# jQuery

|  |  |
| --- | --- |
| 选择器 |  |
| ID、类、标签 |  |
| 并集、交集 | 逗号—注意和后代的区分 |
| 子代、后代 | 》 空格 |
| ：eq(), :odd :even | 索引从0开始的，寄偶 |
| Children() | 相当于子代 |
| Find() | 相当于后代 |
| Siblings() | 兄弟元素，但不包括自己 |
| Parent() | 父元素 |
| CSS |  |
| 设置样式 | Css({“width”:”200px”}) |
| 获取样式 | Css(‘width’) |
| class |  |
| 添加addClass() |  |
| 移除所有removeClass() |  |
| 移除单个removeClass(‘box’) |  |
| 判断hasClass(‘box’) | 返回的是true和false |
| 切换toggleClass(‘box’) |  |
| 动画 |  |
| Show、hide(speed,callback) | 显示、隐藏。修改的是元素的width height opacity |
| Toggle() |  |
| slideUp\slideDown | 滑入、滑出 |
| slideToggle() |  |
| fadeIn\fadeOut | 淡入、淡出。修改的是opacity |
| fadeToggle() |  |
| fadeTo(speed,value,callback) |  |
| 自定义动画  animate({params},speed,cbcak) | 1.{params}:要执行动画的CSS属性  2.speed:执行动画的时长（可选）  3.callback:动画执行完成后立即执行的回调函数 |
| easing: swing\linear |  |
| 动画队列：stop( clearQueue,  jumpToEnd) | 默认值都是false,   1. 是否清除，如果为true 2. 是否跳转最终效果，如果为true,瞬间完成 |
| DOM |  |
| 创建 |  |
| 添加 |  |
| 清空 |  |
| 删除 |  |
| 克隆 |  |
| 操作属性 |  |
| 设置单个属性 | attr(‘name’, ‘value’) |
| 设置多个 | 以对象的形式 |
| 获取属性 |  |
| 移除属性: removeAttr(‘title’) | 如果属性不存在，返回undefined |
| Prop方法 | checked\selected\disable,如果用attr()返回的不是true/false   1. 设置：$(‘:checked’).prop(‘checked’,true) 2. 获取：$(‘:checked’).prop(‘checked’)🡪返回true/false |
| 值与内容 |  |
| Val() | 用于设置和获取表单元素的值：input/select/textarea |
| Html() |  |
| Text() |  |
| 尺寸 |  |
| Width()/height() | 设置或者获取，返回的是number ,200  而使用.css(‘height’)返回的是字符串，200px |
| 坐标值：offset（） | 如果操作的没有设置定位（默认position:static）  会修改为position:relative |
| Position（） | 只能获取，不能设置  返回的是对象：{ left：200，top:200} |
| 操作滚动：  scrollTop(100)/scrollLeft(100) | 有参数就是设置，没有就是获取 |
| 操作事件 | bind() 事件绑定 🡪 被on()代替；delegate() |
| On()：  一次只能绑定一个事件  jQuery1.7之后，on统一所有事件处理方法 | $(‘.box’).on(events,selector,data,handler)  Events:  Selector:如果没有这个后代元素，事件就自己执行  Data:传递给处理函数的数据  Handler:事件处理函数 |
| Delegate()  事件委托：支持动态绑定事件， | $(‘.box’).delegate(‘p’, ’click’, function(){  为.box下面的所有的p标签绑定事件  }) |
| off()  解绑事件 | $(‘.box’).off():解绑元素的所有事件  $(‘.box’).off(‘click’)  $(‘.box’).off(‘click’, ‘\*\*’):  解绑所有代理click事件，元素本身的事件不会被解绑 |
| trigger() | 触发事件：$(‘.box’).trigger(‘click’) |
| triggerHandler() | $(‘.box’).triggerHandler(‘focus’)  这种方式，不会触发浏览器行为。比如：文本框获得焦点的行为 |
| jQuery事件对象 |  |
| Event.type | 事件类型 |
| Event.data | 存储绑定事件时传递的附加数据 |
| Event.target | 点了谁就是谁 |
| Event.currentTarget | 当前DOM元素，等同于this |
| Event.delegateTarget | 代理对象 |
| screenX / screenY | 对应屏幕左上角的值 |
| OffsetX / offsetY | 点击的位置距离元素的左上角的位置 |
| ClientX / clientY | 距离页面左上角的位置（忽视滚动条） |
| PageX / pageY | （会计算滚动条的距离） |
| Event.wich | 鼠标按键类型 |
| Event.keyCode | ASCII |
| Event.stopPropagation() | 阻止事件冒泡 |
| Event.preventDefault() | 阻止浏览器默认行为  Return false |

### Day 01 jQuery 选择器

1. 引包:
   1. 忘记引包：$ is undefined
   2. 引包顺序：NOT 404
   3. 引包路径：
   4. 实现功能
2. Jquery 入口函数
3. Function:
4. 功能：入口函数
   1. $(function(){ })
   2. $(document).ready(function(){});
   3. Window.onload🡪执行时机：js的入口函数会等待图片等资源加载完成。而JQ不会。
5. 加载：DOM树加载完成，window.onload
   1. jquery的入口函数比js的入口函数先执行
   2. jquery不会等待图片资源的加载
   3. js的入口函数会等待图片资源的加载
6. $ 本质是一个函数，根据参数不同，功能不通。
   1. 字符串选择器：html , css
   2. TagName, id, class
   3. 语法：$(selector)
7. DOM对象：🡪通过js方法获取的元素；
   1. 功能：把dom对象转化为jquery对象
   2. 语法：$(DOMobj)
   3. jquery对象: 🡪其实就是DOM对象的包装集(伪数组)，dom对象以伪数组的形式存放在jquery的对象中
8. Dom对象转化成jquery对象 和 jquery对象转化DOM对象
   1. JqueryObj[ index] ---- jqueryObj.get(index)
   2. 练习—开关灯、隔行换色

