LNP/BP Standards Association

RGB v0.10 release: part 2. Standard lib

Dr Maxim Orlovsky
Chief engineering officer at **LNP/BP Standards Association**,

@dr_orlovsky

FBDE A843 3DDC 1E69 FA90 C35E FFC0 2509 47E5 C6F7

Today agenda:

- Overview of all new features coming in RGB v0.10
- Anatomy of RGB smart contracts
- Contract interfaces and why they are needed
- Writing custom RGB contract and issuing an asset
  - Wallet developer perspective
  - End-user perspective

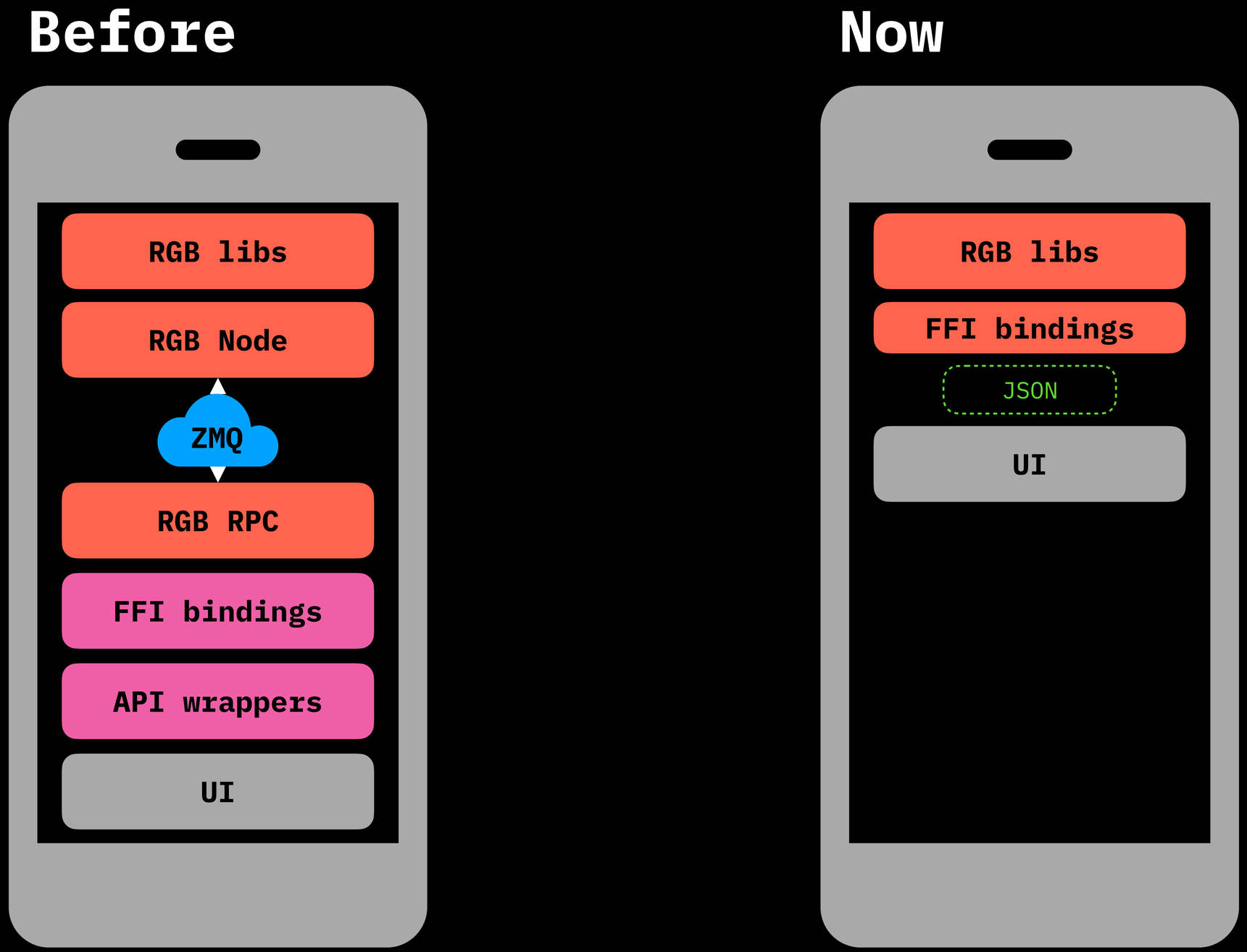
RGB evolution with v0.10 release

Consensus level

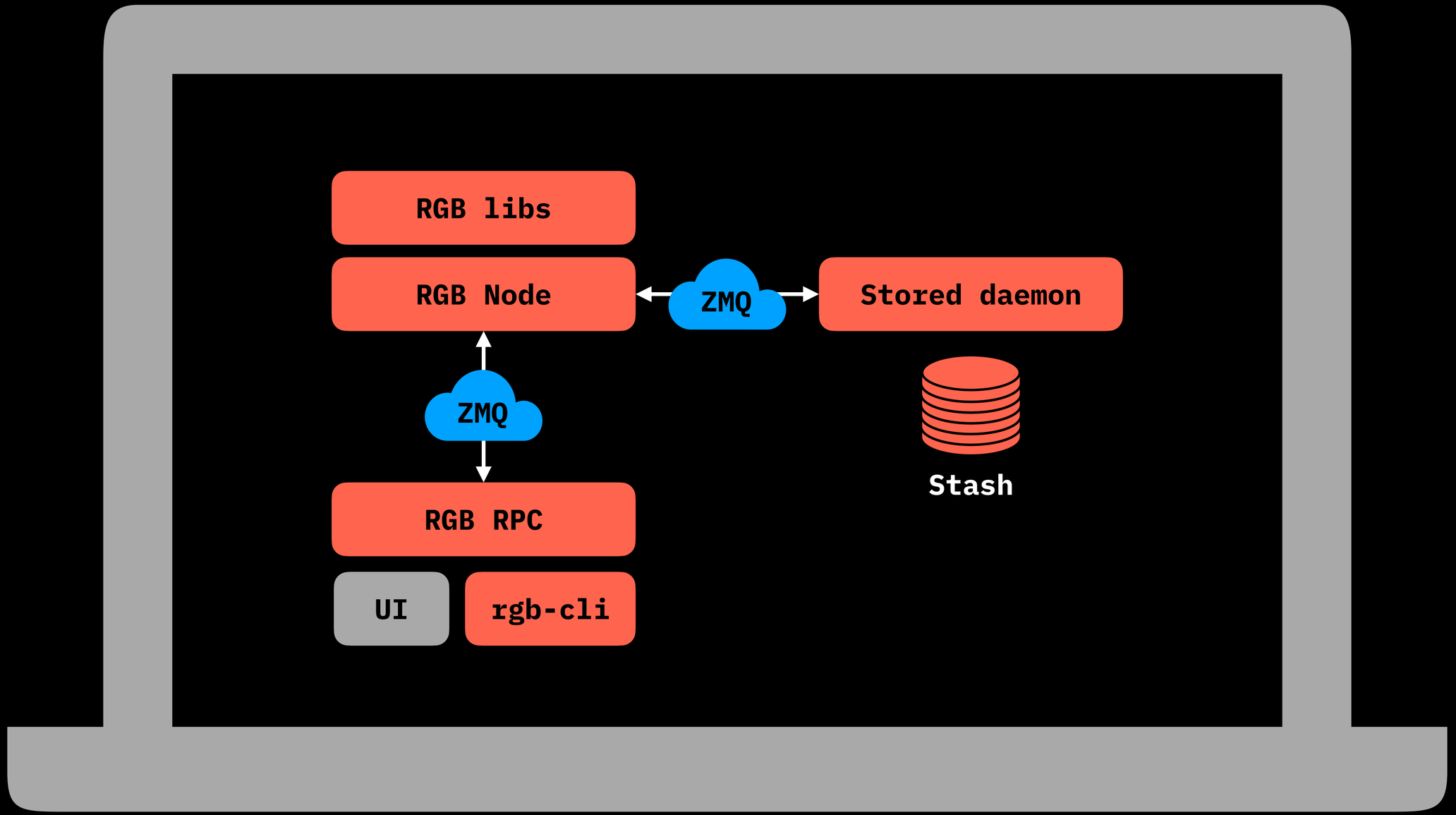
- **Global state** in smart contracts
