

Outline



- 1. Introduction
 - Motivation
 - Definition
 - Benefits
 - Drawbacks
- 2. Architecture
 - Web-based systems
 - Private key management
- 3. Libraries and Frameworks
 - Development tools
 - Local
 - Cloud
 - Test networks
 - Web3

Motivation



- Direct interaction with smart contracts is complicated
- Smart contracts don't provide a graphical user interface on their own
- Programming knowledge or special tools are required to make function calls

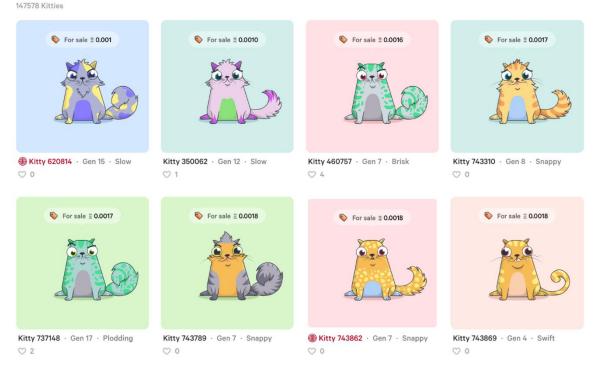


Screenshot of popular Ethereum wallet: MyEtherWallet.com

Motivation



- Building Uls on top of smart contracts to make them accessible to average users.
- The UI abstracts the complicated function calls and allows a user to interact with them
 just like with a regular (web) application.



Screenshot of popular CryptoKitties.co dApp

Definition



- In the community, multiple definitions for dApps exist.
- In general, a dApp is not necessarily based on a Blockchain, e.g., BitTorrent is a decentralized P2P application without any Blockchain involved.
- In this lecture, we consider a dApp as a decentralized application (in the terms of Blockchain) based on one or more smart contracts and accessible via a dedicated user interface.
- In particular, the following properties must hold:
 - The core data records of the application must be stored on the blockchain
 - The functions that change the core data records must be executed on the Blockchain, i.e. via a smart contract

Benefits



The meaningfulness of implementing a distributed application is dependent on the concrete use case and / or the problem that is being solved.

Some general properties of Ethereum-based dApps:

Trust

The source code of any verified smart contract can be checked by anyone.

Payment

Payment is implemented by default since anyone can send / receive Ether.

Accounts

dApps can be build on top of Ethereum's account system, so there is no need to implement an additional user account management system.

Storage

dApps can leverage the Blockchain as common (expensive) data storage.

Drawbacks



Decentralized applications have also some intrinsic disadvantages:

Costs

Any state change and computation costs money. For that reason, only mission-critical data and functionality should leverage the Blockchain.

Time

The current block time of Ethereum is around 14 seconds, i.e., it takes at least 14 seconds from the function call to the definite result of it.

Availability

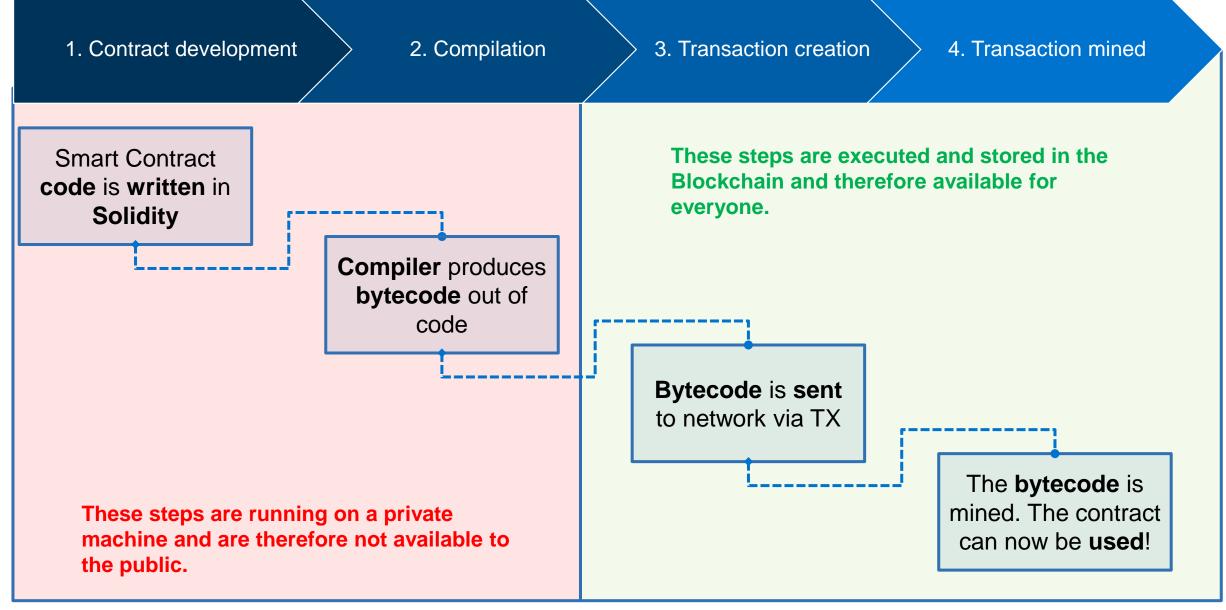
In theory, availability is one of the key advantages of dApps. However, in high transaction scenarios (e.g. the release of crypto kitties) it is possible that the network throttles and is not able to process function calls anymore.

