



黑马程序员™  
www.itheima.com

传智播客旗下  
高端IT教育品牌

# PC 端网页特效

# 目录 Contents

- ◆ 元素偏移量 offset 系列
- ◆ 元素可视区 client 系列
- ◆ 元素滚动 scroll 系列
- ◆ 动画函数封装
- ◆ 常见网页特效案例

# 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元素的位置（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：

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

# 1. 元素偏移量 offset 系列

## 1.2 offset 与 style 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
- offsetWidth 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

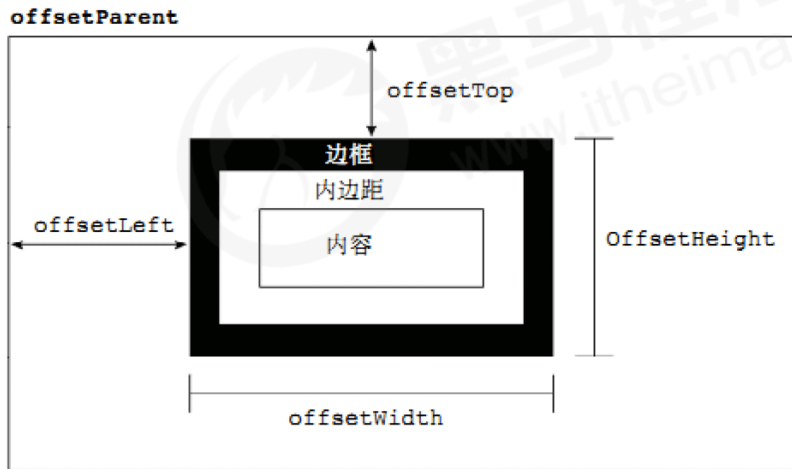
- style 只能得到行内样式表中的样式值
- style.width 获得的是带有单位的字符串
- style.width 获得不包含padding和border 的值
- style.width 是可读写属性，可以获取也可以赋值
- 所以，我们想要给元素更改值，则需要用style改变

# 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元素的位置（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）



# 1. 元素偏移量 offset 系列



案例：获取鼠标在盒子内的坐标



# 1. 元素偏移量 offset 系列



## 案例分析

- ① 我们在盒子内点击，想要得到鼠标距离盒子左右的距离。
- ② 首先得到鼠标在页面中的坐标 ( e.pageX, e.pageY )
- ③ 其次得到盒子在页面中的距离 ( box.offsetLeft, box.offsetTop)
- ④ 用鼠标距离页面的坐标减去盒子在页面中的距离，得到 鼠标在盒子内的坐标
- ⑤ 如果想要移动一下鼠标，就要获取最新的坐标，使用鼠标移动事件 mousemove

# 1. 元素偏移量 offset 系列



## 实现代码

```
var box = document.querySelector('.box');
box.addEventListener('mousemove', function(e) {
  var x = e.pageX - this.offsetLeft;
  var y = e.pageY - this.offsetTop;
  this.innerHTML = 'x坐标是' + x + ' y坐标是' + y;
})
```



# 1. 元素偏移量 offset 系列



## 案例：模态框拖拽

弹出框，我们也称为模态框。

1. 点击弹出层，会弹出模态框，并且显示灰色半透明的遮挡层。
2. 点击关闭按钮，可以关闭模态框，并且同时关闭灰色半透明遮挡层。
3. 鼠标放到模态框最上面一行，可以按住鼠标拖拽模态框在页面中移动。
4. 鼠标松开，可以停止拖动模态框移动。

# 1. 元素偏移量 offset 系列



## 案例分析

- ① 点击弹出层，模态框和遮挡层就会显示出来 `display:block;`
- ② 点击关闭按钮，模态框和遮挡层就会隐藏起来 `display:none;`
- ③ 在页面中拖拽的原理：鼠标按下并且移动，之后松开鼠标
- ④ 触发事件是鼠标按下 `mousedown`，鼠标移动 `mousemove` 鼠标松开 `mouseup`
- ⑤ 拖拽过程：鼠标移动过程中，获得最新的值赋值给模态框的 `left` 和 `top` 值，这样模态框可以跟着鼠标走了
- ⑥ 鼠标按下触发的事件源是最上面一行，就是 `id` 为 `title`
- ⑦ 鼠标的坐标 减去 鼠标在盒子内的坐标，才是模态框真正的位置。
- ⑧ 鼠标按下，我们要得到鼠标在盒子的坐标。
- ⑨ 鼠标移动，就让模态框的坐标 设置为：鼠标坐标 减去 盒子坐标即可，注意移动事件写到按下事件里面。
- ⑩ 鼠标松开，就停止拖拽，就是可以让鼠标移动事件解除

