JS小记

1.onload事件（html的知识）

页面加载之后立即执行一段 JavaScript

<body onload="load()">

所有主流浏览器都支持 onload 属性

onload 属性在对象已加载时触发。

onload 常用在 <body> 中，一旦完全加载所有内容（包括图像、脚本文件、CSS 文件等），就执行一段脚本。

2. **JS放在head和放在body中的区别**

下面分别是head中、body中、html后

结果是，先执行head中js、body中按顺序执行（出现Test1后就执行该js）、html页面加载完成后执行html后的js。

总的来说，就是顺序执行。

<html>

<head>

<script type="text/javascript">

alert("<head>中代码被执行")

</script>

</head>

<body>

<h1 align="center">Test1</h1>

<!-- action内容作为Servlet映射的url名的一部分，

浏览器发出的HTTP请求，会在前面追加相对于其所在的页面,

比如 /Beer1/SelectBeer.do -->

<script type="text/javascript">

alert("<body>中代码被执行")

</script>

<form method="post" action="SelectBeer.do">

Select beer<p>

Color:

<select name="color" size="1">

<option>light

<option>red

<option>green

<option>yellow

</select>

<br>

<br>

<center>

<input type="Submit">

</center>

</form>

</body>

</html>

<script>

document.write("hello");

</script>

## 3. onclick定义和用法

onclick 事件会在对象被点击时发生。

<button class=*"btn btn-default"* onclick="showRemoteDialog()" type=*"button"*>

加载远程数据

</button>

### 支持该事件的 HTML 标签：

<a>, <address>, <area>, <b>, <bdo>, <big>, <blockquote>, <body>, <button>,

<caption>, <cite>, <code>, <dd>, <dfn>, <div>, <dl>, <dt>, <em>, <fieldset>,

<form>, <h1> to <h6>, <hr>, <i>, <img>, <input>, <kbd>, <label>, <legend>,

<li>, <map>, <object>, <ol>, <p>, <pre>, <samp>, <select>, <small>, <span>,

<strong>, <sub>, <sup>, <table>, <tbody>, <td>, <textarea>, <tfoot>, <th>,

<thead>, <tr>, <tt>, <ul>, <var>

### 支持该事件的 JavaScript 对象：

button, document, checkbox, link, radio, reset, submit

示例：

<html>

<body>

Field1: <input type="text" id="field1" value="Hello World!">

<br />

Field2: <input type="text" id="field2">

<br /><br />

点击下面的按钮，把 Field1 的内容拷贝到 Field2 中：

<br />

<button onclick="document.getElementById('field2').value=

document.getElementById('field1').value">Copy Text</button>

</body>

</html>

4. 如果需要在页面加载过程中动态构建内容，js必须要放在body里（在页面加载的过程中），而且要放在最后。一者如果js里在载入时有控制dom元素的逻辑，**需要等元素加载完毕才能识别**。其次也是对加载性能的考虑

## 5. 事件句柄　(Event Handlers)

HTML 4.0 的新特性之一是能够使 HTML 事件触发浏览器中的行为，比如当用户点击某个 HTML 元素时启动一段 JavaScript。下面是一个属性列表，可将之插入 HTML 标签以定义事件的行为。

