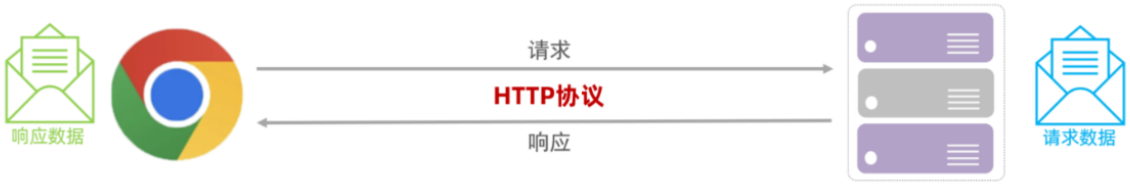
axios

以下为拓展知识

1、什么是请求？响应？



浏览器和服务器是按照HTTP协议进行数据通信的。它规定了通信的步骤、格式以及双方应该如何交互。

2、请求响应分析

```
https://search.bilibili.com/all?
keyword=%E5%A4%A7%E5%9B%BD%E5%B4%9B%E8%B5%B7&from_source=webtop_search&spm_id_fr
om=333.1007&search_source=5
```

大国崛起-哔哩哔哩_Bilibili

网址详情解析：

域名 -- 对应一个计算机ip，以及该 ip 代表的计算机中的 一个程序端口

ip -- 计算机的唯一标识，类似于人类的身份证号码 （服务器 == 计算机）

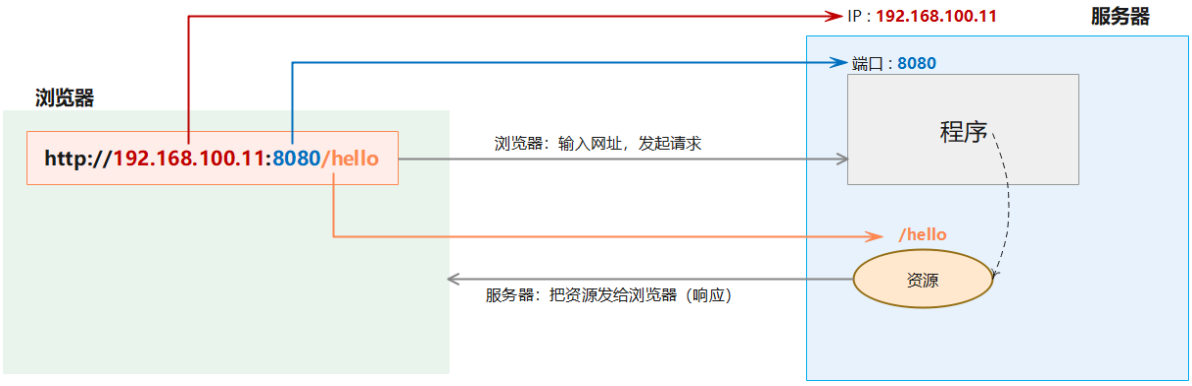
端口 -- 某一个程序的代号

https://search.bilibili.com **浏览器会遵循 https 协议 去访问 search.bilibili.com 这个域名对应的ip地址对应的服务器**

all 相当于 search.bilibili.com **对应的 ip 地址对应的服务器 上 哔哩哔哩后台系统的一个功能**

keyword=大国崛起&from_source=webtop_search&spm_id_from=333.1007&search_source=5

当我们通过浏览器去访问 哔哩哔哩 服务器后台的时候， 携带的参数



通过 http 协议 访问 192.168.100.11 对应的计算机 中的端口为 8080 的服务（程序），hello 是 8080 代表的程序中的一个功能（接口）

