

2.3.2 EditText(输入框)详解

分类 [Android 基础入门教程](#)

Android 基础入门教程(Q群号 : 153836263)

本节引言：

上一节中我们学习了第一个 UI控件**TextView (文本框)**，文中给出了很多实际开发中可能遇到的一些需求 的解决方法，应该会为你的开发带来便利，在本节中，我们来学习第二个很常用的控件**EditText(输入框)**；和TextView非常类似，最大的区别是：EditText可以接受用户输入！和前面一样，我们不是一个一个讲属性，只讲实际应用，要扣属性可以自己查看API文档：[API文档](#)；那么开始本节内容！

1.设置默认提示文本

如下图，相信你对于这种用户登录的界面并不陌生，是吧，我们很多时候都用的这种界面



相比另外这种，下面这种又如何？

- 1.0 Android基础入门教程
 - 1.0.1 2015年最新Android基...
- 1.1 背景相关与系统架构分析
- 1.2 开发环境搭建
 - 1.2.1 使用Eclipse + ADT + S...
 - 1.2.2 使用Android Studio开...
- 1.3 SDK更新不了问题解决
- 1.4 Genymotion模拟器安装
 - 1.5.1 Git使用教程之本地仓...
 - 1.5.2 Git之使用GitHub搭建...
- 1.6 .9(九妹)图片怎么玩
- 1.7 界面原型设计
- 1.8 工程相关解析(各种文件...
- 1.9 Android程序签名打包
- 1.11 反编译APK获取代码&...
- 2.1 View与ViewGroup的概念
 - 2.2.1 LinearLayout(线性布局)
 - 2.2.2 RelativeLayout(相对布...
 - 2.2.3 TableLayout(表格布局)
 - 2.2.4 FrameLayout(帧布局)
 - 2.2.5 GridLayout(网格布局)
 - 2.2.6 AbsoluteLayout(绝对...
- 2.3.1 TextView(文本框)详解
- 2.3.2 EditText(输入框)详解
- 2.3.3 Button(按钮)与ImageB...
- 2.3.4 ImageView(图像视图)
- 2.3.5.RadioButton(单选按钮...
- 2.3.6 开关按钮ToggleButton...
- 2.3.7 ProgressBar(进度条)
- 2.3.8 SeekBar(拖动条)
- 2.3.9 RatingBar(星级评分条)
- 2.4.1 ScrollView(滚动条)



还不赖是吧，当然，不会在这里贴布局，这里只介绍默认提示文本的两个控制属性：

默认提示文本的两个属性如下：

```
android:hint="默认提示文本"
android:textColorHint="#95A1AA"
```

前者设置提示的文本内容，后者设置提示文本的颜色！

2. 获得焦点后全选组件内所有文本内容

当我们点击想当我们的输入框获得焦点后，不是将光标移动到文本的开始或者

2.4.2 Date & Time组件(上)

2.4.3 Date & Time组件(下)

2.4.4 Adapter基础讲解

2.4.5 ListView简单实用

2.4.6 BaseAdapter优化

2.4.7 ListView的焦点问题

2.4.8 ListView之checkbox错...

2.4.9 ListView的数据更新问题

2.5.0 构建一个可复用的自定...

2.5.1 ListView Item多布局的...

2.5.2 GridView(网格视图)的...

2.5.3 Spinner(列表选项框)...

2.5.4 AutoCompleteTextVie...

2.5.5 ExpandableListView(...

2.5.6 ViewPager(翻转视图)...

2.5.7 Toast(吐司)的基本使用

2.5.8 Notification(状态栏通...

2.5.9 AlertDialog(对话框)详解

2.6.0 其他几种常用对话框基...

2.6.1 PopupWindow(悬浮框)...

2.6.2 菜单(Menu)

2.6.3 ViewPager的简单使用

2.6.4 DrawerLayout(官方侧...

3.1.1 基于监听的事件处理机制

3.2 基于回调的事件处理机制

3.3 Handler消息传递机制浅析

3.4 TouchListener PK OnTo...

3.5 监听EditText的内容变化

3.6 响应系统设置的事件(Co...

