

# webAPI 第五天

---

## 一. 三大系列

---

### 1.1 offset系列

- 节点对象.offsetWidth 和 节点对象.offsetHeight
  - 节点对象.offsetWidth
    - 作用：获取当前节点对象的宽度，返回数字，不包含单位。
    - 宽度：width + padding(左右) + border（左右）；
  - 节点对象.offsetHeight
    - 作用：获取当前节点对象的高度，返回数字，不包含单位。
    - 高度：height + padding(上下) + border(上下)
  - 代码：

```
1 <!DOCTYPE html>
2 <html>
3 <head lang="en">
4   <meta charset="UTF-8">
5   <title></title>
6   <style>
7     div {
8       width: 100px;
9       height: 100px;
10      padding:10px;
11      border:5px solid red;
12      background-color: blue;
13    }
14  </style>
15 </head>
16 <body>
17   <div></div>
18   <script>
19     var divNode = document.querySelector('div');
20     //width(100) + padding(左10 右10) +
border(左5 右5)
21     console.log(divNode.offsetWidth);//130
22     //height(100) + padding(上10 下10) +
border(上5 下5)
23     console.log(divNode.offsetHeight);//130
24   </script>
25 </body>
26 </html>
```

- 节点对象.offsetLeft 和 节点对象.offsetTop

- 节点对象.offsetLeft

- 作用：获取当前节点对象的x坐标，相对于其最近的定位的上级元素的坐标。否则，相对于body。

- 节点对象.offsetTop

- 作用：获取当前节点对象的y坐标，相对于其最近的定位的上级元素的坐标。否则，相对于body。

- 代码1：

```
1 <!DOCTYPE html>
2 <html>
3 <head lang="en">
4   <meta charset="UTF-8">
5   <title></title>
6   <style>
7     * {
8       margin:0;
9       padding:0;
10    }
11    .father {
12      width: 300px;
13      height: 300px;
14      background-color:blue;
15      margin:50px auto;
16      border:1px solid blue;
17      position: relative;
18    }
19    .son {
20      width:200px;
21      height: 200px;
22      margin:50px auto;
23      background-color: #000;
24    }
25  </style>
26 </head>
27 <body>
28   <!--父元素是定位的-->
29   <div class="father">
30     <div class="son"></div>
31   </div>
32   <script>
33
34     var sonNode =
```

```
document.querySelector('.son');
34     console.log(sonNode.offsetLeft); //50  参照
    定位的父元素
35     console.log(sonNode.offsetTop); //50  参照
    定位的父元素
36     </script>
37 </body>
38 </html>
```

- 代码2:

```
1 <!DOCTYPE html>
2 <html>
3 <head lang="en">
4   <meta charset="UTF-8">
5   <title></title>
6   <style>
7     * {
8       margin:0;
9       padding:0;
10    }
11    .father {
12      width: 300px;
13      height: 300px;
14      background-color:blue;
15      margin:50px auto;
16      border:1px solid blue;
17    }
18    .son {
19      width:200px;
20      height: 200px;
21      margin:50px auto;
22      background-color: #000;
23    }
24  </style>
25 </head>
26 <body>
27   <!-- 父元素的没有定位-->
28   <div class="father">
29     <div class="son"></div>
30   </div>
31   <script>
32     var sonNode =
document.querySelector('.son');
33     console.log(sonNode.offsetLeft);//406 参
```

```
34 照body console.log(sonNode.offsetTop);//101 参
    照body
35  </script>
36  </body>
37  </html>
```

- 节点对象.offsetParent

- 作用：获取节点对象的最近的定位的上级节点对象
- 代码1：

```
1 <!DOCTYPE html>
2 <html>
3 <head lang="en">
4   <meta charset="UTF-8">
5   <title></title>
6   <style>
7     * {
8       margin:0;
9       padding:0;
10    }
11    .father {
12      width: 300px;
13      height: 300px;
14      background-color:blue;
15      margin:50px auto;
16      border:1px solid blue;
17      position: relative;
18    }
19    .son {
20      width:200px;
21      height: 200px;
22      margin:50px auto;
23      background-color: #000;
24    }
25  </style>
26 </head>
27 <body>
28   <!-- 父元素有定位-->
29   <div class="father">
30     <div class="son"></div>
31   </div>
32   <script>
33
34     var sonNode =
```



