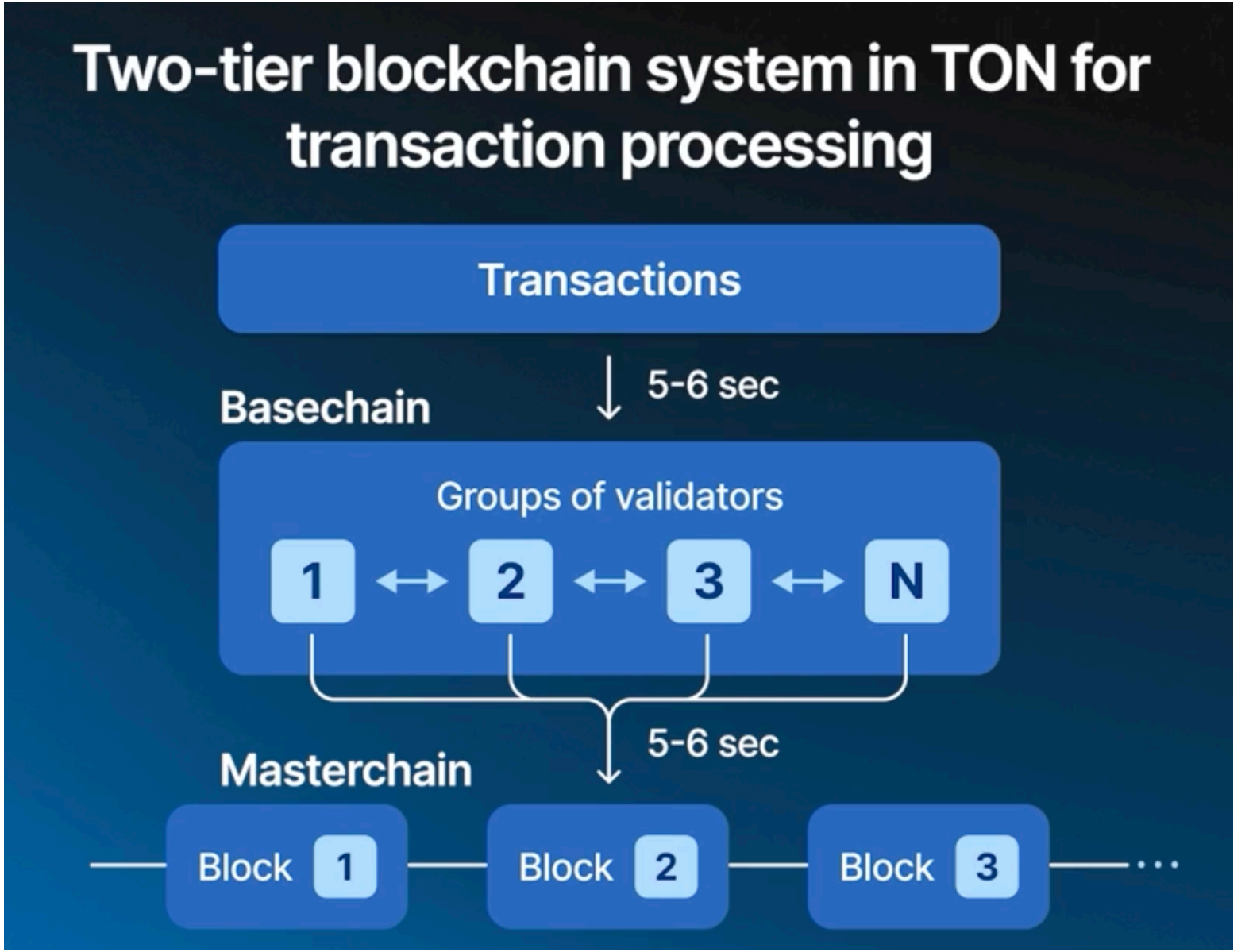
Kenetic

TON 智能合约开发实践

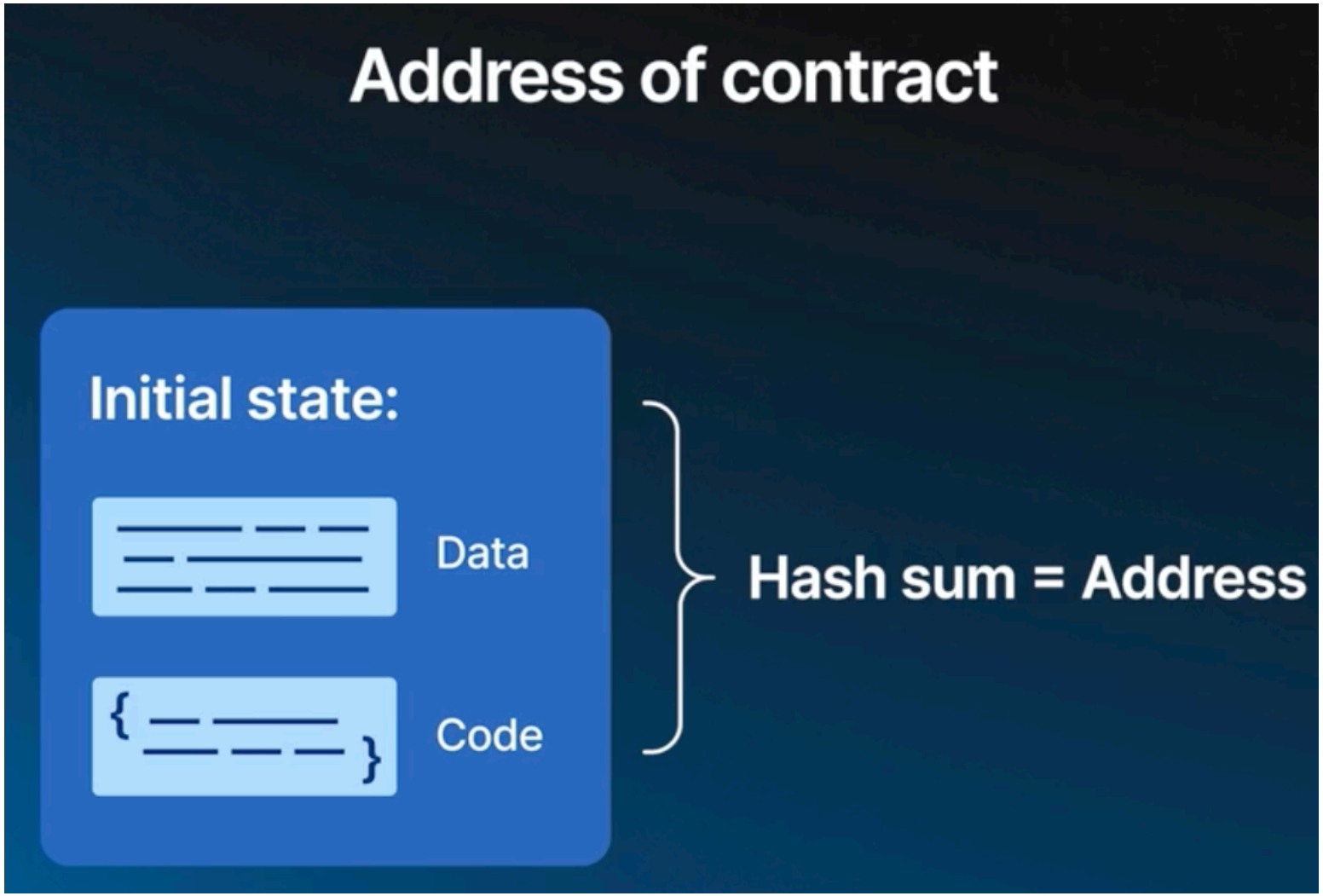
Speaker: Kojh Liang (Head of Research at Kenetic Capital)

