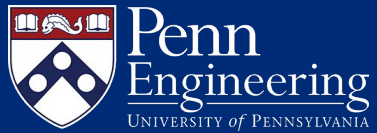
Solidity 101

Web3 Builders workshop series #3



Disclaimer

- This workshop series is **not** designed to teach you everything about blockchain, but it serves as a starting point for you to do your own research
- We will not be going into too much details, but feel free to discuss more about it with us after the main workshop!
- Feel free to interrupt us anytime you want
- **This one is technical lol**
-
- Enjoy :)

Penn Engineering
University of Pennsylvania
Wharton
University of Pennsylvania

Ethereum 101 Review

- EVM
- Smart Contracts
- Applications:
  - DeFi, NFTs, DAOs, Tokens
- How do we create these applications?

Interact with Ethereum/EVM

- Two ways to interact with Ethereum/EVM
  - Simple actions (send money, read data, use an existing contract)-> **do these in a wallet**
  - **Put a smart contract onto the Ethereum** that can be used later (by yourself or other people)
- Solidity is the most popular programming language for EVM programming

Smart contract

A smart contract can:

- Be an **independent entity on blockchain** just like a person (has an address–like an ID)
- It can **own assets** (e.g. a smart contract can own ETH, USDC, NFTs)
- It can **interact with other smart contracts/accounts**

A smart contract can not:

- Change its code -> good and bad
- Do things automatically (someone needs to call it)

Lifecycle of smart contract creation

1. Come up with an idea
2. Turn this idea into code (e.g. Solidity)
3. Turn code (human-readable) into something machine understands(binary)
    a. If you code has bugs, we may find them by trying to run them – “testing”
4. Upload the binary to blockchain
5. You & and everyone else can use the smart contract

Sample Solidity Code

Users > yihengchen > Downloads > example.sol

```
1  pragma solidity ^0.7.0; //specifies the version of solidity to be used
2                          //thus the compiler knows how to compile the code
3
4  contract HelloWorld { //defines a contract named HelloWorld
5
6      string public message;  //defines a string variable named message
7
8      constructor(string memory _message) {   //the constructor function is called when the contract is deployed
9          message = _message;
10     }
11
12     function setMessage(string memory _message) public { //defines a function named setMessage
13         message = _message;
14     }
15
16     function getMessage() public view returns (string memory) { //defines a function named getMessage
17         return message;
18     }
19
20 }
```

Penn Engineering University of Pennsylvania

Wharton University of Pennsylvania

You need tools…

Option 1:

- **node.js(npm)+ hardhat**
- (recommended)
- (however it can take long and requires basic CS knowledge)

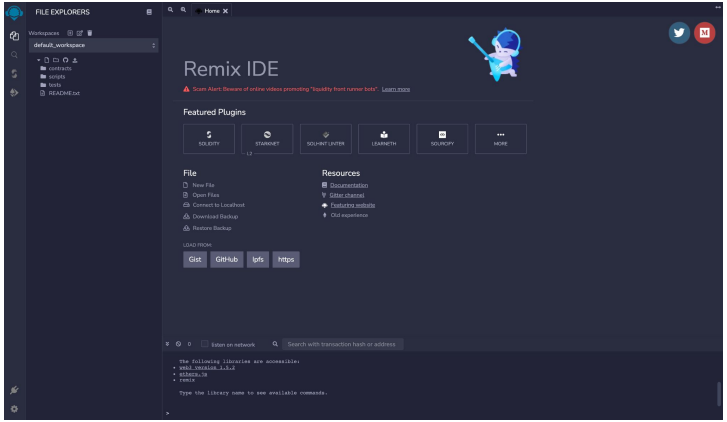
Option 2:

- **Remix Solidity IDE**
- “Integrated”, runs in your browser

Penn Engineering
University of Pennsylvania
Wharton
University of Pennsylvania

Hand-on activities - remix

1. Go to **remix.ethereum.org**
2. Click the “+” next to the “workspaces”
3. Choose a template -> ERC 721
4. “OK”



Penn Engineering University of Pennsylvania
Wharton University of Pennsylvania

From Code to Smart Contract

```
pragma solidity ^0.7.0; //specifies the version of solidity to be used
                        //thus the compiler knows how to compile the code

contract HelloWorld { //defines a contract named HelloWorld

    string public message;  //defines a string variable named message

    constructor(string memory _message) {   //the constructor function is called when the contract is deployed
        message = _message;
    }

    function setMessage(string memory _message) public { //defines a function named setMessage
        message = _message;
    }

    function getMessage() public view returns (string memory) { //defines a function named getMessage
        return message;
    }
}
```

Code (Solidity)

