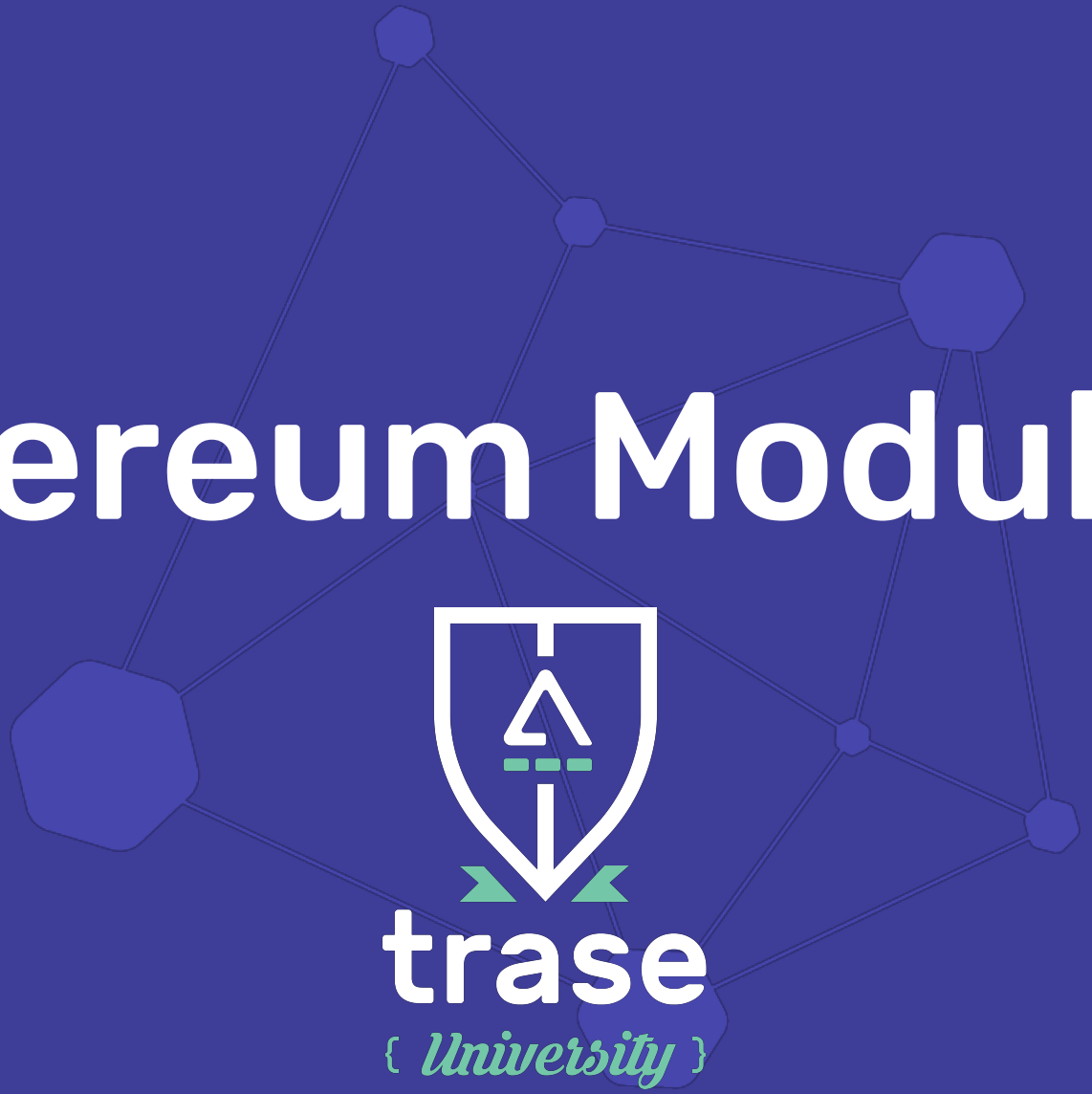
Ethereum Module III



Overview

Web3

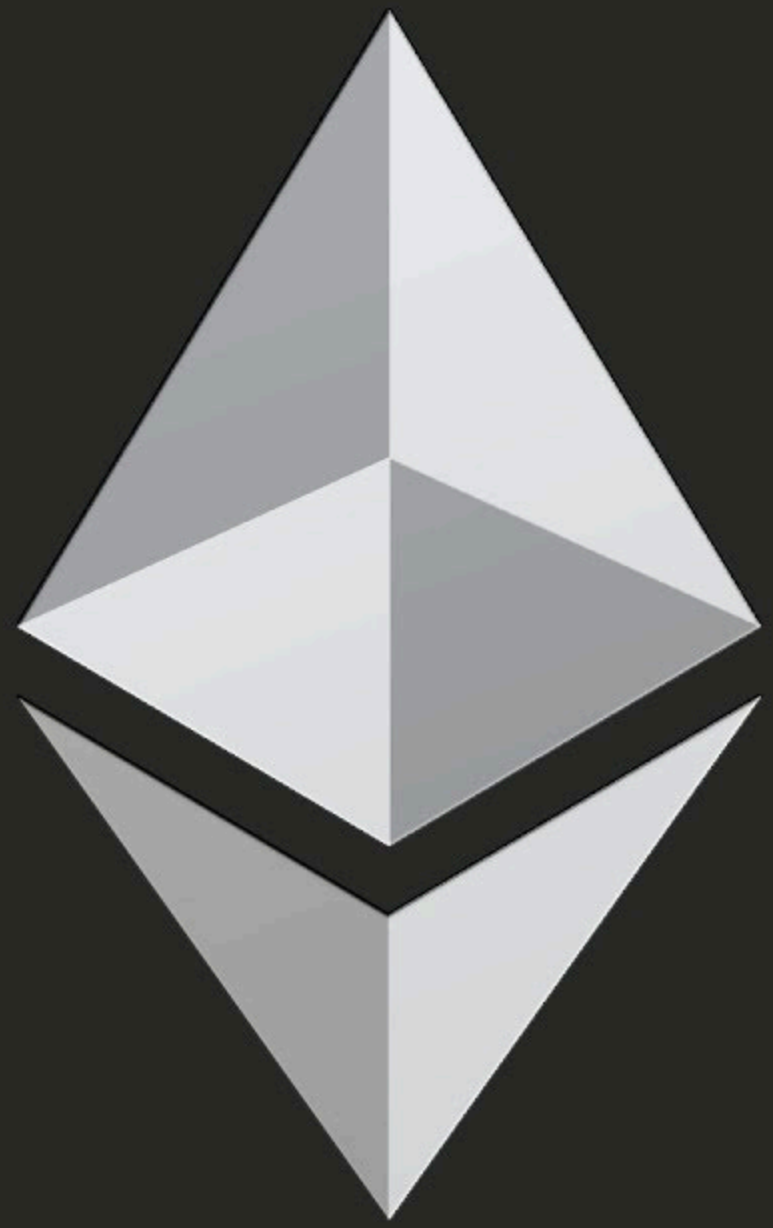
How to interact with a smart contract

Basics

Hands-on w/ Web3

Shipping Smart Contract

trAse



How does the EVM work?

trAse

EVM Bytecode

```
6060604090815260028054 60a060020a60ff02191690556101c09051908101604090815260 3c8252607860208301526101 2c90820152610258606082015261070860808201
52610e1060a0820152611c2060c082015261384060e0820152617080610100820152 61e10061012082015262015180610140820152620 2a3006101608201526205460061
018082015262093a806101a0820152620000a790600390600e620004e4565b50600f60055566071afd498d0000600e553415620000c457600080fd5b60028054600080546
00160a060020a033316600160a060020a0319918216811783556 0a060020a60ff02199093167401000000000000000000000000000000000000000017169091179091556 2
00012f90808060001981640100000000620028f062000136820217045 65b5062000649565b600080620001436200058756 5b600063ffffffff8916891462000158576000 8
0fd5b63ffffffff881688146200016b57600080fd5b61ffff871687146200017c57600080fd5b600287049250600d8361ffff161115620001955760 0d92505b6101006040
5190810160409081528782526001604060020a034216602083015260009082018190526 3ffffffff808c1660608401528a1660808301526 0a082015261ffff80851660c08
30152881660e0820152600680549193506001918083016200020183826200 05cb565b600092835260209092208591600202018151815560208201516001820180546 7ffff
fffffffffff1916600160406002 0a0392909216919091179055604082015181600101600861010 00a81548160016040600 20a030219169083600160406002 0a031602179
055506060820151816001016010610 1000a81548163ffffffff02191690 5b7f0a5311bd2a6608f08a180df2ee7c5946819a649b204b554bb8e39 8
8363ffffffff16021790555060a0820151816001016018610 1000a81548163ffffffff02191690 8363ffffffff16021790555060c0820151816001016 01c6101000a81548
161ffff021916908361ffff16021790555060e0820151600190910180546 1ffff929092167e0100000000000000000000000000000000000000000000000000000000000000
02600160f060020a0390921691909117905550039050 63ffffffff811681146200035e57600080fd5b7f0a5311bd2a6608f08a180df2ee7c5946819a649b204b554bb8e39
825b2c50ad58582846060015163ffffffff16856080015163ffffffff16865160405160016 0a060020a0390951685526020850193909352604080850192909252606084 01
52608083019190915260a0909101905180910390a1620003ef60008683640100000000620025e0620003fb82021704565b9897505050505050505050565b600160a060020a0
380831660008181526008602090815260408083208054600101905585835260079091529020805460016 0a060020a0319169091179055831615620004 8f57600160a06002
0a0383166000908152600860209081526040808320805460001901905583835260 0a8252808320805460016 0a060020a0319908116909155600990925290912080549011
690555b7fddf252ad1be2c89b69c2b068fc378daa952ba7f163c4a11628f55a4df523b3ef8383836040516001 60a060020a0393841681529190921660208201526040808 2
0192909252606001905180910390a1505050565b60028301918390821562000575579160200282016000 5b838211562000541578351838261010 0a81548163ffffffff0
2191690836 3ffffffff160217905550926020019260040160208160030104928301926001030262000 4fb565b8015620005735782816101000a81549063ffffffff021916
90556004016020816003010492830192600103026200054 1565b505b506200058392915062000 5ff565b5090565b6101006040519081016040908152600080835260208 30
```

