

Ethereum Client side DApps

Install Node.js: <https://nodejs.org/en/download>

Install Ganache. Ganache is locally installed Blockchain. It will be used for development and testing.

The screenshot shows the Ganache application window. At the top, there's a navigation bar with icons for ACCOUNTS, BLOCKS, TRANSACTIONS, CONTRACTS, EVENTS, and LOGS. Below this is a status bar with various metrics: CURRENT BLOCK (0), GAS PRICE (20000000000), GAS LIMIT (6721975), HARDFORK (PETERSBURG), NETWORK ID (5777), RPC SERVER (HTTP://127.0.0.1:7545), MINING STATUS (AUTOMINING), and WORKSPACE (QUICKSTART). The main area displays the MNEMONIC (game tail athlete announce law link wrestle exercise wheel tree click clump) and the HD PATH (m/44'/60'/0'/0/account_index). A table lists six accounts, each with an ADDRESS, BALANCE (100.00 ETH), TX COUNT (0), and INDEX (0 to 6). A tooltip warning is visible over the first account's mnemonic: "note: this mnemonic is not secure, don't use it on a public blockchain."

ADDRESS	BALANCE	TX COUNT	INDEX
0xd89eef6d6e97E061D936149d5f81D21568C58896	100.00 ETH	0	0
0x820F84865fF7Ba8590c23871A2f48602Df28B11C	100.00 ETH	0	1
0x11318953c0F89B8Da6562DcBef1ebE54898412bD	100.00 ETH	0	2
0x4CC9050c826d9e6C233b948105a47A6108571801	100.00 ETH	0	3
0x92EC99aAaef553E41E8dF95E98eF8542Ce826cAd	100.00 ETH	0	4
0x58feCA14e451d35B3122D13c98d339e224931b25	100.00 ETH	0	5
0x98A7C3Aecb34f7f2aCEe7f33B87EacE3D7b9f643	100.00 ETH	0	6

Truffle Framework

The Truffle framework provides a compiler for the Smart Contract development, an automated testing tool, deployment tools, network management tools, developer console, script runner and client development tools.

For this demonstration Truffle version 5.0.2 will be used. Python is required

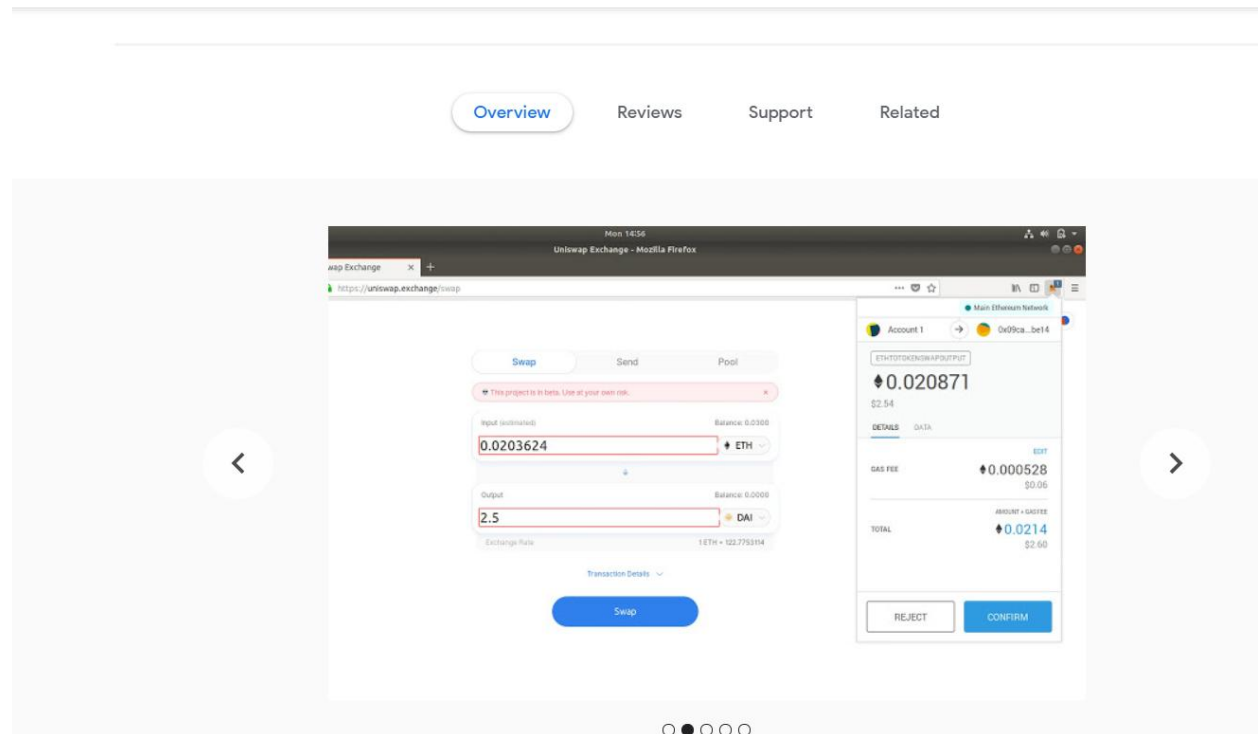
For Python : <https://www.python.org/downloads/>

