Roadmap to Learn Web3 Development with Ethereum

Phase 1: Web2 Foundations (Optional if already known) - (1 Week)

- **HTML/CSS/JavaScript**
    - Learn semantic HTML and responsive CSS (Flexbox, Grid)
    - Understand JavaScript basics (ES6+, DOM manipulation, fetch API)
- **Frontend Frameworks**
    - React.js (Components, State, Props, Hooks)
- **Basic Backend Concepts**
    - REST APIs, Express.js, MongoDB basics

Phase 2: Blockchain & Ethereum Basics - (2 to 3 Days)

- **Blockchain Fundamentals**
    - What is a blockchain, how it works (consensus, nodes, blocks, transactions)
    - Difference between public/private blockchains
- **Ethereum Concepts**
    - Accounts: EOA vs Contract accounts
    - Gas, Transactions, Nonces, Blocks
    - Ethereum Virtual Machine (EVM)
- **Smart Contracts Overview**
    - What are smart contracts and how they run on Ethereum

Resources:

-
- "Ethereum.org Developer Portal"
- "Mastering Ethereum" by Andreas M. Antonopoulos

Phase 3: Smart Contract Development - (5 Days to 1 Week)

- **Solidity Basics**
    - Syntax, data types, functions, modifiers
    - Events, constructors, inheritance
    - Error handling, require/assert/revert
- **Advanced Solidity**
    - Mappings, structs, arrays
    - Visibility, memory vs storage
    - Security considerations (re-entrancy, overflows, tx.origin vs msg.sender)

Tools:

- Remix IDE (for quick prototyping)
- Hardhat or Foundry (for local development and testing)
- Solidity Tutorial for Beginners - Full Course in 4 Hours (2023) (skip the web3.js part)

Phase 4: Web3 Frontend Integration - (3 to 4 days)

- **Web3 Libraries**
    - web3.js or ethers.js (ethers preferred)
    - Connecting to wallets (MetaMask, WalletConnect)
- **Reading/Writing Smart Contract Data**
    - Using ABI to interact with contracts
    - Calling view/pure functions vs sending transactions

Build Projects:

- Simple DApp (e.g., Counter or Todo List)
- Token Creator (ERC-20/ERC-721)

Phase 5: Token Standards (1 to 2 Days)

- **ERC Standards**
    - ERC-20: Fungible Tokens
    - ERC-721: NFTs
    - ERC-1155: Multi-token standard

Build Projects:

- Custom ERC-20 token with frontend UI
- NFT minting site

Phase 6: Testing and Deployment (2 Days)

- **Tools**
    - Write tests (Chai, Mocha, Waffle)
    - Deploy contracts on testnets (Goerli, Sepolia)
    - Use tools like Alchemy, Infura, Tenderly
- **Layer 2 Solutions**
    - Polygon, Avalanche

Phase 7: Contribute & Build Real-World Apps

- **Participate in Hackathons (ETHGlobal, Encode, etc.)**
- **Contribute to Open Source (look at GitHub issues, bounties)**
- **Build a Portfolio (host your DApps, write case studies)**

Suggested Tools & Platforms

- **Development**: Hardhat, Foundry, Remix
- **Wallets**: MetaMask, Rainbow
- **Infra**: Infura, Alchemy, Etherscan API
- **Storage**: IPFS, Filecoin, Arweave
- **Learning**: Cryptozombies, Speedrun Ethereum, Buildspace, LearnWeb3 DAO

Final Note: Exploring Solana and the Anchor Framework

Once you're comfortable with Ethereum development, exploring Solana can give you perspective on different blockchain architectures (e.g., proof-of-history, parallel transaction processing). Solana development is done in Rust and often uses the **Anchor framework**, which provides:

- Simplified smart contract development
- Better error handling and type safety
- Built-in support for common patterns like accounts and instructions

If you're looking to expand your skills beyond EVM-based chains, learning Anchor and Solana is highly recommended. Start with:

- Solana Cookbook
- Anchor Book
- Buildspace's Solana courses

It complements your Ethereum knowledge and prepares you for multichain development. This skill is essential for working on blockchain-based inter-IIT problem statements, as they often involve blockchain development on Solana using Rust.

Stay consistent and build as you learn. The ecosystem evolves rapidly, so stay active on platforms like Twitter, Discord, and GitHub to keep up!