

# Cryptocurrency

# Cryptocurrency

What is it?

What problems does it solve?

What problems does it introduce?



*What is*  
**Bitcoin?**



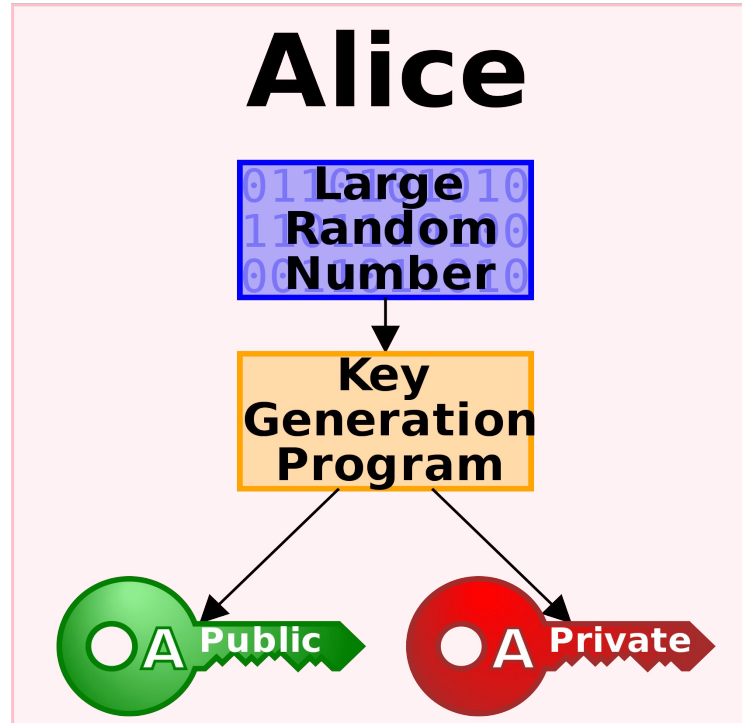
# Concepts

**Digital Signature** (Public Key Cryptography)

**Mining** (Proof of work)

**Blockchain** (Double-spend problem)

# Public/Private Key Cryptography



# Bitcoin Wallet

The screenshot shows the bitaddress.org website in a browser. The page title is "Open Source JavaScript Client-Side Bitcoin Wallet Generator". The main content area is a green-bordered box containing a "Generate New Address" button, a "Print" button, and two QR codes. The left QR code is labeled "SHARE" and the right is labeled "SECRET". Below the QR codes are the Bitcoin address "1Bk8paA7Pguaj9qr9NNGpthrZCBAzKtr4J" and the private key "L4JN6pCy9N1ew04g3aR86xVLvtxnZzXkAkmuxCR6d4BejJ0jvUoP".

English | Español | Français | ελληνικά | Italiano | Deutsch  
Česky | Magyar | 日本語 | 简体中文 | Русский | português

**bitaddress.org**  
Open Source JavaScript Client-Side Bitcoin Wallet Generator

Single Wallet Paper Wallet Bulk Wallet Brain Wallet  
Vanity Wallet Split Wallet Wallet Details

Generate New Address Print

**Bitcoin Address** **Private Key**

**SHARE** **SECRET**

1Bk8paA7Pguaj9qr9NNGpthrZCBAzKtr4J  
L4JN6pCy9N1ew04g3aR86xVLvtxnZzXkAkmuxCR6d4BejJ0jvUoP

**A Bitcoin wallet** is as simple as a single pairing of a Bitcoin address with its corresponding Bitcoin private key. Such a wallet has been generated for you in your web browser and is displayed above.

**To safeguard this wallet** you must print or otherwise record the Bitcoin address and private key. It is important to make a backup copy of the private key and store it in a safe location. This site does not have knowledge of your private key. If you are familiar with PGP you can download this all-in-one HTML page and check that you have an authentic version from the author of this site by matching the SHA256 hash of this HTML with the SHA256 hash available in the signed version history document linked on the footer of this site. If you leave/refresh the site or press the "Generate New Address" button then a new private key will be generated and the previously displayed private key will not be retrievable. Your Bitcoin private key should be kept a secret. Whomever you share the private key with has access to spend all the bitcoins associated with that address. If you print your wallet then store it in a zip lock bag to keep it safe from water. Treat a paper wallet like cash.

