

# **Blockchain e Smart contract per giuristi 3.0**

Biagio Distefano

# Ethereum



Nella Blockchain Ethereum, oltre a operazioni lineari come “Da Tizio a Caio 10 Ether”, è possibile eseguire operazioni complesse, condizionali, che costituiscono la c.d. business logic

Queste operazioni complesse vengono eseguite da tutti i nodi (peer) della rete nella propria Ethereum Virtual Machine (EVM), e vengono validate e inserite nella blockchain (circa) come le transazioni normali

# Smart Contracts



Le istruzioni delle operazioni complesse sono contenute in  
**portafogli speciali**

I portafogli speciali che contengono questa logica sono detti  
***Smart Contracts***

# Smart Contracts



Gli **Smart Contract** vengono scritti in un linguaggio di programmazione che viene tradotto in istruzioni per la **EVM**

Una volta scritto e compilato lo **Smart Contract**, questo viene immesso nella **Blockchain** (cd. *deployment*) e gli verrà assegnato un suo indirizzo portafogli

Conoscendone l'indirizzo, è possibile interagire con lo **Smart Contract** attraverso il proprio portafogli

# Smart Contracts



Scriveremo il nostro **Smart Contract** in un linguaggio di programmazione chiamato **Solidity**

Per scrivere lo smart contract useremo un ambiente di sviluppo integrato (integrated development environment, IDE) chiamato **Remix** direttamente sul browser

**Andate su <http://remix.ethereum.org/>**

# Trovate le risorse su

[https://github.com/raptored01/  
siena-2019-bc-sc](https://github.com/raptored01/siena-2019-bc-sc)

# Remix

The image shows the Remix IDE interface with several annotations in red text and arrows:

- Chiudete questo** (Close this) with an arrow pointing to the close button (X) in the top-left corner of the editor window.
- Create un nuovo file "Compravendita.sol"** (Create a new file "Compravendita.sol") with an arrow pointing to the "browser" folder in the left sidebar.
- Copiate e incollate il codice che trovate nelle risorse sotto "contratti", "Compravendita.sol"** (Copy and paste the code you find in the resources under "contracts", "Compravendita.sol") with an arrow pointing to the "Contratti" (Contracts) tab in the right sidebar.

The main editor window displays the following Solidity code:

```
1 pragma solidity ^0.5.1;
2
3 contract Ballot {
4
5     struct Voter {
6         uint weight;
7         bool voted;
8         uint8 vote;
9         address delegate;
10    }
11
12    struct Proposal {
13        uint voteCount; // could add other data about proposal
14    }
15
16    address chairperson;
17    mapping(address => Voter) voters;
18    Proposal[] proposals;
19
20    /// Create a new ballot with $(_numProposals) different proposals.
21    constructor (uint8 _numProposals) public {
```

The right sidebar shows the "Contratti" (Contracts) tab with the following information:

- Current version: 0.5.1+commit.c8a2cb62.Emscripten.clang
- Select new compiler version (dropdown)
- Auto compile ☐ Enable Optimization ☐ Hide warnings ☐
- Start to compile (Ctrl-S) button
- Swarm button
- Details, ABI, and Bytecode buttons

The bottom terminal window displays the following text:

```
remix.exeCurrent(): Run the script currently displayed in the editor
remix.help(): Display this help message
remix.debugHelp(): Display help message for debugging

- Welcome to Remix v0.7.6 -

You can use this terminal for:

• Checking transactions details and start debugging.
• Running JavaScript scripts. The following libraries are accessible:

  o web3 version 1.0.0
  o ethers.js
  o swarmgw
  o 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.

>
```

# Remix

**Selezionate la versione 0.5.7+commit.6da8b019** →

**Compile il codice** →

**Tutto OK  
(potete ignorare il warning)**

```
1 pragma solidity ^0.5.7;
2
3
4 contract Compravendita {
5
6     address payable public proprietario;
7     string public targa = "ETH2019SI";
8     uint public prezzo = 0;
9     uint constant oneEther = 1 ether;
10    bool public inVendita = false;
11    event Acquisto(address compratore, address venditore, uint prezzo);
12
13    constructor () public {
14        proprietario = msg.sender;
15    }
16
17    function mettiInVendita(uint _prezzo) public {
18        require(msg.sender == proprietario, "Solo il proprietario può mettere in vendita");
19        prezzo = _prezzo * oneEther;
20        inVendita = true;
21    }
22
23    function toglidallaVendita(uint256) public {
24        require(msg.sender == proprietario, "Solo il proprietario può togliere dalla vendita");
25        inVendita = false;
26    }
27
28    function acquista() public payable {
29        require(inVendita == true, "Non in vendita");
30        require(msg.sender != proprietario, "Solo chi non è proprietario può acquistare");
31        require(msg.value == prezzo, "Prezzo errato");
32        proprietario.transfer(msg.value);
33        address venditore = proprietario;
34        proprietario = msg.sender;
35        inVendita = false;
36        emit Acquisto(msg.sender, venditore, msg.value);
37    }
38
39 }
```

Current version: 0.5.7+commit.6da8b019.Emscripten.clang

Select new compiler version

Auto compile Enable Optimization Hide warnings

Start to compile (Ctrl-S)

Compravendita Swarm

Details ABI Bytecode

Static Analysis raised 6 warning(s) that requires your attention. Click here to show the warning(s).

Compravendita

[2] only remix transactions, script

Search transactions



# Remix

**Scegliete il portafogli con cui effettuare le operazioni**

**Immettete il contratto nella blockchain**

**Chiamate le funzioni del contratto**

**Vedete i log del contratto**

**Interrogate le variabili pubbliche del contratto**

```
contract Compravendita {
    address payable public proprietario;
    string public targa = "ETH2019SI";
    uint public prezzo = 0;
    uint constant oneEther = 1 ether;
    bool public inVendita = false;
    event Acquisto(address compratore, address venditore, uint prezzo);

    constructor () public {
        proprietario = msg.sender;
    }

    function mettiInVendita(uint _prezzo) public {
        require(msg.sender == proprietario, "Solo il proprietario può mettere in vendita");
        prezzo = _prezzo * oneEther;
        inVendita = true;
    }

    function toglidallaVendita(uint256) public {
        require(msg.sender == proprietario, "Solo il proprietario può togliere dalla vendita");
        inVendita = false;
    }

    function acquista() public payable {
        require(inVendita == true, "Non in vendita");
    }
}
```

Environment: JavaScript VM

Account: 0xca3...a733c (99.9999999999999310)

Gas limit: 3000000

Value: 0 wei

Deploy

At Address: Load contract from Address

Transactions recorded: 1

Deployed Contracts

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

acquista	
mettiInVendita	uint256 _prezzo
togliDallaVendita	uint256

inVendita

prezzo

proprietario

targa

creation of Compravendita pending...

[vm] from:0xca3...a733c to:Compravendita.(constructor) value:0 wei  
data:0x608...90029 logs:0 hash:0x2d3...b1fb2

# Usiamo uno Smart Contract su una Blockchain reale (Ethereum Ropsten Testnet)

Una versione smart del PRA

Il contratto si trova all'indirizzo

0xD9c453dC11773866e4f89b65A34164ACfb4C2dab

(cliccate sull'indirizzo per vederlo su etherscan)

L'applicazione web si trova qui: <http://smartsiena.ddns.net>