Smart contracts on Ethereum

Unboxing the Truffle Box for Oracle JET

Robert van Mölken

Who am I?

Robert van Mölken
Blockchain / Integration Specialist
Oracle Developer Champion / ACE

Author of the NEW book:
Blockchain across Oracle

Oracle Groundbreaker Ambassador
EXPERT INSIGHT
Robert van Mölken
Blockchain across Oracle
Understand the details and implications of the Blockchain for Oracle developers and customers
Packt
Phil Wilkins, Robert van Mölken
Foreword by Vikas Anand
Implementing Oracle Integration Cloud Service
Understand everything you need to know about Oracle's Integration Cloud Service and how to utilize it optimally for your business
Packt

Linkedin: linkedin.com/in/rvmolken
Blog: technology.vanmolken.nl
Twitter: @robertvanmolken

Oracle Groundbreakers

Oracle Groundbreaker Ambassador

Experts in Modern Development

- Cloud
- Microservices and Containers
- Java, JavaScript/Node.js, PHP, Python
- Blockchain, Internet of Things
- Continuous Delivery
- Open Source Technologies
- SQL/NoSQL Databases
- Machine Learning, AI, Chatbots

developer.oracle.com/ambassador

@groundbreakers

Blockchain across Oracle

Blockchain across Oracle gives you the professional orientation to Blockchain that you need as an Oracle developer in today's changing world. Written by Oracle Developer Champion Robert van Mölken, this book gives you everything you need to get up to speed with the details of Blockchain. You'll really get to understand the Blockchain inside and out - and gain key insights into how the Blockchain affects Oracle developers and customers in this modern and disruptive era.

You'll take a detailed look at the cutting-edge Oracle cloud solutions that allow you to work with the Blockchain as an Oracle developer. You'll learn about Hyperledger Fabric, the opensource Blockchain framework used by Oracle as its core, and how to set up your own Oracle Blockchain Network. You'll design and develop a smart contract and learn how to run it on the Oracle Blockchain Cloud Service.

The final key section of this book looks at how the Blockchain will affect your customers across industry sectors. By studying key trends in the financial services sector, healthcare industry, and the transport industry, you'll discover how the options and possibilities for you and your clients are being transformed by the Blockchain across Oracle. You'll complete this professional orientation with a look at Blockchain future industry and technology directions.

Things you will learn:

- A full introduction to the Blockchain
- How the Blockchain affects Oracle developers and customers
- Core concepts including blocks, hashes, and chains, assets, transactions, and consensus
- How to work with Oracle Cloud to implement a Blockchain Network
- Design, develop, and run smart contracts on the Oracle Blockchain Cloud Service
- Blockchain security and privacy for Oracle developers and clients
- Public and private Blockchain decisions for Oracle architects and developers
- Industry analysis across finance, governance, and healthcare sectors
- Industry trends and the future of the Blockchain technology

