Axios 教程：Vue + Axios 安装及实战 - 手把手教你搭建加密币实时价格看板

作者：HiJiangChuan

原文链接：https://ld246.com/article/1647620900124

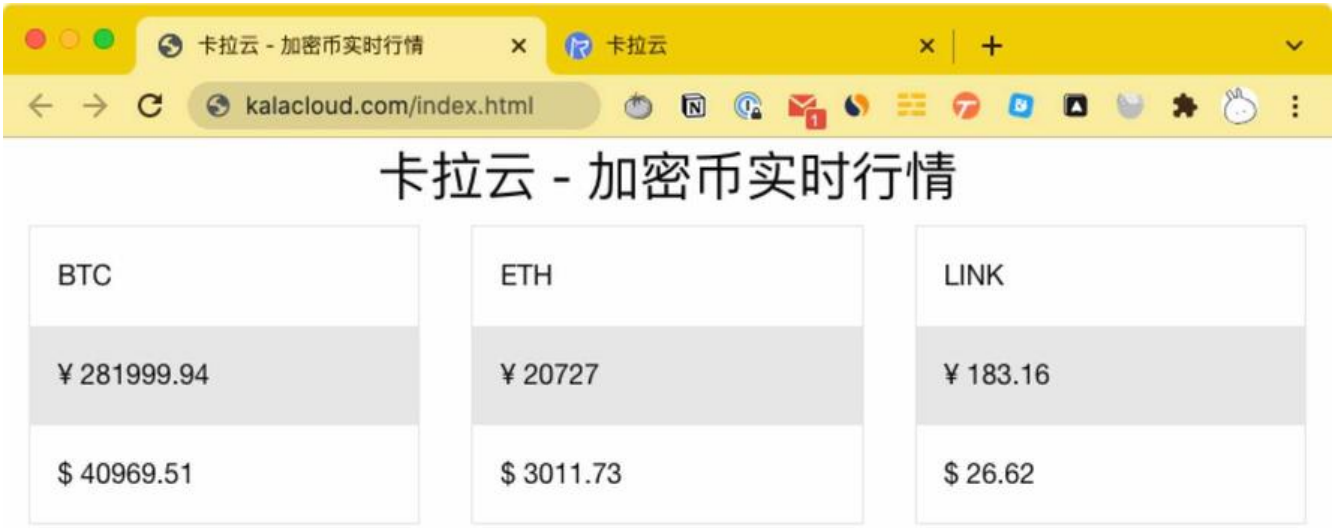
来源网站：链滴

本文首发：《Axios 教程：Vue + Axios 安装及实战 - 手把手教你搭建加密币实时价格看板 - 卡拉云》

Axios 是一个基于 Promise 的 HTTP 请求库，它用在 node.js 和浏览器里。本教程教你如何使用 Axio 库发出 API 请求远程调取数据。

在本教程中，你将学到如何使用 Vue + Axios 搭建一套加密币实时行情看板，你会学到 Axios 如何加密货币行情 API 请求数据，存储数据，然后使用 Vue 在前端展示这些数据，最终完成「实时行情板」的搭建。为了让看板看起来更漂亮，我们将使用 Foundation CSS 框架。

本教程将手把手教你如何通过 Axios 读取 API 数据，搭建一套加密币实时价格看板。



原文链接：Axios 教程：Vue + Axios 安装及实战 - 手把手教你搭建加密币实时价格看板

「加密币实时行情看板」最终效果。前端使用 Vue + Axios，后端调用加密币行情 API，读完本教，你也能搭建一套属于自己的加密币行情数据看板。

如何安装 Axios

可以使用以下简单方法之一将 Axios 添加到我们的项目/代码中：

- npm：

```
npm install axios
```

- bower：

```
bower install axios
```

- yarn：

```
yarn add axios
```

- CDN 方法一：

```
<script src="https://cdn.jsdelivr.net/npm/axios/dist/axios.min.js"></script>
```

- CDN 方法二：

```
<script src="https://unpkg.com/axios/dist/axios.min.js"></script>
```

