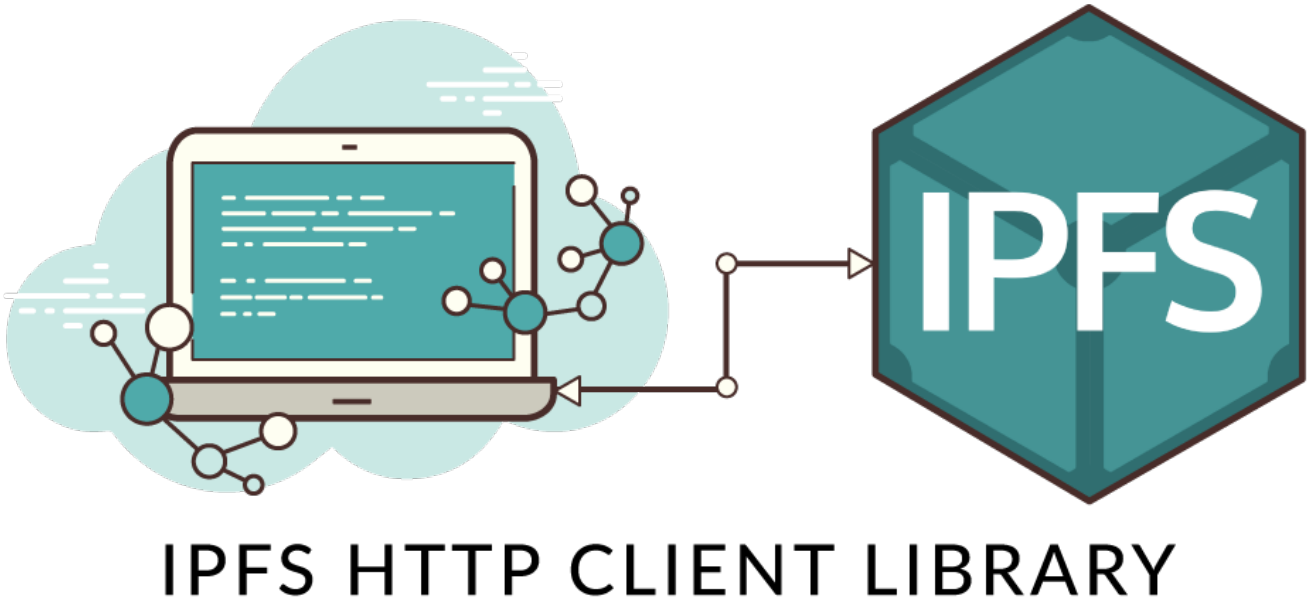
0003 -【IPFS + 区块链 系列】 入门篇 - IPFS + Ethereum （上篇）-js-ipfs-api

> 孔壹学院：国内区块链职业教育引领品牌。
>
> 作者：黎跃春，孔壹学院创始人，区块链、高可用架构师
>
> 微信：liyc1215
>
> 区块链博客：http://liyuechun.org

- 0003 -【IPFS + 区块链 系列】 入门篇 - IPFS + Ethereum （上篇）-js-ipfs-api
  - Ebay项目
  - 目录
  - 1. 内容简介
  - 2. IPFS-HTTP效果图
  - 3. 实现步骤
    - 3.1 安装create-react-app
    - 3.2 React项目创建
    - 3.3 运行React项目
    - 3.4 浏览项目
    - 3.5 安装 `ipfs-api`
    - 3.6 完成UI逻辑
    - 3.7 导入IPFS
    - 3.8 编写上传大文本字符串到IPFS的Promise函数
    - 3.9 上传数据到IPFS
    - 3.10 跨域资源共享CORS配置
    - 3.11 再次刷新网页提交数据并在线查看数据
    - 3.12 从IPFS读取数据
    - 3.13 总结
  - 4. 下篇文章预告
  - 5. 技术交流

Ebay项目



目录

- 1. 内容简介
- 2. IPFS-HTTP效果图
- 3. 实现步骤
  - 3.1 安装create-react-app
  - 3.2 React项目创建
  - 3.3 运行React项目
  - 3.4 浏览项目
  - 3.5 安装 `ipfs-api`
  - 3.6 完成UI逻辑
  - 3.7 导入IPFS
  - 3.8 编写上传大文本字符串到IPFS的Promise函数
  - 3.9 上传数据到IPFS
  - 3.10 跨域资源共享CORS配置
  - 3.11 再次刷新网页提交数据并在线查看数据
  - 3.12 从IPFS读取数据
  - 3.13 总结
- 4. 下篇文章预告

