在线实验，请到PC端体验

基于 CSS3 实现抽奖大转盘

一、实验介绍

1.1 实验内容

本次课程将会教会大家如何使用 CSS3 的特有属性 `transform` 制作抽奖转盘，过程简单且富有乐趣，并在课程中穿插了一些留给大家的思考题，希望大家积极参与并完成思考题，会有很多收获哦。

效果图：

1.2 实验知识点

- CSS3 transform
- jQuery

1.3 实验环境

1. `Terminal`：Linux 命令行终端，打开后会进入 Bash 环境，可以使用 Linux 命令
2. `Firefox`： 浏览器，可以用在需要前端界面的课程里，只需要打开环境里写的 HTML/JS 页面即可
3. `GVim`： 非常好用的编辑器，最简单的用法可以参考课程 Vim编辑器 (http://www.shiyanlou.com/courses/2) （或者使用 `Sublime Text` 编辑器）

实验楼的 `Sublime Text` 编辑器位于：应用程序菜单->开发 下。

1.4 适合人群

本实验难度中等，需要有 HTML, CSS3, jQuery 基础

1.5 代码获取

源代码可通过下述命令获取：

```
$ git clone https://github.com/shiyanlou/lottery
```

二、开发准备

首先建立下面的项目目录结构：

```
lottery
    |__lottery.html
     |__css
    |    |__lottery.css
    |__js
     |    |__lottery.js
     |__img
```

然后我们需要下载实验用到的图片到 `img` 目录下：

```
wget http://labfile.oss.aliyuncs.com/courses/82/20yuan.jpg
wget http://labfile.oss.aliyuncs.com/courses/82/apple.jpg
wget http://labfile.oss.aliyuncs.com/courses/82/iPhone6plus.jpg
wget http://labfile.oss.aliyuncs.com/courses/82/ipadmini.jpg
```

再到 `js` 目录下下载 `jQuery`：

```
wget http://labfile.oss.aliyuncs.com/courses/82/jquery.js
```

三、实验步骤

首先来分析下抽奖转盘的特点,确定一下设计的整体构思。

以一个从没写过抽奖转盘的新手角度来考虑，大概会遇到一下几个难点：

- 1、转盘如何才能开始旋转；
- 2、转盘的速度控制。为了更加逼真的模拟现实中的转盘旋转，需要将转盘速度控制为 慢-快-慢；
- 3、如何等分切割转盘的扇形区域；
- 4、奖品是如何随机抽取的；
- 5、如何判定旋转已经结束；

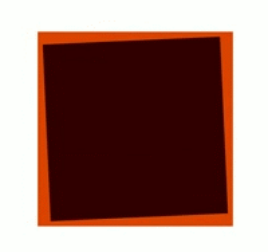
接下来，就将逐个解决以上问题，同时，也一步步完成我们的抽奖转盘。

3.1 让转盘旋转起来

如何让转盘旋转呢？有的同学会选择 CSS3 中的动画，这当然是解决问题的一个好方法，也是大家最为熟悉的一种方法，在这给大家介绍另一个与动画相似的属性 -- 过渡属性 `transition` 。

刚开始，我们就用一个 div 表示整个转盘，利用过渡属性让这个 div 开始旋转。

效果演示：



CSS3 的过渡效果 `transition` 是当元素从一种样式变换为另一种样式时为元素添加效果。可以在鼠标单击、获得焦点、被点击或对元素任何改变中触发。为方便演示，我们通过伪类 `:hover` 来触发过渡效果：

html 代码

```
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
        <link rel="stylesheet" href="css/lottery.css"></link>
        <script type="text/javascript" src="js/jquery.js"></script>
    </head>
    <body>
        <div class="box">
            <div class="lottery"></div>
        </div>
        <script type="text/javascript" src="js/lottery.js"></script>
    </body>
</html>
```

CSS 代码

```
* {
    margin: 0;
    padding: 0;
}
body {
    font-family: "Microsoft YaHei",微软雅黑;
}
.box {
    width: 600px;
    height: 600px;
    background-color: #d40;
    margin: 10px auto;
    padding: 20px;
    position: relative;
}
.lottery {
    width: 600px;
    height: 600px;
    margin: 0 auto;
    position: relative;
    background-color: #330000;
    box-shadow: 0 0 15px #000;
}
.lottery:hover {
    transform: rotate(1822.5deg);
}
```