我们还是来一起搭一个实战项目来学习如何使用 Axios 吧，请务必跟随本教程一起操作。

第 1 步：创建一个最简单的 Vue Web APP

我们先来创建一个最简单的 Vue APP，循序渐进，方便大家理解。

我们新建一个 index.html 文件，先在这个页面里写入一组模拟数据，使用 Vue.js 来显示这个模拟数。后文我们会用真实的 API 来进行替换。

```
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/foundation/6.3.1/css/foun
ation.min.css">
  <meta charset="utf-8">
  <title> 卡拉云 - 加密币实时行情 </title>
</head>

  <body>
    <div class="container" id="app">
      <h3 class="text-center"> 卡拉云 - 加密币实时行情</h3>
      <div class="columns medium-4" >
        <div class="card">
          <div class="card-section">
            <p> 比特币：人民币 </p>
          </div>
```

```
        <div class="card-divider">
          <p> {{ BTCinCNY }} </p>
        </div>
      </div>
    </div>
  </div>

  <script src="https://unpkg.com/vue"></script>
 </body>
</html>
```

在这段 HTML 文件里，我们通过 CDN 加载了 Foundation CSS 框架和 Vue.js。只需简单两行，他就被加载进来，无需下载到本地。

从 Vue.js 中获取的数据会映射到 {{ BTCinCNY }} 里，这就是 Vue 在 HTML 中呈现数据的方式。

我们来定义一下 {{ BTCinCNY }}

在 <script src="https://unpkg.com/vue"></script> 的下面添加以下代码，我们来创建一个新的 Ve app 并定义在 index.html 页面上显示的数据结构：

```
...

  <script>
    const vm = new Vue({
          el: '#app',
          data: { BTCinCNY: 73759.99}
          // 这里是模拟数据，后文我们会用 API 数据替换
        });

  </script>
...
```

这段代码创建了一个新 Vue 应用实例，并将这个实例赋到「 id = app 」到元素上。Vue 把这个过程做加载应用。我们定义了一个新 Vue 实例，通过配置对象对这个应用进行配置，[el](https://v3.cn.vuejs.org/api/application-api.html#el) 指定了加载应用对应的元素 ID，以及包含的数据。

在这个实例中，包含一组「key-value」即 { BTCinCNY: 73759.99 } 这组数据会通过以下代码显示在TML 页面上。

```
<div class="card-divider">
<p> {{ BTCinCNY }} </p>
</div>
```

更新 index.html 我们在浏览器中打开，显示效果如下图

我们再来加上比特币的美元价格，在 index.html 中修改加入美元价格。

```
<script>
const vm = new Vue({
       el: '#app',

       data: { BTCinCNY: 73759.99,BTCinUSD: 3759.91 }
         // 这里是模拟数据，后文我们会用 API 数据替换
     });

</script>
```

然后向标记（div）中添加美元显示的表格部分

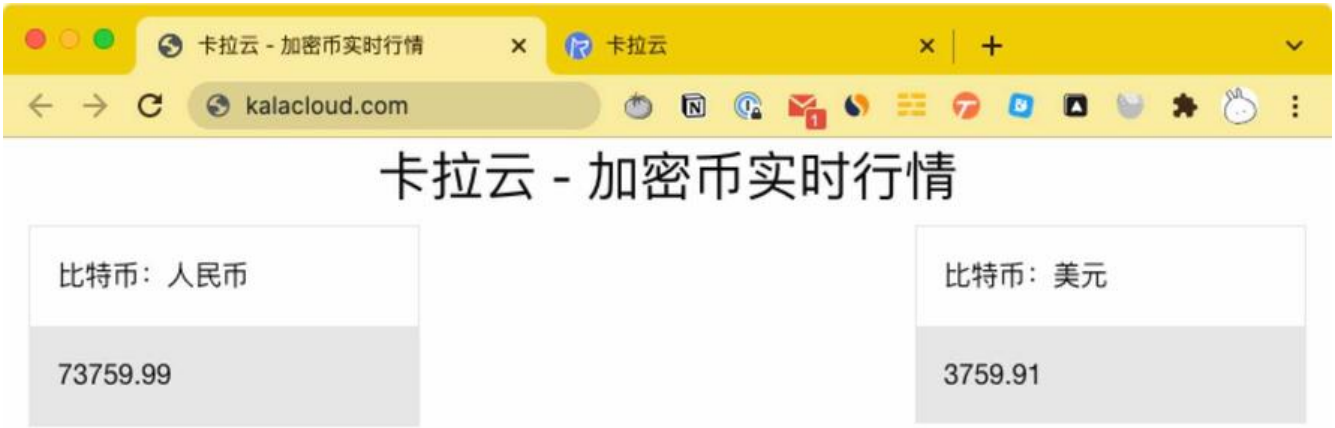
```
...
<div class="columns medium-4" >
      <div class="card">
        <div class="card-section">
          <p> 比特币：美元 </p>
        </div>
        <div class="card-divider">
          {{BTCinUSD}}
        </div>
      </div>
    </div>
...
```

