sui

Intro to **Sui Objects** and Creating Your First **NFT Project**

Twitter: @0xShayan

February 22, 2023

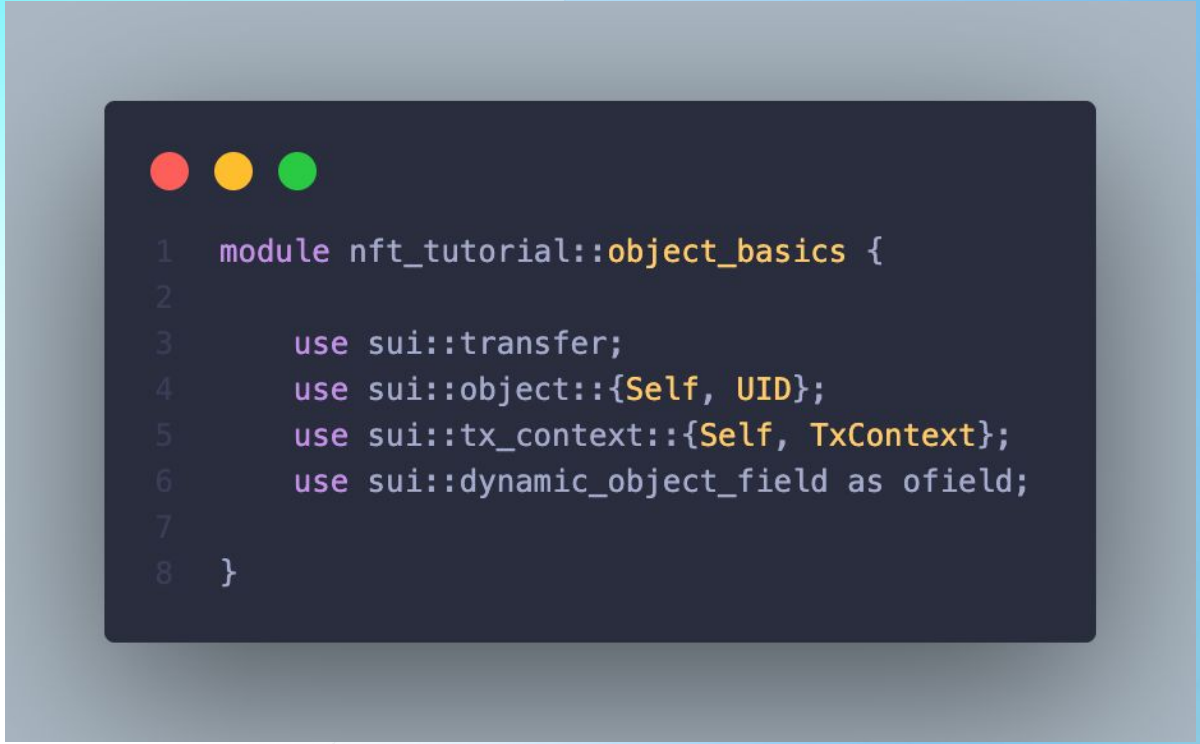
Agenda

sui

Sui Objects

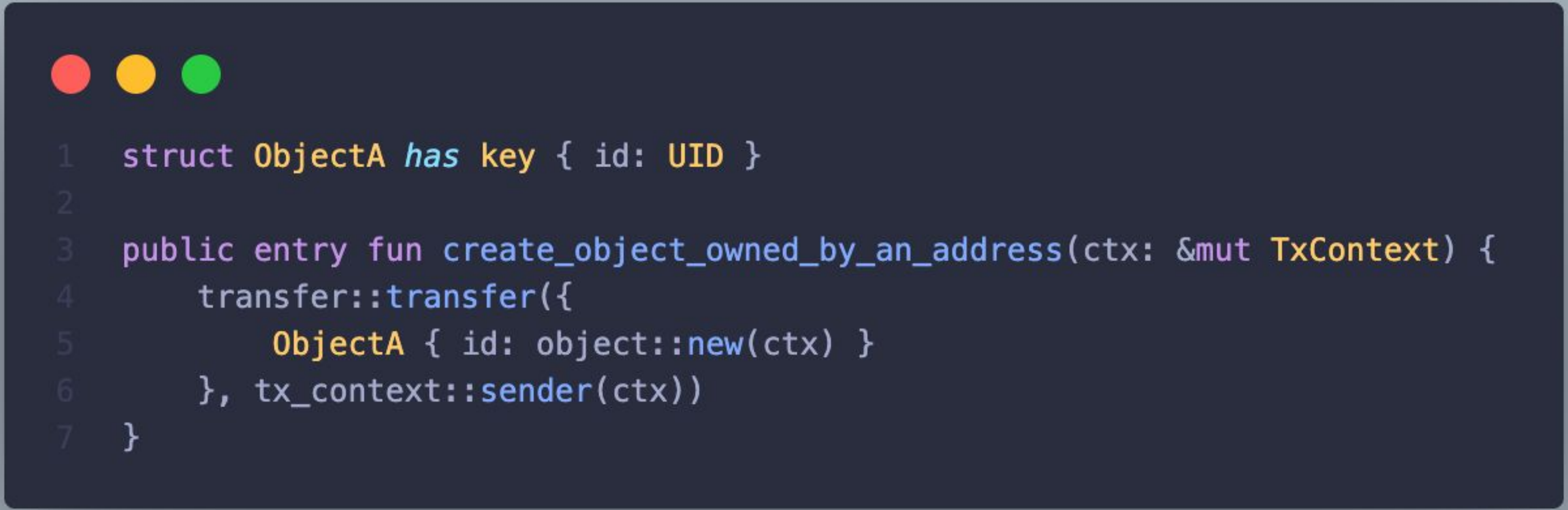
In a new terminal, enter:

```
sui move new nft_tutorial
```



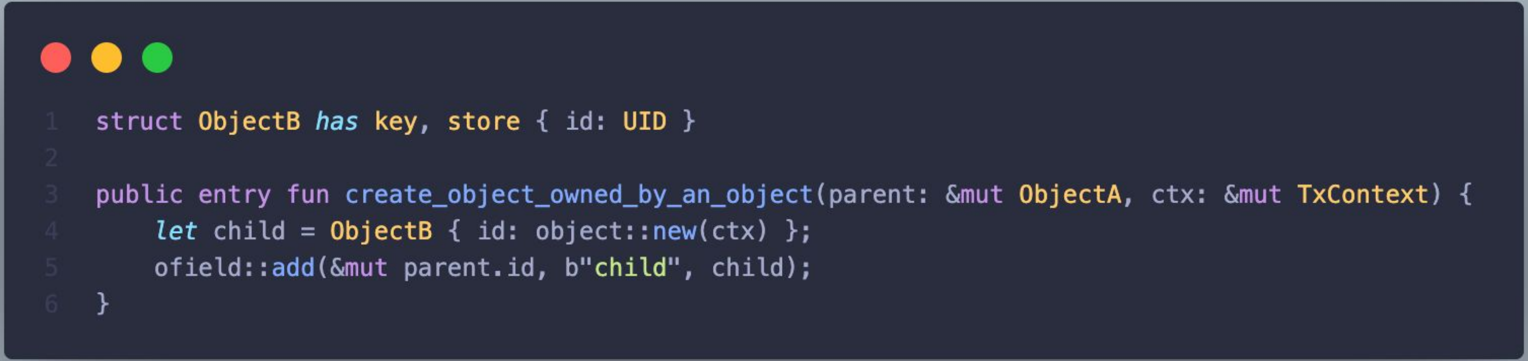
```
module nft_tutorial::object_basics {

    use sui::transfer;
    use sui::object::{Self, UID};
    use sui::tx_context::{Self, TxContext};
    use sui::dynamic_object_field as ofield;

}
```

