# JavaScript

目录

[JavaScript 1](#_Toc492033267)

[ JavaScript 基础部分（ES） 2](#_Toc492033268)

[1、 函数中断退出 2](#_Toc492033269)

[2、 常用事件： 2](#_Toc492033270)

[3、 a 元素阻止默认事件： 2](#_Toc492033271)

[4、 冒泡单击事件的时机(加载顺序) 2](#_Toc492033272)

[5、 转义字符的另一种简单方法 3](#_Toc492033273)

[6、 动态生成页面元素的内容（数组动态拼接） 3](#_Toc492033274)

[7、 Math的方法 4](#_Toc492033275)

[8、 生成随机颜色： 5](#_Toc492033276)

[9、 关联数组 6](#_Toc492033277)

[10、 数组API，slice、splice，join、concat 6](#_Toc492033278)

[11、 String: indexOf/search/match/替换replace/切割split 8](#_Toc492033279)

[12、 栈、队列 push pop (js把十进制数转二进制) 10](#_Toc492033280)

[13、 正则表达式及验证 11](#_Toc492033281)

[14、 计算程序运行时间：循环计时 14](#_Toc492033282)

[ BOM和DOM部分 14](#_Toc492033283)

[15、 弹出的窗口 14](#_Toc492033284)

[16、 7、窗口大小 15](#_Toc492033285)

[17、 鼠标指针位置和页面滚动位置 15](#_Toc492033286)

[18、 4、获得元素距离页面顶部的距离(offset.top) 15](#_Toc492033287)

[ jQuery部分 16](#_Toc492033288)

[19、 获得元素名称： 16](#_Toc492033289)

[20、 父元素增加子元素/添加子元素-prepend和append 16](#_Toc492033290)

[21、 弹出错误/自定义插件时判断是否引入jq文件 17](#_Toc492033291)

[22、 jQuery事件处理的另一种写法： 17](#_Toc492033292)

[23、 jq 按钮只能点一次（事件只处理一次） 17](#_Toc492033293)

[24、 jQuery自带循环获取数组中每一个对象 17](#_Toc492033294)

[25、 jq判断checkbox是否被选择的三种方法 18](#_Toc492033295)

## JavaScript 基础部分（ES）

### 函数中断退出

btn.onclick=function(){

if(b>3){return;}; //如果b>3则退出函数执行，即style.color=“red”不执行

style.color=”red”

}

### 常用事件：

1. onclick 单击
2. ondblclick 双击
3. onfocus 获得焦点
4. onblur 失去焦点；
5. onchange select 当值发生改变时
6. onmouseover/out/enter/leave/hover/move/down/up

onmouseenter/leave ->(jq)进出子元素不会反复出发父元素事件处理函数

onmouseenter/leave -》jq简写🡪hover

1. onkewdown/up 键盘敲击
2. window.Onload 页面所有东西加载完成；注意：一个页面只能执行一次onload
3. Domcontentloaded 仅DOM内容(html和js)加载完成——提前
   * + 如何: $(document).ready(function(){ ... ... })
     + 可简写为: $().ready(function(){ ... ... })
     + 还可简写: $(function(){ ... ... })
     + ES6: $(()=>{ ... ... })
     + 放在body结尾的script中的代码，默认就是DOM内容加载后执行.
     + 注意：一个页面可以增加多个ready
4. **jQuery常用绑定事件处理的另一种写法：**

**$(“.nav”).on(mouseenter,”a”,function(e){})**  //”.nav”是“a”的父元素；“a”是“.nav”的子元素，函数内可以不用e.target的冒泡判断=>if($(e.target).is(“a”)){…}

### a 元素阻止默认事件和取消冒泡停止蔓延

1、e.prenventDefault();

2、e.stopPropagation();

