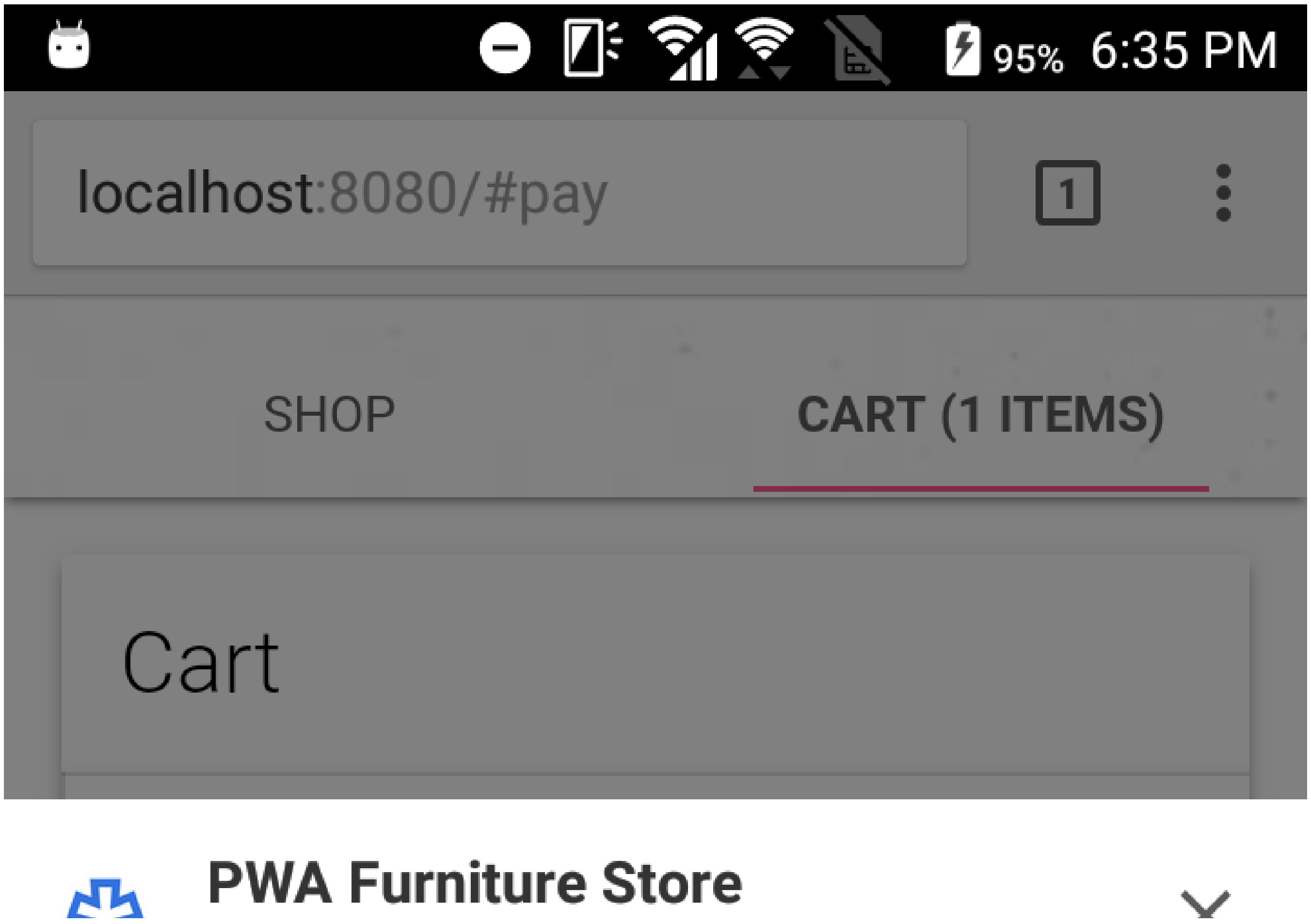
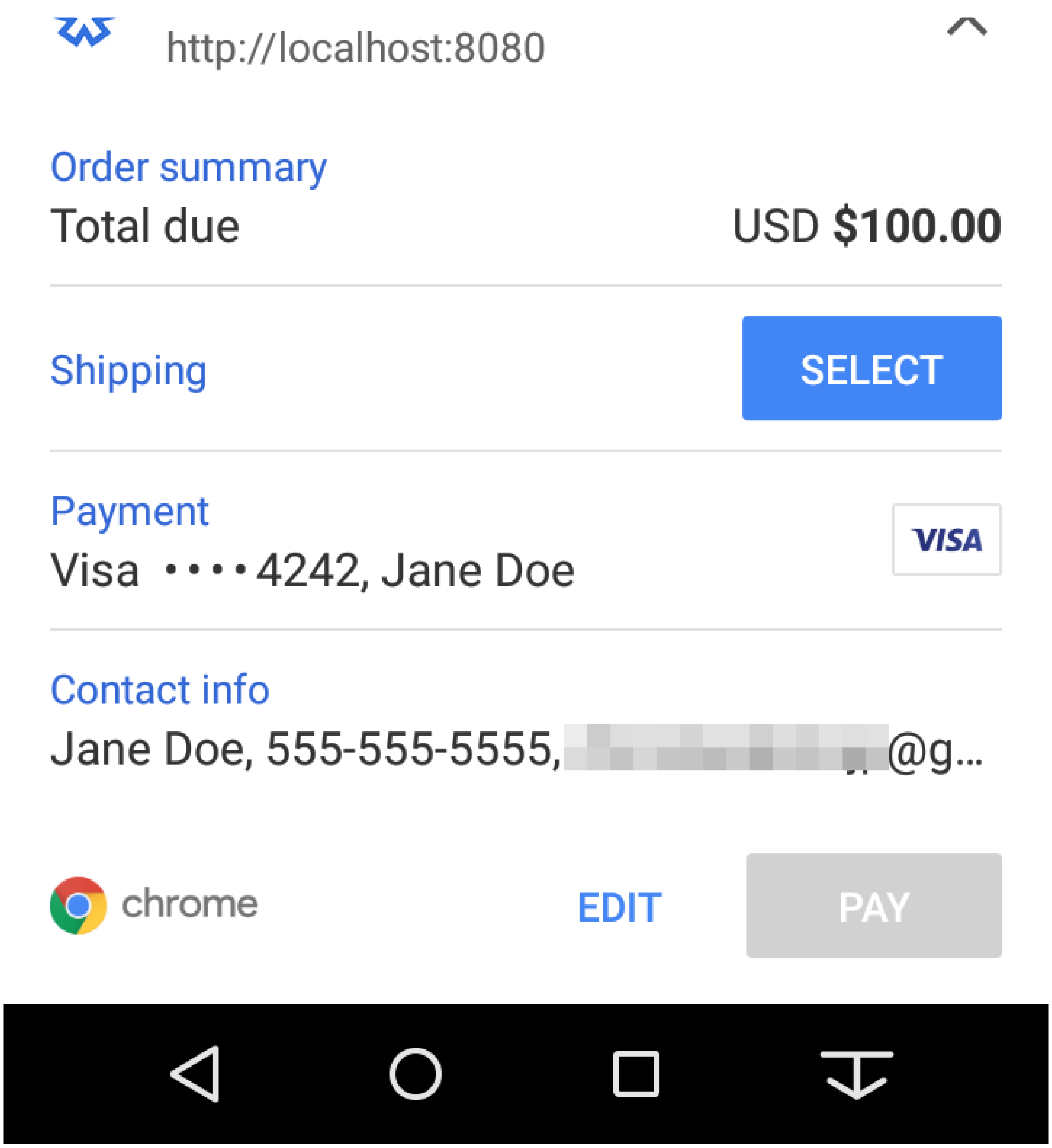
Don't forget the **Chrome Dev Summit**, starting Monday at 10:00am (Pacific) and streaming live on YouTube. Schedule. (https://developer.chrome.com/devsummit/schedule)

Frictionless payment with Payment Request API

Overview

Welcome to "Frictionless payment with Payment Request API" codelab. In this codelab, you will learn how to implement Payment Request API (https://www.w3.org/TR/payment-request/) onto an existing e-commerce website. Let's get started.





http://localhost:8080

Order summary
Total due USD **$100.00**

Shipping SELECT

Payment
Visa ••••4242, Jane Doe

Contact info
Jane Doe, 555-555-5555, [redacted]@g...

chrome EDIT PAY

What you will need

- USB cable to connect an Android device to your computer
- Computer with terminal/shell access
- Connection to the internet

- Chrome for Android version 56 or greater
- A text editor

Get set up

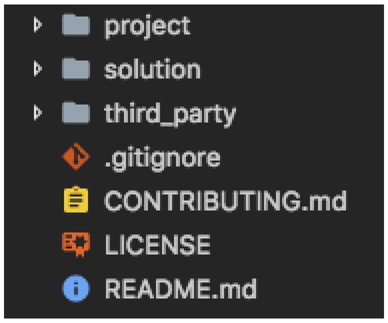
Clone the E-Commerce lab repository's "payment-request-api" branch with Git using the following command:

```
$ git clone -b payment-request-api https://github.com/google-developer-training/
```

Note: If you do not use Git, then download the repo (https://github.com/google-developer-training/pwa-ecommerce-demo/archive/payment-request-api.zip) from GitHub.

The repository consists of a **project** folder, a **solution** folder and others.

- The **project** folder is where you will build the app.
- The **solution** folder is an example solution you will get by following this codelab.



Navigate into the cloned repo and open the **project** folder.

```
$ cd pwa-ecommerce-demo/project
```

Then run **`npm install`** in the command line at the **project** directory.

```
$ npm install
```

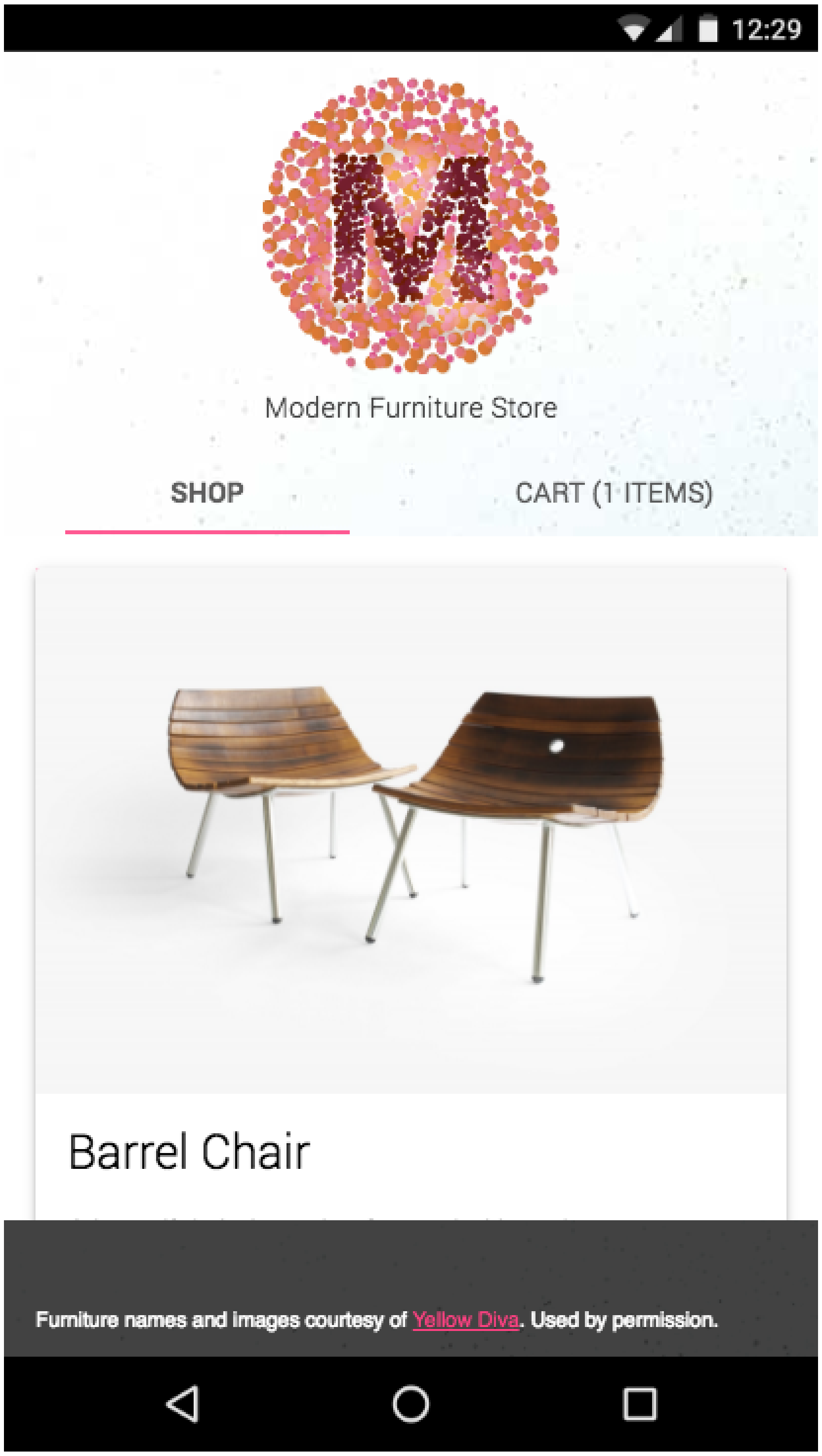
This command should create **node_modules** directory and install all required node modules to your project. It takes a while.

