CREATION AND DEPLOYMENT OF SMART CONTRACTS ON ETHEREUM BLOCKCHAIN

BY

DR HIDAIA MAHMOOD ALASSOULI

Creation and Deployment of Smart Contracts on Ethereum Blockchain

By

Dr. Hidaia Mahmood Alassouli

Hidaia_alassouli@hotmail.com

1. Overview:

This work explains briefly the creation and deployment Of Smart Contract on Ethereum Blockchain. The work consists from the following sections

- Blockchain
- Solidity variables and types
- How to Setup or Install Ethereum on Windows
- How to compile and deploy smart contract on JavaScriptVM
- How to install Ganache Blockchain on Windows and deploy smart contract using it.
- How to compile and deploy Smart Contract on Test Networks,
- Quick example of deploying ERC20 token smart contract.
- Getting started tutorial on Solidity
- Creating ERC-20 smart contract and crowd sale (ICO) smart contract without coding
- ERC-20 smart contract and crowd sale (ICO) smart contract:
- Creating Ethereum ERC-20 Tokens and Crowd Sales (ICO) without coding with Token Wizard:
- Example of creating and deploying an ERC20 token on the test and main network!!!

2. Blockchain:

A blockchain is a digital record of transactions. The name comes from its structure, in which individual records, called blocks, are linked together in single list, called a chain. Blockchains are used for recording transactions made with cryptocurrencies, such as Bitcoin, and have many other applications.

Each transaction added to a blockchain is validated by multiple computers on the Internet. These systems, which are configured to monitor specific types of blockchain transactions, form a peer-to-peer network. They work together to ensure each transaction is valid before it is added to the blockchain. This decentralized network of computers ensures a single system cannot add invalid blocks to the chain.

When a new block is added to a blockchain, it is linked to the previous block using a cryptographic hash generated from the contents of the previous block. This ensures the chain is never broken and that each block is permanently recorded. It is also intentionally difficult to alter past transactions in blockchain since all the subsequent blocks must be altered first.

- **Blockchain Uses:** While blockchain is widely known for its use in cryptocurrencies such as Bitcoin, Litecoin, and Ether, the technology has several other uses. For example, it enables "smart contracts," which execute when certain conditions are met. This provides an automated escrow system for transactions between two parties. Blockchain can potentially be used to allow individuals to pay each other without a central clearing point, which is required for ACH and wire transfers. It has potential to greatly increase the efficiency of stock trading by allowing transactions to settle almost instantly instead of requiring three or more days for each transaction to clear. Blockchain technology can also be used for non-financial purposes. For example, the InterPlanetary File System (IFPS) uses blockchain to decentralize file storage by linking files together over the Internet. Some digital signature platforms now use blockchain to record signatures and verify documents have been digitally signed. Blockchain can even be used to protect intellectual property by linking the distribution of content to the original source.

