1. DOM
   1. 节点
      1. DOM概述

**1.DOM概念**

（1）DOM是js操作网页的接口，全称为文档对象模型（Document Object Model）。

（2）DOM的作用是将网页转为一个js对象，从而可以使用js对网页进行各种操作（如增删）。

（3）浏览器会根据DOM模型，将HTML文档解析成一系列的节点，再由这些节点组成一个树状结构。

（4）DOM的最小组成单位叫做节点（node），DOM树（文档的树形结构）由12种类型的节点组成。

**2.动态集合和静态集合**

（1）动态集合是指DOM结构的变化能够自动反映到所保存的对象中。

（2）NodeList实例对象是一个类数组对象，可以使用slice()方法将其变成真正的数组。NodeList成员是节点对象，包括childNodes和querySelectorAll()方法返回值。childNodes是动态结合，但querySelectorAll()方法返回值是静态集合NodeStaticList。

（3）HTMLCollection对象与NodeList对象类似，也是节点的集合，不同之处为NodeList集合主要是Node节点的集合，而HTMLCollection集合主要是Element元素节点的集合。HTMLCollection集合包括getElementsByTagName()、getElementsByClassName()、getElementsByName()等方法的返回值，以及children、document.links、document.forms等元素集合。与NodeList对象不同，所有的HTMLCollection对象都是动态的。

（4）NamedNodeMap对象常见实例是attributes属性。

**3.元素特性attribute和对象属性property**

（1）元素特性attribute是指HTML元素标签的特性。对象属性property是指元素节点的属性。

（2）像id、title等既是属性，也是特性。修改属性，其对应的特性会发生改变；修改特性，属性也会改变。

**4.获取元素内容的4个属性**

（1）innerHTML

可读可写。返回调用元素的所有子节点（包括注释等）对应的HTML标记。在写模式下，innerHTML会根据指定的值创建新的DOM树，然后用这个DOM树完全替换调用元素原先的所有子节点。

如果将innerHTML属性设为空，等于删除所有它包含的所有节点。

使用innerHTML比appendChild()的性能高出不少。

（2）outerHTML

也是可读可写。包含被查询元素的开始和结束标签。

（3）innerText

可操作元素中包含的所有文本内容。读取时，会将子节点所有文本拼接起来返回。写入值时，结果会删除元素的所有子节点，插入包含相应文本值的文本节点。



（4）outerText

* + 1. 节点概述

节点至少拥有nodeType、nodeName和nodeValue三个基本属性。其中，nodeType返回节点类型的常数值，分别为1-12。nodeName属性返回节点的名称。nodeValue属性返回或设置当前节点的值，格式为字符串。

**1.元素节点element**

（1）对应网页的HTML标签元素。元素节点的节点类型nodeType值是1，节点名称nodeName值是大写的标签名，nodeValue值是null。



（2）元素节点的父节点parentNode指向包含该元素节点的元素节点Element或文档节点Document。

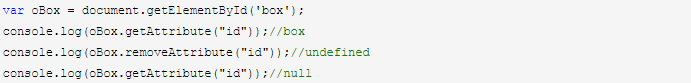
（3）每个元素都有一个或多个特性，可以使用hasAttribute、getAttribute、setAttribute、removeAttribute四种方法操作特性。







注意：通过setAttrbute方法设置的特性名会统一转换成小写形式。



注意：removeAttribute方法不仅会删除元素的特性值，还会删除元素特性。

貌似元素节点的属性可以直接获取、操作，如div.id=?、div.class=?。

（4）元素节点Element是唯一一个使用attributes属性的DOM节点类型（用得较少）。

attributes属性为1个NamedNodeMap，与NodeList类似，也是一个动态的集合。元素的每一个特性都由一个Attr节点表示，每个节点都保存在NamedNodeMap对象中。

attributes属性包含以下四个方法：

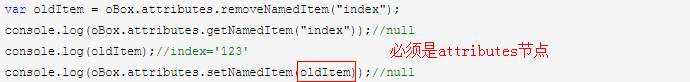
















（5）元素节点其他操作见节点操作。

**2.元素特性节点attribute**

（1）对应网页中HTML标签的属性，尽管特性是节点，它只存在于元素的attributes属性中，并不是DOM文档树的一部分。

（2）特性节点的节点类型nodeType值是2，节点名称nodeName值是属性名，nodeValue值是属性值。



（3）特性节点操作方法不常用，而且直接在元素节点中操作更方便。

**3.文本节点text**

（1）对应网页中的HTML标签中的内容。虽然包含纯文本内容，但却是1个对象。

（2）文本节点的节点类型nodeType值是3。节点名称nodeName值是'#text'，nodeValue值是标签内容值。



（3）文本节点的data属性与nodeValue属性相同。



（4）length属性保存着节点字符的数目，和nodeValue.length、data.length值相同。

（5）createTextNode方法用于创建1个文本节点。



（6）**实际使用中，通常使用innerHTML获取文本**，然后使用字符串方法处理后，在使用innerHTML写回。

**4.文档节点document**

（1）表示HTML文档，即根节点，指向document对象。

（2）文档节点的节点类型nodeType的值为9，节点名称nodeName的值为'#document'，nodeValue的值为null。



（3）document.documentElement属性始终指向HTML页面中的<html>元素。document.body属性指向<body>元素。document.head属性指向文档的<head>元素。title为<title>元素。

