

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

PD-3.1 Remix IDE - online

Remix - Ethereum IDE

Not secure | remix.ethereum.org/#optimize=false&evmVersion=null&version=soljson-v0.5.1+commit.c8a2cb62.js

FILE EXPLORERS

remix

browser

- ballot_test.sol
- ballot.sol

Home

1 tabs

The new layout has arrived

Learn more Use previous version

Environments

Solidity Vyper Workshops

Featured Plugins

Pipeline Debugger

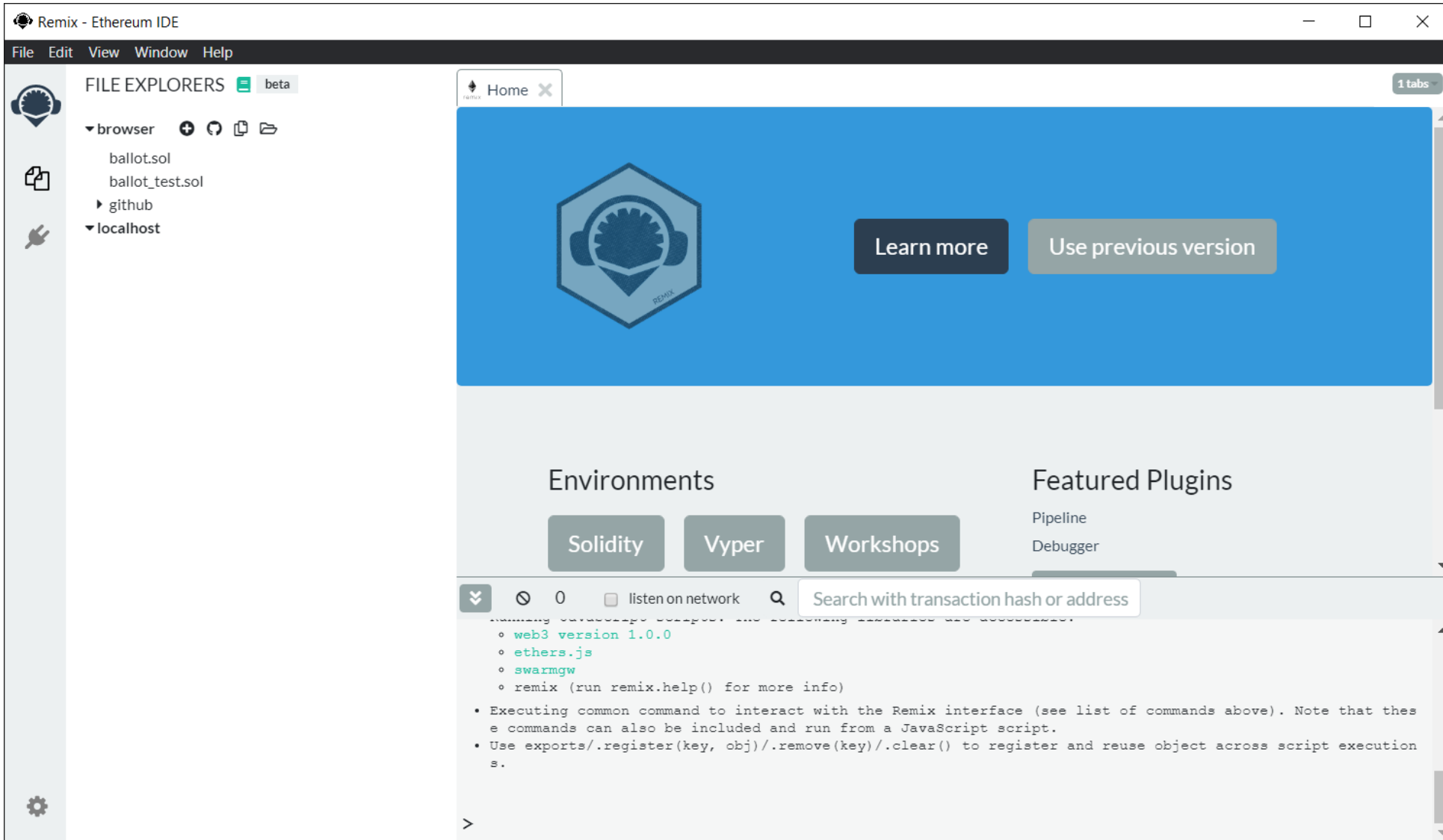
See all Plugins

File

listen on network Search with transaction hash or address

- Use exports.register(key, obj).remove(key).clear() to register and reuse object across script executions.

PD-3.1 Remix IDE – offline version



Disadvantage:
can't use Metamask

PD-3.1 Prepare Remix



PD-3.1 Remix Documentation

Welcome to Remix documentation! [Edit on GitHub](#)

Welcome to Remix documentation!

Remix is a powerful, open source tool that helps you write Solidity contracts straight from the browser. Written in JavaScript, Remix supports both usage in the browser and locally.

Remix also supports testing, debugging and deploying of smart contracts and much more.

