

# DeFi on Solana · Week 2

Speaker script: every slide with its notes. Facts checked 11 to 14 Oct 2026; re-check anything marked "verify".

1

Week 2 of 6

2

Wallets & on-chain basics

Task

Get devnet SOL and USDC, send USDC to a partner, read the transaction on Orb.

Blog post

How I interacted with my first Solana dApp.

## DeFi on Solana · Week 2

Week 2 goal: every mentee leaves with a devnet wallet, has sent a token, and can point at three things in an explorer: the fee, the token instruction, and the token account it created.

Everything in weeks 3 and 4 builds on one picture: wallet → token account → mint. If they leave with only one thing, make it that.

Faucets are the number one thing that breaks in class. Pre-fund a teacher wallet with devnet SOL and USDC before class.

2

Today

1

Keys

2

Accounts

3

Tokens

4

Transactions

→

Send devnet USDC

## Today

No notes. Let the slide speak.

3

Keys

Your wallet doesn't hold coins.

## Keys

No notes. Let the slide speak.

4

wallet is a keyring, not a purse

Wallet app

Phantom, Solflare, Backpack

stores your secret key

signs →

Transaction

"move 5 USDC to Bob"

"proof" it was you

→

The ledger

where the balance lives

on thousands of computers

Lose the phone, keep the phrase: **your money is fine.**  
Lose the phrase: **nobody can help you.**

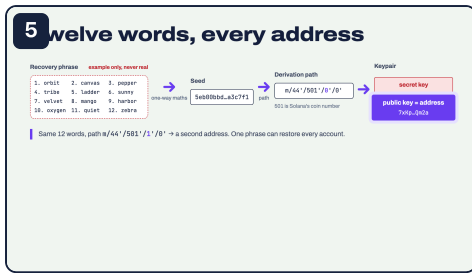
## A wallet is a keyring, not a purse

The single most important mental model this week. Coins are not "in" the wallet app. They are entries on the ledger next to your address. The wallet app holds the key that lets you change those entries.

Consequences:

- If your phone breaks, install Phantom on a new phone, type your recovery phrase, and everything is there. Nothing was lost because nothing was on the phone.
- If someone gets your phrase, they can do the same thing on their phone. They don't need your device.

Compare a bank: your money is in the bank's ledger, and the bank decides who gets access. Here, the key decides.



## Twelve words, every address

Click through slowly.

1. The recovery phrase (also called seed phrase): 12 or 24 common words from a fixed list of 2,048 (the BIP-39 list). The example on screen is made up. Never show a real one.
2. The words are turned into a long number called the seed. This is one-way: you can't get the words back from the seed or anything after it.
3. A derivation path picks one keypair out of that seed. Solana wallets like Phantom use m/44'/501'/0'/0'. 501 is Solana's registered coin number.
4. Out comes a keypair: a secret key and a public key.
5. The public key, written in base58, is your address. That's what you share.
6. Change one number in the path and you get a completely different address from the same 12 words. That's how one phrase restores "Account 1", "Account 2" and so on in Phantom.

Your address is the same on mainnet, devnet and testnet. Balances are separate on each.

**6 sign something**

A real keypair, made in this browser. Sign, then edit the message.

Generate a keypair

Public key = your address

not yet

Secret key

not yet

Message

Pay Bola 5 USDC

others trust

Sign with secret key

Signature

not yet

## Sign something

This is real Ed25519, the same signature scheme Solana uses. The key is generated in the browser and never leaves it.

Demo:

1. Generate. Point out the address looks like a Solana address (base58).
2. Click the blurred secret key to reveal it, then say: this is the one thing you never show anyone. Hide it again.
3. Sign "Pay Bola 5 USDC". Valid.
4. Change it to "Pay Bola 500 USDC" without re-signing. Invalid.

That is what a signature proves: this exact message was approved by whoever holds the secret key for this address. It does not prove who that person is in real life, and it doesn't prove the message is a good idea. A phishing site can get you to sign a perfectly valid transaction that drains you.

