

Background Assignment

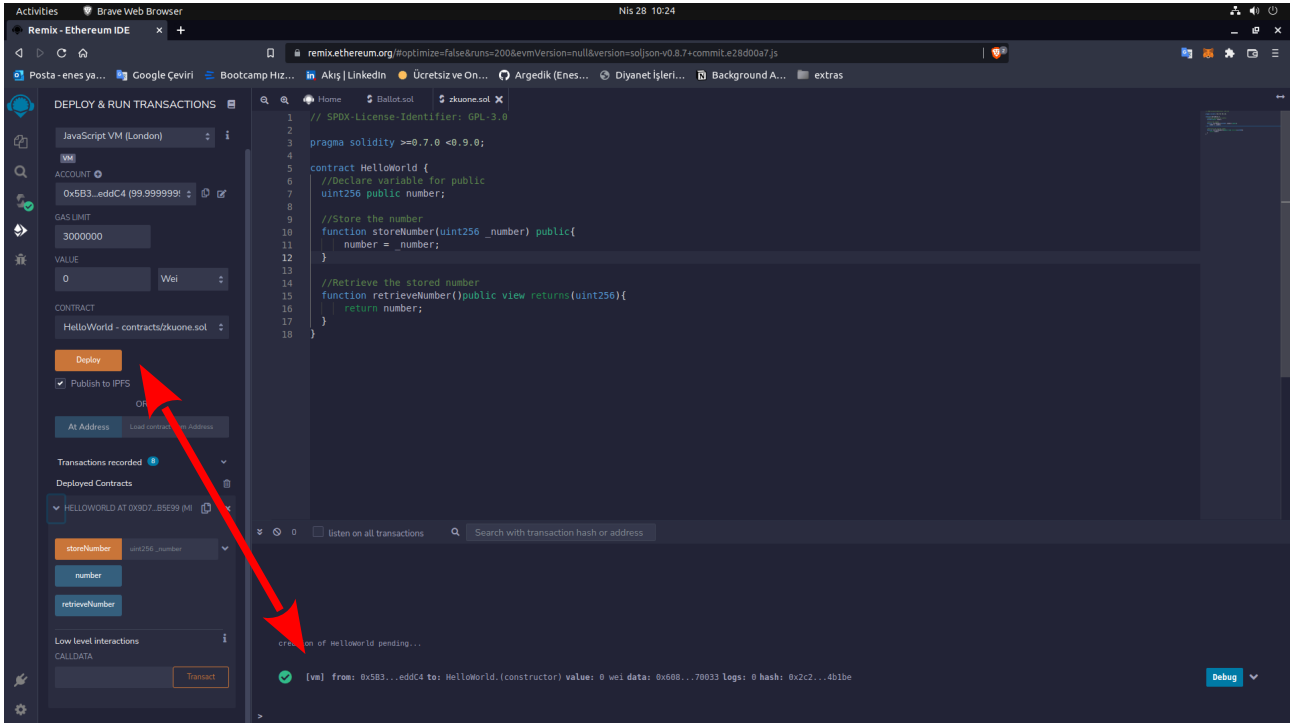
A. Conceptual Knowledge

1. What is a smart contract? How are they deployed? You should be able to describe how a smart contract is deployed and the necessary steps.
 - Smart contracts are simply programs stored on a blockchain that run when predetermined conditions are met. Remix - Remix IDE allows developing, deploying and administering smart contracts for Ethereum like blockchains
2. What is gas? Why is gas optimization such a big focus when building smart contracts?
 - Gas refers to the fee, or pricing value, required to successfully conduct a transaction or execute a contract on the blockchain platform. Miners, who perform all the important tasks of verifying and processing transactions on the network, are awarded this particular fee in return for their computational services.
3. What is a hash? Why do people use hashing to hide information?
 - A hash is a unique identifier for any given piece of content. It's also a process that takes plaintext data of any size and converts it into a unique ciphertext of a specific length. People use hashing to hide information because in the event of a compromise attackers won't get access to the plaintext passwords and there's no reason for the website to ever know the user's plaintext password.
4. How would you prove to a colorblind person that two different colored objects are actually of different colors? You could check out Avi Wigderson talk about a similar problem here.
 - Suppose 2 different objects are tangible objects. I ask him to hold both objects separately in his right and left hand. I ask him to change them behind his back and then show me again. After changing it, I can tell him which hand has the red object and which one is the green object. Even if we play this game 10000 times, I can say 100% correct because I have color knowledge. So, he will believe me that this is of different color.

