**1.1 节点属性**

Node.nodeName //返回节点名称，只读

Node.nodeType //返回节点类型的常数值，只读

Node.nodeValue //返回Text或Comment节点的文本值，只读

Node.textContent //返回当前节点和它的所有后代节点的文本内容，可读写

Node.baseURI //返回当前网页的绝对路径

Node.ownerDocument //返回当前节点所在的顶层文档对象，即document

Node.nextSibling //返回紧跟在当前节点后面的第一个兄弟节点

Node.previousSibling //返回当前节点前面的、距离最近的一个兄弟节点

Node.parentNode //返回当前节点的父节点

Node.parentElement //返回当前节点的父Element节点

Node.childNodes //返回当前节点的所有子节点

Node.firstChild //返回当前节点的第一个子节点

Node.lastChild //返回当前节点的最后一个子节点

//parentNode接口

Node.children //返回指定节点的所有Element子节点

Node.firstElementChild //返回当前节点的第一个Element子节点

Node.lastElementChild //返回当前节点的最后一个Element子节点

Node.childElementCount //返回当前节点所有Element子节点的数目。

1.2操作

Node.appendChild(node) //向节点添加最后一个子节点

Node.hasChildNodes() //返回布尔值，表示当前节点是否有子节点

Node.cloneNode(true); // 默认为false(克隆节点), true(克隆节点及其属性，以及后代)

Node.insertBefore(newNode,oldNode) // 在指定子节点之前插入新的子节点

Node.removeChild(node) //删除节点，在要删除节点的父节点上操作

Node.replaceChild(newChild,oldChild) //替换节点

Node.contains(node) //返回一个布尔值，表示参数节点是否为当前节点的后代节点。

Node.compareDocumentPosition(node) //返回一个7个比特位的二进制值，表示参数节点和当前节点的关系

Node.isEqualNode(noe) //返回布尔值，用于检查两个节点是否相等。所谓相等的节点，指的是两个节点的类型相同、属性相同、子节点相同。

Node.normalize() //用于清理当前节点内部的所有Text节点。它会去除空的文本节点，并且将毗邻的文本节点合并成一个。

//ChildNode接口

Node.remove() //用于删除当前节点

Node.before() //

Node.after()

Node.replaceWith()

1.3document节点属性

document.doctype //

document.documentElement //返回当前文档的根节点

document.defaultView //返回document对象所在的window对象

document.body //返回当前文档的<body>节点

document.head //返回当前文档的<head>节点

document.activeElement //返回当前文档中获得焦点的那个元素。

//节点集合属性

document.links //返回当前文档的所有a元素

document.forms //返回页面中所有表单元素

document.images //返回页面中所有图片元素

document.embeds //返回网页中所有嵌入对象

document.scripts //返回当前文档的所有脚本

document.styleSheets //返回当前网页的所有样式表

//文档信息属性

document.documentURI //表示当前文档的网址

document.URL //返回当前文档的网址

document.domain //返回当前文档的域名

document.lastModified //返回当前文档最后修改的时间戳

document.location //返回location对象，提供当前文档的URL信息

document.referrer //返回当前文档的访问来源

document.title //返回当前文档的标题

document.characterSet属性返回渲染当前文档的字符集，比如UTF-8、ISO-8859-1。

document.readyState //返回当前文档的状态

document.designMode //控制当前文档是否可编辑，可读写

document.compatMode //返回浏览器处理文档的模式

document.cookie //用来操作Cookie

1.3document节点方法

**（1）读写方法**

document.open() //用于新建并打开一个文档

document.close() //不安比open方法所新建的文档

document.write() //用于向当前文档写入内容

document.writeIn() //用于向当前文档写入内容，尾部添加换行符。

**（2）查找节点**

document.querySelector(selectors) //接受一个CSS选择器作为参数，返回第一个匹配该选择器的元素节点。

document.querySelectorAll(selectors) //接受一个CSS选择器作为参数，返回所有匹配该选择器的元素节点。

document.getElementsByTagName(tagName) //返回所有指定HTML标签的元素

document.getElementsByClassName(className) //返回包括了所有class名字符合指定条件的元素

document.getElementsByName(name) //用于选择拥有name属性的HTML元素（比如<form>、<radio>、<img>、<frame>、<embed>和<object>等）

