

知识点列表

编号	名称	描述	级别
1	Html dom	掌握常用的几种对象及对象属性和方法的使用	**
2	BOM 模型	重点掌握 BOM 模型中的 window 对象及常用方法	**
3	事件处理机制	掌握 JS 中的事件处理机制，重点是了解不同浏览器处理事件对象区别的细节，了解并掌握事件冒泡现象	**
4	面向对象基础	了解 JS 中的面向对象基础，掌握 3 种创建对象的方式，	*

注： **"理解级别 ***"掌握级别 ****"应用级别

目录

1. html dom**	4
2. BOM 模型**	8
2.1. 什么是 BOM ? *	10
2.2. window 对象***	10
2.3. Document 对象**	12
2.4. Location 对象**	12
2.5. Navigator 对象**	错误！未定义书签。 2
2.6. Screen 对象 *	12
2.7. window 对象的方法**	1 错误！未定义书签。
2.8. window 对象的属性**	19
3. 事件处理机制**	28
3.1. 事件是如何产生的*	28
3.2. 绑定事件处理代码**	28
3.3. 事件对象***	35
3.4. 事件冒泡**	39
4. 面对对象基础*	43
4.1. 如何定义一个类型*	43
4.2. 如何创建一个 js 对象 ? *	48

1. html dom **

Html dom 指的是在 w3c 的 dom 规范出现之前, 各个浏览器支持的一些 dom 操作。

1) Select 对象

属性

- a. selectedIndex: 用户选择的选项的下标, 下标从 0 开始。
- b. length: 获取或者设置选项的个数。
- c. options: 返回一个数组, 数组元素是 Option 对象。

2) Option 对象

属性

- a. text: 选项的文本内容
- b. value: 选项的值
- c. selected: 当该选项被选上, 值为 true, 否则为 false。

小知识: 创建一个 Option 对象。

```
var op = new Option(text,value);
```

3) Table

属性

- ✓ tHead: 返回表格的 tHead 节点
- ✓ tFoot: 返回表格的 tFoot 节点
- ✓ tBodies: 返回表格的所有 tBody 节点
- ✓ rows: 返回表格所有的行

方法

- ✓ insertRow(index): 在 index 位置插入一条新的行, 该方法返回 TableRow 对象
- ✓ deleteRow(index): 删除 index 处的行

4) TableRow

属性

- ✓ cells: 返回该行的所有单元格

方法

- ✓ deleteCell(index): 删除 index 位置的一个单元格

5) TableCell

表示一个单元格

- 演示 1
w3c dom 方式

```

1=<html>
2=<head>
3=<!--Table TableRow TableCell-->
4=<style>
5=    table{
6=        font-size:24px;
7=        font-style:italic;
8=    }
9=    .s1{
10=        background-color:#ff99ee;
11=    }
12=</style>
13
14=<script src="prototype-1.6.0.3.js">
15=</script>
16
17=<script>
18=    /* w3c dom方式 */
19=    function addRow(){
20=        var name = $F('name');
21=        var salary = $F('salary');
22=        var td1 = document.createElement('td');
23=        var td2 = document.createElement('td');
24=        td1.innerHTML = name;
25=        td2.innerHTML = salary;
26
27=        var tr1 = document.createElement('tr');
28=        tr1.appendChild(td1);
29=        tr1.appendChild(td2);
30=        $('tb').appendChild(tr1);
31=    }
32=</script>
33=</head>
34=<body style="font-size:30px;">
35=    <table width="60%" cellpadding="0"
36=        cellspacing="0" border="1" id="tb">
37
38=        <tr><td>姓名</td><td>薪水</td></tr>
39=        <tr><td>zs</td><td>2000</td></tr>
40=        <tr><td>ww</td><td>3000</td></tr>
41=    </table>
42=

```

```

43  姓名:<input name="name" id="name"/>
44  薪水:<input name="salary" id="salary"/>
45
46  <input type="button" value="添加"
47  onclick="addRow();" />
48
49  </body>
50 </html>
51

```

Mozilla Firefox

文件(F) 编辑(E) 查看(V) 历史(S) 书签(B) 工具(T) 帮助(H)

file:///C:/Docume...or/桌面/js01_1.html +

file:///C:/Documents and Settings/Administrator/桌面/js01_1.html

姓名	薪水
ZS	2000
WW	3000
new	123
new	123
new	123
new	123

姓名: 薪水:

● 演示 2

Table 对象方式

```

1 <html>
2 <head>
3   <!--Table TableRow TableCell-->
4   <style>
5     table{
6       font-size:24px;
7       font-style:italic;
8     }
9     .s1{
10      background-color:#ff99ee;
11    }
12  </style>

```


```

13
14=    <script src="prototype-1.6.0.3.js">
15    </script>
16
17=    <script>
18    /* Table对象方式 */
19=    function addRow2(){
20        var tr1 =
21            $('tb').insertRow($('tb').rows.length);
22        tr1.className = 's1';
23        var td1 =
24            tr1.insertCell(tr1.cells.length);
25        var td2 =
26            tr1.insertCell(tr1.cells.length);
27        td1.innerHTML = $F('name');
28        td2.innerHTML = $F('salary');
29
30        /* 将zs的薪水背景变色 */
31        // $('tb').rows[1].cells[1].className = 's1';
32    }
33    </script>
34    </head>
35=    <body style="font-size:30px;">
36        <table width="60%" cellpadding="0"
37            cellspacing="0" border="1" id="tb">
38
39            <tr><td>姓名</td><td>薪水</td></tr>
40            <tr><td>zs</td><td>2000</td></tr>
41            <tr><td>ww</td><td>3000</td></tr>
42        </table>
43
44        姓名:<input name="name" id="name"/>
45        薪水:<input name="salary" id="salary"/>
46
47        <input type="button" value="添加"
48            onclick="addRow2();" />
49
50    </body>
51    </html>

```

js01_2.html

姓名	薪水
zs	2000
ww	3000
aa	1111
bb	11112

姓名: 薪水: 

