# 5CFE9D;}

.css-kun0x7{fill:transparent;opacity:0.5;margin:0 0.2rem;}.css-kun0x7:hover{fill:#FAF40A;}

.css-1ix0nx7{fill:transparent;opacity:0.5;}.css-1ix0nx7:hover{fill:#F14544;} On this page

# Web3 Development Basics

## Introduction

Developing dApps and interacting with Smart Contracts is quite different from Web2, and at times challenging due to little information on this topic compared to other areas of software development.

This developer guide is a quick overview of the space, including references to libraries and guides that are great starting points. Reading this guide should help you identify areas that you might need to learn a bit more about and prepare you for the following Uniswap-specific guides.

It is assumed that you know the basics about Ethereum and the blockchain, including some terminology. If you already know how to build dApps and interact with ethersJS etc. you can safely skip this guide.

## RPCs

The access point to the blockchain are RPC nodes. They are thestandardized interface to read data from smart contracts, send transactions and interact with on-chain protocols.

RPCs are eitherfull or archival nodes with a (JSON-RPC ) interface.

To support Ethereum's decentralization, one can host a node themselves, for example by using one of the implementations listed below:

- geth
- 
    - The original (reference) implementation of the Ethereum protocol
- erigon
- 
    - A very efficient archival node implementation
- Nethermind
- 
    - An Ethereum implementation focused on stability

As achieving high availability and making sure your node is synced all the time turns out to be quite challenging, there are nodes as a service (RPC) providers that you can use, especially in production environments. When choosing an RPC provider, we suggest you look for an RPC service that supports websockets as they provide far superior performance than HTTP connections. To ensure interoperability, you should also ensure that your RPC provider adheres strictly to the JSON-RPC standard and doesn't require custom requests.

Chainnodes is a robust RPC provider with generous free tier that you can use in both development and production environments. For testing purposes you could also use a free public RPC endpoint, for example fromChainlist .

### JSON-RPC Standard

RPCs communicate over theJSON-RPC standard. To send requests, you take the RPC URL and make a POST request with a JSON body. See the below example:

POST https://mainnet.chainnodes.org/API_KEY

Body:

{ "jsonrpc" :

"2.0" , "method" :

"eth_blockNumber" , "params" :

[ ] , "id" :

1 } Copy At the time of writing, this request would respond with the following:

{ "id" :

1 , "result" :

"0x11527c0" , "jsonrpc" :

"2.0" } Copy Examining the result, we see that the result is an encoded hex string. After decoding it, we see it returns the current blocknumber of our network,18163648 .

You can find the above examples, including more, in this Postman collection .

To check out all possible RPC requests, head over to the Chainnodes Docs .

### Client implementations

As communicating over HTTP with POST requests directly can be quite difficult, especially if you want to properly encode and decode responses and handle failures and exponential backoffs, there are client implementations that can do the heavy lifting for you. These SDKs have developer friendly APIs and internally handle creating the proper RPC requests, sending them to the endpoint you choose and decoding the response for you.

Some of the major implementations are listed below:

- ethers.js
  - Javascript/Typescript SDK for NodeJS and the Browser. Used throughout the Uniswap Docs.
- wagmi
- and viem
  - Javascript/Typescript, great duo for modern Web3 development in the Browser.
- web3js
  - Javascript/Typescript SDK for NodeJS and the Browser by ChainSafe.

Web3 development is not limited to JS. Web3 libraries for various languages include:

- Web3.swift
  - Swift SDK for iOS apps and Backends.
- KEthereum
  - Kotlin SDK for Android development.
- ethers-rs
  - Rust SDK.
- ethclient
  - Go SDK, part of geth, the reference Ethereum node implementation.

At the moment, Uniswap only offers Typescript sdks.

As you can see there are lots of SDKs to make it easier to communicate via RPC with the blockchain.

### Local Development

To simulate RPCs and transactions locally, you can check this guide .

## Indexers

As RPCs are only a slim abstraction of the data stored in the blockchain, there are certain things that are hard or expensive to access with regular RPC requests.

A common example are transactions of a specific Wallet address. Imagine you want to get a list of all transactions that originated from (or to) a specific Wallet. One could think there should be an RPC method calledeth_getTransactionsForWallet or something similar. But due to the nature of how the data is stored, this RPC method is not feasible, and hence not implemented.

Now, instead of accessing those kind of things by iterating through every block in the blockchain, you can use Indexers, that are designed to index data like that on the go and provide easy access to it.

### TheGraph

A well-known, standardized implementation of indexers isTheGraph . It is used by most major protocols, including Uniswap, to index data and make it accessible to users and dashboards.

