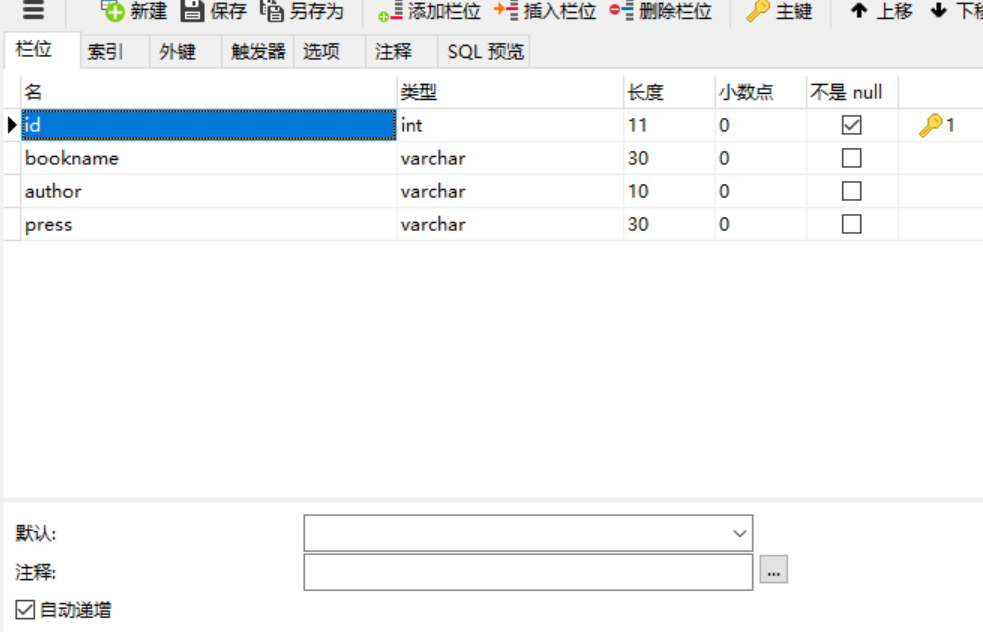
**实验七 node.js平台Web数据库开发**

实验内容：

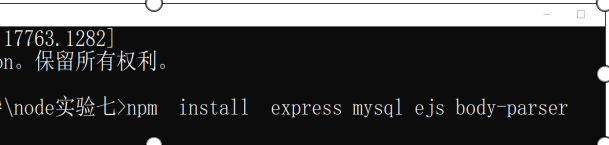
1. 实验准备

1）建一个MySQL,数据库，库名：library，库里建一个表books,表结构如下图，并输入部分数据（如已有数据库则不用建）





2）在上次的实验基础上进一步操作，或按下面步骤重新搭建

a. 在磁盘中建立一个文件夹如：node实验七，进入命令行窗口下载MySQL、express、body-parser、ejs等第三方模块，如图

b. 创建web服务器，响应URL为“localhost:3000/listallbooks”的请求，将查询结果渲染到浏览器中输出

①在实验文件夹下创建一个 booklist.html文件，，代码如下，也可按书上P156所示输入

**<body>**

**<h1>图书列表</h1>**

**<table border=1>**

**<tr>**

**<th>ID</th>**

**<th>图书名称</th>**

**<th>作者</th>**

**<th>出版社</th>**

**</tr>**

**<% mydata.forEach(function(item,index) {  %>**

**<tr>**

**<td><%=item.id%></td>**

**<td><%=item.bookname%></td>**

**<td><%=item.author%></td>**

**<td> <%=item.press%></td>**

**</tr>**

**<%  })  %>**

**</table>**

**</body>**

②创建web服务器，响应响应URL为“localhost:3000/”的请求，将查询结果渲染到浏览器中输出

