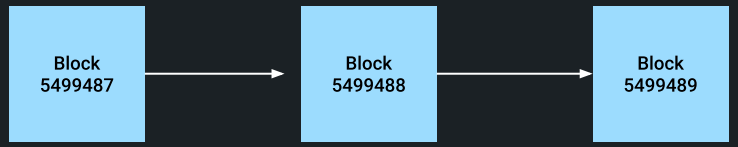

# Crypto beyond currencies

Building decentralized applications on Ethereum

# What is a blockchain?

An indelible [public] distributed database

# How it works ?




- The database is composed by a chain of **blocks**
- Each block introduces new changes to the previous one
- Changes are broadcasted as **transactions** (which require a fee)
- New blocks are added every X seconds
- Blocks are considered final after Y new blocks (**confirmations**)

# Network

- Anyone can set up a **node** (full or light)
- Nodes replicate the blockchain data
- Specialized nodes (**miners**) add new blocks

# This means...

- Anyone can independently query the current state of the database
- Changes take time (and money) but are irreversible once confirmed
- State changes can be broadcasted by anyone in the network

# Why blockchain?

Trustlessness

**Why blockchain?**

Trustlessness

**Why not?**

Performance

# Do you really need a blockchain?

*Probably not, but...*

# Well-known blockchains

*What shall we build first with ~~our new hammer~~*
*a decentralized public ledger?*

# Bitcoin

First blockchain is a currency:
distributed database of
**address => balance**

*This is not entirely true, as Bitcoin tracks unspent transaction outputs; account “balances” is just an abstraction*

# Ethereum

Store programs instead of plain data:
every program has its own state and code

# Smart contracts

- Not a very good name
- Contracts receive transactions, execute code, and update their state
- Contracts also expose getters for querying their state
- An “entry” in Ethereum can be a user address or a smart contract

# Well-known contracts

*What shall we build first with ~~our new hammer~~*
*a Turing-complete decentralized network?*

# Tokens

Smart contract that keeps track of the balances of a custom token

```
interface ERC20 {

  function transfer(address to, uint256 value) public;

  function balanceOf(address owner) public returns (uint256);

  event Transfer(address from, address to, uint256 value);
}
```

*This is a simplified version of the ERC20 interface, some methods have been omitted on purpose*

```
contract ERC20Token is ERC20 {
  mapping(address => uint256) balances;

  function transfer(address to, uint256 value) public {
    require(value <= balances[msg.sender]);
    balances[msg.sender] -= value;
    balances[to] += value;
    emit Transfer(msg.sender, _to, _value);
  }

  function balanceOf(address owner) public returns (uint256) {
    return balances[owner];
  }
}
```

