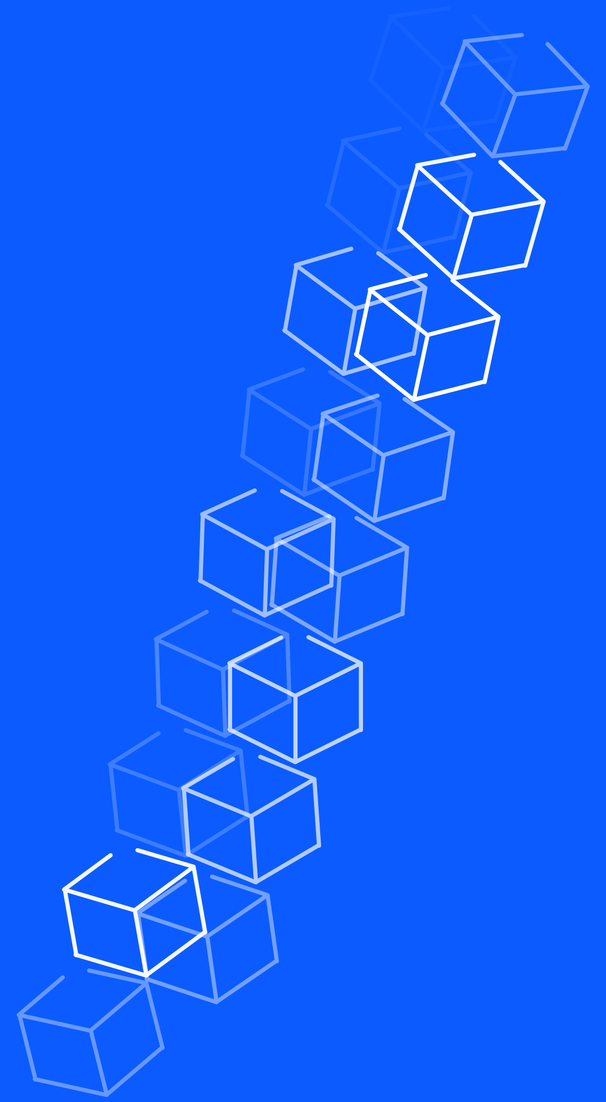
WEB 3

Smart Contract Workshop



waves

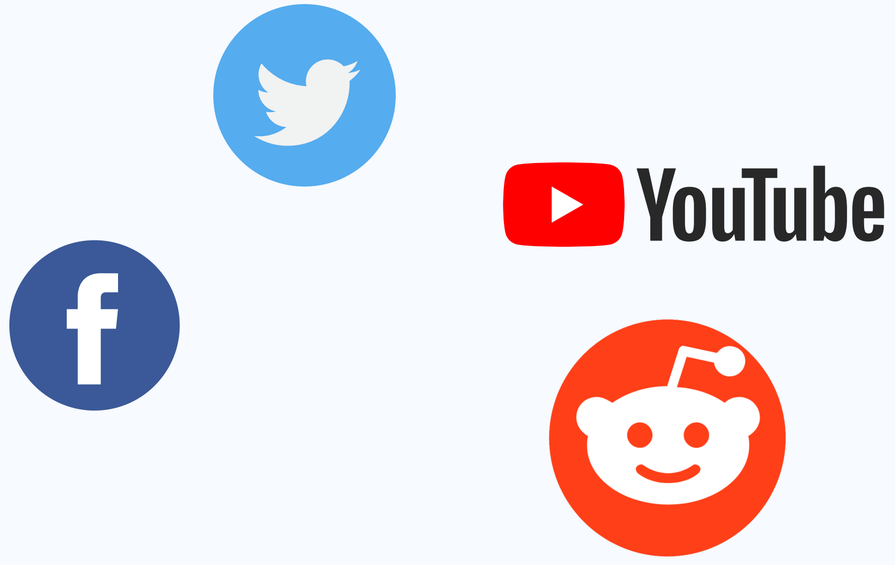
Web 1.0

- “Information should be open & accessible”
- Launch of World Wide Web
- Interconnected information
- Server owners made content

Sir Tim Berners-Lee

Web 2.0

- “Information should be open & accessible”
- Lanch of social media
- Online communities
- User made content



Web 3.0

- “Finance should be open & accessible”
- Lanch of blockchains
- User govern their own finance

Structure

1. Waves Platform - Features & Advantages

2. Brief Introduction to Smart Contracts

3. RIDE - A simple & concise Smart Contract Language

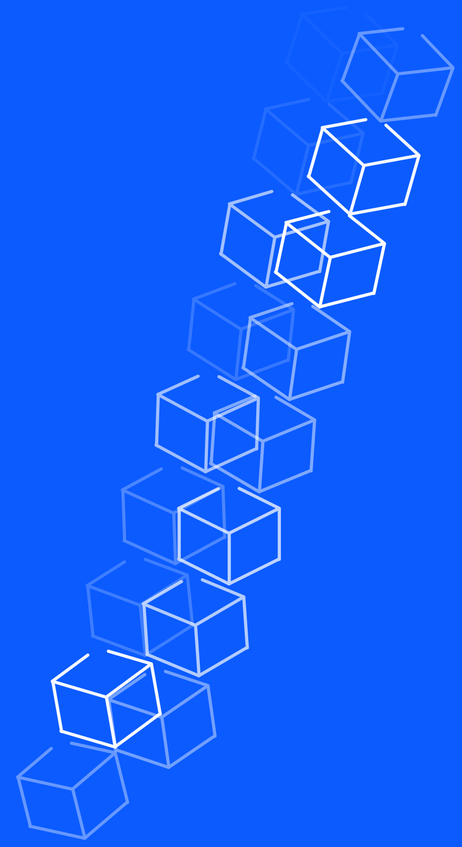
4. Waves Labs - How to get funding for you dapp idea

5. CV Labs - Blockchain Incubator

WEB 3

This workshop is for you!

Questions are appreciated

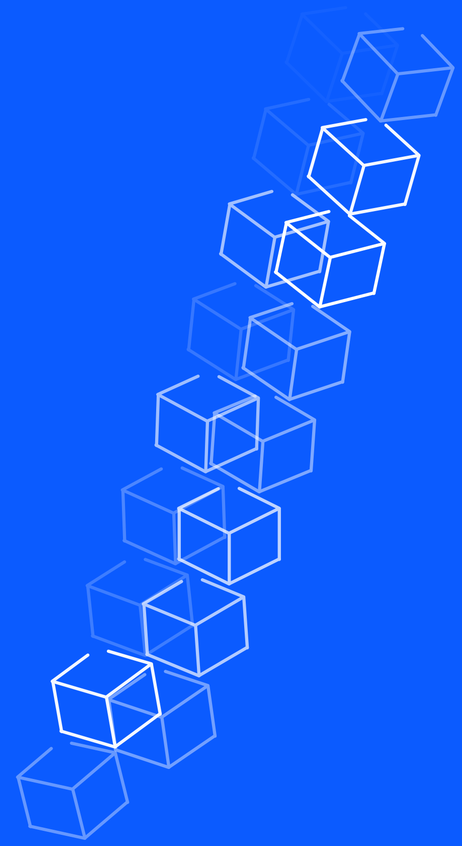


Daniel Lutz / 2019

waves

WEB 3

What is Waves Platform actually?



waves

Daniel Lutz / 2019

History

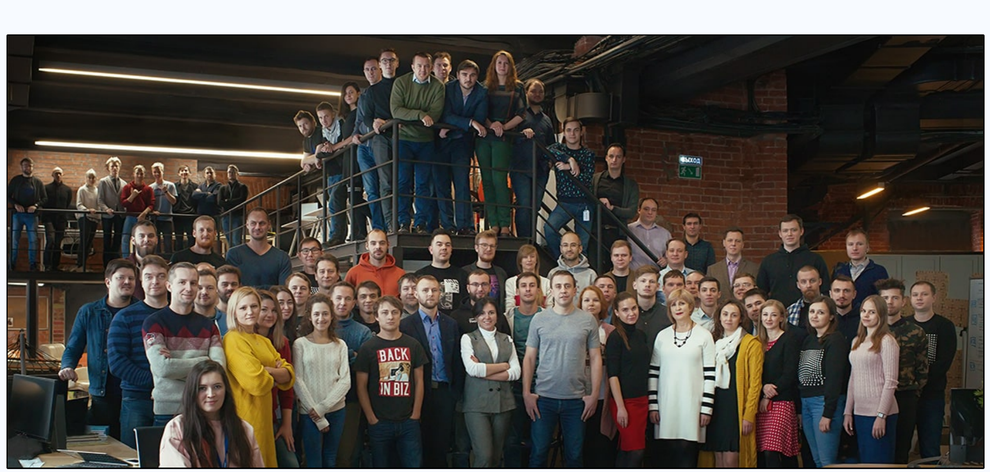
- Launch in Q3 2016
- Focus on **speed**, **scalability**, **security** and **user experience**
- Smart contracts
- **Decentralized OSS** & **centralized company**
- Company founded by **Alexander Ivanov**

waves

https://wavesplatform.com
https://messari.io/asset/waves#profile

Global growth

- More than 200 full time employees
- Offices in Moscow, Zug, Amsterdam
- Ambassadors all over the world
- Incubator (Waves Labs, Ventuary DAO)
- Impactful business partnerships



