Smart contracts with Solidity

Ajinkya Rajguru

Deliverable 4

SJSU: FALL 2022

CS 297

Dr. Chris Pollett

Introduction

- Object Oriented high level language.
- Main purpose: Implementing smart contracts
- Written in .sol files
- Similar to Javascript
- Statically typed

Smart Contracts

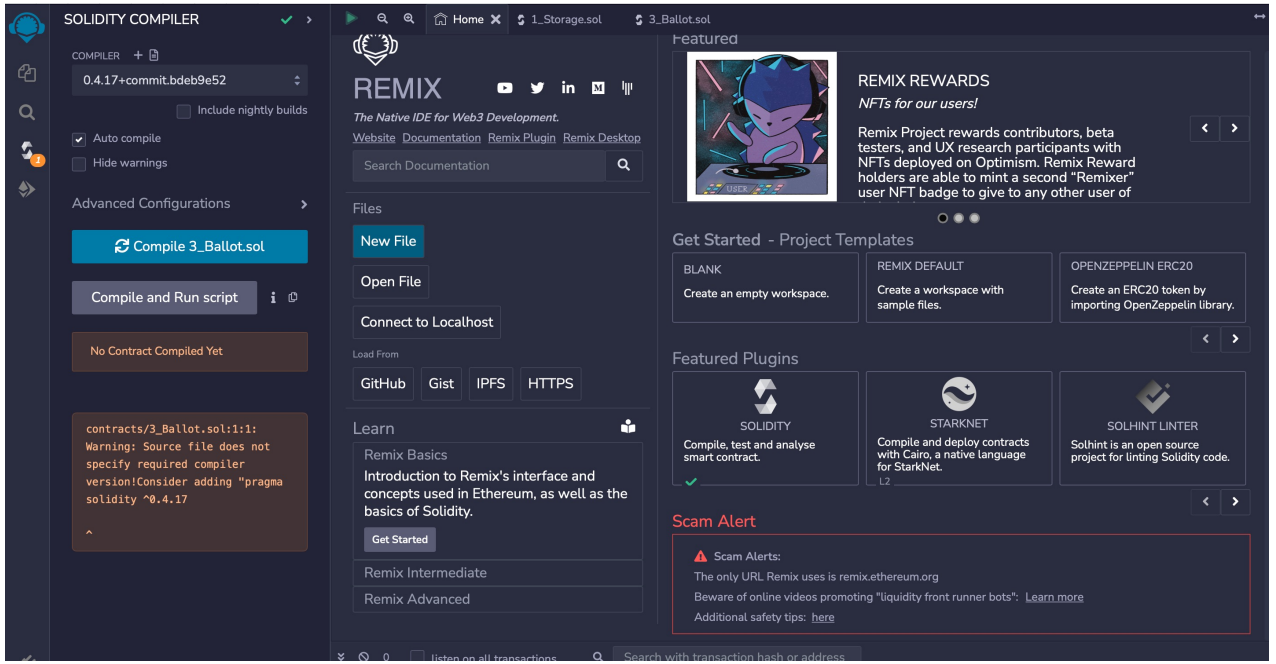
- An account present on a blockchain network which is controlled by code.
- Components of a contract account –
  1. Balance – Amount of ether the account owns
  2. Storage – Data storage for the contract (depends on the application)
  3. Code – Machine code for the contract

Solidity

1. Create contract definition using solidity
2. It is passed to the Solidity compiler
3. The compiler gives out two separate files –
    1. Byte Code which is ready for deployment – deployed into Ethereum network
    2. Application Binary Interface (ABI) – Used to interact with deployed smart contracts (example using Javascript for .js applications)

Remix IDE for Solidity [3]

- A web-based IDE to write .sol contracts
- Contains an inbuild solidity compiler and an editor
- Great for small contracts
- Also contains a mock Ethereum network to deploy and test contracts
- Basic flow of deploying a contract:

