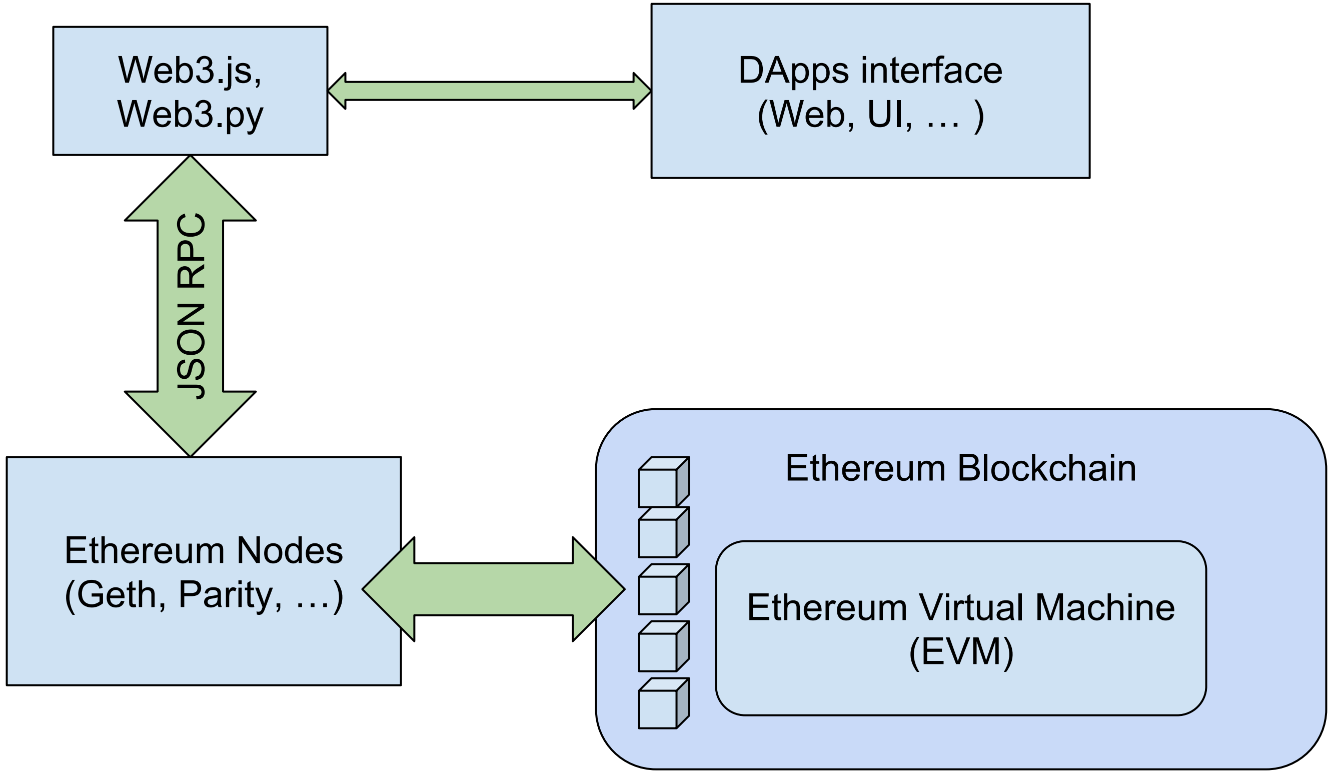
Bootstrap Ethereum Development

Getting your Environment set up and writing your first smart contract

ETHWaterloo

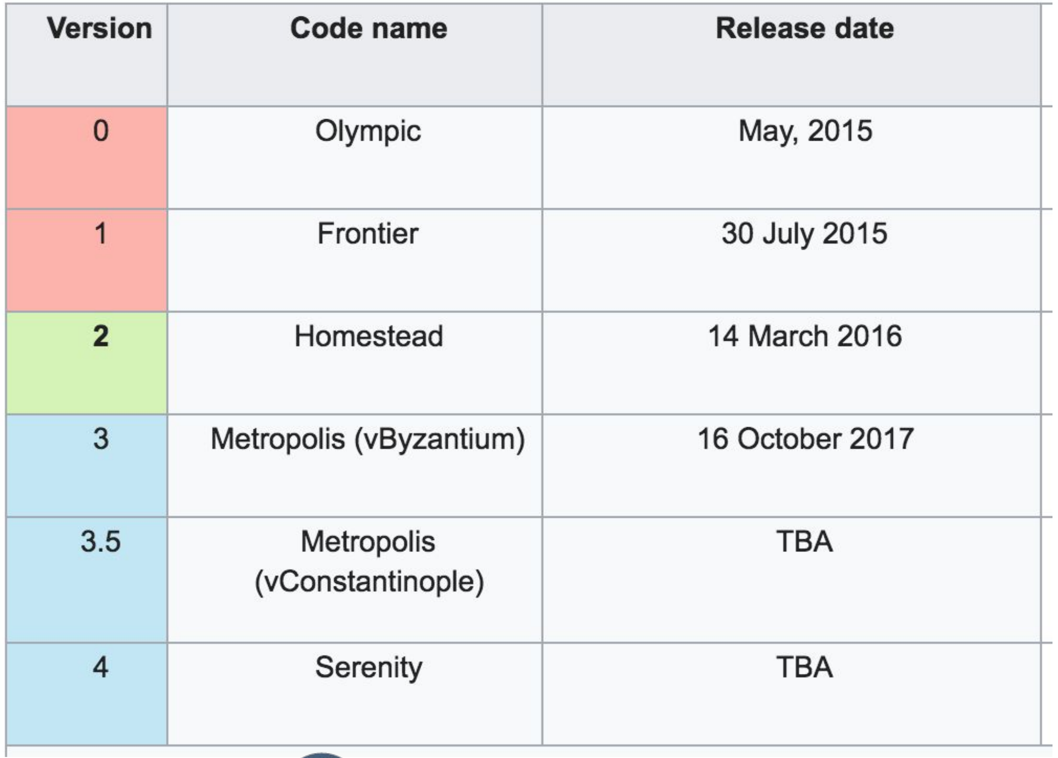
2017 BITACCESS INC
bitaccess.co

Ethereum Infrastructure



bitaccess

Ethereum Blockchain Stages



Version	Code name	Release date
0	Olympic	May, 2015
1	Frontier	30 July 2015
2	Homestead	14 March 2016
3	Metropolis (vByzantium)	16 October 2017
3.5	Metropolis (vConstantinople)	TBA
4	Serenity	TBA

bitaccess

Ethereum Testnets (current)

1. **ROPSTEN** - Proof Of Work ← ~Same as current mainnet Ethereum
2. **KOVAN** - Proof Of Authority (Parity only)
3. **RINKEBY** - Clique Consensus (PoA, Geth only)

https://testnet.etherscan.io/