trAse

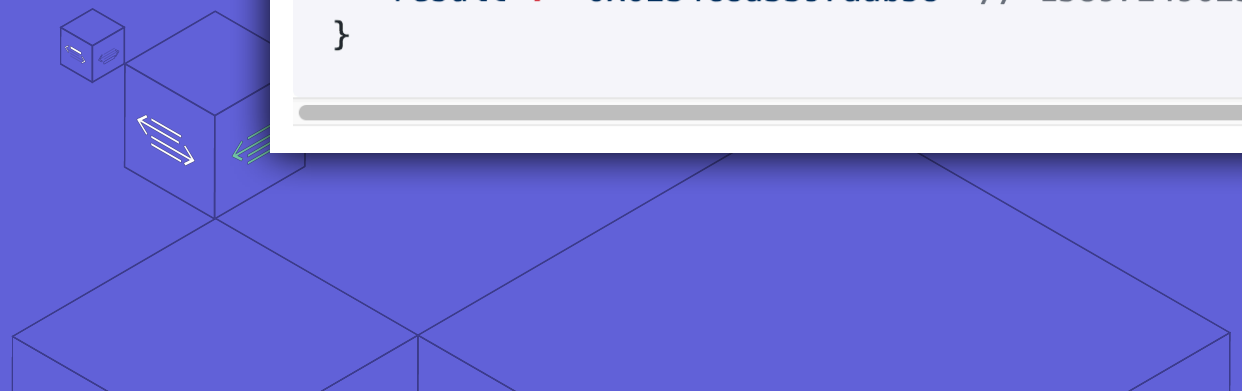
Web3.js?

trAse

Web3.js - RPC “Middleware”

- Low level JSON API
- Params pre-encoded

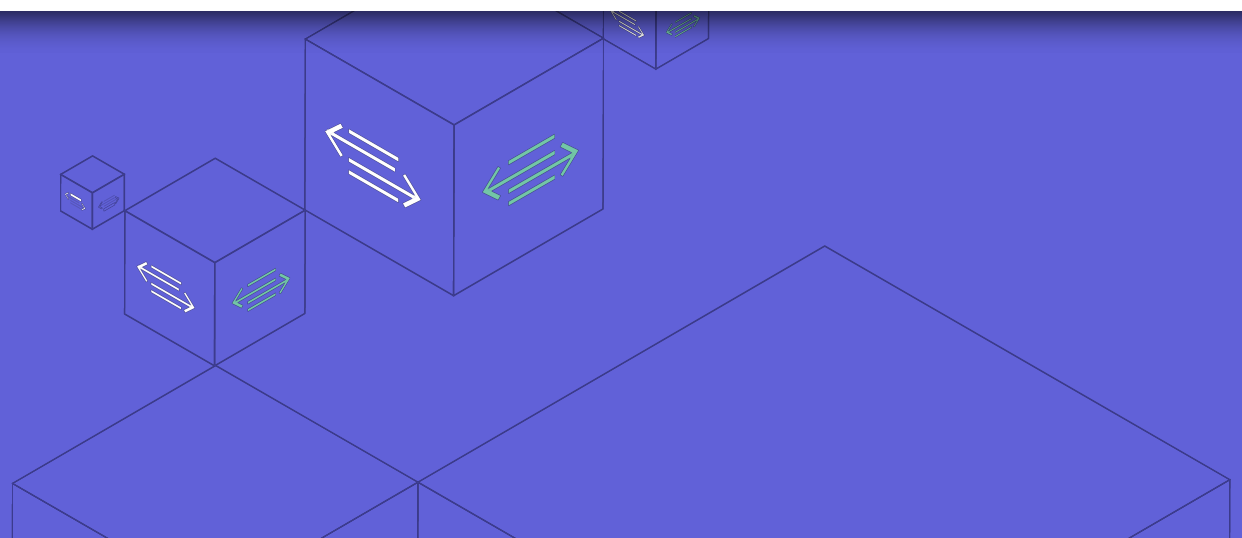
```
// Request
curl -X POST --data '{"jsonrpc":"2.0","method":"eth_getBalance","params":["0x407d73c

// Result
{
  "id":1,
  "jsonrpc": "2.0",
  "result": "0x0234c8a3397aab58" // 158972490234375000
}
```



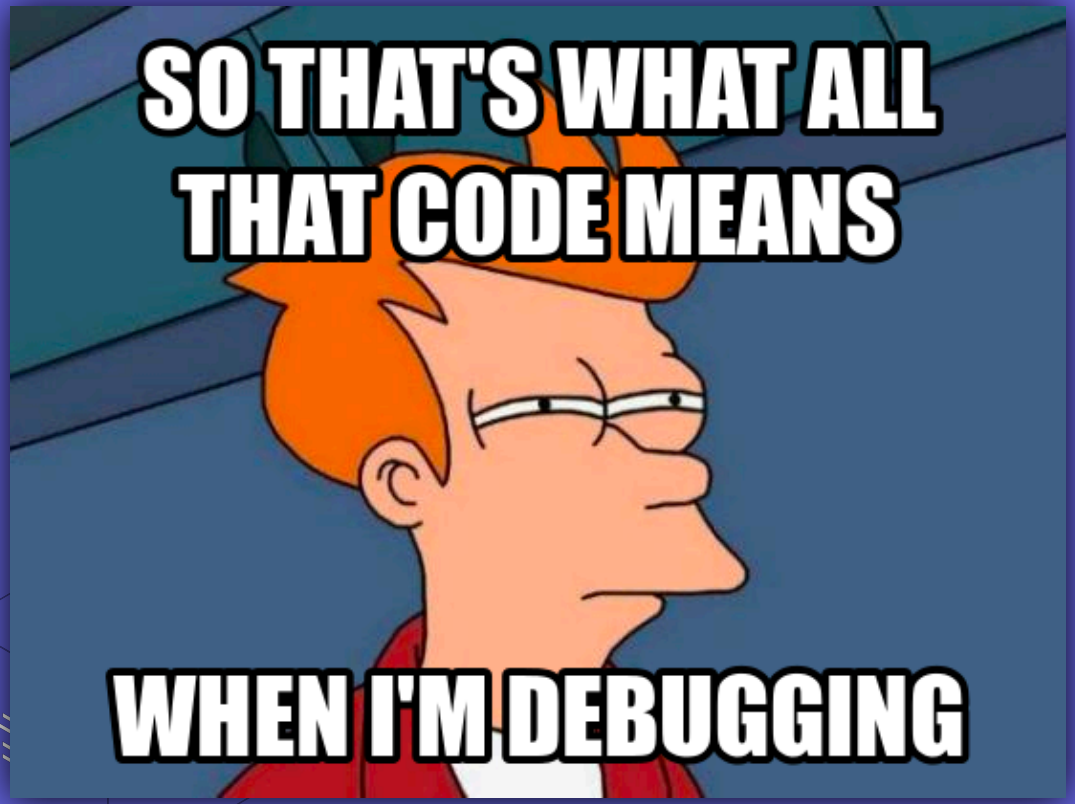
trAse

How is the encoding done?

```
Functie: createRandomZombie
Parameter - string: “Mauro”
web3.sha3(“createRandomZombie(string)”).substr(0,10)  -> 0x7bff0a01
String to hex: “Mauro”  -> 4d6175726f

Encoded data:
0x7bff0a0100000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
0000000000000004d6175726f000000000000000000000000000000000000000000000000000000000000000000
```



Why mention Bytecode?



