

A hands-on tutorial: Working with Smart Contracts in Ethereum

Was prepared with the assistance of Mohammad H. Tabatabaei from
the University of Oslo



UiO : University of Oslo

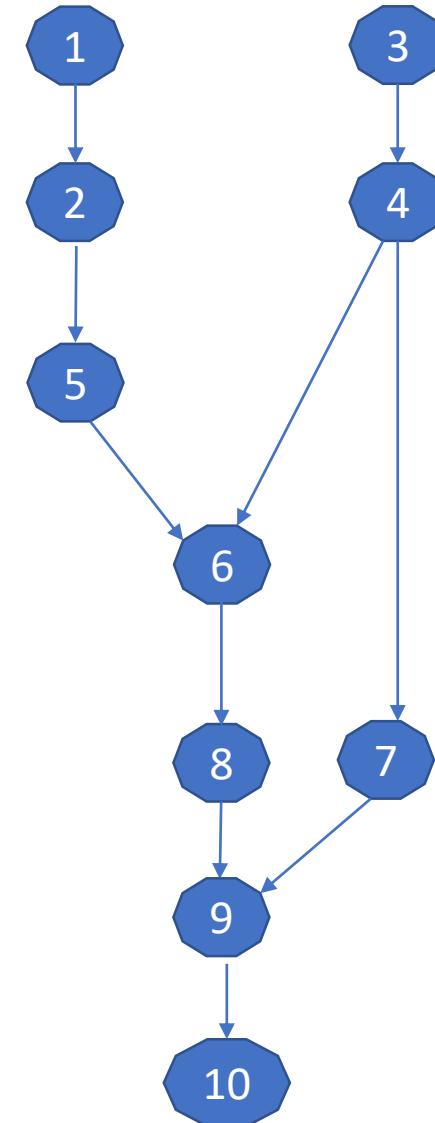
Different tools provide different functionality

	Tools	Remix	Ganache	MyEtherWallet	Geth
	Activities				
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	-	+	-	+

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

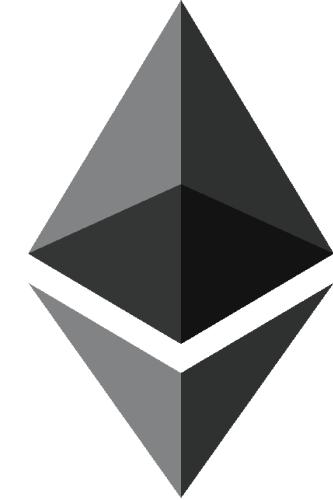
<http://truffleframework.com/ganache/>

<https://github.com/kvhnuke/etherwallet/releases/tag/v3.21.06>



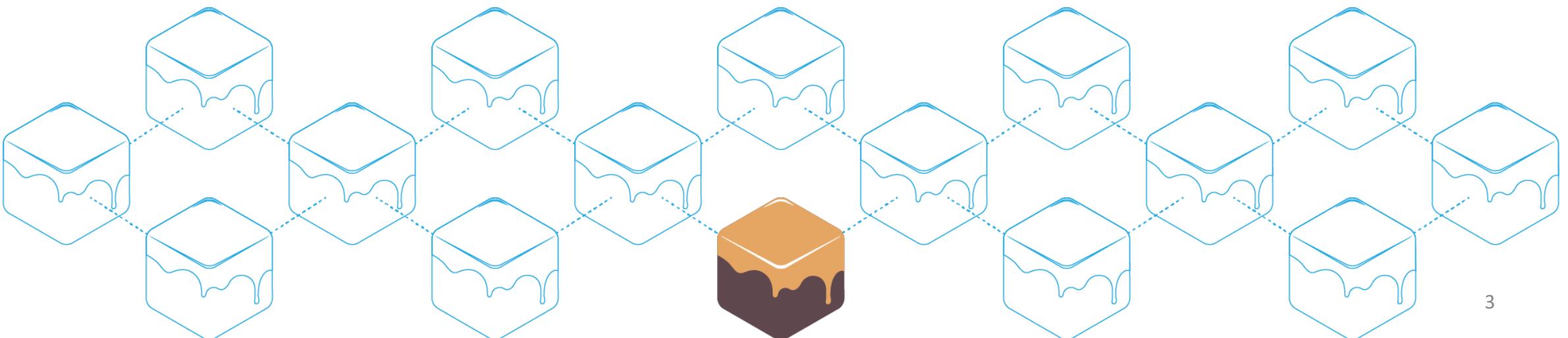
Use which tool for what purpose? (1/2)

- Use Geth for everything?
 - Powerful but command-line only
- What should I use?
 - 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) and transfer funds between user accounts

