目录

[事件 3](#_Toc2591248)

[注意 3](#_Toc2591249)

[事件捕获、目标、冒泡 3](#_Toc2591250)

[绑定 4](#_Toc2591251)

[在html中 4](#_Toc2591252)

[在js中.onclick= 4](#_Toc2591253)

[js绑定addEventListener 4](#_Toc2591254)

[注意 5](#_Toc2591255)

[常用事件 5](#_Toc2591256)

[onload 和 onunload 5](#_Toc2591257)

[onchange 5](#_Toc2591258)

[onmouseover 和 onmouseout 5](#_Toc2591259)

[onmouseenter和onmouseleave 5](#_Toc2591260)

[onclick 5](#_Toc2591261)

[tap 5](#_Toc2591262)

[onwheel onmousewheel onDOMMouseScroll 6](#_Toc2591263)

[详细 6](#_Toc2591264)

[鼠标事件 6](#_Toc2591265)

[键盘事件 6](#_Toc2591266)

[特殊事件 7](#_Toc2591267)

[表单事件 8](#_Toc2591268)

[粘贴板事件 8](#_Toc2591269)

[打印事件 8](#_Toc2591270)

[拖动事件 9](#_Toc2591271)

[多媒体事件 10](#_Toc2591272)

[动画事件 11](#_Toc2591273)

[过渡事件 12](#_Toc2591274)

[其他事件 13](#_Toc2591275)

[事件属性 13](#_Toc2591276)

[属性 14](#_Toc2591277)

[基本信息 14](#_Toc2591278)

[type 14](#_Toc2591279)

[srcElement/target 14](#_Toc2591280)

[currentTarget/this 14](#_Toc2591281)

[位置相关 14](#_Toc2591282)

[clientX/clientY 14](#_Toc2591283)

[offsetX,offsetY/layerX,layerY 14](#_Toc2591284)

[x,y/pageX,pageY 14](#_Toc2591285)

[screenX、screenY 14](#_Toc2591286)

[鼠标键盘按键 14](#_Toc2591287)

[altKey,ctrlKey,shiftKey 14](#_Toc2591288)

[button 15](#_Toc2591289)

[keyCode 15](#_Toc2591290)

[fromElement,toElement 15](#_Toc2591291)

[操作 15](#_Toc2591292)

[cancelBubble 15](#_Toc2591293)

[returnValue 15](#_Toc2591294)

[事件方法 15](#_Toc2591295)

[createEvent 15](#_Toc2591296)

[initEvent 15](#_Toc2591297)

[preventDefault 16](#_Toc2591298)

[stopImmediatePropagation 16](#_Toc2591299)

[stopPropagation 16](#_Toc2591300)

[分割线 16](#_Toc2591301)

[总结 16](#_Toc2591302)

[document.body.appendChild(btn); 16](#_Toc2591303)

[常用对象 16](#_Toc2591304)

[概念 17](#_Toc2591305)

[节点 17](#_Toc2591306)

[其他 17](#_Toc2591307)

[textContent属性 17](#_Toc2591308)

[常见错误 18](#_Toc2591309)

[onscroll 18](#_Toc2591310)

[document方法 19](#_Toc2591311)

[新建节点 19](#_Toc2591312)

[createElement() 19](#_Toc2591313)

[.createTextNode() 19](#_Toc2591314)

[查找节点 19](#_Toc2591315)

[.getElementById() 19](#_Toc2591316)

[.getElementByTagName() 19](#_Toc2591317)

[.getElementByClassName() 19](#_Toc2591318)

[修改节点内容 20](#_Toc2591319)

[改变 HTML 内容innerHTML 20](#_Toc2591320)

[改变 CSS 样式 .style. 20](#_Toc2591321)

[操作 HTML 属性 20](#_Toc2591322)

[创建新的 HTML 元素 21](#_Toc2591323)

[删除已有的 HTML 元素 21](#_Toc2591324)

[改变事件（处理程序） 21](#_Toc2591325)