|  |  |
| --- | --- |
| **属性** | **此事件发生在何时...** |
| [onabort](http://www.w3school.com.cn/jsref/event_onabort.asp) | 图像的加载被中断。 |
| [onblur](http://www.w3school.com.cn/jsref/event_onblur.asp) | 元素失去焦点。 |
| [onchange](http://www.w3school.com.cn/jsref/event_onchange.asp) | 域的内容被改变。 |
| [onclick](http://www.w3school.com.cn/jsref/event_onclick.asp) | 当用户点击某个对象时调用的事件句柄。 |
| [ondblclick](http://www.w3school.com.cn/jsref/event_ondblclick.asp) | 当用户双击某个对象时调用的事件句柄。 |
| [onerror](http://www.w3school.com.cn/jsref/event_onerror.asp) | 在加载文档或图像时发生错误。 |
| [onfocus](http://www.w3school.com.cn/jsref/event_onfocus.asp) | 元素获得焦点。 |
| [onkeydown](http://www.w3school.com.cn/jsref/event_onkeydown.asp) | 某个键盘按键被按下。 |
| [onkeypress](http://www.w3school.com.cn/jsref/event_onkeypress.asp) | 某个键盘按键被按下并松开。 |
| [onkeyup](http://www.w3school.com.cn/jsref/event_onkeyup.asp) | 某个键盘按键被松开。 |
| [onload](http://www.w3school.com.cn/jsref/event_onload.asp) | 一张页面或一幅图像完成加载。 |
| [onmousedown](http://www.w3school.com.cn/jsref/event_onmousedown.asp) | 鼠标按钮被按下。 |
| [onmousemove](http://www.w3school.com.cn/jsref/event_onmousemove.asp) | 鼠标被移动。 |
| [onmouseout](http://www.w3school.com.cn/jsref/event_onmouseout.asp) | 鼠标从某元素移开。 |
| [onmouseover](http://www.w3school.com.cn/jsref/event_onmouseover.asp) | 鼠标移到某元素之上。 |
| [onmouseup](http://www.w3school.com.cn/jsref/event_onmouseup.asp) | 鼠标按键被松开。 |
| [onreset](http://www.w3school.com.cn/jsref/event_onreset.asp) | 重置按钮被点击。 |
| [onresize](http://www.w3school.com.cn/jsref/event_onresize.asp) | 窗口或框架被重新调整大小。 |
| [onselect](http://www.w3school.com.cn/jsref/event_onselect.asp) | 文本被选中。 |
| [onsubmit](http://www.w3school.com.cn/jsref/event_onsubmit.asp) | 确认按钮被点击。 |
| [onunload](http://www.w3school.com.cn/jsref/event_onunload.asp) | 用户退出页面。 |

**鼠标 / 键盘属性**

|  |  |
| --- | --- |
| **属性** | **描述** |
| [altKey](http://www.w3school.com.cn/jsref/event_altkey.asp) | 返回当事件被触发时，"ALT" 是否被按下。 |
| [button](http://www.w3school.com.cn/jsref/event_button.asp) | 返回当事件被触发时，哪个鼠标按钮被点击。 |
| [clientX](http://www.w3school.com.cn/jsref/event_clientx.asp) | 返回当事件被触发时，鼠标指针的水平坐标。 |
| [clientY](http://www.w3school.com.cn/jsref/event_clienty.asp) | 返回当事件被触发时，鼠标指针的垂直坐标。 |
| [ctrlKey](http://www.w3school.com.cn/jsref/event_ctrlkey.asp) | 返回当事件被触发时，"CTRL" 键是否被按下。 |
| [metaKey](http://www.w3school.com.cn/jsref/event_metakey.asp) | 返回当事件被触发时，"meta" 键是否被按下。 |
| [relatedTarget](http://www.w3school.com.cn/jsref/event_relatedtarget.asp) | 返回与事件的目标节点相关的节点。 |
| [screenX](http://www.w3school.com.cn/jsref/event_screenx.asp) | 返回当某个事件被触发时，鼠标指针的水平坐标。 |
| [screenY](http://www.w3school.com.cn/jsref/event_screeny.asp) | 返回当某个事件被触发时，鼠标指针的垂直坐标。 |
| [shiftKey](http://www.w3school.com.cn/jsref/event_shiftkey.asp) | 返回当事件被触发时，"SHIFT" 键是否被按下。 |

# 6. [HTML和JSP应用入门篇（HTML页面获取数据，JSP页面输出数据）](http://www.cnblogs.com/SCAU_que/articles/1779977.html)

1，创建HTML文件，来获取用户输入的数据：

[复制代码](javascript:void(0);)

1 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">  
 2  <html>  
 3 <head>  
 4 <title>用户注册</title>  
 5   
 6 <meta http-equiv="keywords" content="keyword1,keyword2,keyword3">  
 7 <meta http-equiv="description" content="this is my page">  
 8 <meta http-equiv="content-type" content="text/html; charset=GBK">  
 9   
10 <!--<link rel="stylesheet" type="text/css" href="./styles.css">-->  
11   
12 </head>  
13   
14 <body>  
15 <img src="qq1.jpg">  
16 <table>  
17 <tr>  
18 <td>  
19 <img src="sign.JPG">  
20 </td>  
21 <td>  
22 <b >会员注册</b> <br><br>  
23   
24 <form action="index.jsp" name="form1" method="post"> //使用http post请求传输表单数据  
25 用户姓名  
26 <input type="text" name="username">  
27 <br>  
28 &nbsp;  
29 <br>  
30 填写密码  
31 <input type="password" name="pass">  
32 <br>  
33 &nbsp;  
34 <br>  
35 性别  
36 男<input type="radio" checked="checked" name="sex" value="男">  
37 女<input type="radio" name="sex" value="女">  
38 <br>  
39 &nbsp;  
40 <br>  
41 爱好  
42 <input type="checkbox" name="love" value="swimming">游泳  
43 <input type="checkbox" name="love" value="basketball">打球  
44 <br>  
45 &nbsp;  
46 <br>  
47 <input type="submit" name="submit" value="确认提交">   
48 </form>  
49 </td>  
50 </tr>  
51 </table>  
52 </body>  
53  </html>  
54