Signing a message (like "log in to this site") is different from signing a transaction (which moves funds). Wallets show which one you're doing. Read the popup.

7

Who holds the key?

Custodial

An exchange holds it for you.

Easy recovery. They can freeze you. If they fail, you queue with everyone else (FTX).

Non-custodial

You hold it. Phantom, Solflare, Backpack.

Full control. No password reset. Hardware wallets keep the key off your laptop.

The one rule

nobody legitimate will ever ask for your recovery phrase. Not support, not a mentor, not a "wallet validator" site.

- ### Who holds the key?
- Wallet types to name quickly:
- Hot wallet: key on an internet-connected device (browser extension, phone app). Convenient, exposed.
  - Hardware wallet (Ledger and others): the key stays on the device and signs there. Your laptop never sees it. Still vulnerable if you approve a bad transaction.
  - Multisig: several keys must sign. Teams use this for treasuries and admin keys (Squads is the main one on Solana). Week 5's Drift hack shows a multisig can still be tricked.

Security checklist for mentees: write the phrase on paper, never in a photo or cloud note. Use a separate wallet for testing. Check the URL before connecting. Read every signing popup. Revoke or close wallets you don't use.

8

Phantom Support' DMs you: there's a sync bug, enter your recovery phrase at their link to fix it. What do you do?

A Enter it, support needs it to fix things

B Send only the first 6 words

C Ignore and block. It's a theft attempt

D Move funds first, then enter it

### Quiz: 'Phantom Support' DMs you: there's a sync bug, enter your recovery phrase at their link to fix it. What do you do?

Option B is a real thing scammers ask for. Six words out of twelve cuts the guessing problem down enormously. Never share any part of it.

9

Accounts

Everything on Solana is one.

### Accounts

No notes. Let the slide speak.

10

One shape for everything

Your wallet

A token mint

Your token account

A program

address a public key, like 7x1p...oDx

lamports

2,500,000,000 (2.5 SOL)

SOL balance, in billions

data

empty

bytes the owner program needs

owner

System Program

the only program allowed to change it

executable

false

is this account code?

Just SOL. No data. Owned by the System Program, which only lets your key move it.

- ### One shape for everything
- Click through the four tabs. Same fields every time; only the values differ.
- lamports: the SOL balance. 1 SOL = 1,000,000,000 lamports.
  - data: raw bytes. Only meaningful to the program that owns the account.
  - owner: the program allowed to change this account's data and take its lamports. This is the single most important field for security. A token account owned by anything other than the Token Program is fake.
  - executable: true means the account is a program.

Key idea for builders and analysts: programs don't store data inside themselves. They store it in separate accounts they own. A swap pool is an account owned by the DEX program. Your USDC balance is an account owned by the Token Program.

The numbers in the deposit rows are the real rent-exempt minimums: 1,461,600 lamports for an 82-byte mint, 2,039,280 lamports for a 165-byte token account.

## 11 Rent is a deposit, not a fee

### Create an account

Lock a small SOL deposit for the space it uses.

Token account: 9.00283928 SOL

### Use it

The deposit sits there. Nothing is charged over time.

### Close it

The deposit comes back to you.

+9.00283928 SOL

That's why your first USDC send to someone costs more than the second.

## Rent is a deposit, not a fee

The name "rent" is historical. Today every account must hold a minimum balance proportional to its size ("rent-exempt"), and that balance comes back when the account is closed.

The first time you send USDC to someone who has never held it, your transaction creates their USDC account, so you pay its deposit (about 0.002 SOL). Second time, the account exists, so you pay only the tiny fee.

Wallets like Phantom let you close empty token accounts and reclaim the SOL. Some "claim your free SOL" sites do exactly this, legitimately or as bait. Check what a site asks you to sign.

## 12

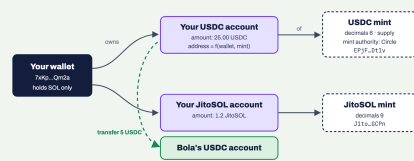
## Tokens

One program, many mints.