Packt
www.packtpub.com

ISBN 978-1-78847-429-0
90000
9 781788 474290

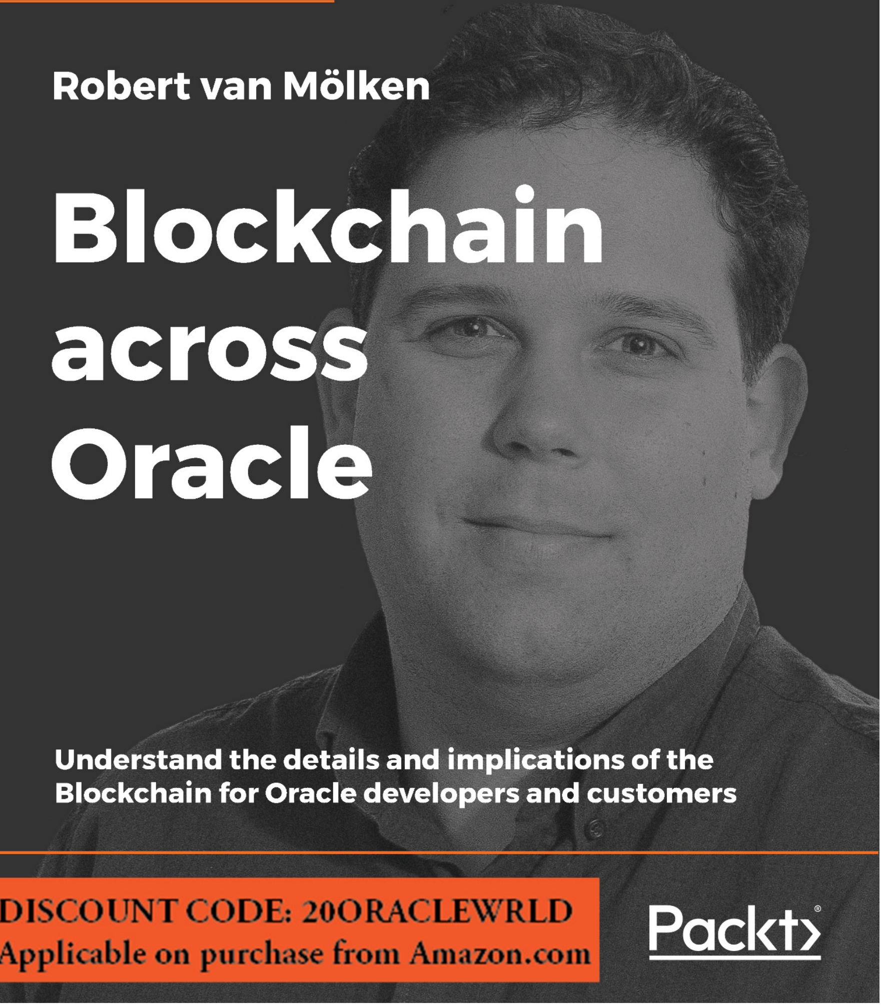
Blockchain across Oracle

Robert van Mölken

EXPERT INSIGHT

Robert van Mölken

Blockchain across Oracle



Understand the details and implications of the Blockchain for Oracle developers and customers

DISCOUNT CODE: 20ORACLEWRLD
Applicable on purchase from Amazon.com

Packt

Smart contracts on Ethereum

- Public versus Permissioned Blockchains
- Ethereum Blockchain App Platform
- Decentralized applications
- Dapp development for Ethereum
- Introducing the Truffle Suite / Boxes
- Introducing the Truffle Box for Oracle JET

Topics discussed in this 45-min Talk

Public versus Permissioned Blockchains

Where do I run my decentralized application?

Public Blockchains

- BitCoin, AltCoins (LiteCoin)
- Ethereum (DApps & Ether)
- NEO, IPFS & Blockstack (DApps)
- IOTA (= DLT for IoT)

Permissioned blockchains

- Eris::Monax & R3/Corda
- Oracle' Blockchain Cloud Service
- Microsoft' Blockchain as a Service
- Hyperledger Fabric & Multichain

Difference between public & permissioned blockchains?

Public or *permissionless* blockchains

- Anyone in the world can read information
- Anyone in the world can send transactions
- Anyone see transactions included if they are valid
- Anyone can participate in the ***consensus process***
- Fully decentralized

Private or *permissioned* blockchains

- Write permissions are kept centralized to one organization or consortium
- Read permissions may be public or restricted
- Include database management, auditing, etc
- Consortium of selected nodes participate in **consensus**
- Partly decentralized

Ethereum Blockchain App Platform

Ethereum Blockchain App Platform

- Decentralized Blockchain Platform (ethereum.org)
- Mainnet is public / shared global infrastructure
- Organizations can start their own (Test) network
- Key features: Smart contracts, Digital Currency
- Purposes: ICO's, Name Service, Voting & Betting, Funding, Ads, Games…

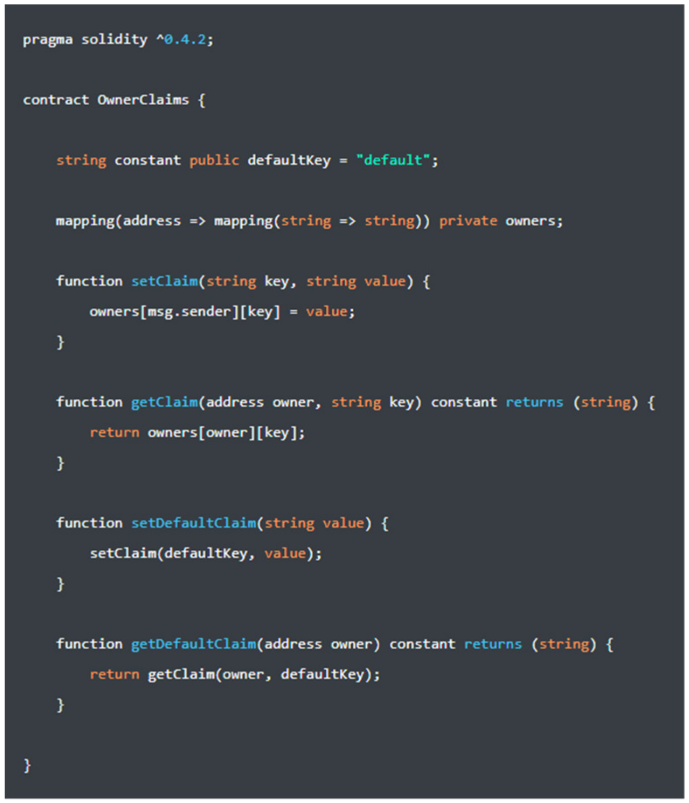
AMIS

What is a decentralized blockchain application (Dapp)?