4. **TestRPC** - Local testnet, restarts on every lunch. Much faster for stand alone smart contract developments

https://github.com/ethereumjs/testrpc

bitaccess

“Traditional” smart contract deployment

1. Install and **Run a full node** (Geth, Parity)
2. Wait to have a **fully synced Node** (takes days usually, unless --fast)
3. **Expose RPC JSON** port of the node (or use Command line web3 interface)
4. (Optional) use Solidity Development frameworks (Truffle, Embark, …)
5. Using Solc (Solidity Compiler) to **compile** Solidity code to bytecode
6. Deploy and fail until magically it works once…

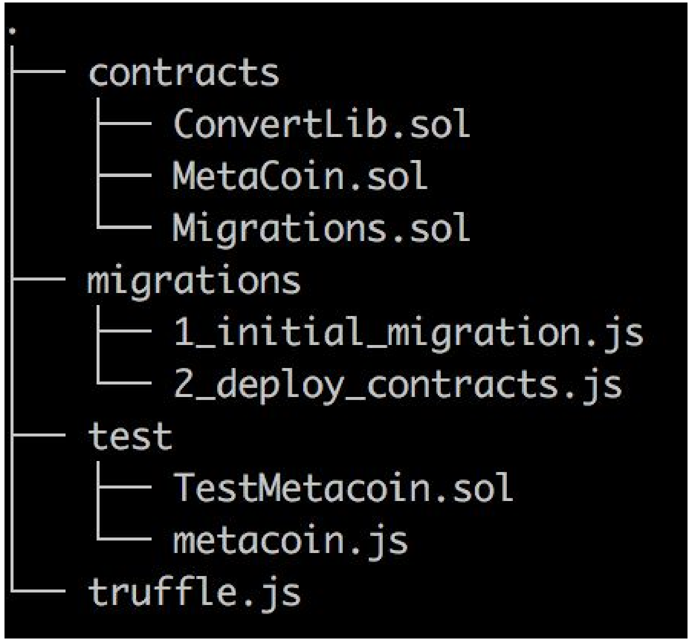
bitaccess

Truffle (The most popular Ethereum development framework)

