

Создание HTML формы.

Инструкция для разработчика веб-формы.

Актуальный исходный код веб-формы реализующий все три вида форм смотрите по ссылке: [go-to-the-mars.html](https://t.me/metabot_test_form_bot)

Пример работы веб-формы приведенной выше смотрите в Telegram боте https://t.me/metabot_test_form_bot

Исходный код примера веб-формы

```
<!DOCTYPE html>
<html lang="ru">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
  <title>Заявка для полета на Марс</title>

  <!-- Подключаем Telegram Web App -->
  <script src="https://telegram.org/js/telegram-web-app.js"></script>

  <!-- Подключаем JQuery и JQ suggestions для dadata -->
  <!-- PS: JQuery подключать не обязательно, у вас могут быть свои библиотеки для работы с
формой и отправки ajax запросов -->
  <script src="https://code.jquery.com/jquery-3.6.0.min.js"
    integrity="sha256-/xUj+30JU5yExlq6GSYGS7k7tPXikynS7ogEvDej/m4="
crossorigin="anonymous"></script>
  <script
src="https://app.metabot24.com/lib/jquery.suggestions/js/jquery.suggestions.min.js"></scri
pt>
  <link href="https://app.metabot24.com/lib/jquery.suggestions/css/suggestions.min.css"
rel="stylesheet" />
```

```

<script language="JavaScript">
    let botId = ID_ВАШЕГО_БОТА
    let botToken = 'ТОКЕН_ВАШЕГО_БОТА'
    let dadataToken = 'ТОКЕН_DADATA'

    // URL для POST запроса на который будет отправлять данные с формы
    // В качестве примера мы отправляем напрямую в бота
    // Но в вашем production боте вы должны отправлять на данные на бэк
    // А с бэка уже пересылать в бота, чтобы, например "не светить" токен бота в коде
html-формы
    let sendUrl = '/api/v1/bots/' + botId + '/call/submit-form'

    let sendBody = {}
    let sendHeaders = {}
    let mode = null
    let tgWebApp = null

    $(function () {
        /* Определяем режим работы формы

            Режим передается в query url (GET параметр mode=)

            Режимы текущей html:
            - '' - пустая строка или null, когда в чат-боте ссылка на форму приходит в виде
ссылки
                это универсальный вариант который будет работать в любом мессенджере
            - 'tg_inline' - для Telegram чат-бота, когда ссылка на форму приходит в виде
inline-кнопки
            - 'tg_keyboard' - для Telegram чат-бота, когда ссылка на форму приходит в виде
keyboard-кнопки

            Для вашего бота может быть достаточно одного из режимов
            В качестве примера просто приведены 3 варианта, чтобы вы могли выбрать
подходящий и понять разницу
        */
        if (typeof (urlParams["mode"]) === 'string') {
            mode = urlParams["mode"]
        }
    })

```

```

// Переменная для доступа к Telegram Web App, чтобы не писать везде
window.Telegram.WebApp

if (mode === 'tg_inline' || mode === 'tg_keyboard') {
  if (window.Telegram && window.Telegram.WebApp) {
    tgWebApp = window.Telegram.WebApp
  }
}

// Инициализация Dadata для автокомплита поля с адресом
$(".dadata-suggestion").suggestions({
  token: dadataToken,
  type: "ADDRESS"
})

// Получаем хэш-код лида из request url (GET параметр q=)
$('#q').val(urlParams["q"])

// Событие нажатия на кнопку "Отправить данные"
$(document).on('click', '#submit-mars-form', (e) => {
  let form = $('#form-mars')
  if (!form[0].checkValidity()) {
    form[0].reportValidity()
    return
  }

  let formData = getFormData(form)

  formData['tg_query_id'] = ''
  formData['mode'] = mode

  if (mode === 'tg_inline') {
    if (tgWebApp && tgWebApp.initDataUnsafe && tgWebApp.initDataUnsafe.query_id) {
      formData['tg_query_id'] = tgWebApp.initDataUnsafe.query_id
    }
  }

  if (tgWebApp) {
    // https://core.telegram.org/bots/webapps#initializing-web-apps
    tgWebApp.expand() // необязательно
  }
}

