Day 09-10---DOM基础

一、DOM概述(079)

1.1 DOM概念

DOM（文档对象模型）是针对 HTML 和 XML 文档的一个 API（应用程序编程接口）。DOM 描绘了一个层次化的节点树，允许开发人员添加、移除和修改页面的某一部分。

W3C DOM标准分为3部分：

- core DOM - 针对任何结构化文档的标准模型
- XML DOM - 针对 XML 文档的标准模板
- HTML DOM - 针对 HTML 文档的标准模型

备注：HTML DOM 是关于如何获取、修改、添加或删除 HTML 元素的标准，即操作HTML的元素

DOM级别

DOM Level 1:于1998年10月成为W3C的推荐标准。DOM 1级由两个模块组成：DOM核心（DOM Core）和DOM HTML。

DOM Level 2:对DOM level 1做了扩展

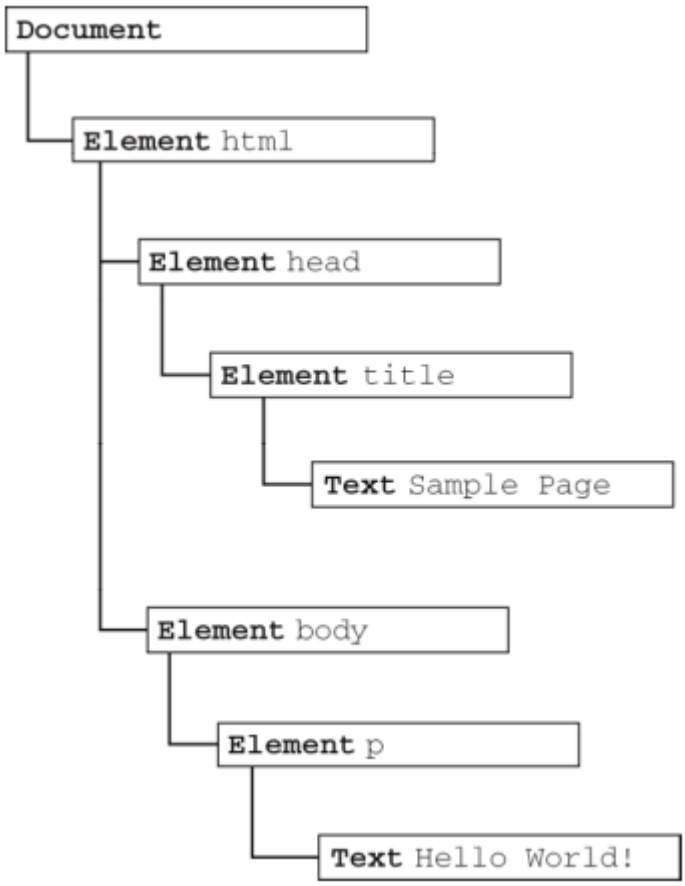
DOM Level 3:对DOM level 2做了进一步的扩展

DOM Level 0: 首先我们的确定的是在DOM标准中并没有DOM 0级的。所谓的DOM 0级是DOM历史坐标中的一个参照点而已，具体说呢，DOM 0级指的是IE4和Netscape 4.0这些浏览器最初支持的DHTML。

1.2 DOM节点概述

> 看下面的一段HTML：

```
<html>
  <head>
    <title>DOM节点分类</title>
  </head>
  <body>
    <p id="p">育知同创欢迎你!</p>
  </body>
  <!-- 注释 -->
</html>
```

- 

> 说明：

1. document节点是每个文档的根节点
2. document节点下面只有一个 html节点，我们称之为文档元素。(documentElement)
    - 文档元素是文档的最外层元素，其他元素都包含在文档元素中。
    - 一个文档只能有一个文档元素，在html中文档元素永远是 元素。
3. 在DOM树中，html文档中每一处标记都可以用树中的一个节点来表示。
    - html(元素)标签，通过元素节点表示
    - 属性，通过属性节点来表示
    - 文档类型，通过文档类型节点来表示
    - 注释，通过注释类型来表示

