



coding-bootcamps.com

Introduction to Blockchain Development
with Ethereum



Coding Bootcamps

By Jim Sullivan from [Coding Bootcamps](https://coding-bootcamps.com)

Ethereum Client-Side Applications

The Building Blocks of Ethereum DApps

About Instructor

- **Jim Sullivan** is a senior blockchain instructor and developer at [DC Web Makers](#).
- As a Software engineer with 18 years of experiences, Jim leads an expert team in Blockchain development, DevOps, Cloud, application development, and the SAFe Agile methodology.
- Jim is an expert in all blockchain platforms like Hyperledger, Ethereum, and Corda.

Prerequisite Courses

Taking the below courses are highly recommended:

[Intro to Blockchain Technology](#)

[Intro to JavaScript](#)

[Learn Node.JS, Express.JS and MongoDB](#)

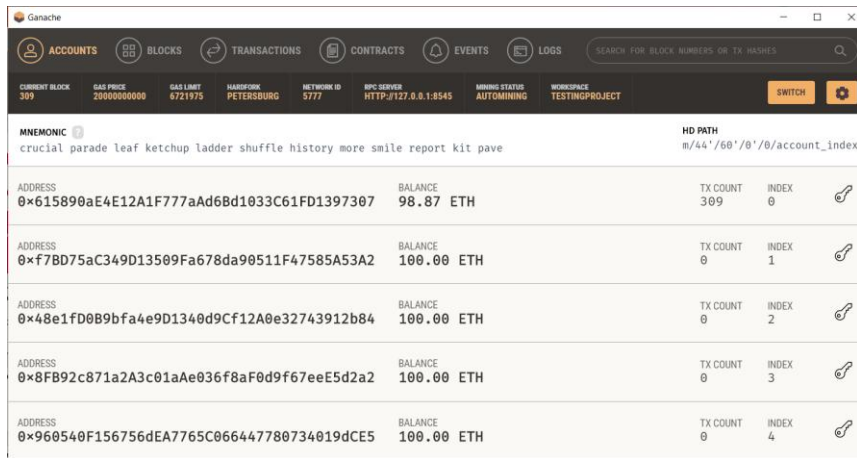
Recap

- What we have learned so far?

Ethereum Client-Side Applications


Building Blocks of Ethereum DApps


- Install Node.js: <https://nodejs.org/en/download>
- Install Ganache: <https://www.trufflesuite.com/ganache>
- Ganache is a private Ethereum Blockchain that runs locally.
- Ganache is used for Ethereum development and testing.
- Ganache comes with a set of Ethereum accounts.
- Each account has 100 ETH.
- Create a Ganache workspace called myList





Ethereum Client-Side Applications


Note of the host and port where Ganache is running


 Ganache


 ACCOUNTS

 BLOCKS

 TRANSACTIONS

 CONTRACTS

 EVENTS

 LOGS

SEARCH FOR

CURRENT BLOCK 309	GAS PRICE 20000000000	GAS LIMIT 6721975	HARDFORK PETERSBURG	NETWORK ID 5777	RPC SERVER HTTP://127.0.0.1:8545	MINING STATUS AUTOMINING	WORKSPACE TESTINGPROJECT
----------------------	--------------------------	----------------------	------------------------	--------------------	-------------------------------------	-----------------------------	-----------------------------

MNEMONIC ?

crucial parade leaf ketchup ladder shuffle history more smile report kit pave

ADDRESS

0x615890aE4E12A1F777aAd6Bd1033C61FD1397307

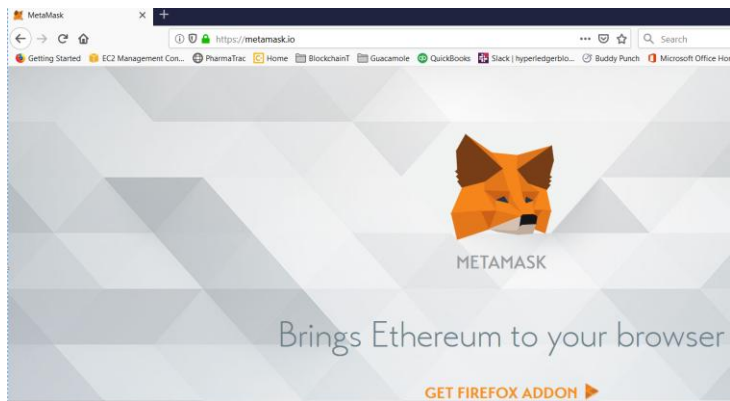
BALANCE

98.87 ETH

Ethereum Client-Side Applications

Building Blocks of Ethereum DApps

- Install Metamask for either Chrome, FireFox, etc.
 - <https://metamask.io/>
 - Metamask is a browser extension
 - Metamask makes a web browser, an Ethereum Browser.
 - Install the Metamask browser extension



Ethereum Client-Side Applications

Building Blocks of Ethereum DApps

- Create a Metamask account



< Back

Restore your Account with Seed Phrase

Enter your secret twelve word phrase here to restore your vault.

