**八、javascriptBOM**

**一、DOM概念**

DOM：

英文全称：document object model

JS：所有东西作为对象。

html 骨架

css 装修

javascript 物业（用户交互）

<div id = "div1">测试div</div>

一共分为三种节点类型：

1、<div></div> 元素节点

2、id = "div1" 属性节点

3、测试div 文本节点

</script>

</head>

<body>

<div id = "div1">测试div</div>

</body>

**二、getElementById**

【功能】通过id获取当前页面上符合条件的元素节点。

document.getElementById(id)

window.onload 是页面加载完成以后加载的函数。

【注】所有的节点都是对象。

【注】所有的元素节点有下述两个属性。

(1)tagName：获取当前元素节点的类型。

(2)innerHTML：元素标签间的内容。

\*/

window.onload = function(){

var oDiv = document.getElementById("div1");

//获取当前元素节点中的某一个属性

// alert(oDiv.id);

// oDiv.id = "div3";

// alert(oDiv.title);

/\*

【注】在访问class属性的时候使用className访问。

\*/

// alert(oDiv.className);

oDiv.className = "box";

/\*

【注】name属性一般情况下，文本输入框才有。

\*/

var oInput = document.getElementById("input\_id");

//alert(oInput.name);

}

</script>

</head>

<body>

<div id = "div1" title = "\_title" class = "\_class"><em>斜体</em></div>

<input type = "text" name = "username" id = "input\_id"/>

<p id = "p1"></p>

</body>

**三、getElementsByTagName**

document.getElementsByTagName(tagName);

node.getElementsByTagName(tagName);

【返回值】 [object HTMLCollection] 可当做数组对待

window.onload = function(){

// var aLis = document.getElementsByTagName("li");

// alert(aLis.length);

// alert(aLis[0].innerHTML);

/\*

【注】想获取当前页面上ul下的li标签

\*/

var oUl = document.getElementById("ul1");

var aLis = oUl.getElementsByTagName("li");

// alert(aLis.length);

alert(aLis[2].innerHTML);

}

</script>

</head>

<body>

<ul id = "ul1">

<li>11111</li>

<li>22222</li>

<li>33333</li>

</ul>

<ol>

<li>44444</li>

<li>55555</li>

<li>66666</li>

</ol>

</body>

**三、getElementsByName**

document.getElementsByName(name的值);

【功能】根据name属性的值，获取，值相同的所有的元素节点。

【注】尽量用在文本输入框的节点上。

[object NodeList] 数组对待

\*/

window.onload = function(){

// var nodes = document.getElementsByName("hello");

// alert(nodes.length);

// alert(nodes[1].tagName);

// 错误：

var oDiv = document.getElementById("div1");

var nodes = oDiv.getElementsByName("hello");

alert(nodes.length);

}

</script>

</head>

<body>

<div id = "div1">

<span name = "hello"></span>

</div>

<div name = "hello"></div>

<p name = "hello"></p>

<input name = "hello" />

</body>

**四、getElementsByClassName**

**1.document.getElementsByClassName(class); IE8以下版本不支持**

node.getElementsByClassName(class);

【注】[object HTMLCollection] 数组

window.onload = function(){

/\*var nodes = document.getElementsByClassName("box");

// alert(nodes.length);

alert(nodes[0].tagName);

alert(nodes[2].tagName);\*/

/\*

【注】获取div1下的class为box的节点

\*/

var oDiv = document.getElementById("div1");

var nodes = oDiv.getElementsByClassName("box");

// alert(nodes[0].tagName);

alert(nodes.length)

}

</script>

</head>

<body>

<div id = "div1">

<span class = "box"></span>

</div>

<p class = "box"></p>

<em class = "box"></em>

</body>

**2.ClassName自定义函数**

/\*

函数内部需要几个值，设置几个参数

\*/

function elementsByClassName(parentNode, className){

var result = []; //用于存储符合条件的元素节点

//1、获取当前节点下，所有的节点

var nodes = parentNode.getElementsByTagName("\*");

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

if(nodes[i].className == className){

//2、存储

result.push(nodes[i]);

}

}

return result;

}

window.onload = function(){

// var nodes = elementsByClassName(document, "box");

// alert(nodes.length);