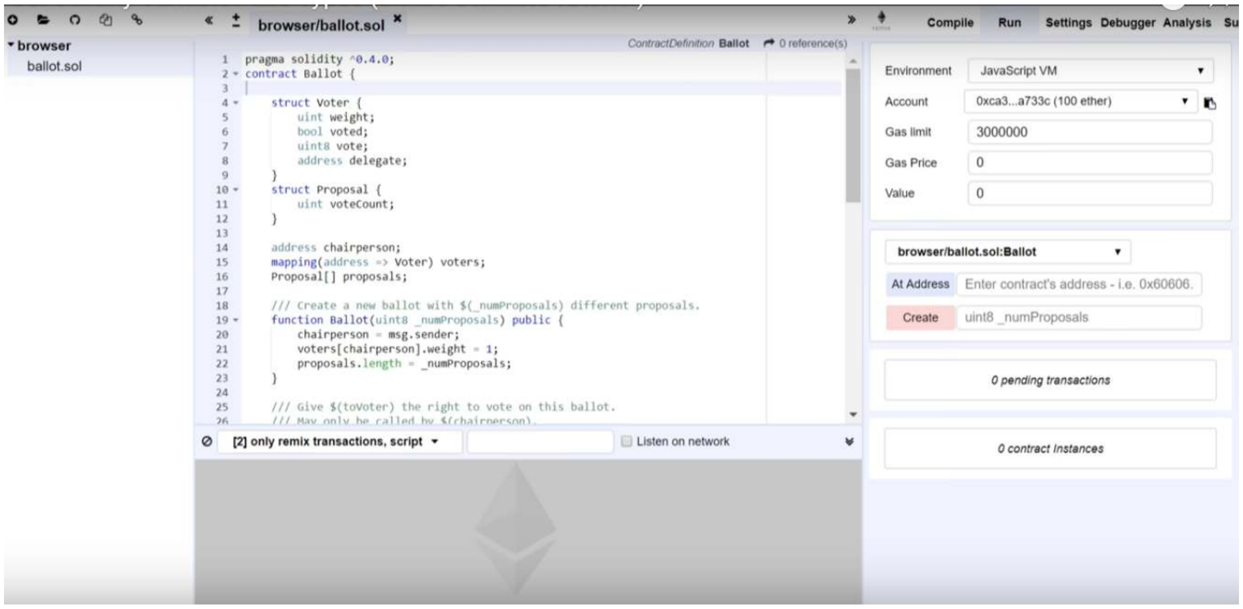
- Blockchain is decentralized database that consists from records and data. The entire data is spread out in bounch of distributed computers to end to be pear to pear p2p network. The records in these databases are stored in form of blocks which linked together in secured way.
- The bitcoin blockchain stores data specific to keep track of crypto currencies balances between different parties.
- Ethereum block chain can store much more deep data than bitcoin block chain. Ethereum block chain allows building decentralized apps. These decentralized apps defined by smart contracts. So smart contract allows individuals to exchange information in trusted confident free manner without relying on third party as bank or lawyer or other way. So these Ethereum smart contracts stored in special transections in Ethereum blockchain which can be used then to build applications. So you can think on smart contracts as decentralized api's. As smart contracts stored in Ethereum blockchains, they need to be validated or mined as any other transactions.

So there is small costs associated with deploying smart contracts.

- How to code smart contract. We do that with solidity. Solidity is a code used. Solidity is a language used to code smart contract. And syntax is similar to java script and is designed with Ethereum virtual machine in mind. So we will use solidity to create smart contract in web based ide called remex

3. Solidity variables and types

- You can write the solidity and test code in https://remix.ethereum.org/.
- Here screen of https://remix.ethereum.org



```
pragma solidity ^0.4.0;
contract Ballot {

    struct Voter {
        uint weight;
        bool voted;
        uint8 vote;
        address delegate;
    }
    struct Proposal {
        uint voteCount;
    }

    address chairperson;
    mapping(address => Voter) voters;
    Proposal[] proposals;

    /// Create a new ballot with $(_numProposals) different proposals.
    function Ballot(uint8 _numProposals) public {
        chairperson = msg.sender;
        voters[chairperson].weight = 1;
        proposals.length = _numProposals;
    }

    /// Give $(toVoter) the right to vote on this ballot.
    /// May only be called by $(chairperson).
```

Environment: JavaScript VM
Account: 0xca3...a733c (100 ether)
Gas limit: 3000000
Gas Price: 0
Value: 0

browser/ballot.sol:Ballot

At Address: Enter contract's address - i.e. 0x60606.

Create: uint8 _numProposals

0 pending transactions

0 contract instances

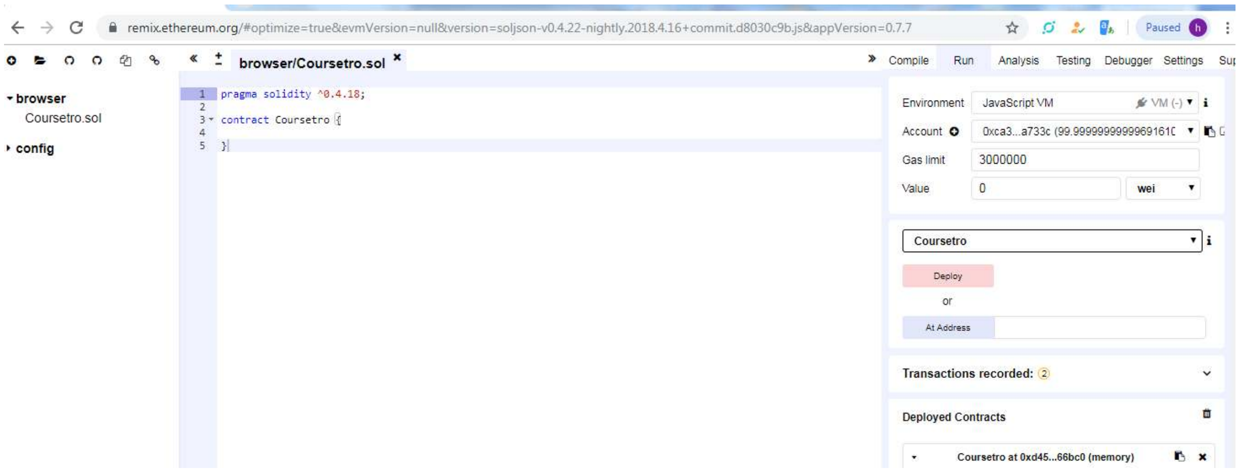
- Create new contract. I called it Coursetro.
- Start off by writing (or pasting) the following code:

```
pragma solidity ^0.4.18;

contract Coursetro {


}
```

