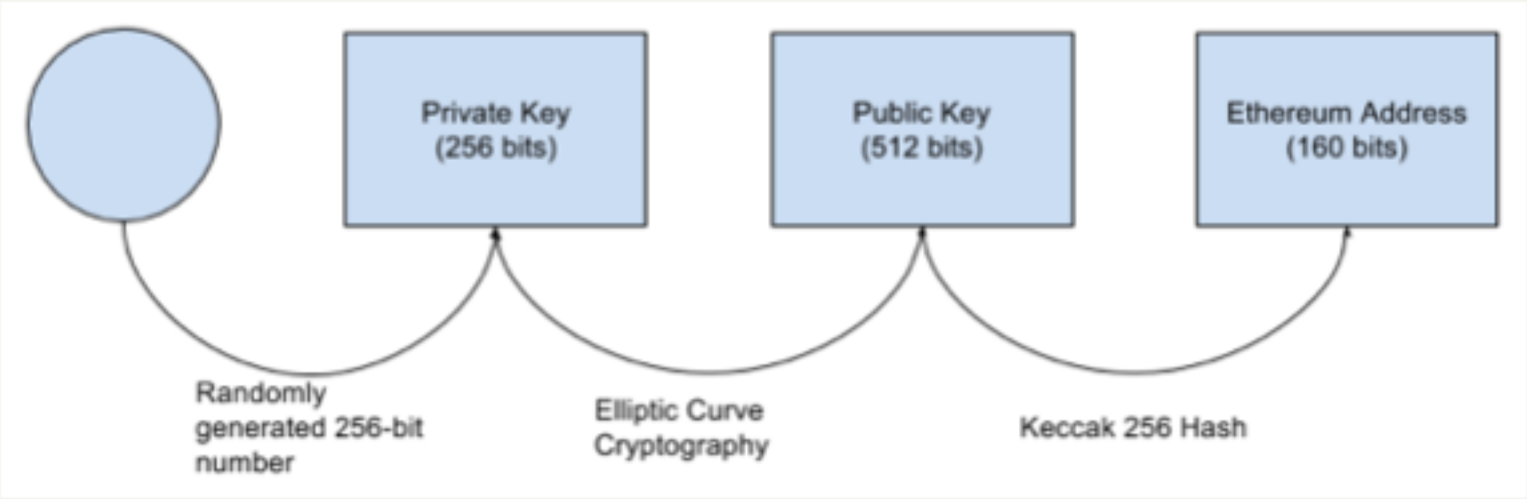
LAST SESSION’S Q&A

How to generate an Ethereum address?

- Step 1: Create a random private key (64 hex characters / 256 bits / 32 bytes)
- Step 2: Derive the public key from this private key (128 hex characters / 512 bits / 64 bytes)
- Step 3: Derive the address from public key by taking the last 40 hex characters / 160 bits / 20 bytes



The chance that 2 Ethereum addresses are duplicate

- One Ethereum address has 40 hex characters, each has 16 possibilities
- Hence, the total possibility addresses are 16 ^ 40 = 1461501637330902918203684832716283019655932542976
- Assume we have 6 billion Ethereum users and each user has 10 addresses, the chance that 2 addresses are duplicate is
  - *(10 * 6 000 000 000) / (16 ^ 40) * 100 % =* **0.0000000000000000000000000000000000041053665947 %**

How to find solidity code for a contract address

- No, you can’t unless the smart contract developers choose to reveal it.

