BLOCKCHAIN: SMART CONTRACTS LECTURE 4 - **SMART CONTRACTS. ADVANCED TOPICS IN SOLIDITY.**

FLORIN CRACIUN

IMPORTANT

Some of the following slides are the property of

Dr. Emanuel Onica & Dr. Andrei Arusoaie

Faculty of Computer Science,

Alexandru Ioan Cuza University of Iași

and are used with their consent.

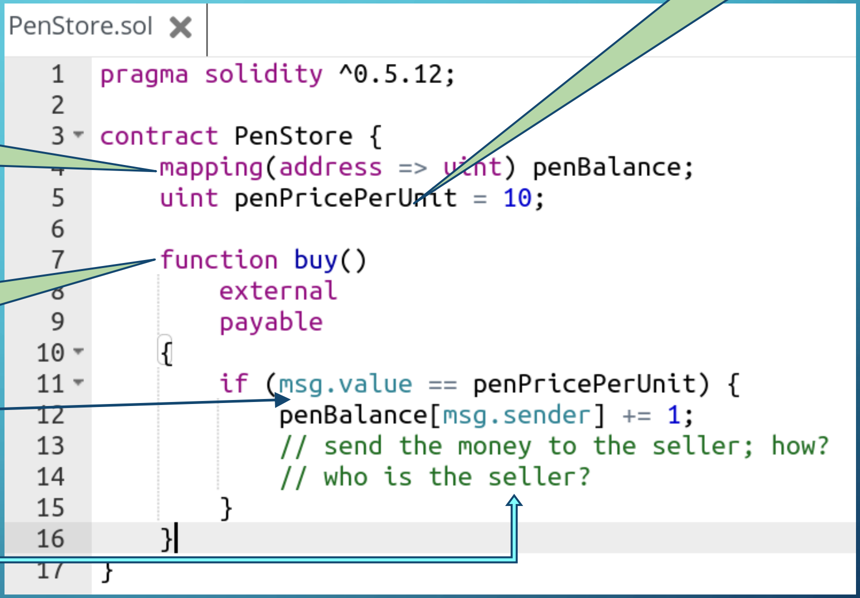
IN OUR LAST LECTURE WE LEARNED …

- … what smart contracts are
- … Solidity:
  - Basic syntax
  - Basic types
  - Visibility and access modifiers
  - Simple contracts

CONTENTS

1. `payable`
2. Constructors
3. Events
4. `require`
5. Custom modifiers
6. Inheritance
7. (some) Ethereum Request for Comment - ERC

A SIMPLE PEN STORE

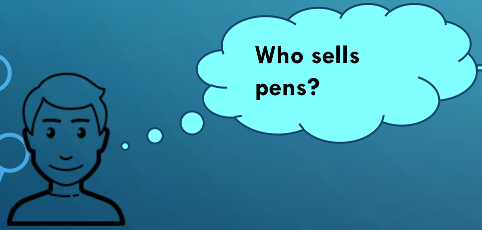


```
pragma solidity ^0.5.12;

contract PenStore {
    mapping(address => uint) penBalance;
    uint penPricePerUnit = 10;

    function buy()
        external
        payable
    {
        if (msg.value == penPricePerUnit) {
            penBalance[msg.sender] += 1;
            // send the money to the seller; how?
            // who is the seller?
        }
    }
}
```

Price per unit

Each user (address) has a corresponding basket of products

The function is `external` and `payable`. Each function that handles `msg.value` is payable.