**Add funds** to this wallet by instructing others to send bitcoins to your Bitcoin address.

**Check your balance** by going to [blockchain.info](http://blockchain.info) or [blockexplorer.com](http://blockexplorer.com) and entering your Bitcoin address.

**Spend your bitcoins** by going to [blockchain.info](http://blockchain.info) and sweep the full balance of your private key into your account at their website. You can also spend your funds by downloading one of the popular bitcoin p2p clients and importing your private key to the p2p client wallet. Keep in mind when you import your single key to a bitcoin p2p client and spend funds your key will be bundled with other private keys in the p2p client wallet. When you perform a transaction your change will be sent to another bitcoin address within the p2p client wallet. You must then backup the p2p client wallet and keep it safe as your remaining bitcoins will be stored there. Satoshi advised that one should never delete a wallet.

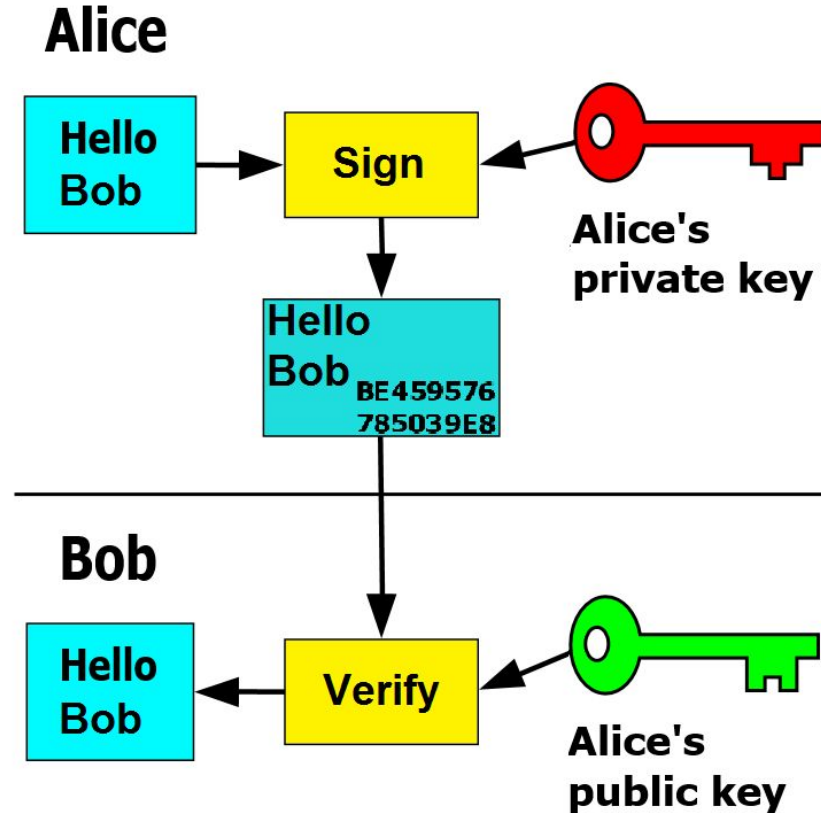
△ ✓ ✖ ☰

Donations: **1NINja1bUmhSoTXozRBBER8LeF9TGzBN**  
[GitHub Repository](#) (zip)

[Version History \(3.3.0\)](#)  
527B 5C82 B1F6 B2DB 7240  
ECBF 8749 7B91 6397 4F5A  
(PGP) (sig)

