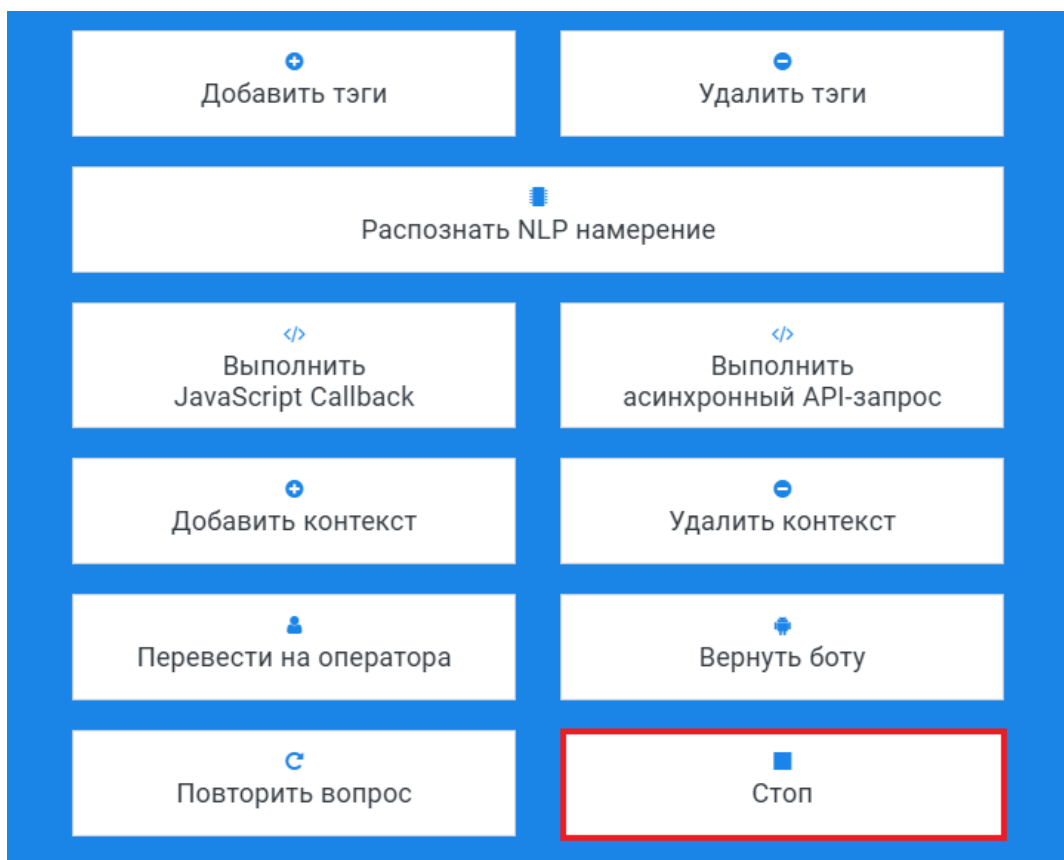


# Стоп

**Результатом работы данной команды является остановка работы бота.**

Функциональную команду Стоп обычно используется в конце скрипта одной или нескольких веток бота.

Добавить в скрипт ее можно, выбрав пункт **Стоп** из списка **Команд**:



После выбора данной команды в списке **Команд** перед вами откроется диалоговое окно настройки свойств:

# Добавить команду



Настройте свойства команды

☐ Использовать условие

Тип команды:

Стоп	Browse
------	--------

Создать

С пунктом **Использовать условие** вы можете познакомиться на странице [Условие выполнения команды JS](#).

По кнопке **Создать** как итог команда появится в редакторе скрипта:

## КОМАНДЫ

Эти команды бот выполняет до того как покажет меню.

КОМАНДА	СОДЕРЖИМОЕ	ОПЕРАЦИИ
Стоп		 
<a href="#">+ Добавить команду</a>		

Необходимо помнить, что данная команда должна быть добавлена в конце скрипта, так как работа бота остановится и все последующие команды не будут выполнены!

Для того, чтобы бот замолчал, но Лид мог в любой момент возобновить работу бота, необходимо перед командой **Стоп** выполнить команду **Установить статус лида**. Данная команда должна установить статус, для которого будет работать стартовый маршрут или будет выполнена команда **Вернуть боту**.

