任务1 整屏滚动的实现（8 小时）

1.1 预期效果

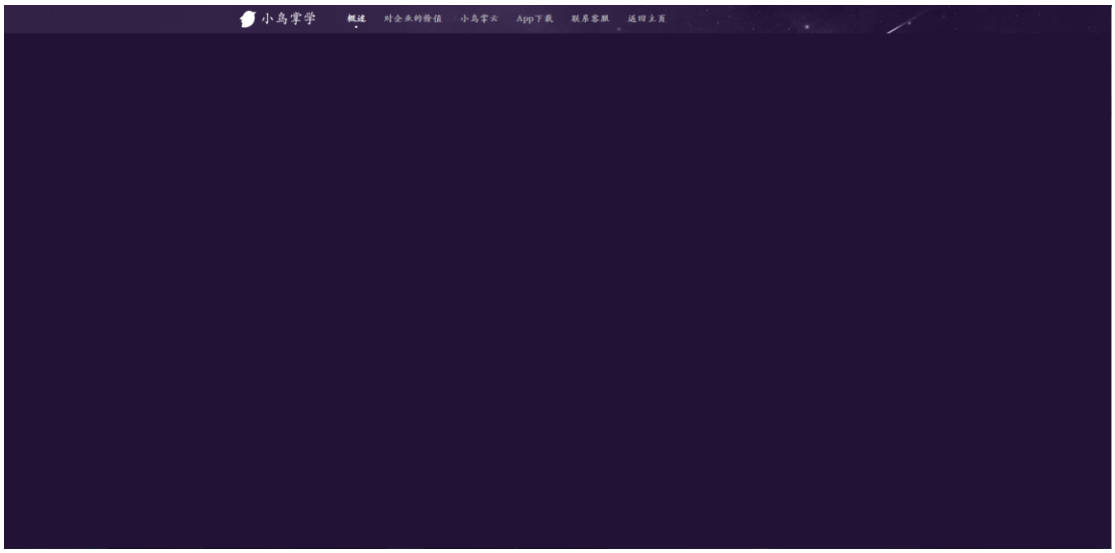

图 1-1、整屏效果图

1.2 掌握技能

表 1-1 任务说明

任务名称	子任务	任务内容
整屏滚动的实现	1	整屏滚动 js 编写

1.3 任务分解

1. 整屏滚动 js 编写

（1）在上一个任务书中写的 js 滚动检测的位置写执行向下或向上滚动的方法（具体方

法在以后步骤再编写），修改后的代码如下。

```
if (e.wheelDelta) {  //判断浏览器 IE，谷歌滑轮事件
    if (e.wheelDelta > 0) { //当滑轮向上滚动时
        //alert("滑轮向上滚动");
        mainSlideUp();
    }
    if (e.wheelDelta < 0) { //当滑轮向下滚动时
        //alert("滑轮向下滚动 ie chrom");
        mainSlideDown();

    }
  } else if (e.detail) {  //Firefox 滑轮事件
    if (e.detail> 0) { //当滑轮向下滚动时
        //alert("滑轮向下滚动")
        mainSlideDown();

    }
    if (e.detail< 0) { //当滑轮向上滚动时
        //alert("滑轮向上滚动 ff");
        mainSlideUp();
    }
 }
```

在 js 滚动检测的对应位置写执行向下或向上滚动的方法，向上滚动方法为 mainSlideUp();向下滚动方法为 mainSlideDown();

（2）在上一个任务书代码后面，接着编写滚动方法代码，先声明所需变量

```
var mainSlideIndex = 0;
var mainSlideGoing = false;
var mainSlideDelay = 0;
var mainSlideTimer =  null;;
```