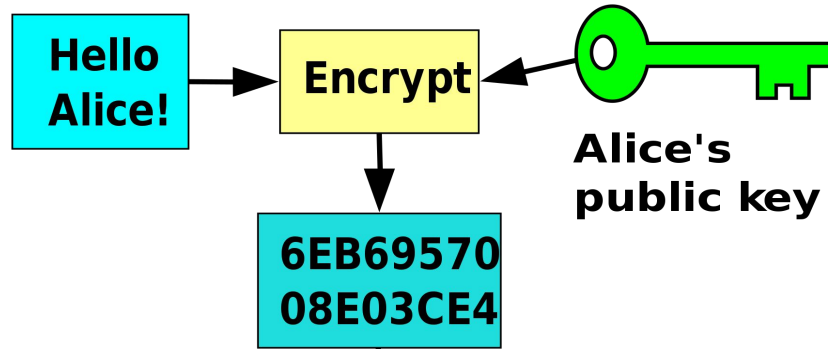
Copyright bitaddress.org. JavaScript copyrights are included in the source. No warranty.

Hello Bob, I want to buy your book for 0.01XBT.

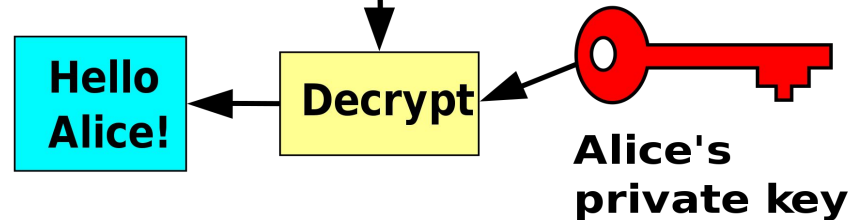


Hello Alice, here's the book.

**Bob**



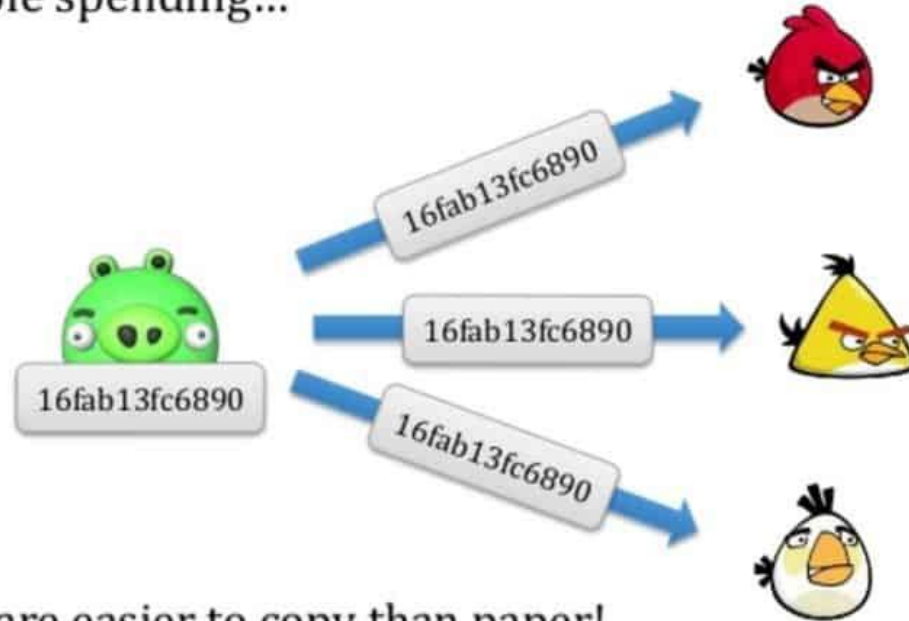
**Alice**





# Double-Spend

Double spending...



Bits are easier to copy than paper!

# Mining



# Hash

SHA256:

“The quick brown fox jumps over the lazy dog.”

ef537f25c895bfa782526529a9b63d97aa631564d5d789c2b765448c8635fb6c

“The quick brown fox, jumps over the lazy dog.”

