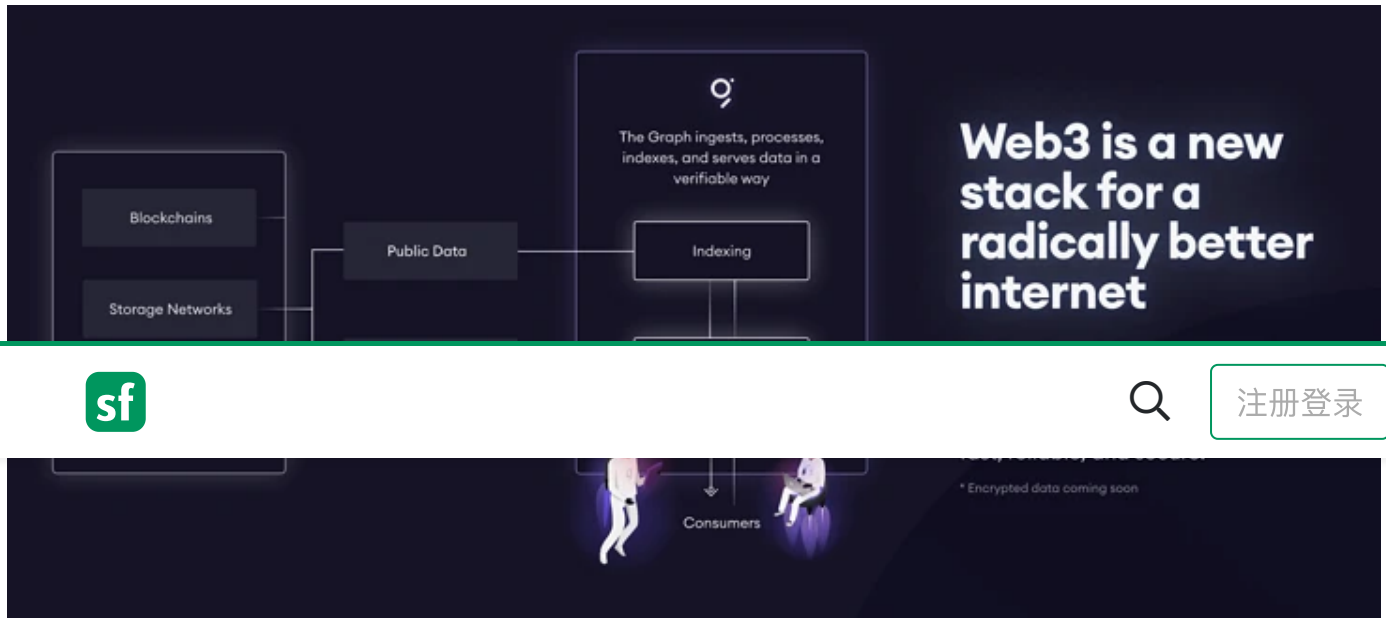
The Graph介绍

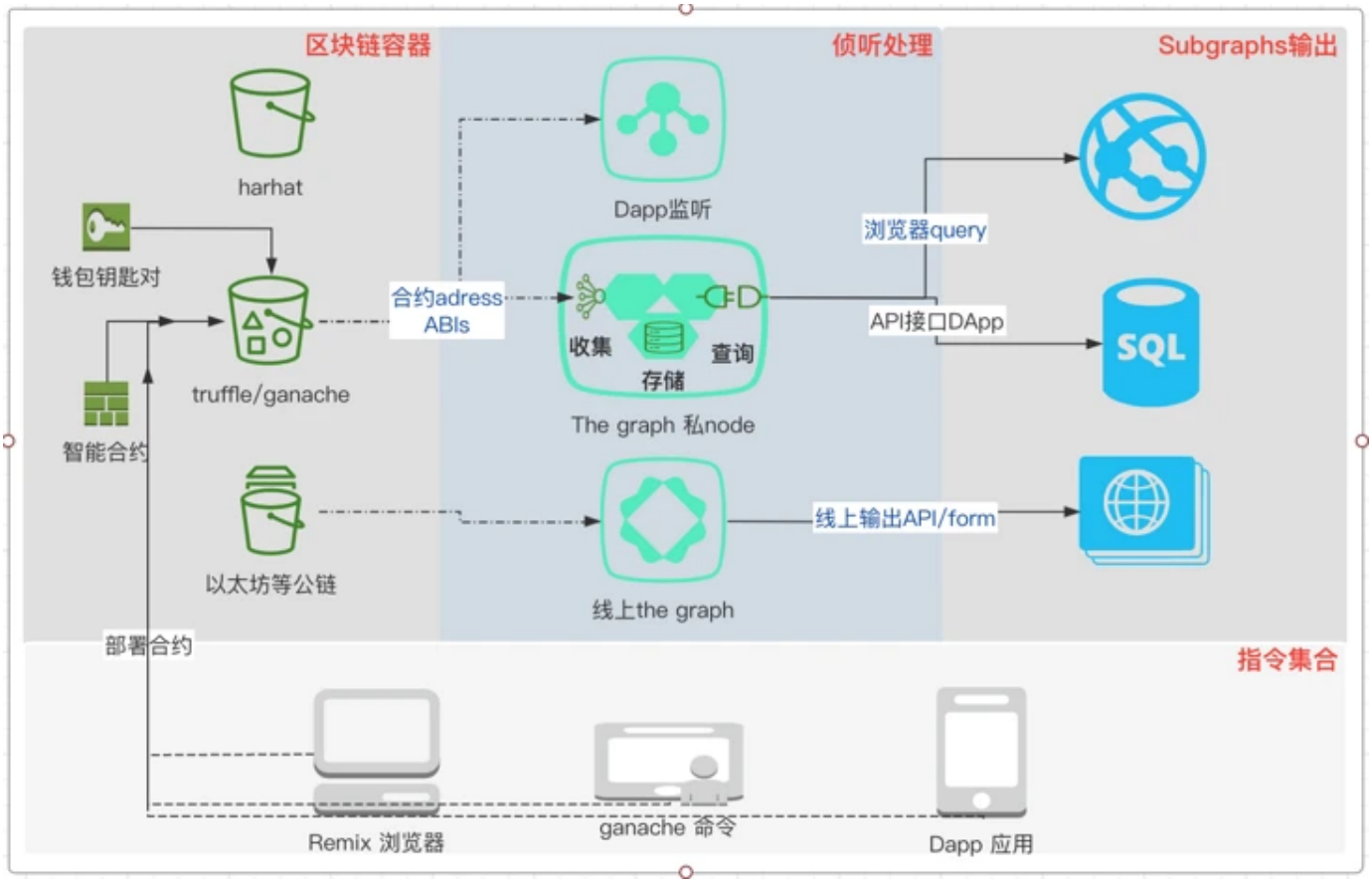
blockgeek 发布于 2021-07-05

The Graph是用于索引和查询区块链数据的去中心化协议。建立subgraphs用标准 GraphQL API 查询这些索引解决了去中心化数据遍历的痛苦。



海量的去中心化数据难以统计计算，现通过一个为区块链数据提供索引的主机服务解决了这个问题。 使用标准 GraphQL API 查询这些索引（“subgraphs”）。 未来，主机服务将演变为具有相同功能的完全去中心化的协议。 两者都由 Graph Node 的开源实现提供支持，它提供了一种用于查询以太坊和IPFS网络数据的公开透明的解决方案，任何人都可以基于其提供的开放API创建并发布索引数据，即subgraph。

本篇入门文档通过一个简单交易量统计例子，带你领略一番如传统数据库信息汇聚之旅（收集与抓取），当具有THe Graph知识储备后，我们就开始吧。



主要研究路线（一路私服到底）

1. **智能合约**被**truffer cmd** 发布到**truffle/ganache** 容器（私链）
2. **合约adress&abis**与**thegraph节点**（私node）
3. markdemo1工程（the graph）监听，收集数据，处理数据，包装通用查询
4. the graph输出查询接口到**Dapp2**（8000->3000）**/query page**（8001）
5. **Dapp2 调用合约**交易触发the graph输出变化

辅助输出：

1. the graph subgraphs官网浏览https://thegraph.com/explorer 个人subgraphs中心 https://thegraph.com/explorer/dashboard
2. truffle/ganache 对智能合约管理（部署、调试）
3. Rimax 浏览器版对合约操作
4. Dapp2对智能合约的操作
5. Dapp1对智能合约的监听
6. truffle/ganache 、ganache cl、the graph私节点 部署和使用
7. 各种网络、本机环境、脚本替换踩过的坑（坑就是是为后来者节约时间）
8. 发布一个线上subgraph到https://thegraph.com/explorer

