1. 元素偏移量offset系列

1.1. offset概述

offset翻译过来就是偏移量，我们使用offset系列相关属性可以动态的得到该属性的位置（偏移）大小等

获得元素距离带有定位父元素的位置
获得元素自身的大小（宽度高度）

注意：返回的数值都不带单位

offset系列常用属性
（图：offset系列常用属性）

offset系列属性	作用
element.offsetParent	返回作为该元素带有定位的父级元素 如果父级都没有定位则返回body
element.offsetTop	返回元素相对带有定位父元素上方的偏移
element.offsetLeft	返回元素相对带有定位父元素左边框的偏移
element.offsetWidth	返回自身包括padding 、 边框、内容区的宽度，返回数值不带单位
element.offsetHeight	返回自身包括padding、边框、内容区的高度，返回数值不带单位

```
<div class="father">
    <div class="son"></div>
</div>
<div class="w"></div>

<script>
    var father = document.querySelector('.father');
    var son = document.querySelector('.son');
    var w = document.querySelector('.w');
    // 可以得到元素的偏移 位置 返回不带单位的数值
    console.log(father.offsetTop);
    console.log(father.offsetLeft);
    // 它以带有定位的父亲为准 如果没有父亲或者父亲没有定位，则以body为准
    console.log(son.offsetLeft);

    // 可以得到元素的大小 宽度和高度 是包含padding+border+width
    console.log(w.offsetWidth);
    console.log(w.offsetHeight);

    // 返回带有定位的父亲，否则返回的是body
    console.log(son.offsetParent);     //返回的是带有定位的父亲，否则返回的是body
    console.log(son.parentNode);      //返回的是最近一级的父亲 亲爸爸 不管父亲有没有定位
</script>
```

1.2. offset与style的区别

offset：

- offset可以得到任意样式表中的样式值
- offset系列获得的数值是没有单位的
- offseWidth包含padding+border+width
- offsetWidth等属性是只读属性，只能获取不能赋值

所以，我们想要获取元素大小位置，用offset更合适

style：

- style只能得到行内样式表中的样式值
- style.width获得的是带有单位的字符串
- style.width获得不包含padding和border的值
- style.width是可读写属性，可以获取也可以赋值

所以，我们想要给元素更改值，则需要用style改变

2. 元素可视区client系列

client翻译过来就是客户端，我们使用client系列的相关属性来获取元素可视区的相关信息。通过client系列的相关属性可以动态的得到该元素的边框大小、元素大小等

（图：client系列属性）

client系列属性	作用
element.clientTop	返回元素上边框的大小
element.clientLeft	返回元素左边框的大小
element.clientWidth	返回自身包括padding 、 内容区的宽度，不含边框，返回数值不带单位
element.clientHeight	返回自身包括padding 、 内容区的高度，不含边框，返回数值不带单位

```
<div></div>
<script>
    // client 宽度 和offsetWidth 最大的区别就是不包含边框
    var div=document.querySelector('div');
    console.log(div.clientWidth);

</script>
```

立即执行函数

```
(function(){})()或者(function(){}())
```

主要作用：创建一个独立的作用域，避免了命名冲突的问题

```
<script>
    // 立即执行函数  不需要调用，能够立马执行的函数
    function fn() {
        console.log(1);
    }
    fn();

    // 写法：
    // (function(){})()
    // 或者：
    // (function(){}());

    (function (a) {
        console.log(a);
    })(100); //第二个小括号可以看做是调用函数    也可以传递参数进来

    (function sum(a, b) {
        console.log(a + b);
    }(1, 2))

    // 立即执行函数最大的作用就是独立创建了一个作用域 里面所有的变量都是局部变量，不会有命名冲突
</script>
```

以下三种情况都会刷新页面都会触发load事件

- a标签的超链接
- F5或者刷新按钮（强制刷新）
- 前进后退按钮

但是火狐中，有个特点，有个“往返缓存”，这个缓存中不仅保留着页面数据，还保存了DOM和JavaScript的状态，实际上是将整个页面都保存在了内存里

所以此时后退按钮不能刷新页面

此时可以使用pageshow事件来触发。这个事件在页面显示时触发，无论页面是否来自缓存。在加载页面中，pageshow会在load事件触发后触发。根据事件对象中的persisted来判断是否是缓存中的页面触发的pageshow事件 ， 注意这个事件给window添加

```
<script>
    window.addEventListener('pageshow',function(){
        alert(11);
    })
</script>
<a href="http://www.baidu.com/">链接</a>
```

