Payture.js

Руководство пользователя

Version 1.0

01.08.2015

[Аннотация к документу]

Олимпийский проспект, 14
Москва, 129090
+7 495 783 83 93
info@payture.com

Оглавление

Описание

Использование библиотеки Payture.js позволяет Продавцу упростить процесс работы с шаблонами при проведении оплаты на стороне платежного шлюза.

Библиотека содержит всю необходимую функциональность для корректного сбора и отправки данных банковской карты Покупателя при оплате и привязке карты.

Payture.js используется в шаблонах оплаты, привязки карт и странице возврата на стороне шлюза в интерфейсах Payture InPay и Payture eWallet.

Библиотека позволяет как использовать стандартный сценарий оплаты, так и настраивать работу шаблона согласно вашим личным требованиям и предпочтениям. Вы также можете реализовать собственный, уникальный сценарий оплаты с помощью готовых методов, описанных в соответствующем разделе данного документа.

Требования к шаблонам

- В шаблоне должно присутствовать скрытое поле Key со значением {key}, в котором передаются параметры платежа.
- HTML-код страницы должен содержать блок с текстом {error}, который выводится при возникновении ошибки.
- На странице шаблона не должно быть ссылок, перенаправляющих пользователей на сторонние вебсайты.
- На странице шаблона запрещено использование внешних ресурсов, т.е. все изображения, файлы стилей и скриптов должны передаваться вместе со страницей шаблона.
- Дополнительные параметры платежа могут быть переданы при инициализации сессии в параметре DATA. Для вывода их на странице оплаты следует использовать запись {param}, где {param} – это имя передаваемого параметра в нижнем регистре.
- Ссылки на используемые ресурсы (стили, изображения) должны иметь вид /Templates/{Key}/{File}, где {Key} – идентификатор продавца. В случае использования нескольких шаблонов крайне желательно хранить все ресурсы в общей папке.

Настройки

Основные параметры

В данной таблице перечислены основные параметры сбора данных, влияющие на работу платежной формы. Вы можете использовать стандартный набор параметров или же передавать свои, что может быть актуально при внесении изменений в HTML-верстку шаблона или при написании собственных сценариев оплаты.

Название	Описание	Формат	По умолчанию
Type	Тип шаблона	Default - стандартный JSON	Default
Event	Событие, по которому происходит сбор и отправка формы	Строка, первое слово – событие, второе – элемент, к которому оно делегируется **Внимание**! Название элемента не должны быть «form»	submit #payForm
Data	Набор полей для ввода данных или значения.	Объект, в качестве ключей используются названия необходимых полей, а в качестве значений могут использоваться: ▪ Соответствующее поле для ввода ▪ Массив полей для ввода (номер карты) ▪ Строка с конкретным значением ▪ Функция, возвращающая строку	CardNumber : [$("input[name=CardNumber0]"), $("input[name=CardNumber1]"), $("input[name=CardNumber2]"), $("input[name=CardNumber3]")], EMonth : $("input[name=EMonth]"), EYear : $("input[name=EYear]"), CardHolder : $("input[name=CardHolder]"), SecureCode : $("input[name=SecureCode]")
TopErrorContainer	Контейнер для вывода ошибок	HTML элемент	#errorTop

TemplateErrors	Текст ошибок, обрабатываемых на стороне клиента	Объект, в качестве ключа используется код ошибки, в качестве значения – текст ошибки	EmptyPan : "Введите от 16 до 19 знаков номера карты", EmptyDate : "Укажите дату, до которой действительна карта", EmptyMonth : "Укажите месяц, до которого действительна карта", EmptyYear : "Укажите год, до которого действительна карта", WrongDate : "Неверно указана дата", EmptyCardHolder : "Укажите имя владельца карты", EmptyCVV : "Укажите код CVV"
ServerErrors	Коды ошибок, возвращаемых сервером	Объект, в качестве ключа используется код ошибки, в качестве значения – текст ошибки	см. перечень кодов ошибок на сайте
RemoveEvent	Событие, по которому происходит удаление карты. **Внимание**! Доступно только для интерфейса eWallet	Строка, первое слово – событие, второе – элемент, к которому оно делегируется	click #removeCard
PaymentKey	Идентификатор продавца в системе. Необходимо для интерфейса eWallet для удаления карт	Строка	
MasterPass	Использование шаблонов Master Pass	Boolean	false
MasterPassUseCVV	Ввод CVV на шаблонах MasterPass	Boolean	false
Inbox	Использование шаблонов для оплаты в интернет-банках	Boolean	false

Дополнительные опции

В данном разделе перечислены дополнительные опции взаимодействия с интерфейсом формы. Большинство из них влияют только на визуальное отображение и работу с элементами формы.

