

# Ethereum - Solidity



ing. Pallaro Fabio - [f.pallaro@synclab.it](mailto:f.pallaro@synclab.it)

# Cosa non vi racconterò?

1. Criptovalute
2. Algoritmi di Hash, funzioni crittografiche ecc.
3. Storia Ethereum
4. Fasi di compilazione SmartContract (solc ecc.)



Di che cosa parleremo?

## **SmartContract**

Uno Smart Contract ci permette di scambiare valuta o modificare lo stato di entità presenti in catena in modo trasparente, senza conflitti evitando servizi di terze parti.



# Caratteristiche SmartContract

- Immutabile (il codice non lo stato)
- Irrevocabile (non posso cancellarlo)
- Incorruttibile (non posso modificarlo)
- Deterministiche (stesso output a parità di input) (EVM macchina a stati)
- Terminabile (lo SC deve concludersi in un tempo limitato)
- Isolato (utilizzo di 'sandbox', i.e. EVM o Docker in altri casi)



# Smart Contract: Cosa posso/non posso fare?

Posso:

- Far eseguire computazioni ai peer della rete (computer distribuito)
- Mantenere la persistenza dei dati (ledger distribuito)
- Trasferire soldi (ether) da un wallet all'altro (anche contract)

Non posso:

- Interagire con qualcosa fuori della mia catena (a meno di oracoli oppure il nuovo protocollo Polkadot molto interessante)
- Schedulare qualcosa che si ripeta periodicamente



# Linguaggi diversi, concetti simili

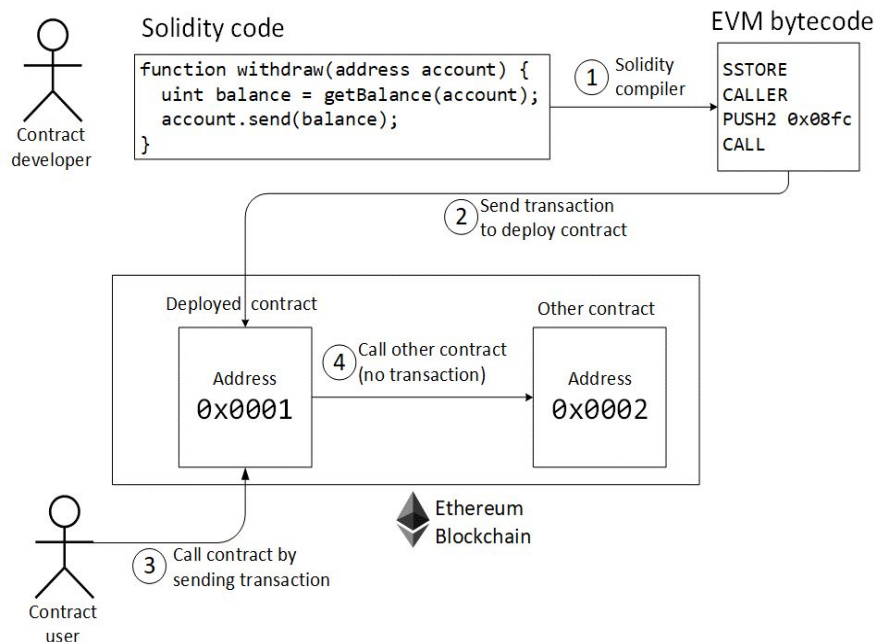
- Solidity ~ Java ~ C++
- Ethereum blockchain ~ RAM ~ Hard Drive
- Ethereum Virtual Machine (EVM) ~ Java Virtual Machine (JVM) ~ x86 CPU
- Contract bytecode ~ Java Bytecode ~ x86 bytecode
- Solidity compilers (solc) ~ gcc ~ gcc-c++ ~ javac
- Contracts ~ Classes
- Deployed contracts ~ Objects
- Deployed contract address ~ memory address ~ file inode



# Flusso di 'deploy' dello SmartContract

Di solito scritti con linguaggi di alto-livello e compilati in bytecode.

Deployati sui peer connessi alla rete.