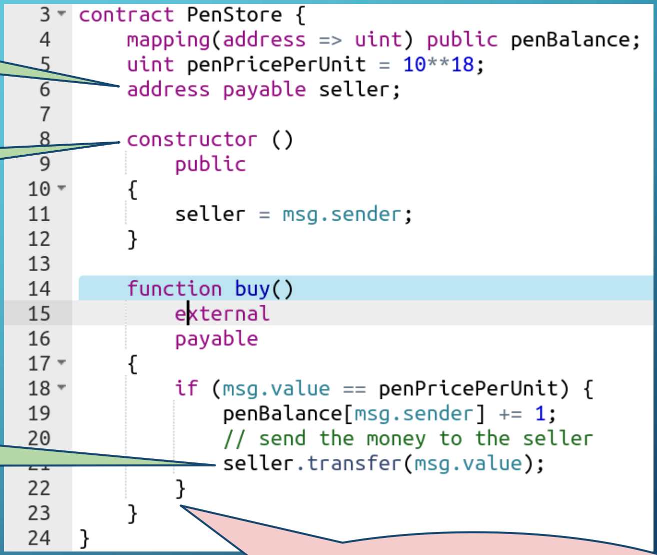
ADDING A SELLER

What if I don't send the exact amount of money to buy()?

seller

constructor

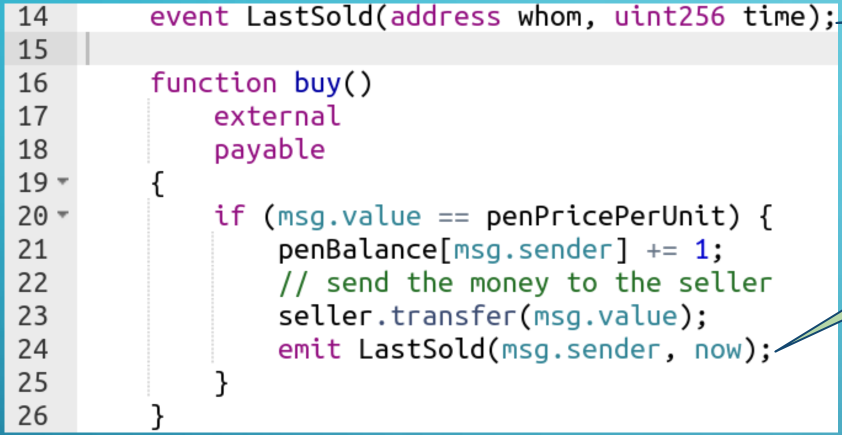
transfer : sends the money to calling address



```
contract PenStore {
    mapping(address => uint) public penBalance;
    uint penPricePerUnit = 10**18;
    address payable seller;

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

    function buy()
        external
        payable
    {
        if (msg.value == penPricePerUnit) {
            penBalance[msg.sender] += 1;
            // send the money to the seller
            seller.transfer(msg.value);
        }
    }
}
```

```
revert('not the
precise amount of
money');
```

ADDING EVENTS



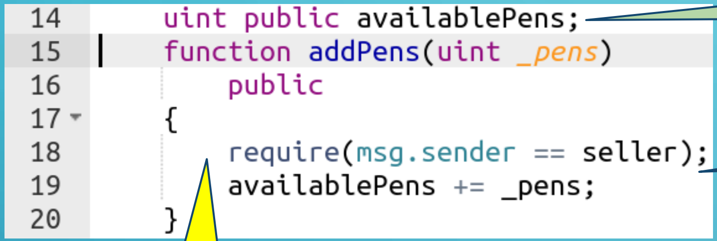
```
    event LastSold(address whom, uint256 time);

    function buy()
        external
        payable
    {
        if (msg.value == penPricePerUnit) {
            penBalance[msg.sender] += 1;
            // send the money to the seller
            seller.transfer(msg.value);
            emit LastSold(msg.sender, now);
        }
    }
```

Event declaration

Trigger event

Events are logged on the blockchain with a timestamp and own tx hash. They cannot be changed or altered! Anything connected to Ethereum JSON-RPC API/Js API can listen for them (e.g., DAPPs).