sui



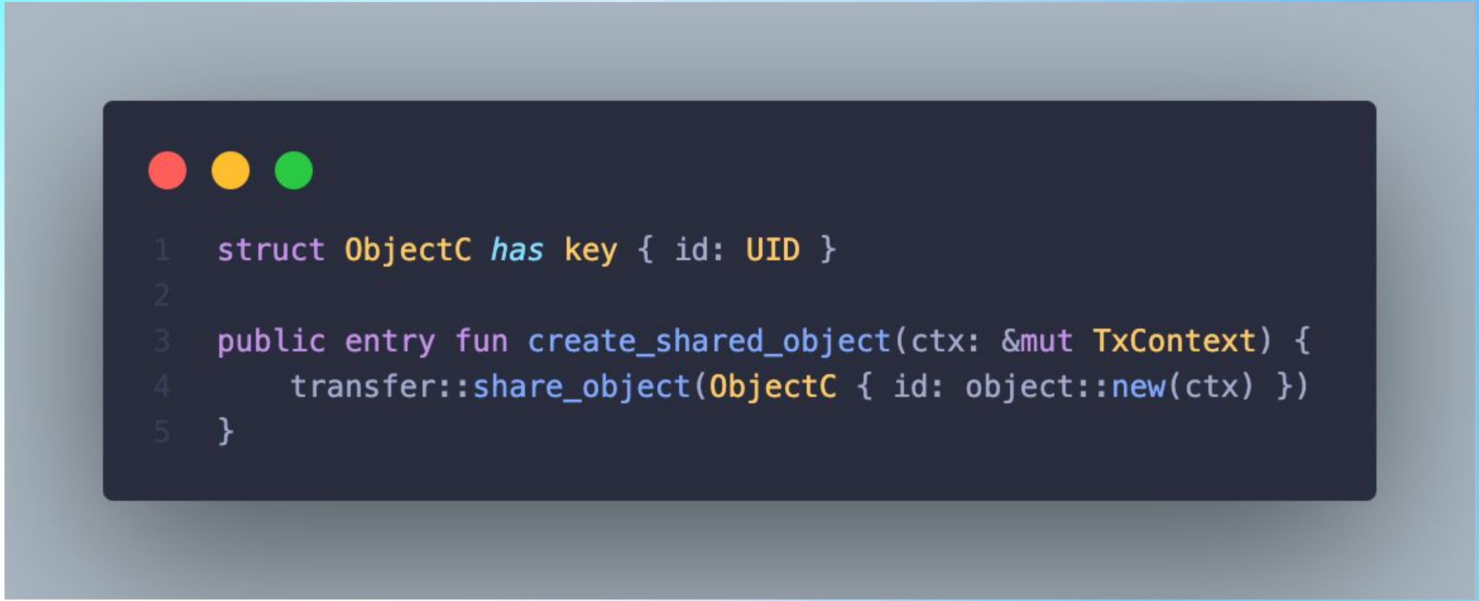
```
struct ObjectA has key { id: UID }

public entry fun create_object_owned_by_an_address(ctx: &mut TxContext) {
    transfer::transfer({
        ObjectA { id: object::new(ctx) }
    }, tx_context::sender(ctx))
}
```

sui



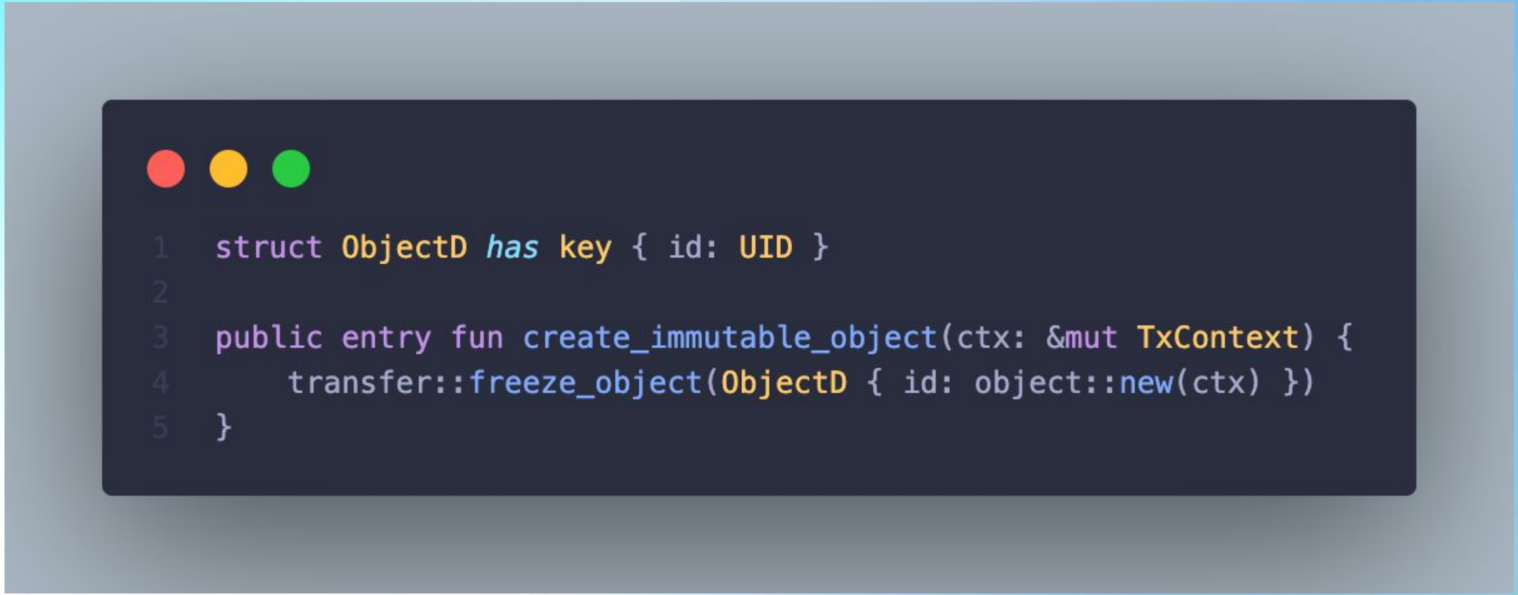
```
struct ObjectB has key, store { id: UID }

public entry fun create_object_owned_by_an_object(parent: &mut ObjectA, ctx: &mut TxContext) {
    let child = ObjectB { id: object::new(ctx) };
    ofield::add(&mut parent.id, b"child", child);
}
```

sui



```
struct ObjectC has key { id: UID }

public entry fun create_shared_object(ctx: &mut TxContext) {
    transfer::share_object(ObjectC { id: object::new(ctx) })
}
```

sui

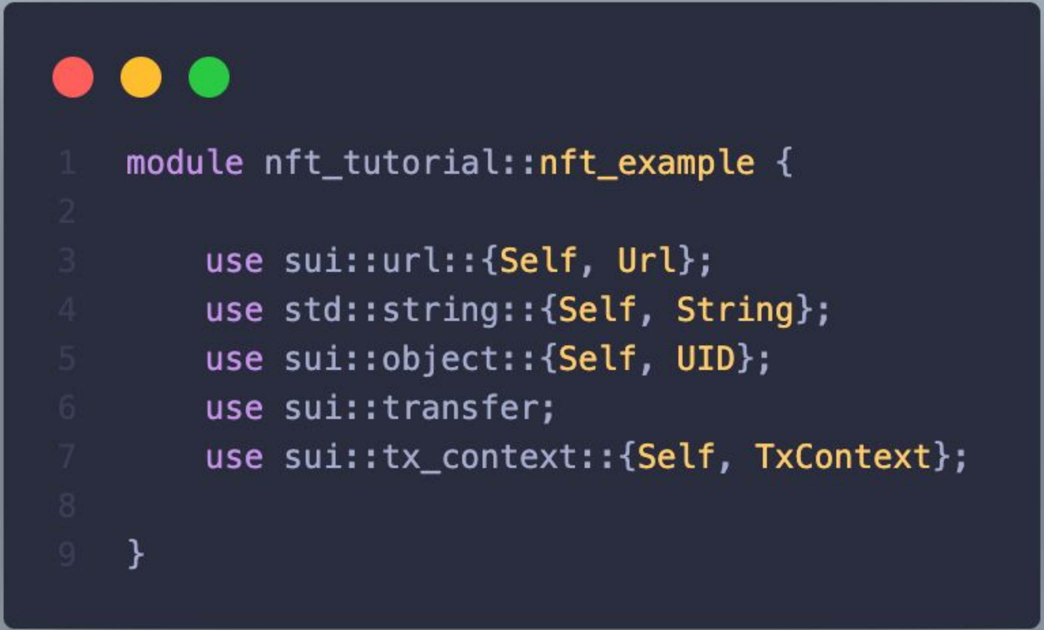


```
struct ObjectD has key { id: UID }

public entry fun create_immutable_object(ctx: &mut TxContext) {
    transfer::freeze_object(ObjectD { id: object::new(ctx) })
}
```

