# 1.原生Ajax

## 需要创建应该XMLHttpRequest对象，然后需要配置，在发送请求获取数据

## 书写步骤

|  |
| --- |
|  |

## 注意：这是浏览器端代码，用nodejs执行会报错

## 响应处理流程

|  |
| --- |
|  |
|  |

## 可能这里也需要使用json-server作为后端来测试或者使用jsonplaceholder网站：<https://jsonplaceholder.typicode.com/>

## 或者教师先生接口：<https://api.oioweb.cn/>

## 或者：<https://api.uomg.com/>

## 或者：[韩小韩Web API接口站 - 免费Web API数据接口调用服务平台 (vvhan.com)](https://api.vvhan.com/)

## 或者：<https://cache.video.iqiyi.com/jp/avlist/202861101/1/>

## 或者：[fangzesheng/free-api: 收集免费的接口服务,做一个api的搬运工 (github.com)](https://github.com/fangzesheng/free-api?tab=readme-ov-file#634)

## 我们来实践一下，为了方便测试，我创建了一个本地服务器：[http server](笔记08-http服务器代码-测试用.docx)

## .最基本的原生ajax代码，创建一个ajax对象

### 新建一个ajax文件夹，在里面新建应该1-ajaxbasic.js文件和应该test.html文件，

|  |
| --- |
|  |

### 1-ajaxbasic.js的内容如下

|  |
| --- |
| function createAjax(){      let xhr      try {           xhr = new XMLHttpRequest()      } catch (error) { //ie           xhr = new ActiveXObject('microsoft.XMLHttp')      }     return xhr  }  let ajax = createAjax()  console.log(ajax); |

### test.html的内容

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <meta name="viewport" content="width=device-width, initial-scale=1.0">      <title>Document</title>  </head>  <body>      <script src="./1-ajaxbasic.js"></script>  </body>  </html> |

### 然后使用vscode的preview server插件运行test.html文件，效果如下

|  |
| --- |
|  |

### 说明ajax对象创建成功

## 为了便于调试，我们还是使用httpserver项目来学习ajax。

## 1>.原生ajax发送get请求

### 1）打开httpserver项目，如图

|  |
| --- |
|  |

### 2.）打开assets/html/index.html，把ul标签以及子标签删除，添加一个按钮和一个div标签，我们想点击按钮，把数据加载到div里面，我们在修改一下代码

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <meta name="viewport" content="width=device-width, initial-scale=1.0">      <title>首页</title>      <link rel="stylesheet" href="../css/index.css">  </head>  <body>      <h1>欢迎来到首页</h1>      <a href="/detail.html">          详情页      </a>      <hr>      <div id="box">          <button id="btn">获取数据</button>          <div id="content"></div>      </div>      <script src="../js/index.js"></script>  </body>  </html> |

### 3）也需要修改一些index.css的代码

|  |
| --- |
| \*{      padding: 0;      margin: 0;  }  hr {      border: 1px solid deeppink;      margin: 5px;  }  #box{      margin: 10px;      width: 500px;      height: 300px;      border: 1px solid green;      padding: 10px;      font-size: xx-large;  }  ul{      list-style: none;  } |

### 4）在index.js中获取按钮对象，标签给他绑定事件，先写一些简单代码测试一下

|  |
| --- |
| let oBtn = document.getElementById("btn")  oBtn.onclick = function(){      alert("btn clicked");  } |

### 测试一下，没有问题

|  |
| --- |
|  |

### 5）在index.js中编写ajax请求代码，先做get请求

|  |
| --- |
| let oBtn = document.getElementById("btn")  oBtn.onclick = function(){      // alert("btn clicked");      //1.创建对象      let xhr = new XMLHttpRequest()      //2.设置请求路径      xhr.open('GET','/get\_data')      //3.绑定监听状态改变的事件处理函数      xhr.onreadystatechange = ()=>{          //  console.log(xhr.readyState); //会输出3个值，分别是2，3，4因为只要状态改变这个函数都能够监听到          if(xhr.readyState==4 && xhr.status === 200){              console.log(xhr.responseText);          }      }      //4.发送请求      xhr.send()  } |

#### 注意：因为此时我们没有路由处理这个请求，得到的是404

|  |
| --- |
|  |

### 6）在myserver.js中添加对/get\_data路径的处理函数

|  |
| --- |
| let http = require('http')  let fs = require('fs')  let path = require('path')  let PORT = 8081  let server = http.createServer((request,response)=>{      //判断浏览器需要哪个资源      let reqUrl = request.url      if(reqUrl==='/' || reqUrl==='/index.html'){          //读取页面内容,返回信息          let filePath = path.join(\_\_dirname,'assets','html','index.html')          let content = fs.readFileSync(filePath)          response.end(content)      } else if(reqUrl==='/detail.html'){          //读取页面内容,返回信息          let filePath = path.join(\_\_dirname,'assets','html','detail.html')          let content = fs.readFileSync(filePath)          response.end(content)      } else if(reqUrl==='/get\_data'){ //处理/get\_data          let data = {              "name":"kenny",              "position":"boss"          }          response.end(JSON.stringify(data))      } else if(reqUrl.endsWith('.js')){          //读取页面内容,返回信息          let filePath = path.join(\_\_dirname,'assets','js','index.js')          let content = fs.readFileSync(filePath)          response.end(content)      } else if(reqUrl.endsWith('.css')){          //读取页面内容,返回信息          let filePath = path.join(\_\_dirname,'assets','css','index.css')          let content = fs.readFileSync(filePath)          response.end(content)      } else {           response.setHeader('Content-type','text/html;charset=utf-8')           response.end('Page Not Found')      }  })  server.listen(PORT,()=>{      console.log(`server is ready at : http://localhost:${PORT}/`);  }) |