To install truffle run the command: `npm install -g truffle@5.0.2`

Metamask

Metamask is a plugin that enables a web browser to browse a Blockchain. For this demonstration we will install Metamask into the Chrome. Browser.

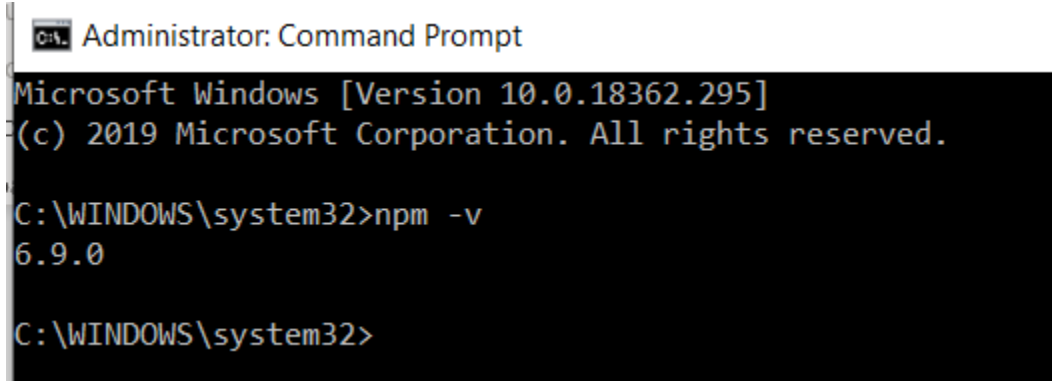
Open Chrome and search the Chrome Web Store for Metamask and install the Metamask extension.



The truffle is a tool that provides a framework for Smart Contract development, and deployment.
Install chocolatey , <https://chocolatey.org/>

