



Lesson 4

FUND ME

Hikmah Nisya - 1103184094
Radzis Araaf Jaya Jamaludin - 1103184234
Raudhatul Rafiqah Assyahiddini - 1103180225

Preparing Tools

Beberapa hal yang perlu digunakan dalam lesson ini.

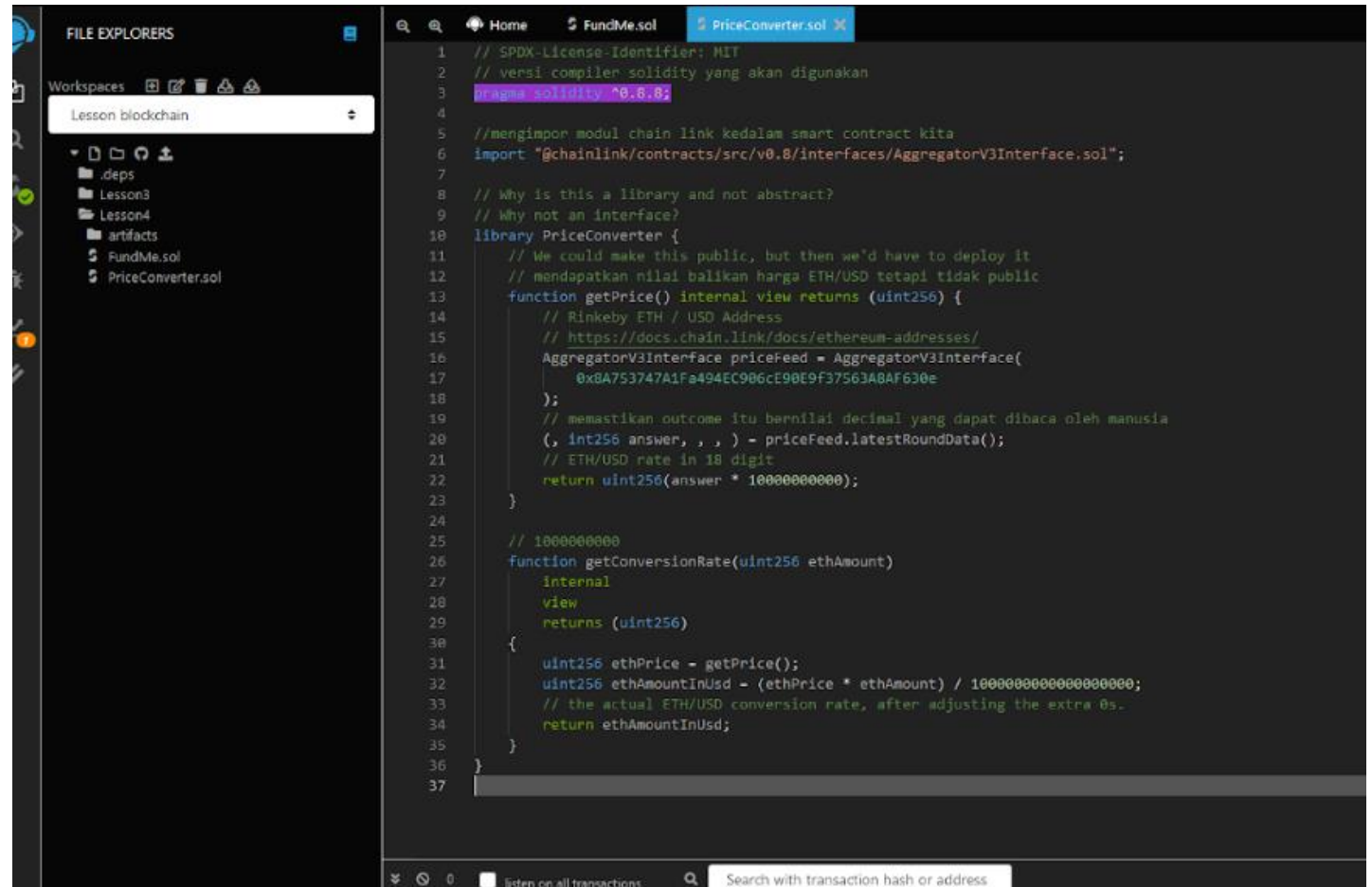
1. github : digunakan untuk kepentingan dalam menyimpan code dan menyimpan dokumentasi code yang telah kita buat.
2. Remix IDE : Remix IDE ini digunakan sebagai tempat code editor.
3. Solidity : Bahasa yang digunakan saat kita mempelajari Blockchain.
4. Waktu : kita membutuhkan banyak waktu untuk mempelajari blockchain, maka dari itu waktu adalah salah satu hal yang sangat dibutuhkan.

