目录

[1 HTML+CSS=13天 7](#_Toc517429075)

[2 JS基础5天 7](#_Toc517429076)

[3 JS加强8天 7](#_Toc517429077)

[4 DOM5天 7](#_Toc517429078)

[5 JS特效5天 7](#_Toc517429079)

[6 JQ3天 7](#_Toc517429080)

[7 创业班第二十二天（HTML5） 7](#_Toc517429081)

[8 创业班第二十三天（CSS3） 7](#_Toc517429082)

[9 创业班第二十四天（CSS3） 7](#_Toc517429083)

[10 创业班第二十五天（CSS3） 7](#_Toc517429084)

[11 创业班第二十六天（CSS3） 7](#_Toc517429085)

[11.1 WEB字体 8](#_Toc517429086)

[12 创业班第二十七天（HTML5） 8](#_Toc517429087)

[12.1 多媒体 8](#_Toc517429088)

[12.2 拖拽 9](#_Toc517429089)

[12.3 全屏 9](#_Toc517429090)

[12.4 web储存 9](#_Toc517429091)

[12.5 地理定位 10](#_Toc517429092)

[12.6 网络状态 11](#_Toc517429093)

[12.7 应用程序缓存 11](#_Toc517429094)

[13 创业班第二十八天（AJAX） 12](#_Toc517429095)

[14 创业班第二十九天（AJAX） 12](#_Toc517429096)

[14.1 PHP基础 12](#_Toc517429097)

[14.2 文件上传 13](#_Toc517429098)

[15 创业班第三十天（AJAX） 13](#_Toc517429099)

[15.1 ajax发送post请求 13](#_Toc517429100)

[15.2 XML 14](#_Toc517429101)

[15.3 json 14](#_Toc517429102)

[15.4 ajax五步封装函数 14](#_Toc517429103)

[16 创业班第三十一天（AJAX） 16](#_Toc517429104)

[17 创业班第三十二天（AJAX） 16](#_Toc517429105)

[17.1 知识点 16](#_Toc517429106)

[17.2 jQuery操作AJAX 16](#_Toc517429107)

[17.3 瀑布流 17](#_Toc517429108)

[17.4 同源以及跨域 18](#_Toc517429109)

[18 创业班第三十三天（移动web） 18](#_Toc517429110)

[18.1 viewport属性 18](#_Toc517429111)

[18.2 移动web样式 19](#_Toc517429112)

[19 创业班第三十四天（移动web） 19](#_Toc517429113)

[19.1 知识点 19](#_Toc517429114)

[19.2 LESS 19](#_Toc517429115)

[20 创业班第三十五天(移动web) 21](#_Toc517429116)

[20.1 知识点 21](#_Toc517429117)

[20.2 过渡结束事件 22](#_Toc517429118)

[20.3 触摸事件 22](#_Toc517429119)

[20.4 精灵图通用代码用法 22](#_Toc517429120)

[20.5 定时器函数 22](#_Toc517429121)

[20.6 排他思想函数 23](#_Toc517429122)

[21 创业班第三十六天(移动web) 24](#_Toc517429123)

[21.1 知识点 24](#_Toc517429124)

[21.2 响应式布局 24](#_Toc517429125)

[21.3 Bootstrap框架 25](#_Toc517429126)

[22 创业班第三十七天(移动web) 25](#_Toc517429127)

[22.1 知识点 25](#_Toc517429128)

[22.2 栅格系统 26](#_Toc517429129)

[22.3 :nth-child()选择器 26](#_Toc517429130)

[23 创业班第三十八天(移动web) 26](#_Toc517429131)

[23.1 知识点 26](#_Toc517429132)

[23.2 zepto.js 27](#_Toc517429133)

[24 创业班JS高级7天 28](#_Toc517429134)

[25 创业班第四十六天(canvas) 28](#_Toc517429135)

[25.1 canvas 28](#_Toc517429136)

[25.2 虚线 29](#_Toc517429137)

[25.3 forEach 29](#_Toc517429138)

[25.4 map 30](#_Toc517429139)

[26 创业班第四十七天(canvas) 30](#_Toc517429140)

[26.1.1 API 30](#_Toc517429141)

[26.2 画弧 31](#_Toc517429142)

[26.3 文字API 31](#_Toc517429143)

[26.3.1 描边文字 31](#_Toc517429144)

[26.3.2 填充文字 31](#_Toc517429145)

[27 创业班第四十八天(canvas) 31](#_Toc517429146)

[27.1 绘制图像.drawImage() 31](#_Toc517429147)

[27.2 状态的保存与回滚 32](#_Toc517429148)

[27.3 平移/旋转/缩放 32](#_Toc517429149)

[28 创业班第四十九天(canvas) 32](#_Toc517429150)

[28.1 复习 32](#_Toc517429151)

[28.2 复习 33](#_Toc517429152)

[29 创业班第五十天(框架封装) 33](#_Toc517429153)

[29.1 学习课程介绍 33](#_Toc517429154)

[29.2 jQuery 34](#_Toc517429155)

[29.3 传入jQuery不同参数的不同返回结果 36](#_Toc517429156)

[29.4 对传入的参数进行处理 36](#_Toc517429157)

[29.5 toString伪数组 36](#_Toc517429158)

[30 创业班第五十一天(框架封装) 37](#_Toc517429159)

[30.1 把入口函数融合到框架中 37](#_Toc517429160)

[30.2 传入函数 37](#_Toc517429161)

[30.3 jQuery主要方法 38](#_Toc517429162)

[31 创业班第五十二天(框架封装) 39](#_Toc517429163)

[31.1 实例方法和静态方法 39](#_Toc517429164)

[31.2 函数 40](#_Toc517429165)

[31.3 常量 40](#_Toc517429166)

[31.4 类 41](#_Toc517429167)

[31.5 object.prototype 41](#_Toc517429168)

[31.5.1 Object 41](#_Toc517429169)

[31.5.2 所有函数默认的原型 41](#_Toc517429170)

[31.5.3 Math 42](#_Toc517429171)

[31.5.4 prototype:显式原型 42](#_Toc517429172)

[31.5.5 \_\_proto\_\_:隐式原型 42](#_Toc517429173)

[31.6 原型链 42](#_Toc517429174)

[31.7 DOM操作 42](#_Toc517429175)

[32 创业班第五十三天(框架封装)9.06 44](#_Toc517429176)

[32.1 知识点 44](#_Toc517429177)

[32.2 链式编程 44](#_Toc517429178)

[32.3 append(),appendTo(),prepend(),prependTo() 44](#_Toc517429179)

[32.4 属性和样式操作 44](#_Toc517429180)

[33 创业班第五十四天(框架封装) 46](#_Toc517429181)

[33.1 事件模块 46](#_Toc517429182)

[34 创业班第五十六天(git) 46](#_Toc517429183)

[34.1 shell 46](#_Toc517429184)

[34.2 命令 46](#_Toc517429185)

[34.3 VI编辑器 48](#_Toc517429186)

[34.4 SSH 48](#_Toc517429187)

[34.5 GIT分布式版本控制 48](#_Toc517429188)

[35 创业班第五十七天(git) 49](#_Toc517429189)

[35.1 git分支 49](#_Toc517429190)

[35.2 GIT远程仓库 50](#_Toc517429191)

[35.3 github 50](#_Toc517429192)

[35.4 步骤 50](#_Toc517429193)

[36 创业班第五十八天(angularJS) 51](#_Toc517429194)

[36.1 MVC开发模式 51](#_Toc517429195)

[36.2 模块化 51](#_Toc517429196)

[36.3 指令: 52](#_Toc517429197)

[36.4 数据绑定 53](#_Toc517429198)

[37 创业班第五十九天(AngularJS) 53](#_Toc517429199)

[37.1 作用域 53](#_Toc517429200)

[37.2 过滤器 53](#_Toc517429201)

[37.3 自定义 54](#_Toc517429202)

[38 创业班第六十天(angularJS) 55](#_Toc517429203)

[38.1 复习 55](#_Toc517429204)

[38.2 jsonp:一种跨域的解决方案 55](#_Toc517429205)

[38.3 自定义服务 56](#_Toc517429206)

[38.4 模块加载 57](#_Toc517429207)

[38.5 config配置服务(配置块) 57](#_Toc517429208)

[38.6 运行块 58](#_Toc517429209)

[38.7 雅虎十三条 58](#_Toc517429210)

[38.8 路由 58](#_Toc517429211)

[38.8.1 单页面应用 59](#_Toc517429212)

[38.9 其他 60](#_Toc517429213)

[38.10 作业 60](#_Toc517429214)

[39 创业班第六十一天(Gulp) 60](#_Toc517429215)

[40 创业班第六十二天(补课) 62](#_Toc517429216)

[40.1 单页面 62](#_Toc517429217)

[40.2 移动端 62](#_Toc517429218)

[40.3 css预处理器 63](#_Toc517429219)

[40.4 git 63](#_Toc517429220)

[41 创业班第六十三天(nodeJS) 64](#_Toc517429221)

[41.1 JS文件过多问题 64](#_Toc517429222)

[41.2 如何进行模块化开发 64](#_Toc517429223)

[41.3 SeaJS 65](#_Toc517429224)

[42 创业班第六十四天(NodeJS) 65](#_Toc517429225)

[42.1 总结 66](#_Toc517429226)

[43 创业班第六十五天(NodeJS) 66](#_Toc517429227)

[43.1 知识点 66](#_Toc517429228)

[43.2 包 67](#_Toc517429229)

[44 创业班第六十六天(NodeJS) 69](#_Toc517429230)

[44.1 复习 69](#_Toc517429231)

[44.2 I/O 70](#_Toc517429232)

[45 创业班第六十七天（NodeJS） 72](#_Toc517429233)

[45.1 复习 72](#_Toc517429234)

[45.2 debug 74](#_Toc517429235)

[46 创业班第六十八天(NodeJS) 75](#_Toc517429236)

[46.1 网络编程 75](#_Toc517429237)

[46.2 跨域 76](#_Toc517429238)

[46.3 开发分工 76](#_Toc517429239)

[46.4 协议 76](#_Toc517429240)

[47 创业班第六十九天(angular) 77](#_Toc517429241)

[47.1 复习 77](#_Toc517429242)

[47.2 网站优化-🡪思想 77](#_Toc517429243)

[47.3 http模块 78](#_Toc517429244)

[48 创业班第七十天(NodeJS) 78](#_Toc517429245)

[48.1 复习 78](#_Toc517429246)

[48.1.1 第一天SeaJS 78](#_Toc517429247)

[48.1.2 第二天nodejs 80](#_Toc517429248)

[48.2 express 80](#_Toc517429249)

[49 易错点： 80](#_Toc517429250)

# HTML+CSS=13天

# JS基础5天

# JS加强8天

# DOM5天

# JS特效5天

# JQ3天

# 创业班第二十二天（HTML5）

# 创业班第二十三天（CSS3）

# 创业班第二十四天（CSS3）

# 创业班第二十五天（CSS3）

# 创业班第二十六天（CSS3）

# 创业班第二十七天（HTML5）

## web储存

1. document.cookie:传统储存方式，4k大小
   1. 解析复杂
   2. 存储数据量小
2. Storage存储
   1. window.sessionStorage（会话存储，容量5M）
      * 1. 生命周期为关闭浏览器窗口
        2. 在同一个窗口下数据可以共享
   2. window.localStorage（本地存储，容量20M）
      * 1. 永久生效，除非手动删除,清理垃圾，存储在硬盘上
        2. 可以多窗口共享
   3. 设置、读取方便
   4. 只能存储字符串，可以将对象json.stringify（），编码后储存
   5. 可能存储在浏览器内存和硬盘上
3. 方法：
   1. window.sessionStorage关闭窗口数据销毁
      * 1. window.sessionStorage.setItem（key，walue）设置储存内容

再次给同一个key赋值，会覆盖之前的数据。

* + - 1. window.sessionStorage.getItem（key）获取存储数据
      2. window.sessionStorage.removeItem（key）删除key
      3. window.sessionStorage.clear（）清除所有数据
  1. window.loaclStorage数据存储在硬盘上，永久存储
     + 1. window.localStorage.setItem（key，walue）设置储存内容

