移动互联网路上的囧小子

- 姓　　名：杨丰盛
- 英文名字：yarin
- 网　　名：半灌水
- 门　　派：移动互联网 ^_^
- 现任公司：云晖软件（成都）有限公司
- 开发经验：J2me、Brew、Android、Iphone、HTML5
- 主要作品：《Android应用开发揭秘》
  《Android技术内幕：系统卷》
- 新浪微博：@杨丰盛（http://weibo.com/yarin）
- 个人主页：http://yarin.blog.51cto.com

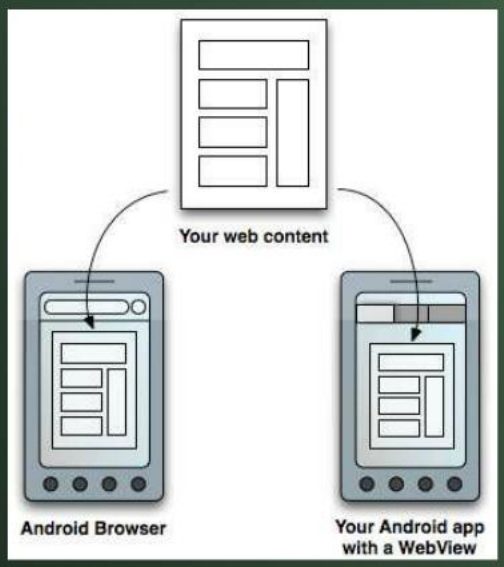
Android应用开发新路线

利用HTML5开发Android应用程序！

Android与HTML5融合

- Android的HTML5应用程序概述
- 如何适配多分辨率的Android设备？
- 如何在Android中构建HTML5应用程序？
- 如何在Android中调试HTML5应用程序？
- 如何在Android中使用HTML5的本地储存？
- 如何在Android中使用HTML5的本地数据库？
- 如何在Android中使用HTML5的地理定位？
- 如何在Android中构建HTML5离线应用？
- 如何使用Canvas进行绘图？

Android HTML5应用概述



(browser) & (SDK+WebView)

HTML5 JavaScript CSS

viewport

适配多分辨率的Android设备

Android设备的多分辨率？

- 物理分辨率
- 视窗大小与WEB页面比例
- 屏幕密度

Android浏览器加载WEB页面时，如果用户没有禁止启用”预览模式“，那么将默认为“预览模式“，通常会缩小WEB页面。而当页面在WebView中显示时，会采用”完全载入“的方式，即保证WEB页面的原始大小。

设备屏幕的密度是基于屏幕的分辨率（由每英寸所包含的点数（dpi））定义的。Android支持三种类别的屏幕密度：低（ldpi），中（mdpi）和高（hdpi）。与中等密度屏幕相比，低密度屏幕每英寸像素较少，高密度屏幕每英寸像素较多。默认情况下，Android浏览器和 WebView是针对中等密度的屏幕。Android浏览器和 WebView 在高密度屏幕上将Web页面缩放约1.5倍（因为中等密度屏幕像素较小），而在低密度屏幕上将Web页面缩放约0.75倍（因为中等密度屏幕像素大）。

怎么办？

- viewport属性
- 用CSS控制设备密度
- 用JavaScript控制设备密度

viewport属性的应用

- viewport需要放置在HTML的<meta>标签中，在<meta>标签的content属性中，就可以定义多个视窗特性。包括视窗的宽度、高度、缩放比例，目标设备密度等，但是，需要注意每个视窗属性必须有逗号隔开。

```
<head>
    <title>Exmaple</title>
    <meta name="viewport" content="width=device-width,user-scalable=no"/>
</head>
```

```
<meta name="viewport"
      content="
            height = [pixel_value | device-height] ,
            width = [pixel_value | device-width ] ,
            initial-scale = float_value ,
            minimum-scale = float_value ,
            maximum-scale = float_value ,
            user-scalable = [yes | no] ,
            target-densitydpi = [dpi_value | device-dpi |
                                 high-dpi | medium-dpi | low-dpi]
      " />
```

CSS控制设备密度

- Android浏览和WebView支持CSS媒体性能（webkit-device-pixel-ratio），允许指定屏幕密度创建一些样式CSS媒体性能。该值应该是“0.75”，“1”或“1.5”，它们分别表示对于低、中、高密度屏幕的设备。