[.write() 21](#_Toc2591326)

[Element方法 21](#_Toc2591327)

[查找节点 21](#_Toc2591328)

[修改节点 22](#_Toc2591329)

[replaceChild() 22](#_Toc2591330)

[插入节点 22](#_Toc2591331)

[插入子节点appendChild(node) 22](#_Toc2591332)

[在之前插入 A.insertBefore(a,b) 22](#_Toc2591333)

[删除节点 23](#_Toc2591334)

[删除子节点 removeChild(node) 23](#_Toc2591335)

[element属性 23](#_Toc2591336)

[innerHTML 23](#_Toc2591337)

[无限循环 23](#_Toc2591338)

[nodeValue 24](#_Toc2591339)

[nodeType 24](#_Toc2591340)

[parentNode、firstChild 、lastChild 24](#_Toc2591341)

[childNodes[] 24](#_Toc2591342)

[事件相关 25](#_Toc2591343)

[scrollTop 25](#_Toc2591344)

[属性 25](#_Toc2591345)

[.attribute[]（4个） 25](#_Toc2591346)

[.isId 25](#_Toc2591347)

[.name 25](#_Toc2591348)

[.value 25](#_Toc2591349)

[.specified 26](#_Toc2591350)

[.attributes 26](#_Toc2591351)

[.length 26](#_Toc2591352)

[.setNamedItem() 26](#_Toc2591353)

[.item() 26](#_Toc2591354)

[removeNamedItem() 26](#_Toc2591355)

[getNamedItem() 26](#_Toc2591356)

[创建属性 26](#_Toc2591357)

[createAttribute 26](#_Toc2591358)

# 事件

## 注意

### 事件捕获、目标、冒泡

捕获由上往下，冒泡由下往上，先有捕获，后有冒泡，也就是由上往下，到目标阶段，然后向上执行一圈。

由于老的浏览器不支持捕获，所以很少有人使用事件捕获。

addEventListener(event, listener, useCapture)

　　　　·参数定义：event---（事件名称，如click，不带on），listener---事件监听函数，useCapture---是否采用事件捕获进行事件捕捉，

　　　　　　　　默认为false，即采用事件冒泡方式

　　　　addEventListener在 IE11、Chrome 、Firefox、Safari等浏览器都得到支持。

## 绑定

### 在html中

<button onclick="displayDate**()**">点我</button>

### 在js中.onclick=

document.getElementById("myBtn").onclick=function(){displayDate()};

### js绑定addEventListener

x.addEventListener("webkitAnimationStart", myStartFunction);

function myStartFunction() {

**this**.innerHTML = "animationstart 事件触发 - 动画已经开始";

this.style.backgroundColor = "pink";

}

和第二种一样

attachEvent(),detachEvent()/addEventListener(),removeEventListener

为制定 DOM对象事件类型注册多个事件处理函数的方法，它们有两个参数，第一个是事件类型，第二个是事件处理函数。在  
attachEvent()事件执行的时候，this关键字指向的是window对象，而不是发生事件的那个元素；

#### 注意

绑定事件方法的第三个参数，就是控制事件触发顺序是否为事件捕获。true,事件捕获；false,事件冒泡。默认false,即事件冒泡。

## 注意

## 常用事件

### onload 和 onunload

当用户进入或离开页面时，会触发 onload 和 onunload 事件。

onload 事件可用于检查访客的浏览器类型和版本，以便基于这些信息来加载不同版本的网页。

onload 和 onunload 事件可用于处理 cookies

### onchange

常用于输入字段的验证，当用户改变输入字段的内容时。

### onmouseover 和 onmouseout

onmouseover 和 onmouseout 事件可用于在鼠标指针移动到或离开元素时触发函数

鼠标经过时自身触发事件，经过其子元素时也触发该事件(因为子元素会冒泡到父元素)