### Day 02 jQuery 操作样式、操作动画、操作dom节点

1. 显示、隐藏
   1. Show.hide 🡪 改变的是width, height, opacity
2. 滑入、滑出
   1. sildeDown, Up,Toggle 🡪 改变的是height
3. 淡入、淡出
   1. fadeIn, Out 🡪 (speed,callback)
   2. fadeToggle – 状态取反
   3. fadeTo – (speed, to, callback)
   4. speed: 动画执行时间，可以是数值，也可以是指定字符串 🡪 必须传入这个参数，不能省略，ex:fadeTo( 1000, 0.5)
   5. to: 指定要淡入或淡出的透明度值
   6. callback: 回调函数
4. 自定义动画：animate(prop, speed, easing, callback )
   1. prop: property 🡪 一个对象，包含多组样式，设置动画最终效果
   2. easing: 有2个参数，为swing和liner。🡪 不传参，默认是swing.
   3. stop( clearQueue, jumpToEnd ) 🡪 停止当前动画 🡪 动画队列
   4. clearQueue: 是否清楚动画队列
   5. jumpToEnd: 是否跳转到动画最终效果
5. 创建节点、添加节点
   1. $(“div”).append(“”); 🡪 追加到所有子元素的后面
      1. Prepend() 🡪 追加到所有子元素的前面
      2. Before() 🡪 当前元素的前面
      3. after() 🡪 当前元素的后面
6. 可以传2种参数：jquery对象 + html字符串。剪切功能：先获取再追加。
   * 1. htmlStr( ) 🡪 html格式的字符串，会将字符串转化成jquery对象。
7. 清空元素 – empty
   1. $(“div”).html(“”) – innerHtml
8. 子元素注册的事件不会被清除，会造成内存泄漏
   1. $(“div”).empty(); 🡪这种就不会造成内存泄漏
9. 移除 – remove
   1. $(“div”).remove(); -- 连自己都干掉，而empty – 只干掉儿子
10. 克隆 – clone
    1. var p = $(“p”).clone(); -- $(“div”).append(p);
    2. clone(true) – 括号里面写ture, 连事件都一起克隆了，默认深度克隆，不克隆事件

### Day 03 jQuery 操作属性、事件

1. 修改属性：
   1. 单属性：.attr() 🡪 有的话，就修改，没有就添加。
   2. 多属性：对象形式，-- { ““:”” }
      1. 获取 attr(“sele”)🡪 如果没有设置，返回的是undefined, 没有设置就是添加，设置了就是修改
2. 删除属性
   1. removeAttr() 🡪 彻底删除；如果不传参数没有任何操作，注意区分removeclass.
      1. 3个单属性（checked, selected, disable）🡪 用prop来操作
3. value获取
   1. 获取和设置2种方式 –表单元素的值
      1. 案例：if($.trim(txt) == “” ) –> 清除前后空格
      2. blur失去焦点；focus获得焦点
      3. val可以用来操作select 和 textarea
4. 内容
5. 尺寸
6. 坐标
   1. offset( ), 获取，返回一个对象 { top:xx, left:xx }
7. 滚动
   1. scroll( ) $( window )/$( document )
   2. scrollTop
   3. scrollLeft
8. 如何给一个对象，连续注册事件，绑定方法 🡪bind( types, data, fn )
   1. types: 字符串，如果有多个就用空格隔开。
   2. enentHandler: bind( “click\_mouseenter”, function( ) { } );
   3. bind方法动态创建的元素不能绑定事件。delegate 方法不推荐使用（事件委托）
9. on方法绑定事件🡪 on( types, selector, data, fn )
   1. 委托div给下面的子元素P标签绑定事件
      1. on( types, selector, eventHandler)
         1. $(“div”).on(“click”, “mouseenter”, “p”, function() {} );
   2. 给自身绑定事件
      1. on(“click”, function (){} );
10. off ( ): 解绑事件（unbind | undelegated）
    1. off ( ) :　不传参数的话，所有的都解绑

$(“div”) .off(“click”): 自身和委托绑定的事件都会解绑

* 1. off( type, “\*\*” ): 解绑委托事件中的type类型

1. 触发事件：手动触发 和 代码触发;直接调用事件🡪 可以当定时器来用
   1. trigger( type, data) 🡪 $(“#txt”).trigger(“focus”);
   2. triggerHandler 🡪 $(“#txt”).triggerHandler(“focus”); 🡪不会触发浏览器的默认行为
2. 事件对象：event.type 和 event.keyCode
   1. Log(e.screenX, e.screenY) – 相对于屏幕，鼠标的位置
   2. Log(e.offsetX) – 相对于点击的元素
   3. Log(e.clientX, clientY) – 相对于页面
   4. Log(e.pageX, e.pageY) – 相对于文档（document）, 加上滚动条的滚动距离
3. $(“#id”).on(“click”,“p”,function(e){
   * 1. e.target – 点击谁就是谁
     2. e.currentTarget & $(this) – 当前对象
     3. e.delegateTarget – 谁代理时间，就是谁
   1. console.log(e);});
4. 阻止事件冒泡：
   1. return false. 在第一个函数下面写上，还能阻止浏览器的默认行为
   2. e.stopPropagation(); -- 阻止事件冒泡
   3. e.preveDefault(): -- 阻止浏览器的默认行为
5. 链式编程：保证返回值是自己，return this;
6. 隐士迭代：不需要手动遍历，jquery内部做的循环
7. 显示迭代：

