Eli Elad Elrom

176 Followers About Follow

Learn how to create your own Dapp with Angular 9 — Part VI.

Eli Elad Elrom Jan 6, 2020 · 6 min read ★

In this six-part article, we will cover how to create a Dapp with Angular. In Part I, which served as an introduction we covered general information regarding developing the dapp benefits and classification. Using Angular, Angular architecture, benefits. You should start there.

In the previous articles, we started developing our dapp. Specifically, we learned about dapp classifications and projects and that you can break your own dapp project into five steps.

We then looked at why to use Angular and its benefits. Next, we created an Angular project, first ensuring the prerequisites were installed and then installing the Angular CLI.

We looked at the pieces that make up Angular such as components, modules, and directives. We also learned how to style a dapp by understanding Angular-style architecture and working with Angular Material.

We started building our own custom components and creating content; we split the app

into a footer, header, and body and created a custom transfer component that you will be using in this article. We created the dapp's smart contract utilizing the following tools the Angular CLI, Truffle, ganache-cli, and MetaMask.

We integrated our smart contract in the dapp's Angular project by installing web3.js as well as truffle-contract. We then created our transfer service in Angular and connected our component class to the service class.

In this final and last article, **Part IV**; we will be linking and connecting our dapp to the Ethereum network and testing.

Connect to MetaMask

At this point, your dapp code is complete. However, if you test your dapp now, web3 won't be able to connect to an account.

What you need to do is connect to MetaMask. There is a privacy issue related to dapps where malicious web sites are able to inject code to view users' activities and Ethereum addresses and then find balance, transaction history, and personal information.

These malicious sites are then able to initiate unwanted transactions on a user's behalf, and the user accidentally may approve an unauthorized transaction and lose funds.

To avoid these issues and to connect your Angular service, you will connect the browser to the network via MetaMask.

MetaMask

To interact with your contracts, without even downloading a desktop app (like mist), is in your Chrome or Firefox browser with a plugin called MetaMask. Just as with Mist, you can utilize MetaMask to connect your contract to mainnet, a public testnet, and a local blockchain (such as the one you created with Ganache), or you can even connect to Truffle Develop. To get started, download the MetaMask plugin for Chrome or Firefox.

- Chrome Web Store: https://chrome.google.com/webstore/detail/metamask/nkbihfbeogaeaoehlefnkodbefgpgknn. Click the Add to Chrome button
- Firefox Add-ons page: https://addons.mozilla.org/en-US/firefox/addon/ether-metamask. Click the Add

to Firefox button.

If you don't have a wallet, when the welcome page comes up;

New to MetaMask? > select "Yes, let's get set up! create a wallet". Select a secure password. Lastly, select "Secret Backup Phrase", see Figure 1;

