

Ethereum & DeFi for JS/TS developer

ARTEM VOROBEV

1inch Network

Цель доклада:

- Разобраться что такое и как устроен DeFi & Ethereum
- Пощупать инструменты руками ???
- Закодить первую приложуху

Обо мне и 1inch Network

- Разрабатываю под Ethereum 4ре года
- Попал в Ethereum через ETH Global хакатоны
- 1inch Network - хакатон проект который смог
- 1inch Network - Google flight для обмена криптовалюты
- 70% трафика
- Daily Volime

Это может быть интересно, потому что в DeFi:

- Работа с финансами идет в permissionless экосистеме
- Разработчики это ключевые лица
- Высокая зарплата и еще большая личная ответственность



**DeFi - смотреть могут не только лишь
все, мало кто может это делать**

Quick intro в Ethereum и DeFi

Концепт Ethereum — виртуальная машина поверх блокчейна

1. Транзакции запускают код в машине
2. Код меняет `state` машины
3. `state` хранит что угодно

Ethereum — за все надо платить эфиrom

1. MOV ADD XOR AND ... у всего есть цена в Gas Units
2. Лимит элементарных операций на тразакцию и блок
3. Концепт газа: Gas Units * Gas Price (п)

- Газ прайс - наглядность (приложить дорогую транзакцию) и Yellow paper

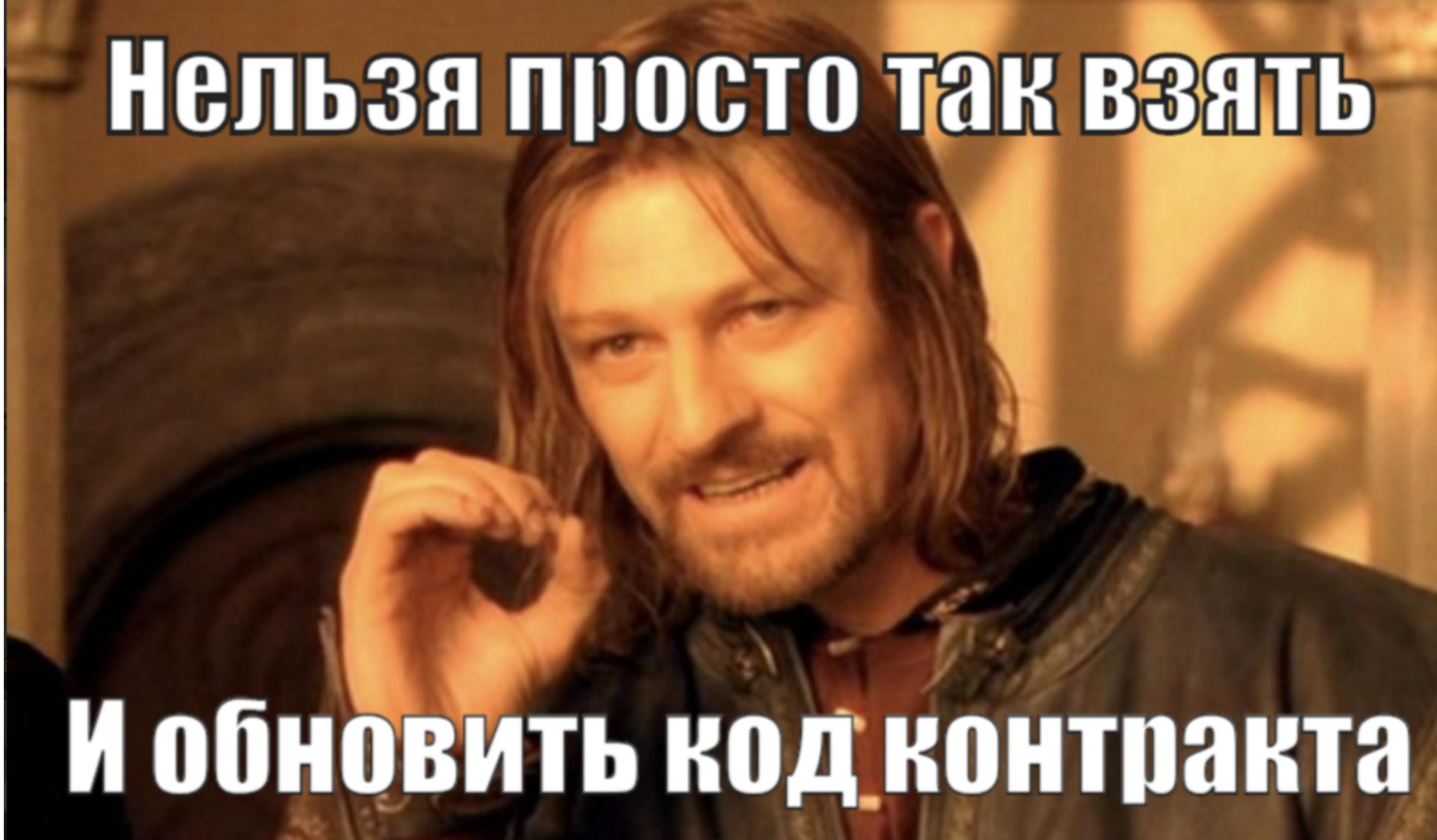
Смарт контракт

- В виртуальную машину можно деплоить программы
- Программа = исполняемый код + состояние
- Это и есть смарт контракт

Код это закон

- Программа делает только то, что в ней запрограммировано