Using TheGraph, you can either access opensubgraphs via the GraphQL querying language, orcreate your own and deploy it.

Throughout the docs you will see how to interact with the Uniswap subgraph to fetch tick data and more without ever touching RPCs.

An important note about Indexers though:

While they can be helpful, you need to be aware that the ultimate source of truth comes from the RPCs. As reorgs happen and certain issues on indexers arise, there might be certain datapoints that are either not fully up-to-date or even completely wrong on Indexers. If you have a use-case that requires perfectly correct data all the time, use RPCs directly. If you are just doing data visualization or dashboards, use Indexers if they fit your use-case.

Some popular subgraphs that you can try to fetch data from the blockchain easily (click on playground to give it a try):

- Uniswap Messari subgraph
- Snapshot Subgraph
- Aave V3 Messari subgraph

## Smart Contract Development

Smart contracts are typically developed using theSolidity language .

There are VSCode plugins that make the development with Solidity easier. One of them is Juan Blanco's "Solidity" that you can find overhere .

You would typically also go for a developer suite with testing, compilation and deployment capabilities:

- Foundry
-
    - Fast and modern developer tools for smart contract engineers.
- Hardhat
-
    - Scriptable, easy and battle-tested JS developer suite for smart contracts.

To read data from smart contracts, or interact with them, use theclient SDKs mentioned above. They help you generate the necessary RPC calls to fetch data from / send transactions to the blockchain and interact with your smart contracts. You can read more about it in theethers.js docs .

## Blockchain Explorers

Manually gathering information and inspecting data stored in a blockchain is a tedious task. Almost all chains have at least one accompanying block explorer to help visualize addresses, transactions, contracts and more.

For Ethereum mainnet, we suggest usingEtherscan . You can use Etherscan to inspectcontracts , transactions, blocks, and much more.

If you are looking to debug a transaction,Tenderly can also be a very helpful tool. In this example of a failed transaction on the Uniswap V2 Router you can easily see why the execution failed and where.

## The Uniswap development suite

Uniswap offers several SDKs that work together and enable you to easily interact with the Uniswap protocol The most important SDKs are:

- sdk-core
- : The core of the Uniswap SDKs, defines classes and types shared across all the SDKs
- v2-sdk
- : An SDK to interact with the Uniswap V2 protocol.
- v3-sdk
- : An SDK to interact with the Uniswap V3 protocol.
- router-sdk
- : Provides abstractions to interact with the (older) SwapRouter contracts.
- universal-router-sdk
- : Abstracts interactions with the Universal Router.
- smart-order-router

- : Searches for the most efficient routes for a trade.
- permit2-sdk
- : Simplifies interactions with Permit2 in JS.
- uniswapx-sdk
- : SDK for the UniswapX protocol.

As you know already, all interactions with the blockchain happens through RPCs. So the Uniswap SDKs, as you will see throughout the guides, requires you to have access to an RPC endpoint likeChainnodes . When reading data, the data is read from the given RPC endpoint. When actually swapping, you will need to sign a transaction using a private key.

We are continuously working on improving the Uniswap development suite, so stay tuned for more updates.

## Next Steps

Go through the basic guides first and try to fetch some data and interact with the Uniswap ecosystem a little bit. You can evensend your first swap transaction on a local fork!

While some concepts in Web3 require thinking outside of the box, this guide should give you a good overview on where to start. You should now be fully equipped to follow our other guides. Edit this page .css-1tclyyl{margin-top:1.5rem;} .css-1c3fvx8{display:-webkit-box;display:-webkit-flex;display:-ms-flexbox;display:flex;-webkit-flex-direction:row;-ms-flex-direction:row;flex-direction:row;-webkit-align-items:center;-webkit-box-align:center;-ms-flex-align:center;align-items:center;-webkit-box-pack:center;-ms-flex-pack:center;-webkit-justify-content:center;justify-content:center;} .css-1wsnqg4{font-size:1rem;padding-right:0.5rem;} Helpful? .css-y2jwfw{fill:transparent;opacity:0.5;}.css-y2jwfw:hover{fill:#5CFE9D;}

.css-kun0x7{fill:transparent;opacity:0.5;margin:0 0.2rem;}.css-kun0x7:hover{fill:#FAF40A;}

.css-1ix0nx7{fill:transparent;opacity:0.5;}.css-1ix0nx7:hover{fill:#F14544;} Previous Local Development Next Getting a Quote * Introduction * RPCs * * JSON-RPC Standard * * Client implementations * * Local Development * Indexers * * TheGraph * Smart Contract Development * Blockchain Explorers * The Uniswap development suite * Next Steps