Remix At Address

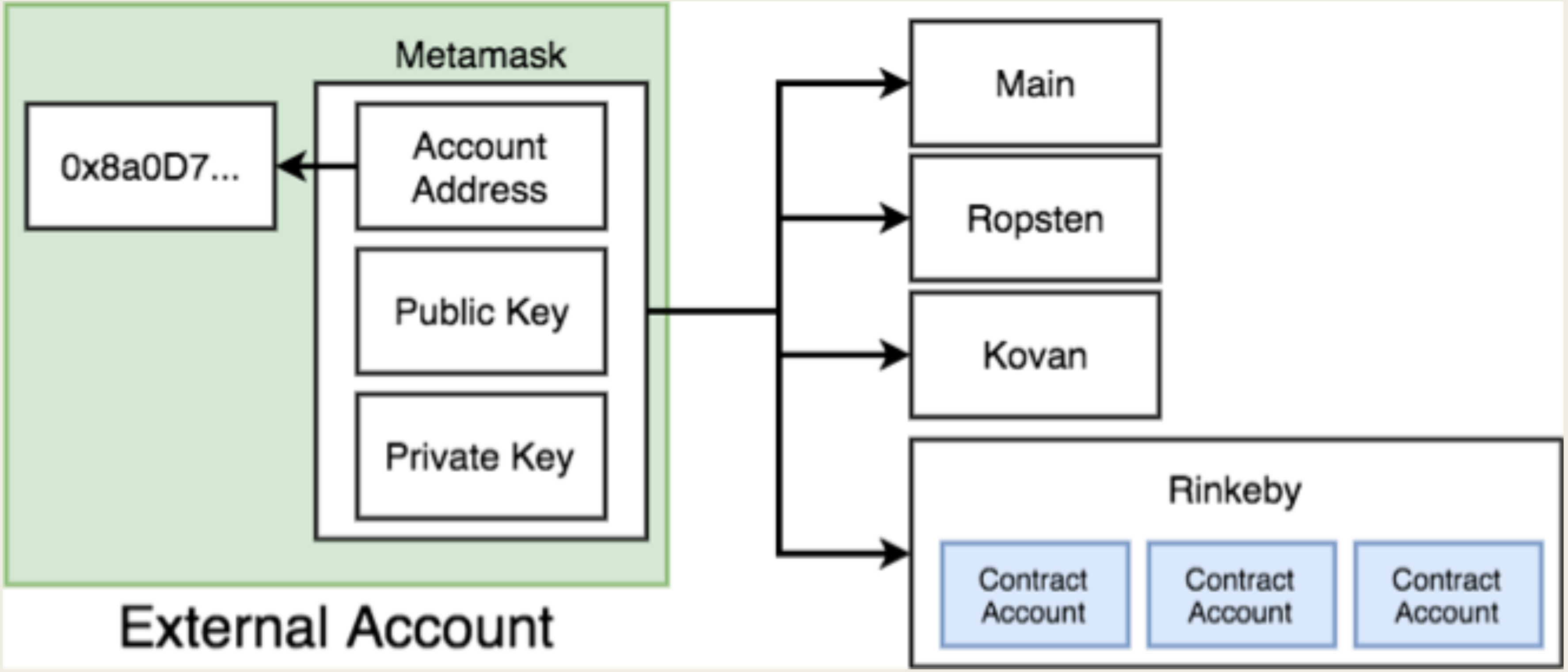
- You have to have ABI

GET TO KNOW EACH OTHER

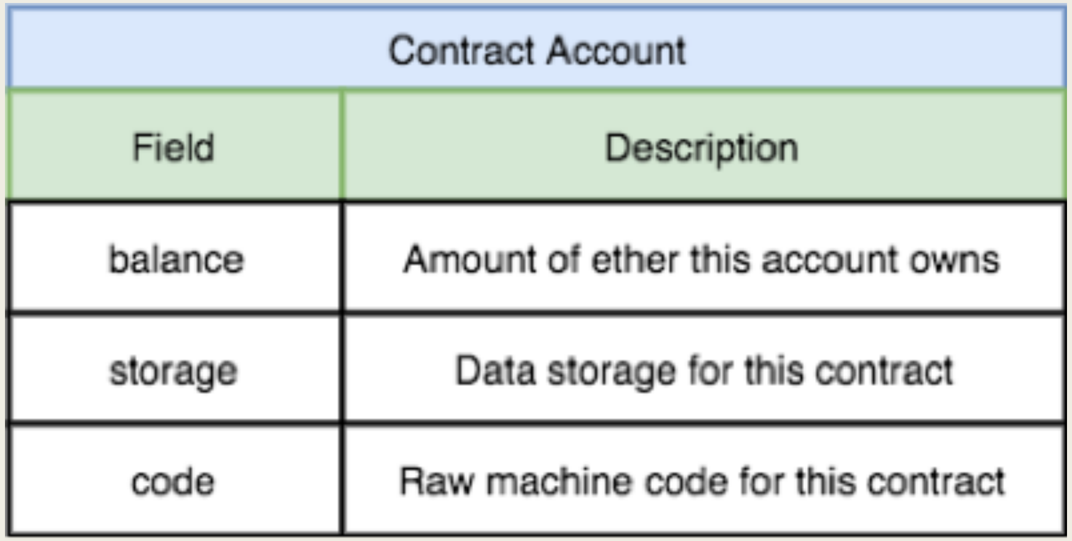
Day 2 Outline

- Let’s go over it again
  - *External Account vs Contract Account*
  - *Bytecode vs ABI*
  - *Inbox Revisit*
  - *Common Function Types*
  - *External Account to External Account Transaction*
  - *External to Create Contract Transaction*
  - *Calling a function vs Sending a Transaction to a function*
  - *Ether vs Wei vs other units*
- Demo – Project Directory
  - *Compiling*
  - *Testing*
  - *Deployment*
- Recap

