

Blockchain Development

Week: 7

Title: Node.js and TX: Lists

Dr Ian Mitchell



Middlesex University,
Dept. of Computer Science,
London

September 26, 2019

Navigation icons: back, forward, search, etc.

©i.mitchell@mdx.ac.uk

CST4025:L7

September 26, 2019

1 / 37

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.

Navigation icons: back, forward, search, etc.

©i.mitchell@mdx.ac.uk

CST4025:L7

September 26, 2019

2 / 37

Approach



Mistakes

We can learn a lot from bad design. Sometimes it is necessary to make mistakes in order to learn. Here we look at implementation of Arrays of items in registries, however, care is to be taken in and warning signs should be given and the reduction in data redundancy is a good thing. Styles of programming will also be looked at, many coders avoid the use of promises and we will look at this approach.

Bad Examples

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

Navigation icons: back, forward, search, etc.

©i.mitchell@mdx.ac.uk

CST4025:L7

September 26, 2019

3 / 37

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?

Navigation icons: back, forward, search, etc.

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

Navigation icons: back, forward, search, etc.

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

Navigation icons: back, forward, search, etc.

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.

Navigation icons

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.

Navigation icons

Trader

CTO



```
1 /**
2  * Sample business network definition.
3  */
4 namespace org.t4.net
5
6 enum Grade {
7     o manager
8     o consultant
9     o intern
10    o clerk
11 }
12
13 asset Commodity identified by tradingSymbol {
14     o String tradingSymbol
15     o String description
16     o Double quantity
17     --> Trader owner
18 }
19
20 participant Trader identified by tradeId {
21     o String tradeId
22     o String firstName
23     o String lastName
24     o Grade Status
25     o String[] commoditiesOwned
26 }
27
28 transaction Trade {
29     --> Commodity commodity
30     --> Trader newOwner
```

Difference from last week?

Navigation icons



```

1 /**
2  * Sample business network definition.
3  */
4 namespace org.t4.net
5
6 enum Grade {
7     o manager
8     o consultant
9     o intern
10    o clerk
11 }
12
13 asset Commodity identified by tradingSymbol {
14     o String tradingSymbol
15     o String description
16     o Double quantity
17     --> Trader owner
18 }
19
20 participant Trader identified by tradeId {
21     o String tradeId
22     o String firstName
23     o String lastName
24     o Grade Status
25     o String[] commoditiesOwned
26 }
27
28 transaction Trade {
29     --> Commodity commodity
30     --> Trader newOwner

```

Difference from last week?

- line 25 - Array
- Array is to represent all the commodities owned



```

1 {
2   "$class": "org.t4.net.Trader",
3   "tradeId": "0227",
4   "firstName": "",
5   "lastName": "",
6   "Status": "manager",
7   "commoditiesOwned": []
8 }

```

```

1 {
2   "$class": "org.t4.net.Trader",
3   "tradeId": "1711",
4   "firstName": "",
5   "lastName": "",
6   "Status": "manager",
7   "commoditiesOwned": [
8     "8084",
9     "7856",
10    "8941",
11    "2139",
12    "2336"
13 ]
14 }

```



```

1 {
2   "$class": "org.t4.net.Commodity",
3   "tradingSymbol": "2139",
4   "description": "",
5   "quantity": 0,
6   "owner": "resource:org.t4.net.Trader#1711"
7 }
8
9 {
10  "$class": "org.t4.net.Commodity",
11  "tradingSymbol": "2336",
12  "description": "",
13  "quantity": 0,
14  "owner": "resource:org.t4.net.Trader#1711"
15 }
16
17 {
18  "$class": "org.t4.net.Commodity",
19  "tradingSymbol": "7856",
20  "description": "",
21  "quantity": 0,
22  "owner": "resource:org.t4.net.Trader#1711"
23 }

```

```

24 {
25  "$class": "org.t4.net.Commodity",
26  "tradingSymbol": "8084",
27  "description": "",
28  "quantity": 0,
29  "owner": "resource:org.t4.net.Trader#1711"
30 }
31
32 {
33  "$class": "org.t4.net.Commodity",
34  "tradingSymbol": "8941",
35  "description": "",
36  "quantity": 0,
37  "owner": "resource:org.t4.net.Trader#1711"
38 }
39

