本课程分为两个部分：第一部分主要学习 Cadence 语言；在学完 Cadence 语言之后，第二部分我们来开发一个完整的去中心化应用(decentralized application)，也就是 Dapp。文字讲义中涉及到知识点和概念的使用参见视频教程，您可通过以下方式联系并加入开发者引擎社区获取：

从本节课开始进入第二部分 DApp *的开发，整个教程基于官方* CryptoDappy *教程。期间会涉及到* js, React *框架等前端知识，如果您对前端不是太熟悉，视频演示教程会在关键节点给大家讲解其含义和作用，以便对整个项目有了解。*

第五课 认识 FCL 与初步使用

本节课主要分为三个部分：

1、初始化 React 项目

2、Flow 客户端库(FCL)

3、CryptoDappy 应用开发

一、初始化 React 项目

1.1、初始化 React App

执行以下命令建立一个最简单的 React App 项目。

1、安装 node

> brew install node

2、初始化 react-app：参见 https://create-react-app.dev/docs/getting-started

> npx create-react-app my-app

> npm install --save react-router-dom

> cd my-app

> npm start;

在作者的机器上执行时有报错，错误截图和解决方式如下图中所示：

（错误解决：https://stackoverflow.com/a/71836018/12490294 ）

```
Installing template dependencies using npm...
npm ERR! code ERESOLVE
npm ERR! ERESOLVE unable to resolve dependency tree
npm ERR!
npm ERR! While resolving: react-18@0.1.0
npm ERR! Found: react@18.0.0
npm ERR! node_modules/react
npm ERR!   react@"^18.0.0" from the root project
npm ERR!
npm ERR! Could not resolve dependency:
npm ERR! peer react@"<18.0.0" from @testing-library/react@12.1.5
npm ERR! node_modules/@testing-library/react
npm ERR!   @testing-library/react@"^12.0.0" from the root project
npm ERR!
npm ERR! Fix the upstream dependency conflict, or retry
npm ERR! this command with --force, or --legacy-peer-deps
npm ERR! to accept an incorrect (and potentially broken) dependency resolution.
```

11

Until this is fixed for now you can delete the `node_modules` folder and `package-lock.json`.
Next, open `package.json` and change
`"react": "^18.0.0"` & `"react-dom": "^18.0.0"` to an earlier version e.g:
`"react": "^17.0.2"` & `"react-dom": "^17.0.2"`.
Finally, you can run `npm install`.

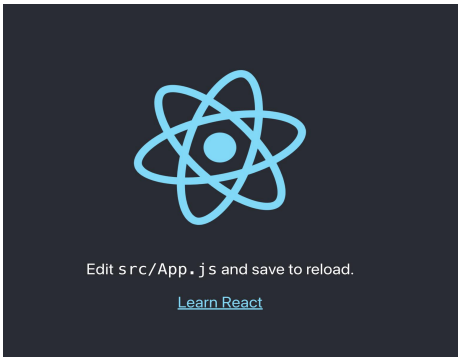
Alternative Solution (Try this first!):

solution suggested by joooni1998):

1. delete both `node_modules` and `package-lock.json`
2. run `npm i web-vitals --save-dev`
3. run `npm install`

and then you can use `npm run build` and `npm start` again

解决之后，再次运行 npm start，浏览器自动弹出或手动访问 http://localhost:3000 展示：



至此 React App 初始化成功。

1.2、教程示例代码介绍

示例代码每个分支代表阶段性的成果，如下表说明。

地址：https://github.com/yifanshaoye/FLowDapp

分支	功能
main	主分支，最新修改，包含所有功能
create_react_app	初始化 react_app
import_fcl_fetch_bookCount	从链上获取数据(demo 书籍的个数)
cryptodappy_init	cryptodappy 项目初始化，相当于完成前端的开发，准备对接区块链获取数据
user_wallet_login	加入用户登录逻辑
scripts_interact_display	演示 scripts 交互：获取 templates 列表，账户余额，检查 collection
transaction_interact_display	演示交易的交互：创建 Collection，获取用户 Dappy, mint 一个 Dappy
watch_transaction	模拟监控展示交易的状态变化

如上表的 create_react_app 即为本小节的成果，如果您还是无法在自己的机器上成功初始化 React App，可以直接下载示例代码，按如下操作：

1）克隆代码切换到 create_react_app；

2）npm install 安装依赖；

3）执行 npm start

二、Flow 客户端库(FCL)介绍

2.1、FCL 作用概览

用第一课的 Flow 开发图示说明，FCL 替换的是原先访问后端的 HTTP/REST 库，所以其工作位置位于

浏览器这一部分。此外，视频 https://www.youtube.com/watch?v=Li9BmEroyvs 中也强调了两点内容：
1、FCL 用于与链上进行交互，交易的签名发送等。在接下来的实践中也会发现，FCL 的封装对开发者十分的友好，简洁易用。
2、FCL 提供了钱包自动发现服务，只要实现了此机制的钱包，都可以被用户看到。这实际上大大简化了应用开发者试图支持不同钱包的工作量。