External Account vs Contract Account

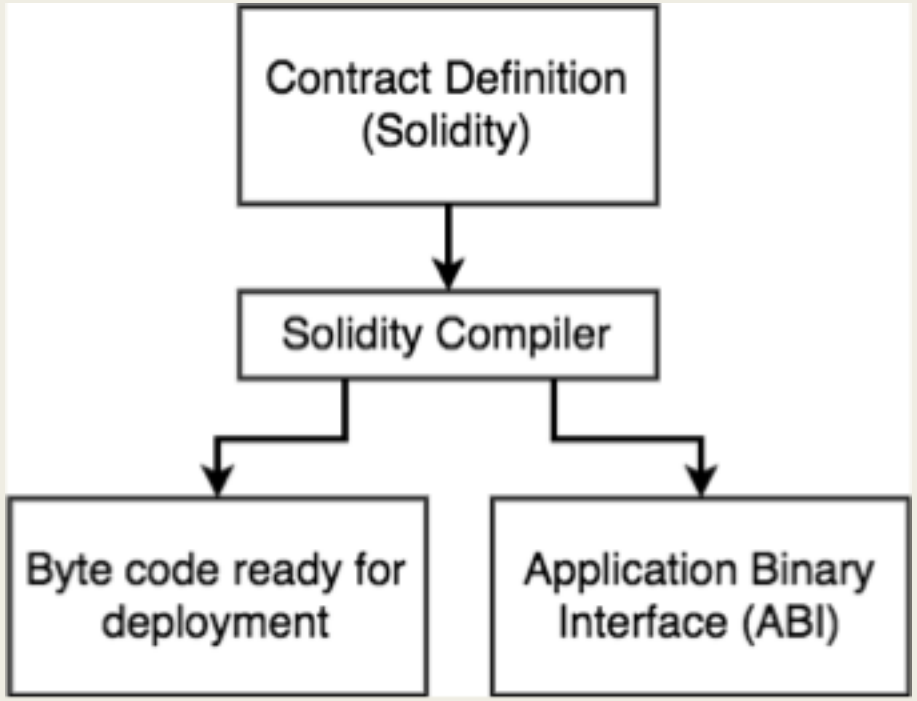


Contract Account



Contract Account	
Field	Description
balance	Amount of ether this account owns
storage	Data storage for this contract
code	Raw machine code for this contract