# DOM

### Selector

1. each();
2. supprot = {}
3. myTrim();
4. getElementById();
5. getElementsByTagName():
6. getElementsByClassName();
7. get();
8. select();

### DOM

1. appendTo:
2. append:
3. prependTo:
4. prepend：
5. remove：
   1. $(“div”).remove()—-触发事件，移除div。
   2. dom方法：parentNode.removeChild( node );
   3. 要找到“div”的父元素，再移除‘div’ .
   4. 2个 this指向
6. next：需要另外一个函数，
   1. 功能：获取下一个紧邻元素节点。
   2. 语法结构：var node = $(“div”).next(“p”);
      * 1. 获取指定P元素节点还不能实现
   3. dom方法：nextsibling,但是这个方法获取的节点还包括文本节点，需要排除，另外，是紧邻的P元素，后面的P元素也不行
   4. 实现方法：
      * 1. while循环，判断下一个节点是 文本 还是 元素 节点。
        2. 一直找到 nodeType == null; 为止。
7. nextAll：也需要另外一个函数，还要改变each()
   * 功能：找到实参下面的所有子元素；
   * 语法结构：$(“div”).nextAll();
   * var getNextElemets = function(){}; node = node.nextSibling;不停的赋值，不停的找，if(node.nodeType===1)
   * this指向不明；

### Event

1. on();
   * 多次绑定事件；
   * 兼容性：addEventListener(3个参数),attachEvent;
     + 1. addEventListener
   * 事件对象：
   * 事件指向:IE的this指向window,手动apply修改
2. off():
3. 入口函数：
   * + 1. 判断支持DOMContantLoaded,🡪文档树的加载

### CSS

mytrim: 从select沙箱内部拿出来。

removeClass( );

addClass( );

hasClass( );

toggleClass( );

### Property

attr();

checked / selected / disabled

node.checked / node.

$checkbox.prop(“chendked”,true)

val();

html();

text();

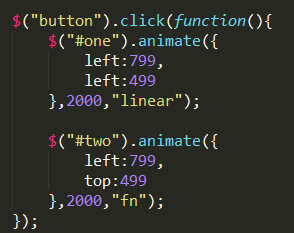
递归

for循环是结束条件->数组长度

find();

### Animation

1. 把每一个不定值都替换成函数，然后调用，返回一个对象
2. 暴露select,扩展赋值， select : select – luo.select(selector,context) – luo.select = Sizzle;



3种get返回的都是dom对象，不是数组

innerText与innerHTML都可以获得页面元素之间的内容

不同的是innerText只会获取文本信息 ，而innerHTML会将标签之间的标签也获取而，而且是原样输出，innerText是IE8及早期的IE浏览器支持的，火狐浏览器不支持，火狐只支textContent.

节点：

|  |  |  |  |
| --- | --- | --- | --- |
|  | 元素节点 | 属性节点 | 文本节点 |
| nodeName | 标签 | 属性 | 文本#text |
| nodeType | 1 | 2 | 3 |
| nodeValue | 始终是 null | 属性值 | 文本内容 |

parentNode: 获取元素所在父元素的节点（包括所有）

parentNode，a不能是数组

childNodes: (标准)，文本节点 和 标签节点

children：只返回标签节点，非标准，但是没有兼容性。

removeChild:

nextSibling: 下一个节点

nextElementSibling

previousSibling

previousElementSibling

firstChild

lastChild

firstElementChild

lastElementChild

# 内置对象

### Math

|  |  |
| --- | --- |
| Math.ceil( ) | 向上取整 |
| Math.floor( ) | 向下取整 |
| Math.max(x, y) | 在x和y之间的最大值 |
| Math.min(x, y) | 最小值 |
| Math.round( ) | 四舍五入 |
| Math.random( ) | 随机数，可以等于0，小于1。Random( )\*600+400 🡪 400到1000的随机数 |
| Math.pow( x, y ) | x 的 y次方 |
| Math.abs( ) | 绝对值 |