1.3 DOM节点分类

> DOM中，共有12中节点类型，每一个节点，必是这12中类型中的一种。

1. Node.ELEMENT_NODE (1) //元素
2. Node.ATTRIBUTE_NODE (2) //元素的属性HELLO
3. Node.TEXT_NODE (3) //<![CDATA[文本]]>中括着的纯文本，它没有子节点
4. Node.CDATA_SECTION_NODE (4) //子节点一定为TextNode
5. Node.ENTITY_REFERENCE_NODE (5) //文档中的实体引用
6. Node.ENTITY_NODE (6) //DTD中的实体定义<!ENTITY foo “foo”>，无子节点
7. Node.PROCESSING_INSTRUCTION_NODE (7) //表示一个文档处理程序使用的特有指令。，无子节点
8. Node.COMMENT_NODE (8) //注释

9. Node.DOCUMENT_NODE (9) //最外层的Root element 包括所有其它节点　根节点
10. Node.DOCUMENT_TYPE_NODE (10) //DTD，<!DOCTYPE.....> 每个文档节点都有一个DocumentType节点，它提供文档类型的定义。
11. Node.DOCUMENT_FRAGMENT_NODE (11)　//可以将文档片段看作轻量级或者更小型的Document节点。定义这种数据类型是因为通常希望只提取文档的一部分来处理。
12. Node.NOTATION_NODE (12) //DTD中的Nation定义　在XML文档中表示一个符号。

> 可以通过获取一个节点的nodeType属性来得到节点的类型。
>
> 经常用到了比较重要的节点：(记住每个节点的数字值)

元素类型	NodeType
元素	1
属性	2
文本	3
注释	8
文档	9

1.4 节点属性（特性）

1. nodeName：节点名称(nodeName 始终包含 HTML 元素的大写字母标签名) nodeName 是只读的

- 元素节点的 nodeName 与标签名相同
- 属性节点的 nodeName 与属性名相同（元素.getAttributeNode("属性名")获取属性节点）
- 文本节点的 nodeName 始终是 #text（通过元素的子节点获取）
- 注释节点的nodeName是#comment（通过元素的子节点获取）
- 文档节点的 nodeName 始终是 #document

2. nodeValue：节点值　（设置或返回节点的值）

- 元素节点的 nodeValue 是 undefined 或 null
- 属性节点的 nodeValue 是属性值
- 文本节点的 nodeValue 是文本本身
- 注释节点的nodeValue是注释里面的内容
- 文档节点的nodeValue 是null

3. nodeType：节点类型

- 元素 element　1
- 属性 attr　2
- 文本 text　3
- 注释 comments　8
- 文档 document　9

二、DOM核心对象---document对象(080)

JavaScript 通过 Document 类型表示文档。在浏览器中， document 对象是 HTMLDocument （继承自 Document 类型）的一个实例，表示整个 HTML 页面。而且， document 对象是 window 对象的一个属性，因此可以将其作为全局对象来访问。

- nodeType 的值为 9；
- nodeName 的值为 "#document" ；
- nodeValue 的值为 null ；
- parentNode 的值为 null ；
- ownerDocument 的值为 null ；
- 其子节点可能是一个 DocumentType （最多一个）、 Element （最多一个）、 ProcessingInstruction 或 Comment

> 通过document对象，不仅可以取得与页面有关的信息，而且还能操作页面的外观及其底层结构。

2.1 获取节点方法1：getElementById(id)(081)

> 在整个文档中，根据元素id来获取元素节点

```
<body>
    <h1 id="myH1" class="hTitle">这个是标题</h1>
    <script type="text/javascript">
        //根据id获取元素
        var h1 = document.getElementById("myH1");
        alert(h1.id);  //查看这个元素的id
        alert(h1.className);  //元素的 className
        alert(h1.innerHTML);    //查看元素包含的所有HTML文本
    </script>
</body>
```

> 注意：

- 如果有多个元素的id相同，则只返回第一个元素。一般情况，不要给多个元素起同名id

2.2 获取节点方法2：getElementsByTagName(tagName)(082)

> 根据标签名获取元素节点。因为一个文档中会有多个同名标签，所以这个方法返回的是多个Element组成的集合
>
> 为了提高性能，在开发的时候应该尽量避免使用这种方式去查找元素，应该使用getElementById

```
<body>
    <h1 id="myH1" class="hTitle">这个是标题1</h1>
    <h1 id="myH2" class="hTitle">这个是标题2</h1>
    <h1 id="myH3" class="hTitle">这个是标题3</h1>
    <script type="text/javascript">
        var h1s = document.getElementsByTagName("h1");
        alert(h1s.length);
        for (var i = 0; i < h1s.length; i++) {
            alert(h1s[i].id);     //可以通过下标访问     ht[i]
           alert(h1s.item(i).id);  //  也可以通过item下标访问 item(i)
        }
    </script>
</body>
```

