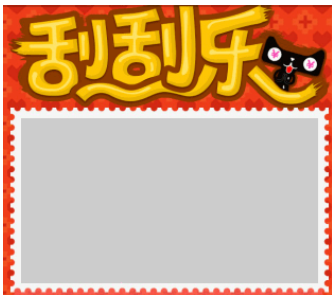


1 开发跨平台刮刮乐应用

1.1 引例描述



亲，请使劲的刮屏幕，就可以刮出巨奖哦.....刮坏 7 块屏幕者可以召集神龙！这只是个玩笑，现在用手指涂抹就可以刮出奖啦，妈妈再也不用担心刮奖弄脏我的指甲！“刮刮乐”有即开、即中即兑、趣味性强等特点，常见于淘宝、京东等电商网站。用户在买完喜爱的商品之后，常常会进行刮刮乐游戏，给用户期待，精美的礼品更是吸引了大量的用户。经过连续的项目练习，小新已经掌握了不少前端开发的技术，但是拷贝到手机上测试也麻烦，每次都用模拟器调试让同事给意见也有很多不便，如何能快速让同事体会到效果并直接给出评价意见呢？结合互联网营销活动的需求，小新要尝试自己完成可以网上体验的刮刮乐。本单元将结合静态服务器 Nginx 的搭建和使用和刮刮乐游戏这两个任务，实现局域网内终端可访问的在线刮刮乐游戏。针对刮刮乐游戏的实现主要处理 3 个待解决的问题：

- 1、如何设计一个公平的抽奖算法，比如控制抽奖 100 次中只有 1-2 次一等奖的机会；
- 2、避免用户通过工具等手段破解抽奖活动，比如通过工具分析出本次抽奖的结果。
- 3、真实在网络中实现刮刮乐刮奖的效果？

1.2 任务陈述

表 1-1 学习任务单

一、达成目标
1. 掌握 Nginx 的配置与用法、理解终端访问服务器的过程并在手机端运行部署的项目；
2. 能够掌握 canvas 操作，理解递归算法并调试；
二、学习任务
1.掌握 Nginx 服务器的配置文件的修改方法；
2.使用 Canvas 设置蒙板并进行擦除操作，可以在 Chrome 浏览器中查看实现效果；
3.设计抽奖算法、并能够优化防止抽奖作弊的功能；
4.理解页面布局，能够应用盒子模型处理边界设置；
5.以小组的方式对完成的页面进行讨论，记录未能成功修改的需求与教师讨论；
三、参考资源
1. 在智慧职教平台 http://www.icve.com.cn/portal/ 注册并学习课程；
2. 搜索和学习正则表达式,可参考 https://developer.mozilla.org/zh-CN/docs/Web/

将遇到的难点和不清晰的地方通过截屏等方式记录，准备与教师面对面讨论问题。

1.3 知识准备

1.3.1 知识点 1 搭建 Nginx 和测试方法



Nginx 是 HTTP 静态服务器,比较著名的还有 Apache。选 Nginx 的原因是使用者众多,包括很多视频网站,大型电商都在使用,还可以方便的和 Node.js 配合使用。

Nginx 的具体下载和按照过程可以参考其它文章,安装后可以看到如下文件结构:

名称	修改日期	类型	大小
conf	2016/4/24 21:53	文件夹	
contrib	2015/4/21 17:13	文件夹	
docs	2015/4/21 17:13	文件夹	
html	2016/9/17 18:26	文件夹	
logs	2016/4/15 22:16	文件夹	
temp	2016/4/5 19:24	文件夹	
www	2016/6/30 20:04	文件夹	
index.js	2016/4/24 21:08	JS 文件	1 KB
nginx.exe	2015/4/21 16:44	应用程序	2,691 KB

Nginx 包含的文件结构

其中 nginx.exe 是 windows 环境下的执行程序,也是启动 Nginx 服务器的入口。其中 conf 文件夹是配置 Nginx 服务器的配置文件夹,里面有许多的配置,本单元只需要重点关注 nginx.conf 文件即可,使用工具比如 VS code、webstorm 或者文本工具打开该文件,输入以下配置代码

```
http{
  ....//略
server{
  ...//略
}

#设置访问静态资源目录配置
location ~* ^.+\. (ico|gif|jpg|jpeg|png|html|htm|css|js|xml|txt)$ {
    root          OUYANG;
    access_log    off;
    expires       30d;
}
```