### 冒泡单击事件的时机(加载顺序)

<**body**>  
 <**h1**>添加/删除节点</**h1**>  
 <**button id="add-block"**>添加区块</**button**>  
 <**div class="container"**></**div**>  
 <**script src="js/jquery-1.11.3.js"**></**script**>  
 <**script**>  
 **var *$container***=**$**(**".container"**);  
 **$**(**"#add-block"**).click(()=>{  
 **var *$block***=**$**(`<div **class**=**"block"**><span **class**=**"close"**>×</span></div>`);  
 *//修改$block的背景色* **var *r***= parseInt(**Math**.random()\*256);  
 **var *g***= parseInt(**Math**.random()\*256);  
 **var *b***= parseInt(**Math**.random()\*256);  
 ***$block***.css(**"background"**, `rgb(${***r***},${***g***},${***b***})`);  
 ***$container***.append(***$block***).prepend(***$block***);

//给子元素.close添加点击事件，如果父元素container内容没有创建则无法添加单机事件

*//$(".container>.block .close").click(e=>{$(e.target).parent().remove()})* });

//最好在父元素.container绑定单机事件 ，冒泡到每个子元素.close

***$container***.delegate(**".close"**,**"click"**,**function**(){**$**(**this**).parent().remove();});  
 </**script**>  
 </**body**>

### 转义字符的另一种简单方法

var pathname=String.raw`c:\windows\temp\new\folder\index.html`

console.log(pathname); // c:\windows\temp\new\folder\index.html

raw:原生

### 动态生成页面元素的内容（数组动态拼接）

**id.innerHTML=**

**"<ANY>"+arr.join("</ANY><ANY>")+"</ANY>"**

//从服务器端收到省份列表数组

var provs= ["北京市","河北省","河南省","山东省","陕西省"];

var html="<option>-请选择-</option><option>"+

provs.join("</option><option>") +"</option>";

//设置selProvs元素的内容为html

selProvs.innerHTML=html;

### Math的方法

1. Math.ceil(num) //上取整: 只要超过，就取下一个整数;
2. Math.floor(num) //下取整: 舍弃小数部分

vs parseInt(str): 1. parseInt可去掉结尾非数字字符floor不行！

2. parseInt要求传入字符串,floor要求数字

1. Math.round(num) // 四舍五入取整

**vs .toFixed(d):** 1. round只能取整, toFixed 可按任意小数位数四舍五入

2. round返回number，可直接用于计算toFixed 返回string，不能直接用于+计算

1. 自定义round(转换任意小数位数四舍五入)

乘方和开平方:

乘方: Math.pow(底数,幂); 比如: Math.pow(2,3)=8

开平方: Math.sqrt(num)

**例子：**

function round(num,d){

//num\* 10的d次方

num\*=Math.pow(10,d);

//对num四舍五入取整

num=Math.round(num);

//num/ 10的d次方

num/=Math.pow(10,d);

return num;//返回num

}

var n=345.678;

console.log(

round(n,2), //345.68

round(n,0), //346

round(n,-1) //350

);

1. Math.max(值1,值2,...)
2. Math.min(值1,值2,...) //max和min不支持查找数组中的最大值/最小值
3. Math.max.apply(null,arr)// apply可打散数组类型的参数为单个值，再传递给调用国的函数
4. **随机数:**

1. Math.random(): 单独使用返回0~1之间的一个小数 // var r=Math.random();

2. 在min~max之间取随机整数:

parseInt(Math.random()\*(max-min+1)+min)

var r=parseInt(Math.random()\*33+1); //1~33之间生成随机整数r

3. 如果从0~max之间取随机整数:

parseInt(Math.random()\*(max +1))

### 日期Date();

* + **javascript  时间戳 转换成24 小时格式**

**（默认是12小时格式：2017/9/1 下午6:16:43）**

var timeTrans = new Date(parseInt(1459408942) \* 1000);

console.log(timeTrans.toLocaleString('chinese',{hour12:false}));

*// var d=new Date(1504261003000).toLocaleString();* **var *d***=**new** Date(1504261003000).toLocaleString(**'chines'**,{**hour12**:**false**});*// 2017/9/1 18:16:43* **console**.log(***d***)