Transparency

Without third party services, it is impossible to access and verify a Smart Contract source code.

Recap: How is a Smart Contract deployed on the Ethereum Blockchain?





Source code is typically not stored on the Blockchain, only byte code.



Switch To Opcodes View

Find Similiar Contracts

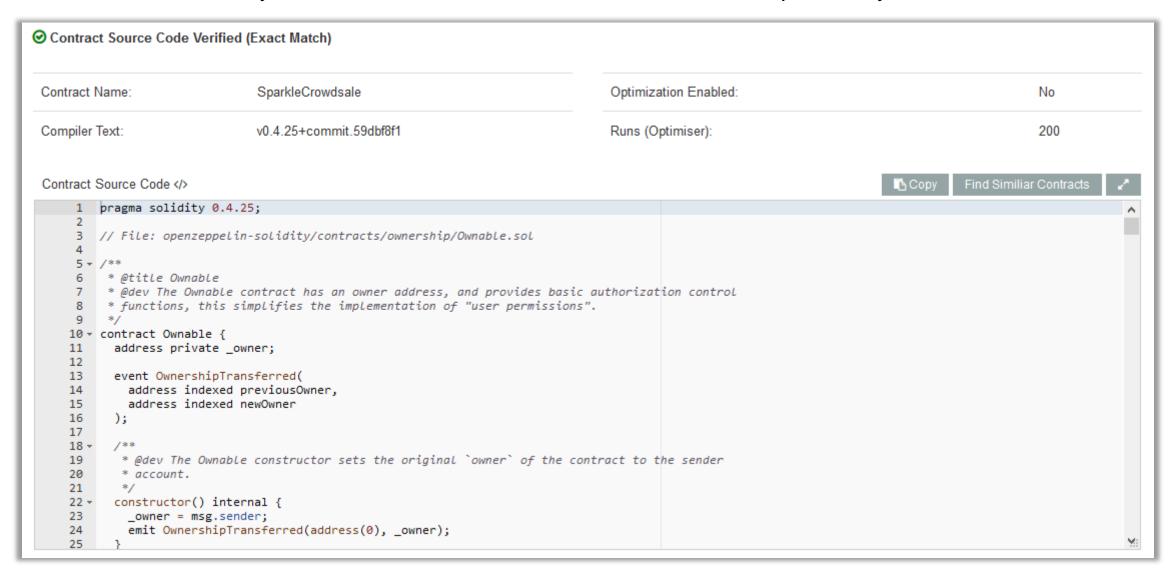
2dd146101be578063313ce5671461024357806370a082311461027457806395d89b41146102cb578063a9059cbb1461035b578063dd62ed3e146103c0575b600080fd5b3480156100aa57600080fd5b506100b3610437565b60405180 80602001828103825283818151815260200191508051906020019080838360005b838110156100f35780820151818401526020810190506100d8565b505050905090810190601f16801561012057808203805160018360200361010905050506104d5565b604051808215151515815260200191505060405180910390f35b34801561019f57600080fd5b506101a86105c7565b6040518082815260200191505060405180910390f35b3480156101ca57600080fd5b50610518080602001828103825283818151815260200191508051906020019080838360005b83811015610320578082015181840152602081019050610305565b50505090810190601f16801561034d5780820380516001836020036040260200160405190810160405280929190818152602001828054600181600116156101000203166002900480156104cd5780601f106104a2576101008083540402835291602001916104cd565b820191906000526020600020905b8ffffffffffffffffff67f8c5be1e5ebec7d5bd14f71427d1e84f3dd0314c0f7b2291e5b200ac8c7c3b925846040518082815260200191505060405180910390a3600190509291505056

Without further analysis, the purpose of this Smart Contract is unclear.

Source code can be made publicly available.



Etherscan.io is the only service which verifies source codes and the respective byte code.



Outline



- 1. Introduction
 - Motivation
 - Definition
 - Benefits
 - Drawbacks

2. Architecture

- Web-based systems
- Private key management

3. Libraries and Frameworks

- Development tools
 - Local
 - Cloud
- Test networks
- Web3

Web-based dApps



Currently available decentralized applications are usually web-based and run in the browser.

State of the ĐApps is a curated and community-driven directory of Ethereum-based decentralized applications. The directory is one of the largest and most relevant of its kind and in some talks referenced by Vitalik Buterin himself ¹.

As of April 2020, the directory contains 2825 (2019: 2,667) dApps.

