(接notes29)

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：
让盒子每次移动的距离慢慢变小，速度就会慢慢落下来

核心算法:
(目标值-现在的位置)/10 作为每次移动的距离步长

停止的条件是:
让当前盒子的位置等于目标位置就停止定时器

4.5. 动画函数多个目标值之间移动

可以让动画函数从800到500
当我们点击按钮的时候，判断步长是正值还是负值
如果是正值，步长往大了取整
如果是负值，步长往小了取整

```
step = step > 0 ? Math.ceil(step) : Math.floor(step);
```

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，在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

4.7. 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)引入js文件

5. 常见网页特效案例

案例：网页轮播图

轮播图也称焦点图，是网页中比较常见的网页特效

功能需求：

- 鼠标经过轮播图模块，左右按钮显示，离开隐藏左右按钮
- 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
- 图片播放的同时，下面小圆圈模块跟随一起变化
- 点击小圆圈，可以播放相应图片
- 鼠标不经过轮播图，轮播图也会自动播放图片

案例分析：

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
此时需要添加load事件

动态生成小圆圈：

核心思路：小圆圈的个数要跟图片张数一致
所以首先得到ul里面的图片的张数（图片放入li里面，所以就是li的个数）

小圆圈的排他思想：

点击当前小圆圈，就添加current类
其余的小圆圈就移除这个current类

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

```
点击小圆圈滚动图片：
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
使用动画函数的前提，该元素必须有定位
注意是ul移动而不是li
```

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作为ul移动距离

点击右侧按钮一次，就让图片滚动一张：
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

图片无缝滚动原理：

把ul第一个li复制一份，放到ul的最后面
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
同时num赋值为0，可以重新开始滚动图片了

克隆第一张图片：
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
添加到ul最后面appendChild

点击右侧按钮，小圆圈跟随变化
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
但是图片有5张，小圆圈只有4个，必须添加一个判断条件
如果circle==4就重新复原为0

自动播放功能：

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

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

案例：返回顶部

滚动窗口至文档中的特定位置
window.scroll(x,y)

带有动画的返回顶部：
此时我们可以继续使用我们封装的动画特效
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

页面滚动了多少，可以通过window.pageYOffset得到
最后是页面滚动，使用window.scroll(x,y)

案例：筋斗云案例

- 鼠标经过某个li，筋斗云跟这到当前li位置
- 鼠标离开这个li，筋斗云复原为原来的位置
- 鼠标点击了某个li筋斗云就会留在点击这个li的位置

案例分析：

- 利用动画函数做动画效果
- 原先筋斗云的起始位置是0
- 鼠标进过某个li，把当前li的offsetLeft位置作为目标值即可
- 鼠标离开某个li，就把目标值设为0
- 如果点击了某个li，就把li当前位置储存起来，作为筋斗云的起始位置

移动端网页特效

1. 触屏事件

1.1. 触屏事件概述

移动端浏览器兼容性较好，我们不需要考虑以前的js兼容性的问题，可以放心的使用原生js书写效果，但是移动端也有自己独特的地方，比如触屏事件touch（也称触摸事件），Android和iOS都有

touch对象代表一个触摸点，触摸点可能是一根手指，也可能是一根触摸笔。触屏事件可响应用户手指（或触摸笔）对屏幕或者触控板操作

常见的触屏事件如下：
（图：常见触屏事件）

触屏touch事件	说明
touchstart	手指触摸到一个 DOM 元素时触发
touchmove	手指在一个 DOM 元素上滑动时触发
touchend	手指从一个 DOM 元素上移开时触发

1.2. 触摸事件对象（TouchEvent）

TouchEvent是一类描述手指在触摸平面（触摸屏、触摸板）的状态变化的事件，这类事件用于描述一个或多个触点，使开发者可以检测触点的移动，触点的增加和减少，等等

touchstart touchmove touchend 三个事件都各自有事件对象
因为我们一般都是触摸元素，所以最常使用的是**targetTouches**

1.3. 移动端拖动元素

touchstart touchmove touchend 可以实现拖动元素

但是拖动元素需要当前手指的坐标值 我们可以使用targetTouches[0]里面的pageX和pageY

移动端拖动的原理：手指移动中，计算出手指移动的距离。然后用盒子原来的位置+手指移动的距离

手指移动的距离：手指滑动的位置减去手指刚开始触摸的位置

拖动元素三步曲：

- 触摸元素touchstart：获取手指的初始坐标，同时获得盒子原来的位置
- 移动手指touchmove：计算手指的滑动距离，并且移动盒子
- 离开手指touchend:

注意：
手指移动也会触发滚动屏幕所以这里要阻止默认的屏幕滚动**e.preventDefault();**