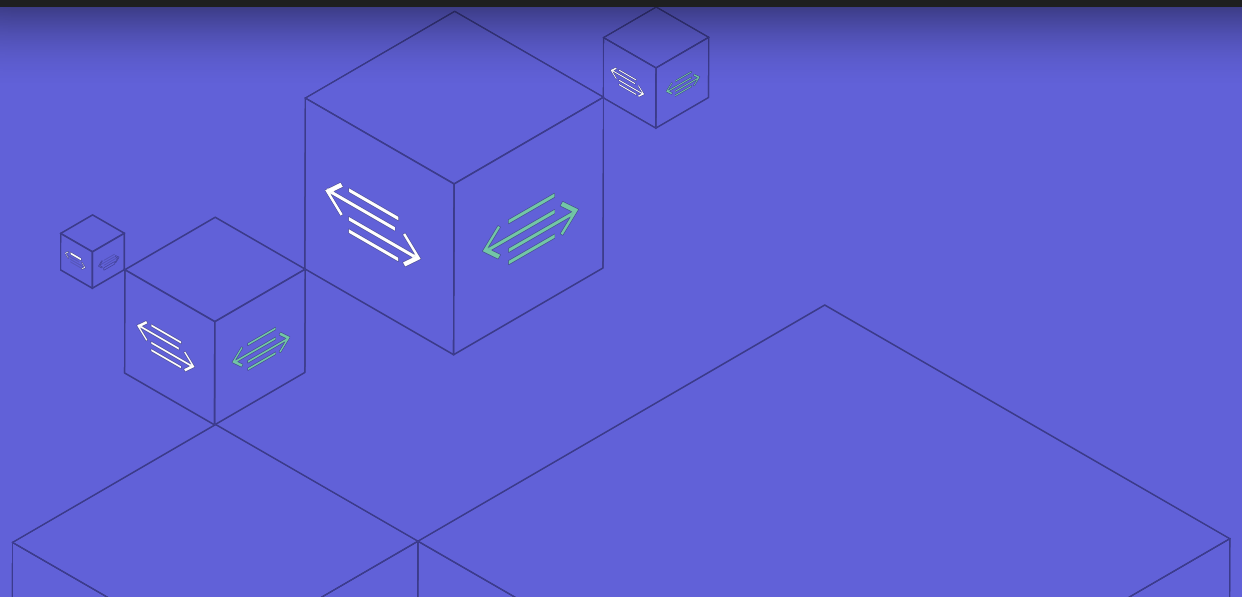
trAse

Actual Web3.js API

trAse

But first... Web3 - Promises

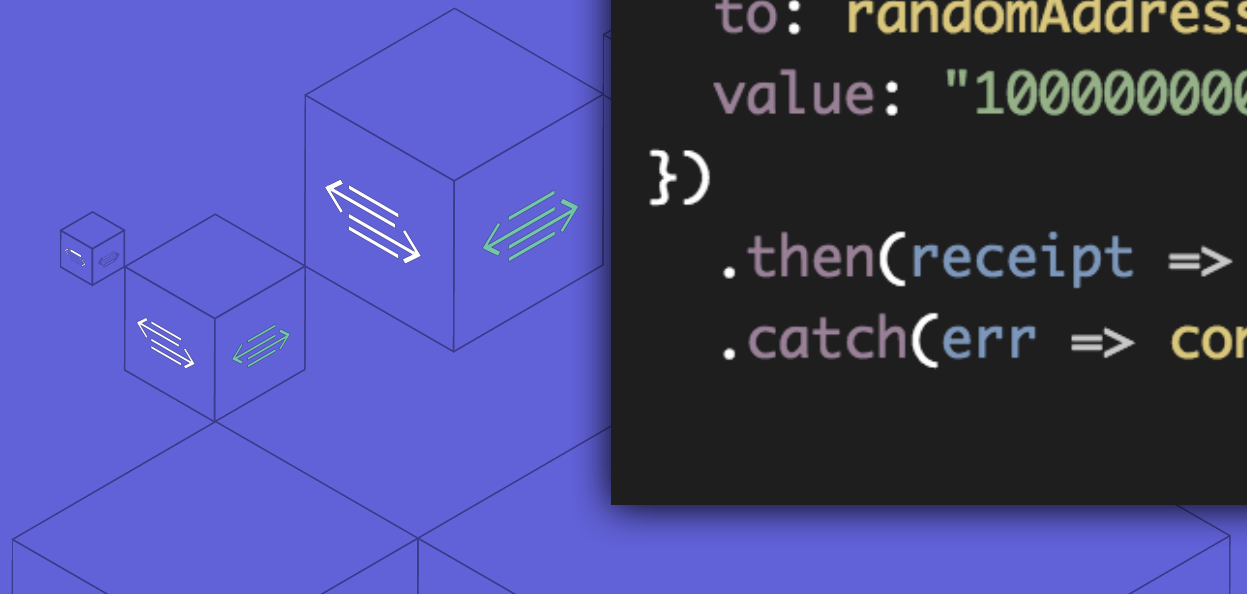
```
web3.eth.getBalance(randomAddress)
   .then(res => console.log(web3.utils.fromWei(res, 'ether')))
--> 0.03 (ether)
```



trAse

Web3 - Promises problem

- What to return?
  - txHash
  - Receipt
  - Confirmation X
  - ....



```
web3.eth.sendTransaction({
   from: myAddress,
   to: randomAddress,
   value: "1000000000" // value is in wei
})
   .then(receipt => console.log(receipt))
   .catch(err => console.log(err));
```

trAse

Web3 - “PromiEvents”

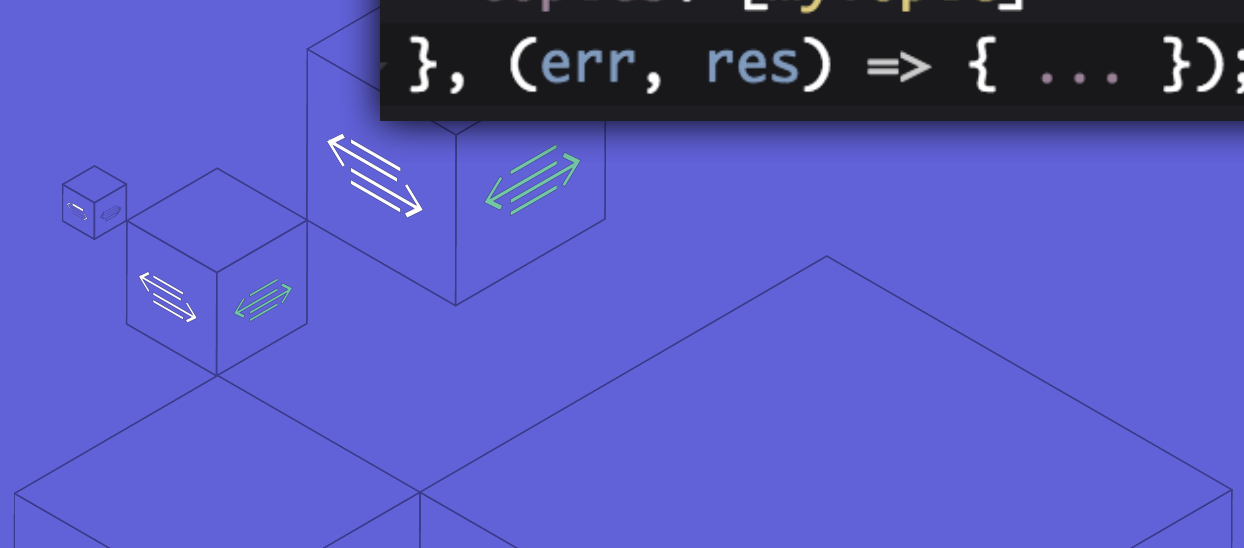
```
web3.eth.sendTransaction({
  from: myAddress,
  to: randomAddress,
  value: "1000000000" // value is in wei
})
  .on('transactionHash', transactionHash => console.log(transactionHash))
  .on('receipt', receipt => console.log(receipt))
  .on('confirmation', (confirmationNumber, receipt) => console.log(`${confirmationNumber} - ${receipt}`))
  .on('error', error => console.log(error))
```

trAse

Web3 - Websockets

- IPC Sockets
- Web sockets

```
const subscription = web3.eth.subscribe("logs", {
  address: myAddress,
  topics: [myTopic]
}, (err, res) => { ... });
```

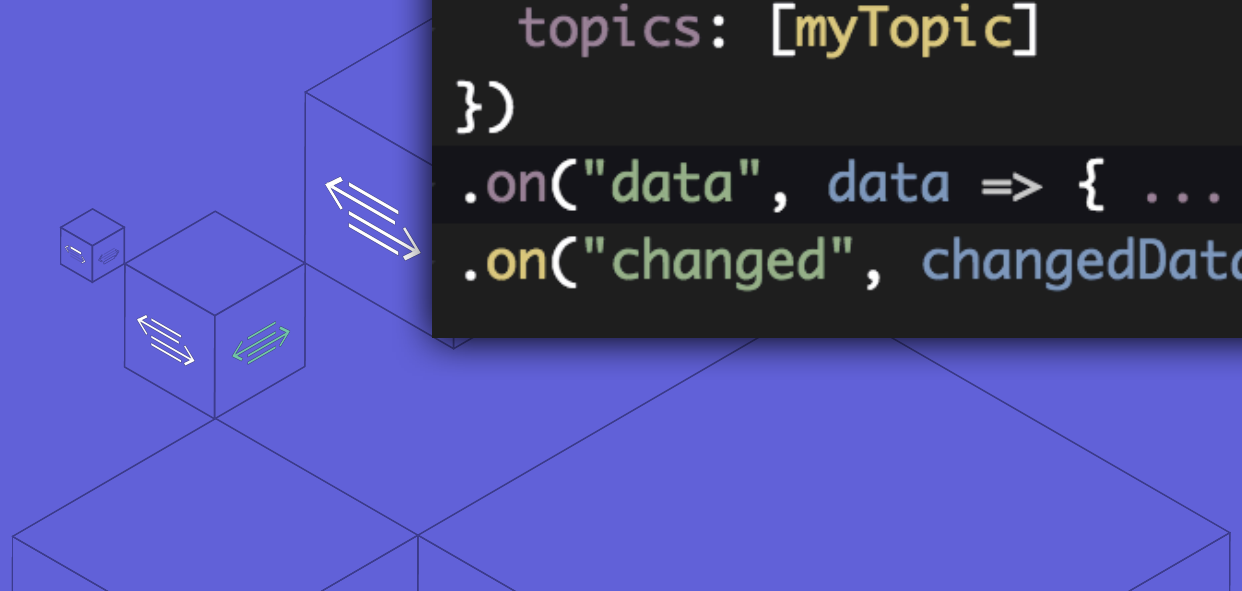


trAse

Web3 - Websockets

- pendingTransaction
- newBlockHeaders
- Syncing
- Logs

```
const subscription = web3.eth.subscribe("logs", {
  address: myAddress,
  topics: [myTopic]
})
.on("data", data => { ... })
.on("changed", changedData => { ... });
```



trAse

Web3 API Overview

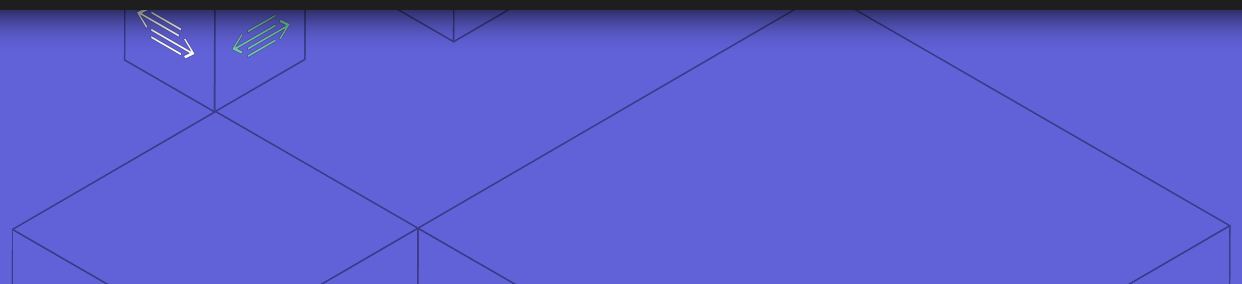
- web3
  - web3.eth
  - web3.eth.subscribe
  - web3.eth.Contract
  - web3.eth.accounts
  - web3.eth.personal
  - web3.eth.Iban
  - web3.eth.abi
  - web3.*.net
  - web3.bzz
  - web3.shh
  - web3.utils