```

```

    tgWebApp.ready() // обязательно
}

// Для универсального режима или режима tg_inline
if (mode !== 'tg_keyboard') {

    // Отправляем данные внутри script_request_params и указываем токен и др
заголовки для Metabot API

    // Но в вашем production, здесь вы должны просто отправить данные на ваш бэк, а
с бэка уже в Metabot API

    // отправлять необязательно через JSON API, можно делать это просто через ACTION
формы SUBMIT кнопку на форме

    //

    // Если вы реализуете универсальный режим и выполняете отправку через action
форма(сабмит без REST API),

    // то ваша форма разрывается на два шага
    // - заполнение формы клиентом
    // - получаем данные на бэке
    //      и после приема данных рендерим опять HTML в коде которого размещаем
JavaScript который закроет страницу (вызовет метод closeForm())

    // Поэтому проще данные на бэк отправить по REST API и сразу же закрыть форму
    // Именно такой вариант и реализован в данной HTML форме
    sendBody = {"script_request_params": formData} //form.serializeArray()
    sendHeaders = {
        "Authorization": "Bearer " + botToken,
        'Content-Type': 'application/json',
        'Accept': 'application/json',
    }

    // Отправка запроса с помощью библиотеки JQuery
    sendJQueryRequest('POST', sendUrl, sendBody, sendHeaders, function (response,
isError, jqXHR, textStatus, errorThrown) {
        if (!isError) {
            // Запрос завершён. Здесь можно обрабатывать результат.
            //console.log(response)

            closeForm()
        } else {
            // Произошла ошибка

```

```

        alert("Ошибка обработки API запроса")

        console.log(jqXHR)
    }
})

// Отправка запроса без дополнительных библиотек (с помощью XHR)
// Могут быть проблемы с отправкой, возможно нужна корректировка кода под ваш
бэкенд вашего сайта

/*sendXhrRequest('POST', sendUrl, sendBody, sendHeaders, function(xhr) {

//https://developer.mozilla.org/ru/docs/Web/API/XMLHttpRequest/send%D0%BF%D1%80%D0%B8%D0%
BC%D0%B5%D1%80_get
    if(xhr.readyState == XMLHttpRequest.DONE && xhr.status == 200) {
        // Запрос завершён. Здесь можно обрабатывать результат.
        console.log(xhr)
        closeForm()
    } else {
        // Произошла ошибка
        alert("Ошибка обработки API запроса")
        console.log(xhr)
    }
})*/*
} else {
    // Если это режим tg_keyboard

    // Внимание! Лимит строки для sendData - 4096 байт !
    // Поэтому такой режим менее универсален, хотя на первый взгляд проще и не
требует бэкенда

    // Но в будущем могут возникнуть проблемы, если форма будет усложнена

    // Если необходимо вывести сообщение (например для отладки),
    //   тк обычные alert и console.log для telegram Web App не сработают
    //tgWebApp.showAlert('Все ок!')

    tgWebApp.sendData(JSON.stringify(formData));

    // Если не выполняем sendData, то закрываем форму сами
    //closeForm()

```

```

    }
  })
})

/**
 * Метод для закрытия формы
 */
function closeForm() {
  if (mode === null || mode === '') {
    location.href = "tg://resolve?domain=metabot_test_form_bot"
    window.close()
  } else {
    tgWebApp.close()
  }
}

```

// <https://stackoverflow.com/questions/11338774/serialize-form-data-to-json>

```

function getFormData($form) {
  var unindexed_array = $form.serializeArray();
  var indexed_array = {};

  $.map(unindexed_array, function (n, i) {
    indexed_array[n['name']] = n['value'];
  });

  return indexed_array;
}