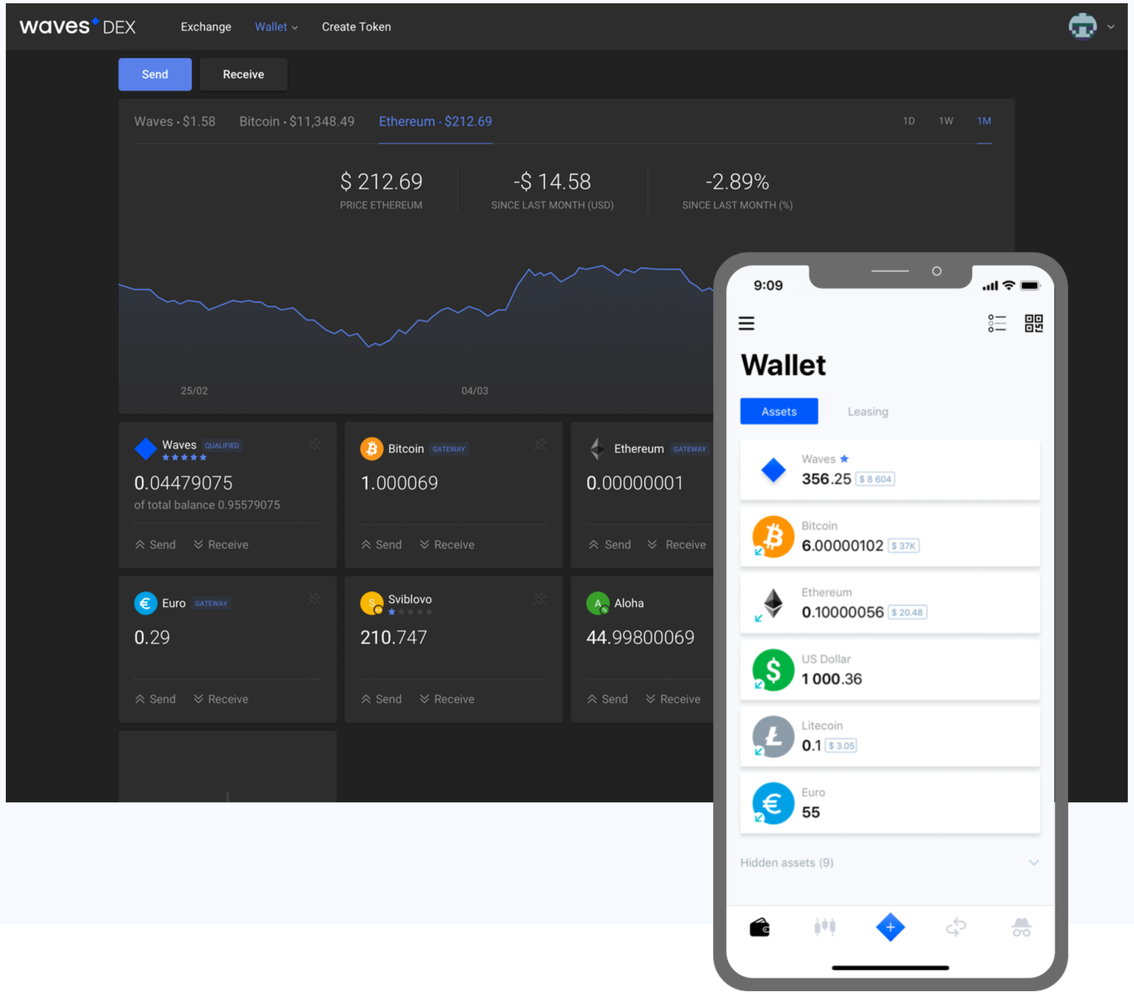
Icon made by srip from www.flaticon.com

Multipurpose wallet

- Accessible UI & great UX
- Integrated **DEX**
- FIAT gateways
- ERC20 tokens

Supported platforms

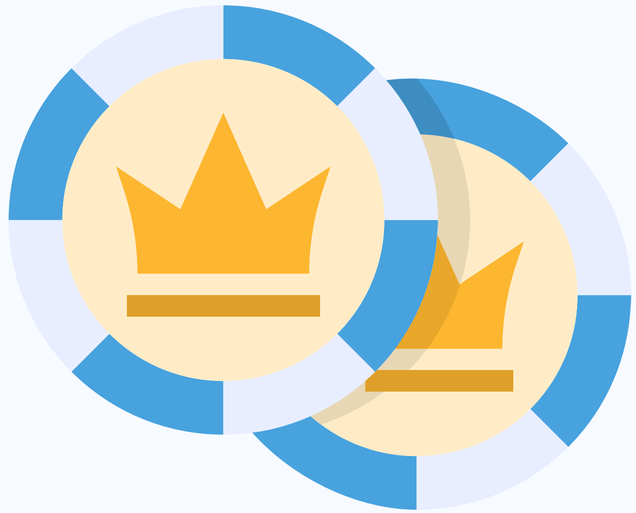
- Windows
- MacOS
- Linux
- Android
- iOS
- Browser



waves

Custom Tokens

- Issue your own token
- Great for loyalty programs, **NFTs** or dapps
- Tradeable on DEX
- **Programmable** (Smart asset)



Icon made by dDara from www.flaticon.com

Issuance Fee

1 $WAVES

---

https://docs.wavesplatform.com/en/smart-contracts/smart-assets.html

Aliases

- Send to alias instead of address
- DNS-like address lookup
- One address can have multiple aliases

Icon made by Eucalyp from www.flaticon.com

Issuance Fee

0.001 $WAVES

https://docs.wavesplatform.com/en/blockchain/account/alias.html

Flat fees

- Flat fee structure
- Transaction can **never “run out of gas”**
- Great UX, because **transaction never fail**
- For more information, check the docs

Transaction fee

A **transaction fee** is a fee that an account owner pays to send a transaction.

A sender can specify any amount of fee but not less than a certain amount. The larger the fee is, the quicker the transaction will be added to the new block.

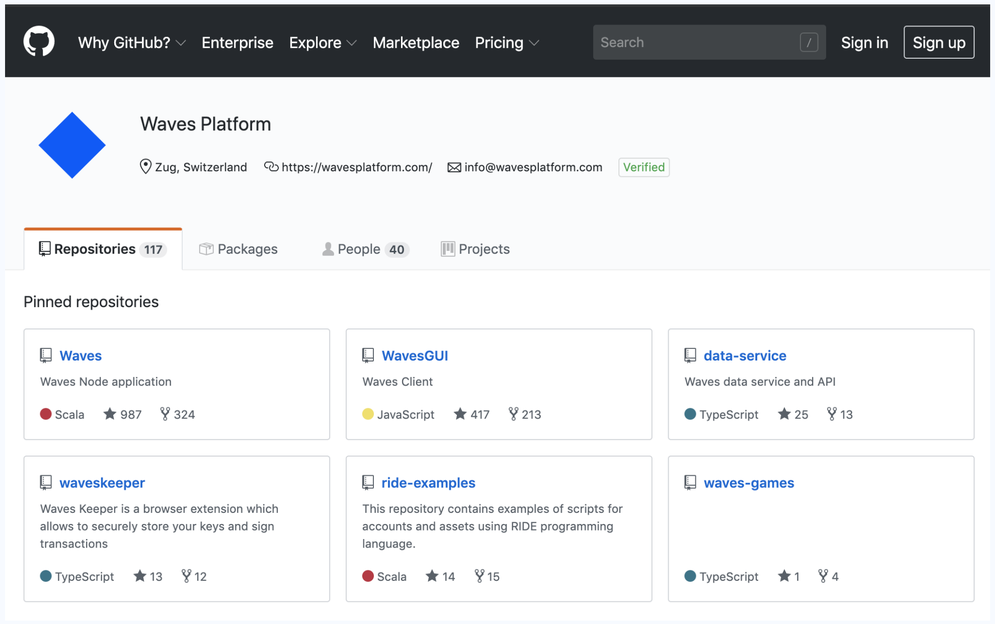
> If a smart account transfers a smart asset, then the fee doubles.
> If a transaction is validated by an account script or an asset script, then the fee is increased by 0.004 WAVES

Transaction type	Transaction type ID	A minimum transaction fee in WAVES	Comments
Alias transaction	10	0.001	
Burn transaction	6	0.001	
Data transaction	12	0.001 per kilobyte	The value is rounded up to the thousandths
Exchange transaction	7	0.003	

https://docs.wavesplatform.com/en/blockchain/transaction/transaction-fee.html

Supportive dev community

- Big communities on multiple platforms
- Ask right questions, get right answers
- Constantly improving developer tools & documentation

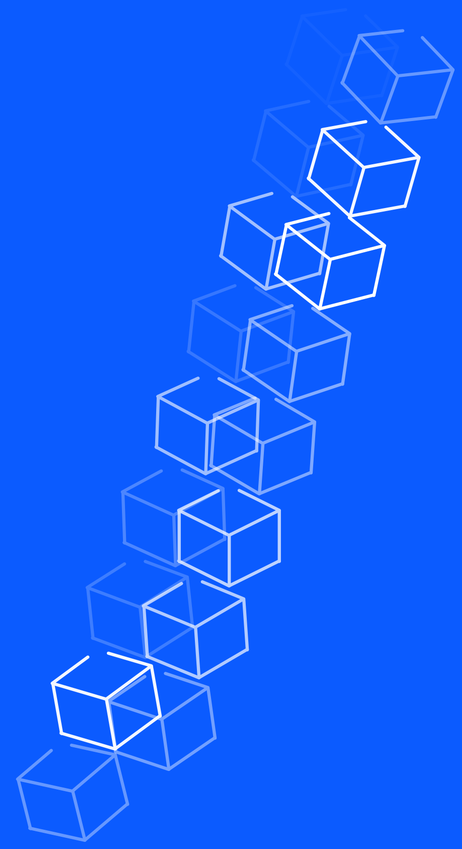


https://github.com/wavesplatform

waves

WEB 3

Quiz
Wavesplatform



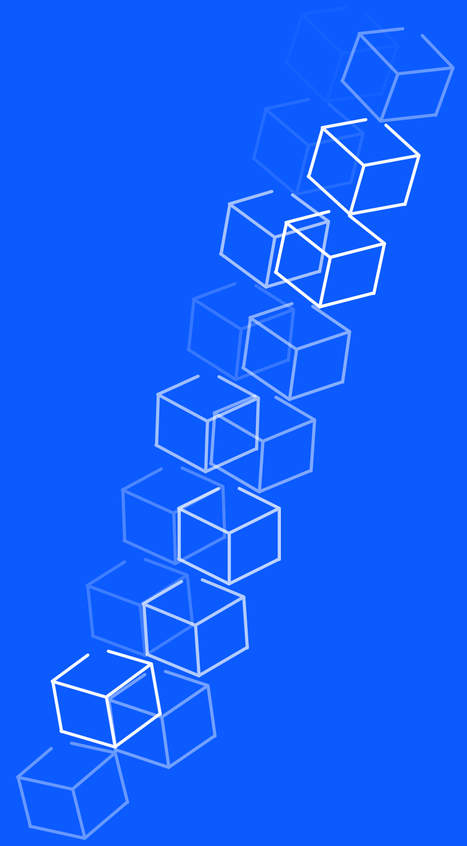
waves

Daniel Lutz / 2019

WEB 3

Smart Contracts

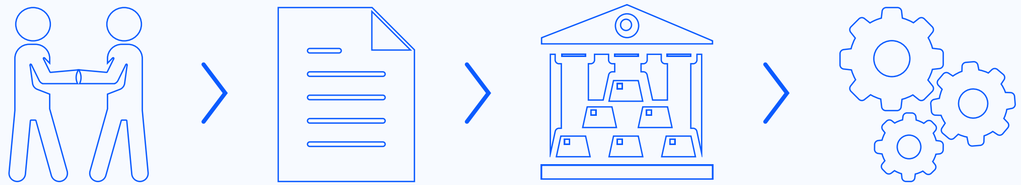
How blockchain helps us enforcing agreements



waves

Daniel Lutz / 2019

TRADITIONAL CONTRACT



WEB3

SMART CONTRACT

Illustration by Aleksei Pupyshev

Benefits of a smart contract

- Increase **transparency**
- **Remove** the need for **trusted 3rd party** validation
- All operations and execution **results are stored in the blockchain**
- Contracts can also be used by machines, not just humans
- Execution costs are minimal

```
vokation)
w(amountWavelets: Int) = {
pientAddress = toBase58String(invokation.caller.bytes)
entDepositAmountWavelets = getDepositByAddress(recipientAdd

tWavelets > currentDepositAmountWavelets)
 throw("Balance of " + recipientAddress + " is to low (" +
 {
let updatedDepositAmountWavelets = currentDepositAmountWave

ScriptResult(
    WriteSet([DataEntry(recipientAddress, updatedDepositAmo
    TransferSet([ScriptTransfer(invokation.caller, amountWa
)
```

