# Android

## OKhttp

# Android studio ide

## 报错：

在将一个Eclipse的项目转移到AndroidStudio的过程中，碰到了的问题如下：

Error:Execution failed for task ':learnChinese:mergeDebugResources'.

> Error: java.util.concurrent.ExecutionException:com.android.ide.common.process.ProcessException:

* 1
* 2

从报错来看可以定位到时资源文件出问题了，一个在Android 上运行好好的项目怎么会在AndroidStudio里面出现这种问题，顿觉头大了，查阅各种资料后来才发现了原来是**.9.png**图片造成的，真是作孽啊～

“点九”是andriod平台的应用软件开发里的一种特殊的图片形式，文件扩展名为：.9.png，也就是说项目资源文件夹中，只要图片的文件后缀是.9.png的，就说明这是点9图片，当我们用Eclipse等编译的时候，它遇到文件后缀是.9.png的文件，首先检查它实际上是不是点9图片，如果是，按点9图片处理;如果不是，按找默认图片处理。也就是说Eclipse自动帮我们完成了转化，容忍了我们的图片格式错误。

AndroidStudio使用的是android-maven-plugin插件，在aAndroidStudio下报错的原因，是因为声明为点9图片的文件，实际上不是真正的点9图片，解析图片出错。android-maven-plugin插件会严格检查图片格式，如果不匹配就报错。

**解决方法：**要么把图片制作成点9图片，要么去掉图片文件名中的.9。

## 下载

## 搭建环境

## 新建项目

## 设置主题

· 打开Setting/Preferences窗口（快捷键MacComman+,WindowsCtrl+Alt+S）

· 选择 Editor —>Colors & Fonts

· 看到3个主题，选择一个

## 设置字体颜色

## 设置快捷键

Ctrl+Alt+S 打开设置界面

Alt+回车 导入包,自动修正  
Ctrl+N   查找类  
Ctrl+Shift+N 查找文件  
Ctrl+Alt+L  格式化代码  
Ctrl+Alt+O 优化导入的类和包  
Alt+Insert 生成代码(如get,set方法,构造函数等)  
Ctrl+E或者Alt+Shift+C  最近更改的代码  
Ctrl+R 替换文本  
Ctrl+F 查找文本  
Ctrl+Shift+Space 自动补全代码  
Ctrl+空格 代码提示  
Ctrl+Alt+Space 类名或接口名提示  
Ctrl+P 方法参数提示  
Ctrl+Shift+Alt+N 查找类中的方法或变量  
Alt+Shift+C 对比最近修改的代码  
Shift+F6  重构-重命名  
Ctrl+Shift+先上键  
Ctrl+Y 删除行（ctrl+x不是删除行，是剪切。如果不选中，则为剪切当行。ths for 貌似掉线）  
Ctrl+D 复制行  
Ctrl+/ 或 Ctrl+Shift+/  注释（// 或者/\*...\*/ ）  
Ctrl+J  自动代码

Alt+ left/right 切换代码视图  
Alt+ Up/Down 在方法间快速移动定位  
Ctrl+Shift+Up/Down 代码向上/下移动。

Ctrl+H 显示类结构图  
Ctrl+Q 显示注释文档

1. Ctrl＋Shift＋F7可以高亮当前元素在当前文件中的使用
2. 1.Ctrl＋E，可以显示最近编辑的文件列表

20.Ctrl＋Shift＋Space在很多时候都能够给出Smart提示

21.Alt＋F3可以快速寻找  
22.Ctrl＋/和Ctrl＋Shift＋/可以注释代码  
23.Ctrl＋Alt＋B可以跳转到抽象方法的实现  
24.Ctrl＋O可以选择父类的方法进行重写  
Ctrl+E 最近打开的文件  
Alt+F1 查找代码所在位置  
Alt+1 快速打开或隐藏工程面板  
Ctrl+Alt+ left/right 返回至上次浏览的位置  
F2 或Shift+F2 高亮错误或警告快速定位  
代码标签输入完成后，按Tab，生成代码。  
选中文本，按Ctrl+Shift+F7 ，高亮显示所有该文本，按Esc高亮消失。  
Ctrl+W 选中代码，连续按会有其他效果  
选中文本，按Alt+F3 ，逐个往下查找相同文本，并高亮显示。  
Ctrl+Up/Down 光标跳转到第一行或最后一行下  
Ctrl+B 快速打开光标处的类或方法   
最常用快捷键  
2.Shift＋Click可以关闭文件  
3.Ctrl＋[或]可以跳到大括号的开头结尾  
4.Ctrl＋Shift＋Backspace可以跳转到上次编辑的地方  
5.Ctrl＋F12，可以显示当前文件的结构  
6.Ctrl＋F7可以查询当前元素在当前文件中的引用，然后按F3可以选择  
7.Ctrl＋N，可以快速打开类  
8.Ctrl＋Shift＋N，可以快速打开文件  
9.Alt＋Q可以看到当前方法的声明  
10.Ctrl＋W可以选择单词继而语句继而行继而函数  
11.Alt＋F1可以将正在编辑的元素在各个面板中定位  
12.Ctrl＋P，可以显示参数信息  
13.Ctrl＋Shift＋Insert可以选择剪贴板内容并插入  
14.Alt＋Insert可以生成构造器/Getter/Setter等  
15.Ctrl＋Alt＋V 可以引入变量。例如把括号内的SQL赋成一个变量  
16.Ctrl＋Alt＋T可以把代码包在一块内，例如try/catch  
17.Alt＋Up and Alt＋Down可在方法间快速移动

1. 在一些地方按Alt＋Enter可以得到一些Intention Action，例如将”==”改为”equals()”

下面的不是很有用  
19.Ctrl＋Shift＋Alt＋N可以快速打开符号  
25.Ctrl＋Q可以看JavaDoc  
26.Ctrl＋Alt＋Space是类名自动完成  
27.快速打开类/文件/符号时，可以使用通配符，也可以使用缩写  
28.Live Templates! Ctrl＋J  
30.Ctrl＋Alt＋Up /Ctrl＋Alt＋Down可以快速跳转搜索结果  
31.Ctrl＋Shift＋J可以整合两行  
32.Alt＋F8是计算变量值

## Git

验证帐号和密码

# Argularjs

## 定义

AngularJS 通过 **ng-directives** 扩展了 HTML。

**ng-app** 指令定义一个 AngularJS 应用程序。

**ng-model** 指令把元素值（比如输入域的值）绑定到应用程序。

ng-model 指令用于绑定应用程序数据到 HTML 控制器(input, select, textarea)的值。

**ng-model** 指令也可以：

* 为应用程序数据提供类型验证（number、email、required）。
* 为应用程序数据提供状态（invalid、dirty、touched、error）。
* 为 HTML 元素提供 CSS 类。
* 绑定 HTML 元素到 HTML 表单。

**ng-bind 指令把应用程序数据绑定到 HTML 视图。**

AngularJS 表达式写在双大括号内：**{{ expression }}**。

AngularJS 表达式把数据绑定到 HTML，这与 **ng-bind** 指令有异曲同工之妙。

AngularJS 将在表达式书写的位置"输出"数据。

## 框架案例

0//引入js文件

<script src="http://cdn.static.runoob.com/libs/angular.js/1.4.6/angular.min.js"></script>

1//定义angularjs模块Module

html:

<div ng-app="myapp">

</div>

js:

var app = angular.module('myapp',[]);

2//定义angularjs控制器Controller

html:

<div ng-app="myapp" ng-controller="mycontroller">

<input type="text" ng-model="ng-model-value" >

您输入的值是：{{ng-model-value}}//双向绑定

您输入的值是：<h1>{{ng-model-value}}</h1>//双向绑定

</div>

//js:**$scope** 对象当作一个参数传递

app.controller('mycontroller',function($scope){

$scope.ng-model-value = '123';

});

//绑定数据

//初始化数据的方式。

<div ng-app="" ng-init="quantity='2';cost=5,person={firstName:'John',lastName:'Doe'}，points=[1,15,19,2,40]">

<p>总价: <span ng-bind="quantity \* cost"></span> </p>

<p>总价: <span > {{" " +quantity+cost}} </span> </p>

<p>姓为 <span ng-bind="person.lastName"></span></p>

<p>第三个值为 {{ points[2] }}</p>

</div>

输出：10 25

## 循环html

<div data-ng-app="" data-ng-init="names=['Jani','Hege','Kai','aa']">

<p>使用 ng-repeat 来循环数组</p>

<ul>

<li data-ng-repeat="x in names">

{{ x }}

</li>

</ul>

</div>

对象数组

<div ng-app="" ng-init="names=[  
{name:'Jani',country:'Norway'},  
{name:'Hege',country:'Sweden'},  
{name:'Kai',country:'Denmark'}]">  
<p>循环对象：</p>  
<ul>  
  <li ng-repeat="x in names">  
    {{ x.name + ', ' + x.country }}  
  </li>  
</ul>  
</div>

## 自定义指令

使用 **.directive** 函数来添加自定义的指令。

你可以通过以下方式来调用指令：

* 元素名
* 属性
* 类名
* 注释

<body ng-app="myapp">

//元素名

<runoob-directive></runoob-directive>

//属性

<div runoob-directive></div>

//类名

<div class="runoob-directive"></div>

//注释

<!-- directive: runoob-directive -->

<runoob-directive2></runoob-directive2>

<script>

var app = angular.module("myapp",[]);

app.directive("runoobDirective",function(){

return {template:"<h1 style='color:#0f0;'>自定义指令</h1>"};

});

app.directive("runoobDirective2",function(){

return {template:"<h1>自定义指令</h1>"};

});

</script>

</body>

**restrict** 值可以是以下几种:

* E 作为元素名使用
* A 作为属性使用
* C 作为类名使用
* M 作为注释使用

在返回函数中

return {  
        restrict : "A",  
        template : "<h1>自定义指令!</h1>"  
 };

## 验证

必须在form表单里验证

Email:

<form ng-app="" name="myForm">

Email:

<input type="email" name="myAddress" ng-model="text">

<span ng-show="myForm.myAddress.$error.email">不是一个合法的邮箱地址</span>

</form>

## 应用状态

<form ng-app="" name="myForm" ng-init="myText = 'test@runoob.com'">

Email:

<input type="email" name="myAddress" ng-model="myText" required>

<p>编辑邮箱地址，查看状态的改变。</p>

<h1>状态</h1>

<p>Valid: {{myForm.myAddress.$valid}} (如果输入的值是合法的则为 true)。</p>

<p>Dirty: {{myForm.myAddress.$dirty}} (如果值改变则为 true)。</p>

<p>Touched: {{myForm.myAddress.$touched}} (如果通过触屏点击则为 true)。</p>

</form>

## **$scope**

//$scope的数组属性

<div ng-app="myApp" ng-controller="myCtrl">

<ul>

<li ng-repeat="x in names">{{x}}</li>

</ul>

</div>

<script>

var app = angular.module('myApp', []);

app.controller('myCtrl', function($scope) {

$scope.names = ["Emil", "Tobias", "Linus"];

});

</script>

## $rootScope

<div ng-app="myApp" ng-controller="myCtrl">

<ul>

<li ng-repeat="x in names">{{x}} {{lastname}}</li>

</ul>

<h1>姓氏为 {{lastname}} 家族成员:</h1>

</div>

<script>

var app = angular.module('myApp', []);

app.controller('myCtrl', function($scope, $rootScope) {

$scope.names = ["Emil", "Tobias", "Linus"];

$rootScope.lastname = "Refsnes";

});

</script>

<p>注意 $rootScope 在循环对象内外都可以访问。</p>

## ng-app

AngularJS 应用程序由 **ng-app** 定义。应用程序在 <div> 内运行。

**ng-controller="myCtrl"** 属性是一个 AngularJS 指令。用于定义一个控制器。

**myCtrl** 函数是一个 JavaScript 函数。

AngularJS 使用**$scope** 对象来调用控制器。

## 控制器方法

<div ng-app="myApp" ng-controller="personCtrl">

名: <input type="text" ng-model="firstName"><br>

姓: <input type="text" ng-model="lastName"><br>

<br>

姓名: {{fullName()}}

</div>

<script>

var app = angular.module('myApp', []);

app.controller('personCtrl', function($scope) {

$scope.firstName = "John";

$scope.lastName = "Doe";

$scope.fullName = function() {

return $scope.firstName + " " + $scope.lastName;

}

});

</script>

## 表达式中添加过滤器

<p>姓名为 {{ lastName | uppercase }}</p>

## $location 服务

var app = angular.module('myApp', []);  
app.controller('customersCtrl', function($scope, $location) {  
    $scope.myUrl = $location.absUrl();  
});

## $http 服务

//$http.get(url) 是用于读取服务器数据的函数

<div ng-app="myApp" ng-controller="myCtrl">

<h1>{{myWelcome}}</h1>

</div>

<p> $http 服务向服务器请求信息，返回的值放入变量 "myWelcome" 中。</p>

<script>

var app = angular.module('myApp', []);

app.controller('myCtrl', function($scope, $http) {

$http.get("welcome.htm").then(function (response) {

$scope.myWelcome = response.data;

});

});

</script>

//读取服务器的json数据

<div ng-app="myApp" ng-controller="siteCtrl">

<ul>

<li ng-repeat="x in names">

{{ x.Name + ', ' + x.Country }}

</li>

</ul>

</div>

<script>

var app = angular.module('myApp', []);

app.controller('siteCtrl', function($scope, $http) {

$http.get("http://www.runoob.com/try/angularjs/data/sites.php")

.success(function (response) {

//当从服务端载入 JSON 数据时，**$scope.names** 变为一个数组。

$scope.names = response.sites;

});

});

</script>

## $timeout 服务

<script>

var app = angular.module('myApp', []);

app.controller('myCtrl', function($scope, $timeout) {

$scope.myHeader = "Hello World!";

$timeout(function () {

$scope.myHeader = "How are you today?";

}, 2000);

});

</script>

## $interval 服务

<p>$interval 访问在指定的周期(以毫秒计)来调用函数或计算表达式。</p>

<script>

var app = angular.module('myApp', []);

app.controller('myCtrl', function($scope, $interval) {

$scope.theTime = new Date().toLocaleTimeString();

$interval(function () {

$scope.theTime = new Date().toLocaleTimeString();

}, 1000);

});

</script>

## 创建自定义服务

//'hexafy'服务名

app.service('hexafy', function() {  
    this.myFunc = function (x) {  
        return x.toString();  
    }  
});

## 在控制器中使用使用服务

app.controller('myCtrl', function($scope, **hexafy**) {  
    $scope.hex = **hexafy**.myFunc(255);  
});

## 在过滤器中使用服务

var app = angular.module('myApp', []);

app.service('hexafy', function() {

this.myFunc = function (x) {

return x.toString(16);

}

});

app.filter('myFormat',['hexafy', function(hexafy1) {

return function(x) {

return hexafy1.myFunc(x);

};

}]);

## Select(选择框)

使用数组或对象创建一个下拉列表选项。

## 使用 ng-options 创建选择框

<div ng-app="myApp" ng-controller="myCtrl">

<select ng-model="selectedName" ng-options="x for x in names">

</select>

</div>

<script>

var app = angular.module('myApp', []);

app.controller('myCtrl', function($scope) {

$scope.names = ["Google", "Runoob", "Taobao"];