1.TON 区块链基础知识(主链和工作链)

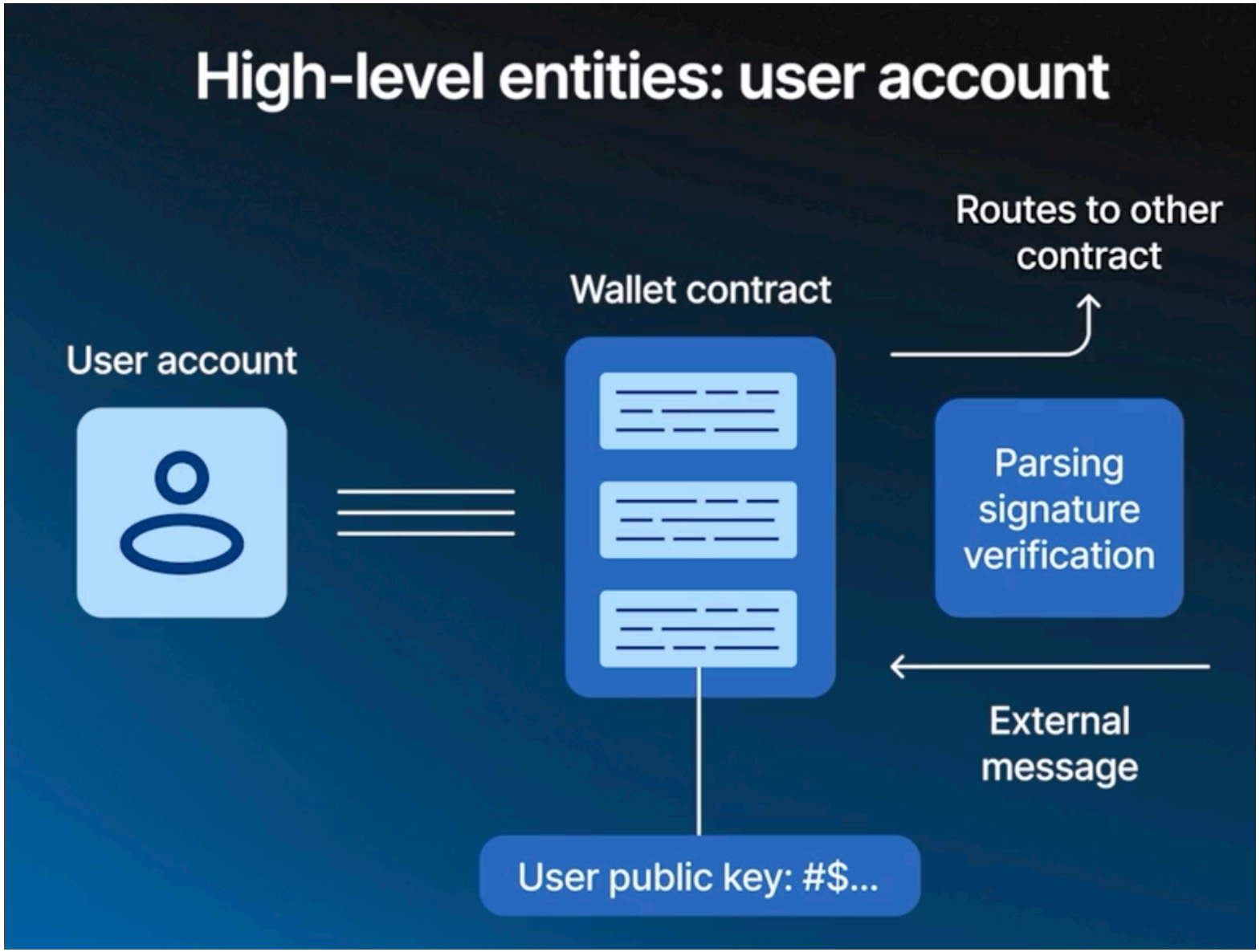


Source: https://stepik.org/lesson/1011444

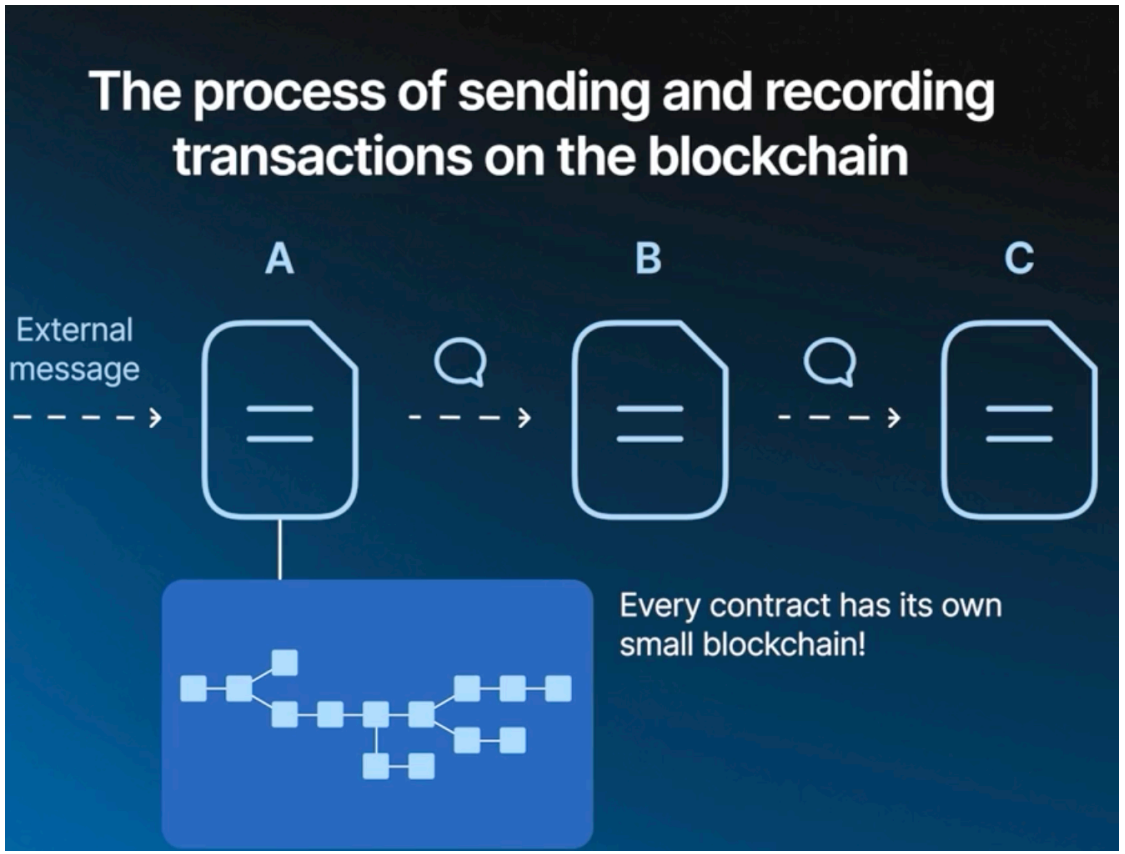
1.TON 区块链基础知识(合约地址)

