**DOM**

概念：

DOM文档对象模型，DOM对象不仅仅是一个普通的内置对象，它还是一个巨大的API的核心对象，他将文档的内容呈现在JS面前，并赋予了JS操作文档的能力。

**1.1 Document**

**1.1.1 getELementById**

方法 获取元素的id,如果找到相应元素，则返回该元素 ，否则但会返回null.

// 用变量接 在文档中 找 元素 用id "d"

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

**1.1.2 getElementsByTagName()**

（1）可以获取该元素名称下所有元素，返回一个伪数组，或者说是一个节点列表。

var lis = document.getElementsByTagName('li');

//伪数组

console.log(lis);

（2）在某一个元素中，并不是在全局文档中利用标签获取元素，这样更加精准。

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

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

console.log(list);

（3）当元素只有一个的时候,返回的也依然是一个列表,而不是一个元素.可以通过数组的下标找到该元素.

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

console.log(div[0]);

**1.1.3 getElementsByClassName('a');**

用类名获取元素

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

console.log(a);

**1.1.4 选择器**

1.querySelector() 依靠选择器找到元素 只能找到元素列表的第一个元素

2.querySelectorAll() 依靠选择器找到元素 可以找到元素列表的所有元素

**1.1.5 getElementsByName()**

方法可以获取相同名称的name元素,返回一个伪数组对象.

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

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

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

console.log(sex);

**1.1.6 getAttribute('')**

可以获取元素的某个属性，要将属性名称放在括号中记得要用引号括起来。返回值就是字符串形式的属性值。

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

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

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

console.log(a[0].getAttribute('href'));

**1.1.7 setAttribute('')**

可以设置元素的某个属性，第一个参数是属性名，第二个参数是属性值，都是需要用引号括起来以逗号分隔。

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

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

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

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

**1.1.8 removeAttribute()**

可删除某个元素的某个属性，括号放入需要删除的属性名，用引号包裹

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

div.removeAttribute('id');

**1.1.9 createElement()**

创建一个元素节点

//用一个变量承接在文档中‘创建’的一个元素

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

//将已经创建的元素放在相放的位置

document.body.appendChild(div);

**1.1.9 createTextNode('一段文字')**

创建一个文本节点

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

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

p.appendChild(text);

**1.1.10 insertBefore()**

appendChild() 可在某个元素前插入创建的元素

区别：

insertBefore()是可以选择插入的位置，她需要插入到某个子元素之前的位置，因此需要那个子元素，

insertBefore()有两个参数，第一个参数是需要插入的元素，第二个参数是在谁之前插入，两个参数用逗号分隔。

<ul id='ul'>

<li id="pg" >苹果</liid>

<li id="jz">桔子</li>

<li>香蕉</li>

</ul>

<script>

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

//获取苹果

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

//获取桔子

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

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

//在父级

ul.insertBefore(li,jz);

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

li.appendChild(t);

</script>

**1.1.11 replaceChild()**

元素的替换是在父元素中的一个子元素需要被另一个新的子元素所代替，使用replaceChild()方法，该方法有两个参数

，第一个参数是将要替换的新元素，第二个是被替换的旧元素，中间用逗号分隔。

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

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

//在父元素中替换子元素，第一个是新的元素，第二个是旧的元素

document.body.replaceChild(myp,mydiv);

**1.1.12 removeChild()**

元素的删除实在父元素的其中一个子元素需要删除，使用removeChild（）方法，将需要删除的元素放入括号中。

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

document.body.removeChild(mydiv);

**1.2 Node 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

**1.2.1 节点属性**

1.nodeName

2. nodeValue

3.nodeType

**1.2.2 childNodes**

获取所有子节点 （包括所有节点类型，不单单只是元素。）

<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>

**1.2.3 fistChild lastChild**

获取第一个和最后一个子节点

用fistChild和lastChild可以拿到该父元素下的第一个和最后一个子节点，注意不一定是元素节点。

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

</ul>

<script>

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

console.log(myul.firstNodes);

console.log(myul.lastChild);

</script>

1.2.4paretNode

获取父节点

使用paretNode属性可以拿到该元素的父节点，注意父节点只有一个。

<body>

<ul id="myul"></ul>

<script>

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

console.log(myul.parentNode);

</script>

</body>

**1.2.5 previousSibling nextSibling**

获取兄弟节点

使用previousSibling和nextSibling可以获得该匀速的前一个和后一个兄弟节点，只能获取一个。

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

console.log(mydiv.previousSibling);

console.log(mydiv.nextSibling);

**1.2.6 offsetWidth和offsetHeight**

元素的宽高属性

需要获取一个元素的高度和宽度，使用style是无法获取的。所以我们选择使用offsetWidth和offsetHeight属性。该属性可以获取元素的占位宽高，它也包含了内边距和边框。

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

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

console.log(mydiv.offsetWidth);

clientWidth和clientHeight

clientWidth和clientHeight属性也可以获取元素的宽和高，但与offsetWidth不同的是不包含边框。

console.log(mydiv.clientWidth);

**1.2.7 offsetLeft和offsetTop**

子元素与父元素的距离

offsetLeft和offsetTop是距离body左边界和上边界的距离，但如果该子元素使用了定位属性，则offsetLeft和offsetTop参照的就不在是body而是距离他最近的父元素。

<div id="baba">

<div id="erzi"></div>

</div>

<script>

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

console.log(erzi.offsetLeft);

console.log(erzi.offsetTop)

</script>

**1.3 Element**