目录

[一绪论： 1](#_Toc44439199)

[1.2汽车仪表的发展历程 1](#_Toc44439200)

[1.3研究背景 2](#_Toc44439201)

[1.4研究内容 2](#_Toc44439202)

[二：功能模块 3](#_Toc44439203)

[2.1水温表和油量表的实现 3](#_Toc44439204)

[2.2转速表和速度表的实现 6](#_Toc44439205)

[2.3滑块部分 7](#_Toc44439206)

[三.项目结构 9](#_Toc44439207)

[四．总体效果图 10](#_Toc44439208)

[五：总结 10](#_Toc44439209)

### 一绪论：

**1.1汽车仪表的概述**

汽车仪表盘系统是车载综合信息系统的重要组成部分，也是汽车要实现的最基本的功能。仪表盘是驾驶员和汽车进行信息交流的重要接口和界面，也是汽车高尖技术的重要组成成分，它为驾驶员提供汽车运行状态的信息，方便驾驶员结合实际，更好的进行操作。随着汽车工业的发展，汽车仪表的各种构件日渐丰富，汽车仪表盘上的信息也逐渐增多，成长为现代汽车信息中枢，并随着现代汽车电子技术的发展而快速发展。

### 1.2汽车仪表的发展历程

汽车仪表按照工作原理分，可以分为4类：

第一类：

基于机械作用力而工作的机械式汽车仪表盘，通常称为机械机心态；

第二类：

基于电测原理，通过各种传感器将被测的非电量转换为电信号，加以测量，被称为电气式仪表；

第三类：

模拟电路电子式汽车仪表；

第四类：

步进电机式全数字汽车仪表；

今后随着信息技术的发展和计算机技术的不断发展，以及降低汽车仪表盘制造成本的要求，汽车仪表盘逐渐向网络化，智能化，和虚拟化的方向发展。

### 1.3研究背景

汽车仪表的发展曲实传统的机电式模拟仪器，只能为驾驶员提供汽车运行时必要而又少量的数据信息，不能满足现代汽车的新技术、高速度的要求，汽车仪表功能已不是单单的知识，而是 通过对汽车各个部件参数的监测和微处理机配套，从而控制汽车各种运行状况的目的。

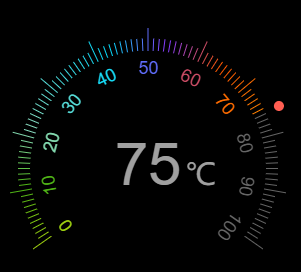
canvas作为html5的新增标签，这个 HTML 元素是为了客户端矢量图形而设计的。它自己没有行为，但却把一个绘图 API 展现给客户端 JavaScript 以使脚本能够把想绘制的东西都绘制到一块画布上。可以在网页显示实时生成图像，并且可以js操作图像内容，为模拟汽车仪表盘提供技术支持。

### 1.4研究内容

为了增强人们对汽车仪表盘的认识，利用html5的新增标签对汽车仪表盘进行模拟，并可以通过设置的滑块对仪表盘的指针和数值进行调整，要求界面简洁，符合大众的审美要求，能满足与用户交互的需求。

