轮播图

匀速animate函数

```
     function animate(obj, target) {
            obj.timeId = setInterval(function() {
                var step = 10;
                step = target - obj.offsetLeft > 0 ? step : -step;
                if (Math.abs(target - obj.offsetLeft) < Math.abs(step)) {
                    clearInterval(obj.timeId);
                    obj.style.left = target + "px";
                    return false;
                }
                obj.style.left = obj.offsetLeft + step + "px"
            }, 20)
        }
```

封装变速animate函数

```
function animate(obj, target, callback) {
    // console.log(callback);  callback = function() {}  调用的时候 callback()
    // 先清除以前的定时器，只保留当前的一个定时器执行
    clearInterval(obj.timeId);
    obj.timeId = setInterval(function() {
        // 步长值写到定时器的里面
        // 把我们步长值改为整数 不要出现小数的问题
        //var step = Math.ceil((target - obj.offsetLeft) / 10);
        var step = (target - obj.offsetLeft) / 10;
        step = step > 0 ? Math.ceil(step) : Math.floor(step);

        if (obj.offsetLeft == target) {
            // 停止动画 本质是停止定时器
            clearInterval(obj.timeId);
            // 回调函数写到定时器结束里面
            // if (callback) {
            //     // 调用函数
            //     callback();
            // }

            callback && callback();
        }
        // 把每次加1 这个步长值改为一个慢慢变小的值  步长公式：(目标值 - 现在的位置) / 10
        obj.style.left = obj.offsetLeft + step + 'px';
    }, 15);
}
```

index.js

```
window.addEventListener('load', function() {
    // 1. 获取元素
    var arrow_l = document.querySelector('.arrow-l');
    var arrow_r = document.querySelector('.arrow-r');
    var focus = document.querySelector('.focus');
    var focusWidth = focus.offsetWidth;
    // 2. 鼠标经过focus 就显示隐藏左右按钮
    focus.addEventListener('mouseenter', function() {
        arrow_l.style.display = 'block';
        arrow_r.style.display = 'block';
        clearInterval(timer);
        timer = null; // 清除定时器变量
    });
    focus.addEventListener('mouseleave', function() {
        arrow_l.style.display = 'none';
        arrow_r.style.display = 'none';
        timer = setInterval(function() {
            //手动调用点击事件
            arrow_r.click();
        }, 2000);
    });
    // 3. 动态生成小圆圈  有几张图片，我就生成几个小圆圈
    var ul = focus.querySelector('ul');
    var ol = focus.querySelector('.circle');
    // console.log(ul.children.length);
    for (var i = 0; i < ul.children.length; i++) {
        // 创建一个小li
        var li = document.createElement('li');
        // 记录当前小圆圈的索引号 通过自定义属性来做
        li.setAttribute('index', i);
        // 把小li插入到ol 里面
        ol.appendChild(li);
        // 4. 小圆圈的排他思想 我们可以直接在生成小圆圈的同时直接绑定点击事件
        li.addEventListener('click', function() {
            // 干掉所有人 把所有的小li 清除 current 类名
            for (var i = 0; i < ol.children.length; i++) {
                ol.children[i].className = '';
            }
            // 留下我自己  当前的小li 设置current 类名
            this.className = 'current';
            // 5. 点击小圆圈，移动图片 当然移动的是 ul
            // ul 的移动距离 小圆圈的索引号 乘以 图片的宽度 注意是负值
            // 当我们点击了某个小li 就拿到当前小li 的索引号
            var index = this.getAttribute('index');
            // 当我们点击了某个小li 就要把这个li 的索引号给 num
            num = index;
            // 当我们点击了某个小li 就要把这个li 的索引号给 circle
            circle = index;
            // num = circle = index;
            console.log(focusWidth);
            console.log(index);
            //如果想要ul动起来。ul一定要有定位

            animate(ul, -index * focusWidth);
```

```
        })
    }
    // 把ol里面的第一个小li设置类名为 current
    ol.children[0].className = 'current';
    // 6. 克隆第一张图片(li)放到ul 最后面
    var first = ul.children[0].cloneNode(true);
    ul.appendChild(first);
    // 7. 点击右侧按钮， 图片滚动一张
    var num = 0;
    // circle 控制小圆圈的播放
    var circle = 0;
    // flag 节流阀
    var flag = true;
    arrow_r.addEventListener('click', function() {
