Трехуровневая схема работы DeFi или сам себе DApp
(очень краткая методичка)

(C) Alex Kruegger, MMXXI
Специально для канала **IDO Research**

Дисклеймер

Данная методичка предназначена, в первую очередь, для крипто энтузиастов, которые хотят научиться разбираться в том крипто мире, в котором они находятся, но при этом не обладают глубокими знаниями о протоколах, внутреннем устройстве блокчейнов и т.д.

Поэтому в данной работе многие понятия сознательно сокращены (впрочем без потери их адекватности и применимости) для лучшего усвоения материала без излишних технических подробностей. Будете критиковать автора - пожалуйста, имейте это в виду.)

Введение

Из крипто-чатов:

*“Ребята объясните доступным языком кто шарит.Зарегал я короче Метамаск,Панкейк и Коинмаркеткап.Хотел приобрести Астру на коинмаркеткап и поэтому вывел bnb из бинанса в метамаск ,а потом произвел конвертацию бнб в астру на коинмаркеткап.Бнб списались из метамаска и по идее Астра должна была улететь **в кошелек Панкейк**,но там она не отображается.Так как я новичок в этой теме может я ,что-то не то исполнил как думаете?”*

*“Мужики, при равных составляющих если разница в комиссии кошельков Metamask и TrustWallet? они же вроде на **разных блокчейнах работают** или это не влияет”*

Перечитывая своим методички я вдруг понял, что работая над ними упустил из виду один, на самом деле самый важный момент в DeFi - как связаны и как взаимодействуют между собой три главных компонента - Блокчейн, Смарт-Контракты и Веб интерфейс.

Без четкого понимания этой трехуровневой структуры Вам будет сложно (и я вижу по постам и вопросам насколько это сложно) представить всю техническую экосистему DeFi. А без этого представления у вас в голове так и будут путаться метамаски, панкейки, блокчейны и прочая неведомая живность.

Данный материал представляет собой более философский, чем технический взгляд на DeFi, но, на мой взгляд, является основополагающим для создания целостной картины.

Итак начнем:

Нижний уровень - блокчейн.

Давайте вспомним, что мы с вами узнали из самой первой методички:

Блокчейны состояния

- Etherium и блокчейны, построенные на его основе (Polygon/Matic, BSC, и т.д.), относятся к блокчейнам состояния.
- Каждый адрес хранит в блокчейне значение своего баланса в нативной монете блокчейна (ETH, BNB, MATIC)
- Каждый смарт-контракт хранит в блокчейне значения своих персистент переменных
- На текущий момент времени (Блок X) состояние блокчейна описывается балансом всех существующих адресов в сети и текущими значениями персистент переменных всех смарт-контрактов в блокчейне.

Вся суть блокчейна отражена в его текущем состоянии, его основная задача хранить историю смены состояний, историю неизменяемую, распределенно хранимую и абсолютно прозрачную.

В блокчейне невозможно что-либо скрыть. Единственной неразрешимой задачей в рамках блокчейна является персонификация (деанонимизация) адреса конкретного кошелька. Все остальное - балансы, транзакции, все переводы, вызовы контрактов и прочее прочее прочее можно проследить, проанализировать и сделать свои какие-то выводы.

Существуют и специальные средства, которые проводят такой анализ блокчейна (ов) на предмет, например, выявления мошеннических схем или или иных интересных взаимодействий используя специализированные графовые базы данных.

Состояние блокчейна, как рассказывалось выше, это **совокупность всех текущих балансов всех существующих адресов в блокчейне плюс состояние долговременной (персистент) памяти всех смарт-контрактов, живущих (задеплоенных) в блокчейне.**

То есть другими словами состояние блокчейна - это мгновенный (в рамках текущего блока) слепок его долговременной памяти, в которой **хранится вся информация на текущий момент**.

Самое главное - это то, что **кроме блокчейна эта информация нигде не хранится**. Метамаск или трастволлет **не хранят ваши монеты**, невозможно вывести BNB с бинанса в метамаск, то, что вы сначала видите ваши BNB на балансе Бинанса, а потом в вашем

метамаске - это всего лишь отражение текущего состояния блокчейна, записанного в текущем его блоке. То есть сначала ваши средства лежали на кастодиальном счете Бинанса, а потом они транзакцией переместились на ваш единый некастодиальный адрес, который вы сами настроили в вашем метамаске.

Итак, блокчейн является единственным хранителем информации о своем состоянии, можно сказать держит мастер-копию. Кроме блокчейна эта информация нигде не хранится.

