**DOM和BOM**

**一：DOM**

**1.1. DOM概述**

**全称Document Object Model，即文档对象模型。**

**DOM是W3C（万维网联盟）的标准，是中立于平台和语言的接口，它允**

**程序和脚本动态的访问和更新文档的内容，结构和样式。**

**作用：用于通过语言或脚本动态地访问和跟新文档的内容，结构和样式。**

**1.1.1 W3CDOM标准被分为3个不同的部分。**

**- 核心DOM - 操作所有结构化文档（xml,html）的通用API**

**-XML DOM - 针对XML文档的标准模型**

**-HTML DOM - 针对HTML文档的简化API**

**-当网页被加载时，浏览器会创建页面的文档对象模型。**

**1.1.2 通过DOM，可以访问所有的HTML元素，连同它们所包含的文本和属性！！!**

**1.2 DHTML：动态网页技术的统称**

**DHTML=HTML+CSS+JS**

**！！! HTML XHTML DHTML XML 区别：（笔试常考）**

**HTML：超文本标记语言，专门编写网页内容的语言。（hyper text Markup language）**

**XHTML:（不是语言）严格的HTML标准**

**DHTML：动态网页技术的统称，=HTML+JS+CSS**

**XML ： 可扩展的标记语言（就是可以自定义标签）**

**专门用来存储/传输自描述的结构化数据**

**逐渐被json替代了，很少有人用XML做传输了，做存储还有。**

**1.3 BOM VS DOM**

**BOM : 浏览器对象模型（API），专门操作浏览器窗口的API**

**没有标准。**

**DOM ：文档对象模型（模型即是一群现成的API），专门操作网页内容的API，可以对网页中任意对象做任何修改。DOM 是标准，90%以上的浏览器都严格兼容。**

**1.3.1 HTML DOM：**

**网页中一切都是对象（元素，属性，文字）**

**同一网页中的所有对象，在内存中父子相连，形成一棵DOM树**

**Document 对象是一棵文档树的**

**1.4 DOM树（重点）**

**1.4.1 什么是节点树：**

**. 树根为document对象**

**HTML DOM 将HTML文档视作树结构。**

**文档中的每个元素，属性，文字，注释都被看作一个节点对象---Node ---是所有节点对象的父类型（祖先）。**

**当网页被加载进内存时，浏览器会为网页创建一个document对象，所有节点对象都是document对象的子节点。**

**Document,封装了对网页中的所有子节点的增加，删除，查找**

**.通过document对象，来操作整个节点树。**

**.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**

**- 整个文档是一个文档节点（document）！！！**

**- 每个html标签是一个元素节点(element) ！！！**

**注：Element 是所有元素节点的祖先，比如div, h1,...都继承element**

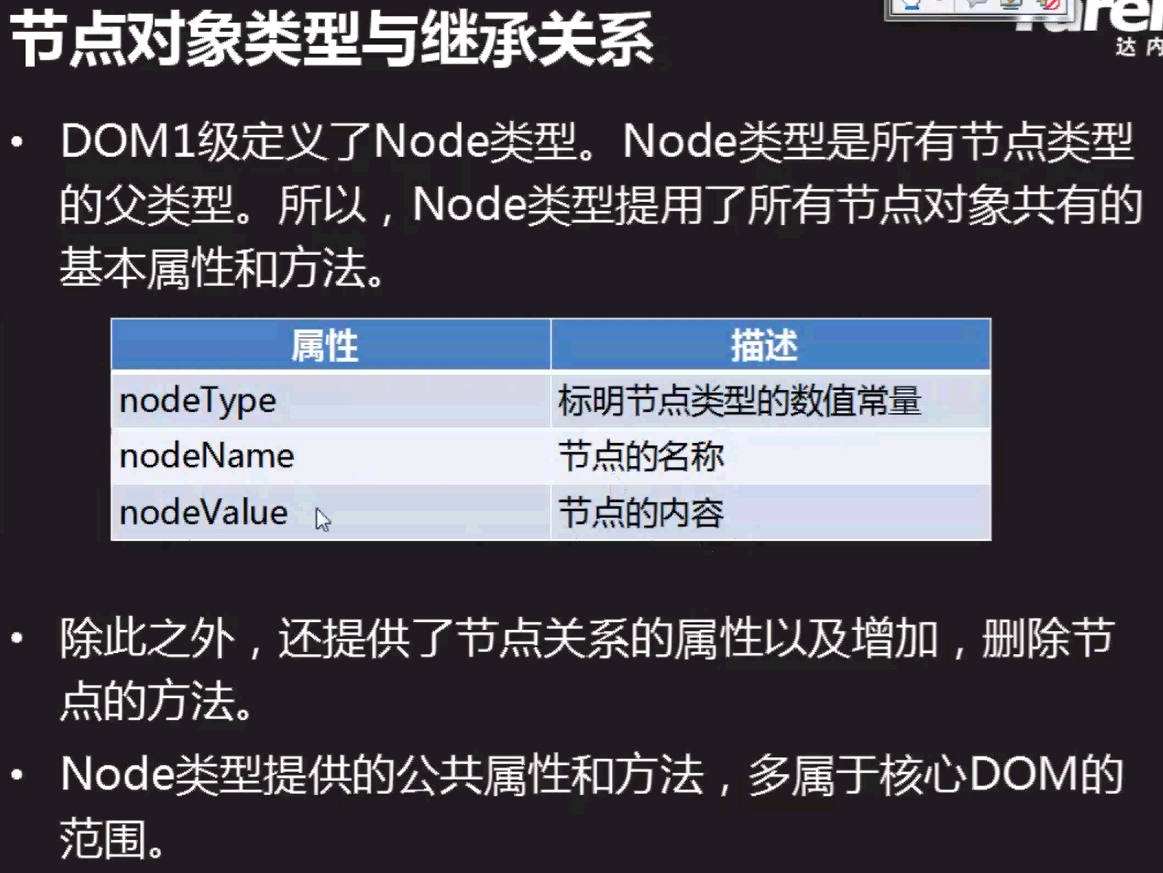