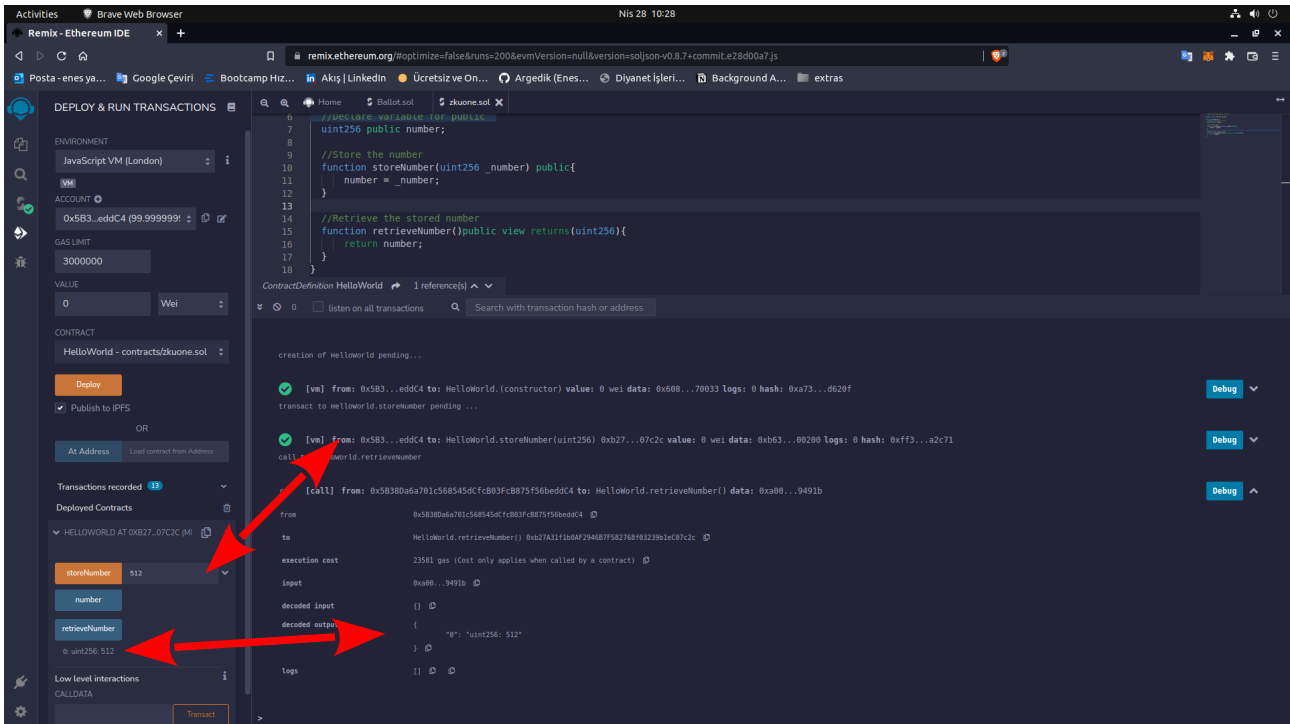
Background Assignment

B. You sure you're solid with Solidity?

Let's start by deploying the "Hello world" smart contract.