3.7 AsyncTask异步任务

3.8 Gestures(手势)

4.1.1 Activity初学乍练

4.1.2 Activity初窥门径

4.1.3 Activity登堂入室

4.2.1 Service初涉

4.2.2 Service进阶

4.2.3 Service精通

4.3.1 BroadcastReceiver牛...

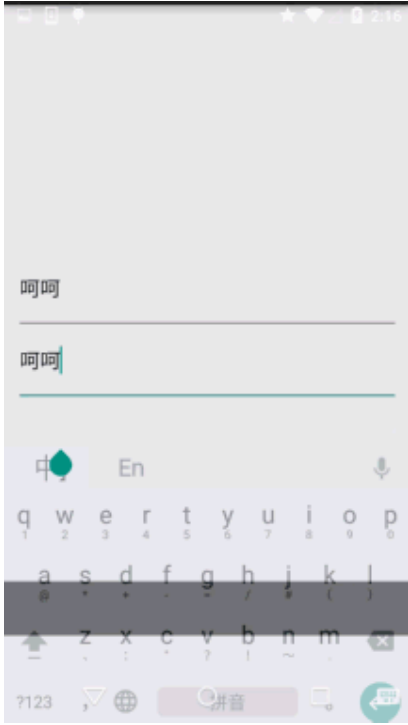
4.3.2 BroadcastReceiver庖...

4.4.1 ContentProvider初探

结尾；而是 获取到输入框中所有的文本内容的话！这个时候我们可以使用**selectAllOnFocus**属性

```
android:selectAllOnFocus="true"
```

比如下面的效果图: 第一个是设置了该属性的，第二个是没设置该属性的，设置为true的EditText获得焦点后 选中的是所有文本！



3.限制EditText输入类型

有时我们可能需要对输入的数据进行限制，比如输入电话号码的时候，你输入了一串字母，这 显然是不符合我们预期的，而限制输入类型可以通过属性来实现！

比如限制只能为电话号码,密码(**textPassword**)：

```
<EditText
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:inputType="phone" />
```

可选参数如下：

文本类型，多为大写、小写和数字符号

```
android:inputType="none"
android:inputType="text"
android:inputType="textCapCharacters"
android:inputType="textCapWords"
android:inputType="textCapSentences"
android:inputType="textAutoCorrect"
android:inputType="textAutoComplete"
```

4.4.2 ContentProvider再探...

4.5.1 Intent的基本使用

4.5.2 Intent之复杂数据的传递

5.1 Fragment基本概述

5.2.1 Fragment实例精讲—...

5.2.2 Fragment实例精讲—...

5.2.3 Fragment实例精讲—...

5.2.4 Fragment实例精讲—...

5.2.5 Fragment实例精讲—...

6.1 数据存储与访问之——文...

6.2 数据存储与访问之——S...

6.3.1 数据存储与访问之——...

6.3.2 数据存储与访问之——...

7.1.1 Android网络编程要学...

7.1.2 Android Http请求头与...

7.1.3 Android HTTP请求方...

7.1.4 Android HTTP请求方...

7.2.1 Android XML数据解析

7.2.2 Android JSON数据解析

7.3.1 Android 文件上传

7.3.2 Android 文件下载（1）

7.3.3 Android 文件下载（2）

7.4 Android 调用 Webservice

7.5.1 WebView(网页视图)基...

7.5.2 WebView和JavaScrip...

7.5.3 Android 4.4后WebVie...

7.5.4 WebView文件下载

7.5.5 WebView缓存问题

7.5.6 WebView处理网页返...

7.6.1 Socket学习网络基础准备

7.6.2 基于TCP协议的Socket...

7.6.3 基于TCP协议的Socket...

7.6.4 基于UDP协议的Socket...