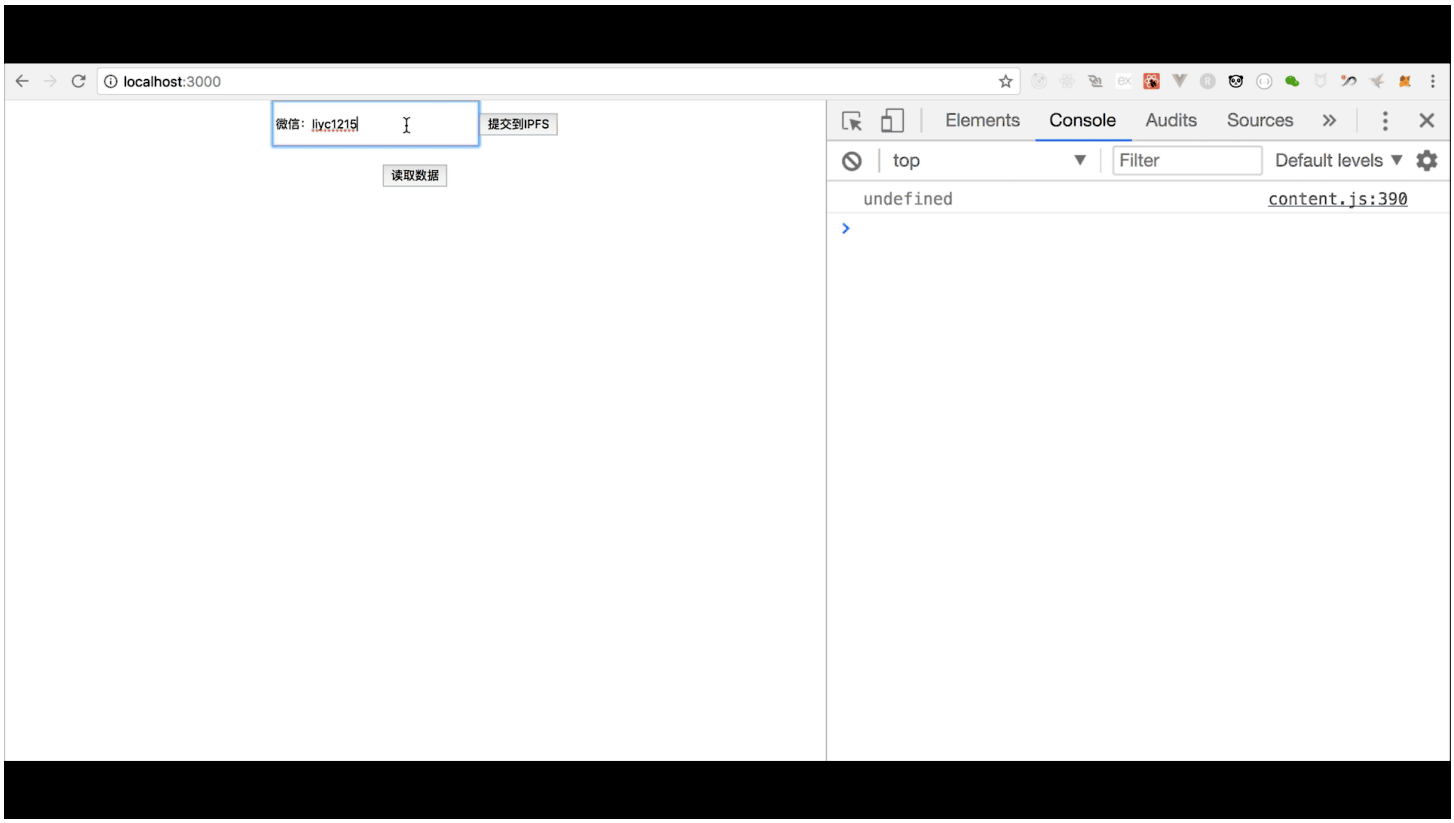
1. 内容简介

- 【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

在前面两篇文章中，第一篇春哥给大家详细介绍了 `IPFS` 环境配置，第二篇介绍了 `IPFS` 如何搭建个人博客，通过这两篇文章相信大家已经对 `IPFS` 有所了解，接下来的这篇文章，我们将为大家讲解 `js-ipfs-api` 的简单使用，如何将数据上传到 `IPFS`，以及如何从 `IPFS` 通过 `HASH` 读取数据。

2. IPFS-HTTP效果图



3. 实现步骤

3.1 安装create-react-app

参考文档： https://reactjs.org/tutorial/tutorial.html

```
localhost:1123 yuechunli$ npm install -g create-react-app
```

3.2 React项目创建

```
localhost:1123 yuechunli$ create-react-app ipfs-http-demo
```

```
localhost:ipfs-http-demo yuechunli$ ls
README.md   package.json    src
node_modules    public      yarn.lock
localhost:ipfs-http-demo yuechunli$
```

3.3 运行React项目

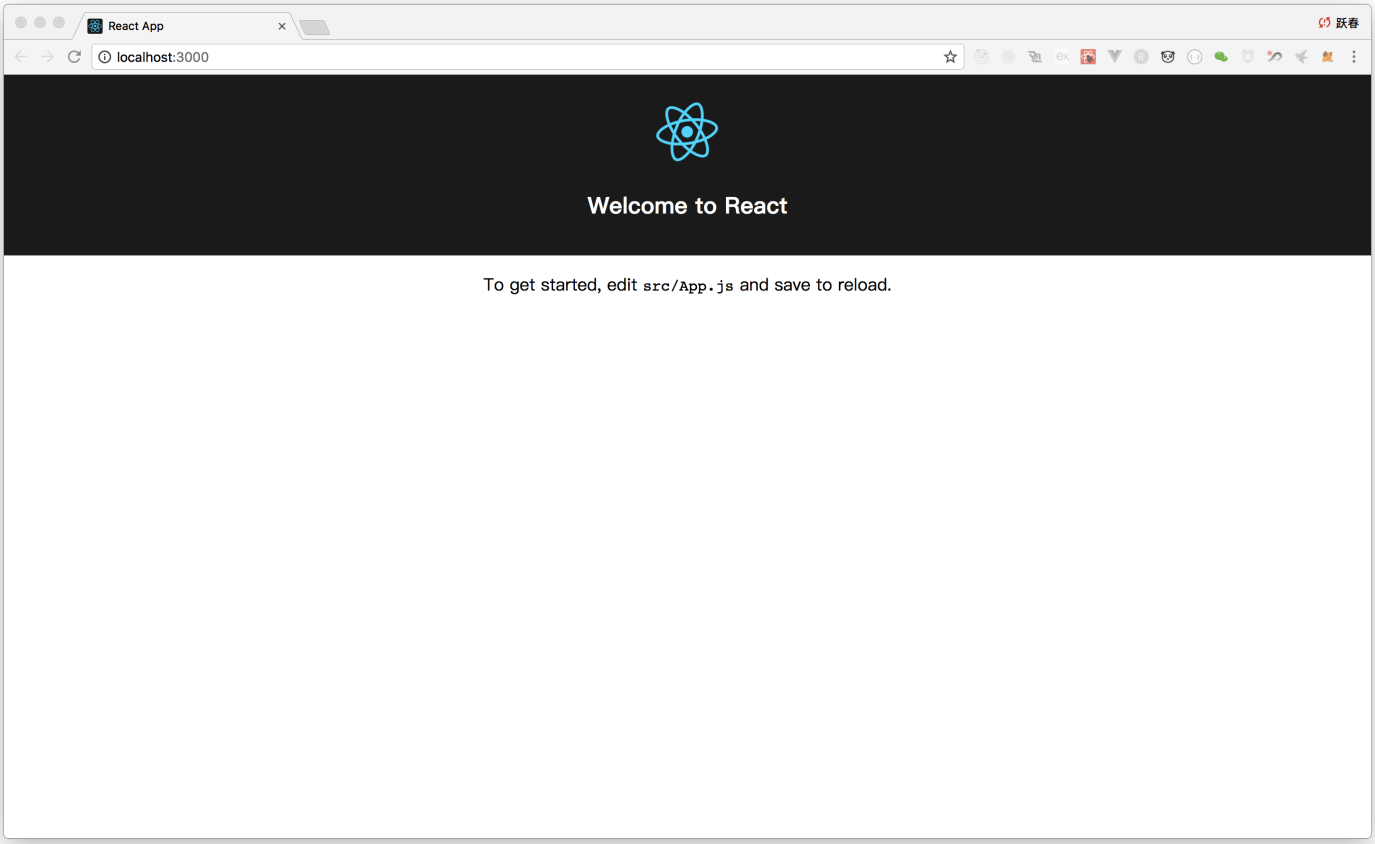
```
localhost:ipfs-http-demo yuechunli$ npm start
```

```
Compiled successfully!

You can now view ipfs-http-demo in the browser.

  Local:            http://localhost:3000/
  On Your Network:  http://192.168.0.107:3000/

Note that the development build is not optimized.
To create a production build, use yarn build.
```