- Программу можно верифицировать - скомпилировать исходный код и сравнить
- Пример, контракт токена [WBTC](#)

А как изменить код контракта?



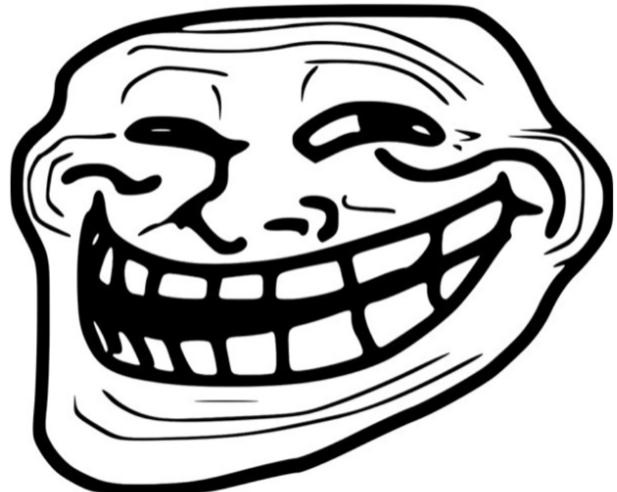
Нельзя просто так взять

И обновить код контракта

Можно:

- Использовать **Proxy** паттерн
- Написать **Upgradable** контракт
- Или даже вызвать деструктор (**selfdestruct**)
- Примеры **OpenZeppelin**

Но ты не написал



ЭТОТ КОД В КОНТРАКТЕ

И это даже хорошо

- Код это закон
- Вопрос доверия

А как понять что контракт обновили ?

А как понять что контракт обновили ?

- Очень просто, у нас же блокчейн
- Можно найти на [etherscan](#)

Если провести аналогию блокчейн и Git:

- Транзакции — коммиты
- Блоки — merge pull request'а в мастер
- У вас нет прав на мастер 😊
- Контракт и транзакции останутся в истории навсегда



Смарт контракты могут работать сообща

1. Пользователь отправляет транзакцию
2. Транзакция запускает метод на контракте
3. Метод контракта может вызвать метод другого контракта

**Что бы контракты были совместимы нужны
интерфейсы**

ERC-20 токены USDT, DAI

**Пример, разрешим контракту 1inch потратить потратить
10,000 BUSD моего кошелька**

Контракты - обменники (AMM, Liquidity Pool)

Пример [uniswap](#)

Обменивает токены по формуле $x * y = \text{const}$

Пример транзакции

Контракты ломбарды

Кредиты в одном токене под залог другого

Контракт может распродать обеспечение

Пример [Compound](#)