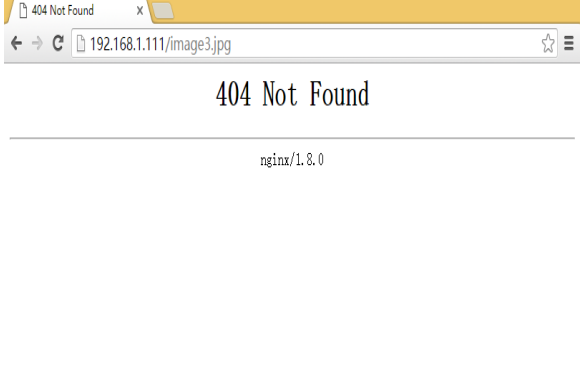
<pre>} </pre>
配置代码

注：请注意以下代码只是参考，输入时请以手打为主，直接复制可能有格式问题，造成奇怪的错误。



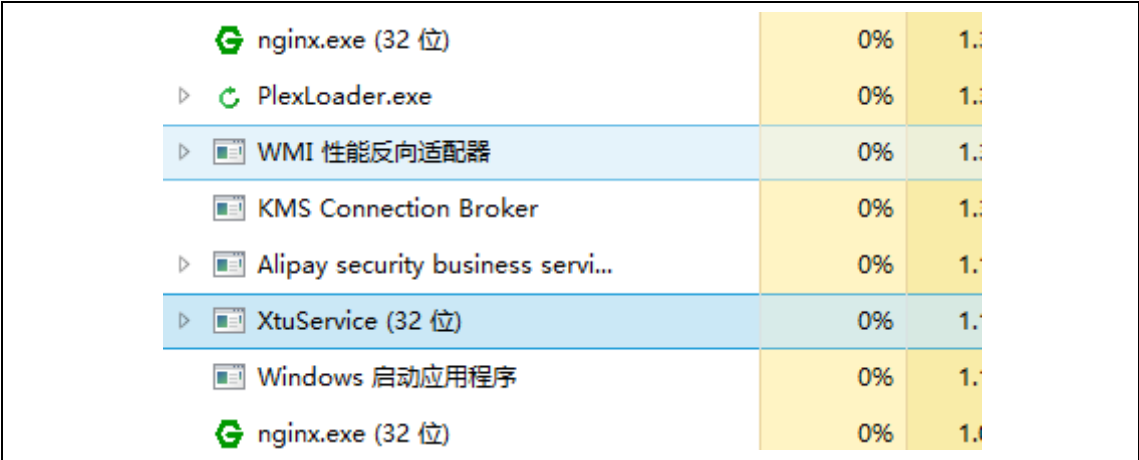
以上配置的意思是告诉 Nginx，一旦接收到带有 .ico|.jpg|.txt 等后缀名的请求,就访问对应的资源目录,比如 http://192.168.1.111/image2.jpg，当接收到这个请求，分析请求包含 image2.jpg 结尾，服务器就去静态资源目录寻找有没有 image2.jpg 的图片，有则显示，没有则显示 404 not found。

以下是配置成功后在 Nginx 服务上放入了一个名称为 image2.jpg 的文件在服务器上，重启服务器，访问静态资源的效果，如图所示：

	
请求成功	请求失败

这里需要注意如果路径中包含中文信息，有较大可能无法启动 Nginx 服务。对于浏览器要访问的文件路径的问题，配置代码中有一项配置进行了重点标记，它是一个根目录，所以符合请求的条件都会从根目录（root）需找对应的资源，因此相应的服务器上也要有对应的根目录文件夹。

不管是第一次启动 Nginx 还是配置完后启动 Nginx，都可以在任务管理器中查看服务器是否启动成功，如果程序列表中没有出现名字 nginx.exe，表示启动失败。


启动成功后的界面



由于没有提升导致 Nginx 启动失败的原因 ,具体解决问题时就必须使用命令

行界面来解决了，运行 CMD 命令，将 Nginx 目录定位到命令行模式中。

```
C:\Program Temp\DevelopmentTools\nginx-1.8.0>
```

CMD 当前路径位置

使用命令行界面启动 nginx.exe (直接输入执行文件名字即可) 便可以看到 nginx 启动时抛出的信息或错误，一般会告诉开发者配置文件的那一行错了再通过网络搜索解决方法。

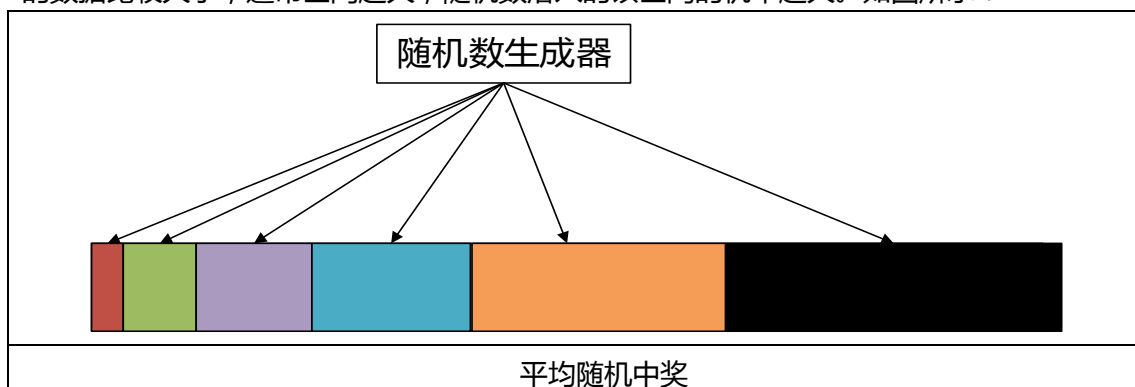


另外大家可能会逐步关心浏览器是如何工作的，浏览器最重要或者说核心的部分“浏览器内核”负责对网页语法的解释(HTML、JavaScript) 并渲染 (显示) 网页的内容以及页面的格式信息。例如点击一个 HTML 的<a>标签链接打开一个新页面，看起来非常理所当然的一件事情，而实际在目前常见的 WebKit 内核里面，要做非常多处理。首先要获取当前点击的位置，带着这个位置坐标信息，执行 HitTest。HitTest 将探测下当前的点击的内容是什么，怎么才能知道当前点击的是什么呢？HitTest 会遍历整个 DomTree, LayerTree 查看当前的坐标位置是什么内容：空白、文字、还是图片、链接？这些信息都可以通过遍历 DomTree, LayerTree 来实现，最终将遍历的探测结果以 HitTestResult 的形式返回，最终通过返回的类型来判断，如果是一个链接，则回调对应的回调函数，最终实现通过点击打开一个新的页面，完成服务器支持终端访问的服务。