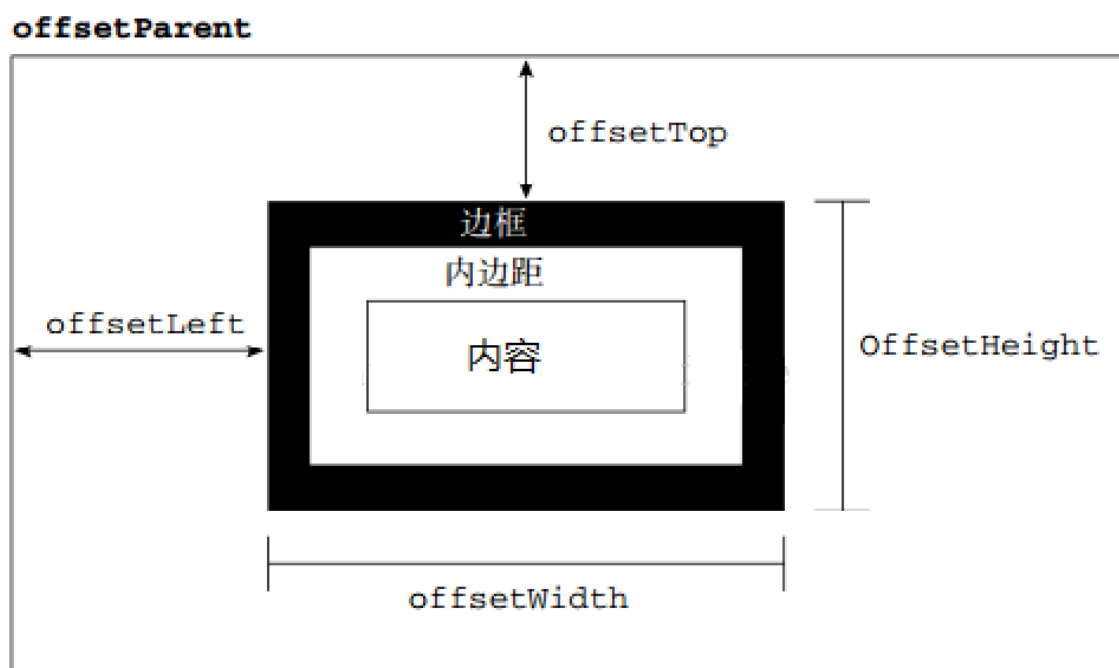
```
document.querySelector('.son');
34     var parent = sonNode.offsetParent;
35     console.log(parent); //获取定位的上级元素，
    div.father
36     </script>
37 </body>
38 </html>
```

◦ 代码2:

```
1 <!DOCTYPE html>
2 <html>
3 <head lang="en">
4   <meta charset="UTF-8">
5   <title></title>
6   <style>
7     * {
8       margin:0;
9       padding:0;
10    }
11    .father {
12      width: 300px;
13      height: 300px;
14      background-color:blue;
15      margin:50px auto;
16      border:1px solid blue;
17    }
18    .son {
19      width:200px;
20      height: 200px;
21      margin:50px auto;
22      background-color: #000;
23    }
24  </style>
25 </head>
26 <body>
27   <!-- 父元素没有定位-->
28   <div class="father">
29     <div class="son"></div>
30   </div>
31   <script>
32     var sonNode =
document.querySelector('.son');
33     var parent = sonNode.offsetParent;
```

```
34     console.log(parent); //因为没有最近的定位的
    上级元素，所以获取body
35     </script>
36 </body>
37 </html>
```

- 图示offset系列



## 1.2 client系列

- 节点对象.clientWidth 和 节点对象.clientHeight
  - 节点对象.clientWidth
    - 作用：获取当前节点对象的宽度，返回数字，不包含单位。
    - 宽度：width + padding(左右);
  - 节点对象.clientHeight
    - 作用：获取当前节点对象的高度，返回数字，不包含单位。
    - 高度：height + padding(上下);

