http://www.w3school.com.cn/

这个网站提供大量的网站开发技术知识，包括W3C的标准技术：HTML、CSS、XML 。以及其他的技术，诸如JavaScript、PHP、SQL等等。还提供了还有非常实用的在线测试功能。

# HTML

页面跳转的几种方式：

超链接

<a href="" target="\_blank" ></a>

# CSS

## JQuery 实现 div弹出层示例

<%@ page contentType="text/html; charset=utf-8" language="java" errorPage="" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<title>jquery弹出层实例，jqueryajax.com原创</title>

<script type="text/javascript" src="http://www.jqueryajax.com/wp-content/uploads/2009/03/jquery-132min1.js"></script>

<script type="text/javascript">

$(function(){

$(".but").click(function(){

var str = "这里是信息内容，这里是信息内容！";

$(".form").html(str);

$("#faqbg").css({display:"block",height:$(document).height()});

var yscroll =document.documentElement.scrollTop;

$("#faqdiv").css("top","100px");

$("#faqdiv").css("display","block");

document.documentElement.scrollTop=0;

});

$(".close").click(function(){

$("#faqbg").css("display","none");

$("#faqdiv").css("display","none");

});

})

</script>

<style type="text/css">

/\*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

冷漠夜空 2008

Email:lmyekong@163.com

QQ:135055506

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\*/

body,h2{margin:0 ; padding:0;}

#faqbg{background-color:#666666; position:absolute; z-index:99; left:0; top:0; display:none; width:100%; height:1000px;opacity:0.5;filter: alpha(opacity=50);-moz-opacity: 0.5;}

#faqdiv{position:absolute;width:400px; left:50%; top:50%; margin-left:-200px; height:auto; z-index:100;background-color:#fff; border:1px #8FA4F5 solid; padding:1px;}

#faqdiv h2{ height:25px; font-size:14px; background-color:#8FA4F5; position:relative; padding-left:10px; line-height:25px;}

#faqdiv h2 a{position:absolute; right:5px; font-size:12px; color:#FF0000}

#faqdiv .form{padding:10px;}

</style>

</head>

<body >

<div id="faqbg"></div>

<div id="faqdiv" style="display:none">

<h2>信息窗口<a href="#" class="close">关闭</a></h2>

<div class="form">

这里是提示信息！！

</div>

</div>

<p>&nbsp;</p>

<p align="center">

<input value="弹出" class="but" type="button" />

<input type="button" value="关闭" class="close" />

</p>

</body>

</html>

# JavaScript

## My97DatePicker时间控件详细使用说明

**onpicking事件**

实现日期选择联动：

<label class="date-title">就诊日期：</label>

<input id="d5221" class="Wdate" type="text" onFocus="var d5222=$dp.$('d5222');WdatePicker({onpicked:function(){d5222.focus();},minDate:'%y-%M-%d',maxDate:'#F{$dp.$D(\'d5222\')}'})"/>

至

<input id="d5222" class="Wdate" type="text" onFocus="WdatePicker({onpicked:doDateSearch, minDate:'#F{$dp.$D(\'d5221\')}'})"/>

选中日期后触发事件，调用doDateSearch方法：

<input id="d5222" class="Wdate" type="text" onFocus="WdatePicker({onpicked:doDateSearch, minDate:'#F{$dp.$D(\'d5221\')}'})"/>

## CKeditor所见即所得网页编辑器

CKEditor是FCKEditor的新版，CKEditor更好用，更简洁

### [CKEditor图片上传实现详细步骤(使用Struts 2)](http://blog.csdn.net/mydwr/article/details/8669594)

本人使用的CKEditor版本是3.6.3。CKEditor配置和部署我就不多说。

CKEditor的编辑器工具栏中有一项“图片域”，该工具可以贴上图片地址来在文本编辑器中加入图片，源文件一栏上可以输入图片的网络地址来引用图片，但是不能上传本地图片。

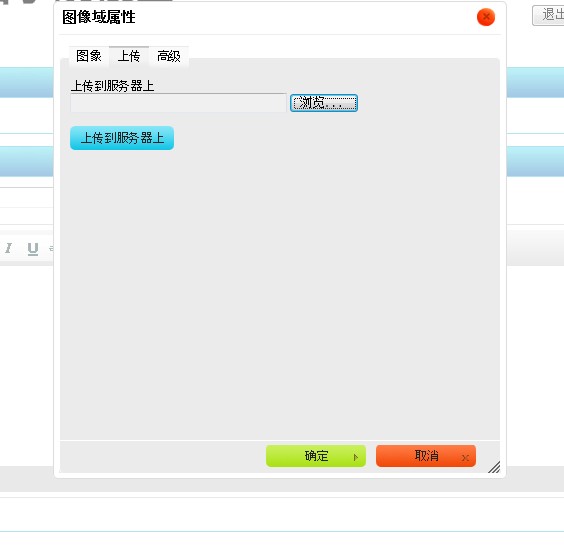
“预览”中有一大堆鸟语，看得很不爽。可以打开ckeditor/plugins/image/dialogs/image.js文件，搜索“b.config.image\_previewText”就能找到这段鸟语了，(b.config.image\_previewText||'')单引号中的内容全删了，注意别删多了。

扫除这个障碍，下面来研究图片上传。

**1.**首先，还是image.js这个文件，搜索“upload”可以找到这一段

id:'Upload',hidden:true

实际上上传功能被隐藏了，把上面的true改成false，再打开编辑器，就能找到上传功能了。



**2.**上面的只是一个上传页面。也就相当于一个HTML的form表单，要配置点击“上传到服务器上”按钮后请求的Action。可以在ckeditor/config.js中配置。

加入：

config.filebrowserUploadUrl="actions/ckeditorUpload";

var pathName = window.document.location.pathname;  
    //获取带"/"的项目名，如：/uimcardprj  
    var projectName = pathName.substring(0, pathName.substr(1).indexOf('/') + 1);  
    config.filebrowserImageUploadUrl = projectName+'/system/upload.do'; //固定路径

"ckeditorUpload"是请求的URL，也就是点击这个按钮就会post到ckeditorUpload地址进行处理，这里指向的是Struts 2的一个Action。当然，也可以用servlet或者ASP、PHP等来处理请求。

**3.**文件上传的控件相当于<input  type="file" name="**upload**" .../>，其name是”**upload**”，知道了name那么就可以在Action中获取这个文件。

private File upload;  //文件

private String uploadContentType;  //文件类型

private String uploadFileName;   //文件名

以上三个私有变量都要有set方法。如果不了解的话可以先学习一下Struts 2文件上传。

**4.**如果上传的图片格式不正确，可以在上传界面进行提示。



这个提示不是ckeditor提示的，要在Action中响应。

HttpServletResponse response =ServletActionContext.getResponse();

response.setCharacterEncoding("GBK");

PrintWriter out = response.getWriter();

if(???){

            out.print("<font color=\"red\"size=\"2\">\*文件格式不正确（必须为.jpg/.gif/.bmp/.png文件）</font>");

            return null;

}

**5.**

  InputStream is = newFileInputStream(upload);

  String uploadPath = ServletActionContext.getServletContext().getRealPath("/img/postImg");   //设置保存目录

  String fileName =java.util.UUID.randomUUID(); //采用UUID的方式随机命名

  fileName+= uploadFileName.substring(uploadFileName.length() - 4);

  File toFile = new File(uploadPath, fileName);

  OutputStream os = new FileOutputStream(toFile);

  byte[] buffer = new byte[1024];

  int length = 0;

  while ((length = is.read(buffer)) > 0) {

        os.write(buffer, 0, length);

  }

  is.close();

  os.close();

这段代码是Struts 2上传图片的核心代码，把图片上传后保存在项目的某个目录下，并随机重命名。

**6.**图片上传成功，在目录下也可以看到图片，至此图片上传成功。但是如何将图片发到编辑器中呢？

点“确定”按钮会有以下提示。



到这里，要在Action中加入一段JS。

String callback =ServletActionContext.getRequest().getParameter("CKEditorFuncNum");

out.println("<scripttype=\"text/javascript\">");

out.println("window.parent.CKEDITOR.tools.callFunction("+ callback + ",'" +"img/postImg/"+ fileName + "','')");

out.println("</script>");

有了这段代码，图片上传成功后，根据这里的

"img/postImg/" + filename

相对地址，就可以使用这个图片，直接转到“图像”页面。



附：Struts 2 Action代码

**[java]** [view plaincopy](http://blog.csdn.net/xiao__gui/article/details/7684505)

1. **public** **class** CkeditorUpload **extends** ActionSupport {
3. **private** File upload;
4. **private** String uploadContentType;
5. **private** String uploadFileName;

8. **public** File getUpload() {
9. **return** upload;
10. }
12. **public** **void** setUpload(File upload) {
14. **this**.upload = upload;
15. }
17. **public** String getUploadContentType() {
18. **return** uploadContentType;
19. }
21. **public** **void** setUploadContentType(String uploadContentType) {
22. **this**.uploadContentType = uploadContentType;
23. }
25. **public** String getUploadFileName() {
26. **return** uploadFileName;
27. }
29. **public** **void** setUploadFileName(String uploadFileName) {
30. **this**.uploadFileName = uploadFileName;   }
32. **public** String execute() **throws** Exception {
33. HttpServletResponse response = ServletActionContext.getResponse();
34. response.setCharacterEncoding("GBK");
35. PrintWriter out = response.getWriter();

