Vue DAY04

axios

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

https://www.axios-http.cn/

在脚手架中安装axios

在项目的根目录下，通过cmd，执行命令，安装axios：

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

基于axios发送GET请求：

```
https://web.codeboy.com/bmdapi/movie-infos?page=1&pagesize=10
```

```
import axios from 'axios'
let instance = axios.create()   // 创建axios对象，通过instance发送请求

instance({
    url: 'https://web.codeboy.com/bmdapi/movie-infos',    // 请求地址
    method: 'GET',   // 请求方式
    params: {page:1, pagesize:20}   // 请求参数
}).then(res=>{
    res就是响应数据
}).catch(err=>{
    err就是请求失败信息
})
```

案例：安装axios，新建：views/Http.vue。当访问：/http时看到它。

基于axios发送POST请求：

```
通过关键字模糊查询电影列表
https://web.codeboy.com/bmdapi/movie-infos/name
post
name:关键字       page:页码       pagesize:每页多少条
```

```
import axios from 'axios'
let instance = axios.create()   // 创建axios对象，通过instance发送请求

instance({
    url: 'https://web.codeboy.com/bmdapi/movie-infos/name',
    method: 'POST',   // 请求方式
    data: 'name=熊&page=1&pagesize=20'   // 请求参数
}).then(res=>{
    res就是响应数据
}).catch(err=>{
    err就是请求失败信息
})
```

我们发现，post请求的参数与get请求 的参数设置不同，需要使用`data`指定参数列表：

```
data: 'name=熊&page=1&pagesize=20'
```

注意，在此处是否直接使用对象传参取决于服务端是否支持。默认情况下`nodejs`与`java`等服务器是不支持。如果服务端不支持直接接收`json`，那么就只能传递参数字符串：`name=熊&page=1&pagesize=20`。

但是如果参数过多，就非常容易拼接错误，有一个`js`模块：`qs`模块，可以将对象转成这种查询参数字符串的格式：

```
import qs from 'qs'
let param = {name:'熊', page:1, pagesize:20}
let str = qs.stringify(param)    // 将param参数对象转为查询字符串格式
// str -->  name=熊&page=1&pagesize=20
```

封装`Axios`

由于`axios`官方提供的`API`使用频繁，细节太多，需要针对自己的项目的各种场景封装一个通用的、方便的`axios`网络请求库。

期望： 无论发送`get`还是`post`请求，不要写太多代码，不要有太大的区别，简单的两个方法：`get()` `post()`方法传递`url`路径、`params`参数即可完成请求的发送。

例如：若发送get请求，则：

```
import myaxios from 'xxx/xx/myaxios'

let url = "https://xxxxxxxx/xxxxx/xxxx"
let params = {page:1, pagesize:20}
myaxios.get(url, params).then(res=>{
    // res就是axios封装的响应结果
})
```

例如：若发送post请求，则：

```
import myaxios from 'xxx/xx/myaxios'

let url = "https://xxxxxxxx/xxxxx/xxxx"
let params = {name:'熊', page:1, pagesize:20}
myaxios.post(url, params).then(res=>{
    // res就是axios封装的响应结果
})
```

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。

```
请求地址：  https://web.codeboy.com/bmdapi/movie-actors
请求方式：  GET
请求参数：  page=1    pagesize=100
```

实现步骤：

1. 新建页面：`views/Actor.vue`，当访问：`/actor`时看到该页面。
2. 提供一个按钮，点击按钮后，加载前100个演员信息，以列表的方式进行呈现。

`vue`的组件化编程 - 自定义组件

```
<div>
    <!-- 演员列表项 -->
    <div class="person">
      <img :src="item.actor_avatar" :title="item.actor_name" />
      <span>{{ item.actor_name }}</span>
    </div>
    <!-- 演员列表项 -->
    <div class="person">
      <img :src="item.actor_avatar" :title="item.actor_name" />
      <span>{{ item.actor_name }}</span>
    </div>
    <!-- 演员列表项 -->
    <div class="person">
      <img :src="item.actor_avatar" :title="item.actor_name" />
      <span>{{ item.actor_name }}</span>
    </div>
</div>
```

