DFINITY 课程作业讲解 - Explorer

第1课 - 使用 SDK 搭建简易网站 & 第2课 - Motoko 语言入门

课程关键内容回顾

- DFINITY 开发者中心，下载 DFINITY Canister SDK

```
sh -ci "$(curl -fsSL https://smartcontracts.org/install.sh)"
# 课程视频中是 v0.8.3，当前版本已是 v0.8.4
```

- 使用 DFINITY Canister SDK

```
# 1. 初始化新项目
# 将以下命令中 <project_name> 替换成自己的项目名称
dfx new <project_name>

# 2. 启动项目，加上 --backgroud 可以运行在后台
dfx start

# 3. 部署项目网络，保存 canister_id
dfx deploy

# 4. 访问部署在本地的项目
# 用前端 assets 的 canister_id 替换以下命令中的 <canister_id>
http://<canister_id>.localhost:8000
```

- DFINITY Cycles 水龙头，Canister 容器 ～ 智能合约，使用 cycles 付费，通常由开发者为自己的容器充值

```
# 1. 登录自己的 Github 账号，并授权认证

# 2. 本地生成 DFX Principal ID
dfx identity get-principal

# 3. 生成 IC 网络的新钱包/使用已有钱包

# 4. 记录 Wallet ID
```

- 部署到 **ic0.app** 主网

```
# 1. 设置在 IC 网络的默认钱包
# 将以下命令中 <Wallet ID> 替换成自己的钱包ID
dfx identity --network ic set-wallet --force <Wallet ID>

# 2. 查询 IC 网络中的钱包余额
dfx wallet --network=ic balance

# 3. 部署项目到 IC 网络，保存 canister_id
dfx deploy --network ic

# 4. 访问部署在 IC 网络的项目
# 将以下命令中 <canister_id> 替换成自己应用的canister_id
https://<canister_id>.ic0.app
```

- ICP 技术优势
  - 48字节 ChainKey 公钥验证，不让用户安装任何插件
  - 毫秒级查询响应，2秒交易最终确认；
  - 简化了区块链 DAPP 架构；
  - Canister，轻量级容器封装代码和数据；
  - JS Agent 库提供底层支持；

  对安全性的需求：数据安全，计算可信 ------> 去中心化
- Motoko 语言
  - Actor 模型，权限管理，代码升级，跨语言调用，数据描述语言 Candid
  - 配置开发环境 VS Code，安装 Motoko 插件（v0.3.9）
- Motoko 语言基础语法
  - 语法接近 JavaScript/TypeScript，不同点：语句赋值（`:=`）和声明（`=`），变量分可变（`var`）和不可变（`let`），类型标注等；
  - 数据类型，`Int`，`Nat`，`Bool`，`Char`，`Text`，`Iter`，`Func`，`Hash`
  - Actors，`Actor`关键字，异步调用
  - 模块导入
- Motoko 语言实践
  - 编译器 moc，解释器环境
  - Actor，公共接口，调用 Canister 中的方法
  - 使用 Candid UI 调试
  - 使用 Motoko playground

```
# 查看sdk安装目录
dfx cache show

# 编译
moc -r main.mo

# 编译指定路径
moc --package base $(dfx cache show)/base -r main.mo

# 调用 Canister 中的方法
dfx canister call <canister> <function_name> <function_parameters>
```

课程作业点评

第1课共有 4 道题目

1. 安装并使用 SDK 在本机搭建一个简易网站 。（2 分，请提交截屏）

   DFINITY Canister SDK

2. 前往 faucet.dfinity.org 领取 cycles 。（2 分，请提交领取成功的截屏，以及 principal id）

   DFINITY Cycles 水龙头

3. 将网站部署到 ic0.app 主网。（3 分，请提交部署的网址 URL）

   dfx deploy --network ic

4. 思考题：假如开发团队不再维护代码了，用户该怎么办？（3 分）

   去中心化应用的理解；

7 组 17 位学员，有 12 位提交了作业，满分的有 6 位。

第2课有 1 道编程题目

1. 用 Motoko 语言实现一个快排函数，封装到智能合约 Canister 中，然后部署到主网提供 Canister ID。（10 分）

   快排函数：quicksort : [var Int] -> ()

   接口：public func qsort(arr: [Int]): async [Int]

   - 快排算法，也称为分区交换排序

```
# low  --> Starting index,  high  --> Ending index
quicksort(arr[], low, high)
{
    if (low < high)
    {
        # pi is partitioning index
        pi = partition(arr, low, high);
        # or pi is at first/last index
        quicksort(arr, low, pi - 1);  // Before pi
        quicksort(arr, pi + 1, high); // After pi
    }
}
```

- Motoko 基础库，`Array`,`Int`
- 不可变数组 <-> 可变数组，`Array.thaw`,`Array.freeze`
- Motoko Actor，公共接口
- Candid UI https://a4gq6-oaaaa-aaaab-qaa4q-cai.raw.ic0.app

7 组 17 位学员，有 11 名提交了作业，满分的有 7 位。

作业常见问题分析

- 本机搭建的网站，没有运行；
- 领取 cycles 时，Github账号不符合条件，或者未提交领取成功的页面；
- 不是用的快排算法
- 封装到 Canister 没有用 Actor
- 没有部署到主网
- Motoko 语法不熟练，没有使用 `Array` 库

```
#1
public func qsort(arr:[Int]): async [Int]{
    var arr_var : [var Int] = Array.thaw(arr);
    quicksort(arr_var);
    let arr_new :[Int] = Array.freeze(arr_var);
    return arr_new;
}

#2
public func qsort(arr: [Int]): async [Int] {
    var arr2: [var Int] = Array.thaw(arr);
    quicksort(arr2);
    Array.freeze(arr2)
};

#3
public func main(arr: [Int]) : async [Int]{
```

```
    var newarr : [var Int] = putArr(arr);
    let size : Nat = newarr.size() - 1;
    sort(newarr);
    freeze(newarr);
};
```

补充资料

- 开发者中心快速入门系列
  - Host a Static Website
  - Claim Your Free Cycles
  - Network Deployment
- Motoko 语言
  - Motoko 基础库
  - Motoko Array
  - Motoko Quicksort 实现
- 快速排序 wiki