38. //对文件进行校验
39. **if**(upload==**null** || uploadContentType==**null** || uploadFileName==**null**){
40. out.print("<font color=\"red\" size=\"2\">\*请选择上传文件</font>");
41. **return** **null**;
42. }
44. **if** ((uploadContentType.equals("image/pjpeg") || uploadContentType.equals("image/jpeg"))
45. && uploadFileName.substring(uploadFileName.length() - 4).toLowerCase().equals(".jpg")) {
46. //IE6上传jpg图片的headimageContentType是image/pjpeg，而IE9以及火狐上传的jpg图片是image/jpeg
47. }**else** **if**(uploadContentType.equals("image/png") && uploadFileName.substring(uploadFileName.length() - 4).toLowerCase().equals(".png")){
49. }**else** **if**(uploadContentType.equals("image/gif") && uploadFileName.substring(uploadFileName.length() - 4).toLowerCase().equals(".gif")){
51. }**else** **if**(uploadContentType.equals("image/bmp") && uploadFileName.substring(uploadFileName.length() - 4).toLowerCase().equals(".bmp")){
53. }**else**{
54. out.print("<font color=\"red\" size=\"2\">\*文件格式不正确（必须为.jpg/.gif/.bmp/.png文件）</font>");
55. **return** **null**;
56. }
58. **if**(upload.length() > 600\*1024){
59. out.print("<font color=\"red\" size=\"2\">\*文件大小不得大于600k</font>");
60. **return** **null**;
61. }
63. //将文件保存到项目目录下
64. InputStream is = **new** FileInputStream(upload);
65. String uploadPath = ServletActionContext.getServletContext()
66. .getRealPath("/img/postImg");   //设置保存目录
67. String fileName = java.util.UUID.randomUUID();  //采用UUID的方式随机命名
68. fileName += uploadFileName.substring(uploadFileName.length() - 4);
69. File toFile = **new** File(uploadPath, fileName);
70. OutputStream os = **new** FileOutputStream(toFile);
71. **byte**[] buffer = **new** **byte**[1024];
72. **int** length = 0;
73. **while** ((length = is.read(buffer)) > 0) {
74. os.write(buffer, 0, length);
75. }
76. is.close();
77. os.close();

80. //设置返回“图像”选项卡
81. String callback = ServletActionContext.getRequest().getParameter("CKEditorFuncNum");
82. out.println("<script type=\"text/javascript\">");
83. out.println("window.parent.CKEDITOR.tools.callFunction(" + callback + ",'" + "img/postImg/" + fileName + "','')");
84. out.println("</script>");

87. **return** **null**;
88. }
89. }

## tabletools

## tastytable

# angularjs

## angular处理json示例

<!DOCTYPE HTML>

<html ng-app="exampleApp">

<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<script src="scripts/libs/angular.js"></script>

<script>

var exampleApp = angular.module('exampleApp',[]);

exampleApp.controller('ShowController',['$scope', function($scope){

$scope.fromJson = function() {

var strUser = "{\"Data\":[{\"TotalNum\":\"1\",\"ParentDeptId\":\"11\",\"BookingEndTime\":\"111\"},{\"TotalNum\":\"2\",\"ParentDeptId\":\"22\",\"BookingEndTime\":\"222\"}],\"ErrorMessage\":\"\",\"Error\":\"0\"}";

var objUser = angular.fromJson(strUser);

alert(objUser.Error);

angular.forEach(objUser.Data, function(obj, key) {

alert(obj.TotalNum);

});

}

}]);

</script>

</head>

<body>

<div id="test">angular处理Json数据</div>

<div>

<div id="myCtrl" ng-controller="ShowController">

<input type="button" id="btnToJson" ng-click="fromJson()" value="objUser.firstName" />

</div>

</div>

</body>

</html>

## Angular.JS 与node.JS架构中基于token身份验证实现

# 1. 前面帖子[身份验证中Cookies与 Tokens比较](http://www.jdon.com/46054)已经说明令牌比cookie好，下面是AngularJS中的实现。 服务器后端 首先安装express-jwt: $ npm install express-jwt 配置express保护/api的每个调用： var expressJwt = require('express-jwt'); // We are going to protect /api routes with JWT app.use('/api', expressJwt({secret: secret})); app.use(express.json()); app.use(express.urlencoded()); Angular应用使用用户证书通过AJAX提交POST:

|  |
| --- |
| app.post('/authenticate', function (req, res) {  *//TODO validate req.body.username and req.body.password*  *//if is invalid, return 401*  **if** (!(req.body.username === 'john.doe' && req.body.password === 'foo[author]bar[/author]')) {  res.send(401, 'Wrong user or password');  **return**;  }   **var** profile = {  first\_name: 'John',  last\_name: 'Doe',  email: 'john@doe.com',  id: 123  };   *// We are sending the profile inside the token*  **var** token = jwt.sign(profile, secret, { expiresInMinutes: 60\*5 });   res.json({ token: token }); }); |

# 2. 下面是调用 /api/restricted获得一个资源，其证书检查由expressJwt中间件执行。

|  |
| --- |
| app.get('/api/restricted', function (req, res) {  console.log('user ' + req.user.email + ' is calling /api/restricted');  res.json({  name: 'foo'  }); }); |

# 3. 前端Angular.js 客户端首先用AngularJS 获得JWT的令牌，为了这样做我们还需要用户证书，我们将将创建一个表单让用户输入他们的用户名和密码。

|  |
| --- |
| <div ng-controller="UserCtrl">  <span></span>  <form ng-submit="submit()">  <input ng-model="user.username" type="text" name="user" placeholder="Username" />  <input ng-model="user.password" type="password" name="pass" placeholder="Password" />  <input type="submit" value="Login" />  </form> </div> |

# 4. 一个控制器处理提交的action:

|  |
| --- |
| myApp.controller('UserCtrl', function ($scope, $http, $window) {  $scope.user = {username: 'john.doe', password: 'foo[author]bar[/author]'};  $scope.message = '';  $scope.submit = function () {  $http  .post('/authenticate', $scope.user)  .success(function (data, status, headers, config) {  $window.sessionStorage.token = data.token;  $scope.message = 'Welcome';  })  .error(function (data, status, headers, config) {  *// Erase the token if the user fails to log in*  delete $window.sessionStorage.token;   *// Handle login errors here*  $scope.message = 'Error: Invalid user or password';  });  }; }); |

# 5. 现在我们已经有JWT存储在sessionStorage中，如果这个令牌被设置，我们可以为每个发出请求设置Authorization 头部，只要用$http方式即可，头部值部分使用Bearer<token>

|  |
| --- |
| myApp.factory('authInterceptor', function ($rootScope, $q, $window) {  **return** {  request: function (config) {  config.headers = config.headers || {};  **if** ($window.sessionStorage.token) {  config.headers.Authorization = 'Bearer ' + $window.sessionStorage.token;  }  **return** config;  },  response: function (response) {  **if** (response.status === 401) {  *// handle the case where the user is not authenticated*  }  **return** response || $q.when(response);  }  }; });  myApp.config(function ($httpProvider) {  $httpProvider.interceptors.push('authInterceptor'); }); |

# 6. 尽管sessionStorage并不是所有浏览器支持，你可以使用 polyfill替代cookie保存在localStorage，这个数据能够持久到浏览器标签页关闭。 最后，我们能够向一个授权资源发出请求：

|  |
| --- |
| $http({url: '/api/restricted', method: 'GET'}) .success(function (data, status, headers, config) {  console.log(data.name); *// Should log 'foo'* }); |

# 7. 服务器后端控制台会显示： user foo@[bar](http://www.jdon.com/blog/bar).com is calling /api/restricted

## 如何使用AngularJS的ngShow和ngHide

今天我们来看看怎样使用Angular的[ngShow](http://docs.angularjs.org/api/ng/directive/ngShow) 和[ngHide](http://docs.angularjs.org/api/ng/directive/ngHide) 指令来完成它们听起来应该完成的，显示和隐藏！

**它们应该做的事**

ngShow 和ngHide 允许我们显示或隐藏不同的元素。这有助于创建Angular应用时因为我们的单页程序会有许多的移动部件随着应用状态的改变而来来去去。

这些指令的最伟大的部分就是我们不必使用CSS或者JS来操作显示还是隐藏。这些都是由老练的Angular来完成。

**用法**

想使用ngShow 或ngHide, 只需将指令添加到你想要显示或者隐藏的元素上即可。

|  |  |  |
| --- | --- | --- |
| 01 | <!-- FOR BOOLEAN VALUES =============================== --> | |
| 02 | <!-- for true values --> |

|  |  |  |
| --- | --- | --- |
| 03 | <div ng-show="hello">this is a welcome message</div> | |
| 04 |  |

|  |  |
| --- | --- |
| 05 | <!-- can also show if a value is false --> |
| 06 | <div ng-show="!hello">this is a goodbye message</div> | |

|  |  |
| --- | --- |
| 07 |  |
| 08 | <!-- FOR EXPRESSIONS =============================== --> | |

|  |  |  |
| --- | --- | --- |
| 09 | | <!-- show if the appState variable is a string of goodbye --> |
| 10 | <div ng-show="appState == 'goodbye'">this is a goodbye message</div> | |

|  |  |
| --- | --- |
| 11 |  |
| 12 | <!-- FOR FUNCTIONS =============================== --> | |

|  |  |  |  |
| --- | --- | --- | --- |
| 13 | <!-- use a function defined in your controller to evaluate if true or false --> | | |
| 14 | | <div ng-hide="checkSomething()"></div> |