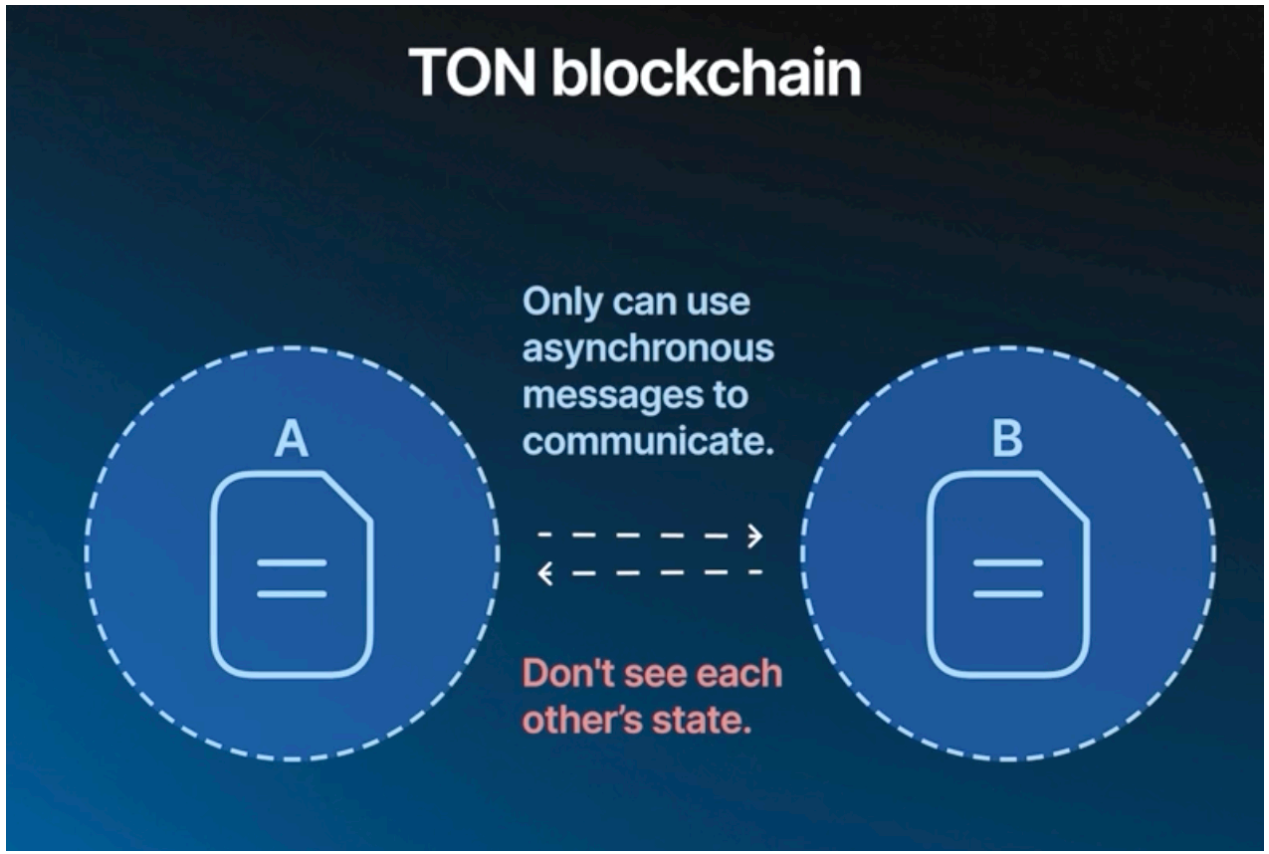


1.TON 区块链基础知识（用户钱包账户也是合约）

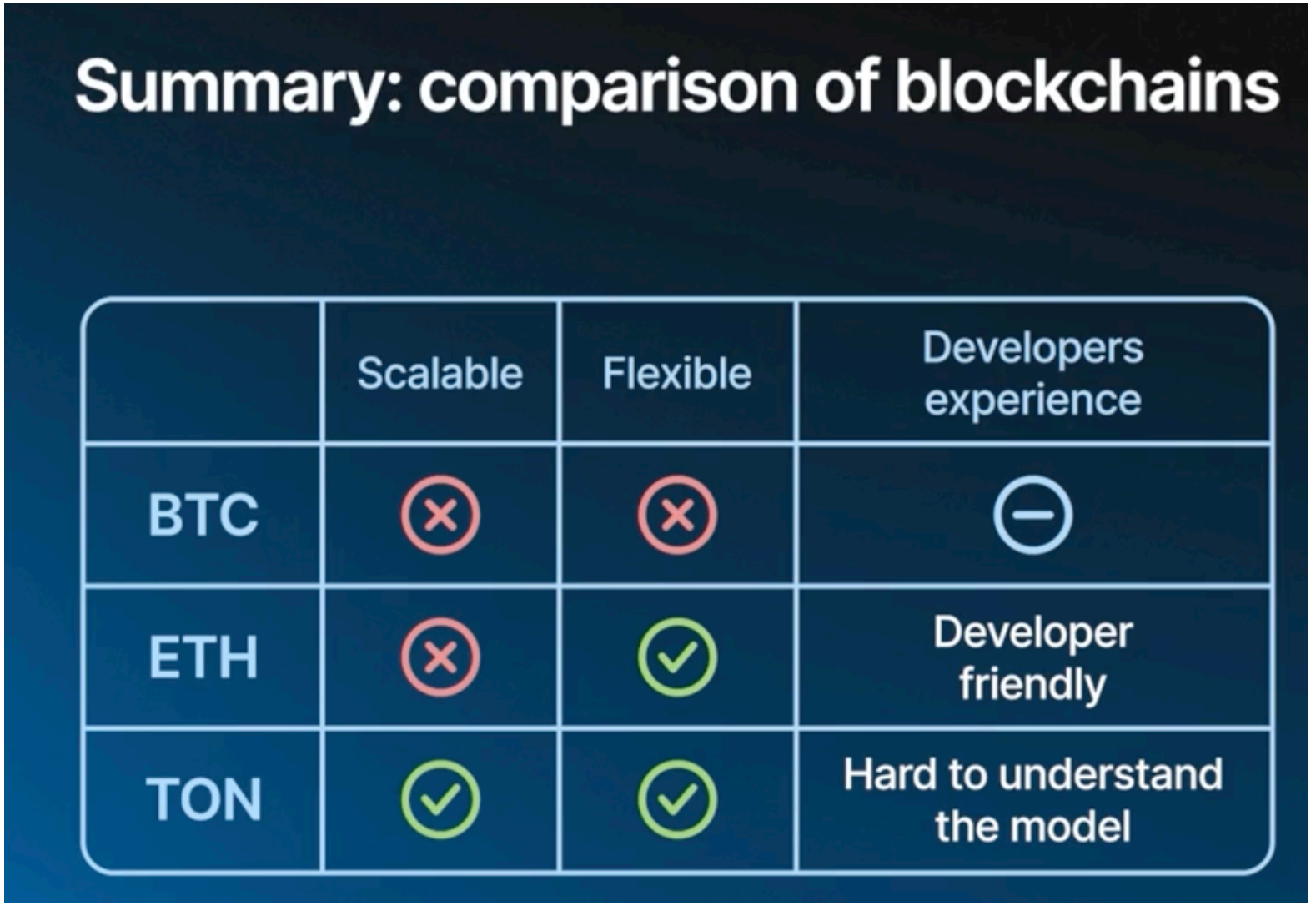


