Vue DAY04

Axios

axios是一个网络通信库，封装了原生了ajax。提供了一些简单的API辅助程序员方便的发送http、https请求。底层基于Promise进行封装。

在脚手架中安装axios

找到项目根目录，打开cmd，执行命令：

```
npm  install  axios
```

安装成功后，将会在package.json中生成axios的依赖项。

基于axios发送GET请求

```
import axios from 'axios'
let instance = axios.create()
instance({
    url: '请求资源路径',
    method: 'GET',
    params: {name:'zs', pwd:'1234'}
}).then(res=>{
    res就是发送请求后，axios封装的响应数据
})
```

测试接口：

```
https://web.codeboy.com/bmdapi/movie-infos?page=1&pagesize=20
```

基于axios发送POST请求

```
import axios from 'axios'
let instance = axios.create()
instance({
    url: '请求资源路径',
    method: 'POST',
    // post请求通过data来传递参数
    // data: 'page=1&pagesize=20&name=xxx',    传递普通kv参数
    // data: {page:1, pagesize:20},            传递json参数
}).then(res=>{
    res就是发送请求后，axios封装的响应数据
})
```

post请求参数的类型有很多种：

1. application/json 传递json字符串
2. x-www-form-urlenocded 普通的k=v&k=v
3. multipart/form-data 用于上传文件

提供一个post请求的接口，用于通过关键字模糊查询电影列表

```
接口地址：
https://web.codeboy.com/bmdapi/movie-infos/name
请求方式：
POST
请求参数：
page=1&pagesize=20&name=熊猫
```

封装Axios

由于axios官方提供的api使用频繁，细节太多，需要针对自己项目常用的场景进行一次封装，使得发请求更加方便。

期望：无论发送get还是post，不要写太多的代码，简单的调用一个get、post方法就行了。传递参数的时候也不要区分params、data等，直接传递对象即可。

```
import myaxios from 'myaxios.js'

let params = {page:1, pagesize:20}
myaxios.get(url, params).then(res=>{})
myaxios.post(url, params).then(res=>{})
```

课堂练习：

模仿上述查询电影列表业务，根据下面的接口，加载演员列表。

```
接口地址：https://web.codeboy.com/bmdapi/movie-actors
请求方式：GET
请求参数：page=1&pagesize=10
返回结果：[演员对象组成的数组]
```

实现思路：新建一个组件：`views/Actors.vue`。当访问：/actors时，可以看到该页面。在页面中提供一个按钮，点击后加载所有的演员，以列表的方式显示。

1. 新建组件 `views/Actors.vue`，提供一个按钮。
2. 配置路由，访问 `/actors` 看到组件。
3. 给按钮绑定事件，点击后使用myaxios发送get请求，打印查询结果。

加载导演列表：

```
接口地址：https://web.codeboy.com/bmdapi/movie-directors
请求方式：GET
请求参数：page=1&pagesize=10
返回结果：[导演对象组成的数组]
```

Vue组件化开发

在Vue项目的开发过程中，官方推荐使用组件化开发的思想来设计并实现页面结构。不建议在一个网页中编写过于复杂的页面结构。推荐将页面内容按照功能模块拆分，形成不同的组件，多个组件组成一个完整的页面：

```
<div>
    <header></header>
    <footer></footer>
</div>
```

案例可以改造为：

```
<template>
  <div>
    <!-- Actors.vue -->
    <h3>演员列表</h3>
    <button @click="listActors">加载演员列表</button>
    <person v-for=""></person>

    <h3>导演列表</h3>
    <button @click="listDirectors">加载导演列表</button>
    <person v-for=""></person>
  </div>
</template>
```

Vue中的自定义组件

被Vue所管理的标签可以认为是vue的**组件**。在项目开发过程中经常会遇到一些需要**复用**的标签结构以及相应的样式、功能。vue提供了自定义组件的语法，可以让开发者将一些重复使用的页面结构、样式、功能组织一起，作为一个整体（新的组件）存在在项目中。这样如果需要使用这个组件时，直接引用即可使用自定义标签来显示。

```
<person avatar="http:头像路径" name="人名"></person>
```

如何设计并实现一个自定义组件?

1. 新建自定义组件文件：`src/components/Person.vue`。
2. 在该文件中编写`template`、`script`、`style`代码。

3. 当需要使用这个组件时，需要引入该组件，使用自定义标签来引用它，显示它。

```
<person></person>
<abc></abc>
<PersonInfo></PersonInfo>
<person-info></person-info>
```

```
import Person from '@/components/Person.vue'
export default {
    // components用于声明当前组件需要引用的子组件
    components: {
        // 属性名就是自定义标签名 :    属性值引用了Person对象
        Person: Person,
        abc: Person,
        PersonInfo: Person
    }
}
```

父组件向子组件传递自定义参数

```
<person avatar="头像路径" name="人名"></person>
```

若子组件需要接收父组件传来的参数（为了动态显示组件内容），则需要在子组件中事先声明自定义属性，通过自定义属性来接收父组件传来的数据，从而实现相应的功能。

1. 在子组件中设计一些自定义属性：

```
export default {
    // props选项用于定义当前组件的自定义属性 avatar name
    // 一旦定义了这些属性，则在使用当前组件时，就可以传参
    // <person avatar="a.jpg" name='张三' ></person>
    // <person avatar="b.jpg" name='李四' ></person>
    // <person avatar="c.jpg" name='王五' ></person>
    // <person avatar="d.jpg" name='赵六' ></person>
    props: ['avatar', 'name']
}
```

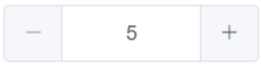
2. 在使用子组件时，就可以给这些属性赋值（通过标签属性赋值的语法）：

```
<person avatar="头像路径" name="人名"></person>
<person :avatar="变量a" :name="变量b"></person>
```

在组件声明自定义属性时，可以用一种相对详细的语法定义属性的细节：

```
export default {
    props: {
        name: {
            type: string,     // 指定属性的数据类型
            default: '姓名'    // 没有指定name时的默认值
        },
        avatar: {
            type: string,
            required: true    // 要求该属性必填
        }
    }
}
```

案例：设计一个计数器组件，可以实现如下功能：




```
<counter></counter>
<counter :value="25"></counter>
<counter :min="1" :max="10"></counter>
<counter :min="1" :max="10" :step="2"></counter>
```

1. 通过counter标签即可看到该组件。
2. 通过value属性，设置计数器的初始值。
3. 通过min定义最小值，通过max属性定义最大值。
4. 如果已经是最小值，则不能再减；同理，如果已经是最大值，则不能再加了。