- **Custom data types** for contract state (global and owned)
- Migration to new virtual machine **AluVM** (no more pre-defined constants restricting development of fully custom smart contracts)
- RGB contract state introspection for scripts

Application level

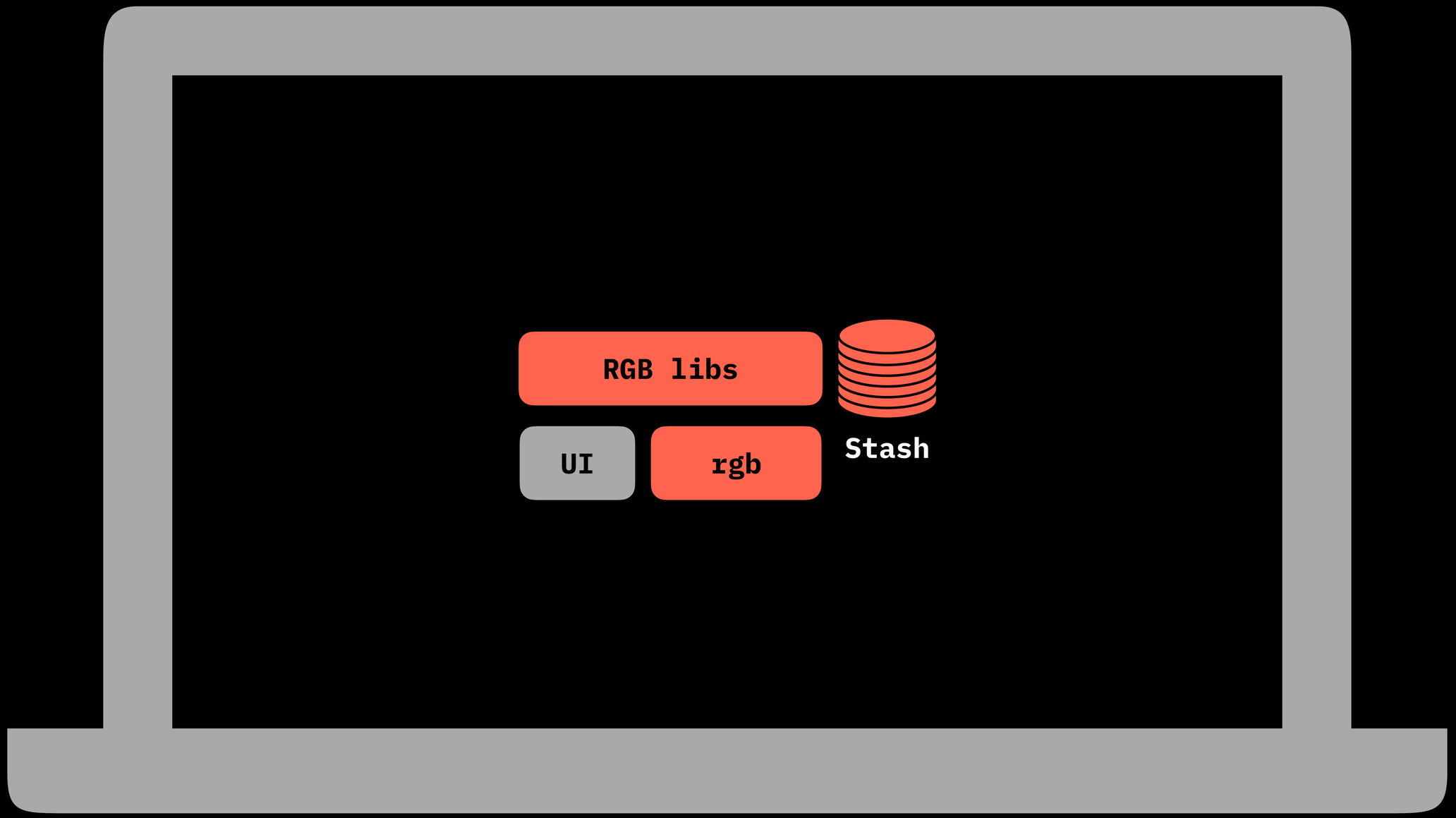
- **Contract interfaces**: unlocking independent developers power and flexibility of wallet integration
- Support for **WASM**
- Better **mobile** support (no ZMQ, no custom threads, ability to backup)
- No need to run a node on desktop! New simple **rgb** command to manage everything related to rgb.


