The Move Book

Constants are immutable values that are defined at the module level. They often serve as a way to give names to static values that are used throughout a module. For example, if there's a default price for a product, you might define a constant for it. Constants are stored in the module's bytecode, and each time they are used, the value is copied.

Constants must start with a capital letter - this is enforced at the compiler level. For constants used as a value, the convention is to use all uppercase letters and underscores between words, which makes constants stand out from other identifiers in the code. An exception is made for error constants , which are written in ECamelCase.

Constants can't be changed and assigned new values. As part of the package bytecode, they are inherently immutable.

A common use case for an application is to define a set of constants that are used throughout the codebase. But due to constants being private to the module, they can't be accessed from other modules. One way to solve this is to define a "config" module that exports the constants.

This way other modules can import and read the constants, and the update process is simplified. If the constants need to be changed, only the config module needs to be updated during the package upgrade.

Naming Convention

Constants must start with a capital letter - this is enforced at the compiler level. For constants used as a value, the convention is to use all uppercase letters and underscores between words, which makes constants stand out from other identifiers in the code. An exception is made for error constants , which are written in ECamelCase.

```bash /// Price of the item used at the shop. const ITEM_PRICE: u64 = 100;

/// Error constant. const EItemNotFound: u64 = 1; ```

Constants can't be changed and assigned new values. As part of the package bytecode, they are inherently immutable.

```bash module book::immutable_constants;

const ITEM_PRICE: u64 = 100;

// emits an error fun change_price() { ITEM_PRICE = 200; } ```

A common use case for an application is to define a set of constants that are used throughout the codebase. But due to constants being private to the module, they can't be accessed from other modules. One way to solve this is to define a "config" module that exports the constants.

```bash module book::config;

const ITEM_PRICE: u64 = 100; const TAX_RATE: u64 = 10; const SHIPPING_COST: u64 = 5;

/// Returns the price of an item. public fun item_price(): u64 { ITEM_PRICE } /// Returns the tax rate. public fun tax_rate(): u64 { TAX_RATE } /// Returns the shipping cost. public fun shipping_cost(): u64 { SHIPPING_COST } ```

This way other modules can import and read the constants, and the update process is simplified. If the constants need to be changed, only the config module needs to be updated during the package upgrade.

## Constants are Immutable

Constants can't be changed and assigned new values. As part of the package bytecode, they are inherently immutable.

```bash module book::immutable_constants;

const ITEM_PRICE: u64 = 100;

// emits an error fun change_price() { ITEM_PRICE = 200; } ```

A common use case for an application is to define a set of constants that are used throughout the codebase. But due to constants being private to the module, they can't be accessed from other modules. One way to solve this is to define a "config" module that

exports the constants.

```bash module book::config;

const ITEM_PRICE: u64 = 100; const TAX_RATE: u64 = 10; const SHIPPING_COST: u64 = 5;

/// Returns the price of an item. public fun item_price(): u64 { ITEM_PRICE } /// Returns the tax rate. public fun tax_rate(): u64 { TAX_RATE } /// Returns the shipping cost. public fun shipping_cost(): u64 { SHIPPING_COST } ```

This way other modules can import and read the constants, and the update process is simplified. If the constants need to be changed, only the config module needs to be updated during the package upgrade.

## Using Config Pattern

A common use case for an application is to define a set of constants that are used throughout the codebase. But due to constants being private to the module, they can't be accessed from other modules. One way to solve this is to define a "config" module that exports the constants.

```bash module book::config;

const ITEM_PRICE: u64 = 100; const TAX_RATE: u64 = 10; const SHIPPING_COST: u64 = 5;

/// Returns the price of an item. public fun item_price(): u64 { ITEM_PRICE } /// Returns the tax rate. public fun tax_rate(): u64 { TAX_RATE } /// Returns the shipping cost. public fun shipping_cost(): u64 { SHIPPING_COST } ```

This way other modules can import and read the constants, and the update process is simplified. If the constants need to be changed, only the config module needs to be updated during the package upgrade.

## Links