remix.ethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.7+commit.e28d00a7.js

Home | Poll.sol | Memory.sol

```
// SPDX-License-Identifier: GPL-3.0
pragma solidity ^0.8.4;
import "hardhat/console.sol";

contract Poll {

    //0. Owner of Contract initialise with constructor
    //1. Name of Election initialise with constructor

    //2. Create Data Structure to represent Candidate
        //name, votes recieved

    //3. Create Data Structure to represent Voter
        //name, hasVoted, Vote

    //4. Candiate[]

    //5. Voter[]

    //6. Function to add Candidate to candiates only Owner
    //7. Function to add Voter to voter only Owner
    //8. Get list of Candidates
    //8. Vote sender should in voter list and should not have voted
    //9. End functionality showing result and call to selfdestruct(owner)
```

Udemy

Overview | Q&A | Notes | Announcements | Reviews | Learning tools

Student feedback

4.9

71%
25%
0%

Course content

- 22. Constructors Revisited — 2min
- 23. OOPS Inheritance — 7min
- 24. OOPS Abstract Contracts — 3min
- 25. Function Modifiers — 5min
- 26. Function Overloading — 2min
- 27. Ether Units — 1min
- 28. Understanding Gas — 3min
- 29. Gas Optimisations — 3min
- 30. Transactions — 5min
- 31. Memory Types Revisited — 7min

Section 3: Polling Smart Contract
0 / 3 | 34min

- 32. Polling Smart Contract - 1 — 17min
- 33. Polling Smart Contract -2 — 16min
- 34. Road Ahead — 1min

remix.ethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.7+commit.e28d00a7.js

Home | Poll.sol | Memory.sol

```
contract Poll {

    //0. Owner of Contract initialise with constructor
    address public owner;

    //1. Name of Election initialise with constructor
    string public electionName;

    constructor(string memory _electionName) {
        electionName = _electionName;
        owner = msg.sender;
    }

    //2. Create Data Structure to represent Candidate
        //name, votes recieved

    //3. Create Data Structure to represent Voter
        //name, hasVoted, Vote

    //4. Candiate[]
```

Play | 1.25x | 3:15 / 17:21 | ContractDefinition Poll | 1 reference(s)

Overview | Q&A | Notes | Announcements | Reviews | Learning tools

Student feedback

4.9

71%
25%
0%

Course content

- 22. Constructors Revisited — 2min
- 23. OOPS Inheritance — 7min
- 24. OOPS Abstract Contracts — 3min
- 25. Function Modifiers — 5min
- 26. Function Overloading — 2min
- 27. Ether Units — 1min
- 28. Understanding Gas — 3min
- 29. Gas Optimisations — 3min
- 30. Transactions — 5min
- 31. Memory Types Revisited — 7min

Section 3: Polling Smart Contract

0 / 3 | 34min

- 32. Polling Smart Contract - 1 — 17min
- 33. Polling Smart Contract -2 — 16min
- 34. Road Ahead — 1min

remix.ethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.7+commit.e28d00a7.js

Home | Poll.sol | Memory.sol

```
    string public electionName;

    constructor(string memory _electionName) {
        electionName = _electionName;
        owner = msg.sender;
    }

    //2. Create Data Structure to represent Candidate
        //name, votes recieved
    struct Candidate {
        string name;
        uint voteRecieved;
    }

    //3. Create Data Structure to represent Voter
        //name, hasVoted, Vote
    struct Voter {
        string voterName;
        bool hasVoted;
        uint voteIndex;
    }

    //4. Candiate[]
```

ContractDefinition Poll 1 reference(s)

Rewind 5s

Overview | Q&A | Notes | Announcements | Reviews | Learning tools

Student feedback

4.9

71%
25%
0%

Course content

- 22. Constructors Revisited — 2min
- 23. OOPS Inheritance — 7min
- 24. OOPS Abstract Contracts — 3min
- 25. Function Modifiers — 5min
- 26. Function Overloading — 2min
- 27. Ether Units — 1min
- 28. Understanding Gas — 3min
- 29. Gas Optimisations — 3min
- 30. Transactions — 5min
- 31. Memory Types Revisited — 7min

Section 3: Polling Smart Contract

0 / 3 | 34min

- 32. Polling Smart Contract - 1 — 17min
- 33. Polling Smart Contract -2 — 16min
- 34. Road Ahead — 1min

remix.ethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.7+commit.e28d00a7.js