3.4 浏览项目

浏览器浏览`http://localhost:3000`。

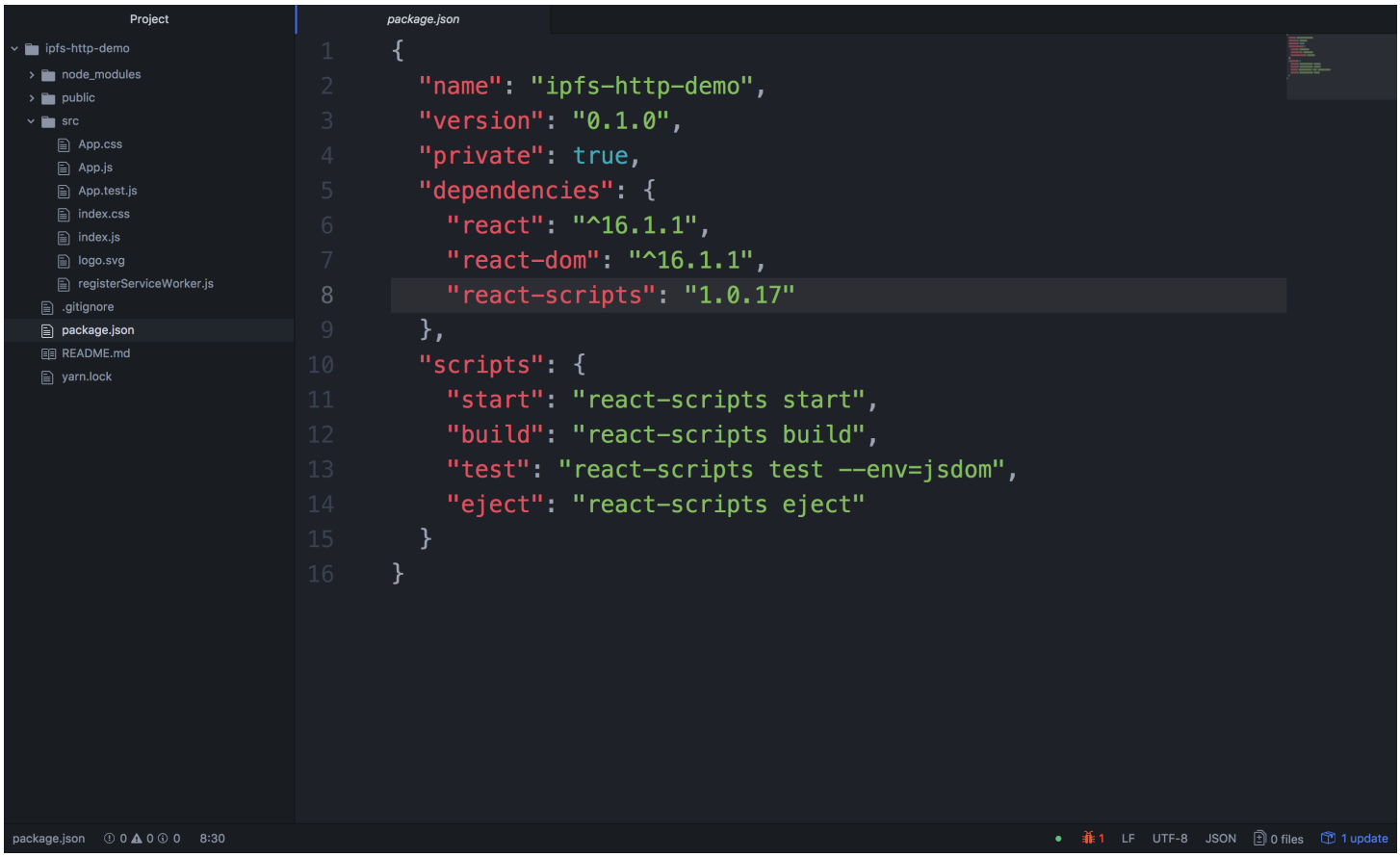
效果如下：



3.5 安装 `ipfs-api`

⚠️：在这里我就不过多的去介绍React的使用以及开发，如果感兴趣的可以去看这套React的视频，学完这套视频你可以直接进企业找React相关的前端开发工作。

- 项目结构

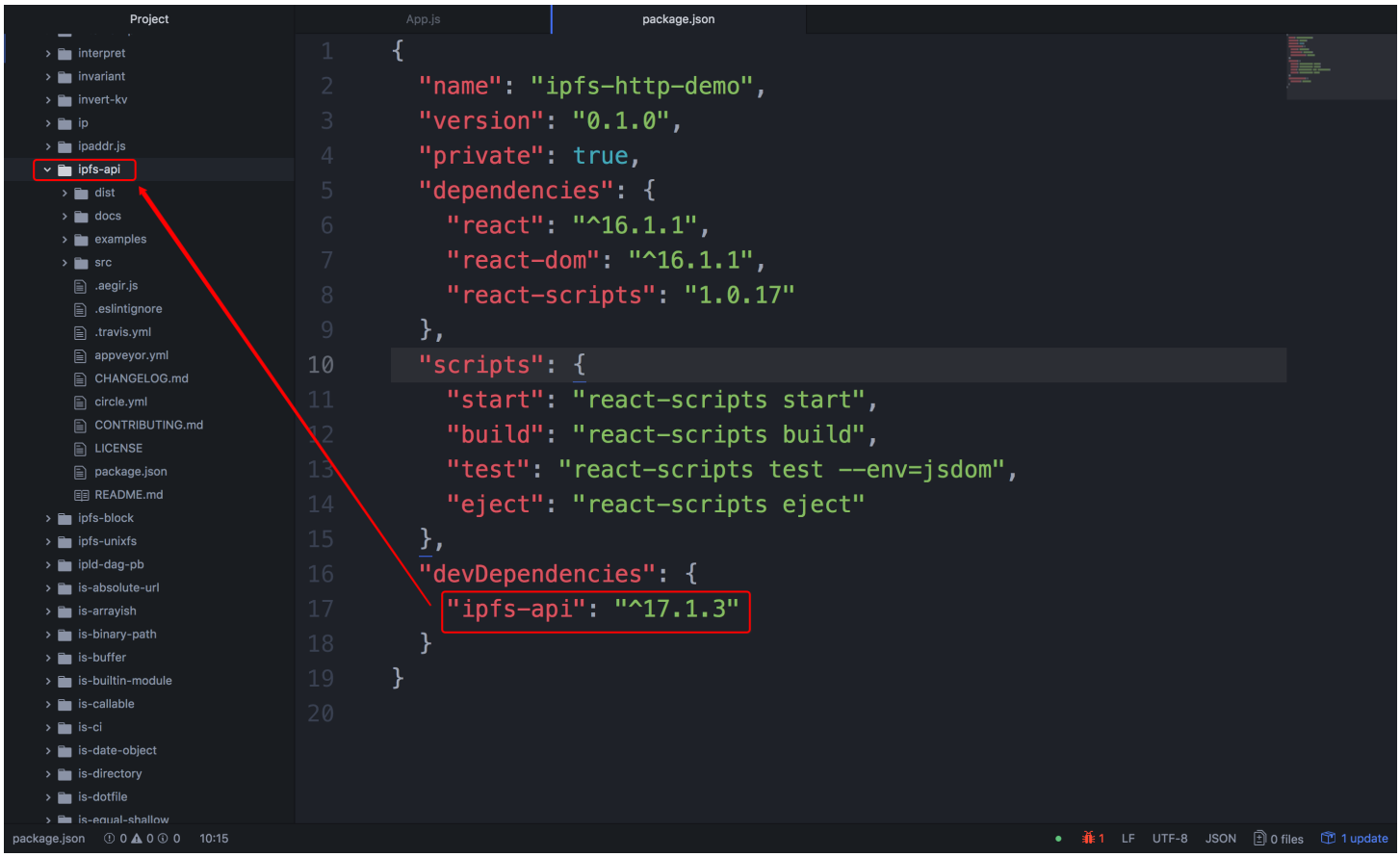


- 安装 `ipfs-api`

切换到项目根目录，安装 `ipfs-api`。

```
$ npm uninstall --save ipfs-api
```

```
localhost:ipfs-http-demo yuechunli$ ls
README.md   package.json    src
node_modules    public      yarn.lock
localhost:ipfs-http-demo yuechunli$ pwd
/Users/liyuechun/Desktop/1123/ipfs-http-demo
localhost:ipfs-http-demo yuechunli$ npm uninstall --save ipfs-api
```



⚠️：**ipfs安装完后，如上图所示，接下来刷新一下浏览器，看看项目是否有问题，正常来讲，一切会正常，🍭🍭🍭，Continue，Continue，Continue.....**

