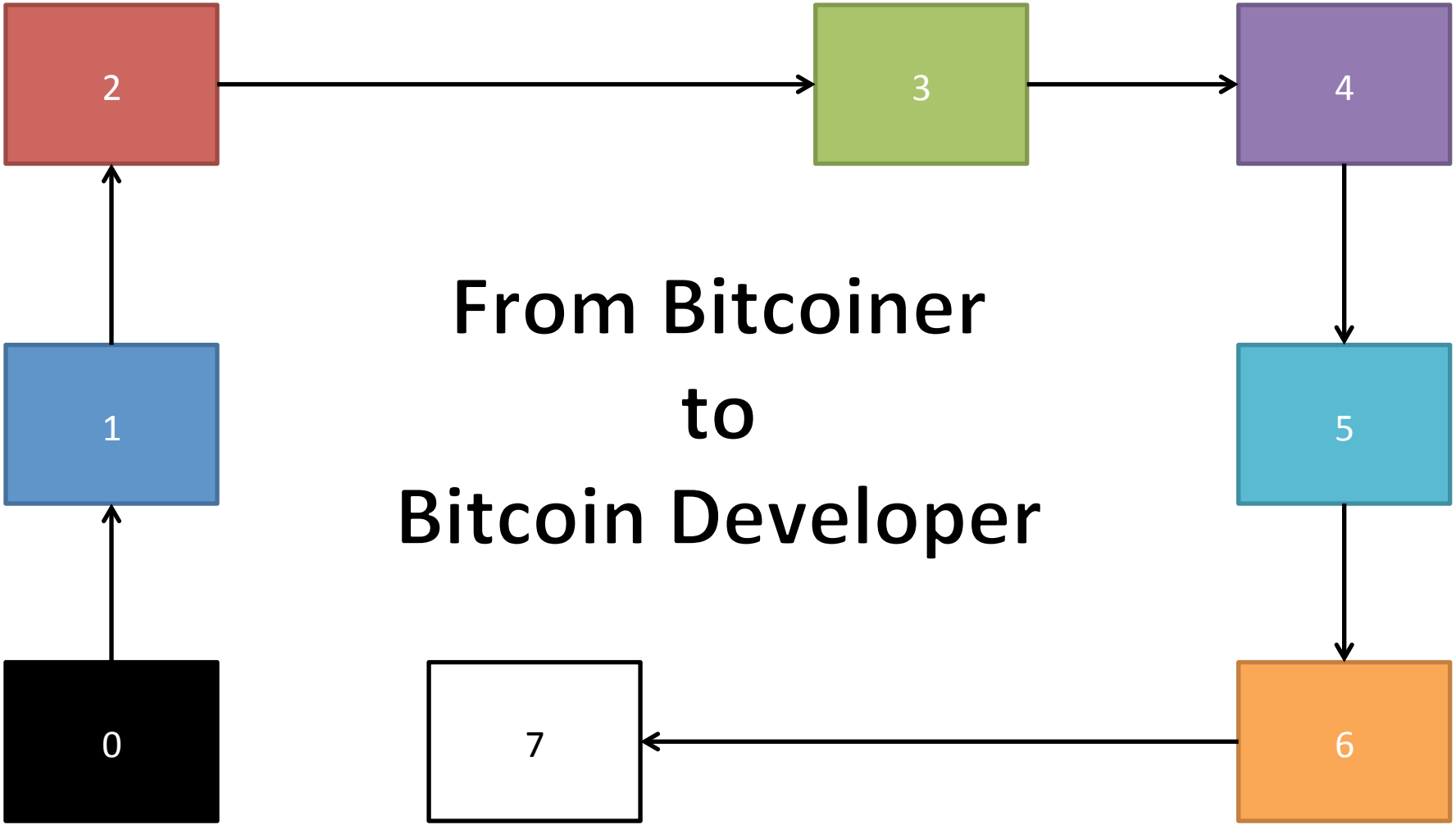

2
3
4
From Bitcoiner
to
Bitcoin Developer
1
5
0
7
6

Who Are *You*?