如下图为 FCL 在 DApp 中的位置作用：

2.2、FCL API 概览

FCL API 包含的内容很多，本课程不可能一一讲解，只是讲解演示一些用处最多的 API，其余的若在实际开发中用到，再查阅相关文档即可。

1、安装 FCL 命令；

> npm install @onflow/fcl --save

2、配置的添加与获取, config()；
3、发送 Script 脚本获取链上数据，query()；
4、登录验证与当前用户；

authenticate()，unauthenticate()，currentUser

5、发送交易变更链上状态, mutate(), tx。

2.3、FCL 初步使用演示

1、准备工作
1）代码准备

如果用自己初始化的项目，由于兼容性问题，建议确认一下 React App 项目 package.json 中依赖的版本号与下图所示统一：

```
"dependencies": {
  "cra-template": "1.1.3",
  "react": "^17.0.2",
  "react-dom": "^17.0.2",
  "react-router-dom": "^5.2.0",
  "react-scripts": "4.0.3"
},
```

或者直接用提供的示例代码切换到 create_react_app 分支。

2）合约准备

视频教程演示中首先尝试与本地模拟器交互，但报错了。于是改为与测试网进行交互。上节课实践了在本地模拟器部署合约，这里将该合约部署到测试环境。此时需要一个测试网的账号，可以在https://testnet-faucet.onflow.org/网站创建一个账号。

首先，修改配置文件 flow.json 的 accounts 和 deployments 配置项，如下图所示。然后执行命令将合约部署到测试网的账户上。

> flow project deploy --network=testnet

```
"accounts": {
    "emulator-account": {
        "address": "f8d6e0586b0a20c7",
        "key": "bf4fc31ede0xxxxxxxxxx5b72f39d7b6974d627b"
    },
    "emulator-creator": {
        "address": "01cf0e2f2f715450",
        "key": {
            "type": "hex",
            "index": 0,
            "signatureAlgorithm": "ECDSA_secp256k1",
            "hashAlgorithm": "SHA3_256",
            "privateKey": "f3873884e24exxxxxxxxabfcb6f2f748"
        }
    },
    "testnet-creator": {
        "address": "f4f8e2aa8f2ee51f",
        "key": "59ec436ab62b166e3113xxxxxx7e11ed01828e38d02c"
    }
},
"deployments": {
    "emulator": {
        "emulator-creator": [
            "HelloWorld"
        ]
    },
    "testnet": {
        "testnet-creator": [
            "HelloWorld"
        ]
    }
}
```

2、导入 FCL 并从链上获取数据

本小节的成果分支为 import_fcl_fetch_bookCount。

1）导入 FCL

新建配置文件 src/config.js，添加如下代码，其中 accessNode.api 配置为测试网的连接节点，类似于 web2 中的域名。

src > JS config.js > ...

```
1  import { config } from "@onflow/fcl"
2
3  config({
4    "accessNode.api": "https://access-testnet.onflow.org"
5  })
```

在项目入口文件 src/index.js 中导入新建的配置文件：

```
import reportWebVitals from './reportWebVitals';

import "./config"

ReactDOM.render(
  <React.StrictMode>
```

2）添加与链上交互逻辑

修改 src/App.js 文件，以下节选：

```
try {
  let res = await query({
    cadence: `
    import HelloWorld from 0xf4f8e2aa8f2ee51f

    pub fun main(): Int {

      let countRef = getAccount(0xf4f8e2aa8f2ee51f).getCapability(HelloWorld.BookShelfCountPath)
          ?? panic("can't borrow count bsf")

      return countRef.getBookCount()

    }
    `,
  });
```

其中，核心点在于使用 query()函数从链上获取数据，入参 cadence 接受一段 Script 脚本的字符串，该脚本获取有几本书籍。

3）更改前端展示

修改 src/App.js 文件的组件返回部分，添加展示书籍本数的逻辑：

```
    >
      Learn React
    </a>
    <br></br>
    Fetch {bookcount} Books
  </header>
</div>
```

重新运行 React App，得到如下执行结果，表名与链上交互成功。

三、CryptoDappy 应用开发

3.1、CryptoDappy 项目介绍

CryptoDappy 是官方的一个 Flow DApp 开发的教程。此教程通过任务的形式，使用 FCL 一步步的构建出一个较完整的应用。示例代码的每个分支代表一个任务。

官方网站：https://www.cryptodappy.com/

示例代码 github：https://github.com/bebner/crypto-dappy

如果仔细的分析一下这个教程，实际包含两方面的内容：一部分就是关于 fcl 的使用，怎么去跟链进行交互等等；另一部分更多是业务场景上的逻辑，在我看来业务是有重复的。比如说在这个教程当中，mint 一个 CryptoDappy 单品和 mint 一个 pack，即一组 CryptoDappy，站在 fcl 使用的这个角度上差别并不大，都是发送一个交易。所以本课程不会去复述官方教程的全部内容，而是充分的覆盖到里面涉及到的 fcl 的应用场景。主要包含三个方面：1）账户登录与当前账户；2）发送脚本 script 并执行；3）发送交易并监控执行。接下来完善账户的登录逻辑。