**var  ejs=require("ejs")**

**var  express=require("express")**

**var  fs=require("fs")**

**var  app=express()**

**var  mysql=require("mysql")**

**var  client=mysql.createConnection({**

**host:"localhost",**

**port:"3306",**

**user:"root",**

**password:"123456",**

**database:"library"**

**})**

**client.connect(function(err){**

**if(err){**

**console.log("连接错误")**

**return**

**}**

**console.log("MySQL数据库连接成功")**

**})**

**app.listen(3000,function(){**

**console.log("server started")**

**})**

**app.get("/listallbooks",function(req,res){**

**fs.readFile("./booklist.html","utf8",**

**function(err,data){**

**client.query("select \* from  books",function(err,results){**

**res.send(ejs.render(data,{mydata:results}))**

**})**

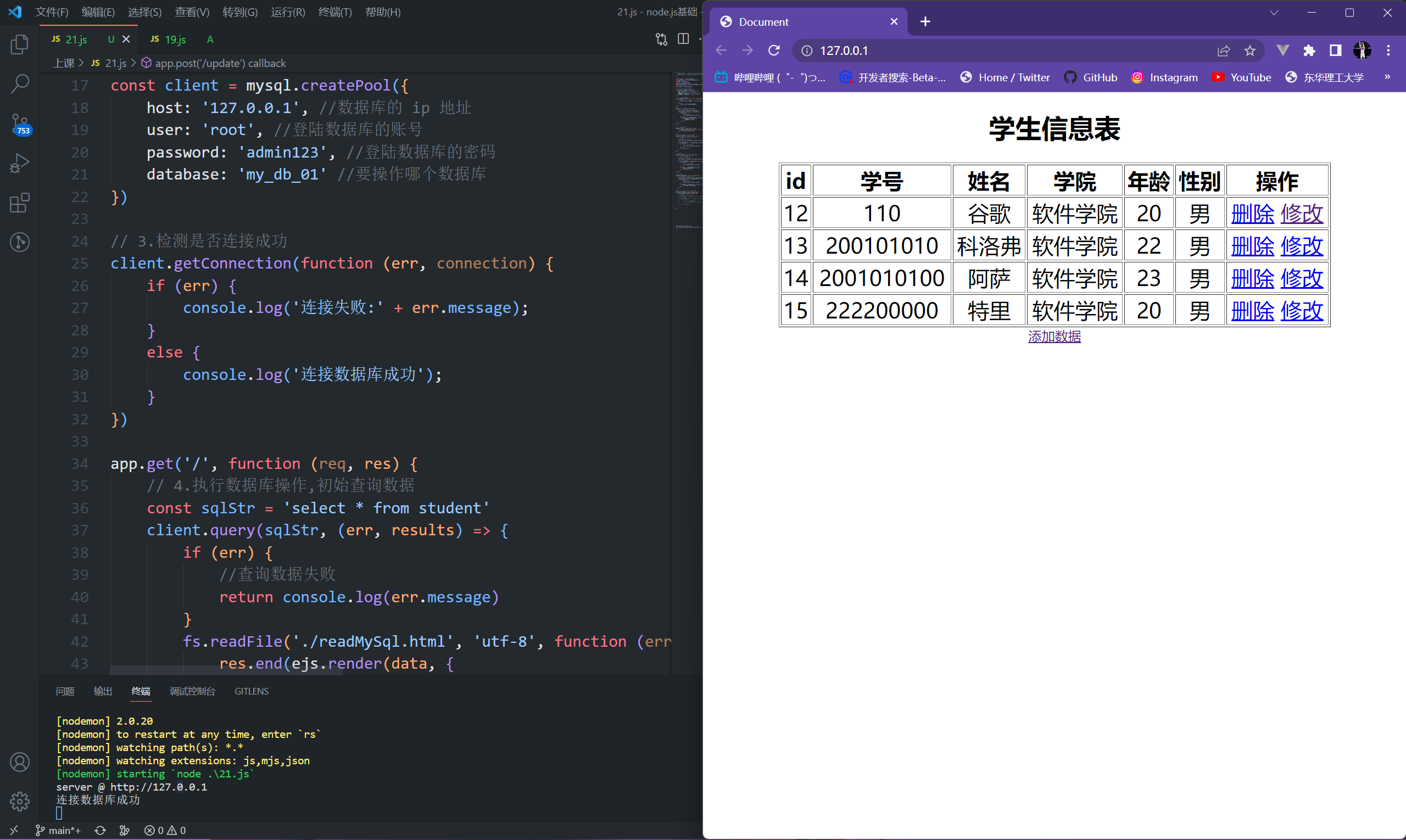
**})**

**})**

在命令行窗口执行启动服务器

然后在浏览器地址栏分别输入以下地址后回车，在浏览器窗口查看结果：

[**http://127.0.0.1:3000/**](http://127.0.0.1:3000/)