A **Dapp** is a *'blockchain enabled'* web-app, that runs on a **peer-to-peer** *network* of computers rather than a single server, where **Smart Contracts** are allowing it to connect to the **blockchain** data. It contains both **front-end** and **back-end** and run **independently** on all nodes!

What does a decentralized app on the blockchain look like?

- Back-end powered by Smart Contracts
- Agreements between parties with automated execution that can act as a complement, or substitute, for legal contracts or business TXs
- Computer program code that is capable of facilitating, executing, and enforcing the negotiation or performance of an agreement
- Example shows contract to capture arbitrary data on the owner submitting claim upon transaction, given a blockchain address is not linked to a physical identity



```
pragma solidity ^0.4.2;

contract OwnerClaims {

    string constant public defaultKey = "default";

    mapping(address => mapping(string => string)) private owners;

    function setClaim(string key, string value) {
        owners[msg.sender][key] = value;
    }

    function getClaim(address owner, string key) constant returns (string) {
        return owners[owner][key];
    }

    function setDefaultClaim(string value) {
        setClaim(defaultKey, value);
    }

    function getDefaultClaim(address owner) constant returns (string) {
        return getClaim(owner, defaultKey);
    }

}
```

What does a decentralized app on the blockchain look like?

A decentralized application also includes:

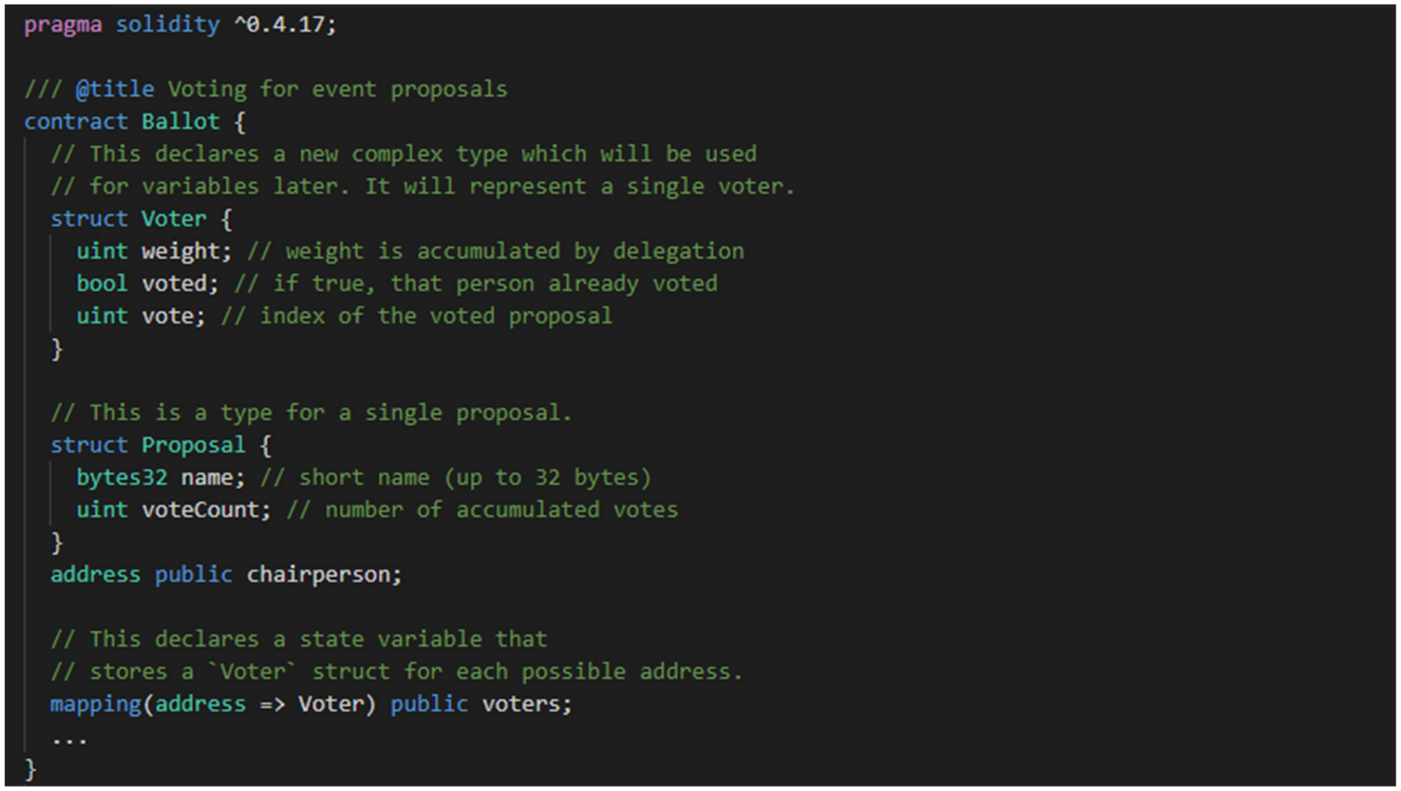
- Data model describing participants, assets, transactions and optional events
- Authorization and permission model
- APIs that let’s the front-end connect with the back-end
- A front-end web application (can run outside of blockchain)