index.html 的完整修改版。请将这段代码更新至 index.html

```
<!DOCTYPE html>
<html>
  <head>
```

```
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/foundation/6.3.1/css/foundation.min.css">
    <meta charset="utf-8">
    <title> 卡拉云 - 加密币实时行情 </title>
  </head>

  <body>
    <div class="container" id="app">
      <h3 class="text-center"> 卡拉云 - 加密币实时行情</h3>
      <div class="columns medium-4">
        <div class="card">
          <div class="card-section">
            <p> 比特币：人民币 </p>
          </div>
          <div class="card-divider">
            <p> {{ BTCinCNY }} </p>
          </div>
        </div>

      </div>
      <div class="columns medium-4">
        <div class="card">
          <div class="card-section">
            <p> 比特币：美元 </p>
          </div>
          <div class="card-divider">
            {{ BTCinUSD }}
          </div>
        </div>

      </div>
    </div>
    <script src="https://unpkg.com/vue"></script>
    <script>
      const vm = new Vue({
        el: '#app',

        data: {
          BTCinCNY: 73759.99,
          BTCinUSD: 3759.91
        }
        // 这里是模拟数据，后文我们会用 API 数据替换
      });
    </script>
  </body>
</html>
```

更新 index.html 后，在浏览器打开显示效果如下：



扩展阅读：《Video.js 使用教程 - 手把手教你基于 Vue 搭建 HTML 5 视频播放器》

第 2 步：分离 JavaScript 和 HTML

在第 1 步中，为了给大家更好的展示工作原理，我们将所有代码都放在 index.html 一个文件中，现我们要把前端和后端数据分成两个独立的文件存放，即 index.html和vueApp.js 。

在 index.html中，显示比特币对应的多种价格。而在 vueApp.js 文件中，用于读取数据。将显示和据页面拆分开来，更便于我们日常维护。

我们先把 index.html 文件中 JavaScript 的代码删掉，将这段指向 vueApp.js 文件

```
...
    <script src="https://unpkg.com/vue"></script>
    <script  language="JavaScript">
    const vm = new Vue({
          el: '#app',
          data: { BTCinCNY: 73759.99,BTCinUSD: 3759.91 }
        });
    </script>
...
```

将 vm 整段删掉，替换为指向 vueApp.js 的 <script>

```
...
    <script src="https://unpkg.com/vue"></script>
    <script src="vueApp.js"></script>
...
```

然后，在 index.html 的存放目录中，新建 vueApp.js 文件，代码如下：

```
const vm = new Vue({
      el: '#app',
      data: { BTCinCNY: 73759.99,BTCinUSD: 3759.91 }
    });
```

保存 vueApp.js 文件并在浏览器重新打开 index.html ，你会发现显示结果没有任何变化。

下一步，我们加入更多加密币的实时行情。

第 3 步：使用 Vue 加载数据

当前页面我们加载了比特币的模拟价格，我们再来加上一个以太币的模拟价格。我们来重构一下视图模拟数据。

打开vueApp.js文件，修改数据，代码如下：

```
const vm = new Vue({
        el: '#app',
        data: {
          results: {"BTC": {"CNY":73759.99,"USD":3166.21},
                    "ETH": {"CNY":33581.77,"USD":2336.25}}}
      });
```