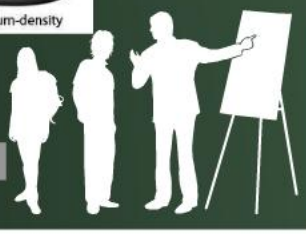
下面为每种密度创建独立的样式：

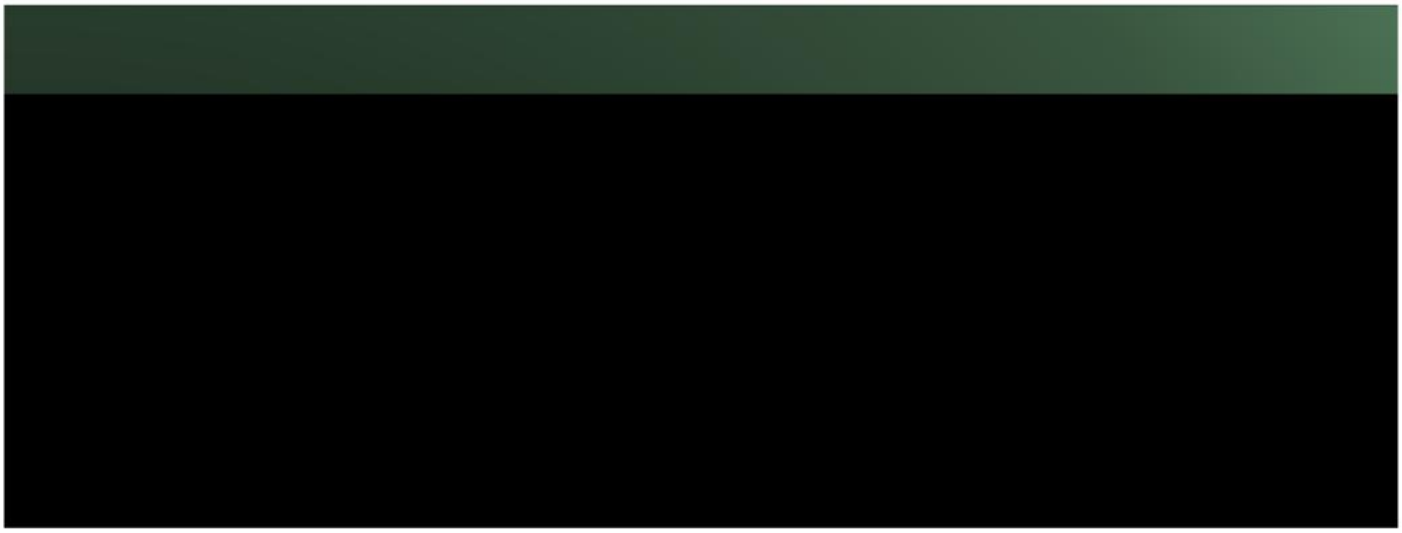
```
<link rel="stylesheet" media="screen and (-webkit-device-pixel-ratio: 1.5)" href="hdpi.css" />
<link rel="stylesheet" media="screen and (-webkit-device-pixel-ratio: 1.0)" href="mdpi.css" />
<link rel="stylesheet" media="screen and (-webkit-device-pixel-ratio: 0.75)" href="ldpi.css" />
```

在一个样式表中，指定不同样式：

```
#header {
    background:url(medium-density-image.png);
}
@media screen and (-webkit-device-pixel-ratio: 1.5) {
    // CSS for high-density screens
    #header {
        background:url(high-density-image.png);
    }
}
@media screen and (-webkit-device-pixel-ratio: 0.75) {
    // CSS for low-density screens
    #header {
        background:url(low-density-image.png);
    }
}
```


```
<meta name="viewport" content="target-densitydpi=device-dpi, width=device-width" />
```





在Android中构建HTML5应用程序

- 使用WebView在Android中构建Web应用
- 处理页面导航
- 浏览网页历史记录
- Android与JavaScript交互

Android WebView应用

WebView 类是Android View类的扩展，它允许Web页面作为Activity布局的一部分显示。它不包括完整Web浏览器的任何功能，如导航控制或地址栏。默认情况下WebView 所能做的就是显示一个网页。

添加WebView到应用程序中:

```
<?xml version="1.0" encoding="utf-8"?>
<WebView xmlns:android="http://schemas.android.com/apk/res/android"
        android:id="@+id/webview"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
/>
```

要在WebView中加载Web页面，使用loadUrl()

```
WebView myWebView = (WebView) findViewById(R.id.webview);
myWebView.loadUrl("http://www.example.com");
```

```
<manifest ... >
        <uses-permission android:name="android.permission.INTERNET" />
        ...
</manifest>
```