sui

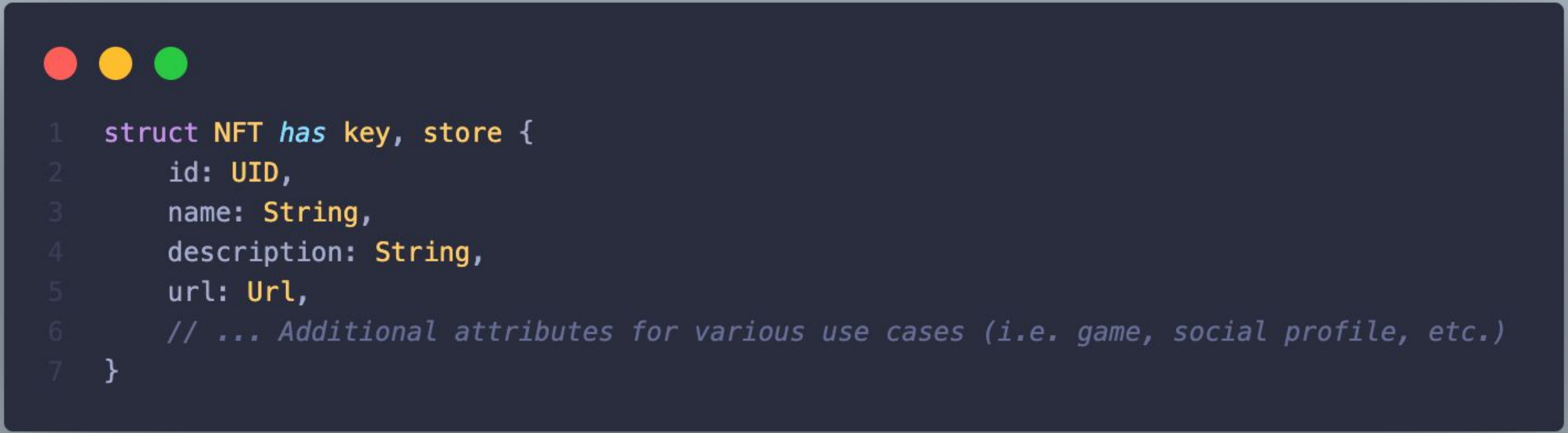
So What’s the Difference Between an **Object** and an **NFT**?

sui

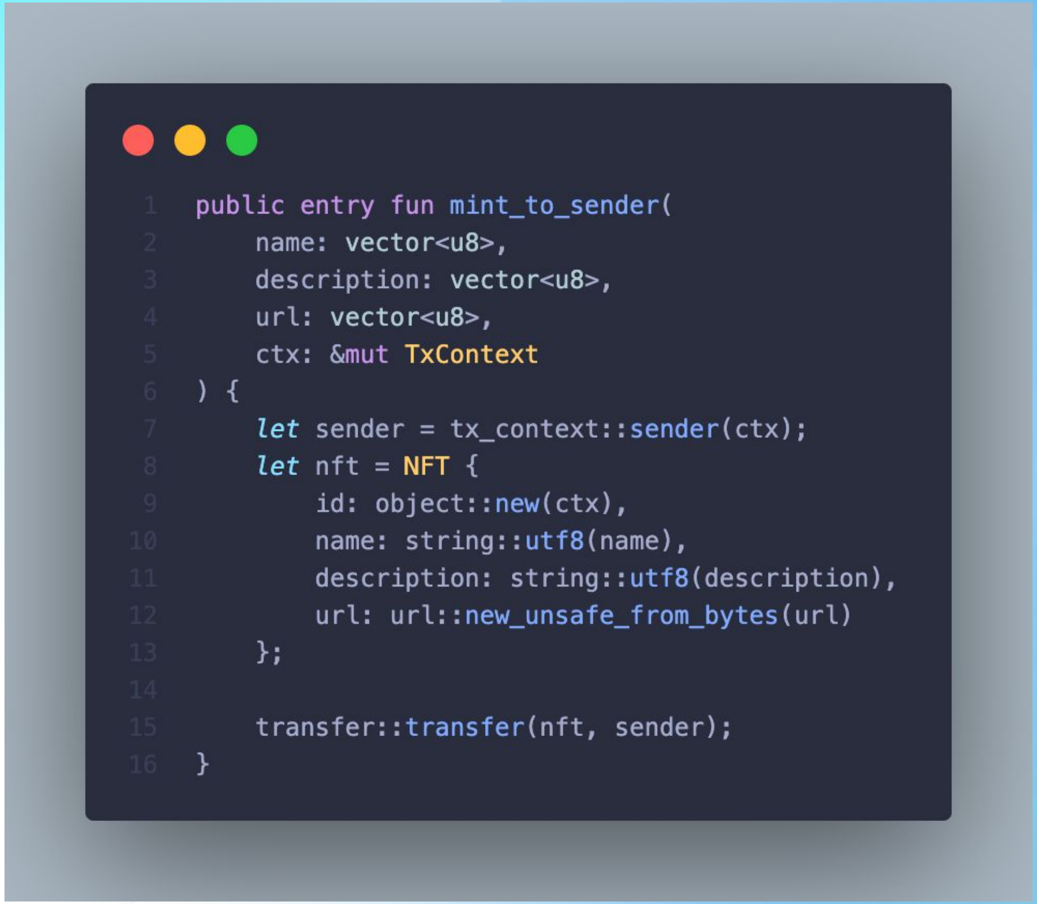


```
module nft_tutorial::nft_example {

    use sui::url::{Self, Url};
    use std::string::{Self, String};
    use sui::object::{Self, UID};
    use sui::transfer;
    use sui::tx_context::{Self, TxContext};

}
```

sui

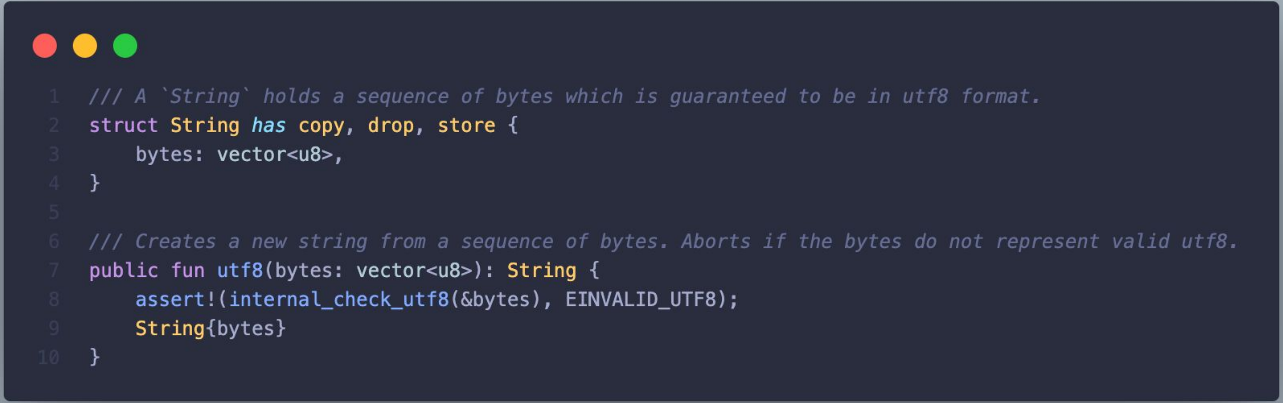


```
struct NFT has key, store {
    id: UID,
    name: String,
    description: String,
    url: Url,
    // ... Additional attributes for various use cases (i.e. game, social profile, etc.)
}
```