先申明以后会用到的变量。

mainSlideIndex 为需要滚动到第几个模块的 index 值。

mainSlideGoing 表示滚动是否真正就行。

mainSlideDelay 表示滚动延迟，用于检测是第一次滚动还是第二次滚动（我们这里实现的整屏滚动是要滚动两次滚轮才滚动，所以这里设置这个变量用于判断）

mainSlideTimer 用于保存滚动触发的定时器。

(3) 向下滚动方法 mainSlideDown()的编写

```
//向下滚动
function mainSlideDown(){
//if 判断用于检测第一次鼠标滚动，让第二次鼠标滚动的时候，再执行页面动效
    if(mainSlideDelay < 1){
        clearInterval(mainSlideTimer);
        mainSlideTimer = setTimeout(function(){
            mainSlideDelay++;
        },100)

    }else if(!mainSlideGoing){
        mainSlideGoing = true;
        mainSlideIndex++;
        if(mainSlideIndex>$(".wrap_block").length-2){
            mainSlideIndex = $(".wrap_block").length-2;
        }
        mainSlideGo();

    }
}
```

if 中的判断条件为是否第一次滚动鼠标，如果第一次，就让 mainSlideDelay++。这里设置 setTimeout 的意义为：鼠标滚动事件是连续触发的，而我们只需要检测一次，这里每次触发的时候，我么你设置一个 100 毫秒的 setTimeout，下次触发的时候再清除掉，（一半滚动连续触发间距都比 100 毫秒要小，所以，这样设置的结果就是，只执行一次 mainSlideDelay++）

else if 里执行第二次滚动滚轮的代码,但需要判断没有滚动再执行才执行,所以判断 else if(!mainSlideGoing)，这时候就需要执行滚动啦，执行滚动之前，我们需要先设置：mainSlideGoing = true，设置相应的 index 值 mainSlideIndex，以让滚动方法，可以用（这里是向下滚动，就让 mainSlideIndex++,但是如果已经滚动到最后一个了，那就停留在最后一个），然后执行公用的滚动的运动方法 mainSlideGo()。此方法我们你下一步在写。

（4）公用的滚动运动方法 mainSlideGo()的编写

```
//滚动方法
function mainSlideGo(){
    $(".main_slide").animate({"top":"-"+
$(".wrap_block").height()*mainSlideIndex
+"px"},600,"easeBothStrong",function(){
        mainSlideGoing = false;
        mainSlideDelay = 0;
        if(mainSlideIndex == 0){

        }else if(mainSlideIndex == 4){

            $(".nav_piece").removeClass("now").eq(mainSlideIndex-1).addC
            lass("now");
            $(".nav_piece").eq(mainSlideIndex).addClass("now");
        }else{

           $(".nav_piece").removeClass("now").eq(mainSlideIndex-1).addCl
           ass("now");
        }
    });
}
```

运动方法就是根据上一步设置的 mainSlideIndex 设置 animate 的 top 值即可，等运动完成，再设置：

mainSlideGoing = false;

mainSlideDelay = 0;

给顶部导航添加相应的当前项：

If(mainSlideIndex == 0)不设置

else if(mainSlideIndex == 4)，最后一页包含 app 下载和联系我们，所以顶部需要对两个位置都设置当前项。

else 里面给对应 index 加当前 class 即可。

（5）向上滚动方法 mainSlideUp()的编写

```
//向上滚动
function mainSlideUp(){
    if(mainSlideDelay < 1){
        clearInterval(mainSlideTimer);
        mainSlideTimer = setTimeout(function(){
            mainSlideDelay++;
        },100)

    }else if(!mainSlideGoing){
        mainSlideGoing = true;
        mainSlideIndex--;
        if(mainSlideIndex<0){
            mainSlideIndex=0;
        }
        mainSlideGo();
    }
}
```

判断逻辑和 mainSlideDown()基本一致，这里就不详细说明了。

同样是设置 mainSlideIndex 并执行运动方法 mainSlideGo();

