#第六章 DOM

##6.1 DOM的概念

#####6.1.1DOM的概念

#####6.1.2DOM和JavaScript的关系

#####6.1.3 DOM节点

> 整个文档是一个文档节点

> 每个html元素是元素节点

> html元素内的文本是文本节点

> 每个html属性都是属性节点

> 注释也是节点，叫注释节点

#####6.1.4 DOM树

DOM树体现着html页面的层级结构，而DOM树有DOM文档树和DOM元素树两种。DOM元素树包含的只有元素节点，而DOM文档树则包括DOM文档中的所有内容。

##6.2 获取元素

#####6.2.1 用ID获取元素

//用变量 在文档中 找 元素 用ID id

var d = document.getElementById('d');

#####6.2.2 用标签名获取元素

var lis = document.getElementByTAgName('li')

//伪数组

console.log(lis);

var ul = document.getElementById('ul');

var list = ul.getElementsByTagName('list');

console.log(list)

var div = document.getElementsByTagName('div');

console.log(div[0]);

#####6.2.3用类名获取元素

var a = document.getElementsByClassName('a');

console.log(a);

#####6.2.4 用name属性值获取元素

<input type="radio" name="sex" value="0">男

<input type="radio" name="sex" value="1">女

var sex = document.getElementsByName('sex');

console.log(sex);

#####6.2.4 用选择器获取元素

var ul = document.getElementById('ul')

var x = document.querySelector('#div>div a');

var lis = ul.querySelectorAll('li');

console.log(lis);

##6.3 获取和设置元素中的其他信息

#####6.3.1 获取元素名

var div = document.getElementById('mydiv');

console.log(div.tagName);

#####6.3.2获取元素节点里的内容

var div = document.getElementById('mydiv');

console.log(div.innerHTML);

div.innerHTML = '你好';

#####6.3.3 获取元素节点中的文本

var div = document.getElementById('mydiv');

console.log(div.innerText);

div.innerText = '你好';

#####6.3.4获取元素的其他信息

var div = document.getElementById('mydiv');

console.log(div.className)

#####6.3.5 获取属性样式

<div style="background-color: blue"></div>

var div = document.getElementsByTagName('div');

//js拿到和获取的是内联

console.log(div[0]);

#####6.3.6 获取元素的属性

var div = document.getElementById('mydiv');

var a = document.getElementsByTagName('a');

console.log(div.getAttribute('data-ww'));

console.log(a.getAttribute('href'));

#####6.3.7 设置元素属性

var div = document.getElementsByTagName('div');

var a = document.getElementsByTagName('a');

div[0].setAttribute('id','mydiv');

a[0].setAttribute('href','http://baidu.com');

#####6.3.8 删除元素属性

var div = document.getElementById('mydiv');

div.removeAttribute('div');

##6.4 增加元素

#####6.4.1创建元素节点

//用一个变量承接在文档中‘创建’的一个元素

var div = document.createElement('div');

//将已经创建的元素放到想放的位置

document.body.appendChild(div);

#####6.4.2创建文本节点

var p = document.getElementById('p');

var text = document.createTextNode('一段文字');

p.appendChild(text);

#####6.4.3 css样式赋予

div.style.cssText =

'width:100px;height:100px;background-color:blue;';

#####6.4.4 在某元素前插入创建的新元素

<ul id="ul">

<li id="pg">苹果</li>

<li id="jz">桔子</li>

<li id="xj">香蕉</li>

</ul>

<script>

var ul = document.getElementById('ul');

//获取苹果

var pg = document.createElement('pg');

//获取桔子

var li = document.createElement('li');

//在父级中插入一个元素（插入元素，再谁前面）

ul.insertBefore(li,jz);

var t = document.createTextNode('火龙果');

li.appendChild(t);

</script>

## 6.5 删除和替换元素

##### 6.5.1 元素的替换

var mydiv = document.getElementById('mydiv');

var pyo = document.createElement('p');

//在父元素中替换子元素，第一个是新的，第二个是旧的

document.body.replaceChild(pyo,mydiv);

######6.5.2 元素的删除

var mydiv = document.getElementById('mydiv');

document.body.removeChild(mydiv);

## 6.6 查找节点（node）

节点是在DOM树上的每一个节点，其中包括：元素，文本，属性，注释等等，常用的有三个，元素，文本，属性，

#####6.6.1 节点的属性

var mydiv = document.getElementById('mydiv');

document.body.nodeName();

document.body.nodeValue();

document.body.nodeType();

#####6.6.2 层次节点

节点可以分为父节点和子节点，兄弟节点两种，当我们知道其中之一的时候，可以用一些方法找到另一个

#####6.6.3获取所有子节点

<ul id="myul">

<li>111</li>

<li>222</li>

<li>333</li>

</ul>

<script>

var myul = document.getElementById('myul');

