目标

学习目标

- 能够写出带有动画的返回顶部案例
- 能够写出筋斗云案例
- 能够写出网页轮播图案例

动画函数封装

也就是将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

封装步骤如下:

```
1. 单独新建一个JS文件
2. 再把我们的animate函数粘贴进去
3. 在要用这个animate函数的页面引入这个JS文件
```

我们封装到`animate.js`文件中

```
// 封装动画函数
// obj是需要做动画的对象
// target是需要移动到哪里,也就是目标值
// callback回调函数,做完动画以后执行的函数  可选参数
function animate(obj,target,callback){
    // 先清除原有的定时器
    clearInterval( obj.timer );

    // 开启定时器
     obj.timer = setInterval(function(){
        // 利用公式得到缓慢动画的步长值
        var step = (target - obj.offsetLeft) / 10;
        // 如果步长值大于0,则向上取整;小于0,则向下取整
        step = step > 0 ? Math.ceil( step) : Math.floor( step );
        // 判断是否达到目标值
        if( obj.offsetLeft == target ){
            // 清除定时器
            clearInterval( obj.timer );
            // 清除定时器以后就代表动画执行完毕,调用回调函数
            if( callback ){
                callback();
            }
        }else{
            obj.style.left = obj.offsetLeft + step + "px"
        }
    },15)
}
```

举例:

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <!-- 引入animate.js文件 -->
    <script src="js/animate.js"></script>
    <style>
        *{
            margin: 0;
            padding: 0;
        }
        p{
            width: 100px;
            height: 100px;
            background: skyblue;
            position: absolute;
            left: 0;
            top: 0;
        }
        button{
            margin-top: 150px;
        }
    </style>
</head>
<body>
    <p>今天天气还可以</p>
    <button>按钮</button>
    <script>
        var btn = document.querySelector("button");
        var p   = document.querySelector("p");
        btn.onclick =  function(){
            // animate函数在animate.js里面中定义了,所以只要引入了animate.js文件,就可以使用animate.js文件里面的东西了
            animate(p,800,function(){
                alert("动画结束了~");
            })
        }
    </script>
</body>
</html>
```

案例-侧边栏动画显示

代码:

```
<!DOCTYPE html>
<html lang="en">


<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Document</title>
    <style>
        .sliderbar {
            position: fixed;
            right: 0;
            bottom: 100px;
            width: 40px;
            height: 40px;
            text-align: center;
            line-height: 40px;
            cursor: pointer;
            color: #fff;
        }

        .con {
            position: absolute;
            left: 0;
            top: 0;
            width: 200px;
            height: 40px;
            background-color: purple;
            z-index: -1;
        }
    </style>
</head>


<body>
    <div class="sliderbar">
        <span>←</span>
        <div class="con">问题反馈</div>
    </div>
    <!-- 引入外部js文件的script标签中,不能写js代码 -->
    <script src="../js/animate.js"></script>


    <script>
        // 获取相关对象
        var sliderbar = document.querySelector(".sliderbar");
        var con = document.querySelector(".con");
        var span = document.querySelector("span");
        // 鼠标移上.sliderbar以后 .con需要动画左移一段距离, 执行完动画以后,把span的内容改成→
        sliderbar.onmouseover = function(){
            animate(con, -160 ,function(){
                span.innerHTML = "→";
            });
        }

        // 鼠标移出.sliderbar以后 .con需要动画移回到原来位置, 执行完动画以后,把span的内容改成←
        sliderbar.onmouseout = function(){
            animate(con, 0 ,function(){
                span.innerHTML = "←";
            });
        }
    </script>
```

```
</body>


</html>
```

window.scroll滚动窗口到文档特定位置

```
1 滚动窗口至文档中的特定位置
2 window.scroll(x,y)
3 注意:里面的x和y不需要带px单位  直接写数字即可
```

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <style>
        body{
            /* 给body设置宽度高度2000是为了让页面出现滚动条 */
            height: 2000px;
            width: 2000px;
            background:skyblue;
        }
        div{
            position: fixed;
            left:50%;
            top:50%;
        }
    </style>
</head>
<body>
    <div>
        <button>按钮1</button>
        <button>按钮2</button>
        <button>按钮3</button>
    </div>


    <script>
        // 滚动窗口至文档中的特定位置
        // window.scroll(水平滚动条的位置,垂直滚动条的位置);
        // window.scroll(x,y);
        // 注意:里面的x和y不需要带px单位 直接写数字即可
        // 说白了window.scroll(x,y);就是控制滚动滚动的距离
        // 一般的网页的滚动只有垂直的,没有水平的


        // 获取所有按钮对象
        var btns = document.querySelectorAll("button");
        btns[0].onclick = function(){
            // window.scroll(0,0)代表把网页水平滚动条跟垂直滚动条移动到 0 0位置
            window.scroll(0,0);
        }
        btns[1].onclick = function(){
            // window.scroll(0,0)代表把网页水平滚动条跟垂直滚动条移动到 0 0位置
            window.scroll(0,100);
        }
        btns[2].onclick = function(){
            // 实现点击一次按钮,垂直滚动条就向上移动几个像素


            // 先获取当前页面被卷去的距离
            var pageScroll = document.body.scrollTop || document.documentElement.scrollTop || window.pageYOffset;


            // 然后再把这个页面被卷去的距离+50
            // pageScroll = pageScroll + 50;
            // console.log( pageScroll );


            // 然后再把这个页面被卷去的距离-50
            pageScroll = pageScroll - 50;
            console.log( pageScroll );



            // 再通过window.scroll(x,y)  移动滚动条的位置
            // 如果pageScroll的值越来越大,垂直滚动条会越往下
            // 如果pageScroll的值越来越小,垂直滚动条会越往上
            window.scroll(0,pageScroll);
        }
    </script>
</body>
</html>
```