Dapp development on Ethereum

Develop an Ethereum Dapp

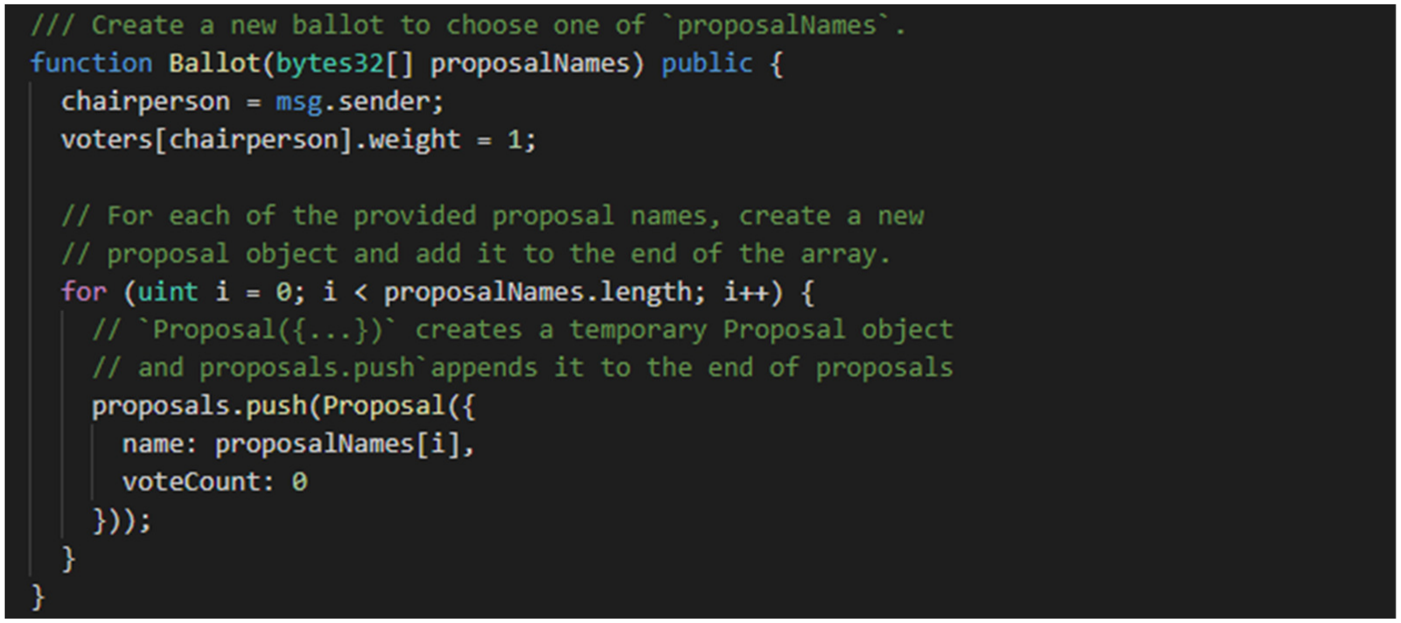
- Smart contracts can be written in:
  - Solidity (influenced by C++, Python, JavaScript)
  - The more stricter Vyper (influenced by Python)
- Compiled to bytecode and run on the EVM (Ethereum Virtual Machine)
- Code available as addresses on blockchain
- Interface (API / SDK):
  - **Web3.js** for JavaScript
  - **Web3j** for Java/Android
  - **Go Ethereum** for Go
- Front-end application implement API/SDK