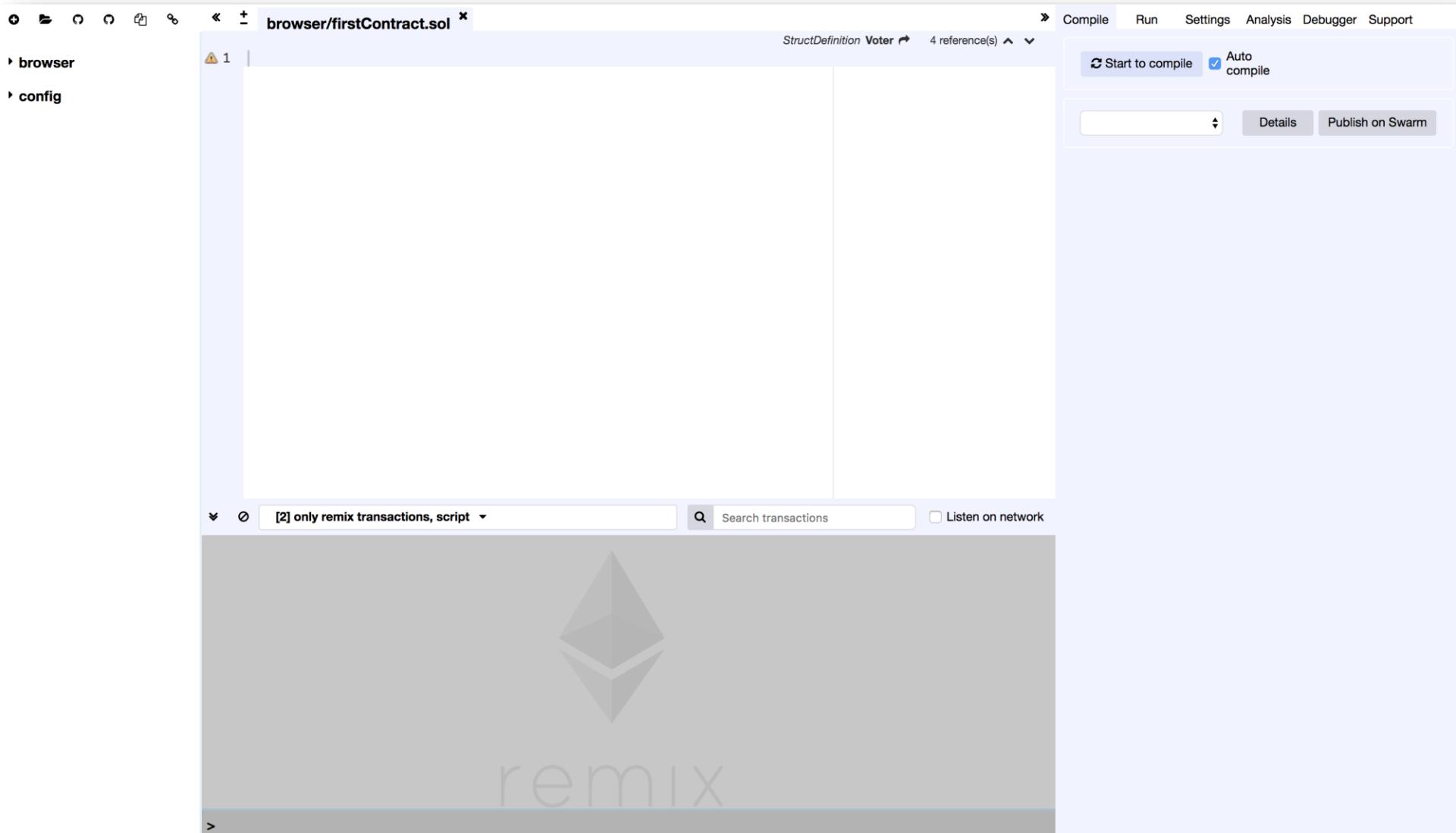


Smart Contracts

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

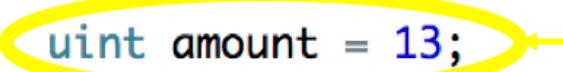
Open Remix : remix.ethereum.org

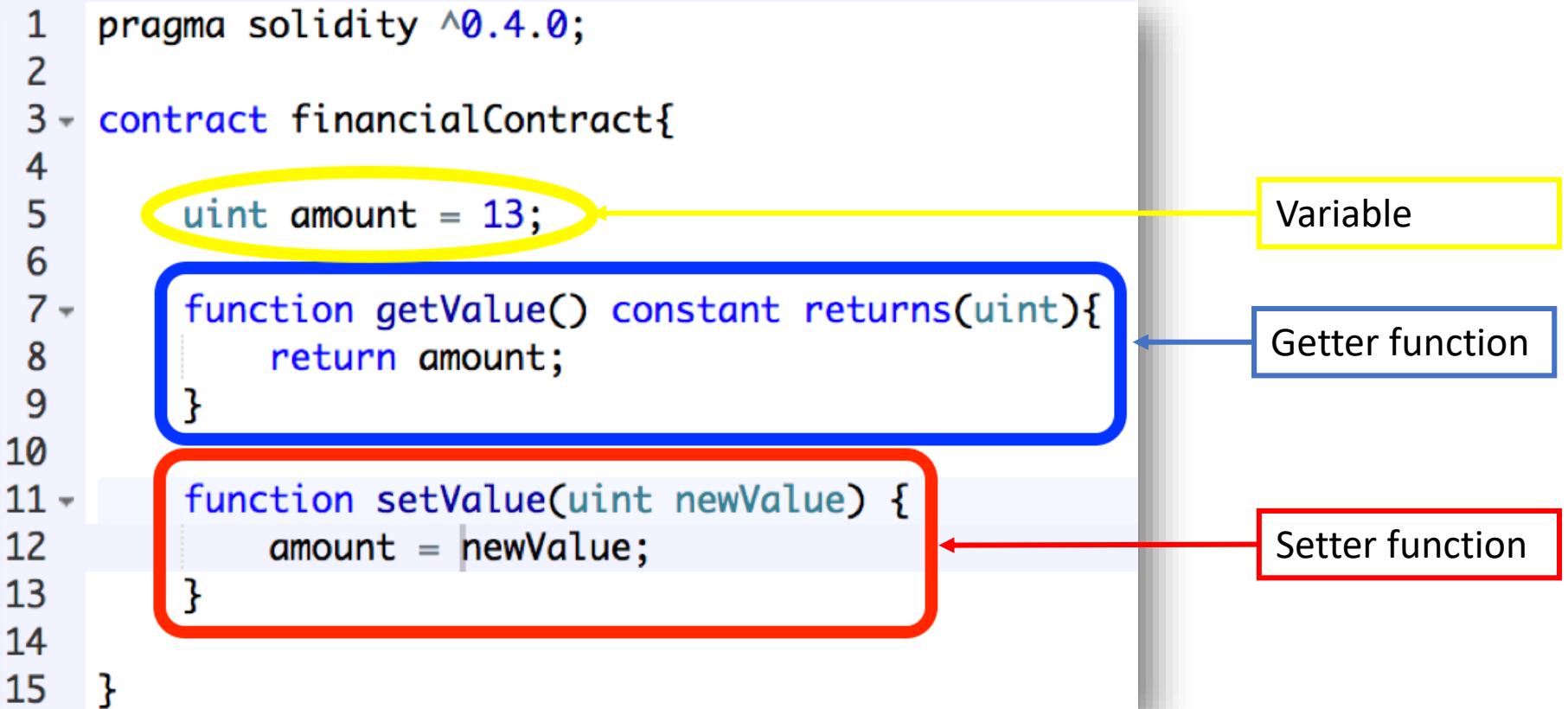
- An open source tool for writing, compiling and testing Solidity contracts



Start Coding

- Setter and Getter: Set and get the information.

```
1 pragma solidity ^0.4.0;
2
3 contract financialContract{
4
5     uint amount = 13; 
6
7     function getValue() constant returns(uint){
8         return amount;
9     }
10
11    function setValue(uint newValue) {
12        amount = newValue;
13    }
14
15 }
```



The diagram illustrates the relationship between a variable and its corresponding getter and setter functions in Solidity. A yellow oval highlights the variable declaration `uint amount = 13;`. A blue rounded rectangle encloses the `getValue()` function, which is labeled as a `Getter function`. A red rounded rectangle encloses the `setValue(uint newValue)` function, which is labeled as a `Setter function`. Arrows point from each label to its respective highlighted code element.

Compile the Contract

- Compile tab: Start to compile button

The screenshot shows the Truffle UI interface. On the left, there is a code editor window titled "browser/firstContract.sol" containing Solidity code for a contract named "financialContract". The code includes functions for getting and setting the value of "amount". On the right, the "Compile" tab is selected, showing various options and status indicators. A red box highlights the "Start to compile" button, which is currently active (indicated by a checkmark). Below it is a checkbox for "Auto compile". Other tabs visible include Run, Settings, Analysis, Debugger, and Support.

```
1 pragma solidity ^0.4.0;
2
3 contract financialContract{
4
5     uint amount = 13;
6
7     function getValue() constant returns(uint){
8         return amount;
9     }
10
11    function setValue(uint newValue) {
12        amount = newValue;
13    }
14
15 }
```