一旦我们设置了我们标记，我们可以多种方式设置hello或者goodbye。你可以将它设置到你的Angular控制器里并且让你的div在应用加载的时候显示或隐藏。

上面的这些都可以用于ng-show或者ng-hide。如果值或表达式或函数返回true，则会隐藏一些东西。

**用作布尔值**

我们将创建使用ng-click的链接，并切换goCats的值为**true**或**false**。

|  |  |
| --- | --- |
| 1 | <a href ng-click="goCats = !goCats">Toggle Cats</a> |

然后我们可以使用ng-show来显示或隐藏分类图像。

[view source](http://www.oschina.net/translate/how-to-use-ngshow-and-nghide#viewSource)[print](http://www.oschina.net/translate/how-to-use-ngshow-and-nghide#printSource)[?](http://www.oschina.net/translate/how-to-use-ngshow-and-nghide#about)

|  |  |
| --- | --- |
| 1 | <img ng-src="http://i.imgur.com/vkW3Lhe.jpg" ng-show="goCats"> |

**ng-src** 我们使用ng-src来调用图片，这样的话Angular将在实例化及检查图像时确认是否将其隐藏。如果我们不这样的话，图片会在站点加载时弹出来，直到Angular意识到应该将其隐藏。

**判断表达式**

在这里我们来判断一个从输入框传来的字符串，我们把ng-mode绑定在一个输入框的里面并命名为aminal变量，并根据这个变量的内容来显示不同的图片。

下面我们会把我们的变量名焦作aminal。

|  |  |
| --- | --- |
| 1 | <input type="text" ng-model="aminal"> |

然后我们会使用ng-show来对字符串进行判断。

|  |  |
| --- | --- |
| 1 | <img ng-src="http://i.imgur.com/vkW3Lhe.jpg" ng-show="aminal == 'cat'"> |

**使用方法**

我们会做一个简单的检查来判断输入的是奇数还是偶数。 我们会创建一个方法在我们的AngularJS文件中：

|  |  |  |
| --- | --- | --- |
| 01 | // set the default value of our number | |
| 02 | $scope.myNumber = 0; |

|  |  |
| --- | --- |
| 03 |  |
| 04 | // function to evaluate if a number is even | |

|  |  |  |
| --- | --- | --- |
| 05 | $scope.isEven = function(value) { | |
| 06 |  |

|  |  |  |
| --- | --- | --- |
| 07 | if (value % 2 == 0) | |
| 08 | return true; |

|  |  |
| --- | --- |
| 09 | else |
| 10 | return false; | |

|  |  |
| --- | --- |
| 11 |  |
| 12 | }; |

方法创建完成之后, 我们接下来要做的就是通过ng-show或者ng-hide来使用它，并传入我们的数字。通过方法传数字这种方法，可以保持Angular控制整洁和可测试。

|  |  |  |
| --- | --- | --- |
| 1 | <!-- show if our function evaluates to false --> | |
| 2 | <div ng-show="isEven(myNumber)"> |

|  |  |  |
| --- | --- | --- |
| 3 | <h2>The number is even.</h2> | |
| 4 | </div> |

|  |  |
| --- | --- |
| 5 |  |
| 6 | <!-- show if our function evaluates to false --> | |

|  |  |
| --- | --- |
| 7 | <div ng-show="!isEven(myNumber)"> |
| 8 | <h2>The number is odd.</h2> | |

|  |  |
| --- | --- |
| 9 | </div> |

**结论**

　　有了这两个方向的指引,我们的应用会有很大的改观. 而这些只是基于 **布尔值**, **表达式**, 以及 **函数**,实现的元素显示和隐藏功能 但这三种模式将能应用到更多的场景.

　　希望本文能帮助你构建更好的AngularJS应用.在后续的文章中我们会介绍 ngShow 和 ngHide 以创建更多动态展示的应用.

## angularjs filter 详解

系统的学习了一下angularjs，发现angularjs的有些思想根php的模块smarty很像，例如数据绑定，filter。如果对smarty比较熟悉的话，学习angularjs会比较容易一点。这篇简单说一下angularjs的filter功能，angularjs的filter功能可分为二种，一种是内置的过滤器，一种是自定义的。

**一，内置的过滤器**

**1，uppercase，lowercase大小转换**

[查看复制打印?](http://blog.51yip.com/jsjquery/1592.html)

1. {{ "lower cap string" | uppercase }}     //结果：LOWER CAP STRING
2. {{ "TANK is GOOD" | lowercase }}         //结果：tank is good

**｜这里的竖线是一种管道功能，如果对linux比较熟悉的话，这块的｜根linux的管道功能，基本是一样的**

**2，json格式化**

[查看复制打印?](http://blog.51yip.com/jsjquery/1592.html)

1. {{ {foo: "bar", baz: 23} | json }}    //结果：{ "foo": "bar", "baz": 23 }

注意：bza没格式前是没有双引号的，格式化后就转换成了json数据了。

**3，date格式化**

[查看复制打印?](http://blog.51yip.com/jsjquery/1592.html)

1. {{ 1304375948024 | date }}                          //结果：May 3, 2011
2. {{ 1304375948024 | date:"MM/dd/yyyy @ h:mma" }}     //结果：05/03/2011 @ 6:39AM
3. {{ 1304375948024 | date:"yyyy-MM-dd hh:mm:ss" }}    //结果：2011-05-03 06:39:08

**4，number格式化**

[查看复制打印?](http://blog.51yip.com/jsjquery/1592.html)

1. {{ 1.234567 | number:1 }}    //结果：1.2
2. {{ 1234567 | number }}       //结果：1,234,567

**5，currency货币格式化**

[查看复制打印?](http://blog.51yip.com/jsjquery/1592.html)

1. {{ 250 | currency }}                 //结果：$250.00
2. {{ 250 | currency:"RMB ￥ " }}       //结果：RMB ￥ 250.00

**6，filter查找**

[查看复制打印?](http://blog.51yip.com/jsjquery/1592.html)

1. {{ [{"age": 20,"id": 10,"name": "iphone"},
2. {"age": 12,"id": 11,"name": "sunm xing"},
3. {"age": 44,"id": 12,"name": "test abc"}
4. ] | filter:'s'}}    //查找含有有s的行
6. //上例结果：[{"age":12,"id":11,"name":"sunm xing"},{"age":44,"id":12,"name":"test abc"}]
8. {{ [{"age": 20,"id": 10,"name": "iphone"},
9. {"age": 12,"id": 11,"name": "sunm xing"},
10. {"age": 44,"id": 12,"name": "test abc"}
11. ] | filter:{'name':'iphone'} }}   //查找name为iphone的行
13. //上例结果:[{"age":20,"id":10,"name":"iphone"}]

**7，limitTo字符串，对像的截取**

[查看复制打印?](http://blog.51yip.com/jsjquery/1592.html)

1. {{ "i love tank" | limitTo:6 }}           //结果：i love
2. {{ "i love tank" | limitTo:-4 }}          //结果：tank
4. {{ [{"age": 20,"id": 10,"name": "iphone"},
5. {"age": 12,"id": 11,"name": "sunm xing"},
6. {"age": 44,"id": 12,"name": "test abc"}
7. ] | limitTo:1 }}     //结果：[{"age":20,"id":10,"name":"iphone"}]

**8，orderBy对像排序**

[查看复制打印?](http://blog.51yip.com/jsjquery/1592.html)

1. {{ [{"age": 20,"id": 10,"name": "iphone"},
2. {"age": 12,"id": 11,"name": "sunm xing"},
3. {"age": 44,"id": 12,"name": "test abc"}
4. ] | orderBy:'id':true }}        //根id降序排
6. {{ [{"age": 20,"id": 10,"name": "iphone"},
7. {"age": 12,"id": 11,"name": "sunm xing"},
8. {"age": 44,"id": 12,"name": "test abc"}
9. ] | orderBy:'id' }}           //根据id升序排

**二，自定filter功能**

**我找了一个基本angularjs的mvc框架，phonecat，自定义filter也是在这基础写的，这个框架挺好用的。**

**1，filters.js添加一个module**

[查看复制打印?](http://blog.51yip.com/jsjquery/1592.html)

1. angular.module('tanktest', []).filter('tankreplace', **function**() {
2. **return** **function**(input) {
3. **return** input.replace(/tank/, "=====")
4. };
5. });

**2，app.js中加载这个module**

[查看复制打印?](http://blog.51yip.com/jsjquery/1592.html)

1. **var** phonecatApp = angular.module('phonecatApp', [
2. 'ngRoute',
3. 'phonecatControllers',
4. 'facebookControllers',
5. 'tanktest'
6. ]);

**3，html中调用**

[查看复制打印?](http://blog.51yip.com/jsjquery/1592.html)

1. {{ "TANK is GOOD" | lowercase |tankreplace}}   //结果：===== is good

**注意：| lowercase |tankreplace管道命令可以有多个**

## timeago 时间显示成 多少小时前、多少分前 的格式

angular.module('healthApp')

.filter('timeago', function(){

return function(date){

return moment(date).fromNow();

};

})

//汉化

.filter('timeagoChinese', function(){

return function(input){

return input.replace('days ago', '天前')

.replace('a day ago', '1天前')

.replace('hours ago', '小时前')

.replace('an hour ago', '1小时前')

.replace('minutes ago', '分前')

.replace('a minute ago', '1分钟前')

.replace('a few seconds ago', '几秒前');

};

})

//页面上的使用方法, createTime 是yyyy-MM-dd HH:mm:ss 格式

<small class="text-muted">{{replyItem.createTime | timeago | timeagoChinese}}</small>

## [angularjs分页教程](http://www.ddhigh.com/2014/09/angularjs%e5%88%86%e9%a1%b5%e6%95%99%e7%a8%8b/)

angularjs强大的双向绑定，以及对SEO的不友好…暂时让它只能在后台管理系统方面大展身手，而管理系统，一个很重要的组件就是分页了。angularjs实现分页也不是很难。看代码