Linux公社 www.linuxidc.com

- Linux公社（LinuxIDC.com）于2006年9月25日注册并开通网站，Linux现在已经成为一种广受关注和支持的一种操作系统，IDC是互联网数据中心，LinuxIDC就是关于Linux的数据中心。

- LinuxIDC.com提供包括Ubuntu，Fedora，SUSE技术，以及最新IT资讯等Linux专业类网站。

处理页面导航

在WebView中，当用户从Web页面里点击一个链接，在Android中，默认行为是启动一个应用程序来处理URL。 通常，默认Web浏览器打开并加载这个目的URL。 但是，您可以为您的 WebView忽略此默认行为 ，由 WebView 打开所有链接。 然后，通过WebView，您可以运行用户向前、向后浏览他们的Web页面的历史 。

```
private class MyWebViewClient extends WebViewClient {
    public boolean shouldOverrideUrlLoading(WebView view, String url) {
        if (Uri.parse(url).getHost().equals("www.example.com")) {
            return false;
        }
        Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse(url));
        startActivity(intent);
        return true;
    }
}
```

```
WebView myWebView = (WebView) findViewById(R.id.webview);
myWebView.setWebViewClient(new MyWebViewClient());
```

浏览网页历史记录

当 WebView 重写URL加载后，它会自动累计访问过Web页面的历史。
你可以用goBack()和goForward()向前和向后浏览历史页面。

```
public boolean onKeyDown(int keyCode, KeyEvent event) {
    if ((keyCode == KeyEvent.KEYCODE_BACK) && myWebView.canGoBack() {
        myWebView.goBack();
        return true;
    }
    return super.onKeyDown(keyCode, event);
}
```

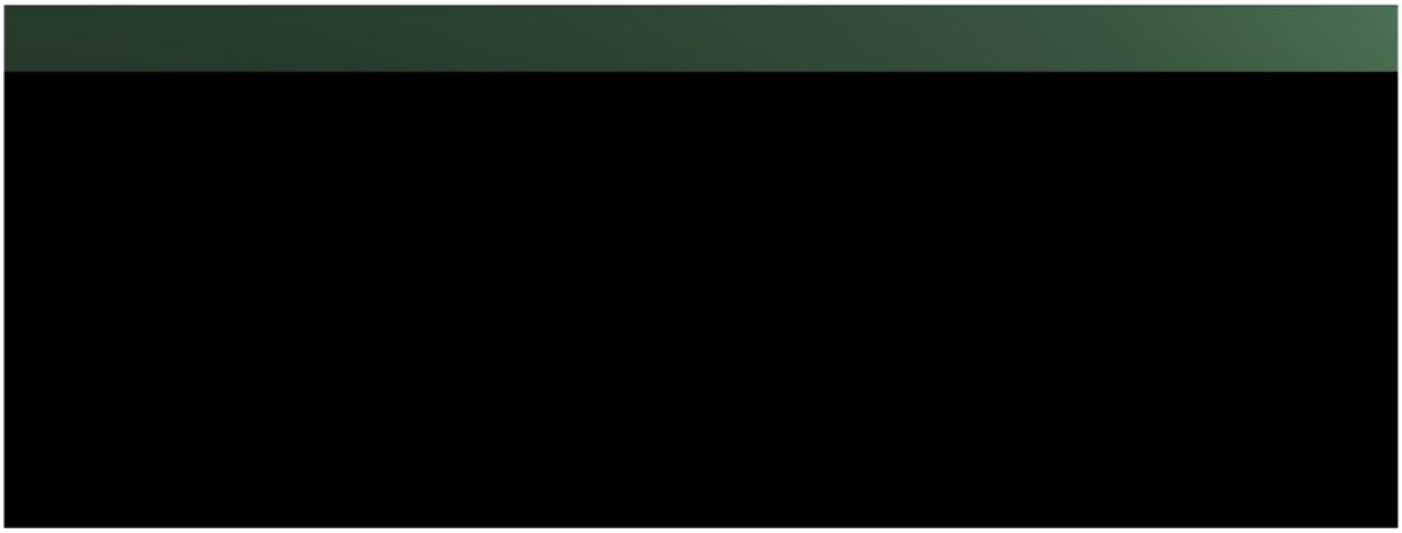
Android与JavaScript交互

- 如果您计划使用JavaScript将Web页面加载到WebView 中，您就必须为您的 WebView 启用JavaScript。一旦启用JavaScript，您就可以在您的应用程序与您的JavaScript代码之间建立接口。
- 默认情况下，在WebView中，JavaScript是禁用的。您可以通过将WebSettings附加到您的WebView中来启用JavaScript。你可以用getSettings()检索WebSettings，然后用setJavaScriptEnabled() 启动JavaScript。

```
WebView myWebView = (WebView) findViewById(R.id.webview);
WebSettings webSettings = myWebView.getSettings();
webSettings.setJavaScriptEnabled(true);
```

