The Move Book

Comments are a way to add notes or document your code. They are ignored by the compiler and don't result in Move bytecode. You can use comments to explain what your code does, add notes to yourself or other developers, temporarily remove a part of your code, or generate documentation. There are three types of comments in Move: line comments, block comments, and doc comments.

You can use a double slash // to comment out the rest of the line. Everything after // will be ignored by the compiler.

Block comments are used to comment out a block of code. They start with / *and end with* / . Everything between / *and* / will be ignored by the compiler. You can use block comments to comment out a single line or multiple lines. You can even use them to comment out a part of a line.

This example is a bit extreme, but it shows all the ways that you can use block comments.

Documentation comments are special comments that are used to generate documentation for your code. They are similar to block comments but start with three slashes /// and are placed before the definition of the item they document.

Unlike some languages, whitespace (spaces, tabs, and newlines) have no impact on the meaning of the program.

Line comment

You can use a double slash // to comment out the rest of the line. Everything after // will be ignored by the compiler.

```bash module book::comments_line;

// let's add a note to everything! fun some_function_with_numbers() { let a = 10; // let b = 10 this line is commented and won't be executed let b = 5; // here comment is placed after code a + b; // result is 15, not 10! } ```

Block comments are used to comment out a block of code. They start with / *and end with* / . Everything between / *and* / will be ignored by the compiler. You can use block comments to comment out a single line or multiple lines. You can even use them to comment out a part of a line.

```bash module book::comments_block;

fun / *you can comment everywhere* / go_wild() { / *here there everywhere* / let a = 10; let b = / *even here* / 10; / *and again* / a + b; } /* you can use it to remove certain expressions or definitions fun empty_commented_out() {

} */ ```

This example is a bit extreme, but it shows all the ways that you can use block comments.

Documentation comments are special comments that are used to generate documentation for your code. They are similar to block comments but start with three slashes /// and are placed before the definition of the item they document.

```bash /// Module has documentation! module book::comments_doc;

/// This is a 0x0 address constant! const AN_ADDRESS: address = @0x0;

/// This is a struct! public struct AStruct { /// This is a field of a struct! a_field: u8, }

/// This function does something! /// And it's documented! fun do_something() {} ```

Unlike some languages, whitespace (spaces, tabs, and newlines) have no impact on the meaning of the program.

## Block comment

Block comments are used to comment out a block of code. They start with / *and end with* / . Everything between / *and* / will be ignored by the compiler. You can use block comments to comment out a single line or multiple lines. You can even use them to comment out a part of a line.

```bash module book::comments_block;

fun / *you can comment everywhere* / go_wild() { / *here there everywhere* / let a = 10; let b = / *even here* / 10; / *and again* / a + b; } /* you can use it to remove certain expressions or definitions fun empty_commented_out() {

} */ ```

This example is a bit extreme, but it shows all the ways that you can use block comments.

Documentation comments are special comments that are used to generate documentation for your code. They are similar to block comments but start with three slashes /// and are placed before the definition of the item they document.

```bash /// Module has documentation! module book::comments_doc;

/// This is a 0x0 address constant! const AN_ADDRESS: address = @0x0;

/// This is a struct! public struct AStruct { /// This is a field of a struct! a_field: u8, }

/// This function does something! /// And it's documented! fun do_something() {} ```

Unlike some languages, whitespace (spaces, tabs, and newlines) have no impact on the meaning of the program.

## Doc comment

Documentation comments are special comments that are used to generate documentation for your code. They are similar to block comments but start with three slashes /// and are placed before the definition of the item they document.

```bash /// Module has documentation! module book::comments_doc;

/// This is a 0x0 address constant! const AN_ADDRESS: address = @0x0;

/// This is a struct! public struct AStruct { /// This is a field of a struct! a_field: u8, }

/// This function does something! /// And it's documented! fun do_something() {} ```

Unlike some languages, whitespace (spaces, tabs, and newlines) have no impact on the meaning of the program.

## Whitespace

Unlike some languages, whitespace (spaces, tabs, and newlines) have no impact on the meaning of the program.