**- 包含在html元素中的文本是文本节点（text）！！！**

**(后两个可以忽略)**

**- 每一个html属性是一个属性节点(attribute)**

**- 注释属于注释节点（comment node）**

**1.4.2**

**Node类型定义了3个公共的属性：**

**（1）nodeType: 节点的类型的数值**

**何时使用 ：专门用于判断获得的节点类型。**

**如果是元素节点，返回1**

**如果是文本节点，返回3**

**（2）nodeName： 节点的名称**

**何时使用：专门用于判断获得的标签名。**

**如果是元素节点，返回标签名**

**如果是文本节点，返回#text**

**\*\*\*强调：nodeName 返回的都是全大写标签名（前提得是是元素节点）**

1. **（次要）nodeValue ： 节点的值**

**元素节点，返回null**

**文本节点，返回文本的内容**

**1.5 childNodes:获得当前节点对象下的所有子节点**

**返回的是类数组对象，可以用[i]和 .length访问**

**1.4. 3 DOM操作**

**通过可编辑的对象模型，javascript获得了足够的能力来创建动态的html**

**- 查找节点**

**- 读取节点信息**

**- 修改节点信息**

**- 创建新节点**

**- 删除节点**

**1.4.4 查找 (！！！重点） 5种**

**1. var elems=document.getElementById(“id值”)**

**2.var elems=document=parent.getElementsByTagName(“标签名”)**

**3.var elems=document.getElementByName(“name属性”)**

**4.var elems=parent.getElementsByClassName(“class属性”);**

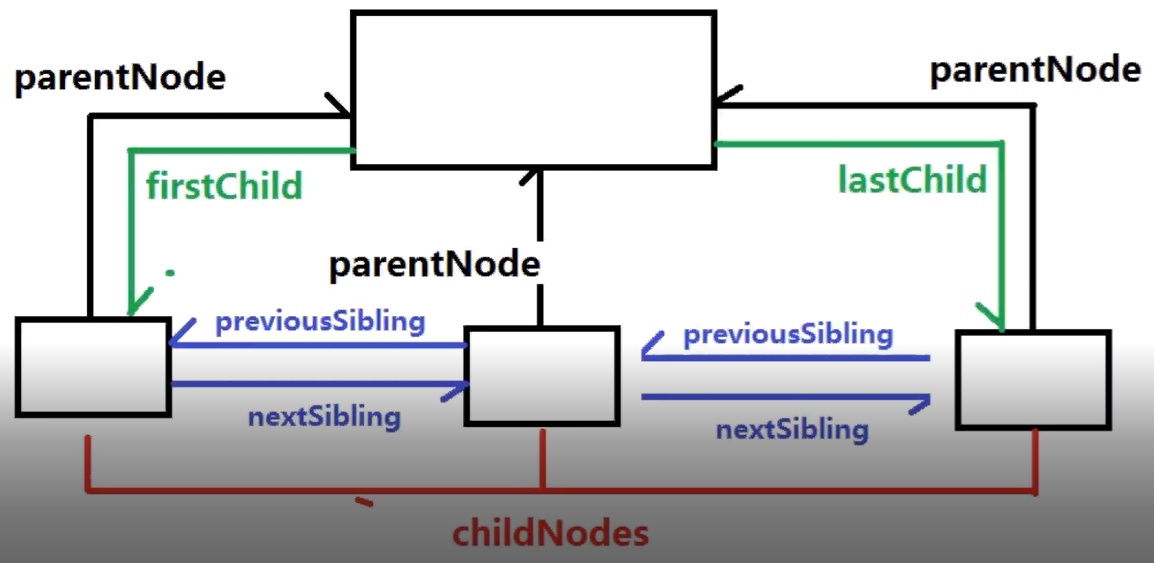
1. **Select API :jQuery的核心**

**Var elem=parent.querySelector(“任意选择器”);**

**Var elems=parent.querySelectorAll(“任意选择器”);**

**Elem:包含所有属性和方法的完整对象的集合**

**1.4.5节点间的关系：**

****

**节点数中：节点间关系：6个属性**

**父子关系4种：parentNode childNodes firstChild lastChlid**

**兄弟关系2种：previewsSibling nextSibling**

**chlidNodes : 类数组对象，**

**\*\*\*动态集合\*\*\*：自己不保存任何实际数据，每使用一次都重新查找一遍。**

**正课**

1. **遍历**
2. **查找**
3. **\*\*\*遍历：**

**从指定元素开始，按照深度优先关系的原则，遍历所拥有的各级子节点。**

**2步**

**(1)定义一个函数，查找找任意父节点下的所有直接子节点**

**(2)以深度优先为原则，递归调用函数本身。**

**何时使用递归调用：2个场景：**

1. **遍历不确定层级深度的树形结构时：**

**eg网页中的元素，网盘的文件夹结构**

1. **\*不确定层级深度的多级管理结构：**

**元素树：仅有元素节点组成的树结构**

**其实有一组和节点树6个属性对应的元素树属性**

**节点树 元素树**

**父对象 parentNode parentElementNode**

**所有子对象 childNodes chlidren**

**第一个子对象 firstChild firstElementChild**

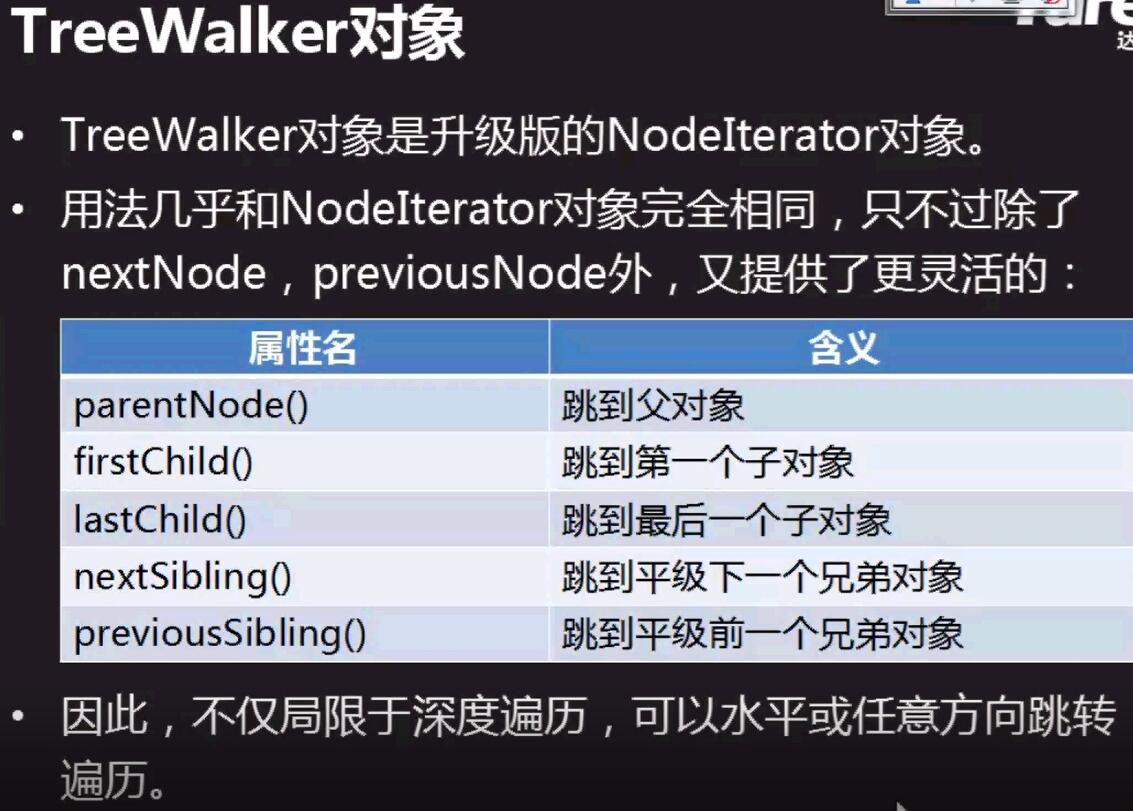
