Использование виджетов для платежей АПК Ассист

- Использование виджета Apple Pay для платежей АПК Ассист
  - Общие сведения
  - Порядок действий при оплате с помощью виджета
  - Список действий для организации оплат с помощь виджета
  - Размещение виджета на странице интернет-магазина
  - Описание виджета
    - Установка виджета
    - Передача параметров заказа
    - Пример скрипта
- Использование виджета Google Pay для платежей АПК Ассист
  - Общие сведения
  - Порядок действий при оплате с помощью виджета
  - Список действий для организации оплат с помощь виджета
  - Размещение виджета на странице интернет-магазина
  - Описание виджета
    - Установка виджета
    - Настройка виджета
    - Пример скрипта

Использование виджета Apple Pay для платежей АПК Ассист

Общие сведения

Виджет Apple Pay позволяет покупателям использовать Apple Wallet для оплаты в интернет-магазине без переадресации на внешние платежные страницы и без ввода данных банковской карты.

Apple Pay поддерживается на устройствах iOS и macOS.

Визуально виджет представлен в виде кнопки Apple Pay, нажатие на которую запускает процесс авторизации и оплаты заказа.

Порядок действий при оплате с помощью виджета

1. Покупатель выбирает товары на сайте интернет-магазина.
2. Интернет-магазин передает данные заказа в виджет.
3. Покупатель нажимает кнопку оплаты Apple Pay.
4. На устройстве с подключенным Apple Pay появляется всплывающее окно с выбором привязанной карты.
5. Покупатель выбирает карту и подтверждает оплату на устройстве с помощью пароля, TouchID или FaceID.
6. Apple Pay формирует зашифрованный пакет с токеном.
7. Виджет получает пакет с токеном от Apple Pay.
8. Виджет передает зашифрованный пакет с токеном и данными платежа АПК Ассист.
9. АПК Ассист расшифровывает пакет с токеном и данными платежа.
10. АПК Ассист проводит оплату токеном через процессинг расчётного банка.
11. АПК Ассист возвращает результаты проведения оплаты интернет-магазину.
12. Интернет-магазин отображает результат оплаты для покупателя.

Список действий для организации оплат с помощь виджета

Для организации оплат с помощь виджета на странице интернет-магазина необходимо выполнить следующие подготовительные шаги:

- оформить заявку на подключение оплаты Apple Pay в АПК Ассист;
- пройти регистрацию в Apple;
- создать сертификат в Личном кабинете предприятия;
- подписать полученный сертификат;
- загрузить подписанный сертификат в Личном кабинете предприятия;
- подготовить страницы интернет-магазина для оплаты;
  - обязательны использование протокола https на странице с виджетом и поддержка протокола TLS версии 1.2;
  - домен должен иметь действующий SSL сертификат.

Размещение виджета на странице интернет-магазина

Для размещения виджета на странице интернет-магазина следует выполнить следующие действия:

- поместить виджет на страницу оплаты интернет-магазина;
- настроить виджет;
- проверить возможность оплаты с помощью виджета;

- передать параметры заказа;
- получить и обработать результат оплаты.

Описание виджета

Виджет представляет собой HTML-код и JS-скрипт, которые необходимо разместить и настроить на странице оплаты интернет-магазина.

Установка виджета

В том месте страницы интернет-магазина, на которой планируется разместить кнопку оплаты Apple Pay, необходимо добавить следующий код:

```
<button id="apple-pay-button"></button>
```

Для кнопки можно задать цвет, тип и размеры. Примеры типов кнопок и их описание доступны на сайте https://developer.apple.com/design/human-interface-guidelines/apple-pay/overview/buttons-and-marks/.

Для настройки внешнего вида кнопки необходимо добавить описание стиля в теге <head></head>.

```
<style>
  #apple-pay-button {
    display: none;
    background-color: black;
    background-image: -webkit-named-image(apple-pay-logo-white);
    background-size: 100% 100%;
    background-origin: content-box;
    background-repeat: no-repeat;
    width: 100%;
    height: 44px;
    padding: 10px 0;
    border-radius: 10px;
  }
</style>
```

Подробнее о настройке внешнего вида кнопки см. https://developer.apple.com/documentation/apple_pay_on_the_web/displaying_apple_pay_buttons.

