Introduction to Solidity: Coding Ethereum Smart Contracts

blockchain nyc

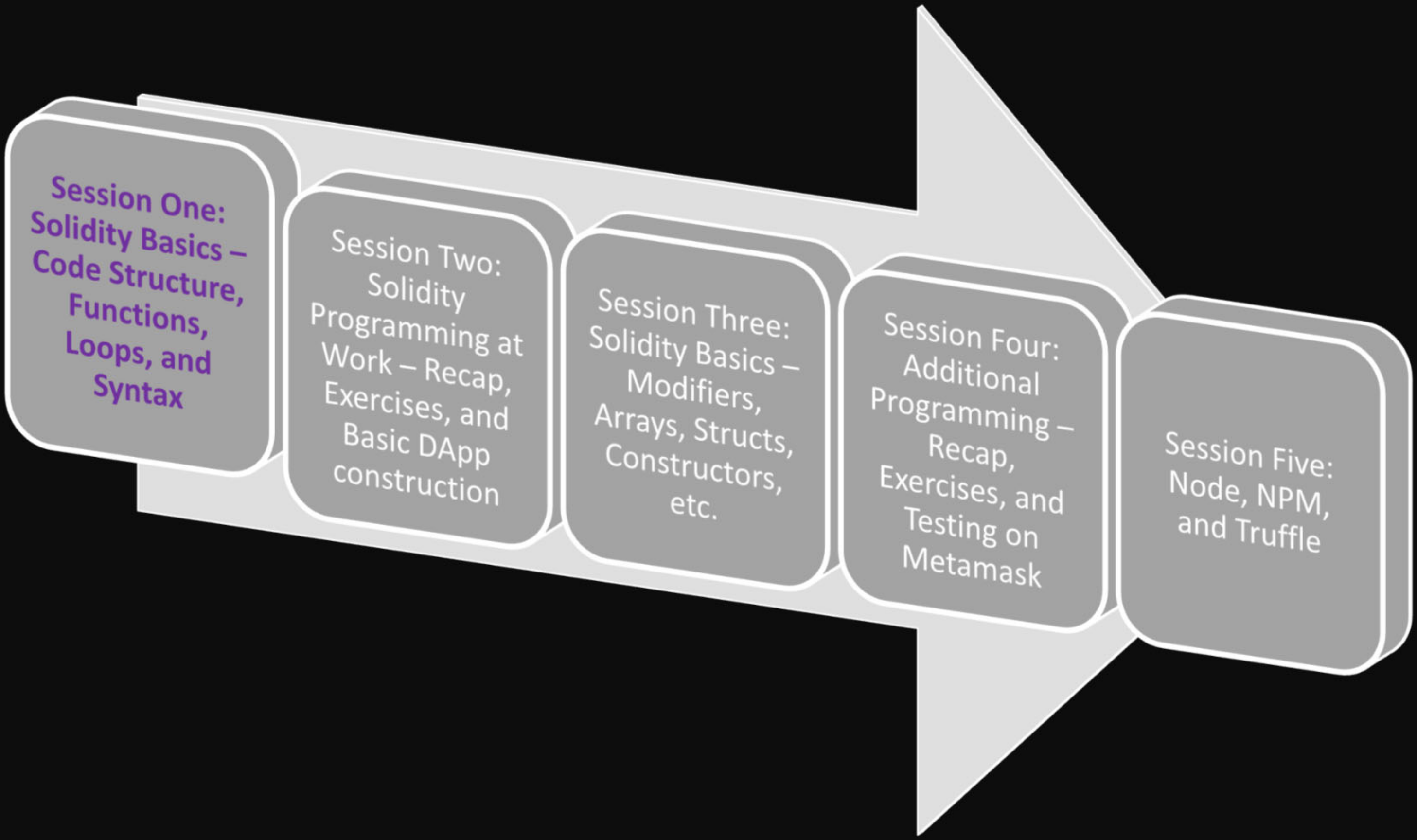
Instructor Bio

Hector Santana

DeFi Analyst, Chainhaus
hector@chainhaus.com

blockchain nyc

Road Map: Bootcamp Details


Session One: Solidity Basics – Code Structure, Functions, Loops, and Syntax
Session Two: Solidity Programming at Work – Recap, Exercises, and Basic DApp construction
Session Three: Solidity Basics – Modifiers, Arrays, Structs, Constructors, etc.
Session Four: Additional Programming – Recap, Exercises, and Testing on Metamask
Session Five: Node, NPM, and Truffle

Agenda

Session One

blockchain nyc

Session One Prerequisites

- Google Chrome
- Remix - https://remix.ethereum.org/

What is Solidity?