- You know
  - Computer Science Basics (Big O, data structures, basic crypto, etc)
  - Computer System Basics (unix, bash, git/github, etc)
  - Bitcoin Basics (transactions, blocks, PoW, etc)
- You can
  - Code in any language proficiently
- You have
  - Free time
- You want
  - To work on core bitcoin protocols

Not you?

- Sorry!

Who am I?

- Co-Founder of MIT DCI
- MIT Bitcoin Project
- Bitcoin Core Contributor
- Freelance work
- MIT SB CS ‘16, MEng EECS ’16
- Contact:
  - twitter: @JeremyRubin
  - email: jr@mit.edu
  - LineID: jeremyrubin

So You Want to Be a Bitcoin Developer

- Foundations
- Development Environment
- Beginning to Code Bitcoin
- Contributing
- General Advice

FOUNDATIONS

Understand Development Philosophy

- Respect all kinds of Bitcoin users
- Scratch your own itch
- Bitcoin use is free speech
- Slow and steady
- **Not everyone has same goal for project**

Communications

- Join bitcoincore.slack.com
- IRC channels
  - #bitcoin, #bitcoin-wizards, #bitcoin-core-dev
  - Weekly meeting in #bitcoin-core-dev
- Follow github.com/bitcoin/bitcoin
- Join Linux Foundation mailing lists
  - bitcoin-core-dev, bitcoin-dev, bitcoin-discuss
- Bitcoin StackExchange

Communications

- Someone already probably asked your question somewhere!
- Be respectful of what other people want to discuss, don’t ‘demand’ answers

DEVELOPMENT ENVIRONMENT

Hardware & System

- Recommend using an Ubuntu LTS 16.04
- You'll want 200+ GB free space
- Recommend: 4+ cores, 8+GB RAM
  - Bitcoin can *run* on less, but *building* is slow!

Fork, Clone, Build

- `git clone git@github.com:bitcoin/bitcoin.git`

  – or your own fork!

- `git checkout –b my-devel-branch`
- Follow build instructions (first build is slow)

Run Bitcoin Nodes

- Copy the binaries you just compiled
  - src/bitcoind and src/bitcoin-cli
- `./bitcoind -debug=bench` will run a node, use `./bitcoin-cli` to test it
- Testnet node: `./bitcoind -testnet -debug=bench`, `./bitcoin-cli -testnet`

Using ctags

- Bitcoin has a lot of code! ctags helps you browse it quickly
- Vim specific, but Emacs/your favorite editor has equivalent
  - `ctags –R .` in src dir to generate
  - Hit `C-]` in vim to jump to definition, `:ts` to select if there are multiple possible locations
  - Hit `C-t` to go back to prior location

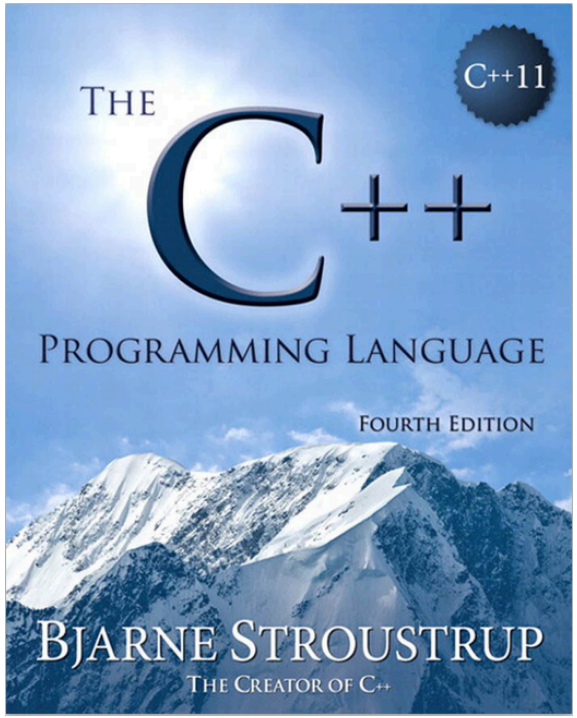
BEGINNING TO CODE BITCOIN

Pick a Good "Bad Idea"