Our Remix project with all its features is available at remix.ethereum.org and more information can be found in these docs. Our IDE tool is available at [our GitHub repository](#).

This set of documents covers instructions on how to use Remix and some tutorials to help you get started.

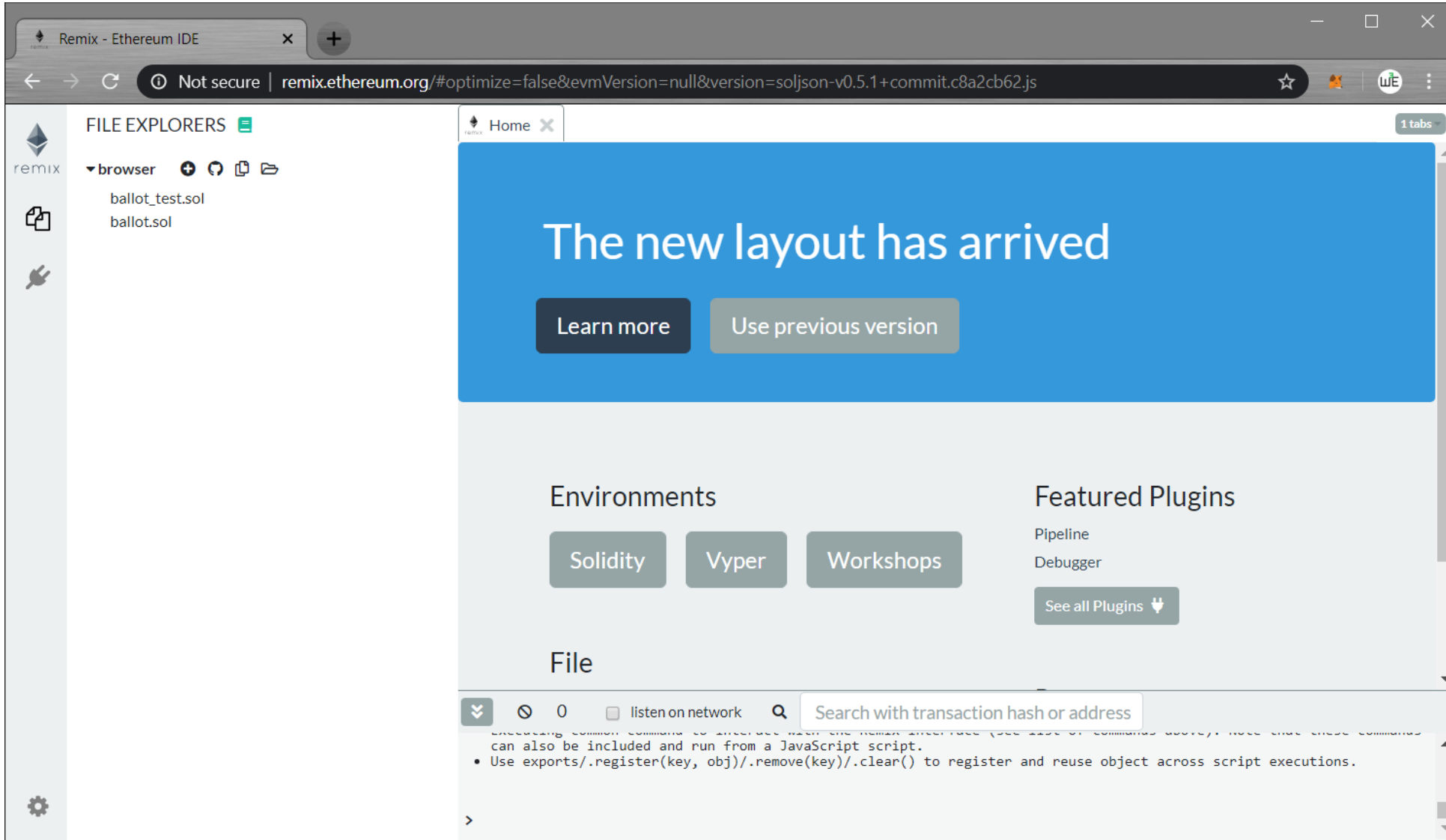
Useful links:

- [Solidity documentation](#)
- [Remix alpha](#) - The version where we test new Remix release (not stable!).
- [Remix on Medium](#)
- [Ethereum StackExchange for Remix](#)
- [Community support channel](#)
- [Deep Developer resources \(Ethereum.wiki\)](#)

PD-3.1 Remix - Workshops

The screenshot displays the Remix IDE interface. At the top, the browser address bar shows the URL `remix.ethereum.org/#optimize=false&evmVersion=null&version=soljson-v0.5.1+commit.c8a2cb62.js`. In the top right corner, the "Environments" menu is open, with "Workshops" highlighted by a green box and a green arrow pointing to the main content area. The main content area features a blue banner with the text "The new layout has arrived" and two buttons: "Learn more" and "Use previous version". Below the banner, there are sections for "Environments" (with buttons for Solidity, Vyper, and Workshops), "Featured Plugins" (with Pipeline, Debugger, and a "See all Plugins" button), and "Resources" (with Documentation and Gitter channel links). On the left side, the "Workshops" section is visible, with a "Discover Solidity" card that includes the text "Beginner workshop to discover Solidity. You will learn how to create a method and change the state." and a "Start" button. The bottom of the interface shows a search bar and a code editor with a snippet of JavaScript code: `Use exports.register(key, obj).remove(key).clear() to register and reuse object across script executions.`