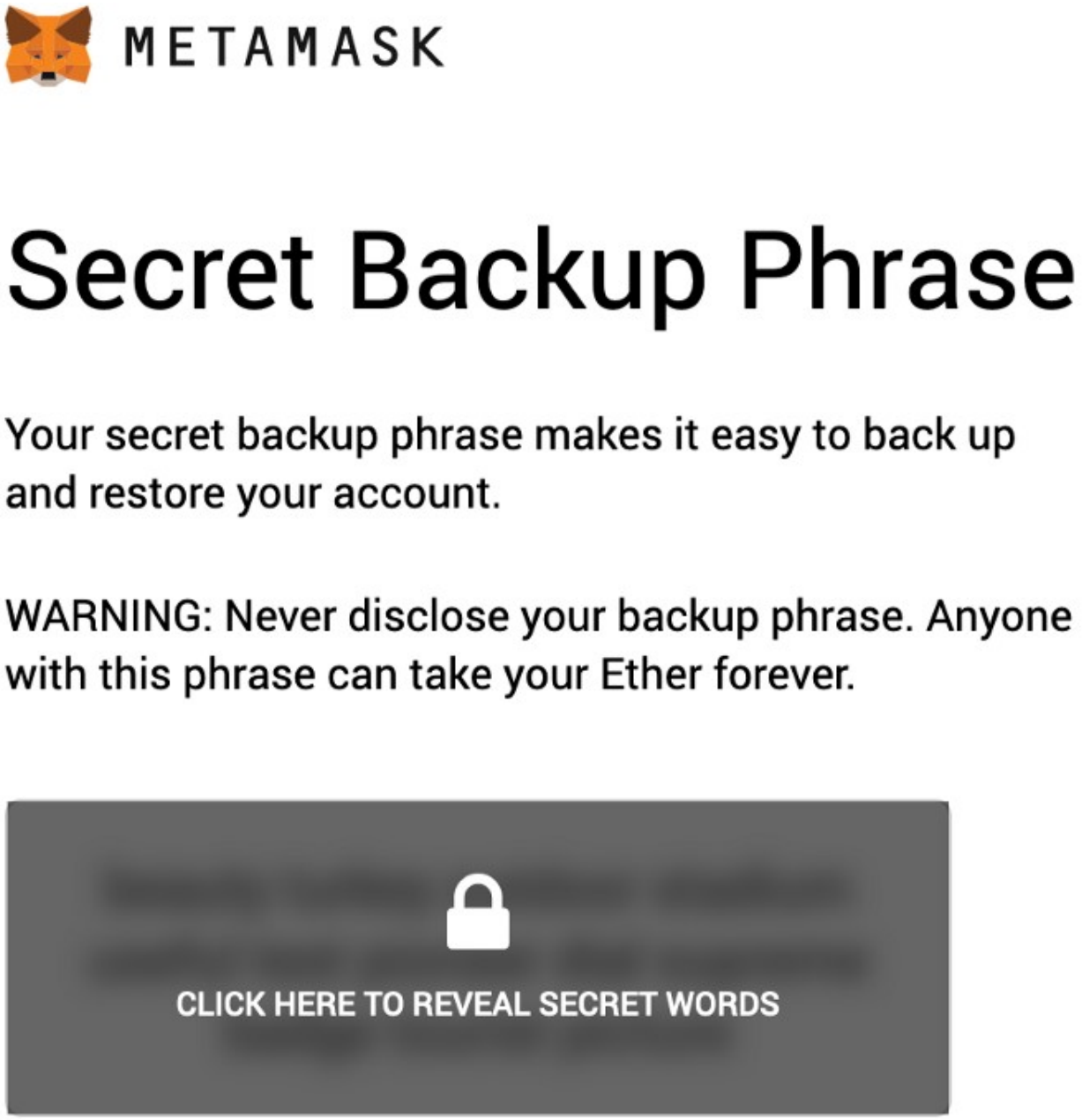


Figure 1. Backup your secret words.

Let's back up for a second. As you'll recall, you started a network via ganache-cli on port 8545 via the following command in Terminal;

```
ganache-cli -p 8545
```

And open another terminal and ensure connect Truffle to the network.

```
cd ~/Desktop/ethdapp/truffle
truffle migrate --network development
```

Then you were able to connect on port 8545 and run commands in Terminal.

Lastly, make sure you Dapp is running, with ng serve command in a 3rd terminal;

```
cd ~/Desktop/ethdapp
ng serve
```

You can now connect MetaMask in a browser. To connect, select MetaMask and select Localhost 8545 in the drop-down menu. See Figure 2.