**4. Контракты Оракулы - поставщики цены/погоды/
результатов выборов, [Chainlink](#)**

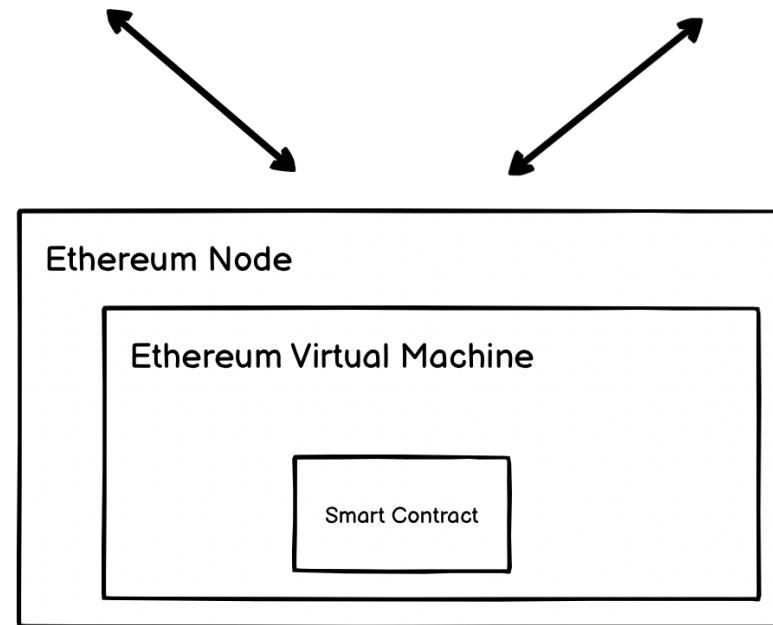
dApp - Decentralized Application

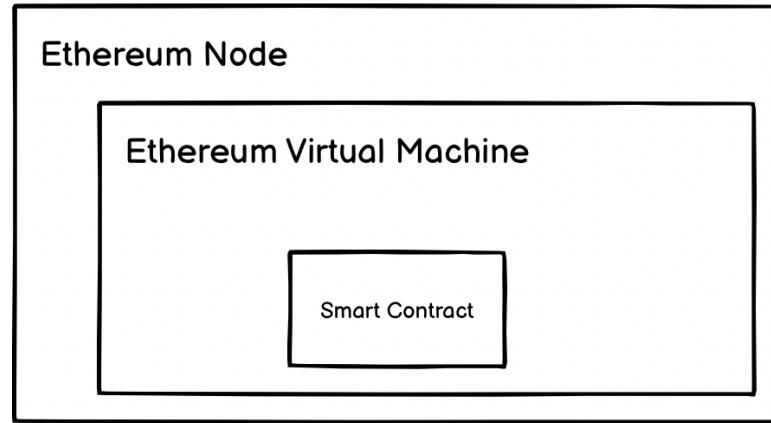
- **Uniswap,**
- **Compound**
- **1inch**, все это dApp

