

Tutorial 2: Deploy and develop a smart contract on Ethereum

Part 1 – Deploy and interact with your smart contract in your Private Network

At the end of your exercise push the code of the smart contract on your github project
Blockchain distributed system & Smart contract where you create a repository name
“Tutorial 2 : Part 1”.

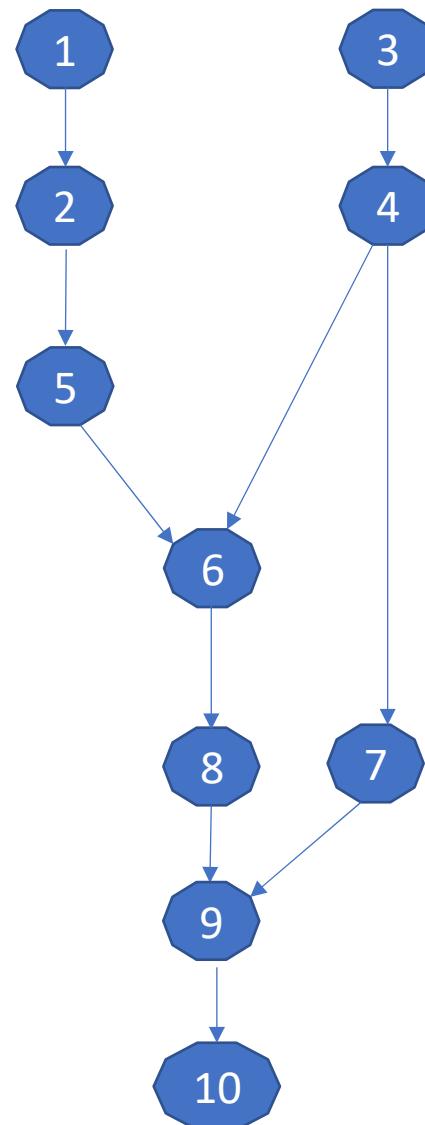
When you finished to complete the Quest please provide your github account with all
files (configuration files, smart contract code...) created for this tutorial. You have to
create also a report with all print screens for each steps.

Different tools provide different functionality

	Tools	Remix	Ganache	MyEtherWallet	Geth
	Activities				
1	Configure the Blockchain	-	-	-	+
2	Deploy the Blockchain	Not Persistent	+	-	+
3	Develop the contract	+	-	-	+
4	Compile the contract	+	-	-	+
5	Create user account	+	+	+	+
6	Deploy the contract	+	-	+	+
7	Create the UI for interacting	+	-	+	+
8	Run the client	+	-	+	+
9	Interact with the contract & have fun	+	-	+	+
10	Monitor the execution	-	+	-	+

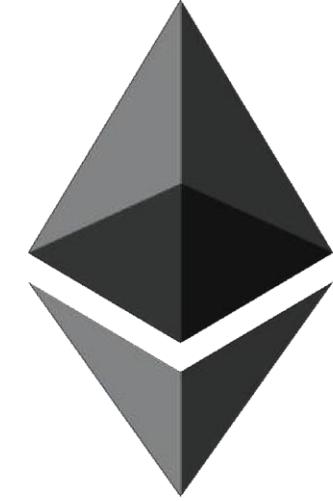
<https://remix.ethereum.org/>

<http://truffleframework.com/ganache/>



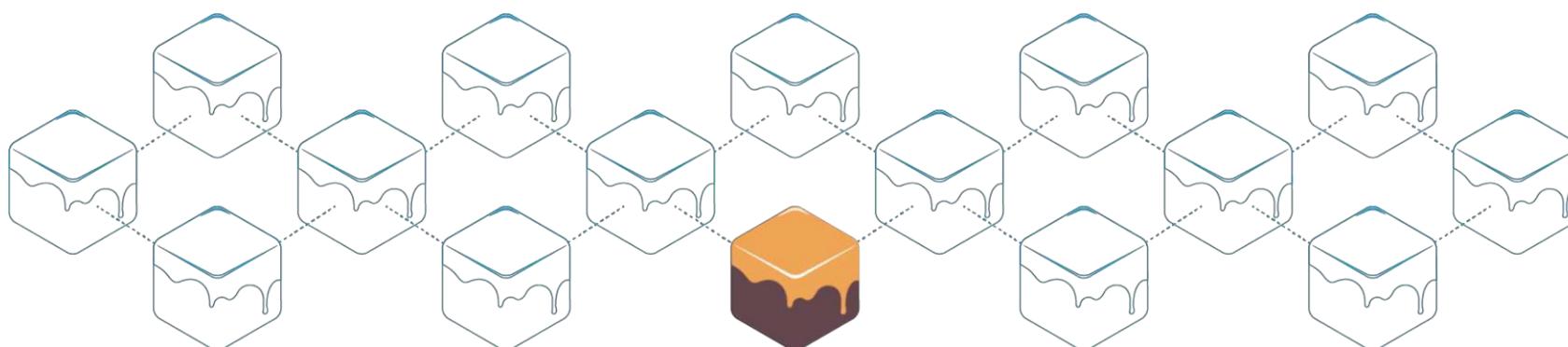
Use which tool for what purpose? (1/2)

- Use Geth for everything?
 - Powerful but command-line only
- What should I use?
 - As a starting point for developing contracts – mostly Remix
- What cannot Remix do?
 - Configure the blockchain
 - Create real (non-test) user accounts and transfer funds between user accounts
 - Monitor the execution
 - Other advanced operations



