NFT Analyzer

Zibo Yang

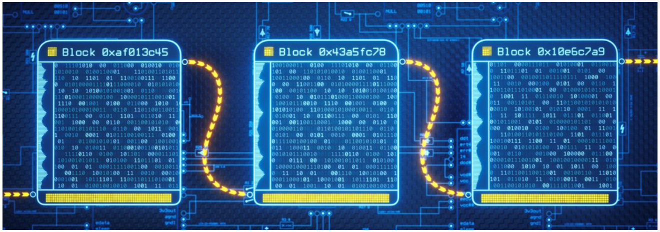
zibo.yang@polytechnique.edu

March 2, 2022

Table of Contents

Cryptocurrency: nothing but a digital currency

Blockchain: the distributed database that is shared among the nodes of the computer network.
Ethereum: one of the most successful cryptocurrencies. (All the transactions information have been recorded on blockchain and open for review on Etherscan)

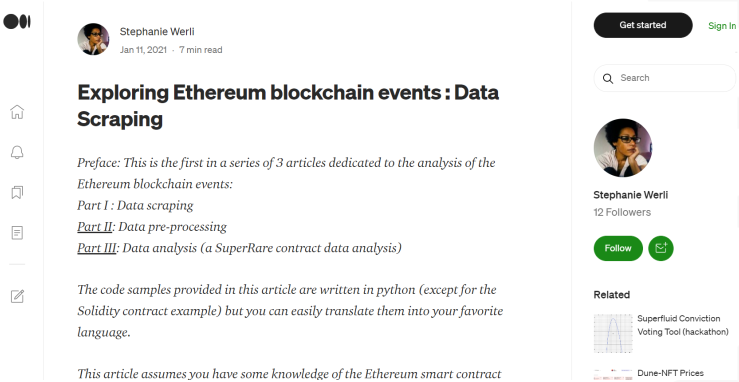


NFT: Non-Fungible Token

NFT: the unique digital token or asset that can’t be replaced with something else and are verified and stored using blockchain technology.(Artwork, Real Estate)

Intention

Idea: Exploring Ethereum blockchain events



Stephanie Werli
Jan 11, 2021 · 7 min read

Exploring Ethereum blockchain events : Data Scraping

Preface: This is the first in a series of 3 articles dedicated to the analysis of the Ethereum blockchain events:
Part I : Data scraping
Part II: Data pre-processing
Part III: Data analysis (a SuperRare contract data analysis)

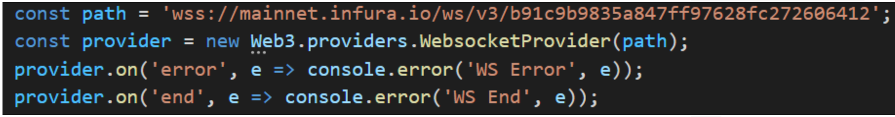
The code samples provided in this article are written in python (except for the Solidity contract example) but you can easily translate them into your favorite language.

This article assumes you have some knowledge of the Ethereum smart contract

Outcome

NFT Analyzer

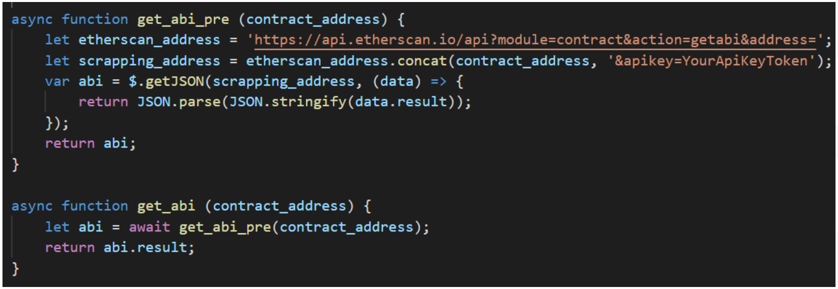
Web3



```
const path = 'wss://mainnet.infura.io/ws/v3/b91c9b9835a847ff97628fc272606412';
const provider = new Web3.providers.WebsocketProvider(path);
provider.on('error', e => console.error('WS Error', e));
provider.on('end', e => console.error('WS End', e));
```