2，编写JSP页面，**接受HTML页面中表单数据**，显示到浏览器上：

1 <%@ page language="java" import="java.util.\*" pageEncoding="GBK"%>  
 2  <%  
 3  String path = **request**.getContextPath();  
 4 String basePath = request.getScheme()+"://"+request.getServerName()+":"+request.getServerPort()+path+"/";  
 5 %>  
 6   
 7 <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">  
 8 <html>  
 9 <head>  
10 <base href="<%=basePath%>">  
11   
12 <title>My JSP 'index.jsp' starting page</title>  
13 <meta http-equiv="pragma" content="no-cache">  
14 <meta http-equiv="cache-control" content="no-cache">  
15 <meta http-equiv="expires" content="0">   
16 <meta http-equiv="keywords" content="keyword1,keyword2,keyword3">  
17 <meta http-equiv="description" content="This is my page">  
18 <!--  
19 <link rel="stylesheet" type="text/css" href="styles.css">  
20 -->  
21 </head>  
22   
23 <body>  
24 <%  
25 String name=request.getParameter("username"); //使用内置对象获取request的数据  
26 String Newname=new String(name.getBytes("ISO-8859-1"),"GBK");  
27 out.print(Newname); //实际业务中将进行业务操作，这里只是示范获取数据而已  
28 out.print("<br>");  
29 String sex=request.getParameter("sex"); //使用内置对象获取request的数据  
30 String Newsex=new String(sex.getBytes("ISO-8859-1"),"GBK");  
31 out.print(Newsex);  
32 out.print("<br>");  
33 String[] love=request.getParameterValues("love"); //使用内置对象获取request的数据  
34 for(int i=0;i<love.length;i++) {  
35 out.print(love[i]+" ");  
36 }  
37   
38   
39 %>  
40 <br>  
41 </body>  
42 </html>  
43

7.**查找 HTML 元素**

通常，通过 JavaScript，您需要操作 HTML 元素。

为了做到这件事情，您必须首先找到该元素。有三种方法来做这件事：

* 通过 id 找到 HTML 元素 document.getElementById(“id”)
* 通过标签名找到 HTML 元素 document.getElementByTagName(“div”)
* 通过类名找到 HTML 元素 document.getElementByClassName(“class”)

# 8. JavaScript HTML DOM

通过 HTML DOM，可访问 JavaScript HTML 文档的所有元素。

当网页被加载时，**浏览器会创建**页面的文档对象模型（Document Object Model）。



通过可编程的对象模型，JavaScript 获得了足够的能力来创建动态的 HTML。

* JavaScript 能够改变页面中的所有 HTML **元素**
* JavaScript 能够改变页面中的所有 HTML **属性**
* JavaScript 能够改变页面中的所有 **CSS** 样式
* JavaScript 能够对页面中的所有**事件**做出反应

# 9. JavaScript HTML DOM - 改变 HTML 内容

HTML DOM 允许 JavaScript 改变 HTML 元素的**内容**。

## 改变 HTML 输出流

JavaScript 能够创建动态的 HTML 内容：

**今天的日期是： Mon Feb 20 2017 11:08:26 GMT+0800**

在 JavaScript 中，document.write() 可用于直接向 **HTML 输出流**写内容。

## 实例

<!DOCTYPE html>  
<html>  
<body>  
<script>  
document.write(Date());  
</script>  
</body>  
</html>

|  |  |
| --- | --- |
| **lamp** | 绝对不要在文档加载完成之后使用 document.write()。这会覆盖该文档。 |

## 改变 HTML 内容

修改 HTML 内容的最简单的方法时使用 **innerHTML 属性**。

如需改变 HTML 元素的内容，请使用这个语法：

document.getElementById(*id*).innerHTML=*新的 HTML*

本例改变了 <p>元素的内容：

## 实例

<html>  
<body>  
<p id="p1">Hello World!</p>  
<script>  
document.getElementById("p1").innerHTML="新文本!";  
</script>  
</body>  
</html>

