CST4125: Blockchain Development
Week: 7
Title: Arrays and Promises

Dr Ian Mitchell

smerf.net
Bedfordshire,
UK

2023

Lecture Objectives

Knowledge

- Search
- Lists, Arrays
- UpdateAll
- Advanced JS - more promises
- Pizza Delivery
- Events
- Emit

Disclaimer

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

Contact and Office Hours

Contact Details

- Name: Dr Ian Mitchell
- Room: TG10
- Address: Middlesex University, Computer Science, London, NW4 4BT
- email: smerf.net

Contact and Office Hours

Contact Details

- Name: Dr Ian Mitchell
- Room: TG10
- Address: Middlesex University, Computer Science, London, NW4 4BT
- email: smerf.net

Office Hours

- During term time only
- When: Autumn Term: Mondays 1100-1300hrs
- Please read notifications or emails
- There are occassions that these could be arranged online, e.g., due to industrial action or inclement weather

Deadlines

				Feedback	
Description	Submission	Weight	Deadline	Formative	Summative
1. Hyperledger	MyLearning	50%	18th December 2022	LW11-12	12/01/2023
2. Ethereum	MyLearning	50%	2nd April 2023	LW23-24	24/04/2023
Resits	MyLearning	50-100%	1st July 2023	None	None
Deferals	MyLearning	50-100%	1st July 2023	None	None

Approach

Mistakes

We can learn a lot from bad design, as long as we don't emulate it. Sometimes it is necessary to make mistakes in order to learn. Here we look at implementation of Arrays of items in registries. Warning: Data duplication is a bad thing. It creates inconsistencies. It causes high maintenance code. Data redundancy is a good thing. It ensures consistency and causes low maintenance code. **Data redundancy should be used at all times.**

Bad Examples

- Trader example
- keep tabs on trader commodities
- restricted view
- add trader
- remove trader

Scenario

- Each trader has a list of commodities they currently own
- For academic purposes
- **Consequences from last week**
- removing a member of staff was difficult. Why?

Scenario

- Each trader has a list of commodities they currently own
- For academic purposes
- **Consequences from last week**
- removing a member of staff was difficult. Why?
- Only the owner can sell assets
- if the member of staff removed had assets

Scenario

- Each trader has a list of commodities they currently own
- For academic purposes
- **Consequences from last week**
- removing a member of staff was difficult. Why?
- Only the owner can sell assets
- if the member of staff removed had assets
- these assets remain locked in, no one can sell them

Scenario

- Each trader has a list of commodities they currently own
- For academic purposes
- **Consequences from last week**
- removing a member of staff was difficult. Why?
- Only the owner can sell assets
- if the member of staff removed had assets
- these assets remain locked in, no one can sell them
- each trader keeps a lists of the assets they own
- requires updating each time a commodity changes ownership, for the seller and the buyer.

Scenario

- Each trader has a list of commodities they currently own
- For academic purposes
- **Consequences from last week**
- removing a member of staff was difficult. Why?
- Only the owner can sell assets
- if the member of staff removed had assets
- these assets remain locked in, no one can sell them
- each trader keeps a lists of the assets they own
- requires updating each time a commodity changes ownership, for the seller and the buyer.
- when a member of staff leaves a nominated member of staff is given all the assets, and then the member of staff is deleted.

Trader

CTO - Enums, Assets & Participants

```
4  namespace org.t2.net
5  enum Grade{
6    o Manager
7    o Consultant
8    o Trader
9    o Clerk
10 }
11 asset Commodity identified by tradingSymbol {
12   o String tradingSymbol
13   o String description optional
14   o Double quantity optional
15   --> Trader owner
16 }
17 participant Trader identified by tradeId {
18   o String tradeId
19   o String firstName
20   o String lastName
21   o Grade status
22   o String[] commoditiesOwned
23 }
```

Difference from last week?

Trader
CTO - Enums, Assets & Participants

```
namespace org.t2.net
enum Grade{
  o Manager
  o Consultant
  o Trader
  o Clerk
}
asset Commodity identified by tradingSymbol {
  o String tradingSymbol
  o String description optional
  o Double quantity optional
  --> Trader owner
}
participant Trader identified by tradeId {
  o String tradeId
  o String firstName
  o String lastName
  o Grade status
  o String[] commoditiesOwned
}
```

Difference from last week?