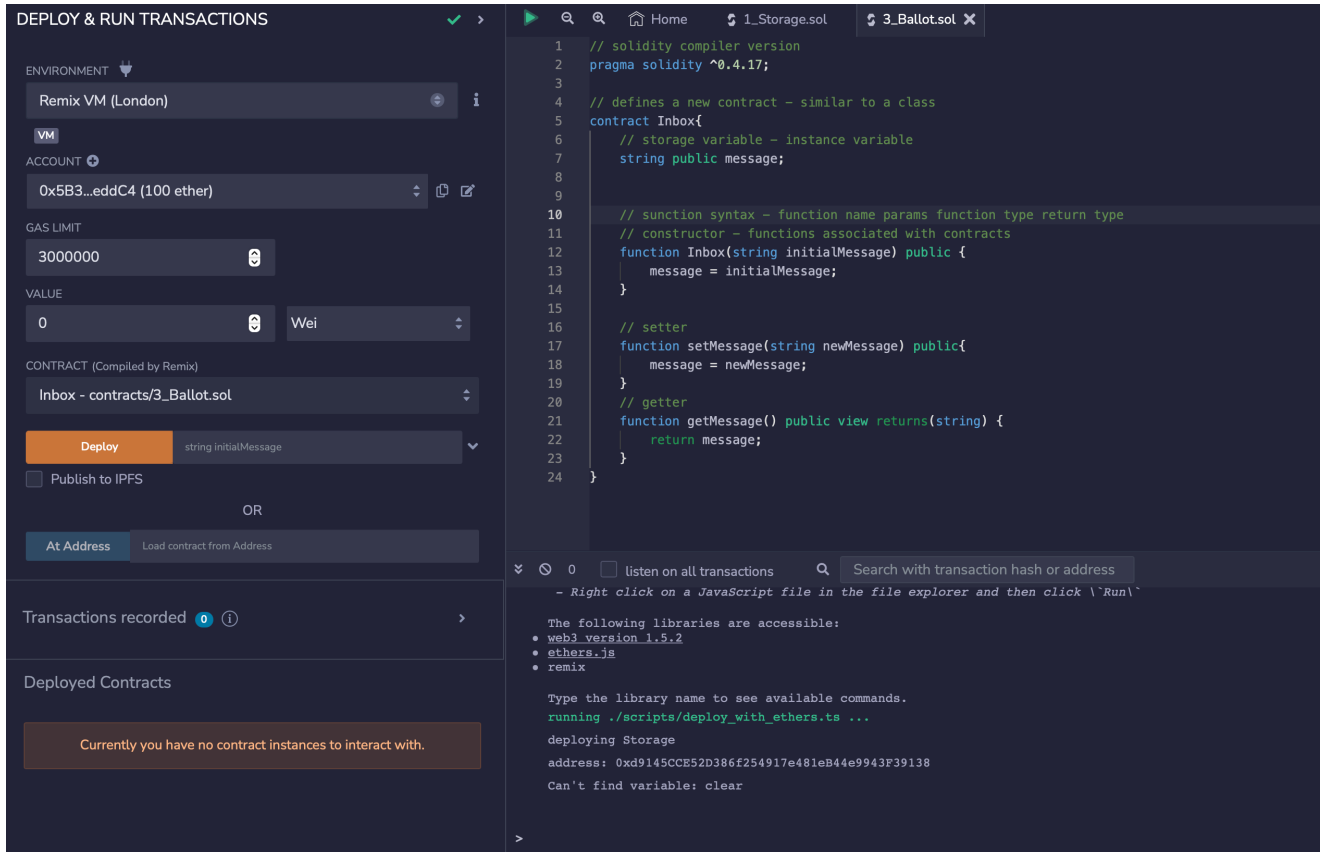


Solidity

1. Common function types include –
    1. Access modifiers – public and private
    2. Return type – View and Constant: The function returns data and does not modify the contracts data.
    3. Pure – Function will not modify or read the contracts data
    4. Payable – This function can include an ether along with its call.