封装animate_scroll函数

分析:

1. 带有动画的返回顶部
2. 此时可以继续使用我们之前封装的动画函数
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)

代码:

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <style>
        *{
            margin: 0;
            padding: 0;
        }
        body{
            height: 3000px;
            background: skyblue;
        }
        button{
            position: fixed;
            left: 50%;
            top: 50%;
        }
    </style>
</head>
<body>
    <button>点我动画回到顶部</button>


    <script>
        // 封装动画滚动函数
        // 思路:把之前的动画函数相关的属性改成页面被卷去的距离
        // obj是需要做动画的对象
        // target是需要移动到哪里,也就是目标值
        // callback回调函数,做完动画以后执行的函数  可选参数
        function animate_scroll(obj,target,callback){
            // 先清除原有的定时器
            clearInterval( obj.timer );


            // 开启定时器
            obj.timer = setInterval(function(){
                // 如果放在定时器的外面,调用一次animate_scroll函数,才会获取一次页面滚动距离
                // 所以要把获取页面被卷去的距离放在定时器里面,这样每次才可以获取最新的被页面卷去的值
                var pageScroll = window.pageYOffset || document.body.scrollTop || document.documentElement.scrollTop;

                // 利用公式得到缓慢动画的步长值
                var step = (target - pageScroll) / 10;

                // 如果步长值大于0,则向上取整;小于0,则向下取整
                step = step > 0 ? Math.ceil( step) : Math.floor( step );


                // 判断是否达到目标值
                if( pageScroll == target ){
                    // 清除定时器
                    clearInterval( obj.timer );
                    // 清除定时器以后就代表动画执行完毕,调用回调函数
                    if( callback ){
                        callback();
                    }
                }else{
                    var scrollY = pageScroll + step;
                    window.scroll(0, scrollY );
                }
            },15)
        }


        // 获取按钮对象
        var btn = document.querySelector("button");
        btn.onclick = function(){
            animate_scroll(window,0,function(){
                alert("不得了,不得了,不得了,页面滚动到顶部了~");
            });
        }
    </script>
</body>
</html>
```

animate_scroll.js

```
// 封装动画滚动函数
// 思路:把之前的动画函数相关的属性改成页面被卷去的距离
// obj是需要做动画的对象
// target是需要移动到哪里,也就是目标值
// callback回调函数,做完动画以后执行的函数  可选参数
function animate_scroll(obj,target,callback){
    // 先清除原有的定时器
    clearInterval( obj.timer );

     // 开启定时器
     obj.timer = setInterval(function(){
         // 如果放在定时器的外面,调用一次animate_scroll函数,才会获取一次页面滚动距离
         // 所以要把获取页面被卷去的距离放在定时器里面,这样每次才可以获取最新的被页面卷去的值
         var pageScroll = window.pageYOffset || document.body.scrollTop || document.documentElement.scrollTop;

         // 利用公式得到缓慢动画的步长值
         var step = (target - pageScroll) / 10;

         // 如果步长值大于0,则向上取整;小于0,则向下取整
         step = step > 0 ? Math.ceil( step) : Math.floor( step );

         // 判断是否达到目标值
         if( pageScroll == target ){
             // 清除定时器
             clearInterval( obj.timer );
             // 清除定时器以后就代表动画执行完毕,调用回调函数
             if( callback ){
                 callback();
             }
         }else{
             var scrollY = pageScroll + step;
             window.scroll(0, scrollY );
         }
     },15)
}


```

案例:带有动画的返回顶部

animate_scroll.js

```
// 封装动画滚动函数
// 思路:把之前的动画函数相关的属性改成页面被卷去的距离
// obj是需要做动画的对象
// target是需要移动到哪里,也就是目标值
// callback回调函数,做完动画以后执行的函数  可选参数
function animate_scroll(obj,target,callback){
    // 先清除原有的定时器
    clearInterval( obj.timer );


    // 开启定时器
    obj.timer = setInterval(function(){
        // 如果放在定时器的外面,调用一次animate_scroll函数,才会获取一次页面滚动距离
        // 所以要把获取页面被卷去的距离放在定时器里面,这样每次才可以获取最新的被页面卷去的值
        var pageScroll = window.pageYOffset || document.body.scrollTop || document.documentElement.scrollTop;

        // 利用公式得到缓慢动画的步长值
        var step = (target - pageScroll) / 10;

        // 如果步长值大于0,则向上取整;小于0,则向下取整
        step = step > 0 ? Math.ceil( step) : Math.floor( step );


        // 判断是否达到目标值
        if( pageScroll == target ){
            // 清除定时器
            clearInterval( obj.timer );
            // 清除定时器以后就代表动画执行完毕,调用回调函数
            if( callback ){
                callback();
            }
        }else{
            var scrollY = pageScroll + step;
            window.scroll(0, scrollY );
        }
    },15)
}
```

代码:

```
<!DOCTYPE html>
<html lang="en">


<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Document</title>
    <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;
        }

        .goBack {
            display: none;
            position: absolute;
            bottom: 0;
            text-decoration: none;
            color:gray;
        }
    </style>
</head>


