Lesson 4 - Tests and Scripts

Writing unit tests for Ballot.sol

- More on Ether.js functions and utilities
- Bytes32 and Strings conversion
- Helper functions inside test scripts
- Unit tests structure and nested tests

Code reference for tests

```
import { expect } from "chai";
import { ethers } from "hardhat";
// eslint-disable-next-line node/no-missing-import
import { Ballot } from "../../typechain";

const PROPOSALS = ["Proposal 1", "Proposal 2", "Proposal 3"];

function convertStringArrayToBytes32(array: string[]) {
  const bytes32Array = [];
  for (let index = 0; index < array.length; index++) {
    bytes32Array.push(ethers.utils.formatBytes32String(array[index]));
  }
  return bytes32Array;
}

describe("Ballot", function () {
  let ballotContract: Ballot;

  this.beforeEach(async function () {
    const ballotFactory = await ethers.getContractFactory("Ballot");
    ballotContract = await ballotFactory.deploy(
      convertStringArrayToBytes32(PROPOSALS)
    );
    await ballotContract.deployed();
  });

  describe("when the contract is deployed", function () {
    it("has the provided proposals", async function () {
      for (let index = 0; index < PROPOSALS.length; index++) {
        const proposal = await ballotContract.proposals(index);
        expect(ethers.utils.parseBytes32String(proposal.name)).to.eq(
          PROPOSALS[index]
        );
      }
    });

    it("has zero votes for all proposals", async function () {});
```

```
    it("sets the deployer address as chairperson", async function () {});
    it("sets the voting weight for the chairperson as 1", async function
() {});
  });

  describe("when the chairperson interacts with the giveRightToVote
function in the contract", function () {
    it("gives right to vote for another address", async function () {});
    it("can not give right to vote for someone that has voted", async
function () {});
    it("can not give right to vote for someone that has already voting
rights", async function () {});
  });

  describe("when the voter interact with the vote function in the
contract", function () {});

  describe("when the voter interact with the delegate function in the
contract", function () {});

  describe("when the an attacker interact with the giveRightToVote
function in the contract", function () {});

  describe("when the an attacker interact with the vote function in the
contract", function () {});

  describe("when the an attacker interact with the delegate function in
the contract", function () {});

  describe("when someone interact with the winningProposal function before
any votes are cast", function () {});

  describe("when someone interact with the winningProposal function after
one vote is cast for the first proposal", function () {});

  describe("when someone interact with the winnerName function before any
votes are cast", function () {});

  describe("when someone interact with the winnerName function after one
vote is cast for the first proposal", function () {});

  describe("when someone interact with the winningProposal function and
winnerName after 5 random votes are cast for the proposals", function ()
{});
});
```

Using scripts to automate operations

- Running a script with yarn and node, ts-node and/or hardhat
- Ballot deployment script

- Passing arguments
- Passing variables to the deployment script
- Environment files
- Providers
- Connecting to a testnet with a RPC Provider
- Running scripts on chain
- Script for giving voting rights to a given address
- Dealing with transactions in scripts

References

https://www.typescripttutorial.net/typescript-tutorial/typescript-hello-world/

https://nodejs.org/docs/latest/api/process.html#processargv

https://docs.ethers.io/v5/api/providers/

https://docs.ethers.io/v5/api/contract/contract-factory/

```
import { ethers } from "hardhat";

const PROPOSALS = ["Proposal 1", "Proposal 2", "Proposal 3"];

async function main() {
  console.log("Deploying Ballot contract");
  console.log("Proposals: ");
  PROPOSALS.forEach((element, index) => {
    console.log(`Proposal N. ${index + 1}: ${element}`);
  });
  // TODO
}

main().catch((error) => {
  console.error(error);
  process.exitCode = 1;
});
```

Running scripts

```
yarn run hardhat .\scripts\Ballot\deployment.ts
```

Running scripts with arguments

```
yarn run ts-node --files .\scripts\Ballot\deployment.ts "arg1" "arg2"
"arg3
```

Hardhat commands

```
npx hardhat accounts
npx hardhat compile
npx hardhat clean
npx hardhat test
npx hardhat node
npx hardhat help
REPORT_GAS=true npx hardhat test
npx hardhat coverage
npx hardhat run scripts/deploy.ts
TS_NODE_FILES=true npx ts-node scripts/deploy.ts
npx eslint '**/*.{js,ts}'
npx eslint '**/*.{js,ts}' --fix
npx prettier '**/*.{json,sol,md}' --check
npx prettier '**/*.{json,sol,md}' --write
npx solhint 'contracts/**/*.sol'
npx solhint 'contracts/**/*.sol' --fix
```

Performance optimizations

For faster runs of your tests and scripts, consider skipping ts-node's type checking by setting the environment variable `TS_NODE_TRANSPILE_ONLY` to `1` in hardhat's environment. For more details see the documentation.

Etherscan verification

To try out Etherscan verification, you first need to deploy a contract to an Ethereum network that's supported by Etherscan, such as Ropsten.

In this project, copy the .env.example file to a file named .env, and then edit it to fill in the details. Enter your Etherscan API key, your Ropsten node URL (eg from Alchemy), and the private key of the account which will send the deployment transaction. With a valid .env file in place, first deploy your contract:

```
hardhat run --network ropsten scripts/deploy.ts
```

Then, copy the deployment address and paste it in to replace `DEPLOYED_CONTRACT_ADDRESS` in this command: The constructor args "Pizza" "Lasagna" "Icecream" translates to the array: ['0x50697a7a6100000000000000000000000000000000000000000000000000000000', '0x4c617361676e6100000000000000000000000000000000000000000000000000', '0x496365637265616d0000000000000000000000000000000000000000000000000']

```
yarn hardhat verify --network ropsten --constructor-args
scripts/Ballot/constructorArgs.ts DEPLOYED_CONTRACT_ADDRESS
```

Local deployment

You can use Hardhat Network in order to have a local Ethereum network node for development.

To do so :

1. Put this in your .env (default mnemonic for hardhat network accounts) `MNEMONIC=test test test test test test test test test test test junk`
2. Run local node `yarn hardhat node`

Homework

- Create Github Issues with your questions about this lesson
- Read the references
- Finish covering other operations with scripts

Weekend Project

- Form groups of 3 to 5 students
- Structure scripts to
    - Deploy
    - Query proposals
    - Give vote right passing an address as input
    - Cast a vote to a ballot passing contract address and proposal as input and using the wallet in environment
    - Delegate my vote passing user address as input and using the wallet in environment
    - Query voting result and print to console
- Publish the project in Github
- Run the scripts with a set of proposals, cast and delegate votes and inspect results
- Write a report detailing the addresses, transaction hashes, description of the operation script being executed and console output from script execution for each step (Deployment, giving voting rights, casting/delegating and querying results).
- (Extra) Use TDD methodology

See REPORT for solutions.