#### 效果

|  |
| --- |
|  |

### 7）添加渲染代码，同时修改index.css一点点

|  |
| --- |
| //index.js  let oBtn = document.getElementById("btn")  oBtn.onclick = function(){      // alert("btn clicked");      //1.创建对象      let xhr = new XMLHttpRequest()      //2.设置请求路径      xhr.open('GET','/get\_data')      //3.绑定监听状态改变的事件处理函数      xhr.onreadystatechange = ()=>{          //  console.log(xhr.readyState); //会输出3个值，分别是2，3，4因为只要状态改变这个函数都能够监听到          if(xhr.readyState==4 && xhr.status === 200){              console.log(xhr.responseText);              //把内容渲染到div              let content = document.getElementById("content")              content.innerHTML = xhr.responseText          }      }      //4.发送请求      xhr.send()  } |
| \*{      padding: 0;      margin: 0;  }  hr {      border: 1px solid deeppink;      margin: 5px;  }  #box{      margin: 10px;      width: 550px;      height: 300px;      border: 1px solid green;      padding: 10px;      font-size: xx-large;  }  ul{      list-style: none;  } |

#### 效果

|  |
| --- |
|  |

## 2>原生ajax发送post请求

### 1）为了方便学习，我们实现一个模拟的登录功能，我们在assets/html/里面新建一个login.html,内容如下

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>      <meta charset='UTF-8' />      <title>用户登录</title>  </head>  <body>      <form method="post" action="/login">          <p>账号：<input type="text" name="name" size="10"></p>          <p>密码：<input type="password" name="password" size="10"></p>          <p><input type="submit" value="确定">              <input type="reset" value="取消">          </p>      </form>  </body>  </html> |

### 2）由于我们是使用Ajax来实现提交，我们需要把form表单的action和method属性删除，同时需要把submit按钮改为普通按钮

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>      <meta charset='UTF-8' />      <title>用户登录</title>  </head>  <body>      <form>           <!--使用ajax技术提交，这里不用写action和method属性 需要就submit按钮改为普通按钮，因为普通按钮不会提交 -->          <p>账号：<input type="text" name="name" size="10"></p>          <p>密码：<input type="password" name="password" size="10"></p>          <p><input type="button" value="确定">              <input type="reset" value="取消">          </p>      </form>      <!-- ajax提交 -->  </body>  </html> |

### 3）我们发现我们的服务器程序myserver.js有很多重复代码，我们来优化一下，把重复代码抽取出来作为一个doResponse方法，在有需要的地方调用即可

|  |
| --- |
| let http = require('http')  let fs = require('fs')  let path = require('path')  let PORT = 8081  function doResponse(response,dir,fileName){      let filePath = path.join(\_\_dirname,'assets',dir,fileName)      let content = fs.readFileSync(filePath)      response.end(content)  }  let server = http.createServer((request,response)=>{      //判断浏览器需要哪个资源      let reqUrl = request.url      if(reqUrl==='/' || reqUrl==='/index.html'){          //读取页面内容,返回信息          doResponse(response,"html","index.html")      } else if(reqUrl==='/detail.html'){          //读取页面内容,返回信息          doResponse(response,"html","detail.html")      } else if(reqUrl==='/login.html'){ //处理/login.html         //读取页面内容,返回信息          doResponse(response,"html",'login.html')      } else if(reqUrl==='/get\_data'){ //处理/get\_data          let data = {              "name":"kenny",              "position":"boss"          }          response.end(JSON.stringify(data))      }else if(reqUrl==='/login'){ //处理/get\_data            response.end()      }      else if(reqUrl.endsWith('.js')){          //读取页面内容,返回信息          doResponse(response,'js','index.js')      } else if(reqUrl.endsWith('.css')){          //读取页面内容,返回信息         doResponse(response,'css','index.css')      } else {           response.setHeader('Content-type','text/html;charset=utf-8')           response.end('Page Not Found')      }  })  server.listen(PORT,()=>{      console.log(`server is ready at : http://localhost:${PORT}/`);  }) |

#### 注意： <input type="button" value="确定">这样子写是没有提交行为的，但是<button>提交</button>是由提交行为的

### 4）我们在login.html里面添加一个script标签，先写一下测试代码

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>      <meta charset='UTF-8' />      <title>用户登录</title>  </head>  <body>      <form>          <!--使用ajax技术提交，这里不用写action和method属性 需要就submit按钮改为普通按钮，因为普通按钮不会提交 -->          <p>账号：<input type="text" name="name" size="10"></p>          <p>密码：<input type="password" name="password" size="10"></p>          <p>              <input type="button" value="确定" id="btnLogin">          </p>      </form>      <script>          let objLogin = document.getElementById("btnLogin")          objLogin.onclick = () => {              alert("login button clicked...")          }      </script>  </body>  </html> |

### 测试一下，效果如下

|  |
| --- |
|  |

### 5）然后我们添加一个div用来显示服务器返回结果，然后我们尝试获取这个div在按钮事件里面修改他的内容

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>      <meta charset='UTF-8' />      <title>用户登录</title>  </head>  <body>      <form>          <!--使用ajax技术提交，这里不用写action和method属性 需要就submit按钮改为普通按钮，因为普通按钮不会提交 -->          <p>账号：<input type="text" name="name" size="10"></p>          <p>密码：<input type="password" name="password" size="10"></p>          <p>              <input type="button" value="确定" id="btnLogin">          </p>      </form>      <div id="result"></div>      <script>          let objLogin = document.getElementById("btnLogin")          let result = document.getElementById("result")          objLogin.onclick = () => {              // alert("login button clicked...")              result.innerHTML = "btn clicked..."          }      </script>  </body>  </html> |