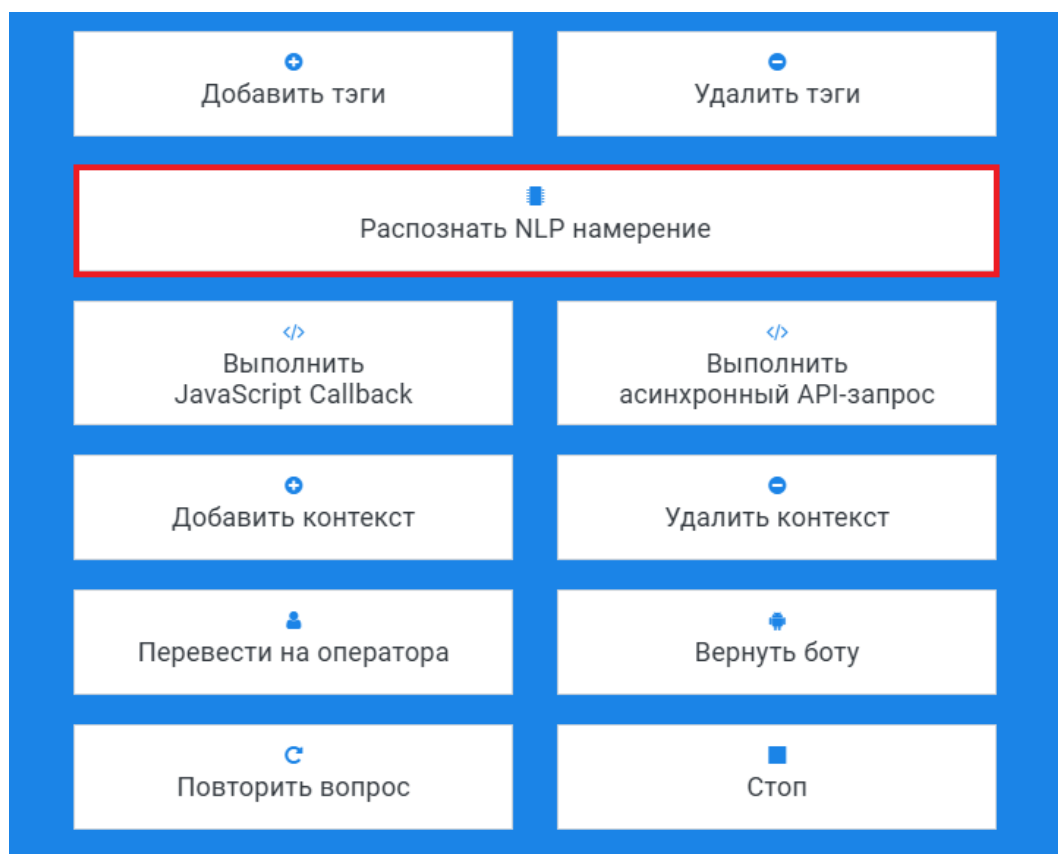
# Распознать NLP намерение

Результатом работы данной команды является выполнение соответствующего скрипта по распознанному намерению.

Если намерение не будет распознано или скрипт по намерению не найден, то будет выполнена следующая команда скрипта.

Подробнее с тем как работают NLP намерения можно узнать из урока [Автоматизируем службу поддержки](#).

Добавить в скрипт ее можно, выбрав пункт **Распознать NLP намерение** из списка **Команд**:



После выбора данной команды в списке **Команд** перед вами откроется диалоговое окно настройки свойств:

# Добавить команду

Настройте свойства команды



☐ Использовать условие

Тип команды:

Распознать NLP намерение	Browse
--------------------------	--------

Текст для обнаружения намерения (будет отправлен в NLP):

заказать услугу
-----------------

Создать

С пунктом **Использовать условие** вы можете познакомиться на странице [Условие выполнения команды JS](#).

В поле **Текст для обнаружения намерения (будет отправлен в NLP)** нужно ввести текст, который соответствует нужному вам намерению в **Dialogflow**. По этому намерению и будет выполнен соответствующий скрипт бота.

По кнопке **Создать** как итог команда появится в редакторе скрипта:

## команды

Эти команды бот выполняет до того как покажет меню.