- My First Project
  - Randomize order of a databases entries to prevent DoS on bad DB access pattern
  - Implemented a proof of concept
  - Asked a developer for feedback
    - Bad
      - We may eventually want DB iterable in order
      - Couldn't demonstrate an example of bad access pattern
    - Good
      - Learning experience

Build C++ Expertise

- Bitcoin is written in C++11
- Learn C++11 well as you start (first ~25 chapters or so, a chapter or two a day)
- cppreference.com!
- C++: “easy” to read, “hard” to write



Use gdb

- Too complicated to get into here
- Basically
  - See what your code is doing step-by-step
  - Find bugs
  - Inspect running programs

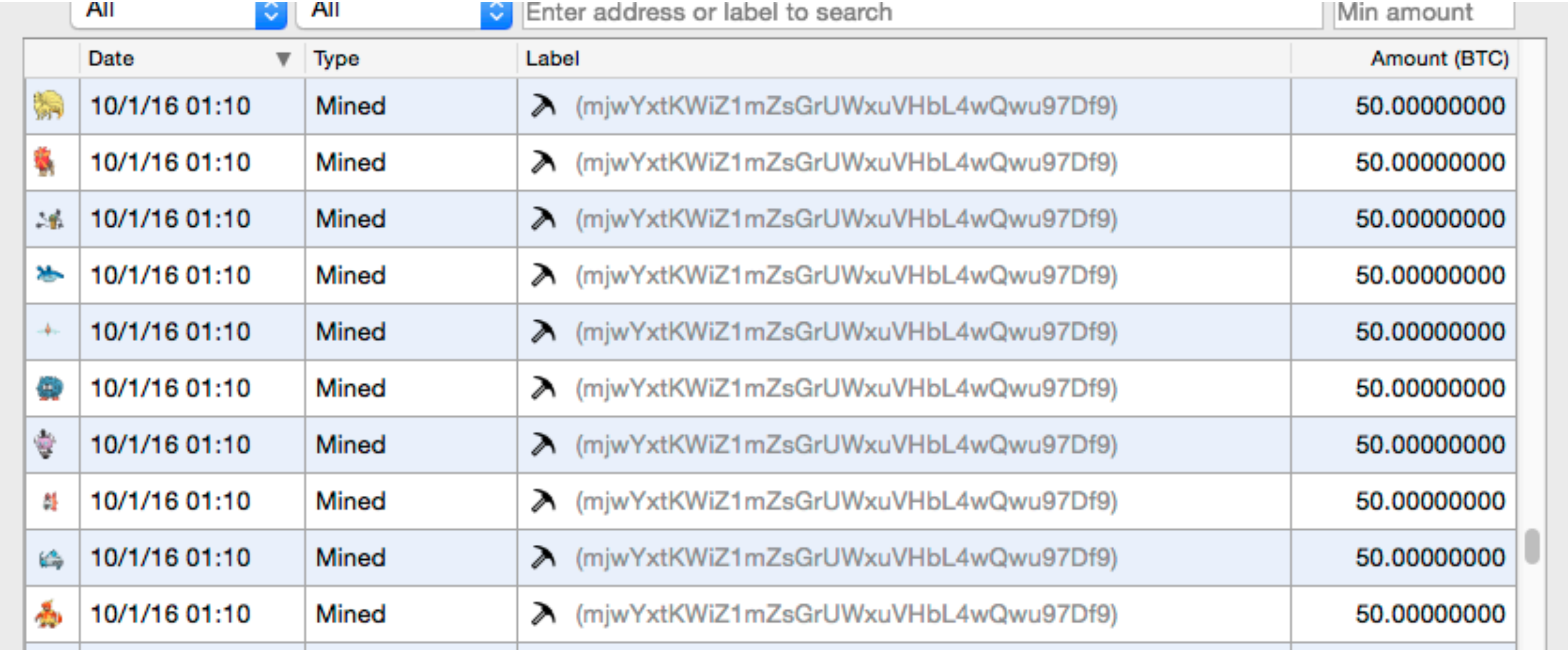
Review Others' Code

- Look through github.com/bitcoin/bitcoin/pulls
- You'll learn
  - What *topics* people are working on
  - *How* people communicate feedback
  - What *kinds* of feedback people get
- If you leave *useful* feedback for someone, **they will be happy**

Testing Bitcoin

- `make check` runs unit tests
- `./qa/pull-tester/rpc-tests.py` to run rpc tests
- Enable Travis CI on your fork for testing to happen automatically

Improve the GUI



	Date	Type	Label	Amount (BTC)
	10/1/16 01:10	Mined	(mjwYxtKWiZ1mZsGrUWxuVHbL4wQwu97Df9)	50.00000000
	10/1/16 01:10	Mined	(mjwYxtKWiZ1mZsGrUWxuVHbL4wQwu97Df9)	50.00000000
	10/1/16 01:10	Mined	(mjwYxtKWiZ1mZsGrUWxuVHbL4wQwu97Df9)	50.00000000
	10/1/16 01:10	Mined	(mjwYxtKWiZ1mZsGrUWxuVHbL4wQwu97Df9)	50.00000000