- 代码:

```
1 <!DOCTYPE html>
2 <html>
3 <head lang="en">
4   <meta charset="UTF-8">
5   <title></title>
6   <style>
7     div {
8       width: 100px;
9       height: 100px;
10      padding:10px;
11      border:5px solid red;
12      background-color: blue;
13      margin: 100px auto;
14    }
15  </style>
16 </head>
17 <body>
18   <div></div>
19   <script>
20     var divNode = document.querySelector('div');
21     //width(100) + padding(左10 右10)
22     console.log(divNode.clientWidth);//120
23     //height(100) + padding(上10 下10)
24     console.log(divNode.clientHeight);//120
25   </script>
26 </body>
27 </html>
```

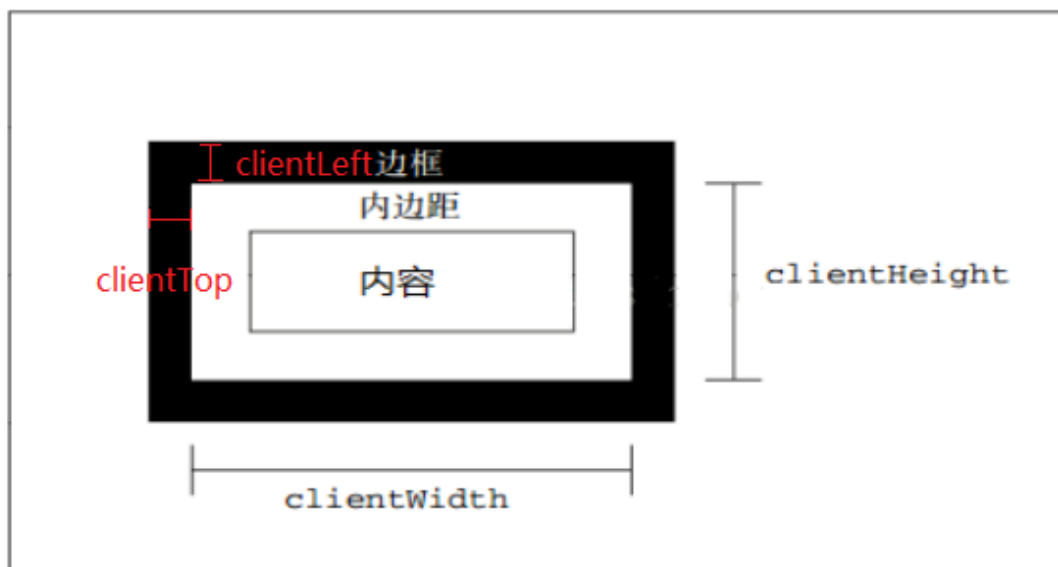
- 节点对象.clientLeft 和 节点对象.clientTop 【几乎不用，了解】

- 节点对象.clientLeft;

- 作用：获取当前节点对象的padding-left的外边界，距离border-left外边界的距离。实际上就是左边框的厚度。
- 节点对象.clientTop;
  - 作用：获取当前节点对象的padding-top的外边界，距离border-top外边界的距离。实际上就是上边框的厚度。
- 代码：

```
1 <!DOCTYPE html>
2 <html>
3 <head lang="en">
4   <meta charset="UTF-8">
5   <title></title>
6   <style>
7     div {
8       width: 200px;
9       height: 200px;
10      padding:10px;
11      border:10px solid red;
12      border-top:50px solid red;
13      background-color: blue;
14      margin: 100px auto;
15    }
16  </style>
17 </head>
18 <body>
19 <div></div>
20 <script>
21   var divNode = document.querySelector('div');
22   console.log(divNode.clientLeft);//10
23   console.log(divNode.clientTop);//50
24 </script>
25 </body>
26 </html>
```