## Tokens

*No notes. Let the slide speak.*

## 13 Wallet → token account → mint



## Wallet → token account → mint

The diagram that weeks 3 and 4 depend on. Draw it on a whiteboard too.

1. Your wallet: a System Program account holding SOL.
2. The USDC mint: one account that defines USDC. Decimals (6), total supply, and who is allowed to mint more (Circle) or freeze accounts.
3. Your USDC token account: holds your USDC balance. Its address is derived from your wallet address plus the mint, so anyone can work out where your USDC lives. That's the Associated Token Account, or ATA.
4. Another token, another token account. One per token you hold.
5. Sending to Bola moves USDC from your token account to Bola's token account. Your wallet signs, but the balance change happens in the token accounts.

Ethereum comparison: an ERC-20 is a contract with a table of balances inside it. On Solana, the code (Token Program) is shared by every token, the definition is the mint account, and each holder's balance is its own account.

14

Which one is USDC?



USDC

USD Coin

BRjpcCH...k1k



USDC

USD Coin

4zMMC9...cDU



USDC

USD Coin

4zMMC9...cDU

Same name, same logo. Which one is Circle's devnet USDC?

Which one is USDC?

Teaching moment straight from the handbook. Orca's devnet examples use their own devUSDC mint (BRjpCH...k1k). Circle's devnet USDC from [faucet.circle.com](https://faucet.circle.com) is a different mint (4zMMC9...cDU). Both are "USDC" on devnet. You cannot swap one for the other 1:1, because they are different tokens.

The third card is made up for this slide, which is the point: anyone can create a token with any name and logo. Scam tokens copy names and logos all the time.

Rule: the mint address is the identity. Check it against the issuer's own documentation. On mainnet, Circle's USDC is EPjFWdd5AufqSSqeM2qN1xzybapC8G4wEGGkZwyTDt1v.

15

Two token programs

SPL Token

The original. Mint, transfer, burn, freeze.

USDC, most older tokens.

Token-2022

Same basics, plus optional extensions.

PYUSD, USDC, many newer tokens.

Transfer fee

Transfer hook

Permanent delegate

Non-transferable

Confidential transfer

Metadata

Both programs have a mint authority and a freeze authority. Check who holds them before you buy anything.

Two token programs

Extensions, in plain words:

- Transfer fee: the token takes a cut of every transfer.
- Transfer hook: every transfer also calls another program, which can block or log it.
- Permanent delegate: one address can move or burn anyone's balance of this token, forever. Useful for regulated assets (a court order), dangerous in a random memecoin. Red flag unless you know why it's there.
- Non-transferable: soulbound, can't be sent.
- Confidential transfer: hides amounts.
- Metadata: name, symbol and logo stored on the mint.

Mint authority: can create more tokens (dilution). Freeze authority: can freeze any holder's account. For USDC, Circle holds both, which is expected for a regulated stablecoin. For a memecoin, either one is a warning. Week 5 turns this into a checklist.

16

Where does your USDC balance actually live?

A Inside your wallet account

B Inside the USDC mint account

C In your own USDC token account

D In the Token Program's code

Quiz: Where does your USDC balance actually live?

No notes. Let the slide speak.

17

Transactions

What you actually sign.

Transactions

No notes. Let the slide speak.

18

side a transaction

Sending 5 USDC to Bob, who has never held USDC. However each part.

signatures

40Pp...a11g (youme)

messages

header

1

0

6

accounts

5 The signer: wallet

1 Your USDC account: wallet

2 Bob: wallet

3 USDC: not used

4 USDC: not used

5 Token Program: program

6 System Program: program

recent blockhash

9z...c...4kL

Instruction 1

Instruction 2

Instruction 3

Account list

Every account the transaction will touch, listed up front. This is what Solana runs transactions in parallel.

A transaction is a list of instructions plus the signatures that authorise them.