# Flusso di esecuzione SmartContract

(nel caso di modifica di stato)

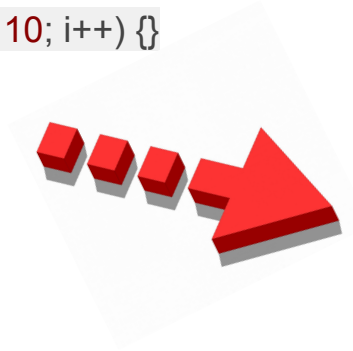
- L'utente invia una transazione all'indirizzo dello SC
- La transazione viene inviata (broadcasting) ai nodi (come tutte le altre transazioni)
- Il miner esegue l'SC e se si conclude con successo viene computato un nuovo stato (con l'aggiornamento delle proprietà)





# Misura del costo di calcolo

```
for (uint i = 0; i < 10; i++) {}
```




```
PUSH1 0x00  
PUSH1 0x00  
MSTORE      ; store 0 at position 0 in memory (loop counter)  
JUMPDEST    ; set a place to jump (PC = 6)  
PUSH1 0x0a   ; push 10 on the stack  
PUSH1 0x00  
MLOAD       ; load loop counter  
PUSH1 0x01  
ADD         ; increment loop counter  
DUP1  
PUSH1 0x00  
MSTORE      ; store updated loop counter  
LT          ; check if loop counter is less than 10  
PUSH1 0x06  
JUMPI       ; jump to position 6 if true
```

# Misurazione Costo di Calcolo

Ogni operazione disponibile  
(OPCODE)  
richiede un consumo di gas

PUSH1 0x00 ; 3 gas  
PUSH1 0x00 ; 3 gas  
MSTORE ; 3 gas  
JUMPDEST ; 1 gas  
PUSH1 0x0a ; 3 gas  
PUSH1 0x00 ; 3 gas  
MLOAD ; 3 gas  
PUSH1 0x01 ; 3 gas  
**ADD** ; 3 gas  
DUP1 ; 3 gas  
PUSH1 0x00 ; 3 gas  
MSTORE ; 3 gas  
LT ; 3 gas  
PUSH1 0x06 ; 3 gas  
JUMPI ; 10 gas



Total 410 gas: 10 for first 4 instructions, then 40 x 10

# Misurazione Costo di Calcolo



## **GAS**

Grandezza dell'effort computazionale  
necessario per eseguire le operazioni richieste

## **ETHER**

Unità di misura che dà il prezzo ai GAS

1 ETHER = 1,000,000,000,000,000,000 WEI = 1 (EXA)WEI

1 (MILLI)ETHER = 0.001 ETHER = 1,000,000,000,000,000 WEI = 1 (PETA)WEI

1 (MICRO)ETHER = 0.000001 ETHER = 1,000,000,000,000 WEI = 1 (TERA)WEI

1 (NANO)ETHER = 0.000000001 ETHER = 1,000,000,000 WEI = 1 (GIGA)WEI

1 (PICO)ETHER = 0.000000000001 ETHER = 1,000,000 WEI = 1 (MEGA)WEI

1 (FEMTO)ETHER = 0.0000000000000001 ETHER = 1,000 WEI = 1 (KILO)WEI

1 (ATTO)ETHER = 0.0000000000000000001 ETHER = 1 WEI

# Costo totale

Ogni transazione ha un costo base di 21000 Gas (costo per la gestione minima dello SC) più tutti i costi necessari per le operazioni che servono per interagire con il contratto.

Il costo viene calcolato a seconda del Gas Price, ovvero da quanto si è disposti a pagare per Gas.

Il Gas viene solitamente espresso in Gwei, che corrisponde ad un milionesimo di Ether. Il costo di un trasferimento viene quindi calcolato così:  $21000 * \text{Gas Price}$ .



# Flusso di mining

- Il sender setta un GasPrice (prezzo che il sender è disposto a spendere per 1 Gas) ed un GasLimit (costo massimo che il sender è disposto a pagare)

(GasPrice minimo (Safe low): al momento di 4,6 Gwei)

I minatori sono incentivati ad accettare le transazioni con GasPrice più alto

- Al termine del mining il sender paga il costo esatto.