3. 元素滚动scroll系列

3.1. 元素scroll系列属性

scroll翻译过来就是滚动的意思，我们使用scroll系列的相关属性可以动态的得到该元素的大小、滚动距等

（图：scroll系列属性）

scroll系列属性	作用
element.scrollTop	返回被卷去的上侧距离，返回数值不带单位
element.scrollLeft	返回被卷去的左侧距离，返回数值不带单位
element.scrollWidth	返回自身实际的宽度，不含边框，返回数值不带单位
element.scrollHeight	返回自身实际的高度，不含边框，返回数值不带单位

```html
<div>
    我是内容
    我是内容
    我是内容
    我是内容
    我是内容
    我是内容
    我是内容
    我是内容
    我是内容
    我是内容
    我是内容
    我是内容
    我是内容
    我是内容
    我是内容
    我是内容
    我是内容
    我是内容
    我是内容
    我是内容
    我是内容
    我是内容
    我是内容
    我是内容
    我是内容
    我是内容
    我是内容
    我是内容
    我是内容
    我是内容
    我是内容
    我是内容
    我是内容
    我是内容
    我是内容
    我是内容
    我是内容
</div>
<script>
    // scroll系列
    var div = document.querySelector('div');
    console.log(div.scrollHeight);
</script>
```

3.2. 页面被卷去的头部

如果浏览器的高（或宽）度不足以显示整个页面时，会自动出现滚动条，当滚动条向下滚动时，页面上面被隐藏掉的高度，我们就称之为页面被卷去的高度，滚动条在滚动时会触发onscroll事件

```
// scroll滚动事件
        div.addEventListener('scroll',function(){
            console.log(div.scrollTop);
        })
```

3.3. 页面被卷去的头部兼容性解决方案

需要注意的是，页面被卷去的头部，有兼容性问题，因此被卷去的头部通常有以下几种写法：

- 声明了DTD，使用document.documetnElement.scrollTop
- 未声明DTD，使用document.body.scrollTop
- 新方法window.pageOffset和window.pageXOffset，IE9开始支持

三大系列总结

(图：三大系列总结)

三大系列大小对比	作用
element.offsetWidth	返回自身包括padding 、 边框、内容区的宽度，返回数值不带单位
element.clientWidth	返回自身包括padding 、 内容区的宽度，不含边框，返回数值不带单位
element.scrollWidth	返回自身实际的宽度，不含边框，返回数值不带单位

它们主要用法：

- offset系列经常用于获得元素位置 offsetLeft offsetTop
- client系列经常用于获取元素大小 clientWidth clientHeight
- scroll经常用于获取滚动距离 scrollTop scrollLeft
- 注意页面的滚动距离通过window.pageXOffset获得

mouseenter和mouseover的区别

mouseenter鼠标事件

当鼠标移动到元素上就会触发mouseenter事件
类似于mouseover，他们两者的区别是：
mouseover鼠标经过自身盒子会触发，经过子盒子还会触发。mouseenter只会经过自身盒子触发

之所以是这样，是因为**mouseenter不会冒泡**

跟mouseenter搭配使用的鼠标离开mouseleave同样不会冒泡

4. 动画函数封装

4.1. 动画实现原理

核心原理：通过定时器setInterval()不断移动盒子位置

实现步骤：

- 获得当前盒子位置
- 让盒子在当前位置加1和移动距离
- 利用定时器不断重复这个操作
- 加一个结束定时器的条件

注意此元素需要添加定位，才能使用element.style.left

```
<div></div>

<script>
    var div = document.querySelector('div');
    var timer = setInterval(function () {
        if (div.offsetLeft >= 400) {
            // 停止动画 本质是停止定时器
            clearInterval(timer);
        }
        div.style.left = div.offsetLeft + 1 + 'px';
    }, 30);
</script>
```

4.2. 动画函数简单封装

注意函数需要传递2个参数，动画对象和移动到的距离

```
<script>
    // 简单动画函数封装
    // obj 目标对象  target目标位置
    function animate(obj, target) {
        var timer = setInterval(function () {
            if (obj.offsetLeft >= target) {
                // 停止动画 本质是停止定时器
                clearInterval(timer);
            }
            obj.style.left = obj.offsetLeft + 1 + 'px';
        }, 30);
    }

    var div = document.querySelector('div');

    //调用函数
    animate(div, 300);
</script>
```

4.3. 动画函数给不同的元素记录不同的定时器

如果多个元素都使用这个动画函数，每次都要var声明定时器。我们可以给不同的元素使用不同的定时器（自己专门用自己的定时器）

核心原理：利用js是一门动态语言，可以很方便的给当前对象添加属性