DFINITY

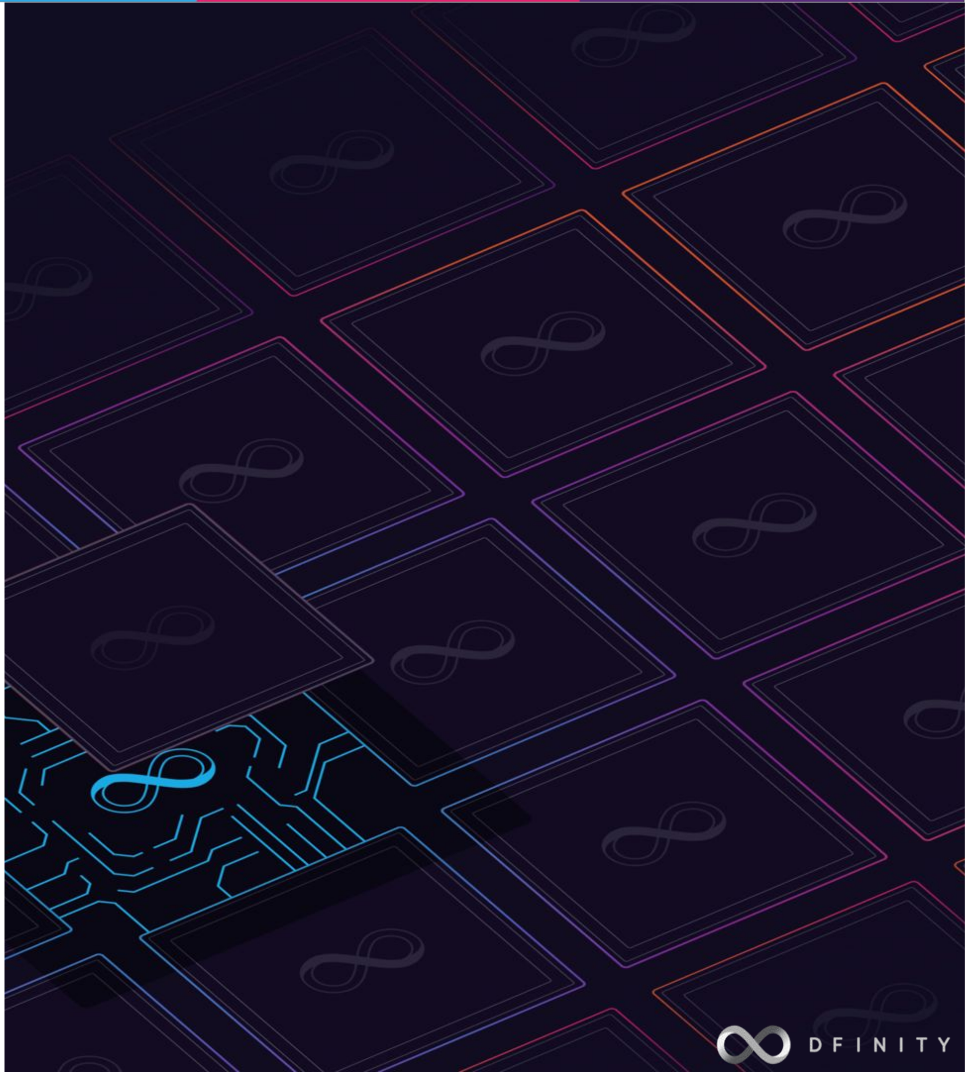
ICP区块链开发进阶课程

2. Canister 开发进阶 I

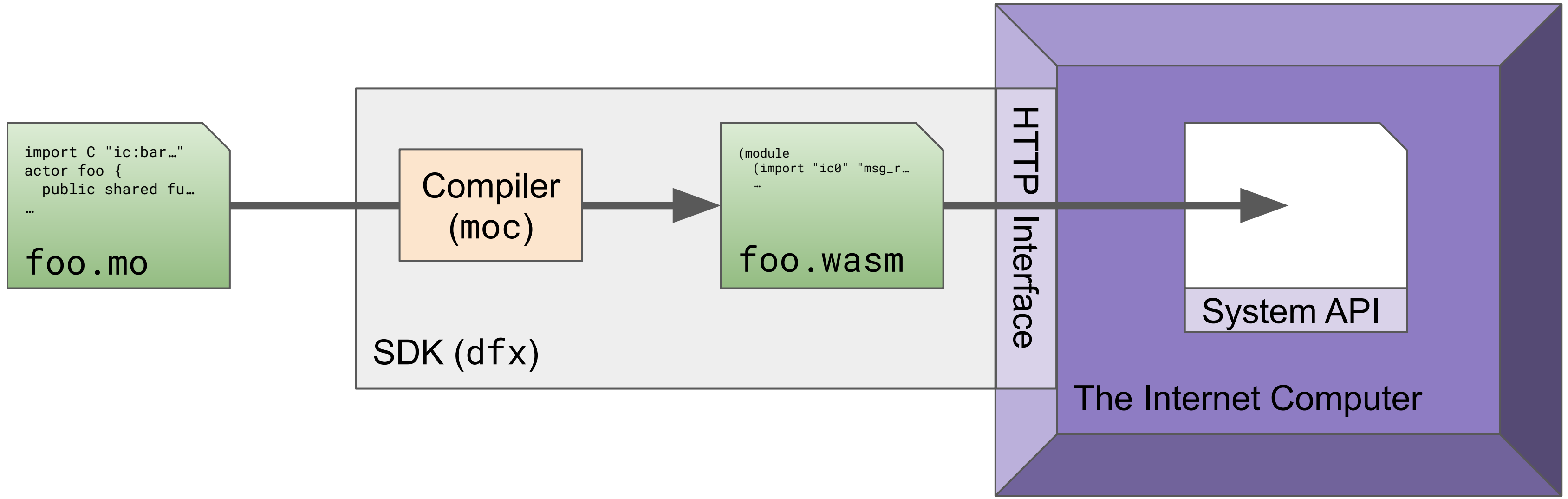
主讲：Paul Liu - DFINITY 工程师

课程大纲

1. Motoko 语言进阶
2. Canister 开发进阶 I ☜
3. Canister 开发进阶 II
4. 整合 ICP 系统服务
5. 项目实例分析



Canister 与系统之间的关系



系统对 Canister 的调用

- canister_init : () -> ()
- canister_pre_upgrade : () -> ()
- canister_post_upgrade : () -> ()
- canister_inspect_message : () -> ()
- canister_heartbeat : () -> ()
- canister_update <name> : () -> ()
- canister_query <name> : () -> ()
- 回调函数，必须符合类型 (env : i32) -> ()

Canister 对系统的调用

```
ic0.msg_arg_data_size : () -> i32;                                          // I U Q Ry F
ic0.msg_arg_data_copy : (dst : i32, offset : i32, size : i32) -> ();        // I U Q Ry F
ic0.msg_caller_size : () -> i32;                                            // I G U Q F
ic0.msg_caller_copy : (dst : i32, offset: i32, size : i32) -> ();           // I G U Q F
ic0.msg_reject_code : () -> i32;                                            // Ry Rt
ic0.msg_reject_msg_size : () -> i32;                                        // Rt
ic0.msg_reject_msg_copy : (dst : i32, offset : i32, size : i32) -> ();      // Rt

ic0.msg_reply_data_append : (src : i32, size : i32) -> ();                  // U Q Ry Rt
ic0.msg_reply : () -> ();                                                   // U Q Ry Rt
ic0.msg_reject : (src : i32, size : i32) -> ();                             // U Q Ry Rt

ic0.msg_cycles_available : () -> i64;                                       // U Rt Ry
ic0.msg_cycles_available128 : (dst : i32) -> ();                            // U Rt Ry
ic0.msg_cycles_refunded : () -> i64;                                        // Rt Ry
ic0.msg_cycles_refunded128 : (dst : i32) -> ();                             // Rt Ry
ic0.msg_cycles_accept : (max_amount : i64) -> ( amount : i64 );             // U Rt Ry
ic0.msg_cycles_accept128 : (max_amount_high : i64, max_amount_low: i64, dst : i32)
                    -> ();                                                  // U Rt Ry

ic0.canister_self_size : () -> i32;                                         // *
ic0.canister_self_copy : (dst : i32, offset : i32, size : i32) -> ();       // *
ic0.canister_cycle_balance : () -> i64;                                     // *
ic0.canister_cycle_balance128 : (dst : i32) -> ();                          // *