Отступление - транзакции

Итак, вся доступная на данный момент информация хранится в блокчейне и называется текущим состоянием блокчейна. Как же и при помощи чего изменяется состояние блокчейна?

Основной и единственной силой, способной изменить состояние блокчейна является транзакция. Только она способна поменять балансы на кошельках или изменить долговременную память смарт-контракта. После того как майнеры включили в следующий блок определенный набор транзакций, смайнили его и включили в цепочку - состояние блокчейна безвозвратно изменилось.

Для начала давайте вспомним что мы знаем про транзакции:

О транзакциях

- Каждая транзакция с вашего адреса (аккаунта) должна быть подписана секретным ключом этого аккаунта.
- Метамаск (да и любой другой кошелек) **ВСЕГДА** требует от пользователя подтвердить действия - подписать транзакцию, отправить транзакцию, дать доступ сайту к кошельку и т.д.
- Единственный способ совершить транзакцию без одобрения пользователь - знать секретный ключ аккаунта и сформировать подпись и транзакцию программно (например так делают все боты)

Итак, мы уже знаем что **бесхозных транзакций не бывает.** Любая транзакция в сети всегда **инициируется от адреса какого-либо кошелька.** Смарт-контракт не может инициировать транзакцию, но может (в рамках пришедшего от кошелька запроса-транзакции) **делать так называемые внутренние вызовы,** дергая методы других контрактов, отправляя монеты и тд. Но все эти действия будут выполняться **в рамках инициировавшей транзакции** и в жестких границах того газа, которое пользователь выделил на исходную транзакцию.

Отправителя (инициатора) транзакцию всегда можно п**роверить на валидность по подписи транзакции секретным ключом кошелька.** Таким образом кто знает секретный

ключ - тот и владелец. Вот почему очень важно держать в секрете вашу мнемоническую фразу и приватные ключи от кошельков.

Любая транзакция делает одну из двух вещей - либо переводит нативную монету блокчейна между кошельками (стандартная транзакция) либо вызывает какой-либо метод какого-либо смарт-контракта (смарт-контрактная транзакция). Либо делает и то и другое одновременно. **В любом случае состояние блокчейна после выполнения транзакции изменяется (может измениться)**.

Смарт-контрактная транзакция может повлечь за собой целую цепочку внутренних вызовов (внутренних транзакций), выполняемых от имени вызываемого смарт контракта. Но **инициатором всей цепочки все равно будет отправитель исходной транзакции**.

Итак, состояние блокчейна может быть изменено только при помощи корректно сформированной, подписанной и включенной в блок транзакции.

Средний уровень - смарт-контракты

Базовые понятия

- Смарт-контракт это программный код, который выполняется на нодах блокчейна, а результат выполнения (если это прописано в программе) сохраняется в блокчейне в специальном хранилище. Назовем данные, сохраняемые в блокчейне - персистент данными.
- Код смарт-контракта после заливки в блокчейн дополнительно заливается сверху слоем эпоксидки, чтобы предотвратить любое случайное или намеренное изменение кода.
- Функции смарт контракта могут быть вызваны извне (с кошелька пользователя или из другого контракта) и делятся на две большие группы:

    - Не меняющие состояние персистент данных (только чтение из блокчейна)
    - Меняющие состояние персистент данных

    Вызов функций первой группы не стоит газа и денег и не уходит дальше ближайшей ноды, к которой мы подцеплены (пример: Balance Of, TotalSupply, Allowance). *В сканерах блокчейна эти функции перечислены во вкладке “READ” контракта.*

    Вызов функций второй группы превращается в полноценную транзакцию, которая майнится, включается в блок и результат которой записывается в блокчейн. (пример: Approve, Transfer, TransferFrom). *В сканерах блокчейна эти функции перечислены во вкладке “WRITE” контракта.*

Расширенные понятия

- Поскольку код контракта залит сверху эпоксидкой, то менять его мы не можем, зато может менять состояние переменные, записанных в персистент хранилище.
- Контракт может считывать значение этих переменных и принимать какое-либо решение, основываясь на считанных значениях - исполнять или не исполнять код, провести или отбросить транзакцию, и т.д.
- Если в контракте предусмотрены внешние функции, меняющие значение таких переменных, то назовем эти функции “рычагами”, при помощи которых админ (обычно текущий владелец или ”owner” контракта) может менять поведение контракта.
- Все такие рычаги обязательно будут присутствовать на вкладке “write” контракта.
- Сам контракт ничего и никогда не инициирует. Он пассивно живет в блокчейне и ждет, когда же к нему обратятся. Все функции контракта вызываются извне либо транзакцией с обычного адреса, либо из другого контракта, но тоже только как продолжение начальной транзакции.
- Обычные транзакции к контракту видны на вкладке “transactions”, запросы же от контракта к контракту (Message calls) живут на вкладке “Internal transactions”.