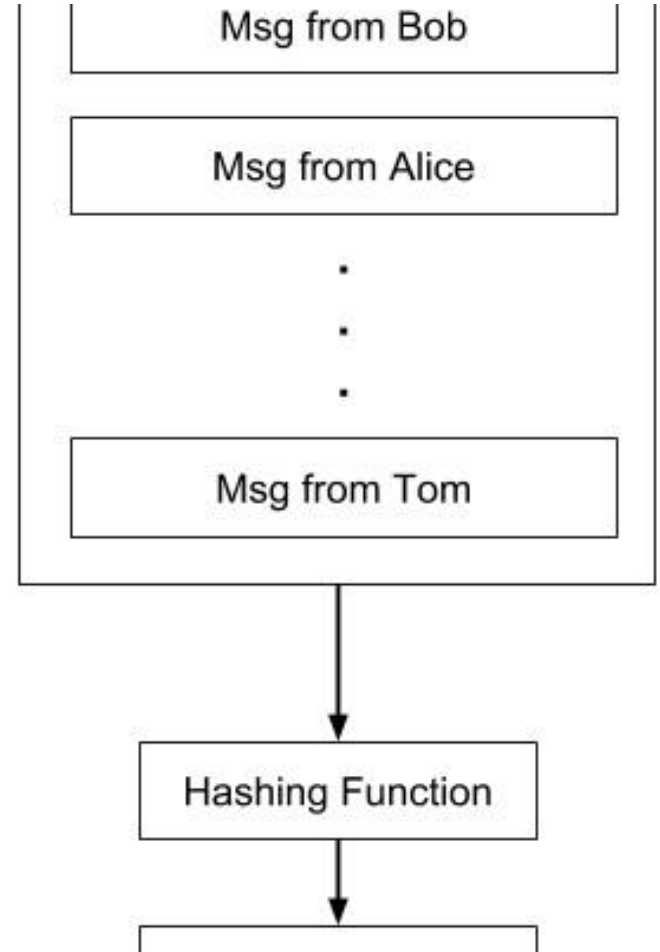
392bfd94c083e025e46d0be3ff9258c8bfc33bcd3296156f76c722f339a98dfb

The text of “War and Peace” <http://www.gutenberg.org/files/2600/2600-0.txt>

1feba561bf9106f3cbf6d78dd0c6056eef6ab59f15a30e64530ea6aea91d4e07

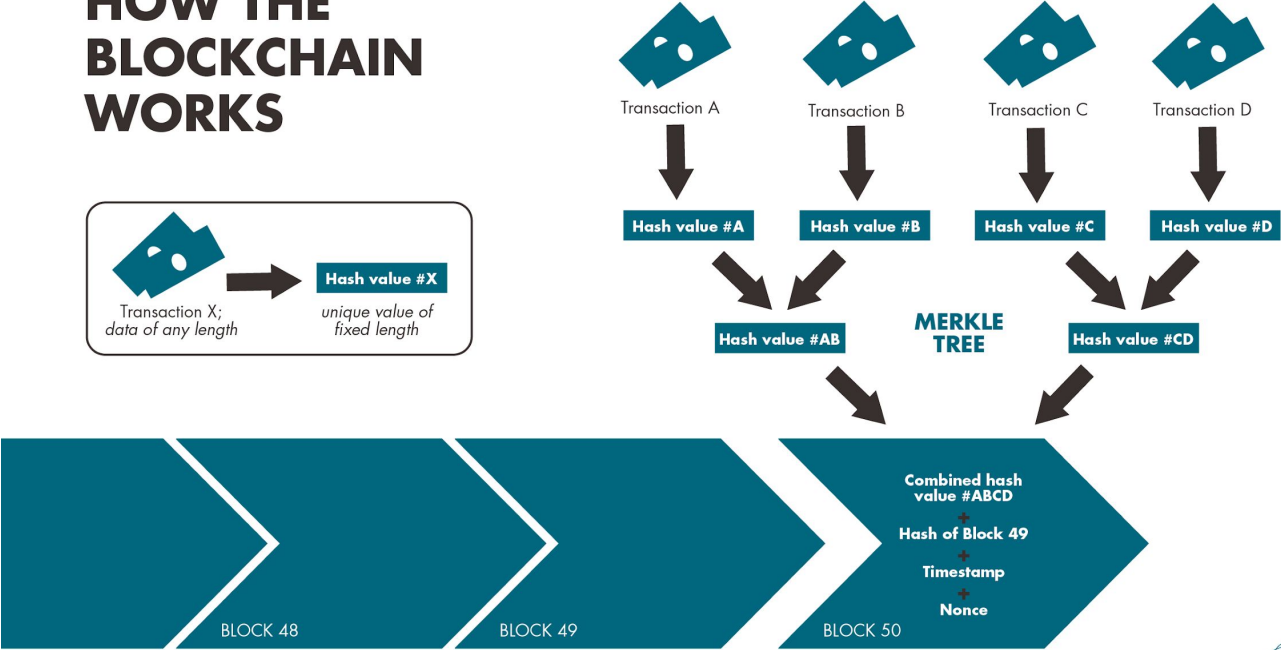
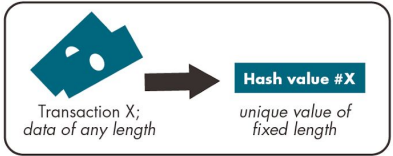
# Mining Process - 12.5 XBT per block