- **What is solidity? -** Solidity is a contract-oriented, high-level programming language for implementing smart contracts.

What is Solidity?

(continued)

- **What is Ethereum? –** A decentralized blockchain platform.

blockchain nyc

What is Solidity?

(continued)

- **What is a smart contract? –** They are computer protocols intended to act in the same manner as a physical financial contract.

The Solidity Toolkit

- **Remix**
- **NPM / Node.js**
- **Truffle**

Your First Smart Contract

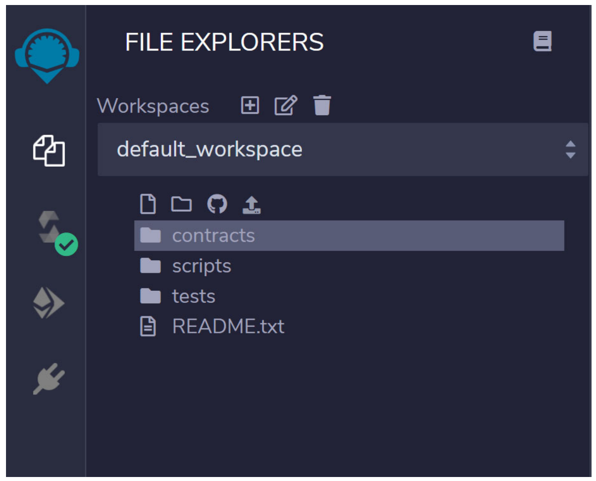
Please head to https://remix.ethereum.org/ and we will begin writing a new Solidity contract from scratch

After the completion of this session, I will provide all solution sets and information via our Slack channel

blockchain nyc

Your First Smart Contract

Breaking Down Remix


FILE EXPLORERS

Workspaces

default_workspace

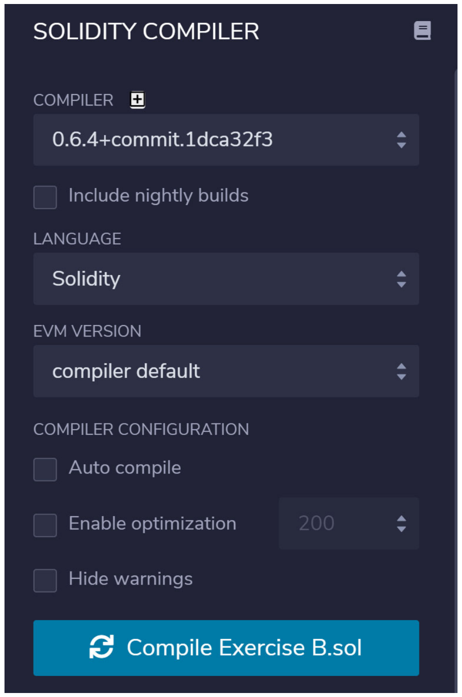
contracts
scripts
tests
README.txt

blockchain nyc

Your First Smart Contract

(continued)

Breaking Down Remix



SOLIDITY COMPILER

COMPILER
0.6.4+commit.1dca32f3

Include nightly builds

LANGUAGE
Solidity

EVM VERSION
compiler default

COMPILER CONFIGURATION

Auto compile

Enable optimization 200

Hide warnings

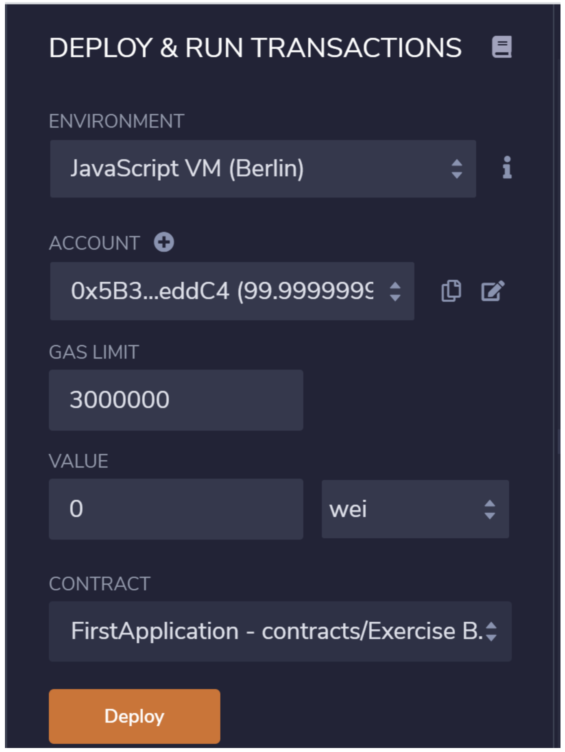
Compile Exercise B.sol

blockchain nyc

Your First Smart Contract

(continued)

Breaking Down Remix



DEPLOY & RUN TRANSACTIONS

ENVIRONMENT

JavaScript VM (Berlin)

ACCOUNT

0x5B3...eddC4 (99.9999999

GAS LIMIT

3000000

VALUE

0 wei

CONTRACT

FirstApplication - contracts/Exercise B.