8.1.1 Android中的13种Draw...

8.1.2 Android中的13种Draw...

8.1.3 Android中的13种Draw...

8.2.1 Bitmap(位图)全解析 P...

8.2.2 Bitmap引起的OOM问题

8.3.1 三个绘图工具类详解

8.3.2 绘图类实战示例

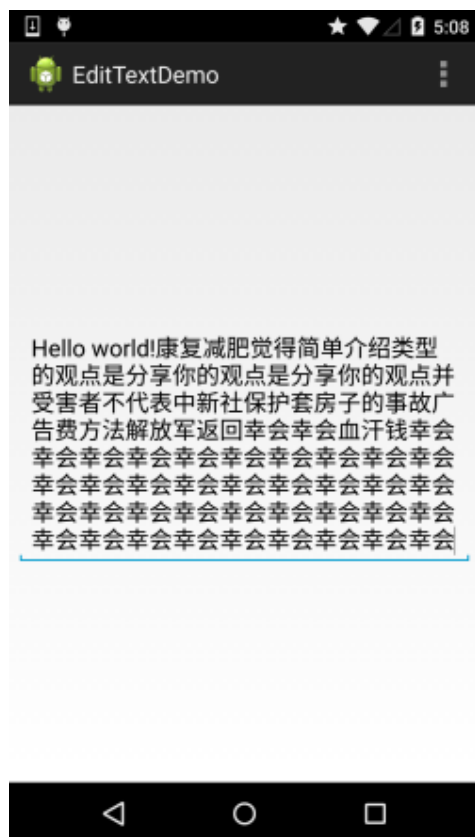
```
android:inputType="textMultiLine"
android:inputType="textImeMultiLine"
android:inputType="textNoSuggestions"
android:inputType="textUri"
android:inputType="textEmailAddress"
android:inputType="textEmailSubject"
android:inputType="textShortMessage"
android:inputType="textLongMessage"
android:inputType="textPersonName"
android:inputType="textPostalAddress"
android:inputType="textPassword"
android:inputType="textVisiblePassword"
android:inputType="textWebEditText"
android:inputType="textFilter"
android:inputType="textPhonetic"
```

数值类型

```
android:inputType="number"
android:inputType="numberSigned"
android:inputType="numberDecimal"
android:inputType="phone" //拨号键盘
android:inputType="datetime"
android:inputType="date" //日期键盘
android:inputType="time" //时间键盘
```

4.设置最小行，最多行，单行，多行，自动换行

EditText默认是多行显示的，并且能够自动换行，即当一行显示不完的时候，他会自动换到第二行



8.3.3 Paint API之——Mask...

8.3.4 Paint API之——Xferm...

8.3.5 Paint API之——Xferm...

8.3.6 Paint API之——Xferm...

8.3.7 Paint API之——Xferm...

8.3.8 Paint API之——Xferm...

8.3.9 Paint API之—— Color...

8.3.10 Paint API之—— Colo...

8.3.11 Paint API之—— Colo...

8.3.12 Paint API之——Path...

8.3.13 Paint API之——Sha...

8.3.14 Paint几个枚举/常量值...

8.3.15 Paint API之——Type...

8.3.16 Canvas API详解(Part 1)

8.3.17 Canvas API详解(Part...

8.3.18 Canvas API详解(Part...

8.4.1 Android动画合集之帧...

8.4.2 Android动画合集之补...

8.4.3 Android动画合集之属...

8.4.4 Android动画合集之属...

9.1 使用SoundPool播放音...

9.2 MediaPlayer播放音频与...

9.3 使用Camera拍照

9.4 使用MediaRecord录音

10.1 TelephonyManager(电...

10.2 SmsManager(短信管理...

10.3 AudioManager(音频管...

10.4 Vibrator(振动器)

10.5 AlarmManager(闹钟服务)

10.6 PowerManager(电源服...

10.7 WindowManager(窗口管理)

10.8 LayoutInflater(布局服务)

10.9 WallpaperManager(壁...

10.10 传感器专题(1)——相...

10.11 传感器专题(2)——方...

10.12 传感器专题(3)——加...

10.12 传感器专题(4)——其...

10.14 Android GPS初涉

11.0 《2015最新Android基...

我们可以对其进行限制，比如

设置最小行的行数：**android:minLines="3"**

或者设置EditText最大的行数：**android:maxLines="3"**

PS:当输入内容超过maxline,文字会自动向上滚动！！

另外很多时候我们可能要限制EditText只允许单行输入，而且不会滚动，比如上面的登陆界面的例子，我们只需要设置

```
android:singleLine="true"</p>
<p>
即可实现单行输入不换行</p>

<hr />

<h2>5. 设置文字间隔，设置英文字母大写类型</h2>

<p>我们可以通过下述两个属性来设置字的间距：
</p>
<pre>
android:textScaleX="1.5"    //设置字与字的水平间隔
android:textScaleY="1.5"    //设置字与字的垂直间隔
```