- http://truffleframework.com
- https://github.com/trufflesuite/truffle

- npm install -g truffle

- truffle init

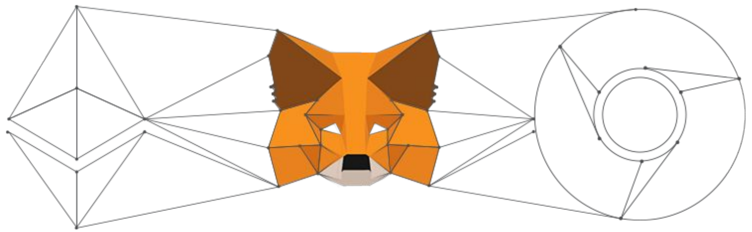


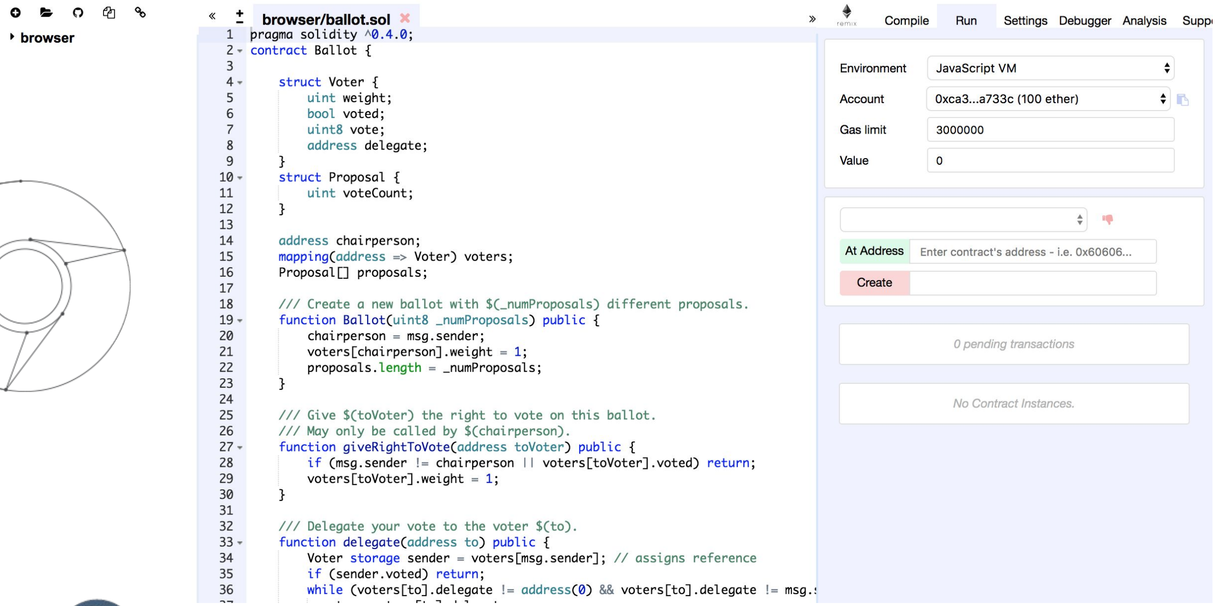
```
.
├── contracts
│   ├── ConvertLib.sol
│   ├── MetaCoin.sol
│   └── Migrations.sol
├── migrations
│   ├── 1_initial_migration.js
│   └── 2_deploy_contracts.js
├── test
│   ├── TestMetacoin.sol
│   └── metacoin.js
└── truffle.js
```

