Introduction to Smart Contracts

What is a smart contract?

A smart contract is a computer program executed in a secure environment that directly controls digital assets

A smart contract is a **computer program** executed in a secure environment that directly controls digital assets

Example: bet on an event

```
if HAS_EVENT_X_HAPPENED() is true:
    send(party_A, 1000)
else:
    send(party B, 1000)
```

A smart contract is a computer program executed in a **secure environment** that directly controls digital assets

Secure environments can be:

- Servers run by trusted parties
- Decentralized computer network (ie. blockchains)
- Quasi-decentralized computer network (ie. private blockchains)
- Servers run by semi-trusted parties using a platform where all computation is auditable (eg. Monetas)
- Hybrid solutions combining the above (eg. state channels)

A smart contract is a computer program executed in a secure environment that **directly controls** digital assets

"Smart contracts" are to some extent a misnomer because they differ from legal contracts in one very important respect: smart contracts do not impose obligations on anyone.

Rather, they hold assets/collateral themselves.

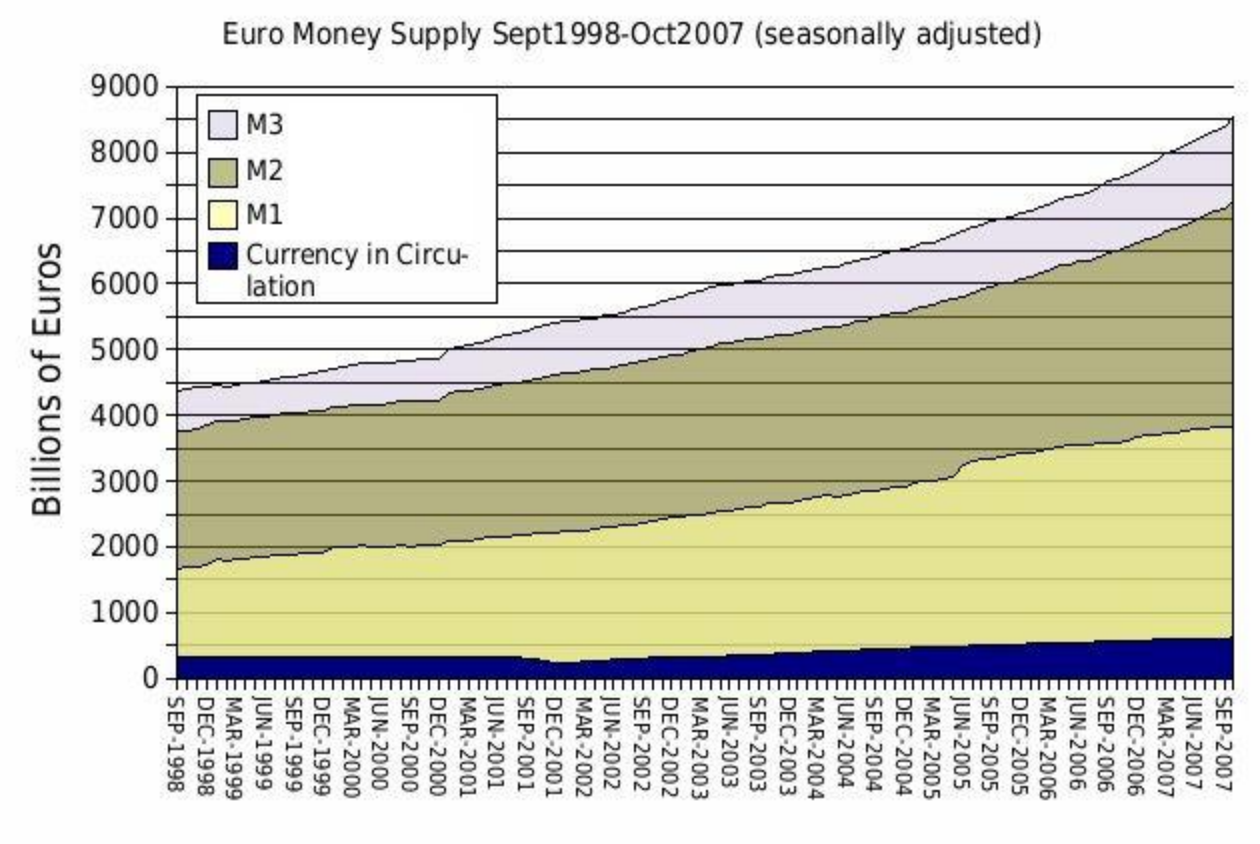
Example:

- Legal contract: “I promise to send you $1000 if X happens”
- Smart contract: “I send $1000 into a computer program which sends it to you if X happens, otherwise it eventually sends it back to me”

