Solidity und Testing mit Ethereum

Hauptseminar - James Friesen

Solidity

- objektorientiert
- basiert auf JavaScript
- statische Typisierung
- kann in Bytecode kompiliert und von der EVM ausgeführt werden

Basic Types

- string
- bool
- int (int8, int16, ..., int256; int = int256)
- uint (Unsigned, nur positiv)
- fixed/ufixed (Nummer mit Dezimalzahlen)
- address (mit Methoden um Währung zu verschicken, hex 20 bytes; 0x72bA7d8E73Fe8Eb666Ea66babC8116a41bFb10e2)

Reference Types

- fixed array
- dynamic array (mit nesting, jedoch noch kommunikationsprobleme zu web3)
- mapping
- struct

Common Function Types

- public
- private (nur im Contract)
- view, constant (nur lesen)
- pure (kein lesen oder schreiben im Contract)
- payable (wenn ausgeführt kann Ether versendet werden)

```
pragma solidity ^0.4.17;

contract LockContract {
    string public message;

    function LockContract(string initialMessage) public {
        message = initialMessage;
    }

    function setMessage(string newMessage) public {
        message = newMessage;
    }

    function setBooked() public {
        message = "Booked!";
    }

    function setFree() public {
        message = "Free!";
    }
}
```

Lokales Testing

- Ganache:
  - schnelle persönliche Ethereum Blockchain
  - testing
  - state inspection
  - advanced mining controlls

```
const assert = require('assert');
const ganache = require('ganache-cli');
const Web3 = require('web3');

const provider = ganache.provider();
const web3 = new Web3(provider);

const { interface, bytecode } = require('../compile');

let accounts;
let lockContract;

beforeEach(async () => {
    // Get a list of all accounts
    accounts = await web3.eth.getAccounts();

    //Use one of those accounts to deploy the contract
    lockContract = await new web3.eth.Contract(JSON.parse(interface))
        .deploy({ data: bytecode, arguments: ['Room uninitialized']})
        .send({ from: accounts[0], gas: '1000000' })

    lockContract.setProvider(provider);
});

describe('lockContract', () => {
    it('deploys a contract', () => {
        assert.ok(lockContract.options.address);
    });

    it('has default message Room uninitialized', async () => {
        const message = await lockContract.methods.message().call();
        assert.equal(message, 'Room uninitialized')
    });

    it('can change the message', async () => {
        await lockContract.methods.setMessage('Room out of order').send({ from: accounts[0]});
        const message = await lockContract.methods.message().call();
        assert.equal(message, 'Room out of order')
    });
```

```
james@james-Lenovo-G780:~/Documents/Workspace/BookNBlock/solidity/lockcontract$ npm test

> lockcontract@1.0.0 test /home/james/Documents/Workspace/BookNBlock/solidity/lockcontract
> mocha



  lockContract
(node:9473) MaxListenersExceededWarning: Possible EventEmitter memory leak detected. 11 data
 listeners added. Use emitter.setMaxListeners() to increase limit
    ✓ deploys a contract
    ✓ has default message Room uninitialized (57ms)
    ✓ can change the message (170ms)
    ✓ can be booked (104ms)
    ✓ can be set free (62ms)


  5 passing (1s)

james@james-Lenovo-G780:~/Documents/Workspace/BookNBlock/solidity/lockcontract$
```

Distributed Testing

- Infura:
  - API Schnittstelle zu Ethereum Blockchains
  - Ersetzt einen lokalen Knoten
- Rinkeby Ethereum Testnet

  https://www.rinkeby.io/

Remix IDE

- Browserbasierte IDE für Solidity
- Erlaubt auch das Ausführen von Smart Contracts
- Übernimmt anfangs die Rolle des Web-Frontends

https://remix.ethereum.org/