Vue DAY04

axios

封装了网络请求的基础模块：`XHR` 或 `fetch`

由于原生`Ajax`请求代码较为复杂，所以经常会封装后进行使用：

1. `jQuery`

   ```
   $.get()
   $.post()
   $.ajax({
       url:,
       method:,
       success: (res)=>{
           ......
       }
   })
   ....
   ```

   简化了原生发送请求时的代码量，抹平了一些技术细节与浏览器兼容性问题。

2. `axios`：一款专业的用`Promise`封装而来的网络请求库。`Api`更加简洁。

   `axios`的官方网站：`http://www.axios-js.com/`

安装`axios`

基于脚手架项目，安装`axios`模块，需要在项目的**根目录**下执行命令：

```
npm install axios
npm i axios
```

基于`axios`发送`get`请求

方式1： `axios.get(url)`

```
// 引入axios
import axios from 'axios'
// 调用get方法发送get请求
let url = 'http://ip:port/api?name=zs&pwd=1234'
axios.get(url).then(res=>{
    // .then()中的回调方法，将会在请求成功，获取响应后自动执行
    // res就是请求成功后，axios封装的响应数据对象
}).catch(err=>{
    // 如果请求失败，执行catch中的回调方法
})
```

```
http://localhost:3000/movie-infos?page=2&pagesize=2
修改前缀:
https://web.codeboy.com/bmdapi/
https://web.codeboy.com/bmdapi/movie-infos?page=2&pagesize=2
```

方式2：axios(url)

```
axios(url).then(res=>{
    // res就是请求成功后，axios封装的响应数据对象
}).catch(err=>{
    // 如果请求失败，执行catch中的回调方法
})
```

方式3：`axios.create().get()`

```
let instance = axios.create()
instance({
    url: '请求地址',
    method: 'get',
    params: {
        page: 1,
        pagesize:5
    }
}).then(res=>{ 获取响应结果res })
```

基于`axios`发送`post`请求

方式1：

```
let url = 'https://web.codeboy.com/bmdapi/movie-infos/name'
let data = `name=${this.keyword}&page=1&pagesize=10`
axios.post(url, data).then(res=>{
    console.log(res)
    this.data = res.data.data
})
```

案例：测试接口：

```
https://web.codeboy.com/bmdapi/movie-infos/name
post请求的参数:
name
page
pagesize
```

方式2:

```
let instance = axios.create()
instance({
    url: '请求路径',
    method: 'post',
    data: `name=${this.keyword}&page=1&pagesize=10`
}).then(res=>{ 获取响应结果res })
```

推荐大家一个 `object` 转 `formdata` 格式字符串的模块: `qs` 。

```
# 安装qs模块
npm i qs
```

调用方法，将对象转成 `formdata` 格式的字符串:

```
import qs from 'qs'
qs.stringify({name:'zs', pwd:'1234'})  ->  "name=zs&pwd=1234"
```

至此发现：使用axios发送get请求，与发送post请求的API设计的不太一致，导致以后调用时增加开发成本，所以工程化开发时通常情况下需要自己封装一个新的 `axios` ，设计一些用着舒爽的 `API` ，增加代码复用性与可维护性。

封装 `Axios`

期望：无论发送get与post，不用写太多代码，简单的调用get、post方法，传url与对象参数即可完成请求的发送与响应的接收。

如下调用:

```
import myaxios from 'MyAxios.js'
// 发送get请求
myaxios.get(url, {id:1}).then(res=>{
    成功后执行
})
// 发送post请求
myaxios.post(url, {id:1, page:1, pagesize:10}).then(res=>{
    成功后执行
})
```

`vue`的自定义组件

被`vue`所管理的标签可以认为是`vue`的**组件**。而在项目开发的过程中，经常会用到一些需要复用的标签结构以及相应样式。vue提供了**自定义组件**的写法，可以让开发者将一些需要复用的页面结构、样式、功能组织在一起，作为一个整体存在在项目中。这样，当需要使用这个组件时，直接引用该组件即可，例如：

```
<div>
    <!-- 该组件就称为一个自定义组件，由开发者自行设计
         标签名自定义，属性自定义，事件自定义 -->
    <person src="图片路径" name="昵称"></person>
</div>
```

如何设计并实现一个自定义组件

目标，设计一个组件，模仿标签的样式，方便的进行使用。

实现步骤：

1. 设计一个自定义组件（包括它的外观、功能、未来怎么复用）。
2. 定义一个组件：
    1. 在components目录下新建`MyTag.vue`。并且在该组件中定义基础样式。
    2. 注意：`<script>export default {}</script>`
3. 当需要使用该组件时，需要先引入再使用：
    1. 引用`MyTag`组件：

    ```
    import MyTag from './components/MyTag.vue'
    components: {
        // 组件名:组件对象
        // 组件名相当于标签名，在页面中可以直接使用
        // vue为了使用方便，自动支持将驼峰命名法 改为 短横线命名法
        // <MyTag></MyTag>      大驼峰
        // <my-tag></my-tag>    短横线
        MyTag: MyTag
    }
    ```

    2. 直接通过标签名，使用该组件：

    ```
    <my-tag></my-tag>     // 直接看到自定义组件的外观
    ```

如何向子组件传参，修改子组件的内容