# 作业二

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **课程名称** | **Web技术** | **班级** | **计科2001** | **学号** | **8204200213** | | **姓名** | **韩非江** |
| **名称** | **作业2** CSU前端 | | | **时间** | | 2022.11.13前 | | |
| **环境** | | | | | | | | |
| Windows | | | | | | | | |
| **目的和内容要求** | | | | | | | | |
| 请认真阅读并分析学校新版主页（[http://www.csu.edu.cn/](http://www.csu.edu.cn/#/_blank)）   * 复制一个同样的页面（主页），可以复制素材，不能复制代码； * 可以简化，但是不能抄袭； * 独立完成； * 请按作业二的模板文档撰写（有截图、过程和说明、总结）。提交名为“作业三-姓名.doc”，和压缩包（html文件+css文件+js文件，写好注释）。   网站主界面功能的**基本要求**：   1. 自动循环轮播图片(display:block) 或者能够定时切换； 2. 鼠标移动显示下拉菜单。   可选项，不仅限于：   1. 悬浮导航栏，非静态界面的header； 2. 右侧菜单栏的固定 及动态效果； 3. 鼠标移到原图上的渐变（transition:0.5s）、放大(transform中的scale)，以及鼠标切换图片等效果； 4. 移动鼠标显示弹出效果； 5. 返回顶部等等。   解释：   * 图片可以循环自动轮播，当鼠标放在banner上时，自动轮播暂时停止，当鼠标移出，自动轮播恢复；当鼠标点击某个按钮时自动显示其对应的图片，轮播从被点击的button对应的图片开始循环。设置定时器可以实现图片的循环自动轮播。   注意及思考：   1. 源码请用压缩包上传，作业文档勿放压缩包里面。 2. 先布局，设计整体及分块的div框架； 3. JS的位置哪合适？浏览器的版本是否合适，比如document.getElementsByClassName()有用吗？窗口用%定义区域大小（左右用百分比 ，上下用px）合适？console.log()检验函数好用吗？ 4. 前端HTML+CSS+JS三者的关联？ 5. 现在请不用jQuery实现。 | | | | | | | | |
| **实验过程、结果分析与总结** | | | | | | | | |
| **1、程序运行过程和结果（请另外提供所完成的html/css等文件（压缩包），在此贴上运行结果的界面截图）：**  **1.1 网页截图**              **1.2 关键代码解析/思路说明**  **下面展开不以网站板块顺序，而是按照技术路线的顺序说明，目的是找出多个不同网站模块之间的相似性，以及总结出最一般性的设计思路和交互思想，同时也便于期末时打印出来方便查看，可谓是“一举三得”。展开顺序为“布局解析”、“JS交互”、“CSS设计”，其中“布局解析”介绍官网布局的抽象思维概括以及官网水平布局、网格布局方式的总结，涉及到全局布局、导航栏布局以及一些复杂布局；“JS交互”展示定时器的使用以及元素选择过程中的问题（选对象还是数组，选第一个还是全部），涉及到下拉框和两种轮播图的实现。“CSS设计”展示鼠标交互事件、链接样式、图片引入方式等，涉及到侧边栏的设计和全局链接的设计，还有居中布局的设计。** 布局解析 **本网站共涉及4种主要布局模式：div+flex布局、div+grid布局、div/a/span+inline-box和div+float。其中flex布局用途最广，占比高达90%以上。 其实官网很多模块使用的是table和ul、dl来进行布局，这样是有一定优点的，比如说ul布局的语义性高，在css未加载成功是也可以表名对应的语义。不过，我的看法是，只用div+flex/grid/float，先把div万能布局练熟、找到布局套路后，再尝试其他复杂布局。**  **下面的展开是先总结学到的布局抽象思想，再总结其中用到的水平布局（因为竖直布局太简单了就不说了）。**  **拿到官网，首先看大盒子，再看小盒子。**大盒子一共有6个部分，要注意侧边栏和返回按钮的部分。**然后针对每个小盒子，先看里面的水平布局和竖直布局是怎么组合的**，注意不要直接关注元素层面，而是先看布局组合（不要因小失大）。**如果确定是竖直布局**，那就直接div即可，默认就是竖直的；当然对于span和a这类标签可以通过block或者直接display=‘’变为竖直排布。**如果是水平布局**，对于span和a默认水平排布，只需要关注颜色和间距即可；对于div需要使用flex布局、float、inline-box。**最后如果是网格布局**，可以使用grid布局和table布局，其中grid布局是我经常使用的。    **先说第一处布局：使用justify-content: space-between实现横向居中排布**  **举例案例：网站导航栏布局。**    **分析布局和元素：**   1. **第一排小字，左右布局，右侧有两种类型** 2. **第二排大字，左右布局，中间有图片**   **因此分别建立div嵌套，然后分别使用flex即可。**  **代码框架如下图：**    展开代码如下：    关键是flex布局，因为直接display:flex的结果就是元素挤在一起；而使用：  justify-content: space-between;  会导致元素直接贴在两边，因此要加上左右margin：  margin-left: 10%;  margin-right: 10%;  **总结1.1：第一种横向居中布局，使用space-between+margin。**  **接下来是第二种横向布局：**  **案例：图片滚动下方的选择按钮**    **分析布局和元素**：四个白色条带水平排布，对应图片的地方为橘色。  代码展示：    具体实现就是分别用div构建矩形，设置背景为白色，并且使用inline-block实现水平布局。    当然，使用前面flex的思路依旧可以实现，不过需要注意的是需要用justify-content=center使元素居中，且还要加元素间的margin把原始紧凑的元素分开，比inline-block复杂很多，代码如下：    **总结1.2：使用inline-block可以快捷实现div水平排布；使用flex布局时需要注意居中和元素间距问题。