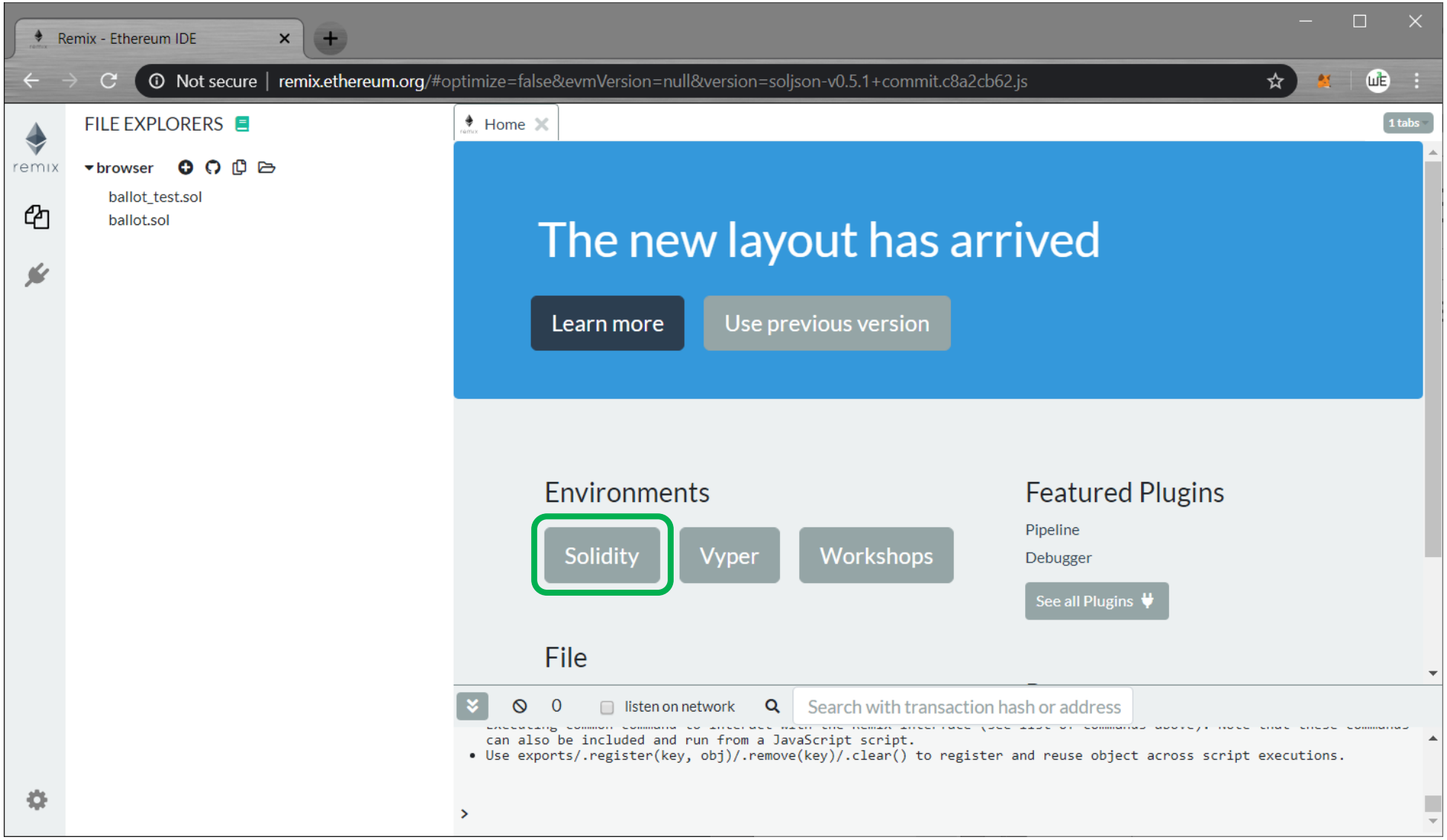
PD-3.0 Remix IDE & Solidity

- PD-3.1 Remix Tools
- PD-3.2 Solidity basic constructs
- PD-3.3 Solidity theory
- PD-3.4 Solidity mini applications
- PD-3.5 Casino & Debug

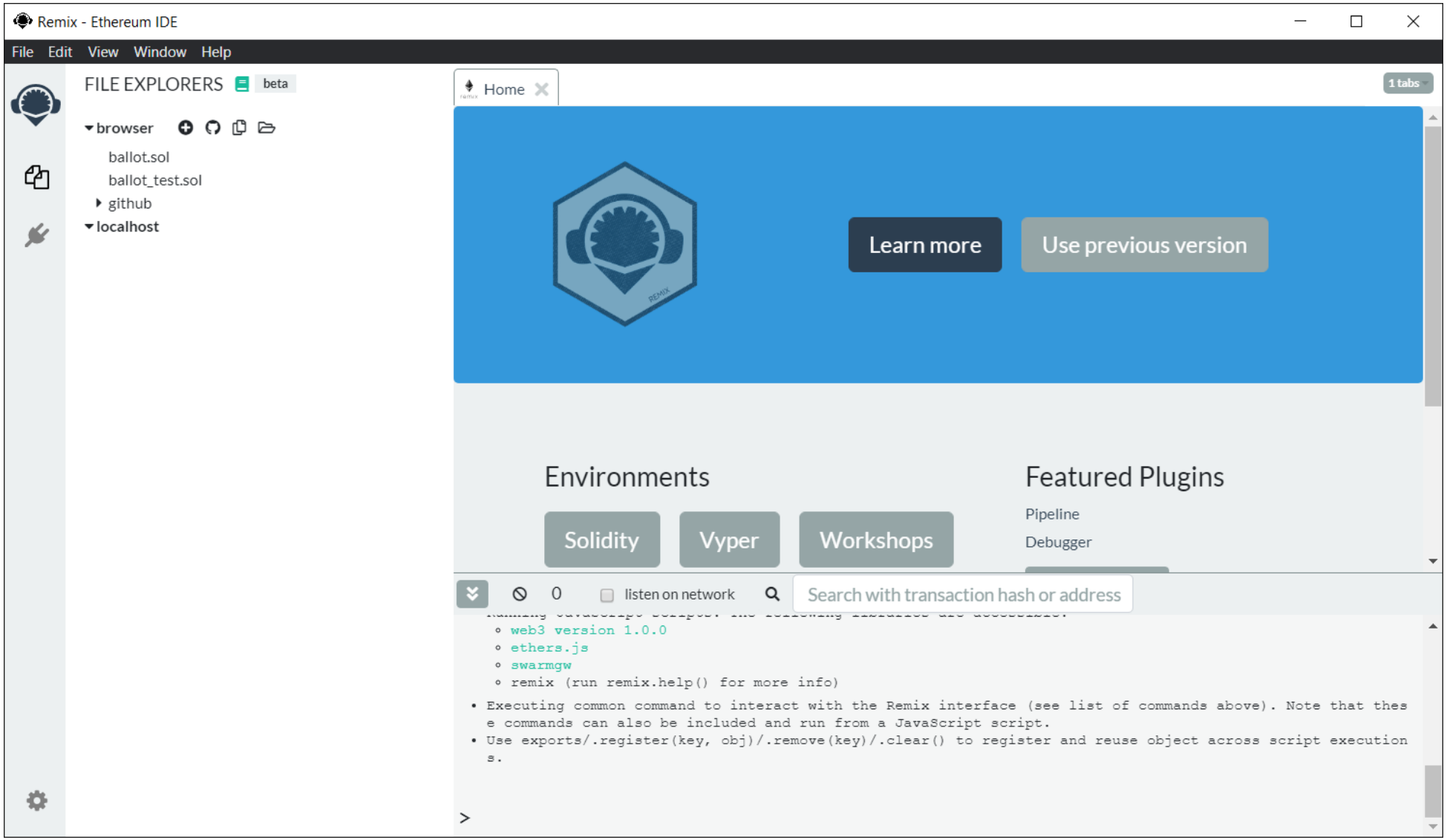
https://solidity.readthedocs.io/en/latest/solidity-in-depth.html