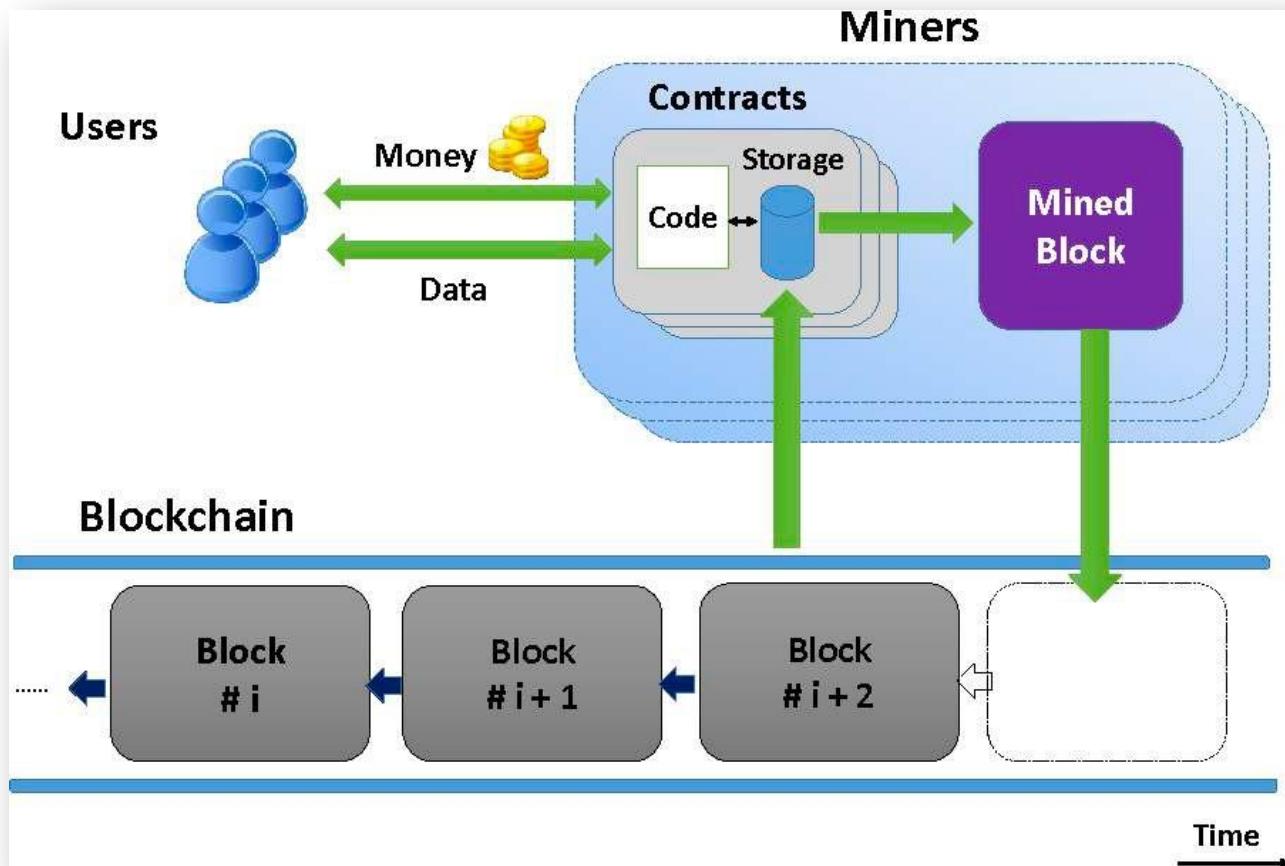
Use which tool for what purpose? (2/2)

- Why use Ganache?
 - To inspect and monitor the execution
 - To visualize certain elements in a better way
- Why use MyEtherWallet?
 - To create a personal wallet (real user account), transfer funds between user accounts, and interact with contracts
 - Metamask as another alternative



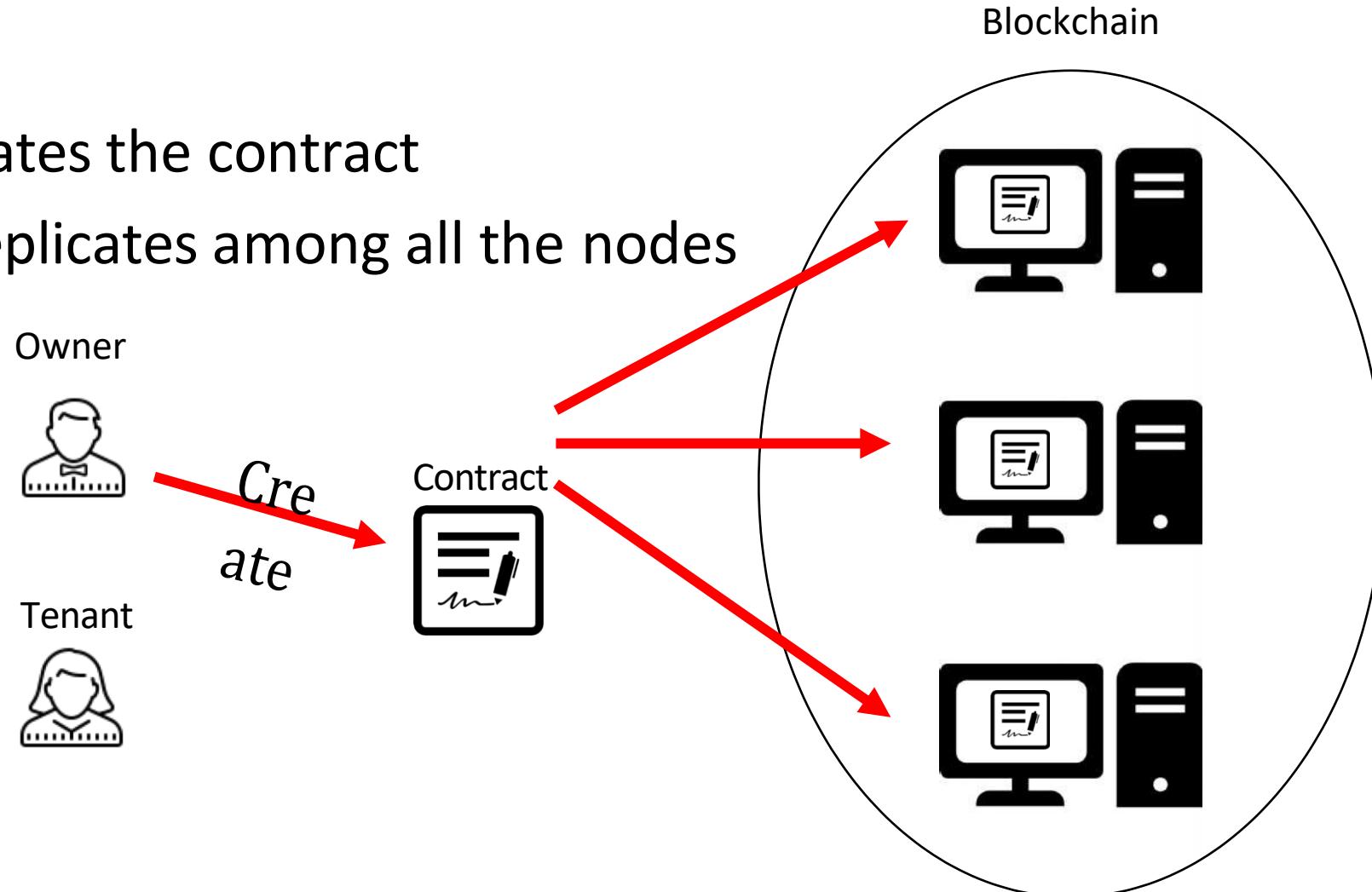
Smart Contracts

- In the form of code
- Stored on a blockchain
- Executes under given conditions



Smart Contracts Example (1/3)