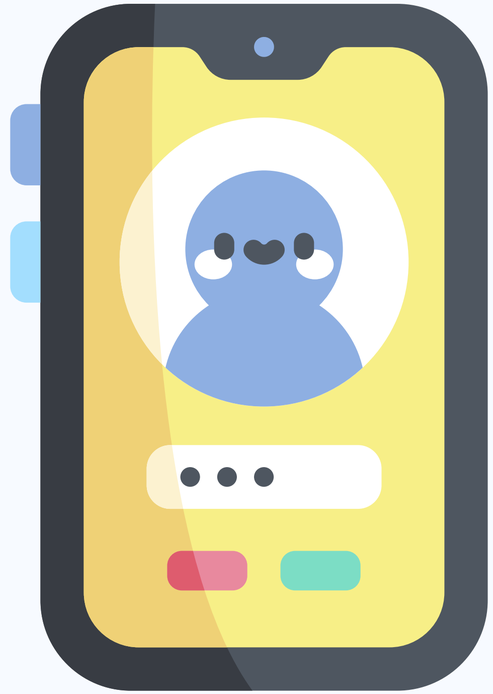
Snippet of a Ride smart contract

What (d)apps can i build with them?

- Exploration of possibilities is still ongoing

Some examples:

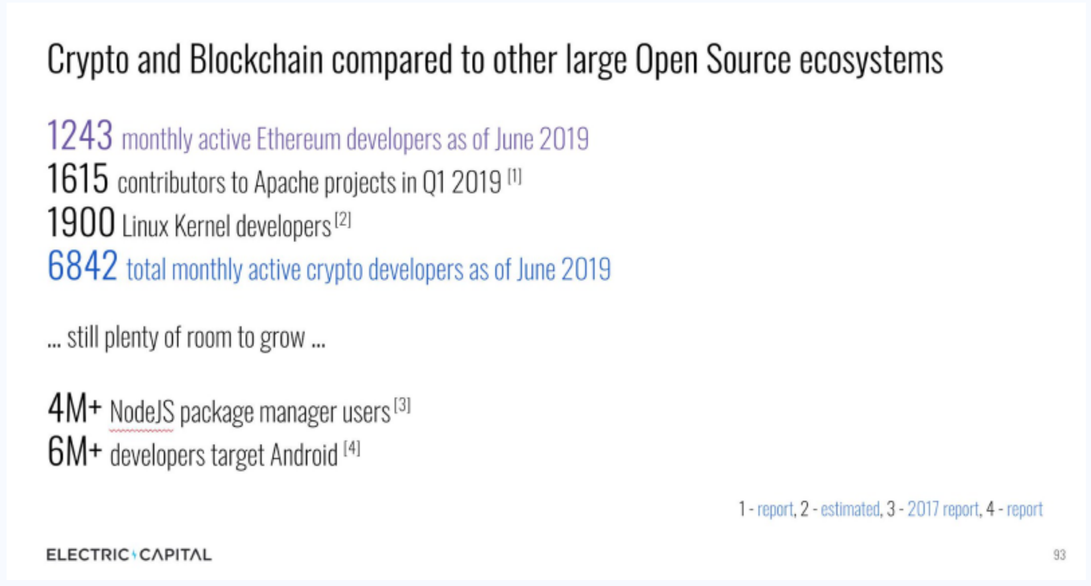
- **Multisig**nature solutions
- Transparent **voting** mechanisms
- **DAOs** (Decentralized Autonomous Organzations)
- Decentralized **swaps** of virtual assets
- **Games**

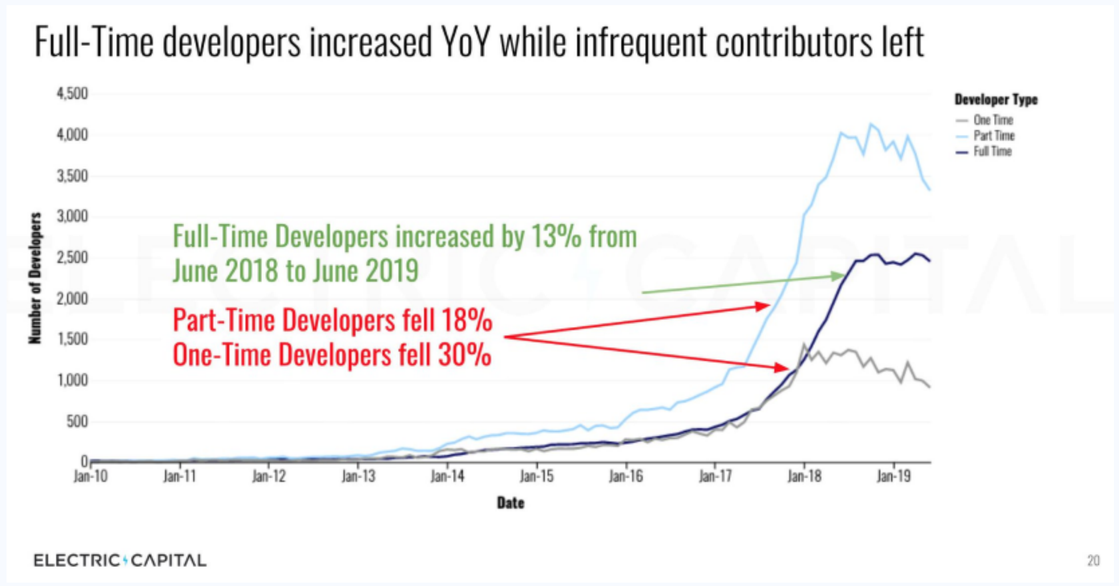


Icon made by Freepik from www.flaticon.com

Job perspectives in the industry

“Blockchain will create **$3.1 trillion in business value by 2030**” - Gartner

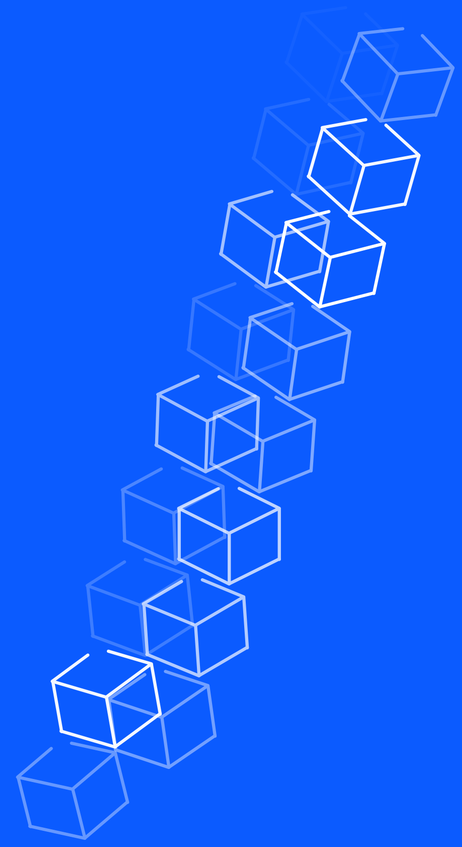




https://www.gartner.com/en/doc/3891569-top-10-strategic-technology-trends-for-2019
https://medium.com/@ElectricCapital/electric-capital-developer-report-h1-2019-7d836d68fecb

WEB 3

Quiz
Smart Contracts



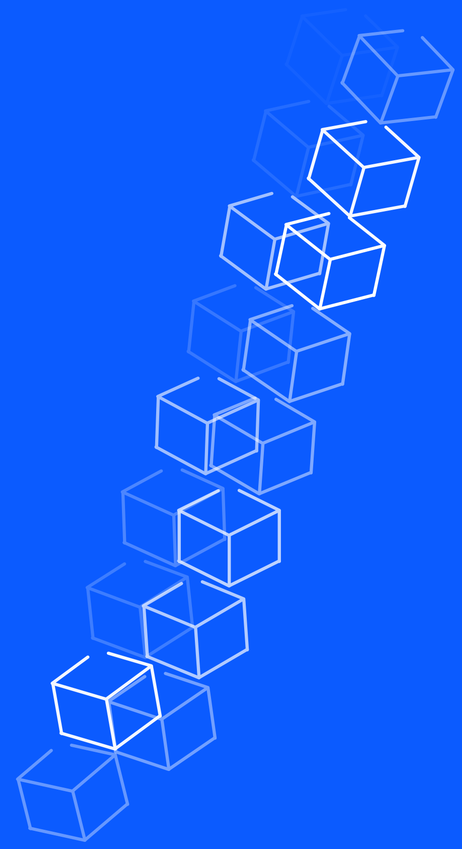
waves

Daniel Lutz / 2019

WEB 3

RIDE

Less code, more power



waves

Daniel Lutz / 2019

What is RIDE?

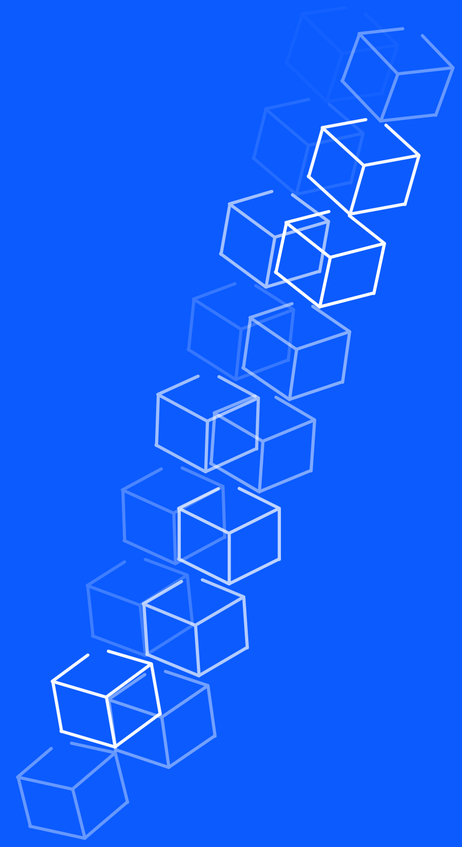
Ride is a **statically typed**, **functional**, **lazy**, **compiled** programming language

- Designed for developing smart contracts
- Concise, human-readable code
- Non-turing-complete

Language design is **inspired by Scala and F#**

WEB 3

What do all these fancy buzzwords mean?



waves

Daniel Lutz / 2019

Statically typed

Variable **types are inferred from value** on declaration.