Согласитесь, жить в блокчейне в котором возможны только транзакции-переводы денег между кошельками достаточно скучно и напоминает старую добрую бухгалтерию. Хотя наш дедушка BTC вполне успешно живет в этой парадигме, прекрасно себя чувствует и свысока смотрит на вот эти всякие новомодные смарт-контракты, мосты, парачейны и прочие интересные штуки.

Справедливости ради надо отметить, что в биткоине тоже есть возможность писать некие скрипты, привязанные к транзакциям, но поверьте, лучше бы этих костылей вообще не было.

Что же привнесли смарт контракты в спокойную бухгалтерскую жизнь? Ни много ни мало они **превратили блокчейн из книги бухучета в облачный дата-центр**. Каждый смарт контракт это по своей сути жестко детерминированный программный код, у которого есть адрес (место жительства в блокчейне) и ABI - описанный формат вызовов функций смарт-контракта.

Напоминаю, что залитый в блокчейн смарт контракт является неизменяемой по определению сущностью. Он будет продолжать свою работу даже если половина земли будет лежать в руинах, главное чтобы жили ноды блокчейна. Только представьте - на данный момент в блокчейне живут готовые к выполнению сотни и тысячи программ, к которым может обратиться любой желающий, если он знает их интерфейс (ABI).

Например в эфире вы можете даже сейчас обратиться к контракту криптокотиков, чтобы считать из него какие-либо данные, можете дернуть метод контракта, к которому уже пару лет никто не обращался - и, если вызов будет корректен, контракт проснется, выполнит запрос и отдаст вам результат.

Например вы можете дернуть контракт панкейк фактори и спросить у него за конкретную LP пару - существует она или нет. Если существует, то вы можете спросить у контракта уже конкретной пары за текущие резервы - получить их и рассчитать мгновенную цену.

Далее вы можете пойти на поклон к контракту панкейк рутера и запросить у него обмен ваших BNB на этот токен, с проскальзыванием, указанием необходимого минимального количества и т.д. Потом отправить ему ваши BNB и получить назад токены.

Ничего не напоминает? Ну конечно же - это классический вариант покупки токенов на панкейке через веб интерфейс. Только мы все это сделали руками, вызывая определенные функции различных контрактов.

Далее вам надо продать ваш токен там же на панкейке. По причинам, которые мы разбирали еще в первой методичке, вам надо сначала апрувнуть контракту токена адрес панкейк рутера как легитимный для снятия токенов с вашего баланса.

И снова мы идем на контракт ТОКЕНА, ищем во вкладке WRITE функцию APPROVE и вызываем ее с параметрами адреса панкейк рутера и количеством токенов, которые мы разрешаем списать с нашего баланса.

Потом идем опять же на контракт рутера и вызываем обратную покупке функцию продажи - меняем наши токены на BNB. Запускаем транзакцию, ждем - и опа, токены продались, а BNB прилетели на наш адрес.

Заметьте все это мы делаем руками, не используя веб интерфейс, либо через вкладку WRITE соответствующего контракта в эксплорере блокчейна (bscan, polygonscan, e t.c.) с подтверждением и подписью нашей транзакции в метамаске, либо вообще из скрипта на чем угодно (автор предпочитает пайтон) с подписью и отправкой через программный код не используя метамаск в принципе (так действуют практически все блокчейн боты).

Итак, смарт-контракт это программа, живущая в блокчейне, которая способна выполнять более интеллектуальную работу чем просто перевод монет с кошелька на кошелек. Вызываемые методы смарт-контракта способны менять состояние блокчейна, и могут быть вызваны как из блокчейн эксплорера, так и из программного кода на языке высокого уровня.

Самый верхний уровень - веб морда

Так что же такое и зачем нам нужен веб интерфейс?

Мы (вы) привыкли общаться с миром DeFi / DApp через различные веб морды - сайты свапалок, фармилок (похожие как две капли воды друг на друга), мостов и прочее прочее прочее. Очень удобно, все по клику, все понятно и красиво.