- Array [ ]
- Array is to represent all the commodities owned

Trader
CTO-Transactions

```
transaction Trade {
  --> Commodity commodity
  --> Trader newOwner
}
transaction AddNewStaff{
  o Trader newStaff
}
transaction RemoveStaff{
  --> Trader removedStaff
}
transaction Initialise{}
```

Difference from last week?

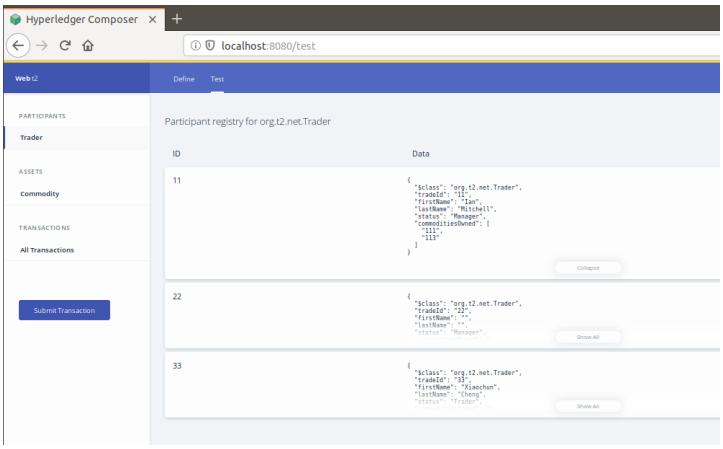
Trader
CTO-Transactions

```
transaction Trade {
  --> Commodity commodity
  --> Trader newOwner
}
transaction AddNewStaff{
  o Trader newStaff
}
transaction RemoveStaff{
  --> Trader removedStaff
}
transaction Initialise{}
```

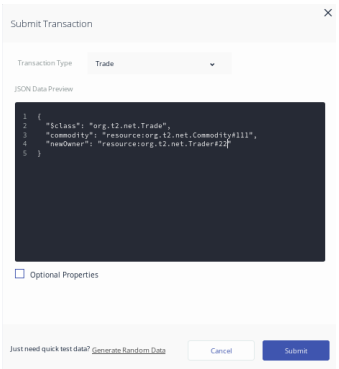
Difference from last week?

- Initialise
- no paramaters

Trade

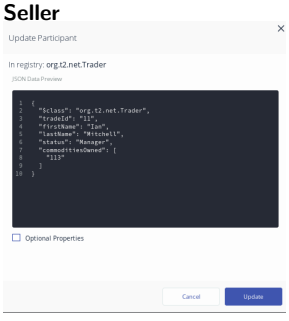


Transaction

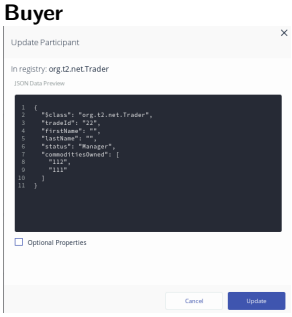


Transaction

Seller



Buyer



Trade Transaction

- Check?

Trade Transaction

- Check?
- Buyer exists?
- Commodity exists?
- Updates?

Trade Transaction

- Check?
- Buyer exists?
- Commodity exists?
- Updates?
- Commodity ownership
- Trader: commoditiesOwned array

Trade Transaction

- Check?
- Buyer exists?
- Commodity exists?
- Updates?
- Commodity ownership
- Trader: commoditiesOwned array
- Buyer: Adding to array
- Seller: Removing from array

Trader Transactions

JS - Initialise

```
40 /**
41 * Initialise system
42 * @param {org.t2.net.Initialise} no param
43 * @transaction
44 */
45 async function Initialise(){
46   let ids=['112','222','333'];
47   let firstNames=['Ian', 'Sukhvinder', 'Xiaochun'];
48   let lastNames=['Mitchell','Hara', 'Cheng'];
49   let grades=['Manager', 'Trader', 'Trader'];
50   let staff = new Array();
51   for(i=0;i<ids.length;i++){
52     let factory = getFactory();
53     let newStaff=factory.newResource(ns,'Trader',ids[i]);
54     newStaff.firstName=firstNames[i];
55     newStaff.lastName=lastNames[i];
56     newStaff.status=grades[i];
57     newStaff.commoditiesOwned=new Array();
58     staff.push(newStaff);
59   }
60   let traderReg=await getParticipantRegistry(ns+'.Trader');
61   await traderReg.addAll(staff);
62 }
```


