

Примеры использования методов

- [Использование методов блокировок](#)
- [Использование методов планирования времени](#)
- [Использование методов работы с файлами](#)

Использование методов блокировок

Ниже представлен JS скрипт на примере которого будет рассмотрена работа с методами блокировок.

```
bot.sendText(' Ожидая получения блокировки' )

let lockName = 'my_lock_1'
let lockPrefix = ''
let ttlSec = 5 // время жизни блокировки
let maxWaitSec = 300 // макс время ожидания блокировки

let isLocked = false
// Получаем блокировку по бизнесу (bot.waitForBusinessLock) или боту (bot.waitForBotLock)
isLocked = bot.waitForBusinessLock(lockName, lockPrefix, ttlSec, maxWaitSec)
if (isLocked) {
    bot.sendText(' Блокировка получена' )
} else {
    bot.sendText(' Время ожидания блокировки истекло' )
}

bot.sendText(' Ожидая истечения блокировки и получаю новую' )

//bot.releaseAllCurrentLocks()

ttlSec = 20
maxWaitSec = 1 // если раскомментировать то время ожидания будет превышено
                // (если проверять превышение то закомментировано выше должно быть maxWaitSec)

// Повторно, получаем новую блокировку по бизнесу с тем же ключем
isLocked = bot.waitForBusinessLock(lockName, lockPrefix, ttlSec, maxWaitSec)
if (isLocked) {
    bot.sendText(' Повторная блокировка получена' )
} else {
```

```

    bot.sendText(' Время ожидания повторной блокировки истекло' )
}

// Блокировка получена или время ожидания истекло
bot.sendText(' Блокировка получена' )

// -----

bot.sendText(' Ищем и удаляем блокировку' )

let hasLock = bot.hasLockForBusiness( lockName)

if (hasLock) {
    // Удаляем блокировку
    // по бизнесу - bot.releaseLockForBusiness / по боту - bot.releaseLockForBot
    // на самом деле это не releaseLock а forceReleaseLock (тк можно разблокировать даже если
ее тут не получали)
    // но зато это гарантирует удаление блокировки, поэтому префикс force в начале убран
    // для удаления же блокировки по тек скрипту можно использовать
bot.releaseCurrentLock(lockName) или bot.releaseAllCurrentLocks()
    //if (bot.releaseLockForBusiness(lockName)) {

    // Вариант чтобы случайно не удалить блокировку, если мы ее не создали в текущем скрипте
    // Например истекло время ожидания получения блокировки (maxWaitSec)
    // bot.releaseCurrentLockForBusiness / bot.releaseCurrentLockForBot
    if (bot.releaseCurrentLockForBusiness(lockName)) {
        bot.sendText(' Блокировка удалена' )
    } else {
        bot.sendText(' Ошибка удаления блокировки' )
    }
} else {
    bot.sendText(' Блокировка не найдена' )
}

// Удаляем блокировку если она получена именно в данном скрипте (через waitForBusinessLock /
waitForBotLock)
// Скрипт сам запоминает какие мы блокировки создавали в данном скрипте и удаляем ее только
если создали тут
//bot.releaseCurrentLock( lockName)

```

```
// На всякий случай
// Удаляем все блокировки полученные именно в данном скрипте
// Скрипт сам запоминает какие мы блокировки создавали в данном скрипте и удаляем их все,
которые мы тут создали
bot.releaseAllCurrentLocks()

bot.sendText(' Дополнительно, почистили все блокировки созданные в данном скрипте, на случай,
если что-то забыли...')
```

Использование методов планирования времени

Установить время задержки для выполнения функции можно двумя способами:

- Метод **setTimeout()**;
- Задерживающий декоратор **delay()**.

Они позволяют вызвать функцию один раз через определённый интервал времени.

setTimeout()

```
setTimeout($func, int $delay, $arg1, $arg2, ...)
```

Атрибуты:

Имя	Тип	Описание
func	function	Вызываемая функция
delay	int	Задержка перед запуском в миллисекундах. Значение по умолчанию – 0
arg1, arg2, ...	mixed	Аргументы, передаваемые в функцию

Пример:

```
function sayHi($phrase, $who) {  
    lead.setAttr('sayHi', $phrase + ', ' + $who);  
}  
  
setTimeout(sayHi, 1000, "Привет", "Джон");
```

или

```
setTimeout("lead.setAttr('sayHi', 'Привет!')", 1000);
```

Отмена таймера:

```
let timerId = setTimeout(...);  
clearTimeout(timerId);
```

delay()

```
delay($f, int $ms)
```

Атрибуты:

Имя	Тип	Описание
f	function	Вызываемая функция
ms	int	Задержка перед запуском в миллисекундах

Пример:

```
function sayHi(text) {  
    lead.setAttr('sayHi', text);  
}  
  
let f1000 = delay(sayHi, 1000);  
  
f1000("Привет!");
```