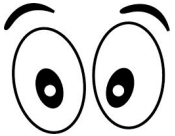
Compiler (e.g. solc)

Compile


```
"data": {
    "bytecode": {
        "functionDebugData": {
            "@_44": {
                "entryPoint": null,
                "id": 44,
                "parameterSlots": 2,
                "returnSlots": 0
            },
            "@_724": {
                "entryPoint": null,
                "id": 724,
                "parameterSlots": 0,
                "returnSlots": 0
            },
            "array_dataslot_t_string_storage": {
                "entryPoint": 328,
                "id": null,
                "parameterSlots": 1,
                "returnSlots": 1
            },
```


Byte code

ABI(Application Binary Interface), metadata



Etherscan

Upload to Ethereum

Wait…

Deploy

Smart Contract

Penn Engineering
University of Pennsylvania

Wharton
University of Pennsylvania

Who do you think you are?

- (Plz leave ur Remix page open)
- You are nobody to the blockchain
- You can’t do anything…. yet

- Let’s go and get you an ID (wallet)

Penn Engineering
University of Pennsylvania

Wharton
University of Pennsylvania

Ethereum accounts and wallets

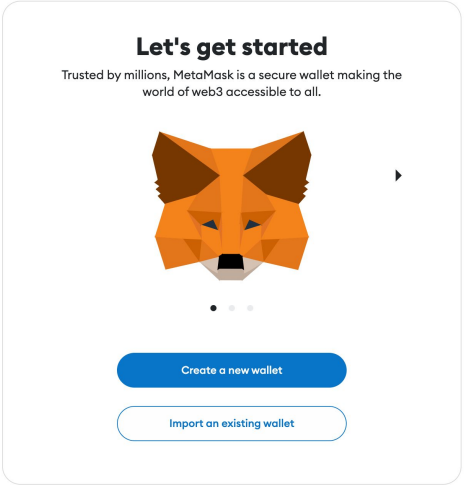
- Ethereum Account
  - An entity that can send transactions, has a balance, and has an address
  - Has a
    - public key (think about this as your username - in fact the account address is based off this)
    - private key (think about it as a password, do not ever reveal this)
- Ethereum Wallet
  - A wallet is an account manager

~~Create your ethereum wallet here:~~

Medium article - how to create your metamask account

Hand-on activities - Metamask

1. Fingers crossed you have a **Chrome browser**
2. Go to **metamask.io**
3. Download download, install install….
4. Create a new wallet with me



Penn Engineering
University of Pennsylvania

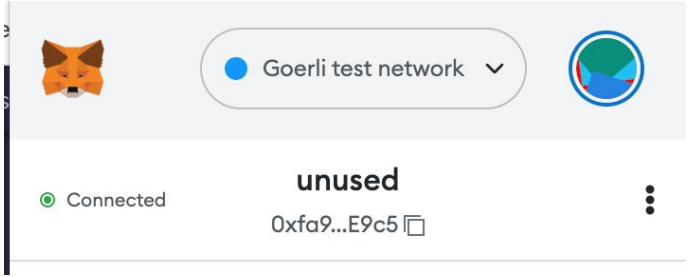
Wharton
University of Pennsylvania

Wallet tips

1. It helps you use web3 websites and interact with smart contracts
2. The secret key +pw is the only way you can access it
3. You lose it, you can’t “forgot password”
4. Only approve/sign transactions if you know what’s going on

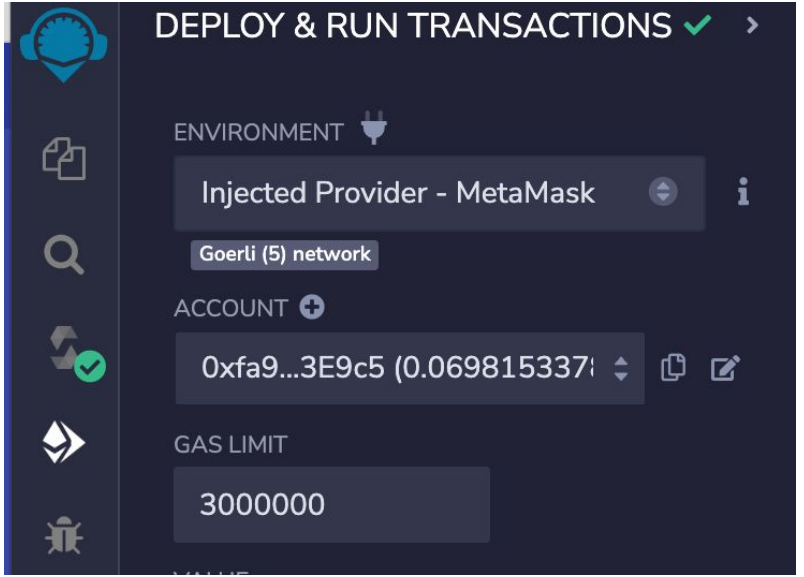
Before we deploy your own “cryptocurrency”