Set Environment (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 ^0.4.0;
2
3 contract financialContract{
4     uint amount = 13;
5
6     function getValue() constant returns(uint){
7         return amount;
8     }
9
10    function setValue(uint newValue) {
11        amount = newValue;
12    }
13
14 }
15 }
```

On the right, the "Run" tab is active, showing the "Environment" configuration. The "Environment" tab is highlighted with a red border. The configuration includes:

- Account: 0x147...c160c (100 ether)
- Gas limit: 3000000
- Value: 0 wei

Below the environment settings, there is a section for the contract "financialContract" with a "Deploy" button. At the bottom, it shows "0 pending transactions".

Set Environment (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 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 transactions of a contract.
- Value: The amount of value for 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: Like Ropsten, Kovan and Rinkeby which are existing public blockchains used for testing and which do not use real funds.
- Public Real: 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

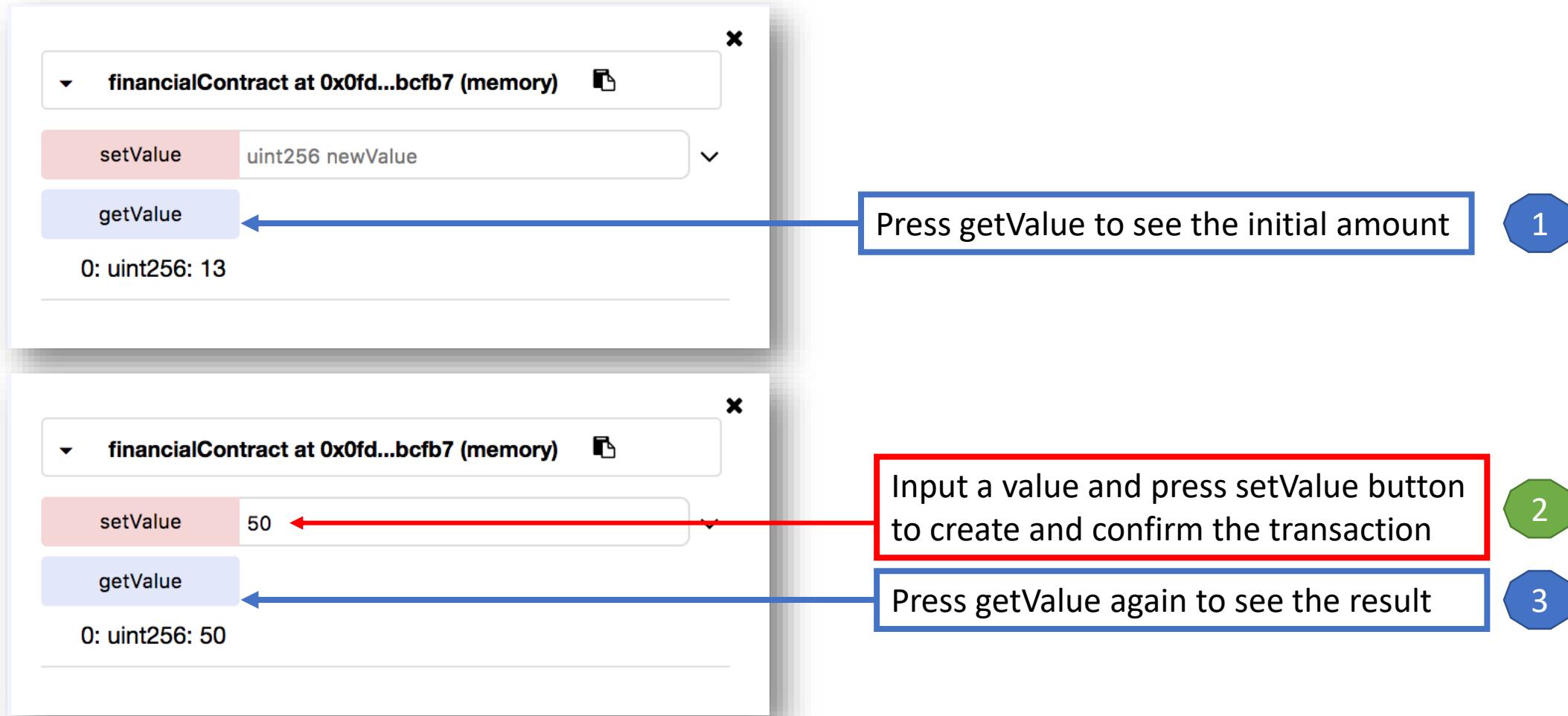
The screenshot shows the Remix IDE interface with the following details:

- Code Editor:** The file `browser/firstContract.sol` contains the following Solidity code:

```
1 pragma solidity ^0.4.0;
2
3 contract financialContract{
4     uint amount = 13;
5
6     function getValue() constant returns(uint){
7         return amount;
8     }
9
10    function setValue(uint newValue) {
11        amount = newValue;
12    }
13 }
14
15 }
```
- Compiler:** The code is compiled with the following settings:
 - Environment: JavaScript VM
 - Account: `0x147...c160c (99.9999999999998675)`
 - Gas limit: 3000000
 - Value: 0 wei
- Run Tab:** A dropdown menu for the contract `financialContract` is open, showing the **Deploy** button, which is highlighted with a red box.
- Transactions:** There are 0 pending transactions.
- Storage:** The storage at address `0x0fd...bcfb7` (memory) is shown, with the `setValue` slot containing `uint256 newValue`.
- Network:** The network status shows 2 remix transactions, and there is an option to Listen on network.

Interact with the Contract

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



Additional features

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

Constructor

- A function with the name of the contract
- Will be called at the creation of the instance of the contract

```
1 pragma solidity ^0.4.0;
2
3 contract financialContract{
4
5     uint amount;
6     address issuer;
7
8     function financialContract(){
9         issuer = msg.sender;
10    }
11
12    function getValue() constant returns(uint){
13        return amount;
14    }
15
16    function setValue(uint newValue) {
17        amount = newValue;
18    }
19
20 }
```

We want to save
the address of the
contract creator

Modifier

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

```
1 pragma solidity ^0.4.0;
2
3 contract financialContract{
4
5     uint amount;
6     address issuer;
7
8     function financialContract(){
9         issuer = msg.sender;
10    }
11
12    modifier ifIssuer(){
13        if(issuer != msg.sender){
14            throw;
15        }else{
16            -
17        }
18    }
19
20    function getValue() constant returns(uint){
21        return amount;
22    }
23
24    function setValue(uint newValue) ifIssuer {
25        amount = newValue;
26    }
27
28 }
```

Only the contract creator is permitted to set value

Receive ether (1/2)

- Transfer money to the contract

Payable keyword
allows receiving
ether

We can get the
balance of the
contract