Передача параметров заказа

После нажатия на кнопку Apple Pay на странице интернет магазина создается сессия ***applePaySession*,** в которую нужно передать параметры заказа.

В качестве параметров заказа следует передать сумму и валюту заказа, а также поддерживаемые типы карт.

```
    Const currency = $('#currency').val(); //
    Const paymentRequest = {
      countryCode: region.toUpperCase(),
      currencyCode: currency.toUpperCase(),
      total: {
        label: 'Your label', //
        amount: $('#amount').val() //
      },
      supportedNetworks:['masterCard', 'visa'],
      merchantCapabilities: [ 'supports3DS' ] //
    };
Const applePaySession = new window.ApplePaySession(1, paymentRequest);
```

Для обработки сессии используются два метода:

- onvalidatemerchant;
- onpaymentauthorizad.

В методе *onvalidatemerchant* осуществляется проверка платежной сессии, метод выполняется при отображении всплывающего окна Apple Pay.

Метод *onpaymentauthorizad* выполняется при подтверждении операции оплаты. В метод передается платежный токен, полученный от Apple Pay, а также параметры заказа.

Список параметров заказа

Параметр	**Описание**
merchant_id	Идентификатор интернет-магазина
amount	Сумма заказа в единицах валюты
currency	Валюта заказа
ordernumber	Уникальный номер заказа на стороне интернет-магазина
comment	Комментарий
email	Email покупателя
firstname	Имя покупателя
lastname	Фамилия покупателя
middlename	Отчество покупателя

В качестве значений могут быть указаны названия полей заполненной платежной формы, например:

```
var data = {
            token : event.payment.token,
            merchant_id : $('#merchant_id').val(),
            email : $('#email').val(),
            amount : $('#amount').val(),
            currency : $('#currency').val(),
            ordernumber : $('#ordernumber').val(),
            email : $('#email').val(),
            firstname : $('#firstname').val(),
            middlename : $('#middlename').val(),
            lastname : $('#lastname').val(),
            comment : $('#comment').val()
};
```

После завершения оплаты сервер возвращает в ответ номер заказа и статус. Для обработки ответа необходимо добавить соответствующий код:

```
$.post("/pay/tokenpay_widget_ap.cfm", JSON.stringify(data)).then(function (result) {
  //  :
  if (!result.hasOwnProperty('firstcode') && JSON.stringify(result.order.orderstate) == '"Approved"' && JSON.
stringify(result.order.orderstate) == '"Delayed"') {
    applePaySession.completePayment(ApplePaySession.STATUS_SUCCESS);
  } else {
    applePaySession.completePayment(ApplePaySession.STATUS_FAILURE);
  }
});
```

Если по каким-то причинам оплата заказа прошла неуспешно, то сервис вернет сообщения об ошибках (ненулевые значения параметров firstcode, secondcode).

Скрипт использует Apple Pay API, который поддерживается:

- iOS 10+ в браузере Safari и в объектах SFSafariViewController, а также на моделях iPhone с помощью Secure Element (SE и 6+);
- macOS 10.12+ в браузере Safari на компьютерах с Touch ID (MacBook Pro) или при подключенном iPhone или Apple Watch для подтверждения платежей.

