Blockchain Bootcamp Day 12

* Can interact with rpc server with web.js
* Interact rpc server from blockchain node.
* Transfer ether from ganache(fake ether) to rpc server.
* Use api to interact with ganache.
* Transfer 1 ether from address1 to address2 both are from ganache.
* Use the web3 package
* Use the web3.providers
* HttpProvider is a classical provider connect via http api.
* IpcProvider is a provider via a file, that uses Ethereum.
* WebsocketProvider it keeps the connection open, therefore will only send one request. 🡪 send data across wire
* Web3.eth 🡪 Uses Ethereum.
* Web3.eth.getBalance(pass address)
* Web3.eth.sendTransaction({from: address, to: address, value: web3.utils.toWei(single digit number, and unit of measurement.) }) takes a from and to send ether from one address to another. Also pass value in wei
* When deploying smart contract to use locally, deploy on http://localhost:7545, then also make to look at EVM array after deploying smart contract.
* Use the first 8 letters of hash(0x608060 for example) in deployed smart contract, call to blockchain using web3 from console.
* Function in web3 can calculate cezzach cache.
* Web3.utils.sha3(“functionName()”)
* web3.eth.call({ from: "0x821F8c4293aa797E2bcAb92f69321e18AC1340B9", to: "0x6Ca5a79743615128E979A2C4080556ec34eF3Fb6", data: web3.utils.sha3("myUint()").substr(0, 10)}).then(console.log);
* Can use web3 to interact with ABI array. Such as web3.eth.Contract(ABI Array of smart contract, address of eoa deploying contract)
* Can interact with methods of the smart contract via the <contract variable>.methods.<method name>.call().then()
* Can send a transaction like this for example that will return a transaction object.
* contract\_3.methods.setUint(59).send({from: "0xD0C5E53d0E854bAA658332cEDC817e7eb786aE12"}).then(console.log);
* web3.js-browser to interact with browser using web3.js
* WebJS using rpc interface
* How does a blockchain node connect to other blockchain nodes?
* When it talks to other blockchain nodes, it uses Ethereum protocol.
* It uses rpc, http, ipc interface.
* Inter process communication, it’s a file where you can pipe commands through
* Websockets -> bi directional duplex protocol, send messages to client, while rpc open a connect from client, and get data back.
* Connect blockchain node to javascript
* Javascript can connect to blockchain using ws, rpc, and ipc via console
* Not limited to web3.js to interact with blockchain via smart contract.
* Ganache opens a rpc server
* Geth uses a IPC endpoint.
* How website and blockchains interact -> webserver -> blockchain node -> browser
* http restful interface, ganache no need to type command,
* geth –rpc, and besu –rpc.(8545 or 9545)
* Expose and API to interact with the blockchain, similar to mysql
* Problems come up wit cors.
* Web3.js -> normal javascript library, abstract json-rpc on the js-console.
* Web.js modules -> eth module for interacting with Ethereum network.
* Access to blockchain is not a magic new protocol.
* Blockchain node must be running in the background and responding.
* Web3.js it is just a javascript library and easier to interact with blockchain due to no need to decode or encode manually.
* Restful services comes with http security like core.
* Standard port is 8545.