2.3 获取节点方法3：getElementByName(name)(083)

> 这个是通过标签的name属性的值来获取元素的。由于会出现多个元素name属性的值相等，所以返回的是多个Element组成的集合.
>
> 注意：不是所有的元素都有name属性，只有 表单标签 才有。而且某些低版本浏览器兼容有问题。慎用

```
<body>
    <form action="">
        爱好：<br/>
        <input type="checkbox" name="lover" value="swimming">游泳<br/>
        <input type="checkbox" name="lover" value="net">上网
    </form>
    <script type="text/javascript">
        var lovers = document.getElementsByName("lover");
        for (var i = 0; i < lovers.length; i++) {
            alert(lovers[i].value);
           //alert(lovers.item(i).value)
        }
    </script>
</body>
```

2.4 获取节点的方法4：querySelector(css选择器)(084)

参数：必须。指定一个或多个匹配元素的 CSS 选择器。 可以使用它们的 id, 类, 类型, 属性, 属性值等来选取元素。

对于多个选择器，使用逗号隔开，返回一个匹配的元素。

返回值：匹配指定 CSS 选择器的 第一个元素 。 如果没有找到，返回 null。

```
<body>
    <p id="demo">id="demo" 的 第一个 p 元素</p>
    <p id="demo">id="demo" 的 第二个 p 元素</p>

    <button onclick="myFunction()">点我修改id = demo 的第一个元素内容</button>
    <script>
        function myFunction() {
            //返回第一个id=demo的元素
            document.querySelector("#demo").innerHTML = "Hello World!";
        }
    </script>
</body>
```

```
document.querySelector("p");     // 获取文档中第一个p元素(元素选择器)
document.querySelector("p.example");  //获取文档中 class="example" 的第一个 <p> 元素:
```

2.5 获取节点的方法5：querySelectorAll(css选择器)(085)

> querySelector只能获取第一个满足条件的元素，如果想获取所有满足条件的元素，可以使用HTML5引入的新方法qureySelectorAll

```
<body>
    <p id="demo">id="demo" 的 第一个 p 元素</p>
    <p id="demo">id="demo" 的 第二个 p 元素</p>

    <button onclick="myFunction()">点我修改id = demo 的所有元素</button>
    <script>
        function myFunction() {
            var all = document.querySelectorAll("#demo");
            for (var i = 0; i < all.length; i++) {
                all[i].innerHTML = "哈哈哈"
            }
        }
    </script>
</body>
```

2.6 获取documentElement元素(086)

> documentElement 属性以一个元素对象返回一个文档的文档元素。HTML 文档返回对象为 HTML元素。

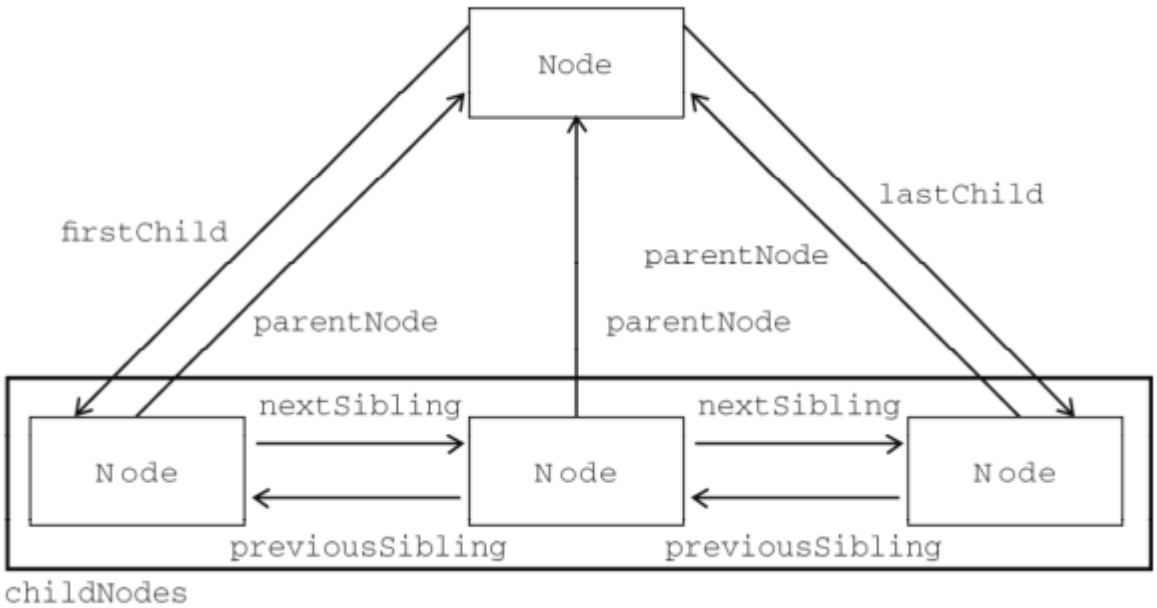
```
<body>
    <script type="text/javascript">
        alert(document.documentElement.nodeName);    // html
        alert(document.documentElement.nodeValue);   // null
        alert(document.documentElement.nodeType);    // 1
    </script>
</body>
```