Внимание! Возможность использования некоторых опций зависит от HTML-кода страницы, наличия или отсутствия определенных блоков, их идентификаторов и классов.

Название	Описание	Формат	Значение
DetectDevice	Определение устройства: мобильный, планшет или ПК. Payture.options.IsMobile и Payture.options.IsTablet принимают соответствующие значения true/false	Boolean	true
IsMobile	Мобильное устройство **Внимание**! При задании этого параметра, параметр DetectDevice должен быть false	Boolean	false
IsTablet	Планшет **Внимание**! При задании этого параметра, параметр DetectDevice должен быть false	Boolean	false
CheckNumbers	Проверка ввода цифр в соответствующие поля – номер карты, дата и CVV **Внимание**! Поле для ввода цифр должно иметь класс «onlyNum»	Boolean	true
CheckLetters	Проверка ввода символов в текстовое поле – держатель карты – и автоматический перевод русских символов в латинские **Внимание**! Поле для ввода латинских символов должно иметь класс «onlyLat»	Boolean	true
DetectCardType	Подсветка логотипы МПС при вводе номера карты – Visa или Master Card. **Внимание**! Блок для логотипа должен иметь идентификатор «CardType» и стили для классов	Boolean	true

	«visa» и «master»		
DisableButton	Деактивация кнопки до того момента, пока необходимые поля не будут корректно заполнены	Boolean	true
GroupCardNumber	Автоматическая смена фокуса в полях ввода номера карты или группировка номера по 4 цифры, если это мобильное устройство **Внимание**! Для автоматической смены фокуса необходимо, чтобы у поля имелись атрибуты «prev» и «next», со значением, соответствующим атрибуту «name» поля для фокуса	Boolean	true
ResizeCardNumberInput	Изменение размера последнего поля для ввода карты при вводе 19-значного номера, в случае если номер карты состоит из нескольких полей.	Boolean	false
HideCardList	Для интерфейса eWallet Скрывать поле для выбора карты, если нет привязанных карт	Boolean	true

Функции

Вы можете задать собственный сценарий поведения страницы перед отправкой формы, при выборе привязанной карты (для интерфейса eWallet) или при удалении карты, при показе и скрытии ошибок, обрабатываемых на клиентской стороне.

Кроме того, если вы используете JSON шаблоны, вы должны описать поведение страницы в случае успеха или неуспеха операции.

Название	Описание	Параметры
onBeforeSubmit	Дополнительная функция, выполняется перед отправкой формы	без параметров
onSelectCard	Функция, выполняемая при выборе карты из списка привязанных карт, заменяет функцию по умолчанию	cardId – идентификатор выбранной карты cardMask – маска выбранной карты в формате «123456xxxxxx1234»
onShowErrors	Показ сообщений об ошибках, заменяет функцию по умолчанию	Success - true/false, Element - строка, с названием поля, соответствует ключам в параметре «Data», ErrorMessage - строка с текстом сообщения
onHideErrors	Скрытие сообщений об ошибках, заменяет функцию по умолчанию	element – название поля, соответствует ключам в параметре «Data»
onRemoveCard	Функция выполняемая при удалении карты, заменяет функцию по умолчанию	cardId – индентификатор удаленной карты
onSuccess	Для JSON шаблонов Функция, выполняемая в случае успеха операции	redirectUrl – адрес перенаправления пользователя
onError	Для JSON шаблонов Функция, выполняемая в случае ошибки	errorCode – код ошибки canRetry – возможность повторной оплаты redirectUrl – адрес возврата key – новый ключ для оплаты