Использование методов работы с файлами

Ниже представлен JS скрипт на примере которого будет рассмотрена работа с файлами.

```
//
-----

// ПОДГОТОВКА КОМПОНЕНТА, ПЕРЕЧИСЛЕНИЕ ВСЕХ ПАРАМЕТРОВ
let imageUploader = {
  "params": {
    "max_file_size_mb": null,
    "min_image_width": null,
    "min_image_height": null,
    "prompt_message": null,
    "prompt_keyboard": null,
    "btn_cancel_caption": null,
    "btn_cancel_no_inline_label_code": null,
    "btn_ok_caption": null,
    "fallback_message": null,
    "success_message": null,
    "files_custom_table_name": null,
    "api_additional_params": null,
    "script_code_for_files_is_not_supported": null,

    // Параметры внутренней логики
    "is_first_immediate_call": false,
    "image_info": null,
    "new_image_url": null,
    "image_upload_result": null,
    "old_image_url": null,
    "telegram_file_id": null
  }
}
//
-----
```

```
-----

//
-----
-----

// ПОДГОТОВКА ЗНАЧЕНИЙ ПО УМОЛЧАНИЮ
// Максимальный размер файла в Мб
imageUploader.params.max_file_size_mb = 2


// Минимальная ширина изображения
// если 0 - то нет ограничения
// если включаем данную опцию то фото принимаем только с включенной галкой телеграмма "Сжать
изображением"
imageUploader.params.min_image_width = 0


// Минимальная высота изображения
// если 0 - то нет ограничения
// если включаем данную опцию то фото принимаем только с включенной галкой телеграмма "Сжать
изображением"
imageUploader.params.min_image_height = 0


// Сообщение-вопрос
imageUploader.params.promt_message = null // для вычисления используйте функцию
this.calcPromtMessage()


imageUploader.params.promt_keyboard = null // для вычисления используйте функцию
this.calcPromtKeyboard()


// Заголовки единственного пункта меню
// (одна кнопка но у нее меняется заголовок, в зависимости от того приняли мы хоть одно фото
или нет)
imageUploader.params.btn_cancel_caption = "Отмена"
// Код кнопки, если inline режим выключен
imageUploader.params.btn_cancel_no_inline_label_code = "1"
imageUploader.params.btn_ok_caption = "Готово"


// Некст отправляемый лиду, если входящий вебхук не содержит изображения
// Если нужно сложное вычисление - то модифицировать функцию this.calcFallbackMessage()
imageUploader.params.fallback_message = "<b>Изображение не распознано</b>" +
    "\r\n\r\n<i>Повторите попытку...</i>"
```



```

// Текст отправляемый в случае успешного сохранения фото
// для вычисления используйте функцию this.calcSuccessMessage()
imageUploader.params.success_message = null

//imageUploader.params.success_message = "<b>Изображение распознано</b>" +
//     "\r\n\r\nожидаем следующий файл" +
//     "\r\nЕсли надоело, нажмите " + "'" + this.getBtnOkCaption() + "'" + "..."

// Название кастомной таблицы куда сохраняем ссылки на фото
imageUploader.params.files_custom_table_name = "person_files"

// Дополнительные параметры отправляемые в Телеграм (что здесь укажем то и будет отправлено в
// POST запросе к Telegram)
imageUploader.params.api_additional_params = {"parse_mode": "HTML",
"disable_web_page_preview": true}

imageUploader.params.script_code_for_files_is_not_supported = "files_is_not_supported"
//
-----
-----

//
-----
-----

// ПОДГОТОВКА МЕТОДОВ ДЛЯ ДОСТУПА К ПАРАМЕТРАМ (Getters & Setters)
// Обязательно описываем методы для всех параметров, чтобы их можно было перекрывать извне
imageUploader.getMaxFileSizeMb = function() {
    return this.params.max_file_size_mb
}
imageUploader.setMaxFileSizeMb = function(maxFileSizeMb) {
    this.params.max_file_size_mb = maxFileSizeMb
    return this
}

imageUploader.getMinImageWidth = function() {
    return this.params.min_image_width
}
imageUploader.setMinImageWidth = function(minImageWidth) {
    this.params.min_image_width = minImageWidth
    return this
}