PD-3.1 Remix IDE - start



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

<https://remix.ethereum.org>

PD-3.1 Remix – solidity icons

The image shows a screenshot of the Remix Ethereum IDE interface. The browser address bar displays the URL: `remix.ethereum.org/#optimize=false&evmVersion=null&version=soljson-v0.5.1+commit.c8a2cb62.js`. The main interface is divided into several sections:

- SOLIDITY COMPILER:** Located on the left, it includes settings for Compiler (0.5.1+commit.c8a2cb62), Language (Solidity), and EVM Version (compiler default). A "Compile <no file selected>" button is present.
- Compiler Configuration:** Includes checkboxes for "Auto compile", "Enable optimization", and "Hide warnings".
- Environments:** A section in the main area with buttons for "Solidity", "Vyper", and "Workshops".
- Featured Plugins:** Includes "Pipeline" and "Debugger" buttons, along with a "See all Plugins" button.
- File:** A section at the bottom with a search bar and a "listen on network" checkbox.

A callout box on the right side of the image, titled "Environments", highlights the "Solidity" button with a green border and a green arrow pointing to the main interface. The "Solidity" button in the main interface is also highlighted with a green border.

PD-3.1 Remix - file explorer

The image shows the Remix IDE interface. The browser address bar displays `remix.ethereum.org/#optimize=false&evmVersion=null&version=soljson-v0.5.1+commit.c8a2cb62.js`. The left sidebar contains the 'FILE EXPLORERS' panel, which shows a 'browser' view with two files: 'ballot_test.sol' and 'ballot.sol'. The 'ballot.sol' file is selected and its content is displayed in the main editor area. The code is a Solidity contract named 'Ballot' with the following structure:

```
1 pragma solidity >=0.4.22 <0.6.0;
2 contract Ballot {
3
4     struct Voter {
5         uint weight;
6         bool voted;
7         uint8 vote;
8         address delegate;
9     }
10    struct Proposal {
11        uint voteCount;
12    }
13
14    address chairperson;
15    mapping(address => Voter) voters;
16    Proposal[] proposals;
17
18    /// Create a new ballot with $_numProposals different proposals.
19    constructor(uint8 _numProposals) public {
20        chairperson = msg.sender;
21        voters[chairperson].weight = 1;
22        proposals.length = _numProposals;
23    }
24
25    /// Give $(toVoter) the right to vote on this ballot.
26    /// May only be called by $(chairperson).
27    function giveRightToVote(address toVoter) public {
28        if (msg.sender != chairperson || voters[toVoter].voted) return;
29        voters[toVoter].weight = 1;
30    }
}
```

The bottom of the interface shows a search bar with the text 'Search with transaction hash or address' and a list of commands for interacting with the blockchain.

Annotations: A green box highlights the 'FILE EXPLORERS' panel in the sidebar. A green arrow points from this box to the 'ballot.sol' file. Another green arrow points from the 'ballot.sol' file to the 'FILE EXPLORERS' panel. A third green arrow points from the top-right corner of the IDE to the 'FILE EXPLORERS' panel.

PD-3.1 Remix – Solidity compiler

The image shows the Remix Ethereum IDE interface. The browser address bar displays `remix.ethereum.org/#optimize=false&evmVersion=null&version=soljson-v0.5.1+commit.c8a2cb62.js`. The left sidebar contains the "SOLIDITY COMPILER" section with the following settings:

- Compiler: 0.5.1+commit.c8a2cb6
- Language: Solidity
- EVM Version: compiler default
- Buttons: **Compile ballot.sol**, Compiler Configuration, Auto compile, Enable optimization, Hide warnings
- Contract: Ballot (ballot.sol)
- Buttons: Publish on Swarm, Publish on Ipfs, Compilation Details
- Bottom: ABI, Bytecode

The main editor displays the Solidity code for `ballot.sol`:

```
1 pragma solidity >=0.4.22 <0.6.0;
2 contract Ballot {
3
4     struct Voter {
5         uint weight;
6         bool voted;
7         uint8 vote;
8         address delegate;
9     }
10    struct Proposal {
11        uint voteCount;
12    }
13
14    address chairperson;
15    mapping(address => Voter) voters;
16    Proposal[] proposals;
17
18    /// Create a new ballot with $( _numProposals ) different proposals.
19    constructor(uint8 _numProposals) public {
20        chairperson = msg.sender;
21        voters[chairperson].weight = 1;
22        proposals.length = _numProposals;
23    }
24
25    /// Give $(toVoter) the right to vote on this ballot.
26    /// May only be called by $(chairperson).
27    function giveRightToVote(address toVoter) public {
28        if (msg.sender != chairperson || voters[toVoter].voted) return;
29        voters[toVoter].weight = 1;
30    }
31
32    /// Delegate your vote to the voter $(to).
33    function delegate(address to) public {
34        Voter storage sender = voters[msg.sender]; // assigns reference
35        if (sender.voted) return;
36        while (voters[to].delegate != address(0) && voters[to].delegate != msg.sender)
37            to = voters[to].delegate;
38        if (to == msg.sender) return;
39        sender.delegate = to;
40    }
41 }
```