### Array

去重，排序，颠倒

|  |  |  |
| --- | --- | --- |
| 迭代 | filter( ): | true的留下，false的放弃；根据限制条件查找数组里的元素, 参数是function |
| forEach(): | 遍历数组的元素 |
| 排序 | reverse( ): | （背面、相反）把数组元素反转 |
| sort( ): | （种类、方式，）把数组元素排序：arr.sort(); 按数字大小和首字母先后顺序 |
| 清空数组 | array = [ ]; | 🡪 推荐，相当于重新赋值数组 |
| arr.length = 0; | length属性可以赋值，其他语言中length是只读。 |
| arr.splice | ( 0, arr.length ): 删除数组中所有的项目 |
| 操作数组 | join( ); | 把数组中的元素连接成字符串输出，默认逗号隔开，可以传参 |
| concat( ); | 连接2个数组， |
| slice( start,  end ); | 拷贝：包括start, 不包括end参数为索引值，如果end为负数，就等于arr.length – end。截取数组中的部分🡪 返回这部分，不影响数组； |
| splice( );  分2种 | 截取、插入：( start, num, ‘a’, ‘b’, ‘c’ ) 开始的索引（包括），截取的数量, 后面是插入的数据，从删除的地方开始插入；🡪返回的是截取数组 |
| 截取：只传2个参数和slice一样，但是会破坏原数组，返回新数组； |
| 追加、删除 | push ( ) | 添加到数组的最后面 –> 返回的是数组的length |
| unshift( ) | 添加到数组的最前面 –> 返回的是数组的length |
| shift( ) | 删除数组最前面的一个，返回了删除了的数据 |
| pop ( ) | 删除数组最后一个，返回删除了得数据 |
| 元素索引 | IndexOf( ) |  |
| lastIndexOf( ) |  |
| 转换数组 | toString( ) | 把数组转换成字符串，每一项用逗号“，”分割 |
| valueOf( ) | 返回数组对象本身 |

### String

|  |  |
| --- | --- |
| .trim( ) | 去除前后空格 |
| toUpperCase | 大写 |
| toLowerCase | 小写 |
| toString( ) | 返回字符串 |
| valueOf( ) | 返回字符串 |
| indexOf() | arr.indexOf(value)🡪 查询数组里面有没有这个值，有的返回索引，没有返回-1，从前往后找，只找一个 |
| lastIndexOf( ) | 查询数组里面有没有这个值，有的返回索引，没有返回-1，从后往前找，只找一个 |
| 截取字符串 | 返回的都是截取的字符串，不影响数组  相同：传入一个数，截取这个索引后面所有的字符，  不同: substring, 两个数都是索引值，包含前面数，不包含后面数  substr, 前面数是索引，后面的是长度，也是包含前面数 |
| Slice( ) |  |
| split( ) | 把字符串转换或者分割成数组，分割的时候必须传参，而且空格会转成内容为空的值 |
| replace( ) | 替换字符串的某元素，🡪 返回替换后的字符串 |
| charAt( index ) | 返回指定索引的字符； |
| charCodeAt ( ) | 返回 ASC II值 ，r = 114. |
| Concat( ) | 拼接字符串，等同于 + ， + 更常用 |
|  |  |

# Ajax

不管浏览器是否有缓存，都去服务器获取页面 🡪 location.reload(true);

1，禁止跨域是因为安全性

2，$get $post的区别

3，常用到的参数

4，缓存的原理

5，跨域和同源策略

### SEO

### http / https

### 性能优化

### 存储

### 兼容性

1， bom事件参数e，ie的是window.event，

2，获取目标元素，主流的是e.target，ie的是e.srcElement

3，ie的是attachEvent ，主流的是addEventListener

4，IE6里面的float之后有个margin的问题

5，获取标签之间的文本内容，

IE8及早期：obj.innerText

火狐早期 ：obj.textContent

### ajax01

一、五部曲

btn.onclick = function(){  
1.var abc = new XMLHttpRequest();

2.abc.open(); 请求方式，请求链接 ‘get’, ‘http://’

3.abc.send();发送数据，代码继、续往下执行，

4.abc.onreadystatuschange = function(){

1)到这一步的时候，响应回来的数据在这里进行处理，但是会出现数据并没有响应回来的情况，所以要判断后再执行

5.if( abc.readState == 4 && abc.status == 200){

满足条件在这里对数据进行处理; 数据存在 异步对象的属性里  
 --> abc.responseText;

}

}

}

二、请求方式

1.get请求

abc.open(‘get’, ‘http:// ?userName=’+variable);

2.post 请求

a).在发送内容之前，要改变请求报文的内容，🡪 固定值

ajax.setRequestHeader(**"Content-type"**,**"application/x-www-form-urlencoded"**);

b).

*// 数据 在send中 发送 发送的 格式为 key=value&key2=value2 字符串  
// name=luonaeduo*ajax.send(**'name=luonaeduo'**);

ajax.send(**'starName='**+name);

### ajax02

ajax请求(通过responseXML后去，设置header)XML解析(使用document方法)

字符串拼接( str += ‘<tr>’; str +=’</tr>’)

jquery选择器--上下文( $(‘star’,ajax.responseXML));

JSON字符串转化为js对象( JSON.parse)

JSON格式的优点( 多语言支持转化为该语言对象)，文件是对象以{ 开头，是数组以【开头，里面的格式和js对象的简便写法一样，键值对，要用“”包裹。

Ajax工具函数封装

Ajax从浏览器发出的：Ajax，php，XML，JSON

Php文件学会了获取get,post请求，关系型数组，读取文件，通过header方法输出一个XML

服务器响应回来的数据，在onreadystatechange处理，如果是XML，通过ajax.responseXML来处获取，本身是document对象，可以用querySelector。但是太麻烦，所以用jQuery：$( ‘star’,ajaxDocument)。

XML实际开发用的比例不高，所以用JSON。

在js里面可以把JSON格式的字符串转化成对象。在php里面也可以转化为php对象。

拿到JSON格式的字符串，在浏览器端如何处理。

再写Php：读取JSON并返回，用echo file\_get\_contents('03.json.json');

又回到html的onreadystatechange解析JSON：先JSON.parse(),如果是对象。。。

如果是数组，还要来一个for循环。

Ajax工具函数封装

1. ajax回调函数的作用

2. 命名空间封装自定义函数(var fox ={ajax\_tool:function(){}})

3. jQuery的基本ajax方法($.get,$.post,$.getJSON)

4. jQuery的.ajax方法使用

5. JQuery注册页面Demo

6. 模板引擎的工作原理(使用正则进行替换)

7. 第三方模板引擎使用(artTemplate)

Ajax五部曲、工具封装、测试、命名空间

正则、模板template、引用外部模板

Extend、给jq设置插件

用callback 不用return，

给jq写插件，要return this;

Extend : 延伸

·Php读取JSON ?