$scope.selectedName = $scope.names[0];

});

</script>

<select ng-model="selectedSite" ng-options="x.site for x in sites">

</select>

<h1>你选择的是: {{selectedSite.site}}</h1>

<p>网址为: {{selectedSite.url}}</p>

</div>

<script>

var app = angular.module('myApp', []);

app.controller('myCtrl', function($scope) {

$scope.sites = [

{site : "Google", url : "http://www.google.com"},

{site : "Runoob", url : "http://www.runoob.com"},

{site : "Taobao", url : "http://www.taobao.com"}

];

});

</script>

//对象

<h1>你选择的是: {{selectedCar.brand}}</h1>

<h2>模型: {{selectedCar.model}}</h2>

<h3>颜色: {{selectedCar.color}}</h3>

<p>注意选中的值是一个对象。</p>

</div>

<script>

var app = angular.module('myApp', []);

app.controller('myCtrl', function($scope) {

$scope.cars = {

car01 : {brand : "Ford", model : "Mustang", color : "red"},

car02 : {brand : "Fiat", model : "500", color : "white"},

car03 : {brand : "Volvo", model : "XC90", color : "black"}

}

});

</script>

<select ng-model="selectedCar" ng-options="y.brand for (x, y) in cars"></select>

<p>你选择的是: {{selectedCar.brand}}</p>

<p>型号为: {{selectedCar.model}}</p>

<p>颜色为: {{selectedCar.color}}</p>

<p>下拉列表中的选项也可以是对象的属性。</p>

</div>

<script>

var app = angular.module('myApp', []);

app.controller('myCtrl', function($scope) {

$scope.cars = {

car01 : {brand : "Ford", model : "Mustang", color : "red"},

car02 : {brand : "Fiat", model : "500", color : "white"},

car03 : {brand : "Volvo", model : "XC90", color : "black"}

}

});

</script>

## 使用ng-repeat 指令来创建下拉列表

<div ng-app="myApp" ng-controller="myCtrl">

<select>

<option ng-repeat="x in names">{{x}}</option>

</select>

</div>

<script>

var app = angular.module('myApp', []);

app.controller('myCtrl', function($scope) {

$scope.names = ["Google", "Runoob", "Taobao"];

});

</script>

## 表格

<div ng-app="myApp" ng-controller="customersCtrl">

<table>

<tr ng-repeat="x in names">

<td>{{ x.Name }}</td>

<td>{{ x.Country }}</td>

</tr>

</table>

</div>

<script>

var app = angular.module('myApp', []);

app.controller('customersCtrl', function($scope, $http) {

$http.get("/try/angularjs/data/Customers\_JSON.php")

.success(function (response) {$scope.names = response.records;});

});

</script>

## 使用 CSS 样式

<style>  
table, th , td {  
  border: 1px solid grey;  
  border-collapse: collapse;  
  padding: 5px;  
}  
table tr:nth-child(odd) {  
  background-color: #f1f1f1;  
}  
table tr:nth-child(even) {  
  background-color: #ffffff;  
}  
</style>

## 显示序号 ($index)

 <td>{{ $index + 1 }}</td>

## ng-disabled 指令

**ng-disabled** 指令直接绑定应用程序数据到 HTML 的 disabled 属性

<div ng-app="" ng-init="mySwitch=false">

<p>

<input type="checkbox" ng-model="mySwitch"/>按钮

</p>

{{ mySwitch }}

</div>

# Mui

## 均匀布局

行布局

mui-row

总共12格 列布局

mui-col-xs-6

## 懒加载

## 分享

var shares = null,

bhref = false;

var Intent = null,

File = null,

Uri = null,

main = null;

var preUrl = window.location.href;

var shareImageUrl; // 分享的项目图片路径

// H5 plus事件处理

function plusReady() {

updateSerivces();

if(plus.os.name == "Android") {

Intent = plus.android.importClass("android.content.Intent");

File = plus.android.importClass("java.io.File");

Uri = plus.android.importClass("android.net.Uri");

main = plus.android.runtimeMainActivity();

}

}

if(window.plus) {

plusReady();

} else {

document.addEventListener("plusready", plusReady, false);

}

/\*\*

\* 更新分享服务

\*/

function updateSerivces() {

plus.share.getServices(function(s) {

shares = {};

for(var i in s) {

var t = s[i];

shares[t.id] = t;

}

}, function(e) {

//outSet("获取分享服务列表失败：" + e.message);

});

}

/\*\*

\* 分享操作

\* @param {String} id

\*/

function shareAction(id, ex) {

var s = null;

if(!id || !(s = shares[id])) {

outLine("无效的分享服务！");

return;

}

if(s.authenticated) {

//outLine("---已授权---");

shareMessage(s, ex);

} else {

//outLine("---未授权---");

s.authorize(function() {

shareMessage(s, ex);

}, function(e) {

outLine("认证授权失败");

});

}

}

/\*\*

\* 发送分享消息

\* @param {plus.share.ShareService} s

\*/

function shareMessage(s, ex) {

var msg = {

content: '发现美引导美，触手可及的美。这个项目我刚刚体验过了，很不错。',

extra: {

scene: ex

}

};

if(bhref) {

// msg.href = SHARE\_URL + 'share/project/' + merchantId + '/' + projectId + '.html?pt=xc&timestamp=' + new Date().getTime() + '&from=singlemessage&isappinstalled=1';

msg.href = 'http://test.ydmxcwx.ydm01.com/template/project/detail?merchantId=' + merchantId + '&projectId=' + projectId+ '&saleId=' + saleId +'&isAppShare=true&merchantname='+encodeURI(encodeURI(TaoMeiTitleName))+'&merchantIdall='+MERCHANTID\_C ;

console.log('分享路径=='+msg.href)

if(sharehrefTitle && sharehrefTitle.value != "") {

msg.title = sharehrefTitle.innerHTML;

}

if(sharehrefDes && sharehrefDes.value != "") {

msg.content = sharehrefDes.content;

}

msg.thumbs = ["http://img01.ydm01.com/images/162/20150622/20150622143820942.jpg"];

msg.pictures = ["http://img01.ydm01.com/images/162/20150622/20150622143820942.jpg"];

if(shareImageUrl) {

msg.thumbs = [shareImageUrl];

msg.pictures = [shareImageUrl];

}

} else {

if(pic && pic.realUrl) {

msg.pictures = [pic.realUrl];

}

}

s.send(msg, function() {

outLine("分享成功！ ");

}, function(e) {

// outLine("分享失败");

});

}

// 分析链接

function shareHref() {

bhref = true;

var ids = [{

id: "weixin",

ex: "WXSceneSession"

}, {

id: "weixin",

ex: "WXSceneTimeline"

}],

bts = [{

title: "发送给微信好友"

}, {

title: "分享到微信朋友圈"

}];

plus.nativeUI.actionSheet({

cancel: "取消",

buttons: bts

},

function(e) {

var i = e.index;

if(i > 0) {

shareAction(ids[i - 1].id, ids[i - 1].ex);

}

}

);

}

// 控制台输出日志

function outSet(msg) {

console.log(msg);

}

// 界面弹出吐司提示

function outLine(msg) {

mui.toast(msg);

}

## 预览图片

mui.previewimage.js

## 友盟统计

//开始统计

function startStatistic(pageid) {

mui.plusReady(function(){

plus.statistic.eventTrig(pageid);

plus.statistic.eventStart(pageid);

});

}

//结束统计

function endStatistic(pageid) {

mui.plusReady(function(){

plus.statistic.eventDuration(pageid);

plus.statistic.eventEnd(pageid);

});

}

## 图片上传

var successFunction;

var errorFunction;

function selectPictureType(successFC, errorFC) {

successFunction = successFC;

errorFunction = errorFC;

if (mui.os.plus) {

var a = [{

title: "拍照"

}, {

title: "从手机相册选择"

}];

plus.nativeUI.actionSheet({

title: "选择图片方式",

cancel: "取消",

buttons: a

}, function(b) {

switch (b.index) {

case 0:

break;

case 1:

getImage();

break;

case 2:

galleryImg();

break;

default:

break;

}

})

}

}

//拍照

function getImage() {

var c = plus.camera.getCamera();

c.captureImage(function(e) {

plus.io.resolveLocalFileSystemURL(e, function(entry) {

var s = entry.toLocalURL() + "?version=" + new Date().getTime();

uploadIMG

(s);

}, function(e) {

console.log("读取拍照文件错误：" + e.message);

});

}, function(s) {

console.log("error" + s);

}, {

filename: "\_doc/head.png"

})

}

/\*

\* 本地选择

\*/

function galleryImg() {

plus.gallery.pick(function(a) {

plus.io.resolveLocalFileSystemURL(a, function(entry) {

plus.io.resolveLocalFileSystemURL("\_doc/", function(root) {

root.getFile("head.png", {}, function(file) {

//文件已存在

file.remove(function() {

entry.copyTo(root, 'head.png', function(e) {

var e = e.fullPath + "?version=" + new Date().getTime();

uploadIMG(e);

//变更大图预览的src

//目前仅有一张图片，暂时如此处理，后续需要通过标准组件实现

},

function(e) {

console.log('copy image fail:' + e.message);

});

}, function() {

console.log("delete image fail:" + e.message);

});

}, function() {

//文件不存在

entry.copyTo(root, 'head.png', function(e) {

var path = e.fullPath + "?version=" + new Date().getTime();

uploadIMG(path);

},

function(e) {

console.log('copy image fail:' + e.message);

});

});

}, function(e) {

console.log("get \_www folder fail");

})

}, function(e) {

console.log("读取拍照文件错误：" + e.message);

});

}, function(a) {}, {

filter: "image"

})

};

function uploadIMG(filePath) {

if (mui.os.plus) {

plus.nativeUI.showWaiting();

}

lrz(filePath).then(function(rst) {

// 处理成功会执行

var imgData = rst.base64.replace("data:image/jpeg;base64,", "");

var URL\_UPLOAD\_IMG = IMAGE\_URL + 'mobileImageUpload.do';

mui.ajax(URL\_UPLOAD\_IMG, {

data: {

data: imgData,

filetype: 'jpg',

userid: CUSTOMERID,

cMerchantId: MERCHANTID

},

dataType: 'json',

type: 'post',

timeout: 10000,

success: successFunction,

error: errorFunction

});

}).catch(function(err) {

// 处理失败会执行

mui.toast('图片获取失败，请重试！');

if (mui.os.plus) {

plus.nativeUI.closeWaiting();

}

}).always(function() {

// 不管是成功失败，都会执行

});

}

function getBase64Image(img) {

var canvas = document.createElement("canvas");

var width = img.width;

var height = img.height;

var maxLength = 100;

if (width > height) {

if (width > maxLength) {

height = Math.round(height \*= maxLength / width);

width = maxLength;

}

} else {

if (height > maxLength) {

width = Math.round(width \*= maxLength / height);

height = maxLength;

}

}

canvas.width = width;

canvas.height = height;

var ctx = canvas.getContext("2d");

ctx.drawImage(img, 0, 0, width, height);

var dataURL = canvas.toDataURL("image/png", 0.8);

return dataURL.replace("data:image/png;base64,", "");

}

function uploadImage(filePath, successFunction, errorFunction) {

// 设置新的图片路径

var img = new Image();

img.src = filePath;

img.onload = function() {

// 获取新图片的字符串数据

var imgData = getBase64Image(img);

// 上传图片的路径

var URL\_UPLOAD\_IMG = IMAGE\_URL + 'mobileImageUpload.do';

//console.log(URL\_UPLOAD\_IMG + '?data=' + imgData + '&filetype=jpg&userid=' + CUSTOMERID + '&cMerchantId=' + MERCHANTID);

mui.ajax(URL\_UPLOAD\_IMG, {

data: {

data: imgData,

filetype: 'jpg',

userid: CUSTOMERID,

cMerchantId: MERCHANTID

},

dataType: 'json',

type: 'post',

timeout: 10000,

success: successFunction,

error: errorFunction

});

}

}

## 本地缓存

var MERCHANTID = "410"; // 商家id 所选分店id

var MERCHANTID\_C = "410"; // 商家id 管理中心

var TaoMeiTitleName = "梦圆皇宫"; // 商家名字

/\* 获取选择的分店id \*/

function getMERCHANTID() {

if (localStorage.getItem('MERCHANTIDKey')) {

try {

var datakey = JSON.parse(localStorage.getItem('MERCHANTIDKey'));

if (datakey[0].MERCHANTID\_C == MERCHANTID\_C) {

MERCHANTID = datakey[0].MERCHANTID;

if(datakey[0].TaoMeiTitleName){

TaoMeiTitleName=datakey[0].TaoMeiTitleName;

}

}

} catch (oException) {

//console.log('获取缓存错误，直接用原始值 MERCHANTID = ' + MERCHANTID);

}

} else {

//console.log('不存在缓存，直接用原始值 MERCHANTID = ' + MERCHANTID);

}

}

## 上拉刷新下拉加载封装

### 使用：

<div id="listSub" class="mui-content">

<ul class="mui-table-view muitableview hotproj" style="padding: 0px 10px 0px 10px;">

</ul>

</div>

mui.init();

var table = document.querySelector('.mui-table-view');

pullupAndDown.setTable(table, '0px');

pullupAndDown.setFunction(requestData);

if (mui.os.plus) {

mui.plusReady(function() {

requestData();

});

} else {

mui.ready(function() {

requestData();

});

}

function requestData() {

objectFunc.config.MERCHANTID\_C = MERCHANTID\_C;

objectFunc.config.pageIndex = pullupAndDown.pageIndex;

objectFunc.config.userId = CUSTOMERID;

ydm.getData('post', /\* 'myDistributorGoods.json' \*/ WEIXINURL+'/api/mall/myDistributorGoodsCustom', objectFunc);

};

var objectFunc = {

config: {

MERCHANTID\_C:MERCHANTID\_C,

userId:CUSTOMERID,

pageIndex:pullupAndDown.pageIndex

},

fnsuc: function(data) {

if(data.status == 100) {

var totalNum = data.totalNum;

pullupAndDown.totalPageNum = data.totalPageNum;

if(pullupAndDown.pageIndex == 1) {

table.innerHTML = '';

//无数据

TipUtils.noDataTips(data.data, 'listSub', '../images/no\_data\_tips.png');

}

moneyNumber = data.moneyNumber;

mui.each(data.data, function(i, item) {

var result = "";

var li = document.createElement('li');

li.className = 'mui-table-view-cell mui-media list';

result += '<a>';

result += '</a>';

li.innerHTML = result;

table.appendChild(li);

});

}

},

fnerror: function() {

//无网络 找到对应的图片路径

TipUtils.noNetworkTips('listSub', '../images/no\_network\_tips.png');

}

};

mui(table).on('tap', '.a-list', function() {

var merchantId = this.getAttribute("merchantId");

var projectId = this.getAttribute("projectId");

var PROJECTNAME = this.getAttribute("PROJECTNAME");

//跳转到分销人员列表

mui.openWindow({

url: 'custom\_my\_distbt\_user\_goods.html',

id: 'custom\_my\_distbt\_user\_goods.html',

extras: {

merchantId: merchantId,

projectId: projectId,

projectName: PROJECTNAME

},

createNew: false,

show: {

autoShow: true,

aniShow: 'slide-in-right',

duration: 100

},

waiting: {

autoShow: true,

title: '正在加载....'