Run `npm run serve` in the same directory to build and run the server in **dist**. Any changes you make to JavaScript files in this codelab will trigger rebuilding the app as long as you keep the command running.

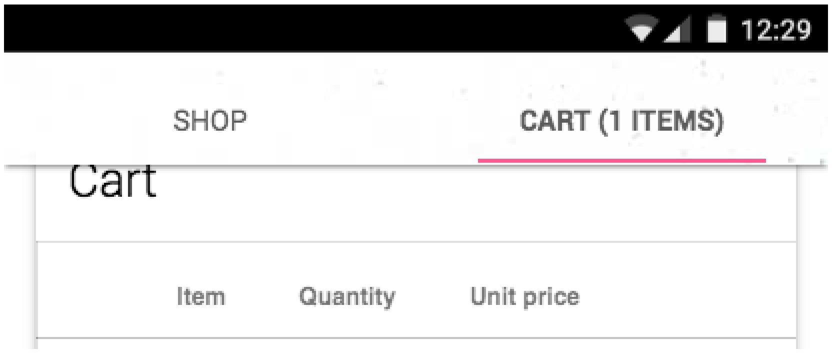
```
$ npm run serve
```

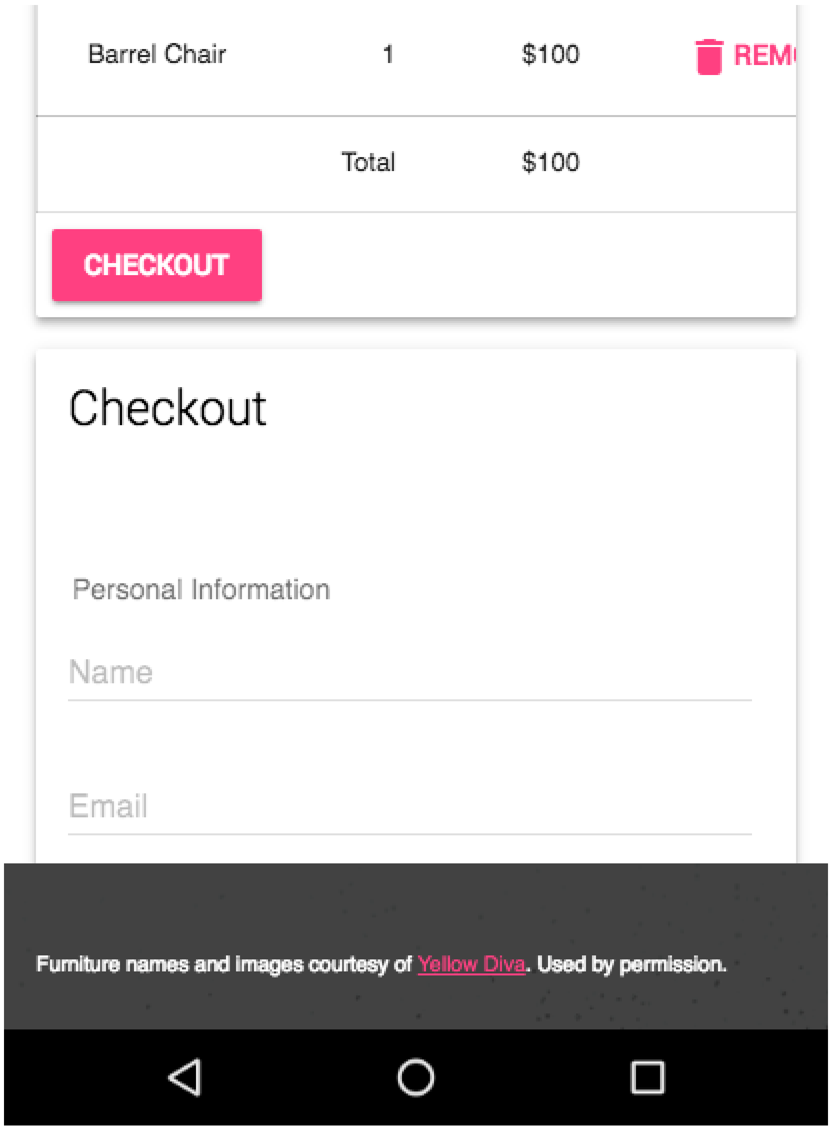
Open your browser and navigate to **http://localhost:8080** to see the initial state of the app working.

At this point, you should be able to see the e-commerce app running. Within this app, you can add items to the cart and proceed to the checkout form. Play with it for a while to understand how the original website works.



Furniture names and images courtesy of Yellow Diva. Used by permission.





Barrel Chair	1	$100	REM
	Total	$100	

Checkout

Personal Information

Furniture names and images courtesy of Yellow Diva. Used by permission.

Note: The information you enter here won't be posted anywhere other than your local server, but you should use fake information. However, since credit card information requires validation, you can use following fake number so it can accept a random CVC: **4242 4242 4242 4242**

From here, you will be implementing the Payment Request API.

The Payment Request API is not yet supported on desktop as of Chrome 58, so you will need an Android device with Chrome installed to test the code. Follow the instructions in the Access Local Servers (https://developers.google.com/web/tools/chrome-devtools/remote-debugging/local-server) article to set up port forwarding on your Android device. This lets you host the e-commerce app on your phone.

Create a PaymentRequest

Detect feature availability

First, let's add add a feature detection for the Payment Request API. And if it's available, let a user process payment with it.

Replace "TODO PAY-3.1" in **app/scripts/modules/app.js** with the following code and remove the dummy conditional of `if (false) {` to add `PaymentRequest` feature detection:

app.js

```
if (window.PaymentRequest) {
```

Explanation

The feature detection is as simple as examining if `window.PaymentRequest` returns `undefined` or not.

Create a PaymentRequest

Create a Payment Request object using the `PaymentRequest` constructor.

Replace "TODO PAY-3.2" in **app/scripts/modules/payment-api.js** with the following code and initialize `PaymentRequest`:

payment-api.js

```
let request = new PaymentRequest(supportedInstruments, details, paymentOptions);
```

Explanation

The constructor takes three parameters.

`supportedInstruments`: The first argument is a required set of data about supported payment methods. This can include basic credit cards as well as payment processors like Android Pay. We'll use only basic credit cards in this codelab.

`details`: The second argument is required information about the transaction. This must include the information to display the total to the user (i.e., a label, currency, and value amount), but it can also include a breakdown of items in the transaction.

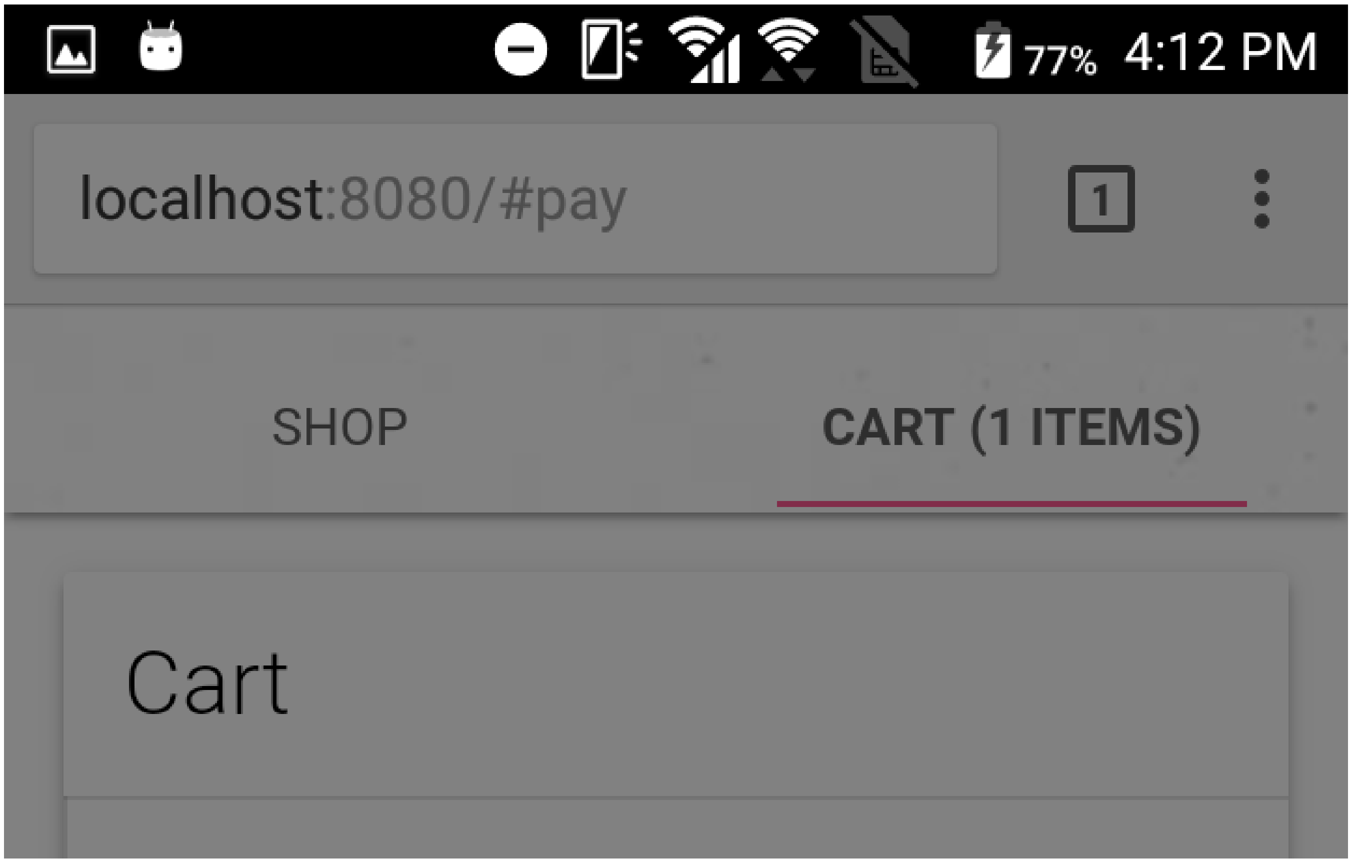
`paymentOptions`: The third argument is an optional parameter for things like shipping. This allows you to require additional information from the user, like payer name, phone, email, and shipping information.

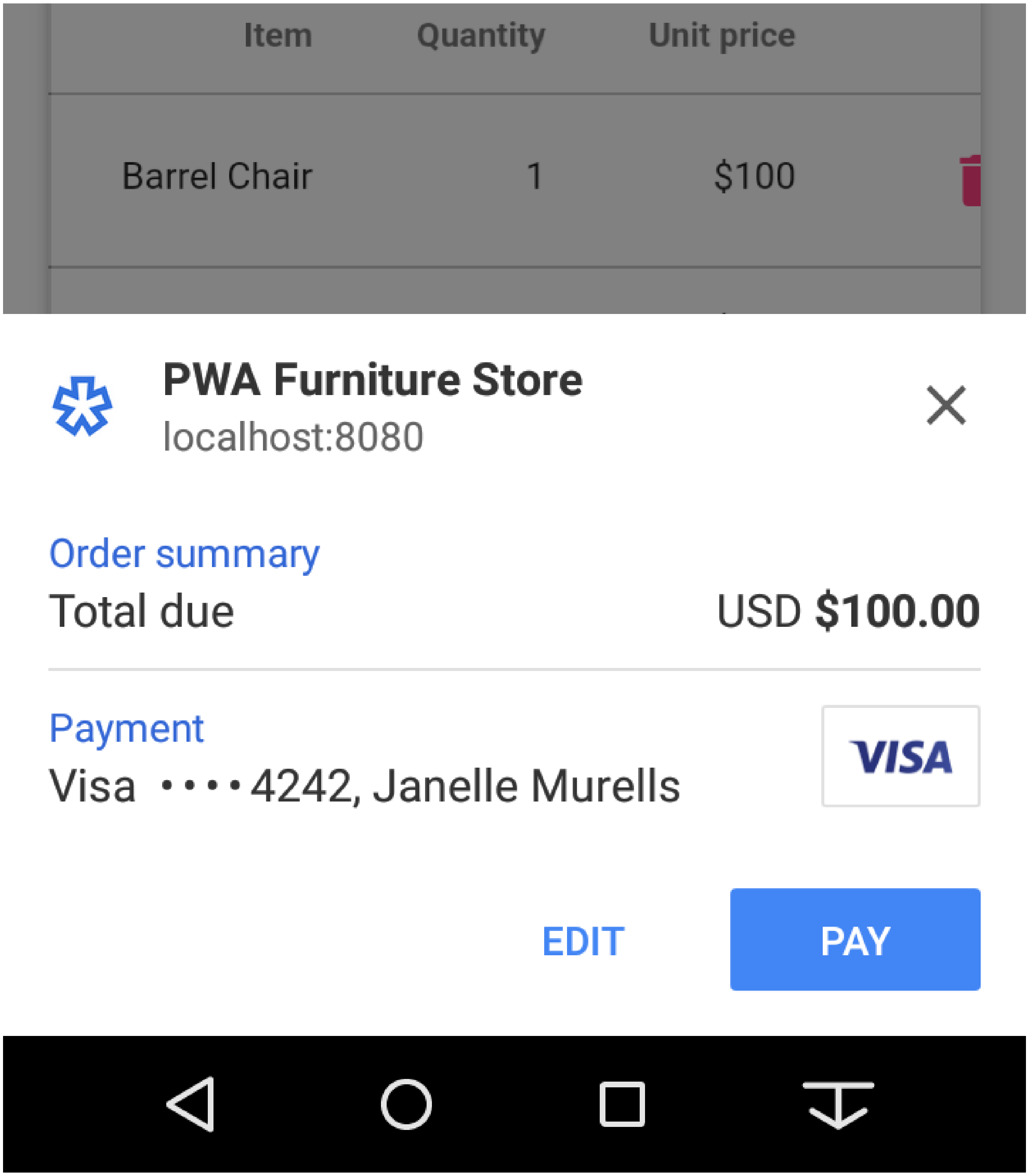
Try it out

You should now be able to try the Payment Request API. If you are not running your server, **`npm run serve`** and try it using your Android device.

```
$ npm run serve
```

The `PaymentRequest` UI displays when you click **Checkout**.





