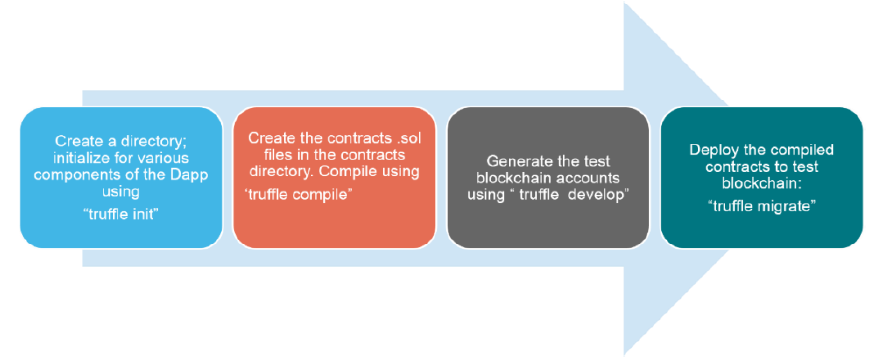
DEMO





# Remix IDE

1. Show Remix IDE and talk about the tools
   1. In (web)browser IDE: with integrated dev. Environment
2. SOLIDITY COMPILER
   1. Select the compiler version
   2. Language
   3. EVM Version
   4. Compile -> artifacts are visible
3. DEPLOY & RUN TRANSACTIONS
   1. Environment -> Javascript VM, Injected Web3, Web3 Provider
   2. Deploy and address options
4. Start explaining the sample contracts

# Truffle Framework

Truffle is a complete dev. Framework for DApp – contracts, test (.sol, & .js),

$ truffle init - boiler plate

$ truffle compile - compiles the contracts and generate artifacts

$ truffle test – run the tests – solidity and js – uses truffle develop testRPC or run Ganache

$ truffle develop – runs a testRPC

$ truffle migrate --reset – deploys contracts on the Ethereum TestRPC/Ganache

# Ganache TestRPC

Ethereum simulator – give the address with balance

# Metamask Wallet

Light weight crypto wallet – a chrome extension, best suited for development

# Front-end Development

* React JS/NodeJS
* HTML/JS/CSS

# Truffle Boxes

$ mkdir dapp && cd dapp

$ truffle unbox metacoin

$ truffle compile

$ truffle migrate –reset

$ truffle test

$ cd app && npm run dev

$ mkdir dappreact

$ npx truffle unbox metacoin

$ add a example smart contract – Ballot.sol / Coin.sol

$ add migration script – migration/2\_deploy\_contract.js

2\_deploy\_contracts.js

var Ballot = artifacts.require("./Ballot.sol");

module.exports = function(deployer) {

  deployer.deploy(Ballot);

};

Add to truffle-config.js

const path = require("path");

module.exports = {

  // See <http://truffleframework.com/docs/advanced/configuration>

  // to customize your Truffle configuration!

  contracts\_build\_directory: path.join(\_\_dirname, "client/src/contracts"),

  networks: {

    development: {

      host: "127.0.0.1",     // Localhost (default: none)

      port: 8545,            // Standard Ethereum port (default: none)

      network\_id: "\*",       // Any network (default: none)

     },

    develop: {

      port: 8545

    }

  }

};

1. Deploy smart contract with Remix IDE

pragma solidity ^0.5.14;

contract SomeContract {

uint public myUint = 10;

function setUint(uint \_myUint) public {

myUint = \_myUint;

}

}

1. Choose the Web3-Provider in Remix! Port 8545 if you are using Ganache Then Deploy the Smart Contract.
   1. Install packaged-version of web3.js

Inside your empty directory install web3js packaged for browser-usage:

“npm install web3.js-browser”

* 1. Create a new index.html file

Inside the root folder (where your node\_modules folder is) create a new index.html file with the following content:

<!DOCTYPE html>

<html>

<head>

<meta charset='utf-8'>

<meta http-equiv='X-UA-Compatible' content='IE=edge'>

<title>My Website</title>

<meta name='viewport' content='width=device-width, initial-scale=1'>

<script src='node\_modules/web3.js-browser/build/web3.js'></script>

</head>

<body>

</body>

</html>

* 1. Open the file in your browser (Chrome or Firefox)

It should be a completely blank page. Then open the developer console in your browser (F12). And enter a few commands to see if you can connect to Ganache:

var web3 = new Web3(Web3.providers.HttpProvider("http://localhost:8545"));

web3.eth.getAccounts().then(console.log);

* 1. Update myUint using the ABI Array

Let’s do the same as we did before in the browser and see if it still works!

Copy the ABI Array from Remix:

And enter the following code in node:

var myContract = new web3.eth.Contract(PASTE\_ABI\_ARRAY\_HERE, 'CONTRACT\_ADDRESS');

Then simply call via a very declarative function name:

myContract.methods.myUint().call().then(console.log).catch(console.error);





myContract.methods.setUint(50).send({from:

'FIRST\_ACCOUNT\_FROM\_GANACHE'}).then(result => {console.log(result);

myContract.methods.myUint().call().then(console.log);}).catch(console.error);