Wallet Seed

Separate each word with a single space

New Password (min 8 chars)

Confirm Password

Restore

Ethereum Client-Side Applications

Building Blocks of Ethereum DApps

- Validate the installed version of Node.js package manager (NPM)
- Open a command prompt as an administrator or open a PowerShell command shell, as an administrator.
- Run: `node -v`



Administrator: Command Prompt

```
C:\>node -v
v10.16.3

C:\>
```

Ethereum Client-Side Applications

Building Blocks of Ethereum DApps

- Install Truffle:
- Open a command prompt or PowerShell prompt as administrator.
- Run: `npm install -g truffle@5.0.2`
- After the install, run: `truffle version`
- Download GitHub repository:
<https://github.com/jjsull1van/coding-bootcamps/tree/master/Ethereum>

Administrator: Command Prompt

```
C:\WINDOWS\system32>truffle version
Truffle v5.0.2 (core: 5.0.2)
Solidity v0.5.0 (solc-js)
Node v10.16.3
```

```
C:\WINDOWS\system32>
```

Ethereum Client-Side Applications

Building Blocks of Ethereum DApps

- Create a Truffle project called ethereumClient
 - CD into the ethereumClient directory
 - Run: `truffle init`
 - Using Windows Explorer, or a file manager, navigate to the ethereumClient folders.
 - At the ethereumClient level, create a file called `package.json`

C:\> Administrator: Command Prompt

```
C:\ethereumClient>truffle init
```

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

```
Unbox successful. Sweet!
```

```
Commands:
```

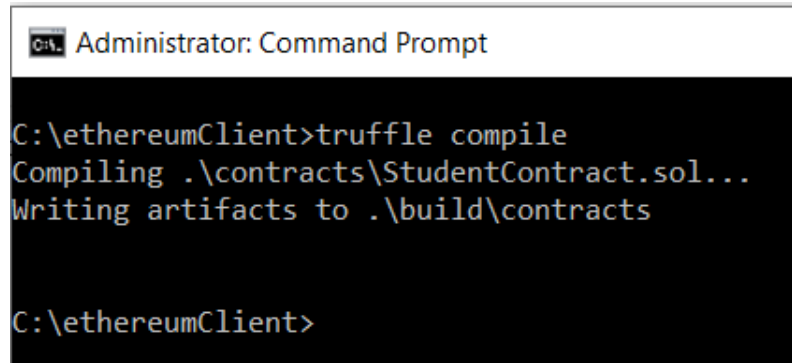
```
Compile:      truffle compile
Migrate:      truffle migrate
Test contracts: truffle test
```

```
C:\ethereumClient>
```

Ethereum Client-Side Applications

Building Blocks of Ethereum DApps

- Compile the contract
 - Navigate into the ethereumClient project
 - Copy the StudentContract.sol file into the Contracts folder.
 - From the top of the project, run: `truffle compile`.
 - Notice the newly created build folder.
 - The build folder contents has Ethereum byte code.



```
Administrator: Command Prompt


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

C:\ethereumClient>
```

Ethereum Client-Side Applications

Building Blocks of Ethereum DApps

- Copy the file `2_deploy_migration.js` into the migrations folder
 - The migrations folder deploys the contract to the Ethereum Blockchain.
 - The deployment target host is based on data in the `truffle-config.js` file. The Smart Contract will be deployed locally.
 - Run the migration from the Truffle project.
 - Run: `truffle migration --reset`

 Administrator: Command Prompt

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

Ethereum Client-Side Applications

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
> contract address: 0x25262C9917bb23f00EDFe4aDB04EA51f1785E3f
> 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 33	GAS PRICE 20000000000	GAS LIMIT 6721975	HARDFORK PETERSBURG	NETWORK ID 5777	RPC SERVER HTTP://127.0.0.1:8545	MINING STATUS AUTOMINING	WORKSPACE BASEETHEREUM
---------------------	--------------------------	----------------------	------------------------	--------------------	-------------------------------------	-----------------------------	---------------------------

MNEMONIC ?

