DOM

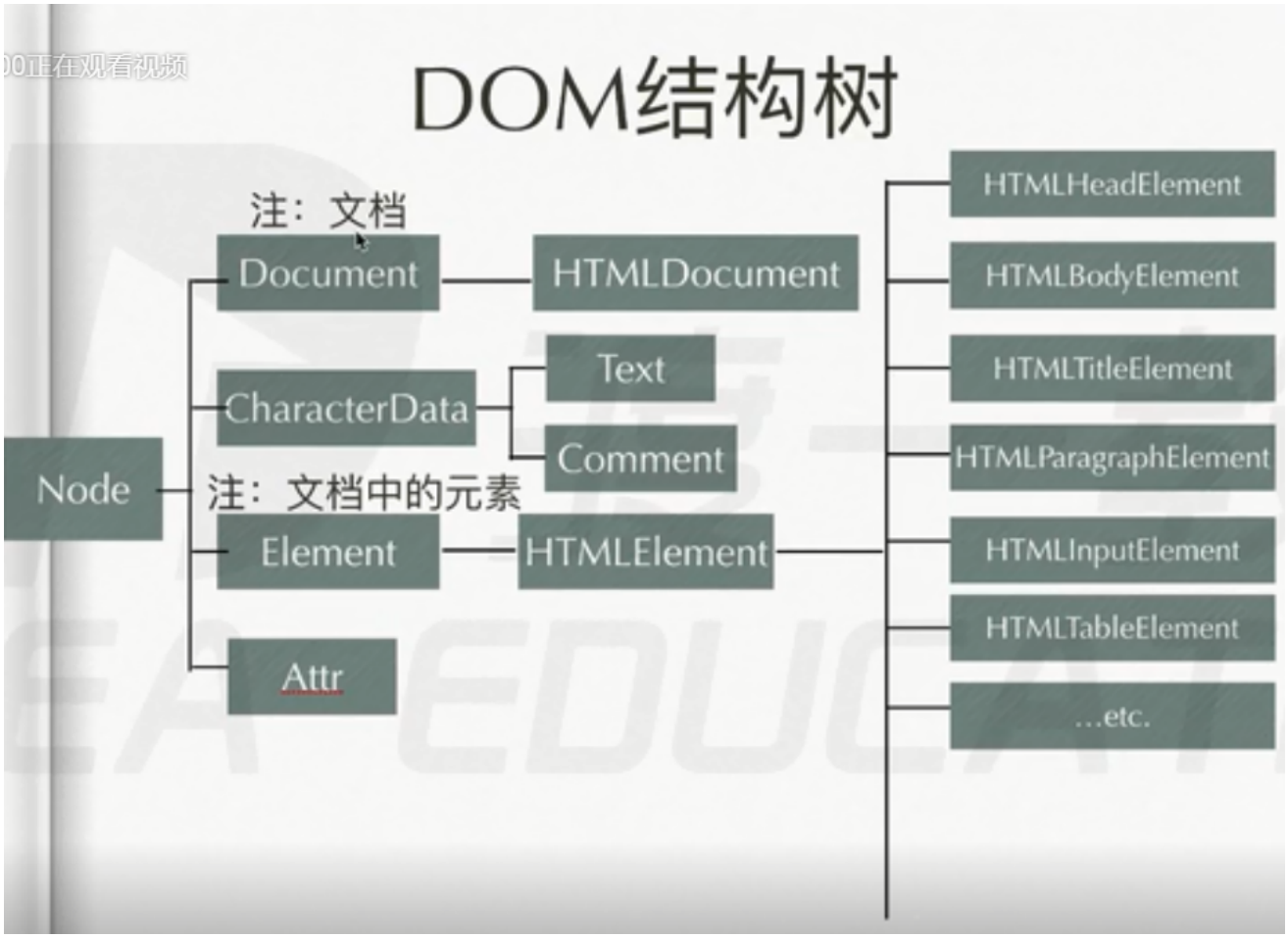
```
a) Document Object Model: 一套操作HTML以及XML方法的集合,定义了表示和修改文档所需的方法;
b) DOM对象即为宿主对象,由浏览器厂商定义,用来操作html和xml功能的一类对象的集合;
c) DOM是对HTML以及XML的标准编程接口;
--------------------------------
简单交互功能:
<div></div>
<script>
    var div = document.getElementsByTagName('div')[0]; //DOM 对象
    div.style.width = "100px";
    div.style.height = "100px";
    div.style.backgroundColor = "red";
    div.style.borderRadius = "50%";
    //定时器 点击实现效果变化
    var count = 1;
    div.onclick = function() {
        count ++;
        if(count % 2 == 0) {
            this.style.backgroundColor = "yellow";
        }else {
            this.style.backgroundColor = "red";
        }
    }
</script>
----------
选项卡功能:
.active{
    background-color: pink;
}
.content{
    display: none;
    width: 100px;
    height: 100px;
    border: 1px solid black;
}
<div class="wrapper">
    <button class="active">1</button>
    <button>2</button>
    <button>3</button>
    <div class="content" style="display: block">a</div>
    <div class="content">b</div>
    <div class="content">c</div>
</div>
<script>
    var btn = document.getElementsByTagName('button'),
        div = document.getElementsByClassName('content'),
        len = btn.length;
    for(var i = 0;i < len;i ++) { //遍历出每一个button
            (function(n) { //立即执行函数
```

```
                btn[n].onclick = function() { //给遍历出的button绑定点击事件
                for(var j = 0;j < btn.length;j ++) { //初始化
                    btn[j].className = "";
                    div[j].style.display = "none";
                }
                this.className = "active"; //当前button变效果
                div[n].style.display = "block"; //对应的改变遍历出的div
            }
        }(i))
    }
</script>
-----------
小方块运动:
var div = document.createElement('div');
document.body.appendChild(div);
div.style.width = "100px";
div.style.height = "100px";
div.style.backgroundColor = "red";
div.style.borderRadius = "50%";