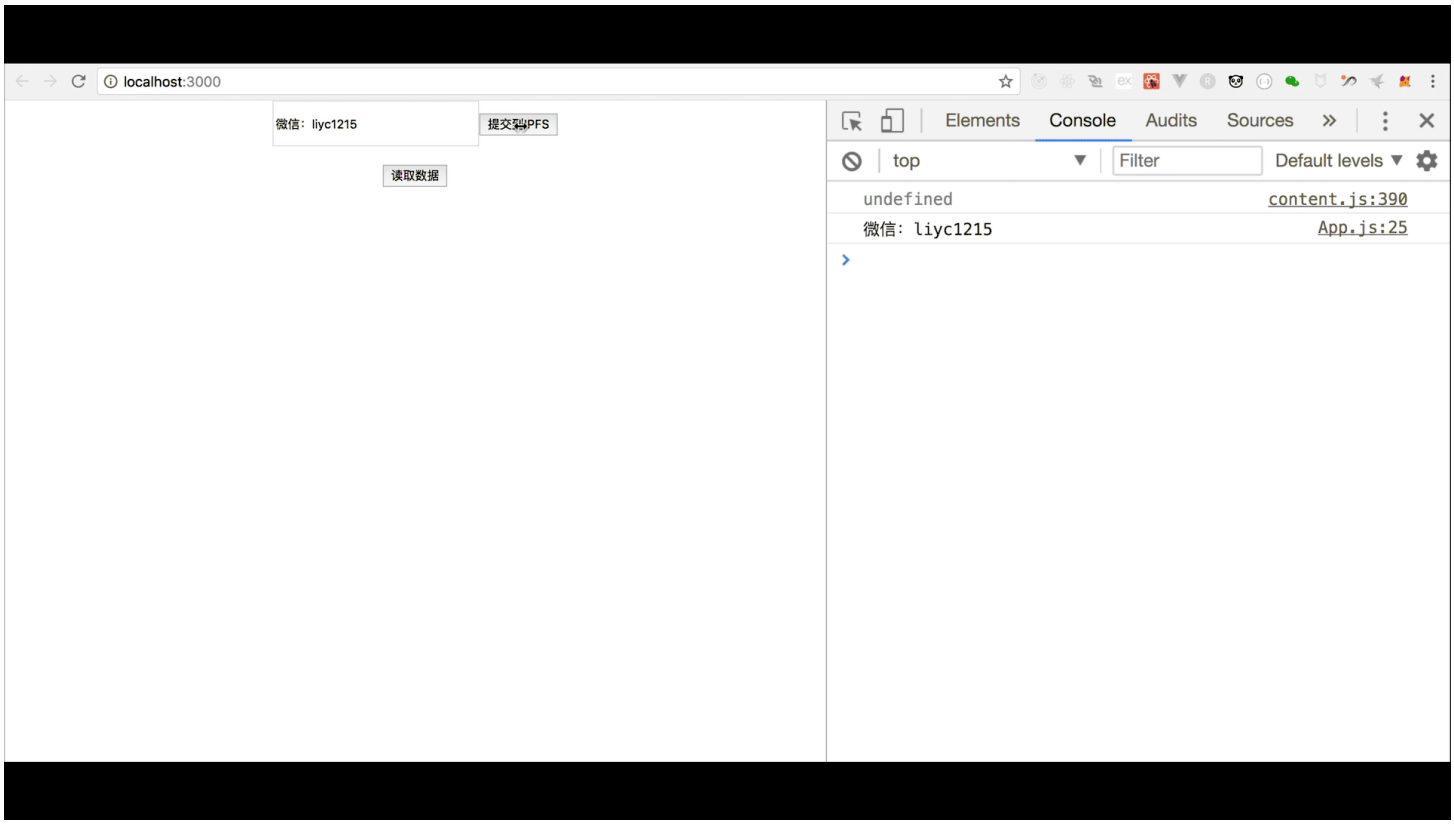
3.6 完成UI逻辑

拷贝下面的代码，将`src/App.js`里面的代码直接替换掉。

```
import React, { Component } from 'react';
import './App.css';

class App extends Component {


      constructor(props) {
          super(props);
          this.state = {
            strHash: null,
            strContent: null
          }
      }

    render() {
      return (
        <div className="App">
```

```
          <input
            ref="ipfsContent"
            style={{width: 200,height: 40,borderWidth:2}}/>
          <button onClick={() => {
            let ipfsContent = this.refs.ipfsContent.value;
            console.log(ipfsContent);
          }}>提交到IPFS</button>

          <p>{this.state.strHash}</p>

          <button onClick={() => {
            console.log('从ipfs读取数据。')
           }}>读取数据</button>
           <h1>{this.state.strContent}</h1>
        </div>
      );
    }
}

export default App;
```

上面的代码完成的工作是，当我们在输入框中输入一个字符串时，点击**提交到IPFS**按钮，将文本框中的内容取出来打印，后续我们需要将这个数据上传到`IPFS`。点击**读取数据**按钮，我们也只是随便打印了一个字符串，后面需要从IPFS读取数据，然后将读取的数据存储到状态机变量`strContent`中并且展示出来。



3.7 导入IPFS

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

3.8 编写上传大文本字符串到IPFS的Promise函数

```
saveTextBlobOnIpfs = (blob) => {
    return new Promise(function(resolve, reject) {
      const descBuffer = Buffer.from(blob, 'utf-8');
      ipfs.add(descBuffer).then((response) => {
        console.log(response)
        resolve(response[0].hash);
      }).catch((err) => {
        console.error(err)
        reject(err);
      })
    })
  }
```

`response[0].hash` 返回的是数据上传到 `IPFS` 后返回的 `HASH` 字符串。

3.9 上传数据到IPFS