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

ADDRESS

0x3b27d562f7A6c9eA80d06dAD76723406e8370af6

BALANCE

99.87 ETH

Ethereum Client-Side Applications

Building Blocks of Ethereum DApps

- Run the Truffle console: `truffle console`
 - Run asynchronous functions to return data from Ethereum, back to truffle console.
 - Get the contracts address on Ethereum
 - Run: `studentlist = await StudentContract.deployed()`

Administrator: Command Prompt - truffle

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

Administrator: Command Prompt - truffle console

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


Ethereum Client-Side Applications

Building Blocks of Ethereum DApps

- Run the addStudent function to add a student to the list on the Ethereum Blockchain.
 - Student Jane Doe was added.
 - Get the student count by querying studentCounter
 - Run: `student = await studentlist.studentCounter()`

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

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

Ethereum Client-Side Applications

Building Blocks of Ethereum DApps

- Query `students()` to output each student in the list from Ethereum.
 - Run: `student = await studentlist.students(1)`, and then 2 and 3.

```
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)>
```

Ethereum Client-Side Applications

Building Blocks of Ethereum DApps

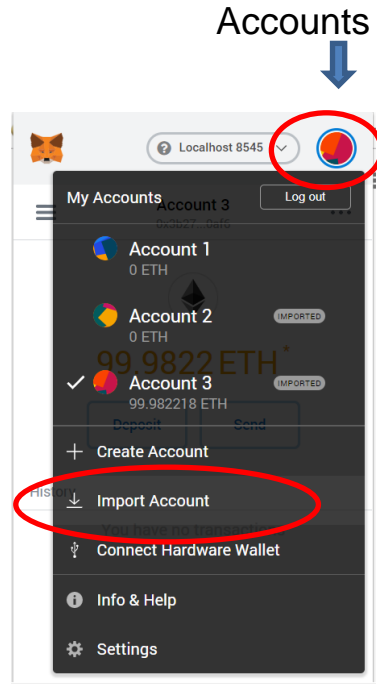
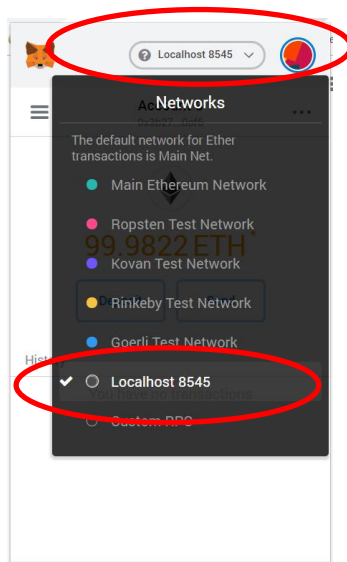
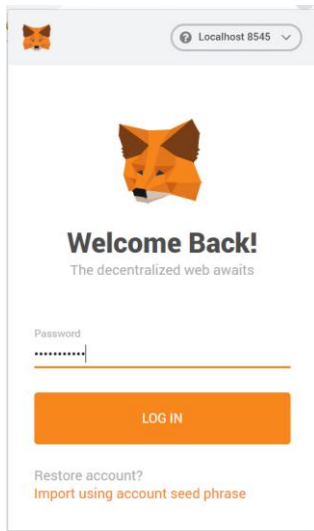
- These examples show how DApps communicate with the Ethereum Blockchain.
 - HTML, and JavaScript will be added to the Truffle project
 - The HTML and JavaScript will perform the same transactions with Ethereum as these examples for DApps.

```
truffle(development)> student = await studentlist.students(3)
undefined
truffle(development)> student
Result {
  '0': <BN: 3>,
  '1': 'Joe',
  '2': 'Dobbs',
  _id1: <BN: 3>,
  _fname: 'Joe',
  _lname: 'Dobbs' }
truffle(development)> student._id1
<BN: 3>
truffle(development)> student._id1.toString()
'3'
truffle(development)>
```