Для того, чтобы понимать суть того, что происходит, вам надо осознать и запомнить две простейшие вещи:

- **Любой веб интерфейс всего лишь окно визуализирующее состояние блокчейна.**
- **Любой веб интерфейс всего лишь удобная прослойка для вызова функций различных смарт контрактов.**

Все. Больше в веб мордах ничего по сути нет. Там не хранится какая-либо информация (помним, что вся инфа хранится в блокчейне), там нет ничего кроме некоего шаблона для визуализации состояния блокчейна и кода вызова определенных функций определенных контрактов при совершении пользователем тех или иных действий (нажатие на кнопку и т.д.). Причем подтверждение этих действий (подпись и отправку транзакции) пользователь должен предоставить веб морде явно нажав кнопки во всплывающем окне метамаска.

Тут таится одна опасность, которую мы уже рассматривали в третьей методичке - если вы подключились к скам сайту, нажали на кнопку “GIVEAWAY 100 BTC” и подтвердили транзакцию в метамаске, а потом с вашего счета куда-то улетели все деньги - то вы сами себе злобные буратины (ССЗБ). Т.к. скам сайт может повесить на эту кнопку абсолютно любую транзакцию (ии), то подтвердив ее вы фактически самолично расписались в тех действиях, которые привели к обнулению баланса.

Перечитайте третью методичку еще раз, повторить такие вещи всегда полезно.

Итак, любой веб интерфейс это всего лишь удобное окно для отображения состояния блокчейна и вызова различных функций смарт-контрактов для его (состояния) изменения.

Подытожим:

Итак, философско-технический взгляд на DApp и блокчейны говорит нам, что:

- Блокчейны состояний хранят всю доступную информацию в слепке своего текущего состояния, записанной в текущем блоке.

- Эта информация включает в себя остатки на балансах счетов и состояние персистент памяти всех смарт контрактов, живущих в блокчейне.
- Вся информация о состоянии хранится в блокчейне и только в блокчейне. Ни кошельки ни веб сайты, никто иной не являются ее носителями и хранителями.

  Из этого правила, кстати, легко выводятся все предыдущие, разобранные в прошлых методичках, например что балансы кошельков какого-то токена хранятся в смарт контракте этого токена, и т.д.

- Единственным методом изменения состояния блокчейна является подписанная и включенная в текущий смайненый блок транзакция.
- Транзакция может включать в себя перевод нативной монеты блокчейна с адреса на адрес и/или вызов той или иной функции какого-либо смарт контракта.
- Выполнение функции смарт контракта может привести к изменению состояния его персистент памяти, а также порождать дополнительные транзакции, которые в свою очередь также могут привести к изменению состояния.
- Любой веб интерфейс является не более чем окном для визуализации текущего состояния блокчейна, а также удобным интерфейсом для вызова различных функций различных контрактов.
- Все действия, которые вы выполняете со смарт контрактами через веб морду, также могут быть выполнены прямыми вызовами соответствующих функций смарт контрактов используя блокчейн эксплореры или скрипты на языках высокого уровня.

Разберем последний пункт более подробно. Говоря простым языком, вы можете не используя медленный и зачастую глючный веб интерфейс сделать следующее:

- Купить токен на свапалке
- Продать токен на свапалке
- Аппрувнуть контракту токена рутер свапалки для возможности продажи
- Аппрувнуть контракту монеты (BUSD, USDC) пресейл-контракт для более быстрой покупки токенов на пресейле
- Заклеймить свои токены
- И т.д. И т.п.

Все, что вам для этого надо знать - это **адрес контракта и его ABI (application binary interface)** или другими словами - **название методов контракта с их параметрами и возвращаемыми значениями.**

Для устоявшихся контрактов, таких как рутер свапалки, ЛП пара, контракт токена и т.д. ABI легко находится в гугле или же (если выложен исходник контракта) в эксплорере на вкладке “CONTRACT”.

Для кастомных контрактов, если залит исходник и ABI, то тоже вся информация лежит в эксплорере на вкладке “CONTRACT”.

А вот если исходного кода нет, то тут вроде бы печаль-тоска, и вкладок WRITE у нас нет и как к этому контракту подступиться напрямую непонятно. Но тут можно сделать две вещи:

- Декодировать бинарный код контракта (мы разбирали это на первом семинаре) и вытащить название и параметры интересующей нас функции
- Или посмотреть в эксплорере на уже выполненную кем-то (через веб, например) легитимную транзакцию с легитимным вызовом функции и корректными параметрами.

В любом случае можно для этой функции с помощью пары несложных преобразований и капельки терпения написать свой ABI, который подгрузить в скрипт и пользоваться для вызова этой функции скриптом. Через эксплорер все равно не получится, а вот через скрипт вполне.

Примечание:

Иногда, правда, без веб интерфейса не обойтись. Происходит это в том случае, если тело транзакции (ее параметры и их наполнение) формируется по каким-то своим секретным правилам в бекенде веб морды и передается в контракт в зашифрованном виде.