基本的样式设计代码，相信大家都很熟悉，这里主要讲解 transition 的使用，它主要包含四个属性值：

- transition-property： 规定应用过渡的 CSS 属性的名称；
- transition-duration： 定义过渡效果花费的时间，默认是 0；
- transition-timing-function： 规定过渡效果的速度曲线，默认是 sase；
- transition-delay： 规定过渡效果间隔多长时间开始，默认是 0。

除 transition-timing-function 属性外的三个属性都容易理解，这里在详细说明一下此属性的具体含义：

规定了过渡效果的速度曲线，即使得过渡效果能随着时间来改变其速递。此属性共有 6 个值：

- liner： 规定以相同速度开始至结束过渡效果（等于 cubic-bezier(0,0,1,1)）；
- ease： 规定慢速开始，然后变快，然后慢速结束过渡效果（等于 cubic-bezier(0.25,0.1,0.25,1)）；
- ease-in： 规定以慢速开始的过渡效果（等于 cubic-bezier(0.42,0,1,1)）；
- ease-out： 规定以慢速结束的过渡效果（等于 cubic-bezier(0,0,0.58,1)）；
- ease-in-out： 规定以慢速开始和结束的过渡效果（等于 cubic-bezier(0.42,0,0.58,1)）；
- cubic-bezier(n,n,n,n)： 自定义过渡效果，数值范围 0 ~ 1。

同学们可以在实例中测试不同的值，以方便更好的理解其区别和原理。

这样，将鼠标滑到棕色的 div 上时，过渡效果便开始了。而在实际的抽奖转盘中，你可以用点击触发过渡效果的执行，这样，我们便解决了转盘如何旋转的问题。

那么第一个思考题来了，思考题**1**：

演示中是使用了CSS3中的过渡属性，请你用CSS3中的动画效果来让转盘旋转起来，并与过渡效果方法作比较。

3.2 、控制转盘的速度

相信很多同学已经发现，其实在使转盘旋转中， transition-timing-function 属性已经很好的帮助我们解决了这个问题，我们只需要设置其值为 ease-in-out 便能够实现转盘以 慢-快-慢 的速度旋转起来。

在这里需要需要额外提出的一点就是，我们常使用的 animation 动画也能通过 animation-timing-function 设置其速度曲线。

思考题**2**:

很简单，用 animation-timing-function 来控制你的转盘旋转速度变化曲线吧，真的很简单，积极动手吧。

3.3 设置环绕一圈的区域

这是制作转盘的一个难点，没做过的可能会想到各种的方法来拼凑出这样一个扇形，但最后很可能造成的 **html** 标签的繁琐嵌套、代码不简洁、后期维护困难等弊病，并且，能否成功的制作出一个转盘，在这一步便可以很直观的体现出来。所以这一部分会作为一个重点，详细讲解。

还是上一个演示的代码，在 **html** 代码中添加“扇形块”（即 8 个 **div**）。

```
<div class="box">
    <div class="lottery">
        <div class="block"></div>
        <div class="block"></div>
        <div class="block"></div>
        <div class="block"></div>
        <div class="block"></div>
        <div class="block"></div>
        <div class="block"></div>
        <div class="block"></div>
        <div class="block"></div>
    </div>
</div>
```

CSS代码添加：

```
.block {
    box-shadow: 0 0 1px #000 inset;
    position: absolute;
    width: 300px;
    height: 300px;
    transform-origin: right bottom 0;
    -webkit-transform-origin: right bottom 0;
}
.block:nth-child(1) {
    transform: rotate(0deg);
}
.block:nth-child(2) {
    transform: rotate(45deg);
}
.block:nth-child(3) {
    transform: rotate(90deg);
}
.block:nth-child(4) {
    transform: rotate(135deg);
}
.............
```