document.getElementById(id) //返回匹配指定id属性的元素节点。

document.elementFromPoint(x,y) //返回位于页面指定位置最上层的Element子节点。

**（3）生成节点**

document.createElement(tagName) //用来生成HTML元素节点。

document.createTextNode(text) //用来生成文本节点

document.createAttribute(name) //生成一个新的属性对象节点，并返回它。

document.createDocumentFragment() //生成一个DocumentFragment对象

**（4）事件方法**

document.createEvent(type) //生成一个事件对象，该对象能被element.dispatchEvent()方法使用

document.addEventListener(type,listener,capture) //注册事件

document.removeEventListener(type,listener,capture) //注销事件

document.dispatchEvent(event) //触发事件

**（5）其他**

document.hasFocus() //返回一个布尔值，表示当前文档之中是否有元素被激活或获得焦点。

document.adoptNode(externalNode) //将某个节点，从其原来所在的文档移除，插入当前文档，并返回插入后的新节点。

document.importNode(externalNode, deep) //从外部文档拷贝指定节点，插入当前文档。

原文链接：<http://caibaojian.com/javascript-api-collection.html>

**1.4 Element节点**[·](http://caibaojian.com/javascript-api-collection.html)

**1.4.1 Element节点的属性**

**（1）特性属性**

Element.attributes //返回当前元素节点的所有属性节点

Element.id //返回指定元素的id属性，可读写

Element.tagName //返回指定元素的大写标签名

Element.innerHTML //返回该元素包含的HTML代码，可读写

Element.outerHTML //返回指定元素节点的所有HTML代码，包括它自身和包含的的所有子元素，可读写

Element.className //返回当前元素的class属性，可读写

Element.classList //返回当前元素节点的所有class集合

Element.dataset //返回元素节点中所有的data-\*属性

**（2）尺寸属性**

Element.clientHeight //返回元素节点可见部分的高度

Element.clientWidth //返回元素节点可见部分的宽度

Element.clientLeft //返回元素节点左边框的宽度

Element.clientTop //返回元素节点顶部边框的宽度

Element.scrollHeight //返回元素节点的总高度

Element.scrollWidth //返回元素节点的总宽度

Element.scrollLeft //返回元素节点的水平滚动条向右滚动的像素数值,通过设置这个属性可以改变元素的滚动位置

Element.scrollTop //返回元素节点的垂直滚动向下滚动的像素数值

Element.offsetHeight //返回元素的垂直高度(包含border,padding)

Element.offsetWidth //返回元素的水平宽度(包含border,padding)

Element.offsetLeft //返回当前元素左上角相对于Element.offsetParent节点的垂直偏移

Element.offsetTop //返回水平位移

Element.style //返回元素节点的行内样式

**（3）节点相关属性**

Element.children //包括当前元素节点的所有子元素

Element.childElementCount //返回当前元素节点包含的子HTML元素节点的个数

Element.firstElementChild //返回当前节点的第一个Element子节点

Element.lastElementChild //返回当前节点的最后一个Element子节点

Element.nextElementSibling //返回当前元素节点的下一个兄弟HTML元素节点

Element.previousElementSibling //返回当前元素节点的前一个兄弟HTML节点

Element.offsetParent //返回当前元素节点的最靠近的、并且CSS的position属性不等于static的父元素。

**1.4.2 Element节点的方法**

**（1）位置方法**

getBoundingClientRect()

// getBoundingClientRect返回一个对象，包含top,left,right,bottom,width,height // width、height 元素自身宽高

