**什么是 DOM？**

通过 JavaScript，您可以重构整个 HTML 文档。您可以添加、移除、改变或重排页面上的项目。

要改变页面的某个东西，JavaScript 就需要对 HTML 文档中所有元素进行访问的入口。这个入口，连同对 HTML 元素进行添加、移动、改变或移除的方法和 属性，都是通过文档对象模型来获得的（DOM）。

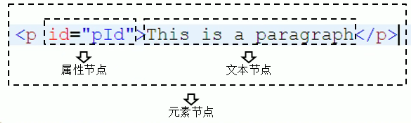
在 1998 年，W3C 发布了第一级的 DOM 规范。这个规范允许访问和操作 HTML 页面中的每一个单独的元素。

所有的浏览器都执行了这个标准，因此，DOM 的兼容性问题也几乎难觅踪影了。

DOM 可被 JavaScript 用来读取、改变 HTML、XHTML 以及 XML 文档。

**节点：Node**

根据 DOM，HTML 文档中的每个成分都是一个节点。节点是构成网页的最基本的组成部分。



DOM 是这样规定的：

文档节点 ：html文档

元素节点 ：html标签

文本节点 ：标签的文本

属性节点：元素的属性

注释属于注释节点

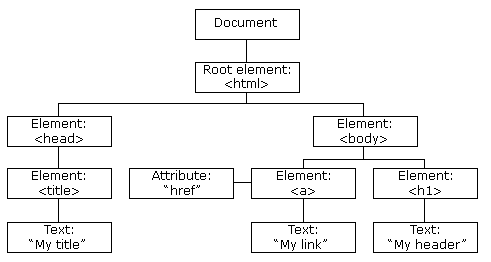
高亮常用节点

**模型：Node层次**

节点彼此都有等级关系。

HTML 文档中的所有节点组成了一个文档树（或节点树）。HTML 文档中的每个元素、属性、文本等都代表着树中的一个节点。树起始于文档节点，并由此继续伸出枝条，直到处于这棵树最低级别的所有文本节点为止。

下面这个图片表示一个文档树（节点树）：



**节点的属性**

**1.nodeName：属性含有某个节点的名称。**

元素节点的 nodeName 是标签名 大写

属性节点的 nodeName 是属性名

文本节点的 nodeName 永远是 #text

文档节点的 nodeName 永远是 #document

**2.nodeValue**

对于文本节点，nodeValue 属性包含文本。

对于属性节点，nodeValue 属性包含属性值。

文档节点和元素节点的nodeValue是null。

**3.nodeType**

|  |  |
| --- | --- |
| 元素类型 | 节点类型 |
| 元素节点 | 1 |
| 属性节点 | 2 |
| 文本节点 | 3 |
| 文档节点 | 9 |
| 注释节点 | 8 |

**事件**

就是用户和浏览器之间的交互行为

比如：点击按钮、鼠标移动、关闭窗口

**处理事件**

我们可以在事件对应的属性中设置一些js代码

这样当事件被触发时，这些代码会被执行

<button id="btn" onclick="alert('点我干嘛');">

我是一个按钮

</button>

这种行内式写法我们称为结构和行为耦合，不方便维护，不推荐使用

可以为按钮的对应事件绑定处理函数的形式来响应事件

这样当事件被触发时，其对应的函数将会被调用，这是一个回调函数，由浏览器调用

//绑定一个事件

btn.onclick = function(){

alert("点我干嘛");