Если при этом контракт выложен без исходников, а его декомпиляция для реверса функции проверки представляется достаточно сложной задачей - то получаем ситуацию, когда нам нужна веб морда для получения корректного пайлоада транзакции.

Нет, саму транзакцию к методу контракта вы можете выполнить, но без этого секретно куска данных вас просто отреджектят. Пример - LZPad, который при сабмите пресейла формирует как раз такую вот хитрую транзу.

А вот, например, DXSale более простой, там можно спокойно написать бота для быстрой покупки, KoalaPad тоже без особых изысков, главное найти контракт пресейла, ну и т.д.

Приложение 1. “Я сам себе ~~сама~~ DApp”

Как уже писалось выше, взаимодействовать с контрактом напрямую можно либо через библиотеки к высокоуровневым языкам программирования (Pyhon, JavaScript, …), либо через интерфейс блокчейн эксплорера. Первый способ подходит для написания ботов и иных скриптов для которых требуется низкоуровневый доступ к контрактам, второй - для быстрых действий с контрактами, когда веб интерфейса либо недостаточно либо по каким-то причинам не применимо в данной ситуации.

Для работы с контрактом через блокчейн эксплорер необходимо следующее:

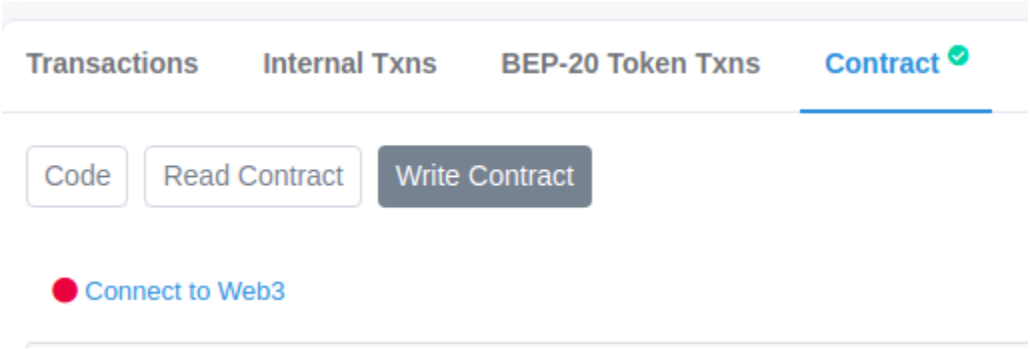
- Адрес контрата в блокчейне
- Открытость контракта (есть исходный код)
- ABI к контракту существует и загружен

Ну и, естественно, наш метамаск подключеный к нужному блокчейну.

В данной методичке мы будем рассматривать примеры контрактов на BSC, но вы же помните из 1-й методички, что эти знания можно также применить к любому Ethereum-based блокчейну - сам Ethereum, Polygon, Fantom, Avalanche, и т.д.

Подключение к к контракту

Заходите во вкладку Контракт, Write и нажимаете на “connect to Web3”. Подтверждаете в метамаске и ждете когда красный сигнал сменится на зеленый. Все, вы подключены.



Методы (функции) контракта.

Внешние вызовы функций контракта, доступные через ABI интерфейс принято называть **методами**.

Что делает каждый из методов можно узнать либо из спецификации, если контракт реализует данную спецификацию (ERC20, ERC721, и т.д.), либо из исходного кода (если он есть), ну или руководствуясь здравым смыслом по названию метода и его параметрам.

Немного о параметрах, у каждого параметра есть тип, которому обязано соответствовать то значение, которое вы в него запишете. Типы соответствуют типам Solidity (мы рассматривали основы Solidity во 2-й методичке) и не должны вызывать у вас оторопи и непоняток. На всякий случай перечислим еще раз с пометками и пояснениями:

- `string` - строка

- uintXXX - беззнаковое целое шириной XXX в битах (обычно uint256)
- address - адрес в блокчейне (кошелька или контракта)Ex: 0xa45….9d
- address[] - массив адресов (несколько значений, через запятую)

Методы на вкладке READ

Все методы перечисленные на вкладке READ для любого контракта вы можете вызывать без подключения к нему, в любое время и без страха - любой такой вызов не требует комиссий, т.к. не порождает транзакцию, а просто не уходит дальше той RPC ноды, к которой вы подключены.

Также такой вызов не может навредить ни вам ни блокчейну, поскольку только считывает данные о его (блокчейна) текущем состоянии.