At the bottom of the interface, there is a search bar and a list of items:

- listen on network
- Search with transaction hash or address
- 0
- and run from a JavaScript script.
- Use `exports/.register(key, obj)/.remove(key)/.clear()` to register and reuse object across script executions.

On the right side, a green box highlights the **Compile ballot.sol** button. A green arrow points from a circular refresh icon in the top right toolbar to this button. Another green arrow points from a larger **Compile ballot.sol** button to the same button.

PD-3.1 Remix – Deploy & Run Transactions

The screenshot displays the Remix Ethereum IDE interface. The browser address bar shows the URL `remix.ethereum.org/#optimize=false&evmVersion=null&version=soljson-v0.5.1+commit.c8a2cb62.js`. The left sidebar contains the 'DEPLOY & RUN TRANSACTIONS' panel with the following settings:

- Environment: JavaScript VM
- Account: 0xCA3...a733c (99.9)
- Gas limit: 3000000
- Value: 0 wei
- Contract: Ballot - browser/ballot.sol
- Deployment method: Deploy (uint8_numProposals)

The main editor shows the Solidity code for the `Ballot` contract:

```
1 pragma solidity >=0.4.22 <0.6.0;
2 contract Ballot {
3
4     struct Voter {
5         uint weight;
6         bool voted;
7         uint8 vote;
8         address delegate;
9     }
10    struct Proposal {
11        uint voteCount;
12    }
13
14    address chairperson;
15    mapping(address => Voter) voters;
16    Proposal[] proposals;
17
18    /// Create a new ballot with $( _numProposals ) different proposals.
19    constructor(uint8 _numProposals) public {
20        chairperson = msg.sender;
21        voters[chairperson].weight = 1;
22        proposals.length = _numProposals;
23    }
24
25    /// Give $(toVoter) the right to vote on this ballot.
26    /// May only be called by $(chairperson).
27    function giveRightToVote(address toVoter) public {
28        if (msg.sender != chairperson || voters[toVoter].voted) return;
29        voters[toVoter].weight = 1;
30    }
31
32    /// Delegate your vote to the voter $(to).
33    function delegate(address to) public {
34        Voter storage sender = voters[msg.sender]; // assigns reference
35        if (sender.voted) return;
36        while (voters[to].delegate != address(0) && voters[to].delegate != msg.sender)
37            to = voters[to].delegate;
38        if (to == msg.sender) return;
```

The bottom panel shows the transaction log with the following entry:

```
[vm] from:0xca3...a733c to:Ballot.(constructor) value:0 wei data:0x608...0000 logs:0 hash:0x1d4...2bdc2
```

Green boxes highlight the browser's search, star, and profile icons, and the '2 tabs' indicator. A green arrow points from the search icon to the browser address bar.

PD-3.1 Remix – Solidity Static Analysis

The screenshot displays the Remix IDE interface. The browser address bar shows the URL: `remix.ethereum.org/#optimize=false&evmVersion=null&version=soljson-v0.5.1+commit.c8a2cb62.js`. The left sidebar, titled "SOLIDITY STATIC ANALYSIS", contains several checked options: "Block.blockhash usage: Semantics maybe unclear", "Selfdestruct: Be aware of caller contracts.", "Gas & Economy", "Gas costs: Warn if the gas requirements of functions are too high.", and "This on local calls: Invocation of local functions via this".

Under the "Results:" section, three orange warning boxes are visible:

- Gas requirement of function Ballot.delegate(address) high: infinite.**
If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
- Gas requirement of function Ballot.winningProposal() high: infinite.**
If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage)
- browser/ballot.sol:59:9:Loops that do not have a fixed number of iterations, for example, loops that depend on storage values, have to be used carefully: Due to the block gas limit, transactions can only consume a certain amount of gas. The number of iterations in a loop can grow beyond the block gas limit which can cause the complete contract to be stalled at a certain point. Additionally, using unbounded loops incurs in a lot of avoidable gas costs. Carefully test how many items at maximum you can pass to such functions to make it successful.**
[more](#)