- Owner creates the contract
- Contract replicates among all the nodes



Smart Contracts Example (2/3)

- Tenant deposits to the contract
- Contract's State changes on all the nodes

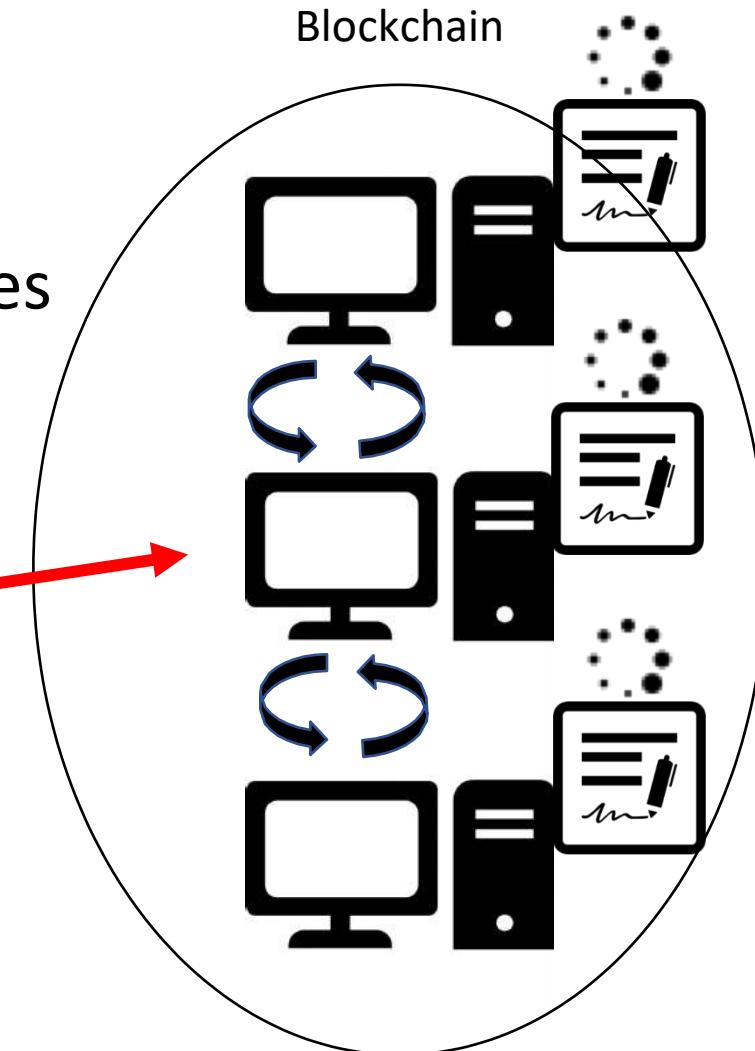
Owner



Tenant

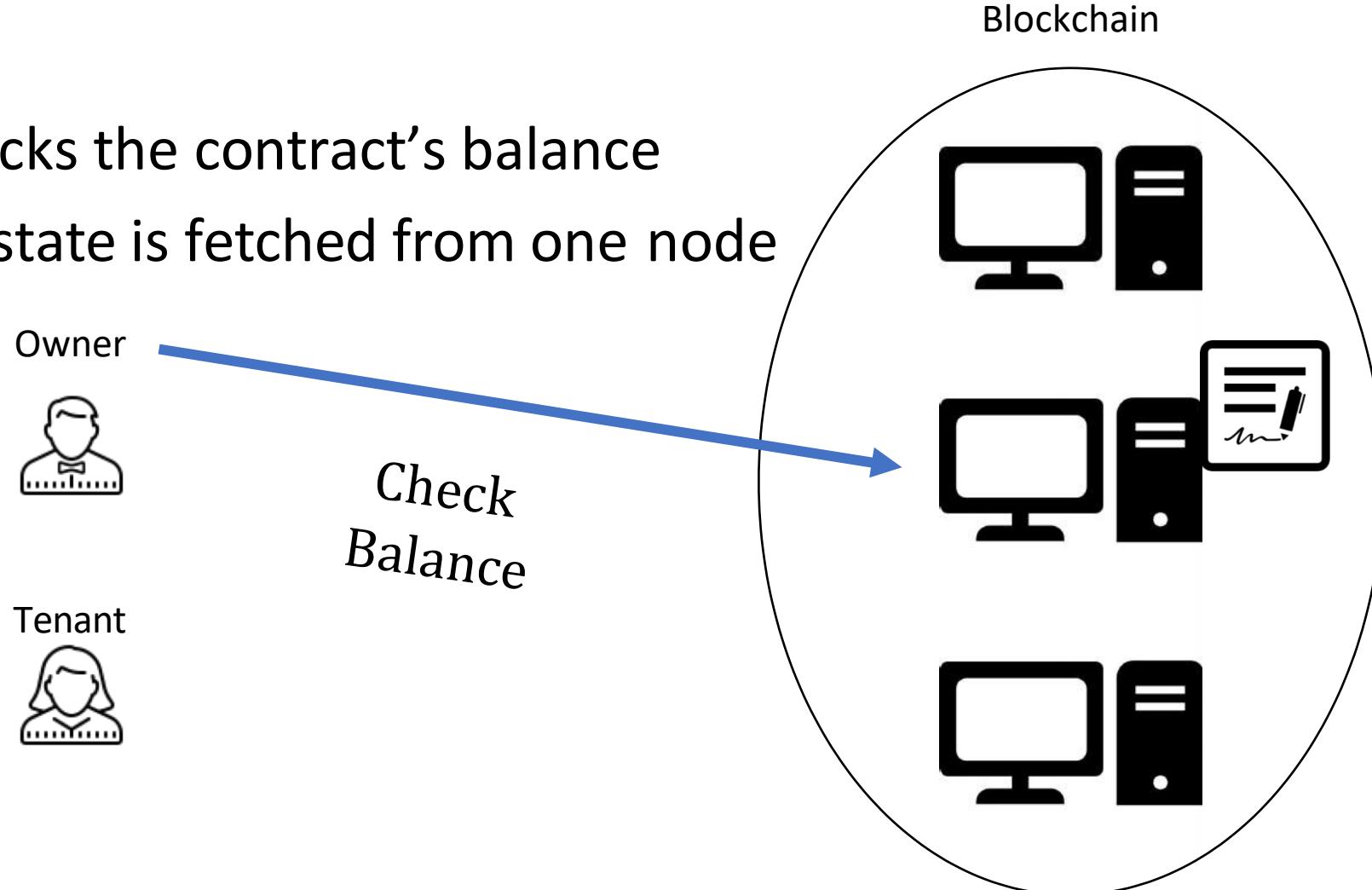


Deposit



Smart Contracts Example (3/3)