}

});

});

var tipDiv; //提示层

//var tipContent = ''; //提示语

var TipUtils = {

//无数据提示：（数据被绑架了）

noDataTips: function(data,parentId,imgPath){

if(!imgPath){

imgPath = '../images/no\_data\_tips.png';

}

var hight = '100px';

if(!parentId){

hight = '50%';

}

if(!data || data.length == 0){

if(!tipDiv) {

tipDiv = document.createElement('div');

tipDiv.innerHTML = '<div style="width: 100%; text-align: center;">'

+'<img style="width: 30%; margin-top: '+hight+';" src="'+imgPath+'" />'

+'<p style="margin-top: 10px; font-size: 15px;">暂无数据</p></div>'

} else {

tipDiv.style.display = 'block';

}

if(parentId) {

parentDiv = document.getElementById(parentId);

} else {

parentDiv = document.body;

}

parentDiv.appendChild(tipDiv);

}else{

if(tipDiv) {

tipDiv.style.display = 'none';

}

}

},

//无网络提示:(网络不给力)

noNetworkTips: function(parentId,imgPath){

if(document.querySelector(".tip-info")) {

return;

}

if(!imgPath){

imgPath = '../images/no\_network\_tips.png';

}

var hight = '100px';

if(!parentId){

hight = '50%';

}

var tipDiv = document.createElement('div');

tipDiv.innerHTML = '<div class="tip-info" style="width: 100%; text-align: center;">'

+ '<img src="'+imgPath+'" style="width: 25%; margin-top: '+hight+';"/>'

+ '<p style="margin-top: 10px; font-size: 15px;">网络不给力哦</p></div>'

if(parentId) {

document.getElementById(parentId).innerHTML = '';

document.getElementById(parentId).appendChild(tipDiv);

} else {

document.body.appendChild(tipDiv);

}

}

}

var ydm = {

//获取数据

getData: function(type,url, dataJson) {

mui.ajax({

url: url,

type: type,

data: dataJson.config,

dataType: 'json',

timeout: 10000,

beforeSend: function() {

},

success: function(data) {

dataJson.fnsuc(data);

},

error: function(xhr, type, errorThrown) {

console.log(JSON.stringify(xhr) + type + "---" + errorThrown);

if(dataJson.fnerror){

dataJson.fnerror(xhr, type, errorThrown);

}

mui.toast(JSON.stringify(xhr) + type + "---" + errorThrown)

}

});

},

//tab切换

tab: function(obj, className, me) {

for(var i = 0; i < obj.length; i++) {

obj[i].classList.remove(className)

}

me.classList.add(className);

},

//获取url传值

GetQueryString: function(name) {

var reg = new RegExp("(^|&)" + name + "=([^&]\*)(&|$)");

var r = window.location.search.substr(1).match(reg);

if(r != null) return unescape(r[2]);

return null;

},

//loading

showWaiting: function() {

var div = document.createElement("div");

div.id = 'ydm\_loading';

div.innerHTML = '<div class="bg\_loading"></div>';

document.body.appendChild(div);

},

closeWaiting: function() {

var child = document.getElementById("ydm\_loading");

document.body.removeChild(child)

}

}

### 封装

(function(window) {

var pullupAndDown = {};

pullupAndDown.init = function() {

//最外层div

var div\_1 = document.createElement('div');

div\_1.className = "mui-scroll-wrapper";

div\_1.id = "pullrefresh";

//第二层div

var div\_2 = document.createElement('div');

div\_2.className = "mui-scroll";

div\_1.appendChild(div\_2);

var data; //请求参数

var getData; //请求方法

pullupAndDown.pageIndex = 1; //公共字段，统一页数为此字段

pullupAndDown.totalPageNum = pullupAndDown.pageIndex;//公共字段，统一页数为此字段

pullupAndDown.setTable = function(tab, height, nodeName, isPullDown) {

if(nodeName) {

var m\_div = document.body.querySelector(nodeName);

m\_div.appendChild(div\_1);

} else {

document.body.appendChild(div\_1);

}

if(isPullDown) {

mui.init({

pullRefresh: {

container: '#pullrefresh',

down: {

callback: pulldownRefresh

},

up: {

contentrefresh: '正在加载...',

callback: pullupRefresh

}

}

});

} else {

mui.init({

pullRefresh: {

container: '#pullrefresh',

down: {},

up: {

contentrefresh: '正在加载...',

callback: pullupRefresh

}

}

});

}

div\_2.appendChild(tab);

div\_1.style.marginTop = height;

}; //设置请求参数

pullupAndDown.setData = function(dat) {

data = dat;

}; //设置请求参数

pullupAndDown.setFunction = function(fuc) {

getData = fuc;

}; //设置请求方法

pullupAndDown.refresh = function() { //设置刷新方法

pullupAndDown.pageIndex = 1;

getData(data);

};

if(mui.os.plus) {

mui.plusReady(function() {

getData(data);

});

} else {

mui.ready(function() {

getData(data);

});

}

/\*\*

\* 下拉刷新具体业务实现

\*/

function pulldownRefresh() {

setTimeout(function() {

pullupAndDown.pageIndex = 1;

getData(data);

mui('#pullrefresh').pullRefresh().endPulldownToRefresh(); //refresh completed

mui('#pullrefresh').pullRefresh().refresh(pullupAndDown.pageIndex < pullupAndDown.totalPageNum);

}, 300);

}

/\*\*

\* 上拉加载具体业务实现

\*/

function pullupRefresh() {

setTimeout(function() {

mui('#pullrefresh').pullRefresh().endPullupToRefresh((++pullupAndDown.pageIndex >= pullupAndDown.totalPageNum));

if(pullupAndDown.pageIndex > pullupAndDown.totalPageNum) {

return;

}

console.log("pullupAndDown.pageIndex " + pullupAndDown.pageIndex)

console.log("pullupAndDown.totalPageNum " + pullupAndDown.totalPageNum)

getData(data);

}, 300);

}

}

window.pullupAndDown = pullupAndDown;

pullupAndDown.init();

})(this);

## 上拉加载下拉刷新注意：

## 父子页面：

mui.init({

subpages:[{

url:'merchant-lists.html',

id:'merchant-lists.html',

styles:{

top: '45px',

bottom: '0px'

}

}]

});

## [上拉加载时，怎么隐藏底部的“没有更多数据了”?](http://ask.dcloud.net.cn/question/5882" \t "http://dev.dcloud.net.cn/mui/pullup/_blank)

mui('#main').pullRefresh().endPullupToRefresh(true); // 没有更多内容

setTimeout(function(){

mui('#main').pullRefresh().disablePullupToRefresh();

setTimeout(function(){

mui('#main').pullRefresh().enablePullupToRefresh();

}, 1000); // 等待1s执行：恢复下拉加载更多功能（可根据需要，不恢复）}, 2000); // 显示“没有更多了”2s后执行：关闭向下加载更多功能（直接隐藏了“没有更多”）

## Demo

<div class="mui-content mui-scroll-wrapper muiscrollwrapper" id="pullrefresh">

<ul class="mui-table-view pullrefresh-card">

<li class="mui-table-view-cell mui-card mui-media">

</li>

</ul>

</div>

mui.init({

pullRefresh: {

container: '#pullrefresh',

down: {

callback: pulldownRefresh

},

up: {

contentrefresh: '正在加载...',

callback: pullupRefresh

}

}

});

var totalPageNum = 1;

var pageIndex = 1;

if(mui.os.plus) {

mui.plusReady(function() {

setTimeout(function() {

mui('#pullrefresh').pullRefresh().pulldownLoading();

}, 200);

});

} else {

mui.ready(function() {

mui('#pullrefresh').pullRefresh().pulldownLoading();

});

}

function pulldownRefresh() {

setTimeout(function() {

pageIndex = 1;

getDatas();

mui('#pullrefresh').pullRefresh().endPulldownToRefresh();

mui('#pullrefresh').pullRefresh().refresh(pageIndex<totalPageNum);

}, 200);

}

function pullupRefresh() {

setTimeout(function() {

pageIndex++;

getDatas();

mui('#pullrefresh').pullRefresh().endPullupToRefresh((pageIndex >= totalPageNum));

}, 200);

}

function getDatas() {

var datado = {

merchantid: merchantid,

userId: userId,

cardType: '0',

pageIndex: pageIndex

};

var request = 'queryvipcardbuylist';

var urldo = WEIXINURL + Suffix + request;

console.log(JSON.stringify(datado))

console.log(urldo)

mui.ajax(urldo, {

// mui.ajax('../../json/vipcardlistzhe.json', {

data: datado,

dataType: 'json',

type: 'post',

timeout: TimeOut,

beforeSend: function(data) {

//plus.nativeUI.showWaiting();

},

error: function(xhr, type, errorthrown) {

mui.toast('网络异常,请稍候再试');

//没网提示

},

success: function(data) {

plus.nativeUI.closeWaiting()

if(data.status != 100) {

mui.toast(data.msg);

return;

}

totalPageNum = data.totalPageNum;

var totalNum = data.totalNum;

var cardList = data.data;

if(pageIndex == 1) {

muitableview.innerHTML = "";

//

}

if(pageIndex > totalPageNum) {

//显示图片

return;

}

//没数据

//有数据

mui.each(cardList, function(i, item) {

var result = "";

var li = document.createElement('li');

li.className = 'mui-table-view-cell mui-media mui-card';

li.innerHTML = result;

muitableview.appendChild(li);

});

}

}

## [初始化](http://dev.dcloud.net.cn/mui/pullup/" \l "init)

mui.init({

pullRefresh : {

container:refreshContainer,//待刷新区域标识，querySelector能定位的css选择器均可，比如：id、.class等

up : {

height:50,//可选.默认50.触发上拉加载拖动距离

auto:true,//可选,默认false.自动上拉加载一次

contentrefresh : "正在加载...",//可选，正在加载状态时，上拉加载控件上显示的标题内容

contentnomore:'没有更多数据了',//可选，请求完毕若没有更多数据时显示的提醒内容；

callback :pullfresh-function //必选，刷新函数，根据具体业务来编写，比如通过ajax从服务器获取新数据；

}

}});

例子：

mui.init({

pullRefresh: {

container: '#pullrefresh',

down: {},

up: {

contentrefresh: '正在加载...',

callback: pullupRefresh

}

}

});

## 上拉刷新：

### [启用上拉刷新](http://dev.dcloud.net.cn/mui/pullup/" \l "enablePullupToRefresh)

mui('#pullup-container').pullRefresh().enablePullupToRefresh();

### [禁用上拉刷新](http://dev.dcloud.net.cn/mui/pullup/" \l "disablePullupToRefresh)

mui('#pullrefresh').pullRefresh().disablePullupToRefresh();

### [重置上拉加载](http://dev.dcloud.net.cn/mui/pullup/" \l "refresh)

mui('#pullrefresh').pullRefresh().refresh(true);

## 结束上拉：

//1、加载完新数据后，必须执行如下代码，true表示没有更多数据了：

//2、若为ajax请求，则需将如下代码放置在处理完ajax响应数据之后

this.endPullupToRefresh(true|false);

mui('#pullrefresh').pullRefresh().endPullupToRefresh(pageIndex >= totalPageNum);

mui('#pullrefresh').pullRefresh().endPullupToRefresh(true);

## 下拉加载：

## 下拉开始

mui('#pullrefresh').pullRefresh().pulldownLoading();

## 下拉结束：

//注意，加载完新数据后，必须执行如下代码，注意：若为ajax请求，则需将如下代码放置在处理完ajax响应数据之后

Ajax（）;s

mui('#refreshContainer').pullRefresh().endPulldownToRefresh();

## [更改下拉刷新文字位置](http://dev.dcloud.net.cn/mui/pulldown/" \l "changeposition)

.mui-bar-nav ~ .mui-content .mui-pull-top-pocket{

top: 180px !important;}

## [滚动到特定位置](http://dev.dcloud.net.cn/mui/pulldown/" \l "scrollTo)

双击标题栏，让列表快速回滚到顶部的功能

var contentWebview = null;//监听标题栏的双击事件document.querySelector('header').addEventListener('doubletap',function () {

if(contentWebview==null){

contentWebview = plus.webview.currentWebview().children()[0];

}

//内容区滚动到顶部

contentWebview.evalJS("mui('#pullrefresh').pullRefresh().scrollTo(0,0,100)");});

## Ajax请求

## 点击事件

//点击li时，执行foo\_1函数mui("#list").on("tap","li",foo\_1);

function foo\_1(){

console.log("foo\_1 execute");

}

## 自动触发按钮的点击事件

var btn = document.getElementById("submit");//监听点击事件btn.addEventListener("tap",function () {

console.log("tap event trigger");});//触发submit按钮的点击事件mui.trigger(btn,'tap');

## 触发自定义事件

webview必须触发loaded事件后，才能执行evalJS或mui.fire方法；

webview.addEventListener('loaded',function () {

webview.show('none', 1, function() {

console.log(1221);

webview.evalJS("test()");

mui.fire(webview, 'pageShow', {});

mui.openWindow({

id:'detail.html'

});

});})

## 监听自定义事件

window.addEventListener('customEvent',showPageB);

function showPageB(event){

//获取参数 event.detail.XXXXX;

plus.nativeUI.closeWaiting();

//这里是直接打开页面，也可以使用动画  
mui.currentWebview.show(); //然后B页面就直接出来了  
}

## 打开页面

mui.openWindow({

url: 'examples/info.html',

id:'info'

});

## 接收页面

//B页面onload从服务器获取列表数据；

window.onload = function(){

//从服务器获取数据

....

