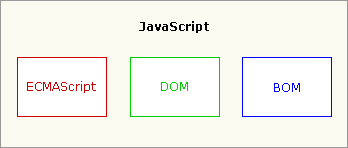
一、简介

1、JS定义：javaScript的简写形式就是JS，是由Netscape公司开发的一种脚本语言，一种广泛用于客户端Web开发的脚本语言，常用来给HTML网页添加动态功能（其编写的程序可以被嵌入到HTML或XML页面中，并直接在浏览器中解释执行）。

HTML标记(也称为结构)，知道了CSS样式(也称为表示)，JavaScript增加行为，为网页添加动态效果。

2、组成部分：核心（ECMAScript）、文档对象模型（Document Object Model，简称DOM）、浏览器对象模型（Browser Object Model，简称BOM）



3、JavaScript作用：

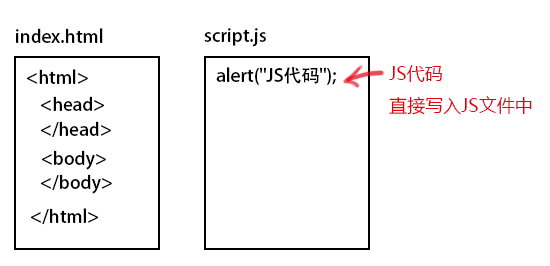
1. 增强页面动态效果(如:下拉菜单、图片轮播、信息滚动等)。
2. 实现页面与用户之间的实时、动态交互(如:用户注册、登陆验证等)。

4、使用JS

1. 插入JS：
2. 注意， <script>标签要成对出现，并把JavaScript代码写在<script></script>之间。
3. <script type="text/javascript"> -- 默认使用

表示在<script></script>之间的是文本类型(text),javascript是为了告诉浏览器里面的文本是属于JavaScript语言。

1. 引用JS外部文件.js



* 注意:在JS文件中，不需要<script>标签,直接编写JavaScript代码就可以了。
* JS文件不能直接运行，需嵌入到HTML文件中执行，我们需在HTML中添加如下代码，就可将JS文件嵌入HTML文件中。

<script src="script.js"></script>

5、JS语法格式

每一句JavaScript代码格式: 语句;

 <script type="text/javascript">

alert("hello!");

</script>

6、注释

 单行注释，在注释内容前加符号 “//”。

 多行注释以"/\*"开始，以"\*/"结束。

7、JS代码的位置

1. script标签如果不写type属性，浏览器会默认为text/javascript，如果写了不同的类型，浏览器就不会再按照解析JS的规则去解析了，发现是不识别的类型可能就会忽略它。
2. text/x-template中的内容不是给浏览器解析的，是用来在页面加载完成后再获取到它对其进行渲染。这个渲染的过程通常是使用各种模板引擎来完成的。
3. 可以将JavaScript代码放在html文件中任何位置，但是我们一般放在网页的head或者body部分。

* 放在<head>部分

最常用的方式是在页面中head部分放置<script>元素，浏览器解析head部分就会 执行这个代码，然后才解析页面的其余部分。

* 放在<body>部分

JavaScript代码在网页读取到该语句的时候就会执行。

1. 注意: javascript作为一种脚本语言可以放在html页面中任何位置，但是浏览器解释html时是按先后顺序的，所以前面的script就先被执行。比如进行页面显示初始化的js必须放在head里面，因为初始化都要求提前进行（如给页面body设置css等）；而如果是通过事件调用执行的function那么对位置没什么要求的。

二、JS基础

1、变量---JS中区分大小写

* 变量命名规则：

 ① 必须以字母、下划线或美元符号开头，后面可以跟字母、下划线、美元符号和数字。

 ② 变量名区分大小写，如:A与a是两个不同变量。

 ③ 不允许使用JavaScript关键字和保留字做变量名。

* 定义变量使用关键字var,语法如下：

 变量要先声明再赋值，如下：

var mychar;

mychar="javascript";

var mynum = 6;

 变量可以重复赋值，如下：

var mychar;

mychar="javascript";

mychar="hello";

2、函数

* 定义函数

基本语法如下:

function 函数名()

{

函数代码;

}

 function定义函数的关键字。

3、运算符优先级

* 操作符之间的优先级（高到低）:

算术操作符 → 比较操作符 → 逻辑操作符 → "="赋值符号

如果同级的运算是按从左到右次序进行,多层括号由里向外。

4、JS中的数据类型 – 总共有6种数据类型

1. String（字符串）：var carname="Bill Gates";
2. Number（数字）：var x1=34.00;
3. Boolean（布尔值）：var x=true
4. Array（数组）：var cars=new Array();cars[0]="Audi";cars[1]="BMW";cars[2]="Volvo";

* myarray.length; //获得数组myarray的长度

JavaScript数组的length属性是可变的，这一点需要特别注意。

可以直接赋值，mynum.length=10;