// top 元素上外边界距窗口最上面的距离

// right 元素右外边界距窗口最上面的距离

// bottom 元素下外边界距窗口最上面的距离

// left 元素左外边界距窗口最上面的距离

// width 元素自身宽(包含border,padding)

// height 元素自身高(包含border,padding)

getClientRects() //返回当前元素在页面上形参的所有矩形。

// 元素在页面上的偏移量

var rect = el.getBoundingClientRect()

return {

top: rect.top + document.body.scrollTop,

left: rect.left + document.body.scrollLeft

}

**（2）属性方法**

Element.getAttribute()：读取指定属性

Element.setAttribute()：设置指定属性

Element.hasAttribute()：返回一个布尔值，表示当前元素节点是否有指定的属性

Element.removeAttribute()：移除指定属性

**（3）查找方法**

Element.querySelector()

Element.querySelectorAll()

Element.getElementsByTagName()

Element.getElementsByClassName()

**（4）事件方法**

Element.addEventListener()：添加事件的回调函数

Element.removeEventListener()：移除事件监听函数

Element.dispatchEvent()：触发事件

//ie8

Element.attachEvent(oneventName,listener)

Element.detachEvent(oneventName,listener)

// event对象

var event = window.event||event;

// 事件的目标节点

var target = event.target || event.srcElement;

// 事件代理

ul.addEventListener('click', function(event) {

if (event.target.tagName.toLowerCase() === 'li') {

console.log(event.target.innerHTML)

}

});

**（5）其他**

Element.scrollIntoView() //滚动当前元素，进入浏览器的可见区域

//解析HTML字符串，然后将生成的节点插入DOM树的指定位置。

Element.insertAdjacentHTML(where, htmlString);

Element.insertAdjacentHTML('beforeBegin', htmlString); // 在该元素前插入

Element.insertAdjacentHTML('afterBegin', htmlString); // 在该元素第一个子元素前插入

Element.insertAdjacentHTML('beforeEnd', htmlString); // 在该元素最后一个子元素后面插入

Element.insertAdjacentHTML('afterEnd', htmlString); // 在该元素后插入

Element.remove() //用于将当前元素节点从DOM中移除

Element.focus() //用于将当前页面的焦点，转移到指定元素上

**二、CSS操作**

**（1）类名操作**

//ie8以下

Element.className //获取元素节点的类名

Element.className += ' ' + newClassName //新增一个类名

//判断是否有某个类名

function hasClass(element,className){

return new RegExp(className,'gi').test(element.className);

}

//移除class

function removeClass(element,className){

element.className = element.className.replace(new RegExp('(^|\\b)' + className.split(' ').join('|') + '(\\b|$)', 'gi'),'');

}

//ie10

element.classList.add(className) //新增

element.classList.remove(className) //删除

element.classList.contains(className) //是否包含

element.classList.toggle(className) //toggle class

**（2）style操作**

element.setAttribute('style','')

element.style.backgroundColor = 'red'

element.style.cssText //用来读写或删除整个style属性

element.style.setProperty(propertyName,value) //设置css属性

element.style.getPropertyValue(property) //获取css属性

element.style.removeProperty(property) //删除css属性

操作非内联样式

//ie8

element.currentStyle[attrName]

//ie9+

window.getComputedStyle(el,null)[attrName]

window.getComputedStyle(el,null).getPropertyValue(attrName)

//伪类

window.getComputedStyle(el,':after')[attrName]

**三、对象**

**3.1 Object对象**

**（1）生成实例对象**

var o = new Object()

**（2）属性**

Object.prototype //返回原型对象

**（3）方法**

Object.keys(o) //遍历对象的可枚举属性

Object.getOwnPropertyName(o) //遍历对象不可枚举的属性

对象实例的方法

valueOf()：返回当前对象对应的值。

toString()：返回当前对象对应的字符串形式。

toLocaleString()：返回当前对象对应的本地字符串形式。

hasOwnProperty()：判断某个属性是否为当前对象自身的属性，还是继承自原型对象的属性。

isPrototypeOf()：判断当前对象是否为另一个对象的原型。

propertyIsEnumerable()：判断某个属性是否可枚举。

**3.2 Array对象**

**（1）生成实例对象**

var a = new Array()