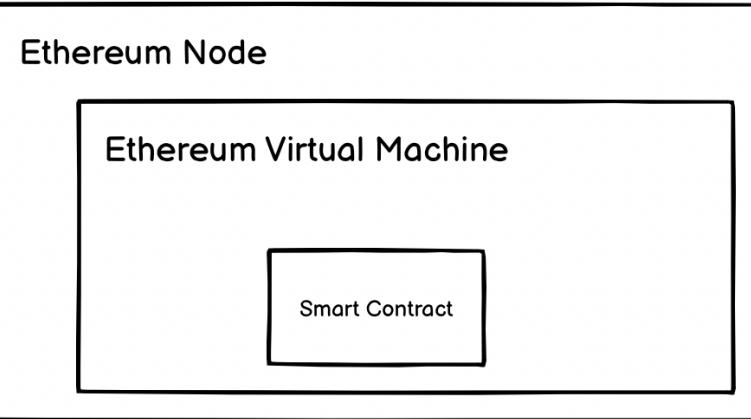


Посмотрим как устроены dApp

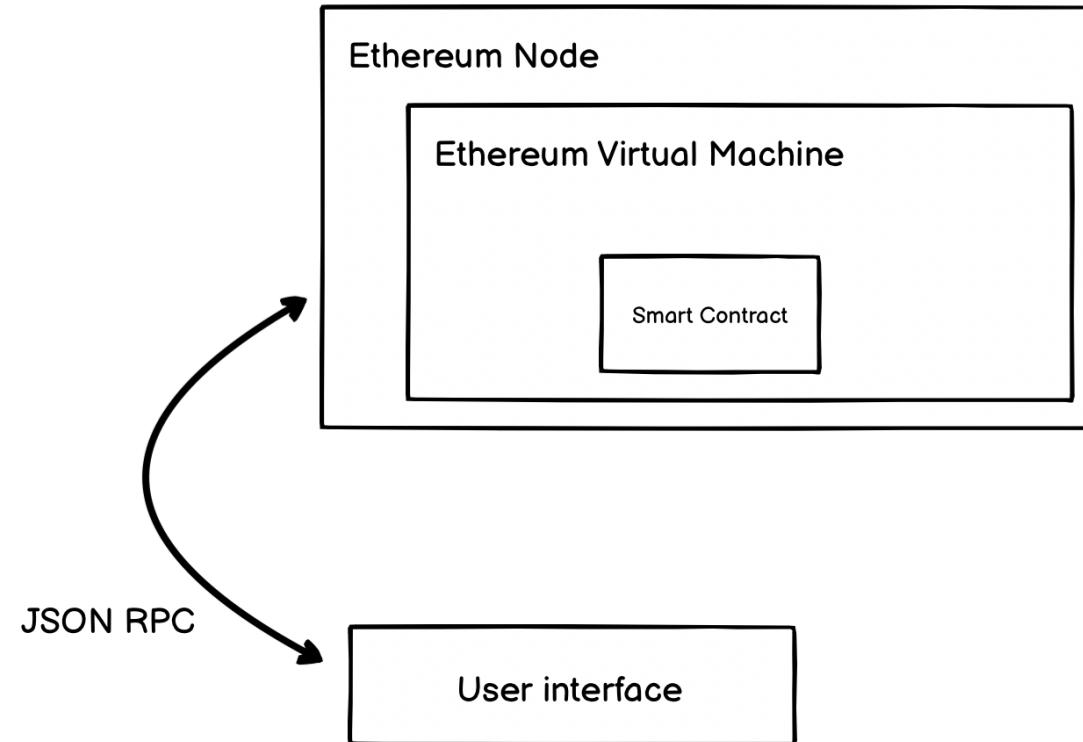
Other Ethereum Node Other Ethereum Node





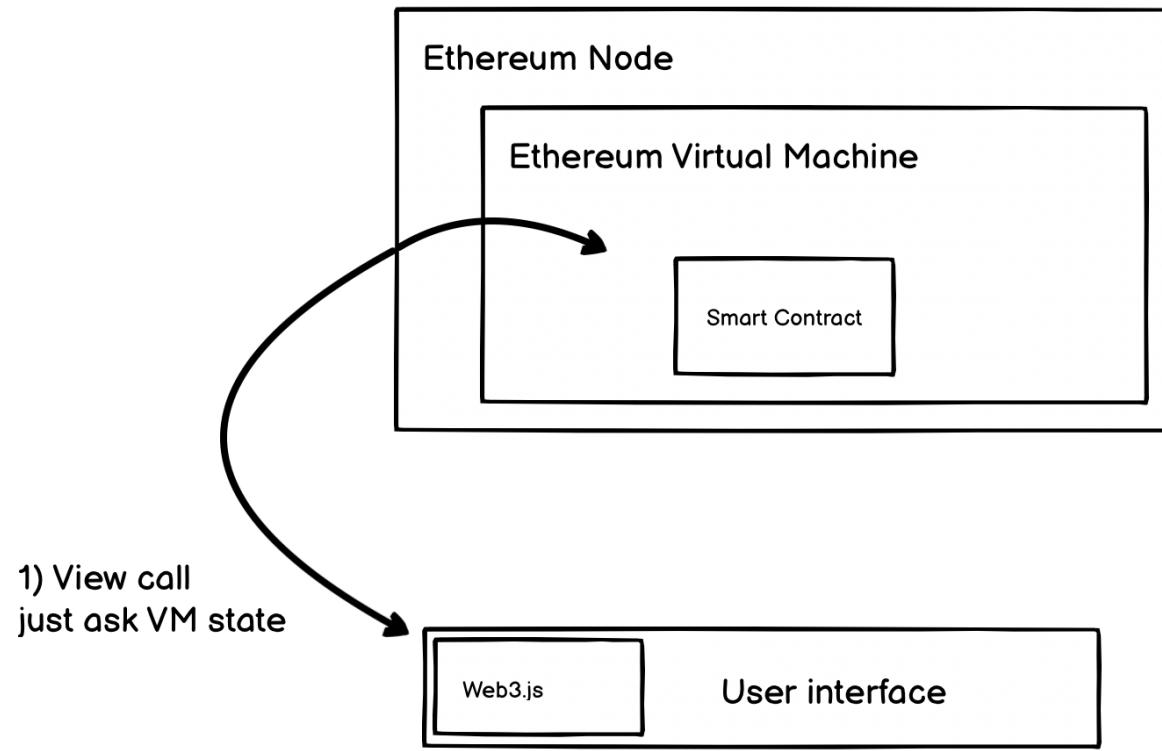


