Blockchain Programming

ESILV 2023-2024

Agenda

TD4: Ethereum

Creating an ICO

ICO

- "An initial coin offering (ICO) or initial currency offering is a type of funding using cryptocurrencies”
- Send in Ethers, receive tokens
- Different levels of contributors (early contribs, VCs, public offers etc)
- Different rewards
- Risks & Rewards: High potential for returns, but also high risk. Many ICOs have resulted in significant profits for early investors, but some have also been scams or failed projects.
- Regulation: Initially, ICOs were less regulated, leading to more risks. Today, many countries have guidelines or regulations around ICOs to protect investors

ERC20: Make money money

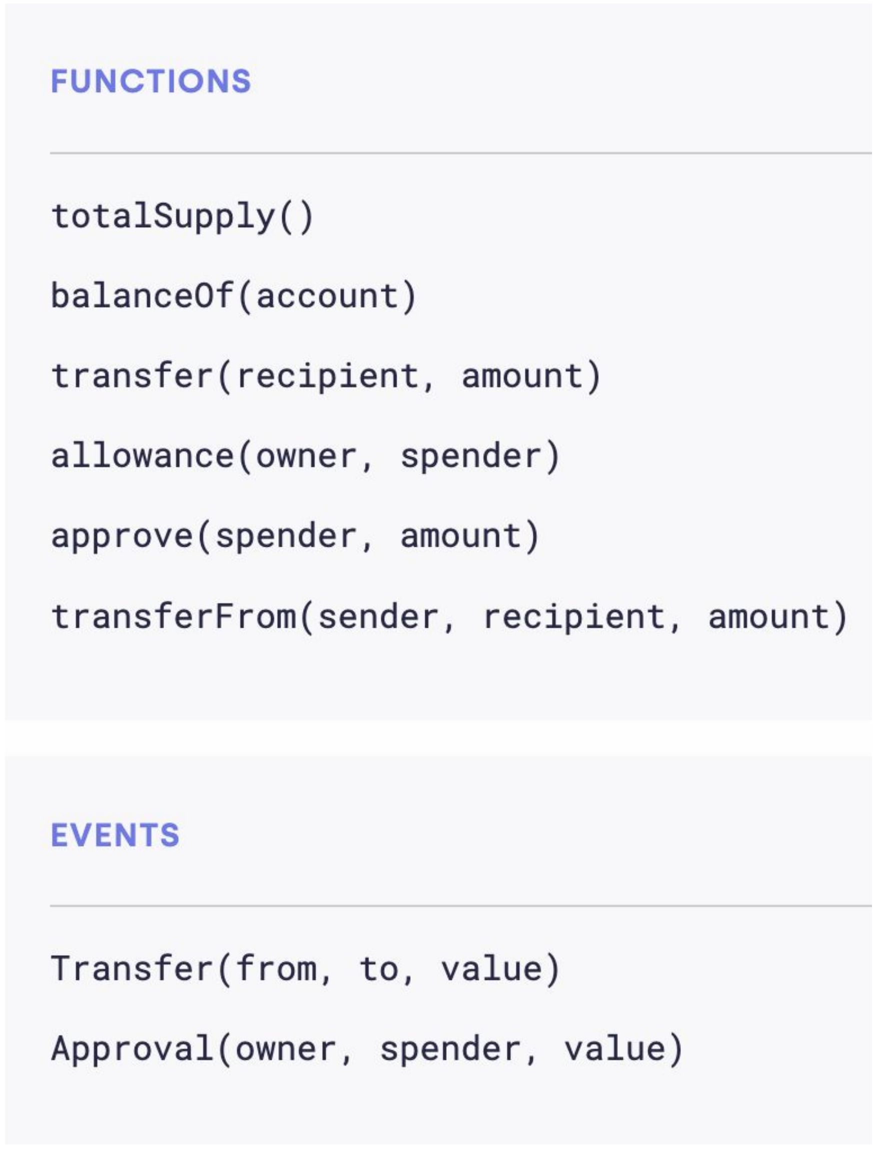
- A standard for token creation
- https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20.md
- A simple interfacer to create, exchange and manipulate tokens
- Adopted by most ICOs
- Widely used to list tokens on exchanges

Famous standards

ERC20

https://eips.ethereum.org/EIPS/eip-20

- Standard interface for fungible tokens on Ethereum
- Currencies, shares etc…
- DeFi key element



FUNCTIONS

```
totalSupply()

balanceOf(account)

transfer(recipient, amount)

allowance(owner, spender)

approve(spender, amount)

transferFrom(sender, recipient, amount)
```

EVENTS

```
Transfer(from, to, value)

Approval(owner, spender, value)
```

Foundry

We can do better !