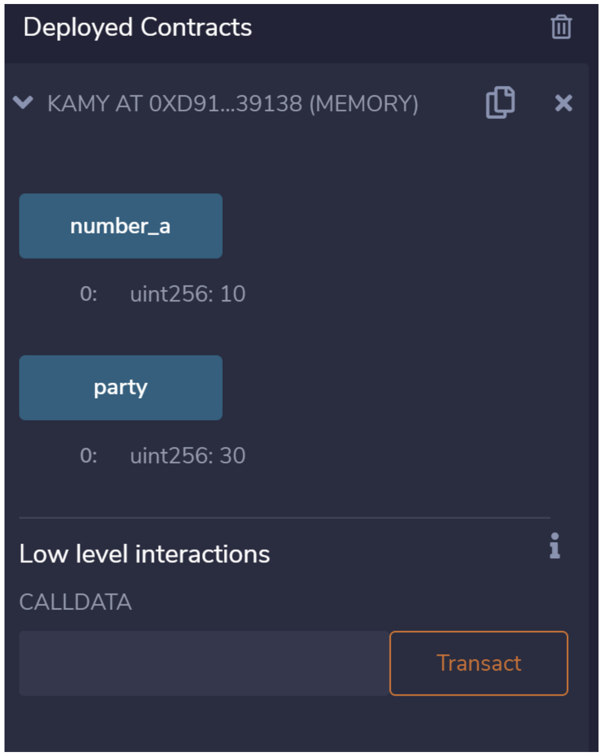
Deploy

blockchain nyc

Your First Smart Contract

(continued)

Breaking Down Remix



Deployed Contracts

KAMY AT 0XD91...39138 (MEMORY)

```
0:    uint256: 10
```

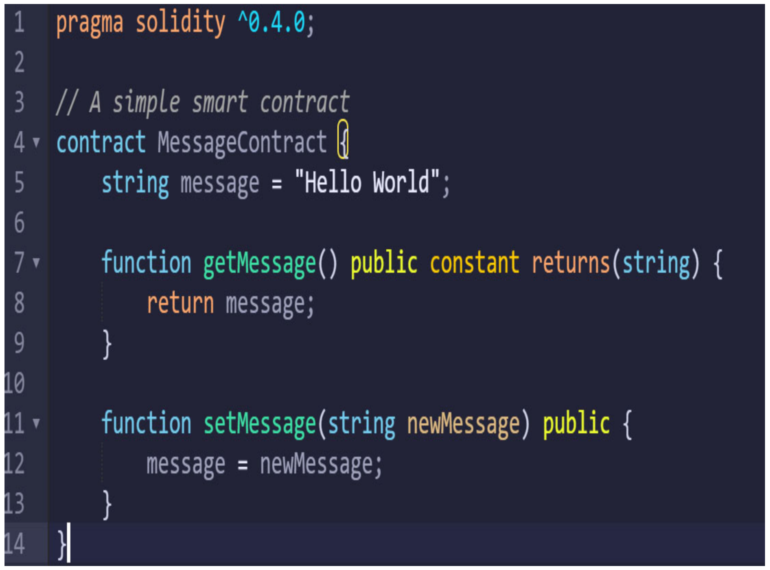
```
0:    uint256: 30
```

Low level interactions

CALLDATA

Your First Smart Contract *(continued)*

Hello World!



```
pragma solidity ^0.4.0;

// A simple smart contract
contract MessageContract {
    string message = "Hello World";

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

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

blockchain nyc

QUESTIONS?

The Solidity Toolkit

- **Remix**
- **NPM / Node.js**
- **Truffle**

blockchain nyc

Breaking Down Solidity: Basic Code Structure

blockchain nyc

Breaking Down Solidity: Basic Code Structure

Creating A Basic Contract

- In order to denote the creation of a contract you must begin a code block with the key word **contract**
  - `i.e. contract TestContract{}`

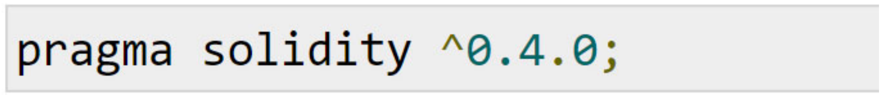
blockchain nyc

Breaking Down Solidity: Basic Code Structure

General Syntax

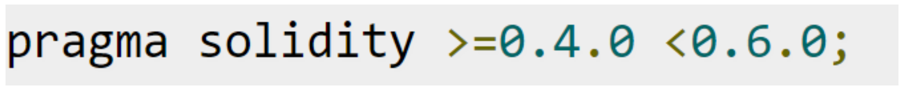
- Solidity is statically typed
- Pragma
- Semicolons

Examples