·模板只能解析对象，不能解析数组

·$.ajax({ 只支持传对象 })

1. 瀑布流\_服务端逻辑

2. 瀑布流\_算法

3. jQuery插件写法

4. 浏览器同源策略的作用(安全)

5. src属性支持跨域访问

6. jsonP的原理(通过get请求发送callBack到服务器,服务器拼接参数并返回)

7. jQ发送jsonp请求

8. 百度开放平台jsonP数据获取

不能在iframe中获取其他服务器网站的dom元素

不支持直接去其他服务器的网站中get请求或者post请求

拖拽：在标签里面添加，ondraggable=true

* 拖拽元素
* 文件读取(FileReader)
* sublime代码段
* 全屏幕(requestFullScreen)
* font-awesome使用
* js控制video标签
* 自定义播放器
* h5定位写法
* 百度地图基本使用
* form提交文件
* ajax2.0(文件上传,直接读取form值,上传进度)

|  |  |
| --- | --- |
| Aside 在旁边  Article 文章  Navigation 导航  Equal 相等的  Local 本地  Range 范围  Complete 完成  Focus 集中、焦点  Patlern 方式、行程  Methon 方式  Custorm 风俗、习惯  Validity 有效、合法性  Output 产量、输出  Audio 听觉的  Loop 循环、回路  Progress 进步、发展  Query 问题、疑问  Selector 挑战者、选择器  Toggle 切换  Attribute 属性  Radius 半径  Shadow 阴影、影子  Alpha 透明度  Radial 径向的、辐射状的  Gradient 倾斜度  Angle 角度  Scale 比列  Skew 歪斜  Shield 盾牌  Clip 修剪  Slice 切分  Cubic 立方体的  Bezier 贝塞尔曲线  Stretch 伸展、张开  *Source 来源、水源*  *Duration 持续*  *Direction 方向*  Forwards 向前、今后  Delay 延迟、延期 | Count 计算  Infinite 无限的  Pause 暂停  Perspective 远景、透视  Flex 曲折  Frame 框架  Animation 卡通、动漫  Fill 填充  Iteration 反复  Transparent 透明的  Align 使成一行  Justify 对齐  Items 项目  Shrink 收缩、退缩  Basis 底部、基础  Extra 特别地、临时演员  Current 现在的  Section 章节、地区  Search 搜索  Result 结果、成绩  Client 客户端  Browser 浏览器  Server 服务器  Explorer 探测者、浏览  Apache 阿帕奇  Echo 反射( 输出 )  Request 请求、需求  Response 响应、反应  Charset 字符集 |

Google chrome

|  |  |
| --- | --- |
| Elements :  Styles  Computed  Event listeners  DOM Breakpoints  Properties | Timeline  Profiles  Application  Security  Audits  Top  Preserve log |
| Console |
| Sources:  Content scripts  Snippets  Watch  Call Stack  Scope  Breakpoints  DOM Breakpoints  XHR Breakpoints  Event Listener Breakpoints  Event Listeners |
| Network :  View  Preserve log  Disable cache  Offine  No throttling  Regex  Hide data URLs  All  XHR  JS  CSS  Img  Media  Font  Doc  Ws  Manifest  Other |

# Angular.js

## linux

1. 文件夹 的增删改查

* mkdir file – 创建文件夹
* rm –rf file – 删除文件夹
* ls – 查看当前文件夹
* touch .gitignore – 创建文件
* rm .gitignore – 删除文件
* git add . – 工作区到暂存区
* git commin –m ‘’ – 暂存区到
* git status – 查看修改信息
* git log – 查看所有提交
* git reflog – 简化
* git reset –-hard 67c0ef – 跳转指定点

1. 创建分支

git brach test – 创建分支

git brach – 查看分支

git checkout master – 切换到master

git merge – 合并分支

git branch –help – 命令行寻求帮助

git branch -d -- 删除分支

git checkout –b newTest -- 创建分支，同时也跳转该分支

1. SSH -- https:
   1. 项目地址：
   2. 码云
      * 1. cmd: ssh-keygen -t rsa
        2. 公钥： id.pub
        3. -- 上传到码云
2. 忽略文件 – 冲突
   1. .gitignore

04day -- tortoisegit

|  |  |
| --- | --- |
| ng-app |  |
| ng-controller |  |
| ng-model |  |
| ng-bind |  |
| ng-repeat |  |
| ngclass |  |
| ng-if |  |
| ng-hide |  |
| ng-show |  |
| ng-switch |  |
| ng-src |  |
| ng-href |  |
| ng-focus |  |
| ng-blur |  |
| $scope | $log $window $location $location.port( ) |
| $watch |  |
| $filter |  |
| $http |  |
| rootScope |  |
| App.service | 里面用到了this，被当做构造函数来调用 |
| App.factory | 当做普通函数来调用，返回的是一个对象 |
| directive |  |
| Angular.merge( ) | 用来合并数据 |