1. Node.js对MySQL数据库增加记录（增加红色部分）
2. 修改booklist.html 在页面上提供增加、删除、修改等操作入口

<body>

<h1>图书列表</h1>

<a href="/bookadd">增加图书</a>

<table border=1>

<tr>

<th>ID</th>

<th>图书名称</th>

<th>作者</th>

<th>出版社</th>

<th>操作</th>

</tr>

<% mydata.forEach(function(item,index) { %>

<tr>

<td><%=item.id%></td>

<td><%=item.bookname%></td>

<td><%=item.author%></td>

<td> <%=item.press%></td>

<td>

<a href="/delete/<%=item.id%>">删除</a>

<a href="/edit?id="+<%=item.id%>>修改</a>

</td>

</tr>

<% }) %>

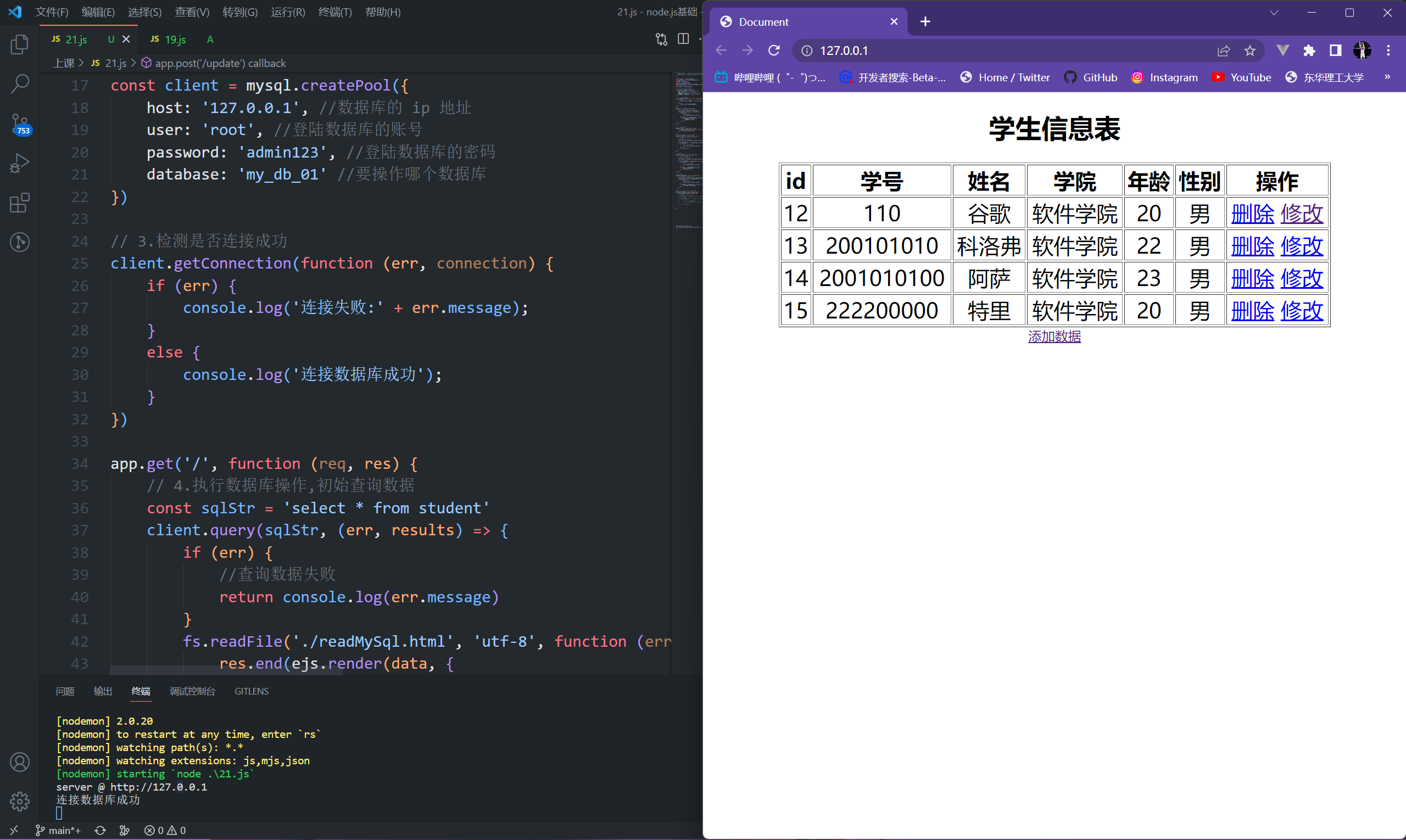
</table>

</body>

在命令行窗口执行启动服务器

然后在浏览器地址栏分别输入以下地址后回车，在浏览器窗口查看结果：

[**http://127.0.0.1:3000/**](http://127.0.0.1:3000/)



