CMSC 398F
Week #10
Solidity Continuation

•••

Announcements

- Quiz 4 will be released today
  - Will be on Solidity
    - Specifically information on cryptozombies.io
- Join the class Piazza!
  - piazza.com/umd/fall2022/cmsc398f

From Last Time

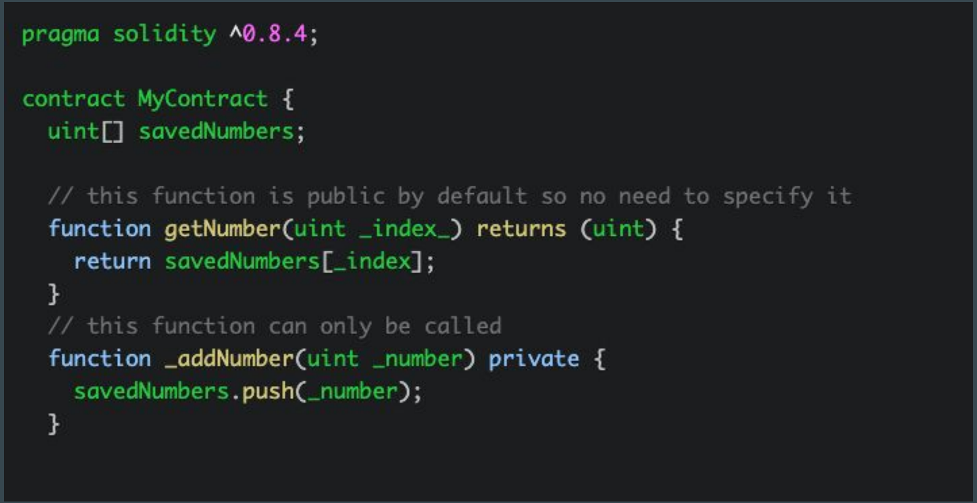
- The EVM is a distributed state machine which can change from block to block according to a predefined set of rules, and which can execute arbitrary machine code
- A "smart contract" is simply a program that runs on the Ethereum blockchain
    - EVM offers a runtime environment for smart contracts to execute
- We will use Solidity to develop Smart Contracts on the Ethereum blockchain
- Solidity has different types of variables
    - State (written directly into the blockchain “storage”)
    - Local (defined within the function. Written into the contract’s “memory”)
    - Global (variables that all contracts have access to)

Visibility Modifiers

- Visibility modifiers define the visibility of state variables or functions
- There are 4 modifiers that Solidity has:
  - Private
  - Internal
  - Public
  - External

