
# Ethereum
# Blockchain Development
# Week 7

## Introduction

The intention of this lab is to look at functions and how variables are stored. All the exercise are to be complete in the VM.

**Code Completion**

Writing code in a new language can be a steep learning curve. The approach here is to provide some code with underscores (_) that you are required to complete. These underscores are there to help you. By completing these exercises you will be improving your skills and knowledge of Solidity.

Each exercise starts on a new page. The red numbers in the right-hand margin are estimated minutes you should spend on each exercise.

# 1 Nodejs

5

The ‘Setup’ is covered in the video (w6-1.mpg) available from Ethereum-myLearning website. All today’s exercises are to be saved in week 7directory. Follow the instructions below:

- Open up the VM machine
- Open a terminal (CTRL-ALT-T)
- In your ’Ethereum’ directory create a week 7directory: `mkdir 7`
- The beginning of this seminar reminds us about arrays, these are simple programs and located in the first section. You can complete these in the VSCode development environment.

Code Completion

```
var a = new Array();
a.push(1);
a.push(2);
a.push(3);
console.log(a);
a.pop();
console.log(a);

_____ populate = ( __ => {
  for (___ i=0; i<5; i++)
    a.____(i);
})

_____ depopulate = ( __ =>{
  for (___ i=0; i<5; i++)
    a.___();
})

populate();
console.log(a);
depopulate();
console.log(a);
```

Figure 2.1: Example of node.js array methods `push` and `pop`.

# 2 Introduction to Arrays

5–10

An array is a data type that stores multiple elements of the same data type. It is often analogous to vectors (1-d) and matrices (2-d). The importance of arrays is that each element in the array can be accessed by an index. `pop` and `push` are defined in the list below.

**pop** : adds an element to the end of the array

**push** : removes the last element in the array

Complete the code in Fig. 2.1 and execute.

Code Completion

```
1 var a = [12,13,14,15,33,34,35,15];
2 console.log(a._______(15))
3 console.log(a._________(15))
```

Figure 2.2: Example of node.js indexOf code. Save as ex2.js. To execute type node ex2.js

## 2.1 indexOf

5

Returns the index of an array, given the item of in the array. Returns the first occurrence.

**indexOf** : Returns the first occurrence.

**lastIndexOf** : Returns the last occurence.

Complete the following in Fig 2.2
What is the expected outcome? Execute the program.

## 2.2 slice

5–10

Using the `slice` command, complete the following for the array, `a`, in Fig. **??**.

1. Display the first 3 elements
2. Display the last 3 elements
3. Display elements 3 to 5
4. Display the elements from the first occurence of 15

## 2.3 concat

5–10

Complete the following.

- In the code listing in Fig. **??** add the following code.
  `let b = [23, 24, 25, 26, 27, 28]`
- using the concat command, merge the two arrays into a third array, `c`
- display the new array, `c`

## 2.4 map

5–10

Using the `map` command, replace each element in the array, `a`, with its square. Display the new array.

Code Completion

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

for(___ i=0; i<_._____; i++){
  ++a[_];
  console.log(a[_]);
}

console.log(a);
```

Figure 3.1: Example of Node.js code using for for and arrays.

Code Completion

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

var sum=0;
a._______( (item)=>___+=item);
console.log(sum);

a._______(item)=> __item );
console.log(a);
```

Figure 3.2: Example of Node.js code using forEach loops and arrays.

# 3 Iteration

5–10

Using a for loop complete the following code to print out the array. This exercise uses the length operator to return the number of elements in an array. It also demonstrates the difference between scope variables and the use of `let` and `var`. Type the code in Fig. 3.1 and execute.

Can you adapt this to calculate the sum of all the integers in an Array?

## 3.1 forEach

5–10

Using the forEach calculate the sum of all the items in the following array, `a`. Complete the code in Fig. 3.3.

Code Completion

```
1 var a = [12,13,14,15,33,34,35,37,37,37];
2
3 function remove(x){
4   return x __ 20;
5 }
6 //remove items less than 20
7 var a1 = a._____(_____);
8 console.log('a1');
```

Figure 3.3: Example of Node.js for removing and filtering elements from an array.