1.3.2 知识点 2 控制中奖概率与种子函数



人们只在乎“抽到头奖”，对于任何一个抽奖类游戏来说让人感到幸运是很重要的体验，用户的活跃度至关重要。在设计前需要分析清楚中奖概率才可以进行设计，设总概率在【0-100】之间，对这些得奖的奖品做一个区间的划分，可以把它们的得奖率看成一个区间的大小，根据计算出来的随机数值，判断该数值落在了哪个区间内，也就是使用随机值数据和区间的数据比较大小，通常区间越大，随机数落入的该区间的机率越大。如图所示：



由此可知上图的随机系统使用随机数作为抽奖核心，把不同占比区间进行比较，奖品占比越大，被抽中的可能性也越大。计算机不会产生绝对随机的随机数，计算机只能产生“伪随机数”。伪随机数并不是“假的随机数”，这里“伪”是指有规律的就是计算机产生的伪随

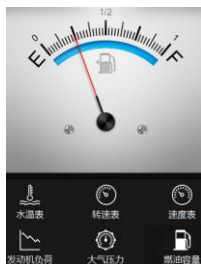
机数既是随机的又是有规律可循。即伪随机数有一部分遵守一定的规律；另一部分不遵守任何规律。比如“世上没有两片形状完全相同的树叶”，即随机性，但是每种树的叶子都有近似的形状即规律性。



随机数是怎样产生的呢？随机数是由“随机种子”产生的。随机种子是用来产生随机数的一个无符号整形数，随机种子是从哪里获得的呢？如果不设置随机种子，在默认情况下随机种子来自系统时钟（即定时/计数器的值），由计算机时钟计数器精确控制时间间隔的控件，时间间隔相同，计数器前后的值之差相同，实际这种时钟取值就有线性规律，所以随机种子是呈线性规律的，最终生成的随机数也是有规律的，而产生的随机数字重复率一旦被发现用户的兴趣就会立即下降。

有些物理事件如用户按键事件产生随机数更呈现随机性，因为人按键不能保证严格的按键时间间隔，即使间隔相差一微秒，计数器前后的值之差就不相同了，对应随机种子的变化就失去了规律，生成的随机数就更没有规律了。这样生成的一组随机数就体现了随机性原则，结果就会更公正。

1.3.3 知识点 3 应用面向对象的设计思想



在 JavaScript 中没有明显的语法可以支持面向对象的开发，但 JavaScript 是一个很灵活的语言，可以利用 JavaScript 自带的一些特性模拟面向对象的思想，模拟面向对象的方法有很多种，这里采用 Prototype（原型）的方法来实现。如前文中提到的抽奖游戏，其中礼品可以作为一个对象来设计，它具有一些属性，比如名字，id，位置，区间等属性，并且可以由外部读/写实例中的属性值。

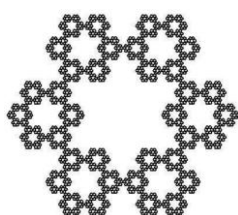
```
//以下为 javascript 原型设计方法 /** * 奖品对象 */
function Gift(index, id, name, startArea, endArea){
    this.index = index;
    this.id = id;
    this.name = name;
    this.startArea = startArea;
    this.endArea = endArea; }
Gift.prototype.getIndex = function () {
    return this.index; };
Gift.prototype.getGiftName = function () {
    return this.name; };
Gift.prototype.getId = function () {
    return this.id; };
Gift.prototype.getStartArea = function () {
    return this.startArea; };
```

<pre> Gift.prototype.getEndArea = function () { return this.endArea; }; Gift.prototype.toString = function () { return "Gift: " + "index=" + this.index + ", id=" + this.id + ", name=" + this.name + ", startArea=" + this.startArea + ", endArea=" + this.endArea; }; </pre>
创建抽奖对象

使用方法也简单，比如在抽奖系统中创建不同等级奖项后可使用一个数组保存起来，代码如下：

<pre> /**礼品数组*/ listGifts: [], core.listGifts.push(new Gift(1, "g1", "1 bonus", 0, 1.0)); core.listGifts.push(new Gift(2, "g2", "2 bonus", 1.0, 20.0)); core.listGifts.push(new Gift(3, "g3", "3 bonus", 20.0, 50.0)); core.listGifts.push(new Gift(4, "g4", "4 bonus", 50.0, 100.0)); //可以通过访问下标 (index) 的方法访问礼品实例的属性 //core.listGifts[0]. getGiftName === "1 bonus" </pre>
创建奖项数组

递归思路



程序调用自身的编程技巧称为递归。它通常把一个大型复杂的问题层层转化为一个与原问题相似的规模较小的问题来求解，递归策略只需少量的程序就可描述出解题过程所需要的多次重复计算，大大地减少了程序的代码量。一般来说，递归需要有边界条件、递归前进段和递归返回段。当边界条件不满足时，递归前进；当边界条件满足时，递归返回。

对于上面的对递归的定义，简单的解释是，让程序自己调用自己，在 JavaScript 就是让函数本身调用函数，举个栗子如下代码：

<pre> //伪代码 function recursion(){ recursion(); } </pre>
递归调用自身程序

上面代码就是一个简单的递归定义，假设上面代码成立，函数自身调用函数，无穷无尽，程序会一直执行下去会变为死循环吗？

前文中的递归解释中还有提到，递归需要有边界条件（也就是一个判断条件，当满足某

个条件时结束递归), 在满足或者不满足的情况下要可以终止递归代码的判断条件。

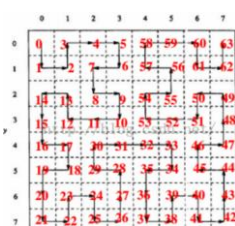
把递归的设计思想带入抽奖系统可以发现比如随机数和礼物占比之间的值计算中可应用递归的思想。不管礼物实例有多少, 只需要定义随机数如何跟礼物属性计算的规则即可。

```
var equalRaw = function (giftIterator, raw) {  
  //边界条件  
  if(giftIterator.hasNext()){  
    var gift = giftIterator.next();  
    if(raw > gift.getStartArea() && raw < gift.getEndArea()){  
      switch (gift.getIndex()){  
        case 1:  
          break;  
        //省略中间代码.....  
        default :  
          break;  
      }  
      return;  
    }  
    equalRaw(giftIterator, raw);  
  }  
}
```