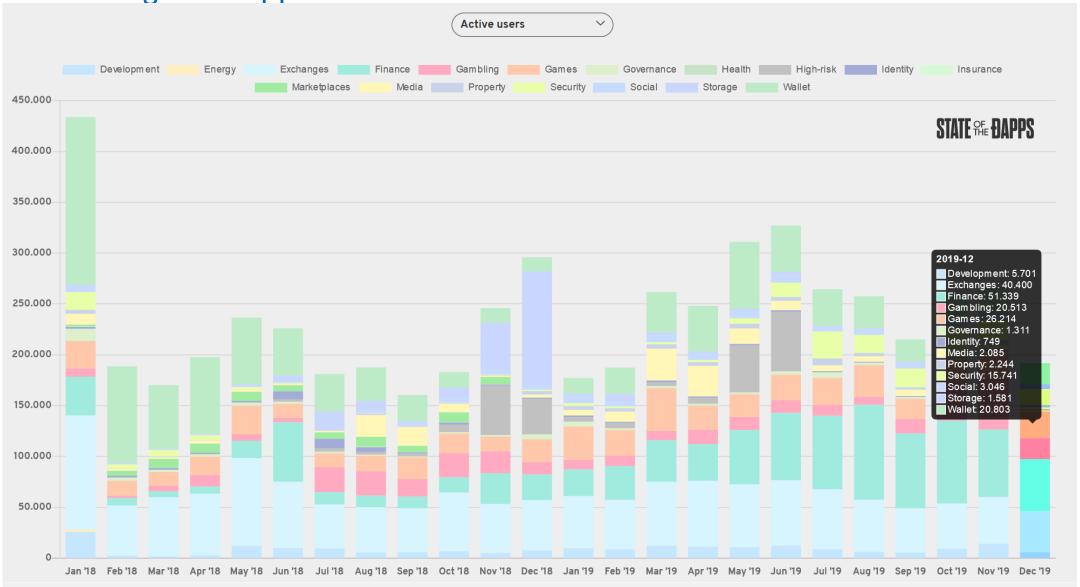
Examples of current dApps

- ICOs Token systems that can be used to sell securities for Ether
- Games Usually collectibles are issued and exchanged for Ether
- Gambling Betting and lottery application
- Exchanges Decentralized token and ether exchanges

https://www.stateofthedapps.com/

Current usage of dApps

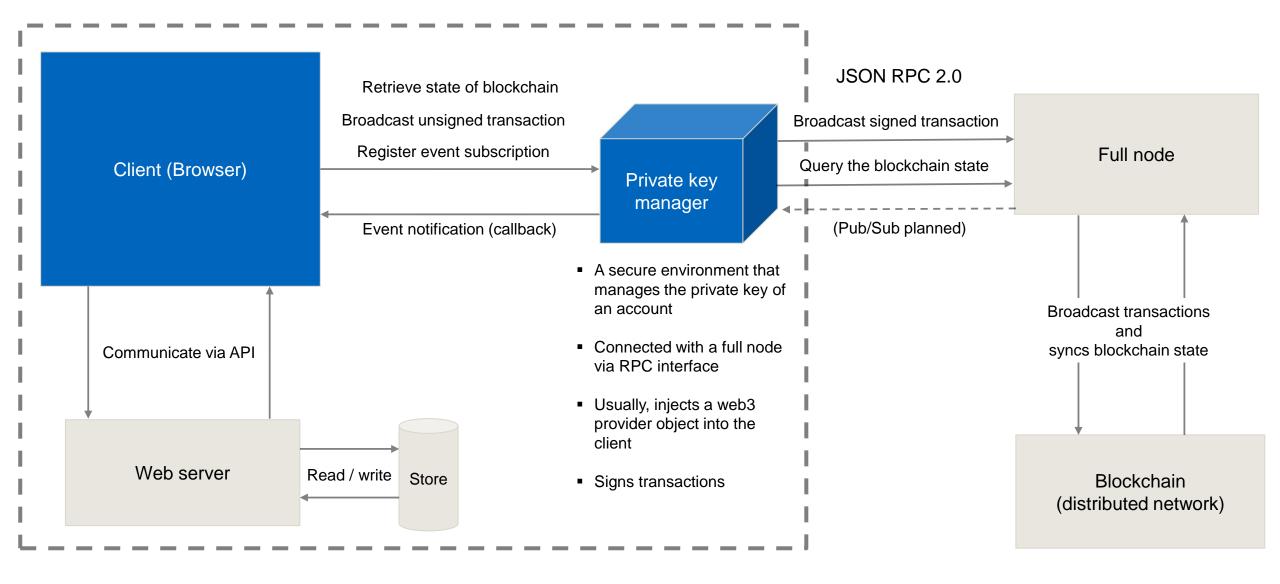




https://www.stateofthedapps.com/de/stats/platform/ethereum#new

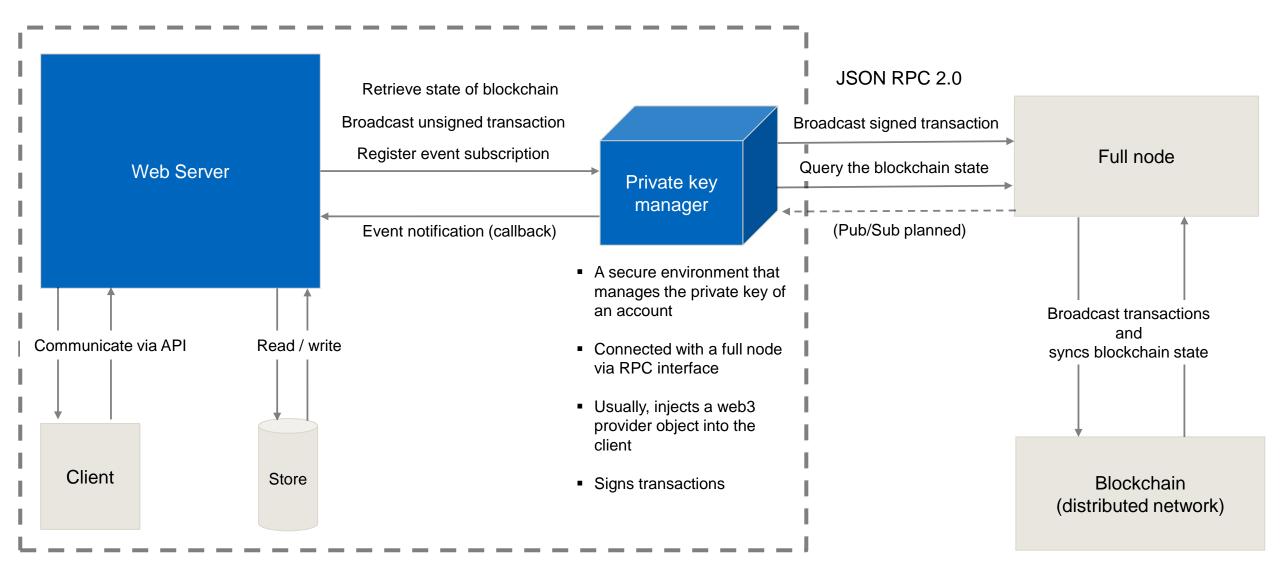
Architecture of a web-based Ethereum dApp