Beware, this is only the first step and there is more work to be done. Let's continue...

Add payment methods

At this point, most of the arguments are empty arrays or objects. Let's configure the payment method (`supportedInstruments`) with proper values.

Replace "TODO PAY-4" and its following JSON block in **scripts/modules/payment-api.js** with this:

payment-api.js

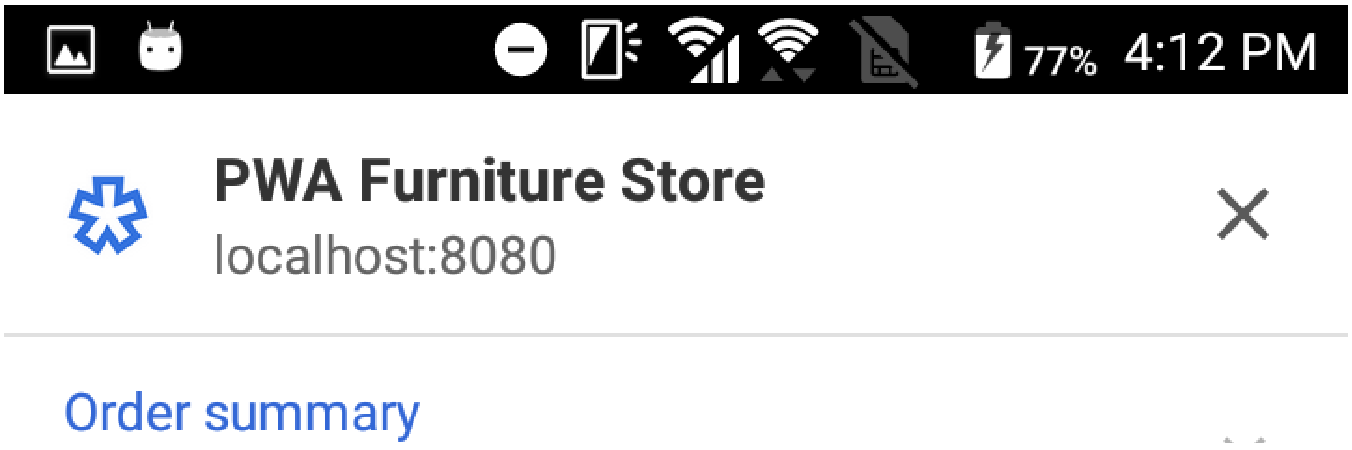
```
{
  supportedMethods: ['basic-card'],
  data: {
    supportedNetworks: ['visa', 'mastercard', 'amex',
      'jcb', 'diners', 'discover', 'mir', 'unionpay']
  }
}
```

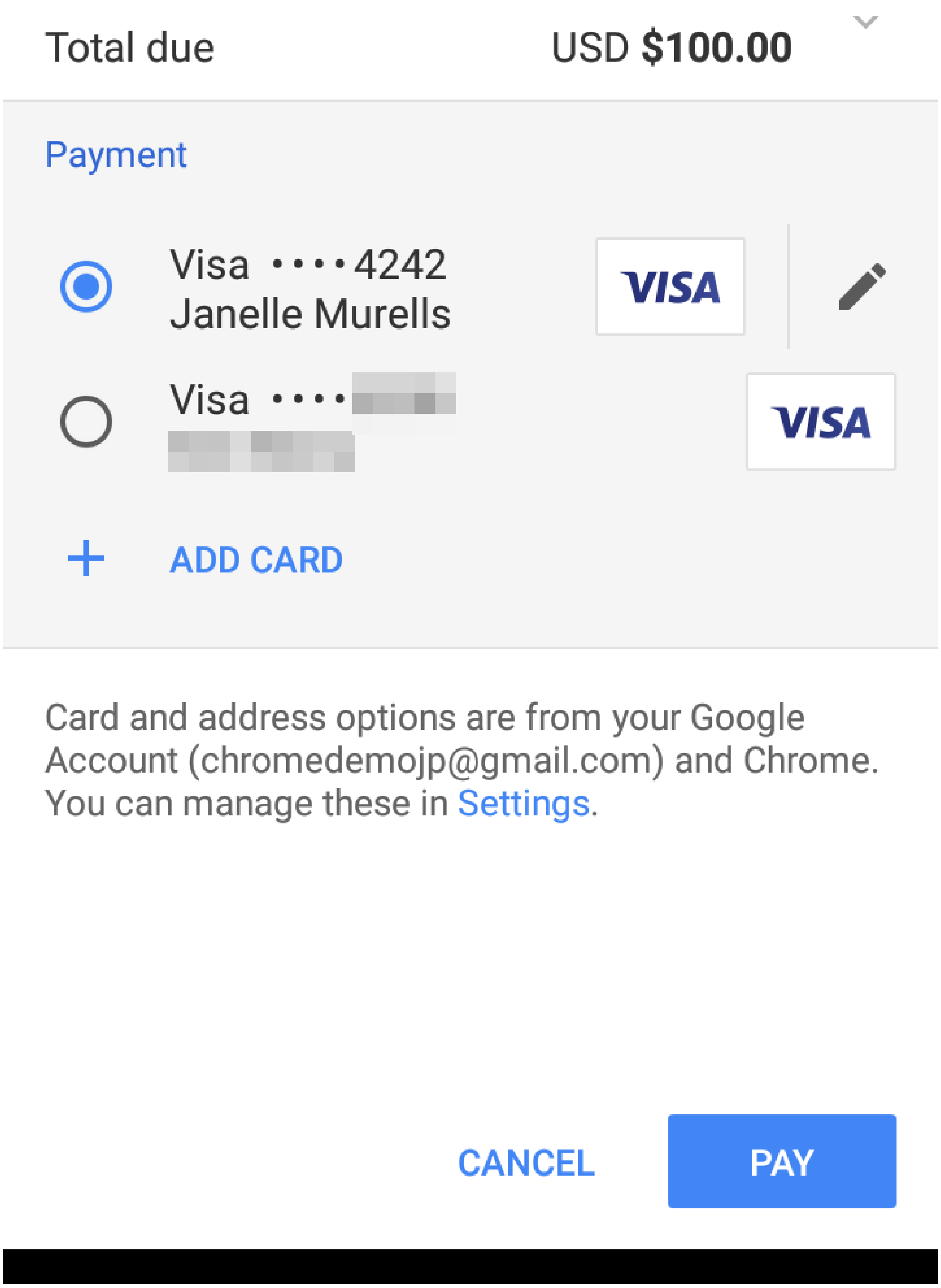
Explanation

The first argument of the `PaymentRequest` constructor takes a list of supported payment methods as JSON objects.

`supportedMethods` takes a list of supported method names as an array. Supported methods can be `basic-card` or a URL representing a payment app. These are defined in the Payment Method Identifiers (https://www.w3.org/TR/payment-method-id/) specification.

In the case of `basic-card`, `supportedNetworks` under `data` takes a list of supported credit card brands as defined at Card Network Identifiers Approved for use with Payment Request API (https://www.w3.org/Payments/card-network-ids). This will filter and show only the credit cards available for the user in the Payment Request UI.





Total due USD **$100.00**

Payment

[✓ Visa ••••4242
Janelle Murells]

Visa •••• [redacted]
[redacted]

Card and address options are from your Google Account (chromedemojp@gmail.com) and Chrome. You can manage these in Settings.

Add payment details

Next, let's provide information about items a user is trying to purchase.

Define the details object

The second argument of the `PaymentRequest` constructor takes details about the purchase. It takes a list of items to display and the total price information.

This portion is already implemented in the `buildPaymentDetails()` function in **app/scripts/modules/payment-api.js**. You don't have to do anything at this time, but see what's happening here.

payment-api.js

```
let details = {
  displayItems: displayItems,
  total: {
    label: 'Total due',
    amount: {currency: 'USD', value: String(total)}
  }
  // TODO PAY-7.2 - allow shipping options
};
return details;
```

Explanation

A required `total` parameter consists of a label, currency and total amount to be charged.

An optional `displayItems` parameter indicates how the final amount was calculated.

The `displayItems` parameter is not intended to be a line-item list, but is rather a summary of the order's major components: subtotal, discounts, tax, shipping costs, etc. Let's define it in the next section.

Define the display items

The **`displayItems`** variable should be defined based on items added to the cart.

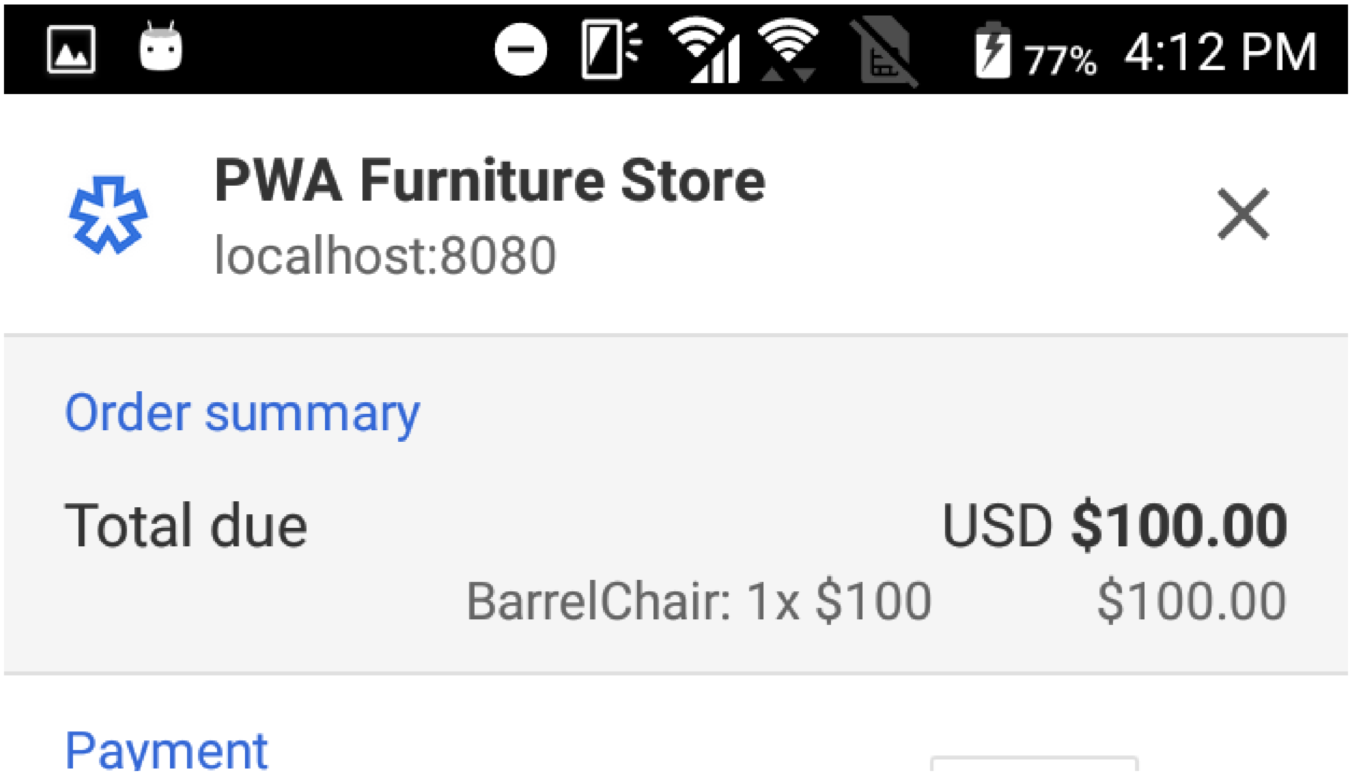
Replace "TODO PAY-5" in **app/scripts/modules/payment-api.js** with the following code and remove the existing **`let displayItems = [];`**:

payment-api.js

```
let displayItems = cart.cart.map(item => {
  return {
    label: `${item.sku}: ${item.quantity}x $${item.price}`,
    amount: {currency: 'USD', value: String(item.total)}
  };
});
```

Explanation

The payment UI should look like this. Try expanding "Order summary":