递归匹配奖项

上面的代码中边界条件就是判断数组是否还有礼物对象, 有则比较。没有则结束迭代, 而随机值和区间占比比较的方法也非常简单, 通过访问礼物实例的属性值和传递进来的参数进行大小比较。

迭代器设计思想:



前文代码中的边界条件判断看起来有些奇怪, 传入了 giftIterator 参数, 这个 giftIterator 对象拥有 hasNext()和 next()方法。迭代器是一种设计模式, 它是一个对象可以遍历并选择序列中的对象, 而开发人员不需要了解该序列的底层结构。迭代器通常被称为“轻量级”对象, 因为创建它的代价小。本项目使用迭代器是为了更好的服务于递归的设计思路, 使代码设计简洁明了。通常迭代器拥有 2 个基本的方法 hasNext(判断下一个数据是否为空)和 next(取得下一个数据)。代码设计如下:

```
iterator = function (array) {  
  var index = 0;  
  return {  
    hasNext: function(){ //检查序列中是否还有元素  
      return index < array.length;  
    },  
  }
```

```
next: function(){    //获得序列中的下一个元素
    return this.hasNext() ? array[index++] : null;
},
}
}
```

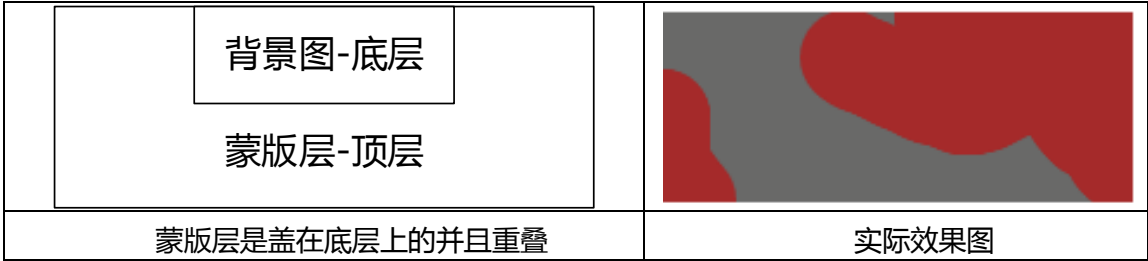
迭代器的实现

上图代码使返回的对象拥有 hasNext 和 next 方法 ,可以看到这 2 个方法其实是对原始的 Array 对象的属性和方法进行了包装 ,使它对外提供一致的服务 ,这样以来隐藏了对 Array 对象的操作细节 ,对外提供一致的方法 (hasNext 和 next), 以后只需传入 Array 类型参数就可以方便的取出对象 ,而不必让开发者关心具体的细节 (比如如何判断为空 ,如何取对象等)。

1.4 任务实施

1.4.1 任务 1 设置蒙板

刮刮乐待解决问题的刮奖效果实际上使用 “一个背景图 + 覆盖一层蒙版” 实现的 ,并且这个蒙版还应可以被擦除。蒙版层由 javascript 代码动态创建 ,它的擦除效果也由代码控制 ,示意图如下所示 :



灰色层就是蒙版层 , 可以被擦除。深红色是底层不可以被擦除 , 用来展示刮奖的结果。

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<!-- 兼容手机-->
<meta name="viewport" content="width=device-width, initial-scale=1.0,
minimum-scale=1.0, maximum-scale=1.0" />
<title>刮刮乐抽奖活动</title>
<link rel="stylesheet" type="text/css" href="css/index.css">
<script src="js/index.js"> </script>
</head>
<body onload="core.init()">
```



```

<div id="background"></div>
</body>
</html>

//-----CSS 代码-----
#background{
background-color: brown;
width:200px;
height:80px;
}

```

参考的 Html 内容

底层的展现已经实现完毕，实际上擦除效果仍然使用 Canvas 实现，默认给 Canvas 设置灰色的颜色，并且设置 Canvas 为透明（这个很重要，决定了刮奖的效果，可以去掉试试看会发生什么）。



之前讲解并使用到了 Canvas 标签，但如果在 html 中没有显示的声明 Canvas 这个标签那要如何使用它？可以在代码里使用 document 对象的 createElement() 方法实现，并且为它设置默认的属性，再通过 appendChild() 方法添加到 html 页面中，以下是代码的实现如图所示：

```

//获得背景标签对象
core.container = document.getElementById("background");
//获得该对象的宽度和高度，通过 parseInt()转换为整数
var containerWidth = parseInt(core.getStyle(core.container, 'width'));
var containerHeight = parseInt(core.getStyle(core.container, 'height'));
//创建 Canvas 标签，添加到背景对象的节点下
core.canvas = core.createCanvas(core.container, containerWidth, containerHeight);
/**创建画布*/
createCanvas: function (parent, width, height) {
var canvas = {};
//该方法创建 Canvas 标签对象
canvas.node = document.createElement("canvas");
//设置它的默认属性
canvas.context = canvas.node.getContext("2d");
canvas.node.width = width || 100;
canvas.node.height = height || 100;
//添加到父节点中
parent.appendChild(canvas.node);
return canvas;
},
/** * 获得对象样式 - 兼容方法* */

```