【案例 1】Table TableRow TableCell

```
<html>
<head>
  <!--Table TableRow TableCell-->
  <style>
    table{
      font-size:24px;
      font-style:italic;
    }
    .s1{
      background-color:#ff99ee;
    }
  </style>

  <script src="prototype-1.6.0.3.js">
  </script>

  <script>
  function addRow(){
    var name = $F('name');
    var salary = $F('salary');
    var td1 = document.createElement('td');
    var td2 = document.createElement('td');
    td1.innerHTML = name;
    td2.innerHTML = salary;
```

```

        var tr1 = document.createElement('tr');
        tr1.appendChild(td1);
        tr1.appendChild(td2);
        $('tb').appendChild(tr1);
    }

    function addRow2(){
        var tr1 =
            $('tb').insertRow($('tb').rows.length);
        //tr1.className = 's1';
        var td1 =
            tr1.insertCell(tr1.cells.length);
        var td2 =
            tr1.insertCell(tr1.cells.length);
        td1.innerHTML = $F('name');
        td2.innerHTML = $F('salary');
        $('tb').rows[1].cells[1].className = 's1';
    }
</script>
</head>
<body style="font-size:30px;">
    <table width="60%" cellpadding="0"
        cellspacing="0" border="1" id="tb">

        <tr><td>姓名</td><td>薪水</td></tr>
        <tr><td>zs</td><td>2000</td></tr>
        <tr><td>ww</td><td>3000</td></tr>
    </table>

    姓名:<input name="name" id="name"/>
    薪水:<input name="salary" id="salary"/>

    <input type="button" value="添加"
        onclick="addRow2();"/>

</body>
</html>

```


The screenshot shows a web browser window with a single tab titled 'js01.html'. Inside the browser, there is a form containing a table with two columns: '姓名' (Name) and '薪水' (Salary). The table has three rows of data: (ZS, 2000) and (WW, 3000). Below the table, there are two input fields labeled '姓名:' and '薪水:', followed by a button labeled '添加' (Add).

姓名	薪水
ZS	2000
WW	3000

姓名: 薪水:

2. BOM 模型 **

2.1. 什么是 BOM? *

BOM (browser object model) 浏览器对象模型，是浏览器内置的一些对象，用来操作窗口。这些对象包括 window、screenlocation、navigator、document、XmlHttpRequest 等。虽然该部分没有规范，但是，各个浏览器都支持这些对象。

2.2. window 对象 ***

1) open 方法

打开一个新的窗口，例如：

```
window.open('1.html',
            'new',
            'height=100,
            width=200,
            top=0,
            left=0,
            toolbar=no,
            menubar=no,
            scrollbar=no,
            resizable=no,
            location=no,
            status=no');
```

- ✓ '1.html' 弹出窗口的文件名；
- ✓ 'new' 弹出窗口的名字；

如果该参数指定了一个已经存在的窗口，则 `open()` 方法不再创建一个新窗口，而只是返回对指定窗口的句柄。

- ✓ `height=100` 窗口高度；
- ✓ `width=200` 窗口宽度；
- ✓ `top=0` 窗口距离屏幕上方的像素值；
- ✓ `left=0` 窗口距离屏幕左侧的像素值；
- ✓ `toolbar=no` 是否显示工具栏，yes 为显示；
- ✓ `menubar` 是否显示菜单栏。
- ✓ `scrollbars` 是否显示滚动栏。
- ✓ `resizable=no` 是否允许改变窗口大小，yes 为是
- ✓ `location=no` 是否显示地址栏，yes 为是。
- ✓ `status=no` 是否显示状态栏内的信息。

2) **close 方法**

关闭窗口（返回值为窗口对象句柄）

3) **status 属性**

设置或返回窗口状态栏中的文本。

4) **document 属性**

获得 Document 对象

5) **location 属性**

获得 Location 对象

6) **navigator 属性**

获得 Navigator 对象

7) **screen 属性**

获得 Screen 对象

8) **opener 属性**

获得打开当前窗口的窗口对象

9) **parent 属性**

获得当前窗口的父窗口

10) **alert 方法**

弹出一个警告对话框

11) **confirm 方法**

弹出一个选择对话框，返回用户是否确认。

12) **prompt 方法**

弹出一个供用户输入信息的对话框，返回用户输入信息。

13) **setTimeout 方法**

`setTimeout(要执行的函数,毫秒数)` 方法用于在指定的毫秒数后调用函数。

14) **setInterval 方法**

按照指定的周期（以毫秒计）来调用函数。

```
var taskId = setInterval(要执行的函数, 毫秒数);
```

返回值 taskId 用于 clearInterval 方法。

15) clearInterval 方法

```
clearInterval(taskId);
```

注：

如果在 JS 中调用的方法之前没有具体对象，那一定是被省略的 window 对象，如 alert();

2.3. Document 对象 **

Document 对象代表整个 html 文档的根节点。

```
var obj = document.getElementById(id);
```

```
var obj = document.createElement(tagName);
```

```
document.write(string);
```

