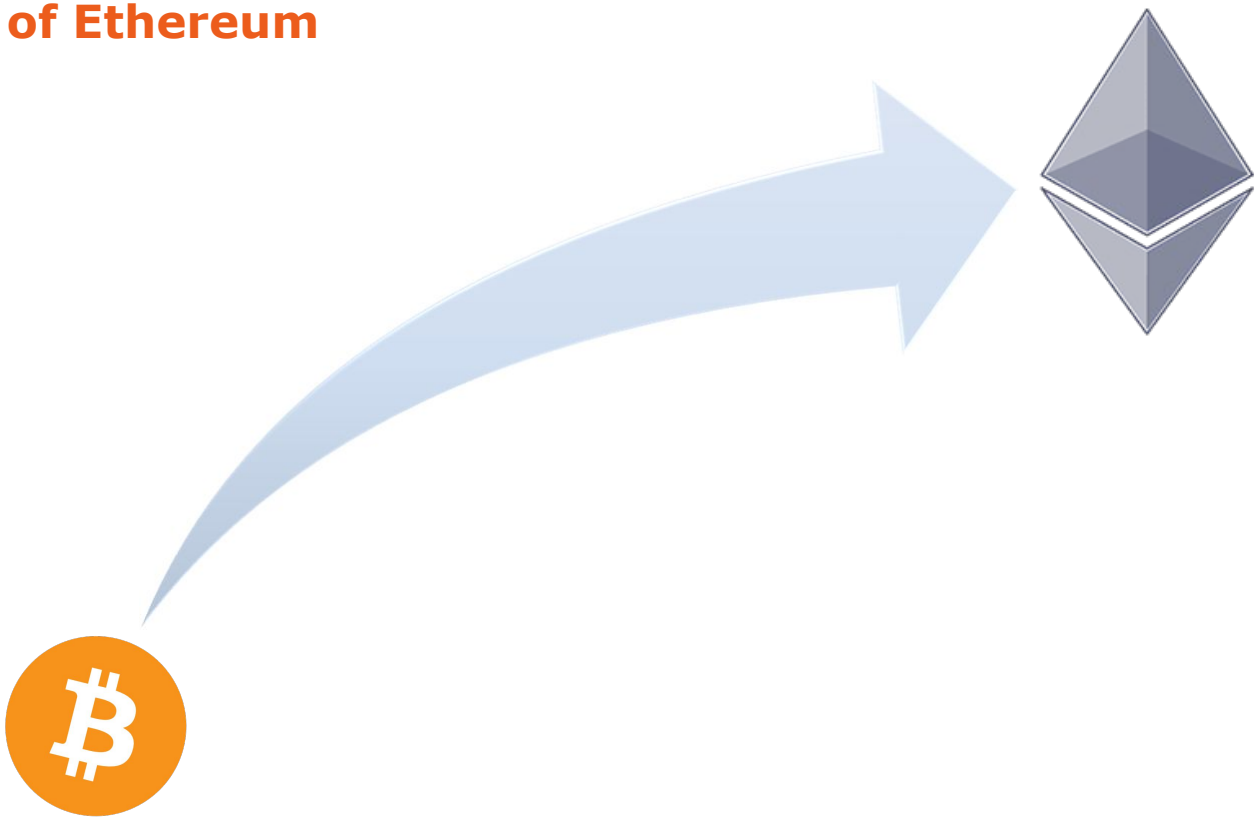
Getting Started with Ethereum

Blockchain Monterrey

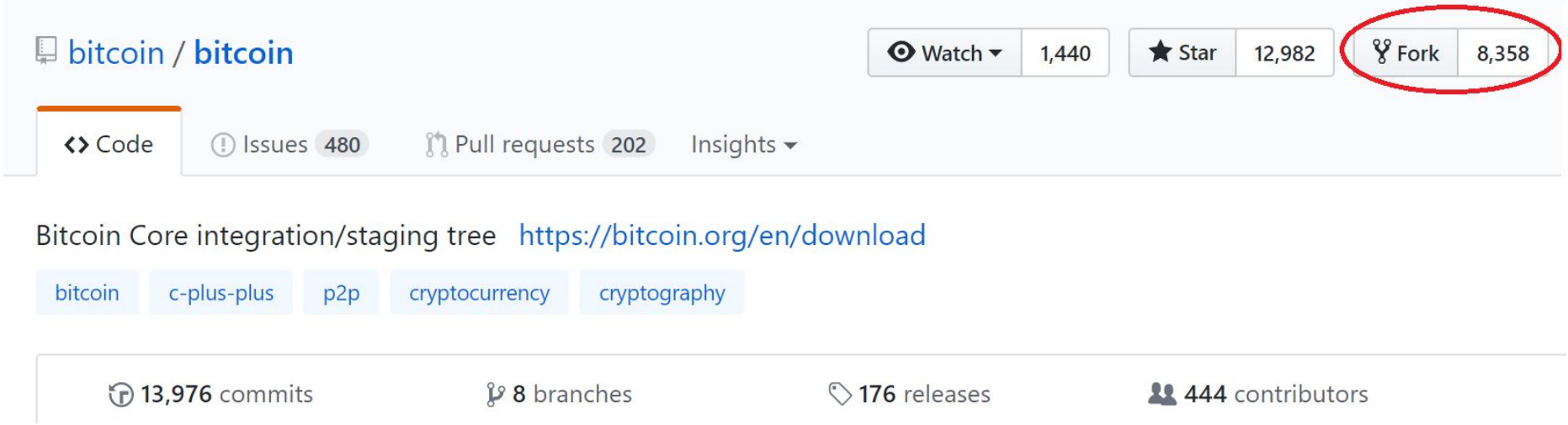
Ethereum Protocol Overview

History of Ethereum



3

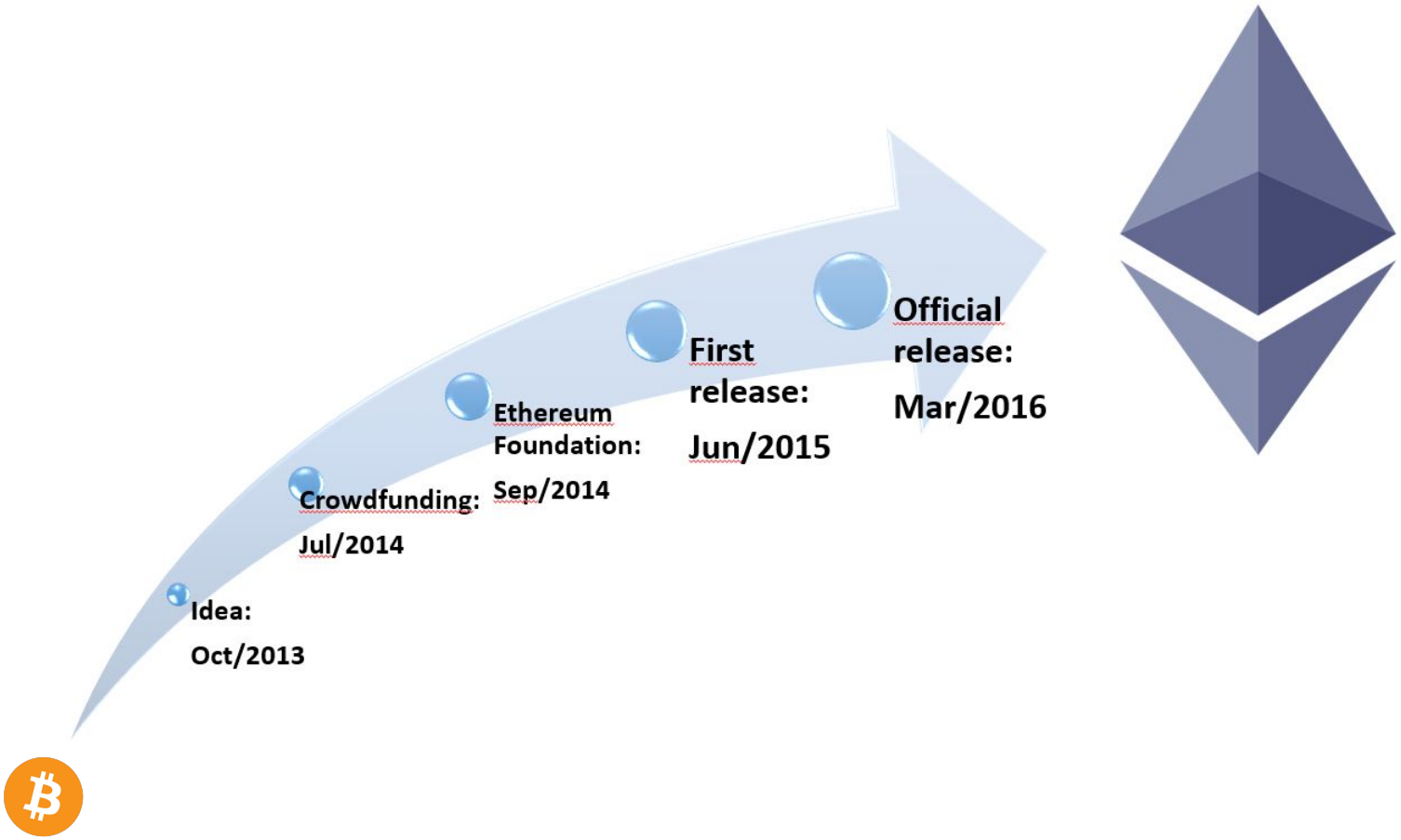
History of Ethereum



History of Ethereum

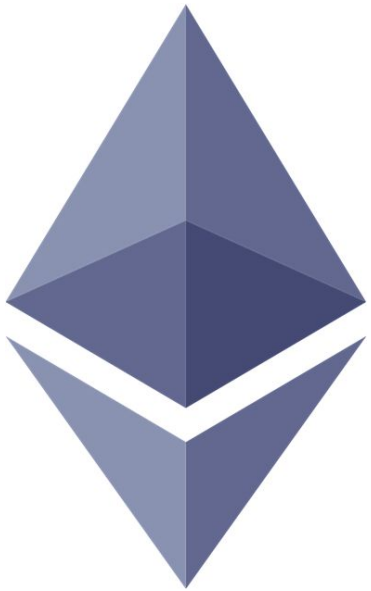
History of Ethereum

History of Ethereum



Getting to Know Ethereum

- Smart Contract platform based on a Blockchain
- Three Built-in programming languages for Smart Contracts
- Strong use of cryptography



Important Concepts

- Cryptography
- Blockchain
  - Accounts and Wallets
  - Transactions
- Smart Contracts
- Smart Contract Programming Language

- **Cryptography**
- Blockchain
  - Accounts and Wallets
  - Transactions
- Smart Contracts
- Smart Contract Programming Language

Cryptography

- Hash functions
- Symmetric Cryptography
- Asymmetric Cryptography
- Signatures

Hash Functions

Artist Statement
Eun Young Choi

Dislocation is a challenge I struggled with all my life. As a daughter of a diplomat, I lived and traveled to many parts of the world. Moving to different countries and cultures every two or three years required that I quickly adjust to a new environment and still maintain a sense of identity and balance. I cannot say that living in different cultures has had a direct influence in my work, but the idea of dislocation and transformation is always present in one form or another.

Storytelling is another element that I play with in my work. It originates in speech and language to give form to concepts, emotions, and desires. I gather bits and pieces of stories and contain them in little containers whether it be a room, a bottle, a zip lock bag or the words themselves. Like insects held captive in a web, the remnants of life, memories and stories are caught and accumulated until they grow into an entirely new entity. It is about little fragments becoming a whole, and in the process, going through transformation and translation in both the physical and intellectual realm.

