# 2017年9月

## 20

### POST请求

### [URL编码](http://www.cnblogs.com/season-huang/p/3439277.html)

**如果你需要编码整个URL，然后需要使用这个URL，那么用encodeURI。**

**当你需要编码URL中的参数的时候，那么encodeURIComponent是最好方法。**

### [JSON对象和JSON字符串](http://www.haorooms.com/post/js_jsons_h)及相互转化

|  |  |
| --- | --- |
| JSON对象: | var str2 = { "name": "haorooms", "sex": "man" }; |
| JSON字符串: | var str1 = '{ "name": "haorooms", "sex": "man" }'; |
| JSON字符串转换为JSON对象 | var obj = str.parseJSON(); //由JSON字符串转换为JSON对象  var obj = JSON.parse(str); //由JSON字符串转换为JSON对象 |
| 将JSON对象转化为JSON字符串 | var last=obj.toJSONString(); //将JSON对象转化为JSON字符  var last=JSON.stringify(obj); //将JSON对象转化为JSON字符 |

### [跨域](https://itoss.me/2016/12/31/%E8%AF%B4%E8%AF%B4%E8%B7%A8%E5%9F%9F%E9%82%A3%E4%BA%9B%E4%BA%8B%E5%84%BF/)

## 22

### Callback理解

|  |  |
| --- | --- |
| 简单理解： | 把方法callback传给函数foo，在foo执行完以后再执行callback。 |
| 示例 | |
| **var** *foo* = **function** (callback) {  var a = “This is foo”;  alert(“I wrote var a!”);//step 1: I wrote var a!  callback(a); }; **function** *init*() {  //此处foo中的参数是一个函数，即callback,这里的word为foo传过来的a  *foo*(**function** (word) {  alert(a); //step 2: This is foo  }  } | |

## 25

### Window.top

遇到具体问题时再详细学习。

先把一个基本的[链接](http://www.cnblogs.com/haogj/archive/2012/03/28/2422485.html)放在这里。

## 26

### 伪类

### setTimeout()

|  |  |
| --- | --- |
| 语法 | 用于在指定的毫秒数后调用函数或计算表达式。 |
| 参数 | 描述 |
| code | 必需。要调用的函数后要执行的 JavaScript 代码串。 |
| millisec | 必需。在执行代码前需等待的毫秒数。 |
| 示例 | setTimeout("alert('5 seconds!')",5000) |

### 雪碧图

## 27

### 发布补丁

27号晚上peacock补丁要发，中间遇到跨域和swan预览界面获取不到的问题。前者运维配置即可解决，后者是因为主干个分支修改代码不一致产生的问题，还好及时发现，改动不大。

从明天开始参与组件库的开发与维护。

## 28

今天的任务是解决前些天剩下的知识点，熟悉组件库。

不在状态，学习很少。

## 29

再次熟悉了一遍组件库的构建目录，项目中需要的文件怎么产生的。

### Gulp入门教程

这一篇就够了：[点击链接](http://www.cnblogs.com/2050/p/4198792.html)

### 自动安装gulpfile中所有依赖的模块（未成功使用）

在gulpfile.js中定义任务的时候，需要引用不同的模块，这些模块需要使用

*npm install --save-dev* 命令进行安装，非常的繁琐。可以使用一个名为gulpfile-install的模块，使用下面命令进行安装：

|  |
| --- |
| npm install -g gulpfile-install |

安装成功之后就可以在项目根目录（存放gulpfile.js的目录）下，直接执行 gulpfile-install 命令就可以自动根据引用的模块进行安装了，非常方便。

### 路由

# 2017年10月

## 09

### Web缓存

没有看

### 组件库button样式调整

已经解决。

### Jsoup

还没有看

### PEACOCK-831任务单user.id

## 10

### 需要注意的

公司为什么雇用你而不是别人到这个岗位？要的不是你的技术，而是你的想法。以往的经验中，经历过的东西转化为想法出来并且可以根据实际情况实现出来才是最重要的，**工具的使用有熟练和不熟练之分，但是能力必须在你身上独一无二的存在**。

## 17

### 界面开发流程相关

前后台开发的中间是接口数据的传递，就算后台接口暂时没有定义好，仍然可以使用swagger上面的接口定义开发前端代码。假数据也是可以根据这个造出来的。

## 18

### 前后端分离

0 前后端分离是趋势，但是也还存在问题（例如SEO，**搜索引擎难以识别**等），短时间内不可能取代不分离的  
1 主要区别是，数据和表现分离，只需要静态的html和动态的接口（例如jsp），数据在浏览器端实现动态加载  
2 理想情况是，先出文档（前后端都认可），**然后后端、前端都按照文档来**，一切以接口规定的为准  
3 跟端口没一毛钱关系，重点在于接口！**靠 API 来分离前后端**，解决前后端大团队、多版本、复杂功能协作的问题

## 20

### ng-if和ng-show

在编写页面的时候要注意使用两者的区别。今天遇到一个问题，要取<th>标签下的<i>标签，使用的是$(th i)，但是始终获取不到，最终查出原因是使用了ng-if，在DOM中没有<th><i>元素，取不到也是正常的，看一下网上给的解释：