Подключение

Для работы с библиотекой Payture.js вам, помимо самой библиотеки, понадобятся библиотека jQuery версии 1.11, jquery migrate, и плагин device.js для определения устройства пользователя. Все необходимые библиотеки уже установлены на сервере Payture; от вас потребуется лишь разместить на своей странице оплаты следующие скрипты:

```
<script type="text/javascript"
src="/Templates/jquery.11.min.js"></script>

<script type="text/javascript" src="/Templates
/jquery.migrate.min.js"></script>

<script type="text/javascript" src="/Templates
/device.min.js"></script>

<script type="text/javascript" src="/Templates
/payture.js"></script>
```

Payture InPay

Стандартное подключение

Для простого подключения скрипта со стандартными параметрами используйте функцию:

```
Payture.InPay()
```

Расширение опций

Вы можете включить или отключить любую опцию при подключении скрипта, например, отключить определение устройства, блокировку кнопки до того момента, пока не будут заполнены необходимые поля, и изменение размеров поля ввода номера карты для 19-значного номера.

```
Payture.InPay({
        DetectDevice : false,
        DisableButton : false,
        ResizeCardNumberInput : true
})
```

Настройка полей для ввода

Обратите внимание, что при изменении параметра «**Data**» в нем должны быть перечислены все необходимые поля. Тоже касается и изменения параметра «**TemplateErrors**».

```
Payture.InPay({
        Data : {
                CardNumber : $("input[name=CN]"),
                CardHolder : "Cardholder Name",
                EMonth : $("[name=EMonth]"),
                EYear : function () { return $("#Year").val(); },
                SecureCode : $("[name=CVV]")
        }
})
```

Изменение функции отображения ошибок

Вы можете написать собственную функцию отображения ошибок на шаблоне, которая заменит стандартную. Функция принимает на вход объект «**error**», состоящий из трех элементов: «**Element**» - имя элемента в объекте «**Data**», «**Success**» - верно или неверно значение данного поля, и «**ErrorMessage**» - сообщение об ошибке.

```
Payture.InPay({
        onShowError : function (error) {
              if (!error.Success) {
                    alert("Ошибка в поле " + error.Element + ": " +
              error.ErrorMessage);
              }
        }
})
```

JSON шаблоны

Для работы с JSON шаблонами необходимо передать параметр «**Type**» и описать поведение в случая успеха или неуспеха операции – для этого используются функции «**OnSuccess**» и «**OnError**» соответственно.

В случае успеха операции в функцию «**OnSuccess**» передается «**redirectUrl**» - адрес перенаправления пользователя, указанный вами при подключении к платежному шлюзу.

В случае ошибки в функцию «**OnError**» передаются параметры:

- **errorCode** – код ошибки от платежного шлюза
- **canRetry** – возможность повторного платежа
- **redirectUrl** – адрес возврата пользователя, указанный при подключении к платежному шлюзу
- **key** – новый ключ, в соответствующее поле подставляется автоматически

```
Payture.InPay({
        Type : "JSON",
        OnSuccess : function (redirectUrl) {
              // Тело функции...
```

```
        },
        OnError : function (errorCode, canRetry, redirectUrl, key) {
              // Тело функции
        }
})
```

Для корректной обработки ответа при оплате картой, защищенной системой аутентификации 3-D Secure, в тело шаблона оплаты и в тело шаблона возврата необходимо вставить HTML код, в который будет вставляться ошибка после перенаправления пользователя со страницы ACS банка.

```
<span class="response_json">{response_json}</span>
```

Если карта защищена 3-D Secure, то в случае ошибки покупатель возвращается на шаблон оплаты, при этом выполнится функция «**onError**».

Если операция по 3-D Secure была успешна, покупатель будет перенаправлен на шаблон страницы возврата, где функция «**Payture.getJSONResponse()**» вернет вам ответ об успехе операции, аналогичный ответу в случае успеха операции с картой, не защищенной аутентификацией 3-D Secure.

Payture eWallet

Шаблон добавления карты

Для подключения скрипта для шаблона добавления карты со стандартными параметрами используйте функцию:

```
Payture.eWalletAdd();
```

Все параметры, опции и собственные функции настраиваются аналогично интерфейсу InPay.