PD-3.1 Remix IDE - online



https://remix.ethereum.org/

PD-3.1 Remix IDE – offline version

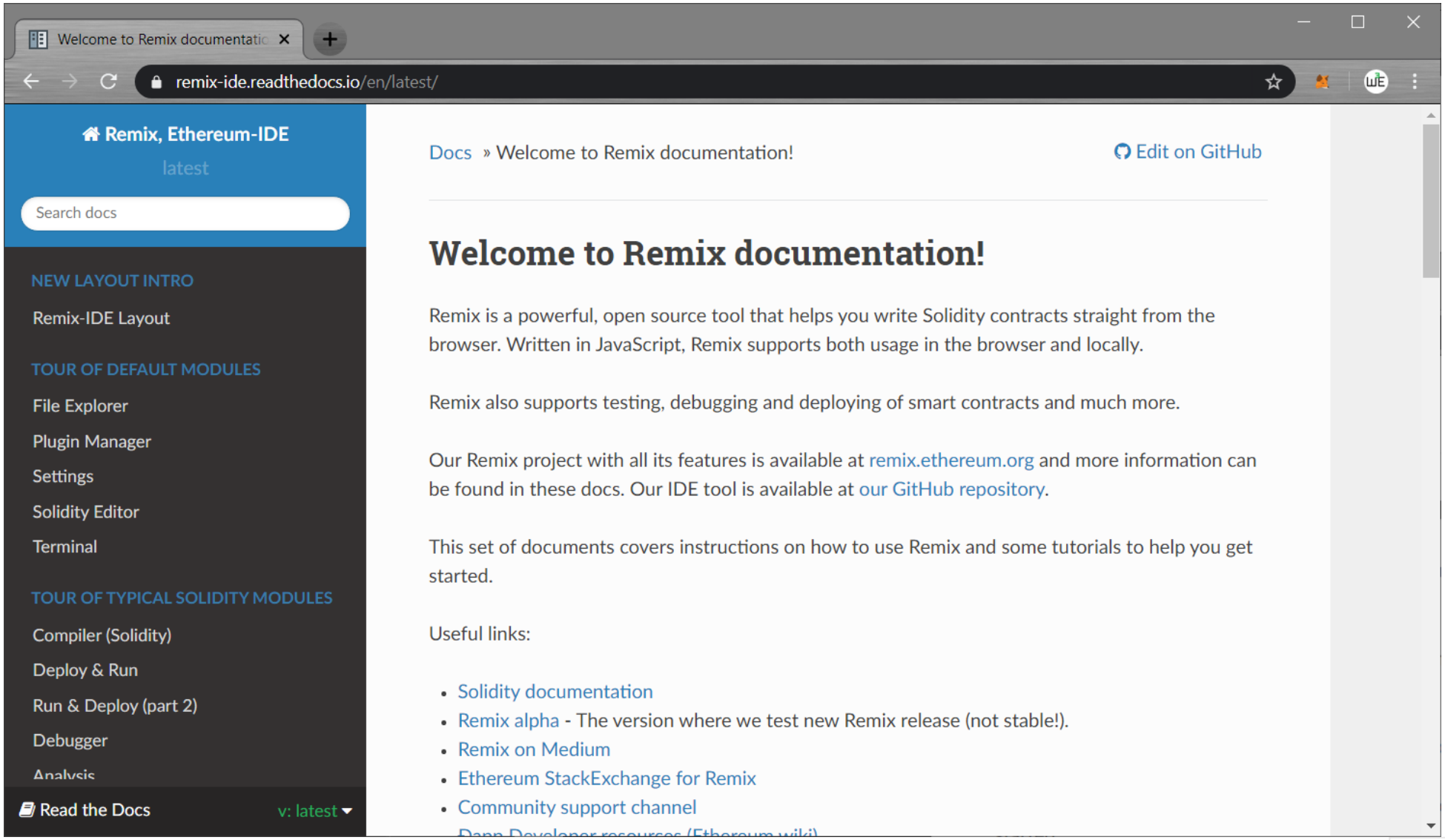


Disadvantage:
can't use Metamask

https://github.com/ethereum/remix-desktop/releases

