Basics overview

This first chapter of the crossword puzzle tutorial will introduce fundamental concepts to smart contract development in a beginner-friendly way. By the end of this chapter you'll have a proof-of-concept contract that can be interacted with viaNEAR CLI and a simple frontend that uses thenear-api-js library .

It's not as bad as you think

Rust is a serious systems programming language. There are pointers, lifetimes, macros, and other things that may look foreign. Don't worry if this is how you feel:

Art byksart.near

The good news is the Rust SDK takes care of a lot of the heavy lifting.

We'll also have the compiler on our side, often telling us exactly what went wrong and offering suggestions. As we go through this tutorial, you'll begin to see patterns that we'll use over and over again.

So don't worry, writing smart contracts in Rust on NEAR doesn't require a heavy engineering background.

Assumptions for this first chapter

- There will be only one crossword puzzle with one solution.
- The user solving the crossword puzzle will not be able to know the solution.
- Only the author of the crossword puzzle smart contract can set the solution.

Completed project

Here's the final code for this chapter:

https://github.com/near-examples/crossword-tutorial-chapter-1

How it works

We'll have a rule about how to get the words in the proper order. We collect words in ascending order by number, and if there's and across and a down for a number, the across goes first.

So in the image above, the solution will benear nomicon ref finance .

Let's begin! Edit this page Last updatedonOct 10, 2022 byBenKurrek Was this page helpful? Yes No

Previous Lazy Minting, Collections, and More! Next Set up Rust and a contract skeleton