```

getStyle: function (object, attr) {
    if(object.currentStyle){
        return object.currentStyle[attr]; //IE 才可以用
    }else{
        return getComputedStyle(object, false)[attr];
    }
},

```

动态创建 Canvas 对象

现在蒙板效果已经实现了，还剩下擦除蒙板的效果，这一功能同样是由 canvas 实现。还记得 canvas 有一个绘制线条和圆点的功能吗？擦除功能效果实际上就是绘制透明颜色的效果，本来 canvas 有一层灰色，现在通过绘制透明功能把灰色清除干净，代码如下所示：

```

//canvas 颜色- 灰色
ctx.fillStyle = "#696968";
ctx.fillRect(0, 0, containerWidth, containerHeight);
//自定义绘制圆点方法
ctx.fillCircle = function (x, y, radius, fillColor) {
    //设置为透明
    ctx.globalCompositeOperation = 'destination-out';
    ctx.fillStyle = fillColor;
    ctx.beginPath();
    ctx.moveTo(x, y);
    //设置圆心大小
    ctx.arc(x, y, radius, 0, Math.PI * 2, false);
    ctx.fill();
};

```

擦除蒙板代码示例

现在刮刮乐基本的功能（擦除效果，展示效果）已经完成，接下来讲解如何设计一个随机抽奖的算法（这个在前文的知识点简单进行了描述）。

1.4.2 任务 2 实现抽奖效果

如何设计一个公平的随机算法呢？假如总概率为 100，当一等奖中奖概率为 1%时，抽 100 次允许中一等奖 1 次该如何设计呢？根据之前章节的讲解中把每个奖品的概率设计为一个区间，再由随机数值大小决定奖品的归属。



把奖品当作一个类来设计，一、二、三等将都属性一个奖品类，该类有以下属性，下标，唯一 Id，名字，开启区间，结束区间等。代码如下所示：

```

//以下为 javascript 原型设计方法 /** 奖品对象 */
function Gift(index, id, name, startArea, endArea){

```

```

        this.index = index;
        this.id = id;
        this.name = name;
        this.startArea = startArea;
        this.endArea = endArea;
    }
    Gift.prototype.getIndex = function () {
        return this.index; };
    Gift.prototype.getGiftName = function () {
        return this.name; };
    Gift.prototype.getId = function () {
        return this.id; };
    Gift.prototype.getStartArea = function () {
        return this.startArea; };
    Gift.prototype.getEndArea = function () {
        return this.endArea; };
    Gift.prototype.toString = function () {
        return "Gift: " + "index=" + this.index
            + ", id=" + this.id
            + ", name=" + this.name
            + ", startArea=" + this.startArea
            + ", endArea=" + this.endArea;
    };
};

```

设置奖区

在程序初始化时创建 4 个对象，分别为一、二、三、四等奖，并把对象存放在数组里，减少频繁创建的操作，代码在前述知识点中已经描述。

通过 Math.random()方法创建一个随机数值，该随机数只能在 0-1 之间，所以要使它在 0-总概率之间浮动，就要乘以该总概率，可以得到[0-总概率]间浮动的一个随机数但它还是一个伪随机的数字，由 js 函数实现。如果想实现独特的真随机数，就应加入一些不可预测的数值，比如时间戳、键盘敲击的次数、CPU 运算速度、读写行为等。为了尽快看到效果，直接使用 JavaScript 提供的方法，代码如下所示：

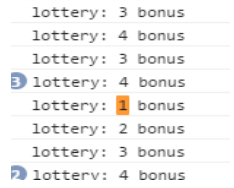
```


//比较随机数值 raw 和区间的大小
equalRaw: function (giftIterator, raw) {
    //判断下一个是否为 null
    if(giftIterator.hasNext()){
        var gift = giftIterator.next();
        //raw 数值在礼物的区间内？
        if(raw > gift.getStartArea() && raw < gift.getEndArea()){

```


<pre> switch (gift.getIndex()){ //对应的奖品，改变对应的图片 case 1: break; // 省略部分代码 default : break; } return; } core.equalRaw(giftIterator, raw); } }, </pre>	
抽奖算法	

算法设计完毕后，现在实际运行下，看下效果是否和要求的一样，在这里测试抽奖 100 次，看各个奖品的得奖次数，代码和 Console 工具所显示的中奖信息如下图所示：

<pre> var i = 0; while(i < 100){ var random = Math.random() * core.countProbability; core.equalRaw(core.iterator(core.listGifts), random); i++; } </pre>	
100 次抽奖测试	一等奖只出现一次

 算法测试完毕后即可具体应用在刮刮乐项目中。为了可以适应在不同的操纵系统中使用还有兼容注册触碰事件操作，通过浏览器的 userAgent 可以拿到用户类型这个数据，但是每个浏览器拿到的数据都不是一致的，它们的格式没有一个规定，写法完全不同，所以很难根据判断某个字符来确定它们的类型，如图所示：

Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/49.0.2623.75 Safari/537.36
Mozilla/5.0 (Linux; Android 5.0; SM-G900P Build/LRX21T) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/48.0.2564.23 Mobile Safari/537.36
Mozilla/5.0 (iPhone; CPU iPhone OS 9_1 like Mac OS X) AppleWebKit/601.1.46 (KHTML, like Gecko) Version/9.0 Mobile/13B143 Safari/601.1
设置不同终端类型后获取的 Agent 信息

 以上是 3 个不同类型的用户 PC，android，IOS，如何匹配字符串中各种字符组合的模式？这里我们通过正则表达式来做判断，需要分析每个字符串的结构和规律才可以写出对应的表达式，关于表达式这一块内容请参考对应的网络资源，以下正则表达式可试用，代码如图所示：

<pre> /** * 获得当前设备类型 */ getDeviceType: function () { </pre>