Home | Poll.sol | Memory.sol

```
//4. Candiate[]
Candidate[] public candidates;

//5. Voter mapping (address => Voter)
mapping(address => Voter) public voterList;

//6. Function to add Candidate to candiates only Owner
modifier onlyOwner() {
    require(msg.sender == owner,'Functionality can be performed by owner only');
    _;
}
//7. Function to add Voter to voter only Owner
function addVoter(string memory _name, address voterAddress) external onlyOwner {
    Voter memory voter = Voter(_name, false, 0);
    voterList[voterAddress] = voter;
}

//8. Get list of Candidates
//8. Vote sender should in voter list and should not have voted
//9. End functionality showing result and call to selfdestruct(owner)
```

1.25x 10:39 / 17:21 ContractDefinition Poll 1 reference(s)

Overview | Q&A | Notes | Announcements | Reviews | Learning tools

Student feedback

4.9

71%
25%
0%

Course content

- 22. Constructors Revisited — 2min
- 23. OOPS Inheritance — 7min
- 24. OOPS Abstract Contracts — 3min
- 25. Function Modifiers — 5min
- 26. Function Overloading — 2min
- 27. Ether Units — 1min
- 28. Understanding Gas — 3min
- 29. Gas Optimisations — 3min
- 30. Transactions — 5min
- 31. Memory Types Revisited — 7min

Section 3: Polling Smart Contract

0 / 3 | 34min

- 32. Polling Smart Contract - 1 — 17min
- 33. Polling Smart Contract -2 — 16min
- 34. Road Ahead — 1min

remix.ethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.7+commit.e28d00a7.js

Home | Poll.sol | Memory.sol

```
mapping(address => Voter) public voterList;

//6. Function to add Candidate to candiates only Owner
modifier onlyOwner() {
    require(msg.sender == owner,'Functionality can be performed by owner only');
    _;
}

function addCandidate(string memory _name) external onlyOwner {
    Candidate memory candidate = Candidate(_name, 0);
    candidates.push(candidate);
}

//7. Function to add Voter to voter only Owner
function addVoter(string memory _name, address voterAddress) external onlyOwner {
    Voter memory voter = Voter(_name, false, 0);
    voterList[voterAddress] = voter;
}

//8. Get list of Candidates
function getCandidates() external view returns (Candidate[] memory) {
    return candidates;
}
```

ContractDefinition Poll | 1 reference(s) | r list and should not have voted

Overview | Q&A | Notes | Announcements | Reviews | Learning tools

Student feedback

4.9

71%
25%
0%

Course content

- 22. Constructors Revisited — 2min
- 23. OOPS Inheritance — 7min
- 24. OOPS Abstract Contracts — 3min
- 25. Function Modifiers — 5min
- 26. Function Overloading — 2min
- 27. Ether Units — 1min
- 28. Understanding Gas — 3min
- 29. Gas Optimisations — 3min
- 30. Transactions — 5min
- 31. Memory Types Revisited — 7min

Section 3: Polling Smart Contract

0 / 3 | 34min

- 32. Polling Smart Contract - 1 — 17min
- 33. Polling Smart Contract -2 — 16min
- 34. Road Ahead — 1min

remix.ethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.7+commit.e28d00a7.js

DEPLOY & RUN TRANSACTIONS

POLL AT 0X7EF...8CB47 (MEMORY)

addCandidate | Rahul
addVoter | string _name, address vote
candidates | uint256
electionName
0: string: GE
getCandidates
0: tuple(string,uint256)[]: Modi,0,Rahul,0
owner
voterList | address

Home | Poll.sol | Memory.sol

```
  require(msg.sender == owner,'Functionality can be performed
  _;


nction addCandidate(string memory _name) external onlyOwner {
  uint candidateIndex = candidates.length + 1;
  Candidate memory candidate = Candidate(_name, candidateIndex
  candidates.push(candidate);


7. Function to add Voter to voter only Owner
nction addVoter(string memory _name, address voterAddress) ext
  Voter memory voter = Voter(_name, false, 0);
  voterList[voterAddress] = voter;


8. Get list of Candidates
nction getCandidates() external view returns (Candidate[] memo
  return candidates;

8. Vote sender should in voter list and should not have voted
9. End functionality showing result and call to selfdestruct(o
```

string _name | 2 reference(s)