再次给同一个key赋值，会覆盖之前的数据。

* + - 1. window. localStorage.getItem（key）获取存储数据
      2. window. localStorage.removeItem（key）删除key
      3. window. localStorage.clear（）清除所有数据

1. 打开页自动获取数据：window.onload = function（）｛获取数据｝

## 地理定位

1. 获取用户地理信息
2. 可以开发基于用户位置的互联网应用
3. 基于位置服务**L**ocation **B**ase **S**ervice
4. 定位方式
   1. IP可以一般精确到城市，运算代价大
   2. GPS 非常精确
   3. BeiDoui（BDS）
   4. wifi信号：室内
   5. 手机信号
   6. 用户自定义数据（用户自己输入自己位置）
5. 浏览器会自动以最优化方式获取用户地理信息
6. navigator.geolocation.getCurrentPosition(successCallback,errorCallback)
7. var wd = **position**.coords.**latitude**; //纬度  
   var js = **position**.coords.**longitude**;//经度

## 网络状态

1. window.addEventListener（‘online’，function（）｛｝）用户网络链接时被调用（on和其他事件的on不同，这里代表打开）
2. window.addEventListener（‘offline’，function（）｛｝）用户网络断开的时候调用
   1. 拔掉网线或者禁用以太网
3. window.addEventListener（‘事件（不带on）’）

## 应用程序缓存

1. 离线应用
2. 只需创建（后缀.appcache）一个CACHE MANIFEST文件
   1. CACHE MANIFEST  
        
      #注释开头用#  
        
      CACHE:  
      #要缓存的文件  
        
      NETWORK  
      #指定必须联网才能缓存的文件  
        
      FALLBACK  
      #当前页面无法访问时，返回的一个页面
3. 调用
   1. <html manifest="hc.appcache">
4. 优势

# 创业班第二十八天（AJAX）

# 创业班第二十九天（AJAX）

## PHP基础

1. php中单双引号不同
   1. 双引号会将变量的值放进去
   2. 单引号就把变量名当作字符串输出
   3. 单双引号混用

Php按照utf-8编码：header("content-type:text/html;charset=utf-8");

1. 关系数组
   1. array(“key”=>”value”,…)
2. get没有数据大小的限制
3. post上传大小没有限制
4. 二维数组
   1. $oneArr = array(array(“”),array(“”)…);
   2. 输出数组：print\_r(数组名)
5. 不指定上传方式，默认是get
6. 异步对象
   1. 创建异步对象
   2. 设置请求的url等参数
   3. 发送请求
   4. 注册事件
   5. 在注册的事件中获取返回的内容并修改页面显示的内容

## 文件上传

1. enctype = multipart/form-data
2. file\_get\_contents(“路径”)：读取文件

# 创业班第三十天（AJAX）

## ajax发送post请求

1. post请求
   1. 创建异步对象
   2. 使用post请求
      * 1. ajax.open（‘post’，‘ajax\_post.php’）
   3. 使用post提交数据要添加http头
      * 1. ajax.setRequestHeader（‘content-type’，‘application/x-www-form-urlencoded’）
   4. 发送请求
      * 1. ajax.send（’ xxx.php?name=jack&psd=123’）；
   5. 注册事件
      * 1. ajax.onreadystatechange = functtion（）｛

if （ajax.readyState == 4 && ajax.status==200）{

}

｝

//创建异步对象  
var ajax = XMLHttpRequest();  
//发送报文  
ajax.open("get","demo.php");  
//相应报文  
ajax.send();  
//注册事件  
ajax.onreadystatechange = function(){  
 if (ajax.**readyState** == 4 && ajax.**status** == 200) {  
 **console**.log(ajax.**responseText**);  
 }  
}

## XML

1. 标签名字随意写
2. <?xml version=”1.0” encoding=”UTF-8”?>
   1. 必须写第一行
3. ajax获取xml
   1. 通过ajax.resonseXML获取返回的xml值

## json

1. 创建一个json对象
   1. var json = '{"name":"jack","sex":"nan"}'
2. 转换为对象
   1. var jsonObj = **JSON**.parse(**ajax**.**responseText**);

## ajax五步封装函数

// 将 get 跟post 封装到一起  
/\*  
 参数1:url  
 参数2:数据  
 参数3:请求的方法  
 参数4:数据成功获取以后 调用的方法  
 \*/  
function ajax\_tool(url,data,method,success) {  
 // 异步对象  
 var ajax = new XMLHttpRequest();  
  
 // get 跟post 需要分别写不同的代码  
 if (method=='get') {  
 // get请求  
 if (data) {  
 // 如果有值  
 url+='?';  
 url+=data;  
 }else{  
 }  
 // 设置 方法 以及 url  
 ajax.open(method,url);  
  
 // send即可  
 ajax.send();  
 }else{  
 // post请求  
 // post请求 url 是不需要改变  
 ajax.open(method,url);  
  
 // 需要设置请求报文  
 ajax.setRequestHeader("Content-type","application/x-www-form-urlencoded");  
  
 // 判断data send发送数据  
 if (data) {  
 // 如果有值 从send发送  
 ajax.send(data);  
 }else{  
 // 木有值 直接发送即可  
 ajax.send();  
 }  
 }  
  
 // 注册事件  
 ajax.onreadystatechange = function () {  
 // 在事件中 获取数据 并修改界面显示  
 if (ajax.**readyState**==4&&ajax.**status**==200) {  
 // console.log(ajax.responseText);  
  
 // 将 数据 让 外面可以使用  
 // return ajax.responseText;  
  
 // 当 onreadystatechange 调用时 说明 数据回来了  
 // ajax.responseText;  
  
 // 如果说 外面可以传入一个 function 作为参数 success  
 success(ajax.**responseText**);  
 }  
 }  
  
}

# 创业班第三十一天（AJAX）

# 创业班第三十二天（AJAX）

## 知识点

1. array\_rand(哪个arr,取几个1)随机的index
2. $arr = json\_decode($str) 将字符串转换为php的数组
   1. json\_encode将object转换为json格式字符串-
3. json格式是字符串
4. print\_t（数组）；php返回一个数组
5. PHP获取数组长度：count（数组）；
6. in\_array(要查询的数组,查询内容)；查询
7. PHP定义数组
   1. $userArr = array(‘值1’， ‘值2’，);
8. 包装对象

var obj = ｛

items：json数组

｝

1. jq中有 数组.each（function（）｛｝）循环方法

## jQuery操作AJAX

1. $.get（url，{可直接写对象name：‘jack’，age：‘18’}，回调函数，参数4（从服务器获取的数据类型，可不写））
   1. $.get(url,date,callback,type)
   2. 如果参数4写json内部会JSON.parse（）自动转换
   3. 不要擅自更换参数位置
2. $.post(url,date,function(date){},type)

$\_ajax({

url:‘’，

success：function（date）{},

type:‘post/get’，

datatype：‘数据类型（json）’，

data：｛

‘name’：‘’，

‘kill’：‘’

｝，

beforeSend：function（）｛

console.log（‘发送之前调用’）；

return false；//阻止这次请求

验证用户数据是否填了

｝，

error：function（）｛

console。log（‘请求失败后调用’）

｝

})

## 瀑布流

1. 实现技术
   1. ajax
   2. jq
      * 1. ajax请求
   3. 模版引擎
      * 1. 渲染页面
2. 为jQuery添加插件
   1. $.fn.extenf();
   2. 注册完毕后，使用$(‘调用者’).函数名 调用

$.fn.extend({

函数名：function(){

函数体；

}

var $\_this = this;

//为了可以链式编程

return $\_this;

})

* 1. $.extenf();
  2. 注册完毕后实用$.xxx

## 同源以及跨域

1. 同源：域名、协议、端口完全相同
2. iframe嵌入另外一个页面

# 创业班第三十三天（移动web）

## viewport属性

<meta

name="viewport"

content="

width=device-width, //设备的宽度

user-scalable=no, //设置用户缩放，一般禁止缩放no/false

initial-scale=1.0, //初始的默认缩放比例，一般默认是指1.0

maximum-scale=1.0,//最大的缩放值

minimum-scale=1.0"//最小的缩放值

/>

1. name = ‘viewport’；告诉浏览器，这个meta标签设置的是viewport属性

## 移动web样式

1. -webkit-tab-highlight-color：transparent；鼠标点击高亮
2. 移动端设置所有的盒子box-sizing：border-box；保证盒子大小不变
3. input默认样式清楚
   1. -webkit-appearance：none；

# 创业班第三十四天（移动web）

## 知识点

1. li标签不可以用ling-height居中，要用父盒子的padding挤
2. h3:first-of-type:找到第一个这种类型的标签
3. h3:last-of-type:找到最后一个这种类型的标签
4. :nth-of-type(n):找到第几个该类型的标签

## LESS

1. @变量名:属性值;定义变量
2. 如果less文件在名为less文件夹里,考拉编译后会自动生成一个css文件夹保存.css文件,否则会在一个文件夹生成.css文件
3. less可以更好的维护和编写css
4. 混合:可以把一个定义好的class完全引入到另一个class中
   1. .类名(@color:默认值){  
       //样式  
       background-color: @color;  
       border: 1px solid @color;  
       color:@color;  
      }  
      选择器{  
       .类名(@color值);  
      }
5. 嵌套
   1. div{  
       width: 100%;  
       ul{  
       width:100%;  
       li{  
       width: 100%;  
       float: left;  
       a{  
       color:red;  
       }  
       .cla{  
       color:green;  
       }  
       }  
       }  
       .list{  
       width:50%;  
       }  
      }
   2. 相当于
   3. div {  
       width: 100%;  
      }  
      div ul {  
       width: 100%;  
      }  
      div ul li {  
       width: 100%;  
       float: left;  
      }  
      div ul li a {  
       color: red;  
      }  
      div ul li .cla {  
       color: green;  
      }  
      div .list {  
       width: 50%;  
      }
6. 伪元素
   1. 交集选择器要加&.current
   2. 如果不加,直接写.current会变成并集选择器
   3. 伪元素也要**&::hover**

# 创业班第三十五天(移动web)

## 知识点

1. opacity:会使子元素也变透明,rgba则不会
2. offsetTop:距离顶部高度
3. offsetHeight:自身的高度
4. onscroll:页面滚动事件
5. windo.document.body.scrollTop;滚动距离
6. .querySelectot/.querySelectorAll(“可以写从css,css3中的选择器”)
7. 移动端一定要加视口属性:meta:vp
8. 时分秒
   1. **hour** = **Math**.floor(totalSecond/3600);  
      **minute** = **Math**.floor((totalSecond%3600)/60);  
      **scon** = totalSecond%3600
9. span.[^icon\_]{}

## 过渡结束事件

ul.addEventListener("webkitTransitionEnd", function () {  
 **console**.log("过渡结束");

}

要加私有化前缀:webkit/o/moz/ms(ie)

## 触摸事件

1. 触摸事件
   1. **touchstart**:开始触摸的时候触发,有触摸点的值
      1. var x = event.touches[***0***].*clientX/Y*;
   2. **touchmove:**手指按下并移动的时候, 有移动的值
      1. var x = event.touches[***0***].*clientX/Y*;
   3. **touchent:**手指抬起来的时候,没有触摸点的值
2. 不能通过dom.ontouchstart的方式绑定
3. 只能通过addEventList(‘touchstart’,function(){})的方式绑定事件
4. 只支持移动设备
5. 封装事件
   1. 左滑,右滑
   2. 捏合
   3. 长按

## 精灵图通用代码用法

[class^="icon\_"]{  
 background: url("../images/sprites.png") no-repeat;  
 background-size: x y ;  
}

## 定时器函数

//定时器函数  
function cutDownTime(){  
 //获取事件源  
 var time = **document**.querySelectorAll('.product:nth-child(1) .title li');  
 var totalSec = 7200;  
 //定义定时器  
 var tim = setInterval(function () {  
 //如果剩余时间为0,停止执行函数  
 if(totalSec<=0){  
 totalSec = 0;  
 return;  
 }  
 totalSec--;  
 //转换时分秒  
 var hour = **Math**.floor(totalSec/3600);  
 var minute = **Math**.floor((totalSec%3600)/60);  
 var sec = totalSec%60;  
  