1. node.js对数据库增加记录
2. 在实验目录下创建一个bookadd.html文件，代码如下

<body>

<h1>添加图书</h1>

<form action="/bookadd" method="post">

请输入书名: <input type="text" name="bookname"><br>

请输入作者：<input type="text" name="author"><br>

请输入出版社: <input type="text" name="press"><br>

<input type="submit" value="添加">

</form>

</body>

1. 在服务器程序中增加如下代码

app.get("/bookadd",function(req,res){

// console.log("bookadd get ")

fs.readFile(

"./bookadd.html",

"utf8",

function(err,data){

res.send(data)

})

})

app.post("/bookadd",function(req,res){

console.log(req.body)

client.query(

"insert into books values(null,?,?,?)",

[req.body.bookname,req.body.author,req.body.press],

function(err,results){

if(err) return console.log("插入错误")

res.redirect("/listallbooks")

}

)

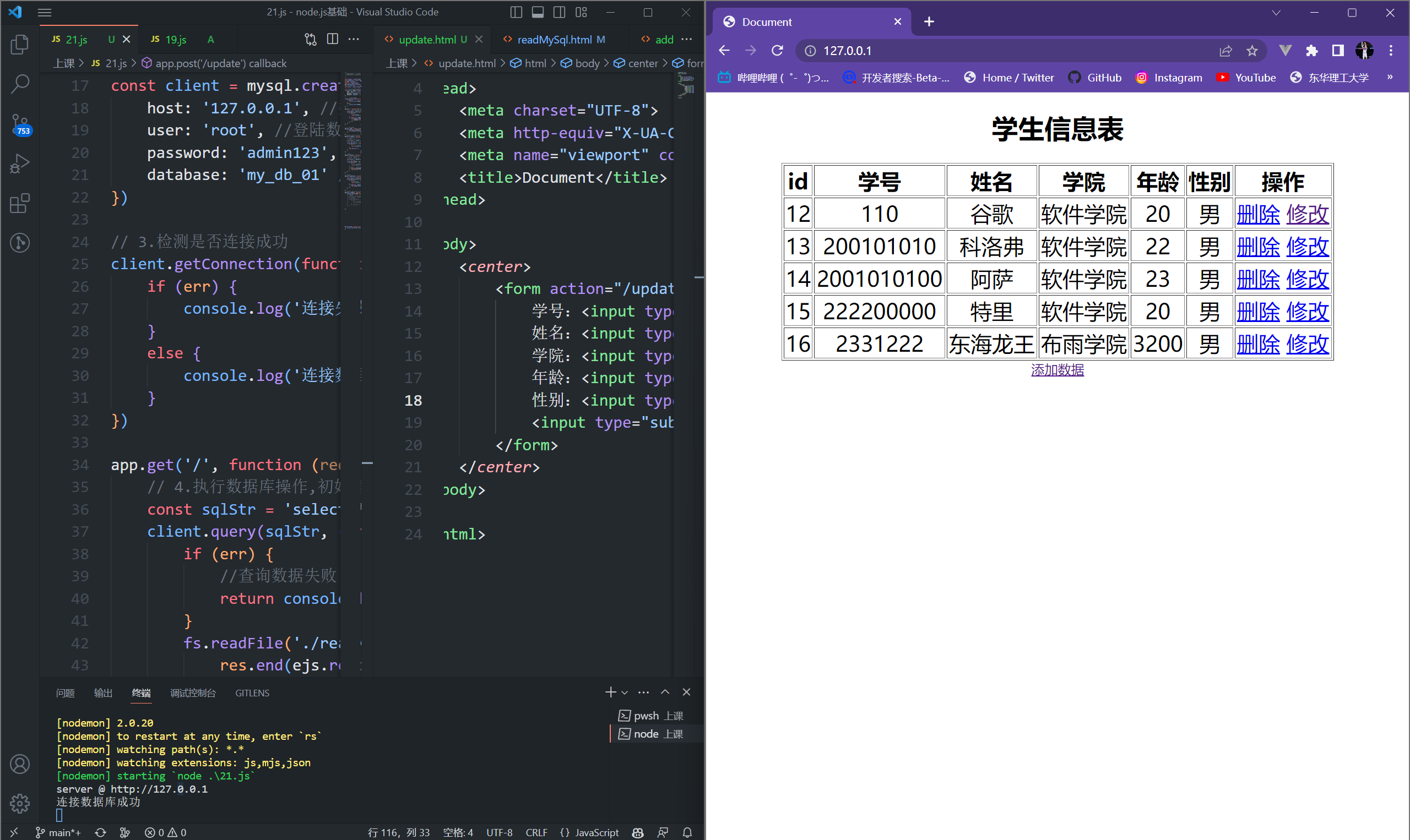
})