Play | 1.25x | 15:25 / 17:21

Overview | Q&A | Notes | Announcements | Reviews | Learning tools

Student feedback

4.9

★★★★★ 71%
★★★★☆ 25%
★★★☆☆ 0%

Course content

- 22. Constructors Revisited — 2min
- 23. OOPS Inheritance — 7min
- 24. OOPS Abstract Contracts — 3min
- 25. Function Modifiers — 5min
- 26. Function Overloading — 2min
- 27. Ether Units — 1min
- 28. Understanding Gas — 3min
- 29. Gas Optimisations — 3min
- 30. Transactions — 5min
- 31. Memory Types Revisited — 7min

Section 3: Polling Smart Contract
0 / 3 | 34min

- 32. Polling Smart Contract - 1 — 17min
- 33. Polling Smart Contract -2 — 16min
- 34. Road Ahead — 1min

Remix - Ethereum IDE
remix.ethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.7+commit.e28d00a7.js

Poll.sol

```
//2. Create Data Structure to represent Candidate
    //name, votes recieved
struct Candidate {
    string name;
    uint candidateIndex;
    uint voteRecieved;
    bool exsists;
}

//3. Create Data Structure to represent Voter
    //name, hasVoted, Vote
struct Voter {
    string voterName;
    bool hasVoted;
    uint voteIndex;
    bool exsists;
}

//4. Candiate[]
Candidate[] public candidates;

//5. Voter mapping (address => Voter)
```

Forward 5s

1.25x 0:47 / 15:48

Overview | Q&A | Notes | Announcements | Reviews | Learning tools

Student feedback

4.9

71%
25%
0%

Course content

- 22. Constructors Revisited — 2min
- 23. OOPS Inheritance — 7min
- 24. OOPS Abstract Contracts — 3min
- 25. Function Modifiers — 5min
- 26. Function Overloading — 2min
- 27. Ether Units — 1min
- 28. Understanding Gas — 3min
- 29. Gas Optimisations — 3min
- 30. Transactions — 5min
- 31. Memory Types Revisited — 7min

Section 3: Polling Smart Contract

1 / 3 | 34min

- 32. Polling Smart Contract - 1 — 17min
- 33. Polling Smart Contract -2 — 16min
- 34. Road Ahead — 1min

Remix - Ethereum IDE
remix.ethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.7+commit.e28d00a7.js

Poll.sol

```
    }

    function addCandidate(string memory _name) external onlyOwner {
        uint candidateIndex = candidates.length + 1;
        Candidate memory candidate = Candidate(_name, candidateIndex,0,true);
        candidates.push(candidate);
    }

    //7. Function to add Voter to voter only Owner
    function addVoter(string memory _name, address voterAddress) external onlyOwner {
        Voter memory voter = Voter(_name, false, 0, true);
        voterList[voterAddress] = voter;
    }

    //8. Get list of Candidates
    function getCandidates() external view returns (Candidate[] memory) {
        return candidates;
    }
    //8. Vote sender should in voter list and should not have voted
    function vote(uint candidateIndex) external {
        //Validate the voter is there in the voting list
        //Voter has not already voted
```

Rewind 5s

1.25x 1:12 / 15:48

Overview | Q&A | Notes | Announcements | Reviews | Learning tools

Student feedback

4.9

71%
25%
0%

Course content

- 22. Constructors Revisited — 2min
- 23. OOPS Inheritance — 7min
- 24. OOPS Abstract Contracts — 3min
- 25. Function Modifiers — 5min
- 26. Function Overloading — 2min
- 27. Ether Units — 1min
- 28. Understanding Gas — 3min
- 29. Gas Optimisations — 3min
- 30. Transactions — 5min
- 31. Memory Types Revisited — 7min

Section 3: Polling Smart Contract

1 / 3 | 34min

- 32. Polling Smart Contract - 1 — 17min
- 33. Polling Smart Contract -2 — 16min
- 34. Road Ahead — 1min

udemy | Learn Solidity in 2 Hours, Become a Blockchain Developer | Your progress | Share

Remix - Ethereum IDE
remix.ethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.7+commit.e28d00a7.js