- 图示client系列



## 1.3 scroll系列

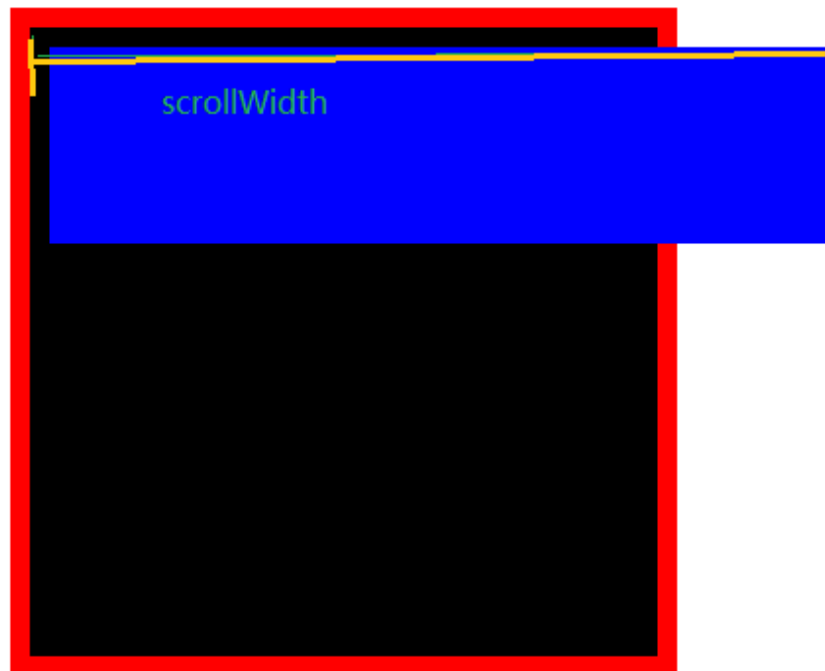
- 节点对象.`scrollWidth` 和 `scrollHeight`
  - 节点对象.`scrollWidth`
    - 作用:获取当前节点对象的宽度，返回数字，不包含单位。
    - 宽度: `width+padding` (左右) + 溢出部分
    - 代码:

```
1 <!DOCTYPE html>
2 <html>
3 <head lang="en">
4   <meta charset="UTF-8">
5   <title></title>
6   <style>
7     * {
8       margin:0;
9       padding:0;
10    }
11    .father {
12      width:300px;
13      height: 300px;
14      background-color: #000;
15      margin:100px auto;
16      /*overflow: auto;*/
17      padding: 10px;
18      border:10px solid red;
19    }
20    .son {
21      width: 400px;
22      height: 100px;
23      background-color: blue;
24    }
25  </style>
26 </head>
27 <body>
28   <div class="father">
29     <div class="son">
30
31     </div>
32   </div>
33   <script>
34     var fNode =
```



```
document.querySelector('.father');
35     /*
36     width + padding(左右) 包含溢出部分
37     */
38     console.log(fNode.scrollWidth); //410
39 </script>
40 </body>
41 </html>
```