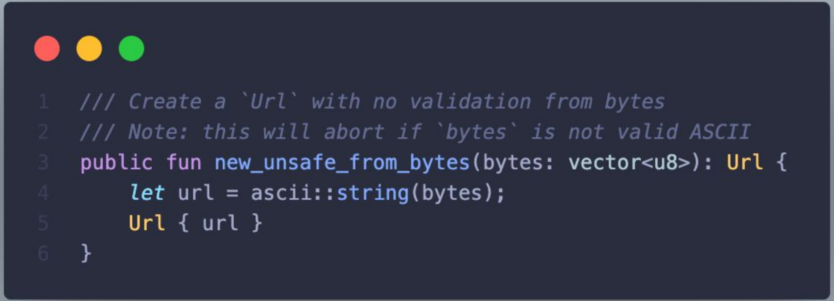


```
public entry fun mint_to_sender(
    name: vector<u8>,
    description: vector<u8>,
    url: vector<u8>,
    ctx: &mut TxContext
) {
    let sender = tx_context::sender(ctx);
    let nft = NFT {
        id: object::new(ctx),
        name: string::utf8(name),
        description: string::utf8(description),
        url: url::new_unsafe_from_bytes(url)
    };

    transfer::transfer(nft, sender);
}
```

sui

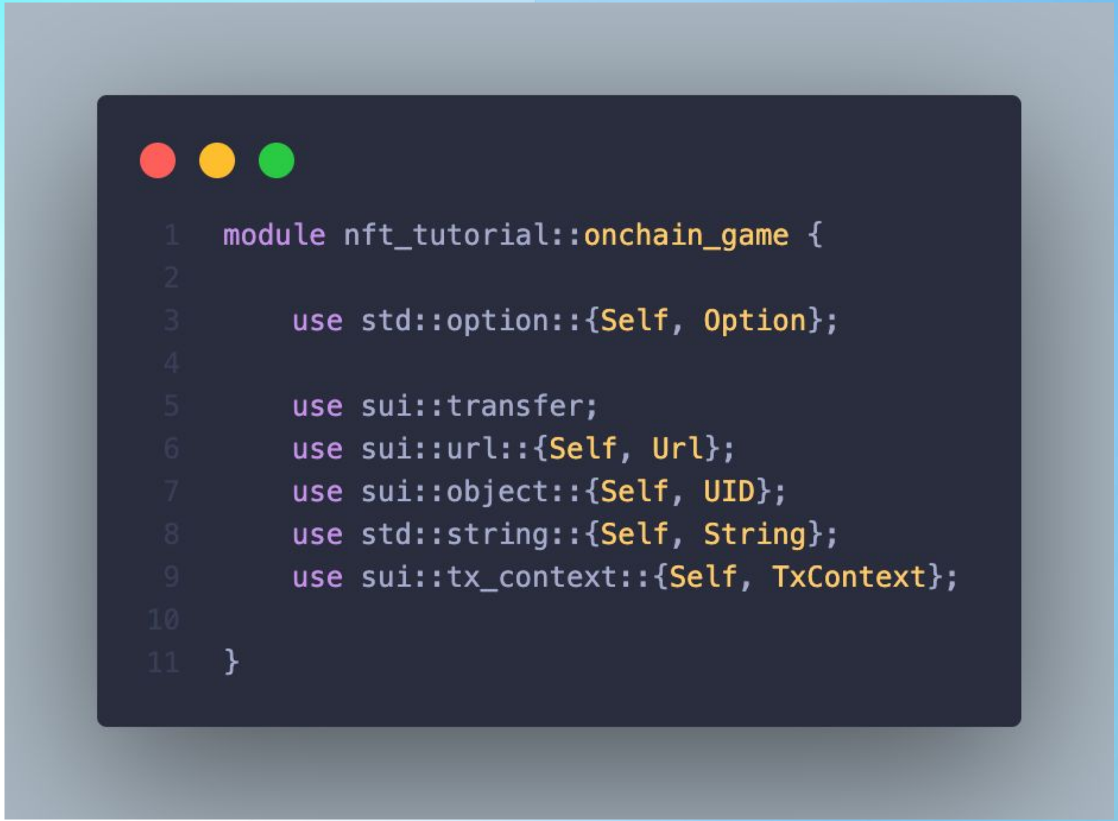


```
/// A `String` holds a sequence of bytes which is guaranteed to be in utf8 format.
struct String has copy, drop, store {
    bytes: vector<u8>,
}

/// Creates a new string from a sequence of bytes. Aborts if the bytes do not represent valid utf8.
public fun utf8(bytes: vector<u8>): String {
    assert!(internal_check_utf8(&bytes), EINVALID_UTF8);
    String{bytes}
}
```



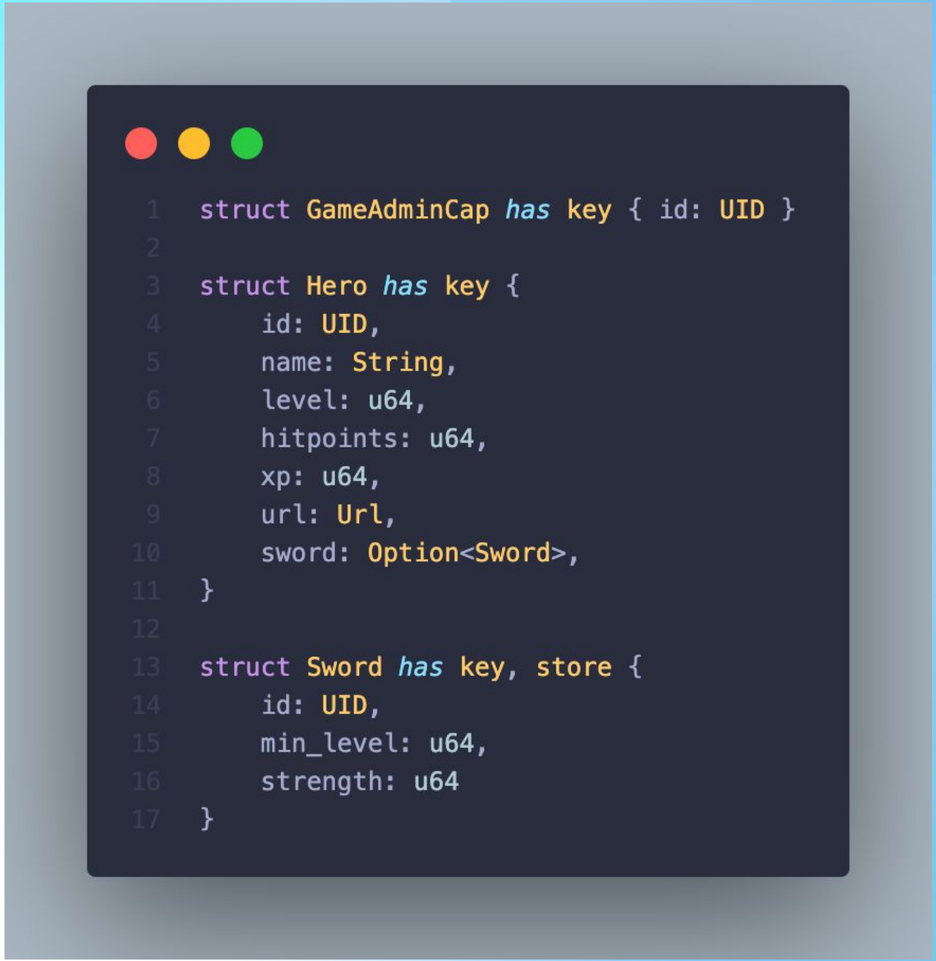
```
/// Create a `Url` with no validation from bytes
/// Note: this will abort if `bytes` is not valid ASCII
public fun new_unsafe_from_bytes(bytes: vector<u8>): Url {
    let url = ascii::string(bytes);
    Url { url }
}
```

sui

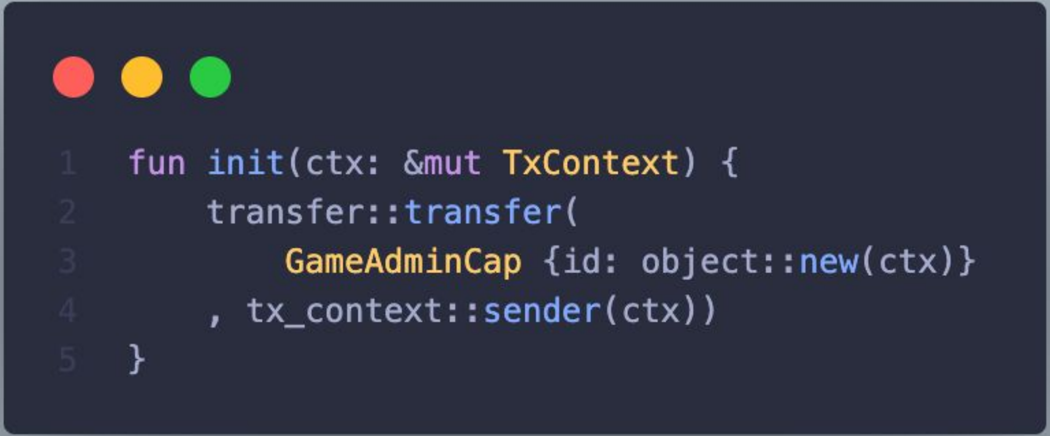


```
module nft_tutorial::onchain_game {

    use std::option::{Self, Option};

    use sui::transfer;
    use sui::url::{Self, Url};
    use sui::object::{Self, UID};
    use std::string::{Self, String};
    use sui::tx_context::{Self, TxContext};

}
```