Architecture of a web-based Ethereum dApp (cont.)





MetaMask – A popular private key manager

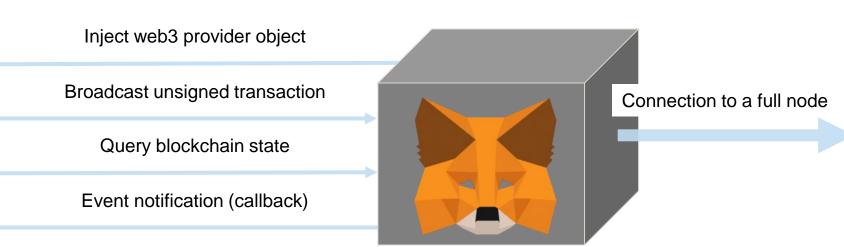
Browser Plugin



Browser

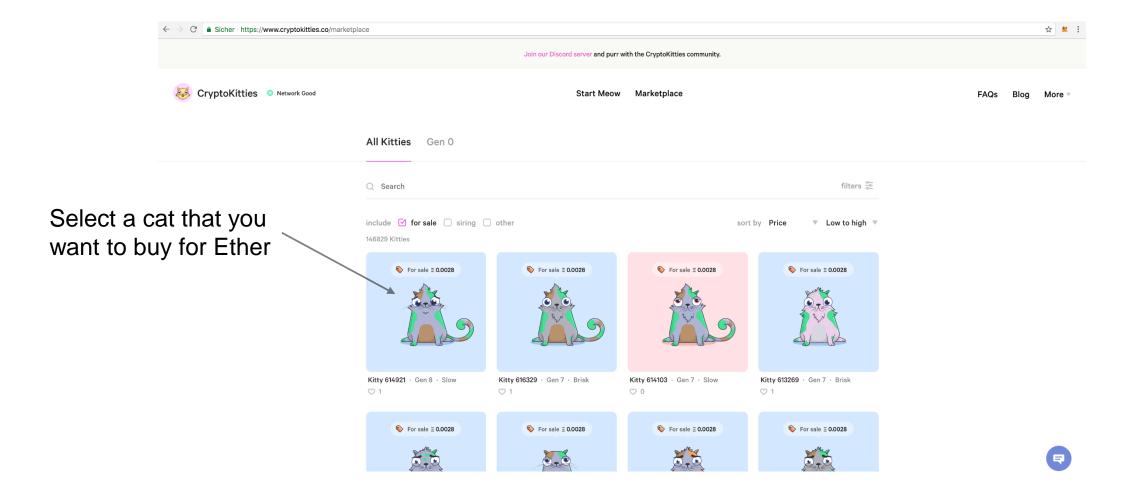


- A browser with MetaMask () plugin installed
- The plugin is required to manage the private keys of an Ethereum account
- It signs transactions and establishes a connection to a full node