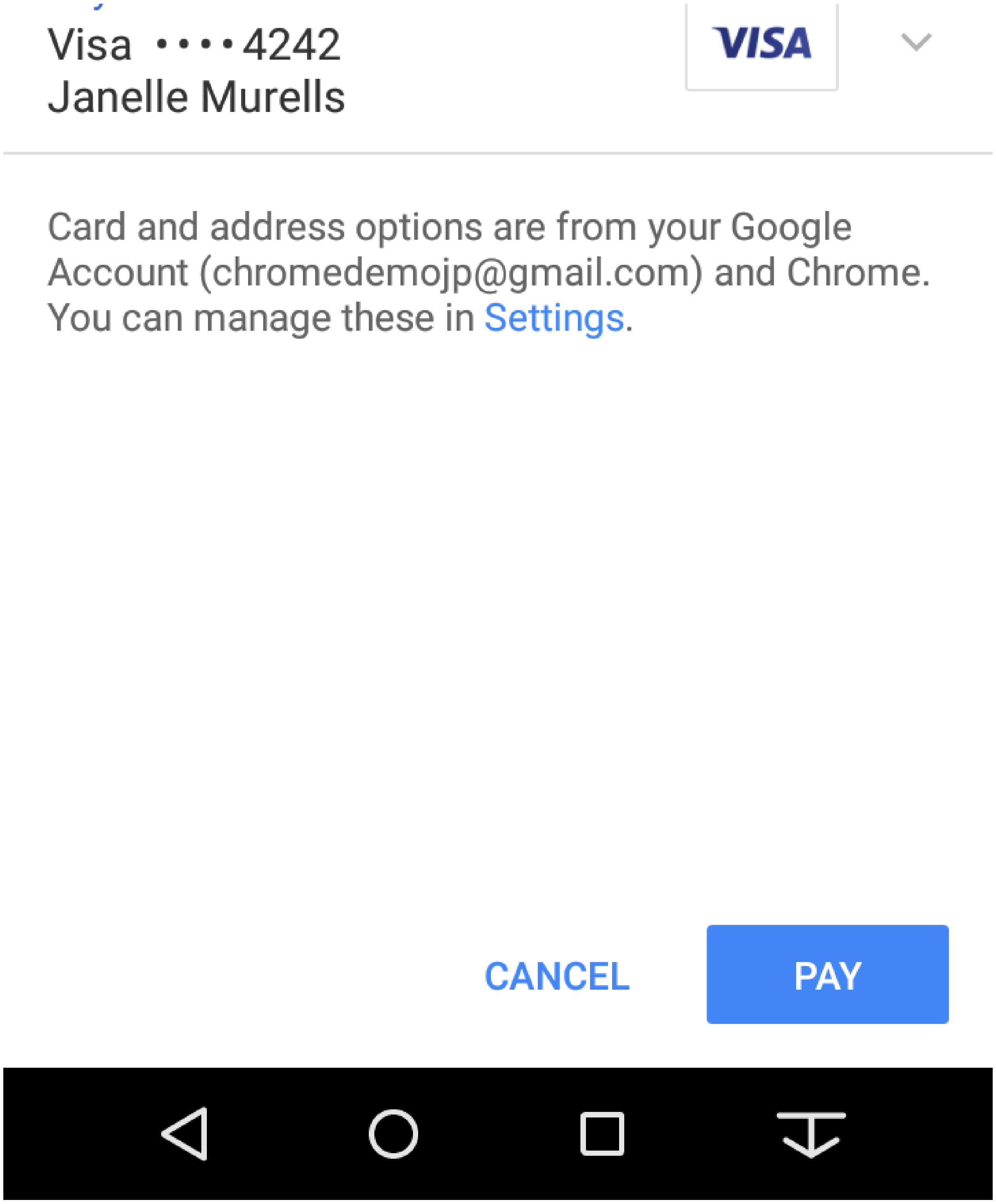
77% 4:12 PM

PWA Furniture Store
localhost:8080

Order summary

Total due USD **$100.00**
BarrelChair: 1x $100 $100.00

Payment


Visa ••••4242
Janelle Murells

Card and address options are from your Google Account (chromedemojp@gmail.com) and Chrome. You can manage these in Settings.

Notice that the display items are present in the "Order summary" row. We gave each item a `label` and `amount`. `label` is a display label information of the item. `amount` is an object that constructs price information for the item.

Complete the PaymentRequest

You've put minimum required options to run a Payment Request. Let's allow a user to complete the payment.

Replace "TODO PAY-6" and the existing `return request.show();` in **app/scripts/modules/payment-api.js** with the following code:

payment-api.js

```
    return request.show()
      .then(r => {
        // The UI will show a spinner to the user until
        // `request.complete()` is called.
        response = r;
        let data = r.toJSON();
        console.log(data);
        return data;
      })
      .then(data => {
        return sendToServer(data);
      })
      .then(() => {
        response.complete('success');
        return response;
      })
      .catch(e => {
        if (response) {
          console.error(e);
          response.complete('fail');
        } else if (e.code !== e.ABORT_ERR) {
          console.error(e);
          throw e;
        } else {
          return null;
        }
      });
```

Explanation

The `PaymentRequest` interface is activated by calling its `show()` method. This method invokes a native UI that allows the user to examine the details of the purchase, add or change information, and pay. A `Promise` (indicated by its `then()` method and callback function) that resolves will be returned when the user accepts or rejects the payment request.

Calling `toJSON()` serializes the response object. You can then POST it to a server to process the payment. This portion differs depending on what payment processor / payment gateway you are using.

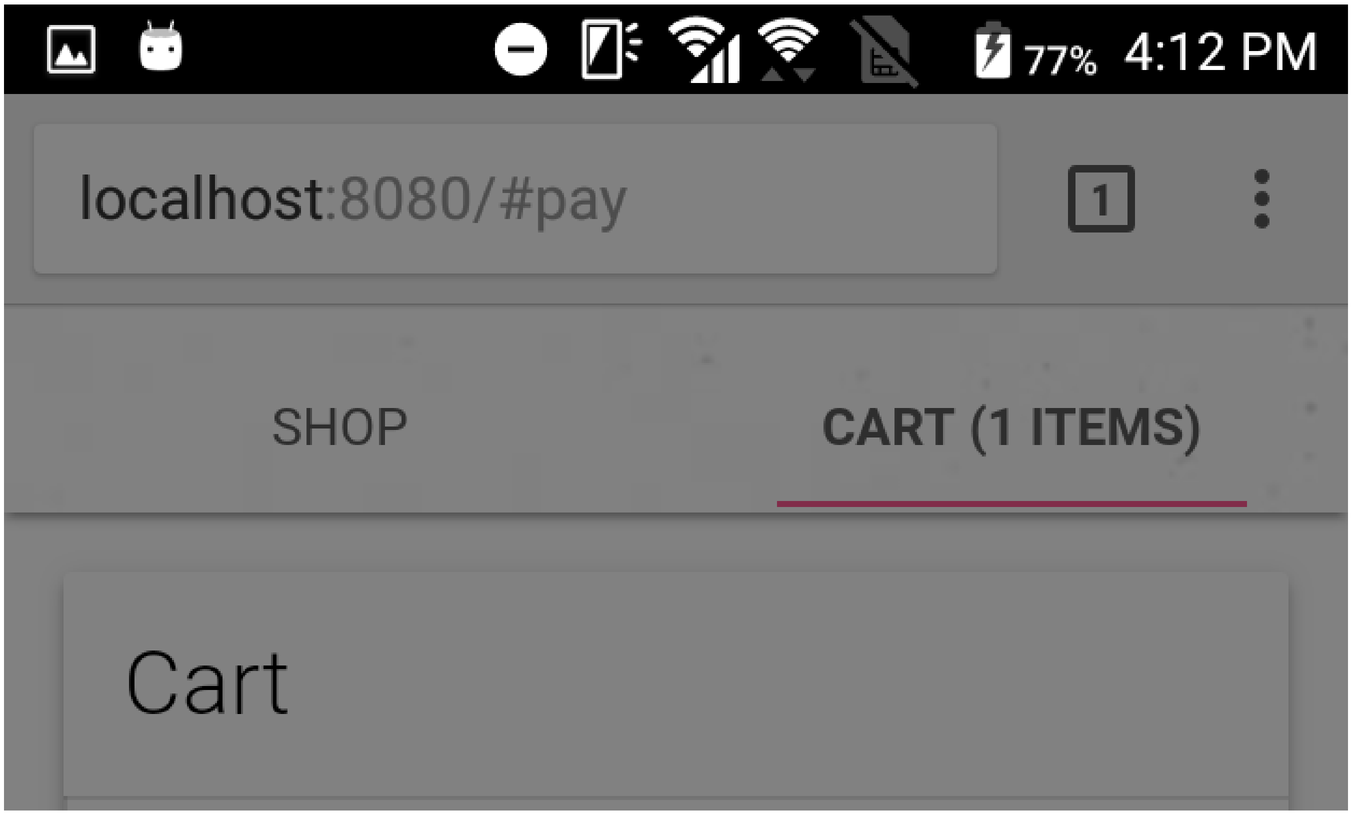
Once the server returns a response, call `complete()` to tell the user if processing the payment was successful or not by passing it `success` or `fail`.

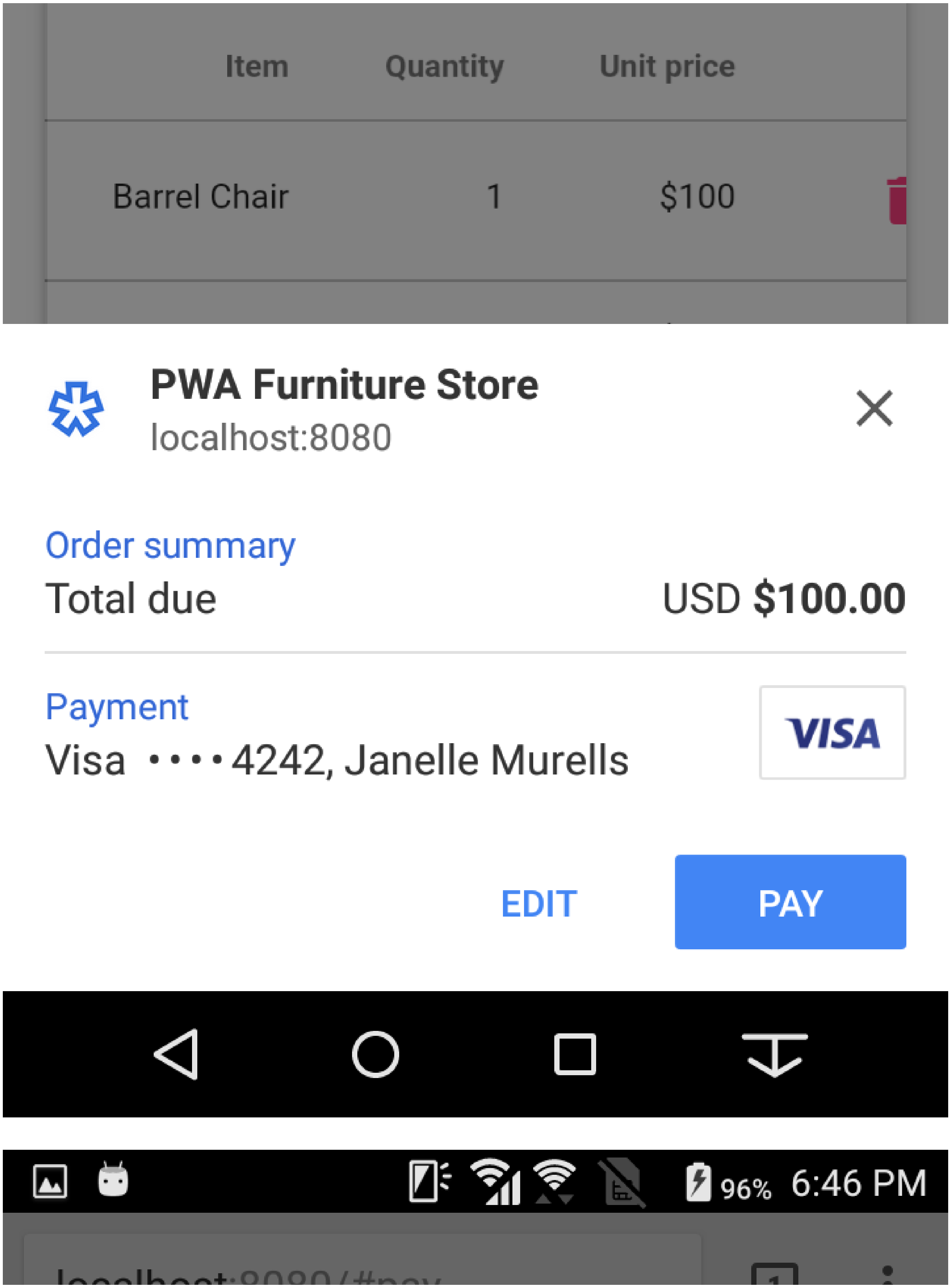
Try out the app

Awesome! Now you have completed implementing the basic Payment Request API features in your app. If you are not running your server, `npm run serve` and try it using your Android device.

```
$ npm run serve
```

The `PaymentRequest` UI displays when you click **Checkout**.