**  **再来看一种比较综合的复杂的水平布局：**  **案例：讲座和公告栏**    **分析布局和元素：**   1. **整体是水平布局，因此最外层有div嵌套+flex。** 2. **对于一个消息体而言内部也是水平布局的，左侧是蓝色方块，右侧是文字描述。** 3. **方块内部的文字为数值排布，可以使用span+block，也可以直接div。**   代码实现（局部）：      HTML思路如上，这里不再赘述。突然感慨我所有HTML的嵌套设计都是为了后面使用flex布局才这样做的，如果用table的话会简化过程，不过不利于系统学习div+flex布局。  关键是CSS中多个flex的使用。  不加flex时是这样的，全部都是竖直布局：    加了一个内部flex后，方块间是水平布局：    再加外层flex后，整体水平布局：    两个flex的代码如下：  其涉及到的前面强调的两个重点：1.flex配合center； 2.flex配合margin    其中日期块字体的样式如下，为了实现span的竖直排布使用了display：block：  <span><em>02</em> 2022-11 </span>    右侧块中有三个图标水平排布，使用i+inline-block实现：  <span><i style="background-image:"></i></span>  <span><i style="background-image:"></i>2021年11月5日报到、6-7日全天会议</span>  <span><i style="background-image:"></i>长沙时代华瑞大酒店</span>    **总结1.3：多个flex嵌套使用实现复杂布局；使用<span><i>的嵌套结构+inline-block实现文字前面显示图标的水平布局。**  **最后来看一个布局，实现叠加效果：**  **案例：轮播图下方文字描述跟着动**。    **布局分析：**   1. 图片和下方蓝色方块竖直排布。图片使用img或div+背景图实现，div设置背景+调整margin或left。 2. 切换按钮水平布局。使用div+背景图实现。   <div class="news-scroll-item">       <img class="news-scroll-item-img" src="https://www.csu.edu.cn/xxx.png">       <div class="img-txt">        <h3 style="text-overflow:clip;">湖南省人民政府副省长陈飞来校调研附属医院</h3>        <div>2022-11-25</div>      </div>  </div>  <div class="news-buttons">       <div class="news-button" onclick="newsScrollclick(0)" style="background: url();"></div>    <div class="news-button" onclick="newsScrollclick(1)" style="background: url();"></div>  <div class="news-button" onclick="newsScrollclick(2)" style="background: url();"></div>    <div class="news-button" onclick="newsScrollclick(3)" style="background: url();"></div>  </div>  对应CSS如下：  /\* 限制显示 \*/  .news-scroll-imgs {      max-width: 590px;      overflow: hidden;  }  /\* 总长度 \*/  .news-imgs {      width: 2360px;      display: flex;      margin-top: 10px;      transition: all .3s ease-in-out;  }  /\* 图片大小保持一致 \*/  .news-scroll-item img{      width: 590px;      height: 380px;  }  下方的文字块在未经css修饰时是直接竖直排布的：    因此一种比较简单的思路就是使用transform移动：  /\* 新闻图片底部图片和文字 \*/  .img-txt {      background: url(https://www.csu.edu.cn/images/bg-q3.jpg) center center no-repeat;      background-size: cover;      padding: 15px 35px 11px 35px;      overflow: hidden;      transform: translate(40px,-20px);      width: 480px;  }  但是因为滚动的原因，文字块也要和图片一样拷贝多份，因此需要合理设置width大小，不然会出现下一个块提前出现的问题，上面的480px是我调出来的：    同时，注意下方按钮也要使用transform一起排版：  /\* 按钮整体位置调整 \*/  .news-buttons {      transform: translate(475px,-55px);  }  **总结1.4：竖直布局+transform可以便捷且暴力的实现叠加效果。**  **好了，水平布局和竖直布局相关的内容大体已经总结完成，下面做一个索引类型的最后总结：**   1. **横向居中布局，使用space-between+margin。** 2. **使用inline-block可以快捷实现div水平排布；** 3. **使用flex布局时需要注意居中和元素间距问题。** 4. **多个flex嵌套使用实现复杂布局；** 5. **使用<span><i>的嵌套结构+inline-block实现文字前面显示图标的水平布局。** 6. **竖直布局+transform可以便捷且暴力的实现叠加效果。**   **下面来介绍网格布局，即grid布局，作为table布局的div替代版本：**  **如下图这种排列规整、行列清晰的就可以使用grid布局**：    首先要有外层div+内层div写出来，这一步很简单，因此省略。  