#### 效果如下：

|  |
| --- |
|  |

### 6）在按钮的点击事件里面获取表单数据，我们先把获取到的数据在控制台输出

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>      <meta charset='UTF-8' />      <title>用户登录</title>  </head>  <body>      <form>          <!--使用ajax技术提交，这里不用写action和method属性 需要就submit按钮改为普通按钮，因为普通按钮不会提交 -->          <p>账号：<input type="text" name="name" id="username"></p>          <p>密码：<input type="password" name="password" id="pwd"></p>          <p>              <input type="button" value="确定" id="btnLogin">          </p>      </form>      <div id="result"></div>      <script>          let objLogin = document.getElementById("btnLogin")          let result = document.getElementById("result")          objLogin.onclick = () => {              // alert("login button clicked...")              // result.innerHTML = "btn clicked..."              //获取表单数据              let username = document.getElementById("username").value              let pwd = document.getElementById("pwd").value              console.log(username, pwd);          }      </script>  </body>  </html> |

#### 注意：需要给两个input标签添加id属性

#### 效果如下：点击按钮是可以获取到用户数据的

|  |
| --- |
|  |

### 7）封装数据，编写ajax请求代码(4步)然后在最后异步的send方法中利用xhr发送给服务器，需要先把它变为字符串，标签获取响应数据，在div显示，注意这个处理是在第3步里面做的

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>      <meta charset='UTF-8' />      <title>用户登录</title>  </head>  <body>      <form>          <!--使用ajax技术提交，这里不用写action和method属性 需要就submit按钮改为普通按钮，因为普通按钮不会提交 -->          <p>账号：<input type="text" name="name" id="username"></p>          <p>密码：<input type="password" name="password" id="pwd"></p>          <p>              <input type="button" value="确定" id="btnLogin">          </p>      </form>      <div id="result"></div>      <script>          let objLogin = document.getElementById("btnLogin")          let result = document.getElementById("result")          objLogin.onclick = () => {              // alert("login button clicked...")              // result.innerHTML = "btn clicked..."              //获取表单数据              let username = document.getElementById("username").value              let pwd = document.getElementById("pwd").value              // console.log(username, pwd);              //封装数据              let params = { username, pwd }              //ajax处理(4个步骤)              //1.创建ajax（XMLHttpRequest）对象              let xhr = new XMLHttpRequest()              //2.设置访问路径和请求方法              xhr.open('POST', '/login')              //3.监听状态改变的数据处理函数              xhr.onreadystatechange = () => {                  if (xhr.readyState === 4 && xhr.status === 200) {                      result.innerHTML = xhr.responseText                  }              }              //4.发送请求              xhr.send(JSON.stringify(params))          }      </script>  </body>  </html> |

### 8）ajax处理代码写完，我们回到myserver.js中添加对/login路由的处理然后返回数据给用户

|  |
| --- |
| let http = require('http')  let fs = require('fs')  let path = require('path')  let PORT = 8081  //定义两个数据用来模拟数据库的记录，如果post传递过来的数据和这两个数据一样，就算是登录成功  const USER = "admin"  const PASS = '12345'  function doResponse(response,dir,fileName){      let filePath = path.join(\_\_dirname,'assets',dir,fileName)      let content = fs.readFileSync(filePath)      response.end(content)  }  let server = http.createServer((request,response)=>{      //判断浏览器需要哪个资源      let reqUrl = request.url      if(reqUrl==='/' || reqUrl==='/index.html'){          //读取页面内容,返回信息          doResponse(response,"html","index.html")      } else if(reqUrl==='/detail.html'){          //读取页面内容,返回信息          doResponse(response,"html","detail.html")      } else if(reqUrl==='/login.html'){ //处理/login.html         //读取页面内容,返回信息          doResponse(response,"html",'login.html')      } else if(reqUrl==='/get\_data'){ //处理/get\_data          let data = {              "name":"kenny",              "position":"boss"          }          response.end(JSON.stringify(data))      }else if(reqUrl==='/login'){ //处理/get\_data          // console.log(request);          let data = ''          request.on('data',(chunk)=>{ //nodejs请求对象处理post添加的数据的方法，需要配合下面的end数件              data += chunk          })          request.on('end',()=>{              // console.log(data);              let { username,pwd } = JSON.parse(data)              // console.log(username,pwd);              //利用上面定义的两个常量来模拟用户验证              if(username === USER && pwd === PASS){                  response.end(`登录成功：欢迎您:${username}`)              } else {                  response.end("登录失败，用户名或密码错误")              }            })        }      else if(reqUrl.endsWith('.js')){          //读取页面内容,返回信息          doResponse(response,'js','index.js')      } else if(reqUrl.endsWith('.css')){          //读取页面内容,返回信息         doResponse(response,'css','index.css')      } else {           response.setHeader('Content-type','text/html;charset=utf-8')           response.end('Page Not Found')      }  })  server.listen(PORT,()=>{      console.log(`server is ready at : http://localhost:${PORT}/`);  }) |

#### 注意：post请求传递过来的字符串在request的on方法里面接收到的是一个Buffer类型，需要转化为字符串，如用一个字符串和它相加

## 3>避免缓存问题和超时处理的问题

### 避免缓存问题的3种方法

|  |
| --- |
|  |

## 我们测试第一种方法，

### 1)修改一下index.js文件，在发送请求的时候在路径后面+Math.random()

|  |
| --- |
| let oBtn = document.getElementById("btn")  oBtn.onclick = function(){      // alert("btn clicked");      //1.创建对象      let xhr = new XMLHttpRequest()      //2.设置请求路径      // xhr.open('GET','/get\_data') //普通方法      xhr.open('GET','/get\_data'+Math.random()) //使得每一次请求都不一样      //3.绑定监听状态改变的事件处理函数      xhr.onreadystatechange = ()=>{          //  console.log(xhr.readyState); //会输出3个值，分别是2，3，4因为只要状态改变这个函数都能够监听到          if(xhr.readyState==4 && xhr.status === 200){              console.log(xhr.responseText);              //把内容渲染到div              let content = document.getElementById("content")              content.innerHTML = xhr.responseText          }      }      //4.发送请求      xhr.send()  } |