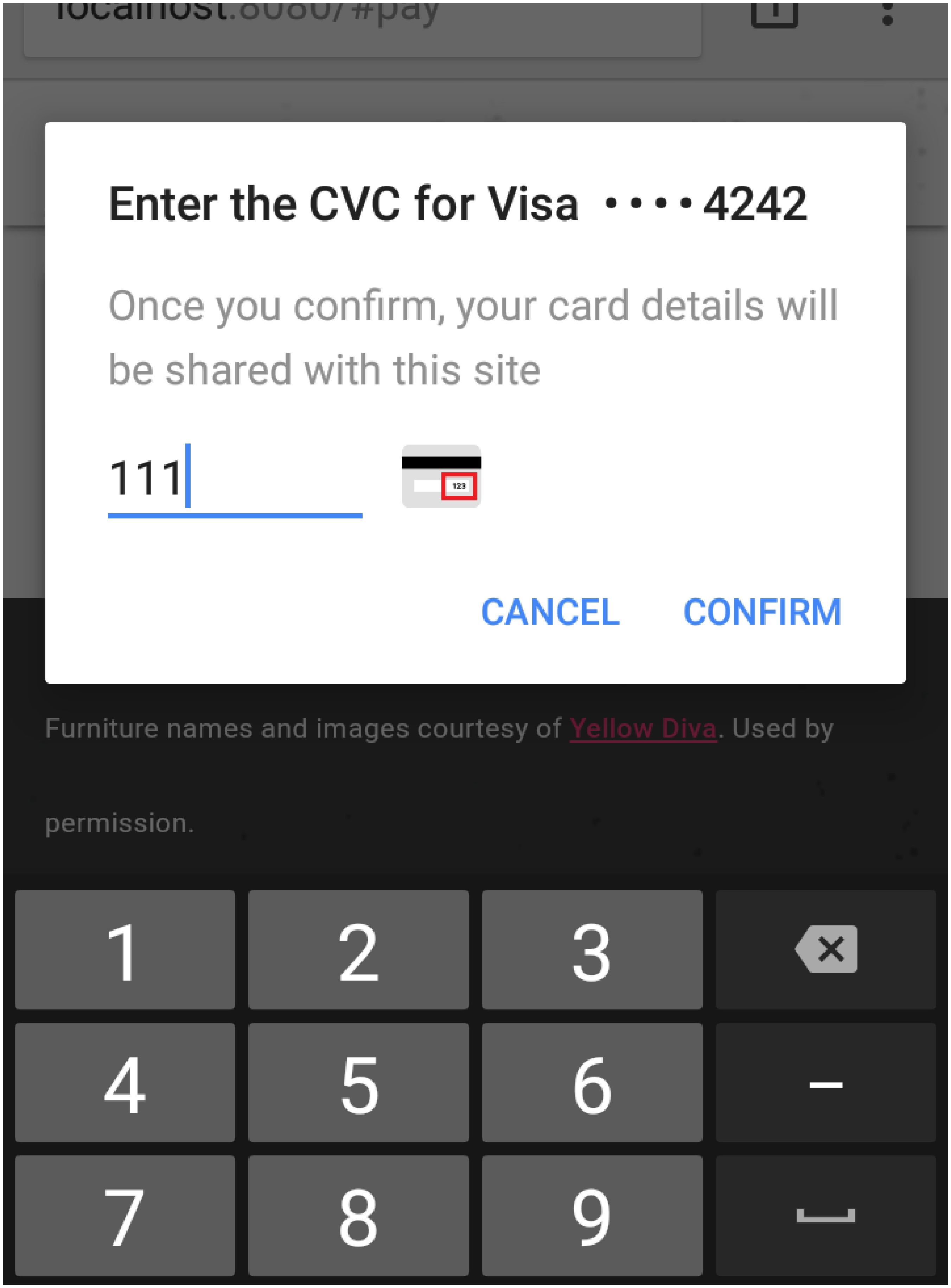


Item	Quantity	Unit price
Barrel Chair	1	$100

PWA Furniture Store
localhost:8080

Order summary
Total due USD **$100.00**

Payment
Visa ••••4242, Janelle Murells

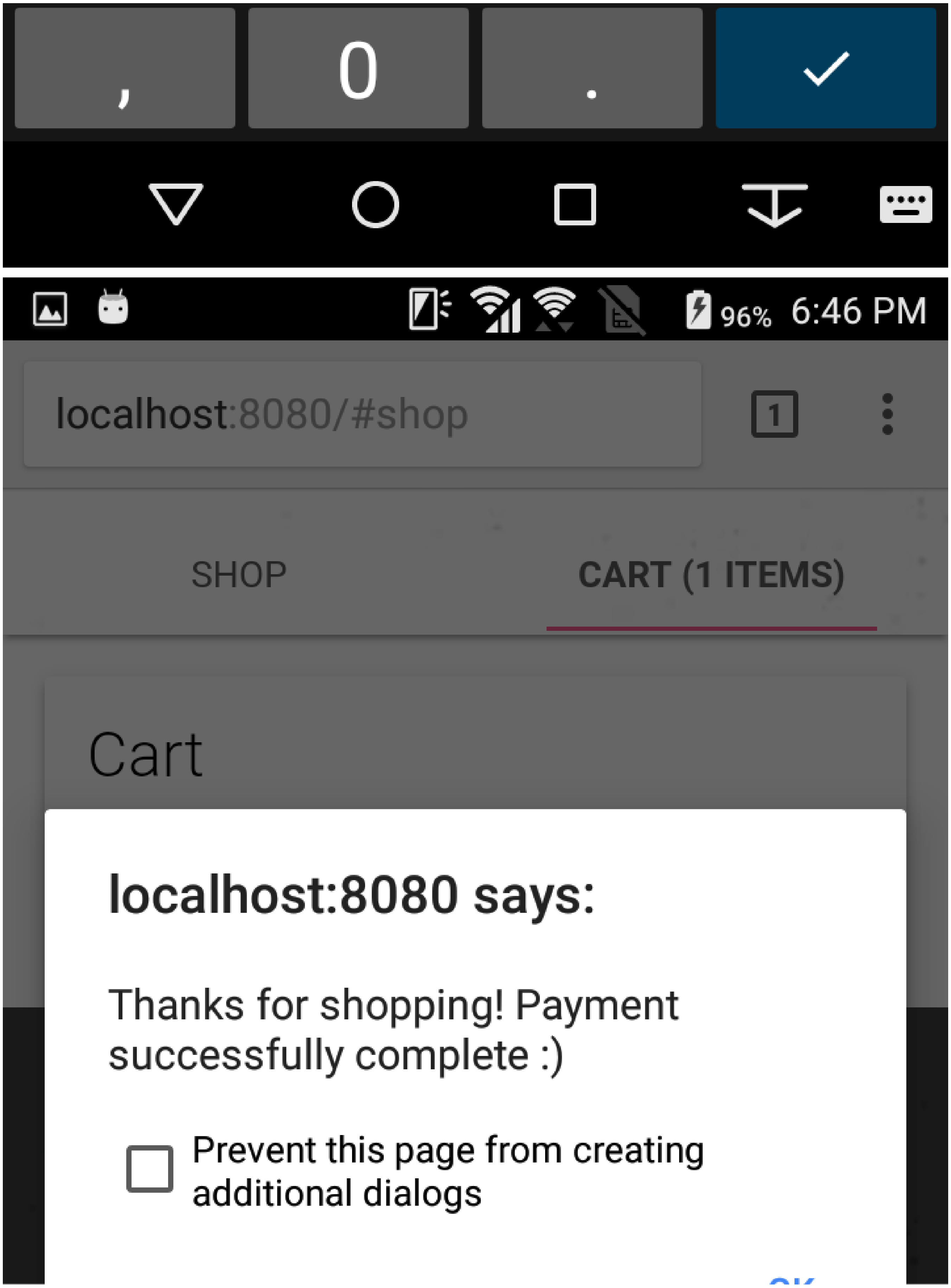


Enter the CVC for Visa ••••4242

Once you confirm, your card details will be shared with this site

111

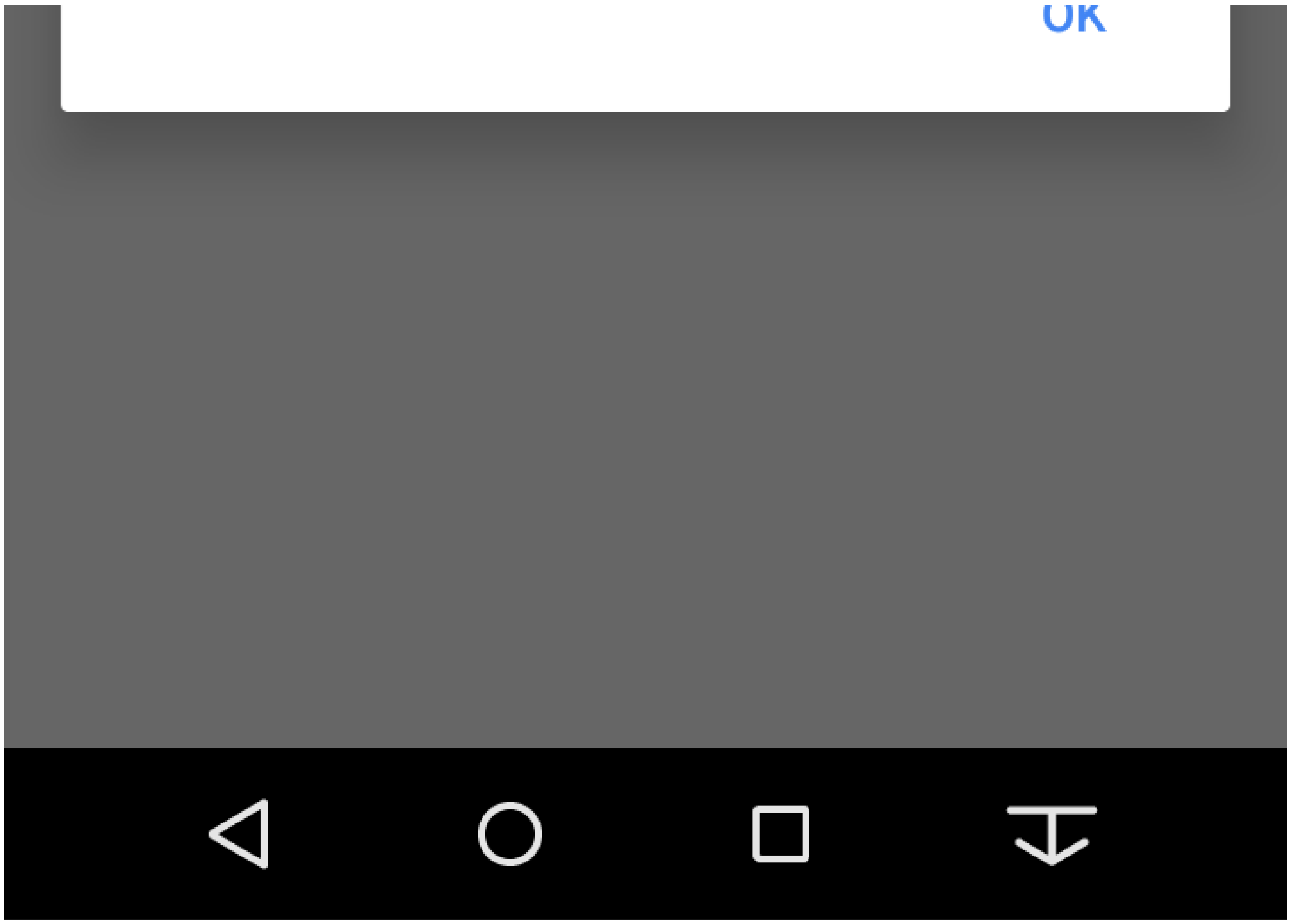
Furniture names and images courtesy of Yellow Diva. Used by permission.



localhost:8080 says:

Thanks for shopping! Payment successfully complete :)

Prevent this page from creating additional dialogs



Allow shipping options

So far you've learned how to integrate the Payment Request API when it doesn't involve shipping items. Moving forward you will learn how to collect shipping information and options from the user.

Define the shipping options

When you want to collect the user's address information in order to ship items, add `requestShipping: true` in the third property of the `PaymentRequest` constructor.

Replace "TODO PAY-7.1" in **app/scripts/modules/payment-api.js** with the following code:

payment-api.js

```
requestShipping: true,
```

You also need to provide list of shipping options.

Replace "TODO PAY-7.2" in **app/scripts/modules/payment-api.js** with the following code:

payment-api.js

```
,
shippingOptions: displayedShippingOptions
```

Luckily `SHIPPING_OPTIONS` is predefined in the **app/scripts/modules/payment-api.js**; you can parse it and construct the `displayShippingOptions` object from it.

