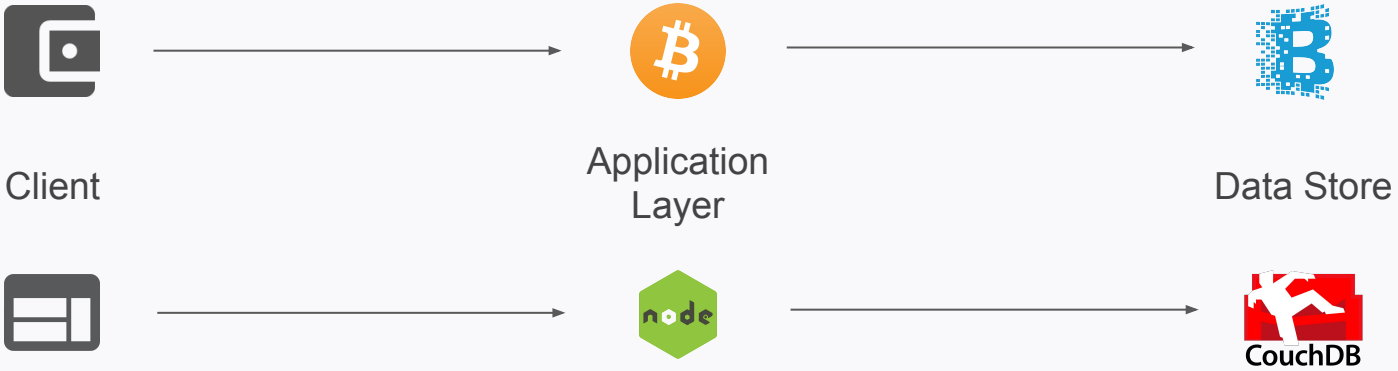
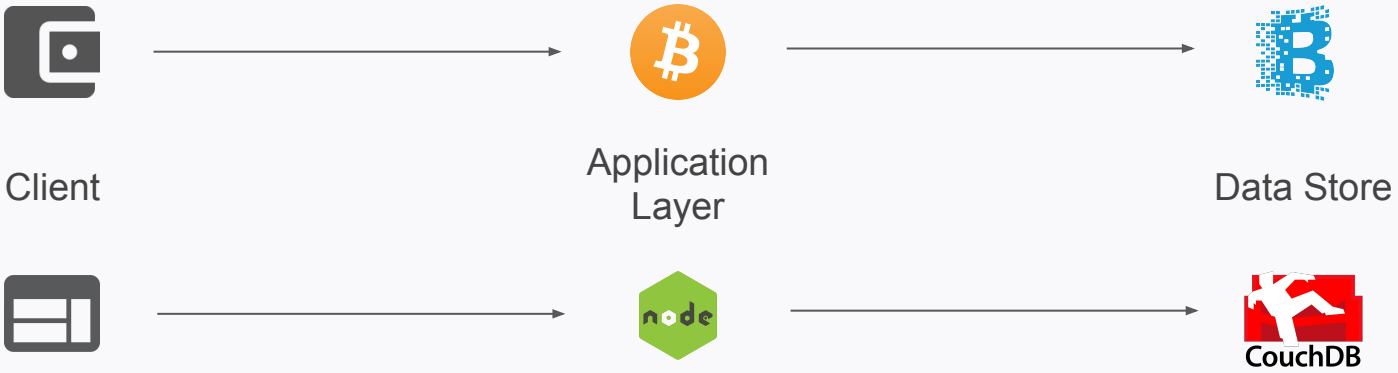
Understanding Blockchain

Kevin Hoyt, IBM
@krhoyt



Bitcoin was the first Blockchain application.

The first Bitcoin transaction took place in 2009, and was for 100 BTC.



Think of **Blockchain** as a database … with some very special characteristics.

Blockchain was first described in a 2008 whitepaper as a cryptographic protocol.

Date	Details	Debit	Credit	Balance
1 Jan 2017	Opening Balance		460.96	460.96
3 Jan 2017	Office Supplies	125.36		335.60
10 Jan 2017	Utilities	101.45		234.15
12 Jan 2017	Comcast Internet Access	492.16		-258.01
15 Jan 2017	Bonfils Blood Center		23.45	-234.56
18 Jan 2017	Kidney Donation		234.56	0.00

A **ledger** is the principal book or computer file for recording and totaling economic transactions.

Fiat money is currency declared legal tender, but not backed by a physical commodity.