控制器

1. app.controller('BusinessCtrl', ['$scope', '$rootScope', 'BusinessService', **function** ($scope, $rootScope, BusinessService) {
2. $rootScope.title = '流水列表';
4. $scope.currentPage = 1;
5. $scope.totalPage = 1;
6. $scope.pageSize = 40;
7. $scope.pages = [];
8. $scope.endPage = 1;
9. //获取总流水
10. BusinessService.total().success(**function** (data) {
11. $scope.total = data;
12. });
14. $scope.load = **function** () {
15. BusinessService.list($scope.currentPage, $scope.pageSize).success(**function** (data) {
16. $scope.items = data.list;
17. //获取总页数
18. $scope.totalPage = Math.ceil(data.count / $scope.pageSize);
19. $scope.endPage = $scope.totalPage;
20. //生成数字链接
21. **if** ($scope.currentPage > 1 && $scope.currentPage < $scope.totalPage) {
22. $scope.pages = [
23. $scope.currentPage - 1,
24. $scope.currentPage,
25. $scope.currentPage + 1
26. ];
27. } **else** **if** ($scope.currentPage == 1 && $scope.totalPage > 1) {
28. $scope.pages = [
29. $scope.currentPage,
30. $scope.currentPage + 1
31. ];
32. } **else** **if** ($scope.currentPage == $scope.totalPage && $scope.totalPage > 1) {
33. $scope.pages = [
34. $scope.currentPage - 1,
35. $scope.currentPage
36. ];
37. }
38. });
39. };
41. $scope.**next** = **function** () {
42. **if** ($scope.currentPage < $scope.totalPage) {
43. $scope.currentPage++;
44. $scope.load();
45. }
46. };
48. $scope.prev = **function** () {
49. **if** ($scope.currentPage > 1) {
50. $scope.currentPage--;
51. $scope.load();
52. }
53. };
55. $scope.loadPage = **function** (page) {
56. $scope.currentPage = page;
57. $scope.load();
58. };
60. }]);

Service

1. //流水业务类
2. app.factory('BusinessService', ['$http', **function** ($http) {
3. **var** list = **function** (page, size) {
4. **return** $http({
5. **params**: {
6. page: page,
7. size: size
8. },
9. url: '/merchants/business/list'
10. });
11. };
13. **var** total = **function** () {
14. **return** $http.**get**('/merchants/business/total');
15. };
17. **var** post = **function** (business) {
18. **return** $http.post('/merchants/business/post', business);
19. };
21. **return** {
22. list: **function** (page, size) {
23. **return** list(page, size);
24. },
25. total: **function** () {
26. **return** total();
27. },
28. post: **function** (business) {
29. **return** post(business);
30. }
31. };
32. }]);

html部分

1. **<ul** **class**="pagination" **ng-show**="totalPage>1">
2. **<li** **ng-class**="{true:'active'}[currentPage==1]"**><a** **href**="javascript:void(0)"
3. **ng-click**="currentPage=1;load()"**>**首页**</a></li>**
4. **<li** **ng-class**="{true:'disabled'}[currentPage==1]"**><a** **href**="javascript:void(0);" **ng-click**="prev()"**>**上一页**</a></li>**
5. **<li** **ng-repeat**="page in pages"**><a** **href**="javascript:void(0);" **ng-click**="loadPage(page)"**>**{{ page }}**</a></li>**
6. **<li** **ng-class**="{true:'disabled'}[currentPage==totalPage]"**><a** **href**="javascript:void(0);" **ng-click**="next()"**>**下一页**</a>**
7. **</li>**
8. **<li** **ng-class**="{true:'active'}[currentPage==totalPage]"**><a** **href**="javascript:void(0)"
9. **ng-click**="currentPage=totalPage;load()"**>**末页**</a></li>**
10. **</ul>**

其实，关键在于生成数字链接那里有点麻烦，仔细想一下执行过程就容易写了。

好了。这里是个完整的分页代码，以后有啥分页的都可以这样写啦~

## 一个angular登陆表单页面

<div class="page-signin row">

<div class="page-form col-md-4 col-md-offset-4 col-sm-8 col-sm-offset-2" >

<form class="form" name="loginForm" ng-submit="doLogin()" ng-controller="LoginCtrl" method="post" novalidate="">

<div class="header-content text-center">

<h1>银医通管理平台</h1>

</div>

<div class="body-content">

<div role="alert" class="alert alert-danger alert-dismissible " ng-show="authenticationError">

{{msg}}

</div>

<div class="list-group">

<div class="list-group-item">

<input type="text" name="username" ng-model="username" placeholder="用户名" required class="form-control"/>

</div>

<div class="list-group-item">

<input type="password" ng-model="password" placeholder="密码" required class="form-control"/>

</div>

</div>

<div class="clearfix"></div>

<div class="row">

<div class="col-md-6 col-sm-6 col-md-offset-3 col-sm-offset-3">

<button type="submit" class="btn btn-primary btn-lg btn-block" ng-disabled="loginForm.$invalid">登陆</button>

</div>

</div>

<hr/>

</div>

</form>

</div>

</div>

## [angularjs学习笔记—事件指令](http://segmentfault.com/a/1190000002634554)

## ngClick

**适用标签：所有**  
**触发条件：单击**

#html

<div ng-controller="LearnCtrl">

<div ng-click="click()">click me</div>

<button ng-click="click()">click me</button>

</div>

#script

angular.module('learnModule', [])

.controller('LearnCtrl', function ($scope) {

$scope.click = function () {

alert('click');

}

});

## ngDblclick

**适用标签：所有**  
**触发条件：双击**

#html

<div ng-controller="LearnCtrl">

<div ng-dblclick="dblclick()">click me</div>

<button ng-dblclick="dblclick()">click me</button>

</div>

#script

angular.module('learnModule', [])

.controller('LearnCtrl', function ($scope) {

$scope.dblclick = function () {

alert('click');

}

});

## ngBlur

**适用标签：**

* **a**
* **input**
* **select**
* **textarea**

**触发条件：失去焦点**

#html

<div ng-controller="LearnCtrl">

<a href="" ng-blur="blur()">link</a>

<input type="text" ng-blur="blur()"/>

<textarea cols="30" rows="10" ng-blur="blur()"></textarea>

<select ng-blur="blur()">

<option>----</option>

<option>jacky</option>

<option>rose</option>

</select>

</div>

#script

angular.module('learnModule', [])

.controller('LearnCtrl', function ($scope) {

$scope.blur = function () {

alert('blur');

}

});

## ngFocus

**适用标签：**

* **a**
* **input**
* **select**
* **textarea**

**触发条件：获取焦点**

#html

<div ng-controller="LearnCtrl">

<a href="" ng-focus="focus()">link</a>

<input type="text" ng-focus="focus()"/>

<textarea cols="30" rows="10" ng-focus="focus()"></textarea>

<select ng-focus="focus()">

<option>----</option>

<option>jacky</option>

<option>rose</option>

</select>

</div>

#script

angular.module('learnModule', [])

.controller('LearnCtrl', function ($scope) {

$scope.focus= function () {

alert('focus');

}

});

## ngChange

**适用标签：input**  
**触发条件：model更新**

输入框的内容改变并不代表model的值更新。按我的理解，一般当两个状态互相切换时，model值会更新。两个状态我称之为合法状态和不合法状态。

不合法的状态：输入的内容不符合type类型，如email类型。输入的内容不符合校验条件，如ngMinlength。不合法的状态下，**model会被更新成undefined**。

合法的状态：输入的内容是符合类型和校验条件的。

#html

<div ng-controller="LearnCtrl">

<input type="text" ng-model="text" ng-change="change()" ng-minlength="5"/>

</div>

#script

angular.module('learnModule', [])

.controller('LearnCtrl', function ($scope) {

//$scope.text='';

$scope.change = function () {

alert('change');

}

});

初始化和不初始化text的条件下，change触发是不一样的哦，这里涉及到model初始化和更新机制。

## ngCopy

**适用标签：**

* **a**
* **input**
* **select**
* **textarea**

官方api上说使用的标签是这些，我没明白a和select复制有啥子用。另外，我换个div实际上也能触发copy事件。一般常用的就是input和textarea。

**触发条件：复制。鼠标右键复制和快捷键Ctrl+C都会触发。**

#html

<div ng-controller="LearnCtrl">

<input type="text" ng-copy="copy()"/>

<textarea cols="30" rows="10" ng-copy="copy()"></textarea>

</div>

#script

angular.module('learnModule', [])

.controller('LearnCtrl', function ($scope) {

$scope.copy = function () {

alert('copy');

}

});

## ngCut

**适用标签：**

* **a**
* **input**
* **select**
* **textarea**

**触发条件：剪切。鼠标右键剪切和快捷键Ctrl+X都会触发。**

#html

<div ng-controller="LearnCtrl">

<input type="text" ng-cut="cut()"/>

<textarea cols="30" rows="10" ng-cut="cut()"></textarea>

</div>

#script

angular.module('learnModule', [])

.controller('LearnCtrl', function ($scope) {

$scope.cut = function () {

alert('cut');

}

});

## ngPaste

**适用标签：**

* **a**
* **input**
* **select**
* **textarea**

**触发条件：粘贴。鼠标右键粘贴和快捷键Ctrl+V都会触发。**

#html

<div ng-controller="LearnCtrl">

<input type="text" ng-paste="paste()"/>

<textarea cols="30" rows="10" ng-paste="paste()"></textarea>

</div>

#script

angular.module('learnModule', [])