增加元素的时候也会改变数组的长度；

1. Undefined（表示变量不含有值）、Null（空值）
2. Object（name： lue）：对象由花括号分隔。在括号内部，对象的属性以名称和值对的形式 (name : value) 来定义。属性由逗号分隔：

var person={firstname:"Bill", lastname:"Gates", id:5566};

* 声明变量类型: 当您声明新变量时，可以使用关键词 "new" 来声明其类型

var carname=new String;

var x= new Number;

var y= new Boolean;

var cars= new Array;

var person= new Object;

三、JS事件

1、事件类型

* JavaScript 创建动态页面。事件是可以被 JavaScript 侦测到的行为。 网页中的每个元素都可以产生某些可以触发 JavaScript 函数或程序的事件。



2、鼠标单击事件—OnClick

* onclick是鼠标单击事件，当在网页上单击鼠标时，就会发生该事件。同时onclick事件调用的程序块就会被执行，通常与按钮一起使用。
* 语法：

<body>

<form>

<input name="点击我" type="button" value="点击我" onclick="openwin()"/>

</form>

</body>

3、鼠标经过事件—onmouseover

* 鼠标经过事件，当鼠标移到一个对象上时，该对象就触发onmouseover事件，并执行onmouseover事件调用的程序。
* 语法：

<input name="确定" type="button" value="确定" onmouseover="message()"/>

4、鼠标移开事件—onmouseout

* 鼠标移开事件，当鼠标移开当前对象时，执行onmouseout调用的程序。

语法：

<a href="http://www.imooc.com" onmouseout="message()">点击我</a>

5、光标聚焦事件—onfocus

* 当网页中的对象获得聚点时，执行onfocus调用的程序就会被执行。
* 语法：

onfocus=””;

6、失焦事件—onblur

* onblur事件与onfocus是相对事件，当光标离开当前获得聚焦对象的时候，触发onblur事件，同时执行被调用的程序
* 语法：

<input name="username" type="text" value="请输入用户名！" onblur="message()" >

7、内容选中事件—onselect

* 选中事件，当文本框或者文本域中的文字被选中时，触发onselect事件，同时调用的程序就会被执行。
* 语法：

<body>

<form>

个人简介：<br>

<textarea name="summary" cols="60" rows="5" onselect="message()">请写入个人简介，不少于200字！</textarea> </form></body>

8、文本框内容改变事件—onchange

* 通过改变文本框的内容来触发onchange事件，同时执行被调用的程序。
* 语法：

<body>

<form>

个人简介：<br>

<textarea name="summary" cols="60" rows="5" onchange="message()">请写入个人简介，不少于200字！</textarea>

</form>

</body>

9、加载事件—onload

* 事件会在页面加载完成后，立即发生，同时执行被调用的程序。

注意：

1. 加载页面时，触发onload事件，事件写在<body>标签内。
2. 此节的加载页面，可理解为打开一个新页面时。

如下代码,当加载一个新页面时，弹出对话框“加载中，请稍等…”。

语法：

<body onload="message()">

欢迎学习JavaScript。

</body>

10、卸载事件—退出页面—onunload

* 当用户退出页面时（页面关闭、页面刷新等），触发onUnload事件，同时执行被调用的程序。

注意：不同浏览器对onunload事件支持不同。

* 语法：

<script type="text/javascript">

window.onunload = onunload\_message;

function onunload\_message(){

alert("您确定离开该网页吗？");

}

</script>

四、JS内置对象

1、内置对象类型

1. String对象：处理所有的字符串操作
2. Math对象：处理所有的数学运算
3. Date对象：处理日期和时间的存储、转化和表达
4. Array对象：提供一个数组的模型、存储大量有序的数据
5. Event对象：提供JavaScript事件的各种处理信息
6. RegExp 对象：表示正则表达式，它是对字符串执行模式匹配的强大工具。

* 内置对象都有自己的方法和属性，访问的方法如下：

对象名.属性名称

对象名.方法名称(参数表)

2、日期对象

日期对象可以储存任意一个日期，并且可以精确到毫秒数（1/1000 秒）。

* 定义一个时间对象 :

var Udate=new Date();

注意:使用关键字new，Date()的首字母必须大写。

使 Udate 成为日期对象，并且已有初始值：当前时间(当前电脑系统时间)。

document.write(Udate);

输出：Fri Aug 11 2017 22:22:03 GMT+0800 (中国标准时间)

如果要自定义初始值，可以用以下方法：

var d = new Date(2012, 10, 1); //2012年10月1日

var d = new Date('Oct 1, 2012'); //2012年10月1日

* 调用对象方法：



getDay() 返回星期，返回的是0-6的数字，0 表示星期天。如果要返回相对应“星期”，通过数组完成。

语法： var mydate=new Date();

document.write(mydate.getDay());-------这是返回一个int数值

3、字符串对象

* 定义：var mystr = "I love JavaScript!";

访问属性：var le=mystr.length;

访问对象方法：var mystr="Hello world!";

var mynum=mystr.toUpperCase();

1. charAt() 方法可返回指定位置的字符。返回的字符是长度为 1 的字符串。

语法:

stringObject.charAt(index).