```

var userAgent = navigator.userAgent.toLocaleLowerCase();
if (userAgent.match(/MicroMessenger/i) == "micromessenger"){
    return DEVICE_CONFIG.WEIXING;
}else if (userAgent.match(/chrome\/([\d.]+)/) != null){
    return DEVICE_CONFIG.CHROME;
}else if (userAgent.match(/applewebkit\/([\d.]+)/) != null){
    return DEVICE_CONFIG.IOS;
}
return DEVICE_CONFIG.CHROME;
},

```

使用正则判断终端类型

刮刮乐只需要判断用户是 PC 端还是移动端的即可，在初始化中分别为判断后的用户类型，分别做注册触碰事件的操作，代码如下所示：

```

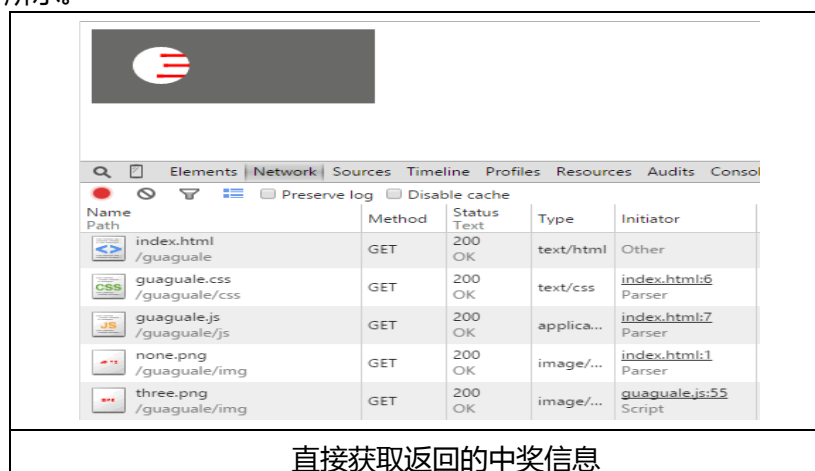
/** * 初始化注册事件* 有 2 种类型事件* PC/移动端 */
initEvent: function (type) {
    console.log(type);
    // xx == yy ? 1:0 如果 xx == yy 那么返回 1，否则返回 0
    core.canvas.node.addEventListener( type == DEVICE_CONFIG.CHROME ?
    "mousedown" : "touchstart", function (event) {
        core.canvas.isDrawing = true;
    });
    core.canvas.node.addEventListener( type == DEVICE_CONFIG.CHROME ?
    "mouseup" : "touchend", function (event) {
        core.canvas.isDrawing = false;
    });
    core.canvas.node.addEventListener( type == DEVICE_CONFIG.CHROME ?
    "mousemove" : "touchmove", function (event) {
        if (!core.canvas.isDrawing){
            return; }
        var x, y;
        //做判断是因为 PC 和移动的触碰参数不一致
        if (type == DEVICE_CONFIG.CHROME){
            x = event.pageX - this.offsetLeft;
            y = event.pageY - this.offsetTop;
        }else{
            x = event.touches[0].pageX - this.offsetLeft;
            y = event.touches[0].pageY - this.offsetTop;
        }
    }
}

```

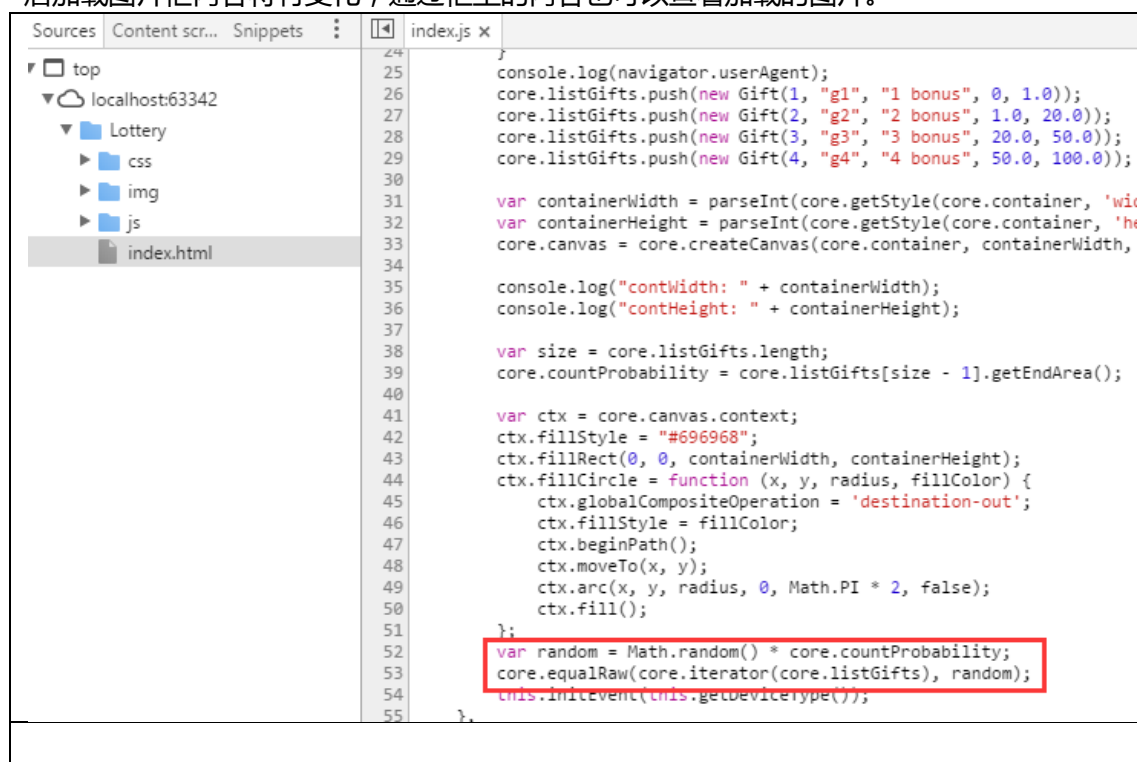
```
core.canvas.context.fillCircle(x, y, 20, "#CCCCFF");
});
},
```

根据终端类型不同确定坐标获取方式

到此整个的刮刮乐游戏基本功能已经实现，但是还存在不足，因为有的人会利用审查元素，在刮涂层之前就可以知道奖项，双击选择审查元素，点击 Network，刷新一下网页，看到了这里有两张图片（如果只有一张，则说明一定是谢谢惠顾奖），一看就知道这是几奖了，如图下所示。



因为代码是前端直接调用图片，所以用调试模式可以看到结果。如果图片是前端请求后台之后再加载上去的话，一样可以用审查元素操作。例如打开审查元素，点击 Network，选择 js 文件（有多个就分别选中查看），再点击右侧的 Preview 即可查看。如果是请求后台后加载图片框内容将有变化，通过框里的内容也可以查看加载的图片。



这个作弊利用了还没刮奖之前奖项图片就加载完毕的处理逻辑。因此应从处

理逻辑中解决，如奖项图片要等到用户刮奖之后再加载。首先把图上的获取奖项代码封装成函数，然后等到触发刮去涂层方法时才调用，代码如下所示。