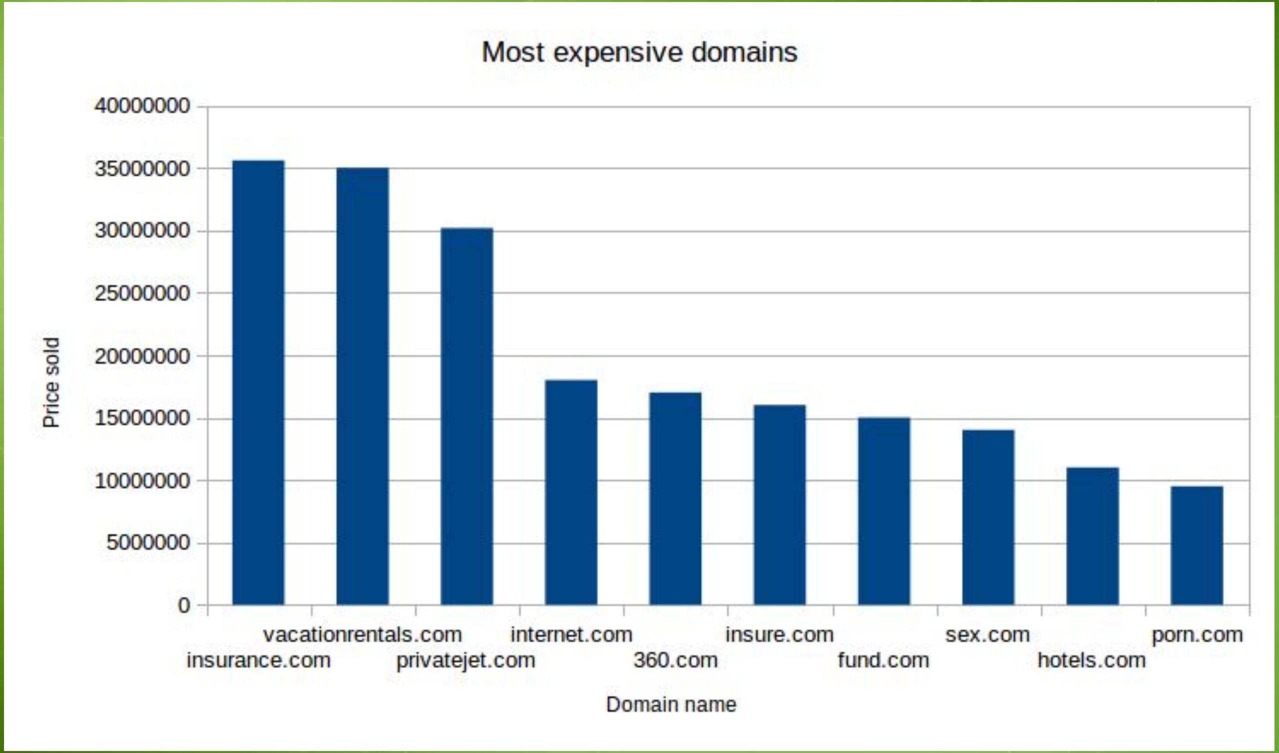
A smart contract is a computer program executed in a secure environment that directly controls **digital assets**

This is in fact a much broader category than you might think.

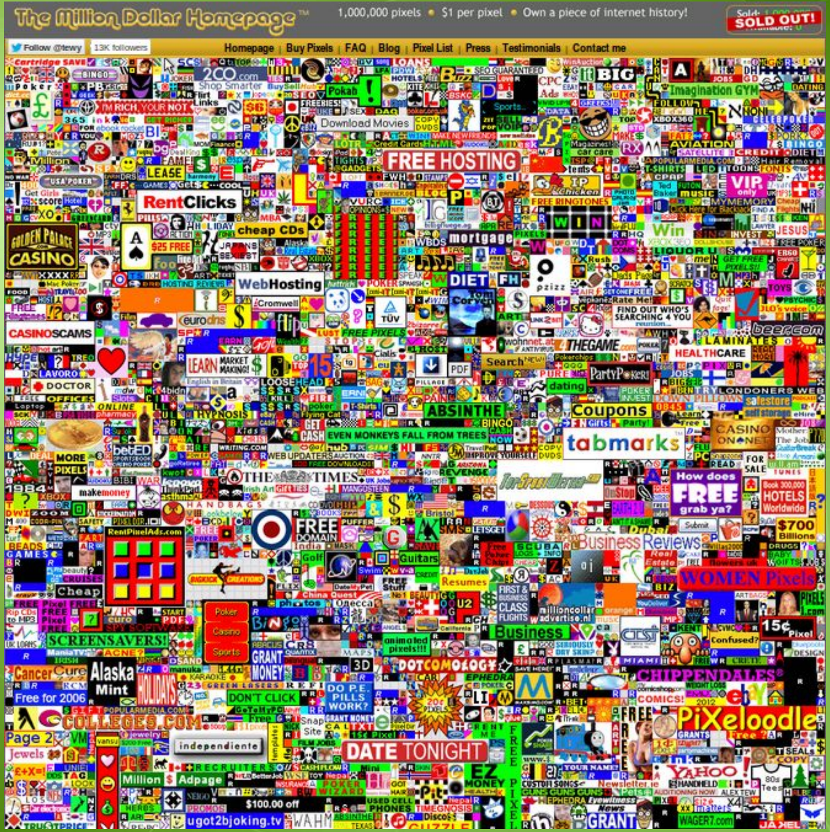
Money


Euro Money Supply Sept1998-Oct2007 (seasonally adjusted)
M3
M2
M1
Currency in Circulation
Billions of Euros
9000
8000
7000
6000
5000
4000
3000
2000
1000
0
SEP-1998
DEC-1998
MAR-1999
JUN-1999
SEP-1999
DEC-1999
MAR-2000
JUN-2000
SEP-2000
DEC-2000
MAR-2001
JUN-2001
SEP-2001
DEC-2001
MAR-2002
JUN-2002
SEP-2002
DEC-2002
MAR-2003
JUN-2003
SEP-2003
DEC-2003
MAR-2004
JUN-2004
SEP-2004
DEC-2004
MAR-2005
JUN-2005
SEP-2005
DEC-2005
MAR-2006
JUN-2006
SEP-2006
DEC-2006
MAR-2007
JUN-2007
SEP-2007

