DOM

2020年4月12日 15:20

DOM是中立于平台和语言的接口，它允许程序和脚本动态地访问和更新文档的内容、结构和样式

对网页进行增删改查的操作（Document Object Model文档对象模型）

（1）DOM查找：【以下parent均指父元素，如div】

- 按id属性查找，精确查找一个元素对象

```
var elem=document.getElementById("id")
```

- Var elem=document.getElementById('id')

注意：getElementById只能用在document上

注意：不是所有元素都有id的

```
<ul id="myList">
        <li id="m1">首页</li>
        <li id="m2">企业介绍</li>
        <li id="m3">联系我们</li>
</ul>
```

```
var ul = document.getElementById('myList');
console.log(ul );
```

- 按标签名查找
- Var elems=parent.getElementsByTagName（‘tag’）
- 查找指定parent节点下所有标签为tag的子代节点

强调：

1. 可用在任意父元素上
2. 不仅查直接子节点，而且查所有子代节点
3. 返回一个动态集合
   即使只找到一个元素，也返回集合
   必须用[0],取出唯一元素

注意：即使找到一个元素，调用时也要用下标的方式调用

```
<ul id="myList">
        <li id="m1">首页</li>
        <li id="m2">企业介绍</li>
        <li id="m3">联系我们</li>
</ul>
```

```
var ul = document.getElementById('menuList');
var list = ul.getElementsByTagName('li');
console.log(list );
```

- 通过name属性查找
- Document.getElementsByName（‘name属性名’）

可以返回DOM树中具有指定name属性的所有子元素

```
<form id="registerForm">
        <input type="checkbox" name="boy"/>
        <input type="checkbox" name="boy"/>
        <input type="checkbox" name="boy"/>
</form>
```

```
var list = document.getElementsByName( 'boy');
console.log( typeof list );
```

- 通过class查找
- Var elems=parent.getElementsByClassName（‘class名’）

```
<div id="news">
        <p class="mainTitle">新闻标题1</p>
        <p class="subTitle">新闻标题2</p>
        <p class=" mainTitle ">新闻标题3</p>
</div>
```

```
var div = document.getElementById('news');
var list = div.getElementsByClassName('mainTitle');
console.log(list );
```

- 通过css选择器查找

（1）只找一个元素时var elem=parent.querySelector（‘selector’）

注意：selector支持一切css中选择器　　class类型的用.class

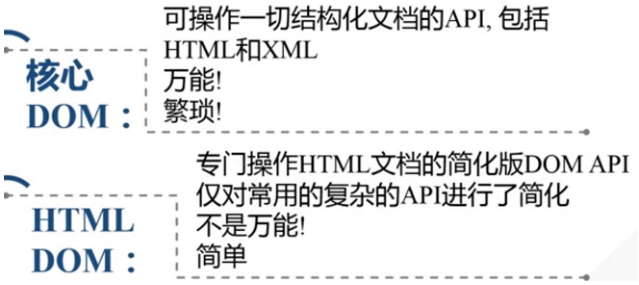
注意：如果选择器匹配有多个，只返回第一个　　id类型的用#id

（2）找多个元素时var elems=parent.querySelectorAll（‘selector’）

注意：selector API返回的是非动态集合

（2）DOM修改：

分为核心DOM和HTML DOM



开发过程中先用简单的再用复杂的，一切以实现效果为目标

- 核心DOM的4个操作：【value为属性值】
  1. 读取属性值：getAttribute{
     - 先获得属性节点对象，再获得节点对象的值：
       Var attrNode=elem.attributes[下标/属性名]
       Var attrNode=elem.getAttributeNode（属性名）
       attrNode.value——属性值
     - 直接获得属性值：
       Var value=elem.getAttribute（“属性名”）

     }
  2. 修改属性值：setAttribute{
     Elem.setAttribute（“属性名”，value）

     ```
     var h1 = document.getElementById(“a1");
     h1.setAttributeNode(“name” ,zhangji);
     ```

     先获取id名为a1的元素保存在h1中，再用h1调用，获取到名字，修改里面的内容

     }
  3. 判断是否包含指定属性：hasAttribute{
     返回的是布尔类型的值（真或假）

     ```
     var bool=elem.hasAttribute("属性名")
     ```

     例：

     Document.getElementById（‘bt1’）.hasAttribute（‘onclick’）

     先获取到id名为bt1的元素，判断其中是否包含onclick属性

     }
  4. 移除属性：removeAttribute{
     跟着什么属性名意思就是移除什么属性

     ```
     elem.removeAttribute("属性名")
     ```

     例：

     ```
     <a id=“alink" class="slink" href=
     "javascript:void(0)" onclick="jump()">百度搜索</a>
     ```

     ```
     var a = document.getElementById('alink');
     a.removeAttribute('class');
     ```

找到id名为alink的元素，调用这个方法移除这个元素中的class属性
}

- 修改样式：

  对于内联样式：elem.style.属性名　　找到该样式

  注意：属性名要去横线，变驼峰

  如：

  ```
  css: background-color => backgroundColor
       list-style-type => listStyleType
  ```

（3）DOM添加：

添加元素步骤：

1. 创建空元素{

   ```
   var elem=document.createElement("元素名")
   ```

   例：

   ```
   var table = document.createElement('table');
   var tr= document.createElement('tr');
   var td= document.createElement('td');
   var td= document.createElement('td');

   console.log( table );
   ```

   }

2. 设置关键属性{　是href打错字了

   ```
   a.innerHTML="go to tmooc"
   a.herf="http://tmooc.cn";
   <a href="http://tmooc.cn">go to tmooc</a>
   ```

   也可以设置样式【第二行是设置多个样式属性】

   opacity是透明度 的值等于1

   ```
   a.style.opacity = "1";
   a.style.cssText = "width: 100px;height: 100px";
   ```

   }

3. 将元素添加到DOM树{

   方法一、【只能在末尾添加】

   可以将父元素追加最后一个子节点

   ```
   parentNode.appendChild(childNode)
   ```

   例：

   ```
   var div = document.createElement( 'div' );
   var txt = document.createTextNode('版权声明');

   div.appendChild(txt);
   document.body.appendChild(div);
   ```

   首先创建两个元素并声明元素名，再将txt加入到div中，再将div加入到body中

   方法二、【可以自定义位置】

   用于在父元素中的指定子节点之前添加一个新的子节点

   ```
   parentNode.insertBefore(newChild, existingChild)
   ```

   例：

   ```
   <ul id="menu">
           <li>首页</li>
           <li>联系我们</li>
   </ul>
   var ul = document.getElementById('menu');
   var newLi = document.createElement('li');

   ul.insertBefore(newLi, ul.lastChild);
   ```

首先获取id名为menu的元素存放在ul中，创建一个li存放在newli中，再将nweli插入到父元素ul中最后一个孩子的前面

注意：尽量少操作DOM树，会导致重新layout【

1、如果同时创建父元素和子元素时，建议在内存中先将子元素添加到父元素中，再将父元素一次性挂到页面

2、如果只添加多个平级子元素时，就要将所有子元素，临时添加到文档片段中。再将文档片段整体添加到页面】

文档片段：内存中，临时保存多个平级子元素的虚拟父元素，用法和普通父元素完全一样

如何使用：【frag会自动释放】

```
1.创建片段
var  frag=document.createDocumentFragment();
2.将子元素临时追加到frag中
frag.appendChild(child);
3.将frag追加到页面
parent.appendChild(frag);
强调: append之后，frag自动释放，不会占用元素
}
```