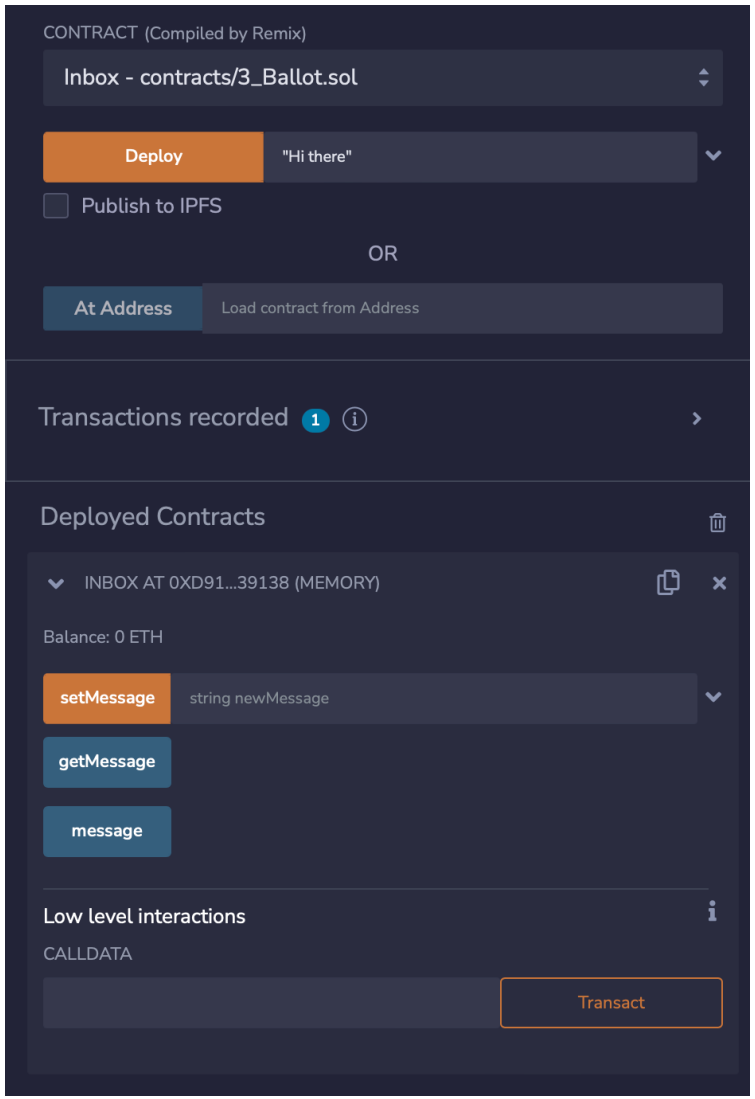
Solidity run and deploy configuration

- Environment – We use a Remix Virtual machine
- Account – Any account can be selected which are already preloaded with 100 ether coins for testing.
- Select a contract – here Inbox
- An empty box appears to put in parameters for the constructor defined by us in the contract