//业务数据获取完毕，并已插入当前页面DOM；

//注意：若为ajax请求，则需将如下代码放在处理完ajax响应数据之后；

mui.plusReady(function(){

//关闭等待框

plus.nativeUI.closeWaiting();

//显示当前页面

mui.currentWebview.show();

});}

## [关闭页面](http://dev.dcloud.net.cn/mui/window/" \l "closewindow)

## 关闭mui的back按键监听

mui.init({

keyEventBind: {

backbutton: false //关闭back按键监听

}});

## 点击包含.mui-action-back类的控件

若希望在顶部导航栏之外的其它区域添加关闭页面的控件，只需要在对应控件上添加.mui-action-back类即可

## 窗口关闭之前处理其它业务逻辑

var old\_back = mui.back;mui.back = function(){

var btn = ["确定","取消"];

mui.confirm('确认关闭当前窗口？','Hello MUI',btn,function(e){

if(e.index==0){

//执行mui封装好的窗口关闭逻辑；

old\_back();

}

});}

# 常用工具库

## //判断是否包含

function isContain(arr, obj) {

var i = arr.length;

while (i--) {

if (arr[i] === obj) {

return true;

}

}

return false;

}

## //根据逗号分割图片地址

function splitString(str) {

if (str.indexOf(",") > 0) {

var strs = new Array(); //定义一数组

strs = str.split(","); //字符分割

for (i = 0; i < strs.length; i++) {

//分割后的字符输出

return strs[0];

}

}

return str;

}

splitString(item.PICTURE\_URL)

# 验证

## /\*date日期转换成指定的格式\*/

var formatdate = function(time, format) {

var t = new Date(time);

var tf = function(i) {

return(i < 10 ? '0' : '') + i

};

return format.replace(/yyyy|MM|dd|HH|mm|ss/g, function(a) {

switch(a) {

case 'yyyy':

return tf(t.getFullYear());

break;

case 'MM':

return tf(t.getMonth() + 1);

break;

case 'mm':

return tf(t.getMinutes());

break;

case 'dd':

return tf(t.getDate());

break;

case 'HH':

return tf(t.getHours());

break;

case 'ss':

return tf(t.getSeconds());

break;

}

});

}

## /\*本地时钟\*/

function clockon(obj) {

var now = new Date();

var year = now.getFullYear(); //getFullYear getYear

var month = now.getMonth();

var date = now.getDate();

var day = now.getDay();

var hour = now.getHours();

var minu = now.getMinutes();

var sec = now.getSeconds();

var week;

month = month + 1;

if (month < 10) month = "0" + month;

if (date < 10) date = "0" + date;

if (hour < 10) hour = "0" + hour;

if (minu < 10) minu = "0" + minu;

if (sec < 10) sec = "0" + sec;

var arr\_week = new Array("星期日", "星期一", "星期二", "星期三", "星期四", "星期五", "星期六");

week = arr\_week[day];

var time = "";

time = year + "年" + month + "月" + date + "日" + " " + hour + ":" + minu + ":" + sec + " " + week;

if(obj==1){

return year;

}

if(obj==2){

return month;

}

if(obj==3){

return date;

}

if(obj==4){

return hour;

}

if(obj==5){

return minu;

}

return time;

}

## 时间戳转化

// 时间倒计时

function reduceTime() {

newNowDate = newNowDate + 1000;

//时间戳转成时间格式

console.log('newNowDate:'+newNowDate)

var newDate = new Date(newNowDate).Format('yyyy-MM-dd hh:mm:ss');

console.log('newDate:'+newDate)

var subDate = getDateDiff(endDate, newDate);

console.log('subDate:'+JSON.stringify(subDate))

document.getElementById('count\_hh').innerHTML = subDate.hh + subDate.dd\*24;

document.getElementById('count\_mm').innerHTML = subDate.mm; //(subDate.hh + subDate.dd\*24)\*60

document.getElementById('count\_ss').innerHTML = subDate.ss;

console.log(document.getElementById('count\_hh').innerHTML);

console.log(document.getElementById('count\_mm').innerHTML);

console.log(document.getElementById('count\_ss').innerHTML);

}

// 计算两个日期相差

function getDateDiff(startDate, endDate) {

var startTime = new Date(Date.parse(startDate.replace(/-/g, "/"))).getTime();

var endTime = new Date(Date.parse(endDate.replace(/-/g, "/"))).getTime();

// var startTime = stringTime(startDate);

// var endTime = stringTime(endDate);

console.log(startTime)

console.log(endTime)

var longTime = Math.abs((startTime - endTime));

console.log(longTime)

var subDate = {

dd: parseInt(longTime / (1000 \* 60 \* 60 \* 24)),

hh: parseInt(longTime / (1000 \* 60 \* 60)) % 24,

mm: parseInt(longTime / (1000 \* 60)) % 60,

ss: parseInt(longTime / 1000) % 60

}

console.log(JSON.stringify(subDate))

return subDate;

}

## /\*价格保留两位小数\*/

var NumberFixed = function(obj,nuber) {

/\*可能需要转换成数字类型\*/

// var obj=Number(obj);

return obj.toFixed(nuber);

}

## 手机号码：

//手机号码检查

function chkPhone(phone) {

var re = /^13[0-9]{9}$|14[0-9]{9}|15[0-9]{9}$|17[0-9]{9}$|18[0-9]{9}$/;

if (!(re.test(phone)) || phone == '') {

return false

}

return true;

}

if (chkPhone(phoneNum) == false) {

mui.toast("请输入正确的手机号码");

}

## //密码位数检查

function chkPwd(val) {

if(!(/^[\w.]{6,20}$/.test(val))) {

return false;

}

}

# 框架：

## 服务器环境：

var Environment = true; //true为服务器 正式环境

if(Environment) {

/\*\* 正式环境 \*\*/

SERVER\_URL = "http://xc.app.ydm01.com/ydmappweb/";

} else {

/\*测试服务器\*/

SERVER\_URL = "http://test.ydm01.com/ydmxc/";

}

var custerRegisterDo = "custerRegisterDo.do";

包括通用的常量

var aniShow = "pop-in";

var aniShowrightin = "slide-in-right";

var TimeOut = 10000;

# Jquery

## 监听input输入

<script src="http://apps.bdimg.com/libs/jquery/2.1.1/jquery.js" type="text/javascript" charset="utf-8"></script>

var phoneNum ;

$('input').bind('input propertychange', function() {

phoneNum = document.getElementById("phone").value;

if(phoneNum.length==11){

document.getElementById("vrcode").style.backgroundColor="#00D0D2"

}else{

document.getElementById("vrcode").style.backgroundColor="#EAEAEA"

}

//进行相关操作

});

## Input获取焦点失去焦点

document.onkeydown = function(event) {

var e = event || window.event || arguments.callee.caller.arguments[0];

if(e && e.keyCode == 13) {

//搜索，确认键

document.activeElement.blur();

}

}

var afoucs = document.getElementById('searchinput');

var btnheight = plus.screen.resolutionHeight;

var sty = 'top:' + (btnheight - 55 - plus.navigator.getStatusbarHeight()) + 'px;';

btnotherrecodes.setAttribute('style', sty);

afoucs.onfocus = function() {

mLog("onfocus11::" + document.body.clientHeight);

}

afoucs.onblur = function() {

mLog("onblur11::" + document.body.clientHeight);

}

## 时间倒计时

time(this);

//倒计时函数

var wait = 60;

function time(o) {

if (wait == 0) {

o.removeAttribute("disabled");

o.style.background = '#00D0D2'

o.innerText = "获取验证码";

wait = 60;

} else {

o.setAttribute("disabled", true);

o.innerText = "发送(" + wait + ")";

o.style.background = "#ddd"

wait--;

setTimeout(function() {

time(o)

},

1000)

}

}

## 点击图片切换

<img id="checkboximg" class="checkboxinput" src="../../images/zhuceshape.png" value="1"/>

var checkboximg=document.getElementById('checkboximg');

checkboximg.addEventListener('click', function() {

if(this.getAttribute('value') == 1){

this.src='../../images/zhuceshapekong.png';

this.setAttribute('value', '0');

}else{

this.setAttribute('value', '1');

this.src='../../images/zhuceshape.png';

}

});

# HTML5

## 添加属性样式

document.getElementById('praise').classList.add('mui-active');

document.getElementById('praise').classList.remove('mui-active');

## 三等份图片

.shouyemy .img3 {

overflow: hidden;

float: left;

width: 32.0%;

text-align: center;

}

.shouyemy .img3 img {

overflow: hidden;

width: 94%;

}

.shouyemy .img2 {

padding: 0px;

margin: 0px;

overflow: hidden;

float: left;

width: 50%;

text-align: center;

}

.shouyemy .img2 img {

padding: 0px;

margin: 0px;

overflow: hidden;

width: 98%;

}

.shouyemy .divimg {

background-color: white;

overflow: hidden;

height: auto;

width: 100%;

float: left;

padding: 0px 3px 2px 3px;

}

result += '<div class="divimg">';

var imgHeight = document.body.clientWidth / 3;

var picarray = item.noteContentImg.split(',');

result += '<div class="item img3" style="background-image: url('+picarray[0]+');height: '+imgHeight+'px;background-size: cover;margin: 0px 2px 0px 0px;" >';

//width: '+imgHeight+'px;

// result += '<img src="' + picarray[0] + '" />';

result += '</div>';

result += '<div class="item img3" style="background-image: url('+picarray[1]+');height: '+imgHeight+'px;background-size: cover;margin: 0px 2px 0px 2px;" >';

// result += '<img src="' + picarray[1] + '" />';

result += '</div>';

result += '<div class="item img3" style="background-image: url('+picarray[2]+');height: '+imgHeight+'px;background-size: cover;margin: 0px 0px 0px 2px;" >';

// result += '<img src="' + picarray[2] + '" />';

result += '</div>';

result += '</div>';

## 刷新界面

var userView = plus.webview.getWebviewById('template/user/index.html'); // 刷新个人中心界面

mui.fire(cartView, 'refresh');

或者

var webviewid = plus.webview.getWebviewById('orderPay');

if(webviewid){

mui.fire(webviewid, 'refersh', {

cardId: cardId,

cardType: cardType

});

}

//接收参数

Window.addEventListener('refersh', function(event) {

cardType = event.detail.cardId;

cardType = event.detail.cardType;

mui('#pullrefresh').pullRefresh().pulldownLoading();//下拉刷新

});

跳转页面：

## a标签跳转

javascript:void(0); 不聚焦，不跳转

单独跳转：href=""

var aniShow = "pop-in";

批量：在整个 .mui-content中，或者某个类中 的a标签跳转

mui('.swiper-banner').on('tap', 'a', function() {

var bannerId = this.getAttribute("bannerId");

var id = this.getAttribute('href');

var href = this.href;

mui.openWindow({

id: id,

url: this.href,

show: {

aniShow: aniShow

},

waiting: {

autoShow: false

},

extras: {

bannerId: bannerId

}

});

});

## openWindow跳转

（再次打开，不会刷新，所以先判断是否存在，存在就刷新，再打开）:

mui.openWindow({

id: '页面名字',

url: '页面名字',

show: {

aniShow: 'pop-in'

},

waiting: {

autoShow: true

},

extras: {

parentId: '123'

}

});

接收参数：

mui.plusReady(function(){

var curwebview = plus.webview.currentWebview();

var parentId = curwebview.parentId;

});

延时：

setTimeout(function(){

},1000);/\*延时2秒后返回\*/

## 关闭app所有打开的界面

，保留自己需要的，包括子界面：

var views = plus.webview.all();

mui.each(views, function(index, view) {

//入口界面 id plus.webview.getLaunchWebview().id

if(view.id != plus.webview.getLaunchWebview().id && view.id != 'template/index/index.html' ) {

view.close();

}

}

## 根据逗号分割图片地址

function splitString(str) {

if(str.indexOf(",") > 0) {

var strs = new Array(); //定义一数组

strs = str.split(","); //字符分割

for(i = 0; i < strs.length; i++) {

//分割后的字符输出

return strs[i];

}

}

return str;

}

## 弹窗：

var btnArray = ['确定', '取消'];

mui.confirm('\n订单已经生成可以在订单管理查看\n', '', btnArray, function(e) {

if (e.index == 0) {

// 确定

} else {

//取消

mui.plusReady(function(){

plus.webview.currentWebview().close();

});

}

});

var btnArray = ['确定'];

mui.confirm('\n订单已经生成可以在订单管理查看\n', '', btnArray, function(e) {

if (e.index == 0) {

// 确定

} else {

//取消

}

});

弹窗编辑：

plus.nativeUI.prompt(title, function(e) {

var txt = (e.index == 0) ? e.value : defaultValue;

console.log('txt::'+txt)

if (txt != '') {

obj.innerHTML = txt;

// 获取对应的属性名

console.log('key::'+key);

if (key) {

// 更新个人资料

updateUserInfo(key, txt);

}

} else {

obj.innerHTML = defaultValue;

}

}, "", defaultValue, ["确定", "取消"]);

时间控件：

var dDate = new Date();

dDate.setFullYear(2014, 7, 16);

var minDate = new Date();

minDate.setFullYear(1970, 0, 1);

var maxDate = new Date();

maxDate.setFullYear(2016, 11, 31);

plus.nativeUI.pickDate(function(e) {

var d = e.date;

birthday.innerText = d.getFullYear() + "-" + (d.getMonth() + 1) + "-" + d.getDate();

// 更新对应的用户资料

updateUserInfo('birthday', birthday.innerHTML);

}, function(e) {

birthday.innerText = "生日";

}, {

title: "请选择日期",

date: dDate,

minDate: minDate,

maxDate: maxDate

});

性别控件：

plus.nativeUI.actionSheet({

title: "性别",

cancel: "取消",

buttons: [{

title: "男"

}, {

title: "女"

}]

}, function(e) {

var index = e.index;

switch (index) {

case 1:

sexvalue.innerHTML = '男';

break;

case 2:

sexvalue.innerHTML = '女';

break;

}

// 更新对应的用户资料

updateUserInfo('sex', index == 1 ? 1 : 0);

});

图片处理：

圆角：

圆形：

style="width: 66px; height: 66px; border-radius: 33px; -webkit-border-radius: 33px;"

居中裁剪：

居中缩略：

自适应：

高斯模糊：

.title\_bg {

width: 100%;

height: 250px;

background-image: no-repeat;

position: relative;

z-index: 0;

}

.title\_bg .title\_bg\_img {

width: 100%;

height: 250px;

-webkit-filter: blur(5px);

background-image: no-repeat;

filter: progid: DXImageTransform.Microsoft.Blur(PixelRadius=2, MakeShadow=false);

background-size: cover;

position: relative;

z-index: 0;

}

其他的控件用z-index: 2;

<div id="" class="title\_bg">

<img class="title\_bg\_img" id="top-bg-img" src="../../images/ydm\_default.jpg" />

</div>

字体颜色大小

color: #999; font-weight: normal;

color: #8f8f94;

font-weight: bold;

密码框：

<div class="mui-input-row mui-password">

<input type="password" class="mui-input-password" placeholder="输入消费码">

</div>

标题栏：

<header class="mui-bar mui-bar-nav">

<a class="mui-action-back mui-icon mui-icon-left-nav mui-pull-left"></a>

<h1 class="mui-title" id="title">商品</h1>

</header>

段落里面三等份：

p span{width: 33.3%; display: block; float: left;}

<p><span>范围：所有商品</span><span style="text-align: center;">折扣：8折</span><span style="text-align: right;">有效期：1年</span></p>

判断是否登录：

登录时保存用户id，如果为空，就为登录

if(CUSTOMERID == '') {

mui.plusReady(function() {

//把当前界面的webviewid传到登录界面，跳转到登录界面，点击返回关闭当前界面，登录成功后，根据传过来的id刷新

//plus.webview.getWebviewById(parentId).reload();

//如果没有传界面id，跳到首页

if(parentId != '') {

plus.webview.getWebviewById(parentId).reload();

} else {

var indexView = plus.webview.getLaunchWebview();

mui.fire(indexView, 'gohome');

}

plus.webview.getWebviewById('template/cart/index.html').reload(); // 刷新购物车界面

plus.webview.getWebviewById('template/user/index.html').reload(); // 刷新个人中心界面

plus.webview.currentWebview().close();//关闭自己

});

return;

}

返回事件：

mui.back = function() {

var btnArray = ['确定', '取消'];

mui.confirm('\n订单已经生成可以在订单管理查看\n', '', btnArray, function(e) {

if (e.index == 0) {

// 确定

} else {

//取消

mui.plusReady(function(){

plus.webview.currentWebview().close();

});