2.4. Location 对象 **

1) Location 对象封装了浏览器地址栏的相关信息。

2) href 属性：指定加载的页面。

比如：

```
location.href='js02.html';
```

2.5. Navigator 对象 **

Navigator 对象封装了浏览器本身的一些信息，比如浏览器的类型、版本、支持的语言等。

2.6. Screen 对象 *

Screen 对象封装了屏幕的一些信息，比如分辨率。

2.7. window 对象的方法 **

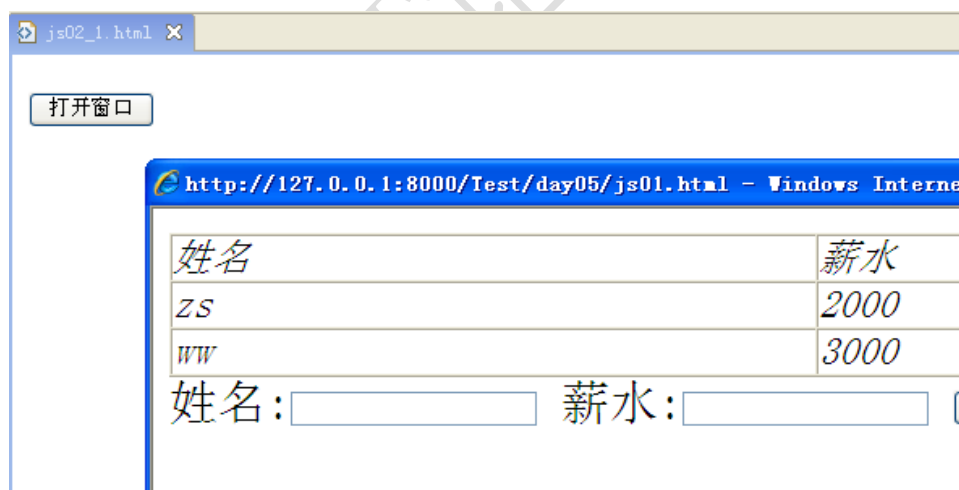
● 演示 1

打开一个窗口，5 秒后自动关闭

```

1 <html>
2   <!--window-->
3   <head>
4     <script>
5       function f1(){
6         /*win: 句柄，就是一个引用类型的变量
7            指向了新打开的窗口。
8            注: window可以省略不写。
9            */
10        var win = window.open('js01.html',
11                               'w1', 'width=400,height=300');
12        setTimeout(function(){
13          win.close();
14        },5000);
15      }
16    </script>
17  </head>
18  <body style="font-size:30px;">
19    <input type="button"
20          value="打开窗口" onclick="f1();" />
21  </body>
22 </html>

```

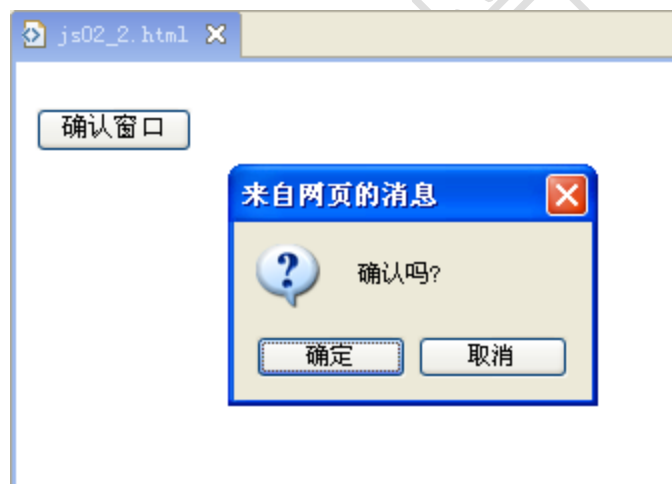


- 演示 2
确认窗口

```

1 <html>
2   <!--window-->
3   <head>
4     <script>
5       function f2() {
6         //点击确认, 返回值为true, 否则为false
7         var flag = confirm("确认吗?");
8         if(flag) {
9           alert('你选择了确认');
10        } else {
11          alert('你选择了取消');
12        }
13      }
14    </script>
15  </head>
16  <body style="font-size:30px;">
17    <input type="button"
18      value="确认窗口" onclick="f2();" />
19  </body>
20 </html>
21

```



● 演示 3

如果确定, 则向 del.do 发请求, 如果取消则不发请求

```

1 <html>
2   <!--window-->
3   <head>
4   </head>
5   <body style="font-size:30px;">
6
7     <a href="del.do"
8     onclick="return confirm('真的要删除吗?');">
9     delete</a>
10
11   </body>
12 </html>
13

```

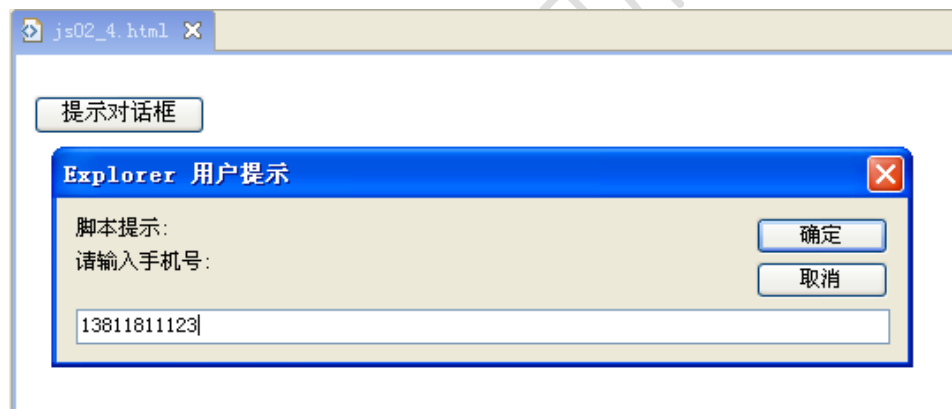


- 演示 4
提示对话框

```

1 <html>
2   <!--window-->
3   <head>
4     <script>
5       function f3(){
6         var msg = prompt('请输入手机号:');
7         alert(msg);
8       }
9     </script>
10  </head>
11  <body style="font-size:30px;">
12
13    <input type="button" value="提示对话框"
14    onclick="f3();" />
15    <br/>
16
17  </body>
18 </html>
19

```



【案例 2.1】window 对象

```

<html>
  <!--window-->
  <head>
    <script>
      function f1(){
        //win:句柄，就是一个引用类型的变量
        //指向了新打开的窗口。