注意： **box** 、 **lottery** 、 **block div** 的 **position** 属性设置，这是对扇形块的制作很重要， **block** 设置边框阴影和蓝色是为了方便区分每一个 **div**，方便开发，这也是前端开发者开发时常用的技巧

添加 8 个 **div**，也就是形成我们最后需要的 8 个扇形。相信大家都知道 **div** 在未被设置特殊样式时是块状元素，即一个四边形展示出来，要改造为扇形，就需要用到 **CSS3** 中 **2D/3D** 转换。

注意：这之后会一边添加代码，一边讲解代码的作用，每一步都是实现最后效果必不可少的。

transform-origin 属性可以自定义被转换元素的位置，即可以看做自定义元素的坐标原点。原点设置的不同，其转换后的结果也必然不同，所以选择好修改的做原点位置也是需要认真选择的。

设置好原点（这里是右下角那一点）后，需要旋转每一个 **div**，由于每一块的旋转角度不同，所以这里需要单独设置。角度计算也很简单，**360** 除以总的块数就得到了旋转的角度。代码中也只列举了前 4 个 **div**,省略了后 4 个 **div** 的样式代码，接下来是思考题时间。

思考题**3**:

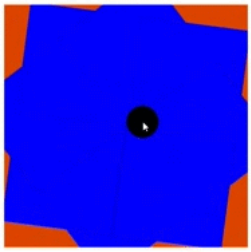
补全后 4 个 **div** 的样式代码，但不会那么简单就结束了。相信你也会认为每一个 **div** 都要写一遍样式很麻烦，那么请你通过编写 **js** 代码，遍历 **div** 设置每一块对应的样式吧。

这一步完成后，便能看到蓝色 div 绕成了一个圈，鼠标滑到 div 之上时，旋转的过渡效果也不受影响。

但是细心的同学会发现滚动条会不停的改变，虽然不影响功能，但是，就连不是处女座的同学也会忍不了的吧。

此时，只要将 `box` 设置属性 `overflow:hidden;` 即可。

效果如图：



3.4 拉扯矩形，颜色分隔

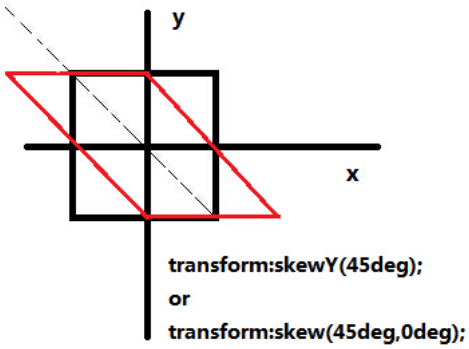
完成上一步，有的同学会心中产生疑问：什么？这是转盘吗？

嗯~~~问的好，这还只是完成了一半的步骤，完成这一步，你将会看到熟悉的抽奖转盘了。

数一数，上个效果图中，从中心发散的线条将蓝色部分分成了多少块呢？没错，是7块。

之所以没有 8 块都显示是因为 html 元素的加载顺序将最早加载的元素遮盖了，想要 8 块都显示，那还是需要用到 CSS3 的 `transform` 属性中的 `skew(Xdeg,Ydeg)`，当然你也可以简写为 `skewX(Xdeg)` 和 `skewY(Ydeg)`。

这是 CSS3 中 2D/3D 转换的拉扯（倾斜、斜扯，ps：名称可按自己的理解来取），用法见一下图解：



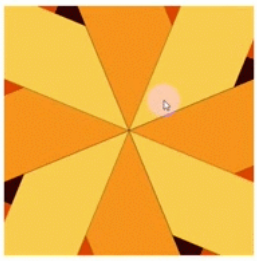
黑线框经过拉扯后便会形成红线框的形状，默认转换原点是元素的中心点，修改原点和拉扯方向、角度都是转换形状的途径之一。各位同学可以在实例中多多测试，体会拉扯的原理，在纸上画坐标，转换结果等，也是个不错的有助理解的方法。