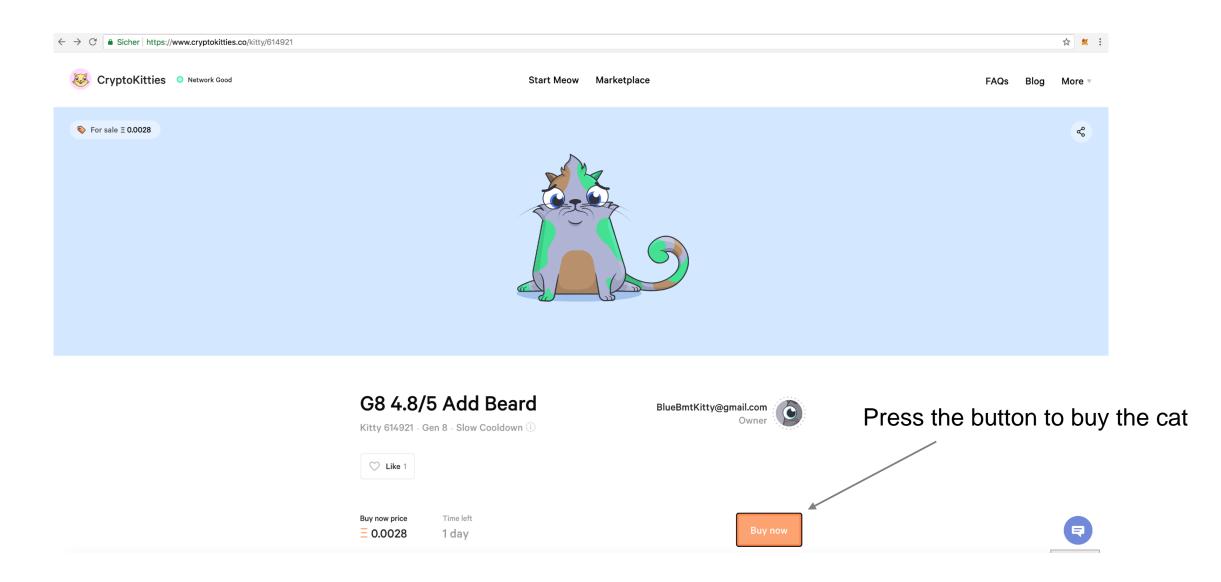
Example: CryptoKitties





Example: CryptoKitties (cont.)

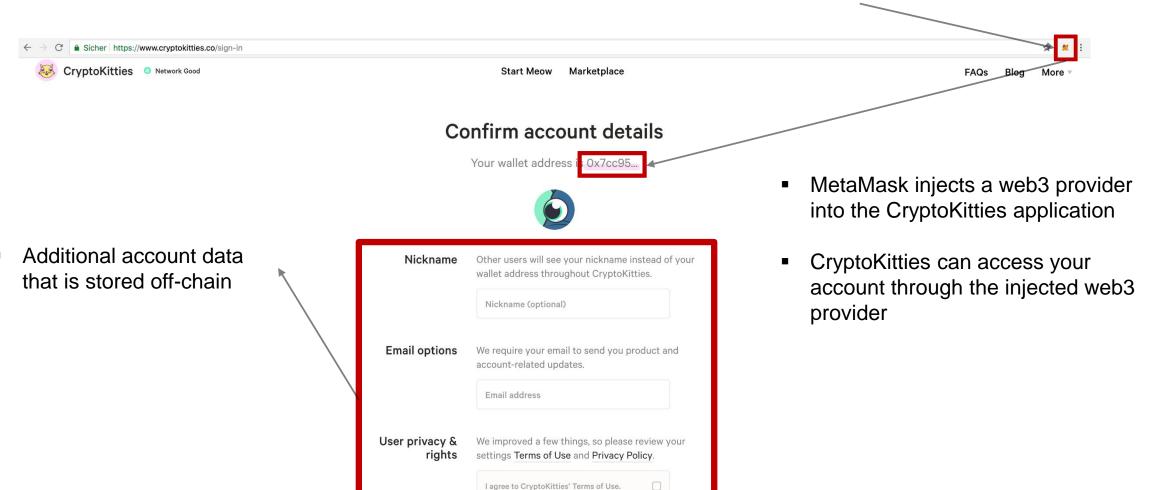




Example: CryptoKitties (cont.)

MetaMask as private key manager





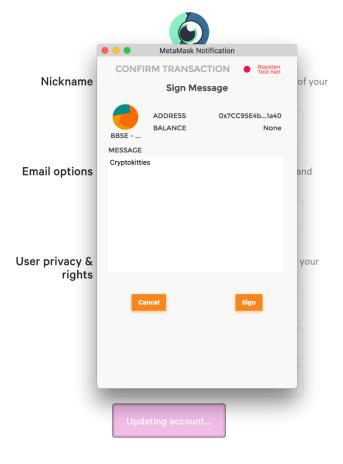
I agree to CryptoKitties' Privacy Policy.

Example: CryptoKitties (cont.)





Your wallet address is 0x7cc95...



- Once the "Sign up" button is pressed, the web application will create a transaction to the CryptoKitties smart contract that stores your account details on the blockchain.
- The unsigned transaction is sent to the private key manager (MetaMask).
- MetaMask asks the user if he/she wants to sign the transaction.
- By clicking "Sign" the transaction is signed and broadcasted to the network



Outline



- 1. Introduction
 - Motivation
 - Definition
 - Benefits
 - Drawbacks

2. Architecture

- Web-based systems
- Private key management

3. Libraries and Frameworks

- Development tools
 - Local IDEs
 - Cloud IDEs
- Test networks
- Web3

Deployment lifecycle



- Compilation of the Solidity source code
- Generate an Application Binary Interface (ABI) in JSON that can be used by other applications to interact with the contract

HelloWorld.sol

```
contract HelloWorld {
    function greet() public returns(bytes32) {
        return "Hello World!";
    }
}
```

```
Compile
```

Deployment lifecycle (cont.)



dApp client or server via Web3

Uses ABI

Uses contract address

Send TX to network

Mined contract

0xb480604052348015600f576

Overview development tools



Local environment

IDE / Code editor









Artifacts

Visual Studio

- Solidity source code files
- Test cases

Build management



Artifacts

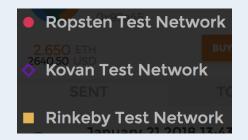
- Compiled .sol files / Bytecode
- Contract ABI / **JSON**

Network deployment

Local test network



Public test network



Main network



Artifacts

- Transaction for creating the contract
- An address for the contract if the creation was successful

Truffle – Development framework for smart contracts



Truffle is a popular framework to facilitate the development of Ethereum smart contracts. The framework provides tools to compile, test and deploy Solidity contracts.