* + **php  时间戳 24 小时转换**

date("Y/m/d H:i:s",1446634507 ) //2015/11/04 18:55:77

### 生成随机颜色：

**function** *rc*(){  
 **var** r=**Math**.floor(**Math**.random()\*256);  
 **var** g=**Math**.floor(**Math**.random()\*256);  
 **var** b=**Math**.floor(**Math**.random()\*256);  
 **return** `rgb(${r},${g},${b})`; **//关联**  
 }  
 **console**.log(*rc*()) //**rgb(91,103,187)  
$**.ajax({  
 **type**:**"GET"**,  
 **url**:**"/salesdata"**,  
 success:**function**(data){  
 **var** ctx=c2.getContext(**"2d"**);  
 **var** x=50;  
 **for**(**var** obj of data){  
 **var** str=obj.**label**;  
 ctx.**fillStyle**=*rc*();**//关联** ctx.fillRect(x,500,50,-obj.value);  
 ctx.**font**=**"20px SimHei"**;  
 ctx.**fillStyle**=**"black"**;  
 ctx.fillText(str,x,500-obj.value-20);  
 x+=100;  
 }  
 },  
 error:**function**(){  
 alert(**"网络故障，请检查"**);  
 }  
})  
</**script**>

### 关联数组

什么是: 可自定义下标名称的数组

强调: length=0，失效

访问元素: 数组名["自定义下标名称"]

**遍历:** for(var key in hash){//in会依次取出hash中每个key

key //下标名称

hash[key] //当前正在遍历的元素内容

}

强调: hash[key] 不要加""， 因为key是个变量，每循环依次都会变化

var fbb=[];

fbb["name"]="范冰冰";//杨幂

fbb["math"]=89; //81

fbb["chs"]=65; //59

fbb["eng"]=95; //89

console.dir(fbb);

console.log(fbb["chs"]);

fbb["chs"]+=10;

console.log(fbb["chs"]);

for(var key in fbb){

console.log(key+":"+fbb[key]);

}

### 数组API，slice、splice，join、concat

* **转字符串：var str=String(arr); 将arr的每个元素都转为字符串，用逗号相连.（拍照）**
* **var str=arr.join("自定义的连接符")**

*//无缝拼接***var *chars***=[**"H"**,**"e"**,**"l"**,**"l"**,**"o"**];  
***console***.log(***chars***.join(**""**)); //Hello

* + - **动态生成页面元素的内容**

**id.innerHTML=**

**"<ANY>"+arr.join("</ANY><ANY>")+"</ANY>"**

//从服务器端收到省份列表数组

var provs= ["北京市","河北省","河南省","山东省","陕西省"];

var html="<option>-请选择-</option><option>"+

provs.join("</option><option>") +"</option>";

//设置selProvs元素的内容为html

selProvs.innerHTML=html;

* **拼接: 将其他数组或元素，拼接到当前数组的末尾，组成新数组**

var arr3=arr1.concat(值1,值2,...,arr2,...)

**var *arr1***=[1,2,3];  
**var *arr2***=[12,23,34];  
**var *arr3***=***arr1***.concat(4,5,***arr2***,6);  
***console***.log(String(***arr1***));  
***console***.log(***arr3***); // [1, 2, 3, 4, 5, 12, 23, 34, 6]  
***console***.dir(***arr3***);

* **选取: 复制出原数组中指定位置的几个元素值，组成新数组.**

**var sub=arr.slice(starti,endi+1);**

强调: 凡是两个参数都是下标位置的函数，都含头不含尾.

1. 可用负数参数直接代表倒数第n个位置:

arr.slice(starti,-n+1); 截取到倒数第n个位置

原理: arr.slice(starti,arr.length-n+1);

2. 省略第二个参数，表示到结尾！

3. 两个参数都省: 表示从头到尾完整复制整个数组

**var *arr***=[0,1,2,3,4,5,6];  
 *//0 1 2 3 4 5 6  
 // -3 2 1***var *sub***=*//arr.slice(2,4+1);  
 //倒数第3个  
 //arr.slice(2,arr.length-3+1);* ***arr***.slice(2,-3+1);  
 **var *c***=***arr***.slice(2,5)  
***console***.log(String(***arr***)); //0,1,2,3,4,5,6  
***console***.log(String(***sub***));// 2,3,4  
***sub***=***arr***.slice(3); ***console***.log(String(***sub***)); //3,4,5,6