 //显示到页面中  
 //十位部分显示  
 time[0].**innerHTML** = **Math**.floor(hour/10);  
 //个位部分显示  
 time[1].**innerHTML** = hour%10;  
  
 //十位部分显示  
 time[3].**innerHTML** = **Math**.floor(minute/10);  
 //个位部分显示  
 time[4].**innerHTML** = minute%10;  
  
 //十位部分显示  
 time[6].**innerHTML** = **Math**.floor(sec/10);  
 //个位部分显示  
 time[7].**innerHTML** = sec%10;  
 },1000)  
}

## 排他思想函数

//排他思想函数,干掉所有人,留下我自己  
function exclusive(elementArr){  
 for(var j = 0;j<elementArr.**length**;j++){  
 //循环为每一项绑定点击事件  
 elementArr[j].onclick = function () {  
 //把每一项的类名都去掉  
 for(var i = 0;i<elementArr.**length**;i++){  
 elementArr[i].**className** = '';  
 }  
 //只留下触发当前点击事件的触发者,加上类名  
 this.**className** = 'current';  
 }  
 }  
}

# 创业班第三十六天(移动web)

## 知识点

1. 定时器的时间和过渡的时间相等或者小于过渡的时间,会出问题
   1. 相等:会一直在滑动状态,触发不了过渡结束事件
   2. 小于过渡事件,会跳, 触发不了过渡结束事件
2. 通用代码封装成函数
3. 确定不了的变量封装成函数
4. 代码中不要写死代码
5. click事件在移动端有延迟
6. 自己封装一个tap的方法,手指点击
7. 最牛逼的程序员看颈椎病康复大全
8. Date.now();,获取当前的系统时间,返回的的是毫秒值
9. console.log():十分耗性能,只会出现在测试的时候,项目上线的时候会删除

## 响应式布局

1. 基本概念
   1. 针对u所有设备
   2. 开发一次即可
   3. 会感染疟疾设备的屏幕尺寸改变而改变布局
2. 缺点
   1. 代码较多
   2. 维护复杂
3. 简单响应式布局
   1. 媒体查询

**@media** *screen and* (*min-width*:500*px*){

body{  
 *background-color*: *#f00*;  
 }  
}  
**@media** *screen and* (*max-width*:700*px*){  
 body{  
 *background-color*: *#ff0*;  
 }  
}

## Bootstrap框架

1. 栅格系统
   1. container
      1. row
   2. 默认一行为12行
2. Bootstrap

# 创业班第三十七天(移动web)

## 知识点

1. 响应式布局依托于媒体查询**@media**
2. 伪元素一个冒号和两个冒号都可以
   1. 一个是老版本的写法
   2. 两个是新版本的写法
3. 父元素设置text-align:center;子元素居中显示
4. **lorem**1000:自动生成文本
5. 不要修改bootstrap的源文件
   1. 拷贝bootstrap的原生的文件
   2. 修改拷贝的原生样式的类名
   3. 修改css
   4. 实用组件的地方替换自己的类名
   5. 不会覆盖元素的css
6. bootatrap的a便签有一个默认样式:
   1. 有内边距
   2. display:block;
7. a标签不支持嵌套
8. 移动web

## 栅格系统

1. bootstrap的栅格系统必须写在**.container**中
2. .container-fluid;自适应100%宽度

# 创业班第三十八天(移动web)

## 知识点

1. bootstrap层次结构复杂,修改样式的时候拷贝css源代码.然后修改根选择器类名
2. hidden-lg\md\ms\xs;在相应的图片尺寸内隐藏
3. bootstrap的js插件里的**标签页插件**里的点击切换和层次关系无关,给标签和内容各加一个父盒子不影响效果
   1. 标签和内容是通过锚点来关联的;href=’#name’;id=’name’
4. 先解析和后解析会造成页面布局不同
5. 给div加box-sizing:border-box;之后,如果内容大小超过固定大小之后还是会撑大父盒子的
6. div:nth-of-type(2n);从第一个div开始计数
   1. 2n:选择索引为2的倍数的标签
   2. n+2:选择索引值大于等于2的标签
   3. 2n+1:从0开始计算(2\*0+1=0),索引为2n+1的标签,会选择到第一个标签
   4. **2n-1:不可以**
7. 媒体查询
   1. **@media** screen and (max-width:1200px){  
       li{  
       width:25px;  
       }  
      }
   2. 小于1200px的时候执行该样式
8. 列表居中方式
   1. 给ul设置display:inline-block;
   2. 给ul的父盒子设置:text-align:center;
9. web

## zepto.js

1. 轻量级,jquery太大
2. 根据需求导入需要用的模块
3. js文件会自动缓存,不修改就不用再次请求
4. 必须先导入event.js事件处理模块
   1. 用$(‘.class’).on(‘click’,function(){})
5. event:事件处理
6. fx:动画
7. touch:触摸事件(单击,滑动,长按...)
8. -支持链式编程
9. jquery改变标签内的文本用.text(‘文本内容’);
10. $(‘div/#box/.box’);选择器
11. $(‘<p>新元素</p>’);创建一个新元素
12. $(‘<p/ >’,{text:’内容’,id:”p1”,css:{color:’red’}})
    1. => <p id=greeting style="color:darkblue">Hello</p>
    2. 创建带有属性的元素

# 创业班JS高级7天

# 创业班第四十六天(canvas)

## canvas

1. 这个图从哪来
   1. 展示绘图效果的标签,行内块元素inline-block
   2. 打开canvas,(通过getContext方法打开),然后在上面绘图
2. canvas绘图步骤
   1. **移动到开始位置**
      * 1. .moveTo(x,y)
   2. **开始画线条**
      * 1. .lineTo(x,y)
   3. **描边显示**
      1. .stroke()
3. 画布大小在canvas标签设置
   1. <canvas id = ‘cns’ width = 500 height = 300></canvas>
   2. 注意：画布的样式设置为行内样式。如果设置到外部无效果
4. **描边颜色**:.strokeStyle = ‘颜色’: 描边之前设置颜色
5. **描边宽度**:.lineWidth = 数值;
6. 解决锯齿:多画一条边(不推荐使用)
7. **闭合路径:**.closePath(),可以省去最后一条边的绘
8. 封装绘制矩形函数
9. **清除路径**.beginPath();为了放置重绘
10. **填充**:.fill() 不定义填充,定义填充颜色也不起作用
11. **填充颜色**.fillStyle = ‘颜色’
12. **如果先填充后描边,描边宽度正常**
13. **先描边,后填充,描边宽度会显示一半**
14. 非零环绕原则:判断哪些区域属于路径内
    1. 在路径包围区域中,找一点每次向外发射一条射线
    2. 和所有环绕他的边相交
    3. 开启一个计数器,从0计数
    4. 如果这个设下遇到顺时针环绕,+1
    5. 遇到逆时针环绕-1
    6. 计算结果非0,即在路径内
15. 先填充,后描边,才能显示设置的宽度
16. canvas在绘制线条的时候,会左右偏移线宽的一般
    1. 线宽是奇数,边缘的颜色会变浅一半
    2. 线宽是偶数,无影响
17. **线帽样式**:.lineCap = ‘
    1. butt(默认值)
    2. round:圆头,半径为线宽的一半
    3. square:方头(各突出线宽的一半)’
18. **交点样式**:.lineJoin = ‘
    1. miter:默认尖角
    2. round:圆角
    3. bevel:平角’

矩形

1. 画一条矩形路径:.rect(starX,starY,width,height);
   1. 只是路径,要在后面描边stroke();
2. 绘制一个描边矩形:strokeRect(starX,starY,width,height)
3. 绘制一个填充矩形:fillRect(starX,starY,width,height)
4. 清除画布.clearRect(starX,starY,width,height)

## 虚线

.setLineDash([实线长度,虚线长度])

传一个值表示实线虚线各占多少

设置值的个数为奇数时,系统存储的是设置值的个数\*2

1. .getLineDash()获取如何设置的线条规则
2. .lineDashOffset = 3 偏移量

数组的两个新方法

## forEach

var arr = 数组.forEach(function(value,index,arr){

把数组内的每一项执行一次函数

不会使用返回结果

}[,this])

## map

数组.map(function(value,index,arr){

把数组内的每一项执行一次函数

返回一个新数组

}[,this])

# 创业班第四十七天(canvas)

### API

1. getConText(‘2d’||’webgl’):获取一个绘图环境(工具)
   1. width,height:画布的宽高
2. .moveTo(x,y):获取起点
3. .lineTo(x,y):画一条路径线
4. .stroke():根据当前路径描边
5. .strokeStyle=’color’:描边颜色
6. .fill():填充
7. .fillStyle=’color’:填充颜色
8. .lineWidth=3:描边宽度
9. .closePath():闭合路径
10. .beginPath():清除路径
    1. 只是清除路径,不清除样式
11. .rect();画一个举行路径
12. .fillRest(x,y,width,height):填充一个矩形
13. .strokeRest(x,y,width,height):描边矩形
14. .clearRest(x,y,width,height):
15. .setLineDash([a,b,c,d]):设置虚线规则
16. .getLineDash():获取虚线规则
17. .lineDashOffset = 2;设置偏移量
18. .lineCap = ‘butt/square/round’
19. .lineJoin = ‘miter/bevel/round’

## 画弧

1. .arc(圆心X坐标,圆心Y坐标,半径,起点弧度,结束点弧度,[是否逆时针画])
2. arc会从上次的结束点到弧的开始点之间画条直线

## 文字API

### 描边文字

1. .strikeText(文字,参考X坐标,参考Y坐标[,限制文字最大长度])
2. .font = ‘加粗 字号/行高 字体’:设置字体样式
   1. 单独设置字号不可以,必须加字体样式才会生效
3. .在canvas中绘制文本，不会产生路径吗

### 填充文字

1. .fillText(文字,参考X坐标,参考Y坐标[,限制文字最大长度])
2. .textAlign = ‘left/right/start’对齐方式
3. .textBaseline = ‘top/middle/bottom/默认基线(alphabetic)’

# 创业班第四十八天(canvas)

## 绘制图像.drawImage()

1. .drawImage()
   1. 三种使用方式
      * 1. 三个参数
           1. .drawImage(获取标签,坐标X,坐标Y)
           2. img.onload = function(){}:img加载完毕之后触发
        2. 五个参数:把图片放置到指定位置,并指定宽高
           1. .drawImage(获取标签,坐标X,坐标Y,width,height)
        3. 九个参数:把裁剪后的图片放到指定的位置,并且知道大小
           1. .drawImage(获取标签,裁剪图片的位置X, 裁剪图片的位置Y, 裁剪图片的width, 裁剪图片的height,坐标X,坐标Y,width,height)

## 状态的保存与回滚

1. .save();状态保存
   1. 把当前的状态(绘制环境的所有属性)复制一份保存
2. .restora();状态回滚
   1. 把最近保存的一次状态作为当前状态
   2. 回滚一次会把上次保存的属性删除
      * 1. 再回滚会拿到上次保存的属性
   3. 可多次回滚,依次往回找保存的属性

## 平移/旋转/缩放

1. **平移.translate**(X平移的量,Y轴平移的量)
2. **旋转.rotate**(旋转的弧度)
3. **缩放.scale**(X轴缩放比值,Y轴缩放比值)
4. 以绘制的图像不会手打影响
5. 只会收到之后的图像
6. 角度转弧度:Math.PI / 180 \* deg (deg是要转换的角度)
7. 绘制一个旋转的图像
   1. 现平移坐标轴到图像的中心
   2. 旋转坐标轴
   3. 绘制图像
      * 1. 平移旋转之后,坐标体系变化
        2. 先保存最初的坐标系
        3. 绘制完之后在回滚

# 创业班第四十九天(canvas)

## 复习

1. 定时器的this指向window
2. 状态的保存与路径无关
3. 内部使用的方法一般用 \_ 开头
4. var s = Math.floor( Math.randow() \* (max-min+1) + min ); //生成min-max之间的随机数

## 复习

1. 原型链知识
   1. Array instanceof Arrat 返回false
2. ‘use strict’; 开启严格模式
3. 构造函数return非基本类型的数据,新对象就是该数据