REQUIRE



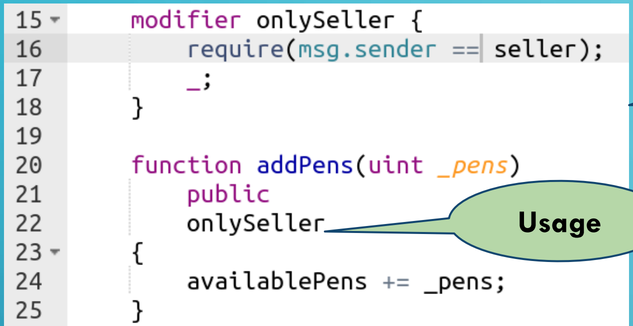
```
uint public availablePens;
function addPens(uint _pens)
    public
{
    require(msg.sender == seller);
    availablePens += _pens;
}
```

Limited stock

Only the seller can add new items

If condition is not met, it throws an error and changes are reverted!

CUSTOM MODIFIERS

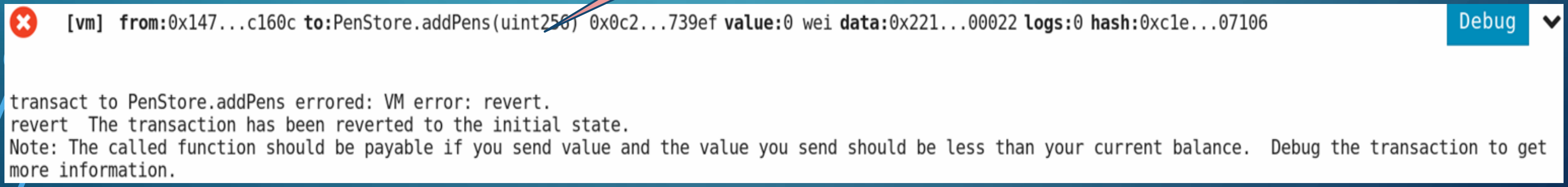


```
    modifier onlySeller {
        require(msg.sender == seller);
        _;
    }

    function addPens(uint _pens)
        public
        onlySeller
    {
        availablePens += _pens;
    }
```

Define a new access modifier which requires that the caller = seller.
The _; *calls* the function where this modifier is attached.

Usage

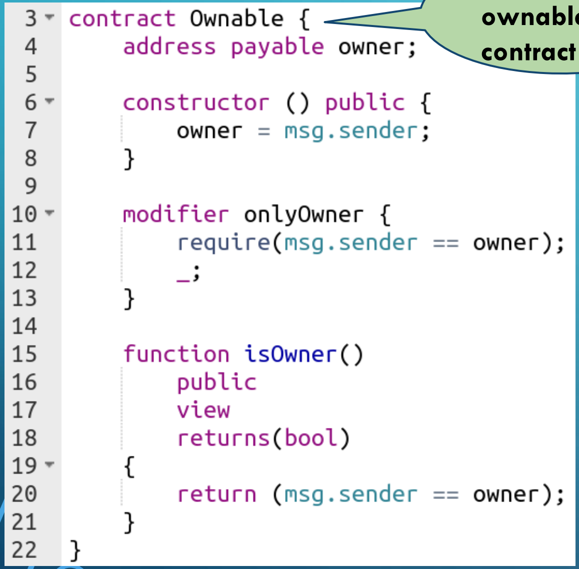
Transaction fails when `onlySeller` is not satisfied



```
[vm] from:0x147...c160c to:PenStore.addPens(uint256) 0x0c2...739ef value:0 wei data:0x221...00022 logs:0 hash:0xcle...07106

transact to PenStore.addPens errored: VM error: revert.
revert  The transaction has been reverted to the initial state.
Note: The called function should be payable if you send value and the value you send should be less than your current balance.  Debug the transaction to get
more information.
```