Bytecode vs ABI



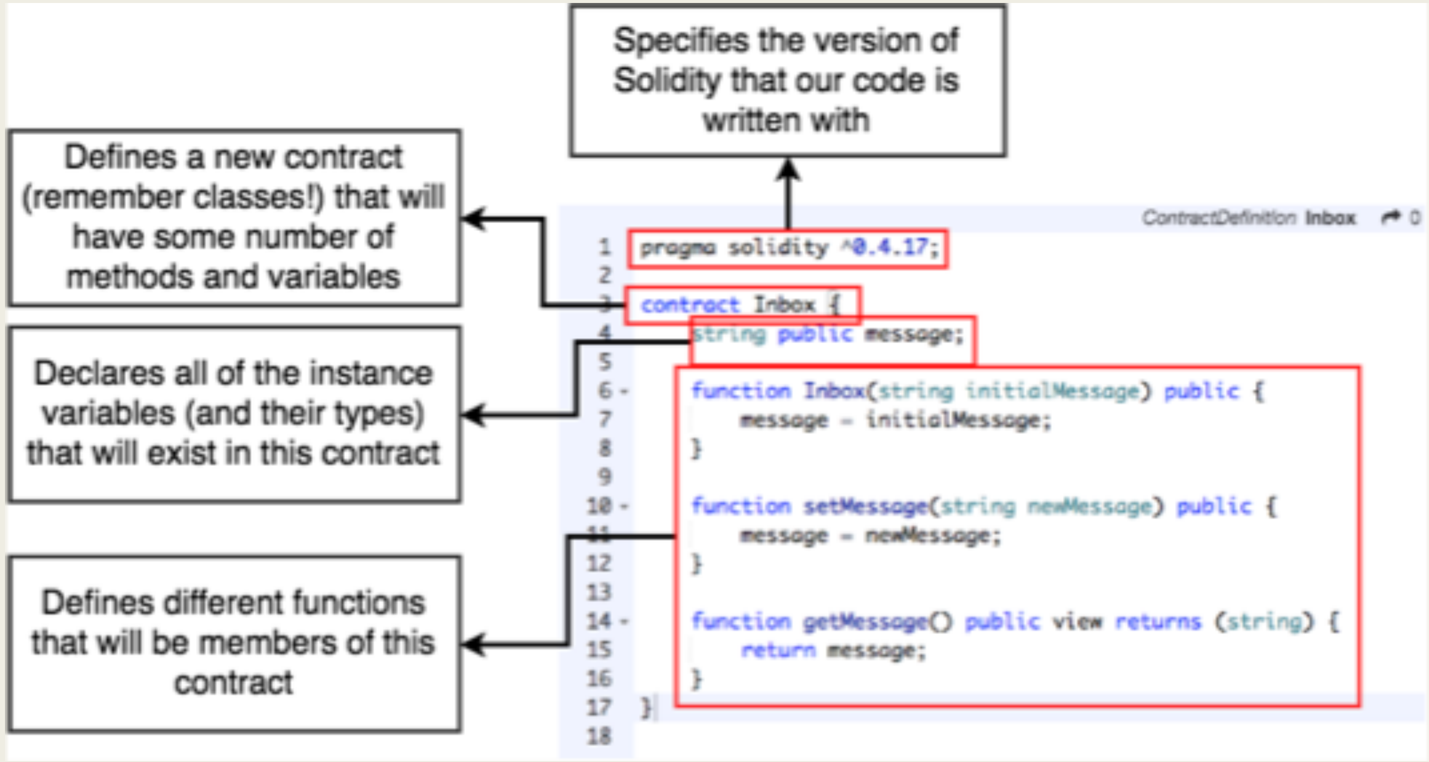
Contract Definition (Solidity)

Solidity Compiler

Byte code ready for deployment

Application Binary Interface (ABI)

Inbox Revisit – Smart contract



```
pragma solidity ^0.4.17;

contract Inbox {
    string public message;

    function Inbox(string initialMessage) public {
        message = initialMessage;
    }

    function setMessage(string newMessage) public {
        message = newMessage;
    }

    function getMessage() public view returns (string) {
        return message;
    }
}
```

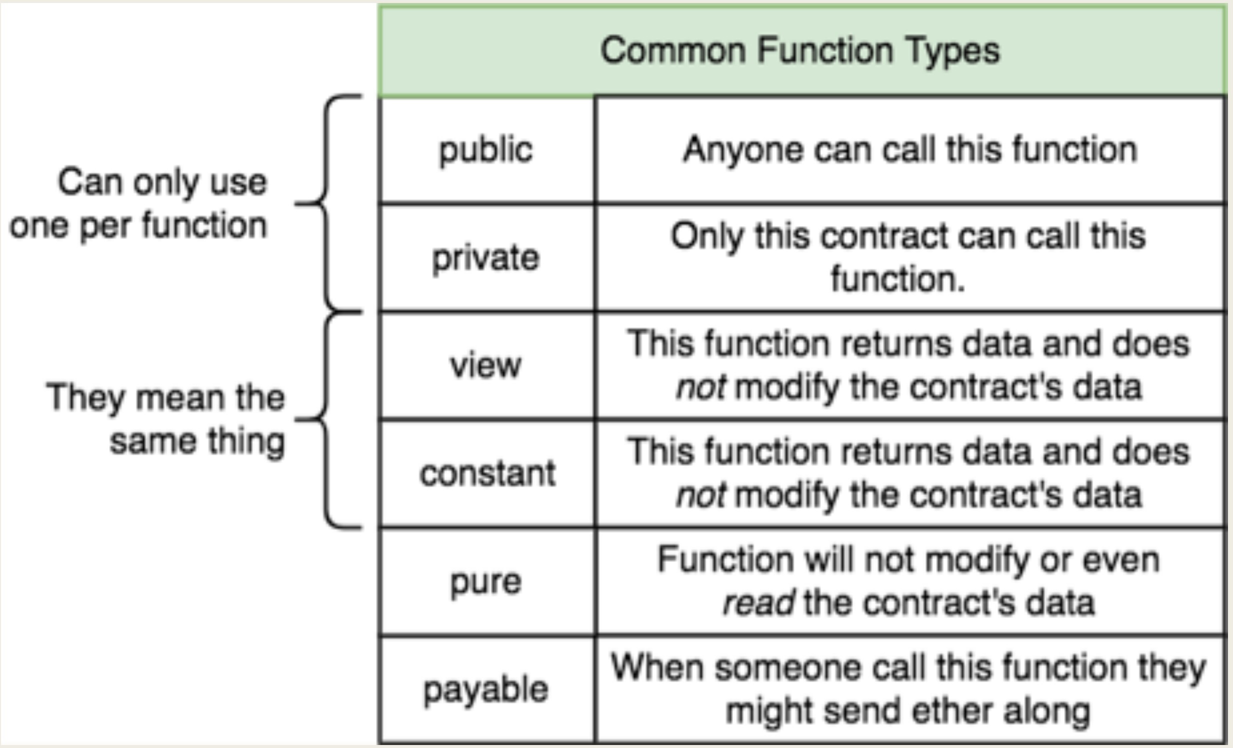
Specifies the version of Solidity that our code is written with