1.fun():函数调用模式,this指向window

2.Obj.fun():方法调用模式,this指向调用fun的对象

3.new Fun():构造函数调用模式:this指向新创建的对象

4.fun.apply/call(this,[]):上下文调用模式:this指向参数设置的内容

# 创业班第五十天(框架封装)

## 学习课程介绍

1. JS原生框架
   1. 模仿jQuery搭建
2. 学习目标
   1. 复习jQuery
   2. 复习原生js方法
   3. 了解jquery的大概实现
   4. 锻炼编码能力
3. 第一天-框架结构
   1. jQuery对象本质
   2. 框架核心结构
   3. 入口函数(构造函数)
4. 第二天-核心方法
   1. 入口函数对函数的处理
   2. 原型上的核心方法
   3. map和each
5. 第三天-Dom操作
   1. 创建dom
   2. 删除dom
   3. 追加dom
6. 第四天-属性样式操作
   1. class属性操作
   2. 公告属性操作
   3. 样式操作
7. 第五天-事件
   1. 事件绑定
   2. 事件解绑
8. 第六天-最后
   1. Ajax
   2. 插件实现原理
   3. 折线图饼图做成框架插件
   4. 刮刮乐插件

## jQuery

1. jquery对外暴露了$和jQuery这两个方法
2. 调用这个方法,可以得到一个jQuery实例对象
3. jquery实例对象是一个伪数组对象
4. jquery和$是一个工厂函数
5. 只有浏览器有window对象
6. **原型简写**
   1. jquery.fn = juqery.prototype = function(){}
7. **jquery插件的实现机制**
   1. 就是给原型添加方法
   2. jquery.prototype.新插件 = function(){}
   3. 也就等于jquery.fn.新插件 = function(){}
8. 任何函数都可以是构造函数,new谁 谁就是构造函数
9. **传入null / undefined / 0 / ‘’ /NaN**
10. **传入的是选择器,会返回获取到的元素**
    1. 按照下标存储起来,同时添加一个length属性代表数量
11. **传入HTML片段,会根据这些片段创建DOM对象**
    1. 然后按照下标储存起来,同时添加一个length熟悉代表数量
    2. document.body.appendChild($(‘<span></sapn>’)[0])
    3. 返回的是一个伪数组
12. **传入一个数组或者(**伪数组**),会把数组中的每一项分别添加到实例身上,同时把length有也加过来**
    1. $([‘a’,’b’])
13. **传入对象,直接把对象按照下标添加到实例身上**
14. **传入基本数据类型,null,ndefined,字符串除外,**
    1. 直接把数据添加到实例身上
    2. 传字符串会被认定为是选择器
15. **传DOM对象,也会添加到实例身上**
16. **传入的是null,undefined,0,NaN,’’**
    1. null,undefined,0,NaN,’’ 转换为boolean类型都会是false
    2. 返回一个this
    3. 构造函数里的this指向新的实例对象
17. **传入的是html片段**
    1. html开头结尾都是<>,并且length>=3,就可认为是一个html片段
       * 1. 创建一个临时的div
         2. 设置这个div的innerHTML为html片段
         3. 这些片段就转变为了div的子元素
         4. 遍历div的子元素分别添加到this身上,记住给实例补充length属性
         5. 可以使用数组的push()来给实例添加,同时可以使用apply简化遍历过程
         6. **实现**
            1. **var tempbox = document.createElement(‘div’)**
            2. **tempbox.innerHTML = selector;**
            3. **[].push.apply(this,tempbox.childNodes)**
18. **传入的是选择器,按照选择器获取页面中的DOM,把DOM添加到实例身上**
    1. 使用querySelectorAll获取所有的元素
    2. 把遍历到的所有元素分别添加到实例身上
19. **传入伪数组和数组**
    1. **排除函数和window,因为函数和window都有length属性**
       * 1. 排除函数 typeof function
            1. if( typeof selector == ‘function’){ return false}
         2. window.length属性代表页面中iframe的数量
         3. window.window === window
         4. if(select.window === window)

{return false}

* + - 1. 如果一个数据的window属性是自己,那么就认为它是window
  1. **toString判断是不是真数组**
     + 1. new Array()创建的数组和[]字面量创建的数组
       2. 稀疏数组
          1. 某些数组下标不存在
  2. **判断是不是伪数组**
     + 1. 伪数组特点
          1. 必须有length属性
          2. 如果length属性值是0,那么这个对象没有任何元素,是空数组
          3. 如果length属性不为0,那么这个伪数组一定有(length-1)为下标是属性值
       2. **判断伪数组**

**必须有length属性**

**要么length为0,要么有length-1这个下标**

* + - 1. 判断过程

1. 传入的是选择器
2. charAt()获取数组指定下标元素
3. 借用其他对象方法
   1. 借用数组方法
      * 1. [].push.apply(this,)
        2. [].pop.apply(this,)
4. replace(/^\s+|\s+$/g,’’):替换首位空字符

## 传入jQuery不同参数的不同返回结果

1. 传入null,undefined,0,NaN,’’返回空对象空实例
2. 传入字符串,需要贩毒案是html片段还是其他
   1. 是片段,创建对象的DOM,然后添加到实例身上
   2. 否则按照选择器获取页面中的DOM,然后把获取到的DOM添加到实例身上
3. 如果是数组或者伪数组,那么把每一项分别添加到实例身上
4. 出了上面的数据类型,剩余的统一添加到实例身上

## toString伪数组

1. 根据内部的this 返回一个类似于这样的字符串
   1. [ object className]
   2. **Object.prototype.toString.call(this的指向)**
   3. Array Data都继承自Function
   4. new Array 是Array类型,Array是Function类型
   5. ({}) === Object.prototype
2. 不能获取用户自定义对象的类型
3. 只能获取内置对象的类型
4. 自定义对象类型都返回[object object]
5. {}:js默认识别为代码块
   1. 代码块就是一对{}
   2. 如果把一对大括号赋值给其他变量,或者参与运算,大括号就变成了自面对新
   3. ():是一个运算符

# 创业班第五十一天(框架封装)

## 把入口函数融合到框架中

1. Function 的length属性是形参的个数
2. window的length属性是ifrim的数量
3. 判断是不是window
   1. w.window === w
4. !!str 把str转换为boolean类型值
5. **静态方法**是加给实例
6. **实例方法**是加给原型
7. 构造函数不能使用自己原型中的方法(Function例外)
8. 实例不能直接使用构造函数身上的静态方法
9. apply可以改变this指向,同时可以把数组或者伪数组怕你平铺传入函数
   1. IE8,apply只能平铺真数组,或者内置伪数组,自定义的伪数组会报错
   2. 需要把自定义伪数组转换为真数组,再使用apply方法
   3. [].push.apply(this,[].slice.call(伪数组))
      * 1. slice()不传参数整体截取

## 传入函数

1. 会在页面解析完成后执行
2. DOMContentLoaded比onload快
3. 如果document.readyState === “complete” 说明dom已经构建完毕
4. 否则判断docment.addEventListeer

如果支持document.addEventListeven(‘DOMContentLoaded’,fn)

否则document.attachEvent(‘onreadyststechange’,function()){

if(document.readyState === ‘compalete’)

fn()

* 1. }

## jQuery主要方法

1. jquery:获取版本号
2. selector
3. length
4. toArray:转换为数组,获取的是原生DOM
   1. return [].slice.call(this)
5. get:获取元素
   1. 不写参数,获取所有
   2. -1:获取最后一个
   3. 传入null或者undefined,转换为数组返回
   4. 如果传入的是证书,获取指定下标的元素返回
   5. 传入的是负数,按照下标倒着(a.length +负数 )获取元素
   6. **实现**
      1. if (*i* == null) /\*null和undefined一样\*/  
         {  
          return this.toArray();  
         }  
         if (*i* > *0*) {  
          return this[*i*]  
         }  
         else {  
          return this[this.*length* - *1*]  
         }
6. **each**
   1. 数组的forEach先获取value再获取index
   2. jQ的each先获取index后获取value
7. **map**
8. slice:
   1. 截取部分实例,构成一个新的jQuery实例返回
   2. 通过数组的slice截取部分实例(slice返回的是数组)
   3. 把截取到的元素按照实例返回
   4. **实现**
      1. var nodes = [].slice.*apply*(this, arguments)  
         return jQuery(nodes)//包装成实例对象
9. first():得到第一个jQuery实例
10. last():得到最后一个jQuery实例
11. eq():
    1. 传入null和undefined,返回一个新的实例
    2. 不穿参数,获取一个新实例
    3. 传入正数,按照指定下标返回元素,再包装成新实例返回
    4. 传参按照指定下标获取元素,获取到的元素是新的jQuery实例
12. push():给实力添加一个新元素
13. sort():给实例排序
    1. .sort(function(a,b){return a.innerHTML>b.innerHTML })
14. splice(开始下标,删除几个,要替换的元素)

# 创业班第五十二天(框架封装)

## 实例方法和静态方法

1. 通过构造函数创建出来的对象,就叫实例
2. 实例和对象的关系
   1. 万物皆对象
   2. 实例就是对象
3. **实例成员**
   1. 提供给实例使用的方法和属性
4. **实例方法**
   1. 提供给实例使用
   2. 构造函数原型中的成员也是实例方法
   3. 提供给所有实例对象使用
   4. **构造函数不能使用实例方法(定义在原型中)**
   5. **实例不能使用静态方法(构造函数.方法)**
5. **静态成员(类成员)**
   1. 提供给构造函数使用的方法和属性
6. **静态方法(类方法)**
   1. 提供给构造函数使用的方法
   2. 构造函数 点 出来的方法
   3. 不创建实例也可以使用
   4. 可以在复用
   5. 实例不能直接使用静态方法,通常会根据静态方法再实现一个实例方法,供实例使用
7. **实例复用静态方法**
   1. 静态方法服务面更广
   2. 实例方法只服务于实例
8. 变量名全大写代表常量
9. 4

## 函数

1. **普通函数**
   1. 直接定义
      * 1. function fn(){}
        2. 会被预解析
   2. 函数声明
      * 1. var fn = function(){}
        2. 不会预解析
2. **方法**
   1. 一个函数有了宿主

var obj = {

fn :　function（）{}

}

1. **构造函数**
   1. 如果一个函数,被通过new用来创建对象
   2. 通过new创建的函数
2. **回调函数**
   1. 被作为参数传递给另一个函数
3. 4

## 常量

1. 常量就是不变的量
   1. 比如:Math.PI
   2. 常量全大写
2. 变量就是可能会变化发量

## object.prototype

### Object

#### new Object

#### 字面量 {}

### 所有函数默认的原型

#### 所有函数默认的原型

##### 内置构造函数的原型

* Object.prototype
  + 终点
* Function.prototype
  + 所有函数继承 ^^^
    - 内置函数
      * Object
      * Function
      * Boolean
    - 自定义函数
      * Person
      * XXX
      * new Object()
* Number.prototype
  + 数字对象
  + new Number()
* Array.prototype
  + 所有数组
    - new Array
    - 字面量 []
* String.prototype