Se non è stato speso tutto il budget iniziale previsto, il gas in più viene ritornato al sender.

- Se fallisce la transazione, il gas usato non viene restituito
- Se la computazione va oltre al gas prestabilito (gasLimit) il contratto viene ucciso (terminabilità dello SC) (out of gas)



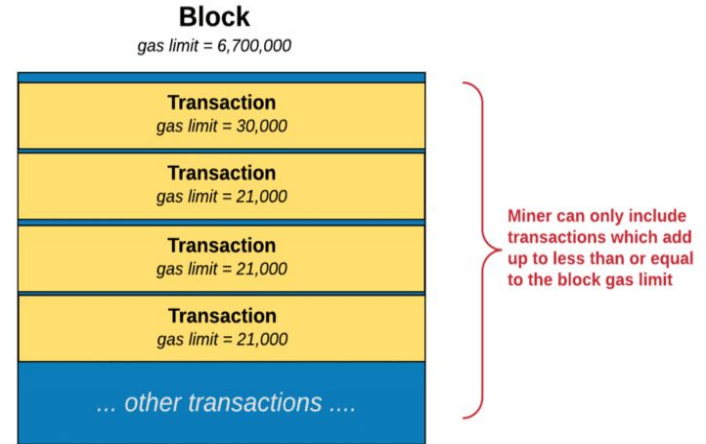
# Esempio Distributore benzina

- Arrivo al distributore ed imposto i 50 € di importo
- Inizio il pieno di benzina
- Se ci stanno tutti (e magari non ho fatto il pieno) me ne vado
- Se a 40 € raggiungo il pieno del serbatoio il distributore dovrebbe restituire i 10 € di differenza
- In Ethereum quando raggiungo il gas Limit si ferma e ritorna errore, oppure se ne uso meno ritorna il resto.



# GasLimit/GasPrice/GasBlock

- Ogni blocco da minare ha una dimensione
- I minatori aggiungono solo transazioni che ci stiano nel blocco



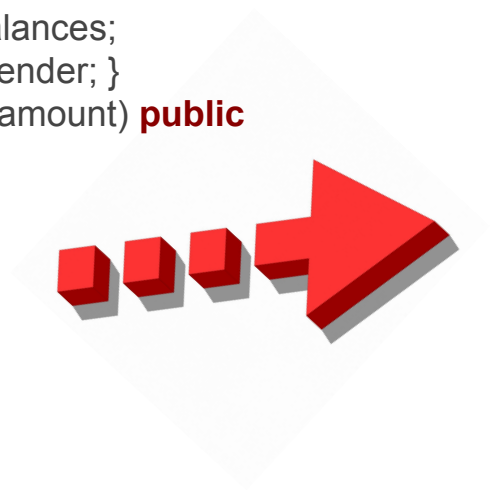
Più alto imposto il gas limit e meno probabilità ho che la mia transazione venga inserita nel nuovo blocco.

Convienne sempre impostare il gas limit poco sopra allo stimato.



# Codice ByteCode EVM / Solidity

```
contract Coin {  
  address public minter;  
  mapping (address => uint) public balances;  
  constructor() public { minter = msg.sender; }  
  function mint(address receiver, uint amount) public  
  {  
    require(msg.sender == minter);  
    require(amount < 1e60);  
    balances[receiver] += amount;  
  }  
}
```