// alert(nodes[2].tagName);

var oDiv = document.getElementById("div1");

var nodes = elementsByClassName(oDiv, "box");

alert(nodes.length);

alert(nodes[0].tagName);

}

</script>

</head>

<body>

<div id = "div1">

<span class = "box"></span>

</div>

<p class = "box"></p>

<em class = "box"></em>

</body>

**五、获取当前有效样式**

<style>

#div1{

width: 100px;

height: 100px;

background-color: red

}

</style>

<script>

window.onload = function(){

var oDiv = document.getElementById("div1");

/\*

如果，获取当前有效样式。

\*/

// alert(oDiv.currentStyle["width"]); //IE下兼容

// alert(getComputedStyle(oDiv)["width"]); //火狐，谷歌

// alert(getStyle(oDiv, "width"));

// alert(getStyle(oDiv, "backgroundColor"));

}

//获取当前有效样式

**function getStyle(element, attr){**

**return element.currentStyle ? element.currentStyle[attr] : getComputedStyle(element)[attr];**

**}**

</script>

</head>

<body>

<div id = "div1" style = "width: 300px"></div>

</body>

**六、随机颜色**

<head>

<meta charset="UTF-8">

<title>Document</title>

<style>

#div1{

width: 100px;

height: 100px;

background-color: red;

}

</style>

<script>

window.onload = function(){

/\*

rgb()

rgba() 0~255整数

\*/

var oDiv = document.getElementById("div1");

oDiv.onclick = function(){

oDiv.style.backgroundColor = randomColor();

}

}

function randomColor(){

return "rgba(" + parseInt(Math.random() \* 255) + ", " + parseInt(Math.random() \* 255) + ", " + parseInt(Math.random() \* 255) + ", 1)";

}

/\*

#div1 id

.box class

div tagName

name=hello name

\*/

function $(){

}

</script>

</head>

<body>

<div id = "div1"></div>

</body>

**七、onscroll**

<script>

/\*

window.onscroll

【注】在页面进行滚动的时候，不停的调用的方法

\*/

var i = 0;

window.onscroll = function(){

document.title = i++;

}

window.onload = function(){

document.onclick = function(){

var scrollTop = document.body.scrollTop || document.documentElement.scrollTop;

// alert(scrollTop);

//设置滚动高度，同样生效。

}

}

</script>

</head>

<body style = "height: 2000px">

</body>

**【例1】顶部悬浮**

<head>

<meta charset="UTF-8">

<title>Document</title>

<style>

#div1{

width: 400px;

height: 50px;

background-color: gray;

position: absolute;

left: 450px;

top: 0px

}

body{

height: 2000px;

}

</style>

<script>

window.onscroll = function(){

var oDiv = document.getElementById("div1");

//获取到滚动高度

var scrollTop = document.body.scrollTop || document.documentElement.scrollTop;

oDiv.style.top = scrollTop + "px";

}

</script>

</head>

<body>

<div id = "div1"></div>

</body>

**【例2】回到顶部**

<head>

<meta charset="UTF-8">

<title>Document</title>

<style>

#up{

position: absolute;

right: 0px;

bottom: 0px;

width: 100px;

height: 100px;

text-align: center;

line-height: 100px;

font-size: 50px;

background-color: gray

}

body{

height: 2000px;

}

</style>

<script>

window.onload = function(){

var oUp = document.getElementById("up");

var tmp = oUp.offsetTop;

window.onscroll = function(){

//保证向上箭头，永远在右下角

var scrollTop = document.body.scrollTop || document.documentElement.scrollTop;

oUp.style.top = tmp + scrollTop + "px";

}

oUp.onclick = function(){

//回到顶部

document.body.scrollTop = "0px";

document.documentElement.scrollTop = "0px";

}

}

</script>

</head>

<body>

<div id = "up">

↑

</div>

</body>

**八、工具库（tool.js）**

//库

**1.获取当前有效样式**

function getStyle(element, attr){

return element.currentStyle ? element.currentStyle[attr] : getComputedStyle(element)[attr];

}

/\*

**2.随机颜色**

\*/

function randomColor(){

return "rgba(" + parseInt(Math.random() \* 255) + ", " + parseInt(Math.random() \* 255) + ", " + parseInt(Math.random() \* 255) + ", 1)";