sui

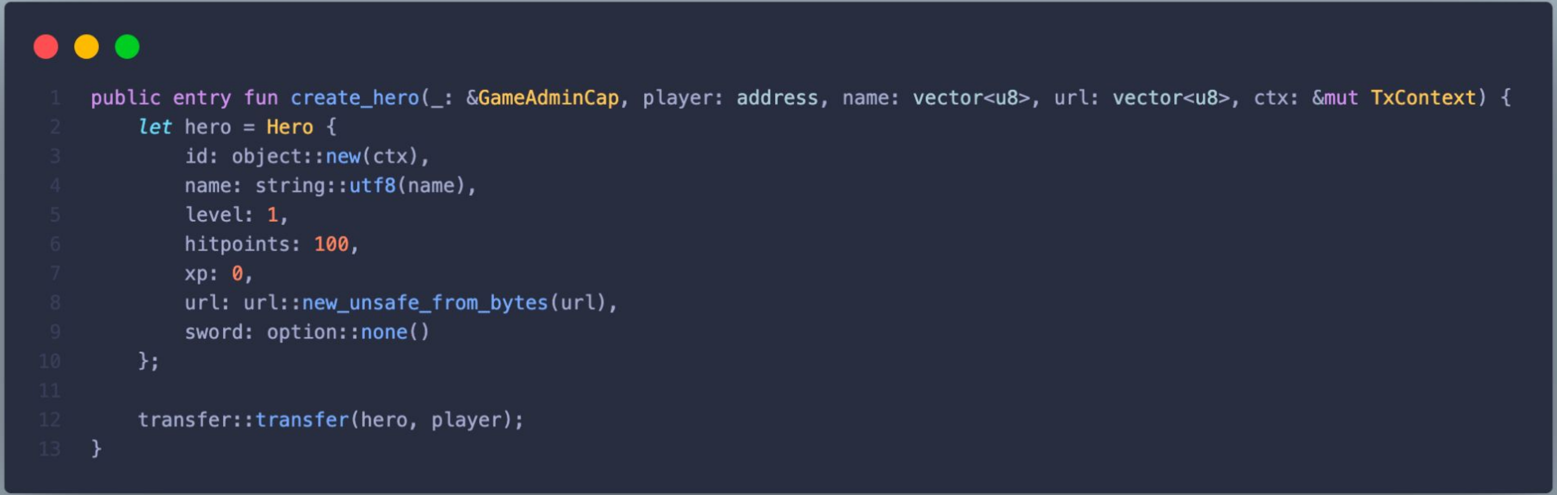

```
struct GameAdminCap has key { id: UID }

struct Hero has key {
    id: UID,
    name: String,
    level: u64,
    hitpoints: u64,
    xp: u64,
    url: Url,
    sword: Option<Sword>,
}

struct Sword has key, store {
    id: UID,
    min_level: u64,
    strength: u64
}
```

sui

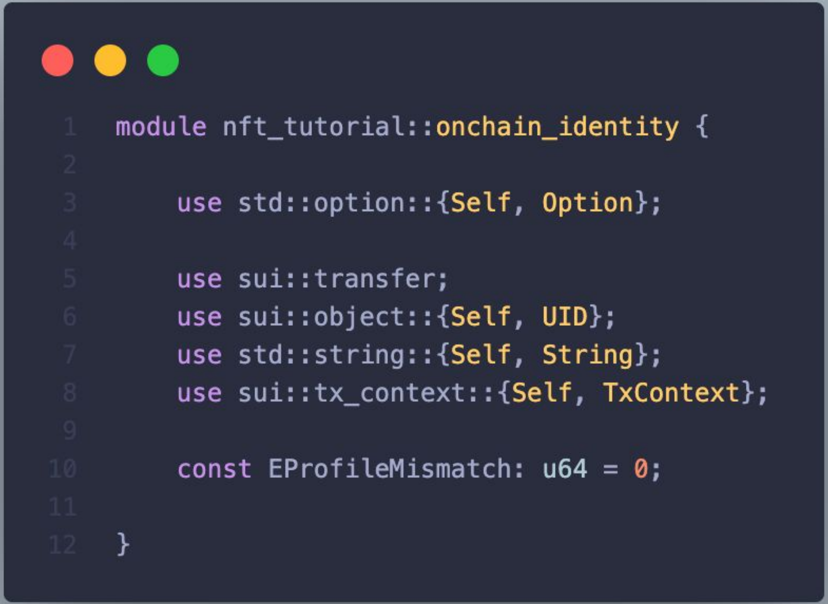


```
fun init(ctx: &mut TxContext) {
    transfer::transfer(
        GameAdminCap {id: object::new(ctx)}
    , tx_context::sender(ctx))
}
```

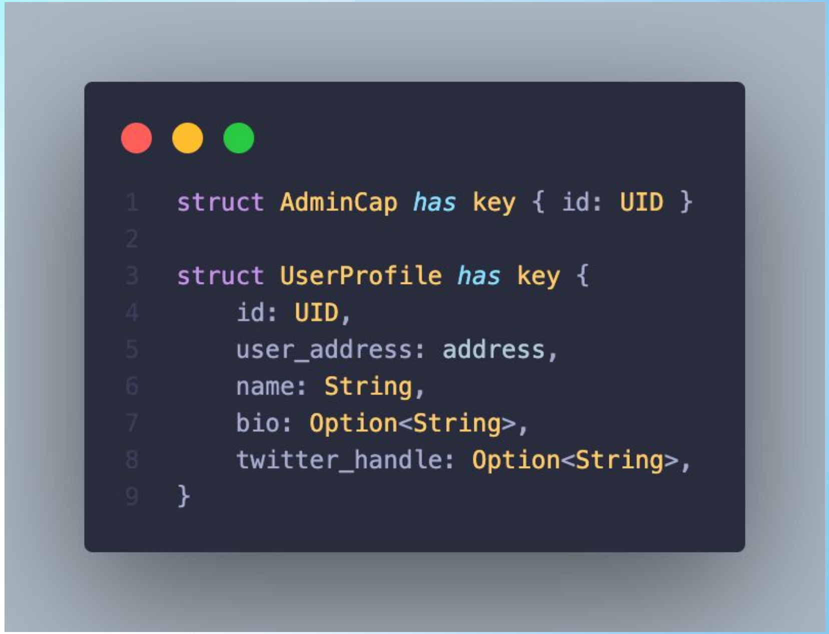
sui



```
public entry fun create_hero(_: &GameAdminCap, player: address, name: vector<u8>, url: vector<u8>, ctx: &mut TxContext) {
    let hero = Hero {
        id: object::new(ctx),
        name: string::utf8(name),
        level: 1,
        hitpoints: 100,
        xp: 0,
        url: url::new_unsafe_from_bytes(url),
        sword: option::none()
    };

    transfer::transfer(hero, player);
}
```

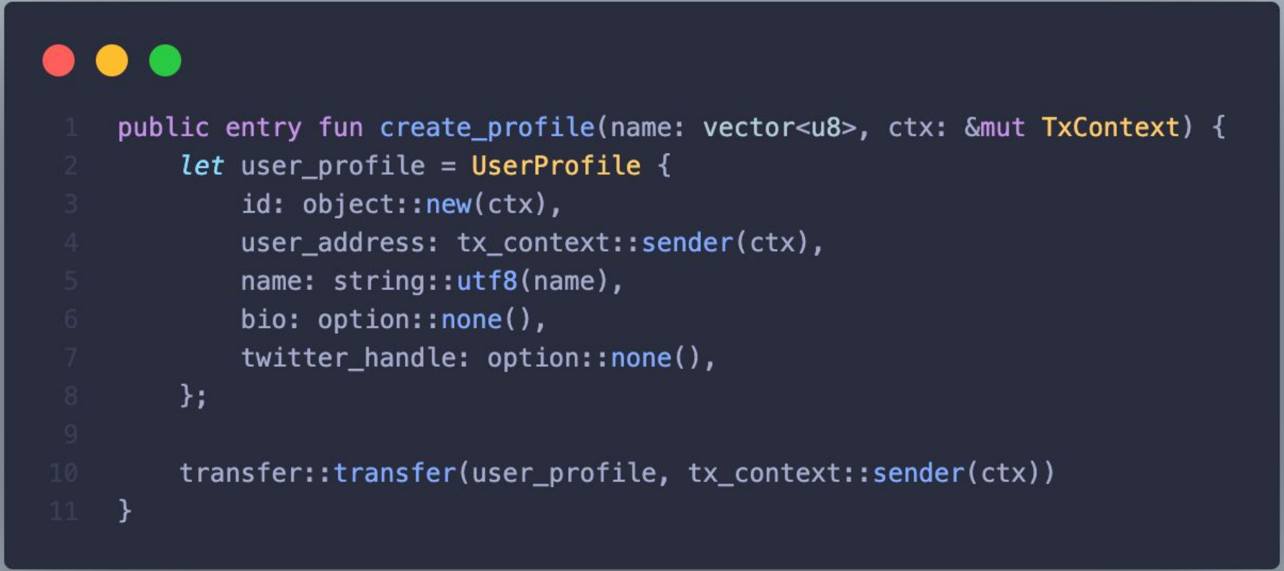

```
module nft_tutorial::onchain_identity {

    use std::option::{Self, Option};

    use sui::transfer;
    use sui::object::{Self, UID};
    use std::string::{Self, String};
    use sui::tx_context::{Self, TxContext};

    const EProfileMismatch: u64 = 0;

}
```

