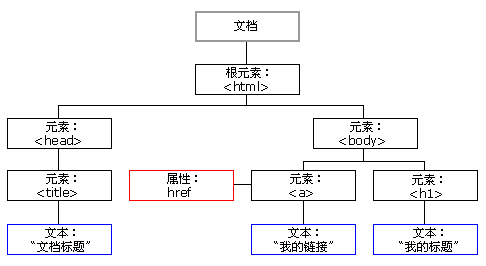
Dom基础知识点

# 一、什么是dom

DOM是JavaScript操作网页的接口，全称为“文档对象模型”（Document Object Model）。文档对象模型 (DOM) 是HTML和XML文档的编程接口。它给文档（结构树）提供了一个结构化的表述并且定义了一种方式—程序可以对结构树进行访问，以改变文档的结构，样式和内容。

浏览器会根据DOM模型，将结构化文档（比如HTML和XML）解析成一系列的节点，再由这些节点组成一个树状结构（DOM Tree）。所有的节点和最终的树状结构，都有规范的对外接口。所以，DOM可以理解成网页的编程接口。

DOM 提供了一种表述形式将文档作为一个结构化的节点组以及包含属性和方法的对象。从本质上说，它将web 页面和脚本或编程语言连接起来了。



Document对象

每个载入浏览器的HTML文档都会成为document对象。document对象包含了文档的基本信息，我们可以通过JavaScript对HTML页面中的所有元素进行访问、修改。

# 二、节点

DOM的最小组成单位叫做节点（node）。文档的树形结构（DOM树），就是由各种不同类型的节点组成。每个节点可以看作是文档树的一片叶子。

节点的类型有七种。

Document：整个文档树的顶层节点

DocumentType，声明节点 ：doctype标签（比如<!DOCTYPE html>）

Element，元素节点：网页的各种HTML标签（比如<body>、<a>等）

Attribute，属性节点：网页元素的属性（比如class="right"）

Text，文本节点：标签之间或标签包含的文本

Comment，注释节点：注释

DocumentFragment，文档节点：文档的片段

这七种节点都属于浏览器原生提供的节点对象的派生对象，具有一些共同的属性和方法。

节点树

一个文档的所有节点，按照所在的层级，可以抽象成一种树状结构。这种树状结构就是DOM。

最顶层的节点就是document节点，它代表了整个文档。文档里面最高一层的HTML标签，一般是<html>，它构成树结构的根节点（root node），其他HTML标签节点都是它的下级。

除了根节点以外，其他节点对于周围的节点都存在三种关系。

1.父节点关系（parentNode）：直接的那个上级节点

2.子节点关系（childNodes）：直接的下级节点

3.同级节点关系（sibling）：拥有同一个父节点的节点

DOM提供操作接口，用来获取三种关系的节点。其中，子节点接口包括firstChild（第一个子节点）和lastChild（最后一个子节点）等属性，同级节点接口包括nextSibling（紧邻在后的那个同级节点）和previousSibling（紧邻在前的那个同级节点）属性。

# 三、获取元素

通过以下六种方式来**获取**指定元素：

**• getElementById()**

**• getElementsByTagName()**

**• getElementsByClassName()**

**• querySelector()和querySelectorAll()**

**• getElementsByName()——仅限表单元素**

**• document.title和document.body**

# 四、创建元素

创建动态元素的三种方式：

第一种：

Document.write():

<body>

<input type="button" id="btn" value="生成元素" onclick="f1()"/>

<input type="text" value=""/>

<script>

        document.write("这是新添加的内容");

</script>

<div id="d">

    <script>

        document.write("<input type='text' value='123'/>");

    </script>

</div>

<script>

    function f1(){

        document.write("<input type='text' value='456'/>");

    }

    f1();

</script>

第二种：

使用innerHTML创建元素：

<body>

<div id="d"></div>

<input type="button" value="生成" id="btn"/>

<script>

    var txt=document.getElementById("txt");

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

    var btn=document.getElementById("btn");

    //此方法字符串具有不可变性，频繁的操作字符串会造成内存的浪费。

//    btn.onclick=function(){

//        for(var i=0;i<10;i++){

//            d.innerHTML+="<input type='text'><br />";

//        }

//    }

    var arr=[];

    btn.onclick=function(){

        for(var i=0;i<10;i++){

            arr.push("<input type='text'><br />");

        }

        d.innerHTML=arr.join("");

    }

</script>

</body>

//重写该元素内全部heml代码，之前的都会被覆盖

第三种：

CreateElement():

<body>

<div id="d"></div>

<script>

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

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

    d.appendChild(ul);

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

    ul.appendChild(li);

    li.innerHTML ="1231321";

    li.id = "abc";

    ul.removeChild(li);

</script>

</body>

# 五、插入元素

<body>

    <ul id="list">

        <li>HTML</li>

        <li>CSS</li>

    </ul>

    <input id="txt" type="text"/><input id="btn" type="button" value="插入"/>

</body>

1. appendChild()——将一个新元素插入到父元素的内部子元素的末尾
2. appendChild()——将一个新元素插入到父元素的内部子元素的末尾