3.2、FCL 账户登录演示

1、代码准备

本小节演示的起始分支为 cryptodappy_init，可以理解为从 create_react_app 分支，进行前端工作的开发完成，开始与链上进行交互的状态。类似于 web2 当中，前后端各自独立开发完毕可以进行联调了。前端在开发过程中，用到的后端数据都先写死。

切到 cryptodappy_init 分支。此时不含任何与 FCL 相关的代码逻辑。npm start 运行打开 CryptoDappy 的主界面。

2、导入 FCL 并添加登录逻辑

本小节的成果分支为 user_wallet_login。

1）导入 fcl 并添加配置

新建 src/config/config.js 文件，添加如下内容：

```
import { config } from "@onflow/fcl";

config({
    "accessNode.api": "https://access-testnet.onflow.org",
    "discovery.wallet": "https://fcl-discovery.onflow.org/testnet/authn"
})
```

其中，discovery.wallet 即为前面提到的钱包自动发现服务机制所需。

在入口文件 src/index.js 中引入：

```
import './components/Atoms.css'
import './config/config'

ReactDOM.render(
  <Providers>
```

2）修改登录逻辑

与登录相关的文件为 src/hooks/use-current-user.hook.js，很明显可以看到是否登录的判断是写死的。与原文件相比，改动如下：

```
1+ import { useState, useEffect } from 'react'
2+ import * as fcl from "@onflow/fcl"
3
4  export default function useCurrentUser() {
5    const [user, setUser] = useState({ loggedIn: false })
6
7    const tools = {
8+     logIn: fcl.authenticate,
9+     logOut: fcl.unauthenticate,
10   }
11
12+  useEffect(() => {
13+    fcl.currentUser.subscribe(setUser)
14+  }, [])
15
16   return [user, user?.addr != null, tools]
17 }
18
```

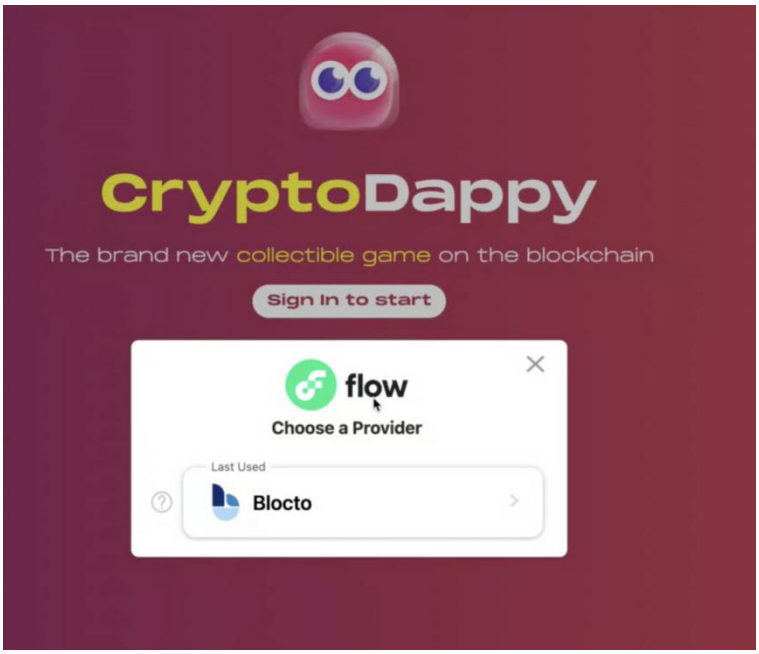
如上，fcl.authenticate 为登录调用函数；fcl.unauthenticate 为登出函数。

fcl.currentUser 表示当前已经登录的的用户信息；subscribe()表示订阅当前用户状态的变化，其入参为一回调函数，当用户状态发生变化时，subscribe()内部变会调用该回调函数。用户信息在 FCL 内部的结构体表示如下表所示：

CurrentUserObject

KEY	VALUE TYPE	DEFAULT	DESCRIPTION
addr	Address	null	The public address of the current user
cid	string	null	Allows wallets to specify a content identifier for user metadata.
expiresAt	number	null	Allows wallets to specify a time-frame for a valid session.
f_type	string	'USER'	A type identifier used internally by FCL.
f_vsn	string	'1.0.0'	FCL protocol version.
loggedIn	boolean	null	If the user is logged in.
services	[ServiceObject]	[]	A list of trusted services that express ways of interacting with the current user's identity, including means to further discovery, authentication, authorization, or other kinds of interactions.

修改完成之后，重新启动运行，点击登录，弹出如下界面。这是测试网，只展示了一个 blocto 的钱包，理论上是可以展示多个钱包。选择一个钱包，按照指引进行登录或注册就可以了。


CryptoDappy
The brand new collectible game on the blockchain
Sign In to start
flow
Choose a Provider
Last Used
Blocto

本节课结束~