sui



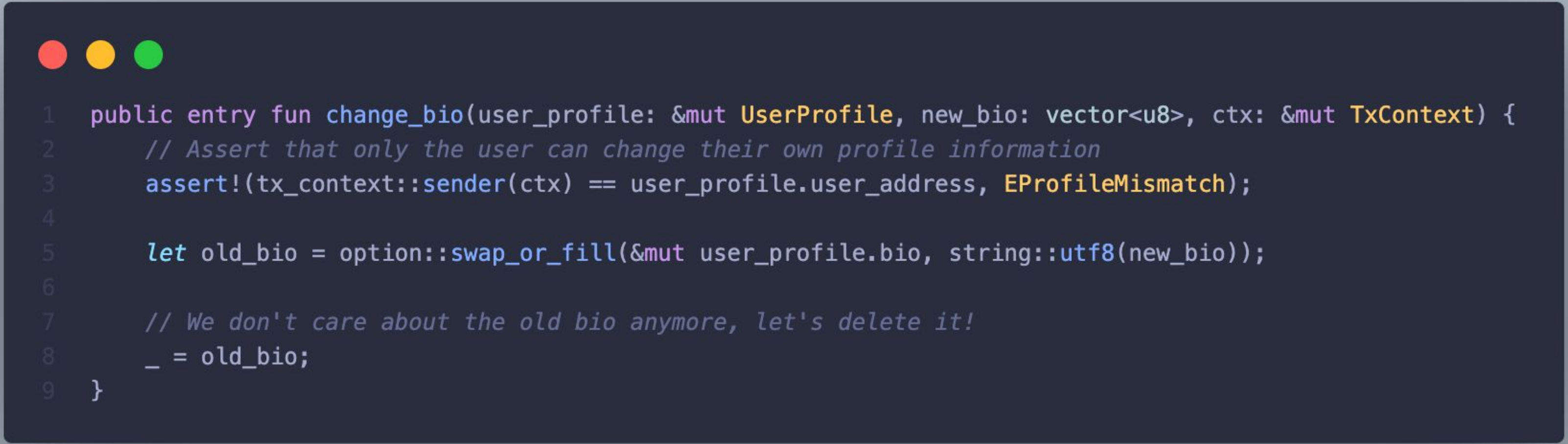
```
struct AdminCap has key { id: UID }

struct UserProfile has key {
    id: UID,
    user_address: address,
    name: String,
    bio: Option<String>,
    twitter_handle: Option<String>,
}
```

sui



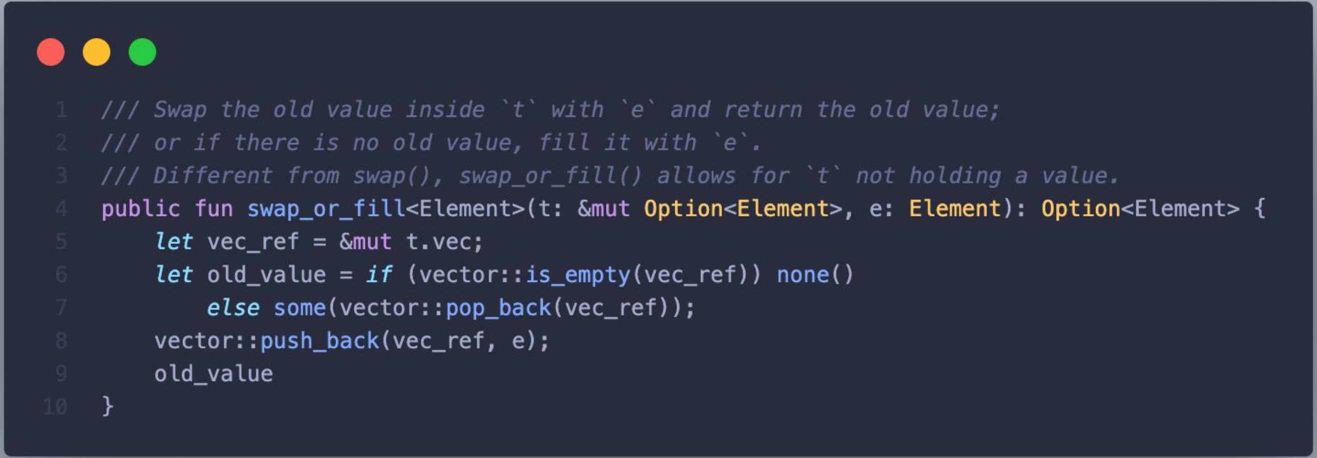
```
public entry fun create_profile(name: vector<u8>, ctx: &mut TxContext) {
    let user_profile = UserProfile {
        id: object::new(ctx),
        user_address: tx_context::sender(ctx),
        name: string::utf8(name),
        bio: option::none(),
        twitter_handle: option::none(),
    };

    transfer::transfer(user_profile, tx_context::sender(ctx))
}
```

sui



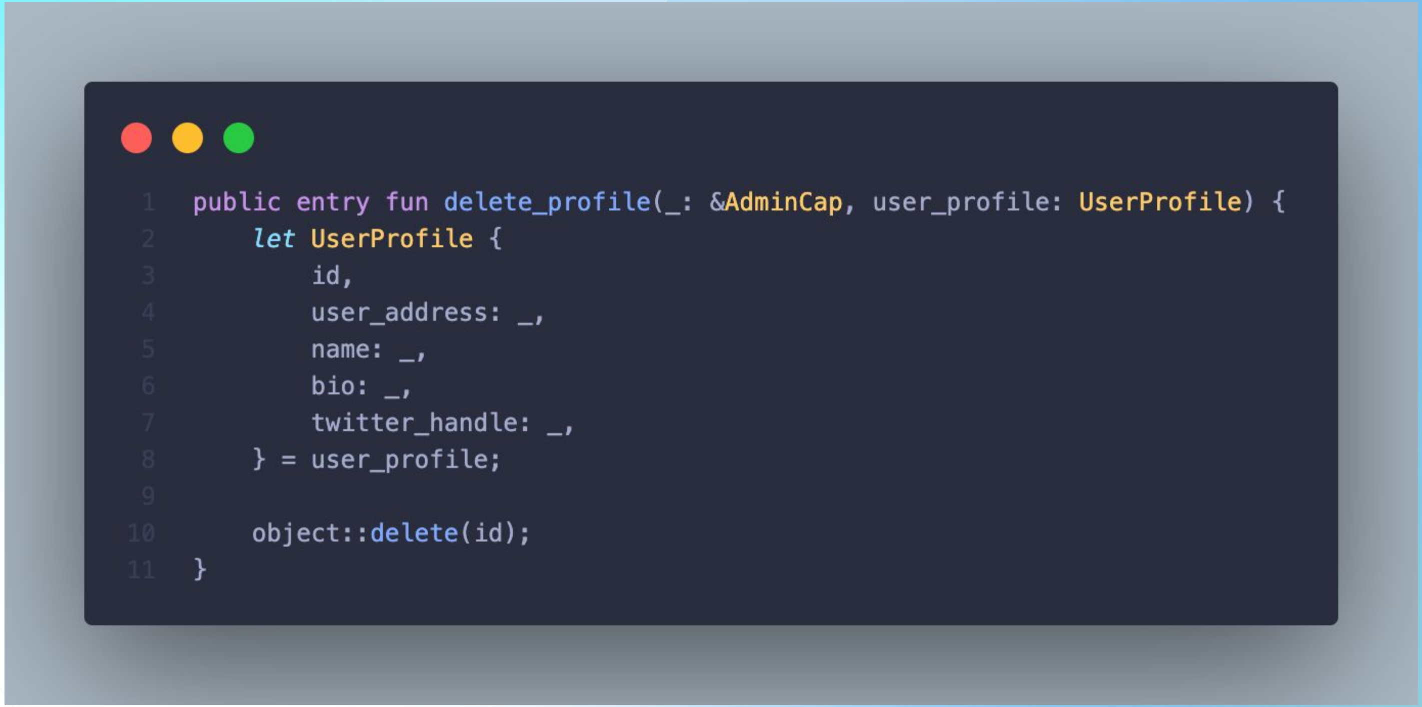
```
public entry fun change_bio(user_profile: &mut UserProfile, new_bio: vector<u8>, ctx: &mut TxContext) {
    // Assert that only the user can change their own profile information
    assert!(tx_context::sender(ctx) == user_profile.user_address, EProfileMismatch);

    let old_bio = option::swap_or_fill(&mut user_profile.bio, string::utf8(new_bio));

    // We don't care about the old bio anymore, let's delete it!
    _ = old_bio;
}
```