Private Modifier

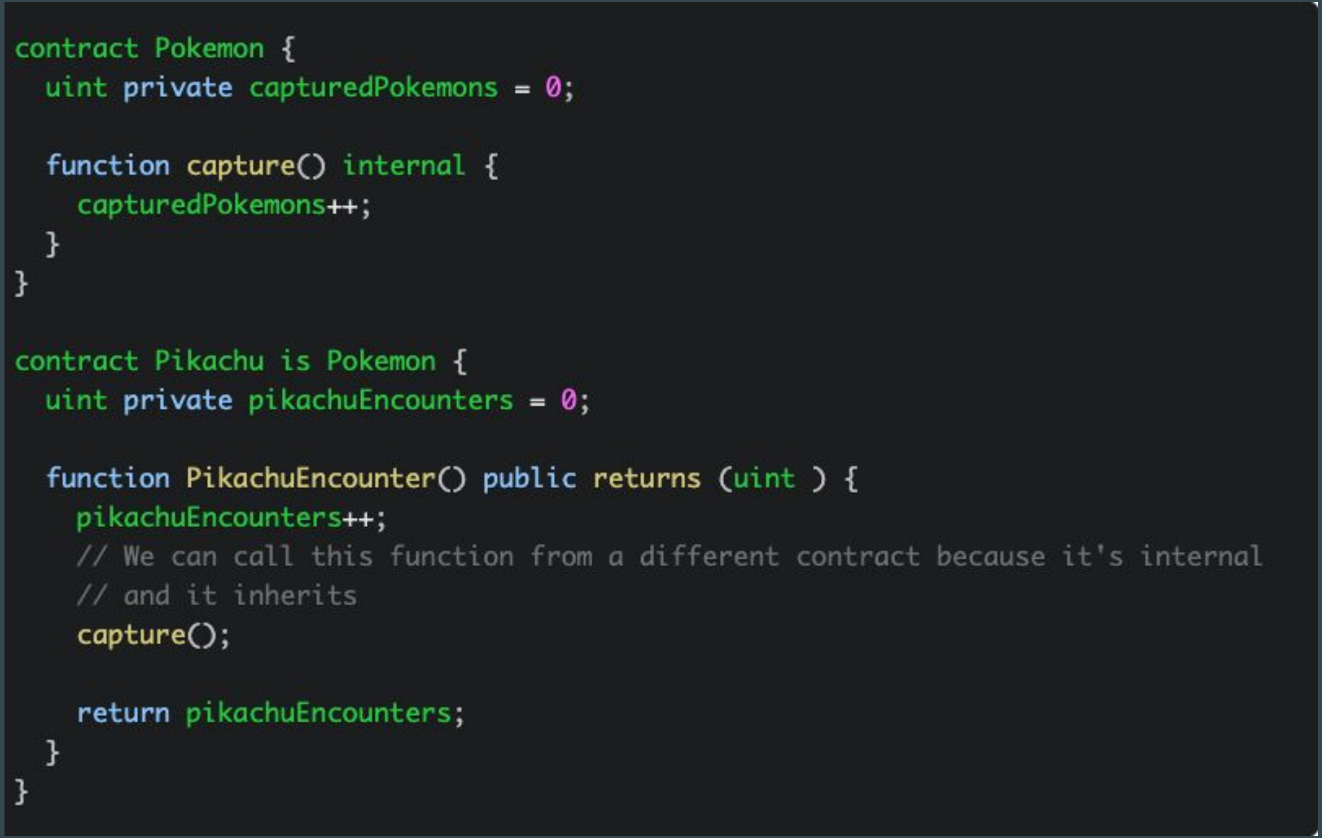
- In Solidity, functions are public by default
- In some cases, we only want only that contract to access some functions
  - This would requires a private modifier



```
pragma solidity ^0.8.4;

contract MyContract {
  uint[] savedNumbers;

  // this function is public by default so no need to specify it
  function getNumber(uint _index_) returns (uint) {
    return savedNumbers[_index];
  }
  // this function can only be called
  function _addNumber(uint _number) private {
    savedNumbers.push(_number);
  }
```

Internal Modifier

- Similar to the private modifier, this keyword only allows visibility to other functions within the same contract
- However it also allows for visibility with any inherited contracts as well
  - Contracts are similar (conceptually) to Java Classes
    - Inheritance
    - Polymorphism

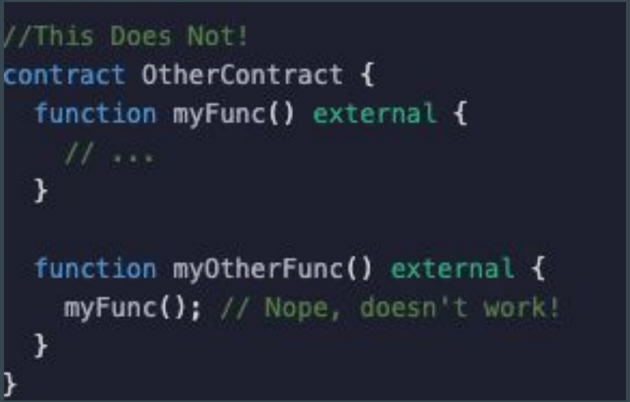

```
contract Pokemon {
  uint private capturedPokemons = 0;

  function capture() internal {
    capturedPokemons++;
  }
}

contract Pikachu is Pokemon {
  uint private pikachuEncounters = 0;

  function PikachuEncounter() public returns (uint ) {
    pikachuEncounters++;
    // We can call this function from a different contract because it's internal
    // and it inherits
    capture();

    return pikachuEncounters;
  }
}
```