- Owner checks the contract's balance
- Contract's state is fetched from one node



Smart Contracts

1. Developing a simple contract
2. Compiling the contract
3. Deploying the contract
4. Interacting with the contract
5. Adding more functions to our code to make it more practical

Open Remix : remix.ethereum.org

- An open source tool for writing, compiling and testing Solidity contracts

The screenshot shows the Open Remix web interface for Ethereum development. On the left, there's a file tree with 'browser' and 'config' selected, and a code editor containing the following Solidity code:

```
1 pragma solidity ^0.5.0;
2
3 contract financialContract {
4
5     uint balance = 313000;
6
7     function getBalance() public view returns(uint){
8         return balance;
9     }
10
11    function deposit(uint newDeposit) public{
12        balance = balance + newDeposit;
13    }
14
15 }
16
```

The right side of the interface includes a toolbar with 'Compile', 'Run', 'Analysis', 'Testing', 'Debugger', 'Settings', and 'Support'. Below the toolbar, it says 'Current version: 0.5.1+commit.c8a2cb62.Emscripten clang'. There are checkboxes for 'Auto compile', 'Enable Optimization', and 'Hide warnings', along with a 'Start to compile (Ctrl-S)' button. A 'financialContract' tab is open, showing tabs for 'Details', 'ABI', and 'Bytecode'. At the bottom, a transaction history section titled '[2] only remix transactions, script' lists:

- Checking transactions details and start debugging.
- Running JavaScript scripts. The following libraries are accessible:
 - [web3 version 1.0.0](#)
 - [ethers.js](#)
 - [swarmgw](#)
 - compilers - contains currently loaded compiler
- Executing common command to interact with the Remix interface (see list of commands above). Note that these commands can also be included and run from a JavaScript script.
- Use `exports/.register(key, obj)/.remove(key)/.clear()` to register and reuse object across script executions.

Solidity

- Object-oriented
- Contract-oriented
- High-level language
- Influenced by C++, Python, and JavaScript
- Target Ethereum Virtual Machine (EVM)



Serpent as an Alternative?

- Low-level language
- Complex compiler

Start Coding

- Setter and Getter: Set and get the information.

```
1 pragma solidity ^0.5.0;
2
3 contract financialContract {
4
5     uint balance = 313000;           ← Variable
6
7     function getBalance() public view returns(uint){
8         return balance;
9     }
10
11    function deposit(uint newDeposit) public{
12        balance = balance + newDeposit;
13    }
14
15 }
```

The diagram illustrates the components of the Solidity code. A yellow oval highlights the variable declaration `uint balance = 313000;`, which is connected by a yellow arrow to a yellow-bordered box labeled "Variable". A blue rounded rectangle surrounds the getter function `function getBalance() public view returns(uint){ return balance; }`, which is connected by a blue arrow to a blue-bordered box labeled "Getter function". A red rounded rectangle surrounds the setter function `function deposit(uint newDeposit) public{ balance = balance + newDeposit; }`, which is connected by a red arrow to a red-bordered box labeled "Setter function".

Compile the Contract

- Compile tab: Start to compile button

The screenshot shows the Truffle UI interface for compiling a Solidity contract. On the left, the code editor displays the file `browser/firstContract.sol` with the following content:

```
1 pragma solidity ^0.5.0;
2
3 contract financialContract {
4     uint balance = 313000;
5
6     function getBalance() public view returns(uint){
7         return balance;
8     }
9
10    function deposit(uint newDeposit) public{
11        balance = balance + newDeposit;
12    }
13
14 }
```

On the right, the toolbars and settings are visible. The "Compile" tab is selected. A message at the top right indicates the current compiler version: "Current version: 0.5.1+commit.c8a2cb62.Emscripten clang". Below it is a dropdown menu for selecting a new compiler version. Underneath are checkboxes for "Auto compile" (checked), "Enable Optimization" (unchecked), and "Hide warnings" (unchecked). The prominent "Start to compile (Ctrl-S)" button is highlighted with a red rectangular border. At the bottom right, there are dropdown menus for "financialContract" and "Swarm".

Set Deployment Parameters (1/2)

- Run tab: Environment = JavaScript VM

The screenshot shows the Truffle UI interface. On the left, there is a code editor window titled "browser/firstContract.sol" containing the following Solidity code:

```
1 pragma solidity ^0.5.0;
2
3 contract financialContract {
4
5     uint balance = 313000;
6
7     function getBalance() public view returns(uint){
8         return balance;
9     }
10
11    function deposit(uint newDeposit) public{
12        balance = balance + newDeposit;
13    }
14
15 }
16
```

On the right, the "Run" tab is selected, showing deployment parameters:

- Environment: JavaScript VM (highlighted with a red box)
- Account: 0xca3...a733c (100 ether)
- Gas limit: 3000000
- Value: 0 wei

Below these settings, there is a "financialContract" dropdown and a "Deploy" button.

Set Deployment Parameters (2/2)

- JavaScript VM: All the transactions will be executed in a sandbox blockchain in the browser. Nothing will be persisted and a page reload will restart a new blockchain from scratch, the old one will not be saved.
- Injected Provider: Remix will connect to an injected web3 provider. Mist and Metamask are example of providers that inject web3, thus they can be used with this option.
- Web3 Provider: Remix will connect to a remote node. You will need to provide the URL address to the selected provider: geth, parity or any Ethereum client.
- Gas Limit: The maximum amount of gas that can be set for all the instructions of a contract.
- Value: Input some ether with the next created transaction (wei = 10^{-18} of ether).

Types of Blockchain Deployment

- Private: e.g., Ganache sets a personal Ethereum blockchain for running tests, executing commands, and inspecting the state while controlling how the chain operates.
- Public Test (Testnet): Like Ropsten, Kovan and Rinkeby which are existing public blockchains used for testing and which do not use real funds. Use faucet for receiving initial virtual funds.
- Public Real (Mainnet): Like Bitcoin and Ethereum which are used for real and which available for everybody to join.

Deploy the Contract on the Private Blockchain of Remix

- Run tab: Deploy button

The screenshot shows the Remix IDE interface with the following details:

- Code Editor:** The file is named "browser/firstContract.sol". The code defines a contract named "financialContract" with two functions: "getBalance" and "deposit". The "getBalance" function returns the current balance, and the "deposit" function adds a new deposit to the balance.
- Environment:** The environment is set to "JavaScript VM".
- Account:** The account is set to "0xca3...a733c (99.9999999999998644)".
- Gas limit:** The gas limit is set to 3000000.
- Value:** The value is set to 0 wei.
- Deployment:** A red box highlights the "Deploy" button under the contract name "financialContract".
- Transactions recorded:** One transaction is recorded, which is the deployment of the contract.
- Deployed Contracts:** The deployed contract is listed as "financialContract at 0x692...77b3a (memory)". Below it, the "deposit" and "getBalance" functions are shown, also highlighted with red boxes.
- Bottom Bar:** The bar includes a search field for transactions and a note about executing common commands.