Home | Poll.sol

```
    }

    //3. Create Data Structure to represent Voter
        //name, hasVoted, Vote
    struct Voter {
        string voterName;
        bool hasVoted;
        bool exsists;
    }

    //4. Candiate[]
    Candidate[] public candidates;

    //5. Voter mapping (address => Voter)
    mapping(address => Voter) public voterList;

    //6. Function to add Candidate to candiates only Owner
    modifier onlyOwner() {
        require(msg.sender == owner,'Functionality can be performed by owner only');
        _;
    }

    function addCandidate(string memory _name) external onlyOwner {
```

Pause | 1.25x | 1:34 / 15:48

Overview | Q&A | Notes | Announcements | Reviews | Learning tools

Student feedback

4.9

71%
25%
0%

Course content

22. Constructors Revisited — 2min
23. OOPS Inheritance — 7min
24. OOPS Abstract Contracts — 3min
25. Function Modifiers — 5min
26. Function Overloading — 2min
27. Ether Units — 1min
28. Understanding Gas — 3min
29. Gas Optimisations — 3min
30. Transactions — 5min
31. Memory Types Revisited — 7min

Section 3: Polling Smart Contract
1 / 3 | 34min

32. Polling Smart Contract - 1 — 17min
33. Polling Smart Contract -2 — 16min
34. Road Ahead — 1min

Remix - Ethereum IDE
remix.ethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.7+commit.e28d00a7.js

Poll.sol

```
    }

    function addCandidate(string memory _name) external onlyOwner {
        uint candidateIndex = candidates.length + 1;
        Candidate memory candidate = Candidate(_name, candidateIndex,0,true);
        candidates.push(candidate);
    }

    //7. Function to add Voter to voter only Owner
    function addVoter(string memory _name, address voterAddress) external onlyOwner {
        Voter memory voter = Voter(_name, false, true);
        voterList[voterAddress] = voter;
    }

    //8. Get list of Candidates
    function getCandidates() external view returns (Candidate[] memory) {
        return candidates;
    }
    //8. Vote sender should in voter list and should not have voted
    function vote(uint candidateIndex) external {
        //Validate the voter is there in the voting list
        //Voter has not already voted
```

1.25x 1:43 / 15:48

Overview | Q&A | Notes | Announcements | Reviews | Learning tools

Student feedback

4.9

71%
25%
0%

Course content

- 22. Constructors Revisited — 2min
- 23. OOPS Inheritance — 7min
- 24. OOPS Abstract Contracts — 3min
- 25. Function Modifiers — 5min
- 26. Function Overloading — 2min
- 27. Ether Units — 1min
- 28. Understanding Gas — 3min
- 29. Gas Optimisations — 3min
- 30. Transactions — 5min
- 31. Memory Types Revisited — 7min

Section 3: Polling Smart Contract

1 / 3 | 34min

- 32. Polling Smart Contract - 1 — 17min
- 33. Polling Smart Contract -2 — 16min
- 34. Road Ahead — 1min

Remix - Ethereum IDE

```
modifier onlyOwner() {
    require(msg.sender == owner,'Functionality can be performed by owner only');
    _;
}

function addCandidate(string memory _name) external onlyOwner {
    uint candidateIndex = candidates.length;
    Candidate memory candidate = Candidate(_name, candidateIndex,0,true);
    candidates.push(candidate);
}

//7. Function to add Voter to voter only Owner
function addVoter(string memory _name, address voterAddress) external onlyOwner {
    Voter memory voter = Voter(_name, false, true);
    voterList[voterAddress] = voter;
}

//8. Get list of Candidates
function getCandidates() external view returns (Candidate[] memory) {
    return candidates;
}
//8. Vote sender should in voter list and should not have voted
function vote(uint candidateIndex) external {
```

1.25x 2:11 / 15:48

Overview | Q&A | Notes | Announcements | Reviews | Learning tools

Student feedback

4.9

71%
25%
0%

Course content

- 22. Constructors Revisited — 2min
- 23. OOPS Inheritance — 7min
- 24. OOPS Abstract Contracts — 3min
- 25. Function Modifiers — 5min
- 26. Function Overloading — 2min
- 27. Ether Units — 1min
- 28. Understanding Gas — 3min
- 29. Gas Optimisations — 3min
- 30. Transactions — 5min
- 31. Memory Types Revisited — 7min