“id”: “abcd-1234”, “first”: “Kevin”, “last”: “Hoyt”, “twitter”: “krhoyt”

442275b5abb1a3f5504516835fdecaf0d3943c71237b6f2eadd7ba9f29536845

“id”: “abcd-1234”, “first”: “Kevin”, “last”: “Hoyt”, “twitter”: “krhoyt”,
“hash”: “442275b5abb1a3f5504516835fdecaf0d3943c71237b6f2eadd7ba9f29536845”

“id”: “abcd-1234”, “first”: “Kevin”, “last”: “Smith”, “twitter”: “krsmith”

“id”: “abcd-1234”, “first”: “Kevin”, “last”: “Smith”, “twitter”: “krsmith”,
“hash”: “442275b5abb1a3f5504516835fdecaf0d3943c71237b6f2eadd7ba9f29536845”

11bcb99bee5436d00a733d7ed1b012a00a210783636d8b8d89aa2ffd38f7d14e

“id”: “abcd-1234”, “first”: “Kevin”, “last”: “Smith”, “twitter”: “krsmith”
“hash”: “11bcb99bee5436d00a733d7ed1b012a00a210783636d8b8d89aa2ffd38f7d14e”

Date	Details	Debit	Credit	Balance
1 Jan 2017	Opening Balance		460.96	460.96
3 Jan 2017	Office Supplies	125.36		335.60
10 Jan 2017	Utilities	101.45		234.15
12 Jan 2017	Comcast Internet Access	492.16		-258.01
15 Jan 2017	Bonfils Blood Center		23.45	-234.56
18 Jan 2017	Kidney Donation		234.56	0.00

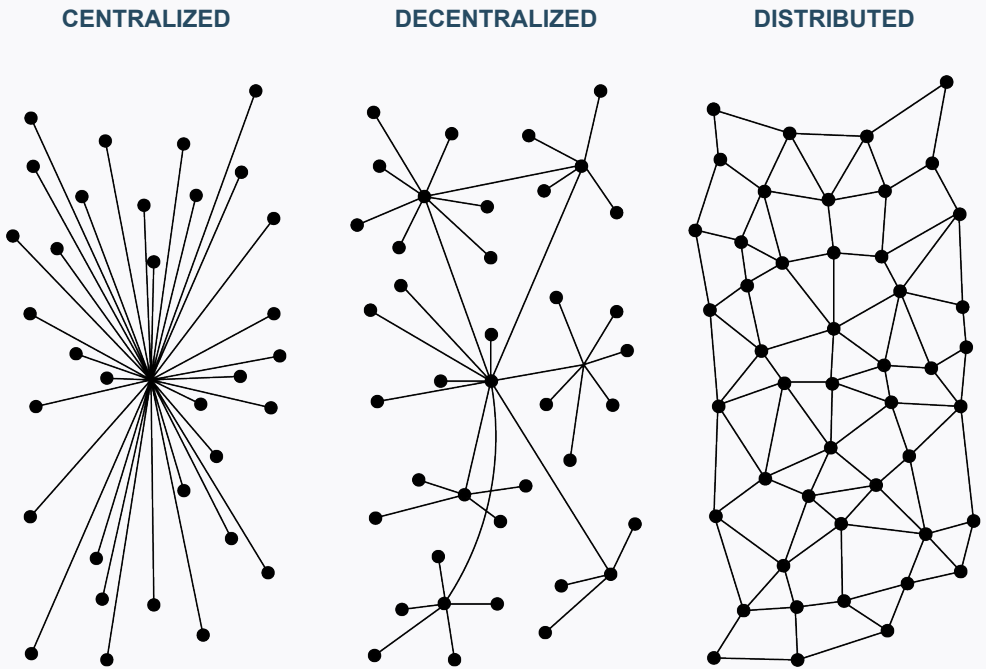
A **block** represents a group of transactions. Data in a block cannot be altered retroactively.

Cracking blockchain hashes is considered impossible even for a quantum computer.

Date	Details	Debit	Credit	Balance
1 Jan 2017	Opening Balance		460.96	460.96
3 Jan 2017	Office Supplies	125.36		335.60
10 Jan 2017	Utilities	101.45		234.15
12 Jan 2017	Comcast Internet Access	492.16		-258.01
15 Jan 2017	Bonfils Blood Center		23.45	-234.56
18 Jan 2017	Kidney Donation		234.56	0.00

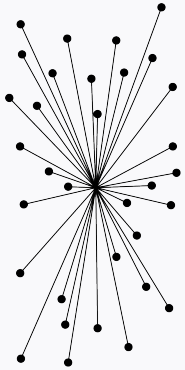
A **chain** refers to the list of ordered records, each with a link to the previous record.

As of January 2017, the Bitcoin blockchain is around 100 gigabytes in size.

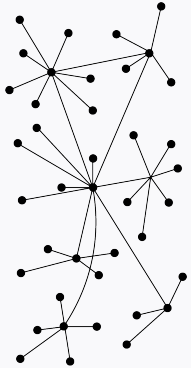


Blockchain is a **distributed** ledger with **decentralized consensus**.