- Types can’t be changed
- **Type safety** helps to prevent bugs & errors

```
13          String
14    let name = "Alice"
15
```

```
8           Int
9     let age = 18
10
```

Functional

Inspired by functional programming paradigm

- **Pure functions**
- **No side effects**
- **Immutability**

```
60    func double (number: Int) = {
61        number * 2
62    }
```

```
9   let age = 18
10
11      Int
12
13      Parsing error: Failure(End:15:1 ..."age = 97\n\n")
14      Peek Problem    No quick fixes available
15  age = 97
```

Not a 100% functional programming language, because *throw()* can terminate script execution.

Lazy

What is not used is not calculated.

- Applies to variables and functions
- **Optimizes node workload** on script execution

```
13  func sayHello (name: String) = {
14      let unusedName = "Bob" # This will not be calculated
15
16      "Hello " + name
17  }
```

```
19  func getAnswerToEverything() = {
20      42
21  }
22
23  func getSpecificAnswer() = {
24      let answer = getAnswerToEverything() # Also not calculated
25      "Specific answer" # ¯\_(ツ)_/¯
26  }
```

Variables

- Declared with *let*
- Immutable
- Lazy calculated

```
68
69    let awesomeCity = "Zürich"
70
```

No variable shadowing

Declaring variables with name that is present in a parent scope will result in a compilation error

```
68
69    let awesomeCity = "Zürich"
70
71    func getLocation () = {
72        let awesomeCity = "Amsterdam" # Compilation error
73    }
```

Data types

```
5    # Basic types
6    let name = "Rider" # String
7    let age = 18 # Int
8    let isCool = true # Boolean
9
```

```
10   # Special types
11   let friends = ["Alice", "Bob", "Eve"] # List
12   let nothing = unit # Unit (Empty type)
13   let publicKey = base58'4dShWB22KwnfZf3P7u1ZLFxdtiYUAQaBWvZfswE2VQ9z' # ByteVector
14
```

Functions

```
52    # Argument type is declared after name
53    func add (firstNumber: Int, secondNumber: Int) = {
54        firstNumber + secondNumber
55    }
56
57    # Can also be written as one-liners (Omitting curly braces)
58    func sub (firstNumber: Int, secondNumber: Int) = firstNumber - secondNumber
59
60    # No overloading (Compilation error)
61    func add(firstNumber: Int, secondNumber: Int, thirdNumber: Int) = {
62        firstNumber + secondNumber + thirdNumber
63    }
```

Conditionals

```
59
60    # If (<conditon>) then <expression> else <expression>
61    # Else branch is always required
62    func getDACHCapital (countryCode: String) = {
63        if(countryCode == "AT") then "Vienna"
64        else if (countryCode == "CH") then "Bern"
65        else "Berlin"
66    }
67
68    # Can be used for assignments
69    let isAllowedToDrinkInUSA = if(age >= 21) then true else false
```

Math functions

pow(Int, Int, Int, Int, Int, Union): Int

Raises the number to a power.

```
pow(base: Int, bp: Int, exponent: Int, ep: Int, rp: Int, round: Union): Int
```

log(Int, Int, Int, Int, Int, Union): Int

Calculates logarithm of the number.

```
log(value: Int, ep: Int, base: Int, bp: Int, rp: Int, round: Union): Int
```

fraction(Int, Int, Int): Int

Converts arbitrarily large signed integer to integer.

```
fraction(value: Int, numerator: Int, denominator: Int): Int
```

https://docs.wavesplatform.com/en/ride/functions/built-in-functions/math-functions.html

Comments

```
15    # This is a comment
16
17    # There are no multiline comments
18    # So we have to write
19    # multiple singleline comments :)
20
```

Caveats

Ride is a **non-turing-complete** language

- No loops
- No recursion

This decision was made in order to guarantee fixed contract execution costs

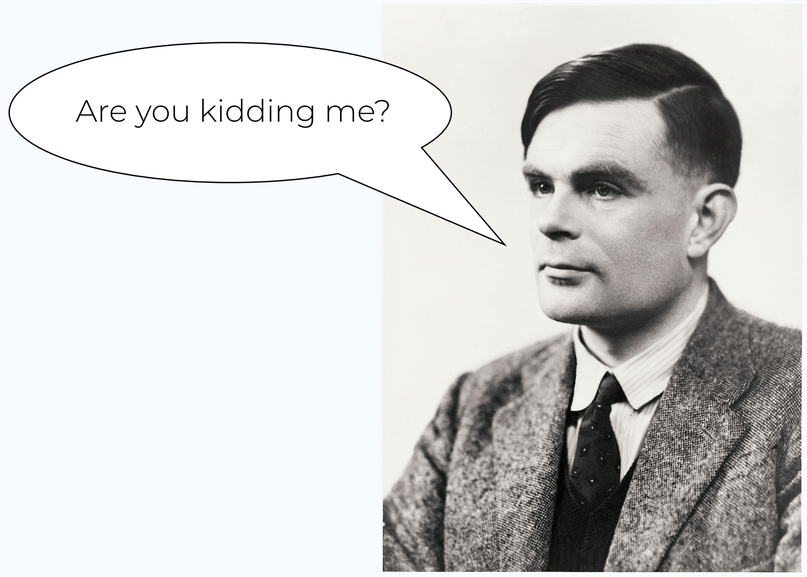
Non-turing completeness

Advantages

- **Fixed execution costs** (complexity is known before execution)
- Termination is guaranteed, which **prevents DDoS attacks**

So I can’t use loops?

There is a **pending proposal** by Ride development team lead *Ilya Smagin* , **which would enable loops** and functions leveraging looping (find, sort, etc.).



Alan Turing
https://artblart.files.wordpress.com/2012/10/alan_turing-web.jpg

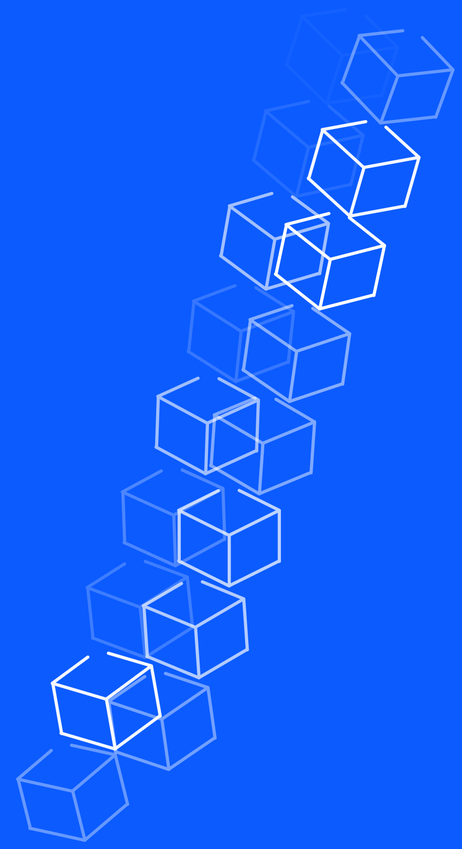
But **for now, you can only work with fixed sized arrays/lists**.

---

https://medium.com/@ilya.smagin/solution-for-loops-for-foreach-in-ride-7b5f41dc76dd

WEB 3

Quiz
RIDE

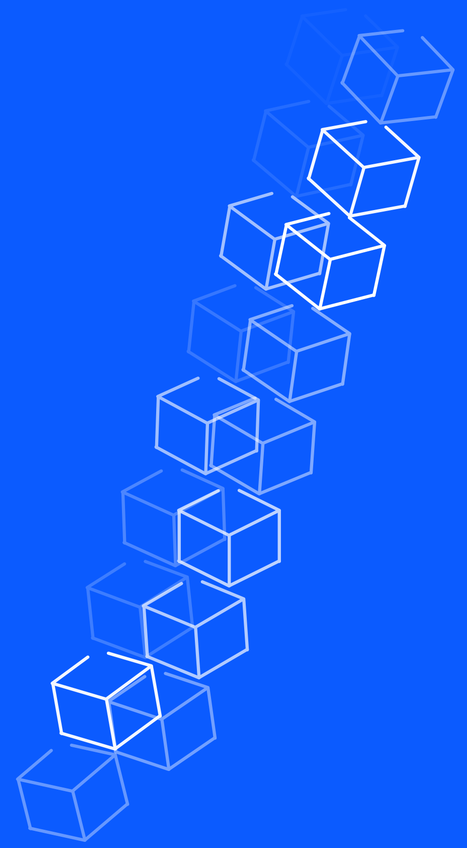


waves

Daniel Lutz / 2019

WEB 3

Let’s RIDE on Waves!



waves

Daniel Lutz / 2019

@Callable: The API of your smart contract

- Functions with an “@Callable” annotation **can be called from the outside**
- Write data and transfer funds
- Invokation is paid by contract owner

Icon made by DinosoftLabs from www.flaticon.com

Accessible data sources

- Transaction data
- Account state
- Blockchain state

---

https://docs.wavesplatform.com/en/ride/functions/callable-function.html
https://docs.wavesplatform.com/en/ride/structures/common-structures/script-result.html

Account State: The database of your smart contract

- Key/Value storage
- Each account has its own data storage
- Can be accessed in a smart contract