КОМАНДА	СОДЕРЖИМОЕ	ОПЕРАЦИИ
Распознать NLP намерение	заказать услугу	 

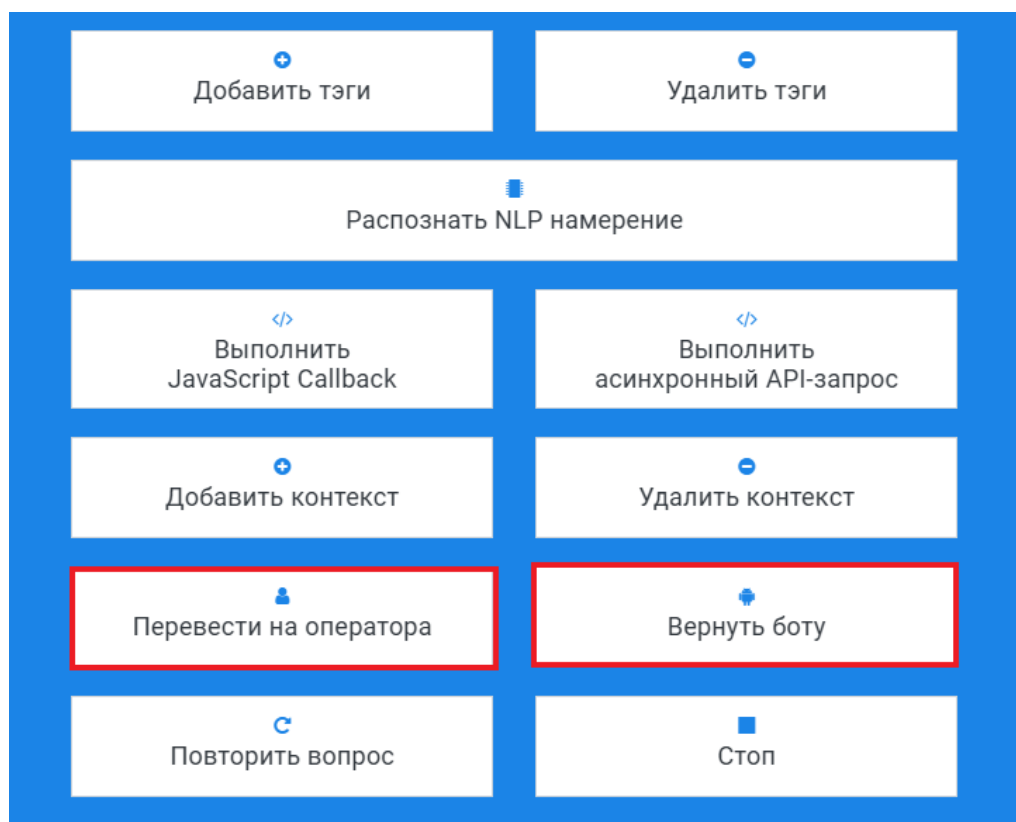
+ Добавить команду

# Перевести на оператора и Вернуть боту

Результатом работы данных команд является перевод диалога на оператора CRM или перевод обратно с оператора на бота.

Функциональная команда **Перевести на оператора** позволяет выполнить перевод на оператора и установить по лиду параметр **Чат передан оператору**. А функциональная команда **Вернуть боту** позволяет выполнить возврат диалога обратно боту и автоматически снимает по лиду этот параметр.

Добавить в скрипт их можно, выбрав пункты **Перевести на оператора** и **Вернуть боту** из списка **Команд**:



После выбора команды в списке **Команд** перед вами откроется диалоговое окно настройки свойств:

# Добавить команду

Настройте свойства команды



☐ Использовать условие

Тип команды:

Перевести на оператора

Browse

Если не поддерживается, то перевести в статус:

4966 - Оператор

Создать

С пунктом **Использовать условие** вы можете познакомиться на странице [Условие выполнения команды JS](#).

В поле **Если не поддерживается, то перевести в статус** можно указать в какой статус при выполнении данной команды будет переведен лид. Поле не обязательное для заполнения.

При выполнении команды **Перевести на оператора** по Лиду будет установлен параметр **Чат передан оператору** и бот будет молчать. Командой Вернуть боту по лиду снимается параметр **Чат передан оператору** и она должна быть установлена в скрипт, который будет запускаться по маршруту с активным параметром **Действует при переводе на оператора**.

По кнопке **Создать** как итог команда появится в редакторе скрипта:

## команды

Эти команды бот выполняет до того как покажет меню.

КОМАНДА	СОДЕРЖИМОЕ	ОПЕРАЦИИ
Перевести на оператора	Оператор	  
Вернуть боту	Начальный статус	  

+ Добавить команду

Как правильно использовать данные команды можно подробнее узнать в уроке **Автоматизируем службу поддержки**.