##### 自定义构造函数默认的原型

* Person
  + Person所有的实例
    - new Person
* XXX
  + XXX的所有实例

### Math

##### 没有实例继承Math

### prototype:显式原型

##### 原型属性

### \_\_proto\_\_:隐式原型

##### 原型对象:就是实例继承的对象

## DOM操作

1. **empty**:清空所有元素内容,返回值是实例
   1. **实现**
   2. 遍历
   3. 清除遍历到的每一个元素
      * 1. 遍历到的元素.innerHTML = ‘’;
   4. foreach(this,function(value,index)){}
   5. for(var I = 0,len = arr.length;len<10;i++)
      * 1. 为了优化
2. **remove**:删除所有的元素,**返回值为实例,为了链式编程**
   1. 遍历
   2. 删除遍历到的每一个元素
      * 1. 获取要删除的节点
        2. 获取其父节点,parentNode
        3. 父节点.removeChild(要删除的节点)
3. 对于empty和remove
   1. 只有操作元素自己,并没有涉及到获取其他内容
   2. 所以可以返回实例,继续链式变成
4. 对于html和text
   1. 获取操作,得到的是获取的值
   2. 设置操作,得到的是实例,可以继续链式编程
5. **html**:设置所有元素的内容,获取第一个元素的内容
   1. **传入0个参数,返回结果为第一个元素的内容**
   2. **传一个参数,返回值是实例,能解析标签**
   3. **传参设置,不传参获取**
   4. **实现**
      * 1. 通过arguments判断有没有传入参数
        2. 没有传入参数,获取第一个元素,返回这个元素的innerHTML的内容
        3. 如果传入了参数,遍历
        4. 设置每一个元素的innerHTML内容为传入的参数
6. **text**:设置所有元素文本内容,获取所有元素的文本内容,返回实例,不解析标签
   1. **实现**
      * 1. 通过arguments判断有没有传入参数
        2. 传入一个参数,遍历,把每一个俄元素的innerHTML加一起返回
        3. 如果传入了一个以上参数,遍历设置
        4. 设置每一个元素的innerHTML内容为传入的参数
7. **appendTo**:把所有的元素,添加到指定元素中
8. **append**:给所有的元素,添加新的内容
9. **prependTo**:把所有的元素添加到指定元素的前面
10. **prepend**:给所有的元素的最前面,添加新的元素
11. arguments:存储的是一个代表实参的对象(伪数组)
    1. 存储了所有传进来的实参
    2. 通过有一个length属性代表所有实参的数量
    3. 参数不定的时候用arguments
12. callback(回调函数)
13. 给jq工厂传入一个jq对象,可以人是传入jq对象的close版本
14. 给jq传入一个DOM对象,得到一个新的实例存储这个DOM
15. 给jq传入一个选择器,得到一个新的实例,存储着获取到的DOM元素

10..54

11..30

# 创业班第五十三天(框架封装)9.06

## 知识点

1. insertBefore(要添加的元素,添加到哪)
2. 第一次添加本体,第二次添加得clone(克隆)

## 链式编程

1. 调用一个方法,该方法可以返回一个对象,可以继续使用该对象的方法
2. 调用该对象的方法,可能仍然返回以一个对象,也可能是原来的对象
3. 那么就可以继续调用返回对象的方法,这样一直下去,直到方法没有返回对象为止

## append(),appendTo(),prepend(),prependTo()

1. **$(‘div’).append(‘字符串’):**传入字符串,直接在原来的innerHTML上累加字符串
2. **$(‘div’).append(jq对象):**传入jq对象,把这个对象的每一项分别添加到所有的div中
   1. **传入dom对象和传入jq对象一样,因为东湖村纳入的dom对象先要包装成jq对象**
   2. **只有字符串要独立处理**
3. 返回的是this,谁调用返回谁

## 属性和样式操作

1. **addClass**:添加新的class
2. **removeClass**:删除所以元素指定的class
3. **hasClass**:判断所有元素有没有指定的class,只要是有一个有就返回true
4. **toggleClass**:有删除,没有则添加
5. **attr**:设置或者获取元素的属性节点
   1. **属性节点**:
      * 1. 标签的属性就是属性节点,
        2. 只能属于DOM对象,
        3. 有DOM节点的特性(nodeType,nodeName等)
           1. nodeType节点类型:

元素节点

属性节点

文本节点

8. 注释节点

9. 文档节点

* + - 1. 所有DOM,都有一个attributes属性
      2. 存储了该DOm所有的属性节点
      3. 通过Attributes属性获取所有的属性节点对象,然后按照下标取出,获取每一个属性节点对象,在通过nodeValue属性得到这个节点只
      4. 通过DOM的getAttributes(属性节点名)**获取**
      5. **设置属性节点值**:通过setAttribute(value值)
  1. **属性:**
     + 1. 对象身上的变量,
       2. 可以属于任意节点,
       3. 属性和存储的东西没有共性(任意存储)
       4. 属性可以通过点的方式添加
       5. 通过 点 获取 或者[]获取

1. **prop**:设置或者获取元素的属性值
2. **val**:获取或者设置元素的value属性
   1. 传参数设置.val()
   2. 不传参获取
   3. 原生获取:元素.getAttrobute(‘属性值’)
      * 1. 获取默认的
        2. 最新的值通过.value获取
3. **css:**

# 创业班第五十四天(框架封装)

## 事件模块

1. on:事件绑定
2. off:事件移除
3. 事件绑定方式
   1. DOM.on事件名 = function(){}
   2. DOM.addEventListeven(事件名,function(){})(IE8不支持)
   3. IE8:DOM.attachEvent(on事件名,function(){})
4. 事件解除方式
   1. DOM.on事件名 = null
   2. DOM.removeEventListeven(事件名,function(){})(IE8不支持)
   3. IE8:DOM.dettachEvent(on事件名,function(){})
5. DOM元素都有nodeType
6. off传入事件类型,解除该事件类型的所有回调
   1. 传入事件类型和回调,那么解除该事件的指定回调

# 创业班第五十六天(git)

## shell

1. 图形化界面shell
2. 命令行shell
3. 认识bash这个shell
4. bash命令格式
   1. 命令[-options][参数]

## 命令

1. cd:切换目录
   1. cd .. 返回上一级
2. ls:查看目录
   1. –a;查看所有
   2. –l:查看详细信息
   3. –a –l == -al
3. pwd:查看当前目录
4. mkdir:创建文件夹
5. touch:创建一个文件
   1. touch js/index.js:在js文件夹下创建文件
6. cat:查看文件所有内容
   1. cat 文件名
7. more/less:查看文件
   1. less:分页阅读;q退出
8. rm:删除文件
9. rmdir:删除文件夹,只能删除空文件夹
10. mv:移动/重命名文件
    1. mv 移动的文件 移动到哪
    2. mv 原文件 修改文件名后的文件 可实现改名
11. cp:复制也可以修改文件名
    1. cp 原文件 修改文件名后的文件 可实现改名
12. head:查看文件前几行
    1. head -5 文件名
    2. head –n -5 文件名
    3. 两种方式一样
13. tail:查看文件后几行
    1. tail -10 文件名
    2. tail –n -10 文件名
    3. 两张方式相同
14. history:查看历史
15. 重定向 > 覆盖 >> 追加 :修改输出的位置
    1. ls > index.html 把ls的结果输出到index.html这个文件内
16. wget 网址 下载
17. tar 解压缩
18. 命令 –help 查看帮助文档
19. curl 网址 模拟网络请求
20. huami:查看当前用户
21. | 管道符:可以将多个命令连接
    1. 上一个命令的执行结果,会被当作下一个命令的参数
22. grep:正则匹配内容

## VI编辑器

1. 命令模式
   1. dd:删除当前行
   2. yy:复制
   3. ctrl+f:向下翻页;ctrl+b:向前翻页
   4. 进入编辑模式
      * 1. i:当前光标处插入
        2. a:当前光标后插入
        3. A:光标移动端行尾
        4. o:光标下面插入行
        5. O:光标上面插入行
2. 底行模式(shift+:)
   1. q,退出
   2. w:保存
   3. wq:保存并退出
   4. e!:返回上一次保存状态
   5. q!:不保存强制退出

[https://](https://755966092.github.io/yzz.github.io/)[755966092.github.io/yzz.github.io/](https://755966092.github.io/yzz.github.io/)

## SSH

1. 对称性加密:加密和解密使用相同的密钥
2. 非对称性加密:加密和解密用不同的密钥
   1. 公钥
   2. 私钥

## GIT分布式版本控制

1. git管理文件的三中状态
   1. 已提交:committd 🡺 git仓库
   2. 以修改:modified 🡺工作目录
   3. 以暂存:staged 🡺 暂存区域
2. 把普通目录变为git工作区
   1. **git init**
   2. **git add** 文件名 把文件放到暂存区
      * 1. **git add –A 全部加入暂存区**
        2. **git add \* 全部加入暂存区**
   3. git commit –m ‘标记’
      * 1. 执行一次commit就新建了一个版本/存盘点
        2. 可以还原到任意一个存盘点
   4. git config –global user.name/email 配置
   5. git status 查看仓库状态
      * 1. modified 被修改的
        2. 绿色的代表暂存区的文件
        3. untarched files 未追踪的文件
   6. git checkout 文件名 将暂存区的内容还原到工作区,(撤销更改)
   7. git reset --hard 提交ID

# 创业班第五十七天(git)

## git分支

1. 分支
   1. 至少要创建一次commit
   2. 一个人干多件事情的时候,为了互相不干扰
   3. 存盘点基于当前分支
   4. 新创建的的分支基于当前分支
      * 1. 会继承父分支的所有提交历史
2. git **branch** 分支名称: 创建分支
3. git branch :查看分支
4. git cheackout 分支名称:切换分支
5. git **merge** 要合并的文件名 合并文件
6. git **branch –d** 删除的分支名
   1. 删除失败可能是上次任务没有完全提交
   2. fit branch –D 强制删除
7. 不要有太多的树杈（子分支）
8. 要有一个“稳定分支”，即**master**分支不要轻意被修改
9. 要有一个开发分支（**developer**），保证master分支的稳定性
10. 所有的功能分支（**feature**）从developer创建
11. 所有功能开发完成后新建发布分支（**release**）

## GIT远程仓库

1. git push 远程仓库地址 分支名称 : 将分支共享到远程仓库
2. git pull 远程仓库地址 分支名称 : 获取分支
3. 共享仓库
   1. .git结尾的目录
   2. 必须是裸仓库(没有工作目录的仓库)
      * 1. git init –-bare: 创建裸仓库
   3. git自动忽略空文件夹
4. git remote add 变量名 共享仓库地址路径
5. git clone 远程仓库地址 [自定义名] 把共享仓库克隆至本地

## github

1. 创建密钥
   1. ssh-keygen –t rsa
   2. [www.ruanyifeng.com](http://www.ruanyifeng.com)

## 步骤

* + - 1. git config 配置本地仓库
         1. git config –-global user.name
         2. git config –global user.email
         3. git config –- list查看配置详情
      2. git init 初始化仓库
         1. git init –bare 可以初始化一个共享(裸)仓库
      3. git status 查看仓库状态
      4. git add 将工作区文件添加到暂存区
         1. 可写文件名单独放
         2. –A / \* 全部放到暂存区
      5. git commit –m ‘备注信息’ 将暂存区文件到到本地仓库
      6. git reset –hard id值 还原
      7. git log 查看本地仓库提交历史
      8. git branch 分支名 创建/查看分支
      9. git checkout ‘分支名称’ 切换分支
         1. **git checkout –b ‘分支名’** 创建并切换分支
      10. git merge ‘分支名’ 合并分支
      11. git branch –d/D 删除分支
          1. D强制删除
      12. git clone 仓库地址 克隆仓库
      13. git push 仓库地址 分支名 上传至仓库
      14. git pull 仓库地址 分支名 下载文件
      15. git remote 查看已添加远程主机
      16. git remote show 主机名 查看主机ip
      17. git remote add 变量名 地址 把地址放到变量名中
      18. git push 地址 本地分支:远程分支 将本地分支上传至远程仓库
      19. develop
      20. feature

# 创业班第五十八天(angularJS)

## MVC开发模式

1. $this - > display () 展示视图
   1. - > 调用方法
2. $this - > assgin(‘赋值给谁’,变量) 赋值

## 模块化

1. 引入angular之后,会有一个angular全局对象,对象下有很多方法
2. **angular.module(‘模块名’,[依赖的模块]) :** 可以创建一个模块
   1. **var App = angular.module(‘模块名字’,[依赖关系])**
      * 1. App就是新创建的模块,这个App又是一个对象
        2. 在此对象下也有很多方法
        3. 实现具体逻辑
   2. App.controller(‘name’,[‘$scope’,function($scope){ }]) : 创建一个控制器
      * 1. $scope是一个空对象,就是module
        2. ng-app:指定模块
        3. ng-controller:指定控制器
        4. 不能嵌套
        5. ng-repeat = ‘ (key,内容) in 数组 ’