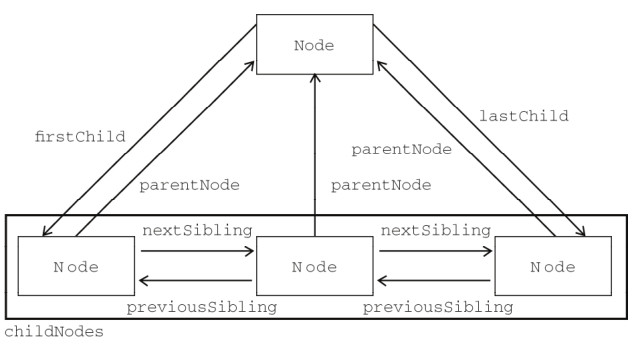


（4）document.URL为页面的完整地址，document.domain为页面的域名，document.referrer为上一个页面的URL（还可以通过window.Location获取）。

（5）write和writeln方法可以在页面被加载的过程中，向页面中动态地加入内容。都接收一个字符串参数，即要写入到输出流中的文本。其中，write会原样写入，而writeln则在字符串的末尾添加一个换行符（每写一个在页面上换一行）。

* + 1. DOM节点关系

节点中的各种关系可以用传统的家族关系来描述，相当于把文档树比喻成家谱。



**1.属性**

（1）parentNode

指向文档树中的父节点，可以为element节点、document节点和documentfragment节点。如果不存在，则返回null。



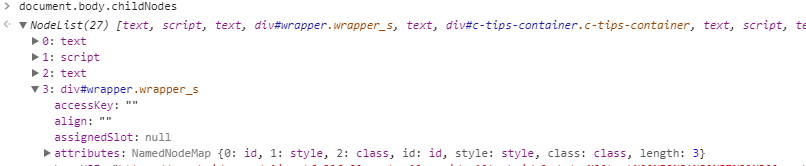
（2）parentElement

返回的是父元素节点（element节点）。



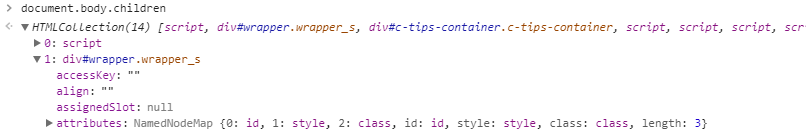
（3）childNodes

返回只读的类数组对象NodeList对象，保存着该节点的第一层子节点。



（4）children

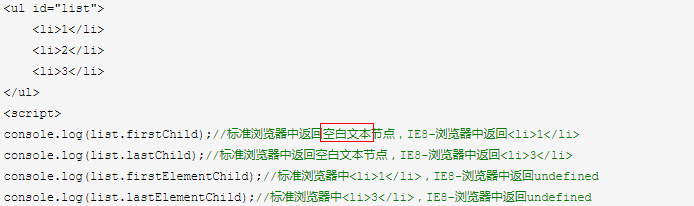
返回只读的类数组对象HTMLCollection对象，保存着该节点的第一层元素子节点（element节点）。



（5）childElementCount

返回子元素节点的个数，相当于children.length。

（6）firstChild、lastChild、firstElementChild、lastElementChild



（7）nextSibling、previousSibling、nextElementSibling、previousElementSibling

同上，nextSibling和previousSibling也包括空白文本节点。

**2.方法**

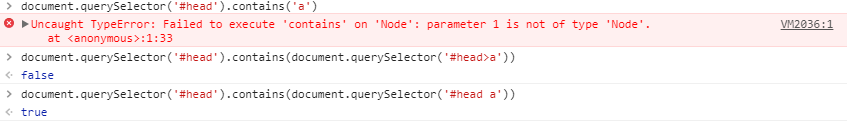
（1）hasChildNodes

是否有子节点，有返回true，比查询childNodes列表的length属性更简单。



（2）contains

接受一个节点作为参数，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。参数为后代节点即可，不一定是第一层子节点。



注意：safari不支持document.contains()方法，只支持元素节点的contains()方法。

（3）isSameNode和isEqualNode

前者表示为相同节点，即引用的同一个节点。后者表示两个节点相等，不一定是同一个节点，但nodeName、nodeValue等相等，childNodes等属性也相等。

* + 1. 节点操作

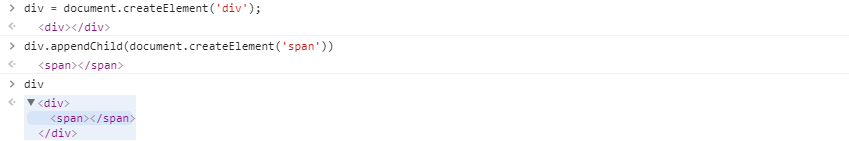
DOM中的节点关系指针都是只读的，如上，不能直接将1个节点的parentNode设置为另外1个节点。操作节点必须通过如下方法。

**1.创建节点**

可以使用createElement创建新元素节点，标签名（必须是标签名）不区分大小写。



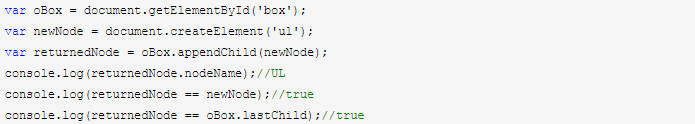
创建的节点即使还没有append到document上，也可以为其添加属性、子节点。



**2.插入节点**

（1）appendChild

向childNodes列表的末尾添加一个节点（注意是节点，不是html字符串），并返回新增节点。



如果插入的节点已经在文档中，将会使该节点从原来的位置转移到新位置。

（2）insertBefore

插入后，被插入的节点会变成参照节点的前一个兄弟节点并返回。如果参照节点是null，则insertBefore与appendChild方法效果相同。如果插入的节点已经在文档中，将会使该节点从原来的位置转移到新位置。