The very first line defines the version of solidity you're going to use.

Next, you define the contract and its name and open it up like a JavaScript class.

- On the right side of the browser under the Run tab, you will see a Deploy button. Click on this. You will notice that underneath the Deploy button shows a new section with the name of the contract and "at 0x..." (memory):


remix.ethereum.org/#optimize=true&evmVersion=null&version=soljson-v0.4.22-nightly.2018.4.16+commit.d8030c9b.js&appVersion=0.7.7
browser/Coursetro.sol
browser
Coursetro.sol
config
Compile
Run
Analysis
Testing
Debugger
Settings
Deploy
or
At Address
Paused

```
pragma solidity ^0.4.18;

contract Coursetro {

}
```

Environment: JavaScript VM
Account: 0xca3...a733c (99.9999999999969161
Gas limit: 3000000
Value: 0 wei
Coursetro
Transactions recorded: 2
Deployed Contracts
Coursetro at 0xd45...66bc0 (memory)

- This means that the smart contract (yes, it's not very smart at this point) lives at an address. To see this full address along with other information, in the debugger if you click on the **Details** button, it will provide you with more information:
- Notice there's a **contractAddress**, this is where the smart contract actually lives. No, it's not live on the Ethereum Blockchain because right now, we're simply working within the Javascript EVM.
- Also notice **gas**. Every time a contract is deployed and modified, nodes on the Ethereum network must verify the contract. It's referred to as being redundantly parallel, as a way to reach consensus.Gas is the name for the execution fee that senders of transactions (in our case, senders of a smart contract transaction) will pay for verification.
- **Elaborating on this further**, from etherdocs.org:
- *"Gas is the name for the execution fee that senders of transactions need to pay for every operation made on an Ethereum blockchain. The name gas is*

inspired by the view that this fee acts as cryptofuel, driving the motion of smart contracts. Gas is purchased for ether from the miners that execute the code. Gas and ether are decoupled deliberately since units of gas align with computation units having a natural cost, while the price of ether generally fluctuates as a result of market forces. The two are mediated by a free market: the price of gas is actually decided by the miners, who can refuse to process a transaction with a lower gas price than their minimum limit. To get gas you simply need to add ether to your account. The Ethereum clients automatically purchase gas for your ether in the amount you specify as your maximum expenditure for the transaction."

- In Solidity, you define a variable by first specifying its type.

What other types are there?

a. **bool**
   This is a Boolean, which returns true or false.
b. **int / uint**
   Both int and uint represent integers, or number values. The primary difference between int and uint (Unsigned Integer), is that int can hold negative numbers as values.
c. **address**
   The address type represents a 20 byte value, which is meant to store an Ethereum address. Variables that are typed as address also have members, including balance and transfer.
d. **bytes1 through 32**
   This is a fixed-size byte array.
e. **bytes**
   A dynamically-sized byte array.

f. **string**
   A dynamically signed string.
g. **mapping**
   Hash tables with key types and value types. We will look at mappings more in depth later on in the course.
h. **struct**
   Structs allow you to define new types. We will also cover this more in depth shortly.

- Let's define a string variable in our contract. Let's also define my age. No one can have a negative age, so we will use an unsigned integer for this:

```
pragma solidity ^0.4.18;

contract Coursetro {

   string fName = 'Gary';
   uint age = 34;


}
```

- Solidity has four types of visibilities for both functions and variables:

1. **Public**
   This allows you to define functions or variables that can be called internally or through messages.

2. **Private**
   Private variables and functions are only available to the current contract and not derived contracts.

3. **Internal**
   Fuctions and variables that can only be accessed internally (current contract or derived).