.controller('LearnCtrl', function ($scope) {

$scope.paste = function () {

alert('paste');

}

});

## ngKeydown

**适用标签：所有**

个人感觉还是input和textarea比较常用

**触发条件：键盘按键按下**

要把$event传过去，一般都是要判断按了哪个按键的。

#html

<div ng-controller="LearnCtrl">

<input type="text" ng-keydown="keydown($event)"/>

<textarea cols="30" rows="10" ng-keydown="keydown($event)"></textarea>

</div>

#script

angular.module('learnModule', [])

.controller('LearnCtrl', function ($scope) {

$scope.keydown = function ($event) {

alert($event.keyCode);

}

});

## ngKeyup

**适用标签：所有**

个人感觉还是input和textarea比较常用

**触发条件：键盘按键按下并松开**

#html

<div ng-controller="LearnCtrl">

<input type="text" ng-keyup="keyup($event)"/>

<textarea cols="30" rows="10" ng-keyup="keyup($event)"></textarea>

</div>

#script

angular.module('learnModule', [])

.controller('LearnCtrl', function ($scope) {

$scope.keyup = function ($event) {

alert($event.keyCode);

}

});

## ngKeypress

**适用标签：所有**

个人感觉还是input和textarea比较常用

**触发条件：键盘按键按下**

#html

<div ng-controller="LearnCtrl">

<input type="text" ng-keypress="keypress($event)"/>

<textarea cols="30" rows="10" ng-keypress="keypress($event)"></textarea>

</div>

#script

angular.module('learnModule', [])

.controller('LearnCtrl', function ($scope) {

$scope.keypress = function ($event) {

alert($event.keyCode);

}

});

## keydown，keypress，keydown三者区别

### 引发事件的按键

非字符键不会引发 KeyPress 事件，但非字符键却可以引发 KeyDown 和 KeyUp 事件。

### 事件引发的时间

KeyDown 和 KeyPress 事件在按下键时发生，KeyUp 事件在释放键时发生。

### 事件发生的顺序

KeyDown -> KeyPress -> KeyUp。如果按一个键很久才松开，发生的事件为：KeyDown -> KeyPress -> KeyDown -> KeyPress -> KeyDown -> KeyPress -> ... -> KeyUp。

* KeyDown触发后，不一定触发KeyUp，当KeyDown 按下后，拖动鼠标，那么将不会触发KeyUp事件。
* KeyPress主要用来捕获数字(注意：包括Shift+数字的符号)、字母（注意：包括大小写）、小键盘等除了F1-12、SHIFT、Alt、Ctrl、Insert、Home、PgUp、Delete、End、PgDn、ScrollLock、Pause、NumLock、{菜单键}、{开始键}和方向键外的ANSI字符。
* KeyDown 和KeyUp 通常可以捕获键盘除了PrScrn所有按键(这里不讨论特殊键盘的特殊键）。
* KeyPress 只能捕获单个字符。
* KeyDown 和KeyUp 可以捕获组合键。
* KeyPress 可以捕获单个字符的大小写。
* KeyDown和KeyUp 对于单个字符捕获的KeyValue 都是一个值，也就是不能判断单个字符的大小写。
* KeyPress 不区分小键盘和主键盘的数字字符。
* KeyDown 和KeyUp 区分小键盘和主键盘的数字字符。
* 其中PrScrn 按键KeyPress、KeyDown和KeyUp 都不能捕获。

## ngMousedown

**适用标签：所有**  
**触发条件：鼠标按下，左右中间按下都会触发**

#html

<div ng-controller="LearnCtrl">

<button ng-mousedown="mousedown($event)">button</button>

</div>

#script

angular.module('learnModule', [])

.controller('LearnCtrl', function ($scope) {

$scope.mousedown = function ($event) {

alert($event.which);

}

});

## ngMouseup

**适用标签：所有**  
**触发条件：鼠标按下弹起，左右中间按下弹起都会触发**

#html

<div ng-controller="LearnCtrl">

<button ng-mouseup="mouseup($event)">button</button>

</div>

#script

angular.module('learnModule', [])

.controller('LearnCtrl', function ($scope) {

$scope.mouseup = function ($event) {

alert($event.which);

}

});

## ngMouseenter

**适用标签：所有**  
**触发条件：鼠标进入**

#html

<div ng-controller="LearnCtrl">

<button ng-mouseenter="mouseenter()">button</button>

</div>

#script

angular.module('learnModule', [])

.controller('LearnCtrl', function ($scope) {

$scope.mouseenter = function () {

alert('mouseenter');

}

});

## ngMouseleave

**适用标签：所有**  
**触发条件：鼠标离开**

#html

<div ng-controller="LearnCtrl">

<button ng-mouseleave="mouseleave()">button</button>

</div>

#script

angular.module('learnModule', [])

.controller('LearnCtrl', function ($scope) {

$scope.mouseleave = function () {

alert('mouseleave');

}

});

## ngMousemove

**适用标签：所有**  
**触发条件：鼠标移动**

#html

<div ng-controller="LearnCtrl">

<button ng-mousemove="mousemove()">button</button>

</div>

#script

angular.module('learnModule', [])

.controller('LearnCtrl', function ($scope) {

$scope.mousemove = function () {

alert('mousemove');

}

});

## ngMouseover

**适用标签：所有**  
**触发条件：鼠标进入**  
个人感觉和ngMouseenter没啥子区别，求大神指导下。最好有代码

## angularJS中的$injector、$rootScope和$scope的概念和关联关系

$injector、$rootScope和$scope是angularJS框架中比较重要的东西，理清它们之间的关系，对我们后续学习和理解 angularJS 框架都非常有用。

1、 **$injector其实是一个IOC容器** ，包含了很多服务(类似于spring框架中的bean)，其它代码能够通过        $injector.get("serviceName")的方式，从injector中获取所需要的服务。详情参考 [***这篇文章***](http://blog.csdn.net/aitangyong/article/details/39937505)

2、 **scope是angularJS中的作用域(其实就是存储数据的地方)，很类似javascript的原型链**。搜索的时候，优先找自己的scope，如果没有找到就沿着作用域链向上搜索，直至到达根作用域rootScope。

3、 $rootScope是由angularJS加载模块的时候自动创建的，每个模块只会有1个rootScope。rootScope创建好会以服务的形式加入到 $injector中。也就是说通过 $injector.get("$ rootScope ");能够获取到某个模块的根作用域。更准确的来说，$rootScope是由angularJS的核心模块ng创建的。

示例1：

// 新建一个模块

var module = angular.module("app",[]);

// true说明$rootScope确实以服务的形式包含在模块的injector中

var hasNgInjector = angular.injector(['app','ng']);

console.log("has $rootScope=" + hasNgInjector.has("$rootScope"));//true

// 获取模块相应的injector对象,不获取ng模块中的服务

// 不依赖于ng模块,无法获取$rootScope服务

var noNgInjector = angular.injector(['app']);

console.log("no $rootScope=" + noNgInjector.has("$rootScope"));//false

// 获取angular核心的ng模块

var ngInjector = angular.injector(['ng']);

console.log("ng $rootScope=" + ngInjector.has("$rootScope"));//true

上面的代码的确可以说明： **$rootScope的确是由核心模块ng创建的，并以服务的形式存在于injector中**。

如果创建injector的时候，指定了ng模块，那么该injector中就会包含$rootScope服务；否则就不包含$rootScope。

示例2：

**<!doctype html>**

<html lang="en">

<head>

<meta charset="utf-8">

<script src="angular-1.2.25.js"></script>

<script>

**var** module = angular.module("app",[]);

*// 控制器里的$injector,是由angular框架自动创建的*

**function** FirstController($scope,$injector,$rootScope)

{

$rootScope.name="aty";

}

*//自己创建了个injector,依赖于app和ng模块*

**var** myInjector = angular.injector(["app","ng"]);

**var** rootScope = myInjector.get("$rootScope");

alert(rootScope.name);*//udefined*

</script>

</head>

<body ng-app="app">

<div id="first" ng-controller="FirstController">

<input type="text" ng-model="name">

<br>

{{name}}

</div>

</body>

</html>

**angular.injector()可以调用多次，每次都返回新建的injector对象**。所以我们自己创建的myInjector和angular自动创建的$injector不是同一个对象，那么得到的rootScope也就不是同一个。更详细的可以看 [***另一篇文章***](http://blog.csdn.net/aitangyong/article/details/39937505)中的

angular.injector()章节。

示例3：

**<!doctype html>**

<html lang="en">

<head>

<script src="angular-1.2.25.js"></script>

<script>

**function** FirstController($scope,$injector,$rootScope)

{

*// true*

console.log("scope parent :" + ($scope.$parent ==$rootScope));

}

</script>

</head>

<body ng-app>

<div id="first" ng-controller="FirstController">

<input type="text" ng-model="name">

<br>

{{name}}

</div>

</body>

</html>

**ng-controller指令给所在的DOM元素创建了一个新的$scope对象,并作为rootScope的子作用域** 。$scope是由$rootScope创建的，$scope不会保护在$injector中。

示例4：

**<!doctype html>**

<html lang="en">

<head>

<meta charset="utf-8">

<title>scope()</title>

<script src="jquery-1.11.1.js"></script>

<script src="angular-1.2.25.js"></script>

<script>

*//记住rootScope，用来判断跨控制器是否相等*

**var** first\_rootScope = null;

*//记住scope，用来判断跨控制器是否相等*

**var** first\_scope = null;

*//记住injector，用来判断跨控制器是否相等*

**var** first\_injectot = null;

*// 第1个angular控制器*

**function** FirstController($scope,$injector,$rootScope)

{

$rootScope.name = "aty";

first\_rootScope = $rootScope;

first\_injectot = $injector;

first\_scope = $scope;