## 指令:

1. angularJS自定义的HTML属性或者标签
2. ng-为前缀
3. **内置指令**
   1. **ng-app**:定义应用
   2. **ng-controller**:定义控制器
   3. **ng-repeat**:循环标签
   4. **ng-bind** = ‘属性名’: 绑定属性 🡺 {{属性名}}
   5. **ng-show** = ‘true / flase’ 🡺 显示内容
   6. **ng-hide** = ‘true / false’ 🡺 隐藏内容
   7. **ng-if** = ‘true / false’ 🡺 控制元素是否存在
   8. **ng-src** / href 🡺 增强图片路径 / 地址
   9. **ng-class** = ’{类名 : true / false}’
   10. **ng-include** = “ ‘模版路径’ ” 引入模版
   11. **ng-disabled** = ‘true / false’ 表单禁用
   12. **ng-readyOnly** = ‘true / false’ 只读标签
   13. **ng-checkbox** = ‘true / false’ 选中复选框
   14. **ng-selected** = ‘true / false’ 默认选中
4. 自定义指令:通过directive自定义指令
   * 1. App.directive('name', function () {  
         return {  
         *restrict*: 'EA',   
         // E:element可作为标签   
         // C:class可作为类名   
         // M:mark可作为注释   
         // A:attribute可作为标签使用   
         *template*:'<p>aaaa</p>',  
         *templateUrl*: '路径',  
         *replace*:'替换'  
         }  
        })

## 数据绑定

1. angularJS:以数据驱动的MVC框架
2. 单项绑定:模型(model)上的数据传递到视图(view)
3. 双向绑定:模型的数据传递到视图,视图的数据也可传递到模型
4. 相关指令:
   1. **模型(model)**数据向**视图(view)**模版绑定:{{}} ng-bind
   2. ng-clock:解决闪烁
   3. ng-bind-template = ‘{{数据1}}{{数据2}}’
   4. ng-init : 初始化model
   5. ng-repeat : 循环
   6. ng-switch on = ‘条件’ : 根据表达式显示或者隐藏对应的部分
      * 1. ng-switch-when = ‘匹配条件’
        2. ng-switch-default:默认选中项
5. 视图的数据向模型绑定(双向绑定(视图数据🡸🡺模型数据))
   1. 视图数据向模型传递要借助表单元素
   2. ng-model(data) –--视图向模型传递数据---🡪 $scope.data
   3. ng-sibmit 回车提交事件
   4. from事件
   5. jrap裁切jQuery插件

# 创业班第五十九天(AngularJS)

## 作用域

1. 只有函数可以新建作用域
2. 子级可以访问父级作用域,父级不可以访问子级
3. controller控制器可以创建作用域
4. 视图可以相互嵌套
5. ng-init初始化模型
6. ng-app所在的标签是根作用域

## 过滤器

1. {{val|过滤器:’格式’}}
2. 内置过滤器(内置九个)
   1. date:时间;
      * 1. yyyy – MM – dd EE h:m:s; 年 - 月 - 日 星期 时:分:秒
   2. currency:货币
   3. filter:给的数组中选定符合条件的,返回一个新数组,
      * 1. 条件可以可以是函数\字符串\对象(**对象后面的括号要加一个空格**)
   4. json:转换为json对象
      * 1. IE7以下,对象最后一个元素不要有逗号
        2. json最后一个对象也没有
   5. limitTo:’数值’; 截取数组
   6. lowercase:转换小写
   7. cuuercase:转换大写
   8. num:’保留几位小数’转换为数值:默认保留三位小数四舍五入
      * 1. number()转换字符串为数值,字符串中不能有任何非数值字符
        2. parseInt()转换字符串为数值,遇到第一个非数字字符停止
        3. parseFloat()转换字符串为数值,遇到小数点后第一个非数字字符停止
   9. ordeyBy:’某一个属性(如果排序的是对象)’:true(降序)/false(升序);对数组排序

## 自定义

1. 自定义指令(directive)
   * 1. //自定义指令  
        App.directive('name', function () {  
         return {  
          
         }  
        })
2. 自定义过滤器(filter)
   * 1. //自定义过滤器  
        App.filter('demo', function () {  
          
         return function(input) {  
         //首字母大写  
         return input[*0*].toUpperCase()+input.slice(*1*);  
         };  
        })
3. 自定义控制器(controller)

# 创业班第六十天(angularJS)

## 复习

1. XMLHttpRequest对象,具备向服务器端发送请求和接受想要的一个能力
2. var xhr = new XMLHttpRequest;
3. xhr.open(‘get/post’);
   1. post请求放在请求主体里
   2. get发送请求的时候没有请求主题
   3. 请求主体就是请求的参数的一些数据
   4. get发送可以不设置请求主体
   5. $\_POST
   6. 直接传对象
      * 1. 请求头
   7. angularJS默认支持json格式
4. xhr.send();
5. alert会阻塞代码执行

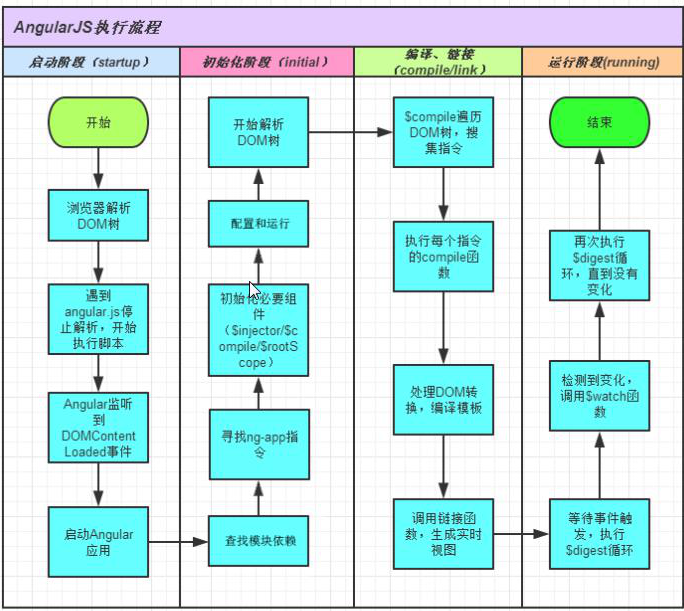
## jsonp:一种跨域的解决方案

1. 标签的src/href属性没有跨域的限制
2. jsonp利用src属性跨域
3. **angularJS的$http**
   * 1. $http({  
         *url*:'', //请求地址  
         *method*:"jsonp", //请求方式  
         *headers*:{}, //请求头  
         *data*:{}, //post传递的数据  
         *parmams*:{  
         *call*:'JSON\_CALLBACK' //回调函数(固定的  
        }, //get请求的数据  
        }).success(function(/\*返回的数据\*/){  
         '对返回的数据处理'  
        })

## 自定义服务

1. 服务的本质是一个对象或者函数
2. **factory**
   1. 一个方法
   2. 0要显式runtun一个对象回去
   3. 新建一个服务有可能依赖其他服务
   4. arr.factory('name',[/\*依赖的服务\*/],function(){  
      //自定义服务,依赖于依赖的服务  
       return function(){  
       return *data* }  
       })
3. servic
   1. app.service('name', ['依赖', function (/\*参数\*/) {  
       //添加具体方法  
       this.test = function () {  
       }  
       /\*相当于 return {  
       text:function(){}  
       }\*/}
4. value
   1. app.value('name','data')
   2. 本质上是一个服务
   3. 表现形式上是一个常量
   4. 常量就是不变的值

## 模块加载

1. 
2. js会阻塞页面解析
3. js不要卸载head里面,一般放在后面
4. 一个服务在代码中就是一个模块
5. 配置块从功能说就是配置服务

## config配置服务(配置块)

1. 大部分服务一般都对应一个provider,用来执行与对应服务相同的功能或对其进行配置
2. **App**.*config*(['$logProvider', '$filterProvider', function ($logProvider,$filterProvider) {  
    //允许一次性配置多个  
    //传递一个数组,依赖前面注入的方式  
    $logProvider.debugEnabled(false)//禁用debug  
    //新增一个过滤器  
    $filterProvider.register('name',function(){  
    return function(input){  
    return input[*0*].toUpperCase()+input.slice(*1*);  
    }  
    })  
   }])

## 运行块

1. 直接运行相应的服务模块
2. App.run(['$http', '/\*依赖服务\*/', '$rootScope', function ($http/\*依赖服务\*/, $rootScope) {  
    $rootScope.*name* = '根作用域';  
    //其他模块都可以访问到  
    //执行服务  
   }])

## 雅虎十三条

1. 尽可能的减少 HTTP 的请求数 [content]

2. 使用 CDN（Content Delivery Network） [server]

3. 添加 Expires 头(或者 Cache-control ) [server]

4. Gzip 组件 [server]

5. 将 CSS 样式放在页面的上方 [css]

6. 将脚本移动到底部（包括内联的） [javascript]

7. 避免使用 CSS 中的 Expressions [css]

8. 将 JavaScript 和 CSS 独立成外部文件 [javascript] [css]

9. 减少 DNS 查询 [content]

10. 压缩 JavaScript 和 CSS (包括内联的) [javascript] [css]

11. 避免重定向 [server]

12. 移除重复的脚本 [javascript]

13. 配置实体标签（ETags） [css]

14. 使 AJAX 缓存

## 路由

锚点就是前端的路由

1. SPA但页面应用
   1. 不产生页面跳转
   2. 把若干功能集成到一个页面
   3. 动态生成数据,通过ajax异步获取
   4. 为了增强用户体验
   5. 可以提升性能
   6. 仿制手机APP的交互

### 单页面应用

* + - 1. hashchange监听锚点变化,绑定给weindow对象

*window*.addEventListener('hashchange', function () {  
 var hash = *location*.*hash*; //获取锚点  
 hash = hash.slice(*1*)//去除#号  
 var xhr = new XMLHttpRequest;  
 xhr.open('get', 'test.php?hash=' + hash);  
 //将锚点作为参数传递给php  
 xhr.send();  
 xhr.*onreadystatechange* = function () {  
 if (xhr.*readyState* == *4* && xhr.*status* == *400*) {  
 var resute = xhr.*responseText*;  
 *document*.querySelector('.content').*innerHTML* = resute  
  
 }  
 }  
})

code.angularjs.org

* + - 1. 简单单页面

<body>  
<a href="#/index">index</a>  
<a href="#/include">include</a>  
<a href="#/no">no</a>  
<div class="content">  
 <div ng-view></div>  
</div>  
</body>  
<script type="text/javascript" src='../../angular.min.js'></script>  
<script type="text/javascript" src='../../angular-route.js'></script>  
<script type="text/javascript">  
 var App = *angular*.module('app', ['ngRoute'])  
 App.*config*(['$routeProvider', function ($routeProvider) {  
 $routeProvider.when('/index', {  
 *template*: '<h1>content</h1>'  
 }).when('/include', {  
 *template*: '<h1>include</h1>'  
 }).when('/no', {  
 *template*: '<h1>no</h1>'  
 }).otherwise({  
 *redirectTo*: '/index'  
 })  
 }])

* + - 1. json\_encode(arr);pHp把数组转换为字符串
      2. 前端路由就是锚点
      3. 路由参数:$routeParams:获取参数
         1. url问号传参数
         2. 占位符传参
      4. foreach($list as $key => $val)

## 其他

1. angular.element():把原生DOM对象转换为jQuery对象
2. Angular’s jqLite
3. bower:用来下载插件的
4. npm install bower -g --registry=https://registry.npm.taobao.org

## 作业

* jQuery设置请求头

# 创业班第六十一天(Gulp)

1. 常用工具:Grunt,Gulp,F.I.S(百度),webpack
2. 项目构建
   1. 使用Sall或Less等CSS预处理器解析css
   2. 合并css,js代码,压缩HTML,css,js,images,可以加速网页打开速度
   3. 构建工具就是通过简单配置可以帮我们实现合并,压缩,效验
   4. npm install 工具名称

npm install –g gulp--registry=https://registry.npm.taobao.org

* 1. 设置淘宝镜像为默认下载地址:npm config set registry=https://registry.npm.taobao.org
  2. 本地安装: npm install gulp --save-dev

**gulp-less** 编译LESS文件

**gulp-autoprefixer** 添加CSS私有前缀

**gulp-cssmin** 压缩CSS

**gulp-rname**重命名

**gulp-imagemin** 图片压缩

**gulp-uglify** 压缩Javascript

**gulp-concat** 合并

**gulp-htmlmin** 压缩HTML

删除空白符:collapsewhitespace:true

**gulp-rev** 添加版本号

版本号为MD5字符

**gulp-rev-collector** 内容替换

**gulp-useref** 合并文件,不压缩

**gulp-if**

在目标根目录创建一个gulpfile.js配置文件

var gulp = *require*('gulp');  
var less = *require*('less');   
gulp.task('less', function () {//引入依赖包  
 gulp.*src*('路径')