PUSH1 0x00

PUSH1 0x00

MSTORE

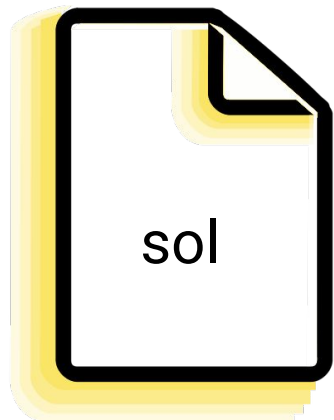
JUMPDEST

PUSH1 0x0a

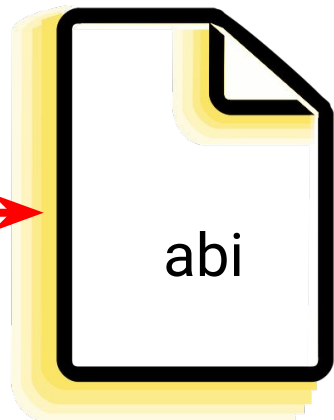
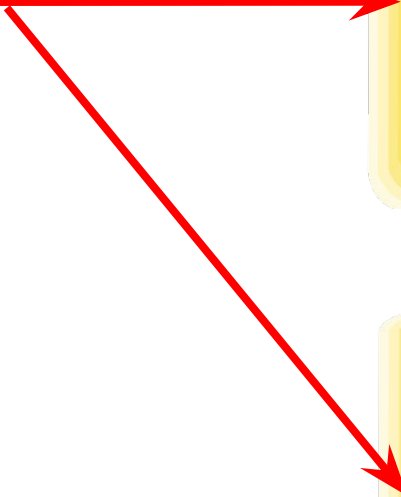
PUSH1 0x00

MLOAD

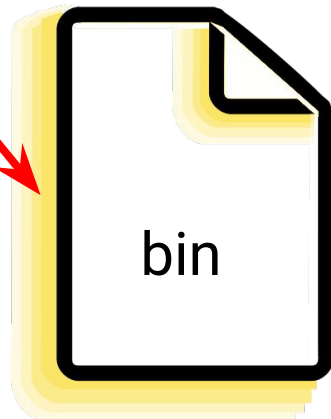
PUSH1 0x01



solc



Application  
Binary  
Interface



Bytecode/E  
VM code



# Solidity

```
contract ConfigInterface {
    address public owner;
    mapping(address => bool) admins;
    mapping(bytes32 => address) addressMap;
    mapping(bytes32 => bool) boolMap;
    mapping(bytes32 => bytes32) bytesMap;
    mapping(bytes32 => uint256) uintMap;

    /// @notice setConfigAddress sets configuration '_key' to '_val'
    /// @param _key The key name of the configuration.
    /// @param _val The value of the configuration.
    /// @return Whether the configuration setting was successful or not.
    function setConfigAddress(bytes32 _key, address _val) returns(bool success);

    /// @notice setConfigBool sets configuration '_key' to '_val'
    /// @param _key The key name of the configuration.
    /// @param _val The value of the configuration.
    /// @return Whether the configuration setting was successful or not.
    function setConfigBool(bytes32 _key, bool _val) returns(bool success);

    /// @notice setConfigBytes sets configuration '_key' to '_val'
    /// @param _key The key name of the configuration.
    /// @param _val The value of the configuration.
    /// @return Whether the configuration setting was successful or not.
    function setConfigBytes(bytes32 _key, bytes32 _val) returns(bool success);

    /// @notice setConfigUint '_key' to '_val'
    /// @param _key The key name of the configuration.
    /// @param _val The value of the configuration.
    /// @return Whether the configuration setting was successful or not.
    function setConfigUint(bytes32 _key, uint256 _val) returns(bool success);

    /// @notice getConfigAddress gets configuration '_key's value
    /// @param _key The key name of the configuration.
    /// @return The configuration value
    function getConfigAddress(bytes32 _key) returns(address val);

    /// @notice getConfigBool gets configuration '_key's value
    /// @param _key The key name of the configuration.
    /// @return The configuration value
    function getConfigBool(bytes32 _key) returns(bool val);

    /// @notice getConfigBytes gets configuration '_key's value
    /// @param _key The key name of the configuration.
```