# Why tokens?

# Decentralized organizations

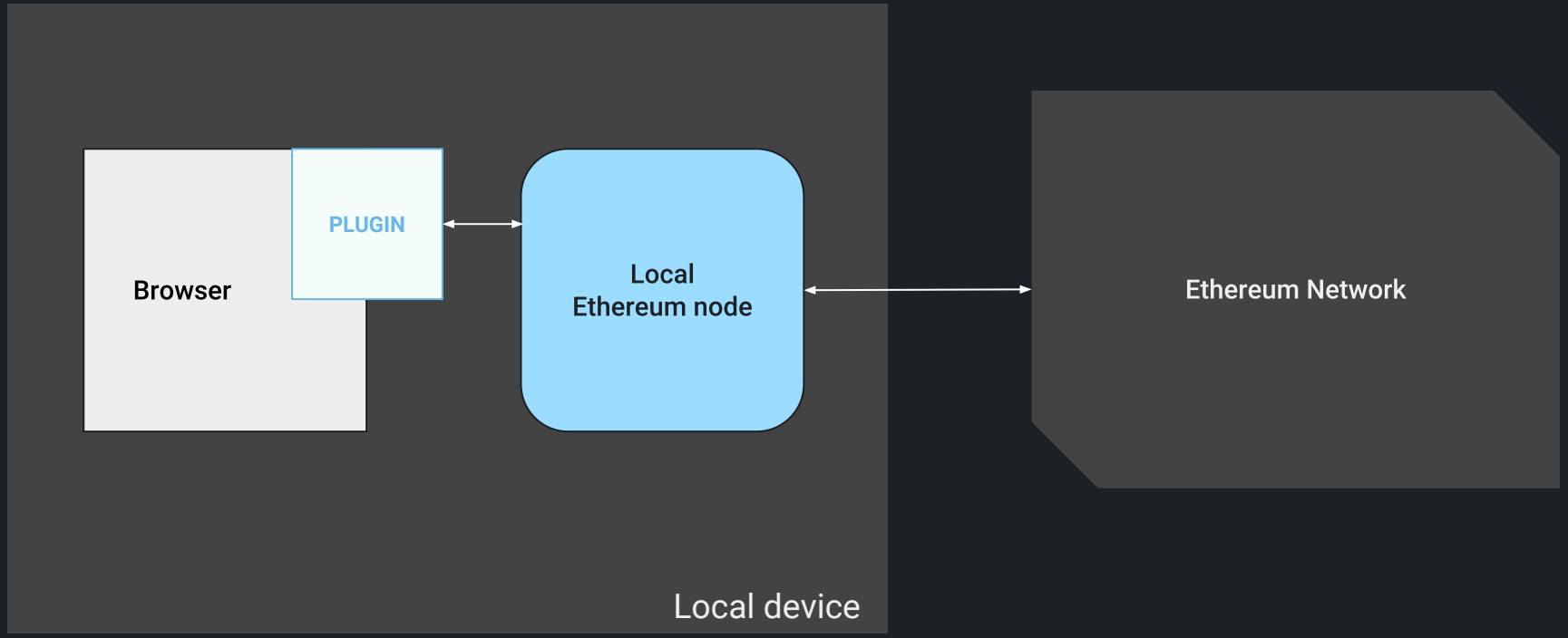
Tokens represent voting power within a transparent organization.

# Decentralized prediction markets