```
if(raw > gift.getStartArea() && raw < gift.getEndArea()){
  switch (gift.getIndex()){
    case 1:
      core.container.style.background = "url(img/one.png)";
      break;
    case 2:
      core.container.style.background = "url(img/two.png)";
      break;
    case 3:
      core.container.style.background = "url(img/three.png)";
      break;
    case 4:
      core.container.style.background = "url(img/none.png)";
      break;
    default :
      break;
  }
  console.log("lottery: " + gift.getGiftName());
  return;
}
```

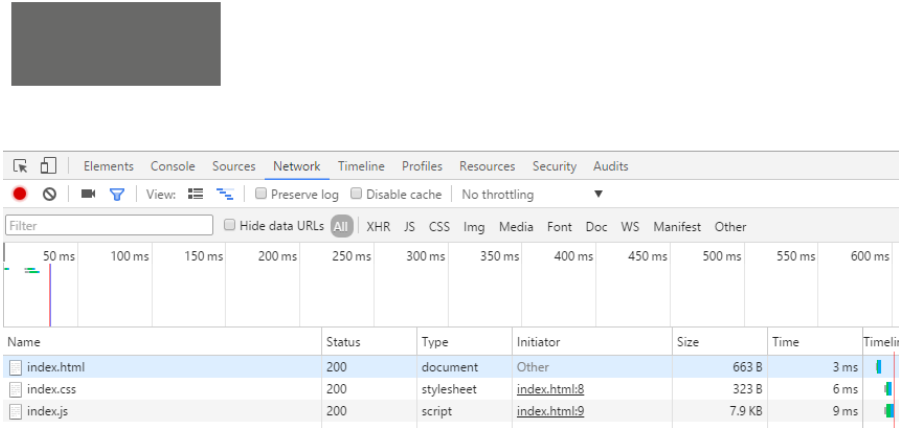
调用该函数。在图上所示代码的基础上修改即可，代码如下所示。

```
if(!core.canvas.isDrawing){
  return;
}
var x, y;
if(type == DEVICE_CONFIG.CHROME){
  x = event.pageX - this.offsetLeft;
  y = event.pageY - this.offsetTop;
}else{
  x = event.touches[0].pageX - this.offsetLeft;
  y = event.touches[0].pageY - this.offsetTop;
}
core.canvas.context.fillCircle(x, y, 20, "#CCCCFF");
if(core.tag){
  var random = Math.random() * core.countProbability;
```