}

});

}

基本页面：

<meta name="viewport" content="width=device-width, initial-scale=1,maximum-scale=1, user-scalable=no">

<meta name="apple-mobile-web-app-capable" content="yes">

<meta name="apple-mobile-web-app-status-bar-style" content="black">

var showMenu = false;

//首页返回键处理

//处理逻辑：1秒内，连续两次按返回键，则退出应用；

var first = null;

mui.back = function() {

if(showMenu) {

closeMenu();

} else {

//首次按键，提示‘再按一次退出应用’

if(!first) {

first = new Date().getTime();

mui.toast('再按一次退出应用');

setTimeout(function() {

first = null;

}, 5000);

} else {

if(new Date().getTime() - first < 5000) {

plus.runtime.quit();

}

}

}

};

弹出选择商家

(function($, doc) {

$.init();

$.ready(function() {

var storedata = new $.PopPicker();

storedata.setData(arr);

showstorePicker.addEventListener('tap', function(event) {

//分店数据显示

storedata.show(function(items) {

storeresult.innerText = items[0].text;

merchantid = items[0].value;

});

}, false);

});

})(mui, document);

//触发对应的id的事件

mui.trigger('id', 'tap');

数组：

var arr = [];

for(var i = 0; i < data.data.length; i++) {

var arr2 = {

"value": data.data[i].ID,

"text": data.data[i].NAME

}

arr.push(arr2);

}

列表头部和内容在一个界面；

列表头部和内容不在一个界面；

图片上传：

selectpicture(

function(data){

if (data.status == 100) {

headImg.src = data.imgurl;

//更改用户头像

updateUserInfo('headurl', data.imgurl);

} else {

headImg.src = defaultPath;//使用本地默认图片

}

if (mui.os.plus) {//关闭等待

plus.nativeUI.closeWaiting();

}

},

function(xhr,type,errorthr){

}

);

var successFunction;

var errorFunction;

function selectpicture(successdata,errordata){

successFunction = successdata;

errorFunction = errordata;

if(mui.os.plus){

var array = [{title:"拍照"},{title:"从手机相册选择"}];

plus.nativeUI.actionSheet(

{title:"选择图片方式",cancel:"取消",buttons:array}

,

function(datas){

switch(datas.index){

case 0:break;

case 1:

getImage();

break;

case 2:

galleryImage();

break;

default:

break;

}

});

}

}

//拍照

function getImage() {

var camera = plus.camera.getCamera();

camera.captureImage(

function(event){

plus.io.resolveLocalFileSystemURL(

event,

function(entry){

var vfilename = entry.toLocalURL()+"?version="+new Date().getTime();

uploadImg(vfileName);

},

function(e){

console.log(e.message);

});

},function(err) {

console,log(err);

},{

filename:"\_doc/head.png"

}

);

}

function uploadImg(vfilename) {

if(mui.os.plus){

plus.nativeUI.showWaiting();

}

//图片上传控件

//需要导入lrz.all.bundle.js lrz.all.bundle.js.map lrz.bundle.js lrz.bundle.js.map

lrz(filePath).then(function(rst) {

// 处理成功会执行

var imgData = rst.base64.replace("data:image/jpeg;base64,", "");

var URL\_UPLOAD\_IMG = IMAGE\_URL + 'mobileImageUpload.do';

mui.ajax(URL\_UPLOAD\_IMG, {

data: {

data: imgData,

filetype: 'jpg',

userid: CUSTOMERID,

cMerchantId: MERCHANTID

},

dataType: 'json',

type: 'post',

timeout: 10000,

success: successFunction,

error: errorFunction

});

}).catch(function(err) {

// 处理失败会执行

mui.toast('图片获取失败，请重试！');

if (mui.os.plus) {

plus.nativeUI.closeWaiting();

}

}).always(function() {

// 不管是成功失败，都会执行

});

}

//选择本地照片

function galleryImage(){

plus.gallery.pick(function(filepath) {

plus.io.resolveLocalFileSystemURL(filepath, function(entry) {

plus.io.resolveLocalFileSystemURL("\_doc/", function(root) {

root.getFile("head.png", {}, function(file) {

//文件已存在

file.remove(function() {

entry.copyTo(root, 'head.png', function(e) {

var e = e.fullPath + "?version=" + new Date().getTime();

uploadImg(e);

//变更大图预览的src

//目前仅有一张图片，暂时如此处理，后续需要通过标准组件实现

},

function(e) {

console.log('copy image fail:' + e.message);

});

}, function() {

console.log("delete image fail:" + e.message);

});

}, function() {

//文件不存在

entry.copyTo(root, 'head.png', function(e) {

var path = e.fullPath + "?version=" + new Date().getTime();

uploadImg(path);

},

function(e) {

console.log('copy image fail:' + e.message);

});

});

}, function(e) {

console.log("get \_www folder fail");

})

}, function(e) {

console.log("读取拍照文件错误：" + e.message);

});

}, function(a) {}, {

filter: "image"

});

｝

分享：

var shares = null,

bhref = false;

var Intent = null,

File = null,

Uri = null,

main = null;

var preUrl = window.location.href;

var shareImageUrl; // 分享的项目图片路径

// H5 plus事件处理

function plusReady() {

updateSerivces();

if(plus.os.name == "Android") {

Intent = plus.android.importClass("android.content.Intent");

File = plus.android.importClass("java.io.File");

Uri = plus.android.importClass("android.net.Uri");

main = plus.android.runtimeMainActivity();

}

}

if(window.plus) {

plusReady();

} else {

document.addEventListener("plusready", plusReady, false);

}

/\*\*

\* 更新分享服务

\*/

function updateSerivces() {

plus.share.getServices(function(s) {

shares = {};

for(var i in s) {

var t = s[i];

shares[t.id] = t;

}

}, function(e) {

//outSet("获取分享服务列表失败：" + e.message);

});

}

/\*\*

\* 分享操作

\* @param {String} id

\*/

function shareAction(id, ex) {

var s = null;

if(!id || !(s = shares[id])) {

outLine("无效的分享服务！");

return;

}

if(s.authenticated) {

//outLine("---已授权---");

shareMessage(s, ex);

} else {

//outLine("---未授权---");

s.authorize(function() {

shareMessage(s, ex);

}, function(e) {

outLine("认证授权失败");

});

}

}

/\*\*

\* 发送分享消息

\* @param {plus.share.ShareService} s

\*/

function shareMessage(s, ex) {

var msg = {

content: '很不错。',

extra: {

scene: ex

}

};

if(bhref) {

msg.href = SHARE\_URL + 'share/project/' + merchantId + '/' + projectId + '.html?pt=xc&timestamp=' + new Date().getTime() + '&from=singlemessage&isappinstalled=1';

if(sharehrefTitle && sharehrefTitle.value != "") {

msg.title = sharehrefTitle.innerHTML;

}

if(sharehrefDes && sharehrefDes.value != "") {

msg.content = sharehrefDes.content;

}

msg.thumbs = ["http://img01.ydm01.com/images/162/20150622/20150622143820942.jpg"];

msg.pictures = ["http://img01.ydm01.com/images/162/20150622/20150622143820942.jpg"];

if(shareImageUrl) {

msg.thumbs = [shareImageUrl];

msg.pictures = [shareImageUrl];

}

} else {

if(pic && pic.realUrl) {

msg.pictures = [pic.realUrl];

}

}

s.send(msg, function() {

outLine("分享成功！ ");

}, function(e) {

// outLine("分享失败");

});

}

// 分析链接

//manifest.json中需要配置微信的资质，appid和appserect，还有QQ的id，这些需要去申请

function shareHref() {

bhref = true;

var ids = [{

id: "weixin",

ex: "WXSceneSession"

}, {

id: "weixin",

ex: "WXSceneTimeline"

}, {

id: "qq"

}],

bts = [{

title: "发送给微信好友"

}, {

title: "分享到微信朋友圈"

}, {

title: "分享到QQ"

}];

plus.nativeUI.actionSheet({

cancel: "取消",

buttons: bts

},

function(e) {

var i = e.index;

if(i > 0) {

shareAction(ids[i - 1].id, ids[i - 1].ex);

}

}

);

}

// 控制台输出日志

function outSet(msg) {

console.log(msg);

}

// 界面弹出吐司提示

function outLine(msg) {

mui.toast(msg);

}

第三方登录：

//微信登录

var auths = null;

// 监听plusready事件

document.addEventListener("plusready", function() {

// 扩展API加载完毕，现在可以正常调用扩展API

plus.oauth.getServices(function(services) {

auths = services;

}, function(e) {

alert("获取分享服务列表失败：" + e.message + " - " + e.code);

});

}, false);

function authLogin() {

var s;

for (var i = 0; i < auths.length; i++) {

if (auths[i].id == 'weixin') {

s = auths[i];

break;

}

}

if (!s.authResult) {

s.login(function(e) {

if (mui.os.plus) {

plus.nativeUI.closeWaiting();

}

mui.toast("登录认证成功！");

authUserInfo();

}, function(e) {

if (mui.os.plus) {

plus.nativeUI.closeWaiting();

}

mui.toast("登录认证失败！");

});

} else {

if (mui.os.plus) {

plus.nativeUI.closeWaiting();

}

mui.toast("已经登录认证！");

}

}

//注销

function authLogout() {

for (var i in auths) {

var s = auths[i];

if (s.authResult) {

s.logout(function(e) {

console.log("注销登录认证成功！");

}, function(e) {

console.log("注销登录认证失败！");

});

}

}

}

// 微信登录认证信息

function authUserInfo() {

var s;

for (var i = 0; i < auths.length; i++) {

if (auths[i].id == 'weixin') {

s = auths[i];

break;

}

}

if (!s.authResult) {

mui.toast("未授权登录！");

} else {

s.getUserInfo(function(e) {

console.log("获取用户信息成功：" + JSON.stringify(s.userInfo));

wechatLogin(s.userInfo);//openid unionid

authLogout();

}, function(e) {

alert("获取用户信息失败：" + e.message + " - " + e.code);

});

}

}

//手机号码检查

function chkPhone(phone) {

var re = /^13[0-9]{9}$|14[0-9]{9}|15[0-9]{9}$|17[0-9]{9}$|18[0-9]{9}$/;

if (!(re.test(phone)) || phone == '') {

return false

}

return true;

}

/倒计时函数

var wait = 60;

//传入id

function time(o) {

if (wait == 0) {

o.removeAttribute("disabled");

o.style.background = '#00D0D2'

o.innerText = "获取验证码";

wait = 60;

} else {

o.setAttribute("disabled", true);

o.innerText = "发送(" + wait + ")";

o.style.background = "#ddd"

wait--;

setTimeout(function() {

time(o)

},

1000)

}

}

问题： 如何定位 什么（元素）文字在什么文字（控件）的右边，间距多少，块与块之间的定位 列表怎么用 div+css 图片圆形

.mui-content class="mui-content" id="navigator" #navigator\*/ \* 选择器 多个 \* .nav .line div \* .nav a.mui-active .icon-d \* .nav.mui-bar {

\* .table1 tbody tr:nth-child(2n) {

background: #fcf8f8;

}

关键词

选择器 伪类 背景： 文本 字体 链接 列表 表格 轮廓 内边距 边框 外边距 外边距合并 定位 对齐 margin 属性来水平对齐

元素 属性 和 值

/\*----------------------------------\*/

大小

height: auto !important; 自适应 加上!importanrt可以覆盖父级的样式

width: 25% ;或则 5px

line-height: 34px; 行高

/\*----------------------------------\*/

颜色

color: #999;

background-color:yellow //文本背景颜色

/\*----------------------------------\*/

背景

background: #f3f3f3;

background-color 设置元素的背景颜色。

background-image 把图像设置为背景。

background-position 设置背景图像的起始位置。

background-repeat 设置背景图像是否及如何重复。

background: url(../images/icon\_a1.png) no-repeat;

background-image: url(/i/eg\_bg\_04.gif);

background-position:center;

background-position:50% 50%;

background-attachment:fixed 背景关联 可视区是固定的

background-color 文本;

裁剪图片 position:absolute;

clip:rect(0px 50px 200px 0px);

background-size: cover; // 背景图片大小 把背景图像扩展至足够大，以使背景图像完全覆盖背景区域。

背景图像的某些部分也许无法显示在背景定位区域中。 repeat : 背景图像在纵向和横向上平铺 no-repeat : 背景图像不平铺 repeat-x : 背景图像在横向上平铺 repeat-y : 背景图像在纵向平铺;

background: #ff0000 url(/i/eg\_bg\_03.gif) no-repeat fixed left; //背景

/\*----------------------------------\*/

边距

margin-left:auto;

margin-right:auto;

margin-top: 10px; 距离顶部

right:0px;

padding: 10px 10px 0px 10px; //上下左右

/\*----------------------------------\*/

border-radius: 100px; 向 div 元素添加圆角边框：

-webkit-border-radius: 100px; 圆角效果

/\*----------------------------------\*/

文字对齐方式

text-align:center;

vertical-align: middle;

margin:auto

text-indent: 5em; 缩进文本

/\*----------------------------------\*/

边框

border:2px solid #a1a1a1; 边框线 单线 颜色

border-bottom: 1px #ddd solid; 底边框

border-collapse 设置是否把表格边框合并为单一的边框。

border-spacing 设置分隔单元格边框的距离。 c

aption-side 设置表格标题的位置。

empty-cells 设置是否显示表格中的空单元格。

table-layout 设置显示单元、行和列的算法。

border-width: 5px;

border-color: transparent;

可以按顺序设置如下属性：

border-width border-style border-color 是指css border-tyle

1.none:表示没 2.solid：表示单线 3.double：表示双线 4.groove：表示凹槽 5.redge:表示皱纹 6.dotted:点线(貌似IE6下效果等同于dashed) 7.dashed:虚线

medium是浏览器解析时自动算进去的样式，不影响你的设置，你大可不必纠结它

上下右左 border-color: #ff0000 #00ff00 #0000ff rgb(250, 0, 255);

样式： border-bottom-style none无边框 dotted 虚线 dashed虚大线 solid实线 列表展示类型 list-style-type: disc 实心 circle 空心 square 方实心 caption-side:bottom 标题位置 border-collapse:collapse;

去边框 outline:#00ff00 dotted thick;