### 2）然后修改myserver.js,把 else if(reqUrl==='/get\_data')改为else if(reqUrl.startsWith('/get\_data'))

|  |
| --- |
| let http = require('http')  let fs = require('fs')  let path = require('path')  let PORT = 8081  //定义两个数据用来模拟数据库的记录，如果post传递过来的数据和这两个数据一样，就算是登录成功  const USER = "admin"  const PASS = '12345'  function doResponse(response,dir,fileName){      let filePath = path.join(\_\_dirname,'assets',dir,fileName)      let content = fs.readFileSync(filePath)      response.end(content)  }  let server = http.createServer((request,response)=>{      //判断浏览器需要哪个资源      let reqUrl = request.url      if(reqUrl==='/' || reqUrl==='/index.html'){          //读取页面内容,返回信息          doResponse(response,"html","index.html")      } else if(reqUrl==='/detail.html'){          //读取页面内容,返回信息          doResponse(response,"html","detail.html")      } else if(reqUrl==='/login.html'){ //处理/login.html         //读取页面内容,返回信息          doResponse(response,"html",'login.html')       }       // else if(reqUrl==='/get\_data'){ //处理/get\_data      //     let data = {      //         "name":"kenny",      //         "position":"boss"      //     }      //     response.end(JSON.stringify(data))      // }      else if(reqUrl.startsWith('/get\_data')){ //处理/get\_data+Math.random()              let data = {                  "name":"kenny",                  "position":"boss"              }              response.end(JSON.stringify(data))          }        else if(reqUrl==='/login'){ //处理/login          // console.log(request);          let data = ''          request.on('data',(chunk)=>{ //nodejs请求对象处理post添加的数据的方法，需要配合下面的end数件              data += chunk          })          request.on('end',()=>{              // console.log(data);              let { username,pwd } = JSON.parse(data)              // console.log(username,pwd);              //利用上面定义的两个常量来模拟用户验证              if(username === USER && pwd === PASS){                  response.end(`登录成功：欢迎您:${username}`)              } else {                  response.end("登录失败，用户名或密码错误")              }            })        }      else if(reqUrl.endsWith('.js')){          //读取页面内容,返回信息          doResponse(response,'js','index.js')      } else if(reqUrl.endsWith('.css')){          //读取页面内容,返回信息         doResponse(response,'css','index.css')      } else {           response.setHeader('Content-type','text/html;charset=utf-8')           response.end('Page Not Found')      }  })  server.listen(PORT,()=>{      console.log(`server is ready at : http://localhost:${PORT}/`);  }) |

### 效果：也是可以的

|  |
| --- |
|  |

### 这里学习到了一点：任何响应动态路径，其中一种方法就是用startsWith方法主要这个路径的全面部分是不变的，不管后面怎么变化都是可以匹配到的

## 用时间戳的方法和上面的方法大同小异，只需要在index.js里面修改代码，服务器不需要修改，效果是一样的

|  |
| --- |
| let oBtn = document.getElementById("btn")  oBtn.onclick = function(){      // alert("btn clicked");      //1.创建对象      let xhr = new XMLHttpRequest()      //2.设置请求路径      // xhr.open('GET','/get\_data') //普通方法      // xhr.open('GET','/get\_data'+Math.random()) //使得每一次请求都不一样,随机数方法      xhr.open('GET','/get\_data'+new Date().getTime()) //使得每一次请求都不一样，时间戳方法      //3.绑定监听状态改变的事件处理函数      xhr.onreadystatechange = ()=>{          //  console.log(xhr.readyState); //会输出3个值，分别是2，3，4因为只要状态改变这个函数都能够监听到          if(xhr.readyState==4 && xhr.status === 200){              console.log(xhr.responseText);              //把内容渲染到div              let content = document.getElementById("content")              content.innerHTML = xhr.responseText          }      }      //4.发送请求      xhr.send()  } |

## 现在我们来使用第三种方法，把index.js的代码改回来，任何在发送请求之前设置一个cache属性

|  |
| --- |
| let oBtn = document.getElementById("btn")  oBtn.onclick = function(){      // alert("btn clicked");      //1.创建对象      let xhr = new XMLHttpRequest()      //2.设置请求路径      xhr.open('GET','/get\_data') //普通方法      // xhr.open('GET','/get\_data'+Math.random()) //使得每一次请求都不一样,随机数方法      // xhr.open('GET','/get\_data'+new Date().getTime()) //使得每一次请求都不一样，时间戳方法      //3.绑定监听状态改变的事件处理函数      xhr.onreadystatechange = ()=>{          //  console.log(xhr.readyState); //会输出3个值，分别是2，3，4因为只要状态改变这个函数都能够监听到          if(xhr.readyState==4 && xhr.status === 200){              console.log(xhr.responseText);              //把内容渲染到div              let content = document.getElementById("content")              content.innerHTML = xhr.responseText          }      }      //4.发送请求      //4.1 关闭缓存功能,不是必须的      xhr.setRequestHeader('Cache-Control','no-cache')      //4.2 发送请求      xhr.send()  } |