Section 3: Polling Smart Contract
1 / 3 | 34min

- 32. Polling Smart Contract - 1 — 17min
- 33. Polling Smart Contract -2 — 16min
- 34. Road Ahead — 1min

Remix - Ethereum IDE
remix.ethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.7+commit.e28d00a7.js

Poll.sol

```
function getCandidates() external view returns (Candidate[] memory) {
    return candidates;
}
//8. Vote sender should in voter list and should not have voted
function vote(uint candidateIndex) external {
    Voter storage voter = voterList[msg.sender];
    //Validate the voter is there in the voting list
    require(voter.exsists,'Voter should be present in voter list.');
    //Voter has not already voted
    require(!voter.hasVoted, 'Voter should not have already voted');

    //Candidate actually is there in the candiate list
    Candidate storage candidate = candidates[candidateIndex];
    require(candidate.exsists,'Candidate should be present in candidate list');

    candidate.voteRecieved++;
    voter.hasVoted = true;

}

//9. End functionality showing result and call to selfdestruct(owner)
function endPoll() external view {
    //for(uint index=0;index < candidates.length;index++) {
```

ContractDefinition Poll | 1 reference(s) | s[index].name,' : ',candidates[index].voteRecieved);

listen on network | Search with transaction hash or address

Overview | Q&A | Notes | Announcements | Reviews | Learning tools

Student feedback

4.9

71%
25%
0%

Course content

- 22. Constructors Revisited — 2min
- 23. OOPS Inheritance — 7min
- 24. OOPS Abstract Contracts — 3min
- 25. Function Modifiers — 5min
- 26. Function Overloading — 2min
- 27. Ether Units — 1min
- 28. Understanding Gas — 3min
- 29. Gas Optimisations — 3min
- 30. Transactions — 5min
- 31. Memory Types Revisited — 7min

Section 3: Polling Smart Contract
1 / 3 | 34min

- 32. Polling Smart Contract - 1 — 17min
- 33. Polling Smart Contract -2 — 16min
- 34. Road Ahead — 1min

udemy | Learn Solidity in 2 Hours, Become a Blockchain Developer | Your progress | Share

Remix - Ethereum IDE
remix.ethereum.org/#optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.7+commit.e28d00a7.js

Home | Poll.sol

```
67        require(voter.exsists,'Voter should be present in voter list.');
68        //Voter has not already voted
69        require(!voter.hasVoted, 'Voter should not have already voted');
70
71        //Candidate actually is there in the candiate list
72        Candidate storage candidate = candidates[candidateIndex];
73        require(candidate.exsists,'Candidate should be present in candidate list');
74
75        candidate.voteRecieved++;
76        voter.hasVoted = true;
77
78    }
79
80    //9. End functionality showing result and call to selfdestruct(owner)
81    function endPoll() external view {
82        for(uint index=0;index < candidates.length;index++) {
83            console.log(candidates[index].name,' : ',candidates[index].voteRecieved);
84        }
85    }
86
87 }
```