С переводом на оператора так же можно работать с помощью JS методов. Все методы и их описание вы можете посмотреть в **Справочнике по функциям JS**

# Выполнить JavaScript Callback

**Результатом работы данной команды является прием файлов в Телегам боте, отправленных пользователем.**

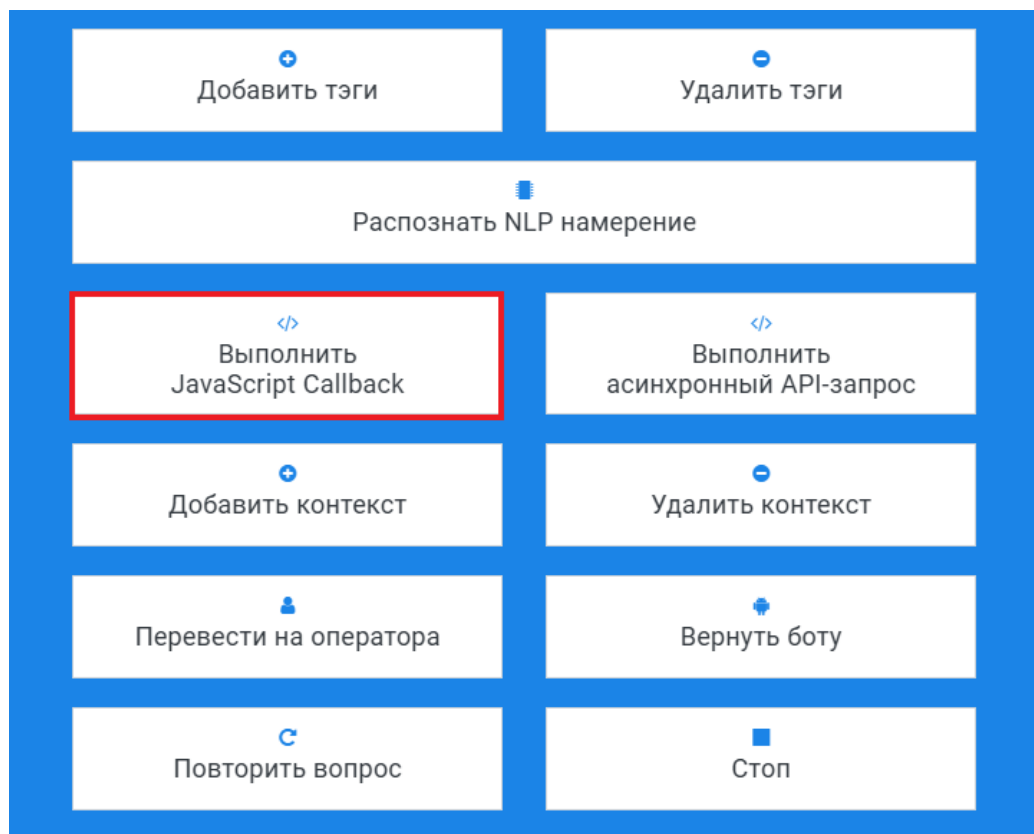
Принцип работы данной команды:

- Когда данная команда запущена, ее состояние за циклируется в текущей точке и текущий и любой следующий входящий вебхук будет обрабатываться этим JavaScript кодом;
- Выйти из команды и перейти к следующей или к другому скрипту можно только с помощью возврата специального значения из JavaScript

Данную команду можно использовать, для:

- Запроса любого значения: текста, файла, фото, медиа;
- Интеграции с любой системой: машиной JSON состояний или любым NLP (за цикливая диалог на NLP), системой обработки голосовых сообщений, системой распознавания изображений.

Добавить в скрипт ее можно, выбрав пункт **Выполнить JavaScript Callback** из списка **Команд**:



После выбора данной команды в списке **Команд** перед вами откроется диалоговое окно настройки свойств:

# Добавить команду

Настройте свойства команды

☐ Использовать условие

Тип команды:

Выполнить JavaScript Callback	Browse
-------------------------------	--------

Сообщение-вопрос отправляемое до старта данной команды:

Прикрепите фото.
------------------

Сообщение при возникновении ошибки:

Фото имеет неподходящий формат.
---------------------------------

☒ Запустить Callback сразу при старте этой команды

JavaScript Callback:

--

Создать

С пунктом **Использовать условие** вы можете познакомиться на странице [Условие выполнения команды JS](#).

