**第一章：原生AJAX**

* 1. AJAX 简介

AJAX的全称为Asynchronous JavaScript And XML，就是异步的JS和XML；通过AJAX可以在浏览器中向服务器发送异步请求，最大的优势是：**无刷新获取数据**；AJAX不是新的编程语言，而是一种将现有的标准组合在一起使用的新方式。

* 1. XML 简介

XML 可扩展标记语言

XML 被设计用来传输和存储数据

XML和HTML类似，不同的是：

1. HTML是用来展示数据，而XML是用来存储和传输数据
2. HTML中都是预定义标签，而XML中没有预定义标签，全都是自定义标签，用来表示一些数据
3. 例如：我有一个学生数据

name: ‘Tom’ , age: 20

用XML表示：

<student>

<name>Tom</name>

<age>20</age>

</student>

* 1. AJAX 特点
     1. AJAX 的优点

1）可以无需刷新页面而与服务器进行通信

2）允许根据用户事件来更新页面内容

* + 1. AJAX的缺点

1）没有浏览历史，不能回退

2）存在跨域问题（同源）

3）SEO不友好

**第二章：HTTP协议**

2.1 HTTP简介

HTTP全称为hypertext transport protocol，超文本传输协议，协议详细规定了浏览器和万维网之间互相通信的规则

2.2 请求报文

重点：格式与参数

四个部分：

|  |  |
| --- | --- |
| 行 | POST（请求类型）  s?ie=utf-8 （URL）  HTTP/1.1（HTTP版本） |
| 头：（格式：名字/冒号/空格/参数） | Host: atguigu.com  Cookie: name=guigu  Content-type:application/x-www-form-urlencoded  User-Agent: chrome 83 |
| 空行 |  |
| 体 | username=admin&password=admin |

注意：如果是GET请求，体没有内容；如果是POST请求，体可以不为空

2.3 响应报文

重点：格式与参数

四个部分：

|  |  |
| --- | --- |
| 行 | HTTP/1.1 200（响应状态码）OK（响应状态字符串） |
| 头：（格式：名字/冒号/空格/参数） | Content-type: text/html;charset=utf-8  Content-length: 2048  Content-encoding: gzip |
| 空行 |  |
| 体 | <html>  <head>  <head>  <body>  <h1>尚硅谷</h1>  <body>  </html> |

其他常见响应状态码：404，403，401，500

2.4 使用chrome调试工具查看请求和响应

Network --> 请求 --> Headers/Request Headers & Form Data（GET请求还可以查看Query String Parameters，查看URL请求参数解析）

Network --> 响应 --> Headers/Response Headers & Response

1. **原生AJAX的使用（Node.js环境）**
   1. 核心对象：XMLHttpRequest, AJAX所有操作都是通过该对象进行的
   2. 使用步骤
2. 创建XMLHttpRequest对象

var xhr = new XMLHttpRequest()

1. 设置请求信息

xhr.open(method , url)

//可以设置请求头，一般不设置

Xhr.setRequestHeader(‘Content-Type’ , ‘application/x-www-form-urlencoded’)

1. 发送请求

Xhr.send(body)

//get请求不传body参数，只有post请求使用

1. 接收响应

//xhr.status 状态码

//xhr.statusText 状态码字符串

//xhr.getAllResponseHeader() 所有响应头

//xhr.response 响应体

xhr.onreadystatechange = function(){

If(xhr.readyState === 4 ){

If(xhr.status >= 200 && xhr.status < 300){

console.log(xhr.response)

}

}

}

1. jQuery中的AJAX（略）

4.1 get请求

4.2 post请求

4.3 通用型方法AJAX

1. axios 发送 AJAX（略）
2. fetch()函数 发送 AJAX（用的少，略）

fetch()函数是全局对象window的一个方法，可以直接调用，返回一个promise对象

1. **同源策略**

7.1 同源策略是什么？

同源策略（Same-Origin Policy）最早由Netscape公司提出，是浏览器的一种安全策略；发送AJAX请求默认遵循同源策略

同源：协议、域名、端口号 必须完全相同

例如：

发送请求的是：http协议、a.com、8000

接收请求的是：http协议、a.com、8000

违背同源策略就是**跨域**

7.2 如何解决跨域

7.2.1 JSONP

1. JSONP是什么？

JSONP (JSON with Padding)，是一个非官方的跨域解决方案，纯粹凭借程序员的聪明才智开发出来，只支持get请求

1. JSONP的工作原理

在网页中有一些标签天生具有跨域能力，比如：img link iframe script

JSONP就是利用script标签的跨域能力来发送请求的

1. JSONP的使用（实例：检测用户名是否存在）

A：在客户端提前声明一个函数，把接收响应的行为封装在函数内

 //声明handle函数

        function handle (data){

            //设置input的边框

            input.style.border = 'solid 1px #f00'

            //修改p标签的提示文本

            p.innerHTML = data.msg

        }

B: 动态创建script标签，设置src发送请求，并把script标签插入到body中

  //绑定失去焦点监听

        input.onblur = function(){

            //获取用户的输入值

            var username = this.value

            //向服务器发送请求 检测用户名是否存在

            //1. 创建script标签

            var script = document.createElement("script")

            //2. 设置src

            script.src = 'http://127.0.0.1:8000/check-username

            //3. 将script标签插入到body中

            document.body.appendChild(script)

        }

C: 服务器返回函数调用（JS代码），并将需要传输的数据作为实参传入

//check-username 服务

app.all('/check-username' , (request , response)=>{

    const data = {

        exist: 1,

        msg: '用户名已经存在'

    }

    var str = JSON.stringify(data)

    response.end(`handle(${str})`)

})

7.2.2 CORS

1. CORS是什么？

CORS（Cross-Origin Resource Sharing），跨域资源共享。CORS是官方的跨域解决方案，它的特点是不需要在客户端做任何特殊操作，完全在服务器中进行处理，支持get和post请求。跨域资源共享标准新增了一组HTTP首部字段，允许服务器声明哪些源站通过浏览器有权限访问哪些资源

1. CORS工作原理

CORS是通过设置一个响应头来告诉浏览器，该请求允许跨域，浏览器收到该响应就会对响应放行

//设置允许跨域

    //设置任何url都可以访问本服务

    response.setHeader('Access-Control-Allow-Origin' , '\*')

    //设置请求可以自定义请求头

    response.setHeader('Access-Control-Allow-Headers' , '\*')

    //设置接收任意类型请求

    response.setHeader('Access-Control-Allow-Method' , '\*')

发生问题及解决方案：

1. 下载express
2. 安装nodemon
3. 解决重复发送AJAX请求问题

在btn.onclick 事件函数中用var xhr = new XMLHttpResquest()  
xhr就是函数中的局部变量，每次点击按钮调用函数，都会声明一个新的xhr变量。  
第一次调用函数时的xhr变量与第二次调用函数时的xhr变量不是同一个。  
并且变量有声明提升的特性，  
var xhr = new XMLHttpResquest()中的var xhr声明在函数执行前会提升到函数中的最前面，变成：  
        btn.onclick = function(){  
            var xhr;  
            //判断：点击按钮，如果正在发送请求，就取消该请求  
            if(isSending){  
                xhr.abort()  
            }  
            xhr = new XMLHttpRequest()

 这样在执行xhr.abort()时 xhr的值是 undefined

附：Express框架的基本使用

* 1. 下载express
  2. 引入express