* + **修改数组: 插入，删除，替换**

强调: 直接修改原数组

1. 删除: arr.splice(starti,n); 删除arr中starti开始的n个元素

强调: 不考虑含头不含尾,因为第二个参数不是下标

简写: 1. 省略第二个参数: 一直删到结尾

2. 支持负数参数: 表示倒数第n个位置

其实splice有返回值: 返回被删除元素组成的临时新数组

var deletes=arr.splice(starti,n);

强调: 即使仅删除一个元素，也会放在数组中返回。

2. 插入: arr.splice(starti,0,值1,值2,...)

原插入位置及其之后的元素，依次向后顺移

vs concat: 1. splice可在任意位置插入新元素,concat只能在末尾拼接

2. splice直接修改原数组，concat不修改原数组，返回新数组

3. splice不支持打散数组参数，concat 支持大散数组参数为单个元素

**var *arr***=[0,1,2,3];  
**var *arr2***=[12,23,34];  
***arr***.splice(2,0,4,5,***arr2***,6);  
***console***.log(String(***arr***)); // 0,1,4,5,12,23,34,6,2,3  
***console***.dir(***arr***);

3. 替换: 先删除元素，再在原位置插入新元素

arr.splice(starti,n,值1,值2,...); 先删除n个，再插入新值

强调: 删除的元素个数和插入的新元素个数不必相同！

**var *arr***=[0,1,2,3,4,5];  
*//先删除2个元素，再插入3个****arr***.splice(2,2,12,23,34);  
***console***.log(String(***arr***));// 0,1,12,23,34,4,5

* + **翻转: arr.reverse();**
  + **排序(原生使用冒泡排序)**

**arr.sort();**

arr.sort(function(a,b){return a-b;})

//升序改降序: 只要颠倒比较器函数返回值的正负号

### String: indexOf/search/match/替换replace/切割split

* + **大小写转换：**

str.toUpperCase() 转大写

str.toLowerCase() 转小写

* + **获得指定位置的字符:**

str.charAt(i) -> str[i]

获得指定位置字符的unicode号:

str.charCodeAt(i) 获得i位置字符的unicode号

将一个unicode号，反向转回字:

String.fromCharCode(code)

选取指定位置的子字符串:

str.substring() => str.slice()

唯一差别: substring不支持负数参数

解决: 可用str.length-n，表示倒数第n的位置

从当前位置开始，取n个

str.substr(starti,n)

* + **查找关键词:**

1. 查找一个固定的关键词的位置: （每次只能找一个）

var i=str.indexOf("关键词",fromi)

简写: 省略fromi，默认从头开始找第一个关键词

**返回值:** 返回找到的下一个关键词的下标位置i

如果没找到，返回-1

**在str中找最后一个关键词的位置**

var i=str.lastIndexOf("关键词")

2. 判断是否包含某类关键词:

var i=str.search(/正则表达式/i);

在str中找第一个符合正则表达式要求的关键词的位置

如果没找到，返回-1

问题: 默认，所有正则区分大小写！

解决: 如果希望不区分大小写时，在第二个/后加i

3. 获得所有关键词的内容:

var kwords=str.match(/正则表达式/ig);

在str中找所有符合正则表达式要求的敏感词，保存在数组中返回。

如果没找到，返回null!