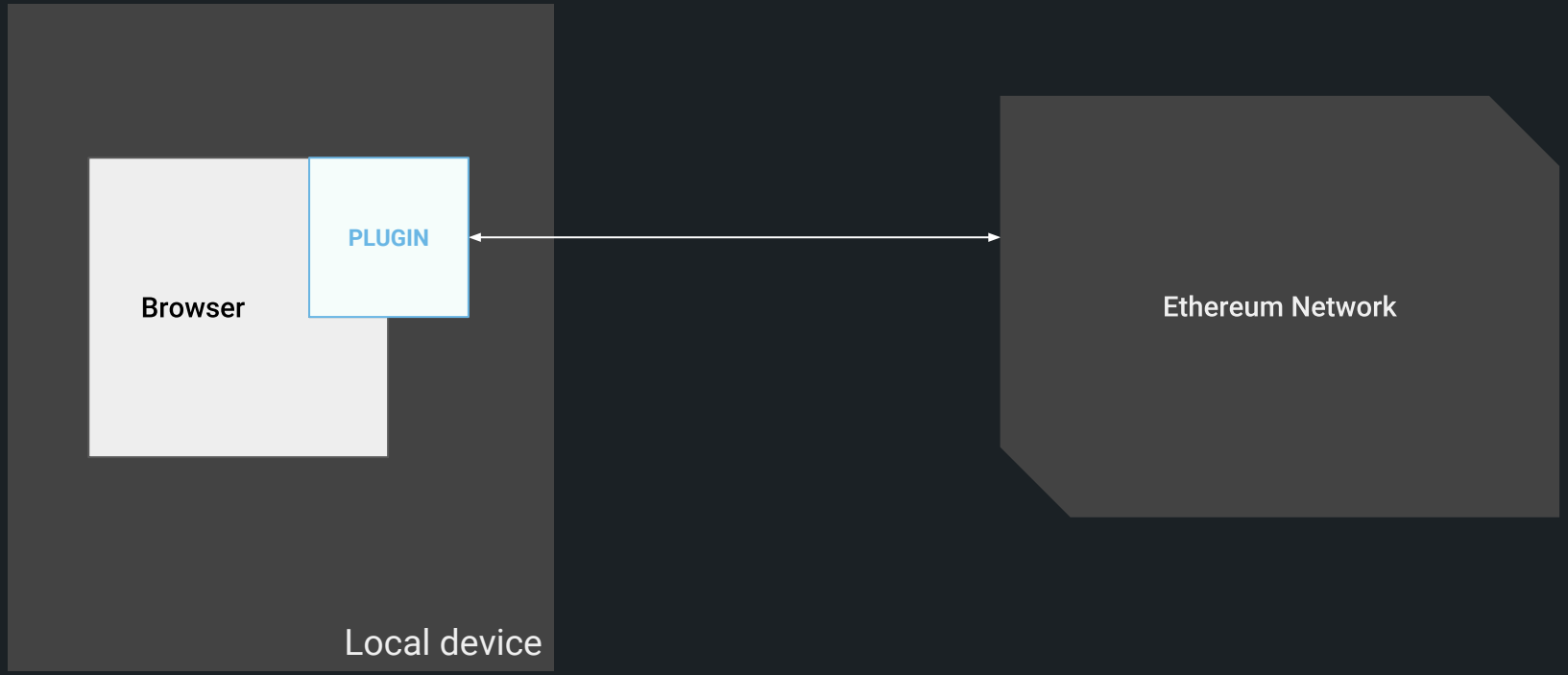
Tokens represent a tradeable bet towards a future event.
The value of the token for each outcome is an indicator of the likelihood of that result.