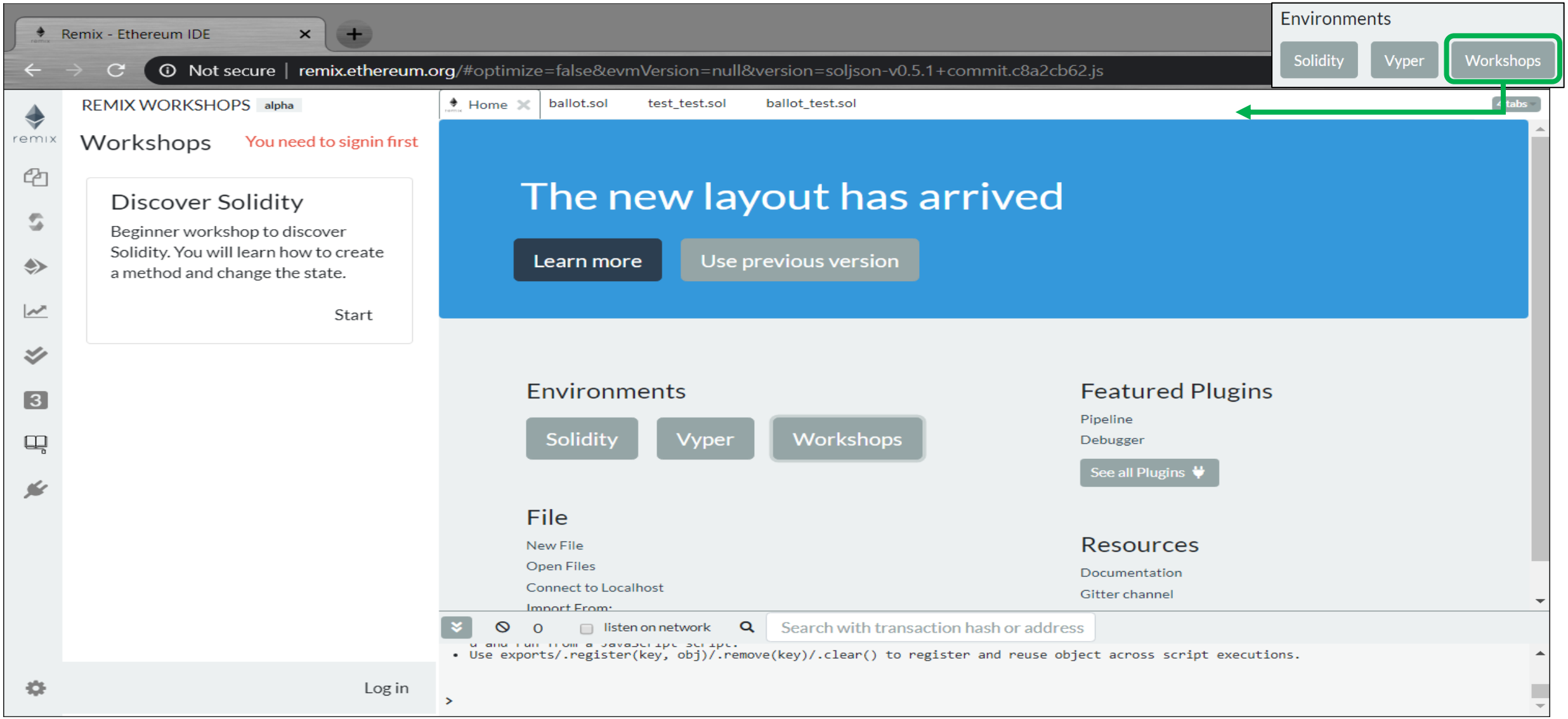
PD-3.1 Prepare Remix

PD-3.1 Remix Documentation

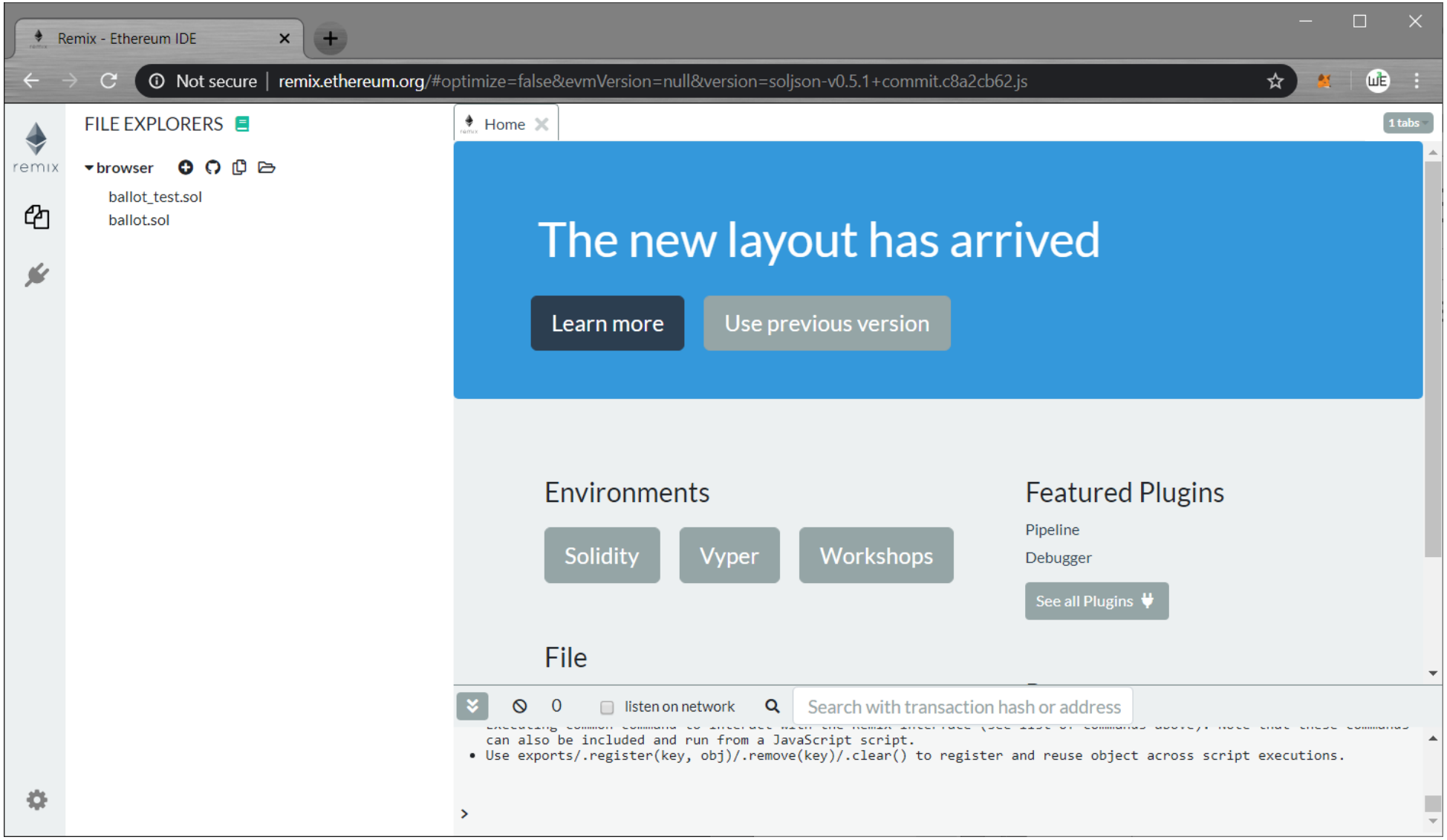


https://remix-ide.readthedocs.io/en/latest/

