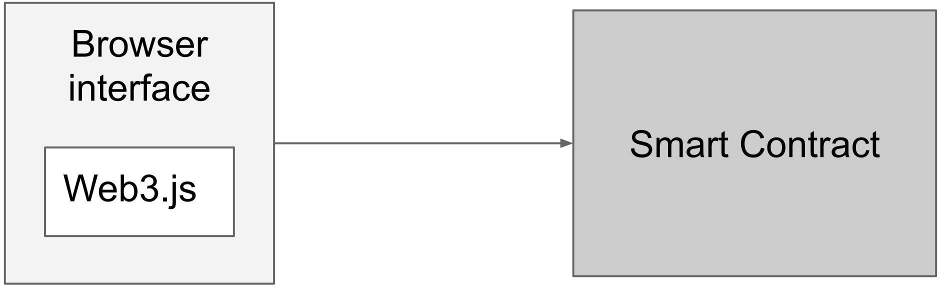
BCDV1011 Design Patterns for Blockchain

Common architectures

Common architectures for Ethereum dApps

- Simple browser dApp
- Simple mobile dApp
- dApp with server
- Hybrid database/dApp
- Simple contract
- Token
- Non-fungible token with factory
- IPFS
- Oracle
- Custodial vs Non-custodial

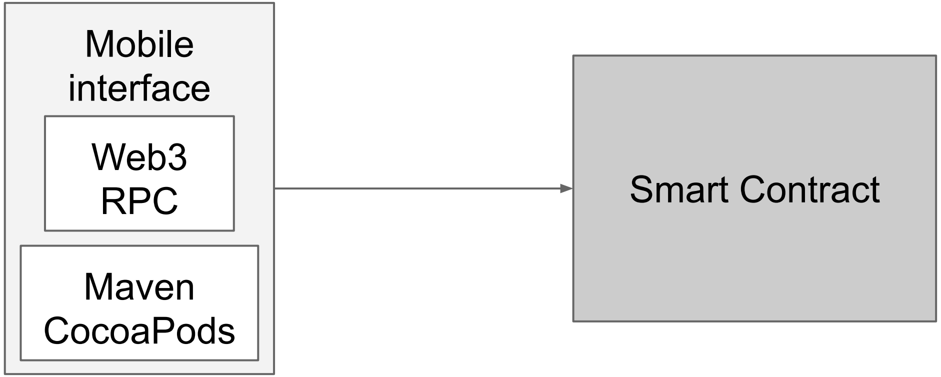
Simple browser dApp



Simple browser dApp

- Browser interface using web3.js to interact with a smart contract
- Easy to setup and deploy
- Uses Metamask to handle transaction signing