在命令行窗口执行启动服务器

然后在浏览器地址栏分别输入以下地址后回车，在浏览器窗口查看结果：

[**http://127.0.0.1:3000/**](http://127.0.0.1:3000/)

点击增加图书超链接，看是否能增加，



1. Node对MySQL数据库的记录进行删除操作

在服务器程序中增加如下代码

app.get("/delete/:myid",function(req,res){

console.log(req.params)

client.query(

"delete from books where id=?",

req.params.myid,

function(){

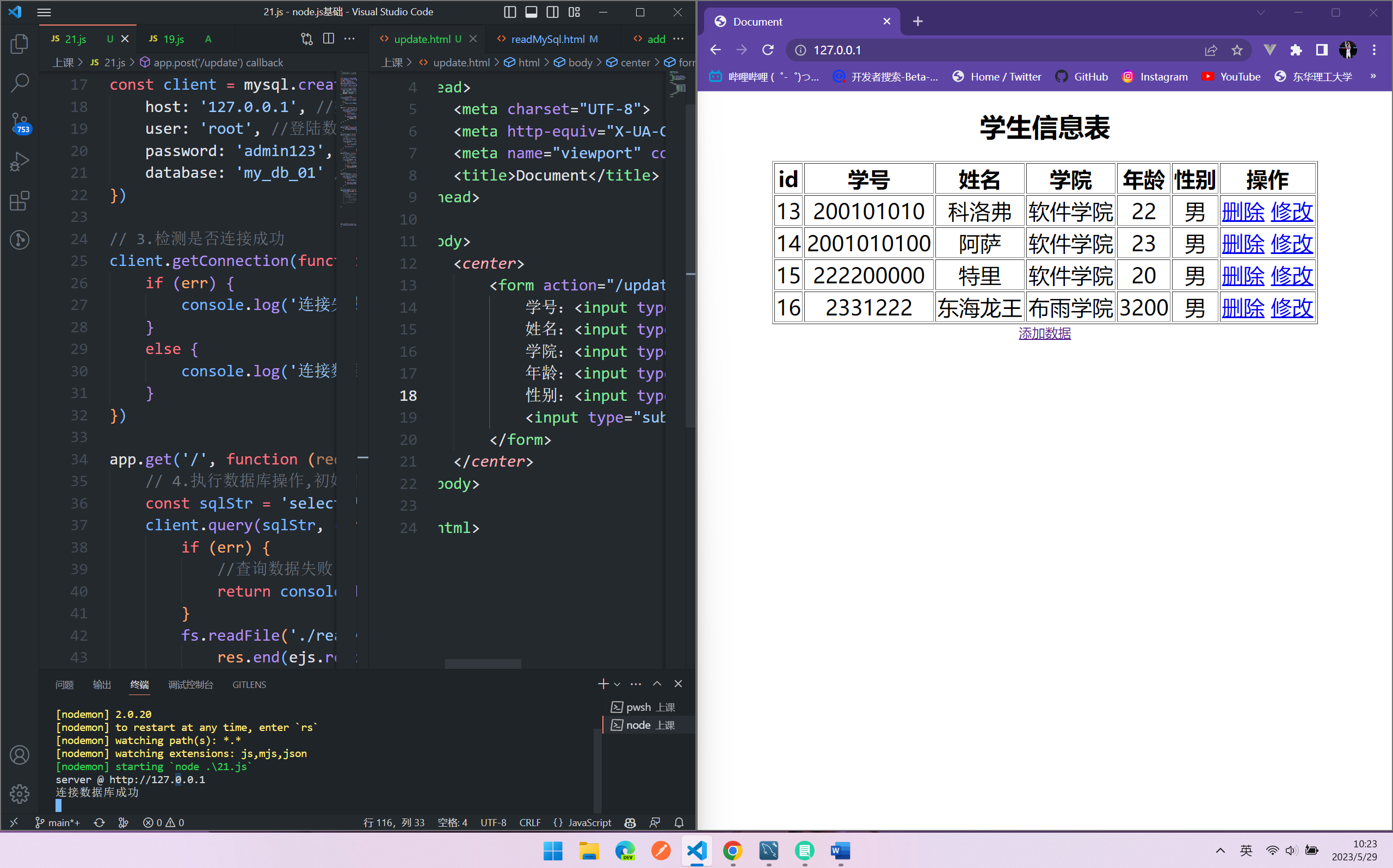
res.redirect("/")