# 1. 元素偏移量 offset 系列



案例：仿京东放大镜



# 1. 元素偏移量 offset 系列



## 案例分析

- ① 整个案例可以分为三个功能模块
- ② 鼠标经过小图片盒子，黄色的遮挡层 和 大图片盒子显示，离开隐藏2个盒子功能
- ③ 黄色的遮挡层跟随鼠标功能。
- ④ 移动黄色遮挡层，大图片跟随移动功能。

# 1. 元素偏移量 offset 系列



## 案例分析

- ① 鼠标经过小图片盒子，黄色的遮挡层 和 大图片盒子显示，离开隐藏2个盒子功能
- ② 就是显示与隐藏



# 1. 元素偏移量 offset 系列



## 案例分析

- ① 黄色的遮挡层跟随鼠标功能。
- ② 把鼠标坐标给遮挡层不合适。因为遮挡层坐标以父盒子为准。
- ③ 首先是获得鼠标在盒子的坐标。
- ④ 之后把数值给遮挡层做为left 和top值。
- ⑤ 此时用到鼠标移动事件，但是还是在小图片盒子内移动。
- ⑥ 发现，遮挡层位置不对，需要再减去盒子自身高度和宽度的一半。
- ⑦ 遮挡层不能超出小图片盒子范围。
- ⑧ 如果小于零，就把坐标设置为0
- ⑨ 如果大于遮挡层最大的移动距离，就把坐标设置为最大的移动距离
- ⑩ 遮挡层的最大移动距离：小图片盒子宽度 减去 遮挡层盒子宽度

# 1. 元素偏移量 offset 系列



## 案例分析

① 移动黄色遮挡层，大图片跟随移动功能。

$$\frac{1}{2} = \frac{x}{4} \quad \text{求 } x ? \quad x = \frac{1 \times 4}{2}$$

② 求大图片的移动距离公式

$$\frac{\text{遮挡层移动距离}}{\text{遮挡层最大移动距离}} = \frac{\text{大图片移动距离}}{\text{大图片最大移动距离}}$$

求大图片移动距离？

# 1. 元素偏移量 offset 系列



## 案例分析

- ① 移动黄色遮挡层，大图片跟随移动功能。
- ② 求大图片的移动距离公式

$$\text{大图片移动距离} = \frac{\text{遮挡层移动距离} * \text{大图片最大移动距离}}{\text{遮挡层最大移动距离}}$$



# 目录 Contents

- ◆ 元素偏移量 offset 系列
- ◆ 元素可视区 client 系列
- ◆ 元素滚动 scroll 系列
- ◆ 动画函数封装
- ◆ 常见网页特效案例

## 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。

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

## 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。



## 2. 元素可视区client系列



### 案例：淘宝 flexible.js 源码分析

立即执行函数 (function() {}()) 或者 (function(){})()

主要作用：创建一个独立的作用域。避免了命名冲突问题

## 2. 元素可视区client系列



### 案例：淘宝 flexible.js 源码分析

下面三种情况都会刷新页面都会触发 load 事件。

1. a标签的超链接
2. F5或者刷新按钮（强制刷新）
3. 前进后退按钮

但是火狐中，有个特点，有个“往返缓存”，这个缓存中不仅保存着页面数据，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。在重新加载页面中，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。**

# 目录 Contents

- ◆ 元素偏移量 offset 系列
- ◆ 元素可视区 client 系列
- ◆ 元素滚动 scroll 系列
- ◆ 动画函数封装
- ◆ 常见网页特效案例