- The first signature is the transaction's ID.
- Every account is listed up front, with flags: does it sign, is it written to. This is what lets the runtime run unrelated transactions in parallel (week 1).
- The recent blockhash expires after about 150 blocks, roughly a minute. That's why a transaction you left in a wallet popup for five minutes fails.
- Instructions run in order, and all succeed or the whole transaction fails. That all-or-nothing property is called atomic.

## The life of a transaction

The diagram illustrates the 6 steps of a transaction. A red bar at the top indicates the transaction is 'In Progress'. The steps are:

- Build**: The app fills the instructions and amounts.
- Sign**: Your wallet shows what will go, you approve.
- Send**: Sent to an SPV node, then forwarded straight to the next node(s). You mempool.
- Execute**: The leader type 1 is a "confirming node". Could still be instructions succeed, or none do.
- Processed**: It is a block or one node. Could still be dropped.
- Confirmed**: Transactions holding 25% of stake voted for the block.
- Finalized**: 51+ blocks built on top. Transaction is permanent.

Send: the signed transaction goes to an RPC node, which forwards it directly to the validators scheduled to be leader soon. Solana doesn't use a shared mempool the way Ethereum does.

Processed, confirmed, finalized are the three "commitment levels" you'll see in explorers and APIs.

- Alpenglow, a new consensus protocol aimed at much faster finality, was approved by validator vote on 2 Sep 2025 (SIMD-0326, 98% yes). It went live on testnet (24 Sep 2026) and devnet (25 Sep 2026), and mainnet was scheduled for about 21 Oct 2026 (epoch 1064). Check [solana.com/upgrades/alpenglow](https://solana.com/upgrades/alpenglow) before quoting finality times: after it ships, the commitment levels on this slide change.

## 20 at it costs

| Item                | Unit                      | Cost (USD) | Cost (DOT)       |
|---------------------|---------------------------|------------|------------------|
| Signatures          | 1                         | 0.000005   | 0.000005         |
| Compute limit       | 200,000 CU                | 0.000002   | 0.000002         |
| Priority price      | 10,000 $\mu$ -tamparts/CU | 0.0023928  | 0.0023928        |
| SOL price (example) | \$150                     |            |                  |
| <b>Total</b>        |                           |            | <b>0.0023928</b> |

☒ Creates the recipient's token account

Drag the sliders.

Priority fee: optional. Compute limit times the price you set per compute unit (in micro-lamports). All of it goes to the validator. Used when the network or a hot account is busy.

Finance connection: at these costs, a \$5 trade is worth doing. On Ethereum mainnet in busy periods, gas could be several dollars, which killed small trades entirely. Fees shape who can use DeFi.

Who pays is a choice. Every transaction names a fee payer, and it doesn't have to be the user: an app can sign as fee payer and cover fees for its users. Contest and bounty apps, games and onboarding flows use this so newcomers can act without holding any SOL. The catch: the sponsor pays for spam too, so sponsored apps rate-limit or require a login.

21. A swap fails because the price moved too much. What happens to the fee?

- A Refunded automatically
- B Charged anyway
- C Charged only if you retry
- D Paid by the DEX

*No notes. Let the slide speak.*

## 22 Adding it on an explorer

https://etherscan.io/tx/0x7a71731c6a87173

Signature  
0x7a71731c6a87173

Result  
Success

From  
0.000005 SOL

To  
2 signatures, no priority fee

Gas  
14,000,000,000

Token balance changes  
You  
-0.000005 SOL  
BANK USDC account (new)  
+0.000005 SOL

Token balance changes  
You  
USDC  
-1.00  
Bank USDC  
+5.00

Instructions  
#1 Associated Token Program : create (idempotent)  
#2 Token Program : transferChecked 5.000000 USDC

Everything an analyst needs is on one page. Write down what you can and can't interpret.

Sample data for teaching.

Point out the SOL balance changes: you paid 0.000005 SOL in fees and 0.00203928 SOL as the deposit for Bola's new USDC account. Total 0.00204428, shown rounded.

Orb's "Explain with AI" button is useful for a first read. Always check it against the raw instructions.

# 23

---

## This week's task

Your first devnet transaction.