bitaccess

Faster, portable smart contract deployment

1. **Metamask Chrome extension** used instead of running a full node
2. Use Remix (**Browser-Solidity**) for coding, compiling, deploying and triggering functions





```
pragma solidity ^0.4.0;
contract Ballot {

    struct Voter {
        uint weight;
        bool voted;
        uint8 vote;
        address delegate;
    }
    struct Proposal {
        uint voteCount;
    }

    address chairperson;
    mapping(address => Voter) voters;
    Proposal[] proposals;

    /// Create a new ballot with $(_numProposals) different proposals.
    function Ballot(uint8 _numProposals) public {
        chairperson = msg.sender;
        voters[chairperson].weight = 1;
        proposals.length = _numProposals;
    }

    /// Give $(toVoter) the right to vote on this ballot.
    /// May only be called by $(chairperson).
    function giveRightToVote(address toVoter) public {
        if (msg.sender != chairperson || voters[toVoter].voted) return;
        voters[toVoter].weight = 1;
    }

    /// Delegate your vote to the voter $(to).
    function delegate(address to) public {
        Voter storage sender = voters[msg.sender]; // assigns reference
        if (sender.voted) return;
        while (voters[to].delegate != address(0) && voters[to].delegate != msg.s
```

Environment: JavaScript VM
Account: 0xca3...a733c (100 ether)
Gas limit: 3000000
Value: 0

Enter contract's address - i.e. 0x60606...

0 pending transactions

No Contract Instances.

bitaccess

Hands on Demo

- **Browser-Solidity**
    - http://ethereum.github.io/browser-solidity/
    - https://github.com/ethereum/browser-solidity

- **Metamask**
    - https://metamask.io
    - https://github.com/MetaMask/metamask-extension

- **HelloWorld.sol**
    - https://gist.github.com/shayanb/d417cfd229c0980d0fbc2a63dde001a5

bitaccess

TRIED TO PUT YOUR ENTIRE APP IN A SMART CONTRACT?
INSTRUCTOR
YOU'RE GONNA HAVE A BAD TIME
imgflip.com

bitaccess

Web3

How you talk to your smart contract.

Steps to build a DApp in a day

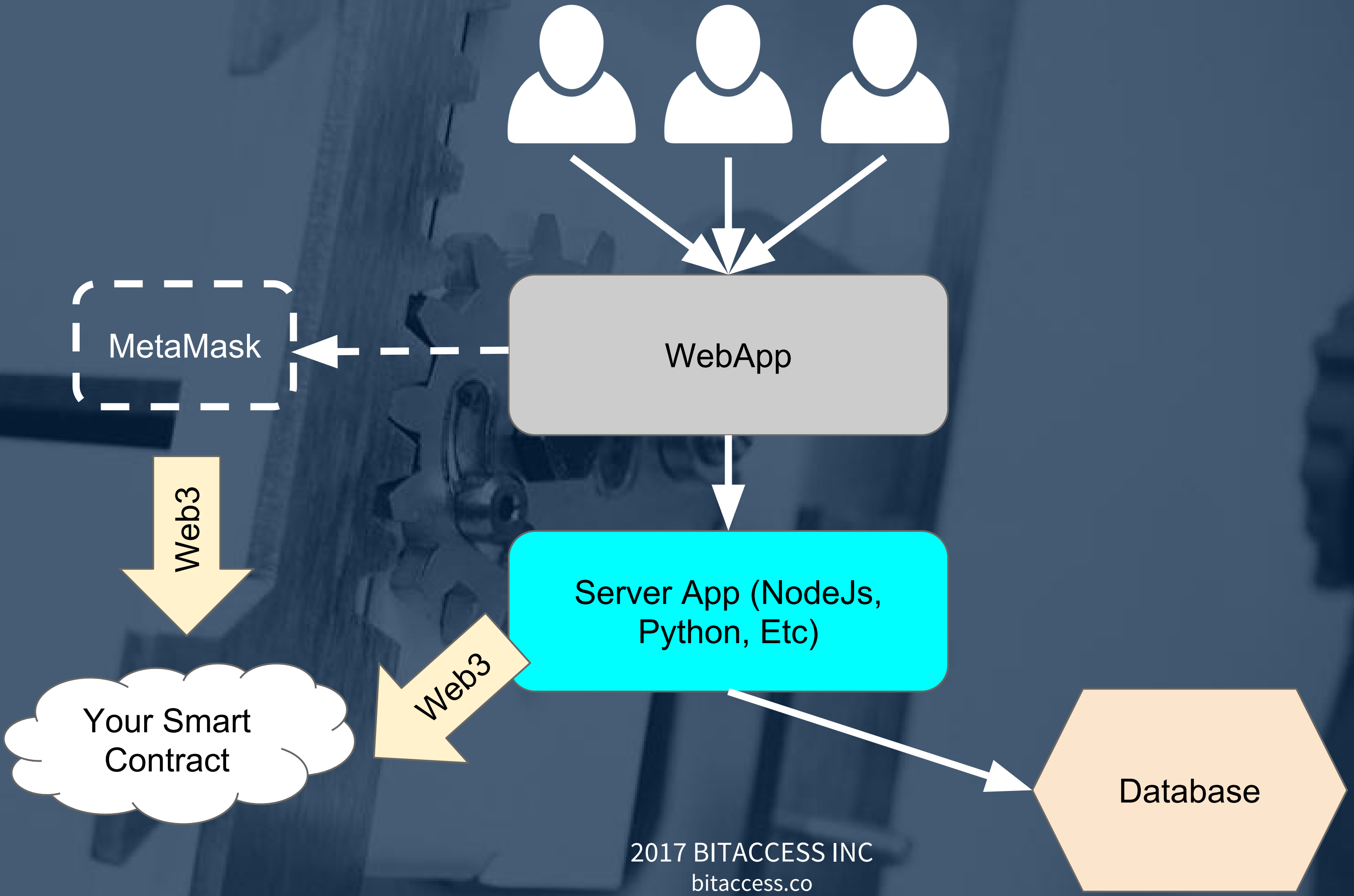
1. Write the App in a language you **know** and **love** *(no blockchain for now)*
2. Write super hacky tests (you will thank yourself later)
3. Put the **<u>simplest possible function</u>** in a smart contract
4. See if you can get the app to work
5. If you are lucky, go to 3