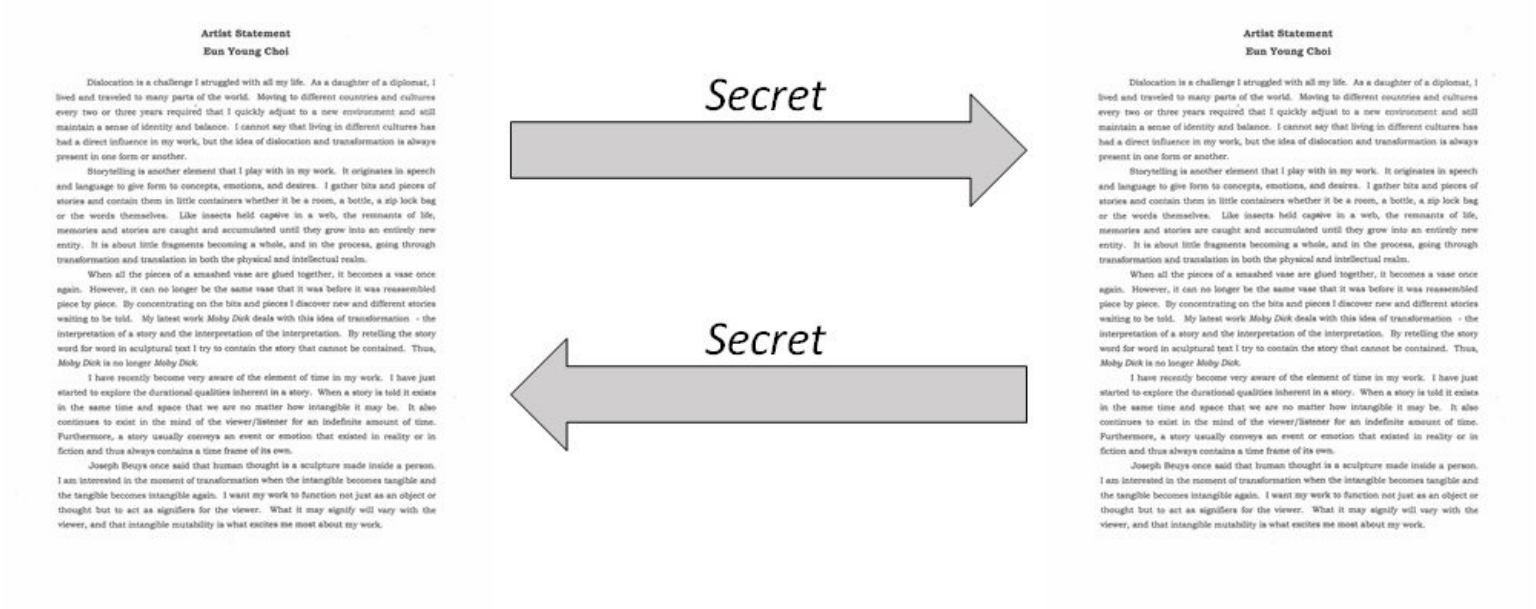
When all the pieces of a smashed vase are glued together, it becomes a vase once again. However, it can no longer be the same vase that it was before it was reassembled piece by piece. By concentrating on the bits and pieces I discover new and different stories waiting to be told. My latest work *Moby Dick* deals with this idea of transformation - the interpretation of a story and the interpretation of the interpretation. By retelling the story word for word in sculptural text I try to contain the story that cannot be contained. Thus, *Moby Dick* is no longer *Moby Dick*.

I have recently become very aware of the element of time in my work. I have just started to explore the durational qualities inherent in a story. When a story is told it exists in the same time and space that we are no matter how intangible it may be. It also continues to exist in the mind of the viewer/listener for an indefinite amount of time. Furthermore, a story usually conveys an event or emotion that existed in reality or in fiction and thus always contains a time frame of its own.

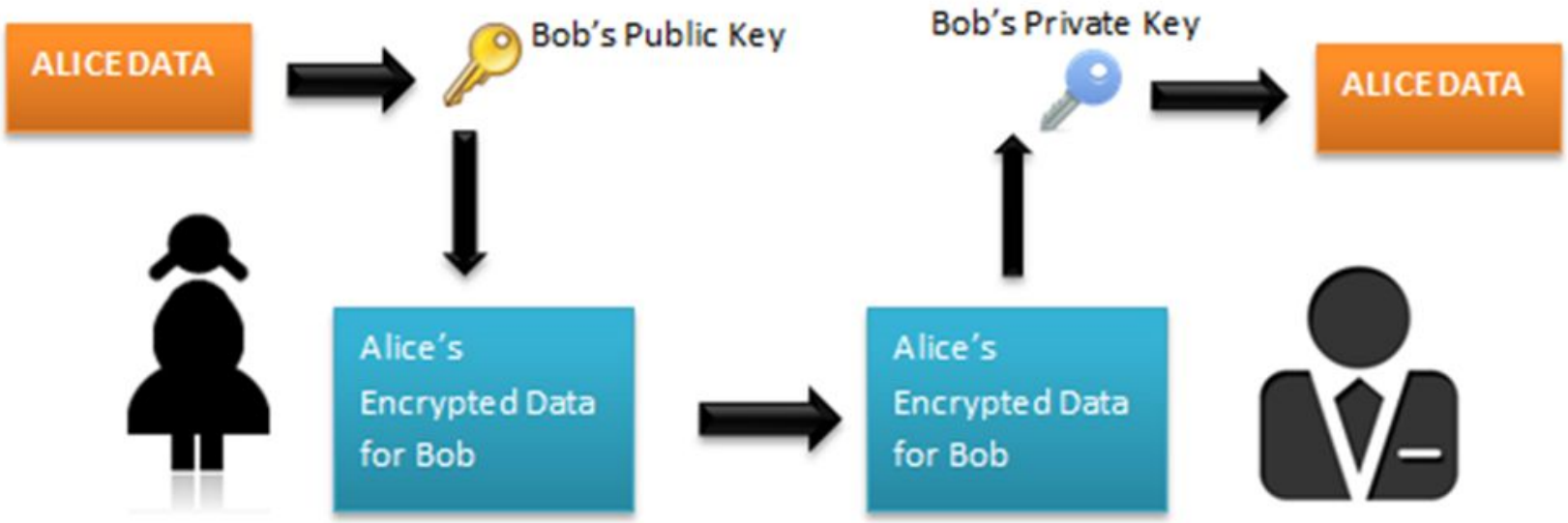
Joseph Beuys once said that human thought is a sculpture made inside a person. I am interested in the moment of transformation when the intangible becomes tangible and the tangible becomes intangible again. I want my work to function not just as an object or thought but to act as signifiers for the viewer. What it may signify will vary with the viewer, and that intangible mutability is what excites me most about my work.