Supported record types

- String
- Integer
- Boolean
- ByteVector

Limitations

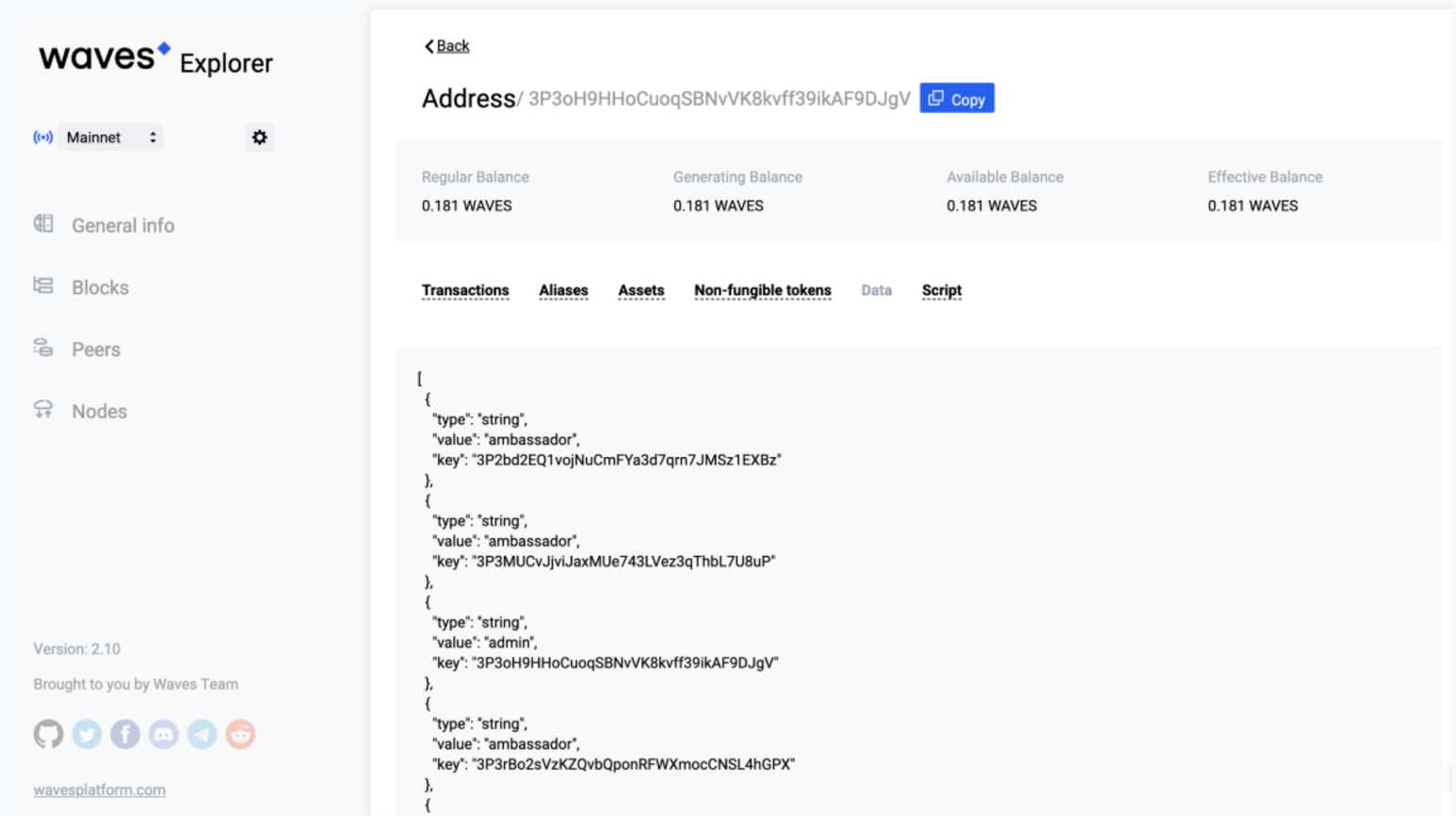
- Unlimited records
- 32 kilobytes per record (Size of Romeo and Juliet)?

Icon made by Smashicons from www.flaticon.com

---

https://docs.wavesplatform.com/en/blockchain/account/account-data-storage.html

Exploring Account state in Waves Explorer



By clicking on the “Data” tab, you can view the current account state of a Waves account.

https://wavesexplorer.com/address/3P3oH9HHoCuoqSBNvVK8kvff39ikAF9DJgV

Transaction types

#	Transaction type
1	Alias transaction
2	Burn transaction
3	Data transaction
4	Exchange transaction
5	Genesis transaction
6	Invoke script transaction
7	Issue transaction
8	Lease cancel transaction
9	Lease transaction
10	Mass transfer transaction
11	Reissue transaction
12	Set asset script transaction
13	Set script transaction
14	Sponsorship transaction
15	Transfer transaction

SetScriptTx

Deploys smart contracts

InvokeScriptTx

Invokes @Callable functions

TransferTx*

Transfers funds from A to B

DataTx*

Writes to account state

** Can be invoked by a smart contract*

https://docs.wavesplatform.com/en/blockchain/transaction-type.html

DataTx - The CRUD interface for Account State

- Creates or updates key/value records

- Up to 100 data entries per contract invokation

Fee

0.001 $WAVES per kilobyte / tx

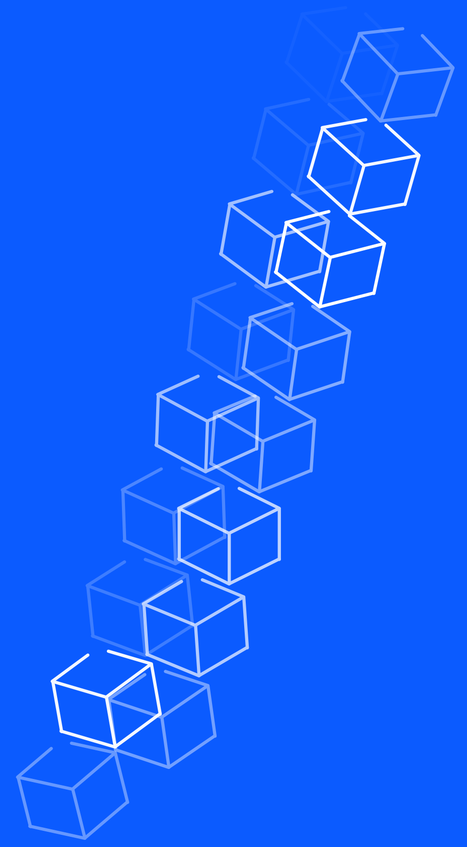
```
104   @Callable(invocation)
105   func writeDataToAccountState () = {
106       WriteSet(
107           [
108               DataEntry("name", "Alice"),
109               DataEntry("age", 1),
110               DataEntry("isRidingOnWaves", true)
111               # ...
112               ]
113           )
114   }
```

---

https://docs.wavesplatform.com/en/blockchain/transaction-type/data-transaction.html
* Rounded up to nearest thousandths

WEB 3

Exercise
Notepad

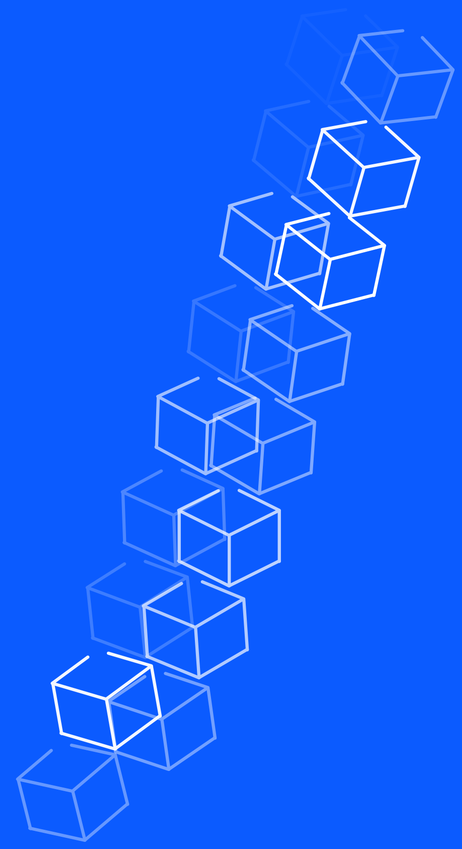


waves

Daniel Lutz / 2019

WEB 3

Quiz
Account State

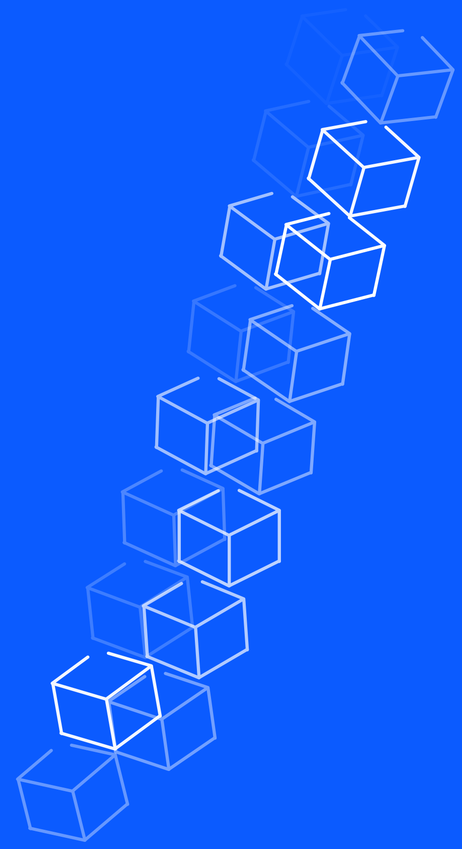


Daniel Lutz / 2019

waves

WEB 3

Quiz
Account State



Daniel Lutz / 2019

waves

TransferTx - Send funds from your smart contract

- Transfer $WAVES or custom tokens

- Up to 10 transfers per invokation

Fee

0.001 $WAVES / tx

```
@Callable(invokation)
func sendFundsToCaller () = {
    let callerAddress = invokation.caller
    let amountWavelets =  1 * 100000000 # 10^8 Wavelets equal 1 $WAVES

    TransferSet(
        [
            ScriptTransfer(invokation.caller, amountWavelets, unit)
            # ...
        ]
    )
}
```

https://docs.wavesplatform.com/en/blockchain/transaction-type/transfer-transaction.html

Online IDE is great for starting, but are there any better tools?

Development Tools

Editors

- Online IDE
- VSCode (Plugin)

Clients

- HTTP API
- Libraries for 9 languages

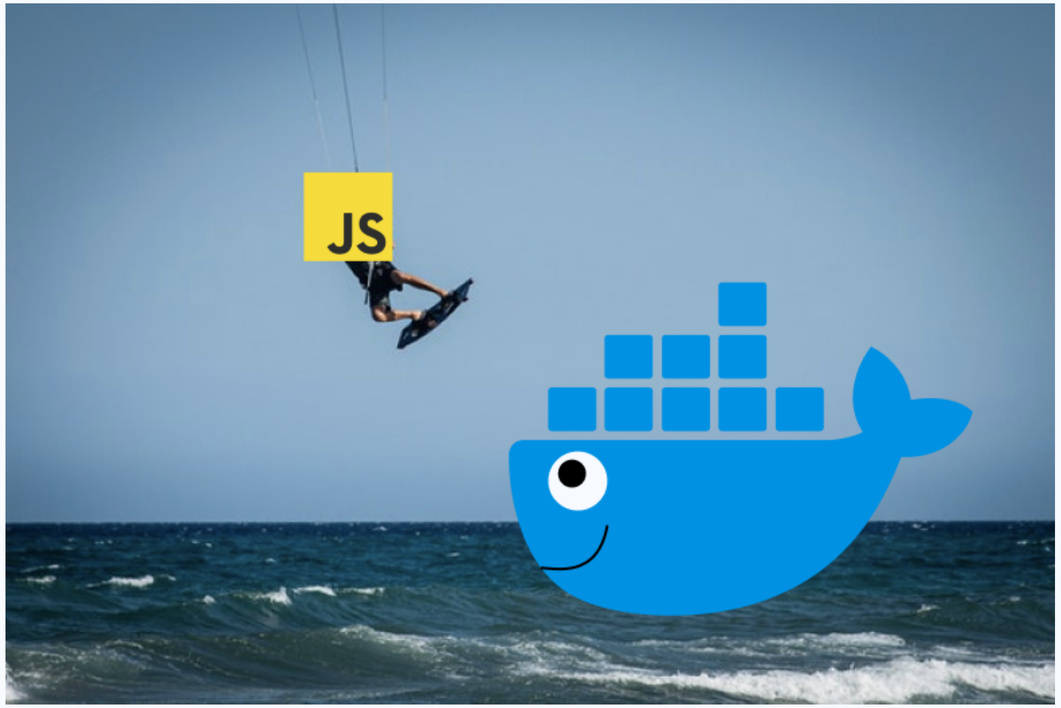
Virtualization/Hosting

- Docker
- Microsoft Azure

https://ide.wavesplatform.com
https://marketplace.visualstudio.com/items?itemName=wavesplatform.waves-ride
https://azure.microsoft.com/en-us/blog/waves-platform-azure-blockchain/