1.TON 区块链基础知识（异步消息机制）

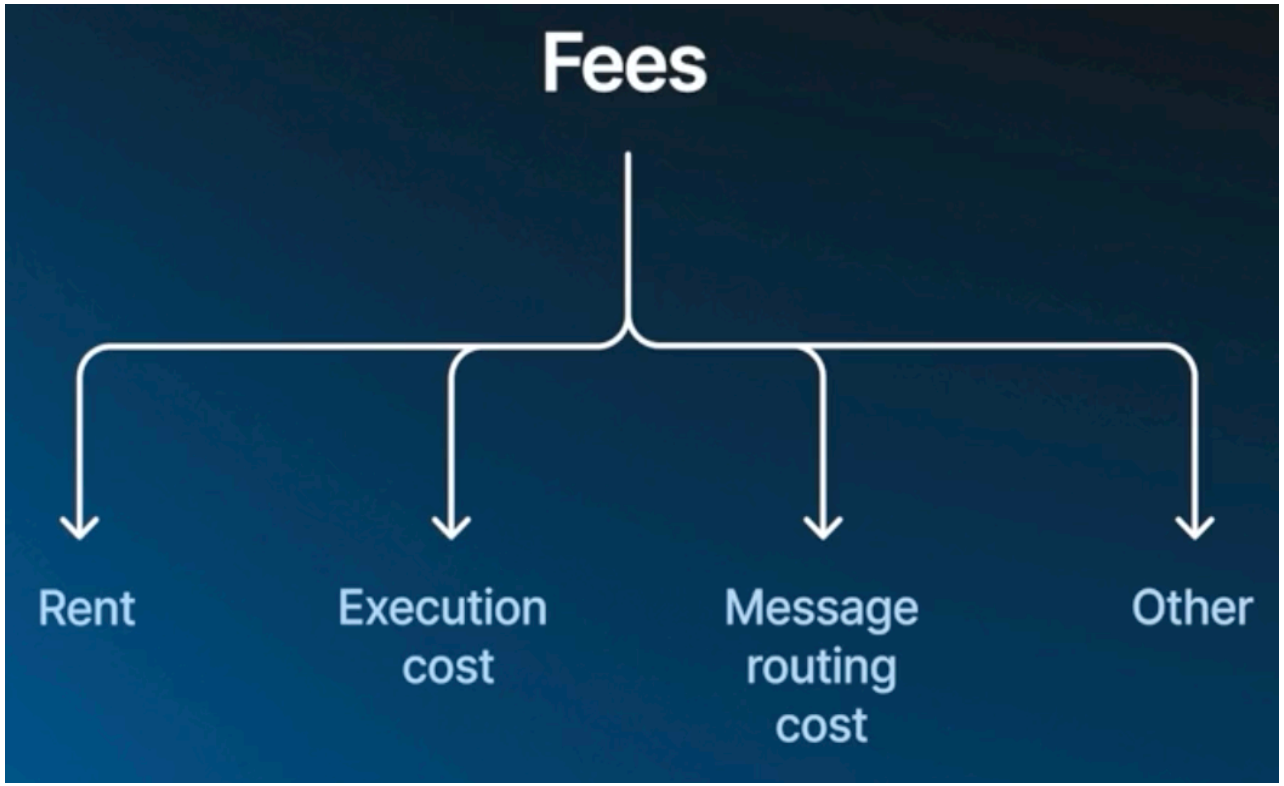




1.TON 区块链基础知识（ BTC，ETH和TON对比）