e3f275bf52ac8e2fbd629c40dff9c29e86997d

Symmetric Cryptography



Asymmetric Cryptography



RSA vs. ECC

RSA

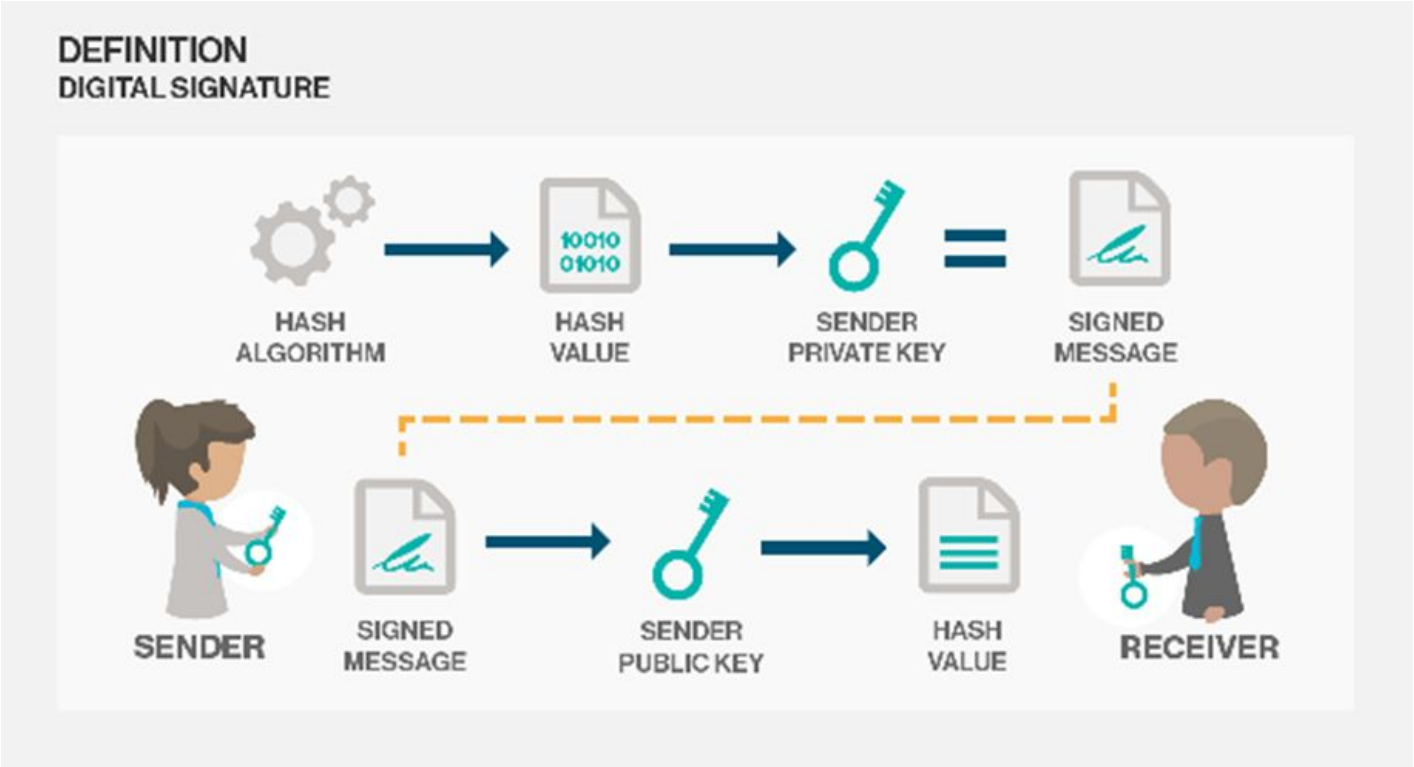
```
-----BEGIN RSA PUBLIC KEY-----
MIICWwIBAAKBgQDEBbsnm/Mvi6LKt9MQY4NxI203W/gnXZWWfiUFsY64ZCbJSze1
LMS/N0M9GTWEcRluT3dUQ1H5qlgGJucWU4URaM+7HhENMiH1Z6CPzpQ7s8fm2/HI
t7UhkCkAzx7+WjvI3aHgbT9KM/GaIKDQblXkQE5x5AqKHK3dL8Lq/sVqGQIDAQAB
AoGAZIi78BifpObs2yNkcKdZJpGQ1i+pGid5LPINctoZ5KAY9GxLafRIA/oH1jIz
rHWFIXpILUNJq86dzqntIxo6yEhKAoiq4r9plwNpp5T1KbpUOlUTEQworqH6EQvZ
BvIb80Mfmm5ka/3saRr1xidpJCLUyLejxFJxJ6XLUhPNfbECQQD1T1yt0eeiTDEa
8pMHLfEbCise4olfl+Udc8zWSHpT83i5re4dctV5P3z9tjFXkA38V70JI3RuRzir
lw92uHijAkEAzJCGREY/5ZehKlcT2gnmbjomDXJlVyBKt0900W42MTdswSHQwRPR
atfWVv2AXvNTPAFXoMVrups1uNK/W5MSEwJAd1v2LF1F56JLAmGlSIkCF7YK5Sw0
y16NDJebw2fgnZiJ1U7b3VhSpnxNxOUxfPmK1I2cXSXzMPVWjacizxFTIQJAA0Ft
kZqYm7vNCdJ282pi63AreN1QNZHC/qXaExcw75mVNoGmQ9xf4dZrh9ji+R/gPD09
OsbJjx+3PCjGeNufVwJAMhvDZpCyAbOHH5A7eqnizJAUiNUy3utv2l2QHSIZFiqh
+42oTgs2+AIh2j8Rn/5HCDLby0cOSTQTi0hRkyQaHA==
-----END RSA PUBLIC KEY-----
```