## 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，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、滚动距离等。

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

## 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，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、滚动距离等。

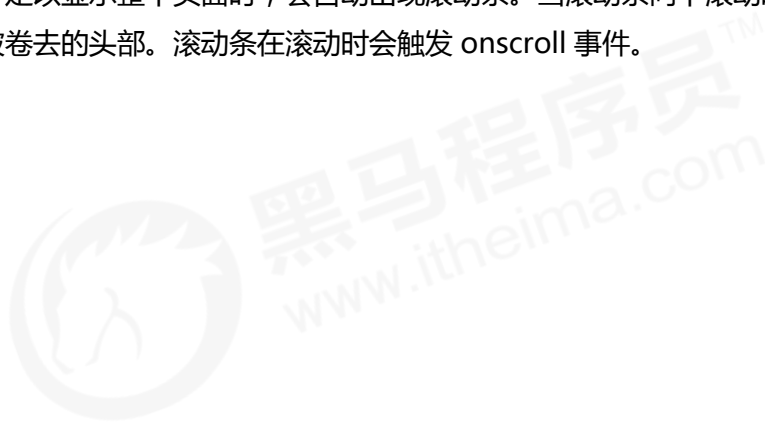




## ■ 3. 元素滚动 scroll 系列

### 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，我们就称为页面被卷去的头部。滚动条在滚动时会触发 onscroll 事件。



## 3. 元素滚动 scroll 系列



### 案例：仿淘宝固定右侧侧边栏

1. 原先侧边栏是绝对定位
2. 当页面滚动到一定位置，侧边栏改为固定定位
3. 页面继续滚动，会让 返回顶部显示出来

## 3. 元素滚动 scroll 系列



### 案例分析

- ① 需要用到页面滚动事件 scroll 因为是页面滚动，所以事件源是 document
- ② 滚动到某个位置，就是判断页面被卷去的上部值。
- ③ 页面被卷去的头部：可以通过 `window.pageYOffset` 获得 如果是被卷去的左侧 `window.pageXOffset`
- ④ 注意，元素被卷去的头部是 `element.scrollTop`，如果是页面被卷去的头部则是 `window.pageYOffset`
- ⑤ 其实这个值 可以通过盒子的 `offsetTop` 可以得到，如果大于等于这个值，就可以让盒子固定定位了

## 3. 元素滚动 scroll 系列

### 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如下几种写法：

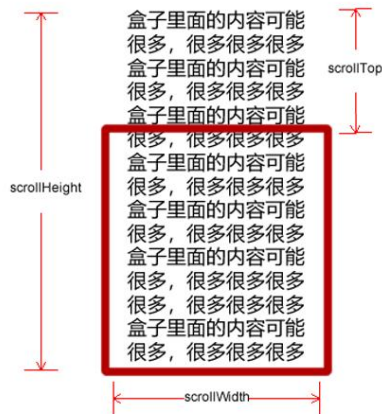
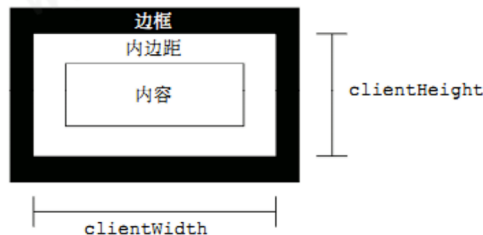
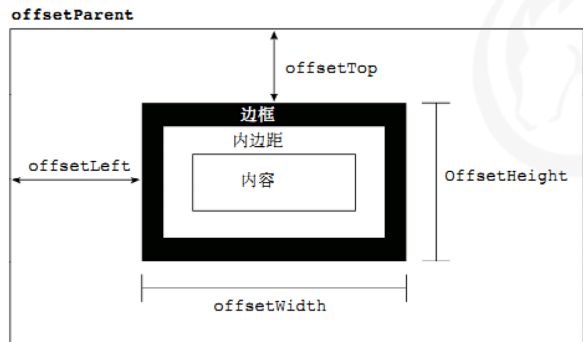
1. 声明了 DTD，使用 `document.documentElement.scrollTop`
2. 未声明 DTD，使用 `document.body.scrollTop`
3. 新方法 `window.pageYOffset` 和 `window.pageXOffset`，IE9 开始支持

```
function getScroll() {  
    return {  
        left: window.pageXOffset || document.documentElement.scrollLeft || document.body.scrollLeft || 0,  
        top: window.pageYOffset || document.documentElement.scrollTop || document.body.scrollTop || 0  
    };  
}
```

使用的时候 `getScroll().left`

# ■ 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



# ■ 三大系列总结

他们主要用法：

1. offset系列 经常用于获得元素位置 `offsetLeft` `offsetTop`
2. client 经常用于获取元素大小 `clientWidth` `clientHeight`
3. scroll 经常用于获取滚动距离 `scrollTop` `scrollLeft`
4. 注意页面滚动的距离通过 `window.pageXOffset` 获得

# ■ mouseenter 和mouseover的区别

## mouseenter 鼠标事件

- 当鼠标移动到元素上时就会触发 mouseenter 事件
- 类似 mouseover，它们两者之间的差别是
- 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
- 之所以这样，就是因为mouseenter不会冒泡
- 跟mouseenter搭配 鼠标离开 mouseleave 同样不会冒泡

# 目录 Contents

- ◆ 元素偏移量 offset 系列
- ◆ 元素可视区 client 系列
- ◆ 元素滚动 scroll 系列
- ◆ 动画函数封装
- ◆ 常见网页特效案例