由于数据的嵌套结构，我们使用 results 作为键，它包含两条记录，一个是比特币价格，一个是以太价格。这种结构合并同类项，让数据展示和读取更简洁。

接着我们打开index.html文件并找到显示加密币的部分：

```
...
    <div class="columns medium-4" >
      <div class="card">
        <div class="card-section">
          <p> 比特币：人民币 </p>
        </div>
        <div class="card-divider">
          {{BTCinCNY}}
        </div>
      </div>
    </div>

    <div class="columns medium-4" >
      <div class="card">
        <div class="card-section">
          <p> 比特币：美元 </p>
        </div>
        <div class="card-divider">
          {{BTCinUSD}}
        </div>
      </div>
    </div>

  </div>
...
```

用以下代码替换上面的代码块：

```
...
  <div class="columns medium-4" v-for="(result, index) in results">
    <div class="card">
```

```
      <div class="card-section">
        <p> {{ index }} </p>
      </div>
      <div class="card-divider">
        <p> ¥ {{ result.CNY }}</p>
      </div>
      <div class="card-section">
        <p> $ {{ result.USD }}</p>
      </div>
    </div>
  </div>
...
```

这段代码使用来[v-for](https://vuejs.org/v2/api/#v-for) 指令就像一个 for 循环。它遍历我们数据的所有 key - value

在浏览器中重新加载 index.html，可以看到以下样式：

![02-03-btc-eth](https://kalacloud.com/static/a62f738c6ad8f746c525fe768fad6141/be796/02-03-btc-eth.jpg https://b3logfile.com/file/2022/03/0a99f514cf6f422cbe96efd6e8ca0028.jpeg)]
https://kalacloud.com/static/a62f738c6ad8f746c525fe768fad6141/fc83b/02-03-btc-eth.jpg)

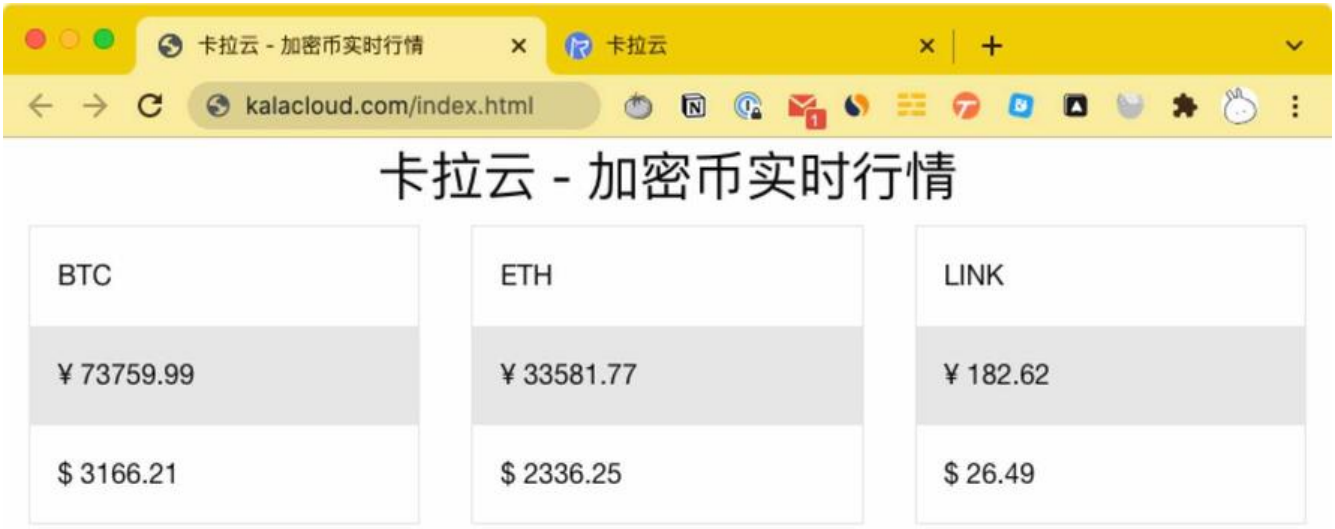
此修改使我们可以向其中的results数据添加新货币vueApp.js并使其显示在页面上而无需进一步更改将另一个模拟条目添加到数据集以进行尝试：

接下来，我们再向 results 里加入一个新的数字货币。这一次，我们无需修改 index.html 就可以自动新。

修改 vueApp.js 文件，直接使用下面的代码替换即可

```
const vm = new Vue({
  el: '#app',
  data: {
    results: {
      "BTC": {
        "CNY": 73759.99,
        "USD": 3166.21
      },
      "ETH": {
        "CNY": 33581.77,
        "USD": 2336.25
      },
      "LINK": {
        "CNY": 182.62,
        "USD": 26.49
      }
    }
  }
});
```