### 二：功能模块

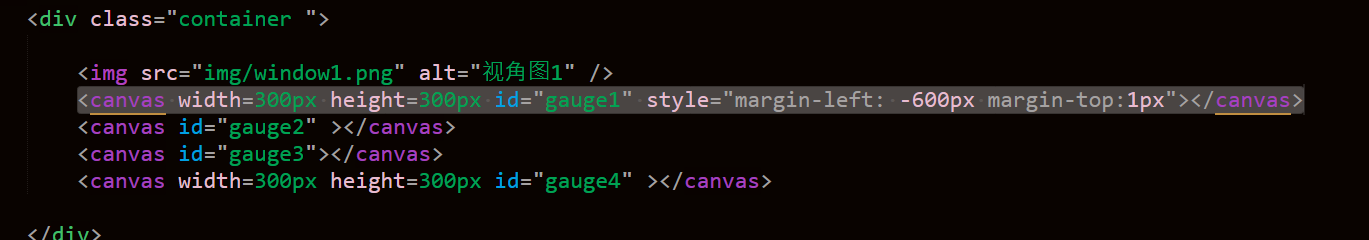
### 2.1水温表和油量表的实现

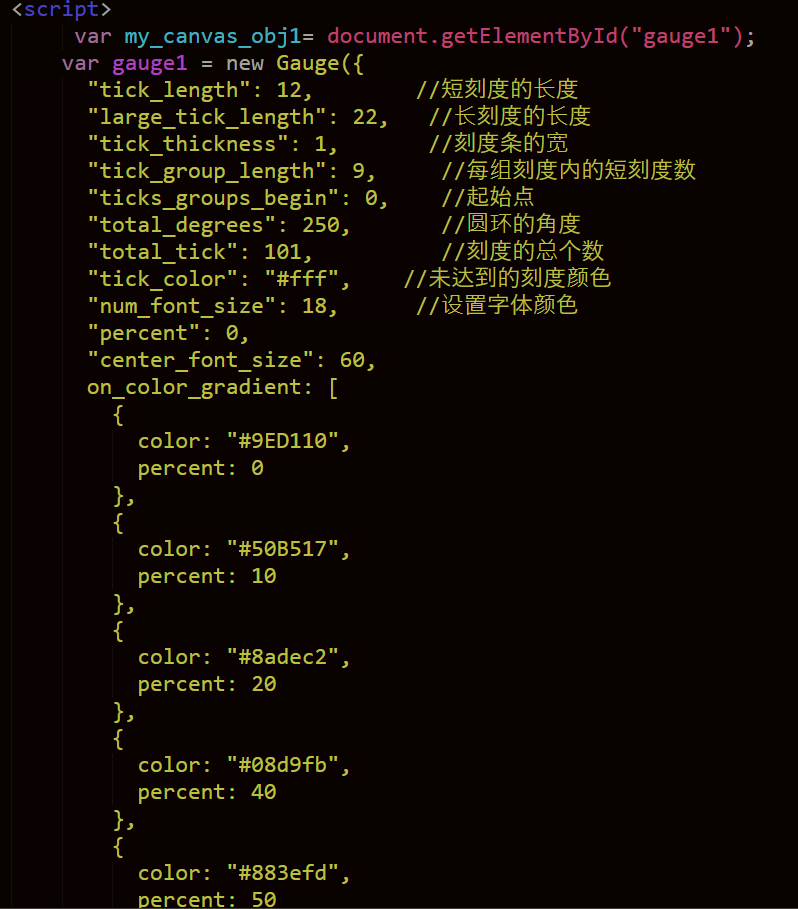


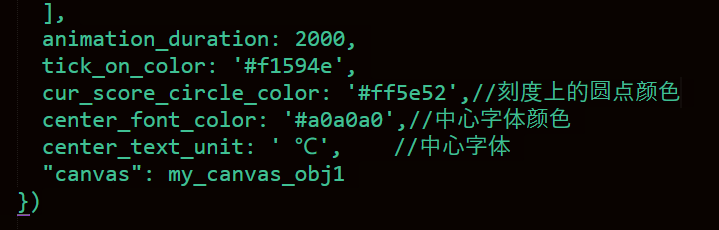
仪表盘中的水温表和油量表的基本原理差不多，只是样式和数值有所改变，这里以水温表为例子。