```

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 Transaction

JS - Does Buyer Exist?



```
1 /**
2  * transaction of a commodity from one trader to another
3  * @param {org.t4.net.Trade} trade - the trade to be processed
4  * @transaction
5  */
6 async function tradeCommodity(tx) {
7   var ns="org.t4.net";
8   var me=getCurrentParticipant();
9   var updateArray = new Array();
10
11   return getParticipantRegistry(ns+".Trader")
12     .then(function (traderRegistry){
13       return traderRegistry.exists(tx.newOwner.getIdentifer())
14     }.then(function(exists){
```

Trader Transaction

JS - Add Commodity to Buyer Array



```
15   if (exists){
16     return traderRegistry.get(tx.newOwner.getIdentifer())
17       .then(function(singleTrader){
18       // add the commodity id from the tx to the new owner in singleTrader
19       singleTrader.commoditiesOwned.push(tx.commodity.getIdentifer().
20       toString());
21       console.log('Update newOwner after sale');
22       updateArray.push(singleTrader);
```

Trader Transaction

JS - Remove Commodity to Seller Array



```
22 // remove the commodity id from the tx from the old owner in traderRegistry
23     let needle=tx.commodity.tradingSymbol.toString();
24     let haystack=me.commoditiesOwned;
25     let filteredHaystack = haystack.filter((item)=>item!==needle);
26     me.commoditiesOwned = filteredHaystack;
27     updateArray.push(me);
28 // update the trader registry using the updated Array of traders
29     traderRegistry.updateAll(updateArray);
```

Navigation icons

©i.mitchell@mdx.ac.uk

CST4025:L7

September 26, 2019

11 / 37

Trader Transaction

JS - Commodity to Ownership updated



```
30         return getAssetRegistry(ns+'.Commodity')
31         .then( function(commodityReg){
32 // update the owner in the commodity
33             tx.commodity.owner=tx.newOwner;
34             return commodityReg.update(tx.commodity);
35         })
36     })
37 }
```

Navigation icons

©i.mitchell@mdx.ac.uk

CST4025:L7

September 26, 2019

12 / 37

Trader Transaction

JS - Buyer does not exist



```
38     else
39     {
40         throw new Error('New owner does not exist');
41     }
42 })
43 })
44 }
```

Navigation icons

©i.mitchell@mdx.ac.uk

CST4025:L7

September 26, 2019

13 / 37

Completing the Transaction



Submit Transaction



Transaction Type

Trade



JSON Data Preview

```
1 {  
2   "$class": "org.t4.net.Trade",  
3   "commodity": "resource:org.t4.net.Commodity#2139",  
4   "newOwner": "resource:org.t4.net.Trader#0227"  
5 }
```



Transaction History



Date, Time	Entry Type	Participant	
2019-07-31, 14:23:59	Trade	1711 (Trader)	view record
2019-07-31, 14:21:22	ActivateCurrentIdentity	none	view record
2019-07-31, 13:30:35	Issuelidentity	admin (NetworkAdmin)	view record



Transaction History block



```
1 {  
2   "$class": "org.t4.net.Trade",  
3   "commodity": "resource:org.t4.net.Commodity#2139",  
4   "newOwner": "resource:org.t4.net.Trader#0227",  
5   "transactionId": "474a3060-b46e-469f-a1cb-51838fe6c0bf",  
6   "timestamp": "2019-07-31T13:23:59.548Z"  
7 }
```



Traders

JSON



```
1 {
2   "$class": "org.t4.net.Trader",
3   "tradeId": "0227",
4   "firstName": "",
5   "lastName": "",
6   "Status": "manager",
7   "commoditiesOwned": [
8     "2139"
9   ]
10 }

1 {
2   "$class": "org.t4.net.Trader",
3   "tradeId": "1711",
4   "firstName": "",
5   "lastName": "",
6   "Status": "manager",
7   "commoditiesOwned": [
8     "8084",
9     "7856",
10    "8941",
11    "2336"
12  ]
13 }
```