```
1 pragma solidity ^0.4.0;
2
3 contract financialContract{
4
5     address issuer;
6
7     function financialContract(){
8         issuer = msg.sender;
9     }
10
11    modifier ifIssuer(){
12        if(issuer != msg.sender){
13            throw;
14        }else{
15            -
16        }
17    }
18
19    function receiveFunds() payable{
20
21    }
22
23    function getValue() constant returns(uint){
24        return this.balance;
25    }
26
27 }
```

Receive ether (2/2)

1

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

Environment: JavaScript VM
Account: 0x147...c160c (99.9999999999998311€)
Gas limit: 3000000
Value: 100 wei

financialContract

Deploy

2

Click the receiveFunds button to transfer the money to the contract

0 pending transactions

financialContract at 0x1df...bda71 (memory)

receiveFunds

setValue uint256 newValue

getValue

Withdraw funds

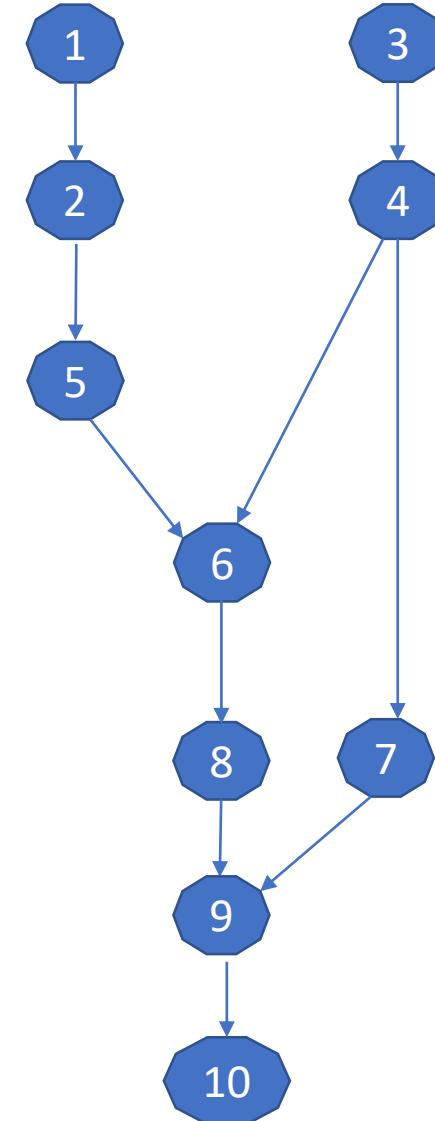
- Transfer ether from the contract to the user account

Transfer some money from the contract to the mentioned account

```
1 pragma solidity ^0.4.0;
2
3 contract financialContract{
4
5     address issuer;
6
7     function financialContract(){
8         issuer = msg.sender;
9     }
10
11    modifier ifIssuer(){
12        if(issuer != msg.sender){
13            throw;
14        }else{
15            -
16        }
17    }
18
19    function receiveFunds() payable{
20
21    }
22
23    function getValue() constant returns(uint){
24        return this.balance;
25    }
26
27    function withdrawFunds(uint funds) ifIssuer{
28        issuer.send(funds);
29    }
30
31 }
```

Now deploying a smart contract on an external blockchain

	Tools	Remix	Ganache	MyEtherWallet	Geth
	Activities				
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

The screenshot shows the Ganache interface with the following details:

- Accounts:** Three accounts are listed, each with a balance of **100.00 ETH**.
 - Address 1:** `0x231eAeEF9EA93F5370a1F633F32E45AF570980E8`
 - Address 2:** `0x970fc818790E900598C57E48b89B6D3D8896D416`
 - Address 3:** `0xb59BD5568d0be42C13fB521f845243F1CDaF2eF1`
- Mnemonic:** `slim rain lawn kiwi elegant behind vibrant dentist puppy reduce kidney there`
- HD Path:** `m/44'/60'/0'/0/account_index`
- RPC Server:** `HTTP://127.0.0.1:7545`
- Miner Status:** `AUTOMINING` (indicated by a yellow progress bar)
- Logs:** A search bar for block numbers or tx hashes.

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 form titled "Create New Wallet" with a password input field containing "Do NOT forget to save this!" and a "Create New Wallet" button. Below the button is a note: "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, a dropdown menu titled "Network ETH (myetherapi.com)" is open, listing various Ethereum networks. 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 list, 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 to access your wallet.
- o MetaMask Chrome Extension . So make sure you are not on a phishy site.
- o Jaxx / imToken : Use your hardware wallet to access your wallet.
- o Mist / Geth / Parity / (UTC / JSON) : Use your hardware wallet to access your wallet.

ETH (myetherapi.com)
ETH (etherscan.io)
ETH (infura.io)
ETH (giveth.io)
ETC (Ethereum Commonwealth)
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.poa.network)
TOMO (core.tomochain.io)
ELLA (ellaism.org)
ETSC (etsc.io)

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.](#)

21

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

The screenshot shows the Ganache application window. At the top, there are tabs for ACCOUNTS, BLOCKS, TRANSACTIONS, and LOGS. Below these are several status indicators: CURRENT BLOCK (0), GAS PRICE (20000000000), GAS LIMIT (6721975), NETWORK ID (5777), and an RPC SERVER entry (HTTP://127.0.0.1:7545). The last two are circled in red. A red arrow points from the RPC SERVER entry in Ganache to the URL input field in the 'Set Up Your Custom Node' dialog box. The dialog box also contains fields for Node Name (Private ETH Node), Port (7545), and a checkbox for HTTP Basic access authentication. Below the dialog are network selection buttons for ETH, ETC, Ropsten, Kovan, Rinkeby, Custom, and Supports EIP-155.

Ganache

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

Instructions can be found here

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 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). A message 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 circled in red), ENS, DomainSale, Check TX Status, View Wallet Info, and Help. The main content area has a large heading "Interact with Contract or Deploy Contract" with "Deploy Contract" highlighted by a red oval and an arrow pointing from the Contracts link in the nav bar. Below this, there are two input fields: "Byte Code" (empty) and "Gas Limit" (set to 300000).

3.21.05 English ▾ Gas Price: 41 Gwei ▾ Network My Ether Node:eth (Custom) ▾
The network is really full right now. Check Eth Gas Station for gas price to use.

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

Interact with Contract or **Deploy Contract**

Byte Code

Gas Limit 300000

Remix

- Type your contract and compile it

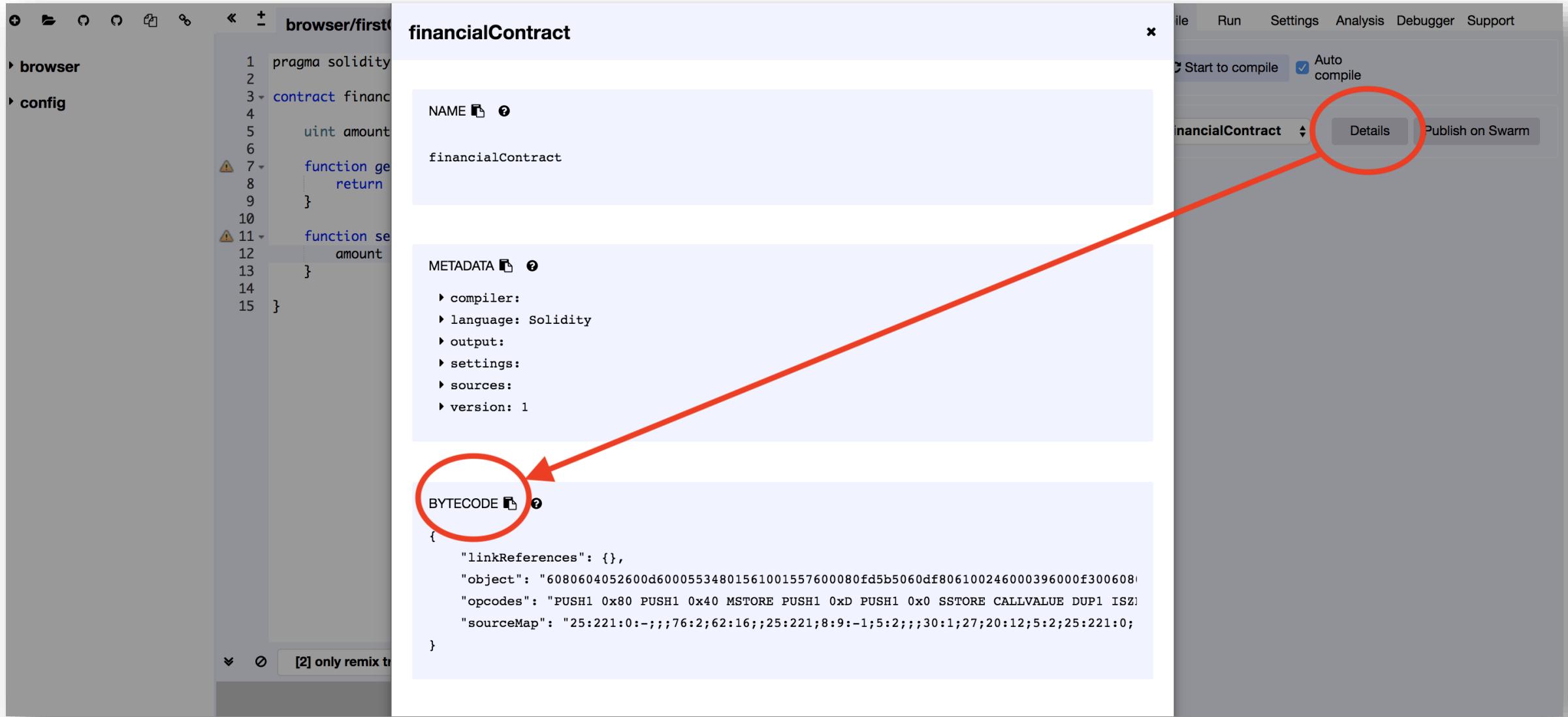
The screenshot shows the Remix browser-based IDE interface. On the left, there is a code editor window titled "browser/firstContract.sol" containing the following Solidity code:

```
1 pragma solidity ^0.4.0;
2
3 contract financialContract{
4
5     uint amount = 13;
6
7     function getValue() constant returns(uint){
8         return amount;
9     }
10
11    function setValue(uint newValue) {
12        amount = newValue;
13    }
14
15 }
```

On the right side of the interface, there is a toolbar with several buttons: "Compile", "Run", "Settings", "Analysis", "Debugger", and "Support". Below the toolbar, there is a status bar showing "uint256 newValue" and "1 reference(s)". Further down, there is a section labeled "financialContract" with a dropdown menu, a "Details" button, and a "Publish on Swarm" button. A red box highlights the "Start to compile" button, which is located next to a checked checkbox labeled "Auto compile".

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 interface with the following details:

- Accounts:** 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
- Account Details (Focused):** ADDRESS 0x231eAeEF9EA93F5370a1F633F32E45AF570980E8, BALANCE 100.00 ETH, TX COUNT 0, INDEX 0. A red arrow points from the "INDEX 0" key icon to the "INDEX 0" column of the list below.
- Account List:** A list of accounts with their addresses and balances:
 - 0x231eAeEF9EA93F5370a1F633F32E45AF570980E8 (BALANCE 100.00 ETH)
 - 0x970fc818790E900598C (BALANCE 0.00 ETH)
 - 0xb59BD5568d0be42C13f (BALANCE 0.00 ETH)
 - 0x280AFA533B9fa1A97a6 (BALANCE 0.00 ETH)
 - 0xD6D9E82AB17c30460E2CAc88425ECcaRf2757c5 (BALANCE 0.00 ETH)
- Modal Window (Bottom Left):** A modal window for the account 0x231eAeEF9EA93F5370a1F633F32E45AF570980E8 shows its address, a "PRIVATE KEY" field containing a long string of hex digits (a53cf8cb7b66d91ca388ef9ce4e45e39997f2773247c27bb2c7cae35a1b3d383), and a "DONE" button.

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.

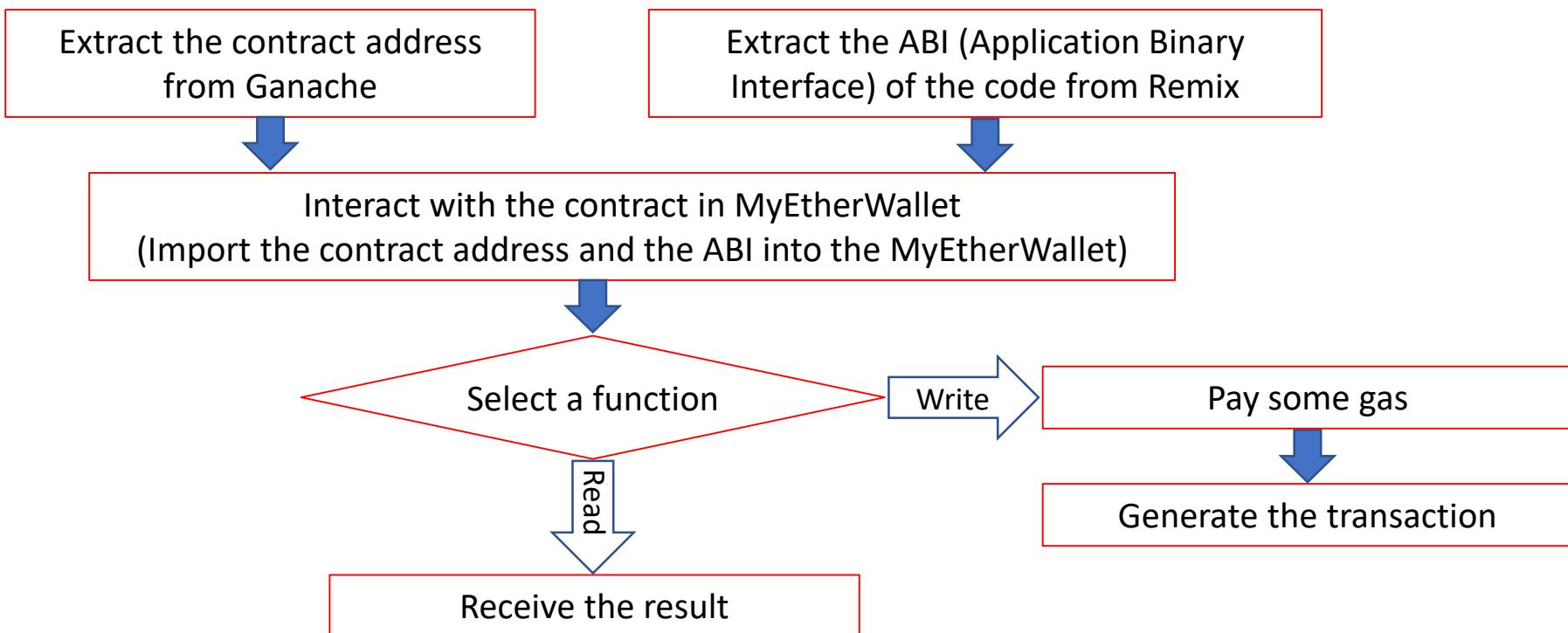
The screenshot shows the Ganache interface with the following account information:

ADDRESS	BALANCE	TX COUNT	INDEX	KEY
