**CODELANDCS BLOCKCHAIN DEVELOPMENT SYLLABUS**

**WEEK 6**

**DAY 2**

**INTRODUCTION TO HARDHAT**

Good day everyone. Today, we will be discussing the Hardhat framework, a popular and robust development environment for Ethereum smart contract development. It was **developed by Nomic Labs** and released in **2019**. In this lecture, we will explore the features and capabilities of the Hardhat framework, its architecture, and some of its use cases.

**What is Hardhat Framework?**

Hardhat is an open-source development environment for building and testing smart contracts on the Ethereum blockchain. It is a **node.js-based framework** that provides a set of tools for developing, testing, and deploying smart contracts. Hardhat is **designed to simplify the smart contract development process** by automating repetitive tasks such as compiling, testing, and deployment, while also providing a flexible and extensible architectur**e.**

**Features of Hardhat Framework**

**Smart Contract Compilation:** Hardhat provides **a built-in Solidity compiler** that compiles smart contracts to EVM bytecode. This feature allows developers to easily test and deploy smart contracts on the Ethereum blockchain.

**Testing:** Hardhat supports automated testing using the Mocha testing framework. It also provides a **built-in test runner** that enables developers to write and execute unit tests, integration tests, and end-to-end tests for their smart contracts.

**Debugging:** Hardhat provides a **built-in debugger** that enables developers to debug their smart contracts using the Truffle debugger.

**Gas Estimation:** Hardhat provides a built-in gas estimation tool that helps developers estimate the amount of gas required for executing their smart contracts on the Ethereum blockchain.

**Deployment:** Hardhat enables developers to deploy their smart contracts to various networks such as the mainnet, test networks, and private networks.

**Plugins:** Hardhat has a modular architecture that enables developers to extend its functionality through plugins. This feature allows developers to add custom features and functionalities to the framework.

**Architecture of Hardhat Framework**

Hardhat has a modular architecture that comprises several components. These components include the **Hardhat core**, plugins, and tasks. The Hardhat core is responsible for managing the overall framework functionality, while the plugins provide additional features and functionalities to the framework. The tasks are scripts that automate repetitive tasks such as compiling, testing, and deployment.

**Use Cases of Hardhat Framework**

**Decentralized Applications:** Hardhat is a popular choice for building decentralized applications (dApps) on the Ethereum blockchain. It provides a suite of tools for developing and testing smart contracts, making it easier for developers to build dApps.

**Prototyping:** Hardhat is an excellent tool for prototyping smart contracts. Its built-in Solidity compiler and automated testing framework allow developers to quickly iterate and test their smart contracts.

**Education:** Hardhat is a useful tool for teaching smart contract development. Its modular architecture and extensive documentation make it easy for developers to learn and experiment with smart contracts.

**Conclusion:**

Hardhat is a powerful development environment for building, testing, and deploying smart contracts on the Ethereum blockchain. It provides a suite of tools for smart contract development, making it easier for developers to build decentralized applications. Its modular architecture and extensible design make it a popular choice for smart contract development.

**HARDHAT FOLDER STRUCTURE**

The folder structure and files of any development environment are crucial to understand as they contain the core components of the framework.

**Folder Structure of Hardhat Framework:**

The folder structure of Hardhat follows a standard pattern that is common in most node.js-based projects. Here is an overview of the folder structure:

**Contracts:** This folder contains the Solidity smart contract files. All smart contracts that are created will be stored in this folder.

**Tests:** This folder contains the test files for the smart contracts. Hardhat uses the Mocha testing framework, and all test files must be stored in this folder.

**Artifacts:** This folder contains the compiled output of the smart contracts. When a smart contract is compiled using Hardhat, the compiled output is stored in this folder.

**Cache:** This folder contains the build cache used by Hardhat. It stores all intermediate files generated during the compilation process.

**Scripts:** This folder contains the deployment scripts for deploying the smart contracts to the Ethereum network.

**node\_modules:** This folder contains all the dependencies of the Hardhat framework.

**.env:** This file contains the environment variables for the Hardhat framework.

**hardhat.config.js:** This file contains the configuration for the Hardhat framework. It is where developers can define the networks they want to deploy their smart contracts to, and other configuration settings.

**Files in Hardhat Framework:**

Here are some of the important files in the Hardhat framework:

**package.json:** This file contains the metadata for the Hardhat project, including its name, version, dependencies, and other project-related information.

**hardhat.config.js:** This file contains the configuration for the Hardhat framework, including the network configurations, plugin configurations, and other settings.

**.env:** This file contains environment variables used by the Hardhat framework.

**.gitignore:** This file specifies files and folders that should be ignored by Git when the project is being version controlled.

**README.md:** This file contains the documentation for the Hardhat project.

**LICENSE:** This file contains the license agreement for the Hardhat project.

**WRITING AND DEPLOYING SMART CONTRACT WITH HARDHAT**

Writing and deploying smart contracts is one of the key functionalities of Hardhat, and we will explore the process in detail.

**Writing Smart Contracts with Hardhat:**

To write smart contracts with Hardhat, you will need to create a new file in the **contracts folder**. You can use any text editor or IDE of your choice to write the smart contracts. **Hardhat supports Solidity, Vyper,** and **other Ethereum-compatible languages.** Once you have written the smart contract code, you can compile it using the **npx hardhat compile command.** This command will **generate an output file in the artifacts folder**, which contains the **bytecode** and **ABI** of the smart contract.

**Deploying Smart Contracts with Hardhat:**

To deploy smart contracts with Hardhat, you will need to **create a deployment script in the scripts folder**. The deployment script is a JavaScript file that uses the Hardhat API to deploy the smart contracts to the Ethereum network. You can use the following steps to deploy smart contracts with Hardhat:

**Step 1:** Define the network configuration in the hardhat.config.js file. This file contains the network settings that you want to deploy your smart contracts to. You can define multiple networks in this file.

**Step 2:** Write a deployment script in the scripts folder. The deployment script should use the Hardhat API to deploy the smart contracts to the Ethereum network. The **script should also specify the contract constructor arguments and the gas price**.

**Step 3:** Run the deployment script using the **npx hardhat run** command. This command will run the deployment script and deploy the smart contracts to the specified network.

Testing Smart Contracts with Hardhat:

Testing smart contracts is an **important part of the development process.** Hardhat **provides a built-in testing framework** that allows you to write and run tests for your smart contracts. You can use the tests folder to write your test scripts. Hardhat uses **Mocha** and **Chai** for testing and provides a range of testing utilities and helpers.

**Conclusion:**

In conclusion, writing and deploying smart contracts with Hardhat is a straightforward process. Hardhat provides a comprehensive set of tools and features that make it easy to develop, test, and deploy smart contracts on the Ethereum network.