```

```
imageUploader.getMinImageHeight = function() {
    return this.params.min_image_height
}

imageUploader.setMinImageHeight = function(minImageHeight) {
    this.params.min_image_height = minImageHeight
    return this
}

imageUploader.getPromptMessage = function() {
    return this.params.prompt_message
}

imageUploader.setPromptMessage = function(promptMessage) {
    this.params.prompt_message = promptMessage
    return this
}

imageUploader.getPromptKeyboard = function() {
    return this.params.prompt_keyboard
}

imageUploader.setPromptKeyboard = function(promptKeyboard) {
    this.params.prompt_keyboard = promptKeyboard
    return this
}

imageUploader.getBtnCancelCaption = function() {
    return this.params.btn_cancel_caption
}

imageUploader.setBtnCancelCaption = function(btnCancelCaption) {
    this.params.btn_cancel_caption = btnCancelCaption
    return this
}

imageUploader.getBtnCancelNoInlineLabelCode = function() {
    return this.params.btn_cancel_no_inline_label_code
}

imageUploader.setBtnCancelNoInlineLabelCode = function(btnCancelNoInlineLabelCode) {
    this.params.btn_cancel_no_inline_label_code = btnCancelNoInlineLabelCode
    return this
}
```

```
imageUploader.getBtnOkCaption = function() {
    return this.params.btn_ok_caption
}

imageUploader.setBtnOkCaption = function(btnOkCaption) {
    this.params.btn_ok_caption = btnOkCaption
    return this
}

imageUploader.getFallbackMessage = function() {
    return this.params.fallback_message
}

imageUploader.setFallbackMessage = function(fallbackMessage) {
    this.params.fallback_message = fallbackMessage
    return this
}

imageUploader.getSuccessMessage = function() {
    return this.params.success_message
}

imageUploader.setSuccessMessage = function(successMessage) {
    this.params.success_message = successMessage
    return this
}

imageUploader.GetFilesCustomTableName = function() {
    return this.params.files_custom_table_name
}

imageUploader.setFilesCustomTableName = function(filesCustomTableName) {
    this.params.files_custom_table_name = filesCustomTableName
    return this
}

imageUploader.getApiAdditionalParams = function() {
    return this.params.api_additional_params
}

imageUploader.setApiAdditionalParams = function(apiAdditionalParams) {
    this.params.api_additional_params = apiAdditionalParams
    return this
}

imageUploader.getScriptCodeForFilesIsNotSupported = function() {
```

```
    return this.params.script_code_for_files_is_not_supported
}
imageUploader.setScriptCodeForFilesIsNotSupported = function(scriptCodeForFilesIsNotSupported)
{
    this.params.script_code_for_files_is_not_supported = scriptCodeForFilesIsNotSupported
    return this
}

imageUploader.getIsFirstImmediateCall = function() {
    this.params.is_first_immediate_call = isFirstImmediateCall // GLOBAL VARIABLE !!!

    return this.params.is_first_immediate_call
}
imageUploader.setIsFirstImmediateCall = function(isFirstImmediateCall) {
    this.params.is_first_immediate_call = isFirstImmediateCall
    return this
}

imageUploader.getImageInfo = function() {
    return this.params.image_info
}
imageUploader.setImageInfo = function(imageInfo) {
    this.params.image_info = imageInfo
    return this
}

imageUploader.getOldImageUrl = function() {
    return this.params.old_image_url
}
imageUploader.setOldImageUrl = function(oldImageUrl) {
    this.params.old_image_url = oldImageUrl
    return this
}

imageUploader.getNewImageUrl = function() {
    return this.params.new_image_url
}
imageUploader.setNewImageUrl = function(newImageUrl) {
    this.params.new_image_url = newImageUrl
    return this
}
}
```

```

imageUploader.getImageUploadResult = function() {
    return this.params.image_upload_result
}

imageUploader.setImageUploadResult = function(imageUploadResul) {
    this.params.image_upload_result = imageUploadResul
    return this
}

imageUploader.getTelegramFileId = function() {
    return this.params.telegram_file_id
}

imageUploader.setTelegramFileId = function(telegramFileId) {
    this.params.telegram_file_id = telegramFileId
    return this
}

//
-----

//
-----

// ПОДГОТОВКА МЕТОДОВ ДЛЯ БИЗНЕС ЛОГИКИ

imageUploader.calcPromtMessage = function() {
    // Todo: this.params.promt_message -> можно использовать для задания шаблона сообщения и
    добавить сюда макроподстановку

    let maxFileSizeMb = this.getMaxFileSizeMb()
    let minImageWidth = this.getMinImageWidth()
    let minImageHeight = this.getMinImageHeight()

    let msg = "<b>Отправьте ваше фото</b>" +
        (maxFileSizeMb > 0 ? ("\r\n\r\n<i>Размер фото: не более " + maxFileSizeMb + " Мб</i>") :
        "") +
        (minImageWidth > 0 ? ("\r\n<i>Ширина: не менее " + minImageWidth + " px</i>") : "") +
        (minImageHeight > 0 ? ("\r\n<i>Высота: не менее " + minImageHeight + " px</i>") : "")

```

```

    return msg
}

imageUploader.calcSuccessMessage = function() {
    return "<b>Изображение распознано</b>" +
        "\r\n\r\nОжидаем следующий файл" +
        "\r\nЕсли надоело, нажмите " + "'" + this.getBtnOkCaption() + "'" + "...
}

// Вычисление сообщения о невалидных данных
imageUploader.calcFallbackMessage = function() {
    return this.getFallbackMessage()
}

// Вычисление заголовка единственного пункта меню
imageUploader.calcBtnCancelCaption = function() {
    if (!this.isAnyImagesReceived()) {
        return this.getBtnCancelCaption()
    } else {
        return this.getBtnOkCaption()
    }
}

// Вычисление кнопок меню
imageUploader.calcPromtKeyboard = function() {
    return [
        [
            {
                "text": this.calcBtnCancelCaption(),
                "callback_data": "btn_cancel",
                "text_button_label": this.getBtnCancelNoInlineLabelCode()
            },
        ]
    ]
}

imageUploader.calcInvalidFileSizePromtMessage = function(yourFileSizeMb) {
    return "Файл не соответствуют требованию: "+
        "\r\nРазмер файла не более " + this.getMaxFileSizeMb() + " Мб." +
        "\r\nРазмер вашего файла: " + yourFileSizeMb +
        "\r\n\r\n" +

```

```

        this.calcPromtMessage()
    }

    imageUploader.calcInvalidWidthOrHeightPromtMessage = function(yourWidth, yourHeight) {
        return "Файл не соответствуют требованиям ширины или высоты, или изображение не распознано"
    +
        "\r\nШирина вашего фото: " + yourWidth +
        "\r\nВысота вашего фото: " + yourHeight +
        "\r\n\r\n" +
        '<b>При отправке фото через Telegram включите галку "Сжать изображение"</b>' +
        "\r\n\r\n" +
        this.calcPromtMessage()
    }

    // Получить атрибут "ID последнего сообщения отправленного в Telegram"
    imageUploader.getLastTelegramMessageId = function() {
        return parseInt(lead.getAttr("last_telegram_message_id"))
    }

    // Есть ли сохраненный атрибут "ID последнего сообщения отправленного в Telegram"
    imageUploader.hasLastTelegramMessageId = function() {
        let messageId = this.getLastTelegramMessageId()
        return (!isNaN(messageId) && messageId > 0)
    }

    // Сохранить атрибут "ID последнего сообщения отправленного в Telegram"
    imageUploader.saveLastTelegramMessageId = function() {
        lead.setAttr("last_telegram_message_id", bot.getTelegramLastMessageId())
    }

    // Очистить атрибут "ID последнего сообщения отправленного в Telegram"
    imageUploader.clearLastTelegramMessageId = function() {
        lead.setAttr("last_telegram_message_id", null)
    }

    // Функция для скрытия кнопок предыдущего меню
    imageUploader.removeInlineKeyboard = function() {
        if (this.hasLastTelegramMessageId()) {
            bot.removeTelegramInlineKeyboard(lead.getData("identification"),
            this.getLastTelegramMessageId())
        }
    }