Public Modifier

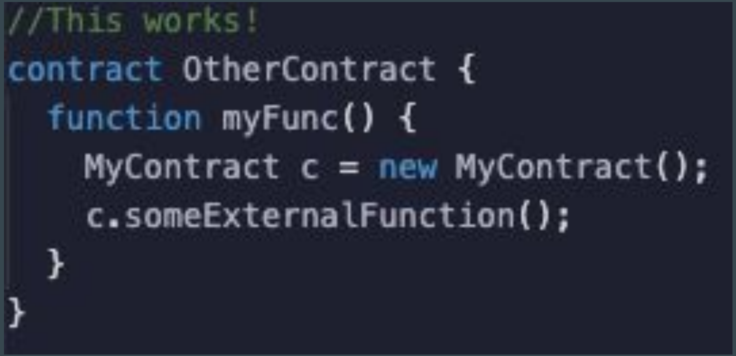
- Both state variables (the 'properties' of your contract) and functions can be marked as public
- Public state variables and functions can both be accessed from the outside and the inside of that contract
- The Solidity compiler automatically generates a getter function for **variables** with the public keyword

External Modifier

- External functions are part of the contract interface and can be called from other contracts and transactions
    - They can't be called within the same contract
- Only functions can be marked as external



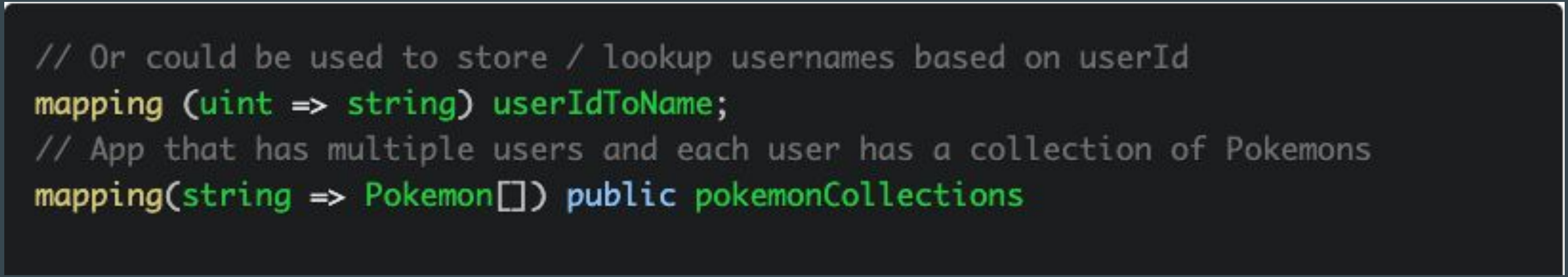
```
//This Does Not!
contract OtherContract {
  function myFunc() external {
    // ...
  }

  function myOtherFunc() external {
    myFunc(); // Nope, doesn't work!
  }
}
```



```
//This works!
contract OtherContract {
  function myFunc() {
    MyContract c = new MyContract();
    c.someExternalFunction();
  }
}
```

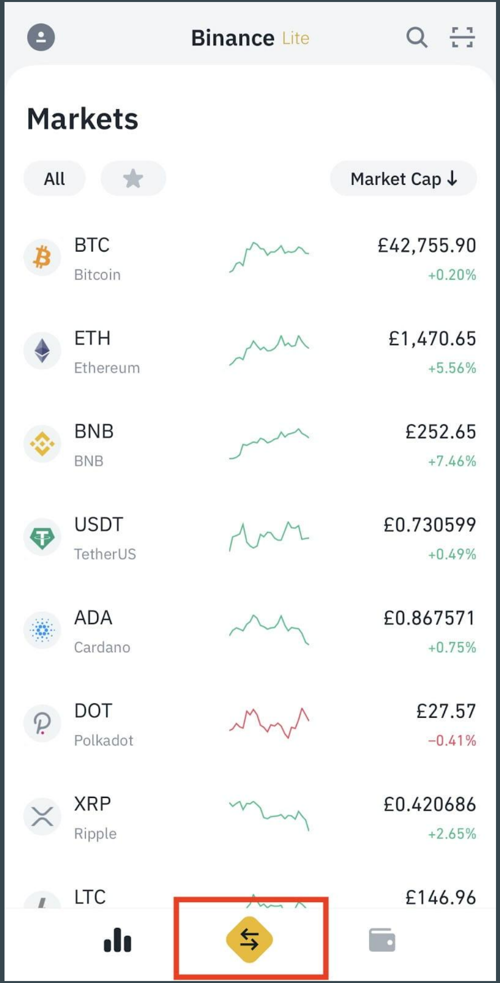
Complex Data Types

- Arrays
  - Static and Dynamic
  - Static means we predefine the size
  - Dynamic means the size can change
- Structs
  - Data types with multiple variables, known as properties
  - Similar to that of C
- Mappings
  - Basically a map, hashmap, dictionary, etc.
  - Keys need to be a simple data type (uint, string, bool, etc.)
  - Values can be complex data types (structs, arrays, other mappings)



```
// Or could be used to store / lookup usernames based on userId
mapping (uint => string) userIdToName;
// App that has multiple users and each user has a collection of Pokemons
mapping(string => Pokemon[]) public pokemonCollections
```

