ETH - Ethereum Training

BILGINC
IT Academy
info@bilginc.com
+44 20 3807 0925
+90 212 282 7700

Duration: **3 Day**

Overview

Ethereum Training London The student will be exposed to Ethereum's adaptable feature set which allows the developer to design decentralized applications for countless applications. Students will also participate in hands-on programming lab sessions to learn, develop, and advance their skills in Ethereum Training London development.

The Ethereum Training London is designed for those seeking an in-depth understanding and development experience of the Ethereum Blockchain platform Ethereum Training London. Students will participate in approximately 50% programming lab time providing practical experience, enhancing their knowledge and existing skill set. Due to the technical programming lab content covered in this course, it is not recommended for those without programming knowledge and experience.

Prerequisites

- Absolutely Necessary: Basic knowledge in JavaScript / HTML
- Advantageous: Basic knowledge in C ++ / Java, data types
- Advantageous: Basic knowledge with git repositories
- This training is intended for professionals concerned with the Ethereum Development preparing for the CBDE certification Ethereum exam.

What You Will Learn

- An excellent overall understanding of the Ethereum Exam architecture and Solidity language.
- All functional components (including smart contracts) required to develop an Ethereum Blockchain.
- The understanding of how to instantiate an Ethereum Course application on the network.
- An in-depth understanding of how transactions are created and implemented on an Ethereum Exam network.

Outline

Blockchain Ethereum Exam and Smart Contract Basics Blockchain Ethereum Certification

- What is Blockchain Ethereum Course and how does it work?
- Centralized vs. Decentralized vs. Distributed
- Blockchain vs. Databases
- Bitcoin vs Ethereum
- What are Smart Contracts?
- How are Smart Contracts used?

Smart Contract Programming Basics

- Advantages and Drawbacks of Smart Contracts

ETH - Ethereum Training

BILGINÇ IT Academy
info@bilginc.com
+44 20 3807 0925
+90 212 282 7700

Duration: **3 Day**

- Layer 1 vs. Layer 2
- High-Level Language vs. Low-Level
- Languages in Comparison Solidity, Vyper, others
- Smart Contracts with Solidity
- The Layout of a Solidity File
- LAB TASKS (Lab 1)
  - Types of Variables in Solidity
  - Function/Variable Visibility
  - Smart Contract Constructors
  - Setter- and Getter-Functions

Understanding Decentralized Information and Web3

- Blockchain Access structures and Architectures
  - Remote Blockchain Nodes vs. Local Blockchain Nodes
- Blockchain Access vs. centralized RESTful API
- Understanding Web3.js API
- Understanding Transactions and Consensus
- Private Keys, Public Keys and Signatures
- Understanding privacy on pubic Blockchains
- Understanding the architecture of KeyStore's such as MetaMask or MIST
- LAB TASKS (Lab 2 – Ropsten Test-Ether and MetaMask)
  - Installing and Configuring MetaMask
  - Obtaining Ropsten (or Testnet) Ether
  - Tracing Ether through Block-Explorers
  - Understanding Infura

Basics of Ethereum and the EVM

- Ethereum Denominations
- Understanding EVM and the ABI Interface
- Calls vs. Transactions
- Concurrency and Events
- Use cases of Events
- LAB TASKS (Lab 3 Web3JS Operations + Lab 4 Events)
  - Install and Use Ganache
  - Work with Web3.js
  - Work with Infura
  - Define Events
  - Listen and React to Events

Solidity Advanced: Modifiers, Mappings, Structs and Inheritance

- Understanding Functions, Mappings and Structs
- When to use Modifiers
- Libraries vs. Inheritance
- LAB TASKS (Lab 5 Modifiers, 6 Mappings and Structs, Lab 7 Inheritance)
  - Understand and use Modifiers
  - Add Mappings and Structs
  - Use Inheritance to increase auditability

Understanding Deployment and Costs Blockchain Ethereum Certification

ETH - Ethereum Training

BILGINC
IT Academy
info@bilginc.com
+44 20 3807 0925
+90 212 282 7700

Duration: **3 Day**

- Understand Development and Deployment Cycles
- Understanding Solidity Compilation and Deployment
- Gas and Gas-Costs
- Upgradeability and Data Migration Techniques
- Understand the moving Parts: Compiler, Blockchain, API, KeyStore
- LAB TASKS (Lab 8 Deployment, Lab 9 Tie it together)
    - Changing from strings to bytes (save gas)
    - Deploy using Ropsten Test-Ether
    - Understand the difference when using a real Blockchain
    - Use a fully functioning distributed Application

