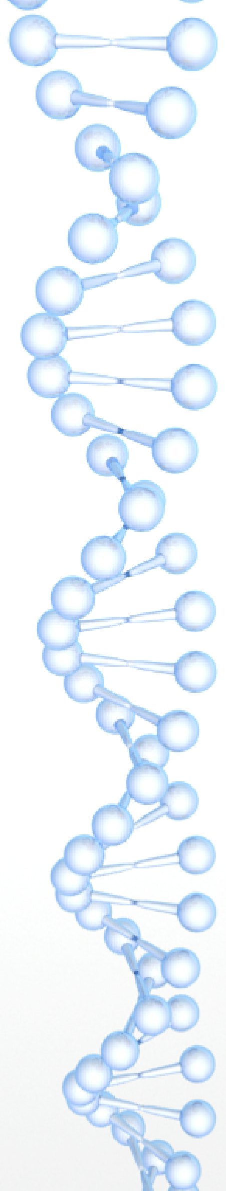
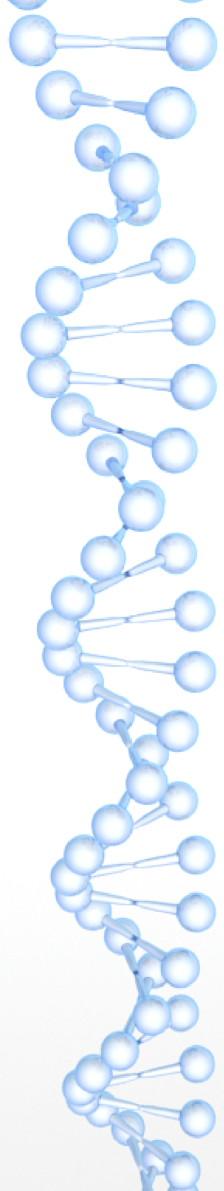


Beyond Testing: Go-Ethereum, Node.js, and Web3.js

University of Missouri – St. Louis
Austin Hester



Go-Ethereum aka geth

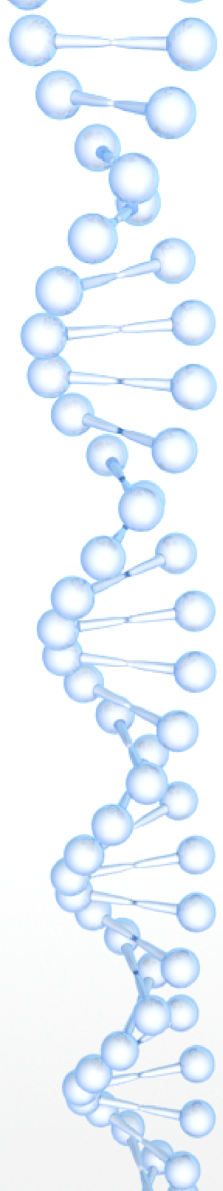
- We have seen `geth` before, and we can use its console to interact with our blockchain and smart contracts.
- Getting a contract instance is simple in geth console; you only need the contract *Application Binary Interface* and *deployed address*.
- Unlike Truffle, we do **not** need to specify the difference between a call and a transaction when using `web3.eth.contract(<abi>).at(<addr>)` to interact with a smart contract.
- Geth console is based off Node.js, and it includes some built-in packages such as `Web3.js` for making `HTTP JSON-RPC` calls.
- We will see later how we can use Node.js directly to interact with our blockchain and smart contracts.