• Executing common command to interact with the Remix interface (see list of commands above). Note that these commands can also be included and run from a JavaScript script.

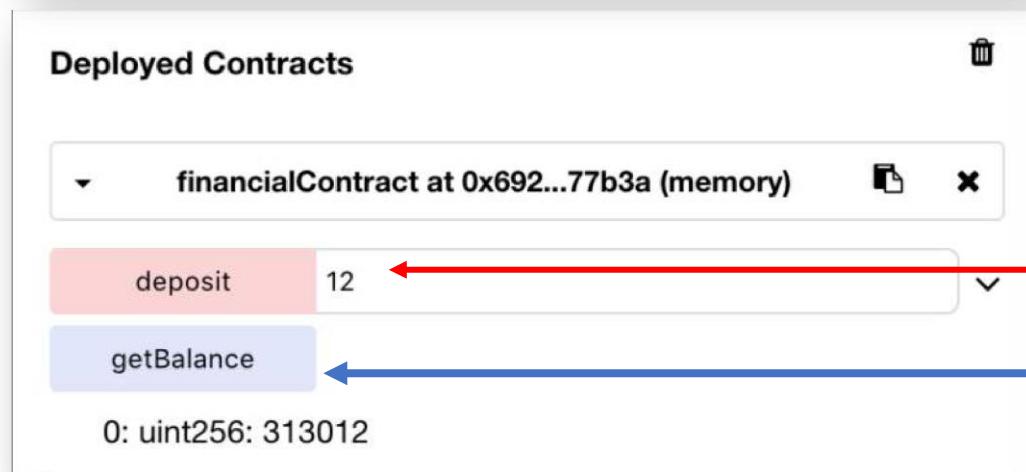
Interact with the Contract

- Setter = Red Button: Creates transaction
- Getter= Blue Button: Just gives information



Press getBalance to see the initial amount

1



Input a value and press deposit button
to create and confirm the transaction

2

Press getBalance again to see the result

3

Additional features

- Transferring funds from an account to the contract
- Saving the address of the contract creator
- Limiting the users' access to functions
- Withdrawing funds from the contract to an account

Receive ether (1/2)

- Transfer money to the contract

Payable keyword
allows receiving
ether

```
1 pragma solidity ^0.5.0;
2
3 contract financialContract {
4
5     function receiveDeposit() payable public{
6         }
7
8
9     function getBalance() public view returns(uint){
10        return address(this).balance;
11    }
12 }
```

Hidden Code:

Address(this).balance += msg.value;

We can get the
balance of the
contract

Receive ether (2/2)

1

Input the value as wei
(10^{-18} of ether)

Environment JavaScript VM VM (-)

Account 0xca3...a733c (99.9999999999998944)

Gas limit 3000000

Value 100 wei

financialContract

Deploy

or

At Address

Load contract from Address

Transactions recorded: 1

Deployed Contracts

financialContract at 0x692...77b3a (memory)

receiveDeposit

getBalance

2

Click the receiveDeposit button to transfer the money to the contract

Constructor

- Will be called at the creation of the instance of the contract

```
1 pragma solidity ^0.5.0;
2
3 contract financialContract {
4
5     address owner;
6
7     constructor() public{
8         owner = msg.sender;
9     }
10
11    function receiveDeposit() payable public{
12
13    }
14
15    function getBalance() public view returns(uint){
16        return address(this).balance;
17    }
18 }
```

We want to save
the address of the
contract creator

Withdraw funds

- Modifier: Conditions you want to test in other functions
- First the modifier will execute, then the invoked function

Only the contract's creator is permitted to withdraw

Transfer some money from the contract's balance to the owner

```
1 pragma solidity ^0.5.0;
2
3 contract financialContract {
4
5     address owner;
6
7 constructor() public{
8     owner = msg.sender;
9 }
10
11 modifier ifOwner(){
12     if(owner != msg.sender){
13         revert();
14     }else{
15         -
16     }
17 }
18
19
20 function receiveDeposit() payable public{
21
22 }
23
24 function getBalance() public view returns(uint){
25     return address(this).balance;
26 }
27
28 function withdraw(uint funds) public ifOwner{
29     msg.sender.transfer(funds);
30 }
31 }
```

Now deploying a smart contract on an external blockchain

	Activities	Tools	Remix	Ganache	MyEtherWallet	Geth
1	Configuring the Blockchain		-	-	-	+
2	Deploying the Blockchain	Not Persistent		+	-	+
3	Developing the contract		+	-	-	+
4	Compiling the contract		+	-	-	+
5	Creating user account		+	+	+	+
6	Deploying the contract		+	-	+	+
7	Creating the UI for interacting		+	-	+	+
8	Run the client		+	-	+	+
9	Interact with the contract & have fun		+	-	+	+
10	Monitoring the execution		-	+	-	+

Run Ganache

Ganache

ACCOUNTS BLOCKS TRANSACTIONS LOGS SEARCH FOR BLOCK NUMBERS OR TX HASHES

CURRENT BLOCK 0 GAS PRICE 20000000000 GAS LIMIT 6721975 NETWORK ID 5777 RPC SERVER HTTP://127.0.0.1:7545 MINING STATUS AUTOMINING

MNEMONIC ? slim rain lawn kiwi elegant behind vibrant dentist puppy reduce kidney there HD PATH m/44'/60'/0'/0/account_index

ADDRESS	BALANCE	TX COUNT	INDEX	
0x231eAeEF9EA93F5370a1F633F32E45AF570980E8	100.00 ETH	0	0	🔑
0x970fc818790E900598C57E48b89B6D3D8896D416	100.00 ETH	0	1	🔑
0xb59BD5568d0be42C13fB521f845243F1CDaF2eF1	100.00 ETH	0	2	🔑

MyEtherWallet

- add your custom network that you want to test your contracts on

The screenshot shows the MyEtherWallet website interface. On the left, there's a large button labeled "Create New Wallet". Below it, a password input field has a red border and contains the text "Do NOT forget to save this!". To the right of the input field is a blue "Create New Wallet" button. A note below the button states: "This password encrypts your private key. This does not act as a seed to generate your keys. You will need this password + your private key to unlock your wallet." At the bottom of this section are links to "How to Create a Wallet" and "Getting Started". On the right side of the screen, there's a "Network ETH" dropdown menu. This menu lists various Ethereum networks and nodes, such as "ETH (myetherapi.com)", "ETH (etherscan.io)", "ETH (infura.io)", etc. A red arrow points from the text "add your custom network that you want to test your contracts on" to the "Add Custom Network / Node" option at the bottom of the dropdown menu, which is also circled in red.