INFURA: blockchain development suite for API and tools.
Web3.js: Ethereum JavaScript API
Contruct variable *web*3 to get access to APIs

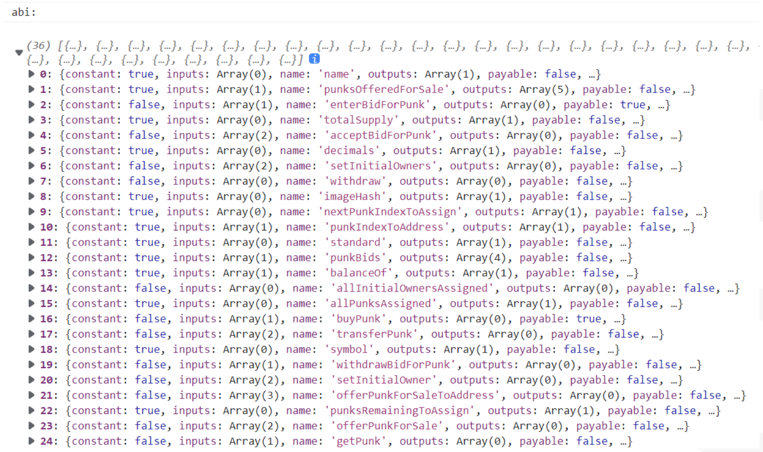
Get ABI



```
async function get_abi_pre (contract_address) {
    let etherscan_address = 'https://api.etherscan.io/api?module=contract&action=getabi&address=';
    let scrapping_address = etherscan_address.concat(contract_address, '&apikey=YourApiKeyToken');
    var abi = $.getJSON(scrapping_address, (data) => {
        return JSON.parse(JSON.stringify(data.result));
    });
    return abi;
}

async function get_abi (contract_address) {
    let abi = await get_abi_pre(contract_address);
    return abi.result;
}
```

ABI: Contract Application Binary Interface (key to request data from smart contract)