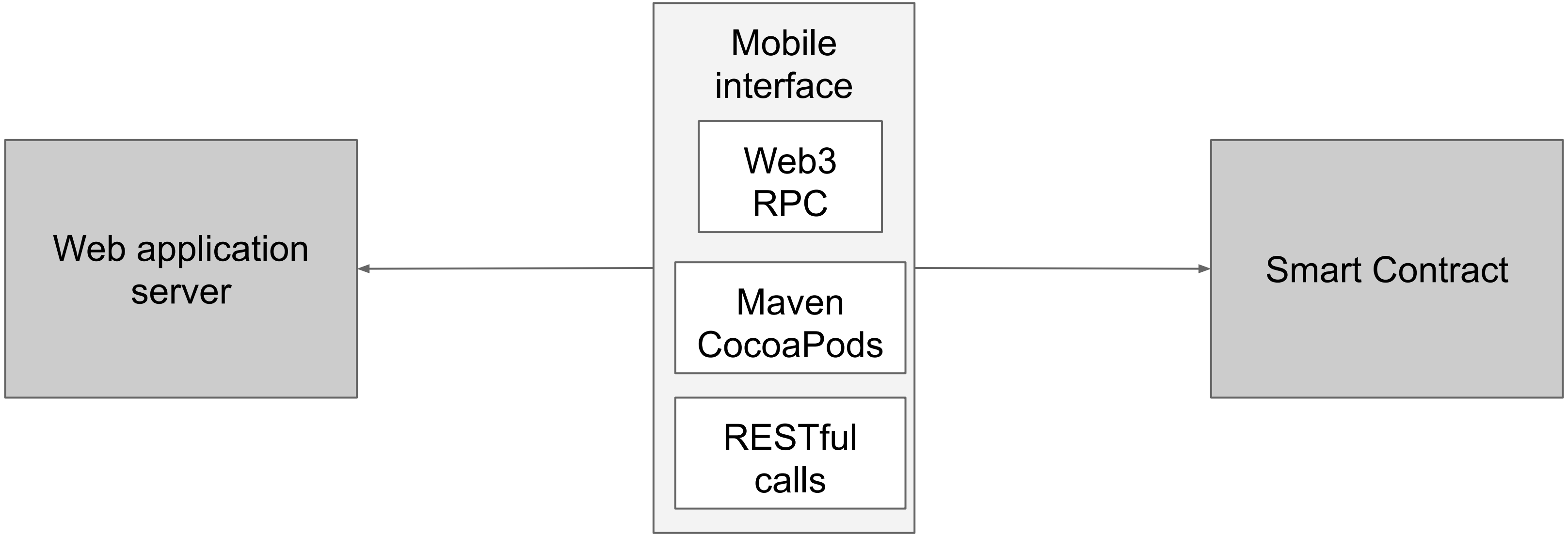
Simple mobile dApp



Simple browser dApp

- Mobile UI, go-etheruem mobile libraries, Web3 RPC
- iOS and/or Android
- Uses libraries for private key management and signing transactions
- Does not require central web application server