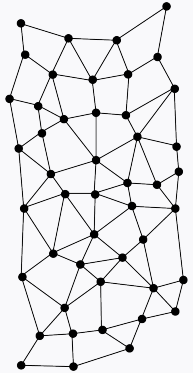
CENTRALIZED



DECENTRALIZED

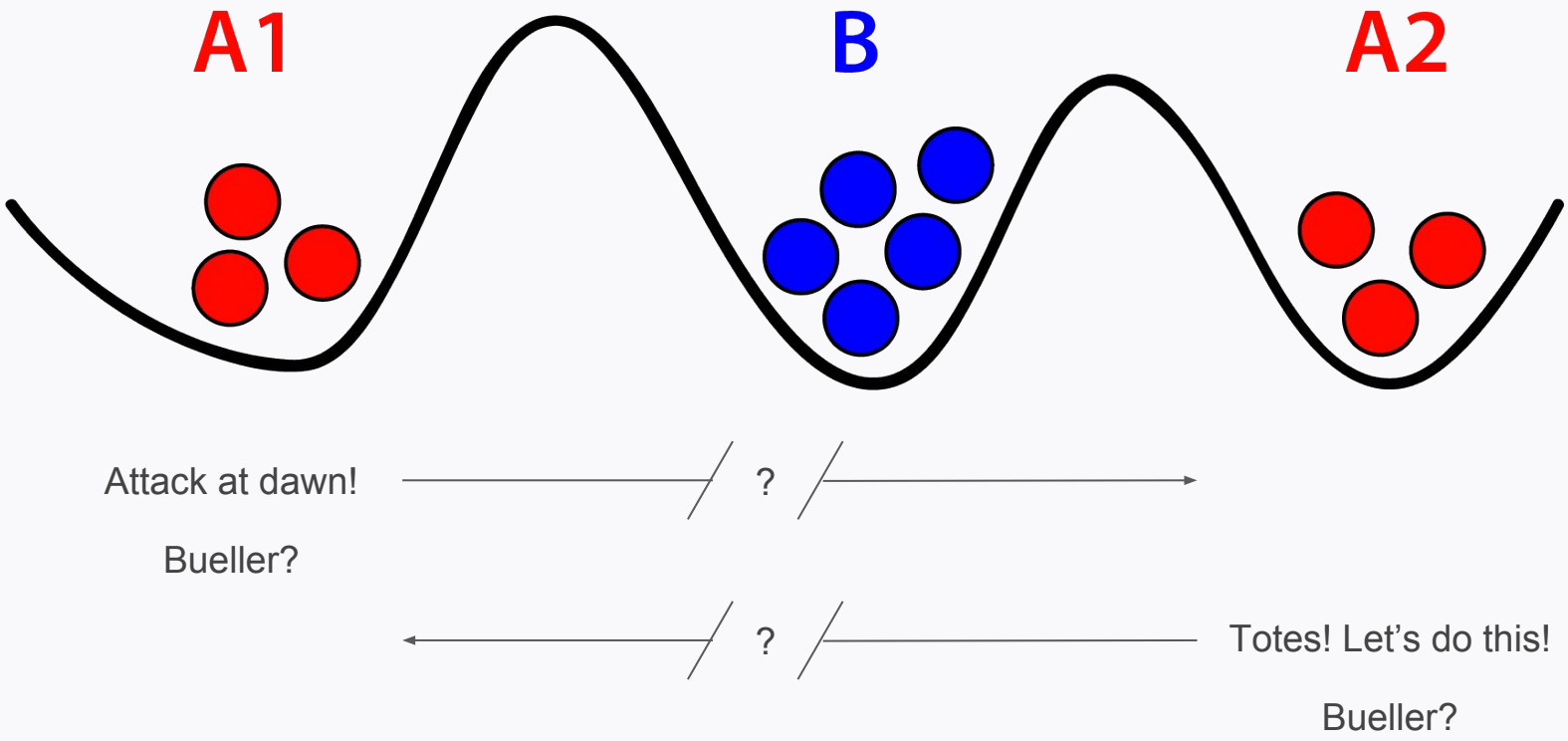


DISTRIBUTED

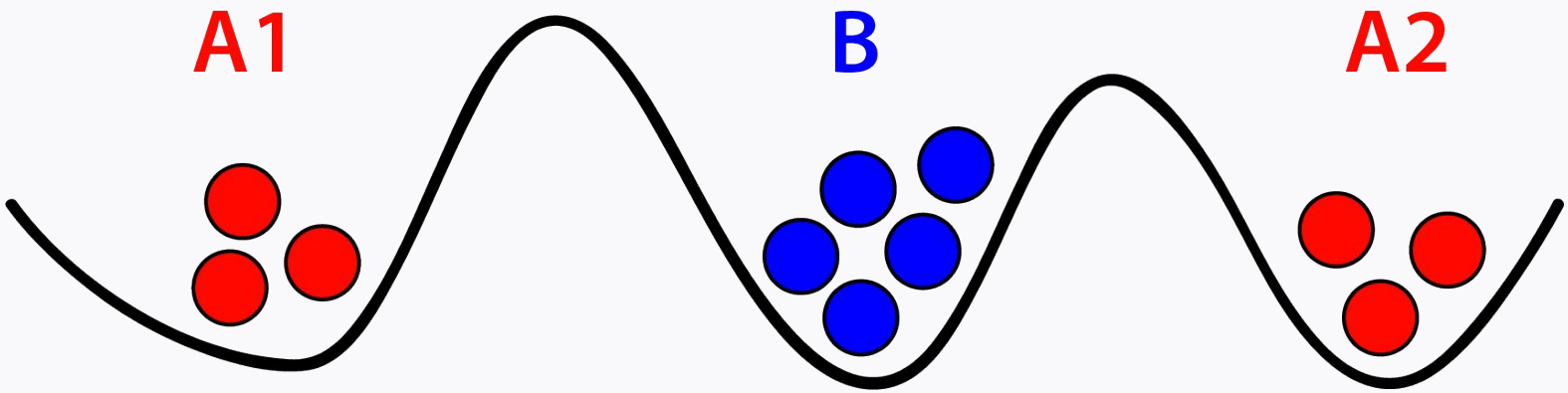


Points of Failure/Maintenance	Single/Easy	Finite	Difficult
Fault Tolerance/Stability	Highly Unstable	Split Into Many	Very Stable
Scalability/Max. Population	Low Scalability	Moderate	Infinite
Ease of Development	Fast	Up Front Cost	Up Front Cost
Evolution/Diversity	Evolve Slowly	Tremendous	Tremendous

