Building a dApp with Taquito

Jev Björsell & Simon Boissonneault-Robert
ECAD Labs Inc.

Taquito

A TypeScript Library for Tezos

Features (summary)

- Smart Contract Abstraction
- Easy for different Transactions
- Various ways to “Forging” operations
- Various options to sign
- Estimation of operations
- Injection and observing the chain
- RPC Client with good type coverage
- Committed to supporting all future Tezos features and protocols

Github: https://github.com/ecadlabs/taquito
Website: https://tezostaquito.io/

Package structure

Top level module is `@taquito/taquito`

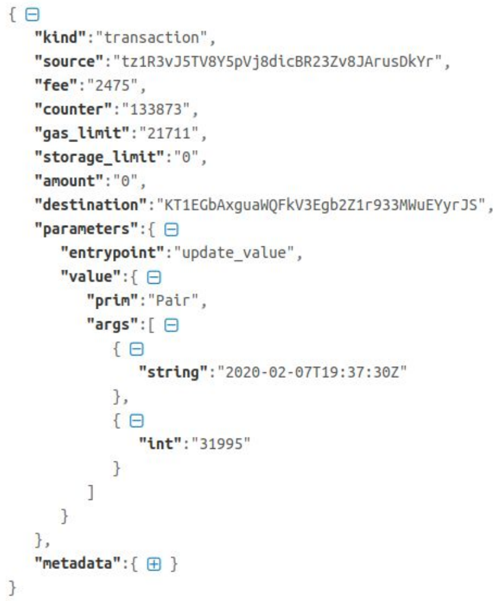
It is a composition of most of the lower level packages.

eg. signer (in-memory, tezbridge, remote-signer)

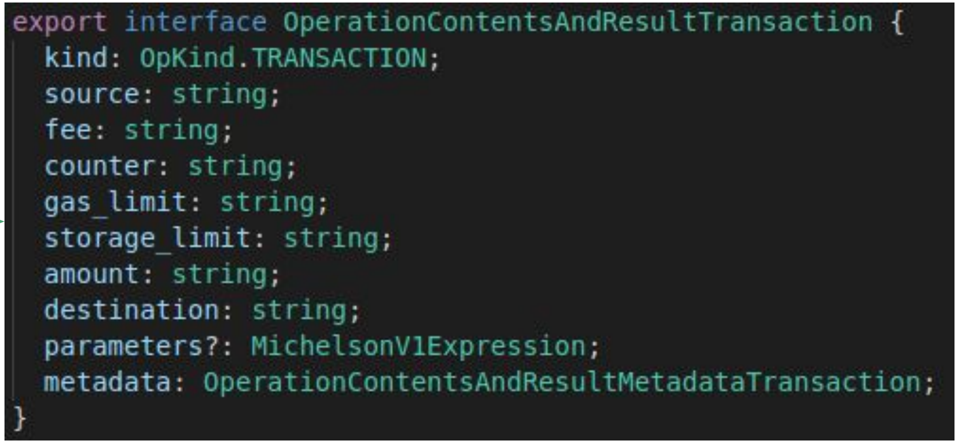
eg. forging (local, rpc, composite)

eg. streamer (rpc polling based)

Types



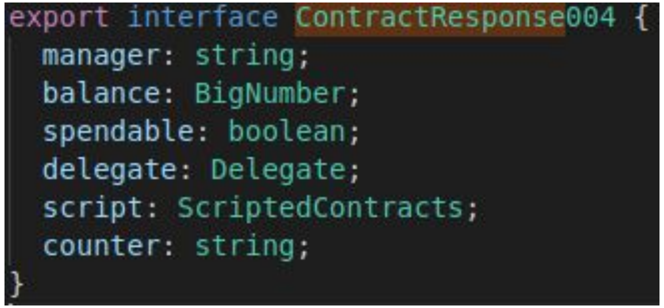
```
{
   "kind":"transaction",
   "source":"tz1R3vJ5TV8Y5pVj8dicBR23Zv8JArusDkYr",
   "fee":"2475",
   "counter":"133873",
   "gas_limit":"21711",
   "storage_limit":"0",
   "amount":"0",
   "destination":"KT1EGbAxguaWQFkV3Egb2Z1r933MWuEYyrJS",
   "parameters":{
      "entrypoint":"update_value",
      "value":{
         "prim":"Pair",
         "args":[
            {
               "string":"2020-02-07T19:37:30Z"
            },
            {
               "int":"31995"
            }
         ]
      }
   },
   "metadata":{ }
}
```