The main editor shows the Solidity code for `ballot.sol`:

```
1 pragma solidity >=0.4.22 <0.6.0;
2 contract Ballot {
3
4     struct Voter {
5         uint weight;
6         bool voted;
7         uint8 vote;
8         address delegate;
9     }
10    struct Proposal {
11        uint voteCount;
12    }
13
14    address chairperson;
15    mapping(address => Voter) voters;
16    Proposal[] proposals;
17
18    /// Create a new ballot with $( _numProposals ) different proposals.
19    constructor(uint8 _numProposals) public {
20        chairperson = msg.sender;
21        voters[chairperson].weight = 1;
22        proposals.length = _numProposals;
23    }
24
25    /// Give $(toVoter) the right to vote on this ballot.
26    /// May only be called by $(chairperson).
27    function giveRightToVote(address toVoter) public {
28        if (msg.sender != chairperson || voters[toVoter].voted) return;
29        voters[toVoter].weight = 1;
30    }
31
32    /// Delegate your vote to the voter $(to).
33    function delegate(address to) public {
34        Voter storage sender = voters[msg.sender]; // assigns reference
35        if (sender.voted) return;
36        while (voters[to].delegate != address(0) && voters[to].delegate != msg.sender)
37            to = voters[to].delegate;
38        if (to == msg.sender) return;
```

The bottom console shows a search bar and a list of commands that can be included and run from a JavaScript script:

- Use `exports/.register(key, obj)/.remove(key)/.clear()` to register and reuse object across script executions.

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

```
1  pragma solidity^0.5.12;
2
3  contract Test {
4      ... string public StoredString="Initial string";
5      ...
6      ... function _ReadString() internal view returns (string memory) {
7          ... | return StoredString;
8          ... }
9      ... function ReadString() public view returns (string memory) {
10         ... | return StoredString;
11         ... }
12     ... function WriteString1(string memory s) public returns (string memory){
13         ... | StoredString = s;
14         ... | return _ReadString();
15         ... }
16     ... function WriteString2(string calldata s) external returns (string memory){
17         ... | return WriteString1(s);
18         ... }
19     ... function EchoString(string memory s) public pure returns (string memory){
20         ... | return s;
21         ... }
22 }
```

ReadString
StoredString
WriteString1
WriteString2
EchoString

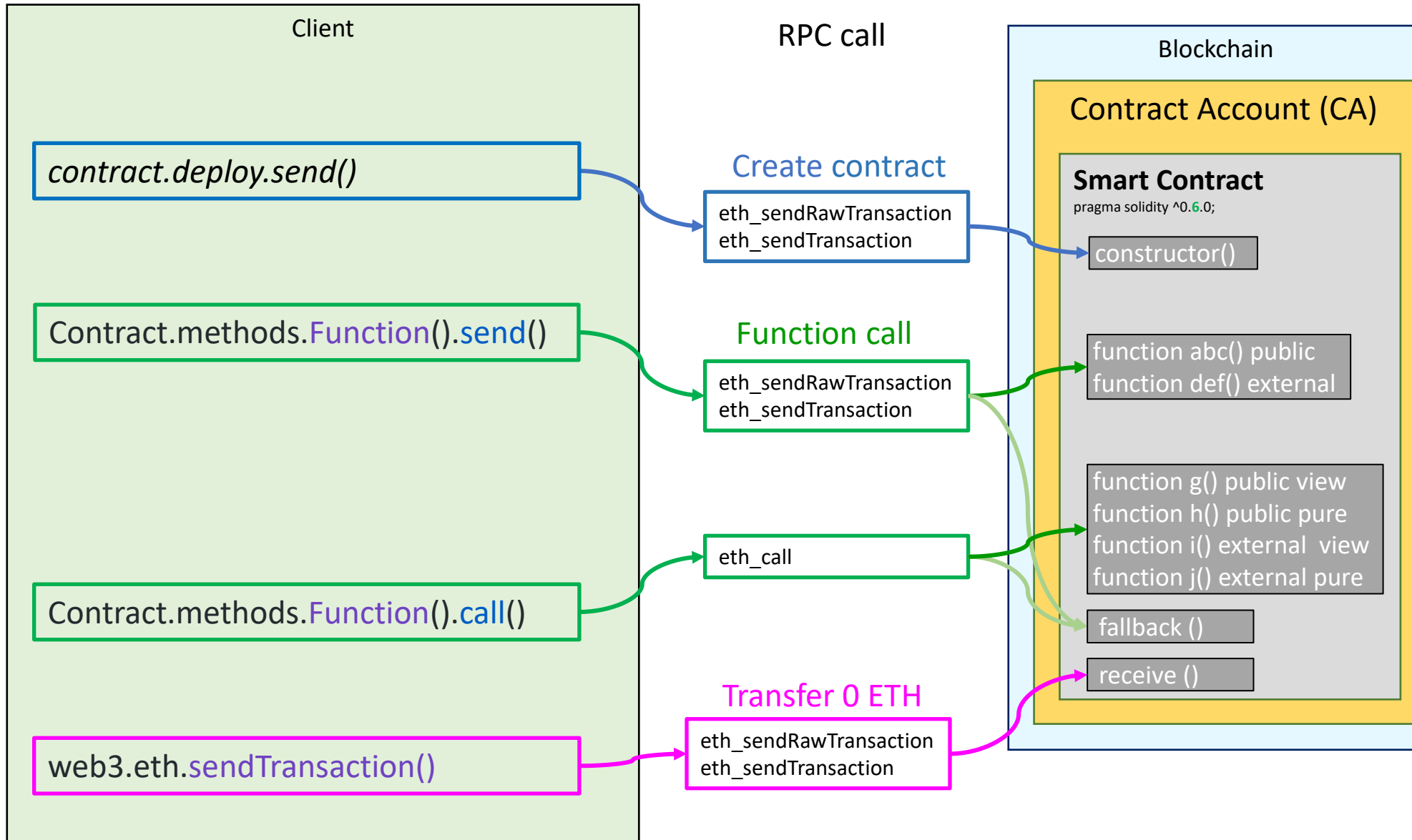
PD-3.3 Function modifier

```
1  pragma solidity ^0.5.12;
2
3  contract TestModifier {
4      address public owner;
5      constructor() public {
6          owner = msg.sender;
7      }
8      modifier onlyOwner() {
9          require(msg.sender == owner, "Not owner");
10         _;
11     }
12     modifier validAddress(address _addr) {
13         require(_addr != address(0), "Not valid address");
14         _;
15     }
16     function changeOwner(address _newOwner)
17         public
18         onlyOwner
19         validAddress(_newOwner) {
20         owner = _newOwner;
21     }
22 } // Test with: "0x0000000000000000000000000000000000000000"
```