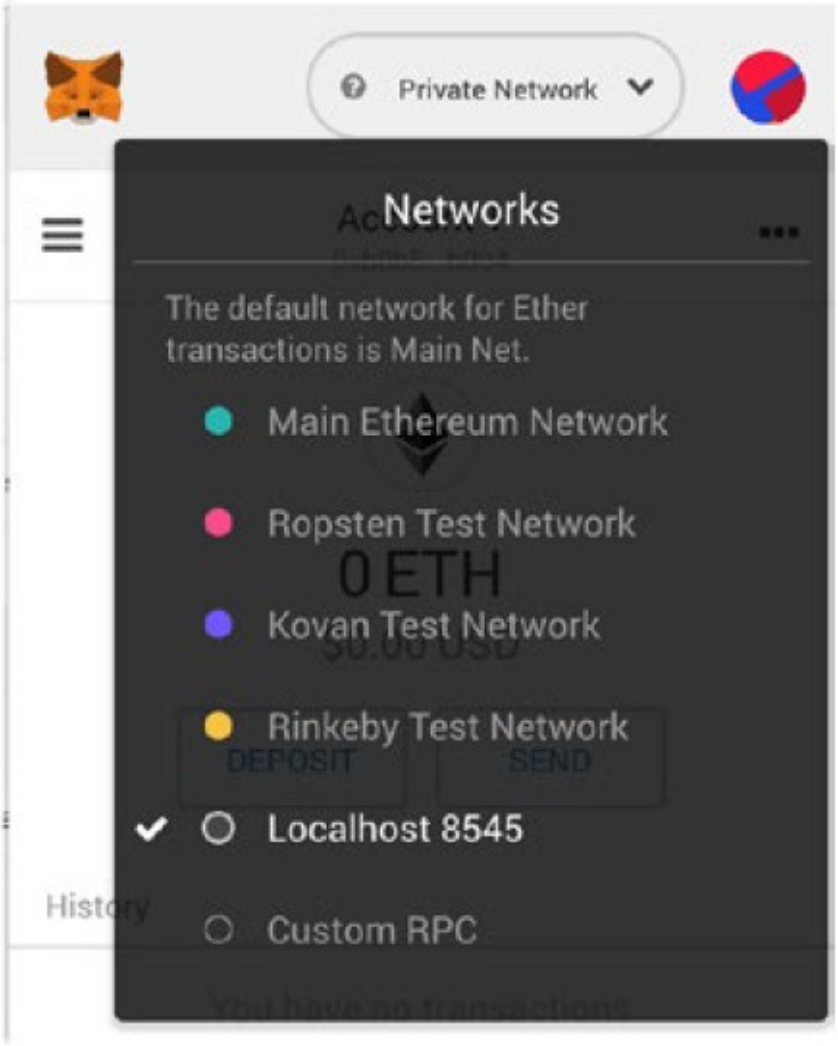


Figure 2. Connecting MetaMask to a private network on port 8545

Notice that you picked port 8545 earlier in the previous article. It's the default port on MetaMask, so it's easy to connect to on your private network by selecting the drop-down menu item instead of pointing to a custom port.

However, when you check the list of accounts, you don't see any accounts. The reason you don't see accounts is that every time you start your network, you need to update the accounts. There are two ways to update MetaMask with the list of accounts.

Option 1

=====

When you run Ganache, use the "m" flag to pass the mnemonic that represents the private keys you had in Ganache. Stop Ganache and now run it again

with your key phrase;

```
ganache-cli -p 8545 -m 'YOUR KEYS PHRASE'
```

Remember to re-deploy the smart contract to the network;

```
truffle migrate — network development
```

Option 2

=====

When you run ganache-cli, you will see the list of accounts, private keys, and mnemonics.

```
ganache-cli -p 8545
```

Look for this output and copy the mnemonic.

```
HD Wallet
==================
Mnemonic: journey badge medal slender behind junk develop
produce spy enemy transfer room
Base HD Path: m/44'/60'/0'/0/{account_index}
```

Then, log out of MetaMask and paste the mnemonic manually. Click the right button and select “Log out,” as shown in Figure 3.