/\*----------------------------------\*/

inherit是继承父类的属性，一般用于字体、颜色、背景等

auto是按情况自适应，一般用于高度、宽度、外边距和内边距等关于长度的属性

/\*----------------------------------\*/

位置

float: left; 右浮动 文字和图片相容

clear: both; 清除div

position:inherit; absolute绝对 固定fixed

-webkit-box-shadow: 0 0 0px rgba(0, 0, 0, .85); 左（负数）右（正数） 上（负数）下（ 正数） 是否模糊 后阴影的面积

box-shadow: 0 0 0px rgba(0, 0, 0, .85);

display: block ; 展示样式

z-index: 9999; 展示优先级 越大越高

overflow: hidden; 溢出 scroll;滑动条 //hidden 隐藏溢出的 auto 如何设置浏览器来自动地处理溢出

white-space: nowrap; nowrap规定段落中的文本不进行换行

text-overflow: clip; ellipsis省略号 clip裁剪不可见

width:12em; 最大多少个字符

定位的内容是： position:relative | absolute | static | fixed static(静态) 没有特别的设定，遵循基本的定位规定，不能通过z-index进行层次分级，这是默认值。 relative(相对定位) 对象不可层叠、不脱离文档流，参考自身静态位置通过 top,

bottom,

left,

right 定位，并且可以通过z-index进行层次分级。 absolute(绝对定位) 脱离文档流，通过 top,

bottom,

left,

right 定位。选取其最近一个最有定位设置的父级对象进行绝对定位，

如果对象的父级没有设置定位属性，absolute元素将以body坐标原点进行定位，

可以通过z-index进行层次分级。 fixed（固定定位） 这里所固定的参照对像是 可视窗口 而并非是body或是父级元素，其总是固定在浏览器窗口的某个位置，

并且不受滚动的影响，是绝对的坐标定位。可通过z-index进行层次分级。 注 ： CSS中定位的层叠分级： z-index: auto | namber;

auto 遵从其父对象的定位 namber 无单位的整数值。可为负数，默认值为0，越大越靠上，值大的元素会覆盖住值小的元素。 color: #e25f69;

/\*----------------------------------\*/

font-weight: bold;段落的字体的粗细 normal 正常 bold粗体

font-size: 14px; 字体大小

word-spacing: 30px;字间隔

color 设置文本颜色

direction 设置文本方向。

letter-spacing 设置字符间距。

text-decoration 向文本添加修饰。

text-shadow 设置文本阴影。

font-family 设置字体系列。

font-size 设置字体的尺寸。

font-size-adjust 当首选字体不可用时，对替换字体进行智能缩放。（CSS2.1 已删除该属性。）

font-stretch 对字体进行水平拉伸。（CSS2.1 已删除该属性。）

font-style 设置字体风格。

font-variant 以小型大写字体或者正常字体显示文本。

font-weight 设置字体的粗细。

/\*----------------------------------\*/

display: none;

none 此元素不会被显示。

block 此元素将显示为块级元素，此元素前后会带有换行符。

inline 默认。此元素会被显示为内联元素，元素前后没有换行符。

inline-block 行内块元素。（CSS2.1 新增的值）

list-item 此元素会作为列表显示。

run-in 此元素会根据上下文作为块级元素或内联元素显示。

compact CSS 中有值 compact，不过由于缺乏广泛支持，已经从 CSS2.1 中删除。

marker CSS 中有值 marker，不过由于缺乏广泛支持，已经从 CSS2.1 中删除。

table 此元素会作为块级表格来显示（类似 <table>），表格前后带有换行符。

inline-table 此元素会作为内联表格来显示（类似 <table>），表格前后没有换行符。

table-row-group 此元素会作为一个或多个行的分组来显示（类似 <tbody>）。

table-header-group 此元素会作为一个或多个行的分组来显示（类似 <thead>）。

table-footer-group 此元素会作为一个或多个行的分组来显示（类似 <tfoot>）。

table-row 此元素会作为一个表格行显示（类似 <tr>）。

table-column-group 此元素会作为一个或多个列的分组来显示（类似 <colgroup>）。

table-column 此元素会作为一个单元格列显示（类似 <col>）

table-cell 此元素会作为一个表格单元格显示（类似 <td> 和 <th>）

table-caption 此元素会作为一个表格标题显示（类似 <caption>）

inherit 规定应该从父元素继承 display 属性的值。

/\*----------------------------------\*/

a:link {color: #FF0000}

a:visited {color: #00FF00}

a:hover {color: #FF00FF}

a:active {color: #0000FF}

未访问的链接

已访问的链接

鼠标移动到链接上

选定的链接

首字母特效 p:first-letter {

color: #ff0000;

font-size: xx-large

}

图片垂直排列 vertical-align:text-top; //文字靠顶部 text-bottom 文字靠底部

CSS 相对定位 #box\_relative {

position: relative;

left: 30px;

top: 20px;

}

快捷键 ctrl + r 运行 重命名对象 ctrl+ F2;

ctrl+shift +f 格式化 搜索框 ctrl+h 查找文件 ctrl+T

/\* http://www.dreamdu.com/css/ http://www.runoob.com/css3/css3-backgrounds.html \*/

/\*

view.getViewTreeObserver().addOnGlobalLayoutListener(new OnGlobalLayoutListener() {

@Override

public void onGlobalLayout() {

showPop(); 自动打开加载popwindows

}

});

\* \*/

/\*

http://ask.dcloud.net.cn/article/199?item\_id=803

版本差异化更新

App资源在线差量升级更新

\* \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*CSS边框\*/

p#fourborders

{

border-width:thin medium thick 10px;

}

/\*

上边框是细边框

右边框是中等边框

下边框是粗边框

左边框是 10px 宽的边框

\* \*/

p#top

{

border-top-width:thin;

}

/\*

border-width -- 定义四个边框的宽度

取值:<border-width>{1,4} | inherit

<border-width>{1,4}: 边框宽度(可以有一个,两个,三个或四个值)

inherit:继承

引用网址:http://www.dreamdu.com/css/property\_border-width/

初始值: medium

继承性: 否

适用于: 所有元素

border:边框,width:宽度.

\* \*/

p#fourborders

{

border-color:red black blue green;

}

/\*如果定义四个值,那么这四个值就分别是上,右,下,左边框(从上边框开始,以顺时针的顺序遍历).定义梦之都段落的上边框为红色,右边框为黑色,下边框为蓝色,左边框为绿色.\*/

/\*

border-color -- 定义四个边框的颜色

取值:[ <color> | transparent ]{1,4} | inherit

<color>{1,4}: 颜色表示法

transparent{1,4}: 透明

inherit: 继承

引用网址:http://www.dreamdu.com/css/property\_border-color/

初始值: 同color属性的默认值

继承性: 否

适用于: 所有元素

border:边框,color:颜色

\* \*/

p#fourborders

{

border-style:dotted dashed solid double;

}

/\*

如果定义四个值,那么这四个值就分别是上,右,下,左边框(从上边框开始,以顺时针的顺序遍历).定义梦之都段落的上边框为点划线,右边框为虚线,下边框为实线,左边框为双线.

border-style -- 定义边框的样式

取值:<border-style>{1,4} | inherit

<border-style>{1,4}: 边框样式

inherit: 继承

引用网址:http://www.dreamdu.com/css/property\_border-style/

初始值: none

继承性: 否

适用于: 所有元素

border:边框,style:样式

none: 无样式

hidden: 除了同表格的边框发生冲突的时候,其它同none

dotted: 点划线

dashed: 虚线

solid: 实线

double: 双线,两条线加上中间的空白等于border-width的取值

groove: 槽状

ridge: 脊状,和groove相反

inset: 凹陷

outset:凸出,和inset相反

\* \*/

p#green

{

border:thin dotted green;

}

/\*

border -- 定义四个边的宽度,样式,颜色

取值:[ <border-width> || <border-style> || <border-color> ] | inherit

[ <border-width> || <border-style> || <border-color> ]: 边框宽度,样式,颜色属性中的一个或多个

inherit: 继承

初始值: none

继承性: 否

适用于: 所有元素

border: 边框

引用网址:http://www.dreamdu.com/css/property\_border/

\* \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*CSS边框\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*margin\*/

p#onemargins

{

margin:12px;

}

/\*

margin -- 定义边外补白

取值: <margin-width>{1,4} | inherit

<margin-width>{1,4}: 边外补白宽度

inherit: 继承

引用网址:http://www.dreamdu.com/css/property\_margin/

初始值: 0

继承性: 否

适用于: 所有元素,除了 table-caption, table 和 inline-table

margin，中文"边、页边"的意思

margin-width的取值

<length>: 长度表示法

<percentage>: 百分比表示法,margin百分比的计算是基于生成的框的包含块的宽度

auto: 自动

\* \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*margin\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*padding\*/

p#onepaddings

{

padding:12px;

}

/\*

padding -- 定义边内补白

取值: <padding-width>{1,4} | inherit

<padding-width>{1,4}: 边内补白宽度

inherit: 继承

引用网址:http://www.dreamdu.com/css/property\_padding/

初始值: 0

继承性: 否

适用于: 所有元素,,除了table-row-group, table-header-group, table-footer-group, table-row, table-column-group 和 table-column

padding，中文"填充"的意思

padding-width的取值

<length>: 长度表示法

<percentage>: 百分比表示法,padding百分比的计算是基于生成的框的包含块的宽度

auto: 自动

\* \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*padding\*/

/\*CSS3 圆角\*/

/\*

四个值: 第一个值为左上角，第二个值为右上角，第三个值为右下角，第四个值为左下角。

三个值: 第一个值为左上角, 第二个值为右上角和左下角，第三个值为右下角

两个值: 第一个值为左上角与右下角，第二个值为右上角与左下角

一个值： 四个圆角值相同

\* \*/

div

{

border:2px solid;

border-radius:25px;

}

/\*四个值 - border-radius: 15px 50px 30px 5px: 左上 右上 右下 左下\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*CSS3盒阴影\*/

div

{

width:300px;

height:100px;

background-color:yellow;

box-shadow: 10px 10px 5px #888888;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*CSS3边界图片\*/

/\*图像平铺\*/

#round

{

-webkit-border-image:url(border.png) 30 30 round; /\* Safari 5 and older \*/

-o-border-image:url(border.png) 30 30 round; /\* Opera \*/

border-image:url(border.png) 30 30 round;

}

/\*图像被拉伸\*/

#stretch

{

-webkit-border-image:url(border.png) 30 30 stretch; /\* Safari 5 and older \*/

-o-border-image:url(border.png) 30 30 stretch; /\* Opera \*/

border-image:url(border.png) 30 30 stretch;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*背景\*/

/\*CSS3 背景\*/

/\*CSS3 background-size 属性\*/

body

{

background:url(/try/demo\_source/img\_flwr.gif);

background-size:80px 60px;

background-repeat:inherit;

padding-top:40px;

}

body

{

background-image:url('html\_table.png');

}

/\*

background-image -- 定义背景图片

取值: <uri> | none | inherit

<uri>: URI

none: 无

inherit: 继承

引用网址:http://www.dreamdu.com/css/property\_background-image/

初始值: none

继承性: 否

适用于: 所有元素

background:背景.image:图片.

\* \*/

body

{

background-image:url('list-orange.png');

background-repeat:no-repeat;

}

/\*no-repeat 不重置（显示一张图片）repeat 重置（多张填满背景） \*/

/\*

\* background-repeat

取值: repeat | repeat-x | repeat-y | no-repeat | inherit

repeat: 平铺整个页面,左右与上下

repeat-x: 在x轴上平铺,左右

repeat-y: 在y轴上平铺,上下

no-repeat: 图片不重复

inherit: 继承

引用网址:http://www.dreamdu.com/css/property\_background-repeat/

初始值: repeat

继承性: 否

适用于: 所有元素

background:背景.repeat:重复.

\* \*/

body

{

background-color:green;

}

/\*

background-color -- 背景色,定义背景的颜色

取值: <color> | transparent | inherit

<color>: 颜色表示法

transparent: 透明

inherit: 继承

引用网址:http://www.dreamdu.com/css/property\_background-color/

初始值: transparent

继承性: 是

适用于: 所有元素

background:背景.color:颜色.

\* \*/

body

{

background-image:url('list-orange.png');

background-repeat:no-repeat;

}

/\*

background-position -- 定义背景图片的位置

取值: [ [ <percentage> | <length> | left | center | right ] [ <percentage> | <length> | top | center | bottom ]? ] | [ [ left | center | right ] || [ top | center | bottom ] ] | inherit

水平

left: 左

center: 中

right: 右

垂直

top: 上

center: 中

bottom: 下

垂直与水平的组合

x-% y-%

x-pos y-pos

inherit: 继承

引用网址:http://www.dreamdu.com/css/property\_background-position/

初始值: 0% 0%

继承性: 否

适用于: 所有元素

background:背景.position:位置.

\* \*/

body

{

background-image:url('list-orange.png');

background-attachment:fixed;

background-repeat:repeat-x;

background-position:center center;

}

/\*

background-attachment -- 定义背景图片随滚动轴的移动方式

取值: scroll | fixed | inherit

scroll: 随着页面的滚动轴背景图片将移动

fixed: 随着页面的滚动轴背景图片不会移动

inherit: 继承

引用网址:http://www.dreamdu.com/css/property\_background-attachment/

初始值: scroll

继承性: 否

适用于: 所有元素

background:背景.attachment:附着.

\* \*/

body

{

background:transparent url('list-orange.png') repeat-x scroll center center;

}

/\*

background -- 五个背景属性可以全部在此定义

取值: [<background-color> || <background-image> || <background-repeat> || <background-attachment> || <background-position>] | inherit

[<background-color> || <background-image> || <background-repeat> || <background-attachment> || <background-position>]: 背景颜色,图像等的一个属性或多个属性

inherit: 继承

初始值: 根据五个背景属性的默认值

继承性: 否

适用于: 所有元素

引用网址:http://www.dreamdu.com/css/property\_background/

background：中文"背景"的意思

\* \*/

div

{

background:url(img\_flwr.gif);

background-size:100% 100%;

background-repeat:no-repeat;

}

/\*规定背景的绘制区域。

\*/

div

{

background-color:yellow;

background-clip:content-box;

}

/\*background-clip: border-box 背景绘制在边框方框内（剪切成边框方框）。|padding-box 背景绘制在衬距方框内（剪切成衬距方框）。|content-box;背景绘制在内容方框内（剪切成内容方框）\*/

#div2

{

background-origin:content-box;

}

/\*CSS3的background-Origin属性\*/

/\*规定背景图片的定位区域。

padding-box 背景图像填充框的相对位置

border-box 背景图像边界框的相对位置

content-box 背景图像的相对位置的内容框

\* \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*背景\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*渐变（Gradients）\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*渐变\*\*/

#grad1 {

height: 500px;

background: -webkit-linear-gradient(red, blue); /\* Safari 5.1 - 6.0 \*/

background: -o-linear-gradient(red, blue); /\* Opera 11.1 - 12.0 \*/

background: -moz-linear-gradient(red, blue); /\* Firefox 3.6 - 15 \*/

background: linear-gradient(red, blue); /\* 标准的语法（必须放在最后） \*/

}

/\*线性渐变 - 从上到下

\* 线性渐变 - 从左到右

background: linear-gradient(to right, red , blue);

从左上角到右下角的线性渐变：

background: linear-gradient(to bottom right, red , blue);

半透明

background: linear-gradient(to right, rgba(255,0,0,0), rgba(255,0,0,0.5));

颜色结点均匀分布的径向渐变：

background: radial-gradient(red, green, blue);

\* \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*渐变\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*形状\*\*/

/\*

shape 参数定义了形状。它可以是值 circle 或 ellipse。其中，circle 表示圆形，ellipse 表示椭圆形。默认值是 ellipse。

\* \*/

#grad2 {

height: 150px;

width: 200px;

background: -webkit-radial-gradient(circle, red, yellow, green); /\* Safari 5.1 - 6.0 \*/

background: -o-radial-gradient(circle, red, yellow, green); /\* Opera 11.6 - 12.0 \*/

background: -moz-radial-gradient(circle, red, yellow, green); /\* Firefox 3.6 - 15 \*/

background: radial-gradient(circle, red, yellow, green); /\* 标准的语法（必须放在最后） \*/

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*形状\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*文本\*\*/

.ls3px

{

letter-spacing: 3px;

}

/\*

letter-spacing -- 定义文本中字母的间距(中文为文字的间距)

取值: normal | <length> | inherit

normal: 正常(主要是根据用户所使用的浏览器等设备)

<length>: 长度表示法

inherit: 继承

初始值: normal

继承性: 是

适用于: 所有元素

letter，中文"字母"， spacing，中文"间隔"

引用网址：http://www.dreamdu.com/css/property\_letter-spacing/

\* \*/

.ws30

{

word-spacing:30px;

}

/\*

word-spacing -- 定义以空格间隔文字的间距(就是空格本身的宽度)

取值: normal | <length> | inherit

normal: 正常

<length>: 长度表示法

inherit: 继承

初始值: normal

继承性: 是

适用于: 所有元素

word:词.spacing:间隔.

