# DOM

浏览器每次加载页面时，都在内存里创建页面及其全部元素的一个内部表示体系，也就是DOM。在DOM里，页面的元素具有一个逻辑化，层次化的结构就像一个由父对象和子对象组成的树形结构。每个对象都有“属性”列表来描述它，而利用JavaScript可以使用一些方法来操作这些属性。

这个层级树的最顶端是浏览器window对象，它是DOM树里一切对象的根。

window对象具有一些子对象：document对象，location对象，history对象

## 全局作用域

由于window对象同时扮演着ECMAScript中Global对象的角色，因此所有在全局作用域中声明的变量、函数都会变成window对象的属性和方法。来看下面的例子。

<script>  
 var *age* = 29;  
 function sayAge(){  
 console.log(this.*age*);  
 }  
 alert(*window*.*age*); *// 29* sayAge(); *// 29  
 window*.sayAge(); *// 29*</script>

## 弹窗

### window.alert()

警告框

alert()方法，这个方法接受一个字符串，并将其显示给用户。具体来说，调用alert()方法的结果就是向用户显示一个系统对话框，其中包含指定的文本和一个OK（“确定”）按钮。

### window.confirm()

确认框

为了确定用户是单击了OK还是Cancel，可以检查 confirm() 方法返回的布尔值：true 表示单击了OK，false 表示单击了Cancel或单击了右上角的 X 按钮。确认对话框的典型用法如下。

<script>  
 if (confirm("你确定吗？")) {  
 alert("很高兴你确定了")  
 } else {  
 alert("很遗憾你不确定")  
 }  
</script>

这种模式经常在用户想要执行删除操作的时候使用，例如删除电子邮件。

### window.prompt()

提示框

显示一个对话框，对话框中包含一条文字信息，用来提示用户输入文字。

语法：

result = window.prompt('text','default value');

result 用来存储用户输入文字的字符串，或者是 null。  
text 提示文字，字符串格式，如果没有任何提示内容，该参数可以省略不写。  
value 文本输入框中的默认值，该参数也可以省略不写。当用户点击"确定"按钮后，文本输入框中的文字被返回。如果文本输入框中为空,则返回一个空字符串。如果用户点击"取消"按钮，则返回null。

## 定时器

### window.setInterval()

setInterval() 方法按照指定的时间间隔重复执行代码，直至间歇调用被取消或者页面被卸载。 它返回一个唯一标识该定时器的ID，因此您以后可以通过调用clearInterval（）将其删除。

语法

var intervalID = scope.setInterval(func, delay,);

参数

func要重复调用的函数。

delay

是每次延迟的毫秒数 (一秒等于1000毫秒)，函数的每次调用会在该延迟之后发生。

返回值

返回的intervalID是一个非零的数字值，它标识通过调用setInterval（）创建的计时器。

### window.setTimeout()

setTimeout()方法设置一个定时器，该定时器在定时器到期后执行一个函数或指定的一段代码。

语法  
var timeoutID = scope.setTimeout(function[, delay]);

delay 可选  
延迟的毫秒数 (一秒等于1000毫秒)，函数的调用会在该延迟之后发生。如果省略该参数，delay取默认值0，意味着“马上”执行，或者尽快执行。

返回值   
返回值timeoutID是一个正整数，表示定时器的编号。这个值可以传递给clearTimeout()来取消该定时器。

超时调用的代码都是在全局作用域中执行的，因此函数中this的值指向window对象

## location对象

location对象包含着当前页面URL的全部信息。

页面的URL是由多个部分组成的：

[协议]//[主机名]：[端口]/[路径][搜索][hash]  
如：http://www.test.com:8080/tools/display.php?section=435#list

### 属性

location对象的一系列属性包含了URL各个部分的数据：

location.href   
返回当前加载页面的完整URL。

例子：<http://127.0.0.1:5500/test.html>http://www.test.com:8080/tools/display.php?section=435#list

location.protocol  
'http'  
返回页面使用的协议。通常是 http: 或 https:

location.host  
'www.test.com:8080'  
返回服务器名称和端口号（如果有）

location.hostname  
'www.test.com'  
返回不带端口号的服务器名称(主机地址）

location.port  
'8080'  
返回 URL 中指定的端口号。如果 URL 中不包含端口号，则这个属性返回空字符串

location.pathname  
'/tools/display.php'  
返回URL中的目录（路径名）和（或）文件名

location.search  
'?section=435'  
返回URL的查询字符串。这个字符串以问号开头

location.hash  
'#list'  
返回 URL 中的 hash（#号后跟零或多个字符），如果 URL 中不包含散列，则返回空字符串

### 导航至新页面

如下所示：

location.assign("http://www.baidu.com");

location.href = "http://www.jquerycn.cn";