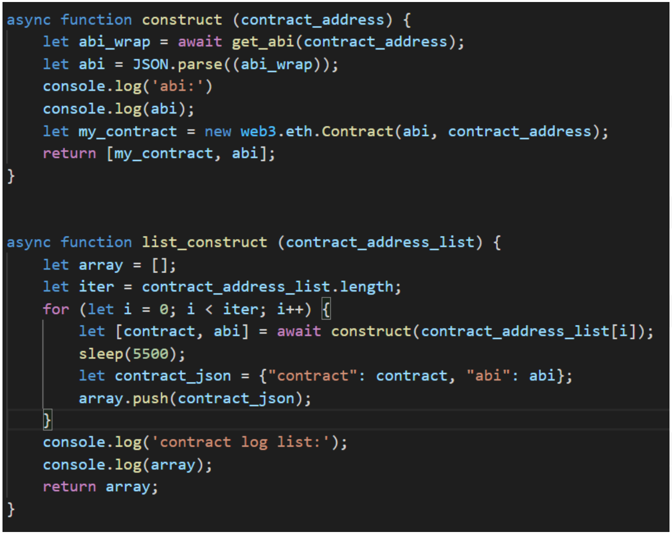
Get ABI



```
abi:

▾ (36) [{…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…},
  {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}]
  ▶ 0: {constant: true, inputs: Array(0), name: 'name', outputs: Array(1), payable: false, …}
  ▶ 1: {constant: true, inputs: Array(1), name: 'punksOfferedForSale', outputs: Array(5), payable: false, …}
  ▶ 2: {constant: false, inputs: Array(1), name: 'enterBidForPunk', outputs: Array(0), payable: true, …}
  ▶ 3: {constant: true, inputs: Array(0), name: 'totalSupply', outputs: Array(1), payable: false, …}
  ▶ 4: {constant: false, inputs: Array(2), name: 'acceptBidForPunk', outputs: Array(0), payable: false, …}
  ▶ 5: {constant: true, inputs: Array(0), name: 'decimals', outputs: Array(1), payable: false, …}
  ▶ 6: {constant: false, inputs: Array(2), name: 'setInitialOwners', outputs: Array(0), payable: false, …}
  ▶ 7: {constant: false, inputs: Array(0), name: 'withdraw', outputs: Array(0), payable: false, …}
  ▶ 8: {constant: true, inputs: Array(0), name: 'imageHash', outputs: Array(1), payable: false, …}
  ▶ 9: {constant: true, inputs: Array(0), name: 'nextPunkIndexToAssign', outputs: Array(1), payable: false, …}
  ▶ 10: {constant: true, inputs: Array(1), name: 'punkIndexToAddress', outputs: Array(1), payable: false, …}
  ▶ 11: {constant: true, inputs: Array(0), name: 'standard', outputs: Array(1), payable: false, …}
  ▶ 12: {constant: true, inputs: Array(1), name: 'punkBids', outputs: Array(4), payable: false, …}
  ▶ 13: {constant: true, inputs: Array(1), name: 'balanceOf', outputs: Array(1), payable: false, …}
  ▶ 14: {constant: false, inputs: Array(0), name: 'allInitialOwnersAssigned', outputs: Array(0), payable: false, …}
  ▶ 15: {constant: true, inputs: Array(0), name: 'allPunksAssigned', outputs: Array(1), payable: false, …}
  ▶ 16: {constant: false, inputs: Array(1), name: 'buyPunk', outputs: Array(0), payable: true, …}
  ▶ 17: {constant: false, inputs: Array(2), name: 'transferPunk', outputs: Array(0), payable: false, …}
  ▶ 18: {constant: true, inputs: Array(0), name: 'symbol', outputs: Array(1), payable: false, …}
  ▶ 19: {constant: false, inputs: Array(1), name: 'withdrawBidForPunk', outputs: Array(0), payable: false, …}
  ▶ 20: {constant: false, inputs: Array(2), name: 'setInitialOwner', outputs: Array(0), payable: false, …}
  ▶ 21: {constant: false, inputs: Array(3), name: 'offerPunkForSaleToAddress', outputs: Array(0), payable: false, …}
  ▶ 22: {constant: true, inputs: Array(0), name: 'punksRemainingToAssign', outputs: Array(1), payable: false, …}
  ▶ 23: {constant: false, inputs: Array(2), name: 'offerPunkForSale', outputs: Array(0), payable: false, …}
  ▶ 24: {constant: false, inputs: Array(1), name: 'getPunk', outputs: Array(0), payable: false, …}
```

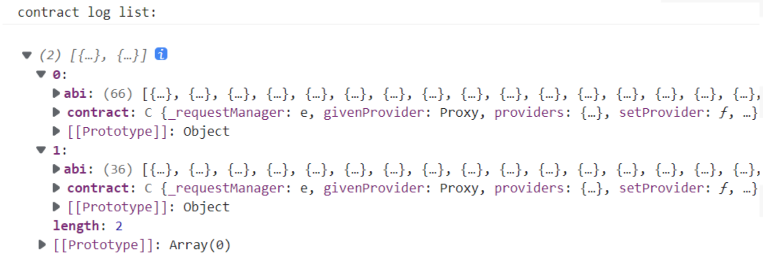
ABI: Contract Application Binary Interface (key to request data from smart contract)