不存在insertAfter方法。

**3.移除节点**

（1）removeChild

入参为要移除的节点，被移除的节点成为方法的返回值。如下，移除1个元素的所有子节点：



（2）remove

不用调用其父节点，直接在当前节点使用remove方法就可以删除该节点，无返回值。



**4.替换节点**



**5.复制节点**

（1）使用cloneNode方法，接受一个布尔值作为参数，表示是否执行深复制。复制后返回的节点副本属于文档所有，但并没有为它指定父节点。

（2）不会复制添加到DOM节点中的js属性，例如事件处理程序等。这个方法只复制特性和子节点，其他一切都不会复制。

* + 1. DOM选择器API

**1.3种元素选择器**

（1）getElementById

id必须唯一，否则只选取第1个。若不存在则返回null。现代浏览器不会将name识别为id。

（2）getElementsByTagName

返回的是包含0或多个元素的类数组对象HTMLCollection。可以使用方括号语法或item方法来访问类数组对象中的项，length属性表示对象中元素的数量。



getElementsByTagName方法既可以用于document对象，也可以用于element元素对象，用于调用该方法的元素的后代元素。

注意，document.all方法已被废弃，要获取文档中的所有元素，可以使用document.getElementsByTagName('\*')方法。

（3）getElementsByClassName

入参为一个包含一个或多个类名的字符串，可以传入多个类名时，类名的先后顺序不重要。返回带有指定类的所有元素的类数组对象HTMLCollection。既可以用于document对象，也可以用于element元素对象，用于调用该方法的元素的后代元素。



**2.h5新增的2种selector方法**

（1）querySelector

接收一个CSS选择符，返回与该模式匹配的第一个元素，如果没有找到匹配的元素，返回null。

貌似各种css3选择器都支持。

（2）querySelectorAll

接收一个CSS选择符，返回一个类数组对象NodeList的实例。没有匹配元素时，返回空的类数组对象，而不是null。

返回的值实际上是带有所有属性和方法的NodeList，而其底层实现则类似于一组元素的快照，而非不断对文档进行搜索的动态查询。这样实现可以避免使用NodeList对象通常会引起的大多数性能问题。

注意：与getElementById和getElementsByTagName方法不同，querySelector和querySelectorAll方法得到的类数组对象是非动态实时的。

尽管selector类方法既可用于文档document类型，也可用于元素element类型。但即使是在元素element上调用时，指定的选择器仍然在整个文档中进行匹配。因此，不要在元素element上使用，以免误解为可以选取该元素的后代元素。

* 1. 视图属性
     1. 概要

**1.三者关系**

（1）元素大小主要有偏移大小offset、滚动大小scroll和客户区大小client。三者关系为：offset为可见的尺寸（包含滚动条的尺寸），scroll通常比offset要大，还包括了因滚动条而没有显示的内容区域大小，client为内容区域（元素自身）的大小，不包含滚动条和边框大小。

（2）client为元素自身的内容区域大小。

有滚动条时，以百度首页为例，设#wrapper的overflow设为scroll，且height为600：





没有滚动条，全部都显示出来时，scrollHeight与clientHeight属性结果相等，scrollWidth与clientWidth属性结果相等。如将#wrapper的overflow设为hidden，有：



即，client为元素自身的内容区域大小，尽管子元素高度为1417，但自身只设置为600。如果有滚动条，那么减去滚动条，就只有583。

（3）存在滚动条时，但元素设置宽高大于等于元素内容宽高时，scroll和client属性的结果相等。如将#wrapper的overflow设为scroll，且height设置1600，有：



即，scroll也只包含内容区域大小。

（4）当滚动条滚动到内容底部时，有以下等式：



**2.其他**

（1）因每次访问客户区offset、client属性都需要重新计算，重复访问需要耗费大量的性能，所以如果需要重复访问，则把它们的值保存在变量中，以提高性能。

（2）所有offset、client属性都是只读。scrollHeight和scrollWidth属性不同的是，scrollLeft和scrollTop是可写的（但通常只对页面元素设置）。

（3）如果给元素设置了display:none，则offset、scroll、client属性都为0。

**3.使用**

（1）尺寸方面

offset没有任何兼容性，获取可见区域尺寸就用它。尽管scrollHeight有兼容性，但没办法，改用还得用。client就是offset去掉滚动条和边框。3者功能不同，都可能用到。

（2）距离方法

获取元素至定位父级距离，显然只会用offset；scroll通常只用页面的scrollTop和ScrollLeft，元素的不会用；而client的距离属性基本上没有用处。

* + 1. 偏移属性

**1.定位父级offsetParent**

（1）与当前元素最近的有定位（position不等于static）的父级元素。

（2）如果元素自身为fixed定位，firefox中其offsetParent<body>，其他浏览器都返回null。

（3）如果元素自身无fixed定位，且父级元素都未经过定位，offsetParent的结果为<body>。

（4）<body>元素的parentNode是null。

**2.4种偏移量属性**

（1）offsetWidth

元素在水平方向上占用的空间大小，无单位（但值以像素px计）。



（2）offsetHeight

元素在垂直方向上占用的空间大小，无单位（但值以像素px计）。



（3）offsetTop

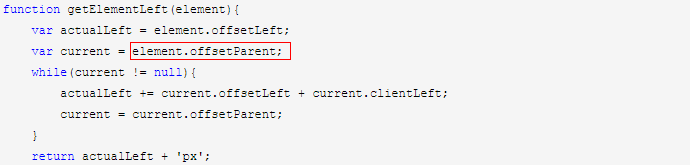
元素的上外边框至offsetParent元素的上内边框之间的像素距离。

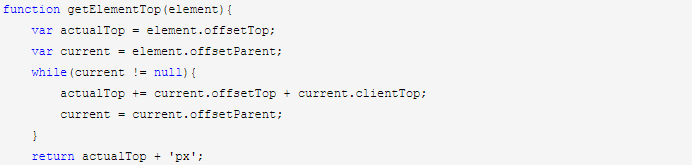
（4）offsetLeft

元素的左外边框至offsetParent元素的左内边框之间的像素距离。

**3.页面偏移**

要知道某个元素在页面上的偏移量，将这个元素的offsetLeft和offsetTop与其offsetParent的相同属性迭代相加，如此循环直到根元素，就可以得到元素到页面的偏移量。