Пример скрипта

Ниже приведен пример скрипта виджета, который может быть размещен на платежной странице интернет-магазина.

```
<script type="text/javascript">
document.addEventListener('DOMContentLoaded', function(){
    if (window.ApplePaySession) {
        //      Apple Pay
```

```
        if (ApplePaySession.canMakePayments) {
            document.getElementById('apple-pay-button').style.display = 'block';
            document.getElementById('apple-pay-button').addEventListener('click', applePayButtonClicked);
        }
    } else {console.log("ApplePaySession not available"); }
});

function applePayButtonClicked() {
    const region = 'RU';
    const currency = $('#currency').val();//
    const paymentRequest = {
        countryCode: region.toUpperCase(),
        currencyCode: currency.toUpperCase(),
        total: {
            label: 'Your label', //
            amount: $('#amount').val()//
        },
        supportedNetworks:['masterCard', 'visa'],
        merchantCapabilities: [ 'supports3DS' ]
    };
    const version = window.ApplePaySession.supportsVersion(3)
                    ? 3
                    : window.ApplePaySession.supportsVersion(2)
                    ? 2 : 1;
    const applePaySession = new window.ApplePaySession(version, paymentRequest);
    console.log("start session");

    //     merchant session.
    applePaySession.onvalidatemerchant = function (event) {
        console.log("onvalidatemerchant in");
        var data = {
            validationUrl: event.validationURL
        };
        console.log(JSON.stringify(data));

        //     ,   API
        $.post("/pay/apple_pay_comm.cfm", data).then(function (result) {
            applePaySession.completeMerchantValidation(result);
        });
    }

  //
    applePaySession.onpaymentauthorized = function (event) {
        console.log("onpaymentauthorized in");
        //var email = event.payment.shippingContact.emailAddress; //    e-mail
        //var phone = event.payment.shippingContact.phoneNumber; //
        //    https://developer.apple.com/reference/applepayjs/paymentcontact

        //
        var data = {
            token : event.payment.token,
            merchant_id : $('#merchant_id').val(),
            email : $('#email').val(),
            amount : $('#amount').val(),
            currency : $('#currency').val(),
            ordernumber : $('#ordernumber').val(),
            email : $('#email').val(),
            firstname : $('#firstname').val(),
            middlename : $('#middlename').val(),
            lastname : $('#lastname').val(),
            comment : $('#comment').val()
        };
        //    ,   API
        console.log(JSON.stringify(event.payment.token));
        $.post("/pay/tokenpay_widget_ap.cfm", JSON.stringify(data)).then(function (result) {
            if (!result.hasOwnProperty('firstcode') && JSON.stringify(result.order.orderstate) == '"Approved"'
&& JSON.stringify(result.order.orderstate) == '"Delayed"') {
            applePaySession.completePayment(ApplePaySession.STATUS_SUCCESS);
            } else {
            applePaySession.completePayment(ApplePaySession.STATUS_FAILURE);
            }
```

```
        });
    };
    applePaySession.begin();
}

</script>
<style>
  #apple-pay-button {
    display: none;
    background-color: black;
    background-image: -webkit-named-image(apple-pay-logo-white);
    background-size: 100% 100%;
    background-origin: content-box;
    background-repeat: no-repeat;
    width: 100%;
    height: 44px;
    padding: 10px 0;
    border-radius: 10px;
  }
</style>
```

Использование виджета Google Pay для платежей АПК Ассист

Общие сведения

Виджет Google Pay позволяет покупателям - клиентам Google осуществлять оплату в интернет-магазине без переадресации на внешние платежные страницы и без ввода данных банковской карты.

Визуально виджет представлен в виде кнопки Google Pay, нажатие на которую запускает процесс авторизации и оплаты заказа. Виджет необходимо разместить на странице интернет-магазина и передать данные заказа. Виджет самостоятельно запрашивает токен и проводит оплату. На стороне интернет-магазина производится обработка ответа и возвращение ответа покупателю.

Использование виджета Google Pay позволяет осуществлять оплату токенизированными картами Google Pay. При этом на устройствах без установленного приложения Google Pay покупателю будет предложено выбрать сохраненную карту из Google.

⚠ Google Pay работает с картами Visa и MasterCard.