Domain names



Million dollar homepage



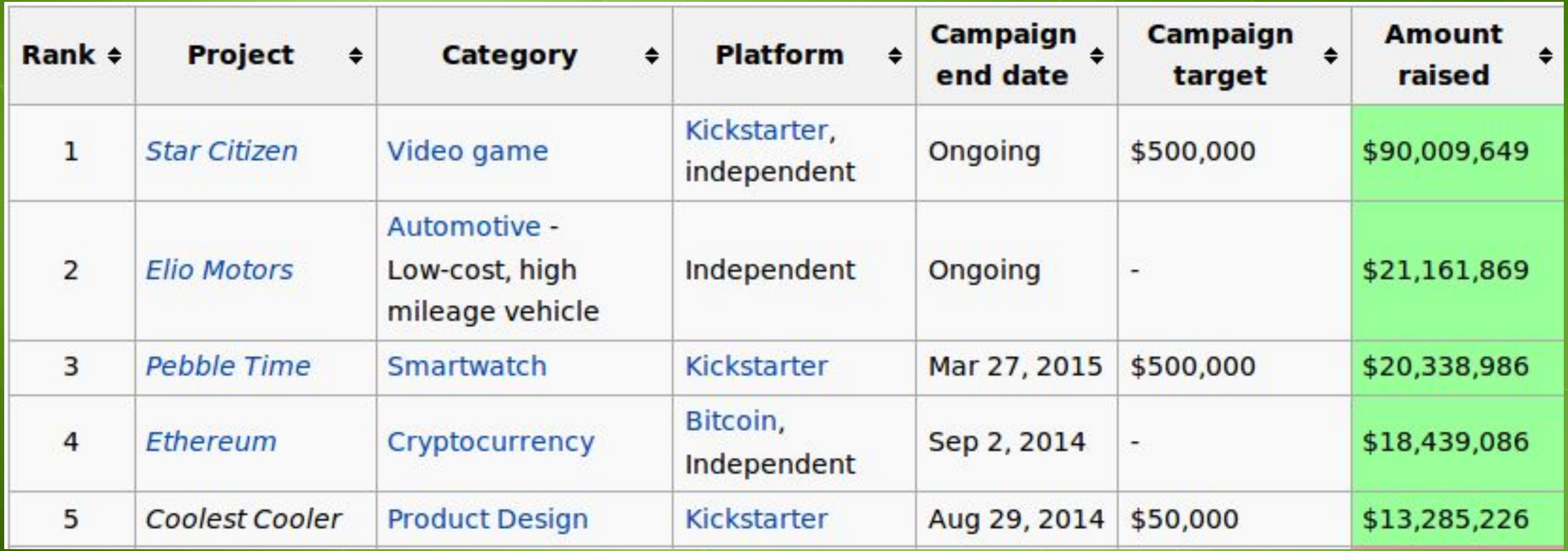
The Million Dollar Homepage™ 1,000,000 pixels • $1 per pixel • Own a piece of internet history!

SOLD OUT!

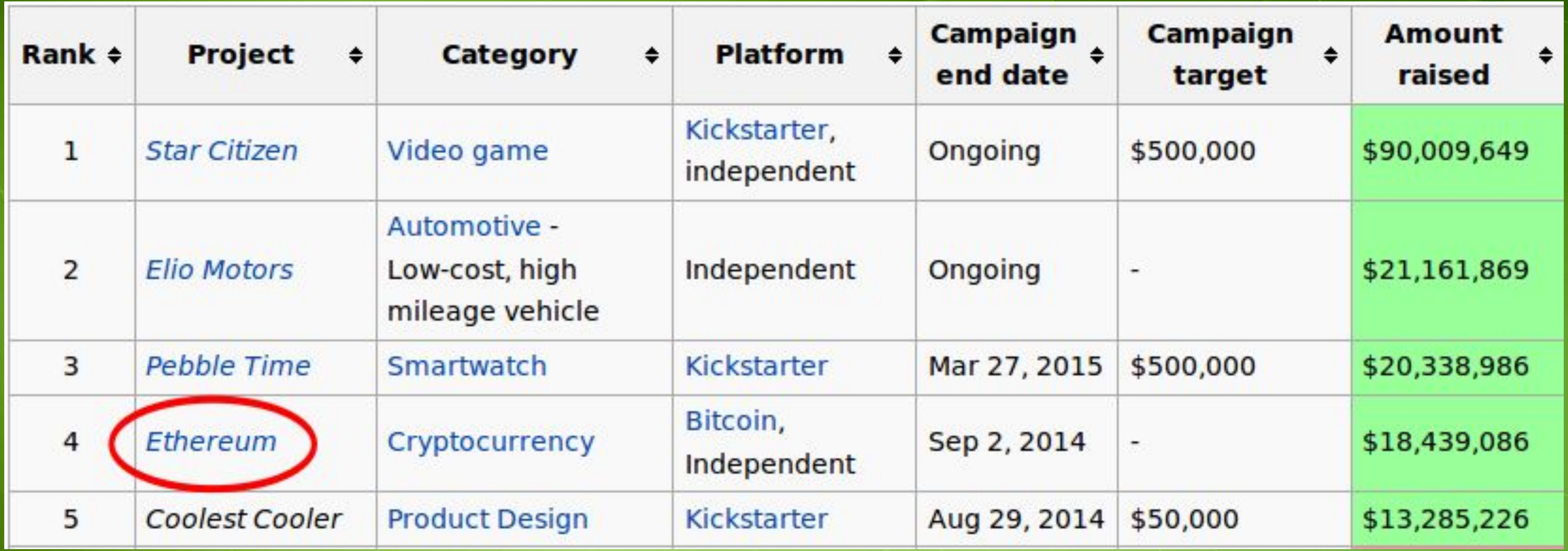
Follow @tewy 13K followers

Homepage | Buy Pixels | FAQ | Blog | Pixel List | Press | Testimonials | Contact me