```
this.saveTextBlobOnIpfs(ipfsContent).then((hash) => {
    console.log(hash);
    this.setState({strHash: hash});
});
```

`ipfsContent` 是从文本框中取到的数据，调用 `this.saveTextBlobOnIpfs` 方法将数据上传后，会返回字符串 `hash` ，并且将 `hash` 存储到状态机变量 `strHash` 中。

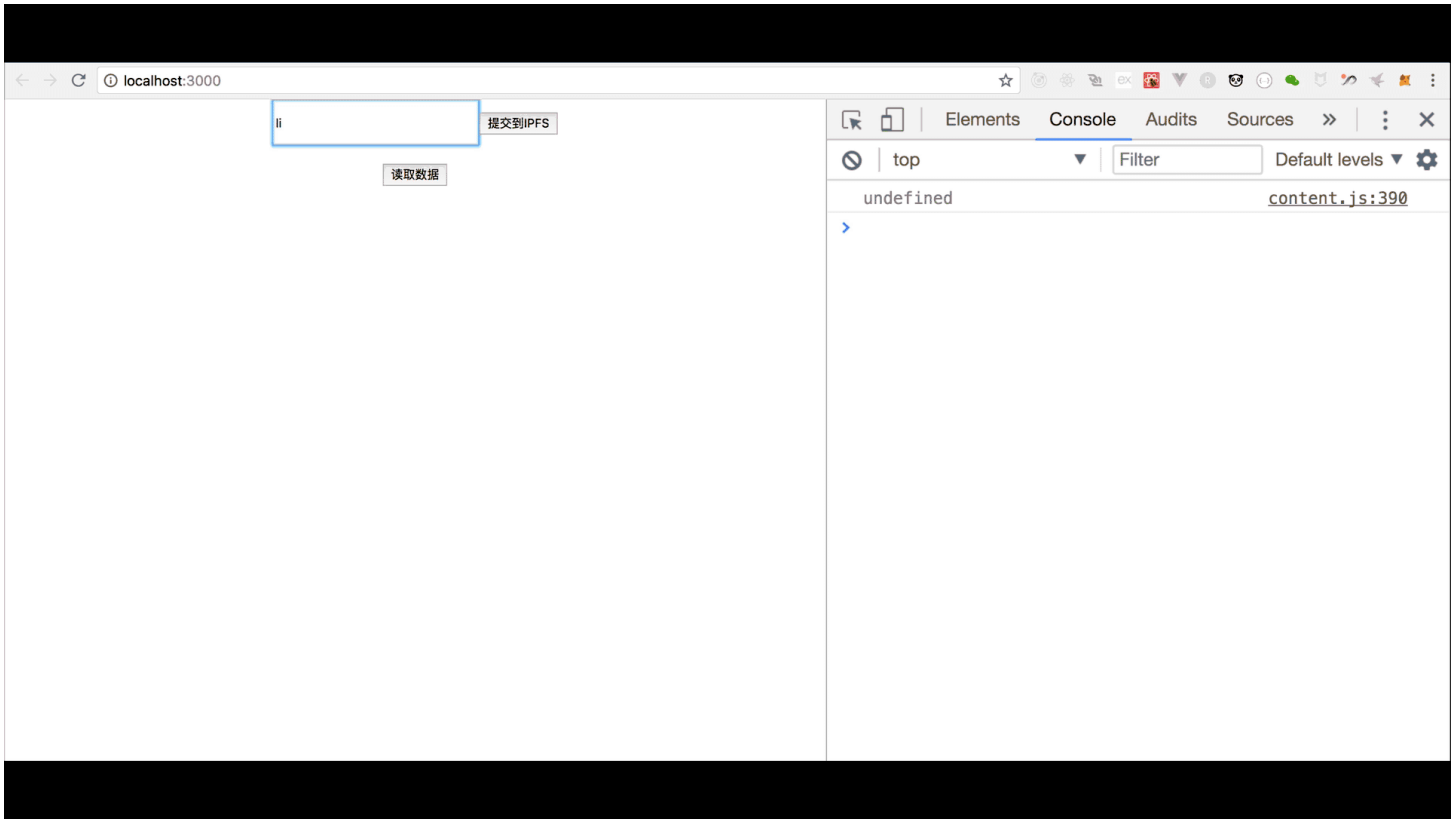
目前完整的代码：

```
import React, {Component} from 'react';
import './App.css';

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

class App extends Component {
```

```
  constructor(props) {
    super(props);
    this.state = {
      strHash: null,
      strContent: null
    }
  }

  saveTextBlobOnIpfs = (blob) => {
    return new Promise(function(resolve, reject) {
      const descBuffer = Buffer.from(blob, 'utf-8');
      ipfs.add(descBuffer).then((response) => {
        console.log(response)
        resolve(response[0].hash);
      }).catch((err) => {
        console.error(err)
        reject(err);
      })
    })
  }

  render() {
    return (<div className="App">
      <input ref="ipfsContent" style={{
          width: 200,
          height: 40,
          borderWidth: 2
        }}/>
      <button onClick={() => {
          let ipfsContent = this.refs.ipfsContent.value;
          console.log(ipfsContent);
          this.saveTextBlobOnIpfs(ipfsContent).then((hash) => {
            console.log(hash);
            this.setState({strHash: hash});
          });
        }}>提交到IPFS</button>

      <p>{this.state.strHash}</p>

      <button onClick={() => {
          console.log('从ipfs读取数据。')
        }}>读取数据</button>
      <h1>{this.state.strContent}</h1>
    </div>);
  }
}
```

```
export default App;
```