//需要构建资源的路径，字符串或数组（可以正则方式书写）  
 .pipe(less()) //管道，将需要构建的资源“输送”给插件。  
 .pipe(gulp.dest('保存路径'))

// 构建任务完成后资源存放的路径（会自动创建）  
}

[www.npmjs.com](http://www.npmjs.com)

npm install === npm i

package.json

# 创业班第六十二天(补课)

## 单页面

1. 基本目录结构
   1. public()
      * 1. css
        2. js
        3. images(UI图)
   2. views(视图)
   3. scripts(业务逻辑js)
   4. uploads(内容图,上传文件)
   5. api(接口目录)
2. 开发步骤(使用angularJS)
   1. 引入
   2. 指定ng-app
   3. 创建模块var app = angular(‘name’,[依赖模块])
   4. 按模块的思想开发将具有类似功能的程序归类
   5. 将使用控制器归类,所有指令归类
   6. 在使用时将这个独立的模块作为app的依赖使用
   7. 通过块配置和when筛选执行哪一个模块
   8. 创建控制器app.controller

## 移动端

1. em相对于自己当前的字号长度
2. rem相对于跟元素的字号长度html元素
3. 浏览器的默认字号16px
4. 1rem = 16px
5. C3属性,PC端尽量不用,移动端支持多
6. 媒体查询
   1. all 所以设备
   2. print 打印机和打印机预览
   3. screen 屏幕电脑屏幕 手机屏幕 屏幕电脑

@media only screen and (max-widthj:320px){

html{

background:pink

}

}

* 1. 关键字
     1. device-width:屏幕宽度

## css预处理器

Less

lessc 源文件.less 转换后.css

* + <link rel= ‘stylesheet/less’ herf=’xx.less’>
  + <script src = ‘js/less.js’></script>
  + less.js内部可以发送ajax请求
  + 通过一些正则进行替换操作
  + 然后将替换的结果生成一个<style>标签
  + style = document.createElement(‘style’)
  + style.innerHTHM = 替换的结果
  + less里@import ‘路径’; 引入其他less文件
  + 盒子都要有高度
  + :not(.class): 排除

## git

1. .gitignore:忽略文件
2. git difftool:比较工作区与暂存区的不同
3. git difftool ‘’ 比较与特点提交的差异
4. git difftool ‘’ ‘’ 比较两次的差异
5. git –hard
   1. -–soft
6. git push 地址 – delete 删除分支

# 创业班第六十三天(nodeJS)

## JS文件过多问题

1. 命名冲突
   1. 模块开发
   2. 代码复用
2. 文件依赖

## 如何进行模块化开发

1. 全局函数
   1. 会造成命名冲突问题
2. 命名空间
   1. var name = { }
   2. 通过约定的方式定义的
   3. 可以解决命名宏图问题,不好
   4. 驼峰命名:autoFileName
   5. 匈牙利命名: 类型+变量名;fnQueryNode
   6. 无法做私有封装
3. 划分私有空间,需要约定
   1. var func = (function () {  
       var add = function (a, b) {  
       return a + b;  
       }  
       return {  
       add: add  
       }  
      })(window.func ||　{});  
      func().add()

## SeaJS

1. SeaJS所在的路径为默认根目录
2. condi : 配置
3. cmd规范 : define用来定义模块的
4. require:加载JS
   1. 原理:创建script标签,引入js文件
5. config:配置

seajs.config({

base:'定义根目录',

alias: {

'别名' : '路径' //方便调用

}

})

1. moudle.exports 🡺 相当于return
   1. 可以返回方法,字符串,对象
   2. 不同之处
   3. 两次moudle.exports下面会覆盖上面
   4. moudle.exports的初始化是一个空对象
   5. 用来打破封装性,返回想要的属性和方法

# 创业班第六十四天(NodeJS)

1. node.js V8引擎解析js
2. 事件驱动,无阻塞I/O模型
3. js引擎套一个壳子,写js代码传给引擎,操作硬件
4. js> 🡺 nodejs 🡺 libuv
5. nvm npm nrm
6. 安装与配置
7. repl交互式运行环境
8. Ctrl + c 退出
9. **fast-init**
10. process.stdout.write(‘log原型’)
11. process全局对象,和现在启动中的node进行交互

## 总结

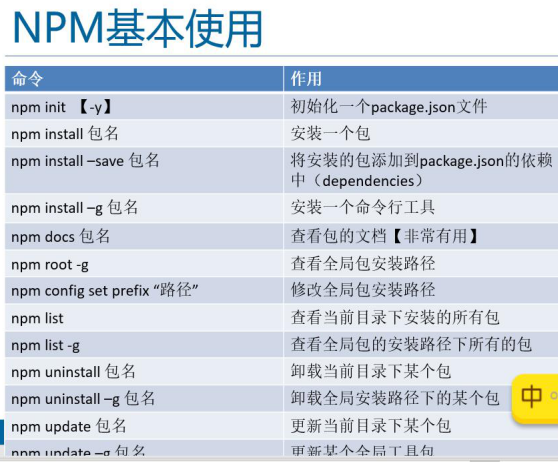
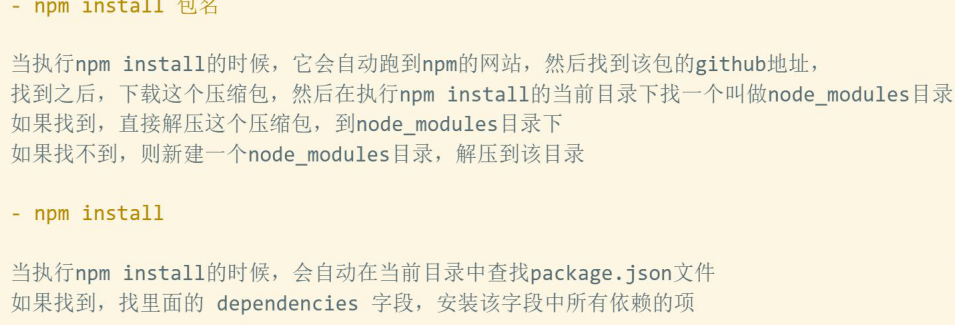
1. node(一门技术)和JS的关系
2. v8引擎
3. 特点
   1. 无阻塞I/O,事件驱动
   2. 轻量/高效
4. nvm的安装
5. global
6. 断言assert(foo==3(田条件),’输出内容’)

# 创业班第六十五天(NodeJS)

## 知识点

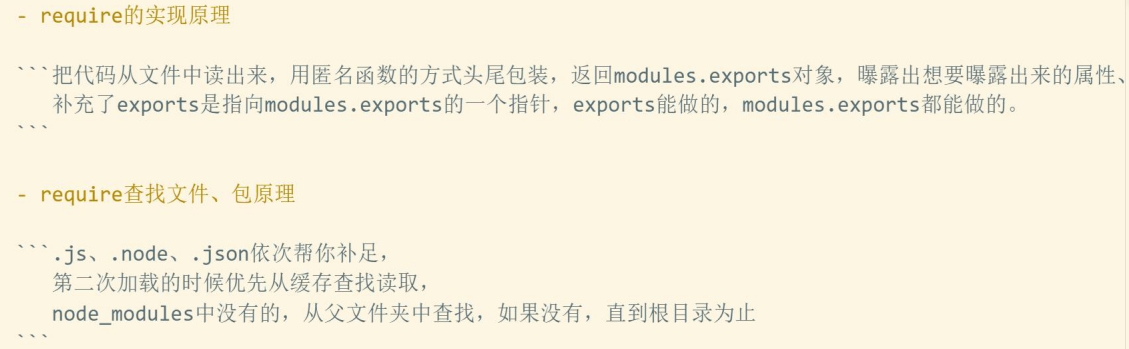
1. node是技术
2. javascript是语言
3. 基于v8引擎,不用考虑兼容性
4. 特点:事件驱动,异步非阻塞i/o模型
5. node优点:轻量高效
6. node.js可运行运行在linux系统上
7. path环境变量
8. REPL运行环境
9. node 运行js文件
10. js文件所在目录
    1. node js文件名
11. \_\_dirname和\_\_filename
    1. 文件名和文件完整路径
12. global
13. cinsole
    1. assert(foo == 1, ‘失败’); 断言
    2. time(‘timer’) ---🡪 timeEnd( ‘timer’ )
14. 一个文件就是一个模块
15. 在一个莫款内的变量-函数-对象都属于这个模块,对外封闭
16. require(‘路径’):引入模块(读取js文件)
17. exports
    1. module.exports的一个别梦-引用,
    2. 打破封装性
18. readFile(‘路径’,’编码格式’,回调函数)
    1. 异步读取文件
19. rreadFileSync(‘路径’,’编码’)
    1. 同步读取
20. eavl把字符串转换成js代码
21. node会按照.js.node.json的此讯依次补足拓展名,一次尝试
22. 田永强<深入迁出nodejs>

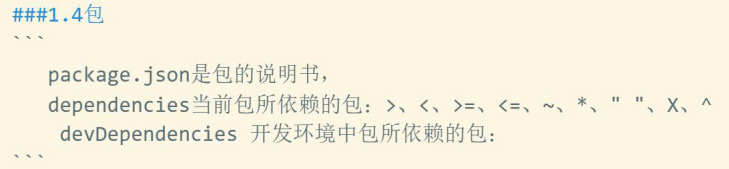
## 包

1. 包是一个文件夹
2. package.json : 包的描述文件,说明文件
3. Bin : 存放可执行二进制文件的目录
4. Lib ; 存放javascript代码的目录
5. Doc : 存放文档的目录
6. Test : 存放单元测试用例的代码
7. 
8. npm init
9. 
10. npm install : 包管理工具
11. 
12. cnpm
13. cnpm
14. 箭头函数
15. ng-bind ;: 可以防止页面闪烁
16. angular底层核心原理,了解胀检查机制

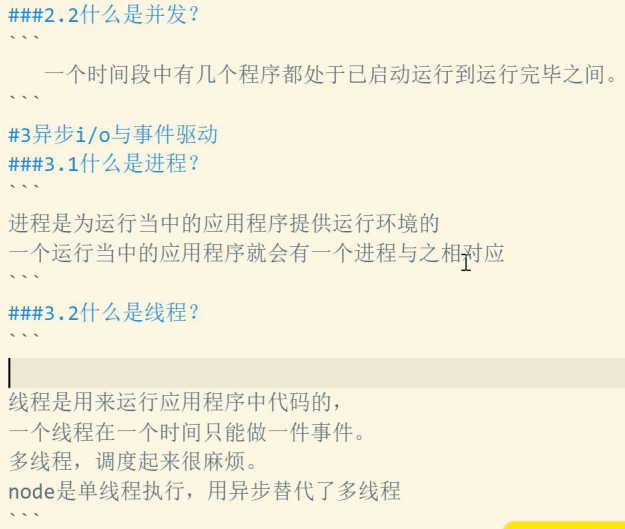
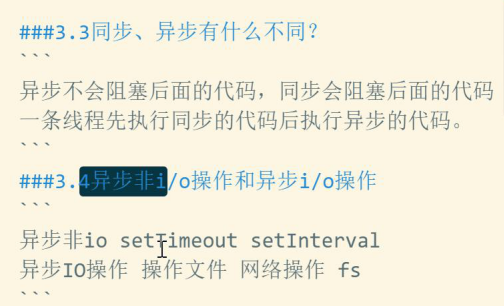
# 创业班第六十六天(NodeJS)

## 复习



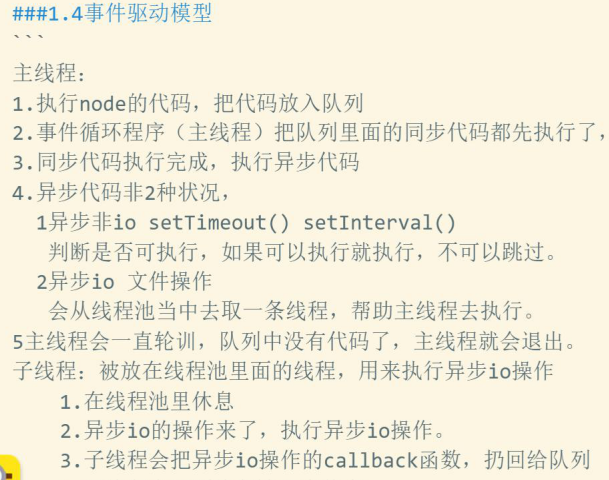
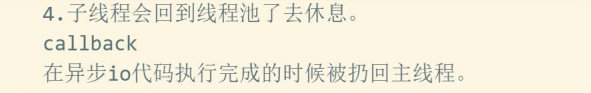
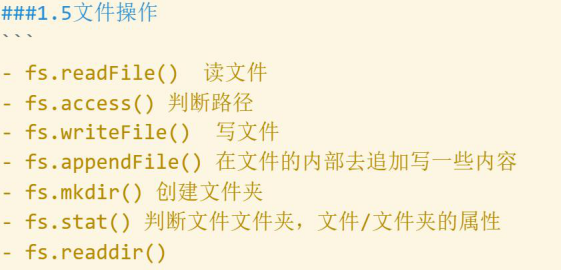


## I/O

1. input / output
2. I / o 服务器上的读写操作
3. **并发:**一段时间段内有几个程序都处于已启动运行到运行完毕之间
4. 
5. 异步:
   1. ajax
   2. setTimeout
   3. setInterval
   4. 先执行同步代码,再执行异步代码
   5. 同步代码会阻塞掉异步和同步代码
   6. 