在 https://thegraph.com/explorer 上有很多subgraph，让我们来看看uniswap-v2

Github	**https://github.com/Uniswap**	**发布地址**
ID	QmWTrJJ9W8h3JE19FhCzzPYsJ2tgXZCdUqnbyuo64ToTBN	subgraph
Queries (HTTP)	https://api.thegraph.com/subg...	大量可体验数据查询

uniswap-v2

打开浏览器体验一下先

2输入查询功能、参数和条件【点击紫色按钮】—>到节点数据中心3里->4.查询出期望结果；当让2这个通用API也可以通过编程方式来调用，传入不同参数，体验下GraphQL API 查询

{ pairs{id}}	{ pairs(where: {id:"0x00004ee988665cdda9a1080d5792cecd16dc1220"}){id}}	{ pairs(orderBy:id){id}}

The graph 用subgraph跟踪智能合约

数据抓取/事件监控数据汇聚及查询

获取存储展示线上智能合约的subgraph简单例子

1. 抓当前时间到上次update时间内的记录
2. 存交易大于2的记录
3. 取某客户所以交易记录的条数和汇总金额
4. 以上仅在想象中（真实the graph 就是踩好坑）

按 the graph 思路来
1) 系统初始化把链上最近的区块拉下了，定时取最近区块
2)subgraph的监听事件（原理就是event监听通通拉下数据，所以1，2，3就无意义）是拉下所有数据
3)让我们来看看我踩过的那些坑

- mac机器环境遗留问题brew upgrade 时文件找不到，临时解决找到一个同目录相似文件替换（一直睡不着觉，后找到方案见附录brew upgrade安装报错问题解决替换homebrew-bottles）代理端口8001和the graph 系统端口冲突，改代理
- yarn，git的代理和镜像要不停切换来应对众所周知的网络现状
- docker 里the graph访问外面truffle/ganache系统配置，脚本生成一个访问不可达ip172.18.2.0，修改为本机IP 192.168.0.136（不能使用127.0.0.1补脑docker访问localhost回到哪里？）
- sel 命令第二次替换合同号要用第一次的合同号：例子程序中有2合同，可能都会再修改一次
- truffle/ganache每次启动会生成不同的账号序列，所以不同时间导入合同的owner不同的，不要记第一个账号，会坑死人（可以用合同的历史交易记录查看owner，见markdemo2）
  4) 写subgraph 需要注意或已经完成的脚手架
- **id:** ID**! 要求每个实体都有一个非空的不重复主键ID**
- dataSources数据源配置初始读取块跳过合同发布之前的那些块：startBlock: 6627917

发布subgraph例子markdemo1

先要部署容器发布一个智能合约（可参见Hardhat + VS code，本篇使用Ganache部署）

Step 1: Ganache and Required Parameters

```
$ ganache-cli -h 0.0.0.0
```

例如：ganache-cli -h 0.0.0.0 -v /db -b 6 -a 8 -e 1000 -d expect chair toe trade spider wedding say item scare fog shrimp garlic

打开存在库，设置自动挖矿时间6，设置8个1000gas账号，导入现有私钥，（-d ，可以是外面，也可用-m产生）启动说明（常用-account –db –blockTime）
相信原文，理解更符合你的口味，技术文档每个人理解的角度不一样关注的重点不一样，用心点

–account： 指定账户私钥和账户余额来创建初始测试账户。可多次设置
请注意，私钥长度为64个字符，并且必须以0x前缀的十六进制字符串形式输入。余额可以输入为一个整数或0x前缀的十六进制值，指定该帐户中wei的数量。
使用--account时，不会为您创建HD钱包。
-u or --unlock: 指定 - unlock...多次传递地址或帐户索引以解锁特定帐户。当与 -- secure一起使用时，--unlock将覆盖指定帐户的锁定状态。
$ ganache-cli --secure --unlock "0x1234..." --unlock "0xabcd..."
-b 或r –blockTime： 指定自动挖矿的blockTime，以秒为单位。默认值为0，表示不进行自动挖矿？？一交易一块（ganache will instantly mine a new block for every transaction）。