//childNodes是所有节点

console.log(myul.childNodes);

</script>

#####6.6.4 获取第一个和最后一个子节点

<ul id="myul"><li>111</li><li>222</li><li>333</li></ul>

<script>

var myul = document.getElementById('myul');

console.log(myul.firstChild);

console.log(myul.lastChild);

</script>

#####6.6.5 获取父节点

<body>

<ul id="myul"><li>111</li><li>222</li><li>333</li></ul>

<script>

var myul = document.getElementById('myul');

console.log(myul.parentNode);

</script>

</body>

#####6.6.6 获取兄弟节点

<h2>h2</h2><div id="mydiv"></div><p>p</p>

<script>

var mydiv = document.getElementById('mydiv')

console.log(mydiv.previousSibling)

console.log(mydiv.nextSibling)

</script>

##6.6.7 元素的宽高属性

#####6.7.1 offsetWidth和offsetHeight

var mydiv = document.getElementById('mydiv');

console.log(mydiv.style);//这种方式只能拿到内联样式

console.log(mydiv.offsetWidth);

#####6.7.2clientWidth和clientHeight

console.log(mydiv.clientWidth);

##6.8 子元素与父元素的距离

<div id="a">

<div id="b"></div>

</div>

<script>

var b = document.getElementById('b');

console.log(b.offsetLeft);

console.log(b.offsetTop);

</script>

#第七章 事件

###7.1 事件种类

#####7.1.1 鼠标事件

单一作用于某个元素的事件

var mydiv = document.getElementById('mydiv');

//点击事件

//mydiv.onclick = function(){

// alert('点击事件');

//}

//双击事件

//mydiv.ondblclick = function(){

// alert('双击事件');

//}

//鼠标按下(按下不松手为按下)

mydiv.onmousedown = function(){

mydiv.innerHTML = '按下事件';

}

//鼠标抬起

//mydiv.onmouseup = function(){

// mydiv.innerHTML = '抬起事件';

//}

//鼠标移动

//需要注意的是只要鼠标移动就触发

//document.onmousemove = function(){

// console.log('移动了');

//}

//mydiv.onmousemove = function(){

// console.log('div');

//}

鼠标移入移出情况

（1）第一套移入移出会多次触发

//鼠标移入

//会触发冒泡（从内逐级向外依次触发）

//mydiv.onmouseover = function(){

// console.log('红');

//}

//box.onmouseover = function(){

// console.log('蓝');

//}

mydiv.onmouseov = function(){

console.log('红');

}

//不会触发多次

box.onmouseover = function(){

console.log('蓝');

}

//鼠标移出

//mydiv.onmouseout = function(){

// console.log('离开红');

//}

//会触发多次，取决于嵌套和鼠标位置

//box.onmouseout = function(){

// console.log('离开蓝');

//}

//mydiv.onmouseleave = function(){

// console.log('离开红');

//}

//会触发多次，取决于嵌套和鼠标位置

box.onmouseleave = function(){

console.log('离开蓝');

}

#####7.1.2 键盘事件

document.onkeydown = function(e){

alert('按下');

}

document.onkeydown = function(e){

alert('抬起了');

}

###7.2 Event事件对象

#####7.2.1 Event事件对象的概念

#####7.2.2 Event事件对象产生的时间

#####7.2.3 Event事件对象接收方式

//事件触发时 调用的函数（event）

document.onclick = function(e){

console.log(e);

}

#####7.2.4 event对象常用属性

e.clientX 和 e.clientY

e.keyCode

#####7.2.5 event的兼容性

document.onclick = function(e){

// 非ie 低版本ie

var ev = e || window.event;

console.log(ev.clientX);

}

###7.3 对象事件（加载完成）

#####7.3.1 页面（元素）加载完成后执行事件

<script>

//页面加载完成之后执行函数内的代码段

window.onload = function(){

var mydiv = document.getElementById('mydiv');

mydiv.onclick = function(){

alert(mydiv.tagName);

}

}

</script>

#####7.3.2 窗口变化大小时触发事件

window.onresize = function(){

console.log(1);

}

###7.4 表单事件

#####7.4.1 获取焦点

#####7.4.2 失去焦点

#####7.4.3 用户输入触发

#####7.4.4 内容改变时触发

in1.onchange = function(){

in1.style.backgroundColor = 'blue';

}

#7.5 事件绑定、事件监听、事件委托

#####7.5.1 事件绑定概念

#####7.5.2 事件绑定的方法

#####7.5.3 解除绑定的方法

<button id="btn2">按钮2</button>

<script>

var btn2 = document.getElementById('btn2');

btn2.onclick = function(){

alert('按钮2');

}

//解除绑定

btn2onclick = null;

</script>

###7.6 事件监听

#####7.6.2 解绑和封装

### 7.7 事件冒泡和事件委托