### 服务器不需要修改代码，也是可以正常工作的

## 其实对应开发者，还有一个方法，就是手动在浏览器的调试工具里面禁用缓存

|  |
| --- |
|  |

## 超时处理

|  |
| --- |
|  |

## 我们用login页面来尝试超时处理打开login.html,添加下面的代码

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>      <meta charset='UTF-8' />      <title>用户登录</title>  </head>  <body>      <form>          <!--使用ajax技术提交，这里不用写action和method属性 需要就submit按钮改为普通按钮，因为普通按钮不会提交 -->          <p>账号：<input type="text" name="name" id="username"></p>          <p>密码：<input type="password" name="password" id="pwd"></p>          <p>              <input type="button" value="确定" id="btnLogin">          </p>      </form>      <div id="result"></div>      <script>          let objLogin = document.getElementById("btnLogin")          let result = document.getElementById("result")          objLogin.onclick = () => {              // alert("login button clicked...")              // result.innerHTML = "btn clicked..."              //获取表单数据              let username = document.getElementById("username").value              let pwd = document.getElementById("pwd").value              // console.log(username, pwd);              //封装数据              let params = { username, pwd }              //ajax处理(4个步骤)              //1.创建ajax（XMLHttpRequest）对象              let xhr = new XMLHttpRequest()              //2.设置访问路径和请求方法              xhr.open('POST', '/login')              //3.监听状态改变的数据处理函数              xhr.onreadystatechange = () => {                  if (xhr.readyState === 4 && xhr.status === 200) {                      result.innerHTML = xhr.responseText                  }              }              //4.发送请求              xhr.send(JSON.stringify(params))              //超时处理              let timer = setTimeout(() => {                  xhr.abort()              }, 2000) //如果超过8秒就中断请求          }      </script>  </body>  </html> |

## 然后可以在服务端模拟延时

|  |
| --- |
| let http = require('http')  let fs = require('fs')  let path = require('path')  let PORT = 8081  //定义两个数据用来模拟数据库的记录，如果post传递过来的数据和这两个数据一样，就算是登录成功  const USER = "admin"  const PASS = '12345'  function doResponse(response,dir,fileName){      let filePath = path.join(\_\_dirname,'assets',dir,fileName)      let content = fs.readFileSync(filePath)      response.end(content)  }  let server = http.createServer((request,response)=>{      //判断浏览器需要哪个资源      let reqUrl = request.url      if(reqUrl==='/' || reqUrl==='/index.html'){          //读取页面内容,返回信息          doResponse(response,"html","index.html")      } else if(reqUrl==='/detail.html'){          //读取页面内容,返回信息          doResponse(response,"html","detail.html")      } else if(reqUrl==='/login.html'){ //处理/login.html         //读取页面内容,返回信息          doResponse(response,"html",'login.html')       }       // else if(reqUrl==='/get\_data'){ //处理/get\_data      //     let data = {      //         "name":"kenny",      //         "position":"boss"      //     }      //     response.end(JSON.stringify(data))      // }      else if(reqUrl.startsWith('/get\_data')){ //处理/get\_data+Math.random()              let data = {                  "name":"kenny",                  "position":"boss"              }              response.end(JSON.stringify(data))          }        else if(reqUrl==='/login'){ //处理/login          // console.log(request);          let data = ''            //模拟网络延时          setTimeout(()=>{              request.on('data',(chunk)=>{ //nodejs请求对象处理post添加的数据的方法，需要配合下面的end数件               data += chunk            })          },5000)          request.on('end',()=>{              // console.log(data);              let { username,pwd } = JSON.parse(data)              // console.log(username,pwd);               //利用上面定义的两个常量来模拟用户验证               if(username === USER && pwd === PASS){                  response.end(`登录成功：欢迎您:${username}`)              } else {                  response.end("登录失败，用户名或密码错误")              }              })        }      else if(reqUrl.endsWith('.js')){          //读取页面内容,返回信息          doResponse(response,'js','index.js')      } else if(reqUrl.endsWith('.css')){          //读取页面内容,返回信息         doResponse(response,'css','index.css')      } else {           response.setHeader('Content-type','text/html;charset=utf-8')           response.end('Page Not Found')      }  })  server.listen(PORT,()=>{      console.log(`server is ready at : http://localhost:${PORT}/`);  }) |

### 效果：发送请求2秒钟后就会中断请求

|  |
| --- |
|  |

### 当然这只是学习用途才这么干，实际开发不会这么干

## 4>ajax方法抽取

|  |
| --- |
|  |

# 小结：这里主要是练习ajax技术发送post请求的方法，服务器比较简陋，我们很快就可以学习完整的服务器技术

# 注意：上面的ajax代码知识学习用的，帮助我们理解ajax的执行流程，实际开发中不会这么写，我们有更好的选择

# 2.jQuery的ajax请求

## jquery中ajax的使用

|  |
| --- |
|  |

## 1)jquery的ajax方法发送get请求或者jquery.get方法

## 为了方便学习我把httpserver项目复制一份，改名httpserver-jq

|  |
| --- |
|  |

## 我们做和原生ajax实现一样的效果

## 我们尝试引入本地jquery文件可是老是出问题，原因是我们没有在服务器里面配置jquery的路径：解决办法有2种，1是使用cdn，第2种就是配置jquery路径。我们这里使用第二种。我们把原来的myserverjq.js复制一份，然后一个改为1-myserverjq.js，另外一个改名2-myserverjq.js我们使用2-myserverjq.js，在里面添加/jquery的路径处理函数