浏览器：

- 输入网址：`http://192.168.100.11:8080/hello`
  - 通过IP地址192.168.100.11定位到网络上的一台计算机

    > 我们之前在navicat中输入的localhost，就是127.0.0.1（本机）

  - 通过端口号8080找到计算机上运行的程序

    > `localhost:3306`，意思是在本地计算机中找到正在运行的3306端口的程序（一般是mysql数据库）

  - /hello是请求资源位置
    - 资源：对计算机而言资源就是数据
      - web资源：通过网络可以访问到的资源（通常是指存放在服务器上的数据）

    > `192.168.100.11:8080/hello`，意思是向ip为192.168.100.11的计算机中的8080端口程序，获取资源位置是/hello的数据

比喻理解：

http 假设是人与人之间的沟通规范，192.168.100.11 假设是银行的坐标地址，端口号8080 假设是银行的业务窗口，/hello就是业务窗口的某个业务。

服务器：

服务器可以理解为数据处理、提供中心，**就像现实生活中的银行**

> - 接收到浏览器发送的信息（如：/hello） **类似于银行接受到客户发起的取钱业务**
> - 在服务器上找到/hello的资源 **类似于从保险箱里面提取取钱的金额**
> - 把资源发送给浏览器 **类似于把取出来的钱给客户**

3、总结：

> 网络三要素：
>
> - IP ： 网络中计算机的唯一标识
> - 端口： 计算机中运行程序的唯一标识
> - 协议： 网络中计算机之间交互的规则
>
> **问题：浏览器和服务器两端进行数据交互，使用什么协议?**
>
> **答案：http协议** 、https协议

4、今天咱们课堂主要讲解GET、POST 请求

GET请求和POST请求都是HTTP协议中常用的两种请求方法，用于与服务器进行数据通信。它们在如何传递数据以及在何种情况下使用上有一些重要区别：

GET请求和POST请求的区别：

区别方式	GET请求	POST请求
请求参数	请求参数在请求行中。 例：`/brand/findAll?name=OPPO&status=1`	请求参数在请求体中
请求参数长度	请求参数长度有限制(浏览器不同限制也不同)	请求参数长度没有限制
安全性	安全性低。原因：请求参数暴露在浏览器地址栏中。	安全性相对高
常用于	常用于从服务器获取数据，参数通过URL的查询字符串传递。	常用于向服务器提交数据，数据传递在请求的主体中。

选择使用哪种请求方法取决于具体的业务需求和用例。一般来说，如果只是获取数据而不修改服务器状态，使用GET请求是合适的。如果需要向服务器提交敏感信息或执行可能会改变服务器状态的操作，使用POST请求更安全。

什么是协议？什么是ip？什么是端口？什么是接口？ get 、 post

正题开始：

一、什么是axios

Axios是一个基于Promise的**JavaScript库**（*Promise是JavaScript中用于处理异步操作的一种机制，它表示一个异步操作的最终结果。Promise可以有三种状态：pending（进行中）、fulfilled（已成功）和rejected（已失败）。*）

Axios用于发送HTTP请求。它可以在浏览器和Node.js环境中使用，并且具有许多功能，使得在客户端和服务器端进行数据通信变得更加简单和方便。

使用Axios可以更好地管理和处理HTTP请求，包括处理错误、设置请求超时、发送请求数据、接收响应数据等。它已经成为开发人员中广泛使用的HTTP请求库之一，并且在许多前端和后端项目中得到了广泛的应用。

二、使用场景

通常**用于网络请求的交互**，它**适用于前端和后端开发**，并且在许多不同的项目中都有应用。

发送HTTP请求：Axios可以用来发送各种类型的HTTP请求，包括GET、POST、PUT、DELETE等。无论是向服务器获取数据还是提交表单数据，Axios都可以处理。

拓展：

客户端 ： 提供给用户的操作界面。

当我们在客户端点击某一个操作按钮，其实会触发一个请求（发送请求），请求会顺着网络传递给服务器，服务器会去接受这个请求并处理（处理请求），处理完后进行返回请求结果（响应结果）。