}

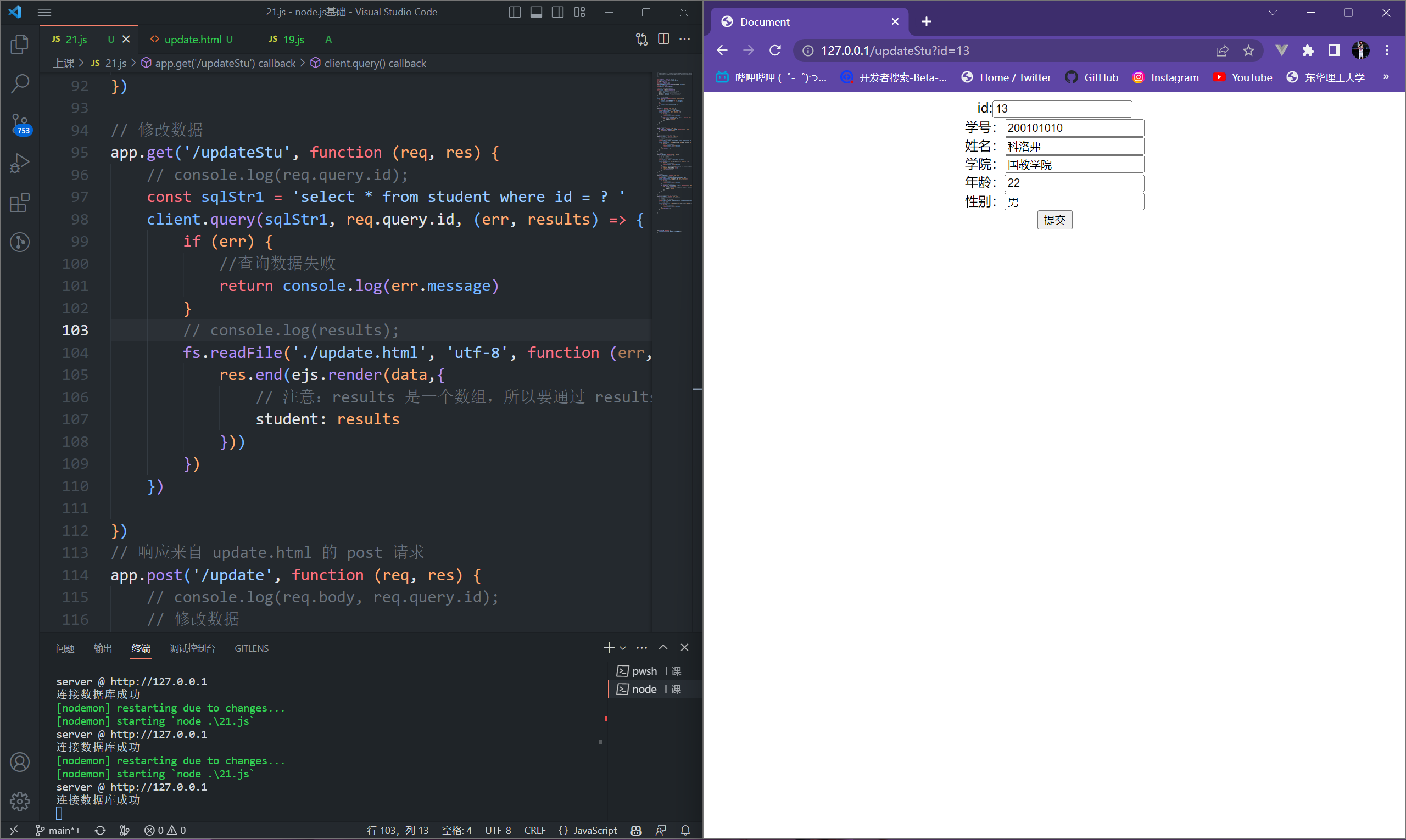
)