2.TON智能合约：费用基础知识



GAS

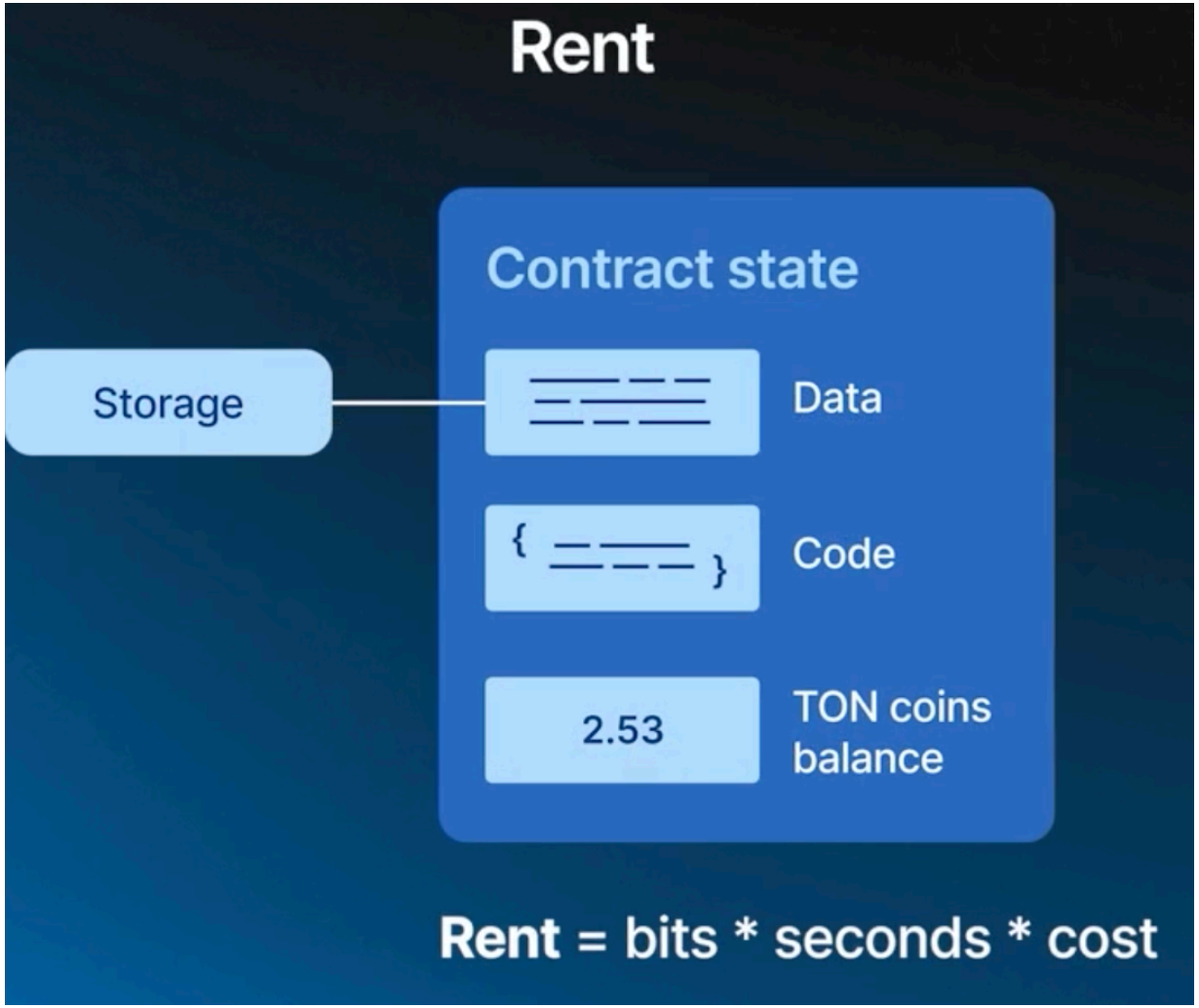
Ethereum (solidity):

