/\*\*

\* --------------------------------------------------------------------------

\* 本人对markdown用的不是很熟练，因此就用word来表达自己的思路了。。

\* --------------------------------------------------------------------------

\*\*/

CSS布局的部分比较简单，九宫圆的布局主要就是用了一个flex布局，基本都是rem和%的单位来适配移动端。下面说一说js实现功能的思路：

第一眼看到这个“作业”的时候，我的想法就是需要一个全局的状态量（STAUS）来表示“设置密码”与“验证密码”这两个状态，当手指点击“设置密码”按钮时将STAUS设置为“SET\_PASSWORD”，当手指点击“验证密码”按钮时将STAUS设置为“VERIFY\_PASSWORD”，这样就能通过验证STAUS来区分两个状态，让每个状态做该做的事。

然后就应该实现绘制连线的功能了，我觉得所有连线的绘制都需要一个“绘制线段”的基础函数，传入参数应为绘制连线的目标元素对象以及线段两个端点的横纵坐标，有了这个函数就能继而实现一端固定、另一端随着触摸点变动的即时射线。

但是些函数功能都是以坐标的形式绘制的，所以在这之前我还得建立一个“坐标系”，我以九宫格的父元素的左上角为顶点建立了一个坐标系，在网页加载初期先用一个全局的对象数组（CIRCLE\_INFO）保存九宫格中每个圆形按钮的圆心坐标、半径以及两个属性：isLinked和linkable，一个用来表示这个圆形按钮是否已经被连接，一个用来表示这个圆形按钮能否被连接。在“已被连接的按钮不能再被连接”这个准则下，这两个属性的意义是一样的，但是如果“已被连接的按钮允许再被连接”呢？一句话，为了功能的灵活性，多设置几个属性总是好的。为了省力，在构建九宫格圆形按钮信息的对象数组的时候使用了一个es5形式的构造函数，该构造函数只有公有属性，因为设置私有属性的话就需要公有函数去get、set私有的属性，有些麻烦，不如直接全都设为公有属性。

现在终于可以完成绘制线段的函数了，这个函数并没有用canvas实现，而是以操作dom的方式实现（每条线段其实就是一个div）.即时射线函数只是绘制线段函数的衍生。

绘制线段的函数以及衍生的绘制即时射线的函数写好之后接下来就是关键部分了：该怎样让触摸点移动到圆形按钮之内时,让线段的一端“吸附”进去，将即时射线的起始固定点替换为这个点。这里我的想法是设置2个（不如说是“一对”）全局变量：CUR\_POINT\_X和CUR\_POINT\_Y用来保存即时射线的起始固定点的x、y坐标值，显然这一对全局变量的值必为一对圆形按钮的圆心坐标值。当手指在屏幕上移动时，时刻计算触摸点到9个圆形按钮的距离，找出距离最短的那一个值及对应的那个圆形按钮在九宫格中的位置（索引），如果这个值小于圆形按钮半径的话，OK，绘制一条线段，起始点的横坐标是之前的CUR\_POINT\_X、CUR\_POINT\_Y值，终点是这个距离触摸点最近的圆形按钮的圆心，然后把这个圆心的x、y坐标值赋给CUR\_POINT\_X、CUR\_POINT\_Y，再绘制一条起始固定端点的x、y坐标值是更新之后的CUR\_POINT\_X、CUR\_POINT\_Y的即时射线，然后把这个圆形按钮的索引值加进全局数组PASSWORD\_ARR，没错，这就是密码的雏形，同时还可以根据这个数组的长度来判别连线的按钮是否少于5个。

这个逻辑完成之后我们就可以像解手机屏幕锁时一样绘制解锁图形啦！

接下来就是要在触摸点离开手机屏幕时让用户进行第二次手势操作或者设置成功，想到这些我立马想到还需要一个判别：什么时候让用户进行第二次手势操作，什么时候让用户密码设置成功。我的想法是设一个全局变量SUCCESSED\_TRY\_COUNT用来保存用户这是第几次进行手势操作，初始为0，第一次手势通过之后将该值设为1，第二次手势通过之后将该值还原为0。每当touchmove结束的时候（touchend）检验SUCCESSED\_TRY\_COUNT的值，如果为0，将PASSWORD\_ARR转为用哈希算法加密的密码存在全局变量PASSWORD中，并显示“请再次输入手势密码”；如果为1，判定此次手势操作获得的密码与之前PASSWORD中密码的正确性，如果错误，显示“两次输入的密码不一致，请重新输入”，并将SUCCESSED\_TRY\_COUNT设回0；如果密码正确，将PASSWORD中的密码（已加密）存入localStorage中，并显示“密码设置成功！”，将按钮跳转至“验证密码”，同时将STATUS改为“VERIFY\_PASSWORD”。当然以上的检验或者判定的前提都是“STATUS= VERIFY\_PASSWORD”。

然后到了检验密码的环节，这个就比较简单了，只要在STATUS=SET\_PASSWORD的前提下将手势操作获得的密码（已加密）与localStorage中的密码（同样经过加密）相比较，如果正确的话，显示“密码验证成功”，开启新世界的大门，如果错误的话，显示密码错误。

在每次判定进入一个模块操作之后都得让一些全局变量还原回undefined，以便下一次的函数执行，因此我写了一个resetCache函数来重置一些全局变量。

到此，整个项目基本告一段落，后续的都是一些锦上添花的小修改，对基础逻辑及功能没有影响，例如加了一个用来进行参数设置的对象，里面可以对线条的颜色、宽度，圆形按钮的背景色等等进行修改。

在线预览：

<http://htmlpreview.github.io/?https://github.com/namedCJ/myFirstRepository/blob/master/手势解锁/index.html>