ECC

```
1GTWEcRluT3dUQ1H5qlgGJucWU4URaM
```

Same level of security

Digital Signatures

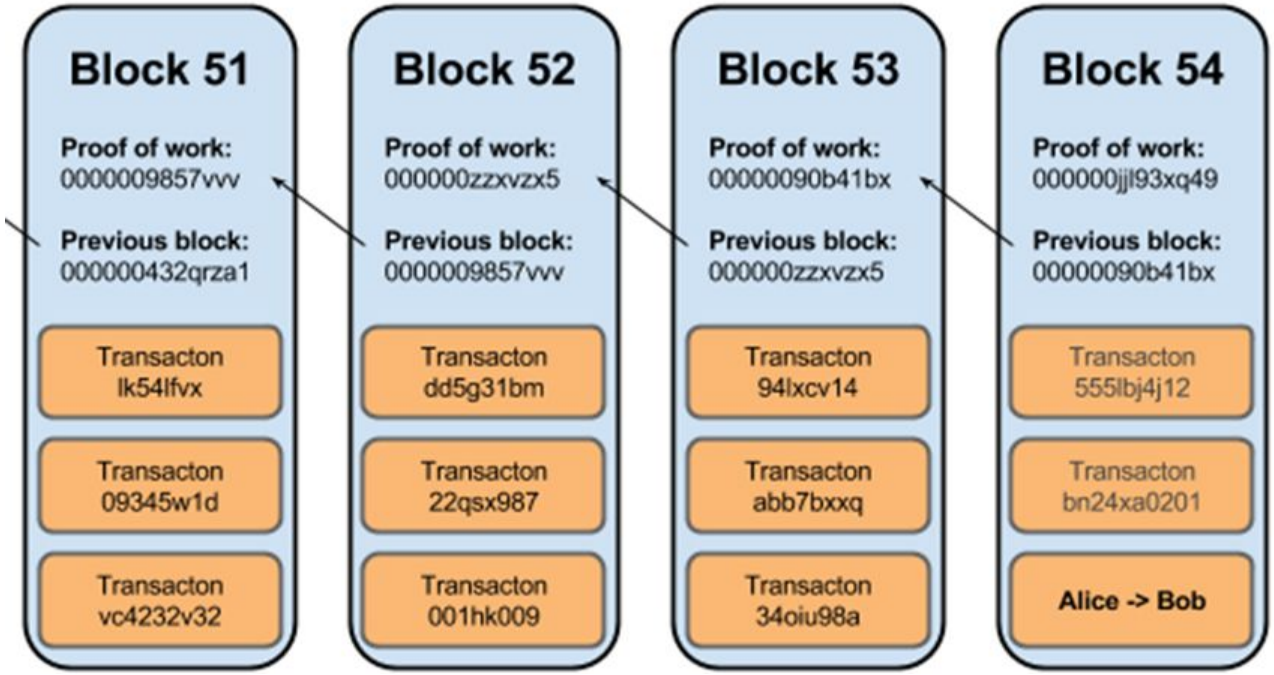


- Cryptography
- **Blockchain**
  - Accounts and Wallets
  - Transactions
- Smart Contracts
- Smart Contract Programming Language

A weird Database:

- Immutable
- Unstoppable
- Transparent
- Tamper proof
- Very secure
- Fully distributed

Blockchain Structure



PROOF OF WORK (POW)

- Refers to a piece of data which is difficult to produce but simple to verify (the nonce)
- Participating users work (mine) to solve difficult mathematical problems and publish their solutions to the blockchain
- Requires real-world resources:
  - Computing power (hashing/mining power)
  - Electricity
  - Time

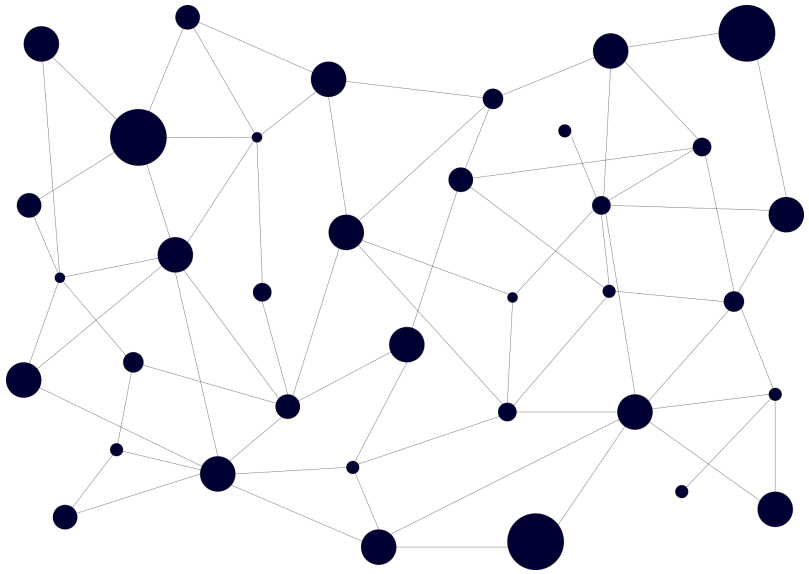
Proof of Work (PoW) - A Bitcoin Mine

Proof of Stake (PoS)

- Person can “mine” depending on how many coins(ETH) they hold.
- Advantages:
  - Lower inflation / network expenditure
  - Lower energy consumption – “virtual mining”
  - Increased scalability and efficiency
  - 51% attacks are more expensive
  - Increased decentralization by community

Blockchain Nodes

- Validates transactions
- Peer to Peer (no central authority)
- Has complete data and history of all transactions



Public vs. Private

- Public blockchain:
  - Permissionless
  - Everyone can participate
  - All data are visible
- Private blockchain:
  - Permissioned
  - Closed to general public
  - Privacy of data

- Cryptography
- Blockchain
  - **Accounts and Wallets**
  - Transactions
- Smart Contracts
- Smart Contract Programming Language

Accounts and Wallets

- Accounts:
  - A pair of private key and public key
  - Allows interaction with a blockchain
  - Are the blockchain actors
- Wallets:
  - A set of one or more accounts
  - Ex: HD Wallet

⇒ Both can be generated off-line ⇐

HD Wallets

- Hierarchical Deterministic Wallets
- Through a **seed**, generates many addresses deterministically
- A password is used to encrypt your wallet

Example of an Account

Private Key:

0x2dcef1bfb03d6a950f91c573616cdd778d9581690db1cc43141f7cca06fd08e

Address:

0xA6fA5e50da698F6E4128994a4c1ED345E98Df50

Collision Probability

- 2^{160} (privkey of 160 bits)
- 1 billion = 2^{30}
- 1 supercomputer generating 1 billion random privkeys per second => $2^{160} : 2^{30} = 2^{130}$
- 2^{32} world population each with 2^{30} supercomputers => 2^{98}
- 1 year = 2^{35} seconds
- 2^{98} seconds = 2^{63} years
- Universe age = 2^{34}

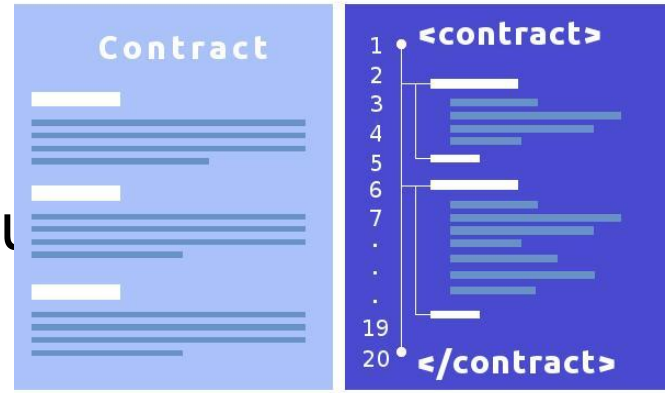
- Cryptography
- Blockchain
  - Accounts and Wallets
  - **Transactions**
- Smart Contracts
- Smart Contract Programming Language