■ 图示：



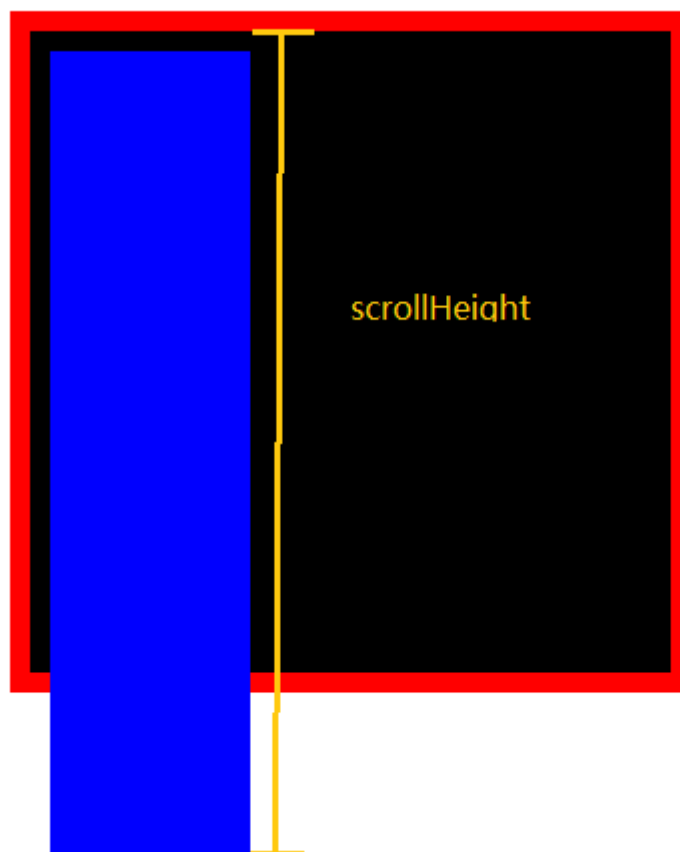
○ 节点对象.scrollHeight

- 作用：获取当前节点对象的高度，返回数字，不包含单位。
- 高度：height + padding(上下) + 溢出部分;
- 代码：

```
1 <!DOCTYPE html>
2 <html>
3 <head lang="en">
4   <meta charset="UTF-8">
5   <title></title>
6   <style>
7     * {
8       margin:0;
9       padding:0;
10    }
11    .father {
12      width:300px;
13      height: 300px;
14      background-color: #000;
15      margin:100px auto;
16      padding: 10px;
17      border:10px solid red;
18    }
19    .son {
20      width: 100px;
21      height: 400px;
22      background-color: blue;
23    }
24  </style>
25 </head>
26 <body>
27   <div class="father">
28     <div class="son">
29
30   </div>
31 </div>
32 <script>
33
34   var fNode =
```

```
document.querySelector('.father');  
34      /*  
35      height + padding(上下) 包含溢出部分  
36      */  
37      console.log(fNode.scrollHeight); //410  
38  </script>  
39  </body>  
40  </html>
```

■ 图示：



- 节点对象.scrollLeft 和 节点对象.scrollTop

- 节点对象.scrollLeft

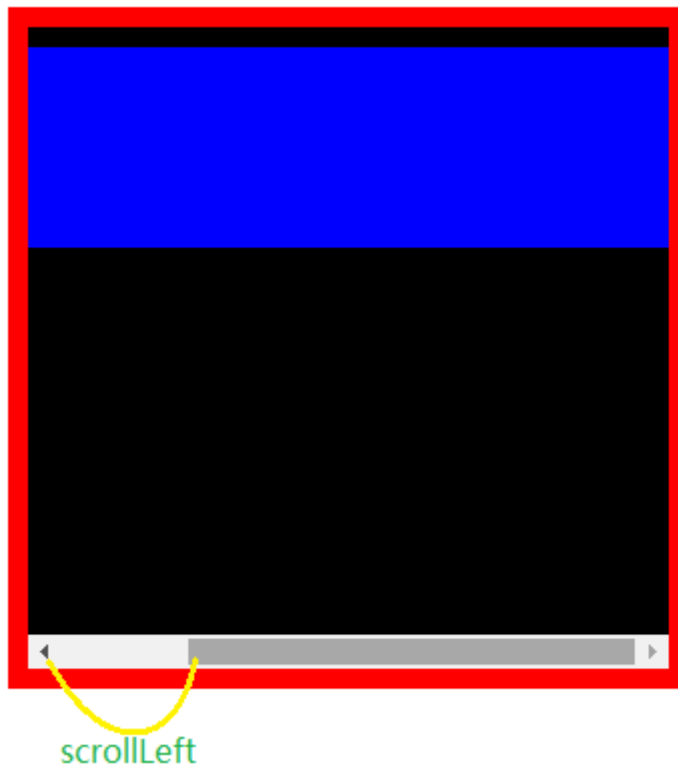
- 作用：获取被卷去的横向宽度

- 代码：

```
1 <!DOCTYPE html>
2 <html>
3 <head lang="en">
4   <meta charset="UTF-8">
5   <title></title>
6   <style>
7     * {
8       margin:0;
9       padding:0;
10    }
11    .father {
12      width:300px;
13      height: 300px;
14      background-color: #000;
15      margin:100px auto;
16      padding: 10px;
17      overflow: auto;
18      border:10px solid red;
19    }
20    .son {
21      width: 400px;
22      height: 100px;
23      background-color: blue;
24    }
25  </style>
26 </head>
27 <body>
28   <div class="father">
29     <div class="son">
30
31     </div>
32   </div>
33   <script>
34     var fNode =
```

```
document.querySelector('.father');
35     fNode.onscroll = function(){
36         console.log(fNode.scrollLeft);
37     }
38     </script>
39 </body>
40 </html>
```