PD-3.1 Remix - Workshops



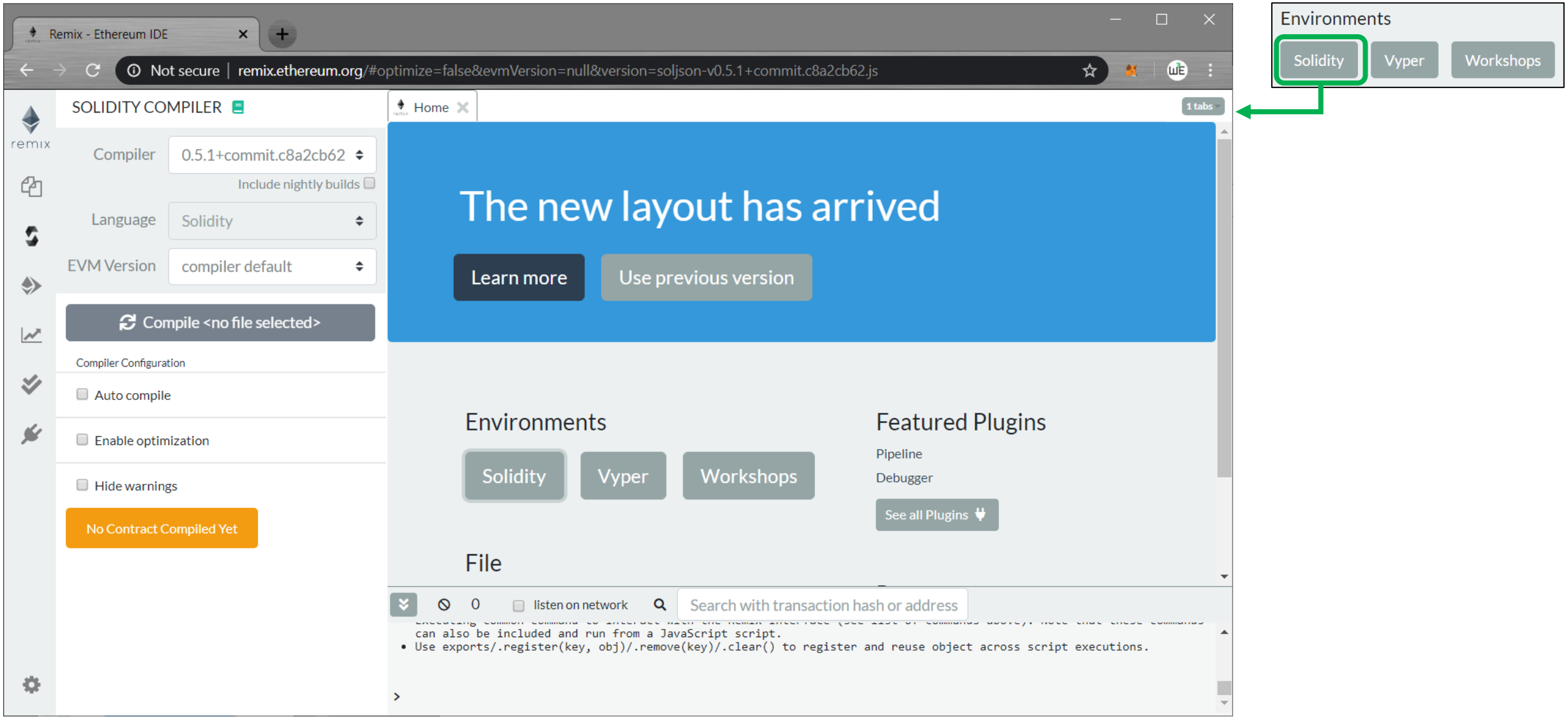
PD-3.1 Remix IDE - start



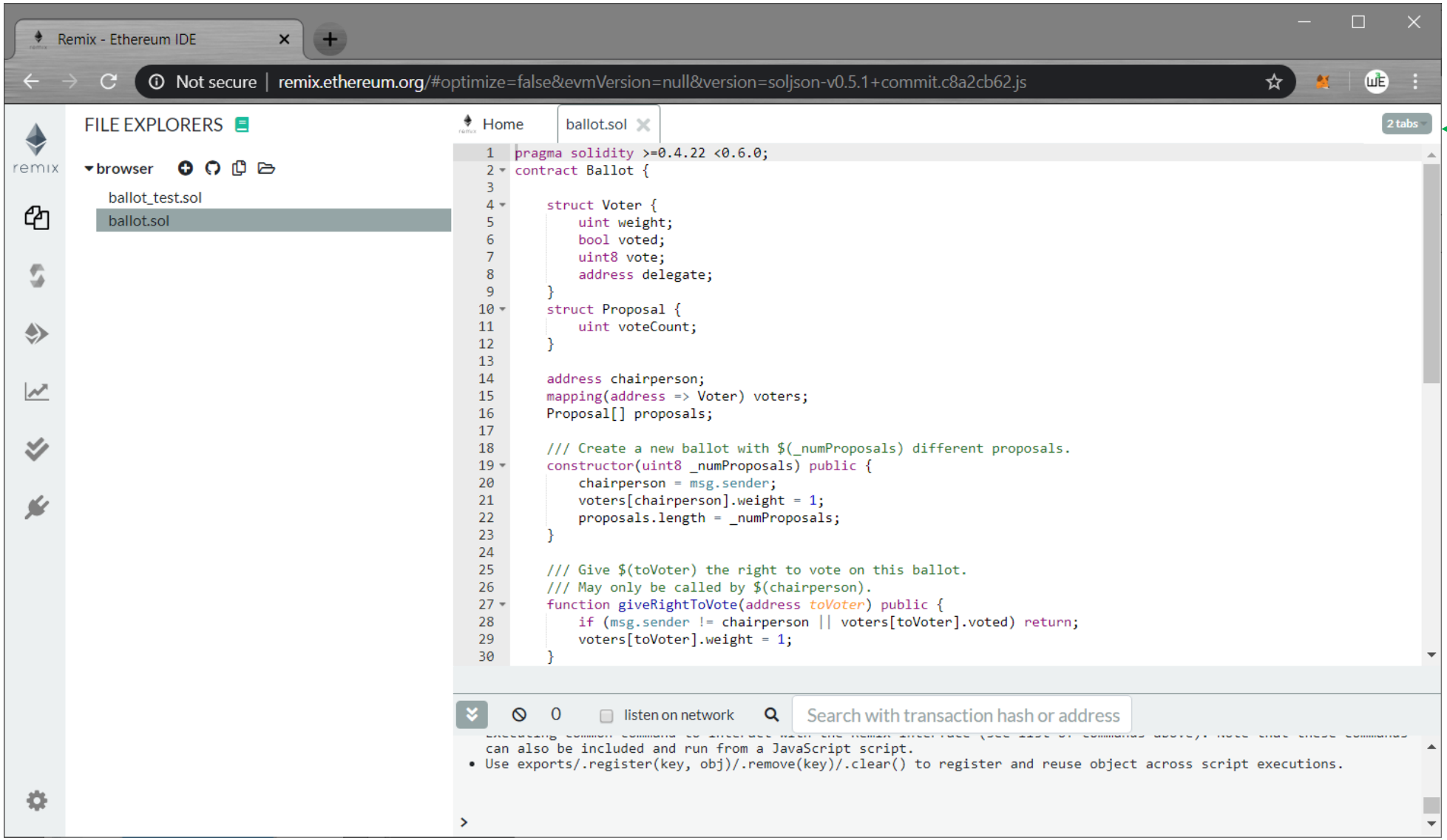
http://web3examples.com/ethereum/install/Prepare_Remix.html