Top 5 crowdfunding campaigns in history

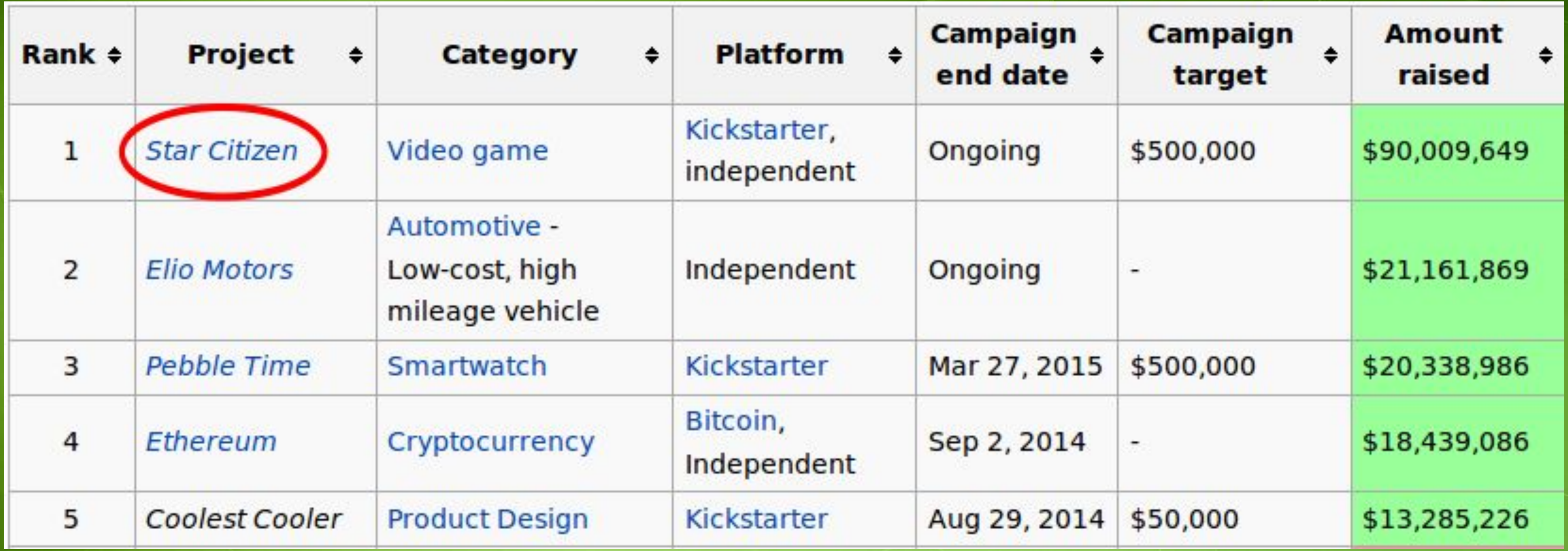


Rank	Project	Category	Platform	Campaign end date	Campaign target	Amount raised
1	*Star Citizen*	Video game	Kickstarter, independent	Ongoing	$500,000	$90,009,649
2	*Elio Motors*	Automotive - Low-cost, high mileage vehicle	Independent	Ongoing	-	$21,161,869
3	*Pebble Time*	Smartwatch	Kickstarter	Mar 27, 2015	$500,000	$20,338,986
4	*Ethereum*	Cryptocurrency	Bitcoin, Independent	Sep 2, 2014	-	$18,439,086
5	*Coolest Cooler*	Product Design	Kickstarter	Aug 29, 2014	$50,000	$13,285,226



Rank	Project	Category	Platform	Campaign end date	Campaign target	Amount raised
1	*Star Citizen*	Video game	Kickstarter, independent	Ongoing	$500,000	$90,009,649
2	*Elio Motors*	Automotive - Low-cost, high mileage vehicle	Independent	Ongoing	-	$21,161,869
3	*Pebble Time*	Smartwatch	Kickstarter	Mar 27, 2015	$500,000	$20,338,986
4	*Ethereum*	Cryptocurrency	Bitcoin, Independent	Sep 2, 2014	-	$18,439,086
5	*Coolest Cooler*	Product Design	Kickstarter	Aug 29, 2014	$50,000	$13,285,226

… one sold 60,102,206 digital tokens whose value is to pay for computational cycles in a decentralized network