}

/\*

**3.通过className获取元素节点**

\*/

function elementsByClassName(parentNode, className){

var result = []; //用于存储符合条件的元素节点

//1、获取当前节点下，所有的节点

var nodes = parentNode.getElementsByTagName("\*");

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

if(nodes[i].className == className){

//2、存储

result.push(nodes[i]);

}

}

return result;

}

**九、操作元素节点属性（Attribute）**

**1.setAttribute();**

【格式】元素节点.setAttribute(attr, value);

**2.getAttribute();**

【格式】元素节点.getAttribute(attr);

var oDiv = document.getElementById("div1");

alert(oDiv.getAttribute("title"));

alert(oDiv.title);\*/

oDiv.setAttribute("title", "xxx");

oDiv.title = "xxx";

【和（元素节点.属性）的区别】

(1)对class属性的访问

alert(oDiv.className);//需通过className访问

alert(oDiv.getAttribute("class"));//可直接通过class访问

(2)自定义属性的设置和访问

oDiv.xxx = "yyy";//如果oDiv中没有xxx属性，则该设置不生效

oDiv.setAttribute("xxx", "yyy");//可以添加xxx = "yyy"的属性

**3.removeAttribute();**

【格式】元素节点.removeAttribute(attr);

oDiv.title = "";//只是将属性值设为空

oDiv.removeAttribute("title");//可将整个属性删除

**十、DOM节点**

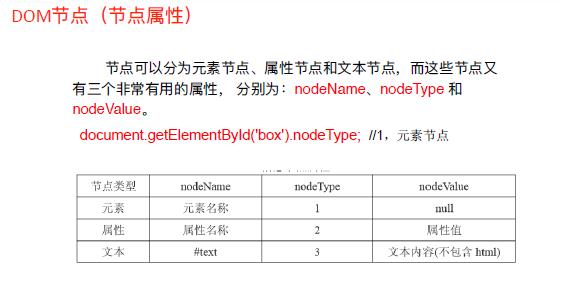
**1.DOM节点属性**

所有的节点类型，都拥有这三个属性。

(1)nodeName

(2)nodeType

(3)nodeValue



**2.文本节点的获取**

获取子节点

(1)firstChild lastChild

(2)parentNode、 返回当前节点的父节点

(3)previousSibling、 返回当前节点兄弟节点中的上一个节点

(4)nextSibling 返回当前节点兄弟节点中的下一个节点

(5)ownerDocument【格式】node.ownerDocument == document

**3.获取当前节点所有的子节点**

childNodes

alert(oDiv.childNodes.length); //object NodeList 数组

**4.剔除空白节点**

如果我们要剔除空白的文本节点

1、确定是文本节点 nodeType == 1

2、判断是一个空的文本节点 /^\s+$/.test(str)

【1】function removeSpaceNode(nodes){

var arr = []; //存储不是空白节点的元素

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

//判断空白节点

if(!(nodes[i].nodeType == 3 && /^\s+$/.test(nodes[i].nodeValue))){

arr.push(nodes[i]);

}

}

return arr;

}

【2】

//删除子节点上的空白节点

function removeSpaceNode2(parent){

for(var i = parent.childNodes.length - 1; i >= 0 ; i--){

if(parent.childNodes[i].nodeType == 3 && /^\s+$/.test(parent.childNodes[i].nodeValue)){

//是空白节点，删除，通过父节点，删除子节点

parent.removeChild(parent.childNodes[i]);

}

}

}

**5.nodeValue和innerHTML的区别**

(1)var oDiv = document.getElementById("div1");

// alert(oDiv.innerHTML); //将div标签内的所有内容都输出

// alert(oDiv.childNodes[1].nodeValue);//输出对应值

(2)设置/赋值

区别：

innerHTML会解析文本中的标签

但是nodeValue是纯文本。

**6.关于this**

(1)this 是一个变量，是系统内置的变量

(2)this 存储值，当前函数的主人是谁，this存储谁

【应用】【例】隔行变色

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Document</title>

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

<script>

/\*

【注】this 是一个变量，是系统内置的变量

【注】this 存储值，当前函数的主人是谁，this存储谁

\*/

window.onload = function(){

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

//清除所有的空白文本节点。

removeSpaceNode2(aLis[0].parentNode);

//通过循环给每一个元素添加移入移出事件

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

aLis[i].onmouseover = function(){

//this 移入的li标签

this.style.backgroundColor = "red";

if(this.previousSibling){

this.previousSibling.style.backgroundColor = "yellow";