Android与JS通信实例演示:

- 在Android中处理JS的警告、对话框等；
- 在JS中调用Android接口；
- 在Android调用JS函数。



在WebView中用控制台API

- 在调试您的WebView的Web页面时，是支持控制台API。在Android 1.6和更低版本下，控制台信息自动发送到Logcat，并加以“WebCore”标签。 如果您是针对Android 2.1（API Level 7）或更高版本，那么你必须提供一个 WebChromeClient 实现 onConsoleMessage() 回调方法，为了使控制台的信息显示到Logcat中。

```
WebView myWebView = (WebView) findViewById(R.id.webview);
myWebView.setWebChromeClient(new WebChromeClient() {
    public void onConsoleMessage(String message, int lineNumber, String sourceID) {
        Log.d("MyApplication", message + " -- From line "
             + lineNumber + " of "
             + sourceID);
    }
});
```

```
WebView myWebView = (WebView) findViewById(R.id.webview);
myWebView.setWebChromeClient(new WebChromeClient() {
    public boolean onConsoleMessage(ConsoleMessage cm) {
        Log.d("MyApplication", cm.message() + " -- From line "
             + cm.lineNumber() + " of "
             + cm.sourceId() );
        return true;
    }
});
```

ConsoleMessage 还包括一个 MessageLevel 表示控制台传递信息类型。 您可以用 messageLevel()查询信息级别，以确定信息的严重程度，然后使用适当的Log方法或采取其他适当的措施。

HTML5本地储存在Android中的应用

HTML5 提供了两种在客户端存储数据的新方法：

- ◆localStorage - 没有时间限制的数据存储
- ◆sessionStorage - 针对一个 session 的数据存储

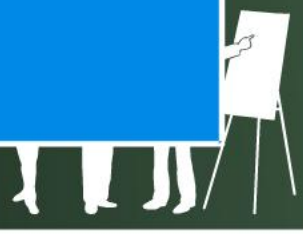
```
<script type="text/javascript">
    localStorage.lastname="Smith";
    document.write(localStorage.lastname);
</script>
```

```
<script type="text/javascript">
    sessionStorage.lastname="Smith";
    document.write(sessionStorage.lastname);
 </script>
```

Web Storage API：

```
//清空storage
localStorage.clear();
//设置一个键值
localStorage.setItem("yarin","yangfegnsheng");
//获取一个键值
localStorage.getItem("yarin");
//获取指定下标的键的名称（如同Array）
localStorage.key(0);
//return "fresh" //删除一个键值
localStorage.removeItem("yarin");
```

setDomStorageEnabled（true）



HTML5地理定位在Android中的应用

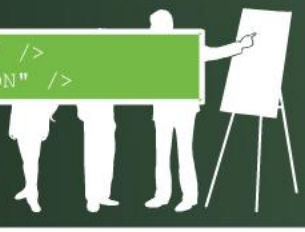
HTML5提供了一组API用来获取用户的地理位置，如果浏览器支持且设备具有定位功能，就能够直接使用这组API来获取当前位置信息。

地理定位演示：

```
//启用地理定位
webSettings.setGeolocationEnabled(true);
//设置定位的数据库路径
webSettings.setGeolocationDatabasePath(dir);

//配置权限
public void onGeolocationPermissionsShowPrompt(String origin,
                GeolocationPermissions.Callback callback) {
    callback.invoke(origin, true, false);
    super.onGeolocationPermissionsShowPrompt(origin, callback);
}
```

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
```



HTML5地理定位API

- W3C 中新添加了一个名为 Geolocation的 API 规范，Geoloaction API的作用就是通过浏览器获取用户的地理位置。我们可以使用navigator.geolocation来简单的获取用户的地理位置信息。

常用的navigator.geolocation对象有以下三种方法：

```
//获取当前地理位置
navigator.geolocation.getCurrentPosition(success_callback_function,
error_callback_function, position_options)

//持续获取地理位置
navigator.geolocation.watchPosition(success_callback_function,
error_callback_function, position_options)