引用网址:http://www.dreamdu.com/css/property\_word-spacing/

\*/

p#underline

{

text-decoration:underline;

}

/\*

text-decoration -- 定义文本是否有划线以及划线的方式

取值:none | [ underline || overline || line-through || blink ] | inherit

none: 定义正常显示的文本

[underline || overline || line-through || blink]: 四个值中的一个或多个

underline: 定义有下划线的文本

overline: 定义有上划线的文本

line-through: 定义直线穿过文本

blink: 定义闪烁的文本

inherit: 继承

引用网址:http://www.dreamdu.com/css/property\_text-decoration/

初始值: none

继承性: 否

适用于: 所有元素

text:文本.decoration:装饰

\* \*/

p#capitalize

{

text-transform:capitalize;

}

/\*

text-transform -- 定义文本的大小写状态,此属性对中文无意义

取值:capitalize | uppercase | lowercase | none | inherit

capitalize: 首字母大写

uppercase: 大写

lowercase: 小写

none: 正常无变化

inherit: 继承

引用网址：http://www.dreamdu.com/css/property\_text-transform/

初始值: none

继承性: 是

适用于: 所有元素

text:文本.transform:转换.

\* \*/

p#left

{

text-align:left;

}

/\*

text-align -- 定义文本的对齐方式

取值: left | right | center | justify | inherit

left: 左对齐

right: 右对齐

center: 居中

justify: 对齐每行的文字

inherit: 继承

初始值: 如果是ltr就为left,如果是rtl就为right

继承性: 是

适用于: 所有元素

text:文本.align:排列.

引用网址:http://www.dreamdu.com/css/property\_text-align/

\* \*/

p

{

text-indent:58%;

}

/\*

text-indent -- 定义文本首行的缩进(在首行文字之前插入指定的长度)

取值: <length> | <percentage> | inherit

<length>: 长度表示法

<percentage>: 百分比表示法

inherit: 继承

初始值: 0

继承性: 是

适用于: 所有元素

text:文本.indent:缩进.

引用网址:http://www.dreamdu.com/css/property\_text-indent/

\* \*/

p#pre

{

white-space:pre;

}

/\*

white-space -- 空格内元素的显示方式

取值: normal | pre | nowrap | pre-wrap | pre-line | inherit

normal: 正常无变化(默认处理方式.文本自动处理换行.假如抵达容器边界内容会转到下一行)

pre: 保持HTML源代码的空格与换行,等同与pre标签

nowrap: 强制文本在一行,除非遇到br换行标签

pre-wrap: 同pre属性,但是遇到超出容器范围的时候会自动换行

pre-line: 同pre属性,但是遇到连续空格会被看作一个空格

inherit: 继承

初始值: normal

继承性: 是

适用于: 所有元素

white:白色.space:间隔,距离.

引用网址:http://www.dreamdu.com/css/property\_white-space/

\* \*/

p#pre

{

white-space:pre;

}

/\*

white-space -- 空格内元素的显示方式

取值: normal | pre | nowrap | pre-wrap | pre-line | inherit

normal: 正常无变化(默认处理方式.文本自动处理换行.假如抵达容器边界内容会转到下一行)

pre: 保持HTML源代码的空格与换行,等同与pre标签

nowrap: 强制文本在一行,除非遇到br换行标签

pre-wrap: 同pre属性,但是遇到超出容器范围的时候会自动换行

pre-line: 同pre属性,但是遇到连续空格会被看作一个空格

inherit: 继承

初始值: normal

继承性: 是

适用于: 所有元素

white:白色.space:间隔,距离.

引用网址:http://www.dreamdu.com/css/property\_white-space/

\* \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*文本\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*前景色\*\*/

div

{

opacity: 0.5;

}

/\*

CSS opacity 属性

opacity -- 不透明度

取值: <alphavalue> | inherit

<alphavalue>: 透明度取值(取值范围[0.0,1.0])

inherit: 继承

初始值: 1(不透明)

继承性: 是

适用于: 所有元素

引用网址:http://www.dreamdu.com/css/property\_opacity/

CSS3提出opacity属性

\* \*/

#p1

{

color:red;

}

/\*

color -- 前景色定义文字的颜色

取值: <color> | inherit

<color>: 颜色表示法

inherit: 继承

初始值: 依赖于用户的设备

适用于: 所有元素

继承性: 是

百分比值: 否(N/A)

媒介: 视觉(visual)

引用网址:http://www.dreamdu.com/css/property\_color/

\* \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 前景色\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*字体\*/

p

{

font-family:Arial, "宋体", Tahoma, "Times NEW Roman";

}

/\*

font-family -- 定义使用的字体

取值: [[ <family-name> | <generic-family> ] [, <family-name>| <generic-family>]\* ] | inherit

[[ <family-name> | <generic-family> ] [, <family-name>| <generic-family>]\* ]:字体名

family-name: 一系列字体名称

generic-family: 一般性字体名称,取值:serif,sans-serif,cursive,fantasy,monospace

inherit:继承

引用网址:http://www.dreamdu.com/css/property\_font-family/

初始值: 根据用户计算机的默认字体

继承性: 是

适用于: 所有元素

font：中文"字体"的意思.family：中文"家族"的意思.

\* \*/

p

{

font-size:12px;

}

/\*

font-size -- 定义字体的大小

取值: <absolute-size> | <relative-size> | <length> | <percentage> | inherit

<absolute-size>:绝对字体尺寸

取值：xx-small | x-small | small | medium | large | x-large | xx-large

xx-small:最小

x-small:较小

small:小

medium:正常(默认值)，根据字体进行调整

large:大

x-large:较大

xx-large:最大

<relative-size>:相对字体尺寸(相对与其父容器的字体尺寸)

取值：larger | smaller

larger:相对于父容器中字体尺寸进行相对增大，使用成比例的em作为单位

smaller:相对于父容器中字体尺寸进行相对减小，使用成比例的em作为单位

<length>:CSS长度表示法

<percentage>:CSS百分比表示法(相对与其父容器的字体尺寸)

inherit:继承

引用网址：http://www.dreamdu.com/css/property\_font-size/

初始值: medium(绝对字体尺寸中的一个取值)

继承性: 是

适用于: 所有元素

font，中文"字体"，size，中文"尺寸,大小"

small，中文"小"，medium，中文"中"，large，中文"大"

\* \*/

p#normal

{

font-style:normal;

}

/\*

font-style -- 定义字体显示的方式

取值: normal | italic | oblique | inherit

normal: 正常

italic: 斜体

oblique: 斜体

inherit: 继承

引用网址:http://www.dreamdu.com/css/property\_font-style/

初始值: normal

继承性: 是

适用于: 所有元素

font:字体.style:样式.

\* \*/

p#small-caps

{

font-variant:small-caps;

}

/\*

font-variant -- 定义小型的大写字母字体，对中文没什么意义

取值:normal | small-caps | inherit

normal:正常

small-caps:定义小型的大写字母

inherit:继承

引用网址：http://www.dreamdu.com/css/property\_font-variant/

初始值: normal

继承性: 是

适用于: 所有元素

font，中文"字体"的意思，variant，中文"不同的、变量"的意思

\* \*/

p#normal

{

font-weight:normal;

}

/\*

font-weight -- 定义字体的粗细

取值: normal | bold | bolder | lighter | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900 | inherit

normal:正常,等同于 400

bold:粗体,等同于 700

bolder:更粗

lighter:更细

100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900:字体粗细的绝对值

inherit:继承

引用网址:http://www.dreamdu.com/css/property\_font-weight/

初始值: normal

继承性: 是

适用于: 所有元素

font：中文"字体"的意思.weight：中文"重量,分量"的意思.

\* \*/

p

{

font:italic small-caps bold 12px/1.2em Arial;

}

/\*

font -- 定义字体的属性

取值:[ [ <font-style> || <font-variant> || <font-weight> ]? <font-size> [ / <line-height> ]? <font-family> ] | caption | icon | menu | message-box | small-caption | status-bar | inherit

[ [ <font-style> || <font-variant> || <font-weight> ]? <font-size> [ / <line-height> ]? <font-family> ]:字体取值

caption | icon | menu | message-box | small-caption | status-bar:定义系统字体

inherit:继承

引用网址:http://www.dreamdu.com/css/property\_font/

初始值: 根据字体其它属性的默认值

继承性: 是

适用于: 所有元素

font:字体

\* \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*字体\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*列表\*/

ul

{

list-style-type:disc;

}

/\*

list-style-type -- 定义列表样式

取值:disc | circle | square | decimal | decimal-leading-zero | lower-roman | upper-roman | lower-greek | lower-latin | upper-latin | armenian | georgian | lower-alpha | upper-alpha | none | inherit

disc: 点

circle: 圆圈

square: 正方形

decimal: 数字

decimal-leading-zero: 十进制数，不足两位的补齐前导0，例如: 01, 02, 03, ..., 98, 99

lower-roman: 小写罗马文字，例如: i, ii, iii, iv, v, ...

upper-roman: 大写罗马文字，例如: I, II, III, IV, V, ...

lower-greek: 小写希腊字母，例如: α(alpha), β(beta), γ(gamma), ...

lower-latin: 小写拉丁文，例如: a, b, c, ... z

upper-latin: 大写拉丁文，例如: A, B, C, ... Z

armenian: 亚美尼亚数字

georgian: 乔治亚数字，例如: an, ban, gan, ..., he, tan, in, in-an, ...

lower-alpha: 小写拉丁文，例如: a, b, c, ... z

upper-alpha: 大写拉丁文，例如: A, B, C, ... Z

none: 无(取消所有的list样式)

inherit:继承

引用网址:http://www.dreamdu.com/css/property\_list-style-type/

初始值: disc

继承性: 是

适用于: 所有属于list的元素

list，中文"列表"的意思.style，中文"样式"的意思.type，中文"类型"的意思.

\* \*/

ul

{

list-style-image:url("/images/list-orange.png");

}

/\*

list-style-image -- 定义列表样式图片

取值:<uri> | none | inherit

<uri>:URI(一般是一个图片的网址)

none:无

inherit:继承

初始值: none

继承性: 是

适用于: 所有属于list的元素

list:列表,style:样式,image:图片.

引用网址:http://www.dreamdu.com/css/property\_list-style-image/

\* \*/

ul#inside

{

list-style-position:inside;

list-style-image:url("/images/list-orange.png");

}

/\*

list-style-position -- 定义列表样式的位置

取值: inside | outside | inherit

inside: 内部

outside: 外部

inherit: 继承

初始值: outside

继承性: 是

适用于: 所有属于list的元素

list: 列表,style: 样式,position: 位置.

引用网址:http://www.dreamdu.com/css/property\_list-style-position/

\* \*/

ul#test1

{

list-style:square inside url("/images/list-orange.png");

}

/\*

list-style -- 定义列表样式

取值: [ <list-style-type> || <list-style-position> || <list-style-image> ] | inherit

[ <list-style-type> || <list-style-position> || <list-style-image> ]:list-style-type,list-style-position,list-style-image中的一个或多个

inherit: 继承

初始值: 根据list的其它三个属性

继承性: 是

适用于: 所有属于list的元素

list: 列表,style: 样式.

引用网址:http://www.dreamdu.com/css/property\_list-style/

\* \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*列表\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*单位\*/

/\*

\* CSS相对长度

em 无 依赖于最近的字体尺寸

ex 无 依赖于英文子母小x的高度

px pixels 像素，依赖于用户的显示设备

CSS绝对长度

in inches 英寸，英制单位，1英寸等于2.54厘米

cm centimeters 厘米，公制单位

mm millimeters 毫米，公制单位

pt points 点，1点等于1/72inches英寸，印刷设计中使用的单位

pc picas 铅字，1个pc等于12个point点，印刷设计中使用的单位

\* \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*单位\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*CSS伪元素\*/

p::first-line

{

color: red;

}

/\*

::first-line中的属性包括：font，color，background，word-spacing，letter-spacing，text-decoration，vertical-align，text-transform，line-height

::first-line伪元素只可以和块类元素（div、p）连用

::first-line -- CSS ::first-line 伪元素，匹配一个段落的第一行的样式

语法: E::first-line

CSS版本：CSS1

引用网址：http://www.dreamdu.com/css/pseudo-element\_first-line/

说明：

匹配一个段落的第一行的样式

first-line，中文"第一行"的意思

\* \*/

p::first-letter

{

font-size: 4em;

font-weight: bold;

border: 1px solid lightblue;

margin-right: 8px;

}

/\*

::first-letter -- CSS ::first-letter 伪元素，匹配第一个字母（中文是第一个文字）的样式

语法: E::first-letter

CSS版本：CSS1

引用网址：http://www.dreamdu.com/css/pseudo-element\_first-letter/

说明：

匹配第一个字母（中文是第一个文字）的样式

可以用于"词首大写"、"大写字母下沉"等样式，这都是常见的印刷效果

first-letter，中文"第一个字母"的意思

::first-letter伪元素只匹配块类元素的一部分

可以应用在::first-letter中的属性包括：font, text-decoration, text-transform, letter-spacing, word-spacing (仅当appropriate), line-height, float, vertical-align (仅当'float'为'none'时), margin, padding, border, color, background

\* \*/

div::before

{

background: lightgreen;

content: "梦之都";

}

/\*

::before -- CSS ::before 伪元素，定义在一个元素的内容之前插入content属性定义的内容与样式

语法: E::before

CSS版本：CSS2

引用网址：http://www.dreamdu.com/css/pseudo-element\_before/

说明：

定义在一个元素的内容之前插入content属性定义的内容与样式

before，中文"前"的意思

\* \*/

div::after

{

background: lightgreen;

content: "梦之都";