Construction



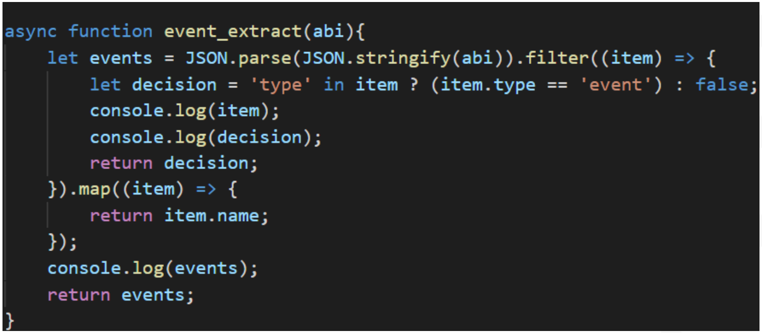
```
async function construct (contract_address) {
    let abi_wrap = await get_abi(contract_address);
    let abi = JSON.parse((abi_wrap));
    console.log('abi:')
    console.log(abi);
    let my_contract = new web3.eth.Contract(abi, contract_address);
    return [my_contract, abi];
}


async function list_construct (contract_address_list) {
    let array = [];
    let iter = contract_address_list.length;
    for (let i = 0; i < iter; i++) {
        let [contract, abi] = await construct(contract_address_list[i]);
        sleep(5500);
        let contract_json = {"contract": contract, "abi": abi};
        array.push(contract_json);
    }
    console.log('contract log list:');
    console.log(array);
    return array;
}
```

Construction



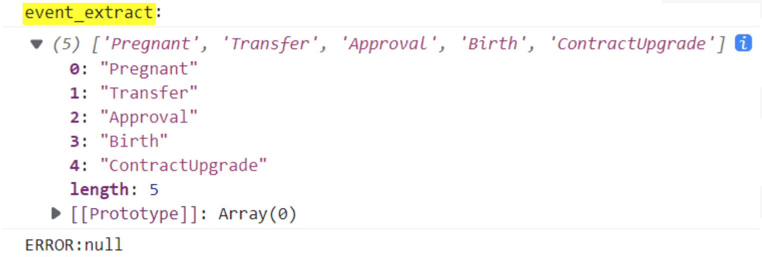
```
contract log list:

▼ (2) [{…}, {…}]
  ▼ 0:
    ▶ abi: (66) [{…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…},
    ▶ contract: C {_requestManager: e, givenProvider: Proxy, providers: {…}, setProvider: ƒ, …}
    ▶ [[Prototype]]: Object
  ▼ 1:
    ▶ abi: (36) [{…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…},
    ▶ contract: C {_requestManager: e, givenProvider: Proxy, providers: {…}, setProvider: ƒ, …}
    ▶ [[Prototype]]: Object
    length: 2
  ▶ [[Prototype]]: Array(0)
```

Extraction

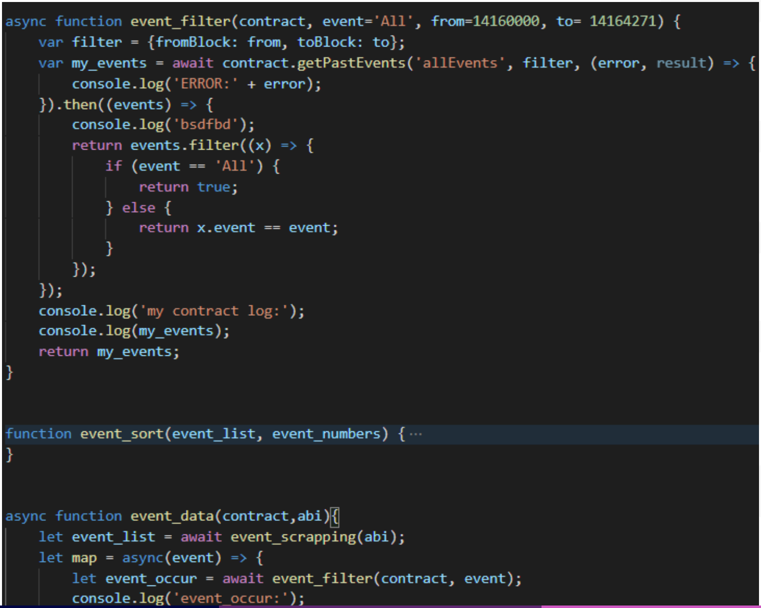


```
async function event_extract(abi){
    let events = JSON.parse(JSON.stringify(abi)).filter((item) => {
        let decision = 'type' in item ? (item.type == 'event') : false;
        console.log(item);
        console.log(decision);
        return decision;
    }).map((item) => {
        return item.name;
    });
    console.log(events);
    return events;
}
```