- Open source under the MIT license and hosted at Github: https://github.com/trufflesuite/truffle
- Built-in network management that allows a developer to deploy a smart contract on various network, e.g. live and test
- Web-pack like automated re-compilation on code changes
- Testing based on Mocha and Chai
- Provides project scaffolding

Ganache – Private Ethereum test network



Ganache is a local blockchain for Ethereum smart contract development. It can be used to deploy, simulate, and test smart contracts.

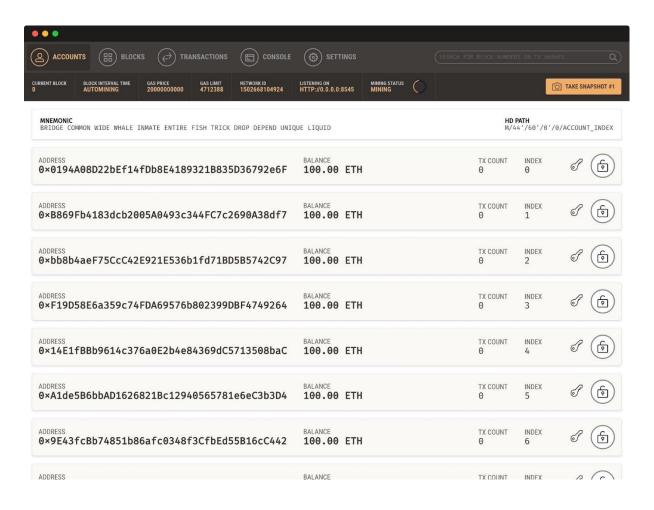


- Open source under the MIT license and hosted at Github: https://github.com/trufflesuite/ganache
- Integrates a custom block explorer interface with additional debugging features.
- Uses workspaces to provide multiple Ethereum blockchains with different settings (blocktime etc.)
- Can be linked to Truffle projects to automate tests for smart contracts

Ganache – Private Ethereum test network (cont.)



Ganache ships with a ready-made and developer friendly block explorer



Overview development tools (cont.)



Cloud environment

IDE / Code editor + Build management



Artifacts

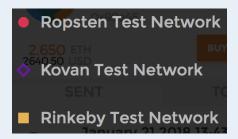
- Solidity source code files
- Compiled contracts bytecode
- Contract ABIs / JSON

Network deployment

Remix emulation



Public test network



Main network



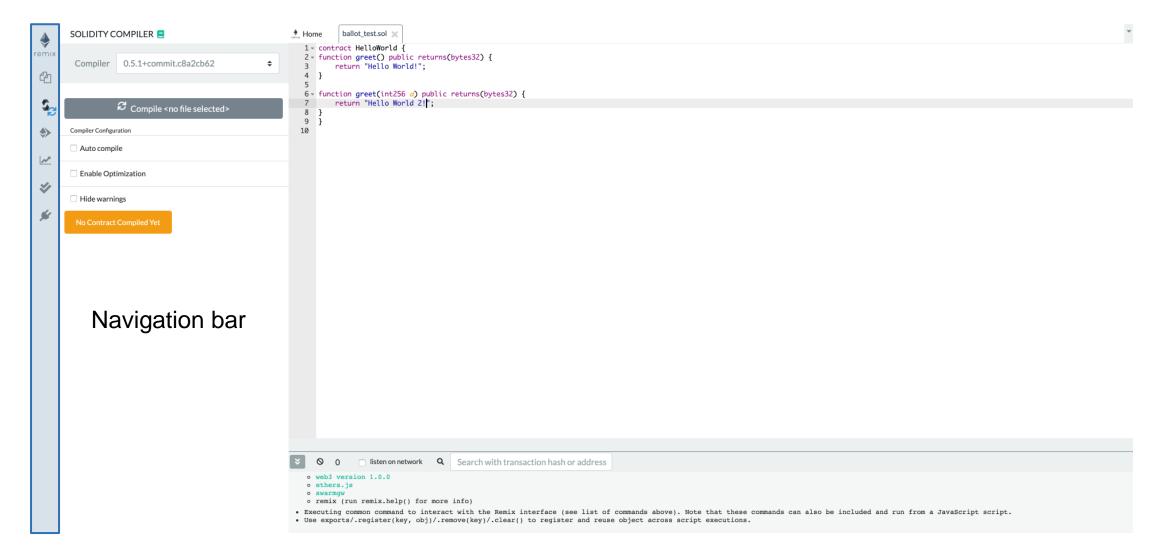
Artifacts

- Transaction for creating the contract
- An address for the contract if the creation was successful

