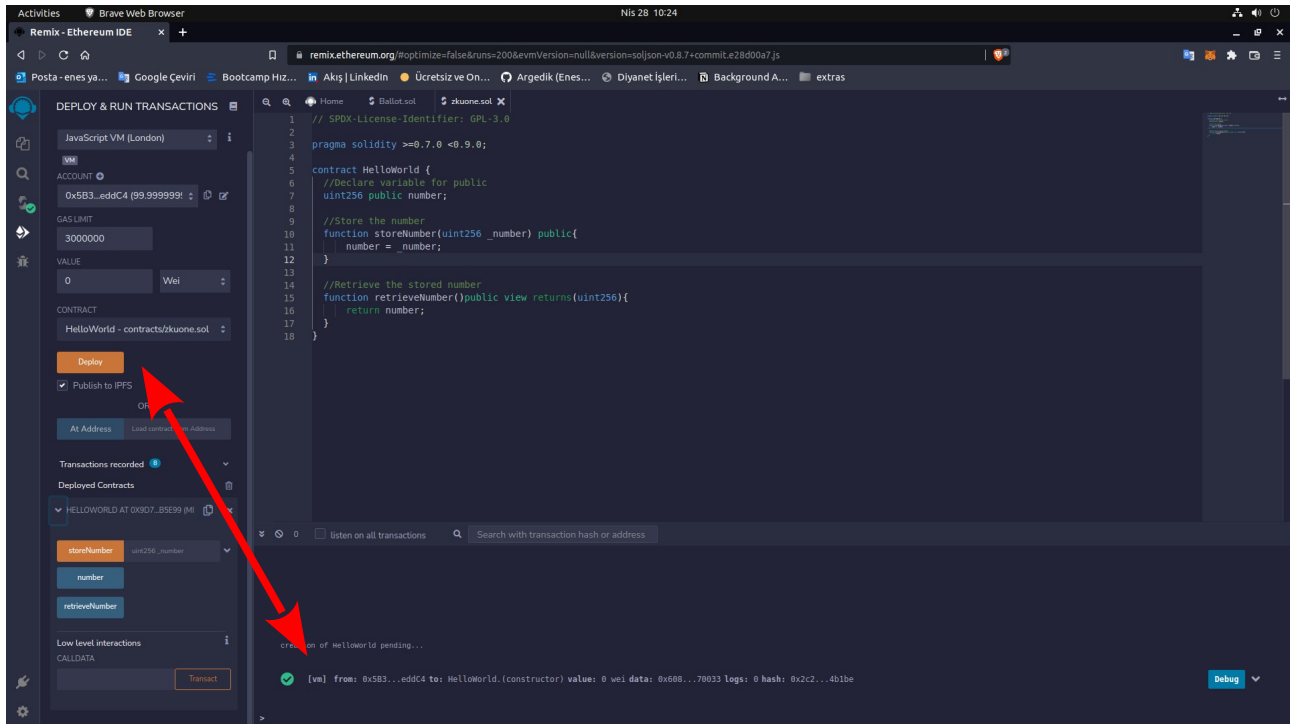
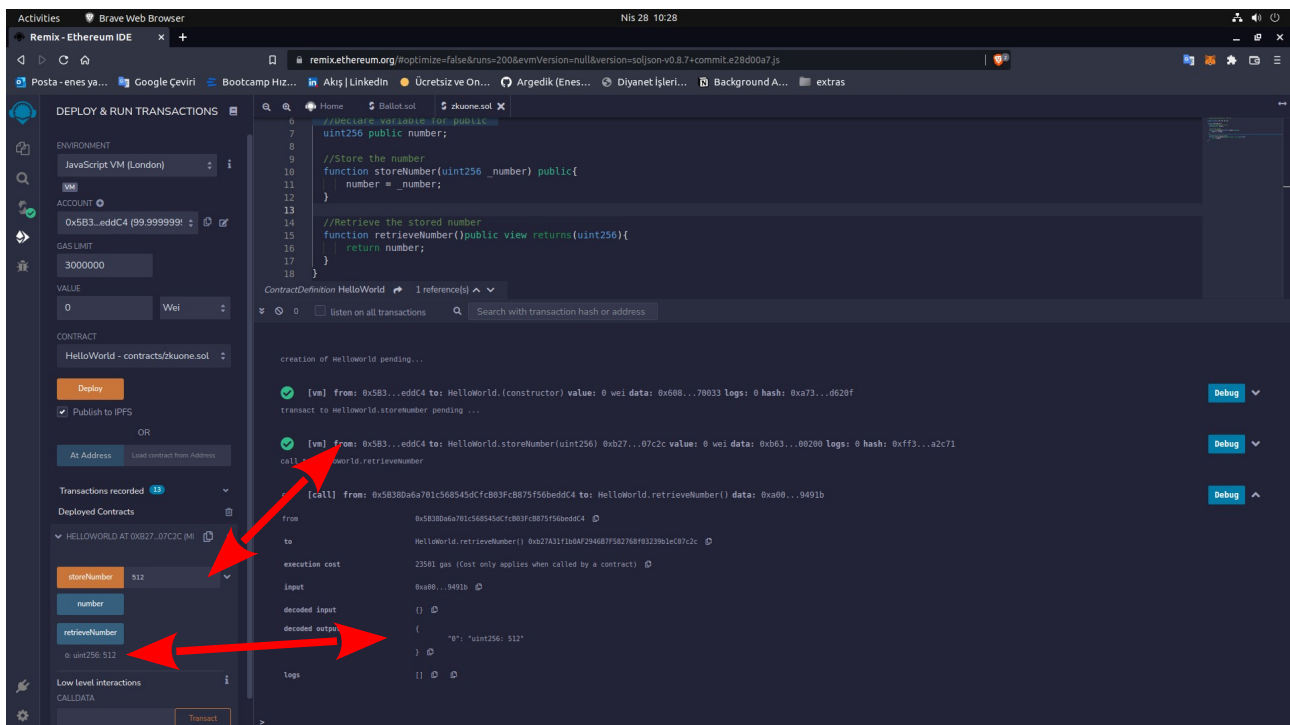