trAse

Web3.eth.Contract

- Creation of a Javascript Smart Contract Object from a compiled Smart Contract

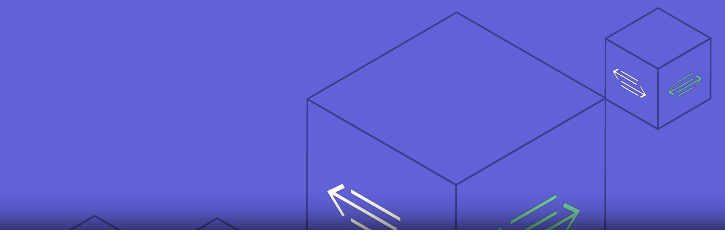
```
const myContract = new web3.eth.Contract(jsonInterface, addressSCToCall, {
    from: myAddress,
    gasPrice: '20000000000' // default gas price in wei, 20 gwei in this case
});
```



trAse

Web3.eth.Contract

- Encoded attributes!!
- Methods
- Events



```
events:
  0xc4b32d7ff75a51e487f3cbb5d3202f2efecca1f4c2b987e0039c1d3220b48a83: ƒ ()
  Transfer: ƒ ()
  Transfer(address,address,uint256,string,uint256): ƒ ()
  allEvents: ƒ ()
  __proto__: Object
```

```
methods:
  0x00d2285c: ƒ ()
  0x2ef60932: ƒ ()
  0x5a348812: ƒ ()
  0x421b2d8b: ƒ ()
  0x3786ce8a: ƒ ()
  0xbefe68c7: ƒ ()
  0xd8ba95a0: ƒ ()
  addAssignableCoins: ƒ ()
  addAssignableCoins(address,uint256): ƒ ()
  addUser: ƒ ()
  addUser(address): ƒ ()
  assignableCoins: ƒ ()
  assignableCoins(address): ƒ ()
  burnCoin: ƒ ()
  burnCoin(address,uint256): ƒ ()
  sendCoin: ƒ ()
  sendCoin(address,uint256,string): ƒ ()
  setAssignableCoins: ƒ ()
  setAssignableCoins(address,uint256): ƒ ()
  spendableCoins: ƒ ()
  spendableCoins(address): ƒ ()
  __proto__: Object
```

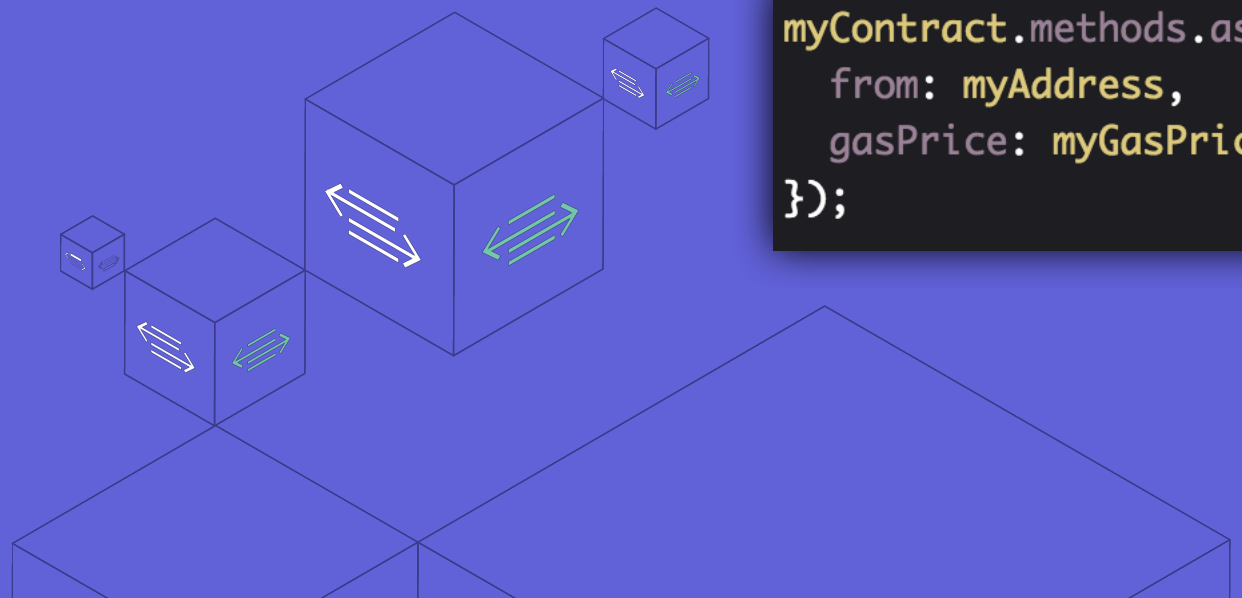
trase

Web3.eth.Contract - Methods

- Call
- Send
- EstimateGas
- EncodeABI

```
myContract.methods.setAssignableCoins(myAddress, 5).send({
  from: myAddress,
  gasPrice: myGasPrice,
  value: myValue
});
```

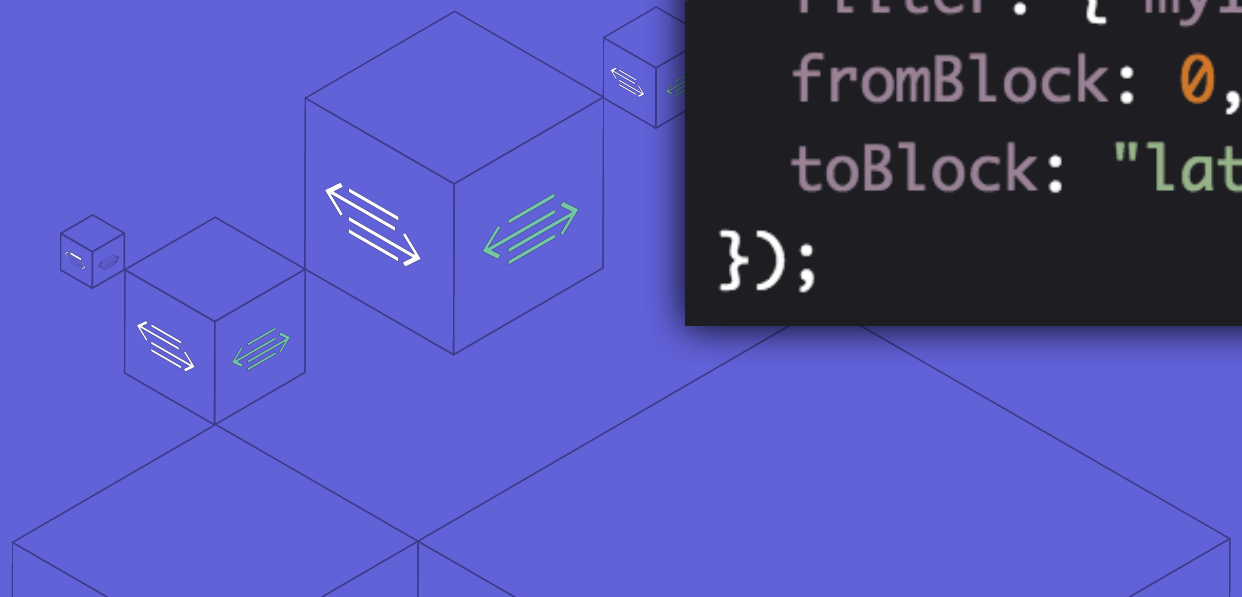
```
myContract.methods.assignableCoins(myAddress).call({
  from: myAddress,
  gasPrice: myGasPrice
});
```



trAse

Web3.eth.Contract - Events

- Filter on indexed parameters
- Choose your block range



```
myContract.events.Transfer({
    filter: { myIndexedParam: [1,2] },
    fromBlock: 0,
    toBlock: "latest"
});
```

trAse

Web3.eth.accounts

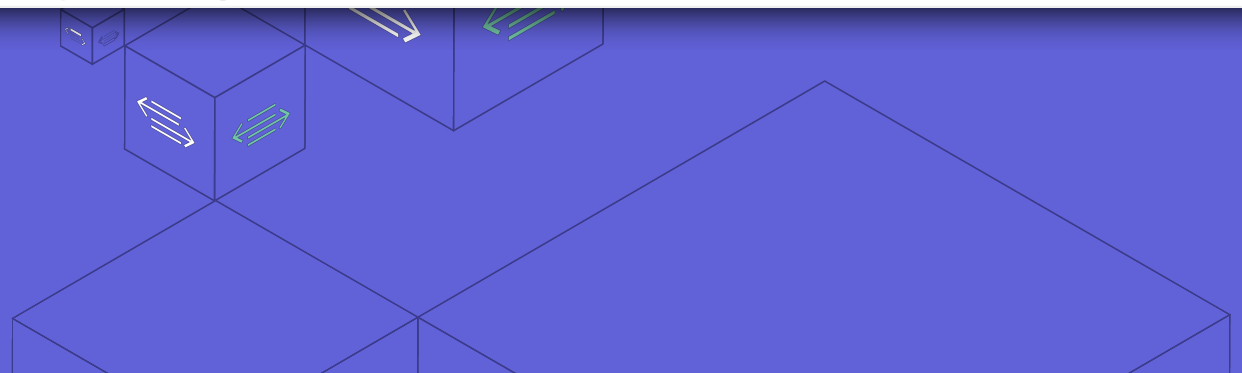
- Accounts toolbox
- Encrypting/Decrypting
- Signing
- …

```
▼ __proto__:
  ▶ create: ƒ create(entropy)
  ▶ decrypt: ƒ (v3Keystore, password, nonStrict)
  ▶ encrypt: ƒ (privateKey, password, options)
  ▶ hashMessage: ƒ hashMessage(data)
  ▶ privateKeyToAccount: ƒ privateKeyToAccount(privateKey)
  ▶ recover: ƒ recover(message, signature, preFixed)
  ▶ recoverTransaction: ƒ recoverTransaction(rawTx)
  ▶ sign: ƒ sign(data, privateKey)
  ▶ signTransaction: ƒ signTransaction(tx, privateKey, callback)
  ▶ _addAccountFunctions: ƒ (account)
  ▶ constructor: ƒ Accounts()
  ▶ __proto__: Object
```