* + 1. 滚动条

**1.概要**

（1）scrollHeight

元素的总高度，包括由于溢出而无法展示在网页的不可见部分。

（2）scrollWidth

元素的总宽度，包括由于溢出而无法展示在网页的不可见部分。

（3）scrollTop

表示被隐藏在内容区域上方的像素数（只是隐藏部分）。元素未滚动时，scrollTop的值为0，如果元素被垂直滚动了，scrollTop的值大于0，且表示元素上方不可见内容的像素宽度。

（4）scrollLeft

表示被隐藏在内容区域左侧的像素数（只是隐藏部分）。元素未滚动时，scrollLeft的值为0，如果元素被水平滚动了，scrollLeft的值大于0，且表示元素左侧不可见内容的像素宽度。

**2.注意事项**

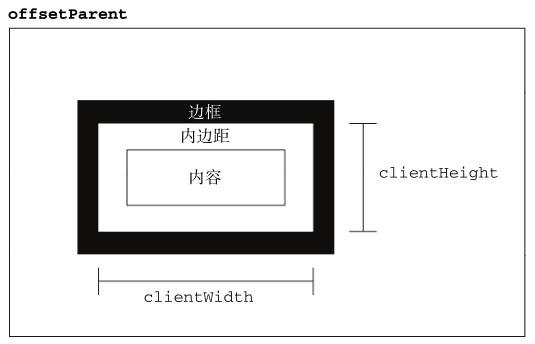
（1）与scrollHeight和scrollWidth属性不同的是，scrollLeft和scrollTop是可写的，但通常也就页面元素用。

（2）scrollHeight属性存在兼容性问题，chrome和safari浏览器中，scrollHeight包含padding-bottom；而ie和firefox不包含padding-bottom，但没有办法。

* + 1. 客户区尺寸

**1.概要**

（1）client指的是元素内容及其内边距所占据的空间大小。



（2）clientHeight：元素节点的客户区高度。



（3）clientWidth：元素节点的客户区宽度。



（4）clientLeft：左边框（border）的宽度。clientTop：上边框（border）的宽度。

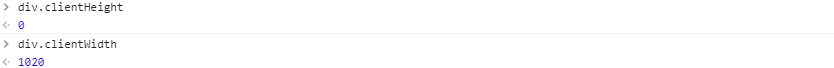
**2.注意**

（1）滚动条宽度、高度都不计算在内。

（2）如果没有高度height值，且设置overflow:scroll，使得滚动条始终存在时。firefox会存在一个最小高度为34px的垂直滚动条，其他浏览器无垂直滚动条。所以，当clientHeight的值小于34px时，firefox会返回34，其他返回0（clientWidth显然继承父元素宽度，始终存在）。

chrome：





firefox：





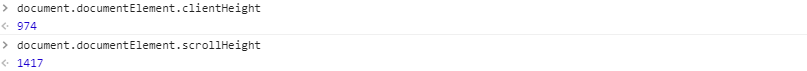
* + 1. 页面

**1.页面大小**

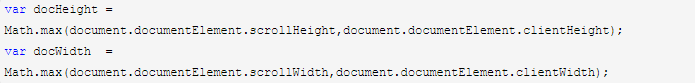
（1）跟offset没关系。只跟client和scroll有关，其中，document.documentElement.clientHeight表示页面的可视区域的尺寸，而document.documentElement.scrollHeight表示html元素内容的实际尺寸。

（2）html元素没有滚动条时（可能内容区更大，但设置了overflow:hidden），safari和chrome表现正常，clientHeight返回可视区域大小，而scrollHeight返回元素内容大小；但IE和firefox的client和scroll属性始终相同，且返回可视区的尺寸大小。

（3）html元素存在滚动条时，各个浏览器都表现正常。clientHeight返回可视区域大小，而scrollHeight返回元素内容大小。



（4）兼容处理（貌似没有用啊）：要取得文档实际内容高度时，要取得<html>元素的scrollHeight和clientHeight的最大值。



（5）注意，client和scroll都不包含滚动条、边框大小。

（6）还可以使用window.innerHeight和innerWidth属性(包含滚动条宽度)获取视口大小，即：浏览器窗口大小减去菜单栏、地址栏等剩余的页面尺寸，由于滚动条是属于页面的，所以包含滚动条。



**2.滚动条偏移**

（1）无论什么浏览器，默认滚动条均来自<html>，而不是<body>。因为<body>元素默认有8px的margin。若滚动条来自<body>元素，则滚动条与页面则应该有8px的间距，实际上并没有间距，所以滚动条来自<html>元素。

（2）可以通过document.documentElement.scrollTop和scrollLeft可以反映和控制页面的滚动；但是chrome和safari浏览器是通过document.body.scrollTop和scrollLeft来控制的。所以，页面的滚动高度兼容写法是：



可以利用scrollTop来实现回到顶部的功能：



（3）现代浏览器还支持以下两个只读属性：

* pageXOffset：表示水平方向上页面滚动的像素值。
* pageYOffset：表示垂直方向上页面滚动的像素值。

（4）现代浏览器支持方法：

* scrollTo(x,y)方法：滚动当前window中显示的文档，让文档中由坐标x和y指定的点位于显示区域的左上角。
* scrollBy(x,y)方法：滚动当前window中显示的文档，x和y指定滚动的相对量。
* Element.scrollIntoView方法：滚动当前元素，进入浏览器的可见区域。接受一个布尔值作为参数。如果为true，表示元素的顶部与当前区域的可见部分的顶部对齐（前提是当前区域可滚动）；如果为false，表示元素的底部与当前区域的可见部分的尾部对齐（前提是当前区域可滚动）。