Ethereum Client-Side Applications

Building Blocks of Ethereum DApps

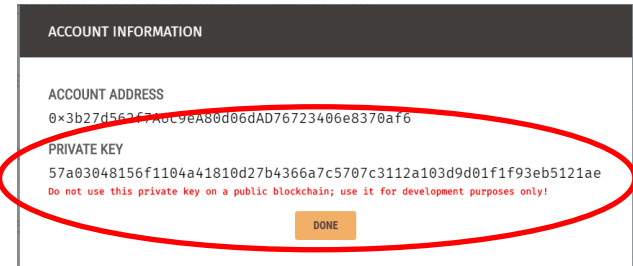
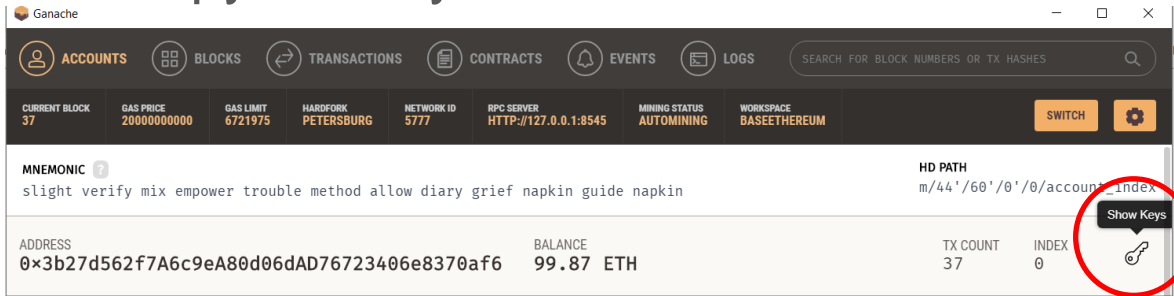
- Metamask
 - Make sure Ganache is running.
 - Connect to Metamask.
 - Connect to the local Ethereum network.
 - Click Accounts button and Import an Ethereum account



Ethereum Client-Side Applications

Building Blocks of Ethereum DApps

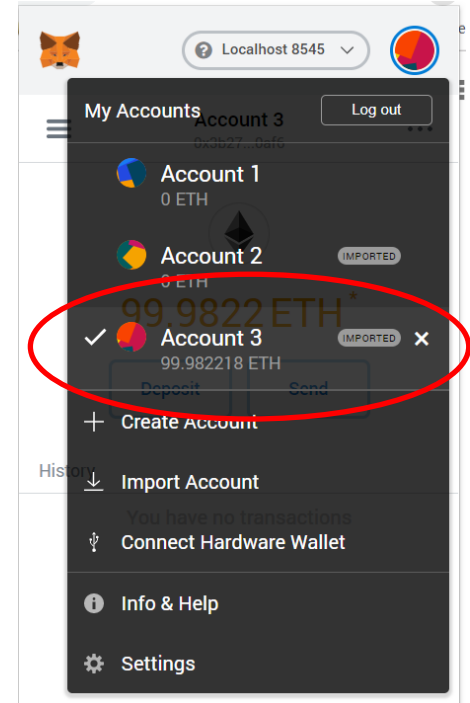
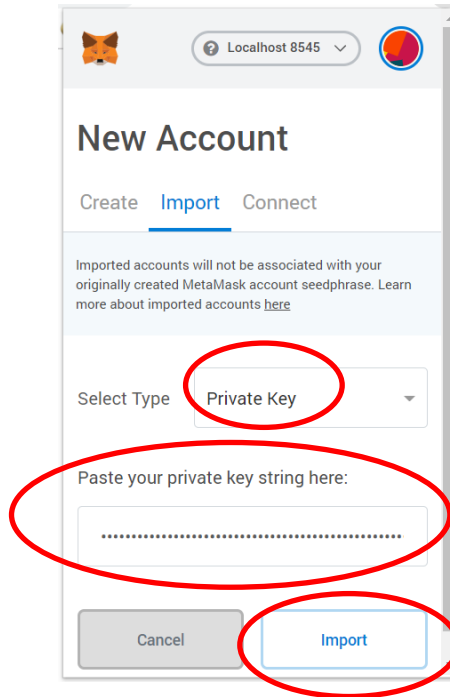
- Go back to Ganache and the account's private key
- Copy the key to the buffer.