sui



```
/// Swap the old value inside `t` with `e` and return the old value;
/// or if there is no old value, fill it with `e`.
/// Different from swap(), swap_or_fill() allows for `t` not holding a value.
public fun swap_or_fill<Element>(t: &mut Option<Element>, e: Element): Option<Element> {
    let vec_ref = &mut t.vec;
    let old_value = if (vector::is_empty(vec_ref)) none()
        else some(vector::pop_back(vec_ref));
    vector::push_back(vec_ref, e);
    old_value
}
```

sui


```
public entry fun delete_profile(_: &AdminCap, user_profile: UserProfile) {
    let UserProfile {
        id,
        user_address: _,
        name: _,
        bio: _,
        twitter_handle: _,
    } = user_profile;

    object::delete(id);
}
```

In Summary:

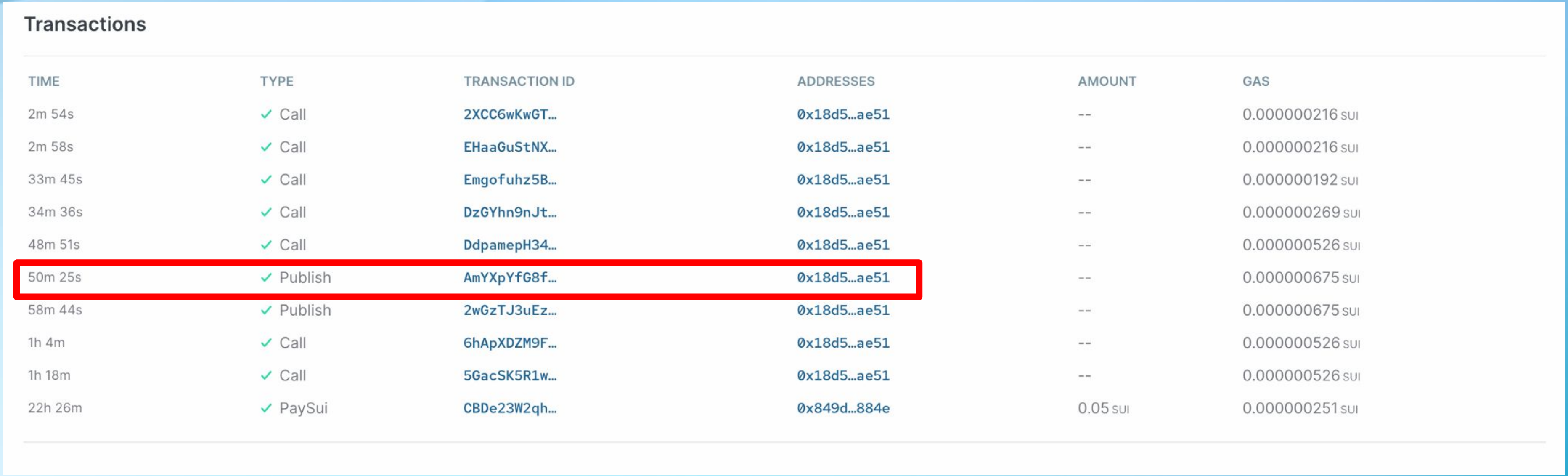
Every Object in Sui is an NFT!

sui

Publishing a Module

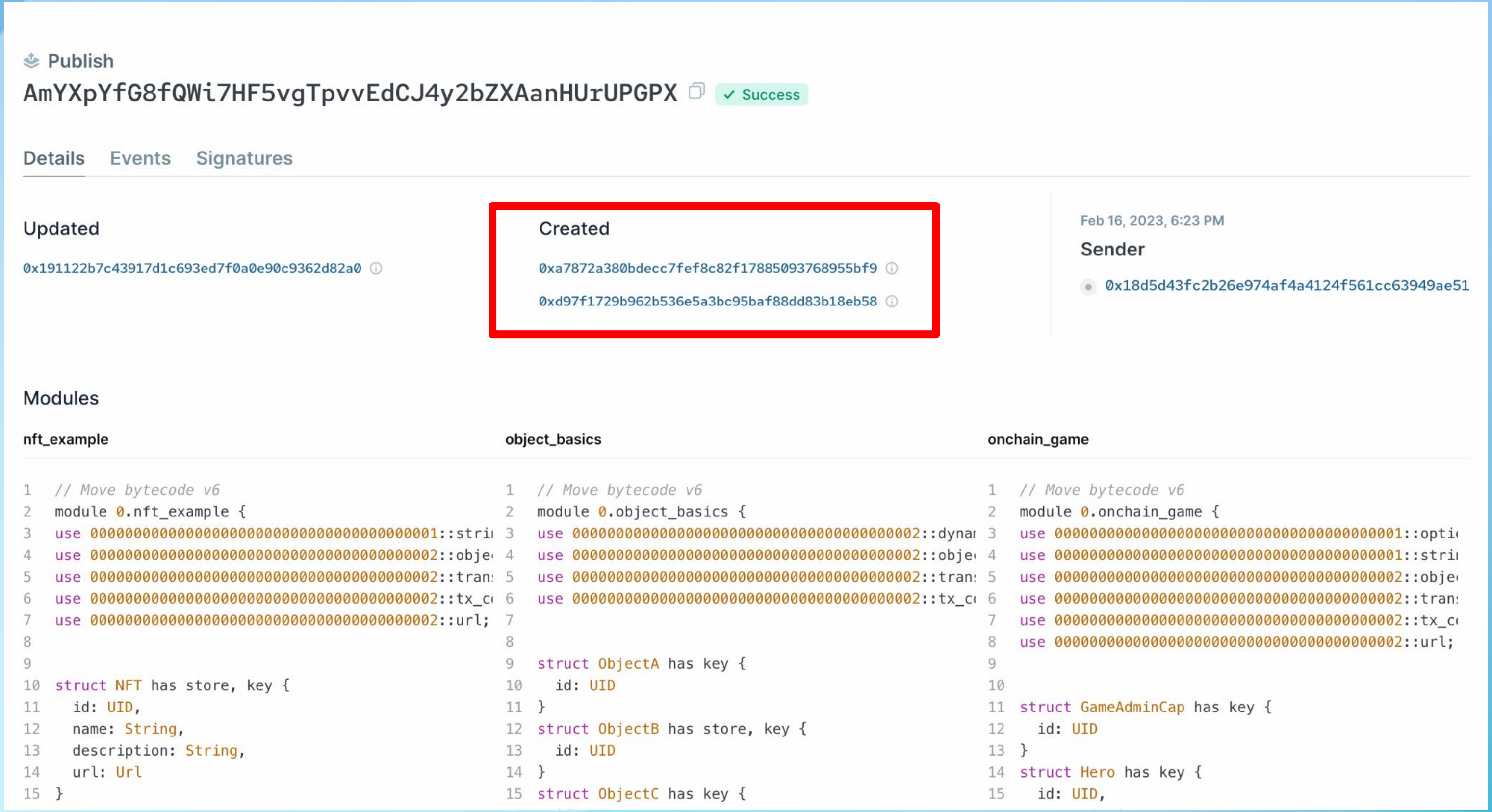
1. `sui client`
2. Download Sui Wallet, import seed phrase
3. `sui client publish nft_example --gas-budget 20000`
4. explorer.sui.io