Example Ballot Contract

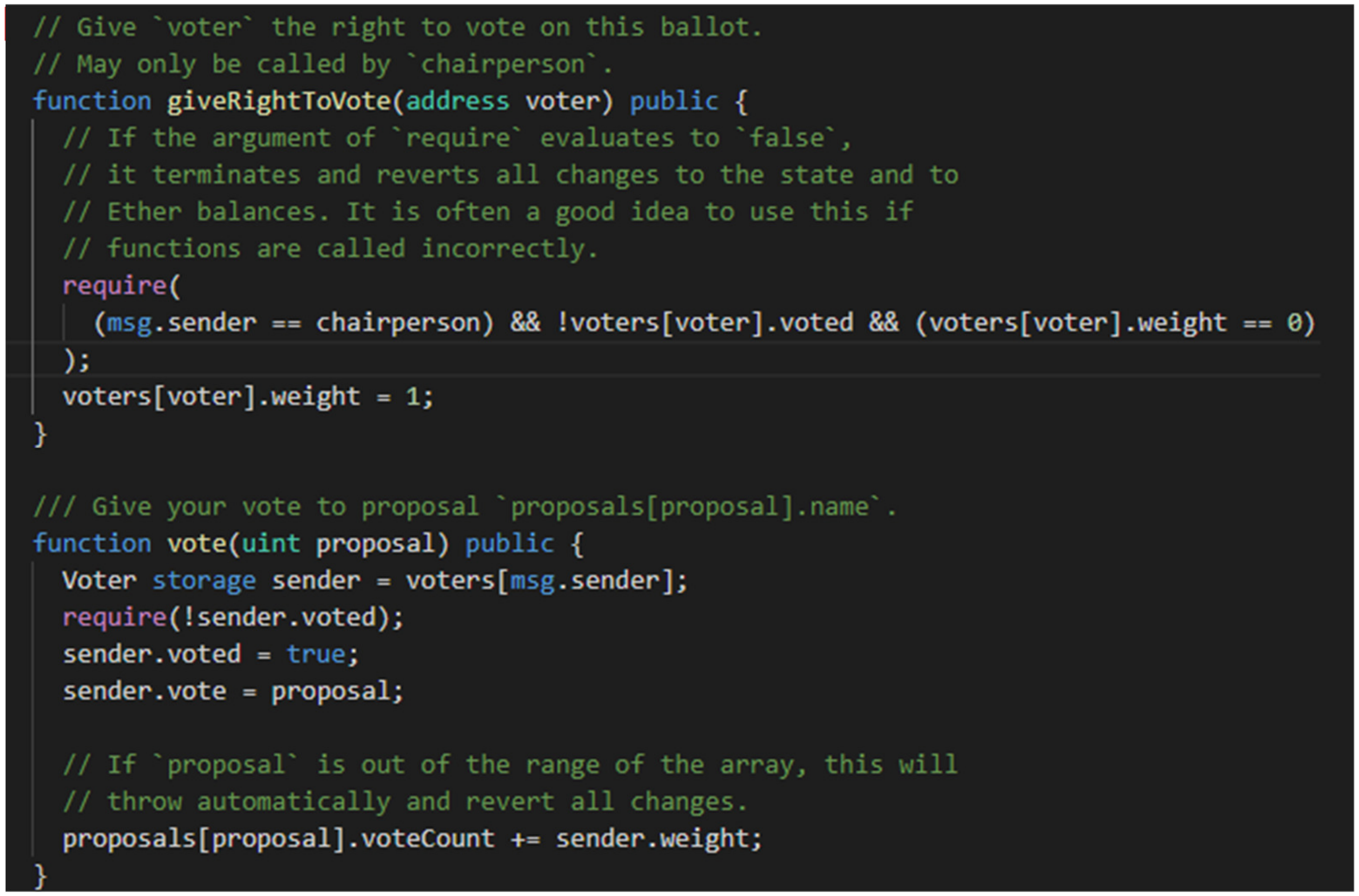


```
pragma solidity ^0.4.17;

/// @title Voting for event proposals
contract Ballot {
  // This declares a new complex type which will be used
  // for variables later. It will represent a single voter.
  struct Voter {
    uint weight; // weight is accumulated by delegation
    bool voted; // if true, that person already voted
    uint vote; // index of the voted proposal
  }

  // This is a type for a single proposal.
  struct Proposal {
    bytes32 name; // short name (up to 32 bytes)
    uint voteCount; // number of accumulated votes
  }
  address public chairperson;

  // This declares a state variable that
  // stores a `Voter` struct for each possible address.
  mapping(address => Voter) public voters;
  ...
}
```