■ 图示：



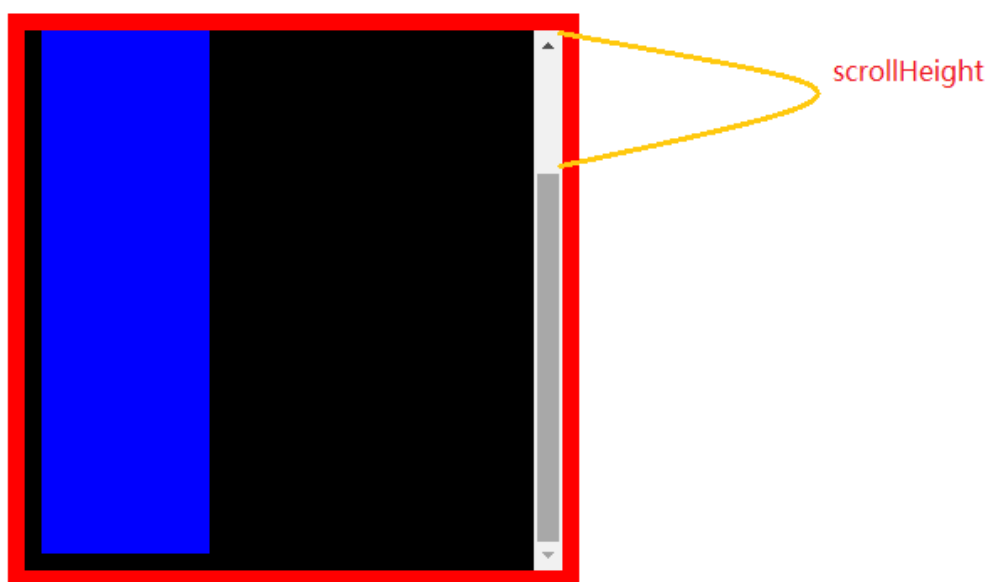
○ 节点对象.scrollHeight

- 作用：获取被卷去的纵向高度
- 代码：

```
1 <!DOCTYPE html>
2 <html>
3 <head lang="en">
4   <meta charset="UTF-8">
5   <title></title>
6   <style>
7     * {
8       margin:0;
9       padding:0;
10    }
11    .father {
12      width:300px;
13      height: 300px;
14      background-color: #000;
15      margin:100px auto;
16      padding: 10px;
17      overflow: auto;
18      border:10px solid red;
19    }
20    .son {
21      width: 100px;
22      height: 400px;
23      background-color: blue;
24    }
25  </style>
26 </head>
27 <body>
28   <div class="father">
29     <div class="son">
30
31     </div>
32   </div>
33   <script>
34     var fNode =
```

```
document.querySelector('.father');
35     fNode.onscroll = function(){
36         console.log(fNode.scrollTop);
37     }
38 </script>
39 </body>
40 </html>
```

■ 图示：



## 二. 事件监听

### 2.1 什么是事件监听 【了解】

绑定事件的另一种方式。事件监听可以监听多个事件处理程序，也可以把指定的事件处理程序从该事件中移除。

### 2.2 事件监听的方式绑定和解绑事件 【重要】

- 标准方式

- 语法:

- 绑定

```
1 事件目标.addEventListener(事件类型,事件处理程序,  
    是否捕获);  
2      事件目标: 要绑定的那个节点对象。  
3      事件类型: 交互行为, 在这里不加on  
4      事件处理程序: 函数  
5      是否捕获: 可选, 布尔值, true是捕获, false是冒  
    泡, 默认为false;
```

- 解绑

```
1 事件目标.removeEventListener(事件类型,事件处理程  
    序的名称);  
2      事件目标: 要解绑事件的那个节点对象  
3      事件类型: 解绑什么类型的实际, 不加on  
4      事件处理程序的名称: 函数的名称;
```

