**学习目标：**

1、firstChild和firstElementChild

2、lastChild 和lastElementChild

3、nextSibling 和nextElementSibling

4、previousSibling 和previousElementSibling

**1、firstChild和firstElementChild**

演示：

<ul id="ul1">

<li>11111</li>

<li>22222</li>

<li>33333</li>

<li>44444</li>

</ul>

oUl.firstChild

元素.firstChild : 只读 属性 第一个子节点

标准下：firstChild会包含文本类型的节点

非标准下：只包含元素节点

oUl.firstElementChild

元素.firstElementChild : 只读 属性

标准下：获取第一个元素类型的子节点

非标准下：没有

兼容处理：

if ( oUl.firstElementChild ) {

// 标准浏览器

oUl.firstElementChild.style.background = 'red';

} else {

// 非标准浏览器

oUl.firstChild.style.background = 'red';

}

或者：

var oFirst = oUl.firstElementChild || oUl.firstChild;

oFirst.style.background = 'red';

封闭函数：处理非正常的情况，如没有子节点。

function firstChild(obj) {

var oFirst = obj.firstElementChild || obj.firstChild;

// console.log(oFirst);

if(oFirst == null || oFirst.nodeType != 1) {

return undefined;

};

return oFirst;

}

真正强大的处理：

oUl.children[0]

**二、元素.lastChild || 元素.lastElementChild 最后一个子节点**

var oLast = oUl.lastElementChild || oUl.lastChild;

oLast.style.background = 'yellow';

**三、元素.nextSibling || 元素.nextElementSibling 当前元素的下一个兄弟节点**

var oNext = oFirst.nextElementSibling || oFirst.nextSibling;

oNext.style.background = 'blue';

**四、元素.previousSibling || 元素.previousElementSibling 当前元素的上一个兄弟节点**

var oPrev = oLast.previousElementSibling || oLast.previousSibling;

oPrev.style.background = 'orange';

**总结**：都有兼容问题。