Следующие поля являются обязательными:

- **Сообщение-вопрос отправляемое до старта данной команды** — сообщение которое будет отправлено в мессенджер при старте данной команды, поле обязательно для заполнения ТОЛЬКО, если выключено поле **Запустить Callback сразу при старте этой команды**;
- **Сообщение при возникновении ошибки** — сообщение которое будет отправлено в мессенджер при возникновении ошибки;
- **Запустить Callback сразу при старте этой команды** — если поле неактивно, то JavaScript код запустится только после того как пользователь получит **Сообщение-вопрос отправляемое до старта данной команды** в мессенджер и что-то

ОТВЕТИТ;

Если поле активно и это первый запуск **JavaScript Callback** данной команды, то в **JavaScript** будет установлена в **true** системная предопределенная переменная **isFirstImmediateCall**, во всех других случаях **isFirstImmediateCall** устанавливается в **false**. Это необходимо, чтобы можно было проинициализировать компонент и, например, выслать текст вопроса или отправить изначально меню и т.д.

- **JavaScript Callback** — здесь указывается исходный код вашего JavaScript компонента.

По кнопке **Создать** как итог команда появится в редакторе скрипта:

Выполнить JavaScript Callback

Сообщение-вопрос отправляемое до старта данной команды:  
Прикрепите фото.

Сообщение при возникновении ошибки:  
Фото имеет неподходящий формат.

Запустить Callback сразу при старте этой команды: Да

JavaScript Callback:

```
const { imageUploader } = require("Common.TelegramComponents.ImageUploader")
imageUploader.setMaxFileSize(2)
imageUploader.setMinImageWidth(0)
imageUploader.setMinImageHeight(0)
imageUploader.setScriptCodeForFilesIsNotSupported("files_is_not_supported")
imageUploader.setFilesCustomTableName("person_files")
imageUploader.setBtnCancelCaption("Отмена")
imageUploader.setBtnOkCaption("Готово")
imageUploader.calcPromptMessage = function() {
 let maxFileSize = this.getMaxFileSizeMb()
 let minImageWidth = this.getMinImageWidth()
 let minImageHeight = this.getMinImageHeight()
 let msg = "Отправьте свое фото"
 (maxFileSizeMb > 0 ? ("\\r\\n\\n<i>Размер фото: не более " + maxFileSizeMb + " Мб</i>") : "") +
 (minImageWidth > 0 ? ("\\r\\n\\nШирина: не менее " + minImageWidth + " пкс</i>") : "") +
 (minImageHeight > 0 ? ("\\r\\n\\nВысота: не менее " + minImageHeight + " пкс</i>") : "")
 return msg
}
imageUploader.calcPromptKeyboard = function() {
 return [
 [
 {
 "text": this.calcBtnCancelCaption(),
 "callback_data": "btn_cancel",
 "text_button_label": this.getBtnCancelNoInlineLabelCode()
 },
],
]
}
return imageUploader.getComponentResult()
```

## Возвращаемые значения

JavaScript Callback всегда должен возвращать данные с помощью «**return**» в определенном формате:

## Для завершения инициализации

- **return null.**

Используется, чтобы платформа ничего не делала со своей стороны, ожидая ответа пользователя. Вместо **retrurn null** можно ничего не указывать после ключевого слова **return**.

Для прерывания цикла

- **return true;**
- **return { "result": true ... другие параметры };**
- **return { "break": true ... другие параметры }.**

Используется для выхода из замыкания и переходу к следующей команде текущего скрипта (если не указан параметр для перехода к другому скрипту).

Прерывание с помощью **throw new Error("break")** недопустимо, в таком случае JS прервется, но из заикливания не выйдет, т.е. будет распознано как **return null**.

Под "**другими параметрами**" здесь подразумевается, что можно указать параметры в JSON-объекте, на которые платформа умеет реагировать. На данный момент для **"result": true**, можно указывать следующие параметры:

- **"run_script_by_code": "код_вашего скрипта"** — если указан этот параметр, то выполняется выход из замыкания и переход к указанному скрипту по коду.

## Для повторения цикла

- **return false;**
- **{ "result": false ... другие параметры }.**

Используется для завершения обработки JavaScript (выхода из JavaScript), без прерывания заикливания. Callback продолжит выполнение цикла и при приеме любого входящего сообщения мы опять попадем в JavaScript код данной команды (если не указан параметр для перехода к другому скрипту).