Play | 1.25x | 9:24 / 15:48

Overview | Q&A | Notes | Announcements | Reviews | Learning tools

Student feedback

4.9

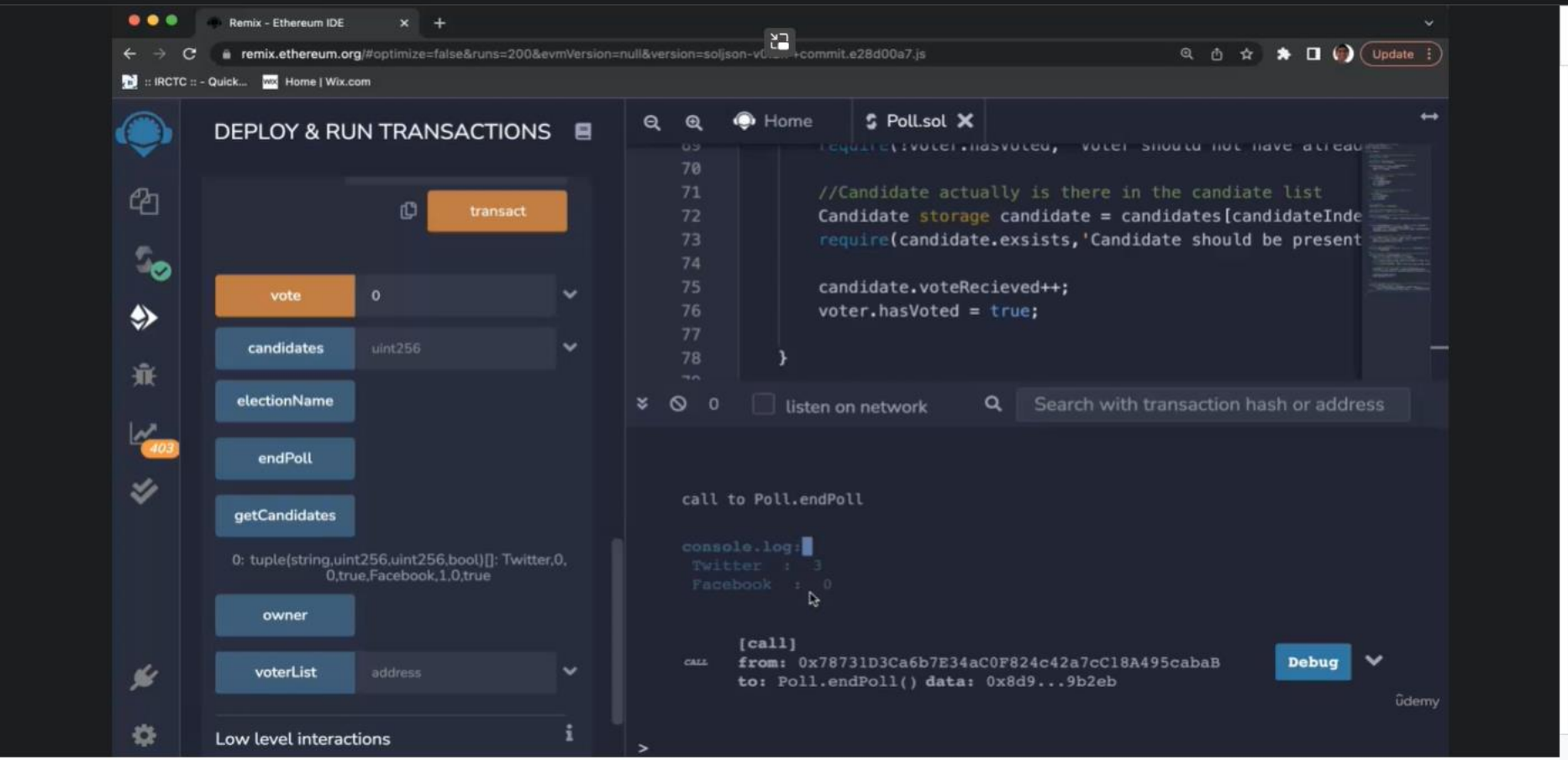
71%
25%
0%

Course content

- 22. Constructors Revisited — 2min
- 23. OOPS Inheritance — 7min
- 24. OOPS Abstract Contracts — 3min
- 25. Function Modifiers — 5min
- 26. Function Overloading — 2min
- 27. Ether Units — 1min
- 28. Understanding Gas — 3min
- 29. Gas Optimisations — 3min
- 30. Transactions — 5min
- 31. Memory Types Revisited — 7min

Section 3: Polling Smart Contract
1 / 3 | 34min

- 32. Polling Smart Contract - 1 — 17min
- 33. Polling Smart Contract -2 — 16min
- 34. Road Ahead — 1min



Overview | Q&A | Notes | Announcements | **Reviews** | Learning tools

Student feedback

4.9

★★★★★ 71%
★★★★☆ 25%
★★★☆☆ 0%

Course content

- 22. Constructors Revisited — 2min
- 23. OOPS Inheritance — 7min
- 24. OOPS Abstract Contracts — 3min
- 25. Function Modifiers — 5min
- 26. Function Overloading — 2min
- 27. Ether Units — 1min
- 28. Understanding Gas — 3min
- 29. Gas Optimisations — 3min
- 30. Transactions — 5min
- 31. Memory Types Revisited — 7min

Section 3: Polling Smart Contract
2 / 3 | 34min

- 32. Polling Smart Contract - 1 — 17min
- 33. Polling Smart Contract -2 — 16min
- 34. Road Ahead — 1min