问题: 默认，正则表达式都只匹配第一个关键词

解决: 只要希望匹配所有，都必须加g

问题: 只能返回关键词内容，无法返回每个关键词位置

**即找所有关键词内容，又找所有关键词位置: (用正则：var arr=reg.exec(str);**

**方法)**

**var *str***=**"明明喜欢我，却不告诉我。别理我，我想静静。静静是谁?你先告诉我明明是谁？"**;  
**var *kwords***=***str***.match(/明明|静静/g);  
***console***.log(String(***kwords***)); //明明,静静,静静,明明

* + **替换：**

1. 简单替换: 将所有敏感词，都替换为统一的新内容

str=str.replace(/正则/ig,"新值");

强调: 1. 必须加g, 替换所有

2. 必须用=保存回去，否则无法获得替换结果

2. 高级替换: 根据本次找到的敏感词不同，动态选择不同的新内容替换。

str=str.replace(/正则/ig,function(kw){

kw //自动获得本次找到的关键词内容

return //根据kw的不同选择替换不同的值

})

<**script**>  
 **var *str***=**"那天，我去了她家。我说: 我草，你家真大！她说想和我去草地野餐。我草草的收拾了草地。她说：我去去就来"**;  
 *//我去，我草，我去草，我草草，我去去* ***str***=***str***.replace(  
 /我[去草]{1,2}/g,  
 **function**(kw){  
 **return** kw.**length**==2?**"\*\*"**:**"\*\*\*"**;  
 }  
 );  
 ***console***.log(***str***);  
</**script**>

* + **切割split：**

1. 简单切割: 按一个固定的字符/字符串切割

var subs=str.split("切割符")

固定套路: 将字符串打散为字符数组: var chars=str.split("");

将来如果希望对字符串，使用数组API时，都要先打散。

2. 复杂切割: 切割符不是固定的，但有一定规律

var subs=str.split(/正则表达式/i);

**var *url***=**"http://item.taobao.com/item.html?a=1&c=2&d=xxx&e"**;  
**function** *getData*(url){  
 **var** result={};  
 **var** temp=url.split(**"?"**)[1].split(**"&"**);  
 **console**.log(temp);  
 **for** (**var** i **in** temp){  
 **var** s=temp[i].split(**"="**);  
 **console**.log(s[1]);  
 result[s[0]]=s[1];*//xx[]为对象属性添加值* }  
 **return** result;  
}  
**console**.dir(*getData*(***url***))

### 栈、队列 push pop (js把十进制数转二进制)

* + **栈(stack):**

什么是: 一端封闭，只能从另一端进出的数组

1. 开头封闭, 结尾出入栈:

入栈: arr.push(值) 简化了arr[arr.length]=值

出栈: var last=arr.pop();

2. 结尾封闭，开头出入栈:

入栈: arr.unshift(值)

强调: 开头入栈的顺序和结尾入栈的结果相反！

何时: 只要希望获得与结尾入栈相反的元素顺序时

出栈: var first=arr.shift();

* + 队列(queue):

什么是: 只能从结尾进入，从开头出的数组

特点: FIFO

何时: 只要希望按先来后到的顺序使用数组元素时

如何:

结尾入队列: arr.push(值)

开头出队列: var first=arr.shift();

**var *n***=90;  
*//定义函数dec2bin,接收一个十进制数dec***function** *dec2bin*(dec){  
 *//定义空数组bin* **var** bin=[];  
 *//反复* **do**{  
 *//将dec%2的余数压入bin中* bin.unshift(dec%2);  
 *//将dec/2再取整，再保存回dec中* dec=*parseInt*(dec/2);  
 *//只要(dec>0),就反复执行* }**while**(dec>0)  
 *//将bin翻转(怎样就不需要翻转？)  
 //返回bin无缝拼接的结果* **return** bin  
}  
*dec2bin*(***n***)  
***console***.log(*dec2bin*(***n***), ***n***.toString(2));

### 正则表达式及验证

* 1. /关键词/ 关键词本身
  2. 字符集

1位数字 [0-9]

1位大写字母[A-Z]

1位小写字母[a-z]

1位字母[A-Za-z]

1位字母或数字[A-Za-z0-9]

1位汉字[\u4e00-\u9fa5]

* 1. 预定于字符集：

\d 一位数字

\w 一位字母/数字或\_

\s 一位空字符: 包括空格，Tab...