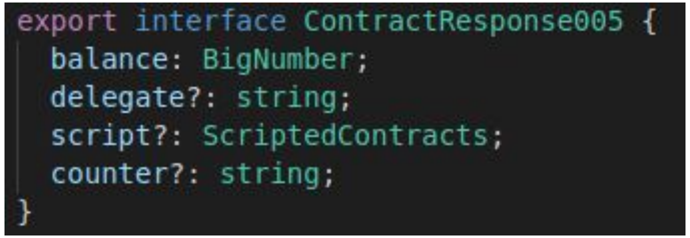
```
export interface OperationContentsAndResultTransaction {
  kind: OpKind.TRANSACTION;
  source: string;
  fee: string;
  counter: string;
  gas_limit: string;
  storage_limit: string;
  amount: string;
  destination: string;
  parameters?: MichelsonV1Expression;
  metadata: OperationContentsAndResultMetadataTransaction;
}
```

RPC Client

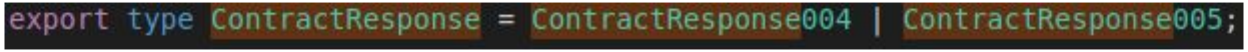
- A typescript module offering convenient methods for accessing the Tezos Nodes RPC
- Methods map one to one with RPC endpoints
- Provide complete typing coverage over RPC response
- Type coverage over current Tezos protocol AND the next protocol



```
export interface ContractResponse004 {
  manager: string;
  balance: BigNumber;
  spendable: boolean;
  delegate: Delegate;
  script: ScriptedContracts;
  counter: string;
}
```



```
export interface ContractResponse005 {
  balance: BigNumber;
  delegate?: string;
  script?: ScriptedContracts;
  counter?: string;
}
```



```
export type ContractResponse = ContractResponse004 | ContractResponse005;
```

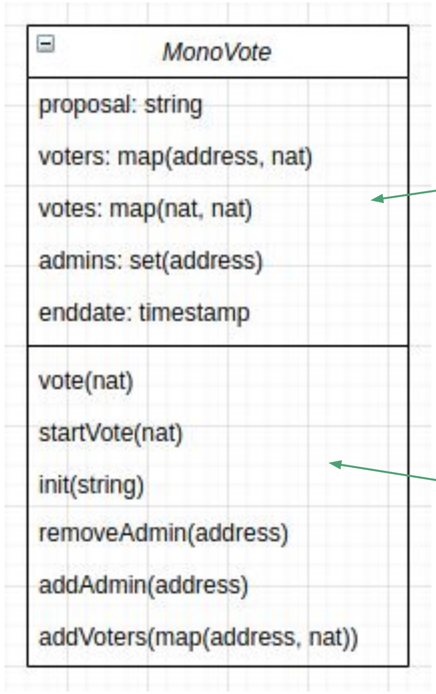
Demo app

https://github.com/ecadlabs/proposal-vote

https://bit.ly/37j2ksL

https://ide.ligolang.org/p/5W3GfXD1AKR2u8eAj8h2Cw

Contract used for the demo



Storage

Entrypoints
(methods)

Installing Taquito

```
-  npm install @taquito/taquito
```

Other package:

```
-  npm install @taquito/signer
-  npm install @taquito/tezbridge-signer
-  npm install @taquito/remote-signer
-  npm install @taquito/local-forging
```

Using a script tag:

```
<script src="https://unpkg.com/@taquito/taquito@6.0.2-beta.0/dist/taquito.min.js"
crossorigin="anonymous"
integrity="sha384-gIjWpwSahQXCejt3IXr83Lxmcfe13gZX97Yp7bpdCMpX/fD0XV3V4hxRHhCVX9+k"></scrip
t>
```

TezosToolkit instance

Singleton

```
Import { Tezos } from ‘@taquito/taquito’
```

Class

```
Import { TezosToolkit } from ‘@taquito/taquito’

const tezos = new TezosToolkit()
```

Injecting providers

```
Tezos.setProvider({ signer, forger, config, stream, protocol, rpc })
```

Defaults:

- No signer
- RPC node
- RPC forger
- RPC polling streamer
- Default configuration
- Protocol is fetch with RPC call

Injecting Provider

```
Import { InMemorySigner } from “@taquito/signer”

Tezos.setProvider({signer: new InMemorySigner(“edsk...”)})
```

```
Import { TezbridgeSigner } from “@taquito/tezbridge-signer”

Tezos.setProvider({signer: new TezbridgeSigner()})
```