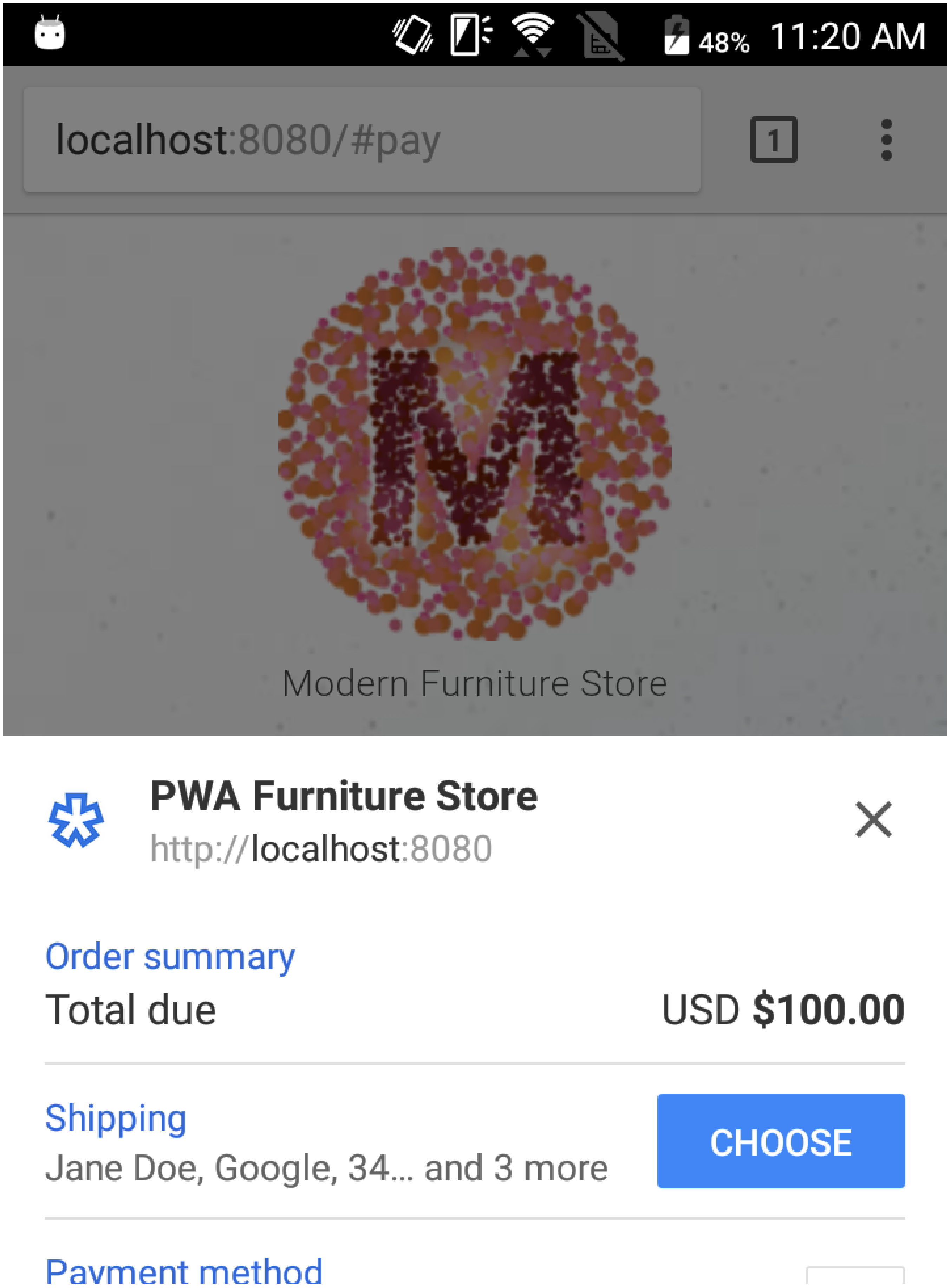
Replace "TODO PAY-7.3" in **app/scripts/modules/payment-api.js** with the following code:

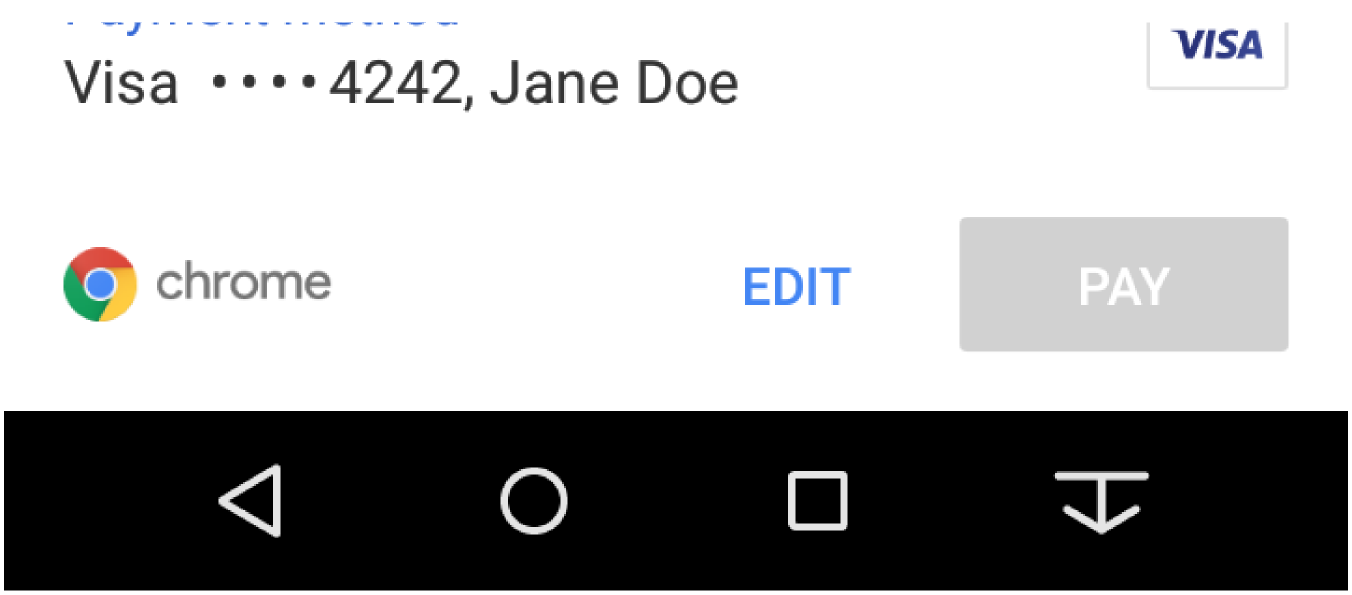
payment-api.js

```
let displayedShippingOptions = [];
if (shippingOptions.length > 0) {
  let selectedOption = shippingOptions.find(option => {
    return option.id === shippingOptionId;
  });
  displayedShippingOptions = shippingOptions.map(option => {
    return {
      id: option.id,
      label: option.label,
      amount: {currency: 'USD', value: String(option.price)},
      selected: option.id === shippingOptionId
    };
  });
  if (selectedOption) total += selectedOption.price;
}
```

Explanation

`id` is a unique identifier of the shipping option item. `label` is a displayed label of the item. `amount` is an object that constructs price information for the item. `selected` is a boolean that indicates if the item is selected.


48% 11:20 AM
localhost:8080/#pay
Modern Furniture Store
PWA Furniture Store
http://localhost:8080
Order summary
Total due
USD $100.00
Shipping
Jane Doe, Google, 34... and 3 more
CHOOSE
Payment method



Notice that these changes add a section to the Payment Request UI, "Shipping". But beware, selecting shipping address will cause UI to freeze and timeout. To resolve this, you will need to handle `shippingaddresschange` event in the next section.

The address information available here is retrieved from the browser's autofill information. Depending on the user's browser status, users will get address information pre-filled without typing any text. They can also add a new entry on the fly.

Add event handlers

What if a user specifies a shipping address that's outside of your target countries and not deliverable? How do you charge a user when the user changes a shipping option? The answer is to receive events upon the user's making changes and update with relevant information.

Add `shippingaddresschange` event listener

When the user changes a shipping address, you will receive the `shippingaddresschange` event.

Replace "TODO PAY-8.1" in **app/scripts/modules/payment-api.js** with the following code:

payment-api.js

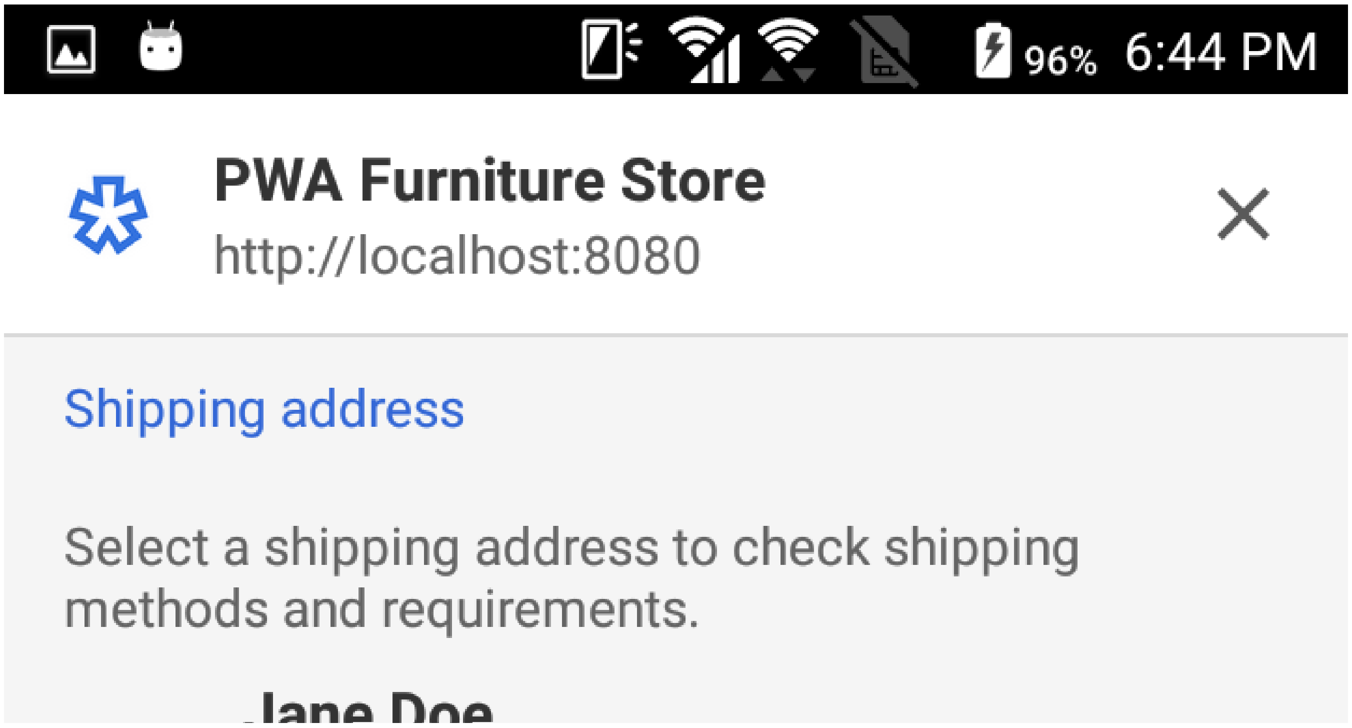
```
// When user selects a shipping address, add shipping options to match
request.addEventListener('shippingaddresschange', e => {
  e.updateWith((_ => {
    // Get the shipping options and select the least expensive
    shippingOptions = this.optionsForCountry(request.shippingAddress.country);
    selectedOption = shippingOptions[0].id;
    let details = this.buildPaymentDetails(cart, shippingOptions, selectedOption
    return Promise.resolve(details);
  })());
});
```

Explanation

Upon receiving the `shippingaddresschange` event, the `request` object's `shippingAddress` information is updated. By examining it, you can determine if

- The item is deliverable.
- Shipping cost needs to be added/updated.

This code looks into the country of the shipping address and provides free shipping and express shipping inside the US, and provides international shipping otherwise. Checkout `optionsForCountry()` function in **app/scripts/modules/payment-api.js** to see how the evaluation is done.


96% 6:44 PM

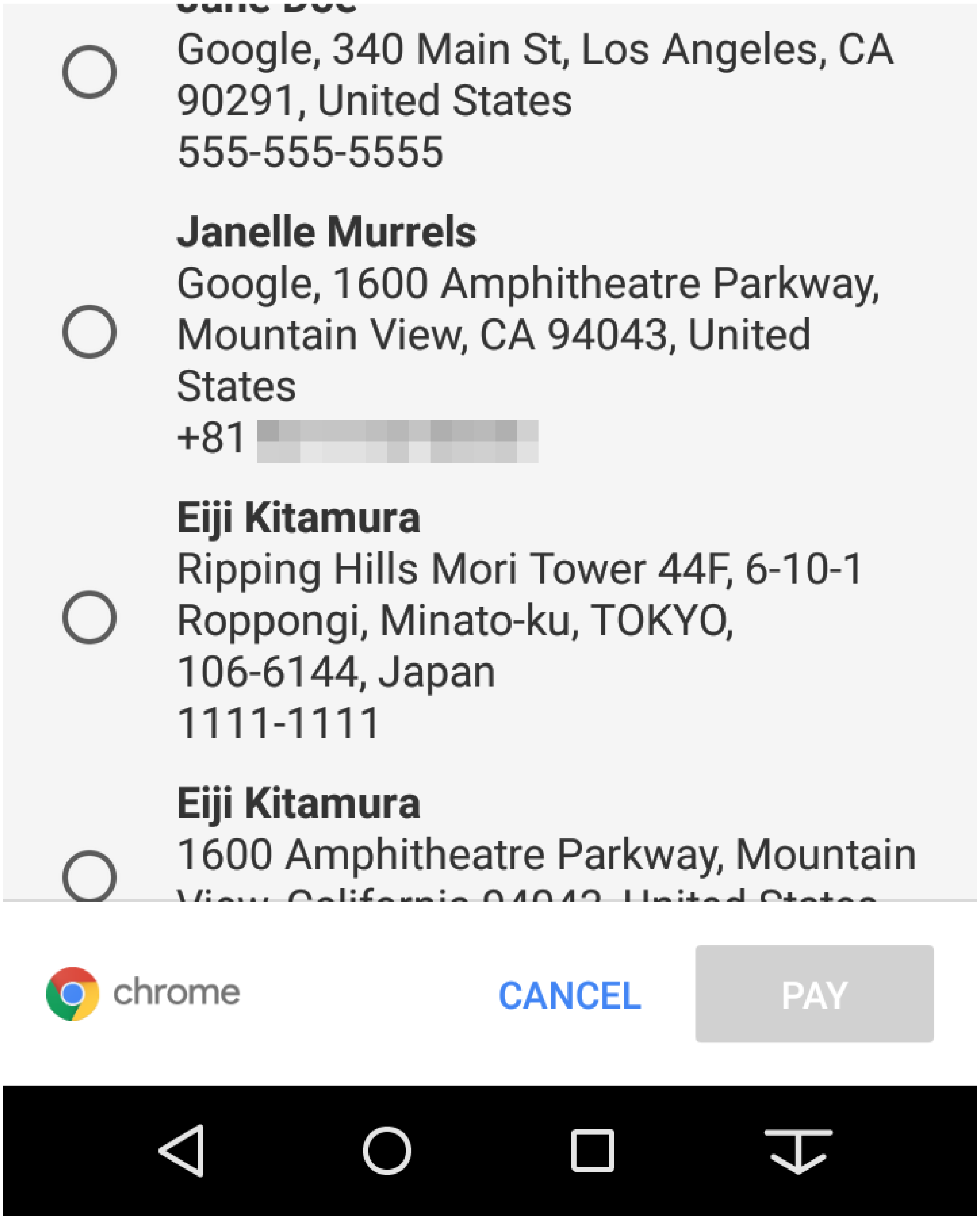
PWA Furniture Store

http://localhost:8080

Shipping address

Select a shipping address to check shipping methods and requirements.