}

*// 第2个angular控制器,主要是来测试跨controller时injector和scope的表现*

**function** SecondController($scope,$injector,$rootScope)

{

console.log("first\_rootScope==second\_rootScope:" + (first\_rootScope==$rootScope));*//true*

console.log("first\_injectot==second\_injector:" + (first\_injectot==$injector));*//true*

console.log("first\_scope==second\_scope:" + (first\_scope==$scope));*//false*

}

</script>

</head>

<body ng-app>

<div id="first" ng-controller="FirstController">

<input type="text" ng-model="name">

<br>

<div id="tips"></div>

</div>

<h2>outside of controller</h2>

<br>

*<!--访问每一个应用(模块)的rootScope-->*

{{$root.name}}

<div id="noControllerDiv"/>

<div ng-controller="SecondController">

</div>

</body>

</html>

ng-app定义了一个angular模块， **每个模块只有一个$rootScope，只有一个$injector，但可以有多个$scope**。

弄清了 $injector、$rootScope和$scope这3者之间的关系，我们看下angular提供的2个API，一个是scope()，一个是injector()。使用 angular.element()返回的DOM对象，都会包含这2个方法，用来获取与之关联的scope和injector。

由于每个模块的injector是唯一的，所以 **angular.element().injector()直接返回元素所在模块的injector**。

angular.element().scope()可以获取到当前元素的scope或父scope。如果当前元素有scope，则返回自己的scope;如果没有则向父亲方向寻找,如果找不到返回rootScope。即 **返回作用域链上，距离该元素最近的scope**。

**<!doctype html>**

<html lang="en">

<head>

<meta charset="utf-8">

<title>scope()</title>

<script src="jquery-1.11.1.js"></script>

<script src="angular-1.2.25.js"></script>

<script>

**function** FirstController($scope,$injector,$rootScope)

{

*//获取body对象*

**var** domBody = document.getElementsByTagName('body')[0];

*// 通过ng-app指令所在的DOM元素获取rootScope*

**var** rtScope = angular.element(domBody).scope();

*//当前元素没有新作用域,获取父作用域即rootScope*

**var** noScope = angular.element("#noControllerDiv").scope();

*// true*

console.log("rtScope==noScope:" + (rtScope==noScope));

*//ng-controller所在的元素,返回的scope*

**var** scopeOnController = angular.element("#first").scope();

*// ng-controller内部的元素返回所在的scope*

**var** inController = angular.element("#tips").scope();

*//true*

console.log("scopeOnController==inController:" + (scopeOnController==inController));

*//验证通过DOM获取的scope是否与注入的$scope和$rootScope一致*

*//true*

console.log("result1:" + (rtScope==$rootScope));

*//true*

console.log("result2:" + (inController==$scope));

}

</script>

</head>

<body ng-app>

<div id="first" ng-controller="FirstController">

<input type="text" ng-model="name">

<br>

<div id="tips"></div>

</div>

<h2>outside of controller</h2>

<br>

*<!--访问每一个应用(模块)的rootScope-->*

{{$root.name}}

<div id="noControllerDiv"/>

</body>

</html>

## [理解angular中的module和injector，即依赖注入](http://blog.csdn.net/aitangyong/article/details/39937505)

依赖注入（DI）的好处不再赘言，使用过spring框架的都知道。angularjs作为前台js框架，也提供了对DI的支持，这是javascript/jquery不具备的特性。angularjs中与DI相关有angular.module()、angular.injector()、 [$injector](http://docs.angularjs.cn/api/auto/service/$injector)、[$provide](http://docs.angularjs.cn/api/auto/service/$provide)。对于一个DI容器来说，必须具备3个要素：服务的注册、依赖关系的声明、对象的获取。比如spring中，服务的注册是通过xml配置文件的<bean>标签或是注解@Repository、@Service、@Controller、@Component实现的；对象的获取可以ApplicationContext.getBean()实现；依赖关系的声明，即可以在xml文件中配置，也可以使用@Resource等注解在java代码中声明。在angular中，module和$provide相当于是服务的注册；injector用来获取对象（angular会自动完成依赖的注入）；依赖关系的声明在angular中有3种方式。下面从这3个方面，介绍下angular的DI。

**1、angular.module()创建、获取、注册angular中的模块**

The angular.module() is a global place for creating, registering and retrieving Angular modules.When passed two or more arguments, a new module is created. If passed only one argument, an existing module (the name passed as the first argument to module) is retrieved。

**[javascript]** [view plaincopy](http://blog.csdn.net/aitangyong/article/details/39937505)[在CODE上查看代码片](https://code.csdn.net/snippets/481507)

1. // 传递参数不止一个,代表新建模块;空数组代表该模块不依赖其他模块
2. **var** createModule = angular.module("myModule", []);
4. // 只有一个参数(模块名),代表获取模块
5. // 如果模块不存在,angular框架会抛异常
6. **var** getModule = angular.module("myModule");
8. // true,都是同一个模块
9. alert(createModule == getModule);

该函数既可以创建新的模块，也可以获取已有模块，是创建还是获取，通过参数的个数来区分。

angular.module(name, [requires], [configFn]);

name：字符串类型，代表模块的名称；

requires：字符串的数组，代表该模块依赖的其他模块列表，如果不依赖其他模块，用空数组即可；

configFn：用来对该模块进行一些配置。

现在我们知道如何创建、获取模块了，那么模块究竟是什么呢？官方的Developer Guide上只有一句话：You can think of a module as a container for the different parts of your app – controllers, services, filters, directives, etc.现在我还不太理解，大致就是说模块是一些功能的集合，如控制器、服务、过滤器、指令等子元素组成的整体。现在解释不了，先遗留。

**2、$provide和模块的关系**

The [$provide](http://docs.angularjs.cn/api/auto/service/$provide) service has a number of methods for registering components with the [$injector](http://docs.angularjs.cn/api/auto/service/$injector). Many of these functions are also exposed on [angular.Module](http://docs.angularjs.cn/api/ng/type/angular.Module).

之前提到过：module和provide是用来注册服务到injector中的。查看官方的API，可以看到$provide提供了provide()、constant()、value()、factory()、service()来创建各种不同性质的服务；angular.Module中也提供了这5个服务注册方法。其实2者功能是完全一样的，就是用来向DI容器注册服务到injector中。

官方API下的auto有[$provide](http://docs.angularjs.cn/api/auto/service/$provide) 和 [$injector](https://code.angularjs.org/1.2.21/docs/api/auto/service/$injector)，Implicit module which gets automatically added to each $injector.按照字面意思是说，每一个injector都有这2个隐含的服务。但1.2.25版本中，感觉没有办法获取injector中的[$provide](http://docs.angularjs.cn/api/auto/service/$provide)。不知道这是为什么?一般来说也不需要显示使用这个服务，直接使用module中提供的API即可。

**[javascript]** [view plaincopy](http://blog.csdn.net/aitangyong/article/details/39937505)[在CODE上查看代码片](https://code.csdn.net/snippets/481507)

1. **var** injector = angular.injector();
2. alert(injector.has("$provide"));//false
3. alert(injector.has("$injector"));//true

**3、angular.injector()**

使用angular.injector();也能获取到注入器，但是没有和模块绑定。这种做法是没有意义的，相当于是你创建了一个空的DI容器，里面都没有服务别人怎么用呢。正确的做法是，在创建注入器的时候，指定需要加载的模块。

**[javascript]** [view plaincopy](http://blog.csdn.net/aitangyong/article/details/39937505)[在CODE上查看代码片](https://code.csdn.net/snippets/481507)

1. // 创建myModule模块、注册服务
2. **var** myModule = angular.module('myModule', []);
3. myModule.service('myService', **function**() {
4. **this**.my = 0;
5. });
7. // 创建herModule模块、注册服务
8. **var** herModule = angular.module('herModule', []);
9. herModule.service('herService', **function**() {
10. **this**.her = 1;
11. });
13. // 加载了2个模块中的服务
14. **var** injector = angular.injector(["myModule","herModule"]);
15. alert(injector.get("myService").my);
16. alert(injector.get("herService").her);

如果加载了多个模块，那么通过返回的injector可以获取到多个模块下的服务。这个例子中如果只加载了myMoudle，那么得到的injector就不能访问herMoudle下的服务。这里**特别需要注意下：angular.injector()可以调用多次，每次都返回新建的injector对象**。

**[javascript]** [view plaincopy](http://blog.csdn.net/aitangyong/article/details/39937505)[在CODE上查看代码片](https://code.csdn.net/snippets/481507)

1. **var** injector1 = angular.injector(["myModule","herModule"]);
2. **var** injector2 = angular.injector(["myModule","herModule"]);
4. alert(injector1 == injector2);//false

**4、angular中三种声明依赖的方式**

angular提供了3种获取依赖的方式：inference、annotation、inline方式。

**[javascript]** [view plaincopy](http://blog.csdn.net/aitangyong/article/details/39937505)[在CODE上查看代码片](https://code.csdn.net/snippets/481507)

1. // 创建myModule模块、注册服务
2. **var** myModule = angular.module('myModule', []);
3. myModule.service('myService', **function**() {
4. **this**.my = 0;
5. });
7. // 获取injector
8. **var** injector = angular.injector(["myModule"]);
10. // 第一种inference
11. injector.invoke(**function**(myService){alert(myService.my);});
13. // 第二种annotation
14. **function** explicit(serviceA) {alert(serviceA.my);};
15. explicit.$inject = ['myService'];
16. injector.invoke(explicit);
18. // 第三种inline
19. injector.invoke(['myService', **function**(serviceA){alert(serviceA.my);}]);