**（2）属性**

a.length //长度

**（3）Array.isArray()**

Array.isArray(a) //用来判断一个值是否为数组

**（4）Array实例的方法**

a.valueof() //返回数组本身

a.toString() //返回数组的字符串形式

a.push(value,vlaue....) //用于在数组的末端添加一个或多个元素，并返回添加新元素后的数组长度。

pop() //用于删除数组的最后一个元素，并返回该元素

join() //以参数作为分隔符，将所有数组成员组成一个字符串返回。如果不提供参数，默认用逗号分隔。

concat() //用于多个数组的合并。它将新数组的成员，添加到原数组的尾部，然后返回一个新数组，原数组不变。

shift() //用于删除数组的第一个元素，并返回该元素。

unshift(value) //用于在数组的第一个位置添加元素，并返回添加新元素后的数组长度。

reverse() //用于颠倒数组中元素的顺序，返回改变后的数组

slice(start\_index, upto\_index); //用于提取原数组的一部分，返回一个新数组，原数组不变。第一个参数为起始位置（从0开始），第二个参数为终止位置（但该位置的元素本身不包括在内）。如果省略第二个参数，则一直返回到原数组的最后一个成员。负数表示倒数第几个。

splice(index, count\_to\_remove, addElement1, addElement2, ...); //用于删除原数组的一部分成员，并可以在被删除的位置添加入新的数组成员，返回值是被删除的元素。第一个参数是删除的起始位置，第二个参数是被删除的元素个数。如果后面还有更多的参数，则表示这些就是要被插入数组的新元素。

sort() //对数组成员进行排序，默认是按照字典顺序排序。排序后，原数组将被改变。如果想让sort方法按照自定义方式排序，可以传入一个函数作为参数，表示按照自定义方法进行排序。该函数本身又接受两个参数，表示进行比较的两个元素。如果返回值大于0，表示第一个元素排在第二个元素后面；其他情况下，都是第一个元素排在第二个元素前面。

map() //对数组的所有成员依次调用一个函数，根据函数结果返回一个新数组。

map(elem,index,arr) //map方法接受一个函数作为参数。该函数调用时，map方法会将其传入三个参数，分别是当前成员、当前位置和数组本身。

forEach() //遍历数组的所有成员，执行某种操作,参数是一个函数。它接受三个参数，分别是当前位置的值、当前位置的编号和整个数组。

filter() //参数是一个函数，所有数组成员依次执行该函数，返回结果为true的成员组成一个新数组返回。该方法不会改变原数组。

some() //用来判断数组成员是否符合某种条件。接受一个函数作为参数，所有数组成员依次执行该函数，返回一个布尔值。该函数接受三个参数，依次是当前位置的成员、当前位置的序号和整个数组。只要有一个数组成员的返回值是true，则整个some方法的返回值就是true，否则false。

every() //用来判断数组成员是否符合某种条件。接受一个函数作为参数，所有数组成员依次执行该函数，返回一个布尔值。该函数接受三个参数，依次是当前位置的成员、当前位置的序号和整个数组。所有数组成员的返回值都是true，才返回true，否则false。

reduce() //依次处理数组的每个成员，最终累计为一个值。从左到右处理（从第一个成员到最后一个成员）

reduceRight() //依次处理数组的每个成员，最终累计为一个值。从右到左（从最后一个成员到第一个成员）

indexOf(s) //返回给定元素在数组中第一次出现的位置，如果没有出现则返回-1。可以接受第二个参数，表示搜索的开始位置

lastIndexOf() //返回给定元素在数组中最后一次出现的位置，如果没有出现则返回-1。

**3.3 Number对象**

**（1）生成对象**

var n = new Number()