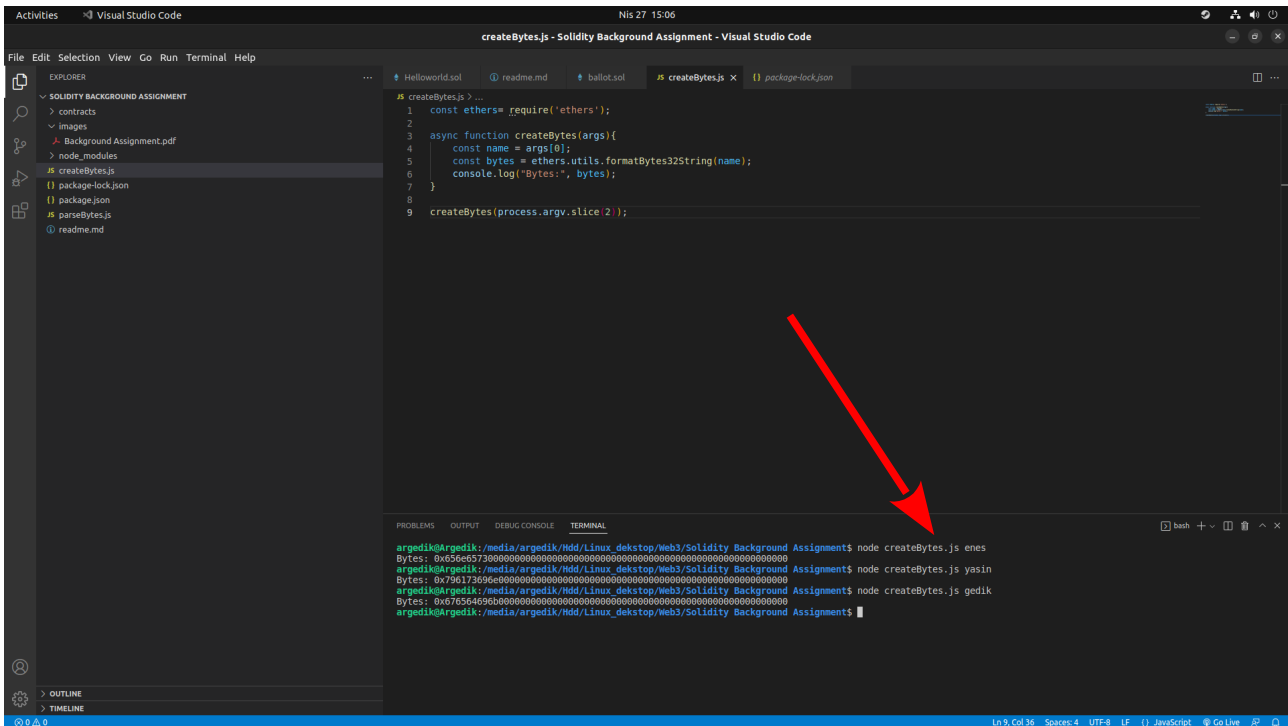


We write the "storeNumber" function to store our unsigned number. For example; "512"
We write the "retrieveNumber" function to display the unsigned number we have stored.



Background Assignment

Here we create several new wallet accounts to be able to use the "ballot contract".



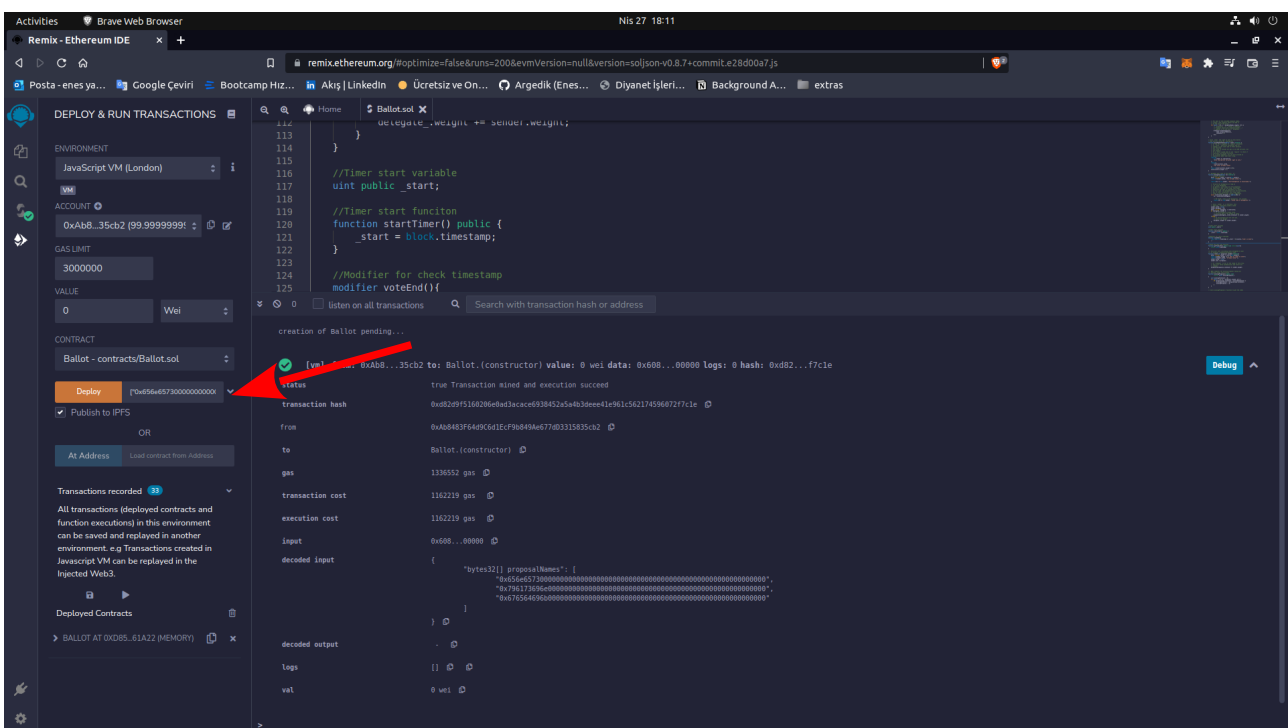
```
createBytes.js > ...
1  const ethers = require('ethers');
2
3  async function createBytes(args){
4    const name = args[0];
5    const bytes = ethers.utils.formatBytes32String(name);
6    console.log("Bytes:", bytes);
7  }
8
9  createBytes(process.argv.slice(2));
```

```
argedik@Argedik:/media/argedik/Hdd/Linux_dektop/Web3/Solidity_Background_Assignment$ node createBytes.js enes
Bytes: 0x656e657300000000000000000000000000000000000000000000000000000000
argedik@Argedik:/media/argedik/Hdd/Linux_dektop/Web3/Solidity_Background_Assignment$ node createBytes.js yasin
Bytes: 0x796173696e0000000000000000000000000000000000000000000000000000
argedik@Argedik:/media/argedik/Hdd/Linux_dektop/Web3/Solidity_Background_Assignment$ node createBytes.js gedik
Bytes: 0x676564696b0000000000000000000000000000000000000000000000000000
argedik@Argedik:/media/argedik/Hdd/Linux_dektop/Web3/Solidity_Background_Assignment$
```