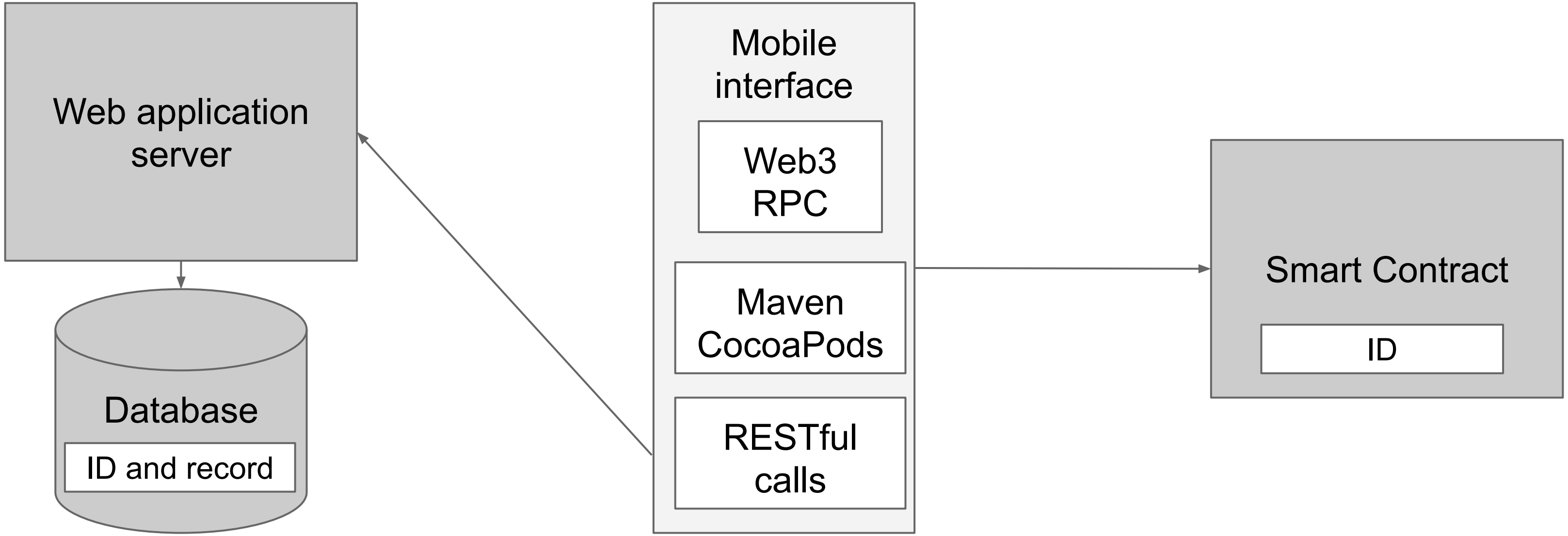
dApp with server



dApp with server

- Mobile or web interface with backend server
- RESTful interface with server
- Server provides
  - More advanced calculations
  - Integration to other systems
  - Access to non-mobile libraries
  - Authentication to services

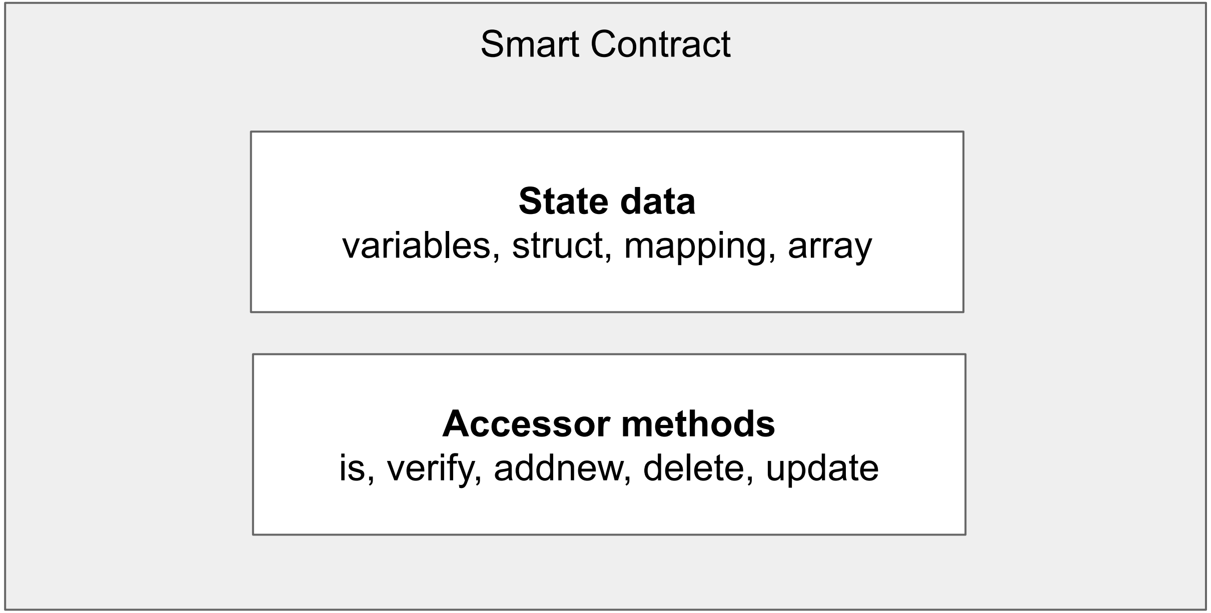
Hybrid database/dApp



Hybrid database/dApp

- Server with database
- ID is stored on blockchain
- Record is stored in central database
- Central database storage is cheaper
- Smart contract does not have access to data record
- NOT decentralized