```

```

}

// Функция для удаления предыдущего сообщения
imageUploader.deletePrevMessage = function() {
  if (this.hasLastTelegramMessageId()) {
    this.removeInlineKeyboard()

    // Можно это делать в окне в 24 часа
    bot.sendMessage("---", null, null, {"endpoint": "deleteMessage", "message_id":
this.getLastTelegramMessageId()})

    this.clearLastTelegramMessageId()
  }
}

// Функция для определения получили ли мы хоть одно валидное фото в данной команде
imageUploader.isAnyImagesReceived = function() {
  let res = parseInt(lead.getAttr("is_any_image_received"))
  res = !isNaN(res) && res > 0
  return res
}

// Функция для сохранения состояния - получили ли мы хоть одно валидное фото в данной команде
imageUploader.setIsAnyImagesReceived = function(value) {
  lead.setAttr("is_any_image_received", value)
}

// Функция для отправки вопроса с меню
imageUploader.sendPromtMessageWithMenu = function(argPromtMessage, argPromtKeyboard,
argAttachments){
  // Скрываем кнопки предыдущего меню
  this.removeInlineKeyboard()

  if (typeof argPromtMessage == "undefined") {
    argPromtMessage = this.calcPromtMessage()
  }

  if (typeof argPromtKeyboard == "undefined") {
    argPromtKeyboard = this.calcPromtKeyboard()
  }
}

```



```

if (typeof argAttachments == "undefined") {
    argAttachments = null
}

// Отправляем кнопки с текстом вопроса
//bot.sendButtons(
//  argPromtMessage,
//  argPromtKeyboard,
//  this.getApiAdditionalParams()
//)

// Отправляем сообщение с кнопками
bot.sendMessage(
    argPromtMessage,
    argPromtKeyboard,
    argAttachments,
    this.getApiAdditionalParams()
)

// Запоминаем ID последнего сообщения
this.saveLastTelegramMessageId()
}

// Сохранение файла в кастомную таблицу
imageUploader.saveFileToCustomTable = function(data) {
    let tableName = this.getFilesCustomTableName()

    if (tableName === null || tableName === "") {
        return
    }

    table.createItem(tableName, data)
}

// Создаем персону для лида, если ее еще нет
imageUploader.checkPersonForCurrentLeadAndGetCommandResult = function() {
    if (!lead.getPersonId()) {
        lead.createPersonForCurrentLead({"comment": null})
    }

    if (!lead.getPersonId()) {

```

```

    return {
        "break": true,
        "run_script_by_code": this.getScriptCodeForFilesIsNotSupported()
    }
}

return null
}

// Если канал лида - не телеграм, то перекидываем в другой скрипт, сообщающий о том что
команда запроса файла не поддерживается
imageUploader.checkChannelAndGetCommandResult = function() {
    if (lead.getChannelCode() !== "telegram") {
        return {
            "break": true,
            "run_script_by_code": this.getScriptCodeForFilesIsNotSupported()
        }
    }

    return null
}

// Для ограничения максимального размера файла
imageUploader.validateFileSizeAndGetCommandResult = function(imagePayload) {
    let maxFileSizeMb = this.getMaxFileSizeMb()
    let isIncorrectFileSize = false
    if (maxFileSizeMb > 0) {
        if (!imagePayload["file_size"] || imagePayload["file_size"] > maxFileSizeMb*1024*1024)
        {
            isIncorrectFileSize = true
        }
    }

    if (isIncorrectFileSize) {
        // Превышен максимальный размер файла

        let yourFileSizeMb = "не определен"
        if (imagePayload["file_size"]) {
            // Конвертируем байты в Мб
            yourFileSizeMb = imagePayload["file_size"] / 1024 / 1024

```

```

        // Округляем до второго знака
        yourFileSizeMb = Math.round(yourFileSizeMb * 100) / 100

        yourFileSizeMb = yourFileSizeMb + " Мб"
    }

    // Повторяем вопрос и меню
    this.sendPromptMessageWithMenu( this.calcInvalidFileSizePromptMessage( yourFileSizeMb) )

    // Выходим из Callback (ждем пока пришлют другой файл)
    return {
        "result": false,
    }
}

return null
}

// Для ограничения ширины и высоты изображения
//  работает, только если фото отправлено с включенной галкой сжатия
imageUploader.validateFileWidthAndHeightAndGetCommandResult = function(imagePayload) {
    let minImageWidth = this.getMinImageWidth()
    let minImageHeight = this.getMinImageHeight()

    let isIncorrectWidth = (minImageWidth > 0 && (!imagePayload['width'] ||
imagePayload['width'] < minImageWidth))
    let isIncorrectHeight = (minImageHeight > 0 && (!imagePayload['height'] ||
imagePayload['height'] < minImageHeight))

    if (isIncorrectWidth || isIncorrectHeight) {
        // Файл не соответствуют требованиям ширины или высоты

        let yourWidth = "Не определено"
        if (imagePayload['width']) {
            yourWidth = imagePayload['width'] * 1
        }

        let yourHeight = "Не определено"
        if (imagePayload['width']) {
            yourHeight = imagePayload['height'] * 1
        }
    }
}