You also need to add the tezbridge plugin to you page

```
<script src="https://www.tezbridge.com/plugin.js"></script>
```

Interacting with a contract

- Taquito smart contract abstraction
- The transaction operation
- Show demo code sample

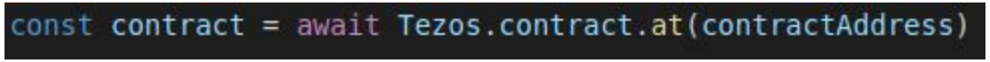
Interacting with a contract

What is Taquito smart contract abstraction?

- Friendly storage representation
- Friendly way of accessing big map
- Expose entry points with methods
- Smart contract error

Interacting with a contract

Fetching the contract abstraction with Taquito



```
const contract = await Tezos.contract.at(contractAddress)
```

Under the hood Taquito does:

- Fetch the contract Michelson code from the chain
- Parse it
- Create a dev friendly abstraction of the contract

Interacting with a contract

Storage

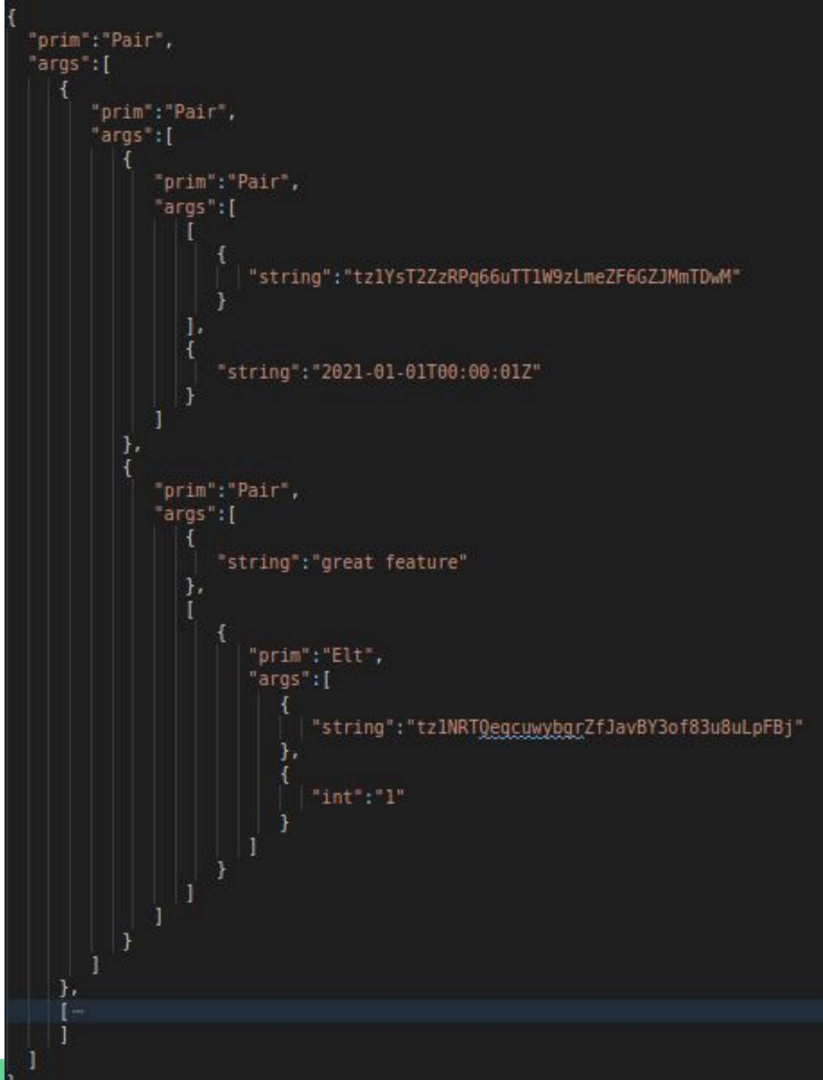
```
const contract = await Tezos.contract.at(contractAddress)
const storage = await contract.storage()
```

```
{
    "admins":[
        "tz1YsT2ZzRPq66uTT1W9zLmeZF6GZJMmTDwM"
    ],
    "enddate":"2021-01-01T00:00:01.000Z",
    "proposal":"great feature",
    "voters":{
        "tz1NRTQeqcuwybgrZfJavBY3of83u8uLpFBj":"1"
    },
    "votes":{
        "1":"3",
        "2":"1",
        "3":"1"
    }
}
```

```
type storage is record
  proposal: hash;
  voters: voters;
  votes: map(nat, nat);
  enddate: timestamp;
  admins: set(address);
end
```