Simple contract



Smart Contract

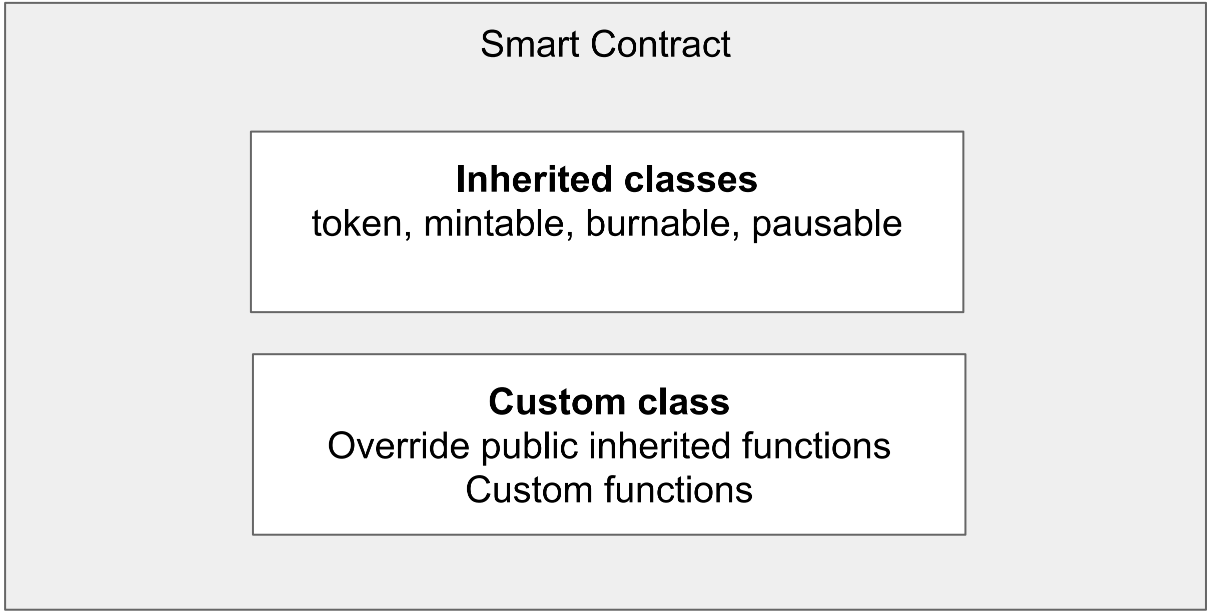
State data
variables, struct, mapping, array

Accessor methods
is, verify, addnew, delete, update

Simple contract

- Custom data storage
- Custom interface

Token



Smart Contract

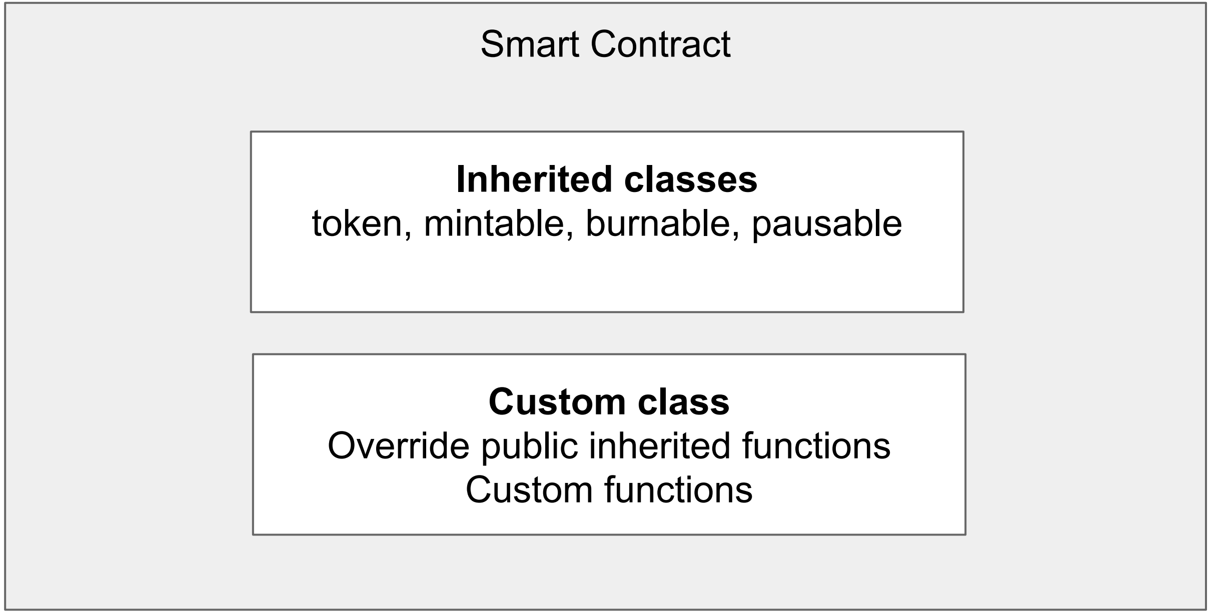
Inherited classes
token, mintable, burnable, pausable