<body>
    <div class="slider-bar">
        <!-- 如果给a链接的href属性设置了#,是代表空链接,但是点击以后会马上跳到顶部 -->
        <!-- 还有另一种空链接  href="javascript:void(0)" -->
        <!-- <a href="#" class="goBack">返回顶部</a> -->
        <a href="javascript:void(0)" class="goBack">返回顶部</a>
    </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>
</body>
<!-- 引入animate_scroll.js -->
<script src="js/animate_scroll.js"></script>
<script>
    function getScroll(){
        return {
            left : window.pageXOffset || document.body.scrollLeft || document.documentElement.scrollLeft || 0,
            top  : window.pageYOffset || document.body.scrollTop || document.documentElement.scrollTop || 0
        }
    }


    // 获取.header对象
    var objHeader = document.querySelector(".header");
    // 获取.banner对象
    var objBanner = document.querySelector(".banner");
    // 获取侧边栏对象
    var sliderBar = document.querySelector(".slider-bar");
    var sliderBarOffsetTop = sliderBar.offsetTop;


    // 为了解决跳跃问题,算出侧边栏顶部距离.banner层顶部的距离
    var sliderBarTop =  sliderBar.offsetTop - objBanner.offsetTop;


    // 获取.main对象
    var main = document.querySelector(".main");
    var mainTop = main.offsetTop;


    // 获取返回顶部对象
    var goBack = document.querySelector(".goBack");


    // 页面滚动事件
    window.onscroll = function(){
        // 获取页面滚动的距离
        var pageTop =  getScroll().top ;
        // top这个变量名最好也不要用,因为在window对象下,也有一个叫做top的属性
        // console.log( PageTop );

        // 1. 当页面滚动到.banner层的时候,把侧边栏变成固定定位;如果在.banner层之前侧边栏都是绝对定位
        if( pageTop > objBanner.offsetTop ){
            sliderBar.style.position = "fixed";
```

```
            // 为什么会跳跃,因为设置了固定定位,但是没有设置垂直方向的偏移量
            sliderBar.style.top = sliderBarTop + "px";
        }else{
            sliderBar.style.position = "absolute";
            sliderBar.style.top = sliderBarOffsetTop + "px";
        }


        // 2. 当页面滚动到主体内容部分时候,显示返回顶部a链接;当页面没有滚动到主体内容部分时候,则隐藏返回顶部a链接
        if( pageTop > mainTop ){// 如果当前滚动的距离大于了main层的上外边距
            goBack.style.display = "block";// 显示返回顶部链接
        }else{
            goBack.style.display = "none";// 隐藏返回顶部链接
        }


        // 3. 点击goBack动画返回顶部
        goBack.onclick = function(){
            animate_scroll(window,0,function(){
                alert("回到顶部了~");
            })
        }
    }
</script>
</html>
```

案例:筋头云案例

要求:

1. 鼠标经过某个li,筋斗云就跟着到这个小li所在位置
2. 鼠标离开某个li,筋斗云复原为原来的位置
3. 鼠标点击某个li,筋斗云就会留在点击这个小li的位置

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

代码:

```
<!DOCTYPE html>
<html>


<head lang="en">
    <meta charset="UTF-8">
    <title></title>
    <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>
</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>
            <li><a href="#">我是佩奇</a></li>
            <li><a href="#">啥是佩奇</a></li>
        </ul>
    </div>
    <script src="../js/animate.js"></script>
    <script>
        // 获取对象
        var lis = document.querySelectorAll("#c_nav ul li");
        var cloud = document.querySelector(".cloud");


        // 用一个变量保存鼠标点击以后的值,这个值可以用于鼠标移出li以后,云要去到哪里
        var current = 0;
        for(var i=0;i<lis.length;i++){
            // 鼠标移上某个li以后,云要动画去到这个li的位置
            lis[i].onmouseover = function(){
                animate(cloud, this.offsetLeft );
            }
            // 鼠标移出某个li以后,云要恢复到指定位置
            lis[i].onmouseout = function(){
                animate(cloud, current );
            }
            // 鼠标点击某个li以后,云要停在对应的li上面
            lis[i].onclick = function(){
                current = this.offsetLeft;
            }
        }

    </script>
</body>


</html>
```

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, 这个效果我们学习,只是为了了解原理,顺便把之前学的知识点综合起来使用,实际工作的时候,有很多轮播图js插件可以使用,很少自己写 比如 swiper 大话主席 bootstrap等等....

功能需求列表:

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。

小技巧: vscode 里面打 ul>li*6>a[href="#"]>img 生成以下代码