div.style.position = "absolute";
div.style.top = "0";
div.style.left = "0";
var speed = 1;
var timer = setInterval(function() { //定时器
    speed += speed/50;
    //div.style.left是字符串形式,需要取整转换为number再进行操作
    div.style.top = parseInt(div.style.top) + speed + "px";
    div.style.left = parseInt(div.style.left) + speed + "px";
    if(parseInt(div.style.top) > 400 && parseInt(div.style.left) > 400) {
        clearInterval(timer);
    }
},30);
-------------------
键盘操纵小方块运动:
<script>
var div = document.createElement('div');
document.body.appendChild(div);
div.style.width = "100px";
div.style.height = "100px";
div.style.backgroundColor = "red";
div.style.position = "absolute";
div.style.top = "0";
div.style.left = "0";
document.onkeydown = function(e) {
    switch(e.which) {
        case 37: //左
            div.style.left = paeseInt(div.style.left) - 5 + "px";
            break;
        case 38: //下
            div.style.top = paeseInt(div.style.top) - 5 + "px";
            break;
        case 39: //右
            div.style.left = paeseInt(div.style.left) + 5 + "px";
```

```
            break;
        case 40: //上
            div.style.top = paeseInt(div.style.top) + 5 + "px";
            break;
    }
}
--------------------
```
不断滑动颜色会变浅:
```
*{
    margin: 0;
    padding: 0;
    list-style: none;
}
li{
    box-sizing: border-box;
    float: left;
    width: 10px;
    height: 10px;
    border: 1px solid black;
}
ul{
    width: 200px;
    height: 200px;
}
<body>
    <ul>
        <li img-data="0"></li> *400
    </ul>
<script>
    var ul = document.getElementsByTagName('ul')[0];
    ul.onmouseover = function(e) {
        var event = e || window.event;
        var target = event.target || event.srcElement; //兼容性处理
        target.style.backgroundColor = "rgb(0,255," + target.getAttribute('img-data')
+")";
        target.setAttribute('img-data', parseInt(target.getAttribute('img-data')) + 9);
    }
```