Под "**другими параметрами**" здесь подразумевается, что можно указать параметры в JSON-объекте, на которые платформа умеет реагировать. На данный момент для **"result": false**, можно указывать следующие параметры:

- **"fallback": "текст вашего фолбэка"** — используется для вывода вместе с фолбэком текста-объяснения;
- **"run_script_by_code": "код_вашего скрипта"** — используется для запуска другого скрипта по коду, если параметр передан, то fallback и повтор вопроса выводиться не будут;
- **"is_repeat_prompt": true / false** — выводить или нет повтор вопроса. Повтор вопроса выводится только, если он указан в настройках команды.

# Доступные макропеременные для команды Отправить текст и Email

**Макропеременная** — это переменная содержащая какое-либо часто используемое значение. На платформе Metabot24 есть ряд системных макропеременных, которые можно использовать в командах **Отправить текст** и **Email**.

Название	Макропеременная
Название бота	{{ \$botName }}
Имя лида	{{ \$name }}
Значение атрибута	{{ \$varName }}
Тэг лида	{{ #tagName }}
Контекст лида	{{ @contextName }}
Последнее распознанное NLP намерение	{{ \$nlpLastIntentName }}
Последний распознанный NLP Action	{{ \$nlpLastActionName }}
Последний текстовый результат NLP	{{ \$nlpLastReponseText }}

Заявка	{{ <b>ticket.*</b> }}
JSON параметры передаваемые через методы JS	{{ <b>\$\$sys_last_script_request_params.varName</b> }}

Некоторые из макропеременных будут описаны подробнее ниже.

## Значение атрибута

{{ **\$varName** }}

 — переменная-атрибут лида, где вместо **varName** нужно указать название атрибута. Содержит значение соответствующей переменной сохраненной в атрибутах лида.

Что такое атрибуты вы можете подробнее узнать на странице [Структура платформы](#) в разделе **11. Тэги и атрибуты**.

## Тэг лида

{{ **#tagName** }}

 — тэг лида, где вместо **tagName** нужно указать название тэга. Содержит значение “да” или “нет”, в зависимости от того установлен или нет соответствующий тэг для лида.

Что такое тэги вы можете подробнее узнать на странице [Структура платформы](#) в разделе **11. Тэги и атрибуты**.

## Контекст лида

{{ **@contextName** }}

 — контекст лида, где вместо **contextName** нужно указать название контекста. Содержит значение “да” или “нет”, в зависимости от того установлен или нет соответствующий контекст для лида.

## Последний текстовый результат NLP

Используется в сочетании с DialogFlow Smalltalk Actions - для вывода в качестве ответа бота.

Узнать, что такое Smalltalk вы можете на странице [Dialogflow: Small talk](#).

## Заявка

{{ **ticket.*** }}

 — где * — любое из полей перечисленных ниже:

- **id** — id заявки;

- **business_id** — id бизнеса;
- **bot_id** — id бота;
- **author_person_id** — id персоны-автора заявки;
- **client_person_id** — id персоны-клиента заявки;
- **assigned_person_id** — id персоны-исполнителя заявки;
- **ticket_type_id** — id типа заявки;
- **status_id** (status.id ; статус.id) — id статуса заявки;
- **title** (заголовок ; название) — название заявки;
- **ticket.description** (ticket.описание) — описание заявки;
- **ticket.createdAt** (ticket.датаСоздания) — дата создания заявки;
- **ticket.updatedAt** (ticket.датаОбновления) — дата последнего обновления заявки;
- **status.name** (статус.имя) — название статуса заявки;
- **authorPerson.fullName** (автор.полноеИмя) — полное имя автора заявки;
- **assignedPerson.fullName** (исполнитель.полноеИмя) — полное имя исполнителя заявки;
- **clientPerson.fullName** (клиент.полноеИмя) — полное имя клиента заявки;
- **ticketType.name** (типЗаявки.имя) — название типа заявки;
- **ticket.field_{ID}** ({ID} — id поля) — содержимое поля заявки;
- **ticket.{name}** ({name} — название поля) — содержимое поля заявки;
- **ticket.{title}** ({title} — заголовок поля) — содержимое поля заявки.

В зависимости от выбранного поля будет содержать соответствующее ему значение.