After compiling our "ballot.sol" file, we enter the addresses we created in the "Deploy" section as a list.

Example;

```
["0x656e657300000000000000000000000000000000000000000000000000000000",
"0x796173696e0000000000000000000000000000000000000000000000000000",
"0x676564696b0000000000000000000000000000000000000000000000000000"]
```



```
112  }
113
114  }
115
116  //Timer start variable
117  uint public _start;
118
119  //Timer start function
120  function startTimer() public {
121    _start = block.timestamp;
122  }
123
124  //Modifier for check timestamp
125  modifier voteEnd() {
```

creation of Ballot pending...

status	transaction hash	from	to	gas	transaction cost	execution cost	input	decoded input	decoded output	logs	val
✓	0x656e657300	0xAb6...35cb2	Ballot (constructor)	136552	1162219	1162219	0x688...0000	{ "bytes32[] proposalsNames": ["0x656e657300", "0x796173696e00", "0x676564696b00"] }	-	1	0 wei

Background Assignment

We start our 5-minute period with "startTimer". We can learn when "Timer" starts with ".start" and when it ends with "getTimeLeft".

The screenshot shows the Remix IDE interface. On the left, the 'DEPLOY & RUN TRANSACTIONS' panel is open, displaying a list of transactions. The 'startTimer' transaction is highlighted, showing its details. The main editor displays the Solidity code for the 'Ballot' contract, including the 'startTimer' function. The right panel shows the transaction details, including the transaction hash, gas used, and the function call.

```
uint public _start;  
//Timer start function  
function startTimer() public {  
    _start = block.timestamp;  
}
```

Transaction details:

- From: 0x583...eddC4 to: Ballot.(constructor) value: 0 wei data: 0x608...0000 logs: 0 hash: 0x307...f4325
- Transaction hash: 0xa708f3668680f0c4e424049f4e11f980af45f9f144091c8d590b064e
- Gas: 49870 gas
- Transaction cost: 43365 gas
- Execution cost: 43365 gas
- Input: 0xa39...f7449
- Decoded input: ()
- Decoded output: ()
- Gas: 0 gas
- Val: 0 wei
- Call to: start

If we can't vote within 5 minutes according to the code we wrote in our contract, we will get the "timer is done" error.