服务器 ： 提供服务的机器。用来运行项目的电脑。

目前我们进行的axios操作就是实现，我们前端去向后端发送请求，并让后端响应结果，拿到后端响应的结果后，可以进行操作。

三、引入方式

1. 通过axios本地包导入（先要将工具包放到写代码的文件夹中）

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

2. 通过cdn的方式引入**(不建议)**

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

四、简单使用(项目已部署到服务器，http://43.136.104.16:3002/...访问)

- 步骤一、创建请求对象
- 步骤二、请求方式
- 步骤三、请求接口地址
- 步骤四、请求数据
- 步骤五、请求结果处理

```
<script>
    //传递一个配置对象作为参数，来创建一个Axios请求对象。该配置对象包含了请求的方法、URL以及其他可选的配置项
    axios({
      method: 'get',         // 发送 get 请求
      url: 'http://43.136.104.16:3002/xx接口',
    }).then((res) => {
      // 处理成功的结果
    }).catch((error) => {
      // 处理错误
    })
   /*    这段代码使用Axios发送了一个GET请求到http://43.136.104.16:3002/xx接口，并在控制台中打印响应结果。
    .then()方法用于处理成功状态。
    .catch()方法用于处理失败状态。                          */
</script>
```

五、请求参数

参数	作用
method	请求方式 默认为GET
url	请求地址
data	**data是作为请求主体被发送的数据 只适用于`post`请求**
params	**是即将与请求一起发送的 URL 参数 适用`get`请求**
timeout	请求时间
.then()	回调函数 用于获取服务器返回的数据
catch	用于处理失败状态

六、使用接口对数据进行增删改查(项目到服务器 可远程调用)

接口文档（指的是对某一个服务器上某一个服务的具体功能的介绍）：http://43.136.104.16/axios

接口地址：http://43.136.104.16:3002/...

1. 增

   http://43.136.104.16:3002/xinzeng

   通过http协议，访问ip为43.136.104.16的服务器，中端口为3000的服务（程序），xinzeng 是这个程序的一个功能，这个功能能够实现用户新增。

   新增过程用post请求方式携带参数，这些参数是程序开发者早就设定好的，用人家的接口就得按照人家的要求传参。

```
axios({
    method : 'post',
    url : 'http://43.136.104.16:3002/xinzeng',
    data : {
        name : '周瑜',
        age : 18
    }
}).then((res)=>{    // 处理成功的结果
    console.log(res.data.message);
})
```

2. 删

```
axios({
    method : 'get',
    url : "http://43.136.104.16:3002/shanchu",
    params : {
        id : 253
    }
}).then((res)=>{    // 处理成功的结果
    console.log(res.data.message);
})
```

3. 改

```
axios({
    method : 'get',
    url : 'http://43.136.104.16:3002/xiugai',
    params : {
        id : 254,
        name : '不渝'
    }
}).then((res)=>{    // 处理成功的结果
    console.log(res.data.message);
})
```

4. 查

```
// alert("进行查询")
axios({
    methos : 'get',
    url : 'http://43.136.104.16:3002/chaxun'
}).then((res)=>{    // 处理成功的结果
    console.log(res.data);
})
```

综合实例：

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <script src="vue.js"></script>
    <!-- 导入axios包 -->
    <script src="axios.js"></script>
    <!-- css 内部样式  -网页皮囊 -->
    <style>
        /* 去掉浏览器的白边 */
```

```
        body {
            padding: 0px;
            margin: 0px;
        }
        /* flex容器 */
        .box {
            display: flex;
            width: 100%;
            height: 100vh;      /*可视化高度*/
            background-color: palevioletred;
        }
        /* flex项目 */
        #div1 {
            margin: auto;    /*结合flex容器当中的代码可以实现居中效果*/
        }
        /* 设定输入框样式 */
        input {
            font-size: 30px;
        }
        /* 设定按钮样式 */
        button {
            font-size: 30px;
            background-color: grey;
            border-radius: 20%;
            color: white;
        }
    </style>
</head>