Custom class
Override public inherited functions
Custom functions

Token

- Pre-existing code - tested, vetted
- Standard interface that can be called without further knowledge of your customizations - exchanges
- ERC-20
  - balanceOf , totalSupply , transfer , transferFrom , approve , and allowance
- ERC-1404 - STO
  - detectTransferRestriction, messageForTransferRestriction
- Deploy token and start transferring

Token



Smart Contract

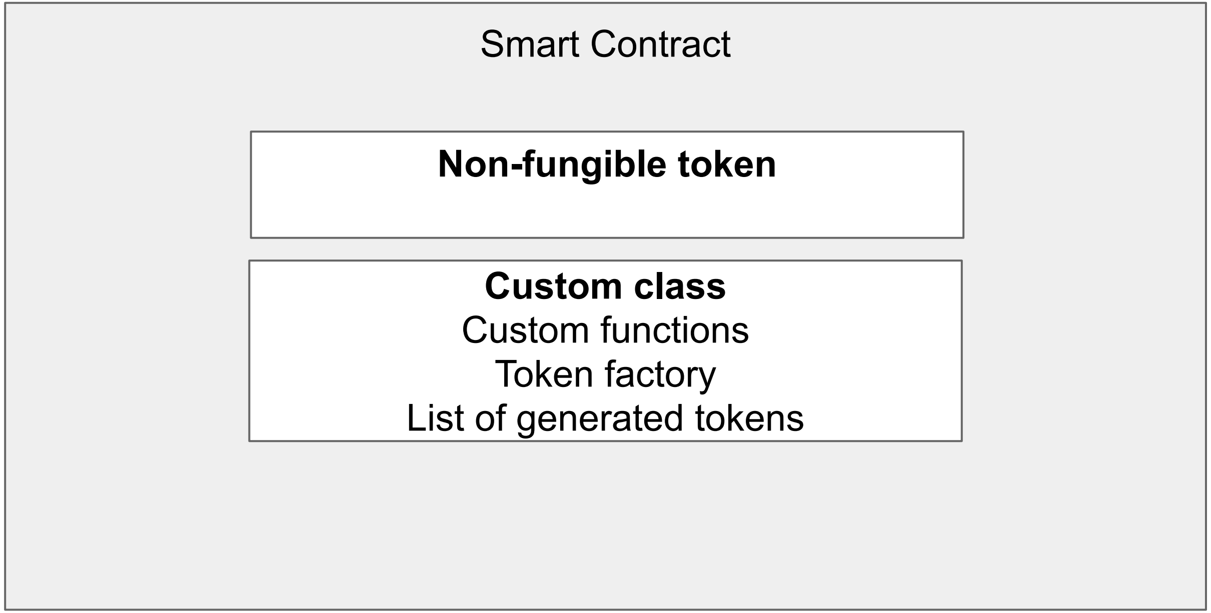
Inherited classes
token, mintable, burnable, pausable

Custom class
Override public inherited functions
Custom functions

Token

- Pre-existing code - tested, vetted
- Standard interface that can be called without further knowledge of your customizations - exchanges
- ERC-20
  - balanceOf , totalSupply , transfer , transferFrom , approve , and allowance
- ERC-1404 - STO
  - detectTransferRestriction, messageForTransferRestriction
- Good for representing fractional ownership in cases like securities