Step 1

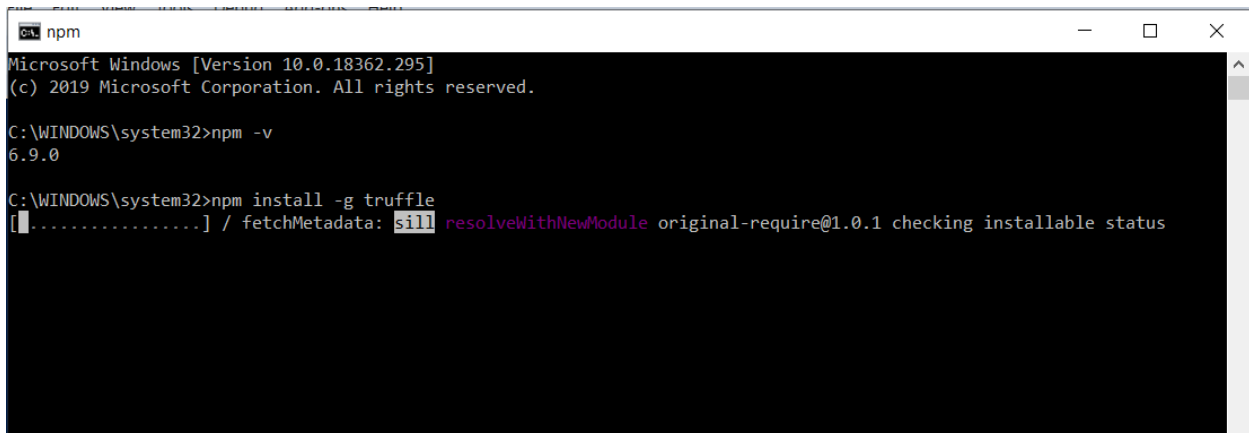
Install NPM (Node Package Manager) on Windows, <https://nodejs.org/en/>



```
Administrator: Command Prompt
Microsoft Windows [Version 10.0.18362.295]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>npm -v
6.9.0

C:\WINDOWS\system32>
```



```
npm
Microsoft Windows [Version 10.0.18362.295]
(c) 2019 Microsoft Corporation. All rights reserved.

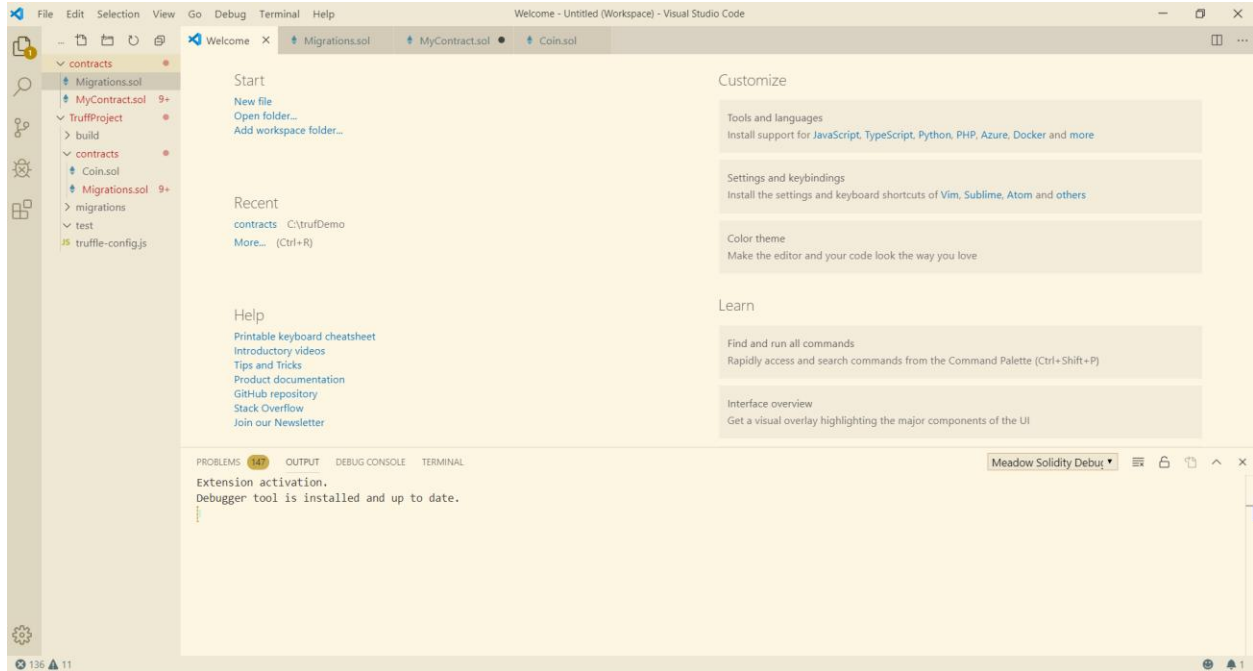
C:\WINDOWS\system32>npm -v
6.9.0

C:\WINDOWS\system32>npm install -g truffle
[.....] / fetchMetadata: sill resolveWithNewModule original-require@1.0.1 checking installable status
```

- Install Visual Studio Code
- Visual Studio Code runs on Windows and Mac, macos 10.10+
- Install the following extensions:
- Solidity
- Solidity Extended
- Solidity Debugger

Introduction to Blockchain Development with Ethereum by Coding-Bootcamps.com

Session 8- Ethereum Client-side Applications




Install: lite-server: `npm install lite-server -g@2.3.0`

Create a project

Open your command shell and create a directory for the project.


You can run: `truffle version`, to validate your truffle version.

Run: `mkdir ethereumClient`

 Administrator: Command Prompt

```
C:\>mkdir ethereumClient
```


For truffle version, run: `truffle version`

 Administrator: Command Prompt

```
C:\ethereumClient>truffle version
Truffle v5.0.2 (core: 5.0.2)
Solidity v0.5.0 (solc-js)
Node v10.16.3

C:\ethereumClient>
```

Start, or initialize the project, run: `truffle inti`

 Administrator: Command Prompt

```
C:\ethereumClient>truffle init

✓ Preparing to download
✓ Downloading
✓ Cleaning up temporary files
✓ Setting up box




Unbox successful. Sweet!

Commands:

  Compile:      truffle compile
  Migrate:      truffle migrate
  Test contracts: truffle test

C:\ethereumClient>
```

The **truffle init** command will create the project framework below

<input type="checkbox"/> Name	Date modified	Type
 contracts	9/24/2019 7:18 AM	File folder
 migrations	9/24/2019 7:18 AM	File folder
 test	9/24/2019 7:18 AM	File folder
 truffle-config.js	9/24/2019 7:18 AM	JavaScript File

