

今日内容：

- 复习昨天内容：事件和运动特效
- 常用方法的学习

1、常用方法

- get()：下标和length属性

```
1 <!DOCTYPE html >
2 <head>
3 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
4 <title>无标题文档</title>
5
6 <script type="text/javascript" src="jquery-1.10.1.min.js"></script>
7 <script>
8
9 //get()：就是把JQ转成原生JS
10
11 $(function(){
12
13     //document.getElementById('div1').innerHTML
14
15     //alert( $('#div1').get(0).innerHTML );
16
17     /*for(var i=0;i<$('#li').get().length;i++){
18         $('#li').get(i).style.background = 'red';
19     }*/
20
21     for(var i=0;i<$('#li').length;i++){
22         $('#li').get(i).style.background = 'red';
23
24         //$('li')[i].style.background = 'red';
```

```

25     }
26
27 });
28
29
30
31 </script>
32 </head>
33
34 <body>
35 <div id="div1">div</div>
36 <ul>
37     <li></li>
38     <li></li>
39     <li></li>
40     <li></li>
41 </ul>
42 </body>
43 </html>

```

•outerWidth() ： 针对隐藏元素和参数true

```

1  <!DOCTYPE html>
2  <head>
3  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
4  <title>无标题文档</title>
5  <style>
6  #div1{ width:100px; height:100px; background:red; display:none;}
7  </style>
8  <script type="text/javascript" src="jquery-1.10.1.min.js"></script>
9  <script>
10
11  //outerWidth()
12
13  //offsetWidth ： 是获取不到隐藏元素的值
14
15  $(function(){

```

```

16
17     //alert( $('#div1').get(0).offsetWidth );
18
19     alert( $('#div1').outerWidth() );
20
21 });
22
23
24
25 </script>
26 </head>
27
28 <body>
29 <div id="div1">div</div>
30
31 </body>
32 </html>

```

•text() ： 合体的特例

```

1 <!DOCTYPE html>
2 <head>
3 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
4 <title>无标题文档</title>
5 <style>
6 #div1{ width:100px; height:100px; background:red; display:none;}
7 </style>
8 <script type="text/javascript" src="jquery-1.10.1.min.js"></script>
9 <script>
10
11 $(function(){
12
13     //alert( $('div').html() );
14
15     //alert( $('div').text() ); //会获取所有的内容(特例)
16
17     $('div').text('<h3>h3</h3>');
18
19

```

```
20 });
21
22
23
24 </script>
25 </head>
26
27 <body>
28 <div>div1<span>span</span></div>
29 <div>div2</div>
30 <div>div3</div>
31
32 </body>
33 </html>
```

•remove() :detach()

```
1 <!DOCTYPE html>
2 <head>
3 <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
4 <title>无标题文档</title>
5 <style>
6 #div1{ width:100px; height:100px; background:red; display:none;}
7 </style>
8 <script type="text/javascript" src="jquery-1.10.1.min.js"></script>
9 <script>
10
11 //detach() : 跟remove方法一样，只不过会保留删除这个元素的操作行为
12
13 $(function(){
14
15     $('div').click(function(){
16         alert(123);
17     });
18
19     var oDiv = $('div').detach();
20
21     $('body').append( oDiv );
22
```

```

23
24 });
25
26
27
28 </script>
29 </head>
30
31 <body>
32 <div>div1<span>span</span></div>
33
34
35 </body>
36 </html>

```

•parents() closest()

```

1 <!DOCTYPE HTML>
2 <html>
3 <head>
4 <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
5 <title>无标题文档</title>
6 <script src="jquery-1.10.1.min.js"></script>
7 <script>
8
9 //parents() : 获取当前元素的所有祖先节点，参数就是筛选功能
10
11 //closest() : 获取最近的指定的祖先节点(包括当前元素自身),必须要写筛选的参数,只能找到一个元素
12
13 $(function(){
14
15     $('#div2').parents('.box').css('background','red');
16
17     $('#div2').closest('.box').css('background','red');
18
19 });
20
21 </script>

```

```
22 </head>
23
24 <body class="box">
25 <div id="div1">aaa
26     <div id="div2" class="box">bbb</div>
27 </div>
28 </body>
29 </html>
```

