一文快速入门FISCO BCOS Node.js SDK

原创 李陈希 FISCO BCOS开源社区 2019-11-04

李陈希

FISCO BCOS核心开发者

喜欢并行的艺术

AUTHOR | 作者

SDK一般是指为了方便开发者为系统开发应用而提供的API、开发辅助工具以及文档的集合。面对不同的开发者群体，FISCO BCOS已经相继推出了面向企业级应用开发者的Java SDK（功能丰富、稳定），以及面向个人开发者的Python SDK（上手迅速、轻便）。

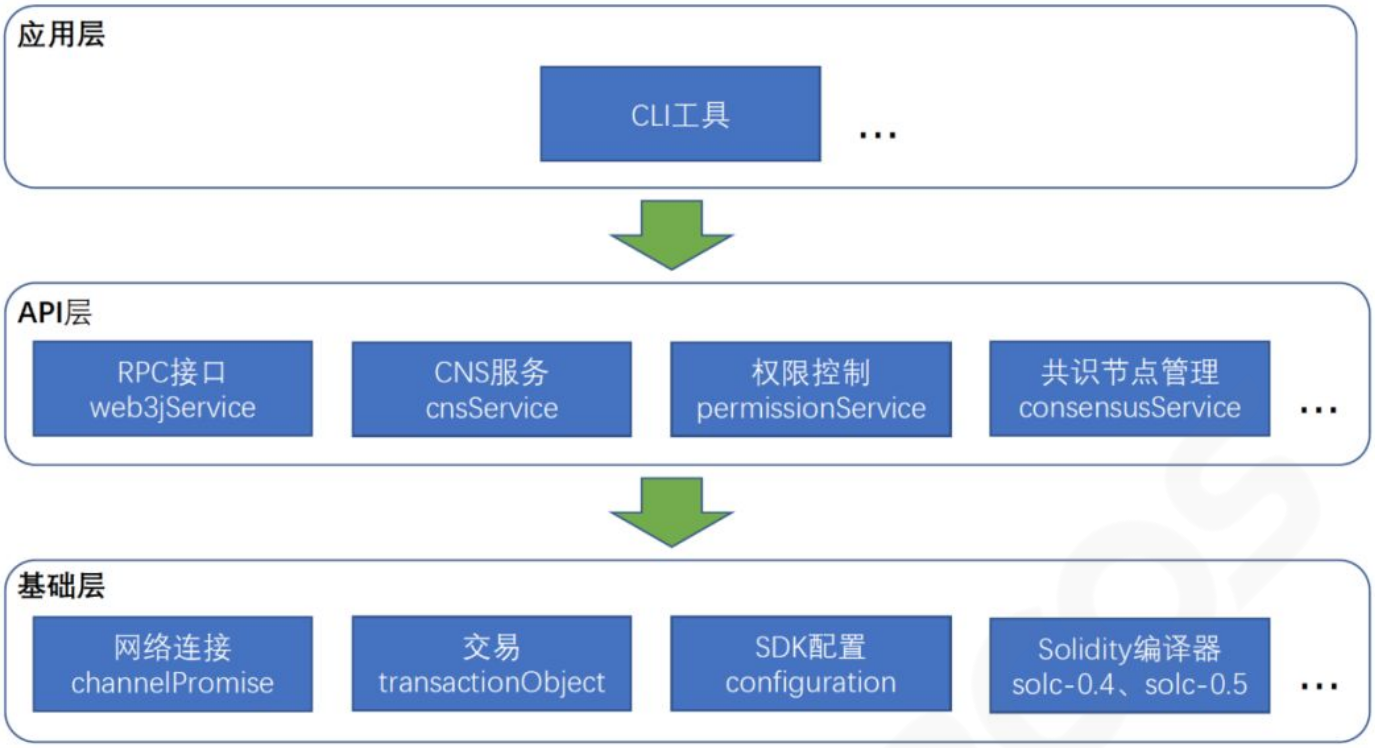
在实际推广过程中，我们发现有部分开发者习惯使用JavaScript构建应用前端，倘若此时能有一个支持使用JavaScript进行后端开发的SDK，那么前后端的语言壁垒能够进一步被打破——开发者只需了解JavaScript，便能完成整个FISCO BCOS应用程序的前后端开发。