Navigation icons

Commodities

JSON



```
1 {
2   "$class": "org.t4.net.Commodity",
3   "tradingSymbol": "2139",
4   "description": "",
5   "quantity": 0,
6   "owner": "resource:org.t4.net.Trader#0227"
7 }
8
9 {
10  "$class": "org.t4.net.Commodity",
11  "tradingSymbol": "2336",
12  "description": "",
13  "quantity": 0,
14  "owner": "resource:org.t4.net.Trader#1711"
15 }
16
17 {
18  "$class": "org.t4.net.Commodity",
19  "tradingSymbol": "7856",
20  "description": "",
21  "quantity": 0,
22  "owner": "resource:org.t4.net.Trader#1711"
23 }

24 {
25   "$class": "org.t4.net.Commodity",
26   "tradingSymbol": "8084",
27   "description": "",
28   "quantity": 0,
29   "owner": "resource:org.t4.net.Trader#1711"
30 }
31
32 {
33   "$class": "org.t4.net.Commodity",
34   "tradingSymbol": "8941",
35   "description": "",
36   "quantity": 0,
37   "owner": "resource:org.t4.net.Trader#1711"
38 }
39 }
```

Navigation icons

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

Navigation icons

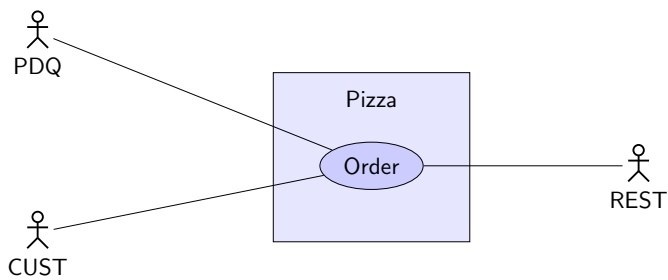
Alternative



- 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

Simplified Use Case



Pizza

CTO - Status



```
8
9 /* ENUMERATOR */
10 enum STATUS {
11     o PLACED
12     o PREPARED
13     o DISPATCHED
14     o DELIVERED
15 }
```

Pizza

CTO - Status



```
8
9  /* ENUMERATOR */
10 enum STATUS {
11     o PLACED
12     o PREPARED
13     o DISPATCHED
14     o DELIVERED
15 }
```

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

Navigation icons

Pizza

CTO - Size & PizzaType



```
27 }
28 enum SIZE {
29     o small
30     o medium
31     o large
32 }
33 enum PIZZATYPE{
34     o americana
35     o carbonara
36     o margherita
37     o marinara
38     o napoli
39     o quattro
40     o romana
41 }
```

- Toppings
- Size
- Pizza Type
- Enumerator Types

Navigation icons

Pizza

CTO - Address - Customer - Restaurant



```
42 }
43 /* CONCEPT */
44 concept ADDRESS{
45     o String Name optional
46     o String NameNumber default="1"
47     o String Street default="High St"
48     o String PostCode default="NW44BT"
49 }
50
51 /* PARTICIPANT */
52 participant customer identified by customerID{
53     o String customerID
54     o ADDRESS deliveryAddress
55 }
56
57 participant pizzaOutlet identified by poID{
58     o String poID
59     o ADDRESS poAddress
60 }
61
62 participant pqc identified by pqcID{
63     o String pqcID
64 }
```

Navigation icons

Pizza

CTO - Order



```
70 }
71 /* current version only allows
72    1 pizza per order
73 * simply rectified by adding
74    array
75 * --> pizzaDetail[] pizzas
76 */
77 asset order identified by
78     orderID{
79         o String orderID
80         --> pizzaDetail pizza
81         --> pizzaOutlet restaurant
82         --> customer consumer
83         o STATUS status
84     }
```

- Where does ID come from?

Pizza

CTO - Order



```
70 }
71 /* current version only allows
72    1 pizza per order
73 * simply rectified by adding
74    array
75 * --> pizzaDetail[] pizzas
76 */
77 asset order identified by
78     orderID{
79         o String orderID
80         --> pizzaDetail pizza
81         --> pizzaOutlet restaurant
82         --> customer consumer
83         o STATUS status
84     }
```

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

Pizza

CTO - Order



```
70 }
71 /* current version only allows
72    1 pizza per order
73 * simply rectified by adding
74    array
75 * --> pizzaDetail[] pizzas
76 */
77 asset order identified by
78     orderID{
79         o String orderID
80         --> pizzaDetail pizza
81         --> pizzaOutlet restaurant
82         --> customer consumer
83         o STATUS status
84     }
```