Application Binary Interface (ABI)

- The *Application Binary Interface* is the standard way to interact with contracts in the Ethereum ecosystem, both from outside the blockchain and for contract-to-contract interaction.

```
pragma solidity ^0.4.0;
contract Foo {
        function bar(uint32 x, bool y) public pure returns (bool r) {
                return (x > 32 || y);
        }
}
```

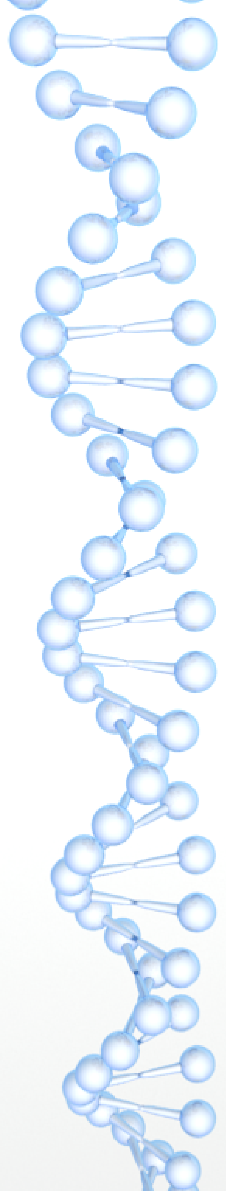
- To call the function `bar` with parameters `68` and `true`, we pass 68 bytes of *input data* along with our transaction to the contract address.
  - `0xcdcd77c0`: the method ID, or the first 4 bytes of the *Keccak hash* of ASCII form of `bar(uint32,bool)`.
  - `0x0000…000044`: the first parameter, uint32 value of `68`, padded to 32 bytes.
  - `0x0000…000001`: the second parameter, boolean `true`, padded to 32 bytes.
- In total: `0xcdcd77c00000...0000440000...000001` will be the input data.



Web3.js, JavaScript API

- Web3.js is a Node.js library with many uses. It allows communication with the blockchain and smart contracts in an external environment.
- There are multiple modules included:
  - `web3-eth` is for Ethereum and smart contracts.
  - `web3-shh` is a p2p communication and broadcast protocol.
  - `web3-bzz` is a swarm protocol for decentralized file storage.
  - `web3-utils` has useful helper functions.
- We will focus on `web3-eth` for now, documentation here:
  - https://web3js.readthedocs.io/en/1.0/web3-eth.html
- To install: `npm install -g web3`
  - The following link will direct you when encountering the inevitable “git clone permission denied”.
  - https://docs.npmjs.com/getting-started/fixing-npm-permissions

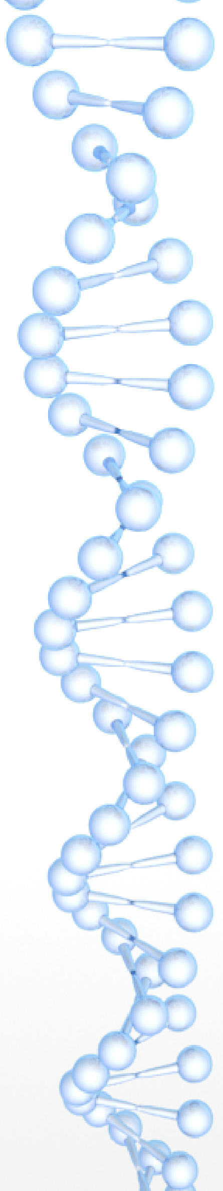
Web3.js Use Cases



- Interact with smart contract instances using Node.js console or external scripts.
- Listen for events triggered by smart contract using `event.watch(function(err, res) { doSomething(); }).`
- Get more information about the blockchain than you'll ever want.
- Send transactions autonomously.
- Compile solidity source files.
- Interact with your accounts.
- Combine all of the above into some extraordinary autonomous regulator.

Web3.js for Interaction with Blockchain

- *Why* would we want to do this?
  - Automation/ scripting.
  - Listening for events and acting upon them.
  - Build a website using web3.js as an API.
  - Anything Node.js can do may be triggered by a smart contract.
    - Can it…
      - Change options of video streaming?
      - Send push notifications to users?
      - Ban/ block a nefarious user?
      - Add a new peer to our blockchain through a website?
    - Anything is possible.