- Foundry is a portable, fast and modular toolkit for Ethereum application development
- Foundry manages the dependencies, compiles the project, runs tests, deploys, and let us interact with the chain from the command-line and via Solidity scripts (Forge, Cast, Anvil).
- The big difference (beside the efficiency) is that we write everything in solidity including the tests. One language for everything

Paradigm

https://twitter.com/gakonst

FORGE

Writing tests in solidity

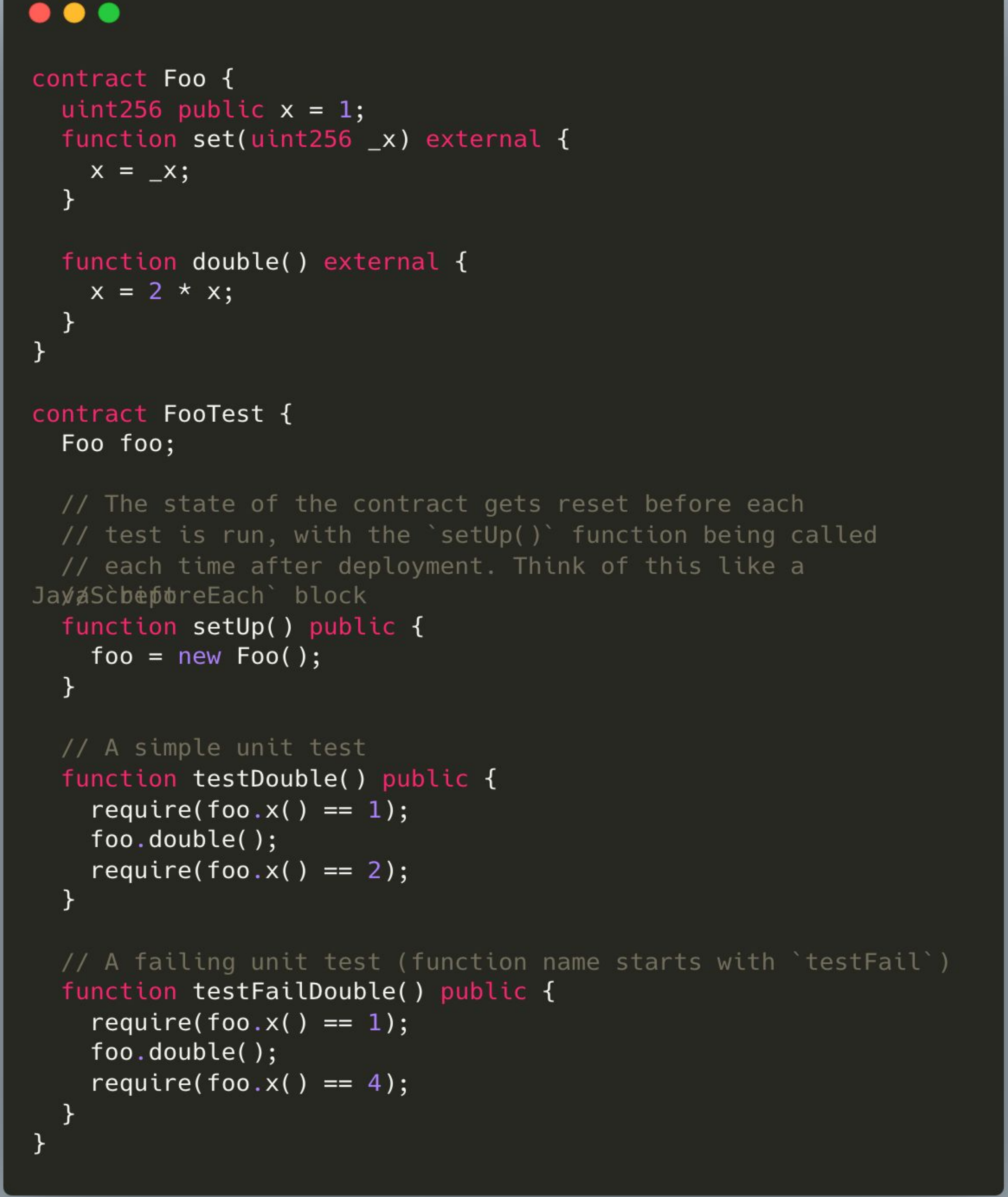
- Testing in JS requires a lot of boilerplate, large dependencies (node_modules/), and config files