注意：

1.字符串中第一个字符的下标是 0。最后一个字符的下标为字符串长度减一（string.length-1）。

2.如果参数 index 不在 0 与 string.length-1 之间，该方法将返回一个空字符串

1. indexOf() 方法可返回某个指定的字符串值在字符串中首次出现的位置。

语法：

stringObject.indexOf(substring, startpos);

说明：

1.该方法将从头到尾地检索字符串 stringObject，看它是否含有子串 substring。

2.可选参数，从stringObject的startpos位置开始查找substring，如果没有此参数将从stringObject的开始位置查找。

3.如果找到一个 substring，则返回 substring 的第一次出现的位置。stringObject 中的字符位置是从 0 开始的。

注意：

1.indexOf() 方法区分大小写。

2.如果要检索的字符串值没有出现，则该方法返回 -1。

1. split() 方法将字符串分割为字符串数组，并返回此数组。

语法：

stringObject.split(separator,limit);

说明：

1、separator,这是要分割的位置字符。如：stringObject.split(“-”);

2、limit,指要分割的次数。

注意：

如果把空字符串 ("") 用作 separator，那么 stringObject 中的每个字符之间都会被分割。

1. substring() 方法用于提取字符串中介于两个指定下标之间的字符。

语法:

stringObject.substring(startPos,stopPos);

注意：

1. 返回的内容是从 start开始(包含start位置的字符)到 stop-1 处的所有字符，其长度为 stop 减start。

2. 如果参数 start 与 stop 相等，那么该方法返回的就是一个空串（即长度为 0 的字符串）。

3. 如果 start 比 stop 大，那么该方法在提取子串之前会先交换这两个参数。

1. substr() 方法从字符串中提取从 startPos位置开始的指定数目的字符串。

语法:

stringObject.substr(startPos,length);

mystr.substr(0,5);

注意：如果参数startPos是负数，从字符串的尾部开始算起的位置。也就是说，-1 指字符串中最后一个字符，-2 指倒数第二个字符，以此类推。

如果startPos为负数且绝对值大于字符串长度，startPos为0。

4、Math对象

**注意**：Math 对象是一个固有的对象，无需创建它，直接把 Math 作为对象使用就可以调用其所有属性和方法。这是它与Date,String对象的区别。

* 属性和方法：





1. ceil() 方法可对一个数进行向上取整。

**语法**：document.write(Math.ceil(3.3));

**注意**：它返回的是大于或等于x，并且与x最接近的整数。

1. floor() 方法可对一个数进行向下取整。

**语法**：document.write(Math.floor(3.3));

**注意**：返回的是小于或等于x，并且与 x 最接近的整数。

1. round() 方法可把一个数字四舍五入为最接近的整数。

 **语法**：document.write(Math.round(8.9)+ "<br>");

 **注意**：

1. 返回与 x 最接近的整数。

2. 对于 0.5，该方法将进行上舍入。(5.5 将舍入为 6)

3. 如果 x 与两侧整数同等接近，则结果接近 +∞方向的数字值 。(如 -5.5 将舍入为 -5; -5.52 将舍入为 -6).

1. random() 方法可返回介于 0 ~ 1（大于或等于 0 但小于 1 )之间的一个随机数。

 **语法**：Math.random();

 **注意**：返回一个大于或等于 0 但小于 1 的符号为正的数字值。

5、数组对象—Array对象

* 定义数组：

定义了一个空数组:

var 数组名= new Array();

定义时指定有n个空元素的数组：

var 数组名 =new Array(n);

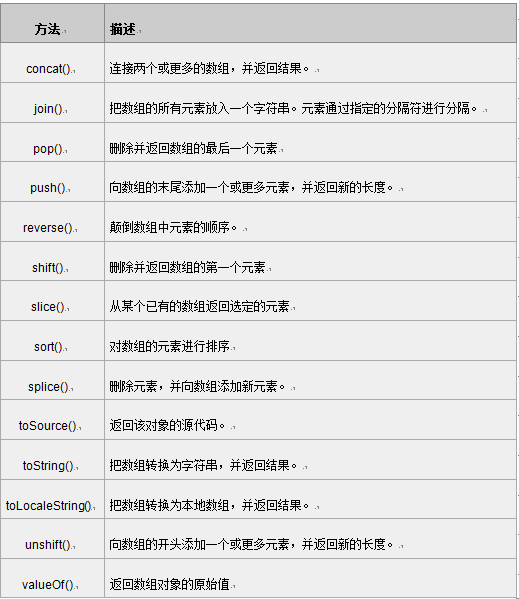
定义数组的时候，直接初始化数据：

var 数组名 = [<元素1>, <元素2>, <元素3>...];