Шаблон оплаты

Для подключения скрипта для шаблона оплаты по карте со стандартными параметрами используйте функцию:

```
Payture.eWalletPay();
```

Удаление карт

Если вы хотите добавить возможность покупателю удалять привязанные карты на странице оплаты, вы должны передавать в настройках параметр «**PaymentKey**». Так же вы можете описать поведение страницы в случае успешного удаления карты.

```
Payture.eWalletPay({
      PaymentKey : “Merchant”,
      OnRemoveCard : function (cardId) {
            // Тело функции...
      }
});
```

JSON шаблоны

Для работы с JSON шаблонами необходимо передать параметр «**Type**» и описать поведение в случая успеха или неуспеха операции – функции «**OnSuccess**» и «**OnError**» соответственно.

В случае успеха операции в функцию «**OnSuccess**» передается параметр «**redirectUrl**» - адрес перенаправления пользователя, указанный вами при подключении к платежному шлюзу.

В случае ошибки в функцию «**OnError**» передаются параметры:

- **errorCode** – код ошибки от платежного шлюза
- **canRetry** – возможность повторного платежа
- **redirectUrl** – адрес возврата пользователя, указанный при подключении к платежному шлюзу
- **key** – новый ключ, в соответствующее поле подставляется автоматически

```
Payture.eWalletAdd({
        // Необходимые и дополнительные параметры
})
Payture.eWalletPay({
        // Необходимые и дополнительные параметры
})
```

Для корректной обработки ответа при оплате картой, защищенной 3-D Secure, в тело шаблона оплаты и в тело шаблона возврата необходимо вставить HTML код, в который будет вставляться ошибка после перенаправления пользователя со страницы ACS банка.

```
<span class="response_json">{response_json}</span>
```

Если карта защищена 3-D Secure, то в случае ошибки покупатель возвращается на шаблон оплаты, при этом выполнится функция «**onError**».

Если авторизация карты в системе 3-D Secure была успешна, покупатель будет перенаправлен на шаблон страницы возврата, где функция «**Payture.getJSONResponse()**» вернет вам ответ об успехе операции, аналогичный ответу в случае успеха операций с картами, не защищенными 3-D Secure.

MasterPass

MasterPass – это передовое решение, предоставляемое платежной системой MasterCard для электронных кошельков и онлайн-магазинов. Платформа MasterPass позволяет хранить реквизиты банковских карт в едином интерфейсе, что упрощает процесс онлайн-покупок и делает его более защищенным.

При оформлении заказа на Интернет-сайте, поддерживающем MasterPass, пользователю нужно только нажать кнопку «Оплата MasterPass» и выбрать банковскую карту для оплаты.

Чтобы активировать поддержку MasterPass на Вашем Интернет-сайте параллельно со стандартным механизмом оплаты, необходимо добавить в шаблон код, содержащий кнопку MasterPass:

```
<a href="" id="masterPass">

   <img alt="Wallet Checkout"
src="https://www.mastercard.com/mc_us/wallet/img/ru/RU/mcpp_wllt_btn_chk
_160x037px.png">

</a>
```

При подключении основного скрипта, добавить параметр «**MasterPass**» со значением «**true**» и параметр «**MasterPassUseCVV**» — запрашивать ли у покупателя CVV (обязательно «true» для интерфейса eWallet).

```
Payture.InPay({

      MasterPass : true

});

Payture.eWalletPay({

      MasterPass : true,

      MasterPassUseCVV : true

});
```

Интернет-Банкинг

Данное решение предоставляет Покупателю возможность использовать для оплаты личный кабинет своего банка.

На странице оплаты Покупателю будет предоставлено на выбор несколько способов оплаты заказа – банковская карта или интернет-банкинг. При выборе интернет-банкинга пользователь будет перенаправлен на страницу нашего партнера для выбора своего банка.

Для формирования списка способов оплаты необходимо на страницу добавить HTML код:

```
<div class="container section" id="paymentType">
      <a class="icon payCard" href="#paymentCard">Банковская
карта</a>
      <a class="icon" href="#inbox" rel="1"
target="_blank">Интернет-банкинг </a>
</div>
```

Для корректной работы контейнер обязательно должен иметь идентификатор «**paymentType**», ссылка на интернет-банкинг должна открываться в новом окне и содержать атрибут «**rel**» со значением «**1**».