- Library issues. For instance number in solidity are not native in JS so you need libs. But since there are several libs, you will have compatibility issues and this sucks.

- You need to learn new tools to test in JS like mocha. So as a dev you would need to be good in solidity AND in JS (not that hard but still).

- "Forge lets you write your tests in Solidity, so you can focus on what matters: writing good tests."

Example



```
contract Foo {
  uint256 public x = 1;
  function set(uint256 _x) external {
    x = _x;
  }

  function double() external {
    x = 2 * x;
  }
}

contract FooTest {
  Foo foo;

  // The state of the contract gets reset before each
  // test is run, with the `setUp()` function being called
  // each time after deployment. Think of this like a
JavaScript `beforeEach` block
  function setUp() public {
    foo = new Foo();
  }

  // A simple unit test
  function testDouble() public {
    require(foo.x() == 1);
    foo.double();
    require(foo.x() == 2);
  }

  // A failing unit test (function name starts with `testFail`)
  function testFailDouble() public {
    require(foo.x() == 1);
    foo.double();
    require(foo.x() == 4);
  }
}
```

Cheatcodes

- You might want to be able to control your entire environment.

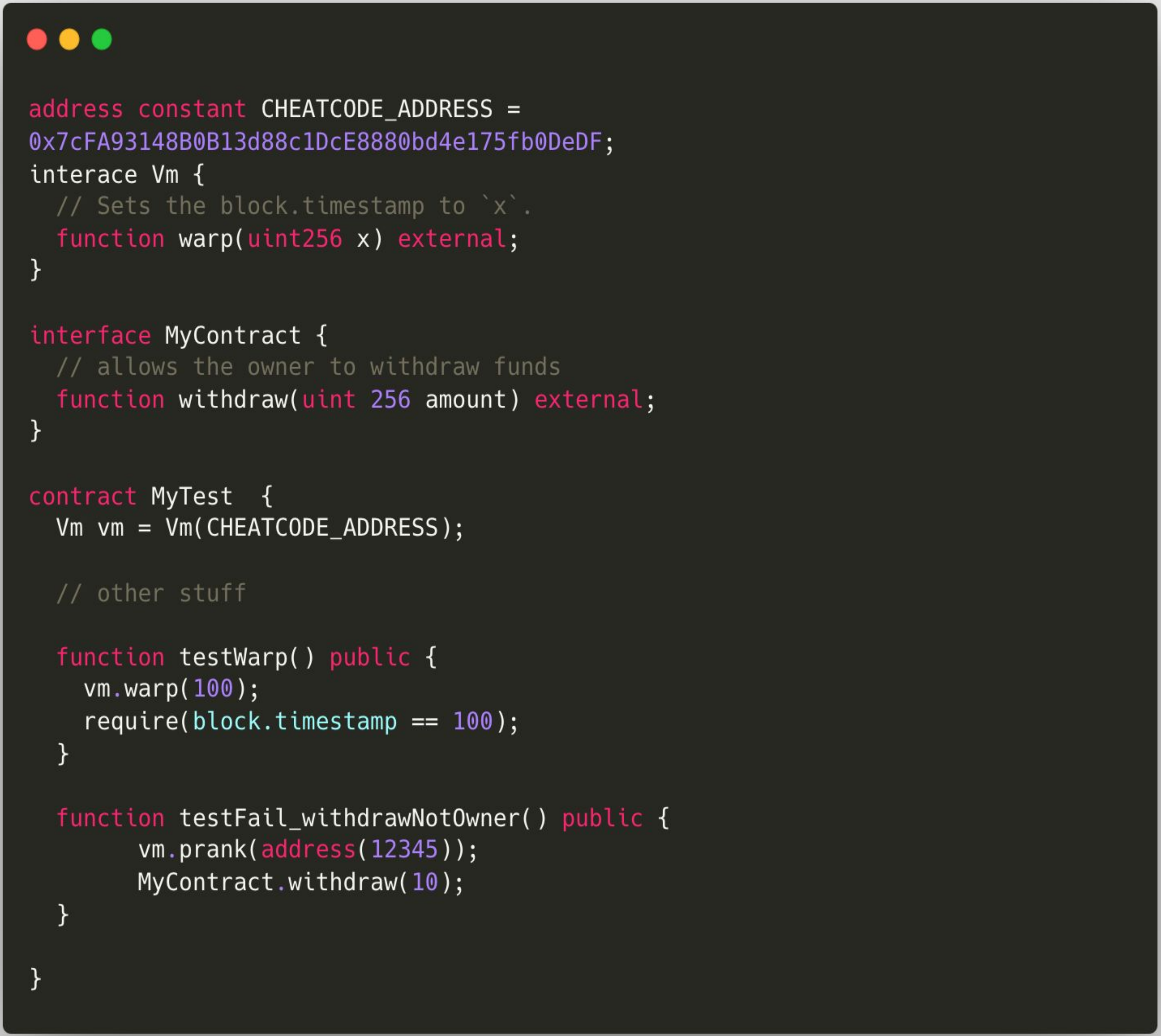
- Forge allows you to manipulate the state of the execution VM.