3.21.05 English ▾ Gas Price: 41 Gwei ▾ Network ETH (myetherapi.com) ▾

New Wallet Send Ether & Tokens Swap Send Offline Contracts ENS DomainSale Check TX Status View Wallet Info Help

The network is really full right now. Ch

Create New Wallet

Enter a password

Do NOT forget to save this!

Create New Wallet

This password encrypts your private key. This does not act as a seed to generate your keys. You will need this password + your private key to unlock your wallet.

How to Create a Wallet • Getting Started

Already have

- o Ledger / TREZOR : Use your hardware wallet.
- o MetaMask Chrome Extension . So make sure you're not on a phishy site.
- o Jaxx / imToken : To access your wallet.
- o Mist / Geth / Parity / JSON-RPC : ETC (ethereum-commonwealth.org)

ETC (epool.io)
Ropsten (myetherapi.com)
Ropsten (infura.io)
Kovan (etherscan.io)
Kovan (infura.io)
Rinkeby (etherscan.io)
Rinkeby (infura.io)
EXP (expanse.tech)
UBQ (ubiqscan.io)
POA (core.poanetwork.io)
TOMO (core.tomocoin.io)
ELLA (ellaism.org)
ETSC (etscnode.com)

Add Custom Network / Node

MyEtherWallet.com does not hold your keys for you. We cannot access accounts, recover keys, reset passwords, nor reverse transactions. Protect your keys & always check that you are on correct URL. You are responsible for your security.

25

Import your RPC server address and the port number from Ganache to MyEtherWallet

The screenshot shows the Ganache interface on the left and the 'Set Up Your Custom Node' dialog on the right.

Ganache Interface (Top Bar):

- ACCOUNTS
- BLOCKS
- TRANSACTIONS
- LOGS

CURRENT BLOCK: 0 | GAS PRICE: 20000000000 | GAS LIMIT: 6721975 | NETWORK ID: 5777 | **RPC SERVER: HTTP://127.0.0.1:7545** | MINING STATUS: AUTOMINING

MNEMONIC:
slim rain lawn kiwi elegant behind vibrant dentist puppy re

ADDRESS:
0x231eAeEF9EA93F5370a1F633F32E45AF570980E8

ADDRESS:

Set Up Your Custom Node Dialog:

Node Name: Private ETH Node

URL: http://127.0.0.1

Port: 7545

HTTP Basic access authentication

ETH ETC Ropsten Kovan Rinkeby Custom Supports EIP-155

Cancel Save & Use Custom Node

MyEtherWallet

- Contracts tab: Deploy Contract

The screenshot shows the MyEtherWallet web application interface. At the top, there is a dark header bar with the MyEtherWallet logo, version 3.21.05, language settings (English), gas price (41 Gwei), and network selection (My Ether Node:eth (Custom)). A note below the network selection says, "The network is really full right now. Check Eth Gas Station for gas price to use." Below the header is a navigation bar with links: New Wallet, Send Ether & Tokens, Swap, Send Offline, Contracts (which is highlighted with a red oval and has an arrow pointing to it from the main title), ENS, DomainSale, Check TX Status, View Wallet Info, and Help.

The main content area features a large button with the text "Interact with Contract or Deploy Contract". The word "Deploy Contract" is circled in red. Below this button, there is a section titled "Byte Code" which contains a large empty text area. At the bottom left, there is a section titled "Gas Limit" with a value of "300000" in a text input field.

Remix

- Type your contract and compile it

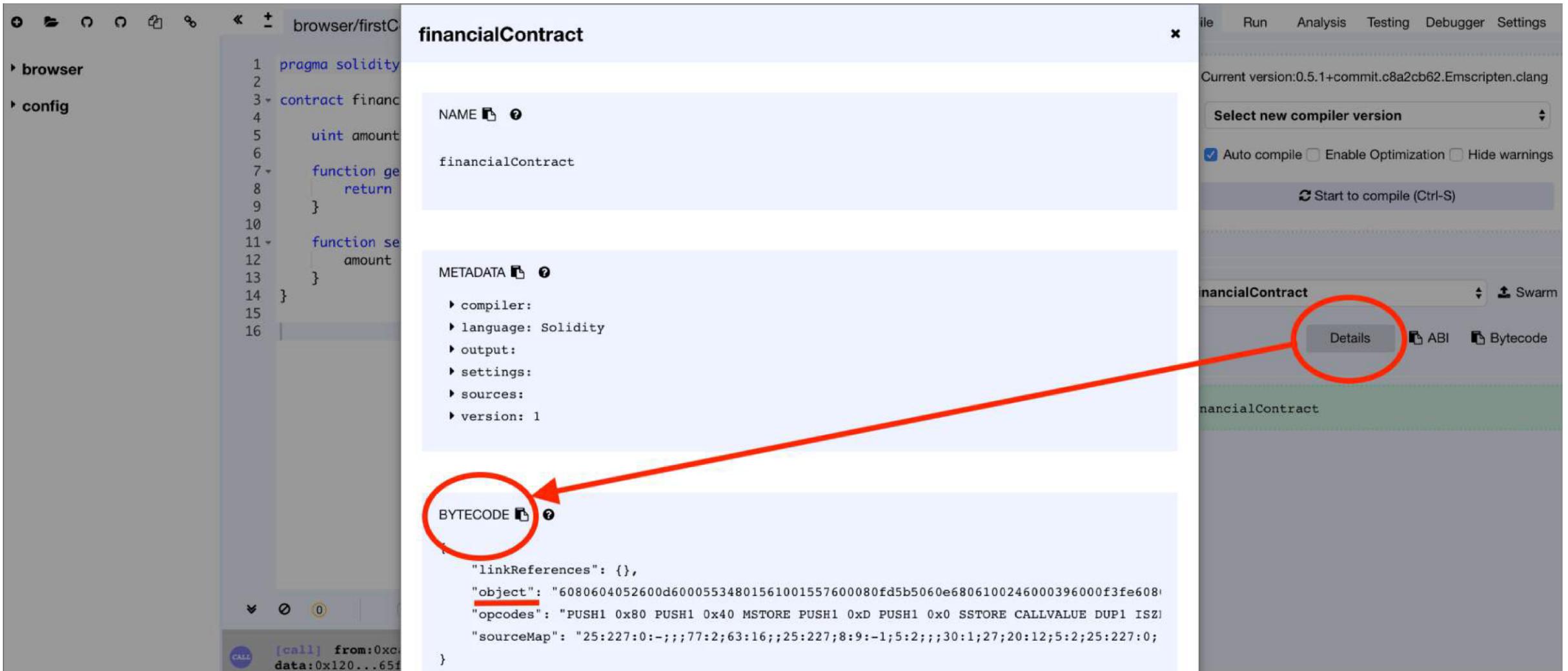
The screenshot shows the Remix IDE interface. On the left, there is a code editor window titled "browser/firstContract.sol" containing the following Solidity code:

```
1 pragma solidity ^0.5.0;
2
3 contract financialContract {
4     uint amount = 13;
5
6     function getValue() public view returns(uint){
7         return amount;
8     }
9
10    function setValue(uint newAmount) public{
11        amount = newAmount;
12    }
13 }
14
15
16
```