查看元素节点:
```
a. document.getElementsByTagName();
b. document.getElementsByClassName(); //IE8和IE8以下版本没有,可以多个class连着一起使用
c. document.getElementById(); //元素id在IE8以下的浏览器,不区分大小写
d. document 代表整个文档
e. document.getElementsByName(); //只有部分标签的name可生效 (表单,表单元素,img,iframe)
f. querySlectorAll(); //css选择器,镜态,不实时更新,类似照片,在IE7及以下版本中没有,H5新增
g. querySlector //css选择器,镜态,不实时更新,类似照片,在ie7和ie7以下的版本中没有,了解即可
--------------------------------------------------------------------------------
```
遍历节点树:
```
a. parentNode ---> 父节点,最顶端的parentNode为#document
b. childNodes ---> 子节点们,包含文本节点
c. firstChild ---> 第一个子节点
d. lastChild ---> 最后一个子节点
e. nextSibling ---> 后一个兄弟节点
```

```
f. previousSibling ---> 前一个兄弟节点
-----------------------------------------------------
<div>
    <strong></strong>
    <span></span>
</div>
var strong = document.getElementsBytagName('strong')[0];
strong.parentNode;
-----------------------------------------------------------
基于元素节点树的遍历:
a. parentElement ---> 返回当前元素父元素的节点(IE不兼容)
b. children ---> 返回当前元素的子节点
c. node.childElementCount === node.children.length 当前元素节点的子元素节点个数(IE不兼容)
d. firstElementChild ---> 返回第一个元素节点(IE不兼容)
e. lastElementChild ---> 返回最后一个元素节点(IE不兼容)
f. nextElementSibling / previousElementSibing ---> 返回后一个/前一个兄弟元素
-------------------------------------------------------------------------
节点类型: ---> 调用nodeType返回数值
a)元素节点    1
b)属性节点    2
c)文本节点    3
d)注释节点    8
e)document  9
f)DocumentFragment  11
-------------------------------
节点的四个属性:
a)nodeName
    元素的标签名,以大写形式表示,只读
b)nodeValue
    Test节点或Comment节点的文本内容,可读写
*c)nodeType
    该节点的类型,只读
d)attributes
    Element节点的属性结合
--------------------------
节点的一个方法:
Node.hasChildNodes();
```




```
document --原型--> HTMLDocument.prototype --原型--> Document.prototype
-----------------------------------------------------------------------
<div>
    <span></span>
</div>
var div = document.getElementsByTagName('div')[0];
var span = div.getElementsByTagName('span')[0];
-------------------------------------------------------------
document.documentElement --指代--> html
document.body --指代--> body
document.head --指代--> head
-------------------------------------------------------------------
1.getElementById方法定义在Document.prototype上,即Element节点上不能使用;
2.getElementsByName方法定义在HTMLDocument.prototype上,即非html中的document不能使用(xml
document,Element);
3.getElementsByTagName方法定义在Document.prototype和Element.prototype上;
4.HTMLDocument.prototype定义了一些常用的属性,body,head分别指代HTML文档中的<body><head>标签;
5.Document.prototype上定义了documentElement属性,指代文档的根元素,在HTML文档中,指代<html>元素;
```

```
小测试:
1.遍历元素节点树 (在原型链上编程);
-----------------------------------
2.封装函数,返回元素e的第n层祖先的元素节点;
    <div>
        <strong>
```

```
            <span>
                <i></i>
            </span>
        </strong>
    </div>
    function retParent(elem, n) {
        while(elem && n) { //容错处理,elem必须有意义,即elem为null时停止
            elem = elem/parentElement;
            n --;
        }
        return elem;
    }
    var i = document.getElementsByTagName('i')[0];
```