}

if(this.nextSibling){

this.nextSibling.style.backgroundColor = "yellow";

}

}

aLis[i].onmouseout = function(){

this.style.backgroundColor = "";

if(this.previousSibling){

this.previousSibling.style.backgroundColor = "";

}

if(this.nextSibling){

this.nextSibling.style.backgroundColor = "";

}

}

}

}

</script>

</head>

<body>

<ul id = "ul1">

<li>11111111111111111</li>

<li>11111111111111111</li>

<li>11111111111111111</li>

<li>11111111111111111</li>

<li>11111111111111111</li>

<li>11111111111111111</li>

</ul>

</body>

</html>

**7.attributes**

attributes 获取当前元素节点上的属性节点（是一个集合）。

数组

可以存储重复相等的数据，有序

集合

①无序

②不重复

window.onload = function(){

var oDiv = document.getElementById("div1");

//集合

// alert(oDiv.attributes); //object NamedNodeMap

//取出，其中某一个属性节点

// alert(oDiv.attributes.getNamedItem("title")); //object Attr

// alert(oDiv.attributes["title"]);

/\*

属性节点中的属性的健值如何去。

nodeName nodeType nodeValue

key 2 value

\*/

}

**十一、DOM节点操作**



**1.write**

window.onload = function(){

document.onclick = function(){

//【注】写入文本，会将原有文本覆盖

document.write("hello world");

}

}

**2.createElement()**

【格式】document.createElement(元素的类型);

【返回值】新生成的元素节点。

【例】创建div节点

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

**3.createTextNode()**

【功能】创建文本节点。

【格式】document.createTextNode(文本);

【返回值】文本节点

【例】创建文本节点

var oTxt = document.createTextNode("<strong>粗体</strong>");

**4.appendChild()**

【格式】parent.appendChild(要插入的节点);

【功能】将节点插入到parent子节点的位置，并且是子节点的末尾

【例】插入一个带文本的节点

oDiv.appendChild(node);

**5.封装一个可创建一个带文本的节点的函数**

function createNodeWithText(tagName, text){

//1、创建元素节点

var node = document.createElement(tagName);

//2、创建文本节点

var oTxt = document.createTextNode(text);

//3、将文本节点插入元素节点内

node.appendChild(oTxt);

//4、返回元素节点

return node;

}

**6.insertBefore()**

【格式】oldNode.parentNode.insertBefore(newNode, oldNode);

【作用】将newNode插入到oldNode之前。

\*可用**insertBefore()封装一个**将newNode插入到oldNode之后的函数

function **insertAfter**(newNode, oldNode){

var parent = oldNode.parentNode;

if(parent.lastChild == oldNode){

//最后一个节点，插入到末尾

parent.appendChild(newNode);

}else{

//找到下一个节点

parent.insertBefore(newNode, oldNode.nextSibling);

}

}

**7.repalceChild()**

【功能】用新节点替换旧节点。

【格式】oldNode.parentNode.replaceChild(newNode, oldNode);

**8.cloneNode()**

【格式】node.cloneNode(true);

【返回值】克隆出来的元素节点。

【参数】true 将内部文本一起进行克隆 默认false

<script>

window.onload = function(){

var oDiv1 = document.getElementById("div1");

var oDiv2 = document.getElementById("div2");

var node = oDiv1.cloneNode(true);

oDiv2.appendChild(node);

}

</script>

</head>

<body>

<div id = "div1">文本内容</div>

<div id = "div2">

</div>

</body>

**9.removeChild();**

【功能】删除节点

【格式】node.parentNode.removeChild(node);

**十二、DOM补充**

**1.查找元素节点**

document.querySelector() 返回找到的第一个元素节点

document.querySelectorAll() 返回找到的所有元素节点

【关键点】 参数

CSS选择器

//#id

// var node = document.querySelector("#div1");

//.class

// var node = document.querySelector(".box");

// var node = document.querySelector("div");

// var node = document.querySelector("[name=\_name]");

// var node = document.querySelector("[title=\_title]");

**2.补充知识**



(1)【注】用firstElementChild、lastElementChil、children等会忽略空白节点去访问

(2)①oDiv.innerHTML

//标签间的内容

②oDiv.outerHTML

//整个标签的内容