getting_started_with_ethereum

March 27, 2018

1 Getting Started with Ethereum

1.1 Requirements

Before you start developing a dapp make sure to run ganache-cli (formerly testrpc) in a seperate terminal.

Install:

```
npm install -g ganache-cli
```

Run:

```
ganache-cli
```

1.2 Imports

- web3 : module for interacting with the ethereum blockchain

  ```
  pip install web3
  ```

- solc : provides functions for compiling solidity

  ```
  pip install py-solc
  ```

```
In [1]: from web3 import Web3, HTTPProvider
        from solc import compile_source
```

1.3 Connect to the Blockchain

```
In [2]: provider = 'http://' + input('Enter the ip of your provider: ')
        provider = provider + ':' + input('Enter the port of your provider: ')
        print('Connecting to RPC at: ' + provider)
        WEB3 = Web3(HTTPProvider(provider))
```

```
Connecting to RPC at: http://13.59.43.120:8545
```

1.4 Compile a Contract

```
In [3]: def get_contract_source(contract_path): #opens file and returns a string of the files
            contract = open(contract_path, 'r')
            contract_str = contract.read()
            contract.close()
            return contract_str
```

```
In [4]: CONTRACT_SOURCE = get_contract_source('greeter.sol') #multiline string containing the
        CONTRACT_COMPILED = compile_source(CONTRACT_SOURCE) #raw output of compiled contract c
        CONTRACT_INTERFACE = CONTRACT_COMPILED['<stdin>:Greeter'] #interface for the contract

        CONTRACT_ABI = CONTRACT_INTERFACE['abi'] #abi of the contract (required for calling an
        CONTRACT_BIN = CONTRACT_INTERFACE['bin'] #bytecode of the contract (required for deplo
        CONTRACT_BIN_RUNTIME = CONTRACT_INTERFACE['bin-runtime'] #bytecode runtime (required f
```

1.5 Deploy the Contract

```
In [5]: def deploy_contract(web3, contract, deployer_address, gas):
                tx_hash = contract.deploy(transaction={"from": deployer_address, "gas": gas})
                receipt = web3.eth.getTransactionReceipt(tx_hash)
                return receipt['contractAddress']
```

```
In [6]: def initialize_contract(web3):
            contract = web3.eth.contract(
                abi=CONTRACT_ABI,
                bytecode=CONTRACT_BIN,
                bytecode_runtime=CONTRACT_BIN_RUNTIME
            )

            deployer_address = input("Enter the address of the deployer: ")
            deployer_address = web3.eth.accounts[0] if deployer_address == "" else deployer_ad

            gas = input("Enter the desired gas for the contract: ")
            gas = 1000000 if gas == "" else int(gas)

            contract_address = deploy_contract(web3, contract, deployer_address, gas)
            print("The contract address is: " + contract_address)
            return contract_address
```

```
In [8]: CONTRACT_ADDRESS = initialize_contract(WEB3)

The contract address is: 0xE2cC7b448617bB0Db302215fB94e6F104EbAa3CC
```

1.6 Interact With the Chain

1.6.1 Read From the Chain

Reading from the Ethereum blockchain costs 0 gas.

In solidity, read functions contain the 'constant' reserved word.

```
In [13]: contract_instance = WEB3.eth.contract(abi=CONTRACT_ABI)
         contract_call = contract_instance.call({'to' : CONTRACT_ADDRESS})
         contract_call.greet()

Out[13]: "'hello there'"
```

1.7 Writing To the Chain

Reading from the Ethereum chain costs gas.

```
In [14]: contract_instance = WEB3.eth.contract(abi=CONTRACT_ABI)
         contract_transact = contract_instance.transact({"from": input('Send from address: '),
         contract_transact.setGreeting(input('Enter desired greeting: '))

Out[14]: '0x8ce694a18e7dba1ab89462d070de8bc804d1717f377e20fb9a98d25713f17104'
```