在此背景下，FISCO BCOS Node.js SDK诞生了。本文将会介绍Node.js SDK的设计及使用。

FISCO BCOS

一、SDK的设计

Node.js SDK遵循分层设计原则，层与层之间界限清晰、功能明确，下图为Node.js SDK的架构示意图：



图中自底向上依次为：

- **基础层**：提供网络通信、交易构造及签名、合约编译等基础功能；
- **API层**：基于基础层提供的功能，对FISCO BCOS常用功能进行了进一步地封装，暴露出API用于给上层的应用层调用。这些API覆盖基本的JSON RPC功能（查询链状态、部署合约等）及预编译合约功能（权限管理、CNS服务等）；
- **应用层**：使用Node.js SDK进行二次开发的应用程序均属于此层，如Node.js SDK自带的CLI（Command-Line Interface，命令行界面）链管理工具。

二、SDK的安装

2.1 环境准备

Node.js SDK依赖下列软件：

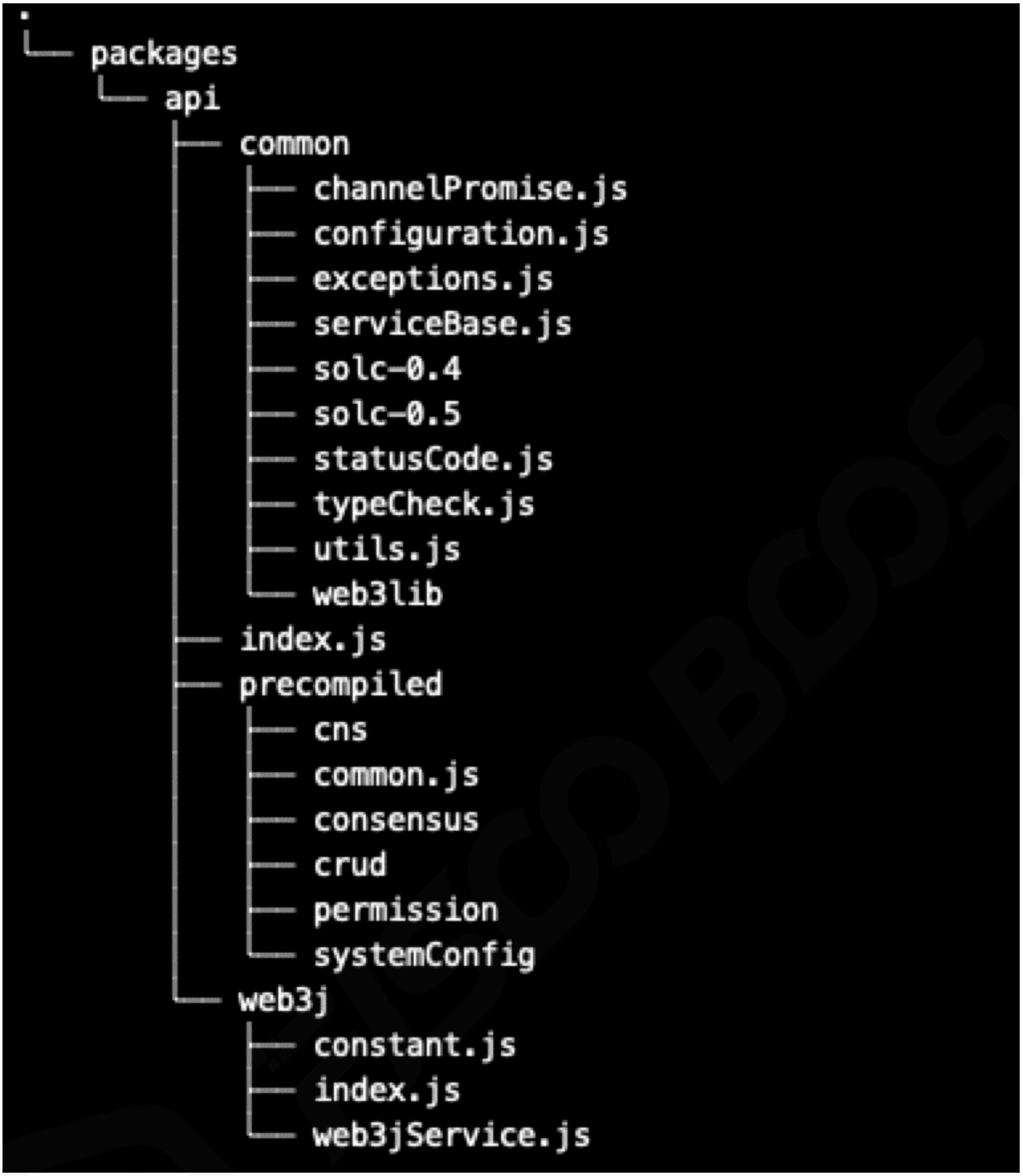
- 8.10.0或以上的版本的Node.js，5.6.0或以上的版本的NPM；

- Python2、g++及make。Solidity编译器solc需要编译后才能使用，编译所需基础软件需要用户提供，其中Python2用于运行构建工具node-gyp，g++和make则是用于编译solc。对于没有g++和make的Windows用户，可以安装 windows-build-tools再进行构建。

2.2 安装SDK

- 从https://github.com/FISCO-BCOS/nodejs-sdk下载sdk；
- 进入nodejs-sdk目录；
- 执行npm i。Node.js SDK使用lerna管理依赖项，此步用于安装lerna；
- 执行npm repoclean。初次安装时无需执行该命令，但如果出现了安装到一半被打断等情况，建议先执行该命令以清除所有依赖项；
- 执行npm run bootstrap，此命令会安装SDK的所有依赖项。

上述所有命令执行完后，API层的目录结构如下图所示：