- You can change the time, the block number, your identity etc…

- With the cheatcodes, you can test pretty much every scenarios.

https://book.getfoundry.sh/cheatcodes/

Example



```
address constant CHEATCODE_ADDRESS =
0x7cFA93148B0B13d88c1DcE8880bd4e175fb0DeDF;
interace Vm {
  // Sets the block.timestamp to `x`.
  function warp(uint256 x) external;
}

interface MyContract {
  // allows the owner to withdraw funds
  function withdraw(uint 256 amount) external;
}

contract MyTest  {
  Vm vm = Vm(CHEATCODE_ADDRESS);

  // other stuff

  function testWarp() public {
    vm.warp(100);
    require(block.timestamp == 100);
  }

  function testFail_withdrawNotOwner() public {
        vm.prank(address(12345));
        MyContract.withdraw(10);
  }

}
```

A lot of other features

- El famoso debug with logs is still possible ! yay 🔥

  --> import "forge-std/console2.sol";

- import libs like openzeppelin or solmate

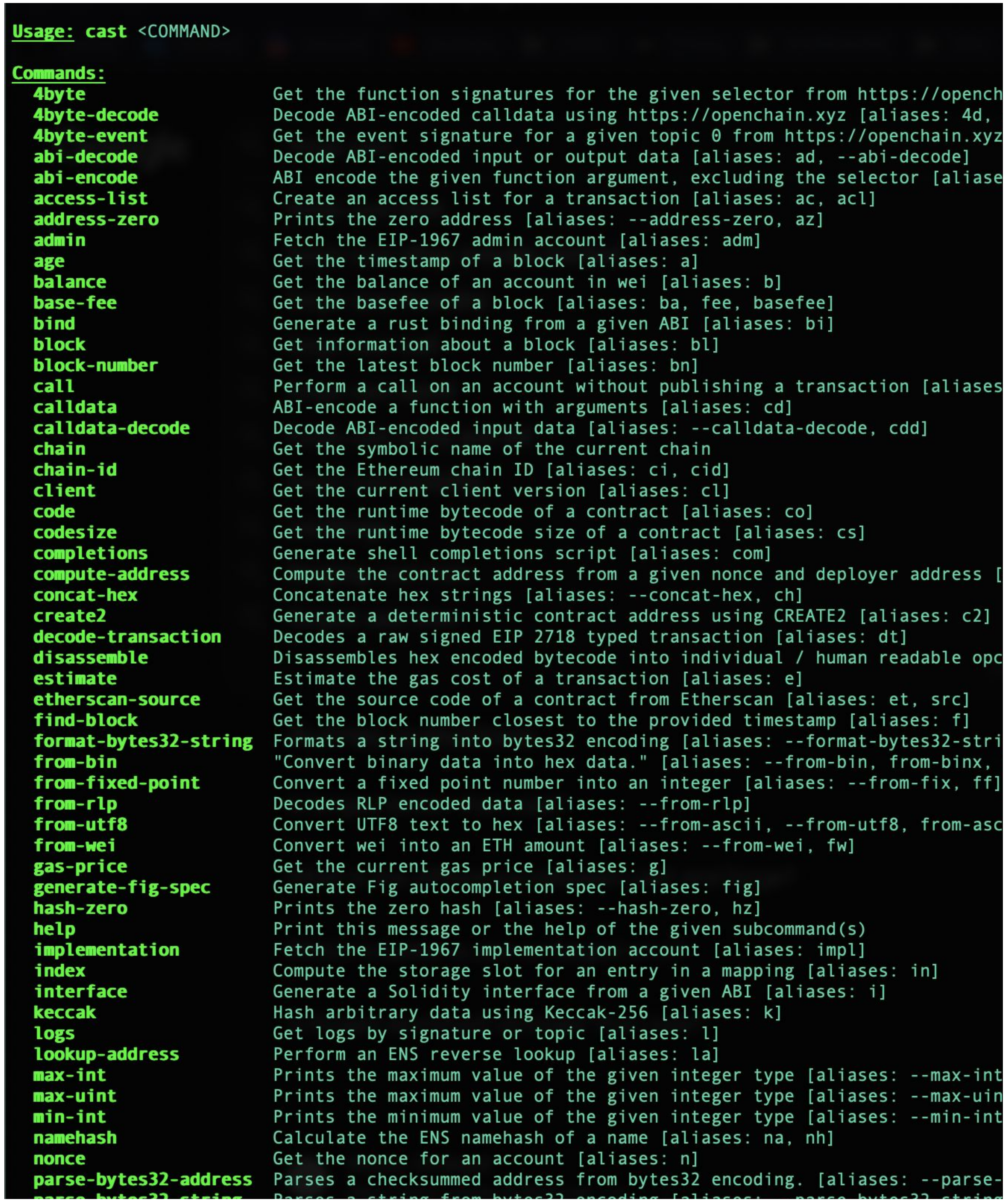
  → forge install transmissions11/solmate

- security tools, fuzzer
- Gas optimizations
- Execution traces
- contracts verification