下载angular：npm I angular

下载angular-route： npm I angular-route

## localStroage

json数据就是对象，能够传输的原因是把内存中的对象变成字符串。

怎么把内存里的对象转化成字符串呢？ 🡪 通过JSON.stringify( obj )传输到后台

后台再怎么操作字符串，把字符串转化成对象？ 🡪 JSON.parse(JSON.stringify( obj ) );

localStorage.clear( ) 🡪 清理方法

localStorage.getItem ( ‘ BASE‘) 🡪 返回的是字符串形式的数组包裹着对象

也可以往里面写：localStorage.setItem(‘todoMvc’,’123’)

存的是数组，传的是对象

## todo

来源：fiftyDay，todomvc.com

模板：<https://github.com/tastejs/todomvc> -- 下面的模板

1.要安装npm，

2.直接git clone下载，上面的地址就行

3.没有css，当前文件夹 $ npm install 把样式下载，

4.删掉.git ，因为项目不需要管理

5.打开报错，左边的文件没有引入，注释掉就可以—base.js文件，

### Douban

1-location 🡪 可以拿到url，

2-监视锚点值，通过监视锚点值得不同来切换数据

$wtach只能监视$scope里面的值。 angular内部有个$location

3-字符串切割：str.substring( 1 ) 🡪 去掉前面的/active

4-好处：用户体验好，减少数据量的发送 🡪 主要体现在移动设备上

meta标签里的文字，百度抓取的时候，只能抓取静态的。但是angular网页里面的文字是通过表达式来渲染的

坏处：不利于SEO的优化，代码量增加

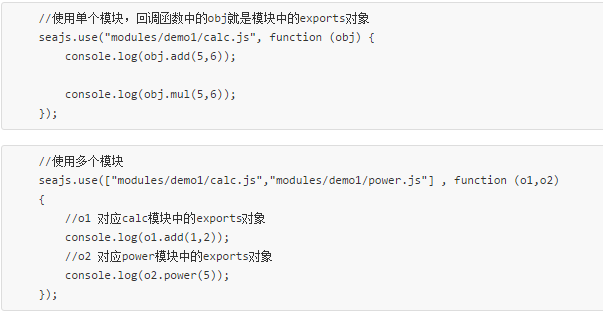
# Node.js

## 模块化的演变过程

* 1. 非模块化带来的问题：命名冲突，文件依赖
  2. 全局函数
     + 1. 问题：命名可能会冲突（变量污染）
  3. 对象封装
     + 1. 解决：解决命名冲突（类似命名空间的方式）
       2. 新问题：没有自己的私人空间（不能设置私有成员）
       3. 代码
  4. 隔离私有空间
     + 1. 解决：不能设置私有成员的问题
       2. 新问题：如何扩展一个功能
       3. 代码
  5. 模块的扩展
     + 1. 对扩展开放，对修改封闭
       2. 解决：扩展一个求平方的功能
       3. 代码
  6. 依赖第三方模块
     + 1. 封装到js文件中
       2. 代码
  7. 使用模块化开发的价值
     + 1. http:// [github.com/seajs/seajs/issues/547](https://github.com/seajs/seajs/issues/547)

## 模块化开发框架sea.js

1. 使用sea.js的步骤
   1. 导入seajs
   2. define()定义个模块，define的回调的三个参数名字不可更改
   3. define( function (require, exports, module) { } );
   4. exports或者module.exports将模块中的成员暴露出来
   5. seajs.use使用一个模块
   6. config()配置一些基本信息（别名、路径）
2. seajs.use使用模块



1. 导出模块中的成员
   1. 使用exports导出成员
      * 1. 代码
   2. 使用module.exports导出
      * 1. 代码
   3. exports和module.exports的关系
      * 1. exports = module.exports = { }
        2. 不能为exports直接赋值，但是可以动态添加成员
2. 模块之间的依赖
   1. require()引用另一个模块
      * 1. 代码
3. config
   1. config的作用就是为了简化调用模块
      * 1. base设置路径
        2. alias设置模块的别名，简化调用
        3. 代码
   2. seajs的原理



## 模块化规范

1. CMD规范
   1. seajs遵守CMD规范
   2. CMD规范地址
   3. CMD规范的特点，所有的require()都是懒加载，用到的时候才去加载，提升初始化时加载的性能
2. AMD规范
   1. require遵守AMD规范，requireJS官网：<http://www.requirejs.org/>
   2. AMD规范的地址
   3. AMD规范的特点，所有的require()都是一个预加载模式

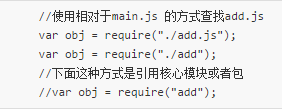
## 路径问题

1. seajs中使用相对路径
   1. demo和 ./demo在seajs中稍有不同
   2. demo 一般表示相对于当前引入的sea.js文件的位置
   3. ./demo 一般表示相对当前编写的文件的位置