其中annotation和inline方式，对于函数参数名称没有要求，是推荐的做法；inference方式强制要求参数名称和服务名称一致，如果JS代码经过压缩或者混淆，那么功能会出问题，不建议使用这种方式。

## angularjs scope rootscope 区别

scope是html和单个controller之间的桥梁，数据绑定就靠他了。rootscope是各个controller中scope的桥梁。用rootscope定义的值，可以在各个controller中使用。下面用实例详细的说明一下。

## 常用ng 标签

**ng-init**

<div class="row" ng-init='getFeedbackList()'>

**ng-change**

<label><span>开始时间：</span><br class="visible-xs-inline-block visible-sm-inline-block"><input ng-change="getFeedbackList(startDate, endDate)" type="date" ng-model="startDate" style="line-height: 1em;padding: .5em 0"></label>

**ng-repeat**

<tr ng-repeat="item in items" icheck-directive>

**ng-click**

<button type="button" class="btn btn-default btn-xs js-reply" ng-click="reply(item.id,item.content,item.replyList,item.username)"><i class="fa fa-edit"></i>&nbsp;

回复

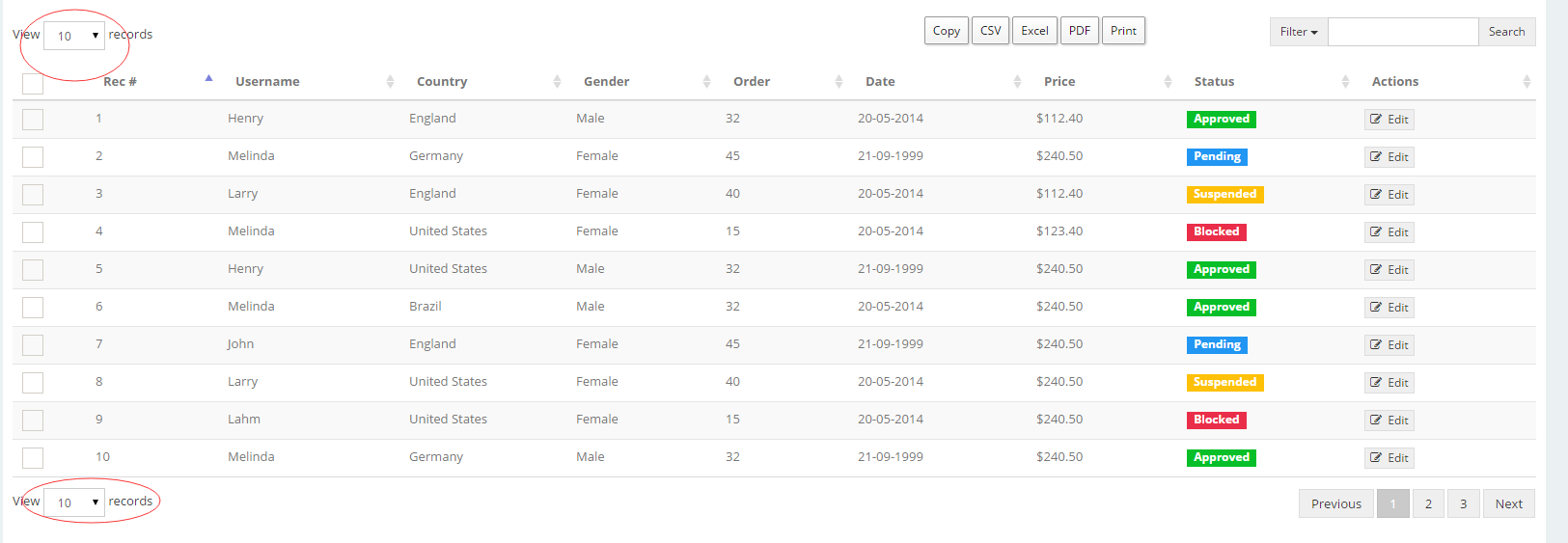
</button>

**ng-include**

<div ng-include="'/views/templates/page.html'"></div>

jquery.dataTables.js文件

\_fnFeatureHtmlLength 方法



# jquery

## jQuery 语法

jQuery 语法是为 HTML 元素的选取编制的，可以对元素执行某些操作。

基础语法是：$(selector).action()

美元符号定义 jQuery

选择符（selector）“查询”和“查找” HTML 元素

jQuery 的 action() 执行对元素的操作

示例

$(this).hide() - 隐藏当前元素

$("p").hide() - 隐藏所有段落

$(".test").hide() - 隐藏所有 class="test" 的所有元素（注意selector中不能有空格）

$("#test").hide() - 隐藏所有 id="test" 的元素

提示：jQuery 使用的语法是 XPath 与 CSS 选择器语法的组合。

**文档就绪函数：**

您也许已经注意到在我们的实例中的所有 jQuery 函数位于一个 document ready 函数中：

$(document).ready(function(){

--- jQuery functions go here ----

});

这是为了防止文档在完全加载（就绪）之前运行 jQuery 代码。

如果在文档没有完全加载之前就运行函数，操作可能失败。

## [jQuery取得select选中的值](http://www.cnblogs.com/greatverve/archive/2010/02/03/1662565.html)

获取选中的值：$("#select1").val();

获取选中的文本：$("#select1  option:selected").text();

## jQuery更改超链接的href

$(document).ready(function(){

$(".js-btn").click(function(){

var hospitalId = $(".js-hospital").val();

if(hospitalId == "" ){

alert("请选择医院");

return false;

}

this.href = this.href + "?hospitalId=" + hospitalId;

});

});

提交超链接是HTML本身就会自己完成的事情，JavaScript只是在提交之前更改一些属性的值，如href的值。另外，JavaScript也可以通过return false 来中断提交。

## jquery获取li列表中当前点击的元素下的各种值

**列表的HTML代码如下：**

<ul>

<li>

<a href="#1" class="case-info J\_CaseInfo js-meeting" >

<div class="expert">

<div class="date J\_Date">

<p>2015-01-26</p>

<p>星期一</p>

<input type="hidden" name="deptId" value="2001">

<input type="hidden" name="doctorId" value="5001">

</div>

<i class="icon pm">aa</i>

</div>

</a>

</li>

<li>

<a href="#2" class="case-info J\_CaseInfo js-meeting" deptid="aaa">

<div class="expert">

<div class="date J\_Date">

<p>2015-01-27</p>

<p>星期二</p>

<input type="hidden" name="deptId" value="2002">

<input type="hidden" name="doctorId" value="5002">

</div>

<i class="icon pm">bb</i>

</div>

</a>

</li>

</ul>

**jquery获取各个元素的代码示例如下：**

<script type="text/javascript">

$(document).ready(function(){

$(".js-meeting").click(function(){

alert($(this).attr('href')); //#1 (获取某个属性的值)

alert(this.href); //http://localhost:8080/opc/hospital/hospitalschedule?hospitalId=2000001#1

alert($(this).html()); //返回html文档： <div class="expert">…… </div>

alert($(this).find('div i').text()); //aa

alert($(this).find('div div p').text()); //2015-01-26星期一

alert($(this).find('div div p').eq(0).text()); //2015-01-26

alert($(this).find('div div p').eq(1).text()); //星期一

alert($(this).find("input[name='doctorId']").val()); //5001

});

});

</script>



## jquery与动态弹出层

## jquery与Json

以下例子演示jquery如何处理服务端返回的Json数据：

$(document).ready(function(){

$(".js-meeting").click(function(){

var hisPlanId = $(this).find("div div input").val();

var url = "${urlPath}" + "/opc/resource/getResourceByPlanId?hospitalId=${hospitalId}&hisPlanId=" + hisPlanId;

$.get(url, function(data, status){

//接收到的数据转化为JQuery对象，由JQuery为我们处理

var jsonData = eval(data);

if(jsonData.success == true){

$("tbody").html("<tr><th width=\"40%\">时间段</th><th width=\"20%\">就诊序号</th><th width=\"20%\">剩余号源</th><th width=\"20%\">操作</th></tr>");

//遍历对象数组，index是数组的索引号，objVal是遍历的一个对象。

$.each(jsonData.obj, function(index, objVal) {

//在table上追加一行

$("tbody").append("<tr><th height=\"40px\">"+objVal.startTime+"</th><th>"+objVal.orderNum+"</th><th>"+objVal.leaveNums+"</th><th><b>"+state+"</b></th></tr>");

});

}

});

});

});

public class Json {

private boolean success = false;

private String msg = "";

private Object obj = null; //实体列表，如 List<ResourceDto>

public boolean isSuccess() {

return success;

}

public void setSuccess(boolean success) {

this.success = success;

}

public String getMsg() {

return msg;

}

public void setMsg(String msg) {

this.msg = msg;

}

public Object getObj() {

return obj;

}

public void setObj(Object obj) {

this.obj = obj;

}

}

服务端的接口代码如下：

@RequestMapping("/getResourceByPlanId")

@ResponseBody

@Description("按条件查询排班")

public Json getResourceByPlanId(Long hospitalId,String hisPlanId){

Json j = new Json();

j.setSuccess(false);

try{

List<ResourceDto> resourceDtoList = infoService.getPlanResource(0, hospitalId, hisPlanId);

j.setSuccess(true);

j.setObj(resourceDtoList);

return j;

}

catch(Exception e){

e.printStackTrace();

}

return j;

