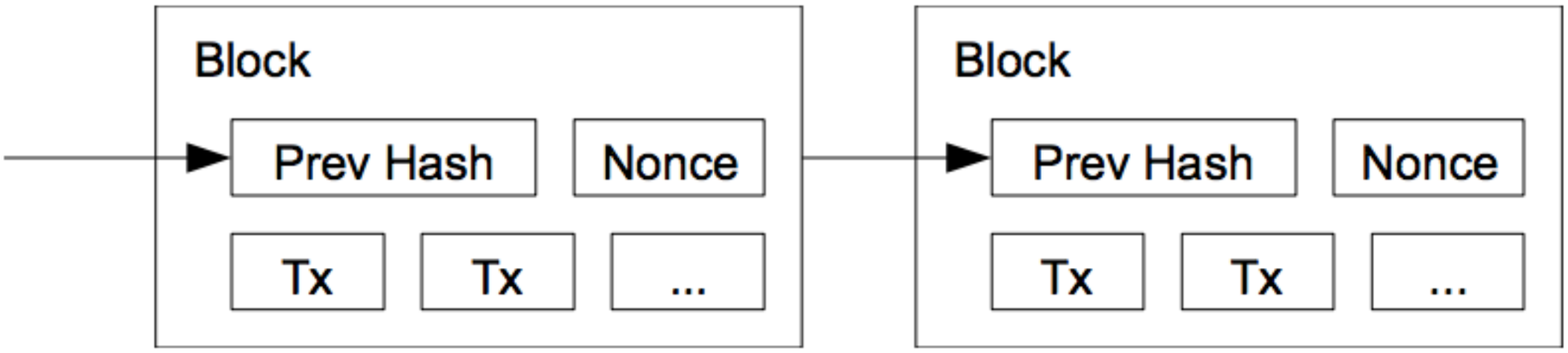
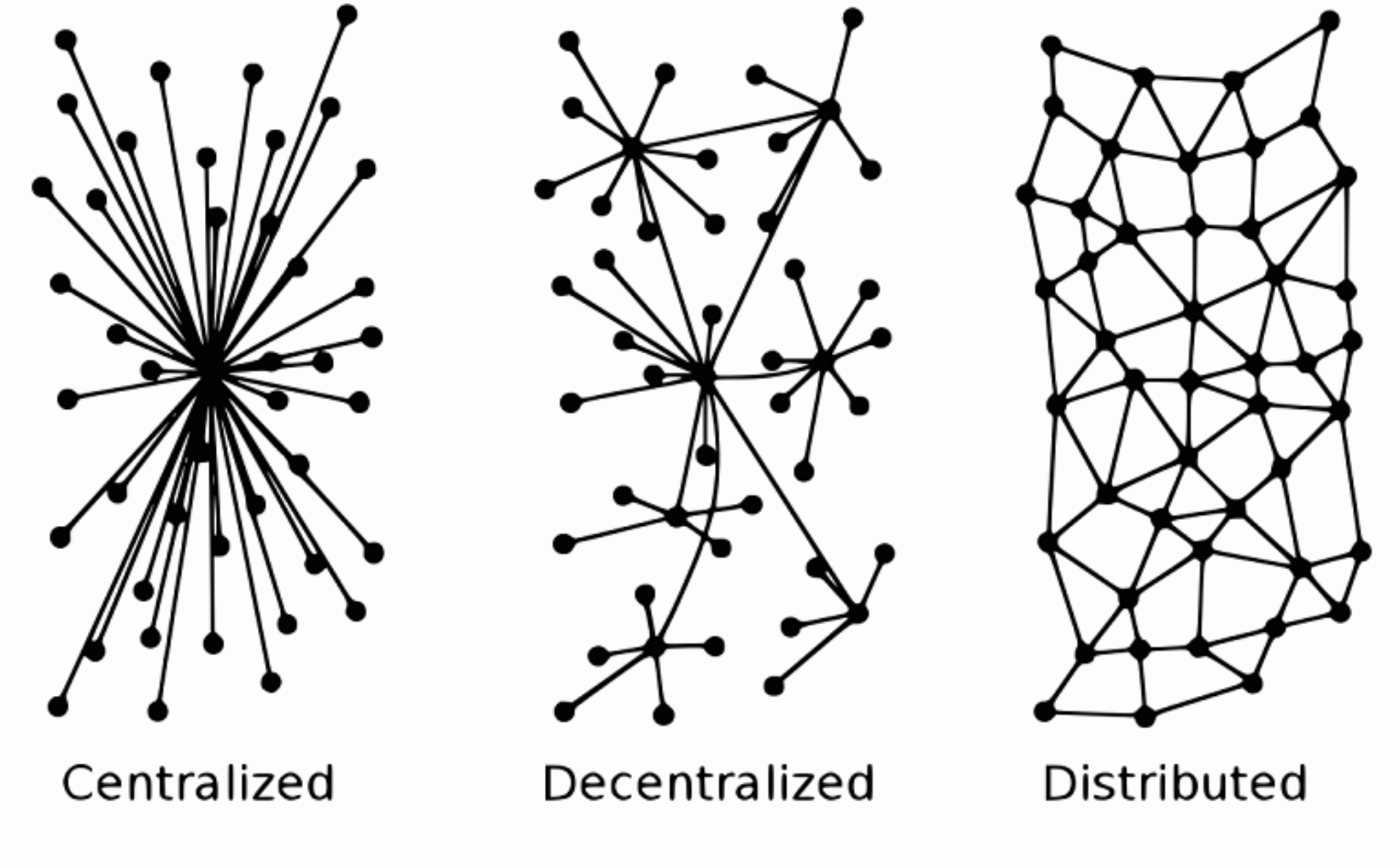
Introduction (Brief)