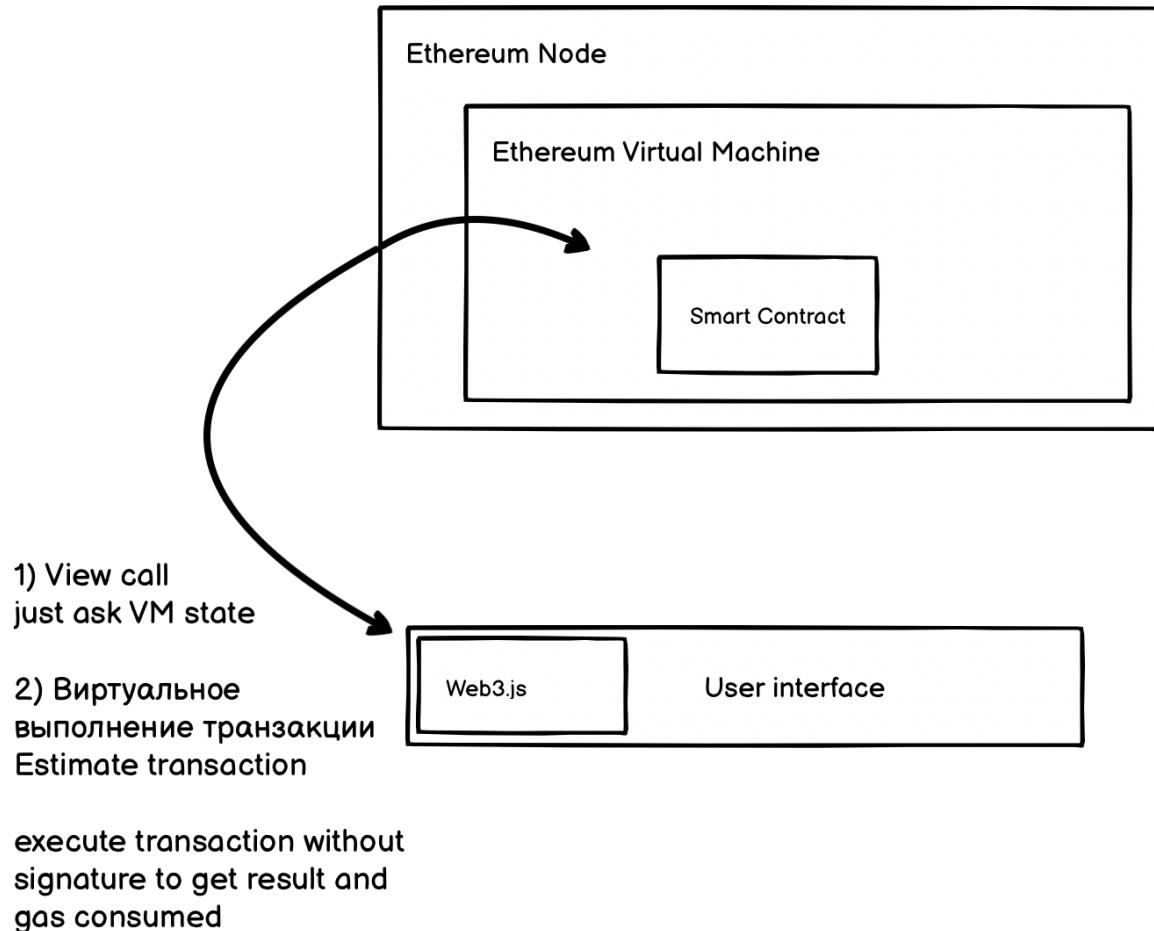
- Usually SPA hosted on CDN with fallback to IPFS
- Usually allow open source to allow everyone to verify
- The matter of trust