CAST

CLI tool - CAST

- Allow you to perform Ethereum RPC calls



```
Usage: cast <COMMAND>

Commands:
  4byte                   Get the function signatures for the given selector from https://opench
  4byte-decode            Decode ABI-encoded calldata using https://openchain.xyz [aliases: 4d,
  4byte-event             Get the event signature for a given topic 0 from https://openchain.xyz
  abi-decode              Decode ABI-encoded input or output data [aliases: ad, --abi-decode]
  abi-encode              ABI encode the given function argument, excluding the selector [aliase
  access-list             Create an access list for a transaction [aliases: ac, acl]
  address-zero            Prints the zero address [aliases: --address-zero, az]
  admin                   Fetch the EIP-1967 admin account [aliases: adm]
  age                     Get the timestamp of a block [aliases: a]
  balance                 Get the balance of an account in wei [aliases: b]
  base-fee                Get the basefee of a block [aliases: ba, fee, basefee]
  bind                    Generate a rust binding from a given ABI [aliases: bi]
  block                   Get information about a block [aliases: bl]
  block-number            Get the latest block number [aliases: bn]
  call                    Perform a call on an account without publishing a transaction [aliases
  calldata                ABI-encode a function with arguments [aliases: cd]
  calldata-decode         Decode ABI-encoded input data [aliases: --calldata-decode, cdd]
  chain                   Get the symbolic name of the current chain
  chain-id                Get the Ethereum chain ID [aliases: ci, cid]
  client                  Get the current client version [aliases: cl]
  code                    Get the runtime bytecode of a contract [aliases: co]
  codesize                Get the runtime bytecode size of a contract [aliases: cs]
  completions             Generate shell completions script [aliases: com]
  compute-address         Compute the contract address from a given nonce and deployer address [
  concat-hex              Concatenate hex strings [aliases: --concat-hex, ch]
  create2                 Generate a deterministic contract address using CREATE2 [aliases: c2]
  decode-transaction      Decodes a raw signed EIP 2718 typed transaction [aliases: dt]
  disassemble             Disassembles hex encoded bytecode into individual / human readable opc
  estimate                Estimate the gas cost of a transaction [aliases: e]
  etherscan-source        Get the source code of a contract from Etherscan [aliases: et, src]
  find-block              Get the block number closest to the provided timestamp [aliases: f]
  format-bytes32-string   Formats a string into bytes32 encoding [aliases: --format-bytes32-stri
  from-bin                "Convert binary data into hex data." [aliases: --from-bin, from-binx,
  from-fixed-point        Convert a fixed point number into an integer [aliases: --from-fix, ff]
  from-rlp                Decodes RLP encoded data [aliases: --from-rlp]
  from-utf8               Convert UTF8 text to hex [aliases: --from-ascii, --from-utf8, from-asc
  from-wei                Convert wei into an ETH amount [aliases: --from-wei, fw]
  gas-price               Get the current gas price [aliases: g]
  generate-fig-spec       Generate Fig autocompletion spec [aliases: fig]
  hash-zero               Prints the zero hash [aliases: --hash-zero, hz]
  help                    Print this message or the help of the given subcommand(s)
  implementation          Fetch the EIP-1967 implementation account [aliases: impl]
  index                   Compute the storage slot for an entry in a mapping [aliases: in]
  interface               Generate a Solidity interface from a given ABI [aliases: i]
  keccak                  Hash arbitrary data using Keccak-256 [aliases: k]
  logs                    Get logs by signature or topic [aliases: l]
  lookup-address          Perform an ENS reverse lookup [aliases: la]
  max-int                 Prints the maximum value of the given integer type [aliases: --max-int
  max-uint                Prints the maximum value of the given integer type [aliases: --max-uin
  min-int                 Prints the minimum value of the given integer type [aliases: --min-int
  namehash                Calculate the ENS namehash of a name [aliases: na, nh]
  nonce                   Get the nonce for an account [aliases: n]
  parse-bytes32-address   Parses a checksummed address from bytes32 encoding. [aliases: --parse-
```

RPCs ELI5

- RPC (Remote Procedure Call): A protocol allowing programs to request services from another program over a network.

