Suggested Resources

Monad is fully EVM bytecode-compatible, with all supported opcodes and precompiles as of theShanghai fork . Monad also preserves the standard Ethereum JSON-RPC interfaces.

As such, most development resources for Ethereum Mainnet apply to development on Monad.

This page suggests aminimal set of resources for getting started with building a decentralized app for Ethereum. Child pages provide additional detail or options.

AsSolidity is the most popular language for Ethereum smart contracts, the resources on this page focus on Solidity; alternatively see resources onVyper orHuff . Note that since smart contracts are composable, contracts originally written in one language can still make calls to contracts in another language.

IDEs

- Remix
- is an interactive Solidity IDE. It is the easiest and fastest way to get started coding and compiling Solidity smart contracts without the need for additional tool installations.
- VSCode
- +Solidity extension
-

Basic Solidity

- CryptoZombies
- is a great end-to-end introduction to building dApps on the EVM. It provides resources and lessons for anyone from someone who has never coded before, to experienced developers in other disciplines looking to explore blockchain development.
- Solidity by Example
- introduces concepts progressively through simple examples; best for developers who already have basic experience with other languages.
-

Intermediate Solidity

- The Solidity Language
- official documentation is an end-to-end description of Smart Contracts and blockchain basics centered on EVM environments. In addition to Solidity Language documentation, it covers the basics of compiling your code for deployment on an EVM as well as the basic components relevant to deploying a Smart Contract on an EVM.
- Solidity Patterns
- repository provides a library of code templates and explanation of their usage.
- TheUniswap V2
- contract is a professional yet easy to digest smart contract that provides a great overview of an in-production Solidity dApp. A guided walkthrough of the contract can be foundhere
- .
- Cookbook.dev
- provides a set of interactive example template contracts with live editing, one-click deploy, and an AI chat integration to help with code questions.
- OpenZeppelin
- provides customizable template contract library for common Ethereum token deployments such as ERC20, ERC712, and ERC1155. Note, they are not gas optimized.
-

Advanced Solidity

- TheSolmate repository
- andSolady repository
- provide gas-optimized contracts utilizing Solidity or Yul.
- Yul
- is a intermediate language for Solidity that can generally be thought of as inline assembly for the EVM. It is not quite pure assembly, providing control flow constructs and abstracting away the inner working of the stack while still exposing the raw memory backend to developers. Yul is targeted at developers needing exposure to the EVM's raw memory backend to build high performance gas optimized EVM code.
- Huff
- is most closely described as EVM assembly. Unlike Yul, Huff does not provide control flow constructs or abstract away the inner working of the program stack. Only the most upmost performance sensitive applications take advantage of Huff, however it is a great educational tool to learn how the EVM interprets instructions its lowest level.

-

Local nodes

Developers often find it helpful to be able to run a 1-node Ethereum network with modified parameters to test interaction with the blockchain:

- Anvil
- is a local Ethereum node packaged in the Foundry toolkit
- Hardhat Network
- is a local Ethereum node packaged in the Hardhat toolkit
-

Installation is most easily done as part of installing the respective toolkit, described in the next section.

Toolkits

Developers often find it helpful to build their project in the context of a broader framework that organizes external dependencies (i.e. package management), organizes unit and integration tests, defines a deployment procedure (against local nodes, testnet, and mainnet), records gas costs, etc.

Here are the two most popular toolkits for Solidity development:

- Foundry
- is a Solidity framework for both development and testing. Foundry manages your dependencies, compiles your project, runs tests, deploys, and lets you interact with the chain from the command-line and via Solidity scripts. Foundry users typically write their smart contracts and tests in the Solidity language.
- Hardhat
- is a Solidity development framework paired with a JavaScript testing framework. It allows for similar functionality as Foundry, and was the dominant toolchain for EVM developers prior to Foundry.
-

Interacting with the Ethereum RPC API

Frontends for dapps typically use JavaScript or Python to submit read or write queries to an RPC node. This code is typically referred to as the "client side", as web developers can roughly equate the blockchain to a backend server.

A few libraries provide standard methods for submitting queries or transactions to an RPC node:

- Python:
-
    - web3.py
- *
- Javascript:
-
    - web3.js
-
    - ethers.js
-
    - To this endWeb3.js
-
    - andWeb3.py
-
    - , Java Script and Python libraries respectively, have developed to make interacting with blockchains more intuitive for web developers.
- *
-

Here is a quick example for creating a frontend:create-eth-app .

Testnets

Monad Testnet will be available for developers in the coming months, however, being bytecode and RPC compatible with the EVM means developers wishing to deploy on Monad can preliminary useEthereum's Testnets .

Further details

Child pages add additional resources:

- EVM behavior

- Further Solidity resources
- Debugging on-chain
- Other languages
-
    - Vyper
-
    - Huff
- *
-

Previous Developing on Monad Next EVM behavior Last updated5 months ago

On this page * IDEs * Basic Solidity * Intermediate Solidity * Advanced Solidity * Local nodes * Toolkits * Interacting with the Ethereum RPC API * Testnets * Further details