### onmouseenter和onmouseleave

鼠标经过时自身触发事件，经过其子元素时不触发该事件。（因为不会冒泡）

### onclick

### tap

tap不是原生事件，是各个移动端框架封装的用来代替click的事件。

在web手机端，clikc会有200-300ms的延时，所以要用tap代替click作为点击事件，singleTap和doubleTap分别作为单次点击和双击，但是使用tap会带来点透事件(事件穿透)。  
tap事件是通过监听绑定document上的touch事件来模拟的，并且tap事件是冒泡到document上才触发的。在移动端的事件触发从早到晚排序：touchstart touchstop click 。 所以click的触发是有延时的，about 300ms由于我们在touchstart阶段就已经隐藏了A，当click被触发时，能够被点击的就是B；

一些类似fastclick的库，优化了click的体验，也可以达到类似的效果。

### onwheel onmousewheel onDOMMouseScroll

**mousewheel DOMMouseScroll**

mousewheel已废弃。

在我的win7系统下，无论IE7, IE10, Opera12,或者是safari5.1，每次往下滚动event.**wheelDelta**值都是-120.

对于FireFox浏览器（Opera浏览器也有），判断鼠标滚动方向的属性为event.**detail**, 向下滚动值为3

FireFox浏览器向下滚动是正值，而其他浏览器是负值。

**onwheel**

在 IE 浏览器中，只能通过 addEventListener() 方法支持 wheel 事件。 在 DOM 对象中没有 onwheel 属性。

一旦事件被onwheel拦截，那么onmousewheel将无法捕获到事件，而DOMMouseScroll完全可以。

## 详细

## 鼠标事件

一类：点击 onclick ondb**l**click oncontextmenu onmousedown onmouseup

二类：移动 onmousemove onmouseover onmouseout(ie中：onmouseenter onmouseleave 而且这两个不冒泡)

## 键盘事件

onkeydowm,onkeyup,onkeypress(按下并松开)

## 特殊事件

1

onerror

onabort

2

onscoll

onresize

3

onhas**h**change

onbeforeunload

4

onunload,onload和onpagehide,onpageshow



## 表单事件



## 粘贴板事件



## 打印事件



## 拖动事件

拖动物体：ondragstart ondrag ondragend

拖动目标：ondragenter ondragover ondrop ondragleave

ondrop在ondragover之后出发

注意

1：ondragenter和ondragover函数一定要包含event.preventdefault()以阻止默认操作。

2：在ondragenter和ondragover函数中不能使用return false。（正常情况下，return false可以实现preventDefault 和stopPropogation的效果。）

3：浏览器本身有拖动的默认效果，因此不要在ondragstart中用return false等，这样会阻止了拖动的默认效果。

4：当你的函数本向在要返回布尔时，以下是一个比较全面的写法：

function (eventObject) {      
    if (eventObject.preventDefault) {  
        eventObject.preventDefault();  
    } else if (window.event)  {  
        window.event.returnValue = false;  
    }  
    return true;  
};

## 多媒体事件



## 动画事件

animationstart,animationiteration(重复播放时),animationend

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title>菜鸟教程(runoob.com)</title>

<style>

#myDIV {

margin: 25px;

width: 550px;

height: 100px;

background: orange;

position: relative;

font-size: 20px;

}

/\* Chrome, Safari, Opera \*/

@-webkit-keyframes mymove {

from {top: 0px;}

to {top: 200px;}

}

@keyframes mymove {

from {top: 0px;}

to {top: 200px;}

}

</style>

</head>

<body>

<p>该实例使用了 addEventListener() 方法为 DIV 元素添加"animationstart", "animationiteration" 和 "animationend" 事件。</p>

<div id="myDIV" onclick="myFunction()">点我开始动画</div>

<script>

var x = document.getElementById("myDIV")

// 使用 JavaScript 开始动画

function myFunction() {

x.style.WebkitAnimation = "mymove 4s 2"; // Chrome, Safari 和 Opera 代码

x.style.animation = "mymove 4s 2";

