Workshop Ethereum DAPPS

How to create an Ethereum Dapp

Sheets
https://web3examples.com/Saxion

Intro Gerard Persoon

- Education
  - Computer science (TU Delft), IT Audit (VU), Startup Validation Lab (Yes!Delft)
- Roles
  - Software developer
  - Line manager & Technical project manager
  - IT Auditor
- Teaching
  - The hague university of applied science (programming blockchains)
  - HES Amsterdam
  - Tilburg University
- Companies
  - Enovation, Ernst & Young, IBM, ABN AMRO, DB Schenker, HMC
- Contact
  - mail@gpersoon.com
  - https://www.linkedin.com/in/gpersoon
  - Twitter: @gpersoon

Content

- 14.00 Demo (Metamask, Remix, Oneclickdapp)
- 14.10 Practice (20 min)
- 14.30 Update & Tips
- 14.40 Practice (20 min)
- 15.00 Group presentation & feedback
- 15.20 Evaluation & subjects next time

Sources

https://remix.ethereum.org

https://metamask.io

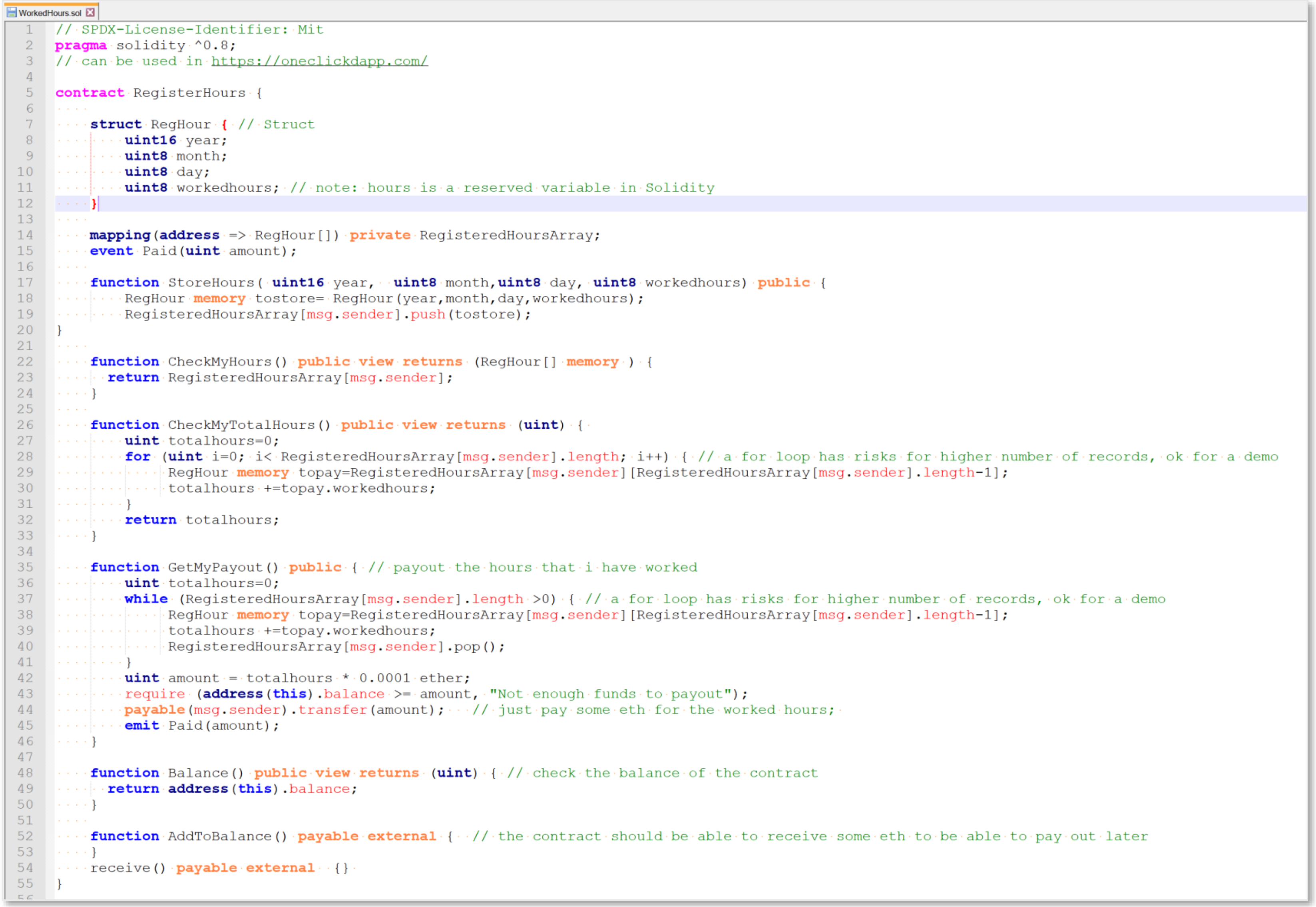
https://oneclickdapp.com

https://goerli-faucet.slock.it

https://web3examples.com/ethereum/solidity_examples

https://docs.soliditylang.org

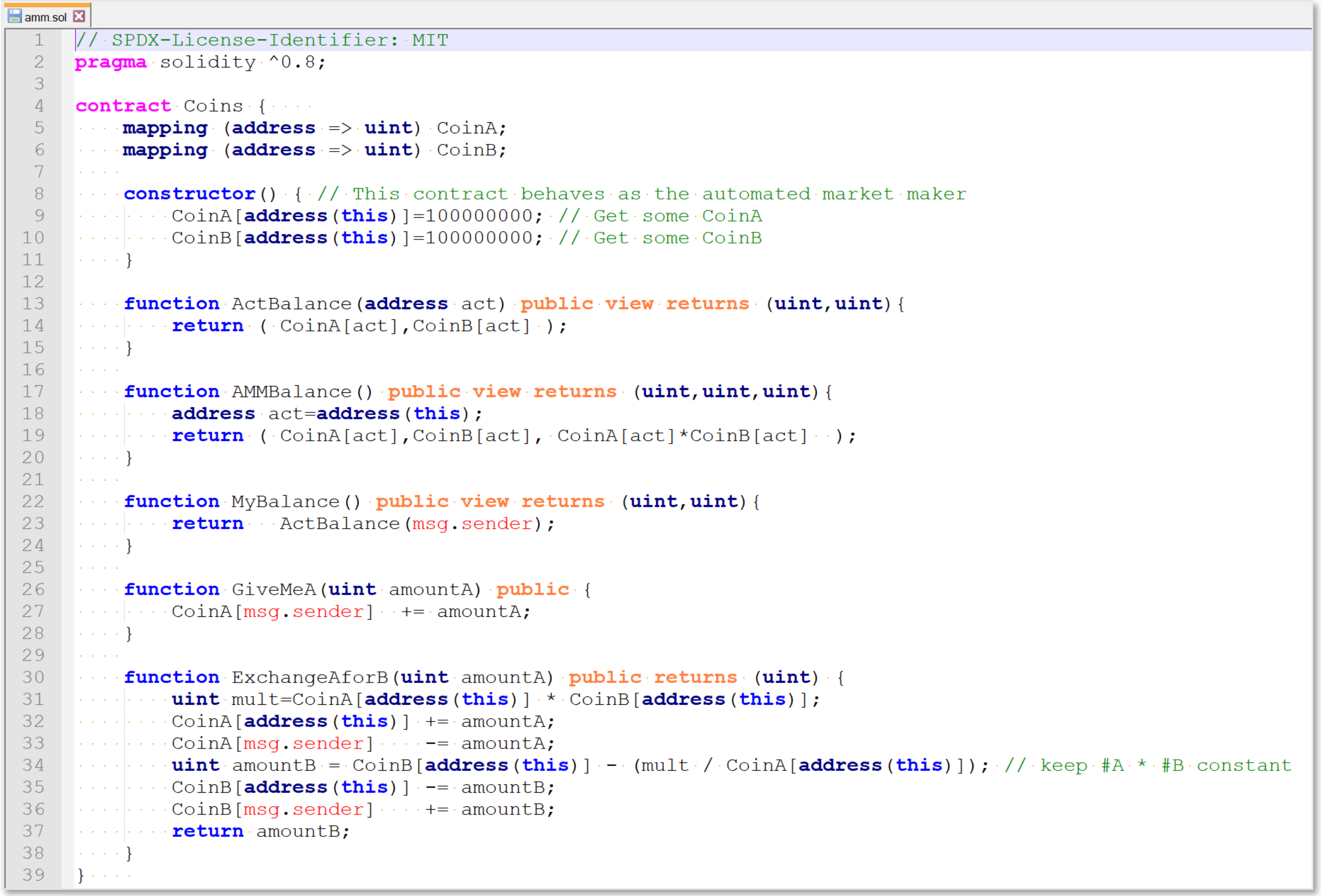
Example: WorkedHours