1) View режим

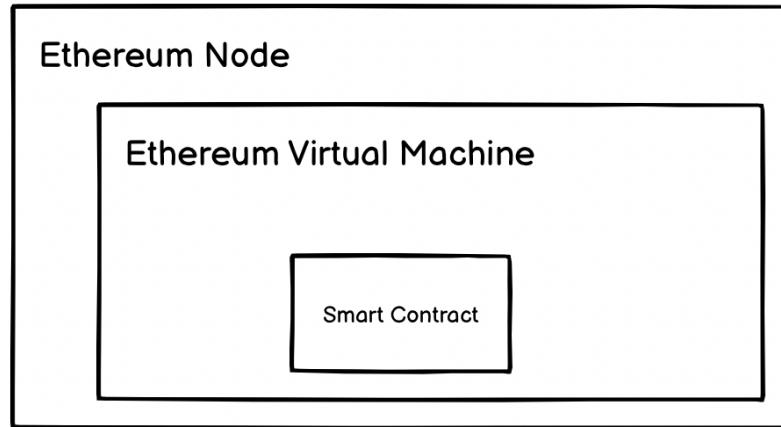
До того как кошелек подключен

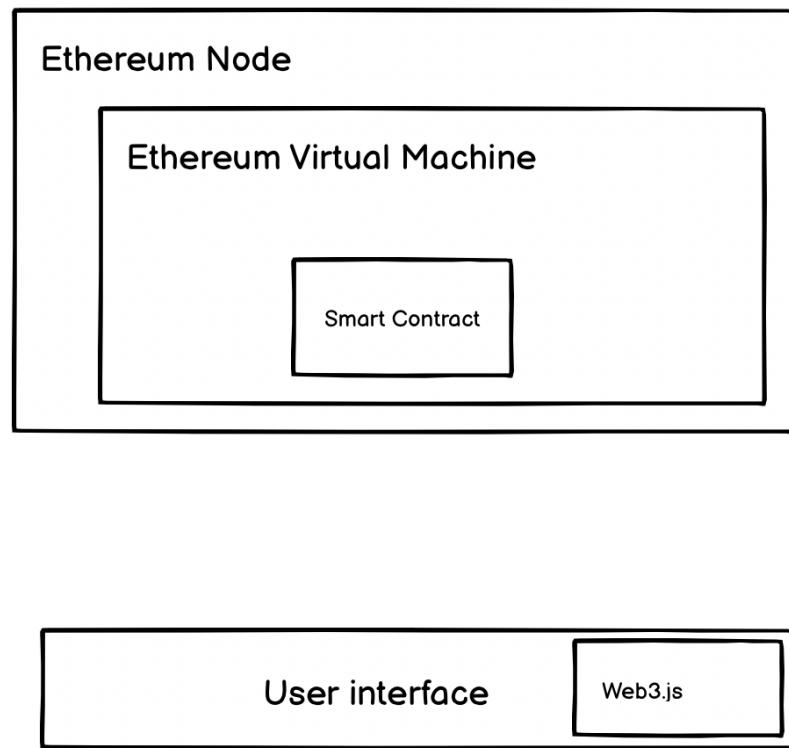




2) Write Режим

Когда кошелек подключен





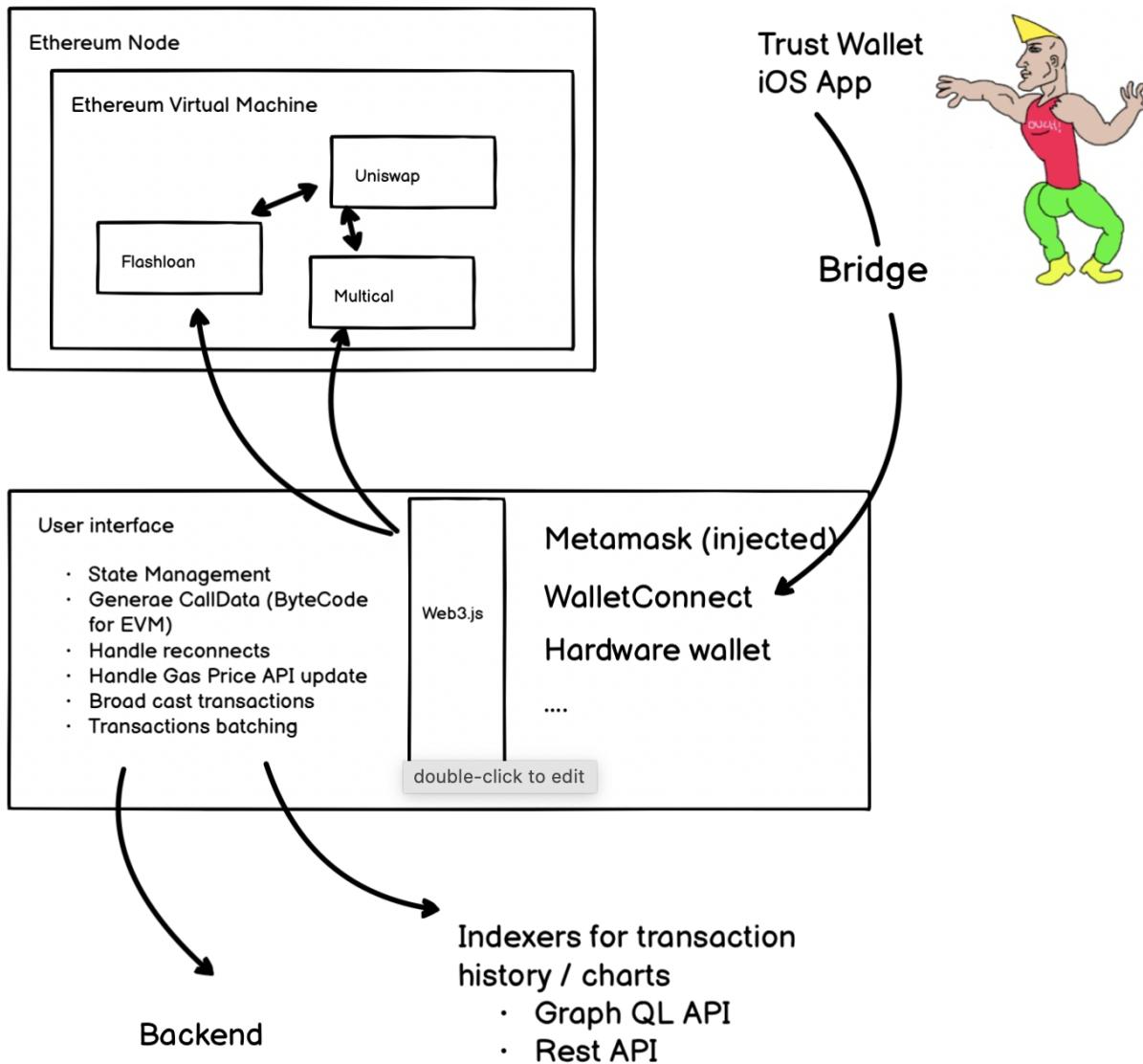
С кошельком появляются

- `SignTransaction`
- `SignAndSendTransaction`

3) Режим Reallife

- Много контрактов
- Нужно составлять сложный байт код вызовов контрактов
- Иногда нужно создавать новые контракты через фабрики
- Много провайдеров кошельков
- Появляется backend
- Некоторые данные можно достать только через индексаторы

Режим реальной жизни



4) Режим Hardcore

Режим Хардкор - Support Several networks