}

// Chrome, Safari 和 Opera

x.addEventListener("webkitAnimationStart", myStartFunction);

x.addEventListener("webkitAnimationIteration", myIterationFunction);

x.addEventListener("webkitAnimationEnd", myEndFunction);

x.addEventListener("animationstart", myStartFunction);

x.addEventListener("animationiteration", myIterationFunction);

x.addEventListener("animationend", myEndFunction);

function myStartFunction() {

this.innerHTML = "animationstart 事件触发 - 动画已经开始";

this.style.backgroundColor = "pink";

}

function myIterationFunction() {

this.innerHTML = "animationiteration 事件触发 - 动画重新播放";

this.style.backgroundColor = "lightblue";

}

function myEndFunction() {

this.innerHTML = "animationend 事件触发 - 动画已经完成";

this.style.backgroundColor = "lightgray";

}

</script>

</body>

</html>

## 过渡事件

transitionend

## 其他事件



# 事件属性

1event代表事件的状态，例如触发event对象的元素、鼠标的位置及状态、按下的键等等；  
  
2.  event对象只在事件发生的过程中才有效。  
firefox里的event跟IE里的不同，IE里的是全局变量，随时可用；firefox里的要用参数引导才能用，是运行时的临时变量。  
在IE/Opera中是window.event，在Firefox中是event；而事件的对象，在IE中是 window.event.srcElement，在Firefox中是event.target，Opera中两者都可用。  
  
3.  下面两句效果相同  
var evt = (evt) ? evt : ((window.event) ? window.event : null);  
var evt = evt || window.event; // firefox下window.event为null, IE下event为null  
  
4.  IE中事件的起泡  
IE中事件可以沿着包容层次一点点起泡到上层，也就是说，下层的DOM节点定义的事件处理函数，到了上层的节点如果还有和下层相同事件类型的事件处理函数，那么上层的事件处理函数也会执行。例如， div 标签包含了 a ，如果这两个标签都有onclick事件的处理函数，那么执行的情况就是先执行标签 a 的onclick事件处理函数，再执行 div 的事件处理函数。如果希望的事件处理函数执行完毕之后，不希望执行上层的 div 的onclick的事件处理函数了，那么就把cancelBubble设置为true即可。

## 属性

## 基本信息

### type

事件的类型，如onlick中的click；

srcElement/target

事件源，就是发生事件的元素；

### currentTarget/this

event.target表示发生点击事件的元素；

this表示的是注册点击事件的元素

this 等于 e.currentTarget

## 位置相关

clientX/clientY

事件发生的时候，鼠标相对于浏览器窗口可视文档区域的左上角的位置；(在DOM标准中，这两个属性值都不考虑文档的滚动情况，也就是说，无论文档滚动到哪里，只要事件发生在窗口左上角，clientX和clientY都是 0，所以在IE中，要想得到事件发生的坐标相对于文档开头的位置，要加上  
document.body.scrollLeft和 document.body.scrollTop)

offsetX,offsetY/layerX,layerY

事件发生的时候，鼠标相对于源元素左上角的位置；

x,y/pageX,pageY

检索相对于父要素鼠标水平坐标的整数；

screenX、screenY

鼠标指针相对于显示器左上角的位置，如果你想打开新的窗口，这两个属性很重要

鼠标键盘按键

altKey,ctrlKey,shiftKey

返回一个布尔值；

button

声明被按下的鼠标键，整数，1代表左键，2代表右键，4代表中键，如果按下多个键，酒把这些值加起来，所以3就代表左右键同时按下；（firefox中 0代表左键，1代表中间键，2代表右键）

keyCode

返回keydown何keyup事件发生的时候按键的代码，以及keypress 事件的Unicode字符；(firefox2不支持 event.keycode，可以用 event.which替代 )

fromElement,toElement

前者是指代mouseover事件中鼠标移动过的文档元素，后者指代mouseout事件中鼠标移动到的文档元素；

操作

cancelBubble

一个布尔属性，把它设置为true的时候，将停止事件进一步起泡到包容层次的元素；(e.cancelBubble = true; 相当于 e.stopPropagation();)

stoppropagation属于W3C标准，试用于Firefox等浏览器，但是不支持IE浏览器。相反cancelBubble不符合W3C标准，而且只支持IE浏览器。

returnValue

一个布尔属性，设置为false的时候可以组织浏览器执行默认的事件动作；(e.returnValue = false; 相当于 e.preventDefault();)