Surfboard: *RIDE on Waves* like a pro

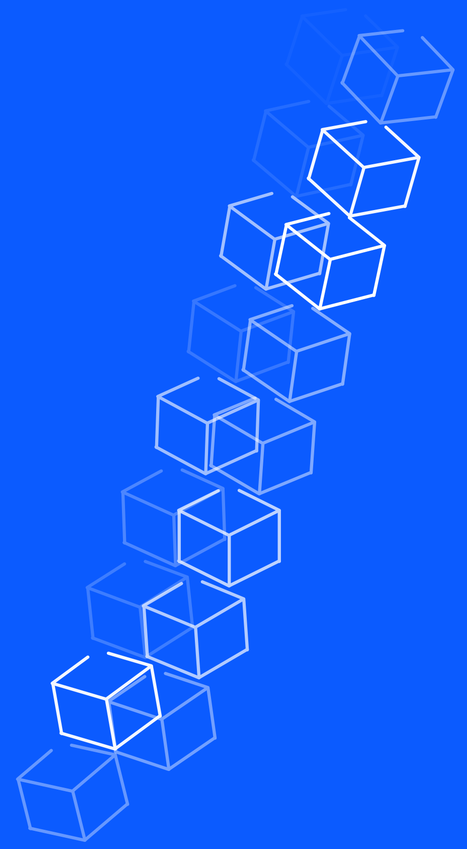
- Supports **local development**
- **Automated testing** with Mocha
- Run your **local blockchain using Docker**
- **REPL** interface to play around & explore language



https://github.com/wavesplatform/surfboard/

WEB 3

Exercise
Multi User Wallet

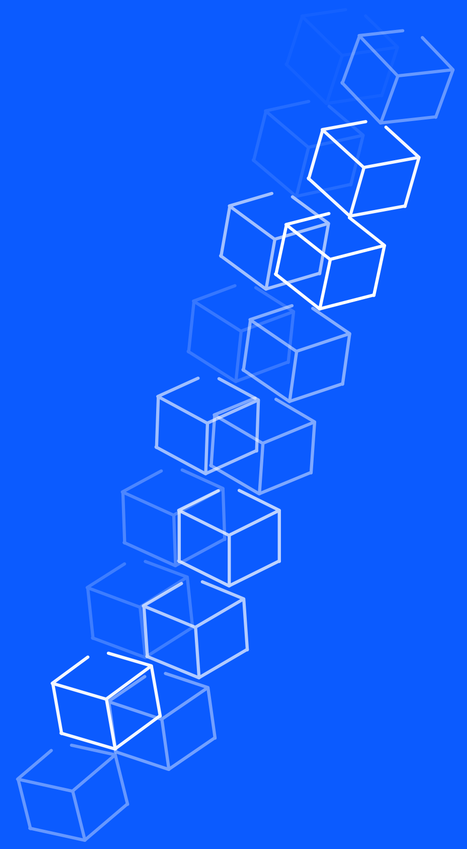


waves

Daniel Lutz / 2019

WEB 3

Quiz
@Callable

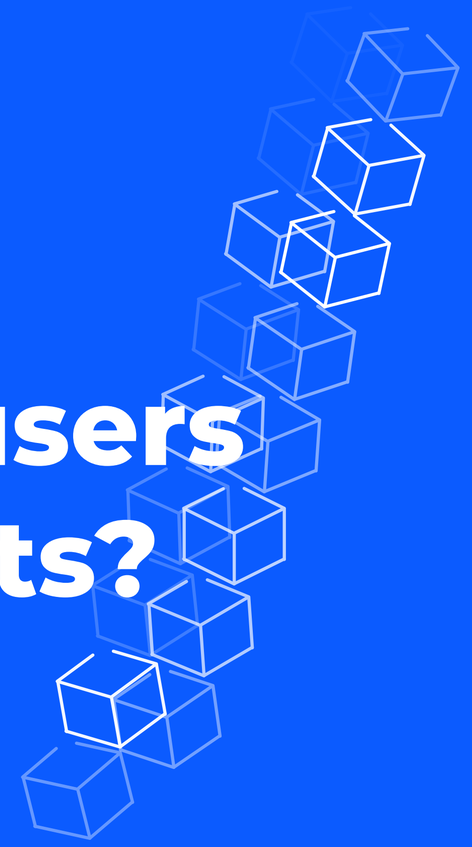


waves

Daniel Lutz / 2019

WEB 3

How can we make it easy for users to interact with smart contracts?



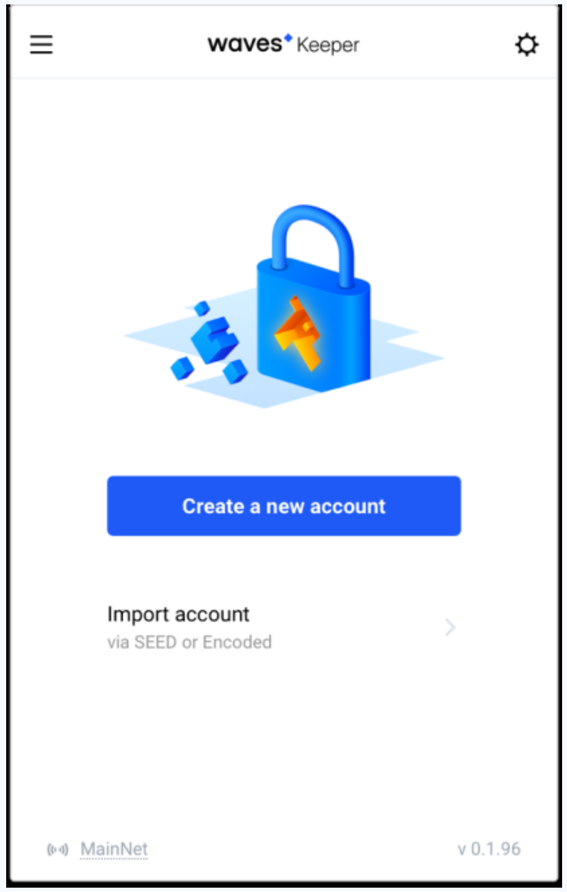
waves

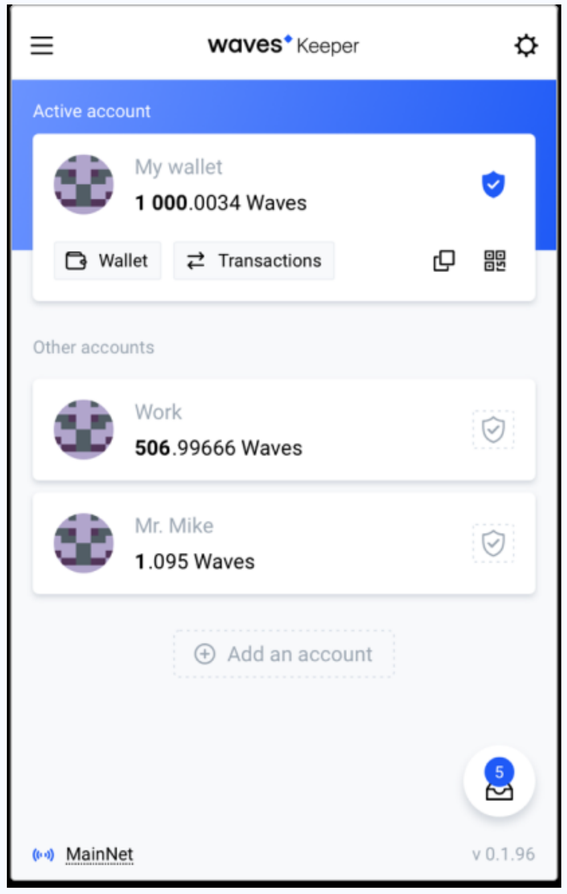
Daniel Lutz / 2019

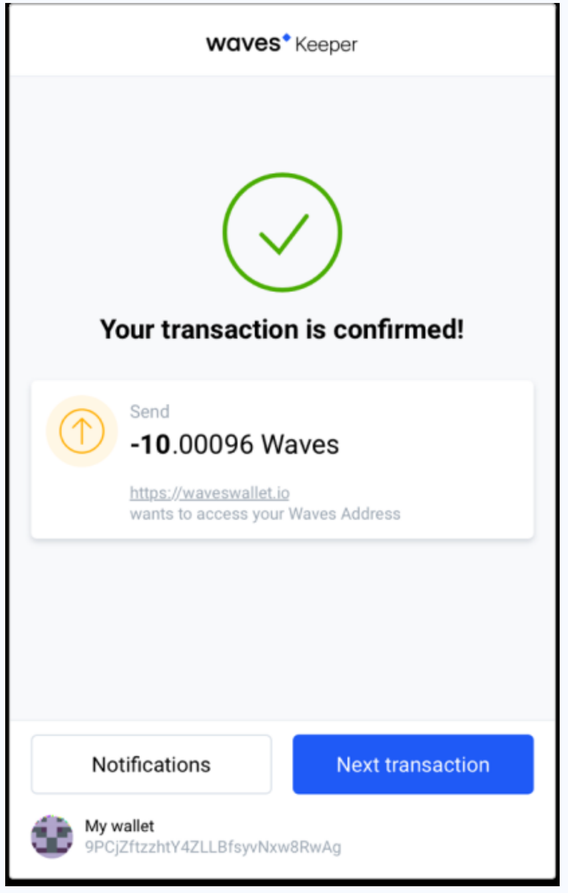
Waves Keeper - Browser extension

- **Secure and seamless** smart contract interaction
- Encrypted account **data is stored locally**
- **Testnet support** for development

https://wavesplatform.com/technology/keeper







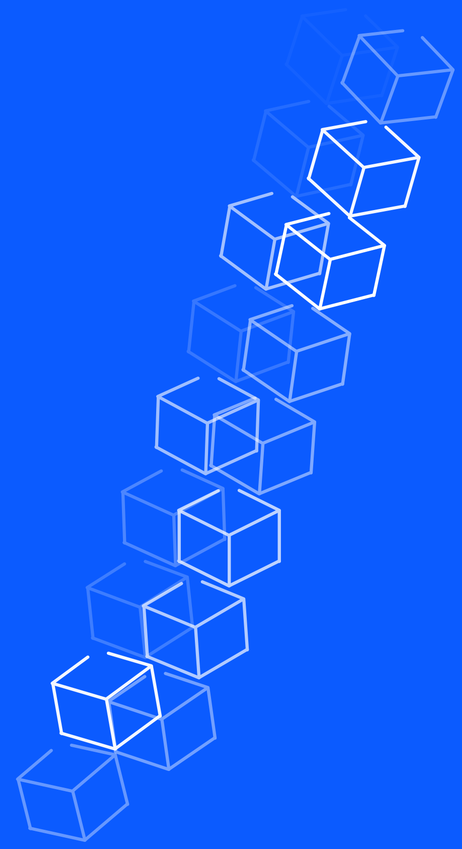
https://wavesplatform.com/technology/keeper