- 代码:



```

1 <!DOCTYPE html>
2 <html>
3 <head lang="en">
4   <meta charset="UTF-8">
5   <title></title>
6 </head>
7 <body>
8   <button id="btn">按钮</button>
9   <button id="jb">解绑事件处理程序一</button>
10  <script>
11    var btn = document.querySelector('#btn');
12    var jb = document.querySelector('#jb');
13    /*事件处理程序一*/
14    var test1 = function(){
15      alert(1);
16    };
17    /*事件处理程序二*/
18    var test2 = function(){
19      alert(2);
20    };
21    btn.addEventListener('click',test1); //绑定
22    1 btn.addEventListener('click',test2); //绑定
23    2 //点击解绑按钮的事件处理程序二
24    jb.onclick = function(){
25      btn.removeEventListener('click',test2);
26    };
27  </script>
28 </body>
29 </html>

```

- IE低版本方式

- 语法:

- 绑定:

```
1 事件目标.attachEvent(事件类型,事件处理程序);  
2      事件目标: 要绑定的那个节点对象。  
3      事件类型: 交互行为, 在这里要加on  
4      事件处理程序: 函数
```

- 解绑:

```
1 事件目标.detachEvent(事件类型,事件处理程序的名称);  
2      事件目标: 要绑定的那个节点对象。  
3      事件类型: 交互行为, 在这里要加on  
4      事件处理程序的名称: 函数名称
```

- 代码:

```

1 <!DOCTYPE html>
2 <html>
3 <head lang="en">
4   <meta charset="UTF-8">
5   <title></title>
6 </head>
7 <body>
8 <button id="btn">按钮</button>
9 <button id="jb">解绑事件处理程序一</button>
10 <script>
11   var btn = document.querySelector('#btn');
12   var jb = document.querySelector('#jb');
13   /*事件处理程序一*/
14   var test1 = function(){
15     alert(1);
16   };
17   /*事件处理程序二*/
18   var test2 = function(){
19     alert(2);
20   };
21   btn.attachEvent('onclick',test1); //绑定事件1
22   btn.attachEvent('onclick',test2); //绑定事件2
23
24   //解绑事件2
25   jb.onclick = function(){
26     btn.detachEvent('onclick',test2); //解绑按钮
    的事件处理程序2
27   };
28 </script>
29 </body>
30 </html>

```

- 兼容处理后绑定和解绑

- 语法:

- 绑定:

```
1  /*
2     功能：绑定事件
3     参数：
4         node 事件目标 节点对象
5         type 事件类型 string 不加on
6         handler 事件处理程序 函数
7     返回值：无
8  */
9  function addEvent(node,type,handler){
10     if(node.addEventListener){ //检测浏览器是
否支持标准方式
11         //支持
12         node.addEventListener(type,handler);
13     }else{
14         //不支持
15         node.attachEvent('on' + type,handler);
16     }
17 }
```

- 解绑:

```

1      /*
2      功能：解绑事件
3      参数：
4          node 事件目标  节点对象
5          type 事件类型  string
6          handlerName 事件处理程序名称  函数
7      返回值：无
8      */
9      function removeEvent(node,type,handlerName)
10     {
11         if(node.removeEventListener){//检测浏览器
12         是否支持标准方式
13             //支持
14             node.removeEventListener(type,handlerName);
15         }else{
16             //不支持
17             node.detachEvent('on' +
18             type,handlerName);
19         }
20     }

```

○ 代码：

```
1 <!DOCTYPE html>
2 <html>
3 <head lang="en">
4   <meta charset="UTF-8">
5   <title></title>
6 </head>
7 <body>
8 <button id="btn">按钮</button>
9 <button id="jb">解绑事件处理程序一</button>
10 <script>
11   /*
12     功能： 绑定事件
13     参数：
14         node 事件目标 节点对象
15         type 事件类型 string 不加on
16         handler 事件处理程序 函数
17     返回值： 无
18   */
19   function addEvent(node,type,handler){
20     if(node.addEventListener){ //检测浏览器是否支持标准方式
21       //支持
22       node.addEventListener(type,handler);
23     }else{
24       //不支持
25       node.attachEvent('on' + type,handler);
26     }
27   }
28
29   /*
30     功能： 解绑事件
31     参数：
32         node 事件目标 节点对象
33         type 事件类型 string
```