The screenshot shows the Remix IDE interface. On the left, the 'DEPLOY & RUN TRANSACTIONS' panel is open, displaying a list of transactions. The 'vote' transaction is highlighted, showing its details. The main editor displays the Solidity code for the 'Ballot' contract, including the 'vote' function and the 'timer is done' error message. The right panel shows the transaction details, including the transaction hash, gas used, and the function call.

```
uint public _start;  
//Timer start function  
function startTimer() public {  
    _start = block.timestamp;  
}  
//Modifier for check timestamp  
modifier voteEnd(){  
    require(block.timestamp <= _start + 5 minutes,"timer is done");  
    _;  
}  
//Check the left time function  
function getTimeLeft() public view returns(uint){  
    return block.timestamp;  
}  
/// Give your vote (including votes delegated to you)  
/// to proposal 'proposals[proposal].name'.  
function vote(uint proposal) voteEnd external{  
    Voter storage sender = voters[map_sender];  
    require(sender.weight != 0, "Has no right to vote");  
    require(!sender.voted, "Already voted");  
}
```

Transaction details:

- From: 0x583...eddC4 to: Ballot.vote(uint256) 0xaE0...9688b value: 0 wei data: 0x012...00002 logs: 0 hash: 0x85c...30f63
- Transaction hash: 0xa5c8c60886a0e0f0b0e7768054100f99964459094c0e0d3c16f4445518530163
- Gas: 3000000 gas
- Transaction cost: 24217 gas
- Execution cost: 24217 gas
- Input: 0x012...00002
- Decoded input: {"uint256 proposal": "2"}

Background Assignment

Let's vote in less than 5 minutes :)

The screenshot displays the Remix Ethereum IDE interface. On the left, the 'DEPLOY & RUN TRANSACTIONS' panel shows a deployed contract named 'BALLOT' at address 0x91...39138. The contract's state is visible, including a 'chairperson' field with address 0x58380a6a701c368545dcf803fc8b75f56beddc4, a 'getTimelLeft' field with value 165112974, a 'proposals' array with one element, and a 'winningProposal' field with value 2. A red arrow points from the 'start' button in the 'DEPLOY & RUN TRANSACTIONS' panel to the 'start' button in the 'BALLOT' contract's 'start' function. The main panel shows the contract's source code, which includes a 'start' function that calls 'startTimer', 'startTimer' which calls 'start', and 'vote' which calls 'vote'. The right panel shows the transaction logs, which include the following entries:

- [vm] from: 0x583...eddC4 to: Ballot.(constructor) value: 0 wei data: 0x608...00000 Logs: 0 hash: 0xdfb...d7e0
- transaction to Ballot.startTimer pending ...
- [vm] from: 0x583...eddC4 to: Ballot.startTimer() 0xd91...39138 value: 0 wei data: 0xa39...f7449 Logs: 0 hash: 0x954...7c077
- call to Ballot.start
- [call] from: 0x58380a6a701c368545dcf803fc8b75f56beddc4 to: Ballot.start() data: 0x53e...f6781
- call to Ballot.getTimelLeft
- [call] from: 0x58380a6a701c368545dcf803fc8b75f56beddc4 to: Ballot.getTimelLeft() data: 0xc7e...284b8
- transaction to Ballot.vote pending ...
- [vm] from: 0x583...eddC4 to: Ballot.vote(uint256) 0xd91...39138 value: 0 wei data: 0x012...00002 Logs: 0 hash: 0x910...16b68
- call to Ballot.voters
- [call] from: 0x58380a6a701c368545dcf803fc8b75f56beddc4 to: Ballot.voters(address) data: 0xa3e...eddC4
- call to Ballot.winningProposal
- [call] from: 0x58380a6a701c368545dcf803fc8b75f56beddc4 to: Ballot.winningProposal() data: 0xe2b...a53f8
- call to Ballot.winningProposal
- [call] from: 0x58380a6a701c368545dcf803fc8b75f56beddc4 to: Ballot.winningProposal() data: 0xe09...ff1bd
- call to Ballot.chairperson
- [call] from: 0x58380a6a701c368545dcf803fc8b75f56beddc4 to: Ballot.chairperson() data: 0x2e4...176cf
- call to Ballot.proposals
- [call] from: 0x58380a6a701c368545dcf803fc8b75f56beddc4 to: Ballot.proposals(uint256) data: 0x013...00000

<https://github.com/Argedik/ZKU-ONE-Background-Assignment>