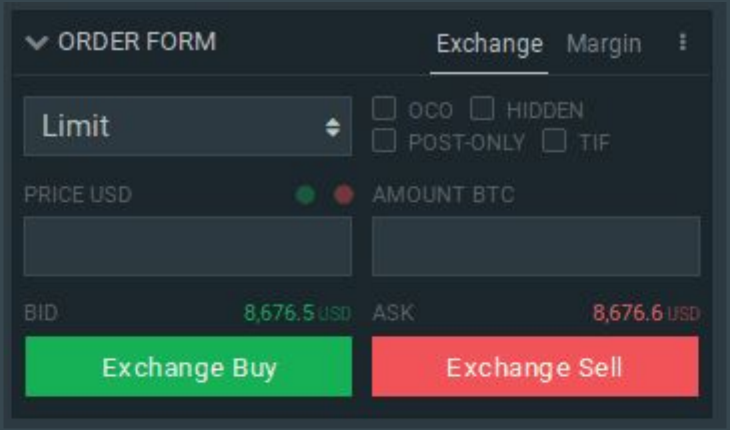
ICOs and Cryptocurrency Markets


Binance Lite

Markets

All ★ Market Cap ↓

BTC	Bitcoin	£42,755.90 +0.20%
ETH	Ethereum	£1,470.65 +5.56%
BNB	BNB	£252.65 +7.46%
USDT	TetherUS	£0.730599 +0.49%
ADA	Cardano	£0.867571 +0.75%
DOT	Polkadot	£27.57 −0.41%
XRP	Ripple	£0.420686 +2.65%
LTC		£146.96


