FRONTIER · TAUGHT WITH ETHEREUM

Blockchain / Web3

Solidity, the EVM, and the audit mindset. You build and test contracts on a local chain, then spend a month learning to break them, because deployed code stays deployed.

Foundational | 5 phases | 23 sessions | 35 contact hours

Syllabus

5 phases, 23 sessions. Each phase ends in something you have built.

1 The Machine Under the Contract

Explain what an Ethereum transaction is and what the network actually guarantees.

4 sessions
$58 alone

1.1 What a Chain Is For

Replication, trust assumptions, finality, and the short list of problems this actually fits.

FREE PREVIEW

1.2 Keys, Accounts, and Signatures

Hashing, keypairs, addresses, and what a wallet is really holding on your behalf.

1.3 Transactions, Gas, and the EVM

Nonces, calldata, state changes, reverts, and gas as a budget you can run out of.

1.4 Your First Contract

Remix with nothing installed: a pragma, a state variable, a deploy, and a call you can watch.

By the end: **A contract deployed on a local chain, plus a written trace of the transaction that deployed it.**

2 Solidity as a Language

Write contract code from scratch even if you have never programmed before.

5 sessions
$58 alone

2.1	**Types, Variables, and Functions**	uint, address, bool, bytes, visibility keywords, arguments, and returning a value.
2.2	**Control Flow and Collections**	Conditions, loops, arrays, mappings, and why an unbounded loop is a bug on this machine.
2.3	**Where Data Lives**	Storage, memory, and calldata, what each one costs, and why the difference is not cosmetic.
2.4	**Failing on Purpose**	require, custom errors, revert, and msg.sender as the only proof of identity you get.
2.5	**Contracts Calling Contracts**	Interfaces, external calls, events, and the fact that nothing you store is private.

By the end: **A contract on a local chain that keeps per-address records, emits an event on every change, and refuses every invalid call.**

3 A Project Someone Else Could Check Out

Move off the browser IDE into a version-controlled repo with tests that prove behavior.

5 sessions
$58 alone

3.1	**Foundry From Zero**	forge, cast, and anvil, a pinned compiler in foundry.toml, and Git Bash or WSL if you are on Windows.
3.2	**Tests Before Trust**	forge test, assertions, setUp, and vm.expectRevert for the calls that are supposed to fail.
3.3	**Fuzzing and Invariants**	Property tests over random input, bounded assumptions, and reading the counterexample forge hands back.
3.4	**Access Control Done Right**	Owners, roles, modifiers, two-step transfers, and the OpenZeppelin code you should not rewrite.
3.5	**Standards You Inherit**	ERC-20 and ERC-721 as interfaces, inheritance, and what an audited library already settled.

By the end: **A Foundry repo with a role-controlled contract and a test for every caller and input that must be refused.**

4 Off Your Laptop

Put a contract on a public testnet and drive it from a page a stranger can open.

4 sessions
$58 alone

4.1 Testnets in 2026
Sepolia, Hoodi, the retirement of Holesky, and faucets gated on a mainnet balance you do not have.

4.2 Deploying for Real
forge script, encrypted keystores, source verification, and the private key that must never reach Git.

4.3 A Front End That Talks to a Contract
viem and wagmi: ABIs, reads, writes, receipts, and a wallet prompt the user can actually read.

4.4 What an L2 Actually Is
Rollups, sequencers, the data posted to L1, L2BEAT stages, and the same contract deployed twice.

By the end: **A verified contract on Sepolia and a web front end that reads its state and sends one signed transaction.**

5 Assume Someone Is Trying to Take It

Find, prove, and fix real vulnerabilities, then write them up the way an auditor does.

5 sessions
$58 alone

5.1 The Audit Mindset
Assets, actors, trust boundaries, and the invariants you write down before you read any code.

5.2 Access Control, the Top Category
Missing checks, wrong roles, initializers left open, and the functions nobody guarded, exploited then patched.

5.3 External Calls and Reentrancy
Checks-effects-interactions, transient reentrancy guards, and return values nobody checked.

5.4 Oracles, Rounding, and Logic
Manipulated prices, integer division, and the business-logic bugs no scanner will ever flag.

5.5 Tools and Their Limits
Slither, Aderyn, forge lint, long fuzz campaigns, and why a clean report is not an audit.

By the end: **A security review of a seeded protocol: findings by severity, a proof-of-concept test each, patches, and residual risk.**

Tools you will use

Solidity 0.8.36 | Foundry | Remix | OpenZeppelin Contracts 5.x | viem | wagmi | Slither | Aderyn

What you will build

1. **Attestation Registry**
   Roles, events, and a test for every caller who must be refused
2. **Non-Transferable Credential**
   An ERC-721 that cannot be moved, with a front end that reads it
3. **Published Audit**
   Findings by severity, a proof-of-concept test each, and the patch

Registration

Phase 1: The Machine Under the Contract	$58
Phase 2: Solidity as a Language	$58
Phase 3: A Project Someone Else Could Check Out	$58
Phase 4: Off Your Laptop	$58
Phase 5: Assume Someone Is Trying to Take It	$58
Whole track, all 5 phases	**$240**

Buying every phase separately costs $290, so the whole track saves $50.

No payment is taken online. Registering creates an order and payment instructions are sent with it. Access opens once payment is confirmed.