ETHEREUM MINI ’18 TOKENY

EVENTS

Ether splitter

```
pragma solidity 0.4.25;

contract Splitter {

    address benficary1;

    address benficary2;

    event Split(address indexed benficary1, address indexed benficary2, uint amount);

    constructor(address _benficary1, address _benficary2) public {
        benficary1 = _benficary1;
        benficary2 = _benficary2;
    }

    function () payable external {
        uint amount = msg.value / 2;
        benficary1.transfer(amount);
        benficary2.transfer(msg.value - amount);
        emit Split(benficary1, benficary2, amount);
    }
}
```

From ethers.js

```
let address = '0x123...def';

let topic = ethers.utils.id("Split(address,address,uint)");

let filter = {
    address
    topics: [ topic ]
}

provider.on(filter, (result) => {
    console.log(result);
    // {
    //    blockNumber: 3606106,
    //    blockHash: "0x878aa7059c93239437f66baeec82332dcb2f9288bcdf6eb1ff3ba6998cdf8f69",
    //    transactionIndex: 6,
    //    removed: false,
    //    address: "0x6fC21092DA55B392b045eD78F4732bff3C580e2c",
    //    data: "0x0000000000000000000000006b5955a67d827cdf91823e3bb8f069e6c89c1d6" +
    //           "0000000000000000000000000000000000000000000016345785d8a0000",
    //    topics: [
    //      "0x179ef3319e6587f6efd3157b34c8b357141528074bcb03f9903589876168fa14",
    //      "0x90a4d0958790016bde1de8375806da3be227ff48e611aefea36303fb86bca5ad"
    //    ],
    //    transactionHash:
"0x0d6f43accb067ca8e391666f37f8e8ad75f88ebd8036c9166fd2d0b93b214d2e",
    //    logIndex: 6
    // }
});
```

EVENTS

- Allow notifying the external world that something happened
- Cheaper than storage
- Allow filtering by a specific type
- No guarantee of support

EVENT COSTS

Instruction	Cost
Arithmetics	1-10
SHA3	30
sstore	20000 gas - first write to a new position 5000 gas - subsequent writes to an existing
log	~ 375 gas + 8gas per byte + 375 per indexed parameter

SAFE MATH

Underflow

```
contract EtherSale {

  int balance = 0;

  function withdraw(int amount) public constant returns (int) {
    // ...
    return balance - amount;
  }

}


EtherSale.new().withdraw(1) == -1
```

Overflow

```
contract EtherSale {

  uint balance = 0;

  function withdraw(uint amount) public constant returns (uint) {
    // ...
    return balance - amount;
  }

}



EtherSale.new().withdraw(1) ==
115792089237316195423570985008687907853269984665640564039457584007913129639935
```

SafeMath

```
library SafeMath {
  function mul(uint256 a, uint256 b) internal pure returns (uint256) {
    if (a == 0) {
      return 0;
    }
    uint256 c = a * b;
    assert(c / a == b);
    return c;
  }

  function div(uint256 a, uint256 b) internal pure returns (uint256) {
    // assert(b > 0); // Solidity automatically throws when dividing by 0
    uint256 c = a / b;
    // assert(a == b * c + a % b); // There is no case in which this doesn't hold
    return c;
  }

  function sub(uint256 a, uint256 b) internal pure returns (uint256) {
    assert(b <= a);
    return a - b;
  }

  function add(uint256 a, uint256 b) internal pure returns (uint256) {
    uint256 c = a + b;
    assert(c >= a);
    return c;
  }
}
```

TOKENS

ETHWORKS

Tokens

- Smart contract
- Lightweight cryptocurrency
- Use Ethereum infrastructure
- You can use Ethereum wallet

Basic Token

```
pragma solidity ^0.4.25;


contract ERC20Basic {
  uint256 public totalSupply;
  function balanceOf(address who) public view returns (uint256);
  function transfer(address to, uint256 value) public returns (bool);
  event Transfer(address indexed from, address indexed to, uint256 value);
}
```