ORDER FORM Exchange Margin

Limit

OCO HIDDEN POST-ONLY TIF

PRICE USD AMOUNT BTC

BID 8,676.5 USD ASK 8,676.6 USD

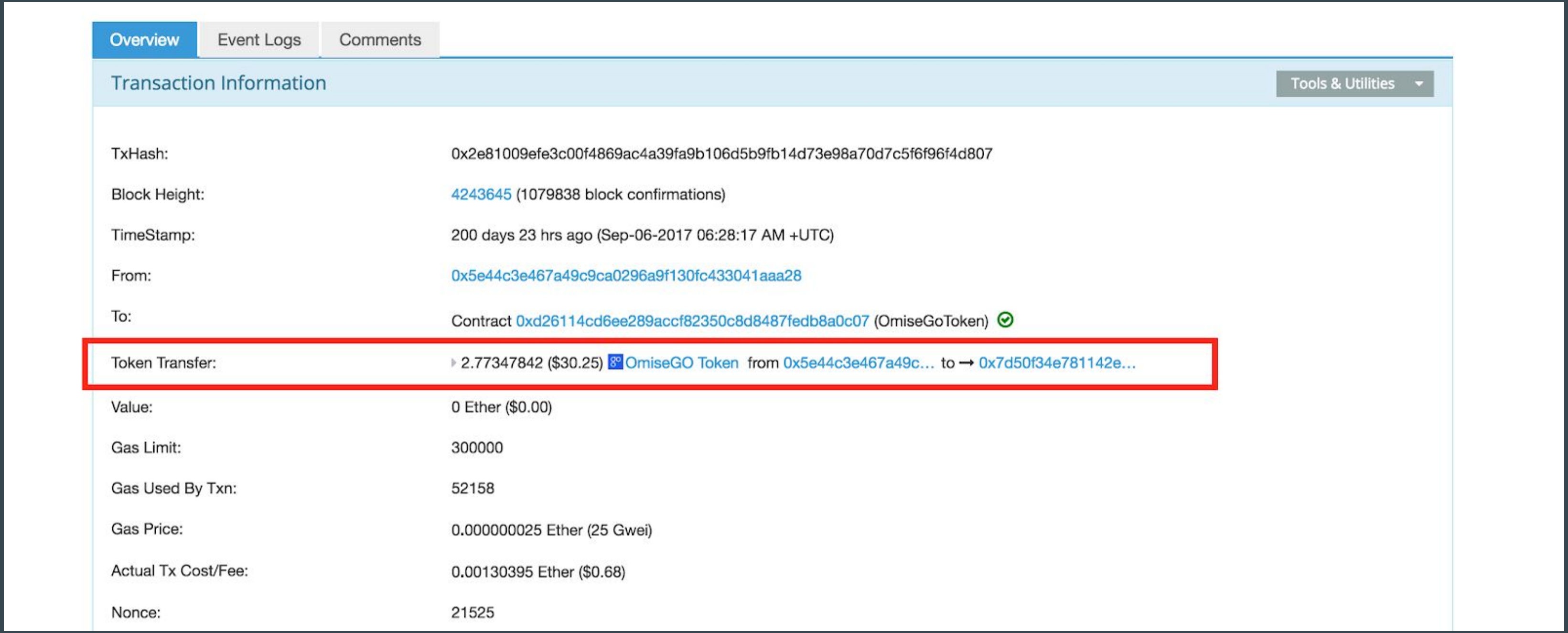
Exchange Buy Exchange Sell

ICOs

- An initial coin offering (ICO) is the cryptocurrency industry’s equivalent of an initial public offering (IPO).
- A company seeking to raise money to create a new coin, app, or service can launch an ICO as a way to raise funds.
- Usually consists of people buying future coins using an existing cryptocurrency, and then receiving those coins when the network launches
  - This attempts to ‘start’ the market of the coin. Assuming the ICO is bought, the digital currency is now in the hands of adopters, investors, etc, and effectively has a price
- Two main types of ICO:
  - ERC20 tokens — coins built on top of the Ethereum network
  - Non-ERC20 tokens — coins that launch their own separate blockchain from every other coin
  - Because non-ERC20 coins are all unique and can have vastly different ICO strategies, we will focus on ERC20 in this course

ERC20 Tokens

- ERC-20 is the technical standard for fungible tokens created using the Ethereum blockchain.
- ERC20 defines certain functions and behaviors that tokens have to implement.
- Following are the functions needed to define a ERC20 Token:
    - totalSupply — gets total supply of the coin
    - balanceOf(address a) — gets the account balance of account with address a
    - transfer(address a, uint256 value) — transfer value coins to address a
    - transferFrom(address src, address dest, uint256 value) — transfer value coins from address src to address dest
    - approve(address spender, uint256 value) — allow spender to withdraw up to value coins from your account
    - allowance(address owner, address spender) — returns the amount which sender is allowed to withdraw from owner


Overview | Event Logs | Comments

Transaction Information

Tools & Utilities

TxHash:	0x2e81009efe3c00f4869ac4a39fa9b106d5b9fb14d73e98a70d7c5f6f96f4d807
Block Height:	4243645 (1079838 block confirmations)
TimeStamp:	200 days 23 hrs ago (Sep-06-2017 06:28:17 AM +UTC)
From:	0x5e44c3e467a49c9ca0296a9f130fc433041aaa28
To:	Contract 0xd26114cd6ee289accf82350c8d8487fedb8a0c07 (OmiseGoToken)
Token Transfer:	2.77347842 ($30.25) OmiseGO Token from 0x5e44c3e467a49c... to → 0x7d50f34e781142e...
Value:	0 Ether ($0.00)
Gas Limit:	300000
Gas Used By Txn:	52158
Gas Price:	0.000000025 Ether (25 Gwei)
Actual Tx Cost/Fee:	0.00130395 Ether ($0.68)
Nonce:	21525

ERC20 Token

- How do you buy into an ICO?
- Remember ERC20 tokens are built on top of Ethereum's network
- This means your Ethereum address is also your token address
- You send your ether to the ICO, they can just send the token right back to the same address upon release.
- The ICO sets up a smart contract on Ethereum's network
- Smart contracts have ethereum addresses, and run whenever they receive any amount of ethereum, including 0 (as we saw in the last picture)

Summary

- Solidity concepts
- Buying and selling cryptocurrency
- ICOs and the types of tokens
-
-

https://cryptozombies.io/ Assignment for this week

- Lesson 2
  - “Zombies Attack Their Victims”

Final Project

- Out next week, due at the end of the semester, 12/11 at midnight
- You will implement your own smart contract by devising your own applications which meets the requirements in the project's README.
- CryptoZombies and Final Project