- -a 或 –accounts： 指定启动时要创建的测试账户数量。
- -e 或 –defaultBalanceEther： 分配给每个测试账户的ether数量，默认值为100。
- -b 或r –blockTime： 指定自动挖矿的blockTime，以秒为单位。默认值为0，表示不进行自动挖矿。
- -d 或 –deterministic： 基于预定的助记词（mnemonic）生成固定的测试账户地址。
- -n 或 –secure： 默认锁定所有测试账户，有利于进行第三方交易签名。
- -m 或 –mnemonic： 用于生成测试账户地址的助记词。
- -p 或 –port： 设置监听端口，默认值为8545。
- -h 或 –hostname： 设置监听主机，默认值同NodeJS的server.listen()。
- -s 或 –seed： 设置生成助记词的种子。
- -g 或 –gasPrice： 设定Gas价格，默认值为20000000000。
- -l 或 –gasLimit： 设定Gas上限，默认值为90000。
- -f 或 –fork： 从一个运行中的以太坊节点客户端软件的指定区块分叉。输入值应当是该节点的HTTP地址和端口，例如**http://localhost:8545**。 可选使用@标记来指定具体区块，例如：http://localhost:8545@1599200。
- -i 或 –networkId：指定网络id。默认值为当前时间，或使用所分叉链的网络id。
- –db： 设置保存链数据的目录。如果该路径中已经有链数据，ganache-cli将用它初始化链而不是重新创建。
- –debug：输出VM操作码，用于调试。
- –mem：输出ganache-cli内存使用统计信息，这将替代标准的输出信息。
- –noVMErrorsOnRPCResponse：不把失败的交易作为RCP错误发送。开启这个标志使错误报告方式兼容其他的节点客户端，例如geth和Parity。

```
-a or --accounts: Specify the number of accounts to generate at startup.
-e or --defaultBalanceEther: Amount of ether to assign each test account. Defaul
-b or --blockTime: Specify blockTime in seconds for automatic mining. If you dor
-d or --deterministic: Generate deterministic addresses based on a pre-defined m
-n or --secure: Lock available accounts by default (good for third party transac
-m or --mnemonic: Use a bip39 mnemonic phrase for generating a PRNG seed, which
-p or --port: Port number to listen on. Defaults to 8545.
```

```
  -g or --gasPrice: The price of gas in wei (defaults to 20000000000)
  -l or --gasLimit: The block gas limit (defaults to 0x6691b7)
  --callGasLimit: Sets the transaction gas limit for eth_call and eth_estimateGas
  -k or --hardfork: Allows users to specify which hardfork should be used. Support
  -f or --fork: Fork from another currently running Ethereum client at a given blo
  forkCacheSize: number - The maximum size, in bytes, of the in-memory cache for c
  -i or --networkId: Specify the network id ganache-cli will use to identify itsel
  --chainId: Specify the Chain ID ganache-cli will use for eth_chainId RPC and the
  --db: Specify a path to a directory to save the chain database. If a database al
  --debug: Output VM opcodes for debugging
  --mem: Output ganache-cli memory usage statistics. This replaces normal output.
  -q or --quiet: Run ganache-cli without any logs.
  -v or --verbose: Log all requests and responses to stdout
  -? or --help: Display help information
  --version: Display the version of ganache-cli
  --account_keys_path or --acctKeys: Specifies a file to save accounts and private
  --noVMErrorsOnRPCResponse: Do not transmit transaction failures as RPC errors. F
```

Step 2: Running a local Graph node

在～graph-node/docker目录下

```
$ docker-compose up;
```

或

```
#!/bin/bash

docker-compose down -v;

if [ -d "data" ]
then
  echo "Found old data for the graph node - deleting it";
  # we need to sudo this to remove system locked files
  sudo rm -rf data/;
fi

docker-compose up;
```

graph-node_1 | Jun 24 01:34:13.558 WARN Trying again after net_version RPC call failed (attempt #18) with result Err(Transport error: Error(Connect, Os { code: 111, kind:

graph-node_1 | Jun 24 01:34:14.517 ERRO Connection to provider failed. Not using this provider, error: deadline has elapsed, provider: mainnet-rpc-0

该报错会引起第三步（Step 3）报错：

Failed to deploy to Graph node http://127.0.0.1:8020/: ***Ethereum network not supported by registrar: mainnet.***
解决：ConnectionRefused一般是对方服务器不能到达
找到docker/docker-compose.yml-e 的ethereum: 'mainnet:http://host.docker.internal:8545'
修改为本地：ethereum: 'mainnet:http://192.168.0.136:8545'