首先画出一个静态的仪表盘，对仪表盘进行分解。仪表盘可以分为最外边的圆环、长刻度线、短刻度线、圆环周围的数值、中间的数值显示等。

在html加入一个<div>,然后加入canvas元素，设置宽、高。然后配置仪表盘的参数，如下所示

通过id绑定到canvas后，进行仪表盘参数配置。如下

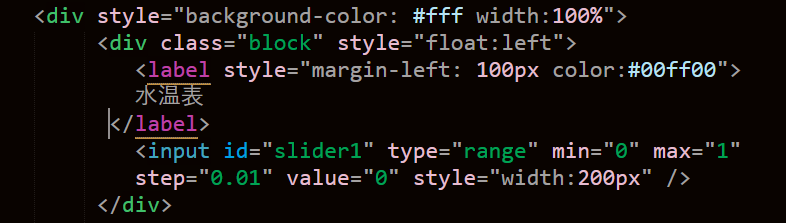




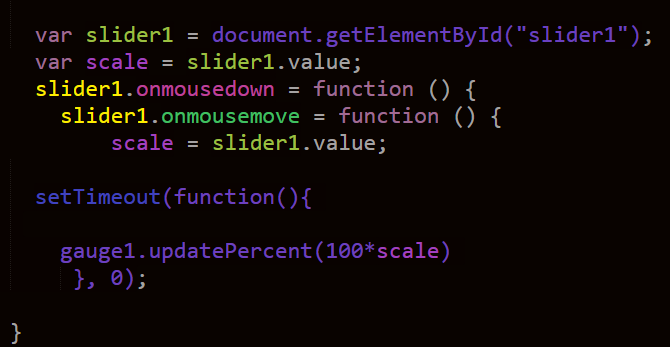
参数配置完成后，调用js中的render()方法绘制初始状态的仪表盘。

在html中插入另一div,在里面插入文本标签和滑块标签。

设置滑块的最大值，最小值、初始值、最小刻度值，长度等参数，同时设置id,便于找到控件。



通过id绑定input控件，获得输入的值，同时监听鼠标动作，当滑块发生滑动时，立马更新canvas,产生一种仪表盘动起来的效果。



实际在js中利用 requestAnimationFrame 函数，在函数回调中执行 render 函数，绘制出不一样的刻度亮/暗图。

 var lastUpdate = +new Date();

      // 动画开始时已经亮起来的刻度的百分比

      var start = this.\_percent;

      // 目标百分比

      var end = this.\_target\_percent;

      // 计算一下每毫秒我们需要完成多少百分比，然后后面每次执行时，根据经过了多少时间来计算出应该完成多少，即刻度应该亮起来多少

      var change\_per\_ms = (end - start)/duration;

      var increasing = change\_per\_ms > 0 ? 1 : 0;

      // 首先计算一下所有刻度亮起来时应该是什么颜色的，保存到一个数组中。

      // 后面动画过程中直接从数组中取值即可,不需要再次计算。

      this.colorArray = this.\_gradientColorArray();

      // 更新逻辑

      var update = function () {

        // 通过时间来计算应该完成的百分比

        var now = +new Date();

        var elapsed = now - lastUpdate;

        \_this.\_percent += elapsed\*change\_per\_ms;

        lastUpdate= now;

        //检测一下是否已经达到我们最终的百分比，如果达到了就停止动画；如果没有就继续绘制

        if ((increasing && \_this.\_percent < \_this.\_target\_percent)

          || (!increasing && \_this.\_percent > \_this.\_target\_percent)) {

          \_this.render();

          \_this.\_requestAnimFrame(update);

        }

        else {

          \_this.\_percent = \_this.\_target\_percent;

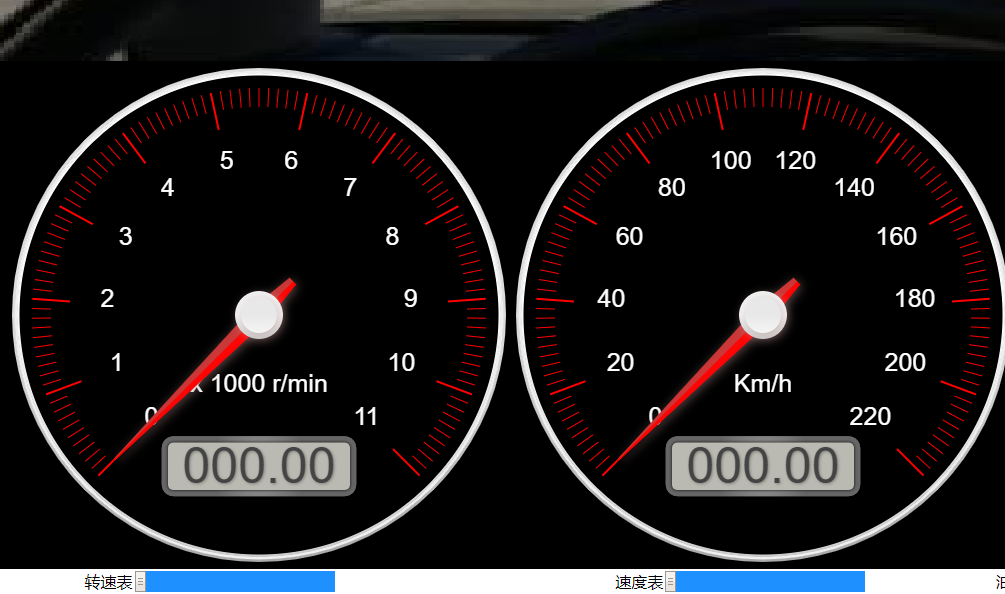
          \_this.render();

        }

      };

      \_this.\_requestAnimFrame(update);