Basic Token implementation

```
import './ERC20Basic.sol';
import '../math/SafeMath.sol';


contract BasicToken is ERC20Basic {
  using SafeMath for uint256;

  mapping(address => uint256) balances;

 function transfer(address _to, uint256 _value) public returns (bool) {
    require(_to != address(0));
    require(_value <= balances[msg.sender]);

    balances[msg.sender] = balances[msg.sender].sub(_value);
    balances[_to] = balances[_to].add(_value);
    Transfer(msg.sender, _to, _value);
    return true;
  }

 function balanceOf(address _owner) public view returns (uint256 balance) {
    return balances[_owner];
  }

}
```

TOKENS AND CONTRACTS

Token Sale

```
contract TokenSale {
  using SafeMath for uint256;

  Token public token = Token(0x…);

  address public wallet = …;

  function buyToken() public payable {
    uint256 weiAmount = msg.value;
    uint256 tokens = weiAmount.mul(10);
    token.transfer(msg.sender, tokens);
    wallet.transfer(msg.value);
  }

}
```

MORE COMPLETE TOKEN SALE

- Dates (start, end)
- Sales limit (i.e. hard cap)
- Configurable treasury and token
- Rate
- Preico
- Allocations

Token Sale

```
contract TokenSale {
  using SafeMath for uint256;

  Token public token = Token(0x…);

  address public wallet = …;

  function buyToken() public payable {
    uint256 weiAmount = msg.value;
    uint256 tokens = weiAmount.mul(10);
    token.transfer(msg.sender, tokens);
    wallet.transfer(msg.value);
  }

}
```

Ether Sale

```
contract EtherSale {
  using SafeMath for uint256;

  Token public token = Token(…);

  address public wallet = …;

  function buyEther() public {
    uint256 weiAmount = ?; //Eeeee…
    uint256 etherAmount = weiAmount.div(10);
    wallet.transfer(msg.sender, etherAmount); //Wait what?
  }

}
```

?

ERC20 Token

```
contract ERC20 is ERC20Basic {
  function allowance(address owner, address spender) public view returns (uint256);
  function transferFrom(address from, address to, uint256 value) public returns (bool);
  function approve(address spender, uint256 value) public returns (bool);
  event Approval(address indexed owner, address indexed spender, uint256 value);
}
```