Before
RGB libs
RGB Node
ZMQ
RGB RPC
FFI bindings
API wrappers
UI
Now
RGB libs
FFI bindings
JSON
UI

Before


RGB libs
RGB Node
ZMQ
Stored daemon
ZMQ
Stash
RGB RPC
UI
rgb-cli

Now



New RGB library stack for app devs

Server-side

Lightning, networking, always online

RGB Node

Desktop

Mobile

WASM/Web

Exchanges

RGB (rgb-contracts)

PSBT operations & descriptors support

RGB Wallet

Custom PSBT libs

RGB Standard Lib

Non-consensus critical

Consensus, validation

RGB Core Lib

RGB on GitHub

rgb Public

RGB smart contracts: client-facing library & command-line for desktop and mobile

Rust 1 2

rgb-node Public

RGB node - the official server-side implementation

Rust 108 36

rgb-wallet Public

RGB wallet & standard libs for web & low-level integrations

Rust 9 9

rgb-core Public

RGB Core Library: consensus validation for private & scalable client-validated smart contracts on Bitcoin & Lightning

Rust 117 23

blackpaper Public

RGB blackpaper

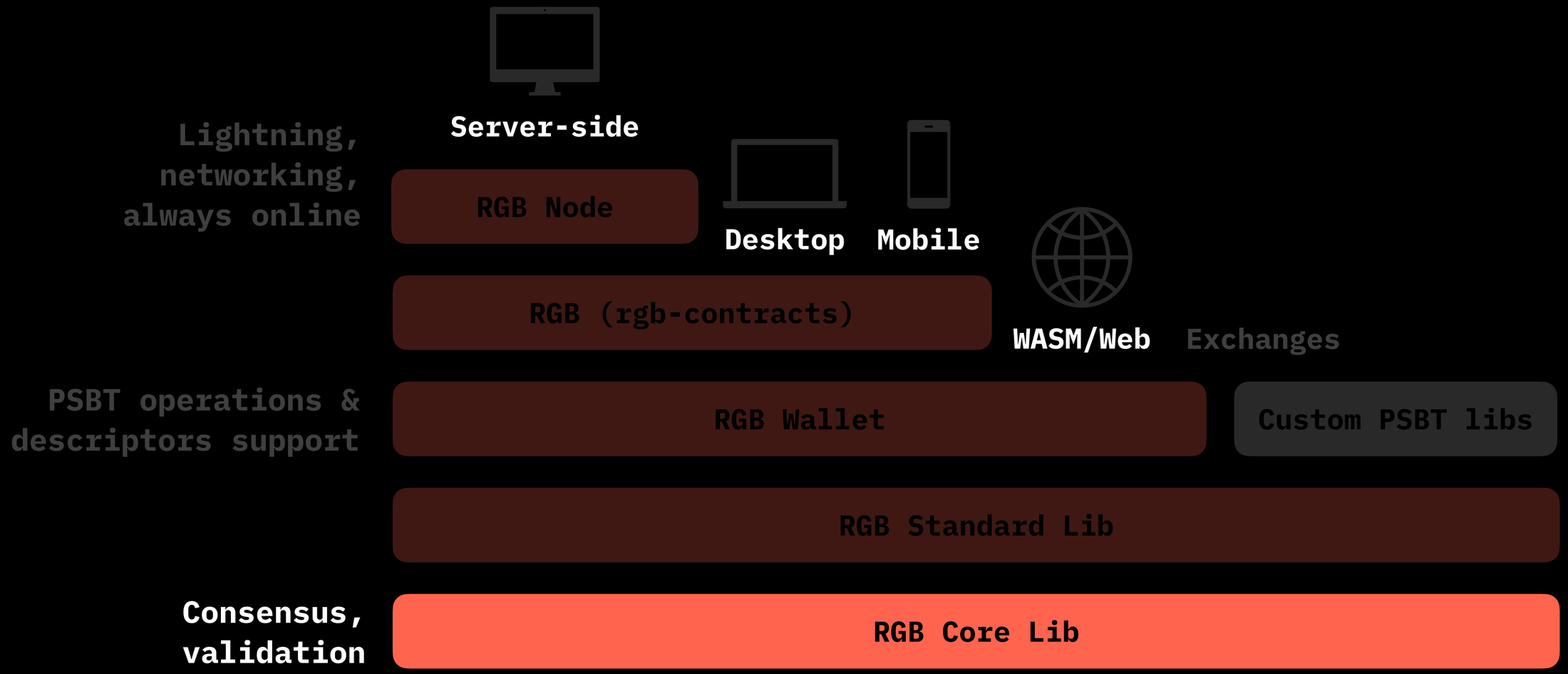
2 1

contractum-lang Public

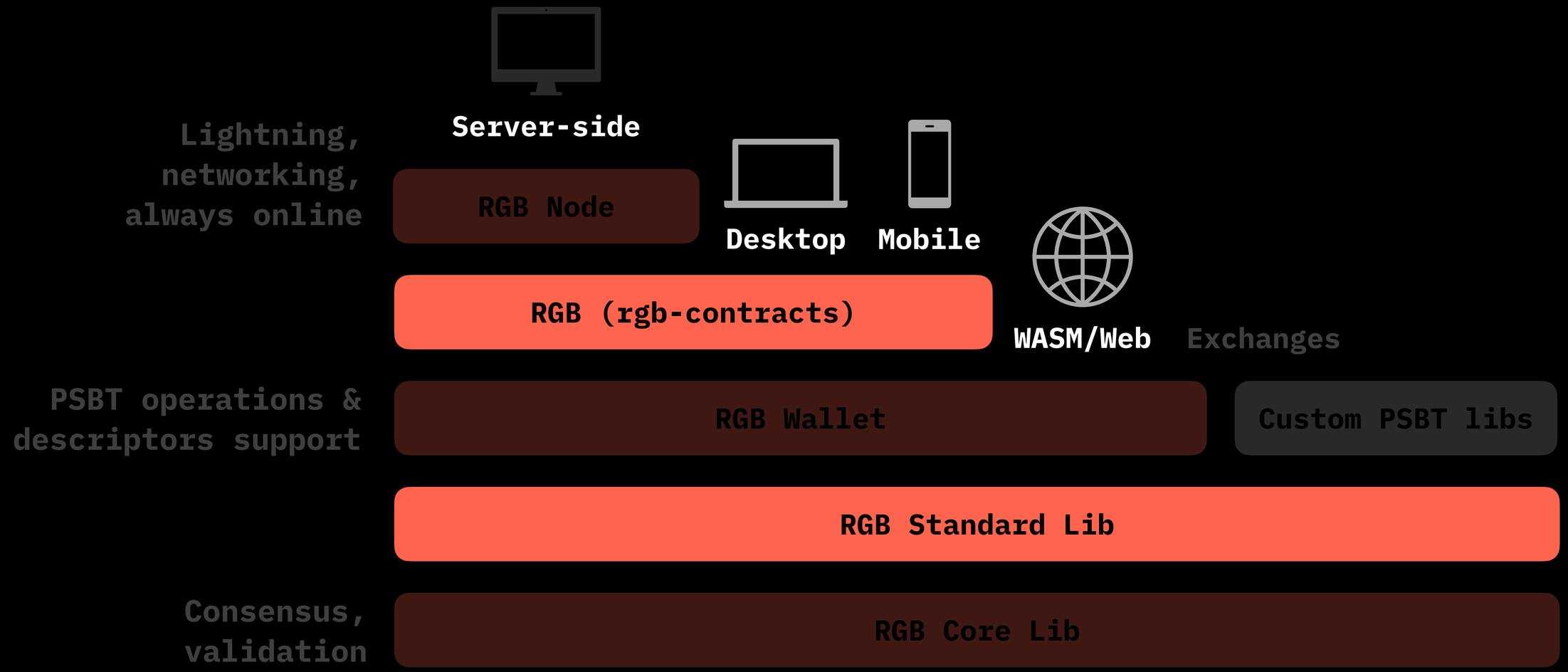
Contractum: RGB contract definition language

25 1

New RGB library stack for app devs



New RGB library stack for app devs



$ **rgb**

One command to rule all contracts

- See what contracts do you have:
  $ rgb
- Issue an asset:
  $ rgb issue rgb20 <schema> <genesis-file>
- Check contract state:
  $ rgb state <short-contract-name>
- Check contract state for a wallet descriptor:
  $ rgb state <short-contract-name> <descriptor>

```
$ cargo install rgb --all-features
```