- siblings()
- nextAll() prevAll()
- parentsUntil() nextUntil() prevUntil()

```
1 <!DOCTYPE HTML>
2 <html>
3 <head>
4 <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
5 <title>无标题文档</title>
6 <script src="jquery-1.10.1.min.js"></script>
7 <script>
8
9 //siblings() : 找所有的兄弟节点，参数也是筛选功能
10
11 //nextAll() : 下面所有的兄弟节点，参数也是筛选功能
12
13 //prevAll() : 上面所有的兄弟节点
14
15 //Until() : 截止
16
17 $(function(){
18
19     $('span').nextUntil('h2').css('background','red');
20
21 });
22
```

```
23 </script>
24 </head>
25
26 <body>
27 <div>div</div>
28 <span>span</span>
29 <p>p</p>
30 <h1>h1</h1>
31 <h2>h2</h2>
32 <em>em</em>
33 </body>
34 </html>
```

•add() slice()

```
1 <!DOCTYPE HTML>
2 <html>
3 <head>
4 <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
5 <title>无标题文档</title>
6 <script src="jquery-1.10.1.min.js"></script>
7 <script>
8
9
10
11 $(function(){
12
13     /*var elem = $('div');
14
15     var elem2 = elem.add('span');
16
17     elem.css('color','red');
18     elem2.css('background','yellow');*/
19
20     $('li').slice(1,4).css('background','red');
21
22 });
```

```
23
24 </script>
25 </head>
26
27 <body>
28 <div>div</div>
29 <span>span</span>
30 <ul>
31     <li></li>
32     <li></li>
33     <li></li>
34     <li></li>
35     <li></li>
36 </ul>
37
38 </body>
39 </html>
```

•serialize() serializeArray()数据串连

```
1 <!DOCTYPE HTML>
2 <html>
3 <head>
4 <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
5 <title>无标题文档</title>
6 <script src="jquery-1.10.1.min.js"></script>
7 <script>
8
9
10
11 $(function(){
12
13     //console.log($('form').serialize()); //string : a=1&b=2&c=3
14
15     console.log( $('form').serializeArray() );
16
17     [
18         { name : 'a' , value : '1' },
19         { name : 'b' , value : '2' },
```



```

20     { name : 'c' , value : '3' }
21   ]
22
23 });
24
25 </script>
26 </head>
27
28 <body>
29 <form>
30   <input type="text" name="a" value="1">
31   <input type="text" name="b" value="2">
32   <input type="text" name="c" value="3">
33 </form>
34
35 </body>
36 </html>

```

- `delegate()` 事件委托：提高性能
- `undelegate()`：取消事件委托

```

1  <!DOCTYPE HTML>
2  <html>
3  <head>
4  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
5  <title>无标题文档</title>
6  <script src="jquery-1.10.1.min.js"></script>
7  <script>
8
9
10
11 $(function(){
12
13   /*$('li').on('click',function(){
14     this.style.background = 'red';
15   });*/
16

```

```

17     $('ul').delegate('li','click',function(){
18
19         this.style.background = 'red';
20
21         $('ul').undelegate();
22
23     });
24
25 });
26
27 </script>
28 </head>
29
30 <body>
31 <ul>
32     <li>11111</li>
33     <li>1111</li>
34     <li>1111</li>
35     <li>111</li>
36     <li>1111</li>
37 </ul>
38
39 </body>
40 </html>

```

•ev.data ev.target ev.type

```

1  <!DOCTYPE HTML>
2  <html>
3  <head>
4  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
5  <title>无标题文档</title>
6  <script src="jquery-1.10.1.min.js"></script>
7  <script>
8
9
10
11  $(function(){
12

```

```
13     $('#div1').on('click',{name:'hello'},function(ev){
14         //alert(ev.data.name);
15
16         //alert( ev.target );
17
18         alert( ev.type );
19
20     });
21
22
23 });
24
25 </script>
26 </head>
27
28 <body>
29 <div id="div1">aaaa</div>
30 </body>
31 </html>
```

● \$下的常用方法

- type()
- trim()
- inArray()
- proxy()
- noConflict()
- parseJSON()

•makeArray()

```
1 <!DOCTYPE HTML>
2 <html>
3 <head>
4 <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
5 <title>无标题文档</title>
6 <script src="jquery-1.10.1.min.js"></script>
7 <script>
8
9
10 //$.css() $.html() $.val() : 只能给JQ对象用
11
12 //$.xxx() $.yyy() $.zzz() : 不仅可以给JQ用, 也可以给原生JS用 : 叫做工
    具方法
13
14
15
16
17 $(function(){
18
19     //var a = null;
20
21     //$.type() : 也是判断类型
22
23     //alert( typeof a );
24
25     //alert( $.type(a) );
26
27     var str = '    hello    ';
28
29     alert('('+$$.trim(str)+')');
30
31 });
32
33 </script>
34 </head>
35
36 <body>
37
```

```
38 </body>
39 </html>
```