```solidity
// SPDX-License-Identifier: Mit
pragma solidity ^0.8;
// can be used in https://oneclickdapp.com/

contract RegisterHours {

    struct RegHour { // Struct
        uint16 year;
        uint8 month;
        uint8 day;
        uint8 workedhours; // note: hours is a reserved variable in Solidity
    }

    mapping(address => RegHour[]) private RegisteredHoursArray;
    event Paid(uint amount);

    function StoreHours( uint16 year,  uint8 month,uint8 day, uint8 workedhours) public {
        RegHour memory tostore= RegHour(year,month,day,workedhours);
        RegisteredHoursArray[msg.sender].push(tostore);
}

    function CheckMyHours() public view returns (RegHour[] memory ) {
      return RegisteredHoursArray[msg.sender];
    }

    function CheckMyTotalHours() public view returns (uint) {
        uint totalhours=0;
        for (uint i=0; i< RegisteredHoursArray[msg.sender].length; i++) { // a for loop has risks for higher number of records, ok for a demo
             RegHour memory topay=RegisteredHoursArray[msg.sender][RegisteredHoursArray[msg.sender].length-1];
             totalhours +=topay.workedhours;
        }
        return totalhours;
    }

    function GetMyPayout() public { // payout the hours that i have worked
        uint totalhours=0;
        while (RegisteredHoursArray[msg.sender].length >0) { // a for loop has risks for higher number of records, ok for a demo
             RegHour memory topay=RegisteredHoursArray[msg.sender][RegisteredHoursArray[msg.sender].length-1];
             totalhours +=topay.workedhours;
             RegisteredHoursArray[msg.sender].pop();
        }
        uint amount = totalhours * 0.0001 ether;
        require (address(this).balance >= amount, "Not enough funds to payout");
        payable(msg.sender).transfer(amount);   // just pay some eth for the worked hours;
        emit Paid(amount);
    }

    function Balance() public view returns (uint) { // check the balance of the contract
      return address(this).balance;
    }

    function AddToBalance() payable external {  // the contract should be able to receive some eth to be able to pay out later
    }
    receive() payable external  {}
}
```