[https://www.tutorialspoint.com/blockchain/bitcoin\\_mining.htm](https://www.tutorialspoint.com/blockchain/bitcoin_mining.htm)



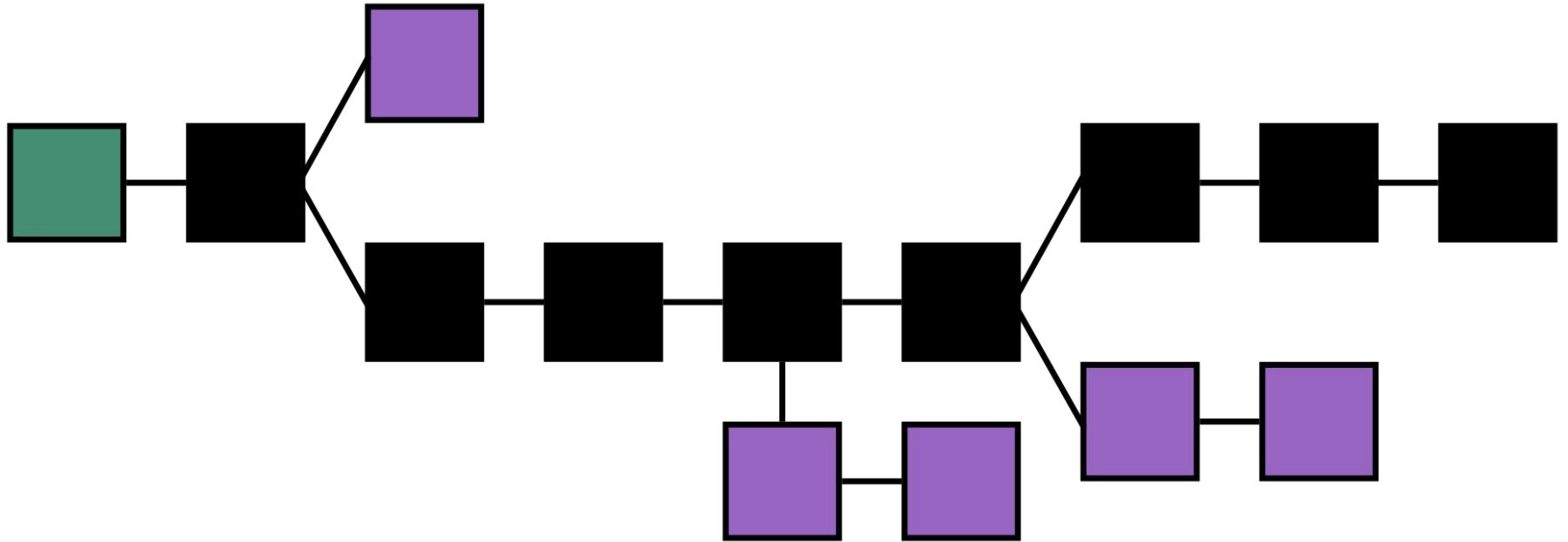
# Blockchain

## HOW THE BLOCKCHAIN WORKS



Reproduction of an original figure in "The Great Chain of Being Sure About Things" by the Economist