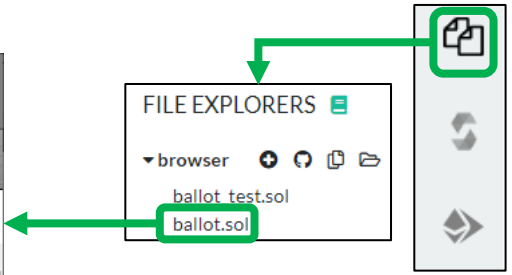
https://remix.ethereum.org

PD-3.1 Remix – solidity icons

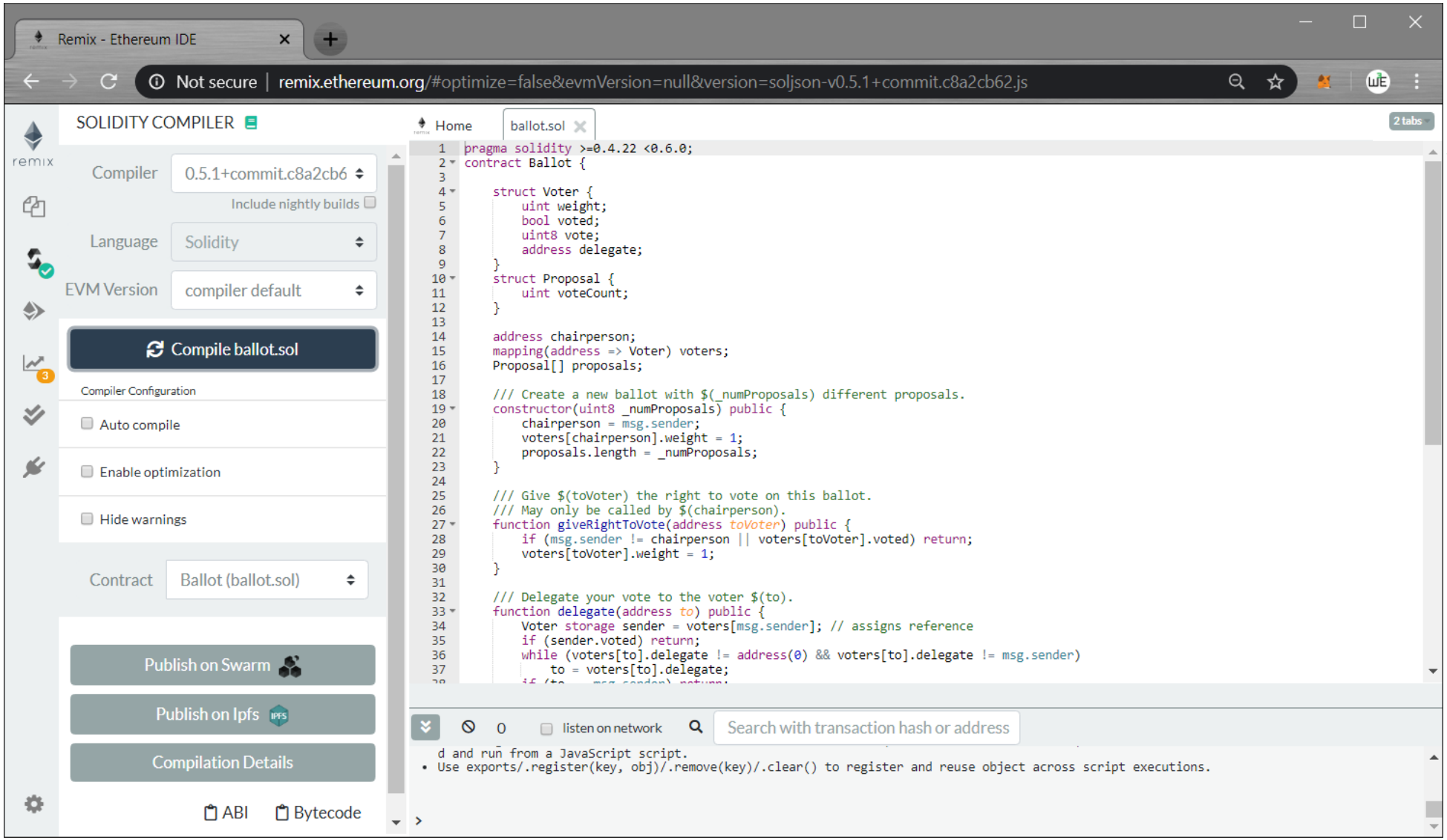


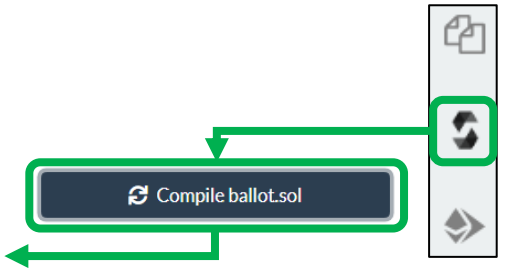
PD-3.1 Remix - file explorer



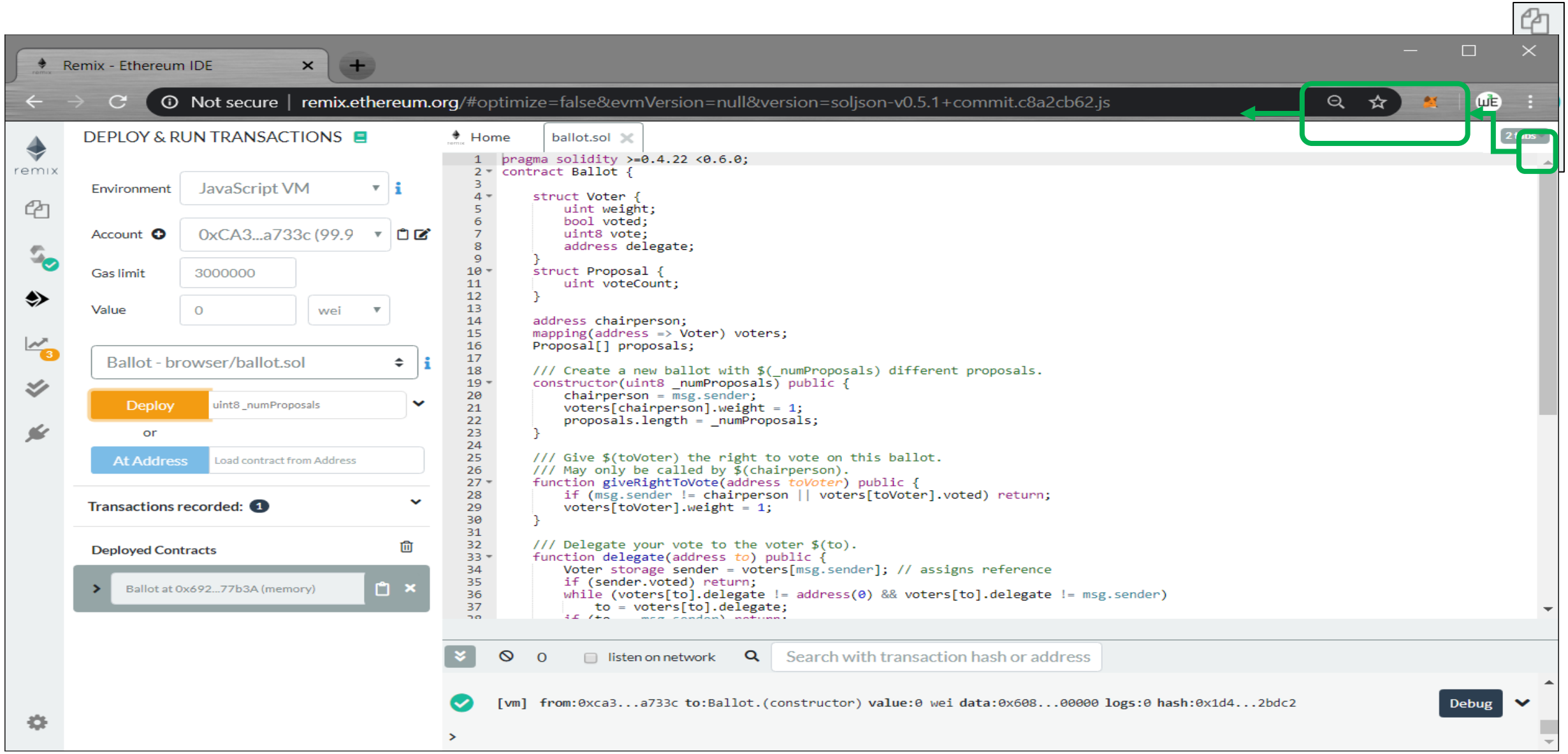


PD-3.1 Remix – Solidity compiler

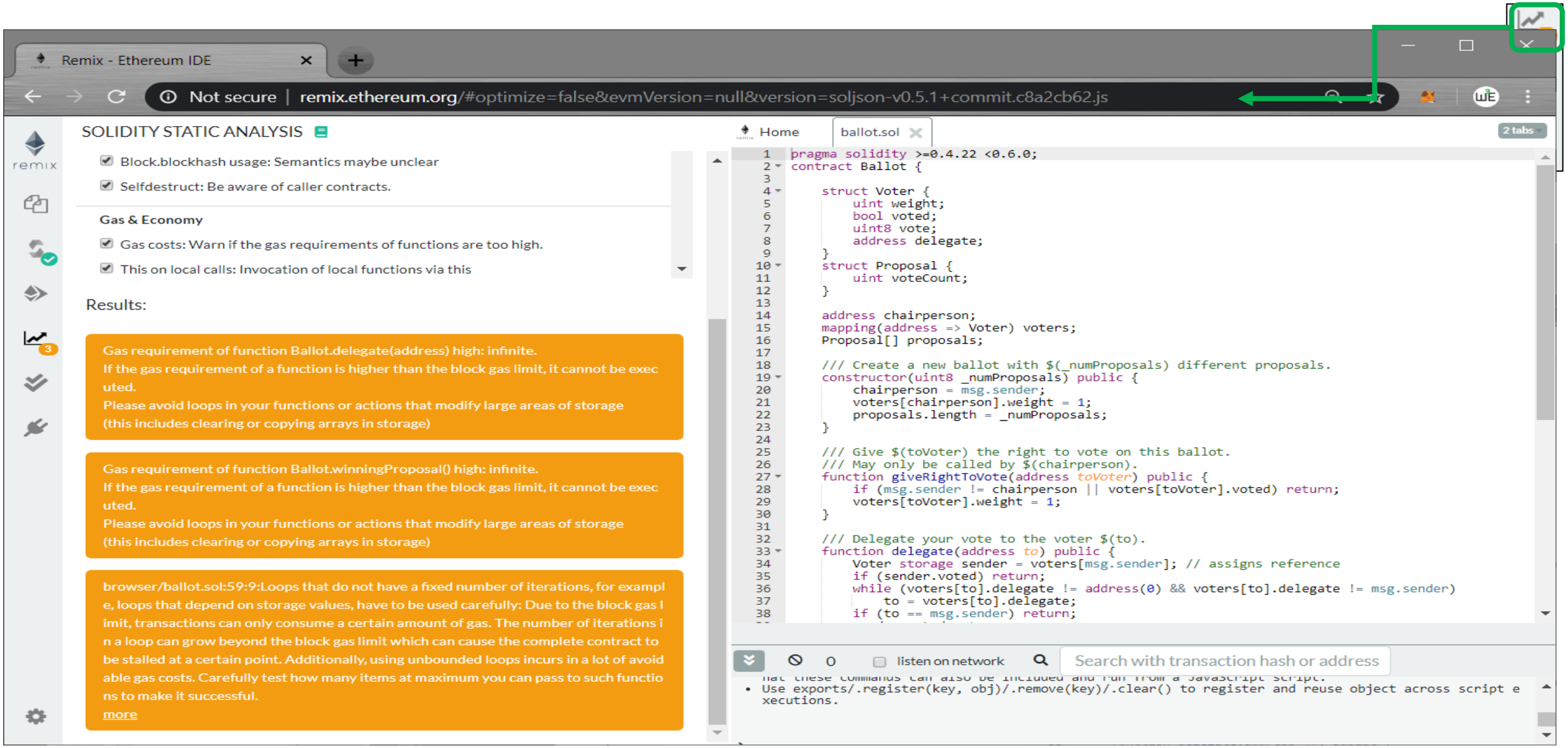




PD-3.1 Remix – Deploy & Run Transactions



PD-3.1 Remix – Solidity Static Analysis



PD-3.2 Function Visibility

- **public -** all can access
  - solidity copies array arguments to memory
- **external -** Cannot be accessed internally, only externally
  - function reads directly from calldata
- **internal -** only this contract and contracts deriving from it can access
- **private -** can be accessed only from this contract

PD-3.2 Visibility

```solidity
pragma solidity^0.5.12;

contract Test {
    string public StoredString="Initial string";

    function _ReadString() internal view returns (string memory) {
        return StoredString;
    }
    function ReadString() public view returns (string memory) {
        return StoredString;
    }