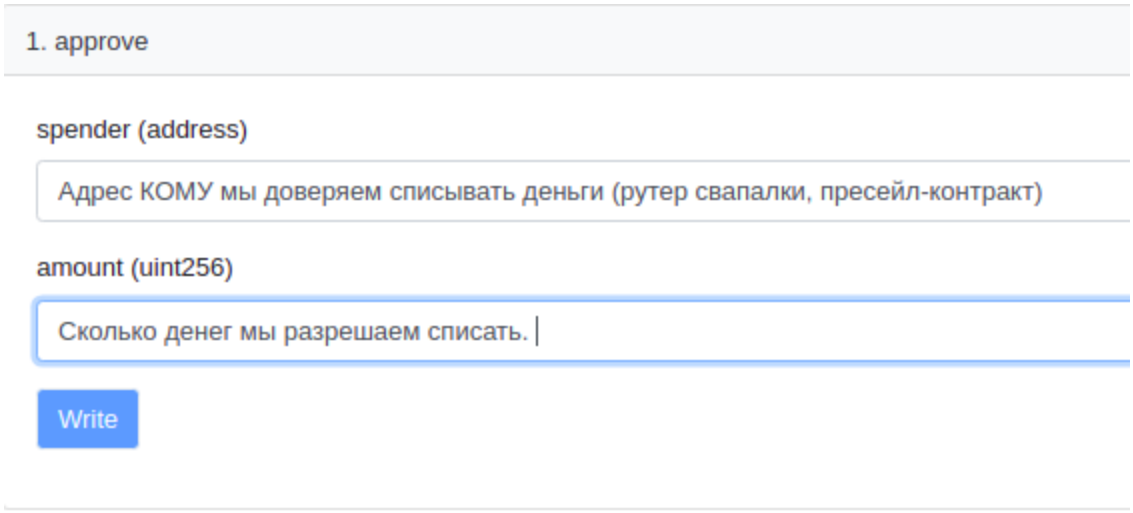
Методы на вкладке WRITE

Методы на вкладке WRITE - это уже полноценные методы взаимодействия с контрактом, которые порождают транзакции и могут привести к изменению состояния блокчейна.

Небольшое замечание - т.к. 90% кода всех базовых контрактов свапалок и прочей инфраструктуры в Ethereum-based блокчейнах было честно ~~спизж~~ позаимствовано из эфира, то не пугайтесь аббревиатурам ETH в контрактах, которые вам встретятся. Считайте, что ETH - это аббревиатура нативной монеты блокчейна (BNB, Matic, AVAX, ...)

Давайте вкратце разберем несколько основных методов базовых контрактов, которые могут быть интересны в повседневной жизни.

=== Token Contract



Если хотим поставить безлимит, то используем константу:

`115792089237316195423570985008687907853269984665640564039457584007913129639935`

Если хотим отозвать аппрув, то ставим amount = 0

=== *SwapRouter Contract*

`addLiquidity/removeLiquidity` - добавить или удалить ликвидность. Можно увидеть в куче монет с автоликвидностью, параметры прозрачны, в инете куча инфы плюс можете посмотреть в исходниках тех самых токенов.

И далее огромный список вариантов как можно купить или продать что-то на свапалке.

9. swapETHForExactTokens

10. swapExactETHForTokens

11. swapExactETHForTokensSupportingFeeOnTransferTokens

12. swapExactTokensForETH

13. swapExactTokensForETHSupportingFeeOnTransferTokens

14. swapExactTokensForTokens

15. swapExactTokensForTokensSupportingFeeOnTransferTokens

16. swapTokensForExactETH

17. swapTokensForExactTokens

Много, страшно, но все абсолютно детерминировано) Поехали расшифровывать:

Короткий анекдот:

Поезд, ночь, станция, фонари. Просыпается мужичок и заполошно спрашивает:
- *Простите, а где стоим?*
- *В Одессе*
- *А почему стоим?*
- *Паровоз меняем*
- *Простите, а на что?*
- *На паровоз*
- *… паровоз на паровоз? Не, это точно не Одесса...*

Итак, поехали:

`swap` - обмен **того, что есть у нас**
`(Exact)` - опционально. Точное количество того, что идет следующим
`[ETH/Tokens]` - что есть **у нас**, т.е. В интерфейсе свапалки верхнее окно