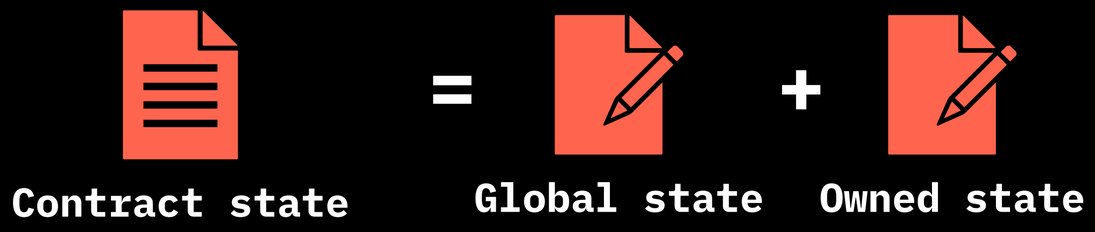
How to install

Understanding contracts & interfaces

Contract state

- **Global state**: nobody owns, everyone knows
  Examples: asset name

- **Owned state**: someone owns, nobody else knows
  Examples: asset amounts, voting rights, NFT content



Contract state

- **Global state:** nobody owns, everyone knows
- **Owned state:** someone owns, nobody else knows
  - **Declarative**: some form of governance rights which can be executed by a contract party
  - **Fungible**: hidden with ZK pedersen commitments
  - **Structured**: structured data types of arbitrary content (up to 64KiB)
  - **Attachments**: any attachments (media, audio, text, binary etc)

Global state accessibility

- Global state is published to distributed data networks (LN/Storm, IPFS, Torrent) or on centralized servers by the creators and participants of the contracts
- It’s up to them to incentivize/pay for that storage since they are interested about its accessibility
- This arguably provides even *stronger* guarantees than bitcoin block storage - and much higher scalability

Global state vs blockchain data

- Blockchain data can be pruned, such that only own state is kept
- There is no incentives for peers to provide past blocks in Bitcoin P2P network other than “common good”
- Global state preservation/availability is paid-for by the parties directly incentivized in its availability

Global and owned state can be structured

- Arbitrary data which can be expressed in JSON, YAML, TOML or other data formats
- May be up to 64KiB in a compact binary-serializied form
- Defined and serialized using Strict Types system
- Parsable via AluVM op codes and inspectable from inside the contract scripts

Owned state & assignments

- When state data are assigned to an UTXO,
  an **assignment** is created, which consists of:
  - **Seal definition** (UTXO plus commitment scheme)
  - **Owned state** (state data)

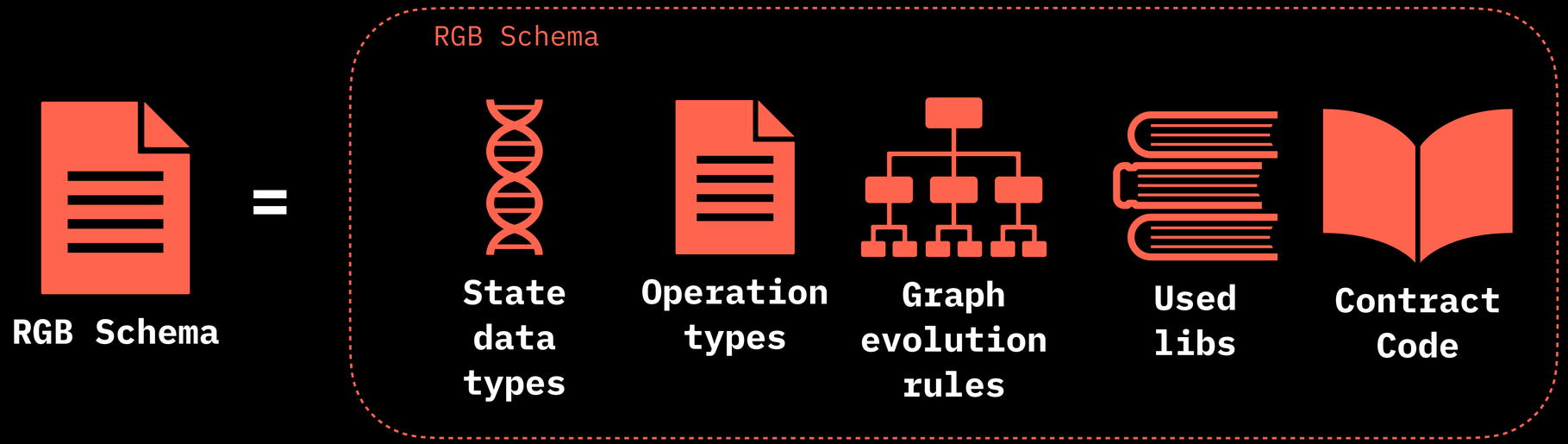
State validation

State is validated with:

- Fixed declarative rules
  (which state can exist and how it can accumulate)
- Custom script logic written for AluVM virtual machine

RGB contract schema defines:

- Which types of state can exist
- Data types used in state
- Which operations can be performed with the contract
- Validation rules and scripts



Schema differentiates **contract devs** from **issuers**, which might not know anything about coding and programming

Schema allows creation of “*contract templates*” avoiding most common pitfals for the issuers

Demo: writing a contract schema

Contract A

A fungible asset

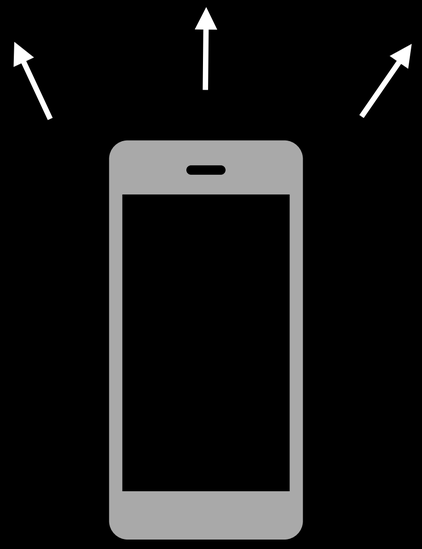
Contract B

An NFT collection

Contract D

DAO with many identities and two tokens + NFTs

WTF?