Bytecode

ABI

```
00152903152949599478C00E1E3E0EC/0500147/147/0100473000314C07/Da...
610341600435600254600090600160a060020a03900116339091161461058f57
060020a0383166000908152602081905260408120548290100015906101a657f
54829010155b00156101b25750600082115b1561043057600160a060020a0383
004356024355b60006103848383610338565b61034160043560243560025460
6004356024355b8181018290101561011f565b610341600435600160a060020a
4600160a060020a031681565b61034160043560243533600160a060020a0316
5b156103985733600160a060020a03166000908152602081905260409020546
65b61034160025460f7740180000000000000000000000000000000000000000
81526001602090815260400083209385168352929052205461011f565b610341
00190f35b60400051600160a060020a03929092168252519081900360200190f
02565b5080820361011f565b5060005b61011f565b33600160a060020a03908f
29083906102c9565b600160a060020a03848116600081815260208181526040
9b60c2b068fc378daa952ba7f163c4a11628f55a4df523b3ef9291829003019d
11660090815260208190526040008220939093559086168152205461046a908
832094909455600181528382703390931602529190915220546104a59083610
320380616045282529182902094909455005180815290519287169391927fdd1
e7929181908390918190a3506001610434565b60035461051d90836182c9565
54390836102c9565b600160a060020a03841660008181526020819052604000
```

```
{
  "constant": false,
  "inputs": [
    {
      "name": "_spender",
      "type": "address"
    },
    {
      "name": "_value",
      "type": "uint256"
    }
  ],
  "name": "approve",
  "outputs": [
    {
      "name": "success",
      "type": "bool"
    }
  ]
}
```