	10/1/16 01:10	Mined	(mjwYxtKWiZ1mZsGrUWxuVHbL4wQwu97Df9)	50.00000000
	10/1/16 01:10	Mined	(mjwYxtKWiZ1mZsGrUWxuVHbL4wQwu97Df9)	50.00000000
	10/1/16 01:10	Mined	(mjwYxtKWiZ1mZsGrUWxuVHbL4wQwu97Df9)	50.00000000
	10/1/16 01:10	Mined	(mjwYxtKWiZ1mZsGrUWxuVHbL4wQwu97Df9)	50.00000000
	10/1/16 01:10	Mined	(mjwYxtKWiZ1mZsGrUWxuVHbL4wQwu97Df9)	50.00000000
	10/1/16 01:10	Mined	(mjwYxtKWiZ1mZsGrUWxuVHbL4wQwu97Df9)	50.00000000

```
diff --git a/src/qt/transactiontablemodel.cpp b/src/qt/transactiontablemodel.cpp
index b29ecf8..ba28338 100644
--- a/src/qt/transactiontablemodel.cpp
+++ b/src/qt/transactiontablemodel.cpp
@@ -462,6 +462,7 @@ QString TransactionTableModel::formatTxAmount(const TransactionRecord *wtx, b
ool

 QVariant TransactionTableModel::txStatusDecoration(const TransactionRecord *wtx) const
 {
+    char buf[256];
     switch(wtx->status.status)
     {
     case TransactionStatus::OpenUntilBlock:
@@ -483,7 +484,8 @@ QVariant TransactionTableModel::txStatusDecoration(const TransactionRecord *w
tx)
         default: return QIcon(":/icons/transaction_5");
         };
     case TransactionStatus::Confirmed:
-        return QIcon(":/icons/transaction_confirmed");
+        snprintf(buf, 256, ":/pokemon/%" PRIu64, 1+(wtx->hash.GetUint64(0)%716));
+        return QIcon(buf);
     case TransactionStatus::Conflicted:
         return QIcon(":/icons/transaction_conflicted");
     case TransactionStatus::Immature: {
(END)
```