保存后，刷新 index.html 页面，可以看到下面的效果



如果你也做出来本教程的效果，那么太棒了。接下来，我们要接真实的 API 了。

扩展阅读：《7 种最棒的 Vue Loading 加载动画组件测评与推荐 - 穷尽市面上所有加载动画效果（Ve loader）类型》

第 4 步：使用 Axios 从 API 读取数据

我们使用 Cryptocompare 加密币行情网站的实时报价 API 来替换模拟价格数据。

加密货币实时价格 API：

https://min-api.cryptocompare.com/data/pricemulti?fsyms=BTC,ETH,LINK&tsyms=CNY,USD

这个 API 请求比特币、以太坊币、ChainLink币的人民币价格和美元价格。

我们可以用curl向 API 发送请求查看响应：

curl 'https://min-api.cryptocompare.com/data/pricemulti?fsyms=BTC,ETH,LINK&tsyms=CNY,SD'

请求返回输出如下：

{"BTC":{"CNY":281999.94,"USD":40928.18},"ETH":{"CNY":20758.02,"USD":3012.21},"LINK":{"CN":182.62,"USD":26.49}}

API 返回结果与我们的模拟价格数据几乎一样，我们现在要做的是用 API 中的数据替换掉模拟数据。

为了发送请求，我们使用 [mounted()](https://vuejs.org/v2/api/#mounted) Vue 函数，结合 Axios 请求库中的 GET 函数获取数据，然后把读取的数据存在 results 数组中。

我们要在 index.html 中加入加载 Axios 库的代码：

```
...
    <script src="https://unpkg.com/vue"></script>
    <script src="https://unpkg.com/axios/dist/axios.min.js"></script>
```

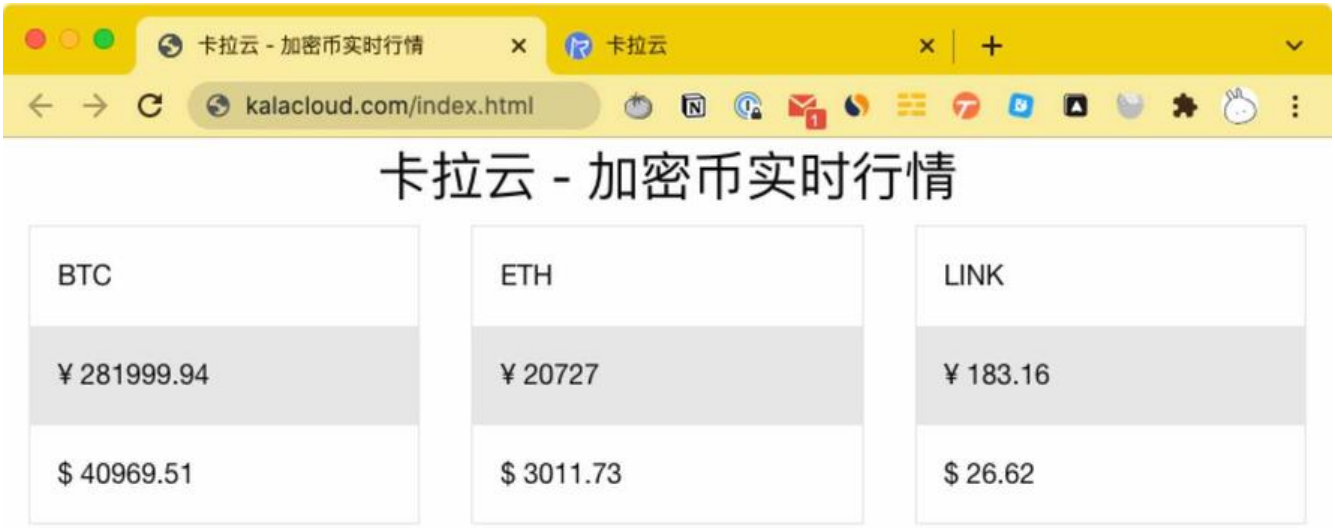
```
    <script src="vueApp.js"></script>
...
```