* 属性：

length 用法：<数组对象>.length；返回：数组的长度，即数组里有多少个元素。它等于数组里最后一个元素的下标加一。

* 方法：



1. concat() 方法用于连接两个或多个数组。此方法返回一个新数组，不改变原来的数组。

语法：arrayObject.concat(array1,array2,...,arrayN);

注意：参数是数组名称。

1. join()方法用于把数组中的所有元素放入一个字符串。元素是通过指定的分隔符进行分隔的。

语法：arrayObject.join(分隔符)

document.write(myarr3.join("-"));//默认分隔字符为”,”.

注意：返回一个字符串，该字符串把数组中的各个元素串起来，用<分隔符>置于元素与元素之间。这个方法不影响数组原本的内容。

1. reverse() 方法用于颠倒数组中元素的顺序。

语法：

arrayObject.reverse(); document.write(myarr.reverse());

注意：该方法会改变原来的数组，而不会创建新的数组。

1. slice() 方法可从已有的数组中返回选定的元素。

语法：

arrayObject.slice(start,end);

document.write(myarr1.slice(1));

说明：

1.返回一个新的数组，包含从 start 到 end （不包括该元素）的 arrayObject 中的元素。

2. 该方法并不会修改数组，而是返回一个子数组。

注意：

1. 可使用负值从数组的尾部选取元素。

2.如果 end 未被规定，那么 slice() 方法会选取从 start 到数组结尾的所有元素。

3. String.slice() 与 Array.slice() 相似。

1. sort()方法使数组中的元素按照一定的顺序排列。

语法:

arrayObject.sort(方法函数);

document.write(myarr.sort(sortNum));

注意：

 function sortNum(a,b) {

return b-a;

}

 sort()函数使用的是冒泡排序，冒泡排序会重复地走访要排序的数列，一次比较两个元素，如果他们的顺序错误就把他们交换过来，一直重复地进行直到说该数列已经排序完成。

 如果a-b>0(即正数)就把a和b的位置交换，也就是较小的一个数会排到前面；

如果b-a>0就把a和b的位置交换，也就是较大的一个数会排到前面。

6、RegExp 对象

* 创建语法格式：、

new RegExp(pattern, attributes);

实例：var reg = new RegExp("[&|?]" + paraName + "=([^&$]\*)", "gi");

* 参数说明：
  1. 参数 pattern 是一个字符串，指定了正则表达式的模式或其他正则表达式。
  2. 参数 attributes 是一个可选的字符串，包含属性 "g"、"i" 和 "m"，分别用于指定全局匹配、区分大小写的匹配和多行匹配。ECMAScript 标准化之前，不支持 m 属性。如果 pattern 是正则表达式，而不是字符串，则必须省略该参数。
  3. i：执行对大小写不敏感的匹配。

g：执行全局匹配（查找所有匹配而非在找到第一个匹配后停止）。

m：执行多行匹配。

* 返回值：

一个新的 RegExp 对象，具有指定的模式和标志。如果参数 pattern 是正则表达式而不是字符串，那么 RegExp() 构造函数将用与指定的 RegExp 相同的模式和标志创建一个新的 RegExp 对象。

如果不用 new 运算符，而将 RegExp() 作为函数调用，那么它的行为与用 new 运算符调用时一样，只是当 pattern 是正则表达式时，它只返回 pattern，而不再创建一个新的 RegExp 对象。

* 抛出：

SyntaxError - 如果 pattern 不是合法的正则表达式，或 attributes 含有 "g"、"i" 和 "m" 之外的字符，抛出该异常。

TypeError - 如果 pattern 是 RegExp 对象，但没有省略 attributes 参数，抛出该异常。

五、浏览器对象 – BOM

1、Window对象

 window对象是BOM的核心，window对象指当前的浏览器窗口。

* 语法：

function openWindow(){

window.open();

}

* 方法



2、计时器对象

* 在JavaScript中，我们可以在设定的时间间隔之后来执行代码，而不是在函数被调用后立即执行。
* 计时器类型：

一次性计时器：仅在指定的延迟时间之后触发一次。

间隔性触发计时器：每隔一定的时间间隔就触发一次。

* 方法：



1. setInterval()在执行时,从载入页面后每隔指定的时间执行代码。

语法:

setInterval(代码,交互时间);

setInterval(“clock()”,100);

参数说明：

代码：要调用的函数或要执行的代码串。

交互时间：

周期性执行或调用表达式之间的时间间隔，以毫秒计（1s=1000ms）。

返回值:

一个可以传递给 clearInterval() 从而取消对"代码"的周期性执行的值。

1. clearInterval() 方法可取消由 setInterval() 设置的交互时间。

 语法：

clearInterval(id\_of\_setInterval);

var i=setInterval("clock()”,100);

clearInterval(i);

 参数说明:

id\_of\_setInterval：由 setInterval() 返回的 ID 值。

1. setTimeout()计时器，在载入后延迟指定时间后,去执行一次表达式,仅执行一次。

 语法:

setTimeout(代码,延迟时间);

无限计数器：

function startCount() {

document.getElementById('count').value=num;

num=num+1;

setTimeout("startCount()",1000);

}

参数说明：

要调用的函数或要执行的代码串。

延时时间：在执行代码前需等待的时间，以毫秒为单位（1s=1000ms)。

1. setTimeout()和clearTimeout()一起使用，停止计时器。

 语法:

clearTimeout(id\_of\_setTimeout)

var i=setTimeout(“clock”,100);

clearTimeout(i);

 参数说明:

id\_of\_setTimeout：由 setTimeout() 返回的 ID 值。该值标识要取消的延迟执行代码块。

3、history对象

* history对象记录了用户曾经浏览过的页面(URL)，并可以实现浏览器前进与后退相似导航的功能。

注意:从窗口被打开的那一刻开始记录，每个浏览器窗口、每个标签页乃至每个框架，都有自己的history对象与特定的window对象关联。

 语法：

window.history.[属性|方法]

 注意：window可以省略。

 属性：window.history.length;

 方法：

back()—加载前一个URL界面; back()相当于go(-1),等于点击倒退按钮。

forward()—加载下一个URL界面; forward()相当于go(1),等于点击前进按钮。

 go()—加载history列表中的某一个页面。

语法：

window.history.go(number);

4、Location对象

* location用于获取或设置窗体的URL，并且可以用于解析URL。

 语法:

location.[属性|方法]

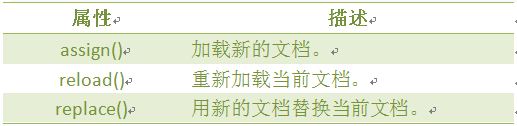
document.write(window.location);

 属性：





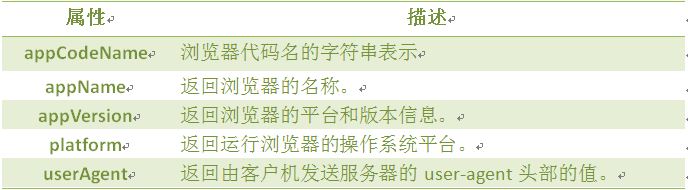
方法：



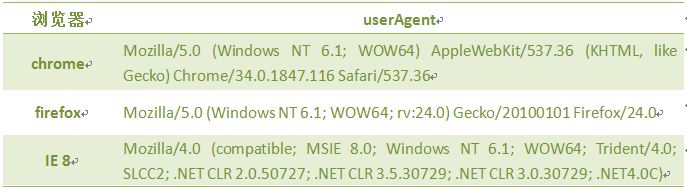
5、Navigator对象

* Navigator 对象包含有关浏览器的信息，通常用于检测浏览器与操作系统的版本。

 属性：



UserAgent：使用userAgent判断使用的是什么浏览器(假设使用的是IE8浏览器),



 语法：

var u\_agent =window.navigator.userAgent ;

var B\_name="不是想用的主流浏览器!";

if(u\_agent.indexOf("Firefox")>-1){

B\_name="Firefox";

}else if(u\_agent.indexOf("Chrome")>-1){

B\_name="Chrome";

}else if(u\_agent.indexOf("MSIE")>-1&&u\_agent.indexOf("Trident")>-1){

B\_name="IE(8-10)";

}

document.write("浏览器:"+B\_name+"<br>");

document.write("u\_agent:"+u\_agent+"<br>");

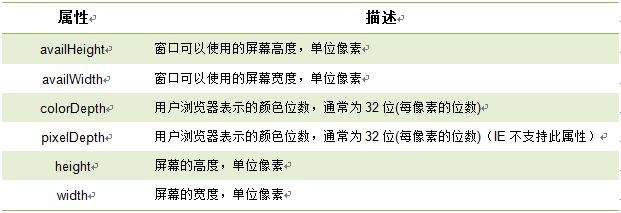
}

6、screen对象

 语法：

Window.screen.属性

属性：



* window.screen 对象包含有关用户屏幕的信息。

1. screen.height 返回屏幕分辨率的高

2. screen.width 返回屏幕分辨率的宽

注意:

1.单位以像素计。

2. window.screen 对象在编写时可以不使用 window 这个前缀。

语法：

document.write( "屏幕宽度："+screen.width+"px<br>");

document.write( "屏幕高度："+screen.height+"px<br>");