}

/\*

::after -- CSS ::after 伪元素，定义在一个元素的内容之后插入content属性定义的内容与样式

语法: E::after

CSS版本：CSS2

引用网址：http://www.dreamdu.com/css/pseudo-element\_after/

说明：

定义在一个元素的内容之后插入content属性定义的内容与样式

after，中文"后"的意思

\* \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*CSS伪元素\*/

/\*

/\*...\*/ -- 注释

azimuth

background-attachment -- 背景图片随滚动轴的移动方式

background-color -- 背景颜色

background-image -- 背景图像

background-position -- 背景图片的位置

background-repeat -- 背景图片的显示方式

background -- 统一背景

border -- 统一边框所有属性

border-bottom -- 统一下边框

border-bottom-color -- 下边框颜色

border-bottom-style -- 下边框样式

border-bottom-width -- 下边框宽度

border-break -- 内容分开的边框

border-collapse

border-color -- 边框颜色

border-left -- 统一左边框

border-image -- 边框图像

border-left-color -- 左边框颜色

border-left-style -- 左边框样式

border-left-width -- 左边框宽度

border-radius -- 圆角边框

border-right-color -- 右边框颜色

border-right-style -- 右边框样式

border-right-width -- 右边框宽度

border-right -- 统一右边框

border-spacing

border-style -- 边框样式

border-top -- 统一上边框

border-top-color -- 上边框颜色

border-top-style -- 上边框样式

border-top-width -- 上边框宽度

border-width -- 统一边框宽度

bottom -- 下偏移属性

caption-side

clear -- 清除

clip -- 了盒子可显示的区域

color -- 颜色

content -- 字符串

counter-increment

counter-reset

cue-after

cue-before

cue

cursor -- 鼠标样式

direction

display -- 显示方式

elevation

empty-cells

float -- 元素漂浮

font-family-- 字体

font-size -- 字体大小

font-style -- 字体样式

font-variant -- 小型的大写字母字体

font-weight -- 字体粗细

font -- 统一字体所有属性

height -- 盒子高度

left -- 左偏移属性

letter-spacing -- 文本中字母的间距(中文为文字的间距)

line-height -- 一段文字中每行文字的间距

list-style-image -- 列表图像

list-style-position -- 列表位置

list-style-type -- 列表类型

list-style -- 统一列表所有属性

margin-right -- 右边外补白宽度

margin-left -- 左边外补白宽度

margin-top -- 上边外补白宽度

margin-bottom -- 下边外补白宽度

margin -- 统一边外补白

max-height -- 最大高度

max-width -- 最大宽度

min-height -- 最小高度

min-width -- 最小宽度

opacity -- 元素的透明度

orphans

outline-color

outline-style

outline-width

outline

overflow -- 盒子中内容的显示方式

padding-top -- 上边内补白

padding-right -- 右边内补白

padding-bottom -- 下边内补白

padding-left -- 左边内补白

padding -- 统一边内补白

page-break-after

page-break-before

page-break-inside

pause-after

pause-before

pause

pitch-range

pitch

play-during

position -- 元素的定位

quotes

richness

right -- 右偏移属性

speak-header

speak-numeral

speak-punctuation

speak

speech-rate

stress

table-layout

text-align -- 文本对齐方式

text-decoration -- 文本是否有划线以及划线的方式

text-indent -- 文本的首行缩进

text-transform -- 文本的大小写状态,此属性对中文无意义

top -- 上偏移属性

unicode-bidi

vertical-align -- 垂直对齐方式

visibility -- 元素是否可见

voice-family

volume

white-space -- 文本的格式

widows

width -- 盒子宽度

word-spacing -- 以空格间隔文字的间距

z-index -- z轴顺序

\*

\* \*/

标签分组：

被分组的选择器就可以分享相同的声明。用逗号将需要分组的选择器分开

h1,h2,h3,h4,h5,h6 {

color: green;

}

上下文选择器

Li列表中的 strong 元素变为斜体字

li strong {

font-style: italic;

font-weight: normal;

}

而不是p中的strong

<p><strong>我是粗体字，不是斜体字，因为我不在列表当中，所以这个规则对我不起作用</strong></p>

包含选择器

从左向右读选择器，可以换成以下说法：“包含 em 的所有 h1 会把以下样式应用到该 em”。

div.sidebar {background:blue;}

div.maincontent {background:white;}

div.sidebar a:link {color:white;}

div.maincontent a:link {color:blue;}

不限定嵌套层数

伪类

a:link {color: #FF0000} /\* 未访问的链接 \*/

a:visited {color: #00FF00} /\* 已访问的链接 \*/

a:hover {color: #FF00FF} /\* 鼠标移动到链接上 \*/

a:active {color: #0000FF} /\* 选定的链接 \*/

定位

对元素进行定位

div、h1 或 p 元素常常被称为块级元素。这意味着这些元素显示为一块内容，即“块框”。与之相反，span 和 strong 等元素称为“行内元素”，这是因为它们的内容显示在行中，即“行内框”。

您可以使用 display 属性改变生成的框的类型。这意味着，通过将 display 属性设置为 block，可以让行内元素（比如 <a> 元素）表现得像块级元素一样。还可以通过把 display 设置为 none，让生成的元素根本没有框。这样的话，该框及其所有内容就不再显示，不占用文档中的空间。

CSS 有三种基本的定位机制：普通流、浮动和绝对定位。

普通流中的元素的位置由元素在 (X)HTML 中的位置决定。

块级框从上到下一个接一个地排列，框之间的垂直距离是由框的垂直外边距计算出来。

表格：

边框折叠为单一边框：

table

{

border-collapse:collapse;

}

table,th, td

{

border: 1px solid black;

}

边框

宽度、样式，以及颜色

子元素选择器

选择只作为 h1 元素子元素的 strong 元素 ，结合符两边可以有空白符

h1 > strong {color:red;}

图片替换

图片点击变化

http://blog.csdn.net/hbcui1984/article/details/42156409

http://www.aseoe.com/api-download/download.html

微信浏览器返回事件

<a id="back" class="mui-action-back mui-icon mui-icon-left-nav mui-pull-left" href="window.history.go(-1)"></a>

手势操作

点击：忘记click

快速响应是mobile App实现的重中之重，研究表明，当延迟超过100毫秒，用户就能感受到界面的卡顿，然而手机浏览器的click点击存在300毫秒延迟（至于为何会延迟，及300毫秒的来龙去脉，请自行谷百），mui为了解决这个问题，封装了tap事件，因此在任何点击的时候，请忘记click及onclick操作

页面关闭：勿重复监听backbutton

mui框架自动封装了页面关闭逻辑，若希望自定义返回逻辑（例如编辑页面的返回，需用户确认放弃草稿后再执行返回逻辑），则需要重写mui.back方法，切勿简单通过addEventListener添加backbutton监听，因为addEventListener只会增加新的执行程序，mui默认封装的监听执行逻辑依然会继续执行，因此若仅addEventListener添加用户确认框，则用户即使选择了取消，也会继续关闭窗口

始终为button按钮添加type属性

若button按钮没有type属性，浏览器默认按照type=submit逻辑处理，这样若将没有type的button放在form表单中，点击按钮就会执行form表单提交，页面就会刷新，用户体验极差。

窗口管理

页面初始化：必须执行mui.init方法

mui在页面初始化时，初始化了很多参数配置，比如：按键监听、手势监听等，因此mui页面都必须调用一次mui.init()方法；

页面跳转：抛弃href跳转

当浏览器加载一个新页面时，若页面DOM尚未渲染完毕，页面会先显示空白，然后等DOM渲染完毕后，再显示具体内容，这是WEB浏览器技术无法逾越的体验障碍；为解决这个问题，建议使用[mui.openWindow方法](http://dcloudio.github.io/mui/javascript/#openwindow)打开一个新的webview，mui会自动监听新页面的loaded事件，若加载完毕，再自动显示新页面；扩展阅读：

http://ask.dcloud.net.cn/article/106

http://ask.dcloud.net.cn/article/25

绑定事件

固定栏

固定栏靠前

所谓的固定栏，也就是带有.mui-bar属性的节点，都是基于fixed定位的元素；常见组件包括：顶部导航栏（.mui-bar-nav）、底部工具条(.mui-bar-footer)、底部选项卡（.mui-bar-tab）;这些元素使用时需遵循一个规则：放在.mui-content元素之前，即使是底部工具条和底部选项卡，也要放在.mui-content之前，否则固定栏会遮住部分主内容；

### 一切内容都要包裹在mui-content中

除了固定栏之外，其它内容都要包裹在.mui-content中，否则就有可能被固定栏遮罩，原因：固定栏基于Fixed定位，不受流式布局限制，普通内容依然会从top:0的位置开始布局，这样就会被固定栏遮罩，mui为了解决这个问题，定义了如下css代码：

设置背景图片

sorticonddl.style.background = 'url(../../images/icon-sortasc.png) no-repeat';

sorticonjg.style.backgroundSize = 'contain';

sorticonddl.style.backgroundSize = 'contain';//不变形

清除缓存：

在js和Css文件引用处：

<link rel="stylesheet" href="../../css/lzh-base.css?timestamp=" + new Date().getTime()+" />

或者

在 URL 参数后加上 "?timestamp=" + new Date().getTime();

在ajax请求中添加cache:false

$.ajax({

url:'www.haorooms.com',

dataType:'json',

data:{},

cache:false,

ifModified :true ,

success:function(response){

//操作

}

async:false

});

用ajax请求服务器最新文件，并加上请求头If-Modified-Since和Cache-Control,如下:

$.ajax({

url:'www.haorooms.com',

dataType:'json',

data:{},

beforeSend :function(xmlHttp){

xmlHttp.setRequestHeader("If-Modified-Since","0");

xmlHttp.setRequestHeader("Cache-Control","no-cache");

},

success:function(response){

//操作

}

async:false

});

清理form表单的临时缓存

<body onLoad="javascript:document.yourFormName.reset()">

meta添加

<META HTTP-EQUIV="pragma" CONTENT="no-cache">

<META HTTP-EQUIV="Cache-Control" CONTENT="no-cache, must-revalidate">

<META HTTP-EQUIV="expires" CONTENT="0">

Canvas

绘制一条对角线

绘图（圆形，点，长方形，椭圆，三角型）

function drawDiagonal(){

var canvas = document.getElementById('diagonal');

var context = canvas.getContext('2d');

//用绝对坐标来创建一条路径

context.beginPath();

context.moveTo(70,140);

context.lineTo(140,70);

//绘制

context.stroke();

}

window.addEventListener('load', drawDiagonal,true);

绘制矩形

fillRect(x, y, width, height)

绘制一个填充的矩形

strokeRect(x, y, width, height)

绘制一个矩形的边框

clearRect(x, y, width, height)

清除指定矩形区域，让清除部分完全透明。

function draw() {

var canvas = document.getElementById('canvas');

if (canvas.getContext) {

var ctx = canvas.getContext('2d');

ctx.fillRect(25,25,100,100);

ctx.clearRect(45,45,60,60);

ctx.strokeRect(50,50,50,50);

}}

绘制路径

1.首先，你需要创建路径起始点。

2.然后你使用画图命令去画出路径。

3.之后你把路径封闭。

4.一旦路径生成，你就能通过描边或填充路径区域来渲染图形。

beginPath()

新建一条路径，生成之后，图形绘制命令被指向到路径上生成路径。

closePath()

闭合路径之后图形绘制命令又重新指向到上下文中。

stroke()

通过线条来绘制图形轮廓。

fill()

通过填充路径的内容区域生成实心的图形。

绘制一个三角形

ctx.beginPath();

ctx.moveTo(75,50);

ctx.lineTo(100,75);

ctx.lineTo(100,25);

ctx.fill();

绘制直线

lineTo(x, y)

绘制一条从当前位置到指定x以及y位置的直线。

// 填充三角形

ctx.beginPath();

ctx.moveTo(25,25);

ctx.lineTo(105,25);

ctx.lineTo(25,105);

ctx.fill();

// 描边三角形

ctx.beginPath();

ctx.moveTo(125,125);

ctx.lineTo(125,45);

ctx.lineTo(45,125);

ctx.closePath();

ctx.stroke();

}

圆弧

绘制圆弧或者圆，我们使用arc()方法。当然可以使用arcTo()，不过这个的现实并不是那么的可靠，所以我们这里不作介绍。

arc(x, y, radius, startAngle, endAngle, anticlockwise)

画一个以（x,y）为圆心的以radius为半径的圆弧（圆），从startAngle开始到endAngle结束，按照anticlockwise给定的方向（默认为顺时针）来生成。

arcTo(x1, y1, x2, y2, radius)

根据给定的控制点和半径画一段圆弧，再以直线连接两个控制点。

该方法有五个参数：x,y为绘制圆弧所在圆上的圆心坐标。radius为半径。startAngle以及endAngle参数用弧度定义了开始以及结束的弧度。这些都是以x轴为基准。参数anticlockwise 为一个布尔值。为true时，是逆时针方向，否则顺时针方向。

for(var i=0;i<4;i++){

for(var j=0;j<3;j++){

ctx.beginPath();

var x = 25+j\*50; // x 坐标值

var y = 25+i\*50; // y 坐标值

var radius = 20; // 圆弧半径

var startAngle = 0; // 开始点

var endAngle = Math.PI+(Math.PI\*j)/2; // 结束点

var anticlockwise = i%2==0 ? false : true; // 顺时针或逆时针

ctx.arc(x, y, radius, startAngle, endAngle, anticlockwise);

if (i>1){

ctx.fill();

} else {

ctx.stroke();

}

}

矩形

Path2D

var rectangle = new Path2D();

rectangle.rect(10, 10, 50, 50);

var circle = new Path2D();

circle.moveTo(125, 35);

circle.arc(100, 35, 25, 0, 2 \* Math.PI);

ctx.stroke(rectangle);

ctx.fill(circle);

# JavaScript

## Md5加密

## js中引入urls.js文件

var urls\_element = document.createElement("script");

urls\_element.setAttribute("type", "text/javascript");

urls\_element.setAttribute("src", "../../js/urls.js");

# 微信端

## 跳转页面传值字符乱码

A跳转到B页面

在A里面

encodeURI(encodeURI(cardName))

window.location.href = '../user/viplistxiaofeizhe.html?cardId='+cardId+'&cardName='+encodeURI(encodeURI(cardName));

B页面接收解析

function GetQueryString(name) {

var reg = new RegExp("(^|&)" + name + "=([^&]\*)(&|$)");

var r = window.location.search.substr(1).match(reg);

if(r != null) return unescape(r[2]);

return null;

}

cardName = GetQueryString('cardName');

document.getElementById('titlehead').innerText = decodeURI(cardName);

# Plus混合开发：

Ios代码：

// 监听plusready事件

document.addEventListener( "plusready", function(){

if(mui.os.ios) {

plus.nativeUI.alert("App全新升级，请去appstore下载安装最新版本", function() {

//跳转到appstore

var UIApplicationClass = plus.ios.importClass("UIApplication");

var openApp=new UIApplicationClass();

var NSURL = plus.ios.import('NSURL');

openApp.openURL(NSURL.URLWithString('https://itunes.apple.com/app/id1137006276'));//商家id，后台配置url和appid

});

}else{

plus.nativeUI.alert("App全新升级，请下载安装最新版本", function() {

plus.runtime.quit();

});

}

}, false );

取消红点

// 监听plusready事件

document.addEventListener("plusready", function() {

// 扩展API加载完毕，现在可以正常调用扩展API

//导入原生类UIApplication

var UIApplication = plus.ios.importClass("UIApplication");

//创建单例sharedApplication

var sharedApplication = new UIApplication();

//设置icon右上角的数字

sharedApplication.plusSetAttribute("applicationIconBadgeNumber",0);

//删除对象

plus.ios.deleteObject(sharedApplication);

}, false);