        if (flag) {
            flag = false; // 关闭节流阀
            // 如果走到了最后复制的一张图片，此时 我们的ul 要快速复原 left 改为 0
            if (num == ul.children.length - 1) {
                ul.style.left = 0;
                num = 0;
            }
            num++;
            animate(ul, -num * focusWidth, function() {
                flag = true; // 打开节流阀
            });
            // 8. 点击右侧按钮，小圆圈跟随一起变化 可以再声明一个变量控制小圆圈的播放
            circle++;
            // 如果circle == 4 说明走到最后我们克隆的这张图片了 我们就复原
            if (circle == ol.children.length) {
                circle = 0;
            }
            // 调用函数
            circleChange();
        }
    });
    // 9. 左侧按钮做法
    arrow_l.addEventListener('click', function() {
        if (flag) {
            flag = false;
            if (num == 0) {
                num = ul.children.length - 1;
                ul.style.left = -num * focusWidth + 'px';
            }
            num--;
            animate(ul, -num * focusWidth, function() {
                flag = true;
            });
            // 点击左侧按钮，小圆圈跟随一起变化 可以再声明一个变量控制小圆圈的播放
            circle--;
            // 如果circle < 0  说明第一张图片，则小圆圈要改为第4个小圆圈（3）
            // if (circle < 0) {
            //     circle = ol.children.length - 1;

            // }
```

```
            circle = circle < 0 ? ol.children.length - 1 : circle;
            // 调用函数
            circleChange();
        }
    });
    function circleChange() {
        // 先清除其余小圆圈的current类名
        for (var i = 0; i < ol.children.length; i++) {
            ol.children[i].className = '';
        }
        // 留下当前的小圆圈的current类名
        ol.children[circle].className = 'current';
    }

    // 10. 自动播放轮播图
    var timer = setInterval(function() {
        //手动调用点击事件
        arrow_r.click();
    }, 2000);
})
```

详细版

```
    <script>
        var arrow_l = document.querySelector('.arrow-l');
        var arrow_r = document.querySelector('.arrow-r');
        var focus = document.querySelector('.focus');
        var focusWidth = focus.offsetWidth;
        // 2. 鼠标经过focus 就显示隐藏左右按钮
        focus.addEventListener('mouseenter', function() {
            arrow_l.style.display = 'block';
            arrow_r.style.display = 'block';
        });
        focus.addEventListener('mouseleave', function() {
            arrow_l.style.display = 'none';
            arrow_r.style.display = 'none';
        });
        var ul = focus.querySelector('ul');
        var ol = focus.querySelector('.circle');
        // console.log(ul.children.length);
        for (var i = 0; i < ul.children.length; i++) {
            // 创建一个小li
            var li = document.createElement('li');
            li.setAttribute('index', i);
            ol.appendChild(li);

            li.addEventListener('click', function() {
                // 干掉所有人 把所有的小li 清除 current 类名
                for (var i = 0; i < ol.children.length; i++) {
                    ol.children[i].className = '';
                }
                // 留下我自己  当前的小li 设置current 类名
                this.className = 'current';
```

```
            var index = this.getAttribute('index');
            animate(ul, -index * focusWidth);
        })

    }

    // 把ol里面的第一个小li设置类名为 current
    ol.children[0].className = 'current';
    // 6. 克隆第一张图片(li)放到ul 最后面
    var first = ul.children[0].cloneNode(true);
    ul.appendChild(first);
    // 7. 点击右侧按钮， 图片滚动一张
    var num = 0;
    // circle 控制小圆圈的播放
    var circle = 0;
    arrow_r.addEventListener('click', function() {


        // 如果走到了最后复制的一张图片，此时 我们的ul 要快速复原 left 改为 0
        if (num == ul.children.length - 1) {
            ul.style.left = 0;
            num = 0;
        }
        num++;
        animate(ul, -num * focusWidth, function() {
            flag = true; // 打开节流阀
        });
        // 8. 点击右侧按钮，小圆圈跟随一起变化 可以再声明一个变量控制小圆圈的播放
        circle++;