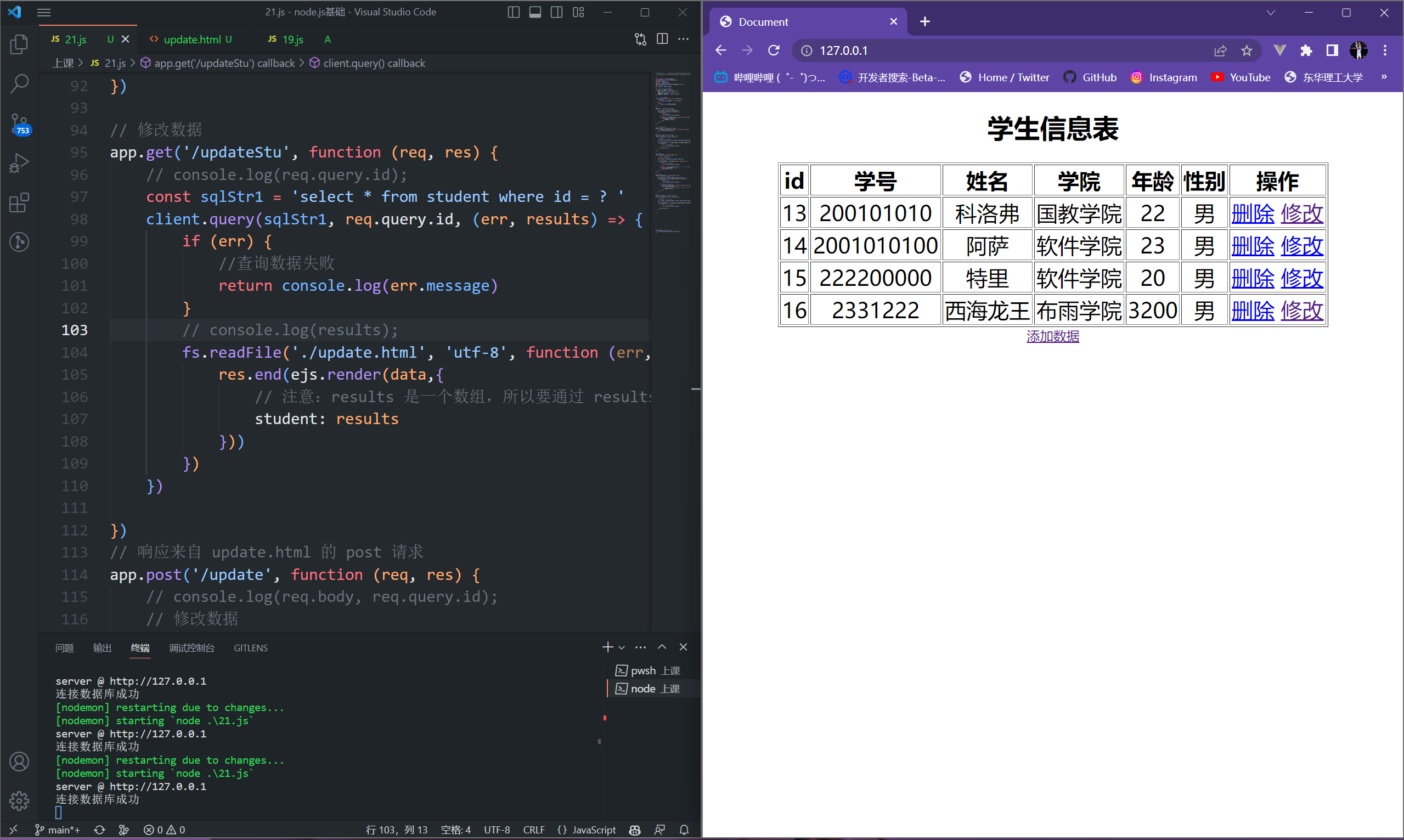
})