Enter the contract code below into a file called: StudentContract.sol, and save it to the project's contract folder

```
//Start here
pragma solidity <=0.5.1;
contract StudentContract {

    uint256 public studentCounter = 0;

    mapping(uint => Student) public students;

    constructor() public {
        addStudent("Michael", "Sims");
    }

    struct Student{
        uint _id1;
        string _fname;
        string _lname;
    }

    event studentListed(uint256 id,
        string fname1,
        string lname1);

    function addStudent(string memory _fname, string memory _lname) public {
        studentCounter++;
        students[studentCounter] = Student(studentCounter, _fname, _lname);
    }

    function studentList() public {
        string memory fname0;
        string memory lname0;
        uint256 sid;
```

```
uint256 sCnt = studentCounter + 1;

for (uint256 i=1; i<sCnt; i++){

    sid = students[i]._id1;
    frame0 = students[i]._fname;
    lname0 = students[i]._lname;

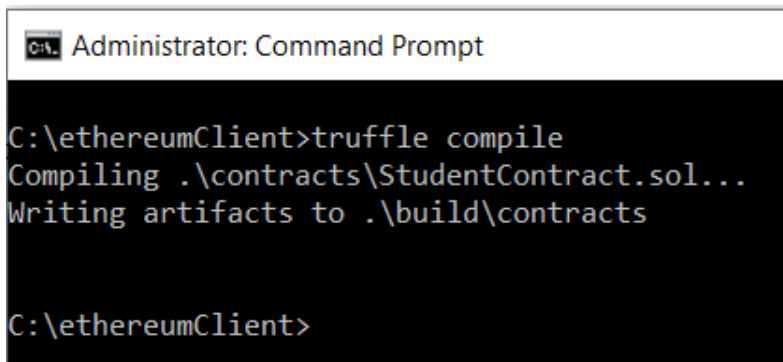
    emit studentListed(sid,frame0,lname0);

}

}

}
//End here
```

From the top of the project, run: truffle compile.




```
C:\ethereumClient>truffle compile
Compiling .\contracts\StudentContract.sol...
Writing artifacts to .\build\contracts


C:\ethereumClient>
```

Migrations

Run: `truffle migrate -reset`

 Administrator: Command Prompt

```
C:\ethereumClient>truffle migrate --reset
```

 Administrator: Command Prompt

```
> gas price:      20 gwei
> value sent:     0 ETH
> total cost:     0.00569688 ETH
```

```
> Saving artifacts
```

```
-----
> Total cost:     0.00569688 ETH
```

2_deploy_migration.js

=====

```
Replacing 'StudentContract'
```

```
> transaction hash: 0x7c393c604d50c0cd0f6e6ab0e40b54d0b7077c60064f48974b7cbe596352a918
> Blocks: 0        Seconds: 0
> contract address: 0x25262C9917bb23f00DEDFe4aDB04EA51f1785E3f
> account:         0x3b27d562f7A6c9eA80d06dAD76723406e8370af6
> balance:         99.87079392
> gas used:        692157
> gas price:       20 gwei
> value sent:      0 ETH
> total cost:      0.01384314 ETH
```

```
> Saving artifacts
```

```
-----
> Total cost:      0.01384314 ETH
```

Summary

=====

```
> Total deployments: 2
> Final cost:        0.01954002 ETH
```

C:\ethereumClient>

Introduction to Blockchain Development with Ethereum by Coding-Bootcamps.com
Session 8- Ethereum Client-side Applications

Successfully deployed to Ganache

Administrator: Command Prompt

```
> gas price:      20 gwei
> value sent:     0 ETH
> total cost:     0.00569688 ETH

> Saving artifacts
-----
> Total cost:     0.00569688 ETH

2_deploy_migration.js
=====

Replacing 'StudentContract'
-----
> transaction hash: 0x7c393c604d50cd0f6e6ab0e40b54d0b7077c60064f48974b7cbe596352a918
> Blocks: 0        Seconds: 0
> contract address: 0x25262C9917bb23f00DEDFe4aDB04EA51f1785E3f
> account:         0x3b27d562f7A6c9eA80d06dAD76723406e8370af6
> balance:         99.87079392
> gas used:        692157
> gas price:       20 gwei
> value sent:      0 ETH
> total cost:      0.01384314 ETH

> Saving artifacts
-----
> Total cost:      0.01384314 ETH

Summary
=====
> Total deployments: 2
> Final cost:       0.01954002 ETH

C:\ethereumClient>
```