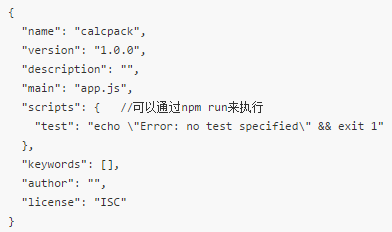
## Node.js的模块化

1.模块化的简介

* 1. 在后台开发语言中，java,C#隐含模块化，node.js默认帮我们提供了模块化的机制。在服务端，我们想要使用一些底层的功能需要导入一些“包”来对其操作，比如操作文件、网络。node.js使用的是js语言，ECMAScript仅仅规定了基本的语法书写，并没有规定文件之间的关联，也就是说每个js文件之间是独立的，node.js帮我们实现了js文件之间的关联（模块化）。
  2. node.js中的模块化是基于CommonJS规范的

1. javascript的局限性
   1. 没有模块系统
   2. 系统提供的接口少，缺少操作文件、I/O流等常用接口
   3. 没有标准接口，缺少Web服务器、数据库等统一接口
   4. 缺少管理系统导师，JS应用中基本没有自动加载和安装依赖的能力
2. CommonJS规范
   1. Node.js开发之初遵守了CommonJS的规范
   2. 使得JS达到想java,Python,PHP等语言开发大型应用的基本能力
   3. CommonJS规定每一个模块都有一个单独的作用域
   4. CommonJS规定每个模块对外公布的成员使用module.exports或者exports
   5. Node.js提供了许多模块：文件，Buffer,I/O流，Socket等
3. Node.js的核心模块
   1. 使用核心模块之前首先要导入模块
   2. path模块
      * 1. 导入模块 var path = require(“path”);
        2. basename() 获取文件名+后缀
        3. dirname() 获取目录
        4. extname() 获取文件的扩展名
        5. join() 合并路径
        6. parse() 把路径转换为一个对象
        7. format() 把一个路径对象转换成一个路径字符串
        8. delimiter 环境变量的分隔符，可以跨平台windows下是“；”，其他平台“：”
        9. path.sep 路径的分隔符 windows平台是“\”,其他下是‘/’。
        10. isAbsolute() 是否是绝对路径
4. url模块
   1. 导入模块 var url = require(“url”)
   2. parse() 把字符串路径转换成对象
   3. format() 把路径对象转换成字符串
5. querystring 模块
   1. 导入模块 var querystring = require(“querystring”);
   2. parse() 把参数字符串解析成对象
   3. stringfy() 把一个对象转换成一个字符串
   4. escape() url进行编码
   5. unescape() url进行解码
6. 核心模块存在哪里
   1. 核心模块存在node.exe中，当node.exe运行的时候，核心模块会被加载，require的时候会加载到内存
   2. 在github上可以找到源码，lib文件夹下
   3. 核心模块的执行速度比较快
7. 自定义模块
   1. 自定义文件模块—-add.js
   2. 只用文件模块—- main.js
      * 1. 注意引用js的方式和核心模块不同
        2. 

1. 包
   1. CommonJS的包规范给程序员提供了组织模块的标准，减少沟通成本
   2. 包的使用：
      * 1. 所有模块放在一个文件夹（包名）
        2. 包放在当前项目中的node\_modules文件夹下
        3. 包中定义一个index.js(文件名不可以更改)导出所有模块
        4. 引用包（约定大于配置）
   3. 导入包的执行过程require(“calc”)
      * 1. 将calc当作核心模块加载，加载不成功
        2. 自动去当前目录中的node\_modules中找文件名为clac的包
        3. 自动去calc找index.js的出口模块（导出的模块）
        4. 如果找不到index.js报错，如果想要改出口模块，需要package.json配置文件
   4. package.js的名称 以及 功能
      * 1. name : 包名称
        2. description : 介绍包的功能
        3. version : 版本号，用于控制版本
        4. keywords : 关键词数组，用于在npm中搜索
        5. main : require引入包时优先检查此字段
        6. dependencies : 标记当前包所依赖的包列表，npm会自动加载依赖的包
        7. Author : 包作者
        8. License : 开源许可



* 1. 创建包的标准方式
     + 1. npm init –y 自动创建 package.json
  2. 一个标准包的结构
     + 1. 名称、功能、--、--、package.js、包描述文件bin、存放可执行文件lib、存放JavaScript代码doc、存放文档test、存放单元测试用例代码README.md、说明文档，描述包的作用和用法
  3. 标准包的执行过程
     + 1. 将calcpack当作核心模块加载，加载不成
       2. 自动去当前目录中的node\_modules中找文件名calcpack的包
       3. 如果在clacpack中有package.json的话，并且指定了main属性的值，优先加载main指定的.js模块（出口模块）
       4. 如果没有package.json，或没有指定main属性，自动去calcpack找index.js的出口模块（导出的模块）
       5. 如果找不到index.js就会报错

1. 发布包
   1. 把包发布到NPM官网<https://www.npmjs.com/>
      * 1. 建立一个包，设置package.json
        2. 在npmjs中注册账号
        3. 在包的根目录下执行
           1. npm adduser添加发布包的用户信息，登录网站
           2. npm puhlish发布或者更新包package.json中一定要指定maintainers:[{

“name”: “nllcode”,

“email”: “xxxx@qq.com” }]

* + - * 1. npm cache clear 清楚npm本地缓存，用于对使用相同版本号发布新版本
        2. npm unpublish@ 删除发布过的版本代码 npm unpublish [haha@1.0.0](mailto:haha@1.0.0)
      1. 错误
         1. only admin can publish this module
         2. 解决： 修改原npm config set registry
      2. 安装包
         1. 从网络安装