```

```

    // Повторяем вопрос и меню
    this.sendPromtMessageWithMenu( this.calcInvalidWidthOrHeightPromtMessage( yourWidth,
yourHeight))

    // Выходим из Callback (ждем пока пришлют другой файл)
    return {
        "result": false,
    }
}

return null
}

// Загружаем файл на CDN и получаем новый URL, если возникла ошибка, выводим ее, обновляем
меню и выходим из Callback
imageUploader.uploadFileToCdnAndGetCommandResult = function(oldImageUrl) {
    this.setNewImageUrl(null)
    this.setImageUploadResult(null)

    let uploadResult = bot.uploadFileToCdnAndGetNewUrl( oldImageUrl)
    this.setImageUploadResult(uploadResult)

    let newImageUrl = null
    if (uploadResult.result && !uploadResult.error) {
        newImageUrl = uploadResult.url
        this.setNewImageUrl(newImageUrl)
    } else {
        // Если возникла ошибка, выводим ее и далее по куду - выходим из Callback
        if (uploadResult.error_message) {
            bot.sendMessage(uploadResult.error_message)
        }

        // Повторяем вопрос и меню
        this.sendPromtMessageWithMenu()

        // Выходим из Callback (ждем пока пришлют другой файл)
        return {
            "result": false,
        }
    }
}

```

```

    return null
}

// Для входящего текста: если написали "отмена"/"пропустить"
// Или для кнопок: нажали на кнопку "отмена" или написали 1, для режима работы без inline
кнопок (inline кнопки отключаются в настройках канала)
imageUploader.checkIncomingMessageForNoImagesAndGetResult = function() {
    if (["отмена", "пропустить", "btn_cancel",
"1"].includes(bot.getIncomingMessage().toLowerCase())) {
        // Удаляем предыдущее меню, если оно есть
        this.removeInlineKeyboard()

        // Для логики с удалением предыдущего сообщения
        // для реализации учесть все цепочки, т.е. можно сделать полноценный графический компонент
как слайдер, чтобы вся обработка была в одном сообщении
        //deletePrevMessage()

        // Выходим из команды и прерываем заикливание
        return {
            "break": true,
        }
    }
}

return null
}

// Сохраняем ссылку на файл в таблицу
// Уведомляем клиента о том что фото загружено
// Пересылаем загруженно фото Лиду
// Повторяем вопрос и меню
// Выходим из callback, но не прерываем команду, ожидаем следующий файл
// ВЫНЕСЕНО В ФУНКЦИЮ - ЧТОБЫ МОЖНО БЫЛО ПЕРЕКРЫВАТЬ ЕЕ ИЗВНЕ
imageUploader.runOnSuccessLogicAndGetResult = function() {
    let telegramFileId = this.getTelegramFileId()
    let newImageUrl = this.getNewImageUrl()

    // Сохраняем ссылку на файл в таблицу
    this.saveFileToCustomTable({
        "person_id": lead.getPersonId(),
        "lead_id": leadId,

```

```

        "telegram_file_id": telegramFileId,
        "type": "image",
        "file_name": null,
        "url": newImageUrl,
        "size": null,
        "mime_type": null,
    })

    // Уведомляем клиента о том что фото загружено
    // Пересылаем загруженно фото Лиду
    // Todo: Вынести в параметры компонента - но с учетом макроса для newImageUrl
    let msg = "<b>Ваше фото успешно загружено</b>\r\n<a href='" + newImageUrl + "'>Ссылка на
загруженное фото</a>"

    // Вариант: Для отправки отдельно фото и отдельно сообщения с меню
    bot.sendMessage(msg, null, [{"type": "image", "url": newImageUrl}],
this.getApiAdditionalParams())
    // Повторяем вопрос и меню
    this.sendPromtMessageWithMenu( this.calcSuccessMessage())

    // Вариант: Для отправки полученного фото и вопроса одним сообщением
    // Повторяем вопрос и меню
    //sendPromtMessageWithMenu(successMessage + "\r\n\r\n" + msg, null, [{"type": "image",
"url": newImageUrl}])

    // Выходим из callback, но не прерываем команду, ожидаем следующий файл
    return {
        "result": false,
    }
}

// Повторяем вопрос и меню
// Выходим из JS, но не прерываем замыкание, ожидаем следующий файл
// ВЫНЕСЕНО В ФУНКЦИЮ - ЧТОБЫ МОЖНО БЫЛО ПЕРЕКРЫВАТЬ ЕЕ ИЗВНЕ
imageUploader.runOnNoImagesLogicAndGetResult = function() {
    // Повторяем вопрос и меню
    this.sendPromtMessageWithMenu( this.calcFallbackMessage())

    // Выходим из JS, но не прерываем замыкание, ожидаем следующий файл
    return {
        "result": false,
    }
}