```
core.equalRaw(core.iterator(core.listGifts), random);
core.tag = false;
}
```

修改触发抽奖的逻辑

现在测试一下，打开审查元素，点击 Network，不管刷新多少次，都只显示一张缺省图片，如图下所示。

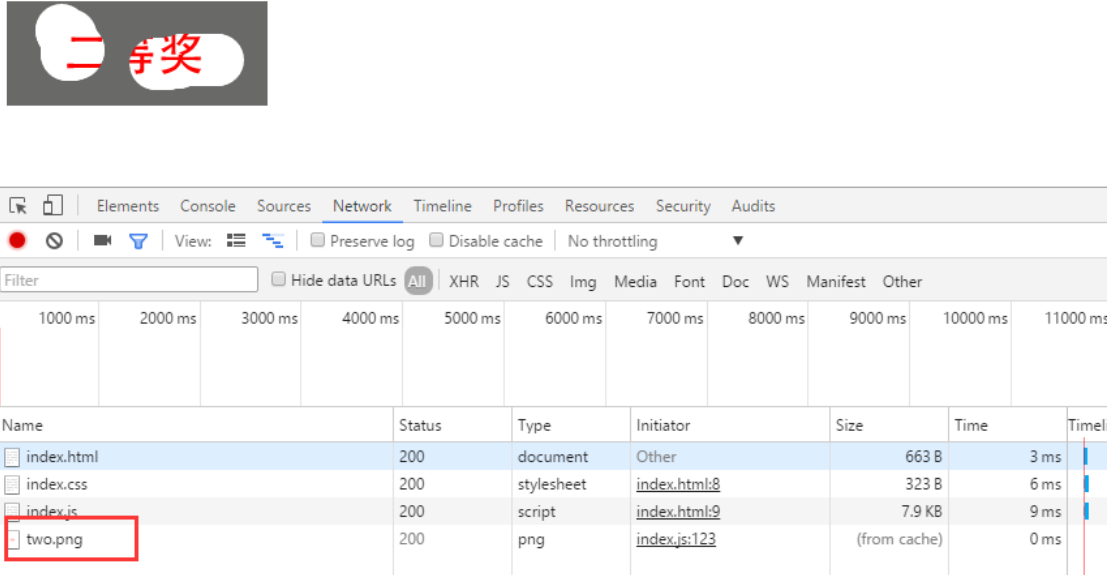


The screenshot shows the Network tab in a browser's developer tools. The timeline at the top shows a sequence of events. Below the timeline, a table lists the loaded resources:

Name	Status	Type	Initiator	Size	Time	Timelin
index.html	200	document	Other	663 B	3 ms	
index.css	200	stylesheet	index.html:8	323 B	6 ms	
index.js	200	script	index.html:9	7.9 KB	9 ms	

获奖图片不提前显示

当按下鼠标或划动手指刮奖时，控制台发生变化后触发了第二张图片，如图所示。




The screenshot shows the Network tab in a browser's developer tools. The timeline at the top shows a sequence of events. Below the timeline, a table lists the loaded resources:

Name	Status	Type	Initiator	Size	Time	Timelin
index.html	200	document	Other	663 B	3 ms	
index.css	200	stylesheet	index.html:8	323 B	6 ms	
index.js	200	script	index.html:9	7.9 KB	9 ms	
two.png	200	png	index.js:123	(from cache)	0 ms	

根据网络信息显示资源内容

1.4.3 任务 3 在线体验刮刮乐效果



前文提到了刮刮乐游戏不止可以在 PC 上运行也可以在其它终端上运行，通过模拟器测试解决了兼容问题，但实际的真机测试是无法完成的。如何让移动设备可以访问该游戏呢？应用前文讲到的 Nginx 服务器，把完成的刮刮乐项

目放进之前配置好的 Nginx 服务器即可，配置如下：


C:\Program Temp\DevelopmentTools\nginx-1.8.0\OUYANG

n Temp

opmentCo

opmentTo

roid



```
http{
....//略
server{
...//略
}

#设置访问静态资源目录配置
location ~* ^.+\.(\.ico|gif|jpg|jpeg|png|html|htm|css|js|xml|txt)$ {
    root OUYANG; //可已修改为自己认为合适的名称
    access_log off;
    expires 30d;
}
}
```

服务器配置代码和服务目录

启动 Nginx 服务器 输入访问地址 如测试输入的 `http://192.168.1.111/image2.jpg`，这次访问应该输入本机地址/项目名/文件名。本机地址是由 `ipconfig` 命令得到，如下：

C:\Windows\system32\cmd.exe

Microsoft Windows [版本 6.3.9600]
(c) 2013 Microsoft Corporation。保留所有权利。
C:\Users\ouyangshan09>ipconfig

Windows IP 配置

以太网适配器 以太网:

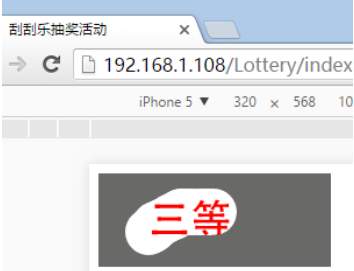
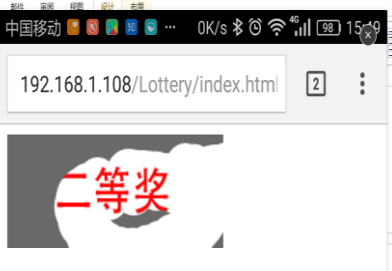
连接特定的 DNS 后缀 :
本地链接 IPv6 地址 : fe80::5a52:5a11:5c5:fe55x3
IPv4 地址 : 192.168.1.108
子网掩码 : 255.255.255.0
默认网关 : 192.168.1.1

以太网适配器 VirtualBox Host-Only Network:

连接特定的 DNS 后缀 :
本地链接 IPv6 地址 : fe80::80b6:e164:b64f:d9a5x8
自动配置 IPv4 地址 : 169.254.217.165
子网掩码 : 255.255.0.0
默认网关 :
以太网适配器 VirtualBox Host-Only Network #2:

Ipconfig 命令效果

例如最终的访问路径是 `http://192.168.1.108/Lottery/index.html`，回车即可看到刮刮乐的效果，如图所示：在测试机（华为某型号手机）效果如下：

	
模拟器效果	真机效果

1.5 单元小节

- 1、设计一个抽奖算法的后台接口（实现搭建一个服务器？）
- 2、在刮奖的那一刻有一点点小卡顿，如何解决？分析原因为什么卡顿？
- 3、实现刮奖完毕的特效？（可以参考下各大电商平台的效果）

1.6 单元练习

- 1、实现更多平台的 userAgent 分辨（本单元只有 iphone 和 Chrome 浏览器）
- 2、实现浮动中奖率的抽奖算法，如抽奖可分成高中低三个门槛，旁边应该滚动 XX 玩家的中奖消息以吸引其他玩家，运用设计模式更好的设计代码。