waves

Daniel Lutz / 2019

WEB 3

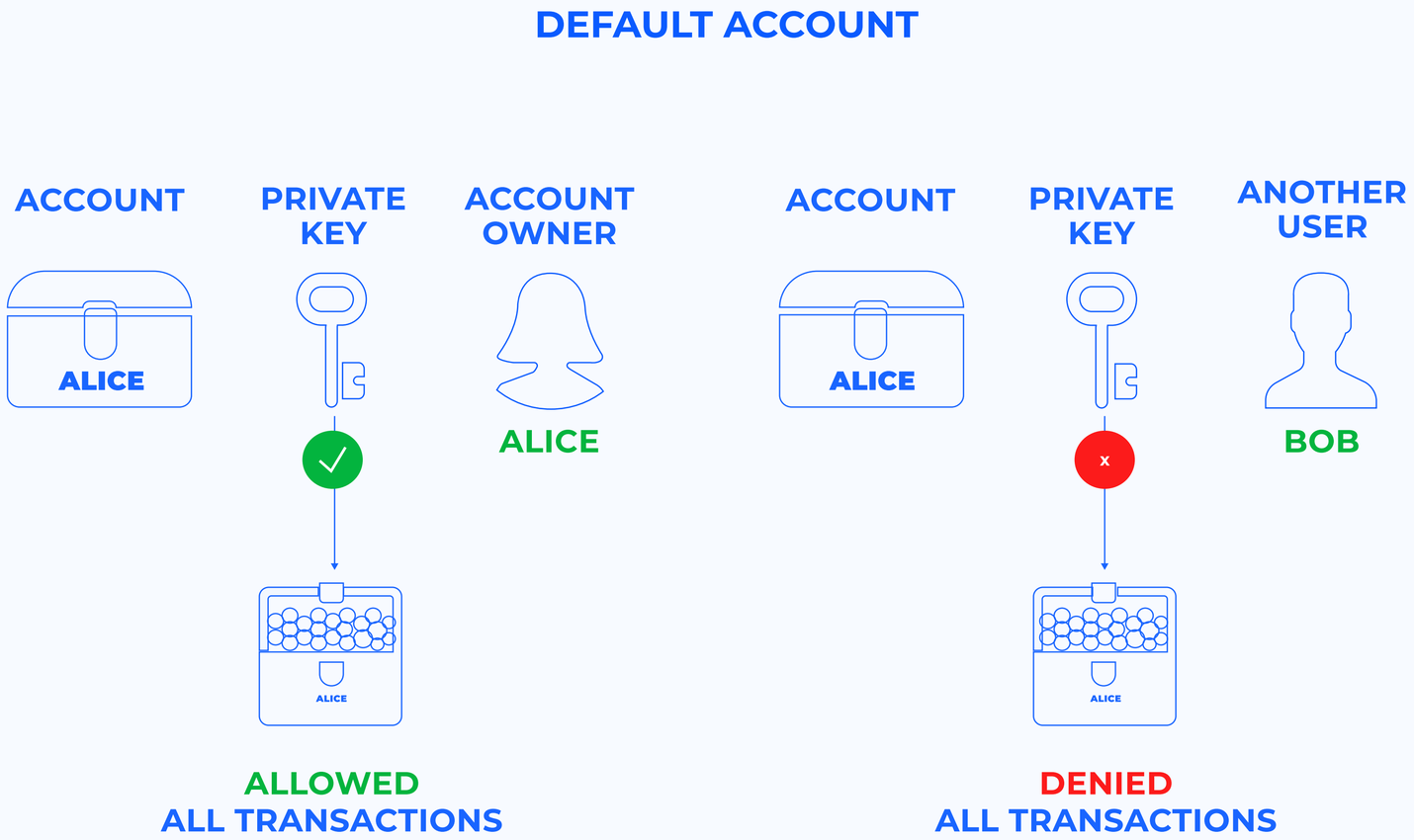
Don’t trust, verify
Building a multisignature contract



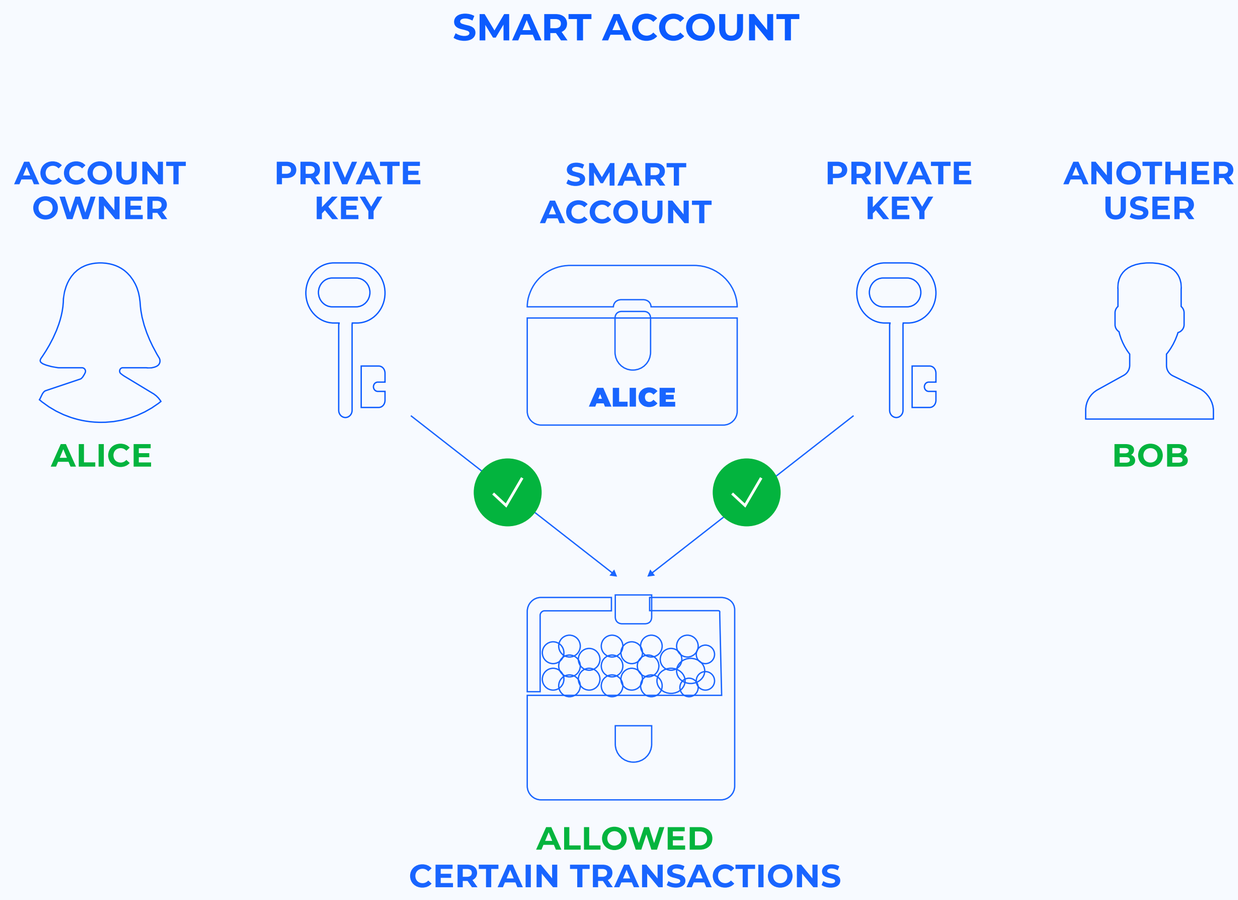
waves

Daniel Lutz / 2019

Multisig?


DEFAULT ACCOUNT
ACCOUNT
PRIVATE KEY
ACCOUNT OWNER
ALICE
ALICE
ALICE
ALLOWED
ALL TRANSACTIONS
ACCOUNT
PRIVATE KEY
ANOTHER USER
ALICE
x
BOB
ALICE
DENIED
ALL TRANSACTIONS

Multisig!


SMART ACCOUNT
ACCOUNT OWNER
PRIVATE KEY
SMART ACCOUNT
PRIVATE KEY
ANOTHER USER
ALICE
ALICE
BOB
ALLOWED
CERTAIN TRANSACTIONS

When to use multisignature?

- Limit trust

- Split responsibility accross multiple parties

Examples

- Company funds

- Inheritance

- Private key distribution for OpSec

Icon made by Smashicons from www.flaticon.com

---

https://docs.wavesplatform.com/en/ride/functions/verifier-function.html
https://docs.wavesplatform.com/en/blockchain/transaction/transaction-validation.html

@Verifier

- Verifies **outgoing transactions***
- Useful for multsig or immutability
- Pattern matching is used to allow between tx types
- Uses the **sigVerify()** helper function

```
let alicePublickey = base58'9A6J6VAeUKNSsoXW6oQVwekaMaW37wdFMgHUkaDBw84H'

@Verifier(tx)
func verify() = {
    let isSignedByAlice = sigVerify(tx.bodyBytes, tx.proofs[0], alicePublickey)

    match tx {
        case _ => isSignedByAlice
    }
}
```

https://docs.wavesplatform.com/en/ride/functions/verifier-function.html
https://docs.wavesplatform.com/en/blockchain/transaction/transaction-validation.html

sigVerify()

- Compares signatures to public keys

Parameters

1. *Message*: The signed transaction
2. *Signature*: The signature of the signer
3. *PublicKey*: The publicKey of the signer

sigVerify(ByteVector, ByteVector, ByteVector): Boolean

Checks that the Curve25519 digital signature is valid, i.e. it was created by the owner of the public key.

```
sigVerify(message: ByteVector, sig: ByteVector, pub: ByteVector): Boolean
```

Parameters

message : ByteVector

Signed data.

sig : ByteVectore

Digital signature.