Trader Transactions

JS - Add New Staff

```
28 /**
29 * transaction to add new staff
30 * @param {org.t2.net.AddNewStaff} tx
31 * @transaction
32 */
33 async function AddNewStaff(tx) {
34     if (getCurrentParticipant().status == 'Manager'){
35         let participantRegistry = await getParticipantRegistry(ns+".Trader");
36         await participantRegistry.add(tx.newStaff);
37     } else
38         throw new Error("Insufficient privileges: manager");
39 }
```

Trader Transaction

JS - Trade

```
var ns="org.t2.net"
/**
 * transaction of a commodity from one trader to another
 * @param {org.t2.net.Trade} tx - the parameter
 * @transaction
 */
async function tradeCommodity(tx) {
    let buyerSeller = new Array();
    let traderReg = await getParticipantRegistry(ns+'.Trader');
    let exist = await traderReg.exists(tx.newOwner.getIdentifier());
    if(exist){
        let buyer = await traderReg.get(tx.newOwner.getIdentifier());
        buyer.commoditiesOwned.push(tx.commodity.getIdentifier().toString());
        buyerSeller.push(buyer)
        let seller = await traderReg.get(tx.commodity.owner.getIdentifier());
        let needle = tx.commodity.getIdentifier().toString();
        let haystack = seller.commoditiesOwned;
        let filteredHaystack = haystack.filter((item)=>item!=needle);
        seller.commoditiesOwned = filteredHaystack;
        buyerSeller.push(seller);
        await traderReg.updateAll(buyerSeller);
        let commodityReg = await getAssetRegistry(ns+'.Commodity');
        tx.commodity.owner = tx.newOwner;
        await commodityReg.update(tx.commodity);
    } else
        throw new Error('Trader/Buyer does not exist');
}
```


Commodities

JSON

ID	Data
111	{ "$class": "org.t2.net.Commodity", "tradingSymbol": "111", "owner": "resource:org.t2.net.Trader#22" }
112	{ "$class": "org.t2.net.Commodity", "tradingSymbol": "112", "owner": "resource:org.t2.net.Trader#22" }
113	{ "$class": "org.t2.net.Commodity", "tradingSymbol": "113", "owner": "resource:org.t2.net.Trader#11" }

Haystack

Javascript

Array Filter

```
var a = [12,13,14,15,33,34,35,37,37,37];

function remove(x){
  return x >= 20;
}

var a1 = a.filter(remove);
console.log('a1');
console.log(a1);

console.log('a2');
var a2 = a.filter( (x)=> x>=20);
console.log(a2);
var a3 = a.filter((x)=> x!=37);
console.log('a3');
console.log(a3);
```


Array Filter

Output

```
a1
[ 33, 34, 35, 37, 37, 37 ]
a2
[ 33, 34, 35, 37, 37, 37 ]
a3
[ 12, 13, 14, 15, 33, 34, 35 ]
```


Alternative

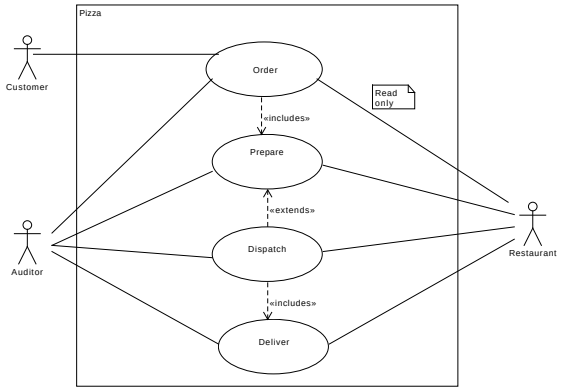
- Pizza Delivery
- with a promise chain
- using await command
- The burden of access is shifted. Where?