# 事件方法

## createEvent

var newEvent = document.createEvent("UIEvents");

## initEvent

已经被移除。

## preventDefault

## stopImmediatePropagation

阻止其他监听事件。当有多个监听事件监听同一个dom节点时使用。

同时也可以阻止冒泡和捕获。

## stopPropagation

阻止冒泡和捕获

# 分割线

# 总结

## document.body.appendChild(btn);

无效：document.body.div.appendChild(btn);

## 常用对象

**document,**

**Anchor（链接 <a></a>）**

**area(<area>不用闭和)**

**ba**

**se(<base/>)**

**button**

**form**

**image(img**

**event**

**select,option**

<select id="mySelect">

<option>苹果</option>

<option>梨子</option>

<option>香蕉</option>

<option>橙子</option>

</select>

**Table, TableHeader, TableRow, TableData**

<table id="myTable" border="1">

<tr>

<td>100</td>

<td>200</td>

</tr>

<tr>

<td>300</td>

<td>400</td>

</tr>

</table>

# 概念

## 节点

元素、属性、文本、注释都是节点

# 其他

## textContent属性

x.innerHTML=a.attributes.getNamedItem("onclick").textContent;

document.getElementById("myList).textContent;

# 常见错误

### onscroll

componentDidMount() {

// 触发

window.addEventListener('scroll', this.orderScroll.bind(this));

// document.body.addEventListener('scroll', this.orderScroll.bind(this)); 不触发

// document.getElementById('userOrder').addEventListener('scroll', this.orderScroll.bind(this)); 不触发

// this.refs.userOrder.addEventListener('scroll', this.orderScroll.bind(this)); 不触发

}

orderScroll() {

console.log('scroll');

}

render() {

return(

<div id="userOrder" ref="userOrder">

<div></div>

</div>

)

}

// 这样写同样不会触发

// render() {

// return(

// <div onScroll={this.orderScroll.bind(this)}>

// <div></div>

// </div>

// )

// }

只有有滚动条的元素才能触发scroll事件。所以你这里面除了window之外当然都不触发。

Document也只是一个节点，所以它的功能普通节点也有，关键只是有没有效果

document.getElementById("main").getElementsByTagName("p");

# document方法

## 新建节点

### createElement()

var btn=document.createElement("BUTTON");

大小写都可以

### .createTextNode()

文本节点

## 查找节点

### .getElementById()

记得加引号

### .getElementByTagName()

getElementsByTagName() 返回带有指定标签名的所有元素。

x=document.getElementsByTagName("p");

document.write("第二个段落的 innerHTML 内容为: " + **x[1].innerHTML**);

### .getElementByClassName()

如果您希望查找带有相同类名的**所有** HTML 元素

## 修改节点内容

### 改变 HTML 内容innerHTML

### 改变 CSS 样式 .style.

document.getElementById("p2")**.style.color**="blue"; document.getElementById("p2").style.fontFamily="Arial"; document.getElementById("p2").style.fontSize="larger"

### 操作 HTML 属性

#### .xxx直接获取属性节点的值(部分属性可以)

#### .attributes()

返回一个数组

#### .getAttribute()

返回属性值

#### setAttribute()

参数：属性节点

方法用于添加新的属性节点。

如果元素中已经存在指定名称的属性，那么该属性将被新属性替代。如果新属性替代了已有的属性，则返回被替代的属性，否则返回 NULL。

#### [getAttributeNode()](http://www.runoob.com/jsref/met-element-getattributenode.html)

#### [hasAttribute()](http://www.runoob.com/jsref/met-element-hasattribute.html)

#### [hasAttributes()](http://www.runoob.com/jsref/met-element-hasattribute.html)

### 创建新的 HTML 元素

#### .appendChild（）

### 删除已有的 HTML 元素

### 改变事件（处理程序）

<input type="button"