**最后子对象 lastChild lastElementChild**

**前一个兄弟 previousSibling previouseElementSibling**

**后一个兄弟 nextSibling nextElementSibling**

**何时使用：只要仅希望遍历元素树节点时，就用元素树**

**问题：IE8不兼容，children可用**

****

**遍历：有4种**

**节点树 元素树**

**API（NodeIterator ,TreeWalker）**

**查找：5个API**

1. **按id查找：**

**Var elem = document.getElementById(“id值”)**

1. **按标签名查找：（向下爬树的主要手段）**

**Var elem =document.getElementsByTagName(“标签名”);**

**\*\*\*elems也是动态集合\*\*\***

1. **按name属性查找：（专门用查找表单中的元素）**

**Var elems=parent.getElementByName(“name属性值”)**

**\*\*\*elems也是动态集合\*\*\***

**二：BOM**

**全称：Browser Object Model 浏览器对象模型**

**JavaScript是由浏览器中内置的javascript脚本解释器程序来执行javascript脚本语言的。**

**为了便于对浏览器的操作，javascript封装了对浏览器的各个对象使得开发者可以方便的操作浏览器。**

## 1.2. BOM对象：

## **1.3. window 对象**

**Window 对象是 JavaScript 层级中的顶层对象。**

**Window 对象代表一个浏览器窗口或一个框架。**

**Window 对象会在 <body> 或 <frameset> 每次出现时被自动创建。**

### 1.3.1.  window中的方法

**document   对 Document 对象的只读引用**

**location   用于窗口或框架的 Location 对象**

**history    对 History 对象的只读引用。**

**document.tilte    设置网页的标题**

**[moveto](http://blog.csdn.net/yuexianchang/article/details/53394054" \t "http://blog.csdn.net/yuexianchang/article/details/_blank)()  将窗口左上角的屏幕位置移动到指定的 x 和 y 位置。**

**moveby()   相对于目前的位置移动。**

**resizeTo()   调整当前浏览器的窗口。**

    open()     打开新窗口显示指定的URL（有的浏览器中是打一个新的选项卡）

    setTimeout(vCode, iMilliSeconds)       超时后执行代码。

setInterval(vCode, iMilliSeconds)      定时执行代码，第一次也是先待，到时再执行。

# 2.     事件

## a) 事件说明

    基本上所有的HTML元素中都可以指定事件属性。

    每个元素支持什么样事件应查询文档。

    所有的事件属性都是以on开头，后面的是事件的触发方式，如：

       onclick，表示单击

       onkeydown，表示键按下

              ...