- Look at examples on github `https://github.com/hyperledger/composer-sample-networks`
- These cannot be used for the coursework.
- Pizza

Alternative

- Pizza Delivery
- with a promise chain
- using await command
- The burden of access is shifted. Where?
- The burden is shifted from JS to ACL
- Look at examples on github
  `https://github.com/hyperledger/composer-sample-networks`
- These cannot be used for the coursework.
- Pizza

Pizza
Use Case Diagram



Pizza
CTO - Status

```
 8 /* ENUMERATOR */
 9 enum STATUS {
10   o PLACED
11   o PREPARED
12   o DISPATCHED
13   o DELIVERED
14 }
```


Pizza
CTO - Status

```
 8 /* ENUMERATOR */
 9 enum STATUS {
10   o PLACED
11   o PREPARED
12   o DISPATCHED
13   o DELIVERED
14 }
```

- lifecycle of order
- PLACED - create order by customer
- PREPARED - update by pizzOutlet
- DISPATCHED - update by pizzaOutlet
- DELIVERED - update by pizzaOutlet

Pizza
CTO - Size & PizzaType

```
27 enum SIZE {
28   o small
29   o medium
30   o large
31 }
32 enum PIZZATYPE{
33   o americana
34   o carbonara
35   o margherita
36   o marinara
37   o napoli
38   o quattro
39   o romana
40 }
```

- Toppings
- Size
- Pizza Type
- Enumerator Types

Pizza
CTO - Address - Customer - Restaurant

```
42 /* CONCEPT */
43 concept ADDRESS{
44   o String Name optional
45   o String NameNumber default="1"
46   o String Street default="High St"
47   o String PostCode default="NW44BT"
48 }
49
50 /* PARTITICPANT */
51 participant customer identified by customerID{
52   o String customerID
53   o ADDRESS deliveryAddress
54 }
55
56 participant pizzaOutlet identified by poID{
57   o String poID
58   o ADDRESS poAddress
59 }
60
61 participant pqc identified by pqcID{
62   o String pqcID
63 }
```

Pizza
CTO - Order

```
70 }
71 /* current version only allows
       1 pizza per order
72  * simply rectified by adding
       array
73 */
74 asset order identified by
       orderID{
75   o String orderID
76   --> pizzaDetail[] pizzas
77   --> pizzaOutlet restaurant
78   --> customer consumer
79   o STATUS status
80 }
```

- Where does ID come from?

Pizza
CTO - Order

```
70 }
71 /* current version only allows
       1 pizza per order
72  * simply rectified by adding
       array
73 */
74 asset order identified by
       orderID{
75   o String orderID
76   --> pizzaDetail[] pizzas
77   --> pizzaOutlet restaurant
78   --> customer consumer
79   o STATUS status
80 }
```

- Where does ID come from?
- User generated, can be pseudo-random
- Comment on mulitple orders
- array of pizzaDetails
- TOPPING is inaccessible
- Usually an order has 3 things:

Pizza
CTO - Order

```
70 }
71 /* current version only allows
       1 pizza per order
72  * simply rectified by adding
       array
73 */
74 asset order identified by
       orderID{
75   o String orderID
76   --> pizzaDetail[] pizzas
77   --> pizzaOutlet restaurant
78   --> customer consumer
79   o STATUS status
80 }
```

- Where does ID come from?
- User generated, can be pseudo-random
- Comment on mulitple orders
- array of pizzaDetails
- TOPPING is inaccessible
- Usually an order has 3 things:
  1. Product: Pizza, sometimes the quantity
  2. Seller: Restaurant
  3. Buyer: Customer
- STATUS: track progress

Pizza
CTO - Transactions

```
89 transaction prepareOrder{
90   --> order pizzaPrepared
91 }
92
93 transaction dispatchOrder{
94   --> order pizzaDispatched
95 }
96
97 transaction deliverOrder{
98   --> order pizzaDelivered
99 }
```