运行程序。看是否能完成删除操作

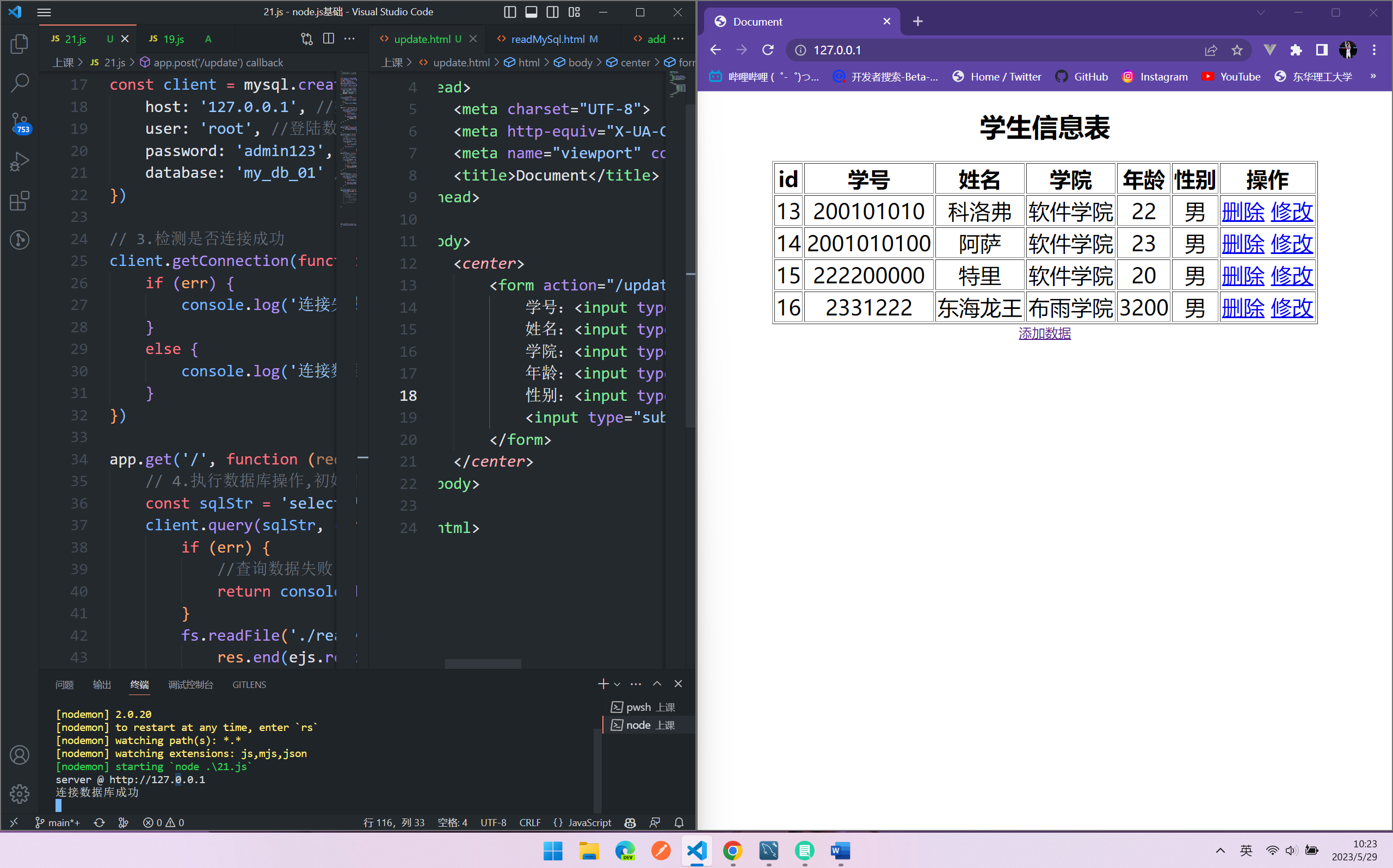


1. 修改操作根据时间参考教材P160代码完成。





1. 在library数据库中建一个student表，表字段有学号、姓名、所在学院，年龄、性别等字段。在表中输入若干数据，编写相应代码，将学生表中性别为男的学生信息显示在浏览器中，能增加学生信息，删除学生信息



1. 收获与心得

本次实验我们通过使用node.js来对数据库进行简单的增删改查，并通过ejs模块将其显示在页面上，注意，使用表单添加数据时需要用body-parse中间件来获取表单的内容，这次实验很好的将前几章学习的内容结合在了一起，例如使用express模块创建基本的服务器，针对不同的请求方式，不同的请求地址响应不同的内容，使用ejs模块将html内容显示在页面上，使用body-parse中间件获取表单提交的内容，通过my-sql模块对数据库进行基本操作等，让我受益匪浅。