Mining, Proof of Work vs. Proof of Authority

- What is Mining in PoW?
    - How blocks are generated
    - PoW vs. PoA (vs. PoS)
- Understanding Go-Ethereum or Ganache/TestRPC for local development
- Understanding Private Blockchains vs. Public Blockchains
-
    - LAB TASKS (Lab 10 Mining)
    - Installing and using Ganache
    - Installing and using Go-Ethereum
    - Connecting to Ganache/Go-Ethereum from Remix and Web3.js
    - Interact with the Blockchain from HTML/JS

Current Problems, Solutions, Outlook, Serenity Blockchain Ethereum Certification

- Ethereum Now and Ethereum Future
- Where we are at with Ethereum
    - Homestead
    - PoW
- Where Ethereum is heading to
    - Serenity
    - PoS + PoW to PoS
    - Sharding
- Recommended Newsletters/Groups

Working in Teams, Testing and Versioning

- Understanding what Truffle is
- Comparison to Embark
- How Manage Code for Teams
- Understanding Migrations
- Understanding Unit-Testing with Truffle
- LAB TASKS (Lab 11 Truffle setup, Lab 12 Truffle Unit Testing)
    - Download and Setup Truffle
    - Adapt the standard Truffle-Project
    - Write A Unit Test

IPFS and distributed File-Storage

- What is IPFS
- Comparison between IPFS, FileCoin, Swarm, Sia, Storj
- LAB TASK (Lab 13 – IPFS)

BİLGİNÇ IT Academy

info@bilginc.com
+44 20 3807 0925
+90 212 282 7700

- Install and work with IPFS
- Upload and retrieve a fully decentralized file

Bonus Section: Compilation

- LAB TASK (Lab 14 – Compilation)
  - Compile a Solidity File with a command-line compiler

Labs:

Lab 0 – Understanding Components

- In this Lab we discuss the components used, where to get them and how to install them.

Lab 1 – Variables

- In this Lab you will get to know Remix and run simple Smart Contracts

Lab 2 – Ropsten and MetaMask

- In Lab you will get Ether from the Ropsten or Rinkeby Testnet and trace the Ethers on their way to the browser-plugin MetaMask.

Lab 3 – Web3.JS Operations

- During this Lab you will understand what Web3.JS is and how it is used to retrieve Information.

Lab 4 – Events

- Here you will learn a classic use-case for events to overcome concurrent workflow issues

Lab 5 – Modifiers

- In this Lab we will work with modifiers

Lab 6 – Mappings and Structs

- Mappings and Structs are powerful data-structures which you will learn in this Lab

Lab 5 – Modifiers

- In this Lab we will work with modifiers

Lab 7 – Inheritance

- When using Solidity it's good to keep things clear and small to be able to audit it. In this Lab we discuss Inheritance

Lab 8 – Gas-Costs and Deployment

- In this Lab you will learn how to optimize the smart contract, deploy the contract and what to pay attention to in practice.

Lab 9 – Final DApp, Tie it Together

- We put all the components together and work with the final distributed Application.

ETH - Ethereum Training

BİLGİNÇ
IT Academy
info@bilginc.com
+44 20 3807 0925
+90 212 282 7700

Duration: **3 Day**

Lab 10 – Mining

- During this lab you will see the difference of mining in a proof of authority consensus network and a proof or work network.

Lab 11 – Truffle Setup

- This Lab will teach how Truffle can be used to work in teams and with code locally. We will install and adapt the truffle default workflow.

Lab 12 – Truffle Unit Tests

- In this Lab we will do one of the most important steps during Contract Development: Create Unit-Tests for our Smart Contract

Lab 13 – IPFS

- During this Lab you will create a fully decentralized file-storage which will store files on IPFS and pointers on Ethereum

Bonus Lab – Solidity Compilation

- In this Lab you will learn how to manually compile a solidity file with the command line solidity compiler