如上述代码中，有很多重复的页面结构，用于描述一个演员。如果遇到了这种情况，`vue`建议使用自定义组件封装这些可以复用的结构，简化每一个组件的写法。

```
<div>
    <header></header>
    <person></person>
    <person></person>
    <person></person>
    <nav></nav>
    <menu></menu>
    <product></product>
    <.....></.....>
    <footer></footer>
</div>
```

`Person.vue`（`vue`自定义组件）

```
<template>
    <!-- 演员列表项 -->
    <div class="person">
      <img :src="item.actor_avatar" :title="item.actor_name" />
      <span>{{ item.actor_name }}</span>
    </div>
</template>
```

如何设计并实现一个自定义组件？（例如`Person`组件）

设计期望：通过person组件，显示演员的个人信息（基础页面外观）

```
<div>
    <person></person>    显示演员信息
</div>
```

实现步骤：

1. 在`src/components`中，新建自定义组件：`Person.vue`，在该文件中编写组件的基本结构、外观样式、功能。

    ```
    <template>
      <div class="person">
        <!-- src/components/Person.vue -->
        <img src="https://p0.pipi.cn/basicdata/25bfd6d7537e7acf3e3"/>
        <span>邓超</span>
      </div>
    </template>

    <script>
    export default {};
    </script>

    <style lang="scss" scoped>
    .person {
    ```

```
  width: 80px;
  text-align: center;
  display: inline-block;
  margin-right: 10px;
  img {
    width: 80px;
  }
  span {
    font-size: 0.9em;
  }
}
</style>
```

2. 如果希望使用该组件，则在父组件中通过`import`引入该子组件即可直接使用该组件。

```
<template>
    <div>
        <person></person>

        abc:<abc></abc>
        MyPerson:<MyPerson></MyPerson>
        m-person:<m-person></m-person>
        my-person:<my-person></my-person>
    </div>
</template>
<script>
import Person from '@/components/Person.vue';  // 引入子组件 Person
export default {
  // components用于指明当前组件中，需要使用哪些子组件
  components: {
    Person, // 有一个子组件，名字（标签名）叫做Person，<person></person>
    'abc': Person,
    'MyPerson': Person,
    'm-person': Person,
  }
};

</script>
```

父组件向自定义的子组件传参 （组件的自定义属性）

```
<div>
  <person name="邓超" avatar="https://xxxxx/xxxx/xxx.jpg"></person>     <person name="孙俪" avatar="https://xxxxx/xxxx/xxxxx.jpg"></person>   <person name="徐峥" avatar="https://xxxxx/xxxx/xxxxz.jpg"></person>   <person name="王宝强" avatar="https://xxxxx/xxxx/xxwbq.jpg"></person>
</div>
```

在使用子组件时，可以通过属性的方式，向子组件传递参数，这样，子组件就可以根据这些参数动态的显示组件内容。而子组件接收这些参数，就需要事先在子组件中声明自定义属性：

```
<template>
  <div class="person">
    <!-- src/components/Person.vue -->
    <img :src="avatar"/>
    <span>{{name}}</span>
  </div>
</template>

<script>
export default {
  // props声明自定义属性，接收父组件传过来的参数：name  avatar
  // 一旦声明了这两个属性，
  // template中可以使用{{name}} {{avatar}}直接引用父组件传过来的参数
  props: ['name', 'avatar']
};
</script>
```

vue还提供了对象的方式来声明自定义属性：

```
<script>
export default {
    props: {
        name: String,   // 指定name属性的数据类型
        avatar: {       // 指定avatar属性的数据类型与是否必填
            type: String,   // String类型
            required: true  // 要求avatar是必需传递的参数
        }
    }
};
</script>
```