. 一位任意字符——通配符

强调: 规则必须和预定义字符集的规则完全匹配时，才能使用预定义字符集。否则，使用普通字符集。

* 1. 量词

**1. 有明确数量边界: 3组**

字符集{m,n} 至少m次，最多n次

字符集{m,} 至少m次，多了不限！

字符集{m} 必须m次

**2. 没有明确边界: 3个**

字符集\* 可有可无，多了不限

字符集+ 至少一个，多个不限

字符集? 可有可无，最多一个

问题: 默认，一个量词只修改相邻的前一个字符集

* 1. 选择和分组:

选择: 或 | 规则1|规则2 只要匹配任意一个规则即可

分组: (多个规则) 将多个规则分为一组

为什么: 一个量词默认值修改一个字符集

何时: 只要希望一个量词修改多个字符集时

* 1. 指定匹配位置:

开头: ^ 比如: ^\s+ 匹配开头位置的空字符

结尾: $ 比如: \s+$ 匹配结束位置的空字符

单词边界: \b 包括: ^ $ 空格 标点符号

何时: 只要要求匹配一个单词时

**例子：**

**身份证号:** 15位数字 2位数字 1位数字或Xx 可有可无,最多一次

\d{15}(\d\d[0-9Xx])?

手机号: +86或0086 可有可无，最多一次空字符 可有可无，多了不限

1在3,4,5,7,8中选一个9位数字

(\+86|0086)?\s\*1[34578]\d{9}

微 信 wei xin w x

(微|w(ei)?)\s\*(信|x(in)?)

**邮箱：**var unReg=/^[a-zA-Z0-9]{3,18}[@]{1}[a-z0-9.]{5,18}$/ig;

* **验证: 正则表达式对象RegExp:**

创建: 2种:

1. 创建一个固定不变的正则表达式:

var reg=/正则表达式/ig;

字符冲突: 如果正则中出现/，要转为\/

2. 如果需要动态生成正则表达式:

var reg=new RegExp("正则",["ig"]);

字符冲突: 如果正则中出现引号，转为\"正则中正常的\ 都要转为\\

例子：

<script>

var names=["明月","白鹭","青天","紫烟"];

var str="窗前明月光,一行白鹭上青天,日照香炉生紫烟";

var reg=new RegExp(names.join("|"),"g");

var kwds=str.match(reg);

console.log(kwds);

var reg=new RegExp("\\d{6}");

</script>

* + API:

**1. 验证: var bool=reg.test(str);**

用reg验证str是否符合规则的要求;返回bool类型

问题: 正则默认只要找到部分匹配就返回true

解决: 要求从头到尾必须完整匹配

今后，凡是验证，必须同时前加^后加$

**2. 查找: 即找位置，又找内容**

**var arr=reg.exec(str);**

在str中找下一个满足reg要求的关键词的内容和位置

返回值: 本次找到的关键词的内容和位置

数组: 0号房间保存本次找到的关键词内容

index的房间保存本次找到的关键词位置

如果找不到，返回null

强调: 反复调用，会自动向后查找下一个！

<script>

var str="那天，我去了她家。我说: 我草，你家真大！她说想和我去草地野餐。我草草的收拾了草地。她说：我去去就来";

var reg=/我[去草]{1,2}/g;

do{

var arr=reg.exec(str);

if(arr!==null){

console.log(

"在位置"+arr["index"]

+"发现敏感词"+arr[0]

+",下次从"+reg.lastIndex+"开始"

);

}else{ break; }

}while(true);

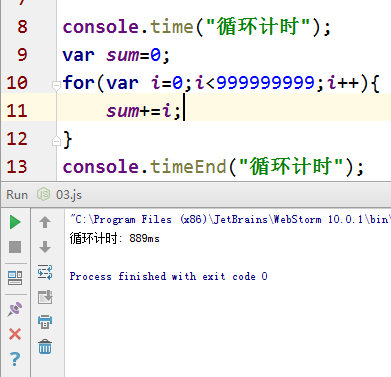
</script>

### 计算程序运行时间：循环计时

console.time(“循环计时”)； //开始计时

function; //…..运行的程序；

console.timeEnd(“循环计时”)； //结束计时，循环计时的内容要一致



## BOM和DOM部分

### 弹出的窗口

* 1. aleart(); 警告
  2. prompt(); 输入
  3. confirm(); 确认

### 7、窗口大小

窗口的完整大小：window.outerWidth/outerHeight;