Defines a new contract (remember classes!) that will have some number of methods and variables

Declares all of the instance variables (and their types) that will exist in this contract

Defines different functions that will be members of this contract

Common Function Types

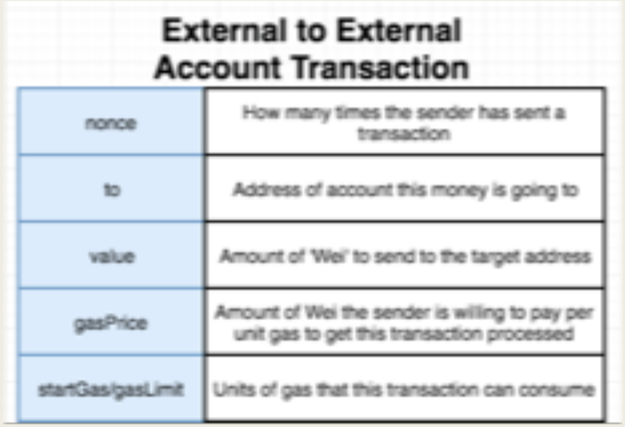


Can only use one per function

They mean the same thing

Common Function Types	
public	Anyone can call this function
private	Only this contract can call this function.
view	This function returns data and does *not* modify the contract's data
constant	This function returns data and does *not* modify the contract's data
pure	Function will not modify or even *read* the contract's data
payable	When someone call this function they might send ether along

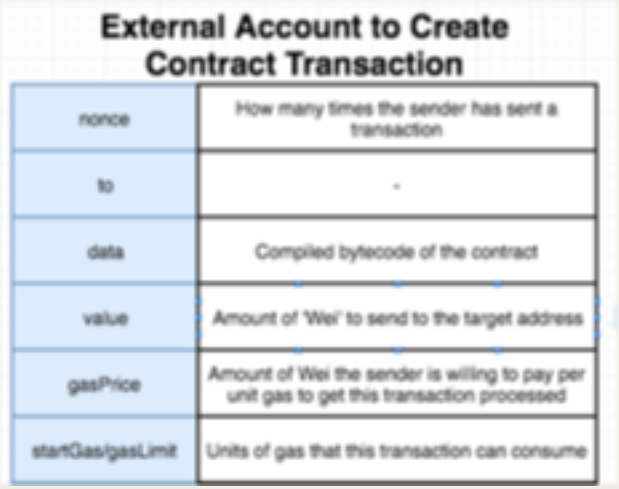
External account to external account transaction



External to External Account Transaction

nonce	How many times the sender has sent a transaction
to	Address of account this money is going to
value	Amount of 'Wei' to send to the target address
gasPrice	Amount of Wei the sender is willing to pay per unit gas to get this transaction processed
startGas/gasLimit	Units of gas that this transaction can consume