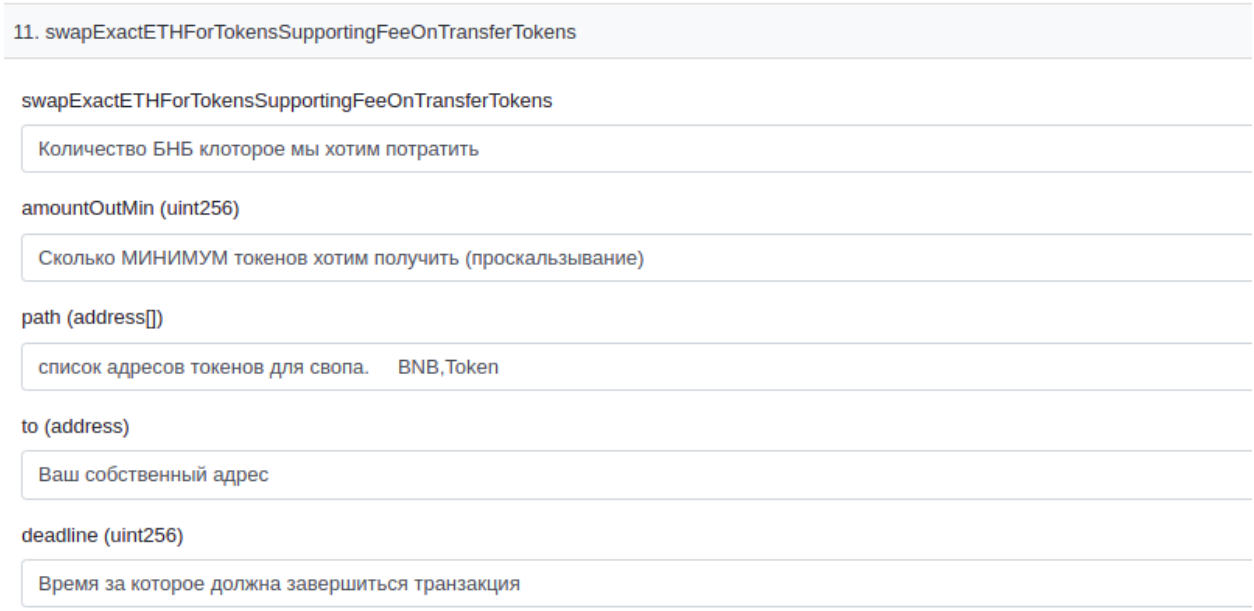
`for` - на что меняем то, что **хотим получить** (см. выше)
`(Exact)` - опционально. Точное количество того, что идет следующим

`[ETH/Tokens]` - что **хотим получить**, т.е. В интерфейсе свапалки нижнее окно

`(SupportingFeeOnTransferTokens)` - должно использоваться с токенами, которые берут комиссию с трансфера. По моему опыту можно использовать всегда. Но это не точно)

Параметры методов (практически идентичные)

Помним, что ETH - это аббревиатура базовой монеты блокчейна.



`amountOutMin` - если не хотим париться с вычислением слипа ставим 0

`path` - путь обмена, в минимальном случае - два адреса **что куда меняем**, например если я хочу купить BABYBANANA за BNB, то должен записать в это поле:

`0xbb4CdB9CBd36B01bD1cBaEBF2De08d9173bc095c, 0x093f07Ff9e7a897e02F133f1Aa77f6458dCFCf47`