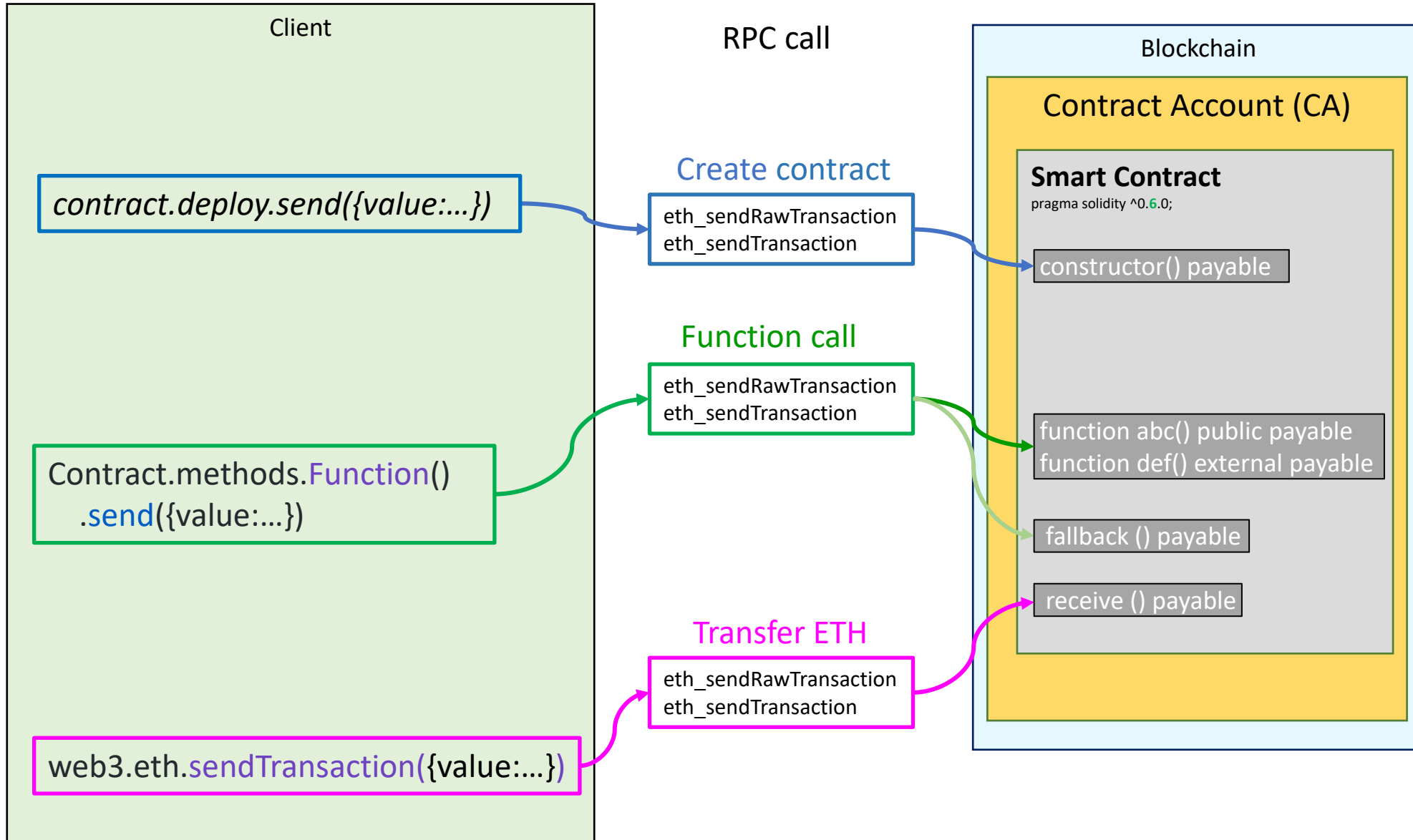
PD-3.3 Memory & Storage

- **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.