Interacting with a contract

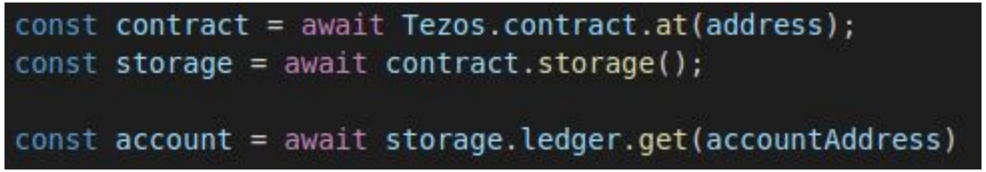
Equivalent michelson representation



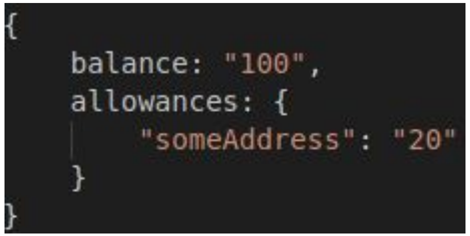
```
{
  "prim":"Pair",
  "args":[
    {
      "prim":"Pair",
      "args":[
        {
          "prim":"Pair",
          "args":[
            [
              {
                "string":"tz1YsT2ZzRPq66uTT1W9zLmeZF6GZJMmTDwM"
              }
            ],
            {
              "string":"2021-01-01T00:00:01Z"
            }
          ]
        },
        {
          "prim":"Pair",
          "args":[
            {
              "string":"great feature"
            },
            [
              {
                "prim":"Elt",
                "args":[
                  {
                    "string":"tz1NRTQeqcuwybqrZfJavBY3of83u8uLpFBj"
                  },
                  {
                    "int":"1"
                  }
                ]
              }
            ]
          ]
        }
      ]
    },
    [⋯
    ]
  ]
```

Interacting with a contract

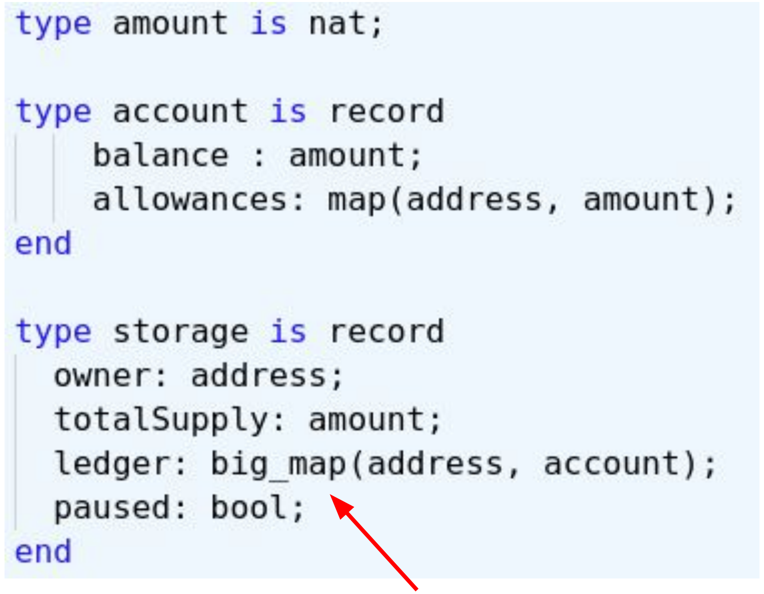
Big map



```
const contract = await Tezos.contract.at(address);
const storage = await contract.storage();

const account = await storage.ledger.get(accountAddress)
```