actual contract code of Singapore-based DigixDAO

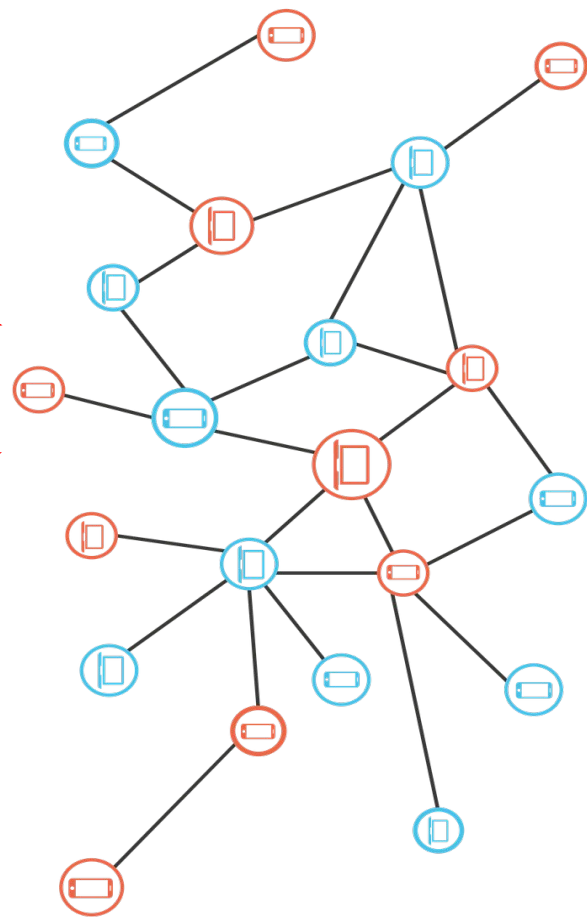


abi



bin

geth















**REMIX !!!**





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



FILE EXPLORERS

▼ browser

ballot\_test.sol

ballot.sol

test.sol

SmartContract\_BC.sol

▼ swarm

40ad61437e1322c0074650782db52e4b5b01bce02740a86318ec32e135221765

c4dfbd4641f3d8dde8593000fb2287c5b3179e68bcb5c7cacebdfb869369df7d

AttoSC.sol

SmartContract\_BC.sol

40ad61437e1322c0074650782db52e4b5b01bce02740a86318ec32e13522176

```
1 pragma solidity <=0.4.19;
2 contract SmartContract_BC{
3     //uint private count = 0;
4     //mapping (uint => contract) smartcontract; //coppia chiave-valore
5     struct PersonalData{
6         uint id;
7         string name;
8         string surname;
9         string gender;
10        string birthplace;
11        string cityofResidence;
12        string address;
13        string taxpayerCode;
14        string mail;
15        string cellphone;
16        string phone;
17        string cryptocurrency;
18        address hashwalletcryptocurrency;
19    }
20    PersonalData[] personalData;
21    uint private personalDataCount = 0;
22    //uint private personalDataIdcount = 0;
23    struct InputData1{
24        bool accepted;
25        string date;
26        string description;
27    }
28    InputData1 inputData1;
29    struct OutputData1{
30        string contractstream;
31    }
32    OutputData1 outputData1;
33    struct InputData2{
34        string description;
35        string cryptotype;
36        bytes32 buyerPrivateKey;
37        address buyersWallet;
38        string buyerName;
39        string amount;
40    }
41    InputData2 inputData2;
42    struct OutputData2{
43        string date;
44        string status;
45    }
46    OutputData2 outputData2;
47    struct InputData3{
48        bytes32 image;
49        string description;
50        string value;
51        string serialNumber;
52        string oldOwner;
53        string newOwner;
54    }
```

0

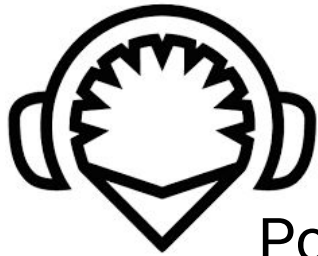
☐ listen on network

Search with transaction hash or address

- Welcome to Remix v0.10.1 -

You can use this terminal for:

- Checking Transactions details and start debugging.
- Running JavaScript scripts. The following libraries are accessible:
  - web3 version 1.0.0
  - ethers.js
  - swarmgw
  - remix (run remix.help() for more info)
- Executing common command to interact with the Remix interface (see list of commands above). Note that these
- Use exports.register(key, obj) to register and reuse object across script executions.



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

Possibilità di connessione:

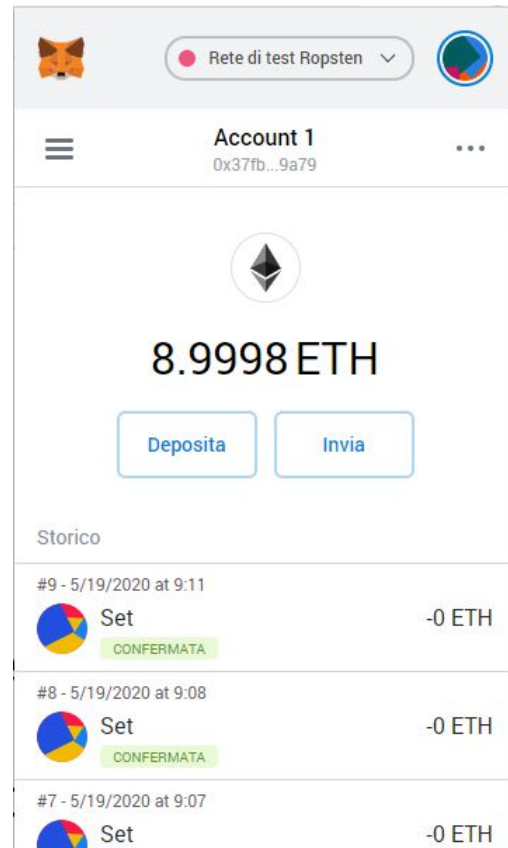
- Simulatore locale Javascript
- Connessione tramite Metamask o simili
- Connessioni a reti pubbliche (tramite client geth o similari)





# Metamask

portafoglio di criptovaluta che ti consente di interagire con applicazioni decentralizzate basate su Ethereum direttamente dal tuo browser.





# Ethereum 2 (Serenity)

- Passaggio da POW a POS (x motivi di consumo energetico e velocità)
- Maggiore sicurezza e scalabilità



# Forging (o minting) nella POS (Proof of Stake)

Occorre dimostrare di **essere in possesso di criptovalute** per poter partecipare al sorteggio e aver diritto di verificare una transazione. Creano i blocchi chi possiede più criptovaluta e dal tempo che la detengono.

Altri «validatori» possono poi attestare di aver visto tale blocco. Quando ci sono abbastanza attestazioni, è possibile aggiungere un blocco alla blockchain. I validatori vengono quindi premiati per la proposta di blocco andata a buon fine.



