
PC 端网页特效

目录 Contents

- 动画函数封装
- 常见网页特效案例

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

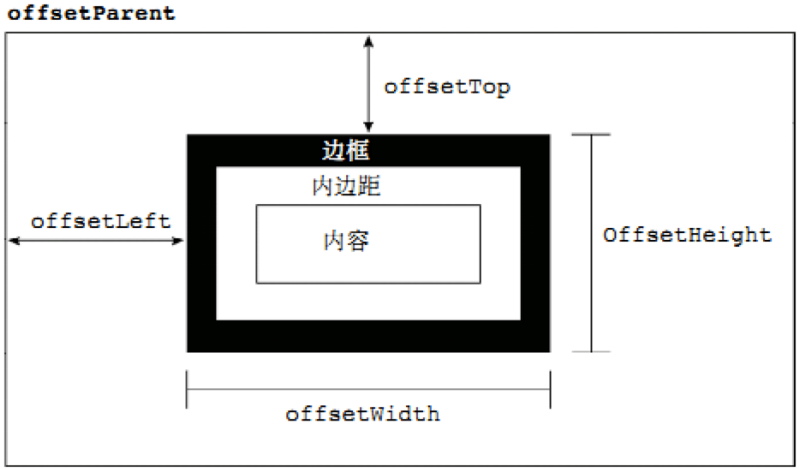
- style 只能得到行内样式表中的样式值
- style.width 获得的是带有单位的字符串
- style.width 获得不包含padding和border 的值
- style.width 是可读写属性，可以获取也可以赋值
- 所以，我们想要给元素更改值，则需要用style改变

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）



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

- 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：(目标值 - 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

- 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