3.封装函数,返回元素e的第n个兄弟元素节点,n为正,返回后面的兄弟元素节点;n为负,返回前面的;n为0,返回自己;

```
    function retSibling(e, n) {
        while(e && n) {
            if(n > 0) {
                e = e.nextElementSibling; //ie9以下不能使用
                n --;
            }else {
                e = e.nextElementSibling;
                n ++;
            }
        }
        return e;
    }
```

4.封装myChildren功能,解决以前部分浏览器的兼容性问题;

```
    //找该元素的子元素节点
    //如何区分元素节点和非元素节点
    Element.prototype.myChildren = function() {
        var child = this.childNodes;
        var len = child.length;
        var arr = [];
        for(var i = 0;i < len; i ++) {
        if(child[i].nodeType == 1) {
            arr.push(child[i]);
            }
        }
        return arr;
    }
```

5.封装hasChildren()方法,不可用children属性;

```
    Element.prototype.myChildren = function() {
        var child = this.childNodes;
        var len = child.length;
        var arr = [];
        for(var i = 0;i < len; i ++) {
        if(child[i].nodeType == 1) {
            return true;
            }
        }
```

```
        return false;
    }
```

```
元素节点的增加 插入 删除 替换
------------------------
增加:
a)document.createElement(); 创建元素节点
    var div = document.createElement('div');
    document.body.appendChild(div);
b)document.createTextNode(); 创建文本节点
c)document.createComment(); 创建注释节点
----------------------------------------
插入:
appendChild();
insertBefore(a, b);
-----------------
<div></div>
var div = document.getElementsByTagName('div')[0];
var text = document.createTextNode('邓宝宝');
var span = document.createElement('span');
div.appendChild(text);
div.appendChild(span);
var text1 = document.createTextNode('demo');
span.appendChild(text1); //appendChild类似于剪切的功能
span.appendChild(text);
-------------------------
<div></div>
<span></span>
var div = document.getElementsByTagName('div')[0];
var span = document.getElementsByTagName('span')[0];
div.appendChild(span);
--------------------------------------------------------------------------------
ParentNode.insertBefore(a, b); //父级调用,往父级里面插元素,在最前面插入元素
eg: div.insertBefore(a, b); //div insert A before B; insert第一个before第二个
--------------------------------------------------------------------------------
删除:
parent.removeChild(); //实质上是剪切,并不是真的删除
child.remove(); //真的是删除
----------------------------
替换:
parent.replaceChild(new,origin); //拿新的元素替换老的元素,老的元素也是被剪切了
------------------------------------------------------------------------
Element节点的属性:
    innerHTML
    innerText(火狐不兼容)/textContent(老版本IE不好使) 不会管标签,把文本都取出来
------------------------------------------------------------------------
Element节点的方法:
    ele.setAttribute(); 设置行间属性
    ele.getAttribute(); 得到行间属性
---------------------------------
<div><div>
div.innerHTML = '123';
div.innerHTML += '456'; 后面追加数字
```

```
div.innerHTML = "<span style="color:#fff;font-size:20px">123</span>";
--------------------------------------------------------------------------
请编写一段js脚本生成下面这段DOM结构,要求使用标准的DOM方法或属性;
提示 dom.className可以读写class
    <div class="example">
        <p class="slogan">sxw</p>
    </div>
    <script>
        var div = document.createElement('div');
        var p = document.createElement('p');
        div.setAttribute('class','example');
        p.setAttribute('class','slogan');
        var text = document.createTextNode('sxw');
        p.appendChild(text);
        div.appendChild(p);
        document.body.appendChild(div);
```