Extraction

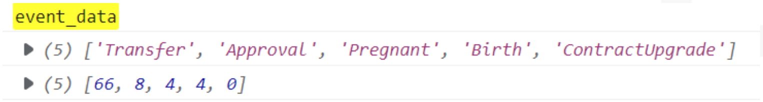

```
event_extract:
▼ (5) ['Pregnant', 'Transfer', 'Approval', 'Birth', 'ContractUpgrade']
    0: "Pregnant"
    1: "Transfer"
    2: "Approval"
    3: "Birth"
    4: "ContractUpgrade"
    length: 5
  ▶ [[Prototype]]: Array(0)
ERROR:null
```

Event



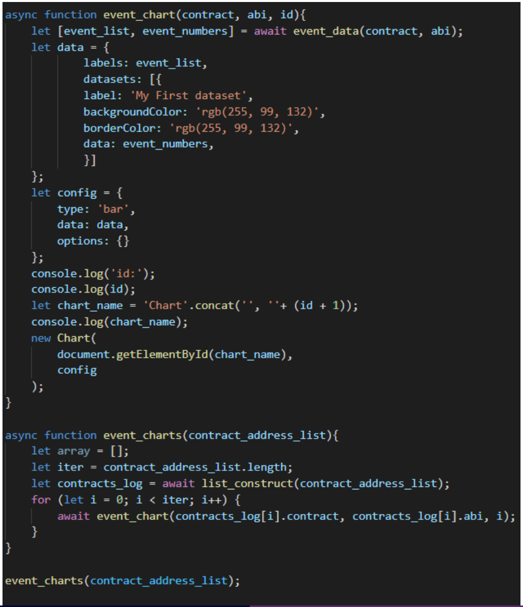
```
async function event_filter(contract, event='All', from=14160000, to= 14164271) {
    var filter = {fromBlock: from, toBlock: to};
    var my_events = await contract.getPastEvents('allEvents', filter, (error, result) => {
        console.log('ERROR:' + error);
    }).then((events) => {
        console.log('bsdfbd');
        return events.filter((x) => {
            if (event == 'All') {
                return true;
            } else {
                return x.event == event;
            }
        });
    });
    console.log('my contract log:');
    console.log(my_events);
    return my_events;
}


function event_sort(event_list, event_numbers) {···
}


async function event_data(contract,abi){
    let event_list = await event_scrapping(abi);
    let map = async(event) => {
        let event_occur = await event_filter(contract, event);
        console.log('event_occur:');
```

Event



```
event_data
▶ (5) ['Transfer', 'Approval', 'Pregnant', 'Birth', 'ContractUpgrade']
▶ (5) [66, 8, 4, 4, 0]
```

Chart



```
async function event_chart(contract, abi, id){
    let [event_list, event_numbers] = await event_data(contract, abi);
    let data = {
            labels: event_list,
            datasets: [{
            label: 'My First dataset',
            backgroundColor: 'rgb(255, 99, 132)',
            borderColor: 'rgb(255, 99, 132)',
            data: event_numbers,
            }]
    };
    let config = {
        type: 'bar',
        data: data,
        options: {}
    };
    console.log('id:');
    console.log(id);
    let chart_name = 'Chart'.concat('', ''+ (id + 1));
    console.log(chart_name);
    new Chart(
        document.getElementById(chart_name),
        config
    );
}

async function event_charts(contract_address_list){
    let array = [];
    let iter = contract_address_list.length;
    let contracts_log = await list_construct(contract_address_list);
    for (let i = 0; i < iter; i++) {
        await event_chart(contracts_log[i].contract, contracts_log[i].abi, i);
    }
}

event_charts(contract_address_list);
```

Last but not least

Any questions/comments on this project are welcome.