CallKeeper IG (Integration Guid)

IT и WEB непрерывно и нелинейно развиваются. И в определённый мо-мент возникает необходимость интеграции между продуктами, которые разрабатывали разные люди, находящиеся в абсолютно разных точках планеты. Синергия для получения максимального результата!
CallKeeper предлагает простую интеграцию с любыми продуктами и си-стемами. Интеграция происходит посредством JavaScript, поэтому потребуются

01.

Общая информация

Номер телефона рассматривается в любом формате, но ОБЯЗАТЕЛЬНО с кодом страны, например:

- +7(452)454-45-45
- 8(445)34 345 34
- 7 45 454 45 554

02.

Как это работает

JavaScript имеет событийно-триггерную архитектуру. Пользователь или сам браузер создают события в зависимости от тех или иных действий. Событие поднимается от элемента до глобального объекта window. А затем исчезает.
Интеграция с CallKeeper происходит посредством создания такого события. В самом простом случае нет необходимости ничего создавать и контроли-ровать исполнение. Достаточно вызвать метод ckEvent из объекта ckForms. После загрузки CallKeeper на сайт, объект ckForms доступен всегда, он глобальный.

Создание простого события

Метод ckForms.ckEvent создаёт событие, которое впоследствии ловит обработчик звонков. Указанный метод содержит в себе все необходимые настройки и не требует дополнительных действий. Единственный необходимый параметр, которые принимает ckEvent — это телефон посетителя. Он должен быть передан, как 'tel':'71111111111'. В результате метод ckForms.ckEvent принимает JavaScript объект, который выглядит так {'tel': '71111111111'}. Номер может быть представлен в любом формате: со скобками, пробелами, дефисами и т.д. Телефон проходит валидацию, и на несуществующий номер звонка не будет.

```
ckForms.ckEvent({
  'tel': '71111111111',
});
```

Событие с указанием телефона менеджера

Здесь объект с параметрами звонка включает в себя дополнительное поле менеджер. Телефон менеджера — это такой номер, с которого произойдёт звонок посетителю сайта. Этот параметр не является обязательным, поэтому его можно пропустить. Единственное правило составление подобного объекта — это соблюдать названия полей. Поле номера телефона посетителя должно быть tel, поле номера телефона менеджера должно быть manager.

```
ckForms.ckEvent({
  'tel': '71111111111',
 'manager': '72222222222',
});
```

Продвинутая функциональность

Ни один параметр, кроме телефона, не является обязательным. От них можно отказаться. Для соединения необходим только телефон посетителя.
Чтобы установить проговариваемые во время звонка параметры нужно использовать поля person, partner, param1, param2. Поле person отвечает за имя клиента, которое он написал в соответствующем поле.
Робот проговорит его оператору, когда он поднимет трубку. Поле partner отвечает за название сервиса, с которым происходит интеграция. Должна со-стоять из русских букв и пробелов. А другие параметры могут хранить любую связанную информацию.

```
ckForms.ckEvent({
  'tel': '71111111111',
  'person': 'Name',
  'manager': '72222222222',
  'mail': {'mail@example.com'},
  'partner': 'партнер',
  'param1': 'Параметр 1',
  'param2': 'Параметр 2'
});
```

CallKeeper

Событие с отчетами на email

```
ckForms.ckEvent({
  'tel': '71111111111',
 'manager': '72222222222',
 'mail': {'mail@example.com'},
});
```

Допустим, что мы хотим отправить по почте сообщение о звонке для ру-ководителя отдела, менеджеру или на какую-то конкретную электронную почту, указанную вручную. Это можно сделать с помощью поля mail. Оно хранит электронный адрес получателя письма. Так же не является обязательным.
А что, если мы хотим отправить письма о неудавшихся звонках на один адрес, а об успешных на другой. Или отправить сразу несколько писем на разные адреса. Для этого нам потребуется объект в формате json. Ключи в этом объекте должны быть либо fail, либо success, либо all, или все вместе, либо он должен быть совсем без ключей и состоять только из адресов почты.

```
{
'success': ['adress@your.mail', 'address@neighbor.mail'],
'fail': ['address@your.mail', 'address@neighbor.mail'],
'all': ['address@your.mail', 'address@neighbor.mail'],
}
```