    function WriteString1(string memory s) public returns (string memory){
        StoredString = s;
        return _ReadString();
    }
    function WriteString2(string calldata s) external returns (string memory){
        return WriteString1(s);
    }
    function EchoString(string memory s) public pure returns (string memory){
        return s;
    }
}
```

```
ReadString
StoredString
WriteString1
WriteString2
EchoString
```

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

PD-3.3 Function modifier

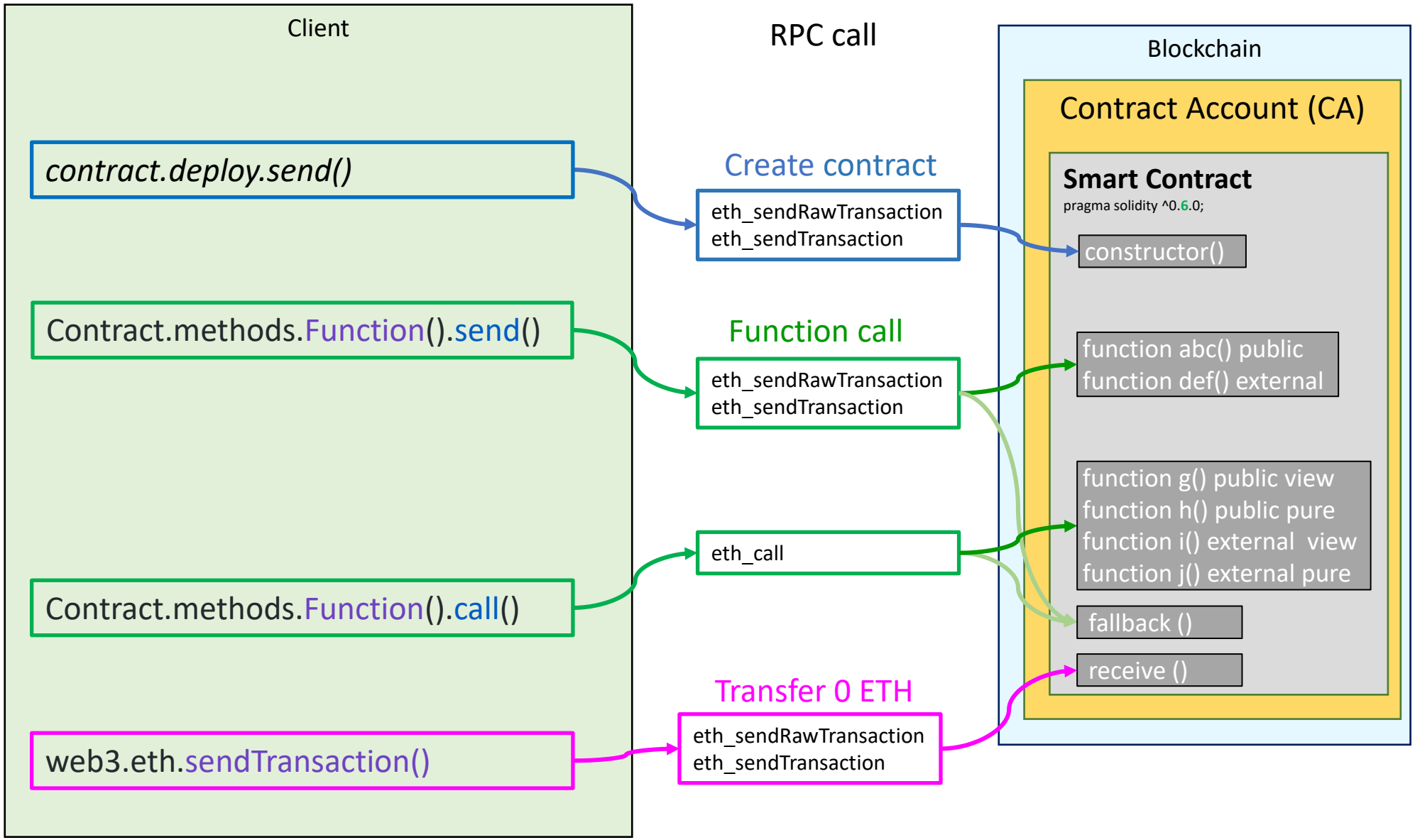
```
pragma solidity ^0.5.12;

contract TestModifier {
    address public owner;
    constructor() public {
        owner = msg.sender;
    }
    modifier onlyOwner() {
        require(msg.sender == owner, "Not owner");
        _;
    }
    modifier validAddress(address _addr) {
        require(_addr != address(0), "Not valid address");
        _;
    }
    function changeOwner(address _newOwner)
        public
        onlyOwner
        validAddress(_newOwner) {
        owner = _newOwner;
    }
} // Test with: "0x0000000000000000000000000000000000000000"
```