Jane Doe



Google, 340 Main St, Los Angeles, CA 90291, United States
555-555-5555

Janelle Murrels
Google, 1600 Amphitheatre Parkway, Mountain View, CA 94043, United States
+81 [redacted]

Eiji Kitamura
Ripping Hills Mori Tower 44F, 6-10-1 Roppongi, Minato-ku, TOKYO, 106-6144, Japan
1111-1111

Eiji Kitamura
1600 Amphitheatre Parkway, Mountain

Note that passing an empty array to `shippingOptions` indicates that shipping is not available for this address. You can display an error message via `shippingOption.error` in that case.

Add `shippingoptionchange` event listener

When the user changes a shipping option, you will receive the `shippingoptionchange` event.

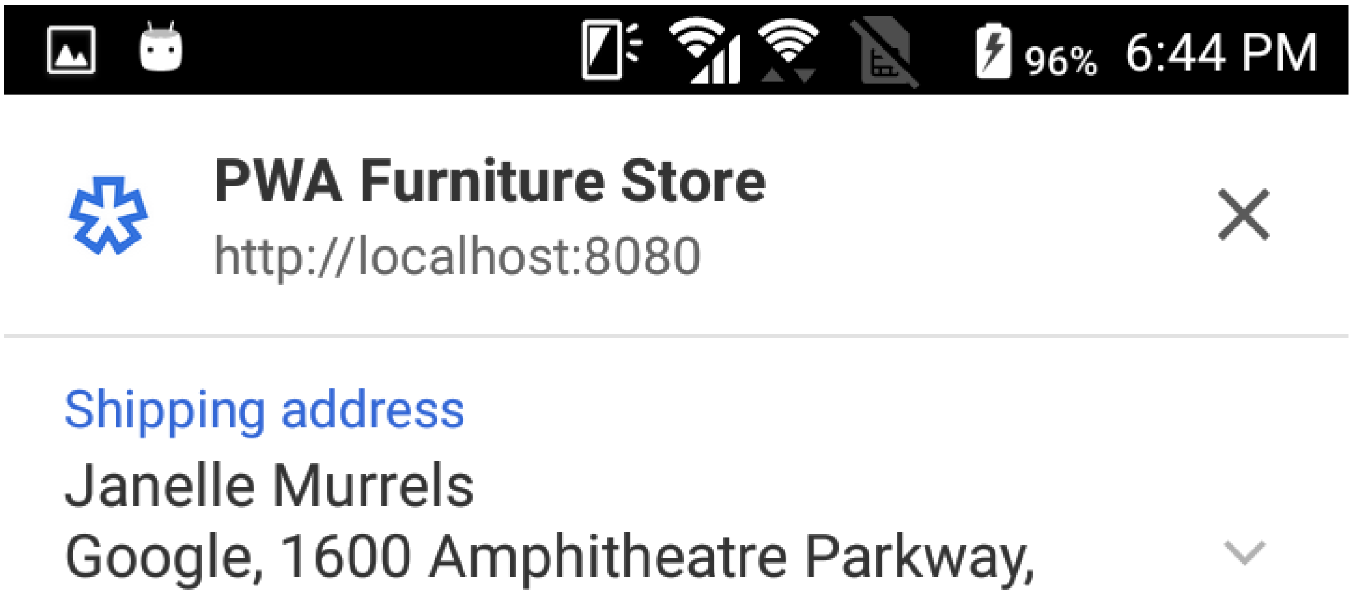
Replace "TODO PAY-8.2" in **app/scripts/modules/payment-api.js** with the following code:

payment-api.js

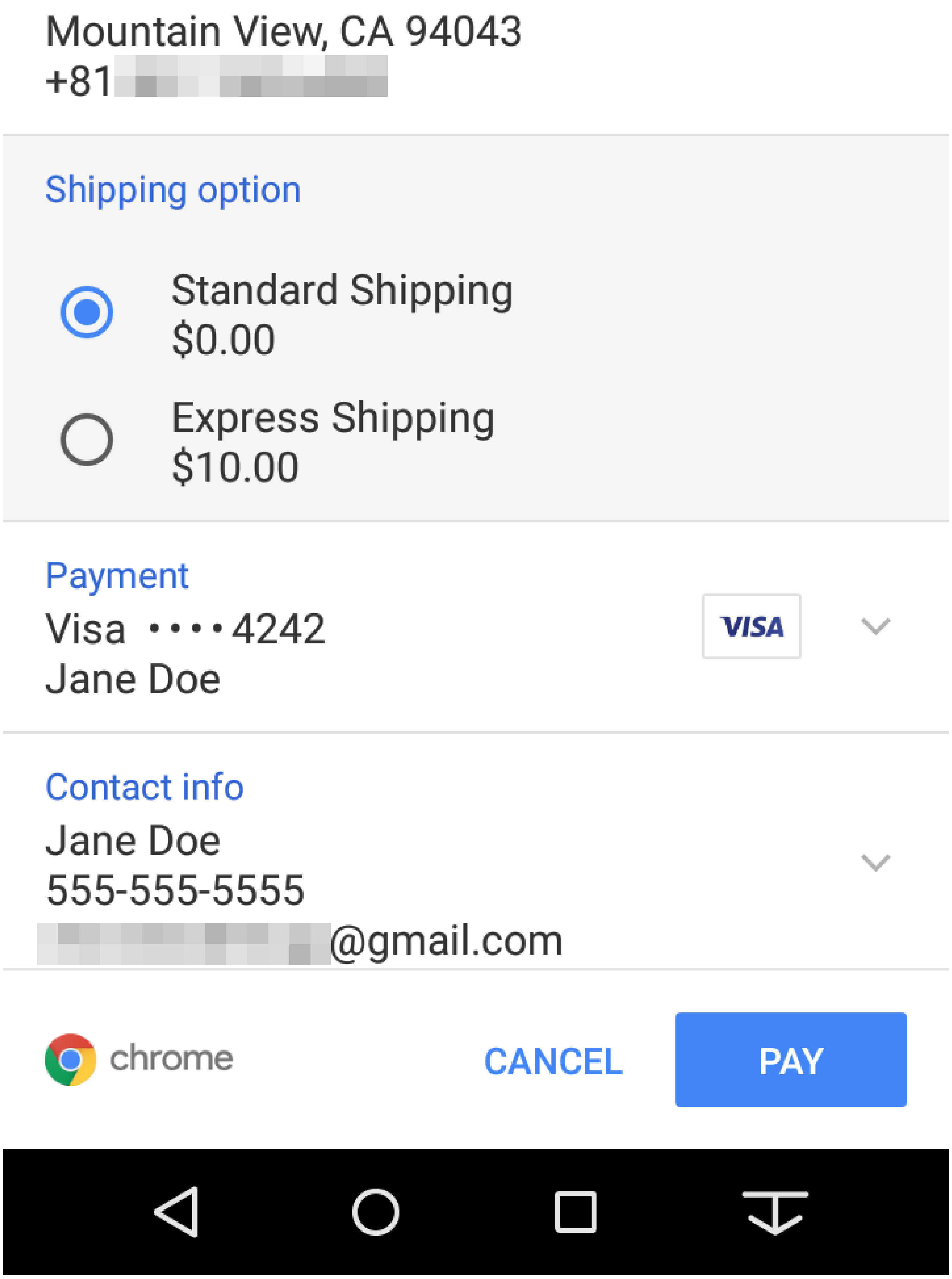
```
// When user selects a shipping option, update cost, etc. to match
request.addEventListener('shippingoptionchange', e => {
  e.updateWith((_ => {
    selectedOption = request.shippingOption;
    let details = this.buildPaymentDetails(cart, shippingOptions, selectedOption
    return Promise.resolve(details);
  })());
});
```

Explanation

Upon receiving the `shippingoptionchange` event, the `request` object's `shippingOption` is updated. It indicates the `id` of the shipping options. Look for the price of the shipping option and update the display items so that the user knows the total cost is changed. Also change the shipping option's `selected` to `true` to indicate that user has chosen the item. Checkout `buildPaymentDetails()` function in **app/scripts/modules/payment-api.js** to see how it works.



Janelle Murrels
Google, 1600 Amphitheatre Parkway,



Mountain View, CA 94043
+81 [redacted]

Shipping option

[✓ Standard Shipping $0.00]

Express Shipping $10.00

Payment
Visa ••••4242
Jane Doe

Contact info
Jane Doe
555-555-5555
[redacted]@gmail.com

Add payment options

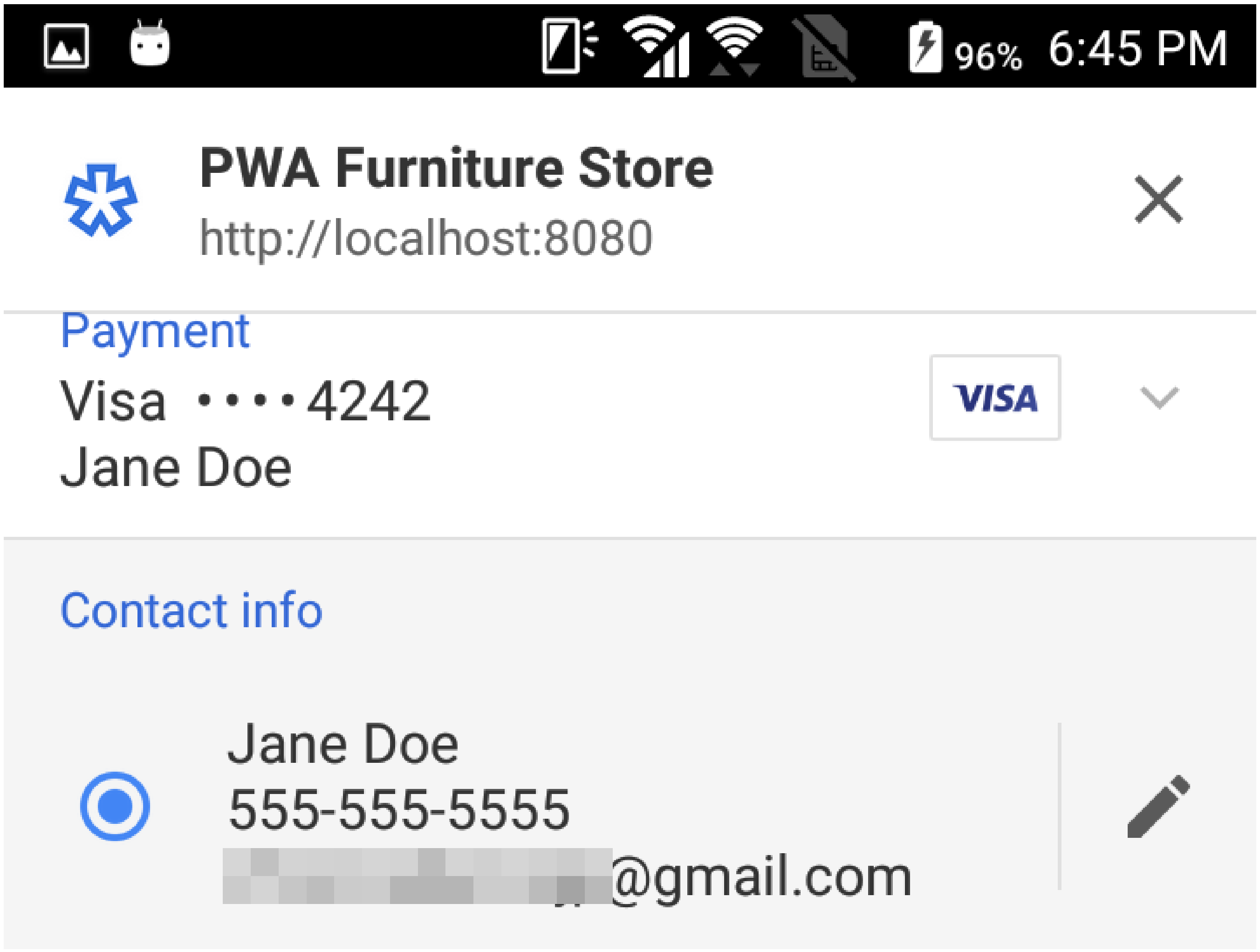
In addition to the shipping address, there are options to collect payer's information such as email address, phone number, and name.