Также при подключении скриптов, необходимо передать параметры «**Type**» со значением «**JSON**» и «**Inbox**» со значением «**true**».

```
Payture.InPay({
      Type : "JSON",
      Inbox : true
});
Payture.eWalletPay({
      Type : "JSON",
      Inbox : true
});
```

Переводы с карты на карту

Функциональность P2P переводов (перевод с карты на карту) позволяет переводить денежные средства с одной карты международных платежных систем Visa и MasterCard на другую.

При совершении перевода с карты на карту должны выполнятся следующие требования, установленные платежными системами: обязательный ввод CVV2/CVC2 кода для карты-источника и обязательная проверка карты-источника с использованием технологии 3-D Secure.

На страницу оплаты необходимо добавить поле input для ввода номера карты, на которую пользователь хочет перевести денежные средства.

```
<input type='text' name='CardTo' placeholder='Номер карты
получателя'>
```

Также неоходимо передать параметр «**Data**», в котором указать добавленное поле:

```
Payture.InPay({

 Data : {

       CardNumber : [$("input[name=CardNumber0]"),
 $("input[name=CardNumber1]"), $("input[name=CardNumber2]"),
 $("input[name=CardNumber3]")],

       EMonth : $("input[name=EMonth]"),

       EYear : $("input[name=EYear]"),

       CardHolder : $("input[name=CardHolder]"),

       SecureCode : $("input[name=SecureCode]"),

       CardTo : $("input[name=CardTo]")

}

});
```

Подключение для интерфейса eWallet – аналогичное, с учетом необходимых для данного интерфейса полей в параметре «**Data**».

Методы

Мы предлагаем набор готовых методов для работы с картами, для использования нестандартных сценариев оплаты, для привязки карты и других действий.

Pay

Данный метод позволяет напрямую производить оплату по карте. Для этого необходимо передать в метод тип шаблона – «**Default**» (стандартный шаблон оплаты) или «**JSON**», тип используемых API – «**InPay**» или «**eWallet**», платежный ключ – обычно значение скрытого поля «**Key**», и объект «**Data**», представляющий собой набор параметров карты.

Если вы используете JSON шаблоны, необходимо так же передать в качестве параметров описание функций «**OnSuccess**» и «**OnError**».

```
Payture.Pay({
        TypeTemplate : “JSON”,
        TypeAPI : “InPay”,
        Key : $(“[name=Key]”).val(),
        Data : {
              CardNumber : “1234567890123456”,
              CardHolder : “TEST”,
              EMonth : 12,
              EYear : 16,
              SecureCode : 123
        },
        OnSuccess : function (redirectUrl) {
              // Тело функции
        },
        OnError : function (errorCode, canRetry, redirectUrl, key) {
              // Тело функции
        }
});
```

Если вы используете интерфейс eWallet, в параметре «**Data**» должен присутствовать идентификатор карты «**CardId**» и флаг привязки карты «**AddCard**».

При использовании интерфейса eWallet, состав параметра «**Data**» отличается для оплаты по привязанной и непривязанной карте: в случае привязанной карты вы должны передать только идентификатор карты «**CardId**» и «**SecureCode**».

```
Payture.Pay({
        …
        Data : {
              CardId : “1234”,
              SecureCode : 123
        }
});
```

В случае непривязанной карты – необходимо передать идентификатор карты с значением «**FreePay**», номер карты, срок действия карты – месяц и год в соответствующих полях, имя держателя карты, CVV, а также флаг привязки карты «**AddCard**» с значением «**true**», если вы хотите привязать карты или «**false**», если нет.

```
Payture.Pay({
        …
        Data : {
              CardId : “FreePay”,
              CardNumber : “1234567890123456”,
              CardHolder : “TEST”,
              EMonth : 12,
              EYear : 16,
              SecureCode : 123,
              AddCard : true
        }
});
```

Результат выполнения данного метода будет зависеть от типа шаблона. Если вы используете стандартный шаблон, вы будете либо возвращены на шаблон оплаты с ошибкой, либо перенаправлены на шаблон возврата пользователя. Если же вы используете JSON шаблон, вы остаетесь на странице оплаты и будут

выполнены функции, описанные вами в параметрах «**OnSuccess**» и «**OnError**», в случае успеха или неуспеха операции соответственно.