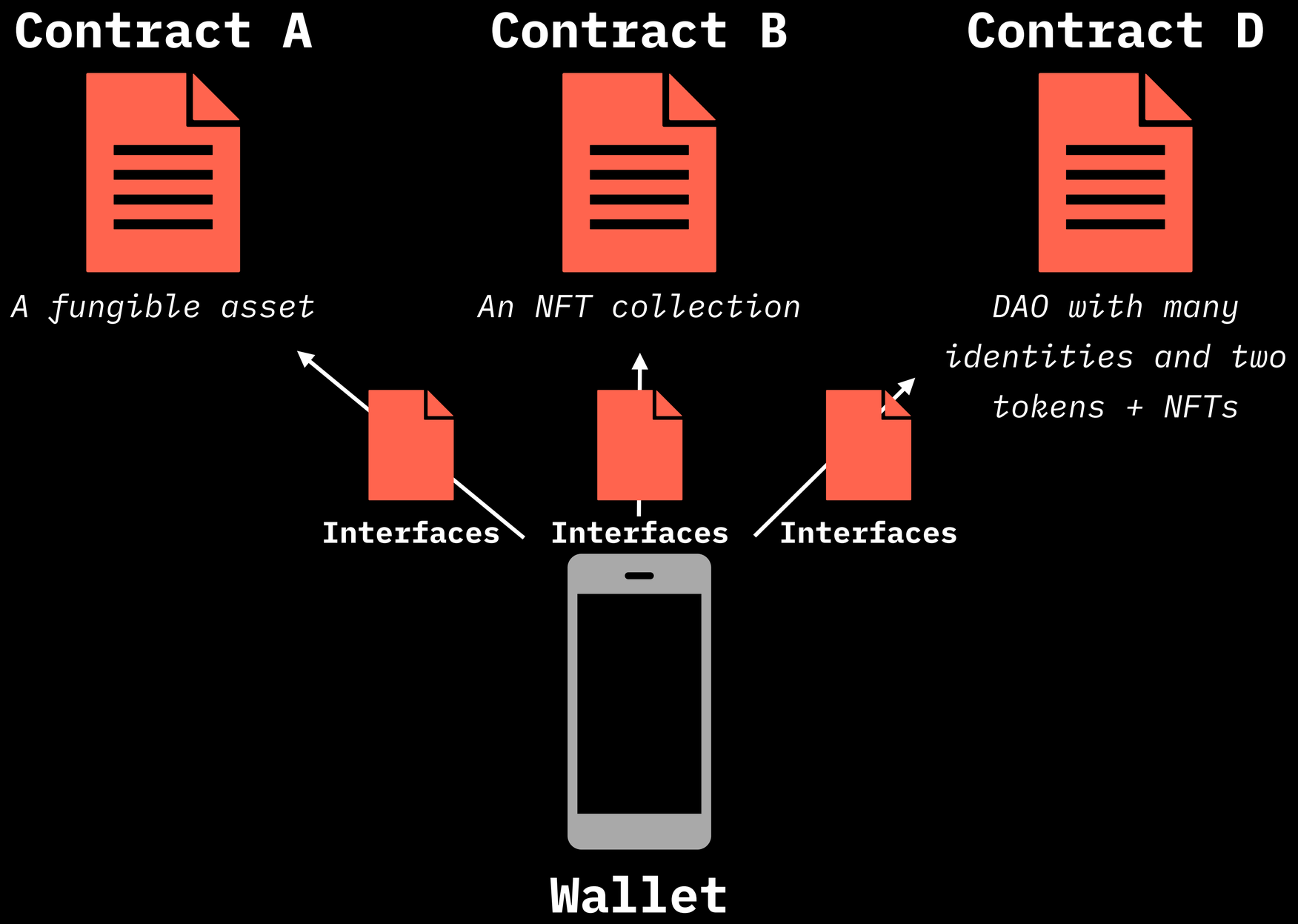


Wallet

Interface

Human- and wallet-readable information about the contract (state, operations)

“Interface” or “trait” in context of OOP languages


Contract A
Contract B
Contract D
A fungible asset
An NFT collection
DAO with many identities and two tokens + NFTs
Interfaces
Interfaces
Interfaces
Wallet

Interfaces help to make sense of different contracts for the user

Interface tells wallet or a user about semantics of the contract:

- What state the contract can have and how to read it/parse it
- What operations can be performed on the contract
- Which arguments those operations take

- If a contract is a “class”,
  interface is “interface” of that class
- In “ethereum world” interface is ERC* standard
- In LNP/BP world interface is a compiled program
  in binary or BASE64-encoded form
- A contract may implement **MANY** interfaces to deliver advanced
  functionality

Interfaces unlock a way for independent developers to write any kind of contract

– and users to use them –

without RGB libs, wallet or software updates!

Interface

Human- and wallet-readable information about the contract (state, operations)

=

Demo: writing an interface

Current interfaces by LNP/BP Association

- **RGB20**: fungible assets - like company shares or shitcoins
- **RGB21**: digital collectibles - like NFTs, books, music, shitposts & stupid memes - but without wasting blockspace (!) like in ordinals
- **RGB22**: digital identity
- **RGB23**: provable history of operations (OpenTimeStamps squared)
- **RGB24**: global domain name system (like ENS, but much better)

These interfaces are created by us and shipped with RGB standard library. But anyone can create their own interface without the need to contact us!