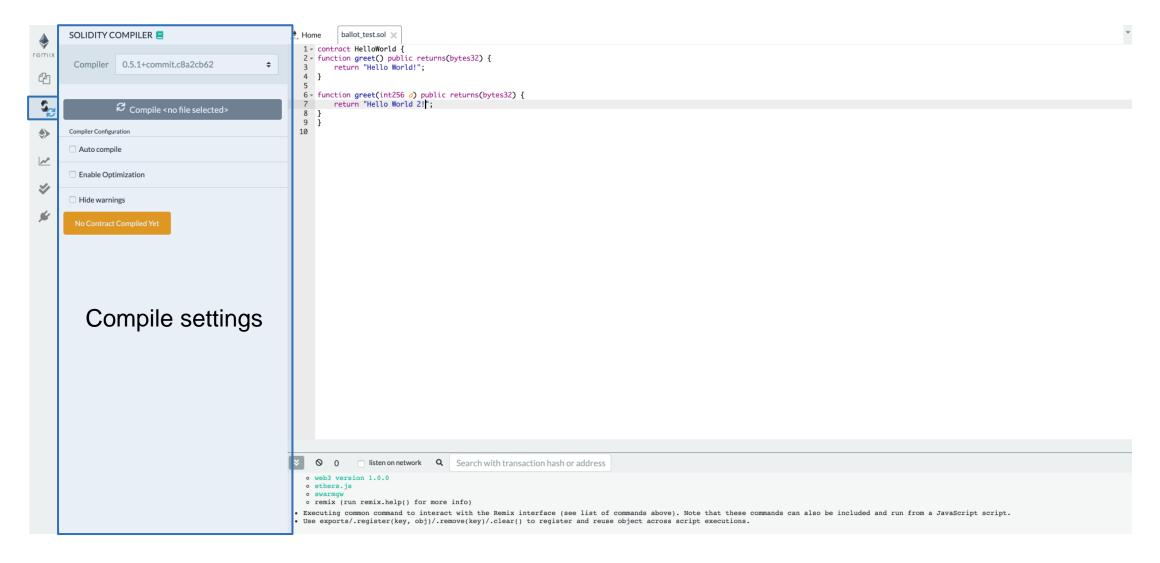




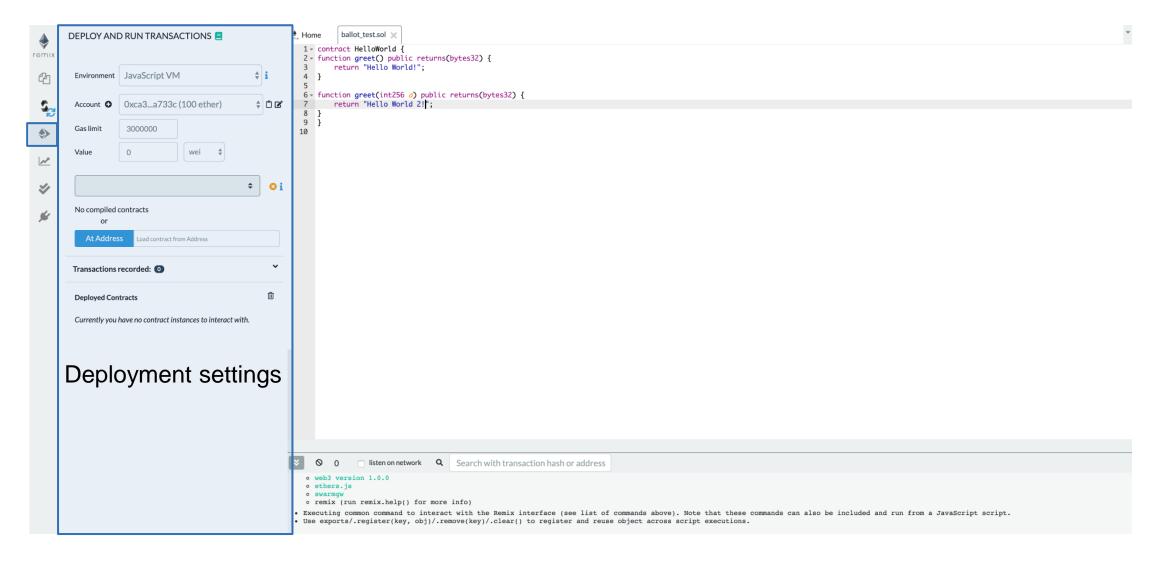












Test networks



Test networks provide a convenient way to publicly deploy and test smart contracts in a realistic environment.

Ropsten

- Free to use
- Public
- Block time of ~30s
- Proof of work consensus
- Geth and Parity compatibility
- Ether distribution via faucet: https://faucet.ropsten.be/
- Anonymous

Rinkeby

- Free to use
- Public
- Block time of ~15s
- Proof of authority consensus, i.e., one central instance decides what transaction will be mined.
- Geth only
- Ether distribution via faucet: <u>https://faucet.rinkeby.io/</u>
- Twitter or Facebook account required

Kovan

- Free to use
- Public
- Block time of ~4s
- Proof of authority consensus, i.e., one central instance decides what transaction will be mined.
- Parity only
- Ether distribution via faucet: <u>https://github.com/kovan-</u> testnet/faucet
- Github account required