# Longest Chain wins



# What problems does it introduce?

Irreversible

Hackable

Not really anonymous

Not really decentralized

**Table 4.** Normalized global results for Energy, Time, and Memory

Total					
	Energy		Time		Mb
(c) C	1.00	(c) C	1.00	(c) Pascal	1.00
(c) Rust	1.03	(c) Rust	1.04	(c) Go	1.05
(c) C++	1.34	(c) C++	1.56	(c) C	1.17
(c) Ada	1.70	(c) Ada	1.85	(c) Fortran	1.24
(v) Java	1.98	(v) Java	1.89	(c) C++	1.34
(c) Pascal	2.14	(c) Chapel	2.14	(c) Ada	1.47
(c) Chapel	2.18	(c) Go	2.83	(c) Rust	1.54
(v) Lisp	2.27	(c) Pascal	3.02	(v) Lisp	1.92
(c) Ocaml	2.40	(c) Ocaml	3.09	(c) Haskell	2.45
(c) Fortran	2.52	(v) C#	3.14	(i) PHP	2.57
(c) Swift	2.79	(v) Lisp	3.40	(c) Swift	2.71
(c) Haskell	3.10	(c) Haskell	3.55	(i) Python	2.80
(v) C#	3.14	(c) Swift	4.20	(c) Ocaml	2.82
(c) Go	3.23	(c) Fortran	4.20	(v) C#	2.85
(i) Dart	3.83	(v) F#	6.30	(i) Hack	3.34
(v) F#	4.13	(i) JavaScript	6.52	(v) Racket	3.52
(i) JavaScript	4.45	(i) Dart	6.67	(i) Ruby	3.97
(v) Racket	7.91	(v) Racket	11.27	(c) Chapel	4.00
(i) TypeScript	21.50	(i) Hack	26.99	(v) F#	4.25
(i) Hack	24.02	(i) PHP	27.64	(i) JavaScript	4.59
(i) PHP	29.30	(v) Erlang	36.71	(i) TypeScript	4.69
(v) Erlang	42.23	(i) Jruby	43.44	(v) Java	6.01
(i) Lua	45.98	(i) TypeScript	46.20	(i) Perl	6.62
(i) Jruby	46.54	(i) Ruby	59.34	(i) Lua	6.72
(i) Ruby	69.91	(i) Perl	65.79	(v) Erlang	7.20
(i) Python	75.88	(i) Python	71.90	(i) Dart	8.64
(i) Perl	79.58	(i) Lua	82.91	(i) Jruby	19.84

"Energy Efficiency across Programming Languages: How does Energy, Time and Memory Relate?", Rui Pereira, Marco Couto, Francisco Ribeiro, Rui Rua, Jácome Cunha, João Paulo Fernandes, and João Saraiva. In *Proceedings of the 10th International Conference on Software Language Engineering (SLE '17)*