Interface

Contract

=

Schema

Requirements for the contract state + contract business logic

"Class" in terms of OOP

+

Interface implementation

Bindings to human & wallet-readable names from the interface

Implementation of an interface/trait for a class/struct

+

Genesis

Initial setup of the contract state

Instance of a class created by the class constructor

Contract components overview:

Contract components	Meaning	OOP terms	Ethereum terms
Interface	*Contract semantics*	Interface (Java), trait (Rust), protocol (Swift)	ERC* standards
Schema	*Contract business logic*	Class	Contract
Interface implementation	*Mapping semantics to business logic*	Impl (Rust), Implements (Java)	ABI
Genesis	*Initial contract state*	Class constructor	Contract constructor

Demo: writing a contract

Distributing data

Consignments

Full information about the contract:

- “Contract consignment”: created by the contract issuer; provides information about the contract to the users
- “Transfer consignment”: created by contract users; transfers some state to a new owner

Bindles

a package containing a piece of RGB data which may be signed:

- Schema
- Interface
- Interface implementation
- Consignments

Bindle & Consignment forms

Binary

```
00000000  52 47 42 43 4e 52 43 ec  6b f5 47 25 5d 8d a6 61  |RGBCNRC.k.G%]..a|
00000010  1f bb e8 f3 76 16 a5 46  f2 07 75 e4 cb 65 1f 06  |....v..F..u..e..|
00000020  a1 64 d4 2a 9d 6e 49 01  01 00 00 00 02 00 00 2a  |.d.*.nI........*|
00000030  b4 84 b5 0d e3 23 7b 6d  9c 7f fe 11 9c ac 7e c4  |.....#{m......~.|
00000040  f0 8d 6a ac 5f 16 ab d6  1a 1a 50 34 f3 ab a4 01  |..j._.....P4....|
00000050  00 01 00 f0 d3 a8 af 46  3a 63 ce fc 4a 3a 21 2d  |.......F:c..J:!-|
00000060  49 f4 91 37 07 6a 78 7e  6f 88 f9 86 30 3d 08 63  |I..7.jx~o...0=.c|
00000070  dd ca 30 01 00 01 00 00  01 08 00 00 02 00 00 01  |..0.............|
00000080  00 01 00 01 00 01 00 01  00 01 00 00 01 00 ff ff  |................|
00000090  00 00 01 00 00 00 00 01  00 00 01 00 ff ff 01 00  |................|
000000a0  00 01 00 ff ff 00 0f 00  00 22 83 36 73 9c 9b 08  |.........".6s...|
000000b0  ca 2b f3 ac 42 34 3d 91  b1 e7 b0 88 2c 59 95 ce  |.+..B4=.....,Y..|
000000c0  3d af 9d 6b f4 7f d3 03  45 01 0b 52 47 42 43 6f  |=..k....E..RGBCo|
000000d0  6e 74 72 61 63 74 0c 43  6f 6e 74 72 61 63 74 4e  |ntract.ContractN|
000000e0  61 6d 65 05 01 e5 80 dc  cf 6f 4e 62 3f 25 eb 43  |ame......oNb?%.C|
000000f0  78 b4 d9 97 05 73 6c a3  6b 66 66 1a 2c 00 f8 a4  |x....sl.kff.,...|
00000100  89 e4 72 b7 ab 2a b4 84  b5 0d e3 23 7b 6d 9c 7f  |..r..*.....#{m..|
00000110  fe 11 9c ac 7e c4 f0 8d  6a ac 5f 16 ab d6 1a 1a  |....~...j._.....|
00000120  50 34 f3 ab a4 01 0b 52  47 42 43 6f 6e 74 72 61  |P4.....RGBContra|
00000130  63 74 07 4e 6f 6d 69 6e  61 6c 06 04 06 74 69 63  |ct.Nominal...tic|
00000140  6b 65 72 92 e7 5c a8 7a  0d 45 ef 31 9d d9 73 90  |ker..\.z.E.1..s.|
00000150  2d 8d a2 d2 68 2e 4c 7e  23 df 47 9b fe 7b e8 42  |-...h.L~#.G..{.B|
00000160  98 35 06 04 6e 61 6d 65  22 83 36 73 9c 9b 08 ca  |.5..name".6s....|
00000170  2b f3 ac 42 34 3d 91 b1  e7 b0 88 2c 59 95 ce 3d  |+..B4=.....,Y..=|
00000180  af 9d 6b f4 7f d3 03 45  07 64 65 74 61 69 6c 73  |..k....E.details|
00000190  36 56 13 c0 31 43 92 f9  73 6b 10 b0 36 64 6f f9  |6V..1C..sk..6do.|
000001a0  6c ea 78 85 a1 9c 19 f2  0d fb d4 b6 64 03 4e 4a  |l.x.........d.NJ|
000001b0  09 70 72 65 63 69 73 69  6f 6e ca 43 47 df c0 05  |.precision.CG...|
```