}

## $.ajax()方法详解

虽然get()和post()函数非常简洁易用，但是对于更复杂的一些设计需求还是无法实现，比如在ajax发送的不同时段做出不同的动作等。jQuery提供一个更为具体的函数：ajax()。

jquery中的ajax方法参数总是记不住，这里记录一下。

**1.url**:   
要求为String类型的参数，（默认为当前页地址）发送请求的地址。

**2.type**:   
要求为String类型的参数，请求方式（post或get）默认为get。注意其他http请求方法，例如put和delete也可以使用，但仅部分浏览器支持。

**3.timeout**:   
要求为Number类型的参数，设置请求超时时间（毫秒）。此设置将覆盖$.ajaxSetup()方法的全局设置。

**4.async**:   
要求为Boolean类型的参数，默认设置为true，所有请求均为异步请求。如果需要发送同步请求，请将此选项设置为false。注意，同步请求将锁住浏览器，用户其他操作必须等待请求完成才可以执行。

**5.cache**:   
要求为Boolean类型的参数，默认为true（当dataType为script时，默认为false），设置为false将不会从浏览器缓存中加载请求信息。

**6.data**:   
要求为Object或String类型的参数，发送到服务器的数据。如果已经不是字符串，将自动转换为字符串格式。get请求中将附加在url后。防止这种自动转换，可以查看　　processData选项。对象必须为key/value格式，例如{foo1:"bar1",foo2:"bar2"}转换为&foo1=bar1&foo2=bar2。如果是数组，JQuery将自动为不同值对应同一个名称。例如{foo:["bar1","bar2"]}转换为&foo=bar1&foo=bar2。

**7.dataType**:   
要求为String类型的参数，预期服务器返回的数据类型。如果不指定，JQuery将自动根据http包mime信息返回responseXML或responseText，并作为回调函数参数传递。可用的类型如下：  
xml：返回XML文档，可用JQuery处理。  
html：返回纯文本HTML信息；包含的script标签会在插入DOM时执行。  
script：返回纯文本JavaScript代码。不会自动缓存结果。除非设置了cache参数。注意在远程请求时（不在同一个域下），所有post请求都将转为get请求。  
json：返回JSON数据。  
jsonp：JSONP格式。使用SONP形式调用函数时，例如myurl?callback=?，JQuery将自动替换后一个“?”为正确的函数名，以执行回调函数。  
text：返回纯文本字符串。

**8.beforeSend**：  
要求为Function类型的参数，发送请求前可以修改XMLHttpRequest对象的函数，例如添加自定义HTTP头。在beforeSend中如果返回false可以取消本次ajax请求。XMLHttpRequest对象是惟一的参数。  
            function(XMLHttpRequest){  
               this;   //调用本次ajax请求时传递的options参数  
            }  
**9.complete**：  
要求为Function类型的参数，请求完成后调用的回调函数（请求成功或失败时均调用）。参数：XMLHttpRequest对象和一个描述成功请求类型的字符串。  
          function(XMLHttpRequest, textStatus){  
             this;    //调用本次ajax请求时传递的options参数  
          }

**10.success**：要求为Function类型的参数，请求成功后调用的回调函数，有两个参数。  
         (1)由服务器返回，并根据dataType参数进行处理后的数据。  
         (2)描述状态的字符串。  
         function(data, textStatus){  
            //data可能是xmlDoc、jsonObj、html、text等等  
            this;  //调用本次ajax请求时传递的options参数  
         }

**11.error**:  
要求为Function类型的参数，请求失败时被调用的函数。该函数有3个参数，即XMLHttpRequest对象、错误信息、捕获的错误对象(可选)。ajax事件函数如下：  
       function(XMLHttpRequest, textStatus, errorThrown){  
          //通常情况下textStatus和errorThrown只有其中一个包含信息  
          this;   //调用本次ajax请求时传递的options参数  
       }

**12.contentType**：  
要求为String类型的参数，当发送信息至服务器时，内容编码类型默认为"application/x-www-form-urlencoded"。该默认值适合大多数应用场合。

**13.dataFilter**：  
要求为Function类型的参数，给Ajax返回的原始数据进行预处理的函数。提供data和type两个参数。data是Ajax返回的原始数据，type是调用jQuery.ajax时提供的dataType参数。函数返回的值将由jQuery进一步处理。  
            function(data, type){  
                //返回处理后的数据  
                return data;  
            }

**14.dataFilter**：  
要求为Function类型的参数，给Ajax返回的原始数据进行预处理的函数。提供data和type两个参数。data是Ajax返回的原始数据，type是调用jQuery.ajax时提供的dataType参数。函数返回的值将由jQuery进一步处理。  
            function(data, type){  
                //返回处理后的数据  
                return data;  
            }

**15.global**：  
要求为Boolean类型的参数，默认为true。表示是否触发全局ajax事件。设置为false将不会触发全局ajax事件，ajaxStart或ajaxStop可用于控制各种ajax事件。

**16.ifModified**：  
要求为Boolean类型的参数，默认为false。仅在服务器数据改变时获取新数据。服务器数据改变判断的依据是Last-Modified头信息。默认值是false，即忽略头信息。

**17.jsonp**：  
要求为String类型的参数，在一个jsonp请求中重写回调函数的名字。该值用来替代在"callback=?"这种GET或POST请求中URL参数里的"callback"部分，例如{jsonp:'onJsonPLoad'}会导致将"onJsonPLoad=?"传给服务器。

**18.username**：  
要求为String类型的参数，用于响应HTTP访问认证请求的用户名。

**19.password**：  
要求为String类型的参数，用于响应HTTP访问认证请求的密码。

**20.processData**：  
要求为Boolean类型的参数，默认为true。默认情况下，发送的数据将被转换为对象（从技术角度来讲并非字符串）以配合默认内容类型"application/x-www-form-urlencoded"。如果要发送DOM树信息或者其他不希望转换的信息，请设置为false。

**21.scriptCharset**：  
要求为String类型的参数，只有当请求时dataType为"jsonp"或者"script"，并且type是GET时才会用于强制修改字符集(charset)。通常在本地和远程的内容编码不同时使用。

案例代码：

[复制代码](javascript:void(0);)

$(function(){

$('#send').click(function(){

$.ajax({

type: "GET",

url: "test.json",

data: {username:$("#username").val(), content:$("#content").val()},

dataType: "json",

success: function(data){

$('#resText').empty(); //清空resText里面的所有内容

var html = '';

$.each(data, function(commentIndex, comment){

html += '<div class="comment"><h6>' + comment['username']

+ ':</h6><p class="para"' + comment['content']

+ '</p></div>';

});

$('#resText').html(html);

}

});

});

});

[复制代码](javascript:void(0);)

**22.顺便说一下$.each()函数**:  
$.each()函数不同于JQuery对象的each()方法，它是一个全局函数，不操作JQuery对象，而是以一个数组或者对象作为第1个参数，以一个回调函数作为第2个参数。回调函数拥有两个参数：第1个为对象的成员或数组的索引，第2个为对应变量或内容。

另外可以参考：http://www.w3school.com.cn/jquery/ajax\_ajax.asp

# note.js

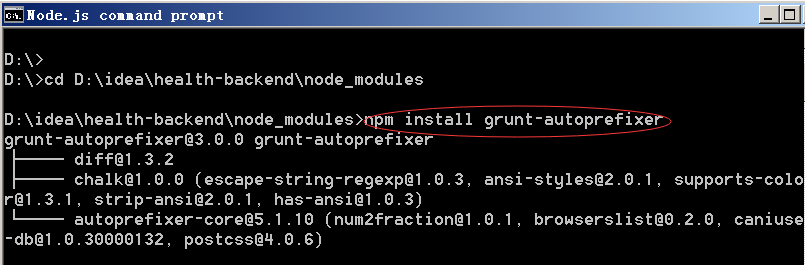
## 安装node.js模块

1、运行node.js的命令行工具

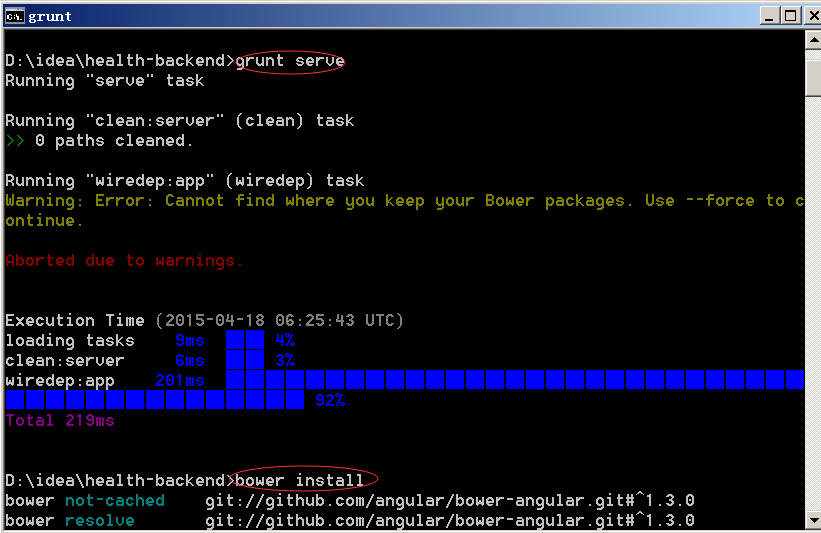
2、进入到需要安装模块的文件夹下

3、执行命令 “npm install 模块名称”进行安装

具体操作如下图：



## 安装bower



如果执行bower install 过程中，提示与git相关的错误，可能是系统环境变量 path 中没有添加git。在Path变量中加入：C:\Program Files (x86)\Git\bin;C:\Program Files (x86)\Git\cmd