- A request to modify the state of the blockchain
- Signed by originating account
- Types:
  - Send value from one account to another account
  - Create smart contract
  - Execute smart contract code

- Cryptography
- Blockchain
  - Accounts and Wallets
  - Transactions
- **Smart Contracts**
- Smart Contract Programming Language

SMART CONTRACTS

- Executable code
- Emulates the logic of contractual clauses
- Just like an account
  - Hold funds
  - Can interact with other accou contracts
- Always passive and reactive



CODE EXECUTION

- Every node has a Ethereum Virtual Machine (EVM) that executes smart contract codes
- **Every node on the blockchain processes every transaction and stores the entire state**
  - It’s bold because it’s important

- Halting problem
  - Cannot tell whether or not a program will run infinitely
- Solution: charge “fee” per computational step to limit infinite loops and stop buggy code
- Every transaction needs to specify an estimation on the amount of gas it will spend
- Amount not spent is refunded

GAS PRICE

- Gas price: current market price of a unit of Gas
- Dynamically set / can be set by the user
- Regulates load of the blockchain network

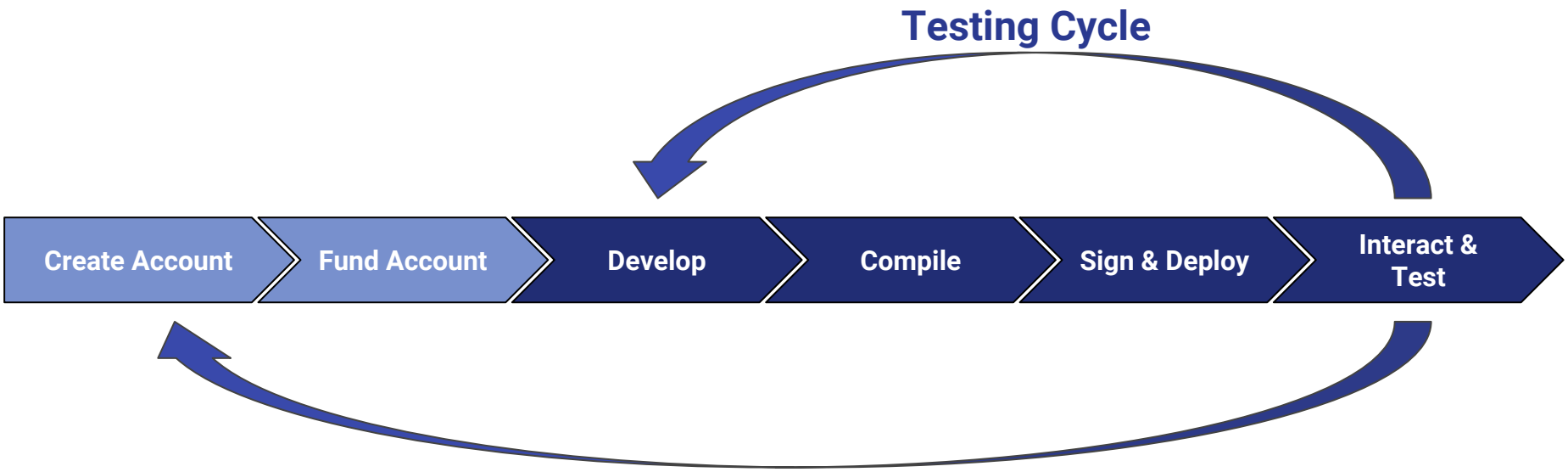
- Cryptography
- Blockchain
  - Accounts and Wallets
  - Transactions
- Smart Contracts
- **Smart Contract Programming Language**

LANGUAGES AVAILABLE

- Solidity (javascript based)
- Serpent (python based)
- LLL (lisp based)

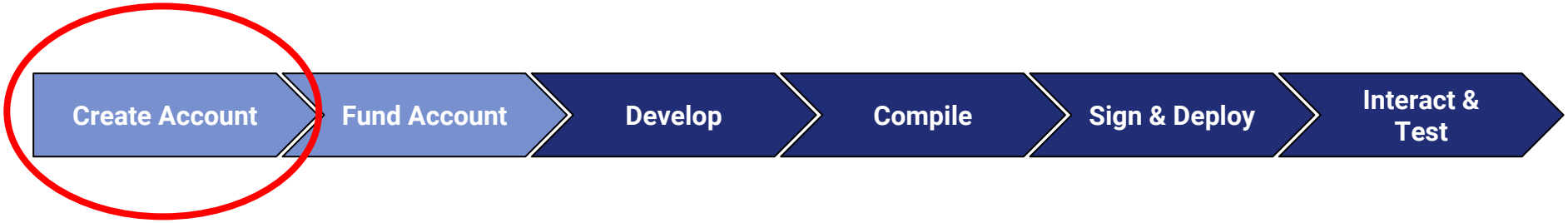
Development Workflow

Development Workflow



- Onboard Additional Users
- Create New Accounts
- Develop New Applications

Development Workflow: Create Account

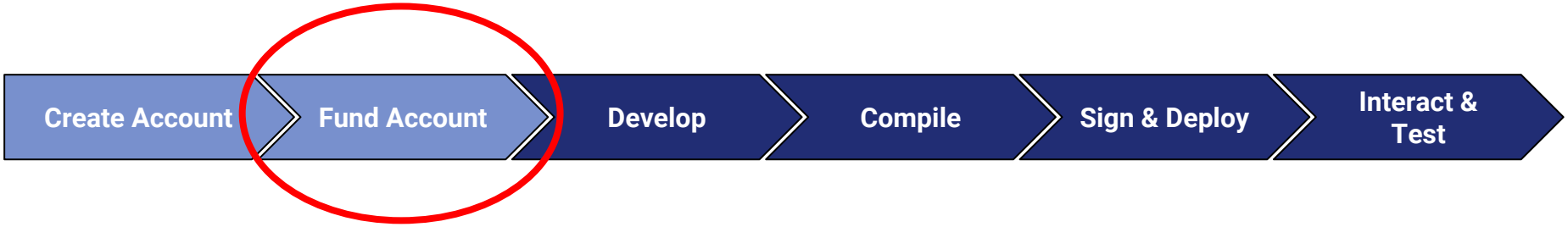


- Programmatically: Go, Python, C++, **JavaScript**, Haskell
- Tools
  - MyEtherWallet.com
  - MetaMask
  - TestRPC
  - Many other websites