https://github.com/web3examples/ethereum/blob/master/solidity_examples/WorkedHours.sol

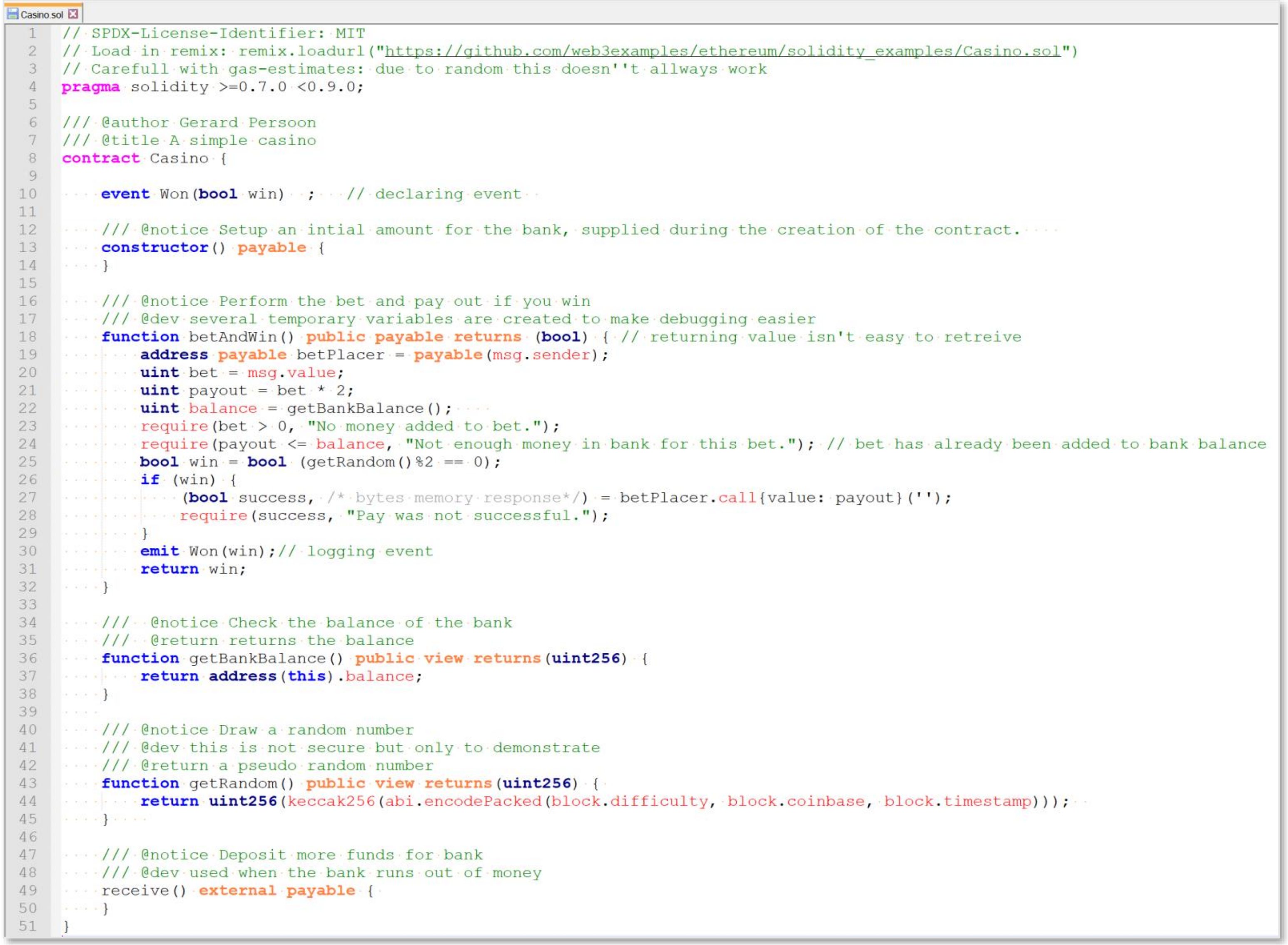
Example: Automated Market Maker



```solidity
// SPDX-License-Identifier: MIT
pragma solidity ^0.8;

contract Coins {
    mapping (address => uint) CoinA;
    mapping (address => uint) CoinB;

    constructor() { // This contract behaves as the automated market maker
        CoinA[address(this)]=100000000; // Get some CoinA
        CoinB[address(this)]=100000000; // Get some CoinB
    }

    function ActBalance(address act) public view returns (uint,uint){
        return ( CoinA[act],CoinB[act] );
    }

    function AMMBalance() public view returns (uint,uint,uint){
        address act=address(this);
        return ( CoinA[act],CoinB[act], CoinA[act]*CoinB[act]  );
    }

    function MyBalance() public view returns (uint,uint){
        return  ActBalance(msg.sender);
    }

    function GiveMeA(uint amountA) public {
        CoinA[msg.sender]  += amountA;
    }

    function ExchangeAforB(uint amountA) public returns (uint) {
        uint mult=CoinA[address(this)] * CoinB[address(this)];
        CoinA[address(this)] += amountA;
        CoinA[msg.sender]    -= amountA;
        uint amountB = CoinB[address(this)] - (mult / CoinA[address(this)]); // keep #A * #B constant
        CoinB[address(this)] -= amountB;
        CoinB[msg.sender]    += amountB;
        return amountB;
    }
}
```