## b) 常用的事件类型：

    鼠标点击相关：

       onclick 在用户用鼠标左键单击对象时触发。

       ondblclick 当用户双击对象时触发。

       onmousedown 当用户用任何鼠标按钮单击对象时触发。

       onmouseup 当用户在鼠标位于对象之上时释放鼠标按钮时触发。

    鼠标移动相关：

       onmouseout  当用户将鼠标指针移出对象边界时触发。

       onmousemove 当用户将鼠标划过对象时触发。

    焦点相关的：

       onblur 在对象失去输入焦点时触发。

       onfocus 当对象获得焦点时触发。

    其他：

       onchange 当对象或选中区的内容改变时触发。

       onload 在浏览器完成对象的装载后立即触发。

       onsubmit 当表单将要被提交时触发。

### location 对象

Location 对象是由 JavaScript runtime engine 自动创建的，包含有关当前 URL 的信息。

location中的重要方法：

    href属性   设置或获取整个 URL 为字符串。

    reload()   重新装入当前页面

### 1.3.2.  screen 对象

       Screen 对象是由 JavaScript runtime engine 自动创建的，包含有关客户机显示屏幕的信息。

属性：

    availHeight   获取系统屏幕的工作区域高度，排除 Microsoft Windows 任务栏。

    availWidth 获取系统屏幕的工作区域宽度，排除 Windows 任务栏。

    height     获取屏幕的垂直分辨率。

    width      获取屏幕的水平分辨率。

示例：

    document.write("屏幕工作区: " + screen.availHeight + "," + screen.availWidth + "<br>");

    document.write("屏幕分辨率: " + screen.height + ", " +screen.width + "<br>");

### 1.3.3.  document对象

    该对象代表整个文档页面

对象的集合：

    all    获取页面所有元素对象

    forms  获取页面所有表单对象

    images 获取页面所有图片对象

    links  获取所有超链接或area对象

具体的内容会在学习DOM时学习。

# 2.     DOM

## 2.1. DOM简介

**全称Document Object Model，即文档对象模型。**

**DOM描绘了一个层次化的树，允许开发人员添加、删除、修改页面的某一部分。**

**浏览器在解析HTML页面标记的时候，其实不是按照一行一行读取并解析的， 而是将HTML页面中的每一个标记按照顺序在内存中组建一颗DOM树，组建好之后，按照树的结构将页面显示在浏览器的窗口中。**

## 2.2. 节点层次

**HTML网页是可以看做是一个树状的结构，如下：**

    html

     |-- head

     |    |-- title

     |    |-- meta

     |    ...

     |--body

     |    |-- div

     |    |-- form

     |    |     |-- input

     |    |     |-- textarea

     ...  ...   ...

**这是一颗树，是由多个节点（Node）组成的，节点的类型有很多种。**

**节点最多有一个父节点，可以有多个子节点。**

**HTML DOM 定义了访问和操作HTML文档的标准方法。**

    document

       代表当前页面的整个文档树。

    访问属性

       all

       forms

       images

       links

       body

    访问方法(最常用的DOM方法)

|  |
| --- |
| <html xmlns="http://www.w3.org/1999/xhtml">  <head>  <script type="text/javascript">  // 获取dom 树, 获取document 对象.  **var** dom = window.document;  // all 获取页面中所有的标签节点 ,注释和文档类型约束.  **function** testAll() {  **var** allArr = dom.all;      alert(allArr.length);  **for** (**var** i = 0; i < allArr.length; i++) {         //获取节点名称         alert(allArr[i].nodeName);      }  }  // anchors 获取页面中的所有的锚连接.  **function** testAnchors() {  **var** anArr = dom.anchors;      alert(anArr.length);  }  // froms  获取所有的form 表单对象  **function** testForms() {  **var** formArr = dom.forms;      alert(formArr.length);      alert(formArr[0].nodeName);  }  // images  **function** testImages() {  **var** imageArr = dom.images;      alert(imageArr.length);  }  // links  获取页面的超链接.  **function** testLinks() {  **var** linkArr = dom.links;      //alert(linkArr.length);  **for** (**var** i = 0; i < linkArr.length; i++) {         //alert(linkArr[i].nodeName);      }  **for** (**var** i **in** linkArr) {         alert(i);      }  }  //testLinks();  // 获取页面的Body  **var** body = dom.body;  alert(body.nodeName);  </script>  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />  <title>javascript</title>  </head>  <body onmousemove="test(this)">      <img src="xxx" alt="这是一个美女"/>         <img src="xxx" alt="这是一个美女"/>         <img src="xxx" alt="这是一个美女"/>         <a href="http://www.baidu.com">百度一下</a>         <a href="http://www.google.com">百度两下</a>         <a href="http://www.baigu.com">百谷一下</a>         <a name="one"></a>         <a name="two"></a>         <form>             <label>姓名:</label><!--默认不写type 就是文本输入框-->             <input  type="text"/>         </form>  </body>  </html> |

## 2.3. 获取节点对象案例

    document.getElementById("html元素的id")

    document.getElementsByTagName("标签名")

    document.getElementsByName("html元素的name")

    示例：

       1，得到所有的div元素并显示信息（innerHTML）。

       2，得到所有div元素中id为"test"的结果。

|  |
| --- |
| **Var** dom = window.document;  **function** testByTagName() {  **var** iptArr = dom.getElementsByTagName("input");  **for** (**var** i = 0; i < iptArr.length; i++) {         alert(iptArr[i].value);      }  }  // window 对象提供了一个事件, onload 事件 onload(页面加载完毕执行该代码) 是一个事件, 给事件一个方法,  //window.onload = testByTagName;  //2，得到所有标签id为"username"的结果。获取旧value值并设置value值  **function** testById() {  **var** user = dom.getElementById("username");      alert(user.value);      user.value = "rose";  }  //testById();  //3. 获取所有标签name 为like的元素.获取value值.  **function** testByName() {  **var** likeArr = dom.getElementsByName("like");  **for** (**var** i = 0; i < likeArr.length; i++) {         alert(likeArr[i].value);      }  }  testByName(); |