Порядок действий при оплате с помощью виджета

1. Покупатель выбирает товары на сайте интернет-магазина.
2. Интернет-магазин передает данные заказа в виджет.
3. Покупатель нажимает кнопку оплаты Google Pay.
4. Открывается специальный диалог браузера, в котором можно выбрать одну из карт, привязанных ранее.
5. Покупатель выбирает карту и подтверждает оплату на устройстве.
6. Приложение Google Pay формирует зашифрованный пакет с токеном.
7. Виджет получает пакет с токеном от Google Pay.
8. Виджет передает зашифрованный пакет с токеном и данными платежа АПК Ассист.
9. АПК Ассист расшифровывает пакет с токеном и данными платежа.
10. АПК Ассист проводит оплату токеном через процессинг расчётного банка.
11. АПК Ассист возвращает результаты проведения оплаты интернет-магазину.
12. Интернет-магазин отображает результат оплаты для покупателя.

Список действий для организации оплат с помощь виджета

Для организации оплат с помощь виджета на странице интернет-магазина необходимо выполнить следующие подготовительные шаги:

- оформить заявку на подключение оплаты Google Pay в АПК Ассист;
- пройти регистрацию в Google и заполнить форму регистрации https://services.google.com/fb/forms/googlepayAPIenable/;
- проверить выполнение требований по брендированию Google;
- подтвердить домен в Google;
- подготовить страницы интернет-магазина для оплаты;
    - обязательно использование протокола https на странице с виджетом;
    - домен должен иметь действующий TSL сертификат.

При использовании необходимо учитывать требования Google https://payments.developers.google.com/terms/sellertos, включая список запрещенных товаров и услуг https://payments.developers.google.com/terms/aup, а также требования к брендированию https://developers.google.com/pay/api/web/guides/brand-guidelines.

Размещение виджета на странице интернет-магазина

Для размещения виджета на странице интернет-магазина следует выполнить следующие действия:

- поместить виджет на страницу оплаты интернет-магазина;
- настроить виджет;
- проверить возможность оплаты с помощью виджета;
    - передать параметры заказа;
    - получить и обработать результат оплаты.

Описание виджета

Виджет представляет собой HTML-код и JS-скрипт, которые необходимо разместить и настроить на странице оплаты интернет-магазина.

Установка виджета

В том месте страницы интернет-магазина, на которой планируется разместить кнопку оплаты Google Pay, необходимо добавить следующий код:

```
<div id="container"></div>
```

В теге *id="container"* будет размещена кнопка оплаты Google Pay. При необходимости идентификатору тега можно присвоить другое значение. Для этого следует внести соответствующие изменения в метод ***createButton* функции *addGooglePayButton*,** например**:**

```
document.getElementById('mynewcontainer').style.display = 'block'; //mynewcontainer –
```

Дополнительно на той же странице необходимо подключить JS-скрипт для вызова Google Pay API и JS-скрипт оплаты.

```
<script async src="https://pay.google.com/gp/p/js/pay.js" onload="onGooglePayLoaded()"></script>
```

Настройка виджета

Для настройки окружения следует задать параметр *environment* в функции ***getGooglePaymentsClient***. Для работы с реальными данными нужно указать значение '*PRODUCTION*', а для тестирования – значение '*TEST*'.

```
function getGooglePaymentsClient() {
  if ( paymentsClient === null ) {
  paymentsClient = new google.payments.api.PaymentsClient({environment: 'TEST'});
  }
  Return paymentsClient;
}
```

Далее необходимо указать тип аутентификации карт, которые будут приниматься к оплате в интернет-магазине. Для этого используется параметр ***allowedCardAuthMethods***, который может принимать следующие значения:

«PAN_ONLY» - аутентификация карт, сохраненных в аккаунте Google;

«CRYPTOGRAM_3DS» - аутентификация карт, хранящихся в виде токенов Google Pay, используется только на мобильных устройствах с установленным приложением Google Pay.

```
const allowedCardAuthMethods = ["PAN_ONLY", "CRYPTOGRAM_3DS"];
```

Также следует определить, карты каких платежных систем будут приниматься к оплате в интернет-магазине. Для этого предназначен параметр ***allowedCardNetworks:***

```
const allowedCardNetworks = ["MASTERCARD", "VISA"];
```