**ngHide/ng-show指令显示或隐藏指定的HTML元素。元素的显示隐藏是根据元素上ng-hide的css样式添加删除实现的。**

**ngIf指令会根据指定的表达式返回的boolean类型值对该元素做添加到/移除出Dom树的操作。**

ng-show只是添加css样式实现显示隐藏功能的。目前来讲，使用ng-show会比较好。

## 23

### 两个页面共享数据

当两个页面共享数据的时候，如果数据不能再第二个页面及时更新就回出错。比如我遇到的一个自定义指令，里面绑定的值不能根据$scope中绑定的数据实时变换，这个时候angularjs的双向绑定就不能起作用了。所以在多页面共享数据的时候，一定要注意数据是否真的可以实现共享。

## 24

### 三读java编程思想

推荐作者读这本书的境界变化，学习任何一门技术都应该有这种守破离的境界。

### 界面编码设计不明确

有时候开发界面的时候UX没能及时更新到，这个时候可能仅仅与界面设计的人一起设计开发页面，手里的资源可能只有prd文档和任务单。其实这个时候没有UX也可以开始开发，只要有界面模型和UI标准，一个是框架，一个是细节。做出来的页面跟UX设计出来的相差不大，不会影响任务的推进。至于细节可以后面与UX设计者沟通。

### 代码提交备注

Web前端组里面约定**每次提交代码的时候要备注任务单号**。这样有利于查找每一行代码为什么做修改。后期排找bug也会容易很多。这是一个习惯问题，是我们组约定必须遵守的规范。

## 25

### 远程工作

昨天刚到家，测试就跟我讲有一个bug需要修复，于是我打开电脑准备连接远程发现公司电脑根本没有开teamviewer，这就有点耽误时间了，还找了在公司的几个人帮忙开机器。以后回家之前记得把teamviewer打开，至少有紧急问题可以立马解决。

### angular.foreach

一个比较通俗的解释：代码如下:

var objs =[{a:1},{a:2}];

angular.forEach(objs, function(data,index,array){

//data等价于array[index]

console.log(data.a+'='+array[index].a);

});

参数如下：

objs：需要遍历的集合

data:遍历时当前的数据

index:遍历时当前索引

array:需要遍历的集合，每次遍历时都会把objs原样的传一次。

也可以不用写后面两个参数:

var objs =[{a:1},{a:2}];

angular.forEach(objs, function(data){

console.log(data.a);

});

## 30

### 规范处理问题

这里分成两部分提及：**1、代码规范**：多个人编写代码，最好能够提前统一一下风格，哪部分的功能放在那里实现，数据格式怎么约定，函数怎么定义，这些都有一个规范框架之后，以后换个人来维护这段代码也能够很快上手知道哪部分的功能应该在哪里找。不过有个问题就是不能把框架定义的太死，如果要求所有代码都按照一个模子编写，很多人会有抵触情绪，不愿意用这代码框架，所以js、html等分开定义一个非常通用的框架是非常重要的；**2、流程规范**：处理问题一般流程是：收到问题、分析任务、解决问题、提交测试、修改bug、切分支、版本发布……，任何一个环节都可能会有各种各样你想不到的问题，比如你在开发前端一个模块，忽然跟你对接接口的人请假，几天来不了，这个时候怎么办？是赶紧催他解决还是往老大那边反应，亦或者找另一个人接受他的任务？这个时候有两个地方需要注意，一、不要着急，觉得这个问题解决不了了，只要往上反映，老大肯定是能够帮你处理的，二、后台跟不上？那我就有借口推迟任务单了！这种懒惰心理会导致你没有处理任务的欲望，想着先放一放也不会影响什么，反正不是我的问题，这种心理很要命，可能会有很多其他问题一并被你拖到后面，这是非常严重的！

# 2017年11月

## 03

**对行内**元素，需要注意如下

* 设置宽度width 无效。
* 设置高度height 无效，可以通过line-height来设置。
* 设置margin 只有左右margin有效，上下无效。
* 设置padding 只有左右padding有效，上下则无效。注意元素范围是增大了，但是对元素周围的内容是没影响的。

行内元素与块级元素有什么不同？

**区别一：**

块级：块级元素会独占一行，默认情况下宽度自动填满其父元素宽度

行内：行内元素不会独占一行，相邻的行内元素会排在同一行。其宽度随内容的变化而变化。

**区别二：**

块级：块级元素可以设置宽高

行内：行内元素不可以设置宽高

**区别三：**

块级：块级元素可以设置margin，padding

行内：行内元素水平方向的margin-left; margin-right; padding-left; padding-right;可以生效。但是竖直方向的margin-bottom; margin-top; padding-top; padding-bottom;却不能生效。

**区别四：**

块级：display:block;

行内：display:inline;

可以通过修改display属性来切换块级元素和行内元素

### Form表单

很久我都好奇为什么用form，布局什么的div、table完全可以啊。看到这句话才明白它的用途：**form标签是用来进行表单提交用的。**简单，知道就行。