### 2.3.1.  案例

显示页面任意一个节点对象的所有子节点的nodeName、nodeType、nodeValue

### 2.3.2.  属性操作练习

    1，写一个form，其中有多个checkbox。

    2，获取所有选中的数量。

    3，实现全选与全不选的效果。

|  |
| --- |
| <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">  <html xmlns="http://www.w3.org/1999/xhtml">  <head>  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />  <title>无标题文档</title>    <script type="text/javascript">  **function** getSum()  {      /\*      需求：通过点击总金额按钮获取被选中的条目的金额的总和，并将总金额显示在按钮右边。      思路：      1，先获取所有的checkbox对象。      2，对这些对象进行遍历。判断哪个对象被选中。      3，对被选中对象的金额进行累加。      4，显示在按钮右边。      \*/    **var** items = document.getElementsByName("item");  **var** sum = 0;  **for**(**var** x=0; x<items.length; x++)      {  **if**(items[x].checked)         {             sum += parseInt(items[x].value);         }      }  **var** str = sum+"元";      document.getElementById("sumid").innerHTML = str.fontcolor('red');  }    **function** checkAll(node)  {      /\*      需求：通过全选checkbox，将其他条目都选中。      思路：      只要将全选checkbox的checked状态赋值给其他的item checked状态即可。          \*/      //var allNode = document.getElementsByName("all")[index];    **var** items = document.getElementsByName("item");  **for**(**var** x=0; x<items.length; x++)      {         items[x].checked = node.checked;      }  }  </script>  </head>    <body>      <div>商品列表</div>  <input type="checkbox" name="all" onclick="checkAll(this)"  /> 全选<br />  <input type="checkbox" name="item" value="3000" />笔记本电脑3000元<br />  <input type="checkbox" name="item" value="3000"  />笔记本电脑3000元<br />  <input type="checkbox" name="item" value="3000"  />笔记本电脑3000元<br />  <input type="checkbox" name="item" value="3000"  />笔记本电脑3000元<br />  <input type="checkbox" name="item" value="3000"  />笔记本电脑3000元<br />  <input type="checkbox" name="item" value="3000"  />笔记本电脑3000元<br />  <input type="checkbox" name="all" onclick="checkAll(this)"  /> 全选<br />  <input type="button" value="总金额：" onclick="getSum()" /><span id="sumid"></span>      </body>  </html> |

## 2.4. 通过节点关系查找节点

    从一个节点出发开始查找：

    parentNode 获取当前元素的父节点。

    childNodes 获取当前元素的所有下一级子元素。

    firstChild 获取当前节点的第一个子节点。

    lastChild  获取当前节点的最后一个子节点。

    nextSibling       获取当前节点的下一个节点。（兄节点）

    previousSibling   获取当前节点的上一个节点。（弟节点）

    示例1：

       firstChild属性最普遍的用法是访问某个元素的文本：

       var text=x.firstChild.nodeValue;

    示例2：

       parentNode 属性常被用来改变文档的结构。

       假设您希望从文档中删除带有 id 为 "maindiv" 的节点：

       var x=document.getElementById("maindiv");

       x.parentNode.removeChild(x);

### 2.4.1.  获取节点对象的信息

    每个节点都包含的信息的，这些属性是：

    nodeType   节点类型

    nodeName   节点名称

    nodeValue  节点值

    nodeType

    nodeType 属性可返回节点的类型。

       ---------------------------------

       元素类型节点类型

       ------------------

         元素     1      就是标签元素，例<div>..</div>

         文本     3      标签元素中的文本

         注释     8       表示为注释

    nodeName

       nodeName 属性含有某个节点的名称。

       --------------------------------

       元素节点的 nodeName 是标签名称

       属性节点的 nodeName 是属性名称

       文本节点的 nodeName 永远是 #text

       文档节点的 nodeName 永远是 #document

    nodeValue

       对于文本节点，nodeValue 属性是所包含的文本。

       对于属性节点，nodeValue 属性是属性值。

       对于注释节点，nodeValue 属性注释内容。

       nodeValue 属性对于文档节点和元素节点是不可用的。

|  |
| --- |
| <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">  <html xmlns="http://www.w3.org/1999/xhtml">  <head>  <script type="text/javascript">  //节点和节点之间的关系.  //获取dom树  **var** dom = window.document;  //获取指定id 的标签节点.  **function** test() {  **var** form = dom.getElementById("form1");      //获取父节点.      //alert(form.parentNode.nodeName);      // 获取子节点(Node 包含文本,注释,标签)  **var** childArr = form.childNodes;      //alert(childArr.length);      /\*      for (var i = 0; i < childArr.length; i++) {      alert(childArr[i]);      }      \*/      // 获取第一个孩子.  **var** first = form.firstChild;      //alert(first);      //最后一个孩子.  **var** last = form.lastChild;      //alert(last);      // 获取下兄弟(获取弟弟)  **var** sibling = form.nextSibling;      //alert(sibling.nodeName);      // 获取大哥  **var** previous = form.previousSibling;      alert(previous.nodeName);  }  test();  </script>  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />    <title>javascript</title>  </head>  <body onmousemove="test(this)">         <a>哈哈</a>         <form id="form1">             <label>姓名:</label>             <input type="text" />         </form>  </body>  </html> |