```

```
//window可以省略不写。
var win = open('js01.html',
'w1','width=400,height=300');
setTimeout(function(){
    win.close();
},5000);
}

function f2(){
    //点击确认，返回值为true,否则为false
    var flag = confirm("确认吗?");
    if(flag){
        alert('你选择了确认');
    }else{
        alert('你选择了取消');
    }
}

function f3(){
    var msg = prompt('请输入手机号:');
    alert(msg);
}

</script>
</head>
<body style="font-size:30px;">

    <input type="button"
    value="打开窗口" onclick="f1();" />

    <input type="button"
    value="确认窗口" onclick="f2();" />

    <a href="del.do"
    onclick="return confirm('真的要删除吗?');">
    delete </a>

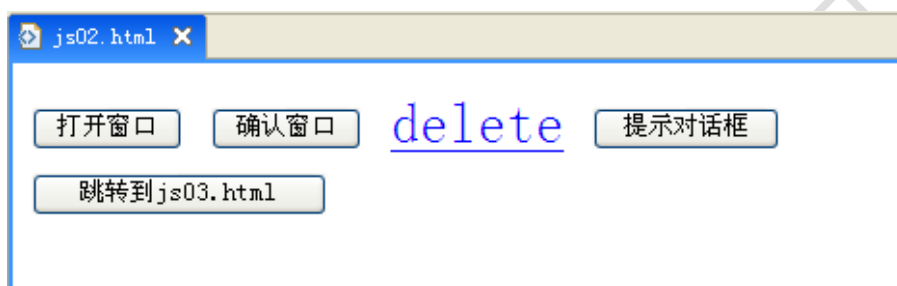
    <input type="button" value="提示对话框"
    onclick="f3();" />
```



```
<br/>

<input type="button"
value="跳转到js03.html"
onclick="location.href='js03.html'"/>

</body>
</html>
```



【案例 2.2】移动的 div

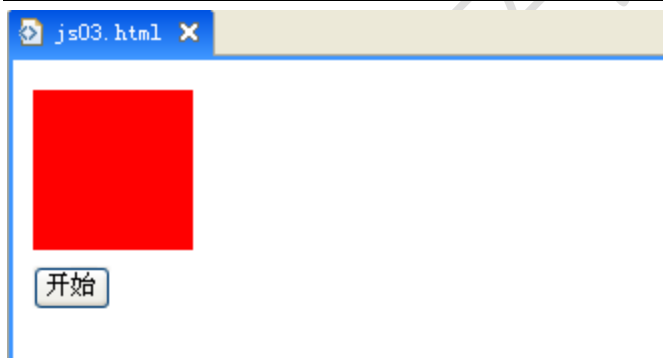
```
<html>
  <head>
    <style>
      #d1{
        width:80px;
        height:80px;
        background-color:red;
        position:relative;
        left:0px;
      }
    </style>
    <script src="prototype-1.6.0.3.js">
    </script>
    <script>
      function f1(){
        var taskId = setInterval(f2,500);
        setTimeout(function(){
          clearInterval(taskId);
```

```

        },5000);
    }
    function f2(){
        var v1 =
            parseInt($('d1').style.left);
        $('d1').style.left =
            v1 + 30 + 'px';
    }
</script>
</head>
<body style="font-size:30px;">
    <div id="d1" style="left:0px;"></div>
    <input type="button" value="开始"
    onclick="f1();" />

</body>
</html>

```



2.8.window 对象的属性 **

- 演示 1
location.href 加载新页面

```

1= <html>
2   <!--window-->
3=  <head>
4   </head>
5=  <body style="font-size:30px;">
6
7     <input type="button"
8     value="跳转到js03.html"
9     onclick="location.href='js03.html'"/>
10
11  </body>
12 </html>
13 |

```

【案例 2.3】navigator 属性

```

<html>
  <head>
    <script>
      function f1(){
        //window可以省略
        var nav = window.navigator;
        //for in循环，用于遍历对象的
        //属性名与属性值
        //每一次读取nav的一个属性名
        //nav[propName]:
        //访问nav的propName属性
        var msg = '浏览器的信息是<br/>';
        for(propName in nav){
          msg += '属性名:' + propName
            + ' 属性值:' + nav[propName]
            + '<br/>';
        }
        document.write(msg);
      }

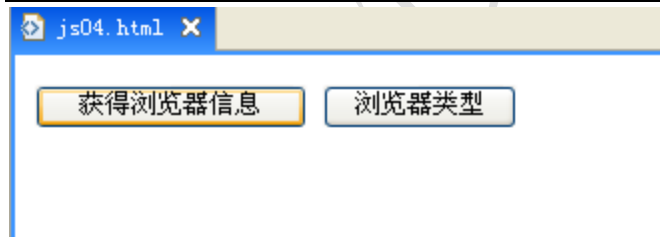
      function f2(){
        if(navigator.userAgent

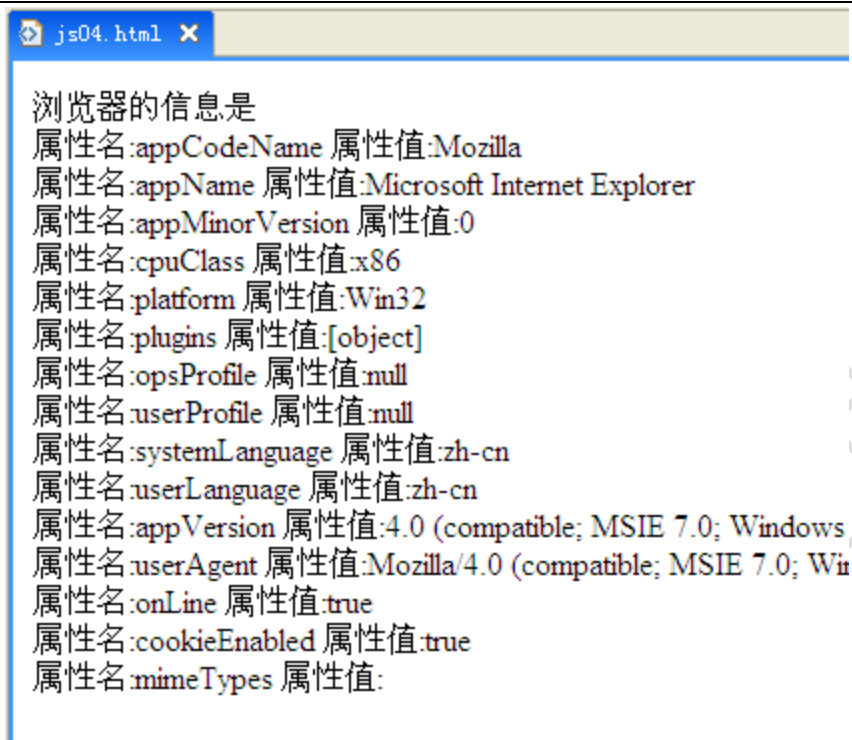
```

```

        .indexOf('Firefox') > 0){
            alert('你使用的是Firefox');
        }else if(navigator.userAgent
            .indexOf('MSIE') > 0){
            alert('你使用的是IE');
        }else{
            alert('你使用的是其它浏览器');
        }
    }
</script>
</head>
<body>
    <input type="button"
    value="获得浏览器信息"
    onclick="f1();" />
    <input type="button"
    value="浏览器类型"
    onclick="f2();" />
</body>
</html>