Adding Web3.js to Your Project

- Web3 communicates to a *local* go-ethereum node using JSON-RPC calls.
- JSON-RPC is a stateless, light-weight remote procedure call (RPC) protocol.
- When running geth, use the `–rpc` option to access the HTTP JSON-RPC and define the port with `–rpcport` <*port*>
- To begin using Web3 in Node.js, simply set the HTTP RPC provider, as in the snippet below.

```
var Web3 = require(‘web3’);
var web3;

if (typeof web3 !== ‘undefined’)
     web3 = new Web3(web3.currentProvider);
else
     web3 = new Web3(new Web3.providers.HttpProvider(“host:port”));
```

Example: Interacting with Contracts

loadContract.js

```
var Web3 = require(‘web3’);
var web3 = new Web3(new Web3.providers.HttpProvider(“host:port”));
module.exports = {
     function loadContract(abi, address) {
          var abi = web3.eth.contract(abi);
          return abi.at(address);
     }
}
```

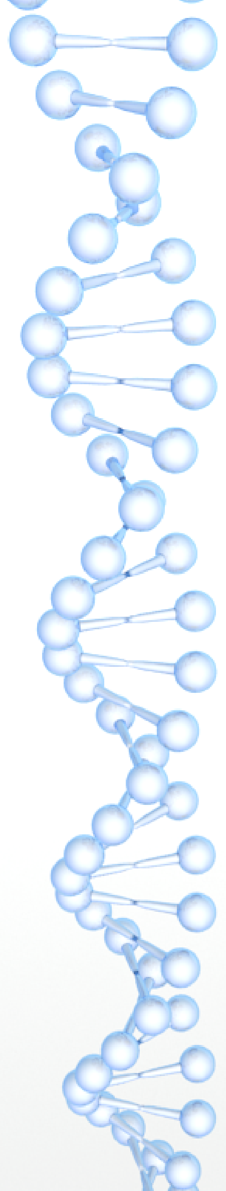
useContract.js

```
var Web3 = require(‘web3’);
var loadContract = require(‘./loadContract.js’);
var web3 = new Web3(new Web3.providers.HttpProvider(“host:port”));
var instance = loadContract(<abi>, <address>);

// make a transfer and get balance
instance.transfer(<to_address>, <value>, {from: <from_address>});

var balance = instance.getBalance(web3.eth.accounts[0]); // return BigNumber
console.log(balance.toString(10));
```

!! Disclaimer: This is just the gist, not tested source.



Example: Listening for Events

- Smart contracts can trigger man-made *events* when anything occurs.
- The following snippet will listen for any event from `MyCoin` and log the results.

listenEvent.js

```
var Web3 = require(‘web3’);
var loadContract = require(‘./loadContract.js’);
var web3 = new Web3(new Web3.providers.HttpProvider(“host:port”));

var MyCoinInstance = loadContract(<MyCoin_abi>, <MyCoin_address>);
var event = MyCoinInstance.allEvents();

event.watch(function(error, result) {
     if (!error)
          console.log(result);
});
```

Preloading Scripts in Geth

- The `–preload <path-to-script>` option for geth allows loading a script when starting a node.
- A great use case for preloading is *automining*. In `private-chain/scripts` there is a script which all miners preload. This script is called `mine_on_demand.js`:
  - Adds a *filter* for pending and latest blocks with a callback function.
  - This filter calls the `checkWork()` function when a new block appears or changes.
  - If there are transactions in the next or previous block, it begins to mine.
  - If there is a transaction with extra data, *esp*. a contract submission, then it sets a counter to mine the next four blocks without discretion.
  - If there are no new transactions and four blocks have past since the latest contract submission, then it stops mining.

References

- https://github.com/ethereum/wiki/wiki/JSON-RPC
- https://web3js.readthedocs.io/en/1.0/getting-started.html
- https://solidity.readthedocs.io/en/develop/abi-spec.html
- https://github.com/ahester57/private-chain/tree/master/doc
- https://ethereum.stackexchange.com/questions/2531/common-useful-javascript-snippets-for-geth/2541#2541