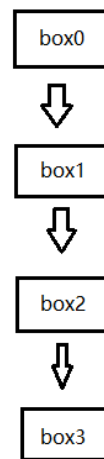
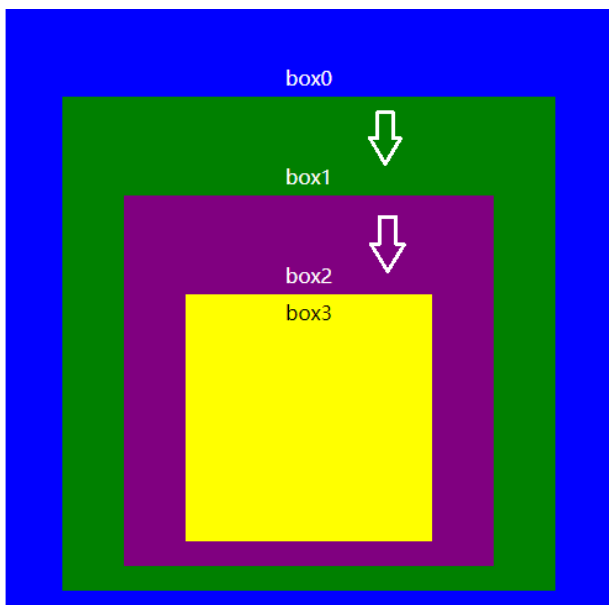
```
34     handlerName 事件处理程序名称 函数
35     返回值：无
36 */
37 function removeEvent(node,type,handlerName){
38     if(node.removeEventListener){//检测浏览器是否
支持标准方式
39         //支持
40
node.removeEventListener(type,handlerName);
41     }else{
42         //不支持
43         node.detachEvent('on' + type,handlerName);
44     }
45 }
46 var btn = document.querySelector('#btn');
47 var jb = document.querySelector('#jb');
48 /*事件处理程序一*/
49 var test1 = function(){
50     alert(1);
51 };
52 /*事件处理程序二*/
53 var test2 = function(){
54     alert(2);
55 };
56 addEvent(btn,'click',test1);
57 addEvent(btn,'click',test2);
58
59 //解绑事件2
60 jb.onclick = function(){
61     removeEvent(btn,'click',test2);
62 };
63 </script>
64 </body>
65 </html>
```

## 2.3 事件捕获 【了解】

- 事件捕获介绍

事件捕获，是事件传播的另外一种方式，和冒泡是相反的。捕获就是反冒泡。基本不用，了解知道就可以

传播方式：上级....→ 目标;



事件捕获：事件的传播方式之一，就是反冒泡。  
事件捕获基本不用。

- 代码：



```
1
2 <!doctype html>
3 <html lang="en">
4 <head>
5     <meta charset="UTF-8">
6     <meta name="Generator" content="EditPlus®">
7     <meta name="Author" content="">
8     <meta name="Keywords" content="">
9     <meta name="Description" content="">
10    <title>Document</title>
11    <style>
12        *{
13            margin:0;
14            padding:0;
15            line-height:30px;
16            text-align:center;
17            color:#fff;
18
19        }
20        .box0{
21            width:400px;
22            height:400px;
23            background-color:blue;
24            padding:50px;
25        }
26        .box1{
27            width:300px;
28            height:300px;
29            background-color:green;
30            padding:50px;
31        }
32        .box2{
33            width:200px;
34            height:200px;
```

```
35         background-color:purple;
36         padding:50px;
37     }
38     .box3{
39         width:200px;
40         height:200px;
41         background-color:yellow;
42         color:#000;
43     }
44 </style>
45 </head>
46 <body>
47 <div class="box0">
48     box0
49     <div class="box1">
50         box1
51         <div class="box2">
52             box2
53             <div class="box3">box3</div>
54         </div>
55     </div>
56 </div>
57
58 <script>
59     var divs = document.getElementsByTagName("div");
60     for(var i = 0;i<divs.length;i++){
61         divs[i].addEventListener('click',function(){
62             alert(this.className);
63         },true);
64     }
65 </script>
66 </body>
67 </html>
```