```





【案例 2.4】 opener 属性

1) js05.html

```
<html>
  <!--opener-->
  <head>
    <script>
      function f1(){
        var w =
          open('js06.html','w1',
            'width=400,height=400');

      }
    </script>
  </head>
  <body style="font-size:30px;">
    <a id="a1" href="javascript;">
```

```
onclick="f1();" >打开一个窗口</a>

</body>
</html>
```

2) js06.html

```
<html>
  <head>
    <script>
      function f1(){
        var obj =
          opener.document
            .getElementById('a1');
        alert(obj.innerHTML);
      }
    </script>
  </head>
  <body style="font-size:30px;">
    <a href="javascript:;"
      onclick="f1();" >click me</a>
  </body>
</html>
```

结果演示

1) 点击 js05.html “打开一个新窗口” 打开新窗口 js06.html



2) 点击打开的新窗口 js06.html 中的 “click me” , 弹出对话框显示 js05.html 页面的内容



【案例 2.5】parent 属性

1) js07.html

```
<html>
  <!--parent-->
  <head></head>
  <frameset rows="20%,*">
    <frame src="top.html" name="topFrame"/>
    <frameset cols="30%,*">
      <frame src="left.html" name="leftFrame"/>
      <frame src="main.html" name="mainFrame"/>
    </frameset>
  </frameset>
</html>
```

2) mail.html

```
<html>
  <!--parent-->
  <head></head>
  <body style="font-size:30px;">
  </body>
</html>
```

3) left.html

```
<html>
  <!--parent-->
```

```
<head></head>
<body style="font-size:30px;">
  <div id="d1">hello</div>
</body>
</html>
```

4) top.html

```
<html>
<!--parent-->
<head>
  <script>
    function f1(){
      var obj =
        parent.leftFrame
        .document.getElementById('d1');
      alert(obj.innerHTML);
    }
  </script>
</head>
<body style="font-size:30px;">
  <a href="javascript;"
    onclick="f1();">
    访问左边窗口中的某个节点</a>
</body>
</html>
```

结果演示

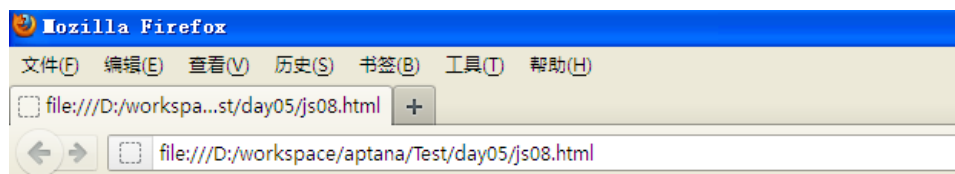
js07.html 由 3 个窗口组成 (top.html、left.html、main.html) , 点击 top.html 中的链接 , 可以访问到 left.html 中的页面内容



【案例 2.6】document 属性

```
<html>
  <head>
    <script>
      //如何改图片
      function change(){
        var obj =
          document.getElementById('img1');
        obj.src='t1.jpg';
      }
    </script>
  </head>
  <body style="font-size:30px;">
    

    <a href="javascript:;"
      onclick="change();">
      看看我长大了变成什么样了</a>
  </body>
</html>
```



看看我长大了变成什么样了

点击链接出现另外一张图片



看看我长大了变成什么样了

3. 事件处理机制 **

3.1. 事件是如何产生的 *

第一种情况，用户对网页做了某些操作，比如，点击了一个按钮，产生点击事件。

第二种情况，用户没有对网页做操作，也可能产生事件，比如浏览器已经将整个页面加载完毕，会产生加载完成事件。当事件产生以后，浏览器会查找产生事件的节点有没有绑定相应的事件处理代码。如果有，则调用该代码来处理。如果没有，会继续向上查找父节点，有没有对应的事件处理代码(事件冒泡)。

3.2. 绑定事件处理代码 **

1) 绑定事件处理代码到 html 标记之上

比如:

```
<a id="a1" href="" onclick="f1();" >click</a>
```

2) 绑定事件处理代码到 dom 节点之上

```
var obj = document.getElementById('a1');  
obj.onclick=f1;
```

注意:

a. f1 不要加"()",加"()"表示立即执行 f1 这个函数。

b. 可以使用匿名函数进行绑定。

即:

```
obj.onclick=function(){  
    //代码。  
}
```

c. 绑定事件处理代码到 dom 节点之上，可以将 js 代码与 html 完全分离，方便代码的维护。

d. 如果要给绑定的函数传参。可以使用匿名函数来解决。

即:

```
obj.onclick=function(){  
    f(参数);  
}
```

3) 使用浏览器自带的绑定方式 (了解)

不同的浏览器，有各自独有的绑定方式，因为不兼容，所以**尽量少用**。

● 演示 1

绑定事件处理代码到 html 标记之上

```

1 <html>
2   <head>
3     <script>
4       function f1(){
5         var obj =
6           document.getElementById('d1');
7         alert(obj.innerHTML);
8       }
9     </script>
10  </head>
11  <body style="font-size:30px;">
12    <div id="d1">hello</div>
13    <input id="b1" type="button"
14      value="Click" onclick="f1();" />
15  </body>
16 </html>
17