(https://medium.com/@bbc4468/centralized-vs-decentralized-vs-distributed-41d92d463868)



The Byzantine Generals' Problem



Consensus guarantees the integrity and consistency of all blockchain transactions.

Hyperledger Fabric uses Practical Byzantine Fault Tolerance (PBFT).

Find a nonce **x** such that:

$$\text{SHA-256}(\text{SHA-256}(\mathbf{r} \,||\, \mathbf{x})) < T / d$$

r = header
(includes header of previous block,
the root of the Merkle tree of transactions)

Proof-of-work deters service abuse by requiring work from the service requestor.

In a Merkle tree, every non-leaf node is labelled with the hash of its child nodes.

Shared Ledger
single source of truth

Secure
(cryptography) tamper proof

Permissioned
participants identity

Private
un-linkable identity

Auditable
prove identity and ownership

Consensus
modular protocol

Smart Contracts
business logic

Digital Assets
record repository

Confidential
permission control

Scalable
100+ year architecture

Blockchain for Business

Think of **Blockchain** as a database …
with some very special characteristics.

- A ledger for recording transactions
- Encrypted data is stored in blocks
- Blocks are chained together
- Cannot be altered retroactively
- Distributed network environment (P2P)
- Decentralized consensus (PBFT)
- Permissioned or permissionless

You might have a use for **Blockchain** if ...

- Your project benefits from a shared database
- May have multiple writers (potentially third-party)
- Peers are not trusted with data they own
- There are restrictions on what data is stored

* Think beyond physical assets (wiki, calendar)

Anders Brownworth

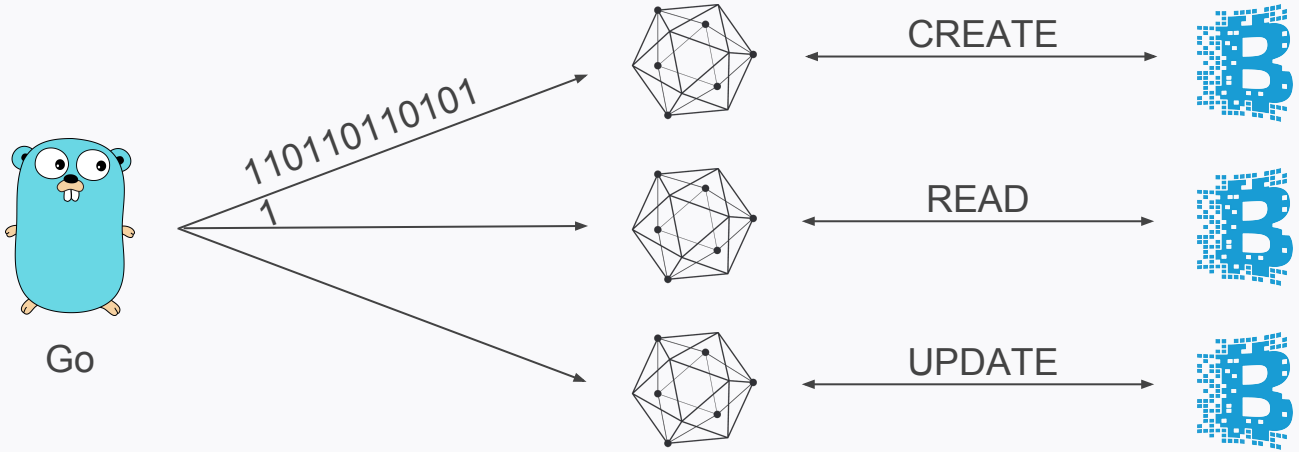
Understanding Blockchain

Kevin Hoyt, IBM
@krhoyt

HYPERLEDGER PROJECT

BY

THE LINUX FOUNDATION



Smart contracts are computer programs that facilitate, verify or enforce a contract.