```
<ul>
    <li><a href="#"><img src="" alt=""></a></li>
    <li><a href="#"><img src="" alt=""></a></li>
    <li><a href="#"><img src="" alt=""></a></li>
    <li><a href="#"><img src="" alt=""></a></li>
    <li><a href="#"><img src="" alt=""></a></li>
    <li><a href="#"><img src="" alt=""></a></li>
</ul>
```

HTML+CSS静态页面

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <style>
        *{
            margin: 0;
            padding: 0;
        }
        li{
            list-style:none;
        }
        /* 去掉图片底部缝隙 */
        img{
            vertical-align: bottom;
        }
        .w{
            width: 1200px;
            margin:0 auto;
        }
        .banner{
            height: 400px;
            position: relative;
            /* 超出banner的内容隐藏 */
            overflow:hidden;
            margin-top: 50px;
        }
        .banner .arrow_left{
            width: 36px;
            height: 44px;
            position: absolute;
            left: 0;
            top: 50%;
            margin-top: -22px;
            /* 隐藏元素 */
            display: none;
        }
        .banner .arrow_right{
```

```
            width: 36px;
            height: 44px;
            position: absolute;
            right: 0;
            top: 50%;
            margin-top: -22px;
            /* 隐藏元素 */
            display: none;
        }
        /* 为什么要给ul设置600%的宽度,因为我们需要把所有的li浮动成一行,ul必须有足够的宽度才可以放得下 */
        .banner ul {
            width:600%;
        }
        .banner ul li{
            float: left;
        }


        .banner .circle{
            position: absolute;
            bottom: 20px;
            left: 50%;
            width: 120px;
            margin-left: -60px;
        }
        .banner .circle li{
            width: 10px;
            height: 10px;
            background:rgba(255,255,255,0.8);
            border-radius: 50%;
            float: left;
            margin-right: 10px;
            /* 鼠标模拟手势 */
            cursor: pointer;
        }
        .banner .circle li:first-child{
            width: 20px;
            border-radius: 10px;
        }
        .banner .circle li:last-child{
            margin-right: 0;
        }
    </style>
</head>
<body>
    <!-- 最外层的盒子 -->
    <div class="banner w">
        <!-- 左侧按钮 -->
        <img src="images/prev.png" class="arrow_left"/>
        <!-- 右侧按钮 -->
        <img src="images/next.png" class="arrow_right"/>


        <!-- 轮播图片 -->
        <ul>
            <li><a href="#"><img src="images/banner01.jpg"></a></li>
            <li><a href="#"><img src="images/banner02.jpg"></a></li>
            <li><a href="#"><img src="images/banner03.jpg"></a></li>
            <li><a href="#"><img src="images/banner04.jpg"></a></li>
            <li><a href="#"><img src="images/banner05.jpg"></a></li>
            <li><a href="#"><img src="images/banner06.jpg"></a></li>
        </ul>


        <!-- 6个小圆点 -->
        <ol class="circle">
            <li></li>
            <li></li>
            <li></li>
            <li></li>
            <li></li>
            <li></li>
```

```
        </ol>
    </div>
</body>
</html>
```

功能1:鼠标经过轮播图模块，左右按钮显示，离开隐藏左右按钮

1. 添加 window.onload 事件
2. 鼠标经过轮播图模块，左右按钮显示，离开隐藏左右按钮
3. 显示 display:block　　隐藏 display:none;

```
<script>
        // 因为获取对象功能用的非常频繁,我们可以自己封装一个函数获取对象
        // 函数名$是可以正常使用的,没有问题,因为jquery获取对象的时候也是使用$,为后面的知识埋铺垫
        // 封装函数的感觉就是,辛苦一时,快乐很久
        function $( cssStr ){
            var obj = document.querySelectorAll( cssStr );

            if( obj.length == 1){// 判断获取输出的对象长度是否为1
                return obj[0];// 如果长度为1就返回第一个元素即可
            }else{
                return obj;// 如果长度不为值,直接返回整个obj
            }
        }


        // 获取左侧按钮对象
        var arrow_left = $(".arrow_left");
        // 获取右侧按钮对象
        var arrow_right = $(".arrow_right");
        // 获取轮播图对象
        var banner = $(".banner");


        // 功能1:鼠标经过轮播图模块，左右按钮显示，离开隐藏左右按钮
        banner.onmouseover = function(){
            arrow_left.style.display = "block";
            arrow_right.style.display = "block";
        }


        banner.onmouseout = function(){
            arrow_left.style.display = "none";
            arrow_right.style.display = "none";
        }
    </script>
```

功能2:　动态生成指示器中的li(也就是动态生成小圆圈)

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. 首先先把HTML结构.circle下的li都删除掉,然后得到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
```

修改CSS样式

```
/* 为了后面可以点击li高亮 */
        /* .banner .circle li:first-child{
            width: 20px;
            border-radius: 10px;
        } */
        /* 我们单独定义一个类名 .current  */
        .banner .circle .current{
            width: 20px;
            border-radius: 10px;
        }
        .banner .circle li:last-child{
            margin-right: 0;
        }
```

实现代码

```
<script>
        // 因为获取对象功能用的非常频繁,我们可以自己封装一个函数获取对象
        // 函数名$是可以正常使用的,没有问题,因为jquery获取对象的时候也是使用$,为后面的知识埋铺垫
        // 封装函数的感觉就是,辛苦一时,快乐很久
        function $( cssStr ){
            var obj = document.querySelectorAll( cssStr );

            if( obj.length == 1){// 判断获取输出的对象长度是否为1
                return obj[0];// 如果长度为1就返回第一个元素即可
            }else{
                return obj;// 如果长度不为值,直接返回整个obj
            }
        }


        // 获取左侧按钮对象
        var arrow_left = $(".arrow_left");
        // 获取右侧按钮对象
        var arrow_right = $(".arrow_right");
        // 获取轮播图对象
        var banner = $(".banner");
        // 获取小圆点对应的ol对象
        var circle = $(".circle");
        // 获取.banner里面的ul对象
        var bannerUl = $(".banner ul");


        // 功能1:鼠标经过轮播图模块，左右按钮显示，离开隐藏左右按钮
        banner.onmouseover = function(){
            arrow_left.style.display = "block";
            arrow_right.style.display = "block";
        }


        banner.onmouseout = function(){
            arrow_left.style.display = "none";
            arrow_right.style.display = "none";
        }