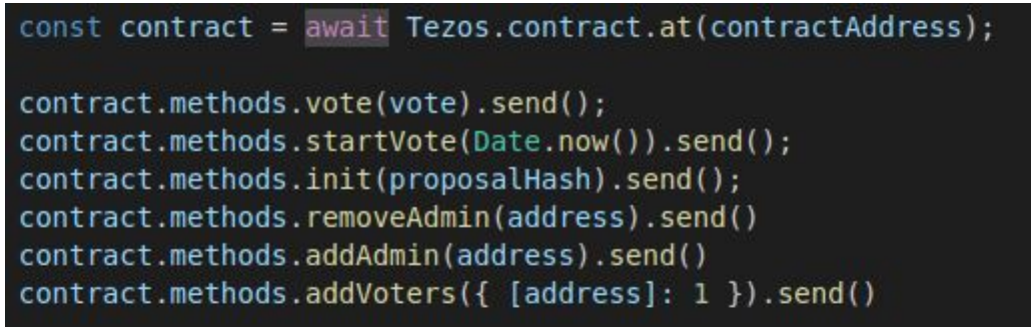
```
{
    balance: "100",
    allowances: {
        "someAddress": "20"
    }
}
```



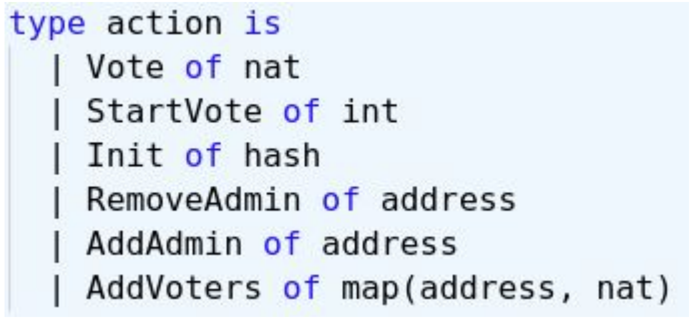
```
type amount is nat;

type account is record
    balance : amount;
    allowances: map(address, amount);
end

type storage is record
  owner: address;
  totalSupply: amount;
  ledger: big_map(address, account);
  paused: bool;
end
```

Interacting with a contract

Methods



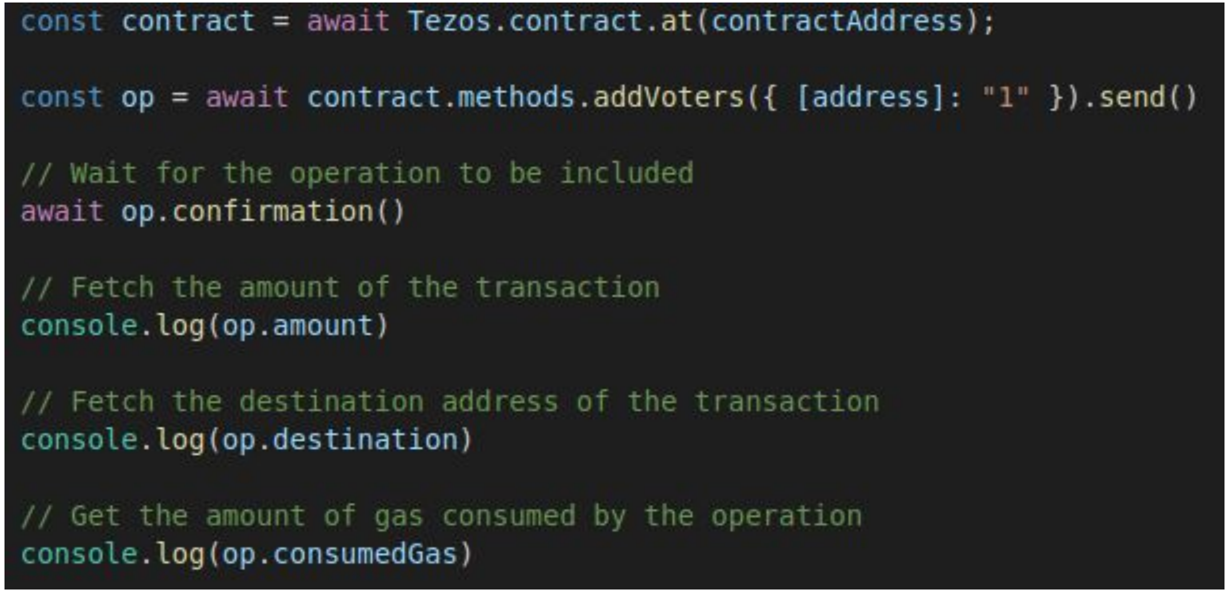
```
const contract = await Tezos.contract.at(contractAddress);

contract.methods.vote(vote).send();
contract.methods.startVote(Date.now()).send();
contract.methods.init(proposalHash).send();
contract.methods.removeAdmin(address).send()
contract.methods.addAdmin(address).send()
contract.methods.addVoters({ [address]: 1 }).send()
```



```
type action is
  | Vote of nat
  | StartVote of int
  | Init of hash
  | RemoveAdmin of address
  | AddAdmin of address
  | AddVoters of map(address, nat)
```