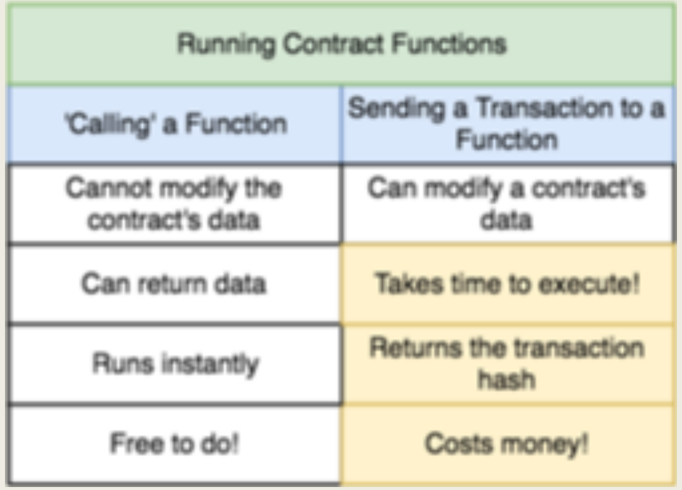
External account to create account transaction



External Account to Create Contract Transaction

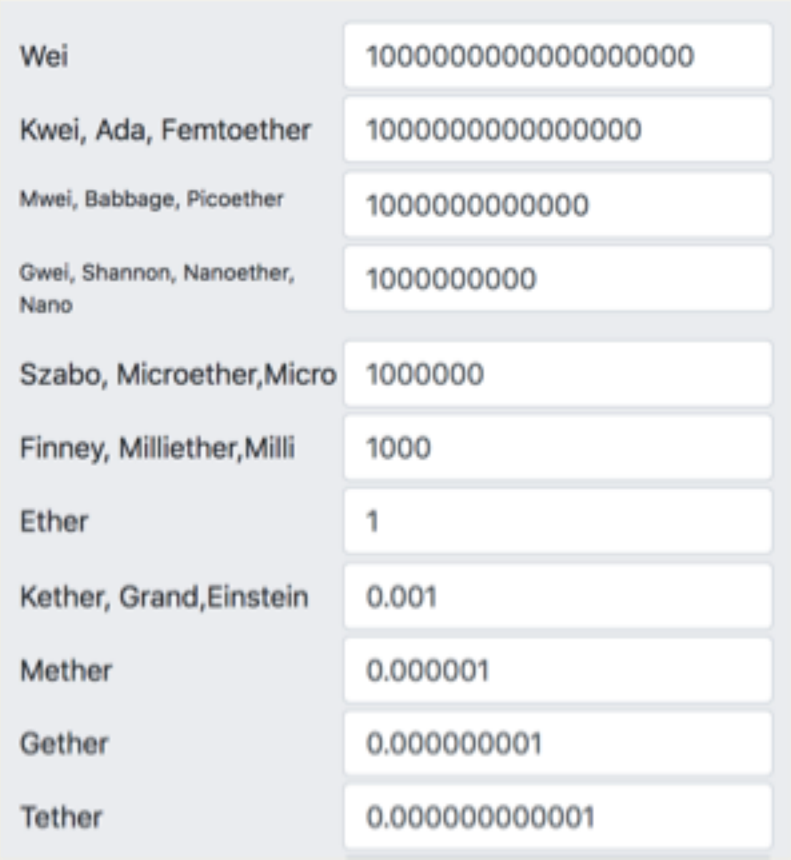
nonce	How many times the sender has sent a transaction
to	-
data	Compiled bytecode of the contract
value	Amount of 'Wei' to send to the target address
gasPrice	Amount of Wei the sender is willing to pay per unit gas to get this transaction processed
startGas/gasLimit	Units of gas that this transaction can consume

Calling a function vs Sending a transaction to a function



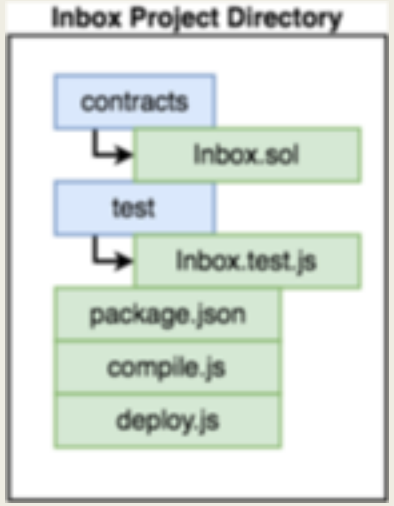
Running Contract Functions	
'Calling' a Function	Sending a Transaction to a Function
Cannot modify the contract's data	Can modify a contract's data
Can return data	Takes time to execute!
Runs instantly	Returns the transaction hash
Free to do!	Costs money!