        // 如果circle == 4 说明走到最后我们克隆的这张图片了 我们就复原
        if (circle == ol.children.length) {
            circle = 0;
        }


        // 先清除其余小圆圈的current类名
        for (var i = 0; i < ol.children.length; i++) {
            ol.children[i].className = '';
        }
        // 留下当前的小圆圈的current类名
        ol.children[circle].className = 'current';
    });


    // 9. 左侧按钮做法
    arrow_l.addEventListener('click', function() {

        if (num == 0) {
            num = ul.children.length - 1;
            ul.style.left = -num * focusWidth + 'px';
        }
        num--;

        animate(ul, -num * focusWidth, function() {
```

```
                flag = true;
            });
            // 点击左侧按钮，小圆圈跟随一起变化 可以再声明一个变量控制小圆圈的播放
            circle--;
            // 如果circle < 0  说明第一张图片，则小圆圈要改为第4个小圆圈（3）
            // if (circle < 0) {
            //     circle = ol.children.length - 1;
            // }
            circle = circle < 0 ? ol.children.length - 1 : circle;
            // 调用函数

            // 先清除其余小圆圈的current类名
            for (var i = 0; i < ol.children.length; i++) {
                ol.children[i].className = '';
            }
            // 留下当前的小圆圈的current类名
            ol.children[circle].className = 'current';

        });


        // 10. 自动播放轮播图
        var timer = setInterval(function() {
            //手动调用点击事件
            arrow_r.click();
        }, 2000);
    </script>
```

仿淘宝返回顶部

```
<style>
        .slider-bar {
            position: absolute;
            left: 50%;
            top: 300px;
            margin-left: 600px;
            width: 45px;
            height: 130px;
            background-color: pink;
        }

        .w {
            width: 1200px;
            margin: 10px auto;
        }

        .header {
            height: 150px;
            background-color: purple;
        }


        .banner {
```

```
            height: 250px;
            background-color: skyblue;
        }

        .main {
            height: 1000px;
            background-color: yellowgreen;
        }

        span {
            display: none;
            position: absolute;
            bottom: 0;
        }
    </style>
</head>

<body>
    <div class="slider-bar">
        <span class="goBack">返回顶部</span>
    </div>
    <div class="header w">头部区域</div>
    <div class="banner w">banner区域</div>
    <div class="main w">主体部分</div>
    <script>
        //1. 获取元素
        var sliderbar = document.querySelector('.slider-bar');
        var banner = document.querySelector('.banner');
        // banner.offestTop 就是被卷去头部的大小 一定要写到滚动的外面
        var bannerTop = banner.offsetTop
            // 当我们侧边栏固定定位之后应该变化的数值
        var sliderbarTop = sliderbar.offsetTop - bannerTop;
        // 获取main 主体元素
        var main = document.querySelector('.main');
        var goBack = document.querySelector('.goBack');
        var mainTop = main.offsetTop;
        // 2. 页面滚动事件 scroll
        document.addEventListener('scroll', function() {
                // console.log(11);
                // window.pageYOffset 页面被卷去的头部
                // console.log(window.pageYOffset);
                // 3 .当我们页面被卷去的头部大于等于了 172 此时 侧边栏就要改为固定定位
                if (window.pageYOffset >= bannerTop) {
                    sliderbar.style.position = 'fixed';
                    sliderbar.style.top = sliderbarTop + 'px';
                } else {
                    sliderbar.style.position = 'absolute';
                    sliderbar.style.top = '300px';
                }
                // 4. 当我们页面滚动到main盒子，就显示 goback模块
                if (window.pageYOffset >= mainTop) {
                    goBack.style.display = 'block';

                } else {
```

```
                goBack.style.display = 'none';
            }

        })
        // 3. 当我们点击了返回顶部模块，就让窗口滚动的页面的最上方
    goBack.addEventListener('click', function() {