Smart contracts are also commonly referred to as chaincode, and are written in Go.

```
func ( t *SimpleChaincode ) Init(
  stub shim.ChaincodeStubInterface,
  function string,
  args []string ) ( []byte, error ) {

  var accounts []Account
  bytes, _ := json.Marshal( accounts )

  if err := stub.PutState( "accounts", bytes ); err != nil {
    return nil, errors.New( "Error initializing accounts." )
  }

  return nil, nil
}
```

```
func ( t *SimpleChaincode ) Invoke(
  stub shim.ChaincodeStubInterface,
  function string,
  args []string ) ( []byte, error ) {

  if function == "account_create" {
    return t.account_create( stub, args )
  } else if function == "account_update" {
    return t.account_update( stub, args )
  } else if function == "account_delete" {
    return t.account_delete( stub, args )
  }

  return nil, errors.New( "Function does not exist." )
}
```

```
func ( t *SimpleChaincode ) account_create(
  stub shim.ChaincodeStubInterface,
  args []string ) ( []byte, error ) {

  // Get bytes from block
  bytes, err := stub.GetState( "accounts" )
  if err != nil {
    return nil, errors.New( "Unable to get accounts." )
  }


  // Unmarshal to data structure
  var accounts []Account
  err = json.Unmarshal( bytes, &accounts )

  // Perform work

  return nil, nil
}
```

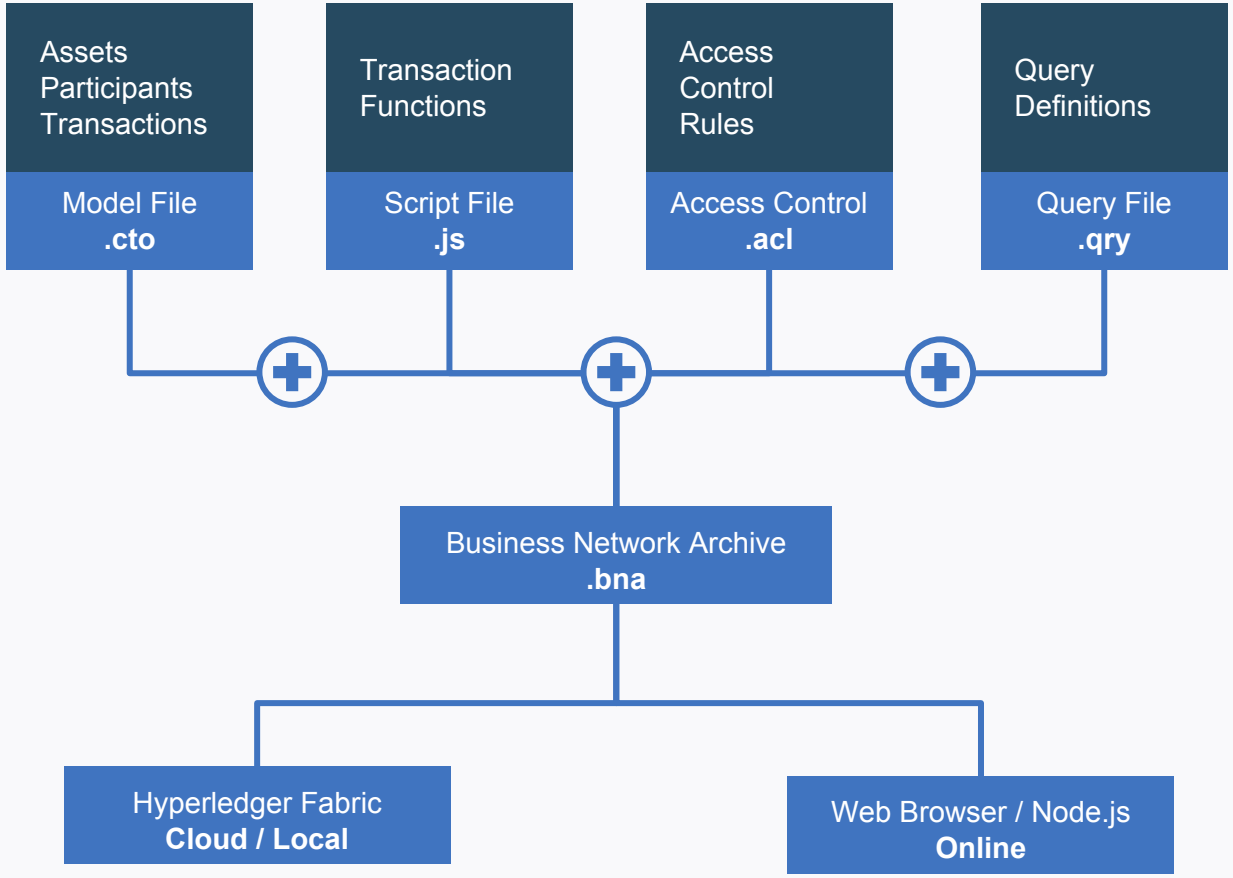
```
// Build JSON values
id := "\"id\": \"" + args[0] + "\", "
name := "\"name\": \"" + args[1] + "\""

// Make into a complete JSON string
// Decode into a single account value
var account Account
content := "{" + id + name + "}"
err = json.Unmarshal( []byte( content ), &account )

accounts = append( accounts, account )

// Encode as JSON
// Put back on the block
bytes, err = json.Marshal( accounts )
err = stub.PutState( "accounts", bytes )
```

```
func ( t *SimpleChaincode ) Query(
  stub shim.ChaincodeStubInterface,
  function string,
  args []string ) ( []byte, error ) {

  if function == "account_by_id" {
    return t.account_by_id( stub, args )
  }

  return nil, errors.New( "Unknown function invocation." )
}
```

```
func ( t *SimpleChaincode ) account_read_all(
  stub shim.ChaincodeStubInterface,
  args []string ) ( []byte, error ) {

  bytes, err := stub.GetState( "accounts" )
  if err != nil {
    return nil, errors.New( "Unable to get accounts." )
  }

  // Unmarshal and search as needed
  // var accounts []Account
  // err = json.Unmarshal( bytes, &accounts )

  return bytes, nil
}
```

Hyperledger Composer

Build Blockchain applications your way.
(Blockchain for JavaScript developers.)