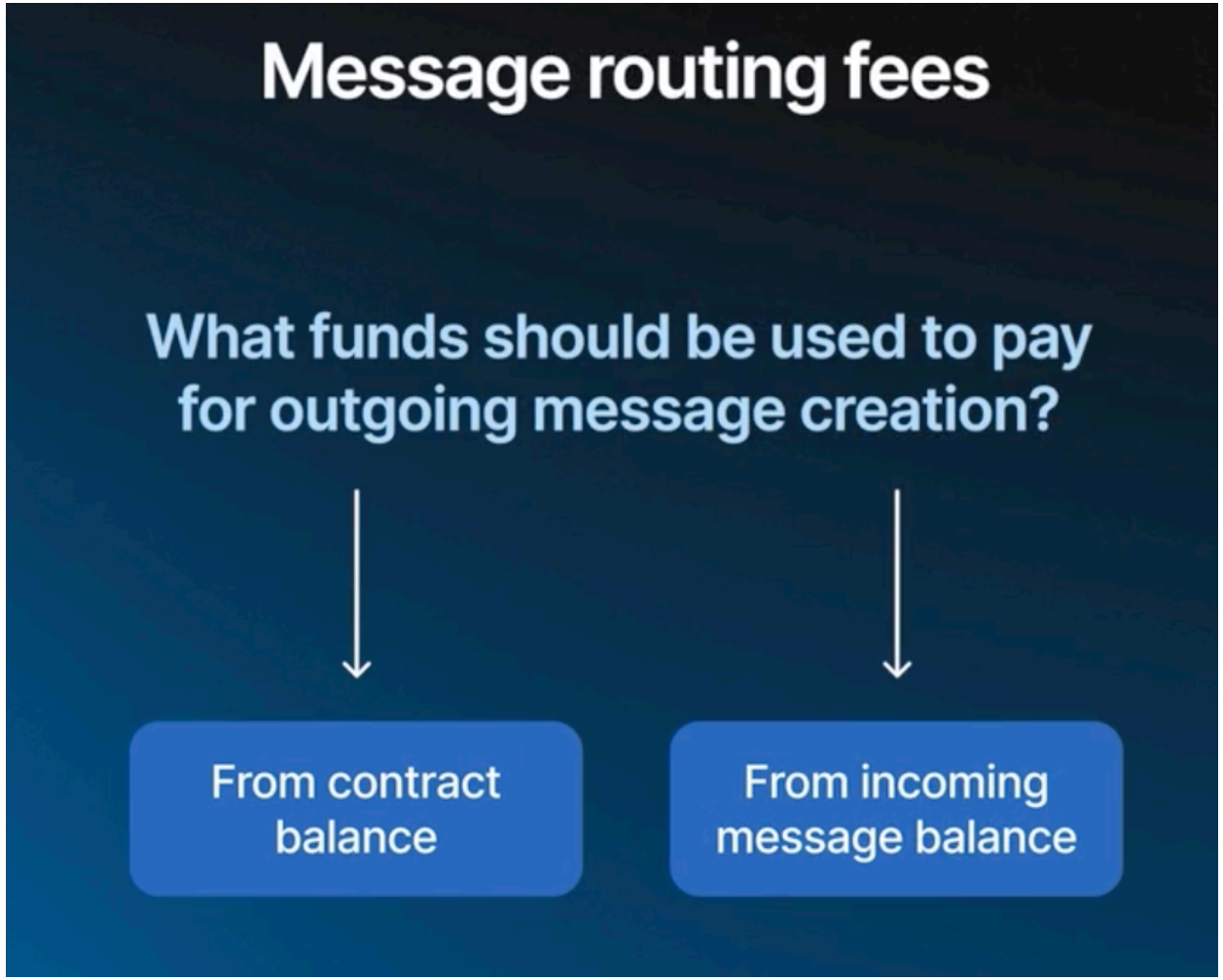
If the user provides too little gas, everything will be reverted as if nothing had happened.

TON:

- If there is not enough gas, the transaction will be partially executed.
- If there is too much gas, the excess must be returned. This is the developer's responsibility.
- TON can't do everything itself because of asynchronous nature.

2.TON智能合约：费用基础知识





3.TON 智能合约开发实践：编程语言的选择

1.Tact（ https://tact-lang.org）

优点：类似面向对象编程，语法和高级语言类似，编写较为简单

缺点：由社区贡献，目前版本未足够稳定，目前不支持一些TVM的高级特性，
编译出的FunC代码需要的Gas比直接使用FunC编写的代码更高

2.FunC（ https://docs.ton.org/develop/func/overview ）

优点：官方支持，有大量已上线的FunC应用模版可参考

缺点：上手难度高且不易懂，类似C语言编程

3.Fift（ https://docs.ton.org/develop/fift/overview ）

接近汇编语言，开发者一般无需使用此语言编写合约

Language	Functionality	Learning Curve	Tutorial	Library	Gas Optimization	Safety
TVM Assembly						
Fift	☆☆☆	☆☆☆	☆	☆☆	☆☆☆	☆☆☆
FunC	☆☆☆	☆☆	☆☆☆	☆☆☆	☆☆	☆☆☆
Tact	☆	☆	☆	☆	☆	☆

Tact -> FunC -> Fift -> Bytecode -> Machine code

3.TON 智能合约开发实践：异常处理和错误码

TVM常见默认错误码

错误码	描述
13，-14，37	gas费用不足
7	变量类型错误
8	写入Cell数据溢出
9	尝试读取超过Cell长度的数据
4，5	整数溢出或除0

TVM所有的默认错误码：
https://docs.ton.org/mandarin/learn/tvm-instructions/tvm-exit-codes

自定义错误码

```
;;maker sure message from  owner
throw_unless(405, equal_slices(owner_address, sender_address));
```

注意的要点：

1.即使合约发生错误而返回亦会消耗gas

2.合约发生错误时只会返回错误码和256位数据给**上一个调用的合约**，不会返回给最初的交易发送者。例子：合约A->合约B->合约C,如果合约C发生错误并且B调用合约C时设置消息为可弹回，则合约C将返回错误代码给合约B。

3.因为合约发生错误时只能返回很少的数据给上一个合约，因此很难实现复杂合约链条的层层错误返回。此时，合约的逻辑应直接根据错误情况发送相应的处理消息给对应的接收方。