**3.获取滚动条自身宽度**

可以新建1个元素，设overflow:scroll，添加页面中，然后获取这个元素的clientWidth。



**4.scroll事件**

scroll事件可以用在有滚动条的元素上，但通常也就window需要。

* + 1. 元素视图方法

（1）Element.getBoundingClientRect()方法返回一个对象，该对象提供当前元素节点的大小、它相对于视口(viewport)的位置等信息。不同浏览器返回有差异，不过因为是对象，不影响使用：







（2）返回的宽高是偏移宽高offset，top、left、right、bottom这四个值均相对于视口距离（注意视口和页面的区别，不是返回页面，要获取页面距离，还得用offset属性）。

（3）现代浏览器均支持。

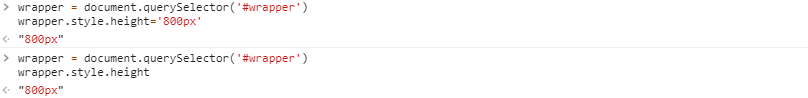
* 1. 元素样式

**1.行内样式**

（1）element元素节点提供style属性，用来操作行内样式。style属性指向cssStyleDeclaration对象。

（2）注意，style属性获取和设置的都是style这个属性中的，即，即使元素有高度，但style中没有height这个样式，那么仍然返回空字符串。

（3）返回和设置都带单位。



（4）如果带连字符，需要转为驼峰式。float最好写作cssFloat，因为float是js关键字，不过貌似直接写也没问题。

（5）使用style.cssText可以获取、设置style特性全部代码（前面的只是1个样式）。注意，这样设置会去掉之前的样式，如下，style内只有width了。



（6）style.length返回样式个数。

（7）css样式模块侦测：利用style对象的某个属性值是否为字符串（不是undefined）。

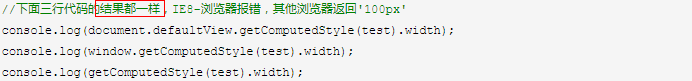


**2.获取计算样式**

（1）因样式层叠原因，可以通过getComputedStyle方法获取元素时实际使用的属性值。该属性为只读。

（2）接收两个参数：要取得计算样式的元素和一个伪元素字符串，目的是获取该元素的伪元素样式。如果不需要伪元素信息，第二个参数可以是null。

（3）getComputedStyle()方法返回一个CSSStyleDeclaration对象，其中包含当前元素的所有计算的样式





（4）对于font、background、border等复合样式，各浏览器处理不一样。chrome会返回整个复合样式，而IE9+、firefox和safari则输出空字符串。

（5）不论以什么格式设置颜色，浏览器都以rgb()或rgba()的形式输出。

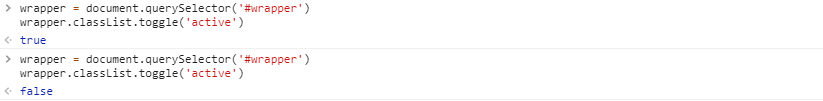
（6）在计算样式中，类似百分比等相对单位会转换为绝对值。

特别注意：类似padding、width等单一样式并不直接反映元素的位置和尺寸信息，这些信息是多个样式综合作用的结果。所以，最好使用前面介绍过的关于元素视图的offset、client、scroll和getBoundingClientRect()等来获取。

**3.css样式操作**

（1）改变元素的少部分样式时，一般会直接改变其行间样式style。但改变大量样式，可以使用CSS类处理。

（2）可以使用classList改变多个类名。

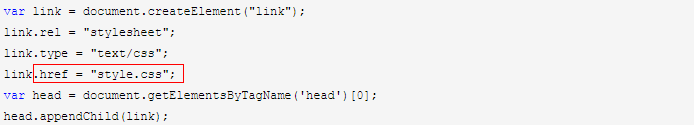


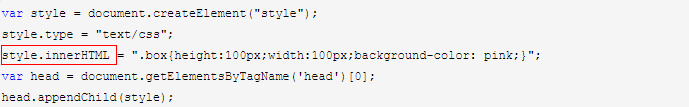
**4.脚本化样式表**

内部样式和外部样式可以分别通过<style>和<link>标签以样式表的形式引入，可以通过document.styleSheets或CSSStyleSheet操作。

**5.动态样式**

（1）所谓动态样式，是指在页面加载时并不存在，在页面加载完成后动态添加到页面的样式。包括两种情况：一种是通过<link>元素插入外部样式表，另一种是通过<style>元素插入内部样式。



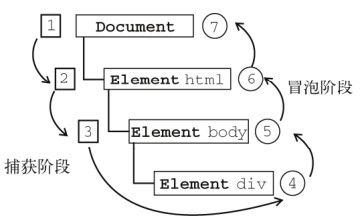


* 1. 事件
     1. 基础

**1.事件流**

（1）又叫事件传播，描述的是从页面中接收事件的顺序。

（2）DOM2级事件规定的事件流包括三个阶段：事件捕获阶段(capture phase)、处于目标阶段(target phase)和事件冒泡阶段(bubbling phase)。