Example Ballot Contract (Cond.)



```
/// Create a new ballot to choose one of `proposalNames`.
function Ballot(bytes32[] proposalNames) public {
    chairperson = msg.sender;
    voters[chairperson].weight = 1;

    // For each of the provided proposal names, create a new
    // proposal object and add it to the end of the array.
    for (uint i = 0; i < proposalNames.length; i++) {
        // `Proposal({...})` creates a temporary Proposal object
        // and proposals.push`appends it to the end of proposals
        proposals.push(Proposal({
            name: proposalNames[i],
            voteCount: 0
        }));
    }
}
```

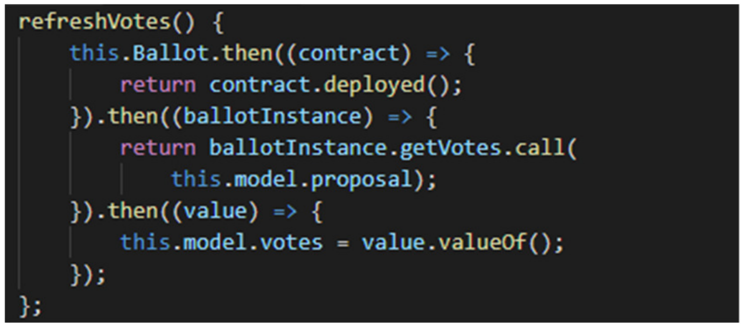


```
// Give `voter` the right to vote on this ballot.
// May only be called by `chairperson`.
function giveRightToVote(address voter) public {
    // If the argument of `require` evaluates to `false`,
    // it terminates and reverts all changes to the state and to
    // Ether balances. It is often a good idea to use this if
    // functions are called incorrectly.
    require(
        (msg.sender == chairperson) && !voters[voter].voted && (voters[voter].weight == 0)
    );
    voters[voter].weight = 1;
}

/// Give your vote to proposal `proposals[proposal].name`.
function vote(uint proposal) public {
    Voter storage sender = voters[msg.sender];
    require(!sender.voted);
    sender.voted = true;
    sender.vote = proposal;

    // If `proposal` is out of the range of the array, this will
    // throw automatically and revert all changes.
    proposals[proposal].voteCount += sender.weight;
}
```

Web3 API – Call versus Transaction

Call

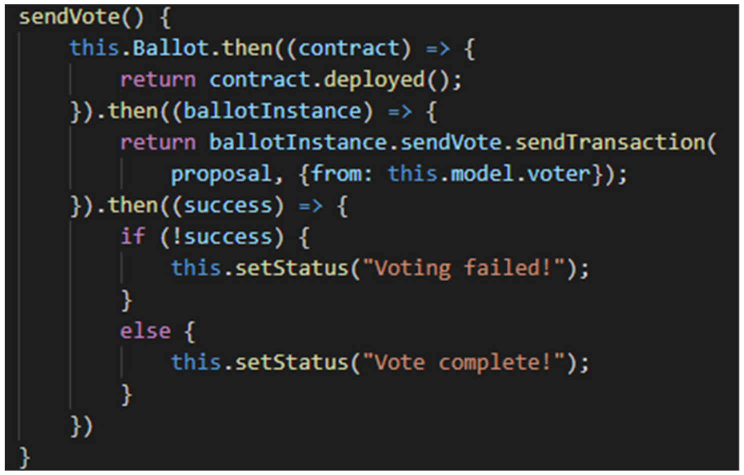
- Executed locally on RPC-server
- Reads information
- Returns a promise (result)



```
refreshVotes() {
    this.Ballot.then((contract) => {
        return contract.deployed();
    }).then((ballotInstance) => {
        return ballotInstance.getVotes.call(
            this.model.proposal);
    }).then((value) => {
        this.model.votes = value.valueOf();
    });
};
```

Transaction

- Executed on EVM in network
- Changes state of assets
- Returns a promise (transaction id)



```
sendVote() {
    this.Ballot.then((contract) => {
        return contract.deployed();
    }).then((ballotInstance) => {
        return ballotInstance.sendVote.sendTransaction(
            proposal, {from: this.model.voter});
    }).then((success) => {
        if (!success) {
            this.setStatus("Voting failed!");
        }
        else {
            this.setStatus("Vote complete!");
        }
    })
}
```

Introducing the Truffle Suite / Boxes

Truffle Suite – From idea to solidity to dapp

- Toolset for developing Smart Contracts
- Truffle
  - Development environment
  - Testing framework
  - Asset pipeline (compilation to deployment)
- Ganache
  - Personal blockchain for Ethereum development
  - Deploy contracts, develop apps, and run tests
- Drizzle
  - Collection of front-end libraries (react) for writing dapps easier
  - Handles synchronization of contract data, transaction data and more

```
contract owned {
    address public
owner;
    constructor()
public {
        owner =
msg.sender;
    }
    modifier
onlyOwner {
require(msg.sender ==
owner);
        _;
    }
    function
transferOwnership(addr
```