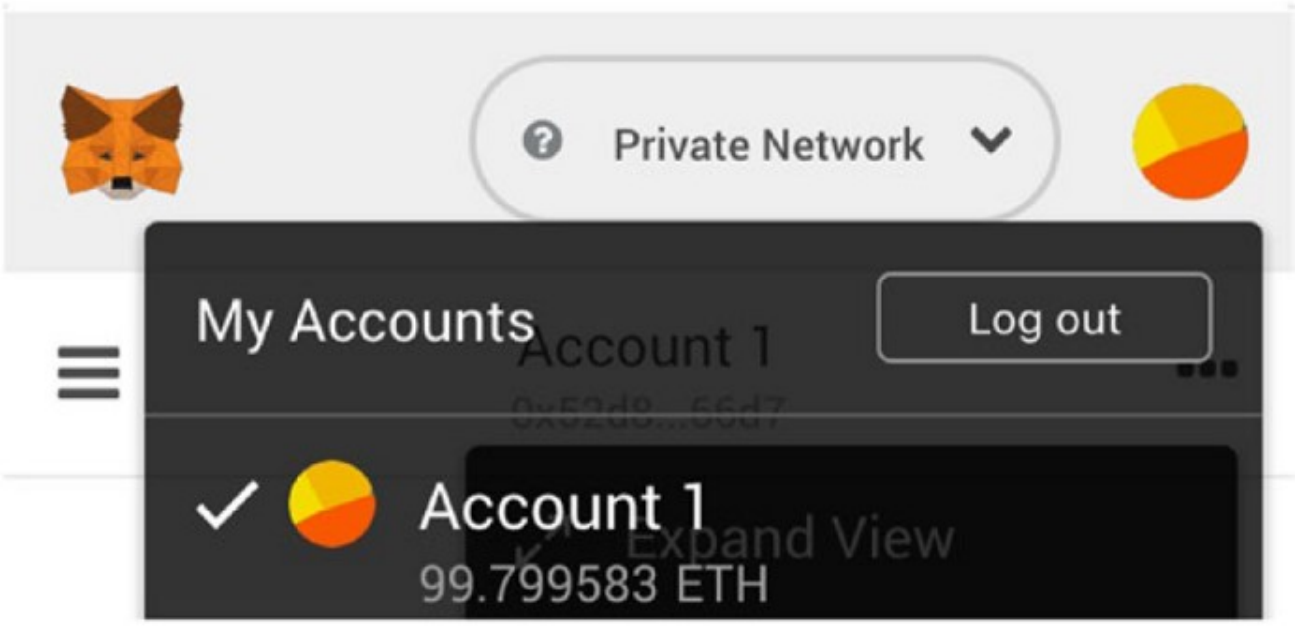


Figure 3. MetaMask log out of the account

After logging out, the welcome screen comes back with a link under it that says, “Import using account seed phrase.” Click that link, as shown in Figure 4.

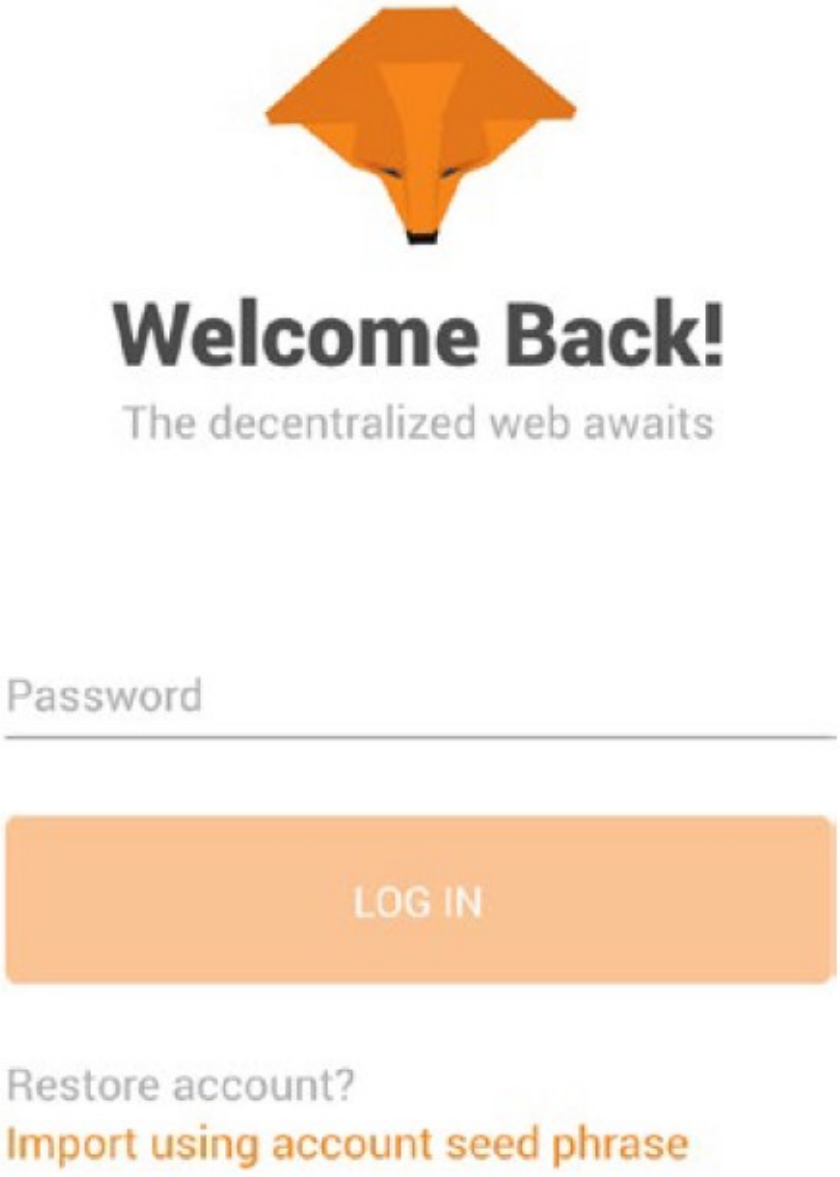


Figure 4. MetaMask welcome page

Now you can paste the mnemonic by selecting a password and clicking Restore, as shown in Figure 5.