ic0.canister_status : () -> i32;                                            // *

ic0.msg_method_name_size : () -> i32                                        // F
ic0.msg_method_name_copy : (dst : i32, offset : i32, size : i32) -> ();     // F
ic0.accept_message : () -> ();                                              // F

ic0.time : () -> (timestamp : i64);                                         // *
```

```
ic0.call_new :                                                              // U Ry Rt H
  ( callee_src  : i32,
        callee_size : i32,
        name_src : i32,
        name_size : i32,
        reply_fun : i32,
        reply_env : i32,
        reject_fun : i32,
        reject_env : i32
  ) -> ();
ic0.call_on_cleanup : (fun : i32, env : i32) -> ();                         // U Ry Rt H
ic0.call_data_append : (src : i32, size : i32) -> ();                       // U Ry Rt H
ic0.call_cycles_add : (amount : i64) -> ();                                 // U Ry Rt H
ic0.call_cycles_add128 : (amount_high : i64, amount_low: i64) -> ();        // U Ry Rt H
ic0.call_perform : () -> ( err_code : i32 );                                // U Ry Rt H

ic0.stable_size : () -> (page_count : i32);                                 // *
ic0.stable_grow : (new_pages : i32) -> (old_page_count : i32);              // *
ic0.stable_write : (offset : i32, src : i32, size : i32) -> ();             // *
ic0.stable_read : (dst : i32, offset : i32, size : i32) -> ();              // *
ic0.stable64_size : () -> (page_count : i64);                               // *
ic0.stable64_grow : (new_pages : i64) -> (old_page_count : i64);            // *
ic0.stable64_write : (offset : i64, src : i64, size : i64) -> ();           // *
ic0.stable64_read : (dst : i64, offset : i64, size : i64) -> ();            // *

ic0.certified_data_set : (src: i32, size: i32) -> ()                        // I G U Ry Rt H
ic0.data_certificate_present : () -> i32                                    // *
ic0.data_certificate_size : () -> i32                                       // *
ic0.data_certificate_copy : (dst: i32, offset: i32, size: i32) -> ()        // *

ic0.debug_print : (src : i32, size : i32) -> ();                            // * s
ic0.trap : (src : i32, size : i32) -> ();                                   // * s
```

