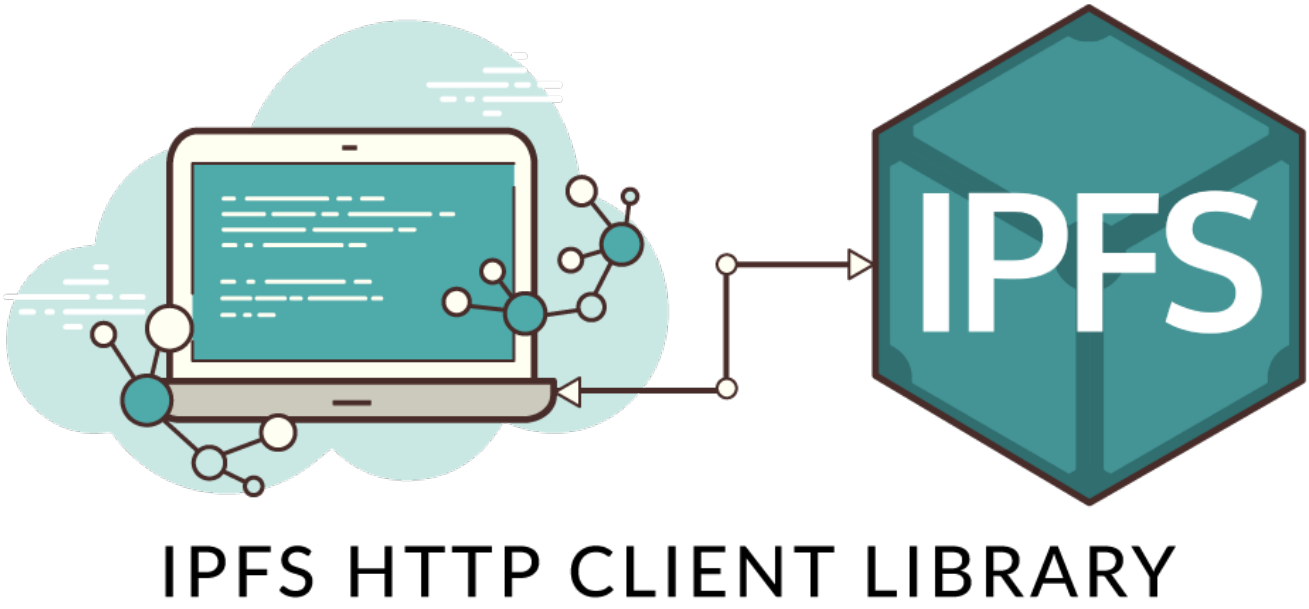
0005 - 【IPFS + 区块链 系列】 入门篇 - IPFS + Ethereum （下篇）-ipfs + Ethereum 大图片存储