<code>0x231eAeEF9EA93F5370a1F633F32E45AF570980E8</code>	99.99 ETH	1	0	🔑
<code>0x970fc818790E900598C57E48b89B6D3D8896D416</code>	100.00 ETH	0	1	🔑
<code>0xb59BD5568d0be42C13fB521f845243F1CDaF2eF1</code>	100.00 ETH	0	2	🔑
<code>0x280AFA533B9fa1A97a6D2E4640412FD86FC5dd36</code>	100.00 ETH	0	3	🔑
<code>0xD6D39E82AB17c30460F2CAc88425ECcaBf2757c5</code>	100.00 ETH	0	4	🔑

Key UI elements highlighted with red circles:

- The "CURRENT BLOCK" value "1" in the top 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

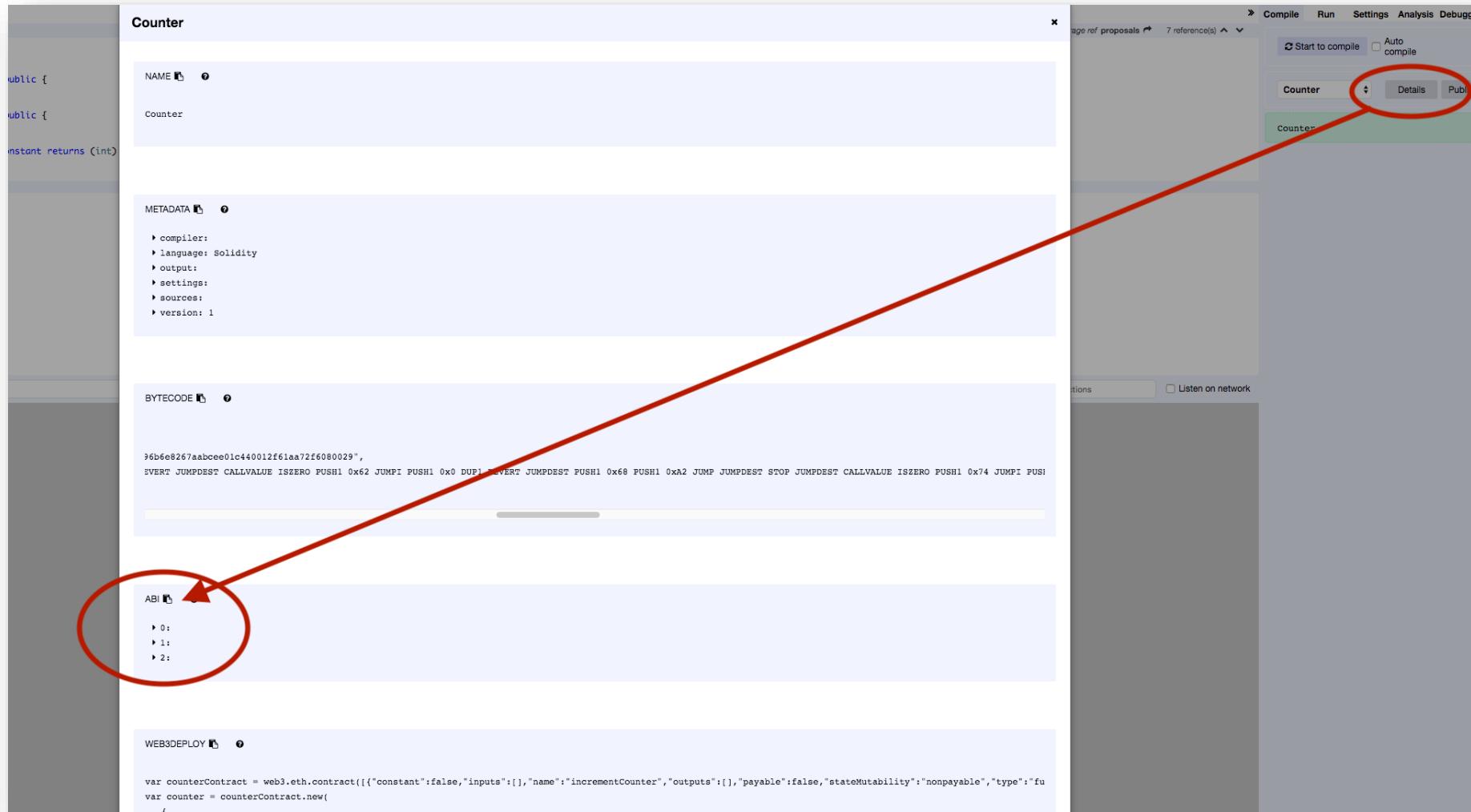
The screenshot shows the Ganache application window. At the top, there are tabs: ACCOUNTS, BLOCKS, TRANSACTIONS (which is highlighted with a red oval), and LOGS. Below the tabs, there are several status indicators: CURRENT BLOCK (1), GAS PRICE (20000000000), GAS LIMIT (6721975), NETWORK ID (5777), RPC 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. In the main area, a transaction is listed with the following details:

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

A red arrow points from the 'TRANSACTIONS' tab to the 'CREATED CONTRACT ADDRESS' field.

Remix

Click on Details button: Copy the ABI
(ABI is the interface that tells MyEtherWallet how to interact with the contract)



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 is 0xf22A8cA21D7eeF564FD5Ea743dd9326197CFAA2d.
- Function Selection**: A button labeled "getValue ▾" is highlighted.
- Return Type**: The return type is indicated as ↳ uint256.
- Value**: The value returned is 13.

MyEtherWallet

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

Read / Write Contract

0xf22A8cA21D7eeF564FD5Ea743dd9326197CFAA2d

setValue ▾

newValue uint256
6

WRITE

Warning!

You are about to execute a function on contract.
It will be deployed on the following network: ETH (Custom).

Amount to Send *In most cases you should leave this as 0.*

0

Gas Limit

41633

Generate Transaction

Raw Transaction