IC Management Canister

```
service ic : {
  create_canister : (record {
    settings : opt canister_settings
  }) -> (record {canister_id : canister_id});
  update_settings : (record {
    canister_id : principal;
    settings : canister_settings
  }) -> ();
  install_code : (record {
    mode : variant {install; reinstall; upgrade};
    canister_id : canister_id;
    wasm_module : wasm_module;
    arg : blob;
  }) -> ();
  uninstall_code : (record {canister_id : canister_id}) -> ();
  start_canister : (record {canister_id : canister_id}) -> ();
  stop_canister : (record {canister_id : canister_id}) -> ();
  canister_status : (record {canister_id : canister_id}) -> (record {
      status : variant { running; stopping; stopped };
      settings: definite_canister_settings;
      module_hash: opt blob;
      memory_size: nat;
      cycles: nat;
  });
  delete_canister : (record {canister_id : canister_id}) -> ();
  deposit_cycles : (record {canister_id : canister_id}) -> ();
  raw_rand : () -> (blob);
}
```

```
type canister_id = principal;
type user_id = principal;
type wasm_module = blob;

type canister_settings = record {
  controllers : opt vec principal;
  compute_allocation : opt nat;
  memory_allocation : opt nat;
  freezing_threshold : opt nat;
};

type definite_canister_settings = record {
  controllers : vec principal;
  compute_allocation : nat;
  memory_allocation : nat;
  freezing_threshold : nat;
};
```

Candid 接口规范

- 与 Protobuf, CBOR 这一类数据序列化协议的差别
  - 可以描述更多数据类型，包括函数类型，递归类型
  - 函数类型可以用于描述服务接口和方法
  - 升级过程中的类型适配
- 多语言支持（包括）
  - Javascript, Motoko, Rust
  - Python, Go, Haskell, AssemblyScript
  - 生成 Candid 类型规范，编码解码，从 Candid 规范导入数据类型

IC 双向消息传递的保证 (bi-directional messaging)

- 但凡发出的消息，必然会收到回答
  - 升级的时候，需要先 stop 再 upgrade
- 每个消息最多被处理一次（没有被处理的，会返还错误给发送方）

```
try {
    ...
    let r = await x.f(...);
    ...
} catch (err) {
    ...
}
```

DFINITY

常见错误

```
var balance = ...;
 public func send(amount: Int, user: Principal) : async Nat {
     let accounts = actor("....");
     if (balance >= amount) {
        let exists = await accounts.exists(user);
        if (exists) {
            balance -= amount;
            await accounts.credit(user, amount);
        }
     }
     ...
 }
```

```
var jobs : Buffer<Job> = ...;
public func dispatch() : async () {
    if (jobs.size() > 0) {
        let n = jobs.size() - 1;
        let job = jobs.get(n);
        try {
            await execute(job);
            ignore jobs.removeLast();
        } catch (err) {
            Debug.print("Error dispatching job")
        }
    }
}
```

Motoko 异常处理

Motoko *try/catch* 仅用于对异步的异常处理

```
public func dec(v: Nat) : async Nat {
    assert(v > 0);
    v - 1
};

public func test(n: Nat) : async Nat {
    try {
        await dec(n)
    } catch(e) {
        Debug.print("Caught error: " #
                    Error.message(e));
        0
    }
};
```

```
func dec(v: Nat) : Nat {
    assert(v > 0);
    v - 1
};

public func test(n: Nat) : async Nat {
    try {
        dec(n)
    } catch(e) {
        Debug.print("Caught error: " #
                    Error.message(e));
        0
    }
};
```

Candid 工具演示

课程作业

实现一个简单的多人 Cycle 钱包：

1. 团队 N 个成员，每个人都可以用它控制和安装 canister。
2. 升级代码需要 M/N 成员同意。

```
create_canister
install_code
start_canister
stop_canister
delete_canister
```

下一节：Canister 开发进阶 II

- Canister 代码升级
- Cycles 管理
- 证书签名及网络安全

DFINITY