测试：



3.10 跨域资源共享CORS配置

跨域资源共享（ CORS ）配置，依次在终端执行下面的代码：

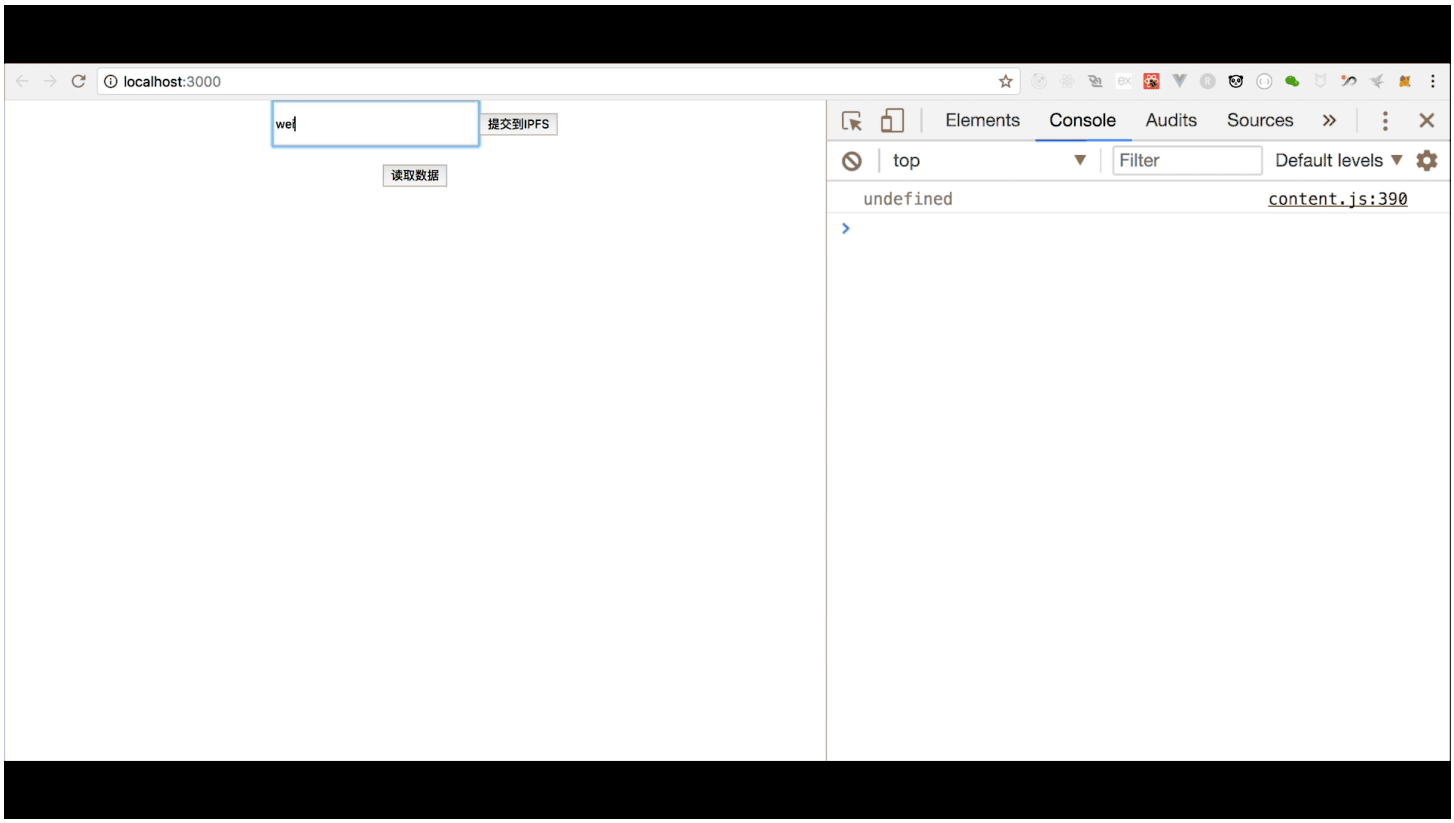
```
localhost:ipfs-http-demo yuechunli$ ipfs config --json API.HTTPHeaders.Access-Control-Allow-Methods '["PUT", "GET", "POST", "OPTIONS"]'

localhost:ipfs-http-demo yuechunli$ ipfs config --json API.HTTPHeaders.Access-Control-Allow-Origin '["*"]'

localhost:ipfs-http-demo yuechunli$ ipfs config --json API.HTTPHeaders.Access-Control-Allow-Credentials '["true"]'

localhost:ipfs-http-demo yuechunli$ ipfs config --json API.HTTPHeaders.Access-Control-Allow-Headers '["Authorization"]'

localhost:ipfs-http-demo yuechunli$ ipfs config --json API.HTTPHeaders.Access-Control-Expose-Headers '["Location"]'
```