1. Go to **goerlifaucet.com**
2. Sign up for a free alchemy account
3. Come back to the page
4. Open metamask and copy your public key
5. (you see that copy icon?)
6. Put it in the box and “Send me ETH”
7. You should see your ETH balance update in a bit



Finally… Let's go deploy - Part 1

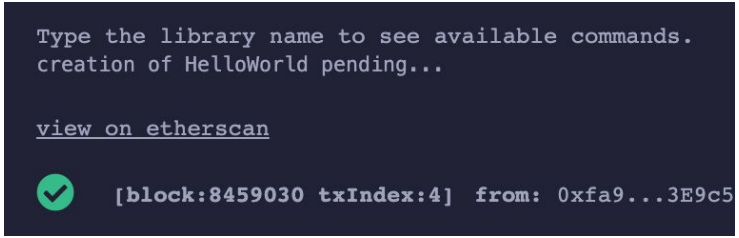
1. Go back to Remix
2. On the left bar, 4th icon down (ethereum) "Deploy and run transactions"
3. Environment -> select **Injected Provider -Metamask**
4. Account-> It should auto select the one we just created
   a. If u have multiple, don't use one u actually hold real money



Penn Engineering University of Pennsylvania | Wharton University of Pennsylvania

Finally… Let's go deploy - Part 2

1. Click **"Deploy"**
   a. Don't see it? Go back to the third tab down and click **"Compile"**
2. Metamask prompt-> confirm
3. Check the command line output at the bottom
4. Wait for green check-> Click **"view on etherscan"**



Remember this animation?

https://www.figma.com/proto/r34qLPnbRKEhyDXgmWzb52/CIS-2330-Animations?page-id=0%3A1&node-id=9%3A425&viewport=951%2C287%2C0.05&scaling=contain&starting-point-node-id=9%3A425&show-proto-sidebar=1

Let’s play with our token/cryptocurrency

1. Let me try to give myself money
2. Need to fix my contract a bit
3. In **deployed contracts,** enter (your address, number) next to **mint**
4. Then check **allowance** (your address)

```
import "@openzeppelin/contracts/token/ERC20/ERC20.sol";

contract MyToken is ERC20 {
    constructor() ERC20("MyToken", "INT") {}

    function mint(address to, uint256 amount) public {
    _mint(to, amount);
}
}
```

Want more Solidity?

Too bad we are not covering…

But let us know if you are interested, we are happy to make a course for it

Additional Resources

- Full guide to Ethereum Development by Nader Dabit
- Buildspace (platform for awesome project tutorials)
- Youtube videos!

A Simple Example

```
pragma solidity 0.8.0;

contract VendingMachine {
    uint funds;
    address owner;

    constructor() {
       owner = msg.sender;
    }

    function vend() public {
        require (msg.value == price);
        address buyer = msg.sender;
        funds += msg.value;

        snack.transfer(buyer, 1);
    }

    function withdraw() onlyOwner public {
        owner.transfer(funds);
    }
}
```

A Simple Example

Solidity compiler version

Compiles with Solidity versions `0.8.0`.

```
pragma solidity 0.8.0;

contract VendingMachine {
    uint funds;
    address owner;

    constructor() {
       owner = msg.sender;
    }

    function vend() public {
        require (msg.value == price);
        address buyer = msg.sender;
        funds += msg.value;

        snack.transfer(buyer, 1);
    }

    function withdraw() onlyOwner public {
        owner.transfer(funds);
    }
}
```

A Simple Example

Contract: collection of
- code (i.e., functions)
- data (i.e., state)

Solidity contracts ≃ classes

Inheritance
State variables
Functions
Events
Enums
…

```
pragma solidity 0.8.0;

contract VendingMachine {
    uint funds;
    address owner;

    constructor() {
       owner = msg.sender;
    }

    function vend() public {
        require (msg.value == price);
        address buyer = msg.sender;
        funds += msg.value;

        snack.transfer(buyer, 1);
    }

    function withdraw() onlyOwner public {
        owner.transfer(funds);
    }
}
```

A Simple Example

Declare a state variable called `funds` of type unsigned integer `uint`

```
pragma solidity 0.8.0;

contract VendingMachine {
    uint funds;
    address owner;

    constructor() {
        owner = msg.sender;
    }

    function vend() public {
        require (msg.value == price);
        address buyer = msg.sender;
        funds += msg.value;

        snack.transfer(buyer, 1);
    }

    function withdraw() onlyOwner public {
        owner.transfer(funds);
    }
}
```