}

浏览器在加载一个页面时,是按照自上向下的顺序加载的，

读取到一行就运行一行,如果将script标签写到页面的上边,

在代码执行时，页面还没有加载。

将js代码编写到页面的下部就是为了，可以在页面加载完毕后再执行js代码

如果想把js代码写在head标签中

就要为window绑定一个onload事件，onload事件会在整个页面加载完成之后才触发

该事件对应的响应函数将会在页面加载完成之后执行，

这样可以确保我们的代码执行时所有的DOM对象已经加载完毕了

出于性能考虑，写在下边更好，页面会先加载

写上边优点是好管理

**获取元素节点**

通过document对象调用

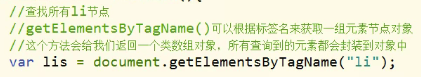
**getElementByID()**

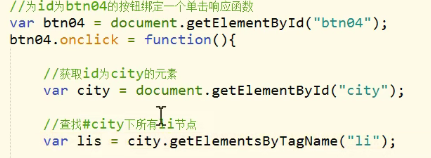
通过id属性获取一个元素节点对象



**getElementsByTagName()**

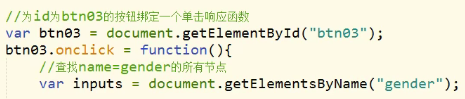
通过标签名获取一组元素节点对象





**getElementsByName()**

通过name属性获取一组元素节点对象



**获取元素节点的子节点**

通过具体的元素节点调用

**1. getElementsByTagName()**

——方法，返回当前节点的指定标签名后代节点

**2. childNodes**

——属性，表示当前节点的所有子节点

childNodes是W3C推出的

会获取包括文本节点在内的所有节点

根据DOM元素节点（标签）之间的空白、换行内容 也会当成文本节点

注意：在IE8及以下的浏览器中，不会将空白文本当成子节点。

**3.children**

——属性可以获取当前元素的所有子元素（标签）

微软推出的 推荐使用

截图.png

**缺点：在IE 678下 会把注释当成元素节点**

**4. firstChild**

——属性，表示获取当前节点的第一个子节点，包括空白文本节点

等于childNodes[0]

**5.firstElementChild**

——属性，表示获取当前节点的第一个子元素，不会获取到文本

注意：因为不支持IE8及以下的浏览器

如果要兼容他们不建议使用

**5.lastChild**

——属性，表示获取当前节点的最后一个子节点，包括空白文本节点

等于childNodes[childnodes.length-1]

**获取父节点和兄弟节点**

通过具体的节点调用

**1. parentNode**

——属性，表示当前节点的父节点，文本节点没有子节点，所以父节点不会是文本节点



**2.previousElementSibling**

——属性，表示当前节点的前一个兄弟元素节点

不会获取到空白的文本节点

注意：在谷歌和ie9+下没有任何问题

在ie678会获取到undefined

**3. previousSibling**

——属性，表示当前节点的前一个兄弟节点

注意：在google下，出现文本节点

在ie678下，可以使用

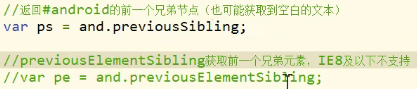
**4.nextElementSibling**

——属性，表示当前节点的后一个兄弟元素节点

不会获取到空白的文本节点

注意：在谷歌和ie9+下没有任何问题

在ie678会获取到undefined



**5. nextSibling**

——属性，表示当前节点的后一个兄弟节点

注意：在google下，出现文本节点

在ie678下，可以使用

**兼容写法：**

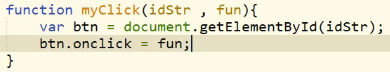
**obj.nextElementSibling || obj.nextSibling**

**定义一个函数，专门用来为指定元素绑定单击响应函数**

**参数:**

idStr 要绑定单击响应函数的对象的id属性值

fun 表示事件的回调函数，当单击元素时，该函数将会被触发





**innerHTML**

innerHTML用于获取元素内部的html代码：开始标签和结束标签中间的内容

对于自结束标签，这个属性没有意义

如果要读取元素节点的属性：

直接使用 元素.属性名

注意：如果读取class的属性，要使用元素.className 因为class是保留字 不能用

**innerText**

该属性可以获取到元素内部的文本内容

它和innerHTML类似，不同的是它会自动将html去除

**全选按钮**

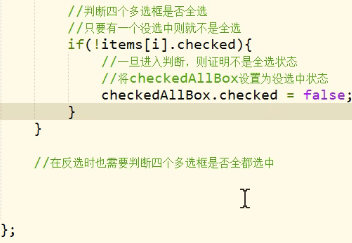


**全不选**

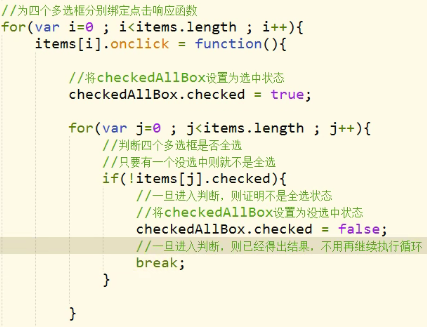


**反选按钮**





**同步选中状态**



**DOM查询的剩余方法**

**body**

在document中有一个属性body，保存的是body的引用

**语法：**

var body = document.body;