**（2）Number对象的属性**

Number.POSITIVE\_INFINITY：正的无限，指向Infinity。

Number.NEGATIVE\_INFINITY：负的无限，指向-Infinity。

Number.NaN：表示非数值，指向NaN。

Number.MAX\_VALUE：表示最大的正数，相应的，最小的负数为-Number.MAX\_VALUE。

Number.MIN\_VALUE：表示最小的正数（即最接近0的正数，在64位浮点数体系中为5e-324），相应的，最接近0的负数为-Number.MIN\_VALUE。

Number.MAX\_SAFE\_INTEGER：表示能够精确表示的最大整数，即9007199254740991。

Number.MIN\_SAFE\_INTEGER：表示能够精确表示的最小整数，即-9007199254740991。

**（4）Number对象实例的方法**

toString() //用来将一个数值转为字符串形式.可以接受一个参数，表示输出的进制。如果省略这个参数，默认将数值先转为十进制，再输出字符串；否则，就根据参数指定的进制，将一个数字转化成某个进制的字符串。

toFixed() //用于将一个数转为指定位数的小数，返回这个小数对应的字符串。

toExponential() //用于将一个数转为科学计数法形式。可传入一个参数，参数表示小数点后有效数字的位数，范围为0到20，超出这个范围，会抛出一个RangeError。

toPrecision() //用于将一个数转为指定位数的有效数字。

**3.4 String 对象**

**（1）生成实例对象**

var s = new String()

**（2）String对象的属性**

s.length //返回字符串的长度

**（3）方法**

s.chatAt(index) //返回指定位置的字符

s.fromCharCode() //该方法的参数是一系列Unicode码点，返回对应的字符串。

s.charCodeAt(index) //返回给定位置字符的Unicode码点（十进制表示）

s.concat(s2) //用于连接两个字符串

s.slice(start,end) //用于从原字符串取出子字符串并返回，不改变原字符串。第一个参数是子字符串的开始位置，第二个参数是子字符串的结束位置（不含该位置）。如果参数是负值，表示从结尾开始倒数计算的位置，即该负值加上字符串长度。

s.substring(start,end) //用于从原字符串取出子字符串并返回，不改变原字符串.第一个参数表示子字符串的开始位置，第二个位置表示结束位置。

s.substr(start,length) //用于从原字符串取出子字符串并返回，不改变原字符串。第一个参数是子字符串的开始位置，第二个参数是子字符串的长度。如果第一个参数是负数，表示倒数计算的字符位置。如果第二个参数是负数，将被自动转为0，因此会返回空字符串。

s.indexOf(s) //返回给定元素在字符串中第一次出现的位置，如果没有出现则返回-1。可以接受第二个参数，表示搜索的开始位置

s.lastIndexOf() //返回给定元素在字符串中最后一次出现的位置，如果没有出现则返回-1。

s.trim() //用于去除字符串两端的空格，返回一个新字符串

s.toLowerCase() //用于将一个字符串全部转为小写,返回一个新字符串，不改变原字符串。

s.toUpperCase() //全部转为大写

s.localeCompare(s2) //用于比较两个字符串。它返回一个整数，如果小于0，表示第一个字符串小于第二个字符串；如果等于0，表示两者相等；如果大于0，表示第一个字符串大于第二个字符串。