TEL - номер телефона посетителя

PERSON - имя посетителя

MANAGER - номер телефона менеджера

MAIL - электронная почта, на которую придёт письмо

PARTNER - название продукта/сервиса, который интегрируется с CallKeeper

PARAM1 - дополнительная информация

PARAM2 - дополнительная информация

Описание полей

Первый шаг — это создать объект, в котором будут указанные поля. Важно соблюдать правильность написания названий полей. Второй шаг — это передать такой объект методу ckForms.ckEvent, как показано выше. Имена полей и соответствия:

Практические советы

Перед использованием любого внешнего метода, его наличие в объекте следует проверить, как и наличие самого объекта. Это предотвратит исключение на странице

```
if (window.ckForms && ckForms.ckEvent) {
      ckForms.ckEvent(callObject);
}
```

Для любого приложения тестирование является основой. Также можно протестировать интеграцию. Просто запустить метод ckForms.ckEvent из консоли браузера. Ему нужно передать соответствующие параметры. Для такой проверки необходим сайт с виджетом CallKeeper. В данном случае, будут соединены два представленных абонента.

```
ckForms.ckEvent({
 'tel': '71111111111',
 'manager': '72222222222',
});
```

Если ни одно из предложенных решений не работает в вашем случае, тогда вам следует написать в техподдержку по адресу support@callkeeper.ru

Создание события вручную

Часто по тем или иным причинам нельзя получить доступ к методу напрямую. Общий смысл интеграции в создании события, которое и перехва-тывает обработчик на определённом этапе. Метод ckForms.ckEvent принимает объект в качестве параметра, сам создаёт событие и отправляет его. Если он не доступен, придётся создать событие самостоятельно. Объект с данными по звонку передается в поле detail, как показано выше.

```
document.head.dispatchEvent(new CustomEvent(
'CallKeeperEvent',
  {
    'bubbles': true,
    'cancelable': true,
    'detail': {
    'tel': '79689920798',
    'person': 'Kolya',
    'manager': '79169169618',
    'mail': {'mail@example.com'},
    'partner': 'партнер',
    'param1': 'Параметр 1',
    'param2': 'Параметр 2',
   },
}
)),
```

```
(function()   {
    var e,detail;
     detail = {
         'tel': '79689920798',
         'person': 'Kolya',
         'manager': '79169169618',
         'mail': {'mail@example.com'},
         'partner': 'партнер',
         'param1': 'Параметр 1',
         'param2': 'Параметр 2',
   },
   e = document.createEvent('CustomEvent');
   e.initCustomEvent('CallKeeperEvent', true,
true, detail);
   document.head.dispatchEvent(e);
})();
```

Крайние меры

В разработке для WEB бывает так, что браузеры не поддерживают те или иные способы работы. Это старые или не популярные браузеры, например IE <8. Иногда бывает, что IE <10 так же не работает по стандарту.
Тогда на помощь приходит устаревший метод. В документации по JS на Mozilla Developer Network такое решение указано как устаревшее. И его ис-пользование не рекомендуется. Однако, можно использовать следующий код.

Возникновение события CallKeeperData

Для того, чтобы получать дополнительную информацию о звонке, после его заказа, существует событие CallKeeperData. Оно происходит когда после заказа звонка с одного из инструментов CallKeeper. Это событие можно отсле-дить, создав его обработчик на элементе body или выше (например window).

Пример обработчика:

```
window.addEventListener('CallKeeperData', function(obj)
{
  console.log(obj.detail);
});

```

OBJ.DETAIL - содержит дополнительную информацию о заказанном звонке

CLIENT - номер телефона клиента

ISAPP - true, если звонок заказан с виджета или лидогенератора

ISFORM - true, если звонок заказан с формы

MANAGER - номер телефона менеджера

PERSON - имя клиента

TITLE - название формы