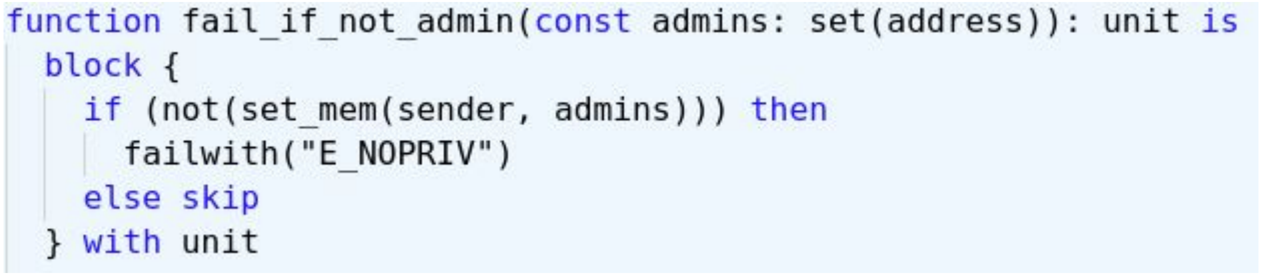
Interacting with a contract

The transaction operation

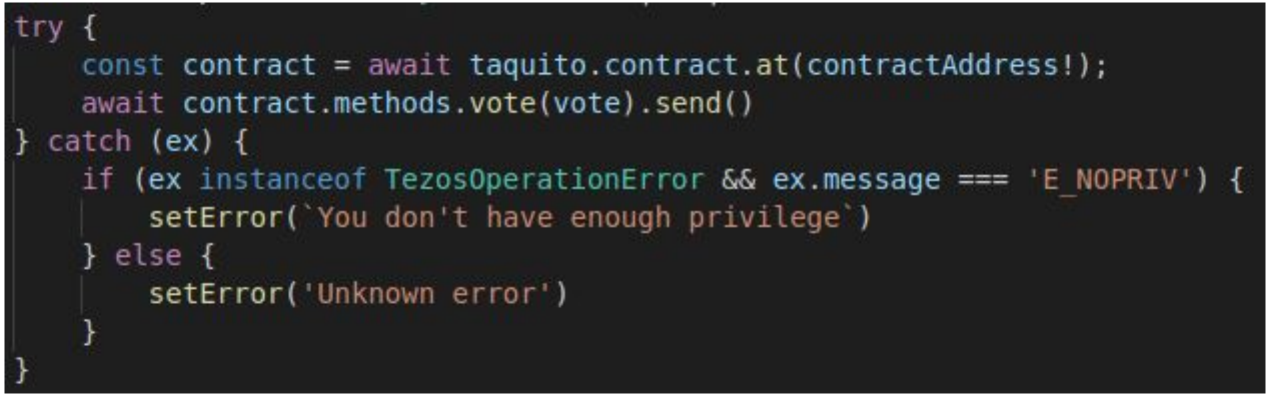

```
const contract = await Tezos.contract.at(contractAddress);

const op = await contract.methods.addVoters({ [address]: "1" }).send()

// Wait for the operation to be included
await op.confirmation()

// Fetch the amount of the transaction
console.log(op.amount)

// Fetch the destination address of the transaction
console.log(op.destination)

// Get the amount of gas consumed by the operation
console.log(op.consumedGas)
```

Interacting with a contract

Script error



```
function fail_if_not_admin(const admins: set(address)): unit is
  block {
    if (not(set_mem(sender, admins))) then
      failwith("E_NOPRIV")
    else skip
  } with unit
```



```
try {
    const contract = await taquito.contract.at(contractAddress!);
    await contract.methods.vote(vote).send()
} catch (ex) {
    if (ex instanceof TezosOperationError && ex.message === 'E_NOPRIV') {
        setError(`You don't have enough privilege`)
    } else {
        setError('Unknown error')
    }
}
```

Questions?
Feedback?

Bonus Material!