Membuat file pada REMIX

Pertama, membuka website resmi : <https://remix.ethereum.org>, lalu kita membuat workspace baru agar bersih dari template yang default diberikan oleh remix. Kemudian kita dapat membuat tiga file yang berekstensi “.sol” seperti:

1. FundMe.sol
2. PriceConverter.sol

Semua file ini kita akan gunakan seiring waktu.



The screenshot displays the Remix IDE interface. On the left, the 'FILE EXPLORERS' panel shows a workspace named 'Lesson blockchain' containing a folder 'Lesson3' and files 'FundMe.sol' and 'PriceConverter.sol'. The main editor area shows the 'PriceConverter.sol' file with the following Solidity code:

```
1 // SPDX-License-Identifier: MIT
2 // versi compiler solidity yang akan digunakan
3 pragma solidity ^0.8.0;
4
5 //mengimpor modul chain link kedalam smart contract kita
6 import "@chainlink/contracts/src/v0.8/interfaces/AggregatorV3Interface.sol";
7
8 // Why is this a library and not abstract?
9 // Why not an interface?
10 library PriceConverter {
11     // We could make this public, but then we'd have to deploy it
12     // mendapatkan nilai balikan harga ETH/USD tetapi tidak public
13     function getPrice() internal view returns (uint256) {
14         // Rinkeby ETH / USD Address
15         // https://docs.chainlink/docs/ethereum-addresses/
16         AggregatorV3Interface priceFeed = AggregatorV3Interface(
17             0x8A753747A1Fa494EC906cE90E9f37563A8AF630e
18         );
19         // memastikan outcome itu bernilai decimal yang dapat dibaca oleh manusia
20         (, int256 answer, , , ) = priceFeed.latestRoundData();
21         // ETH/USD rate in 18 digit
22         return uint256(answer * 10000000000);
23     }
24
25     // 1000000000
26     function getConversionRate(uint256 ethAmount)
27         internal
28         view
29         returns (uint256)
30     {
31         uint256 ethPrice = getPrice();
32         uint256 ethAmountInUsd = (ethPrice * ethAmount) / 1000000000000000000;
33         // the actual ETH/USD conversion rate, after adjusting the extra 0s.
34         return ethAmountInUsd;
35     }
36 }
37
```

Membuat FundMe

Kedua, mempersiapkan code pada “FundMe.sol” yang pertama akan memilih compiler dari soliditynya lalu kita akan mengimpor modul chain link kedalam smart contract kita agar dapat berinteraksi dengan luar blockchain. Lalu pada kontrak kita ada fungsi mapping ke uin256 ke dalam address nya dijadikan immutable agar tidak dapat diubah. Kitapun melakukan set minimal USD yang dapat dikirim ke dalam blockchain dan diubah menjadi ETH.

```
1 // SPDX-License-Identifier: MIT
2 //versi compiler solidity yang akan digunakan
3 pragma solidity ^0.8.0;
4
5 //mengimpor modul chain link kedalam smart contract kita
6 import "@chainlink/contracts/src/v0.8/interfaces/AggregatorV3Interface.sol";
7 //mengimpor code PriceConverter.sol
8 import "../PriceConverter.sol";
9
10 //hasil balikan error jika address didn't match
11 error NotOwner();
12
13 //smartcontract yang bernama FundMe
14 contract FundMe {
15     using PriceConverter for uint256;
16
17     //mapping uint256 kedalam address yang bersifat public dan diberi nama addressToAmountFunded
18     mapping(address => uint256) public addressToAmountFunded;
19     //hasil mapping address tadi dijadikan array yang bersifat public dan diberi nama funders
20     address[] public funders;
21
22     // Could we make this constant? /* hint: no! We should make it immutable! */
23     //membuat fungsi call yang diberi nama i_owner dan immutable
24     address public /* immutable */ i_owner;
25     //membuat variable USD yang minimum ketika dikirimkan
26     uint256 public constant MINIMUM_USD = 50 * 10 ** 18;
27
28     constructor() {
29         i_owner = msg.sender;
30     }
31     //fungsi mengirim uang dari ETH menjadi USD
32     function fund() public payable {
33         //fungsi untuk memeriksa apakah yang dikirimkan memenuhi minimal yang dapat dikirimkan
34         require(msg.value.getConversionRate() >= MINIMUM_USD, "You need to spend more ETH!");
35         // require(PriceConverter.getConversionRate(msg.value) >= MINIMUM_USD, "You need to spend more ETH!");
36         addressToAmountFunded[msg.sender] += msg.value;
37         funders.push(msg.sender);
38     }
39
40     //function untuk API call untuk mendapatkan harga USD/ETH yang terbaru
41 }
```