## 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：

1. 获得盒子当前位置
2. 让盒子在当前位置加上1个移动距离
3. 利用定时器不断重复这个操作
4. 加一个结束定时器的条件
5. 注意此元素需要添加定位，才能使用`element.style.left`

## 4. 动画函数封装

### 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。



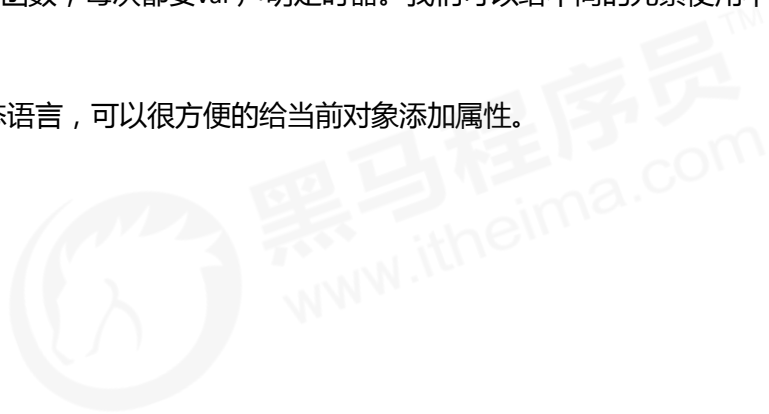


## 4. 动画函数封装

### 4.3 动画函数给不同元素记录不同定时器

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。





## 4. 动画函数封装

### 4.4 缓动效果原理

缓动动画就是让元素运动速度有所变化，最常见的是让速度慢慢停下来

思路：

1. 让盒子每次移动的距离慢慢变小，速度就会慢慢落下来。
2. 核心算法： $(\text{目标值} - \text{现在的位置}) / 10$  做为每次移动的距离 步长
3. 停止的条件是：让当前盒子位置等于目标位置就停止定时器
4. 注意步长值需要取整



## 4. 动画函数封装

### 4.5 动画函数多个目标值之间移动

可以让动画函数从 800 移动到 500。

当我们点击按钮时候，判断步长是正值还是负值

1. 如果是正值，则步长 往大了取整
2. 如果是负值，则步长 向小了取整



## 4. 动画函数封装

### 4.6 动画函数添加回调函数

**回调函数原理**：函数可以作为一个参数。将这个函数作为参数传到另一个函数里面，当那个函数执行完之后，再执行传进去的这个函数，这个过程就叫做**回调**。

回调函数写的位置：定时器结束的位置。





## 4. 动画函数封装

### 4.7 动画函数封装到单独JS文件里面

因为以后经常使用这个动画函数，可以单独封装到一个JS文件里面，使用的时候引用这个JS文件即可。

1. 单独新建一个JS文件。
2. HTML文件引入JS文件。



# 目录 Contents

- ◆ 元素偏移量 offset 系列
- ◆ 元素可视区 client 系列
- ◆ 元素滚动 scroll 系列
- ◆ 动画函数封装
- ◆ 常见网页特效案例



## 5. 常见网页特效案例



### 案例：网页轮播图

轮播图也称为焦点图，是网页中比较常见的网页特效。

功能需求：

1. 鼠标经过轮播图模块，左右按钮显示，离开隐藏左右按钮。
2. 点击右侧按钮一次，图片往左播放一张，以此类推，左侧按钮同理。
3. 图片播放的同时，下面小圆圈模块跟随一起变化。
4. 点击小圆圈，可以播放相应图片。
5. 鼠标不经过轮播图，轮播图也会自动播放图片。
6. 鼠标经过，轮播图模块，自动播放停止。

## 5. 常见网页特效案例



### 案例分析

- ① 因为js较多，我们单独新建js文件夹，再新建js文件，引入页面中。
- ② 此时需要添加 load 事件。
- ③ 鼠标经过轮播图模块，左右按钮显示，离开隐藏左右按钮。
- ④ 显示隐藏 display 按钮。

## 5. 常见网页特效案例



### 案例分析

- ① 动态生成小圆圈
- ② 核心思路：小圆圈的个数要跟图片张数一致
- ③ 所以首先先得到ul里面图片的张数（图片放入li里面，所以就是li的个数）
- ④ 利用循环动态生成小圆圈（这个小圆圈要放入ol里面）
- ⑤ 创建节点 `createElement('li')`
- ⑥ 插入节点 `ol.appendChild(li)`
- ⑦ 第一个小圆圈需要添加 `current` 类