加上拉扯转换的 css 代码：

```
.block:nth-child(odd) {
    background-color: #F7941C;
}
.block:nth-child(even) {
    background-color: #FDCD51;
}
.block:nth-child(1) {
    transform: rotate(0deg) skewY(45deg);
}
.block:nth-child(2) {
    transform: rotate(45deg) skewY(45deg);
}
.block:nth-child(3) {
    transform: rotate(90deg) skewY(45deg);
}
.block:nth-child(4) {
    transform: rotate(135deg) skewY(45deg);
}
.........
```


在原有基础上加上 `skewY(45deg)`，因为是8块，所以拉扯的角度自然也就是 360/8=45 度，并且每一块都一样。

再通过 css 选择器可以交叉设置颜色，完成此步后的效果如下图：



咋一看很像一个风车，但已经越来越向我们的转盘靠近了。

注：以上两步是制作转盘最重要的两步，需要大家好好理解

思考题**4**：

一个好的项目，不论大小，后期的维护性是一个关键，如果有一天需要添加或删除一个扇形块，显然这个转盘就需要重头计算每一块，那请你思考是否能够有效合理的解决这一问题呢？

思考题**5**：

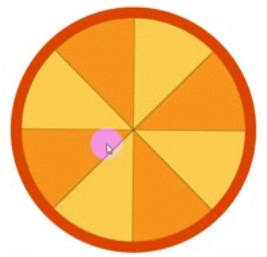
像最开始的效果图那样，试着为你的转盘添加渐变的颜色，使其更为立体。

3.5 切方为圆

这一步比较简单，只需要在 box 和 lottery 中添加以下样式即可：

```
.box {
    border-radius: 50%;
    ...
}
.lottery {
    border-radius: 50%;
    overflow: hidden;
    ...
}
```

效果如图：



3.6 放上奖品吧

转盘的大致形状都完成了，在放上奖品和开始按钮即可。这里放一个奖品为例。

html 代码中添加：

```
<div class="block">
    <div class="content">
        <div class="none">再接再厉</div>
    </div>
</div>
<div class="block">
    <div class="content">
        <div class="text">
            <p>一等奖</p>
            <p>iPhone6 plus</p>
        </div>
        <div class="img">
            <img src="img/iPhone6plus.jpg"/>
        </div>
    </div>
</div>
```

动手实践是学习 **IT** 技术最有效的方式！ 开始实验

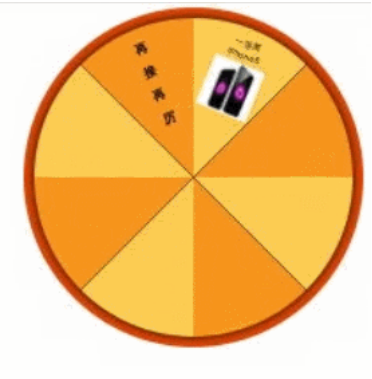
CSS 代码中添加：

```
.content {
    width: 200px;
    height: 230px;
    transform-origin: center center 0;
    transform:  skewY(-45deg) rotate(-22.5deg) translate(10px, 70px);
    -webkit-transform:  skewY(-45deg) rotate(-22.5deg) translate(10px, 70px);
    position: absolute;
    right: 0;
    bottom: 0;
    text-align: center;
}
.content .img {
    width: 100px;
    height: 100px;
    margin: 0 auto;
}
.content .text {
    width: 100px;
    height: 50px;
    margin: 0 auto;
}
.content .none {
    width: 30px;
    height: 200px;
    margin: 0 auto;
    text-align: center;
    font-size: 23px;
    line-height: 200%;
    font-weight: bold;
}
```

还是为了便于区分 div，我们为 div 加上了边框阴影。

由于 block 作了拉扯转换，因此，子元素 content 也同样被拉扯了，为了在转盘中显示正常，我们需要在对 content 进行拉扯，同时还需要调整其位置(修改其 width 、 height 、 right 、 bottom 、 translate() .....)，这是需要细心的一步。