三、Node关系---获取操作

3.1 节点与节点之间的关系

> 说明：

1. 父（parent）节点　父节点拥有任意数量的子节点
2. 子（child）节点 子节点拥有一个父节点
3. 兄弟（sibling）节点 同级的子节点被称为同胞（兄弟或姐妹）。同胞是拥有相同父节点的节点
4. 根 (root) 节点　　一个文档只能有一个根节点。对html文档来说，根节点就是documentElement。根节点不可能有父节点

- 

3.2 childNodes属性：获取一个元素的所有子节点(089)

> childNodes：获取所有子节点。(但是不能获取到子节点的子节点)

```
<body>
    <div id="div">
        <a href="">搜狐</a>
        <div>
            <a href=""></a>
        </div>
         <input type="text" name="user" />
    </div>
    <script type="text/javascript">
        //通过Element元素的childNodes属性获取到这个元素下面的所有直接子节点
        var nodes = document.getElementById("div").childNodes;
        for (var i = 0; i < nodes.length; i++) {
            alert(nodes[i].nodeName)
        }
    </script>
</body>
```

3.3 firstChild属性：获取一个元素的第一个子节点(087)

> 如果选定的节点没有子节点，则该属性返回 NULL。

```
<body>
    <div id="div"><a href="">搜狐</a>
        <div>
            <a href=""></a>
        </div>
         <input type="text" name="user" />
    </div>
    <script type="text/javascript">
        var div = document.getElementById("div");
        var fistChild = div.firstChild; //第一个子节点
        alert(fistChild.nodeName);  // A
    </script>
</body>
```

3.4 lastChild属性：获取一个元素的最后一个子节点(088)

> lastChild 属性可返回文档的最后一个子节点。

```
<body>
    <div id="div"><a href="">搜狐</a>
        <div>
            <a href=""></a>
        </div>
         <input type="text" name="user" /></div>
    <script type="text/javascript">
        var div = document.getElementById("div");
        var lastChild = div.lastChild;  //最后一个节点
        alert(lastChild.nodeName);  // INPUT
    </script>
</body>
```

3.4 parentNode属性：获取一个元素的父节点(092)

```
<body>
    <div id="div"><a href="">搜狐</a>
        <div>
            <a href=""></a>
        </div>
         <input type="text" name="user" /></div>
    <script type="text/javascript">
        var div = document.getElementById("div");
        var parentNode = div.parentNode;     //父节点
        alert(parentNode.nodeName); // BODY
    </script>
</body>
```

3.5 previousSibling属性：获取一个元素的上一个兄弟节点(090)

```
<body>
    <div id="div"><a href="">搜狐</a><div id="div1">
            <a href=""></a>
        </div>
         <input type="text" name="user" /></div>
    <script type="text/javascript">
        var div1 = document.getElementById("div1");
        var preSibling = div1.previousSibling;  //获取div1的父节点
        alert(preSibling.nodeName); // A
    </script>
</body>
```

3.6 nextSibling属性：获取一个元素的下一个节点(091)

```
<body>
    <div id="div"><a href="">搜狐</a>
        <div id="div1">
            <a href=""></a>
        </div>
        <input type="text" name="user" />
    </div>

    <script type="text/javascript">
        var div1 = document.getElementById("div1");
        var neSibling = div1.nextSibling;   // 下一个兄弟节点
        alert(neSibling.nodeName);  // #text
    </script>
</body>
```