再打开 vueApp.js 加入请求 API 地址和修改数据存储方式：

```
const url = "https://min-api.cryptocompare.com/data/pricemulti?fsyms=BTC,ETH,LINK&tsym
=CNY,USD";

const vm = new Vue({
        el: '#app',
        data: {
          results: []
        },
        mounted() {
          axios.get(url).then(response => {
            this.results = response.data
          })
        }
      });
```

使用axios.get 函数，当 API 成功返回数据时，then 开始执行，并将数据保存在 results 变量中。

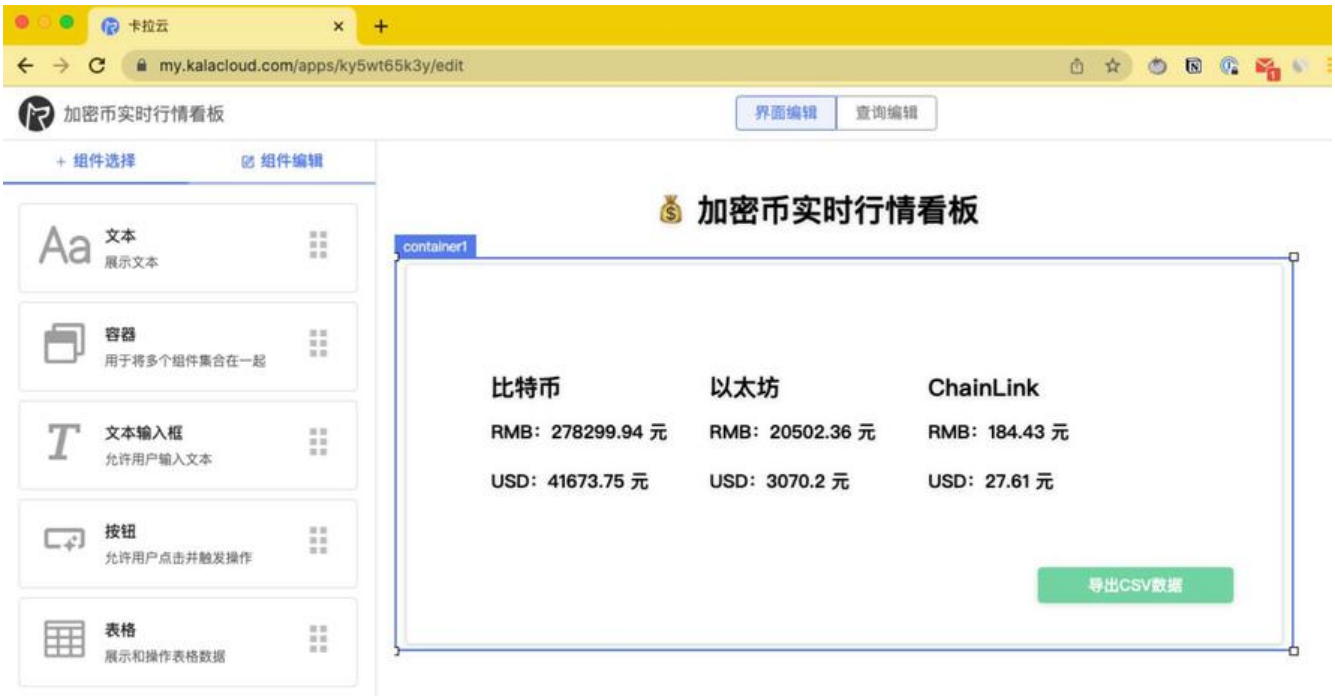
在浏览器重新打开index.html，这时网页上显示的就是真实的加密数字货币实时报价了。



恭喜，你完成了 Vue + Axios 的加密行情看板的搭建。

如果你觉得前端写起来太麻烦，更愿意把宝贵的时间用在深度思考上的话。

我推荐你使用卡拉云，卡拉云无需懂任何前端技术，仅需要拖拽即可快速搭建任何工具系统。



Axios 各类调用方式

Axios 响应对象架构

Axios 请求的响应返回信息包含：

- data: API 返回的数据
- status: HTTP 状态码
- statusText: HTTP 状态信息
- headers：HTTP 标头
- configaxios：请求配置