Replace "TODO PAY-9" in **app/scripts/modules/payment-api.js** with the following payment options:

payment-api.js

```
requestPayerEmail: true,
requestPayerPhone: true,
requestPayerName: true
```

Explanation



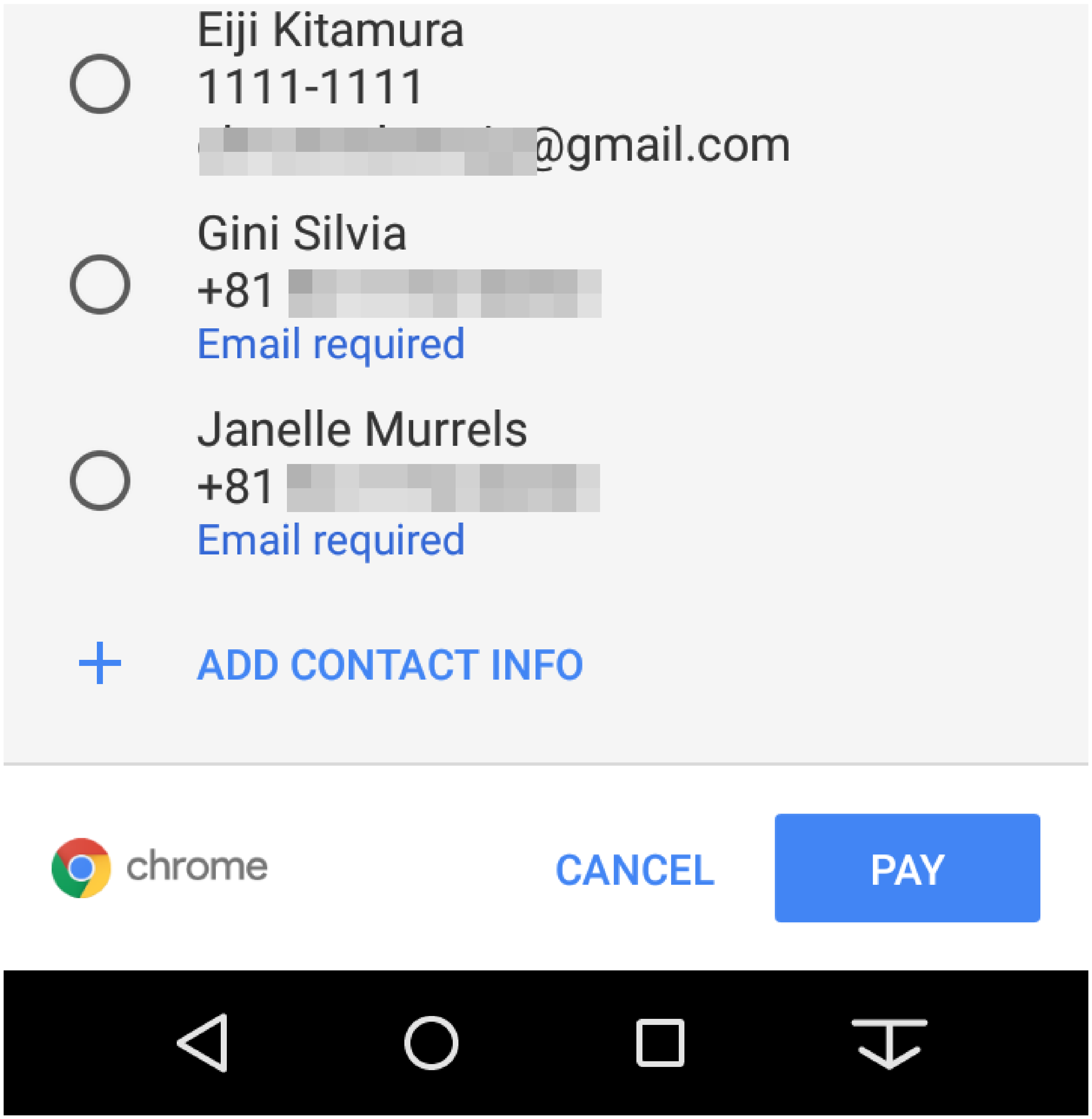
96% 6:45 PM

PWA Furniture Store
http://localhost:8080

Payment
Visa ••••4242
Jane Doe

Contact info

Jane Doe
555-555-5555
[redacted]@gmail.com



Eiji Kitamura
1111-1111
[redacted]@gmail.com

Gini Silvia
+81 [redacted]
Email required

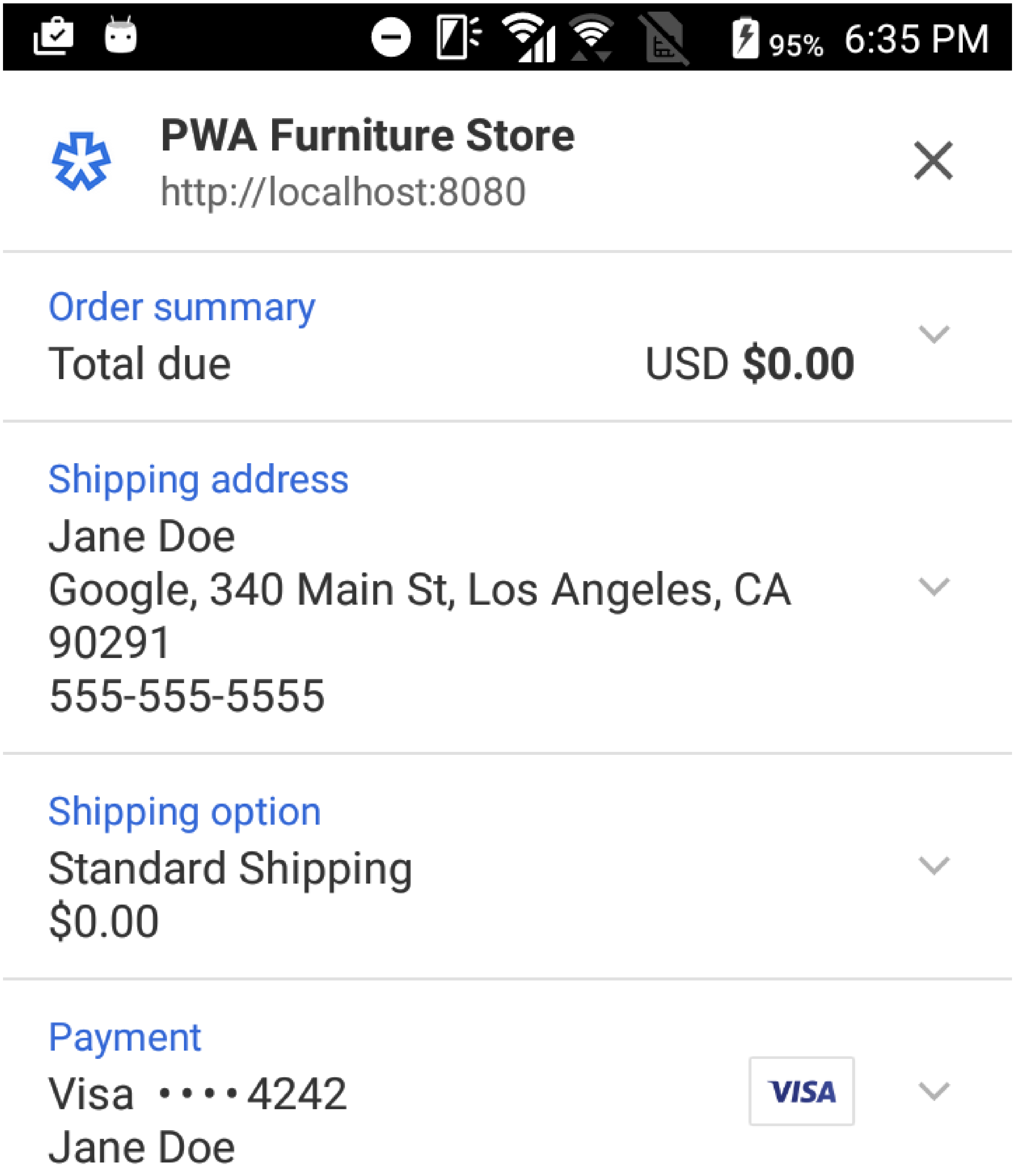
Janelle Murrels
+81 [redacted]
Email required

By adding `requestPayerPhone: true`, `requestPayerEmail: true`, `requestPayerName: true` to the third argument of the `PaymentRequest` constructor, you can request the payer's phone number, email address, and name.

Test it out

Phew! You have now completed implementing the Payment Request API with shipping option. Let's try it once again by running your server if it's stopped.

```
$ npm run serve
```



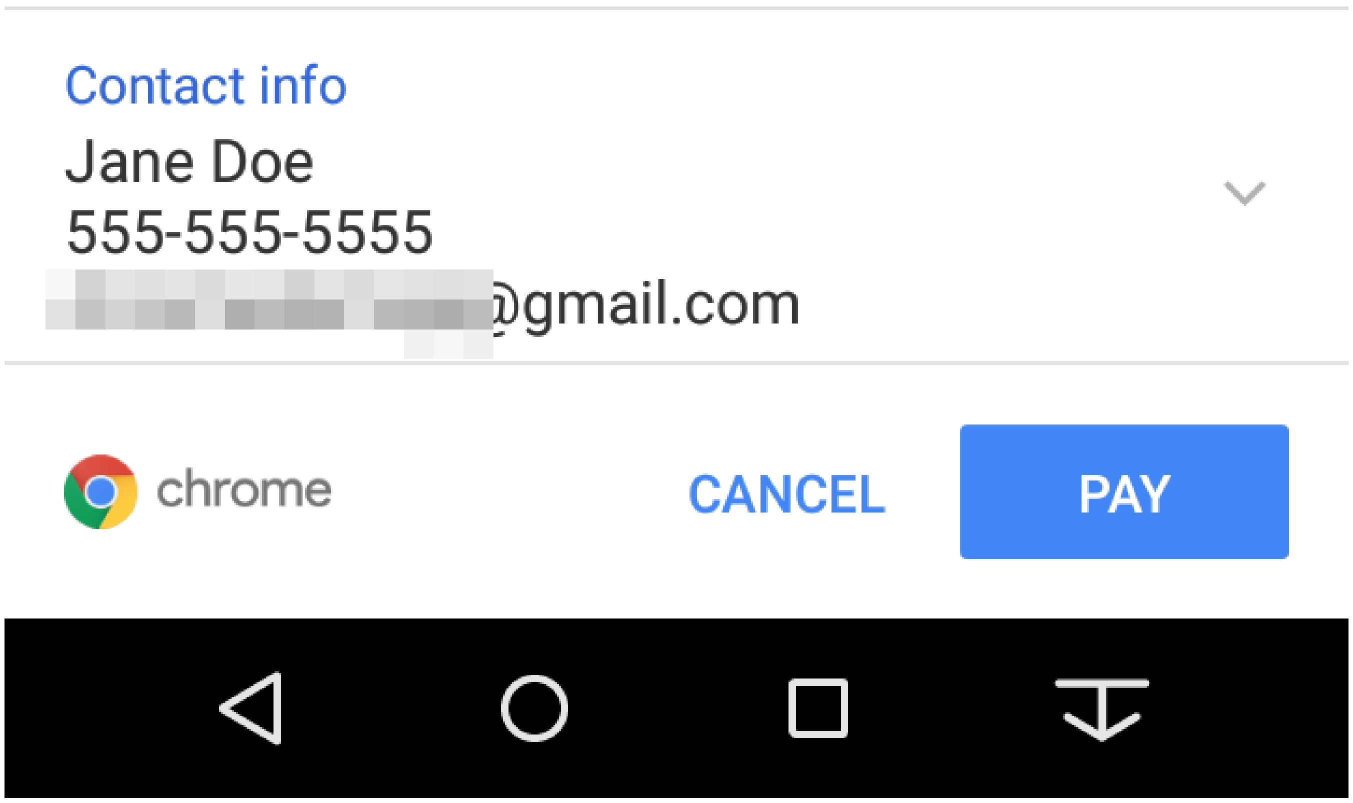
PWA Furniture Store
http://localhost:8080

Order summary
Total due USD **$0.00**

Shipping address
Jane Doe
Google, 340 Main St, Los Angeles, CA 90291
555-555-5555

Shipping option
Standard Shipping
$0.00

Payment
Visa ••••4242
Jane Doe


Contact info
Jane Doe
555-555-5555
[redacted]@gmail.com

Try: add random items to the card, go to checkout, change shipping address and options, and finally make a payment.

Congratulations!

You have added Payment integration to the e-commerce app. Congratulations!

To learn more about the Payment Request API, visit following links.

Resources

- Bringing Easy and Fast Checkout with Payment Request API (https://developers.google.com/web/updates/2016/07/payment-request)
- Payment Request API: an Integration Guide (https://developers.google.com/web/fundamentals/discovery-and-monetization/payment-request/)
- Web Payments session video at Chrome Dev Summit 2017 (https://www.youtube.com/watch?v=U0LkQijSeko)

Specs

- Payment Request API (https://w3c.github.io/browser-payment-api/)
- Payment Handler API (https://w3c.github.io/webpayments-payment-apps-api/)

Demos

- https://paymentrequest.show/demo/ (https://paymentrequest.show/demo/)
- https://googlechrome.github.io/samples/paymentrequest/ (https://googlechrome.github.io/samples/paymentrequest/)
- https://woocommerce.paymentrequest.show/ (https://woocommerce.paymentrequest.show/)

Android Pay

If you are interested in enabling Android Pay on top of the Payment Request API, learn how at Android Pay for the Web Codelab (https://codelabs.developers.google.com/codelabs/android-pay-web/).

Found an issue, or have feedback?

Help us make our code labs better by submitting an issue (https://github.com/google-developer-training/pwa-ecommerce-demo/issues) today. And thanks!

Chromium Blog
The latest news on the Chromium blog.

GitHub
Fork our code samples and other open-source projects.

Twitter
Connect with @ChromiumDev on Twitter.

Videos
Check out our videos.

Events
Attend a developer event and get hacking.