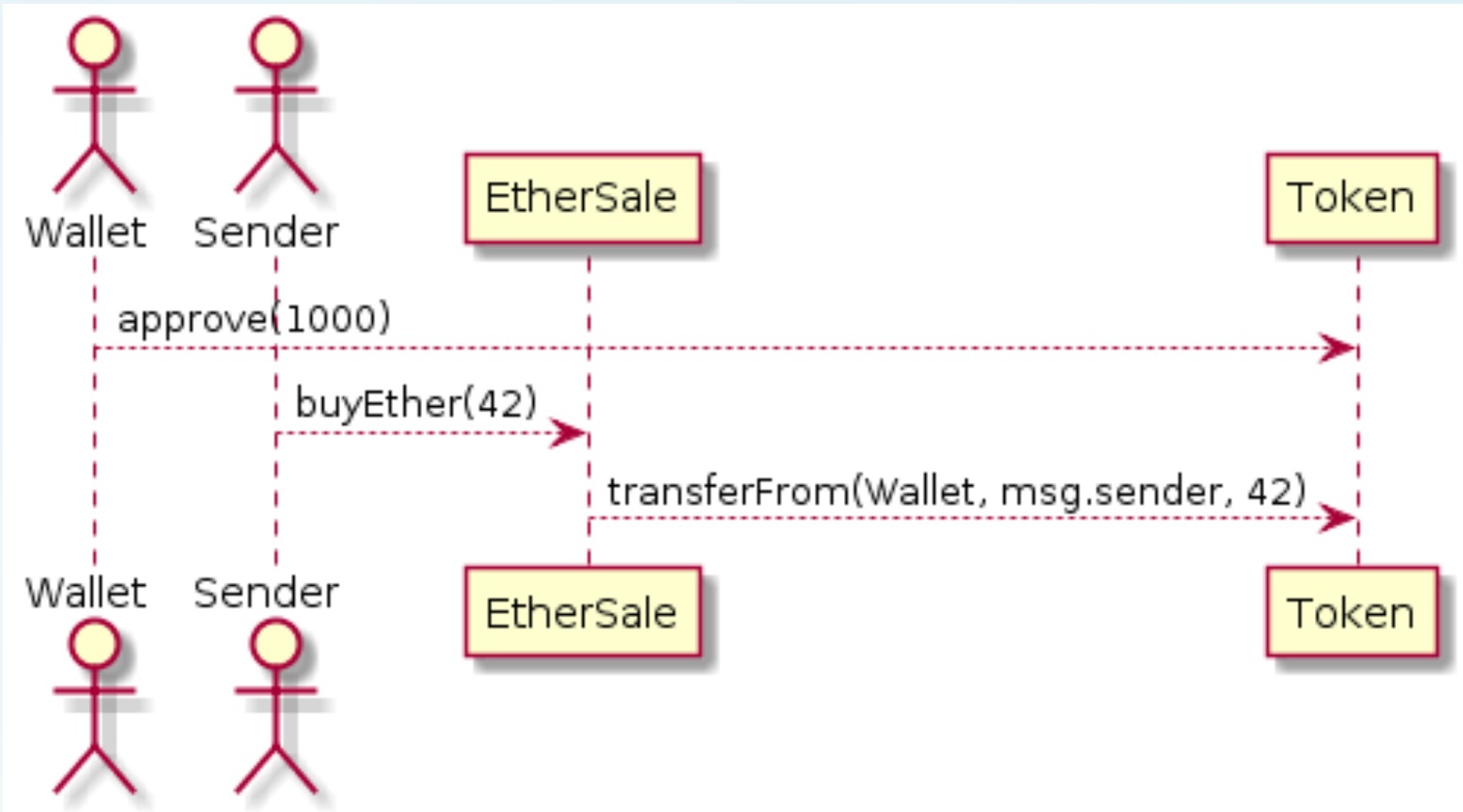
ERC20 Token implementation

```
contract StandardToken is ERC20, BasicToken {

  mapping (address => mapping (address => uint256)) internal allowed;

  function transferFrom(address _from, address _to, uint256 _value) public returns (bool) {
    require(_to != address(0));
    require(_value <= balances[_from]);
    require(_value <= allowed[_from][msg.sender]);

    balances[_from] = balances[_from].sub(_value);
    balances[_to] = balances[_to].add(_value);
    allowed[_from][msg.sender] = allowed[_from][msg.sender].sub(_value);
    Transfer(_from, _to, _value);
    return true;
  }

  function approve(address _spender, uint256 _value) public returns (bool) {
    allowed[msg.sender][_spender] = _value;
    Approval(msg.sender, _spender, _value);
    return true;
  }

  function allowance(address _owner, address _spender) public view returns (uint256) {
    return allowed[_owner][_spender];
  }

}
```

Ether Sale

```
contract EtherSale {
  using SafeMath for uint256;

  Token public token = Token(…);

  address public wallet = …;

  function buyEther(uint weiAmount) public {
    uint256 etherAmount = weiAmount.div(10);
    wallet.transferFrom(wallet, msg.sender, etherAmount);
  }

}



//Token holder:
Token.approve(etherSaleAddress, 20);

//Sender:
EtherSale.buyEther(20);
```

Ether Sale


Wallet
Sender
EtherSale
Token
approve(1000)
buyEther(42)
transferFrom(Wallet, msg.sender, 42)
Wallet
Sender
EtherSale
Token

HOMEWORK

Ether splitter

```
contract Splitter {

    address benficary1;

    address benficary2;

    function Splitter(address _benficary1, address _benficary2) public {
        benficary1 = _benficary1;
        benficary2 = _benficary2;
    }

    function () payable public {
        uint amount = msg.value / 2;
        benficary1.transfer(amount);
        benficary2.transfer(msg.value - amount);
    }

}
```

Token splitter

```
pragma solidity 0.4.25;

contract Splitter {

    constructor(address [] beneficiaries, address feeCollector, ERC20 token) public {
        //...
    }

    function split(uint amount) public {
        //...
    }
}
```

WRITE A TOKEN SPLITTER

- Write a contract doing a token split
- Take constant fee of 1 token

- Cover with tests
- Use approval method
- No unauthorized withdraws!
- No out of gas exceptions

  (up to reasonable beneficiaries number)

Send to: https://goo.gl/CBhzFQ

Thank you.