# 创业班第六十七天（NodeJS）

## 复习

1. i/o：输入输出，文件读写操作，网络操作中的请求和应答
2. 并发：同意时间内，很多程序都处在开始执行到结束之间
3. 进城：为程序提供运行环境
4. 线程：执行应用程序中的代码，同意时间内只能干一件事情
5. node中异步就是帮助完成多线程编程
6. 多线擦编程比较复杂
7. 同步异步
   1. 同步会**阻塞代码**，优先执行，主线程执行
   2. 异步不会阻塞代码执行，后执行
8. 事件驱动模型
   1. 
   2. 
9. 文件操作
10. 
11. 迭代：f(1),f(2),f(3),f(4)...
12. 递归：f(),f(f()),f(f(f())),f(f(f(f())))……….
13. path:
    1. path.join():字符串拼接成路径

$http.({`

url : “./a”,

tethod : ‘get’

}).success(function(data){

data是报文体;

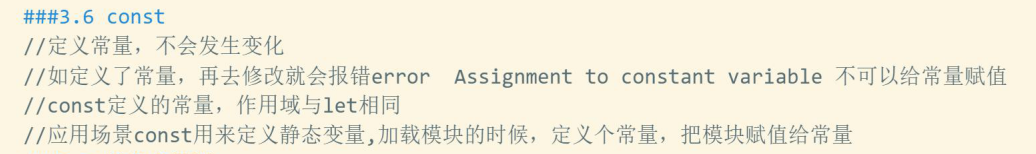
}).then(function(data){

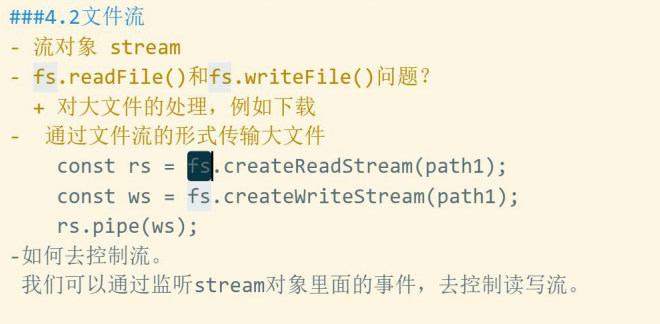
data:是报文,如果取里面的json数据需要data.data

},function(err){

})

## debug

1. 调试 :
   1. node debug xxx.js
      * 1. n下一步
        2. s步入
        3. o退出
        4. –help查看帮助
   2. node-inspector
   3. VSCode调试
   4. 打开项目文件夹 ->  ->
   5. 点击甲壳虫 ->
   6. 设置调试环境node.js ->
   7. 打开launch.json ->
      * 1. "program": "${workspaceRoot}/xxx.js",
2. IDE:集成开发环境
3. use strict : 严格模式
   1. 消除代码运行的一些不安全的地方,保证代码运行的安全
   2. 提高编译器效率,增加运行速度
   3. 为未来新版本的js做好铺垫
   4. ‘use strict’以下的内容都执行严格模式
4. let 定义变量 : 定义的变量值作用于一个或括号内,一个作用域
5. const : 定义常量,引包的时候使用
   1. 常量不可修改值,内容恒定
   2. 
6. 字符串拓展
   1. str.includes(str):是否包含str这个字符串
   2. str.startWith(str):是否是以str开头
   3. str.endWith(str):是否以str结尾
   4. str.repeat(次数):循环打印字符串
   5. 链接字符串 `前${tab上面的点}后`
7. promise
   1. 异步流程控制解决方案
   2. 回调地狱
   3. .then()
8. 异步流程 🡪 回调函数的嵌套
9. Buffer
   1. 一个数组对象,每个元素存放两位十六进制数
   2. 十六进制数字
   3. 文件流:
      * 1. 流对象



1. 'use strict'
2. const fs = require('fs');
3. const rt = fs.createReadStream('./1.wmv');
4. const wt = fs.createWriteStream('./2.wmv');
5. rt.pipe(wt);

# 创业班第六十八天(NodeJS)

## 网络编程

1. cs / bs 架构
2. 编写程序是两台两台联网的计算机可以完成数据交互,完成网络通信
3. 服务器系统:linux的centos , ubuntu
   1. 服务器多远centos
   2. 应用服务器:作为服务器执行共享业务应用程序的底层的系统软件
   3. web 服务器:是一种应用服务器,提供web服务,对内提供web应用程序的运行环境
   4. apache / Nginx / IIS / tomcat : 服务器



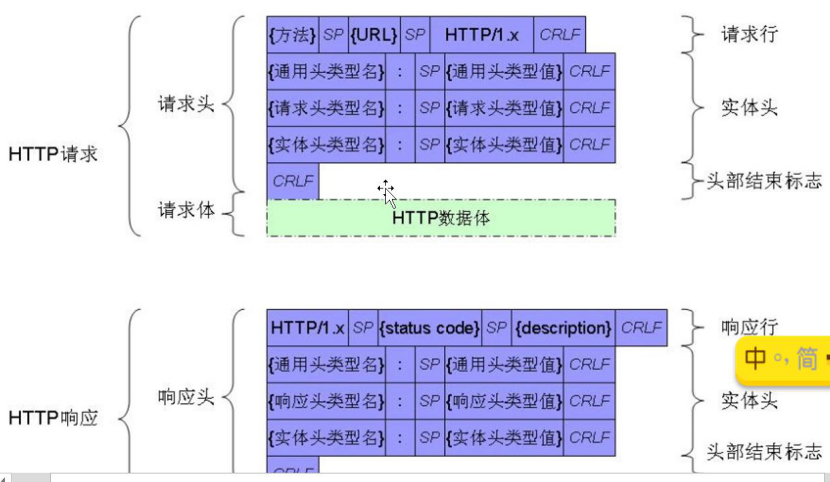
## 跨域

1. 域名不同
2. 二级域名不同
3. 端口不同,协议不同
4. 应用场景
   1. 动静分离
   2. 不适合开接口的情况

## 开发分工

## 协议

1. 协议就是双方约定好的一些数据格式





# 创业班第六十九天(angular)

## 复习

1. 协议:帮助我们简化一些事情(交通规则)
2. http协议:浏览器和服务器之间的关系
3. get/post区别
   1. post有报文体;传值安全
   2. get在地址栏传参数
4. package.json:说明文件
5. 协议:cookie

## 网站优化-🡪思想

1. 能压缩的压缩,能合并的合并
2. 减少script标签,减少http请求
3. 少用js去遍历对象或者写for循环
4. 能提前运算好的提前运算好,使用枚举法(把所有可能发生的情况列举出来)
5. CDN节点,从最近的CDN节点请求数据
6. 流式(响应式)图片 ---🡪 图片压缩
7. 

## http模块

1. request:有请求发送给服务器后触发该事件
2. 触发事件后调用回调函数function(请求对象, 相应对象){}-
3. 请求报文封装到request对象里
4. 实用repsponse里面的方法构造响应把平稳,调用response里面的方法,将一下相应仍会去
5. .listen(3000):监听3000端口
6. node默认的80
7. java默认8080
8. nodejs默认3000
9. writeHead(200,{‘content-Type’:’text/html;charset:utf-8’})

# 创业班第七十天(NodeJS)

## 复习

### 第一天SeaJS

##### 模块化编程

##### 组件化、模块化

* + 1. 开源精神

##### 命名冲突、文件依赖问题

* + 1. 块级作用域
    2. es6.ruanyifeng.com
    3. es6原生支持模块化，支持文件引入

##### 依赖注入

* + 1. angular的依赖注入是为了防止文件压缩
    2. 要什么，在参数上体现

##### 开闭原则

* + 1. 不修改别人的东西

##### AMD 🡪 RequireJS

* + 1. 提前执行依赖模块
    2. 懒加载

##### CMD 🡪 SeaJS

* + 1. 延迟执行依赖的模块

##### SeaJS

* + 1. seajs.use（‘引入模块路径’，回调函数）
    2. 启动模块

define（function（require，exports，module）｛

var obj = require（‘./JS路径’）；

exports.add = obj.add；

exports.add2 = obj.add2；

//只暴露一个

module.exports = add；

｝）

* + 1. 写一个js模块
    2. 通过define定义一个模块
    3. 扔出对象exports.add=obj.add

### 第二天nodejs

##### 双数稳定版，单数激进版本

##### npm3.0扁平化包管理

##### npm2.0树形包管理

## express

1. express –save-dev 脚手架工具
2. 快速构架年框架模型
3. 惊人的路由能力
4. var app = express();创建一个服务器实例
5. express.static（）将某个目录里面的文件变成一个静态网站
6. app.use使用express.statuc()的方法
7. app.get（‘/getData’，function（）｛
8. res.end（‘’）
9. ｝）

# 易错点：

1. **Vertical-align：midden； 去掉图片默认的间距**
2. **谷歌不支持12号以下字体**
3. 浮动的盒子不设置宽，容易掉盒子
4. **清除浮动的方法**
   1. **Clear：both**
   2. **Overflow：hidden**

//双伪元素

.clearfix:berore , .clearfix:after {

Content:”0”;

Display:table;

}

.clearfix:after {

Clear:both;

}

.clearfix{

\*zoom:1;

}

1. 隐藏盒子
2. Display：none；隐藏盒子，并且不占位置
3. Overflow：hidden；隐藏超出部分
4. Visibility：hidden；隐藏盒子，占位置
5. Opacity：0；隐藏盒子，占位置
6. Position-left/top：-999em；隐藏盒子，测试不占位置
7. 关闭a 的链接：
   1. Href=“javascript：（javascript：void(0)）；”；关闭a链接
   2. A的href = “”：效果是刷新页面
   3. Href = “#”：跳到页面顶部
8. 鼠标滚轮事件
   1. window.onmousewheel=function（）｛｝

入口函数的区别：  
window.onload = function({}) 页面全部加载后执行函数

$(document).ready(function(e){//代码}) 文档加载完成，图片未加载执行函数（JQ常用）

$(window).ready(function(e){//代码}) 文档图片均加载完成

鼠标滚动事件的兼容写法：DTD，旧版本的浏览器遵循一些新的规则

未声明DTD，谷歌只识别DTD（ie9+认识他）：document.body.scrollTop

已经声明DTD，IE6/7/8 可以识别，（IE9+在任何时候）：document.documentElement.scrollTop

火狐/谷歌/ie9+以上支持的（不管DTD）：window.pageYOffset

浏览器的四大事件：offsetX获取对象的长度；scrollTop，获取窗口滚动的距离（存在兼容性问题）；clientX，获得窗口的位置；event事件；

Window.onscroll –->鼠标滚轮滚动触发事件