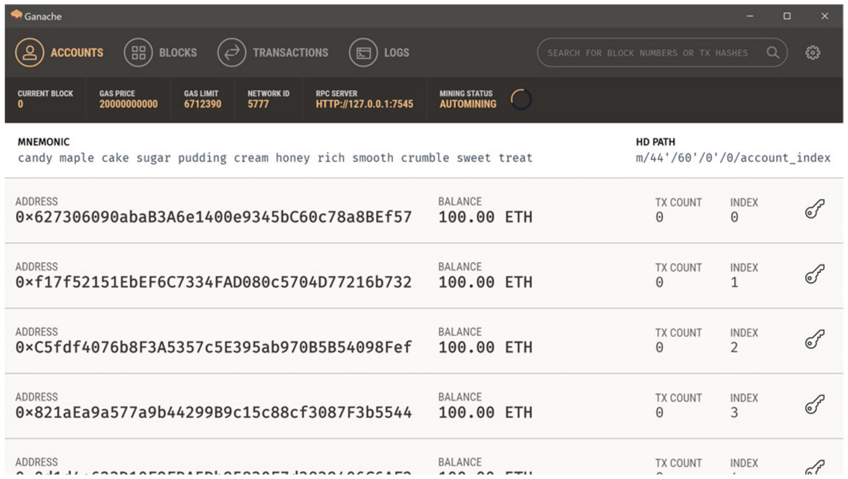
TRUFFLE

AMIS

- Smart contract compilation, linking, deployment and binary management
- Automated contract testing
- Scriptable, extensible deployment & migrations framework
- Network management for deploying to any public & private networks
- Interactive console for direct contract communication.
- Configurable build pipeline with support for tight integration.
- External script runner that executes scripts within a Truffle environment.

Ganache

- One click, personal Ethereum Blockchain (desktop or CLI)
- Status of all accounts (addresses, private keys, transactions, and balance)
- Log output of internal blockchain, incl. responses and other vital debugging
- Configure advanced mining, including setting block times
- Examine all blocks and transactions to gain insight



CURRENT BLOCK	GAS PRICE	GAS LIMIT	NETWORK ID	RPC SERVER	MINING STATUS
0	20000000000	6712390	5777	HTTP://127.0.0.1:7545	AUTOMINING

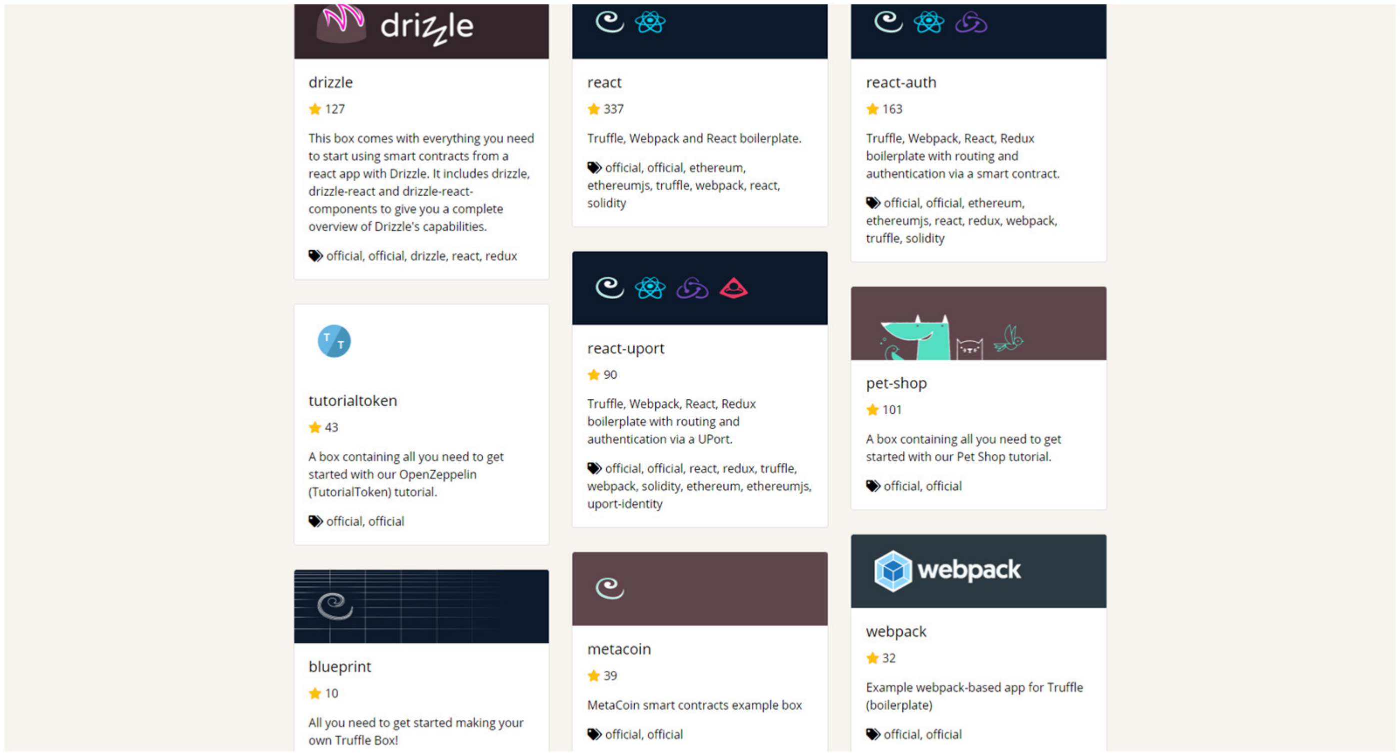
MNEMONIC
candy maple cake sugar pudding cream honey rich smooth crumble sweet treat