关键是grid的css：  /\* 新闻入口 \*/  .news-details {      border-top: 3px solid rgb(64, 94, 193);      height: 400px;      width:700px;      display: grid;      grid-template-columns: 350px 350px;      background: #fff;      align-items: center; /\*文字y轴居中\*/  }  使用grid-template-columns指定每一列的宽度。使用align-items:center使文字竖直居中。  下面的布局看似很像网格，但是因为最后一行有三个元素，因此实际上不能简单的用网格布局实现，因此可以再考虑用div+flex实现。      对于外存div，使用flex+wrap换行+flex-start对其。  内部a标签的宽度设置为43%，进而实现一行两列的效果。  .r2-top-details {      display: flex;      flex-wrap: wrap;      justify-content: flex-start;      border-top: 4px solid rgb(232, 184, 8);      box-shadow: 0 0 10px rgba(0,0,0,.2);      background-color: white;      position: absolute;      transform : translate(-300px, 0px);      color: black;      overflow: hidden;      height: 0;  }  .r2-top-details a{      display: block;      width: 43%;      font-size: 14px;      color: #333;      line-height: 25px;      padding-right: 5px;      overflow: hidden;      transition: .3s;  }  对于其中特殊的一行，直接使用flex实现即可：  .r2-top-details div{      display: flex;      justify-content: flex-start;  }  **总结1.5：网格状布局直接使用grid布局；类网格布局使用外部flex+wrap+内部宽度45-50%来实现。**  布局介绍完毕，包括在作此网站时布局的概述思想，水平布局、竖直布局、网格布局的各种方法。接下来介绍Js交互，主要是针对轮播和下拉栏谈一谈interval的使用。 JS交互的重点 **先介绍用js实现动画播放效果的方法：requestAnimationFrame、setTimeout、setInterval。** requestAnimationFrame 比起 setTimeout、setInterval的优势主要有两点：1、requestAnimationFrame 会把每一帧中的所有DOM操作集中起来，在一次重绘或回流中就完成，并且重绘或回流的时间间隔紧紧跟随浏览器的刷新频率，一般来说，这个频率为每秒60帧。2、在隐藏或不可见的元素中，requestAnimationFrame将不会进行重绘或回流，这当然就意味着更少的的cpu，gpu和内存使用量。 **先说下面第一种轮播图：连续循环轮播。**    **思路：**  **HTML层面：需要设置一个显示窗显示n张图片，超出窗口的都隐藏；需要横向排布2n张图片； 需要把2n张图片放到一个移动容器中，用js改变这个容器的位置即可实现图片滚动。**    JS层面：先设计位置变化函数startnow()。      var imgScroll = document.getElementsByClassName('img-bar')[0];      var mleft = 0;      var mtop = 0;      let timer\_scroll\_imgs = null;      function startnow() {          // 此时立即替换为开头第一组图片，人眼是看不清的          if (mleft <= -1210) {              imgScroll.style.transform='translateX(' + (-mleft) + 'px)';              mleft = 0;          }          imgScroll.style.transform='translate(' + (mleft) + 'px,' + (mtop) + 'px)';          mleft -= 2;      }  调用此函数可实现2n张图片整体左移，如果n张图片已经移动完毕，此时利用人眼视觉间隔立即将第n+1到2n张图片重新替换为0-n张的图片，然后继续移动即可。  下面的关键是如何循环调用这个函数实现动画效果，先试用第一种setInterval()实现：      timer\_scroll\_imgs = setInterval(startnow, 30);      startnow();      imgScroll.onmouseover = function() {          clearTimeout(timer\_scroll\_imgs);      }      imgScroll.onmouseout = function() {          timer\_scroll\_imgs = setInterval(startnow, 50);          startnow();  }  注意setInterval()的调用只是函数外的一条语句，目的是设置一个30ms的定时器。  