本例改变了 <h1> 元素的内容：

## 实例

<!DOCTYPE html>  
<html>  
<body>  
<h1 id="header">Old Header</h1>  
<script>  
var element=document.getElementById("header");  
element.innerHTML="新标题";  
</script>  
</body>  
</html>

## 改变 HTML 属性

如需改变 HTML 元素的属性，请使用这个语法：

document.getElementById(*id*).***attribute****=新属性值*

本例改变了 <img> 元素的 src 属性：

## 实例

<!DOCTYPE html>  
<html>  
<body>  
<img id="image" src="smiley.gif">  
<script>  
document.getElementById("image").src="landscape.jpg";  
</script>  
</body>  
</html>

# 10. JavaScript HTML DOM - 改变CSS

HTML DOM 允许 JavaScript 改变 HTML 元素的**样式**。

## 改变 HTML 样式

如需改变 HTML 元素的样式，请使用这个语法：

document.getElementById(*id*).style.*property*=*新样式*

下面的例子会改变 <p> 元素的样式：

## 实例

<!DOCTYPE html> <html> <head> <meta charset="utf-8"> <title>菜鸟教程(runoob.com)</title> </head> <body> <p id="p1">Hello World!</p> <p id="p2">Hello World!</p> <script> document.getElementById("p2").style.color="blue"; document.getElementById("p2").style.fontFamily="Arial"; document.getElementById("p2").style.fontSize="larger"; </script> <p>以上段落通过脚本修改。</p> </body> </html>

## 使用事件

HTML DOM 允许我们通过触发事件来执行代码。

比如以下事件：

* 元素被点击。
* 页面加载完成。
* 输入框被修改。

本例改变了 id="id1" 的 HTML 元素的样式，当用户点击按钮时：

## 实例

<!DOCTYPE html>  
<html>  
<body>  
  
<h1 id="id1">我的标题 1</h1>  
<button type="button"   
onclick="document.getElementById('id1').style.color='red'">  
点我!</button>  
  
</body>  
</html>

# 11. JavaScript HTML DOM 事件

HTML DOM 使 JavaScript 有能力对 HTML 事件做出反应。

## 实例

**Mouse Over Me**

**Click Me**

## 对事件做出反应

我们可以在事件发生时执行 JavaScript，比如当用户在 HTML 元素上点击时。

如需在用户点击某个元素时执行代码，请向一个 HTML 事件属性添加 JavaScript 代码：

onclick=*JavaScript*

HTML 事件的例子：

* 当用户点击鼠标时
* 当网页已加载时
* 当图像已加载时
* 当鼠标移动到元素上时
* 当输入字段被改变时
* 当提交 HTML 表单时
* 当用户触发按键时

在本例中，当用户在 <h1> 元素上点击时，会改变其内容：

## 实例

<!DOCTYPE html>  
<html>  
<body>  
<h1 onclick="this.innerHTML='Ooops!'">点击文本!</h1>  
</body>  
</html>

[尝试一下 »](http://www.runoob.com/try/try.php?filename=trydhtml_event_onclick2)

本例从事件处理器调用一个函数：

## 实例

<!DOCTYPE html>  
<html>  
<head>  
<script>  
function **changetext**(id)  
{  
    id.innerHTML="Ooops!";  
}  
</script>  
</head>  
<body>  
<h1 **onclick**="**changetext**(**this**)">点击文本!</h1> //this指定当前标签h1  
</body>  
</html>

[尝试一下 »](http://www.runoob.com/try/try.php?filename=trydhtml_event_onclick3)

## HTML 事件属性

如需向 HTML 元素分配 事件，您可以使用事件属性。

## 实例

向 button 元素分配 onclick 事件：

<button onclick="displayDate()">点这里</button>

[尝试一下 »](http://www.runoob.com/try/try.php?filename=tryjs_events1)

在上面的例子中，名为 displayDate 的函数将在按钮被点击时执行。

## 使用 HTML DOM 来分配事件

HTML DOM 允许您**使用 JavaScript 来向 HTML 元素分配事件**：

## 实例

向 button 元素分配 onclick 事件：

<script>  
document.getElementById("myBtn")**.onclick**=function(){displayDate()}; //强行dom别人的属性  
</script>