# Scalabilità Ethereum 2.0

Ethereum 1.0 -> circa 30 transazioni al secondo

Ethereum 2.0 -> circa 100.000 transazioni al secondo

Questo è possibile usando le **shard chains** (catene secondarie su blockchain che affiancheranno la beacon chain principale) che aumenteranno la capacità del network e miglioreranno la velocità delle transazioni estendendo la rete a 64 blockchain.



# Sicurezza Ethereum 2.0

La maggior parte dei network in proof of stake hanno un piccolo gruppo di validatori, Ethereum 2.0 richiede invece la presenza di un minimo di 16.384 validatori, il che rende il tutto molto più decentralizzato e quindi sicuro.



# Compiti per casa

- Installare MetaMask
- Racimolare kether (sufficiente 1KE)  
(tramite Faucet) per la rete test Kovan  
(serve login su github)
- Prendere Confidenza con Remix



# Solidity



# Caratteristiche Solidity

- Linguaggio Orientato agli oggetti
- Forte somiglianza con JavaScript
- Tipizzato
- Supporta ereditarietà multipla



# Hello World

```
Home | HelloWorld.sol x | ballot.sol
1 // SPDX-License-Identifier: GPL-3.0
2 pragma solidity >=0.4.16 <0.7.0;
3
4 contract SimpleStorage {
5     uint storedData;
6
7     function set(uint x) public {
8         storedData = x;
9     }
10
11     function get() public view returns (uint) {
12         return storedData;
13     }
14 }
```



# Parti del codice

```
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 }
```

Variable

Getter function

Setter function

# Funzioni

E' accessibile solo dall'esterno

E' accessibile solo da questo SC o SC ereditati

**function** pippo (<parameter types>)

Tutti possono accedervi

{**public**|**external**|**internal**|**private**}

E' accessibile solo da questo SC

[**pure**|**view**|**payable**]

[**returns** (<return types>)]

{

}

La funzione non accederà mai allo stato delle variabili

La funzione non altererà in nessun modo lo stato delle variabili

La funzione deve ricevere ether altrimenti la transazione fallirà

# Data Location

Solo per funzioni  
external (accessibili  
solo dall'esterno)

la variabile rimarrà  
in un'area  
temporanea

```
function pippo (#type  
memory|storage|calldata #name)  
  
{public|external|internal|private  
  
[pure|view|payable]  
  
[returns (<return types>)]  
  
{  
  
}
```

la variabile viene  
considerata  
persistente

# Commenti nel codice

- single line:

//

- multiline:

/\* \*/

- Ethereum Natural Specification:

/// oppure /\*\* \*/

```
pragma solidity ^0.5.6;

/// @title A simulator for trees
/// @author Larry A. Gardner
/// @notice You can use this contract for only the most basic simulation
/// @dev All function calls are currently implemented without side effects
contract Tree {
    /// @author Mary A. Botanist
    /// @notice Calculate tree age in years, rounded up, for live trees
    /// @dev The Alexandr N. Tetearing algorithm could increase precision
    /// @param rings The number of rings from dendrochronological sample
    /// @return age in years, rounded up for partial years
    function age(uint256 rings) external pure returns (uint256) {
        return rings + 1;
    }
}
```

# DataType

```
bool boolean_var = false; // default false

int8(..256) int_var = -60313; // default 0 - Esiste anche int, alias for int256

uint8(..256) uint_var = 60313; // default 0 - Esiste anche uint, alias for uint256

address address_var = msg.sender; // default 0x0000000000000000000000000000000000000000

bytes1(..32) bytes_var = "a"; // default "" - Esiste byte (byte1) e bytes (byte32)

string str_var = "GeeksforGeeks"; // default ""

enum enum_var { pop, jazz, rock, rave}
```



# Strutture

```
struct Studente {  
    string name;  
    uint matricola;  
}  
  
Studente studente1;  
  
Studente studente2 = Studente ("Fabio Pallaro", 100000);
```