Ethereum Client-Side Applications

Building Blocks of Ethereum DApps

- Select Private Key
- Paste the Private Key
- Click Import



Ethereum Client-Side Applications

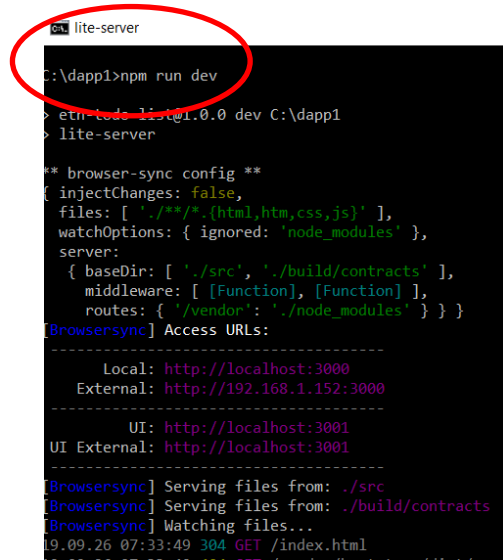
Building Blocks of Ethereum DApps

- Web3
 - Docs: <https://web3js.readthedocs.io/en/v1.2.1/getting-started.html>
 - web3.js is a collection of API libraries which allow connects with a local or remote Ethereum node, using an http or IPC connection.
 - The web3 JavaScript library interacts with the Ethereum Blockchain.
 - Metamask works with web3.js
 - To install web3, run:
 - `npm install web3`, **or** `npm install -g web3@0.19`

Ethereum Client-Side Applications

Building Blocks of Ethereum DApps

- The Lite Server is a good server for Ethereum development
 - Docs: <https://www.npmjs.com/package/light-server>
 - Install Lite Server: `npm install lite-server`
 - Run server: `npm run dev`



```
lite-server
C:\dapp1>npm run dev

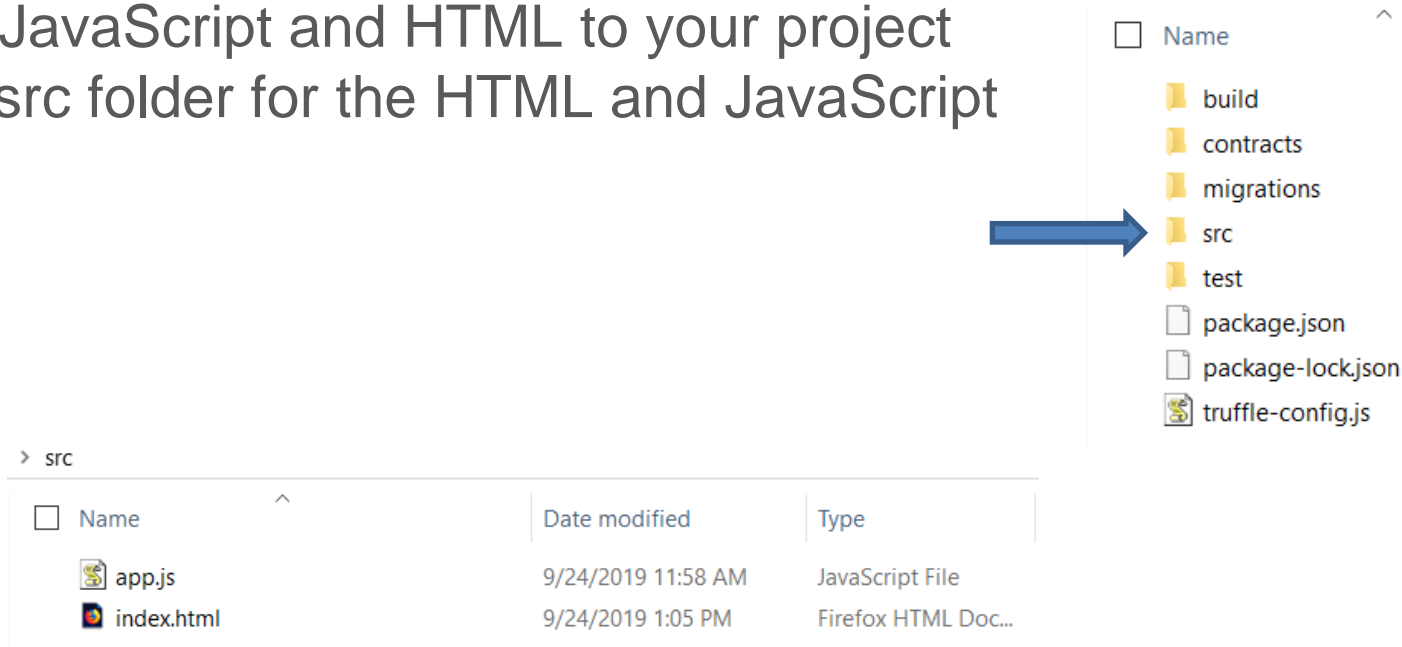
> etl-cto: list@1.0.0 dev C:\dapp1
> lite-server

** browser-sync config **
{ injectChanges: false,
  files: [ './**/*.html', './**/*.css', './**/*.js' ],
  watchOptions: { ignored: 'node_modules' },
  server:
    { baseDir: [ './src', './build/contracts' ],
      middleware: [ [Function], [Function] ],
      routes: { '/vendor': './node_modules' } } }
[Browsersync] Access URLs:
-----
    Local: http://localhost:3000
  External: http://192.168.1.152:3000
-----
    UI: http://localhost:3001
  UI External: http://localhost:3001
-----
[Browsersync] Serving files from: ./src
[Browsersync] Serving files from: ./build/contracts
[Browsersync] Watching files...
19.09.26 07:33:49 304 GET /index.html
```