https://github.com/web3examples/ethereum/blob/master/solidity_examples/amm.sol

Example: Casino

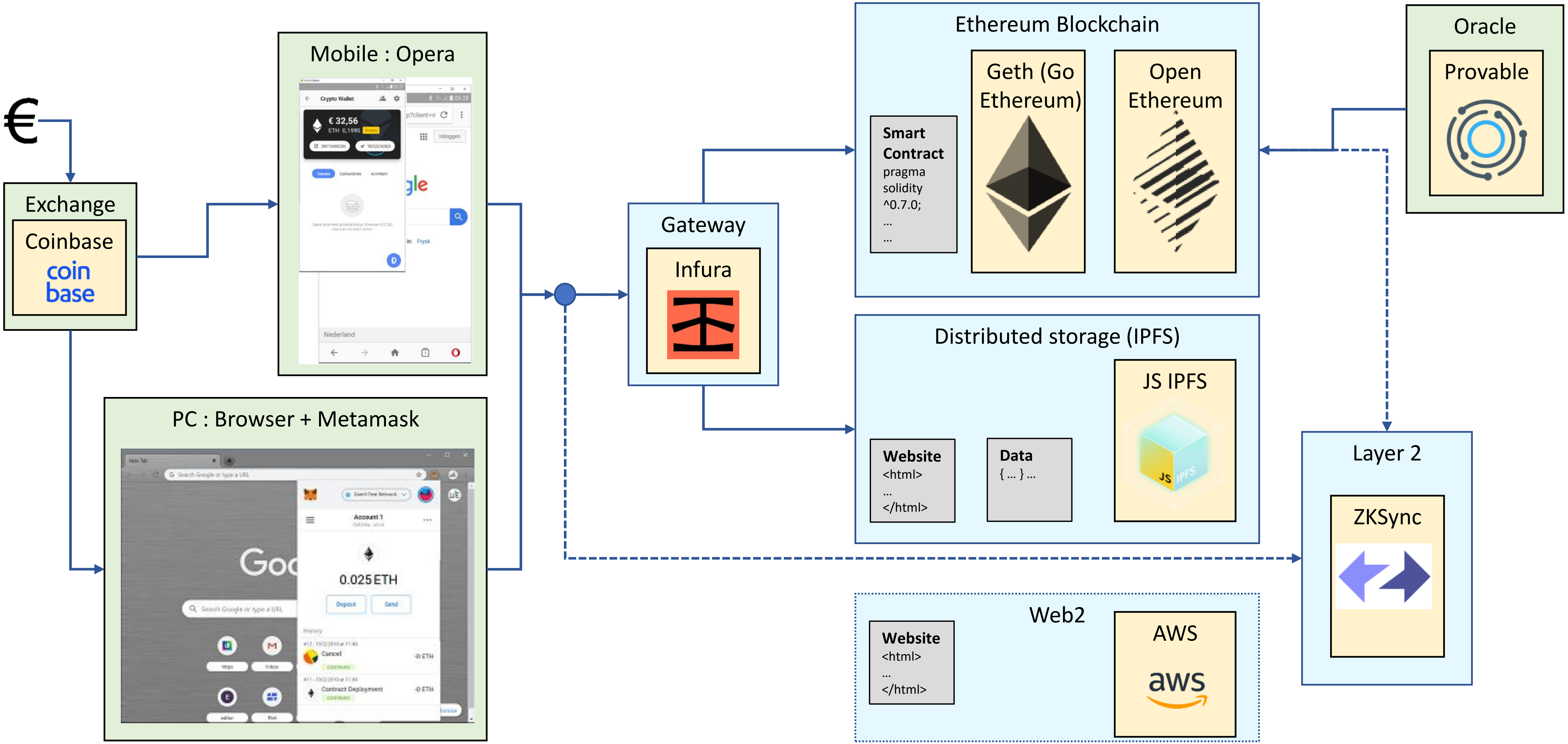


```
// SPDX-License-Identifier: MIT
// Load in remix: remix.loadurl("https://github.com/web3examples/ethereum/solidity_examples/Casino.sol")
// Carefull with gas-estimates: due to random this doesn''t allways work
pragma solidity >=0.7.0 <0.9.0;

/// @author Gerard Persoon
/// @title A simple casino
contract Casino {

    event Won(bool win)  ;   // declaring event

    /// @notice Setup an intial amount for the bank, supplied during the creation of the contract.
    constructor() payable {
    }

    /// @notice Perform the bet and pay out if you win
    /// @dev several temporary variables are created to make debugging easier
    function betAndWin() public payable returns (bool) { // returning value isn't easy to retreive
        address payable betPlacer = payable(msg.sender);
        uint bet = msg.value;
        uint payout = bet * 2;
        uint balance = getBankBalance();
        require(bet > 0, "No money added to bet.");
        require(payout <= balance, "Not enough money in bank for this bet."); // bet has already been added to bank balance
        bool win = bool (getRandom()%2 == 0);
        if (win) {
            (bool success, /* bytes memory response*/) = betPlacer.call{value: payout}('');
            require(success, "Pay was not successful.");
        }
        emit Won(win);// logging event
        return win;
    }

    ///  @notice Check the balance of the bank
    ///  @return returns the balance
    function getBankBalance() public view returns(uint256) {
        return address(this).balance;
    }

    /// @notice Draw a random number
    /// @dev this is not secure but only to demonstrate
    /// @return a pseudo random number
    function getRandom() public view returns(uint256) {
        return uint256(keccak256(abi.encodePacked(block.difficulty, block.coinbase, block.timestamp)));
    }

    /// @notice Deposit more funds for bank
    /// @dev used when the bank runs out of money
    receive() external payable {
    }
}
```

https://github.com/web3examples/ethereum/blob/master/solidity_examples/Casino.sol

SAXION

DAPP architecture



Subjects next time

- Tokens
- More solidity & best practices
- Ethereum nodes (&rpc)
- Ethereum deployment tools (truffle)
- Websites (javascript) to access smart contracts
- IPFS
- Security
- Testing
- Ethereum name system
- Oracles
- TheGraph
- Layer2

Date & Time?