|  |
| --- |
| let http = require('http')  let fs = require('fs')  let path = require('path')  let PORT = 8082  //定义两个数据用来模拟数据库的记录，如果post传递过来的数据和这两个数据一样，就算是登录成功  const USER = "admin"  const PASS = '12345'  function doResponse(response,dir,fileName){      let filePath = path.join(\_\_dirname,'assets',dir,fileName)      let content = fs.readFileSync(filePath)      response.end(content)  }  let server = http.createServer((request,response)=>{      //判断浏览器需要哪个资源      let reqUrl = request.url      if(reqUrl==='/' || reqUrl==='/index.html'){          //读取页面内容,返回信息          doResponse(response,"html","index.html")      } else if(reqUrl==='/detail.html'){          //读取页面内容,返回信息          doResponse(response,"html","detail.html")      } else if(reqUrl==='/login.html'){ //处理/login.html         //读取页面内容,返回信息          doResponse(response,"html","login.html")       }       else if(reqUrl==='/jquery'){ //添加对jquery的路径处理          //读取页面内容,返回信息            doResponse(response,"js",'jquery3.1.0.min.js')      }       // else if(reqUrl==='/get\_data'){ //处理/get\_data      //     let data = {      //         "name":"kenny",      //         "position":"boss"      //     }      //     response.end(JSON.stringify(data))      // }      else if(reqUrl.startsWith('/get\_data')){ //处理/get\_data+Math.random()              let data = {                  "name":"kenny",                  "position":"boss",                  "tool":"jquery"              }              console.log(request.url);              response.end(JSON.stringify(data))          }      else if(reqUrl.startsWith('/jq\_get\_data')){ //处理jquery发送的ajax的get请求              let data = {                  "name":"kenny",                  "position":"boss",                  "tool":"jquery"              }              console.log(request.url);              response.end(JSON.stringify(data))          }        else if(reqUrl==='/login'){ //处理/login          // console.log(request);          let data = ''          request.on('data',(chunk)=>{ //nodejs请求对象处理post添加的数据的方法，需要配合下面的end数件              data += chunk          })          request.on('end',()=>{              // console.log(data);              let { username,pwd } = JSON.parse(data)              console.log(username,pwd);              //利用上面定义的两个常量来模拟用户验证              if(username === USER && pwd === PASS){                  response.end(`登录成功：欢迎您:${username}`)              } else {                  response.end("登录失败，用户名或密码错误")              }            })        } else if(reqUrl.endsWith('.css')){          //读取页面内容,返回信息         doResponse(response,'css','index.css')      } else {           response.setHeader('Content-type','text/html;charset=utf-8')           response.end('Page Not Found')      }  })  server.listen(PORT,()=>{      console.log(`server is ready at : http://localhost:${PORT}/`);  }) |

## 然后可以在index.html中这么使用

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <meta name="viewport" content="width=device-width, initial-scale=1.0">      <title>首页</title>      <link rel="stylesheet" href="../css/index.css">  </head>  <body>      <a href="/login.html">登录</a>      <h1>欢迎来到首页</h1>      <a href="/detail.html">          详情页      </a>      <hr>      <div id="box">          <button id="mybtn">获取数据</button>          <div id="mycontent"></div>      </div>      <!-- <script src="https://cdn.bootcdn.net/ajax/libs/jquery/3.7.1/jquery.js"></script> -->      <script src="/jquery"></script>      <script>          let myBtn = $("#mybtn")          let content = $("#mycontent")          myBtn.click(() => {              // alert("index.html")              $.ajax({                  url: "/jq\_get\_data",                  success: (data) => {                      content.text(data)                      console.log("data:", data);                  }              })          })      </script>  </body>  </html> |

## 修改一下index.css

|  |
| --- |
| \*{      padding: 0;      margin: 0;  }  hr {      border: 1px solid deeppink;      margin: 5px;  }  #box{      margin: 10px;      width: 750px;      height: 300px;      border: 1px solid green;      padding: 10px;      font-size: xx-large;  }  ul{      list-style: none;  } |

## 效果

|  |
| --- |
|  |

### 可见用jquery的ajax方法方式get请求比较简单

## 那么，如果需要传递参数呢？get方法可以直接拼接在url后面

### index.html

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <meta name="viewport" content="width=device-width, initial-scale=1.0">      <title>首页</title>      <link rel="stylesheet" href="../css/index.css">  </head>  <body>      <a href="/login.html">登录</a>      <h1>欢迎来到首页</h1>      <a href="/detail.html">          详情页      </a>      <hr>      <div id="box">          <button id="mybtn">获取数据</button>          <div id="mycontent"></div>      </div>      <!-- <script src="https://cdn.bootcdn.net/ajax/libs/jquery/3.7.1/jquery.js"></script> -->      <script src="/jquery"></script>      <script>          let myBtn = $("#mybtn")          let content = $("#mycontent")          myBtn.click(() => {              // alert("index.html")              $.ajax({                  url: "/jq\_get\_data?name=Mary&email=Mary123@gmail.com",                  success: (data) => {                      content.text(data)                      console.log("data:", data);                  }              })          })      </script>  </body>  </html> |

## 还可以把查询参数放到data属性里面

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <meta name="viewport" content="width=device-width, initial-scale=1.0">      <title>首页</title>      <link rel="stylesheet" href="../css/index.css">  </head>  <body>      <a href="/login.html">登录</a>      <h1>欢迎来到首页</h1>      <a href="/detail.html">          详情页      </a>      <hr>      <div id="box">          <button id="mybtn">获取数据</button>          <div id="mycontent"></div>      </div>      <!-- <script src="https://cdn.bootcdn.net/ajax/libs/jquery/3.7.1/jquery.js"></script> -->      <script src="/jquery"></script>      <script>          let myBtn = $("#mybtn")          let content = $("#mycontent")          myBtn.click(() => {              // alert("index.html")              $.ajax({                  url: "/jq\_get\_data",                  data: {                      name: "Benny",                      email: "Benny123@gmail.com"                  },                  success: (data) => {                      content.text(data)                      console.log("data:", data);                  }              })          })      </script>  </body>  </html> |

### 效果是一样的

## 其实jQuery也提供了ajax的get方法，效果是一样的，写法也是类似的

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <meta name="viewport" content="width=device-width, initial-scale=1.0">      <title>首页</title>      <link rel="stylesheet" href="../css/index.css">  </head>  <body>      <a href="/login.html">登录</a>      <h1>欢迎来到首页</h1>      <a href="/detail.html">          详情页      </a>      <hr>      <div id="box">          <button id="mybtn">获取数据</button>          <div id="mycontent"></div>      </div>      <!-- <script src="https://cdn.bootcdn.net/ajax/libs/jquery/3.7.1/jquery.js"></script> -->      <script src="/jquery"></script>      <script>          let myBtn = $("#mybtn")          let content = $("#mycontent")          myBtn.click(() => {              // alert("index.html")              // $.ajax({              //     url: "/get\_data",              //     data: {              //         name: "Benny",              //         email: "Benny123@gmail.com"              //     },              //     success: (data) => {              //         content.text(data)              //         console.log("data:", data);              //     }              // })              $.get({                  url: "/jq\_get\_data",                  data: {                      name: "Randy lou",                      email: "Randy123@gmail.com"                  },                  success: (data) => {                      content.text(data)                      console.log("data:", data);                  }              })          })      </script>  </body>  </html> |