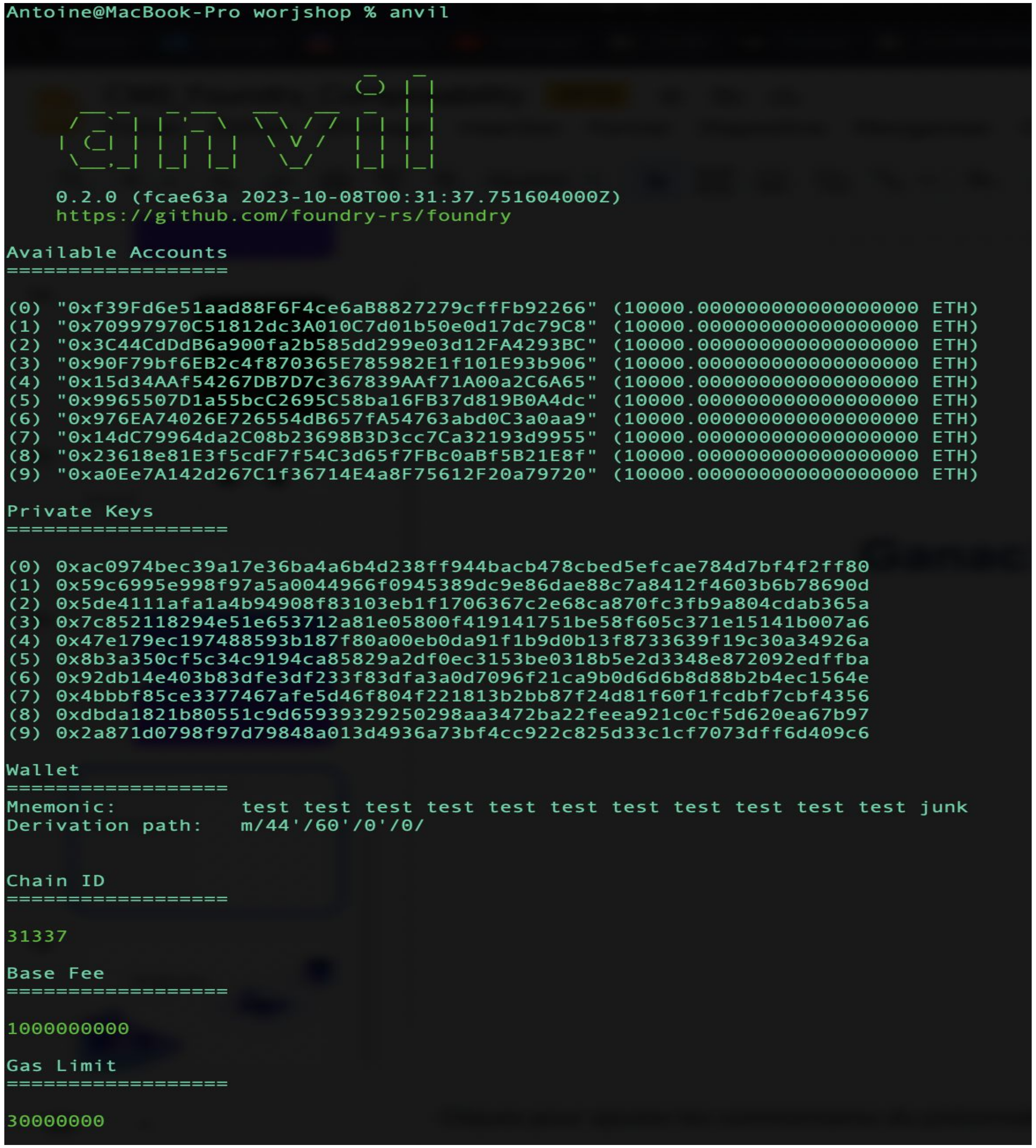
💻 <-> 💻

- Communicate with blockchain nodes remotely.
- Used to fetch data, submit transactions, and more.
- Enables DApps to interact with blockchain nodes.
- Crucial for testing and deploying smart contracts.