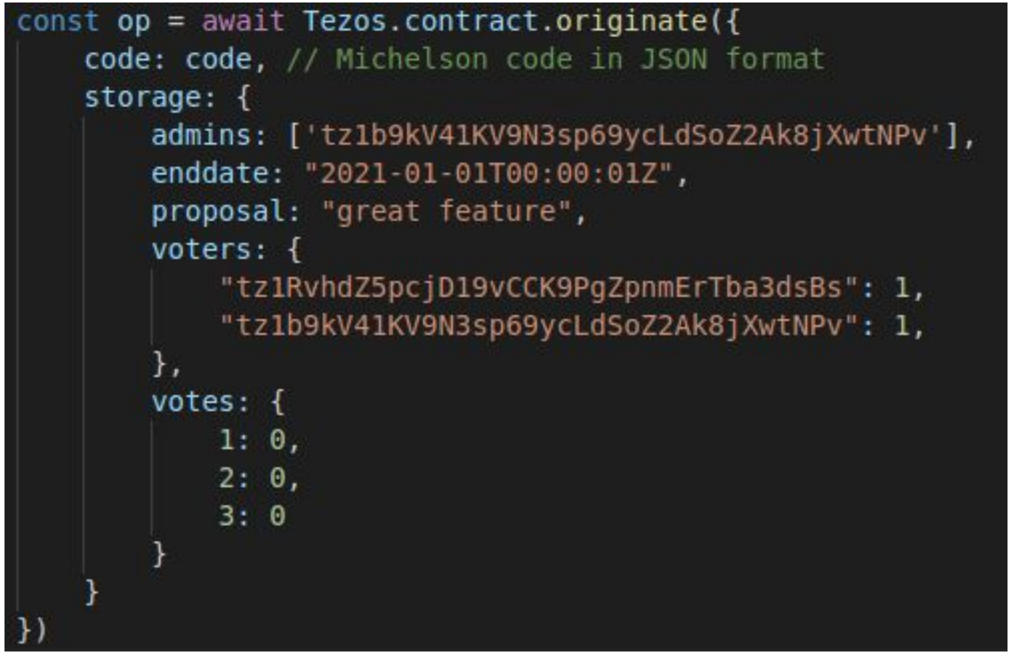
Time permitting

Originate a new Contract using Taquito

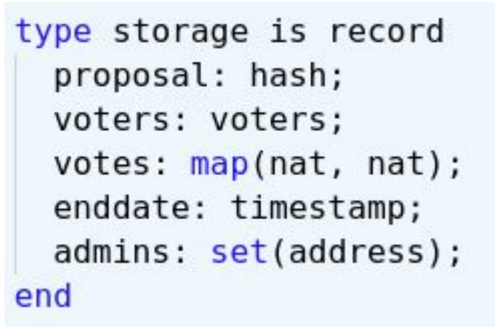
- Storage initialization
- Origination operation
- From ligo to json (using ligo)
- From michelson to json (using tezos client)
- Show demo code sample

Originate a new Contract using Taquito

Using storage property to initialize storage



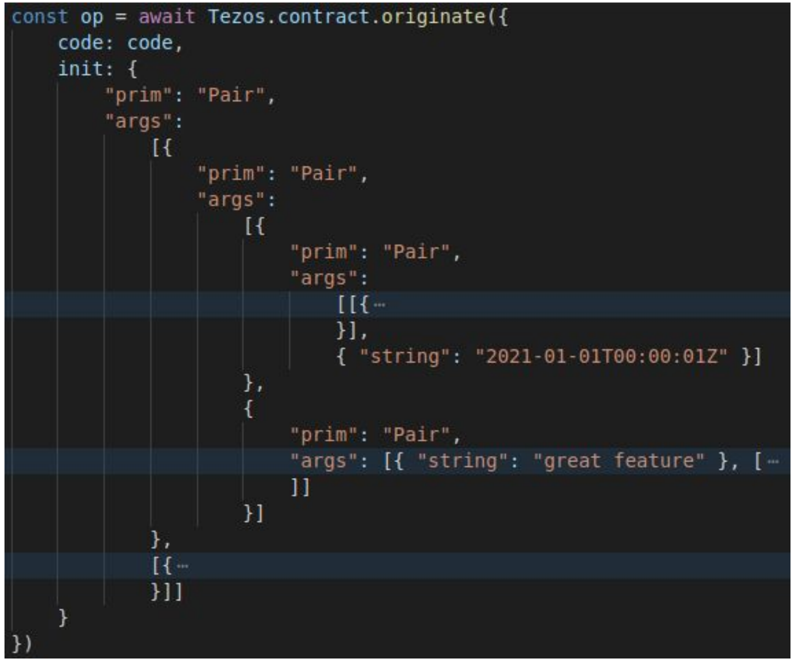
```
const op = await Tezos.contract.originate({
    code: code, // Michelson code in JSON format
    storage: {
        admins: ['tz1b9kV41KV9N3sp69ycLdSoZ2Ak8jXwtNPv'],
        enddate: "2021-01-01T00:00:01Z",
        proposal: "great feature",
        voters: {
            "tz1RvhdZ5pcjD19vCCK9PgZpnmErTba3dsBs": 1,
            "tz1b9kV41KV9N3sp69ycLdSoZ2Ak8jXwtNPv": 1,
        },
        votes: {
            1: 0,
            2: 0,
            3: 0
        }
    }
})
```



```
type storage is record
  proposal: hash;
  voters: voters;
  votes: map(nat, nat);
  enddate: timestamp;
  admins: set(address);
end
```