# User experience

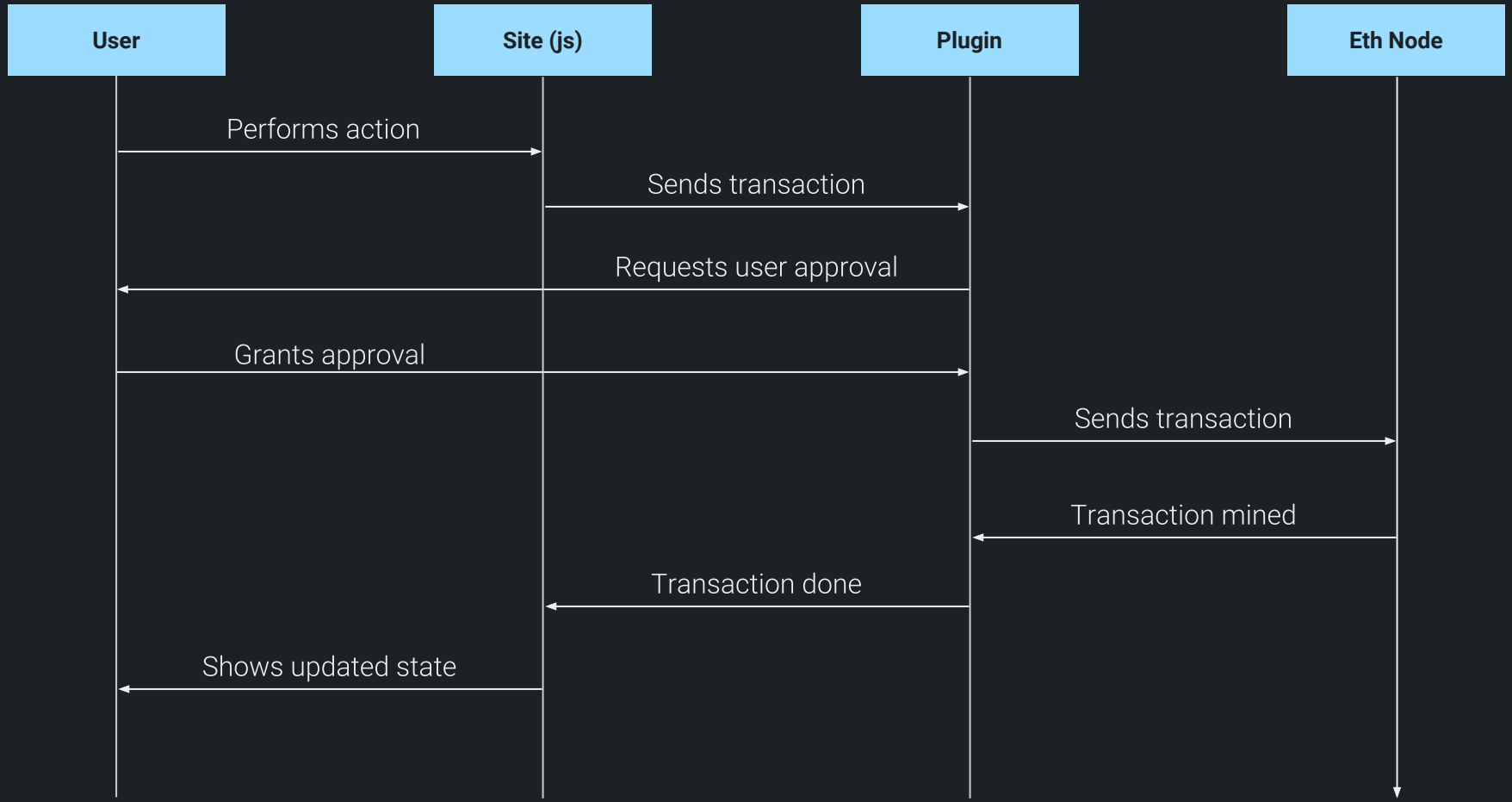
Interacting with Dapps
(decentralized applications)