效果如下：



3.7 开始按钮

我们在 box 里面与 lottery 并列放置一个 div：

```
<div class="btn">开始抽奖</div>
```

然后为其添加样式：

```
.btn:before {
    display: block;
    content: "";
    position: absolute;
    top: -30px;
    left: 30px;
    border-bottom: 40px solid #d40;
    border-left: 20px solid transparent;
    border-right: 20px solid transparent;
}
.btn {
    width: 100px;
    height: 100px;
    border-radius: 50%;
    text-align: center;
    line-height: 100px;
    background-color: #d40;
    position: absolute;
    top: 270px;
    left: 270px;
    cursor: pointer;
    box-shadow: 0 0 7px #000 inset;
    color: #fff;
}
```

添加了按钮我们就可以将 `.lottery:hover` 删除，改成鼠标点击触发了，在 `lottery.js` 添加下面的代码：

```
function clickBtn(){
    var n = Math.floor(Math.random()*8+1);
        lottery.p = 45 * n + 1822.5;
    $(this).unbind().css("cursor","wait");
    $('.lottery').css({
        "transition":"all 5s ease-in-out",
        "transform":"rotate("+lottery.p+"deg)",
        "-webkit-transform":"rotate("+lottery.p+"deg)"
    });
}
$('.btn').bind('click',clickBtn);
```

3.8 抽取奖品

认真完成以上步骤的同学，一定已经看到了转盘已经大致完成了，但是既然是抽奖转盘，那么如何随机抽取出奖品也是大家最为关心的问题。

既然是随机，那么自然会想到 `Math.random()` 方法，但这还需要与我们的转盘关联起来。回头看看我们是如何控制旋转的，没错，角度，并且每一扇形块中心线对应的角度都是很容易知道的。因此这样一分析，是不是发现这个问题也非常简单呢。

列举一个关联的例子：

我们使用数组按顺序保存每一块奖品，再根据角度与圈数的比值，自然就得到了对应的数组元素，也就“抽取到了奖品”，当然，角度是随机产生的，保证了抽取奖品的随机性。

代码如下：

```
var lottery = {
    light:"odd",
    prize:["鼓励奖","再接再厉","三等奖","再接再厉","二等奖","再接再厉","一等奖","再接再厉"],
    p:0
}

function result(){
    var num = (lottery.p - 1822.5)/45;
    if(num % 2 == 0){
        alert("很遗憾，请"+lottery.prize[num-1]+"!");
    }else{
        alert("恭喜您，获得"+lottery.prize[num-1]+"!");
    }
}
```


思考题**6**：

使你的脑洞打开，想想是否还有其他更好的抽奖办法呢？

思考题**7**：

大奖人人都想得到，但总是离我们那么遥远，你如何来控制大奖被抽取的几率呢？

思考题**8**：

绝对不能忍受 **alert( )** 作为最后的结果提示，希望大家动手将这个修改为一个动态的结果提示特效。

3.9 旋转结束判定

在上一步中，有一个 `result()` 方法，显然这是转盘结束后会弹出抽取到哪一个奖品的提示，但是写到这一步你可能会产生这样一个疑问：我什么时候才能知道旋转结束了呢？

能这样想，说明你已经认真投入到了项目课中，十分难得。但问题总有解决办法，**js** 早就为我们考虑到了这一步，接下来为大家介绍很少用到的 `transitionend` 和 `animationend` 事件。

由名字我们可以想到这是过渡效果和动画效果结束时的事件，我们需要做的就是添加事件监听，这遗留的最后一个问题也完美的解决了。

代码如下：

```
$('.lottery')[0].addEventListener('transitionend',function(){
    $('.btn').bind('click',clickBtn).css("cursor","pointer");
    result();
    $(this).css({
        "transition":"none",
        "transform":"none",
        "-webkit-transform":"none"
    })
})
```

由代码我们看到，过渡效果结束后，我们还原了转盘的旋转角度，有时我们会想能否不还原转盘，继续点击按钮开始旋转呢？这就留作思考题吧。

思考题**9**：

实现过渡效果结束后不还原转盘，继续点击旋转。

四、实验总结

本实验主要基于 **CSS3** 实现了酷炫的抽奖大转盘，关键在于对 **CSS3 tansform** 等一些属性的灵活运用。希望大家多多思考实验中的问题，多多练习。

五、课后习题

思考题：绘制一圈闪烁的彩灯吧，让你的转盘更酷炫！