trΛse

Web3.eth.abi - Encoding

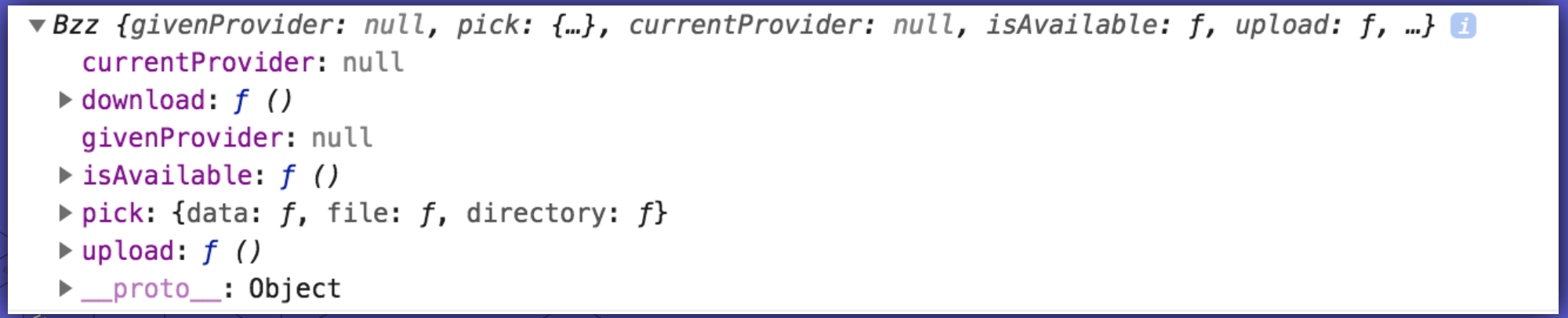
```
▼ ABICoder {_types: Array(7)} i
  ▶ _types: (7) [SolidityTypeAddress, SolidityTypeBool, SolidityTypeInt, SolidityTypeUInt, SolidityTypeDynamicBytes, SolidityTyp
  ▼ __proto__:
    ▶ decodeLog: ƒ (inputs, data, topics)
    ▶ decodeParameter: ƒ (type, bytes)
    ▶ decodeParameters: ƒ (outputs, bytes)
    ▶ encodeEventSignature: ƒ (functionName)
    ▶ encodeFunctionCall: ƒ (jsonInterface, params)
    ▶ encodeFunctionSignature: ƒ (functionName)
    ▶ encodeParameter: ƒ (type, param)
    ▶ encodeParameters: ƒ (types, params)
    ▶ _encodeMultiWithOffset: ƒ (types, solidityTypes, encodeds, dynamicOffset)
    ▶ _encodeWithOffset: ƒ (type, solidityType, encoded, offset)
    ▶ _getOffsets: ƒ (types, solidityTypes)
    ▶ _getSolidityTypes: ƒ (types)
    ▶ _requireType: ƒ (type)
    ▶ constructor: ƒ (types)
    ▶ __proto__: Object
```

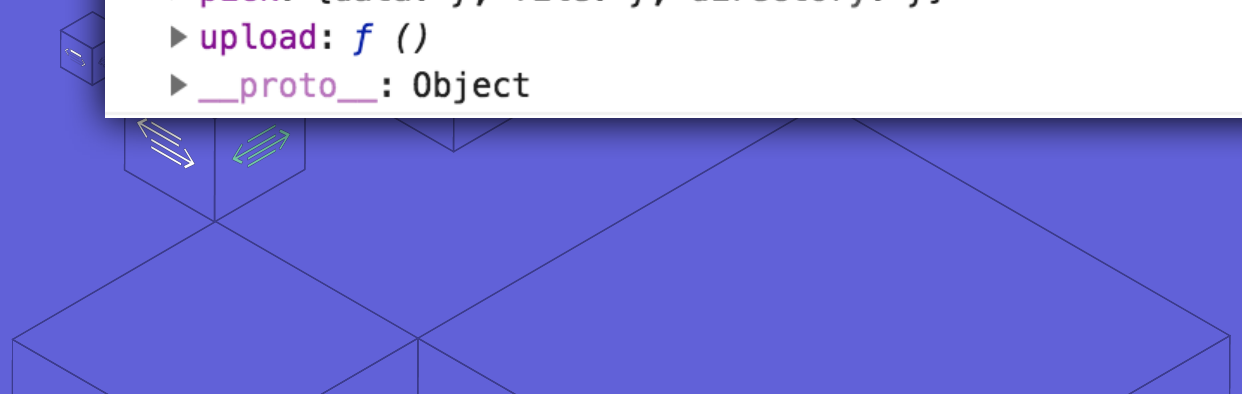


trÆse

Web3.bzz - Swarm Integration

- Decentralised File Storage
- Download/upload - retrieve hash
- File picker





trÂse

Web3.utils

- Useful conversions
- Validity checks

```
{_fireError: ƒ, _jsonInterfaceMethodToString: ƒ, randomHex: ƒ, _: ƒ, BN: ƒ, …}
  BN: ƒ BN(number, base, endian)
  asciiToHex: ƒ (str)
  bytesToHex: ƒ (bytes)
  checkAddressChecksum: ƒ (address)
  fromAscii: ƒ (str)
  fromDecimal: ƒ (value)
  fromUtf8: ƒ (str)
  fromWei: ƒ (number, unit)
  hexToAscii: ƒ (hex)
  hexToBytes: ƒ (hex)
  hexToNumber: ƒ (value)
  hexToNumberString: ƒ (value)
  hexToString: ƒ (hex)
  hexToUtf8: ƒ (hex)
  isAddress: ƒ (address)
  isBN: ƒ (object)
  isBigNumber: ƒ (object)
  isHex: ƒ (hex)
  isHexStrict: ƒ (hex)
  keccak256: ƒ (value)
  leftPad: ƒ (string, chars, sign)
  numberToHex: ƒ (value)
  padLeft: ƒ (string, chars, sign)
  padRight: ƒ (string, chars, sign)
  randomHex: ƒ (size, callback)
  rightPad: ƒ (string, chars, sign)
  sha3: ƒ (value)
  solidtySha3: ƒ ()
  stringToHex: ƒ (str)
  toAscii: ƒ (hex)
  toBN: ƒ (number)
  toChecksumAddress: ƒ (address)
  toDecimal: ƒ (value)
  toHex: ƒ (value, returnType)
  toTwosComplement: ƒ (number)
  toUtf8: ƒ (hex)
  toWei: ƒ (number, unit)
  unitMap: {noether: "0", wei: "1", kwei: "1000", Kwei: "1000", babbage: "1000", …}
  utf8ToHex: ƒ (str)
  _: ƒ (obj)
  _fireError: ƒ (error, emitter, reject, callback)
  _jsonInterfaceMethodToString: ƒ (json)
  __proto__: Object
```

trAse

How to dApp

trAse

Prerequisites

- Truffle framework (Installed globally through NPM)
- Ganache UI - https://trufflframework.com/ganache
- Metasmask - https://metamask.io/
- Any Javascript frontend boilerplate of your choice

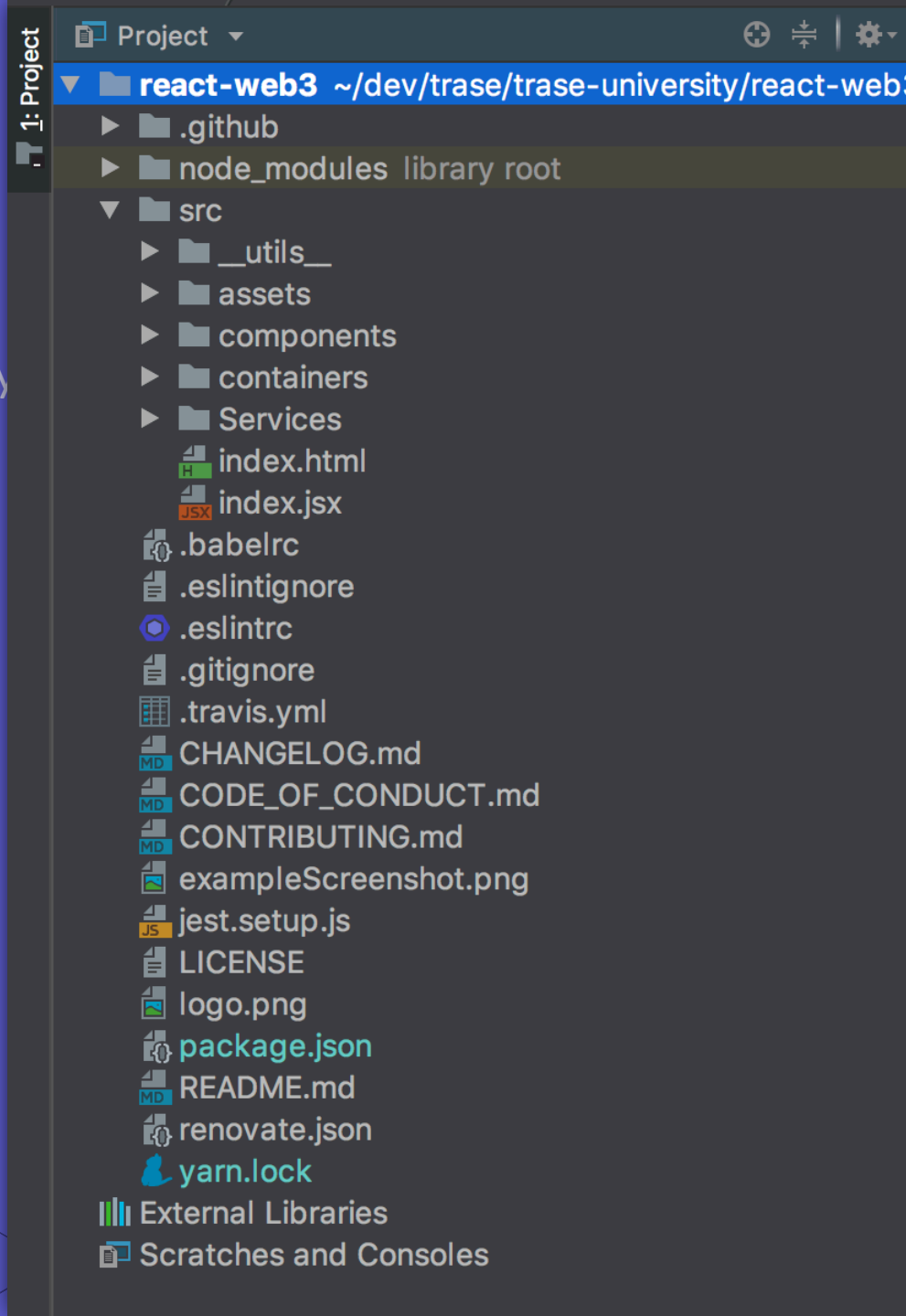
```
npm install -g truffle

yarn global add truffle
```