ETHEREUM CLIENT LANDSCAPE

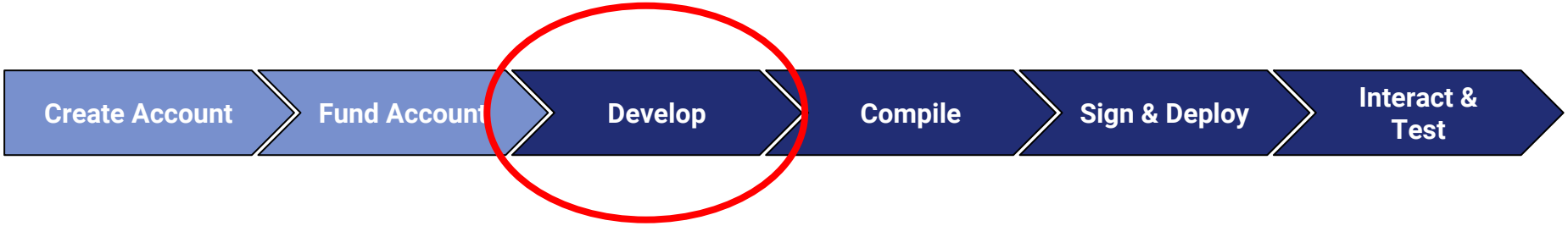
Client	Language	Latest release
go-ethereum	Go	go-ethereum-v1.4.18
Parity	Rust	Parity-v1.4.0
cpp-ethereum	C++	cpp-ethereum-v1.3.0
pyethapp	Python	pyethapp-v1.5.0
ethereumjs-lib	Javascript	ethereumjs-lib-v3.0.0
Ethereum(J)	Java	ethereumJ-v1.3.1
ruby-ethereum	Ruby	ruby-ethereum-v0.9.6
ethereumH	Haskell	no Homestead release yet
Testrpc (only for dev/test)	Javascript	v3.0.0

Development Workflow: Fund Account



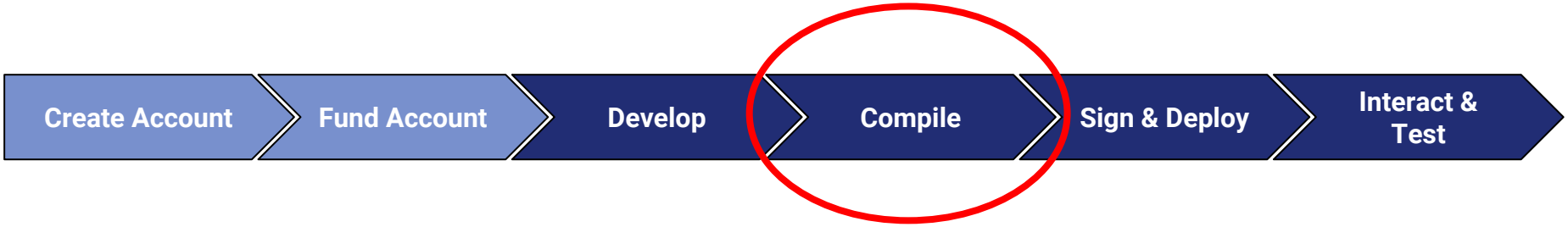
- From friends
- **Faucet**
- Exchanges (for public blockchain)

Development Workflow: Develop



- **An Ethereum application has three main parts:**
  - **Base application**: can be developed in **any** language
  - **Smart contract**: developed in one of the three supported languages, Solidity being the most used.
  - **Connector library**: allows the base application to talk to the smart contracts

Development Workflow: Compile



Compilation
Smart Contract

Bytecodes

ABI

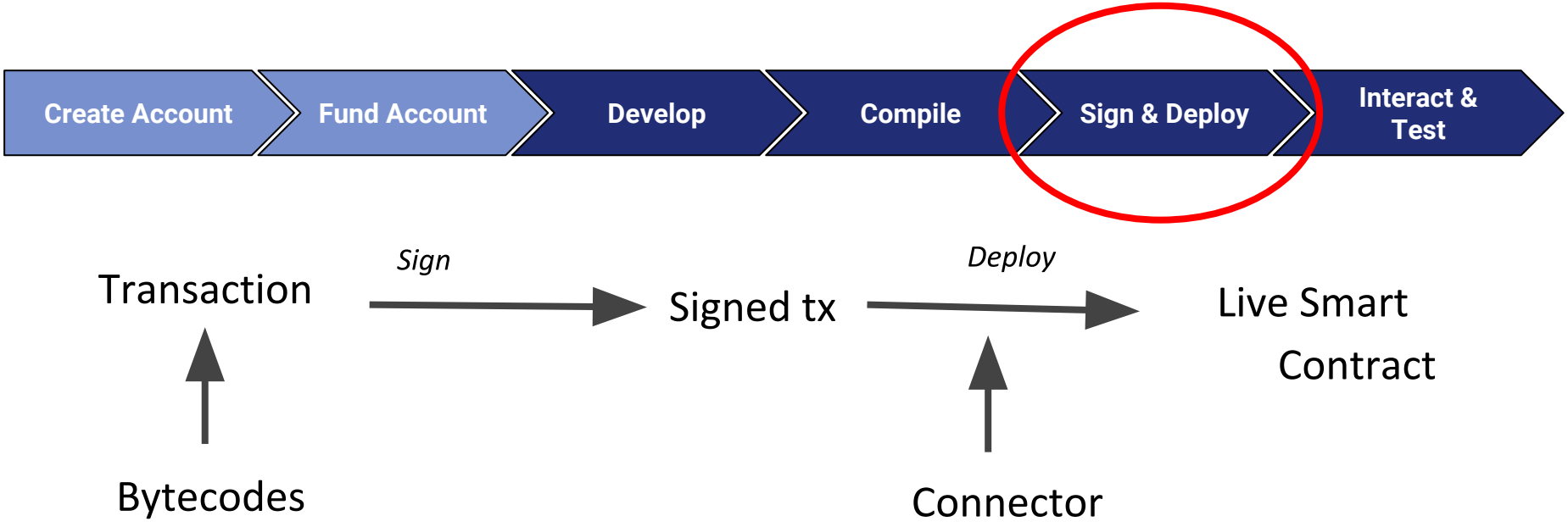
Native compiler

```
sudo add-apt-repository ppa:ethereum/ethereum
sudo apt-get update
sudo apt-get install solc
```

