Building Smart Contracts with Solidity & Foundry

GDG Algiers

A heartfelt thank you to **GDG Algiers** for hosting this workshop and providing a platform for us to explore and learn together. Your commitment to fostering a collaborative learning environment makes initiatives like this possible, and I’m grateful to be part of a community that values growth, innovation, and knowledge sharing.

GLP

Who Am I ?

Lyes Boudjabout

- IT Co-Manager @GDG Algiers
- Software Engineering Student @ESI
- Smart Contracts Developer

GLP

Summary

GLP

Introduction

- In 2009, **Bitcoin** proved **we don’t need banks** to **move money** — it gave us **digital currency** that works anywhere, anytime.
- But then came a bigger question: what if we could **program money** itself? What if we could **build applications** — marketplaces, organizations, even games — that **run automatically**, **without servers or middlemen** ?
- That’s exactly what **Ethereum** introduced: not just digital money, but a **world computer** where **anyone** can deploy **unstoppable code**.
- That code is called a **Smart Contract**.

GLP

Terminology

- **Blockchain** → A digital database or ledger that is distributed among the nodes of a peer-to-peer network
- **Ethereum** → A programmable blockchain that extends Bitcoin's idea with smart contracts.
- **Smart Contract**→ A smart contract is immutable code on the blockchain — once deployed, it cannot be changed, so it must be correct and secure from the start.

What You’ll Walk Away With

- 🚀 **Smart Contracts** → Code that runs on Ethereum, powering different decentralized applications.
- 🖋️ **Solidity Basics** → Writing, compiling, and understanding contract structure.
- 🛠️ **Foundry Toolkit** → Using Forge, Cast, and Anvil for development & testing.
- 🔍 **Testing & Debugging** → Unit tests, fuzzing, and invariant checks for secure contracts.
- 🔒 **Best Practices** → Writing efficient, secure, and gas-optimized code.
- 💡 **Hands-On Experience** → You wrote, tested, and deployed your own smart contract.

GLP

Tools & Requirements

- Programming Concepts Fundamentals.
- Basic Blockchain Concepts:
  - What is Ethereum?
  - What is a transaction?
  - The idea of gas fees.
- Visual Studio Code & Remix IDE.
- Linux (any distribution), MacOS, Or WSL For Windows.

GLP

Let's Dive in !

GLP

Solidity Programming Language

- **What it is** → The primary programming language for writing smart contracts on Ethereum.
- **Purpose** → Defines rules & logic that run on the Ethereum Virtual Machine (EVM).
- **Features:**
  - High-level, contract-oriented language.
  - Syntax inspired by JavaScript, Python, and C++.
  - Supports inheritance, libraries, and user-defined types.
- **Why it matters** → Enables developers to build decentralized apps (dApps) that are transparent, trustless, and unstoppable.

GLP

Coding Section 1

Foundry Framework

- **What it is** → A blazing-fast, Rust-based toolkit for Ethereum smart contract development.
- **Purpose** → Simplifies writing, testing, and deploying Solidity contracts.
- **Why it matters** → Developer-friendly, highly performant, and widely adopted in professional auditing and DeFi projects.
- **Other Ethereum Dev Frameworks:**
  - **Hardhat** → JavaScript-based, plugin ecosystem, debugging & scripting.
  - **Truffle** → Early popular framework, now legacy in many projects.
  - **Brownie** → Python-based framework, integrates well with Vyper.
  - **Embark** → Full-stack dApp framework with deployment automation.

Foundry Framework

Core Components:

⚡ **Forge** → Build, test, fuzz, deploy contracts.
💻 **Cast** → CLI for interacting with Ethereum (send tx, call functions).
🔌 **Anvil** → Local Ethereum node for testing & simulation.
🪓 **Chisel** → Interactive Solidity REPL for rapid prototyping and debugging.

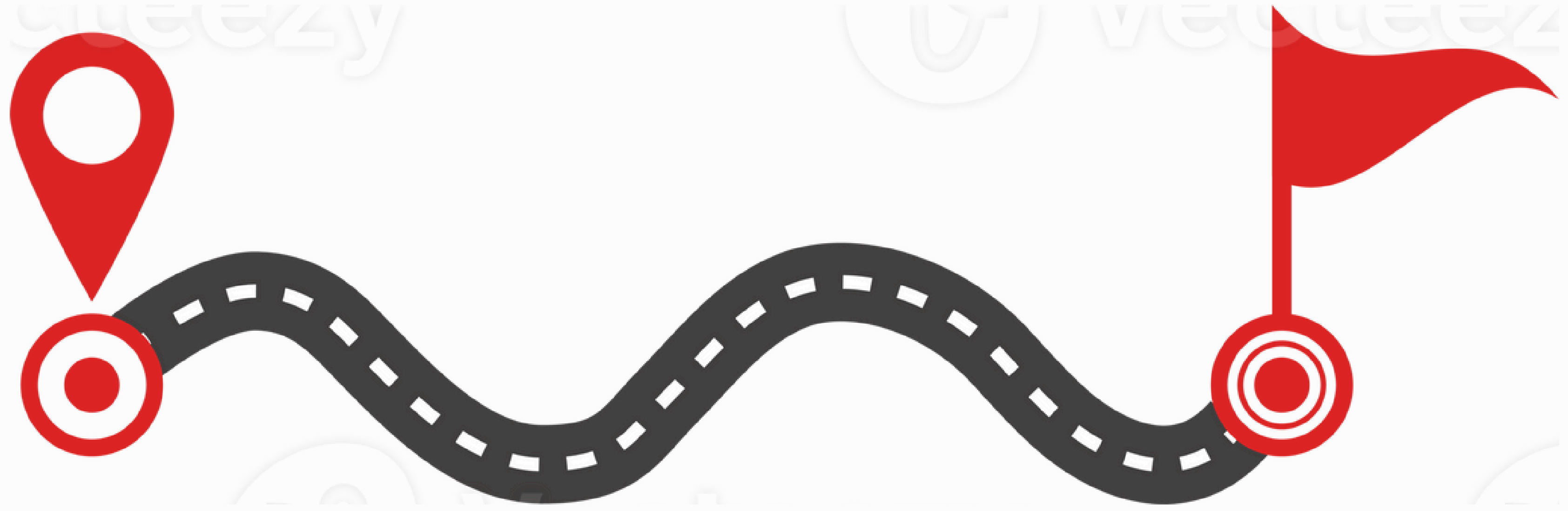
Coding Section 2

Conclusion

- ✅ You've learned the basics of Solidity and how to write smart contracts.
- ✅ You've explored Foundry for testing, deployment, and debugging.
- ✅ You now understand how Ethereum's programmability makes it more than just digital money.

👉 **Next Steps:**

- Experiment with your own contracts.
- Explore fuzzing & invariant testing.
- Contribute to open-source smart contract projects.



GLP

Notice & Contact

For questions or issues related to the content, please contact:

Email: nl_boudjabout@esi.dz

Github: @Lyes-Boudjabout

LinkedIn: Lyes Boudjabout