补充：由于上述都是节点操作，所以获取到的节点会包含一些我们并不关心的内容，比如空节点。下面的几个api只获取元素节点。

children	子元素节点列表
firstElementChild	第一个子元素
lastElementChild	最后一个子元素
nextElementSibling	下一个元素
previousElementSibling	上一个元素
childElementCount	子元素的数量

四、Node关系---修改操作

4.1 createTextNode()：创建文本节点(094)

> document.createTextNode(text)
>
> 参数：文本
>
> 返回值：创建的文本标签
>
> 注意：这个方法是 document的方法，因为创建元素这么大的事，只能document来干，找某个element来创建不靠谱。

```
<body>
    <div id="div"></div>
    <script type="text/javascript">
        var textNode = document.createTextNode("我是一个开心的文本！！！"); //创建一个文本节点
        alert(textNode.nodeName + "\n" + textNode.nodeValue);    // #text     我是一个开心的文本！！！
    </script>
</body>
```

4.2 appendChild() : 给一个元素添加child节点(095)

> element.appendChild(node)
>
> 参数：必须。表示要添加的节点
>
> 返回值：参数对象本身
>
> 注意：append的意思是追加。如果元素已经有child，则追加到别的child之后。也就是说新添加的 child 进去之后一定是 lastChild

```
<body>
    <div id="div"></div>
    <script type="text/javascript">
        var div = document.getElementById("div");
        var textNode = document.createTextNode("我是一个开心的文本！！！");
        div.appendChild(textNode);  //把刚才创建的文本添加到div标签中。
    </script>
</body>
```

4.3 createElement(nodename): 创建元素节点(093)

> document.createElement(nodename)
>
> 参数：必须。要创建的元素的名称
>
> 返回：创建的元素节点

```
<body>
    <div id="div"></div>
    <script type="text/javascript">
        var div = document.getElementById("div");
        var newElement = document.createElement("a");    //创建一个a标签
        var textNode = document.createTextNode("首页");    // 创建一个文本节点
        newElement.appendChild(textNode); //把 文本节点添加到a标签中
        var v = div.appendChild(newElement);     // 把刚刚创建的a标签，添加为div标签的子标签
        alert(v === newElement) // true 证明参数和返回值确实为同一个对象
    </script>
</body>
```

4.4 insertBefore()：在一个节点前插入新节点(096)

> node.insertBefore(newnode,existingnode)
>
> 参数1：要插入的新节点
>
> 参数2：目标节点。会把参数1节点插入这个节点之前。

```
<body>
    <ul id="language">
        <li>java</li>
        <li>c++</li>
    </ul>
    <button onclick="myFunction()">点我在java前插入JavaScript</button>

    <script>
        function myFunction() {
            var newItem = document.createElement("li")
            var textnode = document.createTextNode("JavaScript")
            newItem.appendChild(textnode)

            var languageList = document.getElementById("language")
            //把newItem添加到languageList的第1个子节点之前
            languageList.insertBefore(newItem, list.firstChild);
        }
    </script>
</body>
```

4.5 removeChild(): 移除子节点(099)

> node.removeChild(node)
>
> 参数：必须。要移除的那个子节点

```
<body>
    <ul id="language">
        <li>java</li>
        <li>c++</li>
        <li>oc</li>
        <li>php</li>
        <li>c#</li>
        <li>JavaScript</li>
    </ul>

    <button onclick="myFunction()">点我移除第1条语言</button>
    <script>
        function myFunction() {
            var languageList = document.getElementById("language");
            //获取到languageList下面的所有的 li 标签
            var liList = languageList.getElementsByTagName("li");
            // 移除第一个 li 元素。 如果重复调用，则每次都是移除第1个
            languageList.removeChild(liList[0]);
            alert(liList.length)
        }
    </script>
</body>
```

4.6 replaceChild()：替换子节点(097)

> node.replaceChild(newnode,oldnode)
>
> 参数1：新节点
>
> 参数2：旧节点

```
<body>
    <ul id="language">
        <li>java</li>
        <li>c++</li>
        <li>oc</li>
        <li>php</li>
        <li>c#</li>
        <li>JavaScript</li>
    </ul>
    <button onclick="myFunction()">点我替换第一条语言</button>
    <script>
        function myFunction() {
            var languageList = document.getElementById("language");
            var newLi = document.createElement("li");
            newLi.innerHTML="JavaScript";
            //用新节点去替换旧节点。
            languageList.replaceChild(newLi, languageList.childNodes[1]);
        }
    </script>
</body>
```