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

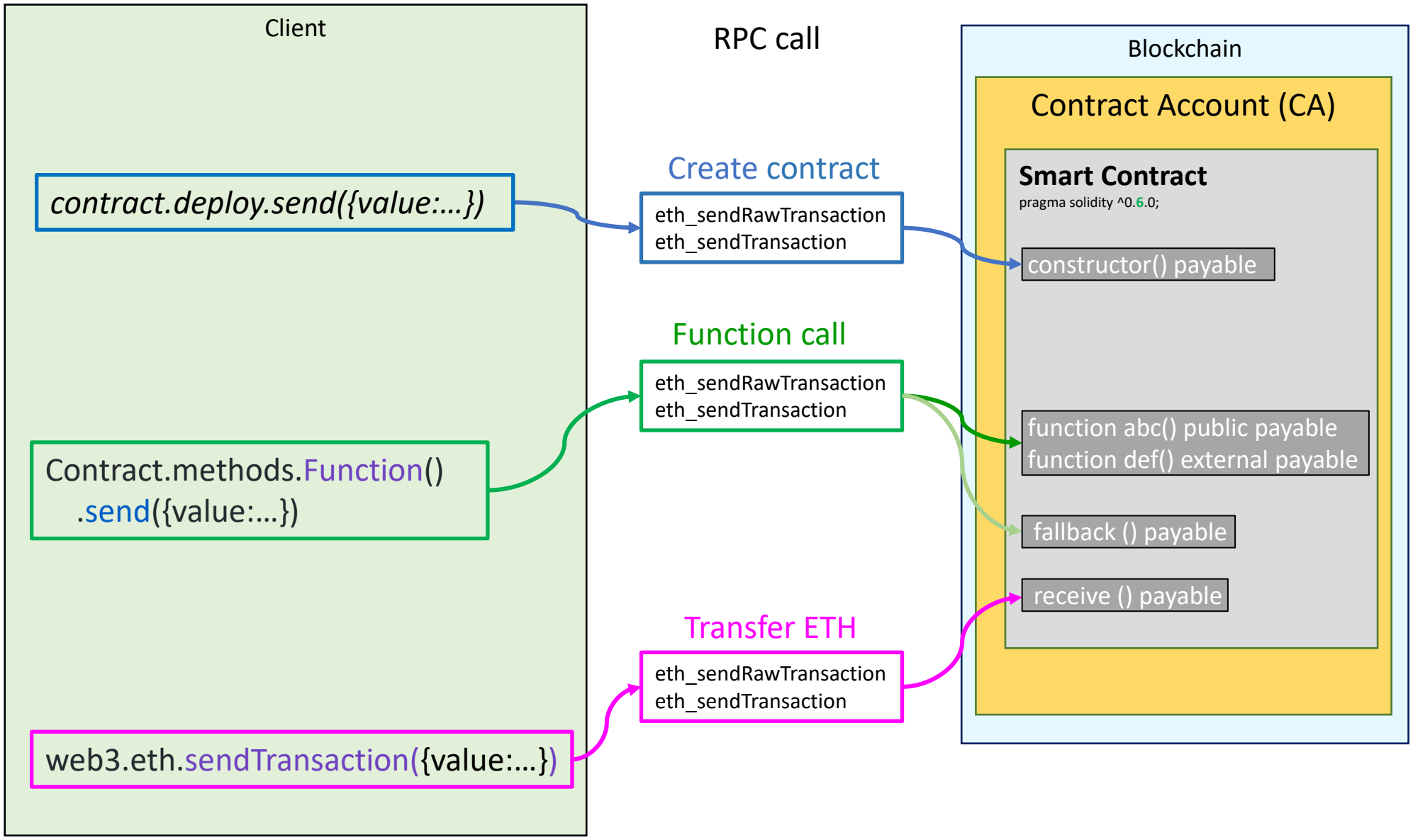
PD-3.3 Memory & Storage

- **Storage**: Where the contract state variables reside, persistent between function calls and quite expensive to use.
- **Memory**: this is used to hold temporary values. It is erased between (external) function calls and is cheaper to use.
- **Stack**: is used to hold small local variables. It is almost free to use, but can only hold a limited amount of values. Local variables (except those saved in storage by default) are saved in the stack.

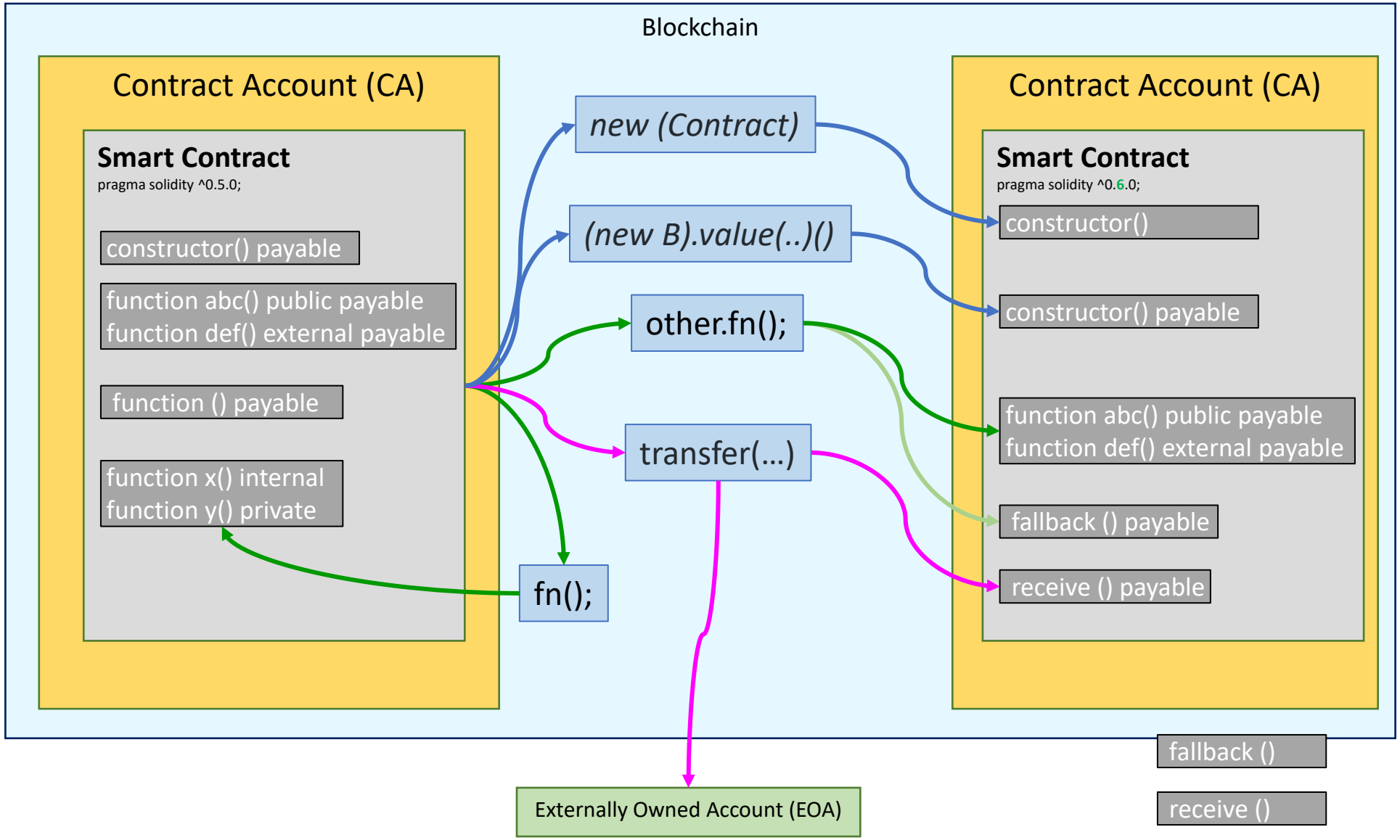
PD-3.3 Functions – without ETH transfer



PD-3.3 Functions – with ETH transfer



PD-3.3 Functions – solidity



PD-3.3 Storage pointer

```
pragma solidity ^0.5.12;

contract Test {
    uint[] aa=[1,2,3,4];

    function Add1() public {
        aa.push(1);
    }

    function Add2() public  {
        uint[] storage aa_pointer =aa;
        aa_pointer.push(2);
    }

    function GetStorage() public view returns(uint[] memory) {
        return aa;
    }
}
```

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

PD-3.3 Error handling

```
pragma solidity ^0.5.12;

contract ErrorHandling {
    uint public i;

    function testAssert1(uint j) public {
        i += j;
        assert(i >= j); // test invariant
    }
    function testAssert2(uint j) public {
        i += j;
        assert(i < j); // misuse of assert
    }
    function testRequire(uint j) public { // test input value
        require(j > 100, "j must be greater than 100");
        i += j;
    }
    function testRevert(uint j) public {
        i += j;
        revert("Undoing state changes");
    }
}
```