## 5. 常见网页特效案例



### 案例分析

- ① 小圆圈的排他思想
- ② 点击当前小圆圈，就添加current类
- ③ 其余的小圆圈就移除这个current类
- ④ 注意：我们在刚才生成小圆圈的同时，就可以直接绑定这个点击事件了。

## 5. 常见网页特效案例



### 案例分析

- ① 点击小圆圈滚动图片
- ② 此时用到animate动画函数，将js文件引入（注意，因为index.js 依赖 animate.js 所以，animate.js 要写到 index.js 上面）
- ③ 使用动画函数的前提，该元素必须有定位
- ④ 注意是ul 移动 而不是小li
- ⑤ 滚动图片的核心算法：点击某个小圆圈，就让图片滚动 小圆圈的索引号乘以图片的宽度做为ul移动距离
- ⑥ 此时需要知道小圆圈的索引号，我们可以在生成小圆圈的时候，给它设置一个自定义属性，点击的时候获取这个自定义属性即可。



## 5. 常见网页特效案例



### 案例分析

- ① 点击右侧按钮一次，就让图片滚动一张。
- ② 声明一个变量num，点击一次，自增1，让这个变量乘以图片宽度，就是ul的滚动距离。
- ③ 图片无缝滚动原理
- ④ 把ul第一个li复制一份，放到ul的最后面
- ⑤ 当图片滚动到克隆的最后一张图片时，让ul快速的、不做动画的跳到最左侧：left为0
- ⑥ 同时num赋值为0，可以从新开始滚动图片了

## 5. 常见网页特效案例



### 案例分析

- ① 克隆第一张图片
- ② 克隆ul 第一个li cloneNode() 加true 深克隆 复制里面的子节点 false 浅克隆
- ③ 添加到 ul 最后面 appendChild

## 5. 常见网页特效案例



### 案例分析

- ① 点击右侧按钮，小圆圈跟随变化
- ② 最简单的做法是再声明一个变量`circle`，每次点击自增1，注意，左侧按钮也需要这个变量，因此要声明全局变量。
- ③ 但是图片有5张，我们小圆圈只有4个少一个，必须加一个判断条件
- ④ 如果`circle == 4`就 从新复原为 0



## 5. 常见网页特效案例



### 案例分析

- ① 自动播放功能
- ② 添加一个定时器
- ③ 自动播放轮播图，实际就类似于点击了右侧按钮
- ④ 此时我们使用手动调用右侧按钮点击事件 `arrow_r.click()`
- ⑤ 鼠标经过focus就停止定时器
- ⑥ 鼠标离开focus就开启定时器

## 5. 常见网页特效案例

### 5.1 节流阀

防止轮播图按钮连续点击造成播放过快。

节流阀目的：当上一个函数动画内容执行完毕，再去执行下一个函数动画，让事件无法连续触发。

核心实现思路：利用回调函数，添加一个变量来控制，锁住函数和解锁函数。

开始设置一个变量 `var flag = true;`

`if(flag) {flag = false; do something}`    关闭水龙头

利用回调函数 动画执行完毕， `flag = true`    打开水龙头

## 5. 常见网页特效案例



### 案例：返回顶部

滚动窗口至文档中的特定位置。

```
window.scroll(x, y)
```

注意，里面的x和y 不跟单位，直接写数字



## 5. 常见网页特效案例



### 案例分析

- ① 带有动画的返回顶部
- ② 此时可以继续使用我们封装的动画函数
- ③ 只需要把所有的left 相关的值 改为 跟 页面垂直滚动距离相关就可以了
- ④ 页面滚动了多少，可以通过 `window.pageYOffset` 得到
- ⑤ 最后是页面滚动，使用 `window.scroll(x,y)`

## 5. 常见网页特效案例



### 案例：筋斗云案例

鼠标经过某个小li，筋斗云跟这到当前小li位置

鼠标离开这个小li，筋斗云复原为原来的位置

鼠标点击了某个小li，筋斗云就会留在点击这个小li 的位置

## 5. 常见网页特效案例



### 案例分析

- ① 利用动画函数做动画效果
- ② 原先筋斗云的起始位置是0
- ③ 鼠标经过某个小li，把当前小li的 offsetLeft 位置 做为目标值即可
- ④ 鼠标离开某个小li，就把目标值设为 0
- ⑤ 如果点击了某个小li，就把li当前的位置存储起来，做为筋斗云的起始位置



黑马程序员

[www.itheima.com](http://www.itheima.com)

传智播客旗下高端IT教育品牌