## 02.js笔记

**（一）获取窗口属性，获取dom尺寸**

理解即可，重于封装库：

1、滚动轮滚动的距离

a) 查看滚动条的滚动距离：

window.pageXOffset/window.pageYOffset

//bom操作，IE8及IE8以下不兼容

document.body.scrollLeft/scrollTop

document. documentElement.scrollLeft/scrollTop

// 兼容性比较混乱，同时取两个值相加，因为不可能存在两个同时有值

b) 封装兼容性方法，求滚动轮滚动距离getScrollOffset()

2、窗口的可视区域尺寸

a) 查看视口的尺寸(会受窗口缩放影响)

window.innerWidth/window.innerHeight

// bom操作，IE8及IE8以下不兼容

document.documentElement.clientWidth/clientHeight

//标准模式下(<!DOCTYPE html>，渲染模式)，任意浏览器都兼容

document.body.clientWidth/clientHeight

//适用于怪异/混杂模式下的浏览器，即语法向后兼容

b) 封装兼容性方法，返回浏览器视口尺寸getViewportOffset()；提示：document. compatMode可区分标准模式与否。

3、元素的几何尺寸

a) 查看元素的几何尺寸，隐藏状态下获取不到相关信息

Element.prototype.getBoundingClientRect();

//兼容性很好

//该方法返回一个对象，对象里边有 left,top,right,bottom等属性。left和 top代表该元素左上角的 X和 Y坐标，right和 bottom代表元素右下角的 X和 Y坐标

//返回的对象里边还有属性x,y，表示的是元素左顶点相对于页面body左顶点的距离

// height和 width属性老版本 IE并未实现

//返回的结果并不是“实时的”,方法每一次调用都返回一个新的对象

b) 查看元素的几何尺寸，隐藏状态下获取不到相关信息

dom.offsetWidth，dom.offsetHeight

//获取视觉高宽（width+paddingLeft+paddingRight+boderLeft+boderRight）,可完全替代getBoundingClientRect()方法

4、元素的位置

a) 查看元素的位置

dom.offsetLeft，dom.offsetTop

//对于无定位父级的元素，返回相对文档的坐标；对于有定位父级的元素返回相对于最近的有定位的父级的坐标。注意：与自己是否是定位元素无关。

dom.offsetParent

//返回最近的定位父级元素，如无，返回body元素，document.body.offsetParent为 null

b) 封装一个求元素相对于文档的坐标getElementPosition

//作业：封装一个求元素相对于文档的坐标getElementPosition

5、让滚动条滚动

a) window上有三个方法：scroll(x,y)/scrollTo(x,y) scrollBy(x,y);

//前两个方法功能类似，用法都是将x,y坐标传入，实现让滚动轮滚动到指定位置。

//而scrollBy() 会在之前的数据基础上做累加。

b) scrollBy(x,y)：利用scrollBy(x,y)实现快速阅读(自动阅读)的功能；

scroll(x,y)/scrollTo(x,y)：利用scrollTo(x,y)多用于“查看更多-点击收起”。

6、让元素在视口中可见

elementNode.scrollIntoView(boolean)

1. 默认为true，将元素和视口的上边缘对齐;
2. 如果传递参数false，则将元素的下边缘和视口的下边缘对齐；

//小练习：点击链接，将页面内的元素移至可视化区域

<div class="menu">

<a href="###" goto="info">公司简介</a>|

<a href="###" goto="products">产品信息</a>|

<a href="###" goto="contact">联系方式</a>|

<a href="###" goto="hr">招聘信息</a>

</div>

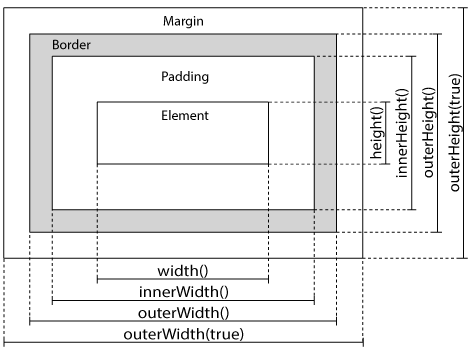
//作业：将封装的那些方法集合起来，装在一个tools.js文件中

**（二）盒子模型**

1、elementNode.getBoundingClientRect()：获取盒子的位置信息以及尺寸信息；

2、盒子更多尺寸及位置信息：

offsetWidth：本身宽度[103]+边框线[3+3]+左右内边距[5+5]=119；-outerWidth()  
offsetHeight：本身高度[104]+边框线[3+3]+上下内边距[4+4]=118；-outerHeight()  
offsetTop：相对有定位属性的父节点上偏移量=50；  
offsetLeft：相对有定位属性的父节点左偏移量=54；  
clientWidth：本身的宽度[103]+左右内边距[5+5]=96; -innerWidth()  
clientHeight：本身的高度[104]+上下内边距[4+4]=95； -innerHeight()  
clientTop：上边框线的宽度=3；----border-top  
clientLeft：左边框线的宽度=3；----border-left  
scrollWidth：盒子的实际宽度(包括不可见部分，不包括边线) =174； -innerWidth()  
scrollHeight：盒子的实际高度(包括不可见部分，不包括边线)=167； -innerHidth()  
scrollTop：滚动条向下滚动的距离=50；  
scrollLeft：滚动条向右滚动的距离=30；  
window**.**innerHeight：浏览器窗口可见区域高度=695；   
window**.**innerWidth：浏览器窗口可见区域宽度=622;