```

```

}

//
-----
----

// Метод для вычисления результата
// Результат будет передан в команду "Выполнить JavaScript CallBack" с замыканием состояния
imageUploader.getComponentResult = function() {
    let checkPersonResult = this.checkPersonForCurrentLeadAndGetCommandResult()
    if (checkPersonResult !== null) {
        return checkPersonResult
    }

    let checkChannelResult = this.checkChannelAndGetCommandResult()
    if (checkChannelResult !== null) {
        return checkChannelResult
    }

    //
    -----
    ----

    // Инициализация компонента
    if (this.getIsFirstImmediateCall()) {
        // !!! ИНИЦИАЛИЗАЦИЯ !!!

        // Запоминаем что фото еще не получено, чтобы определять, какое меню выводить
        this.setIsAnyImagesReceived(0)

        // Сбрасываем ID последнего сообщения
        this.clearLastTelegramMessageId()

        // Отправляем вопрос и меню
        this.sendPromtMessageWithMenu()

        // Выходим из JS, в параметрах ничего не возвращаем, тк это инициализация
        return {}
    }
    //
    -----
    ----

```

```
//
-----

// Получаем фото из входящего вебхука
let images = bot.getImages()
//
-----

//

-----

// ЛОГИКА КОМПОНЕНТА
if (images.length > 0) {
    // Если прислали фото, обрабатываем его, иначе см. блок else

    // Получаем инфу о фото
    let imageInfo = images[0]
    let imagePayload = imageInfo["payload"]
    let oldImageUrl = imageInfo["url"]
    let telegramFileId = imageInfo["file_id"]

    this.setImageInfo(imageInfo)
    this.setOldImageUrl(oldImageUrl)
    this.setTelegramFileId(telegramFileId)

    // Для ограничения максимального размера файла
    let validateFileSizeResult = this.validateFileSizeAndGetCommandResult(imagePayload)
    if (validateFileSizeResult !== null) {
        return validateFileSizeResult
    }

    // Для ограничения ширины и высоты изображения
    //  работает, только если фото отправлено с включенной галкой сжатия
    let validateFileWidthAndHeightResult =
(this.validateFileWidthAndHeightAndGetCommandResult(imagePayload))
    if (validateFileWidthAndHeightResult !== null) {
        return validateFileWidthAndHeightResult
    }
}
```



```

    // Загружаем файл на CDN и получаем новый URL, если возникла ошибка, выводим ее, обновляем
    меню и выходим из Callback
    let uploadFileToCdnResult = (this.uploadFileToCdnAndGetCommandResult(oldImageUrl))
    if (uploadFileToCdnResult !== null) {
        return uploadFileToCdnResult
    }

    // Запоминаем, что корректное фото получено
    this.setIsAnyImagesReceived(1)

    // Сохраняем ссылку на файл в таблицу
    // Уведомляем клиента о том что фото загружено
    // Пересылаем загруженно фото Лиду
    // Повторяем вопрос и меню
    // Выходим из callback, но не прерываем команду, ожидаем следующий файл
    // ВЫНЕСЕНО В ФУНКЦИЮ - ЧТОБЫ МОЖНО БЫЛО ПЕРЕКРЫВАТЬ ЕЕ ИЗВНЕ
    return this.runOnSuccessLogicAndGetResult()
} else {
    // Если прислали НЕ фото

    let checkIncomingMessageResult = this.checkIncomingMessageForNoImagesAndGetResult()
    if (checkIncomingMessageResult !== null) {
        return checkIncomingMessageResult
    }

    // Повторяем вопрос и меню
    // Выходим из JS, но не прерываем замыкание, ожидаем следующий файл
    // ВЫНЕСЕНО В ФУНКЦИЮ - ЧТОБЫ МОЖНО БЫЛО ПЕРЕКРЫВАТЬ ЕЕ ИЗВНЕ
    return this.runOnNoImagesLogicAndGetResult()
}
}

//Todo: Учесть мультиязычность!

module.exports = {
    imageUploader
}

```

```

const {unset, unsetButton} = require("Common.Utills.Arrays")

//
-----
-----// ПОДГОТОВКА КОМПОНЕНТА, ПЕРЕЧИСЛЕНИЕ ВСЕХ ПАРАМЕТРОВlet imageSlider = {
  "params": {
    "script_code_for_files_is_not_supported": null,
    "script_code_for_person_files_is_empty": null,
    "script_code_for_main_menu": null,
    "files_custom_table_name": null,

    // Параметры внутренней логики
    "is_first_immediate_call": false,
  }
}
//
-----
-----//

-----// ПОДГОТОВКА ЗНАЧЕНИЙ ПО УМОЛЧАНИЮ
imageSlider.params.script_code_for_files_is_not_supported = "files_is_not_supported"
imageSlider.params.script_code_for_person_files_is_empty = "main_menu"
imageSlider.params.script_code_for_main_menu = "main_menu"// Название кастомной таблицы откуда
грузим фото
imageSlider.params.files_custom_table_name = "person_files"//
-----
-----//

-----// ПОДГОТОВКА МЕТОДОВ ДЛЯ ДОСТУПА К ПАРАМЕТРАМ (Getters & Setters)// Обязательно
описываем методы для всех параметров, чтобы их можно было перекрывать извне
imageSlider.getScriptCodeForFilesIsNotSupported = function() {
  return this.params.script_code_for_files_is_not_supported
}
imageSlider.setScriptCodeForFilesIsNotSupported = function(scriptCodeForFilesIsNotSupported) {
  this.params.script_code_for_files_is_not_supported = scriptCodeForFilesIsNotSupported
  return this
}

imageSlider.getScriptCodeForPersonFilesIsEmpty = function() {
  return this.params.script_code_for_person_files_is_empty