trAse

1. Frontend

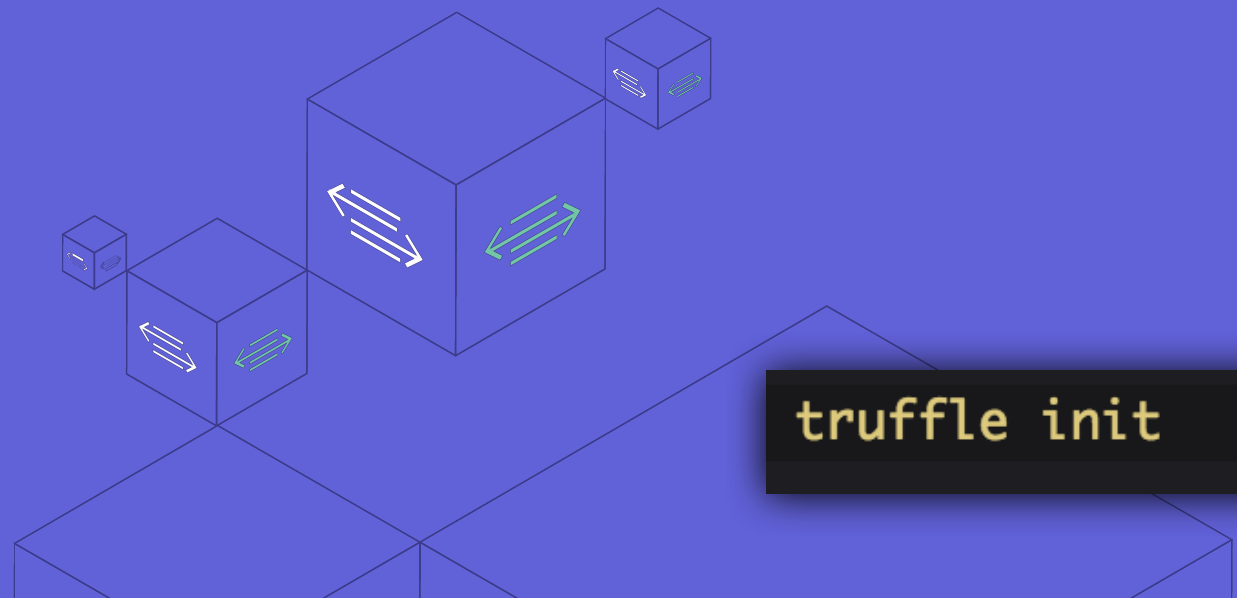
- Create a project with your favourite frontend technology
  - ReactJS
  - Angular
  - EmberJS
  - ...

trase



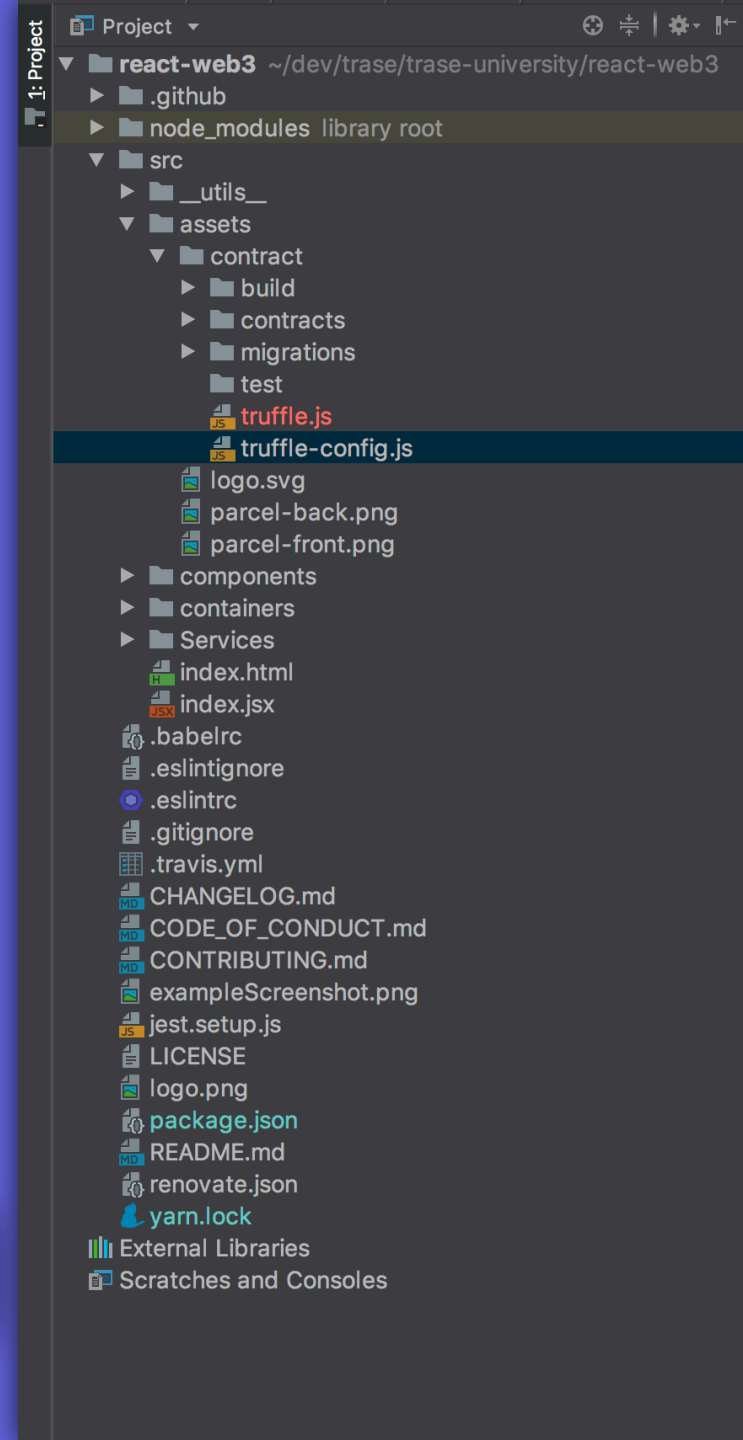
2. Initialise Truffle

- Inside the project initialise Truffle
- Assets folder w/ Smart Contracts
  - **Truffle init** inside assets folder



```
truffle init
```

trAse



3. Write Smart Contract

- Write the desired Smart Contract
- Here’s an example of our Kudo Coin SC

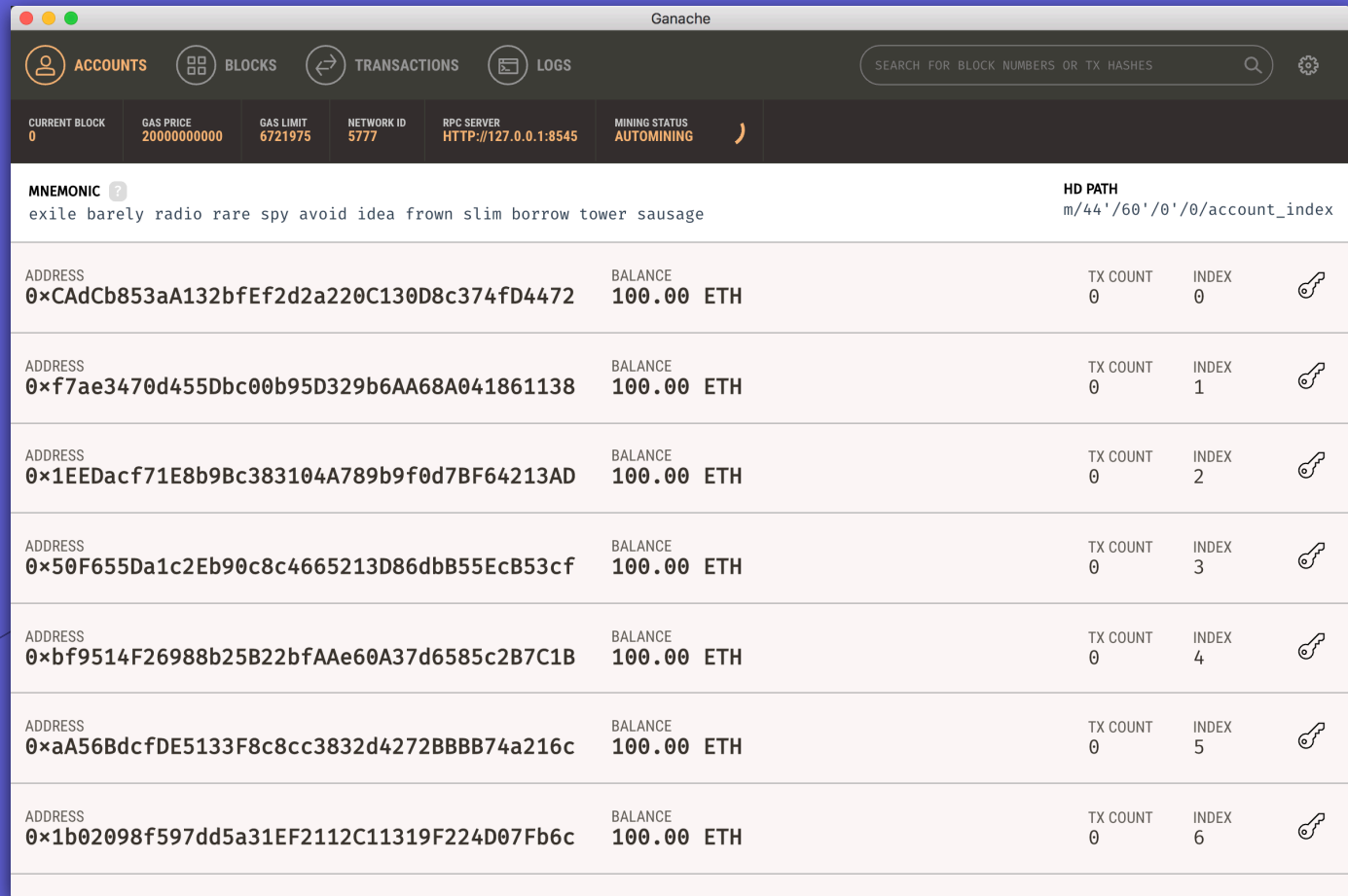
Coin.sol

```
pragma solidity ^0.4.24;

contract Coin {
    address adminAddress;

    mapping (address => uint) public assignableCoins;
    mapping (address => uint) public spendableCoins;

    event Transfer(
        address indexed from,
        address indexed to,
        uint256 amount,
        string reason,
        uint timestamp
    );

    constructor() public {
        adminAddress = msg.sender;
    }

    modifier onlyByCreator {
        require(msg.sender == adminAddress);
        _;
    }

    function addUser(address userAddress) public onlyByCreator {
        assignableCoins[userAddress] = 0;
        spendableCoins[userAddress] = 0;
    }

    function setAssignableCoins(address userAddress, uint amount) public onlyByCreator {
        assignableCoins[userAddress] = amount;
    }

    function addAssignableCoins(address userAddress, uint amount) public onlyByCreator {
        assignableCoins[userAddress] += amount;
    }

    function burnCoin(address addr, uint amount) public onlyByCreator{
        require((spendableCoins[addr] - amount) >= 0);
        spendableCoins[addr] -= amount;
    }

    function sendCoin(address receiver, uint amount, string reason) public {
        require(assignableCoins[msg.sender] >= amount && receiver != msg.sender && amount <= 3);
        spendableCoins[receiver] = spendableCoins[receiver] + (amount);
        assignableCoins[msg.sender] = assignableCoins[msg.sender] - amount;
        emit Transfer(msg.sender, receiver, amount, reason, block.timestamp);
```