Membuat FundMe

Kedua, kita akan menyiapkan fungsi pengambilan dana dan bersifat public. Pada fungsi pengambilan dana pun kita berikan fungsi looping yang tujuannya menemukan address yang nanti akan berinteraksi lalu akan dimasukan ke array lalu akan di proses pada PriceConverter Adapun kita membuat sebuah fungsi untuk melakukan reset nilai array dan nilai dari address yang telah berinteraksi.

```
function getVersion() public view returns (uint256){
    AggregatorV3Interface priceFeed = AggregatorV3Interface(0x8A753747A1Fa494EC906cF90E9f37563BA8AF630e);
    return priceFeed.version();
}

// pengecekan apakah address punya owner atau bukan
modifier onlyOwner {
    // require(msg.sender == owner);
    if (msg.sender != 1_owner) revert NotOwner();
    _;
}

// fungsi mengambil uang yang bersifat pembayaran public
function withdraw() payable onlyOwner public {
    // pengalangan pencarian funder dari index 0 dan ketentuannya adalah jika funderIndex kurang dari
    // funders.length dan funderIndex akan inkrement
    for (uint256 funderIndex=0; funderIndex < funders.length; funderIndex++){
        // mengakses nilai funderIndex element dari looping dan disimpan pada funder
        address funder = funders[funderIndex];
        // meriset nilai balance dari mapping
        addressToAmountFunded[funder] = 0;
    }
    // reset array address
    funders = new address[](0);

    // // transfer ETH jika ada yang call fungsi withdraw
    // payable(msg.sender).transfer(address(this).balance);
    // // send
    // bool sendSuccess = payable(msg.sender).send(address(this).balance);
    // require(sendSuccess, "Send failed");
    // // call
    (bool callSuccess, ) = payable(msg.sender).call{value: address(this).balance}("");
    require(callSuccess, "Call failed");
}

// Explainer from: https://solidity-by-example.org/fallback/
// Ether is sent to contract
// is msg.data empty?
// / \
// yes no
// / \
// receive() fallback()
```

```
require(callSuccess, "Call failed");
}

// Explainer from: https://solidity-by-example.org/fallback/
// Ether is sent to contract
// is msg.data empty?
// / \
// yes no
// / \
// receive() fallback()

fallback() external payable {
    fund();
}

receive() external payable {
    fund();
}

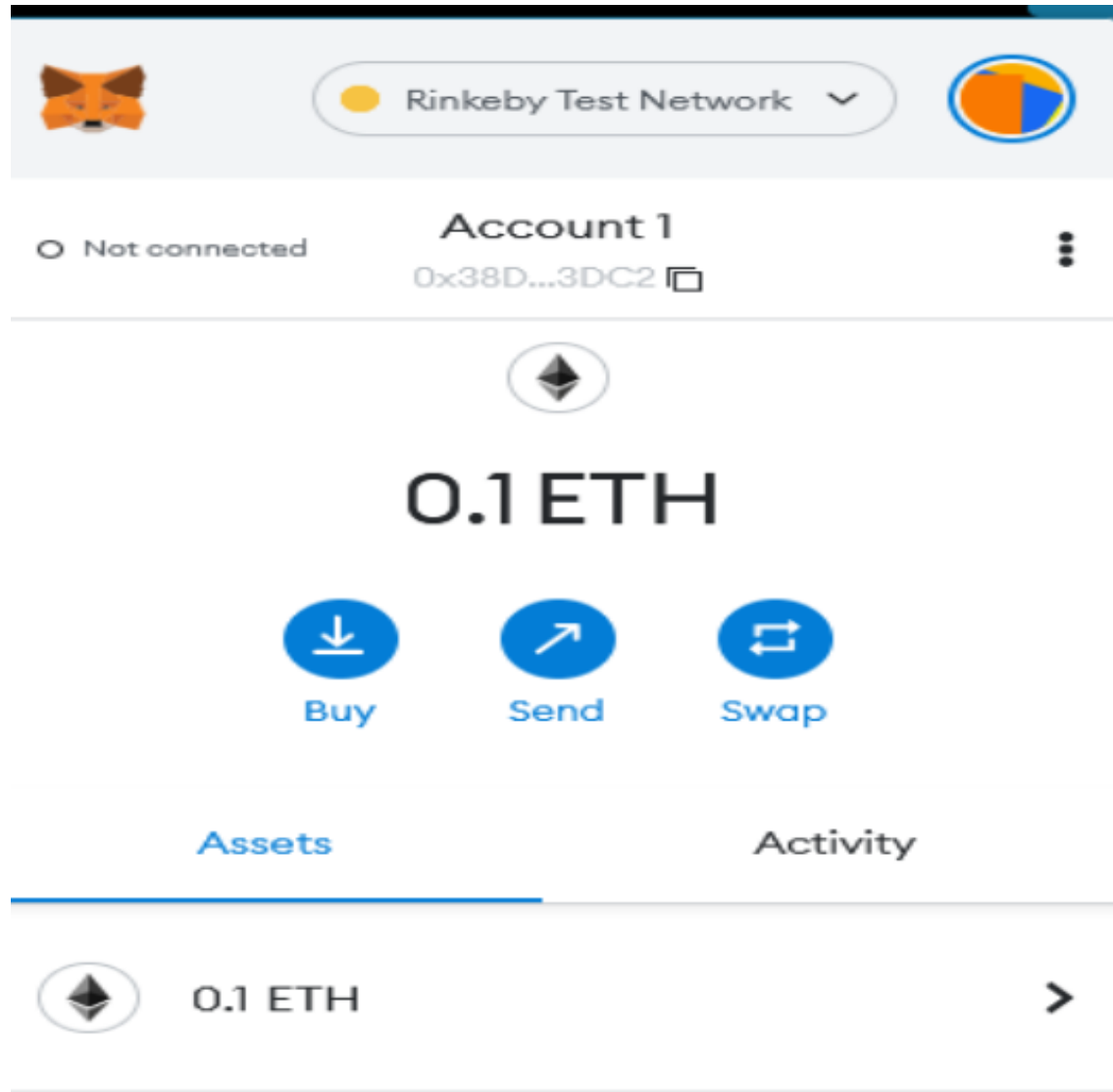
}

// Concepts we didn't cover yet (will cover in later sections)
// 1. Enum
// 2. Events
// 3. Try / Catch
// 4. Function Selector
// 5. abi.encode / decode
// 6. Hash with keccak256
// 7. Yul / Assembly
```