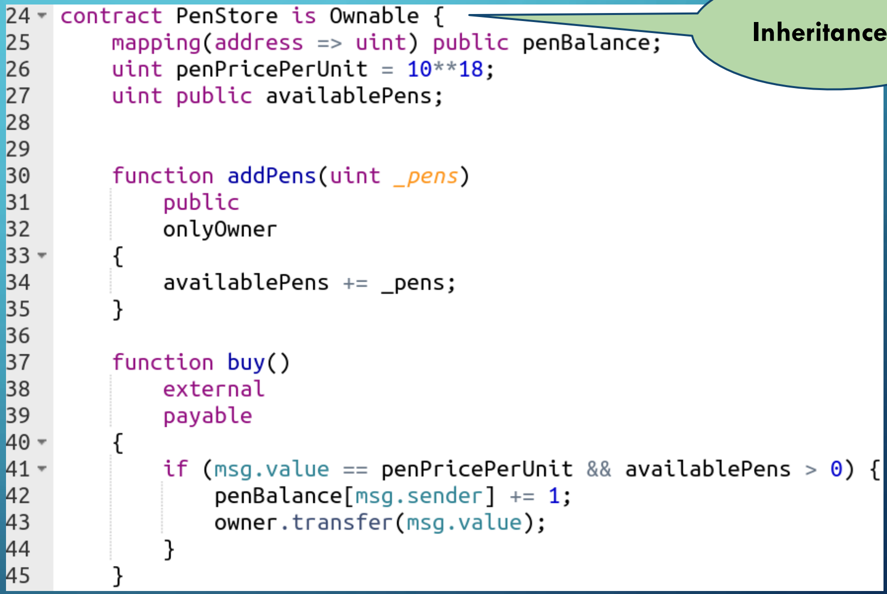
OWNABLE CONTRACTS

- Our seller is a particular type of contract owner
- Our PenStore contract is a particular type of contract: a contract owned by some address and offers a pretty common set of exposed functionalities



```
contract Ownable {
    address payable owner;

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

    modifier onlyOwner {
        require(msg.sender == owner);
        _;
    }

    function isOwner()
        public
        view
        returns(bool)
    {
        return (msg.sender == owner);
    }
}
```

Generic ownable contract



```
contract PenStore is Ownable {
    mapping(address => uint) public penBalance;
    uint penPricePerUnit = 10**18;
    uint public availablePens;


    function addPens(uint _pens)
        public
        onlyOwner
    {
        availablePens += _pens;
    }

    function buy()
        external
        payable
    {
        if (msg.value == penPricePerUnit && availablePens > 0) {
            penBalance[msg.sender] += 1;
            owner.transfer(msg.value);
        }
    }
```

Inheritance

STANDARDIZATION

- ERC: Ethereum Request for Comment
- Application-level standards and conventions
  - Token standards: ERC20
  - Lifecycle: EIP (Ethereum Improvement Proposal) -> Peer-review -> community approval -> ERC standards
- https://eips.ethereum.org/erc

LIST OF (THE MOST IMPORTANT) ERC CATEGORIES

- **Token Standards: ERC 20, ERC223, ERC721, ERC 777**, ERC1155
- Security Token Standards: ERC1400
- **Pseudo-Introspection: ERC165**, ERC1820
- Identity management: ERC725, ERC735, ERC1056, ERC1812
- **Recurring payments:** ERC1337, **ERC1620**
- Meta Transactions: ERC1077, EIP865

A list of available tokens (Oct. 2019): https://etherscan.io/tokens

TOKEN STANDARDS

ERC 20: TOKENS

- A standard for describing Ethereum Tokens
  - Token = digital asset, mostly new coins; in the future they can be stocks or bonds, etc.
  - Tokens are issued to the public through crowd sale:
    - ICO: initial coin offering
- Example: Aragon token

  https://etherscan.io/token/0x960b236A07cf122663c4303350609A66A7B288C0
- A list of ERC20 tokens (Oct 2019): https://eidoo.io/erc20-tokens-list