> 区块链技术博客：http://liyuechun.org
> 区块链视频网站：http://www.kongyixueyuan.com

Ebay项目



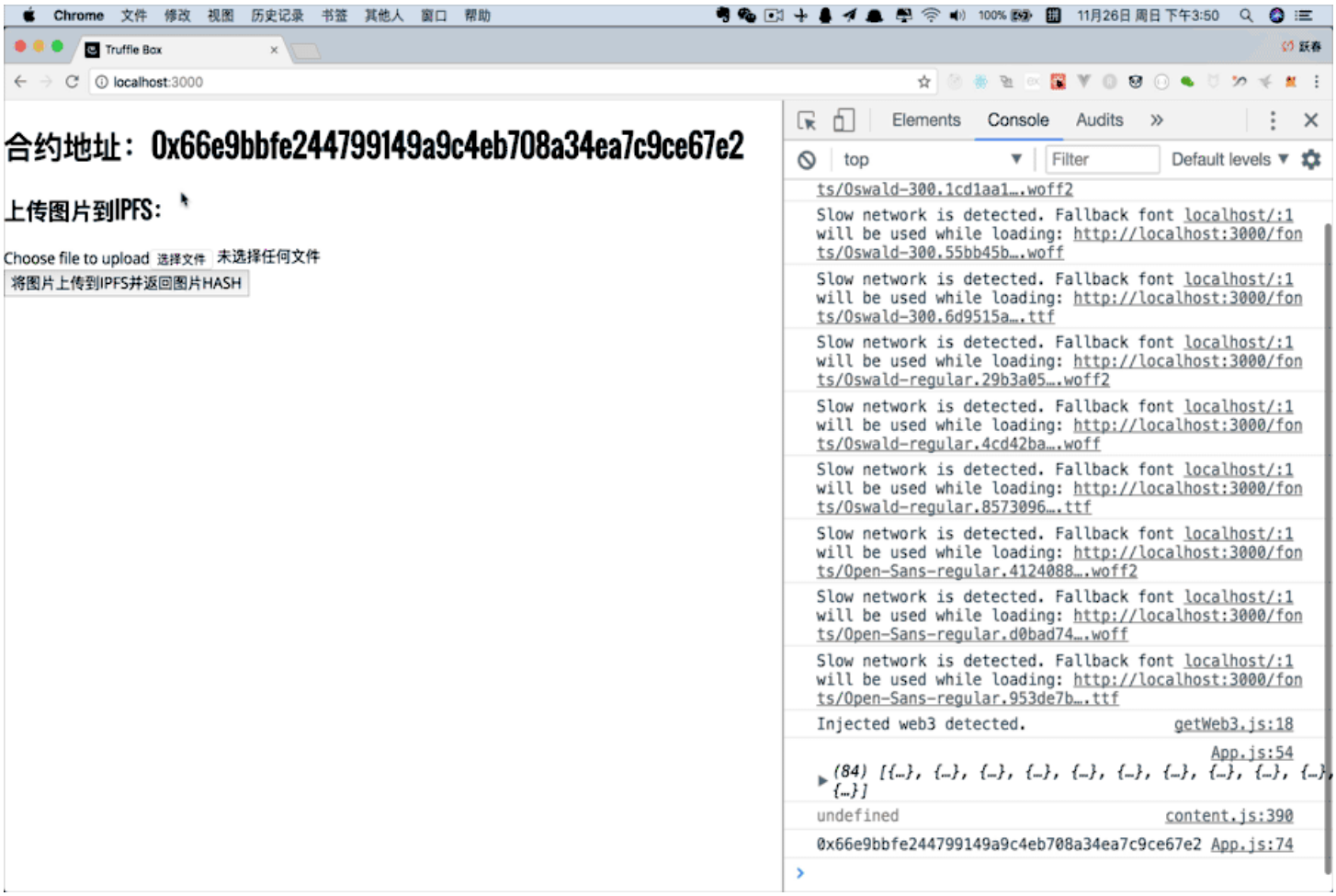
目录

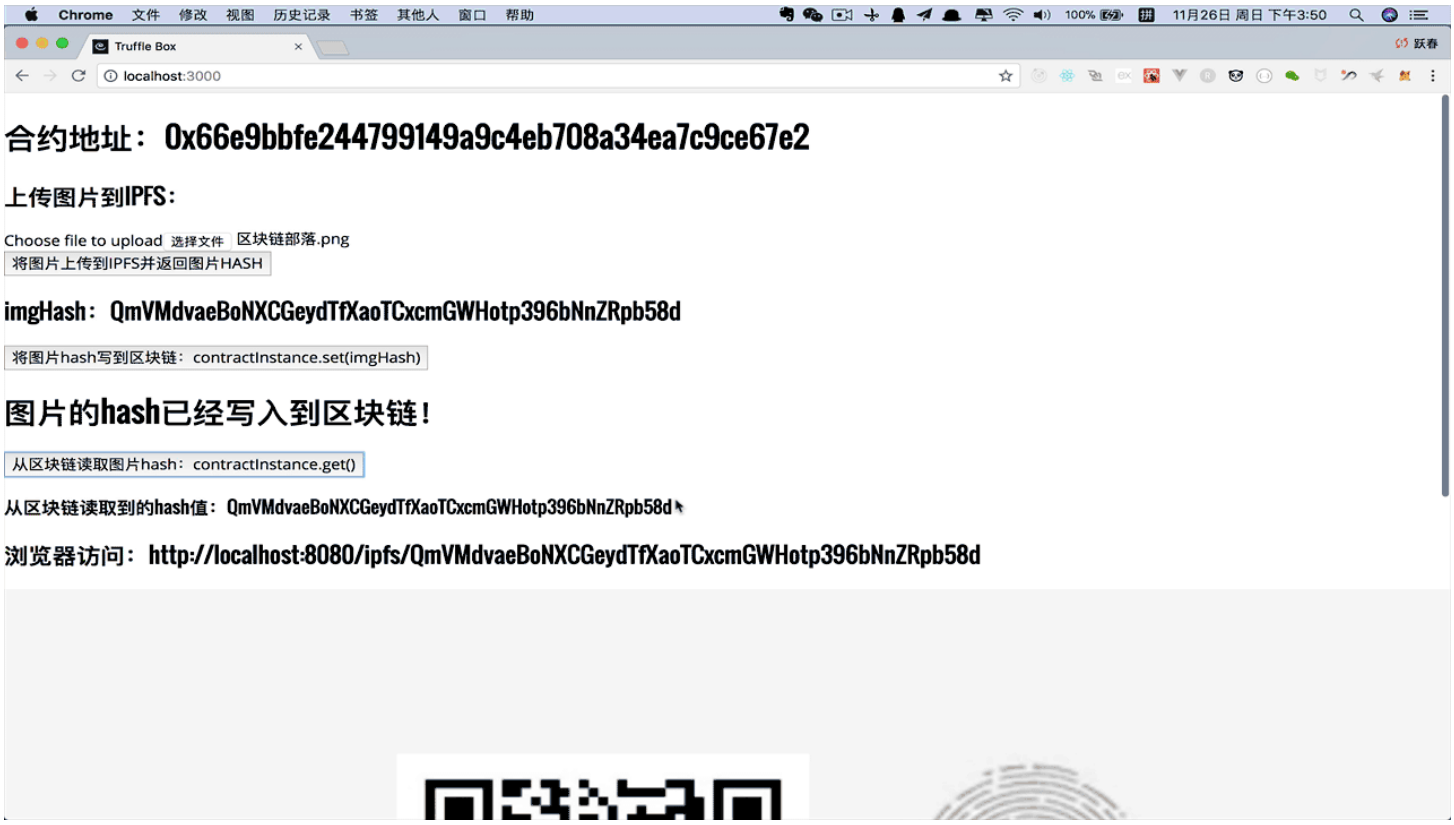
1. 系列文章

- 【IPFS + 区块链 系列】 入门篇 - IPFS环境配置
- 【IPFS + 区块链 系列】 入门篇 - IPFS+IPNS+个人博客搭建
- 【IPFS + 区块链 系列】 入门篇 - IPFS + Ethereum （上篇）-js-ipfs-api - 数据上传到IPFS
- 【IPFS + 区块链 系列】 入门篇 - IPFS + Ethereum （中篇）-js-ipfs-api - 图片上传到IPFS以及下载

2. 项目描述及效果展示

这篇文章通过`truffle unbox react`创建项目，安装`ipfs-api`，将图片存储到`ipfs`，将图片`hash`存储到`Ethereum`区块链，取数据时先从区块链读取图片`hash`，再通过`hash`从`ipfs`读取数据，解决了区块链大数据存储成本高昂的问题。