**onclick="document.body.style.backgroundColor='lavender'**;"

value="修改背景颜色">

或者

function ChangeBackground() { document.body.style.backgroundColor="lavender"; } </script> <input type="button" onclick="ChangeBackground()" value="修改背景颜色" />

### .write()

var txt=document.getElementById("intro").innerHTML; document.write(txt);

# Element方法

## 查找节点

通过属性 parentNode、firstChild 以及 lastChild

## 修改节点

### replaceChild()

## 插入节点

要显示内容通常还要添加文本节点

### 插入子节点appendChild(node)

### 在之前插入 A.insertBefore(a,b)

A:父节点，在b之前插入

<div id="div1">

<p id="p1">这是一个段落。</p>

<p id="p2">这是另外一个段落。</p>

</div>

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

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

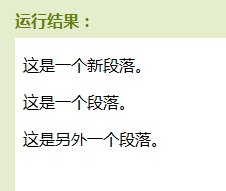
**para.appendChild(node);**

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

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

element.insertBefore(para,child);

这里，para是两个节点结合，插入内容要加入文本节点



## 删除节点

必须要【知道（错），】拿到父元素

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

### 删除子节点 removeChild(node)

有时候是没有效果的

var btn=document.createElement("button");

btn.innerHTML="aaaa";

document.body.appendChild(btn);

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

a.innerHTML="DD";

var b=document.getElementById("demo");

b.removeChild(a);

# element属性

## innerHTML

|  |  |
| --- | --- |
|  | innerHTML 属性可用于获取或改变任意 HTML 元素，包括 <html> 和 <body>。 |

var btn=document.createElement("BUTTON");

btn.innerHTML="aaaa";

document.body.appendChild(btn);

### 无限循环

var txt=document.body.innerHTML;

document.write(txt);

#### this.innerHTML

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

## nodeValue

规定节点的值。

* 元素节点的 nodeValue 是 undefined 或 null
* 文本节点的 nodeValue 是文本本身
* 属性节点的 nodeValue 是属性值

## nodeType



## parentNode、firstChild 、lastChild

## childNodes[]

txt=document.getElementById("intro").childNodes[0].nodeValue;

document.write(txt);

## 事件相关

### scrollTop

|  |  |
| --- | --- |
| *element*.scrollHeight | 返回整个元素的高度（包括带滚动条的隐蔽的地方） |
| *element*.scrollLeft | 返回当前视图中的实际元素的左边缘和左边缘之间的距离 |
| *element*.scrollTop | 返回当前视图中的实际元素的顶部边缘和顶部边缘之间的距离 |

let docH = document.body.scrollHeight, //滚动条自身高度

scrollTop = document.body.scrollTop; //滚动条滚动高度

# 属性

## .attribute[]（4个）

### .isId

是否为id属性

var x=document.getElementById("demo");

x.innerHTML=**x.attributes[0].isId**;

注意：火狐浏览器返回undefined

注意:Internet Explorer 或者 Opera不支持isId属性。

### .name

返回该属性节点的名字（attribute[]只有序号）

### .value

返回该属性节点的值

### .specified

判断该序号属性节点是否被指定值，true，false

## .attributes

### .length

### .setNamedItem()

设置指定属性节点(通过节点)

### .item()

返回节点属性中处于索引号的节点

### removeNamedItem()

删除指定属性节点

### getNamedItem()

从节点列表中返回的指定属性节点。

x.innerHTML=a.attributes.getNamedItem("onclick").textContent;

## 创建属性

### createAttribute

var h=document.getElementsByTagName("H1")[0];

var typ=document.createAttribute("class");

typ.nodeValue="democlass";

h.attributes.setNamedItem(typ);