        // 功能2:    动态生成指示器中的li(也就是动态生成小圆圈)
        // 实现思路   创建节点,添加节点    有几个图片,我们就有几个小圆点,因为图片是放在ul的li标签中,所有ul里面有几个li,就应该有几个小圆点
        // console.log( bannerUl );
        // console.log( bannerUl.children );
        // console.log( bannerUl.children.length );


        for(var i = 0; i < bannerUl.children.length; i++ ){
            var li = document.createElement("li");
            circle.appendChild(li)
        }


        // 单独给.circle层里面第一个li设置current类名
        circle.children[0].className = "current";
    </script>
```

功能3: 被点击小圆圈高亮

1. 小圆圈的排他思想
2. 所有小圆圈都移除这个current类
3. 点击当前小圆圈,就添加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

功能4:点击小圆圈滚动到对应图片

```
1. 点击小圆圈滚动图片
2. 此时用到animate动画函数，将animate.js文件引入
3. 注意是ul 移动 而不是小li
4. 使用动画函数的前提，该元素必须有定位 所以要给.banner ul加个绝对定位
position: absolute;top: 0;left: 0;
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。
 7. 注意设置了ul绝对定位以后,看看左右按钮会不会被挡住
```

引入animated.js文件

```
<!-- 引入animate.js -->
<script src="../js/animate.js"></script>
```

修改对应CSS样式

```css
.banner .arrow_left{
            width: 36px;
            height: 44px;
            position: absolute;
            left: 0;
            top: 50%;
            margin-top: -22px;
            /* 隐藏元素 */
            display: none;
            /* 因为ul设置了绝对定位以后,会压住左侧按钮 */
            z-index:1;
        }
        .banner .arrow_right{
            width: 36px;
            height: 44px;
            position: absolute;
            right: 0;
            top: 50%;
            margin-top: -22px;
            /* 隐藏元素 */
            display: none;
            /* 因为ul设置了绝对定位以后,会压住右侧按钮 */
            z-index:1;
        }

        /* 为什么要给ul设置600%的宽度,因为我们需要把所有的li浮动成一行,ul必须有足够的宽度才可以放得下 */
        .banner ul {
            width:600%;
            /* 如果想要让ul可以移动,需要给ul设置绝对定位 */
            position: absolute;
            left: 0;
            top: 0;
        }
```

```js
// 获取轮播图的宽度
var bannerWidth = banner.offsetWidth;
```

全局变量

```
for(var i = 0; i < bannerUl.children.length; i++ ){
            // 创建节点
            var li = document.createElement("li");

            // 设置自定义属性data-index
            li.setAttribute("data-index",i);

            // 功能3: 被点击小圆圈高亮  排他思想  先清除所有的,再设置自己
            li.onclick = function(){
                // circle.children就是代表circle层下的所有li标签
                for(var j=0;j<circle.children.length;j++){
                    // 清除所有li的class属性值
                    circle.children[j].removeAttribute("class");
                }
                // 单独设置点击的那个li的class属性值
                this.className = "current";


                // 功能4:点击小圆圈滚动到对应图片
                // 1. 点击小圆圈滚动图片
                // 2. 此时用到animate动画函数，将animate.js文件引入
                // 3. 注意是ul 移动 而不是小li
                // 4. 使用动画函数的前提，该元素必须有定位 所以要给.banner ul加个绝对定位
                // position: absolute;top: 0;left: 0;
                // 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。
                // 7. 注意设置了ul绝对定位以后,看看左右按钮会不会被挡住


                // 让.banner里面ul移动 移动多少距离由点击哪个li决定,比如点击第二个li,那就要移动 -1200; 点击了第三个li,要移动-2400...  这个1200其实轮播图的宽度
                // animate(bannerUl,??);

                var index = this.getAttribute("data-index");
                // console.log( index );
                // console.log( bannerWidth );


                animate(bannerUl, -index*bannerWidth );
            }
            // 添加节点
            circle.appendChild(li);
        }
```

功能5: 点击右侧按钮一次，就让图片滚动一张

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 克隆一份，放到ul 的最后面，需要注意修改"ul的宽度"以及"生成小圆点的个数"
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，可以重新开始滚动图片了

提示代码:

```
var num = 0;
 arrow_right.onclick = 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++;
```

```
17      animate(ul,-num*bannerWidth);
18  }
```

```
.banner .arrow_left{
            width: 36px;
            height: 44px;
            position: absolute;
            left: 0;
            top: 50%;
            margin-top: -22px;
            /* 隐藏元素 */
            display: none;
            /* 因为ul设置了绝对定位以后,会压住左侧按钮 */
            z-index:1;
            cursor: pointer;
        }
        .banner .arrow_right{
            width: 36px;
            height: 44px;
            position: absolute;
            right: 0;
            top: 50%;
            margin-top: -22px;
            /* 隐藏元素 */
            display: none;
            /* 因为ul设置了绝对定位以后,会压住右侧按钮 */
            z-index:1;
            cursor: pointer;
        }
```

```html
<!-- 轮播图片 -->
<ul>
    <li><a href="#"><img src="images/banner01.jpg"></a></li>
    <li><a href="#"><img src="images/banner02.jpg"></a></li>
    <li><a href="#"><img src="images/banner03.jpg"></a></li>
    <li><a href="#"><img src="images/banner04.jpg"></a></li>
    <li><a href="#"><img src="images/banner05.jpg"></a></li>
    <li><a href="#"><img src="images/banner06.jpg"></a></li>
    <!-- 为了实现无缝滚动,复制第一个li加到ul后面 -->
    <li><a href="#"><img src="images/banner01.jpg"></a></li>
</ul>
```