对于应用开发者，可以重点关注 nodejs-sdk/packages/api/web3j 及 nodejs-sdk/packages/api/precompiled目录，SDK提供的所有API均位于这两个目录下的模块中，应用开发过程中通过require语句导入相应模块即可。

三、使用SDK进行开发

3.1 初始化

使用Node.js SDK的所有功能前，必须对SDK进行配置，配置以.json格式配置文件的形式提供。主要包括以下配置项：

- **privateKey**：FISCO BCOS基于公私钥体系，每个账户（公钥）均有一个对应的私钥，SDK需要使用该私钥对交易进行签名；

- **nodes**：SDK连接的节点，可以有多个，当节点数大于1时，SDK的每次请求都会随机从nodes调出一个节点发送；

- **authentication**：SDK使用Channel协议与节点进行通信，Channel使用SSL安全传输协议传输数据，在双方建立通信前，需要进行必要的身份验证。因此这个配置项中需要指明SDK私钥文件、证书文件、CA根证书文件的路径，这些文件通常是在生成链的阶段自动生成；

- **groupID**：FISCO BCOS使用多群组架构，同一个节点可以隶属于多个群组，因此需要指明SDK需要连接的链的群组ID；

- **timeout**：受限于网络环境，SDK的请求有可能超时，因此可能会导致调用SDK接口的程序陷入无尽等待，因此需要指定一个超时时间，SDK在请求超时后强制返回并向调用者返回错误。

开发者需要在初始化阶段将配置文件载入Configuration对象中，Configuration对象是全局唯一的且被所有模块共享。

3.2 调用示例

以一个简单的获取当前块高的程序为例，调用步骤如下：

1、初始化Configuration对象，将配置文件路径传入Configuration对象的setConfig函数：

```
const Configuration = require('./nodejs-sdk/packages/api/common/config
Configuration.setConfig(path.join(__dirname, args.config));
```