[尝试一下 »](http://www.runoob.com/try/try.php?filename=tryjs_events2)

在上面的例子中，名为 displayDate 的函数被分配给 id="myBtn" 的 HTML 元素。

按钮点击时Javascript函数将会被执行。

## onload 和 onunload 事件

onload 和 onunload 事件会在用户**进入或离开页面时被触发**。

onload 事件可用于检测访问者的浏览器类型和浏览器版本，并基于这些信息来加载网页的正确版本。

onload 和 onunload 事件可用于处理 cookie。

## 实例

<body onload="checkCookies()">

[尝试一下 »](http://www.runoob.com/try/try.php?filename=tryjs_events_onload)

## onchange 事件

onchange 事件常结合对输入字段的验证来使用。

下面是一个如何使用 onchange 的例子。当用户改变输入字段的内容时，会调用 upperCase() 函数。

## 实例

<input type="text" id="fname" **onchange**="upperCase()">

[尝试一下 »](http://www.runoob.com/try/try.php?filename=tryjsref_onchange)

## onmouseover 和 onmouseout 事件

onmouseover 和 onmouseout 事件可用于在用户的鼠标移至 HTML 元素上方或移出元素时触发函数。

## 实例

一个简单的 onmouseover-onmouseout 实例：

**Mouse Over Me**

[尝试一下 »](http://www.runoob.com/try/try.php?filename=tryjs_events_mouseover)

## onmousedown、onmouseup 以及 onclick 事件

onmousedown, onmouseup 以及 onclick 构成了鼠标点击事件的所有部分。首先当点击鼠标按钮时，会触发 onmousedown 事件，当释放鼠标按钮时，会触发 onmouseup 事件，最后，当完成鼠标点击时，会触发 onclick 事件。

## 实例

一个简单的 onmousedown-onmouseup 实例：

**Thank You**

## 更多实例

[onmousedown 和onmouseup](http://www.runoob.com/try/try.php?filename=trydhtml_event_onmousedown)  
当用户按下鼠标按钮时，更换一幅图像。

[onload](http://www.runoob.com/try/try.php?filename=trydhtml_event_onload)  
当页面完成加载时，显示一个提示框。

[onfocus](http://www.runoob.com/try/try.php?filename=tryjsref_onfocus)  
当输入字段获得焦点时，改变其背景色。

[鼠标事件](http://www.runoob.com/try/try.php?filename=trydhtml_event_onmouse)  
当指针移动到元素上方时，改变其颜色；当指针移出文本后，会再次改变其颜色。

# 12. JavaScript HTML DOM EventListener

**addEventListener() 方法**

Q：**为什么**使用addEventListener方法呢，

**而不**直接使用element.getElementById(“xxx”).document.getElementById("myBtn")

**.onclick=function() { displayDate() }**;？

A：因为直接DOM然后调用onclick属性进行事件赋值时，是绝对化地赋值，即属性值只能有一个，我们就不能添加多个事件属性值了，也就不能触发多个事件，而使用addEventListener方法，它作为事件添加器，可以为元素的事件属性添加多个属性值，因此可以触发多个事件，有种回调的感觉。

**代码示例**：

**不使用时**：

<p>点击按钮执行 <em>displayDate()</em> 函数.</p>

<button id="myBtn">点这里</button>

<script>

document.getElementById("myBtn").onclick=function(){

displayDate()

};

function displayDate(){

document.getElementById("demo").innerHTML=Date();

}

</script>

<p id="demo"></p>

**使用后**：可以添加多个事件句柄

<p>该实例使用 addEventListener() 方法向同个按钮中添加两个点击事件。</p>

<button id="myBtn">点我</button>

<script>

var x = document.getElementById("myBtn"); //dom到id为myBtn的标签

x.addEventListener("**click**", myFunction); //给该标签 添加多个事件句柄

x.addEventListener("**click**", someOtherFunction);

function myFunction() {

alert ("Hello World!")

}

function someOtherFunction() {

alert ("函数已执行!")