**（三）样式操作css：**

1、读写元素style属性：

dom.style.property //通过元素的style属性间接操作css样式

1) dom.style指向CSSStyleDeclaration

2) 可读写行间样式，没有兼容性问题，property碰到float这样的保留字属性，前面应加“css”（eg：float——>cssFloat）

3) 复合属性必需拆解，组合单词必须变成小驼峰式写法

4) 写入的值必须是字符串格式

如果改的属性比较多，推荐使用class进行样式修改；或通过改变引用的CSS文件改变样式。

2、查询计算样式-bom操作

1) window.getComputedStyle(ele,null);//获取当前元素展示的一切显示样式集合

a) 计算样式只读

b) 返回的计算样式的值都是绝对值，没有相对单位

//（em）1em=默认16px/最小12px，相对于font-size

c) IE8及IE8以下不兼容

d) 补充：第二个参数解决的是获取伪元素样式

eg：window.getComputedStyle (ele,"after");

2) ele.currentStyle

a) 计算样式，只读

b) 返回的计算样式的值不是经过转换的绝对值

c) IE独有的属性

//小练习：封装兼容性方法getStyle(elem,property)，解决IE兼容性问题

//小练习：利用上面封装的方法实现界面方块连续移动

**（四）表单操作：**

1、获取表单和表单元素

1) document.forms：返回页面中所有表单元素；

2) 表单和表单元素可以用标准的节点获取方式来获取；

//获取 id是myform表单下所有的复选按钮

document.querySelector('#myform input[type="checkbox"]');

//获取 id是myform表单下所有的name是newsid的复选按钮

document.querySelector('#myform input[type="checkbox"][name="newsid"]');

2、表单和表单类元素的属性和方法

1) 表单.submit()：表单提交；

2) 表单.reset()：表单重置；

3) 元素.type：可读可写，表单元素类型；

4) 元素.form：只读，包含该元素的form表单对象，不存在返回null；

5) 元素.name：可读可写，元素的名称；

6) 元素.value：可读可写，简单理解就是输入框的值；

3、元素的属性-开关按钮：

1) 单选按钮和复选框定义了checked属性，可读可写；

2) 选中时为true，你也可以把他设置为true表示选中；

//radio[1].checked=false/true

4、表单和元素的事件处理程序：

1) 表单.onsubmit：监听表单提交事件，表单提交到服务器前，先执行该事件；

// 该事件由 input\_submit/ button\_submit按钮触发,多用于数据提交前的验证

// 如果事件处理程序返回false，则表单提交取消，数据不会提交到服务器处理；

// 需要注意，表单.submit()方法不会触发onsubmit事件；

2) 表单.onreset：监听表单重置事件，处理过程和onsubmit大致相似；

// 表单.reset ()方法可以触发onreset事件，多用于获取用户确认confirm

5、select元素一些常用操作

1) select节点.options：返回其所有option选项，类数组的HTMLOptionsCollection；

// 可通过selectNode[index]访问option元素

2) select节点.selectedIndex：可读可写，被选中的选项索引，下边从0开始；

// 通过修改selectedIndex的值设置默认选中选项；

3) select节点.value：得到选中的option值；

4) option节点.text：得到选项的文本信息，也就是用户看到的选项文字；

5) option节点.value：得到的是选项的值，和SELECT选中的值有区别；

8)一样可以通过节点操作对select的option进行增删查改操作；

// 小练习：实现联动菜单，下拉框之间有关联的；

var menu = [ {

name:'四川',pid:20,

children:

[

{name:'成都',cid:20001},{name:'德阳',cid:20002},{name:'绵阳',cid:20003},

{name:'乐山',cid:20004},{name:'广元',cid:20005}

]

},

{

name:'河南',pid:10,

children:

[

{name:'郑州',cid:10001},{name:'开封',cid:10002},{name:'洛阳',cid:10003},

{name:'信阳',cid:10004},{name:'安阳',cid:10005}

]

}

];

**（五）获取选中的文字window.getSelection().toString()：**

是鼠标松开的时候，可以获取到选中的文字信息；

//小练习，用户选取界面的文字后，询问用户“要去百度搜索？”；提示：询问用户前判断用户选择的文本是否为空

**（六）让内容可编辑**

可编辑的元素内容，这是要做编辑器的节奏吗？

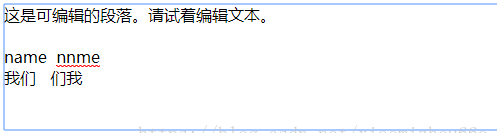
1. contenteditable：该属性使得元素具有可编辑功能，属性值设置为false显示关闭可编辑；

< div class="mydiv" contenteditable="true">

请编辑这里的内容

< /div>

1. spellcheck：该属性给元素开启拼写检查，属性值设置为false关闭拼写检查；



1. 浏览器定义了多项文本编辑命令（eg：copy/selectAll等），使用document. execCommand()来执行这些命令，但是兼容性堪忧；

API：<https://developer.mozilla.org/zh-CN/docs/Web/API/Document/execCommand>

注：界面元素的contenteditable属性值不能设置为true，否则影响copy命令。