```
{"nonce": "0x01", "gasPrice": "0x098bca5a00", "gasLimit": "0xa2a1", "to": "0xf22A8cA21D7eeF564FD5Ea743d"}  
197CFAA2d
```

Signed Transaction

```
0xf8680185098bca5a0082a2a194f22a8ca21d7eeef564fd5ea743dd9326197cf  
aa2d80845b34b96626a04285bd52ad31
```

No, get me out of here! Yes, I am sure! Make transaction.

WRITE

MyEtherWallet

Now if you try `getValue` function again, you will see the change.

Interact with Contract or Deploy Contract

Contract Address

Select Existing Contract  Select a contract...

ABI / JSON Interface

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

Access

Read / Write Contract

0xf22A8cA21D7eeF564FD5Ea743dd9326197CFAA2d

getValue ▾

↳ uint256

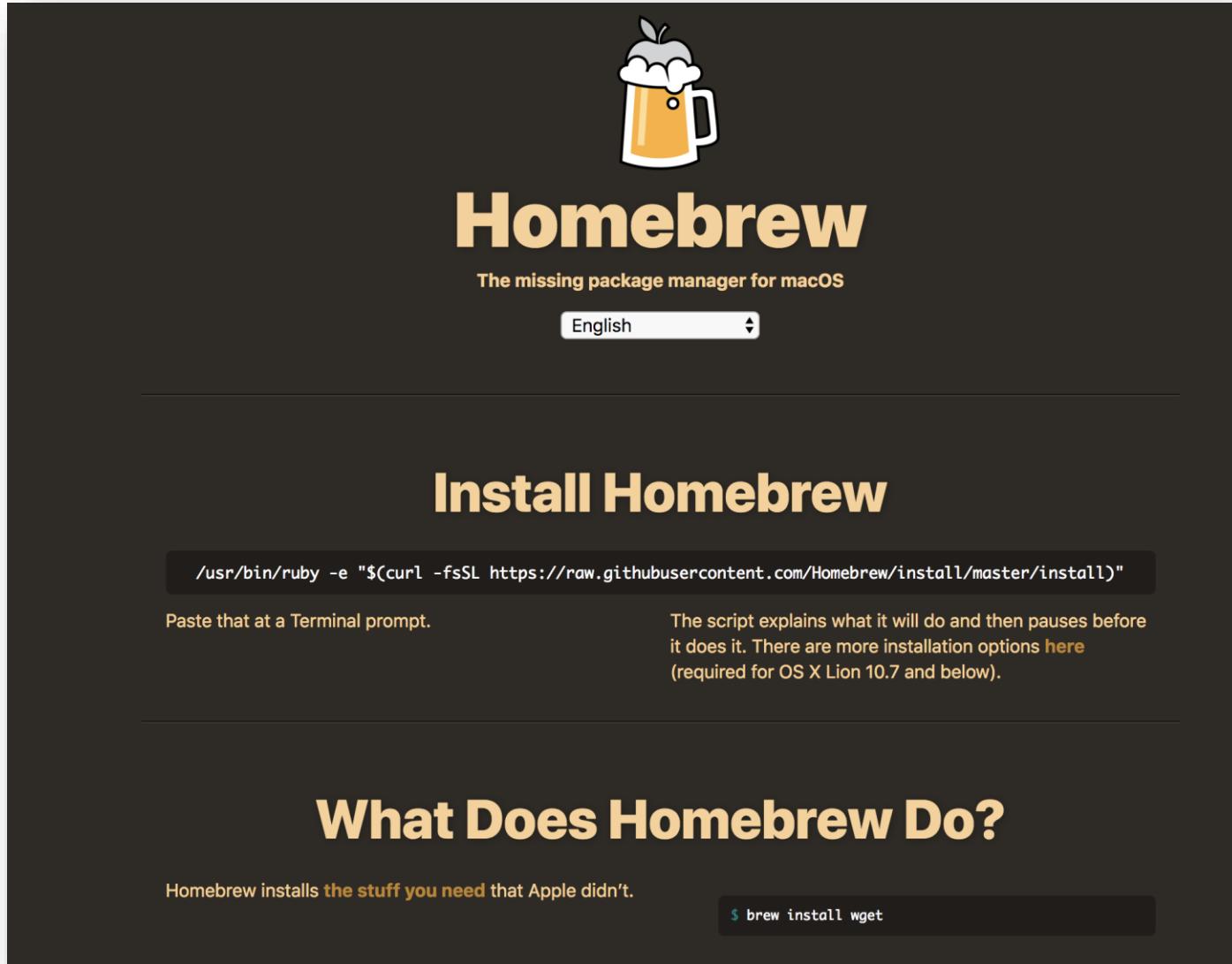
6

Create your own Ethereum Blockchain

- Instead of using Ganache with its default properties for private blockchain you can run your own blockchain
- Install Geth: One of the implementations of Ethereum written in Go
- Create the genesis block
- Create storage of the blockchain
- Deploy blockchain nodes
- Connect MyEtherWallet to your blockchain to interact with it

Homebrew (package manager for mac)

- Install homebrew with the command from its website: <https://brew.sh/>



Geth

- An Ethereum program written in Go

1

```
mohammht — -bash — 80x24
Last login: Wed May 30 10:38:04 on ttys001
ds-install:~ mohammht$ brew tap ethereum/ethereum
```

2

```
ds-install:~ mohammht$ brew install ethereum
```

Geth help

```
[ds-install:~ mohammht$ geth help
NAME:
  geth - the go-ethereum command line interface

  Copyright 2013-2018 The go-ethereum Authors

USAGE:
  geth [options] command [command options] [arguments...]

VERSION:
  1.8.9-stable

COMMANDS:
  account          Manage accounts
  attach           Start an interactive JavaScript environment (connect to node)
  bug              opens a window to report a bug on the geth repo
  console          Start an interactive JavaScript environment
  copydb           Create a local chain from a target chaindata folder
  dump             Dump a specific block from storage
  dumpconfig       Show configuration values
  export            Export blockchain into file
  export-preimages Export the preimage database into an RLP stream
  import            Import a blockchain file
  import-preimages Import the preimage database from an RLP stream
  init              Bootstrap and initialize a new genesis block
  js                Execute the specified JavaScript files
  license           Display license information
  makecache         Generate ethash verification cache (for testing)
  makedag           Generate ethash mining DAG (for testing)
  monitor           Monitor and visualize node metrics
  removedb          Remove blockchain and state databases
  version           Print version numbers
  wallet            Manage Ethereum presale wallets
  help, h           Shows a list of commands or help for one command