ERC 20

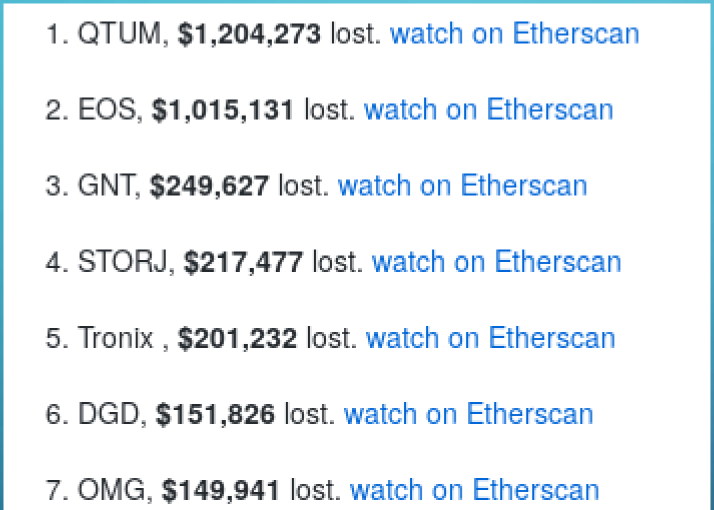
- Link: https://eips.ethereum.org/EIPS/eip-20
- Six (mandatory) functions:
  - `function `**`totalSupply`**`() public view returns (uint256)`
  - `function `**`balanceOf`**`(address _owner) public view returns (uint256 balance)`
  - `function `**`transfer`**`(address _to, uint256 _value) public returns (bool success)`
  - `function `**`transferFrom`**`(address _from, address _to, uint256 _value) public returns (bool success)`
  - `function `**`approve`**`(address _spender, uint256 _value) public returns (bool success)`
  - `function `**`allowance`**`(address _owner, address _spender) public view returns (uint256 remaining)`
- Optional functions + events (transfer + approval)

LIMITATIONS OF ERC 20

- A limitation of ERC 20 is that there is no way for smart contracts to ‘react’ to ERC20 transfer events
    - There is no way imposed by ERC 20 to notify smart contracts that they have received tokens
    - Money can be locked forever in contracts
- Increased gas consumption due to imposed combination of approve/transferFrom
- There is also an attack vector on approve/transferFrom:
    - https://docs.google.com/document/d/1YLPtQxZu1UAvO9cZ1O2RPXBbT0mooh4DYKjA_jp-RLM/edit

TOKENS LOSSES CAUSED BY ERC20 BUGS

Source: https://github.com/ethereum/EIPs/issues/223



1. QTUM, **$1,204,273** lost. watch on Etherscan
2. EOS, **$1,015,131** lost. watch on Etherscan
3. GNT, **$249,627** lost. watch on Etherscan
4. STORJ, **$217,477** lost. watch on Etherscan
5. Tronix , **$201,232** lost. watch on Etherscan
6. DGD, **$151,826** lost. watch on Etherscan
7. OMG, **$149,941** lost. watch on Etherscan

COUNTERPROPOSAL: ERC223

- https://github.com/ethereum/EIPs/issues/223
- Functions:
  - `function totalSupply() constant returns (uint256 totalSupply)`
  - `function balanceOf(address _owner) constant returns (uint256 balance)`
  - `function` **`transfer`**`(address _to, uint _value) returns (bool)`
  - `function` **`transfer`**`(address _to, uint _value, bytes _data) returns (bool)`
  - `function` **`tokenFallback`**`(address _from, uint _value, bytes _data)`
  - + the existing ERC 20 optional functions

ERC 223

- Issues solved:
  - Prevent losing tokens when sent to contracts which are not ready to receive them
    - ERC223Receiver() -> if this is implemented, then it will work properly; if not, then revert.
  - Enable notification of smart contracts that they have received tokens
    - Using events
- ERC223 is not yet implemented by the Ethereum community (they claim it is still incomplete)
  - No backwards compatibility
  - They proposed ERC 777