当前目录安装npm install包名

全局安装npm install包名 –g

* + - * 1. 从本地安装 npm install包的路径
        2. 卸载包npm uninstall包名
      1. require() 加载规则
         1. 优先从缓存加载模块或者包
         2. 加载恩健模块要使用相对路径./../
         3. 文件模块的加载可以不写后缀名，如果不写后缀名按照.js>.node>.json的顺序加载
         4. 加载json文件，推荐写上后缀.json
         5. 加载核心模块或包，不写路径和后缀
         6. module.paths 加载node\_modules的时候，按此数组的顺序加载

1. Node.js的调试
   1. console.log()
   2. node.js自带的调试器
      * 1. node debug a.js
        2. 常用命令
           1. help查看可用命令
           2. n 进入下一步
           3. s 单步进入
           4. o 步出
        3. Webstorm中调试

# H5C3

## First

1; header > nav; main > article + aside; footer

2; 用js创建header标签，追加到界面，还要display: block;

hack写法，用注释的方式就行，用js创建虽然可以，但是会出现2个方块

3; H5新语意标签的兼容性，可以使用js插件解决，不用hack写法，

html5shiv.min.js--> github查询

4；input的新type属性

form > fieldset > legend文字 > label > input

input: color, date, month, week, time, number, range, email, search, tel

pc端看不出效果，要去手机端看。

6; input的新属性，直接写在标签里就行，没有值。

input: text, file, submit,

autofocus: 打开就获得焦点

autocomplete: 自动完成，必须有name属性，还要提交过一次，下次才有提示

submit: 提交按钮，获取form标签里面的input值一起get提交，如果在form标签之外的就获取不到，但是非要获取到的话，这个input标签，就要有一个form属性，而且值和form的id 一样。（ 提交的都必须有name属性 ）

7; 点击input输入框前面的文字就让input获取焦点，前面的文字用label标签包裹，而且还要有个for属性，而且值和input的id 是一样的

15;form的新属性：

datalist标签可以为input标签增加选项，前提是id和input的list属性值保持一致，output和span一样没有任何功能，就是语意增强，配合用来显示结果

5; 表单的验证，非空验证，以及自定义文字验证，但是不能改变样式。

required, required pattern = ‘[0-9]{11}’,

oninvalis事件，this.setCustomValidity(‘’)

8; video标签：src, width，height

conrols: 控制栏

loop: 循环播放

autoplay: 自动播放

poster: 设置封面

9; audio: 没有width, height属性

10; 获取dom元素：document.querySelector(‘’)

11;data- 自定义属性，切换明星，用querySelector，获取页面元素更换bgc和innerText

12; 进度条-

progress:

meter: 这个可以改变颜色

input type = ‘range’, 用oninput绑定事件，本身有value值，

13;自定义进度条 - 用div包裹progress标签，还是用oninput绑定事件，

14;H5 操作class:

function 要传e

classList, add增加，remove移除，toggle切换，

## Second

1; transtion: all 1s ; 改变这个元素的属性值的时候，会有过渡的效果

2; 兄弟选择器：

3; 属性选择器：

li[ skill ] – 所有拥有skill属性的值不考虑

li[skill=fire] – 某个具体值

li[skill\*=it] – 包含了某个值

li[ skill^=sell], li[friend$=s], 开头和结尾

li[people|=dog] – ‘|=’查找的是属性以- 进行分割，第一个值是dog的标签

4; 伪类选择器：

first-child, last-child, nth-child(7), nth-child(2n), li:nth-child(odd)

li:nth-child(7n-1),

.container p:nth-child(3), 索引指的所有子元素，并非仅仅P元素，而且从1开始

li:not([price]), 找到没有price属性的li标签。

li:not([price = ‘18’]), 找到price不等于18的所有li标签

锚连接 h2:target{ 设置属性} – 跳转过去后，h2有属性变化

5; 伪元素选择器：

可以为双标签添加子元素，div:before, div:after

div有relative, 子元素必须要有content属性，可以为空。默认是行内元素

xiaomiDemo: 配合transtion使用，div:hover**::**before{ }

p::first-letter, p::first-line, 对首字母和首行设置样式

input 输入框在获得焦点以后，有边框阴影

6; 边框阴影： transtion: all 1s;

box-shadow: 0 0 0 10px red,

0 0 0 20px blue;

7; 文字阴影： transtion: all 1s;

p:hover { text-shadow: 30px 60px 20px pink; }

1.x轴方向移动距离，2.y轴方向移动距离， 3.阴影模糊的程度，4.阴影的颜色

8; 文字3D

9; 颜色的设置：hsla( )

10; 透明度：background-color: transparent;

11;安卓机器人：transform: rotate(30deg)

12;边框圆角：border-radius

13;盒子模型：

box-sizing: border-box;

14;div:hover{ transform: translate(x,y), rotate(0deg), scale(x,y), skewX(0deg) }

## Third

# ionic

## First

## Second

## Third

indxedDB

数据的增删改查，-通过事务获取表，用事务包裹一层

游标:

浏览器兼容性，查询大量数据，上百M的存储量

## Fourth

中间件：

面向对象的框架：把重复的东西抽离出来

封装indxedDB

购物车功能实现

ping++ 支付平台

TortoiseGit 介绍