Add

Данный метод позволяет производить оплату по карте. Для этого необходимо передать в метод тип шаблона – «**Default**» (стандартный) или «**JSON**», ключ – обычно значение скрытого поля «**Key**», и объект «**Data**», состоящий из параметров карты.

Если вы используете JSON шаблоны, необходимо так же передать в качестве параметров описание функций «**OnSuccess**» и «**OnError**».

```
Payture.Add({
        TypeTemplate : “JSON”,
        Key : $(“[name=Key]”).val(),
        Data : {
              CardNumber : “1234567890123456”,
              CardHolder : “TEST”,
              EMonth : 12,
              EYear : 16,
             SecureCode : 123
        },
        OnSuccess : function (redirectUrl) {
              // Тело функции
        },
        OnError : function (errorCode, canRetry, redirectUrl, key) {
              // Тело функции
        }
});
```

В результате выполнения данного метода, вы будете либо возвращены на шаблон оплаты с ошибкой, либо перенаправлены на шаблон возврата пользователя (если вы используете стандартный шаблон), или, если вы используете JSON шаблон, вы остаетесь на странице оплаты и будут выполнены функции, описанные вами в параметрах «**OnSuccess**» и «**OnError**», в случае успеха или неуспеха операции соответственно.

Validate

Данный метод используется для проверки корректности вводимых значений. На вход принимает объект, состоящий из поля «**Name**», содержащего имя проверяемого поля (соответствует именам полей из набора «**Data**»), и поля «**Value**», содержащего значение проверяемого поля.

```
Payture.Validate({
        Name : “CardNumber”,
        Value : “1234567890”
});
```

В результате метод возвращает объект с полями «**Success**» - успешно или не успешно пройдена валидация, «**ErrorMessage**» - текст ошибки (соответствует передаваемым ошибкам в параметре «**TemplateErrors**»), и «**Element**» - название элемента, согласно именам параметра «**Data**», к которому относится ошибка.

```
{
        Success : false,
        ErrorMessage : “Введите от 16 до 19 знаков номера карты”,
        Element : “CardNumber”
}
```

RemoveCard

Внимание! Данный метод используется только для интерфейса eWallet.

С помощью данного метода вы можете удалить привязанную карту покупателя из списка.

Для работы необходимо передать методу объект из двух параметров:

- **CardId** – идентификатор карты, которую необходимо удалить;
- **PaymentKey** – идентификатор продавца в системе платежного шлюза.

```
Payture.RemoveCard({
      CardId : “1234”,
      PaymentKey : ”Merchant”,
      onRemove : function (CardId) { // Тело функции }
});
```

В результате успешного удаления карты будет вызвана функция, переданная в качестве параметра «**onRemove**». Если такой параметр не был передан, будет выполнена функция обновления списка карт по умолчанию.

GetList

Внимание! Данный метод доступен только для интерфейса eWallet.

Вы можете получить расширенный список карт с дополнительными параметрами, чтобы создать собственный сценарий построения списка карт. Для это в шаблоне оплаты в HTML код вставить следующий скрипт:

```
<script>
      {CardConfigurationJS}
</script>
```

После этого вызвать метод GetList.

```
Payture.GetList();
```

Метод обрабатывает данные карт и возвращает коллекцию карт. Ключом каждой карты является маскированный номер карты формата «123456xxxxxx1234», значением является объект, содержащий в себе «**Id**» - идентификатор карты, строка, флаг «**NoCVV**» - возможность оплаты по данной карте без ввода CVV, флаг «**Expired**» - является ли карта просроченной и переменная типа Date «**LastPay**» - дата последней успешной оплаты по карте.

```
{
      “123456xxxxxx1234” : {
            Id : “1234”,
            NoCVV : true,
            Expired : False,
            LastPay : Date {Fri Dec 05 2014 12:01:10 GMT+0300
   (MSK)}
      }
}
```