Non-fungible token with factory



Smart Contract

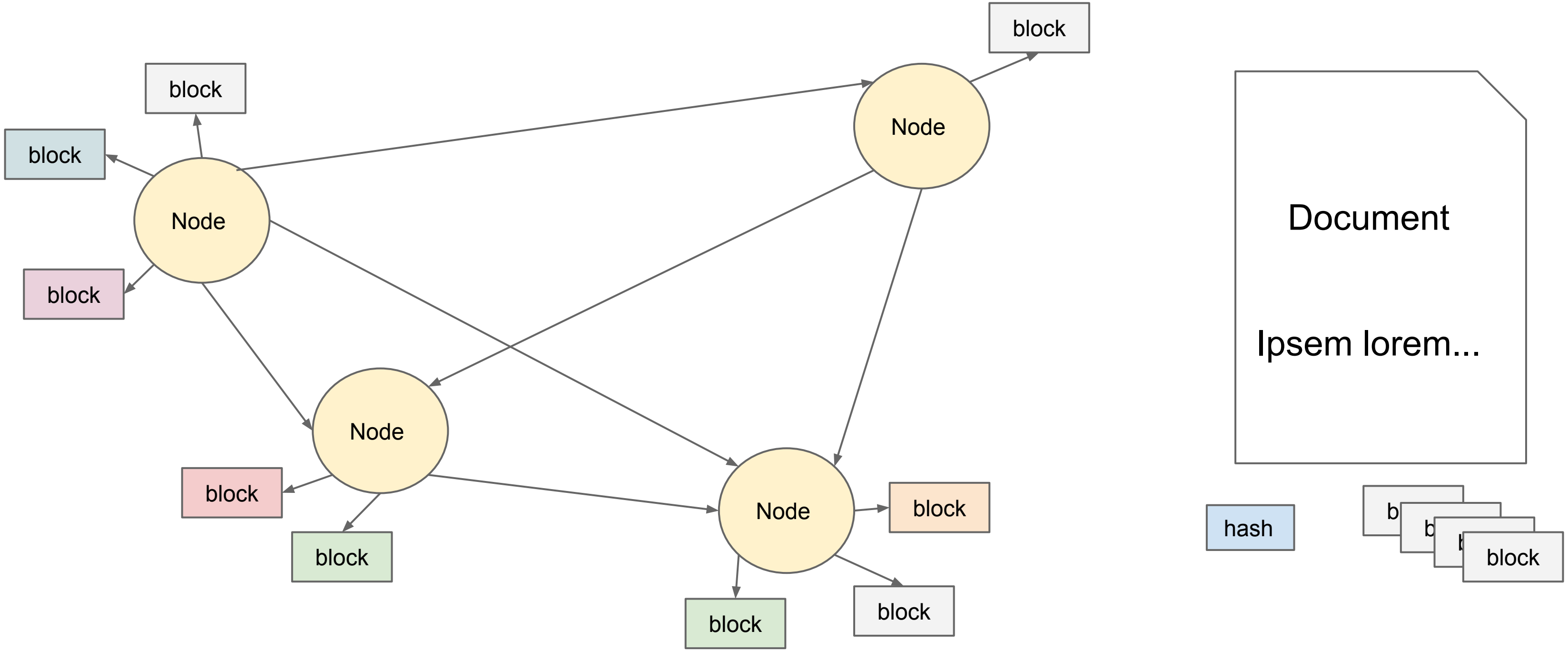
Non-fungible token

Custom class
Custom functions
Token factory
List of generated tokens

Non-fungible token with factory

- ERC-721 non-fungible token with customization
- Custom function to generate new token under conditions
- Keep a list of generate new tokens
- Set the new owner
- The equivalent of minting in the non-fungible world