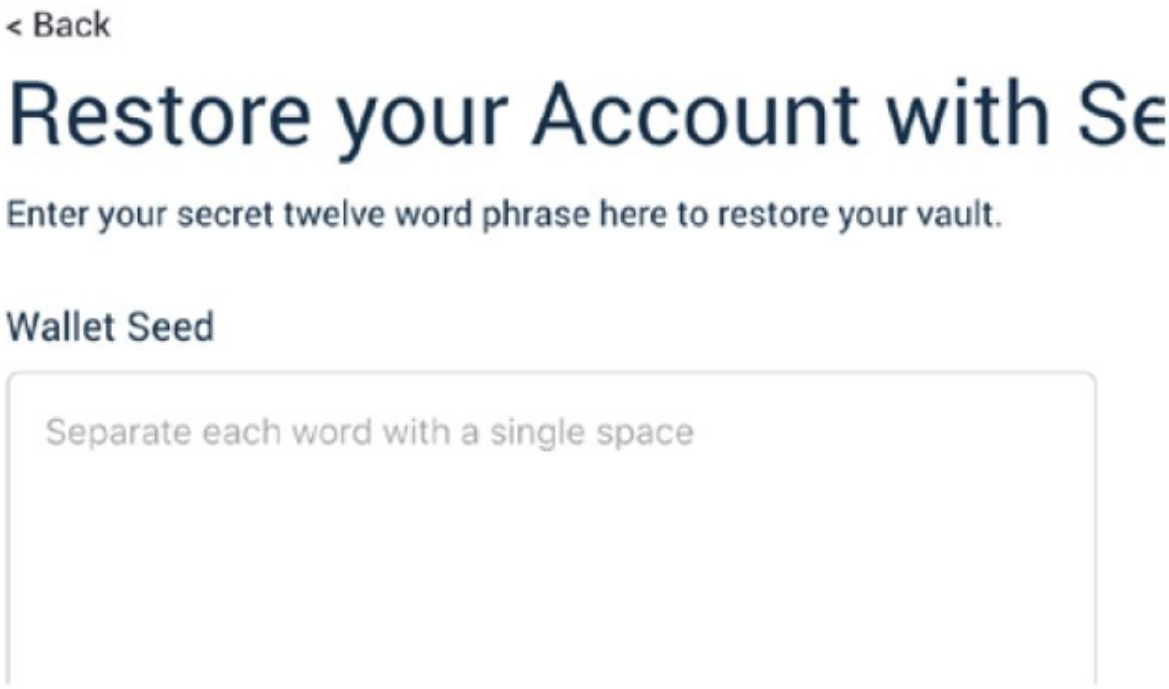


Figure 5. Restoring MetaMask account using a mnemonic

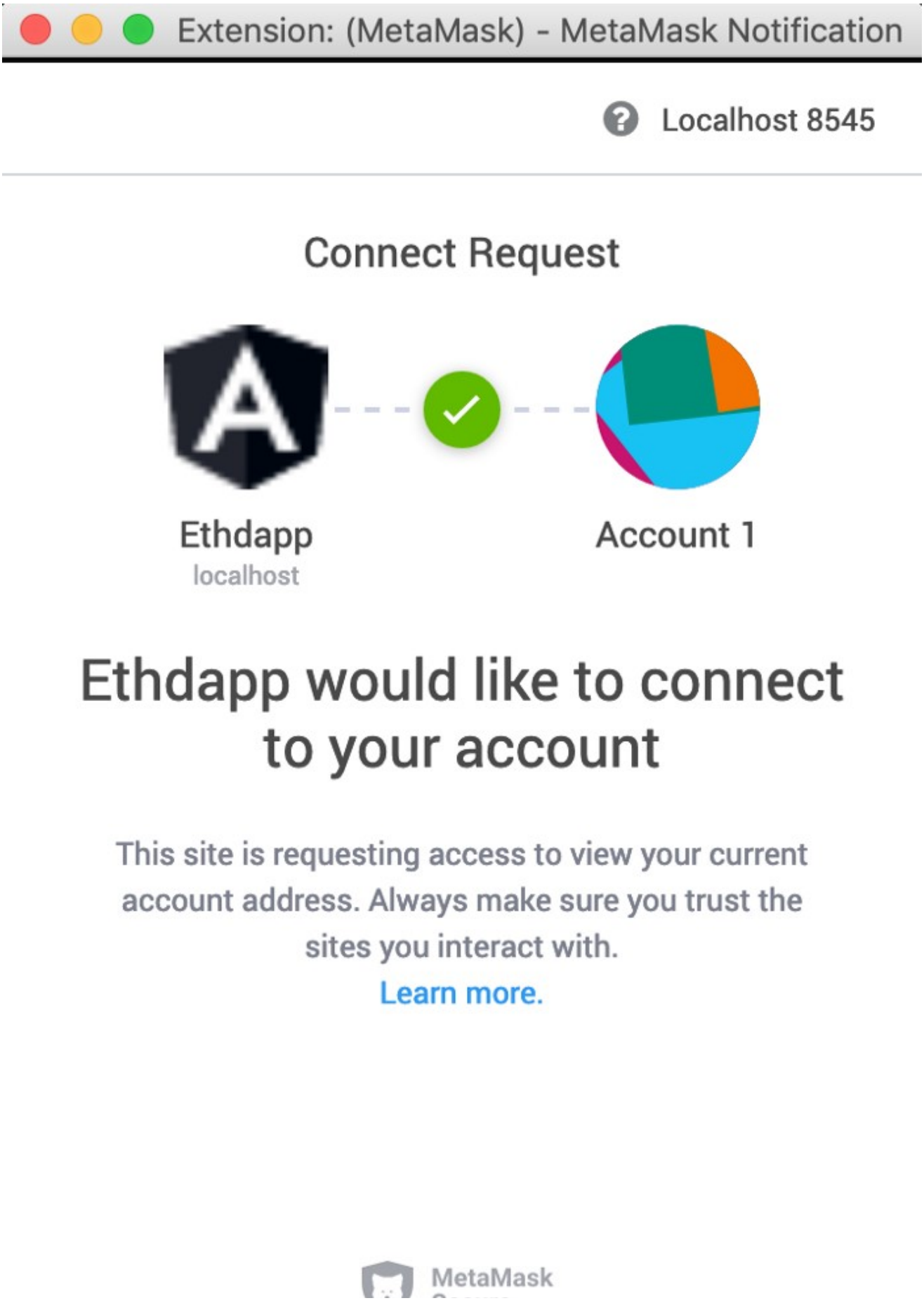
Lastly, create a second account, so we can transfer our funds between these two accounts. This can be done with MetaMask; User > create an account. I called my accounts; “account 1”, “account 2”, but you can call them anything you like.

Test Your Dapp Functionality

Now you are finally ready to test your dapp. Once the browser gets refreshed, the first request will be to enable our dapp to interact with MetaMask. As you recall in our **transfer.service.ts** we set a code to enable the browser to interact;

```
private async enableMetaMaskAccount(): Promise<any> {
  let enable = false;
  await new Promise((resolve, reject) => {
    enable = window.ethereum.enable();
  });
  return Promise.resolve(enable);
}
```

You should expect the following window, see Figure 6,


Extension: (MetaMask) - MetaMask Notification
Localhost 8545
Connect Request
Ethdapp
localhost
Account 1
Ethdapp would like to connect to your account
This site is requesting access to view your current account address. Always make sure you trust the sites you interact with.
Learn more.
MetaMask