Buterin doesn't like this ERC 223…

ERC777: AN IMPROVED ERC 20

- Link: https://eips.ethereum.org/EIPS/eip-777
- `send(_to, _amount, data)`
  - **`data`** is a transfer message/description
- Notifications:
  - Notify the sender
    - If someone sends your tokens to someone you will be notified!
  - Notify the receiver
    - The receiver is also notified, unlike in ERC20!
- Operators: smart contracts that operate your tokens
  - Example: subscriptions can withdraw money from my operator as long as I want to.
  - Replaces approve in ERC20

ERC 777: IMPROVEMENTS OVER ERC 20

- There is a way to detect what kind of address is the receiver address and what interface it supports
  - Done via a registry: ERC 820
    - https://eips.ethereum.org/EIPS/eip-820
    - Pseudo-introspection standard; ERC 1820 superseded 820
- Send tokens in one transaction unlike in ERC 20 (approve + transferFrom)
- No vector attack
- Issue: adoption…

FUNGIBLE VS. NON-FUNGIBLE TOKENS

- Fungible tokens: not unique, can be replaced by another identical token
- Non-fungible: unique, distinguishable tokens
- ERC20, ERC777 : fungible tokens
- ERC 721: non-fungible tokens

ERC 721: NON-FUNGIBLE TOKENS

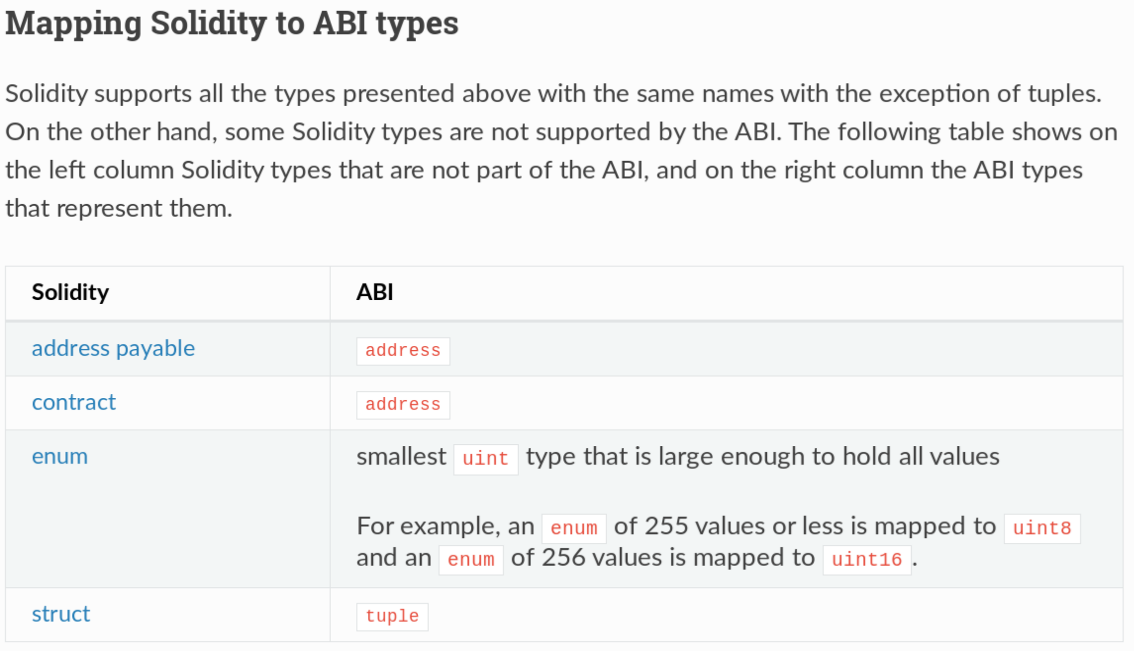
- https://github.com/ethereum/EIPs/blob/master/EIPS/eip-721.md
- http://erc721.org/
- Functions:
    - function balanceOf(address _owner) external view returns (uint256);
    - function ownerOf(uint256 _tokenId) external view returns (address);
    - function safeTransferFrom(address _from, address _to, uint256 _tokenId, bytes data) external payable;
    - function safeTransferFrom(address _from, address _to, uint256 _tokenId) external payable;
    - function transferFrom(address _from, address _to, uint256 _tokenId) external payable;
    - function approve(address _approved, uint256 _tokenId) external payable;
    - function setApprovalForAll(address _operator, bool _approved) external;
    - function getApproved(uint256 _tokenId) external view returns (address);
    - function isApprovedForAll(address _owner, address _operator) external view returns (bool);
- Events:
    - Transfer, Approval, ApprovalForAll