# **Architecture** - Local node




# **Architecture** - Remote node




# **Flow** - Send transaction

# Show me the code

```
const token = ERC20Token.at(TOKEN_ADDRESS)

queryBalance() {
  const balance = await token.balanceOf(sender)
  this.setState({ balance })
}

sendTokens(to, value) {
  await token.transfer(to.address, value, { from: sender })
  this.setState(prev => { balance: prev.balance - value })
}
```

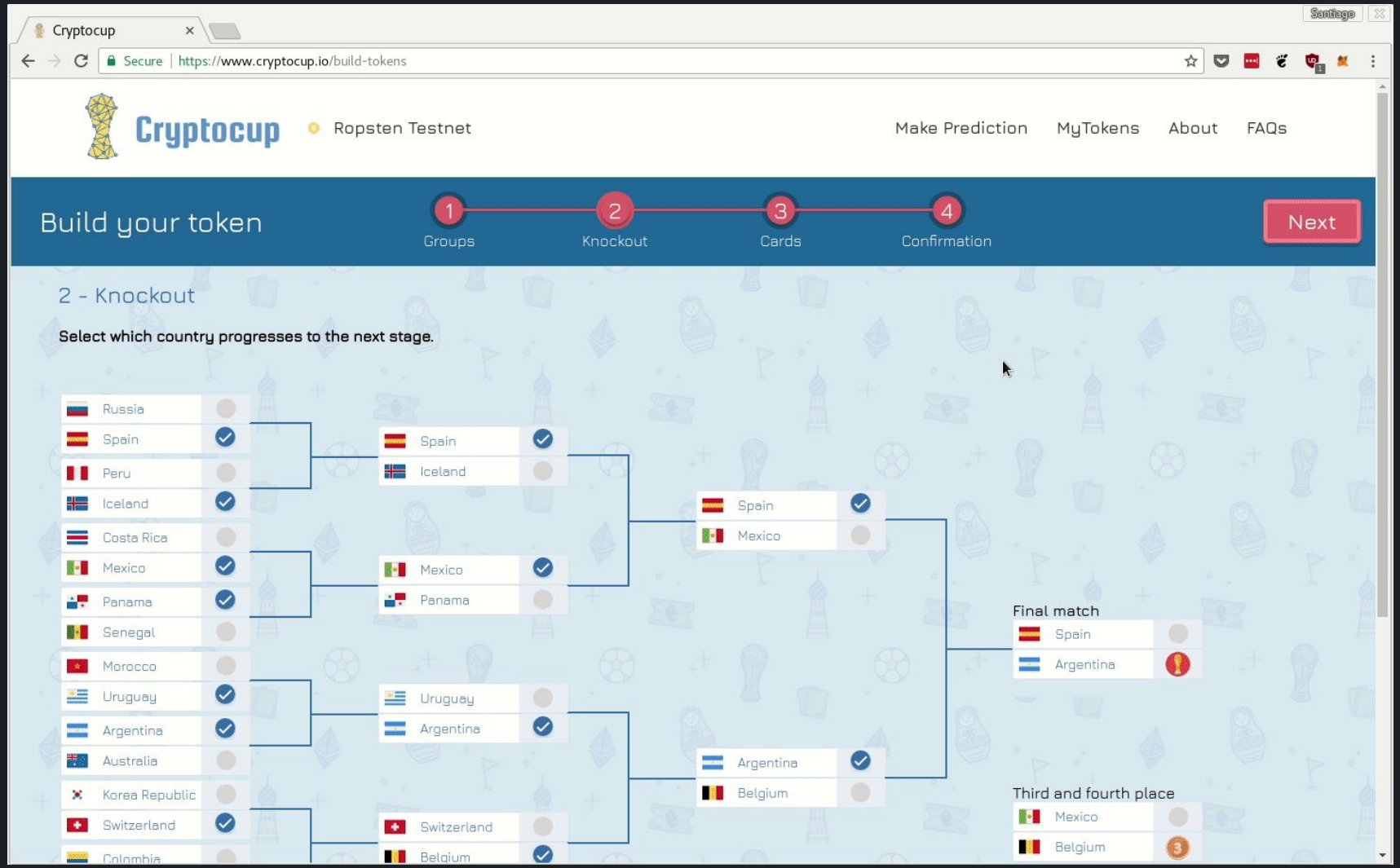
**Javascript** - Query state and send tx

```
const token = ERC20Token.at(TOKEN_ADDRESS)

watchIncoming() {
  const token = ERC20Token.at(TOKEN_ADDRESS)
  token.Transfer({ to: sender }).watch((err, event) => {
    this.setState(prev => { balance: prev.balance + event.value })
  })
}
```