## 还可以这么写

|  |
| --- |
|  |

## 2)jquery的ajax方法发送post请求或者jquery.post方法

## 1》我们在2-myserverjq.js文件中添加一个/jq\_login接口，专门用来处理jquery发送的ajax post请求

|  |
| --- |
| let http = require('http')  let fs = require('fs')  let path = require('path')  let PORT = 8082  //定义两个数据用来模拟数据库的记录，如果post传递过来的数据和这两个数据一样，就算是登录成功  const USER = "admin"  const PASS = '12345'  function doResponse(response,dir,fileName){      let filePath = path.join(\_\_dirname,'assets',dir,fileName)      let content = fs.readFileSync(filePath)      response.end(content)  }  let server = http.createServer((request,response)=>{      //判断浏览器需要哪个资源      let reqUrl = request.url      if(reqUrl==='/' || reqUrl==='/index.html'){          //读取页面内容,返回信息          doResponse(response,"html","index.html")      } else if(reqUrl==='/detail.html'){          //读取页面内容,返回信息          doResponse(response,"html","detail.html")      } else if(reqUrl==='/login.html'){ //处理/login.html         //读取页面内容,返回信息          doResponse(response,"html","login.html")       }       else if(reqUrl==='/jquery'){ //添加对jquery的路径处理          //读取页面内容,返回信息            doResponse(response,"js",'jquery3.1.0.min.js')      }       // else if(reqUrl==='/get\_data'){ //处理/get\_data      //     let data = {      //         "name":"kenny",      //         "position":"boss"      //     }      //     response.end(JSON.stringify(data))      // }      else if(reqUrl.startsWith('/get\_data')){ //处理/get\_data+Math.random()              let data = {                  "name":"kenny",                  "position":"boss",                  "tool":"jquery"              }              console.log(request.url);              response.end(JSON.stringify(data))          }      else if(reqUrl.startsWith('/jq\_get\_data')){ //处理jquery发送的ajax的get请求              let data = {                  "name":"kenny",                  "position":"boss",                  "tool":"jquery"              }              console.log(request.url);              response.end(JSON.stringify(data))          }        else if(reqUrl==='/login'){ //处理/login          // console.log(request);          let data = ''          request.on('data',(chunk)=>{ //nodejs请求对象处理post添加的数据的方法，需要配合下面的end数件              data += chunk          })          request.on('end',()=>{              // console.log(data);              let { username,pwd } = JSON.parse(data)              console.log(username,pwd);              //利用上面定义的两个常量来模拟用户验证              if(username === USER && pwd === PASS){                  response.end(`登录成功：欢迎您:${username}`)              } else {                  response.end("登录失败，用户名或密码错误")              }            })        }   else if(reqUrl==='/jq\_login'){ //处理/jq\_login,程序有一个小bug，用户名不能有空格          // console.log(request.url);          let data = ''          request.on('data',(chunk)=>{ //nodejs请求对象处理post添加的数据的方法，需要配合下面的end数件              data += chunk          })          request.on('end',()=>{          //    console.log(data);              let { username,pwd } = JSON.parse(data)              console.log(username,pwd);              //利用上面定义的两个常量来模拟用户验证              if(username === USER && pwd === PASS){                  response.end(`登录成功：欢迎您:${username}`)              } else {                  response.end("登录失败，用户名或密码错误")              }            })        }      else if(reqUrl.endsWith('.css')){          //读取页面内容,返回信息         doResponse(response,'css','index.css')      } else {           response.setHeader('Content-type','text/html;charset=utf-8')           response.end('Page Not Found')      }  })  server.listen(PORT,()=>{      console.log(`server is ready at : http://localhost:${PORT}/`);  }) |

## 2）.然后我们在login.html中把原生ajax的post请求代码注释掉，添加jquery的ajax post请求的代码

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>      <meta charset='UTF-8' />      <title>用户登录</title>  </head>  <body>      <form>          <!--使用ajax技术提交，这里不用写action和method属性 需要就submit按钮改为普通按钮，因为普通按钮不会提交 -->          <p>账号：<input type="text" name="name" id="username"></p>          <p>密码：<input type="password" name="password" id="pwd"></p>          <p>              <input type="button" value="确定" id="btnLogin">          </p>      </form>      <div id="result"></div>      <!-- <script src="https://cdn.bootcdn.net/ajax/libs/jquery/3.7.1/jquery.js"></script> -->      <script src="/jquery"></script>      <script>          //query方式          let objLogin = $("#btnLogin")          let result = $("#result")          objLogin.click(() => {              let params = {                  username: $("#username").val(),                  pwd: $("#pwd").val()              }              console.log(params);              $.ajax({                  url: "/jq\_login",                  type: "POST",                  data: JSON.stringify(params),                  success: (data) => {                      result.html(data)                  }              })          })            //原生js的ajax方法          // let objLogin2 = document.getElementById("btnLogin")          // let result2 = document.getElementById("result")          // let params = { username, pwd }          // objLogin2.onclick = () => {          //     // alert("login button clicked...")          //     // result.innerHTML = "btn clicked..."          //     //获取表单数据          //     let username = document.getElementById("username").value          //     let pwd = document.getElementById("pwd").value          //     // console.log(username, pwd);          //     //封装数据          //     let params = { username, pwd }          //     //ajax处理(4个步骤)          //     //1.创建ajax（XMLHttpRequest）对象          //     let xhr = new XMLHttpRequest()          //     //2.设置访问路径和请求方法          //     xhr.open('POST', '/login')          //     //3.监听状态改变的数据处理函数          //     xhr.onreadystatechange = () => {          //         if (xhr.readyState === 4 && xhr.status === 200) {          //             result2.innerHTML = xhr.responseText          //         }          //     }          //     //4.发送请求          //     xhr.send(JSON.stringify(params))          //     //超时处理          //     let timer = setTimeout(() => {          //         xhr.abort()          //     }, 2000) //如果超过8秒就中断请求          // }      </script>  </body>  </html> |