}

</script>

## addEventListener() 方法

## 实例

在用户点击按钮时触发监听事件：

document.getElementById("myBtn").addEventListener("click", displayDate);

[尝试一下 »](http://www.runoob.com/try/try.php?filename=tryjs_addeventlistener_displaydate)

addEventListener() 方法用于向指定元素添加事件句柄。

addEventListener() 方法添加的事件句柄**不会覆盖已存在的事件句柄**。

你**可以向一个元素添加多个事件句柄**。

你**可以**向同个元素**添加多个同类型**的事件句柄，如：两个 "click" 事件。

你**可以向任何 DOM 对象**添加事件监听，**不仅仅是 HTML** 元素。**如： window 对象。**

addEventListener() 方法可以更简单的控制事件（冒泡与捕获）。

当你使用 addEventListener() 方法时, JavaScript 从 HTML 标记中分离开来，可读性更强， 在没有控制HTML标记时也可以添加事件监听。

你可以使用 removeEventListener() 方法来移除事件的监听。

## 语法

element.addEventListener(event, function, useCapture);

第一个参数是事件的类型 (如 "click" 或 "mousedown").

第二个参数是事件触发后调用的函数。

第三个参数是个布尔值用于描述事件是冒泡还是捕获。该参数是可选的。

|  |  |
| --- | --- |
| **Note** | 注意:不要使用 "on" 前缀。 例如，使用 "click" ,而不是使用 "onclick"。 |

## 向原元素添加事件句柄

## 实例

当用户点击元素时弹出 "Hello World!" :

element.addEventListener("click", function(){ alert("Hello World!"); });

[尝试一下 »](http://www.runoob.com/try/try.php?filename=tryjs_addeventlistener_add)

你可以使用函数名，来引用外部函数:

## 实例

当用户点击元素时弹出 "Hello World!" :

element.addEventListener("click", myFunction);  
  
function myFunction() {  
    alert ("Hello World!");  
}

[尝试一下 »](http://www.runoob.com/try/try.php?filename=tryjs_addeventlistener_add2)

## 向 Window 对象添加事件句柄

addEventListener() 方法允许你在 HTML DOM 对象添加事件监听， **HTML DOM 对象如： HTML 元素, HTML 文档, window 对象**。或者**其他支出的事件对象如: xmlHttpRequest 对象。**

## 实例

当用户重置窗口大小时添加事件监听：

window.addEventListener("resize", function(){  
    document.getElementById("demo").innerHTML = sometext;  
});

[尝试一下 »](http://www.runoob.com/try/try.php?filename=tryjs_addeventlistener_dom)

## 传递参数

当传递参数值时，使用"匿名函数"调用带参数的函数：

## 实例

element.addEventListener("click", function(){ myFunction(p1, p2); });

[尝试一下 »](http://www.runoob.com/try/try.php?filename=tryjs_addeventlistener_parameters)

## 事件冒泡或事件捕获？

**事件传递有两种方式**：**冒泡**与**捕获**。

事件传递定义了元素**事件触发的顺序**。 如果你将 <p> 元素插入到 <div> 元素中，用户点击 <p> 元素, 哪个元素的 "click" 事件先被触发呢？

**在 冒泡 中**，**内部元素的事件会先被触发**，然后再触发外部元素，即： <p> 元素的点击事件先触发，然后会触发 <div> 元素的点击事件。

**在 捕获 中，外部元素的事件会先被触发**，然后才会触发内部元素的事件，即： <div> 元素的点击事件先触发 ，然后再触发 <p> 元素的点击事件。

addEventListener() 方法可以**指定 "useCapture" 参数来设置传递类型**：

addEventListener(event, function, useCapture);

**默认值为 false, 即冒泡传递**，当值为 **true 时**, 事件使用**捕获传递**。

## 实例

document.getElementById("myDiv").addEventListener("click", myFunction, true);

[尝试一下 »](http://www.runoob.com/try/try.php?filename=tryjs_addeventlistener_usecapture)

## removeEventListener() 方法

removeEventListener() 方法移除由 addEventListener() 方法添加的事件句柄:

## 实例

element.removeEventListener("mousemove", myFunction);

[尝试一下 »](http://www.runoob.com/try/try.php?filename=tryjs_addeventlistener_remove)

## 实例

跨浏览器解决方法:

var x = document.getElementById("myBtn");  
if (x.addEventListener) {                    // 所有主流浏览器，除了 IE 8 及更早版本  
    x.addEventListener("click", myFunction);  
} else if (x.attachEvent) {                  // IE 8 及更早版本  
    x.attachEvent("onclick", myFunction);  
}

[尝试一下 »](http://www.runoob.com/try/try.php?filename=tryjs_addeventlistener_crossbrowser)

所有 HTML DOM 事件，可以查看我们完整的 [HTML DOM Event 对象参考手册](http://www.runoob.com/jsref/dom-obj-event.html)。