//清除持续获取地理位置事件
navigator.geolocation.clearWatch(watch_position_id)
```

其中success_callback_function为成功之后处理的函数，error_callback_function为失败之后返回的处理函数，参数position_options是配置项

HTML5地理定位API

position对象所包含的数据：

Property	Type	Notes
coords.latitude（纬度）	double	Decimal degrees
coords.longitude（经度）	double	Decimal degrees
coords.altitude（海拔）	double or null	Meters
coords.accuracy（精确度）	double	Meters
coords.altitudeAccuracy	double or null	Meters
coords.heading（朝向）	double or null	degrees clockwise from the north
coords.speed	double or null	Meters/second
timestamp	DOMTimeStamp	Like a Date() object

Error代码：

- TIMEOUT:表示获取信息超时。
- PERMISSION_DENIED：表示用户选择了拒绝了位置服务。
- POSITION_UNAVAILABLE：表示位置不可知。

Position选项：

Property	Type	Default	Notes
enableHighAccuracy	boolean	false	true might be slower
timeout	long	(no default)	In milliseconds
maximumAge	long	0	In milliseconds

构建HTML5离线应用

- 为了能够让用户在离线状态下继续访问 Web 应用，开发者需要提供一个 cache manifest 文件。这个文件中列出了所有需要在离线状态下使用的资源，浏览器会把这些资源缓存到本地。

cache manifest文件例子：

```
CACHE MANIFEST
#这是注释
images/sound-icon.png
images/background.png

NETWORK:
test.cgi

CACHE:
style/default.css

FALLBACK:
/files/projects /projects
```

```
<html manifest="clock.manifest">
```

HTML5离线应用的更新缓存机制

应用程序可以等待浏览器自动更新缓存，也可以使用 Javascript 接口手动触发更新。

◆自动更新
浏览器除了在第一次访问 Web 应用时缓存资源外，只会在 cache manifest 文件本身发生变化时更新缓存。而 cache manifest 中的资源文件发生变化并不会触发更新。

◆手动更新
开发者也可以使用 window.applicationCache 的接口更新缓存。方法是检测 window.applicationCache.status 的值，如果是 UPDATEREADY，那么可以调用 window.applicationCache.update() 更新缓存。

```
if (window.applicationCache.status == window.applicationCache.UPDATEREADY) {
    window.applicationCache.update();
}
```

在线状态检测
HTML5 提供了两种检测是否在线的方式：navigator.online 和 online/offline事件。

◆navigator.onLine
navigator.onLine 属性表示当前是否在线。如果为 true，表示在线；如果为false，表示离线。当网络状态发生变化时，navigator.onLine 的值也随之变化。开发者可以通过读取它的值获取网络状态。

◆online/offline事件
当开发离线应用时，通过 navigator.onLine获取网络状态通常是不够的。开发者还需要在网络状态发生变化时立刻得到通知，因此 HTML5 还提供了 online/offline 事件。当在线 / 离线状态切换时，online/offline 事件将触发在 body 元素上，并且沿着 document.body、document 和 window 的顺序冒泡。因此，开发者可以通过监听它们的 online/offline 事件来获悉网络状态。

在Android中构建HTML5离线应用

```
//开启应用程序缓存
webSettingssetAppCacheEnabled(true);
String dir = this.getApplicationContext().getDir("cache",
Context.MODE_PRIVATE).getPath();
//设置应用缓存的路径
webSettings.setAppCachePath(dir);
//设置缓存的模式
webSettings.setCacheMode(WebSettings.LOAD_DEFAULT);
//设置应用缓存的最大尺寸
webSettings.setAppCacheMaxSize(1024*1024*8);


//扩充缓存的容量
public void onReachedMaxAppCacheSize(long spaceNeeded,
            long totalUsedQuota, WebStorage.QuotaUpdater quotaUpdater) {
    quotaUpdater.updateQuota(spaceNeeded * 2);
}
```

使用Canvas绘制图形图像

什么是 Canvas？

HTML5的canvas元素使用JavaScript在网页上绘制图像。
画布是一个矩形区域，您可以控制其每一像素。
canvas 拥有多种绘制路径、矩形、圆形、字符以及添加图像的方法。

创建Canvas元素

向HTML5页面添加canvas元素。
规定元素的 id、宽度和高度：

```
<canvas id="myCanvas" width="200" height="100"></canvas>
```

通过 JavaScript 来绘制

canvas元素本身是没有绘图能力的。所有的绘制工作必须在JavaScript内部完成：

```
<script type="text/javascript">
    //通过id获取canvas元素
    var c=document.getElementById("myCanvas");
    //创建context对象
    var cxt=c.getContext("2d");
    cxt.fillStyle="#FF0000";
    cxt.fillRect(0,0,150,75);
</script>
```

结束

谢谢！

2011-07-16