* 屏幕可用高度宽度

1. screen.availWidth 属性返回访问者屏幕的宽度，以像素计，减去界面特性，比如任务栏。

2. screen.availHeight 属性返回访问者屏幕的高度，以像素计，减去界面特性，比如任务栏。

六、DOM对象

1、简介

文档对象模型DOM（Document Object Model）定义访问和处理HTML文档的标准方法。DOM 将HTML文档呈现为带有元素、属性和文本的树结构（节点树）。

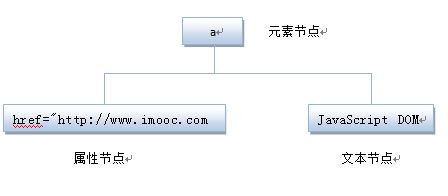


* HTML文档可以说由节点构成的集合，三种常见的DOM节点:

元素节点：上图中<html>、<body>、<p>等都是元素节点，即标签。

文本节点:向用户展示的内容，如<li>...</li>中的JavaScript、DOM、CSS等文本。

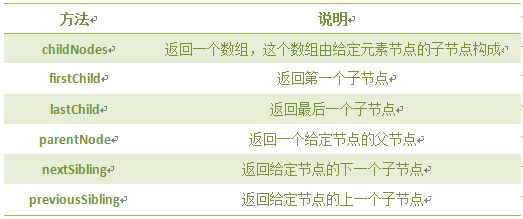
属性节点:元素属性，如<a>标签的链接属性href="http://www.imooc.com"。



* 节点属性：



* 遍历节点树：



* DOM相关操作方法：-- 注意：前两个是document操作。



2、获取DOM对象

1. 通过ID属性获取 – document.getElementById() 方法

语法：

document. getElementById (id)

function getnum(){

var mynode=document. getElementById ("myt");

alert(mynode.length);

}

1. 通过name属性获取 -- document.getElementsByName()方法

语法：

document.getElementsByName(name)

function getnum(){

var mynode=document.getElementsByName("myt");

alert(mynode.length);

}

与getElementById() 方法不同的是，通过元素的 name 属性查询元素，而不是通过 id 属性。

注意:

 因为文档中的 name 属性可能不唯一，所有 getElementsByName() 方法返回的是元素的数组，而不是一个元素。

 和数组类似也有length属性，可以和访问数组一样的方法来访问，从0开始。

1. 通过标签名获取 -- document.getElementsByTagName()方法

* 返回带有指定标签名的节点对象的集合。返回元素的顺序是它们在文档中的顺序。

 语法:

document.getElementsByTagName(Tagname)

function getTagElements()

{

var myI=document.getElementsByTagName("input");

alert(myI.length);

}

 说明:

 Tagname是标签的名称，如p、a、img等标签名。

 和数组类似也有length属性，可以和访问数组一样的方法来访问，所以从0开始。

3、设置节点属性 – DOM对象的方法

* getAttribute()方法--通过元素节点的属性名称**获取**属性的值。

 语法：

 elementNode.getAttribute(name);

 var text=con[i].getAttribute("title");

 说明:

 elementNode：使用getElementById()、getElementsByTagName()等方法，获取到的元素节点。

 name：要想查询的元素节点的属性名字

* setAttribute()方法--**增加**一个指定名称和值的新属性，把一个现有的属性设定为指定的值。

 语法：

 elementNode.setAttribute(name,value)

 Lists[i].setAttribute("title","WEB前端技术");//括号里面都要用双引号

 说明：

 name: 要设置的属性名。

 value: 要设置的属性值。

 注意：

 把指定的属性设置为指定的值。如果不存在具有指定名称的属性，该方法将创建一个新属性。

 类似于getAttribute()方法，setAttribute()方法只能通过元素节点对象调用的函数。

4、节点属性 – DOM对象的属性

* 在文档对象模型 (DOM) 中，每个节点都是一个对象。DOM 节点有三个重要的属性 ：

 nodeName : 节点的名称

 nodeValue ：节点的值

 nodeType ：节点的类型



1. nodeName 属性: 节点的名称，是只读的。

 元素节点的 nodeName 与标签名相同

 属性节点的 nodeName 是属性的名称

 文本节点的 nodeName 永远是 #text

 文档节点的 nodeName 永远是 #document



1. nodeValue 属性：节点的值

 元素节点的 nodeValue 是 undefined 或 null

 文本节点的 nodeValue 是文本自身

 属性节点的 nodeValue 是属性的值

1. nodeType 属性: 节点的类型，是只读的。以下常用的几种结点类型:

 元素类型 节点类型

元素 1

属性 2

文本 3

注释 8

文档 9

 语法：

var my=document.getElementsByTagName("li");

for(var i=0;i<my.length;i++){

document.write("节点名称："+my[i].nodeName+"<br>");

document.write("节点的值："+my[i].nodeValue+"<br>");

document.write("节点类型："+my[i].nodeType+"<br>");