不管她富甲一方还是一无所有，我都可以张开双手坦然拥抱她

```
练习:
1.封装函数insertAfter();功能类似insertBefore();
提示:可忽略老版本浏览器,直接在Element.prototype上编程;
    Element.prototype.insertAfter = function(targetNode, afterNode) {
        var beforeNode = afterNode.nextElementSibling;
        if(beforeNode == null) {
            this.appendChild(targetNode);
        }else {
            this.insertBefore(targetNode.beforeNode);
        }
    }
--------------------------------------------------------------------------------
2.将目标节点内部的节点顺序逆序; 用appendChild,思路:先把倒数第二个放到最后,..逐渐往下移
eg: <div><a></a><em></em></div>
----> <div><em></em><a></a></div>
```

日期对象 Date

```
var date = new Date;
Date(); //返回当前的日期和时间
date.getDate(); //一个月里的第几天
date.getDay(); //一周里的第几天,星期天是第一天
date.getMonth(); //返回月份(0-11),我们使用的时候要 + 1;
date.getFullYear(); //返回年份;
getHours();
getMinutes();
getSeconds(); //这里记录的都是new出来的对象那一刻的时间,不会变化;
getMilliseconds(); //返回毫秒
getTime(); //返回1970年1月1日至今的毫秒数;
```

```
var firstTime = new Date().getTime();
for(var i = 0;i < 1000000; i ++) {
```

```
    //运行了多久啊
}
var lastTime = new Date().getTime();
console.log(lastTime - firstTime);
--------------------------------------
类似闹钟:
var date = new Date(); //获取当前时间
setInterval(function() {
    if(new Date().getTime() - date.getTime() > 3000) { //过了3秒的话
        console.log('时间到了'); //打印一下
    }
},1000);
--------------------------------------
封装函数,打印当前是何年何月何日何时几分几秒？
```

```
js定时器:
setInterval(); 非常不准,循环执行
setTimeout(); 延迟执行,执行一次
clearInterval();
clearTimeout();
----------------------------------------
全局对象window上的方法,内部函数this指向window
注意: setInterval("func()",1000);
--------------------------------
var i = 0;
setInterval(function() {
    i++;
    console.log(i);
},1000);
-------------------------------------
var i = 0;
var timer = setInterval(function() {
    console.log(i ++);
    if(i > 10) {
        clearInterval(timer);
    }
}, 1000);
-------------------------------------
var timer = setTimeout(function(){
    console.log('a');
}, 1000);
clearTimeout(timer);
-----------------------------------
写一个定时器:
input{
    border: 1px solid rgba(0,0,0,0.1);
    text-align: right;
    font-size: 20px;
    font-weight: bold;
}
minutes:<input type="text" value="0">
seconds:<input type="text" value="0">
var minutesNode = document.getElementsByTagName('input')[0],
```

```
    secondsNode = document.getElementsByTagName('input')[1],
    minutes = 0,
    seconds = 0;
var timer = setInterval(function() {
    seconds ++;
    if(seconds == 60) {
        seconds = 0;
        minutes ++;
    }
    secondsNode.value = seconds;
    minutesNode.value = minutes;
    if(minutes == 3) {
        clearInterval(timer);
    }
}, 1000);
```

查看滚动条滚动距离:
window.pageXOffset/pageYOffset ie8及以下不能使用
document.body.scrollLeft === documentElement.scrollLeft/scrollTop ie8以下这两个有效

封装兼容性方法,求滚动轮滚动距离 getScorollOffset();

```
function getScrolloffset() {
    if(window.pageXoffset) {
        return {
            x : window.pageXoffset,
            y : window.pageYoffset,
        }
    }else {
        return {
            x : document.body.scrollLeft + document.documentElement.scrollLeft,
            y : document.body.scrollTop + document.documentElement.scrollTop
        }
    }
}
```

查看视口的尺寸:
window.innerWidth/innerHeight (IE8及IE8以下不兼容)
document.documentElement.clientWidth/clientHeight //标准模式下,任意浏览器都兼容
document.body.clientWidth/clientHeight //适用于怪异模式下的浏览器