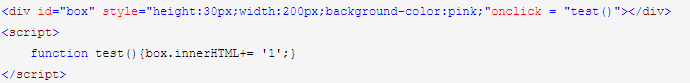
**2.事件处理程序**

（1）又叫事件侦听器，实际上就是事件的绑定函数。有HTML事件处理程序、DOM0级事件处理程序、DOM2级事件处理程序和IE事件处理程序四类。

（2）HTML事件处理程序

如下，其中，this值等于事件的目标元素。事件名要加on。



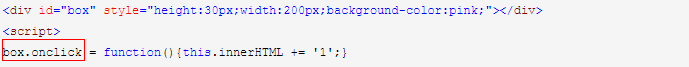


如上面的事件处理程序，浏览器会自动创建一个封装着元素属性值（即里面的代码）的函数，并传入事件对象event。

存在耦合问题等。

（3）DOM0级事件处理程序

将一个函数赋值给一个事件处理程序属性。事件处理程序属性通常全部小写。事件名要加on。



可以通过将事件处理程序属性设置为null来删除事件处理程序。



缺点：只能添加一个事件处理程序。

（4）DOM2级事件处理程序

即，使用addEventListener()和removeEventListener()管理。它们都接受3个参数：要处理的事件名、作为事件处理程序的函数和一个布尔值。最后的布尔值参数如果是true，表示在捕获阶段调用事件处理程序；如果是false，表示在冒泡阶段调用事件处理程序。默认为false。

因移除时传入的参数与添加处理程序时使用的参数相同。这意味着，addEventListener()添加的匿名函数将无法移除。

优点是可以添加多个事件处理程序，并按照他们添加的顺序触发。

（5）同时出现HTML事件处理程序和DOM0级事件处理程序，DOM0级会覆盖HTML事件处理程序。

**3.事件对象**

（1）event对象默认是事件程序的第一个参数。

（2）type属性表示被触发的事件类型，如mouseover、click等。

（3）事件目标，有currentTarget和target。其中，currentTarget表示当前所在的节点，即正在执行的监听函数所绑定的那个节点，与事件中的this指向相同。target属性返回事件的实际目标节点。

（4）事件冒泡相关属性、方法：1）bubbles属性返回一个布尔值，表示当前事件是否会冒泡。该属性为只读属性。2）stopPropagation()方法表示取消事件的进一步捕获或冒泡，无返回值。3）stopImmediatePropagation()方法不仅可以取消事件的进一步捕获或冒泡，而且可以阻止同一个事件的其他监听函数被调用，无返回值。

（5）eventPhase返回一个整数值，表示事件目前所处的事件流阶段，其中，0表示事件没有发生，1表示捕获阶段，2表示目标阶段，3表示冒泡阶段。

（6）取消默认行为：1）cancelable属性返回一个布尔值，表示事件是否可以取消。该属性为只读属性。返回true时，表示可以取消。否则，表示不可取消。2）preventDefault()方法取消浏览器对当前事件的默认行为，无返回值。3）defaultPrevented表示默认行为是否被阻止，返回true时表示被阻止，返回false时，表示未被阻止。

**4.鼠标事件对象**

（1）clientX/Y：鼠标指针在可视区域中的水平和垂直坐标。

（2）screenX/Y：表示鼠标指针相对于屏幕的水平和垂直坐标。

（3）pageX/Y：表示相对于页面的水平和垂直坐标，与clientX/clientY的区别是不随滚动条的位置变化。

（4）offsetX/Y：表示相对于定位父级的水平和垂直坐标（Firefox不支持）。

**5.键盘事件对象**

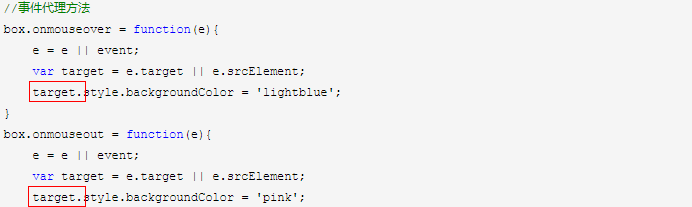
（1）有三种按键事件类型，即keydown，keypress和keyup。

* + 1. 高级操作

**1.事件代理**

（1）由于事件会在冒泡阶段向上传播到父节点，因此可以把子节点的监听函数定义在父节点上，由父节点的监听函数统一处理多个子元素的事件。这种方法叫做事件的代理，也叫事件委托。

（2）事件代理应用事件目标的target属性完成。利用事件代理，可以提高性能及降低代码复杂度。



（3）如果可行的话，也可以考虑为document添加一个事件处理程序。好处在于document对象很快就可以访问，而且可以在页面生命周期的任何时间点上为它添加事件处理程序，而无需等待DOMContentLoaded或load事件。

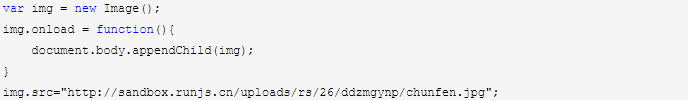
**2.一次性事件**

触发事件之后，立即解绑事件处理程序。

**3.加载事件**

（1）load事件是最常用的一个事件，当页面完全加载后（包括所有图像、javascript文件、CSS文件等外部资源），就会触发window上的load事件。

（2）图像和框架iframe也可以触发load事件。



（3）与load事件不同，DOMContentLoaded事件则在形成完整的DOM树之后就会触发，而不理会图像、javascript文件、CSS文件或其他资源是否下载完毕。注意，网页的javascript脚本是同步执行的，所以定义DOMContentLoaded事件的监听函数，应该放在所有脚本的最前面。否则脚本一旦发生堵塞，将推迟触发DOMContentLoaded事件。

**4.自定义事件**

（1）早期自定义事件创建、触发过程为create-init-dispatch三步骤，已被废弃。



其中，createEvent、initEvent、dispatchEvent方法定义如下：



type值有5种：1）Events：包括所有的事件；2）HTMLEvents：包含如change、blur、scroll等事件；3）MouseEvents：鼠标事件，如click、mousemove等；4）UIEevents：如keydown等；5）MutationEvents。