CustomerSeeSelf: Customers can only see themselves

Rules
ACL - Customer

```
8  rule customerSeeSelf{
9    description: "customer see themselves"
10   participant(p): "org.pqc.uk.customer"
11   operation: ALL
12   resource(r): "org.pqc.uk.customer"
13   condition: (p.getIdentifier()==r.
         getIdentifier())
14   action: ALLOW
15 }
16 rule customerSeePizza{
17   description: "customer see pizza"
18   participant: "org.pqc.uk.customer"
19   operation:READ
20   resource: "org.pqc.uk.pizzaDetail"
21   action: ALLOW
22 }
23 rule customerSeeOrder{
24   description: "customer see pizza"
25   participant(p): "org.pqc.uk.customer"
26   operation:ALL
27   resource(r): "org.pqc.uk.order"
28   //transaction(t): "org.pqc.uk.placeOrder"
29   condition: (p.getIdentifier()==r.consumer.
         getIdentifier())
30   action: ALLOW
31 }
```

CustomerSeeSelf:
: Customers can only see themselves. Condition that ensures the consumer in the order is equal to the customer.

CustomerSeePizza:
: Customers can see the pizzas available

Rules
ACL - Customer

```
rule customerPlaceOrder{
  description: "customer places order"
  participant: "org.pqc.uk.customer"
  operation: ALL
  resource: "org.pqc.uk.placeOrder"
  action: ALLOW
}
rule customerReadRestaurant{
  description: "customer has read access to
      restaurants"
  participant: "org.pqc.uk.customer"
  operation: READ
  resource: "org.pqc.uk.pizzaOutlet"
  action: ALLOW
}
```

customerPlaceOrder:
: Only a customer can place an order and access transaction `placeOrder`

customerReadRestaurant:
: Customers are permitted to read pizzaOutlet details

Rules
ACL - Restaurant

```
rule restaurantSeeSelf{
  description: "restaurants can only view
      their own details"
  participant(p): "org.pqc.uk.pizzaOutlet"
  operation: ALL
  resource(r): "org.pqc.uk.pizzaOutlet"
  condition: (p.getIdentifier()==r.
      getIdentifier())
  action: ALLOW
}
rule restaurantSeeOrders{
  description: "restaurant can only see
      their own orders"
  participant(p): "org.pqc.uk.pizzaOutlet"
  operation: ALL
  resource(r): "org.pqc.uk.order"
  condition: (p.getIdentifier()==r.
      restaurant.getIdentifier())
  action:ALLOW
}
```

restaurantSeeSelf:
: Restaurant can only see themselves

restaurantSeeOrders:
: Restaurant can only see orders placed at their pizzaOutlet

Rules
ACL - Restaurant

```
rule restaurantReadsCustomer{
  description: "restaurant reads
      customer"
  participant: "org.pqc.uk.
      pizzaOutlet"
  operation: READ
  resource: "org.pqc.uk.customer"
  action:ALLOW
}
rule restaurantPlaceOrder{
  description: "restaurant reads
      order"
  participant: "org.pqc.uk.
      pizzaOutlet"
  operation: READ, UPDATE//CANNOT
      CREATE
  resource: "org.pqc.uk.order"
  transaction: "org.pqc.uk.
      prepareOrder"
  action: ALLOW
}
rule restaurantProcessOrder{
  description: "restaurant process
      order"
  participant: "org.pqc.uk.
      pizzaOutlet"
  operation: ALL
  resource: "org.pqc.uk.
      prepareOrder"
  action: ALLOW
```

restaurantReadsCustomer:
: restaurant can read customer details

restaurantPlaceOrder:
: Restaurants cannot place orders, merely read and update the status of them

restaurantProcessOrder:
: Restaurants can process orders from status `PLACED` to `PREPARED` using transaction `prepareOrder`

Rules
ACL - Restaurant

```
rule restaurantDispatchOrder{
  description: "restaurant dispatch
      order access"
  participant: "org.pqc.uk.
      pizzaOutlet"
  operation:ALL
  resource:"org.pqc.uk.
      dispatchOrder"
  action:ALLOW
}
rule restuarantDeliverOrder{
  description: "restaurant deliver
      order access"
  participant: "org.pqc.uk.
      pizzaOutlet"
  operation:ALL
  resource: "org.pqc.uk.
      deliverOrder"
  action:ALLOW
}
```

restaurantDispatchOrder:
: Restaurant can process orders from status `PREPARED` to `DISPATCHED` using transaction `restaurantDispatchOrder`

restaurantDeliverOrder:
: Restaurant can process orders from status `DISPATCHED` to `DELIVERED` using the transaction `restaurantDeliverOrder`