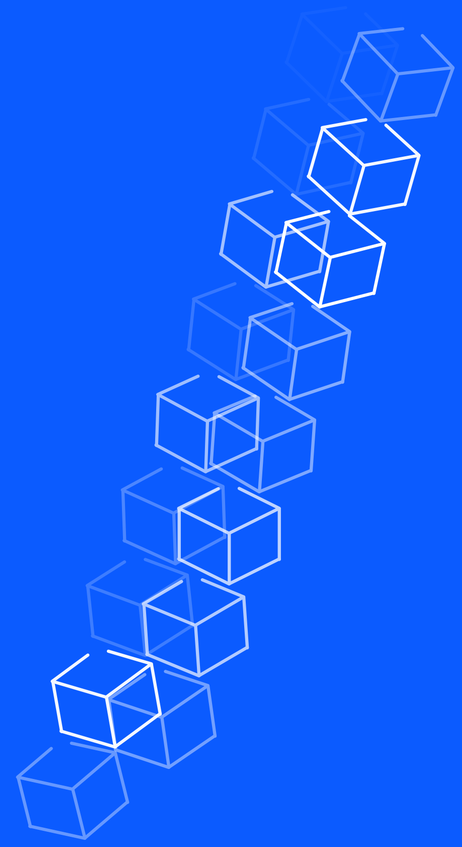
pub : ByteVectore

Public key.

https://docs.wavesplatform.com/en/ride/functions/built-in-functions/verification-functions.html#sig-verify

WEB 3

Exercise
Multisig Wallet

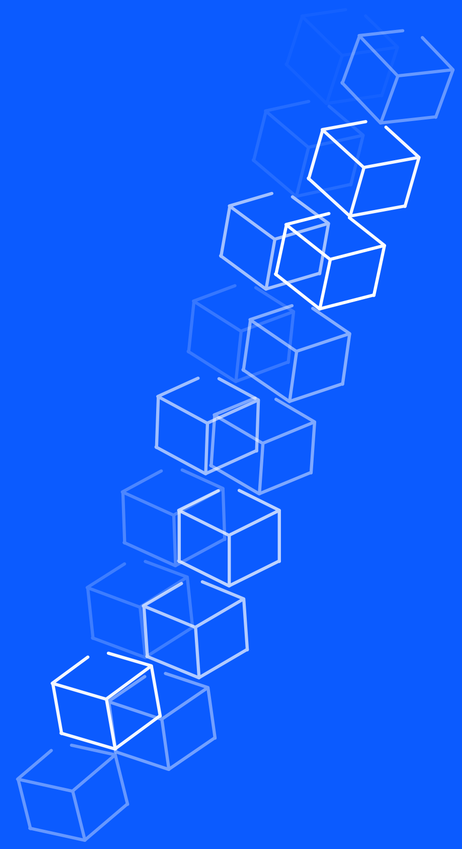


waves

Daniel Lutz / 2019

WEB 3

Quiz
@Verifier

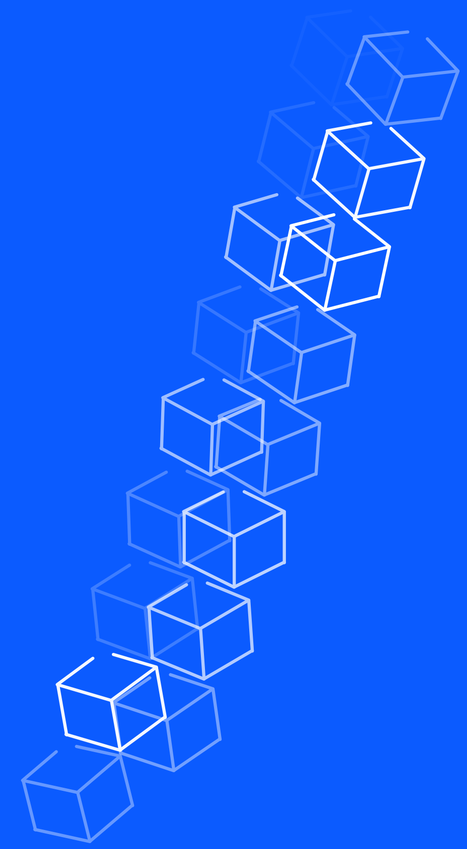


waves

Daniel Lutz / 2019

WEB 3

Challenge
Smart Contract Hackathon



waves

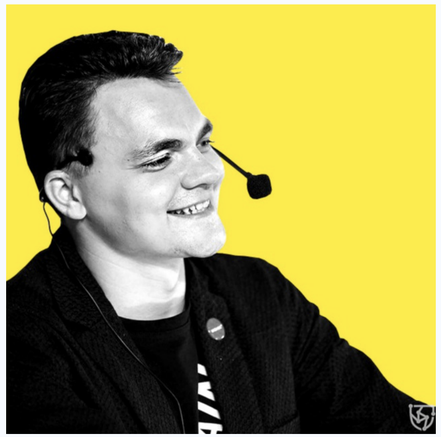
Daniel Lutz / 2019

Mastering Web3

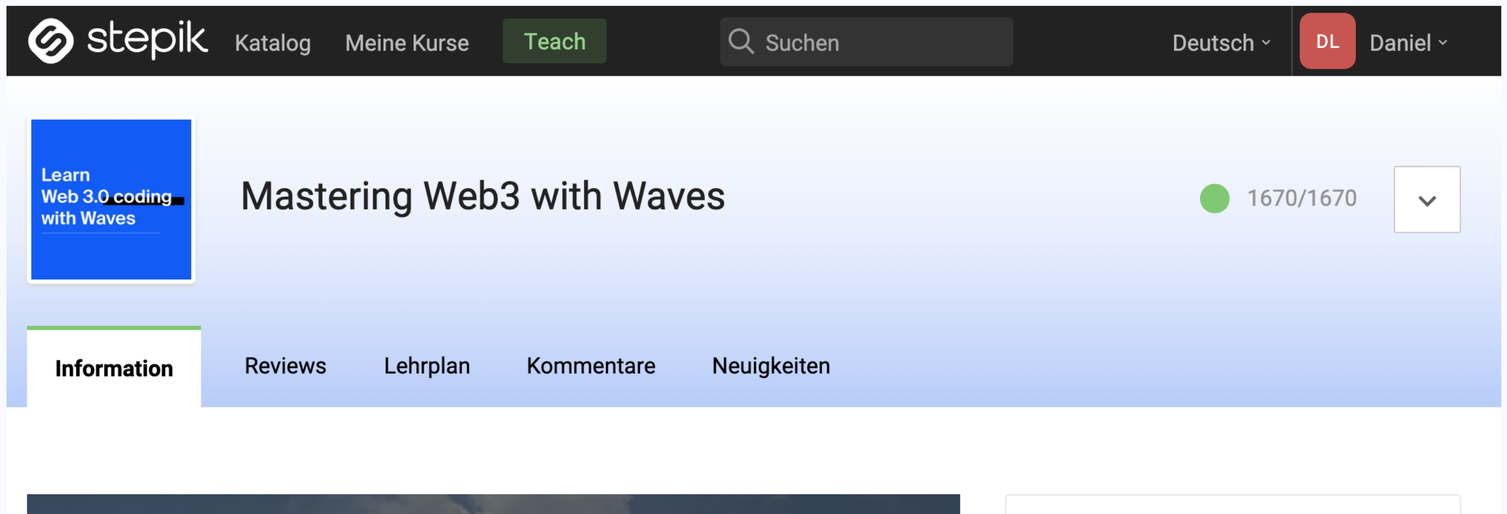
- Online course

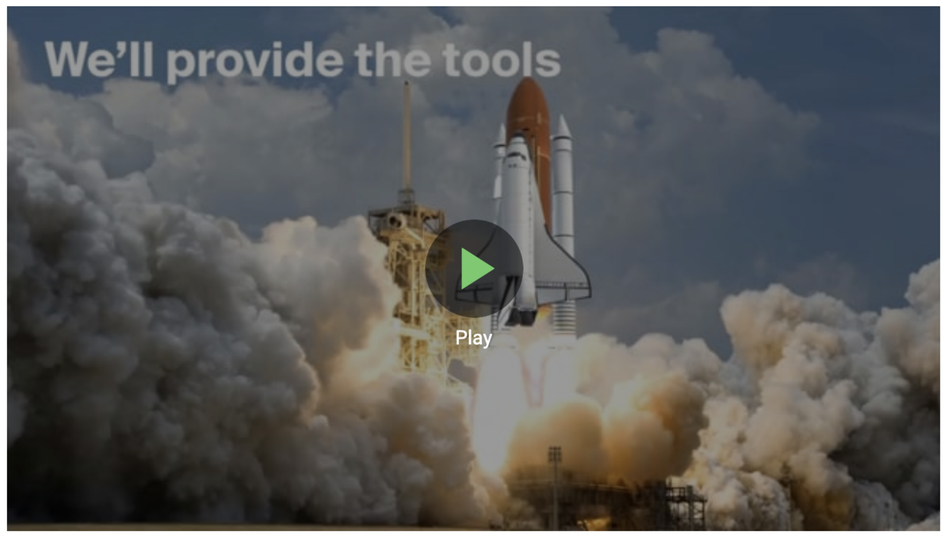
- Build a eCommerce dapp

https://stepik.org/course/54415



@AlekseiPupyshev





4.5 ★★★★½ All reviews

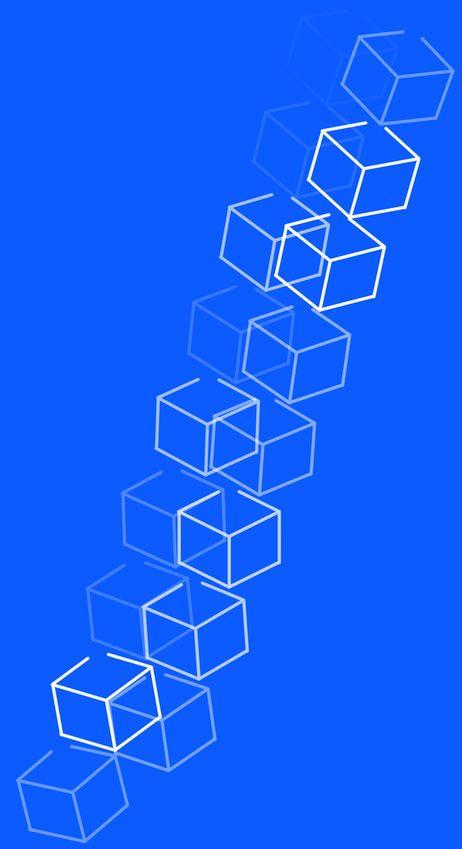
The dApps programming online course aims to prepare developers for the technologies that will become the backbone of the next-generation Web 3.0 Padawans chat (offtopic and projects/ideas discussions) https://t.me/joinchat/AkSt-010qQHYhgcgBeT9Lg Jedi chat (more technical discussions) https://t.me/joinchat/AkSt-1Rd0FAl6-vD45G5OQ

waves

Daniel Lutz / 2019

WEB 3

Network is power



Daniel Lutz / 2019

waves

Let’s connect!

https://t.me/waves_ride_dapps_dev

https://t.me/wavesnews

YouTube

@wavesdapp

@samgin101 (Ilya Smagin)

@ikardanoff (Inal Kardanov)

@AlekseiPupyshev

@sasha35625 (Sasha Ivanov)