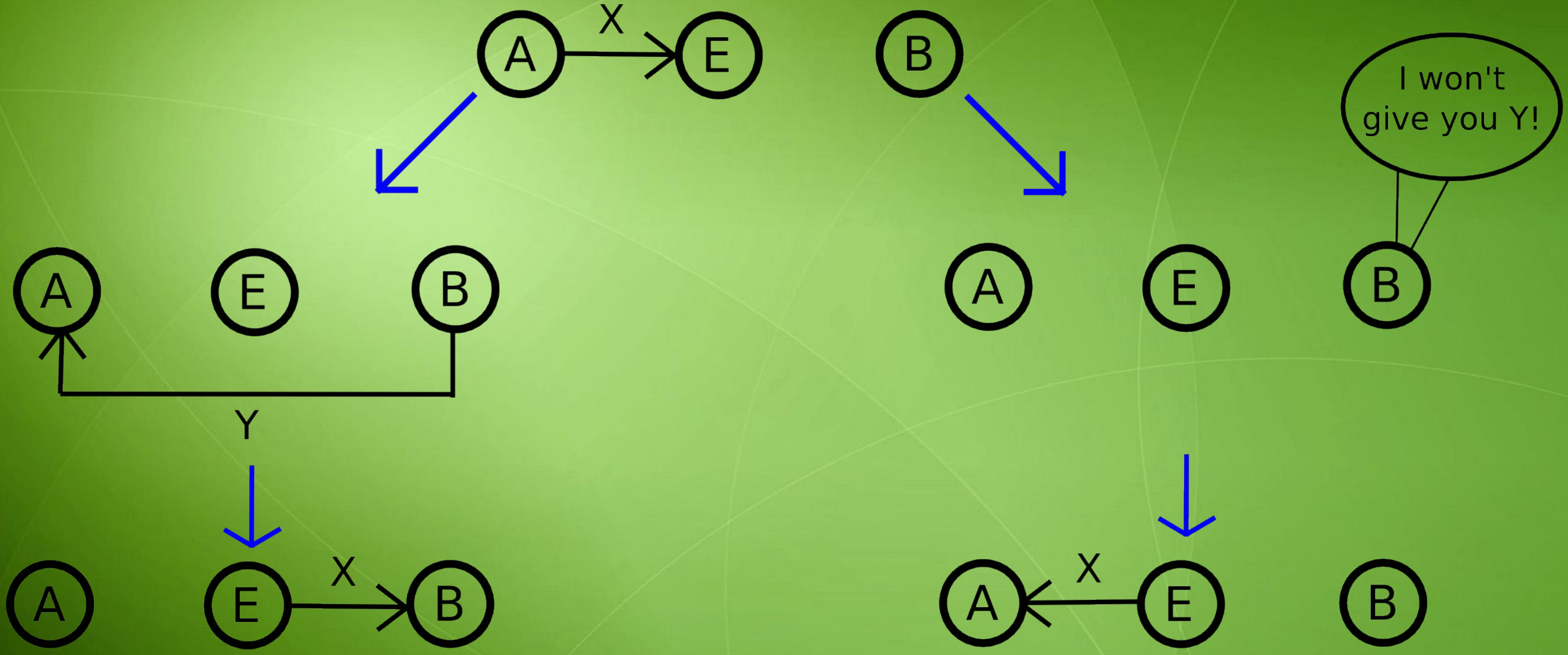


Rank	Project	Category	Platform	Campaign end date	Campaign target	Amount raised
1	*Star Citizen*	Video game	Kickstarter, independent	Ongoing	$500,000	$90,009,649
2	*Elio Motors*	Automotive - Low-cost, high mileage vehicle	Independent	Ongoing	-	$21,161,869
3	*Pebble Time*	Smartwatch	Kickstarter	Mar 27, 2015	$500,000	$20,338,986
4	*Ethereum*	Cryptocurrency	Bitcoin, Independent	Sep 2, 2014	-	$18,439,086
5	*Coolest Cooler*	Product Design	Kickstarter	Aug 29, 2014	$50,000	$13,285,226

… one sold virtual spaceships in their MMORPG for $500 each

A few more examples of what contracts can do…

Example: escrow



Example: contract for difference

```
if timestamp > 1445000000:
    send(1000 - 50 * (GET_CHF_USD_RATE() - 0.93), party_A)
    send(50 * (GET_CHF_USD_RATE() - 0.93), party_B)
```

In reality you also want margin calls

```
if timestamp > 1445000000 or
          GET_CHF_USD_RATE() < 0.95 or
          GET_CHF_USD_RATE() > 1.11:
    send(1000 - 50 * (GET_CHF_USD_RATE() - 0.93), party_A)
    send(50 * (GET_CHF_USD_RATE() - 0.93), party_B)
```