```

//<https://stackoverflow.com/questions/901115/how-can-i-get-query-string-values-in-javascript>

```

var urlParams = (function (a) {
  if (a == "") return {}
  var b = {}
  for (var i = 0; i < a.length; ++i) {
    var p = a[i].split('=', 2)
    if (p.length == 1)
      b[p[0]] = ""
    else
      b[p[0]] = decodeURIComponent(p[1].replace(/\+/g, " "))
  }
}

```

```

    }
    return b
})(window.location.search.substr(1).split('&'))

// Отправка POST запроса по API с помощью JQuery
// https://reqbin.com/code/javascript/wzp2hxwh/javascript-post-request-example
function sendJQueryRequest(method, url, data, jsonHeaders, callback) {
    $.ajax({
        url: url,
        type: method,
        contentType: 'application/json; charset=utf-8',
        dataType: 'json',
        async: false,

        headers: jsonHeaders,
        data: JSON.stringify(data),

        success: function (response) {
            callback(response, false)
        },
        error: function (jqXHR, textStatus, errorThrown) {
            //alert("Произошла ошибка при обработке API запроса")
            callback(null, true, jqXHR, textStatus, errorThrown)
        }
    })
}

// Отправка POST запроса по API с помощью XHR
// https://reqbin.com/code/javascript/wzp2hxwh/javascript-post-request-example
function sendXhrRequest(method, url, data, jsonHeaders, callback) {
    let xhr = new XMLHttpRequest();
    xhr.open(method, url);

    xhr.setRequestHeader("Accept", "application/json")
    xhr.setRequestHeader("Content-Type", "application/json")

    if (typeof (jsonHeaders) != "undefined") {
        for (let key in jsonHeaders) {
            xhr.setRequestHeader(key, jsonHeaders[key])
        }
    }
}

```

```

    }
}

xhr.onload = () => console.log(xhr.responseText)

xhr.onload = function () {
    // Запрос завершён. Здесь можно обрабатывать результат.
    callback(xhr)
};

//Вызывает функцию при смене состояния.
//xhr.onreadystatechange = function() {
//    callback(xhr)
//}

xhr.send(JSON.stringify(data))
}
</script>

```

<!-- Стили формы, в вашей релизации будет свой блок кода, подключаемый в виде css файла -->

```

<style>
body {
    background-color: #070619;
    /*width: 100%;
    height: 100%; */
    font-size: 18px;
}

.main-container {
    width: 95%;
    height: 95%;
}

form {
    color: #fff;
    border: 1px solid silver;
    border-radius: 10px;
    padding: 10px;
}