```



- 演示 2
绑定事件处理代码到 dom 节点之上

```

1=<html>
2=<head>
3=<script>
4=function f1(){
5=    var obj =
6=        document.getElementById('d1');
7=        alert(obj.innerHTML);
8=}
9=</script>
10</head>
11=<body style="font-size:30px;">
12=<div id="d1">hello</div>
13=<input id="b1" type="button"
14=    value="Click" />
15
16=<script>
17=    var b1Obj =
18=        document.getElementById("b1");
19=    b1Obj.onclick = f1;
20=</script>
21</body>
22</html>

```

✓ Script 脚本可以在<body>中写

● 演示 3

绑定事件处理代码到 dom 节点之上 (更好的方式)

1) Js09.html

```

1=<html>
2=<!--绑定事件处理代码-->
3=<head>
4=<script src="myjs.js"></script>
5</head>
6=<body style="font-size:30px;">
7=<div id="d1">hello</div>
8=<input id="b1" type="button"
9=    value="Click"/>
10</body>
11</html>
12

```

2) myjs.js

```

1  /* 当整个html文件加载完成之后
2     执行匿名函数。*/
3  window.onload = function() {
4      var b1Obj =
5      document.getElementById('b1');
6      b1Obj.onclick = f1;
7  }
8
9  function f1() {
10     var obj =
11     document.getElementById('d1');
12     alert(obj.innerHTML);
13 }
14

```

- 演示 4
匿名函数

```

1  /* 当整个html文件加载完成之后
2     执行匿名函数。*/
3  window.onload = function() {
4      var b1Obj =
5      document.getElementById('b1');
6      b1Obj.onclick = f1;
7  }
8

```

等价于

```

1  /* 当整个html文件加载完成之后
2     执行匿名函数。*/
3
4  function fx() {
5      var b1Obj =
6      document.getElementById('b1');
7      b1Obj.onclick = f1;
8  }
9
10 window.onload = fx
11

```

- 演示 5

如果传参数怎么办？使用匿名函数

1) Js09.html

```
1 <html>
2   <!--绑定事件处理代码-->
3   <head>
4     <script src="myjs2.js"></script>
5   </head>
6   <body style="font-size:30px;">
7     <div id="d1">hello</div>
8     <input id="b1" type="button"
9       value="Click"/>
10  </body>
11 </html>
12
```

2) myjs2.js

```
1 //当整个html文件加载完成之后
2 //执行匿名函数。
3 window.onload = function() {
4   var b1Obj =
5     document.getElementById('b1');
6   b1Obj.onclick = function() {
7     f1('world');
8   }
9 }
10
11 function f1(msg){
12   var obj = document.getElementById('d1');
13   alert(obj.innerHTML + " " + msg);
14 }
15
16
```

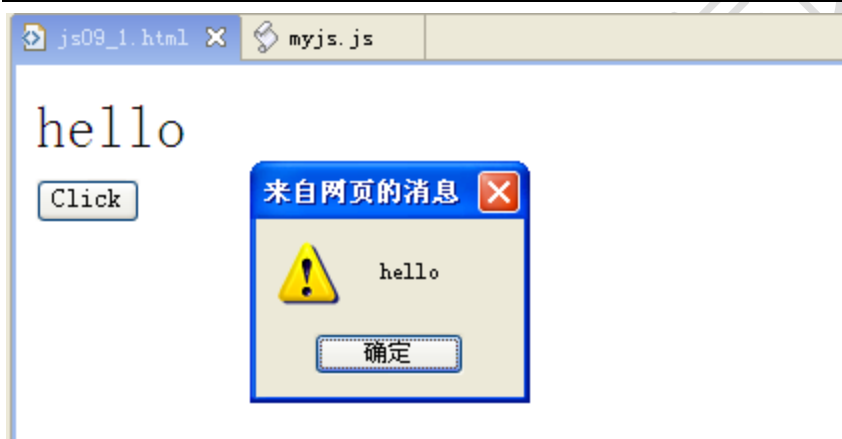
【案例 3.1】 绑定事件处理代码到 html 标记之上 **

```
<html>
  <!--绑定事件处理代码-->
  <head>
    <script>
      function f1(){
        var obj =
```

```

        document.getElementById('d1');
        alert(obj.innerHTML);
    }
</script>
</head>
<body style="font-size:30px;">
    <div id="d1">hello</div>
    <input id="b1" type="button"
        value="Click" onclick="f1();" />
</body>
</html>

```



【案例 3.2】绑定事件处理代码到 dom 节点之上_01 **

1) js09.html

```

<html>
  <!-- 绑定事件处理代码 -->
  <head>
    <script src="myjs.js"></script>
  </head>
  <body style="font-size:30px;">
    <div id="d1">hello</div>
    <input id="b1" type="button"
      value="Click"/>

```



```
</body>
</html>
```

2) myjs.js

```
/* 当整个html文件加载完成之后
   执行匿名函数。*/
window.onload = function(){
    var b1Obj =
        document.getElementById('b1');
    b1Obj.onclick = f1;
}

function f1(){
    var obj =
        document.getElementById('d1');
    alert(obj.innerHTML);
}
```

- ✓ **注意：**不能将 f1 写为 f1()，因为 f1() 表示立即执行该函数；f1 表示 b1Obj 上绑定一个函数 f1，由 onclick 来处理事件
- ✓ window.onload 表示页面加载完成后执行



【案例 3.3】绑定事件处理代码到 dom 节点之上_02 **

1) js09.html

```
<html>
  <!--绑定事件处理代码-->
  <head>
    <script src="myjs2.js"> </script>
  </head>
```

```
<body style="font-size:30px;">
  <div id="d1">hello</div>
  <input id="b1" type="button"
    value="Click"/>
</body>
</html>
```

2) myjs2.js

```
//当整个html文件加载完成之后
//执行匿名函数。
window.onload = function(){
  var b1Obj =
    document.getElementById('b1');
  b1Obj.onclick = function(){
    f1('world');
  }
}

function f1(msg){
  var obj = document.getElementById('d1');
  alert(obj.innerHTML + " " + msg);
}
```