Membuat PriceConverter

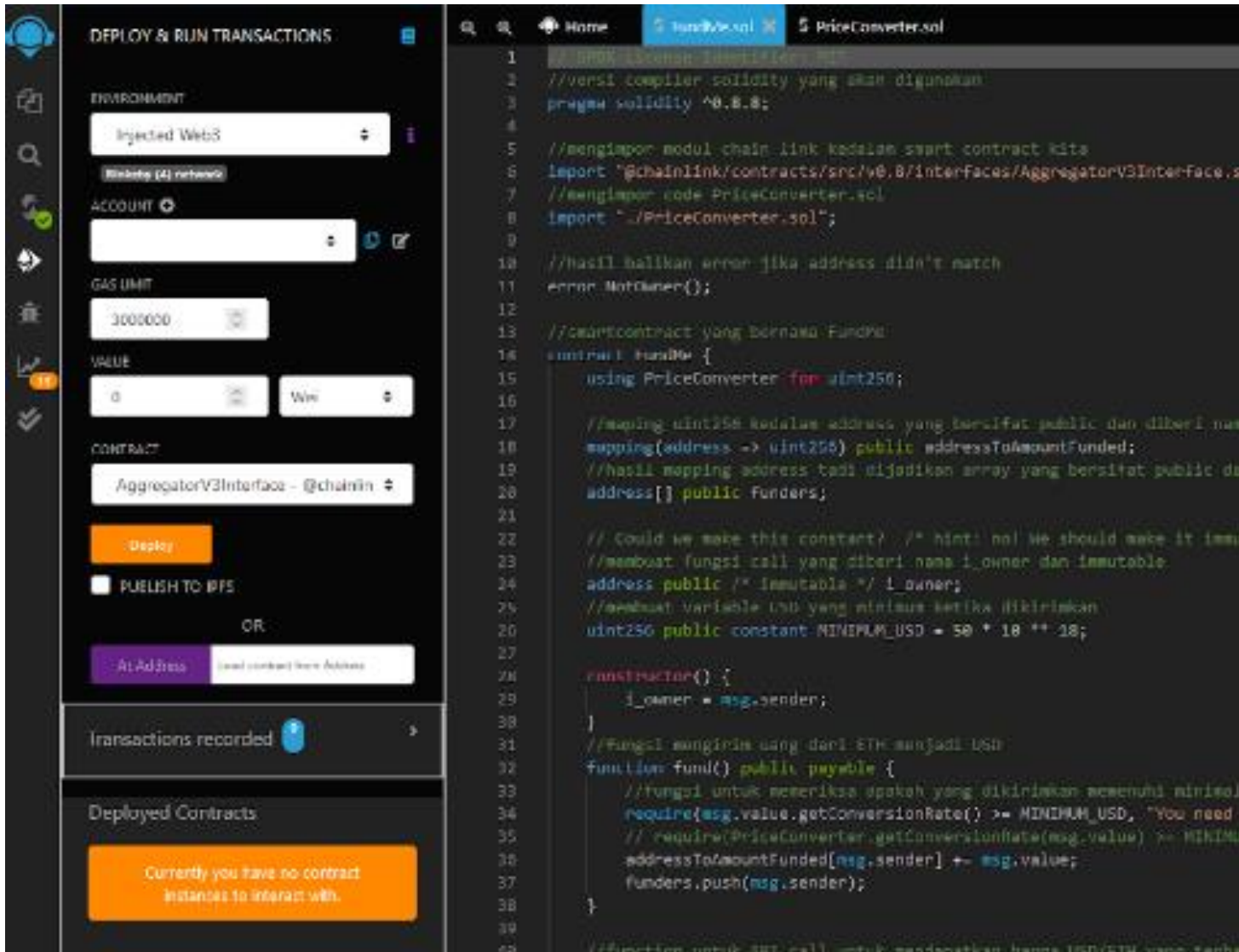
Langkah selanjutnya adalah mempersiapkan smart contract price converter yang gunanya adalah untuk mengeksekusi address yang didapatkan dari fundme lalu digunakan pada smart contract PriceConverter untuk mendapatkan nilai konversi dan mendapatkan nilai integer yang dapat dimengerti oleh manusia.

```
1 // SPDX-License-Identifier: MIT
2 // versi compiler solidity yang akan digunakan
3 pragma solidity ^0.8.8;
4
5 //mengimpor modul chain link kedalam smart contract kita
6 import "@chainlink/contracts/src/v0.8/interfaces/AggregatorV3Interface.sol";
7
8 // Why is this a library and not abstract?
9 // Why not an interface?
10 library PriceConverter {
11     // We could make this public, but then we'd have to deploy it
12     // mendapatkan nilai balikan harga ETH/USD tetapi tidak public
13     function getPrice() internal view returns (uint256) {
14         // Rinkeby ETH / USD Address
15         // https://docs.chain.link/docs/ethereum-addresses/
16         AggregatorV3Interface priceFeed = AggregatorV3Interface(
17             0x8A753747A1Fa494EC96cE90E9f37563A8AF630e
18         );
19         // memastikan outcome itu bernilai decimal yang dapat dibaca oleh manusia
20         (, int256 answer, , , ) = priceFeed.latestRoundData();
21         // ETH/USD rate in 18 digit
22         return uint256(answer * 10000000000);
23     }
24
25     // 10000000000
26     function getConversionRate(uint256 ethAmount)
27         internal
28         view
29         returns (uint256)
30     {
31         uint256 ethPrice = getPrice();
32         uint256 ethAmountInUsd = (ethPrice * ethAmount) / 1000000000000000000;
33         // the actual ETH/USD conversion rate, after adjusting the extra 0s.
34         return ethAmountInUsd;
35     }
36 }
37
```