封装兼容性方法,返回浏览器视口尺寸getViewportOffset();

```
function getViewportOffset() {
    if(window.innerWidth) {
        return {
            w : window.innerWidth,
            h : window.innerHeight
        }
    }else {
        if(document.compatMode === "BackCompat") {
            return {
                w : document.body.clientWidth,
                h : document.body.clientHeight
```

```
            }
        }else {
            return {
                w : document.documentElement.clientWidth,
                h : document.documentElement.clientHeight
            }
        }
    }
}
```

查看元素的几何尺寸（了解即可）
domEle.getBoundingClientRect();
兼容性很好,该方法返回一个对象,对象里面有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,height和width属性老版本IE并未实现,返回的结果并不是实时的;

查看元素的尺寸（视觉上的尺寸）
dom.offsetWidth/dom.offsetHeight
查看元素的位置（当前元素距离自己有定位的父级的距离）
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;
dom.offsetParent 返回最近有定位的父级,如无则返回body,body.offsetParent返回null
eg: 求元素相对于文档的坐标 getElementPosition

让滚动条滚动:
    window上有三个方法
    scrollTo()效果不累加, scrollBy()效果累加;
    三个方法功能类似,用法是将x,y坐标传入。即实现让滚动轮滚动到当前位置。
    区别:scrollBy()会在之前的数据基础之上做累加;
    eg:利用scrollBy()快速阅读的功能;

自动阅读功能:

```
<div style="width:100px;height:100px;background-color:orange;color:#fff;font-size:
40px;font-weight:bold;text-align:center;line-
height:100px;position:fixed;bottom:200px;right:50px;border-
radius:50%;opacity:0.5">start</div>
<div style="width:100px;height:100px;background-color:orange;color:#0f0;font-size:
40px;font-weight:bold;text-align:center;line-
height:100px;position:fixed;bottom:50px;right:50px;border-
radius:50%;opacity:0.5">stop</div>
var start = document.getElementsByTagName('div')[0];
var stop = document.getElmentsByTagName('div')[1];
var timer = 0;
var key = true; //加锁性思维,解决产生多个定时器清除不了的问题
start.onclick = function() {
    if(key) {
            timer = setInterval(function() {
                window.scrollBy(0, 10);
            },100);
            key = false;
        }
}
```

```
stop.onclick = function() {
    clearInterval(timer);
    key = true;
}
```

脚本化CSS

```
读写元素css属性
div.style.backgroundColor = "green";
只能在行间样式上进行添加的才能识别,没有兼容性问题,碰到float这样的保留字属性,前面应加css
eg:div.style.cssFloat
复合属性必须拆解,组合单词变成小驼峰式写法
写入的值必须是字符串格式
-----------------------------------
查询计算样式
window.getComputedStyle(ele,null); //这里第二个参数是填写伪类选择器时候使用的
window.getComputedStyle(div,null).width
计算样式只读
返回的计算样式的值都是绝对值,没有相对单位
IE8及IE8以下不兼容
-----------------------------------
查询样式
ele.currentStyle
计算样式只读
返回的计算样式的值不是经过转换的绝对值
IE独有的属性
封装兼容性方法getStyle(elem,prop);
function getStyle(elem, prop) {
    if(window.getComputedStyle) {
        return window.getComputedStyle(elem, null)[prop];
    }else {
        return elem.currentStyle[prop];
    }
}
-----------
方块运动:
<div style="width: 100px;height: 100px;background-
color:red;position:absolute;left:0;top:0;"></div>
function getStyle(elem, prop) {
    if(window.getComputedStyle) {
        return window.getComputedStyle(elem, null)[prop];
    }else {
        return elem.currentStyle[prop];
    }
}
var div = document.getElementsByTagName('div')[0];
var speed = 3;
var timer = setInterval(function() {
    speed += speed/7;
    div.style.left = parseInt(getStyle(div, 'left')) + speed + 'px';
    if(parseInt(div.style.left) > 500) {
        clearInterval(timer);
    }
```

```
}, 10);
```