用正确的端口运行daemon：

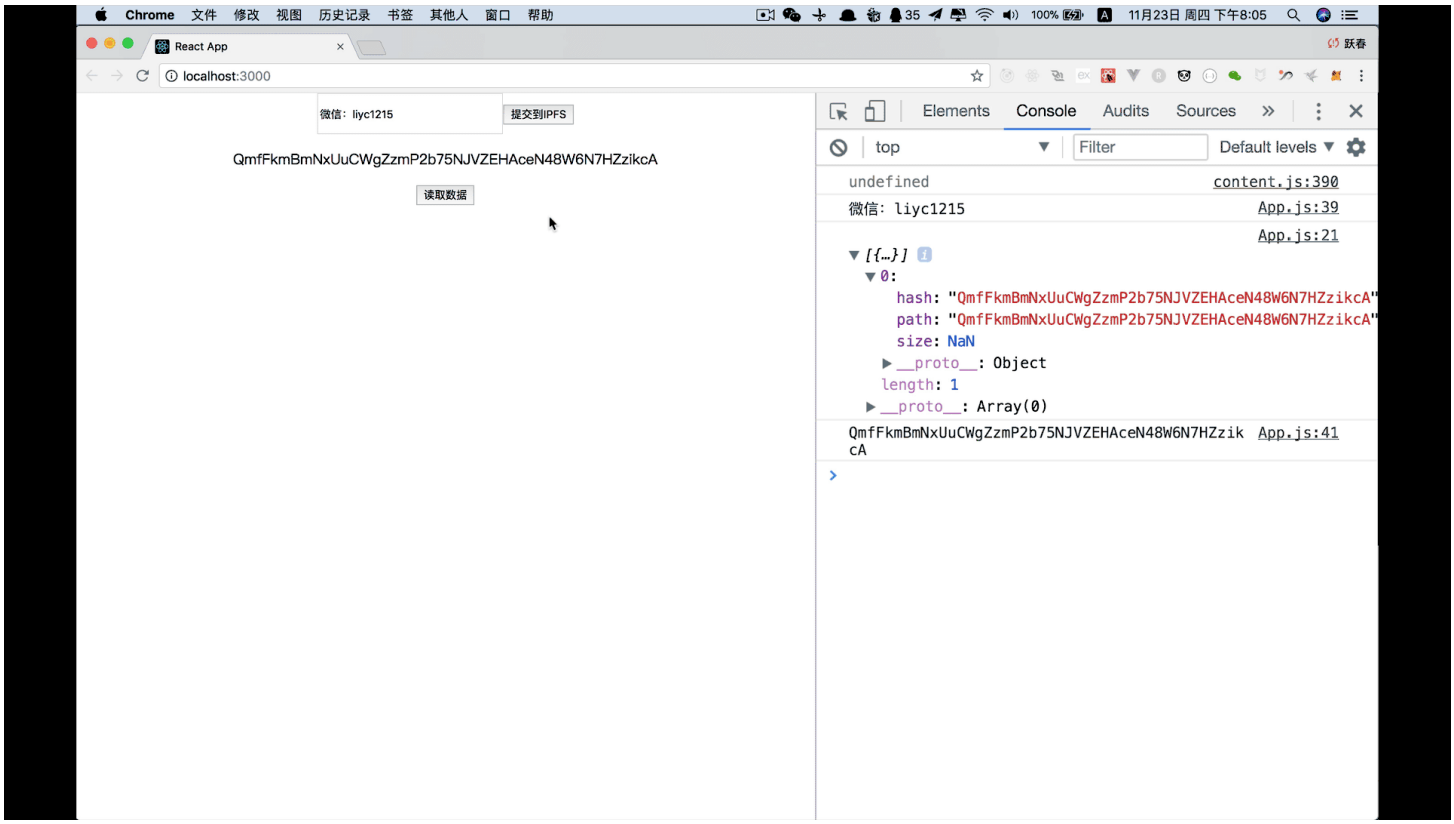
```
localhost:ipfs-http-demo yuechunli$ ipfs config Addresses.API
/ip4/127.0.0.1/tcp/5001
localhost:ipfs-http-demo yuechunli$ ipfs config Addresses.API /ip4/127.0.0.1
/tcp/5001
localhost:ipfs-http-demo yuechunli$ ipfs daemon
```