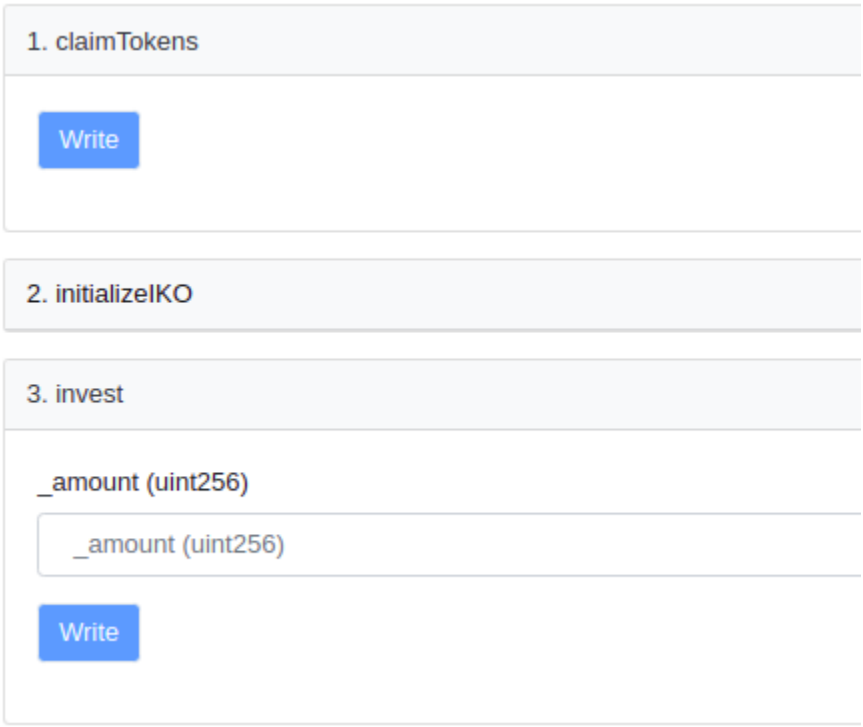
Адрес WBNB Адрес BBNANA

БЕЗ КАВЫЧЕК, СКОБОЧЕК ИЛИ ЕЩЕ ЧЕГО - ДВА ЗНАЧЕНИЯ ЧЕРЕЗ ЗАПЯТУЮ !!!

`deadline` - Проще всего сделать так: идете на https://www.unixtimestamp.com/, внизу добавляете 1 час, нажимаете “Convert” и берете значение из поля “**Unix Timestamp**” внизу в таблице.

=== *Presale Contract*

KoalaDefi IKO
(https://bscscan.com/address/0xa5707412E6F3e06e932b139C3BCAD26a0734Ab91)



`Invest` - позволяет вам засабмиттить сумму в **BUSD WEI** не через веб, а напрямую из контракта.

`claimTokens` - клеймит ваши токены, когда придет время раздачи

DxSale
(https://bscscan.com/address/0x52734dd84c890bc169b239d9e60e574a5cadda3c#code)

Как вы можете видеть он без исходников, но 1) его можно декомпилировать и 2) найти в интернете предыдущую декомпилированную версию с ABI.

После этого вы не сможете вызывать методы контракта через веб эксплорер, но легко и непринужденно сделаете это через JavasScript web3 / Python web3 / …

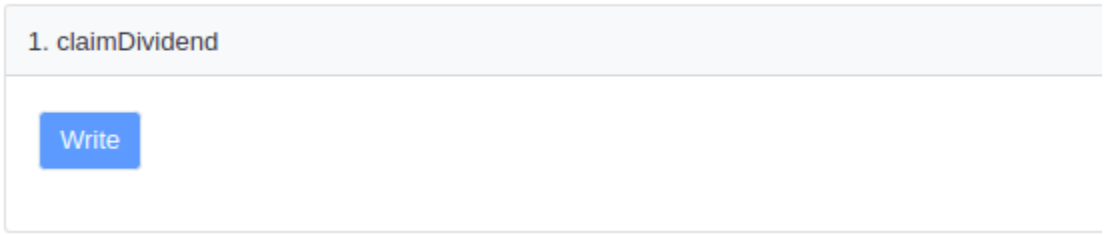
```
def claimTokens()
def buyTokens(address _beneficiary)
```

=== Some Tokens

Ну и научившись работать с контрактами напрямую вы всегда сможете сами сделать некоторые вещи, не дожидаясь стандартного времени, условий и т.д.

Например для ревордных токенов вы можете заклеймить свои реворды через вызов методов:

BBNANA DIVTRACKER:
(https://bscscan.com/address/0x88F8AC6e8fF48291ce26dcAA514Fd482Cbf2823d)



BJOE:
(https://cchain.explorer.avax.network/address/0xe703214808C3603a9dA2609d52bDcF920E0a0edf)

Заключение

В этой очень краткой методичке мы рассмотрели основополагающие вещи, касающиеся того где хранится информация в блокчейнах состояний, что из себя на самом деле представляет веб приложение, а также разобрались как можно работать со смарт контрактом напрямую, эмулируя DApp, не используя ничего кроме блокчейн эксплорера.

Надеюсь, что она дала Вам немного новой и полезной информации)))

Всем профита!

До встречи,
Alex Kruegger (@Kruegger)

****

Если данный материал был Вам полезен и Вы решили отблагодарить автора и мотивировать его на дальнейшую работу в этом направлении, не держите это желание в себе, а шлите лучи поддержки на:

BTC: bc1qsg8566ys77xqq77m3zd32utrj9f8h34rqcp9cx
BSC/ETH/Polygon: 0x909b194faa37574a2927525536d4dcae2a925a8e
Wave: 3PHdoa9JLbDRDjVZdXJUSKHCwXA8RTVo2zG
Solana: 5dV3usDd4ddzqmLVSeeifNhqBkAw67Yzm7jDEgTjv61U
Polkadot: 13Y9D88uGesuwUUNMLiv2b2tNhsqXA2iV9YJuuXYgNq9te8J
KARDIA-CHAIN: 0x3b11e52b748916106b27ae15f5f821bd47481f8f

****