## 2.5. 节点操作

### 2.5.1.  创建新节点

    document.createElement("标签名")    创建新元素节点

elt.setAttribute("属性名", "属性值") 设置属性

    elt.appendChild(e)                     添加元素到elt中最后的位置

    elt.insertBefore(new, child);          添加到elt中，child之前。

                                           // 参数1：要插入的节点  参数

2：插入目标元素的位置

    elt.removeChild(eChild)                删除指定的子节点

|  |
| --- |
| <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">  <html xmlns="http://www.w3.org/1999/xhtml">  <head>  <script>  /\*  创建节点：  document.createElement("标签名")    创建新元素节点  elt.setAttribute("属性名", "属性值") 设置属性  添加节点到文档树上:  elt.appendChild(e)                     添加元素到elt中最后的位置  把元素添加最后一个子节点的后面。  elt.insertBefore(new, child);          添加到elt中，child之前。                              // 参数1：要插入的节点  参数2：插入目标元素的位置    \*/  /\*  function add(){      //      var inputNode = document.createElement("input"); // 创建一个节点的对象      inputNode.setAttribute("type","file"); //给新节点设置type的属性值。      var body = document.getElementsByTagName("body")[0];      body.appendChild(inputNode);    //把新节点添加到body体中。  }  \*/  var count = 1;  function add(){      var trNode  = document.createElement("tr");      var tdNode  = document.createElement("td");      var inputNode  = document.createElement("input");      inputNode.setAttribute("type","button");      inputNode.setAttribute("value",count+"");      count++;      tdNode.appendChild(inputNode);      trNode.appendChild(tdNode);      //trNode添加 到指定 的位置上。      var tbodyNode = document.getElementsByTagName("tbody")[0];      //tableNode.appendChild(trNode);      var button1 = document.getElementById("b1");      tbodyNode.insertBefore(trNode,button1); // 注意： 使用obj.insertBefore(o1,o2)这个方法的时候      //obj必须是o1,o2的直接父节点。      //alert(button1.nodeName+"~~"+trNode.nodeName+"~~"+tableNode.nodeName);  }  </script>  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />  <title>无标题文档</title>  </head>  <body>      <table>          <tr>              <td>                  <input type="button" value="0">              </td>          </tr>          <tr id="b1">              <td>                  <input type="button" value="添加" onclick="add()">              </td>          </tr>      </table> |

|  |
| --- |
| <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">  <html xmlns="http://www.w3.org/1999/xhtml">  <head>  <script>      function addFile(){         var trNode = document.createElement("tr");         var td1  = document.createElement("td");         var td2  = document.createElement("td");         td1.innerHTML="<input type='file'/>";         td2.innerHTML="<a href='#' onclick='deleteFile(this)'>删除附件</a>"         trNode.appendChild(td1);         trNode.appendChild(td2);         //把trNode添加到添加按钮上面         var addButton = document.getElementById("addButton");         var tbody = document.getElementsByTagName("tbody")[0];         tbody.insertBefore(trNode,addButton);      }          function deleteFile(deleteNode){         //找到要删除的tr  a---->td---->tr         var trNode  = deleteNode.parentNode.parentNode;  //获取到了要删除的tr节点。         // 找到trNode的父节点         var tbodyNode  =document.getElementsByTagName("tbody")[0];         tbodyNode.removeChild(trNode);         //trNode.removeNode(true); // removeNode() 在firefox上不支持，在ie支持。      }      </script>  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />  <title>无标题文档</title>  </head>  <body>   <table>      <tr>          <td><input type="file"></td>              <td><a  href="#" onclick="deleteFile(this)">删除附件</a></td>          </tr>         <tr id="addButton">          <td>                <input type="button" value="添加附件"  onclick="addFile()"/>              </td>          </tr>      </table>    </body>  </html> |

### 2.5.2.  案例

1.      生成二级城市联动菜单

|  |
| --- |
| <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">  <html xmlns="http://www.w3.org/1999/xhtml">  <head>  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />  <title>无标题文档</title>  <style type="text/css">  select{      width:100px;      }  </style>  <script type="text/javascript">  **function** selcity()      {         //定义数据对应关系         //城市有很多，所以通过数组存储。每一个省对应一个城市数组，怎么建立对应关系呢？         //每一个省都有自己的角标。通过角标和数组建立对应关系，这就是二维数组。    **var** arr = [['--选择城市--'],['海淀区','朝阳区','东城区','西城区']                    ,['沈阳','大连','鞍山','抚顺']                    ,['济南','青岛','烟台','威海']                    ,['石家庄','廊坊','唐山','秦皇岛']];           //获取选择的省的角标。  **var** selNode = document.getElementById("selid");  **var** index = selNode.selectedIndex;    **var** cities = arr[index];    **var** subSelNode = document.getElementById("subselid");           //有更简单清除方式，只要改变下拉菜单的长度即可。         subSelNode.options.length = 0;         /\*         //清除上一次选择的子菜单内容。         for(var x=1; x<subSelNode.options.length;)         {               alert(subSelNode.options.length+"..."+subSelNode.options[x].innerHTML+"..."+x);             subSelNode.removeChild(subSelNode.options[x]);         }         \*/        **for**(**var** x=0; x<cities.length; x++)         {  **var** optNode = document.createElement("option");               optNode.innerHTML = cities[x];               subselid.appendChild(optNode);         }      }  </script>  </head>  <body>  <select id="selid" onchange="selcity()">      <option>--选择省市--</option>      <option>北京市</option>      <option>辽宁省</option>      <option>山东省</option>      <option>河北省</option>  </select>  <select id="subselid">      <option>--选择城市--</option>  </select>  </body>  </html> |