Mempersiapkan Wallet

Kemudian kita mempersiapkan wallet agar dapat berinteraksi dengan test network. Disini digunakan wallet metamask dan menggunakan test network rinkby. Hal ini digunakan agar chain link pada smart contract kita dapat berinteraksi blockchain memerlukan network baik itu test maupun network ethereum yang sesungguhnya.

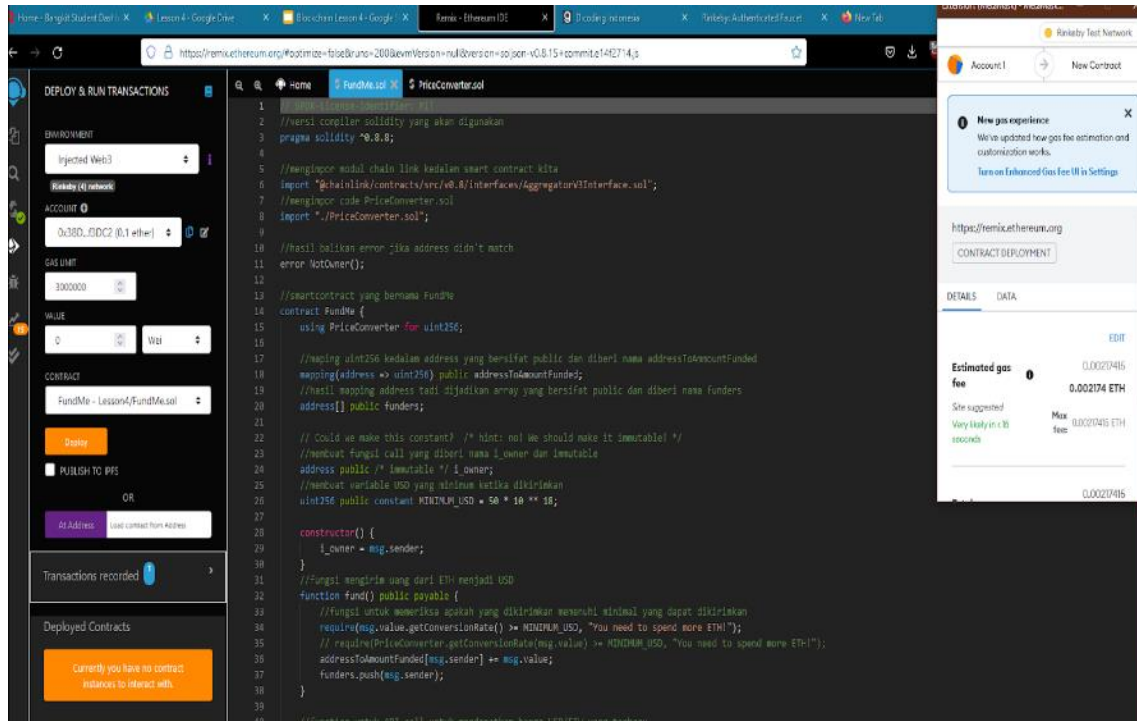


Mempersiapkan Environment Remix

Langkah selanjutnya adalah mengkoneksikan wallet yang sudah terintegrasi dengan rinkby test network dengan remix kita. Hal yang dapat dilakukan adalah mengganti environment pada remix menjadi injected web 3 dan koneksikan dengan mengkoneksikan maka account dengan ETH nya dapat dilihat pada tab account pada remix.

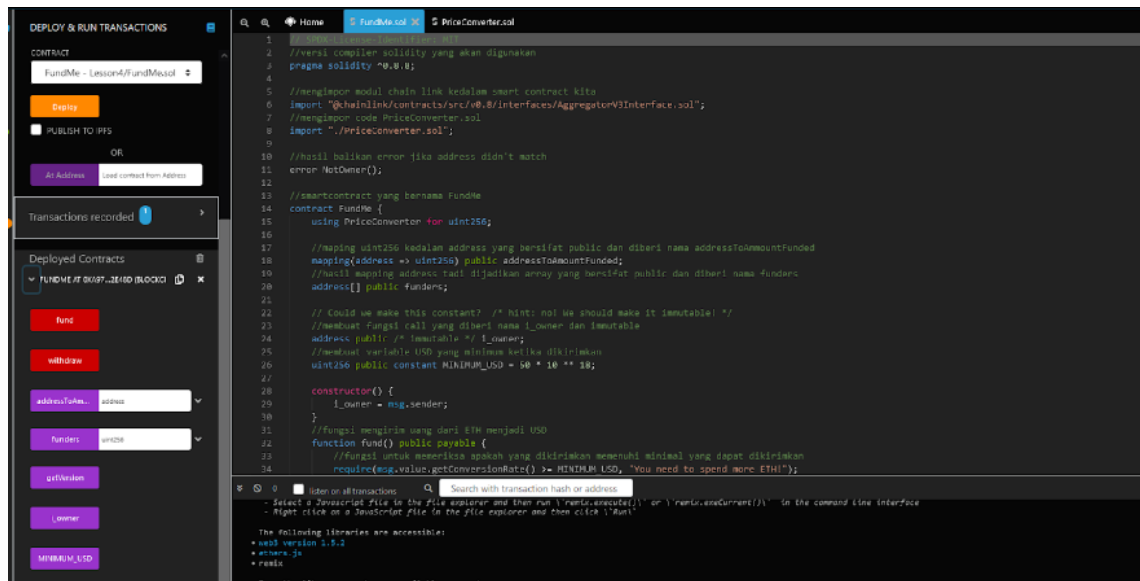
Deploy fundME kedalam Test Network

Kemudian selanjutnya adalah mendeploy smartcontract kita kedalam test network. Pada metamask ada price yang kita harus bayar setiap kita berinteraksi dengan blockchain yang disebut dengan gas fee.



Deploy FundMe kedalam Test Network

Jika smartcontract kita berhasil untuk di deploy maka transaksi kita dapat dilihat pada deployed contract dan pada etherscan.



Explore FundMe

Setelah kita mendeploy kita dapat berinteraksi dengan fungsi dari smart contract kita. Dapat dilihat bahwa pada minimum USD kita mendapatkan nilai 50USD dan owner itu dari account yang mendeploy smartcontractnya.

The screenshot displays the FundMe smart contract deployment and interaction interface. On the left, the 'DEPLOY & RUN TRANSACTIONS' panel shows the contract name 'FundMe - Lesson/FundMe.sol' and a 'Deploy' button. Below this, there are fields for 'As Address' and 'Load contract from Address'. The 'Transactions recorded' section shows a list of transactions, including 'FundMe AT 0x417...204D BLOCKED'. The 'Deployed Contracts' section shows a list of contracts, including 'FundMe AT 0x417...204D BLOCKED'. The main panel displays the Solidity code for the FundMe contract, which includes a constructor, a fund function, and a withdraw function. The code is written in Solidity and includes comments in Indonesian. The bottom panel shows the transaction details for the deployment, including the transaction hash, block number, and gas used.