Axios 响应数据

Axios 响应对象具有data包含解析响应正文的字段。我们可以使用then或 await 接收数据。

如下所示：

```
axios.get('kalacloud.com/api')
  .then(function (response) {
    console.log(response.data);
  });

const { data } = await axios.get(url);
```

Axios 错误处理

使用 catch() 做错误处理

```
axios.get('kalacloud.com/api')
  .then(...)
  .catch(function (error) {
    if (error.response) { // 返回的状态代码不在 2xx 范围内，即错误代码。
      console.log(error.response.data);
      console.log(error.response.status);
      console.log(error.response.headers);
    } else if (error.request) { // 没有返回
      console.log(error.request);
    } else {
      console.log('Error', error.message);
    }
    console.log(error.config);
  });
```

扩展阅读：《顶级好用的 8 款 Vue 弹窗组件测评与推荐》

使用 async-await 处理 Axios 错误

如果你想使用 async-await，只需用 try / catch 块包装 axios 调用。

```
async function getTodo() {
  try {
    const response = await axios.get('kalacloud.com/api');
    console.log(response);
  } catch (error) {
    if (error.response) { // 返回的状态代码不在 2xx 范围内，即错误代码。
      console.log(error.response.data);
      console.log(error.response.status);
      console.log(error.response.headers);
    } else if (error.request) { // 没有返回
      console.log(error.request);
    } else {
      console.log('Error', error.message);
    }
    console.log(error.config);
  }
}
```

Axios GET 请求

```
axios.get('kalacloud.com/api')
  .then(function (response) {
    console.log(response);
  })
  .catch(function (error) {
    console.log(error);
  })
  .then(function () {
  });
```

Async / await:

```
async function getTodo() {
  try {
    const response = await axios.get('kalacloud.com/api');
    console.log(response);
  } catch (error) {
    console.error(error);
  }
}
```

扩展阅读：《ECharts X 轴标签过长导致文字重叠的 4 种解决方案》

Axios GET 带参数请求

你可以使用 params 来带 API 提供的参数。

```
axios.get(
  'kalacloud.com/api',
  {
    params: {
      title: '标题数据'
    }
  }
);
```

当然也可以这么写，他们效果一样。

```
axios.get('/todo?title=标题数据');
```

Axios GET 带 headers 请求

```
axios.get(
  'kalacloud.com/api',
  {
    headers: {
      'x-access-token': 'token-value'
    }
  }
);
```

Axios GET 同时带参数和 headers 请求

```
axios.get(
  'kalacloud.com/api   ',
  {
    params: {
      title: '标题数据'
    },
    headers: {
      'x-access-token': 'token-value'
    }
  }
);
```

扩展阅读：《12 款最棒 Vue 开源 UI 库测评 - 特别针对国内使用场景推荐》

Axios PUT 请求

```
axios.put(
  'kalacloud.com/api/1',
  {
    title: title,
    description: description,
    published: true,
  }
);
```

Axios POST 带 body 请求

```
axios.post(
  'kalacloud.com/api',
  {
    title: title,
    description: description,
  }
);
```

扩展阅读：《最好用的 12 款 Vue Timepicker 时间日期选择器测评推荐》

Axios POST 带 headers 请求

```
await axios.post(
  'kalacloud.com/api',
  {
    title: title,
    description: description,
  },
  {
    headers: {
      "x-access-token": "token-value",
```

```
    },
  }
);
```

Axios PUT 带 headers 请求

```
axios.put(
  'kalacloud.com/api2',
  {
    title: title,
    description: description,
    published: true,
  },
  {
    headers: {
      "x-access-token": "token-value",
    },
  }
);
```