Step 3-0: 发布markdemo1合同

```
$ truffle compile
$ truffle migrate
```

Migrations：0x83Ad4160F00259D6D329c09A1436386a706e3818
GravatarRegistr：0xE54bA45F29b4247D75e86fD7A83a1E44160610D2

```
$ sed -i -e  's/0x2E645469f354BB4F5c8a05B3b30A929361cf77eC/0x83Ad4160F00259D6D32
```

Step 3: Deploying to your local Graph Node

```
$ yarn create-local
$ yarn build && yarn deploy-local
```

最后例子输出效果如下： （http://127.0.0.1:8000/subgraphs/name/moluoping/markg）

Step 4: Dapp 调用

```
$ git clone https://github.com/graphprotocol/ethdenver-dapp/
$ echo 'REACT_APP_GRAPHQL_ENDPOINT=http://localhost:8000/subgraphs/name/moluopir
```

监听智能合约交易事件subgraph例子markdemo2

上一章节简单说明了一个子graph发布过程，下面将在Dapp工程中添加下面代码，演示整个智能合约事件监听过程

```
var Web3 = require("web3")
var web3;
if (typeof web3 !== 'undefined') {
    web3 = new Web3(web3.currentProvider);
} else {
    web3 = new Web3(new Web3.providers.WebsocketProvider("ws://127.0.0.1:8545"))
}
web3.eth.defaultAccount = '0x4386997160134D4a67FD6A14DE7f924315D6F0A4';
console.log('defaultAccount:' + web3.eth.defaultAccount)
var contractAbi = [{"constant":false,"inputs":[{"name":"_imageUrl","type":"strir
//Abi data from
// truffle/ganache: markdemo1/abis/Gravity.json
// Remix: web https://remix.ethereum.org/  tab 'SOLIDITY COMPILER'->bottom 'abi'
var contractAbiTruffle=contractAbi
var contractAbiRemix=contractAbi

var contractaAddressRemix ='0x4ef9BEb56eB0Ab92CDeD01Eff9A60554aE594d45'
// Remix: web https://remix.ethereum.org/  tab 'DEPLOY & RUN TRANSACTIONS'->bott
var contractaAddressTruffle ='0x38FcB3d9c3C8958373D61B702049D695A6725AFE'
//$ truffle migrate

{
    var MyContract = new web3.eth.Contract(contractAbiRemix, contractaAddressRem
    MyContract.methods.updateGravatarName('aaaa').send({from: "0x4386997160134D4
        .then(console.log);
    //from data: this app project http://localhost:3000/F12 history event data r
```

1. 上个例子中truffle/ganache环境中在$truffle migrate 可见智能合约地址，Abi数据可以在markdemo1工程markdemo1/abis/Gravity.json获取，但是初学者先用Remix体验。
2. 看上面代码
   然后就出来了，数据终于跑起来了.....

subgraph 在the graph市场上线

1. github网站注册账号
上github网站登录（无账号注册新账号）

2. 获取thegraph网站令牌
在 https://thegraph.com/explorer/dashboard/ 关联登录后，获取tocken

3. 创建subgraphgraph

```
graph init --from-example moluo..../markdemo3
```

graph init --from-contract <CONTRACT_ADDRESS> \ [--network <ETHEREUM_NETWORK>] \ [--abi <FILE>] \ <GITHUB_USER>/<SUBGRAPH_NAME> [<DIRECTORY>]
moluo.... github账号名/markdemo3 subgraph名字 创建工程并编译和产生相关资源

✔**Subgraph name**·moluo..../markdemo3
✔**Directory to create the subgraph in**·markdemo3
———
✔ Cloning example subgraph
✔ Update subgraph name and commands in package.json
✔ Initialize subgraph repository
✔ Install dependencies with yarn
✔ Generate ABI and schema types with yarn codegen
可以从以上脚本运行看编译过程