（6）点击导航的时候滚动到相应的模块

```
//点击导航的时候，滚动到对应模块
$(".nav_piece h1").click(function(){
    var navIndex = $(this).parent().index(".nav_piece");
    if(navIndex == 4){
        navIndex = 3;
    };
    if(navIndex != 5){
        mainSlideIndex = navIndex+1;
        mainSlideGo();
    }
```

```
});
```

当导航的每个标签点击的时候：

1、获取对于的 index，

2、判断 index 是否是第四个，如果是第四个，就赋值为 3，因为“app 下载”和“联系客服”在同一页。

3、判断 index 不等于 5，这里的第 5 个是返回首页，另外独立设置跳转链接。

其余情况，执行跳转：先给 mainSlideIndex 赋值，然后执行滚动方法 mainSlideGo();

（7）调整页面大小的时候，设置当前整屏模块调整居中

```
//调整页面大小的时候让整屏居中。
if(mainSlideIndex){
    if(GLOBLE.resizeTimer){
        clearInterval(GLOBLE.resizeTimer);
    }
    GLOBLE.resizeTimer = setTimeout(function(){
        mainSlideGoing = true;
        mainSlideGo();
    },200)
}
```

1、先判断是否设置过 mainSlideIndex，且不为 0；因为刚进入页面时，没执行滚动之前，是不需要调整滚动距离以实现居中的，这时作此种判断就能提高页面性能。

2、因为 resize 之间是连续触发的，我们需要只执行一次调整就可以，所以这里设置 setTimeout 定时器，以实现只触发一次。

3、调整的实惠，先，设置 mainSlideGoing = true;然后执行屏幕滚动方法 mainSlideGo()即可。

（8）设置 hash 值的跳转方式（目的：1、方便后续代码调试。2、以后导航设置链接跳转到此页面时，需要根据设置的 hash 值自动滚动到相应模块），代码编写位置：在整屏滚动代码的后面接着写 hash 值检测的代码。（此代码测试的时候，需要给 url 添加 hash 值，不然代码检测不到，如：原来的 url：guanwang/aboutxiaoniao.html，

现在修改为 guanwang/aboutxiaoniao.html#1 点击 enter，按 f5 刷新页面，就能看到自动跳转效果了）

```
var mainHash = window.location.hash.substring(1);

if(mainHash){
    if(mainHash == 0 || mainHash == 1 || mainHash == 2 || mainHash == 3||
mainHash == 4){
        $(".welcome_wrap").slideUp(0,function(){
            GLOBLE.welcomeOver = true; //用于鼠标上下滑动整屏滚动出发的判断条
件
        });
        mainSlideIndex = mainHash;
        mainSlideGo();
        gaishuMove();
        window.location.hash = "";
    }
}
```

1、获取 url 的 hash 值，并赋值给 mainHash。

2、如果存在 mainHash，则执行内部代码

3、判断 mainHash 是否为我们对于页面的对应值。0、1、2、3、4 分别代表页面中.wrap_block 的每一个块儿。

4、如果有对应的 hash 值，就执行跳转代码，首先$(".welcome_wrap")执行收起隐藏，然后设置 GLOBLE.welcomeOver = true;

设置 mainSlideIndex = mainHash;并执行运动方法:

mainSlideGo();gaishuMove();

最后再将 hash 清除 window.location.hash = "";

提示：以后在写后续模块的时候，只需要在 url 后面添加对应模块的 hash 值，就能直接跳转到对于模块查看效果了，不再需要等待欢迎动画执行完。如：我们写到概述模块的时候，就在 url 后面加上 hash 值“1”（ guanwang/aboutxiaoniao.html#1），点击 enter 并按 f5 刷新页面，就能直接滚动到概述模块。当然，觉得此方法比较麻烦的同学，可以在启动动画时双击页面，收起启动动画，然后点击导航跳转到对应模块，也可以实现同样效果。

整屏滚动效果编写完成！

1.4 参考资料

无

1.5 扩展练习

教师自行补充