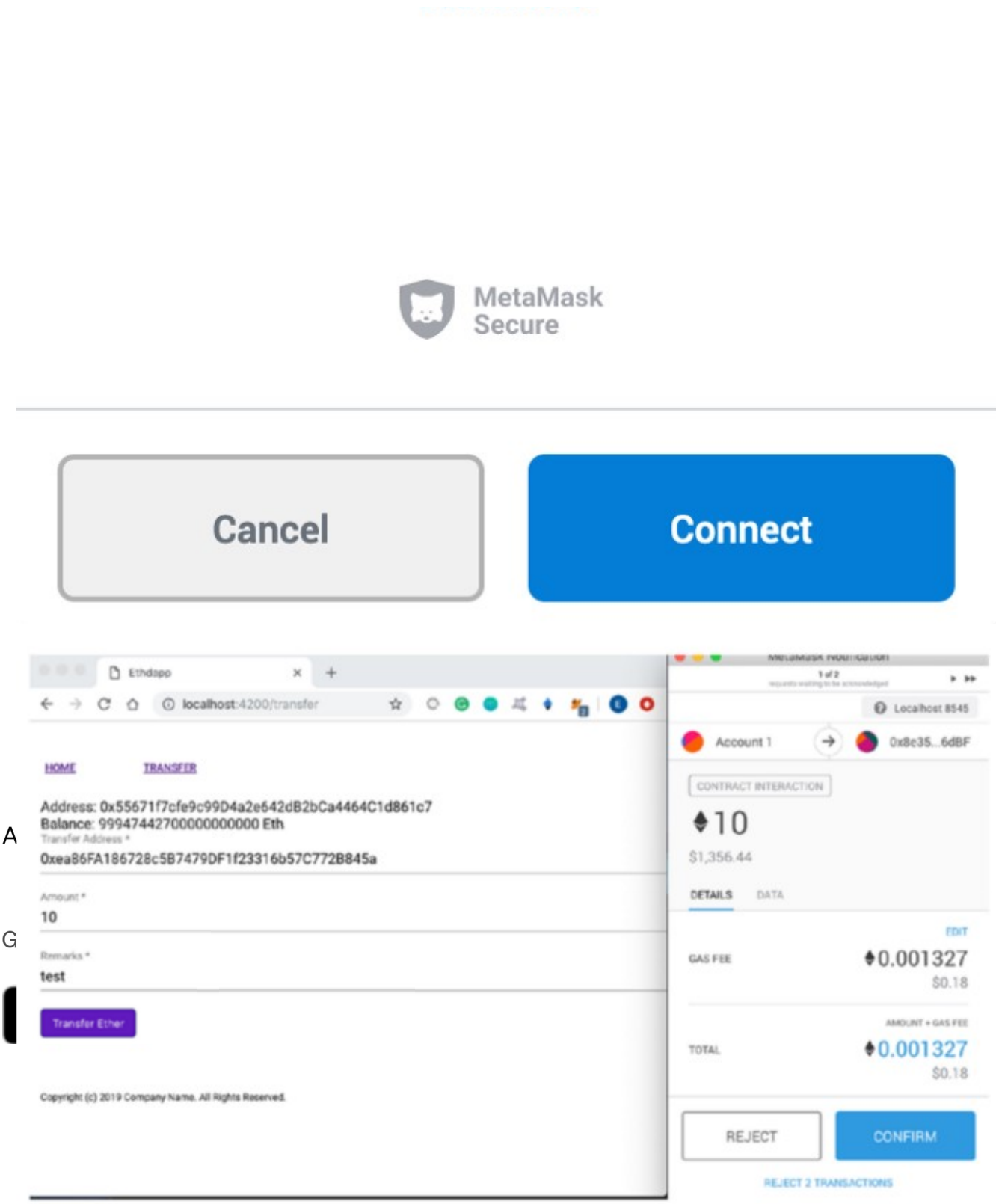


Figure 7. MetaMask notification to complete a transfer

Let's recap!

In our six-part articles, we created a transfer smart contract and Truffle development project as well as connected to the Ganache local development

network. You learned how to work with the Ethereum network via Truffle and MetaMask and how to test your smart contract in Terminal. As you recall we tested the transfer of funds usin our smart contract via the command line.

Lastly, we linked our dapp with a development Ethereum network using an Angular TransferService component that we created. Using the web3 library and @truffle/contract, we made service calls. Lastly, we connected to MetaMask to manage your accounts.

Where to go from here

Continue working and improving the dapp we created. For instance, you could do the following:

— Create a user service class and a shared service class to hold users' information and shared information
— Create a login/logout service
— Create an option to switch between accounts
— Create a side menu to better navigate the app
— Update the smart contract and add more methods and events

To learn more about what's possible with Blockchain as well as develop your own project che The Blockchain Developer.