Creating Smart Contracts with Solidity

Adding UI to a smart contract

Building a dapp

User interfaces for smart contracts

- We can write any smart contract, but requiring user to prepare transactions and send them manually to invoke functions on the smart contract is very inconvenient
- MetaMask helps a lot with that , but still requiring users to enter function arguments is really poor UX
- Fortunately since MetaMask is also browser extension and gets injected on all pages, we can access it and use it from a HTML app

Web3

- Web3 is the de-facto standard for interacting with Ethereum blockchain from Javascript and especialy from the browser
- Can be used standalone or injected by MetaMask
- More information and documentation at https://web3js.org/#/

Prerequisites for using Web3

- The contract needs to be deployed
- We need to have the contract ABI
  - This ABI is a JSON description of the contract functions and the parameters they take, so that they can be called from JavaScript
  - It can be found in artifacts/contracts/ directory after successful compilation under the abi key of the *ContractName.json* file

Deploying a contract to a network

- We’ve been deploying contracts with the ChainIDE or in the test environment
- Hardhat provides deploy scripts, where you can deploy a contract to a network using JavaScript helpers similar to the ones we are using in the automated tests
- These scripts should be places in the scripts/ directory and can be executed with npx hardhat run scripts/*ScriptName.js*

Web3 initialization

- We must import the Web3 library in our page
- Then we will check for injected Web3 provider
- Then we are going to initialize the Web3 library
- Then we load the contract by using it’s address from the deploy step and the ABI from the JSON file with compiled contract
- After that we can use the initialized contract to call contract functions

Example

```
if (window.ethereum) {
        web3 = new Web3(window.ethereum);
        window.ethereum.enable();
} else { /* MetaMask not installed */}

const abi = [....]; // ABI JSON contents
const contractAddress = '0x89D8e......';

const contract =  new web3.eth.Contract(abi,contractAddress);

await contract.methods.balanceOf(account).call();
```

Let’s build