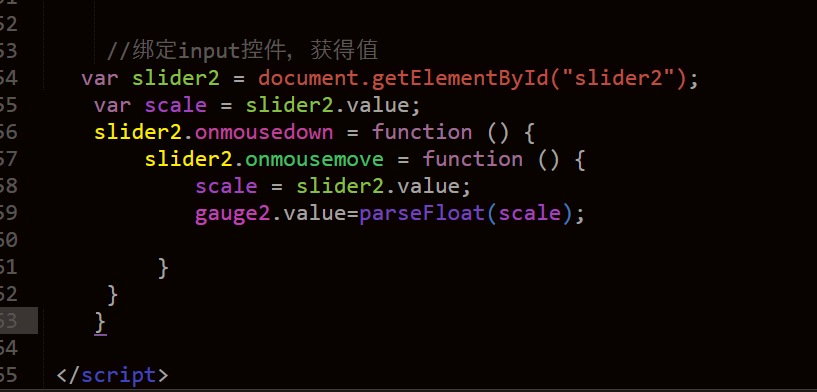
### 2.2转速表和速度表的实现



首先配置仪表盘的基本参数



同时绑定滑块控件，通过滑块来控制仪表盘值得变化，使得仪表盘动起来。



### 2.3滑块部分

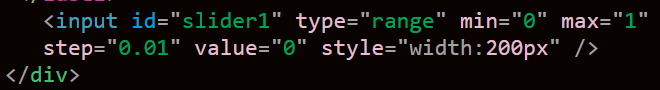
美化前：



美化后效果图如下



滑块通过一下代码实现：



设置好滑块的id,便于绑定控件，进行数据传输，定义好最大值、最小值、默认值、 宽带。

下面对滑块进行美化和各个浏览器之间的适配：

/\*chrome、safari浏览器\*/

input[type='range']::-webkit-slider-runnable-track{

background-color: #1E90FF;

}

/\*firefox浏览器\*/

input[type='range']::-moz-range-track{

background-color: #1E90FF;

}

/\*IE浏览器\*/

input[type="range"]::-ms-track{

/\*color:transparent;去除轨道内竖线\*/

/\*border-color: transparent;/\*去除原有边框\*/

background-color: #fff;

}

/\*chrome，safar等浏览器设置\*/

input[type='range']::-webkit-slider-thumb {

-webkit-appearance: none;

border: 3px solid #000;

height: 7px;

width: 7px;

border-radius: 8px;

background: red;

cursor: pointer;

}

/\*firefox 浏览器设置\*/

input[type='range']::-moz-range-thumb {

border: 3px solid #eee;

height: 7px;

width: 7px;

border-radius: 8px;

background: red;

cursor: pointer;

}

/\*IE浏览器设置\*/

input[type='range']::-ms-thumb {

border: 3px solid #eee;

height: 7px;

width: 7px;

border-radius: 8px;

background: red;

cursor: pointer;

}

/\*firefox浏览器已填充部分\*/

input[type=range]::-moz-range-progress {

background: linear-gradient(to right, red, white 100%, white);

height: 20px;

border-radius: 10px;

}

/\*IE浏览器\*/

/\*滑道未填充的部分\*/

input[type=range]::-ms-fill-upper {

height: 20px;

border-radius: 20px;

background: #ffffff;

}

/\*滑道已填充的部分\*/

input[type=range]::-ms-fill-lower {

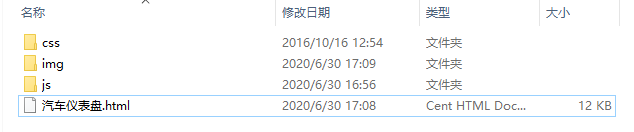
height: 20px;

border-radius: 20px;

background: linear-gradient(to right, red, white 100%, white);

}

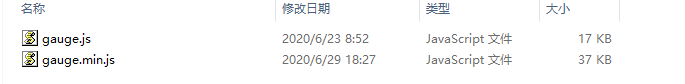
### 三.项目结构



css文件夹如下



js文件夹如下



### 四．总体效果图



### 五：总结

人机交互设计不仅仅是软件的设计，更多是在完成基本功能后，从用户的角度出发，通过良好的交互，减轻用户的负担，界面布局显示符合大多用户的审美，允许用户犯错，提高用户实际体验的好感度。

通过本次课程设计，我将课堂所学的知识运用到实践中，深刻地体会到在一个好的程序，除了功能外，页面的设计，合理的布局，可以帮助用户很快地找到用户的所需要的功能，同时带来观赏性、简便性；巧妙的色彩搭配更是可以很好的获得用户的注意力，给用户带来愉悦感，培养用户的粘性。

未来，我将继续学习完善人机交互的知识，开发程序，设计界面，为用户带来更加好的体验。