Lesson 5

Wallets

Number	Wallet Name	Website	Staking Support
1	Binance Extension Wallet	https://binance-wallet.gitbook.io/binance-chain-wallet/	Yes
2	BNB Chain List	https://www.bnbchainlist.org/	No
3	Trust Wallet	https://trustwallet.com/	Yes
4	Math Wallet	https://mathwallet.org/en-us/	Yes
5	SafePal	https://safepal.io/	No
6	TokenPocket	https://www.tokenpocket.pro/	No
7	Arkane	https://arkane.network/	No
8	MetaMask	https://metamask.zendesk.com/hc/en-us	No
9	Ledger	https://www.ledger.com/	Yes
10	Trezor	https://wallet.trezor.io	No

Transaction introduction

Some practical points about transaction selection

- Miners choose which transactions to include in a block
- Miners can add their own transactions to a block
- Miners choose the order of transactions in a block
- Your transaction is in competition with other transactions for inclusion in the block

We will talk about the consequences of these points in our MEV lesson.

Transaction Processing

Before the transaction executes it needs to pass some validity tests

- The transaction follows the rules for the encoding scheme (now Simple Serialize SSZ, previous RLP)
- The signature on the transaction is valid.
- The nonce on the transaction is valid, i.e. it is equivalent to the sender account's current nonce.
- The gas_limit is greater than or equal to the intrinsic_gas used by the transaction.
- The sender's account balance contains the cost required in up-front payment.

(For details of SSZ see docs)

Token Standards

Use of tokens

From BNB Documentation

- **DeFi Tokens:** DeFi protocols and dApps are aimed at reproducing traditional financial system functions (lending, borrowing, saving, insurance, trading, etc.).
- **Utility Tokens:** These types of tokens are designed to serve particular purposes within a specific application/protocol's ecosystem and allow users to be part of the decision-making on a specific network. #
- **Governance Tokens:** These specialised tokens give holders the right to vote on issues that govern the development and operations of a blockchain project. #
- **Non-Fungible Tokens (NFTs):** NFTs are used to represent ownership rights to a unique digital or real-world asset. These allow the tokenisation of real-world things.
- **Security Tokens:** are a new class of assets that aim to be the crypto equivalent of traditional securities like stocks and bonds.

Original Ethereum Standards

- ERC20: the most widespread token standard for fungible assets, albeit somewhat limited by its simplicity.

- ERC721: the de-facto solution for non-fungible tokens, often used for collectibles and games.

- ERC777
  See Documentation
  These are backwards compatible with ERC20, in addition there are

advanced features such as send / receive hooks, and operators (who can send tokens on behalf of others).

- ERC1155:
  A single contract may include any combination of fungible tokens, non-fungible tokens or other configurations (e.g. semi-fungible tokens).

BNB Standards

See Documentation

Similar to ERC, BNB Chain defines its own set of standards for token issuance, management, and implementation known as BEP (BNB Evolution Proposals). BEPs are token management rules and pre-defined criteria for launching on-chain assets on BNB Chain. The most popular BEP standards are BEP2 and BEP20. BEP2 is the native coin of the Beacon Chain. Whereas BEP20 is popular for use on BSC. It is to note here that BEP20 is very similar to ERC20 and extends its functionality. Note that BNB, which is the native token of the BNB Chain ecosystem, is a BEP2 token.

Comparison of BEP20 and ERC20

ERC20	BEP20
Gas fees are paid for performing any transaction on the Ethereum blockchain. Ethereum has higher gas costs as compared to the BNB Chain.	Gas fees collected to create BEP20 tokens are comparatively lower than ERC20 tokens.
ERC20 transactions are slower, as up to 15 seconds are required for them to take place.	Transaction speed is 5 times faster as BEP20 transactions take up to 3 seconds to take place.
Proof-of-Stake (PoS) algorithm is used for the verification and validation of ERC20 transactions, ensuring higher security.	BEP20 token also offers high security and uses Proof-of-Staked-Authority (PoSA) for verification and validation of transactions.
ERC20 tokens are mostly used for crypto crowdfunding, trading, staking, etc.	BEP20 tokens are mostly used for crypto crowdfunding, trading, staking, etc.

BEP20 tokens are designed to be compatible with BEP2 and ERC20. It extends ERC20 for compatibility with EVM chain and Ethereum smart contracts.

BEP20 Specification

- Methods
  - name
  - symbol
  - decimals

- totalSupply
- balanceOf
- getOwner
- transfer
- transferFrom
- approve
- allowance

- Events
  - Transfer
  - Approval

Solidity Part 3

View Functions and modifying state

From documentation

The following statements are considered modifying the state:

1. Writing to state variables.
2. Emitting events
3. Creating other contracts
4. Using `selfdestruct`.
5. Sending Ether via calls.
6. Calling any function not marked `view` or `pure`.
7. Using low-level calls.
8. Using inline assembly that contains certain opcodes.

Note : Getter methods are automatically marked `view`.

Fallback and Receive functions

receive() external payable { ... }
Called when the contract receives BNB

fallback () external [payable]
Called if a function cannot be found matching the required function signature.
It also handles the case when BNB is received but there is no receive function

Sending tokens to contracts

Contracts are accounts and so can have a balance like an externally owned account

Native tokens

BNB can be sent to a contract, but this can be controlled and reacted to.

If a function should accept BNB as well as the other inputs, it must be marked `payable` , note that you do not need to add any parameters to the function for the BNB, the amount is given in a globally available variable `msg.value`

If BNB is received by your contract, the contract can react in a number of ways.

1. If there is a `receive` function, this will be called, you can use this to for example emit an event to log that your contract has received funds.
2. If there is no `receive` function and there is a `fallback`function that will be called.
3. If neither are present, an error will be thrown.

However it is always possible to send funds to your contract by calling self destruct on an existing contract and specifying your contract as the recipient of the funds.

Other tokens

A contract address can have a balance associated with it, just like any other address.
However contracts cannot initiate transactions, so if a contract wants to control funds, it will need to have a function that can call the token contract.

It is good practice, though not often done, to have functions in your contract to transfer any native or non native tokens that may have accumulated by mistake.