3. 阅读本文需要掌握的知识

阅读本文需要将先学习上面的系列文章，由于本文前端使用了大量的React语法，所以建议学习

一些React语法，还需要学习truffle framework。

4. 源码

其实这篇文章的内容就是上面几篇文章的综合结合体，所以在这里我将不再对代码做过多的概述。

```
import React, {Component} from 'react'
import SimpleStorageContract from '../build/contracts/SimpleStorage.json'
import getWeb3 from './utils/getWeb3'

import './css/oswald.css'
import './css/open-sans.css'
import './css/pure-min.css'
import './App.css'

const ipfsAPI = require('ipfs-api');
const ipfs = ipfsAPI({host: 'localhost', port: '5001', protocol: 'http'});

const contract = require('truffle-contract')
const simpleStorage = contract(SimpleStorageContract)
let account;

// Declaring this for later so we can chain functions on SimpleStorage.
let contractInstance;

let saveImageOnIpfs = (reader) => {
  return new Promise(function(resolve, reject) {
    const buffer = Buffer.from(reader.result);
    ipfs.add(buffer).then((response) => {
      console.log(response)
      resolve(response[0].hash);
    }).catch((err) => {
      console.error(err)
      reject(err);
    })
  })
}

class App extends Component {
  constructor(props) {
    super(props)

    this.state = {
      blockChainHash: null,
      web3: null,
```

```
      address: null,
      imgHash: null,
      isWriteSuccess: false
    }
  }

  componentWillMount() {

    ipfs.swarm.peers(function(err, res) {
      if (err) {
        console.error(err);
      } else {
        // var numPeers = res.Peers === null ? 0 : res.Peers.length;
        // console.log("IPFS - connected to " + numPeers + " peers");
        console.log(res);
      }
    });

    getWeb3.then(results => {
      this.setState({web3: results.web3})

      // Instantiate contract once web3 provided.
      this.instantiateContract()
    }).catch(() => {
      console.log('Error finding web3.')
    })
  }

  instantiateContract = () => {

    simpleStorage.setProvider(this.state.web3.currentProvider);
    this.state.web3.eth.getAccounts((error, accounts) => {
      account = accounts[0];
      simpleStorage.at('0x66e9bbfe244799149a9c4eb708a34ea7c9ce67e2').then((c
ontract) => {
        console.log(contract.address);
        contractInstance = contract;
        this.setState({address: contractInstance.address});
        return;
      });
    })

  }
  render() {
    return (<div className="App">
      {
        this.state.address
          ? <h1>合约地址：{this.state.address}</h1>
```

```
          : <div/>
      }
      <h2>上传图片到IPFS: </h2>
      <div>
        <label id="file">Choose file to upload</label>
        <input type="file" ref="file" id="file" name="file" multiple="multip
le"/>
      </div>
      <div>
        <button onClick={() => {
            var file = this.refs.file.files[0];
            var reader = new FileReader();
            // reader.readAsDataURL(file);
            reader.readAsArrayBuffer(file)
            reader.onloadend = function(e) {
              console.log(reader);
              saveImageOnIpfs(reader).then((hash) => {
                console.log(hash);
                this.setState({imgHash: hash})
              });

            }.bind(this);

          }}>将图片上传到IPFS并返回图片HASH</button>
      </div>
      {
        this.state.imgHash
          ? <div>
              <h2>imgHash: {this.state.imgHash}</h2>
              <button onClick={() => {
                  contractInstance.set(this.state.imgHash, {from: account}).
then(() => {
                    console.log('图片的hash已经写入到区块链! ');
                    this.setState({isWriteSuccess: true});
                  })
                }}>将图片hash写到区块链: contractInstance.set(imgHash)</button>
            </div>
          : <div/>
      }
      {
        this.state.isWriteSuccess
          ? <div>
              <h1>图片的hash已经写入到区块链! </h1>
              <button onClick={() => {
                  contractInstance.get({from: account}).then((data) => {
                    console.log(data);
                    this.setState({blockChainHash: data});
                  })
```

```
                }}>从区块链读取图片hash：contractInstance.get()</button>
            </div>
          : <div/>
      }
      {
        this.state.blockChainHash
          ? <div>
              <h3>从区块链读取到的hash值：{this.state.blockChainHash}</h3>
            </div>
          : <div/>
      }
      {
        this.state.blockChainHash
          ? <div>
              <h2>浏览器访问：{"http://localhost:8080/ipfs/" + this.state.imgH
ash}</h2>
              <img alt="" style={{
                  width: 1600
                }} src={"http://localhost:8080/ipfs/" + this.state.imgHash}/
>
            </div>
          : <img alt=""/>
      }
    </div>);
  }
}

export default App
```

5. 运行程序

- 项目下载

```
$ git clone https://github.com/liyuechun/IPFS-Ethereum-Image.git
$ cd IPFS-Ethereum-Image
$ npm install
```

- 运行程序

```
$ ipfs daemon // 终端一
$ npm start   // 终端二
```

6. 技术交流

- 区块链技术交流QQ群：`348924182`
- 进微信群请加微信：`liyc1215`
- 「区块链部落」官方公众号