Example: CryptoKitties: https://etherscan.io/token/0x06012c8cf97bead5deae237070f9587f8e7a266d

PSEUDO-INTROSPECTION STANDARDS

ERC-165 STANDARD INTERFACE DETECTION

- Standard method to publish and detect what interface a smart contract implements
- Interface = set of function selectors defined by Ethereum ABI (App. Binary Interface)
  - Function selectors = first 4 bytes of the call data

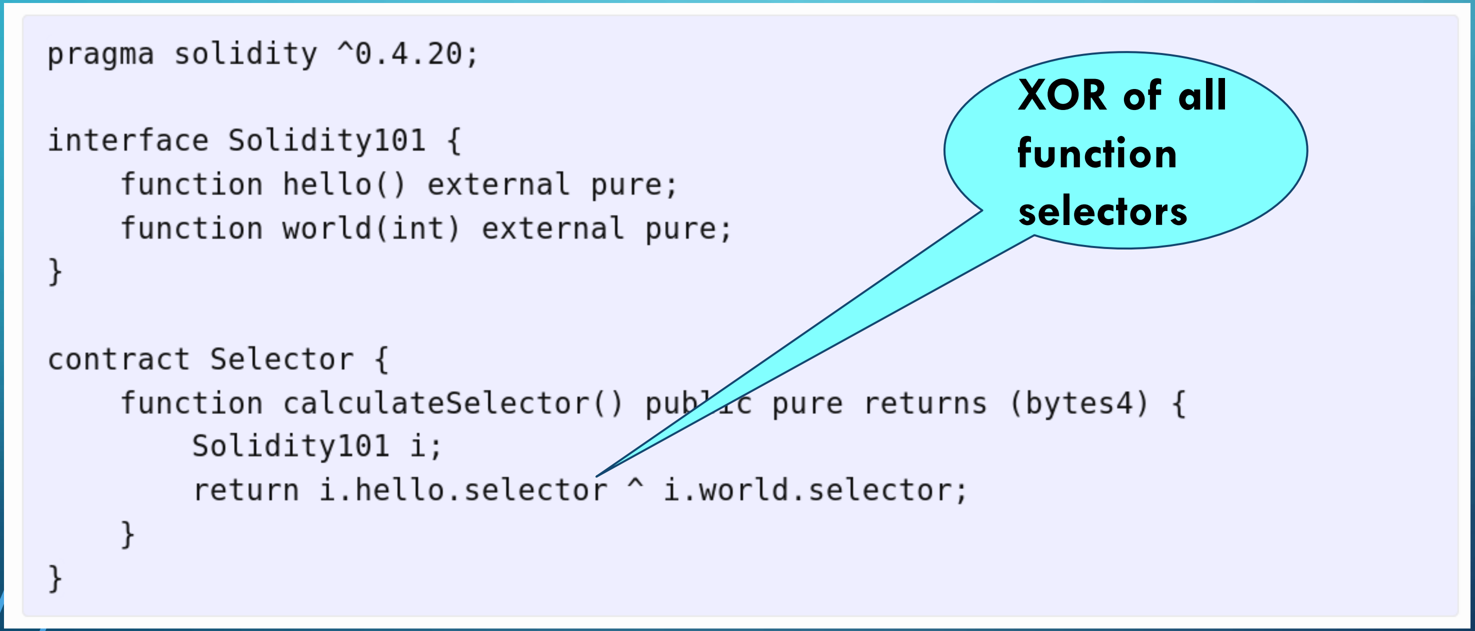


Mapping Solidity to ABI types

Solidity supports all the types presented above with the same names with the exception of tuples. On the other hand, some Solidity types are not supported by the ABI. The following table shows on the left column Solidity types that are not part of the ABI, and on the right column the ABI types that represent them.