- Where does ID come from?
- User generated, can be pseudo-random
- Comment on multiple 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 & Events



```
89
90 transaction prepareOrder{
91   --> order pizzaPrepared
92 }
93
94 event prepareOrderEvent{
95   --> order pizzaPrepared
96 }
97
98 transaction dispatchOrder{
99   --> order pizzaDispatched
100 }
101
102 event dispatchOrderEvent{
103   --> order pizzaDispatched
104 }
105
106 transaction deliverOrder{
107   --> order pizzaDelivered
108 }
109
110 event deliverOrderEvent{
111   --> order pizzaDelivered
112 }
```

Navigation icons

©i.mitchell@mdx.ac.uk

CST4025:L7

September 26, 2019

25 / 37

CustomerSeeSelf: Customers can only see themselves

Navigation icons

©i.mitchell@mdx.ac.uk

CST4025:L7

September 26, 2019

26 / 37

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.
14             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.
30             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

Navigation icons

©i.mitchell@mdx.ac.uk

CST4025:L7

September 26, 2019

27 / 37

Rules

ACL - Customer



```
49 rule customerPlaceOrder{
50   description: "customer places order"
51   participant: "org.pqc.uk.customer"
52   operation: ALL
53   resource: "org.pqc.uk.placeOrder"
54   action: ALLOW
55 }
56 rule customerReadRestaurant{
57   description: "customer has read access to
58     restaurants"
59   participant: "org.pqc.uk.customer"
60   operation: READ
61   resource: "org.pqc.uk.pizzaOutlet"
62   action: ALLOW
63 }
```

customerPlaceOrder:

Only a customer
can place an order
and access
transaction
placeOrder

customerReadRestaurant:

Customers are
permitted to read
pizzaOutlet details

Navigation icons

Rules

ACL - Restaurant



```
33 rule restaurantSeeSelf{
34   description: "restaurants can only view
35     their own details"
36   participant(p): "org.pqc.uk.pizzaOutlet"
37   operation: ALL
38   resource(r): "org.pqc.uk.pizzaOutlet"
39   condition: (p.getIdentifier()==r.
40     getIdentifier())
41   action: ALLOW
42 }
43 rule restaurantSeeOrders{
44   description: "restaurant can only see
45     their own orders"
46   participant(p): "org.pqc.uk.pizzaOutlet"
47   operation: ALL
48   resource(r): "org.pqc.uk.order"
49   condition: (p.getIdentifier()==r.
50     restaurant.getIdentifier())
51   action: ALLOW
52 }
```

restaurantSeeSelf:

Restaurant can only
see themselves

restaurantSeeOrders:

Restaurant can only
see orders placed at
their pizzaOutlet

Navigation icons

Rules

ACL - Restaurant



```
63 rule restaurantReadsCustomer{
64   description: "restaurant reads
65     customer"
66   participant: "org.pqc.uk.
67     pizzaOutlet"
68   operation: READ
69   resource: "org.pqc.uk.customer"
70   action: ALLOW
71 }
72 rule restaurantPlaceOrder{
73   description: "restaurant reads
74     order"
75   participant: "org.pqc.uk.
76     pizzaOutlet"
77   operation: READ, UPDATE//CANNOT
78     CREATE
79   resource: "org.pqc.uk.order"
80   transaction: "org.pqc.uk.
81     prepareOrder"
82   action: ALLOW
83 }
84 rule restaurantProcessOrder{
85   description: "restaurant process
86     order"
87   participant: "org.pqc.uk.
88     pizzaOutlet"
89   operation: ALL
90   resource: "org.pqc.uk.
91     prepareOrder"
92   action: ALLOW
93 }
```

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

Navigation icons

Rules

ACL - Restaurant



```
85 rule restaurantDispatchOrder{
86   description: "restaurant dispatch
      order access"
87   participant: "org.pqc.uk.
      pizzaOutlet"
88   operation: ALL
89   resource: "org.pqc.uk.
      dispatchOrder"
90   action: ALLOW
91 }
92 rule restaurantDeliverOrder{
93   description: "restaurant deliver
      order access"
94   participant: "org.pqc.uk.
      pizzaOutlet"
95   operation: ALL
96   resource: "org.pqc.uk.
      deliverOrder"
97   action: ALLOW
98 }
```