Ether vs Wei vs Other units



Wei	1000000000000000000
Kwei, Ada, Femtoether	1000000000000000
Mwei, Babbage, Picoether	1000000000000
Gwei, Shannon, Nanoether, Nano	1000000000
Szabo, Microether,Micro	1000000
Finney, Milliether,Milli	1000
Ether	1
Kether, Grand,Einstein	0.001
Mether	0.000001
Gether	0.000000001
Tether	0.000000000001

Project Directory



Compiling script

Talk is cheap.

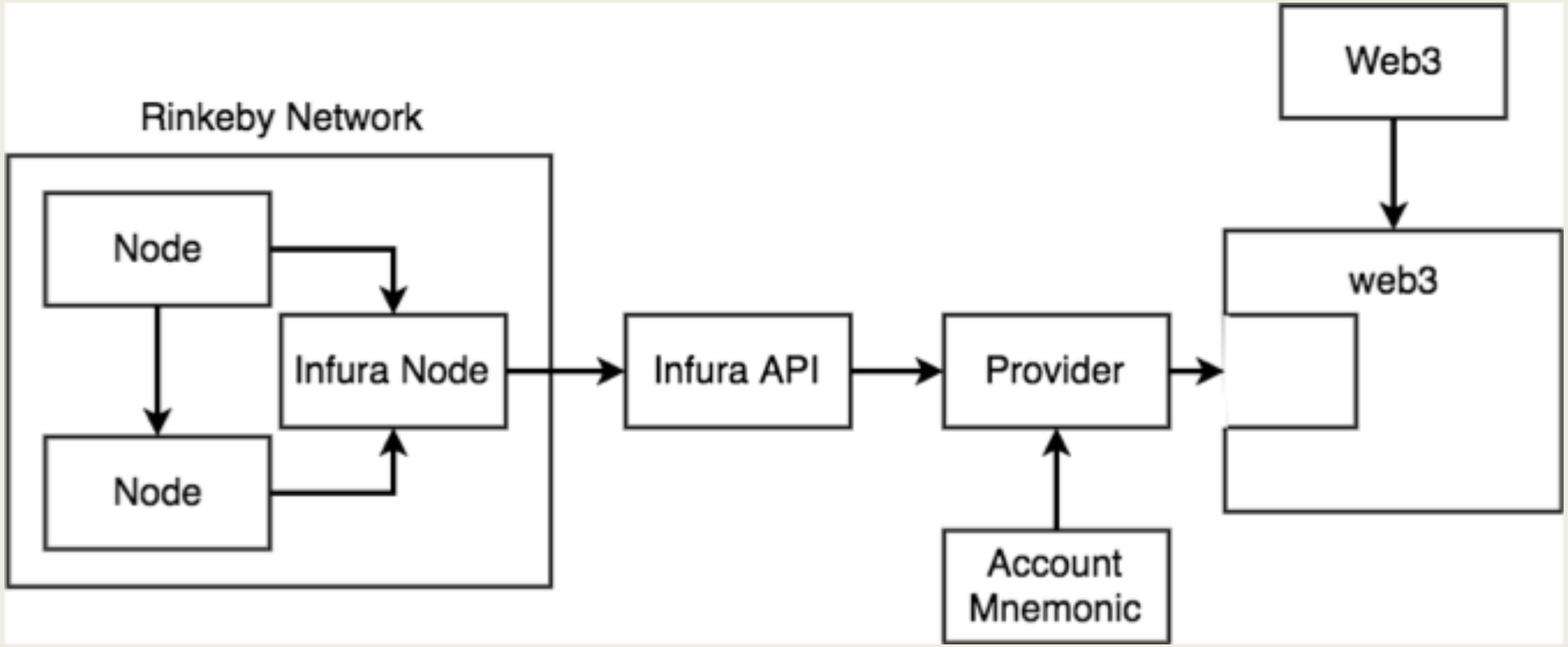
Show me the code.

Linus Torvalds

Testing

- ”npm install - - save mocha ganache-cli web3@1.0.0-beta.26”

Deployment


Web3
Rinkeby Network
Node
Infura Node
Node
Infura API
Provider
web3
Account Mnemonic

HD Wallet

- HD Wallet = Hierarchical Deterministic Wallet
- Mnemonic related

Recap

- Understand how Ethereum development processes look like