bitaccess

Don’t feel like syncing a node?

- To Connect to web3, you need a synced node
- Feel free to use our public GETH nodes:
  - ropsten rpc: http://45.33.89.56:8545
  - ropsten websocket: ws://45.33.89.56:8546
  - rinkeby rpc: http://45.33.89.56:8547
  - rinkeby websocket: ws://45.33.89.56:8548
  - mainnet rpc: http://45.33.89.56:8541
  - mainnet websocket: ws:[//45.33.89.56:8542](//45.33.89.56:8542)
- (Tell us if they crash)

bitaccess


MetaMask
WebApp
Web3
Server App (NodeJs, Python, Etc)
Web3
Your Smart Contract
Database

Using Web3.js

- **NodeJS is preferred**
  - **npm install --save web3**
  - **https://www.npmjs.com/package/web3**
  - **Docs are here: https://web3js.readthedocs.io/en/1.0/**
  - **https://github.com/ethereum/web3.js**

- **Python**
  - **pip install web3**
  - **Docs are here: https://web3py.readthedocs.io/en/latest/**
  - **Web3.py is a port of web3.js, but is further behind**
  - **https://github.com/pipermerriam/web3.py**

bitaccess

Using Web3.js

- You can either connect through RPC or Websockets
- Websockets have (buggy) notifications
  - Things you can get notified about:
    - Pending Transactions (all of them, you will need to filter)
    - New Block Headers
  - Warning! Parity and GETH have different Web3 Interfaces. If you are using Websockets, use GETH with web3.js
- Use RPC unless you absolutely need notifications

bitaccess

Using Web3.js

- Connecting to Web3:

```
var Web3 = require('web3')
var web3 = new Web3(process.env.WEB3_WEBSOCKET_URL || 'ws://45.33.89.56:8546/')
web3.eth.subscribe('pendingTransactions', function (error, transaction) {
  if (error) {
    console.log('pendingTransactions error', error)
  }
}).on('data', function (transaction) {
  web3.eth.getTransaction(transaction, function (err, tx) {
    console.log('Details for', transaction, tx, err && err.message)
  })
}).on('error', function (transaction) {
  console.log('pendingTransactions error', transaction)
})
```

We’re Hiring
We’re Canadian

https://angel.co/bitaccess/jobs