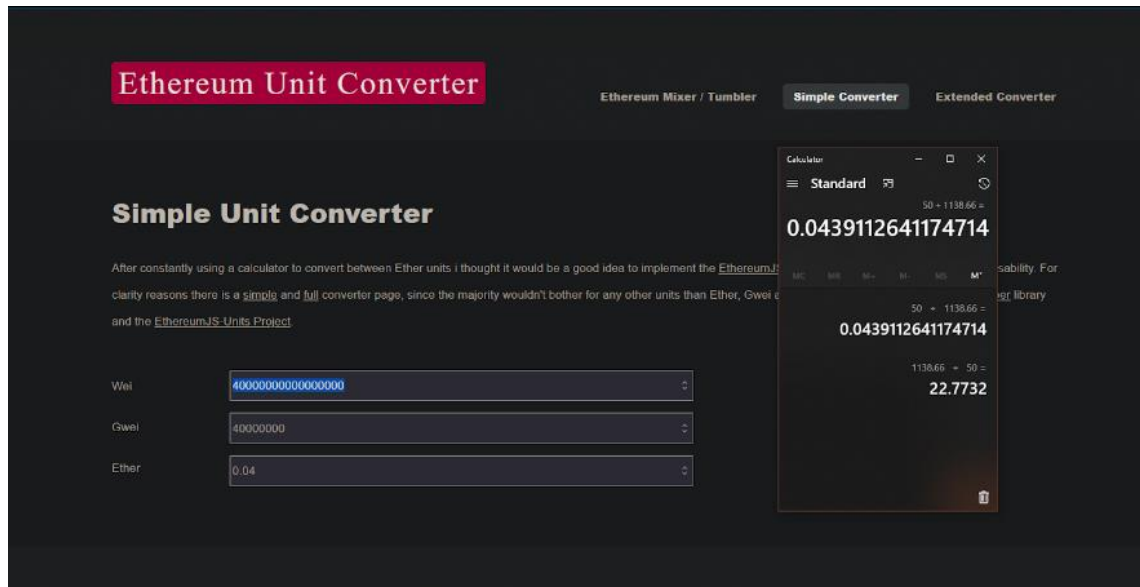
```
1 // SPDX-License-Identifier: MIT
2 // versi compiler solidity yang akan digunakan
3 pragma solidity ^0.8.8;
4
5 //mengimport modul chain link kedalam smart contract kita
6 import "@chainlink/contracts/src/v0.8/interfaces/AggregatorV3Interface.sol";
7 //mengimport code PriceConverter.sol
8 import "../PriceConverter.sol";
9
10 //hasil bilikan error jika address didn't match
11 error NotOwner();
12
13 //smartcontract yang bernama FundMe
14 contract FundMe {
15     using PriceConverter for uint256;
16
17     //mapping uint256 kedalam address yang bersifat public dan diberi nama addressToAccountFunded
18     mapping(address => uint256) public addressToAccountFunded;
19     //hasil mapping address tadi dijadikan array yang bersifat public dan diberi nama funders
20     address[] public funders;
21
22     // Could we make this constant? /* hint: no! We should make it immutable */
23     //membuat fungsi call yang diberi nama i_owner dan immutable
24     address public /* immutable */ i_owner;
25     //membuat variable USD yang minimum ketika dikirimkan
26     uint256 public constant MINIMUM_USD = 50 * 10 ** 18;
27
28     constructor() {
29         i_owner = msg.sender;
30     }
31     //fungsi mengirim uang dari ETH menjadi USD
32     function fund() public payable {
33         //fungsi untuk memeriksa apakah yang dikirimkan memenuhi minimal yang dapat dikirimkan
34         require(msg.value.getConversionRate() >= MINIMUM_USD, "You need to spend more ETH!");
35     }
36
37     //fungsi untuk menarik uang dari USD menjadi ETH
38     function withdraw() public {
39         //fungsi untuk memeriksa apakah yang dikirimkan memenuhi minimal yang dapat dikirimkan
40         require(msg.value.getConversionRate() >= MINIMUM_USD, "You need to spend more ETH!");
41     }
42 }
```

Account 1
0.0978 ETH
Buy Send Swap
Assets Activity
Contract Deployment -0 ETH
Receive 0.1 ETH

view on etherscan
[block:10076448 txIndex:5] from: 0x380...F30C2 to: FundMe (constructor) value: 0 wei data: 0x00...F0B33 logs: 0 hash: 0x0e...F0B33
call to FundMe.FundMe_USD