github.com

Search or jump to... Pull requests Issues Marketplace Explore

jgarzik / pyniner Watch 70 Star 440 Fork 167

Code Issues 1 Pull requests 1 Projects 0 Wiki Insights

Branch: master - pyniner / pyniner.py Find file Copy path

Jeff Garzik Remove remnants of being a pool daemon. 8a5b107 on May 31, 2011

0 contributors

Executable File 264 lines (217 sloc) | 6.75 KB Raw Blame History

```
1 #!/usr/bin/python
2 #
3 # Copyright 2011 Jeff Garzik
4 #
5 # This program is free software; you can redistribute it and/or modify
6 # it under the terms of the GNU General Public License as published by
7 # the Free Software Foundation.
8 #
9 # This program is distributed in the hope that it will be useful,
10 # but WITHOUT ANY WARRANTY; without even the implied warranty of
11 # MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
12 # GNU General Public License for more details.
13 #
14 # You should have received a copy of the GNU General Public License
15 # along with this program; see the file COPYING. If not, write to
16 # the Free Software Foundation, 675 Mass Ave, Cambridge, MA 02139, USA.
17 #
18
19 import time
20 import json
21 import pprint
22 import hashlib
23 import struct
24 import re
25 import base64
26 import httpLib
27 import sys
28 from multiprocessing import Process
29
30 ERR_SLEEP = 15
31 MAX_NONCE = 1000000
32
33 settings = {}
34 pp = pprint.PrettyPrinter(indent=4)
35
36 class BitcoinRPC:
37     OBJID = 1
38
39     def __init__(self, host, port, username, password):
40         authpair = "%s:%s" % (username, password)
41         self.authhdr = "Basic %s" % (base64.b64encode(authpair))
42         self.conn = httpLib.HTTPConnection(host, port, False, 30)
43     def rpc(self, method, params=None):
44         self.OBJID += 1
45         obj = { 'version' : '1.1',
46               'method' : method,
47               'id' : self.OBJID }
48         if params is None:
49             obj['params'] = []
50         else:
51             obj['params'] = params
```

https://solidity.readthedocs.io/en/v0.5.5/solidity-by-example.html

Solidity  
v0.5.5

Search docs

Introduction to Smart Contracts  
Installing the Solidity Compiler

Solidity by Example

- Voting
- Blind Auction
  - Safe Remote Purchase
- Micropayment Channel
- Modular Contracts

Solidity in Depth  
Security Considerations  
Resources  
Using the compiler  
Contract Metadata  
Contract ABI Specification  
Yul  
Style Guide  
Common Patterns  
List of Known Bugs  
Contributing  
LLL  
Keyword Index

Beat Triplebyte's online coding quiz. Get offers from top companies. Skip resumes & recruiters.  
Sponsored - Ads served ethically

Docs » Solidity by Example [Edit on GitHub](#)

## Solidity by Example

### Voting

The following contract is quite complex, but showcases a lot of Solidity's features. It implements a voting contract. Of course, the main problems of electronic voting is how to assign voting rights to the correct persons and how to prevent manipulation. We will not solve all problems here, but at least we will show how delegated voting can be done so that vote counting is **automatic and completely transparent** at the same time.

The idea is to create one contract per ballot, providing a short name for each option. Then the creator of the contract who serves as chairperson will give the right to vote to each address individually.

The persons behind the addresses can then choose to either vote themselves or to delegate their vote to a person they trust.

At the end of the voting time, `winningProposal()` will return the proposal with the largest number of votes.

```
pragma solidity >=0.4.22 <0.6.0;

/// @title Voting with delegation.
contract Ballot {
    // This declares a new complex type which will
    // be used for variables later.
    // It will represent a single voter.
    struct Voter {
        uint weight; // weight is accumulated by delegation
        bool voted; // if true, that person already voted
        address delegate; // person delegated to
        uint vote; // index of the voted proposal
    }

    // This is a type for a single proposal.
    struct Proposal {
        bytes32 name; // short name (up to 32 bytes)
        uint voteCount; // number of accumulated votes
    }

    address public chairperson;

    // This declares a state variable that
    // stores a "voter" struct for each possible address.
    mapping(address => Voter) public voters;

    // A dynamically-sized array of "Proposal" structs.
    Proposal[] public proposals;

    // Create a new ballot to choose one of "proposalsNames".
    constructor(bytes32[] memory proposaNames) public {
        chairperson = msg.sender;
        voters[chairperson].weight = 1;

        // For each of the provided proposal names,
        // create a new proposal object and add it
        // to the end of the array.
        for (uint i = 0; i < proposaNames.length; i++) {
            // Proposal(i,...) creates a temporary
            // Proposal object and "proposals.push(...)"
            // appends it to the end of "proposals".
            proposals.push(Proposal({
                name: proposaNames[i],
                voteCount: 0
            }));
        }
    }
}
```

Read the Docs v: v0.5.5