Web3.js

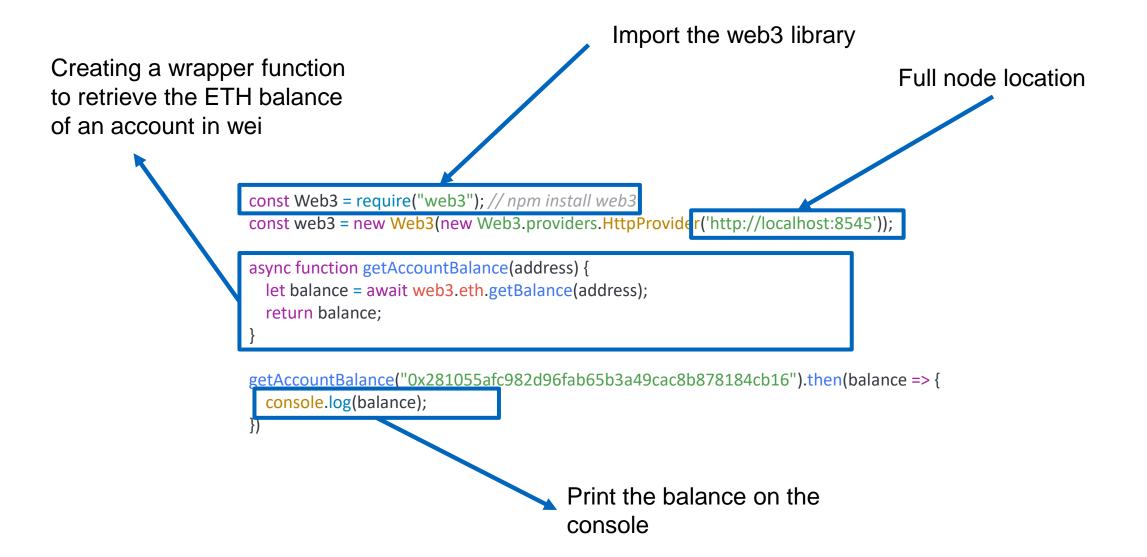


Web3.js is the **official Ethereum JavaScript API** that provides a **wrapper** for the **JSON RPC interface** to interact directly with the blockchain.

- Usually used with a local full node. However, any full node that provides a publicly accessible RPC interface can be used. A frequently used full node provider is Infura.io (https://infura.io/)
- The Framework is available as a npm module (npm install web3)
- Provides methods to deploy and interact with smart contracts
- Provides methods to sign and send transactions
- Uses callback functions and promises for asynchronous events
- Widely used framework to interact with the Ethereum Blockchain
- Can be used server-side (via node) and client-side

Example: Getting the balance of a specific account







Example: Sending Ether to another account

```
var gasPrice = 2;//or get with web3.eth.gasPrice
var gasLimit = 3000000;
var rawTransaction = {
"from": addr,
"nonce": web3.toHex(nonce), //or use web3.eth.getTransactionCount(address, 'pending')
"gasPrice": web3.toHex(gasPrice * 1e9),
"gasLimit": web3.toHex(gasLimit),
"to": toAddress,
"value": amountToSend,
"chainId": 4 // Id of the blockchain (1=main net)
var privKey = new Buffer('0x....', 'hex');
var tx = new Tx(rawTransaction);
tx.sign(privKey);
var serializedTx = tx.serialize();
web3.eth.sendRawTransaction('0x' + serializedTx.toString('hex'), function(err, hash) {
       if (!err) {
                console.log('Txn Sent and hash is '+hash);
        else {
                console.error(err);
});
```

Working with smart contracts in JavaScript



Create contract object in JavaScript that references an existing contract on the blockchain.

- Pass ABI object as first constructor parameter
- Pass contract address as second parameter

```
const MyContract = new web3.eth.Contract([
    "constant": false,
    "inputs": [],
    "name": "greet",
    "outputs": [
        "name": "".
        "type": "bytes32"
    "payable": false,
    "stateMutability": "nonpayable",
    "type": "function"
 "0xC04229E8Edd4402D030cf81efF3e25df0E84BAA1");
```

Call contract method

 Methods defined in the ABI can be accessed via the global methods object

```
MyContract.methods.greet().send({from:
'0xcc8d743....97278fe497ee90...'},
(error, txHash) => {
      if(err) {
      // Handle error here
      else {
      // No error occured
});
```

Recap: Solidity Events



- Convenient way to listen for smart contract function calls and state changes without scanning the whole blockchain.
- Mostly used to trigger callback function in dApp event listeners.
- Fast way for third party (Web3) applications to check what happened in a certain block.

Events and logs are expected to change in the future. Currently, there are discussions in the Ethereum community to remove them from the persistent blockchain state.

Example: Subscribing to future "Transfer" events in JavaScript



Accessing contract events

- Events can be accessed via the global events object
- Each event emitter can emit 3 types of events:
 - data: Fires on each incoming event with the event object as argument.
 - changed: Fires on each event which was removed from the blockchain.¹
 - error: Fires when an error in the subscription occurs.

```
MyContract.events.Transfer({fromBlock: 0},
    function(error, event){ console.log(error) })
        .on('data', (log) => {
            console.log(`Data: ${log}`)
        })
        .on('changed', (log) => {
            console.log(`Changed: ${log}`)
        })
        .on('error', (log) => {
            console.log(`Error: ${log}`)
        });
}
```

¹ Removed from Blockchain means that the event was created inside a block which was later orphaned. See details here: https://ethereum.stackexchange.com/questions/7905/ethereum-event-log-removed-field

Web3 implementation for other languages



Web3 is available for other languages besides JavaScript, too. However, not all of those implementations are officially maintained by the Ethereum foundation.

Python (official)

Web3.py - https://github.com/ethereum/web3.py

Haskell

Hs-web3 - https://github.com/airalab/hs-web3

Java

Web3j - https://github.com/web3j/web3j

Scala

Web3-scala - https://github.com/mslinn/web3j-scala

Purescript

purescript-web3 - https://github.com/f-o-a-m/purescript-web3