仅文档显示区大小: window.innerWidth/innerHeight——用得多

### 鼠标指针位置和页面滚动位置

**1. 鼠标位置:**

相对于屏幕左上角的坐标: e.screenX, e.screenY

相对于文档显示区左上角的坐标: e.clientX,e.clientY

相对于当前元素左上角的坐标: e.offsetX, e.offsetY

**2. 页面滚动:**

事件: window.onscroll=function(){document.body.scrollTop //滚动距离 }

<script>

//为div绑定鼠标按下和鼠标抬起事件

//为body绑定mousemove事件

var pop=document.getElementById("pop");

var canMove=false;//标识是否可移动

var offsetX,offsetY;

pop.onmousedown=function(e){

//准备拖拽

canMove=true;

offsetX= e.offsetX;

offsetY= e.offsetY;

}

document.body.onmousemove=function(e){

if(canMove){

//让pop跟随鼠标移动

var top= e.clientY-offsetY;

var left= e.clientX-offsetX;

pop.style.cssText=

"top:"+top+"px;left:"+left+"px";

} }

pop.onmouseup=function(){ /\*结束拖拽\*/ canMove=false; }

</script>

### 获得元素距离页面顶部的距离(offset.top)

offsetTop=$(“#f1”).offset().top;

### eval(),js计算表达式

var t=eval(10+20-5-5\*2); // t=15

## jQuery部分

### 获得元素名称：

\*原生js：var name = element.tagName;

jq：

1、$( this ).get(0).tagName  
2、$( this )[0].tagName  
3、$( this ).prop("tagName")  
4、$( this ).prop("nodeName")

5、有时可以用is取代：

$(this).is(“b”) //b为元素名称

### 父元素增加子元素/添加子元素-prepend和append

1. **prepend=>向每个匹配的元素内部前置内容**

**例子1：**

<p>I would like to say: </p>

$("p").prepend("<b>Hello</b>");

结果：

[ <p><b>Hello</b>I would like to say: </p> ]

例子2： //将数量，单价和小计拼接为一行，追加到tbody下

$("tbody").prepend(

`<tr>

<td>¥${p.toFixed(2)}</td>

<td>${c}</td>

<td>¥${sub.toFixed(2)}</td>

<td><span class="close">×</span></td>

</tr>`

);

1. **append 向每个匹配的元素内部追加内容。**

<p>I would like to say: </p>

$("p").append("<b>Hello</b>");

结果：

[ <p>I would like to say: <b>Hello</b></p> ]

### 弹出错误/自定义插件时判断是否引入jq文件

if(typeof jQuery!="function"){

throw new Error("myAccordion插件依赖于jQuery，必须先引入jquery.js");

}else{

jQuery.fn.accordion=function(){

this.on("click",".title",e=>

$(e.target)

.next(".content").toggleClass("in")

.siblings(".content").removeClass("in")

);

};

}

### jQuery事件处理的另一种写法：

**$(“.nav”).on(“mouseenter”,”a”,function(){})**  //”.nav”是“a”的父元素；“a”是“.nav”的子元素，函数内可以不用e.target的冒泡判断=>if($(e.target).is(“a”)){…}

### jq 按钮只能点一次（事件只处理一次）

**$**(**"#bt-lottery"**).one(**"click"**,**function**(){  
 **console**.log(1)  
 })

### jQuery自带循环获取数组中每一个对象

**$.each(data,function(idx,obj){});**

//data为数组，idx为数组下标，obj当前遍历的对象（数组内容）

### jq判断checkbox是否被选择的三种方法

**方法一：**if ($("#checkbox-id")get(0).checked) {  
    // do something  
}

**方法二：**if($('#checkbox-id').is(':checked')) {  
    // do something  
}

**方法三：**if ($('#checkbox-id').attr('checked')) {  
    // do something  
}