3.3. 事件对象 ***

1) 获得事件对象

- ✓ IE 浏览器: 可以直接使用 event 获得
- ✓ firefox: 必须给方法添加一个参数 event

✓ 一般为了兼容 ie,firefox , 给方法添加一个参数 event

2) 事件对象的作用

a. 获得鼠标点击的坐标

event.clientX

event.clientY

b. 获得事件源(产生事件的那个对象)

firefox: **event.target** 获得

IE 浏览器: **event.srcElement** 获得

● 演示 1

IE 浏览器可运行, FireFox 会报错

```

2  <!--事件对象-->
3  <head>
4  <style>
5  #d1{
6      width:200px;
7      height:200px;
8      border:1px solid black;
9  }
10 </style>
11 <script>
12 //ie浏览器可以运行
13 function f1(){
14     alert(event.clientX + ':' +
15           event.clientY);
16 }
17 </script>
18 </head>
19 <body style="font-size:30px;">
20 <div id="d1" onclick="f1();">
21 </div>
22 </body>
23 </html>

```



● 演示 2

IE 浏览器和火狐都可以运行

```

2  <!--事件对象-->
3  <head>
4  <style>
5      #d1{ width:200px;
6          height:200px;
7          border:1px solid black; }
8      #d2{ width:200px;
9          height:200px;
10         border:1px solid black; }
11 </style>
12 <script>
13     //firefox,ie通用。
14     function f2(e){
15         alert(e.clientX + ':'
16             + e.clientY);
17     }
18 </script>
19 </head>
20 <body style="font-size:30px;">
21     <div id="d1" onclick="f2(event) ;">
22     </div>
23 </body>

```

● 演示 3

e.target (FF) 和 e.srcElement (IE)

```

1 <html>
2   <!--事件对象-->
3   <head>
4     <style>
5       #d1{
6         width:200px;
7         height:200px;
8         border:1px solid black;
9       }
10      #d2{
11        width:200px;
12        height:200px;
13        border:1px solid black;
14      }
15    </style>
16    <script>
17      function f3(e){
18        /* 演示1: firefox能用 */
19        //var obj = e.target;
20        /* 演示2: ie能用 */
21        //var obj = e.srcElement;
22        /* 演示3: 兼容ie和firefox */
23        var obj = e.target || e.srcElement;
24        alert(obj.innerHTML);
25      }
26    </script>
27  </head>
28  <body style="font-size:30px;">
29    <a href="javascript:;"
30      onclick="f3(event) ">点我</a><br/>
31
32    <a href="javascript:;"
33      onclick="f3(event) ">点你</a><br/>
34  </body>
35 </html>

```



3.4. 事件冒泡 **

1) 什么是事件冒泡？

当一个节点产生事件以后，该事件会依次向上传递(先传给父节点，如果父节点还有父节点，再向上传递)。

2) 如何禁止冒泡？

`event.cancelBubble = true;`

● 演示 1

事件冒泡

```

1 <html>
2   <!--事件对象-->
3   <head>
4     <style>
5       #d1{
6         width:200px;
7         height:200px;
8         border:1px solid black;
9       }
10      #d2{
11        width:200px;
12        height:200px;
13        border:1px solid black;
14      }

```

```

15         </style>
16         <script>
17             function f4(e){
18                 alert("你点击了一个链接");
19             }
20
21             function f5(e){
22                 alert("你点击了一个div");
23             }
24         </script>
25     </head>
26     <body style="font-size:30px;">
27         <div id="d2" onclick="f5(event) ">
28             <a href="javascript:;"
29                 onclick="f4(event) ">点我</a>
30         </div>
31     </body>
32 </html>
33

```

结果演示

点击 div 中的链接 “点我”

1) 出现对话框 “你点击了一个链接”



2) “事件冒泡” 现象

关闭对话框 “你点击了一个链接” ，继续弹出对话框 “你点击了一个 div”



- 演示 2
解决事件冒泡问题

```

16  <script>
17      function f4(e){
18          alert("你点击了一个链接");
19          event.cancelBubble = true;
20      }
21
22      function f5(e){
23          alert("你点击了一个div");
24      }
25  </script>
26  </head>
27  <body style="font-size:30px;">
28      <div id="d2" onclick="f5(event)">
29          <a href="javascript:;"
30              onclick="f4(event)">点我</a>
31      </div>
32  </body>
33 </html>
    
```

【案例 3.4】事件对象 **

```

<html>
  <!--事件对象-->
    
```



```
<head>
  <style>
    #d1{
      width:200px;
      height:200px;
      border:1px solid black;
    }
    #d2{
      width:200px;
      height:200px;
      border:1px solid black;
    }
  </style>
  <script>
    //ie浏览器可以运行
    function f1(){
      alert(event.clientX + ':' +
        event.clientY);
    }

    //firefox,ie通用。
    function f2(e){
      alert(e.clientX + ':' +
        e.clientY);
    }

    function f3(e){
      //var obj = e.target; 只有firefox能用
      //var obj = e.srcElement; //ie能用
      var obj = e.target || e.srcElement;
      alert(obj.innerHTML);
    }

    function f4(e){
      alert("你点击了一个链接");
      e.cancelBubble = true;
    }
  </script>
</head>
```

```

        function f5(e){
            alert("你点击了一个div");
        }
    </script>
</head>
<body style="font-size:30px;">

    <div id="d1" onclick="f2(event);">
    </div>

    <a href="javascript:;"
        onclick="f3(event);">点我</a> <br/>
    <a href="javascript:;"
        onclick="f3(event)">点你</a> <br/>

    <hr/>
    <div id="d2" onclick="f5(event);">
        <a href="javascript:;"
            onclick="f4(event);">点我</a>
    </div>
</body>
</html>

```

4. 面向对象基础 *

4.1. 如何定义一个类型 *

Js 和 Java 不同，JS 没有定义类型的专门的语法，但是，JS 可以利用函数来定义一个类型。

4.2. 如何创建一个 js 对象？ *

1) 使用 new 关键字创建

```
var p = new Person("zs",22);
```

p.play();