另外EditText还为我们提供了设置英文字母大写类型的属性：**android:capitalize** 默认none，提供了三个可选值：

sentences：仅第一个字母大写

words：每一个单词首字母大小，用空格区分单词

characters:每一个英文字母都大写

6.控制EditText四周的间隔距离与内部文字与边框间的距离

我们使用**margin**相关属性增加组件相对其他控件的距离，比如
`android:marginTop = "5dp"` 使用**padding**增加组件内文字和组件边框的距离，比如`android:paddingTop = "5dp"`

7.设置EditText获得焦点，同时弹出小键盘

关于这个EditText获得焦点，弹出小键盘的问题，前不久的项目中纠结了笔者一段时间 需求是：进入Activity后，让EditText获得焦点，同时弹出小键盘供用户输入！试了很多网上的方法都不可以，不知道是不是因为笔者用的5.1的系统的的问题！下面小结下：

首先是让EditText获得焦点与清除焦点的

```
edit.requestFocus(); //请求获取焦点
edit.clearFocus(); //清除焦点
```

获得焦点后，弹出小键盘，笔者大部分时间就花在这个上：

低版本的系统直接requestFocus就会自动弹出小键盘了

稍微高一点的版本则需要我们手动地去弹键盘：第一种：

```
InputMethodManager imm = (InputMethodManager) getSystemService(
Context.INPUT_METHOD_SERVICE);
imm.toggleSoftInput(0, InputMethodManager.HIDE_NOT_ALWAYS);
```

第二种：

```
InputMethodManager imm = (InputMethodManager) getSystemService(
Context.INPUT_METHOD_SERVICE);      imm.showSoftInput(view, Input
MethodManager.SHOW_FORCED);
imm.hideSoftInputFromWindow(view.getWindowToken(), 0); //强制隐
藏键盘
```

不知道是什么原因，上面这两种方法并没有弹出小键盘，笔者最后使用了windowSoftInputMode属性解决了弹出小键盘的问题，这里跟大家分享一下：

android:windowSoftInputMode Activity主窗口与软键盘的交互模式，可以用来避免输入法面板遮挡问题，Android1.5后的一个新特性。

这个属性能影响两件事情：

- 【一】当有焦点产生时，软键盘是隐藏还是显示
- 【二】是否减少活动主窗口大小以便腾出空间放软键盘

简单点就是有焦点时的键盘控制以及是否减少Act的窗口大小，用来放小键盘

有下述值可供选择，可设置多个值，用"|"分开

stateUnspecified：软键盘的状态并没有指定，系统将选择一个合适的状态或依赖于主题的设置

stateUnchanged：当这个activity出现时，软键盘将一直保持在上一个activity里的状态，无论是隐藏还是显示

stateHidden：用户选择activity时，软键盘总是被隐藏

stateAlwaysHidden：当该Activity主窗口获取焦点时，软键盘也总是被隐藏的

stateVisible：软键盘通常是可见的

stateAlwaysVisible：用户选择activity时，软键盘总是显示的状态

adjustUnspecified：默认设置，通常由系统自行决定是隐藏还是显示

adjustResize：该Activity总是调整屏幕的大小以便留出软键盘的空间

adjustPan：当前窗口的内容将自动移动以便当前焦点从不被键盘覆

盖和用户能总是看到输入内容的部分

我们可以在AndroidManifest.xml为需要弹出小键盘的Activity设置这个属性，比如：

```
<activity
    android:name=".MainActivity"
    android:label="@string/app_name"
    android:windowSoftInputMode="stateVisible" >
```

然后在EditText对象requestFocus()就可以了~

8.EditText光标位置的控制

有时可能需要我们控制EditText中的光标移动到指定位置或者选中某些文本！

EditText为我们提供了**setSelection()**的方法，方法有两种形式：

```
● setSelection(int index) : void - EditText
● setSelection(int start, int stop) : void - EditText
```

一个参数的是设置光标位置的，两个参数的是设置起始位置与结束位置的中间括的部分，即部分选中！

当然我们也可以调用**setSelectAllOnFocus(true)**；让EditText获得焦点时选中全部文本！

另外我们还可以调用**setCursorVisible(false)**；设置光标不显示
还可以调用**getSelectionStart()**和**getSelectionEnd**获得当前光标的前后位置