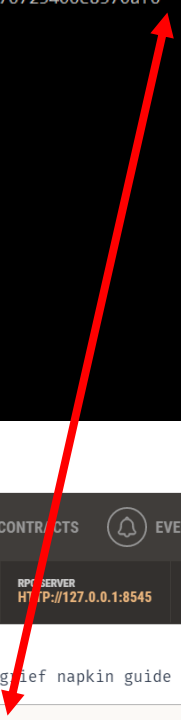
Ganache

ACCOUNTS BLOCKS TRANSACTIONS CONTRACTS EVENTS LOGS SEARCH

CURRENT BLOCK	GAS PRICE	GAS LIMIT	HARDFORK	NETWORK ID	RPC SERVER	MINING STATUS	WORKSPACE
33	20000000000	6721975	PETERSBURG	5777	HTTP://127.0.0.1:8545	AUTOMINING	BASEETHEREUM


MNEMONIC ?
slight verify mix empower trouble method allow diary grief napkin guide napkin

ADDRESS	BALANCE
0x3b27d562f7A6c9eA80d06dAD76723406e8370af6	99.87 ETH




Run: `truffle console`

Revalidate the contract is running on the Blockchain and is up to date

 Administrator: Command Prompt - truffle

```
C:\ethereumClient>truffle console
truffle(development)>
```

Run: `studentlist = await StudentContract.deployed()`
For results, run: `studentlist.address`. This is the contract's address.

 Administrator: Command Prompt - truffle console

```
C:\ethereumClient>truffle console
truffle(development)> studentlist = await StudentContract.deployed()
undefined
truffle(development)> studentlist.address
'0x25262C9917bb23f00DEDFe4aDB04EA51f1785E3f'
truffle(development)>
```

Add a new student, run: `student = await studentlist.addStudent("Jane","Doe")`
For results, run: `student`

```
truffle(development)> student = await studentlist.addStudent("Jane","Doe")
undefined
truffle(development)> student
{ tx:
  '0x8d8e7fb540efe41aab6f7b347a98548f53c07d1898590217ace436dcece7f331',
  ...
}
```

Run: `student = await studentlist.studentCounter()`
For results, run: `student`

Run: `student.toString()`

```
truffle(development)> student = await studentlist.studentCounter()
undefined
truffle(development)> student
<BN: 2>
truffle(development)> student.toString()
'2'
truffle(development)>
```

Run: `student = await studentlist.students(1)`, and then

`student = await studentlist.students(2)`

For results, run: `student`

```
truffle(development)> student = await studentlist.students(1)
undefined
truffle(development)> student
Result {
  '0': <BN: 1>,
  '1': 'Michael',
  '2': 'Sims',
  _id1: <BN: 1>,
  _fname: 'Michael',
  _lname: 'Sims' }
truffle(development)> student = await studentlist.students(2)
undefined
truffle(development)> student
Result {
  '0': <BN: 2>,
  '1': 'Jane',
  '2': 'Doe',
  _id1: <BN: 2>,
  _fname: 'Jane',
  _lname: 'Doe' }
truffle(development)>
```

Run: `truffle(development)> student._id1`

To better format the results, run: `student._id1.toString()`

```
truffle(development)> student._id1
<BN: 2>
truffle(development)> student._id1.toString()
'2'
truffle(development)>
```

Get the students first and last names.

Run: `student._fname`

For string conversion, run: `student._fname.toString()`

```
truffle(development)> student._fname
'Jane'
truffle(development)> student._fname.toString()
'Jane'
truffle(development)>
```

Run: `student._lname`

For string conversion, run: `student._lname.toString()`

```
truffle(development)> student._lname
'Doe'
truffle(development)> student._lname.toString()
'Doe'
truffle(development)>
```

Now let's look at the client side files. We will not set up the client files as the run time environment must be exactly configured.

This is a simulation:

Let's go back to our Ethereum project and create a new directory for the client

Add client files, in folder src:

- `bs-config.json`
- `src/index.html`
- `src/app.js`

Run server: `npm run dev`

More Resources

To read more on blockchain and understand it in depth, the reading the following articles are highly recommended:

- History and Evolution of Blockchain Technology from Bitcoin
- Overview of Blockchain evolution and phases from Ethereum to Hyperledger
- Comprehensive overview and analysis of blockchain use cases in many industries
- Overview of blockchain technology and blockchain development

Also, the following are more tutorials and resources on Ethereum blockchain development.

- How to Write Ethereum Smart Contracts with Solidity in 1 hour
- How to Build Auction DApp with Ethereum and Solidity Programming Language
- How to Work with Ethereum Blockchain Applications through Remix IDE
- Certified Solidity Professional Certification exam
- Learn Ethereum: Build your own DApps with Ethereum and smart contracts book by Brian Wu