ETHEREUM OPTIONS:
  --config value    TOML configuration file
  --datadir "/Users/mohammht/Library/Ethereum" Data directory for the databases and keystore
  --keystore        Directory for the keystore (default = inside the
  datadir)
```

Genesis block

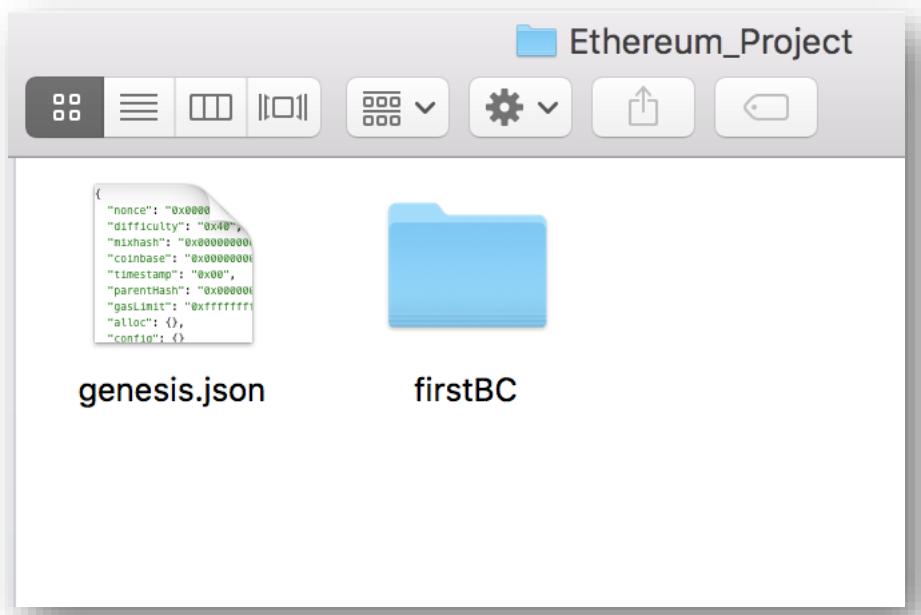
- The first block in the chain and a json file that stores the configuration of the chain

- Create and store the file as genesis.json

Create the storage of the blockchain

- Go to the directory of the genesis.json file
- Specify directory of your blockchain
- Create the storage from the genesis block

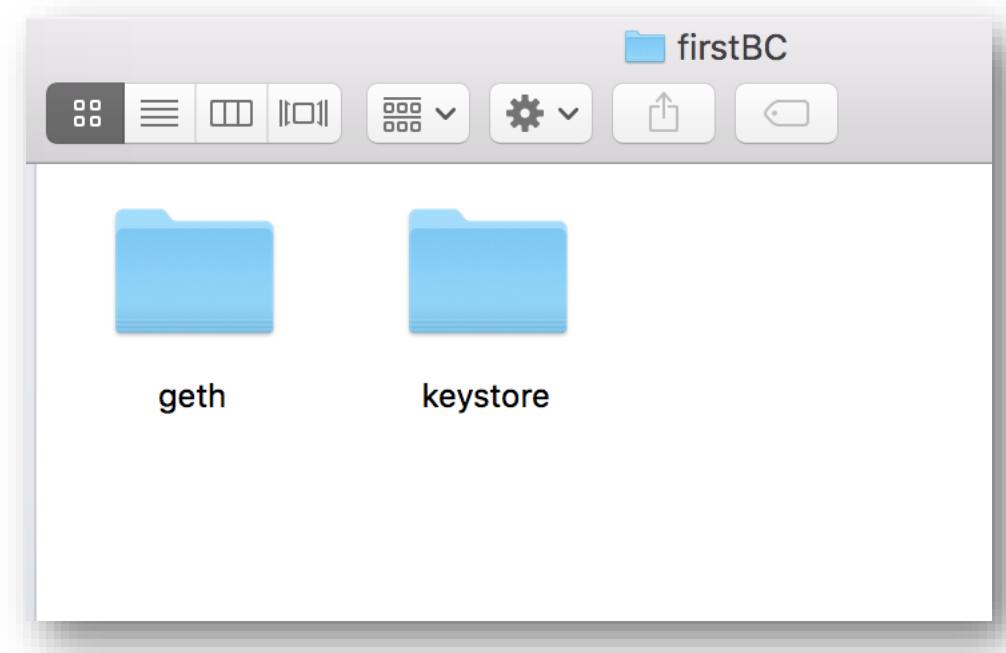
```
[ds-install:Documents mohammht$ cd Ethereum_Project/  
ds-install:Ethereum_Project mohammht$ geth --datadir firstBC init genesis.json]
```



Folder name of your
blockchain

Inside the Blockchain Folder

- **geth folder:** Store your database
- **keystore:** Store your Ethereum accounts



Start the Ethereum peer node

- Start the blockchain

```
geth --datadir fistBC --networkid 100 console
```

- Networkid provides privacy for your network.
- Other peers joining your network must use the same networkid.

Blockchain started

- Type
admin.nodeInfo
to get the
information
about your
current node

```
[> admin.nodeInfo
{
  enode: "enode://4561ccdd7fdf3f0bdbc903b7bef7d472e136fe2b63012151a1dd3c27e52f49bda2ef66631e67022b7ca7b9fba06bb0eda8b47210b198f3eeff7e67414d695ed6@[::]:30303",
  id: "4561ccdd7fdf3f0bdbc903b7bef7d472e136fe2b63012151a1dd3c27e52f49bda2ef66631e67022b7ca7b9fba06bb0eda8b47210b198f3eeff7e67414d695ed6",
  ip: "::",
  listenAddr: "[::]:30303",
  name: "Geth/v1.8.9-stable/darwin-amd64/go1.10.2",
  ports: {
    discovery: 30303,
    listener: 30303
  },
  protocols: {
    eth: {
      config: {
        byzantiumBlock: 4370000,
        chainId: 1,
        daoForkBlock: 1920000,
        daoForkSupport: true,
        eip150Block: 2463000,
        eip150Hash: "0x2086799aeebeae135c246c65021c82b4e15a2c451340993aacfd2751886514f0",
        eip155Block: 2675000,
        eip158Block: 2675000,
        ethash: {},
        homesteadBlock: 1150000
      },
      difficulty: 17179869184,
      genesis: "0xd4e56740f876aef8c010b86a40d5f56745a118d0906a34e69aec8c0db1cb8fa3",
      head: "0xd4e56740f876aef8c010b86a40d5f56745a118d0906a34e69aec8c0db1cb8fa3",
      network: 100
    }
  }
}
>
```

Create an account

- Type *personal.newAccount* to create as many accounts as you need

```
[> personal.newAccount('Type your password here')
  "0xa78eb41a10f096d4d8c4c9ca5196427aaa3fdb33"
> ]
```

- See the created account(s)

```
> eth.accounts
[ "0xa78eb41a10f096d4d8c4c9ca5196427aaa3fdb33", "0x354d952e40fc35a47562d479c86e41f6623e5f8c"]
>
```

Mining

- Type *miner.start()* to start mining

```
[> miner.start()
INFO [05-30|12:07:54] Updated mining threads          threads=0
INFO [05-30|12:07:54] Transaction pool price threshold updated price=180000000000
null
> INFO [05-30|12:07:54] Starting mining operation
INFO [05-30|12:07:54] Commit new mining work          number=1 txs=0 uncles=0 elapsed=22
8.827µs
INFO [05-30|12:07:57] Generating DAG in progress    epoch=1 percentage=0 elapsed=2.013
s
INFO [05-30|12:07:59] Generating DAG in progress    epoch=1 percentage=1 elapsed=4.151
s
INFO [05-30|12:08:03] Generating DAG in progress    epoch=1 percentage=2 elapsed=7.322
s
INFO [05-30|12:08:06] Generating DAG in progress    epoch=1 percentage=3 elapsed=10.70
5s
INFO [05-30|12:08:09] Generating DAG in progress    epoch=1 percentage=4 elapsed=14.04
3s
INFO [05-30|12:08:13] Generating DAG in progress    epoch=1 percentage=5 elapsed=17.56
5s
INFO [05-30|12:08:16] Generating DAG in progress    epoch=1 percentage=6 elapsed=20.99
9s
INFO [05-30|12:08:20] Generating DAG in progress    epoch=1 percentage=7 elapsed=24.40
9s
-
```

Thank you