4. 发布到线上
a. 用令牌登thegraph开发中心

```
$ graph auth https://api.thegraph.com/deploy/880d70986bfc4172902641d870d96f37
```

graph auth https://api.thegraph.com/deploy/ <access-token>
<access-token>为第2步的获取tocken 注意tocken与前面有个空格，这是两个参数

```
#如果报The original error was: Cannot find module 'keytar'
```

b. 创建subgraph名字
https://thegraph.com/explorer/dashboard/

subgraph首字母大写，其他随意。

c. 将你的subgraph发布到thegraph.com

```
$ cd markdemo3
$ yarn deploy
```

如果没有做b步骤，将报错
You may need to create subgraph at https://thegraph.com/explorer...
error Command failed with exit code 1.
c.如果一起顺利，你可以看到：
https://api.thegraph.com/subgraphs/name/moluoping/markdemo3

例子解析（聚焦subgraph的markdemo4）

该例子直接从容器中合同向导生成subgraph工程：

```
$ graph init --from-contract 0x1e1215caD01aD7192832e0DACfA930Caf0132b43  --netwo
```

智能合约是从Remix中部署的，在Remix页面的编译页面可以copy到contract adress，部署页面可以copy到abi并保存成文件，graph init会生成功能必要的脚手架文件，并把合同的第一个事件监听函数作为查询实体自动代码生成（其他智能合约函数被注释）

这个工程就简洁很多，因为没有智能合约及智能合约部署相关文件，下面简单介绍下：
1) subgraph.yaml 本例address，abi来源于命令行根据未来设置，startBlock定义监控链上开始区块
source:
address: '0xc0a47dFe034B400B47bDaD5FecDa2621de6c4d95'
abi: Contract
startBlock: 88888

2) schema.graphql定义可查询的实体
现在好像都是可被GraphGl查询的实体，不可查询的实体好像还没开放
ID必须string字段
GraphGl是图数据库标准，所以多看看多对多关系。

3) mapping.ts 事件处理实现
配置在subgraph.yaml
eventHandlers:

- **event:** Transfer(indexed address,indexed address,uint256)
  handler: handleTransfer
- **event:** Burn(indexed address,uint256)
  handler: handleBurn
  file: ./src/mapping.ts
  event 变量包含了合约相关（事件，合约函数）
  ExampleEntity "../generated/schema" 包含对the subgraph本地持久化相关

4) generated里向导产生的class
本地读写类schema.ts
智能合约函数类Contract.ts

5) 其他
package.json包含了subgraph从编译到本地部署、the graph部署的命令行
其他文件都是copy生成的

THe Graph基础知识

1. theGraph社区
   1) 官方
   https://www.thegraph.com/
   https://thegraph.com/docs/define-a-subgraph
   2) 社区入门：以太坊数据索引平台The Graph使用教程
   https://developer.aliyun.com/article/776668
2. 知识体系
   1) 深入了解The Graph（上
   https://zhuanlan.zhihu.com/p/196773044
   2) 深入了解The Graph（下
   https://thegraph.com/blog/the-graph-network-in-depth-part-2

https://blog.csdn.net/TurkeyCock/article/details/79165602

4) 以太坊测试网络Rinkeby使用教程（没钱买币但又想玩以太坊怎么办？用以太坊测试网络吧~~~~）

https://blog.csdn.net/wuhuimin521/article/details/85135610

3. 应用

1) 使用 TheGraph 完善Web3 事件数据检索

https://learnblockchain.cn/article/1589

原文链接：https://soliditydeveloper.com/thegraph

作者：MarkusWaas

2) 以太坊truffle+ganache合约部署调试及web3.js事件监听过程记录

https://www.e-learn.cn/topic/3716459

原文链接:https://my.oschina.net/u/4277087/blog/4325668

不羁岁月 提交于 2020-08-06 04:50:03

3) 全部教程web3.js 1.0中文手册

http://cw.hubwiz.com/card/c/web3.js-1.0/1/4/8/

https://github.com/bluketalk/sweb3

欢迎区块链行业志同道合的小伙伴添加小极微信，加入blockgeek区块链技术交流群，共同推动区块链技术普及和发展~

web 智能合约 graphql

本文系转载，阅读原文

http://blockgeek.com/129d/76de

阅读 3k • 更新于 2021-07-05

赞 收藏 分享

阿生 阅读 2.9k

NorthZeroCheat介绍

作者MiLi_CHENXI - NORTHZERO内容一项针对于我的世界启动器(网易)进行提升游戏体验的辅助工具箱...

北零 阅读 738

TypeScript介绍

typescript是由微软开发的一个javascript超集，本质是向javascript这门弱类型动态语言添加了静态类型和面...

新竹野猴子 阅读 2.7k