Performance Improvements

- Pick a PR that seems to impact performance and try to measure if it is better or worse
- Can you improve it?
- Example: txChanged

```
2829  -    // Tell wallet about transactions that went from mempool
2830  -    // to conflicted:
2831  -    BOOST_FOREACH(const CTransaction &tx, txConflicted) {
2832  -        SyncWithWallets(tx, pindexNew, NULL);
2833  -    }
2834  -    // ... and about transactions that got confirmed:
2835  -    BOOST_FOREACH(const CTransaction &tx, pblock->vtx) {
2836  -        SyncWithWallets(tx, pindexNew, pblock);
2837  -    }
```

```
2828  +
2829  +    for(unsigned int i=0; i < pblock->vtx.size(); i++)
2830  +        txChanged.push_back(std::make_tuple(pblock->vtx[i], pindexNew, i));
```

```
3055          // Notifications/callbacks that can run without cs_main
```

```
3050          // Notifications/callbacks that can run without cs_main
3051  +
3052  +       // throw all transactions though the signal-interface
3053  +       // while _not_ holding the cs_main lock
3054  +       BOOST_FOREACH(const CTransaction &tx, txConflicted)
3055  +       {
3056  +           SyncWithWallets(tx, pindexNewTip);
3057  +       }
3058  +       // ... and about transactions that got confirmed:
3059  +       for(unsigned int i = 0; i < txChanged.size(); i++)
3060  +           SyncWithWallets(std::get<0>(txChanged[i]), std::get<1>
(txChanged[i]), std::get<2>(txChanged[i]));
3061  +
```

```
2828  +
2829  +    for(unsigned int i=0; i < pblock->vtx.size(); i++)
2830  +        txChanged.push_back(std::make_tuple(pblock->vtx[i], pindexNew, i));
```

```
2831
      int64_t nTime6 = GetTimeMicros(); nTimePostConnect += nTime6 - nTime5;
nTimeTotal += nTime6 - nTime1;
2833  LogPrint("bench", "  - Connect postprocess: %.2fms [%.2fs]\n", (nTime6 -
nTime5) * 0.001, nTimePostConnect * 0.000001);
```

Quick ‘N Dirty Fix

```
2837      for(unsigned int i=0; i < pblock->vtx.size(); i++)
2838  -       txChanged.push_back(std::make_tuple(pblock->vtx[i], pindexNew, i));
```

```
2837      for(unsigned int i=0; i < pblock->vtx.size(); i++)
2838  +       txChanged.emplace_back(pblock->vtx[i], pindexNew, i);
```

```
3022  +   std::vector<std::tuple<CTransaction,CBlockIndex*,int>> txChanged;
3023  +   if (pblock)
3024  +       txChanged.reserve(pblock->vtx.size());
3022      do {
3025      do {
3026  +       txChanged.clear();
3023          boost::this_thread::interruption_point();
3027          boost::this_thread::interruption_point();
3024          if (ShutdownRequested())
3028          if (ShutdownRequested())
3025              break;
3029              break;
3026
3030
3027          const CBlockIndex *pindexFork;
3031          const CBlockIndex *pindexFork;
3028          std::list<CTransaction> txConflicted;
3032          std::list<CTransaction> txConflicted;
3029  -       std::vector<std::tuple<CTransaction,CBlockIndex*,int> > txChanged;
```

Full Fix

```
2803  + * Used to track conflicted transactions removed from mempool and transactions
2804  + * applied to the UTXO state as a part of a single ActivateBestChainStep call.
2805  + */
2806  +struct ConnectTrace {
2807  +    std::vector<std::shared_ptr<const CTransaction>> txConflicted;
2808  +    std::vector<std::pair<CBlockIndex*, std::shared_ptr<const CBlock> > > blocksConnected;
2809  +};
```

```
2806  -bool static ConnectTip(CValidationState& state, const CChainParams& chainparams, CBlockIndex* pindexNew, const CBlock* pblock, std::vector<std::shared_ptr<const CTransaction>> &txConflicted, std::vector<std::tuple<CTransaction,CBlockIndex*,int>> &txChanged)
```

```
2815  +bool static ConnectTip(CValidationState& state, const CChainParams& chainparams, CBlockIndex* pindexNew, const std::shared_ptr<const CBlock>& pblock, ConnectTrace& connectTrace)
```