Background Assignment

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_dekstop/Web3/Solidity_Background_Assignment$ node createBytes.js enes
Bytes: 0x056e573000000000000000000000000000000000000000000000000000000000
argedik@Argedik:/media/argedik/Hdd/Linux_dekstop/Web3/Solidity_Background_Assignment$ node createBytes.js yasin
Bytes: 0x796173696e0000000000000000000000000000000000000000000000000000
argedik@Argedik:/media/argedik/Hdd/Linux_dekstop/Web3/Solidity_Background_Assignment$ node createBytes.js gedik
Bytes: 0x676564696b00000000000000000000000000000000000000000000000000
```

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

Example;

```
["0x656e573000000000000000000000000000000000000000000000000000000000",
"0x796173696e0000000000000000000000000000000000000000000000000000",
"0x676564696b00000000000000000000000000000000000000000000000000"]
```

```
creation of Ballot pending...
[wei] 0 wei data: 0x608...0000 logs: 0x82...7c1e
true Transaction mined and execution succeed
Transaction hash
0xd207f160706b8ad1acae8938452a5a4b36ee41e9b15621749087277c1e
from
0xA0843f649C641eCf9B849a6f7d03115835c2
to
Ballot (constructor)
gas
130652 gas
transaction cost
1162219 gas
execution cost
1162219 gas
input
0x608...0000
decoded input
{"bytes32[]": [{"name": "enes"}, {"name": "yasin"}, {"name": "gedik"}]}
decoded output
-
logs
[]
val
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 Ethereum IDE interface. On the left, the 'DEPLOY & RUN TRANSACTIONS' panel is visible, showing a list of transactions for the 'Ballot' contract. The 'startTimer' transaction is highlighted. The main editor displays the 'Ballot.sol' contract code, which includes a 'startTimer' function. The right panel shows the 'StructDefinition Proposal' and a list of transactions. The 'startTimer' transaction is highlighted, showing its details and logs. Red arrows point from the 'startTimer' transaction in the left panel to the 'startTimer' function in the code and the transaction details in the right panel.

Let's vote in less than 5 minutes :)

The screenshot shows the Remix Ethereum IDE interface. On the left, the 'DEPLOY & RUN TRANSACTIONS' panel is visible, showing a list of transactions for the 'Ballot' contract. The 'vote' transaction is highlighted. The main editor displays the 'Ballot.sol' contract code, which includes a 'vote' function. The right panel shows the 'StructDefinition Proposal' and a list of transactions. The 'vote' transaction is highlighted, showing its details and logs. Red arrows point from the 'vote' transaction in the left panel to the 'vote' function in the code and the transaction details in the right panel.