# 13. JavaScript HTML DOM 元素(节点)

创建新的 HTML 元素

如需向 HTML DOM 添加新元素，您必须首先创建该元素（元素节点），然后向一个已存在的元素追加该元素。

## 实例

<div id="div1">  
<p id="p1">这是一个段落。</p>  
<p id="p2">这是另一个段落。</p>  
</div>  
  
<script>  
var para=document.createElement("p");  
var node=document.createTextNode("这是一个新段落。");  
para.appendChild(node);  
  
var element=document.getElementById("div1");  
element.appendChild(para);  
</script>

[尝试一下 »](http://www.runoob.com/try/try.php?filename=try_elementcreate)

## 例子解析：

这段代码创建新的<p> 元素：

var para=document.createElement("p");

如需向 <p> 元素添加文本，您必须首先创建文本节点。这段代码创建了一个文本节点：

var node=document.createTextNode("这是一个新段落。");

然后您必须向 <p> 元素追加这个文本节点：

para.appendChild(node);

最后您必须向一个已有的元素追加这个新元素。

这段代码找到一个已有的元素：

var element=document.getElementById("div1");

以下代码在已存在的元素后添加新元素：

element.appendChild(para);

## 删除已有的 HTML 元素

以下代码演示了如何删除元素：

## 实例

<div id="div1">  
<p id="p1">这是一个段落。</p>  
<p id="p2">这是另一个段落。</p>  
</div>

<script>  
var parent=document.getElementById("div1");  
var child=document.getElementById("p1");  
parent.**removeChild**(child);  
</script>

[尝试一下 »](http://www.runoob.com/try/try.php?filename=try_elementremove)

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title>菜鸟教程(runoob.com)</title>

<style>

#div1{

color:red;

}

#div2{

color:green;

}

</style>

</head>

<body>

<div id="div1">

<p id="p1">这是一个段落。</p>

</div>

<div id="div2">

aslkjlajsfl

</div>

<script>

var para=document.createElement("p");

var node=document.createTextNode("这是一个新段落。");

para.appendChild(node);

var element=document.getElementById("div2");

element.appendChild(para);

</script>

</body>

</html>

## 实例解析

这个 HTML 文档含有拥有两个子节点（两个 <p> 元素）的 <div> 元素：

<div id="div1">  
<p id="p1">这是一个段落。</p>  
<p id="p2">这是另一个段落。</p>  
</div>

找到 id="div1" 的元素：

var parent=document.getElementById("div1");

找到 id="p1" 的 <p> 元素：

var child=document.getElementById("p1");

从父元素中删除子元素：

parent.removeChild(child);

|  |  |
| --- | --- |
| **lamp** | 如果能够在不引用父元素的情况下删除某个元素，就太好了。 不过很遗憾。DOM 需要清楚您需要删除的元素，以及它的父元素。 |

这是常用的解决方案：找到您希望删除的子元素，然后使用其 parentNode 属性来找到父元素：

var child=document.getElementById("p1");  
child.parentNode.removeChild(child);

如果您希望学到更多有关使用 JavaScript 访问 HTML DOM 的知识，请访问我们完整的 [HTML DOM 教程](http://www.runoob.com/htmldom/htmldom-tutorial.html)。

# 14. js里面的单引号和双引号有什么区别？？

你好，在js当中没什么区别，都是可以括住字符串的，所以你会发现一般写这两个队都对。  
不过在字符串拼接与函数传参混合的时候，由于同种引号不能嵌套，所以需要者两种引号配合使用，才能实现。举例如下：  
td=$("<div class='result'><a href='../student/modifystudent.aspx?sno="+sno+"' target='\_blank'>修改</a></div>");

在效率上是有区别的，在js编译处理时双引号会被转成单引号的。推荐在没有嵌套的地方使用单引号

# 15. JavaScript prototype 属性

## 定义和用法

prototype 属性允许您向对象添加属性和方法

**注意：** Prototype 是全局属性，适用于所有的Javascript对象。

## 语法

object.prototype.name=value

用 prototype 属性给对象添加属性:

<script>  
  
function employee(name,jobtitle,born)  
{  
this.name=name;  
this.jobtitle=jobtitle;  
this.born=born;  
}  
  
var fred=new employee("Fred Flintstone","Caveman",1970);  
employee.prototype.salary=null;  
fred.salary=20000;  
  
document.write(fred.salary);  
  
</script>