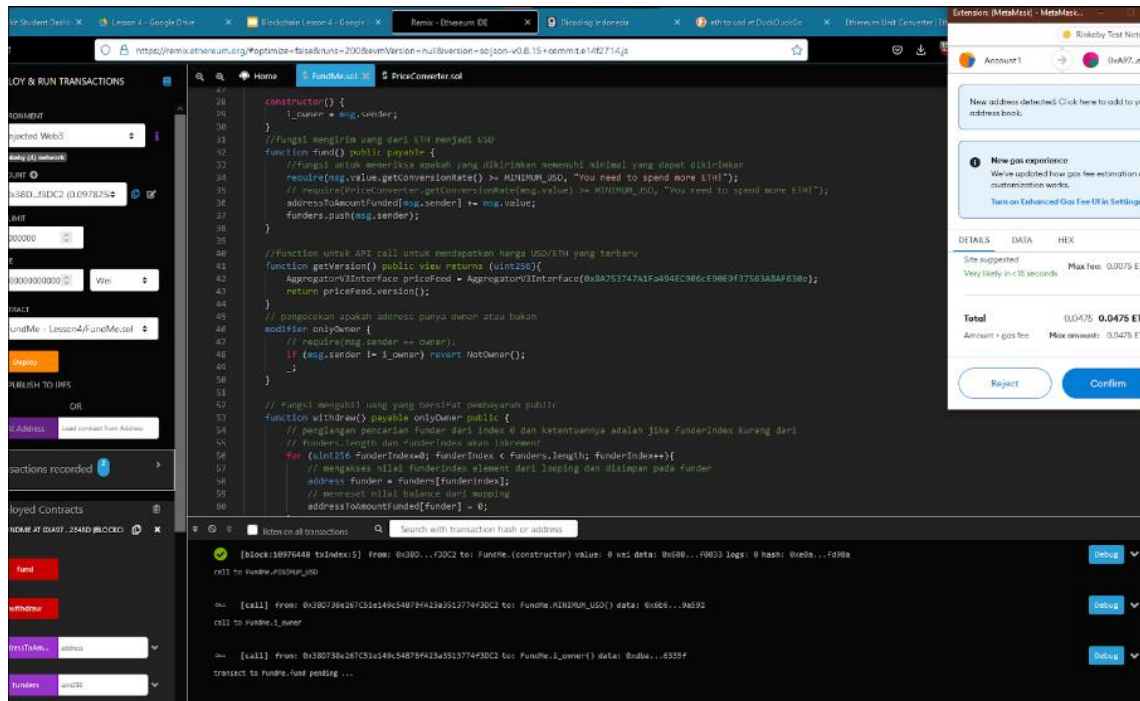
Explore FundMe

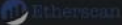
Pada smartcontract kita ada fungsi fund atau mengirim uang, untuk menggunakan fungsi itu dengan benar kita harus menghitung nilai terendah dari ETH yang dapat dikirimkan. Pada saat penulis membuat slide ini harga ETH terhadap USD adalah 1138.66USD maka kita akan membagi 50USD tadi dengan 1138.66USD maka kita akan mendapatkan nilai ETH minimal yang dapat dikirimkan. Lalu kita ubah nilai 0.04 ETH menjadi WEI agar dapat di masukan kedalam remix.



Explore FundMe

Setelah kita memasukan ETH yang akan dikirim menggunakan satuan WEI maka kita akan coba untuk mengirim sejumlah WEI ke account yang berbeda pada test network.





Rinkeby Testnet Network

All Filters

Search by Address / Txn Hash / Block / Token / Erc

Home

Blockchain

Tokens

Misc

Rinkeby

Transaction Details

Overview

Internal Txns

State

[This is a Rinkeby Testnet transaction only]

Transaction Hash:

0a9c8fe0a0e150b1a655804886692efb78cac78aac9e6538611946019e3708d

Status:

Success

Block:

1097607615 Block Confirmations

Timestamp:

4 mins ago (Jul-06-2022 11:11:59 AM +UTC)

From:

0a38d738e267c51e149c54b7bfa23a3513774f3dc2

To:

Contract 0aa9756c992412d5fca8cafdc4778eca421d2e43d

Value:

0.0439112641174714 Ether (\$0.00)

Transaction Fee:

0.000270727507265497 Ether (\$0.00)

Gas Price:

0.000000002500000067 Ether (2.500000067 Gwei)

Gas Limit & Usage by Txn:

108,291 | 108,291 (100%)

Gas Fees:

Base: 0.000000067 Gwei | Max: 2.500000144 Gwei | Max Priority: 2.5 Gwei

Burnt & Txn Savings Fees:

Burnt 0.00000000007255497 Ether (\$0.00) | Txn Savings: 0.00000000008338407 Ether (\$0.00)

Others:

Txn Type: 2 (EIP-1559) | Nonce: 2 | Position: 3

Input Data:

This website uses cookies to improve your experience and has an updated Privacy Policy.

Get it

Explore FundMe

Jika kita melihat history transaction pada etherscan, kita dapat melihat bahwa kita berhasil dalam mengirimkan sejumlah ETH ke dalam account yang berbeda. Jika ini berhasil dilakukan maka fungsi “fund” pada smartcontract kita berhasil digunakan.

Explore FundMe

Jika klik “withdraw” maka kita akan mengekspektasikan bahwa nilai yang tadi kita kirimkan akan Kembali lagi dan terjadi pengurangan dengan gas fee dalam setiap transaksi. jika seperti gambar di samping benar, maka nilai ETH kita tidak 0.1 ETH tetapi 0.09 ETH yang menunjukkan setiap transaksi itu payable dan kita membayar gas.