A Simple Example

Declare a state variable called `owner` of type unsigned integer `address`

```
pragma solidity 0.8.0;

contract VendingMachine {
    uint funds;
    address owner;

    constructor() {
       owner = msg.sender;
    }

    function vend() public {
        require (msg.value == price);
        address buyer = msg.sender;
        funds += msg.value;

        snack.transfer(buyer, 1);
    }

    function withdraw() onlyOwner public {
        owner.transfer(funds);
    }
}
```

A Simple Example

The constructor is called exactly once upon deployment to the blockchain

```
pragma solidity 0.8.0;

contract VendingMachine {
    uint funds;
    address owner;

    constructor() {
        owner = msg.sender;
    }

    function vend() public {
        require (msg.value == price);
        address buyer = msg.sender;
        funds += msg.value;

        snack.transfer(buyer, 1);
    }

    function withdraw() onlyOwner public {
        owner.transfer(funds);
    }
}
```

A Simple Example

The msg object contains various information about the tx

```
{
    "from": PUBLIC_KEY,
    "to": contractAddress,
    "nonce": nonce,
    "gas": 500000,
    "data": calldata
}
```

```
pragma solidity 0.8.0;

contract VendingMachine {
    uint funds;
    address owner;

    constructor() {
        owner = msg.sender;
    }

    function vend() public {
        require (msg.value == price);
        address buyer = msg.sender;
        funds += msg.value;

        snack.transfer(buyer, 1);
    }

    function withdraw() onlyOwner public {
        owner.transfer(funds);
    }
}
```

A Simple Example

Define functions to (1) purchase a snack and (2) take the money out

```
pragma solidity 0.8.0;

contract VendingMachine {
    uint funds;
    address owner;

    constructor() {
       owner = msg.sender;
    }

    function vend() public {
        require (msg.value == price);
        address buyer = msg.sender;
        funds += msg.value;

        snack.transfer(buyer, 1);
    }

    function withdraw() onlyOwner public {
        owner.transfer(funds);
    }
}
```

A Simple Example

require statement is similar to an assertion. If evaluates to false, the tx reverts.

msg.value is the amount of ETH sent with the tx

```
pragma solidity 0.8.0;

contract VendingMachine {
    uint funds;
    address owner;

    constructor() {
        owner = msg.sender;
    }

    function vend() public {
        require (msg.value == price);
        address buyer = msg.sender;
        funds += msg.value;

        snack.transfer(buyer, 1);
    }

    function withdraw() onlyOwner public {
        owner.transfer(funds);
    }
}
```

A Simple Example

transfer function

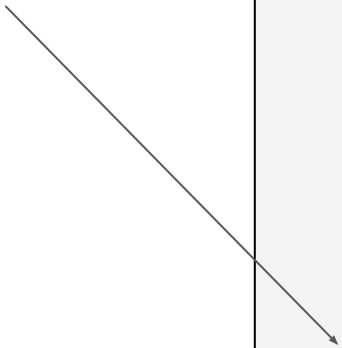
```
asset.transfer(receiver, amount)
```

```
pragma solidity 0.8.0;

contract VendingMachine {
    uint funds;
    address owner;

    constructor() {
       owner = msg.sender;
    }

    function vend() public {
        require (msg.value == price);
        address buyer = msg.sender;
        funds += msg.value;

        snack.transfer(buyer, 1);
    }

    function withdraw() onlyOwner public {
        owner.transfer(funds);
    }
}
```

A Simple Example

```
modifier onlyOwner() {
    require(msg.sender == owner);
    _;
}
```

Modifier is a **precondition** executed prior to the function



```
pragma solidity 0.8.0;

contract VendingMachine {
    uint funds;
    address owner;

    constructor() {
       owner = msg.sender;
    }

    function vend() public {
        require (msg.value == price);
        address buyer = msg.sender;
        funds += msg.value;

        snack.transfer(buyer, 1);
    }

    function withdraw() onlyOwner public {
        owner.transfer(funds);
    }
}
```

A Simple Example

transfer function

```
receiver.transfer(amount);
```

```
pragma solidity 0.8.0;

contract VendingMachine {
    uint funds;
    address owner;

    constructor() {
       owner = msg.sender;
    }

    function vend() public {
        require (msg.value == price);
        address buyer = msg.sender;
        funds += msg.value;

        snack.transfer(buyer, 1);
    }

    function withdraw() onlyOwner public {
        owner.transfer(funds);
    }
}
```