```
pragma solidity ^0.4.0;
```

Fig. 1.1



```
pragma solidity >=0.4.0 <0.6.0;
```

Fig. 1.2

Breaking Down Solidity: Basic Code Structure

General Syntax (continued)

- Reserved Words

Fig. 1.3

Abstract	after	alias	apply
auto	case	catch	copyof
default	define	final	immutable
implements	in	inline	let
macro	match	mutable	null
of	override	partial	promise
reference	relocatable	sealed	sizeof
static	supports	switch	try
typedef	typeof	unchecked	

** Reference: https://www.tutorialspoint.com/*

Exercise A

Create a contract shell based on the notes thus far. Use pragma solidity 0.6.4.

(Activity Length ~3 minutes)

Breaking Down Solidity: Basic Code Structure

Datatypes and Variables

- Comments
- Address

Breaking Down Solidity: Basic Code Structure

Datatypes and Variables
(continued)

- Value Types

Fig. 1.4

Type	Keyword	Values
Boolean	bool	true/false
Integer	int/uint	Signed and unsigned integers of varying sizes.
Integer	int8 to int256	Signed int from 8 bits to 256 bits. int256 is same as int.
Integer	uint8 to uint256	Unsigned int from 8 bits to 256 bits. uint256 is same as uint.
Fixed Point Numbers	fixed/unfixed	Signed and unsigned fixed point numbers of varying sizes.
Fixed Point Numbers	fixedMxN	Signed fixed point number where M represents number of bits taken by type and N represents the decimal points. M should be divisible by 8 and goes from 8 to 256. N can be from 0 to 80. fixed is same as fixed128x18.
Fixed Point Numbers	ufixedMxN	Unsigned fixed point number where M represents number of bits taken by type and N represents the decimal points. M should be divisible by 8 and goes from 8 to 256. N can be from 0 to 80. ufixed is same as ufixed128x18.

** Reference: https://www.tutorialspoint.com/*

blockchain nyc

Breaking Down Solidity: Basic Code Structure

Datatypes and Variables
(continued)

Types of Variables:

- State Variables
- Local Variables
- Global Variables

Variable Name Rules:

- Do not create a variable using a reserved keyword.
- Do not start a variable name with a numeral. All variable naming conventions should begin with an underscore or letter.
- Solidity variables are case-sensitive; lowercase variable names are preferred.

Breaking Down Solidity: Basic Code Structure

Datatypes and Variables
(continued)

Variable Scope:

- Public
- Internal
- Private

Variable Examples:

- `uint public number_a = 40`
- `uint internal number_b = 30`

Breaking Down Solidity: Basic Code Structure

Datatypes and Variables (continued)

- Global Variables

Fig. 1.5

Name	Returns
blockhash(uint blockNumber) returns (bytes32)	Hash of the given block - only works for 256 most recent, excluding current, blocks
block.coinbase (address payable)	Current block miner's address
block.difficulty (uint)	Current block difficulty
block.gaslimit (uint)	Current block gaslimit
block.number (uint)	Current block number
block.timestamp (uint)	Current block timestamp as seconds since unix epoch
gasleft() returns (uint256)	Remaining gas
msg.data (bytes calldata)	Complete calldata
msg.sender (address payable)	Sender of the message (current caller)
msg.sig (bytes4)	First four bytes of the calldata (function identifier)
msg.value (uint)	Number of wei sent with the message
now (uint)	Current block timestamp
tx.gasprice (uint)	Gas price of the transaction
tx.origin (address payable)	Sender of the transaction

** Reference: https://www.tutorialspoint.com/*

blockchain nyc

Appendix:

Additional Learning Resources

- *Crypto Zombies Solidity Tutorial -* *https://cryptozombies.io/*
- *Full YouTube Tutorial -* *https://www.youtube.com/watch?v=ipwxYa-F1uY*
- *Mastering Ethereum: Building Smart Contracts and DApps 1st Edition, by Andreas M. Antonopoulos -* *https://www.amazon.com/Mastering-Ethereum-Building-Smart-Contracts/dp/1491971940*

Solution Set

Hello World

```
pragma solidity ^0.4.0;

// A simple smart contract
contract MessageContract {
    string message = "Hello World";

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

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

Exercise A

```
pragma solidity 0.6.4;
contract kamy {
}
```