# Array e Mapping


```
uint[6] data1;

uint[] storage myArray;

myArray.push(1);      // add an element to the array
myArray.push(3);      // add another element to the array
myArray[0];           // get the element at key 0 or first element in the array
myArray[0] = 20;       // update the first element in the array

mapping(address => uint) balances;

balances[msg.sender] = 100;           //assign a value;
balances[msg.sender]                 //read a value;
balances[unknown_key]                 //will return the default value of the type, i.e 0
```



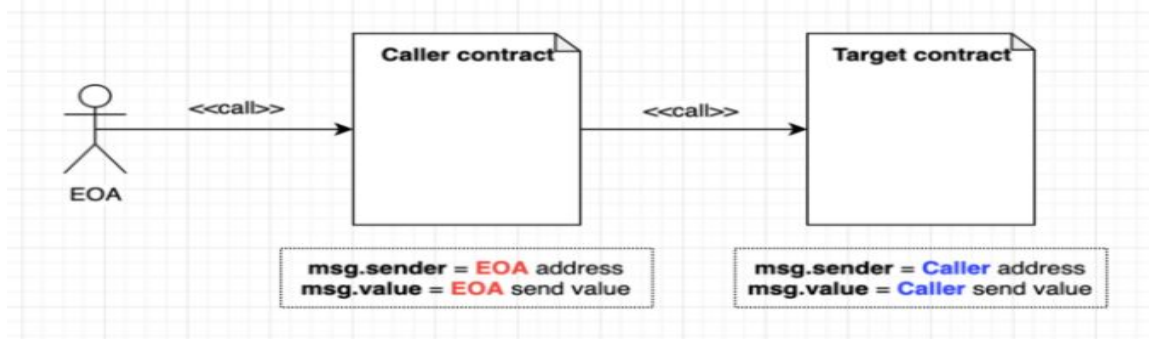
# Chiamate ad altri SmartContract

```
(bool success, bytes memory data) =  
addressMaster.call(abi.encodeWithSignature("fire(uint8)",1));  
  
require(success);  
  
(bool test, uint8 returnCode, string memory tmp)= abi.decode( data, (bool, uint8, string));  
  
return (test, returnCode, tmp);
```

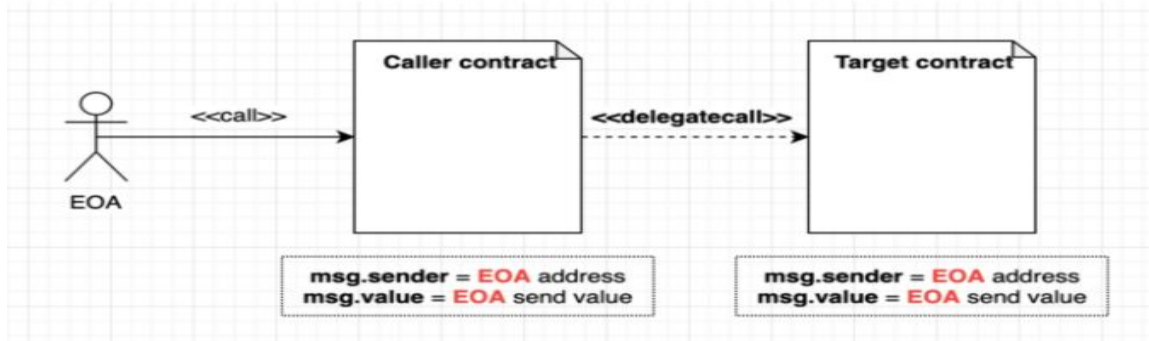




# Chiamate ad altri SmartContract



Context when the contract calls another contract



Context when contract delegatecall another contract

# Compiti per casa

- Installare MetaMask
- Racimolare kether (sufficiente 1KE)  
(tramite Faucet) per la rete test Kovan  
(serve login su github)
- Prendere Confidenza con Remix



Fine!  
Grazie per l'attenzione.



ing. Pallaro Fabio - [f.pallaro@synclab.it](mailto:f.pallaro@synclab.it)

**La fine non è altro che un nuovo inizio.**