9.带表情的EditText的简单实现

相信大家对于QQ或者微信很熟悉吧，我们发送文本的时候可以连同表情一起发送，有两种简单的实现方式：

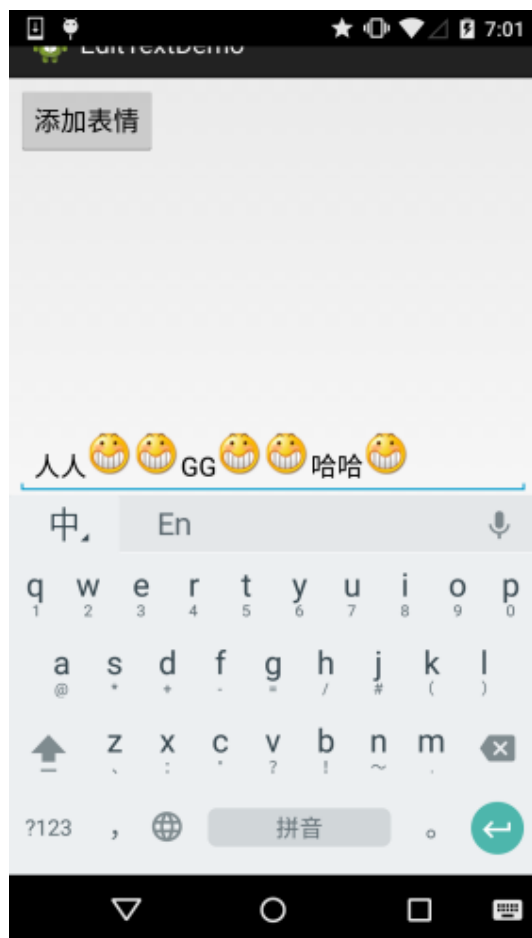
1.使用SpannableString来实现

2.使用Html类来实现

这里笔者用的是第一种，这里只实现一个简单的效果，大家可以把方法抽取出来，自定义一个EditText；

也可以自己动手写个类似于QQ那样有多个表情选择的输入框！

看下效果图(点击添加表情即可完成表情添加)：



代码也很简单：

```
public class MainActivity extends Activity {
    private Button btn_add;
    private EditText edit_one;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        btn_add = (Button) findViewById(R.id.btn_add);
        edit_one = (EditText) findViewById(R.id.edit_one);
        btn_add.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                SpannableString spanStr = new SpannableString("
img");
                Drawable drawable = MainActivity.this.getResources().getDrawable(R.drawable.f045);
                drawable.setBounds(0,0,drawable.getIntrinsicWidth(),drawable.getIntrinsicHeight());
                ImageSpan span = new ImageSpan(drawable, ImageSpan.ALIGN_BASELINE);
                spanStr.setSpan(span,0,4,Spannable.SPAN_EXCLUSIVE_EXCLUSIVE);
                int cursor = edit_one.getSelectionStart();
                edit_one.getText().insert(cursor, spanStr);
            }
        });
    }
}
```