trAse

4. Setup Ganache

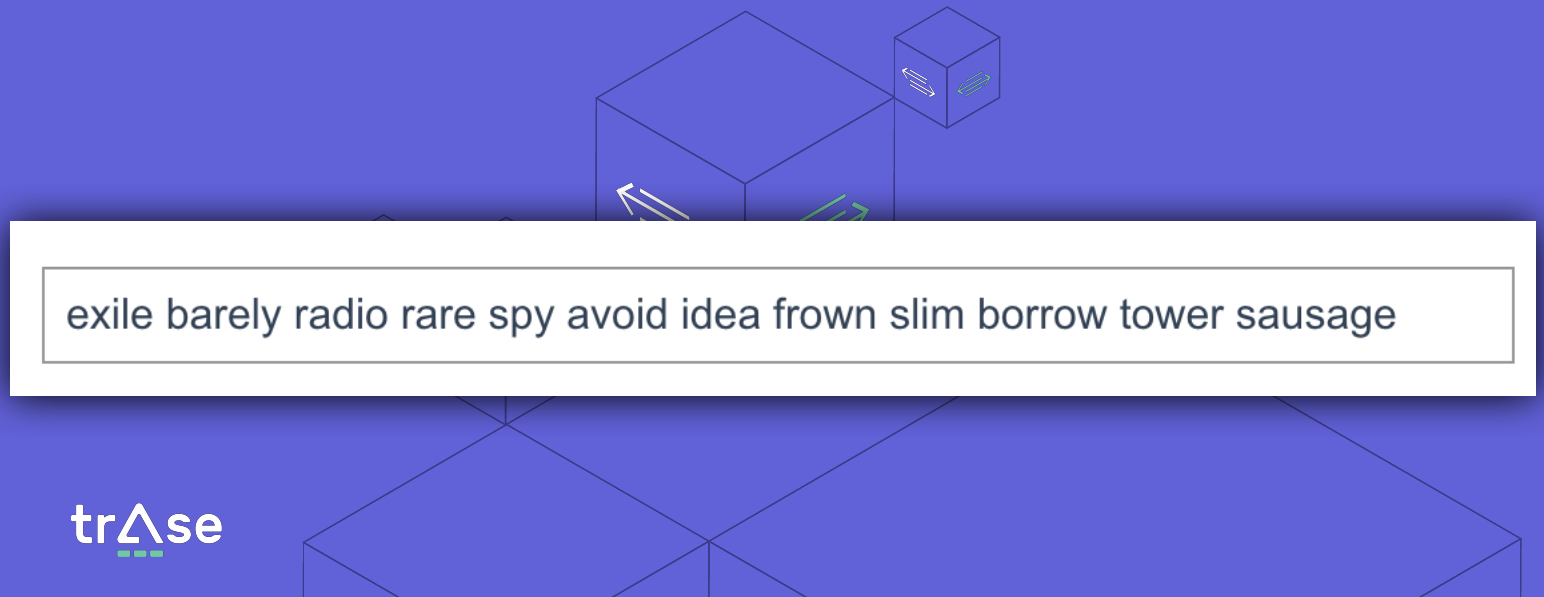
- Open ganache to bootstrap your own local Ethereum network!