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

Navigation icons

Transactions

JS - Place Order



```
7  /*
8  * User submits order to restaurant
9  * @param {org.pqc.uk.placeOrder} placeOrder - pizza order
10 * @transaction
11 */
12 async function placeOrder(tx){
13   const ns='org.pqc.uk';
14   //create new order
15   var factory = getFactory();
16   var newOrder=factory.newResource(ns,'order',tx.orderID);
17   newOrder.pizza = tx.pizza;
18   newOrder.restaurant = tx.restaurant;
19   newOrder.consumer = tx.Customer;
20   newOrder.status = 'PLACED';
21   // add new order to the order registry
22   const orderReg = await getAssetRegistry(ns+'.order');
23   await orderReg.add(newOrder);
24 }
```

Navigation icons

Transactions

JS - Prepare Order



```
25 /*
26 * restaurant prepares order
27 * @param {org.pqc.uk.prepareOrder} prepareOrder - pizza order
28 * @transaction
29 */
30 async function prepareOrder(tx){
31   const ns='org.pqc.uk';
32   currentOrder = tx.pizzaPrepared;
33   if( currentOrder.status !== 'PLACED')
34   {
35     throw new Error('Current order'+currentOrder.orderID+' is in wrong status to be prepared');
36   }
37   else
38   {
39     currentOrder.status = 'PREPARED';
40   }
41   // update order with currentOrder
42   const orderReg = await getAssetRegistry(ns+'.order');
43   await orderReg.update(currentOrder);
44   // emit the event
45   const factory=getFactory();
46   const prepareOrderEvent=factory.newEvent(ns,'prepareOrderEvent');
47   prepareOrderEvent.pizzaPrepared=currentOrder;
48   emit(prepareOrderEvent);
49 }
```

Navigation icons

Transactions

JS - Dispatch Order



```
50 /*
51  * restaurant dispatches order
52  * @param{org.pqc.uk.dispatchOrder} dispatchOrder - pizza dispatched
53  * @transaction
54  */
55 async function dispatchOrder(tx){
56   const ns='org.pqc.uk';
57   Prepare currentOrder=tx.pizzaDispatched;
58   if( currentOrder.status !== 'PREPARED')
59   {
60     throw new Error('Current order has not been prepared');
61   }
62   else
63   {
64     currentOrder.status = 'DISPATCHED';
65   }
66   // update order with currentOrder
67   const orderReg = await getAssetRegistry(ns+'.order');
68   await orderReg.update(currentOrder);
69   // emit the event
70   const factory=getFactory();
71   const dispatchOrderEvent=factory.newEvent(ns,'dispatchOrderEvent');
72   dispatchOrderEvent.pizzaDispatched = currentOrder;
73   emit(dispatchOrderEvent);
74 }
```

Navigation icons

Transactions

JS - Deliver Order



```
75 /*
76  * customer receives order
77  * @param{org.pqc.uk.deliverOrder} deliverOrder - pizza delivered
78  * @transaction
79  */
80 async function deliverOrder(tx){
81   const ns='org.pqc.uk';
82   currentOrder=tx.pizzaDelivered;
83   if( currentOrder.status !== 'DISPATCHED')
84   {
85     throw new Error('Current order has not been dispatched');
86   }
87   else
88   {
89     currentOrder.status = 'DELIVERED';
90   }
91   // update order with currentOrder
92   const orderReg = await getAssetRegistry(ns+'.order');
93   await orderReg.update(currentOrder);
94   // emit the event
95   const factory=getFactory();
96   const deliverOrderEvent=factory.newEvent(ns,'deliverOrderEvent');
97   deliverOrderEvent.pizzaDelivered=currentOrder;
98   emit(deliverOrderEvent);
99 }
```

Navigation icons

References I



- [1] Nitin Gaur et al. *Hands-on Blockchain with Hyperledger: Building Decentralised Applications with Hyperledger Fabric and Composer*. Packt, 2018. ISBN: 9781788994521.
- [2] *Hyperledger Architecture, Volume 1*. 2017.
- [3] *Hyperledger Architecture, Volume 2*. 2018.

Navigation icons



- <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>