3.TON 智能合约开发实践：发送消息的模式

Six Message modes

```
;;Carry all remaining contract balances
int sendMode::CARRY_ALL_REMAINING_BALANCE()  asm "128 PUSHINT";

;;Carry all remaining msg_value of income message
int sendMode::CARRY_ALL_REMAINING_INCOMING_VALUE()  asm "64 PUSHINT";

int sendMode::DESTROY_ACCOUNT_IF_ZERO()  asm "32 PUSHINT";

;; Pay outgoing message fee from contract balance
int sendMode::PAY_GAS_SEPARATELY()  asm "1 PUSHINT";

int sendMode::IGNORE_ERRORS()  asm "2 PUSHINT";

;; Pay outgoing message fee from msg_value of income message
int sendMode::NONE()  asm "0 PUSHINT";
```

https://docs.ton.org/develop/smart-contracts/messages

1.无任何分支后的交易无需考虑Gas fee问题，发送消息时可使用CARRY_ALL_REMAINING_INCOMING_VALUE或CARRY_ALL_REMAINING_BALANCE
模式发送给下一个合约,附带的msg_value为0即可

2.有分支的交易必须考虑Gas fee问题，发送消息时可使用NONE或PAY_GAS_SEPARATELY模式发送给下一个合约，必须附带具体的、足够支撑后续所有流程执行的数额的msg_value

3.TON 智能合约开发实践：合约gas fee的计算

遇到多分支的交易需要计算具体某个合约的gas fee时，有两种方法可以获得：

方法1: 使用Tonviewer（https://tonviewer.com）

1.Total fee = Gas fee + Storage fee + Action fee

2.Forward fee = Action fee + （Forward fee 1 +…+ Forward fee N)

testnet.tonviewer.com/transaction/25cbdf494c43a54fdfd943da4498cc648126f0c12c6309aa0d0fc7d7b2c4f602

Internal message

Source	A 0QAAHhCq…JBKhCn…	Created at:	11.04.2024, 19:50:12	Bounced:	false	Forward Fee:	0.000666672 TON
Value:	0.520000000 TON	Created lt:	20685255000002	Bounce:	true	Init:	-
IHR disabled:	true						
OpCode:	0x68b4959e						Copy Raw body

Account: B 0QAUqd4c…BXSgYB… Interfaces: - ton.cx · toncoin.org

Transaction

Block ID:	(0,20000…9763153)	Time:	11.04.2024, 19:50:12	Compute Phase		Action Phase	
Tx hash:	25cbdf49…b2c4f602	Lt:	20685255000003	Success:	true	Success:	true
Prev. tx hash:	157b2ee3…8bae229b	Prev. tx lt:	20680953000001	Exit code:	0	Result code:	0
Total fee:	0.008606650 TON	Status:	active → nonexist	Vm steps:	153	Total actions:	2
Fwd. fee:	0.003026000 TON	State hash:	3e…df → 90…a4	Gas used:	7598	Skipped actions:	0
Gas fee:	0.007598000 TON	Aborted:	false				
Storage fee:	0.000000000 TON	Destroyed:	true				
Action fee:	0.001008650 TON	Type tx:	TransOrd				

Total fee = Gas fee + Storage fee + Action fee

Fwd. fee = Action fee + (Forward fee 1 +…+ Foward fee N)

Internal message

Destination:	C kQCGdsda…K8QRBJ…	Created at:	11.04.2024, 19:50:12	Bounced:	false	Forward Fee:	0.001220010 TON
Value:	0.488170000 TON	Created lt:	20685255000004	Bounce:	true	Init:	-
IHR disabled:	true						
OpCode:	0x69072dbb						Copy Raw body

Internal message

Destination:	F 0QC2vEae…8GooP5…	Created at:	11.04.2024, 19:50:12	Bounced:	false	Forward Fee:	0.000797340 TON
Value:	0.021206000 TON	Created lt:	20685255000005	Bounce:	true	Init:	-
IHR disabled:	true						
OpCode:	Excess · 0xd53276db						Copy Raw body

```
query_id: 4013967854555713000
```

