Node.js入门和企业级项目开发

第6天课堂笔记

讲师：邵山欢

日期：2017年11月11日

# 一、NodeJS学习成果自查

**只要会写Ajax的接口，结合Mongoose做一套RESTful API的接口，就能学习React了。**

NodeJS无法挑战老牌3P语言的，asp.net、jsp、php。

上课的时候很多时候在讲面向对象，还有一些经验上的东西。

# 用户头像功能继续完善

当用户上传头像、剪裁之后，立即将头像移动位置，从uploads到www/useravatars/文件夹中。

页面单独的js文件喜欢写在</body>之后，所有的公共的js文件（包括各种lib库）都喜欢写在页面的<head>中。

$(function(){

});

等价于

$(document).ready(function(){

});

表示当页面的骨架结构加载完毕之后，执行回调函数。和window.onload类似，但是不一样。

$(document).ready()要早于window.onload事件。

● $(document).ready() 是骨架结构加载完毕

● window.onload 是所有的css、图片、音乐都加载完毕

# 首页的开发

## 3.1 发帖图片开发

点击一个悬浮框的外部，就关闭这个悬浮框。此时要给document绑定监听。

//点击插入图片按钮的事件

$("#insertpic\_button").click(function(event){

//阻止事件继续传播，这样的话就不会点击a标签的时候，也会触发document的点击事件了

event.stopPropagation();

$("#insertpic\_box").removeClass('fade').addClass('show');

});

//点击插入图片框的外部关闭这个框框

$(document).click(function(event){

//你点击的最内层元素

var target = event.target || event.srcElement;

**if($(target).parents("#insertpic\_box").length <= 0 && !$(target).is("#insertpic\_box")){**

$("#insertpic\_box").removeClass('show').addClass('fade');

}

});

## 3.2 图片的异步上传

HTML没有图片异步上传功能。

**先复习一下同步和异步：**

如果有一个form表单，里面有一些输入框，一个提交按钮，form标签有action、method属性，点击提交按钮页面会跳转，这种表单提交我们称之为“同步提交表单”：

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

<p>

<input type="text" name="yonghuming" />

</p>

<p>

<input type="submit">

</p>

</form>

如果表单没有action、method属性，没有submit按钮，而是普通按钮。点击普通按钮的时候，通过ajax提交，此时我们称之为“异步提交”：

<form>

<p>

<input type="text" name="yonghuming" />

</p>

<p>

<input type="button">

</p>

</form>

$("input[type=button]").click(function(){

$.post("/tijiao" , {

yonghuming : $("input[name=yonghuming]").val();

});

});

如果表单中有file控件（让用户选择一个文件），此时这个表单原则上讲、API层级没有任何办法异步提交！

老老实实这么写：

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

<input type="file" />

<input type="submit">

</form>

这样一来，用户体验极差，用户上传一张图片之后必须死等，不能流畅的操作，不能让上传图片的操作放到后台。

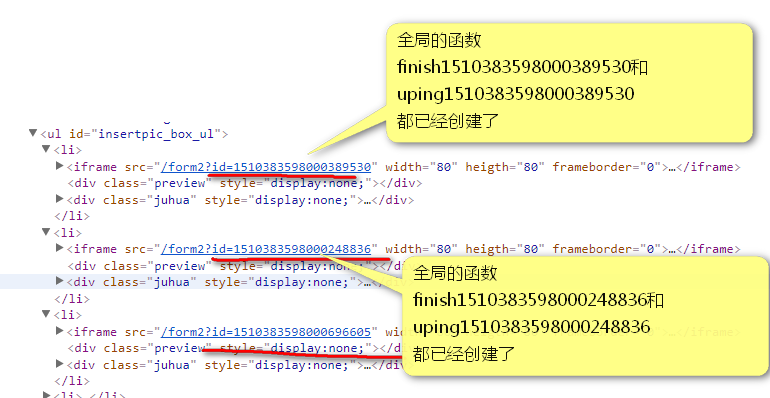
所以中国人发明了一个猫腻写法，解决了这个事情：

* 隐藏<input type="file" />控件
* 写一个外部form表格
* 做一个其他的东西比如+号，点击+号的时候trigger那个file的click事件，会弹出框框
* 监听file控件onchange事件，当onchange事件发生，命令$(form).submit()

理一下逻辑：因为图片上传是不能有异步的，必须用form来模拟，而form提交会跳转页面，所以就要内嵌到iframe里面。此时form上传成功的回调就需要通过form提交到的那个页面/fatupian来写一个<script>window.parent.finish()</scirpt>传给主页面。

|  |  |  |
| --- | --- | --- |
|  |  |  |

麻烦的是，每一个图片上传完毕之后，不能都调用名字叫做finish()的函数，这样就乱套了。此时：我们现在给每一个li这个类，**一上来就创建一个随机数，这个数字就是回调函数的名字**。通过iframe的src引用时候的GET参数传给form2页面，form2页面中提交的网址也不是/fatupian了，而是/fatupian?callback=....



jQueryUI中，sortable表示可以拖拽排序，最后一项不参加排序：

//可以被排序

$("#insertpic\_box\_ul").sortable({

"items" : "li:not(:last)"

});

## 3.3 发问题的实现

所有的问题存储在一个集合中，帖子有哪些属性：

* content内容
* images图片数据
* email发帖人的email
* time发帖时间

我们实现了发帖功能的API接口，这是一个POST请求：

**实话：我们的nodejs课程，你只需要能够流畅写出下面的语句就满分毕业**！

//发帖功能

exports.doSaveQ = function(req,res){

var form = new formidable.IncomingForm();

form.parse(req , function(err , fields , files){

var email = req.session.email;

var content = fields.content;

var images = fields.images;

var time = new Date();

//保存

Q.create({

email : email ,

content : content,

images : images ,

time : time

},function(){

res.json({"result" : 1});

});

});

}

## 3.4 拉取所有帖子

套路：写接口，用Ajax拉取，面向对象上树。

分页的实现：比如每页5条，页码从1开始。第3页就是跳过10条，读5条，此时读取的就是第11~15条。

**第page页，就是跳过(page-1)\*pagesize条，读pagesize条。**

exports.getQ = function(req,res){

var page = url.parse(req.url , true).query.page;

var pagesize = url.parse(req.url , true).query.pagesize;

//得到所有的帖子，按时间倒序

Q.find({}).sort({"time" : -1}).**skip(**(page - 1) \* pagesize**)**.**limit(**pagesize**)**.exec(function(err,results){

res.json({"results" : results});

});

}

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | ... | 19 | 20 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | ... | 19 | 20 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | ... | 19 | 20 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | ... | 20 |
| 1 | ... | 4 | 5 | 6 | ... | 20 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | ... | 5 | 6 | 7 | ... | 20 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | ... | 6 | 7 | 8 | ... | 20 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | ... | 14 | 15 | 16 | ... | 20 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | ... | 15 | 16 | 17 | ... | 20 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | ... | 16 | 17 | 18 | 19 | 20 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | ... | 17 | 18 | 19 | 20 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | ... | 17 | 18 | 19 | 20 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | ... | 17 | 18 | 19 | 20 |