The Move Book

In Move, the copy ability on a type indicates that the instance or the value of the type can be copied, or duplicated. While this behavior is provided by default when working with numbers or other primitive types, it is not the default for custom types. Move is designed to express digital assets and resources, and controlling the ability to duplicate resources is a key principle of the resource model. However, the Move type system allows you to add the copy ability to custom types:

In the example above, we define a custom type Copyable with the copy ability. This means that instances of Copyable can be copied, both implicitly and explicitly.

In the example above, a is copied to b implicitly, and then explicitly copied to c using the dereference operator. If Copyable did not have the copy ability, the code would not compile, and the Move compiler would raise an error.

Note: In Move, destructuring with empty brackets is often used to consume unused variables, especially for types without the drop ability. This prevents compiler errors from values going out of scope without explicit use. Also, Move requires the type name in destructuring (e.g., Copyable in let Copyable {} = a;) because it enforces strict typing and ownership rules.

The copy ability is closely related to the drop ability . If a type has the copy ability, it is very likely that it should have drop too. This is because the drop ability is required to clean up resources when the instance is no longer needed. If a type only has copy , managing its instances gets more complicated, as the instances must be explicitly used or consumed.

All of the primitive types in Move behave as if they have the copy and drop abilities. This means that they can be copied and dropped, and the Move compiler will handle the memory management for them.

All native types in Move have the copy ability. This includes:

All of the types defined in the standard library have the copy ability as well. This includes:

Copying and Drop

The copy ability is closely related to the drop ability . If a type has the copy ability, it is very likely that it should have drop too. This is because the drop ability is required to clean up resources when the instance is no longer needed. If a type only has copy , managing its instances gets more complicated, as the instances must be explicitly used or consumed.

```
bash public struct Value has copy, drop {}
```

All of the primitive types in Move behave as if they have the copy and drop abilities. This means that they can be copied and dropped, and the Move compiler will handle the memory management for them.

All native types in Move have the copy ability. This includes:

All of the types defined in the standard library have the copy ability as well. This includes:

Types with the

All native types in Move have the copy ability. This includes:

All of the types defined in the standard library have the copy ability as well. This includes:

Further reading