Подробнее с заявками вы можете познакомиться на странице [Заявки](#).

## JSON параметры

`{{ $$sys_last_script_request_params.varName }}` — JSON параметры передаваемые через методы JS, например, через **bot.runScriptByCodeForPerson()** в параметре **jobParams**.

Вместо **varName** нужно указать название поля JSON. Содержит значения соответствующей JSON переменной сохраненной в атрибутах лида после вызова метода, перехода по DeepLink или получения запроса API.

# Регулярные выражения

**Регулярное выражение** — это шаблон задающий ограничение на вводимые пользователем значения. Оно может состоять как из обычных, так и из специальных символов и их комбинаций.

Обычные символы интерпретируются буквально, например регулярное выражение **абв** интерпретируется как строгая последовательность соответствующих букв в этом порядке.

Помимо этого, для составления более сложных регулярных выражений используются специальные символы. Их написание и значение подробнее расписаны ниже.

## Специальные символы

Символ	Описание	Пример
Общие символы и символы количества		
\	Для <b>обычных символов</b> , означает что следующий символ будет интерпретироваться как специальный. Для <b>специальных символов</b> , означает что следующий символ будет интерпретироваться буквально.	<b>\a</b> — символ <b>a</b> становится специальным символом, означающим границу слова. <b>*</b> — специальный символ <b>*</b> интерпретируется как обычный символ "звездочка".
^	Следующий символ должен присутствовать в начале строки.	<b>^A</b> — символ <b>A</b> должен быть первым в строке.
\$	Предыдущий символ должен присутствовать в конце строки.	<b>a\$</b> — символ <b>a</b> должен быть последним в строке.
*	Предыдущий символ может быть повторен 0 или больше раз.	<b>a*</b> — символ <b>a</b> может не быть в строке, а может повториться множество раз.
+	Предыдущий символ должен быть повторен 1 или больше раз.	<b>a+</b> — символ <b>a</b> может повториться один раз в строке, а может повториться множество раз.
?	Предыдущий символ должен быть повторен 0 или 1 раз.	<b>a?</b> — символ <b>a</b> может не быть в строке, а может быть один раз.

.	Любой символ кроме переноса строки.	<b>a.c</b> равно <b>abc</b> , <b>a1c</b> и тд.
<b>x y</b>	В строке должен присутствовать либо x либо y.	<b>abc xyz</b> — в строке может быть <b>abc</b> или <b>xyz</b> .
<b>{n}</b>	Предыдущий символ должен быть повторен n раз.	<b>a{2}</b> — <b>a</b> должна быть повторена в строке дважды.
<b>{n,m}</b>	Предыдущий символ должен быть повторен не меньше n раз, но и не больше m раз.	<b>a{2,5}</b> — <b>a</b> должна быть повторена в строке хотя бы дважды и не больше пяти раз.
Скобочные группы		
<b>(x)</b>	Объединение символов в единый элемент (с запоминанием).	<b>abc*</b> — * относится только к <b>c</b> .  <b>(abc)*</b> — * относится ко всему выражению в скобках.
<b>(?:x)</b>	Объединение символов в единый элемент (без запоминания).	
<b>x(?:=y)</b>	Соответствует x только если за ним идет y.	<b>ab(?:=c)</b> — после <b>ab</b> должна быть <b>c</b> .
<b>x(?:!y)</b>	Соответствует x только если за ним не идет y.	<b>ab(?:!c)</b> — после <b>ab</b> не должно быть <b>c</b> .
Соответствия набору символов		
<b>[xyz]</b>	Соответствует любому из символов в скобках.	<b>[abcd]</b> или <b>[a-d]</b> — строка должна состоять из любых символов от <b>a</b> до <b>d</b> .
<b>[^xyz]</b>	Соответствует любому из символов не в скобках.	<b>[^abcd]</b> или <b>[^a-d]</b> — строка должна исключать символы от <b>a</b> до <b>d</b> .
<b>[\b]</b>	Бэкспейс.	<b>[\b]</b> — находит символ бэкспейса в строке.
<b>\b</b>	Граница слова.	<b>\b</b> — перед или после предыдущего символа должен быть пробел или другая граница слова.
<b>\d</b>	Цифровой символ.	<b>\d</b> — любой из символов от 0 до 9.
<b>\D</b>	Нецифровой символ.	<b>\D</b> — любой из символов кроме символов от 0 до 9.
<b>\s</b>	Пробел.	<b>\s</b> — находит пробел, табуляцию, перенос строки.