再用requestAnimationFrame()实现循环调用：  //当前执行时间      var nowTime = 0;      //记录每次动画执行结束的时间      var lastTime = 0;      //我们自己定义的动画时间差值      var diffTime = 2;      //requestAnimationFrame效果      function animloop() {          // 保存动画播放时间信息          timer\_scroll\_imgs = requestAnimationFrame(animloop);          nowTime = timer\_scroll\_imgs;          // 指定间隔执行动作          if(nowTime-lastTime >= diffTime){              lastTime = nowTime              startnow();          }      };      animloop(); // 初始默认调用      // 清除动画      imgScroll.onmouseover = function() {          cancelAnimationFrame(timer\_scroll\_imgs);      }      // 新动画      imgScroll.onmouseout = function() {          nowTime = 0;          //记录每次动画执行结束的时间          lastTime = 0;          //我们自己定义的动画时间差值          diffTime = 2;          animloop();      }  此时requestAnimationFrame是在一个函数中调用的，类似一种递归调用。其返回值是一个和动画运行时间有关的值，可以保留+作差来减缓动画。上面我设置diffTime=2，目的是把原先默认60ms的动画变为120ms，即每60ms执行一次变为120ms执行一次。  总结一下，动画播放共有两种形式可选，大体套路如下：   1. 先写变化函数，里面用if设置动画循环需要的判断，用style改变大小和位置。 2. 对于setInterval，只要在函数外面建立一个计时器即可；对于requestAnimationFrame，需要设置一个loop函数，在内部递归调用。 3. 停止动画使用clearInterval和cancelAnimationFram即可。恢复动画只要重新设置计数器或者重新调用loop函数即可。   **接下来看第二种动画：下拉框。**  类似的思路，其实我们上面已经抽象出动画的最一般设计方式了。下面的实现有一些些不一样，即计时器的生成需要if判断，只有高度小于一定值时生成下拉计时器，大于一定值时生成上拉计时器。   1. 先写两个函数控制下拉和收起。 2. 函数内部用if判断高度，分别生成计时器。 3. 鼠标移动over时：下拉框、清除上拉计时器。 4. 鼠标out时：上拉框、清除下拉计时器。   let r2\_top\_img = document.querySelector('.r2-top-img');      let r2\_top = document.querySelector('.r2-top');      let r2\_top\_o1 = document.querySelector('.r2-top-o1');      let r2\_top\_o2 = document.querySelector('.r2-top-o2');      let r2\_top\_details = document.querySelectorAll('.r2-top-details');      let timer\_over = null;      let timer\_leave = null;      let h = 0;      function showDetails() {          r2\_top\_details[0].style.display = "";          h += 5;          if (h <= 200) {              r2\_top\_details[0].style.height = h + 'px';              timer\_over = setTimeout(showDetails, 10);          } else {              return;          }        }      function hideDetails() {          h -= 5;          if (h <= 0) {              r2\_top\_details[0].style.display = "none";              return;          } else {              r2\_top\_details[0].style.height = h + 'px';              timer\_leave = setTimeout(hideDetails, 10);          }      }      r2\_top\_o1.onmouseover = function() {          clearTimeout(timer\_leave);          showDetails()      };      r2\_top\_o1.onmouseout = function() {          clearTimeout(timer\_over);          hideDetails();      };  **接下来展示第三种js动画：横向渐变轮播图和横向滑动轮播图。**  其中横向渐变轮播图如下：  设计思路为：   1. 重叠重张图片 2. Js交互改变透明度变换     **横向滑动轮播图如下：**  设计思路为：   1. 多张图片水平排布，超出展示框的图片隐藏 2. Js交互实现位置移动     因为我们可以用一个计时器同时更改两个轮播图的改变，因此我们初始的js函数设置如下：一个函数中同时实现透明度变化和位置变化，对应两种轮播方式。      /\* 图片变换 \*/      /\* banner图片变换 \*/      var index = 0; // 默认是第一张图片      var imageitem = document.getElementsByClassName("image-item");      let banner\_img\_num = imageitem.length;      var scrollbuttons = document.getElementsByClassName("scroll-button");      /\* 新闻图片 \*/      var index2 = 0;      var newsItem = document.getElementsByClassName('news-imgs');  let newsButton = document.getElementsByClassName('news-button');      function scroll() {          // banner图片          for (var i = 0; i < imageitem.length; i++) {              let isCurImg = i == index;              imageitem[i].style.opacity = isCurImg ? 1000 : 0;              scrollbuttons[i].style.background = isCurImg ? '#FF9900' : '#FFFFFF';          }          index = (index + 1) % banner\_img\_num;          // 新闻图片          for (var i = 0; i < newsButton.length; i++) {              newsButton[i].style.background = 'url(https://www.csu.edu.cn/images/img6.png) center center no-repeat';          }          newsButton[index2].style.background = 'url(https://www.csu.edu.cn/images/img7.png) center center no-repeat';          let offset = -590\* index2;          newsItem[0].style.transform = 'translateX(' + offset + 'px)';          index2 = (index2 + 1) % 4;      }  通过改变index的数值，结合document函数获取dom元素，进而有针对性的改变元素style。即只要把当前index对应的dom元素设置为显示状态，其余index对应的dom元素都为隐藏状态即可。通过透明度和overflow:hidden实现。  还是一样的套路，下面直接进行第一次调用和计时器声明即可：  scroll();  setInterval(scroll, 5000);  不同的是这两种轮播图都有下方的按钮，因此还需要scroll函数同步改变index对应按钮的样式。此外，还需要点击按钮后切换到对应index的图片：  function scrollclick(e) {          index = e;          for (var i = 0; i < imageitem.length; i++) {              let isCurImg = i == index;              imageitem[i].style.opacity = isCurImg ? 1 : 0;              scrollbuttons[i].style.background = isCurImg ? '#FF9900' : '#FFFFFF';          }      }      function newsScrollclick(e) {          index2 = e;          for (var i = 0; i < newsButton.length; i++) {              newsButton[i].style.background = 'url(https://www.csu.edu.cn/images/img6.png) center center no-repeat';          }          newsButton[index2].style.background = 'url(https://www.csu.edu.cn/images/img7.png) center center no-repeat';          newsItem[0].style.transform = 'translateX(' + -590\* index2 + 'px)';          index2++;          index2 = index2 % 4;      }  我们只需要在HTML中对应的dom元素处加上交互即可：      **轮播图的介绍完毕，下面进行一次总结：**  A.轮播图有三类，分别为 1.持续滚动轮播图； 2.渐隐渐入轮播图； 3.滑动轮播图。  B.其实现思路有两种：1.图片重叠+透明度变化； 2.图片水平排布+位置移动  C.有按钮的交互：设置index，按钮和图片对应。Js改变index实现图片切换。移动时，可使用transform移动index\*w的位置来实现。  D.实现渐变效果：使用transition。  transition: all .5s ease-in-out;    **Js动画设计的套路为：**   1. 设计包含边界if判断的style修改的变换函数 2. 计时器在变换函数内部 3. 计时器在变换函数外部 4. 初始化调用变换函数 5. 交互函数中清除计时器、生成新的计时器   主要的JS设计内容已经介绍了，下面再总结一些实验过程中dom元素选择的注意点：  document.getElementsByClassName(名字)  获得数组，需要用index获取孩子。  2.  document.getElementById(名字)  直接获得dom元素。  3.  document.querySelector(.名字)  直接获得第一个dom元素  4.  document.querySelectorAll(.名字)  获得数组。   1. 对选择的元素添加交互的方式：   r2\_top\_o1.onmouseover = function() {          Xxxxxx      };   1. 