}

5、访问其他节点的方法

1. 访问子节点—返回的是一个数组

访问选定元素节点下的所有子节点的列表，返回的值可以看作是一个数组，他具有length属性。

**语法**：

elementNode.childNodesl;

var my=document.getElementsByTagName("div")[0].childNodes;

for(var i=0;i<my.length;i++){

document.write("节点类型："+my[i].nodeType+"<br>");

document.write("节点的值："+my[i].nodeValue+"<br>");

document.write("节点名称："+my[i].nodeName+"<br>");

}

**注意**：

如果选定的节点没有子节点，则该属性返回不包含节点的 NodeList。

1. 访问第一个/最后一个子节点 – 通过属性访问

firstChild 属性返回‘childNodes’数组的第一个子节点。如果选定的节点没有子节点，则该属性返回 NULL。

语法：

node.firstChild

说明：

与elementNode.childNodes[0]是同样的效果。

lastChild 属性返回‘childNodes’数组的最后一个子节点。如果选定的节点没有子节点则该属性返回 NULL。

语法：

node.lastChild

说明：

与elementNode.childNodes[elementNode.childNodes.length-1]是同样的效果。

注意:

上一节中，我们知道Internet Explorer 会忽略节点之间生成的空白文本节点，而其它浏览器不会。我们可以通过检测节点类型，过滤子节点。 (以后章节讲解)

语法：

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

document.write(x.firstChild.nodeType);

document.write(x.lastChild.nodeType);

1. 父节点

获取指定节点的父节点

语法：

elementNode.parentNode

注意:

父节点只能有一个。

访问祖节点:

elementNode.parentNode.parentNode

1. 访问兄弟节点

 **nextSibling 属性**可返回某个节点之后紧跟的节点（处于同一树层级中）。

 语法：

nodeObject.nextSibling

 说明：如果无此节点，则该属性返回 null。



**previousSibling 属性**可返回某个节点之前紧跟的节点（处于同一树层级中）。

 语法：

nodeObject.previousSibling

 说明：

如果无此节点，则该属性返回 null。

 注意:

两个属性获取的是节点。Internet Explorer 会忽略节点间生成的空白文本节点（例如，换行符号），而其它浏览器不会忽略。

 解决问题方法:

判断节点nodeType是否为1, 如是为元素节点，跳过。

语法：

while (x && x.nodeType!=1){

x=x.nextSibling;

}

6、对节点操作的方法

1. 插入节点

appendChild()在指定节点的最后一个子节点列表之后添加一个新的子节点。

语法:

appendChild(newnode)

var otest = document.getElementById("test");

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

newnode.innerHTML="PHP";

otest.appendChild(newnode);

insertBefore() 方法可在已有的子节点前插入一个新的子节点。

语法：

var otest = document.getElementById("test");

var node = document.getElementsByTagName("li")[1];

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

newnode.innerHTML="PHP";

otest.insertBefore(newnode,node);

1. 删除节点

removeChild() 方法从子节点列表中删除某个节点。如删除成功，此方法可返回被删除的节点，如失败，则返回 NULL。

注意:

把删除的子节点赋值给 x，这个子节点不在DOM树中，但是还存在内存中，可通过 x 操作，如果要完全删除对象，给 x 赋 null 值。

var h=content.removeChild(content.childNodes[1]);

h=null;

语法：

<script type="text/javascript">

function clearText() {

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

var h=content.removeChild(content.childNodes[1]);

document.write("删除节点的内容："+h.innerHTML);

for(var i=0;i<content.childNodes.length;i++){

document.write(content.childNodes[i].innerHTML+"<br>");

}

}

1. 替换节点—这些方法都是父节点来操作子节点的

replaceChild ()实现子节点(对象)的替换。返回被替换对象的引用。

node.replaceChild (newnode,oldnew )

参数：

newnode : 必需，用于替换 oldnew 的对象。

oldnew : 必需，被 newnode 替换的对象。

注意:

当 oldnode 被替换时，所有与之相关的属性内容都将被移除。

newnode 必须先被建立。

语法：

function replaceMessage(){

var oldnew=document.getElementById("oldnode");

var newnode=document.createElement("i");

newnode.innerHTML="JavaScript";

oldnew.parentNode.replaceChild(newnode,oldnew);

}

1. 创建元素节点—返回一个对象

createElement()方法可创建元素节点。此方法可返回一个 Element 对象。

语法：

document.createElement(tagName)

参数:

tagName：字符串值，这个字符串用来指明创建元素的类型。

注意：

要与appendChild() 或 insertBefore()方法联合使用，将元素显示在页面中。

1. 创建文本节点—返回一个对象

createTextNode() 方法创建新的文本节点，返回新创建的 Text 节点。

语法：

document.createTextNode(data)