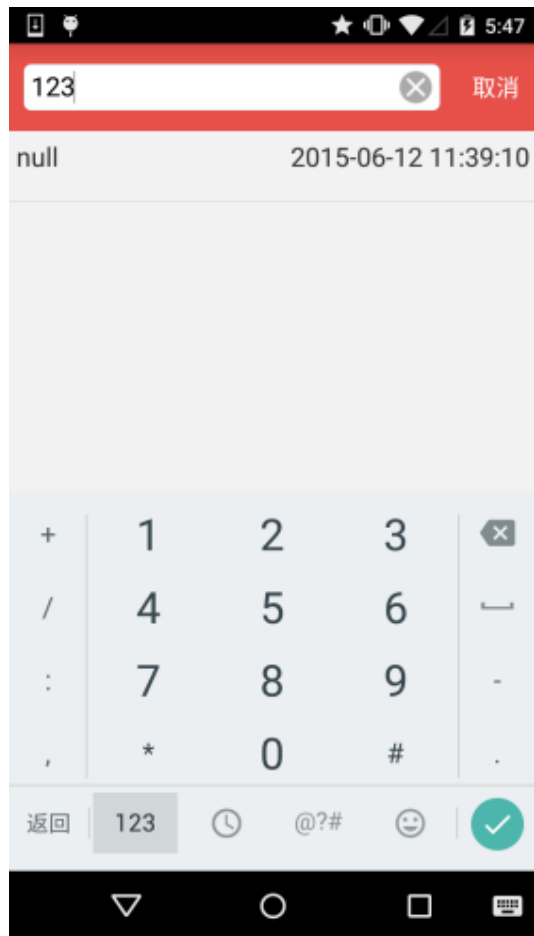


```
}  
}
```

PS：对了，别忘了放一个图片哦~

10.带删除按钮的EditText

我们常常在App的输入界面上看到：



当我们输入内容后，右面会出现这样一个小叉叉的图标，我们点击后会清空输入框中的内容！

实现起来其实也很简单：

为EditText设置addTextChangedListener，然后重写TextWatcher（）里的抽象方法，这个用于监听输入框变化的；然后
setCompoundDrawablesWithIntrinsicBounds设置小叉叉的图片；最后，
重写onTouchEvent方法，如果点击区域是小叉叉图片的位置，清空文本！
实现代码如下：

```
public class EditTextWithDel extends EditText {  
  
    private final static String TAG = "EditTextWithDel";  
    private Drawable imgInable;  
    private Drawable imgAble;  
    private Context mContext;  
  
    public EditTextWithDel(Context context) {
```

```

        super(context);
        mContext = context;
        init();
    }

    public EditTextWithDel(Context context, AttributeSet attrs)
    {
        super(context, attrs);
        mContext = context;
        init();
    }

    public EditTextWithDel(Context context, AttributeSet attrs,
int defStyleAttr) {
        super(context, attrs, defStyleAttr);
        mContext = context;
        init();
    }

    private void init() {
        imgInable = mContext.getResources().getDrawable(R.drawable.delete_gray);
        addTextChangedListener(new TextWatcher() {
            @Override
            public void onTextChanged(CharSequence s, int start
, int before, int count) {
            }

            @Override
            public void beforeTextChanged(CharSequence s, int s
tart, int count, int after) {
            }

            @Override
            public void afterTextChanged(Editable s) {
                setDrawable();
            }
        });
        setDrawable();
    }

    // 设置删除图片
    private void setDrawable() {
        if (length() < 1)
            setCompoundDrawablesWithIntrinsicBounds(null, null,
null, null);
        else
            setCompoundDrawablesWithIntrinsicBounds(null, null,
imgInable, null);
    }

    // 处理删除事件
    @Override
    public boolean onTouchEvent(MotionEvent event) {

```

```
        if (imgAble != null && event.getAction() == MotionEvent
.ACTION_UP) {
            int eventX = (int) event.getRawX();
            int eventY = (int) event.getRawY();
            Log.e(TAG, "eventX = " + eventX + "; eventY = " + e
ventY);

            Rect rect = new Rect();
            getGlobalVisibleRect(rect);
            rect.left = rect.right - 100;
            if (rect.contains(eventX, eventY))
                setText("");
        }
        return super.onTouchEvent(event);
    }
    @Override
    protected void finalize() throws Throwable {
        super.finalize();
    }
}
```

本节小结：

本节给大家介绍了Android UI控件中的EditText(输入框)控件，用法有很多，当然上述情况肯定满足不了实际需求的，实际开发中我们可能需要根据自己的需求来自定义EditText！当然，这就涉及到了自定义控件这个高级一点的主题了，在进阶部分我们会对Android中的自定义控件进行详细的讲解！现在会用就可以了~

← 2.3.1 TextView(文本框)详解 2.3.3 Button(按钮)与ImageButton(图像按钮) →



	<div>在线实例</div> <ul style="list-style-type: none">· HTML 实例· CSS 实例· JavaScript 实例· Ajax 实例· jQuery 实例· XML 实例· Java 实例	<div>字符集&工具</div> <ul style="list-style-type: none">· HTML 字符集设置· HTML ASCII 字符集· HTML ISO-8859-1· HTML 实体符号· HTML 拾色器· JSON 格式化工具	<div>最新更新</div> <ul style="list-style-type: none">· PHP接收 json , 并...· Foundation CSS ...· Foundation CSS ...· Foundation 图标...· Foundation 网络...· Foundation 块状...	<div>站点信息</div> <ul style="list-style-type: none">· 意见反馈· 免责声明· 关于我们· 文章归档
				<div>关注微信</div>

· Foundation 网

格...



Reserved. 备案号：闽ICP备
15012807号-1