Теперь нужно вставить идентификатор интернет-магазина в скрипт виджета (параметр *gatewayMerchantId*):

```
const tokenizationSpecification = {
  type: 'PAYMENT_GATEWAY',
  parameters: {
    'gateway': 'belassist',
    'gatewayMerchantId': '02510116604241796260'
  }
}
```

Подключенный JS-скрипт Google Pay API запускает обработчик, который выполняет проверку устройства и отображает кнопку:

```
function onGooglePayLoaded() {
  const paymentsClient = getGooglePaymentsClient()
  paymentsClient.isReadyToPay(getGoogleIsReadyToPayRequest()) //
    .then(funtion(response) {
      If (response.result) {
      addGooglePayButton(); //
       prefetchGooglePaymentData();
       }
    })
    .catch(function(err) { //
      console.error(err); //
    })
}
```

Для отображения кнопки скрипт вызывает функцию ***addGooglePayButton*. При этом можно настроить вид кнопки** Google Pay. Для этого необходимо внести изменения в метод ***createButton***:

- *buttonColor* – задает цвет кнопки, возможные значения black (черный) и white (белый);
- *buttonType* - задает тип кнопки, возможные значения long (длинная) и short (короткая).

Подробная информация о добавлении кнопки приведена в документации Google Pay API https://developers.google.com/pay/api/web/reference/object?hl=ru#ButtonOptions.

Пример функции ***addGooglePayButton***, которая добавляет большую кнопку черного цвета:

```
function addGooglePayButton() {
  const paymentsClient = getGooglePaymentsClient();
  const button =
     paymentsClient.createButton({onClick: onGooglePaymentButtonClicked, buttonColor:'black',
buttonType:'long'}); //
        document.getElementById('container').appendChild(button); //
        document.getElementById('container').style.display = 'block';
}
```

При настройке кнопки необходимо учитывать требования по брендированию Google.

Если устройство поддерживает оплату с помощью Google Pay, то будет отображена кнопка в соответствии с настройками.

Нажатие на кнопку инициирует передачу информации об операции в Google Pay с помощью функции ***getGoogleTransactionInfo*** . Для операции необходимо передать следующие параметры:

- *currencyCode* — валюта заказа;
- *totalPrice* — сумма заказа;
- *totalPriceStatus* — статус суммы заказа.

Параметр *totalPriceStatus* может принимать следующие значения:

- *NOT_CURRENTLY_KNOWN* — используется для проверки возможностей оплаты;
- *ESTIMATED* — сумма заказа может быть изменена позже, в зависимости от ответа;
- *FINAL* — сумма заказа окончательная, не изменится в процессе.

Пример функции ***getGoogleTransactionInfo***:

```
function getGoogleTransactionInfo() {
  return {
    currencyCode: $('#currency').val(), //
    totalPriceStatus: 'FINAL', //
    totalPrice: $('#amount').val() //
  };
}
```

После подтверждения оплаты покупателем результат будет возвращен в функцию ***processPayment***.

Для проведения платежа в виджет следует передать данные заказа (функция ***processPayment).***

Список параметров функции ***processPayment***

Параметр	**Описание**
merchant_id	Идентификатор интернет-магазина
amount	Сумма заказа в единицах валюты
currency	Валюта заказа
ordernumber	Уникальный номер заказа на стороне интернет-магазина
comment	Комментарий
email	Email покупателя
firstname	Имя покупателя
lastname	Фамилия покупателя
middlename	Отчество покупателя

В качестве значений могут быть указаны названия полей заполненной платежной формы. Пример функции ***processPayment***:

```
function processPayment(paymentData) {
  var data = {
        paymentData : paymentData,
        merchant_id : $('#merchantId').val(), // -
        amount : $('#amount').val(), //
        currency : $('#currency').val(), //
        ordernumber : $('#ordernumber').val(), //
        email : $('#email').val(), //email
        firstname : $('#firstname').val(), //
        middlename : $('#middlename').val(),
        lastname : $('#lastname').val() //
        comment : $('#comment').val() //
  };
```