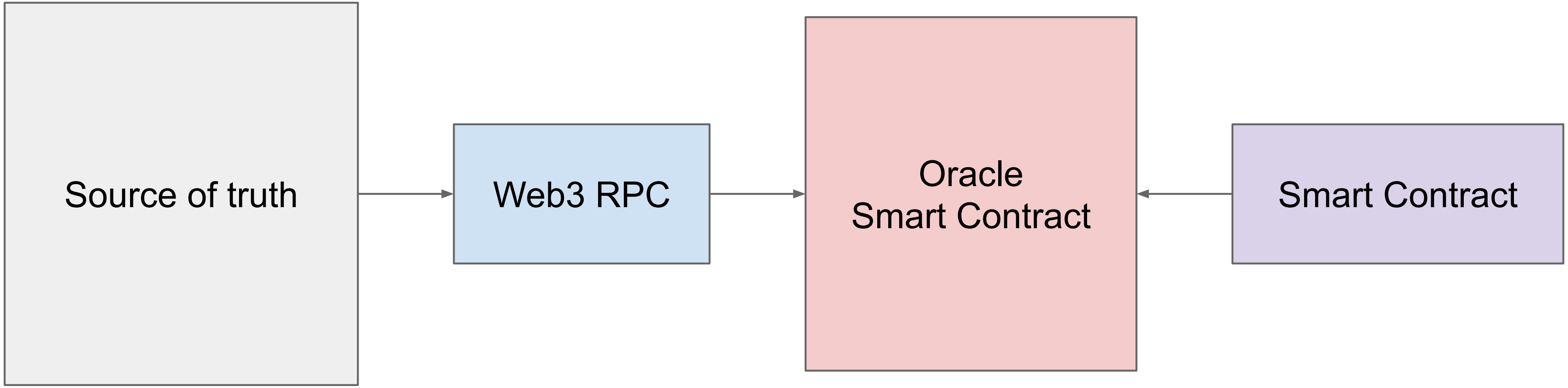
IPFS with blockchain



IPFS with blockchain

- Storing data on Ethereum is expensive
- Storing data in a central database is not distributed
- IPFS is distributed
- IPFS uses the cryptographic hash as the storage and lookup index
- IPFS breaks the file into blocks and the blocks are stored all over the network
- IPFS maintains an index to find the closest copies of all of the blocks to retrieve the file
- Store the hash in the blockchain

Oracle with blockchain

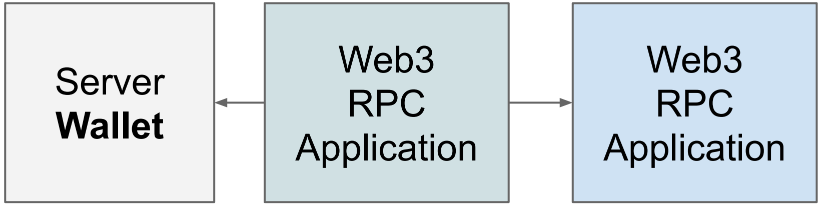


Oracle with blockchain

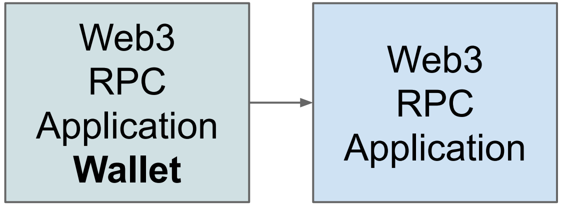
- Smart contracts can only work with data on the blockchain
- In reality you will have cases where you need to make your contract use off-chain data
- Build an application to put the data into a smart contract
- Your smart contract can access the oracle smart contract to access the off-chain data
- You can never fully trust off-chain data

Custodial vs non-custodial

Custodial



Non-Custodial



Custodial vs non-custodial

Custodial

- Your application holds the private key
- You sign the transactions for the user
- Avoids user losing key
- Simplifies UI
- Not secure

Non-custodial

- User holds their key in their own wallet
- Your dApp needs to support various methods for transaction signing
- User can lose key
- Secure

Fully decentralized

Requirements to be fully decentralized

- Non-custodial
- Governance organization
- Open source smart contracts
- Open source UI code
- All data on chain
- Anyone can run UI
- Transparent processes from governance to source