HD PATH
m/44'/60'/0'/0/account_index

ADDRESS	BALANCE	TX COUNT	INDEX
0×627306090abaB3A6e1400e9345bC60c78a8BEf57	100.00 ETH	0	0
0×f17f52151EbEF6C7334FAD080c5704D77216b732	100.00 ETH	0	1
0×C5fdf4076b8F3A5357c5E395ab970B5B54098Fef	100.00 ETH	0	2
0×821aEa9a577a9b44299B9c15c88cf3087F3b5544	100.00 ETH	0	3
[illegible]	[illegible]	[illegible]	[illegible]

Truffle Boxes

Boilerplates containing helpful modules, Solidity contracts & libraries, front-end views and more



drizzle
★ 127
This box comes with everything you need to start using smart contracts from a react app with Drizzle. It includes drizzle, drizzle-react and drizzle-react-components to give you a complete overview of Drizzle's capabilities.
official, official, drizzle, react, redux

tutorialtoken
★ 43
A box containing all you need to get started with our OpenZeppelin (TutorialToken) tutorial.
official, official

blueprint
★ 10
All you need to get started making your own Truffle Box!

react
★ 337
Truffle, Webpack and React boilerplate.
official, official, ethereum, ethereumjs, truffle, webpack, react, solidity

react-uport
★ 90
Truffle, Webpack, React, Redux boilerplate with routing and authentication via a UPort.
official, official, react, redux, truffle, webpack, solidity, ethereum, ethereumjs, uport-identity

metacoin
★ 39
MetaCoin smart contracts example box
official, official

react-auth
★ 163
Truffle, Webpack, React, Redux boilerplate with routing and authentication via a smart contract.
official, official, ethereum, ethereumjs, react, redux, webpack, truffle, solidity

pet-shop
★ 101
A box containing all you need to get started with our Pet Shop tutorial.
official, official

webpack
★ 32
Example webpack-based app for Truffle (boilerplate)
official, official

Introducing the Truffle Box for Oracle JET

Truffle Box for Oracle JET

- Box uses Truffle and Ganacha (in future if possible drizzle)

- Boilerplate example based on Oracle JET 6.0
  - Complete JavaScript development toolkit
  - Open source JavaScript libraries and technologies
    (incl. webpack, web components, accessibility support and I18n)
  - Full lifecycle management for template based SPA
  - Advanced two-way binding with a common model layer
  - Powerful routing system supporting single-page application navigation
  - Rich set of UI components and built-in mobile support

- Incorporates simple ballot contract for conference proposals

- Box available at **github.com/robertvanmolken/oraclejet-truffle-box**

How to install Oracle JET Box?

1. Install Truffle, Oracle JET CLI and Ganache CLI globally

```
npm install -g truffle
npm install -g @oracle/ojet-cli
npm install -g ganache-cli
```

2. Install and verify TypeScript for Oracle JET

```
npm install @type/oracle__oraclejet
npm list @type/oracle__oraclejet
```

3. Download the box. This also takes care of dependencies

```
truffle unbox robertvanmolken/oraclejet-truffle-box
```

4. Run the box using instructions in readme:

```
https://raw.githubusercontent.com/robertvanmolken/oraclejet-truffle-box/master/README.md
```

Demo

```
contract owned {
    address public
owner;
    constructor()
public {
    owner =
msg.sender;
    }
    modifier
onlyOwner {
require(msg.sender ==
owner);
    _;
    }
    function
transferOwnership(addr
```