```

```

}

imageSlider.setScriptCodeForPersonFilesIsEmpty = function( scriptCodeForPersonFilesIsEmpty) {
    this.params.script_code_for_person_files_is_empty = scriptCodeForPersonFilesIsEmpty
    return this
}

imageSlider.getScriptCodeForMainMenu = function() {
    return this.params.script_code_for_main_menu
}

imageSlider.setScriptCodeForMainMenu = function( scriptCodeForMainMenu) {
    this.params.script_code_for_main_menu = scriptCodeForMainMenu
    return this
}

imageSlider.GetFilesCustomTableName = function() {
    return this.params.files_custom_table_name
}

imageSlider.setFilesCustomTableName = function( filesCustomTableName) {
    this.params.files_custom_table_name = filesCustomTableName
    return this
}

imageSlider.getIsFirstImmediateCall = function() {
    this.params.is_first_immediate_call = isFirstImmediateCall // GLOBAL VARIABLE !!!

    return this.params.is_first_immediate_call
}

imageSlider.setIsFirstImmediateCall = function( isFirstImmediateCall) {
    this.params.is_first_immediate_call = isFirstImmediateCall
    return this
}

//
-----
-----//
-----
-----// Методы компонента// Текущая позиция слайдера
imageSlider.getSliderPos = function() {
    return lead.getAttr("slider_pos")*1
}

// Установить текущую позицию слайдера

```

```
imageSlider.setSliderPos = function(sliderPos) {
    lead.setAttr("slider_pos", sliderPos)
}

// Получить атрибут "ID последнего сообщения отправленного в Telegram"
imageSlider.getLastTelegramMessageId = function() {
    return parseInt(lead.getAttr("last_telegram_message_id"))
}

// Есть ли сохраненный атрибут "ID последнего сообщения отправленного в Telegram"
imageSlider.hasLastTelegramMessageId = function() {
    let messageId = this.getLastTelegramMessageId()
    return (!isNaN(messageId) && messageId > 0)
}

// Сохранить атрибут "ID последнего сообщения отправленного в Telegram"
imageSlider.saveLastTelegramMessageId = function() {
    lead.setAttr("last_telegram_message_id", bot.getTelegramLastMessageId())
}

// Очистить атрибут "ID последнего сообщения отправленного в Telegram"
imageSlider.clearLastTelegramMessageId = function() {
    lead.setAttr("last_telegram_message_id", null)
}

// Функция для скрытия кнопок предыдущего меню
imageSlider.removeInlineKeyboard = function() {
    if (this.hasLastTelegramMessageId()) {
        bot.removeTelegramInlineKeyboard(lead.getData("identification"),
this.getLastTelegramMessageId())
    }
}

// Функция для удаления предыдущего сообщения (слайдера)
imageSlider.deletePrevMessage = function() {
    if (this.hasLastTelegramMessageId()) {
        this.removeInlineKeyboard()

        // Можно это делать в окне в 24 часа
        bot.sendMessage("---", null, null, {"endpoint": "deleteMessage", "message_id":
this.getLastTelegramMessageId()})
    }
}
```

```

        this.clearLastTelegramMessageId()
    }
}

// Останавливаем анимацию ожидания на кнопке
imageSlider.answerCallbackQuery = function() {
    let callbackQueryId = null
    let webhookPayload = bot.getWebhookPayload()
    if (webhookPayload && webhookPayload['callback_query'] &&
webhookPayload['callback_query']['id']) {
        callbackQueryId = webhookPayload['callback_query']['id']
        if (typeof callbackQueryId == "string" && callbackQueryId.length > 0) {
            bot.sendPayload('answerCallbackQuery', {
                "callback_query_id": callbackQueryId,
            })
        }
    }
}

// Создаем персону для лида, если ее еще нет
imageSlider.checkPersonForCurrentLeadAndGetCommandResult = function() {
    if (!lead.getPersonId()) {
        lead.createPersonForCurrentLead({"comment": null})
    }

    if (!lead.getPersonId()) {
        return {
            "break": true,
            "run_script_by_code": this.getScriptCodeForFilesIsNotSupported()
        }
    }

    return null
}

// Если канал лида - не телеграм, то перекидываем в другой скрипт, сообщающий о том что
команда запроса файла не поддерживается
imageSlider.checkChannelAndGetCommandResult = function() {
    if (lead.getChannelCode() !== "telegram") {
        return {

```

```

        "break": true,
        "run_script_by_code": this.getScriptCodeForFilesIsNotSupported()
    }
}

return null
}
//
-----//
-----// Метод для вычисления результата// Результат будет передан в команду "Выполнить
JavaScript CallBack" с замыканием состояния
imageSlider.getComponentResult = function() {
    // Создаем персону для лида, если ее еще нет
    let checkPersonResult = this.checkPersonForCurrentLeadAndGetCommandResult()
    if (checkPersonResult !== null) {
        return checkPersonResult
    }

    // Если канал лида - не телеграм, то перекидываем в другой скрипт, сообщающий о том что
    команда запроса файла не поддерживается
    let checkChannelResult = this.checkChannelAndGetCommandResult()
    if (checkChannelResult !== null) {
        return checkChannelResult
    }

    //
    -----
    // Инициализация компонента
    if (this.getIsFirstImmediateCall()) {
        // !!! ИНИЦИАЛИЗАЦИЯ !!!

        // Запоминаем что фото еще не получено, чтобы определять, какое меню выводить
        this.setSliderPos(0)

        // Сбрасываем ID последнего сообщения
        this.clearLastTelegramMessageId()
    }
    //