```css
/* 为什么要给ul设置600%的宽度,因为我们需要把所有的li浮动成一行,ul必须有足够的宽度才可以放得下 */
.banner ul {
    /* width:600%; */
    /* 因为手动复制了添加了一个li到ul的子元素列表后面,这个时候ul里面就有7个li */
    width: 700%;

    /* 如果想要让ul可以移动,需要给ul设置绝对定位 */
    position: absolute;
    left: 0;
    top: 0;
}
```

```javascript
var num = 0; // 我们会定义一个变量保存现在是第几张图片,如果num=0,就代表现在是第1个图片;如果num=1,就代表现在是第2个图片
arrow_right.onclick = function(){
    // 功能5: 点击右侧按钮一次, 就让图片滚动一张
    // 核心原理:就是点一次右侧按钮,就让ul负一次1200px
    // 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 克隆一份, 放到ul 的最后面, 需要注意修改ul的宽度以及生成小圆点的个数
    // 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: 也就是把 left 为0
    // 6. 同时num 赋值为0, 可以重新开始滚动图片了
    // console.log( num );

    if(num == bannerUl.children.length - 1){// 代表移动到我们克隆的那张图片
        bannerUl.style.left = 0;
        num = 0;
    }
    num++;
    animate(bannerUl, -num*bannerWidth );
}
```

功能6: 克隆ul第一个li,添加到ul的子节点列表最后面

1. 上一个功能中的无缝滚动,我们是直接复制ul中的第一个li实现的,
2. 这样会导致小圆圈多一个,而且不灵活

2. 所以,把刚才我们手动添加的li对应的HTML结构删除掉,以及.banner ul样式的width改回600%,

我们使用js中的克隆技术,克隆第一个li

3. 克隆ul第一个li cloneNode() 加true深克隆,会复制里面的子节点　　如果不加参数或者false就是浅克隆,不会复制里面的子节点

4. 把克隆好的节点添加到ul最后面 appendChild

5. 核心代码

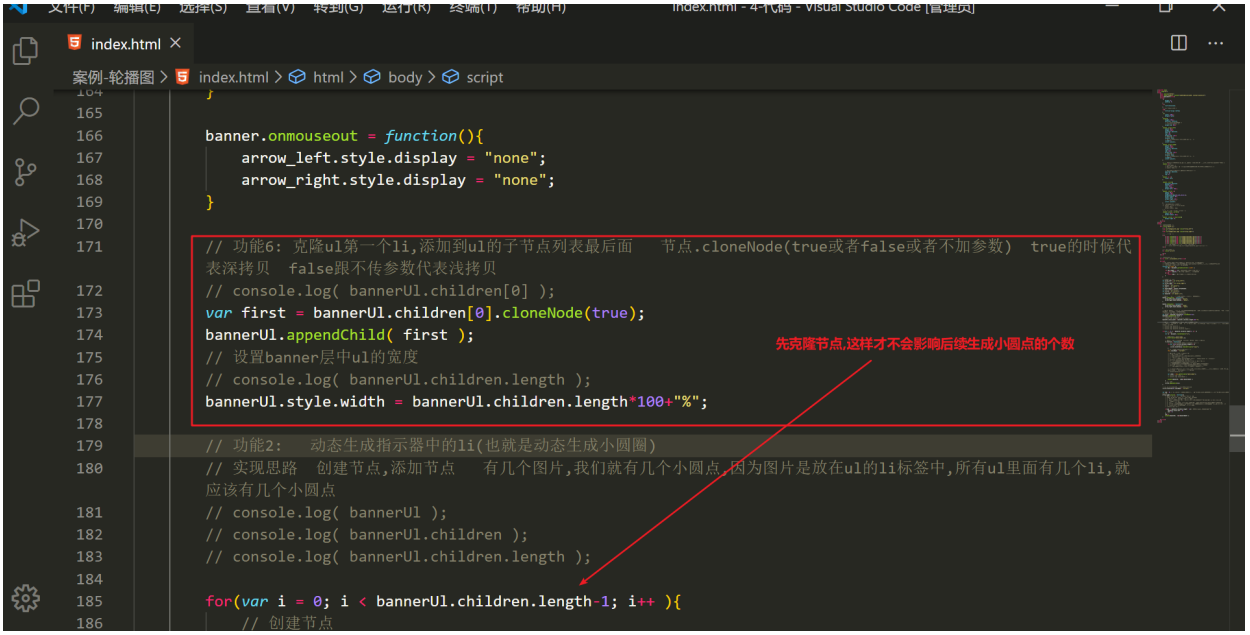
```
var first = ul.children[0].cloneNode(true);
ul.appendChild(first);
// 设置ul的宽度,不然最后一个li无法显示出来
ul.style.width = ul.children.length*100+"%";
```

```
<!-- 轮播图片 -->
<ul>
    <li><a href="#"><img src="images/banner01.jpg"></a></li>
    <li><a href="#"><img src="images/banner02.jpg"></a></li>
    <li><a href="#"><img src="images/banner03.jpg"></a></li>
    <li><a href="#"><img src="images/banner04.jpg"></a></li>
    <li><a href="#"><img src="images/banner05.jpg"></a></li>
    <li><a href="#"><img src="images/banner06.jpg"></a></li>
    <!-- 为了实现无缝滚动,复制第一个li加到ul后面 -->
    <!-- <li><a href="#"><img src="images/banner01.jpg"></a></li> -->
</ul>
```

注释,我们使用js添加第一个节点