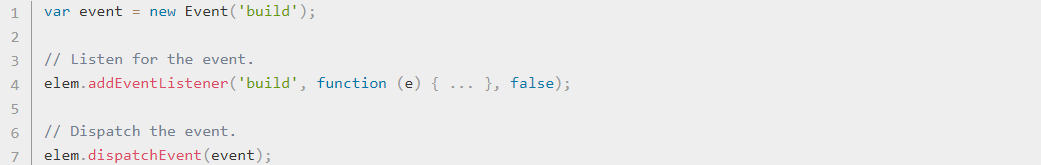


type为自定义事件的名称。bubbles为Boolean值，决定是否事件是否应该向上冒泡。cancelable为Boolean值，决定该事件的默认动作是否可以被取消。



event为被派发的事件对象。target为用来初始化事件和决定将会触发目标。如果事件事件句柄中调用了preventDefault，则返回false，否则返回true。

（2）现代浏览器（ie9都支持）都支持新的事件自定义方法。



如果想要给自定义事件对象传参，可以使用CustomEvent方法。



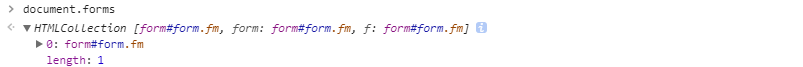
在addEventListener中添加的事件句柄内，可以获取event事件对象，并可以使用事件对象的preventDefault等方法。

* 1. 表单对象

**1.概要**

（1）表单里嵌套表单是不允许的。

（2）可以通过document.forms获取页面所有表单对象，类型为HTMLCollection。



（3）form元素的action和name属性为必需项。如果name="test"，则Javascript可以使用document.forms.test来获取该表单。

（3）每个表单都有elements属性，该属性是表单中所有元素的集合。



（4）还可以通过form[name]来获取表单内的元素控件。



（5）表单有reset和submit事件。



（6）还可以使用表单方法reset和submit重置和提交表单。



**2.表单元素**

（1）除了<fieldset>元素之外，所有表单字段都拥有相同的一组属性：name、value、type、disabled、readOnly、defaultValue等。

（2）每个表单字段都有两个方法：focus()和blur()。

（3）所有表单字段都支持下列3个事件：focus、blur、change。

**3.焦点管理**

（1）可以使用document.activeElement获取当前获得焦点的元素。

（2）用户通常可以使用tab键移动焦点，使用空格键激活焦点。可以使用tabindex设置tab激活焦点顺序。



（3）focus()方法用于将浏览器的焦点设置到表单字段，即激活表单字段，使其可以响应键盘事件。



（4）blur()方法的作用是从元素中移走焦点。

（5）焦点事件focus、blur等。

1. BOM

**1.概要**

（1）BOM全称brower object model（浏览器对象模型），用于管理窗口及窗口间的通讯，其核心对象是window。称其为窗口，可能并不准确。因为，有的浏览器窗口可能包含多个标签页，每个标签页都有自己的window对象。

（2）窗口大小：outerWidth和outerHeight属性用于表示浏览器窗口本身的尺寸；innerWidth和innerHeight属性用于表示页面大小，实际上等于浏览器窗口尺寸大小减去浏览器自身边框及菜单栏、地址栏、状态栏等的宽度。



（3）由于<iframe>本身也有window属性，如果页面中存在框架，那么框架中的innerWidth和innerHeight属性指的是框架本身的innerWidth和innerHeight属性。

（4）window.open()方法可以导航到一个特定的URL，也可以打开一个新的浏览器窗口。返回值是新窗口的Window对象。方法close()将关闭一个窗口。