inArray() : 找位置

proxy() : 改变this指向

```
1  <!DOCTYPE HTML>
2  <html>
3  <head>
4  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
5  <title>无标题文档</title>
6  <script src="jquery-1.10.1.min.js"></script>
7  <script>
8
9
10 //$.css() $.html() $.val() : 只能给JQ对象用
11
12 //$.xxx() $.yyy() $.zzz() : 不仅可以给JQ用, 也可以给原生JS用 : 叫做工
    具方法
13
14
15 //inArray() : 类似于 indexOf
16
17 $(function(){
18
19     //var arr = ['a','b','c','d'];
20
21     //alert( $.inArray('b',arr) );
22
23
24     //proxy() : 改变this指向的
25
26     function show(n1,n2){
27         alert(n1);
28         alert(n2);
```

```

29     alert(this);
30 }
31
32 //show();
33
34 //$.proxy(show , document,3)(4);
35
36
37 $(document).click( $.proxy(show>window,3,4) );
38
39 });
40
41 </script>
42 </head>
43
44 <body>
45
46 </body>
47 </html>

```

noConflict()

- parseJSON()
- makeArray()

```

1  <!DOCTYPE HTML>
2  <html>
3  <head>
4  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
5  <title>无标题文档</title>
6  <script src="jquery-1.10.1.min.js"></script>
7  <script>
8
9
10 //$.css()  $.html()  $.val()  : 只能给JQ对象用
11
12 //$.xxx()  $.yyy()  $.zzz()  : 不仅可以给JQ用, 也可以给原生JS用 : 叫做工

```

具方法

```
13
14
15 // $ , jQuery
16
17 // noConflict() : 防止冲突的
18
19 /*var aa = $.noConflict();
20
21 var $ = 10;
22
23 aa(function(){
24
25     aa('body').css('background','red');
26
27 });*/
28
29
30 //var str = '{"name":"hello"}';
31 //alert($.parseJSON( str ).name);
32
33
34 window.onload = function(){
35
36     var aDiv = document.getElementsByTagName('div'); //类数组
37
38     $.makeArray(aDiv).push();
39
40 };
41
42 </script>
43 </head>
44
45 <body>
46 <div></div>
47 <div></div>
48 <div></div>
49 </body>
50 </html>
```

2、ajax函数的运用

- ajax()：json形式的配置参数

- url success

- error contentType

- data type

- dataType cachetimeout

- 抽象出来的方法

- get()

- post()

- getJSON()

»支持jsonp的形式：指定?callback=?

```
1 <!DOCTYPE HTML>
2 <html>
3 <head>
4 <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
5 <title>无标题文档</title>
6 <script src="jquery-1.10.1.min.js"></script>
7 <script>
8
9 /*$.ajax({
10     url : 'xxx.php',
11     data : 'name=hello&age=20',
12     type : 'POST',
13     success : function(data){
```



```

14         alert(1);
15     },
16     error : function(){
17         alert(2);
18     }
19 });*/
20
21 $.get('xxx.php',function(){
22
23 });
24
25 $.post('xxx.php',function(){
26
27 });
28
29 $.getJSON('xxx.php?callback=?',function(data){
30     data
31 });
32
33 随机({});
34
35 </script>
36 </head>
37
38 <body>
39 </body>
40 </html>

```

3、jq如何调用json数据

复习json：

```

1 <!DOCTYPE html>
2 <html>
3     <head>
4         <meta charset="UTF-8">
5         <title></title>
6         <script src="jquery-1.10.1.min.js" type="text/javascript"

```

```

1  charset="utf-8"></script>
2
3      <script type="text/javascript">
4          $(function(){
5              var json={
6                  'name': '张三',
7                  'sex': '女',
8                  'tel': 164656656
9              }
10
11              alert(json.name);
12          });
13      </script>
14  </head>
15  <body>
16
17  </body>
18  </html>

```

遍历json : for in 循环

```

1  <!DOCTYPE html>
2  <html>
3      <head>
4          <meta charset="UTF-8">
5          <title></title>
6          <script src="jquery-1.10.1.min.js" type="text/javascript"
7  charset="utf-8"></script>
8          <script type="text/javascript">
9              $(function(){
10                  var json={
11                      'name': '张三',
12                      'sex': '女',
13                      'tel': 164656656
14                  }
15
16                  for(var i in json){
17                      alert(i+' : '+json[i]);
18                  }
19              }
20          </script>
21      </head>
22      <body>
23
24      </body>
25  </html>