```
/* 为什么要给ul设置600%的宽度,因为我们需要把所有的li浮动成一行,ul必须有足够的宽度才可以放得下 */
.banner ul {
    /* width:600%; */
    /* 因为手动复制了添加了一个li到ul的子元素列表后面,这个时候ul里面就有7个li */
    /* width: 700%; */

    /* 如果想要让ul可以移动,需要给ul设置绝对定位 */
    position: absolute;
    left: 0;
    top: 0;
}
```

也注释了,使用js动态设置ul的宽度



功能7: 点击右侧按钮,小圆圈跟随变化

1. 点击右侧按钮,小圆圈跟随变化
2. 最简单的做法就是再声明一个变量currentCircle,每次点击自增1,
注意,后期左侧按钮 也需要用这个变量,所以要声明全局变量

3. 但是li一共有7个(本来6个,加上我们克隆了一个),
我们的小圆圈只有6个,所以必须加一个判断条件

4. 如果这个变量的值等于 ol.children.length 说明走到最后我们克隆的这张图片了
我们就复原为0

5.核心代码如下:

```
// 全局变量
var currentCircle = 0;// currentCircle记录当前是第几个小圆点高亮,0代表第1个

// 以下代码写在arrow_right点击的事件函数中
currentCircle++;

if( currentCircle == circle.children.length ){
    currentCircle = 0;
}

for(var j=0;j<circle.children.length;j++){
    circle.children[j].removeAttribute("class");
}
```

```
25  circle.children[currentCircle].className = "current";
```

代码:

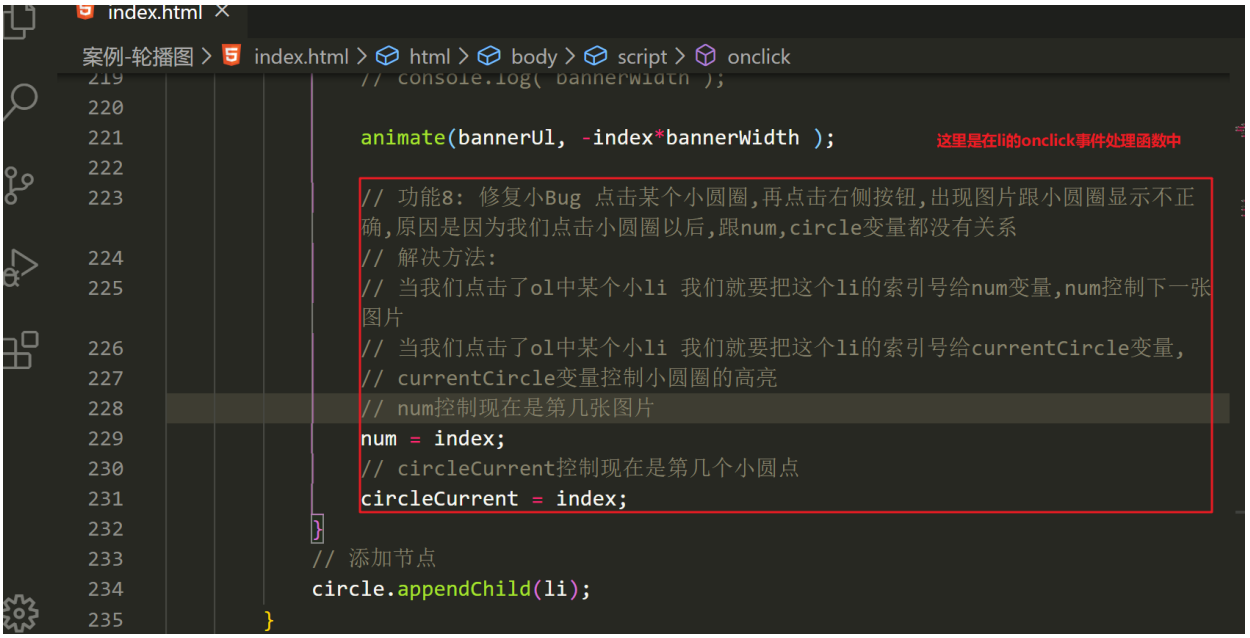
```
var num = 0; // 我们会定义一个变量保存现在是第几张图片,如果num=0,就代表现在是第1个图片;如果num=1,就代表现在是第2个图片
var circleCurrent = 0;// 我们定义一个变量保存现在是第一个小圆点,如果是circleCurrent=0,就代表现在是第1个小圆点
arrow_right.onclick = function(){
    // 功能5: 点击右侧按钮一次，就让图片滚动一张
    // 核心原理:就是点一次右侧按钮,就让ul负一次1200px
    // 1. 点击右侧按钮一次，就让图片滚动一张。
    // 2. 声明一个变量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 克隆一份，放到ul 的最后面， 需要注意修改ul的宽度以及生成小圆点的个数
    // 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：也就是把 left 为0
    // 6. 同时num 赋值为0，可以重新开始滚动图片了
    // console.log( num );


    if(num == bannerUl.children.length - 1){// 代表移动到我们克隆的那张图片
        bannerUl.style.left = 0;
        num = 0;
    }
    num++;
    animate(bannerUl, -num*bannerWidth );