4.6 cloneNode()：克隆节点(098)

> node.cloneNode(deep):
>
> cloneNode() 方法可创建指定的节点的精确拷贝。拷贝所有属性和值。
>
> 该方法将复制并返回调用它的节点的副本。如果传递给它的参数是 true，它还将递归复制当前节点的所有子孙节点。否则，它只复制当前节点。

```
<body>
    <ul id="language">
        <li>java</li>
        <li>c++</li>
        <li>oc</li>
        <li>php</li>
        <li>c#</li>
        <li>JavaScript</li>
    </ul>
    <button onclick="myFunction()">点我clone整个列表</button>
    <script>
        function myFunction() {
            var languageList = document.getElementById("language");
            //克隆languageList整个节点，传入true表示深度copy，所有的子节点都copy
            var newList = languageList.cloneNode(true);
            document.getElementsByTagName("body")[0].appendChild(newList);
        }
    </script>
</body>
```

五、元素属性操作(100)

> 在HTML中，一个元素会有很多属性，比如id属性，class属性，title属性等等，如何操作这些呢？

5.1 getAttribute():获取属性值

> element.getAttribute(attributename)
>
> 返回元素的指定属性的值。
>
> 参数：元素的某个属性的名 (id, className, title)

```
<body>
    <a id="id1" href="http://www.yztcedu.com" target="_blank">育知同创官网</a>.
    <br />
    <p>
        <button onclick="myFunction()">点我获取超级链接的 id 属性值，tafget 属性值</button>
    </p>
    <script>
    function myFunction() {
        var a1 = document.getElementById("id1");
        alert("id = " + a1.getAttribute("id") + "\ntarget = " + a1.getAttribute("target"));
    }
    </script>
</body>
```

5.2 setAttribute(): 设置属性值

> element.setAttribute(attributename,attributevalue)
>
> 作用：创建建或改变某个新属性。如果指定属性已经存在,则只设置该值。如果属性不存在，则创建该属性并设置属性值。
>
> 参数1：属性名
>
> 参数2：新的属性值

```
<body>
    <input value="OK" >
      <br/>
    <button onclick="myFunction()">点我把上述文本框的 type 属性设置为 button 样式</button>
    <script>
        function myFunction() {
            var inputs = document.getElementsByTagName("input");
            //把input的 type属性设置为button
            inputs[0].setAttribute("type", "button");
        };
    </script>
</body>
```

5.2 removeAttribute() : 移除属性

> element.removeAttribute(attributename)
>
> 参数：必需。规定要删除的属性的名称。

```
<body>
    <input id="input1" type="button" value="点我可以转换我的状态" onclick="myFunction();">
    <script>
        function myFunction() {
            var input1 = document.getElementById("input1");
            var typeValue = input1.getAttribute("type");
            if(typeValue){
                input1.removeAttribute("type"); //如果type属性有值就把这个属性去掉
            }else {
                input1.setAttribute("type", "button");  //如果type属性不存就添加属性。
            }
        };
    </script>
</body>
```

六、元素节点的常用属性(101)

> 元素节点除了有所有的节点都有的属性比如：nodeName、nodeValue、nodeType外，还有一些特有的属性，来方便操作标签元素。

6.1 tagName属性

> tagName属性是nodeName的另一中写法。注意区别是：tagName只在标签节点中有，而nodeName在所有类型的(12种)节点中都存在。
>
> 它的值表示的就是标签的名字。

```
<body>
    <p id="p1">我是一个p标签</p>
    <script type="text/javascript">
        var p1 = document.getElementById("p1");
        alert(p1.tagName);  // p
    </script>
</body>
```

6.2 innerHTML属性

> innerHTML 属性设置或返回标签的开始和结束标签之间的 HTML。
>
> 值为为文本
>
> 注意：这个属性的值是这个标签的开始和结束部分之间的所有内容，但是

```
<body>
    <p id="p1"><a href="http://www.yztcedu.com">点我进入育知同创官网</a></p>

    <P><button onclick="getHTML();">点我获取超级链接的所有内容</button></P>

    <p><button onclick="setHTML();">点我更改整个超级链接</button></p>

    <script type="text/javascript">
        var p1 = document.getElementById("p1");
        function getHTML () {
            alert(p1.innerHTML);
        }
        function setHTML () {
            p1.innerHTML = "<a href='https://www.baidu.com'>点我进入百度主页</a>"
        }
    </script>
</body>
```