        // 里面的x和y 不跟单位的 直接写数字即可
        // window.scroll(0, 0);
        // 因为是窗口滚动 所以对象是window
        animate(window, 0);
    });
    // 动画函数
    function animate(obj, target, callback) {
        // console.log(callback);  callback = function() {}  调用的时候 callback()

        // 先清除以前的定时器，只保留当前的一个定时器执行
        clearInterval(obj.timer);
        obj.timer = setInterval(function() {
            // 步长值写到定时器的里面
            // 把我们步长值改为整数 不要出现小数的问题
            // var step = Math.ceil((target - obj.offsetLeft) / 10);
            var step = (target - window.pageYOffset) / 10;
            step = step > 0 ? Math.ceil(step) : Math.floor(step);
            if (window.pageYOffset == target) {
                // 停止动画 本质是停止定时器
                clearInterval(obj.timer);
                // 回调函数写到定时器结束里面
                // if (callback) {
                //     // 调用函数
                //     callback();
                // }
                callback && callback();
            }
            // 把每次加1 这个步长值改为一个慢慢变小的值  步长公式：(目标值 - 现在的位置) / 10
            // obj.style.left = window.pageYOffset + step + 'px';
            window.scroll(0, window.pageYOffset + step);
        }, 15);
    }
</script>
```

筋斗云导航栏

```
    <style>
        * {
            margin: 0;
            padding: 0
        }

        ul {
            list-style: none;
        }
```

```
        body {
            background-color: black;
        }

        .c-nav {
            width: 900px;
            height: 42px;
            background: #fff url(images/rss.png) no-repeat right center;
            margin: 100px auto;
            border-radius: 5px;
            position: relative;
        }

        .c-nav ul {
            position: absolute;
        }

        .c-nav li {
            float: left;
            width: 83px;
            text-align: center;
            line-height: 42px;
        }

        .c-nav li a {
            color: #333;
            text-decoration: none;
            display: inline-block;
            height: 42px;
        }

        .c-nav li a:hover {
            color: white;
        }

        .c-nav li.current a {
            color: #0dff1d;
        }

        .cloud {
            position: absolute;
            left: 0;
            top: 0;
            width: 83px;
            height: 42px;
            background: url(images/cloud.gif) no-repeat;
        }
    </style>
    <script src="animate.js"></script>
    <script>
        window.addEventListener('load', function() {
            // 1. 获取元素

            var cloud = document.querySelector('.cloud');
```

```
            var c_nav = document.querySelector('.c-nav');
            var lis = c_nav.querySelectorAll('li');
            // 2. 给所有的小li绑定事件
            // 这个current 做为筋斗云的起始位置
            var current = 0;
            for (var i = 0; i < lis.length; i++) {
                // (1) 鼠标经过把当前小li 的位置做为目标值
                lis[i].addEventListener('mouseenter', function() {
                    animate(cloud, this.offsetLeft);
                });
                // (2) 鼠标离开就回到起始的位置
                lis[i].addEventListener('mouseleave', function() {
                    animate(cloud, current);
                });
                // (3) 当我们鼠标点击，就把当前位置做为目标值
                lis[i].addEventListener('click', function() {
                    current = this.offsetLeft;
                });
            }
        })
    </script>
</head>

<body>
    <div id="c_nav" class="c-nav">
        <span class="cloud"></span>
        <ul>
            <li class="current"><a href="#">首页新闻</a></li>
            <li><a href="#">师资力量</a></li>
            <li><a href="#">活动策划</a></li>
            <li><a href="#">企业文化</a></li>
            <li><a href="#">招聘信息</a></li>
            <li><a href="#">公司简介</a></li>

        </ul>
    </div>
</body>
```