    // 功能7: 点击右侧按钮,小圆圈跟随变化
    circleCurrent++;
    // console.log( circleCurrent );

    // 排他思想
    // circle.children就是代表circle层下的所有li标签
    for(var k=0;k<circle.children.length;k++){
        // 清除所有li的class属性值
        circle.children[k].removeAttribute("class");
    }


    if( circleCurrent == circle.children.length){
        circleCurrent = 0;
    }
    // 单独设置点击的那个li的class属性值
    circle.children[circleCurrent].className = "current";
}
```

这里是右侧按钮事件处理函数里面的代码

功能8: 修复小Bug 点击某个小圆圈,再点击右侧按钮,出现图片跟小圆圈显示不正确,原因是因为我们点击小圆圈以后,跟num,circle变量都没有关系

解决方法:

```
1  当我们点击了ol中某个小li 我们就要把这个li的索引号给num变量,num控制下一张图片
2  当我们点击了ol中某个小li 我们就要把这个li的索引号给currentCircle变量,
3  currentCircle变量控制小圆圈的高亮
```



功能9: 实现左侧按钮功能

1. 把右侧按钮功能代码复制一份
2. 修改相关代码
3. 注意边界值的处理 当我们现在是第一张图片,我们点击了左侧按钮,应该马上不做动画最后一张
4. 核心代码如下:

```
arrow_left.onclick = function(){
    if(num == 0 ){
        num = bannerUl.children.length-1;
        bannerUl.style.left = -num*bannerWidth+"px"
    }

     num--;
     animate( bannerUl , -num*bannerWidth );

     circleCurrent--;
     if( circleCurrent < 0 ){
         circleCurrent= circle.children.length - 1;
     }

     for(var j=0;j<circle.children.length;j++){
         circle.children[j].removeAttribute("class");
     }
     circle.children[circleCurrent].className = "current";
}
```

```
// 功能9：实现左侧按钮功能
arrow_left.onclick = function(){
    // 边界值的判断非常麻烦

    if(num == 0){
        // 当我们现在是第一张图片,我们点击了左侧按钮,应该马上不做动画最后一张
        // 并且修改num的值   因为num就是代表现在图片是第几张
        num = bannerUl.children.length-1;
        bannerUl.style.left = -num*bannerWidth + "px";
    }


    num--;
    animate(bannerUl, -num*bannerWidth );


    circleCurrent--;
    if( circleCurrent < 0 ){// 判断小圆点小于0的时候,我们就让小圆点的值去到最后一个
        circleCurrent = circle.children.length - 1;
    }


    for(var k=0;k<circle.children.length;k++){
        circle.children[k].removeAttribute("class");
    }


    circle.children[circleCurrent].className = "current";
}
```

今日总结

不用做xmind

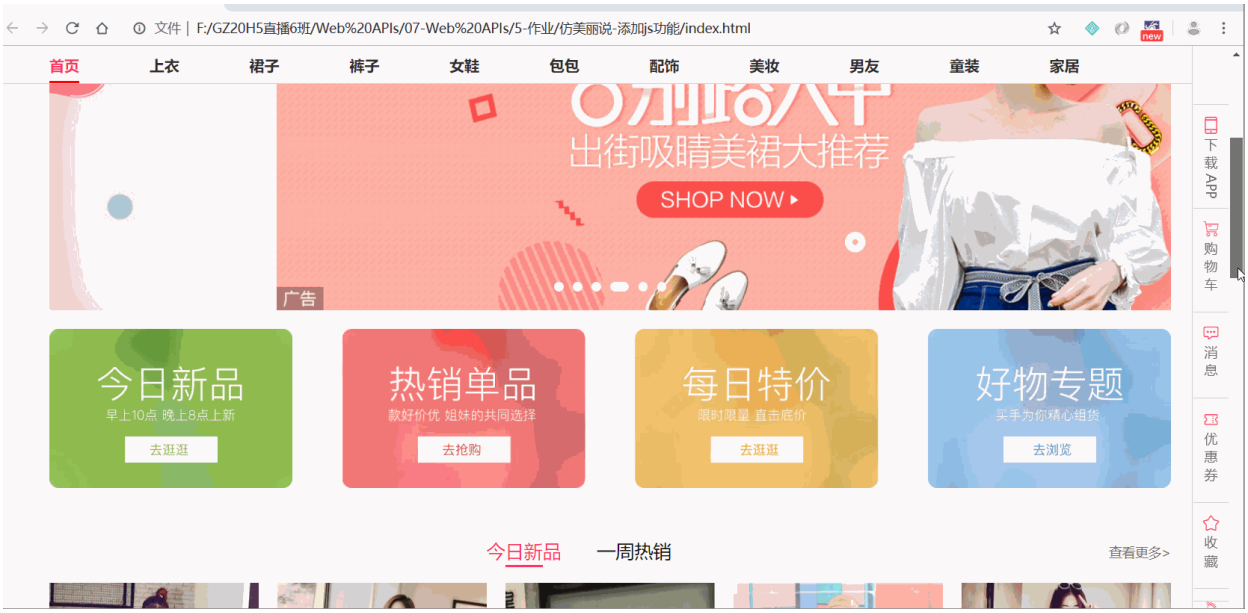
今日作业

完成美丽网页中的js功能

功能1：宝贝、店铺点击可以相互切换

功能2: 选项卡

功能3:导航栏吸顶效果



功能4: 动画回到顶部

功能5: 轮播图(可以不做自动播放的功能,但是其他功能要完成)