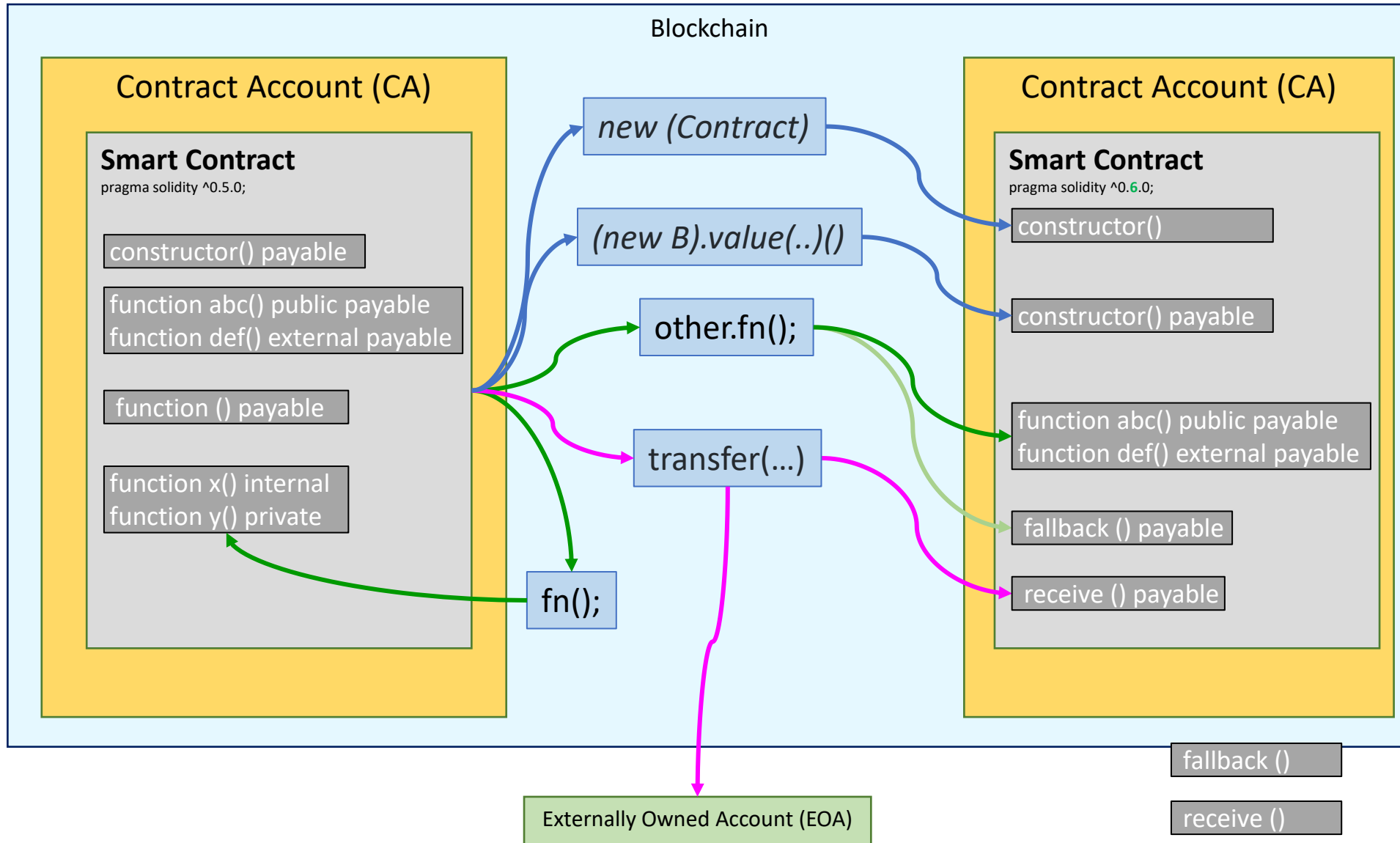
PD-3.3 Functions – without ETH transfer



PD-3.3 Functions – with ETH transfer



PD-3.3 Functions – solidity



PD-3.3 Storage pointer

```
1  pragma solidity ^0.5.12;
2
3  contract Test {
4      uint[] aa=[1,2,3,4];
5
6      function Add1() public {
7          aa.push(1);
8      }
9
10     function Add2() public {
11         uint[] storage aa_pointer =aa;
12         aa_pointer.push(2);
13     }
14
15     function GetStorage() public view returns(uint[] memory) {
16         return aa;
17     }
18 }
```