<code>\S</code>	Не пробел.	<code>\S</code> — любой из символов кроме пробела, табуляции, переноса строки.
<code>\w</code>	Цифробуквенные символы и нижнее подчеркивание.	<code>\w</code> — любой из символов от 0 до 9, от А до Я и <code>_</code> .
<code>\W</code>	Нецифробуквенные символы.	<code>\W</code> — любой из символов кроме символов от 0 до 9, от А до Я и <code>_</code> .
<code>\0</code>	NULL	<code>\0</code> — находит пустое множество.

# Примеры регулярных выражений и их разбор

## Дата формата ДД.ММ.ГГГГ

**Выражение:** `(0?[1-9]|[12][0-9]|3[01])\.(0?[1-9]|1[012])\.(19|20)\d\d`

<code>(0?[1-9] [12][0-9] 3[01])</code>	
<code>0?[1-9]</code>	0 может быть повторен 1 или 0 раз, а за ним идет любая цифра кроме 0.
<code>[12][0-9]</code>	1 или 2, а за ними любая цифра.
<code>3[01]</code>	3 и 0 или 1.
<code> </code>	или.
<code>(0?[1-9] 1[012])</code>	
<code>1[012]</code>	1 и 0 или 1 или 2.
<code>\.(19 20)\d\d</code>	
<code>\.</code>	обычный символ точки.
<code>19 20</code>	19 или 20.
<code>\d</code>	любая цифра.

**Возможная строка:** 11.03.2005

## Email

**Выражение:** [^@\\s]+@[^@\\s]+\\.([^@\\s]+)

[^\s@]+	
[\s@]	любой из символов кроме @ и пробела.
+	предыдущий символ может быть повторен 1 или более раз.
@[^\s@]+	
@	символ @.
\.[^\s@]+	
\.	обычный символ точки.

**Возможная строка:** email@pp.com

## Другие регулярные выражения

## Email без латинских букв

```
(?:[a-z0-9!#$%&'*+,-./:;=?@^_`{|}~]+(?:\.[a-z0-9!#$%&'*+,-./:;=?@^_`{|}~]+)*)*"(?:[x01-x08x0bx0cx0e-x1fx21x23-x5bx5d-x7f]|\[\x01-x09x0bx0cx0e-x7f\])"@(?:[a-z0-9](?:[a-z0-9]*[a-z0-9])?|\.)\.[a-z0-9](?:[a-z0-9]*[a-z0-9])?|(?:(?:2(5[0-5])?[0-4][0-9])|1[0-9][0-9]|1[0-9]?[0-9])\.)\{3(?:2(5[0-5])?[0-4][0-9])|1[0-9][0-9]|1[0-9]?[0-9])[a-z0-9]*[a-z0-9](?:[x01-x08x0bx0cx0e-x1fx21-x5ax53-x7f]|\[\x01-x09x0bx0cx0e-x7f\])\+\\)
```

Для валидации номера телефона без рамок кодов страны

`[\+|-]?[\d| ]+[\.\,]?[\d| ]*`

**Возможная строка:** +7 900 700 30 10 / 79007003010 / 8900 700 30 10 / + 37 900 700 30 10

## Валидация номера с кодом определенных стран и возможностью выйти

$$(\wedge(\backslash+?7|\backslash+?37)\backslash d\{10\}$|отмена|назад|выход|стоп)$$

**Возможная строка:** +79007003010 / +379007003010 / +77007003010

**Имя или слово с заглавной**

[А-Я]{1}[а-я]{2,20}

**Возможная строка:** Метабот

# Условие выполнения команды JS

При создании и редактировании каждой команды скрипта можно увидеть поле **Использовать условие**. Данное поле при активации позволяет описать JS условие выполнения или невыполнения команды.

После его активации появляется новое поле — **Условие выполнения**:

Добавить команду

Настройте свойства команды

☒ Использовать условие

Условие выполнения:

Тип команды:

Отправить текст

Browse

Текст:

Создать

Его можно заполнить JavaScript кодом. Стандартный код условия выглядит так:

```
if (lead.isTagExist('some_tag')) {
 return true;
}
```

В данном случае представлен код с проверкой существования тега. В самом условии может проверяться что угодно (теги, атрибуты, их значение и т.д.). Главное, для корректной работы условия, код обязательно должен вернуть **true** или **false**.