3.11 再次刷新网页提交数据并在线查看数据

- 上传数据，并且查看返回hash值



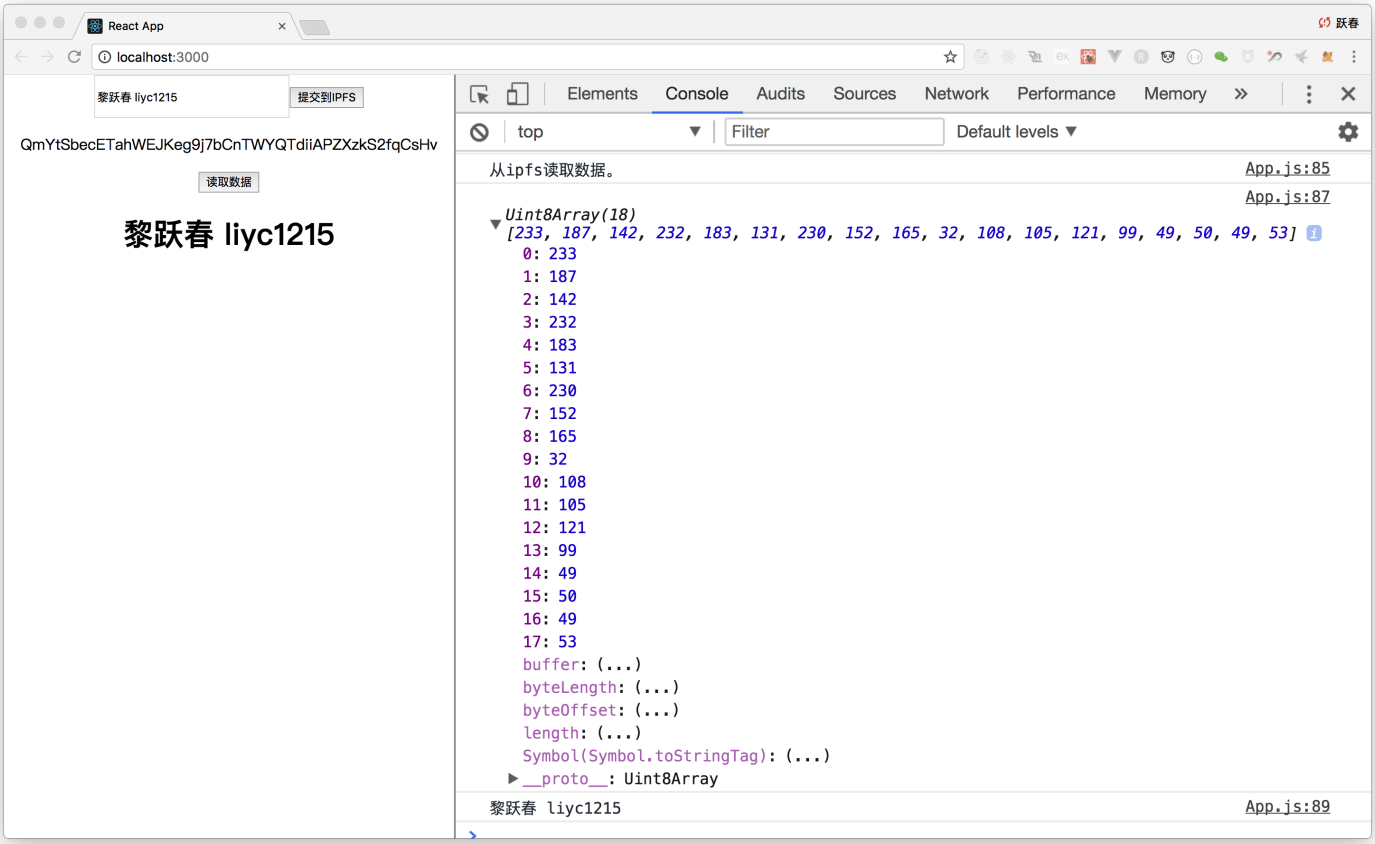
- 在线查看上传到IPFS的数据



3.12 从IPFS读取数据

- `ipfs.cat`

```
ipfs.cat(this.state.strHash).then((stream) => {
    console.log(stream);
    let strContent = Utf8ArrayToStr(stream);
    console.log(strContent);
    this.setState({strContent: strContent});
});
```



`stream` 为 `Uint8Array` 类型的数据，下面的方法是将 `Uint8Array` 转换为 `string` 字符串。

- `Utf8ArrayToStr`

```
function Utf8ArrayToStr(array) {
    var out, i, len, c;
    var char2, char3;

    out = "";
    len = array.length;
    i = 0;
    while(i < len) {
    c = array[i++];
    switch(c >> 4)
      {
        case 0: case 1: case 2: case 3: case 4: case 5: case 6: case 7:
          // 0xxxxxxx
          out += String.fromCharCode(c);
          break;
        case 12: case 13:
          // 110x xxxx   10xx xxxx
          char2 = array[i++];
          out += String.fromCharCode(((c & 0x1F) << 6) | (char2 & 0x3F));
          break;
        case 14:
```

```
          // 1110 xxxx  10xx xxxx  10xx xxxx
          char2 = array[i++];
          char3 = array[i++];
          out += String.fromCharCode(((c & 0x0F) << 12) |
                         ((char2 & 0x3F) << 6) |
                         ((char3 & 0x3F) << 0));
          break;
        default:
          break;
      }
    }

    return out;
}
```

- 完整源码

```
import React, {Component} from 'react';
import './App.css';

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

function Utf8ArrayToStr(array) {
  var out,
    i,
    len,
    c;
  var char2,
    char3;

  out = "";
  len = array.length;
  i = 0;
  while (i < len) {
    c = array[i++];
    switch (c >> 4) {
      case 0:
      case 1:
      case 2:
      case 3:
      case 4:
      case 5:
      case 6:
      case 7:
        // 0xxxxxxx
        out += String.fromCharCode(c);
```

```
        break;
      case 12:
      case 13:
        // 110x xxxx   10xx xxxx
        char2 = array[i++];
        out += String.fromCharCode(((c & 0x1F) << 6) | (char2 & 0x3F));
        break;
      case 14:
        // 1110 xxxx  10xx xxxx  10xx xxxx
        char2 = array[i++];
        char3 = array[i++];
        out += String.fromCharCode(((c & 0x0F) << 12) | ((char2 & 0x3F) << 6
) | ((char3 & 0x3F) << 0));
        break;
      default:
        break;
    }
  }

  return out;
}

class App extends Component {

  constructor(props) {
    super(props);
    this.state = {
      strHash: null,
      strContent: null
    }
  }

  saveTextBlobOnIpfs = (blob) => {
    return new Promise(function(resolve, reject) {
      const descBuffer = Buffer.from(blob, 'utf-8');
      ipfs.add(descBuffer).then((response) => {
        console.log(response)
        resolve(response[0].hash);
      }).catch((err) => {
        console.error(err)
        reject(err);
      })
    })
  }

  render() {
    return (<div className="App">
      <input ref="ipfsContent" style={{
```

```
            width: 200,
            height: 40,
            borderWidth: 2
          }}/>
        <button onClick={() => {
            let ipfsContent = this.refs.ipfsContent.value;
            console.log(ipfsContent);
            this.saveTextBlobOnIpfs(ipfsContent).then((hash) => {
              console.log(hash);
              this.setState({strHash: hash});
            });
          }}>提交到IPFS</button>

        <p>{this.state.strHash}</p>

        <button onClick={() => {
            console.log('从ipfs读取数据。')
            ipfs.cat(this.state.strHash).then((stream) => {
              console.log(stream);
              let strContent = Utf8ArrayToStr(stream);
              console.log(strContent);
              this.setState({strContent: strContent});
            });
          }}>读取数据</button>
        <h1>{this.state.strContent}</h1>
      </div>);
    }
}

export default App;
```

3.13 总结

这篇文章主要讲解如何配置React环境，如何创建React项目，如何安装`js-ipfs-api`，如何上传数据，如何设置开发环境，如何下载数据等等内容。通过这篇文章的系统学习，你会掌握`js-ipfs-api`在项目中的使用流程。

这是`【IPFS + 区块链 系列】 入门篇 – IPFS + Ethereum （上篇）-js-ipfs-api`，下篇讲解如何将IPFS和以太坊智能合约结合进行数据存储。

4. 下篇文章预告

这是`【IPFS + 区块链 系列】 入门篇 – IPFS + Ethereum （上篇）-js-ipfs-api`，下篇讲解如何将IPFS和以太坊智能合约结合进行数据存储。

5. 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