2.动态生成年、月、日字段

|  |
| --- |
| <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">  <html xmlns="http://www.w3.org/1999/xhtml">  <head>  <script type="text/javascript">  /\*\*   \* @author Administrator   \*/  //通过js创建年,月,日  //获取Dom  **var** dom = window.document;  **function** myYear() {      //获取年的select  **var** year = dom.getElementById("year");      //创建年  **var** minYear = 1900;  **var** maxYear = **new** Date().getFullYear();  **for** (**var** i = minYear; i <= maxYear; i++) {         //创建Option  **var** opt = dom.createElement("option");         //设置Option,标签体.         opt.innerHTML = i;         opt.value = i;         //挂载节点         year.appendChild(opt);      }  }  **function** myMonth() {  **var** month = dom.getElementById("month");      //创建月  **for** (**var** i = 1; i <= 12; i++) {         //创建Option  **var** opt = dom.createElement("option");         //设置Option,标签体.  **if** (i < 10) {             opt.innerHTML = "0" + i;             opt.value = i;         } **else** {             opt.innerHTML = i;             opt.value = i;         }         month.appendChild(opt);      }      month.onchange = myDay;  }  **function** myDay() {      clear();      //创建天      // 大月1 3 5 7 8 10 12 ,小月4 6 9 11    闰年2月非闰年的2月      //获取年  **var** year = dom.getElementById("year").value;      //获取月  **var** month = dom.getElementById("month").value;  **if** (year == "") {         alert("请选择年");  **return**;      }  **if** (month == "") {         alert("请选择月");  **return**;      }      //获取天select  **var** day = dom.getElementById("day");      //一个月至少有28天.  **for** (**var** i = 1; i <= 28; i++) {  **var** opt = dom.createElement("option");  **if** (i < 10) {             opt.innerHTML = "0" + i;             opt.value = "0" + i;         } **else** {             opt.innerHTML = i;             opt.value = i;         }         day.appendChild(opt);      }      //大月  **var** isBigMonth = month == 1 || month == 3 || month == 5 || month == 7 || month == 8 || month == 10 || month == 12;      //小月  **var** isSmallMonth = month == 4 || month == 6 || month == 9 || month == 11;      //闰年    可以整除4但不能整除100 或者年份可以整除400.  **var** isLeapYear = (year % 4 == 0 && year % 100 != 0) || year % 400 == 0;      //判断,如果是大月,添加3天  **if** (isBigMonth) {         //添加3天  **for** (**var** i = 29; i <= 31; i++) {  **var** opt = dom.createElement("option");             opt.innerHTML = i;             opt.value = i;             day.appendChild(opt);         }      } **else if** (isSmallMonth) {         //添加2天  **for** (**var** i = 29; i <= 30; i++) {  **var** opt = dom.createElement("option");             opt.innerHTML = i;             opt.value = i;             day.appendChild(opt);         }      } **else if** (isLeapYear) {         //如果是闰年,添加一天.专门处理闰年2月.  **var** opt = dom.createElement("option");         opt.innerHTML = 29;         opt.value = 29;         day.appendChild(opt);      }  }  **function** clear() {  **var** day = dom.getElementById("day");  **var** optArr = day.childNodes;  **for** (**var** i = optArr.length - 1; i >= 0; i--) {         day.removeChild(optArr[i]);      }  }  **function** getBirthday() {      //获取Dom  **var** dom = window.document;      myYear();      myMonth();  }  getBirthday();  </script>  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />  <title>javascript</title>  </head>  <body>      生日:         <select id="year">             <option>年</option>         </select>         <select id="month">             <option>月</option>         </select>         <select id="day">             <option>日</option>         </select>    </body>  </html> |

3.  生产一个验证码

|  |
| --- |
| <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">  <html xmlns="http://www.w3.org/1999/xhtml">  <head>  <script type="text/javascript">  /\*\*   \* @author Administrator   \*/  //验证码  ,4位的,由字符,数字组成.  **function** createCode() {  **var** datas = ["A", "B", "C", "D", "E", "F", "G", "H", "Z", "0", "1", "2", "3", "4", "5", "6", "7", "8", "9"];      //随机的从数组中取出4个元素.  **var** mess = "";  **var** index = 0;  **for** (**var** i = 0; i < 4; i++) {         //生成随机数.而且是在数组的长度范围内.         //0-9之间的随机数. Math.floor(Math.random()\*10)         //0到数组长度(不包含)之间的浮点数.,向下取整,  **var** index = Math.floor(Math.random() \* datas.length);         mess += datas[index];      };      //  **var** codeSpan = window.document.getElementById("codeSpan");      codeSpan.style.color = "red";      codeSpan.style.fontSize = "20px";      codeSpan.style.background = "gray";      codeSpan.style.fontWeight = "900";      codeSpan.style.fontStyle = "italic";      codeSpan.style.textDecoration = "line-through";      codeSpan.innerHTML = mess;      codeSpan.value = mess;  }  createCode();  </script>  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />  <title>javascript</title>  </head>  <body>      <span id="codeSpan"></span><a href="#" onclick="createCode()">看不清楚</a>    </body>  </html> |

# 3.     正则表达式

位置：

       ^      开头

       $      结尾

次数：

       \*      0或多个

       +      1或多个

       ?      0或1个

       {n}     就是n个

       {n,}   至少n个

       {n,m}  最少n个，最多m个

通配符：

       \d     任意数字

       \D     任意非数字

       \s     任意空白

       \S     任意非空白

       .      任意字符（除'\n'外）

组合：

       [a-z]

       [0-9]