课堂练习：设计一个自定义组件，实现计数器效果。

实现步骤：

1. 新建一个 `vue` 文件： `src/components/Counter.vue` 。实现计数器功能。
2. 在父组件 `Actor.vue` 中，引入该组件，显示计数器效果：

```
<counter></counter>
<counter></counter>
<counter></counter>
```

3. 为 `counter` 组件设计自定义属性，用于规约计数器的最小值、最大值等参数。

```
<counter :min="0" :max="10"></counter>
<counter :min="-5" :max="5"></counter>
<counter :min="1" :max="11" :step="2"></counter>
```

父子组件搭建页面的基本结构

Father.vue

```
<template>
  <div>
    <!-- views/Father.vue 父组件-->
    <h1>测试示例 - 自定义组件</h1>
    <counter></counter>
    <counter></counter>
  </div>
</template>
<script>
import Counter from '../components/Counter.vue'
export default {
    components: {
        Counter: Counter
    }
};
</script>
```

Counter.vue

```
<template>
  <div>
    <button @click="n--">-</button>
    {{n}}
    <button @click="n++">+</button>
  </div>
</template>
```

在页面中使用counter标签就相当于将子组件中的内容搬到父组件的相应位置，进行显示。

父组件向子组件传参：

Father.vue

```
<template>
    <div>
        <counter></counter>
        <counter />
        <counter :min="0" :max="5" />
    </div>
</template>
```

父组件中传递参数 ①

Counter.vue

③ template中使用属性

```
<template>
    <!-- Counter.vue 计数器组件 -->
    <button :disabled="n==min" ....>-</button>
    <span>{{n}}</span>
    <button :disabled="n==max" ....>+</button>
</template>

<script>
    export default {
        props : ['min', 'max']
    }
</script>
```

② 此处声明属性

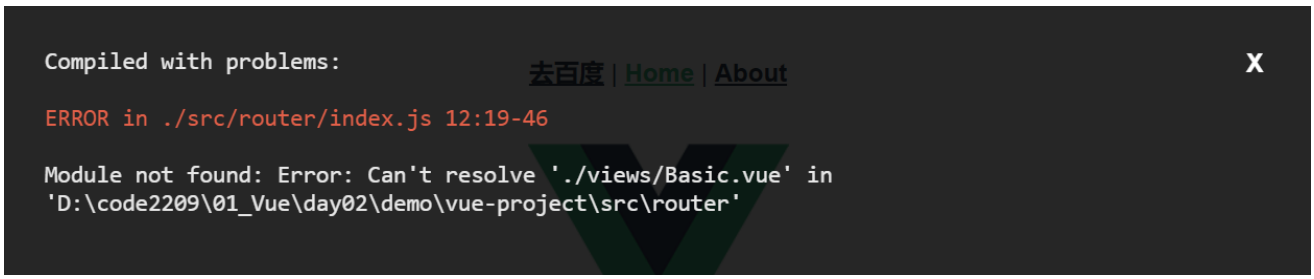
经典错误集合

```
▶[Vue warn]: Property or method "clickMe" is not defined on the instance but referenced during render. Make sure that this property is reactive, either in the data option, or for class-based components, by initializing the property. See: https://v2.vuejs.org/v2/guide/reactivity.html#Declaring-Reactive-Properties.

(found in <Root>)                                                                                    vue.js:5108
▶[Vue warn]: Invalid handler for event "click": got undefined

(found in <Root>)                                                                                    vue.js:5108
```

属性或方法`clickMe`并没有在vue对象中定义。要么在data段中声明，要么定义成组件的成员。



`Module not found: Error: Can't resolve './views/Basic.vue' in 'D:\code2209\01_Vue\day02\demo\vue-project\src\router`

模块找不到错误。好好检查代码中编写的这个路径是否可以找到对应的组件。当前这个错误就是找不到这个组件（`'./views/Basic.vue'`）而出的错误。