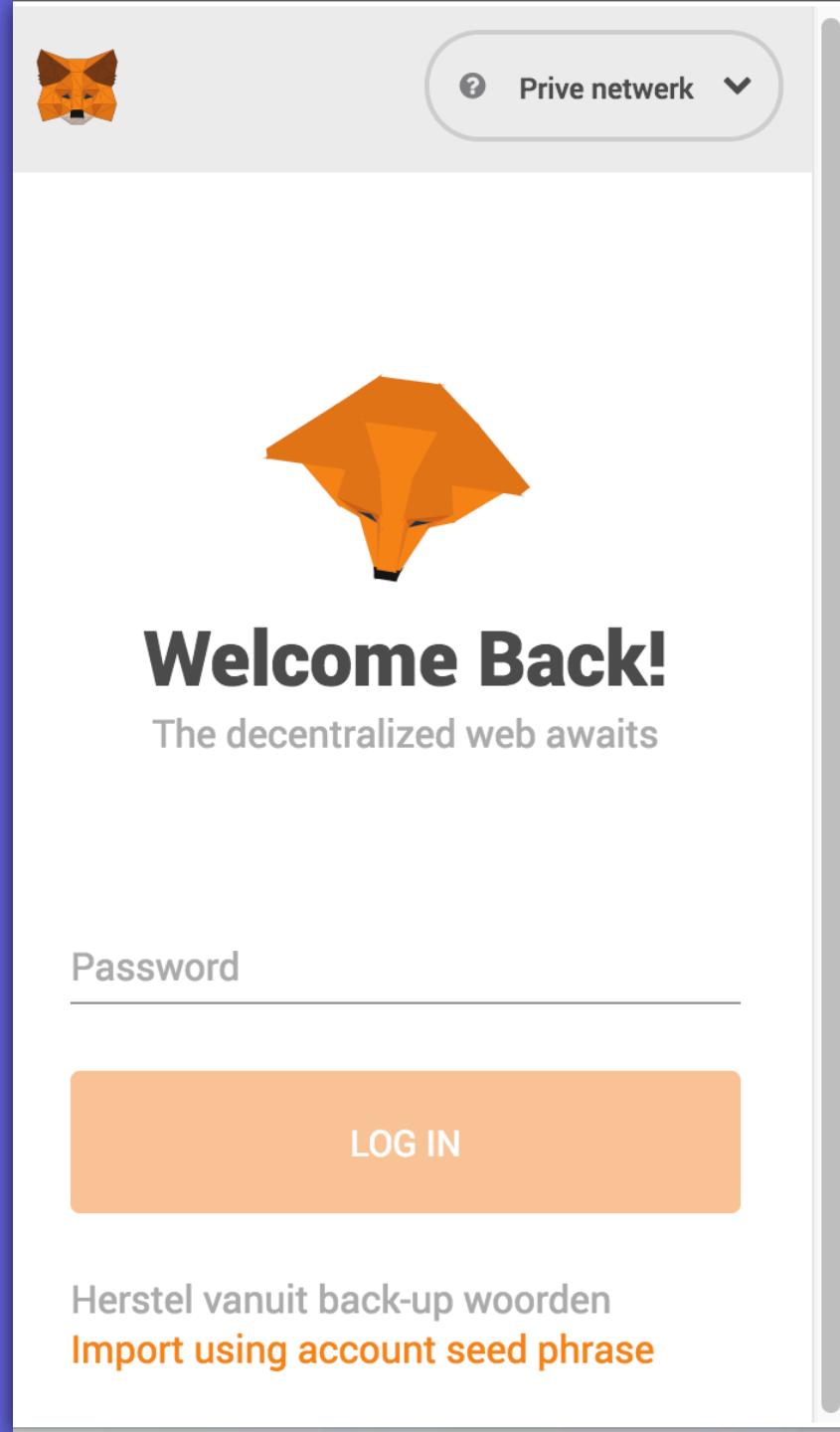


trAse

5. Import ganache accounts into Metamask

- Metamask needs to be opened with the seed (Mnemonic) of Ganache





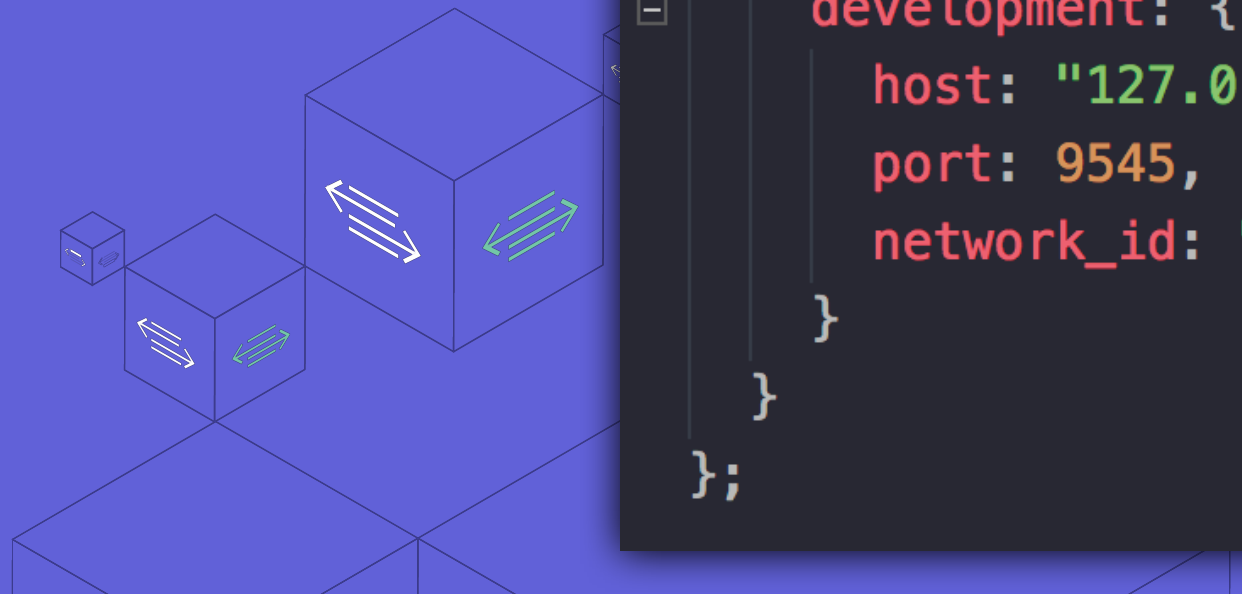
trΛse

6.1. Setup Truffle for Deployment

- Setup Truffle with local blockchain

JS truffle-config.js
JS truffle.js

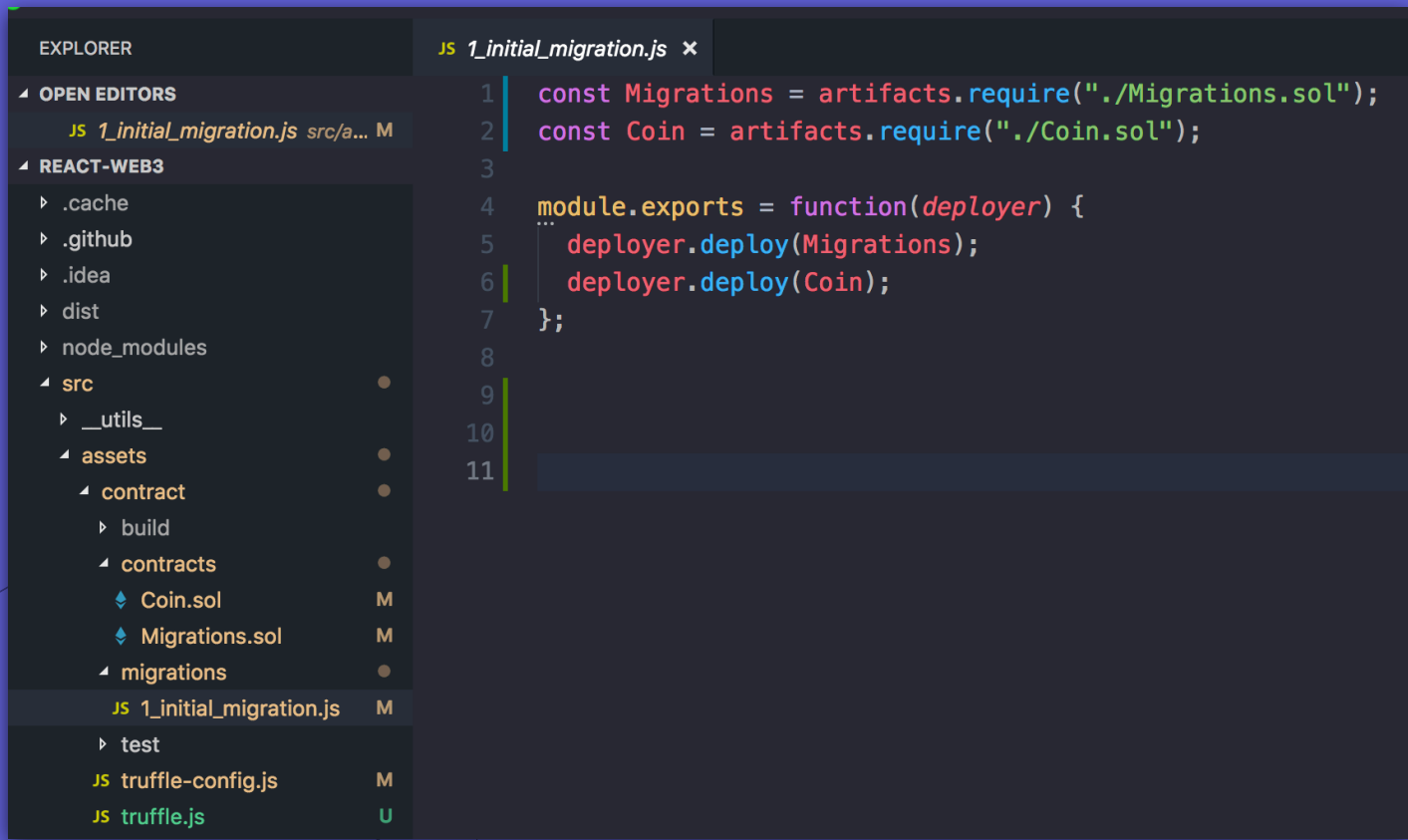
```
module.exports = {
  networks: {
    development: {
      host: "127.0.0.1",
      port: 9545,
      network_id: "*" // Match any network id
    }
  }
};
```



trAse

6.2. Setup Truffle for Deployment

- Compile and Deploy Smart contract for easy testing and interacting with the Smart Contract



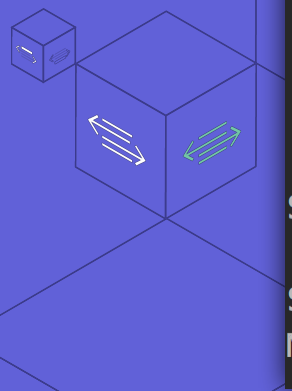
trAse

7. Deploy Smart Contract

- Truffle deploy

Useful flags:

- —reset
- —network=development



```
Maurices-MacBook-Pro:contract mauricedalderup$ truffle deploy --network=development --reset
Using network 'development'.

Running migration: 1_initial_migration.js
  Replacing Migrations...
  ... 0xef06d1e9aa8819d15540de0a9e3f36988c4be94cf9e9c03b92096c0360c4c550
  Migrations: 0x82d50ad3c1091866e258fd0f1a7cc9674609d254
  Replacing Coin...
  ... 0x86099144904c9a83e9ea06fc1f2ee42be01175f2b6c509fb2c2a7d5ad4feb244
  Coin: 0xdda6327139485221633a1fcd65f4ac932e60a2e1
Saving successful migration to network...
  ... 0xec2ef8aebf901496d7c061085cc50058e1e33730ab2d85b22b0e8f196e9da3a0
Saving artifacts...
Maurices-MacBook-Pro:contract mauricedalderup$
```

trAse

8. Frontend

- Create a service to interact with Web3

Web3Service.jsx

```
1  import Web3 from "web3";
2  import _ from "lodash";
3
4  export default class Web3Service {
5    constructor(contract = null, connectionUrl = null) {
6      this.web3 = new Web3(connectionUrl || Web3.givenProvider);
7
8      this.contract = {};
9      if (contract) this.contract = this.addContract(contract);
10   }
11
12   addContract = async contractJSON => {
13     this.contract = await new this.web3.eth.Contract(contractJSON.abi);
14     const key = _.findKey(contractJSON.networks, "address");
15     this.contract.options.address = contractJSON.networks[key].address;
16   };
17
18   getContract = () => this.contract;
19
20   createAccount = () => this.web3.eth.accounts.create();
21 }
22
```

trÂse

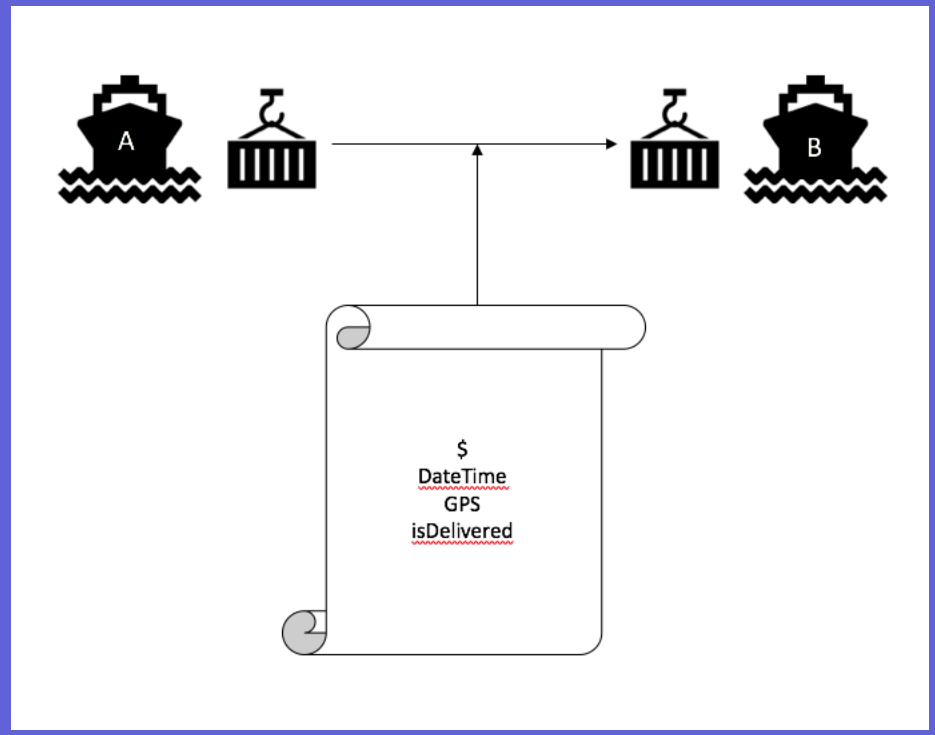
Questions?

trAse


trAse

Case

Ethereum



trase

Same case as in Solidity

- Extend it with Web3 funcitonality

trAse

Demo - Remix IDE

trAse

Trase Cases

trase