Example: multisig

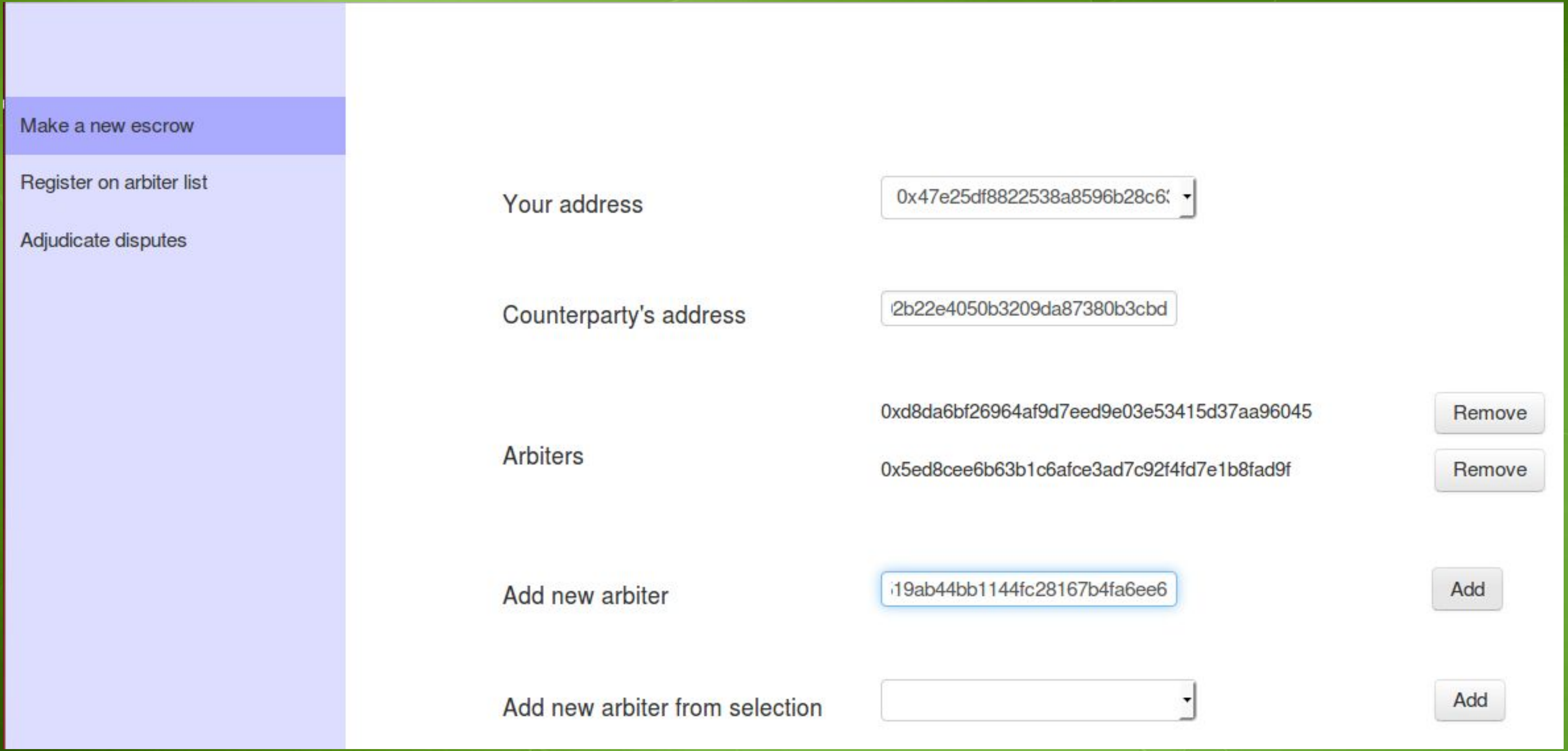
- Code is more complicated, but...
- Basically, need M of N “owners” to agree in order for a transaction to be sent

```
data operations[](to, value, sigs, sigMask, done)
data owners[7]
def sign(my_owner_ID, send_ID):
     if self.owners[my_owner_ID] == msg.sender:
          if not (self.operations[send_ID].sigMask & 2**my_owner_ID):
               self.operations[send_ID].sigMask |= 2**my_owner_ID
               self.operations[send_ID].sigs += 1
               if self.operations[send_ID].sigs == 4 && !self.operations[send_ID].done:
                    send(self.operations[send_ID].to, self.operations[send_ID].value)
                    self.operations[send_ID].done = 1
```

Example: multisig

- Can also do more complex access policies
  - eg. need 1 of 7 to withdraw up to $1000 per day, 4 of 7 to withdraw more
- Mandatory waiting periods
- Dead man switches + self-executing “digital wills”
- Put an entire organizational governance policy onto a smart contract == proto-”DAO”

… crypto doesn't mean you can't have a pretty interface!



Your address 0x47e25df8822538a8596b28c6

Counterparty's address 2b22e4050b3209da87380b3cbd

Arbiters
0xd8da6bf26964af9d7eed9e03e53415d37aa96045
0x5ed8cee6b63b1c6afce3ad7c92f4fd7e1b8fad9f

Add new arbiter 19ab44bb1144fc28167b4fa6ee6

Add new arbiter from selection

Example: computational markets

- Put up a smart contract bounty for a solution to a mathematical problem
- Put up a smart contract bounty for submitting proofs of retrievability for a file with a given root hash

Example: prediction markets

- Users trade shares that pay $1 if an event happens, $0 if it does not
- Requires external mechanism to report on whether the event took place
    - Can use multiple mechanisms to reduce trust if desired
- Idea: give society a real-time view what probability the market thinks the event has of happening

An interesting use case

- Hashcash: require sender of every email to spend ~$0.01 of computational effort to produce a nonce that must be attached to each email
- Goal: fight spam
- Alternative: require sender of every email to create a security deposit, gives recipient right to destroy $1 by clicking “Report Spam”
- More powerful spam protection at lower average cost

Smart contracts vs legal contracts

- This distinction is commonly brought up, but it is important to understand the two are very different
- A smart contract is more like a vending machine than a legal contract

Smart contracts vs legal contracts

Legal contracts	**Smart contracts**
Good at subjective (ie. requiring human judgement) claims	Good at objective (ie. mathematically evaluable) claims
High cost	Low cost
Ex-post enforcement (which of course provides ex-ante incentivization)	Ex-ante prevention
Relies on penalties	Relies on collateral/security deposits
Jurisdiction-bound	Potentially international (“a-legal”)

Smart contracts vs legal contracts

- **Example:** smart contracts are not very effective for loans, because if someone has the capital to provide liquid collateral for a loan they do not need the loan in the first place
  - Can use illiquid collateral though (eg. domain names)
- **Example:** legal contracts are not very effective for the anti-spam use case because amounts at stake are so small, and spammers can locate themselves in favorable jurisdictions and evade detection

Smart contracts are like robots

CONTRACT

...party A agrees to exercise reasonable care in providing…….