Solidity	ABI
address payable	`address`
contract	`address`
enum	smallest `uint` type that is large enough to hold all values For example, an `enum` of 255 values or less is mapped to `uint8` and an `enum` of 256 values is mapped to `uint16` .
struct	`tuple`

HOW TO COMPUTE AN INTERFACE IDENTIFIER



```
pragma solidity ^0.4.20;

interface Solidity101 {
    function hello() external pure;
    function world(int) external pure;
}

contract Selector {
    function calculateSelector() public pure returns (bytes4) {
        Solidity101 i;
        return i.hello.selector ^ i.world.selector;
    }
}
```

XOR of all function selectors

ERC 165 INTERFACE

```
pragma solidity ^0.4.20;

interface ERC165 {
    /// @notice Query if a contract implements an interface
    /// @param interfaceID The interface identifier, as specified in ERC-165
    /// @dev Interface identification is specified in ERC-165. This function
    ///  uses less than 30,000 gas.
    /// @return `true` if the contract implements `interfaceID` and
    ///  `interfaceID` is not 0xffffffff, `false` otherwise
    function supportsInterface(bytes4 interfaceID) external view returns (bool);
}
```

HOW TO DETECT IF CONTRACT IMPLEMENTS ERC 165

- **supportsInterface**
  - returns true for the computed **id** or any other interface supported
  - returns false, otherwise
- Detect whether a contract implements or not ERC 165:
  - if (contract.supportsInterface(**01ffc9a70**01ffc9a70...0)) = false
    - then: no support for ERC 165
    - else
      - if (contract.supportsInterface(**01ffc9a70**ffffffff0...0)) = true or fail
        - then: no support for ERC 165
        - else : support for ERC 165

ERC 1820: REGISTRY CONTRACT

- Define a universally registry smart contract
  - Any address can register which interface it supports and which contract contains the implementation
  - Anyone can query the registry

```
/// @dev The interface a contract MUST implement if it is the implementer of
/// some (other) interface for any address other than itself.
interface ERC1820ImplementerInterface {
    /// @notice Indicates whether the contract implements the interface 'interfaceHash' for the address 'addr' or not.
    /// @param interfaceHash keccak256 hash of the name of the interface
    /// @param addr Address for which the contract will implement the interface
    /// @return ERC1820_ACCEPT_MAGIC only if the contract implements 'interfaceHash' for the address 'addr'.
    function canImplementInterfaceForAddress(bytes32 interfaceHash, address addr) external view returns(bytes32);
}
```

RECURRING PAYMENTS

ERC 1620: MONEY STREAMING

- Continuous payments over a finite period of time
- https://eips.ethereum.org/EIPS/eip-1620
- How it works:
  - Provider sets up a money streaming contract
  - A possible 'client' deposits funds in the contract
  - The payee is able to withdraw:
    - `rate * (current block height - starting block height)`
  - The terms can be changed if parties agree
  - The stream can be stopped
    - anytime by any party with no blockchain consensus
    - if the time period expired

EXPLORE YOURSELF OTHER ERCS

- **Token Standards: ERC 20, ERC223, ERC721, ERC 777, ERC1155**
- **Security Token Standards: ERC1400**
- **Pseudo-Introspection: ERC165, ERC1820**
- **Identity management: ERC725, ERC735, ERC1056, ERC1812**
- **Recurring payments: ERC1337, ERC1620**
- **Meta Transactions: ERC1077, EIP865**

Link: https://eips.ethereum.org/erc