ANVIL

Ganache like devnet

- use it for testing the contracts from frontends or for interacting over RPC



```
Antoine@MacBook-Pro worjshop % anvil

                             _   _
                            (_) | |
      __ _   _ __   __   __  _  | |
     / _` | | '_ \  \ \ / / | | | |
    | (_| | | | | |  \ V /  | | | |
     \__,_| |_| |_|   \_/   |_| |_|

    0.2.0 (fcae63a 2023-10-08T00:31:37.751604000Z)
    https://github.com/foundry-rs/foundry

Available Accounts
==================

(0) "0xf39Fd6e51aad88F6F4ce6aB8827279cffFb92266" (10000.000000000000000000 ETH)
(1) "0x70997970C51812dc3A010C7d01b50e0d17dc79C8" (10000.000000000000000000 ETH)
(2) "0x3C44CdDdB6a900fa2b585dd299e03d12FA4293BC" (10000.000000000000000000 ETH)
(3) "0x90F79bf6EB2c4f870365E785982E1f101E93b906" (10000.000000000000000000 ETH)
(4) "0x15d34AAf54267DB7D7c367839AAf71A00a2C6A65" (10000.000000000000000000 ETH)
(5) "0x9965507D1a55bcC2695C58ba16FB37d819B0A4dc" (10000.000000000000000000 ETH)
(6) "0x976EA74026E726554dB657fA54763abd0C3a0aa9" (10000.000000000000000000 ETH)
(7) "0x14dC79964da2C08b23698B3D3cc7Ca32193d9955" (10000.000000000000000000 ETH)
(8) "0x23618e81E3f5cdF7f54C3d65f7FBc0aBf5B21E8f" (10000.000000000000000000 ETH)
(9) "0xa0Ee7A142d267C1f36714E4a8F75612F20a79720" (10000.000000000000000000 ETH)

Private Keys
==================

(0) 0xac0974bec39a17e36ba4a6b4d238ff944bacb478cbed5efcae784d7bf4f2ff80
(1) 0x59c6995e998f97a5a0044966f0945389dc9e86dae88c7a8412f4603b6b78690d
(2) 0x5de4111afa1a4b94908f83103eb1f1706367c2e68ca870fc3fb9a804cdab365a
(3) 0x7c852118294e51e653712a81e05800f419141751be58f605c371e15141b007a6
(4) 0x47e179ec197488593b187f80a00eb0da91f1b9d0b13f8733639f19c30a34926a
(5) 0x8b3a350cf5c34c9194ca85829a2df0ec3153be0318b5e2d3348e872092edffba
(6) 0x92db14e403b83dfe3df233f83dfa3a0d7096f21ca9b0d6d6b8d88b2b4ec1564e
(7) 0x4bbbf85ce3377467afe5d46f804f221813b2bb87f24d81f60f1fcdbf7cbf4356
(8) 0xdbda1821b80551c9d65939329250298aa3472ba22feea921c0cf5d620ea67b97
(9) 0x2a871d0798f97d79848a013d4936a73bf4cc922c825d33c1cf7073dff6d409c6

Wallet
==================
Mnemonic:          test test test test test test test test test test test junk
Derivation path:   m/44'/60'/0'/0/


Chain ID
==================

31337

Base Fee
==================

1000000000

Gas Limit
==================

30000000