## history对象

它基本上就是访问过的URL列表。这个对象的方法让我们能够使用这个列表，但不能直接的修改这些URL。

history对象包含浏览器以前访问的页面地址。

### 属性

这个对象只有一个属性，就是它的长度，表示用户访问过的页面的数量。包括当前加载的页  
history.length 返回一个整数

### 方法

history.back()  
前往上一页, 等价于 history.go(-1)  
相当于点击浏览器的“后退”按钮

history.forward()  
前往下一页，等价于 history.go(1)  
相当于点击浏览器的“前进”按钮

window.history.go(num)  
它有一个参数，可以为正的或负的整数，可以跳到历史记录列表里的相对位置：  
history.go(-3); 回退3个页面  
history.go(2); 前进2个页面

## document对象

### 节点类型

#### nodeType

每个节点都有一个nodeType属性，用于表明节点的类型。

最常用的节点类型是1,2和3，也就是元素节点、它们的属性节点和文本节点

#### nodeName和nodeValue属性

要了解节点的具体信息，可以使用nodeName和nodeValue这两个属性，这两个属性的值完全取决于节点的类型。

从存在方式来看，文本节点总是包含在元素节点里的，但不是每个元素节点都包含文本节点。

nodeName属性以字符串形式返回节点的名称。这个属性是只读的，不能修改它的值。  
当nodeName返回元素名称时，并不包含HTML源代码里使用的尖括号<>

节点类型 nodeName nodeType nodeValue  
元素 元素（标签）名 1 null  
属性 属性名称 2 属性值  
文本 #text 3 文本内容  
注释 #comment 8 注释内容  
文档 #document 9 null

#### 文档类型节点

<script>  
 console.log(*document*.nodeType)  
 console.log(*document*.nodeName)  
 console.log(*document*.nodeValue)  
</script>

### 获取元素节点

#### querySelector()

element = document / parentNode.querySelector(selector);

返回匹配指定选择器的元素节点对象，它只返回第一个节点对象。如果没有匹配则返回null。  
**前面不必是document**

#### querySelectorAll()

elementList = document / parentNode.querySelectorAll(selectors);  
返回文档中匹配指定 CSS 选择器的所有元素，

#### getElementById()

前面必须是document ；小括号中必须是id的名称，不带#号  
如果找到相应的元素则返回该元素的节点对象；如果不存在，则返回null

#### getElementsByTagName()

前面不必是document；小括号中必须是标签名称；  
根据给定的标签名返回所有的元素节点集合，如果没有搜索到元素则这个集合为空

#### getElementsByClassName()

根据给定的class名称返回所有的节点集合  
当在document对象上调用时，会搜索整个DOM文档，包含根节点。

### 通过层次关系获取节点

#### childNodes

childNodes  
只读，子节点集合（包括元素节点+非元素节点）

#### firstChild

firstChild （只读）第一个子节点（包括元素节点+非元素节点）。

#### lastChild

lastChild（只读）最后一个子节点（包括元素节点+非元素节点）

#### previousSibling

previousSibling（只读）上一个兄弟节点（包括元素节点+非元素节点）

#### nextSibling

nextSibling（只读）下一个兄弟节点（包括元素节点+非元素节点）

#### parentNode

parentNode父节点，只读

#### ownerDocument

ownerDocument 获取某个节点的文档根节点，即#document

#### hasChildNodes()

语法：  
element.hasChildNodes()

方法返回一个布尔值,表明当前节点是否包含有子节点.

有三种方法可以判断当前节点是否有子节点。

node.firstChild !== null  
node.childNodes.length > 0  
node.hasChildNodes()

### 通过层次关系获取元素节点

#### children

children  
只读，元素类型的子节点集合

#### firstElementChild

（只读）第一个元素类型的子节点。相当于children[0]

#### lastElementChild

（只读）最后一个元素类型的子节点。

#### previousElementSibling

（只读）上一个元素类型的兄弟节点

#### nextElementSibling

（只读）下一个元素类型的兄弟节点

#### 获取已定位的上级节点

元素.offsetParent

如果上级元素没有定位，父节点是body

如果上级元素有定位，则父节点offsetParent是离元素最近的一个有定位属性的上级节点

### 创建节点

#### createElement()

document.createElement(标签名称) ，前面必须是document，不能是其他元素，如 mydiv.createElement(标签名称) 这样写是错误的。

#### createTextNode()

类似于createElement()，但它的参数不是nodeName，而是元素需要的文本内容：

var newTextNode = document.createTextNode("这里是一些文本内容")