Blockchain



- Writes all transactions (ordered by fee)
- Hashes are easily verified

Decentralization



What are Smart Contracts?

- Maintains a data store
- As an externally owned account with a more complicated access policy
- Manage an ongoing contract or relationship between multiple users
- Provide functions to other contracts

Gas

- Every single operation that is executed inside the EVM is actually simultaneously executed by every full node

- In order to prevent deliberate attacks and abuse, the Ethereum protocol charges a fee per computational step

Synapse AI

AI economies on the blockchain

Synapse AI - What

- https://synapse.ai/
- Decentralized AI Network
- Data is used to build AI
- AI agents can interact, negotiate, and contract w/ one another
- We create AI economies at scale

Synapse AI - Vision

- Global AI brain networked to all other agents and devices
- Global autonomous logistics and supply chain
- Decentralized network facilitating smarter apps, bots, robots, humans

Synapse + Ethereum

- On-chain: Access control, Provenance (Data Lineage), Contracting/Provisioning

- Off-chain: Oracles, Data store, Compute

Synapse AI - Next

- Marketplace Alpha is Launched
- Running Bug Bounty program - $$$
- Partnership program - $$$
- Developer program - $$$
- https://synapse.ai/marketplace

Synapse AI - Next Next

- Light Clients
- ICO Tier II
- Partnerships (Data/ML)

Drum Roll…..

(finally)