方式一：

扣减当前合约需要消耗的gas，剩下再分配分支的Gas。

Output_msg_value = Incoming_msg_value – Total_gas

方式二：

提前精确定义每条分支需要的Gas fee。

Output_msg_value =. specific_branch_gas

3.TON 智能合约开发实践：合约gas fee的计算

遇到多分支的交易需要计算具体某个合约的gas fee时，有两种方法可以获得：

方法2：使用 Sandbox的printTransactionFees （安装： npm install @ton/sandbox）

console.log

(index)	op	valueIn	valueOut	totalFees	inForwardFee	outForwardFee	outActions	computeFee	exitCode	actionCode
0	'N/A'	'N/A'	'5 TON'	'0.003911 TON'	'N/A'	'0.001 TON'	1	'0.001937 TON'	0	0
1	'no body'	'5 TON'	'0.593635 TON'	'0.021111 TON'	'0.000667 TON'	'0.031365 TON'	1	'0.010656 TON'	0	0
2	'0x178d4519'	'0.593635 TON'	'0.573497 TON'	'0.016986 TON'	'0.020911 TON'	'0.003377 TON'	2	'0.01586 TON'	0	0
3	'0x7362d09c'	'0.001 TON'	'0 TON'	'0.000309 TON'	'0.001027 TON'	'N/A'	0	'0.000309 TON'	0	0
4	'0xdd9911fa'	'0.572497 TON'	'0.476437 TON'	'0.053199 TON'	'0.001226 TON'	'0.05866 TON'	5	'0.033645 TON'	0	0
5	'0x78ccd28'	'0.095288 TON'	'0.07635 TON'	'0.012702 TON'	'0.007822 TON'	'0.003138 TON'	2	'0.011656 TON'	0	0
6	'0x78ccd28'	'0.095288 TON'	'0.07635 TON'	'0.012702 TON'	'0.007822 TON'	'0.003138 TON'	2	'0.011656 TON'	0	0
7	'0x78ccd28'	'0.095288 TON'	'0.07635 TON'	'0.012702 TON'	'0.007822 TON'	'0.003138 TON'	2	'0.011656 TON'	0	0
8	'0x78ccd28'	'0.095288 TON'	'0.07635 TON'	'0.012702 TON'	'0.007822 TON'	'0.003138 TON'	2	'0.011656 TON'	0	0
9	'0x78ccd28'	'0.095288 TON'	'0.07635 TON'	'0.012702 TON'	'0.007822 TON'	'0.003138 TON'	2	'0.011656 TON'	0	0
10	'0x0'	'0.001692 TON'	'0 TON'	'0.000309 TON'	'0.000873 TON'	'N/A'	0	'0.000309 TON'	0	0
11	'0x8a7827a7'	'0.074658 TON'	'0.054862 TON'	'0.015609 TON'	'0.001221 TON'	'0.001196 TON'	1	'0.01521 TON'	0	0
12	'0x0'	'0.001692 TON'	'0 TON'	'0.000309 TON'	'0.000873 TON'	'N/A'	0	'0.000309 TON'	0	0
13	'0x8a7827a7'	'0.074658 TON'	'0.054862 TON'	'0.016717 TON'	'0.001221 TON'	'0.001196 TON'	1	'0.016318 TON'	0	0
14	'0x0'	'0.001692 TON'	'0 TON'	'0.000309 TON'	'0.000873 TON'	'N/A'	0	'0.000309 TON'	0	0
15	'0x8a7827a7'	'0.074658 TON'	'0.054862 TON'	'0.016717 TON'	'0.001221 TON'	'0.001196 TON'	1	'0.016318 TON'	0	0
16	'0x0'	'0.001692 TON'	'0 TON'	'0.000309 TON'	'0.000873 TON'	'N/A'	0	'0.000309 TON'	0	0
17	'0x8a7827a7'	'0.074658 TON'	'0.054862 TON'	'0.017317 TON'	'0.001221 TON'	'0.001196 TON'	1	'0.016918 TON'	0	0
18	'0x0'	'0.001692 TON'	'0 TON'	'0.000309 TON'	'0.000873 TON'	'N/A'	0	'0.000309 TON'	0	0
19	'0x8a7827a7'	'0.074658 TON'	'0.054862 TON'	'0.016717 TON'	'0.001221 TON'	'0.001196 TON'	1	'0.016318 TON'	0	0
20	'0xd53276db'	'0.054862 TON'	'0 TON'	'0.000309 TON'	'0.000798 TON'	'N/A'	0	'0.000309 TON'	0	0
21	'0xd53276db'	'0.054862 TON'	'0 TON'	'0.000309 TON'	'0.000798 TON'	'N/A'	0	'0.000309 TON'	0	0
22	'0xd53276db'	'0.054862 TON'	'0 TON'	'0.000309 TON'	'0.000798 TON'	'N/A'	0	'0.000309 TON'	0	0
23	'0xd53276db'	'0.054862 TON'	'0 TON'	'0.000309 TON'	'0.000798 TON'	'N/A'	0	'0.000309 TON'	0	0
24	'0xd53276db'	'0.054862 TON'	'0 TON'	'0.000309 TON'	'0.000798 TON'	'N/A'	0	'0.000309 TON'	0	0

at printTransactionFees (node_modules/@ton/sandbox/dist/utils/printTransactionFees.js:34:13)

费用的详细介绍和计算：https://docs.ton.org/develop/smart-contracts/fees

3.TON 智能合约开发实践：合约gas fee的限制

单个合约可以消耗的最大Gas fee为 1 TON，如果超过将触发合约异常，因此要注意在单个合约消耗的Gas fee不可以过高。

要注意可能会触发执行合约时Gas fee过高的点：

1.在合约中For循环的次数过大，导致消耗过高的Gas fee。
（解决方案：必须确保For 循环的次数一定在有限范围）

2.在合约中使用了Dictionary或Tuple等数据结构，且元素个数过大，导致查找一次元素时需要消耗极高的Gas fee。
（解决方案：如果要使用Dictionary或Tuple等数据结构，确保元素的数量在小量的有限范围）

3.合约中需要同时发送多个消息到其他合约，因为每次发送消息都需要消耗Gas，如果发送消息的数量过大，有可能达到Gas fee的上限。
（解决方案：必须确保在合约中发送的消息数量在有限范围）

3.TON 智能合约开发实践：合约竞争条件问题的识别

合约竞争条件问题（race condition）的产生：因为交易里的所有合约的执行都是异步执行，交易里的某个合约执行完后，传给下个合约的状态，可能已被其他交易更改了状态，也就是获得的状态不是最新的。（特别地，如果一个原子性的操作被分割在两个或多个不同的合约时，极易产生合约竞争条件问题）

例子：有两个合约walletA和walletB，walletA给walletB转账5 token时的处理流程为：

检查A余额是否足够，如果足够，则发转账消息给B

B收到A的消息，Token +5，发消息给A，通知A的token可以-5

A收到B的通知消息，A的token -5

上述例子会因并发执行而产生竞争条件问题。因为如果A当前token为5，A实际可以同时发起两笔交易给B转账5个token，B将会收到10个token。从而出现类似双花的问题。

识别潜在的竞争条件：列出交易的所有流程步骤，以及每个步骤的状态更改，建立一个二维表格，通过观察每个步骤的状态的交集，分析并发状态下是否可能存在竞争条件问题。

3.TON 智能合约开发实践：合约竞争条件问题的识别

例子：

FT2(A transfer FT to B)		Step 1(Sender wallet)	Step 2.1(collection)	Step 2.2(NFT Item)	Step2.3(Sender wallet)	Step3.1(Receiver wallet)
Involved data		jetton_balance, owned_nft_number, owned_nft_dict	current_total_supply	contract active status	owned_nft_dict	jetton_balance
FT1(A transfer FT to B)	Involved data					
Step 1(Sender wallet)	jetton_balance, owned_nft_number, owned_nft_dict	jetton_balance owned_nft_number owned_nft_dict			owned_nft_dict	
Step 2.1(collection)	current_total_supply		current_total_supply			
Step 2.2(NFT Item)	contract active status			contract active status:non-active		
Step 2.3(Sender wallet)	owned_nft_dict	owned_nft_dict			owned_nft_dict	
Step 3.1(Receiver wallet)	jetton_balance					jetton_balance

Thanks!

TRC-404 Group
(https://t.me/trc404bot)

@ton_trc404

TRC-404 MiniApp
(https://t.me/trc404chat)