2、获取当前块高的API位于Web3jService中，构造一个该对象：

```
const Web3jService = require('./nodejs-sdk/packages/api').Web3jService
let web3jService = new Web3jService();
```

3、调用Web3jService的getBlockNumber接口，获取返回值并在控制台中输出：

```
web3jService.getBlockNumber().then(blockNumber => {
        console.log(blockNumber)
    });
```

需要注意的是，上述代码中使用了Promise.protoype.then方法。Promise正如其名字所示，封装了一个异步的操作，并且“承诺”在这个操作结束后，一定会调用用户在then或者catch中指定的回调函数。

由于Node.js天然支持异步的特性，Promise的概念在Node.js SDK处处存在（细心的读者可能已经注意到，Node.js SDK基础层中负责Channel通信的模块叫做channelPromise）。Node.js SDK的API调用约定是：所有的API调用时均返回Promise对象，开发者需要使用await或then...catch...方法才获取到调用结果。因此开发者在调用API时需要小心，如果直接使用了API的返回值，会很容易导致bug。

四、使用CLI工具

Node.js SDK除了提供API外，还提供了一个小巧的CLI工具供用户直接在命令行中对链进行操作，同时CLI工具也是一个展示如何使用Node.js SDK进行二次开发的示例。

CLI工具位于packages/cli目录下，若需要使用则需要进入该目录并执行./cli.js脚本，使用之前同样需要进行配置，配置文件位于packages/cli/conf/config.json文件中。以下给出了几个使用示例：

1、查看所连节点的版本：

```
lichenxi@lichenxideMacBook-Pro ~/Work/nodejs-sdk/packages/cli master ● ./cli.js getClientVersion | jq
{
  "id": 1,
  "jsonrpc": "2.0",
  "result": {
    "Build Time": "20191023 02:28:52",
    "Build Type": "Darwin/appleclang/Debug",
    "Chain Id": "1",
    "FISCO-BCOS Version": "2.1.0",
    "Git Branch": "performance",
    "Git Commit Hash": "d2a183b7110e70e486a0d4f8e7f70b48fbc9a31f",
    "Supported Version": "2.0.0"
  }
}
```

2、获取当前块高：

```
lichenxi@lichenxideMacBook-Pro ~/Work/nodejs-sdk/packages/cli master ● ./cli.js getBlockNumber| jq
{
  "id": 1,
  "jsonrpc": "2.0",
  "result": "0x5"
}
```

3、部署合约，部署之前合约需要放置于packages/cli/contracts目录中。

```
lichenxi@lichenxideMacBook-Pro ~/Work/nodejs-sdk/packages/cli master ● ./cli.js deploy HelloWorld | jq
{
  "contractAddress": "0x861b700f6a86baa8d31d61518149add8e62d770a",
  "status": "0x0"
}
```

4、调用HelloWorld合约的get方法：

```
lichenxi@lichenxideMacBook-Pro ~/Work/nodejs-sdk/packages/cli master ● ./cli.js call HelloWorld 0x861b700f6a86baa8d31d61518149add8e62d770a get | jq
{
  "status": "0x0",
  "output": {
    "0": "Hello, World!"
  }
}
```

五、Node.js SDK的未来需要你

当前，Node.js SDK还在成长，在某些地方仍然需要进一步打磨，比如需要CLI工具能够解析SQL语句，或者SDK的性能需要优化……

秉承开源的精神，我们相信社区的能量能够将Node.js SDK 变得更加方便易用，欢迎广大开发者

踊跃在issue或PR中贡献自己的idea和力量！

FISCO BCOS

FISCO BCOS的代码完全开源且免费

下载地址↓↓↓

https://github.com/FISCO-BCOS/FISCO-BCOS