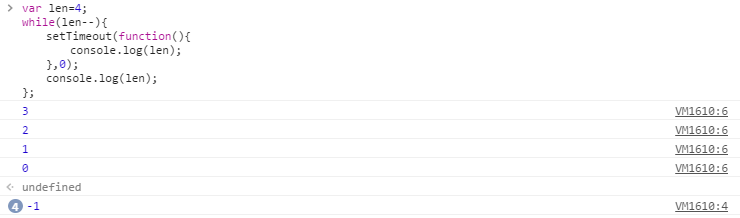
**2.对话框**

（1）confirm：若是true表示单击OK，false表示单击Cancel或者右上角的关闭按钮。

（2）window.print()方法可以用来显示打印对话框。

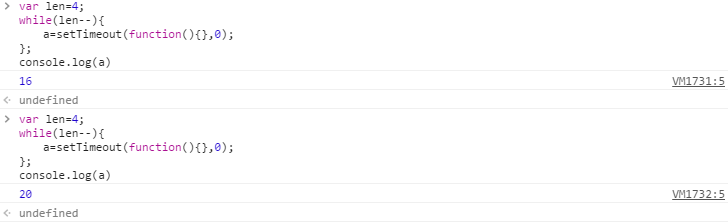
**3.定时器**

（1）setTimeout即使设置延迟为0，也会在其他语句执行后执行。如：



先执行console.log(len)，再执行注册的setTimeout。

（2）setTimeout返回的是1个id，且每次增加1。将该整数传入clearTimeout函数，取消对应的定时器。

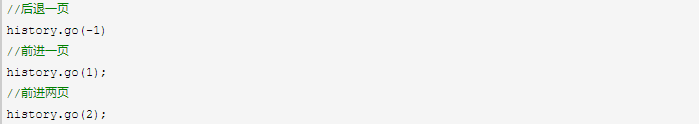


（3）setTimeout和setInterval中的this均指向window，需要处理。

**4.history对象**

（1）history.length属性保存着历史记录的URL数量。

（2）history对象提供了一系列方法，允许在浏览历史之间移动，包括go()、back()和forward()。



go()方法无参数时，相当于history.go(0)，可以刷新当前页面。



（3）HTML5为history对象新增history.pushState()和history.replaceState()，用来在浏览历史中添加和修改记录。



**4.screen对象**

可从screen对象属性中获取屏幕（显示器）的信息。

（1）width：屏幕的像素宽度。

（2）height：屏幕的像素高度。

（3）availWidth：显示器可用宽度，为屏幕的像素宽度减去任务栏等系统部件宽度之后的值。

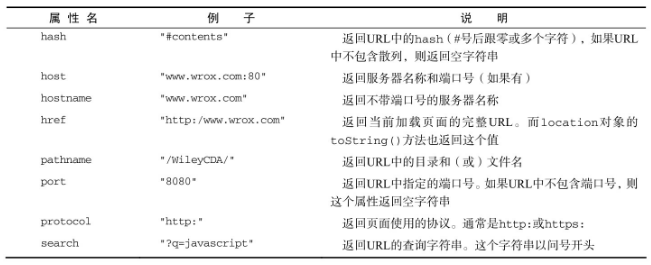
（4）availHeight：显示器可用高度，为屏幕的像素高度减去任务栏等系统部件高度之后的值。

**5.navigator对象**

用于获取浏览器版本等信息。

**6.location对象**

location将URL解析成独立的片段。



1. ajax
   1. ajax

**1.ajax历史**

在ajax技术出现以前，有如下方式实现类似功能：

（1）利用http协议204特性，链接到后台文档，后台接收到请求后回复header('HTTP/1.1 204 No Content')。浏览器接收到204后不会跳转。

（2）利用图片src加载特性。

（3）利用css/js加载特性，如图片src。

（4）利用iframe特性，创建1个form，将form的target设为iframe的name，并在action中设置对于操作。iframe需要设为长宽边框0。

**2.原理**

（1）核心是XMLHttpRequest对象，1个专门发送http请求的js对象。

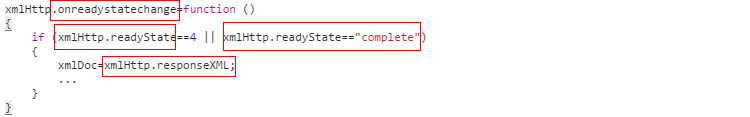
（2）创建XMLHttpRequest对象。



（3）利用XMLHttpRequest对象发送请求，需要明确：（1）用什么方法GET/POST/DELETE等。（2）请求资源url。（3）同步/异步。



（4）获取服务器返回的数据：通过绑定监听函数获取服务器端返回的数据。



XMLHttpRequest对象有onReadyStateChange事件属性，当每次状态改变时会自动触发该事件函数，在该函数中可以通过XMLHttpRequest对象的responseXML或responseText获取返回值（只有这两个属性可以获取）。前者获取的是xml，后者获取的是text。服务器返回数据时需要在header中指明Content-Type为text/xml还是text/html。

**3.属性**

（1）status：http状态码。

（2）statusText：http返回状态文字描述。

（3）readyState：XHR对象自身的状态码（0：创建XHR对象成功。1：打开url成功，此时可以setRequestHeader，对应open方法。2：对方接收到header，对于send方法。3：对方收到主题信息。4：成功，收到结果，断开连接）。

（4）onReadyStateChange：事件属性，当状态发生变化时，触发事件函数。

（5）responseText：服务器返回的主题信息。

（6）responseXML：服务器响应的以XML形式的主题信息。

**4.方法**

（1）open(请求方式，url，同步/异步)。

（2）send(null/参数)，参数为k1=v1&k2=v2。

（3）setRequestHeader(key,value)：设置请求头。

（4）abort：中断请求。

（5）getResponseHeader（key）：获取相应头信息。

（6）getAllResponseHeader：获取所有响应头信息。

**5.返回类型**

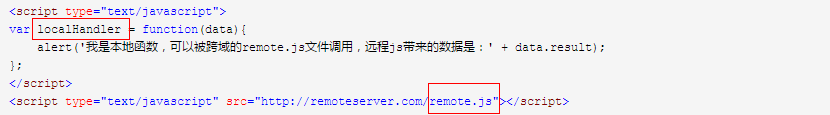
（1）不考虑h5，返回类型只有XML和文本文档2种形式。

（2）默认返回text/html。

* 1. jsonp

利用web页面上调用js文件（src）时则不受是否跨域的影响，在服务器上设法把数据装进js格式的文件里，供客户端调用和进一步处理。JSONP是一种非正式传输协议，核心就是允许用户传递一个callback参数给服务端，然后服务端返回数据时会将这个callback参数作为函数名来包裹住JSON数据，这样客户端就可以随意定制自己的函数来自动处理返回数据。

**1.原理举例1**

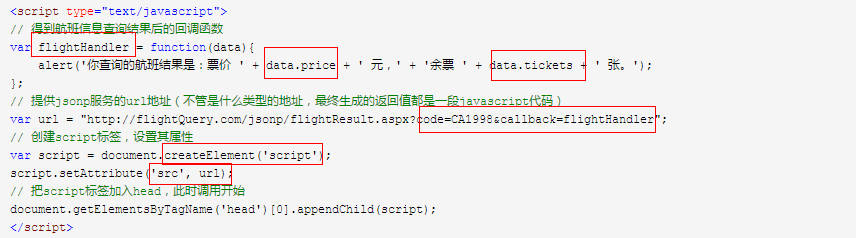


remote.js文件代码为：

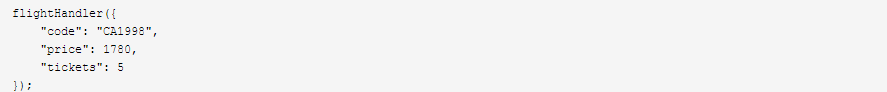


从而实现本地函数被跨域的远程js调用成功，并且还接收到了远程js带来的数据。

**2.原理举例2**



传递给服务器callback的目的是告诉服务器端本地方法的名称。服务器端返回：



**3.jquery中使用jsonp**