Assets
Participants
Transactions
Model File
.cto
Transaction
Functions
Script File
.js
Access
Control
Rules
Access Control
.acl
Query
Definitions
Query File
.qry
Business Network Archive
.bna
Hyperledger Fabric
Cloud / Local
Web Browser / Node.js
Online

Asset
Cake

Participants
Baker, Customer

Transaction
CakeSale

```
enum Style {
  o BIRTHDAY
}

asset Cake identified by id {
 o String id
 o Double value
 --> Baker owner
}

participant Baker identified by id {
 o String id
 o Integer store
}

transaction CakeSale {
 --> Cake inventory
 --> Customer buyer
}
```

Primitives

String: UTF8 encoded

Double: 64-bit double precision value

Integer: 32-bit signed whole number

Long: 64 bit signed whole number

DateTime: ISO-8601 with optional time zone and offset

Boolean: either true or false

Validators

```
String city default = “Omaha”
String vin regex = /^[A-z][A-z][0-9]{7}/
DateTime updatedAt optional
```

Relationships

```
--> Baker owner
```

Composed of:

```
Namespace.Type#Identifier
org.acme.Baker#123456
// Unidirectional
// Must resolve
```

Arrays

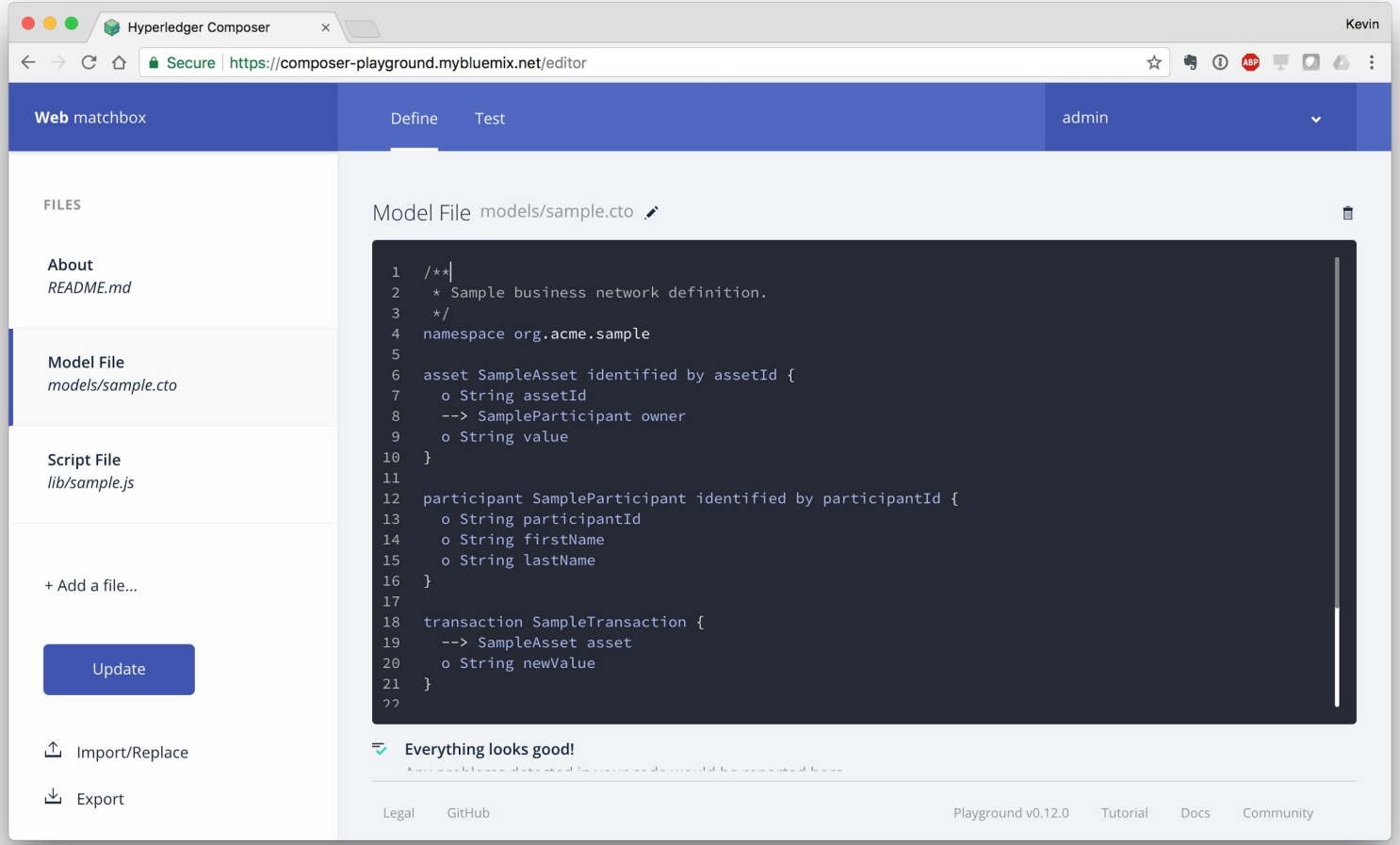
```
Integer[] temperature
--> Weather[] readings
```

Composer expresses the logic for a business network using JavaScript functions. These functions are automatically executed when a transaction is submitted for processing.