       等

组：

       (正则)     匹配括号中正则表达式对应的结果，并暂存这个结果。

       (?:正则)   匹配括号中正则表达式对应的结果，但不暂存这个结果。

       \数字      使用第n个组匹配的结果

使用正则的工具（RegExp类与相关方法）

创建：

       // 方式一

       var regex = new RegExp("正则表达式", "标志");

       // 方式二

       var regex = /正则表达式/标志

参数说明：

正则表达式：

    参见上面的规则

标志：

    g （全文查找出现的所有 pattern）

    i （忽略大小写）

    m （多行查找）

方法：

       Regexp.test( str )

       String.replace( regex, str )

练习：校验表单

|  |
| --- |
| <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">  <html xmlns="http://www.w3.org/1999/xhtml">  <head>  <script>        //用户名的规则：第一位是字母，只有由数字与字母组成，6位。      function checkName(){         //获取到了用户名的值         var userName = document.getElementById("username").value;         var userSpan = document.getElementById("userId");         var reg = /^[a-z][a-z0-9]{5}$/i;         if(reg.test(userName)){             //符合规则             userSpan.innerHTML="正确".fontcolor("green");             return true;         }else{             //不符合规则             userSpan.innerHTML="错误".fontcolor("red");             return false;         }      }          //校验密码  6位      function checkPass(){         var  password  = document.getElementById("pwd").value;         if(password.length>0){             var reg = /^\w{6}$/;             var passSPan = document.getElementById("passId");             if(reg.test(password)){                //符合规则                passSPan.innerHTML="正确".fontcolor("green");                return true;             }else{                //不符合规则                passSPan.innerHTML="错误".fontcolor("red");                return false;             }         }        }          //检验密码是否正确      function ensurepass(){         var  password1 = document.getElementById("pwd").value; //第一次输入的密码         var password2 = document.getElementById("ensurepwd").value;         if(password2.length>0){             var enSpan  = document.getElementById("ensure");             if(password1.valueOf()==password2.valueOf()){                enSpan.innerHTML="正确".fontcolor("green");                return true;             }else{                enSpan.innerHTML="错误".fontcolor("red");                return false;             }         }      }          //校验邮箱      function checkEmail(){         var  email  = document.getElementById("email").value;         var reg = /^[a-z0-9]\w+@[a-z0-9]{2,3}(\.[a-z]{2,3}){1,2}$/i;  // .com .com.cn         var emailspan = document.getElementById("emailspan");         alert(reg.test(email));         if(reg.test(email)){             //符合规则             emailspan.innerHTML="正确".fontcolor("green");               return true;         }else{             //不符合规则             emailspan.innerHTML="错误".fontcolor("red");             return false;         }      }          //校验兴趣爱好：至少要算中其中的一个。      function checkHoby(){         var likes  = document.getElementsByName("like");         var hobySpan =document.getElementById("hobbySpan")         var flag  = false;         for(var i =  0 ; i<likes.length ; i++){             if(likes[i].checked){                flag =true;                break;             }         }           if(flag){             //符合规则             hobySpan.innerHTML="正确".fontcolor("green");             return true;         }else{             //不符合规则             hobySpan.innerHTML="错误".fontcolor("red");             return false;         }      }            //总体校验表单是否可以提交了  如果返回的true表单才可以提交。上面的表单项必须要每个都填写正确。      function checkForm(){         var userName = checkName();         var pass  = checkPass();         var ensure  = ensurepass();         var email = checkEmail();         var hoby = checkHoby();         if(userName&&pass&&ensure&&email&&hoby){             return true;         }else{             return false;         }        }                </script>  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />  <title>正则表达式</title>  </head>  <body>  <form action="success.html" method="get" onsubmit="return checkForm()"> <!--如果表单提交时候触发的方法返回是false,那么该表单不允许提交，如果返回的是true允许提交 -->             <table border="1px" width="50%" align="center" cellspacing="0px" cellpadding="3px">                <tr>                    <td width="25%">姓名:</td>                    <td>                       <input type="text" name="username" id="username" onblur="checkName()"/>                          <span id="userId"></span>                    </td>                </tr>                <tr>                    <td >密码:</td><td>                       <input type="password"  name="pwd" id="pwd" onblur="checkPass()"/>                          <span id="passId"></span>                    </td>                </tr>                <tr>                    <td>确认密码:</td><td>                <input type="password" name="ensurepwd" id="ensurepwd" onblur="ensurepass()" />                    <span id="ensure"></span>                    </td>                </tr>                <tr>                    <td>邮箱</td><td>                       <input type="text" name="email" id="email" onblur="checkEmail()"/>                        <span id="emailspan"></span>                      </td>                </tr>                <tr>                    <td>性别</td><td>                       <input type="radio" checked="ture" name="gender" id="male" value="male"/>                    男                        <input type="radio" name="gender" value="female"/>                      女</td>                </tr>                  <tr>                    <td>爱好:</td><td>                       <input type="checkbox" checked="checked"  name="like" />                    eat                       <input type="checkbox" name="like" />                    sleep                       <input type="checkbox" name="like"/>                    play                      <span id="hobbySpan"></span>                      </td>                </tr>                <tr>                    <td>城市</td><td>                    <select name="city" id="city">                       <option value=""> 请选择</option>                       <option value="bj"> 北京 </option>                       <option value="gz"> 广州 </option>                       <option value="sh"> 上海 </option>                    </select>                        </td>                </tr>                <tr>                    <td>自我介绍</td><td>                <textarea cols="15" rows="5"  name="myInfo" id="myInfo"></textarea></td>                </tr>                <tr align="center">                    <td colspan="2"><!--提交按钮-->                    <input type="submit"/>                    </td>                </tr>             </table>         </form>  </body>  </html> |