## 3.2 filter

10–15

Returns an array of all the elements that satisfy a test.

Complete the following code:

Using filter how could you count the number of occurrences of 37 (hint: use length)?

# 4 Trader

5–10

Follow instructions from last week and upload the trader network.

- In the CTO add a `commoditiesOwned` array in the Trader
- Create new participants or Traders with this new structure as in the lecture: Trader 111, 222, and 333.
- Create new assets or Commodities assigned to these traders.
- Create a new transaction Trade that will:
  - trade a commodity from the buyer to the seller
  - update the commoditiesOwned in the buyer
  - update the commoditiesOwned in the seller
- Test the function by making an appropriate transaction

# 5 Pizza

15–20

Download the `.bna` file from the website. Open composer-playground in a compatible browser. Using this `.bna` file you can either upload it into playground, or extract individual files and cut and paste them appropriately. After uploading to composer playground, and logging in as Administrator complete the following:

1. Create some customer details:
    - Enter the following values, ‘C001’, ‘Andrei’, ‘111’, ‘Wood St’, ‘E108UT’ for ‘customerID’, ‘Name’, ‘NameNumber’, ‘Street’ and ‘PostCode’, respectively.
    - Enter the following values, ‘C002’, ‘Xu’, ‘222’, ‘Station Rd’, ‘SE78HD’ for ‘customerID’, ‘Name’, ‘NameNumber’, ‘Street’ and ‘PostCode’, respectively.
2. Create some restaurant details:
    - Enter the following values, ‘R001’, ‘Piece Of Pizza’, ‘110’, ‘High St’ and ‘SE104BT’ for ‘poID’, ‘Name’, ‘NameNumber’, ‘Street’ and ‘PostCode’, respectively.
    - Enter the following values, ‘R002’, ‘PizzaSlice’, ‘220’, ‘High St’ and ‘SE222BT’ for ‘poID’, ‘Name’, ‘NameNumber’, ‘Street’ and ‘PostCode’, respectively.
3. Create some pizza details:
    - ‘0001’, ‘americana’ for ‘pID’ and ‘pizzaType’, respectively.
    - ‘0002’, ‘carbonara’ for ‘pID’ and ‘pizzaType’, respectively.
    - ‘0003’, ‘margherita’ for ‘pID’ and ‘pizzaType’, respectively.
4. Create some auditor details:
    - Enter the following value, ‘P122A’ for ‘pqcID’.
5. Enter the ‘ID Registry’ and create customers ‘C1’ and ‘C2’ for customers with ID’s ‘C001’ and ‘C002’, respectively.

6. Enter the ‘ID Registry’ and create pizzaOutlets ‘R1’ and ‘R2’ for customers with ID’s ‘R001’ and ‘R002’, respectively.
7. Using the transaction, ‘placeOrder’, create two new orders for the customer with ‘customerID’, ‘C001’. The new orders should have the following details:
    - ‘1111’, ’C001’, ’0001’, ’R001’ for ‘orderID’, ‘Customer’, ‘pizza’ and ‘restaurant’, respectively.
    - ‘1112’, ’C001’, ’0003’, ’R002’ for ‘orderID’, ‘Customer’, ‘pizza’ and ‘restaurant’, respectively.
8. Using the transaction, ‘placeOrder’, create two new orders for the customer with ‘customerID’, ‘C002’. The new orders should have the following details:
    - ‘2221’, ’C002’, ’0002’, ’R001’ for ‘orderID’, ‘Customer’, ‘pizza’ and ‘restaurant’, respectively.
    - ‘2222’, ’C002’, ’0003’, ’R002’ for ‘orderID’, ‘Customer’, ‘pizza’ and ‘restaurant’, respectively.
9. Alter the code to allow for multiple orders?
10. Alter the code to allow for toppings to be included in orders?
11. Can the restaurant change or view its orders. Alter the code to allow the restaurant to view the menu.
12. Alter the code to allow the restaurant to change its own menu.
13. Alter the code to allow a dispatch company to handle the delivery process
14. Alter the code to allow the quality auditor to view and read all the transactions.

# References