```

```
18         });
19     </script>
20 </head>
21 <body>
22
23 </body>
24 </html>
```

上面的数据的json形式比较简单。json形式一般有两种：

第一种：

```
1 var json={'name':'张三','sex':'女','tel':164656656}
```

第二种：数组和json的结合

```
1         var json = {
2             '01':
3             [
4                 {id:1,name:'张晓','语文':80,'英语',75},
5                 {id:2,name:'李成','语文':88,'英语',75},
6                 {id:3,name:'罗明','语文':90,'英语',75}
7             ]
8         },
9         '02':
10        [
11            {id:1,name:'李梅','语文':80,'英语',75},
12            {id:2,name:'刘德华','语文':88,'英语',75},
13            {id:3,name:'王祖贤','语文':90,'英语',75}
14        ]
15
16    }
```

对于第一种类型的json我们采用for in 循环就很简单的遍历了json。
但是第二种就没有那么简单。

对于第一种数据的使用：

```
1  <!DOCTYPE html>
2  <html>
3
4      <head>
5          <meta charset="UTF-8">
6          <title></title>
7          <script src="jquery-1.10.1.min.js" type="text/javascript"
charset="utf-8"></script>
8          <script type="text/javascript">
9              $(function() {
10                  var json = {
11                      'name': '张三',
12                      'sex': '女',
13                      'tel': 164656656
14                  }
15
16                  $('#btn').click(function(){
17                      var str='';
18                      str=str+'姓名是: '+json.name+'<br>';
19                      str=str+'性别是: '+json.sex+'<br>';
20                      str=str+'电话是: '+json.tel;
21
22                      $('#div').html(str);
23                  });
24              });
25
26          </script>
27      </head>
28
29      <body>
30          <input type="button" id="btn" value="点击" />
31          <div></div>
32      </body>
```

```
33
34 </html>
```

each遍历数组和json：

```
1 <script type="text/javascript">
2     $(function() {
3         var anObject = {
4             one: 1,
5             two: 2,
6             three: 3
7         }; //对json数组each
8         $.each(anObject, function(name, value) {
9             alert(name);
10            alert(value);
11        });
12        var anArray = ['4', '5', '6'];
13        $.each(anArray, function(n, value) {
14            alert(n);
15            alert(value);
16        });
17    })
18
19 </script>
```

再看一个例子：

```
1 <!DOCTYPE html>
2 <html>
3
4     <head>
5         <meta charset="UTF-8">
6         <title></title>
7         <script src="jquery-1.10.1.min.js" type="text/javascript"
charset="utf-8"></script>
```

```

8      <script>
9          var arr = [{
10              name: "john",
11              lang: "js"
12          }, {
13              name: "nailwl",
14              lang: "jquery"
15          }, {
16              name: "吴磊",
17              lang: "ext"
18          }
19      ];
20      $.each(arr, function(index, content) {
21          alert("the man's no. is: " + index + ",and " +
22      content.name + " is learning " + content.lang);
23      });
24      </script>
25  </head>
26  <body>
27  </body>
28
29 </html>

```

jquery 对象的 `$.each()` 方法，此方法可用于例遍任何对象
回调函数拥有两个参数：

第一个为对象的成员或数组的索引例遍数组，同时使用元素
索引和内容

```

$.each( [0,1,2], function(index, content){
    alert( "item #" + index + " its value is: " + content );
});

```

第二种数据形式的遍历：

```
1 <!DOCTYPE html>
2 <html>
3
4 <head>
5     <meta charset="UTF-8">
6     <title></title>
7     <script src="jquery-1.10.1.min.js" type="text/javascript"
charset="utf-8"></script>
8     <script type="text/javascript">
9         $(function() {
10
11
12             var ss = {
13                 'a':
14                 [
15                     {id:1,name:'张晓','语文':80,'英语':75},
16                     {id:2,name:'李成','语文':88,'英语':75},
17                     {id:3,name:'罗明','语文':90,'英语':75}
18                 ]
19                 ,
20                 'b':
21                 [
22                     {id:1,name:'李梅','语文':80,'英语':75},
23                     {id:2,name:'刘德华','语文':88,'英语':75},
24                     {id:3,name:'王祖贤','语文':90,'英语':75}
25                 ]
26
27             }
28
29
30             //var a=ss.a
31             $.each(ss,function(index,value){
32                 $.each(value,function(n,content){
33                     alert('学号是'+content.id+'名字
是:'+content.name);
34                 })
35             })
36         }
37
38
39
```

```
40         );
41
42     });
43     </script>
44 </head>
45
46 <body>
47
48 </body>
49
50 </html>
```