```



```
margin: 10px auto 0 auto;
width: 95%;
max-width: 500px;
height: 100%;
}
```

```
form .form-title {
  font-size: 20px;
  text-align: center;
  font-weight: bold;
}
```

```
form .form-group {
  margin: 10px;
}
```

```
form .form-group input {
  font-size: 18px;
}
```

```
form .form-group input[ type=checkbox] {
  width: 20px;
  height: 20px;
}
```

```
form .form-group input.dadata-suggestion {
  max-width: 75%;
}
```

```
form .form-group select {
  font-size: 18px;
}
```

```
.suggestions-suggestions {
  color: #000
}
```

```
form .form-button {
```

```
text-align: center;
margin-bottom: 10px;
}
```

```
form .send-button {
  font-size: 18px;
  border: 1px solid #fff;
  border-radius: 3px;
  background-color: #fff;
  padding: 5px;
  font-weight: bold;
  text-decoration: none;
}
```

```
form .form-button button {
  font-size: 18px;
  font-weight: bold;
}
```

```
.mars {
  position: absolute;
  left: calc(55vw);
  /*right: 0;*/
  top: 0;
  z-index: -10;
  opacity: 0.8;
}
```

```
</style>
```

```
</head>
```

```
<body>
```

```
<!-- HTML КОД формы -->
```

```
<div class="main-container">
```

```
<!-- Для отправки с помощью submit укажите атрибут action данной формы-->
```

```
<form id="form-mars" autocomplete="off">
```

```
<div class="form-title">Заявка для полета на Марс</div>
```

```
<input type="hidden" id="q" name="q" value="">
```

```
<div class="form-group">
  <label for="name">
    Ваше имя:
  </label>
  <div>
    <input type="text" name="name" id="name" placeholder="Юрий Гагарин" required
autofocus>
  </div>
</div>
```

```
<div class="form-group">
  <label for="email">
    Почта:
  </label>
  <div>
    <input type="email" name="email" id="email" placeholder="yuri@gagarin.ru">
  </div>
</div>
```

```
<div class="form-group">
  <label for="age">
    Возраст:
  </label>
  <div>
    <input type="number" name="age" id="age" min=12 max=777 step=1>
  </div>
</div>
```

```
<div class="form-group">
  <label for="specialization">
    Профессия:
  </label>
  <div>
    <select name="specialization" id="specialization" required>
      <option value="engineer" selected>Инженер</option>
      <option value="scientist">Учёный</option>
      <option value="psychologist">Психолог</option>
      <option value="other">Другая</option>
    </select>
  </div>
</div>
```

```

        </select>
    </div>
</div>

<div class="form-group">
    <label for="address">
        Ваш адрес проживания:
    </label>
    <input class="dadata-suggestion" type="text" name="address" id="address"
placeholder="" required>
</div>

<div class="form-group">
    <label for="is_qualified">
        Прошел курсы в Центре<br>подготовки космонавтов
    <input type="checkbox" name="is_qualified" id="is_qualified" value="1">
    </label>
</div>

<div class="form-group">
    <label for="has_experience">
        Я уже летал в космос (имею опыт)
    <input type="checkbox" name="has_experience" id="has_experience" value="1">
    </label>
</div>

<!-- Если нужна отправка фото -->
<!--<div class="form-group">
    <label>
        Фото:
    <input type="file" accept="image/jpeg" name="photo" required>
    </label>
</div>-->

<div class="form-button">
    <!--<button type="submit">Отправить заявку</button>--> <!-- Для отправки с помощью
submit формы -->
    <a class="send-button" id="submit-mars-form" href="#">Отправить заявку</a>
</div>

```

```
</form>
```

```

```

```
</div>
```

```
</body>
```

```
</html>
```

При использовании данного примера замените в следующих строках данные на актуальные для вашего бота:

- let botId = ID_ВАШЕГО_БОТА;
- let botToken = 'ТОКЕН_ВАШЕГО_БОТА';
- let dadataToken = 'ТОКЕН_DADATA'.

И измените код отправки заполненных данных, чтобы данные сначала отправлялись на ваш бэк, а затем с бэка в Metabot API. Также уберите отправку данных внутри параметра script_request_params, чтобы данные к вам на бэк отправлялись без этого параметра, но когда отправляете в Metabot API учитывайте эту особенность.

Речь идет про строку:

```
sendBody = {"script_request_params": formData}
```

Которую для вас нужно заменить на:

```
sendBody = formData
```

Также для понимания принципа работы изучите комментарии в HTML-коде.

Код не требует большого уровня владения HTML или JS для создания подобной формы, но заметим, что важной частью является UI/UX и может потребоваться профессиональный Frontend-разработчик, также необходимо учесть отработку всех исключительных ситуаций, например если произошел сбой API, о чем было упомянуто в теоретической части, во введении, заметим что выше приведен просто пример, все аспекты не проработаны досконально, тк это возможно только на конкретном рабочем примере для вашего бота.

Версия #4

Юрий Гарашко создал 14 June 2023 13:53:53

Ирина Петрова обновил 5 February 2024 14:20:16