```

```
-----  
-----  
  
//  
-----  
-----  
  
// ЛОГИКА КОМПОНЕНТА  
//  
-----  
-----  
  
//  
-----  
-----  
  
// Определяем, есть ли фото, если нет, то перекидываем в Главное меню  
let wc = [{"person_id", lead.getPersonId()}, {"type", "image"}]  
  
let imagesCount = table.count(  
  this.GetFilesCustomTableName(), // tableName  
  wc  
)  
  
if (!imagesCount) {  
  bot.sendText("У вас нет загруженных фото! Сначала загрузите их!")  
  
  return {  
    "break": true,  
    "run_script_by_code": this.getScriptCodeForPersonFilesIsEmpty()  
  }  
}  
//  
-----  
-----  
  
//  
-----  
-----  
  
// Подготовка переменных  
let isAnyOtherMessageReceived = false  
let prevSliderPos = this.getSliderPos()
```

```

let hasPrevPhoto = prevSliderPos > 0
let hasNextPhoto = prevSliderPos < (imagesCount-1)

// Для отладки:
//bot.sendText(hasPrevPhoto + ":" + hasNextPhoto)
//bot.sendText(imagesCount + ":" + prevSliderPos)
//
-----
-----

//
-----
-----

// Обработка нажатых кнопок или входящего текста
if (["btn_static_main_menu", "0"].includes(bot.getIncomingMessage().toLowerCase())) {
    // Удаляем предыдущее меню, если оно есть
    //this.removeInlineKeyboard()

    // Удаляем слайден, если он есть
    this.deletePrevMessage()

    // Выходим из команды
    return {
        "break": true,
        "run_script_by_code": this.getScriptCodeForMainMenu(),
    }
} else if (["btn_static_prev_photo", "1"].includes(bot.getIncomingMessage().toLowerCase()))
{
    // Останавливаем анимацию ожидания на кнопке
    this.answerCallbackQuery()

    let _sliderPos = this.getSliderPos()
    if (hasPrevPhoto) {
        _sliderPos--
        hasPrevPhoto = _sliderPos > 0
        hasNextPhoto = _sliderPos < (imagesCount-1)
    } else {
        // Фото закончились
    }
    this.setSliderPos(_sliderPos)
} else if (["btn_static_next_photo", "2"].includes(bot.getIncomingMessage().toLowerCase()))

```



```

{
    // Останавливаем анимацию ожидания на кнопке
    this.answerCallbackQuery()

    let _sliderPos = this.getSliderPos()
    if (hasNextPhoto) {
        _sliderPos++
        hasPrevPhoto = _sliderPos > 0
        hasNextPhoto = _sliderPos < (imagesCount-1)
    } else {
        // Фото закончились
    }
    this.setSliderPos(_sliderPos)
} else {
    // Если получаем любое текстовое сообщение то лучше посылать фото не в том же сообщении, а
    в новом,
    // чтобы слайдер не уходил вверх в переписке
    //if (bot.getIncomingMessage() !== "") {
        isAnyOtherMessageReceived = true
    //}
}

if (isAnyOtherMessageReceived) {
    this.deletePrevMessage()
}
//
-----
-----

//
-----
-----

// Выполнение запросов к кастомной таблице
let sliderPos = this.getSliderPos()

// Для отладки
//bot.sendText(prevSliderPos + ":" + sliderPos)

// Для отладки
//bot.sendText(imagesCount + ":" + sliderPos)

```

```

let limit = 1
let offset = sliderPos
let images
let image

//let loadImage = function () {
  imagesCount = table.count(
    this.GetFilesCustomTableName(), // tableName
    wc
  )

  images = table.find(
    this.GetFilesCustomTableName(), // tableName
    null, // columns
    wc,
    null, // orderBy
    limit,
    offset
  )

  image = images[0]
//}

//loadImage()

if (!images.length) {
  return {
    "break": true,
    "run_script_by_code": this.getScriptCodeForPersonFilesIsEmpty()
  }
}
//

```

//

```

// Отображение слайдера
// ps: "_static_" - вхождение такого текста означает что платформа Метабот
// не будет автоматически удалять inlineKeyboard после приема вебхука с

```

нажатием на кнопку

```
let inlineKeyboard = [
  [
    {"text": "<", "callback_data": "btn_static_prev_photo", "text_button_label": "1"},
    {"text": ">", "callback_data": "btn_static_next_photo", "text_button_label": "2"},
  ],
  [
    {"text": "Главное меню", "callback_data": "btn_static_main_menu", "text_button_label":
"0"},
  ]
]

// Удаление кнопок меню, если действие недоступно
if (!hasPrevPhoto) {
  unsetButton(inlineKeyboard, "btn_static_prev_photo")
}
if (!hasNextPhoto) {
  unsetButton(inlineKeyboard, "btn_static_next_photo")
}

// Для отладки
//bot.sendText( this.hasLastTelegramMessageId() + ":" + this.getLastTelegramMessageId())

if ( this.hasLastTelegramMessageId()) {
  if (prevSliderPos !== sliderPos) {
    newMedia = {"type": "photo", "media": image.url, "caption": "Фото #" + image.id}
    bot.sendMessage("---", inlineKeyboard, null, {"endpoint": "editMessageMedia",
"message_id": this.getLastTelegramMessageId(), "media": newMedia})
    //this.saveLastTelegramMessageId() // Тут этого делать не нужно! Тк тут мы обновляем
существующее сообщение !
  }
} else {
  bot.sendMessage("Фото #" + image.id, inlineKeyboard, [{"type": "image", "url":
image.url}])
  this.saveLastTelegramMessageId()
}

return {
  "result": false,
}
//
```

```
-----  
-----  
}  
  
//Todo: Учесть мультиязычность! module.exports = {  
  imageSlider  
}
```