CONTRIBUTING

Important Contributions to Bitcoin

- Documentation
- Novel ideas and research
- Review others' code & ideas
- Rigorous Testing
- Conference/Community organizing
- Tools for developers
- **Write new code!**

Write Good Code

1. Find an issue that you think is important
2. Write
    1. Patches that you think solves it
    2. Clear documentation
    3. Tests that cover the code
3. Push to a branch on your fork

Seek Early Feedback

- Write a message to:
  - A Bitcoin contributor who works on similar things
  - TheBlueMatt, theuni, jonasschnelli, and myself are friendly default contacts
  - ping #bitcoin-core-dev with a request for feedback
- Be gracious! Negativity on your work is not negativity to you!

Run New Nodes

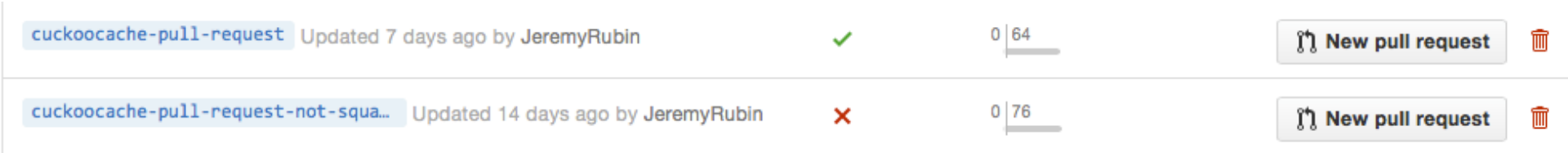
- Main and test net
  - Recommend using a different server
  - Compare debug logs to compare to your default nodes. Is your version better/correct?

Restructuring Code Changes

- Tests should be a separate commit
  - Either ‘Tests after Code’ or ‘Code after Tests’
- Small commits that individually make sense
  - Sometimes you rewrite code you don’t actually need to make that work
- Don’t pack too much in one pull-request
  - Better to leave a 10x improvement to an initial 2x and a follow up 5x later
- `git rebase –i` is your friend

Open a Pull-Request on Github

- Make sure Travis is enabled and passing on your fork for your branch



- Write up a few paragraphs motivating and describing the changes
- Look at other merged PRs for examples

Waiting Game

- Review takes time!
- Respond to feedback as you get it

Your First Contribution

- Experience Bitcoin PR process with low stakes trial
- Try to
  - Add documentation
  - Add tests
  - Fix typos

GENERAL ADVICE

Good to Read

- Bitcoin SoK Paper [Felten et al.]
- Pull Requests/Issues on Repo
- Bitcoin Improvement Proposal notes [BIPs]
- Kanzure's Archive [diyhpl.us/~bryan/papers2/bitcoin]
- Peter Todd's Blog [petertodd.org]
- Computer Systems Security (6.858, 6.857, 6.875) [css.csail.mit.edu]
- Game Theory with Engineering Applications [OCW]
- Follow CCS/Oakland/Crypto/FC/… conferences
- Twitter

Bad to Read

- Mostly Avoid:
  - Comments on reddit
  - Articles in NYT/Economist/etc about Bitcoin development
- There's enough "Good to Read" to keep you busy and happy!

Socialize

- Lots of really great people to talk to!
- Go to Scaling Bitcoin conference
- Local Bitcoin Meetups (SF Bitcoin Devs)
- Twitter

Be Patient

- Bitcoin is **security** focused software
  - Development will be **slow**…
  - Developers will **nitpick** your code…
  - Broken code will **upset** you…
  - but… Your work is **high impact!**

THANKS