Ethereum Client-Side Applications

Building Blocks of Ethereum DApps

- Adding JavaScript and HTML to your project
 - Add src folder for the HTML and JavaScript



Ethereum Client-Side Applications

Index.html

```
<div class="container-fluid">
  <div class="row">
    <main role="main" class="col-lg-12 d-flex justify-content-center">
      <div id="loader" class="text-center">
        <p class="text-center">Please wait .....</p>
      </div>
      <div id="content">
        <form onSubmit="App.studentCounter(); return false;">
          <input id="newTask" type="text" class="form-control" placeholder="Add Student..." required>
          <input type="submit" hidden="">
        </form>
        <ul id="studlist" class="list-unstyled">
          <div class="taskTemplate" class="checkbox" style="display: none">
            <label>
              <input type="checkbox" />
              <span class="content">Student content placeholder ...</span>
            </label>
          </div>
        </ul>
      </div>
    </main>
  </div>
</div>
```

- The JavaScript will call the contract, and return the results to the HTML i.e. the DApp

app.js

```
App = {
  loading: false,
  contracts: {},

  load: async () => {
    await App.loadWeb3()
    await App.loadAccount()
    await App.loadContract()
    await App.render()
  },

  // https://medium.com/metamask/https-medium-com-metamask-wallet-provider
  loadWeb3: async () => {
    if (typeof web3 !== 'undefined') {
      App.web3Provider = web3.currentProvider
      web3 = new Web3(web3.currentProvider)
    } else {
      window.alert("Please connect to Metamask.")
    }
  },

  // Modern dapp browsers
```

```
loadContract: async () => {
  // Create a JavaScript version of the smart contract
  const student = await $.getJSON('StudentContract.json')
  App.contracts.StudentContract = TruffleContract(student)
  App.contracts.StudentContract.setProvider(App.web3Provider)

  // Hydrate the smart contract with values from the blockchain
  App.studentlist = await StudentContract.deployed()
},

render: async () => {
  // Prevent double render
  if (App.loading) {
    return
  }
}
```

Ethereum Client-Side Applications

Summary

- Ganache and Node.js
- Metamask
- The Truffle suite
- Migrating a deploying a contract to Ethereum (Ganache)
- Using the Truffle console to call the contract on the Ethereum Blockchain
- Using the Truffle console as a DApp
- Connecting the browser to Ethereum (Ganache)
- Web3
- The Lite Server and client-side HTML and JavaScript

Assessment

1. Ganache is an Ethereum Blockchain that runs locally?
2. True or False: Metamask is a browser extension that support Ethereum?
3. True or False: Truffle does not include create project?
4. True or False: Truffle migrates and deploys contracts to Ethereum
5. True or False: The Truffle console can run deployed contracts?
6. True or False: The Truffle console simulates DApps?
7. True or False: No key is required to connect to Ganache?
8. True or False: Web3 is a Python framework for Ethereum?
9. True or False: JavaScript and HTML make calls to the contract.
10. True or False: The Lite Server is not used for development.

More Resources

- [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](#)

More Resources...

- [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

More Blockchain Training

- [Blockchain Management with Hyperledger for System Admins](#)
- [Hyperledger Fabric and Composer for Developers](#)
- [Intro to Blockchain Cybersecurity](#)
- [Learn Solidity Programming by Examples](#)
- [Learn Blockchain Dev with Corda R3](#)
- [Intro to Hyperledger Sawtooth for System Admins](#)

Next Session

Testing Ethereum Contracts



coding-bootcamps.com

Thank you



Coding
Bootcamps