BASE64 “RGB Armored”

```
----- BEGIN RGB SCHEMA -----
Id: 31ci37ipberq62njEHxPCzXrWyPeg7Hp7k3QxMmUv7Jy
Checksum: script-null-analyze

AAAAAgAAKrSEtQ3jI3ttnH/+EZysfsTwjWqsXxar1hoaUDTzq6QBAAEA8NOor0Y6Y878SjohLUn0
kTcHanh+b4j5hjA9CGPdyjABAAEAAAEIAAACAAABAAEAAQABAAEAAQAAAQD//wAAAQAAAAABAAAB
AP//AQAAAQD//wAPAAAigzZznJsIyivzrEI0PZGx57CILFmVzj2vnWv0f9MDRQELUkdCQ29udHJh
Y3QMQ29udHJhY3ROYW1lBQHlgNzPb05iPyXrQ3i02ZcFc2yja2ZmGiwA+KSJ5HK3qyq0hLUN4yN7
bZx//hGcrH7E8I1qrF8Wq9YaGlA086ukAQtSR0JDb250cmFjdAdOb21pbmFsBgQGdGlja2Vykudc
qHoNRe8xndlzkC2NotJoLkx+I99Hm/576EKYNQYEbmFtZSKDNnOcmwjKK/OsQjQ9kbHnsIgsWZXO
Pa+da/R/0wNFB2RldGFpbHM2VhPAMUOS+XNrELA2ZG/5bOp4haGcGfIN+9S2ZANOSglwcmVjaXNp
b27KQ0ffwAUxChrlWSt91vPpNkXlIxMF++mS7zPOLNrjHjZWE8AxQ5L5c2sQsDZkb/ls6niFoZwZ
8g371LZkA05KAAQCAARub25lxr3uYkgVMfD14HWM2mhgU8BQxPxK0SGaXFuOW/3BtrEBBHNvbWXQ
oLrLVkdFLcU1qzqVNaaZpV+ob/I5EpcZvy5QBGOUFD5Vj/oPDDjCK2G+TRELiTaU1RxTTHBvNIrz
EyUeyQM2AAhhzZA987T7tNgo01xT+Xbj+B3J4Rggsqp2Zh5NKKJVEAAAAAAAAAAA//8AAAAAABE
b7txuCcDfm33+B2Dgog2XhUrpNxAbWpDkg3JimxvJQAIYc2QPfO0+7TYKNNcU/l24/gdyeEYILKq
dmYeTSiiVRAoAAAAAAAAAP8AAAAAAAAAVXfuITiV7tOkpOgi23z2B49LXiDXcnC6PBC/iXPGzUsA
A2ADXzMyIANfMzMhA18zNCIDXzM1IwNfMzYkA18zNyUDXzM4JgNfMzknA180MCgDXzQxKQNfNDIq
A180MysDXzQ0LANfNDUtA180Ni4DXzQ3LwNfNDgwA180OTEDXzUwMgNfNTEzA181MjQDXzUzNQNf
NTQ2A181NTcDXzU2OANfNTc5A181ODoDXzU5OwNfNjA8A182MT0DXzYyPgNfNjM/A182NEADXzY1
QQNfNjZCA182N0MDXzY4RANfNjlFA183MEYDXzcxRwNfNzJIA183M0kDXzc0SgNfNzVLA183NkwD
Xzc3TQNfNzhOA183OU8DXzgwUANfODFRA184MlIDXzgzUwNfODRUA184NVUDXzg2VgNfODdXA184
OFgDXzg5WQNfOTBaA185MVsDXzkyXANfOTNdA185NF4DXzk1XwNfOTZgA185N2EDXzk4YgNfOTlj
BF8xMDBkBF8xMDFlBF8xMDJmBF8xMDNnBF8xMDRoBF8xMDVpBF8xMDZqBF8xMDdrBF8xMDhsBF8x
MDltBF8xMTBuBF8xMTFvBF8xMTJwBF8xMTNxBF8xMTRyBF8xMTVzBF8xMTZ0BF8xMTd1BF8xMTh2
BF8xMTl3BF8xMjB4BF8xMjF5BF8xMjJ6BF8xMjN7BF8xMjR8BF8xMjV9BF8xMjZ+BF8xMjd/XNDD
```

RGB identifiers

TommyFuelPagoda07EnUZgFtu28sWqqH3womkTopXCkgAGsCLvLnYvNcPLRt

- Base on Baid58 encoding (Base58 + readable mnemonic + HRP)
  - Shorter than Bech32
  - More human-safe than Bech32
  - Abandons invalid promises of Bech32 (QR friendliness, “easy to read”, “error correction”)
- May be structured as URLs by adding prefix (for instance, *rgb:* for contract ids)

Keeping & backing up the data

Stash

- Consensus-critical client-side-validated data in the original form
- Must not be accessed or used directly
- Must be backed up

Inventory

- Derivatives from the stash for simplified runtime access:
  - Current contract state
  - Different indexes
- May be backed up
- Can be re-generated from stash (but this is very complicated)

Stock is in-memory
  Stash + Inventory
for WASM

Where the data are kept

- OS-specific locations:
  - *~/.rgb* (Linux, UNIX)
  - *~/Application Support/RGB Smart Contracts* (MacOS)
  - *~/AppData/RGB Smart Contracts* (Windows)
  - *~/Documents* (Android)
- Subdirs for networks (*mainnet*, *testnet*, etc)
- What's there
  - Today - just a single binary file ("*stock.dat*")
  - Tomorrow - stash & inventory directories
    + multiple files in them

What’s next

- **Next week:**
  - RGB wallet functionality
  - State transfers
- **Spring & summer:**
  - Lightning network operations
  - Nodes (RGB, LNP, Storm)
  - DEX
- **Autumn:** toolchain for RGB devs —
  Contractum, compilers, package managers, code/schema repositories.