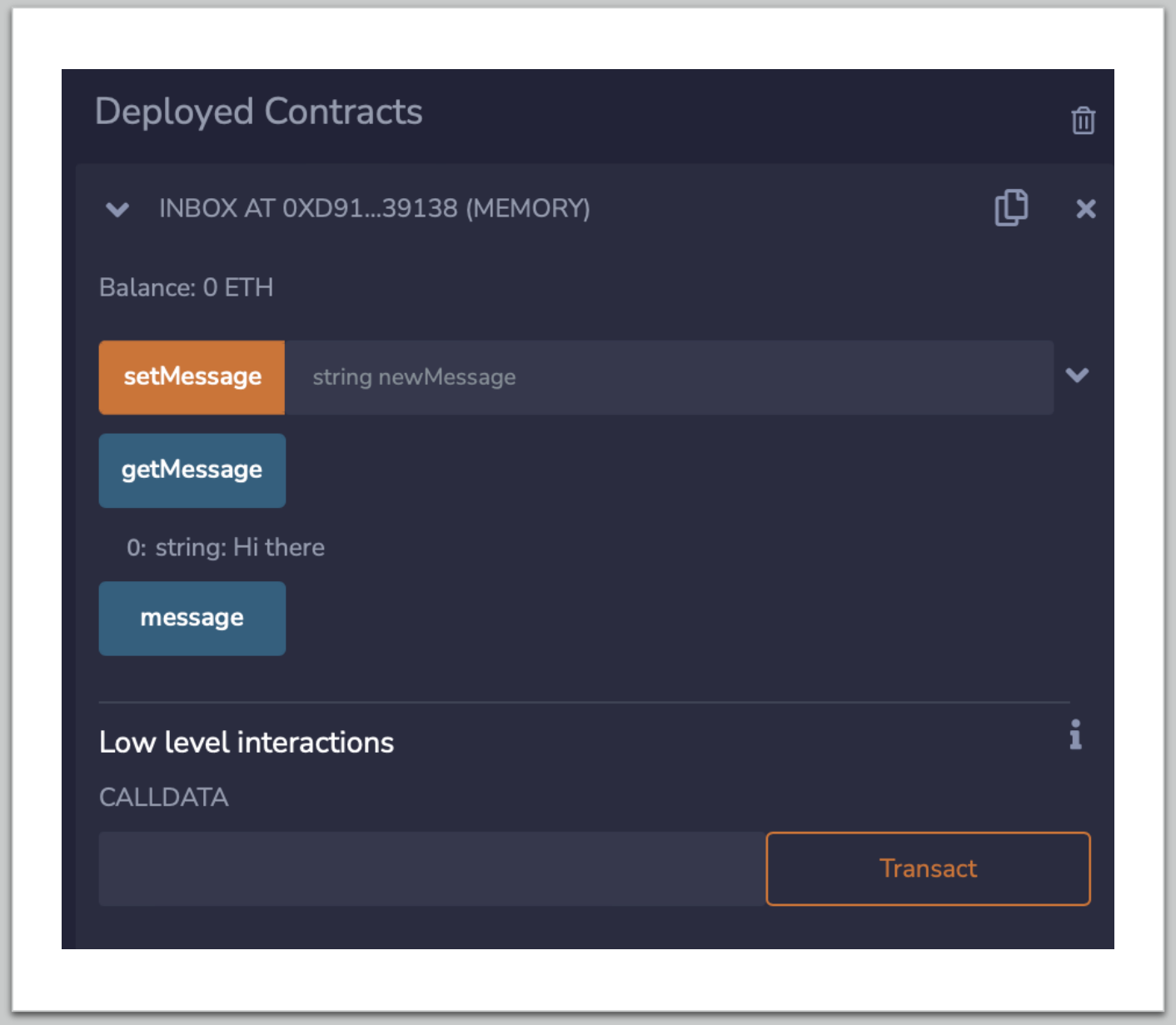
Inbox contract

- Deployed by inserting a string ”Hi there”.
- Below shoes the instance created.
- setMessage, getMessage and message buttons allow us to interact with the contract.