交互可对内部所有孩子生效。  CSS设计重点 **Css代码量较大，第一个模块已经展示了很多了，这里我按照官网设计过程中display的使用、position的使用、CSS调整位置的方式、居中的方式、a标签样式、hover样式、动画渐变的顺序来叙述。**    **首先对于display来说，其目的是：**   1. 使用flex和grid布局 2. 使用block、inline-block、inline改变布局   在官网的设计中，用的最多的就是display来进行辅助布局了，具体的布局过程已经在第一个模块中进行叙述，此处不再说明。      **对于position来说，其目的是辅助设置位置。**   1. 如果是relative，则相对于body定位。 2. 如果是absolute，则相对于父亲relative定位，如果父亲没有声明relative，则相对于body定位。   一般而言对最外层设置为relative，内部设计为absolute。    **Css调整位置的方式主要是使用margin、padding、left top right bottom、text-align、justify-content、transform-translate。**  准确来说，(left right top bottom)与margin、padding是没有可比性的。因为，前者用于定位，后者属于盒子模型。但是，（left right top bottom)有时可以实现与margin类似的效果。    设置(left right top bottom)时，需设置对应元素的position，不同的position，表现不同。若为relative，相对于元素在文档流中的正常位置偏移；若为absolute，相对于元素的包含块边界偏移；若未设置positon或设置为static，则不生效。  总结来说就是若需要对元素设置自身的内外边距，则使用margin和padding；若需要移动元素的位置，则使用(left right top bottom)。  **居中的方式有如下方式，具体的布局方式前面已经说明过了：**  **文字居中：**  **方法1：   display: flex;   justify-content: center; /\*文字水平居中\*/   align-items: center; /\*文字y轴居中\*/  方法2：   text-align: center;   line-height: height;**  **图片居中：**  **img {   display: block;   margin-left: auto;   margin-right: auto;   width: 40%;  }**  **div居中：**  **.center {    margin: auto;   width: 50%;   border: 3px solid green;   padding: 20px;  }**    **a标签超链接的样式设计：**  **.ita:link {**  **color: red;**  **text-decoration: none;**  **}**  **.ita:visited {**  **color: green;**  **text-decoration: none;**  **}**  **.ita:hover {**  **color: hotpink;**  **text-decoration: underline;**  **}**  **.ita:active {**  **color: blue;**  **text-decoration: underline;**    **动画渐变方式如下：**  **<style>   div {   width: 100px;   height: 100px;   background: red;   transition: width 2s, height 2s, transform 2s;  }  ​  div:hover {   width: 300px;   height: 300px;   transform: rotate(180deg);  }  </style>**  **div {   transition-property: width;   transition-duration: 2s;   transition-timing-function: linear;   transition-delay: 1s;  }**   1. **在实验过程中遇到的问题与解决方法：**   **遇到的问题：**  **HTML如何实现复杂布局**  **Js怎么实现动画（轮播、下拉）**  **Js怎么给一排按钮添加交互**  **Css的渐变效果设计**  **解决方法：**  **A.自己思考（最推荐，自己的想法才是实验最精华的部分）**  **B.问dhc（最高效，veryyyy推荐）**  **C.上课听同学分析思路（高效，不过需要注意他们的思路不一定完美，要自己思考改进）**  **D.直接借鉴官网源码（推荐，不过官网代码很冗余和乱，需要自行判断）**  **E.搜blog（质量不一，耗时间）**   1. **实验过程中的发现与收获，未解决或需进一步解决的问题：**   **注：具体的实现细节总结已经在前面的板块用红字标出，此处只说明抽象的收获。**  **之前只写过个人简历和日历、猜数字这几个小东西，最多也就是简单竖直横向布局一些就行了，因此我其实对很多复杂布局还不熟、没有系统总结过有哪些常用布局方式，只知道无脑用flex。利用本次的实验报告我系统梳理了各种布局的方式，学到了很多。**  **以前从来没试过Js动画，这次尝试了以后感觉比Css动画好写多了，虽然有一定的性能开销，但是更加灵活。**  **有一些小问题比如，Js获取dom元素的时候获得的是一个obj还是数组；使用absolute还是relative； transform移动的方式......通过本次实验我都进一步学习和掌握了。**  **最后，本次实验我全部用的div布局，感觉现在已经可以用纯div实现很复杂的布局了，不过有时用table和ul、dl是更好的选择，所以我进一步需要解决的问题是灵活使用div以外的元素进行布局，以便优化性能和增加语义信息。** | | | | | | | | |
| **指导老师评阅意见** | | | | | | | | |
|  | | | | | | | | |
| 指导老师： 年 月 日 | | | | | | | | |