扩展阅读：《Vue + Node.js 搭建「文件上传」管理后台》

Axios DELETE 请求

```
axios.delete('kalacloud.com/api/2');
```

Axios DELETE 带 headers 请求

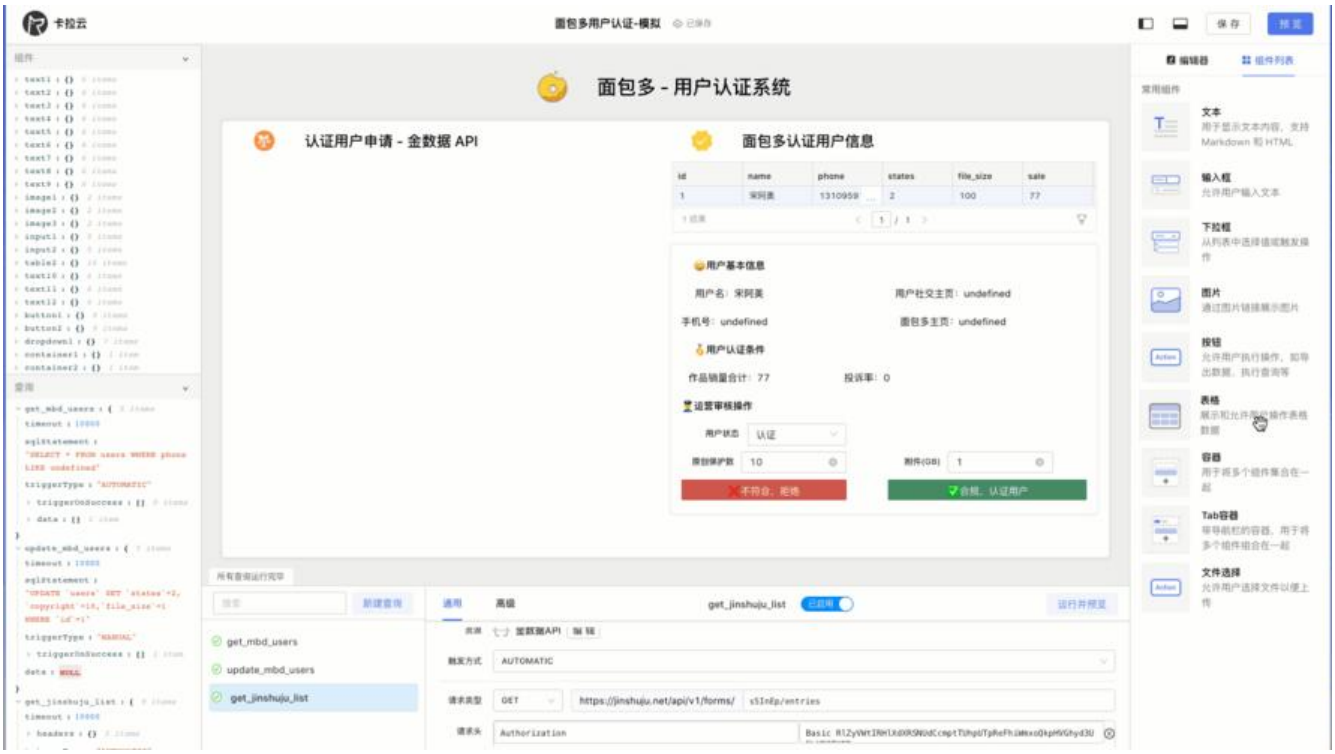
```
axios.delete(
  'kalacloud.com/api/2',
  {
    headers: {
      "x-access-token": "token-value",
    },
  }
);
```

Vue + Axios 搭建总结

首先恭喜你使用 Vue + Axios 搭建完成「加密币实时行情看板」，这说明你已经基本掌握了这项技能前端开发就是这么繁琐和重复，需要你不断练习掌握。

那么，有没有完全不用写前端，直接填上数据库地址或 API 地址就能搭出可用的后台工具呢?

有。 推荐使用卡拉云，卡拉云帮你解决了前后端搭建的全部问题，你完全不用掌握任何前端后端技术只需要几行代码，即可接入数据库 & API 。再复杂的项目，也只需正常开发的 10% 的工时即可完成。



卡拉云是一套低代码开发工具，仅需简单拖拽即可快速搭建后台工具。立即试用卡拉云，一分钟快速建属于你自己的后台工具。

扩展阅读：

- MySQL 时间戳用什么类型 - MySQL 时间函数详解
- 最好用的七大顶级 API 接口测试工具
- 最好用的 5 款 React 富文本编辑器
- 如何在 MySQL / MariaDB 中跳过多张表导出或指定多张表导出备份
- 如何将 MySQL / MariaDB 的查询结果保存到文件
- 如何在 MySQL 中导入和导出 CSV / Excel 文件