**html**

根标签

**语法：**

var html = document.documentElement;//保存的html根标签

**all**

代表页面中所有的元素

**语法：**

var all = document.all;

等于：var all = document.getElementByTagName("\*");

**getElementByClassName(" ");**

根据元素的class属性值查询一组元素节点对象

但是该方法不支持IE8及以下版本的浏览器

**语法：**

var box1 = document.getElementByClassName("box1");

**document.querySelector()**

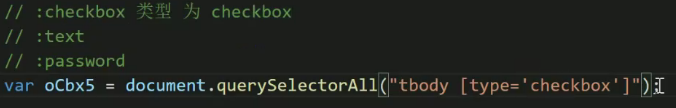
——需要一个选择器的字符串作为参数，可以根据一个css选择器来查询一个元素节点对象

——虽然IE8中没有getElementByClassName()但是可以使用querySelector()

——使用该方法总会返回唯一的一个元素，如果满足条件的有多个，那么它只会返回第一个

**语法：**

var box1 = document.querySelector( ".box1");



**document.querySelectorAll()**

——该方法和querySelector()类似，不同的是它会将所有符合条件的元素封装到一个数组中返回

——即使符合条件的元素只有一个，它也会返回一个数组

**语法：**

var box1 = document.querySelectorAll( ".box1 div");

这两个方法是html5新增的都是用document对象来调用，两个方法使用相同，

都是传递一个选择器字符串作为参数，方法会自动根据选择器

字符串去网页中查找元素。

**DOM对象方法**

**createElement()**

创建元素节点对象

**语法：**

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

**用法：**

它需要一个标签名作为参数,将会根据该标签名创建元素节点对象，并将创建好的对象作为返回值返回

**creatTextNode()**

创建文本节点对象

**语法：**

var gztext = document.creatTextNode("广州")

**用法：**

需要一个文本内容作为参数，将会根据该内容创建文本节点，并将新的节点返回

**creatComment()**

添加注释

**语法：**

需要添加注释的节点.creatComment("注释内容");

向需要添加注释的节点添加指定内容的注释

**appendChild()**

把新的子节点添加到指定的父节点

**语法：**

li.appendChild(gztext);

city.appendChild(li);

向父节点中添加一个新的子节点

**用法：**

父节点.appendChild(子节点)

**insertBefore()**

在指定的子节点前面插入新的子节点

**语法：**

city.insertBefore(li , bj);

**用法：**

父节点.insertBefore(新节点，旧节点);

注意：如果它的第二个参数是空的，那么则等价于appendChild()

**replaceChild()**

替换子节点

——可以使用指定的新子节点去替换旧子节点

**语法：**

city.replaceChild(li , bj)

**用法：**

父节点.replaceChild(新节点，旧节点)

**cloneNode()**

cloneNode()方法可以把子节点复制出来

**参数：**

true表示深克隆 复制标签和子节点 ,

false表示浅克隆 只复制自身的结构（标签）

**语法：**

var newNode = box.firstChild.cloneNode(true);

**removeChild()**

删除子节点

**语法：**

city.removeChild(bj)

=bj.parentNode.removeChild(bj)

**用法：**

父节点.removeChild(子节点)

=子节点.parentNode.removeChild(子节点)——更常用

**remove()**

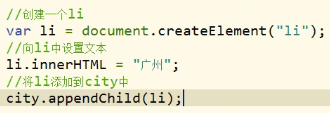
删除 在IE低版本下有兼容问题

**语法：**

bj.remove()//自杀模式

使用innerHTML也可以完成一些DOM增删改的操作

一般会两种方式结合使用



**confirm**

用于弹出一个带有确认和取消的提示框

需要一个字符串作为参数，该字符串会作为提示文字显示出来

如果用户点击确认则会返回true，点击取消则会返回false

**点击超链接**

点击超链接以后,超链接会跳转页面,这个是超链接的默认行为，

但是此时我们不希望出现默认行为，可以通过在响应函数的最后return false来取消默认行为