参数：

data : 字符串值，可规定此节点的文本。

例子：

var element=document.createElement("div");

element.className="message";

var textNode=document.createTextNode("I Love JavaScript!");

element.appendChild(textNode);

document.body.appendChild(element);

7、网页相关的操作方法

1. 浏览器窗口可视区域大小

 获得浏览器窗口的尺寸（浏览器的视口，不包括工具栏和滚动条）的方法:

 对于IE9+、Chrome、Firefox、Opera 以及 Safari：

 window.innerHeight - 浏览器窗口的内部高度

 window.innerWidth - 浏览器窗口的内部宽度

 对于 Internet Explorer 8、7、6、5：

 document.documentElement.clientHeight表示HTML文档所在窗口的当前高度。

 document.documentElement.clientWidth表示HTML文档所在窗口的当前宽度。

 Document对象的body属性对应HTML文档的<body>标签

 document.body.clientHeight

 document.body.clientWidth

 在不同浏览器都实用的 JavaScript 方案：

 var w= document.documentElement.clientWidth

|| document.body.clientWidth;

 var h= document.documentElement.clientHeight

|| document.body.clientHeight;

1. 网页尺寸scrollHeight

 scrollHeight和scrollWidth，获取网页内容高度和宽度。

 针对IE、Opera:

 scrollHeight 是网页内容实际高度，可以小于 clientHeight。

 针对NS、FF:

 scrollHeight 是网页内容高度，不过最小值是 clientHeight。也就是说网页内容实际高度小于 clientHeight 时，scrollHeight 返回 clientHeight 。

 浏览器兼容性

 var w=document.documentElement.scrollWidth

|| document.body.scrollWidth;

 var h=document.documentElement.scrollHeight

|| document.body.scrollHeight;



注意:

区分大小写

 scrollHeight和scrollWidth还可获取Dom元素中内容实际占用的高度和宽度。

1. 网页尺寸offsetHeight

 offsetHeight和offsetWidth，获取网页内容高度和宽度(包括滚动条等边线，会随窗口的显示大小改变)。

 值

offsetHeight = clientHeight + 滚动条 + 边框。

 浏览器兼容性

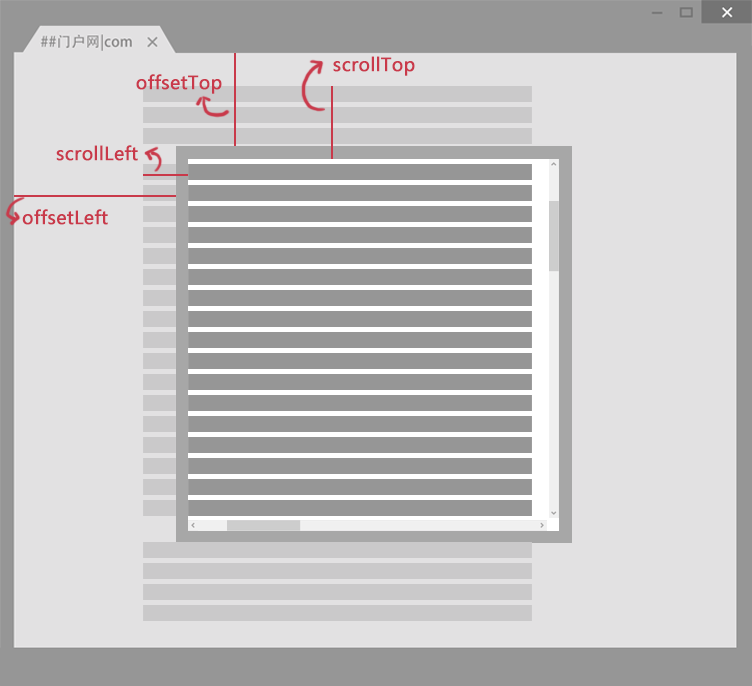
var w= document.documentElement.offsetWidth

|| document.body.offsetWidth;

var h= document.documentElement.offsetHeight

|| document.body.offsetHeight;

1. 网页卷去的距离与偏移量—滚动之后会变化



scrollLeft:设置或获取位于给定对象左边界与窗口中目前可见内容的最左端之间的距离 ，即左边灰色的内容。

scrollTop:设置或获取位于对象最顶端与窗口中可见内容的最顶端之间的距离 ，即上边灰色的内容。

offsetLeft:获取指定对象相对于版面或由 offsetParent 属性指定的父坐标的计算左侧位置 。

offsetTop:获取指定对象相对于版面或由 offsetParent 属性指定的父坐标的计算顶端位置 。

注意:

 1. 区分大小写

 2. offsetParent：布局中设置postion属性(Relative、Absolute、fixed)的父容器，从最近的父节点开始，一层层向上找，直到HTML的body。