```

Tasks list

- Create a Github repository & share it with the teacher. Create your report in the README.md (2 pts)
- Install Foundry & create a Forge project (2 pts)
- Create an ERC20 token contract (2 pts)
    - Chose a ticker
    - Chose a total supply
    - Chose a decimal number
- Implement all ERC20 functions (inherit from Open Zeppelin) (1 pts)
- Create a getToken() function which exchanges ETH tokens (1)
- Create a script to deploy your contract(s) (2 pts)
    - Migrate to Anvil

Tasks list

- Implement customer allow listing (3 pts)
  - Create a mapping to track allowed users
  - Create an admin function to add customers to allow list
  - Create a modifier to allow only allowlisted users to call getToken()
- Implement multi level distribution (3 pts)
  - Differentiate levels of participation for users (tier 1, 2, 3)
  - Index quantity of tokens sent in getToken() on tier level
- Implement air drop functions (2 pts)
  - Create a function to mint and send token to an arbitrary address, by admin
- Deploy to a testnet (2 pts)
  - Create an account on Infura (or Alchemy) and configure Forge
  - Credit tokens to teacher
- Teacher Github: 0xEniotna

ATTENTION

  - Don’t post your private key on Github
    - Don’t post your private key on Github
    - Don’t post your private key on Github
    - Don’t post your private key on Github
    - Don’t post your private key on Github
    - Don’t post your private key on Github
    - Don’t post your private key on Github
    - Don’t post your private key on Github
    - Don’t post your private key on Github
    - Don’t post your private key on Github
    - Don’t post your private key on Github
    - Don’t post your private key on Github
    - Don’t post your private key on Github
    - Don’t post your private key on Github
    - Don’t post your private key on Github
    - Don’t post your private key on Github
    - Don’t post your private key on Github
    - Don’t post your private key on Github
    - Don’t post your private key on Github
- Also, you never push the compiled files on Github (what appears when you run forge build)

Ressources

- https://book.getfoundry.sh/getting-started/installation
- https://solidity-by-example.org/hello-world/
- https://docs.soliditylang.org/en/v0.8.21/

Merci

pour votre attention !