6.3 innerText属性

> innerText值获取标签中的文本内容，子标签本身不会获取到。
>
> 去修改的时候，即使带有标签也会把标签作为纯文本来对待，而不会解析为标签

```
<body>
    <p id="p1"><a href="http://www.yztcedu.com">点我进入育知同创官网</a></p>

    <P><button onclick="getHTML();">点我获取超级
      链接的所有内容</button></P>

    <p><button onclick="setHTML();">点我更改整个超级链接</button></p>

    <script type="text/javascript">
        var p1 = document.getElementById("p1");
        function getHTML () {
            alert(p1.innerText);
        }
        function setHTML () {
            p1.innerText = "<a href='https://www.baidu.com'>点我进入百度主页</a>"
        }
    </script>
</body>
//其实还有一个属性：textContent 和innerText的作用相同。
```

点我进入育知同创官网

点我获取超级链接的所有内容

点我更改整个超级链接

点我进入百度主页

点我获取超级链接的所有内容

点我更改整个超级链接

6.4 id属性

> 就是指的元素节点的id属性的值，与getAttribute("id")的值是一样的。

6.5 className属性

> className 属性设置或返回元素的 class 属性。(因为class在js中是关键字，所以这个地方把属性名字改成了className)

```
<body>
    <p class="p1">这是一个段落</p>
    <script type="text/javascript">
        var p1 = document.querySelector(".p1");
        alert(p1.className);
    </script>
</body>
```

6.7 value属性

> 如果一个标签可以拥有value值，则可以可以通过element.value来获取。
>
> 一般表单数据才具有vlaue：input、textarea、select

```
<body>
    <input id="in1" type="text" value="abc" />
    <p>
       <button id="btn" onclick="getValue();" value="abc">获取上面输入框的value属性</button>
    </p>
    <script type="text/javascript">
        var in1 = document.getElementById("in1");
        function getValue () {
            alert(in1.value);    // 获取标签的value值
        }
    </script>
</body>
```

七、样式表的属性---css脚本化

> 可以通过JavaScript访问css的属性，并修改css属性

7.1 获取和修改行间样式表

> 有2中方式访访问到行间样式表
>
> 1. element.style.css属性名
> 2. element.style["属性名"]

```
<body>
    <div id="box1"></div>
    <script type="text/javascript">
        var box1 = document.getElementById("box1");
        box1.style.width = "100px"; //设置css属性.    设置所有属性都要使用引号括起来。
        box1.style["height"] = "100px";
        box1.style.backgroundColor = "blue";
        alert(box1.style["width"]);
    </script>
</body>
```

7.2 获取内部样式表和外部样式表

> 1. 对ie浏览器：对象.currentSytle["属性名"]
> 2. 其他浏览器：window.getComputedStyle(对象，null)["属性名"]
>
> 注意：内部样式表中的属性和外部样式表中的属性只能获取不能修改。如果想修改需要通过行间样式表修改，行间样式表的优先级最高，会覆盖内部样式表和外部样式表。

```
<body>
    <div id="box1"></div>
    <script type="text/javascript">
        var box1 = document.getElementById("box1");
        // alert(box1.currentStyle["width"]); //只支持IE浏览器
        // alert(window.getComputedStyle(box1, null)["height"]); //支持浏览器外的其他浏览器
        alert(getStyle(box1, "backgroundColor"));
        /*
            为了简化书写和兼容浏览器，一般封装一个方法出来
        */
        function getStyle (obj, attributeName) {
            if(obj.currentStyle){    //如果存在对象，则是在ie浏览器
                return obj.currentStyle[attributeName];
            }else { //其他浏览器
                return window.getComputedStyle(obj, null)[attributeName];
            }
        }
    </script>
</body>
```

补充：

> offsetWidth、offsetHeight 可以获取宽和高，包括 border 和 padding，其实是这个元素的实际占据的空间。但是只能获取不能修改

CSS样式属性	Style对象属性
color	color
font-size	fontSize
font-family	fontFamily
background-color	backgroundColor
background-image	backgroundImage
display	display

结论：一般情况下，css的样式属性中出现“-”号，则对应的style属性是：去掉“-”号，把“-”号后面单词的第一字母大写。如果没有“-”号，则两者一样。