Solana Dev 101 - How to Create a Solana Transaction

Introduction

GM! In this blog we'll be looking into how to create and send a transaction with Javascript on Solana as part of the Solana Dev 101 Series. This is a beginner-friendly guide, so don't worry if you're new to Solana or blockchain tech.

If you have any questions, don't hesitate to join the Helius Discord or tag Helius on Twitter for assistance.

Prerequisites

To get started we will need need to have Node.js and npm installed our machine. If needed you can download them [here

](https://nodejs.org/en/download/).

We are also going to need to install the Solana web3.js library, which is a suite of JavaScript tools and libraries designed to interact with the Solana blockchain. You can install it with npm by using your terminal to run the following code:

Step 1: Setting Up the Connection

First, let's create a connection to the Solana cluster. We will be using Helius to connect to a devnet cluster. You will need to create a Helius account and API key which you can get for free by signing up here.

Step 2: Creating a New Wallet

Now we need a wallet to send transactions from. A wallet is just a pair of public and private keys and this can be created easily. A public key is simply your wallet address, and then your private key is used to sign transactions. You can read up more about public key cryptography [here

](https://www.ibm.com/docs/en/ztpf/2020?topic=concepts-public-key-cryptography). We can create a new file system wallet, the keypair to this wallet will be stored on your local machine. If you have already installed the Solana CLI tools you can run the following command in the terminal to get a new wallet.

If you do not the you will need to install the Solana CLI Tool Suite. You can get more information on how to do this in the Solana docs here: https://docs.solana.com/cli/install-solana-cli-tools

WARNING: You should never share your private key with anyone.

We can then pull this wallet into our code:

You can copy your secret key from the json file system wallet that you created in the previous step and use that.

Step 3: Funding the Wallet

To send transactions, we will need to have some SOL in our wallet. Every transaction on Solana requires a small fee for it to be processed. For testing purposes, we will use Solana's faucet to fund our wallet. Head over tosolfaucet.com and enter the public key form the wallet you just created and request some SOL.

Step 4: Creating a Transaction

Now that we have a funded wallet, we can create a transaction. In this example, we're going to create the transaction and add the transfer instruction to this transaction. We will use this to transfer 0.1 SOL to wallet of our choosing. A transaction can do a multitude of things depending on what instruction you give, in this example we will be sending SOL with the transfer instruction.

In the code above we create a new PublicKey object with a wallet of our choosing. The PublicKey object, rather than just the

address, is needed to create a transaction. The transfer instruction also takes 3 parameters:

1. fromPubkey

: This represents the public key of the sender's account.

1. toPubkey

: This represents the public key of the receiver's account.

1. lamports

: This represents the amount of lamports (the smallest unit of currency in Solana) to be transferred. solanaWeb3.LAMPORTS_PER_SOL

indicates 1 SOL.

Step 5: Signing and Sending the Transaction

Once the transaction has been created we will need to sign that transaction and send it to the network. The Solana Web3 library has a function called sendAndConfirmTransaction() to allow for us for us to to do all of this easily. The sendAndConfirmTransaction method sends the transaction to the network, waits for confirmation, and returns a signature for the transaction.

The sendAndConfrimTransaction method takes three parameters:

1. connection

: This is the Solana network connection object that we made using a Helius RPC connection. This parameter specifies the connection over which the transaction will be sent and confirmed.

1. transaction

: This is the transaction object that we just made contains the transfer instruction.

1. [sender]

: This is an array containing all the signer accounts for the transaction. In this case, it contains a single element sender

, which represents the account signing and authorising the transaction.

And there we go! We just created and sent a transaction on Solana using JavaScript. You can use node sendTransaction.js to run the script.

Conclusion

Creating and sending a transaction on Solana with JavaScript is a straightforward process, thanks to Solana's comprehensive JavaScript library. With just a few lines of code we are able to interact with the Solana blockchain which opens up a world of possibilities.

This is a basic example and highlights the structure and flow of creating and sending a transaction, real-world applications may involve more complex transactions and require additional security measures.

I hope this guide has been helpful. Happy coding!

You can find the complete code [here

](https://github.com/owenventer/SolanaDev101/blob/main/createAndSendTransactions/sendTransaction.js).

Resources

- [Solana's official JavaScript API documentation

](https://solana-labs.github.io/solana-web3.js/)