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

PD-3.3 Contract factory

```
// tryout in remix
pragma solidity ^0.5.12;

contract ContractFactory {
    string public whoami = "ContractFactory";
    uint ChildNr;
    function TestCreate() public returns(ChildContract) {
        ChildContract Child = new ChildContract(++ChildNr);
        return Child;
    }
    function MyAddress () public view returns (address) {
        return address(this);
    }
}
contract ChildContract {
    uint public MyId;
    string public whoami = "ChildContract";
    constructor (uint Instance) public {
        MyId = Instance;
    }
    function MyAddress () public view returns (address) {
        return address(this);
    }
}
```

http://remix.ethereum.org/

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

PD-3.3 Functions & Events

```
pragma solidity ^0.5.11;

contract Functions {

    event Log(string message,uint add, uint balance);

    constructor() public payable {
        emit Log ("In constructor",msg.value, address(this).balance);
    }

    function abc() public payable {
        emit Log("In function abc",msg.value, address(this).balance);
    }

    function() external payable {
        emit Log("In fallback function",msg.value, address(this).balance);
    }
}
```

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

PD-3.4 ToDo list

```
pragma solidity >=0.5.11;

contract ToDoList {
    struct Task { uint id;uint date; string content; string author; bool done; uint dateComplete;  }
    uint public lastTaskId;
    mapping(uint => Task) public tasks;

    modifier taskExists(uint id) { require (tasks[id].id != 0,"id not found"); _; }
    
    function createTask(string memory _content, string memory _author) public {
        lastTaskId++;
        tasks[lastTaskId] = Task(lastTaskId, now, _content, _author, false, 0);
    }

    function toggleDone_storage(uint id) taskExists(id) public {
        Task storage task = tasks[id];
        task.done = !task.done;
        task.dateComplete = task.done ? now : 0;
    }
    
    function toggleDone_memory(uint id) taskExists(id) public { // local variable: doesn't work as expected
        Task memory task = tasks[id];
        task.done = !task.done;
        task.dateComplete = task.done ? now : 0;
    }
}
```

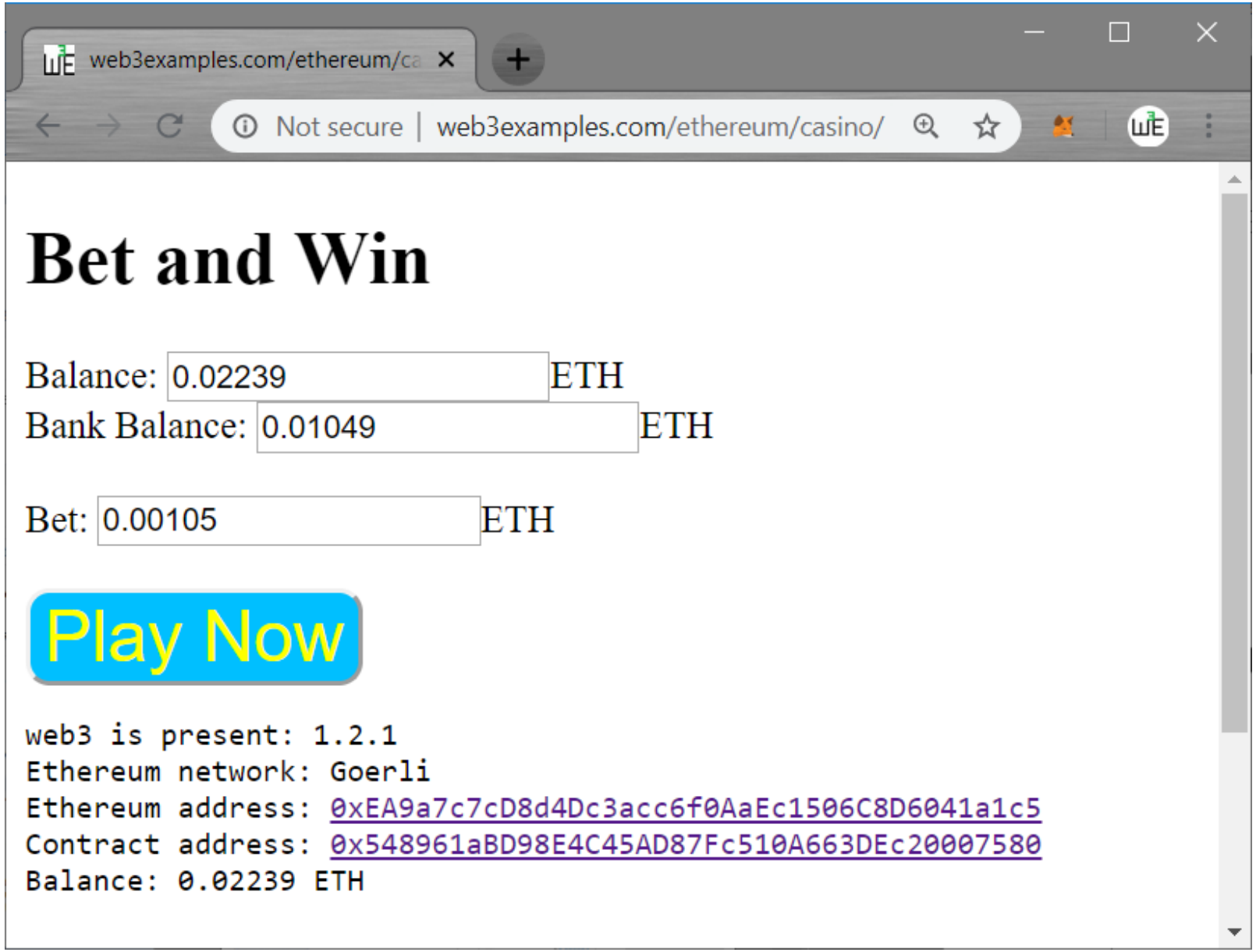
https://github.com/web3examples/ethereum/blob/master/solidity_examples/ToDoList.sol

PD-3.4 Ballot

```
pragma solidity >=0.5.11 <0.6.0;
pragma experimental ABIEncoderV2;

contract Ballot {
    struct Voter {uint vote;}
    mapping(address => Voter) public voters;
    struct Proposal {string name;uint voteCount;}
    Proposal[] public proposals;

    constructor(string[] memory proposalNames) public {
        proposals.push(Proposal({name: "not used",voteCount: 0}));
        for (uint i = 0; i < proposalNames.length; i++) {
            proposals.push(Proposal({name: proposalNames[i],voteCount: 0}));
        }
    }
    function vote(uint proposal) public {
        Voter storage sender = voters[msg.sender];
        require(proposal !=0, "Reserved");
        require(proposal < proposals.length,"Doesn't exist");
        require(sender.vote ==0, "Already voted");
        sender.vote = proposal;
        proposals[proposal].voteCount++;
    }
    function Status() public view returns (uint nrProposals, uint nrVotes) {
        nrProposals = proposals.length-1;
        for (uint p = 0; p < proposals.length; p++)
            nrVotes += proposals[p].voteCount;
    }
    function CheckwinningProposal() public view returns (uint winProposal,uint winCount,string memory winName) {
        for (uint p = 0; p < proposals.length; p++) {
            if (proposals[p].voteCount > winCount) {
                winCount = proposals[p].voteCount;
                winProposal = p;
            }
        }
        winName = proposals[winProposal].name;
    }
}
```

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

PD-3.5 Casino & debug



https://github.com/web3examples/ethereum/tree/master/casino/contracts

http://web3examples.com/ethereum/demo/Remix_ropsten_debug_gas.html

http://web3examples.com/ethereum/casino/

PD-3.5 Casino

```
pragma solidity >=0.5.0 <0.7.0;
contract Casino {
    event Won(bool win)  ;
    constructor() public payable {  }
    function betAndWin() public payable  {
        address payable betPlacer = address(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.");
        bool win = bool (getRandom()%2 == 0);
        if (win)
            betPlacer.transfer(payout);
        emit Won(win);
    }
    function getBankBalance() public view returns(uint256 ret) {
        return address(this).balance;
    }
    function getRandom() public view returns(uint256) {
        return uint256(keccak256(abi.encodePacked(block.difficulty, block.coinbase, block.timestamp)));
    }
}
```

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