PD-3.3 Error handling

```
1  pragma solidity ^0.5.12;
2
3  contract ErrorHandling {
4      uint public i;
5
6      function testAssert1(uint j) public {
7          i += j;
8          assert(i >= j); // test invariant
9      }
10     function testAssert2(uint j) public {
11         i += j;
12         assert(i < j); // misuse of assert
13     }
14     function testRequire(uint j) public { // test input value
15         require(j > 100, "j must be greater than 100");
16         i += j;
17     }
18     function testRevert(uint j) public {
19         i += j;
20         revert("Undoing state changes");
21     }
22 }
```

PD-3.3 Contract factory

```
1 // tryout in remix
2 pragma solidity ^0.5.12;
3
4 contract ContractFactory {
5     string public whoami = "ContractFactory";
6     uint ChildNr;
7     function TestCreate() public returns (ChildContract) {
8         ChildContract Child = new ChildContract(++ChildNr);
9         return Child;
10    }
11    function MyAddress () public view returns (address) {
12        return address(this);
13    }
14 }
15 contract ChildContract {
16     uint public MyId;
17     string public whoami = "ChildContract";
18     constructor (uint Instance) public {
19         MyId = Instance;
20     }
21     function MyAddress () public view returns (address) {
22         return address(this);
23     }
24 }
```

<http://remix.ethereum.org/>

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

PD-3.3 Functions & Events

```
1  pragma solidity ^0.5.11;
2
3  contract Functions {
4
5      ... event Log(string message, uint add, uint balance);
6
7      ... constructor() public payable {
8          ... emit Log("In constructor", msg.value, address(this).balance);
9      ... }
10
11     ... function abc() public payable {
12         ... emit Log("In function abc", msg.value, address(this).balance);
13     ... }
14
15     ... function() external payable {
16         ... emit Log("In fallback function", msg.value, address(this).balance);
17     ... }
18 }
```

PD-3.4 ToDo list

```
1  pragma solidity >=0.5.11;
2
3  contract ToDoList {
4      struct Task { uint id; uint date; string content; string author; bool done; uint dateComplete; }
5      uint public lastTaskId;
6      mapping(uint => Task) public tasks;
7
8      modifier taskExists(uint id) { require (tasks[id].id != 0, "id not found"); _; }
9
10     function createTask(string memory _content, string memory _author) public {
11         lastTaskId++;
12         tasks[lastTaskId] = Task(lastTaskId, now, _content, _author, false, 0);
13     }
14
15     function toggleDone_storage(uint id) taskExists(id) public {
16         Task storage task = tasks[id];
17         task.done = !task.done;
18         task.dateComplete = task.done ? now : 0;
19     }
20
21     function toggleDone_memory(uint id) taskExists(id) public { // local variable: doesn't work as expected
22         Task memory task = tasks[id];
23         task.done = !task.done;
24         task.dateComplete = task.done ? now : 0;
25     }
26 }
```


PD-3.4 Ballot

```
1  pragma solidity >=0.5.11 <0.6.0;
2  pragma experimental ABIEncoderV2;
3
4  contract Ballot {
5      struct Voter {uint vote;}
6      mapping(address => Voter) public voters;
7      struct Proposal {string name;uint voteCount;}
8      Proposal[] public proposals;
9
10     constructor(string[] memory proposalNames) public {
11         proposals.push(Proposal({name: "not used",voteCount: 0}));
12         for (uint i = 0; i < proposalNames.length; i++) {
13             proposals.push(Proposal({name: proposalNames[i],voteCount: 0}));
14         }
15     }
16     function vote(uint proposal) public {
17         Voter storage sender = voters[msg.sender];
18         require(proposal !=0, "Reserved");
19         require(proposal < proposals.length,"Doesn't exist");
20         require(sender.vote ==0, "Already voted");
21         sender.vote = proposal;
22         proposals[proposal].voteCount++;
23     }
24     function Status() public view returns (uint nrProposals, uint nrVotes) {
25         nrProposals = proposals.length-1;
26         for (uint p = 0; p < proposals.length; p++)
27             nrVotes += proposals[p].voteCount;
28     }
29     function CheckwinningProposal() public view returns (uint winProposal,uint winCount,string memory winName) {
30         for (uint p = 0; p < proposals.length; p++) {
31             if (proposals[p].voteCount > winCount) {
32                 winCount = proposals[p].voteCount;
33                 winProposal = p;
34             }
35         }
36         winName = proposals[winProposal].name;
37     }
38 }
```

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

PD-3.5 Casino & debug

The screenshot shows a web browser window with the address bar displaying "web3examples.com/ethereum/casino/". The page content includes:

- Bet and Win** (Section Header)
- Balance: ETH
- Bank Balance: ETH
- Bet: ETH
- Play Now** (Button)
- web3 is present: 1.2.1
- Ethereum network: Goerli
- Ethereum address: [0xEA9a7c7cD8d4Dc3acc6f0AaEc1506C8D6041a1c5](#)
- Contract address: [0x548961aBD98E4C45AD87Fc510A663DEc20007580](#)
- Balance: 0.02239 ETH

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

http://web3examples.com/ethereum/debug/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)));
    }
}
```