2) 使用 json 语法创建 (今天仅作了解)

```
var p = {"name":"zs","age":22};  
var p2 = {"name":"ls",  
          "address":{  
            "city":"beijing",  
            "street":"gjie"  
          }  
};
```

注意：

- 属性值如果是字符串必须用"引起来。如果属性值不是字符串，不要用"。
- 属性值的类型可以是：
string,number,boolean,null,object。

3) 使用 Object

Object 是所有 js 类型的父类。

```
var p = new Object();
```

//js 是一种动态语言，可以在运行时，

//为对象增加新的属性和方法。

```
p.name = "zs";
```

```
p.age = 22;
```

```
p.play = function(){ alert('hello'); };
```

● 演示 1

使用 new 关键字创建对象

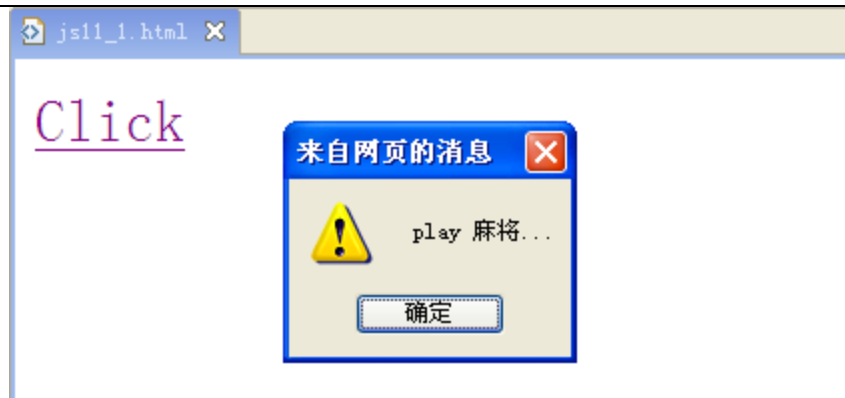
```

1 <html>
2   <!--js面向对象的基础-->
3   <head>
4     <script>
5       /* 定义了一个Person类型 */
6       function Person(name,age){
7         /* this修饰的变量称为属性
8            属性没有修饰符(private...) */
9         this.name = name;
10        this.age = age;
11
12        /* 如果属性的值是一个方法, 此时
13           该属性可以认为是该类型的一个方法 */
14        this.play = function(){
15          alert('play 麻将...');
16        };
17      }
18
19      /* 1. 使用new关键字, 创建类型的实例 */
20      function f1(){
21
22        var p = new Person("zs",22);
23        /* 1) 调用Person p的方法 */
24        p.play();
25        /* 2) 调用Person p的属性 */
26        alert(p.name + ' ' + p.age);
27      }
28    </script>
29  </head>
30  <body style="font-size:30px;">
31    <a href="javascript:;"
32      onclick="f1();">Click</a>
33  </body>
34 </html>
35

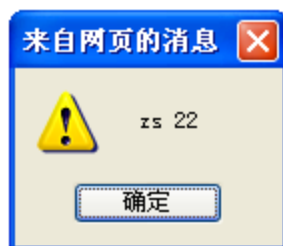
```

结果演示

1) 弹出对话框 “play 麻将”



2) 弹出对话框 “zs 22”



- 演示 2
使用 JSON 创建对象

```

4  <script>
5      //使用json语法创建js对象|
6      function f2(){
7          var p = {"name":"zs","age":22};
8          //alert(p.name);
9
10         var p2 = {"name":"ls",
11                   "address":{"
12                       "city":"beijing",
13                       "street":"gjie"
14                   }}
15         };
16         alert(p2.address.city + ' ' +
17               p2.address.street);
18     }
19 </script>
20 </head>
21 <body style="font-size:30px;">
22     <a href="javascript:;"
23       onclick="f2();">Click</a>
24 </body>
25 </html>

```

Click



- 演示 3
使用 Object 创建 JS 对象

```

1 <html>
2   <!--js面向对象的基础-->
3   <head>
4     <script>
5       //使用Object创建js对象
6       function f3(){
7         var p = new Object();
8         p.name = "zs";
9         p.age = 22;
10        p.play = function(){
11          alert('play...');
12        }
13        alert(p.name);
14        p.play();
15      }
16
17    </script>
18  </head>
19  <body style="font-size:30px;">
20    <a href="javascript:;"
21      onclick="f3();">Click</a>
22  </body>

```

js11_3.html x js11_2.html

Click



【案例 4】js 面向对象的基础 *

```

<html>
  <!--js面向对象的基础-->
  <head>
    <script>

```

```

/* 定义了一个Person类型 */
function Person(name,age){
    /* this修饰的变量称为属性
       属性没有修饰符(private...) */
    this.name = name;
    this.age = age;

    /* 如果属性的值是一个方法，此时
       该属性可以认为是该类型的一个方法 */
    this.play = function(){
        alert('play 麻将...');
    };
}

/* 1. 使用new关键字，创建类型的实例 */
function f1(){

    var p = new Person("zs",22);
    /* 1) 调用Person p的方法 */
    //p.play();
    /* 2) 调用Person p的属性 */
    alert(p.name + ' ' + p.age);
}

/* 2. 使用json语法创建js对象 */
function f2(){
    var p = {"name":"zs","age":22};
    //alert(p.name);

    var p2 = {"name":"ls",
               "address":{
                   "city":"beijing",
                   "street":"gjie"
               }
    };
    alert(p2.address.city + ' ' +
          p2.address.street);
}

```



```

/* 3. 使用Object创建js对象 */
function f3(){
    var p = new Object();
    p.name = "zs";
    p.age = 22;
    p.play = function(){
        alert('play...');
    }
    //alert(p.name);
    p.play();
}

</script>
</head>
<body style="font-size:30px;">
    <a href="javascript;"
    onclick="f3();">Click</a>
</body>
</html>

```

达内IT培训集团