#### cloneNode()

如果文档中已有的节点与需要新建的节点很相像，就可以使用cloneNode()来新建节点。

该方法以一个布尔值作为参数，要复制的标签.cloneNode(这里填true或false)

var myDiv = document.querySelector("id1");  
var newDiv = myDiv.cloneNode(true);

参数true，复制节点和它的全部子节点  
参数false ，只复制节点，不包括它的子节点

### 操作子节点

#### appendChild()

将获取的节点或创建的节点添加到指定父节点的子节点列表末尾。

#### insertBefore()

这个方法有两个参数：要插入的新节点，指示插入位置的节点（插入到这个节点前面）

语法：  
parentNode.insertBefore(newNode, Node);

#### replaceChild()

可以把父元素现有的一个子节点替换为另一个节点。它有两个参数：一个是新的子节点，一个是现有子节点。

父节点.replaceChild（newNode，oldNode）

#### removeChild()

专门用于从DOM树里删除子节点；返回的是已删除的节点：

父节点.removeChild(子节点)

### 操作元素节点的属性

#### 点和中括号方法

HTML元素默认的标准属性，可以通过点方式或中括号[]方式访问或修改

obj.id 元素节点的 id名称  
obj.title 元素节点的title属性值

#### DOM方法

操作属性的DOM方法主要有3个：

获取属性：obj.getAttribute('属性名称')  
返回属性值，是一个文本字符串

设置属性：obj.setAttribute('属性名'，'属性值')

移除属性：obj.removeAttribute('属性名称')

### 操作元素节点的内容

#### innerHTML

可读可写

element.innerHTML

#### innerText

element.innerText

可读可写

#### value

element.value

## 操作元素节点的CSS

### style对象

格式：Element.style.css属性   
必须驼峰式命名，如：fontSize ,不能 font-size

这种方法不会清空对象原有的style样式，会追加  
实际上是加了个行内式，只能读取行内式的css属性值，这种方法无法读取内部样式表里面的css属性

示例：  
Element.style.styleFloat = 'right'; Element.style.color = 'red';   
Element.style.border = '2px solid green';

格式：Element.style.cssText   
实际上是加了个行内式，这种方法无法读  
这种方法会清空对象原有的style样式，不会追加

示例：  
Element.style.csstext = 'width:200px;height:200px;';

### getComputedStyle方法

使用style的方法只能获取行内式的css属性，如何获取样式表里面的css属性？

var style = window.getComputedStyle（element ）;

示例：

console.log(getComputedStyle(box1).backgroundColor)  
console.log(getComputedStyle(box1).width)

### className

className可读可写。使用className方式会清空对象原有的class，不会在原有的class上面追加

格式：Element.className   
示例：Element.className = 'bgcolor'

### classList

#### length

如果类属性未设置或为空，那么 elementClasses.length 返回 0。虽然

#### item()

item ( Number )

按集合中的索引返回类值。

示例：

box1.classList.item(1)

#### add()

add( String [, String] )

添加指定的类值。如果这些类已经存在于元素的属性中，那么它们将被忽略。

示例：

box1.classList.add('mycolor')

div.classList.add("foo", "bar", "baz");

#### remove()

remove( String [,String] )

删除指定的类值。

示例：

box1.classList.remove('mycolor')

div.classList.remove("foo", "bar", "baz");

#### toggle()

toggle ( String )

即如果类存在，则删除它并返回false，如果不存在，则添加它并返回true。

#### contains()

contains( String )

检查元素的类属性中是否存在指定的类值。

示例：

console.log(elment.classList.contains("foo"));

#### replace()

replace( oldClass, newClass )

用一个新类替换已有类。

示例：

elment.classList.replace("foo", "bar");

## 元素节点的其他属性

### 获取元素的宽高

（返回的都是数字，都不带单位）

Element.clientWidth = 自身样式宽 + padding   
Element.clientHeight = 自身样式高 + padding

Element.offsetWidth = 自身样式宽 + padding + border  
Element.offsetHeight = 自身样式高 + padding + border

Element.scrollWidth = 自身样式宽 + padding + 溢出部分宽度  
Element.scrollHeight = 自身样式高+ padding + + 溢出部分高度

### 获取元素到定位上级的距离

元素.offsetLeft[Top]

只读属性 当前元素到定位上级的距离

如果没有定位上级，offsetLeft[Top]是到 html 的距离。  
如果有定位上级，offsetLeft[Top]是到 定位上级 的距离。

### 滚动条的位置

可读可写

scrollTop滚动条的垂直位置

scrollLeft ：滚动条的水平位置

IE,FF文档的滚动条的位置: document.documentElement.scrollTop

如何让滚动条始终显示在最底部：

mybox.scrollTop = mybox.scrollHeight - mybox.clientHeight;