Introduction to Ethereum's Solidity

Smart contract & Dapp programming

Table of Contents

Tools

Editors

Tools

https://github.com/tomlion/vim-solidity

Truffle

Tools

https://trufflframework.com

Testrpc

Tools

https://github.com/ethereumjs/testrpc

OpenZeppelin

Tools

https://openzeppelin.com

Web3.js

Tools

https://github.com/ethereum/web3.js

Remix

Tools

https://ethereum.github.io/browser-solidity/

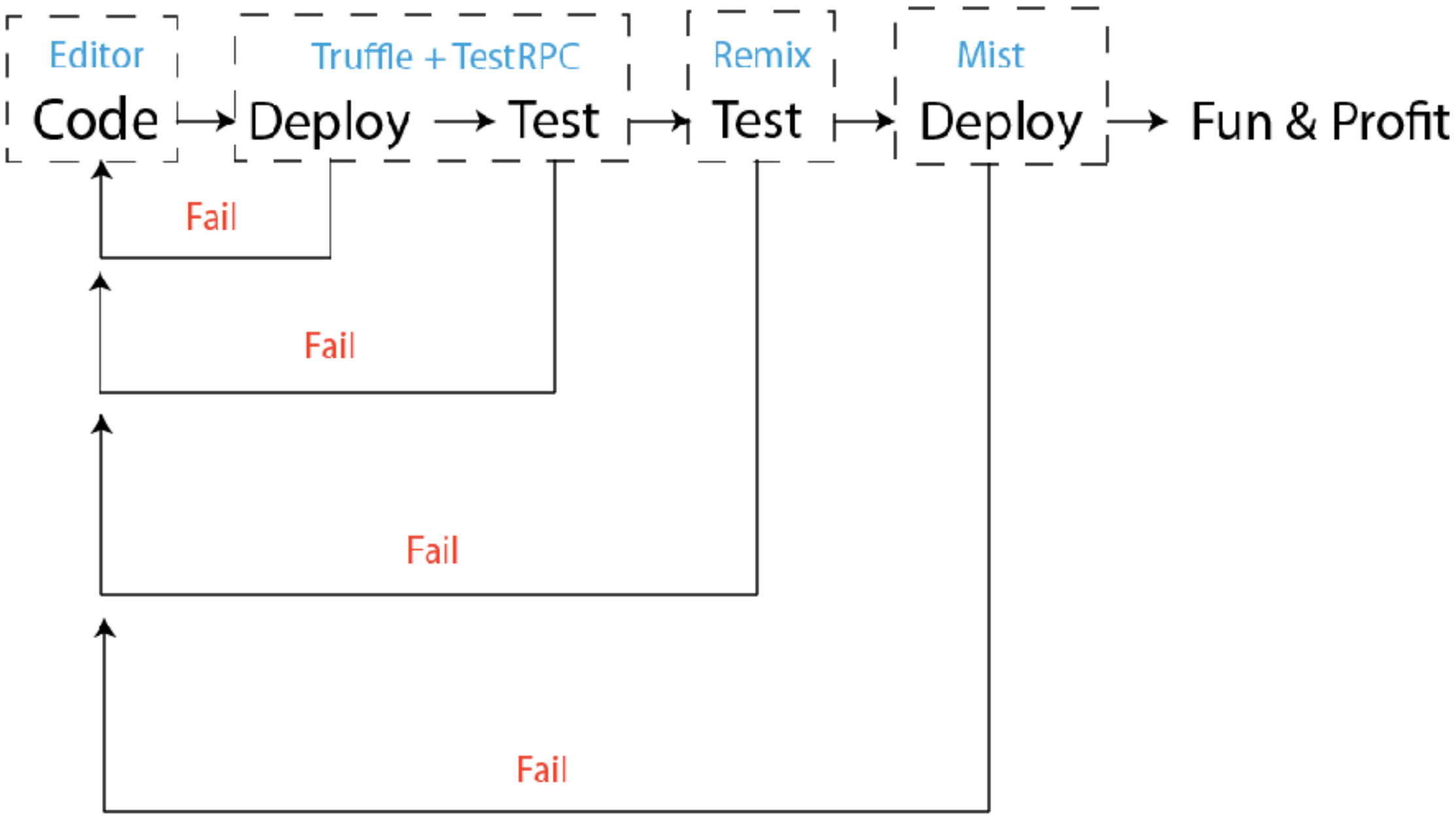
Dapp

Tools (Honorable Mention)

https://dapp.readthedocs.io

Workflow

Tools



Introduction

Ethereum’s Solidity

Sample Code

```
pragma solidity ^0.4.0;

contract SimpleStorage {
    uint storedData;

    function set(uint x) {
        storedData = x;
    }

    function get() constant returns (uint) {
        return storedData;
    }
}
```

Initialization

- **Pragma** - pragmas are instructions for the compiler about how to treat the source code

- **Contract** - A contract in the sense of Solidity is a collection of code (its *functions*) and data (its *state*) that resides at a specific address on the Ethereum blockchain.

- `uint storedData;` declares a state variable called `storedData` of type `uint` (unsigned integer of 256 bits).