После завершения оплаты с помощью Google Pay сервер возвращает в ответ номер заказа и статус. Для обработки ответа необходимо добавить соответствующий код в эту функцию ***processPayment:***

```
$.post("/pay/tokenpay_widget_gp.cfm", JSON.stringify(data)).then(function (result) {
    //
    //    {"order":{"ordernumber":"2019.03.11-664","orderstate":"Approved"}}
});
```

Если по каким-то причинам оплата заказа прошла неуспешно, то сервис вернет сообщения об ошибках (ненулевые значения параметров firstcode, secondcode).

Скрипт использует Google Pay API, который поддерживается:

- на всех мобильных устройствах, независимо от операционной системы;
- в браузерах Google Chrome, Mozilla Firefox, Apple Safari, Microsoft Edge, Opera или UCWeb UC Browser.

В Google Pay отсутствует возможность привязать специальную тестовую карту, поэтому при тестировании будет отображаться реальная карта. Однако, в тестовой среде Google эта карта будет подменяться на тестовую и в скрипт будут возвращаться данные тестовой карты. Это позволяет использовать привязанную реальную карту без опасений, что с нее будут списаны средства.

Пример скрипта

Ниже приведен пример скрипта виджета, который может быть размещен на платежной странице интернет-магазина.

```
<script type="text/javascript">
const baseRequest = {
  apiVersion: 2,
  apiVersionMinor: 0
};

const allowedCardNetworks = ["MASTERCARD", "VISA"];
const allowedCardAuthMethods = ["PAN_ONLY", "CRYPTOGRAM_3DS"];

const tokenizationSpecification = {
  type: 'PAYMENT_GATEWAY',
  parameters: {
    'gateway': 'belassist',
    'gatewayMerchantId': '02510116604241796260'
  }
}

const baseCardPaymentMethod = {
  type: 'CARD',
  parameters: {
    allowedAuthMethods: allowedCardAuthMethods,
    allowedCardNetworks: allowedCardNetworks,
    billingAddressRequired: true,
    billingAddressParameters: {"format": "MIN"}
   }
}

const cardPaymentMethod = Object.assign(
  {},
  baseCardPaymentMethod,
  {
    tokenizationSpecification: tokenizationSpecification
  }
);

let paymentsClient = null;

function getGoogleIsReadyToPayRequest() {
  return Object.assign(
      {},
      baseRequest,
      {
        allowedPaymentMethods: [baseCardPaymentMethod]
      }
  );
}

/**
* Configure support for the Google Pay API
*
* @see {@link https://developers.google.com/pay/api/web/reference/object#PaymentDataRequest|PaymentDataRequest}
* @returns {object} PaymentDataRequest fields
*/
function getGooglePaymentDataRequest() {
  const paymentDataRequest = Object.assign({}, baseRequest);
  paymentDataRequest.allowedPaymentMethods = [cardPaymentMethod];
  paymentDataRequest.transactionInfo = getGoogleTransactionInfo();
  paymentDataRequest.merchantInfo = {
    // @todo a merchant ID is available for a production environment after approval by Google
```

```
    // See {@link https://developers.google.com/pay/api/web/guides/test-and-deploy/integration-
checklist|Integration checklist}
    merchantId: '16590966430175452581',
    merchantOrigin: 'www.assist.ru',
    merchantName: 'ASSIST Merchant'
  };
  return paymentDataRequest;
}

/**
* Return an active PaymentsClient or initialize
*
* @see {@link https://developers.google.com/pay/api/web/reference/client#PaymentsClient|PaymentsClient
constructor}
* @returns {google.payments.api.PaymentsClient} Google Pay API client
*/
function getGooglePaymentsClient() {
  if ( paymentsClient === null ) {
    paymentsClient = new google.payments.api.PaymentsClient({environment: 'TEST'});
  }
  return paymentsClient;
}

/**
* Initialize Google PaymentsClient after Google-hosted JavaScript has loaded
*
* Display a Google Pay payment button after confirmation of the viewer's
* ability to pay.
*/
function onGooglePayLoaded() {
  const paymentsClient = getGooglePaymentsClient();
  paymentsClient.isReadyToPay(getGoogleIsReadyToPayRequest())