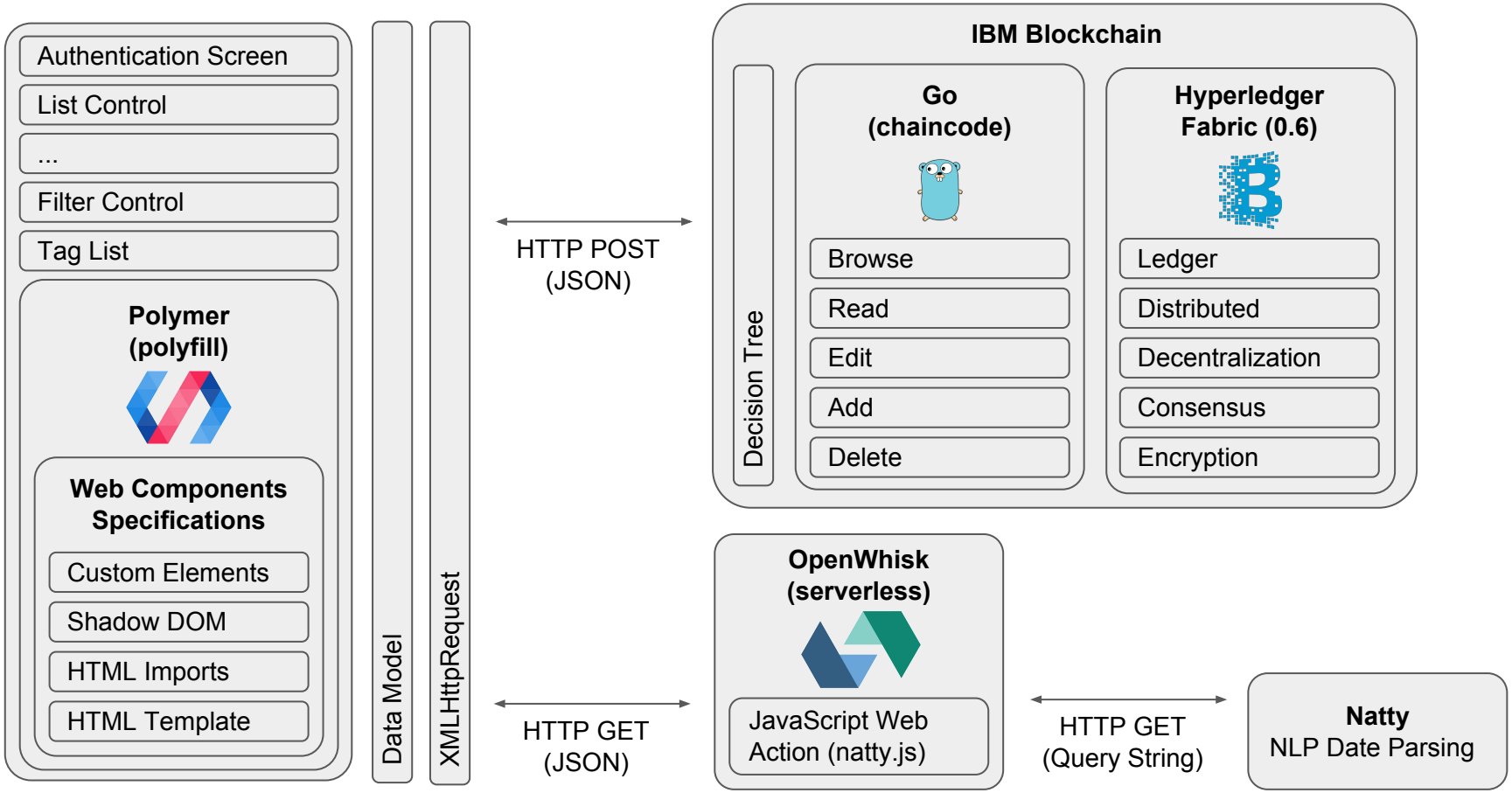
```
/**
 * Make an offer on a vehicle
 * @param {org.acme.Offer} offer - the offer
 * @transaction
 */
function makeOffer( offer ) {
  var listing = offer.listing;

  if( listing.state !== 'FOR_SALE' ) {
    throw new Error( 'Listing is not FOR SALE.' );
  }

  if( listing.offers == null ) {
    listing.offers = [];
  }

  listing.offers.push( offer );

  return getAssetRegistry( 'org.acme.Listing' )
    .then( function( registry ) {
      return registry.update( listing );
    } );
}
```