Variables & Types

- Function Scoped (Global / Local variables)
- uint (unsigned)
- address (20 byte)
- enums, structs
- arrays (notation is reversed)
  As an example, an array of 5 dynamic arrays of uint is uint[][5]
  Can also be marked `public` to create a getter
- bytes, strings: special arrays (bytes cheaper than bytes[])
- constant - constant state variables

Function Structure

```
function (<parameter types>) {internal|external} [pure|constant|
view|payable] [returns (<return types>)]
```

Function Types

- **Internal** (default) - can only be called inside the current contract.
- **External** - can be passed via and returned from external function calls.
- (Calling a function `f` is internal `this.f` is external)

- **Contract functions themselves are `public` by default, only when used as the name of a type, the default is internal.**

Function Types Cont’d

- **Pure:** promise not to read from or modify the state.
- **View:** promise not to modify the state.
- **Constant:** (alias to view)

Notes on `constant`

- Functions that have `constant` typically read data, therefore the functions require a `return`

- Functions without `constant` typically write data therefore the function doesn’t need a `return`

Mappings

- **Syntax**:
  mapping(<KeyType> => <ValueType>) <variable>

- **KeyType** (keccak256): can be almost any type except for a mapping, a dynamically sized array, a contract, an enum and a struct.

- **ValueType**: can actually be any type, including mappings.

- **Default Values**: all zeros

- Only State Variables

- Not iterable (sorta)

Structs

```
struct Campaign {
    address beneficiary;
    uint fundingGoal;
    uint numFunders;
    uint amount;
    mapping (uint => Funder) funders;
}

uint numCampaigns;
mapping (uint => Campaign) campaigns;
```

Special Variables & Functions

- **msg.value:** Amount of WEI being sent
- **msg.sender:** Address of agent sending message (initializer)
- <address>.balance
- <address>.transfer(<amount>);
- selfdestruct(address recipient)

MOST Special of Functions

- function () payable {
  }
- a function has no name
- default function for contract (fall back)
- process payments sent to contract by default here

Modifiers

- **Modifiers** modify (wrap) a function

- **Example**:
  function withdrawAllTheMoney() public onlyOwner {
    msg.send(balance); // pr0tect me plz
  }

- modifier onlyOwner {
    if(msg.sender != owner) {
      revert();
    }
    _; // This is where the function goes
  }

Events

- **Syntax:** event Name (<types>)

- Used as Debug/Logging/Event watching

Pro-tips

- Explicitly declare types
- Explicitly convert types
- integer overflows, underflows (uint256 == 2**256)

Pro-tips (2)

- All denominations in WEI (1ETH = 10E-18 WEI)

Pro-tips (3) Returning strings from functions

- “You see this BigNumber because you are returning 'string'. Solidity doesn't recommend using any variable length return values like arrays or string which can be less or greater than 32 bytes. Try changing that type to a bytes32 instead of string and on client side use web3.toUTF8(returnValue). This has worked for me. Similar info here https://forum.ethereum.org/discussion/8053/is-it-possible-to-return-a-string-from-a-function“ -Shashikant Soni (gitter: ethereum/remix)

BORING!

SHOW ME THE MONEY

Put this code into Remix:
https://github.com/SynapseOrg/Introduction-to-Solidity/blob/master/TokenSale.sol

Contract Deployment

How to money?

- **EthLance!** https://ethlance.com/

…but I need help?

- **Good (slow):**
  https://solidity.readthedocs.io/
  https://www.reddit.com/r/ethdev/

- **Good (fast):**
  https://gitter.im/ConsenSys/smart-contract-best-practices
  https://gitter.im/ConsenSys/truffle
  https://gitter.im/ethereum/solidity

- **Not As Good:** https://gitter.im/ethereum/remix

- **Not As Good:** #eth-dev on IRC

…but I need MOAR halp

- **Come to Hack Days or Contact me!**
- **Twitter:** @dpg
- **Github:** dpgailey
- **Facebook:** Dan Gailey
- **Email:** dan@synapse.ai

Resources?

- Ethereum Dev Tutorial: https://github.com/ethereum/wiki/wiki/Ethereum-Development-Tutorial

- Cost of a real world contract: https://hackernoon.com/costs-of-a-real-world-ethereum-contract-2033511b3214

- EthGasStation: https://ethgasstation.info/

- EthStats: https://ethstats.net/

What’s next?

- Build Dapps
- Decentralize everything
- Disrupt existing monopolies
- Profit

Synapse.ai

Decentralized Data + AI

hello@synapse.ai