### 效果是一样的

|  |
| --- |
|  |

## 然后我们使用jquery的post方法

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>      <meta charset='UTF-8' />      <title>用户登录</title>  </head>  <body>      <form>          <!--使用ajax技术提交，这里不用写action和method属性 需要就submit按钮改为普通按钮，因为普通按钮不会提交 -->          <p>账号：<input type="text" name="name" id="username"></p>          <p>密码：<input type="password" name="password" id="pwd"></p>          <p>              <input type="button" value="确定" id="btnLogin">          </p>      </form>      <div id="result"></div>      <!-- <script src="https://cdn.bootcdn.net/ajax/libs/jquery/3.7.1/jquery.js"></script> -->      <script src="/jquery"></script>      <script>          //query方式          let objLogin = $("#btnLogin")          let result = $("#result")          objLogin.click(() => {              let params = {                  username: $("#username").val(),                  pwd: $("#pwd").val()              }              console.log(params);              // $.ajax({              //     url: "/jq\_login",              //     type: "POST",              //     data: JSON.stringify(params),              //     success: (data) => {              //         result.html(data)              //     }              // })              $.post({                  url: "/jq\_login",                  type: "POST",                  data: JSON.stringify(params),                  success: (data) => {                      result.html(data)                  }              })          })            //原生js的ajax方法          // let objLogin2 = document.getElementById("btnLogin")          // let result2 = document.getElementById("result")          // let params = { username, pwd }          // objLogin2.onclick = () => {          //     // alert("login button clicked...")          //     // result.innerHTML = "btn clicked..."          //     //获取表单数据          //     let username = document.getElementById("username").value          //     let pwd = document.getElementById("pwd").value          //     // console.log(username, pwd);          //     //封装数据          //     let params = { username, pwd }          //     //ajax处理(4个步骤)          //     //1.创建ajax（XMLHttpRequest）对象          //     let xhr = new XMLHttpRequest()          //     //2.设置访问路径和请求方法          //     xhr.open('POST', '/login')          //     //3.监听状态改变的数据处理函数          //     xhr.onreadystatechange = () => {          //         if (xhr.readyState === 4 && xhr.status === 200) {          //             result2.innerHTML = xhr.responseText          //         }          //     }          //     //4.发送请求          //     xhr.send(JSON.stringify(params))          //     //超时处理          //     let timer = setTimeout(() => {          //         xhr.abort()          //     }, 2000) //如果超过8秒就中断请求          // }      </script>  </body>  </html> |
|  |

## jquery的post方法可以简写

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>      <meta charset='UTF-8' />      <title>用户登录</title>  </head>  <body>      <form>          <!--使用ajax技术提交，这里不用写action和method属性 需要就submit按钮改为普通按钮，因为普通按钮不会提交 -->          <p>账号：<input type="text" name="name" id="username"></p>          <p>密码：<input type="password" name="password" id="pwd"></p>          <p>              <input type="button" value="确定" id="btnLogin">          </p>      </form>      <div id="result"></div>      <!-- <script src="https://cdn.bootcdn.net/ajax/libs/jquery/3.7.1/jquery.js"></script> -->      <script src="/jquery"></script>      <script>          //query方式          let objLogin = $("#btnLogin")          let result = $("#result")          objLogin.click(() => {              let params = {                  username: $("#username").val(),                  pwd: $("#pwd").val()              }              console.log(params);              // $.ajax({              //     url: "/jq\_login",              //     type: "POST",              //     data: JSON.stringify(params),              //     success: (data) => {              //         result.html(data)              //     }              // })              // $.post({              //     url: "/jq\_login",              //     type: "POST",              //     data: JSON.stringify(params),              //     success: (data) => {              //         result.html(data)              //     }              // })              $.post("/jq\_login",                  JSON.stringify(params),                  (data) => {                      result.html(data)                  }              )          })            //原生js的ajax方法          // let objLogin2 = document.getElementById("btnLogin")          // let result2 = document.getElementById("result")          // let params = { username, pwd }          // objLogin2.onclick = () => {          //     // alert("login button clicked...")          //     // result.innerHTML = "btn clicked..."          //     //获取表单数据          //     let username = document.getElementById("username").value          //     let pwd = document.getElementById("pwd").value          //     // console.log(username, pwd);          //     //封装数据          //     let params = { username, pwd }          //     //ajax处理(4个步骤)          //     //1.创建ajax（XMLHttpRequest）对象          //     let xhr = new XMLHttpRequest()          //     //2.设置访问路径和请求方法          //     xhr.open('POST', '/login')          //     //3.监听状态改变的数据处理函数          //     xhr.onreadystatechange = () => {          //         if (xhr.readyState === 4 && xhr.status === 200) {          //             result2.innerHTML = xhr.responseText          //         }          //     }          //     //4.发送请求          //     xhr.send(JSON.stringify(params))          //     //超时处理          //     let timer = setTimeout(() => {          //         xhr.abort()          //     }, 2000) //如果超过8秒就中断请求          // }      </script>  </body>  </html> |

# 3.axios，自己学习

## 注意：1.ajax技术不利于seo 2.ajax技术必须在网络协议的情况下使用也就是客户端向服务器发送请求的发生，直接将文件拖进浏览器是没有效果的。