<body>
    <!-- html代码 -网页骨架   -->
    <div class="box">
        <div id="div1">
            <h1>增删改查：</h1>
            <!-- 输入框都使用 v-model 进行了双向绑定，绑定到了挂载的Vue对象中data数据源中
的键 -->
            <!-- 按钮都填了 @click点击事件，每个事件对应Vue对象方法区中的一个具体的函数 -->
            <input type="text" v-model="add_username" placeholder="请输入新增的用户
昵称">
            <input type="text" v-model="add_age" placeholder="请输入新增的用户年龄">
            <button @click="add">增</button>
            <br><br>
            <input type="text" v-model="del_id" placeholder="请输入被删除的用户id">
            <button @click="del">删</button>
            <br><br>
            <input type="text" v-model="upd_username" placeholder="请输入修改后的用
户昵称">
            <input type="text" v-model="upd_id" placeholder="请输入被修改的用户id">
            <button @click="upd">改</button>
            <br><br>
            <button @click="sel" style="background-color: black;">查</button>
            <br>
            <!-- 插值法，用于渲染挂载的Vue对象中data数据源中的数据 -->
            {{select_data}}
            <!-- v-for循环，用于遍历序列类型数据的 -->
```

```
            <li v-for="(x,index) in select_data" :key="index">{{x}}</li>
        </div>
    </div>

    <!-- js代码 -网页灵魂 -->
    <script>
        // 创建了一Vue对象并且挂载到了 id选择器为div1的盒子上
        new Vue({
            el: "#div1",
                //数据源
            data: {
                del_id: null,
                add_username: null,
                add_age: null,
                upd_username: null,
                upd_id: null,
                select_data: null   // 用来存储查询到的数据合集
            },
                // 方法区，函数区
            methods: {
                add() {    //新增方法
                    axios({
                        method : 'post',
                        url : 'http://43.136.104.16:3002/xinzeng',
                        data : {
                            name : this.add_username,
                            age : this.add_age
                        }
                    }).then((res)=>{    // 处理成功的结果
                        console.log(res.data.message);
                    })
                },
                del() {   // 删除方法
                    axios({
                        method : 'get',
                        url : "http://43.136.104.16:3002/shanchu",
                        params : {
                            id : this.del_id
                        }
                    }).then((res)=>{    // 处理成功的结果
                        console.log(res.data.message);
                    })
                },
                upd() {   //修改方法
                    axios({
                        method : 'get',
                        url : 'http://43.136.104.16:3002/xiugai',
                        params : {
                            id : this.upd_id ,
                            name : this.upd_username
                        }
                    }).then((res)=>{    // 处理成功的结果
                        console.log(res.data.message);
                    })
                },
```

```
                sel() {   // 查询方法
                    axios({
                        methos : 'get',
                        url : 'http://43.136.104.16:3002/chaxun'
                    }).then((res)=>{    // 处理成功的结果
                        console.log(res.data);
                        this.select_data = res.data
                    })
                }
            }
        })

        /*
            前后端交互需求：
                ！！！注意：我们是以前端的角度去进行交互，而不是后端，咱们暂时没有学习后端。！！！
                交互步骤：
                    - 步骤一、创建请求对象
                    - 步骤二、请求方式
                    - 步骤三、请求接口地址
                    - 步骤四、请求数据
                    - 步骤五、请求结果处理
            新增方法:
                需求一：当在输入框输入了用户昵称和年龄后，点击新增按钮，能够实现用户新增
                需求二：新增完后触发自动查询
            删除方法:
                需求一：用户从输入框输入 用户 id 后，点击删除按钮，可以对该条信息进行删除操作
                需求二：删除完后触发自动查询
            修改方法:
                需求一：当用户输入 被更改用户 id 和 修改后的用户昵称信息后 ，点击修改按钮，能够修改成功
                需求二：修改完后触发自动查询
            查询方法:
                需求一：当用户点击查询按钮，能够查询的到目前服务器中保存的所有用户信息
        */
    </script>
</body>

</html>
```