On the right side of the interface, there is a toolbar with various tabs: Compile, Run, Analysis, Testing, Debugger, Settings, and Swarm. Below the toolbar, there is a section labeled "Select new compiler version" with a dropdown menu showing "Current version:0.5.1+commit.c8a2cb62.Emscripten clang". Underneath this, there are three checkboxes: "Auto compile" (checked), "Enable Optimization" (unchecked), and "Hide warnings" (unchecked). A red box highlights the "Start to compile (Ctrl-S)" button, which is located in a blue bar below the checkboxes. At the bottom of the interface, there are tabs for "financialContract" (selected), "Details", "ABI", and "Bytecode".

Remix

Click on Details Button: access ByteCode to import it to MyEtherWallet



Ganache

Access your private key for signing your contract in MyEtherWallet.

The screenshot shows the Ganache application interface. At the top, there are tabs for ACCOUNTS, BLOCKS, TRANSACTIONS, and LOGS. Below the tabs, status information includes CURRENT BLOCK (0), GAS PRICE (20000000000), GAS LIMIT (6721975), NETWORK ID (5777), RPC SERVER (HTTP://127.0.0.1:7545), and MINING STATUS (AUTOMINING). A search bar and a gear icon are also present.

MNEMONIC: slim rain lawn kiwi elegant behind vibrant dentist puppy reduce kidney there

HD PATH: m/44'/60'/0'/0/account_index

ADDRESS	BALANCE	TX COUNT	INDEX
0x231eAeEF9EA93F5370a1F633F32E45AF570980E8	100.00 ETH	0	0
0x231eAeEF9EA93F5370a1F633F32E45AF570980E8	100.00 ETH	0	1
0x970fc818790E900598C		0	2
0xb59BD5568d0be42C13f		0	3
0x280AFA533B9fa1A97a6		0	4
0xD6D30E87AB17c3046AE9CAC88475ECcaRF2757c5		0	5

A modal window is open for the first account, showing the ADDRESS (0x231eAeEF9EA93F5370a1F633F32E45AF570980E8), BALANCE (100.00 ETH), and HD PATH (m/44'/60'/0'/0/account_index). The PRIVATE KEY (a53cf8cb7b66d91ca388ef9ce4e45e39997f2773247c27bb2c7cae35a1b3d383) is highlighted with a red oval and a red arrow points from it to the key icon in the header of the main table row. A 'DONE' button is at the bottom of the modal.

MyEtherWallet

1. Paste the contract's ByteCode from Remix

2. Gas Limit will automatically be calculated

3. Paste your private key from Ganache

4. Click Unlock

5. Now you have access to your wallet

MyEtherWallet

Click on *Sign Transaction* button to deploy your contract

Ganache

You can see now you have one transaction for your address and your balance has been changed because of the amount of gas you paid for creating the contract.

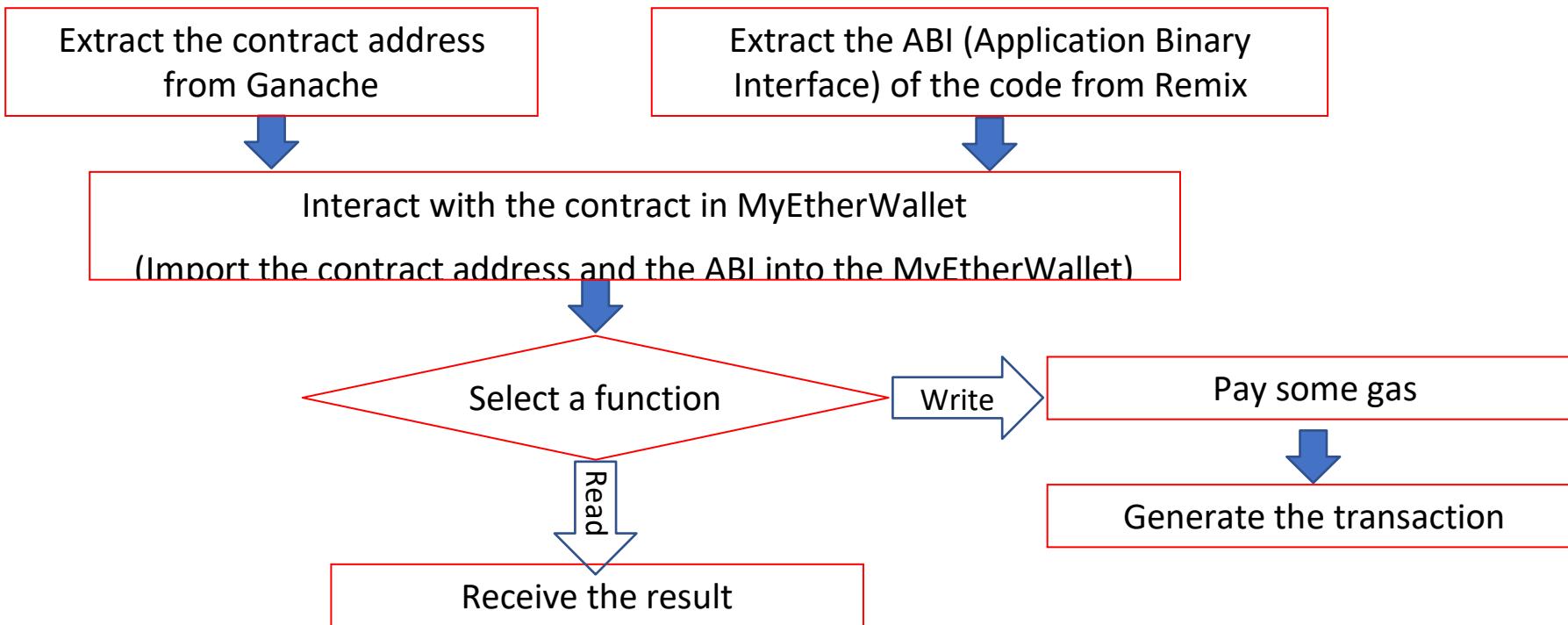
The screenshot shows the Ganache interface with the following details:

ADDRESS	BALANCE	TX COUNT	INDEX	Key
0x231eAeEF9EA93F5370a1F633F32E45AF570980E8	99.99 ETH	1	0	🔑
0x970fc818790E900598C57E48b89B6D3D8896D416	100.00 ETH	0	1	🔑
0xb59BD5568d0be42C13fB521f845243F1CDaF2eF1	100.00 ETH	0	2	🔑
0x280AFA533B9fa1A97a6D2E4640412FD86FC5dd36	100.00 ETH	0	3	🔑
0xD6D39E82AB17c30460F2CAc88425ECcaBf2757c5	100.00 ETH	0	4	🔑

Key UI elements highlighted with red circles:

- The "CURRENT BLOCK" value "1" in the top navigation bar.
- The "BALANCE" value "99.99 ETH" for the first account.
- The "TX COUNT" value "1" for the first account.

Interacting with the smart contract



Ganache

Transactions tab: Copy the created contract address

The screenshot shows the Ganache application window. At the top, there are tabs for ACCOUNTS, BLOCKS, TRANSACTIONS (which is highlighted with a red oval), and LOGS. Below the tabs, there are several status indicators: CURRENT BLOCK (1), GAS PRICE (2000000000), GAS LIMIT (6721975), NETWORK ID (5777), RPC SERVER (HTTP://127.0.0.1:7545), and MINING STATUS (AUTOMINING). A search bar at the top right allows searching for block numbers or tx hashes. The main area displays a transaction entry. The TX HASH is `0x1e40cc28802d152e810bd9f40bea83d83b1655fc9bace6e801ec6db5fcd84b1a`. The FROM ADDRESS is `0x231eAeEF9EA93F5370a1F633F32E45AF570980E8`. The CREATED CONTRACT ADDRESS is `0xf22A8cA21D7eeF564FD5Ea743dd9326197CFAA2d`. To the right of the transaction details, there are buttons for CONTRACT CREATION, GAS USED (124604), and VALUE (0).

TX HASH	CONTRACT CREATION
<code>0x1e40cc28802d152e810bd9f40bea83d83b1655fc9bace6e801ec6db5fcd84b1a</code>	
FROM ADDRESS	GAS USED
<code>0x231eAeEF9EA93F5370a1F633F32E45AF570980E8</code>	124604
CREATED CONTRACT ADDRESS	VALUE
<code>0xf22A8cA21D7eeF564FD5Ea743dd9326197CFAA2d</code>	0

Remix

Copy the ABI

(ABI is the interface that tells MyEtherWallet how to interact with the contract)

The screenshot shows the Remix IDE interface. On the left, there is a code editor window titled "browser/firstContract.sol" containing the following Solidity code:

```
1 pragma solidity ^0.5.0;
2
3 contract financialContract {
4     uint amount = 13;
5
6     function getValue() public view returns(uint){
7         return amount;
8     }
9
10    function setValue(uint newAmount) public{
11        amount = newAmount;
12    }
13 }
14 }
```

On the right, the interface includes a toolbar with "Compile", "Run", "Analysis", "Testing", "Debugger", "Settings", and "Support". Below the toolbar, there is a dropdown for "Select new compiler version" and checkboxes for "Auto compile", "Enable Optimization", and "Hide warnings". A large blue button labeled "Start to compile (Ctrl-S)" is present. Under the "financialContract" tab, there are three tabs: "Details" (selected), "ABI" (highlighted with a red circle), and "Bytecode". A green box at the bottom contains the ABI string: "financialContract [abi]".

MyEtherWallet

Contracts tab:

Interact with Contract = Paste the contract address from Ganache and the ABI from Remix

New Wallet Send Ether & Tokens Swap Send Offline **Contracts** ENS DomainSale Check TX Status View Wallet Info Help

Interact with Contract or Deploy Contract

Contract Address
0xf22A8cA21D7eeF564FD5Ea743dd9326197CFAA2d

Select Existing Contract
Select a contract...

ABI / JSON Interface

```
    "outputs": [],
    "payable": false,
    "stateMutability": "nonpayable",
    "type": "function"
}
```

Access

MyEtherWallet

You now can interact with the contract by selecting a function and invoking it

New Wallet Send Ether & Tokens Swap Send Offline Contracts ENS DomainSale Check TX Status View Wallet Info Help

Interact with Contract or Deploy Contract

Contract Address

0xf22A8cA21D7eeF564FD5Ea743dd9326197CFAA2d

Select Existing Contract

Select a contract...

ABI / JSON Interface

```
    "outputs": [],
    "payable": false,
    "stateMutability": "nonpayable",
    "type": "function"
}
```

Access

Read / Write Contract

0xf22A8cA21D7eeF564FD5Ea743dd9326197CFAA2d

Select a function ▾

getValue
SetValue

MyEtherWallet

If you select the getValue function you will receive the value without paying any gas
(There is no operation cost for getting information)

The screenshot shows the MyEtherWallet interface with the following details:

- Read / Write Contract**: The address listed is 0xf22A8cA21D7eeF564FD5Ea743dd9326197CFAA2d.
- Function Selection**: A button labeled "getValue ▾" is highlighted.
- Return Type**: The return type is indicated as "↳ uint256".
- Gas Estimate**: A large, semi-transparent gray bar at the bottom contains the number "13", representing the gas estimate for the function call.

MyEtherWallet

If you choose a function that updates the state of the contract,
you will need to pay gas for it in a transaction.

The image shows two screenshots of the MyEtherWallet interface. The left screenshot displays the 'Read / Write Contract' section for a specific Ethereum address. A red circle highlights the 'newValue uint256' input field, which contains the value '6'. An arrow points from this field to the 'WRITE' button at the bottom of the form. The right screenshot shows a 'Warning!' dialog box. It informs the user that they are about to execute a function on a contract and provides options for the transaction amount (set to 0), gas limit (set to 41633), and nonce. A red circle highlights the 'Generate Transaction' button. Another red circle highlights the 'Raw Transaction' section, which shows the JSON object: {"nonce": "0x01", "gasPrice": "0x098bc5a00", "gasLimit": "0xa2a1", "to": "0xf22A8cA21D7eeF564FD5Ea743dd9326197CFAA2d"}. Below the dialog is a message '197CFAA2d' and a 'WRITE' button.