**Javascript** - Watch events

# Demo time


Cryptocup
Secure | https://www.cryptocup.io/build-tokens
Cryptocup
Ropsten Testnet
Make Prediction
MyTokens
About
FAQs
Build your token
1 Groups
2 Knockout
3 Cards
4 Confirmation
Next
2 - Knockout
Select which country progresses to the next stage.
Russia
Spain
Peru
Iceland
Costa Rica
Mexico
Panama
Senegal
Morocco
Uruguay
Argentina
Australia
Korea Republic
Switzerland
Colombia
Spain
Iceland
Mexico
Panama
Uruguay
Argentina
Switzerland
Belgium
Spain
Mexico
Argentina
Belgium
Final match
Spain
Argentina
Third and fourth place
Mexico
Belgium

# Challenges: Scalability

Low network throughput (~10 tx/s)
High fees & long confirmation times


# Payment channels

Solution for instant low-fee peer-to-peer transactions
First Ethereum implementation is in adult entertainment (live cams)

# Challenges: Computation

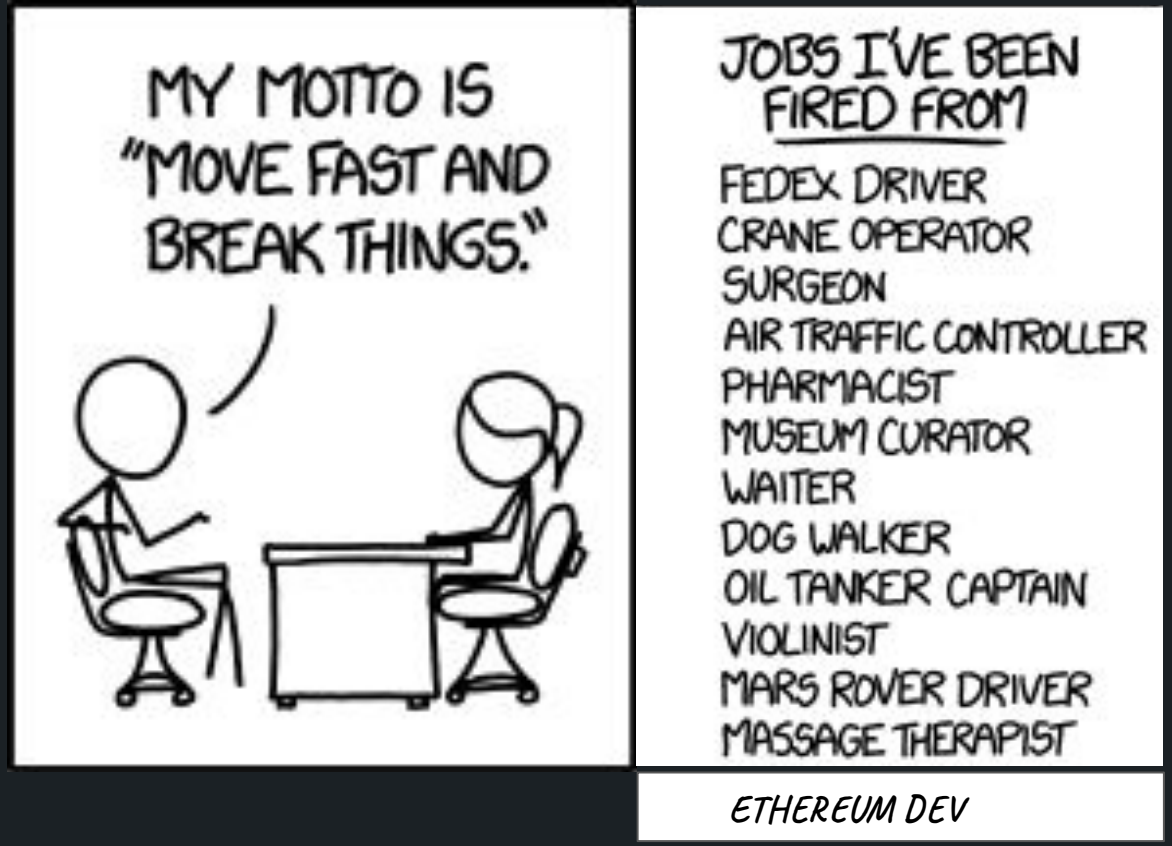
Code execution per transaction
is heavily bounded by design


# Off-chain computation

Run expensive computations outside the blockchain
and run verification games on a smart contract

# Challenges: Security

Protect immutable code that
can be executed by anyone




https://xkcd.com/1428/ (may differ from the original)

I want you to write a program that has to run in a concurrent environment under Byzantine circumstances where **any adversary can invoke your program with any arguments of their choosing.** The environment in which your program executes (and hence any direct or indirect environmental dependencies) is also under adversary control. If you make a single exploitable mistake or oversight in the implementation, or even in the logical design of the program, then either you personally or perhaps the users of your program could lose a substantial amount of money. Where your program will run, **there is no legal recourse if things go wrong.** Oh, and **once you release the first version of your program, you can never change it.** It has be right first time.

*https://blog.acolyer.org/2018/03/08/zeus-analyzing-safety-of-smart-contracts/*

# openzeppelin

< zpl.in/openzeppelin >

**Secure standard library**

Open-source repository of community reviewed
smart contracts standard library

# zeppelin_os

< zpl.in/zeppelin_os >

**Upgradeability & Security**

Library for upgradeability management and
community-contributed Kernel of standard libraries

# zeppelin

< zpl.in >

# ETHBUENOSAIRES

## HACKATHON + WORKSHOPS

< May 25-27 @ Area3 >

**Part of the ETHGlobal community & first event in LatAm**

150+ hackers, 12+ speakers

# Wrapping up

- A blockchain is a distributed database
- Blockchains trade performance for trustlessness
- Ethereum supports running code in the shape of "smart contracts"
- End-users interact with a dapp (client code + smart contracts)
- Main challenges are scalability, computation, and security

# Thank you!

## Learn more

zpl.in/openzeppelin
zpl.in/zeppelin_os

## Contact

santiago@zeppelin.solutions
@smpalladino