Originate a new Contract using Taquito

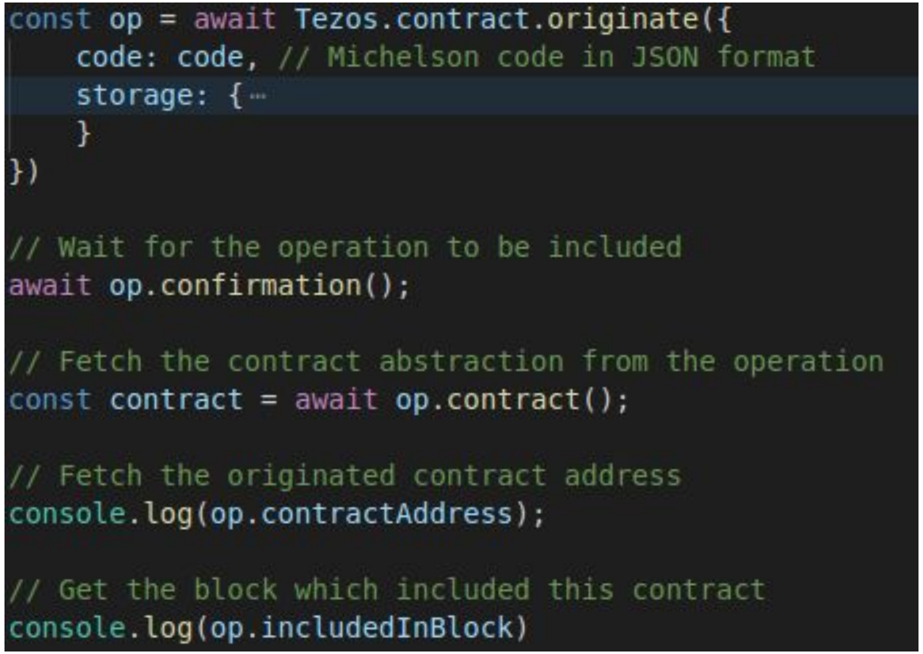
Using init to initialize storage



```
const op = await Tezos.contract.originate({
    code: code,
    init: {
        "prim": "Pair",
        "args":
            [{
                "prim": "Pair",
                "args":
                    [{
                        "prim": "Pair",
                        "args":
                            [[{⋯
                            }],
                            { "string": "2021-01-01T00:00:01Z" }]
                    },
                    {
                        "prim": "Pair",
                        "args": [{ "string": "great feature" }, [⋯
                        ]]
                    }]
            },
            [{⋯
            }]]
    }
})
```

Originate a new Contract using Taquito

The origination operation



```
const op = await Tezos.contract.originate({
    code: code, // Michelson code in JSON format
    storage: {⋯
    }
})

// Wait for the operation to be included
await op.confirmation();

// Fetch the contract abstraction from the operation
const contract = await op.contract();

// Fetch the originated contract address
console.log(op.contractAddress);

// Get the block which included this contract
console.log(op.includedInBlock)
```

How to go from Ligo to JSON format?

From ligo to JSON format:

```
ligo compile-contract --michelson-format=json mono_vote.ligo main
```

How to go from Michelson to JSON format?

From michelson to JSON format:

```
tezos-client originate contract mono-vote \
      transferring 0 from account1 running \
      "$(cat ./mono_vote.tz)" --init "$(cat ./mono_vote.init.tz)" \
      --dry-run --verbose-signing --burn-cap 3
```