By defining ACL rules you can determine which users/roles are permitted to create, read, update or delete elements in a business network's domain model.

```
rule VehicleOwner {
  description: "Allow the owner of a vehicle total access"
  participant( m ): "org.acme.vehicle.auction.Member"
  operation: ALL
  resource( v ): "org.acme.vehicle.auction.Vehicle"
  condition: ( v.owner.getIdentifier() == m.getIdentifier() )
  action: ALLOW
}
```


Hyperledger Composer
Kevin
Secure https://composer-playground.mybluemix.net/editor
Web matchbox
Define
Test
admin
FILES
About
README.md
Model File
models/sample.cto
Script File
lib/sample.js
+ Add a file...
Update
Import/Replace
Export
Model File models/sample.cto
/**
* Sample business network definition.
*/
namespace org.acme.sample
asset SampleAsset identified by assetId {
o String assetId
--> SampleParticipant owner
o String value
}
participant SampleParticipant identified by participantId {
o String participantId
o String firstName
o String lastName
}
transaction SampleTransaction {
--> SampleAsset asset
o String newValue
}
Everything looks good!
Legal
GitHub
Playground v0.12.0
Tutorial
Docs
Community

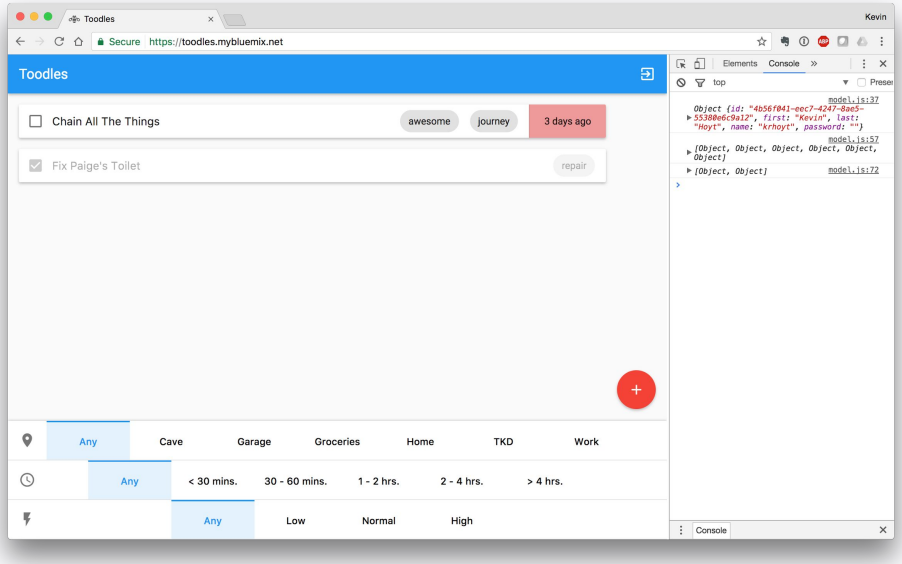
Appendix


Authentication Screen
List Control
...
Filter Control
Tag List
Polymer
(polyfill)
Web Components
Specifications
Custom Elements
Shadow DOM
HTML Imports
HTML Template
Data Model
XMLHttpRequest
HTTP POST
(JSON)
IBM Blockchain
Decision Tree
Go
(chaincode)
Browse
Read
Edit
Add
Delete
Hyperledger
Fabric (0.6)
Ledger
Distributed
Decentralization
Consensus
Encryption
HTTP GET
(JSON)
OpenWhisk
(serverless)
JavaScript Web
Action (natty.js)
HTTP GET
(Query String)
Natty
NLP Date Parsing


Preflight Request
Resource: GET http://app.com/service/users
Access-Control-Request-Method: OPTION
Origin: http://yourdomain.com
Preflight Response
Access-Control-Allow-Origin: *
Access-Control-Request-Method: GET, POST, OPTION
Request
Resource: GET http://app.com/service/users
Access-Control-Request-Method: GET
Origin: http://yourdomain.com
Browser
Server

```
let xhr = new XMLHttpRequest();
xhr.open( 'POST', Blockchain.URL, true );
xhr.setRequestHeader( 'Content-Type', 'application/json' );
xhr.send( JSON.stringify( {
  jsonrpc: '2.0',
  method: 'invoke',
  params: {
    chaincodeID: {
      name: '9e675fb998c004215503fe44'
    },
    ctorMsg: {
      function: 'account_create',
      args: ['abc-123', 'krhoyt']
    },
    secureContext: 'user_type1_0',
    type: 1
  },
  id: 1
} ) );
```

```
let xhr = new XMLHttpRequest();
xhr.open( 'POST', Blockchain.URL, true );
xhr.setRequestHeader( 'Content-Type', 'application/json' );
xhr.send( JSON.stringify( {
  jsonrpc: '2.0',
  method: 'query',
  params: {
    chaincodeID: {
      name: '9e675fb998c004215503fe44'
    },
    ctorMsg: {
      function: 'account_by_id',
      args: ['abc-123']
    },
    secureContext: 'user_type1_0',
    type: 1
  },
  id: 1
} ) );
```



https://github.com/IBM/todo-list-fabric