      .then(function(response) {
        if (response.result) {
          addGooglePayButton();
          // @todo prefetch payment data to improve performance after confirming site functionality
           prefetchGooglePaymentData();
        }
      })
      .catch(function(err) {
        // show error in developer console for debugging
        console.error(err);
      });
}

/**
* Add a Google Pay purchase button alongside an existing checkout button
*
* @see {@link https://developers.google.com/pay/api/web/reference/object#ButtonOptions|Button options}
* @see {@link https://developers.google.com/pay/api/web/guides/brand-guidelines|Google Pay brand guidelines}
*/
function addGooglePayButton() {
  const paymentsClient = getGooglePaymentsClient();
  const button =
      paymentsClient.createButton({onClick: onGooglePaymentButtonClicked, buttonColor:'black',
buttonType:'long'});
      document.getElementById('container').appendChild(button);
      document.getElementById('container').style.display = 'block';
}

/**
* Provide Google Pay API with a payment amount, currency, and amount status
*
* @see {@link https://developers.google.com/pay/api/web/reference/object#TransactionInfo|TransactionInfo}
* @returns {object} transaction info, suitable for use as transactionInfo property of PaymentDataRequest
*/

function getGoogleTransactionInfo() {
  return {
    currencyCode: $('#currency').val(),
    totalPriceStatus: 'FINAL',
```

```
    // set to cart total
    totalPrice: $('#amount').val()
  };
}

/**
* g payment data to improve performance
*
* @see {@link https://developers.google.com/pay/api/web/reference
/client#prefetchPaymentData|prefetchPaymentData()}
*/
function prefetchGooglePaymentData() {
  const paymentDataRequest = getGooglePaymentDataRequest();
  // transactionInfo must be set but does not affect cache
  paymentDataRequest.transactionInfo = {
    totalPriceStatus: 'NOT_CURRENTLY_KNOWN',
    currencyCode: $('#currency').val()
  };
  const paymentsClient = getGooglePaymentsClient();
  paymentsClient.prefetchPaymentData(paymentDataRequest);
}

/**
* Show Google Pay payment sheet when Google Pay payment button is clicked
*/
function onGooglePaymentButtonClicked() {
  const paymentDataRequest = getGooglePaymentDataRequest();
  paymentDataRequest.transactionInfo = getGoogleTransactionInfo();

  const paymentsClient = getGooglePaymentsClient();
  paymentsClient.loadPaymentData(paymentDataRequest)
      .then(function(paymentData) {
        // handle the response
        processPayment(paymentData);
      })
      .catch(function(err) {
        // show error in developer console for debugging
        console.error(err);
      });
}

/**
* Process payment data returned by the Google Pay API
*
* @param {object} paymentData response from Google Pay API after user approves payment
* @see {@link https://developers.google.com/pay/api/web/reference/object#PaymentData|PaymentData object
reference}
*/
function processPayment(paymentData) {
  var data = {
          paymentData : paymentData,
          merchant_id : $('#merchantId').val(),
          amount : $('#amount').val(),
          currency : $('#currency').val(),
          ordernumber : $('#ordernumber').val(),
          email : $('#email').val(),
          firstname : $('#firstname').val(),
          middlename : $('#middlename').val(),
          lastname : $('#lastname').val(),
          comment : $('#comment').val()
      };
// For debug
console.log(data);

  $.post("/pay/tokenpay_widget_gp.cfm", JSON.stringify(data)).then(function (result) {
      // For debug
          console.log(result);
          // Example result:
          // {"order":{"ordernumber":"2019.03.11-664","orderstate":"Approved"}}

  });
```

```
}
</script>
<script async src="https://pay.google.com/gp/p/js/pay.js" onload="onGooglePayLoaded()"></script>
```

Наверх