Transactions
JS - Place Order

```
/*
 * User submits order to restaurant
 * @param {org.pqc.uk.placeOrder} placeOrder - pizza order
 * @transaction
 */
async function placeOrder(tx){
    const ns='org.pqc.uk';
//create new order
    var factory = getFactory();
    var newOrder=factory.newResource(ns,'order',tx.orderID);
    newOrder.pizza = tx.pizza;
    newOrder.restaurant = tx.restaurant;
    newOrder.consumer = tx.Customer;
    newOrder.status = 'PLACED';
// add new order to the order registry
    const orderReg = await getAssetRegistry(ns+'.order');
    await orderReg.add(newOrder);
}
```


Transactions
JS - Prepare Order

```
/*
 * restaurant prepares order
 * @param {org.pqc.uk.prepareOrder} prepareOrder - pizza order
 * @transaction
 */
async function prepareOrder(tx){
    const ns='org.pqc.uk';
    currentOrder = tx.pizzaPrepared;
    if( currentOrder.status !== 'PLACED')
    {
        throw new Error('Current order'+currentOrder.orderID+' is in wrong status to be
      prepared');
    }
    else
    {
        currentOrder.status = 'PREPARED';
    }
// update order with currentOrder
    const orderReg = await getAssetRegistry(ns+'.order');
    await orderReg.update(currentOrder);
// emit the event
    const factory=getFactory();
    const prepareOrderEvent=factory.newEvent(ns,'prepareOrderEvent');
    prepareOrderEvent.pizzaPrepared=currentOrder;
    emit(prepareOrderEvent);
}
```

Transactions
JS - Dispatch Order

```
/*
 * restaurant dispatches order
 * @param{org.pqc.uk.dispatchOrder} dispatchOrder - pizza dispatched
 * @transaction
 */
async function dispatchOrder(tx){
    const ns='org.pqc.uk';
    Prepare currentOrder=tx.pizzaDispatched;
    if( currentOrder.status !== 'PREPARED')
    {
        throw new Error('Current order has not been prepared');
    }
    else
    {
        currentOrder.status = 'DISPATCHED';
    }
// update order with currentOrder
    const orderReg = await getAssetRegistry(ns+'.order');
    await orderReg.update(currentOrder);
// emit the event
    const factory=getFactory();
    const dispatchOrderEvent=factory.newEvent(ns,'dispatchOrderEvent');
    dispatchOrderEvent.pizzaDispatched = currentOrder;
    emit(dispatchOrderEvent);
}
```


Transactions
JS - Deliver Order

```
/*
 * customer receives order
 * @param{org.pqc.uk.deliverOrder} deliverOrder - pizza delivered
 * @transaction
 */
async function deliverOrder(tx){
    const ns='org.pqc.uk';
    currentOrder=tx.pizzaDelivered;
    if( currentOrder.status !== 'DISPATCHED')
    {
        throw new Error('Current order has not been dispatched');
    }
    else
    {
        currentOrder.status = 'DELIVERED';
    }
// update order with currentOrder
    const orderReg = await getAssetRegistry(ns+'.order');
    await orderReg.update(currentOrder);
// emit the event
    const factory=getFactory();
    const deliverOrderEvent=factory.newEvent(ns,'deliverOrderEvent');
    deliverOrderEvent.pizzaDelivered=currentOrder;
    emit(deliverOrderEvent);
}
```


References I

[1] The Linux Foundation. *Hyperledger Architecture, Volume 1*. hyperlink. [Accessed: Jan 2021]. 2017.

[2] The Linux Foundation. *Hyperledger Architecture, Volume 2*. hyperlink. [Accessed: Jan 2021]. 2018.

[3] Nitin Gaur et al. *Hands-on Blockchain with Hyperledger: Building Decentralised Applications with Hyperledger Fabric and Composer*. Packt, 2018. ISBN: 9781788994521.

Web Resources

- http://hyperledger.org
- https://nodejs.org
- https://hyperledger.github.io/composer/latest/api/runtime-factory
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array
- https://github.com/hyperledger/composer-sample-networks
- https://hyperledger.github.io/composer/latest/business-network/bnd-create