s.match(regexp) //用于确定原字符串是否匹配某个子字符串，返回一个数组，成员为匹配的第一个字符串。如果没有找到匹配，则返回null。

s.search() //返回值为匹配的第一个位置。如果没有找到匹配，则返回-1。

s.replace(oldValue,newValue) //用于替换匹配的子字符串，一般情况下只替换第一个匹配（除非使用带有g修饰符的正则表达式）。

s.split() //按照给定规则分割字符串，返回一个由分割出来的子字符串组成的数组。还可传入第二个参数，决定了返回数组的成员数。

**3.5 Math对象**

**（1）属性**

Math.E：常数e。

Math.LN2：2的自然对数。

Math.LN10：10的自然对数。

Math.LOG2E：以2为底的e的对数。

Math.LOG10E：以10为底的e的对数。

Math.PI：常数Pi。

Math.SQRT1\_2：0.5的平方根。

Math.SQRT2：2的平方根。

**（2）数学方法**

Math.abs()：返回参数的绝对值

Math.ceil()：向上取整，接受一个参数，返回大于该参数的最小整数。

Math.floor()：向下取整

Math.max(n,n1,...)：可接受多个参数，返回最大值

Math.min(n,n1,..)：可接受多个参数，返回最小值

Math.pow(n,e)：指数运算, 返回以第一个参数为底数、第二个参数为幂的指数值。

Math.sqrt()：返回参数值的平方根。如果参数是一个负值，则返回NaN。

Math.log()：返回以e为底的自然对数值。

Math.exp()：返回e的指数，也就是常数e的参数次方。

Math.round()：四舍五入

Math.random()：返回0到1之间的一个伪随机数，可能等于0，但是一定小于1。

**（3）三角函数方法**

Math.sin()：返回参数的正弦

Math.cos()：返回参数的余弦

Math.tan()：返回参数的正切

Math.asin()：返回参数的反正弦（弧度值）

Math.acos()：返回参数的反余弦（弧度值）

Math.atan()：返回参数的反正切（弧度值）

**3.6 JSON对象**

**（1）方法**

JSON.stringify()

//用于将一个值转为字符串。该字符串应该符合JSON格式，并且可以被JSON.parse方法还原。

//（JSON.stringify(obj, selectedProperties)）还可以接受一个数组，作为第二个参数，指定需要转成字符串的属性。

//还可以接受第三个参数，用于增加返回的JSON字符串的可读性。如果是数字，表示每个属性前面添加的空格（最多不超过10个）；如果是字符串（不超过10个字符），则该字符串会添加在每行前面。

JSON.parse() //用于将JSON字符串转化成对象。

**3.7 console对象**

console.log(text,text2,...) //用于在console窗口输出信息。它可以接受多个参数，将它们的结果连接起来输出。如果第一个参数是格式字符串（使用了格式占位符），console.log方法将依次用后面的参数替换占位符，然后再进行输出。

console.info() //在console窗口输出信息，同时，会在输出信息的前面，加上一个蓝色图标。

console.debug() //在console窗口输出信息，同时，会在输出信息的前面，加上一个蓝色图标。

console.warn() //输出信息时，在最前面加一个黄色三角，表示警告；

console.error() //输出信息时，在最前面加一个红色的叉，表示出错，同时会显示错误发生的堆栈

console.table() //可以将复合类型的数据转为表格显示。

console.count() //用于计数，输出它被调用了多少次。

console.dir() //用来对一个对象进行检查（inspect），并以易于阅读和打印的格式显示。

console.dirxml() //用于以目录树的形式，显示DOM节点。

console.assert() //接受两个参数，第一个参数是表达式，第二个参数是字符串。只有当第一个参数为false，才会输出第二个参数，否则不会有任何结果。

//这两个方法用于计时，可以算出一个操作所花费的准确时间。

console.time()

console.timeEnd()

//time方法表示计时开始，timeEnd方法表示计时结束。它们的参数是计时器的名称。调用timeEnd方法之后，console窗口会显示“计时器名称: 所耗费的时间”。

console.profile() //用来新建一个性能测试器（profile），它的参数是性能测试器的名字。

console.profileEnd() //用来结束正在运行的性能测试器。

console.group()

console.groupend()

//上面这两个方法用于将显示的信息分组。它只在输出大量信息时有用，分在一组的信息，可以用鼠标折叠/展开。

console.groupCollapsed() //用于将显示的信息分组，该组的内容，在第一次显示时是收起的（collapsed），而不是展开的。

console.trace() //显示当前执行的代码在堆栈中的调用路径。

console.clear() //用于清除当前控制台的所有输出，将光标回置到第一行。