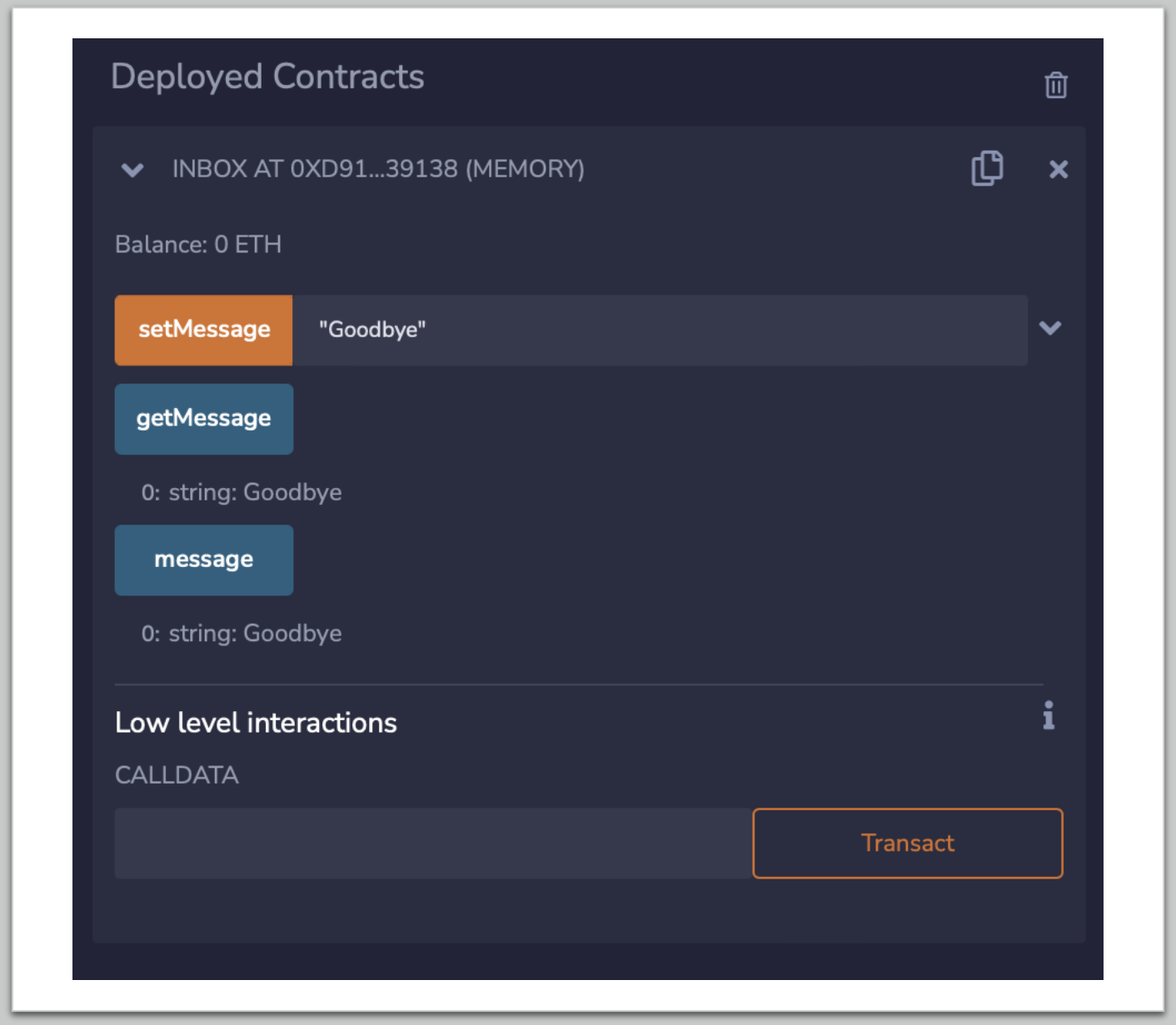
Inbox contract

- We receive Hi there after clicking the getMessage button.



Inbox contract

- setMessage as Goodbye
- We get the message by clicking on getMessage as well as message.
- message is a public variable which can also check the value for message