So why are smart contracts useful?

Smart contracts are mostly useful where...

- Participants are in many countries, or theoretically could be in any country
- Monetary amounts are small
- Conditions are easily programmatically verifiable
- Fast and certain resolution is desired
- Users want the privacy of not revealing details of their agreement to any external third party

Partially smart contracts

- eg. Legalese.io (Singapore)
- Make “smart” contracts which are processed automatically in many cases, but in corner cases explicitly relegate decision-making authority to a human third party
- Potentially mix best of both worlds

Now how do you build them?

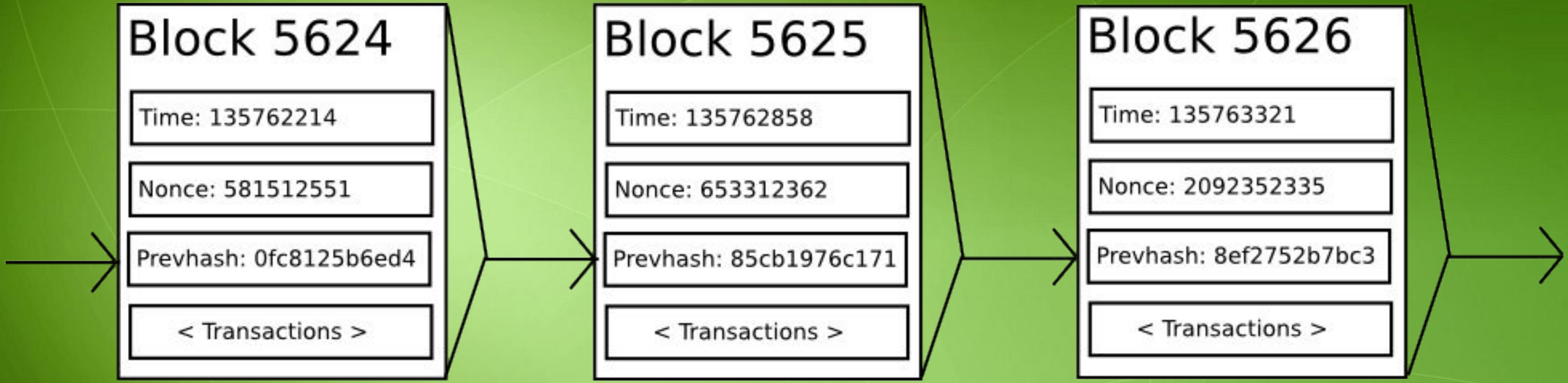
Ethereum

- Blockchain with built-in programming language
- Designed for maximum abstraction and generality
  - ○ Programming language makes it ideal for smart contracts

Quick overview of blockchains

- Decentralized computer network that simulates a computer running a program (ie. state machine)
- What is “state”?
  - Balances (Bitcoin)
  - Domain names (Namecoin)
  - Computer code and storage (Ethereum)
- Transactions sequentially processed and update the state

Quick overview of blockchains



- Transactions grouped into blocks
- Guarantee that eventually everyone in the network will agree on the order of blocks up to any given point
- Consensus mechanisms, eg. “Proof of work” / mining

Ethereum

- Two types of accounts
  - User accounts (controlled by external private key)
  - Contracts (controlled by code)
- Anyone can create an application or smart contract by writing it as a contract on Ethereum

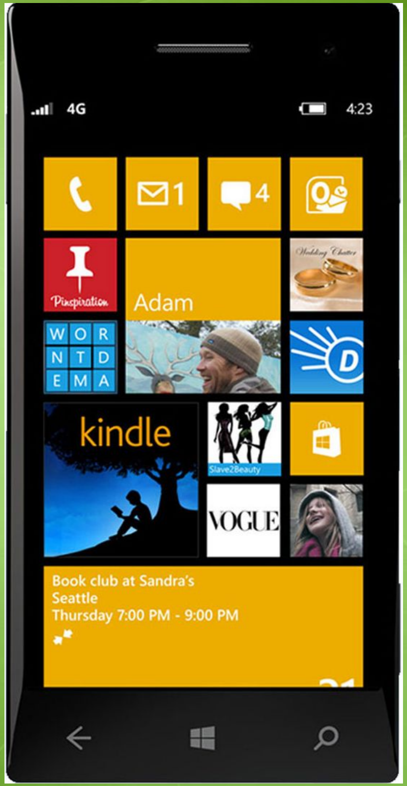
Why Ethereum?

- Seeming public consensus circa 2013: blockchains are useful for… stuff
- Not just money! Asset issuance, crowdfunding, domain registration, title registration, gambling, prediction markets, internet of things, voting, hundreds of applications!

Problem: most existing blockchain protocols were designed like this:

Or, at best, like this:

So… why not make a protocol that works like this?