*No notes. Let the slide speak.*

## 24 The task

### Users

1. Install **Phantom** (or Solflare, Backpack). Turn on **Testnet Mode** and join **Solana Devnet**. Check the banner.
2. Get devnet SOL: [faucet.solana.com](https://faucet.solana.com)
3. Get devnet USDC: [faucet.circle.com](https://faucet.circle.com)
4. Send USDC to a partner.
5. Open it on **Orb** with `testnet:devnet`. Find the fee, the token instruction, the token account it created.

### Builders

- Everything on the left, plus:
1. Create your own token with the `spl-token` CLI.
  2. Mint 1,000 to yourself.
  3. **Revoke the mint authority.**
  4. Look at the mint account again. What changed?

## The task

Phantom: Settings → Developer Settings → Testnet Mode, then choose Solana Devnet. A banner shows when testnet mode is on.

Circle's devnet USDC mint:

4zMMC9srt5Ri5X14GAgXhaHii3GnPAEERYPJgZJDncDU. Have mentees confirm the mint in Orb.

If the SOL faucet rate-limits the room (it limits by IP), send each mentee 1 devnet SOL from the teacher wallet you funded before class.

Pair people up for step 4 so everybody both sends and receives.

## 25 Builders: your own token

```
1 solana airdrop --url devnet
2 solana airdrop --url devnet --amount 1000000000000000000
3
4 spl-token create-token
5 P = pubkey your dev SOL address
6
7 spl-token create-account <P>
8 P = pubkey your token address (ATA) for 10
9
10 spl-token mint <P> 1000000000000000000
11 spl-token revoke <P>
12
13 spl-token authorize <P> mint --disable
14 spl-token display <P> -f mint authority (dev net)
```

## Builders: your own token

Step through with clicks.

After revoking the mint authority, nobody can ever mint more of this token. That's a one-way door. Revoking is what serious token launches do to prove a fixed supply, and it's the first thing to check on any token you're thinking of buying (week 5).

If `solana airdrop` is rate-limited, use [faucet.solana.com](https://faucet.solana.com) in a browser with the address from `solana address`.

Optional extension: create a Token-2022 token with `spl-token create-token --program-2022` and add an extension like a transfer fee.

## 26

### Blog post

#### "How I interacted with my first Solana dApp."

Include one transaction link and an annotated explorer screenshot.

## Blog post

*No notes. Let the slide speak.*

## 27

### Teacher prep

For the mentor. Skip in class.

## Teacher prep

*No notes. Let the slide speak.*

## 28 You, before class

- Make a **fresh wallet** and run the whole task on it. Note which faucet rate-limits you.
- **Pre-fund a teacher wallet** with devnet SOL and USDC. Faucets break when a whole room shares one Wi-Fi.
- Prepare one whiteboard diagram: **wallet → token account (ATA) → mint**. Weeks 3 and 4 build on it.
- Read Solana's intro to transactions in the official docs.

#### Faucet limit

Circle's faucet gives 20 USDC every 2 hours per address (confirmed in Circle's FAQ, 14 Oct 2026). Plan the transfer task around that.

## You, before class

*No notes. Let the slide speak.*

|                     |                                                      |             |                                                      |
|---------------------|------------------------------------------------------|-------------|------------------------------------------------------|
| Keypair             | A secret key and the public key made from it.        | Address     | A public key written in base58.                      |
| Recovery phrase     | 12 or 24 words that rebuild every key.               | Signature   | Proof that a key holder approved this exact message. |
| Account             | Imports, data, owner, executable. Everything is one. | Lamport     | One billionth of a SOL.                              |
| Rent                | A refundable deposit for an account's storage.       | Mint        | The account that defines a token.                    |
| Token account (ATA) | Holds one wallet's balance of one token.             | Instruction | One call to one program inside a transaction.        |
| Blockhash           | Recent block ID that makes a transaction expire.     | Devnet      | Solana's free test network for apps.                 |

Words from week 2

No notes. Let the slide speak.