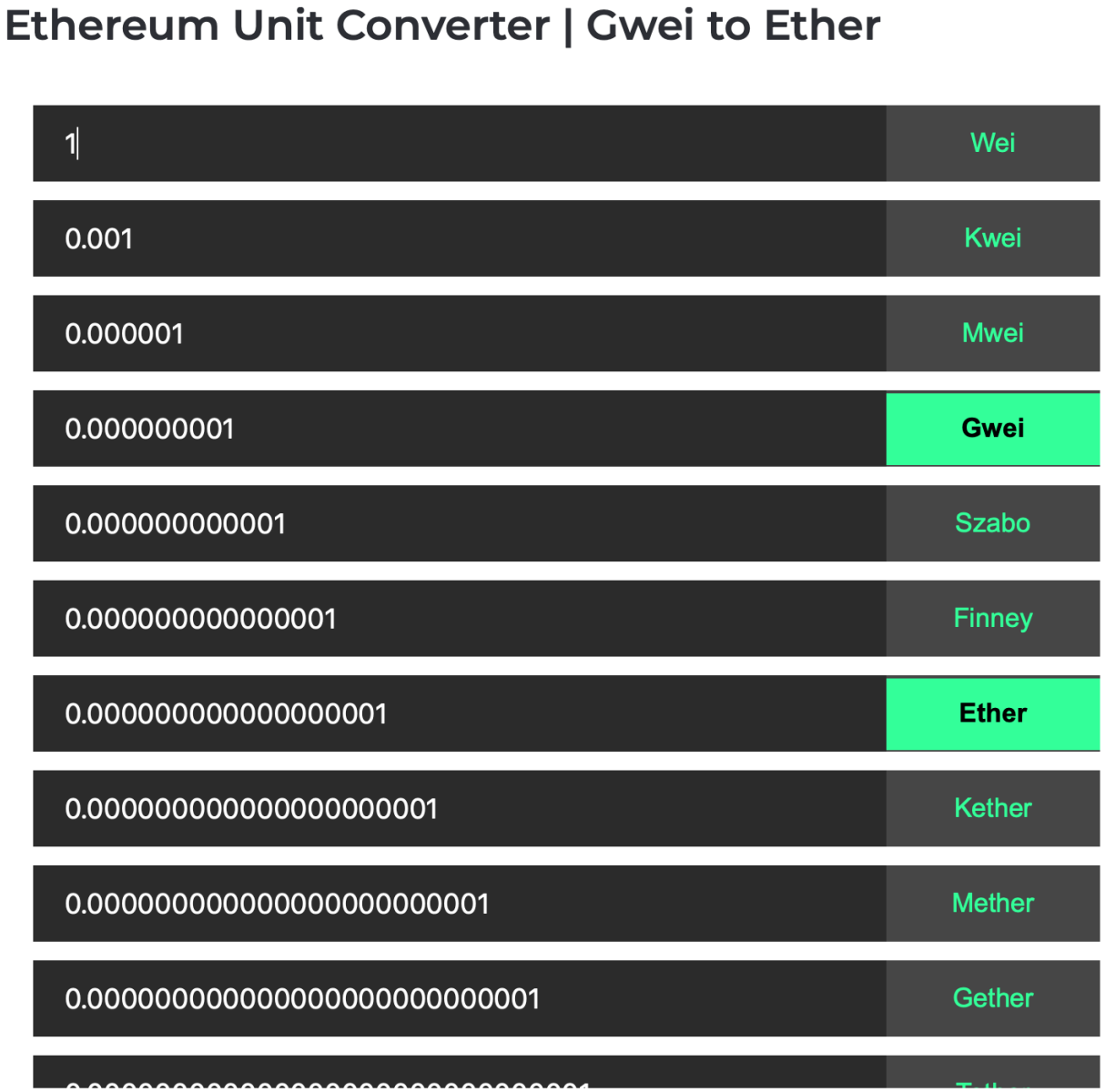
Inbox Contract - console

- Each function call produces the following console logs.
- This includes the receiver and sender along with the execution cost in gas unit

Creating a contract - What happens at the background

- Similar to transfer of money on a network.
- We create a transaction to create a contract
- Contract transaction contains:
  - Nonce – number of times the sender has sent a transaction
  - To – field is blank as opposed to while sending a money
  - Data – bytecode of contract (exposed to the world)
  - v, r and s – crypto pieces of data
  - Value – Amount of Wei (Wei is a smaller unit of ether – **1 ether =(10^{18}) wei**)
  - gas – cost to run our code on another machine: Gas cost sheet – [5] https://docs.google.com/spreadsheets/d/1n6mRqkBz3iWcOlRem_mO09GtSKEKrAsfO7Frgx18pNU/edit#gid=0
  - gasPrice – cost willing to pay for a transaction
  - gasLimit – The unit of gas this transaction can consume

[4] Ethereum unit converter



Reading and Modifying functions

- Reading Transactions –
  - Calling a function can include returning a data and does not include modifying the contract data.
  - Runs instantly and is free
- Sending Transactions –
  - Sending a transaction to a function can include modifying the contract data which returns the transaction hash.
  - Takes time to execute and costs money

References

[1] https://docs.soliditylang.org/en/v0.8.17/introduction-to-smart-contracts.html#simple-smart-contract

[2] https://www.udemy.com/share/1013Fs3@Qax1kH8XyDPQcM-COaQ_hYkoQetC7yzJJa_KJQC1oKYYhJXWlDonqhJw9-eXS_6-2Q==/

[3] https://remix.ethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v0.4.17+commit.bdeb9e52.js&language=Solidity

[4] https://coinguides.org/ethereum-unit-converter-gwei-ether/

[5] https://github.com/djrtwo/evm-opcode-gas-costs