DNS: The “Hello World” of Ethereum

```
data domains[](owner, ip)

def register(addr):
   if not self.domains[addr].owner:
      self.domains[addr].owner = msg.sender

def set_ip(addr, ip):
   if self.domains[addr].owner == msg.sender:
      self.domains[addr].ip = ip
```

How Ethereum Works

- Every transaction specifies a TO address it sends to (unless it’s creating a contract)
- The TO address’s code runs
- Code can:
    - Send ETH to other contracts
    - Read/write storage
    - Call (ie. start execution in) other contracts (can be used recursively)

How Ethereum Works

- Every (full) node on the blockchain processes every transaction and stores the entire state, just like Bitcoin

How Ethereum Works

- Halting problem
  - Cannot tell whether or not a program will run infinitely
- Solution: charge fee per computational step (“gas”)
- Special gas fees also applied to operations that take up storage
- There is a “gas limit” per block, analogous to block size limit in bitcoin

Ether

- Cryptographic token inside of Ethereum
- Two primary applications
  - Given to miners as a reward for securing the network
  - Used to pay transaction/gas fees

High-level programming languages

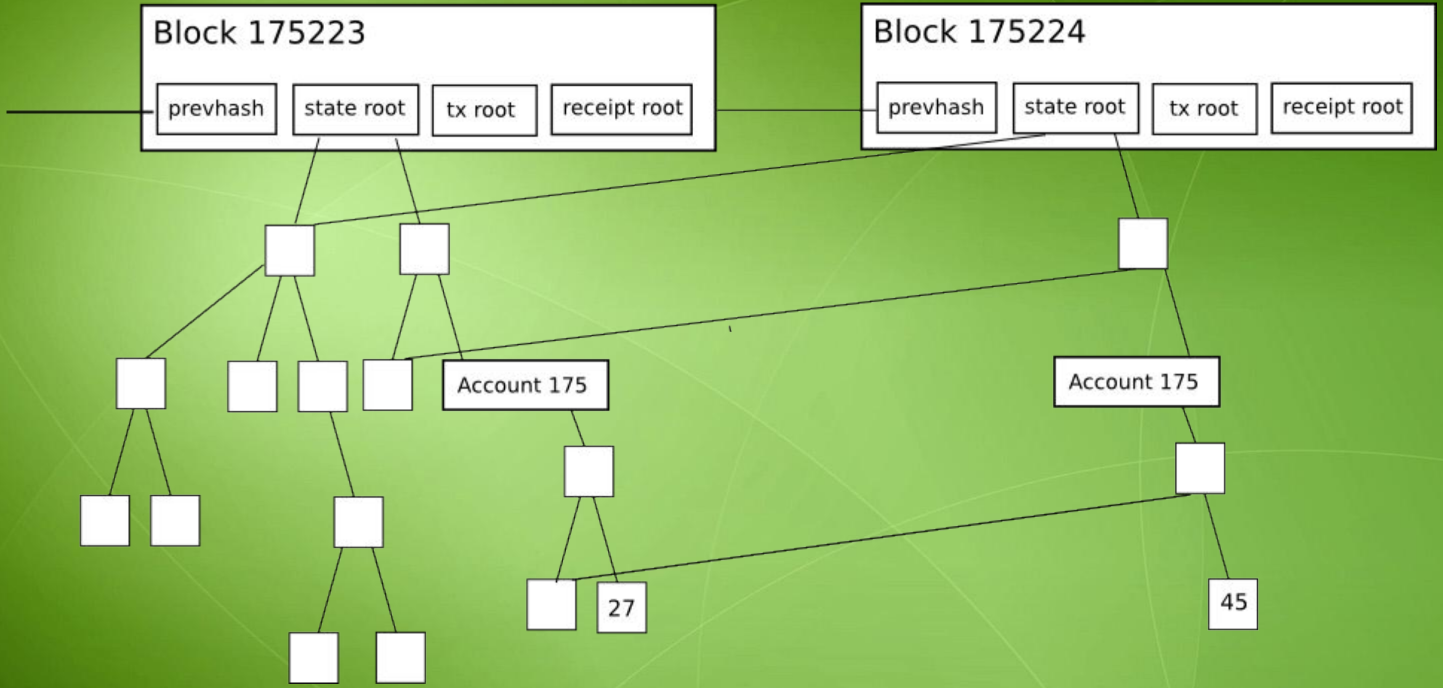
- Multiple languages exist
  - LLL
  - Serpent
  - Solidity
- Compile to EVM code (executed by all nodes processing blockchain)
- Another tool “compiles” function calls with arguments into bytes passed as transaction data

Light client friendliness

- Problem: not every computer can process every transaction in every block on the blockchain
  - Smartphones
  - IoT devices
  - Eventually, even many regular laptops

Merkle trees

Merkle trees



- Allow users to efficiently look up and verify small parts of the blockchain on-demand without processing the whole thing

What do people use it for?

- Digital assets
  - Ether
  - Stablecoins (Maker, String, etc)
- Registries
  - Decentralized domain names, chat usernames, e-commerce sites
- Finance
  - Hedging, derivatives, etc
- Prediction markets

What do people use it for?

- Economic / social experimentation
  - Monetary policy (eg. stablecoins)
  - Basic income / decentralized insurance projects
  - Decentralized autonomous organizations
- IoT
  - Tracking state and ownership of hardware
  - Automated sharing economy (eg. slock.it)
- Identity
  - Decentralized single-sign-in
  - Reputation (incl economic approaches)

Challenges of blockchains

- Scalability
  - State channels
  - Sharding
- Privacy
  - Ring signatures
  - zk-SNARKs
- Speed
- Efficiency
  - Proof of stake

Smart contracts and blockchains

- The two do naturally go well together
  - Automatic execution + decentralized trust model
- However, smart contracts can of course be applied in other contexts
  - Centralized financial systems
  - Hardware devices
- Also, not all Ethereum contracts are smart contracts in the sense of controlling digital assets (eg. data publishing use cases)