AMBIENT-IT

+33 (0)1 48 10 09 46
formation@ambient-it.net
140 Avenue Jean Lolive 93500 Pantin - France

Updated on 01/07/2024

Sign up

# Solidity training

4 days (28 hours)

## Presentation

Solidity is an object-oriented programming language for writing Smart Contracts. It is used to implement smart contracts on various blockchain-based platforms such as Ethereum. It is based on ECMAScript syntax, making it a familiar choice for web developers.

Ethereum is a platform for the development of decentralized applications (dApps), based on Blockchain technology, with no risk of interruptions, fraud or intrusions. Thanks to Ethereum, it is now possible to program a whole range of applications where it is usually necessary to have a trusted third party (crowdfunding, voting, organizations, land registry...).

In this course, participants will learn how to write Smart Contracts using Solidity.

As with all our training courses, this one will introduce you to the latest version, Solidity 0.8.

## Objectives

- Understanding the fundamentals of smart contract development on the Ethereum blockchain
- Acquire programming skills in Solidity to create smart contracts
- Master essential tools and technologies such as Hardhat, OpenZeppelin and Ethers.js
- Learn to design and implement smart contracts for real-life use cases
- Acquire the skills needed to develop decentralized applications (DAPP) on Ethereum
- How to write unit tests to ensure the robustness and reliability of smart contracts
- Explore best security practices and deployment strategies on blockchain
- Gaining confidence to create innovative and functional blockchain solutions

## Target audience

- Developers
- Architects
- Design engineers

## Prerequisites

- At least 6 months' programming experience
- Knowledge of Javascript and/or React is a plus
- Test My Knowledge

## Technical requirements

- Node.js installed
- A code editor (such as Visual Studio Code)

## Further information

If you'd like to find out more about tomorrow's new challenges and the decentralization of today's web, our Web3 training course may be of interest to you.

## Solidity training program

### Day 1 - Introduction to blockchain and the basics of Smart Contracts

### Basic concepts of the Ethereum blockchain

- The network and the currency-Ethereum and Ether (ETH)
- The different types of Ethereum networks
- Complete, lightweight archive nodes
- Connecting to Ethereum networks
- Understanding the differences between ETH, WEI and GAZ
- Understanding and calculating transaction costs
- Blocks and the Ethereum blockchain
- Hash functions
- The 2 types of account on Ethereum
- The difference between a transaction and a message call
- The different consensus mechanisms on blockchain -PoW and PoS
- "The Merge and the Beacon channel

### Smart Contracts

- Smart contracts: what are they and how do they work?

- Anatomy and design of an intelligent contract
- Compiling, deploying and debugging smart contracts on Remix
- Setting up a local development environment with Hardhat
- Compiling, testing and deploying smart contracts with Hardhat
- Using Ganache for local deployment
- Project with exercises: develop, compile, debug and deploy your first intelligent contract on Remix and Hardhat

## DAY 2 - SOLIDITY

### The Solidity language

- Syntax and basics
- Advanced topics
- Examples and coding exercises for each theme
- Project with exercises: creating a voting contract using most of the topics discussed above

## DAY 3 - CREATING A DAPP - OPENZEPPELIN AND ERC20 TOKENS

### Decentralized applications - DApps

- Installing and configuring the Metamask portfolio
- Metamask RPC API
- Using the Ethers.js library - Provider, Signer, Contract and Utils
- Using Alchemy (third-party provider of Ethereum nodes)
- Project with exercises: creating a DApp using Ethers.js and the Metamask RPC API

### OpenZeppelin

- How to use OpenZeppelin's modular, reusable contracts
- Minimize risk with OpenZeppelin contracts
- The different types of OpenZeppelin contracts: tokens, access control, governance, security...

### ERC20 tokens

- Understanding the ERC20 standard
- Creation of an ERC20 token (implementing all the methods and events defined by the ERC20 standard)
- Deployment of the contract on the Goerli test network
- Testing methods and events using Ethers.js

## DAY 4 - UNIT TESTING AND THE NFT DAPP PROJECT

### Unit tests on smart contracts

- Using the Mocha framework with Chai to test smart contracts
- Use the Hardhat-Chai assertion library to evaluate reversed transactions, events, Ether balance changes, etc.
- Making testing more efficient with Hardhat Network Helpers

### NFT's

- Understanding NFT's and the ERC721 technical standard
- Difference between ERC721 and ERC1155
- Creating an ERC721 token using the ERC721URIStorage extension
- Use IPFS and Pinata to store image and NFT metadata
- Deployment of the contract on the Goerli test network and display of the NFT on OpenSea
- Creation of a DAPP (in React) to create NFT's with customized parameters
- Using the Metamask portfolio (via API RPC) with DAPP

### OPTIONAL CONTENT (+1 day)

### Re-entry attacks

- Smart contract protection against re-entry attacks
- Checks-Effects-Interaction" design model
- Creation of a vulnerable contract (Bank) and an attacking contract
- Simulation of a re-entry attack with a script using Ethers.js
- Vulnerable contract enhancement to prevent re-entry attacks

### Creation of a DAO - Decentralized Autonomous Organization

- Understanding how CAD works and how to create and execute proposals
- Creation of a DAO with various contracts from the OpenZeppelin library: ERC20Votes, TimelockController, Governor...
- Deploying DAO on the local Hardhat network
- Creation and execution of CAD proposals using Ethers.js

## Companies concerned

This course is aimed at both individuals and companies, large or small, wishing to train their teams in a new advanced computer technology, or to acquire specific business knowledge or modern methods.

## Positioning on entry to training

Positioning on entry to training complies with Qualiopi quality criteria. As soon as

On final registration, the learner receives a self-assessment questionnaire which enables us to assess his or her estimated level of proficiency in different types of technology, and his or her expectations and personal objectives for the forthcoming training course, within the limits imposed by the selected format. This questionnaire also enables us to anticipate any connection or security difficulties within the company (intra-company or virtual classroom) which could be problematic for the follow-up and smooth running of the training session.

## Teaching methods

Practical course: 60% Practical, 40% Theory. Training material distributed in digital format to all participants.

## Organization

The course alternates theoretical input from the trainer, supported by examples, with brainstorming sessions and group work.

## Validation

At the end of the session, a multiple-choice questionnaire verifies the correct acquisition of skills.

## Sanction

A certificate will be issued to each trainee who completes the course.