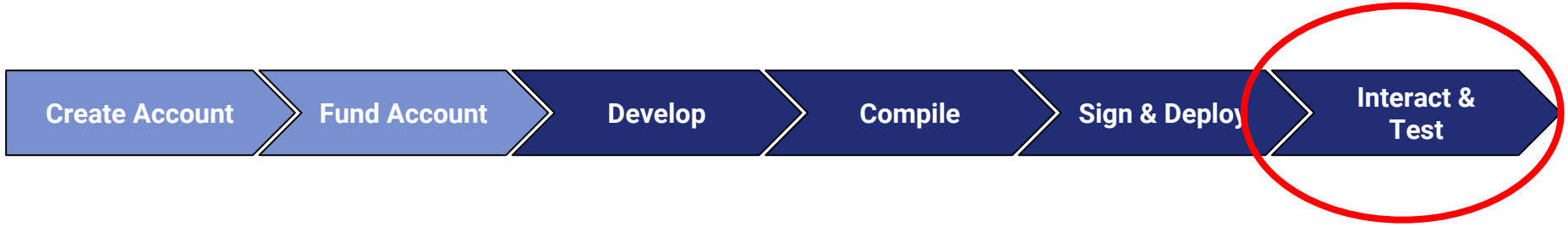
Javascript compiler

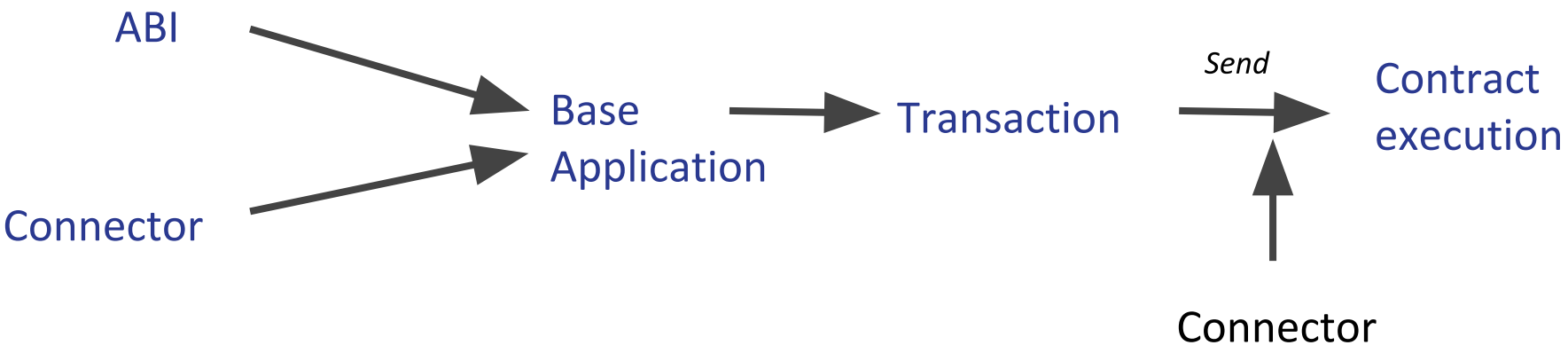
```
npm install solc

var solc = require('solc');
```

Development Workflow: Sign & Deploy



Development Workflow: Interact & Test

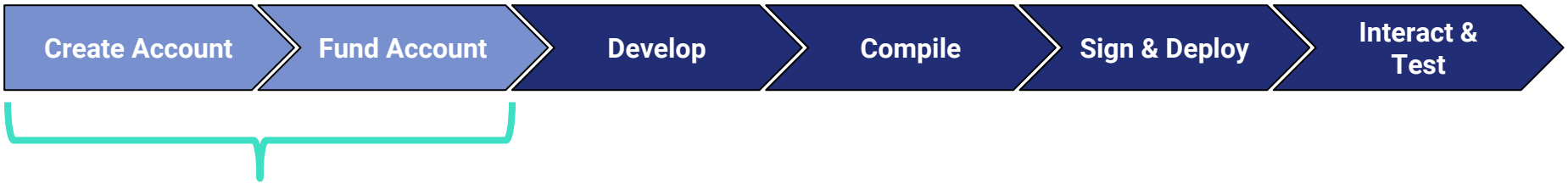




Ethereum Connector Landscape

Library	Language	Project Page
web3.js	JavaScript	https://github.com/ethereum/web3.js
web3j	Java	https://github.com/web3j/web3j
Nethereum	C# .NET	https://github.com/Nethereum/Nethereum
ethereum-ruby	Ruby	https://github.com/DigixGlobal/ethereum-ruby

Development Workflow: TestRPC



TestRPC

- Local development (not persistent)
- https://github.com/ethereumjs/testrpc
- Provides 10 pre-funded accounts

LOCAL ETHEREUM NODE FOR DEVELOPMENT

- EthereumJS TestRPC: https://github.com/ethereumjs/testrpc is suited for development and testing
- It's a complete blockchain-in-memory that runs only on your development machine
- It processes transactions instantly instead of waiting for the default block time – so you can test that your code works quickly – and it tells you immediately when your smart contracts run into errors
- It also makes a great client for automated testing
- Truffle knows how to use its special features to speed up test runtime by almost 90%.

Development Workflow: Platforms

TRUFFLE
- http://truffleframework.com/tutorials/
- Javascript based

EMBARK
- https://github.com/iurimatias/embark-framework
- Javascript based

DAPPLE
- https://github.com/NexusDevelopment/dapple
- Javascript based

POPULUS
- http://populus.readthedocs.org
- Python based

TRUFFLE

Truffle is a development environment, testing framework and asset pipeline for Ethereum, aiming to make life as an Ethereum developer easier. With Truffle, you get:

- Built-in smart contract compilation, linking, deployment and binary management.
- Automated contract testing with Mocha and Chai.
- Configurable build pipeline with support for custom build processes.
- Scriptable deployment & migrations framework.
- Network management for deploying to many public & private networks.
- Interactive console for direct contract communication.
- Instant rebuilding of assets during development.
- External script runner that executes scripts within a Truffle environment.

Thank You

Miguel De La Cruz - Blockchain Developer

miguel@privasee.io
ma@quetzichain.com