sui



Transactions

TIME	TYPE	TRANSACTION ID	ADDRESSES	AMOUNT	GAS
2m 54s	✓ Call	2XCC6wKwGT…	0x18d5…ae51	--	0.000000216 SUI
2m 58s	✓ Call	EHaaGuStNX…	0x18d5…ae51	--	0.000000216 SUI
33m 45s	✓ Call	Emgofuhz5B…	0x18d5…ae51	--	0.000000192 SUI
34m 36s	✓ Call	DzGYhn9nJt…	0x18d5…ae51	--	0.000000269 SUI
48m 51s	✓ Call	DdpamepH34…	0x18d5…ae51	--	0.000000526 SUI
50m 25s	✓ Publish	AmYXpYfG8f…	0x18d5…ae51	--	0.000000675 SUI
58m 44s	✓ Publish	2wGzTJ3uEz…	0x18d5…ae51	--	0.000000675 SUI
1h 4m	✓ Call	6hApXDZM9F…	0x18d5…ae51	--	0.000000526 SUI
1h 18m	✓ Call	5GacSK5R1w…	0x18d5…ae51	--	0.000000526 SUI
22h 26m	✓ PaySui	CBDe23W2qh…	0x849d…884e	0.05 SUI	0.000000251 SUI



Publish

AmYXpYfG8fQWi7HF5vgTpvvEdCJ4y2bZXAanHUrUPGPX ✓ Success

Details Events Signatures

Updated

0x191122b7c43917d1c693ed7f0a0e90c9362d82a0

Created

0xa7872a380bdecc7fef8c82f17885093768955bf9

0xd97f1729b962b536e5a3bc95baf88dd83b18eb58

Feb 16, 2023, 6:23 PM

Sender

0x18d5d43fc2b26e974af4a4124f561cc63949ae51

Modules

nft_example

```
// Move bytecode v6
module 0.nft_example {
use 0000000000000000000000000000000000000001::stri
use 0000000000000000000000000000000000000002::obje
use 0000000000000000000000000000000000000002::tran
use 0000000000000000000000000000000000000002::tx_c
use 0000000000000000000000000000000000000002::url;


struct NFT has store, key {
  id: UID,
  name: String,
  description: String,
  url: Url
}
```

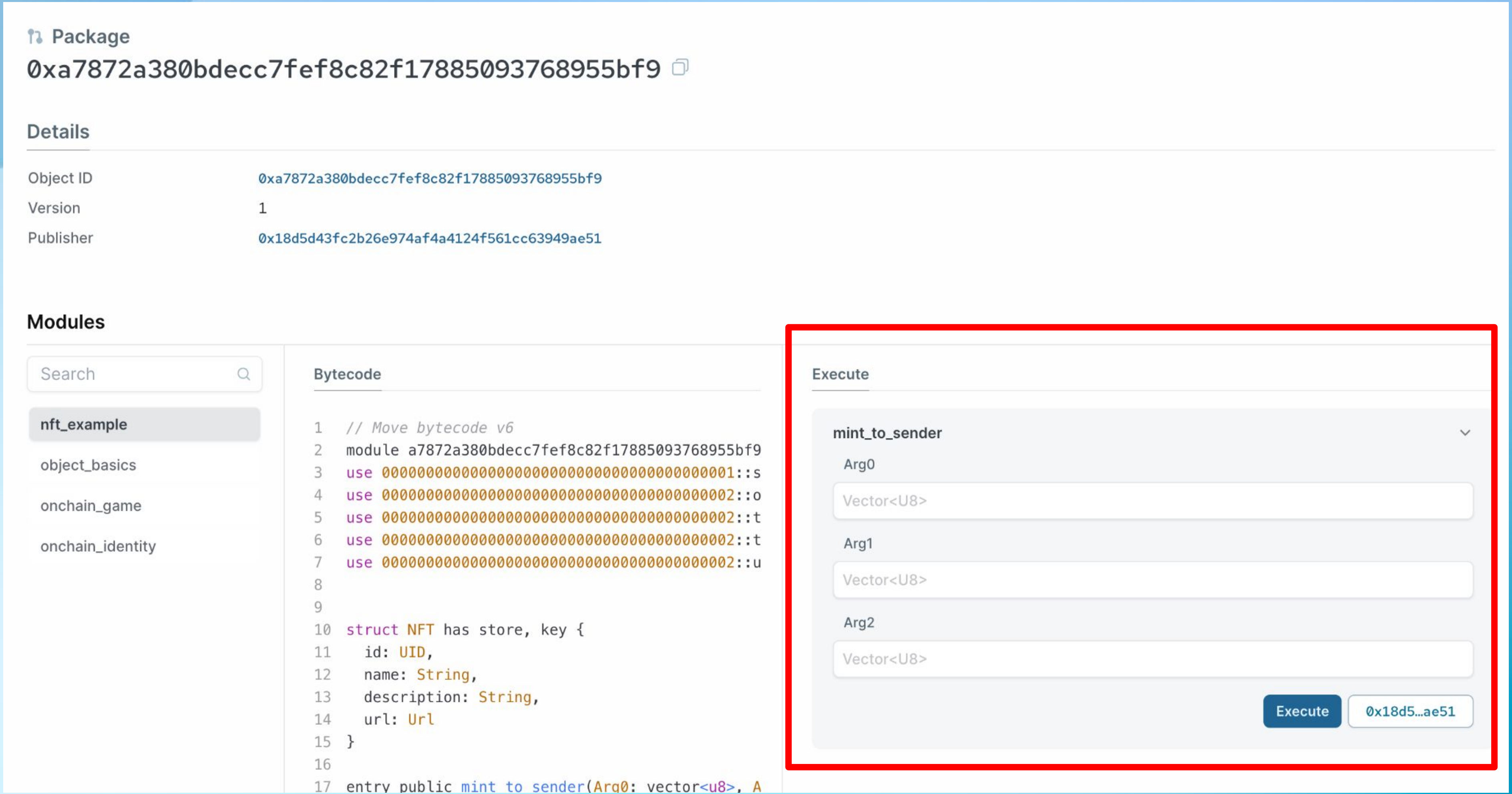
object_basics

```
// Move bytecode v6
module 0.object_basics {
use 0000000000000000000000000000000000000002::dynar
use 0000000000000000000000000000000000000002::obje
use 0000000000000000000000000000000000000002::tran
use 0000000000000000000000000000000000000002::tx_c


struct ObjectA has key {
  id: UID
}
struct ObjectB has store, key {
  id: UID
}
struct ObjectC has key {
```

onchain_game

```
// Move bytecode v6
module 0.onchain_game {
use 0000000000000000000000000000000000000001::optio
use 0000000000000000000000000000000000000001::stri
use 0000000000000000000000000000000000000002::obje
use 0000000000000000000000000000000000000002::tran
use 0000000000000000000000000000000000000002::tx_c
use 0000000000000000000000000000000000000002::url;


struct GameAdminCap has key {
  id: UID
}
struct Hero has key {
  id: UID,
```

sui



Package

0xa7872a380bdecc7fef8c82f17885093768955bf9

Details

Object ID	0xa7872a380bdecc7fef8c82f17885093768955bf9
Version	1
Publisher	0x18d5d43fc2b26e974af4a4124f561cc63949ae51

Modules

Search

- nft_example
- object_basics
- onchain_game
- onchain_identity

Bytecode

```
// Move bytecode v6
module a7872a380bdecc7fef8c82f17885093768955bf9
use 0000000000000000000000000000000000000001::s
use 0000000000000000000000000000000000000002::o
use 0000000000000000000000000000000000000002::t
use 0000000000000000000000000000000000000002::t
use 0000000000000000000000000000000000000002::u


struct NFT has store, key {
  id: UID,
  name: String,
  description: String,
  url: Url
}

entry public mint_to_sender(Arg0: vector<u8>, A
```

Execute

mint_to_sender

Arg0

Vector<U8>

Arg1

Vector<U8>

Arg2

Vector<U8>

Bibliography/ Further Reading

docs.sui.io/learn

examples.sui.io

docs.sui.io/devnet/build/cli-client#publish-packages

sui

What's Next!

sui

Next Workshop:

Introduction to Dynamic Objects + Best Design Practices

8 March 2023

sui

Sui Denver Builder House!

February 28th – March 3rd

lu.ma/suidenver

sui

Survey + Questions?

Twitter: @0xShayan

sui